ALBERT-LUDWIG-UNIVERSITÄT FREIBURG

Faculty of Engineering
Department of Computer Science

Euclidean Embedding of Co-Proven Queries

MASTER THESIS

Hannes Schulz

Advisor : Dr. Andreas Karwath
Second Advisor : Prof. Dr. Martin Riedmiller

Submitted on : May 19, 2009

Acknowledgements

At this point I would like to thank all those who have supported me during the time of writing my thesis.

I am especially grateful to Dr. Kristian Kersting. This thesis – and I – greatly profited from his inexhaustible pool of ideas, technical knowledge and stimulating discussions.

Many thanks to Dr. Andreas Karwath, whose door was always open for my doubtful questions and for his time spent on introducing me to some of more the technical details of ILP.

I would like to thank Amir Globerson, the author of the CODE algorithm, for providing his source code.

I also thank Prof. Dr. Martin Riedmiller for making this thesis possible.

Finally, I would like to thank my parents, whose support I could count on throughout my studies.

Abstract

Relational data is complex. Inductive logic programming (ILP) algorithms, which can deal with this complexity, typically use large quantities of logical formulæ as input and output. A basic step of learning becomes difficult for the untrained user: Understanding the data. Visual representations on the other hand make it easy to shift from one perspective to another while exploring and analyzing data. In this thesis, we describe the first method for embedding the contents of a relational database and the rules governing the data into a single, common Euclidean space based on their co-proven statistics. The embedding is designed to place objects which are often co-proven close to each other, while unrelated objects are placed far from one another. We analyze the properties of the embeddings and demonstrate our method on real-world datasets, showing that ILP results can be intuitively represented and indeed be captured at a glance.

Deutsche Zusammenfassung

Relationale Daten sind sehr komplex. Inductive logic programming (ILP)-Algorithmen können mit dieser Komplexität umgehen. Als Eingabe benutzen, als Ausgabe erzeugen diese Algorithmen jedoch große Mengen an logischen Formeln, sodass ein entscheidender Schritt beim Lernen erschwert wird: Das Verstehen der Daten. Visuelle Repräsentationen, auf der anderen Seite, machen es dem Benutzer leicht während der Exploration und Analyse der Daten von einer Perspektive zur anderen zu wechseln. In dieser Arbeit beschreiben wir die erste uns bekannte Methode um die Inhalte einer relationalen Datenbank und die sie bestimmenden logischen Regeln in einen gemeinsamen euklidischen Raum einzubetten. Dabei ist die Einbettung so gewählt, dass sie die Distanzen der eingebetteten Objekte klein ist wenn diese oft miteinander bewiesen werden konnten. Wir analysieren die Eigenschaften unserer Methode und zeigen Einbettungen anwendungsnaher Datenbanken. Unsere Experimente zeigen dass ILP-Resultate tatsächlich intuitiv repräsentiert und auf einen Blick erfasst werden können.

Contents

List of Figures

1 Introduction

Large databases containing complex relational information were built over the last years. The numerous examples include molecules, social networks and the semantic web. However, data in its vast amount is meaningless and consequently, there is a large interest in algorithms that break it down to meaningful pieces for common tasks such as pattern mining, classification or regression. These tasks are significantly complicated by properties of relational databases: The entities in a relational database can take part in an arbitrary number of relations and additional information about them can be derived using so-called views. Furthermore, evident in the relations, the entities are not independent and thus violate common machine learning assumptions. Feature vectors, commonly used in machine learning, can therefore in general not be used as an equivalent representation (Džeroski, 2003). This also means that we cannot make use of the many algorithms which depend on feature vectors.

The relatively young discipline of inductive logic programming (ILP, Muggleton 1992; Lavrac and Džeroski 1994) has found a way to deal with relational information, essentially by fully keeping the complexity of the data in a logical representation. ILP then combines techniques from machine learning and logic programming to search the space of logical formulæ (queries) for solutions. These rules can abstract from the data, that is, they may contain variables.

This flexibility comes at a cost. Firstly, learning algorithms, even when simply generalizations of feature vector based algorithms (e. g. Van Laer and De Raedt, 2001) become more complex. Secondly, the solutions found are hard to interpret. Consider, for example the well-known Mutagenesis database (Srinivasan et al., 1996), containing 188 organic molecules. A naïve search for frequent patterns, that is, patterns that occur in more than 12 entities (molecules) up to a fixed complexity threshold with the WARMR algorithm resulted in 16 million syntactically different patterns. The vast majority of these patterns probably deal with common properties of organic molecules. Examples could be carbon atoms connected by single- and double bonds with hydrogen and oxygen. Mostly, these are not very surprising. Nevertheless, they can give insights into the structure of the data contained in the database. For a classification task, the user faces similar problems: The rules found by an ILP algorithm can be quite large and because they cannot be overlooked as a whole, classification performance is the main check for their correctness. We conclude that there is a gap between what ILP algorithms can do on multi-relational data and our ability to comprehend the results.

In this work, we propose the – to the best of our knowledge – first visualization for relational datasets and queries, tailored to provide the human user with insights into the structure of the data at hand. We use a two-step approach. In a first step, we use standard ILP algorithms to discover interesting patterns in a multi-relational database. During the second step we place both, the patterns and the entities in our database, into a common Euclidean space and optimize the distances such that co-occurring (i. e. co-proven) patterns appear close to each other, and entities in the database are placed close to the patterns they co-occur with. We compare various embedding algorithms as to how good the distances in the embedding reflect the co-occurrence relationship and find that the CODE algorithm of Globerson et al. (2007) yields the best results. In the resulting image we can further visualize properties of the embedded entities in a natural way. Finally, our method can be extended to interactive data exploration.

Part of this work is going to be published as "ILP, the Blind, and the Elephant: Euclidean Embedding of Co-Proven Queries" (Schulz, Kersting, Karwath, 2009) in the proceedings of the ILP conference.

The remainder of this work is organized as follows. In Section 2 we define important ILP concepts. We further provide an overview of distance measures in ILP and important embedding methods. We derive specific instances of embedding algorithms for relational data in Section 3, which we then compare on standard datasets in Section 4. Section 5 relates our method to the broader field of visual analytics and interactive tools in the domain of genetics which may benefit from our embedding method. We conclude in Section 6.

2 Preliminaries

Before introducing our embedding algorithm, we look into some important notions and algorithms. This section is divided into three parts. We first introduce ILP and define basic ILP concepts which we want to embed later. Embedding algorithms usually try to retain a distance measure. Therefore, some choices for measure distances between logical formulæ as used in ILP are provided in Section 2.2. With distances available, we have the choice of a wide variety of algorithms for dimensionality reduction and embedding, some of which we consider in Section 2.3.

2.1 Inductive Logic Programming

Inductive Logic Programming (ILP) is a scientific discipline at the intersection of machine learning and logic programming. The term was introduced by Stephen Muggleton (Muggleton and King, 1991; Muggleton, 1992). ILP strives to explain a set of observations by a theory and some background knowledge. Inputs and outputs are specified using a more or less restricted subset of First Order Logic (FOL).

ILP inherits its choice of FOL from logic programming. Its advocates claim that FOL is expressive and intuitive. It is expressive, since very complex concepts including variables can be described (Dantsin et al., 2001). FOL rules are said to be intuitive because their semantics easily map to natural language sentences (while the other direction is harder, e. g. Fuchs et al., 1999).

Clausal Theories and Interpretations As unconstrained FOL is infeasible to deal with, one restricts the language used to describe concepts for the ILP algorithms. We start from the syntax of FOL and define the basic constructs for the *learning from interpretations* setting as defined by De Raedt and Džeroski (1994), thereby fixing the vocabulary for the remaining chapters. To illustrate the definitions, we use examples derived from the Mutagenesis dataset.

An *alphabet* is a set of symbols for constants, functors and predicates.

Example 1: Constant. *The elements* `c`, `h`, `cl` *are constants.*

Example 2: Predicate. `atom` *and* `bond` *are predicate symbols.*

A *term* t is either a constant, a variable or a compound term $f(t_1, \ldots, t_n)$ composed of an n-ary function symbol f and n terms t_i. An *atom* is a logical formula of the form $p(t_1, \ldots, t_n)$, where p is an n-ary predicate symbol and t_i are terms.

Example 3: Term.

(i) $atom(K, I, E, W)$ *is a term with variables* K, I, E *and* W, *where* K *is a unique key for the molecule,* I *is a atom-id unique in* K, E *is an element constant and* W *is the weight of the atom.*

(ii) $atom(2, d2\_1, c, 22)$ *is a variable-free term*

(iii) $bond(K, E1, E2, T)$ *is a term, where* K *is a unique key for the molecule,* $E1$ *and* $E2$ *are atom-ids of elements in* K *and* T *designates the bond type, e.ġ. single or double.*

A *literal* is an atom or the negation $\neg A$ of an atom A. Atoms are positive literals; negated atoms are negative literals. A *clausal theory* (the FOL generalization of the conjunctive normal form, or CNF) has the form

$$\bigwedge_{V_1^1,\ldots,V_{v_1}^1} (l_1^1 \vee \cdots \vee l_{n_1}^1) \wedge \cdots \wedge \bigwedge_{V_1^k,\ldots,V_{v_k}^k} (l_1^k \vee \cdots \vee l_{n_k}^k),$$

where l_j^i are literals and $V_1^i, \ldots, V_{v_i}^i$ are all variables occurring in $l_1^i \vee \cdots \vee l_{n_i}^i$. Generally, ILP is restricted to clausal theories or subsets thereof (e.g. Horn clauses contain at most one positive literal in each clause; queries do not have positive literals); we will therefore limit our discussion to this subset of FOL.

Example 4: Clausal Theory.
$active(K) \leftarrow atom(K, E1, c, W1), bond(K, E1, E2, T), atom(K, E2, cl, W2)$ *is a (Horn) clausal theory. The variables (capital letters) are implicitly all-quantified. The body (right of* $\leftarrow$*) is a query.*

We define a (Herbrand) *interpretation* as a set of *ground* (i. e. variable free) atoms. For interpretations, the closed-world assumption holds: All atoms in the interpretation are true, all atoms not in the interpretation are false.

Example 5: Interpretation.
$atom(1, d1\_1, c, 22)$, $atom(1, d1\_2, cl, 23)$, $bond(1, d1\_1, d1\_2, single)$ *is an interpretation.*

A *substitution* $\theta = \{V_1 \leftarrow t_1, \ldots, V_n \leftarrow t_n\}$ is an assignment of terms $t_1, \ldots, t_n$ to variables $V_1, \ldots, V_n$. The formula $F\theta$, where F is a term, atom, literal or expression and $\theta = \{V_1 \leftarrow t_1, \ldots, V_n \leftarrow t_n\}$ is a substitution, is the formula obtained by simultaneously replacing all variables $V_1, \ldots, V_n$ in F by the terms $t_1, \ldots, t_n$.

We can now define what it means that a clausal theory or *query* T is true in an interpretation I,

$$T \text{ true in } I \Longleftrightarrow \bigwedge_{\theta} (T\theta \text{ is ground} \rightarrow T\theta \text{ true in } I). \tag{1}$$

A ground clausal theory T is true in an interpretation I if and only if each clause of T is true in I; ground clauses are true if one of the positive (negative) atoms in the clause is true (false) according to I (De Raedt, 1997).

Example 6: Substitution.
The substitution $\theta = \{K \leftarrow 1, E1 \leftarrow d1\_1, E2 \leftarrow d1\_2, W1 \leftarrow 22, W2 \leftarrow 23, T \leftarrow single\}$ *applied to the body of Example 4 is true in Example 5.*

With the main concepts defined, we now have a look at two common tasks of ILP: pattern mining and concept learning.

Discovery of Interesting Patterns A common task for ILP algorithms is to find interesting patterns in relational data. Dehaspe and Toivonen (1999) define the task as follows: Given a database $\mathbf{r}$, a class $\mathcal{L}$ of sentences (patterns) and a selection predicate q, the task is to find the theory of $\mathbf{r}$ with respect to $\mathcal{L}$ and q, i. e. the set $Th(\mathcal{L}, \mathbf{r}, q) = \{Q \in \mathcal{L} | q(\mathbf{r}, Q) \text{ is true}\}$. For frequent pattern mining, the selection predicate $q(\mathbf{r}, Q)$ is true if and only if the frequency of the pattern Q exceeds the frequency threshold in database $\mathbf{r}$. For relational databases, the APRIORI algorithm (Agrawal and Srikant, 1994) can be used to find frequent patterns: Starting from very general patterns, it specializes the ones which are frequent. ILP-techniques extend this basic idea such that more general relational data can be analyzed and patterns with complex structure can be discovered.

Using the terms defined above, we set $\mathbf{r}$ to be a set of interpretations and $\mathcal{L}$ to be a set of clausal theories (queries). While the set of interpretations is fixed, ILP systems typically take a declarative language bias as parameter, reducing the number of candidates for $\mathcal{L}$ (Adé et al., 1995). Frequent pattern miners such as WARMR (Dehaspe and De Raedt, 1997; Dehaspe and Toivonen, 1999) then spend much effort on further restricting the search by eliminating logical redundancy. For example, it is crucial not to specialize queries which are instances of a query already determined to be infrequent.

Concept Learning Another typical task for ILP is the inductive learning of a logical concept definition (Džeroski, 2007). Consider, for example, a database of family relations in the form of the interpretations, such as

$father(\text{john}, \text{peter})$	(John is the father of Peter) and
$mother(\text{mary}, \text{peter})$	(Mary is the mother of Peter).

An ILP learner could now induce the concept of

$parent(X, Y) \leftarrow mother(X, Y).$	(Some X is parent of Y if X is
$parent(X, Y) \leftarrow father(X, Y).$	either mother or father of Y)

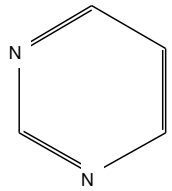
simply from labelled examples from the database, e. g. $parent(\text{john}, \text{peter})$ (labelled positive) or $parent(\text{john}, \text{mary})$ (labelled negative). The general definition (in contrast to the original examples it contains variables) of $parent(\cdot, \cdot)$ above can come in handy as *background knowledge* for learning the definition of $grandparent(X, Y)$:

$$grandparent(X, Y) \leftarrow parent(X, Z) \wedge parent(Z, Y).$$

More formally stated, the task of an ILP learner is to construct a theory $\mathcal{H}$ using positive examples in the set $\mathcal{P}$, negative examples in $\mathcal{N}$ and background knowledge in $\mathcal{B}$ such that

$$\bigwedge_{e \in \mathcal{P}} (\mathcal{B} \wedge \mathcal{H} \models e) \qquad \text{and} \tag{2}$$

$$\bigwedge_{e \in \mathcal{N}} (\mathcal{B} \wedge \mathcal{H} \not\models e)\,. \tag{3}$$



$atom(\mathrm{n}, 1)$, $atom(\mathrm{c}, 2)$, $atom(\mathrm{c}, 3)$, $atom(\mathrm{c}, 4)$,
$atom(\mathrm{n}, 5)$, $atom(\mathrm{c}, 6)$, $bond(1, 2, 2)$, $bond(2, 3, 1)$,
$bond(3, 4, 2)$, $bond(4, 5, 1)$, $bond(5, 6, 2)$, $bond(6, 1, 1)$

Figure 1: The chemical compound *pyrimidine*: Left: Structural formula, right: possible logical representation without hydrogen. Although logic is rich and expressive, the picture says more than the 12 terms.

If Equation (2) holds, $\mathcal{H}$ is said to be complete, if Equation (3) holds, $\mathcal{H}$ is said to be consistent.

We briefly discussed two applications of ILP, both of which yield a set of theories about the data: Frequent pattern mining yields a set of frequent patterns, concept learning possible definitions of a concept. Taking one step back, we have to relate to the following basic questions:

(i) How can we understand the data so that we can exploit it optimally?
(ii) How can we reach an understanding of theories produced by ILP algorithms?

We will now address both issues separately.

Understanding the Data. Even considering the way first order logic maps to natural language, large logic-based databases are inaccessible to the human understanding. Consider, for example, the common case of molecular databases, which typically contain molecules described by the elements they consist of and the atomic bonds between them. Even the description of a single molecule is hard to read (consider Figure 1), insights into relationships of objects in the database are even harder to determine. Without insights into the structure of the database, however, the choice of learning algorithm and parameters remains largely heuristic.

Understanding Theories About Data. In contrast to for example deductive reasoning, in ILP the theory is inferred from background knowledge and examples by induction. Therefore it is not necessarily correct; in fact, there may be infinitely many possible theories explaining a given set of observations. An ILP learner consequently has to restrict itself. The restrictions constrain the language used for constructing the theory (language bias) and the selection of the theory. The latter bias could be realized by Occam's razor (Blumer et al., 1987) or (more technically) Plotkin's Relative Least General Generalization (Plotkin, 1971).

A "good" theory must be carefully chosen to balance between how much of the data is explained (coverage, composed of relaxed conditions in Equations (2) and (3)) and how complicated its description is (simplicity). The rationale here is that simpler explanations tend to have better generalization properties and are therefore better fit for prediction purposes. However, for non-trivial datasets there is no obvious answer as to where to draw this line.

After learning, the rules learned by ILP algorithms leave important questions unanswered or at least hard to access:

(i) Which examples are covered?
(ii) Is the theory overly specific or general?
(iii) How does it relate to other possible theories?

This work addresses the problems of understanding data and theories from a novel perspective, by providing a generic procedure of visualizing relational datasets with respect to the determined queries and theories. We propose to use spatial concepts (i. e. distance) to intuitively represent the inherent structure. The next section considers ways to use the structure of queries and the database to define distance measures.

2.2 Distances in ILP

In this section, we will give a short overview on distance functions for logical theories (Section 2.1), the objects we want to embed. An in-depth discussion of distance functions in logics can be found in Ramon (2002).

In the ILP domain, distance functions serve an important purpose: They relieve us from the burden of discrete objects and thereby enable us to use algorithms designed for real-valued attributes: Close objects are more related than objects far from each other. Once distances between objects are available,

- instance based learning (Emde and Wettschereck, 1996) can be employed to predict properties of an unknown query object from similar, known objects;
- clustering algorithms (Jain and Dubes, 1988) can identify groups of objects which are similar within the group, but dissimilar between groups; and
- a dimensionality reduction technique such as the ones described in Section 2.3 can be employed to embed the objects into a low-dimensional space for visualization;

Distance Measures for Sequences We shall look for symmetric functions which are real-valued, positive and reflexive[1]. The simplest case is if the terms we are comparing are in fact sequences from a finite set of constants, e. g. letters or amino acids (Karlin and Ghandour, 1985). We can then define a distance function $d(\cdot,\cdot)$ specifying a cost for changing from one constant to another, as well as for removal and insertion of constants. We then determine the distance between the two sequences s and t with length $|s| = n$ and $|t| = m$ using some variant of the string edit distance (e.g. Wagner and Fischer, 1974):

$$\begin{aligned} \operatorname{dist}(s,t) &= D_{n,m}, \qquad \text{where} \\ D_{0,0} &= 0 \qquad\quad \text{and} \\ \bigwedge_{i\in\{1\ldots n\}} \bigwedge_{j\in\{1\ldots m\}} D_{i,j} &= \min \begin{cases} D_{i-1,j-1} & +0 & \text{(match)} \\ D_{i-1,j-1} & +d(s_i,t_j) & \text{(replacement)} \\ D_{i,j-1} & +d(\varepsilon,t_j) & \text{(insertion)} \\ D_{i-1,j} & +d(s_i,\varepsilon) & \text{(deletion)} \end{cases} \end{aligned}$$

[1] For a reflexive function $f : D \times D \mapsto \mathbb{R}$, we require that $\bigwedge_{x\in D} f(x,x) = 0$.

Distance Measures for Sets If we loosen the sequence constraint, that is, we allow sets A and B of constants instead of sequences s and t, we can use symmetric difference distance (Ramon, 2002), for example

$$\begin{aligned}\text{dist}(A,B) &= |(A \setminus B) \cup (B \setminus A)| \qquad \text{or, normalized,}\\ \text{dist}(A,B) &= \frac{|(A \setminus B) \cup (B \setminus A)|}{|A \cup B|}.\end{aligned}$$

If we again want to incorporate knowledge about distances $d(\cdot,\cdot)$ between constants into our distance function, we can use the Hausdorff metric defined as

$$\text{dist}_H(A,B) = \max\left\{\sup_{a\in A}\inf_{b\in B} d(a,b),\ \sup_{b\in B}\inf_{a\in A} d(a,b)\right\}, \tag{4}$$

which is dominated by the largest minimal distance between members of the two sets and ignores all smaller distances.

Distance Measures for Terms and Interpretations We can generalize even further, by allowing ground terms in addition to constants. One way to do this is by measuring the distance of both terms to their least general generalization (Hutchinson, 1997). Another, wide-spread distance function for terms is the Nienhuys-Cheng distance (Nienhuys-Cheng, 1997), inductively defined over the constants $\mathcal{T}$ and the predicates $\mathcal{F}$ as

$$\begin{aligned}\bigwedge_{t\in\mathcal{T}} \text{dist}_{nc}(t,t) &= 0\\ \bigwedge_{p/n\in\mathcal{F}}\bigwedge_{q/m\in\mathcal{F}} \text{dist}_{nc}(p(s_1,\ldots,s_n),q(t_1,\ldots,t_m)) &= 1\\ \bigwedge_{p/n\in\mathcal{F}} \text{dist}_{nc}(p(s_1,\ldots,s_n),p(t_1,\ldots,t_n)) &= \frac{1}{2n}\sum_{i=1}^{n}\text{dist}_{nc}(s_i,t_i).\end{aligned}$$

It is easy to see that the Nienhuys-Cheng distance can be combined with the Hausdorff metric to yield a distance function for (ground) interpretations. We solely have to substitute $\text{dist}_{nc}(\cdot,\cdot)$ for $d(\cdot,\cdot)$ in Equation (4). Finally, the methods introduced can be generalized such that they can also deal with terms containing variables (Ramon, 2002). Thereby, the distance of any two terms can be computed.

Non-Syntactic Distance Measures The distance measures described so far have one property in common: They are purely syntactically defined. That is, given only two terms, we can determine their distance. While this type of distance measure has its applications, we want to point out that it is not necessarily a good choice for our purpose, namely understanding empirically collected items using queries which match them. To expand on this, we observe that for a given data-set, we can have the following relations:

- Two queries q and p are equivalent with respect to the data E, that is

$$\bigwedge_{e\in E}(q(e) \leftrightarrow p(e)). \tag{5}$$

Empirically, there is no difference between p and q, so their syntactic difference can tell us little about the dataset we are analyzing. We might, however, prefer one of the queries, e. g. the one with smaller description length, as an explanation for the data in which they are true and can ignore the other.

- Two queries q and p are not related with respect to the data

$$\bigwedge_{e \in E} ((q(e) \to \neg p(e)) \wedge (p(e) \to \neg q(e))). \tag{6}$$

Again, the syntactic structure of p and q is not relevant, i. e. it is not necessarily the reason for their exclusivity. While it could be that $p \equiv \neg q$, it is also possible that we observe a property of the dataset, not of logical formulæ.

- Two queries q and p are somewhat related with respect to the data

$$\bigwedge_{e \in E} (q(e) \to p(e)), \tag{7}$$

$$\bigwedge_{e \in E} (p(e) \to q(e)), \qquad \text{or simply} \tag{8}$$

$$\bigvee_{e \in E} (q(e) \wedge p(e)). \tag{9}$$

The observations described in Equations (5) to (9) naturally lead to a distance measure for queries p and q with respect to a dataset E:

$$\mathrm{dist}_E(p, q) = \left|\{e | e \in E \wedge p(e) \wedge q(e)\}\right| \tag{10}$$

When normalized, the $\mathrm{dist}_E(\cdot, \cdot)$ corresponds to the empirical co-occurrence probability

$$\bar{p}(p, q) = \mathrm{dist}_E(p, q) / \sum_p \sum_q \mathrm{dist}_E(p, q). \tag{11}$$

A distance measure alone is not enough for a low-dimensional representation. Specifically, the distances described above may turn out to be contradictory, or the distances can only be retained when the objects are placed in a high-dimensional space. These problems are addressed by algorithms for dimensionality reduction and Euclidean embedding, which we consider in the next section.

2.3 Dimensionality Reduction and Euclidean Embedding

Machine learning often deals with complex objects described by a possibly large set of features. Assuming that there is some structure in the data, it may be worth looking at a low-dimensional manifold within the high-dimensional data:

(i) By choosing a favorable projection, interesting properties of the data can be emphasized.

(ii) A low-dimensional representation lacks irrelevant features (dimensions) which impede understanding and are an obstacle for many machine learning algorithms.

(iii) Low-dimensional spaces can be visualized.

Numerous unsupervised dimensionality reduction algorithms have been proposed (e. g. Fodor, 2002). In this section, we will consider some milestones and describe their properties.

Linear Dimensionality Reduction Techniques The most common algorithms principal component analysis (PCA, Jolliffe 2002) and independent component analysis (ICA, Hyvrinen 1999). Both methods depend on the examples being in some Euclidean space in the first place. In PCA, first an orthogonal coordinate system is determined such that the components (axis) of the examples are decorrelated. The solution to this problem is given in closed form by the singular value decomposition (Golub and Kahan, 1965) of the unbiased matrix X containing the original data points

$$X = W\Sigma V^T.$$

In a final step, components with less importance can be discarded by considering only the first n columns of W. The low-dimensional representation of X is then given by

$$Y = W_n X.$$

PCA is globally optimal, whereas most of the other techniques discussed here only result in a local optimum. ICA strives to identify components which are independent of each other (in the statistical sense) and therefore has to rely on higher order objective functions e. g. mutual information or kurtosis. The projection is reached by gradient descent on the objective function. For ICA, the number of target components has to be chosen in advance.

Non-Linear Dimensionality Reduction Techniques Both PCA and ICA in their original form yield a linear projection of the data to the low-dimensional space. The field of non-linear dimensionality projection has many degrees of freedom. At the most constrained end of the spectrum, non-linear PCA (Karhunen, 2001) applies a fixed, non-linear function $g(\cdot)$ during the projection. The choice of the function is arbitrary; its gradient is used to find a locally optimal solution

$$W^* = \arg\max_W E\left\{\|x - Wg(W^T x)\|^2\right\}.$$

Multidimensional Scaling (MDS, Cox and Cox 2001) is arguably the most general non-linear technique. It strives to retain a given dissimilarity measure δ_{ij} between objects i and j as distance d_{ij} after the projection of i and j to a low-dimensional space. Commonly, this is done by minimizing a stress function such as

$$S = \sqrt{\left(\frac{\sum_i \sum_j (d_{ij} - \delta_{ij})^2}{\sum_i \sum_j d_{ij}}\right)}$$

using an approach in which first, the positions of the embedded entities and second, the axis scales are adjusted iteratively (originally proposed by Takane et al., 1977).

Focusing on Local Structure to Extract Low-Dimensional Manifolds The techniques enumerated above cannot, however, deal with the case that the high-dimensional space contains a low-dimensional manifold as in the classic swiss-roll dataset depicted in Figure 2,

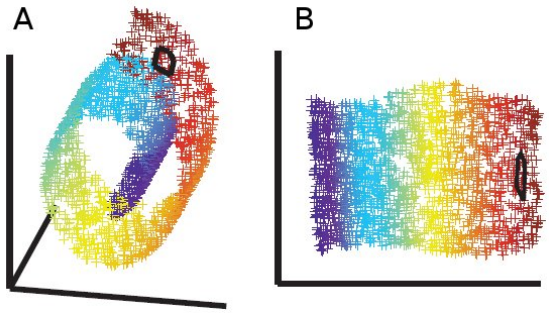


Figure 2: Visualization of swiss roll dataset (Roweis and Saul, 2000). (A) A low-dimensional manifold is embedded in a high-dimensional space. (B) The manifold extracted using Locally Linear Embedding (LLE).

since the objective function usually incorporates terms relating all data points to all other data points. Locally Linear Embedding (LLE, Roweis and Saul 2000) and IsoMap (Tenenbaum et al., 2000), for example, were developed such that the close-range relationships dominate the embedding. In Figure 2, the LLE algorithm avoids the fallacy of embedding the data points depicted in red next to the bright blue ones because it only considers the k nearest neighbours of each data point to determine the parameters for its embedding. The algorithm proposed by Roweis and Saul (2000) is globally optimal and also applicable to the case where only a dissimilarity measure is available. Similar results can be achieved using IsoMap, which first constructs a locally connected graph on the high-dimensional dataset, then calculates distances on the graph using Dijkstra's algorithm (Dijkstra, 1959) and finally performs MDS. Even though the use of MDS allows only for local optima, there is evidence that IsoMap outperforms LLE even on such artificial datasets as the swiss roll (Friedrich, 2002).

Embedding as Graph We can also view our dataset as a weighted graph with stronger or weaker ties between the contained items. We can now try to embed this graph to a two-dimensional space. A large number of works deals with the 2D-layout of planar graphs (Di Battista et al., 1994), and heuristic algorithms exist for non-planar graphs (e.g. Davidson and Harel, 1996; Herman and Marshall, 2000). However, there is a catch in this analogy, since not-connected nodes can be placed quite close to each other in the layout. Furthermore, many real-world graphs seem to have "small-world" properties: The average minimal path length is very short and there are tightly connected clusters (Travers and Milgram, 1969; Newman, 2003). Graph layout algorithms that try to find good paths for edges are prone to fail.

A method developed for the embedding of small-world graphs, especially in the context of social networks, is the Linlog algorithm (Noack, 2004, 2007). In contrast to energy model based embeddings, Linlog does not try to enforce uniform short path lengths, thereby placing nodes with high degree in the center. Instead, Linlog tries to identify subgraphs using the normalized cut (Gomory and Hu, 1961) and then minimizes within-cluster distances and maximizes between-cluster distances. Distances between similar clusters are retained such that they reflect the size of the cut between them.

Another well-known general purpose graph embedding method is spectral embedding (Brand and Huang, 2003). In contrast to LINLOG, no clusters in the graph are assumed. The common procedure is to do an eigenanalysis of the weighted adjacency matrix W or its Laplacian L, where

$$L_{ij} = \begin{cases} \sum_k W_{ik} & \text{if } i = j \\ -W_{ij} & \text{otherwise.} \end{cases}$$

We can then use the eigenvectors associated with the n smallest non-zero eigenvalues as the n components of the coordinates in the embedding. This embedded representation has various interesting properties useful e. g. for graph clustering (Belkin and Niyogi, 2002) and graph comparison (Luo et al., 2003). However, the interpretation of the embeddings is difficult. As Brand and Huang (2003) derived, the coordinates represent angles between nodes on a hypersphere; we will see that the resulting embeddings are inherently structured, but not intuitive.

Co-Occurrence Data Embedding In Section 2.1 we already alluded to the structure of the data we would like to embed for visualization. To recapitulate, we are given interpretations and queries, which do not easily map to a low-dimensional Euclidean space, let alone a common one. In Section 2.2 we defined an empirically grounded distance measure for logical queries: The co-occurrence probability (Equation (11)). We can extend this notion to distances between interpretations and queries. More formally, we know that

(i) the query q is true in the set of interpretations I_q; and
(ii) in the interpretation i, the queries Q_i are true.

In addition to Equation (11), we can derive a second dissimilarity measure from these observations, namely $\text{dist}_{qi}(\cdot, \cdot)$, which should be small if query q co-occurs with interpretation i. Given that we only have dissimilarity measures, our choice of algorithms seems to be limited to versions of MDS or LLE. We can do better than these general purpose algorithms, however, by considering how the distances were constructed and following an appropriate gradient, as demonstrated in recent publications (Globerson et al., 2007; Iwata et al., 2007). In fact, as the authors demonstrated, this optimization problem can be formulated as a convex problem and then optimally solved using semidefinite programming (SDP, e. g. Boyd and Vandenberghe 2004).

In their algorithm "Euclidean Embedding of Co-Occurrence Data" (CODE), Globerson et al. derive their similarity measure from statistics extracted from the data, such as joint or conditional probabilities of objects co-occurring. The embedding is therefore based on the empirical estimate of the joint probability distribution $p(x, y)$ of two random variables X and Y. The authors derive a least squares gradient descend algorithm which optimizes random initial positions of objects representing the values of X and Y, such that their relations in space respect the empirical co-occurrence measure. Additionally, the authors introduce extensions to add further embedding constraints, such as co-occurrence-statistics within a random variable or embedding based on the co-occurrence statistics of more than two variables. Their very general approach thus has many specific instances. In Section 3 we will focus on and derive a few instances for our application.

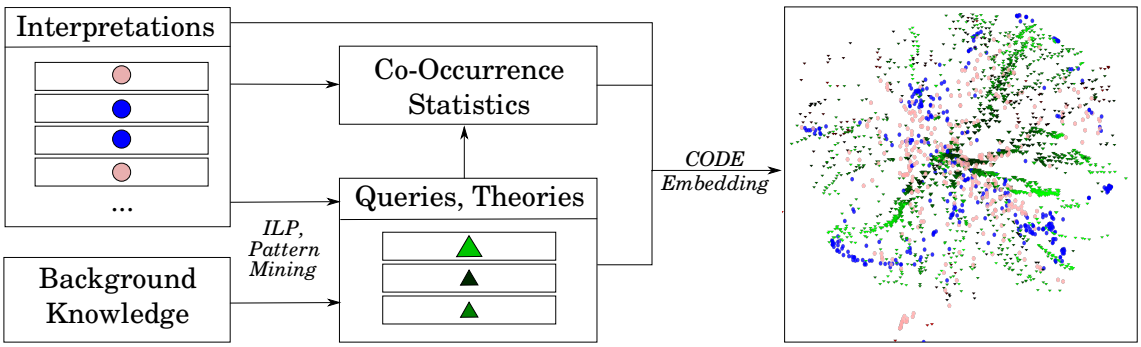


Figure 3: Schematic Overview. First, interpretations and background knowledge are used by the ILP algorithm or pattern miner, generating frequent patterns and queries. Their co-occurrence with the interpretations is used to embed both, interpretations and queries, into a common Euclidean space, where additional properties can be visualized.

3 Euclidean Embedding of Relational Data and Queries

With co-occurrence as distance measure, we can derive instances of the embedding algorithms discussed in the previous section. We first consider the CODE algorithm, as it is specifically tailored to retain co-occurrence relationships as distances. In Section 3.1, we construct a distance measure for the interaction between queries and interpretations and derive a gradient to optimize the embedding. We improve the speed of convergence of the original CODE implementation by choosing RPROP to minimize the objective function as described in Section 3.2. In Sections 3.3 and 3.4 we then briefly discuss how the LINLOG algorithm, PCA and spectral embedding can be used to embed relational data.

3.1 Constructing the Interaction-Model

In this section, we motivate a model which models the co-occurrence of queries with interpretations as Euclidean distance. We will loosely follow Globerson et al. (2007). We assume that we are given the co-occurrence matrix C, where $C_{xy} = 1$ if query x is true in interpretation y. From C we can construct the joint probability distribution $p(x, y)$ which describes the probability of observing a query x in interpretation y

$$p(x, y) = C_{xy} / \sum_i \sum_j C_{ij}.$$

The marginals of the distribution, $p(x) = \sum_y p(x, y)$ and $p(y) = \sum_x p(x, y)$, reflect the probability of query x being true and interpretation y occurring, respectively. Note that it does not make much sense to speak of the probability of an interpretation y occurring, since this quantity is created artificially by the number of queries x which are true in y. Queries which occur in many interpretations seem to be interesting, however. To summarize:

(i) We are interested in how a query occurs absolutely; and
(ii) We are not interested in how many queries an interpretation interacts with.

We should therefore make our embedding sensitive to the marginal of the queries $p(x)$, but insensitive to the marginal of an interpretation $p(y)$.

To relate the distance and statistical dependency, we assume that there are embedding functions $\Phi : Q \mapsto \mathbb{R}^n$ and $\Psi : I \mapsto \mathbb{R}^n$ mapping queries and interpretations to vectors, respectively. The term d_{xy} then denotes the distance between the embeddings

$$d_{xy} = \|\Phi(x) - \Psi(y)\| .$$

Following Globerson et al. (2007), we start with the requirement that

$$\frac{p(x,y)}{p(y)} \propto e^{-d_{xy}^2}$$

and therefore

$$p_{UM}(x,y) \equiv \frac{1}{Z} p(y) e^{-d_{xy}^2},$$

where $Z = \sum_x \sum_y p(y) e^{-d_{xy}^2}$ ensures that $p(x,y)$ is a probability distribution. The subscript UM is derived from the model classification scheme introduced in Globerson et al. (2007): A U, in contrast to M, is used for variables whose the marginal does not occur in the model, the order reflects the order of parameters of $p(\cdot,\cdot)$.

The quality of our embedding functions Φ and Ψ can be measured with the log-likelihood of the model, given by

$$l(\Phi,\Psi) = \sum_{x,y} p(x,y) \log p_{UM}(x,y).$$

We will use a gradient ascend method to maximize the log-likelihood of the embedding. For this purpose we derive the gradient of $l(\cdot,\cdot)$:

$$\begin{aligned} l(\Phi,\Psi) = & \sum_{x,y} p(x,y) \log p_{UM}(x,y) \qquad (12) \\ = & \sum_{x,y} p(x,y) \left(-d_{xy}^2 - \log Z + \log p(y) \right) \\ = & -\sum_{x,y} p(x,y)\, d_{xy}^2 - \log Z + const, \\ = & -\sum_{x,y} p(x,y)\, d_{xy}^2 - \log \sum_{x,y} p(y) e^{-d_{xy}^2} + const, \end{aligned}$$

where $const = \sum_{x,y} \log p(y)$ is a term which does not depend on d_{xy} and therefore not on the parameters Φ and Ψ. We can now determine the partial derivatives with respect to $\Phi(x)$ and $\Psi(y)$. First, note that

$$\nabla d_{xy}^2 = 2 \left(\Phi(x) - \Psi(y) \right).$$

Function `RCODE` – calculate embedding of interpretations and queries

Input: Joint probability table $p \in \mathbb{R}^{|Q|\times|I|}$
Input: Dimension of Embedding n
Output: Embeddings $\Phi : Q \mapsto \mathbb{R}^n$, $\Psi :\mapsto \mathbb{R}^n$

$l_{best} \leftarrow -\infty$
for $nrestart \leftarrow 1$ **to** $nrestart_{max}$ **do**
  initialize $\Phi_1 \in \mathbb{R}^{|Q|\times n}, \Psi_1 \in \mathbb{R}^{|I|\times n}$ randomly
  for $t \leftarrow 1$ **to** t_{max} **do**
    $l_t \leftarrow l(\Phi_t, \Psi_t)$ `/* determine log-likelihood */`
    $(G_t^\Phi, G_t^\Psi)^T \leftarrow \nabla l(\Phi_t, \Psi_t)$ `/* determine gradient */`
    if $t > 1$ **then**
      $\Delta\Phi_t \leftarrow$`Get_RPROP_Update(`G_t^Φ, G_{t-1}^Φ, l_t,l_{t-1}`)`
      $\Delta\Psi_t \leftarrow$`Get_RPROP_Update(`G_t^Ψ, G_{t-1}^Ψ, l_t,l_{t-1}`)`
      $\Phi_{t+1} \leftarrow \Phi_t + \Delta\Phi_t$
      $\Psi_{t+1} \leftarrow \Psi_t + \Delta\Psi_t$
  if $l_{best} < l_{t_{max}}$ **then**
    $l_{best} \leftarrow l_{t_{max}}$
    $(\Phi, \Psi) \leftarrow (\Phi_{t_{max}}, \Psi_{t_{max}})$

the derivative then becomes

$$\begin{aligned}
\frac{\partial l(\Phi,\Psi)}{\partial \Phi(x)} &= -\sum_{x,y} p(x,y) 2\left(\Phi(x) - \Psi(y)\right) - \frac{1}{Z}\sum p(y) e^{-d_{xy}^2}\left(-2\left(\Phi(x) - \Psi(y)\right)\right) \\
&= 2\left(-\sum_{x,y} p(x,y)\left(\Phi(x) - \Psi(y)\right) + \frac{1}{Z}\sum p(y) e^{-d_{xy}^2}\left(\Phi(x) - \Psi(y)\right)\right) \\
&= 2\left(-\sum_{x,y} p(x,y)\Phi(x) + \sum_{x,y} p(x,y)\Psi(y) + \sum_{x,y}\frac{1}{Z}p(y)e^{-d_{xy}^2}\Phi(x) - \sum_{x,y}\frac{1}{Z}p(y)e^{-d_{xy}^2}\Psi(y)\right) \\
&= 2\left(-\left\langle\Phi(x)\right\rangle_{p(x,y)} + \left\langle\Psi(y)\right\rangle_{p(x,y)} + \left\langle\Phi(x)\right\rangle_{p_{UM}(x,y)} - \left\langle\Psi(y)\right\rangle_{p_{UM}(x,y)}\right) \\
&= 2\left(\left\langle\Phi(x)\right\rangle_{p_{UM}(x,y)} - \left\langle\Phi(x)\right\rangle_{p(x,y)}\right) + 2\left(\left\langle\Psi(y)\right\rangle_{p(x,y)} - \left\langle\Psi(y)\right\rangle_{p_{UM}(x,y)}\right)
\end{aligned}$$

Here we use the generalized average notation $\langle f(x)\rangle_{p(x)} = \sum_x p(x) f(x)$. Analogously, we can derive

$$\begin{aligned}
\frac{\partial l(\Phi,\Psi)}{\partial \Psi(y)} &= -\sum_{x,y} p(x,y)\left(-2\left(\Phi(x) - \Psi(y)\right)\right) - \frac{1}{Z}\sum p(y) e^{-d_{xy}^2}\left(2\left(\Phi(x) - \Psi(y)\right)\right) \\
&= 2\left(\left\langle\Phi(x)\right\rangle_{p(x,y)} - \left\langle\Phi(x)\right\rangle_{p_{UM}(x,y)}\right) + 2\left(\left\langle\Psi(y)\right\rangle_{p_{UM}(x,y)} - \left\langle\Psi(y)\right\rangle_{p(x,y)}\right)
\end{aligned}$$

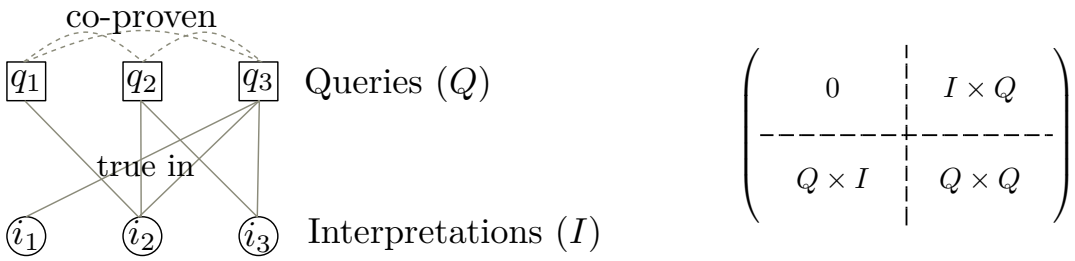


Figure 4: Data representation for LINLOG embedding. Left: The interpretations and queries can be seen as a bipartite graph. The graph can be extended using co-occurrence probabilities between queries. Right: The adjacency matrix of the resulting graph has four blocks with co-occurrence probabilities of queries (Q) and interpretations (I).

3.2 The RCODE Embedding Algorithm

Using the gradient from the previous section, we can use a gradient descent algorithm to maximize the log-likelihood of the embedding. Our algorithm is summarized on the preceding page.

We maximize the log-likelihood function iteratively (Line 4) using unconstrained optimization techniques with random restarts (Line 2) as suggested by Globerson et al. (2007). In their work[2], the authors used BFGS (e.g. Avriel, 2003) to update Φ and Ψ in each iteration. BFGS is a quasi-Newton method which tries to iteratively maximize a function using its gradient and a Hessian which is approximated using previously determined gradients. The step-size of BFGS is computed as a function of the gradient at the current position and then adjusted by a line search to a locally optimal value. Thus, for each iteration the function and its gradient have to be determined several times.

Since determining the gradient is the costly part of the embedding, we employ the IRPROP$^+$ (Igel and Husken, 2000) algorithm shown on the next page, a derivate of RPROP (Riedmiller and Braun, 1993). Instead of estimating a second order derivative, RPROP relies solely on the sign of the derivative and one learning-rate per dimension to find its step size. The rationale is: If the sign does not change (Lines 5 to 8), increase the learning rate and continue in the same direction; if the sign changes (Lines 2 to 4), proceed in the opposite direction using a smaller rate. RPROP has been shown to be very competitive and robust despite of its heuristic approach, not only for the artificial neural net application it was originally designed for (e.g. Schiffmann et al., 1993; Hannan and Bishop, 1997; Kretzschmar et al., 2008). Igel and Husken (2000) propose an extension to RPROP, called IRPROP. In its IRPROP$^+$ variant, they modify the original algorithm such that no update is performed when the sign of the gradient changed but the objective function did not worsen (Line 7), with the hope of jumping over some local minimum.

Using IRPROP$^+$, we can modify CODE (pseudo-code on the preceding page) to ensure that we have exactly one evaluation of the log-likelihood function (Line 5) and its gradient (Line 6) per iteration. Additionally to the time savings per iteration, this change also yields better convergence properties, as we will see in Section 4.5.1.

To distinguish our variant from the original CODE algorithm of Globerson et al., we refer to it as RCODE, Resilient Co-Occurrence Data Embedding.

[2]Many thanks to Amir Globerson for providing the source code for CODE.

Function `Get_RPROP_Update`

Input: Gradients G^t, G^{t-1}
Input: Objective function values l_t, l_{t-1}
Output: Parameter update ΔW^t

foreach *Component* G_{ij} *of* G **do**
  if $G_{ij}^t \cdot G_{ij}^{t-1} < 0$ **then**
    $\Delta_{ij}^t \leftarrow \min(\Delta_{ij}^{t-1} \cdot \eta^+, \Delta_{max})$
    $\Delta W_{ij}^t \leftarrow \operatorname{sign} G_{ij}^t \cdot \Delta_{ij}^t$
  else if $G_{ij}^t \cdot G_{ij}^{t-1} > 0$ **then**
    $\Delta_{ij}^t \leftarrow \max(\Delta_{ij}^{t-1} \cdot \eta^-, \Delta_{min})$
    if $l_t < l_{t-1}$ **then** $\Delta W_{ij}^t \leftarrow -\Delta W_{ij}^{t-1}$; `/* iRPROP variant only */`
    $G_{ij}^t \leftarrow 0$
  else if $G_{ij}^t \cdot G_{ij}^{t-1} = 0$ **then**
    $\Delta W_{ij}^t \leftarrow \operatorname{sign} G_{ij}^t \cdot \Delta_{ij}^t$

3.3 Linlog Embedding

We can also embed relational data using the Linlog algorithm described in Section 2.3. However, Linlog is designed to embed a single graph. A straight forward way to create a graph from the co-occurrence data is a bi-partite graph with interpretations forming one set of nodes, queries the other. The relation that query q is true in interpretation i is represented by an edge (q, i). As with CODE, this information is not sufficient to yield embeddings which represent the co-occurrence relationships in the data (Section 3.5.1). Therefore, we also insert edges between co-occuring queries and weight the edges according to the number of co-occurrences. The graph and the resulting adjacency matrix are visualized in Figure 4. Similar to CODE, we can scale the $Q \times Q$ block in the adjacency matrix to lower the importance of the inter-query co-occurrence probabilities.

After Linlog embedding, we proceed with post-processing as described in Section 3.5.3.

3.4 Spectral and PCA-based Embedding

To evaluate our method, we further consider spectral embedding and PCA-based embedding for relational data. This section briefly describes how to apply the techniques described in Section 2.3 in our setting.

PCA We perform PCA directly on the query-query co-occurrence matrix and derive the coordinates in the embedding from the first two principal components.

Spectral Embedding We take the coordinates from the eigenvectors of the query-query matrix associated with the smallest non-zero eigenvalues.

As the performance of both methods is not promising (see below), we do not consider embedding interpretations into the same space.

3.5 Extensions

The embeddings generated using the algorithms can be extended such that they provide more information to the user. In this section, we discuss how additional information can be incorporated into the embedding process itself (Sections 3.5.1 and 3.5.2) and the final image (Section 3.5.3). Finally, we consider how we can interactively explore the embeddings.

3.5.1 Using Co-Occurrence Probabilities of Co-Proven Queries

Using only the algorithm discussed in Section 3.2 does not yield good results. Specifically, for logical queries we have that the co-occurrence probabilities are basically binary up to normalization: Either the query is true in the interpretation, or it is not. As a result, all interpretations are placed approximately equally far away from all their associated queries. Such a "degenerate" picture is shown in figure Figure 5. While the image already contains a lot of the information we are interested in, we can force the embedding algorithm to put more emphasis onto the inter-query co-occurrence probabilities.

Assuming that co-occurrences between queries Q and interpretations I are stored in the binary matrix

$$C_{Q,I} \in \{0,1\}^{|Q|\times|I|},$$

we can calculate the query-query co-occurrence matrix to be

$$C_{Q,Q} = C_{Q,I}C_{Q,I}^T.$$

As suggested by Globerson et al. (2007), we add a second term to the log-likelihood function in Equation (12), such that

$$l(\Phi,\Psi) = \sum_{x,y} p(x,y)\log p_{UM}(x,y) + \eta \sum_{x^{(1)},x^{(2)}} p(x^{(1)},x^{(2)})\log p_{UU}(x^{(1)},x^{(2)}), \tag{13}$$

where η is a constant weighting the relation between both gradients. We can visually verify the effect of maximizing the combined log-likelihood function in Figure 9. In the following, when referring to RCODE, we will refer to this variation of the algorithm.

3.5.2 Embedding Interpretations Based on Query Entropy

As mentioned in Section 2.1, concept learning is a typical ILP task and can – for our purposes – be summarized as the search for queries which determine the concept of an interpretation. In our visualization, we can colorize queries which are good candidates. However, the embedding algorithm knows nothing about the varying importance of the queries and hence treats them equally. It follows that the position of interpretations in the embedding is dominated by the frequent patterns and not by the "important" ones.

We suggest to use a two-step approach to this problem. We first embed queries and interpretations by maximizing Equation (13). In a second step, we fix the positions of the queries and use RCODE to determine the position of the interpretations only with respect to selected "important" queries. For a classification task with N classes, we can use a low entropy

$$H(q) = -\sum_{c=1}^{N} p_{c,q}\log p_{c,q} \tag{14}$$

Figure 5: Left: Embedding of Mutagenesis dataset using only query-interpretation co-occurrence measures. Because the co-occurrence probability is binary for each pair, interpretations (circles) are embedded equally far away from all their queries (triangles). Right: Embedding using additional co-occurrence probabilities from co-proven queries.

as a selection criterion, where $p_{c,q}$ represents the empirical probability of the query q being true in an interpretation of class c.

When using a threshold to determine the "important" queries, some interpretations might be left with no queries at all. We can solve this problem by generalizing the threshold approach to a weighting approach: We multiply the query-interpretation co-occurrence matrix with the negative entropy of the respective queries and normalize such that the sum over all co-occurrences of an interpretation is one. When reapplying the RCODE algorithm with fixed query-positions, each interpretation is placed close to the queries which are most significant for the determination of its class. The difference between one-step and two-step embedding for the Mutagenesis dataset (Section 4.2) is depicted in Figure 6.

3.5.3 Finding Regional Representatives

The RCODE algorithm described in Section 3.2 yields positions for interpretations and queries. While the embeddings (i. e. Figures 5 and 6) already visualize the structure, the resulting pictures are not very meaningful. They lack information on what the structure represents. In principle, we could tag all embedded entities with meaningful descriptions, however, as the number of entities increases the amount of information becomes too much. Instead, we can pick out regional representatives that provide just enough information to allow the user to navigate and understand the dataset without resulting in information overload. This section describes an algorithm to automate the process of finding meaningful regional representatives for the embedded queries.

As a first step, we will focus on finding a selection criterion, that is, we define which queries we are interested in. The selection criterion should prefer queries which represent the area in which they are placed. We can assume that this is the case for queries which are true in many interpretations. Furthermore, using Occam's razor (Blumer et al., 1987) we can assume that queries with a shorter description length are more general than others. A possible selection criterion selcrit : $Q \mapsto \mathbb{R}$ for queries $q \in Q$ could therefore be defined

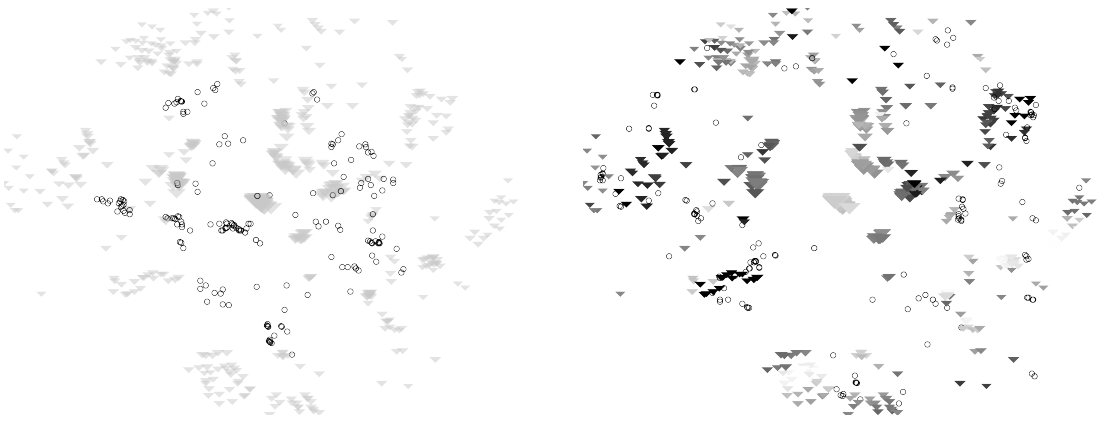


Figure 6: Left: Standard one-step embedding; Right: Two-step embedding. In the second step, the query (triangle) positions are fixed and the interpretations (circles) are embedded according to their co-occurrence with low-entropy (dark) queries.

as

$$\text{selcrit}_{freq}(q) := \text{frequency}(q) / \,\text{descriptionlength}(q)$$

In classification tasks, it could also be desirable to focus on queries which can distinguish well between classes, e. g. based on the well-known F-score:

$$\begin{aligned} F(q) &= 2(\text{precision}(q) \cdot \text{recall}(q))/(\text{precision}(q) + \text{recall}(q)) \\ \text{selcrit}_F(q) &:= F(q) / \,\text{descriptionlength}(q) \end{aligned}$$

We should note that the selection criterion will only be used to locally compare queries: Queries with a wholly different co-occurrence statistic do not compete for selection. To become indifferent to large-scale differences in the embedding, we can first use a technique from digital image processing: A high-pass filter. We subtract the local average value from the selection criterion, such that

$$\text{selcrit}'(q) := \text{selcrit}(q) - \frac{1}{|\,\text{neighbours}(q)|} \sum_{n \in \text{neighbours}(q)} \text{selcrit}(n),$$

where the neighbourhood of a query is defined in terms of a maximum distance in the embedding and must be chosen to select the desired density of labels in the final image.

After applying the high-pass filter, there may still be very similar queries in a close range. We use a local winner-takes-all rule to increase the value of the selection criterion of queries which are strongest in their area and inhibit the others. Formally, for all queries q, we determine

$$q^* = \operatorname*{arg\,max}_{n \in \text{neighbours}(q)} \left(\text{selcrit}'(n)\right)$$

and then set

$$\begin{aligned} \text{selcrit}'(q^*) &:= \alpha\,\text{selcrit}'(q) && \text{with } \alpha > 1, \text{ and} \\ \text{selcrit}'(n) &:= \frac{1}{\alpha}\,\text{selcrit}'(n) && \text{for } n \neq q^*. \end{aligned}$$

We iterate this process and then select the highest scoring queries for display.

3.5.4 Interactive Extensions

The 2D (and also 3D, for that matter) embeddings are ambiguous, as mentioned in Section 4.4: Two objects may be placed at the same position in the embedding although they have completely different co-occurrence statistics. As an example, consider the rectangle A, B, C, D with the corners representing queries. The interpretation q occurring with A and C will be placed in the center of the rectangle, just as the interpretation p occurring with B and D will be. To disambiguate such situations, we can use interactive data exploration. We implemented a tool which allows the user to point to interpretations and see the associated queries, and thereby determine whether clusters in the image are true clusters in the data.

Considering the other direction, some interpretations which are co-occurring with a query might be scattered because they co-occur with many other queries. To detect such cases, we can use a similar mechanism as above: Pointing to queries shows the associated interpretations.

Finally, we suggest to introduce group selection and zooming to the interactive tool. Group selection with a database backend could be used to determine the common features of the selected queries.

4 Experiments

To analyze our visualization method, we embed the datasets described in Section 4.2 using pattern miners briefly introduced in Section 4.1. Section 4.3 shows how we can pre-process the data by reducing the information gathered by the pattern miner. After embedding, we visualize additional properties of queries and interpretations as explained in Section 4.4.

Scalability and convergence of our approach are analyzed in Section 4.5.1. We then compare the our algorithm against various other embedding methods in Section 4.5.2 with regards to correlation of spatial distance with co-occurrence and, qualitatively, the readability of the embedding. Finally, Section 4.6 discusses selected embeddings in detail.

4.1 Pattern Miners

In this work, we use MOLFEA (Helma et al., 2002) and C-ARMR[3] (De Raedt and Ramon, 2004) to mine databases of molecules. Both systems are inspired by the APRIORI algorithm (Agrawal and Srikant, 1994): they construct general queries and specialize them only if they are frequent. Only queries more frequent than some threshold are retained and further expanded, i.e., specialized. While MOLFEA constructs chains of molecules (atom,

[3] in the CLASSIC'CL implementation (Stolle et al., 2005)

bond, atom, ...), C-ARMR constructs general queries and can take background knowledge into account as well.

4.2 Datasets

NIPS database The Neural Information Processing Systems database[4] contains data extracted via OCR from NIPS1-12, papers of the pre-electronic submission era. Amongst other statistics, the co-occurrence of 13,649 words in 1,740 submissions have been determined.

We simply set "interpretations" to the papers and the "queries" to words. Consequently we have the number of occurrences of a query in the interpretation as the respective entry in the co-occurrence matrix, when a word occurs in a paper. Part of this dataset was also embedded by Globerson et al. (2007) to demonstrate the capabilities of their CODE algorithm. In this work, we embed the complete dataset to demonstrate that RCODE works in principle, and that it can deal with large amounts of data.

Mutagenesis Dataset On Mutgenesis (Srinivasan et al., 1996), the problem is to predict the mutagenicity of a set of compounds. In our experiments, we use the atom and bond structure information only (including the predefined predicate like `ball3s`, `ring_size_5s`, and others). The dataset consists of two sets: a regression friendly set with 188 entries (125 positives, 63 negatives) and a regression unfriendly set with 42 entries (13 positives and 29 negatives).

Here, "interpretations" are all (regression friendly and unfriendly) molecules. We use C-ARMR (Section 4.1) to find queries with minimum frequency 15 and delta 5 up to depth 8, resulting in a set of 504 queries with unique co-occurrence statistics. Additionally, we record in how many interpretations of a class (active or inactive mutagenicity) each query was true.

Estrogen Dataset The estrogen database was extracted from the EPA's DSSTox NCTRER Database[5]. The original dataset was published by Fang et al. (2001), and is specially designed to evaluate QSAR approaches. The NCTRER database provides activity classifications for a total of 232 chemical compounds, which have been tested regarding their binding activities for the estrogen receptor. The database contains a diverse set of natural, synthetic, and environmental estrogens, and is considered to cover most known estrogenic classes spanning a wide range of biological activity (Fang et al., 2001).

The database distributed by the EPA's DSSTox is in SDF (Structure Data Format), and contains, in addition to the original database, a number of annotations: 6 indicator variables extracted from the original publication (Fang et al., 2001), logP (octanol/water partition coefficient) values, chemical class assignments (6 main classes, 20 subclasses), as well as the activity category ER-RBA (estrogen receptor relative binding affinity). This classification yields 131 active and 101 inactive compounds (with regard to their ER-RBA).

Again, "interpretations" are molecules. This time, however, we use MOLFEA (Section 4.1) to generate atom-bond chains as queries. We find all queries with minimum frequency 10, resulting in 482 queries with unique co-occurrence statistics.

[4] `http://www.cs.toronto.edu/~roweis/data.html`
[5] `http://www.epa.gov/ncct/dsstox/sdf_nctrer.html`

AIDS Dataset The DTP AIDS Antiviral Screening Database originating from the NCI's development therapeutics program NCI/NIH[6] consists of SMILES representations of 41,768 chemical compounds (Kramer et al., 2001). Each data entry is classified as either active, moderately active, or inactive. A total of 417 compounds are classified as active, 1,069 as moderately active, and 40,282 as inactive. We have converted this dataset into SDF format using the OpenBabel toolkit and randomly sampled 400 active and 400 moderate/inactive compounds.

As in the estrogen dataset, we use MOLFEA-generated atom-bond chains as queries to the interpretations (molecules). A minimum frequency of 15 and maximum length 30 results in 3310 queries with unique co-occurrence statistics.

4.3 Pre-Processing: Removing Redundant Queries

The number of possible frequent queries is huge even for small datasets, in fact, it becomes prohibitively large for known embedding algorithms very quickly. For example, the number of frequent queries with less than eight conjunctions as determined by C-ARMR in the 230 molecule-database "Mutagenesis" (Section 4.2) is at 16 million. One reason for this explosion is that when queries are constructed in a very general way, redundant queries cannot be distinguished from truly novel ones. Consider, again, the pyrimidine compound depicted in Figure 1. The query $pyrimidine(X)$ might match a certain number of molecules. However, a query looking for a unique substructure of pyrimidine, e. g. `N=C-C`, will get an equal number of results as the query for the whole molecule. Such redundancy is hard to detect syntactically, although C-ARMR tries to eliminate obvious cases, e. g. using δ-freeness and s-freeness concepts (Stolle et al., 2005).

Luckily it is not required to embed all queries, nor is it desirable for our method: In CODE, queries get assigned different positions when they are different with respect to the queries in which they are true. Consequently, we can reduce a set of redundant queries to one, embed it, and after the embedding provide the user with the list of associated, syntactically different queries.

The objective function in Equation (13) takes into account all queries, therefore the embedding changes when the redundant ones are removed. Consequently, we have to explain why we can assume that collapsing redundant queries to a single one does not change the embedding in an unfavorable way. The rationale is as follows: Assume that the query q_0 co-occurs with exactly the same interpretations as the queries $q_1, q_2, \cdots q_n$ for some large n. Let us further assume that the queries are frequent. Then, when embedding all queries including $q_0, \cdots, q_n$, the queries will have a major influence on the outcome. Now assume that we delete $q_1, \cdots q_n$. Since there were many syntactically different queries corresponding to q_0, some of them are prone to have specializations which are frequent as well and have similar, but not equal co-occurrence statistics. As a cluster, q_0 and the (semantically different) specializations of $q_0, \cdots, q_n$ will also have a large influence on the embedding. Therefore, we can assume that removing semantically equivalent queries (and thereby making the embedding possible in the first place) does not produce major distortions.

[6] `http://dtp.nci.nih.gov/`

4.4 Post-Processing: Visualizing Side-Information in Embedding

Given an embedding of queries and interpretations, we still have the options of – at least – shape, hue, brightness and size to make the embedding more understandable. The choice depends of course on the data available. We will briefly mention some of the options.

Features/Queries Both should be clearly distinguished; however, we can gain insights from just looking at one of the two entity sets. For example, we can see whether there are clusters of similar interpretations or just how the queries are related among each other.

Note however, that the embedding is ambiguous and clusters can be composed of queries with quite different co-occurrence statistics. For now, we can let the user find such cases using interactive exploration (Section 3.5.4).

Frequency The amount of interpretations a query applies to is valuable information. We use size to indicate the $\log(frequency)$. As we will see in Section 4.5.2, this somewhat disambiguates the problem of the non-transitivity of co-occurrence, which cannot be addressed correctly by the embedding algorithm.

Class In the AIDS and Mutagenesis dataset we have two classes of molecules. It is therefore interesting to visualize how molecules of different classes relate to different queries and how good they can be distinguished. We suggest to use different colors for this purpose. This naturally extends to a regression setting e. g. the case of the estrogen dataset, where activity is measured on a scale between 0 and 100.

Frequency/Class Specificity Measures We should also address the combination of the frequency and class, since this is what yields a "good" theory as defined in Section 2.1. In this work, we mostly use the brightness of a query to show how specific it is to a class and hue to depict the class. For example, in Figure 12, green queries correspond to many "active" molecules, red queries correspond to "inactive" ones. Dark queries are undecided, which yields an intuitive transition between classes. Other candidates for visualization are the F_2 measure or the χ^2 value of a query.

4.5 Analysis of Embeddings

4.5.1 Convergence and Scalability Properties of RCODE

To demonstrate the advantages of our modifications to CODE, we repeatedly embed Molfea-generated patterns from the AIDS dataset. We embed the 800 molecules together with the resulting 3310 queries using CODE and RCODE 10 times each. Both algorithms robustly converge (Figure 7), however, RCODE gives slightly better results.

We additionally compared execution times for both algorithms. CODE was available in a Matlab implementation using a highly optimized mex-function to calculate the gradient, RCODE was implemented using C++. The RCODE implementation was approximately three times faster per iteration, which we can attribute to the fact that it uses less evaluations of the gradient than the original BFGS-based solution.

To demonstrate the scalability of our approach, we embedded the $13,649$ words and $1,740$ papers from the NIPS database. The result is shown in Figure 8. We used the post-processing from Section 3.5.3 with frequency as the selection criterion to select words

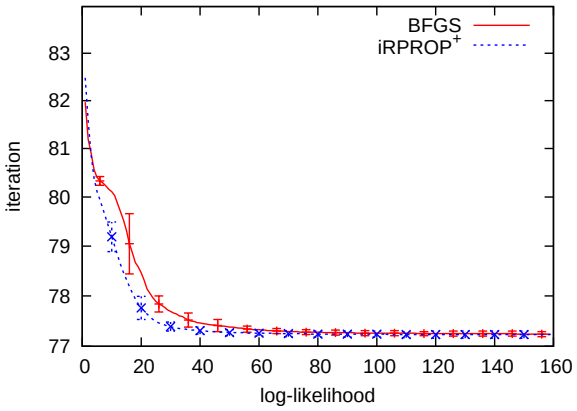


Figure 7: Performance of RCODE, our iRPROP$^+$ vs. the original BFGS implementation. We embedded 800 interpretations of the AIDS dataset and 3310 Molfea-generated queries ten times with random initial positions. For each iteration and algorithm, we averaged and plotted the value of the log-likelihood function. Error bars represent standard deviation. Note that embedding using RCODE took 9 minutes on average, while original implementation using BFGS took 29 minutes.

for display. From this automated selection alone, we can determine areas such as graphical models, scheduling, language processing and biologically inspired papers in the embedding.

4.5.2 Quality of Embedding

We will now analyze how faithful our embedding method is to the objective function and how it deals with conflicting information.

As a start, we will concentrate on how well inter-query relationships are represented in the embedding. For each pair of queries, we determine the co-occurrence and the distance in the embedding and histogram which co-occurrence-distance relations occur how often. The result is depicted in Figure 9. As expected, we find an inverse proportional relationship between the two variables. That is, the probability of being placed close to each other rises with the number of co-occurrences. However, queries which do not occur with each other can occur at almost any distance in the image. We can explain this observation by the intransitivity of co-occurrence: A query A might co-occur often with queries B and C, thus (A, B) and (A, C) are pairs which we would like to see close together. Although B and C could be unrelated, the embedding mechanism can be forced to place them close to each other. In the ILP setting, this happens quite often, when A is a general query and B and C are specializations which cover different interpretations. We can aid the user in detecting such situations by visualizing the frequency of a query (number of interpretations in which the queries are true), for example using color or size. In the figures we use the size as an indicator of frequency.

In Section 3.5 we already considered that using only query-interpretation co-occurrences induces undesirable embeddings, since all interpretations are placed at the same distance to "their" queries. To circumvent this problem, we introduced an additional component to the gradient which forced queries which often occur together to be close to each other. The (expected) effect on the distance between queries in the embedding depending on their co-occurrence is depicted in Figure 9: The plot based only on query-interpretation co-occurrence shows a degenerated inverse proportional relation, which is improved when query-query co-occurrence statistics are also taken into account.

We can further analyze other embedding methods. In Section 2.3 we suggested some candidate algorithms. Globerson et al. (2007) already compared the CODE algorithm to correspondence analysis, PCA, MDS and IsoMap. We chose Linlog (Section 3.3) PCA, and spectral embedding (Section 3.4) to evaluate their fitness for relational data,

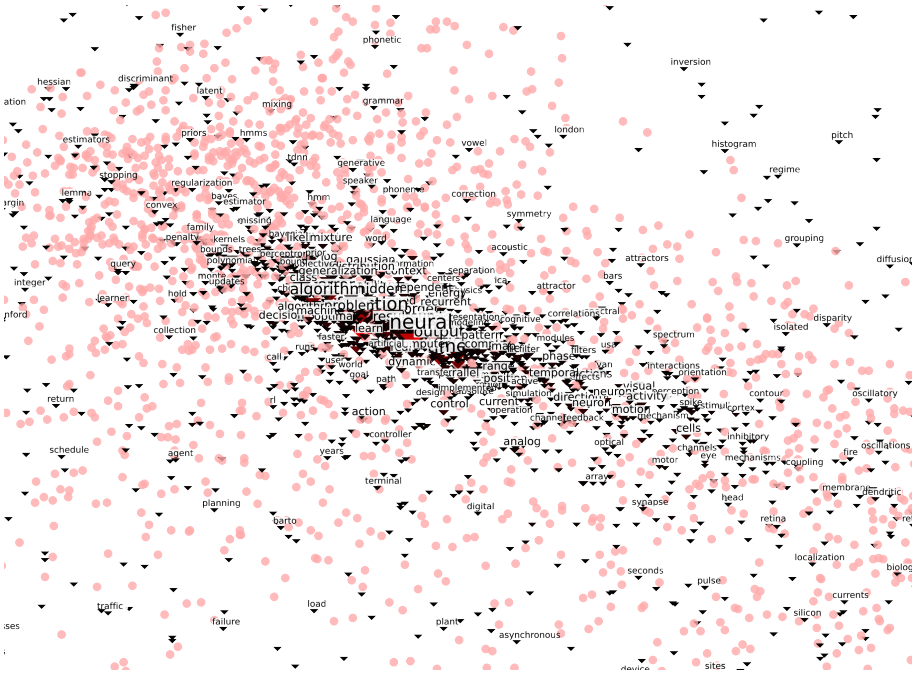


Figure 8: Embedding of 13,649 words (triangles) and 1,740 documents (circles) of the NIPS database. The common words are in the center, top left deals with graphical models, bottom left with scheduling. The top center area deals with language processing, while biologically inspired papers are in the bottom right corner. More papers seem to be published in the area of graphical models.

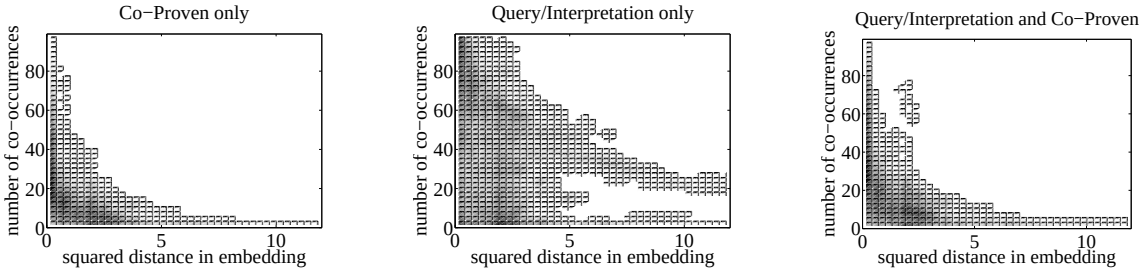


Figure 9: Comparison of squared distances of queries in the embedding to their co-occurrence probabilities. We used a 2D histogram of size 40×40 to visualize the observed relations in the Mutagenesis dataset. Left: using only co-proven query co-occurrence statistics (correlation $\rho = -0.5$); Center: using only query-interpretation co-occurrence statistics ($\rho = -0.47$); Right: Embedding using query-interpretation and co-proven query co-occurrence statistics ($\rho = -0.51$).

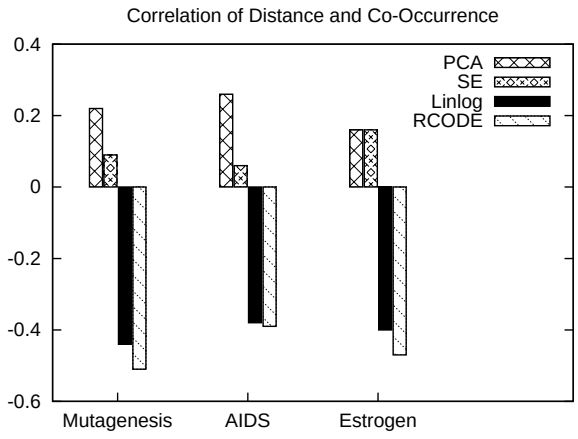


Figure 10: Comparison of correlations between distances of queries in the embedding and their number of co-occurrences. For LINLOG and RCODE we averaged the results from 10 random restarts.

specifically, how well the co-occurrence relation is represented as distances in the image and, qualitatively, how readable the resulting representation is. The embeddings are displayed in Figure 11, the correlations are compared in Figure 10.

PCA While some relations are retained – high-frequency queries are on the left, a top-down arch represents the more specific queries – the inherent structure is not represented very well. Specifically, distance-relations do not inversely correlate with co-occurrence ($\rho = 0.22$), which makes the plot hard to interpret.

Spectral Embedding The embedding of the AIDS dataset shows a typical embedding of a "power law" graph as described by Chung and Lu (2004), for which low-dimensional embeddings are nearly useless for visualization and clustering (Lang, 2005). As expected, the distance-relationships in the image are hard to interpret and do not correlate inversely with the co-occurrence ($\rho = 0.09$).

Linlog Embedding In contrast to the previous methods, LINLOG produces embeddings which exhibit the desired negative correlation between distance and co-occurrence ($\rho = -0.44$). The embedding of the Mutagenesis dataset reflects the clear clusters in the dataset very well, however, within-cluster relationships of queries are hard to see. In the AIDS dataset embedding, this effect becomes even more pronounced as clusters are hard to identify. The clear tree structure observable in CODE embeddings, which gives hints to specializations of queries, is hidden.

Provided that both PCA and spectral embedding do not provide a reading for spatial distance, embedding the interpretations into the same Euclidean space is of no value. For LINLOG it is useful and also has an interesting aspect: Interpretations are internally assigned to a cluster, which attracts them more than other clusters to which they have connections as well. As a result, part of the ambiguities of a mapping to the 2D plane are avoided or at least hidden. The cluster assignment can be modulated similar to the extensions proposed in Section 3.5 by assigning weights to the bi-partite graph in Figure 4.

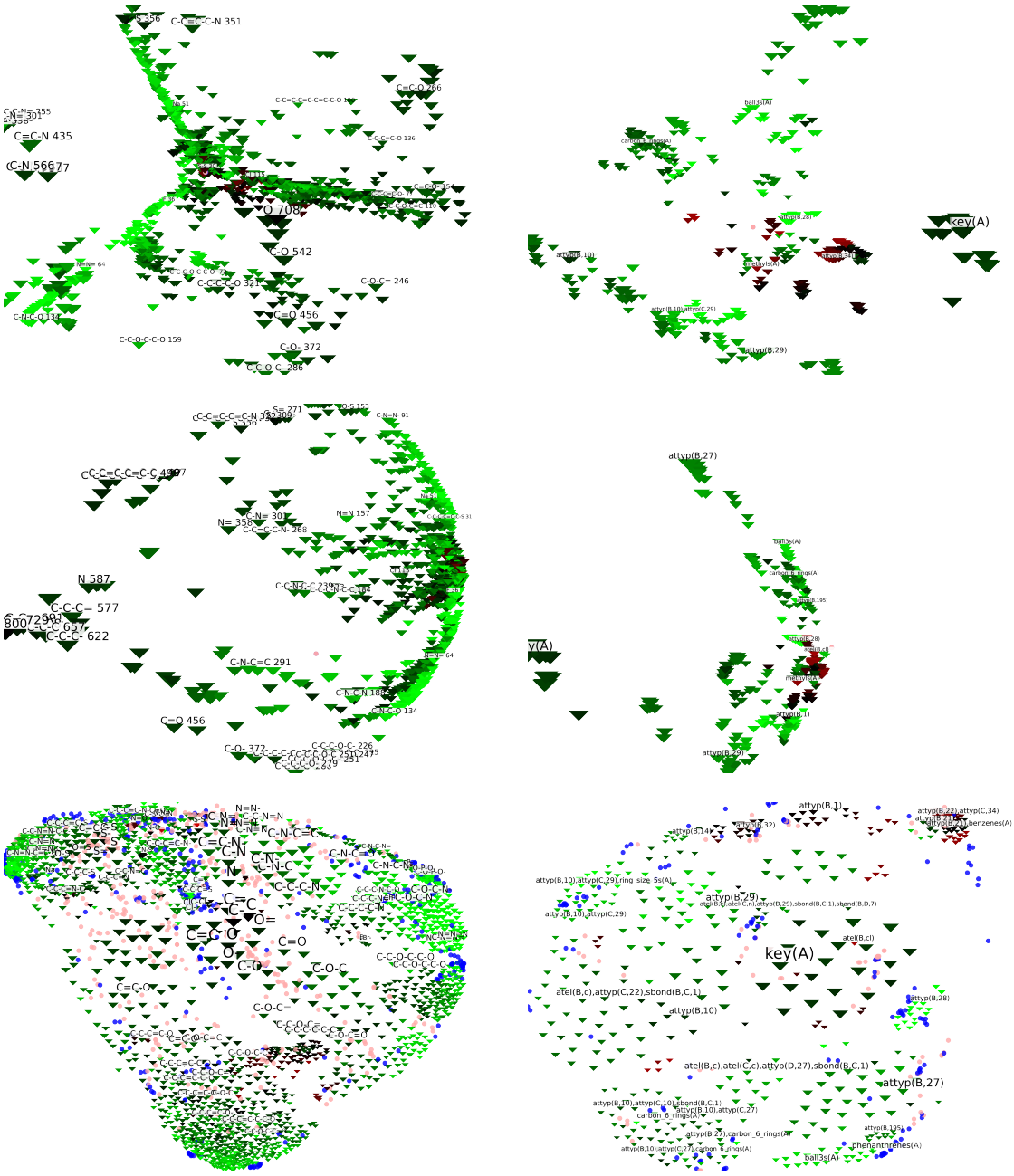


Figure 11: Embedding of AIDS (left) and Mutagenesis (right) dataset using non-CODE based algorithms. Top: Spectral embedding using eigenvectors associated with second and third largest eigenvalues of Query co-occurrence matrix. Center: PCA embedding using main two components of query-query co-occurrence matrix. Bottom: LINLOG embedding on extended bipartite graph.

4.6 Showcases

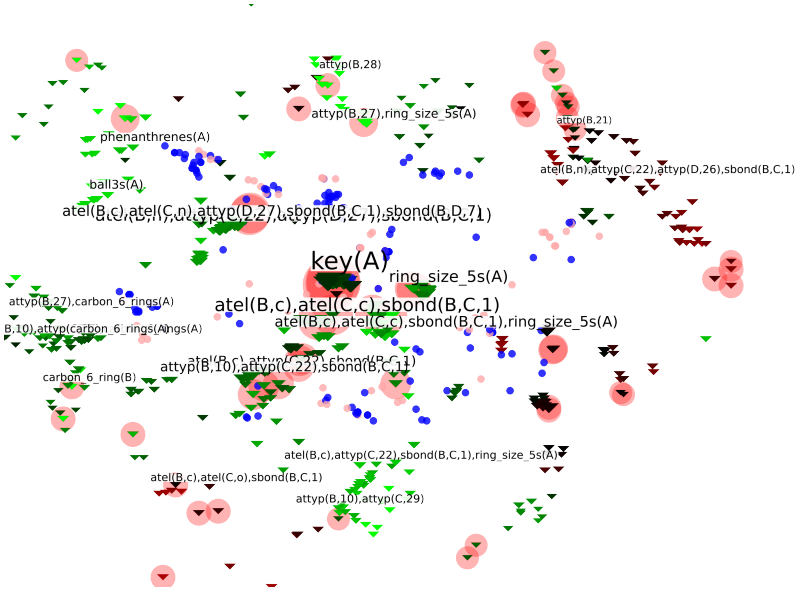


Figure 12: Mutagenesis Dataset: Embedded Queries (triangles) and interpretations (circles).

The Mutagenesis Dataset

- Molecules of the Mutagenesis dataset (circles) and C-ARMR-generated queries (triangles) are displayed. The color of the molecules indicates their class as active (blue) or inactive (salmon). Frequent queries appear larger in the image, queries which hint to active (inactive) molecules are displayed in green (red), while black queries do not exhibit a class preference.
- The key of a molecule says nothing about its properties and occurs in every molecule. Therefore, it is placed in the center.
- The queries with textual description reflect the logical program learned by ALEPH (Srinivasan, 2000) to identify active molecules:

```
active(A) :- attyp(A,B,195), phenanthrenes(A).
active(A) :- attyp(A,B,29), ring_size_5s(A).
active(A) :- ball3s(A).
active(A) :- attyp(A,B,21), attyp(A,C,38), sbond(A,C,B,1).
active(A) :- attyp(A,B,21), attyp(A,C,38), sbond(A,C,B,1).
active(A) :- attyp(A,B,195), attyp(A,C,10).
active(A) :- attyp(A,B,27), ring_size_5s(A).
active(A) :- attyp(A,B,29), attyp(A,C,27).
active(A) :- attyp(A,B,16), ring_size_5s(A).
```

- We further visualize which queries were considered by ALEPH to find this solution using red circles surrounding the queries.

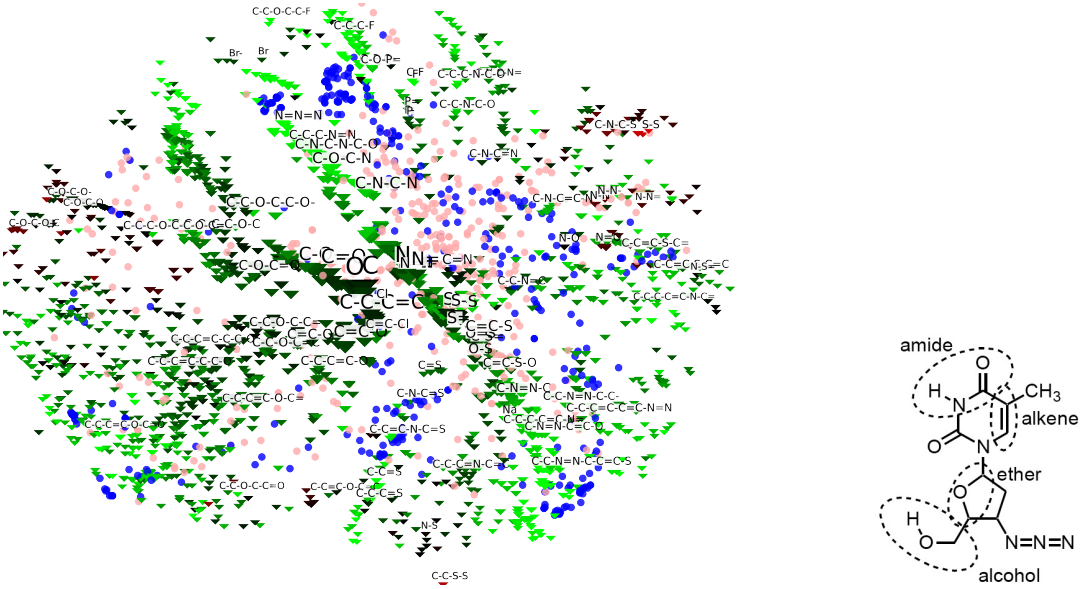


Figure 13: Left: AIDS Dataset, Embedded queries (triangles) and interpretations (circles). Right: Azidothymidine (AZT), a highly potent inhibitor of HIV-1 replication. AZT is prominent in the top of the embedding.

The AIDS Dataset

- As with all organic compounds, short sequences of carbon (`C`) and oxygen (`O`) are frequent in all molecules (placed in the center) and are not relevant to class information (black).
- The molecules are displayed in blue (active) or salmon (inactive).
- The embedding clearly indicates compounds in green color that are derivatives of Azidothymidine (AZT, right hand side), a potent inhibitor of HIV-1 replication. AZT contains a number of functional groups like an ether group (`C-O-C`) and a nitrogen group (`N=N=N`), one of the prominent features of AZT. These features, as well as the bridge between the two aromatic rings (`C-N-C-N-C-O`) can clearly be seen to be prominent in top of the embedding and attract many of the interpretations.
- At the bottom right, three groups of sulfur-related active queries can be distinguished, most likely from the group of dyes and polyanions[7].
- The information implicitly represented by the restricted chain form of queries could be made more explicit by using a graph miner and displaying the actual graphs instead of molecule chains. We are currently working on the realization of this idea.

[7] `http://dtp.nci.nih.gov/docs/aids/searches/list.html#DPA`

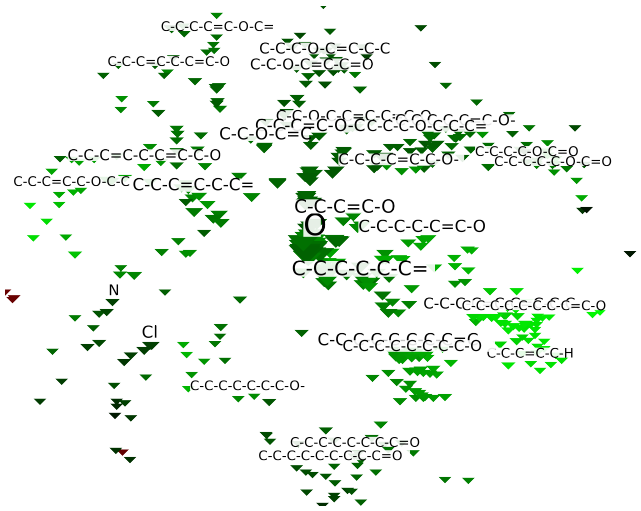


Figure 14: Embedding of Estrogen dataset. Here, black/colored queries indicate low/high activity.

Estrogen Dataset

- As with all organic compounds, carbon (`C`) and oxygen (`O`) are frequent in all molecules (placed in the center) and are not relevant to class information (black).
- Queries indicated in green color reflect a high average activity of the interpretations in which they are true.
- In their original publication Fang et al. have identified that a phenolic ring connected by one to three atoms to another benzene ring is one of the key features that have to be present regarding the likelihood of a compound being an ER ligand. A phenolic ring is a 6-carbon benzene ring with an attached hydroxyl (`OH`) group. In the embedding, it can be seen that this is reflected in features like `C-C-C=C-O`, which indicates that there is a path of one carbon atom to (a part of) a ring structure (`C-C=C`) connected to an oxygen.

  This implicit information could be made more explicit by using a graph miner and displaying the actual graphs instead of molecule chains. We are currently working on the realization of this idea.

5 Related Work

This work's contribution is a generic visualization method for relational data by embedding queries and interpretations into a common low-dimensional Euclidean space. To the authors knowledge, this is the first attempt at designing such an algorithm. We can, however, relate our method to the general field of information visualization, visual analytics and previous visualization methods for specific domains.

Visual analytics is a scientific field concerned with sensemaking and reasoning (Thomas, 2005). The aim is to provide interactive tools with visual feedback, thereby facilitating the process of understanding and dealing with large amounts of data. Visual analytics tools supply the human expert with tools that do the more mechanical tasks, while the expert guides the process with her intuition and experience, both of which are hard to put into machines. Our work addresses a crucial step in this process, namely information visualization. It provides an intuitive and generic visual representation of large databases.

Previous work in this area is highly domain specific, for example in the domain of bioinformatics, which by definition deals with large databases. In the genetic age, visualization centers around the domain of common properties in sequences, especially genomes (e. g. Schneider and Stephens, 1990; Cho et al., 1998; Schatz et al., 2007).

The bioinformatics-centered system Fusion presented by Indukuri (2004) comes closest to our approach in integrating ILP mechanism into an interactive application to visualize and explore data. Fusion employs an ILP data miner to discover frequent patterns in a database of genes. The genes can then be displayed in 2D scatterplots, where the axis reflect properties of the genes, with the genes matching selected queries marked. The user can change the axis of the scatterplots or select queries and thereby explore the interaction of measured properties and matching queries. Our method provides a more direct way of relating patterns and database with each other, since no measured axis are assumed and we use the co-occurrence information to find the distances of the queries in the plane instead of leaving it to the user to find an advantageous viewpoint.

Previous attempts at defining distance measures for logical data were largely motivated by the syntax of logical formulæ. The thesis of Ramon (2002) provides such measures, some of which are discussed in Section 2.2. Karwath and Kersting (2007) use syntax-based distances for sequence alignments and provide a visualization of the resulting alignments. In Karwath et al. (2008), the authors show how to discriminatively learn the parameters of a syntax-based distance measure to separate classes in a classification task.

Michaels et al. (1998) used mutual information to define a distance measure for genes. Their visualization method named Euclidean distance cluster analysis hierarchically determines clusters using this distance measure, places the entities in a plane and draws the clusters as connecting and branching lines. While the clustering could be incorporated in our approach as well (c. f. Section 3.3), the approach by Michaels et al. does not scale well and the connections between the genes are implicit, not embedded into the same space as the genes. The last point makes a data exploration as suggested in this work impossible.

Finally, for the distance measure used in this work we exploit a probabilistic model of co-occurrence of relational queries and interpretations. We can therefore place our visualization method in the larger context of statistical relational learning which develops techniques for representation, reasoning and learning in domains with complex relational and rich probabilistic structure (Getoor and Taskar, 2007).

6 Conclusion

In this work, we proposed a novel visualization method for relational datasets, tailored to provide the human user with insights into the structure of a dataset and the algorithms applied to it. The dataset is first represented as logical interpretations. We run pattern miners from the inductive logic programming (ILP) literature to find interesting patterns (logical queries). For the queries, we record in which interpretations they were true and with which other – co-proven – queries they occurred. We showed that co-occurrence statistics can be used as a similarity measure which is much more grounded in the data than commonly used syntax-based distance measures. Consequently, we applied an instance of the Co-Occurrence Data Embedding algorithm (CODE, Globerson et al. 2007) to embed queries and interpretations into a common Euclidean space. The resulting embeddings compare favorably with PCA, spectral embedding and Linlog in retaining the similarity relationships of the embedded entities as distances. The convergence speed of the unconstrained minimization could be greatly improved by choosing an RPROP (Riedmiller and Braun, 1993) variant instead of the originally proposed BFGS implementation.

We embedded data from various real-world molecular databases and implemented an automated tagging system. Further information can be provided to the user through coloring to represent underlying concepts such as class-membership and interactivity to disambiguate placement in the 2D embedding. The resulting images promise to provide insights into the database which were previously not possible (Barry Hardy, personal communication). Part of this work will be published in the proceedings of the ILP conference (Schulz, Kersting, Karwath, 2009).

To the authors knowledge, this work is the first attempt at a general visualization method for ILP. Therefore, various topics remain to be explored. The visualization could be improved by finding a better way to deal with non-transitivity of co-occurrence i. e. by relaxing the distance constraints between two specializations of the same query.

The visualization of ILP learning algorithms should be further evaluated. There are two aspects here in need of consideration: The visualization of the search procedure and the visualization of the search results. The search procedure typically proceeds sequentially and considers various subsets of the entities in the database, a process which could be reflected in the image. Search results can take various forms, such as order-dependent rules or trees. While queries constituting the search results can be embedded with our method, clarification as to which interpretations they apply to when executed in tree or rule order would greatly improve the usefulness of our approach. So far, our proposed method can only provide hints to properties determined theories.

In this work we concentrated on the visualization aspects of co-occurrence embeddings; however, low-dimensional embeddings could have uses in other domains as well. Low-dimensional representations are particularly useful for e. g. clustering and instance-based learning because they do not suffer from the curse of dimensionality. Our embeddings could thus be used as input for other applications. For this work we showed how to embed all available data at once. For the future, we also consider out-of-sample techniques, where novel data can quickly be placed in the embedding, increasing the number of use-cases of the clustering or instance-based learning techniques discussed above. Finally, the CODE algorithm is not limited to embedding entities of two types, in our case, interpretations and queries. For example, for classification tasks it is also possible to embed class labels into the same Euclidean space using their co-occurrence with interpretations and queries.

References

Adé, H., De Raedt, L., and Bruynooghe, M. (1995). Declarative bias for specific-to-general ILP systems. *Machine Learning*, 20(1):119–154.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499. Morgan Kaufmann, San Francisco, CA, USA.

Avriel, M. (2003). *Nonlinear programming: analysis and methods*. Dover Publications.

Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 1:585–592.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. (1987). Occam's razor. *Inf. Process. Lett.*, 24(6):377–380.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Brand, M. and Huang, K. (2003). A unifying theorem for spectral embedding and clustering. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*.

Cho, R., Campbell, M., Winzeler, E., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T., Gabrielian, A., Landsman, D., Lockhart, D., et al. (1998). A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2(1):65–73.

Chung, F. and Lu, L. (2004). The average distance in a random graph with given expected degrees. *Internet Mathematics*, 1(1):91–113.

Cox, T. and Cox, M. (2001). *Multidimensional Scaling*. CRC Press.

Dantsin, E., Eiter, T., Gottlob, G., and Voronkov, A. (2001). Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425.

Davidson, R. and Harel, D. (1996). Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331.

De Raedt, L. (1997). Logical settings for concept-learning. *Artificial Intelligence*, 95(1):187–201.

De Raedt, L. and Džeroski, S. (1994). First-order jk-clausal theories are PAC-learnable. *Artificial Intelligence*, 70(1):375–392.

De Raedt, L. and Ramon, J. (2004). Condensed representations for inductive logic programming. In *Proceedings of 9th International Conference on the Principles of Knowledge Representation and Reasoning*, pages 438–446.

Dehaspe, L. and De Raedt, L. (1997). Mining association rules in multiple relations. pages 125–132.

Dehaspe, L. and Toivonen, H. (1999). Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36.

Di Battista, G., Eades, P., Tamassia, R., and Tollis, I. (1994). Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry-Theory and Application*, 4(5):235–282.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Džeroski, S. (2007). *Introduction to statistical relational learning*, chapter 3, pages 57–92. MIT Press.

Džeroski, S. (2003). Multi-relational data mining: an introduction. *ACM SIGKDD Explorations Newsletter*, 5(1):1–16.

Emde, W. and Wettschereck, D. (1996). Relational instance-based learning. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 122–130. Morgan Kaufmann.

Fang, H., Tong, W., Shi, L., Blair, R., Perkins, R., Branham, W., Hass, B., Xie, Q., Dial, S., Moland, C., , and Sheehan, D. (2001). Structure-activity relationships for a large diverse set of natural, synthetic, and environmental estrogens. *Chem. Res. Tox*, 14:280–294.

Fodor, I. (2002). A survey of dimension reduction techniques. *Manuscript*.

Friedrich, T. (2002). Nonlinear dimensionality reduction with locally linear embedding and isomap. Master's thesis, The University of Sheffield.

Fuchs, N., Schwertel, U., and Torge, S. (1999). Controlled natural language can replace first-order logic. In *Automated Software Engineering, 1999. 14th IEEE International Conference on.*, pages 295–298.

Getoor, L. and Taskar, B. (2007). *Introduction to statistical relational learning*. MIT Press.

Globerson, A., Chechik, G., Pereira, F., and Tishby, N. (2007). Euclidean Embedding of Co-occurrence Data. *The Journal of Machine Learning Research*, 8:2265–2295.

Golub, G. and Kahan, W. (1965). Calculating the singular values and Pseudo-Inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, 2(2):205–224.

Gomory, R. and Hu, T. (1961). Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, pages 551–570.

Hannan, J. and Bishop, J. (1997). A comparison of fast training algorithms over two real problems. In *Fifth International Conference on Artificial Neural Networks*, pages 1–6.

Helma, C., Kramer, S., and De Raedt, L. (2002). The molecular feature miner MolFea. In *Proceedings of the Beilstein-Institut Workshop*.

Herman, I. and Marshall, M. (2000). Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*.

Hutchinson, A. (1997). Metrics on Terms and Clauses. In *Proceedings of the 9th European Conference on Machine Learning*, pages 138–145. Springer-Verlag London, UK.

Hyvrinen, A. (1999). Survey on independent component analysis. *Neural Computing Surveys*, 2(4):94–128.

Igel, C. and Husken, M. (2000). Improving the Rprop learning algorithm. In *Proceedings of the second international ICSC symposium on neural computation (NC 2000)*, pages 115–121.

Indukuri, K. (2004). Fusion: A Visualization Framework for Interactive ILP Rule Mining with Applications to Bioinformatics. Master's thesis, Virginia Polytechnic Institute and State University.

Iwata, T., Saito, K., Ueda, N., Stromsten, S., Griffiths, T., and Tenenbaum, J. (2007). Parametric Embedding for Class Visualization. *Neural Computation*, 19(9):2536–2556.

Jain, A. and Dubes, R. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

Jolliffe, I. (2002). *Principal component analysis*. Springer New York.

Karhunen, J. (2001). Nonlinear independent component analysis. *Independent Component Analysis: Principles and Practice*, page 113.

Karlin, S. and Ghandour, G. (1985). Multiple-alphabet amino acid sequence comparisons of the immunoglobulin-chain constant domain. *Proceedings of the National Academy of Sciences*, 82(24):8597–8601.

Karwath, A. and Kersting, K. (2007). Relational Sequence Alignments and Logos. *Lecture Notes in Computer Science*, 4455:290.

Karwath, A., Kersting, K., and Landwehr, N. (2008). Boosting Relational Sequence Alignments. In *Eighth IEEE International Conference on Data Mining, 2008. ICDM'08*, pages 857–862.

Kramer, S., De Raedt, L., and Helma, C. (2001). Molecular feature mining in HIV data. In Provost, F. and Srikant, R., editors, *Proc. KDD-01*, pages 136–143, New York, NY, USA. ACM Press.

Kretzschmar, H., Stachniss, C., Plagemann, C., and Burgard, W. (2008). Estimating Landmark Locations from Geo-Referenced Photographs. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, pages 2902–2907.

Lang, K. (2005). Fixing two weaknesses of the spectral method. In *NIPS*, pages 715–722. MIT Press.

Lavrac, N. and Džeroski, S. (1994). Inductive logic programming. *Journal of Logic Programming*, 19(20):629–679.

Luo, B., C Wilson, R., and Hancock, E. (2003). Spectral embedding of graphs. *Pattern Recognition*, 36(10):2213–2230.

Michaels, G., Carr, D., Askenazi, M., Fuhrman, S., Wen, X., and Somogyi, R. (1998). Cluster analysis and data visualization of large-scale gene expression data. In *Pacific Symposium on Biocomputing*, volume 3, pages 42–53.

Muggleton, S., editor (1992). *Inductive Logic Programming.* Academic Press, New York, NY.

Muggleton, S. and King, R. (1991). Predicting protein secondary-structure using inductive logic programming. Technical report, Turing Institute, Glasgow, Scotland.

Newman, M. (2003). The Structure and Function of Complex Networks. *SIAM Review*, 45:167.

Nienhuys-Cheng, S. (1997). Distance between herbrand interpretations: A measure for approximations to a target concepts. pages 213–226.

Noack, A. (2004). An energy model for visual graph clustering. In *Proceedings of the 11th International Symposium on Graph Drawing (GD 2003), LNCS 2912*, pages 425–436. Springer-Verlag.

Noack, A. (2007). Energy Models for Graph Clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480.

Plotkin, G. (1971). *Automatic Methods of Inductive Inference.* Department of Machine Intelligence, and Perception, University of Edinburgh.

Ramon, J. (2002). *Clustering and Instance Based Learning in First Order Logic.* PhD thesis, K.U. Leuven, Leuven, Belgium.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *In Proceedings of the IEEE International Conference on Neural Networks.*

Roweis, S. and Saul, L. (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding.

Schatz, M., Phillippy, A., Shneiderman, B., and Salzberg, S. (2007). Hawkeye: an interactive visual analytics tool for genome assemblies. *Genome Biology*, 8(3):R34.

Schiffmann, W., Joost, M., and Werner, R. (1993). Comparison of optimized backpropagation algorithms. In *Proceedings of the European Symposium on Artificial Neural Networks, ESANN*, volume 93, pages 97–104.

Schneider, T. and Stephens, R. (1990). Sequence logos: a new way to display consensus sequences. *Nucleic Acids Research*, 18(20):6097–6100.

Schulz, H., Kersting, K., and Karwath, A. (2009). ILP, the Blind, and the Elephant: Euclidean Embedding of Co-Proven Queries. In *ILP*, Lecture Notes in Computer Science. Springer. to be published.

Srinivasan, A. (2000). The aleph manual. *Computing Laboratory, Oxford University.*

Srinivasan, A., Muggleton, S., King, R., and Sternberg, M. (1996). Theories for Mutagenicity: A Study of First-Order and Feature-based Induction. *Artificial Intelligence Journal*, 85:277–299.

Stolle, C., Karwath, A., and De Raedt, L. (2005). CLASSIC'CL: An Integrated ILP System. In Hoffmann, A., Motoda, H., and Scheffer, T., editors, *Discovery Science*, volume 3735 of *Lecture Notes in Computer Science*, pages 354–362. Springer.

Takane, Y., Young, F., and De Leeuw, J. (1977). Nonmetric individual differences multidimensional scaling: an alternating least squares method with optimal scaling features. *Psychometrika*, 42(1):7–67.

Tenenbaum, J., Silva, V., and Langford, J. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction.

Thomas, J. (2005). *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Center. http://nvac.pnl.gov/agenda.stm.

Travers, J. and Milgram, S. (1969). An experimental study of the small world problem. *Sociometry*, pages 425–443.

Van Laer, W. and De Raedt, L. (2001). How to upgrade propositional learners to first order logic: A case study. *Relational data mining*, pages 235–261.

Wagner, R. and Fischer, M. (1974). The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Declaration

I hereby declare that this thesis has been composed by me without any assistance and I have not used any sources or tools other than those cited. Furthermore I declare that this thesis has not been accepted in any other previous application for a degree.

Location, Date

Signature