Computational Geometry: Theory and Applications 3 (1993) 289–305
Elsevier

The complexity of finding minimal Voronoi covers with applications to machine learning*

David Heath and Simon Kasif

Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21218, USA

Communicated by Ronald Graham
Submitted 6 August 1992
Accepted 19 February 1993

Abstract

Our goal in this paper is to examine the application of Voronoi diagrams, a fundamental concept of computational geometry, to the nearest neighbor algorithm used in machine learning. We consider the question "Given a planar polygonal tessellation T and an integer k, is there a set of k points whose Voronoi diagram contains every edge in T?" We show that this question is NP-hard. We encountered this problem while studying a learning model in which we seek the minimum sized set of training examples needed to teach a given geometric concept to a nearest neighbor learning program. That is, given a concept which can be described by a planar tessellation, we are seeking to construct the smallest set of points whose Voronoi diagram is consistent with the given tessellation. In a sense, this question captures the difficulty of teaching the nearest neighbor algorithm a simple structure, using a minimal number of examples.

In addition, we consider the natural inverse to the problem of computing Voronoi diagrams. Given a planar polygonal tessellation T, we describe an algorithm to find a set of points whose Voronoi diagram is T, if such a set exists.

1. Introduction

Computing the Voronoi Diagram of a set of points is a well-known problem in the field of computational geometry, with applications in many fields, including that of machine learning. The nearest neighbor algorithm as typically used in machine learning stores a collection of examples and their respective classifications. Given a new example for which the classification is unknown, the

Correspondence to: David Heath, Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21218, USA.

* This research supported in part by Air Force Office of Scientific Research under Grant AFOSR-89-1151, National Science Foundation under Grant IRI-88-09324 and NSF/DARPA under Grant CCR-8908092.

nearest neighbor algorithm uses the classification of the nearest point to classify the unknown example. This method therefore implicitly creates a Voronoi paritioning of the space of examples which is used during classification.

In a previous paper, we introduced the helpful teacher model (see Salzberg et al. [6]), in which we find the smallest set of training examples which teaches a given concept to a learning program. Given a concept which can be described by a planar tessellation, we would like to compute the smallest set of examples which teaches the concept to the nearest neighbor learning algorithm. In that paper, we describe several techniques for teaching machine learning algorithms geometric concepts using a small set of examples. Specifically, we describe a technique for teaching the standard nearest neighbor algorithm a concept in the form of a polygonal tesselation, given a permissible error area. We left open the question of the difficulty of teaching nearest neighbor such a concept when no error can be tolerated.

This suggests the following question "Given a planar polygonal tessellation T and an integer k, is there a set of k points whose Voronoi diagram contains every edge in T?" In a sense, this question captures the difficulty of teaching the nearest neighbor algorithm a simple structure, using a minimal number of examples. We provide a non-trivial reduction that demonstrates that the complexity of this question is NP-hard.

We also consider the related question of finding an inverse of the Voronoi Diagram. We present an algorithm for determining if a planar tessellation is a Voronoi Diagram. The problem of recognizing Voronoi Diagrams has some history in Computational Geometry. A simplified version of this problem appeared as an exercise by Preparata and Shamos [5]. Ash and Bolker [1] were able to solve the problem of recognizing Voronoi Diagrams in the plane when all vertices of the tessellation have odd valence. Suzuki and Iri [7] present an algorithm for finding Voronoi Diagrams that approximate a given tessellation, but these approximations may not be minimal in size.

2. Problem definition

Given a set of points P in the plane, a *Voronoi diagram* is a partition of the plane into a set of $|P|$ polygonal regions, such that any point in one of these regions is closer to the unique member of P in the region than it is to any other point in P. See Preparata and Shamos [5] for an introduction to the Voronoi diagram.

Given a polygonal tesselation T of the plane, we say that polygonal tessellation V is a *finer* tessellation if every region in T is partitioned into one or more regions of V. This implies that every (possibly unbounded) line segment which forms part of the boundary of T is covered by a union of line segments in V. See Fig. 1 for an example of two tessellations, one of which is finer than the other. Note that

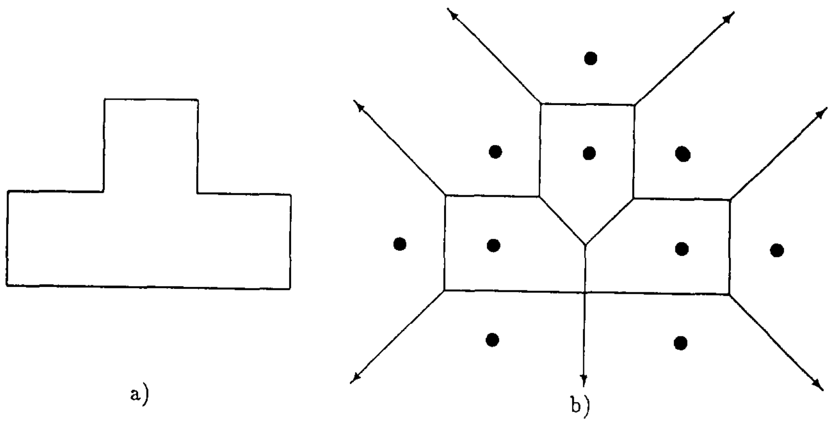

Fig. 1. a) Tessellation, b) An example finer tessellation.

the interior region of the *coarser* tessellation is partitioned into three regions in the finer, and the outer region is partitioned into seven regions. The bottom horizontal segment of the coarser tessellation is covered by the two bottom horizontal segments in the finer tessellation.

We are concerned with finding a set P of points in the plane whose Voronoi diagram is a finer tessellation than a given tessellation T. We call such a set of points a *Voronoi cover* of T. We refer to the cardinality of P as the *size* of the Voronoi cover. If a Voronoi Cover contains n points, we say it is an n-Voronoi Cover. Note that Fig. 1 is an example of a Voronoi cover of size ten.

We consider the question "Given a planar polygonal tessellation T and an integer k, is there a Voronoi cover of T that contains no more than k points?" We refer to this question as the *Voronoi cover problem*. We show that the Voronoi cover problem is NP-hard. This problem is a natural inverse of the Voronoi diagram, a well-known computational geometry construct.

3. Minimal Voronoi covers

To prove that the Voronoi cover problem is NP-hard, we will reduce a known NP-hard problem, Planar 3-SAT (Lichtenstein [3]) to it.

An instance of Planar 3-SAT is described as follows: We are given a set of boolean variables $u = \{u_1, u_2, \ldots, u_n\}$. Each variable can take one of two possible values: *true* or *false*. A clause is a disjunction of (possibly negated) variables that has value true when any of its constituents are true and has value false otherwise. We are also given a boolean expression over the variables that consists of a conjunction of clauses. The conjunction is true if and only if every one of the clauses that composes it is true. The question "Is there an assignment to the variables U, such that the given conjunction of clauses is true?" is known as *satisfiability,* and the assignment is known as a *satisfying assignment.* In the variation 3-SAT, each clause is restricted to contain no more than three variables. Planar 3-SAT is a further restriction of the problem in which it is possible to construct a planar bipartite graph which has a vertex for every variable and for every clause. A vertex representing a variable is connected to a vertex representing a clause if the variable appears (possibly negated) in the clause. See Lichtenstein [3] for a proof that this problem is NP-complete.

In the next sections we will show how to take an instance of Planar 3-SAT and convert it in polynomial time to an instance of the Voronoi cover problem such that there is a solution to the Planar 3-SAT problem if and only if there is a solution to the Voronoi cover problem. We will create a tessellation composed of three major parts. Each variable $\{u_1, u_2, \ldots, u_n\}$ will be represented by one type of structure, each clause will be represented by another. Finally, the edges in the implicit plane graph defined above will be represented by a third type of structure. Note that in a strict definition of the Planar 3-SAT problem, we are not given the planar embedding of the implicit graph, but we can find such an embedding in polynomial time (see, e.g., Hopcroft and Tarjan [2]). We will assume that each clause contains exactly three literals, whereas the definition above allows for there to be fewer. We can get around this problem by duplicating literals if necessary.

4. Construction

What follows is an intuitive overview of the construction. For every variable u_i, we will create a *variable gadget.* The variable gadget is a small set of adjacent polygons. A minimal Voronoi cover for any one gadget takes one of two forms. Which form it takes will be used to store the value (true or false) of the associated variable. In a solution to the Voronoi cover problem, the form of the variable gadgets directly gives us a satisfying assignment to the Planar 3-SAT problem.

For each clause, we create a *clause gadget.* The size of the minimal Voronoi cover for a clause gadget depends on the truth value of the associated clause. If any of the three literals in the clause gadget are true, then the minimal Voronoi cover for the gadget requires one fewer point than if none of the three literals are true. This is the crucial part of the reduction. We can choose a k such that there is a Voronoi cover of size k for the entire construction if and only if we are able to 'save' one point in every clause gadget, i.e., that each clause is true.

Finally, we must have a way of communicating the truth value of each variable to the clauses the variable appears in. This corresponds to the edges in the plane graph we can construct from the planar 3-SAT problem we are given. Each such edge will be transformed to a path of *connector gadgets* which connect the variable gadgets to the clause gadgets. Note that the connector paths do not cross.

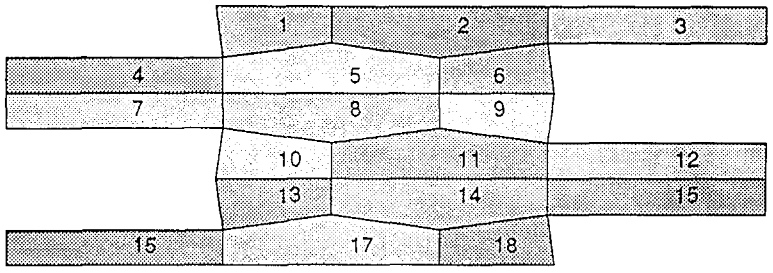

Fig. 2. An example variable gadget.

4.1. Variable gadget

An example variable gadget is shown in Fig. 2. This particular gadget has three complemented outputs and three non-complemented outputs, although it can be made with more. It is composed of eighteen bounded regions, each of which has either four or five sides. The dimensions of the regions are chosen so that a minimal Voronoi cover of the gadget has the following properties:

(1) Each quadrilateral will be covered by exactly one Voronoi region. That is, only one point will be placed in each quadrilateral by the Voronoi cover.

(2) Half of the pentagon regions will contain one point of the Voronoi cover; the other half will each contain two points.

(3) A minimal Voronoi cover for a three-output variable gadget will contain exactly forty points. Similar bounds can be obtained for gadgets with more outputs.

By definition, a Voronoi diagram contains exactly one point in each region. Each region in the variable gadget will contain at least one Voronoi region. This immediately tells us that a minimal Voronoi cover for a variable gadget will have at least eighteen points.

To define the boundary edges of the variable gadget, a Voronoi cover must also contain some points that fall outside the gadget. Consider a particular boundary edge. It defines two regions, one bounded and one unbounded (the exterior of the gadget). Every boundary edge must be the perpendicular bisector of a line segment between a point inside the bounded region and a point outside the gadget. Some polygons require that each boundary edge be defined by a different exterior point. In other tessellations, including the variable gadget, it is possible for some pairs of edges to be defined by the same exterior point (but different points in the interior of the gadget).

Consider two adjacent boundary edges. If the exterior angle at the intersection of the two edges is at least 180°, then these two edges cannot share an exterior point. This also applies to non-adjacent edges by extending the edge segments to their intersection point. We can use this fact to show that any Voronoi cover for the variable gadget must have at least eighteen exterior points. First note that there are 24 exterior edges to the tessellation. The exterior of the tessellation has

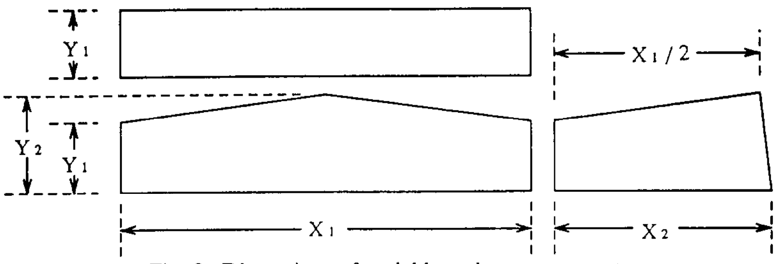

Fig. 3. Dimensions of variable gadget components.

six concave angles, or six pairs of adjacent edges which could possibly share an edge in a Voronoi cover. The six concave angles occur at the adjacent edges of the following pairs of regions in Fig. 2: (1, 4), (3, 6), (9, 12), (15, 18), (13, 16), (7, 10).

We next must consider whether any non-adjacent pair of edges could share an exterior point in a Voronoi cover. For this, we will need to consider the dimensions of the gadget. See Fig. 3. We only need consider the following pairs of regions: (3, 9), (3, 12), (6, 12), (7, 13), (7, 16), (13, 16). Each of these pairs involves a rectangular region of height Y_1. In the cases involving two such rectangles (e.g. (3, 12)), the distance between the rectangles is $2Y_2 > 2Y_1$. A shared point would have to be within distance Y_1 of each rectangle. This is clearly not possible.

Consider the cases involving a rectangle and a non-rectangular quadrilateral. Due to the angle of the quadrilateral edge, the shared point must be further than Y_2 from the rectangle. Once again, this is not possible. By this argument, a Voronoi cover of the variable gadget must contain at least eighteen external points, for a total of thirty-six points.

As we indicated above, at least half of the pentagons in the gadget must contain two points. We will consider a pair of pentagons that share a diagonal edge. See Fig. 4. By way of contradiction, assume it is possible for a Voronoi cover of this figure to have only one point in regions B and C. The line segment between these two points must have the common edge shared by regions B and C as its perpendicular bisector. (We say that one point is a *reflection* of the other through the (B, C) common edge). Region A must also contain at least one point, such that the line segment between it and the point in C is the perpendicular bisector of the (A, C) common edge. A similar point must exist in region D. Any point in B which is a reflection of both a point in C and a point in D must lie in the shaded portion of B. The same holds for a region C. Note that

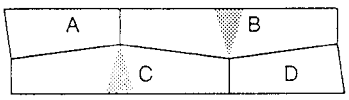

Fig. 4. Two adjacent pentagons in the variable gadget.

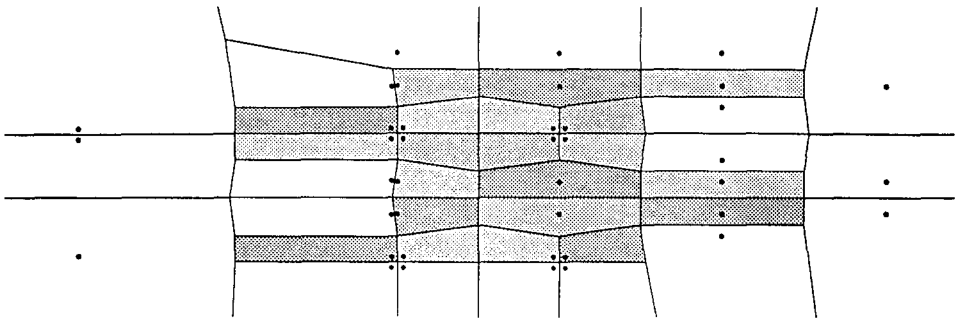

Fig. 5. Voronoi cover for variable gadget in true case.

when the (B, C) common edge has sufficiently small slope, that it is not possible to place a point in each shaded region such that they are reflections of each other. Thus, at least one of $\{B, C\}$ must contain two points in any Voronoi cover. Extrapolating to the complete variable gadget, it is clear that the best we can hope for is for every other pentagon to contain one point, while the rest contain two. Thus, we can raise the lower bound on the number of points in a minimal Voronoi cover of the variable gadget to thirty-nine.

Finally, note that two of the pentagons are on the border of the gadget, and that at least one of these must contain two points in a Voronoi cover that satisfies the above criteria. Both points in this region must have a reflection outside the gadget. This means that any Voronoi cover for the gadget must have at least forty points.

Now we describe two minimal Voronoi covers for the variable gadget. See Figs. 5 and 6 for these two covers of the variable gadget. Figure 5 represents the case

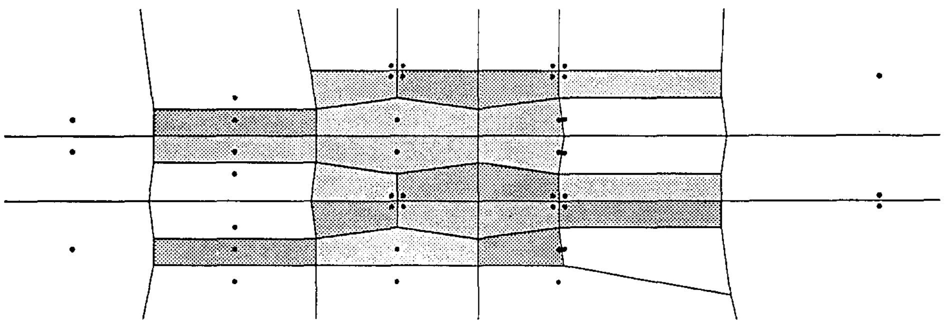

Fig. 6. Voronoi cover for variable gadget in false case.

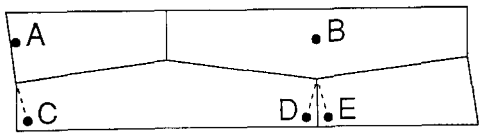

Fig. 7. Positioning of points in voronoi cover of variable gadget.

where the associated variable is true. Figure 6 represents the case where the variable is false. Modulo certain slight movements of the points, these are the only two minimal Voronoi covers for the variable gadget. A small piece of either cover is shown in Fig. 7. Points A and B can be moved vertically and still maintain a minimal Voronoi cover. As they are moved, points C, D, and E must also move along the dashed lines. If we let m be the absolute value of the slope of the segment separating the two pentagons, then points C, D, and E will be within distance $2Y_1/(1/m - m)$ of the nearest vertical segment.

4.2. Clause gadget

In this subsection, we describe a small tessellation whose minimal Voronoi cover depends on its interaction with the other gadgets to which it is connected. The gadget is shown in Fig. 8. The dimensions X_1 and Y_1 are the same as those which appeared in the variable gadget. Obviously, any Voronoi cover for this gadget must have a point in each of the three *connector* rectangles, and one in the large *clause* rectangle. Additionally, these interior points must be reflected through the edges of the rectangles. A minimal Voronoi cover for this gadget will share one external point of the clause rectangle with one external point of a connector rectangle. When the clause gadget is part of the complete circuit, this sharing can only happen when at least one of the associated variables is true. Thus, it will correctly compute the disjunction of the variables.

The following relationships exist among the dimensions of the variable gadget:

• $Y_d > X_1 + 2Y_1$. This ensures that the connector rectangles are far enough apart that they cannot share any of their external points. Thus, there will be twelve external points (and three internal points) needed by any Voronoi cover of the three connector rectangles.

• $Y_c = 3Y_1 + 2Y_d$. The clause rectangle is equal in height to the three spaced connector rectangles.

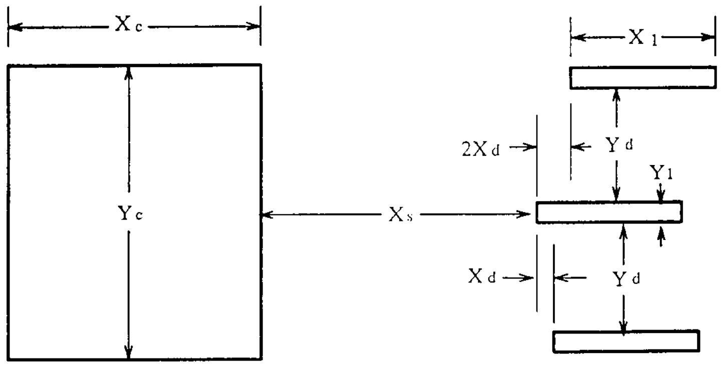

Fig. 8. Clause gadget.

- $X_d = X_1/8$.
- $X_s > 5X_1/8 + 2(2Y_d + 4Y_1)^2/X_1$.
- $X_c = X_s - X_1/8$.

To justify the last three requirements, we must see how the clause gadget is used. The three connector rectangles will be placed adjacent to other shapes that form a path back to the variable gadgets. This will restrict the positioning of points within the connector rectangles in a minimal Voronoi cover. We will assume that the points within the connector rectangles have the following properties:

- The vertical coordinate of the points is unknown (but of course, it must lie within the connector rectangle).
- If the literal (possibly negated variable) associated with a connector gadget is true, then the point will be centered horizontally within the rectangle ($X_1/2$ from both vertical edges).
- If the associated literal is false, then the point will be within distance X_T of the leftmost edge of the rectangle (but may be closer).

The clause gadget works on the following principle. The large clause rectangle can be covered with five points (one inside, four outside) in a Voronoi cover. However, when any of the three points inside the connector gadgets are centered, rather than close to the left edge, the clause rectangle can share its rightmost external point with the leftmost external point of a connector rectangle. The connector rectangles are shifted horizontally from each other, so that if more than one associated literal is true, only one of the left external points of the connector rectangles need be reflected into the clause rectangle. We demonstrate this with some examples. See Figs. 9, 10, and 11. These figures are not drawn to scale. The clause rectangle was made narrower and closer to the connector rectangles for illustrative purposes. In these figures, we assume that the points in the connector rectangles are fixed, and we find the minimal Voronoi cover given these fixed points.

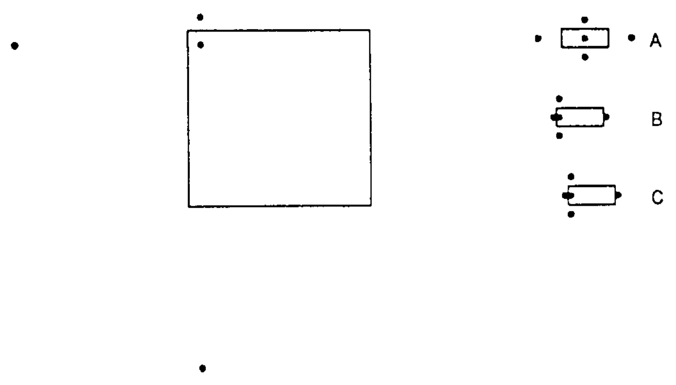

Fig. 9. Clause gadget. A is true, B and C are false.

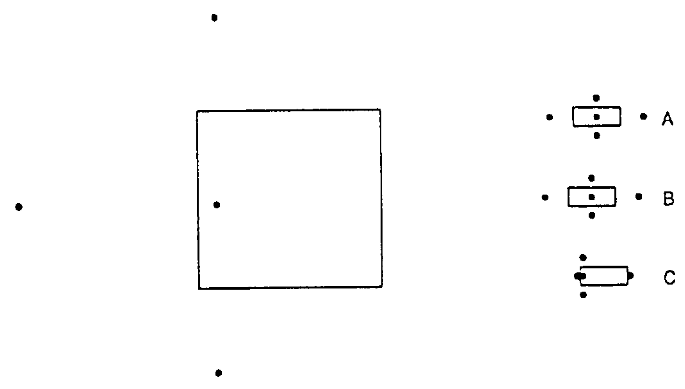

Fig. 10. Clause gadget. A and B are true, C is false.

In Fig. 9, one literal (A) is true. Note that the leftmost external point for the connector rectangles associated with literal A is close enough to the clause rectangle that the two rectangles can share it as an external point. In Fig. 10, two literals (A and B) are true. Once again, the clause rectangle can share an external point with one of the connector rectangles. In this case, that rectangle is B. Now we see why the connector rectangles must be staggered horizontally. Every point on the rightmost edge of the clause rectangle is closer to B's external point than to A's external point. This means that A's external point does not interfere with the definition of the rightmost edge of the clause rectangle. If they were not staggered, then the clause rectangle could not share an external point with a connector rectangle when two or more literals were true. In Fig. 11, none of the literals are true. The leftmost external points of all three connector rectangles are very close to their respective rectangles and too far from the clause rectangle to be shared. Thus, this case requires one more point in the minimal Voronoi cover.

So, if we can fix the points in the connector rectangles to be horizontally centered in the case of true literals and very close to the left edge in the case of

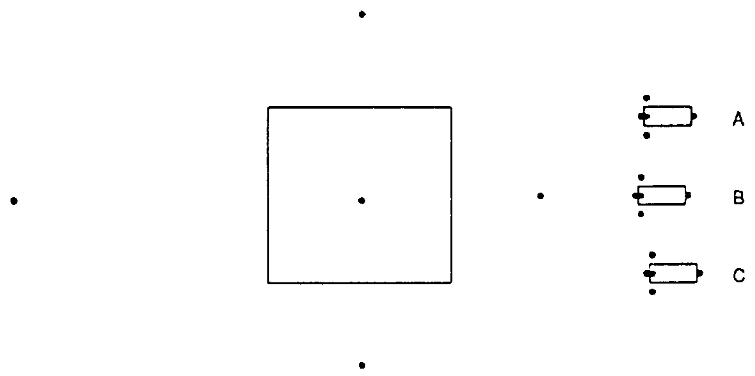

Fig. 11. Clause gadget, A, B, and C are false.

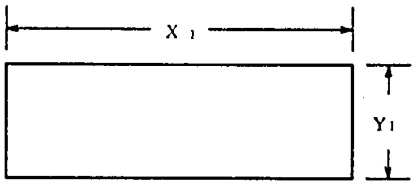

Fig. 12. Simplest connector gadget.

false literals, then the minimal Voronoi cover for the clause gadget will contain nineteen points if the clause is true and twenty points otherwise.

4.3. Connector gadget

The third type of tessellation we use is the *connector gadget.* This will be used for conveying the truth value of a literal to each of the clauses it appears in. In its simplest form, a connector gadget is just a rectangle. It is of size X_1 by Y_1, where X_1 and Y_1 are defined by the dimensions of the variable gadget. An example is shown in Fig. 12.

Clearly, this is a very simple gadget. It becomes more useful when we combine several of them together. Fig. 13 shows three coupled connector gadgets. The Voronoi cover has been constrained to contain one point centered horizontally within gadget A, as shown by the large circle. Note that in a minimal (constrained) Voronoi cover, that the other two gadgets will also contain horizontally centered points. Fig. 14 is the same set of gadgets, but the constrained point has been fixed at a distance δ from the leftmost edge of the rectangle it lies in. Note that in this case, the positioning of points in the rectangles alternates in the minimal constrained Voronoi cover. Every other rectangle has a point that is distance δ from the rightmost edge. This means that the connector gadget can convert a left-constrained point in one gadget to a right-constrained point in the next.

At this point, the purpose of the connector gadget should be clear. The variable gadget will 'choose' a truth value. The output of the variable gadget is obtained at its connector rectangles. Each of these rectangles will contain either a center-constrained or edge-constrained point in a minimal Voronoi cover. The connector gadgets couple the constraint to the clause gadget, where it remains of the same type (center or edge). There are several problems that this simple connector gadget cannot handle. Two of these are that the paths from the

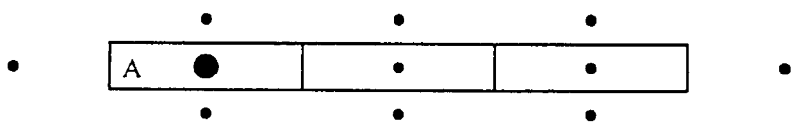

Fig. 13. Coupled connector gadgets, large point constrained to center.

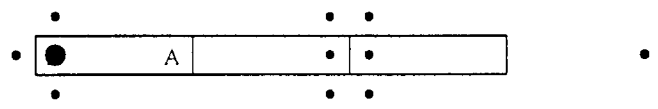

Fig. 14. Coupled connector gadgets, large point constrained to left.

variable gadgets to the clause gadgets may not be a straight line and may not have a length divisible by X_1.

We solve these problems by creating *angle connector gadgets*. These gadgets perform the same function as simple connector gadgets, propagating truth values via point constraints; however, they also change the angle of the path of connectors. An example is shown in Fig. 15. The connector gadget is shown twice, each time with a simple connector gadget on each end, to show how they are used together. In Fig. 15a, one connector gadget contains a center-constrained Voronoi cover point. In Fig. 15b, one connector contains an edge-constrained point. In both cases, a minimal Voronoi cover for the construction propagates the constraint type to the other end of the gadget. The gadget also changes the angle of propagation. An angle connector gadget requires twelve points in a minimal Voronoi cover, four of which are shared with adjacent connectors. Note that the quadrilaterals of the angle gadget which abut against simple connectors are longer than the connectors themselves (X_1). This is so that they correctly handle the edge-constrained case. The exact length does not matter, as long as it is sufficiently large. If we assume that the connection angle is no less than 90 degrees, then a length of $X_1 + Y_1$ is sufficient. The fact that the length is allowed to be longer can be used to handle path lengths not divisible by X_1.

Finally, the paths from variables to clauses must not cross and must not come sufficiently close to permit sharing of external points between connector gadgets. Because the original problem (planar 3-SAT) is planar, paths will never cross. By using the angle gadgets to move the paths apart we can prevent external points from coinciding. This can be achieved by moving the paths so that they are no closer than $X_1 + 2Y_1$.

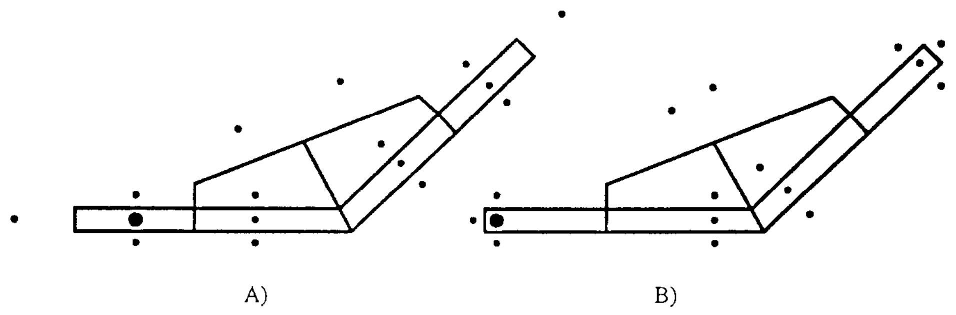

Fig. 15. Angle connector gadgets. a) is fixed center, b) is fixed left.

This distance is automatically achieved at the clause gadget. One way to accomplish this at the variable gadget is to increase the number of outputs of the variable gadget beyond the number strictly needed to feed the clause gadgets. When $2Y_2 > X_1 + Y_1$, using every other output is sufficient to achieve the correct separation.

4.4. *Putting it all together*

Given an instance of Planar 3-SAT, we convert it in polynomial time to a plane graph with a vertex for every variable and for every clause. Next, we construct the variable and clause gadgets. We connect them up with connector gadgets. For each variable u_i we compute $|\mathrm{mvc}(u_i)|$, the number of points in a minimal Voronoi cover for the gadget. For each clause C_i, we compute $|\mathrm{mvc}(C_i)|$, the size of a minimal Voronoi cover for the clause gadget, assuming that at least one of its inputs is true. Finally, we compute the number of points needed in a minimal Voronoi cover of the connector gadgets. Let k be the sum of the sizes of the minimal Voronoi covers for all the gadgets. Now, we ask if there is a minimal Voronoi cover of size k for the entire tessellation.

4.1. *The Voronoi cover problem is NP-hard.*

Proof sketch. Assume that there is a satisfying assignment to the Planar 3-SAT problem. Then, the following is a Voronoi cover of size k:

• For each variable u_i, the Voronoi cover for the gadget associated with u_i is the true-variable gadget or the false-variable gadget according to the truth value of u_i in the satisfying assignment.

• For each connector gadget, we can generate a Voronoi cover of minimal size by assuming that the outputs of the variable gadgets are fixed and using the 'constrained' covers described above.

• For each clause gadget C_i, at least one incoming literal must be true. This means that a minimal Voronoi cover for the clause can be constructed with $|\mathrm{mvc}(C_i)|$ points such that its interface to the connector gadgets is defined properly.

Consider the case when there is no satisfying assignment to the Planar 3-SAT problem. Assume by way of contradiction that there is a Voronoi cover of size k for the entire tessellation. This implies that each gadget in the construction must have a Voronoi cover defined by a subset of the points in the complete Voronoi cover. By our construction, the Voronoi covers of the gadgets must be minimal. Thus, the Voronoi covers of the variable gadgets define a satisfying assignment. □

5. Recognizing Voronoi diagrams

In this section, we describe an algorithm that can determine if a polygonal planar tessellation of n regions is a Voronoi diagram, and if so can find a set of points whose Voronoi diagram is that tessellation. This problem can be considered a restriction of the Voronoi Cover problem in which the size of the cover is equal to the number of regions in the tessellation, *i.e.*, no regions are split. The set of points (if any) found by the algorithm is a Voronoi Cover. A description of the algorithm follows.

We assume that the tessellation is given in the following form. We are given a list of polygonal regions, each of which is a sorted list of edges. Given any edge, we can find the two regions that border it in constant time. Each edge is represented by a line $ax + by + c = 0$, where a, b, and c are integers.

Each of the regions of a Voronoi diagram is convex. The first step of the algorithm is to ensure that this property holds for the tessellation in question by testing the convexity of each region. This step is not strictly necessary, but makes the following discussion easier. This step can be completed in linear time.

Our technique is to transform our problem into an instance of two-variable linear programming. Megiddo [4] has shown that two-variable linear programming can be solved in linear time.

We pick a region R_s to be the *root* region. Let (X_s, Y_s) be the unique Voronoi point in region R_s. X_s and Y_s will be the two variables in our linear program. We will solve the problem either by finding legal values for X_s and Y_s or by finding that no such values exist. In the first case, we can find an n-Voronoi cover; in the second, we know that there is no such cover.

We will use the dual graph D of the input tessellation in our algorithm. We define D as follows. A vertex in D represents both a region of the tessellation and the Voronoi point contained in the region. Two vertices in D are connected with an edge if and only if the regions they represent are adjacent in the tessellation. In linear time, we can find a breadth-first spanning tree for this graph. The root region defines the root of this tree and the regions adjacent to the root region become children of the root of the tree. Note that each edge of the tree corresponds to an edge in the tessellation, but there may be many edges in the tessellation with no corresponding edges in the spanning tree.

First, we make a simple observation. Let $p_i(X_i, Y_i)$ and $p_j = (X_j, Y_j)$ be the Voronoi points in two adjacent regions, R_i and R_j, separated by an edge $ax + by + c = 0$. Given p_j, we can find p_i by "flipping" p_j over the common edge. Precisely, there is a linear relationship between p_i and p_j, that

$$X_i = \frac{-(a^2+b^2)}{a^2-b^2} X_j + \frac{-2ab}{a^2-b^2} Y_j + \frac{-2ac}{a^2-b^2}$$

and

$$Y_i = \frac{2ab}{a^2-b^2} X_j + \frac{a^2+b^2}{a^2-b^2} Y_j + \frac{2bc}{a^2-b^2}.$$

There is a unique path from the root of the spanning tree to each vertex in D. This path represents a chain of linear transforms from p_s to every Voronoi point in the Voronoi cover. We will use these paths to define the Voronoi points in each region in terms of the Voronoi point in the root region, p_s. Define depth(i) to be the depth of region i in the spanning tree. The Voronoi point for a region of depth $d > 0$, can be calculated inductively using the above transform from its parent in the tree, which has depth $d - 1$. Clearly, the vertex at root of the tree is already defined in terms of p_s. So, we can express each Voronoi point as a linear function of X_s and Y_s.

Given the formulation of each of the Voronoi points, we can constrain them to lie within the regions they define. For a region R_i with k edges, this will entail adding k constraints of the form

$$aX_i + bY_i + c \geqslant 0 \quad \text{or} \quad aX_i + bY_i + c \leqslant 0,$$

where

$$ax + by + c = 0$$

is an edge of the region. We will always express X_i and Y_i in terms of X_s and Y_s, as above.

The constraints we construct in the above step guarantee that each region contains one Voronoi point, and that if two regions are adjacent in the spanning tree, then their Voronoi points are reflections of each other through the edge common to both regions. However, this does not ensure that points belonging to regions adjacent in the tessellation, but not in the spanning tree are reflections of each other. So, for each pair of adjacent regions in the tessellation, we must ensure that the corresponding Voronoi points are reflections of each other.

Let R_i and R_j be two adjacent regions that are not adjacent in the spanning tree. Let $p_i = (X_i, Y_i)$ and $p_j = (X_j, Y_j)$ be the Voronoi points for these regions, respectively. Then, p_i must be the reflection of p_j through the common edge $ax + by + c = 0$. That is,

$$X_i = \frac{-(a^2 + b^2)}{a^2 - b^2} X_j + \frac{-2ab}{a^2 - b^2} Y_j + \frac{-2ac}{a^2 - b^2}$$

and

$$Y_i = \frac{2ab}{a^2 - b^2} X_j + \frac{a^2 + b^2}{a^2 - b^2} Y_j + \frac{2bc}{a^2 - b^2}.$$

Finally, we feed the linear program to an algorithm for finding a solution for X_s and Y_s, or determining that no such solution exists.

This algorithm relies on properties of the Voronoi diagram—that each Voronoi point is contained in the region that it defines and that the Voronoi points in adjacent regions are reflections through the common edge between the regions. If there is no solution to the linear program we construct, then there is no set of points with these properties and input tessellation cannot be a Voronoi diagram.

Now consider the case when there is a solution to the linear program we create. We claim that the tessellation is a Voronoi diagram, and that we can obtain an n-Voronoi Cover for the tessellation by picking any feasible solution p_s and mapping it into a Voronoi Cover point for each region via the linear transform above. For each region R, call the point obtained in such a manner p_R. Let P be the set of n such points. To prove that the initial tessellation is a Voronoi diagram, we must merely show that, for each region R, any point in R is at least as close to p_R as it is, to any other point in P. Note that if there is a point inside R that is further from p_R than it is from some other point $p_{R_i} \in P$, then there must be a point p on the boundary of R which is also closer to p_{R_i} than it is to p_R.

So, by way of contradiction, assume that the tessellation is not a Voronoi diagram, *i.e.*, there is a point p on the boundary of a pair of adjacent regions R_1 and R_2 which is closer to point $p_{R_i} \in P$ than it is to p_{R_1} (and, by symmetry, to p_{R_2}). We analyze this assumption with two cases, shown in Fig. 16.

In the first case, regions R_1 and R_i are adjacent in the tessellation and share a common edge. This edge divides the plane into two half-planes. Points in the half-plane that contains p_{R_1} are closer to p_{R_1} than they are to p_{R_i}. Points in the other half-plane are closer to p_{R_i}. Thus, p must be in the half-plane that contains p_{R_i}. R_1 is defined as the intersection of the closed half-planes defined by the edges of R_1. For any edge e of R_1, no point in the half-plane defined by e which does not contain p_{R_1} can lie in R_1. Thus, p cannot lie in R_1, which is a contradiction.

In the second case, regions R_1 and R_i are not adjacent in the tessellation, so do not share an edge. Let l be a line segment from p_{R_i} to p. l must cross at least one edge of the tessellation because R_1 and R_i are not adjacent. Let k be the number of edges crossed by l. Starting at p_{R_i}, let R_j be the second region encountered along l (R_i is the first). Note that $R_j \neq R_1$. The edge between R_i and R_j separates the plane into those points that are closer to p_{R_i} on one side, and those that are closer to p_{R_j} on the other. The point p is on the R_j side and is therefore closer to p_{R_j} than it is to p_{R_i} or, by transitivity, to p_{R_1}.

If p_{R_j} is adjacent to p_{R_1}, then the argument in case one, above, may be applied to result in a contradiction. Otherwise, we can repeat the argument in case two, but the value of k defined as above will be one less. k can never be more than the number of edges in the tessellation, and case two only applies when k is positive, so the above argument eventually reaches a contradiction in case one.

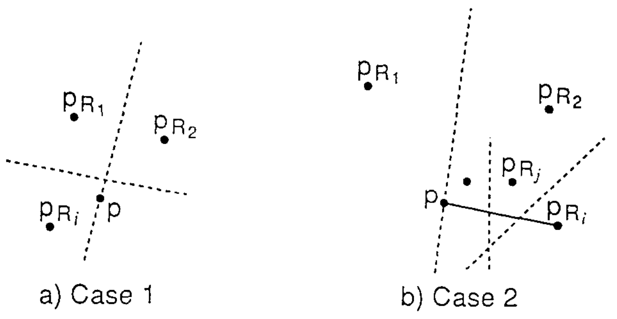

Fig. 16. Showing tessellation to be a Voronoi diagram.

Each edge in the tessellation contributes two equations to the linear program. It may appear that the linear program is linear in size and should therefore be solvable in linear time. However, the coefficients in each equation can grow to require linear size for their representation. Thus, in the worst case, the linear program can be quadratic in size, and can be found in quadratic time.

Theorem 5.1. *Given a polygonal tessellation in the plane, it is possible to find the set of points whose Voronoi diagram is the tessellation, or determine that no such set exists, in quadratic time.*

6. Conclusion

We have presented an efficient algorithm for recognition of Voronoi Diagrams and computing the set of points that generate a given Voronoi Diagram. On the other hand, we have shown that this problem becomes NP-hard when we generalize it to the Voronoi cover problem, where regions of the given tessellation can be subdivided into multiple Voronoi regions. This suggests that it is difficult to teach concepts to the nearest neighbor learning algorithm using a minimum number of examples.

Possible research directions include finding provably good approximations to tesellations by Voronoi Diagrams. This could be useful in communication of concepts to nearest neighbor learning algorithms.

References

[1] P. Ash and E. Bolker, Recognizing Dirichlet tessellations, Geom. Dedicata 19 (1985) 175–206.

[2] J. Hopcroft and R. Tarjan, Efficient planarity testing, J. ACM 21 (1974) 549–568.

[3] D. Lichtenstein, Planar formulae and their uses, SIAM J. Comput. 11 (1982) 329–343.

[4] N. Megiddo, Linear-time algorithms for linear programming in R^3 and related problems, SIAM J. Comput. 12 (1983) 759–776.

[5] F. Preparata and M. Shamos, Computational Geometry—An Introduction (Springer, New York, 1985).

[6] S. Salzberg, A. Delcher, D. Heath and S. Kasif, Learning with a helpful teacher, Technical Report JHU-90/14, Department of Computer Science, The Johns Hopkins University, 1990.

[7] A. Suzuki and M. Iri, Approximation of a tessellation of the plane by a Voronoi diagram, J. Oper. Res. Soc. Japan 29 (1986) 69–96.