Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. 180

Frequent Subgraph Retrieval in Geometric Graph Databases

Sebastian Nowozin,[1] Koji Tsuda[1]

November 2008

[1] Department Schölkopf, email: nowozin;tsuda@tuebingen.mpg.de

Frequent Subgraph Retrieval in Geometric Graph Databases

Sebastian Nowozin, Koji Tsuda

Abstract. Discovery of knowledge from geometric graph databases is of particular importance in chemistry and biology, because chemical compounds and proteins are represented as graphs with 3D geometric coordinates. In such applications, scientists are not interested in the statistics of the whole database. Instead they need information about a novel drug candidate or protein at hand, represented as a query graph. We propose a polynomial-delay algorithm for geometric frequent subgraph retrieval. It enumerates all subgraphs of a single given query graph which are frequent geometric ϵ-subgraphs under the entire class of rigid geometric transformations in a database. By using geometric ϵ-subgraphs, we achieve tolerance against variations in geometry. We compare the proposed algorithm to gSpan on chemical compound data, and we show that for a given minimum support the total number of frequent patterns is substantially limited by requiring geometric matching. Although the computation time per pattern is larger than for non-geometric graph mining, the total time is within a reasonable level even for small minimum support.

1 Introduction

Frequent subgraph mining algorithms (FSM) such as gSpan [1] and extensions have been successfully applied to different applications such as chemical compound classification [2] and molecular QSAR analysis [3], graph clustering [4] and computer vision [5]. In FSM, we deal with a large database of attributed graphs, where nodes and edges have discrete labels. The purpose of FSM is to enumerate all frequently appearing subgraphs in the database. Typically, all subgraphs whose frequency is above a minimum support threshold are enumerated. FSM helps users to analyze the database and can be used to create a feature space for subsequent machine learning algorithms as well.

One domain of graph mining is geometric graph mining, where each node of a graph has 2D or 3D coordinates and subgraph patterns have to match geometrically under a given transformation class [6, 7, 8]. To deal with real world data, a geometric pattern is matched with a certain tolerance ϵ, because a perfect match does not happen in real data. Due to the tolerance, geometric mining is significantly more difficult than ordinary graph mining. There are many different geometric representations of the same pattern, all of which are ϵ-isomorphic to each other. To enumerate all patterns satisfying minimum support, we need to define a canonical pattern for these equivalent patterns. We call this problem the *pattern ambiguity problem*. To the best of our knowledge, there is no effective solution to this problem.

Extracting such geometric patterns from molecular 3D structures is one of the central topic in computational biology, and numerous approaches have been proposed. Most of them are optimization methods, which detect one pattern at a time by minimizing a loss function (e.g., [9, 10, 11]). They are different from our approach enumerating all patterns satisfying a certain geometric criterion. In particular, they do not have a minimum support constraint. Instead they try to find a motif that matches all graphs.

Kuramochi et al. [6] used a heuristic way to extend a geometric pattern, so there is no explicit description of the set they enumerated. Deshpande et al. [7] used non-geometric graph patterns from chemical molecules augmented with average inter-atomic distances. Their patterns contain geometric information, but only in a summarized form. Arimura et al. [8] proposed a method to enumerate all maximum geometric patterns, but they assumed zero tolerance and therefore small variations of the vertex coordinates are treated as different patterns. Huan et al. [12] applied an enumerative approach to detect structural motifs, but proteins are represented as graphs with discrete labels in advance by thresholding the distances among atoms.

We deal with the problem of frequent subgraph retrieval (FSR), which is similar to but different from frequent subgraph mining. In mining, frequently appearing subgraph patterns in the graph database are enumerated. In retrieval, the user has a *query graph* apart from the database. FSR is frequent subgraph mining, where the set

of patterns is restricted to the set of subgraphs of the query graph. In biology and chemistry applications, this setting makes sense, because users access databases to gain some knowledge about a newly discovered chemical compound or protein at hand. In such a case, geometric patterns that do not match the query graph are not useful. Furthermore, the statistics or characteristics of the whole database are not of interest here, which is strikingly different from the market basket analysis setting addressed by the usual frequent mining algorithms.

Our FSR algorithm is based on reverse search [13], and the pattern set to be enumerated is unambiguously described as an exact subgraph of the query graph, thereby avoiding the pattern ambiguity problem. Straightforwardly, one can create a retrieval version of the popular gSpan frequent subgraph mining algorithm [1], which we call gSpan-retrieval. It does not use geometric information and the search tree is pruned as soon as the pattern is not included in the given query graph. However, gSpan-retrieval is exponential-delay, due to the minimum DFS code checks in the gSpan algorithm. While our proposed method is slower than gSpan-retrieval in practice, we prove that our method has polynomial-delay [14].

In experiments, we used four chemical datasets with 3D atomic coordinates. In efficiency comparison with gSpan-retrieval, we will show that the geometric information reduces the number of frequent patterns found significantly. While the computational time per pattern of our method is much larger than that of gSpan-retrieval, when the minimum support threshold is decreased, the number of patterns of gSpan explodes super-exponentially to an intractable level, whereas our method can keep the number of patterns small due to the discriminative geometry required for matching. To prove that our small set of geometric patterns are still informative, naive Bayes prediction of chemical activities is performed based on our patterns and the patterns by gSpan-retrieval. On the four data sets examined, the prediction accuracy is improved significantly on three sets.

2 Method

In order to describe our method we first define the notion of geometric graph and geometric ϵ-subgraph, followed by the problem statement of frequent geometric subgraph retrieval.

Definition 1 (Geometric graph) *A labeled, connected, undirected geometric graph $G = (V, E, \mathcal{L}^V, \mathcal{L}^E, C^V)$ consists of a vertex set $V \subset \mathbb{N}$, an edge set $E \subseteq V \times V$, a vertex labeling $\mathcal{L}^V : V \to \mathbb{N}$, an edge labeling $\mathcal{L}^E : E \to \mathbb{N}$ and vertex coordinates $C^V : V \to \mathbb{R}^d$, assigning each vertex a vector in $\mathbb{R}^d$. Let $\mathcal{G}$ denote the set of all possible geometric graphs and let $\emptyset \in \mathcal{G}$ denote the empty graph.*

Definition 2 (Geometric ϵ-subgraph relation) *Given two geometric graphs $g_1 = (V_1, E_1, \mathcal{L}^{V_1}, \mathcal{L}^{E_1}, C^{V_1})$, $g_2 = (V_2, E_2, \mathcal{L}^{V_2}, \mathcal{L}^{E_2}, C^{V_2})$, $g_1, g_2 \in \mathcal{G}$, and a geometric tolerance $\epsilon \geq 0$, we define $g_1 \subseteq_\epsilon g_2$ to be true if and only if $\exists(m, T)$, $m : V_1 \to V_2$ injective and complete on V_1, $T : \mathbb{R}^d \to \mathbb{R}^d$ with T being a rigid transformation, such that*

1. *$\forall i \in V_1 : \mathcal{L}^{V_1}(i) = \mathcal{L}^{V_2}(m(i))$, the vertex labels are matched, and*
2. *$\forall (i,j) \in E_1 : (m(i), m(j)) \in E_2, \mathcal{L}^{E_1}(i,j) = \mathcal{L}^{E_2}(m(i), m(j))$, all edges of g_1 also exist in g_2 with the correct label, and*
3. *$\forall i \in V_1 : \|C^{V_1}(i) - T(C^{V_2}(m(i)))\| \leq \epsilon$, the geometry under transformation matches up to the required tolerance.*

We write $g_2 \supseteq_\epsilon g_1$ iff $g_1 \subseteq_\epsilon g_2$. Also, we write $g_1 \subset_\epsilon g_2$ iff $g_1 \subseteq_\epsilon g_2$ and $|V_1| < |V_2|$.

Problem 1 (Frequent geometric subgraph retrieval) *For a given geometric* query graph *$q \in \mathcal{G}$, a geometric graph database $G = \{g_1, \ldots, g_{|G|}\}$, $g_i \in \mathcal{G}$, a minimum support threshold $s > 0$, and a geometric tolerance $\epsilon \geq 0$, find all $g \in \mathcal{G}$ with $g \subseteq_0 q$ such that g is frequent enough in G, i.e. we have*

$$|\{i = 1, \ldots, |G| : g \subseteq_\epsilon g_i\}| \geq s.$$

Comparing Problem 1 with a general frequent mining problem we note that it allows tolerant matching into the database for counting the support, but requires exact matching in the query graph. This simplification with regards to the query graph makes it tractable to solve the problem efficiently, as the set $\{g \in \mathcal{G} : g \subseteq_0 q\}$ is finite. This also overcomes the pattern ambiguity problem by making the query graph the canonical reference.

Our proposed method solves Problem 1 and is composed of two parts, i) the enumeration of all $g \subseteq_0 q$ by means of reverse search and, ii) the geometric matching $g \subseteq_\epsilon g_i$ within the graph database. Later we will integrate these two steps, enabling us to prune large parts of the search space efficiently without losing any frequent ϵ-subgraph. We now discuss the two parts separately.

2.1 Subgraph Enumeration by Reverse Search

We generate candidate frequent geometric subgraphs by recursively enumerating the set of all exact geometric subgraphs of the query graph, i.e. $\{g \in \mathcal{G} : g \subseteq_0 q\}$ and explicitly checking the frequency in the geometric graph database.

Enumerating all subgraphs of a given graph is itself a non-trivial problem. To solve the subgraph enumeration problem efficiently, we use the *reverse search* algorithm [13]. Because the potential number of subgraphs in a given graph is exponential in the number of nodes and edges, establishing a polynomial time complexity in terms of the input size is impossible. Instead, the alternative notions of *polynomial delay* and *output-polynomial* time complexity are used, where polynomial delay means there exist a bound on the time complexity between succeeding outputs which is polynomial in the input dimensions [14]. Output-polynomial time complexity means there exist a bound on the total time complexity which is a polynomial in terms of the input and output dimensions; for substructure mining algorithms output polynomial time follows from polynomial delay. For our proposed algorithm we will prove the existence of a polynomial delay bound.

In the general reverse search framework, we enumerate elements from a set $\mathcal{X}$ by means of a *reduction mapping* $f : \mathcal{X} \to \mathcal{X}$. The reduction mapping reduces any element from $\mathcal{X}$ to a "simpler" one in the neighborhood of the input element. The mapping is chosen such that when it is applied repeatedly, we eventually reduce it to some *solution elements* in the set $\mathcal{S} \subset \mathcal{X}$. Formally, we write $\forall x \in \mathcal{X} : \exists k \geq 0 : f^k(x) \in \mathcal{S}$.

In our concrete subgraph enumeration problem, we identify $\mathcal{X}$ with the set of all possible connected labeled graphs $\mathcal{G}$. This setting is similar to the enumeration of all connected induced subgraphs in Avis and Fukuda [13]; but here we distinguish connected subgraphs also on the presence of edges. To this end, the mapping $f : \mathcal{G} \to \mathcal{G}$ removes either one edge or one vertex-edge. By evaluating the mapping repeatedly the graph is shrunk to the empty graph. Thus, here we have $\mathcal{S} = \{\emptyset\}$.

Precisely, we define the reduction mapping as follows.

Definition 3 (Reduction Mapping f_q) *Given a geometric query graph $q \in \mathcal{G}$, a geometric graph $g \in \mathcal{G}$ such that $g \subseteq_0 q$, i.e. g is an exact geometric subgraph of the query graph, and let $U^{E_q} : E_q \to \mathbb{N}$ and $U^{V_q} : V_q \to \mathbb{N}$ be an index labeling assigning unique indices to edges and vertices of q, respectively. We then define the mapping $f_q : \mathcal{G} \to \mathcal{G}$ which reduces g to $f_q(g)$, with $g \supset_0 f_q(g)$ by performing the following actions.*

1. *If g contains one or more edges forming a cycle, the graph $f_q(g)$ is the same as g, with the edge corresponding to the highest-index cycle-edge in q removed.*

2. *If g contains no edges forming a cycle, the graph $f_q(g)$ is the same as g, with the leaf vertex corresponding to the highest-index vertex in q and its adjacent edge removed, if it exist. A vertex is said to be a* leaf*, if it only has one adjacent edge.*

Clearly the above mapping is well-defined because each possible graph $g \subseteq_0 q$ is successively reduced to the empty graph in a unique way. By considering $f_q(g)$ for all possible $g \in \mathcal{G}$, $g \subseteq_0 q$, a *reduction tree* is defined, with $\emptyset \in \mathcal{G}$ being the root node. Reverse search performs subgraph enumeration by inverting the reduction mapping such that the tree is explored from the root node towards the leaves.

The inverse mapping $f_q^{-1} : \mathcal{G} \to \mathcal{G}^*$ generates for a given graph $g \subseteq_0 q$ a set X of enlarged graphs $g' \in X$, $g' \supset_0 g$, $g' \subseteq_0 q$ such that $f_q(g') = g$. The mapping follows uniquely from definition 3 but for clarity we give it explicitly as follows.

Definition 4 (Inverse reduction mapping f_q^{-1}) *Given a graph $g \in \mathcal{G}$ and a query graph $q \in \mathcal{G}$, and index labellings $U^{E_q} : E_q \to \mathbb{N}$, $U^{V_q} : V_q \to \mathbb{N}$, the mapping $f_q^{-1} : \mathcal{G} \to \mathcal{G}^*$ outputs the following sets for $f_q^{-1}(g)$.*

1. *If $g = \emptyset$, so that g is the empty graph, then*

$$f_q^{-1}(\emptyset) = \big\{(\{i\}, \emptyset, \{i \to \mathcal{L}^{V_q}(i)\}, \emptyset, \{i \to C^q(i)\}) : i \in V_q\big\},$$

   the set of all extensions containing a single vertex from the query graph.

2. *If $g \neq \emptyset$, so g is non-empty, then*

$$f_q^{-1}(g) = X_E(g, q) \cup X_V(g, q),$$

$$T(\alpha,\beta,\gamma,t_x,t_y,t_z) \quad := \quad \left[\begin{array}{cccc} \cos(\alpha)\cos(\beta), & \cos(\alpha)\sin(\beta)\sin(\gamma)-\sin(\alpha)\cos(\gamma), & \cos(\alpha)\sin(\beta)\cos(\gamma)+\sin(\alpha)\sin(\gamma), & t_x \\ \sin(\alpha)\cos(\beta), & \sin(\alpha)\sin(\beta)\sin(\gamma)+\cos(\alpha)\cos(\gamma), & \sin(\alpha)\sin(\beta)\cos(\gamma)-\cos(\alpha)\sin(\gamma), & t_y \\ -\sin(\beta) & \cos(\beta)\sin(\gamma) & \cos(\beta)\cos(\gamma), & t_z \\ 0, & 0, & 0, & 1 \end{array}\right]$$

Figure 1: General 3D rigid transformation matrix used.

where $X_E(g,q) : \mathcal{G} \times \mathcal{G} \to \mathcal{G}^*$ is the set of "edge-only extensions", and $X_V(g) : \mathcal{G} \times \mathcal{G} \to \mathcal{G}^*$ is the set of "edge-vertex extensions". The sets are defined as follows.

$$\begin{aligned} X_E(g,q) = \{&(V_g, E_g \cup \{(i,j)\}, \mathcal{L}^{V_g}, \\ &\mathcal{L}^{E_g} \cup \{(i,j) \to \mathcal{L}^{E_q}(i,j)\}, C^g) : \\ &(i,j) \notin E_g, U^{E_q}(i,j) > U^{E_q}(s,t) \\ &\textit{for all cycle edges } (s,t) \in E_g \cup \{(i,j)\}\} \end{aligned}$$

This definition of $X_E(g,q)$ outputs exactly all extensions g' of g such that only one edge is added and we have $f_q(g') = g$. The edge-vertex extension set $X_V(g,q)$ is defined as follows.

- *if $|E_g| \geq |V_g|$, i.e. g contains a cycle, then*

$$X_V(g,q) = \emptyset.$$

- *if $|E_g| < |V_g|$, i.e. g is a tree, then*

$$\begin{aligned} X_V(g,q) = \{&(V_g \cup \{i\}, E_g \cup \{(j,i)\}, \\ &\mathcal{L}^{V_g} \cup \{i \to \mathcal{L}^{V_q}(i)\}, \\ &\mathcal{L}^{E_g} \cup \{(j,i) \to \mathcal{L}^{E_q}(j,i)\}, \\ &C^g \cup \{i \to C^q(i)\}) : \\ &i \notin V_g,\ U^{V_q}(i) > U^{V_q}(s) \textit{ for all} \\ &\quad s \in V_g \cup \{i\}, \\ &\quad s \textit{ being a leaf node in } V_g \cup \{i\}\}, \end{aligned}$$

The vertex-edge extension set $X_V(g,q)$ adds edge-vertex pairs from q to obtain $g' \supset_0 g$, $g' \subseteq_0 q$ such that $f_q(g') = g$.

When we recursively evaluate f_q^{-1} we can traverse the tree and thus efficiently enumerate all subgraphs of q without generating duplicate subgraphs. Before we give a detailed description of the overall algorithm, we consider the second subproblem of geometrically matching the enumerated graphs in the graph database.

2.2 Matching with tolerances

Checking the geometric ϵ-subgraph relation requires one to test whether $g \subseteq_\epsilon g_i$ is satisfied. In this paper we limit ourselves to the practically relevant case of 3D vertex coordinates, i.e. $C^V : V \to \mathbb{R}^3$ and rigid transformations. However, our general algorithm works for arbitrary dimensions and transformations.

For the 3D case, we parametrize the general rigid transformation matrix using homogeneous coordinates by a translation vector (t_x, t_y, t_z) and three rotation angles (α, β, γ) around the canonical axes. The resulting matrix is shown in Figure 1. Consider the question whether there exist a set $(\alpha, \beta, \gamma, t_x, t_y, t_z)$ of parameters such that we can *align* two given point sets $X = \{x_1, \ldots, x_N\}$, $Y = \{y_1, \ldots, y_N\}$, $x_i, y_i \in \mathbb{R}^3$ such that the norm between each original point and the matched point under the transformation satisfies a given tolerance $\epsilon \geq 0$, i.e. whether

$$\begin{aligned} \exists\, (\alpha,\beta,\gamma,t_x,t_y,t_z) : \forall i = 1,\ldots,N : \\ \|x_i^h - T(\alpha,\beta,\gamma,t_x,t_y,t_z) y_i^h\| \leq \epsilon, \end{aligned}$$

where x^h and y^h are the homogeneous extensions of x and y, respectively.[1] The "no-effect" transformation is simply $T_0 := T(0,0,0,0,0,0)$. We directly minimize the squared error

$$\sum_{i=1}^{N} \|x_i^h - T(\alpha,\beta,\gamma,t_x,t_y,t_z)y_i^h\|^2$$

using BFGS [15]. This six-dimensional minimization problem is non-convex, however for the case of $\mathbb{R}^3$ we consider here we are guaranteed to obtain the optimal solution, essentially because we add one vertex at a time and all six parameters can be uniquely recovered from three or more vertex correspondences.[2]

By solving the above optimization problem we can determine whether a good alignment exists. Now we have the two ingredients – subgraph enumeration and geometric matching – necessary to state the whole algorithm.

2.3 Algorithm

The algorithm uses location sets $L_G(g)$ which map the occurrence of a geometric subgraph g into the geometric graph database G. Let

$$L_G(g) \quad = \quad \{(i,m,T) : g \subseteq_\epsilon g_i \text{ with node-matching } m \text{ under rigid transformation } T\}$$

For convenience of notation in the description of the algorithm, let $m : \mathbb{N}^* \to \mathbb{N}^*$ be the per-element extension of the matching $m : \mathbb{N} \to \mathbb{N}$, m^{-1} be the inversion of m, and $T : (\mathbb{R}^d)^* \to (\mathbb{R}^d)^*$ be the per-element extension of the rigid transformation. Algorithm 1 consists of three functions, FREQGEO, MINE and FILTERLOCATION. The main function is MINE, which is called once for each node visited in the reverse search tree. The main loop in line 10 calls MINE for all child nodes of the current node, effectively walking the reverse search tree in a depth-first fashion. While this tree walk is performed, a *location list* of occurrences of the current geometric subgraph is kept and updated by means of the FILTERLOCATION function.

The FILTERLOCATION function is called for each $g' \supset_0 g$ extension where all matches of g are recorded in a location list. It filters the location list such that only those matches in $g_i \in G$ are kept which fulfill $g' \supseteq_\epsilon g_i$. To this end, for all current matchings, all possible extension edges are enumerated in a set Q and a new graph g_{test} is constructed (line 28) and tested for geometric matching (line 31–36). If it does not mach, the location list entry is discarded. If it does match, the location list entry is updated to include the proper enlarged matching.

The predicate "g is visited for the first time" in line 8 of the algorithm tests whether it is globally the first visit of g as a ϵ-subgraph. Without limit of generality we can assume $q \in G$, then this check can be performed in $O(1)$, as then all occurrences of $g \subseteq_\epsilon q$ are in the location list $L_G(g)$ in a unique order; therefore we simply have to check if g is the first element in q's location list.[3] Based on the value of this predicate, the REPORT function can decide among the following two behaviors.

- To report only the first occurrence of $g \subseteq_\epsilon q$, immediately returning whenever the predicate is false,
- To report every occurrence of $g \subseteq_0 q$, $g' \subseteq_0 q$, even if both $g \subseteq_\epsilon g'$ and $g' \subseteq_\epsilon g$ hold.

For all experiments we use the first option.

2.4 Complexity Analysis

We now show that Algorithm 1 for the important case of coordinates in $\mathbb{R}^3$ enumerates all frequent geometric subgraphs of a given query graph with *polynomial delay* and therefore in *output polynomial time*.

[1] $x^h = [x', 1]'$.

[2] To see this, first note that we fix the vertex correspondences between the two graphs in advance and we search over rigid transformations which are invertible. Additionally before we add a point to the set we are guaranteed that the old set matches under a transformation T^* to the given tolerance. For the initial case of $N = 1$ its easy to see there always exist an optimal transformation with zero objective by merely shifting the vertex. For $N \geq 2$, the new set matches if the new point matches with a small change to T^*. Because there is already at least one point in the set to be matched this constrains T to lie in a small region around T^*. Because T is smooth in all six parameters, for small enough ϵ and if the new point matches, we will always have $\|T^* - T\|$ small enough. Then the solution in our parametrization will be close to the previous solution.

[3] If $q \notin G$, we define $G' = G \cup \{q\}$ and perform mining on G' with minimum support increased by one.

Algorithm 1 Frequent Geometric Subgraph Retrieval (FreqGeo)

Input:

Graph database: $G = (g_1, \ldots, g_{|G|}), g_i \in \mathcal{G}$.
Query graph: $q \in \mathcal{G}$.
Minimum required support: $s \geq 1$.
Matching tolerance: $\epsilon \geq 0$.

Algorithm:

function FREQGEO(G, q, s, ϵ)
  MINE($G, \emptyset, q, s, \epsilon, \{(i, \emptyset, T_0) : i = 1, \ldots, |G|\}$)
end function
function MINE($G, g, q, s, \epsilon, L_G(g)$)
  if $|L_G(g)| < s$ **then** ▷ Minimum support satisfied?
    return
  end if
  REPORT($g, L_G(g), g$ is visited for the first time) ▷ Report $g \subseteq_0 q$ as frequent geometric subgraph
  $X \leftarrow f_q^{-1}(g)$ ▷ All valid extensions of g in q
  for $g' \in X$ **do** ▷ For all extensions, recurse
    MINE($G, g', s, \tau,$ FILTERLOCATION($G, L_G(g), g', g, \epsilon$))
  end for
end function
function FILTERLOCATION($G, L_G(g), g', g, \epsilon$)
  $L_G(g') \leftarrow \emptyset$
  if $g = \emptyset$ **then** ▷ First extension
    $k \leftarrow$ single node index in $V_{g'}$
    for $(i, m, T) \in L_G(g)$ **do**
      for $j \in \{s \in V_{g_i} : \mathcal{L}^{V_{g_i}}(s) = \mathcal{L}^{V_{g'}}(k)\}$ **do**
        $T^* \leftarrow T(0, 0, 0, t_x, t_y, t_z)$ such that $C_{g_i}(j)$ and $C_{g'}(k)$ match
        $L_G(g') \leftarrow L_G(g') \cup \{(i, \{k \rightarrow j\}, T^*)\}$
      end for
    end for
    return $L_G(g')$
  end if
  $(v_{\text{old}}, v_{\text{new}}, e^*) \leftarrow g' \setminus g$ ▷ Newly added vertex or edge
  for $(i, m, T) \in L_G(g)$ **do** ▷ For all current matches, $m : V_g \rightarrow V_{g_i}$
    $Q \leftarrow \{(m(v_{\text{old}}), j) \in E^{g_i} : \mathcal{L}^{E_{g_i}}(m(v_{\text{old}}), j) = \mathcal{L}^{E_{g'}}(v_{\text{old}}, v_{\text{new}})\}$
    ▷ Set of possible extension edges in g_i whose label matches e^*.
    for $(k_1, k_2) \in Q$ **do** ▷ For each edge, check geometric consistency
      $g_{\text{test}} \leftarrow (V_{g_i}(m^{-1}(V_{g'})), E_{g_i}(m^{-1}(E_{g'})), \mathcal{L}^{V_{g_i}(m^{-1}(V_{g'}))},$
        $\mathcal{L}^{E_{g_i}(m^{-1}(E_{g'}))}, C_{g_i}(m^{-1}(V_{g'})))$
      if g' is an edge-vertex extension over g **then**
        $m \leftarrow m \cup \{v_2 \rightarrow k_2\}$
      end if
      if $\exists\, T^* : \forall i = 1, \ldots, |V_{g_{\text{test}}}| : \|C^{V_{g_{\text{test}}}} - T^*(C^{V_{g'}})\| \leq \epsilon$ **then**
        $L_G(g') \leftarrow L_G(g') \cup \{(i, m, T^*)\}$ ▷ Geometry matched, add match
      end if
    end for
  end for
  return $L_G(g')$
end function

Theorem 1 (Polynomial delay) *For a geometric graph database G of N geometric graphs, a query graph $q \in \mathcal{G}$, a minimum support $s \geq 1$ and matching tolerance $\epsilon \geq 0$, the time between two successive calls to* REPORT *in line 8 is bounded by a polynomial in the dimensions of input data.*

Proof sketch. Let $M := \text{argmax}_i |V_{g_i}|$, $F := \text{argmax}_i |E_{g_i}|$. For coordinates in $\mathbb{R}^3$, any subgraph $g \subseteq_0 q$ with $|V_g| \geq 3$ with non co-linear vertex coordinates has a unique transformation T mapping g into q and for each g_i with $g \subseteq_\epsilon g_i$ there is a unique T mapping g into g_i. Therefore, for any such graph g, the number of matching locations into the database G can only decrease in case g is enlarged. Considering the number of variations for $|V_g| < 3$, it is easy to see that the location list always satisfies $|L_G(g)| \leq M^3 N$.

Consider the function FILTERLOCATION. There exist a number $K = \gamma(M)$ of BFGS iterations with $\gamma : \mathbb{N} \to \mathbb{N}$ polynomial, such that the geometry test in line 36 of the algorithm can be performed in $O(K)$ to the required accuracy.[4] Additionally, the number of extensions $|Q|$ is bounded by $O(F)$. Together with the invariant $|L_G(g)| \leq M^3 N$ the complexity of a single call to FILTERLOCATION is bounded by $O(M^3 NFK)$.

Consider the function MINE. The mapping $f_q^{-1}(g)$ can be produced in $O(MF)$ using Tarjan's algorithm for enumerating cycle edges and cutting edges (bridges), which has $O(V + E)$ linear time for a graph of V vertices and E edges [16].

The time complexity between two successive calls to REPORT can now be bounded by considering two cases after REPORT has been called once.

- Case 1. There is an extension g' fulfilling the minimum support condition. Then REPORT is called within $O(M^4 NF^2 K)$ time.

- Case 2. There is no extension g' fulfilling the minimum support condition. Then, no recursion happens and MINE *returns* in $O(M^4 NF^2 K)$ time to its parent node in the reverse search tree. The maximum number of times this can happen successively is bounded by the depth of the reverse search tree, which is bounded by $O(F)$, because each level – except for the first – adds one edge. Therefore, in $O(M^4 NF^3 K)$ time the algorithm either calls REPORT again or finishes.

Thus, the total time between two successive calls to REPORT is bounded by $O(M^4 NF^3 K)$. □

Theorem 2 (Output polynomial time) *For a geometric graph database G of N geometric graphs, a query graph $q \in \mathcal{G}$, a minimum support $s \geq 1$ and matching tolerance $\epsilon \geq 0$, let R be the number of frequent geometric subgraphs reported by* FREQGEO. *Then, the total computation time for* FREQGEO *is bounded by a polynomial in the dimensions of input and output data.*

Proof. From polynomial delay and R frequent geometric graphs outputted it follows immediately that $O(M^4 NF^3 KR)$ bounds the total time complexity. □

3 Experiments and Results

We will now evaluate the proposed algorithm in three experiments. As data sets we use chemical compound data sets for structure-activity relationship (SAR) prediction of Sutherland, O'Brien and Weaver [17]. The data sets contain specific molecules; the BZR data set contains ligands for the benzodiazepine receptor, the COX data set contains cyclooxygenase-2 inhibitors, the DHFR data set contains inhibitors of dihydrofolate reductase and the ER data set contains estrogen receptor ligands. Besides containing related molecules, the data set additionally comes with activity labels. Not all of the samples present in the data sets are labeled; the labeled samples have a binary label, being either -1 or 1. Table 1 shows the number of molecules in the different data sets as well as the number of labeled samples in the provided training and testing splits. In the first experiment we assess empirically the runtime and number of retrieved subgraphs of our algorithm and compare it to gSpan-retrieval, a modified version of the popular gSpan graph mining algorithm [1]. The second experiment is concerned with the discriminativeness of the geometric subgraph features. In the third experiment we show a retrieval application in which similar compounds are retrieved using geometric subgraph features, where similarity is measured by means of the χ^2-distance on geometric subgraph feature histograms.

[4] BFGS has super-linear, sub-quadratic convergence.

Dataset name	Number of Chemicals	Training samples	Test samples
BZR 3D	405	181	125
COX2 3D	467	178	125
DHFR 3D	756	233	160
ER 3D	1009	266	180

Table 1: Datasets used in the experiments.

3.1 Experiment 1: Runtime

To assess the empirical runtime of our proposed algorithm we conduct the following experiment. We select ten random chemicals from the 405 molecules of the BZR 3D database. For minimum supports in the range 20 to 405 we perform subgraph retrieval using FreqGeo and gSpan-retrieval, a modified version of the open-source gSpan implementation available in the gboost toolbox[5]. We modify gSpan such that at each extension step an additional pruning condition is checked for: the current frequent subgraph must appear also in a given query graph. Thus, the two algorithms perform the same task, except that FREQGEO requires the 3D geometry to match, while the modified gSpan retrieval algorithm does not use 3D geometry at all. We would expect a somewhat better performance by modifying a more recent graph mining algorithm such as a recent version of GASTON [18, 19], but the simplicity of the gSpan enumeration allows a more straightforward modification. The runtime over all ten query molecules is averaged.

Regarding the runtime, we expect a tradeoff between the following two effects in FREQGEO geometric subgraph retrieval.

- More expensive extension and matching due to the 3D geometry check.

  In each node of the reverse search tree we extend our current subgraph and check that the extension matches under a 3D rigid transformation. The matching involves finding a minimizer of the non-linear optimization problem and is therefore more expensive to compute. In gSpan-retrieval no such check needs to be performed.

- More discriminative matching for larger subgraphs due to 3D geometry.

  The number of matching subgraphs with minimum support should increase slower in geometric subgraph retrieval than for gSpan-based subgraph retrieval. This is because in larger subgraphs, the geometry becomes the most effective pruning condition.

Figure 2 shows the averaged runtime for both gSpan-retrieval and FREQGEO subgraph retrieval, Figure 3 the number of frequent subgraphs found. We see that gSpan is faster by roughly two orders of magnitude for high values of minimum support and by roughly one order to magnitude for low minimum support. For a minimum support below 90, the used implementation of gSpan-retrieval starts to exhausts the main memory of our system for some of the query graphs, whereas we successfully use FREQGEO to mine down to a minimum support of 20. Additionally, while generally very fast, the runtime seems to increase super-exponentially for gSpan, whereas FREQGEO's runtime increases much slower. This might be explained by the steep increase in the number of frequent subgraphs for gSpan, as visible in Figure 3. For FREQGEO, the number of frequent geometric subgraphs increases much slower.

The smaller number of frequent patterns returned by FREQGEO provides for a better interpretability in practice: a small set of geometric subgraphs returned by our algorithm (say, around 1000 for a minimum support of 20) is more interpretable than a large one (around 30,000 for a minimum support of 90) of non-geometric frequent subgraphs returned by gSpan-retrieval. Thus, geometry aids interpretability.

One can ask whether a two-step procedure, where first gSpan-retrieval is run and then the resulting frequent subgraphs are explicitly filtered for geometric matches into the reference graph would be a possible alternative to our approach. The explicit geometric filtering step would not save computation for performing the geometric matching, on the contrary it would need to perform more geometric matching tests for all returned frequent subgraphs and localization sets because the subgraph-supergraph structure of the enumeration tree is not available during the geometric filtering step. Additionally, in the first step, the frequency during subgraph enumeration would always be measured in terms of non-geometric subgraphs. We have seen in Figure 3 that the frequency of non-geometric

[5] `http://www.kyb.tuebingen.mpg.de/bs/people/nowozin/gboost/`

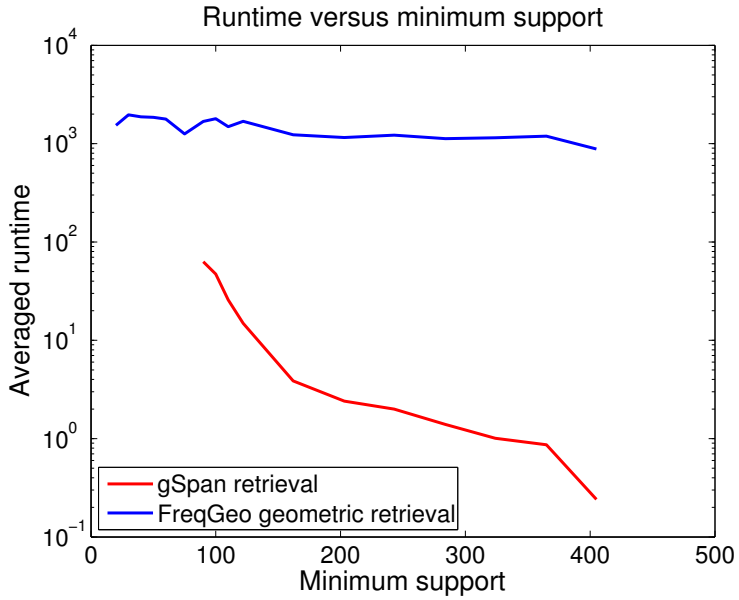

Figure 2: Runtime comparison between gSpan subgraph retrieval and FreqGeo geometric subgraph retrieval for different levels of minimum support. Note the log-scale. The plotted times are the mean over runs for ten different query graphs; the same query graphs have been used for both algorithms. Minimum support values below 90 exhaust more than 4GB main memory in the gSpan retrieval algorithm.

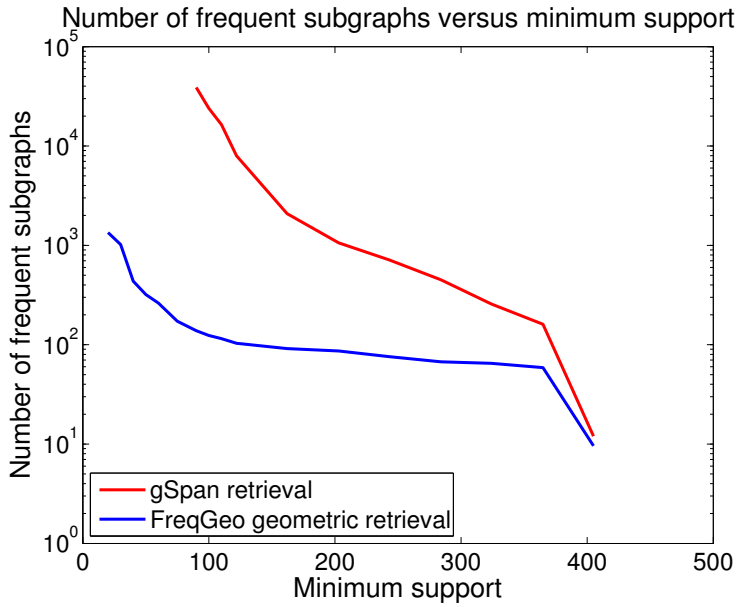

Figure 3: Number of frequent subgraphs returned by the retrieval version of gSpan and by the FreqGeo algorithm, averaged over ten query graphs. Note the log-scale.

subgraphs is orders of magnitudes larger than for the corresponding geometric frequency of the specific geometric subgraph in the reference graph. Therefore, many more nodes in the enumeration tree would have to be visited. Note that on an abstract level we essentially perform a two step generation/filtering procedure as well, but we are doing so in an *integrated* way in FREQGEO: we enumerate subgraphs structurally and filter them geometrically *at each node* of the enumeration tree. This allows us to exploit the geometric subgraph-supergraph ordering to prune most of the enumeration tree.

3.2 Experiment 2: Discriminative Power of Geometric Features

We have shown that the number of patterns is reduced dramatically by introducing the geometric constraints. However, it could be the case that important information is lost by pattern set reduction. To prove it is not the case, we perform simple supervised classification to compare our geometric patterns with non-geometric ones.

In principle, retrieval methods are not directly amenable to supervised learning, because the feature space created by patterns depends on a query graph. It would be possible to create a feature space by collecting patterns retrieved from several query graphs, but then the problem is how to select query graphs optimally. Although geometrical pattern mining for supervised learning is an interesting problem, we do not have a definite answer yet. For supervised classification based on non-geometric patterns, please refer to [5] and references therein. For 3D QSAR analysis, several prediction methods such as CoMFA have been proposed [20, 21]. In this experiment, we would like to measure the remaining information in our geometric patterns, instead of confronting state-of-the-art methods in accuracy.

For each of the chemical compound database in the previous experiment, the training and testing subsets are combined. On these four geometric graph sets we perform geometric subgraph retrieval on ten randomly selected chemical compounds from the training set of each database. Then, for each graph set, we use the binary class labels of only the training subset to train a naive Bayes classifier on the retrieved features. The trained classifier is used to predict the labels of the test sets. The same experiment is performed with the retrieval version of gSpan with identical settings but not making use of geometric information.

The subgraph retrieval, both for FREQGEO and gSpan-retrieval is performed using a minimum support of 75. For the naive Bayes classifier, we use the class conditional m-estimate of the probability of a graph s_j appearing as subgraph of a given test graph g as

$$p(s_j \subseteq_\epsilon g | y = 1) \approx \frac{\sum_{n=1}^{N} I(s_j \subseteq_\epsilon g_i) + mp}{\sum_{n=1}^{N} I(y_n = 1) + m},$$

where m is the *equivalent sample size* and p is the prior probability, see e.g. [22]. We use $m = 1$ and $p = \frac{1}{2}$ for all experiments. The probability estimate for $p(s_j \subseteq_\epsilon g | y = -1)$ is used analogously.

With these per-feature probability estimates we consider the naive Bayes classifier log-odds

$$\begin{aligned} \phi(g) &:= \log \frac{p(y=1|g)}{p(y=-1|g)} \\ &= \log \frac{p(y=1)}{p(y=-1)} + \sum_j \log \frac{p(s_j \subseteq_\epsilon g | y=1)}{p(s_j \subseteq_\epsilon g | y=-1)}. \end{aligned}$$

The classification decision is simply made by considering whether $\phi(g) > 0$. If it is, g is predicted to be in the positive class, otherwise its negative. To evaluate the predictive performance for the test set, we consider the ROC Area Under Curve (AUC) and ROC Equal Error Rate (EER) [23]. The results shown in Table 2 demonstrate that for some of the data sets using geometry improves the performance of our simple classifier. It implies that the information required in activity prediction is well preserved in our small geometric pattern set.

	FREQGEO		gSpan retrieval	
Dataset name	ROC AUC	ROC EER	ROC AUC	ROC EER
BZR 3D	**0.6585±0.026**	**0.6559±0.029**	0.6096±0.025	0.6125±0.024
COX2 3D	**0.7109±0.010**	**0.6325±0.012**	0.6952±0.029	0.6159±0.019
DHFR 3D	**0.7447±0.030**	**0.6732±0.030**	0.6868±0.051	0.6311±0.060
ER 3D	0.7241±0.065	0.6743±0.061	0.7222±0.064	0.6741±0.050

Table 2: Test set classification scores, averaged over ten molecules.

3.3 Experiment 3: Similar Geometric Subgraph Retrieval

As last experiment we assess the retrieval application. In this setting, a query graph is given to the algorithm and similar graphs should be retrieved from the database.

In order to rank the graphs from a database G by similarity to a query graph q, we perform frequent geometric subgraph retrieval on q in the database G and consider the histogram h^q and h^g of occurrences of the retrieved frequent subgraphs in q and $g \in G$. The χ^2-distance is a natural metric for these histogram representation, and we simply rank all database graphs by the χ^2-distance as

$$\chi^2(h^q, h^g) = \sum_{\{n: h_n^q + h_n^g > 0\}} \frac{(h_n^q - h_n^g)^2}{h_n^q + h_n^g},$$

where h_n is the number of occurrences of the n'th frequent geometric subgraph found. In Table 3 the top five matching graphs for a number of query graphs in the BZR 3D database are shown. We used a minimum support of 20 graphs. While subjective, the order and ranking scores clearly demonstrate the ability of retrieving similar chemical compounds, both by graph structure and geometry.

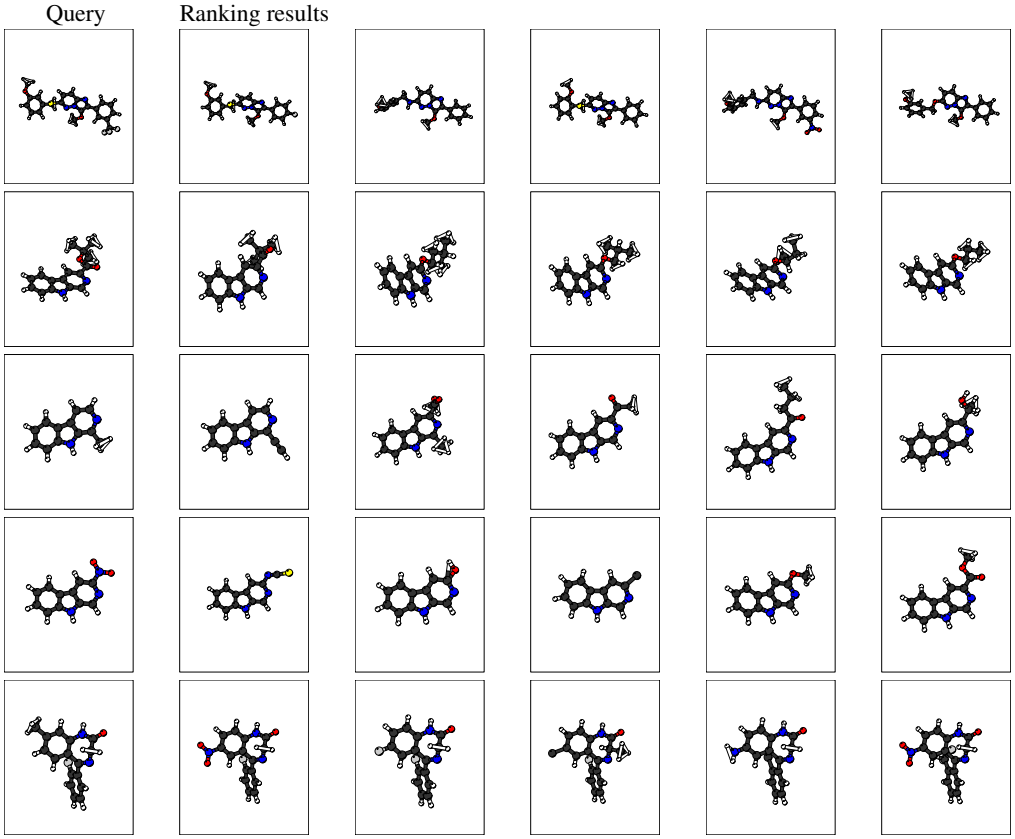

Table 3: Example rankings on the BZR 3D data set, for five query graphs.

4 Conclusion

We have proposed and experimentally evaluated an algorithm able to retrieve frequent geometric subgraph patterns. The retrieved subgraphs are allowed to match up to a prespecified geometric tolerance. We believe our contribution is a novel and clean method to robustly use geometry information for subgraph retrieval and will be especially useful in computational biology. Previous approaches require either exact matches and thus are not robust, or discretize or heuristically summarize geometric information into edge or vertex attributes, the effects of which on are hard to understand.

The algorithm proposed in this paper overcomes the *pattern ambiguity problem* of these previous approaches by changing the usual frequent substructure mining setting that makes geometric subgraph mining so difficult. Instead of asking for globally frequent geometric subgraphs in a graph database, we remove all ambiguities by establishing a single query graph as reference for geometric matching. This simplifies the problem, but it also makes sense when

statistics such as similarity to the query graph are the object of interest, for example in classification and retrieval applications where mining is used to generate meaningful features. We demonstrated scalability and usefulness for the application in chemical compound classification and retrieval.

Our implementation and the experiments are available as open-source at `http://www.kyb.mpg.de/bs/people/nowozin/freqgeo/`.

References

[1] Xifeng Yan and Jiawei Han. gspan: graph-based substructure pattern mining. In *Proc. IEEE International Conference on Data Mining ICDM 2002*, pages 721–724, 2002.

[2] Taku Kudo, Eisaku Maeda, and Yuji Matsumoto. An application of boosting to graph classification. In *NIPS*, 2004.

[3] Hiroto Saigo, Tadashi Kadowaki, and Koji Tsuda. A linear programming approach for molecular qsar analysis. In *MLG 2006*, 2006.

[4] Koji Tsuda and Taku Kudo. Clustering graphs by weighted substructure mining. In William W. Cohen and Andrew Moore, editors, *ICML*, volume 148 of *ACM International Conference Proceeding Series*, pages 953–960. ACM, 2006.

[5] Sebastian Nowozin, Koji Tsuda, Takeaki Uno, Taku Kudo, and Gökhan H. Bakır. Weighted substructure mining for image analysis. In *CVPR*. IEEE Computer Society, 2007.

[6] M. Kuramochi and G. Karypis. Discovering frequent geometric subgraphs. In *Proc. IEEE International Conference on Data Mining ICDM 2002*, pages 258–265, 2002.

[7] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. Knowledge and Data Engineering*, 17(8):1036–1050, 2005.

[8] Hiroki Arimura, Takeaki Uno, and Shinichi Shimozono. Time and space efficient discovery of maximal geometric subgraphs. In *Discovery Science 2007*, pages 42–55, 2007.

[9] Benjamin J Polacco and Patricia C Babbitt. Automated discovery of 3d motifs for protein function annotation. *Bioinformatics*, 22(6):723–730, 2006.

[10] R. B. Russell. Detection of protein three-dimensional side-chain patterns: new examples of convergent evolution. *J Mol Biol*, 279(5):1211–1227, 1998.

[11] L. Holm and C. Sander. Dali: a network tool for protein structure comparison. *Trends Biochem Sci*, 20(11):478–480, 1995.

[12] Jun Huan, Deepak Bandyopadhyay, Wei Wang, Jack Snoeyink, Jan Prins, and Alexander Tropsha. Comparing graph representations of protein structure for mining family-specific residue-based packing motifs. *J Comput Biol*, 12(6):657–671, 2005.

[13] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65:21–46, 1996.

[14] David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27:119–123, March 1988.

[15] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999. 2nd edition.

[16] Robert Sedgewick. *Algorithms in C, Part 5: Graph Algorithms, Third Edition*. Addison-Wesley, 2002. ISBN 0201316633.

[17] Jeffrey J. Sutherland, Lee A. O'Brien, and Donald F. Weaver. Spline-fitting with a genetic algorithm: A method for developing classification structure-activity relationships. *Journal of Chemical Information and Computer Sciences*, 43(6):1906–1915, 2003.

[18] Siegfried Nijssen and Joost N. Kok. The gaston tool for frequent subgraph mining. In *Proceedings of the International Workshop on Graph-Based Tools, Grabats 2004, Rome, Italy, October 2, 2004*. Elsevier, 2004.

[19] Jeroen Kazius, Siegfried Nijssen, Joost N. Kok, Thomas Bäck, and Adriaan P. Ijzerman. Substructure mining using elaborate chemical representation. *Journal of Chemical Information and Modeling*, 46(2):597–605, 2006.

[20] H. Hong, H. Fang, Q. Xie, R. Perkins, D.M. Sheehan, and W. Tong. Comparative molecular field analysis (CoMFA) model using a large diverse set of natural, synthetic and environmental chemicals for binding to the androgen receptor. *SAR and QSAR in Environmental Research*, 14(5-6):373–388, 2003.

[21] L.M. Shi, H. Fang, W. Tong, J. Wu, R. Perkins, and R.M. Blair. QSAR models using a large diverse set of estrogens. *J. Chem. Inf. Comput. Sci.*, 41:186–195, 2001.

[22] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, Boston, 1997.

[23] Trevor Hastie, Robert Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, July 2001.