Masterarbeit

Learning to Construct Graphs with Real Vector Attributes Using Graph Kernels

José Luis Licón Saláiz

November 6, 2015

Albert-Ludwigs-Universität Freiburg im Breisgau
Technische Fakultät
Institut für Informatik

Eingereichte Bachelorarbeit gemäß den Bestimmungen der Prüfungsordnung der Albert-Ludwigs-Universität Freiburg für den Studiengang Master of Science (M. Sc.) Informatik vom 2011

Bearbeitungszeitraum
06. 05. 2015 – 06. 11. 2015

Gutachter
Prof. Dr. Rolf Backofen
Dr. Frank Hutter

Betreuer
Dr. Fabrizio Costa

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Contents

Acknowledgements

Danksagungstext

Zusammenfassung

Cette nouvelle édition differe peur de la précédente. J'ai ajouté quelques pages sur les correspondances entre les points de deux surfaces, en particulier sur la théorie des surfaces applicables. D'autre part, pour ne pas augmenter les dimensions du volume, j'ai dû supprimer quelques paragraphes consacrés à des questions accesoires, comme la transcendance du nombre e, les intégrales pseudo-elliptiques, etc., qui, malgré leur intérêt, ne sont pas indispensables à un candidat à la licence.

J'adresse de nouveu mes sincères remerciements à M. René Gosse, qui a continué à me prêter son concours pour la correction des épreuves.

Abstract

Abstract/Summary text

1. Introduction

The present work deals with a problem that can be called *constructive learning*: given a set of data which are to be used as training data, produce new instances which share some characteristics with the original data.

Most importantly, the kind of data that will be addressed is structured data, that is, graphs. In contrast to the case of non-structured data, such as in drawing samples from a given probabilistic distribution, the generation of new instances of structured data is not so well-studied and is in principle more difficult. The difficulties arise from the fact that the added structure imposes restrictions on what kind of instances can be generated; this restrictions may be moreover domain-dependent in the sense that the rules to determine the feasibility of new structures will in general be different for different types of information. A graph that might be a valid representation of a protein might be meaningless as a representation of a crystalline structure, say.

The problem is complicated further when we take into account the fact that there are properties that we want the new structures to have. This will introduce further restrictions on which structures are feasible, and will introduce a new modelling hypothesis as well. This hypothesis is best described as the assertion that the structural and geometrical properties of a given graph will be decisive in defining whether it has or lacks the desired property. A canonical example of this can be seen in chemoinformatics, where one of the tasks of drug discovery is to produce models of the biochemical activity of molecules based on their spatial structure. This is quantified by defining measures of similarity between the structures and their respective sub-structures, which will be the motivation behind one of the central concepts in this work: that of a *graph kernel*. Kernel methods for graphs typically depend on spatial relations between pairs of nodes in each graph only, and not in the relations between larger subgraphs. Here lies the main contribution of this thesis: studying the extent to which these methods can be generalized to incorporate this additional information.

Another concept central to the approach presented here is that of a *graph grammar*, a set of production rules which will provide a means to abstract all the relations between substructures that are present in the data, and use these as guides in the creation of new structures. This concept will also need to be adjusted to accommodate the case in which the relational information between entire subgraphs is used in the modelling.

The core of the thesis is chapter 2, which encompasses all the necessary components in a solution to this problem under the name *graph sampling*. The general concept of a graph kernel is introduced, then a specific kernel is defined which will allow us to model graphs in such a way that their structural information can be incorporated into the model. A way to circumvent the Graph Isomorphism Problem will be presented, as well as a method to discretize the information present in a graph's nodes if it comes in the form of vectors. Having defined this, we will be able to define ways to encode the local structural properties of the graphs in their node labels, and use this in the construction of the graph grammar. With this we will then be able to obtain probability distributions over sets of graphs and use these in the generation of new instances.

An empirical evaluation of this is presented in chapter 3, where we will be able to measure the performance impact of using this structural information both in classification models for graphs and in the generation of new ones. The information used here will be drawn from biological assessment experiments, or bioassays for short, which will enable us to fit binary classification models to the molecule data sets (they are always divided in active and inactive molecules for each bioassay). Having these it will then be possible to induce the necessary grammars over the data, and then construct new instances and evaluate them by using them to retrain the classification models. Chapter 4 will close with some remarks about the results obtained and possibilities for further work in this direction.

2. Graph Sampling

2.1. Overview

This chapter defines the conceptual framework on which the graph sampling methods are based. Some elementary concepts from graph theory are introduced, followed by the definition of graph kernels. After defining a kernel function that will be used throughout the rest of the work, a general method of handling vector-valued information in a graph's node labels is described. Two kinds of such information are defined, which will be informative in constructing classification models over graphs. A grammar for graphs is then defined, and finally the sampling algorithm is built by bringing all these elements together.

2.2. Problem Description

Graphs can be used to represent objects from a number of domains in science and engineering in a concise fashion. A few examples: any kind of network can be naturally represented as a graph; probabilistic graphical models represent the structure of conditional dependence between random variables using directed graphs; in physics and chemistry, the structure of molecules can be represented as graphs, with the vertices representing atoms and edges the bonds between them.

Beyond this, there may be situations in which the ultimate matter of interest is not in the graphical representation itself, but in what information it may provide about the underlying entity it represents. An immediate example is taken from biochemistry: given a family of compounds, each represented by its corresponding molecular graph, can the information stored in these graphs be used to predict which compounds will be biochemically active and which ones will not? This is in fact possible, and the key idea is to transform the graphs into a vectorial representation which then allows the use of kernel functions and machine learning algorithms to estimate probability densities over the graphs themselves and produce classification models.

Example 2.2.1. Some more examples of graphs.

The next question, and the one this thesis is focused on, concerns the production of new structures which share the properties of the original ones. Recovering the

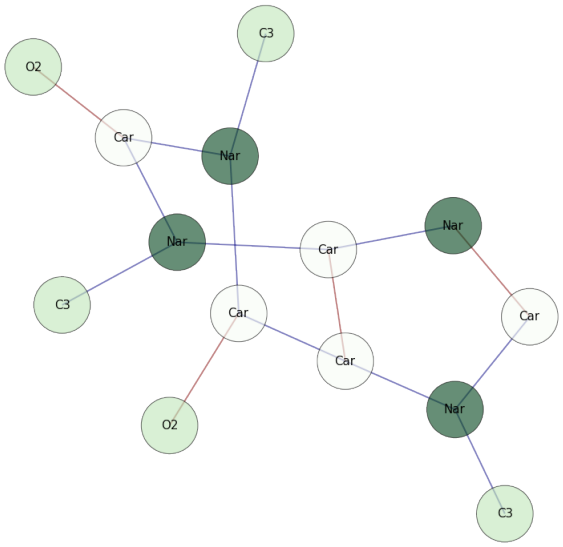

Figure 2.1.: Molecular structure of caffeine represented as a graph. Hydrogen atoms have been removed for clarity.

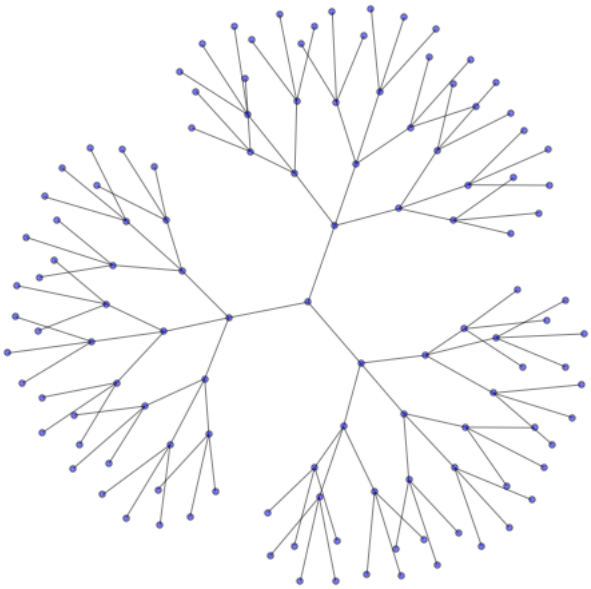

Figure 2.2.: A *tree*, a particular type of graph.

example from biochemistry described above, after classifying a set of molecules into active and non-active we might be interested in producing new molecules which could be classified as being active based on their structural properties, yet are different from the ones seen before. It is precisely this process of generating new structures from the given information that will be referred to as *graph sampling*, in analogy to the "classical" concept of sampling, understood as the drawing of random values from a given probability distribution. In Figure 2.3 we can see an example of this, where a histogram summarizes 1000 values drawn from a Poisson distribution.

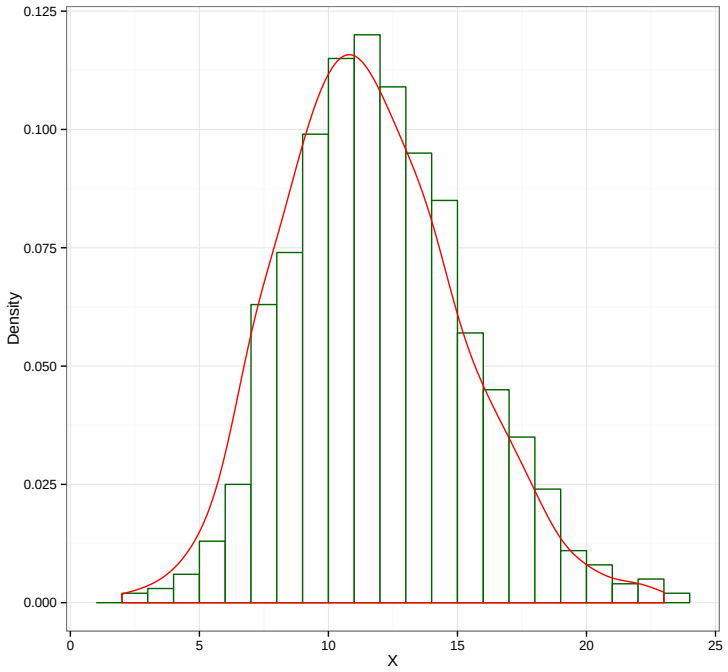

Figure 2.3.: Histogram showing 1000 draws sampled from a predictive posterior distribution, represented by the red line.

Among the numerous techniques for sampling which exist, the Metropolis-Hastings algorithm is a praticularly well-known method, which we will briefly review here due to its relevance to the graph sampling algorithm that will be developed later on. A more in-depth presentation can be found e.g. in [CG95].

The Metropolis-Hastings algorithm is an instance of a Markov Chain Monte Carlo sampler, so called because given an arbitrary probability density $f(x)$, these methods will construct an irreducible, aperiodic Markov chain that has f as its stationary distribution. The way Metropolis-Hastings does this can be summarized as follows: let $f(x)$ be the density being sampled from, and x^0 an initial value. Furthermore, we assume we have another probability density p from which we can generate candidate values, and which depends on the current state of the process,

i.e. $p = p(x, x^t)$. This is called the *proposal distribution*. The procedure is described in 1.

Algorithm 1 Metropolis-Hastings sampling algorithm

Input: x^0: Starting point
$f(x)$: target distribution
$p(x, x^t)$: proposal distribution N: maximum number of iterations

1: **for** $t = 1, \ldots, N$ **do**
2: Generate a value $x*$ from $p(x^{t-1}, \cdot)$.
3: Generate u from a uniform distribution $U(0, 1)$.
4: Compute the ratio

$$R = \frac{f(x^*)p(x^*, x^{t-1})}{f(x^{t-1})p(x^{t-1}, x^*)}$$

5: Compute the acceptance probability, $\alpha = \min\{R, 1\}$
6: The next value of the process is then given by

$$x^{t+1} = \begin{cases} x^* & \text{if } \alpha \geq u \\ x^t & \text{otherwise.} \end{cases}$$

7: **end for**

Output: $(x^t)_{t=1}^N$ will then be a realization of a Markov chain with a stationary distribution matching $f(x)$.

Three important observations can be made at this point. First, by virtue of the definition of the ratio R, we can either provide the target density itself, f, or a density that is proportional to it, $F = cf(x)$ for some constant c. Second, the algorithm is fully specified by the proposal distribution. Third, the proposal distribution need not be symmetric, but if it is then the acceptance probability reduces to

$$\alpha = \min\left\{\frac{f(x^*)}{f(x^{t-1})}, 1\right\}.$$

What this means is that if $f(x^*) \geq f(x^{t-1})$, then the chain will certainly move to x^+, otherwise it only moves with probability $f(x^*)/f(x^{t-1})$.

2.3. Graph Kernels

Before presenting the mathematical machinery necessary to elucidate the problem just described it is necessary to give a formal definition to some concepts. Most of these are elementary concepts in graph theory and can be consulted in any reference book, for instance the work by Gross and Yellen ([GY03]).

Definition 2.3.1. A *graph* G is defined by two sets, V and E, and denoted by $G = (V,E)$. The elements of V will be called *nodes* or *vertices*, while the elements of E will be called *edges*.

Definition 2.3.2. Let G be a graph. Then:

- G is a *rooted graph* if one of its vertices, v, is defined as its root. The graph is then denoted by G^v.
- The *distance between two vertices* u, $v \in V(G)$, denoted by $d(u,v)$ is the length of the shortest path between them.
- The *neighborhood of radius* r *of a vertex* v is defined as $N_r(v) = \{u \in V(G) \mid d(u,v) \leq r\}$.
- If $W = \{w_1, \ldots, w_k\} \subset V(G)$, the *subgraph induced by* W is the graph with W as its vertex set, and which contains every edge of G with endpoints in W.
- For $v \in V(G)$, the *neighborhood subgraph* of radius r of v is the subgraph induced by $N_r(v)$. It is denoted by $\mathcal{N}_r^v$.
- G is a *labeled graph* if its vertices and/or edges are labeled by symbols from a finite set Σ.
- Two graphs $G_1 = (V_1,E_1)$, $G_2 = (V_2,E_2)$ are *isomorphic* if there exists a bijection $\phi V_1 \to V_2$ such that there is an edge $uv \in E_1$ if and only if there is an edge $\phi(u)\phi(v) \in E_2$, and the label information is preserved.
- A *graph invariant* is a graph property that is identical for two isomorphic graphs.

The concept of kernel that will be used in this work is the same as in current machine learning literature, but it has its roots in N. Aronszajn's seminal 1950 work, *Theory of Reproducing Kernels* (cf. [Aro50]). A more modern overview of the concepts is given in [HSS08].

Definition 2.3.3. Let X be a set, and $K : X \times X \to \mathbb{R}$ a mapping. K is said to be a *(positive semidefinite) kernel* if it fulfils the following conditions:

- $K(x,y) = K(y,x)$ for any pair $x,y \in X$ (symmetry).
- For any $x_1, \ldots, x_n \in X$, the matrix defined by $(K_{ij}) = K(x_i,x_j$ is positive semidefinite.

Example 2.3.1. Let x, $y \in \mathbb{R}^n$. Then the following functions are positive semidefinite kernels:

1. $K(x,y) = x^T y + c$, the *linear kernel*.
2. $K(x,y) = \exp -\left(\frac{\|x-y\|^2}{\sigma^2}\right)$, the *Gaussian kernel*. ($\sigma > 0$)
3. $K(x,y) = (\alpha x^T y + 1)^d$, the *polynomial kernel*. ($\alpha \in \mathbb{R}$, $d \in \mathbb{N}$)

For a given function $\phi : X \to \mathbb{R}$, if K can be represented as the Euclidean inner product, $K(x,y) = \langle \phi(x), \phi(y) \rangle$, then K is a kernel function. It can be shown that under certain conditions, a given kernel function K can be represented by a certain function ϕ (cf. [Aro50] for details). The function ϕ is called the *feature map* of K, and the vector space induced by it is then called the *feature space* of K.

New kernels can be constructed from existing ones, in particular we have the following

Proposition 2.3.1. *If K and K' are kernels, their sum $K + K'$ is also a kernel.*

Definition 2.3.4. If $E \subset X$, then a kernel $K : E \times E \to \mathbb{R}$ induces a new kernel, called its *zero-extension*:

$$K_0(x,y) = \begin{cases} K(x,y) & \text{if } x,y \in E \\ 0 & \text{if } x \notin E \text{ or } y \notin E \end{cases}$$

Consider now the case where $x \in X$ is a composite object, and $x_i \in X_i$ are its component parts, for $i = 1,\dots,D$. We assume that the X_i are countable sets. Consider furthermore the relation R on the sets $X_1,\dots,X_D,X$ defined by $R(x_1,x_2,\dots,x_D,x)$ if and only if $x_1,\dots,x_D$ are the component parts of x. With this we define the function R^{-1} by

$$R^{-1}(x) = \{(x_1,\dots,x_D) \mid R(x_1,\dots,x_D,x)\}.$$

Definition 2.3.5. Let $x,y \in X$, with $\bar{x} = (x_1,\dots,x_D)$, $\bar{y} = (y_1,\dots,y_D)$ decompositions of x and y respectively. If K_i is a kernel function on X_i for $i = 1,\dots,D$, consider the following symmetric function:

$$K(x,y) = \sum_{\substack{\bar{x} \in R^{-1}(x) \\ \bar{y} \in R^{-1}(y)}} \prod_{i=1}^{D} K_i(x_i,y_i),$$

defined on $E = \{x \in X \mid R^{-1}(x) \neq \varnothing\} \subset X$. The *R-convolution* of $K_1,\dots,K_D$, denoted by $K_1 * \cdots * K_D$ is then defined as the zero-extension of K to the entire set X.

Theorem 2.3.2. *If each of the K_i is a kernel on X_i and R is a finite relation on $X_1,\dots,X_D,X$, then $X_1 * \cdots X_D$ is a kernel on X. This kind of kernel function is called a* convolution *or* decomposition *kernel.*

Proof. See Theorem 1 in [Hau99]. □

We will now make use of the concept of decomposition kernel to define a specific kernel that can be applied to graphs. This function, called the *Neighborhood Subgraph Pairwise Distance Kernel* (NSPDK), was defined in the paper [CG10] by Costa and de Grave, and the presentation of the concepts here closely follows the original one.

Definition 2.3.6. The *exact matching kernel* for graphs, δ, is defined by

$$\delta(x,y) = \begin{cases} 1 & \text{if } x \text{ and } y \text{ are isomorphic} \\ 0 & \text{otherwise.} \end{cases}$$

Consider a graph G and two rooted graphs A^v, B^u. Define the relation $R_{r,d}$ between the three graphs as being true if and only if A^v, $B^u \in \{N_r^v \mid v \in V(G)\}$, i.e., if both rooted graphs are (isomorphisms of) neighborhood subgraphs for some vertices u, v of G such that $D(u,v) = d$.

We can then define the decomposition kernel $\kappa_{r,d}$ over the graph G as the kernel induced by the relation $R_{r,d}$ defined above:

$$\kappa_{r,d}(G,G') = \sum_{\substack{A_v,B_u \in R_{r,d}^{-1}(G) \\ A'_v,B'_u \in R_{r,d}^{-1}(G')}} \delta(A_v,A'_{v'})\delta(B_u,B'_{u'}).$$

This kernel can be normalised as follows:

$$\hat{\kappa}(G,G') = \frac{\kappa_{r,d}(G,G')}{\sqrt{\kappa_{r,d}(G,G)\kappa_{r,d}(G',G')}},$$

and thus we arrive at the desired kernel:

Definition 2.3.7. Let G, G' be two graphs, and r^*, d^* two pre-defined values for radius and distance. Then the Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) is given by

$$K(G,G') = \sum_{r=1}^{r^*} \sum_{d=0}^{d^*} \hat{\kappa}_{r,d}(G,G').$$

A key feature of this kernel is the exact matching kernel over two graphs, δ. Obtaining its exact value is equivalent to solving the graph isomorphism problem, which involves finding a reasonable (i.e., polynomial-bound) algorithm for determining exactly when two graphs are isomorphic. This problem is, at the moment of writing these lines, still unsolved and has been deemed intractable, so an approximate solution is needed: the one used here comes in the form of *graph invariants* and will be explained more in-depth in the following section. A good overview of the graph isomorphism problem itself, as well as different approaches to an approximate solution to it, is given by Read and Corneil in [RC77].

2.4. Graph-invariant Encodings

The task is now to provide an approximation to the value of $\delta(G,G')$ for two arbitrary graphs G and G'. Again following the presentation given in [CG10] we will lay down the general principles of an encoding, which will be extended in the remainder of this work.

The approximate implementation the exact matching kernel has two main parts:

1. Compute a time-efficient graph-invariant encoding via a label function.
2. Compare the graph labels via a hash function.

This reduces the isomorphism test between two graphs to a comparison between integer numbers, with the following caveat: it is possible that two non-isomorphic graphs will be assigned the same identifier by using this procedure.

For the first step, the label function, we begin by defining label functions for nodes and edges, L^n and L^e. For a given rooted graph G_h, the function L^n assigns to a node $v \in V(G_h)$ the concatenation of the lexicographically sorted list of distance-label pairs, $\{(D(v,u),L(u)) \mid u \in G_h\}$, augmented with the distance from v to the root node h, $D(v,h)$. Based on this the edge label function L^e assigns to an edge uv the label $(L^n(u),L^n(v),L(uv))$. Finally the graph encoding for the graph, $L^g(G_h)$, is given by the concatenation of the lexicographically sorted list of the newly formed edge labels, $\{L^e(uv) \mid uv \in E(G_h)\}$.

The second step consists of using a Markle-Damgård hashing function to map the resulting string $L^g(G_h)$ to a 32-bit integer value (cf. [Dam90] for details on this). Using this the features calculated by the NSPDK function can be encoded explicitly: a feature is in this case a pair of rooted neighborhood graphs, say A and B, the roots of which are at a given distance d. These graphs will be encoded by the procedure just described, resulting in the pseudo-identifiers $H(\mathscr{L}^g A)$ and $H(\mathscr{L}^g B)$. We can then represent the feature by the triplet $(d,H(\mathscr{L}^g A),H(\mathscr{L}^g B))$, and the pseudo-identifier for the feature will be the hash value of this triplet. This will allow us to work directly on sparse vector representations.

For this procedure to be applicable, a fundamental property is required of the vertex labels: they are assumed to be of a discrete nature. More precisely, it is assumed that for each node $v \in G_h$, the value of $L^n(v)$ will always be an element of a finite "alphabet". The graph-invariant encoding obtained in this fashion will be called *discrete encoding* in the sequel.

2.5. The case of vector labels

As presented above, the graph-invariant encoding being discussed here has at least one aspect that is readily susceptible of generalisation: the initial vertex labels

$L(v)$. These labels, as handled by the algorithm just described, do not necessarily contribute any information about the structure of the graph beyond the individual identifier of each node. Consider the following examples:

- In the "natural" representation of a molecule where each node represents an atom, the node labels will give a specific piece of information about each specific atom, such as its atom type, its atomic number, its charge, and so forth.
- A graph representing a social network will have one node representing an individual in the network. The node labels will then contain specific information about this individual, such as name, address, age, nationality, to name but a few.

This brings us to one of the central questions explored in the development of this thesis: what kind of information can be used in the node labels to inform a kernel function about the local structure of the graphs being studied? The ultimate objective of this is to test the hypothesis that, since the kernel function will induce a similarity measure, the accuracy of a classification algorithm will improve with this new annotation in the graph's nodes.

In this section we will work with vector-labelled graphs, that is, where every vertex $v \in V(G)$ will be assigned a label $L^n(v) \in \mathbb{R}^d$, for some value of d. We will see how such labels can be produced in a meaningful manner, that is, in a way that they represent information about local structure that can be used by the graph kernel. Before that, however, we will need to define a procedure that converts these labels into something that is actually usable by the graph-invariant encoding function defined before. In other words, what is needed is a discretization procedure.

Algorithm 2 Vector label discretizer

Input: $\mathscr{G}$: Set of vector-labelled graphs
n: maximum number of clusters
label_size: number of discretization steps

1: discretization_models ← empty list
2: n_clusters ← logarithmically spaced sequence between $\log_{10} s$ and $\log_{10} n$
3: M ← matrix with all the label data from all graphs
4: **for** $c \in$ n_clusters **do**
5: model_c ← a K-means clustering model trained on the data in M, with c clusters
6: Append model_c to discretization_models.
7: **end for**

Output: A list of discretization models, with as many elements as required by *label_size*.

2.5.1. Metric Encoding

Having defined a way to discretize the vector labels of a graph, we now turn to the question of what information it is actually useful to encode in the vertex labels, starting from the base case in which these are all discrete. To this end let us start with the following assumptions about a graph G:

- The nodes of G can be embedded in n-dimensional Euclidean space. In particular, for each $v \in V(G)$ we have its *coordinate vector*, which we represent by $(v_1, \ldots, v_n)$.
- The node labels of G are all elements of a finite set, Σ.

Having this, the idea behind this encoding is the following: given an integer parameter k, every node of a graph G will have as label a vector that represents the distance, or a monotone function of it, of the node v to the k nearest nodes; to make this more specific the k nearest nodes are selected not from the entire vertex set, but from each of the subsets of $V(G)$ defined by $V_i(G) = \{v \in V(G) \mid L^n(v) = i\}$, for all $i \in \Sigma$. This vector will then encode the vicinity of every vertex, as defined by its proximity to other vertices of every possible kind.

The function to generate this encoding is described in algorithm 3.

Example 2.5.1. An image representing the metric encoding for molecular graphs.

Algorithm 3 Metric encoding algorithm

function L_M(v, k, θ, β)
  let $s = |\Sigma|$, res $= \varnothing$
  $D \leftarrow$ array of pairwise distances of all $u \in V(G)$
  for each value $\sigma_i \in \Sigma$ **do**
    $V_{\sigma_i} \leftarrow \{u \in V(G) \mid L(u) = \sigma_i\}$
    $D \supset d_i \leftarrow$ all distances from v to elements of V_{σ_i}
    if $|V_{\sigma_i}| < k$ **then**
      $d_i \leftarrow d_i \cup (M, \cdots, M)$ such that $|d_i| = k$, for some $M \gg 1$
    end if
    for any $d \in d_i$ such that $d > \theta$ **do**
      $d \leftarrow \theta$
    end for
    $d_i \leftarrow \beta(d_i)$
    res $\leftarrow$ res $\cup\, d_i$
  end for
  return res
end function

2.5.2. Topological Encoding

The encoding presented in subsection 2.5.1 is not the only possible graph-invariant encoding that satisfies the requirements posed by the problem at hand. A second encoding was devised and tested for this thesis, called *topological* because it was inspired by recent work in applied topology, in particular by what is now called *persistent homology*.

We again require that, for any graph G being analysed, its nodes can be embedded in Euclidean space (although for the sake of generality, any metrizable topological space would do). The key observation here is that the metric structure of the graph defines a particular "'shape" for it, in a way that is independent of the information contained in its edges. Before making this more precise, we need to define an important concept:

Definition 2.5.1. If S is a discrete set, an *abstract simplicial complex* is a collection X made by finite subsets of S that is closed under restriction: any non-empty subset of X is again an element of X. For each $\sigma \in X$, if $|\sigma| = k + 1$, σ is called a *k-simplex*.

Example 2.5.2. A graph is an example of simplicial complex. Its vertices form a set of 0-simplices, while its edges form a set of 1-simplices.

Definition 2.5.2. Let $\epsilon > 0$, and $E \subset \mathbb{R}^n$ a discrete set. The *Vietoris-Rips complex* of scale ϵ on E, $VR_\epsilon(E)$, is the simplicial complex formed by the pairwise sets of points in E at a distance no larger than ϵ.

Although this concept is of central important to the modern field of computational topology, it really has its roots in the work of Leopold Vietoris in 1927 (cf. [Vie27]).

Example 2.5.3. A nice picture of a Vietoris-Rips complex.

One could legitimately ask: which is the "optimal" value of the ϵ parameter? A simple example shows why this question is not necessarily meaningful:

Example 2.5.4. Image showing the behaviour of the VR complex for increasing values of epsilon on the same data set.

Thus, a more pertinent question would be: which features revealed by the Vietoris-Rips complexes for a range of values of ϵ are essential to the structure of the data, and which are merely noise? This is precisely the question which the study of persistent homology aims to address, by constructing increasing sequences of Vietoris-Rips complexes, (VR_i), indexed by increasing values of the parameter ϵ_i, and associating topological features in the complexes to parameter intervals. The idea is then that the features which persist through larger intervals will be associated with essential topological features of the space in question. We do not pursue this matter further here and instead go back to the problem of determining a graph-invariant

encoding that captures local structural information from the graph itself. An reader interested in persistent homology can consult the available literature, notably the work of Carlsson ([Car09]) and Ghrist ([Ghr07]).

By looking again at the examples of the construction of a Vietoris-Rips complex, we can see that for each value of the ϵ parameter, each of the points in the space can be seen to posses some information about its immediate vicinity, and that this information can be obtained from the simplex or simplices in which the point is located. We could take, for instance, the size of the associated simplices, or their degree, or even the degree of the homological features associated with them. In this work we chose the first quantity, as it represented the least difficulties in its implementation. This is described in detail in algorithm 4.

Algorithm 4 Topological encoding algorithm

function L_T(v, D_{max}, n)
  $D \leftarrow$ array of pairwise distances of all $u \in V(G)$
  $\{\epsilon_i\} \leftarrow \left\{D_{max}\frac{i}{n} \mid i = 1,\ldots,n\right\}$
  $N \leftarrow |V(G)|$
  $l = \bar{0} \in \mathbb{R}^n$
  for each value ϵ_i **do**
    $P \leftarrow \{u \in V(G) \mid d(u,v) \leq \epsilon_i\}$
    $l_i \leftarrow \frac{|P|}{N}$
  end for
  return l
end function

2.6. Graph Grammar

The sampling approach proposed here is based in the Metropolis-Hastings algorithm, a Markov chain Monte Carlo method used to sample from probability densities which cannot be directly sampled from. This algorithm depends on a *proposal distribution* $g(x)$ used to generate new values, and an *acceptance distribution* $A(x)$ used to determine whether the generated values can be accepted as conforming to the desired distribution. In the present case it is necessary to adapt these concepts to the case of structured information in the form of graphs, and the first step toward this consists in recasting the problem into one of *grammatical inference*. The presentation given here, as well as in the next section, is based on the paper [Cos14].

Definition 2.6.1. A *core graph* $C_R^v(G)$ is a neighborhood graph of G, of radius R, rooted in v. An *interface graph* $I_{R,T}^v(G)$ is the difference graph of two neighborhood graphs, both rooted in v, and with radii $R-1$, $R+T$.

Definition 2.6.2. A core graph is said to be *corresponding to* an interface graph if it is the smallest neighborhood graph in the definition of the interface graph. *Congruent* cores share at least one corresponding interface (modulo isomorphism).

Definition 2.6.3. A *graph grammar* is a finite set of production rules. A production rule is a triple $S = (M, D, E)$ where

- M is the "mother" graph
- D is the "daughter" graph
- E is an embedding mechanism

In this case both mother and daughter graphs will be unions of cores and corresponding interfaces, where the interfaces are required to be isomorphic. The embedding mechanism will then be the isomorphism between them. The way a production rule is applied is to swap congruent cores, and rewire the resulting graph to the host graph.

Since we are confronted again with the task of identifying isomorphic graphs, the graph-invariant encodings described in the previous subsections will come into play again. In particular, the grammar will be stored as a data structure that contains all the congruent cores for a given interface. What will be actually stored are the integer pseudo-identifiers (hashed values) described at the end of section 2.4. The grammar itself is implemented as a function that takes as input a set of graphs, $\mathcal{G}$, and two parameters: a maximum radius $\hat{R}$ and a maximum thickness $\hat{T}$; in output it produces a mapping M from the set of interfaces to the set of cores. This mapping will indicate which cores are congruent with each of the interfaces.

This grammar will be used to estimate the proposal distribution in the sampling algorithm, in a way that will be described in the next section. A first version of this grammar is described in algorithm 5.

The question remains of how to adapt this algorithm for the case of graphs with vector labels. Two modifications are necessary: first, a label discretizer needs to be incorporated to the process. Second, the hash value for the core graphs will now also take into consideration the interface to which it is attached. In other words the core hash value will now be calculated for the union of core and interface graphs. Introducing the vector labels will greatly increase the number of different cores and interfaces, and by modifying the core hash value like this it is expected that the matches will be more context-sensitive. The modified algorithm is described in algorithm 6.

2.7. Graph Sampling

As described before, the sampling algorithm is based on the idea of the Metropolis-Hastings algorithm (cf. algorithm 1). We therefore need to specify a proposal distribution and a way to calculate the acceptance probability. We assume that we have a

Algorithm 5 Locally Substitutable Graph Grammar Induction

Input: $\mathscr{G}$: Set of graphs to induce the grammar
$\hat{R}$: maximum radius
$\hat{T}$: maximum thickness
Output: Map M : interface $\mapsto$ core

for $G \in \mathscr{G}$ **do**
  for $v \in V(G)$ **do**
    for $R \leftarrow 1$ to $\hat{R}$ with $R \leftarrow R+2$ **do**
      $c \leftarrow C^v_R(G)$
      $n_c \leftarrow$ PseudoIdentifier(c)
      for $T \leftarrow 2$ to $\hat{T}$ with $T \leftarrow T+2$ **do**
        $i \leftarrow I^v_{R,T}(G)$
        $n_i \leftarrow$ PseudoIdentifier(i)
        $M(n_i) \leftarrow n_c$
      end for
    end for
  end for
end for

Algorithm 6 Locally Substitutable Graph Grammar Induction - vector labels

Input: $\mathscr{G}$: Set of graphs to induce the grammar
$\hat{R}$: maximum radius
$\hat{T}$: maximum thickness
$\mathscr{D}$, a label discretizer
Output: Map M : interface $\mapsto$ core

for $G \in \mathscr{G}$ **do**
  Apply the discretizer $\mathscr{D}$ to G to obtain the new vertex labels.
  for $v \in V(G)$ **do**
    for $R \leftarrow 1$ to $\hat{R}$ with $R \leftarrow R+2$ **do**
      for $T \leftarrow 2$ to $\hat{T}$ with $T \leftarrow T+2$ **do**
        $c \leftarrow C^v_{R+T}(G)$
        $n_c \leftarrow$ PseudoIdentifier(c)
        $i \leftarrow I^v_{R,T}(G)$
        $n_i \leftarrow$ PseudoIdentifier(i)
        $M(n_i) \leftarrow n_c$
      end for
    end for
  end for
end for

set of seed graphs, $\mathcal{G}_S$, that will be used as initial conditions for the Markov chain; also required is another set of graphs $\mathcal{G}$ that will be used in the induction of a graph grammar.

We will also need a probability measure to be defined over the graphs being generated, which will allow us to calculate the acceptance probability for the sampling algorithm. We can do this using a scheme called *one-class-SVM* introduced by Schölkopf et al (cf. [SPST$^+$01]). The idea presented there is that, since we can operate directly on the feature representation of the data, it is possible to construct an support vector machine (SVM) classifier function f that separates most of the given data from the origin with maximum margin, thus defining a domain which contains most of the data points. With this, and the additional assumption that the distance of a point to this hyperplane is proportional to the actual probability density around it, we can then estimate the probability density values on the set of graphs.

The sampling algorithm works as follows: for each vertex v of a given seed graph G, the sampler will decompose the neighborhood of v into cores and interfaces and will draw congruent cores from the grammar. The probability of a core being drawn is proportional to its frequency in the data used to induce the grammar, if the graphs being processed have discrete node labels, or proportional to the core's score as calculated by the SVM function f. This gives us the proposal density p. The core obtained from the grammar is then swapped into the old graph, generating a new one, G'. Using f we can also estimate the probability for each of the two graphs, which we use in calculating the acceptance probability:

$$\alpha = \frac{f(G')p(G',G)}{f(G)p(G,G')}.$$

The proposal distribution is symmetric, however, so the probability of acceptance reduces to $\min\{1, f(G')/f(G)\}$. It thus becomes a question of whether G or G' has a greater distance from the separating hyperplane of the SVM model, and therefore a larger estimated probability.

Algorithm 7 describes the entire procedure. Two new elements here are the Substitute and Accept functions, which play the role of the proposal distribution and acceptance probability respectively, as seen in algorithm 1.

Algorithm 7 Metropolis-Hastings Graph Sampler

Input: $\mathcal{G}$: Set of graphs to induce the grammar
$\hat{R}$: max. radius
$\hat{T}$: max. thickness
$\mathcal{G}_S$: set of seed graphs
I_{max}: max. number of iterations
L^n: a graph-invariant encoding
$\mathcal{D}$: a discretizer for the node labels
Output: Set of novel graphs $\mathcal{G}_O$

function GRAPHLEARN($\mathcal{G}$, $\hat{R}$, $\hat{T}$, $\mathcal{G}_S$, I_{max}, L^n)
  for $G \in \mathcal{G}$ **do**
    Annotate the nodes of G with L^n
  end for
  if The graphs have vector-valued node labels **then**
    Apply the discretizer $\mathcal{D}$ to every graph $G \in \mathcal{G}$
  end if
  Induce the graph grammar and the one-class-SVM:
  $M \leftarrow$ local substitutable graph grammar
  $f \leftarrow$ one-class-SVM
  for $G_s \in \mathcal{G}_S$ **do**
    Initialize the working set for G_s:
    $\mathcal{G}_W \leftarrow G_s$
    for $t = 1, \ldots, I_{max}$ **do**
      for $G \in \mathcal{G}_W$ **do**
        for $v \in V(G)$, $R \leq \hat{R}$, $T \leq \hat{T}$ **do**
          $n_c \leftarrow C_R^v(G)$
          $n_i \leftarrow I_{R,T}^v(G)$
          for $c \in M[n_i]$ **do**
            $G' \leftarrow$ Substitute(G, v, c)
            if Accept(f, G, G') **then**
              $\mathcal{G}_W \leftarrow G'$
              $\mathcal{G}_W \setminus G$
            end if
          end for
        end for
      end for
    end for
    $\mathcal{G}_O \leftarrow \underset{G \in \mathcal{G}_W}{\arg\max}\, f(G)$
  end for
  return $\mathcal{G}_O$
end function

3. Experimental Setup

3.1. Overview

This chapter will present an empirical evaluation of the techniques previously described. The first part will present and compare the results of binary classification models for molecular data using the graph kernel from section 2.3, both with discrete and vector-valued labels. Other relevant information, such as runtimes and dependence on parameters, is also discussed. The second part of the chapter will compare the results produced by graph sampling algorithm, again for discrete and vector-labelled data, starting from several of the classification models that have been introduced.

3.2. Classification Models

3.2.1. Data Overview

The graph-invariant encodings and sampling algorithms were tested on information taken from the public PubChem bioassay databases. A total of 54 bioassays were considered, more detailed information about them can be consulted in Appendix A. The model fitting and evaluations were carried out using the Python package *Explicit Decomposition with Neighborhoods* (EDeN, cf. [CGS+15]).

3.2.2. Hyperparameter optimization

The first step was the training of a classification model on the available data, using three graph-invariant encodings:

1. *Discrete*, which uses the original data as node label (in this case the atom type).
2. *Metric*, which uses the encoding described in subsection 2.5.1.
3. *Topological*, which uses the encoding described in subsection 2.5.2.

It is important to point out that even though the same data, understood as distinct chemical compounds, was used in all three cases, the existence of conformational

isomers for practically all the compounds involved resulted in more data being used for the two encodings that depend on vector-valued data. A set of conformational isomers is, as per the IUPAC definition (cf. [IUP14]), a set of chemical compounds each with the same atomic constitution, but with a different arrangement of the atoms in space for each compound; these multiple arrangements differ from one another by rotations about formally single bonds. The number of such conformational isomers, or *conformers* for short, depends on the molecule, but up to 10 were used for each original compound. An equal atomic constitution yields the same graph representation for all conformers; however, since the spatial coordinates of the atoms will be different, we can expect that the vector labels used in the metric and topological encodings will also be different, thus yielding different data points.

Before training all models, it was required to perform a so-called hyperparameter optimization for the two vector encodings, which depend on the following parameters:

- *Metric*: k, the number of nearest nodes per category; and θ, the threshold distance beyond which nodes are not considered any more.
- *Topological*: D_{max}, the maximum distance to be explored; and n, the number of sampling intervals to be considered.

The result of this step were 162 classification models, one for each bioassay and encoding combination. For each bioassay, the data was randomly split into 70% train - 30% test. The training data was then used to find the best parameter values for each encoding, and the models thus selected were then evaluated (using the ROC measure) on the test data. Figure 3.1 shows a summarized version of the distribution of performance measures for all three encodings.

Figure 3.2 shows a scatterplot with the results for the discrete encoding plotted against both the metric and topological encodings, in different colours. The respective loess curves and a reference line at 45° are also displayed.

The average runtime for optimization of the models using the discrete encoding was 2.99 minutes; for the metric encoding it was 110.98 minutes; and for the topological encoding it was 114.74 minutes. The difference sounds staggering, but one must remember that the amount of data is also very different: the discrete-encoded models had on average 262 molecules each, while the vector-encoded models had to work with 2371 molecules on average. An overview of this can be seen in Figure 3.3. A linear model was fitted on each of the three cases, and the coefficients for the independent variable (no. of molecules) can be seen in Table 3.2

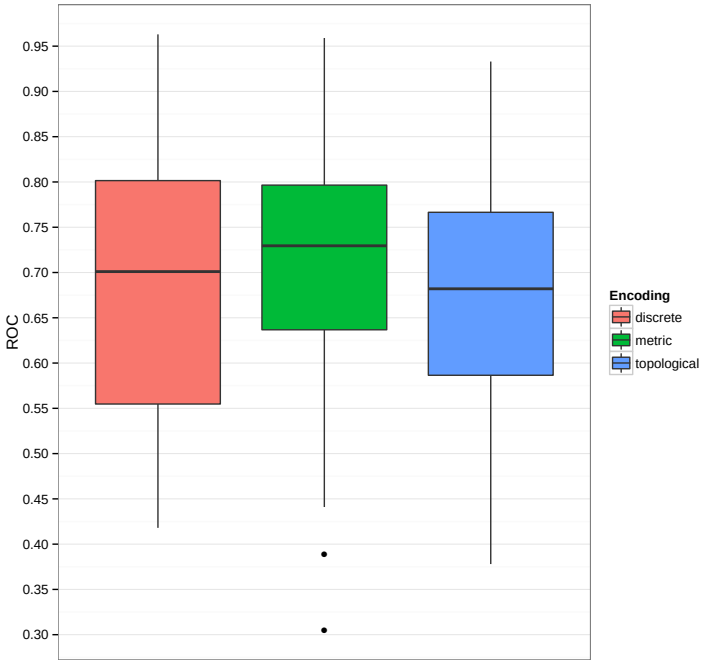

Figure 3.1.: ROC scores for the three encodings across the 54 datasets after performing hyperparameter optimization.

Table 3.1.: Coefficients for dataset size in the linear models with runtime as dependent variable.

Encoding	Coefficient	p-value
Discrete	0.006	2.21e-13
Metric	0.0431	2e-16
Topological	0.05235	2e-16

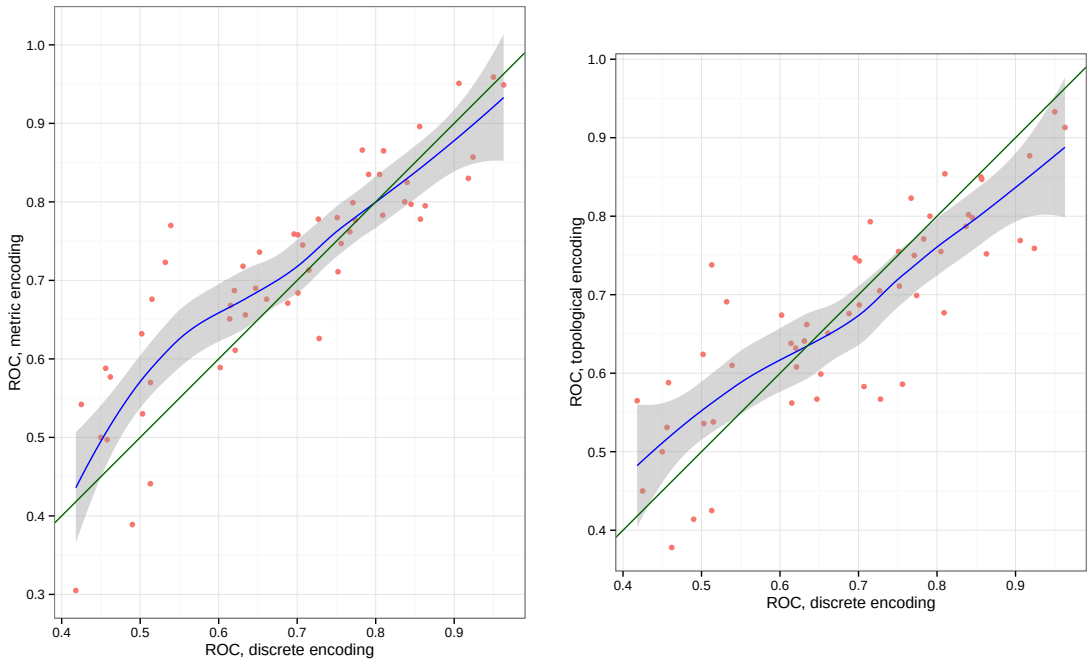

Figure 3.2.: Scatterplots showing the pairwise comparison of ROC scores between the three encodings. Also indicated is a loess curve for each, and a green reference line at a 45° angle.

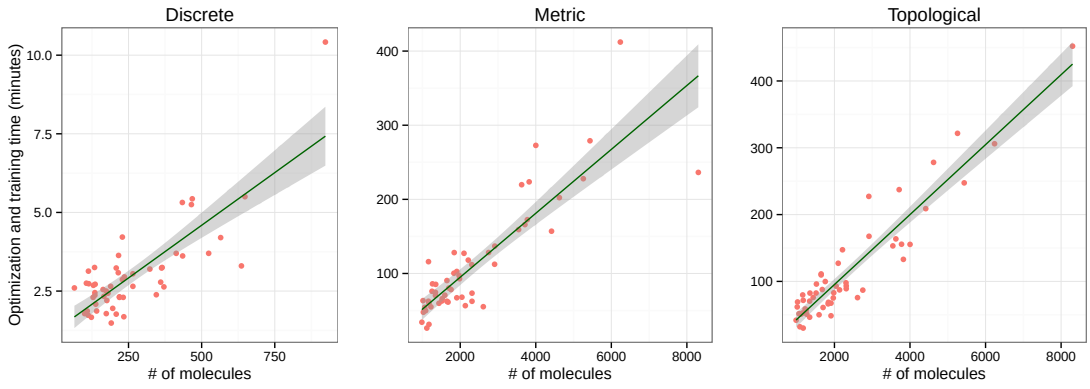

Figure 3.3.: Scatterplots showing the relations between dataset size in number of molecules and total runtime for the hyperparameter optimization. Also pictured on each panel is the least-squares regression line (green).

3.2.3. Cross-validation Estimates

After this each of the 54 models for each encoding was again fitted and evaluated on 10 different random splits, again using 70% of the data for training and 30% for testing.

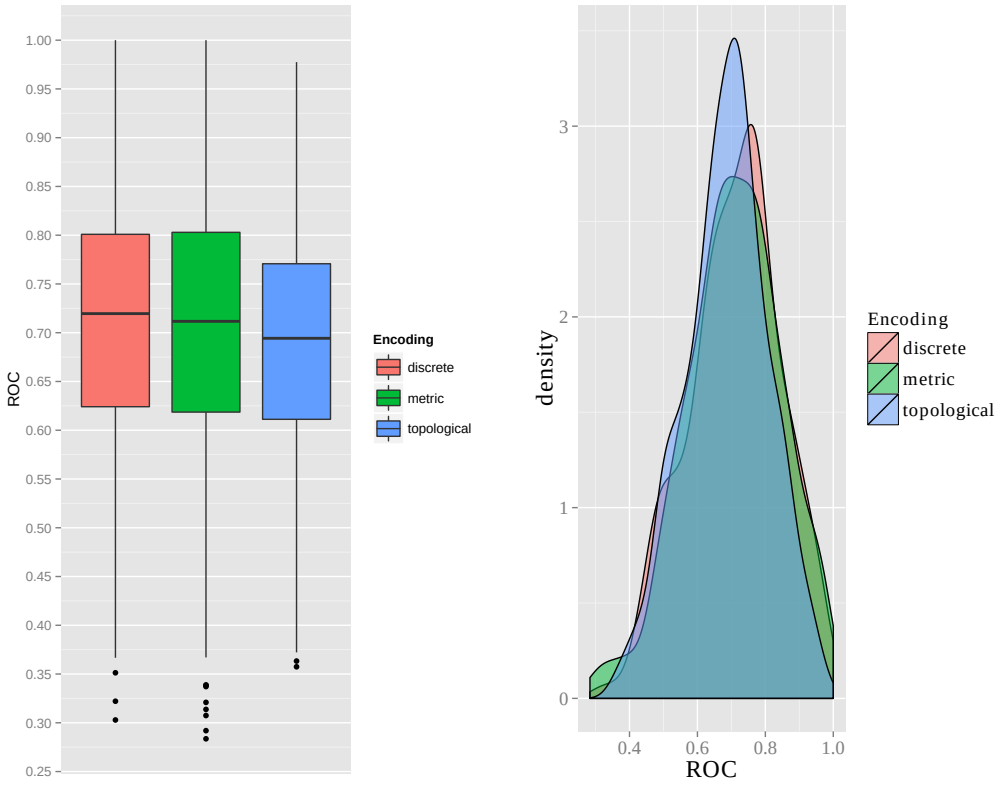

Figure 3.4.: A summary of cross-validation estimates. The left panel shows boxplots of all the repetitions over all datasets for each encoding; the right panel shows estimated density plots of the same information.

The models were then separated into three categories based on the mean performance registered by the model using the discrete encoding on every AID: the first group was formed by models with an average ROC not larger than 0.66; the second group included the models with an average ROC between 0.66 and 0.82; the final group included all those models with an average ROC greater than 0.82. Figure 3.4 shows a summary of these results, as boxplot and a density estimate.

The results for the three encodings are compared for every AID in Figure 3.5, Figure 3.6, and Figure 3.7.

Table 3.2.: Coefficients for dataset size in the linear models with runtime as dependent variable.

Encoding	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Discrete	0.302	0.624	0.719	0.709	0.801	1
Metric	0.287	0.618	0.711	0.707	0.803	1
Topological	0.357	0.611	0.694	0.687	0.770	0.97

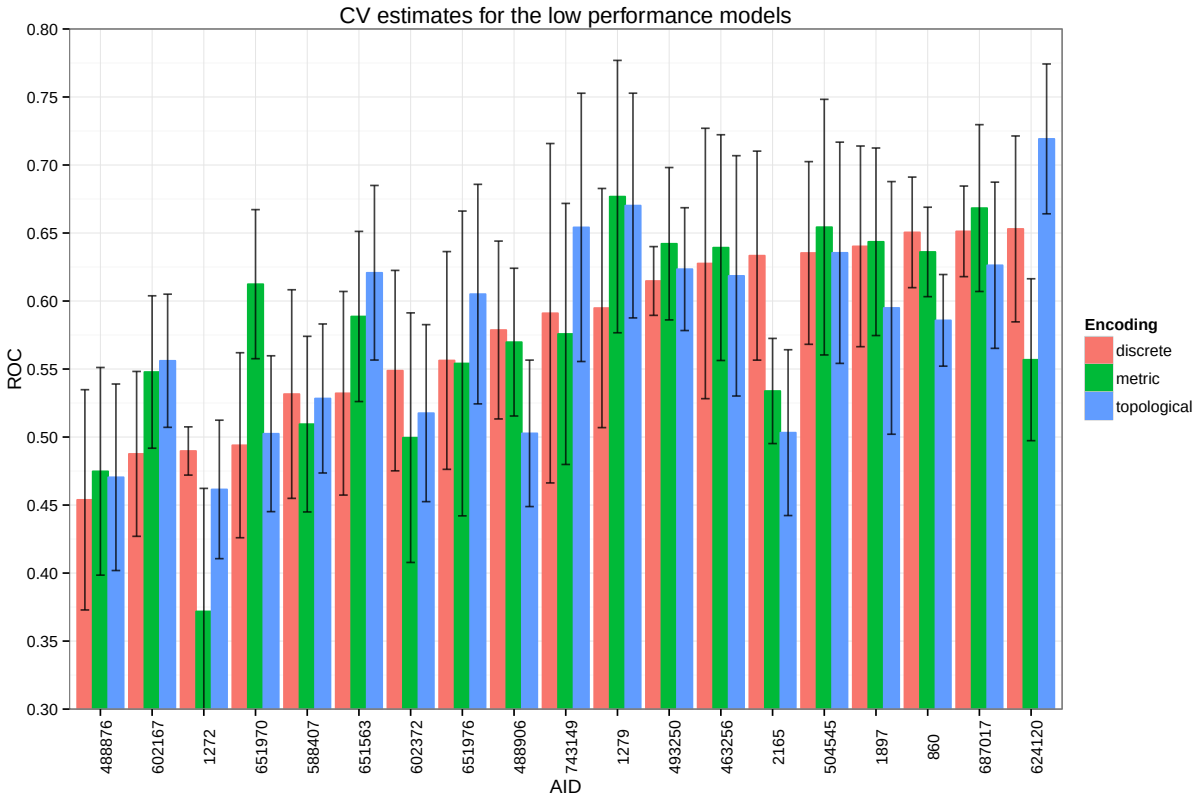

Figure 3.5.: Three-way comparison of ROC scores for all the low performance models, i.e., all those where the discrete encoding obtained a score under 0.66. The dots indicate the mean values, the horizontal bars indicate the mean ±1 standard deviation.

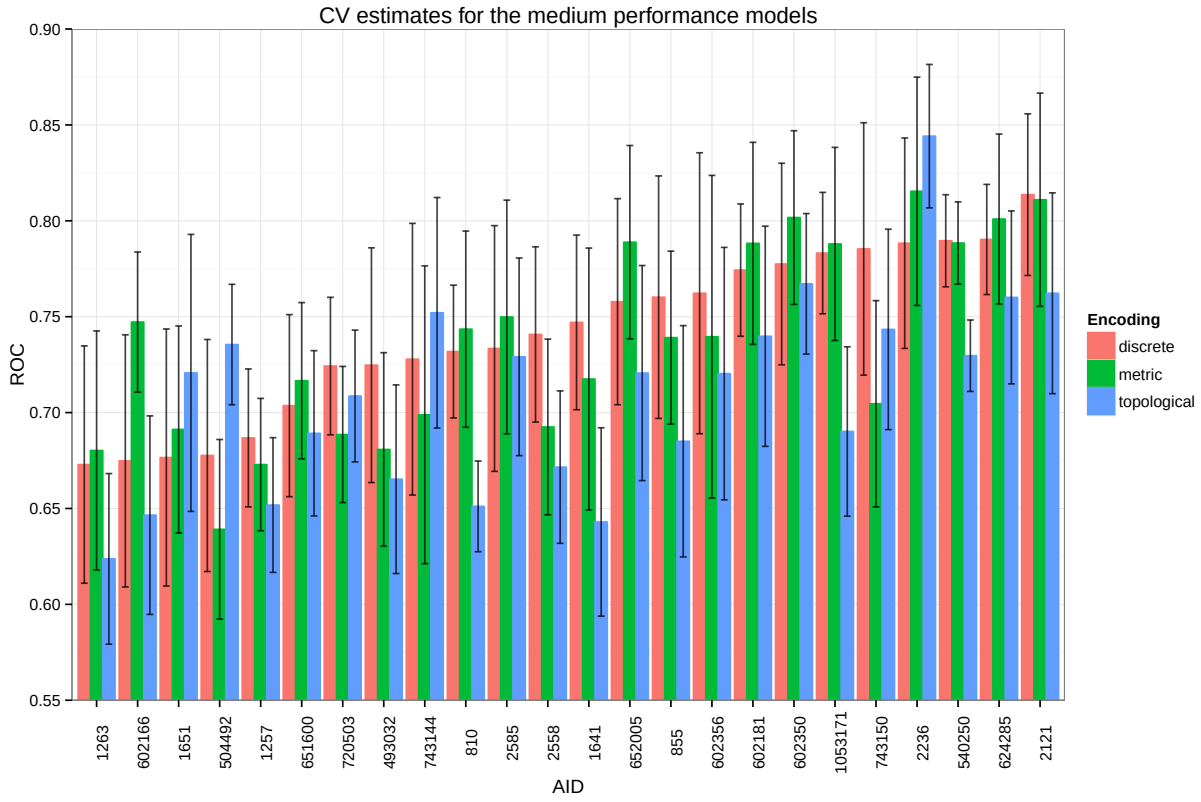

Figure 3.6.: Three-way comparison of ROC scores for all the medium performance models, i.e., all those where the discrete encoding obtained a score between 0.66 and 0.82. The dots indicate the mean values, the horizontal bars indicate the mean ± 1 standard deviation.

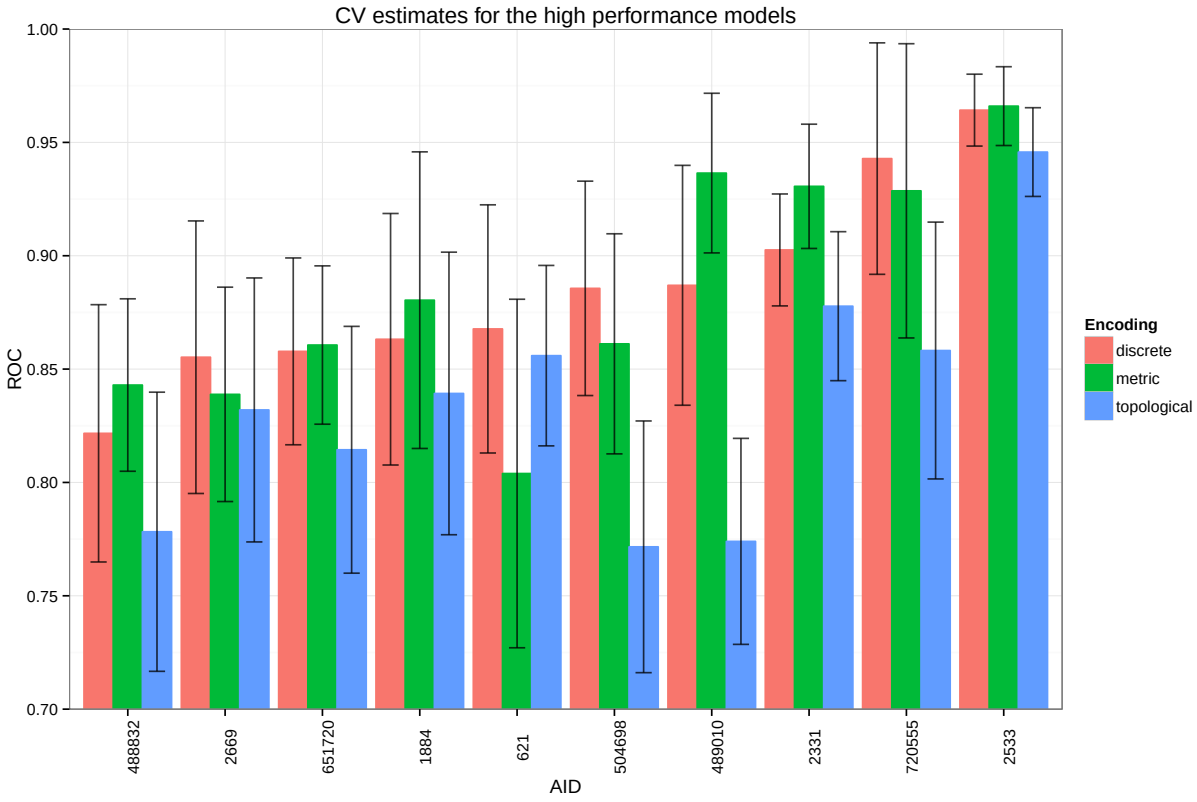

Figure 3.7.: Three-way comparison of ROC scores for all the high performance models, i.e., all those where the discrete encoding obtained a score of at least 0.82. The dots indicate the mean values, the horizontal bars indicate the mean ± 1 standard deviation.

At this point a comparison was also made between the performance of the discrete encoding and the two vector encodings, with respect to the different possible parameter values. These parameters belonged to one of two possible classes, the first of which encompasses the two parameters corresponding to the label discretization procedure, here called n and *label_size* (cf. algorithm 2). The second class corresponds to the parameters specific to each of the vector encodings: for the metric encoding the number of nearest neighbors k and the threshold distance θ (cf. algorithm 3); for the topological encoding the number of intervals n and the maximum distance D_{max} (cf. algorithm 4).

The first two parameters are shown in figures Figure 3.8 and Figure 3.9. The results for both encodings are shown simultaneously since both use the same discretizer parameters. Other than a general performance advantage of the metric encoding over the topological one, these plots reveal no clear dependence structure of the performance on these parameters.

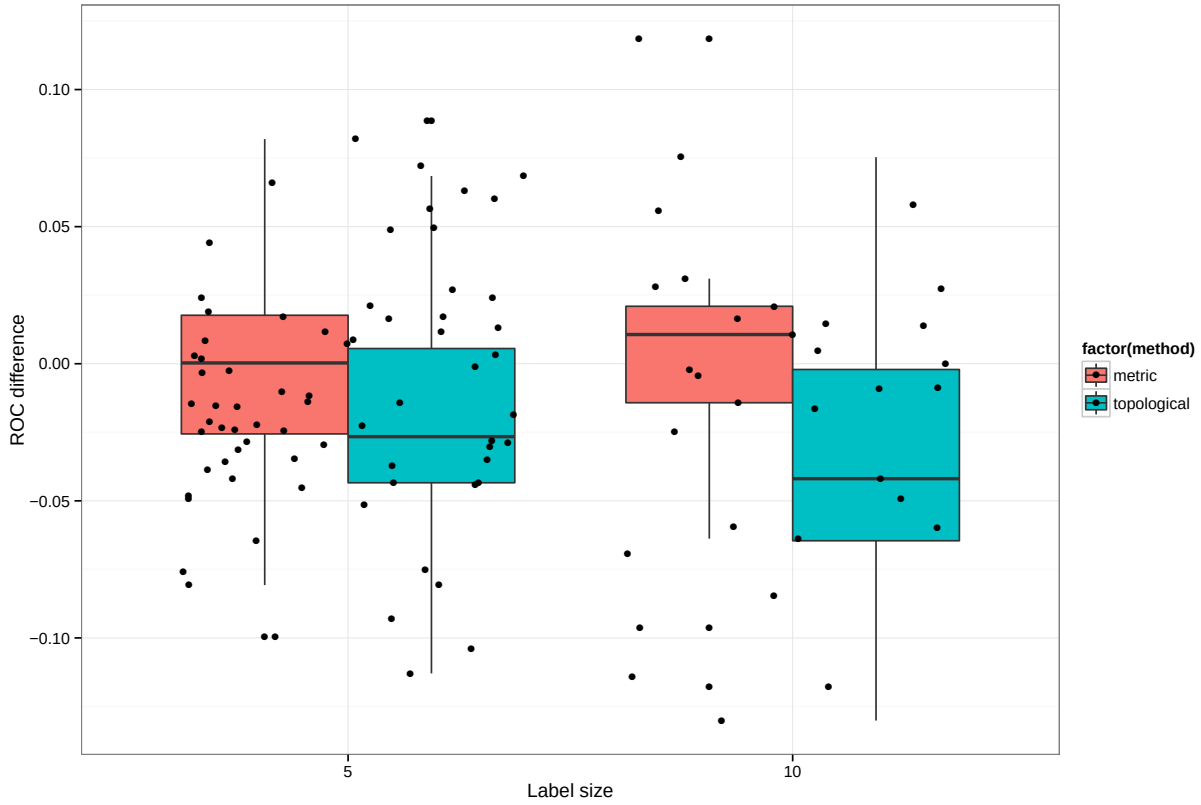

Figure 3.8.: The difference in ROC scores, $ROC_{discrete} - ROC_{vector}$, is displayed for each of the two vector encodings, and split according to the two possible values of the label size, a parameter of the label discretization procedure.

For the parameters specific to each encoding a more detailed view is necessary. Figure Figure 3.10 shows the results for the number of nearest neighbors used in in the

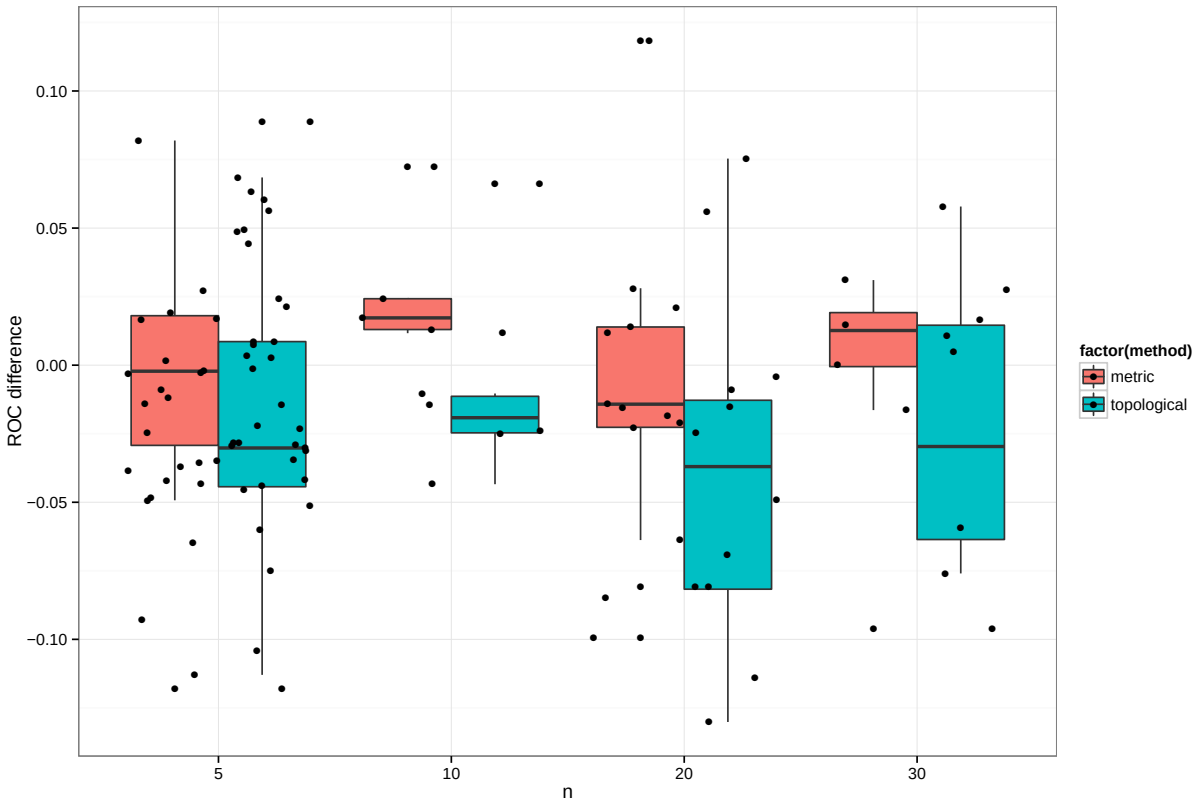

Figure 3.9.: The difference in ROC scores, $ROC_{discrete} - ROC_{vector}$, is displayed for each of the two vector encodings, and split according to the two possible values of the maximum number of clusters, a parameter of the label discretization procedure.

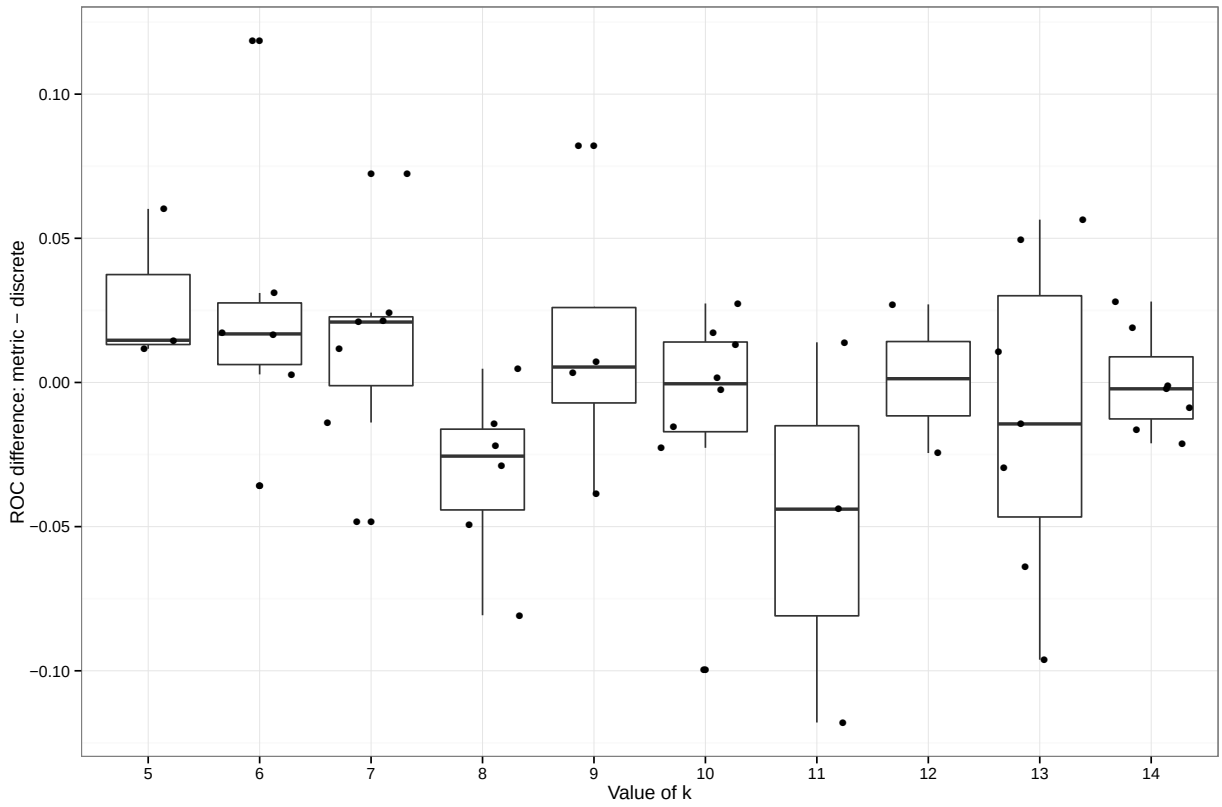

Figure 3.10.: Performance improvement of the metric encoding as a function of the number of nearest neighbors, k.

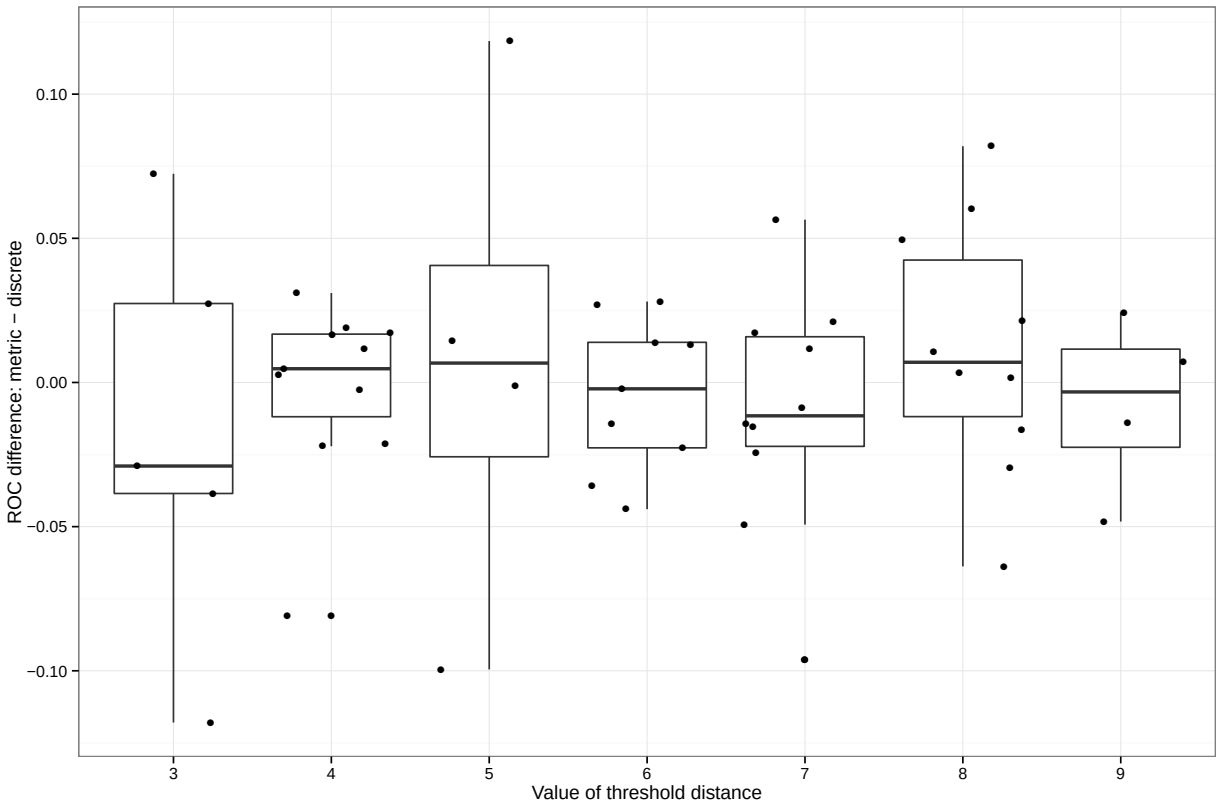

Figure 3.11.: Performance improvement of the metric encoding as function of the threshold distance (θ).

metric encoding. We can here point out the fact that a considerable cases where this encoding showed better performance than the discrete encoding are concentrated in the area with $k \leq 9$.

Figure Figure 3.11 shows the difference in ROC values for each of the values assumed by the parameter θ, that is, the threshold distance. Unlike the previous case this parameter appears to have no clear influence on the algorithm's performance.

Moving on to the parameters for the topological encoding, figure Figure 3.12 shows no clear relation between the values of the number of intervals used, n, except perhaps that most of the good results appear either in the region with $n \leq 9$ or $n = 19$.

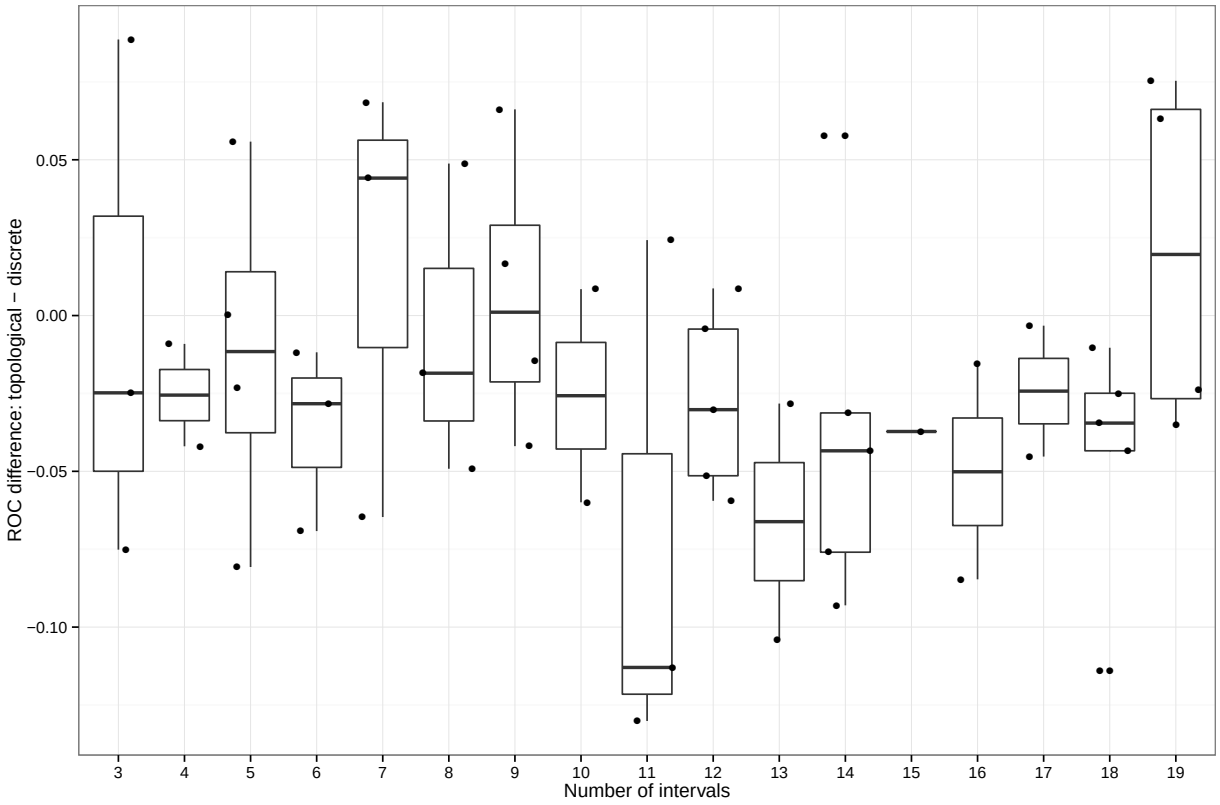

Figure 3.12.: Performance improvement of the topological encoding as a function of the number of intervals n.

Finally, figure Figure 3.13 plots the relation for the maximum distance explored by the algorithm, D_{max}, and performance. There is again no clear functional relation present.

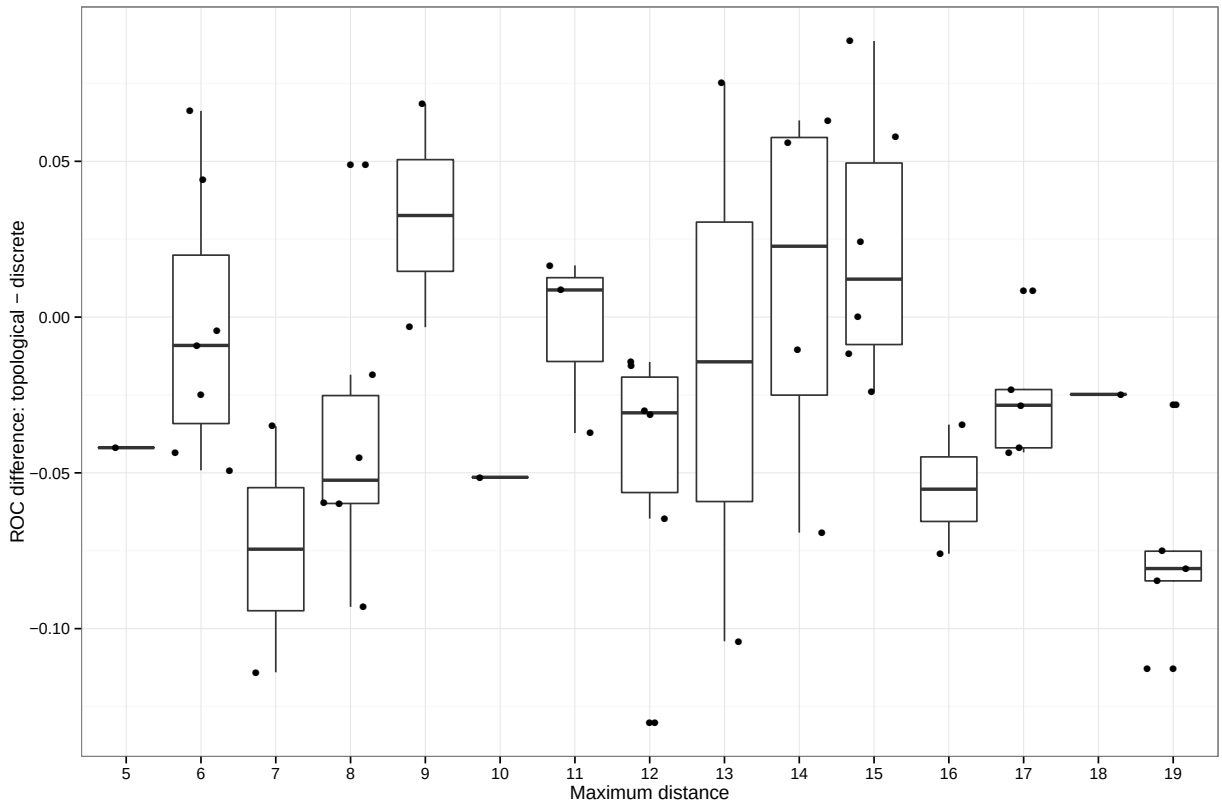

Figure 3.13.: Performance improvement of the topological encoding as a function of the maximum distance explored D_{max}

A different hypothesis that was considered during the evaluation of the results was the dependence of the vector encoding's performance dependence on the average size of the molecules contained on the data sets. A larger molecule, that is, one with more atoms, would translate to a graph with more nodes. The reasoning behind this reflects the idea that, since both vector encodings try to capture information about the local structural features present in the graphs, larger graphs should increase the information provided by the encoding.

Figure 3.14 shows scatterplots of molecular size and ROC score difference between the discrete encoding and both vector encodings (metric and topological, respectively). It is interesting to note that, despite the apparent plausibility behind this hypothesis, these graphs show that it does not hold water. More dramatically, figure Figure 3.14 would even suggest that performance actually *decreases* with increasing molecule size for the metric encoding.

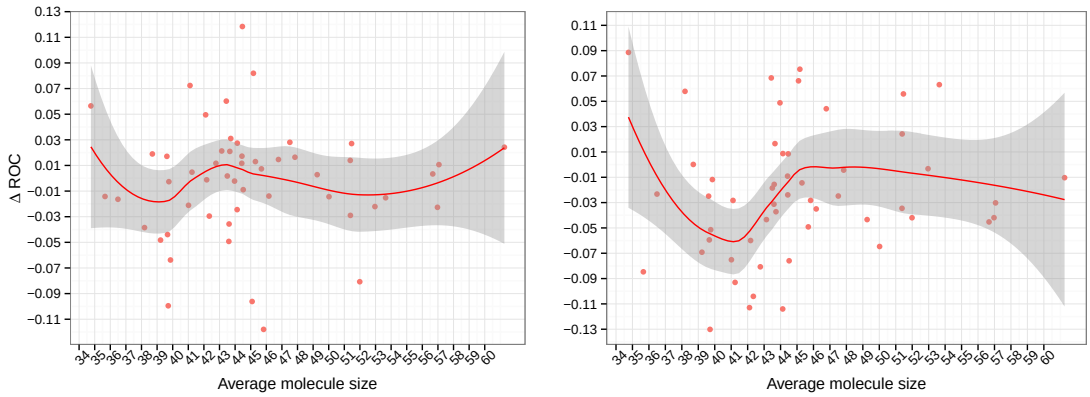

Figure 3.14.: Performance improvement as a function of molecular size. The left panel shows the metric encoding, the right panel shows the topological one.

3.3. Graph Sampling

The final performance test was an evaluation of the predictive performance of the sampled structures. The classification models were again produced by EDeN, while all the graph grammar induction and graph sampling was done with the Python package GraphLearn (cf. [CSs15]). Out of the 54 available bioassay datasets, a subset was chosen for which the performance of either of the vector encodings was "significantly" better than the that of the discrete encoding. Here we consider cases with "significant" improvement to be those where the mean value of the ROC score for the vector encoding lies at least one standard deviation above the mean of the ROC score for the discrete encoding. This leaves a total of 7 datasets. For the metric encoding they correspond to the bioassays with AIDs 2331, 493250, 602166, and 651970 (cf. Figure 3.15). For the topological encoding they are the AIDs 2235, 602167, and 651563 (cf. Figure 3.16).

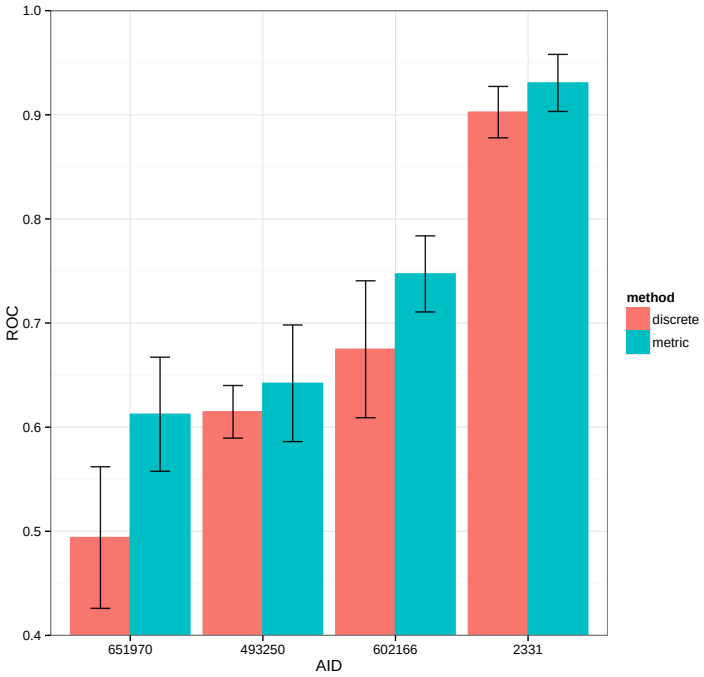

Figure 3.15.: Datasets for which the mean performance obtained by the metric encoding was at least one standard deviation above the mean performance obtained by the discrete encoding.

Regarding the nature of the datasets, we observe that AIDs 602166, 602167, 651970, and 651563 come from luminescence-based bioassays. In our original dataset there are 8 datasets which come from such bioassays, and figure Figure 3.17 shows the results for these. Interestingly the vector encodings outperform the discrete encoding in almost every case.

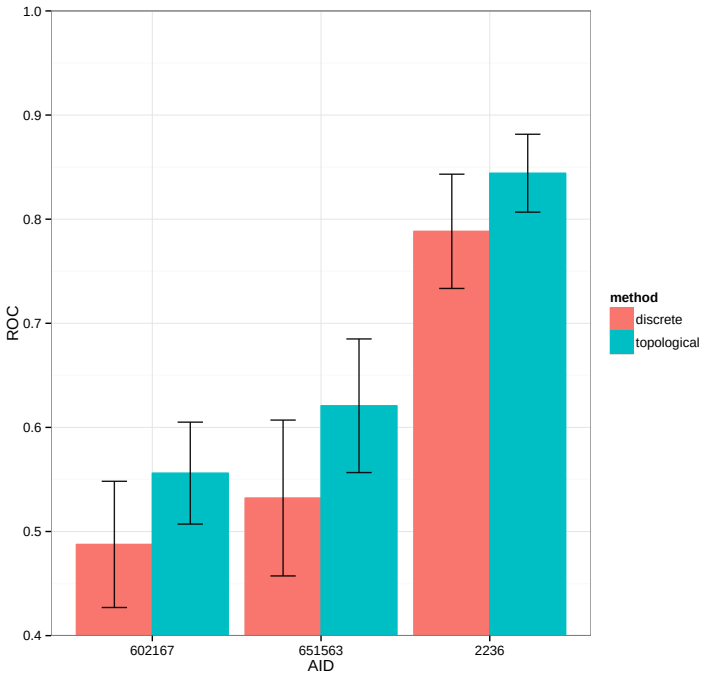

Figure 3.16.: Datasets for which the mean performance obtained by the topological encoding was at least one standard deviation above the mean performance obtained by the discrete encoding.

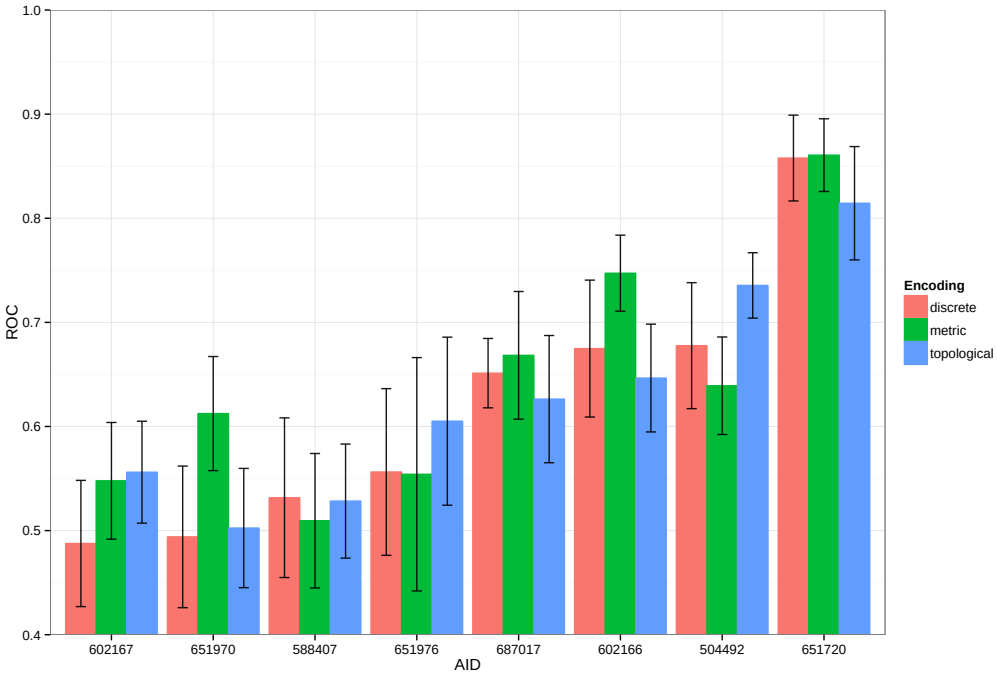

Figure 3.17.: Three-way comparison for datasets from luminescence based bioassays.

We now go back to the 7 datasets selected due to the vector encodings presenting a significant improvement. For each of these, the data are randomly split in 70% training, 30% test data. Then, for increasing values of p, a random subset of $p\%$ of the original training set is used as seed set to perform the graph sampling. Three different classification models are then trained on the original data, on the generated samples, and on the original data plus the samples. This step is repeated 5 times, and the results for each are recorded. These are presented as *learning curves*, which show the change in performance as a function of the proportion p of the original training set used to train the models and generate the samples. The best two results are shown here in Figure 3.18 (metric encoding) and Figure 3.19 (topological encoding). The results for the seven datasets are presented in Table 3.3 and Table 3.4. The important result here is an improvement of the model's performance when adding the sampled structures to the original training set, which is used a proxy measure of their "quality".

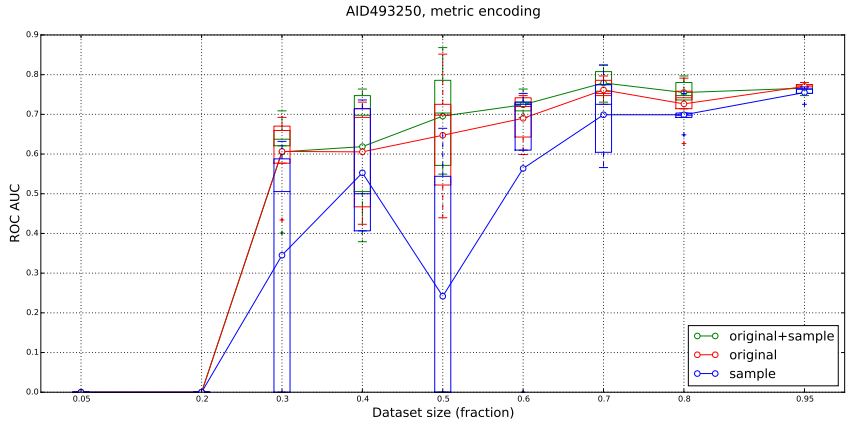

Figure 3.18.: AID 493250, metric encoding

It is also possible to compare the number of components of the graph grammar for each of the available encodings. As has been mentioned before, it is to be expected that the vector-labelled encodings will produce richer grammars than the discrete encoding, owing to the larger number of structures which are geometrically distinct even if they are identical with respect to their atomic makeup. Figure 3.20 shows an example where this hypothesis holds for the metric encoding; the same was found to be true in general. However, grammars produced using the topological encoding tend to exhibit only a slight increase in the number of structures. Figure 3.21 shows an example of this.

Finally, Figure 3.22 presents an average over the 7 datasets of the total number of samples produced, for increasing sizes of the seed set. The metric encoding again produces a clearly larger number of samples than the discrete encoding; the topological encoding on the other hand produces *less* samples. As an explanation for this we advance the hypothesis that, since this encoding does not keep track of the

Table 3.3.: Discrete vs. metric encoding

%	AID							
	2331		493250		602166		651970	
	Discrete	Metric	Discrete	Metric	Discrete	Metric	Discrete	Metric
0.05	NA	NA	NA	NA	NA	NA	NA	NA
0.2	NA	NA	NA	NA	NA	NA	NA	NA
0.3	0.012	-0.018	NA	-0.001	0.073	0.010	0.002	-0.030
0.4	-0.008	-0.002	-0.029	0.013	0.043	0.005	0.010	0.001
0.5	0.006	0.034	0.007	0.048	0.030	-0.012	-0.013	0.030
0.6	-0.013	0.023	0.020	0.034	-0.068	0.002	-0.047	0.013
0.7	-0.003	-0.002	-0.029	0.176	-0.044	0.028	0.089	-0.037
0.8	0	0	0.020	0.028	-0.022	-0.014	0	-0.008
0.95	0.025	-0.018	-0.027	-0.004	-0.025	0.006	-0.065	0.018

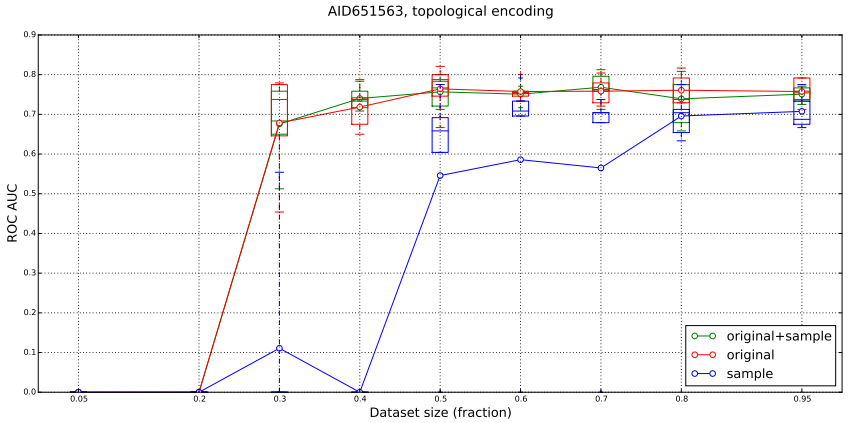

Figure 3.19.: AID 651563, topological encoding

Table 3.4.: Discrete vs. topological encoding

%	AID					
	2236		602167		651563	
	Discrete	Topological	Discrete	Topological	Discrete	Topological
0.05	NA	NA	NA	NA	NA	NA
0.2	NA	NA	NA	NA	NA	NA
0.3	0.002	NA	-0.039	-0.024	-0.014	-0.002
0.4	0.001	-0.012	0.025	0.013	-0.035	0.021
0.5	-0.014	-0.003	-0.017	0.004	0.001	-0.007
0.6	-0.048	0.019	0	0.026	-0.064	-0.006
0.7	0.088	-0.005	0	0.044	0.014	0.010
0.8	0.003	0.015	0.066	-0.010	-0.101	-0.021
0.95	-0.068	0.021	-0.039	0.002	0.032	-0.006

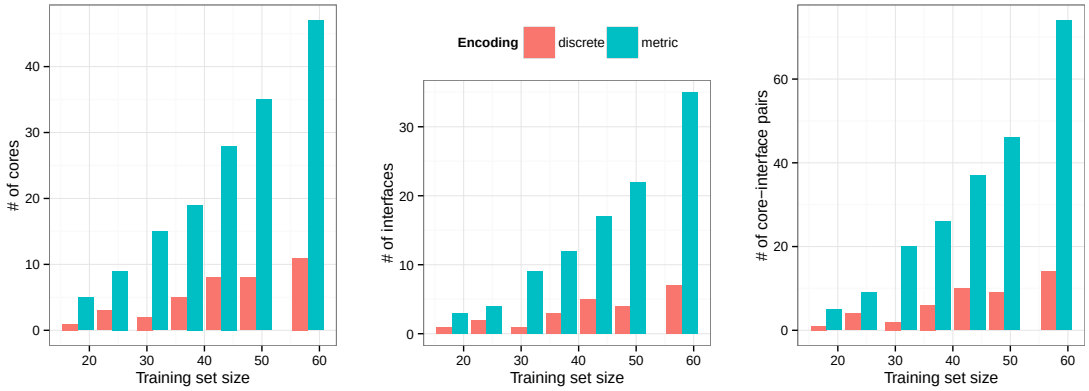

Figure 3.20.: Number of grammar components for several subsets of AID 493250 for the discrete and metric encodings.

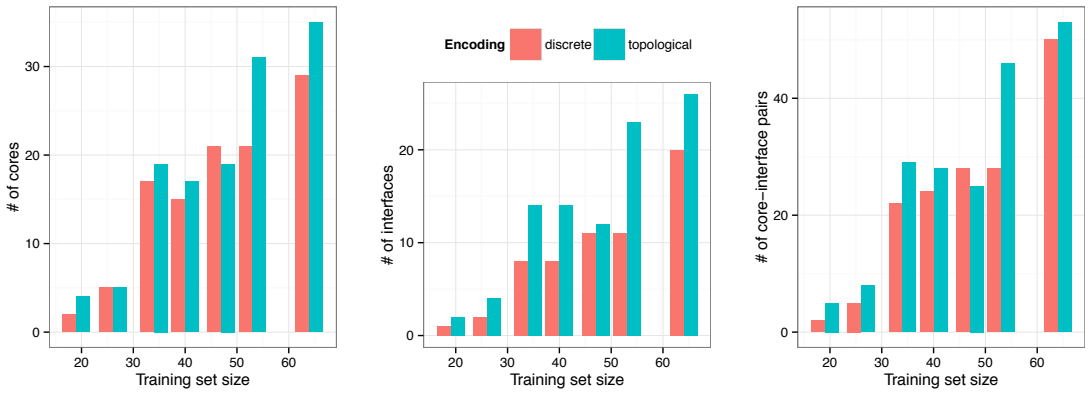

Figure 3.21.: Number of grammar components for several subsets of AID 651563 for the discrete and topological encodings.

type of the atoms which make up the individual structures as the metric encoding does, it is more difficult for the sampling algorithm to consistently produce feasible structures. As can be seen in Figure 3.23, this distinction between the two vector encodings is maintained when we consider runtimes: the entire process takes consistently longer for the metric encoding than for the topological one.

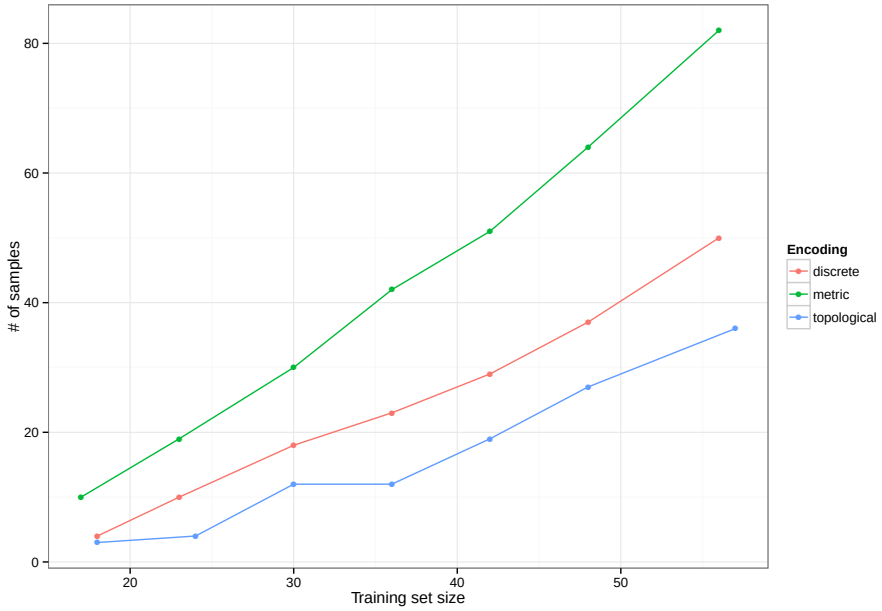

Figure 3.22.: Evolution of the average number of samples generated for increasing seed set sizes.

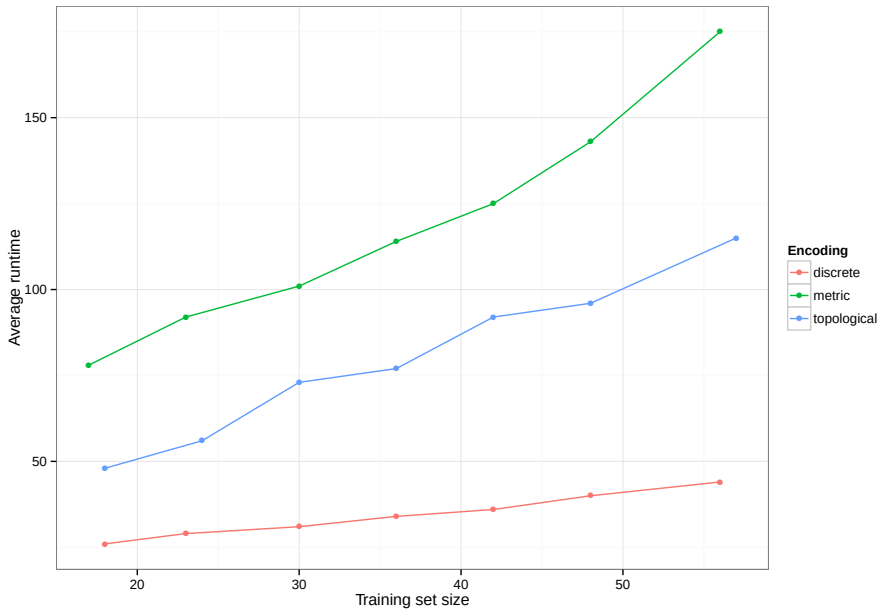

Figure 3.23.: Evolution of average runtime for sample generation and model training for increasing training/seed set sizes.

4. Discussion and Outlook

Perhaps the most important observation to be made about the results presented in the last chapter is that, even if the three different methods presented here have a similar performance on average, there are still interesting cases in which one clearly outperforms the others. See, for example, the results for the bioassays 651970, 1279, or 624120 in Figure 3.5; 602166, 743150, or 2236 in Figure 3.6. It seems reasonable to attribute this to a specific structural properties of the information which are effectively detected by the algorithms. A more careful study of this remained beyond the scope of the present work.

Also of note is the low influence that the encoding and discretization parameters appear to have on the model's performance. Any future exploration of the methods here described could therefore take as starting point either a proper subset of the parameters used here, or obtain an entirely new parametrization. Thanks to the fact that the methods described here can deal with arbitrary vector-valued labels in graphs, the graph-invariant encodings presented here are in no way the only ones that can be used. In particular, for the metric encoding, we would consider using diverse similarity functions to transform the distances calculated by the algorithm. For this work only the function $f(d) = \frac{1}{1+d}$ was used.

Regarding the topological encoding, it is doubtlessly possible to represent the information provided by a sequence of Vietoris-Rips complexes in a more concise and elegant way than what was attempted here. It would also be possible to use a different topological representation, e.g. a Čech complex, which is more computationally expensive to obtain but in theory offers a more faithful topological representation of the underlying space. Finally, it would also be worthwhile to include atom types in the labels, which would likely increase the local resolution of the models.

As for the grammar induction and sampling, the current implementation does not rely on parallelization. This no doubt limited the possibilities for exploring the different possible parameter configurations for both grammar and sampling. It would also be desirable to have precise results regarding the sampling algorithm's speed of convergence, as well as the length of the necessary burn-in period as a function of the initial conditions.

A. BioAssay metadata

All the following information was obtained from the National Center for Biotechnology Information (http://www.ncbi.nlm.nih.gov), the PubChem BioAssay Database (http://pubchem.ncbi.nlm.nih.gov).

Name Counterscreen for inhibitors of LEDGF/p75-dependent integration: TR-FRET-based biochemical high throughput dose response counterscreen assay to identify activators of HIV-1 Integrase multimerization

AID 1053171

Protein target integrase, partial [Human immunodeficiency virus 1]

 # active subst. (conf.) 101 (700)

 # inactive subst. (conf.) 128 (1141)

Name Cell-based high throughput confirmation assay for antagonists of neuropeptide Y receptor Y2 (NPY-Y2)

AID 1257

Protein target neuropeptide Y receptor Y2 [Homo sapiens]

 # active subst. (conf.) 228 (2021)

 # inactive subst. (conf.) 479 (4217)

Name Confirmation cell-based high throughput screening assay to measure STAT1 inhibition

AID 1263

Protein target signal transducer and activator of transcription 1 isoform alpha [Homo sapiens]

 # active subst. (conf.) 109 (988)

 # inactive subst. (conf.) 90 (769)

Name Dose response cell-based screening assay for antagonists of neuropeptide Y receptor Y2 (NPY-Y2)

AID 1272

Protein target neuropeptide Y receptor Y2 [Homo sapiens]

 # active subst. (conf.) 72 (633)

 # inactive subst. (conf.) 47 (633)

Name Dose response counterscreen for neuropeptide Y receptor Y2 (NPY-Y2): Cell-based high throughput assay to measure NPY-Y1 antagonism

AID 1279

Protein target neuropeptide Y receptor Y1 [Homo sapiens]

 # active subst. (conf.) 74 (653)

 # inactive subst. (conf.) 45 (371)

Name TR-FRET counterscreen for FAK inhibitors: dose-response biochemical high throughput screening assay to identify inhibitors of Proline-rich tyrosine kinase 2 (Pyk2)

AID 1641

Protein target PTK2B protein tyrosine kinase 2 beta [Homo sapiens]

 # active subst. (conf.) 93 (732)

 # inactive subst. (conf.) 118 (916)

Name Fluorescence counterscreen assay for potentiators or agonists of NPY-Y2: Cell-based high-throughput screening assay to identify potentiators or agonists of NPY-Y1.

AID 1651

Protein target neuropeptide Y receptor Y1 [Homo sapiens]

 # active subst. (conf.) 84 (527)

 # inactive subst. (conf.) 369 (3186)

Name Dose response, multiplexed high-throughput screen for small molecule regulators of RGS family protein interactions, specifically RGS19-Galphao.

AID 1884

Protein target G protein signalling regulator 19 [Homo sapiens]

 # active subst. (conf.) 54 (436)

 # inactive subst. (conf.) 131 (1157)

Name Dose Response Confirmation Via Multiplex HTS Assay For Inhibitors Of MEK Kinase PB1 Domains, Specifically MEK5 Binding To MEK Kinase 2 Wildtype

AID 1897

Protein target mitogen-activated protein kinase kinase kinase kinase 2 [Homo sapiens]

 # active subst. (conf.) 93 (443)

 # inactive subst. (conf.) 93 (572)

Name Confirmation Dose Response Screen For Compounds That Protect HERG From Block By Proarrhythmic Agents

AID 2121

Protein target putative potassium channel subunit

 # active subst. (conf.) 115 (1049)

 # inactive subst. (conf.) 123 (1051)

Name Late Stage Results From The Probe Development Effort To Identify Inhibitors Of The Janus Kinase 2 Mutant JAK2V617F.

AID 2165

Protein target Janus kinase 2 (a protein tyrosine kinase) [Homo sapiens]

 # active subst. (conf.) 158 (1411)

 # inactive subst. (conf.) 69 (634)

Name hERG counter screen for compounds that inhibit/block inward-rectifying potassium ion channel Kir2.1

AID 2236

Protein target putative potassium channel subunit [Homo sapiens]

 # active subst. (conf.) 74 (352)

 # inactive subst. (conf.) 246 (1861)

Name Fluorescence-based biochemical high throughput confirmation assay for inhibitors of Protein Phosphatase 5 (PP5).

AID 2331

Protein target PPP5C protein [Homo sapiens]

 # active subst. (conf.) 102 (849)

 # inactive subst. (conf.) 360 (3152)

Name Confirmation Concentration-Response Assay for Activators of Human Muscle isoform 2 Pyruvate Kinase: for Probe SAR

AID 2533

Protein target pyruvate kinase isozymes M1/M2 isoform M2 [Homo sapiens]

 # active subst. (conf.) 86 (820)

 # inactive subst. (conf.) 116 (858)

Name Mode of action - Automated patch clamp assay for KCNQ2 potentiators on Retigabine insensitive KCNQ2 Mutant W236L cell line

AID 2558

Protein target potassium voltage-gated channel, KQT-like subfamily, member 2 [Rattus norvegicus]

 # active subst. (conf.) 91 (810)

 # inactive subst. (conf.) 845 (7493)

Name QHTS Confirmation Assay For Inhibitors Of Bloom's Syndrome Helicase (BLM)

AID 2585

Protein target bloom syndrome protein [Homo sapiens]

 # active subst. (conf.) 85 (669)

 # inactive subst. (conf.) 55 (491)

Name QHTS Assay Multiplex Screening To Identify Dual Action Probes In A Cell Model Of Huntington: Cytoprotection (Protease Relase)

AID 2669

Protein target huntingtin [Homo sapiens]

 # active subst. (conf.) 81 (669)

 # inactive subst. (conf.) 58 (489)

Name TR-FRET-based biochemical high throughput dose response assay to identify NR2E3 inverse agonists

AID 463256

Protein target photoreceptor-specific nuclear receptor [Homo sapiens]

 # active subst. (conf.) 83 (778)

 # inactive subst. (conf.) 31 (283)

Name Confirmation Concentration-Response Assay for Enhancers of SMN2 Splice Variant Expression for Further Probe SAR

AID 488832

Protein target survival motor neuron protein isoform d [Homo sapiens]

 # active subst. (conf.) 78 (779)

 # inactive subst. (conf.) 31 (306)

Name Dose Response Confirmation Of UHTS Inhibitors Of Mouse Intestinal Alkaline Phosphatase Using Human Intestinal Alkaline Phosphatase

AID 488876

Protein target Alkaline phosphatase, intestinal [Homo sapiens]

 # active subst. (conf.) 55 (465)

 # inactive subst. (conf.) 95 (788)

Name Dose Response Confirmation Of UHTS Inhibitors Of Mouse Intestinal Alkaline Phosphatase Using Tissue Nonspecific Alkaline Phosphatase.

AID 488906

Protein target alkaline phosphatase, tissue-nonspecific isozyme isoform 1 precursor [Homo sapiens]

 # active subst. (conf.) 103 (870)

 # inactive subst. (conf.) 78 (653)

Name Intein Inhibitors As Potential Tuberculosis Drugs

AID 489010

Protein target DNA recombination protein RecA [Mycobacterium tuberculosis H37Rv]

 # active subst. (conf.) 76 (659)

 # inactive subst. (conf.) 61 (515)

Name Fluorescence polarization-based biochemical high throughput confirmation assay for inhibitors of human platelet-activating factor acetylhydrolase 1B, catalytic subunit 3 (PAFAH1B3)

AID 493032

Protein target platelet-activating factor acetylhydrolase IB subunit gamma [Homo sapiens]

 # active subst. (conf.) 118 (811)

 # inactive subst. (conf.) 274 (2099)

Name MEX-5 Measured in Biochemical System Using Plate Reader - 2024-01_Inhibitor_Dose_CherryPick_Activity_3

AID 493250

Protein target Zinc finger protein MEX-5, Muscle excess 5- C. elegans

 # active subst. (conf.) 121 (967)

 # inactive subst. (conf.) 368 (2858)

Name SAR Analysis of small molecule inhibitors of Sentrin-specific proteases (SENPs) using a Caspase-3 Selectivity assay

AID 504488

Protein target caspase-3 preproprotein [Homo sapiens]

 # active subst. (conf.) 117 (946)

 # inactive subst. (conf.) 150 (1366)

Name SAR Analysis of small molecule inhibitors of Sentrin-specific protease 6 (SENP6) using a Luminescent assay

AID 504492

Protein target SUMO-1-specific protease [Homo sapiens]

 # active subst. (conf.) 111 (952)

 # inactive subst. (conf.) 156 (1360)

Name Development of CDK5 inhibitors Measured in Biochemical System Using Plate Reader - 2083-01_Inhibitor_Dose_CherryPick_Activity_Set2

AID 504545

Protein target CDK5 [Homo sapiens]

 # active subst. (conf.) 88 (665)

 # inactive subst. (conf.) 63 (508)

Name Fluorescence-based biochemical high throughput dose response assay for activators of the calcium sensitivity of cardiac Regulated Thin Filaments (RTF)

AID 504698

Protein target troponin C, slow skeletal and cardiac muscles [Homo sapiens]

 # active subst. (conf.) 76 (492)

 # inactive subst. (conf.) 168 (1477)

Name Fluorescence-based cell-based high throughput confirmation assay for inhibitors of TLR9-MyD88 binding

AID 540250

Protein target toll-like receptor 9 [Homo sapiens]

 # active subst. (conf.) 345 (2573)

 # inactive subst. (conf.) 330 (2685)

Name Luminescence-based cell-based high throughput dose response assay for agonists of heterodimerization of the mu 1 (OPRM1) and delta 1 (OPRD1) opioid receptors

AID 588407

Protein target mu-type opioid receptor isoform MOR-1 [Homo sapiens]

 # active subst. (conf.) 126 (963)

 # inactive subst. (conf.) 103 (736)

Name Luminescence-based cell-based high throughput dose response assay for inhibitors of the Steroid Receptor Coactivator 3 (SRC3; NCOA3)

AID 602166

Protein target nuclear receptor coactivator 3 isoform a [Homo sapiens]

 # active subst. (conf.) 119 (954)

 # inactive subst. (conf.) 110 (955)

Name Counterscreen for inhibitors of the Steroid Receptor Coactivator 3 (SRC3; NCOA3): Luminescence-based cell-based high throughput dose response assay to identify inhibitors of the Herpes Virus Virion Protein 16 (VP16)

AID 602167

Protein target transactivating tegument protein VP16 [Human herpesvirus 1]

 # active subst. (conf.) 84 (718)

 # inactive subst. (conf.) 145 (1191)

Name TRFRET-based biochemical high throughput dose response assay for inhibitors of the interaction of the Ras and Rab interactor 1 protein (Rin1) and the c-abl oncogene 1, non-receptor tyrosine kinase (Abl)

AID 602181

Protein target ras and Rab interactor 1 [Homo sapiens]

 # active subst. (conf.) 82 (713)

 # inactive subst. (conf.) 148 (1269)

Name Turbidometric Biochemical Primary HTS to identify inhibitors of Protein Disulfide Isomerase Measured in Biochemical System Using Plate Reader - 2137-01_Inhibitor_Dose_CherryPick_Activity

AID 602350

Protein target Prolyl-4 hydroxylase, beta polypeptide

 # active subst. (conf.) 111 (753)

 # inactive subst. (conf.) 322 (2793)

Name SAR Analysis Of Small Molecule Agonists NTR1 In A Image Based Assay Set 3

AID 602356

Protein target neurotensin receptor type 1 [Homo sapiens]

 # active subst. (conf.) 78 (670)

 # inactive subst. (conf.) 74 (682)

Name Dose response confirmation of uHTS small molecule inhibitors of Striatal-Enriched Phosphatase via a fluorescence intensity assay

AID 602372

Protein target tyrosine-protein phosphatase non-receptor type 5 isoform a [Homo sapiens]

 # active subst. (conf.) 80 (746)

 # inactive subst. (conf.) 86 (768)

Name TR-FRET secondary assay for HTS discovery of chemical inhibitors of anti-apoptotic protein Bfl-1

AID 621

Protein target B-cell leukemia/lymphoma 2 related protein A1a [Mus musculus]

 # active subst. (conf.) 89 (697)

 # inactive subst. (conf.) 96 (740)

Name Chemical Optimization Of In Vitro Pharmacology And DMPK Properties Of The Highly Selective MAChR 4 (M4) Positive Allosteric Modulator (PAM) Series With Greatly Improved Human Receptor Activity (HM4 Calcium Potency)

AID 623938

Protein target muscarinic acetylcholine receptor M4 [Homo sapiens]

 # active subst. (conf.) 55 (198)

 # inactive subst. (conf.) 40 (229)

Name Validation For Compounds That Inhibit KCNQ1 Potassium Channels On Automated Electrophysiology Assay

AID 624120

Protein target potassium voltage-gated channel subfamily KQT member 1 isoform 1 [Homo sapiens]

 # active subst. (conf.) 87 (716)

 # inactive subst. (conf.) 405 (3701)

Name Cytotoxicity (24 hours) Measured in Cell-Based System Using Plate Reader - 2137-02_Inhibitor_Dose_CherryPick_Activity

AID 624285

Protein target Prolyl-4 hydroxylase, beta polypeptide

 # active subst. (conf.) 72 (476)

 # inactive subst. (conf.) 369 (3150)

Name Dose ResponseConfirmation of SKN-1 Inhibitor hits in a fluorescence ratio assay - Set 2

AID 651563

Protein target SKiNhead family member (skn-1) [Caenorhabditis elegans]

 # active subst. (conf.) 83 (589)

 # inactive subst. (conf.) 282 (2022)

Name qHTS Assay for Iinhibitors of HIV-1 Budding by Blocking the Interaction of PTAP/TSG101: Hit Validation

AID 651600

Protein target tumor susceptibility gene 101 protein [Homo sapiens]

 # active subst. (conf.) 163 (979)

 # inactive subst. (conf.) 221 (1772)

Name Counterscreen for inhibitors of the interaction of the lipase co-activator protein, abhydrolase domain containing 5 (ABHD5) with perilipin-5 (MLDP; PLIN5): Luminescence-based biochemical high throughput dose response assay to identify inhibitors of Hepatocyte nuclear factor 4 (HNF4) dimerization

AID 651720

Protein target hepatocyte nuclear factor 4-alpha isoform b [Homo sapiens]

 # active subst. (conf.) 133 (1216)

 # inactive subst. (conf.) 101 (617)

Name Luminescence-based cell-based high throughput dose response assay for inverse agonists of the liver receptor homolog-1 (LRH-1; NR5A

AID 651970

Protein target nuclear receptor subfamily 5 group A member 2 isoform 2 [Homo sapiens]

 # active subst. (conf.) 86 (819)

 # inactive subst. (conf.) 153 (1487)

Name Late stage luminescence-based cell-based high throughput dose response assay for inverse agonists of the liver receptor homolog-1 (LRH-1; NR5A2)

AID 651976

Protein target nuclear receptor subfamily 5 group A member 2 isoform 2 [Homo sapiens]

 # active subst. (conf.) 54 (525)

 # inactive subst. (conf.) 46 (460)

Name Dose response confirmation of small molecule inhibitors of Low Molecular Weight Protein Tyrosine Phosphatase, LMPTP, in an orthogonal absorbance-based assay

AID 652005

Protein target low molecular weight phosphotyrosine protein phosphatase isoform c [Homo sapiens]

 # active subst. (conf.) 73 (622)

 # inactive subst. (conf.) 265 (2293)

Name Luminescence-based cell-based high throughput dose response assay for inhibitors of the orphan nuclear receptor subfamily 0, group B, member 1 (DAX1; NR0B1): repression of SF-1 (NR5A1) activated StAR promoter by full-length DAX-1

AID 687017

Protein target nuclear receptor subfamily 0 group B member 1 [Homo sapiens]

 # active subst. (conf.) 55 (427)

 # inactive subst. (conf.) 194 (1708)

Name QHTS For Inhibitors Of WRN Helicase: BLM Helicase Counterscreen For WRN Inhibitors

AID 720503

Protein target Bloom syndrome protein [Homo sapiens]

 # active subst. (conf.) 368 (2974)

 # inactive subst. (conf.) 110 (802)

Name QHTS For Inhibitors Of Bloom's Syndrome Helicase (BLM): Helicase DNA Unwinding Fluorescent Orthogonal Confirmatory Assay For SAR

AID 720555

Protein target Parkin [Homo sapiens]

 # active subst. (conf.) 73 (680)

 # inactive subst. (conf.) 45 (436)

Name SA12 PAX8: cytotoxicity COV362 Measured in Cell-Based System Using Plate Reader - 7054-16_Inhibitor_Dose_DryPowder_Activity

AID 743144

Protein target Human Pax8, human paired-box protein

 # active subst. (conf.) 78 (734)

 # inactive subst. (conf.) 50 (490)

Name SA8-Pax8: cytotoxicity OV-90 Measured in Cell-Based System Using Plate Reader - 7054-10_Inhibitor_Dose_DryPowder_Activity

AID 743149

Protein target Human Pax8, human paired-box protein

 # active subst. (conf.) 89 (850)

 # inactive subst. (conf.) 51 (494)

Name SA11 PAX8: cytotoxicity OVCAR4 Measured in Cell-Based System Using Plate Reader -

7054-13_Inhibitor_Dose_DryPowder_Activity

AID 743150

Protein target Human Pax8, human paired-box protein

 # active subst. (conf.) 75 (704)

 # inactive subst. (conf.) 65 (640)

Name Dose-response biochemical assay for inhibitors of Focal Adhesion Kinase (FAK)

AID 810

Protein target Focal adhesion kinase 1 (FADK 1) (pp125FAK) (Protein-tyrosine kinase 2)

 # active subst. (conf.) 110 (863)

 # inactive subst. (conf.) 100 (785)

Name Dose Response Confirmation For Small Molecule Inhibitors Of Eukaryotic Translation Initiation

AID 855

Protein target eukaryotic translation initiation factor 4E [Mus musculus]

 # active subst. (conf.) 78 (541)

 # inactive subst. (conf.) 486 (4083)

Name Discovery of novel allosteric modulators of the M1 muscarinic receptor: Antagonist Dose-Response Counterscreen

AID 860

Protein target cholinergic receptor, muscarinic 4 [Rattus norvegicus]

 # active subst. (conf.) 272 (1741)

 # inactive subst. (conf.) 447 (3694)

Bibliography

[Aro50] ARONSZAJN, Nachman: Theory of Reproducing Kernels. In: *Transactions of the American Mathematical Society* 68 (1950), Nr. 3, S. 337–404

[Car09] CARLSSON, Gunnar: Topology and Data. In: *Bulletin of the American Mathematical Society* 46 (2009), Nr. 2, S. 255–308

[CG95] CHIB, Siddharta ; GREENBERG, Edward: Understanding the Metropolis-Hastings Algoriothm. In: *The American Statistician* 49 (1995), Nr. 4, S. 327–335

[CG10] COSTA, Fabrizio ; GRAVE, Kurt D.: Fast Neighborhood Subgraph Pairwise Distance Kernel. In: *Proceedings of the 27th International Conference on Machine Learning*. Haifa, Israel, 2010

[CGS+15] COSTA, Fabrizio ; GRÜNING, Björn ; SALAIZ, Jose Luis L. ; SMAUTNER ; MATICZKA, Daniel ; FRANKBOOTH ; CERDÁN, Diego: *EDeN: EDeN 0.2*. 2015. – Available online at http://dx.doi.org/10.5281/zenodo.27945

[Cos14] COSTA, Fabrizio: The Constructive Learning Problem: learning an efficient sampler for graphs. In: *Artificial Intelligence Journal SI: Constraints, Mining and Learning* (2014)

[CSs15] COSTA, Fabrizio ; SALAIZ, Jose Luis L. ; SMAUTNER: *GraphLearn*. 2015. – Available online at https://github.com/antworteffekt/GraphLearn

[Dam90] DAMGÅRD, Ivan B.: A Design Principle for Hash Functions. In: *Advances in Cryptology - CRYPTO' 89 Proceedings* Bd. 435. Springer New York, 1990, S. 416–427

[Ghr07] GHRIST, Robert: Barcodes: The Persistent Topology of Data. In: *Bulletin of the American Mathematical Society* 45 (2007), Nr. 1, S. 61–75

[GY03] GROSS, J.L. ; YELLEN, J.: *Handbook of Graph Theory*. 1st. CRC, 2003 (Discrete Mathematics and Its Applications)

[Hau99] HAUSSLER, David: Convolution Kernels on Discrete Structures / University of California at Santa Cruz. 1999 (UCSC-CRL-99-10). – Forschungsbericht

[HSS08] HOFMANN, Thomas ; SCHÖLKOPF, Bernhard ; SMOLA, Alexander J.: Kernel Methods in Machine Learning. In: *The Annals of Statistics* 36 (2008), Nr. 3, S. 1171–1220

[IUP14] IUPAC: *Compendium of Chemical Terminology*. 2014. – Available online at http://goldbook.iupac.org/C01258.html

[RC77] READ, Ronald C. ; CORNEIL, Derek G.: The Graph Isomorphism Disease. In: *Journal of Graph Theory* 1 (1977), S. 339–363

[SPST+01] SCHÖLKOPF, Bernhard ; PLATT, John C. ; SHAWE-TAYLOR, John ; SMOLA, Alex J. ; WILLIAMSON, Robert C.: Estimating the Support of a High-Dimensional Distribution. In: *Neural Computation* 13 (2001), S. 1443–1471

[Vie27] VIETORIS, Leopold: Über den höheren Zusammenhang kompakter Räume und eine Klasse von zusammenhangstreuen Abbildungen. In: *Mathematische Annalen* 97 (1927), Nr. 1, S. 454–472