# EFFICIENT COMPUTATION OF VORONOI NEIGHBORS BASED ON POLYTOPE SEARCH IN PATTERN RECOGNITION

Juan Mendez[1] and Javier Lorenzo[2]
[1]*Dept. Informatica y Sistemas, Univ. Las Palmas de Gran Canaria, Las Palmas, Spain*
[2]*IUSIANI, Univ. Las Palmas de Gran Canaria, Las Palmas, Spain*

Keywords: Pattern Recognition, Machine Learning, Nearest Neighbors, Voronoi adjacency, Linear programming.

Abstract: Some algorithms in Pattern Recognition and Machine Learning as neighborhood-based classification and dataset condensation can be improved with the use of Voronoi tessellation. The paper shows the weakness of some existing algorithms of tessellation to deal with high dimensional datasets. The use of linear programming can improve the tessellation procedures by focusing in Voronoi adjacency. It will be shown that the adjacency test based on linear programming is a version of the polytope search. However, the polytope search procedure provides more information than a simple Boolean test. The paper proposes a strategy to use the additional information contained in the basis of the linear programming algorithm to obtain other tests. The theoretical results are applied to tessellate several random datasets, and also for much-used datasets in Machine Learning repositories.

## 1 INTRODUCTION

Pattern Recognition(PR) and Machine Learning (ML) are disciplines where the knowledge about the spatial organization of the data can improve the performance of the learning and classification procedures. Voronoi and Delaunay tessellations provide partitions of some representation spaces useful in applications concerning the spatial organization of data collections. The tessellation process makes a partition of the space in disjunct regions or cells called Delaunay or Voronoi polytopes/polyhedra. Unfortunately Delaunay/Voronoi based approaches have not been very successful in PR and ML (if compared with Statistical one) because the computational complexity of these methods. When the attributes that define each instance of the dataset are defined in $R$, every instance can be represented as a point in $R^n$, where $n$ is the dimensionality of the problem. Thus, the processing of datasets with real attributes can exploit their geometrical equivalence and take advantage of many well-founded geometrical procedures.

Many PR procedures, for example Neighborhood-based Classification or Dataset Condensation, only need the adjacency relations between instances instead of full details of Voronoi or Delaunay tessellations. The Voronoi adjacency deals with the problem of checking if a pair of training instances have a common boundary, that is if both are neighbors in the Voronoi tessellation.

The Nearest Neighbor (NN) and $k$-NN are the most used algorithms in the family of neighborhood-based procedures. Voronoi based is only a category of search procedures in spaces that are coded by means data structures, as Delaunay/Voronoi or other spatial related threes (Navarro, 2002). The $k$ parameter in $k$-NN is usually chosen by means of a cross-validation process over the training samples (Duda et al., 2001). Instead of using a fix $k$ value for the whole dataset, it will be useful to define a neighborhood that locally adapts to the data without the need for cross-validation (Gupta et al., 2008; Chin et al., 2007). The *natural neighbors* for a test point $q$ can be defined from the Voronoi tessellation of the training set as the set of training instances $p_i$ whose Voronoi cell contains (or are adjacent to the cell containing) $q$. This definition follows the previously introduced by Sibson (Sibson, 1981) and Gupta *et alt* (Gupta et al., 2008). The natural neighbors are in a subset of instances that encloses or surrounds the test point.

Procedures of dataset editing, pruning or condensing are useful in ML applications where massive dataset are used to train practical classifiers, eg. SVM or Neural Networks. In such cases volumes of the training sets are drastically reduced with low or null loss in the information. The condensation pro-



Mendez J. and Lorenzo J. (2012).
EFFICIENT COMPUTATION OF VORONOI NEIGHBORS BASED ON POLYTOPE SEARCH IN PATTERN RECOGNITION.
In *Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, pages 357-364
DOI: 10.5220/0003755703570364

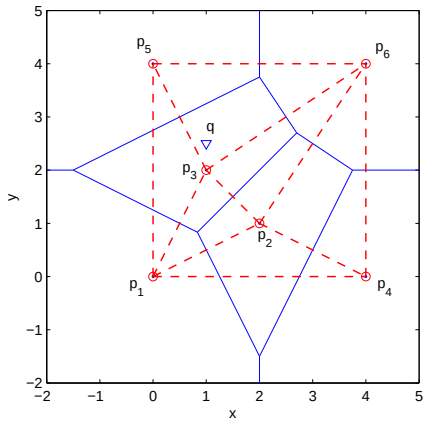


Figure 1: A simple dataset(Fukuda, 2004), $P = \{(0,0),(2,1),(1,2),(4,0),(0,4),(4,4)\}$, showing the Voronoi polyhedra as well as the Delaunay polytopes. The nearest neighbor of point $\mathbf{q}$ is $\mathbf{p}_3$ and its natural neighbors are: $\{\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_5,\mathbf{p}_6\}$.

cedures that are decision-boundary consistent (Bhattacharya et al., 1992; Duda et al., 2001) based on Voronoi adjacency do not modify the boundary between classes. Therefore, any improvement in the computation of the Voronoi tesselation will imply a reduction in the computational cost of any procedure that can be obtained from this tesselation as the *k*-NN. In this approach of using spatial information provided by Dlaunay/Vornoy methods is clustering method(Koivistoinen et al., 2006) is an agglomerative clustering algorithm which access density information by constructing a Voronoi diagram for the input samples.

The Voronoi tessellation procedure uses the metric distance to define the boundary planes between regions or cells. Metric distance, as well as vector norm, only can be used on spaces with a metric structure. However, many applications in ML deal with data collections without such a level of structured domains. One way to transform the experimental raw space in a metric space is to use the statistical Mahalanobis distance (Web, 2002). An equivalent approach is the use of an orthonormal linear transformation as performed in the Karhunen-Loewe (KL) transformation (Web, 2002). In this case, the Euclidean distance in the transformed space is equivalent to the Mahalanobis distance in the experimental space.

There are several methods to compute the Voronoi and its dual the Delaunay tessellations (Watson, 1981; Bowyer, 1981; Ramasubramanian and Paliwal, 1997). Perhaps one of the more successful approaches is the one based on representation in an extended space by mapping the instances in $R^{n+1}$, and attempting to search their convex hull. The projection of the solution in $R^n$ generates the tessellation. The greatest problem with the computation of Voronoi tessellation is the computational complexity. For a dataset with $m$ instances it is in $O(m \log m)$ for 2D cases, and for a space with dimension $n$, it is in $O(m^{n/2})$ in the general case (Gupta et al., 2008; Barber et al., 1996), which is clearly exponential with the problem dimensionality. Figure 2 shows the results of the program *qvoronoi*, a member of the *qhull* package (Barber et al., 1996), for some UCI datasets(Asuncion and Newman, 2007). It is highly efficient in computing low dimensional datasets, but can not tessellate high dimension datasets.

The computational complexity of Voronoi tessellations can be reduced with the use of Grabriel graphs(Gabriel and Sokal, 1969), which have been used as lower cost alternatives for Voronoi adjacency (Aupetit, 2003; Aupetit and Catz, 2005). However, Gabriel graphs are subsets of Voronoi graphs and do not provide the full information about neighboring relations.

Computing the Voronoi or Delaunay tessellation in higher dimensional spaces can become unpractical. However, computing only the Voronoi adjacency can be done very efficiently by using Linear Programming(LP)(Fukuda, 2004). The relationship between Voronoi and LP problems has a sound theoretical background (Agrell, 1993; Kalai, 1997; Fukuda et al., 1997; Avis and Fukuda, 1992; Bremner et al., 1997) and can be continually improved with the advances in computer hardware because Linear Programming (LP) can be efficiently programmed in matrix processors as GPUs(Greeff, 2005) and multiprocessor systems(Yarmish and van Slyke, 2001).

As it was stated above, any reduction in the computation of the Voronoi adjacency will imply an improvement in methods like the *k*-NN and condensation techniques. The aim of this paper is to present a method for an efficient computation of the Voronoi adjacency graph. This computation is based on Linear Programming and it introduces some innovations over papers previously referenced. The first one is the modification of the Voronoi adjacency test proposed by Fukuda (Fukuda, 2004) by showing that it can be reduced to the polytope search procedure. The second innovation is to show that the use of the dual problem (Winston, 1994; Bazaraa et al., 1990) of the adjacency test brings computational advantages. And last innovation, but not least, the proposal of an adjacency search strategy without backtracking. This strategy assures the computation of the correct value for all adjacency pairs without needing the computation of adjacency test for all the pairs.

The paper is structured as follows, firstly, the adjacency test for an instance pair is formulated, modified

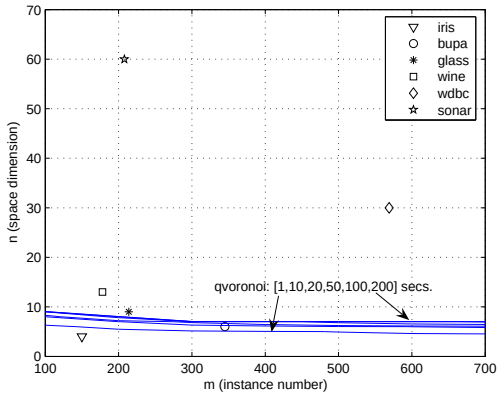


Figure 2: Efficiency area of qvoronoi and its relation with some of most used datasets in ML. It is very efficient, but only for low dimensionality problems.

and transformed to its dual form. Then, the procedure of polytope search is formulated and transformed to its dual form. It will be shown that the adjacency test is a version of the polytope search. However, the polytope search procedure provides more information than a simple Boolean test. The paper proposes a strategy to use the additional information contained in the basis of the linear programming algorithm to obtain other tests. The experiments were realized with both artificial and real datasets. Real datasets with numerical features were taken from the UCI repository to allow comparisons with the results presented in this work.

## 2 EFFICIENT COMPUTATION OF VORONOI ADJACENCIES

One way to compute the Voronoi polyhedron of a dataset $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$ in $R^n$ is based on the construction of an extended paraboloid representation in $R^{n+1}$. If $\mathbf{x} \in R^n$, the $n+1$ paraboloid coordinate is: $x_{n+1} = \sum x_1^2 + , \cdots , + x_n^2 = \|\mathbf{x}\|^2$. If $p_{ij}$ are the coordinate values of $\mathbf{p}_i$, its extended representation is: $\overline{\mathbf{p}}_i = (p_{i1}, \ldots, p_{in}, \|\mathbf{p}_i\|^2)$. The set of tangent $(n+1)$-planes in every instance of the dataset generates a polyhedron whose projection in $R^n$ is the Voronoi diagram (Fukuda, 2004). The polyhedron is defined by the following set of linear equations:

$$2\sum_{j=1}^{n} p_{ij}x_j - x_{n+1} \leq \|\mathbf{p}_i\|^2 \qquad i = 1, \ldots, m \qquad (1)$$

The adjacency of two instances $\mathbf{p}_a$ and $\mathbf{p}_b$ is verified if they have a common separating plane in $R^n$, therefore, the tangent planes in $R^{n+1}$ in each instance have an intersection and a common edge. This condition is verified if a solution exits for the following linear system:

$$\begin{array}{lcll} 2\sum_{j=1}^{n} p_{ij}x_j - x_{n+1} & \leq & \|\mathbf{p}_i\|^2 & i \neq a,b \\ 2\sum_{j=1}^{n} p_{aj}x_j - x_{n+1} & = & \|\mathbf{p}_a\|^2 & \\ 2\sum_{j=1}^{n} p_{bj}x_j - x_{n+1} & = & \|\mathbf{p}_b\|^2 & \end{array} \qquad (2)$$

This feasibility test of this linear system is related to the solution of the following problem of linear programming, where $f()$ is an objective function subject to the following constraints:

$$\begin{array}{l} \text{maximize} \quad f(x_1, \ldots, x_n, x_{n+1}) \\ 2\sum_{j=1}^{n} p_{ij}x_j - x_{n+1} \leq \|\mathbf{p}_i\|^2 i \neq a,b \\ 2\sum_{j=1}^{n} p_{aj}x_j - x_{n+1} = \|\mathbf{p}_a\|^2 \\ 2\sum_{j=1}^{n} p_{bj}x_j - x_{n+1} = \|\mathbf{p}_b\|^2 \end{array} \qquad (3)$$

This problem can be solved by introducing slack and surplus variables and using the Two-Phase Method(Winston, 1994). The feasibility of this problem is obtained in the first phase by solving the next linear problem, whose goal is the minimization of the sum of all the surplus variables:

$$\begin{array}{l} \text{minimize} \quad Z = s_a + s_b \\ 2\sum_{j=1}^{n} p_{ij}x_j - x_{n+1} + s_i = \|\mathbf{p}_i\|^2 \quad i = 1, \ldots, m \\ s_i \geq 0 \quad i = 1, \ldots, m \end{array} \qquad (4)$$

The feasibility test for the original problem of Equation (2) is that the optimal solution become null, $Z^* = 0$, equivalent to: $s_a^* = s_b^* = 0$. This problem can be modified as:

$$\begin{array}{l} \text{minimize} \quad Z' = \sum_{j=1}^{n}(p_{aj} + p_{bj})x_j - x_{n+1} \\ 2\sum_{j=1}^{n} p_{ij}x_j - x_{n+1} \leq \|\mathbf{p}_i\|^2 \quad i = 1, \ldots, m \end{array} \qquad (5)$$

Where $-Z = 2Z' - \|\mathbf{p}_a\|^2 - \|\mathbf{p}_b\|^2$, and the slack and surplus variables have been hidden. The linear programming dual of this problem is:

$$\begin{array}{l} \text{minimize} \quad Z'' = \sum_{i=1}^{m} \|\mathbf{p}_i\|^2 z_i \\ \sum_{i=1}^{m} p_{ij}z_i = \frac{1}{2}(p_{aj} + p_{bj}) \quad j = 1, \ldots, n \\ \sum_{i=1}^{m} z_i = 1 \\ z_i \geq 0 \quad i = 1, \ldots, m \end{array} \qquad (6)$$

The optimal solution of the dual must be: $Z''^* = Z'^* = \frac{1}{2}(\|\mathbf{p}_a\|^2 + \|\mathbf{p}_b\|^2)$.

### 2.1 Polytope Search

Voronoi polyhedra can be unbound, but a bounded polyhedron is called a polytope. Delaunay polytopes are the dual of Voronoi polyhedra. The test

for Voronoi adjacency, as defined in Equation (6), is related to the problem of the polytope search. This problem is related to find the Delaunay polytope that encloses a test point $\mathbf{q} \in R^n$: more precisely, in obtaining the subset of the dataset instances which define the polytope enclosing the test point. The polytope degree ranges from 1 to $n+1$ depending on the number of instances included, or the degree of degeneracy of the polytope. Unfortunately, not all the polytopes found are of the biggest degree of $(n+1)$, however, a lower degree provides valuable information because a $k$-polytope includes $k(k-1)/2$ Voronoi adjacencies. If we are trying to find the enclosing polytope of a point $\mathbf{q}$, the problem can be solved by using linear programming and finding the solution for $y_0 \in R$ and $\mathbf{y} \in R^n$ verifying(Fukuda, 2004):

$$\begin{array}{ll} \text{minimize} & Z = y_0 + \sum_{j=1}^{n} q_j y_j \\ & -y_0 - \sum_{j=1}^{n} p_{ij} y_j \leq \|\mathbf{p}_i\|^2 \quad i = 1,\ldots,m \end{array} \tag{7}$$

The Delaunay polytope containing the test point is the one whose corresponding inequalities are satisfied as equality when the problem reaches optimal. That is, whose dual variables are not null. This linear programming algorithm has two different stop states. In the first, the enclosing polytope is found if the problem reaches the optimality. In the second one, the problem gets unbound and no solution is provided because the test point is outside the convex hull of the dataset instances. If the solution is optimal but degenerate, a $k$-polytope is obtained with $1 \leq k \leq n+1$. The enclosing polytope can be obtained easily by solving the dual of the Equation (7):

$$\begin{array}{ll} \text{minimize} & W = \sum_{i=1}^{m} \|\mathbf{p}_i\|^2 z_i \\ & \sum_{i=1}^{m} \mathbf{p}_i z_i = \mathbf{q} \\ & \sum_{i=1}^{m} z_i = 1 \\ & z_i \geq 0 \quad i = 1,\ldots,m \end{array} \tag{8}$$

If the test point is outside the convex hull, the problem in Equation (7) becomes unbound, while its dual in Equation (8) becomes unfeasible. If the problem is not degenerate, the number of non-null problem variables $\mathbf{z} \in R^m$ is $n+1$ that define the enclosing Delaunay polytope. If the problem is degenerate the number of non-null variables is lower. However the number of problem variables in the final basis provides some additional information: if a problem variable is null but is included in the final basis, we can infer that the $k$-polytope is a subset of a $(k+1)$-polytope defined by $k$ non-null variables and this null one that also is included in the basis. Therefore, knowledge of the final basis provides extra information in cases of degeneracy.

For computational purposes the polytope search procedure can be express as: $\mathbf{B} \leftarrow POLYTOPE(\mathbf{P}, \mathbf{q})$. Where $\mathbf{B}$ is the set of instances in $P$ included in the basis of the linear programming. The scalar $K = \text{card}(\mathbf{B})$ is an upper bound of the polytope degree, it verifies: $1 \leq k \leq K \leq n+1$. When the test point is outside the convex hull, we get $K = 0$ for computational purpose. The Voronoi adjacency test for two instances in a dataset in Equation (6) corresponds to the polytope search procedure in Equation (8) by testing the middle point between the pair: $\mathbf{q} = \frac{1}{2}(\mathbf{p}_a + \mathbf{p}_b)$. This test point is always within the convex hull, therefore the unfeasible solution is not possible.

A Voronoi adjacency graph is constructed by taking each dataset instance as a node and the Boolean link $v_{ij} \in \{0,1\}$ as the value of the adjacency test between instances $\mathbf{p}_i$ and $\mathbf{p}_j$. The test for every middle point assures knowledge of the $v_{ij}$ value, but in every test also other $v_{hl}$ link values are also obtained depending on the cardinality of $\mathbf{B}$. In the best case, a number of: $n(n+1)/2+1$ links of the Voronoi adjacency graph are obtained. In the worst case only a link value is obtained: it occurs when the middle point of two instances $\mathbf{p}_i$ and $\mathbf{p}_j$ is just another instance of the dataset $\mathbf{q} = \mathbf{p}_h$. The best case happens when the middle point is within a Delaunay polytope that does not include the test pair. In this case $K = n+1$, therefore, $n(n+1)/2$ links with true values are obtained as well as a false one: $v_{ij} = 0$. The Algorithm 1 shows the proposed procedure. It initialize all the links to false values and only positive adjacencies are added throughout the following steps. Table 1 contains a trace of the computed pairs for dataset in Figure1, where the values for the basis variables are shown.

## 2.2 Gabriel Adjacency

Gabriel adjacency is a subset of Voronoi adjacency, its definition resembles the general definition of Delaunay polytope. A set of $(n+1)$ instances defines a Delaunay polytope if the $n$-sphere that they describe has no instance into. While, two instances are Gabriel neighbors (Devroye et al., 1996) if no other instance is included in the $n$-sphere that is centered in the middle point: $\frac{1}{2}(\mathbf{p}_a + \mathbf{p}_b)$ and has a radius: $\frac{1}{2}\|\mathbf{p}_a - \mathbf{p}_b\|$, that is:

$$\|\mathbf{p}_i - \frac{1}{2}(\mathbf{p}_a + \mathbf{p}_b)\| \geq \frac{1}{2}\|\mathbf{p}_a - \mathbf{p}_b\| \qquad \forall i \neq a,b \tag{9}$$

Based on: $\|\mathbf{u} - \mathbf{v}\|^2 = \|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 - 2\mathbf{u}\cdot\mathbf{v}$, it can be simplified as:

$$\mathbf{p}_i \cdot \mathbf{p}_i - \mathbf{p}_i \cdot \mathbf{p}_a - \mathbf{p}_i \cdot \mathbf{p}_b + \mathbf{p}_a \cdot \mathbf{p}_b \geq 0 \quad \forall i \neq a,b \tag{10}$$

The Delaunay test, which involves $n+1$ instances to define the sphere, is more expensive than the

**Algorithm 1:** Computes the Voronoi adjacency graph. The input is the dataset instances, $\mathbf{P} = \mathbf{p}_1, \ldots, \mathbf{p}_m$, and the output the graph, $\mathbf{V} = \{v_{ij}\}$.

**procedure** ADJACENCY1($\mathbf{P} = \mathbf{p}_1, \ldots, \mathbf{p}_m, \mathbf{V} = \{v_{ij}\}$)
  Initialize: $\forall_{ij}, v_{ij} \leftarrow 0$
  **for** $i \leftarrow 1, m-1$ **do**
    **for** $j \leftarrow i+1, m$ **do** ▷ only the upper triangular
      **if** $v_{ij} = 0$ **then**
        $\mathbf{q} \leftarrow \frac{1}{2}(\mathbf{p}_i + \mathbf{p}_j)$ ▷ the middle point between $\mathbf{p}_i$ and $\mathbf{p}_j$
        $\mathbf{B} \leftarrow$POLYTOPE($\mathbf{P}$,$\mathbf{q}$) ▷ gets the basis of the polytope
        $K \leftarrow \text{card}(\mathbf{B})$ ▷ by solving Equation (8)
        **if** $K \geq 2$ **then** ▷ if degeneracy: $1 \leq K \leq n+1$
          **for** $h \leftarrow 1, K-1$ **do**
            $c \leftarrow B_h$
            **for** $l \leftarrow h+1, K$ **do**
              $d \leftarrow B_l$
              $v_{cd} \leftarrow 1$ ▷ $\mathbf{p}_c$ and $\mathbf{p}_d$ are Voronoi neighbors
              $v_{dc} \leftarrow 1$ ▷ and the symmetrical

Gabriel adjacency, which uses two instance to define a smaller sphere. If an instance pair verifies the Gabriel test, it also verifies the Voronoi neighbor test, but not the converse. This property can be used to introduce a cheaper but incomplete pre-test of adjacency.

The Gabriel test is advantageous if compared to the general Delaunay test, but this advantage is unclear when compared with Voronoi adjacency obtained with LP, because it provides only a link value in every test. To increase the performance of the polytope-based adjacency test, one algorithm is proposed that use the cheaper Gabriel test. The Boolean procedure GABRIEL($\mathbf{P}$, $i$, $j$) is used to test for the adjacency of the instances $\mathbf{p}_i$ and $\mathbf{p}_j$. In the Algorithm 2, prior to the polytope test, a pre-test is included for every instance. If the first test fails, the second one computes the pairs values.

## 3 RESULTS

A systematic test have been performed to show the strength and weakness of existing tessellation procedures. One of the more used and faster packages is the before mentioned qhull(Barber et al., 1996). To solve several problems in computational geometry in $R^n$, it uses the computation of the convex hull in $R^{n+1}$ as the kernel procedure. The family of programs based on qhull are very fast for problems with low dimensionality. However, they suffer the curse of dimensionality when applied in high dimension problems such as those used in ML.

The test uses several random datasets whose instances are within the unit cube centered at the origin. The dimension used are: $n = 2, 3, \ldots, 9, 10, 20, \ldots, 70$ and te number of instances is: $m = 100, 200, \ldots, 700$. The computation time were taken from the program *qvoronoi*, a member of the qhull package. To illustrate the obtained results for these datasets a contour plot was generated as shown in Figure 2. Some values from 1 to 200 seconds are plot to illustrate the efficiency area of the procedure efficiency. The number of dimensions and instances of some of the most used dataset in the UCI Machine Learning Repository(Asuncion and Newman, 2007) as *iris*, *bupa*, *glass*, *wine*, *wdbc* and *sonar* are also plotted. The Figure shows that while *bupa* dataset (345 instances and dimension 6) can be effectively tessellated in 5.76 sec. on a test computer (Intel Pentium M, 1.6 Ghz and 1 GB of RAM), the *glass* dataset (214 instances and dimension 9) took several hours to complete. A practical conclusion were obtained, namely, that datasets with a dimension greater than eight *can not* be tessellated effectively with this procedure.

The Algorithm 1 defines how to compute the Voronoi adjacencies, which are coded as a graph. The Boolean links $v_{ij} \in \{0,1\}$ are symmetric: $v_{ij} = v_{ji}$, therefore, only the upper triangular is computed. The Algorithm was implemented in C++ using double precision real numbers. The same systematic test that had been conducted for the qvoronoi was performed for the implementation of the Algorithm 1. The contour plot of the efficiency is shown in Figure 3. The entire range of the UCI datasets is covered in the range of 200 seconds in this test computer. The efficiency area seems to cover a more extended area in the $n$ vs. $m$ plane, which allows to cover a wide-range of practical ML applications. In low dimension datasets, qhull significatively outperforms the proposed implementation, but for $n \geq 8$, it is outperformed.

A performance factor is defined about how many middle point tests are required to obtain all the adjacency links of a dataset. The factor is defined as: $\gamma = 2N_{test}/m(m-1)$, where $N_{test}$ is the number of

Table 1: Computation of Voronoi neighbors for the dataset in Figure 1. The Algorithm 1 takes 8 test to compute the 15 adjacent pairs, and used $N_s = 24$ iterations of Simplex Dual algorithm. The $k$ parameter is the polytope degree, whereas $K$ is the number of $\mathbf{z}$ variables in the basis. The filled-in $\mathbf{z}$ variables are those in the basis.

| pair | $\mathbf{q}$ | | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | $z_6$ | $k$ | $K$ | links | $N_s$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1,2 | 1.0 | 0.5 | 0.50 | 0.50 | | | | | 2 | 2 | $v_{12}$ | 2 |
| 1,3 | 0.5 | 1.0 | 0.50 | | 0.50 | | | | 2 | 2 | $v_{13}$ | 2 |
| 1,4 | 2.0 | 0.0 | 0.50 | 0.00 | | 0.50 | | | 2 | 3 | $v_{12}, v_{14}, v_{24}$ | 3 |
| 1,5 | 0.0 | 2.0 | 0.50 | | 0.00 | | 0.50 | | 2 | 3 | $v_{13}, v_{15}, v_{35}$ | 3 |
| 1,6 | 2.0 | 2.0 | | 0.40 | 0.40 | | | 0.20 | 3 | 3 | $v_{23}, v_{26}, v_{36}$ | 3 |
| 2,5 | 1.0 | 2.5 | | | 0.75 | | 0.19 | 0.06 | 3 | 3 | $v_{35}, v_{36}, v_{56}$ | 4 |
| 3,4 | 2.5 | 1.0 | | 0.75 | | 0.19 | | 0.06 | 3 | 3 | $v_{24}, v_{26}, v_{46}$ | 4 |
| 4,5 | 2.0 | 2.0 | | 0.40 | 0.40 | | | 0.20 | 3 | 3 | $v_{26}, v_{23}, v_{36}$ | 3 |

**Algorithm 2:** A modification of Algorithm 1 that computes all the Grabriel pre-tests previously to the polytope ones.

**procedure** ADJACENCY2($\mathbf{P} = \mathbf{p}_1, \ldots, \mathbf{p}_m$,$\mathbf{V} = \{v_{ij}\}$)
  Initialize: $\forall_{ij}, v_{ij} \leftarrow 0$
  **for** $i \leftarrow 1, m-1$ **do**
    **for** $j \leftarrow i+1, m$ **do**
      **if** GABRIEL($\mathbf{P}$, $i$, $j$) **then** ▷ tries Gabriel adjacency
        $v_{ij} = 1$
        $v_{ji} = 1$
  **for** $i \leftarrow 1, m-1$ **do**
    **for** $j \leftarrow i+1, m$ **do** ▷ only the upper triangular
      **if** $v_{ij} = 0$ **then**
        $\mathbf{q} \leftarrow \frac{1}{2}(\mathbf{p}_i + \mathbf{p}_j)$ ▷ the middle point between $\mathbf{p}_i$ and $\mathbf{p}_j$
        $\mathbf{B} \leftarrow$POLYTOPE($\mathbf{P}$,$\mathbf{q}$) ▷ gets the basis of the polytope
        $K \leftarrow$ card($\mathbf{B}$) ▷ by solving Equation (8)
        **if** $K \geq 2$ **then** ▷ if degeneracy: $1 \leq K \leq n+1$
          **for** $h \leftarrow 1, K-1$ **do**
            $c \leftarrow B_h$
            **for** $l \leftarrow h+1, K$ **do**
              $d \leftarrow B_l$
              $v_{cd} \leftarrow 1$ ▷ $\mathbf{p}_c$ and $\mathbf{p}_d$ are Voronoi neighbors
              $v_{dc} \leftarrow 1$ ▷ and the symmetrical

Table 2: Tessellation results for several datasets coded in normalized coordinates after the KL transformation: $T_q$ the computational time used by qvoronoi, $T_1$ the used by the proposed Algorithm 1, $N_{test}$ the number of middle points tested, $N_s$ the number of Simplex iterations used to tessellate the whole dataset, and $\gamma$ the performance factor. The $T_2$ column contains the computational time for the Algoritm 2

| Dataset | n | m | $T_q$(sec.) | $T_1$(sec.) | $N_{test}$ | $N_s$ | $\gamma$ | $T_2(sec.)$ |
|---|---|---|---|---|---|---|---|---|
| iris | 4 | 150 | 0.04 | 0.503 | 9349 | 96844 | 0.837 | 0.515 |
| bupa | 6 | 345 | 6.71 | 10.136 | 45891 | 783100 | 0.773 | 10.162 |
| glass | 9 | 214 | n/a | 2.708 | 9262 | 256613 | 0.406 | 2.733 |
| wine | 13 | 178 | n/a | 0.863 | 3082 | 69167 | 0.196 | 0.898 |
| wdbc | 30 | 569 | n/a | 58.886 | 18109 | 664385 | 0.112 | 61.358 |
| sonar | 60 | 208 | n/a | 6.003 | 5237 | 46555 | 0.243 | 6.752 |

tested pairs necessary to achieved the computation of all Voronoi adjacencies. It depends of each dataset, and in general would have a general dependence on $m$ an $n$. Low factor values are equivalent to high tessellation performance, because its inverse provides the average number of adjacency relations obtained for each test. The tessellation cost not only depends on the size of dataset $m$ and $n$, it also depends on the distribution of instances. The Table 2 contains the computation time for the ML Repository datasets, as well as the $N_{test}$ and the Simplex iterations used. Normalized coordinates are used after the KL transformation because raw data coordinates are meaningless when used in a metric distance.

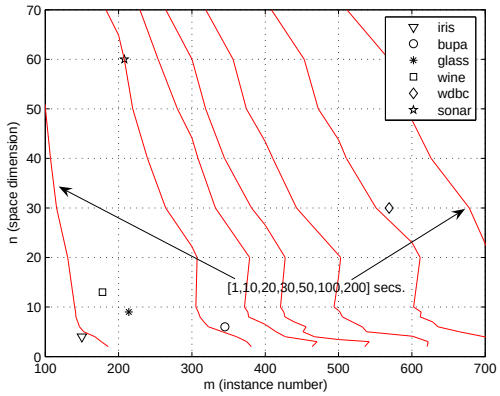


Figure 3: Efficiency area of the tessellation procedure based on polytope search. The covered area is more extensive that the covered by qhull and includes the datasets. However, for low dimensional problems qhull significatively outperforms it.

It should be mentioned that the plotted points of each of the ML Repository dataset in the Figure 3 are only qualitative because the plotted background data are related to random datasets. No qvoronoi values are available for glass dataset and larger, because these tests had not finished after several hours of computation. Therefore, they are not comparable for practical purposes. These data are computed on an Intel Xeon, 3.06 Ghz, 512K in L2 cache and 1.5GB of RAM. The last column contains the $T_2$ computed for the Algorithm 2, only slight differences are detected between the two Algorithms.

The cost analysis of proposed procedure depends on the analysis cost of the LP problem for finding a Polytope enclosing a point. The cost to obtain all the Voronoi adjacencies $C_{AllAd}(n,m)$ is:

$$C_{AllAd}(n,m) = \gamma(n,m)\frac{m(m-1)}{2}C_{Poly}(n,m) \quad (11)$$

Where $\gamma(n,m)$ is the fraction of the $m(m-1)/2$ pairs that that are tested. It runs, in the considered cases of the dataset in UCI, from 0.837 for iris dataset to 0.112 for wdbc dataset. In general it would have a general dependence on $m$ and $n$ that future works could clarify. We think that it depend for every specific dataset, and that a general dependence as $\gamma(n,m)$ is only valid as an average for random datasets.

The cost to obtain a polytope, $C_{Poly}(n,m)$, is the cost to solve a LP problem. Although we have used in practice the Simplex Dual Algorithm for practical proposes, any of the available LP Algorithms can be used. This algorithm choice is no central of our proposal; such as future works will test the relative efficiency of other choices (Simplex based variants as well as interior methods). A founded opinion (Dantzig and Thapa, 2003) is that the efficiency of good implementations of simplex-based methods and interior point methods are similar for practical applications of linear programming. However, for specific types of LP problems, it may be that one type of algorithm is better than another, but it cannot be decided without an exhaustive test.

Is very difficult to define the theoretical cost of a LP Algorithm because we have to decide between the cost for *worst-case* and the cost for *average-case* in the defined application. Although the worst-case complexity of the Simplex Algorithm is exponential in the problem dimension, it was widely known that in practice it is probably a polynomial-time (Wright, 2004), that is in practice the Simplex method almost always converges on real-world problems in a number of iterations that is polynomial in the problem dimension.

If the average cost of LP problem for polytope finding is on the class of $O(f(m)g(n)$, where $f(m)$ and $g(n)$ are polynomial, so we can conclude that the practical cost $C_{AllAdj}(n,m)$ falls in the class $O(m^2 f(m)g(n))$ also polinomial. That is very advantageous to qhull based approaches (which are in exponential $O(m^{n/2})$ class) for large values of the space dimensionality $n$, but unadvantageous for small ones.

## 4 CONCLUSIONS

Machine Learning applications impose unattainable goals on traditional tessellation techniques, while linear programming provides alternative approaches to perform the tessellation of high dimensional datasets. Linear programming provides a sound theoretical background for the tessellation problem as well as an inspirational source for efficient implementations. A modification of the Voronoi adjacency test had shown that it is basically the polytope search procedure, enabling the implementation of a more efficient algorithm for high dimensional datasets. It is more efficient that a *single* adjacency test because in each trial it provides a polytope, that is *many* adjacency values. These perform best if focusing on the$\gamma$ parameter, which is related to the fraction of all the all-to-all needed test. The reason for this is that, the higher dimensionality the greater the number of instances included in each polytope. This is the counterpart of the curse of the dimensionality. Perhaps this would be the reason why it permits a relative good performance at high dimensionality. The qhull-based and the linear programming-based implementations are complementary because each is good in different domains. A suitable use of both algorithms can

efficiently tessellate many massive datasets in Machine Learning. The use of a pre-test based on the Gabriel adjacency, which provides a faster but incomplete graph of neighboring relations, does not significatively increase performance because, while it is fast, it provides only one link value while the polytope provides several link values in each test.

## REFERENCES


Agrell, E. (1993). A method for examining vector quantizer structures. In *Proceeding of IEEE International Symposium on Information Theory*, page 394.

Asuncion, A. and Newman, D. (2007). UCI machine learning repository.

Aupetit, M. (2003). High-dimensional labeled data analysis with gabriel graphs. In *European Symposium on Artificial Neuron Networks*, pages 21–26.

Aupetit, M. and Catz, T. (2005). High-dimensional labeled data analysis with topology representating graphs. *Neurocomputing*, 63:139–169.

Avis, D. and Fukuda, K. (1992). A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Comput. Geom.*, 8(3):295–313.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483.

Bazaraa, M. S., Jarvis, J. J., and Sherali, H. S. (1990). *Linear Programming and Networks Flows*. Wiley.

Bhattacharya, B., Poulsen, R., and Toussaint, G. (1992). Application of proximity graphs to editing nearest neighbor decision rules. Technical Report SOCS 92.19, School of Computer Science, McGill University.

Bowyer, A. (1981). Computing Dirichlet tessellations. *The Computer Journal*, 24(2):162–166.

Bremner, D., Fukuda, K., and Marzetta, A. (1997). Primal-dual methods for vertex and facet enumeration. In *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*, pages 49–56, New York, NY, USA. ACM.

Chin, E., Garcia, E. K., and Gupta, M. R. (2007). Color management of printers by regression over enclosing neighborhoods. In *IEEE International Conference on Image Processing. ICIP 2007*, volume 2, pages 161–164.

Dantzig, G. B. and Thapa, M. N. (2003). *Linear Programming 2: Theory and Extensions*. Springer Verlag.

Devroye, L., Gyorfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*.

Duda, R., Hart, P., and Stork, D. (2001). *Pattern Classification*. John Wiley.

Fukuda, K. (2004). Frecuently asked questions in polyhedral computation. Technical report, Swiss Federal Institute of Technology, Lausanne, Switzerland.

Fukuda, K., Liebling, T. M., and Margot, F. (1997). Analysis of backtrak algoritms for listing all vertices and all faces of convex polyhedron. *Computational Geometry*, 8:1–12.

Gabriel, K. R. and Sokal, R. R. (1969). A new stattistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–270.

Greeff, G. (2005). The revised simplex algorithm on a GPU. Technical report, Dept. of Computer Science, University of Stellenbosch.

Gupta, M. R., Garcia, E. K., and Chin, E. (2008). Adaptive local linear regression with application to printer color management. *IEEE Trans. on Image Processing*.

Kalai, G. (1997). Linear programming, the simplex algorithm and simple polytopes. *Math. Program.*, 79:217–233.

Koivistoinen, H., Ruuska, M., and Elomaa, T. (2006). A voronoi diagram approach to autonomous clustering. *Lecture Notes in Computer Science*, (4265):149–160.

Navarro, G. (2002). Searching in metric spaces by spatial approximation. *The VLDB Journal*, 11:28–46.

Ramasubramanian, V. and Paliwal, K. (1997). Voronoi projection-based fast nearest-neighbor search algorithms: Box-search and mapping table-based search techniques. *Digital Signal Processing*, 7:260–277.

Sibson, R. (1981). *Interpreting multivariate data*, chapter A brief description of natural neighbour interpolation, pages 21–36. John Wiley.

Watson, D. F. (1981). Computing the n-dimensional tessellation with application to voronoi polytopes. *The Computer Journal*, 24(2):167–172.

Web, A. (2002). *Statistical Pattern Recognition*. John Wiley, 2nd edition.

Winston, W. L. (1994). *Operations Research Applications and Algorithms*. Wadsworth.

Wright, M. H. (2004). The interior-point revolution in optimization: History, recent developments,nnd lasting consequences. *Bull. of AMS*, 42(1):39–56.

Yarmish, G. and van Slyke, R. (2001). retroLP, an implementation of the standard Simplex method. Technical report, Dept. of Computer and Information Science, Brooklyn College.