Preliminary Version of journal article published
in Annals of Mathematics and Artificial Intelligence

Learning with Cone-Based Geometric Models and Orthologics

Mena Leemhuis · **Özgür L. Özçep** · **Diedrich Wolter**

Received: date / Accepted: date

Abstract Recent approaches for knowledge-graph embeddings aim at connecting quantitative data structures used in machine learning to the qualitative structures of logics. Such embeddings are of a hybrid nature, they are data models that also exhibit conceptual structures inherent to logics. One motivation to investigate embeddings is to design conceptually adequate machine learning (ML) algorithms that learn or incorporate ontologies expressed in some logic. This paper investigates a new approach to embedding ontologies into geometric models that interpret concepts by geometrical structures based on convex cones. The ontologies are assumed to be represented in an orthologic, a logic with a full (ortho)negation. As a proof of concept this cone-based embedding was implemented within two ML algorithms for weak supervised multi-label learning. Both algorithms rely on cones but the first addresses ontologies expressed in classical propositional logic whereas the second addresses a weaker propositional logic, namely a weak orthologic that does not fulfil distributivity. The algorithms were evaluated and showed promising results that call for investigating other (sub)classes of cones and developing fine-tuned algorithms based on them.

Keywords Concept Learning · Knowledge Graph Embedding · Support-Vector Machine · Multi-Label Learning · Orthologic

1 Introduction

Recent approaches to knowledge-graph embeddings (KGE) [28] aim at linking quantitative data structures used in machine learning (ML), such as (low-dimensional) Euclidean spaces,

This paper is a considerably extended version of the conference paper [17].

Mena Leemhuis
University of Lübeck, Germany
E-mail: leemhuis@ifis.uni-luebeck.de

Özgür L. Özçep
University of Lübeck, Germany
E-mail: oezcep@ifis.uni-luebeck.de

Diedrich Wolter
University of Bamberg, Bamberg, Germany
E-mail: diedrich.wolter@uni-bamberg.de

to the qualitative structures of logics. The benefit of this linkage is twofold as expressed by the following desiderata for ML algorithms:

(QC) Learnt models should adhere to Qualitative Constraints stated in a background knowledge base or in an ontology.

(LR) It should be possible to identify on top of the learnt models emergent logic-qualitative structures, thereby enabling Logic-based Reasoning.

A well-known example of an embedding-based approach fulfilling (LR) is TransE [3]. TransE interprets concepts as vectors and functional relations as vector translations in Euclidean space. Reasoning can then be performed by geometric operations, for example the translation representing "female form of" that links concepts "queen" and "king" can be applied to identify the female form of "man".

While TransE and similar embedding approaches present an important step towards linking logics with ML, they fall short of being *fully expressive* in the sense of [19], i.e., they expose serious shortcomings w.r.t. the logic of compound concepts that can be expressed alongside atomic concepts. In case of TransE [3] limitations arise from the fact that translations are functions and thus only functional relations can be expressed. Recent embedding approaches [19, 14, 16] try to remedy this by exploring a more logic-like embedding of concepts in the following sense: Embeddings correspond to logical structures in the Tarskian style semantics—with the exception that the domain is fixed to some continuous spaces and the extensions of relations and concepts are geometrically-shaped sets in this space, i.e., geometrical objects of a specific class. In the case of [14] the class of geometric objects is that of convex sets, which allows a fragment of Datalog to be represented; [16] use hyperspheres and obtain the lightweight description logic $\mathscr{E}\mathscr{L}^{++}$.

Those recent embedding approaches (and also to some extent TransE [3]) are considered to present a tremendous progress towards finding and exploiting emergent logical structures in the data in comparison to classical ML algorithms and classifiers based on neural networks or support vector machines (SVMs). But is this really the case? Is an explicit reference to logics and/or logical structures really necessary to enable logical constraining and reasoning as expressed by (QC) and (LR)? Maybe, classical approaches such as SVMs implicitly exhibit logical structures that could be used to fulfill (QC) and (LR)? The answer we want to justify in this paper is "No!" and rests on the following working hypothesis:

(WH) *One needs to account for a logic with an appropriate class of logico-geometrical structures during the training phase in order to design a learning algorithm that fulfils (QC) or (LR).*

In this paper we argue for (WH) along two lines and sketch those two lines in the rest of this introduction: First, if one tries to interpret a trained classifier by means of logical structures, counter-intuitive results with respect to (QC) and (LR) emerge. Second, by choosing a specific class logico-geometrical structure we are able to construct an approach that fulfills (QC) and (LR). The majority of paper addresses the second line by extending our previous work [20, 17] to develop a logico-geometric approach to classification that ensures prior logic information to be respected. Such approach can be particular useful where classifiers need to be geared towards avoiding false positives. Before we embark on the technicalities of our approach, we explain why the first line of arguments—problematic grounding of logical reasoning in trained classifiers—presents a serious shortcoming we must overcome in order to achieve hybrid AI approaches capable of learning and reasoning.

Let us consider the task of multi-label learning. Any approach to multi-label learning must implicitly capture the inherent logic structure underlying the training data. Classical

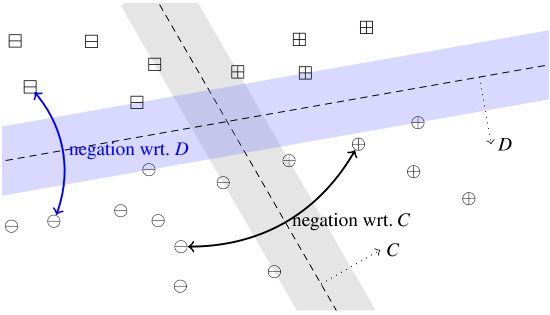

Fig. 1 SVM classification by hyperplanes as an example of missing homogeneity in classifiers

multi-label learning proceeds by setting the label $y_i = 1$ for an instance a if label i is relevant for a and $y_i = 0$ if it is not relevant [12]. Clearly, such approach cannot cope with negative information in the sense of the absence of a label is indicative for some class. To incorporate this kind of information, a classifier is required that differentiates $y_i = +1$ (definitive in class), $y_i = -1$ (definitive not in class) and $y_i = 0$ (not relevant). We refer to this kind of negation as *atomic*. If negation can be applied to arbitrary terms like $\neg(A \vee B)$, we speak of *full negation* While atomic negation can be accomplished with simple techniques such as SVM with a neutral class in-between positive and negative instances, full negation cannot be achieved and severe limitations remain. Let us consider a simple example depicted in Figure 1 showing a classification achievable when using a neutral class. For ease of presentation we assume the classifier is a SVM using a simple linear kernel. The actual problem also arises with complex kernels and also with other classifiers.

Figure 1 shows two concepts C and D with the hyperplanes determined by a SVM (dashed lines) that separates the positive and negative training data for the individual concepts. Data points labeled with $+$ belong to C, labels enclosed by a circle belong to D. A point labeled $\boxplus$ thus is within C and outside D. Consider the conjunction of concepts C and D which is naturally represented by the set intersection of the concepts C and D. The geometrical structure of half-spaces as used by linear kernel SVMs lacks homogeneity in the sense that it is not closed under logic operations, i.e., it cannot represent $C \wedge D$. Clearly, we could construct a specific kernel to single out $C \wedge D$, but this will only shift the problem towards other logic combinations which cannot be represented. In other words, classifiers whose geoemtric structures lack homogeneity show shortcomings when trying to learn multiple concepts which are interrelated by an underlying logic structure. Note that in order to support arbitrary concept conjunctions and disjunctions, full negation is required as both are tied together by DeMorgan's law.

The problem discussed above applies to any form of classifier in which geometric and logic structures are not balanced. Limitations exist with a wide range of approaches that embed logic structures geometrical since full negation, which is an important building block in most logics, is hardly supported. Contemporary approaches either completely lack the treatment of negation (such as TransE [3] and its successors) or provide only atomic negation [30,8,16] or only go little beyond, for example by supporting disjointness constraints [14]. We note that already Gärdenfors [10, p. 202] considered the representation of negation (and quantification) as particularly difficult.

As for the second line of argument for the working hypothesis we are going to show that cone-based models feature homogeneity and show how cone-based models can be learnt in algorithms that fulfil (QC) and (LR). Cone-based models allow representing logics within the band from classical propositional logic at one extreme (having say distribution of "and" over "or"); minimal orthologic [13] (missing distribution) at the other extreme, and much in between (in particular quantum logics [22] supporting a weakening of distributivity, namely orthomodularity).

Because cone-based models induce orthologics they support full (ortho)negation of orthologics, i.e., a negation that fulfills antitonicity (contraposition), the intuitionistic absurdity principle (anything follows from a sentence stating A and its negation) and double negation elimination. Moreover, weaker orthologics (i.e. those weaker than classical Boolean logics) allow modeling uncertainty and partial information. This is a clear benefit for multi-label learning scenarios as typically many entities are undetermined w.r.t. a given label. For example, one would like to refrain from assigning a class label like "can swim" or its negation "cannot swim", if neither evidence is given in the training data.

The rest of the paper is structured as follows: In Sect. 2 we recap necessary fundamentals of lattice theory, orthologics, and convex cones. Sect. 3 is the main theoretical contribution: it introduces the cone structure on which our embedding is based. Sect. 4 develops the main embedding algorithms and Sect. 5 presents an evaluation. After a discussion of related work in Sect. 6 we conclude the paper in Sect. 7.

2 Preliminaries

The class of logics that are in the focus of this paper are orthologics [13]. These characterize the class of ortholattices and thus motivate us to develop the theory of cones based on ortholattice ground. Therefore, we now give a short introduction to lattices and ortholattices. This is followed by the necessary bits on orthologics. We draw a link of orthologics to description logics (DL), which are the favoured class of logics to represent ontologies. Further we discuss the geometrical notions, in particular that of cones, that we are going to use as interpretations of the logics for embeddings.

2.1 Lattices

A *lattice* $(L, \leq)$ is a structure with a partial order $\leq$ such that for any pair of elements $a, b \in L$ there is a smallest upper bound $a \vee b$ and a largest lower bound $a \wedge b$. It is *bounded* if it contains a smallest element $\mathbb{0}$ and a largest element $\mathbb{1}$. A lattice is called *distributive* iff for all $a, b, c \in L$: $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ (and dually: $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$).

The binary *modularity relation* on a lattice [9], $(L, \leq)$ is defined for all $a, b \in L$ as follows: $M(a,b) :\Leftrightarrow \forall c \leq a : a \wedge (b \vee c) = (a \wedge b) \vee c$. A pair (a,b) is said to be a *modular pair* iff $M(a,b)$ holds. A lattice is called *modular* iff $M(a,b)$ for all a, b. The modularity relation generalises the property of distributivity.

In a lattice an element a^* is called a *complement of a* iff $a \wedge a^* = \mathbb{0}$ and $a \vee a^* = \mathbb{1}$. A lattice is said to be *complemented (uniquely complemented)* iff each a has a complement (has exactly one complement). A bounded lattice L is called an *ortholattice* iff it has an orthocomplement $\cdot^{\perp}$, i.e., a function such that for all $a, b \in L$ the following three conditions hold:

- $a \leq b$ entails $b^{\perp} \leq a^{\perp}$ (antitonicity)

Axioms

$A \vdash A \qquad A \mathbin{\&} B \vdash A \qquad A \mathbin{\&} B \vdash B \qquad A \dashv\vdash {\sim}{\sim}A$

$A \mathbin{\&} {\sim}A \vdash B \qquad A \vee B \dashv\vdash {\sim}({\sim}A \mathbin{\&} {\sim}B)$

Rules

$$\frac{A \vdash B,\ B \vdash C}{A \vdash C} \qquad \frac{A \vdash B,\ A \vdash C}{A \vdash B \mathbin{\&} C} \qquad \frac{A \vdash B}{{\sim}B \vdash {\sim}A}$$

Fig. 2 Minimal Orthologic *Omin*

- $a^{\perp\perp} = a$ (double negation elimination)
- $\mathbb{0} = a \wedge a^{\perp}$ (intuitionistic absurdity)

Any ortholattice satisfies de Morgan's laws, i.e., for any $a, b \in L$ it holds that $(a \wedge b)^{\perp} = a^{\perp} \vee b^{\perp}$ (and dually: $(a \vee b)^{\perp} = a^{\perp} \wedge b^{\perp}$). Roughly, ortholattices can be understood as Boolean algebras without the distributivity rule. An ortholattice is called *orthomodular* iff the following condition [22, pp. 35] holds:

(OMr) If $a \leq b$ and $a^{\perp} \leq c$ then $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$. (orthomodularity)

Clearly any modular ortholattice is also orthomodular.

2.2 Orthologics

The class of logics that are in the focus of this paper are orthologics [13]. Let $P = \{P_i \mid i \in \mathbb{N}\}$ be a set of proposition symbols and assume that we have logical symbols for binary conjunction &, unary negation $\sim$, and binary disjunction $\vee$. The set of propositional formulae $Fml(P)$ over P is defined as usual. A, B, C stand for propositional formulae in $Fml(P)$.

We consider a deduction calculus with a derivability relation $\vdash$. We use the short notation $A \dashv\vdash B$ for $A \vdash B$ and $B \vdash A$. Moreover, for a finite set of formulae $\Gamma = \{B_1, \ldots, B_n\}$ the notation $\Gamma \vdash A$ is a shorthand for $B_1 \mathbin{\&} \cdots \mathbin{\&} B_n \vdash A$. The calculus of *minimal orthologic Omin* according to Goldblatt [13] is given in Fig. 2. Any logic extending *Omin* is called an orthologic. Minimal orthologic *Omin* can be shown to exactly characterize the class of ortholattices (= the class of lattices with an orthonegation) as formally described in Proposition 1: this proposition states that an entailment holds in *Omin* if and only if it holds for the class of all ortholattices. A typical instance of the family of orthologics is classical propositional logic, which has a Boolean negation operator and fulfills the law of distributivity of the and-junctor over the or-junctor (and vice versa). Other prominent orthologics are quantum logics, which fulfill a weakening of distributivity, the orthomodularity rule.

Any logic $\mathscr{L}$ containing the rules of Fig. 2 is called an *orthologic*. For any orthologic the well-known Lindenbaum-Tarski construction leads to an ortholattice: The binary relation $\dashv\vdash$ can be shown to be an equivalence relation inducing for each formula C an equivalence class $[C]$. Define operations $\wedge$, $\vee$, $^{\perp}$ on the equivalence classes by setting $[C] \wedge [D] = [C \mathbin{\&} D]$, $[C] \vee [D] = [C \vee D]$ and $[C]^{\perp} = [{\sim}C]$. These yield an ortholattice.

Goldblatt [13] defines the semantics of orthologics based on a structure $(X, \perp)$ called an *orthoframe*. It consists of a domain/carrier X and a binary *orthogonality relation* $\perp \subseteq X \times X$, i.e., a relation that is irreflexive and symmetric. An orthoframe induces an operation $(\cdot)^{*}$ over subsets $Y \subseteq X$ defined by $Y^{*} = \{x \in X \mid x \perp Y\} = \{x \in X \mid x \perp y \text{ for all } y \in Y\}$. Observe the

Table 1 Syntax and semantics for Boolean $\mathscr{ALC}$ for an interpretation $\mathscr{I}$

Name	Syntax	Semantics
top	$\dot{\top}$	Δ
bottom	$\dot{\bot}$	$\emptyset$
conjunction	$C \dot{\sqcap} D$	$(C \dot{\sqcap} D)^{\mathscr{I}} = \mathscr{I}(C \dot{\sqcap} D) = C^{\mathscr{I}} \cap D^{\mathscr{I}}$
disjunction	$C \dot{\sqcup} D$	$(C \dot{\sqcup} D)^{\mathscr{I}} = \mathscr{I}(C \dot{\sqcup} D) = C^{\mathscr{I}} \cup D^{\mathscr{I}}$
negation	$\dot{\neg} C$	$(\dot{\neg} C)^{\mathscr{I}} = \mathscr{I}(\dot{\neg} C) = \Delta^{\mathscr{I}} \setminus C^{\mathscr{I}}$

correspondence to polarity of cones. A set $Y \subseteq X$ is called $\perp$-*closed* iff $Y = Y^{**}$. This says that if $x \notin Y$ then there is a z such that not $x \perp z$, and for all $y \in Y$: $z \perp y$.

An *orthomodel* for a logic $\mathscr{L}$ over $Fml(P)$ is defined as a structure $\mathscr{I} = (X, \perp, (\cdot)^{\mathscr{I}})$ such that $(X, \perp)$ is an orthoframe and $(\cdot)^{\mathscr{I}}$ assigns to each $P_i \in P$ a $\perp$-closed set over X. In a natural way one can extend the assignment function to arbitrary formulae $(A \,\&\, B)^{\mathscr{I}} = (A)^{\mathscr{I}} \cap (B)^{\mathscr{I}}$, $(\sim A)^{\mathscr{I}} = ((A)^{\mathscr{I}})^{*}$. ($\vee$ is treated by de Morgan's law). The *semantical entailment relation* $\models$ then can be defined as $\mathscr{I} : A \models B$ iff $(A)^{\mathscr{I}} \subseteq (B)^{\mathscr{I}}$. If U is a class of orthoframes, then $A \models_U B$ means that $\mathscr{I} : A \models B$ for any orthomodel definable in any orthoframe in U.

Using a canonical-model construction, Goldblatt establishes the following (soundness and) completeness result for the class θ of all orthoframes:

Proposition 1 **([13])** *$\Gamma \vdash_{Omin} A$ iff $\Gamma \models_{\theta} A$.*

2.3 Description Logics

The family of description logics (DLs) is a family of variable-free fragments of FOL that are designed, in particular, for the representation of ontologies. Hence, any DL vocabulary (signature) contains a set of constants N_c in order to talk about individuals, a set of concept names N_C to talk about concepts, i.e., unary predicates and role names (corresponding to binary relations). Then any DL defines how to construct (complex) concepts based on this vocabulary, how to set up a terminology relating those concepts in inclusion axioms and how to set up assertions about the instances of concepts and the roles.

In this paper, we will completely neglect roles. (For a discussion of full $\mathscr{ALC}$ we refer the readers to [20]). As a result, the syntax that we consider is the propositional part of the semi-expressive DL $\mathscr{ALC}$ [2] and we term $\mathscr{ALC}$-concepts without roles *Boolean $\mathscr{ALC}$-concepts*. The set of Boolean $\mathscr{ALC}$ concepts C is defined according to the following context-free grammar:

$$C \to A \mid \dot{\bot} \mid \dot{\top} \mid C \dot{\sqcup} C \mid C \dot{\sqcap} C \mid \dot{\neg} C, \tag{1}$$

with atomic concepts $A \in N_C$ and an arbitrary concepts C. Note that A is meant as a meta-variable for atomic concept symbols. We also allow B and indexed variants as atomic concept symbols. Next to C we also consider D and indexed variants as meta-variables for complex concepts.

An $\mathscr{ALC}$ interpretation $(\Delta, \cdot^{\mathscr{I}})$ consists of the domain Δ (the space of possible elements) and an interpretation function $\cdot^{\mathscr{I}} = \mathscr{I}(\cdot)$ mapping constants to elements in Δ and concept names to subsets of Δ. The semantics of arbitrary concepts is given in Table 1.

An ontology $\mathscr{O}$ is a pair $(\mathscr{T}, \mathscr{A})$. The *terminological box (tbox)* $\mathscr{T}$ contains general concept inclusions of the form $C \dot{\sqsubseteq} D$ stating that C is a subconcept of D, for arbitrary concepts C and D. The *assertional box (abox)* $\mathscr{A}$ consists of facts of the form $C(i)$, $i \in N_c$, which says that individual i is in the extension of C.

The link to orthologics that we exploit here is that the tbox can be read as entailment relations in an orthologic. In detail: The tbox can be associated with some transformation α with an orthologic along the following lines. With each concept symbol A one can associate a propositional symbol $\alpha(C) = P_A$. This association now can be extended to arbitrary concepts by associating concept constructors with their corresponding propositional connectives. Concretely: $\alpha(C \dot{\sqcap} D) = \alpha(C)\ \&\ \alpha(D)$; $\alpha(C \dot{\sqcup} D) = \alpha(C) \vee \alpha(D)$; and $\alpha(\dot{\neg} C) = {\sim}\alpha(C)$. An inclusion axiom $C \dot{\sqsubseteq} D$ then is associated with an entailment relation $\vdash$ by $\alpha(C \dot{\sqsubseteq} D) = \alpha(C) \vdash \alpha(D)$. For example, an inclusion axiom of the form $A_1 \dot{\sqcap} \dot{\neg} A_2 \dot{\sqsubseteq} B$ is associated with the entailment $P_{A_1}\ \&\ {\sim}P_{A_2} \vdash P_B$.

Propositional $\mathscr{ALC}$ with the semantics according to Table 1 then corresponds to classical propositional logic, i.e. the orthologic which allows for distributivity of & over $\vee$ and vice versa. The main point in our paper is that we will allow other interpretations of the Boolean $\mathscr{ALC}$ concept constructors that correspond to weaker orthologics: Whereas the interpretation of the concept-and $\dot{\sqcap}$ will still correspond to intersection, the interpretation of the negation $\dot{\neg}$ will not be that of set complement and the interpretation of the concept-or $\dot{\sqcup}$ will not be that of set union. The general definition of interpretation according to Definition 1 allows to embed concepts with nodes in an ortholattice. Concept inclusion $\dot{\sqsubseteq}$ as used in tboxes then is associated with an entailment relation $\vdash$ that fulfils the general axioms and constraints of orthologics (see Figure 2) but not necessarily, say, the distributivity axiom.

Due to the correspondence with orthologics one may ask whether we choose to work with DL concepts at all and not directly with orthologics? The main reason is the distinction between the terminological box and the assertional box offered by DLs. With the assertional box we have additional means to refer to elements in the domain and thereby require concepts to contain some individuals. This is particularly useful for the supervised learning scenarios where we want to learn concepts based on training examples which are known to belong to specific concepts.

2.4 Cones

We consider geometric objects in finite dimensional Euclidean spaces $\mathbb{R}^n$ which are equipped with a dot product $\langle \cdot, \cdot \rangle$. For any $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n) \in \mathbb{R}^n$ the *dot product* is defined as $\langle x, y \rangle = \Sigma_{1 \leq i \leq n} x_i \cdot y_i$. The dot product induces a norm for vectors $||x|| = \langle x, x \rangle$, and hence a metric $d(x,y) = ||x - y||$. Based on a metric d one can define the open balls $B_{\varepsilon,x} = \{y \mid d(x,y) < \varepsilon\}$, which in turn lead to the notion of *open* sets O (for each $x \in O$ there is a ball $B(\varepsilon,x) \subseteq O$ contained in O) and *closed* sets (= complements of the open sets).

A *face* of a convex polytope P is defined either as the polytope P itself or as $P \cap h$ where h denotes a hyperplane for which P is fully contained in one of the closed halfspaces determined by h [18]. A $(d-1)$-dimensional face of a d-dimensional polytope is called *facet* [18]. In the following, the notion of a facet is used for not necessarily convex polytopes. Therefore, it is slightly adapted: A *facet* of P is defined as $P \cap h$ where h denotes a hyperplane for which the greatest convex polytope $X \subseteq P$ containing $P \cap h$ is fully contained in the closed halfspace determined by h. A *convex cone* a is a set such that from $x, y \in a$ it follows that $\lambda x + \mu y \in a$ for any $\lambda, \mu \in \mathbb{R}_{\geq 0}$. We consider convex cones that are closed in the canonical topology of $\mathbb{R}^n$ as defined above. One of the nice properties of closed convex cones is that they allow a polarity operation that takes the role of an orthocomplement.

The *polar cone* [23] a° for a is defined for Euclidean spaces with a dot product $\langle \cdot, \cdot \rangle$ as $a^\circ = \{x \in \mathbb{R}^n \mid \forall y \in a : \langle x, y \rangle \leq 0\}$.

Now consider the subset-relation $\sqsubseteq = \subseteq$ on closed convex cones in $\mathbb{R}^n$ as a partial order. Closed convex cones are closed under set intersection, so $\cap$ is a meet operator $\wedge$ w.r.t. $\leq$. Closed convex cones are not closed under set union. Instead they have to be closed up by the conic hull operator. The *conic hull* of a set b, for short $conH(b)$, is the smallest convex cone containing b. So, we can define the join operation $\vee$ by $a \vee b = conH(a \cup b)$. Considering $\mathbb{R}^n$ as the largest lattice element $\mathbb{1}$ and $\{0\}$, for $0 \in \mathbb{R}^n$, as the smallest lattice element $\mathbb{0}$ makes the resulting structure a bounded lattice.

The polarity operator for closed convex cones fulfills the properties of an orthocomplement. Hence the set of all closed convex cones (over $\mathbb{R}^n$) forms an ortholattice. As de Morgan's laws hold in any ortholattice, one gets in particular the following characterization of the conic hull: $conH(a \cup b) = (a^\circ \cap b^\circ)^\circ$. We denote the set of all closed convex cones in $\mathbb{R}^n$ by $\mathscr{C}_n$.

Proposition 2 *For any $n \geq 1$, $\mathscr{C}_n$ is an ortholattice.*

We use dedicated symbols for the signature of ortholattices when we talk about the ortholattice of closed convex cones: $\sqsubseteq = \subseteq$ stands for the lattice order $\leq$, $\sqcap = \cap$ stands for lattice meet $\wedge$, $\sqcup$ stands for lattice join $\vee$, and $^\circ$ stands for orthocomplement $^\perp$, $\top\!\!\top = \mathbb{R}^n$ stands for the largest element $\mathbb{1}$ and $\perp\!\!\perp = \{0\}$ stands for the smallest element $\mathbb{0}$.

The following proposition shows that cones are structures that naturally arise when an orthogonality relation is constrained to rely on an arbitrary symmetric positive semidefinite bilinearform (e.g., scalar product) since $\perp$-closure enforces cones.

Proposition 3 *For a given vector space V (over $\mathbb{R}$ or any other field with an order $\leq$) and a symmetric positive semidefinite bilinearform $\langle \cdot, \cdot \rangle$ let $X = V \setminus \{\vec{0}\}$ and define for all $u, v \in X$: $u \perp v$ iff $\langle u, v \rangle \leq 0$. Then: $(X, \perp)$ is an orthoframe and the $\perp$-closed sets are closed convex cones (without $\vec{0}$).*

Proof Due to symmetry/positive semidefiniteness of $\langle \cdot, \cdot \rangle$ the relation $\perp$ is symmetric/irreflexive. Now consider a $\perp$-closed set Y. We have to show that for all $\lambda \geq 0$ and $y \in Y$ also $x := \lambda y \in Y$ (or $\lambda y = \vec{0}$) and with $u, v \in Y$ also $u + v \in Y$ (or $u + v = \vec{0}$). So let $\lambda \geq 0$ and $y \in Y$. Further assume $\lambda y \neq \vec{0}$, thus $\lambda \neq 0$. Due to $\perp$-closure there is a y' with $y' \perp Y$ and $y' \not\perp x$. But this means in particular $\langle y', y \rangle \leq 0$ and $\langle y', x \rangle = \langle y', \lambda y \rangle = \lambda \langle y', y \rangle > 0$. As $\lambda \neq 0$ this is a contradiction. Now let $u, v \in Y$ and $\lambda \geq 0$ and assume that $u + v \neq \vec{0}$. So u is not $-v$. Towards contradiction assume $x := u + v \notin Y$. Due to $\perp$-closure there is a y' with $y' \perp Y$ and $y' \not\perp x$. The first means in particular we have $\langle y', u \rangle \leq 0$ and $\langle y', v \rangle \leq 0$; the latter means that $\langle y', u + v \rangle = \langle y', u \rangle + \langle y', v \rangle > 0$, a contradiction.

3 Cone-Based Geometric Models

We are going to tackle the problem of multi-label classification by following the general idea of knowledge graph embedding: interpreting labels (concepts) with geometric objects in some continuous space and using the regularities in the geometric space to predict (new) instances of a concept. To do so we define in this section the kinds of geometric objects we are going to use. Thus, assume we have given DL vocabulary σ without roles, i.e., only concept symbols N_C and individual constants N_c. We assume that there is some continuous space E (such as $\mathbb{R}^n$) and a class of geometric objects Δ defined over E and equipped with operations that lead to a bounded ortholattice: $\mathscr{K} = (\Delta, \wedge, \vee, \cdot^{\perp}, \mathbb{1}, \mathbb{0})$ over Δ equipped with three operators, two binary operators $\wedge, \vee$, a unary operator $\cdot^{\perp}$ (the latter used in postfix notation) and two constants $\mathbb{1}, \mathbb{0}$. Each member of $\mathscr{K}$ is some object defined over Δ and all of them share some geometrically defined property. Moreover, we require this algebra to be a bounded ortholattice so that we have the notion of an ortholattice order $\leq$.

Definition 1 A *geometric interpretation* w.r.t. σ and $\mathscr{K}$ is defined to be a structure $\mathscr{I} = (\Delta^{\mathscr{I}}, (\cdot^{\mathscr{I}}))$ such that

- each $i \in N_c$ is mapped to an element in Δ
- each $C \in N_C$ is mapped to an element in $\mathscr{K}$

A geometric interpretation is extended to arbitrary concepts by

- $(\dot{\top})^{\mathscr{I}} = \mathbb{1}$
- $(\dot{\bot})^{\mathscr{I}} = \mathbb{0}$
- $(C \mathbin{\dot{\sqcap}} D)^{\mathscr{I}} = C^{\mathscr{I}} \wedge D^{\mathscr{I}}$
- $(C \mathbin{\dot{\sqcup}} D)^{\mathscr{I}} = C^{\mathscr{I}} \vee D^{\mathscr{I}}$, and
- $(\dot{\neg} C)^{\mathscr{I}} = (C^{\mathscr{I}})^{\perp}$.

We define *satisfaction/modelling* relation $\models$ as follows for abox axioms: $\mathscr{I} \models C(i)$ iff $\mathscr{I}(i) \in \mathscr{I}(C)$. For the interpretation of tbox axioms we exploit the order of the ortholattice: $\mathscr{I} \models C \mathrel{\dot{\sqsubseteq}} D$ iff $C^{\mathscr{I}} \leq D^{\mathscr{I}}$.

Having the definition of a geometric interpretation we can define in a usual way what it means that a geometrical interpretation $\mathscr{I}$ is a model of an ontology $(\mathscr{T}, \mathscr{A})$, namely $\mathscr{I} \models (\mathscr{T}, \mathscr{A})$ iff $\mathscr{I} \models ax$ for all $ax \in \mathscr{T} \cup \mathscr{A}$.

3.1 Geometric Interpretations based on Convex Cones

We consider the class of convex cones $\mathscr{C}_n = \mathscr{K}_{cc-n} = (\mathbb{R}^n, \sqcap, \sqcup, ^{\circ}, \top\!\!\top, \bot\!\!\bot)$ as the basic class of geometric objects with the order relation interpreted as subset relation $\subseteq$ as defined above. A geometric model of a tbox represents the inclusion axioms in a geometric way, in particular, when $A^{\mathscr{I}} \sqsubseteq B^{\mathscr{I}}$, then A is a subregion of B in the model. However, when we consider geometric interpretations w.r.t. $\mathscr{C}_n$ in more detail, we see that they are quite different from classical interpretations.

For example, assume that the signature σ contains only the concept symbols A, B, C. The geometric interpretation $\mathscr{I}_1$ gives the configuration of convex cones as illustrated on the left of Fig. 3, where $\mathscr{I}_1(A) = a$ and $\mathscr{I}_1(B) = b$. There are areas where according to $\mathscr{I}_1$ an object is neither in a nor in its polar a°, namely all those objects not contained in the gray area. When representing negation with polarity, any point neither contained in a a nor its polar cone a° represents an entity for which class membership of the class A is unknown. This ability is a special feature of this geometric model. The geometric interpretation $\mathscr{I}_2$ illustrated on the right of Fig. 3, again with $\mathscr{I}_2(A) = a, \mathscr{I}_2(B) = b$ and $\mathscr{I}_2(C) = c$ illustrates that distributivity does not hold: $a \sqcup b$ is the convex hull of a and b so $c \sqcap (a \sqcup b) = c$. But on the other hand $c \sqcap b = c \sqcap a = \bot\!\!\bot$ and hence $(c \sqcap b) \sqcup (c \sqcap a) = \bot\!\!\bot \neq c$.

3.2 Geometric Interpretations Based on Axis-Aligned Cones

When the ontology is assumed to be classical, in particular to fulfill distributivity, the class of convex cones has to be restricted. Thus, we consider also a special class of convex structures—namely the class of *axis-aligned cones (al-cones)*.

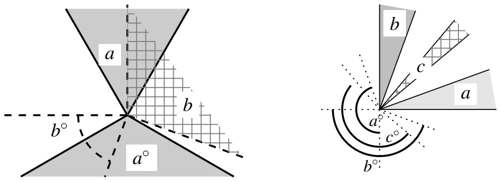

Fig. 3 Examples for geometric models based on convex cones. Left: model showing that excluded middle does not hold. Right: Showing that distributivity does not hold

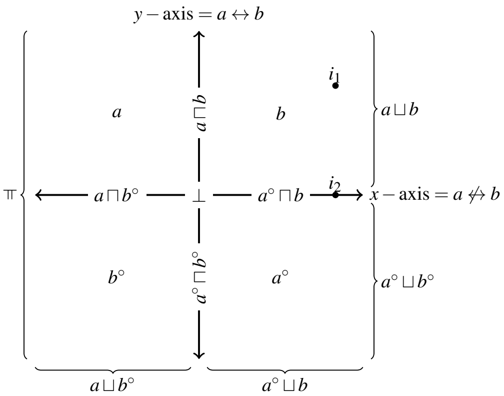

Fig. 4 Example of a geometric interpretation based on axis-aligned cones. For better reading we used $a \not\leftrightarrow b$ for $(a \sqcap b^\circ) \sqcup (a^\circ \sqcap b)$ and $a \leftrightarrow b$ for $(a \sqcap b) \sqcup (a^\circ \sqcap b^\circ)$

Definition 2 An *al-cone* in the n-dimensional space is of the form

$$a = (a_1, ..., a_n) \text{ where each } a_i \in \{\mathbb{R}, \mathbb{R}_+, \mathbb{R}_-, \{0\}\}. \tag{2}$$

For better readability, subsequently $\mathbb{R}, \mathbb{R}_+, \mathbb{R}_-, \{0\}$ are replaced by $u, +, -, 0$.

For the purpose of embedding, every concept of an ontology is assigned to an al-cone as defined in (2) with respect to the tbox axioms of the ontology. Due to the simple form of al-cones geometric operations can be done dimension-wise. So, e.g., the intersection of $(+,-)$ and $(+,+)$ reduces to considering the intersection of the first components $+$ and $+$ (giving $+$) and the intersection of the second components $-$ and $+$, giving 0. In the embedding with al-cones, constants are placed in a region were the corresponding abox axioms are valid. Special cases are the top concept $\top$, represented as $\{u\}^n$ which thus covers the whole space and the bottom concept $\bot$, which is represented as the point of origin $\{0\}^n$.

Figure 4 gives an example of a geometric model for an empty tbox and two concepts A and B. The abox consists of $B(i_1), B(i_2)$ and $\neg A(i_2)$. The element i_1 is in a region where it is neither in A nor in $\neg A$ and thus represents partial information.

The geometric model for a given tbox is constructed based on the set K of all possible fully specified concepts k in the ontology. A concept is fully specified when it contains every atomic concept or its negation. The geometric model has the dimension $d = \left\lceil \frac{|K|}{2} \right\rceil$. A conjunction between fully specified concepts is not possible, so every k is placed on one half-axis. The al-cone for each atomic concept can be determined by constructing the union of all k in which it appears positively. The corresponding negative concept can be found by negating the positive concept. With an empty tbox with n concepts this results in 2^n fully specified concepts and thereby in a geometric model with $d = 2^{n-1}$ dimensions.

With a non-empty tbox the number of possible k decreases, but it is still exponential in the most cases. The construction of the model is similar to the empty case (using the Lindenbaum-Tarski algebra induced by the tbox). Real world ontologies could be of large scale, therefore the exponentiality could be problematic. However, it is possible to find dimension reduction strategies, e.g. considering only a subset of the possible atomic concepts is most times adequate, as many of them are unlikely to appear. This allows for approximating the geometric model in a non-exponential dimension.

For example, the construction of the geometric model with an empty tbox is conducted as follows: The fully specified concepts are $A \dot\sqcap B, A \dot\sqcap \dot\neg B, \neg A \dot\sqcap B$, and $\dot\neg A \dot\sqcap \dot\neg B$. The geometric representation of each of this fully specified concepts is placed on an individual half axis. Thus, the geometric representation $\mathscr{I}(\cdot)$ is for example

$$\mathscr{I}(A \dot\sqcap B) = (0,+) \tag{3}$$

$$\mathscr{I}(A \dot\sqcap \dot\neg B) = (-,0) \tag{4}$$

$$\mathscr{I}(\dot\neg A \dot\sqcap B) = (+,0) \tag{5}$$

$$\mathscr{I}(\dot\neg A \dot\sqcap \dot\neg B) = (0,-). \tag{6}$$

The representations of the other concepts are unions of the representations of the fully specified concepts and thus the resulting model is the one shown in Figure 4.

3.3 The Geometric Model of Pairs of Unions of Convex Cones

Convex cones are useful structures because they are appropriate for logical reasoning with partial information, and al-cones can be used for a multi-label-learning approach as demonstrated in the next section. However, there is a catch with using al-cones: As stated in [17] the dimension of the embedding space increases exponentially in the number of concept symbols used in the ontology. We consider this not as a problem of the approach itself but as the problem of what could be considered in ML speak a wrong bias, namely, assuming that a classical propositional logic such as propositional $\mathscr{ALC}$ is the right ontology language. So, in this paper we additionally consider a second implementation, which is again based on SVMs. But rather than using axis-aligned cones (corresponding to propositional $\mathscr{ALC}$) we use the broader class of pairs of unions of convex cones, for short: puccs. These structures lead to a weaker logic, namely a weak orthologic which can appropriately handle uncertainties in the data (see below).

But why do we not rely directly on convex cones and, instead, use pairs of unions of convex cones? It turns out that directly learning convex cones in the feature or kernel space based on underlying SVMs is difficult. The reason is that for an SVM that learns with (ordinary) cones but without a kernel, the positive instances of a concept have to be perpendicular to the negative ones. It may be the case that this perpendicular configuration already holds in the data—but it is not necessarily the case. However, using an SVM that learns with (ordinary) cones and applies the kernel trick, it is necessary to find a kernel that maps the data into a space in a way yielding this perpendicular configuration. On the one hand, this can be accomplished by a kernel that maps the instances in a way that each instance is perpendicular to each other instance, as then cone separability is trivially given. But this leads to an extremely high-dimensional space and thus results in overfitting and loss of information.

One the other hand it is possible to use a kernel which results in a greater angle between positive and negative than between positives. However, for this kernel it is necessary to know beforehand the ideal separation to choose the kernel in a way which reproduces this ideal separation. Therefore, the separation is needed to be known beforehand which contradicts the SVM-approach. To sum up: Having suitable data, learning with convex cones is possible, however, using arbitrary data, the restriction to polarity-based negation is too strong. Nonetheless, the SVMs should be used as an efficient basis algorithm as they allow for the kernel trick and thus the dimension of the solution could be kept small, and they guarantee a maximum-margin classifier.

As a consequence of the considerations above we can see that a generalization of convex cones is necessary that on the one hand keeps the expressiveness of the cone model but on the other hand is learnable by using SVMs. This is accomplished by softening the definition of the negation operation. In the following, structures are considered which are still based on convex cones but which allow also set-unions of them to be constructed.

In more technical terms, we use structures that are pairs of set-theoretic unions of convex cones, named puccs for short. The reason to use pairs is that for unions of convex cones per se it is difficult to define a negation as some geometric operation. The second component in a pucc is to be thought as a dual of the first component: every natural operation on the first argument is mimicked by its dual operation on the second argument, e.g., intersection on the first argument becomes some form of union on the second argument. So, considering pairs eases to handle a (complex) class and its negation in parallel. A pucc is defined in the following way:

Definition 3 A structure $(\mathscr{S},\mathscr{S}')$ with $\mathscr{S},\mathscr{S}' \subseteq \mathbb{R}^n$ is called a *pair of unions of convex cones*, *pucc* for short, if it fulfills the following constraints:

1. Let $X \in \{\mathscr{S},\mathscr{S}'\}$. Then for all $x \in X$ and all $\lambda \geq 0$ it holds that $\lambda x \in X$.
2. $(\mathscr{S},\mathscr{S}') = Cn_k((\mathscr{S},\mathscr{S}'))$ ($Cn_k((\mathscr{S},\mathscr{S}'))$ as defined in Equation (7) below)
3. $\mathscr{S} \cap \mathscr{S}' = \{0\}$ (where $\cap$ denotes set intersection)
4. Let $X \in \{\mathscr{S},\mathscr{S}'\}$. Then there is no $x \notin X$ such that $\langle x,y\rangle \leq 0$ for all $y \in X'$.

The closure operator $Cn_k(\cdot)$ is motivated by the conic hull operator for convex cone. As the structures contained in the pucc are not necessarily convex or connected, it can't be used in its original form and therefore is adapted.

$$\begin{aligned} &Cn_k((\mathscr{S},\mathscr{S}')) = \\ &\big(\{conH(x_1 \cup x_2) \mid conH(x_1 \cup x_2) \cap \mathscr{S}' = \{0\} \text{ for facets } x_1,x_2 \in \mathscr{S}\} \cup \mathscr{S}, \\ &\{conH(x_1 \cup x_2) \mid conH(x_1 \cup x_2) \cap \mathscr{S} = \{0\} \text{ for facets } x_1,x_2 \in \mathscr{S}'\} \cup \mathscr{S}'\big) \end{aligned} \tag{7}$$

where $conH(\cdot)$ describes the classical conic hull. The basic idea is that the convex hull is added if it is not in conflict with the third constraint.

Using this definition of a pucc, it is possible to directly translate the geometric model based on these puccs into a lattice.

Definition 4 The translation of the geometric model into a lattice:

- Each pucc $(\mathscr{S},\mathscr{S}')$ is an element a of the lattice.
- The orthocomplement $a^{\perp}$ of a lattice element a is denoted using the $\cdot'$ notation again and is defined by permuting the elements of the pucc:

$$(\mathscr{S},\mathscr{S}')' = (\mathscr{S}',\mathscr{S})$$

- The intersection of two lattice elements $a \wedge b$ is defined by closing up the pair with set intersection in the first argument and set-union in the second argument:

$$(\mathscr{A},\mathscr{A}') \sqcap (\mathscr{B},\mathscr{B}') = Cn_k((\mathscr{A} \cap \mathscr{B}, \mathscr{A}' \cup \mathscr{B}'))$$

- The disjunction of two lattice elements $a \vee b$ is defined via intersection and De Morgan:

$$(\mathscr{A},\mathscr{A}') \sqcup (\mathscr{B},\mathscr{B}') = Cn_k((\mathscr{A} \cup \mathscr{B}, \mathscr{A}' \cap \mathscr{B}'))$$

  Note that by definition of Cn_k it follows that it commutes with the orthocomplement: $Cn_k((\mathscr{S},\mathscr{S}'))' = Cn_k((\mathscr{S},\mathscr{S}')')$.
- The bottom element $\mathbb{0}$ of the lattice is given as follows

$$\mathbb{0} = (\{0\}, \mathbb{R}^n)$$

- the top element $\mathbb{1}$ of the lattice is dually defined as

$$\mathbb{1} = (\mathbb{R}^n, \{0\})$$

Note that—due to the definition of *puccs* as pairs— two elements of the lattice are identical iff identity in both arguments holds: $(\mathscr{A},\mathscr{A}') = (\mathscr{B},\mathscr{B}')$ iff $\mathscr{A} = \mathscr{B}$ and $\mathscr{A}' = \mathscr{B}'$.

Having this definition, the question arises, whether the puccs are closed regarding the definition of negation, conjunction and disjunction, meaning whether using these operations always results again in a pucc. The following proposition shows that this is indeed the case.

Proposition 4 *Puccs are closed under conjunction, disjunction, and negation.*

Proof

- All constraints are trivially closed under negation.
- Conjunction and disjunction are related via De Morgan, meaning proving closure w.r.t. one of them is sufficient. In the following it is proven that if $(\mathscr{A},\mathscr{A}')$ and $(\mathscr{B},\mathscr{B}')$ are puccs, then $Cn_k((\mathscr{A} \cap \mathscr{B}, \mathscr{A}' \cup \mathscr{B}'))$ is a pucc, too.
  The considerations are done per constraint of Definition 3:
  1. By $Cn_k()$ only convex hulls are added, meaning the convexity is kept.
  2. The second condition is $(\mathscr{S},\mathscr{S}') = Cn_k(\mathscr{S},\mathscr{S}')$:
     This means $Cn_k((\mathscr{A} \cap \mathscr{B}, \mathscr{A}' \cup \mathscr{B}')) = Cn_k(Cn_k((\mathscr{A} \cap \mathscr{B}, \mathscr{A}' \cup \mathscr{B}')))$. $\mathscr{A} \cap \mathscr{B}$ is not changed by $Cn_k()$. This is the case as $\mathscr{A}$ and $\mathscr{B}$ are parts of puccs. If for the intersecting parts of the puccs $Cn_k()$ would add any elements, this would mean that they haven't been structures beforehand.
     The remaining question is, whether using $Cn_k()$ two times will change the result. Assume there is a facet $x_1 \in Cn_k((\cdot, \mathscr{A}' \cup \mathscr{B}'))$, which is not in $\mathscr{A}' \cup \mathscr{B}'$ and for which for another facet x_2 it is valid that $conH(x_1 \cup x_2) \cap \mathscr{S}' = \{0\}$. But as x_1 is based on the conic hull of some x_3 and x_4 in $\mathscr{A}' \cup \mathscr{B}'$, this means that $conH(x_1 \cup x_2)$ will be included in $conH(x_3 \cup x_2)$ and $conH(x_4 \cup x_2)$. Therefore, using $Cn_k()$ two times will not change the result and the constraint is valid.
  3. It needs to be tested, whether $\mathscr{S} \cap \mathscr{S}' = \{0\}$. Following the argumentation of 2. it follows that $\mathscr{S} = (\mathscr{A} \cap \mathscr{B})$ and $\mathscr{S}' = \{conH(x_1 \cup x_2) \mid conH(x_1 \cup x_2) \cap (\mathscr{A} \cap \mathscr{B}) = \{0\}$ for facets $x_1, x_2 \in \mathscr{A}' \cup \mathscr{B}'\} \cup (\mathscr{A}' \cup \mathscr{B}')$. As every element of $\mathscr{S}'$ does not intersect with $\mathscr{S}$ and $(\mathscr{A},\mathscr{A}')$ and $(\mathscr{B},\mathscr{B}')$ are puccs, the union of the elements does not intersect, too.

4. Let $x \notin \mathscr{S}$ and for all $y \in \mathscr{S}'$ ($\mathscr{S}$ and $\mathscr{S}'$ as defined above) $\langle x, y \rangle \leq 0$. This means that $x \in \mathscr{A} \backslash \mathscr{B}$ (or $\mathscr{B} \backslash \mathscr{A}$) as $(\mathscr{A}, \mathscr{A}')$ and $(\mathscr{B}, \mathscr{B}')$ are puccs. But as $x \in \mathscr{A} \backslash \mathscr{B}$, and $x \notin \mathscr{S}'$, therefore $x \notin \mathscr{B}$, $x \notin \mathscr{B}'$ and therefore $(\mathscr{B}, \mathscr{B}')$ does not fulfill the constraint. A contradiction.

Therefore, the geometric model is closed under these operations. This fact justifies our talk of "translating the geometric model into a lattice" as done in Definition 4. Having this translation, the question arises which logical structure is represented by this geometric model.

The logic induced by pairs of unions of convex cones The resulting logic is an orthologic and hence rule of De Morgan is fulfilled.

Proposition 5
The lattice $\mathscr{L}_{pucc}$ induced by the geometric model of puccs has the following properties:

1. *$\mathscr{L}_{pucc}$ fulfills the rule of De Morgan.*
2. *$\mathscr{L}_{pucc}$ is an ortholattice.*

Proof
1. The rule of De Morgan is fulfilled by definition of the union and only mentioned here for completeness.
2. The ortholattice-rules are considered case by case:
   (a) $a \wedge a^{\perp} = \perp$. Translated into the geometric model, this means the equality $(\mathscr{A}, \mathscr{A}') \cap (\mathscr{A}', \mathscr{A}) = (\{0\}, \mathbb{R}^n)$ should hold. Thus, $(\mathscr{A}, \mathscr{A}') \cap (\mathscr{A}', \mathscr{A}) = Cn_k((\mathscr{A} \cap \mathscr{A}', \mathscr{A}' \cup \mathscr{A}))$ and as discussed above $(\mathscr{A} \cap \mathscr{A}', \mathscr{X})$ with $\mathscr{X} = \{conH(x_1 \cup x_2) \mid conH(x_1 \cup x_2) \cap (\mathscr{A} \cap \mathscr{A}') = \{0\}$ for facets $x_1, x_2 \in \mathscr{A} \cup \mathscr{A}'\} \cup (\mathscr{A} \cup \mathscr{A}')$. By definition 3.3, this is $(\{0\}, \{conH(x_1 \cup x_2) \mid$ facets $x_1, x_2 \in \mathscr{A} \cup \mathscr{A}'\} \cup (\mathscr{A} \cup \mathscr{A}'))$. Thus the remaining question is, whether $\{conH(x_1 \cup x_2) \mid$ facets $x_1, x_2 \in \mathscr{A} \cup \mathscr{A}'\} \cup (\mathscr{A} \cup \mathscr{A}') = \mathbb{R}^n$ and therefore, whether $conH(\mathscr{A} \cup \mathscr{A}') = \mathbb{R}^n$. This is the case by definition 3.4, as that constraint forces the pucc to populate each dimension of the model.
   (b) $a \leq b$ implies $b^{\perp} \leq a^{\perp}$: This can be reformulated to $b = a \vee b$ implies $a^{\perp} = b^{\perp} \vee a^{\perp}$. Translated into the geometric model, this means $(\mathscr{B}, \mathscr{B}') = Cn_k(\mathscr{A} \cup \mathscr{B}, \mathscr{A}' \cap \mathscr{B}')$. It follows that $\mathscr{B}' \subseteq \mathscr{A}'$ and $\mathscr{A} \subseteq \mathscr{B}$ and therefore $(\mathscr{A}', \mathscr{A}) = Cn_k(\mathscr{A}' \cup \mathscr{B}', \mathscr{A} \cap \mathscr{B})$ and therefore $a^{\perp} = a^{\perp} \vee b^{\perp}$.
   (c) $a^{\perp\perp} = a$. In the geometric model, this is $(\mathscr{A}, \mathscr{A}')'' = (\mathscr{A}', \mathscr{A})' = (\mathscr{A}, \mathscr{A}')$.

   Therefore, the resulting lattice is an ortholattice.

We note here that the class of puccs is a rich family of structures: it covers the class of geometric models based on convex cones and hence also the restricted class of al-cones. This motivates the usage of puccs as generalization of convex cones with polarity.

Proposition 6 *There is a closed subclass of the puccs that represents a Boolean algebra.*

Proof Consider puccs $(\mathscr{A}, \mathscr{A}')$ with $\mathscr{A} \cup \mathscr{A}' = \mathbb{R}^n$. These puccs are obviously closed under conjunction, disjunction and negation and they build up a Boolean algebra.

Proposition 7 *A classical cone model is representable by a geometric model based on puccs.*

Proof Assume the geometric model consists only of puccs of the form $(\mathscr{A}, \mathscr{A}^{\circ})$, where $\mathscr{A}$ is a convex cone and $\mathscr{A}^{\circ}$ is the polar of $\mathscr{A}$. This actually is a pucc, as for the constraints in definition 3, (1.) and (2.) are fulfilled by convexity, (3.) is fulfilled as the cone lattice is an ortholattice, (4.) is fulfilled by polarity. Having this puccs, they are closed under disjunction, conjunction and negation (meaning assigning this operations lead to a convex cone and its polar)

- negation: trivial
- conjunction: $\mathscr{A} \cap \mathscr{B}$ is a convex cone, as $\mathscr{A}$ and $\mathscr{B}$ are convex cones. As the conic hull between two arbitrary elements in $\mathscr{A}' \cup \mathscr{B}'$ never intersects with $\mathscr{A} \cap \mathscr{B}$ (because of the ortholattice rules), this means, that it is exactly represented by the conic hull.
- disjunction follows by De Morgan.

4 Cone-Based SVMs

Now that we have defined and discussed two cone models, we can design learning approaches for multi-label classification based on al-cones on the one hand and puccs on the other hand. Both approaches rely on SVMs as they are a widely used learning strategy for binary decision problems because of their good generalization qualities. For an introduction to SVMs see for example [4].

Both presented approaches have the aim to incorporate ontological information in the training process. The fist approach is based on classical propositional logic (Boolean algebra), the second one extends the usage area to a weak orthologic. Both approaches are based on SVMs, the first one is based on al-cones and uses the ontological information to determine the instances used for training each classifier, the second approach is based on learning the geometric model directly in the feature space.

The multi-label learning problem is stated as follows: Let X be a set of training instances with numeric features. Each instance $x_i \in X$ has labels $y_{i,j} \in \{-1,0,1\}$ for each label j, meaning each instance could have assigned the label as positive, negative or unknown. Hence the training set X can be treated as an abox. The (prediction) task is to determine the labeling for a new element. For the first approach, additionally a tbox is given.

4.1 Multi-Label Classification with Al-Cones

The first approach relies on al-cones and thus can be used to embed $\mathscr{ALC}$-ontologies. This approach is based on the assumption that the ontology (the tbox and the abox) of the data is given and so considers mainly the scenario (QC) mentioned in the introduction. The idea for embedding $\mathscr{ALC}$ tboxes with al-cones is to incorporate the training data into the geometric model and learn with the resulting spatial information.

In detail, the approach works as follows: Every element x of the training data is mapped to a subspace of the vector space by creating a code vector $cv(x)$ with $cv(x) = \{+,-,0,u\}^d$ based on its labeling (its abox-information) and based on the geometric model of the given tbox. In this way an element is not represented by an individual point in space but by an al-cone. In every al-cone there could be several individuals. Thus the training elements are embedded into the geometric model, and therefore into the ontology space.

The point is that elements that have the same annotation in one dimension of the geometric model have some similar attributes based on the ontology. The assumption is that these attributes are actually modeled in the data. Therefore, it is assumed that for each dimension a classifier can be trained that separates a (possibly latent) attribute of the feature space.

Therefore, for each dimension $1 \leq k \leq d$ all elements are separated into classes as follows:

$$X_{+,k} = \{x \mid cv(x)_k = +\} \tag{8}$$
$$X_{-,k} = \{x \mid cv(x)_k = -\} \tag{9}$$
$$X_{0,k} = \{x \mid cv(x)_k = 0\} \tag{10}$$

A code-vector with a u at dimension k is ignored. Now, for each dimension a classifier is trained. All classifiers are trained in the feature space, however, the belonging to the positive, zero or negative class depends on the sign of the element in the specific dimension of the geometric model.

We describe in the following in natural language the algorithm for training the classifiers. The pseudo-code of the training is depicted in Algorithm 1, for the classifier in Algorithm 2. The algorithm gets the set of training instances X, the abox (thus the labels of the instances) and the tbox-information as input. Based on the tbox, a geometric model is created as depicted in section 3.2. Then, for each instance $x \in X$, a code-vector cv is created, which decodes the position of this instance in the geometric model. Then, for each dimension k of the geometric model, the set of positive, negative and zero instances is determined based on the respective code-vector and based on these, a SVM is trained for each dimension. For the classification, the new instance x is classified with each classifier in the set of classifiers Cl, thus for each dimension. This leads to its position in the geometric model and enables to deduce the resulting labels based on the given geometric model.

Algorithm 1 al-cone-SVM(X,abox,tbox)

```
GM = create_geometric_model(tbox) //generate the geometric model
Cl = [] //Initialize list of classifiers of size dimension of $GM$
for $x \in X$ do
    cv($x$) = get_code_vector(GM, abox($x$)) //generate the code vector of an element based on its labeling
end for
for $k$ in dimension of GM do
    $X_+ = []; X_- = []; X_0 = []$ //Initialize positive, negative and undetermined training set
    for $x \in X$ do
        if cv($x$)[$k$] $=='+'$ then
            $X_+$.append($x$)
        else if cv($x$)[$k$] $=='-'$ then
            $X_-$.append($x$)
        else if cv($x$)[$k$] $=='0'$ then
            $X_0$.append($x$)
        else
            pass
        end if
    end for
    Cl[$k$] = train_SVM($X_+, X_0, X_-$) //Train a SVM for dimension $k$ in the feature space based on the distribution of elements in this dimension
end for
```

When only one of the three classes in dimension i is used, then training of the classifier is not possible and all elements are assigned to the existing class. When in one dimension there are only two of the three labels used, then a binary classifier is trained and the third class is ignored. A special case appears when all three classes exists. In a geometric model, an element x can be classified as $+$ in one dimension either when it belongs actually to the

Algorithm 2 Classify(x, Cl)

```
for $k$ in dimension of GM do
    cv$(x)[k]$ = Cl$[k]$.classify(x) //Get the classification result for each dimension of the geometric model
end for
classification = GM.get_labels($cv(x)$) //Deduce labels from code-vector
```

positive or when it consists of missing information and would be originally a zero. However, an element which is classified as zero in this dimension actually is zero. Therefore, positive and negative elements get more misclassified than zero ones. Therefore, two classifiers are trained, one separating $+$ from the rest, one separating $-$ from the rest. For both classifiers the misclassification cost of zero is increased to incorporate the above mentioned fact. Even the elements which are classified as $+$ only because of missing information add information to the training process, as the probability is high that they share some attributes of the actual positives.

For classification, the classification result for the test element is determined for each dimension separately. The results of every dimension are concatenated and produce a code-vector (an al-cone) for the test element. This code vector is then placed in the geometric model. An element e is said to belong to a concept C if the code-vector of e is covered by the representation of C.

Using al-cones, the ontology representable is restricted to the Boolean Algebra. To circumvent this restriction and thus to be able to represent weaker ontologies, in the following first SVMs and then puccs are considered.

4.2 Multi-Label Classification with SVMs

To circumvent the restriction to propositional logic (Boolean algebras), we consider an approach that allows for more general structures than al-cones. In order to justify the introduction of a new class of structures, we show that classical SVMs do not exhibit (even implicitly) sufficiently rich logical structure. Therefore, in the next section, an extension of the approach based on puccs is presented, which can serve the purpose of logical reasoning (LR) (see introduction) based on a learnt ontology.

To incorporate a logic with negation, it is necessary to train a classifier that actually has some form of negation, i.e., a classifier that considers not only statements regarding a concept's extension (what must be in its extension) but also statements about what is clearly in the antiextension of the concept. Thus, as a simple multi-label learning approach, a SVM can be trained for each label which separates the positive occurrences of the label from the negated ones.

In a tri-class-SVM labels y_i in the set $\{-1,0,1\}$ are considered, where 0 denotes the undetermined class. The approach in this paper is based on the one presented by Smieja and colleagues in [24]. They proposed a method that forces the zero class to lie in the margin of the classifier. This is done by incorporating an additional loss for the zero-elements, resulting in the cost function

$$SVM_{[-1,1]}(w) = SVM(w) + C \cdot \sum_{i:y_i=0} \Phi(w^T x_i)$$

where $SVM(w)$ is the cost function of the classical SVM and $\Phi(w^T x_i)$ depicts the distance of x_i to the margin-space, thus, $\Phi(w^T x_i) = 0$ when x_i is inside the margin and $\Phi(w^T x_i) =$

$-1+|(w^T x_i)|$ else. Smieja and colleagues [24] showed that this approach is equivalent to adding the neutral instances to both, the positive and the negative instances, i.e. equivalent to setting $X'_+ = X_+ \cup X_0$ and $X'_- = X_- \cup X_0$. Having done this, a classical SVM can be trained, where X_+ is the set of positive labeled instances (X_- the negative labeled and X_0 the undetermined ones).

This approach incorporates the information of the zero label only for improving the positive and negative results, but does not allow for predicting zero labels. However, this is necessary for the approach presented below, therefore, the tri-class-SVM is extended with a simple prediction technique. For the (original) positive and negative class, the margin is calculated, all elements inside the margin are classified as zero, all elements outside as plus resp. minus. It is not necessary that the margin is equal for plus and minus.

Thus, each classifier calculates for a given instance whether the specific label is positively relevant, negatively relevant or undetermined (meaning irrelevant). Now, in order to conduct any interesting form of logical reasoning, it is necessary to logically combine labels. An intuitive approach for dealing with the conjunction and the disjunction of labels is this: An element x is in $A \cap B$ if $x \in A$ and $x \in B$ and $x \in A \cup B$ if $x \in A$ or $x \in B$. The resulting logical structure is that of a Boolean algebra.

However, instead of exploiting the knowledge on simple labels in the training data, it is also possible to learn the label $A \cup B$ directly by training a SVM which separates $A \cup B$ from $\neg A \cap \neg B$. An (simplified) example is shown in Fig. 5. There, the deduced space of $A \cup B$ is the set theoretic union, whereas the learnt $A \cup B$ denotes a larger area. This is the case because the undetermined elements are either actually not determinable or they are missing information. Therefore, it is possible that some undetermined instances can be assigned to $A \cup B$. This is done in this case as the resulting structure is more general and avoids overfitting and thus is more plausible than the set-theoretic union. Thus, the learned region does not necessarily fulfill the rules of Boolean logic. There is no way to change the deduction method to incorporate these union techniques because hyperplanes are generally not closed under conjunction and disjunction.

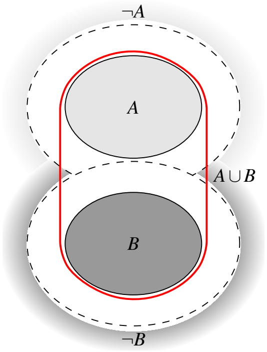

Fig. 5 Labels A and B and $A \cup B$ trained with a classical SVM. The deduced solution of $A \cup B$ denotes the set-theoretic union, whereas the learned solution (denoted in red) is more general and assigns some of the undetermined data as belonging to $A \cup B$

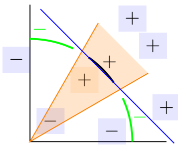

Fig. 6 Example for a cone-SVM. The positive elements are denoted by the black circle part, the negative elements by the green part. The separating plane for the SVM is marked in blue, the learned pucc is in red. The negative part of the pucc is omitted for better readability

This results in the fact, that the deduced logic is Boolean, the learned logic is not (necessarily). Therefore, this approach can't be used as a embedding approach, as logical reasoning on the one hand and analytical reasoning on the other hand does not lead to the same results.

4.3 Multi-Label Classification with SVMs Using Pairs of Unions of Convex Cones

To circumvent the problems of the classical SVM-approach regarding interpretability as embedding, an approach is needed which is able to define the logical operations on the concepts in a way which mimics the operations directly done in the learning approach. Therefore, a geometric structure is needed, which on the one hand can be learned by using SVMs, but which on the other hand has a suitable structure to find a definition of its operations on a logic perspective.

This is done by using puccs as geometric representation, as they are closed under conjunction, disjunction, and negation. Thus, embedding based reasoning is possible. This can be done by keeping the advantages of the SVM, the kernel trick and the maximum margin. The general idea is to use not only one SVM to learn a separating hyperplane, but several SVMs to learn a pucc.

The setting is the following: A multi-label learning problem is given. The instances have numeric features and they are labeled with one or more positive or negative labels. The labeling does not need to be exhaustive.

4.3.1 The Single-Label Case

First, the approach is explained for a binary problem, meaning having only one label and no undetermined data. After that, it is extended to the general case. A set of instances X is given, with $x_i \in X$ and a label $y_i \in \{-1,1\}$ for each x_i.

For the binary case, a feature space is considered, where the instances are centered around the origin of the Euclidean space, normalized and which can be separated in a pucc as defined above. Consider therefore the simple example in Figure 6. A separation can be found by using the classical SVM-approach (depicted in blue), or by learning the optimal pucc (depicted in orange). This pucc is generated by learning several hyperplanes (based on SVMs) and combining them. These seems to be a broad restriction to the usability of cones. However, this is not the case, as the advantage of the SVM-approach the kernel-trick can be used. The idea is not to model the puccs directly in the feature space, but to use the transition into the kernel-space for representing puccs and thus the logical structure implicitly in there. A kernel which directly leads to centralized data is the rbf-kernel.

A rbf-kernel is of the form $K(x,y) = \exp(-\frac{1}{2\sigma^2}||x-y||^2)$[4]. It contains only normalized elements, as $K(x,y) = \langle\phi(x),\phi(y)\rangle$, meaning the kernel is the dot product of the transformations of x and y and $K(x,x) = 1$. Because of the normalization, the instances build a hypersphere-part in the infinite dimensional space. As the rbf-kernel is based on an exponential, the value of the kernel is restricted to $K(x,y) \in (0,1]$. This means that only an (infinite dimensional) hyperoctant of the kernel-space is populated and the puccs are defined only for this region. This simplifies the definition of puccs done in Definition 3.

Proposition 8 *Considering rbf-kernels, the definition of puccs of Definition 3 can be simplified to only consisting of constraints (2.) and (3.).*

Proof Considering it point by point:

1. As only normalized instances are considered, it is possible to add the ray containing a normalized element x without changing any classification result as unseen instances are also mapped to a normalization. Therefore it could be assumed that instead of points, rays are considered.

2.,3. Stay relevant as both could also appear in this restricted version.

4. By definition there are no x, y with $\langle x,y \rangle \leq 0$, therefore, this constraint is fulfilled trivially.

Having this simplification, it is possible to show that a rbf-kernel-space which is linear separable is also separable by a pucc and furthermore, this pucc has some nice properties.

Proposition 9 *If a set of instances X is separable by a hyperplane (using a rbf-kernel), the rbf-kernel-space is separable by using a pucc. Specifically, at least one structure in the pucc is a convex cone.*

Proof The first part is trivially true, as each element could be represented by a ray. This would result in a pucc.

Cutting a hypersphere with a hyperplane results in a convex cutting edge and therefore, at least one structure of the pucc is convex.

The presented approach is not restricted to rbf-kernels, as every kernel which results in centralized data can be used. However, in the following the rbf-kernel is used, as it is a widely used kernel in many areas.

The idea of the learning approach is to learn the facets of the puccs as hyperplanes of a SVM. Thus, several SVMs need to be trained to learn one pucc.

As a pucc is centered around the point of origin, it is necessary to adapt the general notion of a SVM-classifier. A classical SVM allows for a bias term b, which need to be $b = 0$ for a centered hyperplane. The basic formulation of the SVM for the linear separable case and without bias is

$$\begin{aligned} &\min ||w||, \\ &\text{so that } y_i(w^T \cdot x_i) - 1 \geq 0 \; \forall i. \end{aligned} \tag{11}$$

Using the kernel trick for a SVM without bias leads to

$$\begin{aligned} \max \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathrm{x}_i, \mathrm{x}_j) \\ \text{so that } 0 \leq \alpha_i \leq C \; \forall i, \end{aligned}$$

where α_i are the Lagrange-multipliers and $\alpha_i > 0$ denotes that x_i is a support vector, meaning it has an influence on the creation of the hyperplane [4]. The kernel-function is denoted by $K(x_i, x_j)$. The parameter C is the error term which allows to separate non-separable data.

It is not known beforehand which instance is a support-vector for which hyperplane. This means, a technique is needed to find these influences.

Using classical clustering ideas is challenging because of the usage of the kernel-trick and therefore the loss of spatial information. Therefore, a technique is used which is based on the idea of incorporating spatial information by directly training hyperplanes and interpreting them.

The first idea is to train one hyperplane and consider all missclassified elements to train the next one. However, this approach has several drawbacks: First, it is possible that all positive instances are classified correctly, but the negative ones are not. Then it is not possible to train another hyperplane, however, considering the whole data, it could be possible to do so. Second, it is not necessarily the case that the classification of new elements is unambiguous, as correct classified elements in the first place are not considered anymore.

Therefore, the approach is enhanced to a hierarchical method. First, a centered SVM is trained based on the whole dataset. This leads to one facet of the pucc. Then all positive classified elements are considered. If the actual classification of these elements is positive, then nothing happens. If some elements are misclassified, then another classifier is trained, which separates actual positives and actual negatives. This is done stepwise (also for the negative classified part) until the classification error is under a threshold and leads to a tree of classifiers. For classification of a new element, this tree can be traversed until a leave is reached. This leave then denotes the class of the new element.

The pseudo-code of the learning algorithm can be seen in Algorithm 3 and the pseudo-code of the classifier in Algorithm 4. Based on the features X and their classification Y, a SVM is used to determine a decision boundary. As separability needs not to be given, for both sides of the decision boundary additional classifiers needs to be trained. Therefore, first X is separated into the set *left_el* which denotes the instances of X which are negatively classified by the classifier *plane* and in *right_el* for the positive ones. Then it is determined how many elements in *left_el* (resp. *right_el*) are misclassified. If this is above a given *threshold*, then a new SVM is trained in the same way, using *left_el* (resp. *right_el*) and the labels of the elements *left_el* (resp. *right_el*) as input. The new SVM is set as the *left_child* (*right_child*) of the SVM trained before. If the number of misclassifications is smaller than the threshold, no further training is necessary and *left_child* (*right_child*) is set to -1, i.e., they remain empty. The algorithm terminates when all paths terminate in empty children.

A new element can be classified by traversing the tree of classifiers. For a new element x, the root-classifier, thus, the first classifier trained, is used to predict a classification. Having a classification as negative, the negative part of the tree is traversed, meaning the left child of the classifier is considered. There are two possibilities: Either *left_child* does not exists (thus a -1 is returned), then the *classification* is the classification of the instance, or *left_child* is itself a classifier, then the classify-function is called for the *left_child*. For a positive classification and the right child it is analogue. This is done until a leaf is reached and a classification is returned.

This hierarchical method is chosen, as it is on the one hand possible to enhance the relevance of the given instances for training when going deeper in the tree and on the other hand it is easily possible to find a stopping criterion by defining a minimum number of instances left to avoid overfitting.

The incorporation of undetermined instances The presented approach is based on the assumption of exhaustive labeled data. However, this is not always the case. As discussed above, in the following the focus lies undetermined data. To handle this, an extension of the above approach is proposed.

Algorithm 3 Cone-SVM(X,Y)

```
plane = SVM$(X,Y)$ //Calculate a SVM (without bias and with a rbf-kernel)
for $x \in X$ do
    if plane.classify$(x) == -1$ then
        left_el = $x$ //find the elements classified as positive/negative
    else
        right_el = $x$
    end if
end for
if incorrect_classified_el_in_left_el > threshold then
    left_child = Cone-SVM(left_el, class_left_el) //Set the child as a new classifier based on the elements
    classified as positive, when the number of incorrect elements is high enough.
else
    left_child = -1
end if
if incorrect_classified_el_in_right_el > threshold then
    right_child = Cone-SVM(right_el, class_right_el) //Set the child as a new classifier based on the elements
    classified as negative, when the number of incorrect elements is high enough.
else
    right_child = 1
end if
```

Algorithm 4 Classify(x, Cone_SVM)

```
classification = Cone_SVM.predict($x$) //Traverse the tree by starting at the root node
if classification == -1 then
    if Cone_SVM.left_child == -1 then
        return -1
    else
        return Cone_SVM.left_child.classify(x)
    end if
else
    if Cone_SVM.right_child == 1 then
        return 1
    else
        return Cone_SVM.right_child.classify(x)
    end if
end if
```

These instances are assumed to lie in a region in between the positive and negative instances. This restriction is necessary, as the learned structure should result in a pucc and therefore needs to fulfill the closure properties.

The first step is to adapt the approach presented above by switching the binary SVMs to tri-class-SVMs. This allows to find the regions classified as undetermined. The training procedure is done as above, the only difference appears when a tri-class-SVM correctly classifies an element as undetermined. These elements are not considered for the next training step (for the training of the children), as they are correctly classified and they do not incorporate any additional information and would worsen the result.

The classification of new elements is done in a similar way than above. The only special case is when one classifier results in zero. Then, it is possible that the instance actually is classified as zero or not. Therefore, both children are considered. When both of them result in the same classification, e.g. positive, then the instance is in between two positive regions and thus classified as positive to satisfy the closure property of a pucc. When both classifiers result in different classifications, then it is classified as zero, as it lies in between the positive and negative.

However, this is not sufficient to handle the undetermined instances. It is not necessarily the case that the undetermined data lies inside the margin between positive and negative, as the structure of puccs allows for a lot more shapes of the undetermined regions. Therefore, the case could appear that at some point for example only positive and undetermined elements are left. Then a binary classifier separating the two is learned.

In the following, it is argumented that this approach actually leads to a pucc. Introducing the binary positive/zero- (resp- negative/zero)-classifiers could lead to a problem regarding the closure with Cn_k. This could appear when a undetermined region is in between two positive regions and thus is not undetermined but only consists of incomplete knowledge. Therefore, it is necessary to determine whether all the learned binary classifiers are actually necessary. If one (known) instance is classified as zero and both children are classified as zero with a binary +/0-classifier, then both +/0-classifiers can be deleted, as the resulting structure would not be a pucc, as Cn_k would not be valid.

Having this control mechanism, it can be concluded that each construct actually is a pucc, thus the training approach is suitable for learning puccs. Additionally, each classical separable problem (by using a rbf-kernel) is separable by a cone-SVM. The problem here is that it is possible that it is necessary to train a classifier for each element of the training set, thus, the number of SVMs could be large. However, this is only a special case. An advantage of the cone-SVM is that it is not restricted to SVM-separable elements, it is also possible to separate structures which are a lot more complicated.

Proposition 10

1. *Each learned construct is a pucc.*
2. *Each problem classical separable (by using a rbf-kernel) is separable by a cone-SVM.*

Proof 1. the constraints of a pucc out of Definition 3 are considered case by case. It is sufficient to show the second and third condition, as the other ones are trivially fulfilled (see Proposition 8)

   ii. Assume that the constraint $(\mathscr{S},\mathscr{S}') = Cn_k((\mathscr{S},\mathscr{S}')$ is not fulfilled. Assume, it is not fulfilled for $\mathscr{S}$. This means, there are facets $x_a, x_2 \in \mathscr{S}$ not intersecting with any element of $\mathscr{S}'$, but incorporates at least one element not in $\mathscr{S}$. In the learning setting this means that two hyperplanes are learned to separate the positive, having no negative element in between. Therefore, the hyperplanes would not have been learned and thus, this case can not happen. The only exception is the case where both hyperplanes belong to binary $+/0$-classifiers. However, this case has been directly considered in the learning approach.

   iii. $\mathscr{S} \cap \mathscr{S}' = \{0\}$ is the case by definition of the learning approach. Each element is unambiguously classified. Elements lying directly on a hyperplane could be in both classes, but this problem can be solved by defining those elements to be in the positive.

2. Each problem classical separable is separable by a cone-SVM, when the number of trained SVMs is not restricted. This is the case as it is possible to train a SVM for each incorrect classified element until all of them are correct classified. This would lead to overfitting, thus in reality, it is necessary to restrict the number of trained SVMs. In this case, it is possible that cone-separability is not possible.

This approach is in the following extended to its main purpose of multi-label learning.

4.3.2 The Multi-Label Learning Approach

The aim of our multi-label learning approach is to learn and represent the inherent logical structure of the data based on the combination of several puccs. The interrelations of the labels and thus the underlying ontology is assumed to be unknown.

The basic idea for this puccs based approach is to decompose the problem into several ternary pucc-SVMs, a pucc-SVM for each label. Each of these have the same hyperparameters in order to ensure the possibility of combining the resulting concepts.

Having learned the geometric representation of all labels, it is possible to do logical reasoning or to determine relations between labels. Therefore, it is possible to deduce puccs which result out of label combinations. For example, if one is interested in the geometric representation of the label $A \cap B$, then it is necessary to deduce a pucc of the form $(\mathscr{C}, \mathscr{C}') = Cn_k((\mathscr{A} \cap \mathscr{B}, \mathscr{A}' \cup \mathscr{B}'))$.

To reach this, it is necessary to combine the trees of classifiers of A and B to have one tree of classifiers for C. One computationally simple method is to combine the trees by first using the first classifier of the tree of $(\mathscr{A}, \mathscr{A}')$, on the next level the first of $(\mathscr{B}, \mathscr{B}')$ and so on. Each leaf where the classification of the $\mathscr{A}$-tree and the $\mathscr{B}$-tree is positive, is classified as positive, in all other cases, the leaf is classified as negative. The undetermined cases are implicitly defined as explained for the single-label case above. At the end, the method for checking whether the binary classifiers fulfill the pucc-constraint has to be done. Because of its computational simplicity, this approach only approximately depicts the hull $Cn_k((\mathscr{A} \cap \mathscr{B}, \mathscr{A}' \cup \mathscr{B}'))$. With the same argumentation as in Proposition 10 it can be shown that the resulting tree actually is a pucc, meaning that every undetermined element added to Cn_k is also added to the region determined by the new tree. We note that in the algorithm sketched above the conic hull of two facets is not calculated but only the regions based on the hyperplanes are combined. This might lead to the undesirable outcome that the resulting region is greater than the original Cn_k-closed one. This can be solved by complicating the combination approach. However, it is assumed that the minor errors in the accuracy of the union will not be worth the computational complexity of such an approach.

As a bottom line, the multi-labeling algorithm shows a main benefit of using puccs: the hull Cn_k of puccs can be approximately modeled via simple combinations of well-known and ML-proven data structures such as decision trees. The algorithm, on the one hand, learns the geometric representation of each label(-combination) and, on the other hand, allows us to deduce each geometric representation based on some learned labels. As deducing and learning both result in a pucc, it is—in contrast to the classical SVM-approach—possible to interpret the achieved embedding as one that allows for logical reasoning. As stated above, there are cases where both are only approximately equivalent. However, even for these cases, the closure property of puccs is fulfilled and therefore the equivalence can be approximated. This fact shows that, in contrast to classical SVMs, our approach ensures that, theoretically, deduced compound concepts and learned compound concepts are the same.

5 Experiments

5.1 Evaluation of the Multi-Label Classification Algorithm Based on Al-Cones

The proposed approach is evaluated in two areas that involve multi-label classification. In a first setting, we consider the classical multi-label learning scenario. In a second second setting, we consider Zero-Shot Learning which is a closely related and challenging task.

5.1.1 Multi-Label Learning

Implementation As approach for doing the binary classification, a support vector machine with a polynomial kernel is used, because it is an established method for handling bioinformatic datasets like the one used. For the test of the method the assumption is used that not having a positive label means that it could be contained or not.

Data The method can be used for any ontology expressed in Boolean $\mathscr{ALC}$. Here the Gene Ontology (GO) [1] is used. It does not contain negation or union and is hence a directed acyclic graph. The relations of GO have not been considered. The data set for the experiments is that of Saccharomyces cerevisiae [26]. First the concepts of the training elements are extended in the way that all ancestor concepts of the given concepts are contained. Then every concept without enough elements representing it was deleted to facilitate the training process. The number of concepts was reduced to eight and for every element the most specific concepts were determined. With these concept labels the training and testing was conducted. This results in a size of the training set of 1000 instances and 200 instances in the test set. The smallest concept has 58 instances, the biggest 314, with an average of 160 instances.

For comparison purposes we implemented the approach of Wan and Xu [27]. The approach presented in [27] does not use ontology information. It is based on a variant of the 1-vs-1-classifier. Any two concepts are compared to one another in a ternary way. A separation of elements of concept A, of concept B, and of concept $A \sqcap B$ is learned for all concepts A, B. Via a voting-scheme and a threshold the concepts of an element are obtained. We have chosen this approach as baseline as it is very similar to the method proposed in this paper. The main difference is that in the baseline ontology information is not used. The experiments thus allow us to study the utility of exploiting ontology information. For better comparability—instead of the Tri-class SVM as used in the approach by Wan and Xu—we use in our implementation the SVM-architecture presented above. The execution time is heavily dependent on the number of dimensions, in this scenario, it takes a few minutes.

Results and Discussion Using a six-fold cross validation, classification of the test set yields the data shown in Table 2. Accuracy, recall and precision are determined by reducing both predicted and test labels to the most basic concepts and evaluate them with standard multiclass metrics. As can be seen, performance measures for the presented approach are similar to the baseline. An advantage of the presented approach is that it can only yield ontologically correct results, while the baseline may lead to classifications that do not agree with the ontology. One example would be classifying an instance with a child concept without classifying it with the parent concept. In every dimension, more training elements and thus more pieces of information can be used as compared to the usual 1-vs-1 case as not only elements with the same concept, but also with some similar attributes are used in the same class for training.

The similarity in the results of both approaches is caused by the simple structure of GO, which incorporates no negation or disjunction—only concept subsumption occurs. Without negated elements or negated concept inclusions there is no knowledge about concept exclusions and therefore the space of possible concepts per element cannot be restricted. Another reason could be the choice of the binary classifier and a different classifier could perhaps improve quality of generalization. The presented approach has improvement potential w.r.t. the error tolerance. Concepts at the bottom of the tree have only a small al-cone where the elements could be placed. This means, that even a small misclassification in one dimension

Table 2 Results for the presented method and the approach of Wan and Xu [27]

	Accuracy	Precision	Recall
This approach	0.185 ± 0.03	0.190 ± 0.02	0.164 ± 0.03
Wan, Xu [27]	0.197 ± 0.03	0.199 ± 0.03	0.278 ± 0.08

could prevent the correct classification. One possible solution is to incorporate knowledge of the certainty of the classification for each dimension. For a test element an uncertain result in a dimension could be changed to 0 to reduce its influence.

In a second experiment our method was tested with an empty tbox. This resulted in an accuracy near to zero and demonstrates the usefulness of the ontology information for training. Without this information the knowledge about dependencies of elements cannot be used and elements which have similar attributes can not be separated from elements without similar attributes. With an empty tbox impossible separations are tried to be learned as well. Therefore classifying a test element results in a code-vector not containing any information and thus no assigned concept. This shows that the approach can actually use the knowledge represented in the ontology.

5.1.2 Zero-Shot Learning

Zero-Shot Learning (ZSL) is a multi-class learning task in which each instance has to be assigned exactly one label. The distinct feature of ZSL is that instances to be classified by a label not seen during training [29]. To be able to label previously unseen classes, some auxiliary information is needed. This can be done in form of per-class-attribute information, meaning for each class a set of attributes is given and specified whether one is positive or negative. For example, a ZSL task may require a previously unseen class 'zebra' to be identified, provided the information that zebras exhibit the featured 'striped'. A ZSL learner would need to identify an instance of the zebra class, robustly discriminating it from horses (which share many attributes of zebras but being striped) and tigers (which are striped but unlike zebras otherwise). Formally, $\{(x_i, y_i) | i = 1 \dots n\}$ are the training instances x_i denoted with labels $y_i \in \mathscr{Y}^{tr}$ that belong to the training classes. Test instances $\{(x_j, y_j) | j = n \dots n + m\}$ with labels $y_j \in \mathscr{Y}^{ts}$ constitute test classes, where $\mathscr{Y}^{tr}$ and $\mathscr{Y}^{ts}$ are disjoint.

One prominent approach for solving the ZSL problem is EXEM, presented in [6], which is based on the insight that instances of unseen labels will cluster around the semantic embedding of that class. Therefore, based on the seen classes, a predictive function is learned to predict the semantic embeddings (called *exemplars*) of the unseen classes. The approach works in detail as follows: A transformation function ϕ is learned, which is able to map for each class c the attribute vector a_c to the visual exemplar in the feature space v_c, therefore, $\phi(a_c) \approx v_c$. Vector v_c of a known class is the mean of all instances of that class obtained by performing a class-unspecific PCA projection computed of training data from seen classes. Then, d Support Vector Regressors (SVR) with rbf-Kernel are learned, where d is the dimension of the PCA. For more details, see [6]. Using these regressors, the visual exemplars of unseen classes can be predicted based on its attribute vector. For a test instance, 1-nearest neighbor (1NN) is used to select the nearest visual exemplar based on (standardized) Euclidean distance.

Whereas [6] is solely based on Euclidean distance, it is possible to extend the approach to use the given attribute knowledge not only for distance-based but also for geometric reasoning. The attribute representation of each unseen class is interpreted as tbox-information and each label as atomic concept. Therefore, a geometric model of dimension $|\mathscr{Y}^{te}|/2$ can be created that associates an atomic concept (a label) with each half-axis. This is done by predicting not only the visual exemplars of the unseen classes but also visual exemplars of unseen superclasses, e.g., *A or B*. Thus, an attribute vector can be determined which could be A or B with the same probability. This vector would have positive (resp. negative) attributes, when both A and B have this attribute (positive, resp. negative). When A and B differ in an attribute, this would be set to zero. Then, based on both types of visual exemplars and the geometric model, the implementation of the geometric model as stated above can be used for training. Introducing superclasses for training leads on the one hand to the advantage of increasing the number of training instances and gives more information on the geometry of the trained classes. On the other hand, it enables information about the certainty of the classifier's decisions to be obtained (we discuss this features in the result section further below).

Dataset For evaluation, we consider a standard dataset for ZSL, the Animals with Attributes (AWA2) dataset proposed in [29]. AWA2 comprises 40 classes for training and ten for testing, each class having 85 either positive or negative attributes. In total $37,322$ instances of animals are contained. The classes have 746 images on average, the least populated class 100 and the most populated class 1645 images [29]. As feature space the embedding of the images as 2048-dim top-layer pooling unit of the ResNet are used [29]. As train/test split, the standard split is used, as is done in the baseline approach [6].

Implementation For the baseline approach, we use the implementation[1] provided alongside [6] for their approach EXEM. For adaption to AWA2 we changed the kernel used to a polynomial kernel because of better classification properties on this dataset. The approach is evaluated by using a class-based five-fold cross-validation (thus, all instances of one class are either completely in the training or completely in the validation set to ensure ZSL also for the cross validation) to determine the best hyperparameters and tested on the proposed test split. The hyperparameters of the regressor have been taken based on the minimization of the Euclidean distance, as done in [6]. Based on this, the hyperparameters of the SVM have been tuned as a trade-off between overall recall and recall for returning one label, as the result should be as specific as possible by being able to return labels to as many instances as possible. Execution time of the training is very short (few seconds on standard laptop), due to consideration of visual exemplars as representants of the class.

Result and Discussion Figure 7 presents precision and recall obtained for our approach in comparison to the baseline and is accompanied by Table 3 which presents the numerical values and their variance. Individual marks in the plot show precision and recall obtained for a classifier output containing 1, 2, 3, ... possible class labels.

For example, the best class label obtained in the baseline approach achieves about 0.75 precision and 0.75 recall, whereas a single class label obtained by our approach achieves 0.9 precision and 0.15 recall. As can be seen, the baseline outperforms our approach in terms of recall as it can potentially generate any class label, even those not agreeing with the auxiliary information. By contrast, our approach is restricted to labels agreeable with auxiliary information given, which in case of sparse and noise training data may inhibit perfect recall. With respect to precision our approach outperforms the baseline, both in terms of the maximal precision achievable and in terms of the precision relative to the number of class

[1] https://github.com/pujols/Zero-shot-learning-journal

Table 3 Precision and recall with respect to the amount of labels returned for classification with negated attribute vectors

# returned labels	Precision		Recall	
	our approach	EXEM	our approach	EXEM
1	0.895 ± 0.059	0.710 ± 0.048	0.094 ± 0.041	0.710 ± 0.048
2	0.465 ± 0.043	0.443 ± 0.014	0.549 ± 0.092	0.885 ± 0.028
3	0.399 ± 0.039	0.317 ± 0.005	0.816 ± 0.112	0.950 ± 0.016
4	0.378 ± 0.045	0.243 ± 0.003	0.926 ± 0.064	0.973 ± 0.012
5	0.373 ± 0.047	0.020 ± 0.001	0.955 ± 0.031	0.987 ± 0.007
6	0.372 ± 0.048	0.166 ± 0	0.959 ± 0.015	0.995 ± 0.002
7	0.372 ± 0.048	0.1425 ± 0	0.960 ± 0.013	0.998 ± 0
8	0.372 ± 0.048	0.125 ± 0	0.960 ± 0.013	0.999 ± 0
9	0.372 ± 0.048	0.111 ± 0	0.960 ± 0.013	0.999 ± 0
10	0.372 ± 0.048	0.100 ± 0	0.960 ± 0.013	1 ± 0

labels generated. In other words, the proposed method is well-suited to reject classifier decisions that are not agreeable with background information. Such feature can be important in applications where wrong decisions must be avoided. This is accomplished by an approach capable of respecting the inherent structure of the feature space.

The results with respect to the proposed approach can be explained based on the fact that it splits the feature space into subspaces representing the areas of the geometric model. The region close to the visual exemplar is represented by a half-axis of the geometric model, further away from the visual exemplars (near the partial information visual exemplar), the union of several half-axes can be found. Thus, is is possible to determine the certainty of a classification result simply by counting the number of possible labels (for illustration, compare the horizontal lines in Figure 7 with the marks). Considering the baseline, it is possible to increase the recall rate simply by changing from 1-nearest neighbor to also include the 2-nearest, 3-nearest, and so one. Such an approach does however provide no information about which distance is still agreeable with the auxiliary information. By contrast, our approach has the ability to state which labels are definitely incorrect for a given instance.

To demonstrate the ability of rejecting label that are not agreeable, we conduct an additional experiment involving incorrect attributes. We negate the attribute representation of the test set (meaning $a'_c = -a_c$). Based on this, kNN is executed, resp. the SVMs of our geometric model are trained. The change has the effect that the stated visual prototype is not associated with the original label center. As expected, both approaches show a recall and precision near zero. However, with our approach one is able to recognize the problem as it returns for no element less than four labels, as can be seen in Table 4. As explained above, the amount of class labels is indicative for the uncertainty of our classifier. Using the capabilities of modeling negation, it can be stated which labels are definitely not correct for an instance. Therefore, the ability of the geometric model to model negation and disjunction is used.

5.2 Evaluation of the Multi-Label Classification Algorithm Based on Pairs of Unions of Convex Cones

In the following, a single-label data set is used and the classification result is compared with that of a classical SVM. This is done, as the first step is to show the functionality of the approach independent of any reasoning capabilities. As the multi-label approach is based on a combination of many single-label classifiers, it is sufficient to show the general

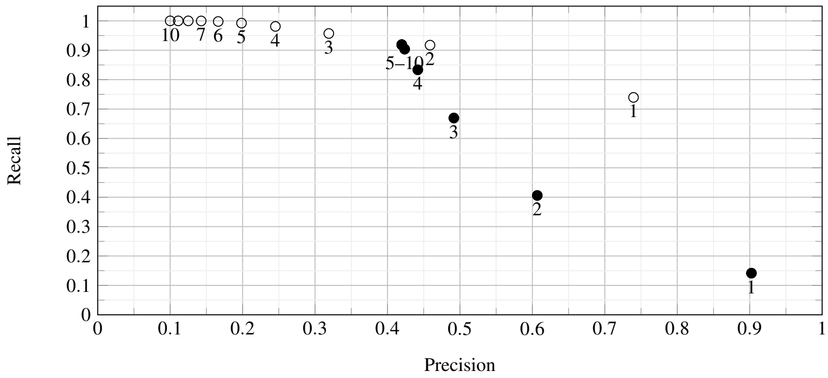

Fig. 7 Accuracy and Precision for our approach (black) and the other approach (white), based on the proposed test set. Numbers representing the number of labels returned.

Table 4 Uncertainty in classification of maliciously crafted AWA2 variant: percentage of instances labeled with exactly *i* labels

number of returned labels	instances with *i* labels (in %)
0	0
1	0
2	0
3	0
4	0
5	0.032
6	0.236
7	0.441
8	0.258
9	0.032
10	0.001

functionality for one single-label classifier. To ensure comparability with the classical SVM-approach, a binary learning problem (not a ternary) has been used and both approaches have been trained with a rbf-kernel.

Implementation Both approaches have been implemented in python[2]. As there isn't a SVM implementation without a bias-term available, the quadratic programming solver cvxopt[3] has been used to solve the dual of the SVM directly. For comparison, the SVM-implementation of python scikitlearn[4] has been used.

Data As a data set, the rna-dataset of Uzilov et al [25] is used. It has eight features and a binary outcome, whether an element is a specific rna or not. As the implementation at this stage is based on quadratic programming, only a subset of the size of 2500 elements is used for training, because of time constraints. However, this is not a general restriction of the

[2] `https://www.python.org/`

[3] `http://cvxopt.org/`

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html` based on [5]

Table 5 Results for the pucc-SVM and the standard SVM

	Accuracy	Precision	Recall	F1-score
Pucc-SVM	0.877 ± 0.01	0.823 ± 0.03	0.812 ± 0.03	0.818 ± 0.02
Standard SVM	0.885 ± 0.02	0.849 ± 0.04	0.806 ± 0.02	0.827 ± 0.03

presented approach, as it is possible to adapt the known optimization strategies for SVMs for SVMs without bias.

Results and Discussion For evaluating the quality of the approach, the standard F1-score is used and a six-fold cross-validation has been performed. In addition, accuracy, precision and recall are reported. Performance of both approaches are subject to the choice of the kernel parameter σ and the misclassification cost C, which has been tuned independently both approaches. All results are given in Table 5. For the pucc-SVM, the parameters are $\sigma = 5$ and $C = 10$, for the classical SVM, $\sigma = 0.0001$ and $C = 100$.

As can be seen from the data, both approaches lead to a comparable result. Something that is noticeable about the parameters of both approaches is that they differ substantially. For the parameter C, the result of the pucc-SVM with $C = 100$ is with a f1-score of 0.799 ± 0.03 only slightly worse than the result for $C = 10$. However, for the parameter σ, a small value of σ used with the pucc-SVM (a large value of σ for the classical SVM, respectively) will worsen the results significantly. To understand this, it is necessary to recap that the rbf-kernel models the distance between elements. Using a high value for σ increases these distances. The pucc-SVM has its strength in fitting several hyperplanes. This is especially useful if the distribution of elements is more sparse.

The result shows that the approach of learning puccs is suited for learning as it leads to results comparable to the classical SVM with a rbf-kernel. As the multi-label case is based on a separation in single-label problems, this means that also the multi-label case should result in at least comparable results.

However, the main advantage lies not in the general classification accuracy but in the ability to model the underlying logic. Fine-tuning of learning performance and investigation of multi-label problems are aim of further research.

6 Related Work

Only recently investigations have started that—in logical terms—shed light on the expressivity of concept hierachies that result from embedding algorithms. These investigate useful contributions for finding the right balance in the overall expressivity versus feasibility dilemma. Earlier approaches to KGE were motivated by efficient learning algorithms, hence resolving the dilemma strictly in favour of feasibility. In those approaches including the well-known TransE [3], heads and tails of KGE triples are represented as real-valued vectors and relations as vectors, matrices or tensors, i.e., geometric operations. In many occasions, the geometric operations lead to relations that are functional or are constrained by other means. Hence, those operations provide not much expressivity from a logical point of view. In the following we discuss only those KGE approaches that explicitly mention the kind of geometric objects used for embedding concepts and relations and the logic that characterizes them. A logic characterization of the approaches discussed is summarized in Tab. 6. For other relevant literature on KGE the reader is referred to the overview of [28].

Geometrical Structure	Logic	Concept lattice	Negation	Approach/Reference
Convex sets	Quasi-chained Datalog$^{\pm}$	distributive	atomic	[14]
Hyperspheres	$\mathscr{E}\mathscr{L}^{++}$	distributive	atomic	[16]
Axis-aligned Cones	$\mathscr{A}\mathscr{L}\mathscr{C}$	distributive	full Boolean	[20]
Closed Subspaces in Hilbert Space	Minimal Quantum Logic	orthomodular	orthonegation	[11]

Table 6 Comparison of approaches for embedding w.r.t. the geometric objects and the induced logics

The authors of [14] identify a fragment of existential Datalog (fulfilling the quasi-chainedness property) as an appropriate logic for arbitrary convex regions in euclidean spaces. The authors of [16] find a correspondence for hyperspheres and the lightweight description logic $\mathscr{E}\mathscr{L}^{++}$. [20] identifies axis-aligned cones as an appropriate geometrical class for the semi-descriptive logic $\mathscr{A}\mathscr{L}\mathscr{C}$. While [14, 16] do not allow for full negation of concepts to be represented, [20] defines negation for the model of axis-aligned cones that uses polarity, which possibly gives rise to some interesting logic structure. On the other hand, in [20] binary relations are allowed to be arbitrary pairs of vectors, whereas [14] models also relations (of any arity) by convex regions.

In all three approaches the expressible concept hierarchy fulfills distributivity of conjunction over disjunction. The approach of [11] considers minimal quantum logic which does not fulfill distributivity but (only) a weakening: orthomodularity (see preliminaries for a technical definition). But, as argued for in [7], the ability to express non-distributive concept hierarchies means a benefit, as it enables modelling uncertainty w.r.t. the extensions of concepts. In fact, in many ML learning tasks, concepts do not have a crisp boundary, but rather have areas of uncertainty and as such provide partial information only.

The need to deal in a learning scenario with many interrelated concepts instead of a single concept marks also the main motivation for the now-rich research on multi-label learning. In this learning scenario, during training, instances are assigned to one or more labels (alias concepts). In contrast to the multi-classification case, the correlations of the labels may be different form being (just) pairwise disjoint and jointly exhaustive. The most intuitive idea for multi-label learning is to reduce it (or how it is called sometimes: apply problem-transformation on it [21]) to the problem of learning each label independently (by a SVM, say) and then try to regain the dependencies/correlations by some further considerations. There are different ways known in the literature for the method of reduction (one versus one, one versus all, say [12]), and also different ways to regain the dependencies of the concepts such as that of *binary relevance* [31]. But there are also approaches such as that of [15] that try to account for the possible dependencies of the labels directly from the beginning—unfortunately those approaches prove to work only for very small datasets according to [21].

Though all those different approaches to multi-label learning account for the dependencies of concepts in a concept hierarchy, they do not handle those dependencies in a logic disguise—in contrast to the KGE approaches discussed above and also in contrast to the approach developed in this paper.

7 Conclusion

In view of the two cone-based embedding approaches presented in this paper and in view of their promising evaluations we would like to conclude here with the statement of our initial working hypothesis: *learning algorithms that aim at providing means to integrate logical constraints formulated in an ontology (QC) or that aim at producing results that are suited for logical reasoning (LR) have to account for geometric logical structures in the training phase (and not in a post-processing phase, say).* In particular, in justifying the working hypothesis, we showed that the first approach reaches this goal by creating an external geometric model based on the ontological knowledge and learns under its constraints. The fact that this approach is restricted to propositional logic and that it leads to the problem of an exponential blow-up of the geometric model motivated introducing a wider class of geometric-logical objects: pairs of unions of convex cones, puccs. We showed how to extend an SVM-based training approach to account for puccs such that the learning results can be considered as concepts in an orthological terminology—thereby providing means to apply logical reasoning.

Future works concerns strengthening the justification of the working hypothesis along the following lines of research: showcase in realistic scenarios how to apply logical reasoning based on the learned embeddings; extend the embedding approaches to handle logics that also incorporate (n-ary) relations (along the line of [20], say); combine cones with other classical machine learning approaches and compare them with the SVM-based ones developed in this paper; and last but not least, develop a ML embedding approach tailored towards cones—i.e., from scratch, without relying on classical ML approaches.

Declarations

Funding

The research of Mena Leemhuis is funded by AI-Lab Lübeck[5]. Diedrich Wolter acknowledges financial support by Technologie-Allianz Oberfranken and the BMBF-funded “Dependable Intelligent Systems Lab”.

Conflict of interest

The authors declare that they have no conflict of interest.

Availability of data and material

All data used for the experiments are publicly available (as indicated by pointers to the literature in the paper).

Code availability

Software code for all algorithms will be made available in case the paper gets accepted.

[5] https://ai-lab.digital-hub-luebeck.de/en

References

1. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., et al.: Gene ontology: tool for the unification of biology. Nature genetics **25**(1), 25 (2000). DOI 10.1038/75556
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: C.J.C. Burges, L. Bottou, Z. Ghahramani, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States., pp. 2787–2795 (2013). URL http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data
4. Burges, C.J.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery **2**(2), 121–167 (1998). DOI 10.1023/a:1009715923555
5. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST) **2**(3), 1–27 (2011)
6. Changpinyo, S., Chao, W.L., Sha, F.: Predicting visual exemplars of unseen classes for zero-shot learning. 2017 IEEE International Conference on Computer Vision (ICCV) pp. 3496–3505 (2017)
7. Conradie, W., Palmigiano, A., Robinson, C., Wijnberg, N.: Non-distributive logics: from semantics to meaning. arXiv e-prints arXiv:2002.04257 (2020)
8. Deng, J., Ding, N., Jia, Y., Frome, A., Murphy, K., Bengio, S., Li, Y., Neven, H., Adam, H.: Large-scale object classification using label relation graphs. In: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (eds.) Computer Vision — ECCV 2014, *Lecture Notes in Computer Science*, vol. 8689, pp. 48–64. Springer International Publishing (2014). URL http://dx.doi.org/10.1007/978-3-319-10590-1_4
9. Fofanova, T.: Encyclopedia of Mathematics, chap. Semi-modular lattice. Springer Science+Business Media B.V. / Kluwer Academic Publishers (2001)
10. Gärdenfors, P.: Conceptual Spaces: The Geometry of Thought. The MIT Press, Cambridge, Massachusetts (2000)
11. Garg, D., Ikbal, S., Srivastava, S.K., Vishwakarma, H., Karanam, H., Subramaniam, L.V.: Quantum embedding of knowledge for reasoning. In: H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, R. Garnett (eds.) Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc. (2019)
12. Gibaja, E., Ventura, S.: Multi-label learning: a review of the state of the art and ongoing research. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **4**(6), 411–444 (2014). DOI 10.1002/widm.1139
13. Goldblatt, R.I.: Semantic analysis of orthologic. Journal of Philosophical Logic **3**(1), 19–35 (1974). DOI 10.1007/BF00652069. URL https://doi.org/10.1007/BF00652069
14. Gutiérrez-Basulto, V., Schockaert, S.: From knowledge graph embedding to ontology embedding? an analysis of the compatibility between vector space representations and rules. In: M. Thielscher, F. Toni, F. Wolter (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018., pp. 379–388. AAAI Press (2018)
15. Ji, S., Tang, L., Yu, S., Ye, J.: Extracting shared subspace for multi-label classification. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, pp. 381—389. Association for Computing Machinery, New York, NY, USA (2008). DOI 10.1145/1401890.1401939. URL https://doi.org/10.1145/1401890.1401939
16. Kulmanov, M., Liu-Wei, W., Yan, Y., Hoehndorf, R.: El embeddings: Geometric construction of models for the description logic el++. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19) (2019)
17. Leemhuis, M., Özçep, Ö.L., Wolter, D.: Multi-label learning with a cone-based geometric model. In: M. Alam, T. Braun, B. Yun (eds.) Ontologies and Concepts in Mind and Machine - 25th International Conference on Conceptual Structures, ICCS 2020, Bolzano, Italy, September 18-20, 2020, Proceedings, *Lecture Notes in Computer Science*, vol. 12277, pp. 177–185. Springer (2020). DOI 10.1007/978-3-030-57855-8_13. URL https://doi.org/10.1007/978-3-030-57855-8_13
18. Matoušek, J. (ed.): Lectures on Discrete Geometry. Springer New York (2002). DOI 10.1007/978-1-4613-0039-7
19. Mehran Kazemi, S., Poole, D.: SimplE Embedding for Link Prediction in Knowledge Graphs. arXiv e-prints arXiv:1802.04868 (2018)

20. Özçep, Ö.L., Leemhuis, M., Wolter, D.: Cone semantics for logics with negation. In: C. Bessiere (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020 [scheduled for July 2020, Yokohama, Japan, postponed due to the Corona pandemic], pp. 1820–1826. ijcai.org (2020). DOI 10.24963/ijcai.2020/252. URL https://doi.org/10.24963/ijcai.2020/252
21. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: W. Buntine, M. Grobelnik, D. Mladenić, J. Shawe-Taylor (eds.) Machine Learning and Knowledge Discovery in Databases, pp. 254–269. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
22. Redei, M.: Quantum Logic in Algebraic Approach. Fundamental Theories of Physics. Springer Netherlands (1998). URL https://books.google.de/books?id=7ltemAP8MDUC
23. Rockafellar, R.T.: Convex Analysis. Princeton University Press, Princeton, NJ (1997)
24. Śmieja, M., Tabor, J., Spurek, P.: SVM with a neutral class. Pattern Analysis and Applications **22**(2), 573–582 (2017). DOI 10.1007/s10044-017-0654-3
25. Uzilov, A.V., Keegan, J.M., Mathews, D.H.: Detection of non-coding rnas on the basis of predicted secondary structure formation free energy change. BMC Bioinformatics **7**(1), 173 (2006). DOI 10.1186/1471-2105-7-173
26. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning **73**(2), 185 (2008). DOI 10.1007/s10994-008-5077-3
27. Wan, S.P., Xu, J.H.: A multi-label classification algorithm based on triple class support vector machine. In: 2007 International Conference on Wavelet Analysis and Pattern Recognition, vol. 4, pp. 1447–1452 (2007). DOI 10.1109/ICWAPR.2007.4421677
28. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering **29**(12), 2724–2743 (2017). DOI 10.1109/TKDE.2017.2754499
29. Xian, Y., Lampert, C.H., Schiele, B., Akata, Z.: Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**, 2251–2265 (2017)
30. Yih, W., Zweig, G., Platt, J.C.: Polarity inducing latent semantic analysis. In: J. Tsujii, J. Henderson, M. Pasca (eds.) Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea, pp. 1212–1222. ACL (2012). URL http://www.aclweb.org/anthology/D12-1111
31. Zhang, M.L., Li, Y.K., Liu, X.Y., Geng, X.: Binary relevance for multi-label learning: an overview. Frontiers of Computer Science **12**(2), 191–202 (2018). DOI 10.1007/s11704-017-7031-7. URL https://doi.org/10.1007/s11704-017-7031-7