Distance Based Generalisation

V. Estruch C. Ferri J. Hernández-Orallo M.J. Ramírez-Quintana

DSIC, Univ. Politècnica de València , Camí de Vera s/n, 46020 València, Spain.
{vestruch,cferri,jorallo,mramirez}@dsic.upv.es

Abstract. Many distance-based methods in machine learning are able to identify similar cases or prototypes from which decisions can be made. The explanation given is usually based on expressions such as "because case a is similar to case b". However, a more general or meaningful pattern, such as "because case a has properties x and y (as b has)" is usually more difficult to find. Even in this case, the connection of this pattern with the original distance-based method is generally unclear, or even inconsistent. In this paper, we study the connection between the concept of distance (or similarity) and the concept of generalisation. More precisely, we define several conditions a sensible distance-based generalisation must have. From here, we are able to tell whether a generalisation operator for a pattern representation language is consistent with the metric space defined by the underlying distance. We show sensible pattern languages and generalisation operators for typical data types: nominal, numerical, sets and lists. We also explore a possible relationship between the well-known concepts of lgg and distances between terms, and the definition of generalisation presented in this paper.

keywords: distance-based methods, generalisation operators, lgg, metric space.

1 Introduction

The concept of distance is key in many areas such as case-based reasoning [1], machine learning [6], diagnosis, information retrieval [2], etc. Distance, as the mathematical concept of dissimilarity, allows many learning techniques to be applied to quite different kinds of data and situations, provided we are able to define a distance for the instances or cases at hand. Distance-based methods, then, are easily adaptable to any kind of applications. However, the problem of converting the similarity traits into a numerical value (the distance) is that the information on the matches or coincidences is lost. Consequently, many distance-based methods cannot give an explanation of their answers. For instance, a distance-based method [5] such as k-nearest neighbours can output that the film X is likely to be appropriate for a customer because the k-nearest neighbours of X were appropriate for the customer, but are not able to give a general *pattern* of why it is the case, such as the film X is likely to be appropriate for the customer because X is an action movie where the good guys win, and the customer liked all the films with these traits s/he hired before.

The connection of distance and pattern, or more precisely, the connection of distance and generalisation is not new. Many learning techniques (e.g. clustering or classification) generate a prototype (or centroid) and the generalisation area is based on a certain distance ball from the prototype. However, a good distance-based generalisation can have no meaningful pattern which is able to express the generalisation or, in other words, a generalisation can have no good representation. For instance, the generalisation "all the films with more than 4 traits in common with film X" is a well-defined general region according to a distance based on the number of common traits, but it lacks a meaningful pattern. Vice versa, a meaningful pattern can lead to very bad or unintuitive generalisations. For instance, the pattern "all the documents that contain the string *declarative programming*" is a meaningful pattern but is not a good generalisation if edit distances are taken into account, since two very similar documents can just differ on this string or two very different documents can just match on this sequence.

Consequently, if we are using a distance-based method to determine similar objects or to establish centroids or prototypes, we need a representation pattern that is consistent with the metric space defined by the underlying distance. In order to do this, we have to identify a series of conditions that a pattern representation language and a generalisation operator must fulfil.

Let us illustrate this idea with a more detailed example. Consider an intruder-detection problem where we want to detect whether a user might be an intruder. For each user, we record the machines (1, 2, 3) where they have made a 'ping' (p), have connected (c) or have failed to connect (f). For each user we record a 'sequence' of actions. For instance, "$p_1 f_2 c_2$" means a user that first *pinged* to machine 1, then failed to connect to machine 2 and finally connected to machine 2. With this data we apply case-based reasoning to determine for a new user whether s/he is an intruder or not. More specifically, we compute edit distances between the sequences of actions and then use a k-nearest neighbour (k-nn) to determine the class of each new case. The following picture shows the case of determining whether "$p_1 f_1 f_2 f_3$" is an intruder when we use 7-nn and the seven nearest examples (with distances 2, 4, 4, 3, 3 for the positives and 3, 5 for the negatives), as shown in Figure 1. Since among the 7-nearest neighbours there are five positive cases and two negative cases, the sequence "$p_1 f_1 f_2 f_3$" is labelled as a possible 'intruder'.

A different thing is when we want to extract a pattern to explain the 7-nn classification. The pattern can be determined taking into account the five positive examples and trying to ascertain what they have in common. A possible pattern for the 5 positive cases could be $\{*f_1 f_1 *, *f_3 f_3 *\}$, meaning that any instance containing two consecutive f_1 or two consecutive f_3 is an intruder. Apparently, this is a good pattern, since it covers all the five positive examples, and none of the two negative examples. Additionally, it is not too specific and it is not too general, and it is meaningful. Despite this idealistic picture, the pattern conceals a surprise: it does not cover the centre point "$p_1 f_1 f_2 f_3$"! Even worse, the point "$p_1 f_1 f_2 f_3$" has a distance 2 to "$p_1 f_1 f_1 f_3$" and has distance 3 to "$p_3 f_1 f_2 f_3 f_3$", where these two latter examples have distance 5. That is, two examples are

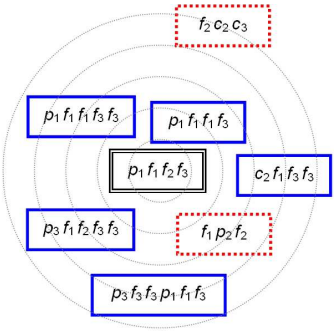

Fig. 1. 7-nn with 5 positive (solid line) and 2 negative (dotted line) cases for classifying the case "$p_1\,f_1\,f_2\,f_3$".

in the pattern, but a middle point is not in it. These two observations show that any meaningful pattern in a general situation may be inconsistent w.r.t. the underlying distance. The pattern is then useless to explain or represent the behaviour of the distance-based technique.

Consequently, if we use a pattern representation language for explaining some distance-based generalisation we have to check first whether the language and the generalisation operator meet some properties.

In this paper, we study the connection between the concept of distance (or similarity) and the concept of generalisation, and the pattern representation languages that are able to express sensible generalisations. This is the first step in order to make the idea of obtaining meaningful explanations of the answers given by a distance-based learning method applicable to the broadest kind of distance-based techniques as possible. In the following section, we analyse which generalisations are considered unintuitive in a metric space, and we define two properties a generalisation operator must satisfy. In Section 3 we show sensible pattern languages and binary generalisation operators for the typical data types: nominal, numerical, sets and lists. Next, we analyse the relationship between our generalisation concept and the lgg operator defined by Plotkin. We show that, using the metric defined in [9] which is based on the lgg concept, this operator satisfies the conditions to use it as a pattern constructor for the first-order logic language.

Finally, in the conclusions, we discuss on the application of this integrated view to distance-based methods such as k-nearest neighbours, clustering, distance-based decision trees and other case-based reasoning techniques. We also propose some ideas for the future work.

2 Generalisation in Metric Spaces

In this section, we propose a sensible notion of generalisation for metric spaces. From now, we will work with metric spaces[1].

Definition 1. *Given* $\Delta : X \times X \rightarrow 2^X, \Delta$ *is a* k*-generalisation* ($k \geq 1$) *if* $\forall a, b \in X$ *these two conditions hold:*
1 (No Isolation)

$$\forall x \in \Delta(a,b) \text{ and } \forall \epsilon > 0, \text{ if } \{x\} \subsetneq B(x,\epsilon) \text{ then } B(x,\epsilon) \cap \Delta(a,b) \neq \{x\}$$

where $B(x,\epsilon) = \{x' \in X : d(x,x') < \epsilon\}$.
2 (Scope)

$$\forall x \in X, \text{ if } d(a,x) + d(x,b) \leq k \cdot d(a,b) \text{ then } x \in \Delta(a,b)$$

Note that Condition 2 implies that $a, b \in \Delta(a,b)$. We employ the term k-generalisation instead of generalisation, because in this way we have a more flexible definition. The value of k establishes the level of generality, i.e. for greater values of k more instances are allowed inside the generalisation.

Both conditions are important when considering generalisations. The first one (No isolation) restricts the definition in the following way; if there is an example that belongs to the generalisation, for any ball centered in this example if there are other examples in this ball, at least one must belong to the generalisation. This condition rejects the generalisations that have isolated points. The following example shows how this condition is useful to reject unintuitive generalisations.

Example 1. Let us consider a metric space formed by three elements a, b and c where $d(a,b) = 1$, $d(b,c) = 2$, $d(a,c) = 2$. If we define a generalisation $\Delta(a,c) = \{a,c\}$, we can see that it trivially verifies Condition 2 ($d(a,b) + d(b,c) > d(a,c)$). However this generalisation does not satisfy condition 1 (consider a ball of radius 2 centered in a, then $B(a,2) = \{a,b\}$ but $b \notin \Delta(a,c)$).

The second condition (Scope) rejects the generalisations that are much too specific in the sense that do not consider all the points between two generalised points. The following example illustrates this:

Example 2. Let us consider the metric space $\Re$ where the metric is defined as the absolute difference. If we define $\Delta(4,5)$ formed by $[3.8, 4.2] \cup [4.8, 5.2]$ we can see that this is not a valid generalisation since it is much too restricted. $\Delta(4,5)$ satisfies Condition 1, but not Condition 2 because $d(4.5,5) + d(4.5,4) = d(4,5)$, and 4.5 is not inside the generalisation.

[1] A metric space (X,d) is a set of points with an associated distance function (also called a metric) $d : X \times X \rightarrow \Re$ which satisfies the following conditions: $\forall x, y \in \Re$, $d(x,y) \geq 0$, $d(x,y) = 0$ if and only if $x = y$ (identity of indiscernibles), $d(x,y) = d(y,x)$ (symmetry), $d(x,z) \leq d(x,y) + d(y,z)$ (triangle inequality).

3 Suitable Generalisation for Several Data Types

In this section we study some generalisation operators for several well-known data types: nominal, numerical, sets and lists. First, we define a distance function d over each data type T such that the pair (T, d) is a metric space. Then, we define a binary operator which generalises a pair of elements of T and we show that this function verifies the conditions defined in Section 2. Obviously, the scope condition establishes the dependency between a generalisation and the underlying distance function. In fact, in some cases, it is possible to define a function over a data type which is a generalisation operator if we use a certain metric, but it does not verify the definition using a different metric. We illustrate this point by means of an example over lists.

3.1 Nominal Data Types

A nominal or discrete data type T is a finite collection of values $\{a_1, \ldots, a_n\}$ such that $a_i \neq a_j$ for all $i \neq j$. For instance, $\{red, yellow, blue\}$ is a nominal data type. Let d be the discrete metric defined as $d(a, b) = 0$ if $a = b$ and $d(a, b) = 1$ if $a \neq b$, where $a, b \in T$. Then, (T, d) is a metric space. The next proposition shows that the set of two discrete values is an admissible generalisation for this data type.

Proposition 1. *Given $T = \{a_1, \ldots, a_n\}$ a discrete data type and d the discrete distance, then the operator Δ defined as $\Delta(a_i, a_j) = \{a_i, a_j\}$ is a 1-generalisation.*

Proof. We first prove the no isolation condition. Any ball centered in a_i (equivalently in a_j) which verifies the premise of this condition contains, at least, a_j (equivalently a_i), by the definition of d. But a_j (equivalently a_i) also belongs to $\Delta(a_i, a_j)$ (by definition of $\Delta(a_i, a_j)$). Therefore, $a_j \in B(a_i, \epsilon) \cap \Delta(a_i, a_j)$, which verifies condition 1.

Also, $\Delta(a_i, a_j)$ trivially satisfies condition 2 since there does not exist any $x \in T$ (different from a_i and a_j) which verifies $d(a_i, x) + d(x, a_j) = d(a_i, a_j)$ because the definition of the distance.

The generalisation derived from this operator resembles the "explanation" computed by decision tree learning algorithms such as $ID3$ or $C4.5$ [8] for nominal splits.

3.2 Numerical Data Types

The next data type we consider is the real number set $\Re$. The usual distance function over $\Re$ is defined as $d(a, b) = \mid a - b \mid$. In this case, the pair $(\Re, d)$ is also a metric space.

There are many ways of generalising a pair of real numbers, but not all of them are sensible generalisations in the sense of Definition 1. The following proposition shows that given a pair of real numbers a and b, $a \leq b$, the interval $[a, b]$ is a good generalisation.

Proposition 2. *Let $\Re$ be the real number set with the usual metric and let $a, b \in \Re$ such that $a \leq b$. Then $\Delta(a, b) = [a, b]$ is a 1-generalisation.*

Proof. Any ball centered in any of the interval limits contains real numbers which belong to the interval. Hence, $\Delta(a, b)$ verifies condition 1. On the other hand, any $x \in \Re$ that verifies $d(a, x) + d(x, b) = d(a, b)$ (premise of condition 2) also belongs to the interval $[a, b]$, which proves that condition 2 is satisfied.

Other possible generalisations also based on the interval concept are, for instance, $\Delta'(a, b) = [a - \mid a - b \mid, b]$ or $\Delta''(a, b) = [a - \delta, b + \delta]$, $\delta \geq 0$. In this last case, if $\delta = (k - 1) \mid a - b \mid /2$, then it is a k-generalisation. Note that if c is the upper bound of this interval, then $d(a, c) + d(c, b) \leq k \cdot d(a, b)$.

3.3 The Set Data Type

Let $\Sigma = \{a_1, a_2, \ldots\}$ be a set of items (not necessarily finite). Let us consider the set of all finite sets over Σ, denoted by S_Σ, and define the function $d : S_\Sigma \times S_\Sigma \rightarrow \Re$ as the cardinality of the symmetric set difference between two sets belonging to S_Σ. Then, the pair (S_Σ, d) is a metric space. Now, given $A, B \in S_\Sigma$ we define $\Delta(A, B)$ as,

$$\Delta(A, B) = \{C \in S_\Sigma : A \cap B \subseteq C\}$$

Note that, as follows from the definition of $\Delta(A, B)$, if A and B are two disjoint sets, then $\Delta(A, B) = S_\Sigma$.

Proposition 3. *The above operator Δ defined over sets is a 1-generalisation.*

Proof. According to the introduced metric, the minimum permitted distance between two different sets is equal to 1. Hence, it is sufficient to prove that condition 1 holds for balls that contain sets which are at the minimum distance. In this case, for any V in $\Delta(A, B)$, it is possible to build a set W which keeps at distance 1 and belongs to $\Delta(A, B)$, by only inserting a new item into V. Then, the first condition of the generalisation definition holds.

Let us glance at the second one. We prove it by contradiction. Suppose that there exists a set C which verifies $d(A, C)+d(C, B) = d(A, B)$, but $C \not\subseteq \Delta(A, B)$. Then, the following relations hold:

$$\begin{gathered} d(A, C) + d(C, B) = d(A, B) \\ \Downarrow \text{ (by the definition of } d) \\ (\mid A \mid + \mid C \mid -2 \mid A \cap C \mid) + (\mid B \mid + \mid C \mid -2 \mid B \cap C \mid) = \\ \mid A \mid + \mid B \mid -2 \mid A \cap B \mid \\ \Updownarrow \text{ (by simplification)} \\ \mid C \mid = \mid A \cap C \mid + \mid B \cap C \mid - \mid A \cap B \mid \quad (1) \end{gathered}$$

On the other hand, if C verifies $d(A, C) + d(C, B) = d(A, B)$ it is possible to transform A into B going through C. Then, C contains at least some elements that also belong to A and some elements that also belong to B. Hence,

$$\mid C \mid \geq \mid A \cap C \mid + \mid B \cap C \mid - \mid A \cap B \cap C \mid$$

where $\mid A \cap B \cap C \mid$ is the number of elements in C belonging to both A and B. By replacing this expression in (1), we obtain

$$\begin{gathered} \mid A \cap C \mid + \mid B \cap C \mid - \mid A \cap B \mid \geq \mid A \cap C \mid + \mid B \cap C \mid - \mid A \cap B \cap C \mid \\ \Updownarrow \text{ (by simplification)} \\ \mid A \cap B \mid \leq \mid A \cap B \cap C \mid \quad (2) \end{gathered}$$

By hypothesis, $C \not\subseteq \Delta(A, B)$. Then $A \cap B$ is not included in C and $\mid A \cap B \cap C \mid = 0$. But then, $\mid A \cap B \mid = 0$ by (2) which implies that $\Delta(A, B) = S_\Sigma$. Therefore, $C \subseteq \Delta(A, B)$, which contradicts the hypothesis.

3.4 The List Data Type

The last data type we deal with is the list or sequence data type, i.e. words constructed from symbols of a finite alphabet $\Sigma = \{a_1, \ldots, a_n\}$. Before defining a generalisation operator for this data type, we introduce some preliminar concepts.

Definition 2. *The alignment of two words $s, t \in \Sigma^*$ is the process of juxtaposing them such that there exists matched symbols.*

For instance, if $s = abc$ and $t = bca$, there are two alignments of s and t.

$$\begin{array}{lll} (i) & a\ b\ c & (ii)\ a\ b\ c \\ & b\ c\ a & \qquad\quad b\ c\ a \end{array}$$

Definition 3. *Given two words s and t, the edit distance d between s and t is defined as the minimum number of deletions or insertions required to transform s into t.*

For instance, in the above example, $d(s, t) = 2$. The edit distance is obtained when the number of symbols matched in an alignment is maximal. If it is the case, we say that the alignment is maximal. Note that, in general, it is possible to have more than one maximal alignment. For instance, if $s = cbc$ and $t = abcbd$ then we have two maximal alignments giving $d(s, t) = 4$:

$$\begin{array}{lll} (i) & \quad c\ b\ c & (ii)\ c\ b\ c \\ & a\ b\ c\ b\ d & \qquad a\ b\ c\ b\ d \end{array}$$

By $M_{s,t}$ we denote the number of matched symbols in the maximal alignments of s and t.

If there is an alignment of s and t, then it is possible to find a more general expression which includes s and t as special cases. We call this expression pattern of s and t. More formally,

Definition 4. *Given $s, t \in \Sigma^*$, a pattern p of s and t is an expression constructed from an alignment of s and t by keeping the matched symbols and replacing the un-matched symbols by $*$.*

For instance, following with the above example, there are two patterns, $*bc*$ and $*cb*$.

Definition 5. *A pattern p covers a word w, if w can be obtained from p by replacing any ocurrence of $*$ by a (possibly empty) sequence of alphabet symbols. In this case, we say that w is an instance of p.*

Note that, it follows from this definition that a pattern represents the set of its instances.

Definition 6. *Given the metric space (Σ^*, d), where d is the edit distance, and $x, y \in \Sigma^*$, we define the operator $\Delta(x, y)$ as the set of words covered by the patterns obtained when the edit distance between x and y is calculated.*

For instance, in our example, $\Delta(cbc, abcbd) = \{bc, abc, cbca, \ldots, cb, cba, acb, \ldots\}$ and can be represented by the set of patterns $\{*bc*, *cb*\}$. In order to show that this operator is a k-generalisation, the first question to rise is whether it is necessary that $\Delta(x, y)$ contains instances of the patterns of x and y obtained from all maximal alignments or if it is sufficient only to consider one of them. We clarify this point with our example (see the left hand side of Figure 2). Given s and t, the word $w = cbd$ verifies $d(s, p) + d(p, t) = d(s, t)$ and, however, it is not an instance of the pattern $*bc*$. But $*cb*$ is also a pattern for s and t and w is an instance of it. Hence, we need to take the patterns obtained from all the maximal alignments into account.

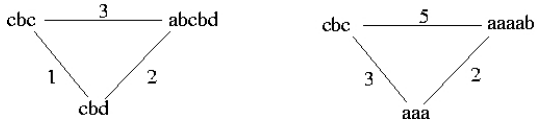

Fig. 2. (Left picture) All the maximal patterns must be considered in the generalisation.**(Right picture)** Edit distance with substitution does not work.

As we have said at the begining of this section, it is possible to define an operator which is not a generalisation w.r.t. a distance function, but indeed verifies the definition w.r.t. another distance metric. We show this fact with an example using the edit distance with substitution (right hand side of Figure 2). Let $w_1 = cbc$ and $w_2 = aaaab$ be two words. In this case, there is only one maximal alignment giving the pattern $*b*$, but $w_3 = aaa$ verifies $d(w_1, w_3) + d(w_3, w_2) = d(w_1, w_2)$ and, however, it is not an instance of this pattern.

Proposition 4. *Let (Σ^*, d) be a metric space, and let $x, y \in \Sigma^*$. Then, the operator $\Delta(x, y)$ in Definition 6 is a 1-generalisation.*

Proof. The proof is quite similar to that of proposition 3. The minimum distance between two words is 1. Then, if we extract any word, namely u, from $\Delta(x, y)$, we can find a different expression, namely v, which is at a distance 1 from u and belongs to $\Delta(x, y)$. For this purpose, it is enough to change one symbol from u which does not match with the explicit symbols in v. This proves that condition 1 holds.

Now, we prove condition 2 by contradiction. Suppose that there exists a word z such that $d(x, z) + d(z, y) = d(x, y)$ and $z \notin \Delta(x, y)$. By the definition of the edit distance, given two words s and t, $d(s, t) =\mid s \mid + \mid t \mid -2 \cdot M_{s,t}$, where $\mid s \mid$ is the length of s. Hence

$$d(x, z) + d(z, y) = d(x, y) \Leftrightarrow$$

$$\mid x \mid + \mid z \mid -2 \cdot M_{x,z} + \mid z \mid + \mid y \mid -2 \cdot M_{z,y} =\mid x \mid + \mid y \mid -2 \cdot M_{x,y} \Leftrightarrow$$

$$\mid z \mid = M_{x,z} + M_{z,y} - M_{x,y} \quad (3)$$

On the other hand, we know that

$$\mid z \mid \geq M_{x,z} + M_{z,y} - Agree_{<x,z>,<y,z>}$$

where $Agree_{<x,z>,<y,z>}$ is the number of coincidental symbols in the alignment of x with z, and y with z. By replacing this expression in (3), we obtain

$$M_{x,y} < Agree_{<x,z>,<y,z>}$$

But if $z \notin \Delta(x, y)$ then $Agree_{<x,z>,<y,z>} < M_{x,y}$ which is a contradiction. Thus, we conclude that condition 2 holds.

4 The lgg operator in metric spaces of atoms

One of the most popular generalisation operator in ILP is the least general generalisation, *lgg*, introduced by Plotkin in [7]. In this section we study the relationship between this operator and the Definition 1 in the framework of a first-order logic language. We will show that using a metric defined over this language and based on the lgg, the Δ function and the lgg are connected so that we could use the lgg as a pattern constructor over the first-order language. In order to do this, we first establish a distance over the set of atoms and then we provide a Δ definition which is also based on the lgg operator.

In that follows L denotes a first order language defined over the signature $\langle \mathcal{F}, \Pi, \mathcal{X} \rangle$ where $\mathcal{F}$ (respectively Π) is a family indexed on N (non negative integers) being $\mathcal{F}_\setminus$ (Π_n) a set of $n-$adic function (predicate) symbols and $\mathcal{X}$ is a (infinite) denumerable set of variable symbols. In case of no ambiguity predicate and function symbols will be referred as symbols, and variable symbols as variables. Also $H_\mathcal{X}$ and $B_\mathcal{X}$ denote the non-ground Herbrand Universe and the non-ground Herbrand base respectively such as is introduced in [3].

4.1 A distance based on lgg

In [9] is presented a distance between non-ground atoms such that the set of atoms in L along with this distance is a metric space. Basically, the mentioned distance between two atoms is expressed as a pair of integer values (F, V) reflecting the differences of them w.r.t. their lgg. The distance defintion is based on an auxiliar function $size(a) = (F, V)$ which reflects the structure of the atom a. Roughly speaking, F is a function which counts the number of predicate and function symbols occurring in a, and the function V returns the sum of the squared frequency of appearance of each variable in a. More formally,

Definition 7. *Given a_1 and a_2 two atoms, then*

$$d(a_1, a_2) = [size(a_1) - size(lgg(a_1, a_2))] + [size(a_2) - size(lgg(a_1, a_2))]$$

Example 3. Consider the atoms $a_1 = p(a, b)$ and $a_2 = p(b, b)$. The distance $d(a_1, a_2)$ is calculated as follows. First, we compute the lgg of both atoms, that is, $lgg(a_1, a_2) = p(X, b)$ and then, we measure each atom structure by means of the function $size$: $size(a_1) = size(a_2) = (3, 0)$ and $size(lgg(a_1, a_2)) = (2, 1^2)$. Finally, the distance between a_1 and a_2 is

$$d(a_1, a_2) = [(3, 0) - (2, 1)] + [(3, 0) - (2, 1)] = (1, -1) + (1, -1) = (2, -2)$$

Note that with this definition of distance the proximity relation (how far two atoms are) is not as intuitive as in a conventional metric space where its associated distance returns only a positive real number (and not a pair of values). For this reason, the authors introduce a total order relation over the pair of values which allows to specify a proximity notion. Given two ordered pairs $A = (F_1, V_1)$ and $B = (F_2, V_2)$, $A < B$ iff $F_1 < F_2$ or $F_1 = F_2$ and $V_1 < V_2$. Let us illustrate how this order relation can be used to determine the proximity among atoms.

Example 4. Let $a_1 = p(a, b)$, $a_2 = p(a, a)$ and $a_3 = p(b, b)$ be three atoms. Since $d(a_1, a_3) = (2, -2)$ and $d(a_2, a_3) = (4, -8)$ we can conclude according to the order relation that a_3 is closer to a_1 than to a_2.

4.2 Defining Δ

Now we are ready to define a generalisation Δ over the metric space of atoms.

Definition 8. *Given a_1 and a_2 two atoms from $B_{\mathcal{X}}$, the Δ function is defined as follows:*

$$\Delta(a_1, a_2) = \{a \in B_V : a = lgg(a_1, a_2)\sigma\}$$

where σ is a substitution and $e\sigma$ denotes the instance of an expression e by σ.

Note that with this defintion $lgg(a_1, a_2)$ is an atom that also belongs to $\Delta(a_1, a_2)$. Additionally, it is the most general atom in $\Delta(a_1, a_2)$. For these reasons we use the lgg as the canonical representant of the set Δ, or in other words, $lgg(a_1, a_2)$ would be used as the pattern representing $\Delta(a_1, a_2)$.

Next, we will prove as this Δ function verifies the conditions of the Definition 1.

In order to prove the *Scope* condition a formal problem arises. Note that this one is formulated thinking of standard metric functions, which return a positive real value. In principle, the concept of ball, $B(x, \epsilon)$, would not make sense in the current metric space. But we can address this proof thanks to the following observation. This condition restricted to discrete spaces is equivalent to say that given any a_3 belonging to $\Delta(a_1, a_2)$ then, at least, one of the nearest atoms to a_3 must belong to $\Delta(a_1, a_2)$. This alternative definition can already be managed for the current metric space using the established order relation over the set of pairs (F, V).

Before tackling the proof, some preliminary definitions and propositions are introduced. We consider the usual representation of a term as a labelled tree. Then, a position p in a term t is represented by a sequence of natural numbers. $p \cdot q$ denotes the concatenation of positions p and q.

Definition 9. *Let a be an atom, and let t_1 and t_2 be two (sub)-terms in a at positions $p = p_1 \cdot p_2 \cdot \ldots \cdot p_n$ and $q = q_1 \cdot q_2 \cdot \ldots \cdot q_m$, respectively. We will say that p is deeper than q if $n > m$. Additionally, by saying that a (sub)-term t is placed at $p \cdot *$ in a, we mean that the exact position of t has the sequence p as a prefix.*

Example 5. Given the atom $a = p(b, f(g(c)))$, the position of b in a is 1 and the position of $g(c)$ in a is $2 \cdot 1$. Thus, $g(c)$ is placed at a deeper position in a than b is. We can also say that c is placed at $2 \cdot *$ since its position is $2 \cdot 1 \cdot 1$.

Definition 10. *Let a be an atom, the skeleton of a (denoted by $sk(a)$) is just the term obtained from a by replacing any variable by a dot.*

Example 6. Given the atom $a = p(a, f(X), g(h(X)))$ then, $sk(a) = p(a, f(\cdot), g(h(\cdot)))$.

Note that the dot symbol simply informs about an unknown subterm at that position, just like an anonymous variable. Thus, the skeleton would be interpreted as an atom with all its variables different each others.

Definition 11. *Let sk_1 and sk_2 two skeletons, we will say that sk_1 and sk_2 overlap (denoted by $sk_1 \cap sk_2$), if there exists a subset of symbols in sk_1 such that each symbol of this set occurs at same position in sk_2.*

Example 7. Given $sk_1 = p(a, \cdot, g(b))$ and $sk_2 = p(\cdot, \cdot, g(b))$ then, sk_1 and sk_2 overlap.

Definition 12. *Let a_1 and a_2 two atoms. Then, we will say that the $sk(a_2)$ is a sub-skeleton of $sk(a_1)$ (denoted by $sk(a_2) \subset sk(a_1)$), if for each symbol in $sk(a_2)$, the same symbol occurs at the same position in $sk(a_1)$.*

Example 8. Given the atoms $a_1 = p(a, f(X), g(h(X)))$ and $a_2 = p(a, X, g(Y))$ then, $sk(a_2) = p(a, \cdot, g(\cdot))$ is a sub-skeleton of $sk(a_1) = p(a, f(\cdot), g(h(\cdot)))$.

Definition 13. *Let sk_1 and sk_2 be two skeletons, we will say that both ones are equal ($sk_1 = sk_2$) if $sk_1 \subset sk_2$ and $sk_2 \subset sk_1$.*

Given a_1, a_2 and a_3 three atoms such that $d(a_1, a_2) = d(a_1, a_3) + d(a_3, a_2)$, the following proposition establishes that a_3 overlaps a_1 and a_2 at the same positions as a_1 and a_2 overlap, and a_3 overlaps at the rest of positions either a_1 or a_2.

Proposition 5. *Let a_1, a_2 and a_3 be three atoms. If the identity $d(a_1, a_2) = d(a_1, a_3)+d(a_3, a_2)$ is verified then, the $sk(lgg(a_1, a_2)) \subset sk(a_3)$ and those (sub)-terms in a_3 which are not covered by the $sk(lgg(a_1, a_2))$ will be the same that those (sub)-terms placed at the same position in a_1 or in a_2.*

Proof. Let $d(a_1, a_2) = (F_1, V_1)$, $d(a_2, a_3) = (F_2, V_2)$ and $d(a_1, a_3) = (F_3, V_3)$ be the distances among a_1, a_2 and a_3 (for convenience the function $F(\cdot)$ will be applied over skeletons as well). Then, using the distance definition, F_1 can be written as $F_1 = F(a_1)+F(a_2)-2F(lgg(a_1, a_2))$, and the same thing for the rest of F_i values.

Now, by the well-known equality between tuples, we have that $F_1 = F_2 + F_3$ and operating in both sides of the equation we obtain,

$$F(a_3) - F(lgg(a_1, a_3)) - F(lgg(a_2, a_3)) = -F(lgg(a_1, a_2))$$

Note that the atom $lgg(a_i, a_j)$ contain less or equal number of symbols than a_i and a_j. So, if the skeletons of $lgg(a_1, a_3)$ (for simplicity, denoted by $sk_{1,3}$) and $lgg(a_2, a_3)$ (denoted by $sk_{2,3}$) did not have some parts in common then, the left hand side of the identity above cannot be negative. Thus, both skeletons must overlap. This overlapping will be expressed by $sk_{1,3} \cap sk_{2,3}$. Clearly, the symbols in a_3 belonging to $sk_{1,3} \cap sk_{2,3}$ are counted twice. Note that $sk_{1,3} \cap sk_{2,3}$ is equal to $sk(lgg(lgg(a_1, a_3), lgg(a_2, a_3)))$ and that it is a sub-skeleton of $lgg(a_1, a_2)$, then $F(sk_{1,3} \cap sk_{2,3}) \leq F(lgg(a_1, a_2))$. Hence, we can write the equation above as,

$$F(a_3) - F(sk^*(a_1, a_3)) - F(sk^*(a_2, a_3)) - 2F(sk_{1,3} \cap sk_{2,3}) =$$

$$= -F(lgg(a_1, a_2))$$

where $sk^*(a_1, a_3)$ (equivalently $sk^*(a_2, a_3)$) stands for that part of the skeleton of $lgg(a_1, a_3)$ (equivalently $lgg(a_2, a_3)$) which does not take part of $sk_{1,3} \cap sk_{2,3}$. If we represent the operations $F(a_3)-F(sk^*(a_1, a_3))-F(sk^*(a_2, a_3))$ by A then, A is strictly positive since $sk^*(a_1, a_3)$ and $sk^*(a_2, a_3)$ does not overlap.

The only way of forcing $A-2F(sk_{1,3}\cap sk_{2,3})$ to be negative, is that the $sk(a_3)$ can be perfectly ensambled from the skeletons of $sk^*(a_1, a_3)$, $sk^*(a_2, a_3)$ and $sk_{1,3}\cap sk_{2,3}$ (this fact implies that $A = F(sk_{1,3}\cap sk_{2,3})$) and finally, that $F(sk_{1,3}\cap sk_{2,3}) = F(lgg(a_1, a_2))$. Precisely, this last equality means that $sk_{1,3} \cap sk_{2,3} = sk(lgg(a_1, a_2))$ and that those subterms in a_3 not covered by the $sk(lgg(a_1, a_2))$ coincide with those subterms placed a the same position in a_1 or a_2. Summing up, a_3 is built from the (sub)-terms in a_1 or in a_2.

Before presenting next proposition, a preliminary concept must be introduced.

Definition 14. *Let a_1, a_2 and a_3 be three atoms such that $d(a_1, a_2) = d(a_1, a_3) + d(a_3, a_2)$. Then, we will say that a variable occurring in $lgg(a_1, a_2)$ at position p is reflected in $lgg(a_1, a_3)$ or in $lgg(a_2, a_3)$, if there exists a variable (modulo renaming) at position $p \cdot *$ in $lgg(a_1, a_3)$ or in $lgg(a_2, a_3)$.*

Example 9. In Figure 3, the variable X in $lgg(a_1, a_2)$ is reflected in $lgg(a_2, a_3)$, whereas the variable Y in $lgg(a_1, a_2)$ is reflected in $lgg(a_1, a_3)$. Note that the reflexion is not a one-to-one association. A variable in $lgg(a_1, a_2)$ could be reflected in $lgg(a_1, a_3)$ or $lgg(a_2, a_3)$ several times. For example imagine that a variable X occurs in $lgg(a_1, a_2)$ at position $2 \cdot 1$, and two variables Y and Z are placed at position $2 \cdot 1 \cdot 1$ and $2 \cdot 1 \cdot 2$ respectively in $lgg(a_1, a_3)$, then X is reflected twice in $lgg(a_1, a_3)$.

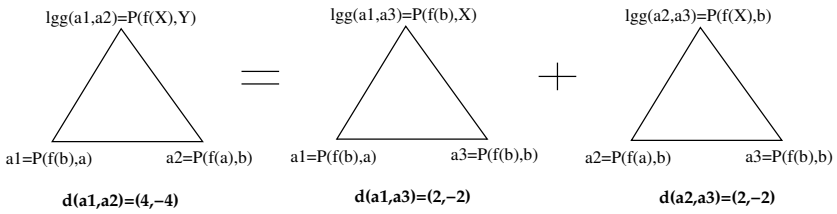

Fig. 3. An illustrative example.

The next proposition shows the relationship between the variables in the lgg's of three terms a_1, a_2 and a_3 which satisfy $d(a_1, a_2) = d(a_1, a_3) + d(a_3, a_2)$.

Proposition 6. *Let a_1, a_2 and a_3 be three atoms such that $d(a_1, a_2) = d(a_1, a_3) + d(a_3, a_2)$. Then, each variable appearing in $lgg(a_1, a_2)$ will be reflected either in $lgg(a_1, a_3)$ or in $lgg(a_2, a_3)$ only once. If a variable in $lgg(a_1, a_2)$ has multiple occurrences and one of these occurrences is reflected in $lgg(a_1, a_3)$ (equivalently $lgg(a_2, a_3)$) then, the rest of occurrences of the same variable will be also reflected in $lgg(a_1, a_3)$ (equivalently $lgg(a_2, a_3)$). Hence, $lgg(a_1, a_3)$ (equivalently $lgg(a_2, a_3)$) is a more specific atom than $lgg(a_1, a_2)$.*

Proof. The first part of the proposition is a derived consequence from Proposition 5. Recall that if $sk(lgg(a_1, a_2) \subset sk(a_3)$ then, for each variable in position $p \cdot *$ in $lgg(a_1, a_3)$ or in $lgg(a_2, a_3)$ there exists a variable (modulo renaming) which occur in $lgg(a_1, a_2)$ at position $p \cdot *$. Additionally, those terms in a_3 which are not covered by the $sk(lgg(a_1, a_2))$ coincide with those terms placed at the same position either in a_1 or in a_2. Thus, all the variables in $lgg(a_1, a_2)$ will be reflected either in $lgg(a_1, a_3)$ or in $lgg(a_2, a_3)$ only once. An immediate effect of this part of the proposition is that the number of variables in $lgg(a_1, a_2)$ is

equal to the number of variables in $lgg(a_1, a_3)$ plus the number of variables in $lgg(a_1, a_2)$.

Now let us prove the second part of the proposition (we use the function $V(a, X)$ employed in [9] which values the occurrences of variable X in the atom a.). Imagine that a variable X occurs n times in $lgg(a_1, a_2)$ then, $V(lgg(a_1, a_2), X) = n^2$. Now, let us suppose that n_1 occurrences of X are reflected in $lgg(a_1, a_3)$ and the rest of them, n_2, in $lgg(a_2, a_3)$ then, we would have $V(lgg(a_1, a_3), X) + V(lgg(a_2, a_3), X) = n_1^2 + n_2^2$. As $n = n_1 + n_2$ trivially we have $n^2 < n_1^2 + n_2^2$. So the only possibility for $V(lgg(a_1, a_2), X) = V(lgg(a_1, a_3), X) + V(lgg(a_2, a_3), X)$ to be hold is that the occurrences of any variable, namely Y, in $lgg(a_1, a_3)$ or in $lgg(a_2, a_3)$ increase. But it cannot happen because a_3 is built from (sub)-terms in a_1 and in a_2. Therefore, all the occurrences of one variable X in $lgg(a_1, a_2)$ are reflected either in $lgg(a_1, a_3)$ or in $lgg(a_2, a_3)$. This fact implies that the $lgg(a_1, a_3)$ and the $lgg(a_2, a_3)$ are more specific atoms than $lgg(a_1, a_2)$.

Now, we are ready to proof the feasibility of Δ for the current metric space.

Proposition 7. *The Δ function defined in Definition 8 is a 1-generalisation.*

Proof. – (*No Isolation*). Given an atom a, its nearest atoms are obtained by changing one of its constant (sub)-term by a variable. Calling a' to this new atom, it is trivial to see that the $lgg(a, a') = a'$ and, as we know, the number of symbols in a' is one less than the number of symbols in a whereas the number of variable occurrences in a' is one more than in a. Thus, the distance between a and a' is

$$\begin{aligned} d(a, a') &= [size(a) - size(a')] + [size(a') - size(a') \\ &= (1, -1) + (0, 0) = (1, -1) \end{aligned}$$

Therefore, given an atom a all its nearest atoms are $(1, -1)$ away. Such as the symbols are counted, the distance is not affected by the relative position of those different sub-terms between two atoms. Hence, it does not matter if the substituted (sub)-term constant symbol in a is placed at a higher or at a deeper position, the distance between a and a' will be the same. Thus, given two atoms a_1 and a_2, their generalisation $\Delta(a_1, a_2)$ and a new atom a_3 belonging to $\Delta(a_1, a_2)$, from the Definition 8 we know that there exists a substitution σ such that $a_3 = lgg(a_1, a_2)\sigma$. If we change any of the constant symbols appearing in σ by a variable (denoting this new substitution by σ'), we will obtain a new atom $a'_3 = lgg(a_1, a_2)\sigma'$ such that $d(a_3, a'_3) = (1, -1)$. Obviously, a'_3 belongs to $\Delta(a_1, a_2)$ and a'_3 is one of the nearest possible atoms to a_3.

– (*Scope*) The aim of this proof will consist of showing that if this identity $d(a_1, a_2) = d(a_1, a_3) + d(a_3, a_2)$ is preserved, where a_i are atoms, then a_3 is an instance of the $lgg(a_1, a_2)$ and consequently, a_3 belongs to $\Delta(a_1, a_2)$. By Proposition 6 it is followed that there exists a substitution σ_1 such that $lgg(a_1, a_3) = lgg(a_1, a_2)\sigma_1$. Then, by lgg definition, we know that there also exists a substitution σ_2 such that $a_3 = lgg(a_1, a_3)\sigma_2$. Hence, $a_3 = lgg(a_1, a_2)\sigma_1\sigma_2$ and a_3 belongs to $\Delta(a_1, a_2)$.

5 Conclusions

In this paper we have analysed the connection of three different, but highly connected, notions: distance, pattern and generalisation. Although these notions are extremely related, there have not been many theoretical works that have studied the relationships among these topics. This work intends to be a fist step for this purpose. We have introduced a generalisation definition to identify the representation patterns which can behave as proper generalisations in the context of an underlying metric space. We have shown that given a metric not every general pattern is a good generalisation. On the other hand, we have shown that we can have suitable generalisation patterns for the most usual metrics defined for well-known data types.

As an immediate future work, we are studying the extension of the notion to generalisation operators applied to pairs of a set and an element, thus allowing the generalisation to be applied incrementally, and the extension of the definition to the Cartesian product of different types. All this could constitute an integrated framework that can be applied to many distance-based methods. For instance, in k-nearest neighbours, we could use an incremental generalisation operator to generate patterns which would be consistent with the clustering or classification performed by the k-nearest neighbour based on the underlying distance, starting from the closest elements to the farthest. Other case-based reasoning techniques based on distances can benefit from this as well. In a similar way, we could be able to give explanations to other methods that use distances. For instance, we have defined a distance-based decision tree, where the splits are determined by metric conditions, as the centre-splitting technique does. The generalisation operators introduced in section 3 can be used to give a comprehensible representation to the partitions, and hence to the overall decision tree, even if the problem contains non-standard data types such as lists or sets. In some way, this is a general approach to obtaining comprehensible patterns when distances are computed from structured data types. For instance, in [4], they compute kernels, and hence distances, for structured data types.

Finally, we would also like to study the "nested composability" of the generalisation operator, i.e., lists of lists, or sets of lists of trees, that could deal with deeply complex structures. Other topics of research would be to analyse the associativity of the incremental generalisation operator (which we consider a minor issue for the applications considered), or the definition of a restricted generalisation if we use negative cases into account.

References

1. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, January 1991.
2. Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.
3. M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. Declarative Modeling of the Operational Behavior of Logic Languages. *Theoretical Computer Science*, 69(3):289–318, 1989.

4. T. Gartner, J. W. Lloyd, and P. A. Flach. Kernels and distances for structured data. *Machine Learning*, 57, 2004.
5. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comp. Surveys*, 31(3):264–323, 1999.
6. T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
7. G. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.
8. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
9. J. Ramon, M. Bruynooghe, and W. Van Laer. Distance measures between atoms. In *CompulogNet Area Meeting on Computational Logic and Machine Learing*, pages 35–41. University of Manchester, UK, 1998.