# APPLICATIONS OF CENTERED KERNEL TARGET ALIGNMENT IN INDUCTIVE LOGIC PROGRAMMING

BY

BENJAMIN H OTT

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE AND STATISTICS

UNIVERSITY OF RHODE ISLAND

2019

DOCTOR OF PHILOSOPHY DISSERTATION

OF

BENJAMIN H OTT

APPROVED:

Dissertation Committee:

Major Professor Lutz Hamel

Natallia Katenka

Marco Alvarez

Nancy Eaton

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2019

## ABSTRACT

This study aimed both to apply centered kernel target alignment (CKTA) to inductive logic programming (ILP) in several different ways and to apply a complete refinement operator in a practical setting. A new genetic algorithm (GA) results from the research, utilizing a complete, locally finite refinement operator and also incorporating CKTA both as a fitness score and as a means for the promotion of diversity. As a fitness score, CKTA can either be used standalone or as a contributor to a hybrid score which utilizes the accuracy (weighted or normal) of the learned logic hypothesis as well. In terms of diversity promotion, CKTA is used for incest avoidance and as a means for creating diverse ensembles. This is the first study to employ CKTA for diversity promotion of any kind. It is also the first to apply CKTA to ILP. The kernels in this study are created via dynamic propositionalization, where the features are learned jointly with the kernel to be used for classification via a genetic algorithm. In this sense, genetic kernels for ILP are created. The results show that the methods proposed herein are promising, encouraging future work. It is worth noting that the applications of CKTA in this study are not specific to ILP. They can also be used more generally in any other domain using kernels.

## ACKNOWLEDGMENTS

As it turns out, the doctoral journey requires a bit of support, and different kinds of support at that. There were some individuals whose support and encouragement made the work within the pages that follow possible. To these people, I am eternally grateful and I consider myself fortunate to know them and to have earned their support.

First and foremost, I would like to thank my loving wife, Rebecca, for her support of my research and for the saint like patience she exhibited those days and nights that I spent researching and writing the contents of this thesis. I would also like to thank my children for understanding that their father was unable to play with them some nights. Secondly, I would like to thank my advisor, Dr. Hamel, for challenging me, and for offering sage advice and insightful guidance to me throughout the course of my research. Third, I would like to thank Igor Maznista for making his prolog parser available. It was written and tested well and served as a great starting point in these studies. I would also like to thank the members of my research committee, Dr. Katenka, Dr. Eaton, and Dr. Alvarez, as well as my defense chair, Dr. Dash, for reading and commenting on my thesis and for being a part of my educational journey. Lastly, I would like to thank my family and friends for their encouragement and for providing such a stalwart support network. I would like to distinctly thank my parents for their words of encouragement and for the wonderful example of perseverance that they set for me. I would also like to thank my father-in-law and my mother-in-law for setting great examples and encouraging me throughout my studies.

## DEDICATION

I dedicate this thesis to my parents, Rose and Terry, my wife Rebecca, her parents, Bruce and Jean, and my children. They have all made tremendous sacrifices to allow me to complete these studies. No words can adequately express my gratitude.

## Contents

**Page**

Page

## List of Figures

## List of Tables

**Table** **Page**

# CHAPTER 1

## Introduction

Inductive logic programming (ILP) is a subfield of machine learning which utilizes logic programming in order to describe background knowledge, facts, hypotheses, etc. ILP utilizes background knowledge (theory), positive examples, and negative examples in order to learn new hypotheses. The goal is then to generate a hypothesis, which, when combined with the background knowledge, implies all of the positive examples and none of the negative ones. This contrasts with deductive logic where given background theory, we find out what can be inferred from it. In this sense, inductive logic can be used to create new hypotheses whereas deductive logic is used to determine what is true given your background theory. The ability to generate new hypotheses in this fashion is what makes inductive logic programming appealing. As we collect more data points in the form of observed examples, we would like to know how this information can be used to generate new knowledge.

Using logic programming allows theories to be human readable as logic programming employs a high-level symbolic representation. For instance, with logic programming, a clause may be as follows:

Promotion(person) $\leftarrow$ WorksHard(person), ValuedContributor(person), MeetsRequirementsOfAdvancedPosition(person)

This clause indicates that if a person works hard, is a valued contributor, and meets the requirements of an advanced position, they will be promoted. This is clear based on the knowledge representation used. This clause is for example only, clearly in the real world this can be more complicated. The clarity of this clause contrasts with other subfields of AI in which data is encoded numerically. For

instance, the above clause may look as follows when described via weights in a neural network setting:

wWorkHard = 0.2

wValuedContributor = 0.3

wMeetsRequirementsOfAdvancedPosition = 0.4

With the neural network, weights for each input are assigned to neurons. Activation functions then determine the output of a neuron. The weights are harder to interpret with the neural network, especially when there are multiple layers to the network. However, "numeric neural networks perform inductive learning in such a way that the statistical characteristics of the data are encoded in their sets of weights" [1] which can be appealing in a variety of application spaces. While each approach has its place, logic programming is particularly useful when a human readable description of the data is desired or where a symbolic representation of the data is a natural choice. For these reasons, inductive logic programming has been and continues to be quite popular in bioinformatics [2, 3]. It is also used in network analysis, web mining, and natural language processing [4].

One focus of this study is to explore the application of centered kernel target alignment to inductive logic programming. Competitive, new approaches to inductive logic programming are created as a result of this exploration, combining several areas of study, including genetic algorithms (GA), inductive logic programming, and statistical learning (kernel methods). The following applications of CKTA are proposed in this study:

1. as a fitness score for genetic algorithms (GA)
2. as a means for promoting diversity
    (a) as a mechanism for incest avoidance in GA

(b) for member selection in ensembles

Note that while this study is focused on the application to ILP, these applications of CKTA are not limited to ILP. They could easily be applied to other problems where kernel learning is utilized. In addition, this study also proposes an ILP learning algorithm which employs a complete, locally finite refinement operator in a practical manner. To the author's knowledge, all ILP algorithms to date use non-complete refinement operators guided by heuristic searches.

This study will improve on the **g**enetic **l**ogic **p**rogramming **s**ystem (GLPS) introduced by Wong and Leung[5, 6]. The GA resulting from this study differentiates itself from other recent kernel based logic programming approaches (and all other ILP approaches known to the author) in that it affords the possibility of searching the complete refinement graph (refinement graphs will be detailed later). However, it should be noted that while this is possible with the given approach, the search space could be infinite and hence limited by computational resources and time. Other methods, such as kFOIL, **k**ernelized **f**irst **o**rder **i**nductive **l**ogic [7, 8] utilize a non-complete refinement operator. This means that the refinement operator is not able to search the entire refinement graph. Hence, the algorithms employing them are not guaranteed to find an optimal hypothesis. kFOIL further employs a beam search on top of the refinement operator, only keeping the top n performing refinements of a given clause. This restriction is somewhat natural as compromises are generally necessary in order to have reasonable execution times with real world data. However, in doing so, completeness of the search is compromised (i.e. not every hypothesis is available in the search and hence the optimal hypothesis could be overlooked).

This study is also the first to apply centered kernel target alignment (CKTA) to inductive logic programming. While Landwehr et al [7, 8] utilized KTA (i.e.

non-centered), the usage of centering is a key distinction as CKTA has been shown to be correlated to model accuracy while KTA has not [9]. Additionally, the first attempt by any community to utilize CKTA for the promotion of diversity in ensemble methods and for incest avoidance in GA is proposed.

In this thesis, we first discuss some background material which is fundamental to the ideas proposed herein. This background material includes an overview of refinement operators, logic programming, GLPS, kernels and kernel methods. After this scaffolding has been provided, the new ideas from this work are proposed. Finally, experimental results are provided, along with conclusions and recommendations for future work.

# CHAPTER 2

# Background

The background knowledge required in order to understand the methods presented herein are provided in this chapter. We first discuss logic programming. Next, we explain GLPS as defined by Wong and Leung [5]. Finally, we discuss kernel methods, and centered kernel target alignment.

## 2.1 Concepts of Logic Programming

In this section we discuss various aspects of logic programming. We first present refinement operators as a mathematical construct. Next, we introduce the basic concepts of logic programming. Then we define the subsumption order for clausal logic. Finally, we revisit refinement operators applied to logic programming which are defined using the subsumption order on clauses.

### 2.1.1 Refinement Operators

Before diving into the mathematical details of refinement operators, a brief intuition about what they are and how they are used is worthwhile. Refinements provide a means by which hypotheses can be generalized or specialized. Refinement operators are definitions of how these generalizations and specializations occur. We can use refinement operators to induce graphs of hypotheses. In the context of inductive logic programming, the nodes represent clauses and the edges represent refinements (i.e. an edge exists from clause A to clause B if there is a refinement from clause A to clause B). Generalization occurs by moving upwards in the refinement graph and specialization occurs by moving downwards in the refinement graph. See, for example, Figure 1.

With little additional background in inductive logic programming, a discussion

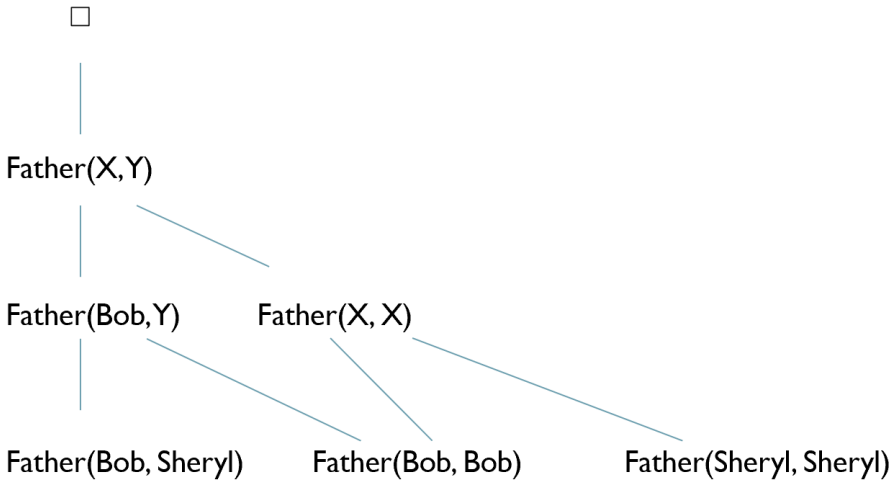


Figure 1: Refinement Example

of refinement operators as a formal mathematical construct can take place. After presenting refinement operators in the strict mathematical sense, the basic concepts of logic programming will be necessary in order to thoroughly understand the applications of the construct in the field of inductive logic programming.

Refinement operators provide a means of specializing or generalizing a hypothesis and are referred to as either downward or upward refinement operators accordingly. This can be thought of as walking up or down the refinement graph. In order to provide a means of comparison between clauses, and thereby a means to assign meaning to "up and down the refinement graph", quasi-orders are used. A quasi-order is a relation $R$, on a set $S$, which is reflexive and transitive. Then $< S, R >$ is said to be a quasi-ordered set. Some of the basis characteristics of a relation $R$ on a set $S$ are as follows [10]:

1. $R$ is reflexive if for all $x \in S, xRx$ holds

2. $R$ is symmetric if for all $x, y \in S, xRy$ implies that also $yRx$

3. $R$ is transitive if for all $x, y, z \in S, xRy$ and $yRz$ implies $xRz$

4. $R$ is antisymmetric if for all $x, y \in S, xRy$ and $yRx$ implies that $x = y$

Note that quasi-orders on sets require characteristics (1) and (3) only. Quasi-orders differ from partial orders (their more popular relative) in that they leave out (4). So, while a partial order is a quasi-order, the converse is not true. Quasi-orders are often denoted by the symbol $\succeq$. It is worth noting that quasi-orders can be turned into partial orders by defining an equivalence relation, $\approx$, on the set of interest, where $x \approx y$ if and only if $x \succeq y$ and $y \succeq x$.

Given that we have a quasi-order, we can define a refinement operator. If $< S, \succeq >$ is a quasi-ordered set, then a function $\rho$ such that $\rho(D) \subseteq \{E \mid D \succeq E\}$ for every $D \in S$ is referred to as a downward refinement operator. Upward refinement operators are defined similarly where $E$ and $D$ trade places in the set ordering (i.e. $\rho(D) \subseteq \{E \mid E \succeq D\}$). An ideal downward refinement operator is one which is locally finite, complete, and proper [10]. These concepts are defined as follows:

1. $\rho$ is locally finite if for every $D \in S$, $\rho(D)$ is finite and computable

2. $\rho$ is complete if for every $D, E \in S$ such that $D \succ E$, there is an $F \in \rho*(D)$ such that $E \approx F$ (i.e. $E$ and $F$ are equivalent under $\succeq$) where $\rho*$ is the set of all refinements (this effectively means that every specialization is reachable)

3. $\rho$ is proper if for every $D \in S$, $\rho(D) \subseteq \{E \mid D \succ E\}$ (avoid the case where repeated application of the operator generates equivalent clauses - i.e. gets stuck)

In the context of this research, the quasi-orders and refinement operators of interest will be defined on clauses. The two most popular orderings defined on clauses are the subsumption order and the implication order. We will focus solely on the subsumption order, as subsumption between clauses is decidable [10]. Furthermore, it is possible to create a complete and locally finite refinement operator for languages which have a finite number of constants, function symbols,

and predicate symbols via the subsumption order (which will be explained shortly). The refinement operator is used to induce a refinement graph on the set of clauses where an edge would exist between clauses $D$ and $E$ if $E \in \rho(D)$.

### 2.1.2 Basic Concepts of Inductive Logic Programming

In order to explain refinements in this context more fully, subsumption should first be defined. In order to understand subsumption, a discussion regarding some basics of first order logic are necessary. Consider the following simple example:

**Example 1 - Father Program**

*Program*

Father(X,Y) $\leftarrow$ Parent(X,Y), Male(X)

*Facts*

Parent(Bob, Sheryl)

Male(Bob)

With the above information, we may want to know if Bob is Sheryl's father [i.e. Father(Bob, Sheryl)?]. A definite clause is one which only contains negative clauses and can be thought of as a clause which does not have a head. Definite goals are also referred to as queries as they can be used to query the knowledge base (i.e. background knowledge, hypothesis, etc.) in order to see if a given statement, or a given conjunction of statements, is true. The query about whether or not Bob is Sheryl's father could be resolved as follows:

1. start with the clause (the only one in this program): Father(X, Y) $\leftarrow$ Parent(X,Y), Male(X)

2. notice that Male(Bob) can be resolved with Male(X) in the father clause leading to the substitution of X with Bob: Father(Bob, Y) $\leftarrow$ Parent(Bob,Y) (using Male(Bob), X/Bob)

3. resolve Parent(Bob, Sheryl) with the clause from (2), leading to the substitution of Sheryl for Y: Father(Bob, Sheryl) (using Parent(Bob, Sheryl), Y/Sheryl)

4. the proof is complete!

So, Bob is Sheryl's father after all. This deduction can also be visualized as shown in Figure 2:

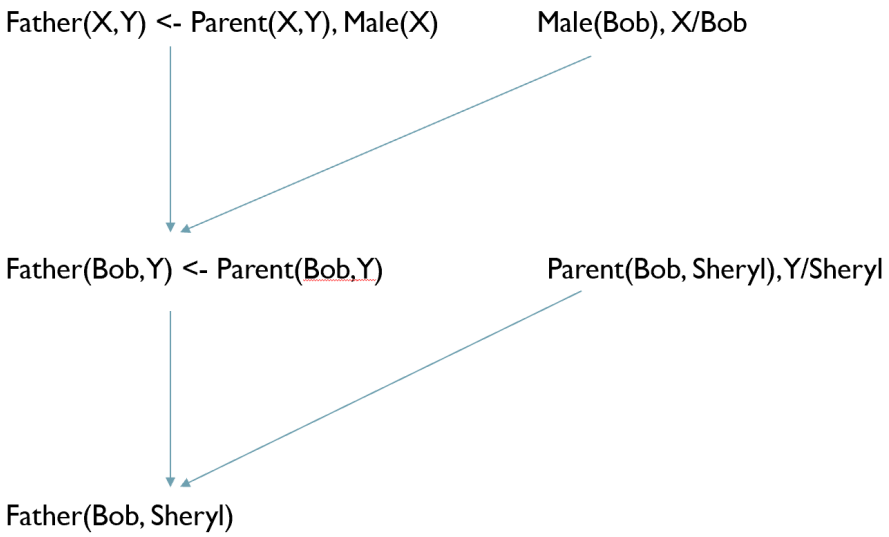


Figure 2: Deduction of Father(Bob, Sheryl)

With logic programming, we basically search the program rules and facts to see whether or not our questions are true. This is a very simple example for the sake of demonstration. Using this example, we can define some basic terms of logic programming. Father, Parent, and Male are predicates. Bob and Sheryl are constants. X and Y are variables. Terms are constants, variables, or n-ary functions (also known as functors) with n-terms specified as input. Atoms are predicates with their places filled in by terms [i.e. Parent(Bob, Sheryl), Parent(X,Y) , etc.]. Literals are positive or negative atoms. Father(Bob, Sheryl) is ground (since there are no variables). A clause is a finite disjunction of zero or more literals. Hence,

an example clause might take the form $\neg\phi \vee \neg\gamma \vee \sigma$, where $\neg\ \phi$, $\neg\ \gamma$, and $\sigma$ are literals with $\neg$ being the negation operator. If we note that the truth tables for $(\neg\ \phi \vee \gamma)$ and for $(\phi \rightarrow \gamma)$ are the same, we can rewrite the example clause as follows: $\phi \wedge \gamma \rightarrow \sigma$. This flip in interpretation is common in ILP literature (i.e. the flip from disjunctions to implications - note that the disjunctions are also sometimes viewed as sets of literals). The equivalence of these interpretations is worth a strong mental note for anyone who is interested in logic programming. In logic programming, it is common to (1) replace the conjunction symbol with a comma and (2) place the positive literals on the left, which in the example here would result in: $\sigma \leftarrow \phi, \gamma$. Here the conjunction of negative literals on the right is referred to as the body of the clause and the positive literal on the left is referred to as the head of the clause. The definition of Father is a definite clause (one positive literal in the head of the clause and zero or more negative literals in the body). A Horn clause is either a definite clause or a definite goal, where a definite goal is a clause with only negative literals (this can be thought of as a clause which does not have a head). Definite goals are also referred to as queries (as previously mentioned). We will also formally define a substitution $\theta$ as a set $\{x_1/t_1, x_2/t_2, \ldots, x_n/t_n\}$ where term $t_i$ is substituted for distinct variable $x_i$. The pairing $x_i/t_i$ is a binding for $x_i$.

Inductive logic programming (ILP) utilizes a set of positive and negative examples, along with background information in order to produce a hypothesis, typically a set of human readable clauses, which implies all positive examples (completeness) and no negative ones (consistency). A hypothesis which is complete and consistent is said to be correct [10]. Note that the examples and the background information are typically provided as human readable clauses as well. While the definition of ILP is not strictly required for the definition of subsumption, it is very important

to this study, as this study aims to provide a new approach to ILP.

### 2.1.3 Subsumption Order

With these definitions in place, we can describe the subsumption order. For clauses $C_1$ and $C_2$, we say that $C_1$ subsumes $C_2$, denoted by $C_1 \succeq C_2$, if there exists a substitution $\theta$ such that $C_1\ \theta \subseteq C_2$ (meaning that all literals in $C_1\ \theta$ also appear in $C_2$). $C_1$ properly subsumes $C_2$, denoted by $C_1 \succ C_2$, if $C_1 \succeq C_2$ and $C_2 \not\succeq C_1$. The clauses are subsume equivalent, denoted by $C_1 \sim C_2$, if $C_1 \succeq C_2$ and $C_2 \succeq C_1$. Note that this definition was taken from [10]. The definition is clearly reflexive (using the identity substitution) and transitive (since if $C_1 \succeq C_2$ and $C_2 \succeq C_3$ by substitutions $\theta_1$ and $\theta_2$ respectively, then applying $\theta_1$ to $C_1$ and applying $\theta_2$ to the result would yield $C_1 \succeq C_3$ - note that the same result would occur by simply applying the composition of the substitutions, e.g. $\theta_1\ \theta_2$, directly to $C_1$). Hence, we have a quasi-order defined on clauses. As an example, Father(x, y) $\succeq$ Father(Bob, Sheryl) since with the substitution { x/Bob, y/Sheryl }, the first clause actually becomes the second one (i.e. clearly all literals of the first clause, Father(x,y) , are represented in the second clause, Father(Bob, Sheryl), after performing the substitution). As another example, P1(x) $\succeq$ P1(a) $\vee$ P2(x), using the substitution { x/a } as the substitution yields P1(a) which is a subset of the right-hand side. As a final example, the empty clause subsumes all other clauses.

### 2.1.4 Refinement Operators Revisited

Using the quasi-order defined by the subsumption order, we can define a complete and locally finite refinement operator for languages which have a finite number of constants, function symbols, and predicate symbols. We will only define the downward operator, $\rho$, as the upward operator is similar. The following four rules

for a clausal language **C** follow from [10], although it was first defined in [11]. Note that the rules are for some clause $C$ in **C**.

1. For each variable $x$ in $C$ and each n-ary function symbol $f$ in $C$, $\rho(C)$ contains $C\{x/f(z_1, z_2, \ldots, z_n)\}$ where $z_1, z_2, \ldots, z_n$ do not appear in $C$. In other words, you can replace variables with the most general functions (since functions are more specific than variables).

2. For each variable $x$ in $C$ and each constant $a$ in $C$, $\rho(C)$ contains $C\{x/a\}$. In other words, you can replace variables with constants (since constants are more specific than variables).

3. For distinct variables $x$ and $y$ in $C$, $\rho(C)$ contains $C\{x/y\}$. In other words, you can change some variable in a clause to match some other variable already appearing in the clause (since this is a valid substitution and subsumption is defined in terms of substitutions).

4. For each n-ary predicate $P$ in $C$, $\rho(C)$ contains $C \cup \{P(z_1, z_2, \ldots, z_n)\}$ and $C \cup \{\neg P(z_1, z_2, \ldots, z_n)\}$ where $z_1, z_2, \ldots, z_n$ do not appear in $C$ and where $\neg$ indicates negation. In other words, you can add most general literals (since these will lend to more specific clauses).

The proof that this downward refinement operator is both complete and locally finite for languages which have a finite number of constants, function symbols, and predicate symbols is outside the scope of this work. However, [10] can be consulted for the proof. For this study we will simply utilize these results. An example refinement graph from [12] is provided in Figure 3.

Most ILP algorithms compromise in the search for an ideal hypothesis by using non-complete refinement operators (i.e. operators which cannot search the whole space) since the hypothesis space is potentially so large (potentially infinite).

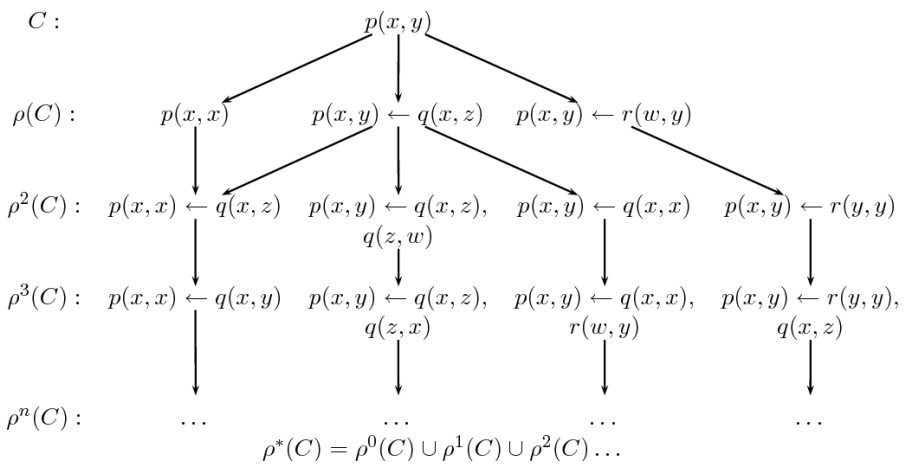


Figure 3: Example refinement graph. Note that some of the refinements in the above graph apply more than one of the rules for a complete, locally finite, downward operator in a single refinement step.

FOIL [13, 14, 15] is one such algorithm [16]. Even modern kernel methods such as kFOIL use refinement operators such as those proposed by Quinlan many years prior [7]. In these systems, non-complete refinements are performed on clauses in order to improve a theory.

These refinements, while possibly performing locally optimally, sometimes result in a less effective theory as the interaction between clauses as a whole (i.e. the global theory) is not considered [17]. Sometimes, combinations of locally non-optimal clauses may be more effective globally. This limitation may be overcome, or the effects of it mitigated to some extent, by the crossover component of the genetic algorithm proposed in this study.

### 2.2 Genetic Logic Programming System (GLPS)

GLPS [5] is the **g**enetic **l**ogic **p**rogramming **s**ystem. Genetic algorithms follow an evolutionary scheme and typically start with a seed population of solutions which are refined through successive generations. Each successive generation is produced by breeding the more promising solutions of the previous generation and mutating them slightly. The breeding is typically referred to as crossover

and allows for good solutions to be combined into potentially better solutions. Mutations allow pieces of the solutions to be changed, essentially adding new genetic material into the search space of the evolutionary scheme. Following the nomenclature of biological evolution, the promising solutions are identified by a fitness function (enforcing the idea of survival of the fittest). The process (i.e. reproduce and mutate current population to create the next generation, calculate the fitness of the members of the new generation, use the fitness to select hypotheses for reproduction in next generation) continues from generation to generation until some stopping criterion is reached, typically either some maximum number of generations or achieving a target fitness score. Genetic algorithms are typically used to find approximate solutions to optimization problems.

The genetic algorithm proposed in GLPS only utilizes crossover (i.e. no mutation). In GLPS, a hypothesis is treated as a forest of AND-OR trees. The AND trees represent individual clauses in the hypothesis. The OR trees represent a target concept. In other words, the AND trees represent one way some concept can be true (i.e. one clause with the target concept as its head) while the OR trees indicate all the ways that the same concept can be true. A group of OR trees (i.e. for all target concepts) represents the entire hypothesis. Note that the AND trees are typically sub-trees of the OR trees. For example, a clause (AND tree) might be: R(x,y) $\leftarrow$ P(x,y), S(y). This will create an AND tree with R(x,y) as the root and P(x,y) and S(y) as the leaves. A target concept, along with its associated AND-OR tree, might be as follows:

**Example 2 - AND-OR Tree**

R(x,y) $\leftarrow$ P(x,y), S(y)

R(x,y) $\leftarrow$ T(x,y), U(x)

R(x,y) $\leftarrow$ V(x,y), U(x), Q(y)

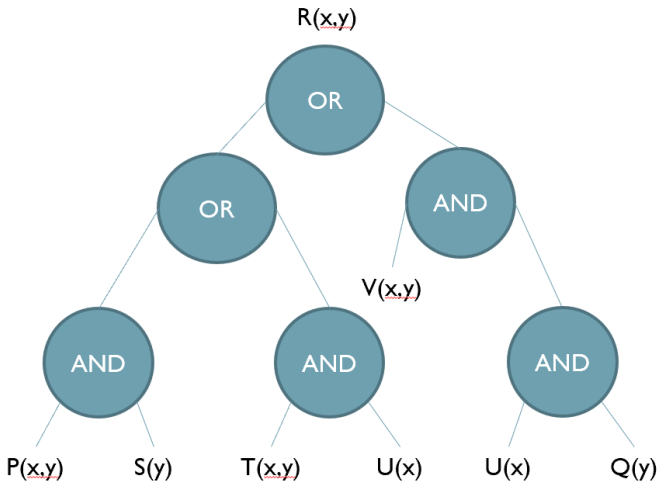


Figure 4: AND-OR tree representing the clauses above.

A hypothesis would consist of a forest of these AND-OR trees, one for each target concept. In GLPS, the initial population for the genetic algorithm would consist of a number of such hypotheses and was created either randomly using the symbols from the problem at hand or by running some variant of FOIL [13]. The fitness function in GLPS was simply the weighted accuracy on the training set (i.e. if there were 100 examples, 10 positive and 90 negative, the positive examples would get a weight of 9 and the negative examples a weight of 1). This would be applied to each member of the population and then crossover would be performed using the fitness score to select hypotheses for breeding (as described earlier). Note that each member of the population in this context represents a candidate solution (hypothesis) for the ILP problem under consideration.

In order to understand the crossover approach of GLPS, let us define a rule as an AND-OR tree for a target concept [such as the one for R(x,y) depicted in Figure 4]. Then, crossover is defined in terms of lists of numbers, from the empty list to lists with three numbers. The empty list {} refers to the whole logic program. The list $\{m\}$ refers to the $m$th rule of the program. The list $\{m, n\}$ refers to the clause or set of clauses specified by the $n$th node of the $m$th rule. The list $\{m, n, l\}$ refers

to the literal or set of literals specified by the $l$th node of the $n$th clause of the $m$th rule. These lists represent the four different types of crossover in GLPS. Lists of each length are given different probabilities of occurrence. Note that the empty list {} would mean identical reproduction (i.e. if two hypotheses performed a crossover based on the list {}, this would indicate that both hypotheses survived fully intact to the next generation). List such as $\{m\}$ represent a rule swap. Lists of the form $\{m, n\}$ change which clauses go into the rules (and could change the number of clauses also) and lists of the form $\{m, n, l\}$ change which literals go into the clauses (and could change the number of literals in the clause also). Note that two lists from parent programs are only compatible when they have the same number of elements, implying that these are the only possible points where crossover can occur [6].

The shortcomings of GLPS were in that it did not allow for mutation and it used a simple fitness function, the weighted accuracy of the learned hypothesis. By not allowing for mutation, the genetic algorithm is "stuck" with the genetic material that it was given in the first generation and is only allowed to shuffle this information around into potentially more useful genes (logic clauses in the context of this study). The simple fitness function also does not provide confidence in the generalization of the learned hypotheses. This study will address these weaknesses by adding a refinement operator for mutating theories and by utilizing centered kernel target alignment as the fitness function for the genetic algorithm. Cortes et al [9] have shown that high centered kernel target alignment values correlate with hypotheses which generalize well.

### 2.3 Kernel Methods

In order to understand centered kernel target alignment, we should first understand kernels. Kernels are mathematical constructs which appear in both func-

tional analysis (theory) and in statistical learning theory (application). At the highest level, they are functions which calculate the value of an inner product in a feature space created by a mapping function applied to data. They themselves operate on the data directly (i.e. in the input space). Because these functions are performed directly on the data, the data does not actually need to be mapped into the feature space. However, the kernel is guaranteed to calculate the value of the inner product in the space defined by the mapping. This is wonderful news, especially in terms of computation requirements. Stated more formally, a kernel is a function that takes the following form for all $x, y \in X$:

$$k(x, y) =< \phi(x), \phi(y) > \tag{1}$$

Note that here that $\phi$ is a mapping from the input space $X$ to a feature space (where an inner product can be defined). The most well-known kernels are the linear kernel, the polynomial kernel, and the radial basis kernel. Clearly, kernels need not be linear. This affords a chance to solve problems which may not be solvable in a linear space in some other non-linear space.

Some popular algorithms can be expressed in terms of a dot product. The dot product is an inner product and actually *is* the linear kernel. If we express problems in terms of dot products, we can exchange the dot product with an inner product, and further replace this inner product with a kernel function. Then, the resulting algorithm can handle non-linear data! Furthermore, this can be done without mapping the data explicitly into the feature space, but rather implicitly through the kernel function (which acts on the input space)! This is known as the kernel trick [18, 19]. This trick can be used to develop kernel methods for principal component analysis, canonical correlation analysis, Fischer discriminant analysis, ridge regression, spectral clustering, and more [20]. One of the most popular kernel

methods is the support vector machine which can be used in various capacities, the most popular being classification and regression.

Kernel methods are very powerful and are popular due to their ability to handle nonlinear data. In fact, the data input to a kernel function need not be numeric. Kernels can be defined on structured data such as graphs, trees, etc. [21]. They have even been defined on words. How is this possible? The kernel function calculates a number representing the similarity measurement between two inputs from the space X. If we create a matrix containing the kernel function values for a sample of N inputs in X, we create an N x N kernel matrix (a specialized Gram matrix - since the inner product is replaced by the kernel function). Kernel matrices are positive, semidefinite matrices [20]. To show that a proposed kernel function is in fact a kernel, essentially amounts to showing that any kernel matrix constructed from the input space will result in a positive semi-definite matrix. Hence, so long as kernels defined on graphs, trees, logic clauses, words, etc. satisfy this criterion, they are, in fact, valid kernels.

The centered kernel target alignment (CKTA or centered KTA) for two kernel matrices K and K' is defined as follows:

$$\rho(K, K') = \frac{< K_c, K'_c >_F}{||K_c||_F ||K'_c||_F} \tag{2}$$

where $K_c$ is the centered K matrix, $< K_c, K'_c >_F$ is the Frobenius product and $||K_c||_F$ is the Frobenius norm, which is the square root of the Frobenius product of a matrix with itself [9]. This definition is different from kernel target alignment (KTA) in that *centered* kernel matrices are used. $\rho(K, K')$ takes on values in the interval $[0, 1]$. Note that the Frobenius product is the sum of all entries in the matrix formed by the Hadamard product [20]. It is also equivalent to $tr(K_c K_c'^T)$. Hence, $||K_c||_F$, the Frobenius norm, is the square root of $tr(K_c K_c^T)$. Noting that

$K$ is symmetric, $tr(K_c K_c^T)$ is equal to $tr(K_c^2)$, which is equal to the sum of the squared eigenvalues of the matrix $K_c$. If $K_c$ has eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, then $||K_c||_F = \sqrt{\lambda_1^2 + \lambda_2^2 + \ldots + \lambda_n^2}$. Hence, $||K_c||_F$ can be interpreted as the diagonal from the origin to the corner of the box formed along the eigenvectors of the matrix with lengths given by the associated eigenvalues. In this sense, the denominator normalizes the Frobenius product of the matrices. This may be familiar if we consider that the Frobenius product is an inner product. If we change the expression to simple vectors with the dot product (most popular inner product), then the left-hand side of the equation would be the cosine of the angle between the vectors. Borrowing this intuition, the kernel target alignment essentially provides a score for how well the kernel matrices are aligned in n-dimensional space. We can also note that the Frobenius product is essentially the dot product of the vectorized versions of the matrices, formed by appending the rows together into one large row.

The "centered" part of centered KTA comes from subtracting the expected value (i.e. mean) in the feature space for each input $x$ in the kernel computation. So, where the kernel $K$ would be computed per input pair as $\phi(x) \cdot \phi(y)$, the centered kernel, $K_c$, is computed as $(\phi(x) - \theta) \cdot (\phi(y) - \theta)$, where $\theta$ is given as $\frac{\sum_1^m \phi(x_i)}{m}$. This computation does not need to be performed explicitly (i.e. subtracting out the mean in feature space). Rather it can be performed using the following expression [9]:

$$K_c = \left[\mathbf{I} - \frac{\mathbf{1}\mathbf{1}^T}{N}\right] K \left[\mathbf{I} - \frac{\mathbf{1}\mathbf{1}^T}{N}\right] \tag{3}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{1}$ is a column vector of all ones, $N$ is the size of the kernel matrix (i.e. the kernel matrix has size $N$ x $N$), and $K$ is the original kernel matrix (not centered).

A high centered KTA leads to a model which generalizes well [9, 22]. In fact, Cortes showed that centered KTA generalizes better than KTA. Furthermore, Cortes showed that kernel target alignment (when not centered) does not correlate well with performance. The difference in the performance between non-centered and centered KTA is quite significant in some cases. Consider the following table from [9]. In the first row, the correlations of model accuracy with centered KTA are provided and in the second row, the correlations of KTA with model accuracy are provided. The results are based on well-known data sets from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/) and the Delve data sets (http://www.cs.toronto.edu/~delve/data/datasets.html). The same methods were used to produce both results, only the alignment score differed between the two experiments.

| | KINEMATICS (REGR.) | IONOSPHERE (REGR.) | GERMAN (CLASS.) | SPAMBASE (CLASS.) | SPLICE (CLASS.) |
|---|---|---|---|---|---|
| $\widehat{\rho}$ | 0.9624 | 0.9979 | 0.9439 | 0.9918 | 0.9515 |
| $\widehat{A}$ | 0.8627 | 0.9841 | 0.9390 | 0.9889 | -0.4484 |

Figure 5: Centered KTA, $\hat{\rho}$, and KTA, $\hat{A}$, vs model accuracy

As can be seen in the chart above, centered KTA consistently outperforms KTA with respect to correlation with accuracy, sometimes in a dramatic fashion (see the results on the splice data set).

In this study, centered KTA will be utilized in various capacities. The kernels used to calculate the KTA will be the same as the ones proposed for usage in kFOIL [8]. These kernels are fairly similar to those proposed by Muggleton et al in [23]. It is worth noting that kFOIL optionally employed KTA (*not centered*) in order to perform their beam search (if KTA was not selected, SVMs were trained instead). The beam search is performed by taking the top n performers of a refinement, where n is the beam width, and exploring them. Employing centered KTA in this beam

search could also improve kFOIL (although this improvement was not planned for this study). Before describing the methodology for the experimentation, we will describe the kernels created in support of kFOIL (which will also be used in this study).

Landwehr et al formed linear kernels as the number of clauses in the hypothesis which succeed on both examples supplied to the kernel. Polynomial kernels were formed by adding one to the linear kernel and raising it to a power. This may be most clearly conveyed via an example borrowed from [8] which in turn borrowed from [24]. This example is about the structure of molecules. Here bond(*compound, atm1, atm2, bondtype*) indicates that the *compound* has a bond of *bondtype* between atoms *atm1* and *atm2*. atm(*compound, atom, element, atomtype, charge*) indicates that in *compound*, *atom* has element *element* of *atomtype* and partial charge *charge*. For example, the following encodes the fact that atom d2_1 in compound d2 is an aromatic carbon atom with partial charge 0.067: atm(d2, d2_1, c, 22, 0.067) [24]. A subset of background information is given for molecules m1 and m2 for the sake of the example.

**Example background information for examples m1 and m2 ...**

*atm(m1, a1_1, c, 22, -0.11). bond(m1, a1_1, a1_2, 7).*

*atm(m1, a1_3, c, 22, 0.02). bond(m1, a1_3, a1_4, 7).*

*atm(m1, a1_26, o, 40, -0.38). bond(m1, a1_18, a1_26, 2).*

...

*atm(m2, a2_1, c, 22, -0.11). bond(m2, a2_1, a2_2, 7).*

*atm(m2, a2_3, c, 27, 0.02). bond(m2, a2_3, a2_4, 2).*

*atm(m2, a2_26, o, 40, -0.38). bond(m2, a2_18, a2_26, 7).*

**Assuming that both molecules are mutagenic, a possible hypothesis $H = c_1, c_2, c_3$ for the mutagenicity of the molecules for this domain might**

**be:**

1. $c_1$ = *mutagenic(X)* $\leftarrow$ *atm(X, A, o, 40, C)*

2. $c_2$ =*mutagenic(X)* $\leftarrow$ *atm(X, A, c, 22, C), atm(X, B, E, 22, 0.02)*

3. $c_3$ =*mutagenic(X)* $\leftarrow$ *atm(X, A, c, 27, C), bond(X, A, B, 2)*

Using the above background information and the hypothesis, example m1 is covered by the first and second clauses, while example m2 succeeds on the first and last clauses. For each example, we create a vector in the feature space spanned by the truth values of the hypothesis by creating an n dimensional vector, where $n$ is the number of clauses in the hypothesis, and then for the $i$th clause in the hypothesis, assign a value of 1 in the $i$th position in the feature vector if the background information together with the $i$th clause imply the example. This approach yields the following feature vectors for examples m1 and m2:

$\phi_H(m1) = (1, 1, 0), \phi_H(m2) = (1, 0, 1)$

Note that this approach performs the embedding into the feature space. Hence, we do not take advantage of the kernel trick (i.e. performing the inner product in the feature space without mapping to the feature space). If we use the above results, the simple linear kernel yields the following (with $K_L$ being the linear kernel defined by Landwehr et al [8, 7]):

$K_L(m1, m2, H) =< \phi_H(m1), \phi_H(m2) >= 1$

$K_L(m1, m1, H) =< \phi_H(m1), \phi_H(m1) >= 2$

$K_L(m2, m2, H) =< \phi_H(m2), \phi_H(m2) >= 2$

$\Rightarrow K_L(m1, m2, H) = \#ent_H(m1 \wedge m2)$, where $\#ent_H(f) = |\{c \in H | B \wedge c \vDash f\}|$

Phrased slightly differently, the last equality indicates that the result of the kernel is the number of clauses in H (the hypothesis) which, together with B (the background theory), logically entail f (note that $\vDash$ is the symbol for logical

entailment). The polynomial kernel, $K_P$, and the Gaussian kernel [also known as radial basis function (RBF) kernel], $K_{RBF}$, are defined similarly as:

$$K_P(m1, m2, H) = (\#ent_H(m1 \wedge m2) + 1)^p$$

$$K_{RBF}(m1, m2, H) = exp(-\frac{\#ent_H((m1 \vee m2) \wedge \neg(m1 \wedge m2))}{2\sigma^2})$$

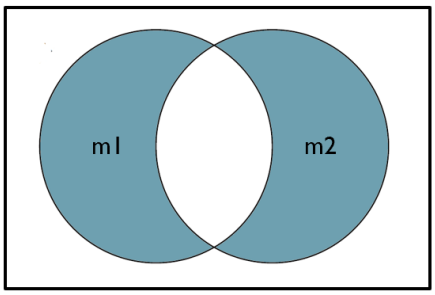


Figure 6: Symmetric difference between sets m1 and m2 highlighted in aqua

The numerator in the power of the exponential equation in the RBF kernel is the symmetric difference between the sets of clauses in the hypothesis which entail the different examples. Since the kernels are defined in terms of sets of clauses which entail both examples and since the feature mappings are essentially indicator functions showing whether or not each clause in the hypothesis entails the example under consideration (i.e. can be viewed as defining the set of clauses in the hypothesis that entail the given example), this can be seen as the natural application of the RBF function in this context. Instead of expressing a difference between real numbers the symmetric difference of the sets is used. The kernels described above will be utilized in this study; however, they will be centered.

We know from the closure properties of kernels that if $k_1$ is a valid kernel, $\phi : X \longrightarrow \mathbb{R}^N$, then $k(x, z) = k_1(\phi(x), \phi(z))$ is also a kernel [20]. With this in mind, we can define the mapping $\phi_{H,B}(x)$ for a logical hypothesis $H$ with $n$ clauses and background knowledge $B$ as follows:

$$\phi_{H,B}(x)_i = \begin{cases} 1, & \text{if } B \cup c_i \vDash p(x) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The above mapping will map each example to a vector in $\{0,1\}^n$, where a 1 occurs in position $i$ if clause $i$ (i.e. $c_i$) from hypothesis $H$, along with background $B$, implies the target predicate $p$ for example $x$. A zero will be in position $i$ otherwise. Note that $\{0,1\}^n \subset \mathbb{R}^n$. Hence, we can now employ any kernel $k_1$ to this $\phi_{H,B}$ mapping in order to create another valid kernel. Applying the following kernels as $k_1$ to the mapping $\phi_{H,B}$, lends to the kernels $K_L$, $K_P$, and $K_{RBF}$ defined above (with $\gamma = \frac{1}{2\sigma^2}$):

1. linear: $u'v$, no parameters
2. polynomial: $(u'v + 1)^{degree}$, parameter is $degree$
3. radial basis function: $exp(-\gamma * |u - v|^2)$, parameter is $\gamma$

In the remainder of this work, the mapping $\phi_{H,B}$ will be assumed, and we will refer to the kernels as linear, polynomial, and gaussian (RBF).

# CHAPTER 3

## Approach

As was eluded to previously, this study aims to employ CKTA to inductive logic programming in the following ways:

1. as a fitness score for genetic algorithms (GA)
2. as a means for promoting diversity
   (a) as a mechanism for incest avoidance in GA
   (b) for ensembles (member selection)

Note that GA could be replaced with any other stochastic search strategy which could benefit from the usage of a fitness score, or quality metric of sorts (i.e. Monte Carlo Tree Search, Beam search, etc.). Additionally, the diverse ensemble strategy discussed herein can be utilized for *any* kernel-based ensemble, not only the logic kernels which are the focus of this study.

As GA utilizes a fitness function and employs selection strategies for choosing parents for crossover, it was a natural fit for experimenting with these applications of CKTA; hence, this study, with the exception of the ensemble methods proposed, focuses on applying these strategies in a GA setting. We also aim to practically employ a complete refinement operator in our stochastic search. This application of a complete refinement operator is specific to the ILP setting (i.e. unlike the CKTA work proposed herein, which can be utilized for *any* kernel-based algorithm in any problem domain, the complete refinement operator can only be utilized in the ILP domain).

In this chapter, the key ideas from this research will be presented. First, we will discuss how the genetic logic programming system (GLPS) was modified.

Next, the novel approach to using centered kernel target alignment (CKTA) for promoting diverse ensembles will be presented. Finally, the language bias employed in this study will be discussed.

## 3.1 Modified GLPS

GLPS is utilized in this study as it provides a framework in which all of the ideas proposed herein can be applied. In this section, the modifications to GLPS, resulting in the GA utilized in this research, will be presented. At the highest level, the following figure conveys the search strategy, which is typical for GAs.

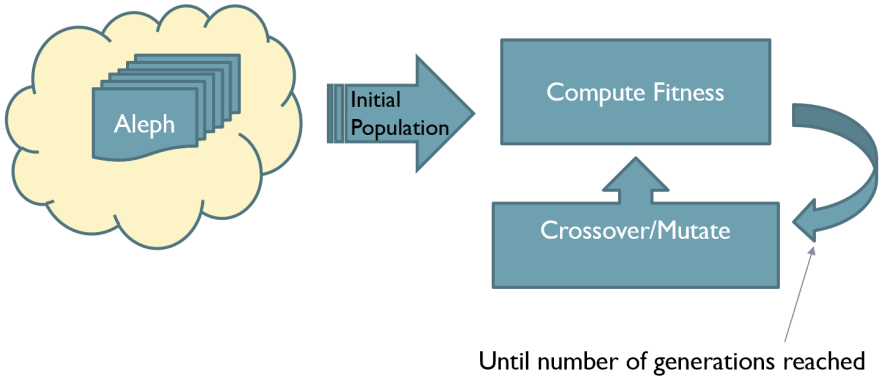


Figure 7: Basic GA Approach Utilized in this Research

In the following subsections, the various components of the GA utilized in this research will be expounded upon.

### 3.1.1 Initial Population

In this study, the initial population is created utilizing the state of the art ILP system Aleph (**A le**arner for **p**roposing **h**ypotheses) [25]. The different members of the population are created by simply shuffling the samples before presenting them to Aleph. As discussed in [26], shuffling the samples in this manner will cause Aleph to produce different hypotheses because of Aleph's greedy search approach. Aleph will cover the first sample provided to it and then add new rules as new uncovered

samples are provided. So, the order of the samples presented to it matters, a fact that we exploit to create the initial population. This contrasts with the usage of FOIL, **F**irst **O**rder **I**nductive **L**earner, utilized by GLPS. Note, however, that any logic programming system could be used to create an initial population. An initial population could even randomly be generated, with the only drawback being that the algorithm will likely take longer to converge on a promising solution in this case. Aleph was chosen because (1) it is a state of the art inductive learner and (2) it can put us "in the vicinity" of an optimal solution. Note that Aleph is an inductive logic programming system which has consistently been utilized for more than a decade [27, 7, 28] as a benchmark for comparison. A forest of AND-OR trees is created for this initial population in the same manner as is used in GLPS.

### 3.1.2 Scoring

After creation of the initial population, each member of the population has its fitness score calculated. Rather than setting the fitness function equal to the classification accuracy on the training data as in [5, 29], this study aims to use centered kernel target alignment (KTA) [9], along with a couple of other novel choices for scoring. In order to facilitate more diverse experimentation, the code for this research was set up to allow the choice of five different scoring functions for this fitness computation:

1. Accuracy
2. Weighted accuracy (this is what was used in GLPS)
3. CKTA
4. accCKTA (Accuracy * CKTA)
5. wAccCKTA (Weighted accuracy * CKTA)

Additionally, there were options to compute both accuracy (normal or weighted) and CKTA and use CKTA as the fitness while logging the accuracy. The CKTA can further be parameterized to utilize one of the following kernels, along with their pertinent kernel parameters (note that the sigmoid kernel, while supported in the code, was not utilized during experimentation):

1. linear: $u'v$, no parameters
2. polynomial: $(\gamma u'v + coef)^{degree}$, parameters are $\gamma$, $coef$, and $degree$
3. radial basis function: $exp(-\gamma * |u - v|^2)$, parameter is $\gamma$
4. sigmoid: $tanh(\gamma u'v + coef)$, parameters are $\gamma$ and $coef$

These kernels were applied to the output of the $\phi_{H,B}$ defined in Section 2.3. For "labels" in the ILP setting, we will assign positive examples a label of "+1" and negative examples a label of "-1". Hence, the target matrix (for the KTA) will consist of "+1" and "-1" values as the target matrix is the outer product of the label vector for the sample.

It is worth noting that accCKTA and wAccCKTA defined above are also perfectly valid scoring functions, with the nice property of being in the interval $[0, 1]$. If there are several scores, or fitness functions, $s_1, s_2, ..., sn$, which all produce values lying in the interval $[0, 1]$, then the product of those scores also lies in the interval $[0, 1]$ as is shown in Equation 5. This is true because (1) both $0^n$ and $1^n$ are equal to 0 and 1 respectively when $n \in \mathbb{N}$ and (2) 0 and 1 represent the minimum and maximum values of the score function respectively.

$$\prod_{i=1}^{n} s_i \in [0, 1], \text{when } s_i \in [0, 1] \forall i \tag{5}$$

Note that if all $s_i$ were in $[a, b]$, rather than $[0, 1]$, then the product above would lie in the interval $[a^n, b^n]$, as shown in Equation 6.

$$\prod_{i=1}^{n} s_i \in [a^n, b^n], \text{when } s_i \in [a, b] \forall i \tag{6}$$

Using centered KTA to guide the search for an optimal hypothesis is explored in this study with the hope that it will aid in the discovery of a hypothesis which generalizes well. Scores such as accCKTA are also very interesting, being the product of the accuracy of the hypothesis (i.e. standalone as a logic program) and the CKTA (which should lend to an accurate kernel-based classifier). accCKTA is particularly interesting in that it strikes a balance between finding a good logic hypothesis which is sufficient in its own right and finding a feature space induced by the configured kernel which has the capability to generalize well.

After this initial population has been created and scored utilizing the above parameters, the GA can attempt to find increasingly optimal theories, hopefully pulling us out of any local maxima in which the initial hypotheses may be trapped.

### 3.1.3 Crossover

In this study, we will utilize the same crossover approach as described for GLPS with two differences. First, the AND-OR trees will be shuffled prior to crossover so that all reasonable combinations of clauses/literals will be available for crossover. This is different from GLPS as no shuffling was included in GLPS, which limits the possible combinations of clauses/literals to be used for crossover between two hypotheses. For this reason, in the results section, the GLPS results are flagged with a star (i.e. GLPS*), indicating that the algorithm has been enhanced with shuffling for crossover. The second difference is that parent hypotheses (programs) will be chosen differently based on configuration, as indicated in Algorithm 1.

Note that approach appearing in the else statement of Algorithm 1 is typical for GA and is the mechanism utilized by GLPS. The approach in the if statement (i.e. when *incestAvoidanceEnabled* is true) is novel to this research and will be

**Algorithm 1:** Crossover Approaches

**Data:**
$f_i$, $0 \leq i < m$, the fitness scores for each of the $m$ hypotheses
$h_i$, $0 \leq i < m$, the $m$ hypotheses
**Result:** Parent hypotheses for crossover
Select parent hypothesis one, $P_1$, randomly, but proportional to fitness (i.e. choose hypothesis $h_i$ with probability $\frac{f_i}{\sum_{j=0}^{m-1} f_j}$)
Suppose that $h_a$ was chosen as $P_1$ (i.e. index $a$ was selected).
**if** *incestAvoidanceEnabled* **then**
    Adjust scores for each other hypothesis (i.e. those which are not $P_1$) by dividing its original score by the CKTA between it and parent $P_1$ as follows: $adjusted\_score(h_i) = \frac{score(h_i)}{\rho(h_i, P_1)}, i \neq a$
    Choose parent hypothesis $P_2$ randomly, but proportional to the $adjusted\_score$ defined above (i.e. choose hypothesis $h_i$ with probability $\frac{adjusted\_score(h_i)}{\sum_{j=0, j\neq a}^{m-1} adjusted\_score(h_j)}, i \neq a$)
**else**
    Choose parent hypothesis two, $P_2$ randomly from the remaining hypotheses in the generation, but proportional (i.e. choose hypothesis $h_i$ with probability $\frac{f_i}{\sum_{j=0, j\neq a}^{m-1} f_j}, i \neq a$) to fitness
**end**
**return** $(P_1, P2)$

described further in the following subsection.

### Using CKTA for Incest Avoidance

The approach in the if statement (i.e. when *incestAvoidanceEnabled* is true) of Algorithm 1 is a way to select the parent hypotheses for crossover using a novel twist proposed in this research. This twist assists in incest avoidance [i.e. breeding between two very similar (i.e. sibling) hypotheses]. We would like to maintain a more diverse population of hypotheses in order to encourage a more optimal result, noting that this could lead to a useful ensemble of hypotheses when the algorithm terminates [30]. Intuitively, if each population contains nothing but very similar hypotheses, then the search likely will not "explore new territory" as the genetic algorithm is provided very similar genetic material from each of the hypotheses

in this case. An approach using all similar hypotheses will also be more likely to get stuck in a non-optimal solution, hence why we would like to encourage diversity in our populations. Centered KTA can be used in this capacity as well. The hypotheses chosen for crossover during creation of the next generation can be chosen such that they are diverse (i.e. have varying centered KTAs, which can be enforced by choosing hypothesis which do not align well with each other) but are in alignment with the target concept [i.e. the kernels for both hypotheses align with the target kernel matrix (produced via an outer product of the labels), meaning that their centered KTA with respect to the target kernel matrix is high but their centered KTA with respect to each other is low].

To accomplish this, when selecting parent hypotheses for crossover, we first select one hypothesis randomly, proportional to the fitness. Call this selected parent $P1$. Then, we adjust the score of the other hypotheses, essentially adding a reward for being different from the already selected parent, $P1$. The score for hypothesis $H_i$ would be updated as follows:

$$adjusted\_score(H_i) = \frac{score(H_i)}{\rho(H_i, P1)} \tag{7}$$

Note that $\rho(H_i, P1)$ is the centered KTA between hypothesis $H_i$ and the already selected parent $P1$. Recall that centered kernel target alignment is a similarity measure which takes on values between zero and one. When closer to one, it indicates that the two kernels are very similar and when closer to zero, it indicates that the two kernels are very different. Hence, the adjusted score essentially boosts the score of the other hypotheses based on being different from the already selected parent. Once this adjustment occurs, the second parent, $P2$ is selected from the remaining hypotheses in proportion to their adjusted score. Once the two parents have been selected, crossover is performed as described in GLPS. In

this study, the adjusted scores are then discarded for selection of the next set of parents (i.e. we go back to the original fitness scores from before any parents were selected). Continuously adjusting the scores during crossover (i.e. not resetting between selection of sets of parents) would make for an interesting future study.

It is worth noting that the *adjusted_score* above can be considered a special, degenerate case of the diversity adjusted score $\gamma$, as described in Equation 8 (defined in section 3.2.1 where we are only selecting two members, $\alpha$ is set to 0, and $\nu$ is set to 1). As was eluded to, experimenting with other forms of $\gamma$ for incest avoidance could provide for an interesting future study.

**Elitism**

Elitism was utilized in the GA created for this research. Elitism consists of ensuring that the top performing hypotheses in a generation is given a spot in the next generation of hypotheses (i.e. the top performing hypotheses will simply be cloned into the next generation). By even just allowing the top performer to be carried to the next generation, the maximum fitness score is guaranteed to increase monotonically from generation to generation. In the code base, a configuration parameter specifying the proportion of members to be considered elite is included. Typically, a small number of hypotheses are considered elite so that the majority of the next generation is created via crossover.

### 3.1.4 Mutation

Recall that the two shortcomings of GLPS were in its fitness score and its lack of mutation. Improvements to the fitness score have already been described (i.e. CKTA, accCKTA, wAccCKTA, etc.). To address the lack of mutation, we will also allow for mutation via a randomly applied complete and locally finite refinement operator, allowing new genetic material into the mix and likely allowing for the

discovery of better solutions.

Mutation will include randomly applied complete, locally finite, upward and downward refinements of clauses (by randomly applying one of the rules of the downward and upward refinement operators to clauses). The refinement operators used will be based on the subsumption order as described in 2.1.4. Recall that these operators are both complete and locally finite assuming that we have a finite number of constants, function symbols, and predicate symbols. The mutation approach employed in the GA in this study is described Algorithm 2.

**Algorithm 2:** Randomly applied complete refinement operators

**Data:** $h_i$, a child hypothesis resulting from crossover, $0 \leq i < m$
**Result:** Mutated Hypothesis
`/* Mutation is done by randomly applying a complete, locally finite, refinement operator (either upward or downward). Note that all probabilities are configurable. */`
Suppose there are $n$ clauses in hypothesis $h_i$
**for** *j in 0 ... (n-1)* **do**
  *doMutation* $\longleftarrow$ randomly set to true with probability $P_m$
  **if** *doMutation* **then**
    *isUpward* $\longleftarrow$ randomly assigned with probability $P_u$
    **if** *isUpward* **then**
      select refinement type from constant, variable, or literal removal using configured probabilities (approximately equal in this study)
      given the refinement type, choose the parameters for the upward refinement *and* perform the upward refinement on clause $c_j$ of hypothesis $h_i$
    **else**
      select refinement type randomly from constant, variable, or literal addition using configured probabilities (approximately equal in this study)
      given the refinement type, choose the parameters for the downward refinement *and* perform the downward refinement on clause $c_j$ of hypothesis $h_i$
    **end**
  **end**
**end**

The GA approach in this study should provide an improvement over GLPS

because it includes complete upward/downward refinements (i.e. allows the search space the possibility of completeness) and because it uses centered KTA as a fitness function. The completeness of the search space should also make the approach competitive with other algorithms, such as [29], proposed by Muggleton et al, where a stochastic search is used to explore the hypothesis space on the fringe of the refinement graph under the subsumption order and GA is used to evolve and re-combine the clauses generated via this stochastic search; however, a comparison to Muggleton's stochastic search is not planned in this work. Comparisons of the approach proposed herein with [29], while not planned for this study, may be interesting future work.

To the author's knowledge, this work is the first employing a complete refinement operator in a practical manner. It is worth mentioning that the completeness of the refinement operators is subjected to some size restriction (i.e. we are only going to produce so many generations in the GA so we will not be able to search the entire search space). Furthermore, as the rules of the refinement operators will be randomly applied, not all refinements will be explored during each generation. Additionally, unlimited computing resources are not available, so in the case where the search space is infinite, one would be unable to search the entirety of the refinement graph in any practical setting. While these notes are not meant to indicate any expectation of impaired performance relative to other current methods, it is worth mentioning in order to level set expectations (i.e. despite the complete refinement operator, we could still arrive at a non-optimal solution). Regardless, it is believed that this approach will at least be competitive with the current state of the art, if not improving it.

### 3.1.5 Terminal Conditions for the Search

Once the last generation of the GA has been reached, the hypothesis with the highest centered KTA will be selected as the final hypothesis. Currently, reaching the last configured generation is the only stopping criterion of the algorithm. Whether or not adding a sufficient score criterion would be beneficial is debatable; however, this will likely be included in the future to allow the opportunity to stop early, should the sufficient score be reached. Once the final hypothesis is selected, it will be evaluated on the test data in order to assess its quality. Depending on the scoring type selected, appropriate measures can be taken. If CKTA was used, an SVM can be created to classify the samples used for training. If accuracy or weighted accuracy was chosen, the hypothesis accuracy can simply be computed from the resulting logic program in prolog, utilizing Aleph (note that Aleph "sits" on top of Yap prolog).

### 3.1.6 Dynamic Propositionalization

This study employs dynamic propositionalization. This is similar to Landwehr et al's nFOIL [31] and kFOIL [8, 7] algorithms. It contrasts with Muggleton et al's support vector inductive logic programming (i.e. SVILP) [23], which utilizes static propositionalization. Static propositionalization occurs when a set of features is learned for the data and then a classifier (or another statistical model) is built using this feature set (after the feature set has been created). In dynamic propositionalization, the set of features for classification are jointly optimized with the classifier [7].

When utilizing a score including CKTA, the GA proposed herein learns a feature set which results in a high CKTA. In other words, features are learned which maximize CKTA. Per Section 3.1.2, we can also jointly optimize CKTA with the hypothesis' standalone accuracy as a logic program by using a hybrid scoring

functions, a nice benefit to the GA proposed herein. kFOIL is also able to perform dynamic propositionalization. However, it utilizes either KTA (less accurate) or support vector machines (SVMs) which are much more computationally expensive. kFOIL also utilizes a beam search and heuristic driven refinement operator. These limitations should give the GA proposed herein an edge over kFOIL in terms of performance. Dynamic propositionalizations are interesting in both this study and in kFOIL as they essentially entail learning a kernel for the data. In this study, the kernel is learned via a genetic algorithm (GA).

### 3.2 Ensemble Creation

For ensemble creation, two strategies are explored in this research. One is typical and one is novel to this research. We will assume that the ensemble consists of $m$ classifiers. The ensembles are created using the final population from a given GA run. The first strategy simply selects the top $m$ performing classifiers of the final population of the GA for usage in the ensemble. The second strategy selects the best remaining hypothesis not already included in the ensemble based on a compromise between the hypothesis' accuracy and the hypothesis' diversity with respect to the previously selected members of the ensemble until $m$ members have been selected for the ensemble. Both strategies employ a max voting scheme once the members of the ensemble are selected. Simple graphics are provided in Figure 8 and Figure 9 to illustrate the difference between the two strategies (note the labeling of the arrows).

The top $m$ classifier approach is straightforward and hence will not be discussed further. However, the diversity approach merits further discussion.

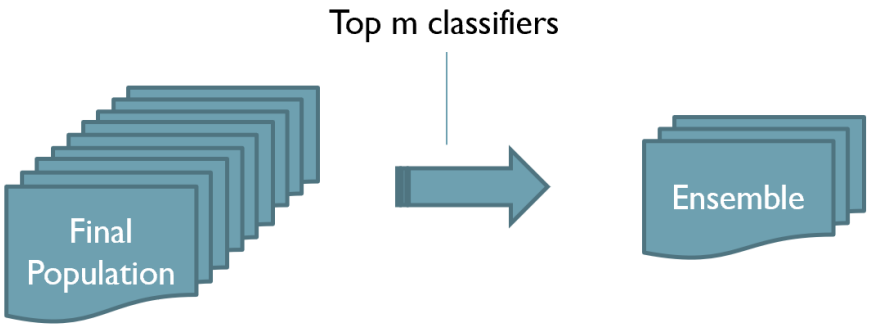


Figure 8: Ensemble Member Selection Using Top $m$ Classifiers

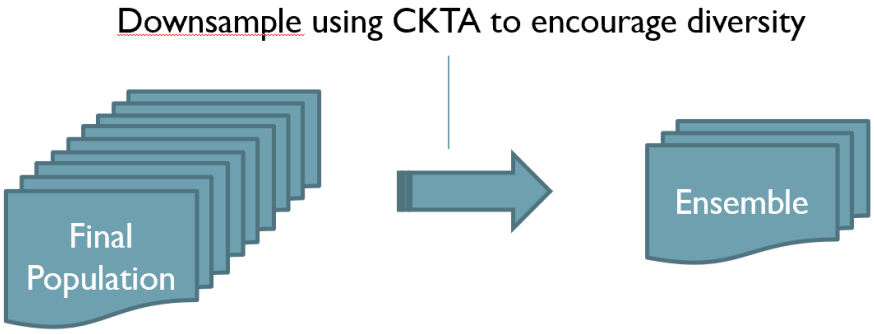


Figure 9: Ensemble Member Selection Based on Diversity

### 3.2.1 Diversity Adjusted Scoring for Ensemble Member Selection

Given that the ensemble already includes $k$ members of the population, the diversity adjusted score for hypothesis $H_i$, where $H_i$ is not already included in the ensemble is computed as follows:

$$\gamma(H_i) = score(H_i) * \prod_{j=1, a_j \in A, A \cap i = \emptyset}^{k} \frac{score(H_i)^{\alpha}}{\rho(H_i, H_{a_j})^{\nu}} \tag{8}$$

where $A$ is the set of indices of hypotheses already included in the ensemble, $a_j \in A$ is an index for a hypothesis already included in the ensemble, $\alpha \in \{0, 1\}$, $\nu \in \mathbb{R}$, and $\nu \geq 0$. $\nu$ is referred to as the diversity factor in this research. $score(H_i)$ is defined in Section 3.1.2 (i.e. accuracy, weighted accuracy, CKTA, accCKTA, wAccCKTA). Note that if $\alpha = \nu = 0$, we have the degenerate case where the adjusted score is simply the initial score. When $\alpha = 1$, we essentially add an addi-

tional penalty based on the initial score of the hypothesis. This should only be set to 1 in the case where $\nu \neq 0$ because otherwise, the initial score will unnecessarily end up being raised to a power. When $\nu$ is large, diversity is strongly encouraged as hypothesis $H_i$ is rewarded for being different from each of the hypotheses already included in the ensemble. Hence, $\alpha$ serves as a repeated penalty for having an initially bad score while $\nu$ rewards for being different from the hypotheses already included in the ensemble. In this manner, we can balance the performance of a hypothesis with its diversity with respect to other hypotheses during member selection. Setting $\alpha$ to one will help us to avoid the case where a hypothesis is very different but performs extremely poorly (since the diversity rewards will be offset by the performance penalty). It is worth noting that $\alpha$ need not be limited to $\{0, 1\}$. However, because it was limited in this fashion during this study, it was described in this fashion above.

### 3.3 Language Bias

In this study, we will bias our language to:

1. use horn clauses (see section 2.1.2)
2. be function-free
3. have a finite number of constants and predicate symbols

Understanding what function symbols are helps to understand what function-free means. Function symbols are mappings from terms to terms. This is a necessary distinction because functions map only to terms. They do not produce a valid formula in predicate logic, even when their terms are filled in (i.e. no true/false value). This contrasts with predicate symbols which do produce valid formulas. An example in the domain of natural numbers, in a setting where all natural numbers were constants in the language, could be $x$ getting mapped to $x^2$. It is worth

noting that an n-ary function can be mapped to a (n+1)-ary predicate symbol which takes the input and output arguments of the function and evaluates to true when the output argument is correct per the input arguments. This is the basis of Rouveirol's work [32], which showed that limiting languages to be function-free does not reduce the expressiveness of the language.

Restricting the study to languages of this nature is common in ILP research. It should be noted that the refinement operator used in this study (defined in Section 2.1.4) maintains its completeness in this setting, as these operators are both complete and locally finite assuming that we have a finite number of constants, function symbols, and predicate symbols. Making the language function-free also has the following benefits:

1. it makes the theories produced by the research decidable [10]

2. it does not reduce expressiveness of the language much as flattening can be used to transform functions into new predicate symbols [10, 32, 33, 34]

3. it removes the need for substitutions of the type $C\{x/f(z_1, z_2, \ldots, z_n)\}$, as was described in Section 2.1.4, in the refinement operator

# CHAPTER 4

## Experiments

Four data sets were used in the experimentation performed in support of this study, two mutagenesis data sets (retrieved from [35]) and two Alzheimer's data sets (retrieved from [36]). An overview of these data sets is provided in Table 1 below, following the overview style of [7]. These data sets were chosen as they are quite popular benchmark data sets for ILP studies. All of these data sets involve predicting properties of some set of compounds.

Table 1: Overview of all data sets used in experiments, including number of classes, number of available examples, accuracy of majority class predictor, number of relations that are used in rules, and the number of facts in the background knowledge

| Data Set | #Cls | #Ex | Maj. Class | #Rel | #Fact |
|---|---|---|---|---|---|
| Mutagenesis friendly | 2 | 188 | 66.50% | 4 | 10324 |
| Mutagenesis unfriendly | 2 | 42 | 69.10% | 4 | 2109 |
| Alzheimer amine | 2 | 686 | 50.00% | 20 | 3754 |
| Alzheimer toxic | 2 | 886 | 50.00% | 20 | 3754 |

During experimentation, 10-fold cross validation (CV) was used unless otherwise specified. The 10 folds for each data set used in these experiments are available at [37]. For the 10-fold cross validation, random assignment of compounds into approximately equally sized sets was performed. An elitism ratio of 0.1 (i.e. 10%) was used during experimentation. As the experiments utilized population sizes of 20 or 30, this elitism setting lead to 2 or 3 hypotheses respectively being considered elite during each experiment (i.e. the top 2 or 3 hypotheses in a generation were cloned into the subsequent generation by the GA). The goal of this study is to show that the new approach proposed herein is, at a minimum, competitive with the baseline modified GLPS, Aleph, and, when possible, kFOIL. Recall that Aleph is a

state of the art ILP learner frequently used in benchmark studies. An improvement over kFOIL's performance is expected since the new approach (1) uses a complete refinement operator and (2) utilizes centered KTA (versus either a trained SVM, which is more computationally expensive or simple KTA, which is less accurate, as is used by kFOIL). Recall that high centered KTA values imply models which generalize more effectively than those with a high KTA (non-centered). The subtle difference of centering makes a substantial difference with respect to performance [9].

### 4.1 Results Nomenclature

In the tables of results that follow, the following conventions are used for the names appearing in the 'Config' column:

1. if CKTA appears in the name, then centered kernel target alignment was used for the fitness, as described in Section 2.3

2. if Poly<k> appears in the name, then a polynomial kernel $K_P$ of degree <k> was used, as described in Section 2.3

3. if Gauss<k> appears in the name, then a gaussian kernel $K_{RBF}$ with a $\gamma$ value of <k> was used, as described in Section 2.3

4. if Linear appears in the name, then a linear kernel $K_L$ was used, as described in Section 2.3

5. if withIncestAvoidance appears in the name, then incest avoidance, as described in 3.1.3 was used

6. if AccCKTA appears, then the accuracy of logic hypothesis was multiplied by the CKTA in order to create a hybrid fitness as described in Equation 5

7. if wAccCKTA appears, then the weighted accuracy of logic hypothesis was multiplied by the CKTA in order to create a hybrid fitness, again as described in Equation 5

8. if WMutation appears, then in the case where the baseline algorithm did not include a complete and locally complete refinement operator, it was enhanced to use one

9. if GLPS and a * appears in the name, then GLPS [5] with the AND-OR tree shuffling enhancement was used

10. if Aleph appears in the name, then one generation with no mutation was used and the scoring function was simply the accuracy of the logic program (i.e. hypothesis); note that having the different members of the population created by shuffling the samples will produce different results as described in [26] because Aleph will cover the first sample provided to it and then add new rules as new uncovered samples are provided

The 'C-val' column specifies the C value used for C-SVM (support vector machine) classification. The constant C in this case is a regularization parameter, allowing one to compromise between (a) data points being on the correct side of the hyperplane created by the SVM and (b) allowing 'slack' which permits samples to appear on the wrong side of the hyperplane created by the SVM [20, 19]. Allowing this slack can lead to SVMs which generalize much better. Smaller C values allow more points to appear on the wrong side, while larger C values strongly discourage points from appearing on the wrong side. If a "Logic-NA" appears in the 'C-val' column, then the logic hypothesis was evaluated in Aleph as a logic program and no SVM was created. This is true even when CKTA was used as the fitness function (i.e. score function) because the quality of the logic program coming out of the

CKTA algorithm was also of interest in this research (not just the quality of the feature space induced by the kernel).

For the ensemble results, the 'Ensemble Type' column is encoded as follows: <strategy>_C<numCandidates>_E<numEnsMems>_D<diversityFactor>. <numCandidates> indicates that the top <numCandidates> in terms of score will be the candidates for the ensemble. <numEnsMems> indicates the number of hypotheses to be included in the ensemble. <diversityFactor> is the same as the diversity factor detailed in Section 3.2.1. The <strategy> can be any of the following:

1. NAIVE means that the top <numEnsMems> based on score were used in a max voting scheme

2. NO_PEN indicates that diversity adjusted ensemble member selection was performed as described in Equation 8 using an $\alpha$ value of 0 and a $\nu$ value of <diversityFactor> (i.e. **no pen**alty for the hypothesis' initial score)

3. PEN indicates that diversity adjusted ensemble member selection was performed using an $\alpha$ value of 1 and a $\nu$ value of <diversityFactor> (i.e. there is a **pen**alty for the hypothesis' initial score)

Note that in all cases, the <numEnsMems> of the created ensemble are utilized in a max voting scheme. 10-fold cross validation is also performed for these ensembles unless otherwise specified.

## 4.2 Additional Results Information

In order to validate the theory presented by Cortes et al [9], we plot CKTA (or a CKTA hybrid score) vs classifier accuracy for all members of the final generation of the first fold of the best CKTA-based GA run. Additionally, we show

a kernel PCA using the kernel from the best CKTA-based GA run. The kernel PCA is performed in order to show how the kernel based approaches, such as those presented herein and in Landwehr et al [7, 8], can be used in order to provide interesting visualizations of the logic data embedded in the feature space induced by the kernel. These visualizations can often prompt further investigation. These visualizations are provided solely for the sake of demonstration, as they are not the focus of this study, but rather a useful byproduct of it.

### 4.3 Mutagenesis

The mutagenesis data describes relationships from QUANTA (a molecular modeling package) for 230 compounds of interest [24], and four variables from a former study of these compounds [38]. The data is meant to predict the mutagenicity of nitroaromatic compounds, which can occur in both exhaust fumes from automobiles and "during the synthesis of industrial compounds". Nitroaromatic compounds having a high mutagenicity have been identified as being carcinogenic.

The former study divided the compounds into two groups, a group of 188 compounds (the friendly group) which could have mutagenicity accurately predicted from four regression variables of interest and a group of 42 compounds (the unfriendly group) which were not amenable to regression with these variables. The friendly data set has 10324 facts while the unfriendly data set has 2109 [7]. The four regression variables of interest from the previous study were as follows per [24]:

1. logP : log of compound's octanol/water partition coefficient (hydrophobicity)
2. eLUMO : energy of the compounds lowest unoccupied molecular orbital, obtained from a quantum mechanical molecular model
3. I1 : an 'indicator variable' that is set to 1 for all compounds containing 3 or

more fused rings

4. Ia : an 'indicator variable' that takes the value 1 for "...five examples of acenthrylenes and shows that these are much less active than expected for some unknown reason" [38]

Compounds with log mutagenicity greater than zero are considered active (positive examples) while compounders with negative or zero log mutagenicity are considered inactive (negative examples). The 188 (i.e. friendly) and 42 (i.e. unfriendly) groups have the following samples:

Table 2: Mutagenesis Data Summary

| | **Active** | **Inactive** | **Total** |
|---|---|---|---|
| **Friendly** | 125 | 63 | 188 |
| **Unfriendly** | 13 | 29 | 42 |
| **All** | 138 | 92 | 230 |

During experimentation, 10-fold cross validation (CV) was used for the 188 (friendly) group while leave-one-out (a.k.a. jack-knife or 42-fold) sampling was used for the 42 (unfriendly) group. For 10-fold CV, random assignment of compounds into approximately equally sized sets was performed.

### 4.3.1 Mutagenesis Friendly

For the friendly mutagenesis data, a population size of 40 was used and 30 generations were created by the GA in all runs (apart from 'Aleph' which only utilized one generation). A box plot of all configurations, sorted from left to right by descending mean accuracy and ascending standard deviation, is provided in Figure 10. A table of the top performing models is provided in Table 3. The full table of results can be found in A.1.1.

| Config | C-Val | mean | stddev |
|---|---|---|---|
| CKTA_Gauss1 | 1 | 0.866959 | 0.07243 |
| GLPS* | logic-NA | 0.861988 | 0.086627 |
| Aleph | logic-NA | 0.861696 | 0.061859 |
| CKTA_Gauss1 | 10 | 0.861696 | 0.076249 |
| CKTA_withIncestAvoidance_Gauss1 | 10 | 0.861111 | 0.063782 |
| CKTA_withIncestAvoidance_Gauss1 | 0.1 | 0.860819 | 0.08545 |
| wAccCKTA_Linear | 1 | 0.85614 | 0.083352 |
| CKTA_withIncestAvoidance_Gauss1 | 1 | 0.855848 | 0.071994 |
| wAccCKTA_withIncestAvoidance_Linear | logic-NA | 0.855848 | 0.071994 |
| wAccCKTA_Linear | 10 | 0.855848 | 0.087438 |
| wAccCKTA_withIncestAvoidance_Linear | 10 | 0.850877 | 0.089467 |
| CKTA_withIncestAvoidance_Poly2 | 0.1 | 0.850585 | 0.106309 |
| wAccCKTA_withIncestAvoidance_Linear | 1 | 0.845029 | 0.097476 |
| wAccCKTA_Gauss1 | 0.1 | 0.840351 | 0.078992 |
| wAccCKTA_Gauss1 | logic-NA | 0.840351 | 0.078992 |
| CKTA_Gauss1 | 0.1 | 0.840058 | 0.062043 |
| CKTA_withIncestAvoidance_Poly2 | 1 | 0.840058 | 0.08407 |

Table 3: Top Results for Mutagenesis Friendly

Above we see that the GA guided by centered kernel target alignment using a Gaussian kernel with a $\gamma$ value of 1 performed the best. A C-value of 1.0 was used for the C-SVM classifier created at the end of the GA along with this kernel. In this case, the approach presented herein is competitive with GLPS* and Aleph and, for the identified parameters, outperforms them.

Using the CKTA_Gauss1 GA run (the best CKTA-based run from above), centered kernel target alignment and accuracy of the C-SVM classifier were computed for all members of the final generation on the first fold (i.e. FOLD0 in [37]) on both the training data and the test data. While this is a small sample, we

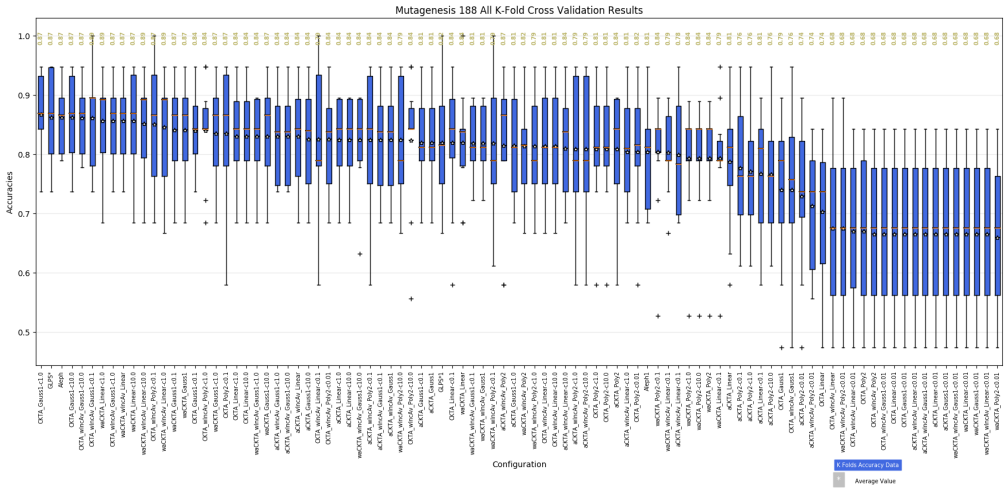


Figure 10: Box Plot for 10-fold CV Results. Stars represent the mean. The gold numbers at the top represent median. From left to right, the configurations are plotted based on descending mean and ascending standard deviation.

would expect a positive correlation between the CKTA and the classifier accuracy for the training and test data. The results are shown in Figure 11 with linear line fits overlaid for both the training and test data results. The positive correlation between CKTA and classifier accuracy, albeit slight, boosts our confidence in the theory proposed by Cortes et al [9].

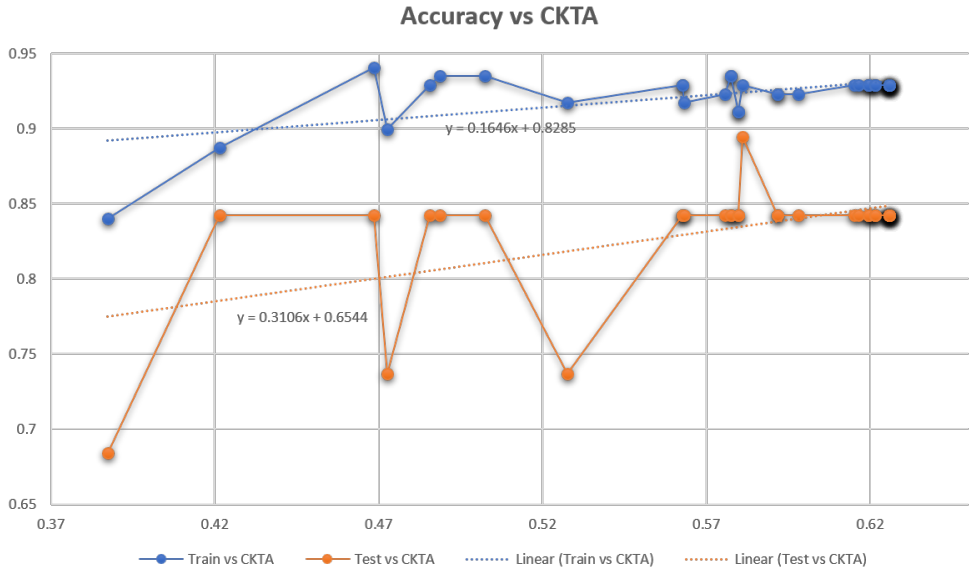


Figure 11: Train and Test Data vs CKTA; Linear fits for each are also shown

An example visualization using Kernel PCA is shown in Figure 12. In this visualization, the markers are sized based on the number of points of the given marker type at the location. Interestingly, 51 points are at the large red circle and 5 are at the next largest red circle. This means that 88.9% (56 of 63) of the Inactive friendly mutagenesis data points are mapped to these points. This kind of visualization can be interesting in order to find areas of confusion in the feature space (i.e. points where both Active and Inactive points occur) and to see how the data is distributed in the feature space. This can provide some intuition about potential clusters in the feature space. If clusters are apparent, kernel K means could be performed in order to discover the centroids of these clusters.

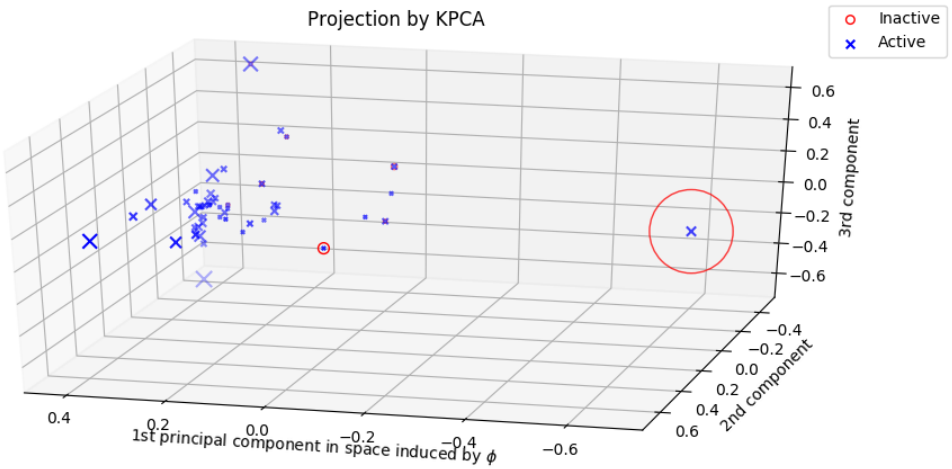


Figure 12: Kernel PCA Using the Gaussian Kernel for the Friendly Mutagenesis Data

Table 4 shows the results using kFOIL and a couple of variants of kFOIL. The 'kFOIL' column provides the results of the unaltered kFOIL algorithm as defined in [7] using KTA. The 'CKTA FOIL' column provides the results of the kFOIL algorithm altered to use CKTA. The 'Centered Data' column provides results for kFOIL using KTA, but with the data centered in feature space (note that this differs from strict CKTA in that the target matrix is not centered). For each

variant, three kernels were used:

1. Gaussian with $\gamma$ equal to 1
2. polynomial with degree 2
3. linear

For each of these kernels, the mean and standard deviation are reported, with the mean appearing above the standard deviation. Note that for this data set and these parameters, the best performer was the original kFOIL algorithm with KTA. However, it should also be noted that none of these results are competitive with the GA results. They are also not competitive with Aleph, a discouraging result for kFOIL.

| | **CKTA Foil** | **Center Data** | **kFOIL** |
|---|---|---|---|
| **Linear** | 0.759942<br>0.109518 | 0.759942<br>0.109518 | **0.776901**<br>0.076584 |
| **Poly 2** | 0.760526<br>0.095775 | 0.760526<br>0.095775 | 0.728655<br>0.114273 |
| **Gauss 1.0** | 0.722807<br>0.145571 | 0.722807<br>0.145571 | 0.760234<br>0.077007 |

Table 4: Experiment Results Using kFOIL and kFOIL Variants

Using the CKTA_Gauss1 score function (since it was the best performing), ensembles were created for each of the folds. The results are sorted by descending mean accuracy and ascending standard deviation and shown in Table 5. A box plot for the same results is shown in Figure 13. While the top result does not outperform the non-ensemble results above, it is worth noting that the top performing ensemble type is one which encourages diversity and does not follow the naive top $m$ classifiers approach.

| Ensemble Type | mean (10-fold CV) | stddev (10-fold CV) |
|---|---|---|
| PEN_C25_E7_D2 | 0.866959 | 0.072430 |
| NO_PEN_C15_E5_D1 | 0.861696 | 0.076249 |
| NO_PEN_C15_E5_D2 | 0.861696 | 0.076249 |
| PEN_C15_E5_D1 | 0.861696 | 0.076249 |
| PEN_C15_E5_D2 | 0.861696 | 0.076249 |
| PEN_C25_E7_D1 | 0.861696 | 0.076249 |
| NAIVE_C15_E5_D1 | 0.861696 | 0.076249 |
| NAIVE_C15_E5_D2 | 0.861696 | 0.076249 |
| NAIVE_C25_E7_D1 | 0.861696 | 0.076249 |
| NAIVE_C25_E7_D2 | 0.861696 | 0.076249 |
| NAIVE_C25_E7_D3 | 0.861696 | 0.076249 |
| PEN_C25_E7_D3 | 0.861404 | 0.067810 |
| NO_PEN_C25_E7_D1 | 0.856140 | 0.071421 |
| NO_PEN_C25_E7_D2 | 0.856140 | 0.071421 |
| NO_PEN_C25_E7_D3 | 0.856140 | 0.071421 |

Table 5: Mutagenesis Friendly Ensemble Results Using Gauss 1 Kernel

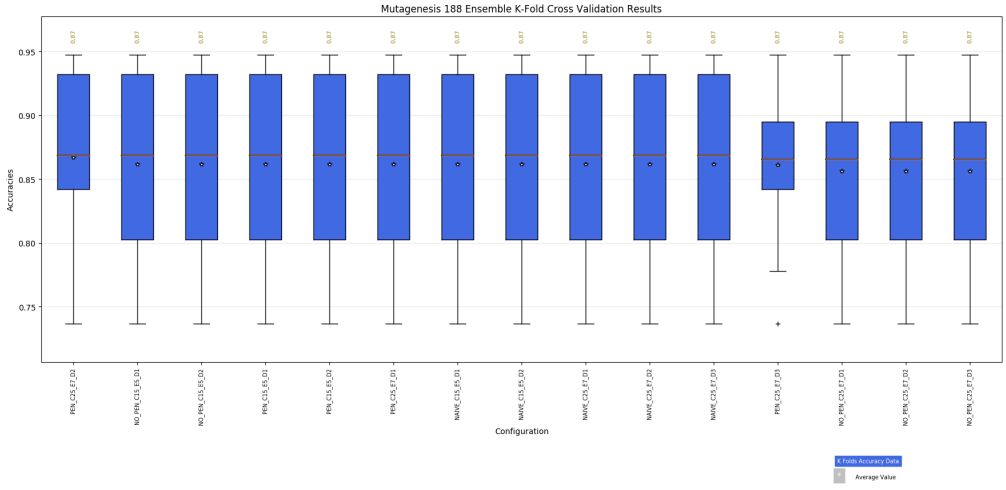


Figure 13: Box Plot for 10-fold CV Results. Stars represent the mean. The gold numbers at the top represent median. From left to right, the configurations are plotted based on descending mean and ascending standard deviation.

### 4.3.2 Mutagenesis Unfriendly

For the friendly mutagenesis data, a population size of 20 was used and 20 generations were created by the GA in all runs. A box plot of all configurations, sorted from left to right by descending mean accuracy and ascending standard deviation, is provided in Figure 14. Note that because leave-one-out cross validation was used, the resulting accuracy for each fold is either 0 or 1 (i.e. 0% or 100%). Hence, the most interesting data points in the box plot are the *mean* values, which are represented by stars. The large blue bars for the configurations to the right simply indicate that the CV results for these configurations had more 0 values. A table of the top performing models is provided in Table 6. The full table of results can be found in A.1.2.

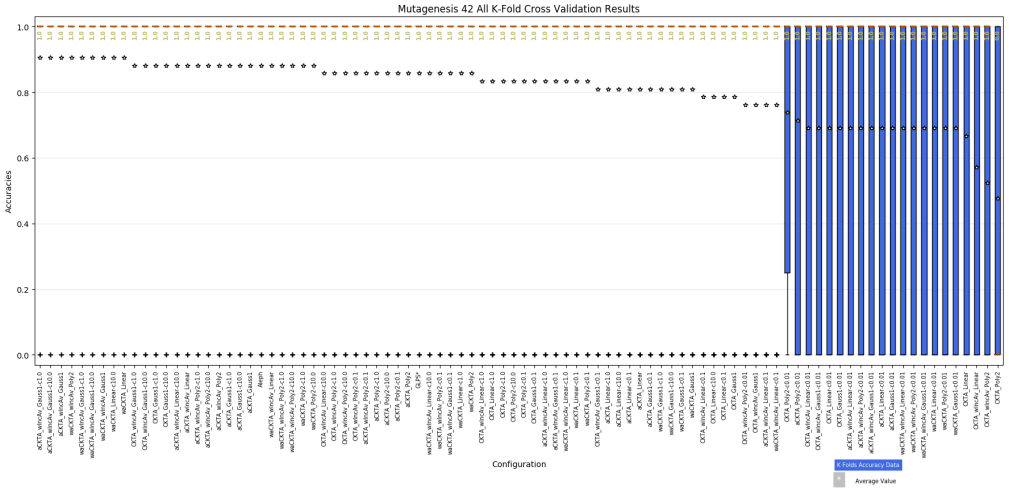


Figure 14: Box Plot for leave-one-out CV Results. Stars represent the mean. The gold numbers at the top represent median. From left to right, the configurations are plotted based on descending mean and ascending standard deviation.

| Config | C-Val | mean | stddev |
|---|---|---|---|
| AccCKTA_withIncestAvoidance_Gauss1 | 1 | 0.904762 | 0.297102 |
| AccCKTA_withIncestAvoidance_Gauss1 | 10 | 0.904762 | 0.297102 |
| AccCKTA_withIncestAvoidance_Gauss1 | logic-NA | 0.904762 | 0.297102 |
| wAccCKTA_withIncestAvoidance_Poly2 | logic-NA | 0.904762 | 0.297102 |

| wAccCKTA_withIncestAvoidance_Gauss1 | 1 | 0.904762 | 0.297102 |
|---|---|---|---|
| wAccCKTA_withIncestAvoidance_Gauss1 | 10 | 0.904762 | 0.297102 |
| wAccCKTA_withIncestAvoidance_Gauss1 | logic-NA | 0.904762 | 0.297102 |
| wAccCKTA_Linear | 10 | 0.904762 | 0.297102 |
| wAccCKTA_Linear | logic-NA | 0.904762 | 0.297102 |
| CKTA_withIncestAvoidance_Gauss1 | 1 | 0.880952 | 0.32777 |
| CKTA_withIncestAvoidance_Gauss1 | 10 | 0.880952 | 0.32777 |
| CKTA_Gauss1 | 1 | 0.880952 | 0.32777 |
| CKTA_Gauss1 | 10 | 0.880952 | 0.32777 |
| AccCKTA_withIncestAvoidance_Linear | 10 | 0.880952 | 0.32777 |
| AccCKTA_withIncestAvoidance_Linear | logic-NA | 0.880952 | 0.32777 |
| AccCKTA_withIncestAvoidance_Poly2 | 1 | 0.880952 | 0.32777 |
| AccCKTA_withIncestAvoidance_Poly2 | 10 | 0.880952 | 0.32777 |
| AccCKTA_withIncestAvoidance_Poly2 | logic-NA | 0.880952 | 0.32777 |
| AccCKTA_Gauss1 | 1 | 0.880952 | 0.32777 |
| AccCKTA_Gauss1 | 10 | 0.880952 | 0.32777 |
| AccCKTA_Gauss1 | logic-NA | 0.880952 | 0.32777 |
| Aleph | logic-NA | 0.880952 | 0.32777 |
| wAccCKTA_withIncestAvoidance_Linear | logic-NA | 0.880952 | 0.32777 |
| wAccCKTA_withIncestAvoidance_Poly2 | 1 | 0.880952 | 0.32777 |
| wAccCKTA_withIncestAvoidance_Poly2 | 10 | 0.880952 | 0.32777 |
| wAccCKTA_Poly2 | 1 | 0.880952 | 0.32777 |
| wAccCKTA_Poly2 | 10 | 0.880952 | 0.32777 |

Table 6: Top Results for Mutagenesis Unfriendly Data

Nine of the configurations shared the best mean accuracy. Out of these nine, seven of them utilized incest avoidance during crossover. This implies that the incest avoidance measure is a useful hyperparameter for the GA. For the remainder of this section, we will focus on the first entry in the table. This entry utilized GA with the fitness score being the centered kernel target alignment (using a Gaussian

kernel with a $\gamma$ value of 1) times the accuracy of the logic program. The run also utilized incest avoidance during crossover. A C-value of 1.0 was used for the C-SVM classifier created at the end of the GA along with this kernel. In this case, the approach presented herein outperformed both GLPS* and Aleph.

Using the AccCKTA_withIncestAvoidance_Gauss1 GA run (the best CKTA-based run from above), the score (AccCKTA - centered kernel target alignment times the accuracy of the logic program) and accuracy of the C-SVM classifier created using the learned kernel were computed for all members of the final generation on the first fold (i.e. FOLD0 in [37]) on both the training data and the test data. The results are shown in Figure 15 with linear line fits overlaid for the training data results. No linear fit was added for the test data since it quickly converged to one. The positive correlation between CKTA and classifier accuracy again boosts our confidence in the theory proposed by Cortes et al [9]. It also justifies the usage of hybrid scores as described in Equation 5. These hybrid scores utilize both the accuracy of learned logic program (i.e. hypothesis) and the centered kernel target alignment of the kernel induced by this hypothesis, thereby balancing between accuracy as a standalone logic program and alignment with the target in the feature space.

The first 3 principal components of a Kernel PCA are shown in Figure 16 using the Gaussian kernel with a $\gamma$ value of 1, as this kernel produced the best results in the experimentation detailed above. In this visualization, the markers are sized based on the number of points of the given marker type at the location. 28 Inactive points appear at the large red circle, out of 29 total Inactive points (i.e. 96.6% of the Inactive points are mapped to this location). 7 Active points appear at the large blue x, out of 13 total Active points (i.e. 53.8% of the Active points). 32 out of 42 total points, or 83.3% of all points are mapped to one of these two

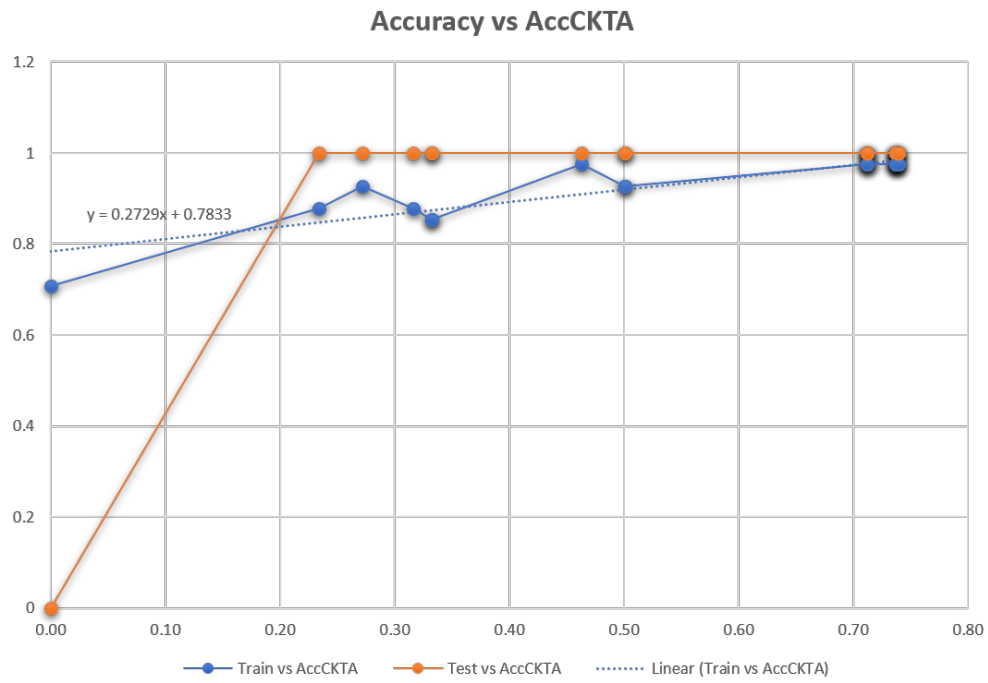


Figure 15: Train and Test Data vs AccCKTA; Linear fits for each are also shown

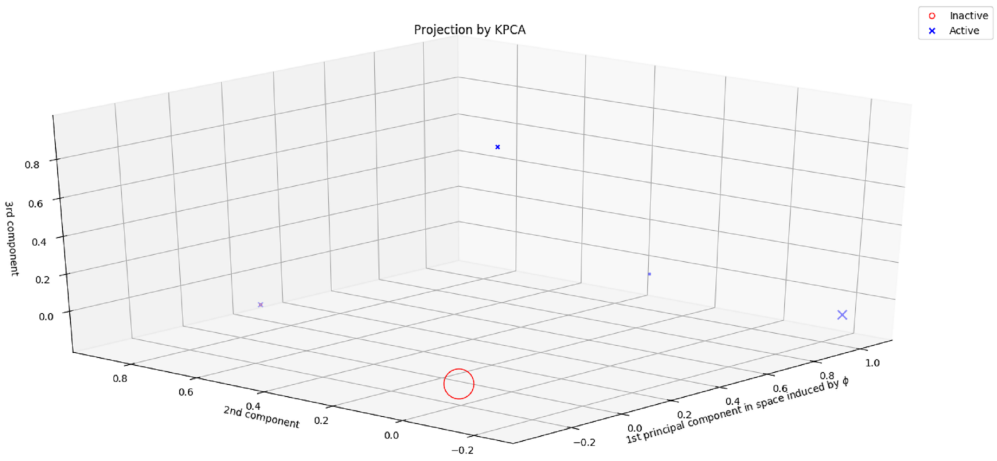


Figure 16: Kernel PCA Using the Gaussian Kernel for the Unfriendly Mutagenesis Data

locations.

In Figure 16, we can see one area where there appears to confusion in the feature space (i.e. points where both Active and Inactive points occur). One Inactive point and three Active points were mapped to this location. This area

is zoomed in on in Figure 17. It is likely that the samples mapped to this area, and likely the other small blue x's, caused confusion to the models as the top performing models had a mean accuracy of 90.4762%, meaning that on average, 4 of the 42 points were misclassified by these top performers across the K folds (note that $38/42 = 0.904762$). We could easily add the labels of the samples to the points in order to identify these trouble points so that they could be further investigated. This will not be performed in this study, but is noted here to show how kernel PCA, using the kernels learned by the GA, can be utilized as an analysis tool for ILP. Note that the kernels are learned as the hypothesis $H$, required for the $\phi_{H,B}$ mapping (see 4), is learned by the GA in such a way that it maximizes the scoring function.

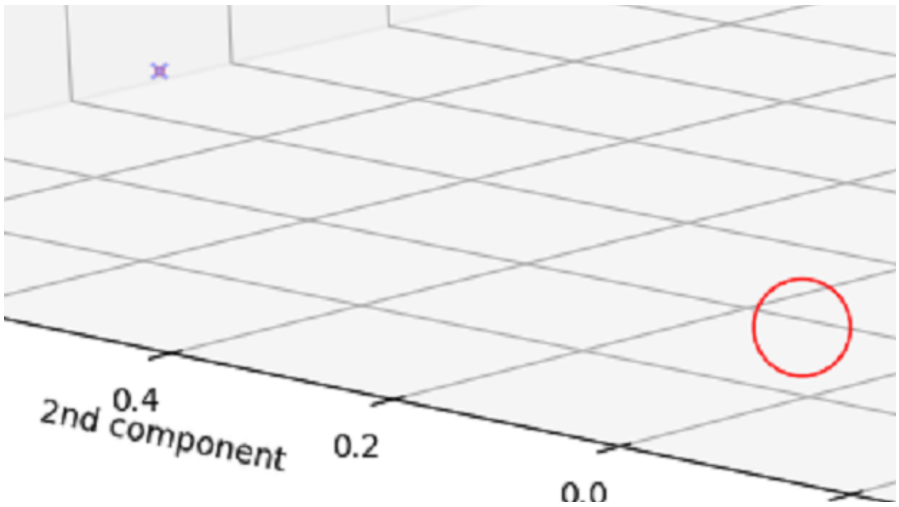


Figure 17: Closeup of Kernel PCA for the Unfriendly Mutagenesis Data Showing Confusion

The following table show the results using kFOIL, along with the 'CKTA FOIL' and 'Centered Data' kFOIL variants of kFOIL previously defined. For each variant, three kernels were used:

1. Gaussian with $\gamma$ equal to 1
2. polynomial with degree 2

3. linear

For each of these kernels, the mean and standard deviation are reported, with the mean appearing above the standard deviation. Note that for this data set and these parameters, the best performers all included centering the data (i.e. were either using 'CKTA Foil' or 'Centered Data' kFOIL). However, it should also be noted that these results do not match the top GA results. However, they are competitive with Aleph in this case and outperform GLPS*.

| | **CKTA Foil** | **Center Data** | **kFOIL** |
|---|---|---|---|
| **Linear** | 0.857143<br>0.354169 | **0.880952**<br>0.327770 | 0.833333<br>0.3771955 |
| **Poly 2** | **0.880952**<br>0.327770 | **0.880952**<br>0.327770 | 0.833333<br>0.3771955 |
| **Gauss 1.0** | **0.880952**<br>0.327770 | **0.880952**<br>0.327770 | 0.833333<br>0.3771955 |

Table 7: Experiment Results Using kFOIL and kFOIL Variants

Using the AccCKTA_Gauss1 score function (since it was the best performing), ensembles were created for each of the folds. The results are sorted by descending mean accuracy and ascending standard deviation and shown in Table 8. A box plot for the same results is shown in Figure 18. While the top result does not outperform the non-ensemble results above, it is worth noting that ensembles using diversity are again among the top performing ensemble types, again implying that the diverse ensembles show promise.

| **Ensemble Type** | **mean (10-fold CV)** | **stddev (10-fold CV)** |
|---|---|---|
| NO_PEN_C15_E5_D1 | 0.904762 | 0.297102 |
| PEN_C15_E5_D1 | 0.904762 | 0.297102 |
| PEN_C15_E5_D2 | 0.904762 | 0.297102 |
| PEN_C25_E7_D1 | 0.904762 | 0.297102 |

| | | |
|---|---|---|
| NAIVE_C15_E5_D1 | 0.904762 | 0.297102 |
| NAIVE_C15_E5_D2 | 0.904762 | 0.297102 |
| NAIVE_C25_E7_D1 | 0.904762 | 0.297102 |
| NAIVE_C25_E7_D2 | 0.904762 | 0.297102 |
| NAIVE_C25_E7_D3 | 0.904762 | 0.297102 |
| NO_PEN_C15_E5_D2 | 0.880952 | 0.327770 |
| NO_PEN_C25_E7_D1 | 0.857143 | 0.354169 |
| NO_PEN_C25_E7_D2 | 0.857143 | 0.354169 |
| NO_PEN_C25_E7_D3 | 0.857143 | 0.354169 |
| PEN_C25_E7_D2 | 0.857143 | 0.354169 |
| PEN_C25_E7_D3 | 0.857143 | 0.354169 |

Table 8: Mutagenesis 42 Ensemble Results Using Gauss 1 Kernel

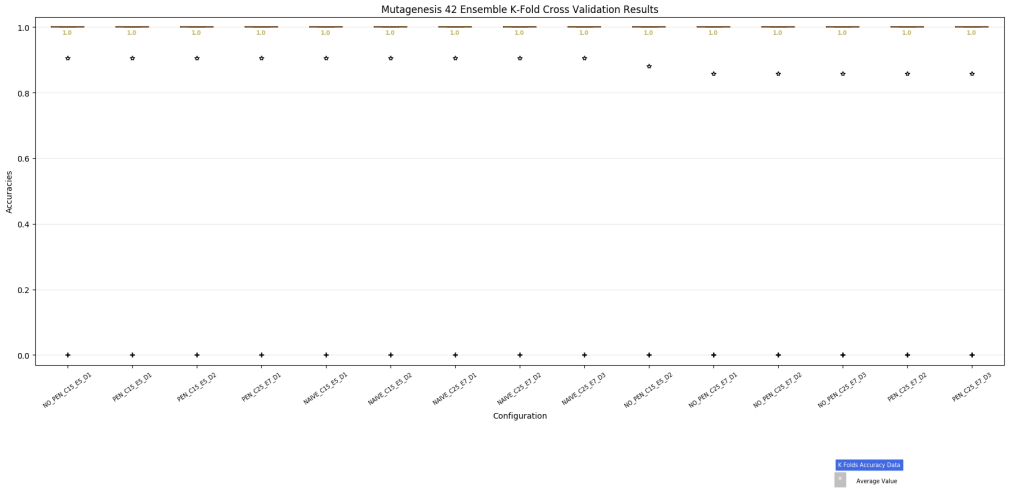


Figure 18: Box Plot for 10-fold CV Results. Stars represent the mean. The gold numbers at the top represent median. From left to right, the configurations are plotted based on descending mean and ascending standard deviation.

### 4.4 Alzheimer's

The Alzheimer's data consists of logical comparisons (relations) between pairs (c1, c2) of analogues of Tacrine, an Alzheimer's drug, in order to determine if compound c1 has more of a particular property than compound c2 (the predicate

returns true if c1 $>$ c2 and false otherwise). Two of the properties were observed as part of this study, namely low toxicity and inhibit amine reuptake [39, 40]. The logical comparisons are transitive and anti-symmetric (i.e. if $c1 > c2$, then $c2 \not> c1$ - or more formally, if $R(c1, c2)$ holds, with $c1 \neq c2$, then $R(c2, c1)$ does not hold). For some pairs of compounds, the result of the comparison could not be determined and hence the relation is not complete [7].

The low toxicity data contains 886 examples and the amine reuptake data contains 686 examples. Both contain 3,754 facts [7].

Note that I was unable to get kFOIL to run on the Alzheimer's data set. Furthermore, the results reported in the kFOIL paper for Aleph seem suspect (accuracy is too high for all methods compared to the experiments that I have run). This could be caused by the usage of different background information between the studies or by the data being treated differently between the studies. In this study each sample was treated independently, and the folds drawn as such. Perhaps in the kFOIL study, the samples were considered in pairs (i.e. R(c1, c2) and R(c2, c1) were forced to be in the same training set). The difference is unclear. Hence, no comparison to kFOIL or its variants were performed for the Alzheimer's data. Also note that, in the interest of time, incest avoidance was not attempted for the Alzheimer's data sets since they are larger and computing the CKTA for large data sets can be time consuming. For incest avoidance, the computation needs to be performed between the first selected parent and all other members of the population during crossover, which occurs during the creation of each successive generation. This computational burden can, to some extent, be reduced by caching (i.e. if the CKTA between a pair of hypotheses has already been computed, reuse it in the future); however, it is still a bit slow. Future work could include speeding up these computations in other fashions (i.e. utilizing more sophisticated caching

schemes, etc.).

### 4.4.1 Inhibit Amine Reuptake

For the Alzheimer's inhibit amine reuptake data, a population size of 30 was used and 30 generations were created by the GA in all runs. A box plot of all configurations, sorted from left to right by descending mean accuracy and ascending standard deviation, is provided in Figure 19. A table of the top performing models is provided in Table 9. The full table of results can be found in A.1.3.

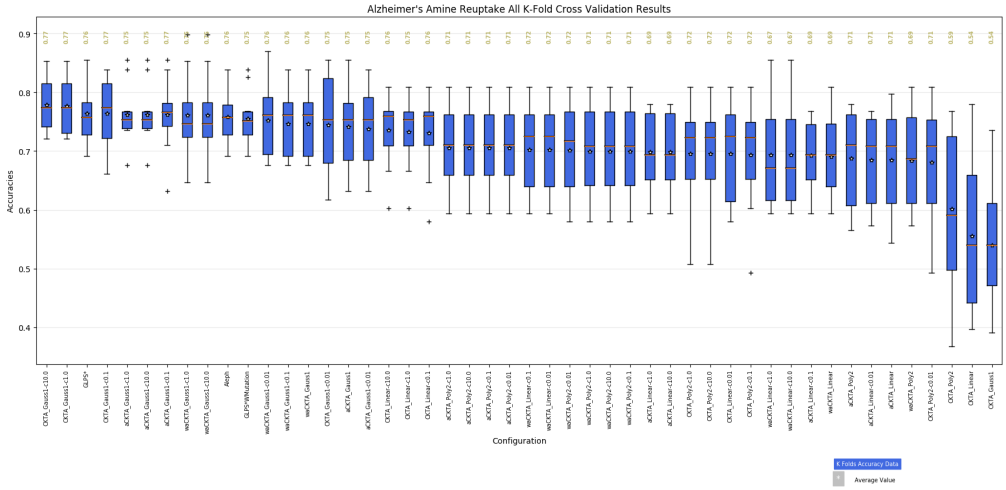


Figure 19: Box Plot for 10-fold CV Results. Stars represent the mean. The gold numbers at the top represent median. From left to right, the configurations are plotted based on descending mean and ascending standard deviation.

| Config | C-Val | mean | stddev |
|---|---|---|---|
| CKTA_Gauss1 | 10 | 0.778389 | 0.047727 |
| CKTA_Gauss1 | 1 | 0.776939 | 0.049248 |
| CKTA_Gauss1 | 0.1 | 0.765217 | 0.055672 |
| GLPS* | logic-NA | 0.763853 | 0.056995 |
| AccCKTA_Gauss1 | 1 | 0.762340 | 0.051351 |
| AccCKTA_Gauss1 | 10 | 0.762340 | 0.051351 |
| AccCKTA_Gauss1 | 0.1 | 0.762340 | 0.062660 |
| wAccCKTA_Gauss1 | 1 | 0.760806 | 0.073323 |

| wAccCKTA_Gauss1 | 10 | 0.760806 | 0.073323 |
|---|---|---|---|
| Aleph | logic-NA | 0.758035 | 0.049117 |
| GLPSWMutation* | logic-NA | 0.755136 | 0.048355 |
| wAccCKTA_Gauss1 | 0.01 | 0.751982 | 0.063437 |
| wAccCKTA_Gauss1 | 0.1 | 0.746228 | 0.057962 |
| wAccCKTA_Gauss1 | logic-NA | 0.746228 | 0.057962 |
| CKTA_Gauss1 | 0.01 | 0.744629 | 0.089664 |
| AccCKTA_Gauss1 | logic-NA | 0.741880 | 0.076501 |

Table 9: Top Results for the Inhibit Amine Reuptake Data

Above we see that the GA guided by centered kernel target alignment using a Gaussian kernel with a $\gamma$ value of 1 performed the best. A C-value of 10.0 was used for the C-SVM classifier created at the end of the GA along with this kernel. In this case, the approach presented herein is competitive with GLPS* and Aleph and, for the identified parameters, outperforms them. It also outperformed GLPSWMutation*.

Using the CKTA_Gauss1 GA run (the best CKTA-based run from above), centered kernel target alignment and accuracy of the C-SVM classifier were computed for all members of the final generation of the first fold (i.e. FOLD0 in [37]) on both the training data and the test data. While this is a small sample, we would expect a positive correlation between the CKTA and the classifier accuracy for the training and test data. The results are shown in Figure 20 with linear line fits overlaid for both the training and the test data results. While there is strange behavior towards the left of the plot, where high accuracies are associated with smaller CKTA values, there is still a positive correlation between CKTA and classifier accuracy overall, albeit slight, again validating the theory proposed by Cortes et al [9]. It should also be noted that the highest CKTA value achieved was around 0.28, which is quite low. A larger spread of CKTA values vs accuracies

may show a more interesting correlation. However, different hyperparameters may be necessary to achieve such a spread, as the CKTA seemed to converge around 0.28 with these hyperparameters for this data set.

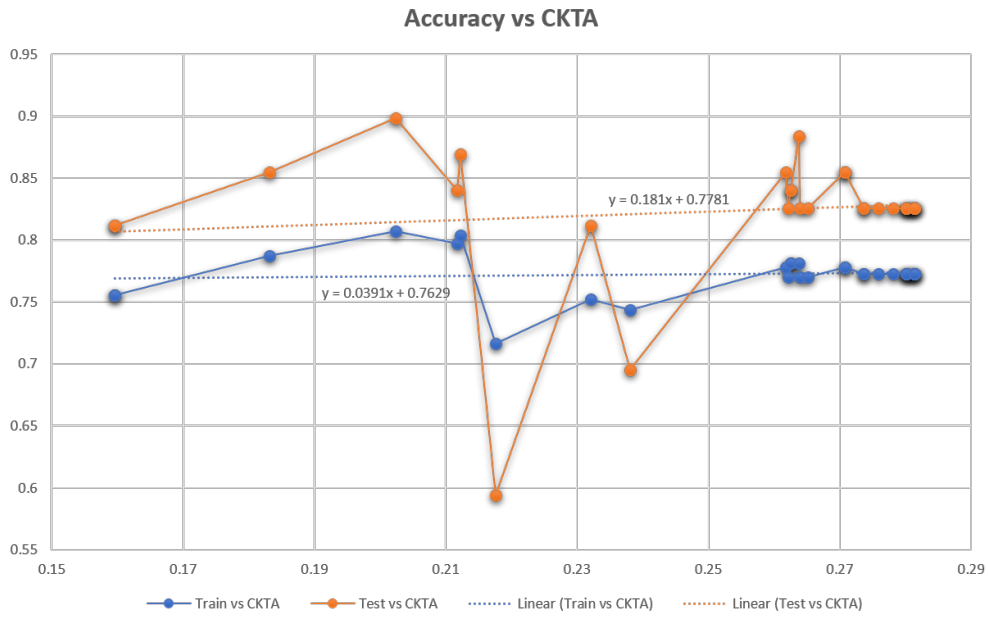


Figure 20: Train and Test Data vs CKTA; Linear fits for each are also shown

The first 3 principal components of a Kernel PCA are shown in Figure 21 using the Gaussian kernel with a $\gamma$ value of 1, as this kernel produced the best results in the experimentation detailed above. In this visualization, the markers are again sized based on the number of points of the given marker type at the location. Note that because the relation is anti-symmetric, of the 686 samples, 343 are positive while 343 are negative.

209 of the "< Inhibit Amine Reuptake" points appear at the largest red circle, out of 343 total (i.e. 60.9%). There is also quite a bit of overlap between the "< Inhibit Amine Reuptake" points and the ">= Inhibit Amine Reuptake" points in feature space. This is not surprising as half of these points are logical inversions of the other half. There also appears to be a few clusters in the data (between 4 and 6). It would be interesting to perform a kernel k-means clustering on this data and

to analyze the resulting clusters to see what makes the compounds within each cluster similar to one another. This will not be performed in this study, but is noted here to show how kernel PCA, using the kernels learned by the GA, can be utilized as an analysis tool for ILP, as a means to visualize the predicate data is provided.

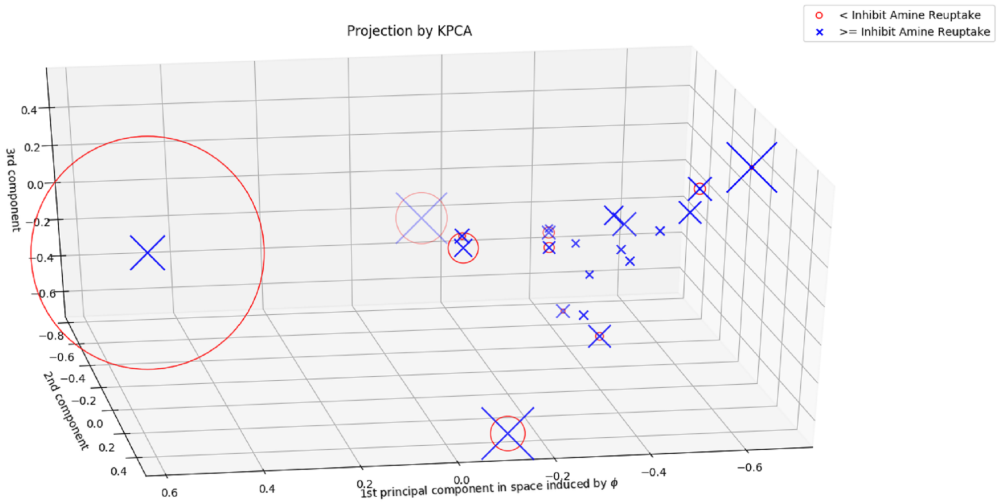


Figure 21: Kernel PCA Using the Gaussian Kernel for the Alzheimer's Inhibit Amine Reuptake Data

Using the CKTA_Gauss1 score function (since it was the best performing), ensembles were created for each of the folds. The results are sorted by descending mean accuracy and ascending standard deviation and shown in Table 10. A box plot for the same results is shown in Figure 22. The top result outperforms the non-ensemble results above. Additionally, it is worth noting that the top performing ensemble type is one which encourages diversity and does not follow the naive top $m$ classifiers approach, implying that the diverse ensembles again show promise. That the top ensemble result outperforms the best non-ensemble result is also encouraging since, because the ensembles were created from the last generation of the CKTA_Gauss1 run, the members of the ensemble were, at best equal to the non-ensemble member. This is a demonstration of the efficacy of ensembles

in general, and, as a diverse ensemble has the best results, of the potential of the diverse ensemble creation methodology proposed in this work.

| Ensemble Type | mean (10-fold CV) | stddev (10-fold CV) |
|---|---|---|
| PEN_C15_E5_D1 | 0.782736573 | 0.05367584 |
| PEN_C25_E7_D1 | 0.782736573 | 0.05367584 |
| PEN_C15_E5_D2 | 0.781287298 | 0.055159209 |
| NAIVE_C15_E5_D1 | 0.779838022 | 0.049991801 |
| NAIVE_C15_E5_D2 | 0.779838022 | 0.049991801 |
| PEN_C25_E7_D2 | 0.779816709 | 0.062469986 |
| NO_PEN_C15_E5_D1 | 0.778388747 | 0.051490654 |
| NO_PEN_C15_E5_D2 | 0.778367434 | 0.051169486 |
| NAIVE_C25_E7_D1 | 0.776896846 | 0.04591173 |
| NAIVE_C25_E7_D2 | 0.776896846 | 0.04591173 |
| NAIVE_C25_E7_D3 | 0.776896846 | 0.04591173 |
| PEN_C25_E7_D3 | 0.771099744 | 0.061464559 |
| NO_PEN_C25_E7_D3 | 0.765196078 | 0.063814177 |
| NO_PEN_C25_E7_D2 | 0.76372549 | 0.064370302 |
| NO_PEN_C25_E7_D1 | 0.76372549 | 0.069052846 |

Table 10: Inhibit Amine Reuptake Ensemble Results Using Gauss 1 Kernel

These results would likely be improved if ensembles were created based on the final populations from multiple GA runs (so that different kernel types, etc. are used in the creation of the ensemble). Furthermore, alternative approaches to max voting could be explored (i.e. using weighting based on something similar to $\gamma$ as defined in Equation 8). Both of these topics would make for very interesting future work.

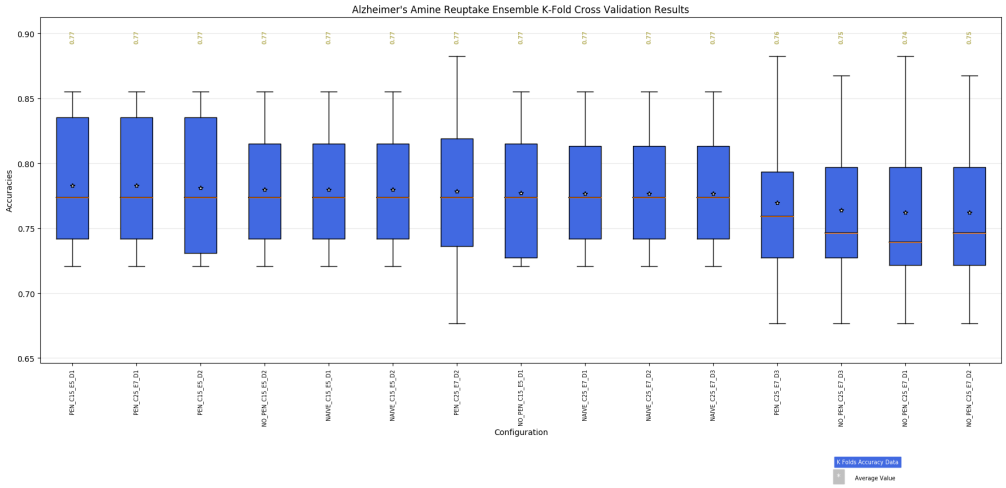


Figure 22: Box Plot for 10-fold CV Results. Stars represent the mean. The gold numbers at the top represent median. From left to right, the configurations are plotted based on descending mean and ascending standard deviation.

### 4.4.2 Toxicity

For the Alzheimer's toxicity data, a population size of 30 was used and 30 generations were created by the GA in all runs. A box plot of all configurations, sorted from left to right by descending mean accuracy and ascending standard deviation, is provided in Figure 23. A table of the top performing models is provided in Table 11. The full table of results can be found in A.1.4.

| Config | C-Val | mean | stddev |
|---|---|---|---|
| Aleph | logic-NA | 0.795748 | 0.040715 |
| AccCKTA_Gauss1 | 1 | 0.795748 | 0.041059 |
| AccCKTA_Gauss1 | 10 | 0.795748 | 0.041059 |
| AccCKTA_Gauss1 | 0.1 | 0.794625 | 0.040464 |
| GLPS* | logic-NA | 0.794625 | 0.040464 |
| CKTA_Gauss1 | 1 | 0.793488 | 0.040613 |
| CKTA_Gauss1 | 10 | 0.793488 | 0.040613 |
| CKTA_Gauss1 | 0.1 | 0.793488 | 0.044565 |
| GLPSWMutation* | logic-NA | 0.792377 | 0.040892 |
| AccCKTA_Gauss1 | 0.01 | 0.792377 | 0.04666 |

| AccCKTA_Gauss1 | logic-NA | 0.792377 | 0.04666 |
|---|---|---|---|
| wAccCKTA_Gauss1 | 1 | 0.785636 | 0.047058 |
| wAccCKTA_Gauss1 | 10 | 0.785636 | 0.047058 |
| wAccCKTA_Gauss1 | 0.1 | 0.783376 | 0.047913 |
| wAccCKTA_Gauss1 | 0.01 | 0.780005 | 0.045267 |
| wAccCKTA_Gauss1 | logic-NA | 0.780005 | 0.045267 |

Table 11: Top Results for Alzheimer's Toxicity Data

Above we see that the best among the CKTA guided GA runs entry utilized a fitness score of the centered kernel target alignment (using a Gaussian kernel with a $\gamma$ value of 1) times the accuracy of the logic program. A C-value of 1.0 was used for the C-SVM classifier created at the end of the GA along with this kernel (although the C-value of 10.0 performed equally as well). In this case, the approach presented herein was competitive with GLPS* and Aleph. However, Aleph was the best performing as it's standard deviation for the 10-fold CV was lower.

Using the AccCKTA_Gauss1 GA run (the best CKTA-based run from above), the score (AccCKTA - centered kernel target alignment times the accuracy of the logic program) and accuracy of the C-SVM classifier created using the learned kernel were computed for all members of the final generation on the first fold (i.e. FOLD0 in [37]) on both the training data and the test data. We again expect a positive correlation between the AccCKTA and the classifier accuracy for the training and the test data. The results are shown in Figure 24 with linear line fits overlaid for both the training and the test data results. The positive correlation between CKTA and classifier accuracy boosts our confidence in the theory proposed by Cortes et al [9]. It also justifies the usage of hybrid scores as defined in Equation 5, utilizing both the accuracy of the learned logic program (i.e. hypothesis) and the centered kernel target alignment of the kernel induced by this hypothesis. These

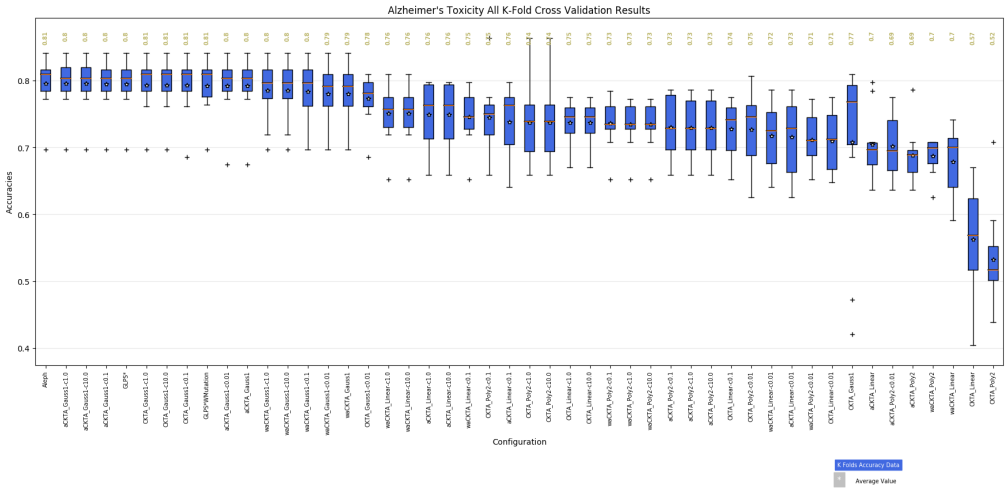


Figure 23: Box Plot for 10-fold CV Results. Stars represent the mean. The gold numbers at the top represent median. From left to right, the configurations are plotted based on descending mean and ascending standard deviation.

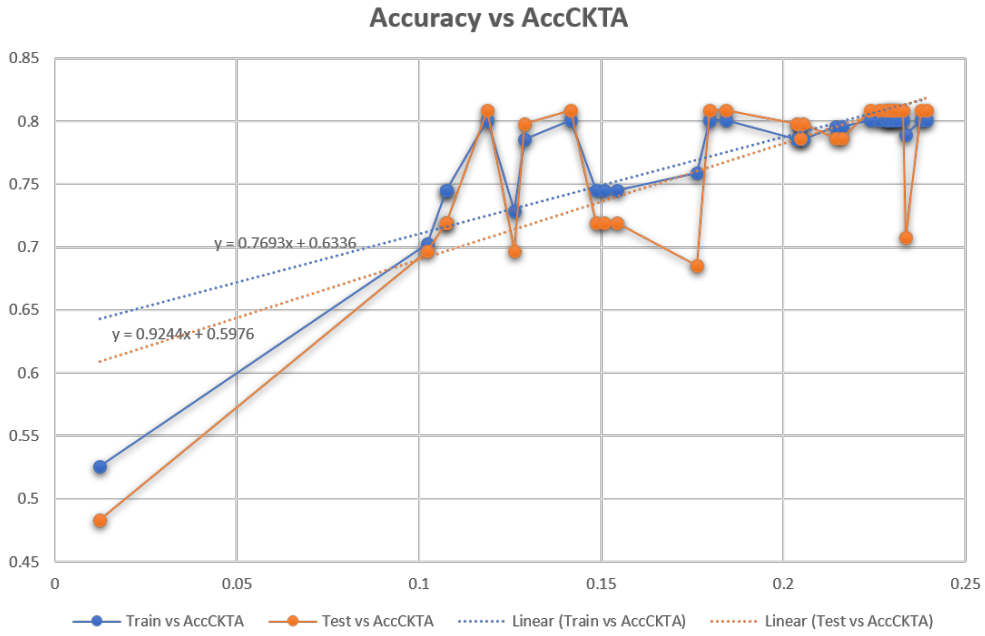


Figure 24: Train and Test Data vs AccCKTA; Linear fits for each are also shown

scores balance between accuracy as a standalone logic program and alignment with the target in the feature space induced by the kernel.

The first 3 principal components of a Kernel PCA are shown in Figure 25 using the Gaussian kernel with a $\gamma$ value of 1, as this kernel produced the best results

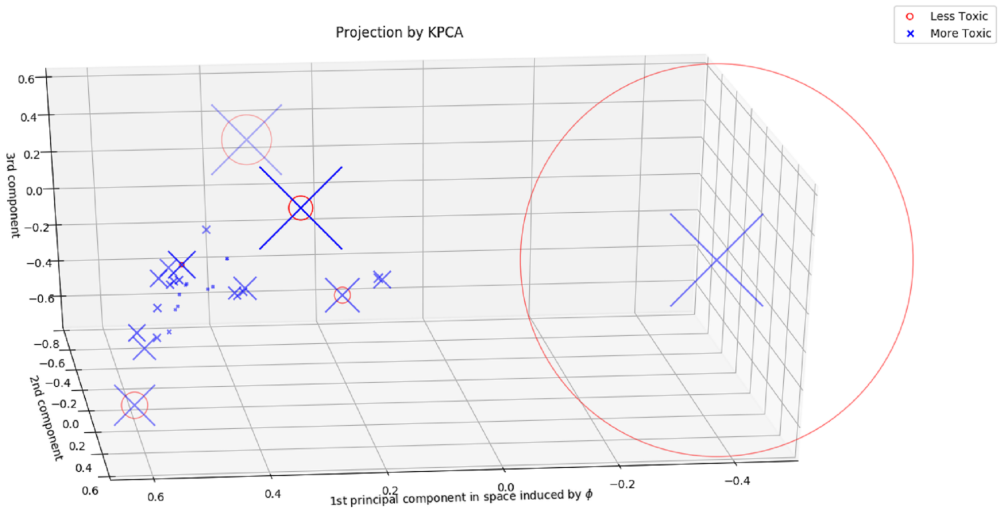


Figure 25: Kernel PCA Using the Gaussian Kernel for the Alzheimer's Toxicity Data

in the experimentation detailed above (among the CKTA-based GA runs). In this visualization, the markers are sized based on the number of points of the given marker type at the location. Note that because the relation is anti-symmetric, of the 886 samples, 443 are positive while 443 are negative.

347 of the "Less Toxic" points appear at the largest red circle, out of 443 total (i.e. 78.3%). There is also quite a bit of overlap between the "Less Toxic" points and the "More Toxic" points in feature space. Again, this is not surprising as half of these points are logical inversions of the other half. There also appears to be a few clusters in the data. A kernel k-means clustering could be performed on this data and the resulting clusters analyzed see what makes the compounds within each cluster similar to one another. Again, this will not be performed as part of this study, but is noted here to show how kernel PCA, using the kernels learned by the GA, can be utilized as an analysis tool for ILP, motivating further investigation.

Using the AccCKTA_Gauss1 score function (since it was the best performing), ensembles were created for each of the folds. The results are sorted by descending mean accuracy and ascending standard deviation and shown in Table 12. A

box plot for the same results is shown in Figure 22. The top result marginally outperforms the non-ensemble results above, including the results for Aleph. Additionally, it is worth noting that the top performing ensemble type is one which encourages diversity and does not follow the naive top $m$ classifiers approach, implying that the diverse ensembles again show promise. That the top ensemble result outperforms the best non-ensemble result is also encouraging since, because the ensembles were created from the last generation of the AccCKTA_Gauss1 run, the members of the ensemble were at best equal to the non-ensemble member. This, again, is a demonstration of the efficacy of ensembles in general, and, as a diverse ensemble has the best results, of the potential of the diverse ensemble creation methodology proposed in this work.

As with the inhibit amine reuptake data, these results would likely be improved if ensembles were created based on the final populations from multiple GA runs (so that different kernel types, etc. are used in the creation of the ensemble). Furthermore, alternative approaches to max voting could be explored (i.e. using weighting based on something similar to $\gamma$ as defined in Equation 8).

| **Ensemble Type** | **mean (10-fold CV)** | **stddev (10-fold CV)** |
|---|---|---|
| NO_PEN_C15_E5_D1 | 0.796871808 | 0.040931389 |
| NO_PEN_C15_E5_D2 | 0.796871808 | 0.040931389 |
| PEN_C15_E5_D2 | 0.796871808 | 0.040931389 |
| PEN_C15_E5_D1 | 0.795748212 | 0.041058538 |
| PEN_C25_E7_D1 | 0.795748212 | 0.041058538 |
| PEN_C25_E7_D2 | 0.795748212 | 0.041058538 |
| NAIVE_C15_E5_D1 | 0.795748212 | 0.041058538 |
| NAIVE_C15_E5_D2 | 0.795748212 | 0.041058538 |
| NAIVE_C25_E7_D1 | 0.795748212 | 0.041058538 |
| NAIVE_C25_E7_D2 | 0.795748212 | 0.041058538 |
| NAIVE_C25_E7_D3 | 0.795748212 | 0.041058538 |

| NO_PEN_C25_E7_D1 | 0.790130235 | 0.045393375 |
|---|---|---|
| NO_PEN_C25_E7_D2 | 0.790130235 | 0.045393375 |
| NO_PEN_C25_E7_D3 | 0.790130235 | 0.045393375 |
| PEN_C25_E7_D3 | 0.789006639 | 0.046544278 |

Table 12: Toxicity Ensemble Results Using Gauss 1 Kernel

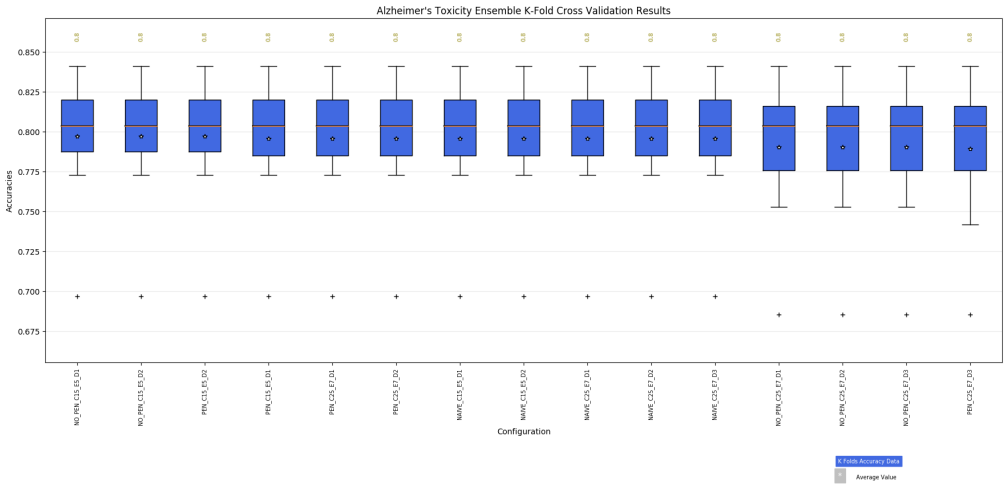


Figure 26: Box Plot for 10-fold CV Results. Stars represent the mean. The gold numbers at the top represent median. From left to right, the configurations are plotted based on descending mean and ascending standard deviation.

## 4.5 Experiment Summary

In order to assist in the distillation of the more comprehensive results above, a summary table for each of the data sets is provided in this section, along with some observations. These tables imply that the proposed applications of CKTA in the ILP domain, both to GA and to ensemble methods, have promise. The tables include the following information:

1. The best CKTA GA result (i.e. best result from the algorithm proposed in this study), ensemble or otherwise, is shown. If an ensemble was the best performing result, an additional 'Ensemble Type' column is included. Recall that the ensembles are created from the last generation of the respective GA

run. If there was a tie in the results from different configurations, the first one appearing in the comprehensive results above was included.

2. The best performing, unaltered kFOIL algorithm is included, if applicable (note that I was unable to get kFOIL to run on the Alzheimer's data sets). Additionally, note that the 'Centered Data' and 'CKTA Foil' variants were improvements investigated during this study and were not part of the original, unaltered kFOIL. However, it is worth noting that even the best of these altered kFOIL algorithms did not outperform the best CKTA GA results. If there was a tie in the results from different configurations, the first one appearing in the comprehensive results above was included.

3. The Aleph result.

4. The GLPS* result.

For each algorithm appearing in the tables, the mean and standard deviation of the 10-fold cross validation is also provided.

### 4.5.1 Mutagenesis Friendly

In Table 13 we see that the CKTA_Gauss1 GA performed the best for the Mutagenesis friendly data. Using 10-fold cross validation, it performed on average ∼0.5% better than GLPS* and Aleph, the next best performing algorithms. CKTA_Gauss1 GA also performed ∼9% better than the best unaltered kFOIL algorithm. That the CKTA GA proposed herein was able to outperform these algorithms is a promising result. Recall that Aleph is a state of the art ILP system while kFOIL is a state of the art kernel-based approach to ILP proposed in 2010. As such, these algorithms can be difficult to outperform.

| Config | mean | stddev |
|---|---|---|
| CKTA_Gauss1 | 0.866959 | 0.07243 |
| GLPS* | 0.861988 | 0.086627 |
| Aleph | 0.861696 | 0.061859 |
| kFOIL_Linear | 0.776901 | 0.076584 |

Table 13: Mutagenesis Friendly Summary

### 4.5.2 Mutagenesis Unfriendly

In Table 14 we see that the AccCKTA_withIncestAvoidance_Gauss1 GA performed the best for the Mutagenesis unfriendly data. Using 10-fold cross validation, it performed on average ∼2.4% better than Aleph, the next best performing algorithm on this data set. AccCKTA_withIncestAvoidance_Gauss1 GA also performed ∼7.1% better than the best unaltered kFOIL algorithm. This seems to imply that both the hybrid scoring (CKTA times accuracy in this case) and the incest avoidance mechanism (based on diversity) have merit. These can be viewed as additional hyperparameters to tune during a search for optimal hypotheses.

| Config | mean | stddev |
|---|---|---|
| AccCKTA_withIncestAvoidance_Gauss1 | 0.904762 | 0.297101 |
| Aleph | 0.880952 | 0.327770 |
| GLPS* | 0.857143 | 0.354169 |
| kFOIL_Linear | 0.833333 | 0.377196 |

Table 14: Mutagenesis Unfriendly Summary

### 4.5.3 Alzheimer's Inhibit Amine Reuptake

In Table 15 we see that an ensemble based on CKTA_Gauss1 GA performed the best for the Alzheimer's inhibit amine reuptake data. Furthermore, this ensemble utilized the diversity mechanism proposed in this study (i.e. diverse member

selection for ensembles). The ensemble was created from the last generation of a GA run using CKTA_Gauss1 GA. An interesting observation is that while no member of the ensemble individually outperformed the best member of the final population of the CKTA_Gauss1 GA (for the obvious reasons), the ensemble, which only contained 5 members, was able to exceed the performance of the best member by just over 0.4%. Using 10-fold cross validation, the ensemble performed on average ∼1.9% better than GLPS*, the next best performing algorithm on this data set. These results imply that ensembles created using the diverse member selection scheme proposed in this study can help to boost performance. They also show the promise of the CKTA GA proposed herein.

| Config | Ensemble Type | mean | stddev |
|---|---|---|---|
| CKTA_Gauss1 | PEN_C15_E5_D1 | 0.782737 | 0.053676 |
| GLPS* | NA | 0.763853 | 0.056995 |
| Aleph | NA | 0.758035 | 0.049117 |
| kFOIL[1] | NA | NA | NA |

Table 15: Alzheimer's Inhibit Amine Reuptake Summary

### 4.5.4 Alzheimer's Toxicity

In Table 16 we see that an ensemble based on AccCKTA_Gauss1 GA performed the best for the Alzheimer's toxicity data. Furthermore, this ensemble utilized the diversity mechanism proposed in this study (i.e. diverse member selection for ensembles). Using 10-fold cross validation, the ensemble performed on average ∼0.1% better than Aleph, the next best performing algorithm on this data set. While this result isn't quite as strong as the others, the algorithms proposed herein were still competitive and were able to narrowly edge out the other algorithms to which they were compared during a 10-fold cross validation.

[1]Unable to run the kFOIL algorithm on this data set

| Config | Ensemble Type | mean | stddev |
|---|---|---|---|
| AccCKTA_Gauss1 | NO_PEN_C15_E5_D1 | 0.796872 | 0.040931 |
| Aleph | NA | 0.795748 | 0.040715 |
| GLPS* | NA | 0.794625 | 0.040464 |
| kFOIL[2] | NA | NA | NA |

Table 16: Alzheimer's Toxicity Summary

### 4.6 Discussion

In this chapter, we experimented with employing CKTA to ILP in a few different ways, including as a fitness score for GA and as a means for promoting diversity, both for diverse member selection for ensembles and for incest avoidance in crossover. We also examined the application of a complete refinement operator in a practical setting, where we randomly selected a refinement type for a randomly selected clause, effectively serving the role of mutation in the GA. These approaches lead to promising results when applied in the ILP domain and were competitive with other current state of the art ILP algorithms. We also showed that the kernels learned via the GA can be used to visualize the data via kernel PCA. Visualizing the data in the feature space induced by the learned kernel (via kernel PCA) can guide a researcher in different directions such as investigating points of confusion in the feature space or using a clustering algorithm in the feature space and further analyzing these clusters to see what makes the data mapped to them similar.

[2]Unable to run the kFOIL algorithm on this data set

# CHAPTER 5

## Conclusions and Future Work

This study aimed to employ CKTA to inductive logic programming in the following ways:

1. as a fitness score for genetic algorithms (GA)
2. as a means for promoting diversity
   (a) as a mechanism for incest avoidance in GA
   (b) for ensembles (member selection)

In addition, it applied a complete refinement operator in a practical setting. As was shown in the previous chapter, all of these contributions lead to promising results when applied in the ILP domain and were competitive with other current state of the art ILP algorithms. We also showed that the kernels learned via the GA can be used to visualize the data via kernel PCA. Visualizing the data in the feature space induced by the learned kernel (via kernel PCA) can guide a researcher in different directions such as investigating points of confusion in the feature space or applying a clustering algorithm in the feature space and further analyzing clusters to see what makes the data mapped to them similar.

This research provides many opportunities for future research, especially since this research presented a first of kind application for centered kernel target alignment (i.e. diversity encouragement). Before closing, we will discuss a few areas for future research organized into three sections, genetic algorithm improvements, computational speed improvements, and finally, ensembles and kernel combinations.

## 5.1 Genetic Algorithm Improvements

The genetic algorithm proposed within this paper could be improved in several different ways. The selection of parent clauses for crossover is one area which could stand improvement. As implemented in this study, the scores are only temporarily adjusted for the selection of a second parent for crossover, given that the first parent has already been selected. This could be enhanced such that scores of all hypotheses are continuously adjusted during crossover (i.e. keep on adjusting the scores and maintain them - do not reset them between selection of parents). The parents could also have their scores adjusted in a different manner. For instance, they could continuously adjust their scores, but base the adjustments on relationships to existing offspring [i.e. adjust scores of candidate parents for crossover for generation $i$ based on the already existing members of the next generation, $i+1$, (which have already been created via crossover of parents from generation $i$)]. Both of these strategies could lead to a "next" generation which is more diverse. Note that this would also impact the incest avoidance approach discussed herein. Each of these different approaches could utilize different options put forth by the diversity formulation of Equation 8 (again recalling that the incest avoidance approach presented herein was a simple, degenerate case of Equation 8). Furthermore, the diversity formulation of Equation 8 could be updated so that $\alpha$ can take on values which are not in the set $\{0, 1\}$.

The GA could also be updated to include a sufficient score termination criterion. However, in practice, this could be difficult to set as it is not clear prior to experimentation where the fitness scores will converge (other than that they will be in the interval $[0, 1]$ - recall for example the inhibit amine reuptake results where the CKTA converged to 0.28); hence, it would be difficult to set the "sufficient" score. Regardless, it could be set to value closer to one (0.9 for instance),

which could result in reduced run times for simpler data sets. Alternatively, or in addition, the GA could measure how much fitness improvement occurred within the last $k$ generations, and, if the total improvement was less than some threshold, terminate the search. This would still be useful even when one does not have a priori knowledge about where the fitness score will converge.

The mutation approach could also be improved. In this study a single mutation was applied randomly to a hypothesis. The mutation was from a complete, locally finite, refinement operator, either upward or downward, again based on a random selection given that a mutation was to be performed. Some mutations are not very practical. For instance, adding a most general literal to a clause effectively does not change it. For instance, $C = P(x) \leftarrow R(x, y), Q(x)$ is not significantly impacted by the refinement which adds the literal $T(u, v)$ that is most general with respect to clause $C$, (i.e. $C = P(x) \leftarrow R(x, y), Q(x), T(u, v)$). In fact, these refinements were ignored during evaluation of the hypotheses in this study. Effectively, this will only change the behavior to add in the condition that something (which could be anything completely unrelated to x and y) satisfies $T(u, v)$. For instance, suppose $C = URIStudent(x) \leftarrow isHuman(x), isEnrolledAtURI(x)$. Then $C$ could be refined to $C = URIStudent(x) \leftarrow isHuman(x), isEnrolledAtURI(x), hasKeys(z)$. This clearly does not make much practical sense. Also, this is only saying that whoever or whatever has the keys does not even need to be associated with the person (the $x$ variable in the clause above). With this in mind, we could add some changes to mutations so that they could randomly decide between adding a literal which is most general (to maintain completeness) and adding a literal which is not most general (i.e. which has a variable matching some other variable already appearing in the clause). This could lead to each of the following with the $hasKeys$ addition

from the example.

1. $C = URIStudent(x) \leftarrow isHuman(x), isEnrolledAtURI(x), hasKeys(z)$

2. $C = URIStudent(x) \leftarrow isHuman(x), isEnrolledAtURI(x), hasKeys(x)$

Clearly, doing something like this could lead to more practical refinements more quickly as $C = URIStudent(x) \leftarrow isHuman(x), isEnrolledAtURI(x), hasKeys(x)$ is actually a reasonable clause (although students probably are not required to have keys, it is at least reasonable that a human enrolled at a university would have keys). The mutations could also be improved in the GA by allowing multiple mutations at once (i.e. randomly select whether or not to perform any mutation, and then, if mutations are to be applied, randomly select the number of mutations to apply). This could make the mutations more impactful. Only applying one rule from a complete refinement operator here and there does not seem to significantly impact results, especially when there are only a few members in the population or only a few generations being created in the GA. Performing multiple mutations could alleviate this issue.

Finally, the algorithm proposed herein could be compared to other algorithms, such as those proposed by [29, 41, 23]. Comparing to additional algorithms would enhance the strength of the results proposed within this work.

## 5.2 Computational Speed Improvement

As was noted in the Alzheimer's experimentation in Section 4.4, incest avoidance is computationally expensive for larger data sets as computing the CKTA repeatedly, with different kernel matrices, is expensive. This is because the kernel matrices have the size of the data set squared (i.e. $n^2$) and the kernel computations themselves are not "free". Furthermore, two different kernels need to be computed. Then the results are multiplied together and added in a Frobenius product ($n^2$

more multiplications and additions). For CKTA, when the second kernel is actually the target kernel formed by the outer product of the sample labels, this is less expensive since the target matrix can be computed once when the program starts and re-used throughout execution. For this reason, utilizing CKTA as a fitness function is significantly less expensive than using it for incest avoidance. The expense is also less important during diverse ensemble member selection since this is something that only happens once (versus incest avoidance, which is used during crossover on *every* generation).

It would be interesting to look into ways to improve the speed of computation. This could entail sophisticated caching, parallelization, etc. Maintaining a cache between generations would likely be of great service to the logical kernels proposed herein. If all of the hypotheses in a generation are viewed as a large collection of clauses (i.e. the unique clauses from all hypotheses), then it is likely clear that this set does not change much from one generation to the next. With this in mind, the $\phi$ values (i.e. the $\{0, 1\}$ values described in Equation 4) for unique clauses could be computed and maintained in a cache, with new $\phi$ values only being computed if they are not in this cache. A similar concept could be applied to hypotheses as a whole (i.e. to kernel values). Of course, both of these strategies would come at the cost of using additional memory.

Speeding up the kernel computations would provide benefits to CKTA based incest avoidance, fitness scores using CKTA, and to diverse ensemble member selection techniques utilizing CKTA.

### 5.3 Ensembles and Kernel Combinations

The ensembles within this study could improve their diversity by utilizing the final generations from multiple GA runs, rather than a single one (so that different kernel types, etc. are used in the creation of the ensemble). If memory is not

an issue, members from generations other than the last generation could also be utilized. Additionally, alternative approaches to max voting could be explored (i.e. using weighting based on something similar to $\gamma$ as defined in Equation 8, etc.). While the term "generation" and "final generation" in particular is utilized here, as this study has focused on GA, this diverse ensemble strategy could be applied to any set of hypotheses using kernels which are to be considered for ensemble creation.

Several promising kernels from final generations (or from any generation if compute resources are not an issue) could also be used as base kernels and combined into a new kernel using methods such as those described in [9]. The new kernel would form a convex combination of the base kernels with the goal of maximizing the new kernel's alignment with the target. Note that kernels can be combined together via multiplication, addition, multiplying by scalars, etc. to create other kernels due to the closure properties of kernels [20]. Again, this could be applied to the kernels from any set of hypotheses to be considered for ensemble creation, not just the final generation. The new kernel created in this manner could then be used to create a kernel-based classifier (i.e. SVM, etc.).

## 5.4 Closing

In closing, this study aimed both to apply centered kernel target alignment (CKTA) to inductive logic programming (ILP) in several different ways and to apply a complete refinement operator in a practical setting. A new genetic algorithm (GA) resulted from the research, utilizing a complete, locally finite refinement operator and also incorporating CKTA both as a fitness score and as a means for the promotion of diversity. As a fitness score, CKTA was used as both a standalone fitness score or as a contributor to a hybrid score utilizing accuracy (weighted or normal) of the learned logic hypothesis as well. In terms of diversity promotion,

CKTA was used for incest avoidance and as a means for creating diverse ensembles. This is the first study to employ CKTA for diversity promotion of any kind and the first to apply CKTA to ILP. The kernels in this study were created via dynamic propositionalization, where the features were learned jointly with the kernel to be used for classification via a genetic algorithm. In this sense, genetic kernels for ILP were created. The results have shown that the methods proposed herein are promising, encouraging future work. It is worth noting that the applications of CKTA in this study are not specific to ILP. They can also be used more generally in any other domain using kernels.

## LIST OF REFERENCES


[1] A. S. d. Garcez, L. C. Lamb, and D. M. Gabbay, *Neural-Symbolic Cognitive Reasoning*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[2] I. Bratko and R. D. King, "Applications of inductive logic programming," *SIGART Bulletin*, vol. 5, pp. 43–49, 01 1994.

[3] L. Kelley, P. J Shrimpton, S. Muggleton, and M. J E Sternberg, "Discovering rules for protein-ligand specificity using support vector inductive logic programming," *Protein engineering, design & selection : PEDS*, vol. 22, pp. 561–7, 08 2009.

[4] L. De Raedt and K. Kersting, "Probabilistic inductive logic programming," in *Probabilistic Inductive Logic Programming*, L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 1–27. [Online]. Available: http://dl.acm.org/citation.cfm?id=1793956.1793958

[5] Man Leung Wong and Kwong Sak Leung, "Inducing logic programs with genetic algorithms: the Genetic Logic Programming System," *IEEE Expert*, vol. 10, no. 5, pp. 68–76, Oct. 1995. [Online]. Available: http://ieeexplore.ieee.org/document/464935/

[6] Man Leung Wong and Kwong Sak Leung, "Combining genetic programming and inductive logic programming using logic grammars," in *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, vol. 2. Perth, WA, Australia: IEEE, 1995, pp. 733–736. [Online]. Available: http://ieeexplore.ieee.org/document/487476/

[7] N. Landwehr, A. Passerini, L. De Raedt, and P. Frasconi, "Fast learning of relational kernels," *Machine Learning*, vol. 78, no. 3, pp. 305–342, Mar. 2010. [Online]. Available: http://link.springer.com/10.1007/s10994-009-5163-1

[8] N. Landwehr, A. Passerini, L. De Raedt, and P. Frasconi, "kfoil: Learning simple relational kernels," in *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, ser. AAAI'06. AAAI Press, 2006, pp. 389–394. [Online]. Available: http://dl.acm.org/citation.cfm?id=1597538.1597601

[9] C. Cortes, M. Mohri, and A. Rostamizadeh, "Algorithms for learning kernels based on centered alignment," *Journal of Machine Learning Research*, vol. 13, pp. 795–828, 2012.

[10] S.-H. Nienhuys-Cheng and R. d. Wolf, *Foundations of Inductive Logic Programming*, J. Siekmann and J. G. Carbonell, Eds. Berlin, Heidelberg: Springer-Verlag, 1997.

[11] P. D. Laird, *Learning from Good and Bad Data.* Boston, MA: Kluwer Academic Publishers, 1988.

[12] A. Tamaddoni Nezhad, "Logic-based machine learning using a bounded hypothesis space: the lattice structure, refinement operators and a genetic algorithm approach," Ph.D. dissertation, Imperial College London, 2013.

[13] J. R. Quinlan, "Learning logical definitions from relations," *Mach. Learn.*, vol. 5, no. 3, pp. 239–266, Sept. 1990. [Online]. Available: https://doi.org/10.1023/A:1022699322624

[14] J. R. Quinlan and R. M. Cameron-Jones, "FOIL: A midterm report," in *Machine Learning: ECML-93*, J. Siekmann, G. Goos, J. Hartmanis, and P. B. Brazdil, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, vol. 667, pp. 1–20. [Online]. Available: http://link.springer.com/10.1007/3-540-56602-3_124

[15] J. R. Quinlan and R. M. Cameron-Jones, "Induction of logic programs: FOIL and related systems," *New Generation Computing*, vol. 13, no. 3-4, pp. 287–312, Dec. 1995. [Online]. Available: http://link.springer.com/10.1007/BF03037228

[16] L. Badea, "Perfect refinement operators can be flexible," in *Proceedings of the 14th European Conference on Artificial Intelligence, IOS Press*, Amsterdama, 2000, pp. 266–270.

[17] N. Fanizzi, S. Ferilli, N. Di Mauro, and T. M. A. Basile, "Spaces of theories with ideal refinement operators," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, ser. IJCAI'03. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 527–532. [Online]. Available: http://dl.acm.org/citation.cfm?id=1630659.1630737

[18] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, USA: MIT Press, 2001.

[19] L. H. Hamel, *Knowledge Discovery with Support Vector Machines.* New York, NY, USA: Wiley-Interscience, 2009.

[20] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis.* New York, NY, USA: Cambridge University Press, 2004.

[21] T. Gärtner, "A survey of kernels for structured data," *SIGKDD Explor. Newsl.*, vol. 5, no. 1, pp. 49–58, July 2003. [Online]. Available: http://doi.acm.org/10.1145/959242.959248

[22] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola, "On kernel-target alignment," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 367–373. [Online]. Available: http://papers.nips.cc/paper/1946-on-kernel-target-alignment.pdf

[23] S. Muggleton, H. Lodhi, A. Amini, and M. J. E. Sternberg, "Support vector inductive logic programming," in *Proceedings of the 8th International Conference on Discovery Science*, ser. DS'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 163–175. [Online]. Available: http://dx.doi.org/10.1007/11563983_15

[24] A. Srinivasan, S. H. Muggleton, M. J. E. Sternberg, and R. D. King, "Theories for mutagenicity: A study in first-order and feature-based induction," *Artif. Intell.*, vol. 85, no. 1-2, pp. 277–299, Aug. 1996. [Online]. Available: http://dx.doi.org/10.1016/0004-3702(95)00122-0

[25] A. Srinivasan, Aug 2019, online manual. [Online]. Available: http://www.cs.ox.ac.uk/activities/programinduction/Aleph/aleph.html

[26] I. de Castro Dutra, D. Page, V. Santos Costa, and J. Shavlik, "An Empirical Evaluation of Bagging in Inductive Logic Programming," in *Inductive Logic Programming*, G. Goos, J. Hartmanis, J. van Leeuwen, S. Matwin, and C. Sammut, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, vol. 2583, pp. 48–65. [Online]. Available: http://link.springer.com/10.1007/3-540-36468-4_4

[27] N. Qomariyah and D. Kazakov, "Learning from ordinal data with inductive logic programming in description logic," in *Late Breaking Papers of the 27th International Conference on Inductive Logic Programming*, N. Lachiche and C. Vrain, Eds., vol. 2085. CEUR Workshop Proceedings, 3 2018, pp. 38–50, an earlier version of this paper was accepted for publication in 2017 and entered into PURE. This extended version of the paper was subject to another round of reviews, and was published as an open access paper in these online proceedings on 29 March 2018 (see link above in this record).

[28] U. Ruckert and S. Kramer, "Margin-based first-order rule learning," *Machine Learning*, vol. 70, no. 2-3, pp. 189–206, Mar. 2008. [Online]. Available: http://link.springer.com/10.1007/s10994-007-5034-6

[29] S. Muggleton and A. Tamaddoni-Nezhad, "QG/GA: a stochastic search for Progol," *Machine Learning*, vol. 70, no. 2-3, pp. 121–133, Mar. 2008. [Online]. Available: http://link.springer.com/10.1007/s10994-007-5029-3

[30] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 368–386, June 2013.

[31] N. Landwehr, K. Kersting, and L. D. Raedt, "Integrating naïve bayes and foil," *J. Mach. Learn. Res.*, vol. 8, pp. 481–507, Dec. 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1314498.1314516

[32] C. Rouveirol and J. F. Puget, *Proceedings of the fourth European Working Session on Learning*. Pitman, 1989, pp. 201–211.

[33] S. Muggleton, "Inductive logic programming: Derivations, successes and shortcomings," *SIGART Bull.*, vol. 5, no. 1, pp. 5–11, Jan 1994. [Online]. Available: http://doi.acm.org/10.1145/181668.181671

[34] C. Rouveirol, "Flattening and saturation: Two representation changes for generalization," *Machine Learning*, vol. 14, pp. 219–232, Feb 1994. [Online]. Available: https://doi.org/10.1023/A:1022678217288

[35] S. Muggleton, mutagenesis data. [Online]. Available: https://www.doc.ic.ac.uk/~shm/mutagenesis.html#progol

[36] S. Muggleton, alzheimer's data. [Online]. Available: https://www.doc.ic.ac.uk/~shm/alzheimers.html#kingetal

[37] B. Ott, Aug 2019, k folds data. [Online]. Available: https://github.com/benott-cs/ILPData

[38] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of Medicinal Chemistry*, vol. 34, no. 2, pp. 786–797, 1991.

[39] G. M. Shutske, F. A. Pierrat, K. J. Kapples, M. L. Cornfeldt, M. R. Szewczak, F. P. Huger, G. M. Bores, V. Haroutunian, and K. L. Davis, "9-amino-1,2,3,4-tetrahydroacridin-1-ols. synthesis and evaluation as potential alzheimer's disease therapeutics," *Journal of Medicinal Chemistry*, vol. 32, no. 8, pp. 1805–1813, 1989.

[40] R. D. King, A. Srinivasan, and M. J. E. Sternberg, "Relating chemical activity to structure: An examination of ilp successes," *New Generation Computing*, vol. 14, no. 1, pp. 109–109, Mar 1996. [Online]. Available: https://doi.org/10.1007/BF03037220

[41] A. Tamaddoni-Nezhad and S. Muggleton, "The lattice structure and refinement operators for the hypothesis space bounded by a bottom clause," *Machine Learning*, vol. 76, no. 1, pp. 37–72, July 2009. [Online]. Available: http://link.springer.com/10.1007/s10994-009-5117-7

[42] B. Ott, Aug 2019, cKTAGA Code. [Online]. Available: https://github.com/benott-cs/CKTAGA

[43] I. Maznitsa, Aug 2019, java based Edinburgh Prolog parser. [Online]. Available: https://github.com/raydac/java-prolog-parser

# APPENDIX A

## Complete Experiment Results

### A.1 Complete Results

The complete results for all hyperparameters used for each data set are presented in tabular form below.

#### A.1.1 Mutagenesis Friendly

| Config | C-Val | mean | stddev |
|---|---|---|---|
| CKTA_Gauss1 | 1 | 0.866959 | 0.07243 |
| GLPS* | logic-NA | 0.861988 | 0.086627 |
| Aleph | logic-NA | 0.861696 | 0.061859 |
| CKTA_Gauss1 | 10 | 0.861696 | 0.076249 |
| CKTA_withIncestAvoidance_Gauss1 | 10 | 0.861111 | 0.063782 |
| CKTA_withIncestAvoidance_Gauss1 | 0.1 | 0.860819 | 0.08545 |
| wAccCKTA_Linear | 1 | 0.85614 | 0.083352 |
| CKTA_withIncestAvoidance_Gauss1 | 1 | 0.855848 | 0.071994 |
| wAccCKTA_withIncestAvoidance_Linear | logic-NA | 0.855848 | 0.071994 |
| wAccCKTA_Linear | 10 | 0.855848 | 0.087438 |
| wAccCKTA_withIncestAvoidance_Linear | 10 | 0.850877 | 0.089467 |
| CKTA_withIncestAvoidance_Poly2 | 0.1 | 0.850585 | 0.106309 |
| wAccCKTA_withIncestAvoidance_Linear | 1 | 0.845029 | 0.097476 |
| wAccCKTA_Gauss1 | 0.1 | 0.840351 | 0.078992 |
| wAccCKTA_Gauss1 | logic-NA | 0.840351 | 0.078992 |
| CKTA_Gauss1 | 0.1 | 0.840058 | 0.062043 |
| CKTA_withIncestAvoidance_Poly2 | 1 | 0.840058 | 0.08407 |
| wAccCKTA_Gauss1 | 1 | 0.835088 | 0.084331 |
| CKTA_Linear | 1 | 0.830409 | 0.080561 |
| CKTA_Linear | 10 | 0.830409 | 0.080561 |
| wAccCKTA_Gauss1 | 10 | 0.829825 | 0.092401 |

| | | | |
|---|---|---|---|
| CKTA_Poly2 | 0.1 | 0.829825 | 0.113344 |
| AccCKTA_withIncestAvoidance_Gauss1 | 1 | 0.829532 | 0.082196 |
| AccCKTA_withIncestAvoidance_Gauss1 | 10 | 0.829532 | 0.082196 |
| AccCKTA_withIncestAvoidance_Linear | logic-NA | 0.829532 | 0.086674 |
| AccCKTA_Gauss1 | 10 | 0.825146 | 0.088665 |
| CKTA_withIncestAvoidance_Linear | 0.1 | 0.825146 | 0.124618 |
| CKTA_withIncestAvoidance_Poly2 | 0.01 | 0.824854 | 0.089014 |
| AccCKTA_Linear | 1 | 0.824561 | 0.074789 |
| AccCKTA_Linear | 10 | 0.824561 | 0.074789 |
| wAccCKTA_withIncestAvoidance_Gauss1 | 10 | 0.824561 | 0.082611 |
| AccCKTA_withIncestAvoidance_Poly2 | 0.1 | 0.824561 | 0.122629 |
| AccCKTA_withIncestAvoidance_Gauss1 | 0.1 | 0.824269 | 0.090096 |
| AccCKTA_withIncestAvoidance_Gauss1 | logic-NA | 0.824269 | 0.090096 |
| wAccCKTA_withIncestAvoidance_Poly2 | 10 | 0.824269 | 0.101911 |
| CKTA_withIncestAvoidance_Poly2 | 10 | 0.823392 | 0.119205 |
| AccCKTA_Gauss1 | 0.1 | 0.819591 | 0.078435 |
| AccCKTA_Gauss1 | logic-NA | 0.819591 | 0.078435 |
| wAccCKTA_withIncestAvoidance_Gauss1 | 1 | 0.819298 | 0.066589 |
| GLPS*1 | logic-NA | 0.819298 | 0.108479 |
| CKTA_Linear | 0.1 | 0.819298 | 0.111519 |
| wAccCKTA_Linear | logic-NA | 0.819006 | 0.093815 |
| wAccCKTA_withIncestAvoidance_Gauss1 | 0.1 | 0.818713 | 0.063838 |
| wAccCKTA_withIncestAvoidance_Gauss1 | logic-NA | 0.818713 | 0.063838 |
| wAccCKTA_withIncestAvoidance_Poly2 | 0.1 | 0.818713 | 0.115134 |
| AccCKTA_withIncestAvoidance_Poly2 | logic-NA | 0.81462 | 0.135728 |
| AccCKTA_Gauss1 | 1 | 0.814327 | 0.089512 |
| wAccCKTA_withIncestAvoidance_Poly2 | logic-NA | 0.813743 | 0.089436 |
| wAccCKTA_withIncestAvoidance_Poly2 | 1 | 0.813743 | 0.092813 |
| CKTA_withIncestAvoidance_Linear | 1 | 0.813158 | 0.096274 |
| CKTA_withIncestAvoidance_Linear | 10 | 0.813158 | 0.096274 |
| AccCKTA_withIncestAvoidance_Linear | 10 | 0.809064 | 0.092629 |

| | | | |
|---|---|---|---|
| AccCKTA_withIncestAvoidance_Poly2 | 1 | 0.808772 | 0.122754 |
| AccCKTA_withIncestAvoidance_Poly2 | 10 | 0.808772 | 0.122754 |
| CKTA_Poly2 | 1 | 0.80848 | 0.102957 |
| CKTA_Poly2 | 10 | 0.80848 | 0.102957 |
| AccCKTA_Poly2 | logic-NA | 0.80848 | 0.103637 |
| AccCKTA_withIncestAvoidance_Linear | 1 | 0.803509 | 0.092678 |
| CKTA_Poly2 | 0.01 | 0.803509 | 0.108014 |
| Aleph1 | logic-NA | 0.803216 | 0.099339 |
| wAccCKTA_Poly2 | 0.1 | 0.803216 | 0.108883 |
| wAccCKTA_withIncestAvoidance_Linear | 0.1 | 0.802924 | 0.073096 |
| AccCKTA_withIncestAvoidance_Linear | 0.1 | 0.798538 | 0.104415 |
| wAccCKTA_Poly2 | 1 | 0.79269 | 0.104033 |
| wAccCKTA_Poly2 | 10 | 0.79269 | 0.104033 |
| wAccCKTA_Poly2 | logic-NA | 0.79269 | 0.104033 |
| wAccCKTA_Linear | 0.1 | 0.79269 | 0.109148 |
| AccCKTA_Linear | logic-NA | 0.787427 | 0.113329 |
| AccCKTA_Poly2 | 0.1 | 0.776316 | 0.10733 |
| AccCKTA_Poly2 | 1 | 0.77076 | 0.102169 |
| AccCKTA_Linear | 0.1 | 0.766667 | 0.113344 |
| AccCKTA_Poly2 | 10 | 0.765497 | 0.105415 |
| CKTA_Gauss1 | logic-NA | 0.739474 | 0.130243 |
| CKTA_withIncestAvoidance_Gauss1 | logic-NA | 0.739181 | 0.150267 |
| AccCKTA_Poly2 | 0.01 | 0.72924 | 0.12679 |
| AccCKTA_withIncestAvoidance_Poly2 | 0.01 | 0.712573 | 0.111094 |
| CKTA_Linear | logic-NA | 0.702632 | 0.130648 |
| CKTA_withIncestAvoidance_Linear | logic-NA | 0.674854 | 0.141193 |
| wAccCKTA_withIncestAvoidance_Poly2 | 0.01 | 0.674854 | 0.141193 |
| CKTA_withIncestAvoidance_Linear | 0.01 | 0.669591 | 0.123201 |
| CKTA_Poly2 | logic-NA | 0.669591 | 0.130481 |
| CKTA_withIncestAvoidance_Poly2 | logic-NA | 0.664327 | 0.126095 |
| CKTA_withIncestAvoidance_Gauss1 | 0.01 | 0.664327 | 0.126095 |

| | | | |
|---|---|---|---|
| CKTA_Linear | 0.01 | 0.664327 | 0.126095 |
| CKTA_Gauss1 | 0.01 | 0.664327 | 0.126095 |
| AccCKTA_withIncestAvoidance_Linear | 0.01 | 0.664327 | 0.126095 |
| AccCKTA_withIncestAvoidance_Gauss1 | 0.01 | 0.664327 | 0.126095 |
| AccCKTA_Linear | 0.01 | 0.664327 | 0.126095 |
| AccCKTA_Gauss1 | 0.01 | 0.664327 | 0.126095 |
| wAccCKTA_withIncestAvoidance_Gauss1 | 0.01 | 0.664327 | 0.126095 |
| wAccCKTA_Linear | 0.01 | 0.664327 | 0.126095 |
| wAccCKTA_Gauss1 | 0.01 | 0.664327 | 0.126095 |
| wAccCKTA_withIncestAvoidance_Linear | 0.01 | 0.664327 | 0.126095 |
| wAccCKTA_Poly2 | 0.01 | 0.659064 | 0.12381 |

Table A.1: Mutagenesis Friendly Complete Results

### A.1.2 Mutagenesis Unfriendly

| Config | C-Val | mean | stddev |
|---|---|---|---|
| AccCKTA_withIncestAvoidance_Gauss1 | 1 | 0.904762 | 0.297102 |
| AccCKTA_withIncestAvoidance_Gauss1 | 10 | 0.904762 | 0.297102 |
| AccCKTA_withIncestAvoidance_Gauss1 | logic-NA | 0.904762 | 0.297102 |
| wAccCKTA_withIncestAvoidance_Poly2 | logic-NA | 0.904762 | 0.297102 |
| wAccCKTA_withIncestAvoidance_Gauss1 | 1 | 0.904762 | 0.297102 |
| wAccCKTA_withIncestAvoidance_Gauss1 | 10 | 0.904762 | 0.297102 |
| wAccCKTA_withIncestAvoidance_Gauss1 | logic-NA | 0.904762 | 0.297102 |
| wAccCKTA_Linear | 10 | 0.904762 | 0.297102 |
| wAccCKTA_Linear | logic-NA | 0.904762 | 0.297102 |
| CKTA_withIncestAvoidance_Gauss1 | 1 | 0.880952 | 0.32777 |
| CKTA_withIncestAvoidance_Gauss1 | 10 | 0.880952 | 0.32777 |
| CKTA_Gauss1 | 1 | 0.880952 | 0.32777 |
| CKTA_Gauss1 | 10 | 0.880952 | 0.32777 |
| AccCKTA_withIncestAvoidance_Linear | 10 | 0.880952 | 0.32777 |
| AccCKTA_withIncestAvoidance_Linear | logic-NA | 0.880952 | 0.32777 |
| AccCKTA_withIncestAvoidance_Poly2 | 1 | 0.880952 | 0.32777 |

| | | | |
|---|---|---|---|
| AccCKTA_withIncestAvoidance_Poly2 | 10 | 0.880952 | 0.32777 |
| AccCKTA_withIncestAvoidance_Poly2 | logic-NA | 0.880952 | 0.32777 |
| AccCKTA_Gauss1 | 1 | 0.880952 | 0.32777 |
| AccCKTA_Gauss1 | 10 | 0.880952 | 0.32777 |
| AccCKTA_Gauss1 | logic-NA | 0.880952 | 0.32777 |
| Aleph | logic-NA | 0.880952 | 0.32777 |
| wAccCKTA_withIncestAvoidance_Linear | logic-NA | 0.880952 | 0.32777 |
| wAccCKTA_withIncestAvoidance_Poly2 | 1 | 0.880952 | 0.32777 |
| wAccCKTA_withIncestAvoidance_Poly2 | 10 | 0.880952 | 0.32777 |
| wAccCKTA_Poly2 | 1 | 0.880952 | 0.32777 |
| wAccCKTA_Poly2 | 10 | 0.880952 | 0.32777 |
| CKTA_withIncestAvoidance_Linear | 10 | 0.857143 | 0.354169 |
| CKTA_withIncestAvoidance_Poly2 | 1 | 0.857143 | 0.354169 |
| CKTA_withIncestAvoidance_Poly2 | 10 | 0.857143 | 0.354169 |
| CKTA_withIncestAvoidance_Poly2 | 0.1 | 0.857143 | 0.354169 |
| AccCKTA_withIncestAvoidance_Poly2 | 0.1 | 0.857143 | 0.354169 |
| AccCKTA_Linear | 0.1 | 0.857143 | 0.354169 |
| AccCKTA_Poly2 | 1 | 0.857143 | 0.354169 |
| AccCKTA_Poly2 | 10 | 0.857143 | 0.354169 |
| AccCKTA_Poly2 | 0.1 | 0.857143 | 0.354169 |
| AccCKTA_Poly2 | logic-NA | 0.857143 | 0.354169 |
| GLPS* | logic-NA | 0.857143 | 0.354169 |
| wAccCKTA_withIncestAvoidance_Linear | 10 | 0.857143 | 0.354169 |
| wAccCKTA_withIncestAvoidance_Poly2 | 0.1 | 0.857143 | 0.354169 |
| wAccCKTA_withIncestAvoidance_Gauss1 | 0.1 | 0.857143 | 0.354169 |
| wAccCKTA_Linear | 1 | 0.857143 | 0.354169 |
| wAccCKTA_Linear | 0.1 | 0.857143 | 0.354169 |
| wAccCKTA_Poly2 | logic-NA | 0.857143 | 0.354169 |
| CKTA_withIncestAvoidance_Linear | 1 | 0.833333 | 0.377195 |
| CKTA_withIncestAvoidance_Linear | 0.1 | 0.833333 | 0.377195 |
| CKTA_Linear | 1 | 0.833333 | 0.377195 |

| | | | |
|---|---|---|---|
| CKTA_Linear | 0.1 | 0.833333 | 0.377195 |
| CKTA_Poly2 | 1 | 0.833333 | 0.377195 |
| CKTA_Poly2 | 10 | 0.833333 | 0.377195 |
| CKTA_Poly2 | 0.1 | 0.833333 | 0.377195 |
| CKTA_Gauss1 | 0.1 | 0.833333 | 0.377195 |
| AccCKTA_withIncestAvoidance_Linear | 1 | 0.833333 | 0.377195 |
| AccCKTA_withIncestAvoidance_Gauss1 | 0.1 | 0.833333 | 0.377195 |
| AccCKTA_Linear | 1 | 0.833333 | 0.377195 |
| AccCKTA_Linear | 10 | 0.833333 | 0.377195 |
| wAccCKTA_withIncestAvoidance_Linear | 1 | 0.833333 | 0.377195 |
| wAccCKTA_Poly2 | 0.1 | 0.833333 | 0.377195 |
| CKTA_withIncestAvoidance_Gauss1 | 0.1 | 0.809524 | 0.397437 |
| AccCKTA_Linear | logic-NA | 0.809524 | 0.397437 |
| AccCKTA_Gauss1 | 0.1 | 0.809524 | 0.397437 |
| wAccCKTA_Gauss1 | 1 | 0.809524 | 0.397437 |
| wAccCKTA_Gauss1 | 10 | 0.809524 | 0.397437 |
| wAccCKTA_Gauss1 | 0.1 | 0.809524 | 0.397437 |
| wAccCKTA_Gauss1 | logic-NA | 0.809524 | 0.397437 |
| CKTA_Linear | 10 | 0.785714 | 0.4153 |
| CKTA_Gauss1 | logic-NA | 0.785714 | 0.4153 |
| AccCKTA_withIncestAvoidance_Linear | 0.1 | 0.785714 | 0.4153 |
| wAccCKTA_withIncestAvoidance_Linear | 0.1 | 0.785714 | 0.4153 |
| CKTA_withIncestAvoidance_Poly2 | 0.01 | 0.761905 | 0.431081 |
| CKTA_withIncestAvoidance_Gauss1 | logic-NA | 0.761905 | 0.431081 |
| CKTA_Poly2 | 0.01 | 0.738095 | 0.445001 |
| AccCKTA_Poly2 | 0.01 | 0.714286 | 0.45723 |
| CKTA_withIncestAvoidance_Linear | 0.01 | 0.690476 | 0.467901 |
| CKTA_withIncestAvoidance_Gauss1 | 0.01 | 0.690476 | 0.467901 |
| CKTA_Linear | 0.01 | 0.690476 | 0.467901 |
| CKTA_Gauss1 | 0.01 | 0.690476 | 0.467901 |
| AccCKTA_withIncestAvoidance_Linear | 0.01 | 0.690476 | 0.467901 |

| | | | |
|---|---|---|---|
| AccCKTA_withIncestAvoidance_Poly2 | 0.01 | 0.690476 | 0.467901 |
| AccCKTA_withIncestAvoidance_Gauss1 | 0.01 | 0.690476 | 0.467901 |
| AccCKTA_Linear | 0.01 | 0.690476 | 0.467901 |
| AccCKTA_Gauss1 | 0.01 | 0.690476 | 0.467901 |
| wAccCKTA_withIncestAvoidance_Linear | 0.01 | 0.690476 | 0.467901 |
| wAccCKTA_withIncestAvoidance_Poly2 | 0.01 | 0.690476 | 0.467901 |
| wAccCKTA_withIncestAvoidance_Gauss1 | 0.01 | 0.690476 | 0.467901 |
| wAccCKTA_Linear | 0.01 | 0.690476 | 0.467901 |
| wAccCKTA_Poly2 | 0.01 | 0.690476 | 0.467901 |
| wAccCKTA_Gauss1 | 0.01 | 0.690476 | 0.467901 |
| CKTA_Linear | logic-NA | 0.666667 | 0.477119 |
| CKTA_withIncestAvoidance_Linear | logic-NA | 0.571429 | 0.50087 |
| CKTA_withIncestAvoidance_Poly2 | logic-NA | 0.52381 | 0.505487 |
| CKTA_Poly2 | logic-NA | 0.47619 | 0.505487 |

Table A.2: Mutagenesis Unfriendly Complete Results

### A.1.3 Alzheimer's - Inhibit Amine Reuptake

| Config | C-Val | mean | stddev |
|---|---|---|---|
| CKTA_Gauss1 | 10 | 0.778389 | 0.047727 |
| CKTA_Gauss1 | 1 | 0.776939 | 0.049248 |
| CKTA_Gauss1 | 0.1 | 0.765217 | 0.055672 |
| GLPS* | logic-NA | 0.763853 | 0.056995 |
| AccCKTA_Gauss1 | 1 | 0.762340 | 0.051351 |
| AccCKTA_Gauss1 | 10 | 0.762340 | 0.051351 |
| AccCKTA_Gauss1 | 0.1 | 0.762340 | 0.062660 |
| wAccCKTA_Gauss1 | 1 | 0.760806 | 0.073323 |
| wAccCKTA_Gauss1 | 10 | 0.760806 | 0.073323 |
| Aleph | logic-NA | 0.758035 | 0.049117 |
| GLPSWMutation* | logic-NA | 0.755136 | 0.048355 |
| wAccCKTA_Gauss1 | 0.01 | 0.751982 | 0.063437 |
| wAccCKTA_Gauss1 | 0.1 | 0.746228 | 0.057962 |

| wAccCKTA_Gauss1 | logic-NA | 0.746228 | 0.057962 |
|---|---|---|---|
| CKTA_Gauss1 | 0.01 | 0.744629 | 0.089664 |
| AccCKTA_Gauss1 | logic-NA | 0.741880 | 0.076501 |
| AccCKTA_Gauss1 | 0.01 | 0.737553 | 0.069953 |
| CKTA_Linear | 10 | 0.736125 | 0.062369 |
| CKTA_Linear | 1 | 0.733227 | 0.060617 |
| CKTA_Linear | 0.1 | 0.730413 | 0.069353 |
| AccCKTA_Poly2 | 1 | 0.705541 | 0.072494 |
| AccCKTA_Poly2 | 10 | 0.705541 | 0.072494 |
| AccCKTA_Poly2 | 0.1 | 0.705541 | 0.072494 |
| AccCKTA_Poly2 | 0.01 | 0.705541 | 0.072494 |
| wAccCKTA_Linear | 0.1 | 0.702643 | 0.077222 |
| wAccCKTA_Linear | 0.01 | 0.702643 | 0.077222 |
| wAccCKTA_Poly2 | 0.01 | 0.701194 | 0.079289 |
| wAccCKTA_Poly2 | 1 | 0.699829 | 0.078533 |
| wAccCKTA_Poly2 | 10 | 0.699829 | 0.078533 |
| wAccCKTA_Poly2 | 0.1 | 0.699829 | 0.078533 |
| AccCKTA_Linear | 1 | 0.698210 | 0.069203 |
| AccCKTA_Linear | 10 | 0.698210 | 0.069203 |
| CKTA_Poly2 | 1 | 0.695460 | 0.089384 |
| CKTA_Poly2 | 10 | 0.695460 | 0.089384 |
| CKTA_Linear | 0.01 | 0.695354 | 0.085698 |
| CKTA_Poly2 | 0.1 | 0.694011 | 0.092826 |
| wAccCKTA_Linear | 1 | 0.693947 | 0.091877 |
| wAccCKTA_Linear | 10 | 0.693947 | 0.091877 |
| AccCKTA_Linear | 0.1 | 0.692327 | 0.063824 |
| wAccCKTA_Linear | logic-NA | 0.690963 | 0.070768 |
| AccCKTA_Poly2 | logic-NA | 0.688086 | 0.082911 |
| AccCKTA_Linear | 0.01 | 0.685017 | 0.076567 |
| AccCKTA_Linear | logic-NA | 0.684974 | 0.085534 |
| wAccCKTA_Poly2 | logic-NA | 0.683610 | 0.084299 |

| | | | |
|---|---|---|---|
| CKTA_Poly2 | 0.01 | 0.680733 | 0.100672 |
| CKTA_Poly2 | logic-NA | 0.601790 | 0.139118 |
| CKTA_Linear | logic-NA | 0.555243 | 0.130625 |
| CKTA_Gauss1 | logic-NA | 0.539812 | 0.107743 |

Table A.3: Alzheimer's - Inhibit Amine Reuptake Complete Results

### A.1.4 Alzheimer's - Toxicity

| Config | C-Val | mean | stddev |
|---|---|---|---|
| Aleph | logic-NA | 0.795748 | 0.040715 |
| AccCKTA_Gauss1 | 1 | 0.795748 | 0.041059 |
| AccCKTA_Gauss1 | 10 | 0.795748 | 0.041059 |
| AccCKTA_Gauss1 | 0.1 | 0.794625 | 0.040464 |
| GLPS* | logic-NA | 0.794625 | 0.040464 |
| CKTA_Gauss1 | 1 | 0.793488 | 0.040613 |
| CKTA_Gauss1 | 10 | 0.793488 | 0.040613 |
| CKTA_Gauss1 | 0.1 | 0.793488 | 0.044565 |
| GLPSWMutation* | logic-NA | 0.792377 | 0.040892 |
| AccCKTA_Gauss1 | 0.01 | 0.792377 | 0.04666 |
| AccCKTA_Gauss1 | logic-NA | 0.792377 | 0.04666 |
| wAccCKTA_Gauss1 | 1 | 0.785636 | 0.047058 |
| wAccCKTA_Gauss1 | 10 | 0.785636 | 0.047058 |
| wAccCKTA_Gauss1 | 0.1 | 0.783376 | 0.047913 |
| wAccCKTA_Gauss1 | 0.01 | 0.780005 | 0.045267 |
| wAccCKTA_Gauss1 | logic-NA | 0.780005 | 0.045267 |
| CKTA_Gauss1 | 0.01 | 0.77311 | 0.037073 |
| wAccCKTA_Linear | 1 | 0.750549 | 0.045186 |
| wAccCKTA_Linear | 10 | 0.750549 | 0.045186 |
| AccCKTA_Linear | 1 | 0.7494 | 0.053041 |
| AccCKTA_Linear | 10 | 0.7494 | 0.053041 |
| wAccCKTA_Linear | 0.1 | 0.746029 | 0.043905 |
| CKTA_Poly2 | 0.1 | 0.744969 | 0.055819 |

| | | | |
|---|---|---|---|
| AccCKTA_Linear | 0.1 | 0.738113 | 0.055089 |
| CKTA_Poly2 | 1 | 0.737079 | 0.061255 |
| CKTA_Poly2 | 10 | 0.737079 | 0.061255 |
| CKTA_Linear | 1 | 0.737015 | 0.033765 |
| CKTA_Linear | 10 | 0.737015 | 0.033765 |
| wAccCKTA_Poly2 | 0.1 | 0.735981 | 0.037681 |
| wAccCKTA_Poly2 | 1 | 0.734844 | 0.036211 |
| wAccCKTA_Poly2 | 10 | 0.734844 | 0.036211 |
| AccCKTA_Poly2 | 0.1 | 0.730235 | 0.04926 |
| AccCKTA_Poly2 | 1 | 0.729099 | 0.047995 |
| AccCKTA_Poly2 | 10 | 0.729099 | 0.047995 |
| CKTA_Linear | 0.1 | 0.728013 | 0.044142 |
| CKTA_Poly2 | 0.01 | 0.72689 | 0.05698 |
| wAccCKTA_Linear | 0.01 | 0.717748 | 0.053195 |
| AccCKTA_Linear | 0.01 | 0.715475 | 0.057385 |
| wAccCKTA_Poly2 | 0.01 | 0.711172 | 0.041389 |
| CKTA_Linear | 0.01 | 0.709895 | 0.047779 |
| CKTA_Gauss1 | logic-NA | 0.707572 | 0.142736 |
| AccCKTA_Linear | logic-NA | 0.704265 | 0.050619 |
| AccCKTA_Poly2 | 0.01 | 0.702056 | 0.047148 |
| AccCKTA_Poly2 | logic-NA | 0.688407 | 0.040816 |
| wAccCKTA_Poly2 | logic-NA | 0.687308 | 0.027289 |
| wAccCKTA_Linear | logic-NA | 0.67823 | 0.049884 |
| CKTA_Linear | logic-NA | 0.562283 | 0.082722 |
| CKTA_Poly2 | logic-NA | 0.532699 | 0.076225 |

Table A.4: Alzheimer's - Toxicity Complete Results

# APPENDIX B

## Resources Used for Experimentation

The following code, hardware, data, and third party software/tools were utilized for this research.

### B.1 Code

The code used for this dissertation is available at [42]. Igor Maznitsa wrote a wonderful prolog parser [43] which was used as a springboard for the code base of this study. The prolog parser was modified to support Aleph constructs, as this was necessary for the experimentation performed in support of this study. After the parser was written, the GA code was written to adapt the prolog constructs into the appropriate data structures for the GA (i.e. AND-OR trees, etc.).

### B.2 Hardware

All experiments were performed on a Lenovo ThinkPad with the following specifications:

1. 64 GB RAM
2. Intel Core i7-6820HQ CPU operating at 2.70GHz
3. 1 TB SSD drive

Additionally, when space became an issue, two 2TB external SSDs using USB-C were also used.

### B.3 Data

The data sets used for this study are available at [37]. Results are typically difficult to reproduce in the machine learning community because authors either do

not make their code available or do not make their data sets available. Within the past few years, researchers have begun to recognize this issue and to take measures to correct it (see for example, OpenAI gym for reinforcement learning). Hopefully, making the code and data used for this study publicly available will be of use to other researchers who may be interested in advancing this research.

### B.4 Third Party Software and Tools

As was mentioned, Maznitsa's prolog parser [43] was used as a springboard for this research. Other third party tools used in this research include:

1. Yap prolog version 6.2.3
2. Aleph version 5
3. java openjdk version 1.8.0_131
4. Intellij Community 2017.1.4
5. perl 5 version 16

## BIBLIOGRAPHY


Badea, L., "Perfect refinement operators can be flexible," in *Proceedings of the 14th European Conference on Artificial Intelligence, IOS Press*, Amsterdama, 2000, pp. 266–270.

Bhowan, U., Johnston, M., Zhang, M., and Yao, X., "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 368–386, June 2013.

Bicer, V., Tran, T., and Gossen, A., "Relational Kernel Machines for Learning from Graph-Structured RDF Data," in *The Semantic Web: Research and Applications*, Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., and Pan, J., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, vol. 6643, pp. 47–62. [Online]. Available: http://link.springer.com/10.1007/978-3-642-21034-1_4

Bratko, I. and D. King, R., "Applications of inductive logic programming," *SIGART Bulletin*, vol. 5, pp. 43–49, 01 1994.

Cannon, E. O., Amini, A., Bender, A., Sternberg, M. J. E., Muggleton, S. H., Glen, R. C., and Mitchell, J. B. O., "Support vector inductive logic programming outperforms the naive Bayes classifier and inductive logic programming for the classification of bioactive chemical compounds," *Journal of Computer-Aided Molecular Design*, vol. 21, no. 5, pp. 269–280, Apr. 2007. [Online]. Available: http://link.springer.com/10.1007/s10822-007-9113-3

Conceicao, J. P. D., "The Aleph System Made Easy," Master's thesis, Universidade do Porto, 2008.

Cortes, C., Mohri, M., and Rostamizadeh, A., "Generalization bounds for learning kernels," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. USA: Omnipress, 2010, pp. 247–254. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104355

Cortes, C., Mohri, M., and Rostamizadeh, A., "Two-stage learning kernel algorithms," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. USA: Omnipress, 2010, pp. 239–246. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104354

Cortes, C., Mohri, M., and Rostamizadeh, A., "Algorithms for learning kernels based on centered alignment," *Journal of Machine Learning Research*, vol. 13, pp. 795–828, 2012.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. S., "On kernel-target alignment," in *Advances in Neural Information Processing Systems 14*, Dietterich, T. G., Becker, S., and Ghahramani, Z., Eds. MIT Press, 2002, pp. 367–373. [Online]. Available: http://papers.nips.cc/paper/1946-on-kernel-target-alignment.pdf

de Castro Dutra, I., Page, D., Santos Costa, V., and Shavlik, J., "An Empirical Evaluation of Bagging in Inductive Logic Programming," in *Inductive Logic Programming*, Goos, G., Hartmanis, J., van Leeuwen, J., Matwin, S., and Sammut, C., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, vol. 2583, pp. 48–65. [Online]. Available: http://link.springer.com/10.1007/3-540-36468-4_4

De Raedt, L. and Kersting, K., "Probabilistic inductive logic programming," in *Probabilistic Inductive Logic Programming*, De Raedt, L., Frasconi, P., Kersting, K., and Muggleton, S., Eds. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 1–27. [Online]. Available: http://dl.acm.org/citation.cfm?id=1793956.1793958

De Raedt, L. and Ramon, J., "Condensed representations for inductive logic programming," in *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning*, ser. KR'04. AAAI Press, 2004, pp. 438–446. [Online]. Available: http://dl.acm.org/citation.cfm?id=3029848.3029905

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., and Hansch, C., "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of Medicinal Chemistry*, vol. 34, no. 2, pp. 786–797, 1991.

Diligenti, M., Gori, M., Maggini, M., and Rigutini, L., "Multitask kernel-based learning with logic constraints," in *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2010, pp. 433–438. [Online]. Available: http://dl.acm.org/citation.cfm?id=1860967.1861053

Diligenti, M., Gori, M., Maggini, M., and Rigutini, L., "Bridging logic and kernel machines," *Machine Learning*, vol. 86, no. 1, pp. 57–88, Jan. 2012. [Online]. Available: http://link.springer.com/10.1007/s10994-011-5243-x

Eiben, A., Schut, M., and de Wilde, A., "Boosting Genetic Algorithms with Self-Adaptive Selection," in *2006 IEEE International Conference on*

*Evolutionary Computation*. Vancouver, BC, Canada: IEEE, 2006, pp. 477–482. [Online]. Available: http://ieeexplore.ieee.org/document/1688348/

Fanizzi, N., Ferilli, S., Di Mauro, N., and Basile, T. M. A., "Spaces of theories with ideal refinement operators," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, ser. IJCAI'03. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 527–532. [Online]. Available: http://dl.acm.org/citation.cfm?id=1630659.1630737

Frasconi, P. and Passerini, A., "Learning with kernels and logical representations," in *Probabilistic Inductive Logic Programming*, De Raedt, L., Frasconi, P., Kersting, K., and Muggleton, S., Eds. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 56–91. [Online]. Available: http://dl.acm.org/citation.cfm?id=1793956.1793961

Friedman, J. H. and Popescu, B. E., "Predictive learning via rule ensembles," *The Annals of Applied Statistics*, vol. 2, no. 3, pp. 916–954, Sept. 2008. [Online]. Available: http://projecteuclid.org/euclid.aoas/1223908046

Fung, G. M., Mangasarian, O. L., and Shavlik, J. W., "Knowledge-based support vector machine classifiers," in *Proceedings of the 15th International Conference on Neural Information Processing Systems*, ser. NIPS'02. Cambridge, MA, USA: MIT Press, 2002, pp. 537–544. [Online]. Available: http://dl.acm.org/citation.cfm?id=2968618.2968685

Fung, G. M., Mangasarian, O. L., and Shavlik, J. W., "Knowledge-Based Nonlinear Kernel Classifiers," in *Learning Theory and Kernel Machines*, Goos, G., Hartmanis, J., van Leeuwen, J., SchÃűlkopf, B., and Warmuth, M. K., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, vol. 2777, pp. 102–113. [Online]. Available: http://link.springer.com/10.1007/978-3-540-45167-9_9

Garcez, A. S. d., Lamb, L. C., and Gabbay, D. M., *Neural-Symbolic Cognitive Reasoning*, 1st ed. Springer Publishing Company, Incorporated, 2008.

Gärtner, T., "A survey of kernels for structured data," *SIGKDD Explor. Newsl.*, vol. 5, no. 1, pp. 49–58, July 2003. [Online]. Available: http://doi.acm.org/10.1145/959242.959248

Hamel, L. H., *Knowledge Discovery with Support Vector Machines*. New York, NY, USA: Wiley-Interscience, 2009.

He, D., Wang, Z., Yang, B., and Zhou, C., "Genetic Algorithm with Ensemble Learning for Detecting Community Structure in Complex Networks," in *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*. Seoul, Korea: IEEE, 2009, pp. 702–707. [Online]. Available: http://ieeexplore.ieee.org/document/5368902/

Howley, T. and Madden, M. G., "The Genetic Kernel Support Vector Machine: Description and Evaluation," *Artificial Intelligence Review*, vol. 24, no. 3-4, pp. 379–395, Nov. 2005. [Online]. Available: http://link.springer.com/10.1007/s10462-005-9009-3

Kelley, L., J Shrimpton, P., Muggleton, S., and J E Sternberg, M., "Discovering rules for protein-ligand specificity using support vector inductive logic programming," *Protein engineering, design & selection : PEDS*, vol. 22, pp. 561–7, 08 2009.

King, R. D., Srinivasan, A., and Sternberg, M. J. E., "Relating chemical activity to structure: An examination of ilp successes," *New Generation Computing*, vol. 14, no. 1, pp. 109–109, Mar 1996. [Online]. Available: https://doi.org/10.1007/BF03037220

Laird, P. D., *Learning from Good and Bad Data.* Boston, MA: Kluwer Academic Publishers, 1988.

Landwehr, N., Kersting, K., and Raedt, L. D., "Integrating naïve bayes and foil," *J. Mach. Learn. Res.*, vol. 8, pp. 481–507, Dec. 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1314498.1314516

Landwehr, N., Passerini, A., De Raedt, L., and Frasconi, P., "kfoil: Learning simple relational kernels," in *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, ser. AAAI'06. AAAI Press, 2006, pp. 389–394. [Online]. Available: http://dl.acm.org/citation.cfm?id=1597538.1597601

Landwehr, N., Passerini, A., De Raedt, L., and Frasconi, P., "Fast learning of relational kernels," *Machine Learning*, vol. 78, no. 3, pp. 305–342, Mar. 2010. [Online]. Available: http://link.springer.com/10.1007/s10994-009-5163-1

Lodhi, H. and Muggleton, S., "Modelling Metabolic Pathways Using Stochastic Logic Programs-Based Ensemble Methods," in *Computational Methods in Systems Biology*, Danos, V. and Schachter, V., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, vol. 3082, pp. 119–133. [Online]. Available: http://link.springer.com/10.1007/978-3-540-25974-9_10

Man Leung Wong and Kwong Sak Leung, "Combining genetic programming and inductive logic programming using logic grammars," in *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, vol. 2. Perth, WA, Australia: IEEE, 1995, pp. 733–736. [Online]. Available: http://ieeexplore.ieee.org/document/487476/

Man Leung Wong and Kwong Sak Leung, "Inducing logic programs with genetic algorithms: the Genetic Logic Programming System," *IEEE Expert*, vol. 10, no. 5, pp. 68–76, Oct. 1995. [Online]. Available: http://ieeexplore.ieee.org/document/464935/

Maznitsa, I., Aug 2019, java based Edinburgh Prolog parser. [Online]. Available: https://github.com/raydac/java-prolog-parser

Muggleton, S., mutagenesis data. [Online]. Available: https://www.doc.ic.ac.uk/~shm/mutagenesis.html#progol

Muggleton, S., alzheimer's data. [Online]. Available: https://www.doc.ic.ac.uk/~shm/alzheimers.html#kingetal

Muggleton, S., "Inductive logic programming," *New Gen. Comput.*, vol. 8, no. 4, pp. 295–318, Feb. 1991. [Online]. Available: http://dx.doi.org/10.1007/BF03037089

Muggleton, S., "Bayesian Inductive Logic Programming," *Oxford University Computing Laboratory*, p. 10, 1994.

Muggleton, S., "Inductive logic programming: Derivations, successes and shortcomings," *SIGART Bull.*, vol. 5, no. 1, pp. 5–11, Jan 1994. [Online]. Available: http://doi.acm.org/10.1145/181668.181671

Muggleton, S., Lodhi, H., Amini, A., and Sternberg, M. J. E., "Support vector inductive logic programming," in *Proceedings of the 8th International Conference on Discovery Science*, ser. DS'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 163–175. [Online]. Available: http://dx.doi.org/10.1007/11563983_15

Muggleton, S. and Tamaddoni-Nezhad, A., "QG/GA: a stochastic search for Progol," *Machine Learning*, vol. 70, no. 2-3, pp. 121–133, Mar. 2008. [Online]. Available: http://link.springer.com/10.1007/s10994-007-5029-3

Muggleton, S. H., Lin, D., Chen, J., and Tamaddoni-Nezhad, A., "MetaBayes: Bayesian Meta-Interpretative Learning Using Higher-Order Stochastic Refinement," in *Inductive Logic Programming*, Zaverucha, G., Santos Costa, V., and Paes, A., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, vol. 8812, pp. 1–17. [Online]. Available: http://link.springer.com/10.1007/978-3-662-44923-3_1

Nienhuys-Cheng, S.-H. and Wolf, R. d., *Foundations of Inductive Logic Programming*, Siekmann, J. and Carbonell, J. G., Eds. Berlin, Heidelberg: Springer-Verlag, 1997.

Nohejl, A., "Grammar-based genetic programming," Master's thesis, Charles University in Prague, 2011.

Ott, B., Aug 2019, k folds data. [Online]. Available: https://github.com/benott-cs/ILPData

Ott, B., Aug 2019, cKTAGA Code. [Online]. Available: https://github.com/benott-cs/CKTAGA

Passerini, A., Frasconi, P., and Raedt, L. D., "Kernels on prolog proof trees: Statistical learning in the ilp setting," *J. Mach. Learn. Res.*, vol. 7, pp. 307–342, Dec. 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1248547.1248558

Perolini, A., "Genetic algorithms and kernel matrix-based criteria combined approach to perform feature and model selection for support vector machines," *World Academy of Science, Engineering and Technology*, vol. 64, pp. 85–94, 01 2010.

Qomariyah, N. and Kazakov, D., "Learning from ordinal data with inductive logic programming in description logic," in *Late Breaking Papers of the 27th International Conference on Inductive Logic Programming*, Lachiche, N. and Vrain, C., Eds., vol. 2085. CEUR Workshop Proceedings, 3 2018, pp. 38–50, an earlier version of this paper was accepted for publication in 2017 and entered into PURE. This extended version of the paper was subject to another round of reviews, and was published as an open access paper in these online proceedings on 29 March 2018 (see link above in this record).

Quinlan, J. R., "Learning logical definitions from relations," *Mach. Learn.*, vol. 5, no. 3, pp. 239–266, Sept. 1990. [Online]. Available: https://doi.org/10.1023/A:1022699322624

Quinlan, J. R. and Cameron-Jones, R. M., "FOIL: A midterm report," in *Machine Learning: ECML-93*, Siekmann, J., Goos, G., Hartmanis, J., and Brazdil, P. B., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, vol. 667, pp. 1–20. [Online]. Available: http://link.springer.com/10.1007/3-540-56602-3_124

Quinlan, J. R. and Cameron-Jones, R. M., "Induction of logic programs: FOIL and related systems," *New Generation Computing*, vol. 13, no. 3-4, pp. 287–312, Dec. 1995. [Online]. Available: http://link.springer.com/10.1007/BF03037228

Rouveirol, C., "Flattening and saturation: Two representation changes for generalization," *Machine Learning*, vol. 14, pp. 219–232, Feb 1994. [Online]. Available: https://doi.org/10.1023/A:1022678217288

Rouveirol, C. and Puget, J. F., *Proceedings of the fourth European Working Session on Learning*. Pitman, 1989, pp. 201–211.

Ruckert, U. and Kramer, S., "Margin-based first-order rule learning," *Machine Learning*, vol. 70, no. 2-3, pp. 189–206, Mar. 2008. [Online]. Available: http://link.springer.com/10.1007/s10994-007-5034-6

Scholkopf, B. and Smola, A. J., *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, USA: MIT Press, 2001.

Shawe-Taylor, J. and Cristianini, N., *Kernel Methods for Pattern Analysis.* New York, NY, USA: Cambridge University Press, 2004.

Shutske, G. M., Pierrat, F. A., Kapples, K. J., Cornfeldt, M. L., Szewczak, M. R., Huger, F. P., Bores, G. M., Haroutunian, V., and Davis, K. L., "9-amino-1,2,3,4-tetrahydroacridin-1-ols. synthesis and evaluation as potential alzheimer's disease therapeutics," *Journal of Medicinal Chemistry*, vol. 32, no. 8, pp. 1805–1813, 1989.

Srinivasan, A., Aug 2019, online manual. [Online]. Available: http://www.cs.ox.ac.uk/activities/programinduction/Aleph/aleph.html

Srinivasan, A., Muggleton, S. H., Sternberg, M. J. E., and King, R. D., "Theories for mutagenicity: A study in first-order and feature-based induction," *Artif. Intell.*, vol. 85, no. 1-2, pp. 277–299, Aug. 1996. [Online]. Available: http://dx.doi.org/10.1016/0004-3702(95)00122-0

Tamaddoni Nezhad, A., "Logic-based machine learning using a bounded hypothesis space: the lattice structure, refinement operators and a genetic algorithm approach," Ph.D. dissertation, Imperial College London, 2013.

Tamaddoni-Nezhad, A. and Muggleton, S., "The lattice structure and refinement operators for the hypothesis space bounded by a bottom clause," *Machine Learning*, vol. 76, no. 1, pp. 37–72, July 2009. [Online]. Available: http://link.springer.com/10.1007/s10994-009-5117-7

Tamaddoni-Nezhad, A. and Muggleton, S. H., "Searching the Subsumption Lattice by a Genetic Algorithm," in *Inductive Logic Programming*, Goos, G., Hartmanis, J., van Leeuwen, J., Cussens, J., and Frisch, A., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, vol. 1866, pp. 243–252. [Online]. Available: http://link.springer.com/10.1007/3-540-44960-4_15

Verbaeten, S. and Van Assche, A., "Ensemble Methods for Noise Elimination in Classification Problems," in *Multiple Classifier Systems*, Goos, G., Hartmanis, J., van Leeuwen, J., Windeatt, T., and Roli, F., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, vol. 2709, pp. 317–325. [Online]. Available: http://link.springer.com/10.1007/3-540-44938-8_32

Yalabik, I., Yarman-Vural, F. T., Ucoluk, G., and Sehitoglu, O. T., "A pattern classification approach for boosting with genetic algorithms," in *2007 22nd international symposium on computer and information sciences.* Ankara, Turkey: IEEE, Nov. 2007, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/4456870/

Yang, X., Song, Q., and Wang, Y., "A weighted support vector machine for data classification," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 21, no. 05, pp. 961–976, Aug. 2007. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218001407005703