Conjunctive and disjunctive version spaces with instance-based boundary sets

Summary

The thesis presents research on computational approaches to the concept-learning task. In this task, a learner receives training instances, each labelled as a positive or negative example of some unknown target concept. The goal is to find descriptions of the concept in a given concept language that accurately classify future unlabelled instances. To identify such descriptions the learner uses an acceptance criterion defined on the training data. One of the simplest acceptance criteria is the consistency criterion. It states that a description is consistent with the training instances of a target concept if the description correctly classifies those instances. When this criterion is employed the complete solution of the concept-learning task is a version space. Hence, the version space is defined as a set of all descriptions in the concept language that are consistent with the training instances of the target concept.

Version-space learning can be viewed as a search process in the concept language according to the constraints imposed by the training instances. The search is realised by the operations: *Initialise*, *Update*, and *Retraction*. The *Initialise* operation initialises the version space to be learned before the training instances are presented. Whenever a new training instance is presented the *Update* operation removes those descriptions from the version space that classify the instance incorrectly. Conversely, if a previously presented training instance is revoked the *Retraction* operation adds those descriptions to the version space that (1) classify the instance incorrectly, and (2) are consistent with the remaining training instances. The search process also requires the state-test operations *Converged?* and *Collapsed?*. The *Converged?* operation determines whether the version space consists of equivalent concept descriptions. The *Collapsed?* operation determines whether the version space is empty. If the version space is not empty, it can be used for instance classification. The instance classification is realised by the operation *Classify*. It is based on the rule of unanimous voting: an instance is classified if and only if all the descriptions in a version space agree on a classification of the instance. Since version spaces are sets, they are characterised by additional useful set operations *Member?*, *Intersection*, *Subset?* and *Equal?*.

The rule of unanimous voting can classify beyond training instances if the concept language is restricted and the target concept is represented in that language. The

concept language is restricted in the sense that it is incomplete. The concept language is incomplete if and only if at least one concept from the domain of discourse cannot be expressed in the language. Therefore, the inductive bias[1] of version spaces is a restriction bias.

In this thesis, we consider the definition of version spaces together with their basic operations (described above) as an abstract data type. Given a concept language, an implementation of the version-space abstract data type requires designing (1) a version-space representation, and (2) algorithms for the basic version-space operations based on that representation. A version-space representation is a finite data structure from which the version space to be learned can be derived. Given a concept language, for a version-space representation we distinguish two types of adequacy for the basic version-space operations. The representation is epistemologically adequate, if each operation can be implemented by an algorithm based on the representation. The representation is heuristically adequate, if the algorithm of each operation is tractable. So, the heuristical adequacy of a version-space representation is closely related to the tractability of the representation. A version-space representation is tractable for a concept language if it can be computed in time polynomial in the number of the training instances and the relevant properties of the language.

The standard version-space representation is by boundary sets. They are defined to contain the minimal and maximal descriptions of version spaces described in partially-ordered concept languages. Hence, the boundary sets delimit version spaces, although it is a well-known fact that they are intractable for most concept languages.

There are four problems in the domain of version spaces of which the intractability of the boundary sets is only one. We list the four problems below.

- **Problem with Incomplete Concept Languages:** if a target concept is not represented in a concept language, the version space to be learned is empty.
- **Computational Problem:** the standard version-space representation by boundary sets is intractable for most concept languages.
- **Retraction Problem:** most of the existing version-space representations are computationally inefficient for the operation *Retraction*.
- **Problem with Noisy Training Instances:** if at least one training instance is noisy, the description of the target concept is removed from the version space to be learned.

The problems described above are formulated in the problem statement investigated in this thesis:

[1] The inductive bias is a set of assumptions, that together with the training data, determine how a concept learner classifies future unlabelled instances.

Do there exist version spaces and their corresponding representations that can simultaneously handle:

(1) the problem with incomplete concept languages;

(2) the computational problem;

(3) the retraction problem; and

(4) the problem with noisy training instances?

In chapter 1 we formulate the problem statement. This thesis is an attempt to find answers to that problem statement. The next chapters describe them in detail.

In chapter 2 we formalise the concept-learning task. The task is viewed as a search task. In this context we introduce the notion of inductive bias.

In chapter 3 we propose to consider version spaces as an abstract data type. The data type is specified by the definition of version spaces and their set of basic operations. We analyse the properties of version spaces. Along with the main advantages two principal problems of version spaces are emphasised that correspond to the first and fourth part of the problem statement.

In chapter 4 we try to answer the first part of the problem statement. The key idea is to extend concept languages using either logical conjunction or disjunction. We prove that the conjunctive and disjunctive extensions of concept languages are more complete than the languages themselves (i.e., they can represent more concepts). This is the reason for introducing conjunctive and disjunctive version spaces. They are defined as version spaces in the conjunctive and disjunctive extensions of concept languages. We show that conjunctive and disjunctive version spaces are more complete than (ordinary) version spaces. Hence, they are solutions of more concept-learning tasks. In addition, we prove that conjunctive (disjunctive) version spaces are nonempty for all possible concept-learning tasks if and only if the conjunctive (disjunctive) extensions of concept languages are complete. Since the conjunctive and disjunctive extensions of concept languages are not necessarily complete, conjunctive and disjunctive version spaces are a partial solution to the problem of incomplete concept languages (the first part of the problem statement). Following the main line of chapter 3 we consider conjunctive and disjunctive version spaces as abstract data types. We prove that the conjunctive and disjunctive version-space abstract data types can be implemented using the version-space abstract data type.

In chapter 5 we consider the problem of implementing the version-space abstract data type. In this context we survey three existing version-space representations: (1) list representation, (2) boundary sets, and (3) unilateral boundary sets. They are presented in terms of their epistemological and heuristical adequacy for the basic version-space operations as well as in terms of their tractability. We improve the representations by developing algorithms of those basic version-space operations of which the implementation details were not considered previously. An analysis of the

representations and the algorithms of the operation *Retraction* leads to an adequate formulation of the problems that correspond to the second and third part of the problem statement.

In chapter 6 we answer the second and third part of the problem statement. The key idea is to consider the version space to be learned as an intersection of version spaces based on particular training instances. We prove that their boundary sets taken together delimit the version space. Therefore, the indexed families of these boundary sets are proposed as a version-space representation called instance-based boundary sets. We prove the epistemological adequacy of the instance-based boundary-set representation with respect to a subset of the basic version-space operations including the operation *Retraction* for the classes of intersection-preserving and union-preserving languages, introduced in the chapter. We derive the conditions for the tractability and the heuristical adequacy of the representation for the same subset of operations. A fair analysis of these conditions shows that they can easily be met in practice. Therefore, we conclude that the instance-based boundary-set representation simultaneously solves the computational problem and the retraction problem, that correspond to the second and third part of the problem statement. (This implies that the representation can be used for efficient implementations of the version-space abstract data type.)

In chapter 7 we answer the first, second, and third part of the problem statement. We show that the representations from chapters 5 and 6 can represent conjunctive and disjunctive version spaces and are epistemologically adequate with respect to the basic conjunctive and disjunctive version-space operations (for the concept languages for which they have been designed). Consequently, we conclude that the conjunctive and disjunctive version-space abstract data types can be implemented using each of these representations. Moreover, we determine when the representations are heuristically adequate for the basic conjunctive and disjunctive version-space operations, and when they tractably represent conjunctive and disjunctive version spaces. In this context, the conjunctive and disjunctive version spaces represented by instance-based boundary sets are considered as a solution to the problem with incomplete concept languages, the computational problem, and the retraction problem, that correspond to the first, second, and third part of the problem statement.

In chapter 8 we answer the fourth part of the problem statement. We propose to apply a particular classification algorithm, namely the k-nearest neighbour algorithm on the instance-based boundary sets of conjunctive and disjunctive version spaces. This results in two instance-based classification schemes. We show that their inductive bias is a preference bias since during the classification there exists a strong preference for certain concept descriptions over others. Nevertheless, the schemes do not require any change in the instance-based boundary-set representation of conjunctive and disjunctive version spaces. Hence, they can be added to the conjunctive and disjunctive version-space abstract data types realised with instance-based boundary sets. We test the systems, based on the classification schemes, on

standard datasets. We note that their classification performance is comparable with that of the C4.5 decision-tree learner. In addition, we combine the systems with the condensed nearest-neighbour algorithm and the edited nearest-neighbour algorithm. The first combination aims at reduction of the computational complexity, but the experimental results show a significant drop in the classification performance. The second combination aims to achieve robustness of the classification performance with respect to noise in training instances. Such a robustness is confirmed by experimental results.

Next to the instance-based classification schemes, described above, in chapter 8 we consider a second approach for applying preference biases with instance-based boundary sets although it is only briefly sketched. The approach is a methodology for designing a broad spectrum of separate-and-conquer algorithms based on instance-based boundary sets.

From these results we conclude that the instance-based boundary sets can be used for designing learning procedures that implement different types of inductive biases including those that can handle noise in training instances. Thus, the fourth part of the problem statement is solved.

Chapter 9 ends the thesis. A proper evaluation of the problem statement is presented. In summary, it reads:

If the concept languages are intersection-preserving or union-preserving, the conjunctive and disjunctive version spaces represented by instance-based boundary sets can solve simultaneously:

(1) the problem with incomplete concept languages;

(2) the computational problem;

(3) the retraction problem; and

(4) the problem with noisy training instances.

Samenvatting

Dit proefschrift beschrijft onderzoek naar computationele benaderingen om een bepaald begrip (een concept) te leren. Een lerende agent krijgt een aantal oefenvoorbeelden voorgelegd die elk zijn voorzien van een label dat aangeeft of een voorbeeld wel of niet tot het concept behoort. De agent zoekt op basis van de verzameling oefenvoorbeelden naar een goede beschrijving in een gegeven concepttaal. Het doel is om beschrijvingen te vinden die de agent in staat stelt om toekomstige voorbeelden (die niet zijn voorzien van een label) correct te classificeren.

Hiertoe zoekt de agent naar beschrijvingen in de concepttaal die voldoen aan een acceptatiecriterium. Een eenvoudig voorbeeld van een dergelijk criterium is het consistentiecriterium. Volgens dit criterium is een beschrijving consistent met de voorbeelden behorende bij een te leren concept indien de beschrijving de oefenvoorbeelden correct classificeert. Een op basis van het consistentiecriterium verkregen oplossing van de taak om een concept te leren vormt een zogeheten versieruimte. Een versieruimte is de verzameling van alle beschrijvingen die consistent zijn met de oefenvoorbeelden van het te leren concept.

Het leren van concepten via versieruimten kan worden beschouwd als het zoeken naar een beschrijving die in overeenstemming is met de oefenvoorbeelden. Het zoekproces wordt uitgevoerd in termen van de operaties: *Initialise*, *Update*, en *Retraction*. De operatie *Initialise* initialiseert de versieruimte alvorens de oefenvoorbeelden worden aangeboden. Zodra er een gelabeld voorbeeld wordt aangeboden, verwijdert de operatie *Update* alle beschrijvingen in de versieruimte die inconsistent zijn met het oefenvoorbeeld. Wanneer een eerder aangeboden oefenvoorbeeld wordt ingetrokken dan voegt de operatie *Retraction* alle beschrijvingen toe die dat voorbeeld als incorrect classificeren en die tevens consistent zijn met de overige oefenvoorbeelden.

Daarnaast maakt het zoekproces gebruik van twee testoperaties: *Converged?* en *Collapsed?*. De operatie *Converged?* bepaalt of de versieruimte bestaat uit equivalente conceptbeschrijvingen. De operatie *Collapsed?* bepaalt of de versieruimte leeg is. Indien de versieruimte niet leeg is kan deze gebruikt worden voor de classificatie van voorbeelden middels de operatie *Classify*. Deze operatie hanteert de regel van eenstemmigheid. Volgens deze regel wordt (i) een voorbeeld als positief geclassificeerd (zoals behorend tot het geleerde concept) wanneer alle beschrijvingen in de

versieruimte consistent zijn met het voorbeeld; (ii) een voorbeeld als negatief geclassificeerd (d.w.z., niet behorend tot het geleerde concept) wanneer alle beschrijvingen in de versieruimte inconsistent zijn met het voorbeeld; en (iii) een voorbeeld als niet classificeerbaar beschouwd in alle andere gevallen. Aangezien versieruimten verzamelingen zijn, gebruiken we nog een aantal bruikbare operaties op verzamelingen, namelijk *Member?*, *Intersection*, *Subset?* en *Equal?*.

De regel van eenstemmigheid kan voorbeelden buiten de oefenvoorbeelden classificeren als de concepttaal beperkt is en het beoogde concept gerepresenteerd wordt in die taal. De concepttaal is beperkt in de zin dat deze onvolledig is. Een concepttaal is onvolledig dan en slechts dan als minstens één concept uit het domein niet in die taal kan worden uitgedrukt. Daarom is de inductieve bias[2] van een versieruimte een restrictie-bias.

In dit proefschrift worden de definitie van versieruimten alsmede de bijbehorende operaties beschouwd als een abstract datatype. Bij een gegeven concepttaal vereist de implementatie van het abstracte datatype van een versieruimte de specificatie van (1) een versieruimterepresentatie, en (2) algoritmen voor de elementaire versieruimte-operaties. Een versieruimterepresentatie is een eindige datastructuur in de concepttaal waaruit de te leren versieruimte kan worden afgeleid. Bij een gegeven concepttaal gebruiken we twee manieren om adequaatheid van de versieruimte-operatoren aan te geven: epistemologisch adequaat en heuristisch adequaat. Een representatie is epistemologisch adequaat indien iedere operatie geïmplementeerd kan worden door een algoritme dat gebaseerd is op die representatie. Een representatie is heuristisch adequaat indien iedere operatie geïmplementeerd kan worden door een algoritme dat "tractable" is.

Het heuristisch adequaat zijn van een versieruimterepresentatie voor de elementaire versieruimte-operaties hangt nauw samen met de "tractability" van de representatie. Een versieruimterepresentatie is "tractable" voor een concepttaal indien deze kan worden berekend voor alle mogelijke verzamelingen van de geleerde voorbeelden in een tijd die polynomiaal is met betrekking tot de omvang van de oefenvoorbeelden en de relevante eigenschappen van de concepttaal.

"Boundary sets" vormen de standaardrepresentatie voor versieruimten. Bij deze representatie wordt uitgegaan van partieel geordende concepttalen. "Boundary sets" bevatten de minimale en maximale beschrijvingen van versieruimten. "Boundary sets" bakenen versieruimten af, hoewel het algemeen bekend is dat zij voor de meeste concepttalen "intractable" zijn. Het domein van de versieruimten kent vier problemen waarvan de "intractability" van de "boundary-set" representaties er één is. De vier problemen zijn:

- **Het probleem van onvolledige concepttalen:** als een te leren concept niet in een concepttaal gerepresenteerd is, dan is de bijbehorende versieruimte leeg.

[2] De inductieve bias is een verzameling aannamen die, tezamen met de geleerde voorbeelden, bepalen op welke wijze ongelabelde (niet geleerde) voorbeelden worden geclassificeerd.

- **Het computationele probleem:** de standaard-versieruimterepresentatie in termen van "boundary sets" is niet "tractable" voor de meeste concepttalen.
- **Het retractieprobleem:** de meeste bestaande versieruimterepresentaties zijn computationeel inefficiënt met betrekking tot de operatie *Retraction*.
- **Het probleem van ruis in de oefenvoorbeelden:** als de te leren voorbeelden ruis bevatten, worden beschrijvingen van het te leren concept verwijderd uit de versieruimte.

Deze vier problemen zijn vervat in de centrale probleemstelling van dit proefschrift:

Bestaan er versieruimten en bijbehorende representaties die tegelijkertijd een oplossing bieden aan de volgende vier problemen:

(1) Het probleem van onvolledige concepttalen;

(2) Het computationele probleem;

(3) Het retractieprobleem; en

(4) Het probleem van ruis in de oefenvoorbeelden?

Dit proefschrift poogt een antwoord te vinden op deze probleemstelling. De structuur van het proefschrift is als volgt. Hoofdstuk 1 geeft een introductie en beschrijft de probleemstelling.

Hoofdstuk 2 formaliseert de taak om een begrip (concept) te leren. De taak wordt beschouwd als een zoektaak. Het begrip inductieve bias wordt geïntroduceerd.

In hoofdstuk 3 worden versieruimten beschreven in termen van een abstract datatype. Het datatype wordt gespecificeerd in termen van de definitie van versieruimten en de bijbehorende verzameling van elementaire operaties. De eigenschappen van versieruimten worden geanalyseerd. Naast een bespreking van de voornaamste voordelen, worden twee problemen van versieruimten nader uitgewerkt: het probleem van onvolledige concepttalen (deel 1 van de probleemstelling) en het probleem van ruis in de voorbeelden (deel 4 van de probleemstelling).

Hoofdstuk 4 richt zich op het probleem van de onvolledige concepttalen. Het centrale idee is om de concepttaal uit te breiden met behulp van logische conjuncties of disjuncties. Er wordt bewezen dat conjunctieve en disjunctieve uitbreidingen van een concepttaal volledige zijn dan de oorspronkelijke concepttaal (d.w.z., zij bevatten meer concepten). Derhalve worden conjunctieve en disjunctieve versieruimten geïntroduceerd. Deze versieruimten zijn gedefinieerd als versieruimten die gebaseerd zijn op de conjunctieve en disjunctieve uitbreidingen van de concepttaal. Vervolgens wordt bewezen dat conjunctieve en disjunctieve versieruimten volledige zijn dan (gewone) versieruimten. De uitgebreide versieruimten zijn daarom oplossingen

voor een groter aantal concept-leertaken. Bovendien wordt bewezen dat conjunctieve (disjunctieve) versieruimten niet leeg zijn voor alle mogelijke concept-leertaken dan en slechts dan als de conjunctieve (disjunctieve) uitbreidingen van de concepttalen volledig zijn. Aangezien de conjunctieve en disjunctieve uitbreidingen van concepttalen niet noodzakelijkerwijs volledig zijn, vormen conjunctieve en disjunctieve versieruimten enkel een gedeeltelijke oplossing voor het probleem van de onvolledige concepttalen. In aansluiting op hoofdstuk 3 worden conjunctieve en disjunctieve versieruimten beschouwd als abstracte datatypen. Er wordt bewezen dat de conjunctieve en disjunctieve versieruimten geïmplementeerd kunnen worden met behulp van het abstracte datatype voor versieruimten.

Hoofdstuk 5 richt zich op het probleem van de implementatie van het abstracte datatype voor versieruimten. Er worden drie bestaande versieruimterepresentaties onderzocht: (1) lijstrepresentaties, (2) "boundary sets", en (3) "unilaterale boundary sets". De representaties worden gepresenteerd in termen van de begrippen epistemologisch adequaat en heuristisch adequaat voor de ("tractability" van) elementaire versieruimte-operaties. Deze representaties worden verbeterd door het ontwikkelen van algoritmen voor elementaire versieruimte-operaties waarvan de implementatie nog niet eerder werd onderzocht. Analyse van de representaties en bijbehorende algoritmen van de operatie *Retraction* leiden tot een adequate formulering van het computationele probleem (deel 2 van de probleemstelling) en het retractieprobleem (deel 3 van de probleemstelling).

Hoofdstuk 6 behandelt het computationele probleem en het retractieprobleem. Het stelt een nieuwe versieruimterepresentatie voor die kan worden gebruikt voor de implementatie van het abstracte datatype van versieruimten. Het centrale idee is om de te leren versieruimte te beschouwen als een doorsnijding van versieruimten die gebaseerd zijn op specifieke verzamelingen oefenvoorbeelden. Er wordt bewezen dat de gezamenlijke "boundary sets" de versieruimte afbakenen. Vervolgens wordt de "instance-based boundary-set" representatie voorgesteld. Deze versieruimterepresentatie bestaat uit geïndexeerde families van "boundary sets". Met betrekking tot een deelverzameling van de elementaire versieruimte-operaties wordt bewezen dat de "instance-based boundary-set" representatie epistemologisch adequaat is. De deelverzameling bevat de operatie *Retraction* voor de klassen van "intersection-preserving" en "union-preserving" talen. Daarnaast worden de voorwaarden voor de "tractability" en het heuristisch adequaat zijn van de representaties voor dezelfde deelverzameling afgeleid. Uit een analyse van deze voorwaarden blijkt dat er in praktische situaties gemakkelijk aan voldaan kan worden. Om die reden worden "instance-based boundary sets" beschouwd als een oplossing voor zowel het computationele probleem als het retractieprobleem.

Hoofdstuk 7 beantwoordt de eerste drie delen van de vraagstelling. Er wordt aangetoond dat de in de hoofdstukken 5 en 6 voorgestelde representaties geschikt zijn voor conjunctieve en disjunctieve versieruimten en dat zij tevens epistemologisch adequaat zijn met betrekking tot de elementaire conjunctieve en disjunctieve

versieruimte-operaties (voor de concepttalen waarvoor zij zijn ontworpen). Hieruit wordt geconcludeerd dat de abstracte datatypen van conjunctieve en disjunctieve versieruimten kunnen worden geïmplementeerd met behulp van deze representaties. Bovendien wordt bepaald wanneer de representaties heuristisch adequaat zijn voor de elementaire conjunctieve en disjunctieve versieruimte-operaties, en wanneer zij "tractable" conjunctieve en disjunctieve versieruimten representeren. Conjunctieve en disjunctieve versieruimten, geïmplementeerd middels "instance-based boundary sets", worden beschouwd als een oplossing voor het probleem van onvolledige concepttalen, het computationele probleem, en het retractieprobleem.

Hoofdstuk 8 behandelt het probleem van ruis in de oefenvoorbeelden. Hiertoe wordt voorgesteld om het "k-nearest neighbour" classificatie-algoritme toe te passen op de "instance-based boundary sets" van conjunctieve en disjunctieve versieruimten. Dit leidt tot twee "instance-based" classificatieschema's. Er wordt aangetoond dat de inductieve bias van deze schema's een preferentie-bias is. Gedurende classificatie wordt de voorkeur gegeven aan bepaalde conceptbeschrijvingen boven andere. Niettemin vereisen geen van beide schema's aanpassingen aan de "instance-based boundary-set" representaties van de conjunctieve en disjunctieve versieruimten. Om die reden kunnen zij worden toegevoegd aan de abstracte datatypen van de conjunctieve en disjunctieve versieruimten die zijn geïmplementeerd met "instance-based boundary sets".

Classificatiesystemen, gebaseerd op beide schema's, zijn getest op standaarddatasets. Uit de resultaten blijkt dat de prestatie vergelijkbaar is met de prestatie van het op een beslisboom gebaseerde C4.5-classificatiesysteem. Daarnaast zijn de systemen gecombineerd met het "condensed nearest neighbour" algoritme en het "edited nearest neighbour" algoritme. De eerste combinatie beoogt een reductie van de computationele complexiteit, maar levert een aanzienlijk lagere prestatie op. De tweede combinatie beoogt een meer robuuste classificatie in geval van ruis in de voorbeelden. De experimentele resultaten bevestigen dat het gecombineerde systeem beter bestand is tegen ruis in de oefenvoorbeelden.

Naast de hierboven beschreven "instance-based" classificatieschema's, wordt in het kort een tweede aanpak geschetst voor het toepassen van preferentie-biases bij "instance-based boundary sets". De aanpak is een methodologie voor het ontwerpen van een breed spectrum van verdeel-en-heers-algoritmen die zijn gebaseerd op "instance-based boundary sets".

Dit alles leidt tot de conclusie dat "instance-based boundary sets" kunnen worden gebruikt voor het ontwerpen van leerprocedures met verschillende typen van inductieve biases, waaronder die voor het omgaan met ruis in de oefenvoorbeelden.

Hoofdstuk 9 besluit met het beantwoorden van de probleemstelling:

Indien de onderliggende concepttalen "intersection-preserving" of "union-preserving" zijn, lossen conjunctieve en disjunctieve versieruimten gerepresenteerd door "instance-based boundary sets" de volgende vier problemen op:

(1) Het probleem van onvolledige concepttalen;

(2) Het computationele probleem;

(3) Het retractieprobleem; en

(4) Het probleem van ruis in de oefenvoorbeelden.