Bridging the Gap between Distance and Generalisation

V. Estruch C. Ferri J. Hernández-Orallo M.J. Ramírez-Quintana
DSIC, Universitat Politècnica de València, Spain.
{vestruch,cferri,jorallo,mramirez}@dsic.upv.es

March 22, 2012

Abstract

Distance-based and generalisation-based methods are two families of artificial intelligence techniques that have been successfully used over a wide range of real-world problems. In the first case, general algorithms can be applied to any data representation by just changing the distance. The metric space sets the search and learning space, which is generally instance-oriented. In the second case, models can be obtained for a given pattern language, which can be comprehensible. The generality-ordered space sets the search and learning space, which is generally model-oriented. However, the concepts of distance and generalisation clash in many different ways, especially when knowledge representation is complex (e.g. structured data). This work establishes a framework where these two fields can be integrated in a consistent way. We introduce the concept of distance-based generalisation, which connects all the generalised examples in such a way that all of them are reachable inside the generalisation by using straight paths in the metric space. This makes the metric space and the generality-ordered space coherent (or even dual). Additionally, we also introduce a definition of minimal distance-based generalisation that can be seen as the first formulation of the Minimum Description Length (MDL)/Minimum Message Length (MML) principle in terms of a distance function. We instantiate and develop the framework for the most common data representations and distances, where we show that consistent instances can be found for numerical data, nominal data, sets, lists, tuples, graphs, first-order atoms and clauses. As a result, general learning methods that integrate the best from distance-based and generalisation-based methods can be defined and adapted to any specific problem by appropriately choosing the distance, the pattern language and the generalisation operator.

Keywords: learning from structured data representations, comprehensible models, distance-based methods, generalisation operators, minimal generalisation.

1 Introduction

Distances (and consequently, metric spaces) and generalisation (and its inverse, specialisation) play an important role in many artificial intelligence techniques that have been developed to date: on the one hand, a distance constitutes a good formalisation for the concept of similarity (or dissimilarity to be more precise). Similarity offers a well-founded inference principle for learning and formal reasoning since it is commonly assumed that *similar objects have similar*

properties. On the other hand, generalisation assumes that data, which can be described with the same pattern, share some properties. Thus, a pattern generalises the regularities seen in the data, where data become instances of one or more patterns.

These two concepts lead to very different approaches. Basically, distance-based methods are quite flexible (since a wide variety of similarity functions can be found for the most common data types), but, unlike generalisation-based methods, they do not infer a model (pattern).

Although there is important and relevant work on the hybridisation of these two approaches and, as a consequence, distances are used in many model-based techniques, these two approaches do not work well together in at least one important aspect. Given a distance, the generalisation process from specific cases to patterns is not led by the distance. As a result, a pattern that is obtained by generalisation is not usually consistent with any underlying distance. Conversely, distance-based methods that obtain a prototype or centroid cannot express the pattern that defines the concept. For instance, if we consider the edit distance over the lists *bbab*, *bab* and *aaba*, we see that the list *ab* is close to the previous lists (distances are 2, 1, and 2 respectively). However, a typical pattern that can be obtained by some model-based methods, **ba**, does not cover the list *ab*. The pattern does cover the list *dededfafbakgagggeewdsc*, which is at distance 20 from the three original lists. The pattern and the distance are clearly inconsistent. Conversely, if we obtain the prototype from the previous three lists (e.g. *bab*), we have a distance-based representative of the set, but we do not have a pattern that shows the common regularities.

In this paper, we analyse the relation between distance and generalisation, and we define some simple conditions that a generalisation operator should have in order to be consistent with a distance. These operators will be called distance-based generalisation operators. While many generalisation operators are not compatible with many distances, we will also show that we can find pairs of distances and generalisation operators that work well together for the most common datatypes: symbolic constants, numbers, tuples, lists, sets, graphs, first-order atoms and clauses. Additionally, we will see that in some cases the operator can also be minimal.

A work that addresses three fundamental concepts in artificial intelligence (distance, generalisation and knowledge representation) must show many links with existing work. For instance, in conceptual clustering [45], concepts are obtained by postprocessing clusters. However, the distance is disregarded and the concepts are inconsistent with the metric space. Another attempt to combine these concepts is the work from [40, 41] where the relation between models and distances is explored by introducing a way to derive a distance function from a generalisation operator. Nonetheless, this work is restricted to first-order objects using the well-known *lgg* operator [38]. How this idea can be brought to other datatypes has not yet been investigated.

Additionally, to the best of our knowledge, the opposite relation, that is, how to find generalisation operators given a distance has not been treated. Thus, it is an interesting (and novel) approach to set a distance and to see the generalisation operators that can be compatible with it (some might be constructed using the distance, and some others might not).

Thus, the present work is not just another hybridisation but an attempt to bridge the gap between the two worlds and make them compatible or even dual: every distance can have an associated generalisation operator and vice versa. As a result, this work includes several key contributions to these ideas, from which we highlight the following:

- A broad analysis of the relation between distance and generalisation and the definition of the notion of distance-based generalisation (section 2).
- The definition of two novel concepts, *nerve* and *skeleton* (section 3), to handle non-binary

generalisations in metric spaces.

- The definition of minimality of a distance-based generalisation operator with respect to a pattern language under a distance-based formulation of the Minimum Description Length (MDL) / Minimum Message Length (MML) principle [42, 47] (section 4).
- The introduction of distance-based generalisation operators and the theoretical analysis of their properties (such as minimality) for many common distances, pattern languages, and data representations: symbolic constants, numbers, tuples, lists, sets, graphs, first-order atoms, and clauses (sections 5 to 8), as well as their composability (section 9).

An extra contribution is the realisation of the relevance of the representation language when trying to combine a distance with a generalisation operator. Different results can be obtained with the same learning algorithm but with different languages and distances. We illustrate this with examples at the end of the paper.

Overall, this is an extensive work which introduces a new general framework and analyses specific instances for several kinds of languages. We obtain a number of theoretical results that characterise pairs of distances and operators that work well together. A distance-based generalisation operator is a tool which can be used to analyse and adapt existing techniques or to develop new ones that take the best from both generalisation-based and distance-based methods. Therefore, these operators can be employed in unsupervised contexts (e.g. to transform a classical clustering algorithm into a symbolic learner) or in supervised contexts because the negative examples are taken into account by the learning algorithm. Furthermore, it can also be used in other areas of artificial intelligence such as theory (belief) revision, case-based reasoning and knowledge integration, and can find insightful links with other areas such as fuzzy sets, rough sets, interval computing, kernel methods, granular computing and others, as it is incipiently explored in some parts of this paper (especially in section 11).

This paper is organised as follows. Section 2 analyses the gap that exists between distance and generalisation and includes the most relevant work related to this issue. After some intuitive ideas on what a distance-based generalisation should be, section 3 introduces the formal definitions and sets the framework for the rest of the paper. Section 4 introduces the notion of minimality, which is based on a distance-oriented adaptation of the MDL/MML principle. The following sections (5 to 8) are devoted to illustrating the setting and obtaining results for several datatypes: nominal, numerical, sets, lists, graphs and first-order objects. Section 9 deals with composability and shows that tuples allow distance-based generalisation operators for nested datatypes. To illustrate the application of previous results, in section 10 we include a very simple classification example and a clustering example, where several data representation languages, pattern languages, distance functions and generalisation operators are used. Section 11 discusses several variants and extensions of the framework, related notions and some future work. Finally, section 12 presents the conclusions.

2 The Gap between Distance and Generalisation

Distances in artificial intelligence are used profusely as an adequate formalisation of dissimilarity. There are many distances available for every datatype: numeric data, symbolic data, strings/lists, sets, graphs, ... [23]. In machine learning, many distance-based methods are usually associated to instance-based methods (e.g. nearest-neighbour [8]) or the so-called "local

model" techniques [1] (e.g. k-means, self-organising maps, locally weighted regression, etc.). The great advantage of distance-based methods is that the same algorithm can be applied to whatever kind of datatype provided that there is a distance defined for that datatype. However, the main disadvantage of distance-based methods is that the pattern (if any) is defined in terms of the distance to one or more instances, prototypes or centroids. This can be a serious restriction when comprehensible models are required. This is especially the case for problems with complex data (graphs, strings, trees). We extract no knowledge from expressions such as "elements which are near to graph x and graph y".

In order to obtain effective patterns, other methods exist that rely on the idea of generalisation. Basically, these methods express regularities in data by means of patterns or models. According to [34] (Chapter 2), generalisation-based methods define a generality-ordered space (usually a lattice or a direct graph) over a set of symbolic[1] patterns and generalisation/specialisation operators from which the search through this lattice is performed. For instance, most decision tree and rule learners, Inductive Logic Programming (ILP) methods, grammar learning methods and conceptual clustering are generalisation-based methods.

Therefore, the great advantage of generalisation-based methods is that the result is expressed in a pattern language that is intuitive for the datatype at hand. The main disadvantage is that, although many algorithms share similar ideas, each algorithm has to be completely redesigned if the datatype is changed. For instance, grammar and first-order inference share similar learning schemas, but they differ in the way the main ideas are put into practice.

From the short description above, the possibility of combining both learning techniques arises. This can mainly be done in two different ways, that is, moving from generalisations to distances or vice versa. The former possibility is explored in [41, 39]. The latter is typically explored in explanation-based learning, especially in conjunction with case-based reasoning (CBR) [2] or in cases where there are very few examples (anecdotal evidence). Other approaches such as [11, 12, 25, 43] focus on combining distance-based (often instance-based) methods with model-based learners in order to take the best of both. However, this connection is made at a shallow level (e.g. a model-based post-processing is made over an instance-based learner), and the patterns that are obtained may have no relation with the underlying distance.

By a *true* connection between distance and generalisation, we mean that generalisations must take the underlying distance into consideration (or at least the two are consistent). For instance, if we have a pattern that generalises, a black crow and a brown dove, we expect to go from one to the other through similar (close) objects without leaving the generalisation. For instance, we expect a brown crow to be in the generalisation. This would mean that we can gradually move from a black crow to a brown dove through a brown crow (or a black dove). The notion of 'gradually' or 'close' must be established in terms of the underlying metric space. This 'duality' is well known in cognitive sciences [26], where similarities (distances) and concepts (generalisations) are two sides of the same coin.

However, the clash appears since most generalisation operators and distance functions originate incompatible spaces. For instance, the pattern $*ab*$ for strings aba and bab excludes the shortest-distance transformation $aba \rightarrow ba \rightarrow bab$. In fact, this transformation explains why the (edit) distance between aba and bab is 2. The intuitive idea behind the concept of distance-based generalisation is that the elements in between (in terms of the distance) should

[1] By symbolic we mean close to natural language such as patterns based on (fuzzy) logic, regular expressions, if-then-else rules, numerical intervals, etc.

be included in the generalisation. This will be done by transporting the triangle inequality property of distances to generalisation operators. Another reason for the gap between the concepts of distance and generalisation is that distances are always defined between two objects whereas generalisations can be defined over an arbitrary set of elements. This suggests that the connection between distance and generalisation depends on how the examples are arranged.

Finally, another important 'disagreement' between distance and generalisation is the notion of fitting. In a metric space, the concepts of underfitting and overfitting are usually understood in terms of the attraction function and the distances to some prototype elements (the closer the border is to these elements, the more fitting we have). In a generality-ordered space, the concepts of underfitting and overfitting (and hence minimality) are understood in terms of coverage (the lower the coverage, the more fitting we have). If both spaces are divergent, one can see overfitting in one space and underfitting in the other (or viceversa).

Given the previous analysis of the relationship and the deficient connection between distances and generalisation, there can be several ways to bridge this "gap". We base our proposal on three notions: *reachability*, *intrinsicality* and *minimality*.

Our notion of reachability means that given two elements and their generalisation, we should be able to reach both elements from each other by making small "steps" (or transformations) to other elements which must also be in the generalisation. While in cognitive science similarity is seen as transformation [27], in a more formal context the concept of short step must be understood using distances. In the previous example, pattern $aba \vee bab$ is worse than pattern $*ab*$, since the latter connects strings aba and bab through ab. In a continuous space this can be understood as having a curve that connects both objects inside the generalisation.

The previous notion forces any generalisation to connect the elements that led to the generalisation, but this path does not need to be straight or even short. To avoid this, a second notion, which we call intrinsicality, is introduced. An intrinsic generalisation means that given two objects, their generalisation should cover all the elements that are between them in terms of the underlying distance. More precisely, all the elements in the shortest path between two elements (in Euclidean spaces, the straight segment that connects them) must be included in the generalisation. In the previous example, both ab and ba should be included in the generalisation.

Figure 1 shows examples of both properties in $\mathbb{R}^2$ with the Euclidean distance: neither reachability nor intrinsicality are satisfied in generalisation $G1$; in $G2$ the elements e_A and e_B are reachable but not by means of an intrinsic path because some of the elements in between are excluded; finally, reachability and intrinsicality are satisfied in $G3$ and $G4$.

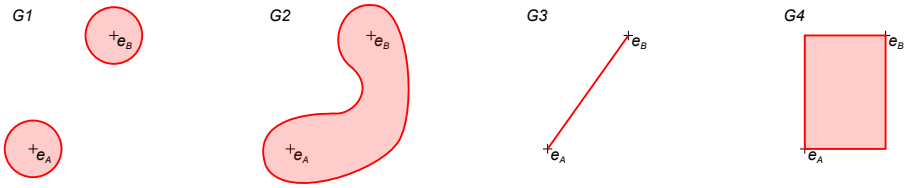


Figure 1: Generalising the elements e_A and e_B in $\mathbb{R}^2$.

Although the notions above are informal, it seems clear that for two elements and continuous spaces, the condition of ensuring that every intermediate point is in the set (intrinsicality) implies the notion of reachability (since it defines a path from both elements).

However, for more than two elements, both the notion of reachability and intrinsicality can

be understood in many ways. Consequently, for sets of more than two elements, we will impose the notion of intrinsicality for *some* pairs of elements. The pairs of elements that will have to comply with the intrinsicality property will be set by a path or connected graph which we will call *nerve*. The nerve does not need to be the minimum spanning tree, although this could be a typical choice. With this, we ensure that all the elements in the set are reachable from any of them by moving from one element to another through direct (intrinsical) paths. This makes the metric space and the generalisation compatible.

In figure 2, generalisations $G1$ and $G2$ do not connect the five elements to be generalised. $G3$ does connect them, but there is no segment included in $G3$ that links the elements e_B and e_C with e_A, e_D or e_E in a straight way. Only the generalisations G_4, G_5 and G_6 can connect the five elements through straight segments.

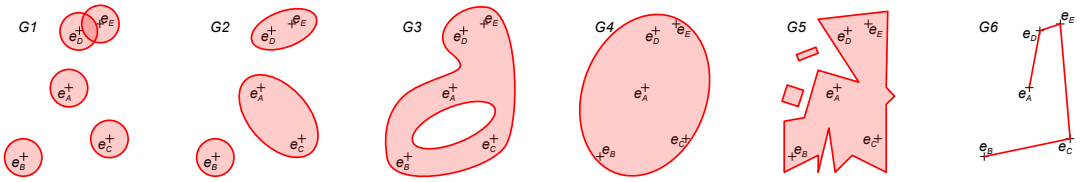


Figure 2: Generalising the elements $E = \{e_A, e_B, e_C, e_D, e_E\}$. Generalisations $G1$, $G2$ and $G3$ are not reachable through a path of straight segments. Generalisations $G4$, $G5$ and $G6$ include a path of segments connecting them.

There is still one remaining aspect to be considered, *minimality*, understood not only in terms of fitting the set (i.e., semantic minimality) but also as the simplicity of the pattern (i.e., syntactic minimality). Figure 2 shows that some of the generalisations are more specific than others (semantics), and some are simpler than others (syntax). We have $G4$, which is a simple generalisation; $G5$ a complex generalisation and finally, $G6$ a very simple and very specific generalisation. The notion of minimality that we will work with combines both simplicity and specificity and also bases the semantic part on the underlying distance. This is remarkable, since, for non-metric spaces, the generality of two patterns is incomparable if none is included in the other or we do not have the notion of volume.

3 Distance-based Generalisation Operators

The discussion on the three notions (reachability, intrinsicality and minimality) in section 2 has been made at an informal level. In this section, we switch to a more formal level defining precisely what a generalisation is and how the notions of local intrinsicality (through the triangle inequality) and global reachability (through the notion of nerve) define the concept of distance-based generalisation operator. We leave the notion of minimality for section 4.

3.1 Generalisation operators

Let us start with the definition of generalisation.

Definition 1. (Generalisation) *A generalisation of a set of elements E is any set $G \supset E$.*

From a predictive/explanatory point of view, a generalisation that is just expressed as a set of elements (e.g. $\{2, 4, 6, \ldots\}$) is much less useful than a generalisation expressed as a pattern (e.g. $\{x : odd(x)\}$). That is, a pattern $p \in \mathcal{L}$ can be considered as an intensional (and hopefully comprehensible) manner of describing a set of elements in a pattern language $\mathcal{L}$.

Definition 2. (Pattern) *A pattern p for $E \subset X$ is any correct expression in $\mathcal{L}$ such that $E \subset Set(p)$, where $Set(p)$ denotes the set of all elements $x \in X$ which are covered by p.*

In this way, we can extend the well-known operations for sets to patterns. For example, we can say that a pattern p_1 is included in a pattern p_2 if $Set(p_1) \subseteq Set(p_2)$. Note that a given set can be denoted or covered by several patterns. The pattern language $\mathcal{L}$ will be defined according to the problem to be solved, and specially according to the kind of patterns that the user can understand. Note that if there is no representation bias, $\mathcal{L}$ can even be defined to exhaustively cover 2^X or, if X is infinite, $\mathcal{L}$ can be defined to be Turing-complete.

For instance, given the strings abb and abc, and the regular pattern $ab*$, then $Set(ab*) = \{ab, abc, aba, abb, abaa, ...\}$, and we say that $ab*$ covers the elements abb and abc because $abb \in Set(ab*)$ and $abc \in Set(ab*)$. Furthermore, depending on our pattern language $\mathcal{L}$, some sets can be expressed as generalisations but others cannot. To give an example, the set of words of even length cannot be expressed with regular languages.

Given a pattern language $\mathcal{L}$, its expressiveness can always be improved by combining patterns via logical operators. Pattern disjunction becomes quite useful in this sense, since we can express sentences such as "elements belonging to a pattern p_1 or a pattern p_2".

Definition 3. (Pattern disjunction) *Given two patterns p_1 and p_2, the expression $p_1 + p_2$ represents a pattern such that $Set(p_1 + p_2) = Set(p_1) \cup Set(p_2)$. For simplicity, the pattern $p = p_1 + \ldots + p_n$ will be expressed as $p = \sum_{i=1}^{n} p_i$.*

[A generalisation operator simply maps sets of elements into patterns covering them:

Definition 4. (Generalisation operator) *Let X be a space of elements and let $\mathcal{L}$ be a pattern language. For every finite set E of elements in X, a generalisation operator Δ is a function such that $\Delta(E) = p$ where $p \in \mathcal{L}$ and $E \subset Set(p)$.*

Definition 5. (Binary generalisation operator) *A binary generalisation operator is a generalisation operator restricted to two elements $\{e_1, e_2\}$ which is denoted as $\Delta(e_1, e_2)$.*

Note that these definitions say nothing about the nature of the resulting pattern, but that it must cover the original elements (see figures 1 and 2).

3.2 Similarities, distances and the triangle inequality

As discussed in the introduction, similarities and distances have been used profusely in many areas of artificial intelligence, especially in case-base reasoning (CBR) [20], clustering and other kinds of instance-based learning. Although in some cases the similarity measure does not need to conform to some common properties (e.g. symmetry or the triangle inequality), distances (metrics) are generally preferrable because of its conceptual or practical implications (see, e.g., [9] for a taxonomy of similarity measures and distances). For instance, there are many techniques in CBR or similarity-based retrieval which assume the triangle inequality [20], in order to, e.g., reduce the number of comparisons, since the search can exploit the triangle inequality.

Now, we introduce the notion of distance. A distance d is a function $d : X \times X \rightarrow \mathbb{R}$, such that for every $e_1, e_2, e_3 \in X$, $d(e_1, e_1) = 0$ (identity), $d(e_1, e_2) = d(e_2, e_1)$ (symmetry) and $d(e_1, e_2) \leq d(e_1, e_3) + d(e_3, e_2)$ (subadditivity / triangle inequality)[2]. There is an alternative way of expressing what a distance is, by rewriting the triangle inequality as a recursive function: $d(e_1, e_2) = min_{t \in \mathcal{T}}\{d(e_1, t(e_2)) + l(t)\}$, and defining a language of transformation operators $\mathcal{T}$ and an effort, cost or length ($l(t)$) of the transformation operator t. Many discrete distances are defined in this way[3]. For example, the Hamming distance of strings of size n and number of symbols m requires a $\mathcal{T}$ which is composed of $n \times m$ transformation operators (just changing a symbol), and all of them have the same cost. In a Euclidean space in $\mathbb{R}^n$, transformations would be just movements in any direction and of any length, i.e., $\mathcal{T}$ is just the set of vectors in $\mathbb{R}^n$. As a result, a distance is just seen as the length of (one of) the minimal (possibly infinite) sequence(s) of transformations between the objects. It is this view of transformation between objects which justifies the use of distances (instead of just similarity functions) and also motivates the comparison with the notion of generalisation.

At this point, a generalisation of e_A and e_B must explain the value of $d(e_A, e_B)$, in such a way that this generalisation must cover those elements that are exactly placed between e_A and e_B (see cases where this is true and false in figure 1). This leads to the two definitions below:

Definition 6. (Intermediate element relative to a distance) *Given a metric space (X, d) and two elements $e_1, e_2 \in X$, we say that an element $e_3 \in X$ is between e_1 and e_2, or is an intermediate element relative to d, if $d(e_1, e_2) = d(e_1, e_3) + d(e_3, e_2)$.*

Definition 7. ***(Binary distance-based pattern and binary distance-based generalisation operator)*** *Let (X, d) be a metric space, $\mathcal{L}$ a pattern language, and a set of elements $E = \{e_1, e_2\} \subset X$. We say that a pattern $p \in \mathcal{L}$ is a binary distance-based (db) pattern of E if p covers all the elements between e_1 and e_2. Additionally, we say that Δ is a binary distance-based generalisation (dbg) operator if $\Delta(e_1, e_2)$ is a binary distance-based pattern.*

The above definition is a simple formalisation of the notion of intrinsicality. Interestingly, this works for both continuous and discrete spaces. An example is introduced next:

Example 1. *Let us suppose that the elements $e_A = (1, 1)$ and $e_B = (3, 4)$ in figure 1 are in the metric space $(\mathbb{R}^2, d)$, where d is the Euclidean distance. The intermediate elements are those in the segment $\overline{e_A e_B}$. Therefore, only patterns representing the generalisations $G3$ and $G4$ are dbg. However, if d is set to the Manhattan distance[4], the intermediate elements are placed in the rectangle delimited by e_A and e_B. Therefore, $G4$ would be a dbg but not $G3$.*

3.3 Nerves and skeletons

For the case of more than two elements to be generalised, the concept of "nerve" of a set of elements E is needed to define *dbg* operators. This corresponds to the notion of reachability.

[2]Note that we do not include the 'identity of indiscernibles' (i.e. $d(e_1, e_2) = 0 \rightarrow e_1 = e_2$), so a distance can be just a pseudometric.

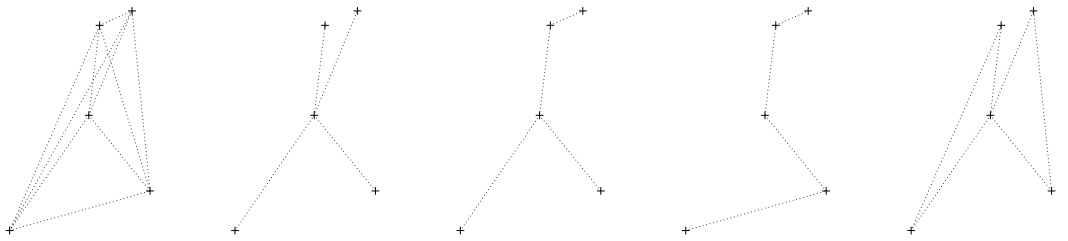
[3]In some cases, transformations are not symmetrical (e.g. inserting and deleting a character in a string), so we talk about quasimetrics (if the 'indentity of indiscernibles' is relaxed then we have a pseudoquasimetric, which is sometimes called hemimetric). In the rest of the paper, we will assume symmetry, although this condition is not strictly necessary to materialise the notion of reachability and intrinsicality, since the notion of being in between could be just be defined using both directions.

[4]Given two points $e_A = (a_1, a_2)$ and $e_B = (b_1, b_2)$ in $\mathbb{R}^2$, the Manhattan distance is defined as $d(e_A, e_B) = |a_1 - b_1| + |a_2 - b_2|$.

Definition 8. *Given a set of elements $E \subset X$, a nerve η of E is any connected*[5] *graph taking the elements belonging to E as vertices.*

Definition 9. ***(Nerve function)*** *Let (X, d) be a metric space and let Γ_X be the set of undirected and connected graphs over subsets of X. A nerve function $N : 2^X \to \Gamma_X$ maps every set $E \subset X$ into a nerve $\eta \in \Gamma_X$, such that each element e in E is inequivocally represented by a vertex in η and vice versa. We say the obtained graph $N(E) = \eta$ is a nerve of E.*

Note that the set and the graph may be infinite. For instance, E might be the set of natural numbers and the nerve its sequential connection. Observe that if $|E| \leq 1$, we have the empty graph, and if $|E| = 2$, the only possible nerve is a one-edged graph.



(a) (b) (c) (d) (e)

Figure 3: Five nerves for a set of five points in $\mathbb{R}^2$. **(a)** A complete nerve. **(b)** A star nerve. **(c)** The minimum distance nerve for the Euclidean distance. **(d)** The minimum distance nerve for the Manhattan distance. **(e)** An arbitrary nerve.

Some typical nerve functions (see figure 3) are: the complete graph, a radial/star graph around a vertex, the Minimum Distance Nerve, which is the nerve that gives the smallest sum of segment distances, which can be calculated by a Minimum Spanning Tree algorithm (an undirected counterpart to the Travelling Salesman path). Only the latter two have to be derived using the underlying distance. Obviously, not all these nerves can be computed efficiently.

Definition 10. ***(Skeleton)*** *Let (X, d) be a metric space, $\mathcal{L}$ a pattern language, a set $E \subseteq X$, and η a nerve of E. Then, the skeleton of E for η, denoted by skeleton(η), is defined as a set which only includes all the elements $z \in X$ between x and y, for every $(x, y) \in \eta$.*

Figure 4 shows the skeleton for different distances. We see that the shape and, especially, the width is significantly different according to each distance.

The basic idea now is that we look for generalisations that include the skeleton. We can say that the nerve and the skeleton configure a support set upon which the generalisations should be constructed. From here, we can define the notions of distance-based pattern and operator.

Definition 11. ***(Distance-based pattern and distance-based pattern for a nerve η)*** *Let (X, d) be a metric space, $\mathcal{L}$ a pattern language, E a finite set of examples. A pattern p is a db pattern of E if there exists a nerve η of E such that skeleton$(\eta) \subset Set(p)$. If the nerve η is known, then we will say that p is a db pattern of E for η.*

[5]Here, the term connected refers to the well-known property for graphs.

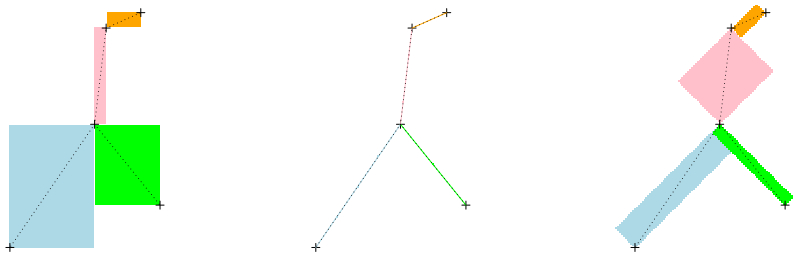


Figure 4: Three skeletons for the points in $\mathbb{R}^2$ and the nerve shown in figure 3 (c). **(Left)** Manhattan distance. **(Middle)** Euclidean distance. **(Right)** Chebyshev (box) distance.

Definition 12. ***(Distance-based generalisation operator)*** *Let (X, d) be a metric space and $\mathcal{L}$ be a pattern language. We will say that a generalisation operator Δ is a dbg operator if for every $E \subseteq X$, $\Delta(E)$ is a db pattern of E.*

The above definition can be characterised for one nerve function in particular.

Definition 13. ***(Distance-based generalisation operator for a nerve function** N**)*** *Let (X, d) be a metric space and $\mathcal{L}$ be a pattern language. A generalisation operator Δ is a dbg operator for a nerve function N if for every $E \subseteq X$, $\Delta(E)$ is a db pattern of E for $N(E)$.*

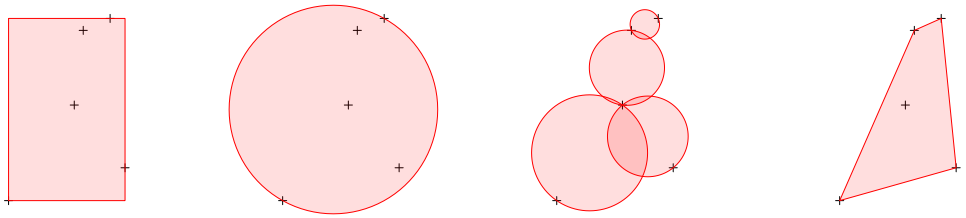


Figure 5: Four patterns in $\mathbb{R}^2$ for the same set of points. **(a)** An axis-parallel rectangle. **(b)** A circle. **(c)** A pattern as a set of circles. **(d)** A pattern as the convex hull.

In figure 5 we draw the result of several generalisation operators over the same set of elements. Pattern (a) is the smallest axis-parallel rectangle which includes all the elements. Pattern (b) is the smallest circle which includes all the elements. Pattern (c) is a set of circles. Pattern (d) is the smallest polygon covering the elements, i.e., the convex hull.

If we use the Manhattan distance, then we can clearly see that pattern (a) is *db* for the complete nerve, so it is *db* for any nerve. We see that pattern (b) is *db* for the minimum distance nerve, so it is *db* (but it is not *db* for any nerve, e.g., the complete nerve). In fact, in general the smallest enclosing circle is not a *dbg* for the Manhattan distance. In contrast, pattern (c) is *db* for only a few nerves. Finally, we see that pattern (d) is not *db* for any nerve. If we use the Euclidean distance, things change completely, patterns (a), (b) and (d) are now db for any nerve, while pattern (c) is *db* for some nerves. This happens because the skeleton for the Euclidean distance is very thin, as we saw in figure 4. Finally, for the Chebyshev (or box) distance, patterns (a) and (b) are not db for any nerve, pattern (b) is *db* for the star nerve (but just for this set of points), and pattern (c) is *db* for some nerves.

As we have seen, in some cases, we will be able to show that a generalisation operator is *db* for any nerve. In other cases, we will only require a nerve for which we can prove the generalisation operator is *db*. In relation to this, it is possible to define n-ary operators from binary operators when the nerve function is given beforehand.

Proposition 1. *Let $\mathcal{L}$ be a pattern language endowed with the operation $+$ (see definition 3) and let Δ^b be a binary dbg operator in $\mathcal{L}$. Given a finite set of elements E and a nerve function N, the generalisation operator Δ_N defined as follows is a dbg operator for N.*

$$\Delta_N(E) = \sum_{\forall (e_i,e_j)\in N(E)} \Delta^b(e_i, e_j)$$

Proof. It follows from the definition of *dbg* operator. □

In figure 5, this is exactly the case of pattern (c), which is constructed by joining (+) the smallest circle embracing two points for the pairs which appear in the nerve. Since the binary operator which calculates the circle embracing two points (Δ^b) is db for the Manhattan, Euclidean and Chebyshev distances, we have from proposition 1 that using a nerve function N and joining the patterns (Δ_N) we get a db generalisation operator. These examples and the previous proposition highlights the relevance of the nerve and the idea of composition.

3.4 Convex hulls, tight spans and minimality

For $\mathbb{R}^n$, some of the previous notions seem familiar. First, the notion of 'betweenness', which is derived from the triangle inequality, generally corresponds to a straight line between points, the shortest path between two examples. This can be generalised for some non-Euclidean spaces (and some volumes in Euclidean spaces) through the notion of *geodesic* [29, sec.1.4], such as the distance between points on the surface of a sphere (e.g., the Earth). However, this notion is not valid for many other non-Euclidean metric spaces, because there are many shortest paths.

Definition 14. *A set S is convex, relative to a distance d, if every intermediate element between two elements in S according to d (see definition 6) is in S.*

Definition 15. *Given a set of elements E, the convex hull of E relative to a distance d, denoted by $CH(E, d)$, is the intersection of all the convex sets, relative to d, which contain all the elements in E.*

For instance, patterns (a) and (d) in figure 5 match $CH(E, d)$ with d being the Manhattan distance and the Euclidean distance respectively. Note that $CH(E, d)$ is different to the smallest set containing all the elements in E and the elements in between, which would be the skeleton using the complete nerve. The connection between convex hull and skeleton can be expressed as follows:

Proposition 2. *Given a set of elements E, any nerve function N, and a distance d, we have that $skeleton(N(E)) \subset CH(E, d)$.*

Proof. Direct since $CH(E, d)$ contains E and any intermediate element in E. □

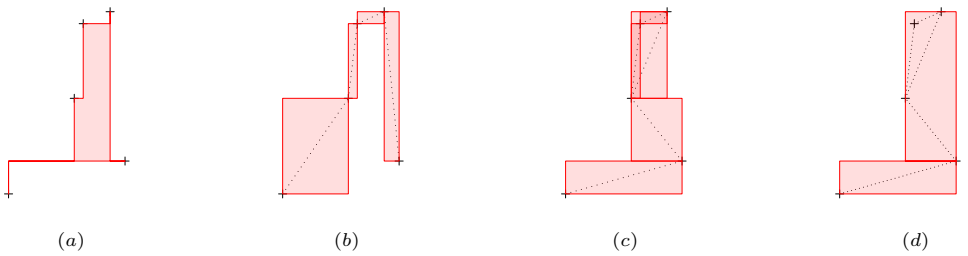


Figure 6: Four patterns in $\mathbb{R}^2$ for the same set of points using the Manhattan distance. **(a)** The tight span. **(b)** A db pattern for one given nerve. **(c)** A db pattern for another given nerve. **(d)** A different db pattern for the same nerve.

Outside Euclidean spaces, we have the notion of tight span or metric envelope [10]. In figure 6 (a) we see the tight span of five examples using the Manhattan distance.

The notion of tight span is rather similar to the notion of skeleton, except for the issue that for a tight span we only require one minimal path, while in the skeleton we require all, and for the fact that a skeleton may not be convex or even have holes. This is clear if we compare the skeleton in figure 4 (middle) (or the corresponding 'hollow' skeleton that would be generated by a complete nerve) with the convex hull in figure 5 (d) for the Euclidean distance. In addition, there are few similarities between the tight span and *db* patterns as figure 6 shows for the Manhattan distance.

The notion of tight span is not the appropriate extension of convex hull in the context of generalisations. In fact, the tight span is usually known as hyperconvex hull, because it is not exactly the extension of convex hull for non-Euclidean spaces.

4 Minimal Distance-based Generalisation Operators

Given the definition of *dbg* operator in section 3, we can now guarantee that a pattern obtained by a *dbg* operator from a set of elements ensures that all the original elements are reachable inside the pattern through intrinsic (direct) paths. However, the generalisation can contain many other, even distant, elements, as figure 2 shows, with a generalisation which is arbitrarily general ($G4$) and/or arbitrarily complex and whimsical ($G5$). If we define a general criterion to determine the fitness and/or the simplicity of a generalisation, we could set an order relation and compute the minimal (hence optimal) generalisation. This typical approach has given interesting generalisation operators in the past. The least general generalisation (lgg) operator [38] in the field of ILP [36] is the best example of this. However, for other pattern languages, this may not be the case. In order to obtain a general criterion for minimal generalisation, we will start with the concept of generality/fitness, which will lead us to the concept of simplicity.

A first idea of the concept of "less general than" could be the inclusion operator ($\subset$). Simply, a pattern p for E given by a dbg operator Δ would be less general than a pattern p' for E given by Δ', if $Set(\Delta(E)) \subset Set(\Delta'(E))$. However, this entails several problems:

- Most generalisations are not comparable, since neither $Set(\Delta(E)) \subset Set(\Delta'(E))$ nor vice versa. Also, the inclusion operator between sets ($\subset$) ignores the underlying distance. Consider patterns $\Delta_1(E) = p_1$ and $\Delta_2(E) = p_2$, where $Set(p_1)$ fits E better than $Set(p_2)$ does, as figure 7 shows. The two patterns are not comparable via the inclusion operator.

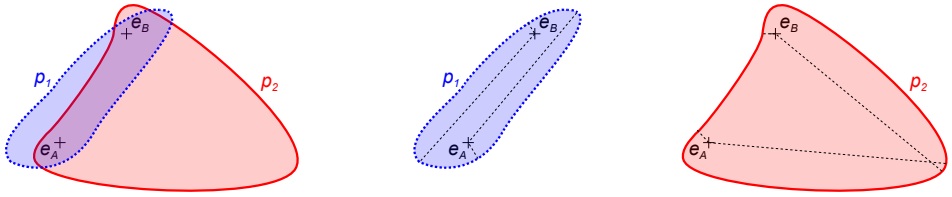


Figure 7: Two patterns p_1 and p_2 generalising $E = \{e_A, e_B\}$ that are not comparable via the inclusion operator. (**Left**) Both patterns. (**Middle**) The minimum and maximum distances to the border for p_1. (**Right**) The minimum and maximum distances to the border for p_2.

- The inclusion operator ignores the complexity of the pattern, which would lead to overfitting or, in some cases, to the inexistence of a minimum. For example, consider $\mathbb{R}^2$ with the Euclidean distance and pattern language $\mathcal{L}$ as the set of finite unions of rectangles in $\mathbb{R}^2$. As figure 8 shows, the pattern p_0, can be a reasonable *dbg* of e_A and e_B. However, a pattern such as p_1 is also *db* and $Set(p_1) \subset Set(p_0)$. The pattern p_2 is *db* as well and $Set(p_2) \subset Set(p_1)$. However, the complexity of p_2 is unnecessary for the data set at hand. Furthermore, note that, in this case, the minimum does not exist because for any *db* pattern we can always find another *db* pattern included in the previous one.

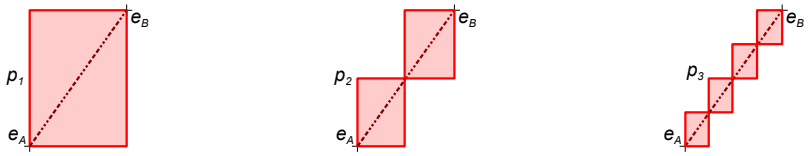


Figure 8: Generalising two points with rectangles in a Euclidean space. The skeleton is shown as the straight line connecting them. (**Left**) Pattern with one rectangle. (**Middle**) More specific pattern with two rectangles. (**Right**) Yet more specific pattern with four rectangles.

Therefore, these two drawbacks suggest the definition of the notion of optimality based on a more abstract and generic principle rather than the inclusion between sets. An abstract, well-founded and widely-used principle that connects the notions of fitness and simplicity is the well-known MDL/MML principle [42, 47]. This principle says that a high level of "complexity" of a pattern is reasonable only if a sufficient number of examples justifies it. In the example above, a pattern made of a thousand small rectangles becomes too complex, even if it fits the evidence E really well. In general, this principle evaluates the optimality of a model by means of a trade-off between the fitness and the simplicity of the given model.

In our framework, the optimality of a generalisation will be defined in terms of a cost function, denoted by $k(E, p)$, which considers both the complexity of the pattern p and how well the pattern p fits E in terms of the underlying distance. More formally,

Definition 16. (Cost function) *A cost function* $k : 2^X \times \mathcal{L} \to \mathbb{R}^+ \cup \{0\}$ *is a mapping where* $E \subset X$ *is finite,* p *is any pattern covering* E *and* $k(E, p)$ *can be infinite only when* $Set(p) = X$*. The function* $k(E, p)$ *will be expressed as* $c(p) + c(E|p)$ *where* $c(p)$ *is the syntactic cost (or complexity) function (which measures how complicated the pattern is) and* $c(E|p)$ *is the semantic cost (or fitness) function (which measures how the pattern fits the data* E*).*

The novel point here is that $c(E|p)$ will be expressed in terms of the distance employed, so this can be considered the first distance-based formulation of the MDL/MML principle. Let us see the details for $c(p)$ and $c(E|p)$.

This function $c(p)$ will strongly depend on the sort of data and the pattern space $\mathcal{L}$ we are dealing with. If, for example, the generalisation of two real numbers is a closed interval containing them, then some simple choices for $c(p)$ would be the length of the interval, a constant value or the bits required to code the rational numbers defining the interval. If we consider first-order atoms, its complexity could be given by the number of (variable and ground) symbols in an atom. In languages where the structures (that is, the basic patterns) can be replicated (sets of hyper-rectangles, decision trees, neural networks, etc.), then some candidate functions would be the number of structures used or the VC-dimension [46].

As stated above, the function $c(E|p)$ must be based on the underlying distance. All the definitions we present here are based on or inspired by the well-known concept of border of a set, which *can only be defined with a distance* (since it uses the notion of ball).

Definition 17. (Border) *Let (X, d) be a metric space and $B_r(e)$ the* **closed ball centred on e with radius** *r. We will say that an element e belonging to set $A \subset X$ is a border point, if for every $\epsilon > 0$, $B_\epsilon(e)$ is not totally included in A. The border of A will be denoted by ∂A.*

Intuitively, if a pattern p_1 fits E better than a pattern p_2, then the border of p_1 (∂p_1) will somehow be nearer to E than the border of p_2 (∂p_2). Note that this makes even more sense if the distance and the generalisation are compatible. As the concept of border of a set is something inherent to metric spaces, the function $c(E|p)$ can be defined independently from the datatype, as shown in Table 1. Figure 7 shows how these functions can be calculated.

	$\mathcal{L}$	$c(E\|p)$	Description
c_{inf}	Any	$\sum_{\forall e \in E} r_e$ with $r_e = inf_{r \in \mathbb{R}} B_r(e) \not\subset Set(p)$	Infimum of uncovered elements
c_{sup}	Any	$\sum_{\forall e \in E} r_e$ with $r_e = sup_{r \in \mathbb{R}} B_r(e) \subset Set(p)$	Supremum of covered elements
c_{min}	Any	$\sum_{\forall e \in E} min_{e' \in \partial Set(p)} d(e, e')$	Minimum to the border
c_{max}	$Set(p)$ is bound	$\sum_{\forall e \in E} max_{e' \in \partial Set(p)} d(e, e')$	Maximum to the border

Table 1: Some definitions of the function $c(E|p)$. These functions can be combined and the distances might be normalised (e.g. dividing them by the mean or the median) beforehand.

Example 2. *Consider the data language X composed of all the strings with alphabet $\Sigma = \{a, b\}$ and the edit distance with the same cost for insertions and deletions (substitutions not directly allowed). Consider a pattern language $\mathcal{L}$ composed of constants in Σ with variables which can take one or no symbols. For the set of examples $E = \{b, bb, bba\}$ and patterns: $p_1 = bV_1V_2$, $p_2 = V_1bV_3$ and $p_3 = V_1V_2V_3$, if we apply the functions in Table 1 we have that $\partial Set(p_1) = \{b, ba, bb, baa, bab, bba, bbb\}$ (which equals $Set(p_1)$), $\partial Set(p_2) = \{b, ba, bb, ab, aba, abb, bba, bbb\}$ (which equals $Set(p_2)$), and $\partial Set(p_3) = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$ (which is a strict subset of $Set(p_3)$, since all the strings of size 2, 1 and 0 are not in the border). From here, we calculate the functions in Table 1: $c_{inf}(E|p_1) = c_{inf}(E|p_2) = 1 + 1 + 1 = 3$, $c_{inf}(E|p_3) = 3+2+1 = 6$. The results of c_{sup} would be equal to $c_{inf} - 3$, the results of c_{min} equal to c_{inf}, and, finally, $c_{max}(E|p_1) = 2+2+2 = 6$, $c_{max}(E|p_2) = 2+3+3 = 8$ and $c_{max}(E|p_3) = 4+5+4 = 13$.*

In general, the functions $c(p)$ and $c(E|p)$ can be combined to obtain many possibilities for $k(E, p)$. Additionally, in many cases we will use cost functions such that the fitness part $c(E|p)$ is also consistent with the pristine notion of "more general than" in terms of the inclusion operator. The following definition formalises cost functions of this kind.

Definition 18. (Inclusion-preserving function) *A function f over sets of examples E and patterns p, denoted by $f(E, p)$, is inclusion-preserving iff for every $E \subset X$ and for every two patterns p and p' both covering E, if $Set(p) \subset Set(p')$ then $f(E|p) \leq f(E|p')$.*

This definition can be applied to cost functions $k(\cdot,\cdot)$ (or to its semantic part, $c(\cdot|\cdot)$).

It is easy to see that all the functions described in Table 1 are inclusion-preserving. It is also trivial that if the syntactic part of the cost function $c(\cdot)$ is constant, then if the semantic (fitness) part $c(\cdot|\cdot)$ is inclusion-preserving the whole cost function k will be inclusion-preserving.

We can now finally introduce the definition of minimal distance-based generalisation operator and minimal distance-based generalisation operator for a nerve function.

Definition 19. (Minimal distance-based generalisation operator for one nerve function N) *Let (X,d) be a metric space, N a nerve function, and Δ a dbg operator for N defined in X using a pattern language $\mathcal{L}$. Given a cost function, k, we will say that Δ is a minimal distance-based generalisation (mdbg) operator if for every dbg operator Δ' for N,*

$$k(E,\Delta(E)) \leq k(E,\Delta'(E)), \textit{for every finite set } E \subset X. \tag{1}$$

If no nerve function is set beforehand, minimality can still be established. In this case, it is sufficient to find just one nerve for each generalisation which follows equation (1). Figure 9 shows several patterns using the minimum distance nerve.

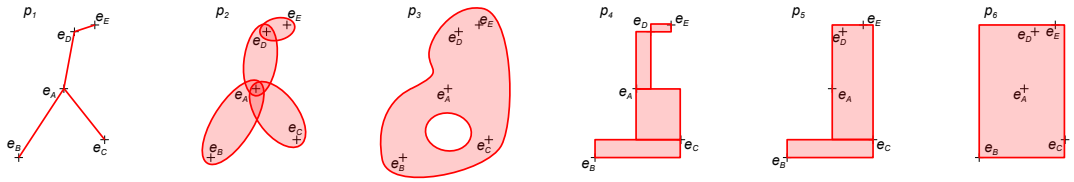


Figure 9: Patterns p_1, p_2 and p_3 are *db* with the Euclidean distance and the minimum distance nerve. Assuming the syntactic cost function equal, p_1 is the *mdbg* (since it matches the skeleton). Patterns p_2 and p_3 are not included in each other, but are comparable. In fact, $k(E,p_2) < k(E,p_3)$ for the semantic cost functions in Table 1. Patterns p_4, p_5 and p_6 are *db* with the Manhattan distance and the minimum distance nerve. Considering the syntactic cost function as the number of rectangles, and assuming functions in Table 1 are normalised, p_6 has lower cost than p_4 and p_5 for functions c_{inf}, c_{sup}, c_{min} (but it is not the *mdbg*), while p_5 has lower cost than p_4 and p_6 for function c_{max} (but it is not the *mdbg*). For the four semantic functions and this syntactic function, the *mdbg* may not exist (not unique).

The notion of *mdbg* says nothing about how to compute the *mdbg* operator, and, as we will see later, this might be difficult. A way to proceed is to first try to simplify the optimisation problem as much as possible. A first attempt might be the following:

Definition 20. (Naive generalisation operator) *Let (X,d) be a metric space and k a cost function. The naive generalisation operator Δ^0 is defined for every set $E \subset X$ as follows:*

$$\Delta^0(E) = argmin_{p \in \mathcal{L}: E \subset Set(p)}\ k(E,p)$$

The naive generalisation operator returns the simplest pattern (in terms of k) which covers the evidence. Δ^0 can be computed easily in most pattern languages $\mathcal{L}$ if k is inclusion-preserving. However, the resulting generalisation might not be *db*. Another attempt is the following:

Definition 21. (Skeleton generalisation operator for a nerve function N) *Let (X, d) be a metric space, N a nerve function and k a cost function. The skeleton generalisation operator $\bar{\Delta}_N$ is defined for every set $E \subset X$ as follows:*

$$\bar{\Delta}_N(E) = argmin_{p \in \mathcal{L}: skeleton(N(E)) = Set(p)}\ k(E, p)$$

which means the simplest pattern that covers the skeleton of the evidence (given a nerve) and nothing more. Clearly, it is a *dbg* operator because it includes the skeleton, but it might not exist because it cannot be expressed. Note that if the language $\mathcal{L}$ has universal expressiveness, this is related to the notion of convex hull seen in section 3.4. From here:

Proposition 3. *Given a metric space (X, d), a set $E \subset X$, a nerve function N, a pattern language $\mathcal{L}$ that can express any skeleton$(N(E))$ and a cost function k. If k is inclusion-preserving and $\bar{\Delta}_N$ exists, then $\bar{\Delta}_N$ is a mdbg operator for N and $(\mathcal{L}, k)$.*

Proof. Since $\bar{\Delta}_N$ exists, we can set $p = \bar{\Delta}_N(E)$. For any other *db* pattern p' of E, by definition of $\bar{\Delta}_N$, we necessarily have $Set(p) \subset Set(p')$. Since $\bar{\Delta}_N$ is *db* and k is inclusion-preserving then $\bar{\Delta}_N$ is a *mdbg* operator for N. □

For some datatypes (such as nominal, numerical or first-order objects as shown in the following sections), the techniques introduced by definition 21 and proposition 3 to obtain a minimal distance-based generalisation operator will be sufficient. However, this will not be applicable in many other cases, either because the pattern language is not expressive enough or because of the complexity of the datatypes.

5 Nominal Data

Given the discussion and definitions introduced in sections 3 and 4, it is time to apply the setting to several datatypes. We start with the simplest datatype, the nominal or categorical datatype. Along with numerical datatypes, this is one of the main components of what is called propositional learning, i.e., flat data that is expressed in terms of attributes and instances.

Nominal (categorical) attributes can express a set of possibilities (e.g. colours, qualities, genders, etc.). Therefore, in general, the metric space here is composed of a set X which is just a finite set of values and a distance d. Although the datatype is very simple, many distances can be defined on it. Some of the most-commonly used distances are the discrete (overlapping) 0-1 distance (which returns 1 when both values match and 0 otherwise) and the VDM (Value Difference Metric) distance [44], although there are many others. Distances over nominal data can also be defined from a relation order previously defined over the set of nominal data.

5.1 Extensional pattern language and simple cost functions

In many applications, nominal attributes are used in patterns in the form of conditions such as "x_1 = black" or "$x_1 \neq$ black". All of them can be expressed as "$x_1 \in$ S", where S is any subset of X since X is finite. So, formally, the pattern language $\mathcal{L}^{nom}$ is defined by the set 2^X.

Once the pattern language is fixed, we use two cost functions $k_0(E, p)$ and $k_1(E, p)$, with $k_0 = c_0(p) + c(E|p)$ (where $c_0(p) = 0$ and $c(E|p)$ is any inclusion-preserving cost function) and $k_1 = c_1(p) + c(E|p)$ where $c_1(p) = |Set(p)|$, also making k_1 inclusion-preserving. From here:

Proposition 4. *Given a set of values X, any distance d, any nerve function N, the pattern language $\mathcal{L}^{nom} = 2^X$, and any of the cost functions k_0 and k_1, then $\bar{\Delta}_N$ exists and is the mdbg operator for N.*

Proof. For every $E \subset X$, there is only one way to express $skeleton(N(E))$. Consequently, $\bar{\Delta}_N$ exists. In addition, as k_0 and k_1 are inclusion-preserving and $skeleton(N(E))$ can be expressed in $\mathcal{L}^{nom}$, then, by proposition 3, $\bar{\Delta}_N$ is the *mdbg* operator for N. □

It is also easy to see that if the distance is the discrete distance (1 when both values match and 0 otherwise), then $\bar{\Delta}_N(E) = E$ and thus $\bar{\Delta}_N = \Delta^0$.

Example 3. *Let $X = \{vhigh, high, medium, low, vlow\}$ be an ordered set such that the distance between two elements is the absolute difference of their rank. For instance, $d(vhigh, high) = 1$ and $d(vhigh, vlow) = 4$. Given $E = \{vhigh, medium, low\}$ and any possible nerve for these elements, the mdbg is the pattern computed by $\bar{\Delta}_N$, which, in this case, is $\bar{\Delta}_N(E) = \{vhigh, high, medium, low\}$. Note that Δ^0, although minimal, is not distance-based.*

5.2 Hierarchical pattern language and a simple cost function

Example 3 defines a distance from a total order relation. However, we can work on cases where there is a partial order relation. Consider, for example, the case where there is a hierarchy of elements, such that $x\,R\,y$ if $x\,is\_a\,y$. `Fish` R `Vertebrate` since a `Fish` is_a `Vertebrate`. Figure 10 shows a tree hierarchy where elements that are directly connected are at a distance 1 and the rest are the sum of distances of the shortest path that connects them.

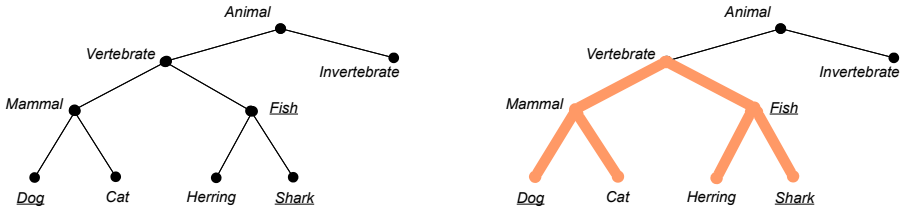


Figure 10: A set of nominal elements with a hierarchical relation from which we can infer a distance. The underlined values represent a possible evidence E and, on the right, its skeleton.

The pattern language $\mathcal{L}^{hnom}$ can be defined as $\{is\_a(v)\}$ where $v \in X$, which means covering any element w such that $w\,R\,v$. Given figure 10, we can say that the pattern $is\_a($`Vertebrate`$)$ covers the elements `Vertebrate`, `Mammal`, `Dog`, `Cat`, `Fish`, `Herring` and `Shark`.

Once the pattern language is fixed, we use the cost function k_0. Under these conditions:

Proposition 5. *Consider a set of values X, a distance d induced by a partial order relation R over X such that $\langle E, R\rangle$ is a tree, a nerve function N, the pattern language $\mathcal{L}^{hnom}$ and the cost function k_0. $\Delta(E) = is\_a(u)$ is the mdbg for $(\mathcal{L}^{hnom}, k_0)$ and N, where u is the minimum upper bound of E.*

Proof. It is easy to show that $\Delta(E) = \bar{\Delta}_N(E)$ (see that $skeleton(N(E)) = Set(is\_a(u))$). This minimum upper bound exists because $\langle E, R\rangle$ is a tree. Hence, any $skeleton(N(E))$ can be expressed in $\mathcal{L}^{hnom}$ and, given that k_0 is inclusion-preserving, by proposition 3, Δ is the *mdbg* for N. □

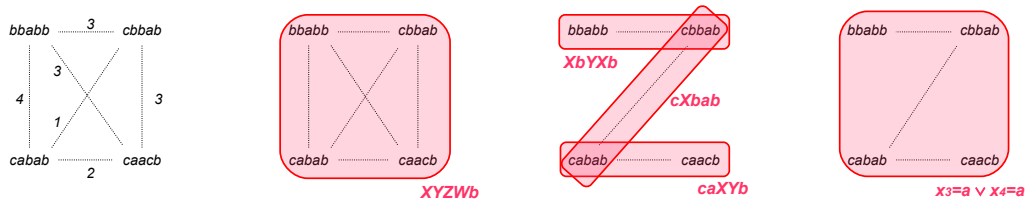
In the example shown in figure 10, if we have the evidence $E = \{\texttt{Dog}, \texttt{Fish}, \texttt{Shark}\}$, the generalisation operator defined in proposition 5 would compute $is\_a(\texttt{vertebrate})$, which clearly covers the intermediate elements, which are `mammal` and `vertebrate`, and is also minimal.

Many other kinds of languages, distances and cost functions might be analysed for nominal datatypes, but as the previous examples show, the *mdbg* operator is easy to find in these cases, possesing an intuitive behaviour in terms of the underlying distance.

5.3 Multidimensional nominal data

Tuples (or vectors) of nominal data are custom in propositional learning. One common distance is the Hamming distance, which is just defined as the sum of the individual distances.

Consider, e.g., a 5-dimensional space X with features $(x_1, x_2, x_3, x_4, x_5)$ and a set E with examples $e_1 = (b, b, a, b, b)$, $e_2 = (c, a, b, a, b)$, $e_3 = (c, b, b, a, b)$ and $e_4 = (a, a, a, c, b)$, as shown in figure 11. If we consider the Hamming distance, we can see that different pattern languages and generalisations are *mdbg*, while others are not.



(a) (b) (c) (d)

Figure 11: Patterns for a set of nominal vectors using the Hamming distance. **(a)** The elements and their distances. **(b)** A complete nerve and the mdbg (X,Y,Z,W,b). **(c)** A minimal-distance nerve and the mdbg (X,b,Y,X,b) ∨ (c,X,b,a,b) ∨ (c,a,X,Y,b). **(d)** A generalisation which is not *db*. It does not include (c,b,b,b,b), which is in between elements (b,b,a,b,b) and (c,b,b,a,b).

In fact, conjunctions of one-value conditions are always *db* for a pair of elements, and their disjunction following a nerve are *db* for the whole set (as case (c) in figure 11). This type of conjunctions of one-value conditions is exactly what we find in some conceptual clustering algorithms and many frequent itemset mining algorithms which search for association rules. In addition, if we restrict to Boolean attributes we have a well-studied relation between the type of formulas, their expressiveness and their features in the area of formal concept analysis. We have not included (trivial) proofs for these languages and distance since, as we will show, these are special cases of the results in section 9.2, when composing base distances and using the Manhattan (or Hamming) distance as in this case.

Nominal data, either individually or in the form of a tuple has generally been treated with patterns and distances which are compatible (although in some cases without being conscious of this compatibility). We will address more interesting (and complex) datatypes in subsequent sections. Next, we see another straightforward case, numerical data.

6 Numerical Data

Real numbers in $\mathbb{R}$ with the distance defined as the absolute difference of two real numbers ($d(e_i, e_j) = |e_i - e_j|$) is a metric space. For vectors of real numbers, i.e. $\mathbb{R}^n$, we have many aggregated distance functions which also lead to metric spaces.

In this section, we first analyse the well-known concept of mathematical interval (as usual in interval analysis [35] and interval computing [31]) in $\mathcal{R}$ as a pattern and, next, we deal with some other concepts such as rectangles and circles in $\mathbb{R}^n$, showing how close the notions of *mdbg* and traditional ideas such as convexity are in this simple data type.

6.1 $\mathbb{R}$: Single interval pattern language

The first pattern language $\mathcal{L}_0^{num}$ we consider is just the set of all the finite closed intervals in $\mathbb{R}$. We use two cost functions: $k_0(E, p) = c_0(p) + c(E|p)$ and $k_1(E, p) = c_1(p) + c(E|p)$, where $c_0(p) = 0$, $c_1(p) = |b - a|$ being $p = [a, b]$ and where $c(E|p)$ is any inclusion-preserving function. Note that k_1 is also inclusion-preserving. We can easily show that intervals are *mdbg* operators.

Proposition 6. *For every finite set of elements $E = \{e_1, \cdots, e_n\}$ such that $e_i < e_{i+1}$ $(1 \leq i \leq (n-1))$, the mapping $\Delta(E) = [e_1, e_n]$ is a mdbg operator for $(\mathcal{L}_0^{num}, k_0)$ and $(\mathcal{L}_0^{num}, k_1)$.*

Proof. It is easy to see that $skeleton(N(E)) = [e_1, e_n]$ can always be expressed in $\mathcal{L}_0^{num}$ and $\Delta(E) = \bar{\Delta}_N(E)$. Since both k_0 and k_1 are inclusion-preserving, then by proposition 3, Δ is the *mdbg* operator $(\mathcal{L}_0^{num}, k_i)(i = 0, 1)$ and N. □

Intervals are common in some pattern languages in artificial intelligence. For instance, decision trees use conditions that are expressed as the membership to an interval, which is usually expressed by a splitpoint a (so leading to two intervals, $x \leq a$ and $x > a$). Clearly, for infinite intervals, we should use c_0 (or other functions leading to non-infinite values).

6.2 $\mathbb{R}$: Multiple interval pattern language

However, things are more complex when using a more expressive pattern language than $\mathcal{L}_0^{num}$.

We define $\mathcal{L}_1^{num}$ as the pattern language of multiple intervals in which a pattern is expressed as the union of intervals (i.e., the operator + seen in definition 3). For instance, given the evidence $E = \{3, 4, 7, 8, 15, 16, 18, 20\}$, a pattern p covering E could be $[3, 8] \cup [15, 20]$. Working with several intervals is useful in, e.g., attribute discretisation, numerical partitions in decision-tree learning and numerical clustering. The naive generalisation operator for $\mathcal{L}_1^{num}$ and the cost function k_0 will be $\Delta^0(E) = \bigcup_{e \in E}[e, e]$. For the previous evidence, $\Delta^0(E) = [3, 3] \cup [4, 4] \cup [7, 7] \cup [8, 8] \cup [15, 15] \cup [16, 16] \cup [18, 18] \cup [20, 20]$. Trivially, Δ^0 is minimal for k_0 and k_1 since, for any other pattern p covering E, $Set(p)$ contains $Set(\Delta^0(E))$ because $Set(\Delta^0(E)) = E$. Given that k_0 and k_1 are inclusion-preserving, by definition 18, Δ^0 is minimal for them.

However, Δ^0 is not distance-based since for any nerve η of E, $skeleton(\eta) \not\subset set(\Delta^0(E))$. Moreover, Δ^0 is not minimal for many other cost functions. Let $k_2(E, p) = c_2(p) + c_2(E|p)$ be the cost function where $c_2(p) = n^2$ with n the number of intervals in p and $c_2(E|p)$ is the c_{min} function in Table 1. Note that Δ^0 is not minimal for k_2. For instance, continuing with the example, the pattern $p_1 = [3, 8] \cup [15, 20]$ has a lower cost than $\Delta^0(E)$ since $k_2(E, \Delta^0(E)) = 64 + 0 = 64$ and $k_2(E, p_1) = 4 + 5 = 9$. In fact, for this example and cost function, p_1 also has a lower cost than the pattern $p_2 = [3, 20]$ ($k_2(E, p_2) = 1 + 21 = 22$).

6.3 $\mathbb{R}^n$: Hyper-rectangle pattern language

Many of the examples in previous sections have been shown in $\mathbb{R}^2$, since the patterns and examples can be graphically shown. In general, however, we may work with more dimensions, with several distances and pattern languages. It is clearly different to work with a Euclidean distance with hyper-rectangles than using a Manhattan distance with ellipses.

Among all the possible pattern languages, we start with axis-parallel hyper-rectangles which can be considered as the most straigthforward generalisation of intervals in $\mathbb{R}^n$, and can be seen just as interval vectors, used in conjuctions of conditions of rule-based systems and decision trees (e.g. $2 \leq x_1 < 10 \wedge 1 \leq x_2 < 5$). Axis-parallel hyper-rectangles (or interval vectors) are common in some algorithms [24] and pattern languages, especially in interval computing (e.g., the interval rule matrices approach [28] for classification).

Let us define the pattern language $\mathcal{L}_0^{rect}$ as the set of all axis-parallel hyper-rectangles in $\mathbb{R}^n$. We use two cost functions: $k_0(E,p) = c_0(p) + c(E|p)$ where $c_0(p) = 0$ and $k_1(E,p) = c_1(p) + c(E|p)$, where $c_1(p) = Vol(p)$ with $Vol(p)$ being the length of the interval in $\mathbb{R}^1$, the area in $\mathbb{R}^2$ and the volume for $n > 2$. In both cases, $c(E|p)$ is any inclusion-preserving function.

The following proposition shows that the smallest axis-parallel hyper-rectangle is an *mdbg* operator for the Manhattan and Euclidean distances.

Proposition 7. *For every finite set of elements $E = \{e_1, \cdots, e_n\}$, the mapping $\Delta(E)$ which calculates the smallest axis-parallel hyper-rectangle covering E is a mdbg operator for $(\mathcal{L}_0^{rect}, k_0)$ and $(\mathcal{L}_0^{rect}, k_1)$ for the Manhattan and Euclidean distances for every nerve.*

Proof. It is easy to see that for these two distances the $skeleton(N(E))$ (for every nerve N) is always a subset of $set(\Delta(E))$, so it is a *dbg* operator. Since both k_0 and k_1 are inclusion-preserving, and $\Delta(E)$ computes the smallest axis-parallel rectangle and this is unique, then Δ is the *mdbg* for $(\mathcal{L}_0^{rect}, k_0)$ and $(\mathcal{L}_0^{rect}, k_1)$. □

This is also commonly referred to as the minimum (or smallest) bounding or enclosing box in computer graphics. It is also used in clustering and classification when using rules (covering algorithms) or support vector machines [7]. This is exactly what we saw in the pattern (a) in figure 5. Additionally, intervals are a special case of this (since the Manhattan distance and the Euclidean distance match for one dimension). This shows that axis-parallel rectangles are consistent patterns with the Euclidean and Manhattan distances, which is in accordance to the common use of these distances in clustering in conjuction with decision rules, decision trees and other models based on axis-parallel conditions (univariate conditions).

There are many other variants using rectangles that could be explored, such as the union of hyperrectangles, or the consideration of oblique rectangles, i.e., non-axis parallel rectangles.

6.4 $\mathbb{R}^n$: sphere pattern language

Hyperspheres are an alternative generalisation of intervals in $\mathbb{R}^n$. In fact, any closed shape is a generalisation of intervals in $\mathbb{R}^n$. It is interesting to note that hyperspheres are balls[6] centered on a point by using the Euclidean distance. As demonstrated above, the previous section and this one link the concept of *mdbg* to topological notions and concepts in $\mathbb{R}^n$.

[6] The concept of ball can be defined for metric spaces in general and not only for the Euclidean space.

Let us formally define the pattern language $\mathcal{L}_0^{sphere}$ as the set of all hyper-spheres in $\mathbb{R}^n$. The same cost functions as in section 6.3 are used.

The generalisation calculating the smallest sphere is referred to as the "minimal enclosing sphere' or the "minimal bounding sphere" [32]. This sphere exists and is unique for $\mathbb{R}^n$. Finding this sphere for a set of points in $\mathbb{R}^2$ is known as the 'bomb problem' or 'smallest circle problem'. This problem has many applications in computer graphics for $n = 2$ and $n = 3$, and is also common in clustering, since the centre can be used as a prototype for the cluster. The notion of minimal enclosing sphere is especially successful in support vector clustering [5], where "data points are mapped by means of a kernel to a high dimensional feature space, where we search for the minimal enclosing sphere", along with the minimal enclosing hyper-rectangle seen above.

The following proposition shows that minimal spheres are *mdbg* for the Euclidean distance.

Proposition 8. *For every finite set of elements $E = \{e_1, \cdots, e_n\}$, the mapping $\Delta(E)$ which calculates the smallest hypersphere covering E is a mdbg operator for $(\mathcal{L}_0^{sphere}, k_0)$ and $(\mathcal{L}_0^{sphere}, k_1)$ for every nerve using the Euclidean distance in $\mathbb{R}^n$.*

Proof. For the Euclidean distance the $skeleton(N(E))$ (for every nerve N) is always a subset of $set(\Delta(E))$ since spheres are convex and $set(\Delta(E))$ contains all the elements in E, so it is a dbg operator. Since both k_0 and k_1 are inclusion-preserving, and $\Delta(E)$ computes the smallest hypersphere and this is unique then Δ is the *mdbg* for $(\mathcal{L}_0^{sphere}, k_0)$ and $(\mathcal{L}_0^{sphere}, k_1)$. □

We found this in pattern (b) in figure 5. Also, for two or more dimensions, smallest enclosing spheres are not *mdbg* for the Manhattan and Chebyshev distances. This shows that spheres are only consistent with the Euclidean distance, as they are balls derived from this distance.

Finally, it would be worth investigating the results for the union of spheres a well as the use of hyper-ellipses. This is left for future work. Similarly, other (convex) bounding volumes would also be of interest, as well as other non-convex patterns.

7 Sets

Sets are quite common in structured learning. Many structures can be directly modelled by means of sets of words (or multi-sets if repeated elements are taken into account). Besides, other data types such as trees or graphs can also be indirectly represented using sets, e.g., in the representation of phylogenetic trees derived from DNA data where trees are viewed as sets of labelled edges. In this section, we will define distance-based generalisation operators for some set pattern languages, cost functions and distances.

7.1 Distance functions for sets

There are several distances for sets, depending on whether we consider finite or infinite sets, and on whether we have an underlying distance for the elements in the set. One distance which works well for infinite sets, but requires an underlying distance d is the Hausdorff distance:

$$d_H(e_1, e_2) = max\{sup_{r_1 \in e_1} inf_{r_2 \in e_2} d(r_1, r_2), sup_{r_2 \in e_2} inf_{r_1 \in e_1} d(r_1, r_2)\} \quad (2)$$

If the underlying distance is the 0-1 distance ($d(x, y) = 1$ if $x \neq y$, 0 otherwise) then this definition boils down to 1 if the sets are different and 0 if the sets are equal. However, in other

cases, it leads to interesting distances, as when the sets are intervals and the underlying distance d is the absolute difference. In this case, the Hausdorff distance is a somewhat awkward distance between intervals, the maximum of the distance between the left limits and the distance between the right limits (e.g., $d([3,5],[8,15]) = 10$). This can be generalised to $\mathbb{R}^n$.

Another common distance for sets is the size of the symmetric difference between two sets, i.e., $d_S(e_1, e_2) = |(e_1 - e_2) \cup (e_2 - e_1)|$. This distance only works when we assume the sets to be finite. In this way, the distance between two sets depends on the number of elements that they do not have in common. This metric function somewhat assumes that the 0-1 distance underneath. Normally, the symmetric difference is employed when the elements involved are viewed as indivisible objects. For instance, given the following sets of sequences $e_1 = \{ab, a^4\}$, $e_2 = \{ab, d^4\}$ and $e_3 = \{ab, a^3\}$, then $d_S(e_1, e_2) = d_S(e_1, e_3) = d_S(e_2, e_3) = 2$. Note that this is different to the Haussdorf distance using the edit distance underneath, which would lead to $d_H(e_1, e_2) = 8$, $d_H(e_2, e_3) = 7$ but $d_H(e_1, e_3) = 1$. There are other similarity [13] and distance [40] functions for sets. The use of distances for sets which re-use the underlying distance will be discussed again in section 9. Given the discussion above, in the rest of this section, we will just work with the symmetric difference, and we will drop the subscript.

7.2 Pattern languages and cost functions

The space X is composed of elements in $A = \{a_1, a_2, a_3, \ldots\}$, the alphabet of ground symbols, i.e. $X = 2^A$. Two different pattern languages are considered $\mathcal{L}_0^{set}$ and $\mathcal{L}_1^{set}$. The patterns in $\mathcal{L}_0^{set}$ are sets from the alphabet $\Sigma = A \cup V$ where $V = \{V_1, V_2, \ldots\}$ is a set of variables. An element e is covered by a pattern p if a substitution σ exists over the variables in p, such that $e = \sigma(p)$. Consequently, no pattern (except from $\emptyset$) covers the empty set. For example, $p = \{a_1, a_2, a_3, V_1, V_2, V_3\}$ covers $\{a_1, a_2, a_3, a_4\}$ with (among others) $\sigma = \{V_1/a_3, V_2/a_3, V_3/a_4\}$. The function $Gr(p)$ defined over a pattern $p \in \mathcal{L}_0^{set}$ returns the set of all the ground symbols in p. For example, $Gr(p) = \{a_1, a_2, a_3\}$. Finally, the pattern language $\mathcal{L}_1^{set}$ is obtained from $\mathcal{L}_0^{set}$ by means of the operation $+$ (see definition 1 in section 3).

Example 4. *Given $A = \{a_1, a_2, a_3, a_4\}$ and the patterns $p_1 = \{a_1, a_2, V_1\}$, $p_2 = \{a_3, V_1, V_2\}$ and $p_3 = p_1 + p_2$, then p_1 represents all the sets whose cardinality is 2 or 3 and contain the elements a_1 and a_2. Similarly, p_2 represents all the sets with cardinality between 1 and 3 and contain the element a_3. Finally, p_3 represents all the sets covered by the patterns p_1 and p_2.*

Let us define the cost functions $k_0(E, p) = c_0(p) + c(E|p)$ and $k_1(E, p) = c_1(p) + c(E|p)$ where $c_0(p) = 0$, $c_1(p) = |p|$ and $c(E|p)$ is the function c_{inf} in Table 1. Note that k_0 is inclusion-preserving.

Example 5. *Let $E = \{e_1, e_2, e_3\}$ where $e_1 = \{a_1, a_2\}$, $e_2 = \{a_2, a_3, a_4\}$ and $e_3 = \{a_3, a_4, a_5\}$ are the elements to be generalised. Given the following patterns generalising E in $\mathcal{L}_1^{set}$:*

$$p_1 = \{V_1, V_2, V_3\} \qquad p_2 = \{a_2, V_1, V_2\} + \{a_3, a_4, V_1\}$$

Let us calculate $k_0(E, \cdot)$ and $k_1(E, \cdot)$ for all the patterns above. To this end, let us denote by e'_1, e'_2 and e'_3 the nearest elements to e_1, e_2 and e_3, respectively, which are not covered by a given pattern. The table 2 collects the possible instances for e'_1, e'_2 and e'_3 for a concrete pattern.

Computation of cost functions				k_0	k_1
p_1	$e'_1 = \{a_1, a_2, a_3, a_4\}$ $e'_2 = e'_1$ $e'_3 = \{a_1, a_3, a_4, a_5\}$	$c(\{e_1\}\|p_1) = d(e_1, e'_1) = 2$ $c(\{e_2\}\|p_1) = d(e_2, e'_2) = 1$ $c(\{e_3\}\|p_1) = d(e_3, e'_3) = 1$	$c_1(p_1) = 3$	4	7
p_2	$e'_1 = \{a_1\}$ $e'_2 = \{a_1, a_2, a_3, a_4\}$ $e'_3 = \{a_3, a_5\}$	$c(\{e_1\}\|p_2) = d(e_1, e'_1) = 1$ $c(\{e_2\}\|p_2) = d(e_2, e'_2) = 1$ $c(\{e_3\}\|p_2) = d(e_3, e'_3) = 1$	$c_1(p_2) = 6$	3	9

Table 2: Computation of the cost functions k_0 and k_1 for patterns p_1 and p_2.

7.3 Single set pattern language ($\mathcal{L}_0^{set}$)

First, we define a particular generalisation operator over $\mathcal{L}_0^{set}$ called $\uparrow$-transformation. This is just a bottom-up operator that permits us to move through $\mathcal{L}_0^{set}$.

Definition 22. ***($\uparrow$-transformation)*** *Given a finite set of patterns $\{p_1, \ldots, p_n\} \in \mathcal{L}_0^{set}$, we define the mapping $\uparrow\colon 2^{\mathcal{L}_0^{set}} \to \mathcal{L}_0^{set}$ as $\uparrow(\{p_1, \ldots, p_n\}) = \{a_1, \ldots, a_m, V_1, \ldots, V_q\}$*

$$\textit{such that } \begin{cases} (1) & G = \{a_1, \ldots, a_m\} = \cap_{i=1}^{n} Gr(p_i) \\ (2) & q = max\{|p_i| : \forall 1 \leq i \leq n\} - |G| \end{cases}$$

It follows from the definition that $\uparrow(\{p\}) = p$. It is clearly a generalisation since given a collection of patterns $p_1, \ldots, p_n$, the $\uparrow$-transformation returns a pattern p such that $Set(p_i) \subset Set(p)$, for every pattern p_i. However, this generalisation concerns the syntactic aspects of the patterns while it disregards the underlying distance.

The $\uparrow$-transformation will provide us with the intuition for the definition of *dbg* operators. The reasoning is straightforward. Given two sets e_1 and e_2, if a pattern $p \in \mathcal{L}_0^{set}$ generalises e_1 and e_2, necessarily $Gr(p)$ must be a subset of e_1 and e_2, and, additionally, $|p| \geq max\{|e_1|, |e_2|\}$. Therefore, to ensure that p is based on distances, for any element e_3 between e_1 and e_2, e_3 must contain $Gr(p)$ and p has to have as many variables as needed to cover e_3. This is what the next lemma and propositions below state focusing first on binary *dbg* operators.

Lemma 1. *Let e_1, e_2 and e_3 be finite sets. If e_3 is between e_1 and e_2 then $|e_3| \leq |e_1 \cup e_2|$.*

Proof. We will proceed by contradiction. Let us suppose that $|e_3| > |e_1 \cup e_2|$. Then:

$$\begin{aligned} d(e_1, e_3) + d(e_3, e_2) &= |e_1 - (e_1 \cap e_3)| + |e_3 - (e_1 \cap e_3)| + |e_3 - (e_3 \cap e_2)| + |e_2 - (e_3 \cap e_2)| \\ &= |e_3 - e_1| + |e_3 - e_2| \;>\; |(e_1 \cup e_2) - e_1| + |(e_1 \cup e_2) - e_2| \\ &= d(e_1, e) + d(e, e_2) = d(e_1, e_2) \end{aligned}$$

Hence, $\forall$ e_3 between e_1 and $e_2 \Rightarrow |e_3| \leq |e_1 \cup e_2|$ □

Proposition 9. ***(Binary distance-based generalisation operator for sets)*** *A binary generalisation operator Δ^b is a dbg operator for (X, d) and $\mathcal{L}_0^{set}$ iff for every pair of elements e_1 and e_2, $\Delta^b(e_1, e_2) = p \in \mathcal{L}_0^{set}$, $Gr(p) \subset e_1 \cap e_2$ and $|p| \geq |e_1 \cup e_2|$.*

Proof. ($\Rightarrow$) As Δ^b is a generalisation operator then necessarily $Gr(p) \subset e_1 \cap e_2$. Otherwise, e_1 or e_2 would not be covered by p. Working as in lemma 1, we see that the element $e_3 = e_1 \cup e_2$ is between e_1 and e_2. Since p is a distance-based pattern of e_1 and e_2, $e_3 \in Set(p)$ and necessarily the number of symbols in p must be enough to cover $e_1 \cup e_2$, then $|p| \geq |e_1 \cup e_2|$.

($\Leftarrow$) For every three elements e_1, e_2 and e_3, if e_3 lies between e_1 and e_2 then $e_1 \cap e_2 \subset e_3$ (see proposition 3 in [16]). Therefore, $Gr(p) \subset e_1 \cap e_2 \subset e_3$ Combining this and lemma 1 we have that for every element e_3 between e_1 and e_2, then e_3 is covered by $\Delta^b(e_1, e_2)$ and the generalisation is *db*. □

In $\mathcal{L}_0^{set}$, the difference between a pattern p computed by a *dbg* operator and a pattern p' computed by any other generalisation operator (e.g. the $\uparrow$-transformation) relies on their cardinality, since more variables are needed in the *db* patterns to capture the intermediate elements.

Example 6. *Given the elements* $e_1 = \{a_1, a_2\}$ *and* $e_2 = \{a_1, a_3\}$ *and the patterns* $p_1 = \{a_1, V_1\}$, $p_2 = \{V_1, V_2\}$ *and* $p_3 = \{a_1, V_1, V_2\}$. *Pattern* p_3 *is db, but* p_1 *and* p_2 *are not since the element* $\{a_1, a_2, a_3\}$ *is between* e_1 *and* e_2 *and is not covered by either* p_1 *or* p_2.

The characterisation of a binary *dbg* operator has an immediate extension for n-ary *dbg* operators if we previously fix a nerve function N.

Proposition 10. ***(n-ary distance-based generalisation operator for sets for a nerve function N)*** *A generalisation operator* Δ *is a dbg operator* (X, d), $\mathcal{L}_0^{set}$ *and a nerve function* N *iff for every finite set of elements* $E = \{e_1, \ldots, e_n\}$ *such that* $\Delta(E) = p \in \mathcal{L}_0^{set}$, $Gr(p) \subset \cap_{i=1}^n e_i$ *and* $|p| \geq max\{|e_i \cup e_j| : \forall (e_i, e_j) \in N(E)\}$.

Proof. ($\Leftarrow$) Given any (e_i, e_j) in $N(E)$ and let e be an element between e_i and e_j. From proposition 3 in [16], we can write that $e_i \cap e_j \subset e$. Hence,

$$Gr(p) \ \subset \cap_{l=1}^n e_l \ \subset e_i \cap e_j \ \subset \ e \tag{3}$$

Additionally, by lemma 1 we can affirm that $|e| \leq |e_i \cup e_j|$. Hence,

$$|e| \leq |e_i \cup e_j| \leq max\{|e_i \cup e_j| : \forall (e_i, e_j) \in N(E)\} = |p| \tag{4}$$

Therefore, by combining (3) and (4), $e \in Set(p)$ and Δ is *db*. ($\Rightarrow$) As $\Delta(E) = p$ is a generalisation of E then $E \subset Set(p)$ and necessarily for every $e_i \in E$, $Gr(p) \subset e_i$. In addition, for every $(e_i, e_j) \in N(E)$, we define $\Delta^b(e_i, e_j) = \Delta(E) = p$. Now, from proposition 9, we can write $\forall (e_i, e_j) \in N(E)$, $|p| \geq |e_i \cup e_j|$ which means $|p| \geq max\{|e_i \cup e_j| : \forall (e_i, e_j) \in N(E)\}$.

□

The relationship between binary and n-ary *dbg* operators is given below.

Proposition 11. *Let* Δ *be an* n*-ary dbg operator for a nerve function* N. *Then there exists a binary dbg operator* Δ^b *such that for every finite set* E

$$\Delta(E) = \uparrow (\{\Delta^b(e_i, e_j) : \forall (e_i, e_j) \in N(E)\})$$

Proof. It directly follows from the definition of the $\uparrow$-transformation and proposition 9. □

After characterising the family of *dbg* operators for sets, we will study those that are minimal. Basically, the condition of being a *mdbg* operator depends on the number of variables in the pattern computed by the operator. If $E = \{e_1, \ldots, e_n\}$ and $\cap_{i=1}^n e_i \neq \emptyset$, then the number of variables in $\Delta(E)$ is enough to be greater than $max\{|e_i \cup e_j| : \forall e_i, e_j \in E\}$ (see proposition 10). However, the number of variables required remains unclear when $\cap_{i=1}^n e_i = \emptyset$.

Now we can obtain the *mdbg* operator for a nerve function. This is shown below.

Lemma 2. *Given a finite set of elements $E = \{e_1, \ldots, e_n\}$, a nerve function N and a generalisation operator Δ. If for every set E, Δ satisfies the following conditions,*

1. $Gr(\Delta(E)) \subset \cap_{i=1}^{n} e_i$
2. $|\Delta(E)| = max\{|e_i \cup e_j| : \forall (e_i, e_j) \in N(E)\}$
3. *If* $\cap_{i=1}^{n} e_i = \emptyset$ *then* $Gr(\Delta(E)) = \emptyset$.

then Δ is a minimal distance-based generalisation operator for N.

Proof. From proposition 10, Δ is distance-based. We can see that Δ is also minimal. If $\cap_{i=1}^{n} e_i \neq \emptyset$ then, by condition 1, $Gr(\Delta(E)) \neq \emptyset$ and by definition of $c(E|p)$ for $\mathcal{L}_0^{set}$, $k_0(E, \Delta(E)) = |E|$ which is the lowest value the function $k_0(E, \cdot)$ can attain. On the contrary, if $\cap_{i=1}^{n} e_i = \emptyset$ then $\Delta(E)$ is only made up of variables, but according to proposition 10, $\Delta(E)$ contain the minimum number of variables required to be distance-based. Therefore, $\Delta(E)$ is minimal. □

Proposition 12. ***(Minimal distance-based generalisation operator for sets for a nerve function N)*** *Given a finite set of elements $E = \{e_1, \ldots, e_n\}$ and a nerve function N. Let Δ^b be a binary dbg operator that satisfies*

1. $\forall (e_i, e_j) \in N(E)$, $Gr(\Delta^b(e_i, e_j)) = e_i \cap e_j$.
2. $\forall (e_i, e_j) \in N(E)$, $|\Delta^b(e_i, e_j)| = |e_i \cup e_j|$

Then, the n-ary dbg operator Δ defined from Δ^b in proposition 11 is minimal for N.

Proof. From Condition 1 and by the definition of the $\uparrow$-transformation, we have that $Gr(\Delta(E)) = \cap_{i=1}^{n} e_i$. We also have that $Gr(\Delta(E)) = \emptyset$ if $\cap_{i=1}^{n} e_i = \emptyset$. Finally, from Condition 2 and again by definition of the $\uparrow$-transformation, we have that $|\Delta(E)| = max\{|e_i \cup e_j| : \forall (e_i, e_j) \in N(E)\}$. Therefore Δ satisfies lemma 2 and Δ is minimal for the nerve function N. □

Observe that the *mdbg* operator for a nerve function is not unique because nothing about the maximum number of variables required by the pattern is said when $\cap_{i=1}^{n} e_i \neq \emptyset$.

7.4 Multiple set pattern language ($\mathcal{L}_1^{set}$)

Now, we have to define *dbg* operators in $\mathcal{L}_1^{set}$. Of course, if an operator Δ is *db* in $\mathcal{L}_0^{set}$, then Δ is *db* in $\mathcal{L}_1^{set}$. But this is not practical because we are not taking advantage of $\mathcal{L}_1^{set}$. Proposition 1 establishes a more suitable way to do that. However, we do not know how to define Δ^b in $\mathcal{L}_1^{set}$ yet. Moreover, giving a characterisation of all the binary *dbg* operators in $\mathcal{L}_1^{set}$, as we did in $\mathcal{L}_0^{set}$, is not immediate due to the expressiveness of $\mathcal{L}_1^{set}$. A feasible solution for using proposition 1 consists in using binary operators defined in $\mathcal{L}_0^{set}$. See the example below.

Example 7. *Given $E = \{e_1, e_2, e_3, e_4, e_5\}$ where $e_1 = \{a_1, a_2, a_3, a_4, a_5\}$, $e_2 = \{a_1, a_2\}$, $e_3 = \{a_1, a_3\}$, $e_4 = \{a_3, a_4\}$ and $e_5 = \{a_3, a_5\}$. Let Δ^b be a binary dbg operator such that $\Delta^b(e_1, e_2) = \{a_1, a_2, V_1, V_2, V_3\}$, $\Delta^b(e_1, e_3) = \{a_1, a_3, V_1, V_2, V_3\}$, $\Delta^b(e_1, e_4) = \{a_3, a_4, V_1, V_2, V_3\}$, and $\Delta^b(e_1, e_5) = \{a_3, a_5, V_1, V_2, V_3\}$. Now, by applying proposition 1, for the nerve of figure 12 we have that*

$$\Delta_N(E) = \{a_1, a_2, V_1, V_1, V_3\} + \{a_1, a_3, V_1, V_2, V_3\} + \{a_3, a_4, V_1, V_2, V_3\} + \{a_3, a_5, V_1, V_2, V_3\}$$

is a db pattern for E.

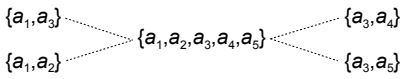


Figure 12: Fixing a nerve in order to define the operator $\Delta_N(E)$.

Regarding the computation of *mdbg* operators in this pattern language, we find that things are not as intuitive as in $(\mathcal{L}_0^{set}, k_0)$. We just outline the main points concerning this issue. First, the main difference with the previous case is that the *mdbg* operator for one nerve in $(\mathcal{L}_1^{set}, k_1)$ cannot be directly obtained from binary operators (as shown in proposition 2). Furthermore, we need to introduce a refinement operator for generalising patterns in $\mathcal{L}_1^{set}$. The *mdbg* operator is obtained when this refinement operator is applied over an optimal partition of the elements belonging to the skeleton of a set E for a nerve $N(E)$. Due to the nature of these operations, it seems that the computation of the *mdbg* operator in $(\mathcal{L}_1^{set}, k_1)$ cannot be carried out in feasible time. Hence, a reasonable approximation is a greedy search guided by the cost function (see [14] for details).

8 Other Data Types

Besides sets and tuples of nominal and numerical data, there are other data types, such as lists, first-order logic and graphs, which are widely used in machine learning. This section summarises the most important results achieved when applying our framework to these data types. We refer the reader to [17, 18, 15, 14] for a complete account of these results.

8.1 Lists

In [15] we define (minimal) distance-based generalisation operators for lists (sequences), using the edit distance, and two different pattern languages: $\mathcal{L}_0^{list}$ and $\mathcal{L}_1^{list}$. The first language is made up of patterns which consist of finite sequences of ground and variable symbols. The language $\mathcal{L}_1^{list}$ extends $\mathcal{L}_0^{list}$ in that the disjunction of patterns is permitted (see definition 3 in section 3). Additionally, we have defined a cost function for each language.

We have proved that for more than two sequences, the widely-used concept of *maximum common subsequence* does not necessarily lead to distance-based generalisation operators. In order to obtain this sort of operators, we need to introduce a new concept: the sequence associated to an optimal alignment. This kind of sequence leads to certain patterns that, when combined, allows us to define distance-based operators. As for the minimality of these operators, we have shown this is a computational hard problem in $\mathcal{L}_1^{list}$. For this reason, we have introduced a greedy search algorithm to approximate minimal generalisations.

8.2 First-order objects

Although great effort has been made in the field of inductive logic programming to establish the notion of generalisation on a formal basis, there is little work concerning the relationship between generalisation and distances over first-order objects (except the research done in [41] where a distance for atoms is obtained from *lgg*). We use our framework to precisely explore the connection between generalisation and distances for first-order atoms and clauses.

With respect to atoms, the main result is obtained under the following conditions: the distance is the one introduced in [41], the pattern language is the Herbrand Base with variables induced by a signature, and the cost function refers to any inclusion-preserving cost function. In this case, we can affirm that Plotkin's *lgg* operator for atoms is a n-ary *mdbg* operator. Moreover, both properties (*db* and *mdb*) are independent from the nerve function. However, this result does not necessarily hold for every cost function, pattern language or distance. This also suggests the possibility that other generalisation operators for atoms can be obtained by changing the cost function, which could be an alternative to redesigning existing ILP methods. With respect to clauses, we have used the distance defined in [40], the pattern language as the set of all the logic programs we can define given a signature, and k is any inclusion-preserving cost function. Under these conditions, it can be shown that Plotkin's *lgg* operator for clauses is not *db*. In [14], we introduce a *dbg* operator for clauses.

8.3 Graphs

Graphs can be embedded in a metric space by means of a distance we proposed in [17]. This distance consists of a slight variation of Bunke's distance published in [6]. The pattern language is made up of patterns denoted by $[g]$, covering all the graphs having the subgraph g. Finally, the cost function refers to any inclusion-preserving cost function. Under these conditions, we can state that every generalisation operator for graphs is a *dbg* operator. Thus, our distance allows us to consider frequent pattern mining algorithms for graphs as generalisation operators implying that the algorithms in this field can be used directly as implementations of *dbg* operators. This generic result does not hold when the original Bunke's distance is employed. In [14], we also show that a *dbg* operator that computes the maximum common subgraph is a *mdbg* operator, but other *mdbg* operators may exist.

9 Composability

Complex structures are constructed by aggregating simpler structures. A top-down analysis makes details visible which otherwise are unaccessible at a coarse level. A bottom-up analysis produces more abstract features and concepts which make the whole object more meaningful. Granular computing [4] is an umbrella term for many techniques and areas that deal with this whole-part relationship. The composability in metric spaces is one way to address this process.

A composite distance is a distance which is constructed as a function of a base distance over elements or parts of the objects. It is this notion of 'elements' or 'parts', which makes a difference on whether we calculate distances over collections of elements (horizontal composability) or whether we calculate distances over collections of features (vertical composability). This leads to different views of composability, which are explored below.

9.1 Types of composability

One of the goals of a generalisation or pattern is to group a collection of elements. This pattern, however, can also be treated as an element. For instance, intervals are sets of real numbers, but they can also be treated as elements. If the elements at the lower level have a distance, we can define composite distances at the upper level as well.

A general way of obtaining a composite distance from a base distance for any X and $\mathcal{L}$ is the Hausdorff distance (see equation 2 in section 7.1). For the case of intervals, and using d_H, it is easy to see that the *mdbg* of two intervals is the smallest interval containing them, which in interval computing is known as the interval (or convex) hull operator. This can be generalised to hyperrectangles (over the Manhattan distance) and hyperspheres (over the Euclidean distance).

The composability is completed through the use of the nerve, which can guide the way a generalisation is led from examples to groups. In fact, using the minimum distance nerve, and choosing elements and groups which are closest first, leads to a dendrogram, which only requires binary generalisation operators from individuals to an overall single cluster. This way of hierarchical clustering using distances and generalisation operators is known as distance-based conceptual clustering and has been studied by [22].

Another view of composability is when we regard an element as a structure of features or variables. This structure can be a tuple, a set, a list, a hierarchy, a graph or any other structure. We have seen some of these structures in preceding sections, but have not addressed the composability from a base distance. In general, there are several ways of obtaining a distance for the structure. One general approach is the use of kernels, which unveil part of the structure of the example, transforming that into a high dimensional space, where a distance can be derived from a kernel (see, e.g. [23]). In other cases, when having a base distance, we can make a distance-preserving embedding, or multi-dimensional scaling. The increase of dimensionality allows for the use of very simple pattern languages which are distance-based. For instance, axis-parallel models using the Manhattan distance in the form of sets of rules, or support vector clustering using spheres [5] or hyper-rectangles [7].

A second approach is to derive or group the features. A Watanabe-Kraskov variable agglomeration tree [48] can perform variable clustering. Feature construction [37], discretisations, rough sets or granular computing [4] can also be used to change the metric space or derive a new one. In the following section, we analyse the common case of a tuple of different datatypes, each of them with its own distance, and show how to derive a composite distance and the generalisation operators that consistently emerge.

9.2 Tuple composability with the Manhattan distance

A tuple is a widely-used structure for knowledge representation. Indeed, in classical machine learning, examples are tuples of nominal and numerical data. Structured learning usually requires more expressive representation languages. However, in general, it might be useful to integrate both kinds of data. Imagine a problem dealing with protein classification where we have the primary structure along with some chemical traits. Each instance could be represented by means of a tuple, where one of the components would be a list of aminoacids and the other could be numerical information corresponding to its chemical features.

Unless we state otherwise, we denote those objects that will be the components of a tuple by means of superscript symbols followed by a closing parenthesis. Thus, the space X we are working with corresponds to $(X^{1)}, \ldots, X^{n)})$ where every $(X^{i)}, d_i)$ $(i = 1, \ldots, n)$ is a metric space endowed with a pattern language $\mathcal{L}^{i)}$ and a cost function $k^{i)}$. An element e in X is written as $e = (e^{1)}, \ldots, e^{n)})$ and is called a n-tuple. In contrast to sections 5, 6, 7 and 8, no particular pattern language or cost function will be introduced for X; these will be obtained from the distance, the pattern language and the cost function defined over each $X^{i)}$.

We will use the Manhattan distance $d(x, y) = \sum_{i=1}^{n} d_i(x^{i)}, y^{i)})$ as a distance for the space

of tuples. For other distances, e.g. Euclidean, we have seen results for $X = \mathbb{R}^n$ in section 6. However, it is difficult to extend these results to other base data types. For the Manhattan distance, this is possible in general with a very simple pattern language, as we see next.

For the pattern language, we define the basic pattern language $\mathcal{L}_0^{tuple}$ as follows:

$$\mathcal{L}_0^{tuple} = (\mathcal{L}^{1)}, \ldots, \mathcal{L}^{n)})$$

Thus, a pattern $p \in \mathcal{L}_0^{tuple}$ can be unfolded as $p = (p^{1)}, \ldots, p^{n)})$ where $p^{i)} \in \mathcal{L}^{i)}$. According to this, it is reasonable to define the mapping $Set(\cdot)$ over $\mathcal{L}_0^{tuple}$ as:

$$e \in Set(p) \Leftrightarrow \ e^{i)} \in Set(p^{i)}), \forall i = 1, \ldots, n$$

Example 8. *Let $\mathcal{L}^{1)}$ and $\mathcal{L}^{2)}$ be the pattern languages $\mathcal{L}_0^{num}$ and $\mathcal{L}_0^{set}$ introduced in sections 6.1 and 7.2, respectively. Given the pattern $p = ([-10, 10], \{a, b, V\})$, the element $e_1 = (-1, \{a, b, c\})$ is covered while $e_2 = (0, \{a, c\})$ is not covered since $\{a, c\} \notin Set(\{a, b, V\})$.*

It does not matter how complex the pattern languages $\mathcal{L}^{i)}$ are to define $\mathcal{L}_0^{tuple}$. For example, suppose that every $\mathcal{L}^{i)}$ from Example 8 is endowed with the + operation. Then, we would have patterns such as $p = ([0, 1] + [8, 11], \{a, V_1, V_2\} + \{b, V_1V_2\})$ or $p = ([-7, 7], \{b, V_1\} + \{a, V_1, V_2\} + \{V\})$. In the following, we will assume that every $\mathcal{L}^{i)}$ is endowed with +.

Finally, the cost function over X and $\mathcal{L}_0^{tuple}$ must be defined. For a set of elements $E = \{e_i\}_{i=1}^n$ and a pattern p covering E, we can consider

$$K(E, p) = \sum_{j=1}^{n} k^{j)}(\{e_i^{j)}\}_{i=1}^n, p^{j)})$$

Example 9. *Let $X^{1)}$ be the metric space of $\mathbb{R}$ with the absolute difference and let $\mathcal{L}^{1)}$ be the interval pattern language ($\mathcal{L}_0^{num}$). The cost function $k^{1)} = c_{inf}$, i.e., the distance to the border of the interval. Now, given a set E with two examples $e_A = (2.5, 1.4)$ and $e_B = (1.1, 2.3)$, with the pattern $p = ([0, 4], [0, 3])$ covering E (see figure 13), we have that:*

$$K(E, p) \quad = \quad k^{1)}(\{2.5, 1.1\}, [0, 4]) + k^{2)}(\{1.4, 2.3\}, [0, 3]) = 2.6 + 2.1 = 4.7$$

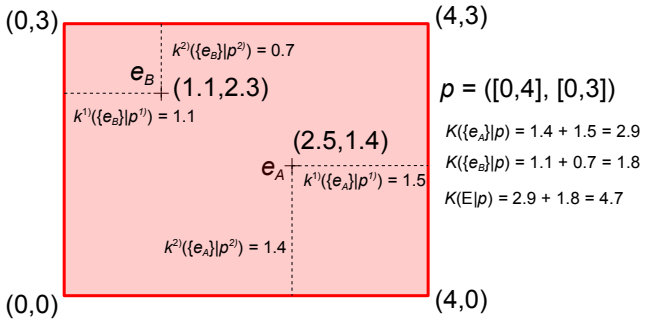


Figure 13: Computing cost function K for the set E with two examples $e_A = (2.5, 1.4)$ and $e_B = (1.1, 2.3)$, and the pattern $p = ([0, 4], [0, 3])$.

In [14], we study how to define *dbg* and *mdbg* operators for several distances (Box and Manhattan) with K and other cost functions. Next, we show the results for $(X, d, \mathcal{L}_0^{tuple}, K)$ with d being the weighted Manhattan distance. We need some definitions first:

Definition 23. ***(Set projection)*** *Given a set of m-tuples $E = \{e_i\}_{i=1}^n \subset X$, then the projection of E over the j-th component, $1 \leq j \leq m$, is just the set $E^{j)} = \{e_i^{j)}\}_{i=1}^n$.*

Definition 24. ***(Nerve projection)*** *Let E be a finite set of m-tuples. Given a nerve function N, we define $N^{k)}$ as the nerve function defined over $E^{k)}$, $1 \leq k \leq m$, such that*

$$(e_i^{k)}, e_j^{k)}) \in N^{k)}(E^{k)}) \Leftrightarrow (e_i, e_j) \in N(E)$$

Proposition 13. *Given the n-tuples e_1, e_2, $e_3 \in X$ and d the Manhattan distance, the n-tuple e_3 is between e_1 and e_2 iff $e_3^{i)}$ is between $e_1^{i)}$ and $e_2^{i)}$, for every $1 \leq i \leq n$.*

Proof. ($\Rightarrow$): We will proceed by contradiction. Thus, e_3 is between e_1 and e_2 and there exists an index i such that $1 \leq i \leq n$ and $e_3^{i)}$ is not between $e_1^{i)}$ and $e_2^{i)}$. Since d is a distance, then:

$$\begin{array}{l} d_j(e_1^{j)}, e_2^{j)}) = d_j(e_1^{j)}, e_3^{j)}) + d_j(e_3^{j)}, e_2^{j)}) \ \ \forall j \neq i, 1 \leq j \leq n \\ d_i(e_1^{i)}, e_2^{i)}) < d_i(e_1^{i)}, e_3^{i)}) + d_i(e_3^{i)}, e_2^{i)}) \end{array}$$

If we sum the formulae above indexed by the indices j and i, we have:

$$\begin{array}{lcl} \sum_{i=1}^n d_i(e_1^{i)}, e_2^{i)}) & < & \sum_{i=1}^n d_i(e_1^{i)}, e_3^{i)}) + d_i(e_3^{i)}, e_2^{i)}) \Leftrightarrow \\ d(e_1, e_2) & < & d(e_1, e_3) + d(e_3, e_2) \end{array}$$

which is not possible since e_3 is between e_1 and e_2. Therefore, $e_3^{i)}$ must lie between $e_1^{i)}$ and $e_2^{i)}$. ($\Leftarrow$): It is direct. Given that $d_i(e_1^{i)}, e_2^{i)}) = d_i(e_1^{i)}, e_3^{i)}) + d_i(e_3^{i)}, e_2^{i)}) \ \ \forall i, 1 \leq i \leq n$, if we sum the formulae above, we automatically have $d(e_1, e_2) = d(e_1, e_3) + d(e_3, e_2)$. □

The following result shows how to define a *dbg* operator in X from the *dbg* operators in $X^{i)}$.

Proposition 14. *Given a finite set of n-tuples E, a nerve function $N(\cdot)$ and the generalisation operators $\{\Delta_i\}_{i=1}^n$ defined over $X^{i)}$. The generalisation operator $\Delta : X \to \mathcal{L}_0^{tuple}$ as*

$$\Delta(E) = (\Delta_1(E^{1)}), \ldots, \Delta_n(E^{n)}))$$

is a dbg operator in (X, d) where d is the Manhattan distance, for N if Δ_i is a dbg operator in $(X^{i)}, d_i)$ for the nerve projection $N^{i)}$, for every $i = 1, \ldots, n$.

Proof. For every $(e_1, e_2) \in N(E)$, if the element $e_3 \in X$ is between e_1 and e_2 then, according to proposition 13 we have

$$d_i(e_1^{i)}, e_2^{i)}) = d_i(e_1^{i)}, e_3^{i)}) + d_i(e_3^{i)}, e_2^{i)}) \ \ \forall i, 1 \leq i \leq n$$

By the definition of nerve projection (see definition 24 in section 9) we can write $(e_1^{i)}, e_2^{i)}) \in N^{i)}(E^{i)}) \ \ \forall i, 1 \leq i \leq n$. As Δ_i is a *dbg* operator for $N^{i)}$ then $e_3^{i)} \in Set(\Delta_i(E^{i)})) \ \ \forall i, 1 \leq i \leq n$. Hence, we can conclude that $e_3 \in Set(\Delta(E))$. □

(Remark 1) If Δ_i is not *db* for $N^{i)}$, then Δ is not *db* for N.

We see that this is an extension of the results for hyper-rectangles in section 6.3 independently of the base metric spaces (not necessarily real numbers). What we want to know next is whether this kind of patterns is minimal in $(\mathcal{L}_0^{tuple}, K)$.

Corollary 1. *Let Δ be the dbg operator and let Δ_i be the group of dbg operators defined in proposition 14. If for every i, Δ_i is minimal for $N^{i)}$, then Δ is minimal for N.*

Proof. We proceed by contradiction. Let us suppose that Δ is not minimal. Then, there exists a finite set of elements E and a *db* pattern $p = (p_1, \ldots, p_n) \in \mathcal{L}$ of E for $N(E)$ such that $K(E, p) < K(E, \Delta(E))$. As K is the sum of n positive functions $(k^{i)})$, then necessarily $\exists i,\ 1 \leq i \leq n : k_i(E^{i)}, p^{i)}) < k_i(E^{i)}, \Delta_i(E^{i)}))$, which is impossible since $p^{i)}$ is a *db* pattern of $E^{i)}$ for $N^{i)}(E^{i)})$ and Δ_i is minimal for $N^{i)}$.

□

10 Examples

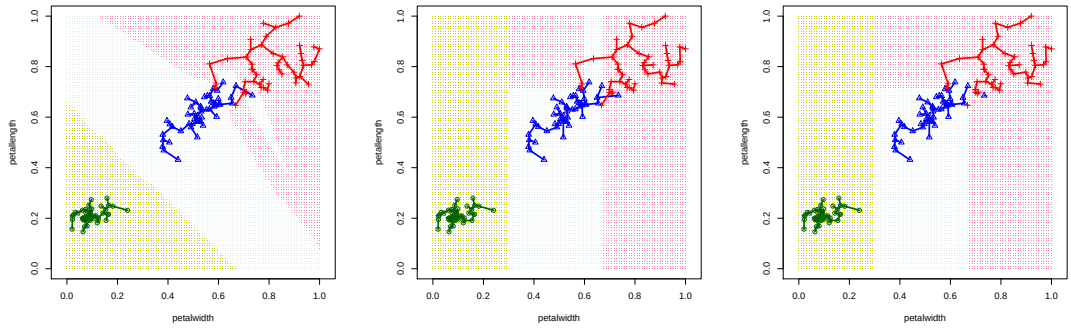
In this section we include some examples which illustrate how the setting introduced in this paper can be used to analyse existing (or develop new) techniques in machine learning, data mining and artificial intelligence in general. In fact, there are some algorithms in this line, such as the so-called distance-based decision trees [19][14][33] or distance-based conceptual clustering [22]. These approaches did not use the concepts of *dbg* or *mdbg* directly, but they work upon the relation and consistency between distance and generalisation.

10.1 A classification example

While the concept of *dbg* is appropriate in general, the notion of *mdbg* is especially useful for classification, since an *mdbg* does not extend very far beyond the given data and tries to avoid over-fitting. The notion of nerve is also fundamental, since the convex hull for all the examples (i.e., the complete nerve) typically loses the shape. The notion of minimum spanning tree class descriptor [30] is of particular interest here, as it uses the minimum spanning tree as a base for generalising the data. We will use the minimum spanning tree (i.e., the minimum distance nerve) to analyse some machine learning algorithms.

Figure 14 shows an example for the dataset iris from the UCI repository [21], where only the two most informative variables are used: petalwidth and petallength. This dataset has three classes. We show the minimum-distance nerves calculated for each class by using the minimum spanning tree algorithm. On the left, these nerves are calculated with the Euclidean distance. We see the clasification given by 1-NN. If we remove the two links which cross, this classification is consistent with the nerves (distance-based). In the middle, we see the minimum-distance nerves but now calculated with a Manhattan distance. These nerves are compared to a decision tree, where we see that even though decision trees use axis-parallel patterns, it is not consistent with the nerves. On the right, we have removed the links that cross from the nerves, and a further split on the decision tree to make the decision tree distance-based for the nerves.

The example shows that some algorithms generate patterns which are more consistent with some distances than others. For instance, k-NN generates a consistent partition with the



(a) (b) (c)

Figure 14: Nerves, patterns and classification for the iris dataset (only petalwidth and petallength used). **(a)** The minimum-distance nerves for the three classes (given by 1-NN) using the Euclidean distance. **(b)** The minimum-distance nerves for the three classes (given by a decision tree) using the Manhattan distance. **(c)** Same as (b) but two elements have been removed from the nerve and the decision tree has been corrected. Now, it is distance-based.

skeleton of each class using the minimum distance nerve. On the contrary, decision trees may generate unconnected regions which are closely connected by the distances.

10.2 A clustering example

One of the advantages of the setting presented in this paper is that it is applicable to any pair of datatype and distance, for which several *dbg* and *mdbg* may exist. Given the results for the numeric data, nominal data, sets, lists, graphs and first-order objects, and their tuple-composability described in the Sections 5 to 9, it is insightful to illustrate how our framework works for multiple data representations, pattern languages and distances over the same problem. Now, we are going to address a clustering problem. We will work with a small set of research publications extracted from the DBLP bibliographic server. The dataset contains 44 titles from the same author[7]. We are interested in finding clusters in these publications according to the names contained in the titles. Our purpose is twofold. On the one hand, we aim to extract distance-based patterns describing those clusters computed by an agglomerative hierarchical clustering algorithm. On the other hand, we want to compare the different cluster descriptions that are obtained with several distance functions, pattern languages and data representations.

First, we need to prepare the data. The preprocessing task simply focuses on removing the stop words ("a","an", "the", etc.) from the data set. We also added an artificial word to make all titles have the same length. To find the groups in the data, we employ a clustering algorithm that builds the hierarchy from the individual elements by progressively merging clusters. The *average linkage criterion* determines what clusters must be joined. Basically, each cluster is represented by a prototype (the element in the cluster whose average distance to the rest is lowest). We define the radius of a cluster as the average distance from all the elements of the cluster to its prototype. Then, the two clusters C_1 and C_2 that have the nearest prototypes

[7]The datasets and the source code for this section can be downloaded from *http://users.dsic.upv.es/∼flip/mdbgclusters/mdbgclusters.zip*.

will be joined, if the radius of C_1 plus the radius of C_2 is equal to or greater than $l * d$, where d is the distance between the prototypes and l is a coefficient in $]0, 1]$. Below, we only show results for $l = 0.95$, which give a high number of clusters but are more informative than with lower values of l. For other values of l, we refer the reader to [14].

Once the clustering algorithm is finished, the minimum spanning tree is used as a nerve for each cluster and the *mdbg* operator is applied to each cluster independently. We explore three different options for data representation, for the distance and the *mdbg*. One option is to have lists with the edit distance, another is tuples with the Manhattan distance (equivalent here to the Hamming distance) and finally sets with the symmetric distance.

We begin with lists. In this case, the edit distance is used to measure the similarity between two given titles, where each word in the title is treated as a single symbol. The language $\mathcal{L}_0^{list}$ uses constants (in this example, words) and variables as mentioned in section 8.1. Variables cannot be repeated in the same pattern. We will use the notation V^n to represent n consecutive and *different* variables. As usual, we define $\mathcal{L}_1^{list}$ using the disjunction operator + introduced in definition 3 in section 3. Finally, the cost function is $k(E, p) = c(p) + c(E|p)$, where the syntactic part $c(p)$ is equal to the number of both ground and variable symbols in p, and $c(E|p)$ is c_{inf} in Table 1 in section 4. With this setting for lists, the patterns are shown in Table 3.

Cluster with 5 elements	p_1 =[V^5, learning, V^{11}]+[V^5, probability, V^2, ranking]
Cluster with 4 elements	p_2 =[V^3, data, mining, V^9]
Cluster with 3 elements	p_3 =[V^3, machine, learning, V^5]
Cluster with 3 elements	p_4 =[V^2, scaling, inductive, V^5]
Cluster with 3 elements	p_5 =[V^6, relational, V^8]
Cluster with 2 elements	p_6 =[inductive, policy, V^3]
...	...
Cluster with 2 elements	p_{11} =[V^3, active, feature, value, acquisition, V^2]

Table 3: Some of the patterns covering more than one element using $\mathcal{L}_1^{list}$.

Since the generalisation operators used are *dbg* we know that each element which is between two elements in the cluster is also inside the cluster. The use of lists for this dataset is able to capture some patterns which depend on the order of words, but in other cases the position of the words in a title might not be meaningful to group them. Therefore, the next two data representations focus on the word presence rather than on their positions.

The use of tuples here corresponds to the well-known vector-space model for texts. In this model, each title is represented by a vector of word frequencies. In our case, we simplify this representation by only using 0 and 1, which respectively mean absence or presence of a word in a title, respectively. We define $d(\cdot, \cdot)$ as the discrete distance and introduce the pattern language $\mathcal{L}^{attribute} = \{0, 1, V\}$, where V is a variable symbol representing 0 or 1. The *dbg* operator is then $\Delta(e_1, e_2) = e_1$ if $e_1 = e_2$, otherwise it is V. Note that this is equivalent to the multidimensional nominal case seen in section 5.3. According to proposition 14, a pattern over the space of tuples is an n-dimensional tuple over $\{0, 1, V\}$, such that 1 means that all the titles in E have this word, 0 means that none of the titles in E have this word, and the symbol V denotes that some may have the word and some others may not. For example, given the dictionary $(fuzzy, inductive, machine, logic, programming)$, the pattern $(1, 1, V, V, 0)$ represents all the titles containing the words "fuzzy" and "inductive" but not the word "programming".

With this notation, Table 4 shows some of the obtained clusters. One observation is that the patterns have many 0s, i.e., many words that cannot appear in the clusters.
Finally, the titles are viewed as sets of words. Recall that, in this case, the distance is given by

Cluster	WITH	WITHOUT
Cluster with 3 elements (p_1)	scaling, learning	accuracy, acquisition,...,weakening
Cluster with 2 elements (p_2)	selection, bias	accuracy, acquisition,...,weakening
Cluster with 2 elements (p_3)	scaling, inductive, algorithms	accuracy, acquisition,...,weakening
Cluster with 2 elements (p_4)	ranking, prob., active, estimation, class	accuracy, acquisition,...,weakening
...	...	
Cluster with 2 elements (p_8)	bands, confidence, empirical, roc	accuracy, acquisition,...,weakening

Table 4: Some patterns covering more than one element using a tuple representation.

the size of the symmetric difference. We employ the pattern language $\mathcal{L}_0^{set}$ defined in subsection 7.2 along with the generalisation operator defined in proposition 10 from section 7. The obtained results are shown in Table 5. It is easy to see that tuple-based and set-based representations produce the same clusters. However, each pattern language associated to each representation provides certain information that is disregarded by the other. For instance, with a set-based representation we can know how many different words two examples have.

Cluster with 3 elements	$p_1 = \{$scaling, learning, $V^7\}$
Cluster with 2 elements	$p_2 = \{$selection, bias, $V^7\}$
Cluster with 2 elements	$p_3 = \{$scaling, inductive, algorithms, $V^6\}$
Cluster with 2 elements	$p_4 = \{$ranking, probability, active, estimation, class, $V^2\}$
...	...
Cluster with 2 elements	$p_8 = \{$bands, confidence, roc, empirical, $V^4\}$

Table 5: Some patterns covering more than one element using $\mathcal{L}_0^{set}$.

This simple example shows some relevant traits of our framework. As expected, data representation has a great influence on the patterns obtained. When data representation changes, data tends to be grouped differently, thus affecting the patterns. This is observed in the list pattern p_1 in Table 3 where the subpattern $[V^5, probability, V^2, ranking]$ covers the titles:

$$\begin{array}{lcl} t_1 & \equiv & \text{Active Sampling for Class Probability Estimation and Ranking} \\ t_2 & \equiv & \text{Tree Induction for Probability-Based Ranking} \\ t_3 & \equiv & \text{Active Learning for Class Probability Estimation and Ranking} \end{array}$$

However, when titles are viewed as sets (or tuples), t_2 is not grouped with t_1 and t_3, which results in the pattern $\{ranking, V^2, probability, active, estimation, class\}$ (see pattern p_4 in Table 5 with $l = 0.95$ or pattern p_4 in Table 4 with $l = 0.95$). Nonetheless, what is novel here is that patterns are also influenced by the distance itself. For instance, the titles below:

$$\begin{array}{lcl} t_1 & \equiv & \text{Confidence Bands for ROC Curves: Methods and an Empirical Study} \\ t_2 & \equiv & \text{ROC confidence bands: an empirical evaluation} \end{array}$$

are put in the same cluster for lists, tuples and sets. However, the pattern for lists differs from the pattern for tuples and sets, with these last two being equivalent. That is,

$$\begin{array}{lcl} \text{lists} & = & [V, \text{confidence, bands, } V^3, \text{empirical, } V^3] \\ \text{tuples} & = & \text{WITH: bands, confidence, empirical, roc} \\ \text{sets} & = & \{V^4, \text{bands, confidence, roc, empirical}\} \end{array}$$

Even though both titles have the word *ROC*, this word is disregarded when computing the distance between the lists representing the titles, since, in the list, the position of the word affects the distance. Thus, the pattern that we obtain is in accordance with the distance in

that it reveals traits that are taken into account by the distance. The consequence is that a pattern saying *"all the titles having the words ROC and ..."* would be admissible for the elements in the cluster but not for the distance. However, things are different when we move to tuples or sets because the word *ROC* is now considered by the distance.

11 Variants and Extensions

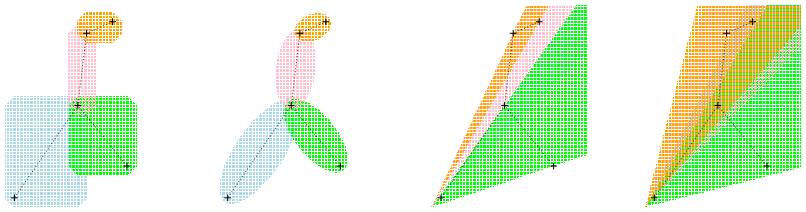
There are several sorts of flexibility in the setting presented so far. We can combine several distances, generalisation operators, nerves and cost functions. Also, there are some decisions that have been made which could be reconsidered in order to make the setting different or to show connections with other approaches or disciplines.

Next, we extend the notion of coverage in three ways: relaxing the triangle inequality, the use of non-crisp pattern languages, and the handling of noise and negative examples.

11.1 Variants on the distance and the triangle inequality

Regarding variants for the distance, one first option would be to consider similarities which do not follow the symmetric or triangle inequality. In fact, some of these similarities have been commonly used, such as the cosine similarity or the so-called Bregman divergences, especially for clustering [3], and its connection with loss functions. In the case of Bregman divergence, the notion of Minimum Enclosing Bregman Ball has been developed, which would be the counterpart to the sphere pattern language seen in section 6.4. If one considers similarites without the triangle inequality, the notion of being in between can still be used, but the strict equality in definition 6 should be relaxed by using a margin or tolerance.

Definition 25. (Intermediate element relative to a dissimilarity with margin ϵ) *Given a space X, with dissimilarity function d and two elements $e_1, e_2 \in X$, we say that an element $e_3 \in X$ is ϵ-between e_1 and e_2, if $d(e_1, e_2) + \epsilon \geq d(e_1, e_3) + d(e_3, e_2)$.*



(a) (b) (c) (d)

Figure 15: Four skeletons for the points in $\mathbb{R}^2$ and nerve shown in figure 3 (right) **(a)** Manhattan distance with margin 0.1. **(b)** Euclidean distance with margin 0.05. **(c)** Cosine dissimilarity with margin 0. **(d)** Cosine dissimilarity with margin 0.05.

Using the previous definition for skeletons, we find that for positive values of ϵ we have bigger skeletons. For negative values of ϵ the skeletons are empty for distances and smaller for similarities (depending on ϵ). We can see several examples in figure 15.

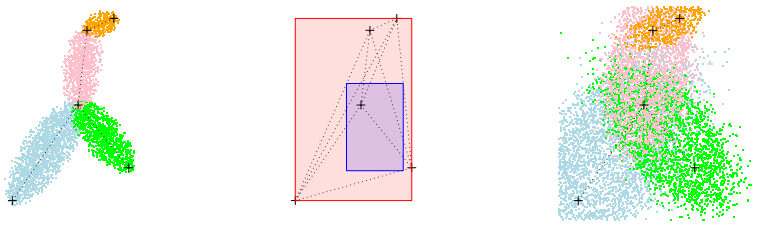
This notion of skeletons with margin is, in fact, close to the notion of the one-class classifier constructed by the minimum spanning tree class descriptor [30]. A somewhat similar idea would be to define a *rough* distance, where the distance returns an interval instead of an exact value.

11.2 Variants on non-crisp pattern languages

There are many notions from fuzzy sets and related notions which could be used in some of the constructs in our setting. One of the most natural applications of fuzzy sets would be the use of a fuzzy membership function for the skeleton, so the skeleton becomes a fuzzy set.

Definition 26. ***(Fuzzy Skeleton)*** *Let (X, d) be a metric space, $\mathcal{L}$ a pattern language, a set $E \subseteq X$, and η a nerve of E. Then, the fuzzy skeleton of E for η and a margin ϵ, denoted by $\mu_\epsilon(\eta)$, is defined as a fuzzy set with membership function $\mu(z) = max_{(e_1,e_2)\in\eta} \frac{sim(z,e_1,e_2)}{\epsilon}$ where $sim(z, e_1, e_2) = d(e_1, e_2) + \epsilon - (d(e_1, z) + d(z, e_2))$.*

Figure 16 (left) shows a fuzzy skeleton, i.e. a skeleton using a fuzzy membership function which is linearly proportional to the maximum margin that an element has relative to two elements connected by the nerve. In definition 26, the membership function for more than two elements is constructed using the nerve and a *max* function, but other options would be possible. For instance, an alternative, different, option would be to use the bounded sum (also known as Lukasievwicz co-norm), i.e. $\mu(z) = min(1, \sum_{(e_1,e_2)\in\eta} sim(z, e_1, e_2)/\epsilon)$.



(a) (b) (c)

Figure 16: Variants of the notion of skeleton and distance-based generalisation over some nerves in figure 3 (right) **(a)** Fuzzy Euclidean distance with margin 0.05 over the minimum distance nerve. **(b)** A possible view of a rough set for a skeleton using the Euclidean distance and the convex hull. **(c)** A Gaussian mixture using the nerve.

From here it is easier to define fuzzy *db* operators, by setting a minimum level of membership, such as "90% of the fuzzy skeleton membership function must be inside the pattern".

Also related, rough sets are are especially useful when the expressiveness of the language cannot fit the concept perfectly, some examples cannot be separated with the pattern expressiveness or we are working with a pseudometric, where different elements might have distance 0, therefore, indistinguisable. A similar situation may take place in our setting whenever the pattern language is very different to the distance. For instance, in figure 16 (b), we see a pattern language composed of axis-parallel rectangles and a Euclidean distance. Although there can be *mdbg* operators using this combination, as we saw in section 6.3, we see that they may be much too general, as the red pattern in figure 16 (b). An approach to better fit the points might be to construct the convex hull of the skeleton and look for the maximal pattern inside this convex hull using cost functions. In this case, if the cost function is, e.g., $1/area$, the blue pattern in figure 16 (b) would represent the lower approximation of the concept. As a result, the generalisation of the points would be a rough set determined by two patterns.

Finally, we could go probabilistic, and, e.g., compare the previous extensions to a Gaussian mixture with a multi-variate normal distribution for each pair of elements connected by the

nerve using the segment connecting both points as central vector and half the distance as width to generate the covariance matrix, as shown in figure 16 (c). From this Gaussian mixture, a notion of probabilistically distance-based pattern could be derived.

11.3 Variants on noise and negative examples

Additionally, another line would be to upgrade the current framework in order to deal with noisy data and negative examples. This issue has traditionally been solved by the learning algorithm rather than the generalisation operator embedded in it; the learner takes care of noisy data and decides how to use the generalisation operator when negative examples are provided.

Generalisation operators that are tolerant to noise could be defined through a modification of the definition of nerve (which is just a graph but not necessarily a connected graph), in order to relax the reachability property to only connect elements at small distances, or to define a probabilistic notion of nerve. Likewise, there can also be several ways to extend our setting to cope with negative examples. One possibility would be to alter the definition of nerve in order to prevent nerves of one class from crossing the nerves of other classes (as we did in figure 14 (c)) or to keep a nerve from passing near an element of a different class.

12 Conclusions

The main purpose of this work was to bridge the gap between distance and generalisation. Given a metric space, it is not desirable to obtain a pattern that generalises a set of elements that completely ignores the underlying distance. Although other different notions of consistency between distance and generalisation could be defined, we have supported our particular choice with a collection of arguments and examples. Thus, the setting is based on several conditions/properties. Firstly, a distance-based generalisation should include the elements that are in between the original elements (in terms of the distance). That is, consistency between a pattern and the distance is understood as a covering condition in such a way that the pattern must cover the elements that are in the path that defines the distance between the original elements (e.g., the segment in $\mathbb{R}$). Secondly, we extend this concept for more than two elements through the notion of nerve, thereby bridging another gap between a concept that is binary in essence (distance) and a concept that is not necessarily binary (generalisation).

Finally, another interesting property that has been addressed is minimality. This notion is formalised through cost functions, whose semantic part is exclusively defined in terms of distances, completing the connection between the concepts of distance, pattern and generalisation. Hence, in our framework, the *mdbg* operators are based on a cost function. This is more flexible than when the *mdbg* operator is solely based on the notion of inclusion.

The connections and definitions we introduce, as well as the operators and results obtained for many different datatypes, may have many implications and broad applicability in different fields of artificial intelligence at several levels:

- Distance-based methods will have a general way to obtain a generalisation operator producing a pattern that explains the classification or clustering, as we illustrate at the end of this paper with some simple distance-based methods applied to the same problem with several distances, languages and generalisation operators.

- Several specific algorithms or paradigms that have been developed in some specific areas in artificial intelligence or machine learning (e.g., in ILP) could be extended or adapted to other areas by simply changing the generalisation operator.
- It provides new tools for the problem of learning from complex data. It can take advantage of developments in distance-based and kernel-based methods for structured data [23], while preserving comprehensibility.
- Distance-based generalisation operators are better than other generalisation operators not only because they are consistent with an underlying distance, but also because this distance can be used to guide the generalisation search, given the duality between the metric space and the generality-ordered space (e.g. nearest elements are generalised first), possibly improving bottom-up and top-down model-based methods.
- Belief or theory revision can be guided when the distance-based condition is violated. New evidence which is not covered by the theory should be included by extending the theory taking into account that new intermediate objects should be covered as well.

This work deals with fundamental issues, explores their applicability to many common datatypes, distances, pattern languages and presents results for most of them. Nevertheless, there is still future work ahead. At a theoretical level, many other distances and pattern languages exist under the proposed framework. Many links with areas such as fuzzy sets, rough sets, interval computing, granular computing, support vector clustering, etc., have been outlined here, but should be fully explored in more detail. Similarly, we think our work can have direct application in areas such as theory (belief) revision, case-based reasoning, knowledge integration and information retrieval where both concepts and distances can be used to select or deal with similar objects, documents or any other kind of structured information.

Overall, this work has clarified the relationship between some core concepts in artificial intelligence (distance, generalisation and minimality) and has presented a setting that can be used to integrate distance-based and generalisation-based approaches in a consistent way.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work has been partially supported by the EU (FEDER) and the Spanish MICINN, under grant TIN2010-21062-C02-02, the Spanish project "Agreement Technologies" (Consolider Ingenio CSD2007-00022) and the GVA project PROMETEO/2008/051.

References

[1] E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2004.

[2] E. Armengol, E. Plaza, and S. Ontanón. Explaining similarity in CBR. In *ECCBR 2004 Workshop Proceedings*, pages 155–164. Citeseer, 2004.

[3] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005.

[4] A. Bargiela and W. Pedrycz. *Granular computing: an introduction*. Springer, 2003.

[5] A. Ben-Hur, D. Horn, H. Siegelmann, and V. Vapnik. Support vector clustering. *The Journal of Machine Learning Research*, 2:125–137, 2002.

[6] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(9):689–694, 1997.

[7] O. Chapelle and V. Vapnik. Model selection for support vector machines. *Advances in neural information processing systems*, 12:230–236, 1999.

[8] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(3):21–27, 1967.

[9] P. Cunningham. A taxonomy of similarity mechanisms for case-based reasoning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1532–1543, 2008.

[10] M. Develin. Dimensions of tight spans. *Annals of Combinatorics*, 10:53–61, 2006.

[11] P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24(2):141–168, 1996.

[12] K. Driessens and S. Dzeroski. Combining model-based and instance-based learning for first order regression. In L. D. Raedt and S. Wrobel, editors, *ICML*, pages 193–200, 2005.

[13] T. Eiter and H. Mannila. Distance measures for point of sets and their computation. *Acta Informatica*, 34, 1997.

[14] V. Estruch. Bridging the gap between distance and generalisation: Symbolic learning in metric spaces. PhD Thesis, DSIC-UPV `http://www.dsic.upv.es/~vestruch/thesis.pdf`, 2008.

[15] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. Ramírez-Quintana. Generalisation Operators for Lists Embedded in a Metric Space. *LNCS*, 5812:117–139, 2010.

[16] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Distance based generalisation. In *15th Intl. Conf. on ILP*, volume 3625 of *LNCS*, pages 87–102, 2005.

[17] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Distance based generalisation for graphs. In *Proc. Workshop MLG*, pages 133–140, 2006.

[18] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Minimal distance-based generalisation operators for first-order objects. In *In Proc. of the 16th Int. Conf. on ILP*, pages 169–183, 2006.

[19] V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Web categorisation using distance-based decision trees. *E. N. Theor. Comput. Sci.*, 157(2):35–40, 2006.

[20] G. Finnie and Z. Sun. Similarity and metrics in case-based reasoning. *International journal of intelligent systems*, 17(3):273–287, 2002.

[21] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[22] A. Funes, C. Ferri, J. Hernández-Orallo, and M. Ramírez-Quintana. An Instantiation of Hierarchical Distance-Based Conceptual Clustering for Propositional Learning. *Advances in Knowledge Discovery and Data Mining*, pages 637–646, 2009.

[23] T. Gaertner, J. W. Lloyd, and P. A. Flach. Kernels and distances for structured data. *Machine Learning*, 57(3):205–232, 2004.

[24] B. Gao. *Hyper-rectangle-based discriminative data generalization and applications in data mining*. PhD thesis, Simon Frasier University, 2006.

[25] A. Golding and P. Rosenbloom. Improving rule-based systems through case-based reasoning. In *National Conference on Artificial Intelligence*, pages 22–27, 1991.

[26] U. Hahn and N. Chater. Concepts and similarity. *in K.Lamberts and D. Shanks (eds.) Knowledge, concepts, and categories*, pages chapter 2, 43–92, 1997.

[27] U. Hahn, N. Chater, and L. B. Richardson. Similarity as transformation. *Cognition*, 87(1):1 – 32, 2003.

[28] C. Hu. Interval Rule Matrices for Decision Making. *Hu et al. (eds) 'Knowledge Processing with Interval and Soft Computing'*, pages Chapter 6, 135–146, 2008.

[29] J. Jost. *Riemannian geometry and geometric analysis*. Springer Verlag, 2008.

[30] P. Juszczak, D. Tax, et al. Minimum spanning tree based one-class classifier. *Neurocomputing*, 72(7-9):1859–1869, 2009.

[31] R. Kearfott and C. Hu. Fundamentals of Interval Computing. *in Hu et al. (eds) 'Knowledge Processing with Interval and Soft Computing'*, pages Chapter 1, 1–12, 2008.

[32] P. Kumar, J. Mitchell, and E. Yildirim. Approximate minimum enclosing balls in high dimensions using core-sets. *Journal of Experimental Algorithmics (JEA)*, 8:1–1, 2003.

[33] F. Martínez-Plumed, V. Estruch, C. Ferri, J. Hernández-Orallo, and M. Ramírez-Quintana. Newton trees. *Advances in Artificial Inteligence, LNAI*, 6464:174–183, 2011.

[34] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[35] R. Moore, R. Kearfott, and M. Cloud. *Introduction to interval analysis*. Society for Industrial Mathematics, 2009.

[36] S. H. Muggleton. Inductive logic programming: Issues, results, and the challenge of learning language in logic. *Artificial Intelligence*, 114(1–2):283–296, 1999.

[37] S. Piramuthu and R. Sikora. Iterative feature construction for improving inductive learning algorithms. *Expert Systems with Applications*, 36(2):3401–3406, 2009.

[38] G. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.

[39] L. D. Raedt and J. Ramon. Deriving distance metrics from generality relations. *Pattern Recognition Letters*, 30(3):187–191, 2009.

[40] J. Ramon and M. Bruynooghe. A framework for defining distances between first-order logic objects. In *In Proc. of Int. Conf. on ILP*, volume 1446 of *LNCS*, pages 271–280, 1998.

[41] J. Ramon, M. Bruynooghe, and W. V. Laer. Distance measures between atoms. In *CompulogNet Area Meeting on Computational Logic and Machine Learning*, pages 35–41. University of Manchester, UK, 1998.

[42] J. Rissanen. Hypothesis selection and testing by the MDL principle. *The Computer Journal*, 42(4):260–269, 1999.

[43] S. Salzberg. A nearest hyperrectangle learning method. *Mach Learning*, 6:251–276, 1991.

[44] A. Stanfill and D. Waltz. Toward memory-based resoning. *Communications of the ACM*, 29:1213–1228, 1986.

[45] R. E. Stepp and R. S. Michalski. *Conceptual Clustering: Inventing Goal Oriented Classifications of Structured Objects*, pages 471–498. Morgan Kaufmann, Los Altos, CA, 1986.

[46] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Applications*, 16(2):264–280, 1971.

[47] C. S. Wallace and D. L. Dowe. Minimum Message Length and Kolmogorov Complexity. *Computer Journal*, 42(4):270–283, 1999.

[48] S. Watanabe. *Knowing and guessing: A quantitative study of inference and information.* Wiley New York, 1969.