# Communicative Version Space Learning by Relevance Sharing

M. Sevenster[1] M. van Someren[2]

University of Amsterdam
Roetersstraat 15
1018 WB Amsterdam
The Netherlands

**Abstract** In many application settings different agents learn different concepts which are related because they need the same set of attributes. An extension of the Candidate Elimination algorithm is described that detects which attributes are relevant and exchanges this information with learners in related domains using a blackboard. This speeds up learning and reduces the memory load during learning.

## 1 Introduction

Many tasks involve learning that takes place at physically different locations which are related in content. Examples are medical data that are collected in different locations, logs of computer users, etc. What is to be learned in different locations is likely to be related: the same diseases may be studied, though in different contexts, users may have similar interests or similar structures. If the nature of the relation between different contexts is known, then this can be exploited in the learning process. The most direct way is of course to collect all data and apply a standard learning method. The context can be added as an additional variable. However, for some tasks this is not sufficient. It may be that information about specific cases cannot be exchanged for external reasons, in particular privacy or the amount of data may prohibit collecting and using all data in one physical location.

In such cases we are dealing with target hypotheses that are similar in the sense that it is most likely that the same attributes are needed to predict (or recognize) a criterion but details of the relation are different. So in jargon: the

[1] sevenstr@science.uva.nl
[2] maarten@swi.psy.uva.nl

same attributes are relevant, but that does not hold for the attribute-values. Among practitioners it is well known that selection of attributes is at least as important as the learning method that is employed to construct the model. The success and failure of applications depends heavily on the selection of attributes in the modeling stage of the project. In consulting practice, this is a well-known fact: expertise in modeling (relevant attributes and type of relation) is considered of greater value than technical details of an analysis.

This suggests a need for methods that do not exchange specific observations but that instead exchange information about the relevance of attributes for predicting closely related criterion variables, such as the same attributes observed in different settings or contexts. The approach is based on the Candidate Elimination algorithm because this lends itself well for analysis and because it includes a clear inductive bias. The basic Candidate Elimination algorithm is extended with an explicit bias, a component that finds relevant and irrelevant attributes (that can be communicated to other learners) and a component that adapts the version space to changes in bias (relevance) that are found by other agents.

The setting that we consider here consists of a set of $n$ learning agents. Each agent receives positive and negative nominal training examples, drawn according to an unknown distribution over the example space, and uses these to learn a conjunctive nominal target hypothesis. That is, the target hypothesis can be described by a conjunction of attribute-value pairs. The target hypotheses that agents are learning may be different but we assume that they share relevance of attributes: the same attributes are relevant for all target hypotheses. After presenting the method for conjunctive hypotheses and a noise-free domain, we discuss the prospects of relaxing these assumptions.

## 2 The Candidate Elimination Algorithm

The Candidate Elimination algorithm [5], from here on referred to as CE algorithm, modifies version spaces. A version space is the set of all hypotheses that are consistent with the examples that have been observed. A hypothesis $h$ is consistent with database $D$ iff $h$ classifies each example $D$ correctly. The version space that is the result of learning from examples $D$, is denoted as $\mathcal{VS}(D)$. The CE algorithm uses a generality order to represent a version space $\mathcal{VS}(D)$ as two boundary sets, such that

$$\mathcal{VS}(D) = \langle S, G \rangle, \tag{1}$$

with $S$ the set containing the most specific hypothesis and $G$ the most general hypotheses consistent after learning database $D$. Hypothesis $h_1$ is more specific than or equally specific to $h_2$ (denoted as $h_1 \leq h_2$) iff each example, that is classified positively by $h_2$ is also classified positively by $h_1$. Each hypotheses $h$ is in $\mathcal{VS}(D)$ iff

$$(\exists s \in S)(\exists g \in G)(s \leq h \leq g), \tag{2}$$

with $\mathcal{VS}(D) = \langle S, G \rangle$.

New training examples bring two bounds closer together and learning stops when the version space converges to a single hypothesis. A disadvantage of the CE algorithm is the space needed to represent the sets $S$ and $G$. Haussler [4] showed that the $G$-set can grow exponentially in the number of attributes (or in the number of examples), even when learning conjunctive hypotheses. Smirnov [11] partially solved this problem by representing the $G$-set by negative examples. The size of the $G$-set grows, in the worst-case, still exponentially in the number of attributes, but grows linearly in the number of examples. In order to reduce the number of attributes we are currently testing on a variant of the CE algorithm that is extended with the capability of attribute selection.

# 3 The Relevance Communicating Candidate Elimination Algorithm

The Relevance Communicating Candidate Elimination algorithm (RCCE algorithm) works like the original CE algorithm but is extended in two ways: (1) it learns which attributes are relevant and (2) it exchanges this information with other learners and uses their relevance information for feature selection.

## 3.1 Finding Relevant and Irrelevant Attributes

After revising the $S$-set from a positive example, a learner $\mathcal{L}$ that follows RCCE finds all attributes that are irrelevant in all hypotheses in the $S$-set. Since attribute $a$ is irrelevant in version space $\mathcal{VS}(D)$ iff there is no hypothesis $h \in \mathcal{VS}(D)$ in which $a$ is mentioned, with $\mathcal{VS}(D) = \langle \{s\}, G \rangle$. Therefore, if $a$ is not mentioned in $s$, it is irrelevant in $\mathcal{VS}(D)$. [3] After finding an irrelevant attribute, $\mathcal{L}$ writes this information on the blackboard. This information is useful to other learners in the domain, since we assumed that all learners in the domain share relevance of attributes. Conversely, if all hypotheses $h \in \mathcal{VS}(D)$ have attribute $a$ mentioned we call it relevant. If attribute $a$ is relevant in version space $\mathcal{VS}(D)$ can be concluded from the $G$-set of $\mathcal{VS}(D)$. Namely, if for each hypothesis $g \in G$ holds that $a$ is mentioned, $a$ is relevant in $\mathcal{VS}(D)$. Proving from the examples in $D$ that attribute $a_i$ is relevant in $\mathcal{VS}(D)$ boils down to finding a set $D'$, such that $D' \subseteq D$ and only containing positive examples and a single negative example $x$ such that

[3] Proving from the examples in $D$ that attribute $a_i$ is irrelevant in $\mathcal{VS}(D)$ boils down to finding two positive examples $x$ and $y$ in $D$, such that $(x)_i \neq (y)_i$, with $(z)_j$ denoting the $j$-th value in example $z$.

$$\textit{Hamming-distance}(s, x) = 1, \tag{3}$$

with $s$ obtained from $\mathcal{VS}(D^+) = \langle\{s\}, G\rangle$. Again due to the relevance assumption, learner RCCE writes this information on the blackboard.

### 3.2 Using Relevance Information to Update a Version Space

If a learner $\mathcal{L}$ that follows RCCE is told (by another learner) that attribute $a$ is relevant or irrelevant in $\mathcal{VS}(D)$ then this information can be used to reduce the version space:

```
IF a is irrelevant in VS(D) = ⟨S, G⟩
   THEN generalize all s ∈ S over a
        generalize all g ∈ G over a
        remove all g ∈ G such that (∃g' ∈ G)(g ≤ g')

IF a is relevant in VS(D) = ⟨S, G⟩
   THEN specialize all g ∈ G over a
        remove all g ∈ G such that (∃g' ∈ G)(g ≤ g')
```

These steps are included in a communicating version of RCCE.

## 4 Communicating CE Learning

The communicating version of the method exchanges knowledge about relevance via a blackboard: when a learner notices that an attribute is (ir)relevant, it writes this information on the blackboard that can be accessed by all agents. The communicating version exploits the properties of CE learning with a conjunctive hypothesis language that learning can be "factored" over attributes ([3], ch.7): the relation between an attribute and the positive class does not depend on the values of other attributes. Therefore also the relation between *relevance* of an attribute and the positive class does not depend on value or relevance of other attributes. Since we assume that the target hypothesis may vary between learners but the *relevance* is the same, information about relevance from different learners can be pooled and exploited. Formally the blackboard is a triple of the form

$$\langle BBR, BBI, BBU\rangle, \tag{4}$$

in which the variables denote the set of attributes that are known to be relevant, irrelevant and unknown, respectively. It is initialized as follows: $BBR = BBI = \emptyset$ and $BBU$ the set of all attributes under which learner $\mathcal{L}$ is learning. Updating the version space from examples follows the standard CE algorithm. There are several options for organizing the communication: a learner can read

the blackboard after a given amount of time, before or after observing a new example but there are more possibilities. Which is best depends on the relative costs and benefits of communication. Here we assume that at each cycle a training example is processed and the blackboard is read and written. Learning continues until a threshold is reached after which the target hypothesis is considered roughly learned. The threshold is based on the notion of $\epsilon$-exhaustion, introduced in the next section. In each cycle, the version space is updated from a training example, relevant and irrelevant attributes are collected and written on the blackboard, new information about relevance is read from the blackboard and the version space is updated, following the rules in Section 3.2, from this. For the sake of completeness we show the pseudo-code of the RCCE algorithm using the blackboard:

Relevance Communicating Candidate Elimination algorithm

RCCE$(D, \epsilon, \delta)$:
$\mathcal{VS} = \langle\{\bot\}, \{\vec{*}\}\rangle$, $m = 0$, $R = \emptyset$, $I = \emptyset$
WHILE $m < \frac{1}{\epsilon}(\ln\frac{1}{\delta} + \ln|\mathcal{VS}(\emptyset)|) \ \wedge \ |\mathcal{VS}| > 1$ DO
  pick $\langle x, t(x)\rangle$ from $D$
  CE$(\langle x, t(x)\rangle, \mathcal{VS}) = \mathcal{VS}$
  collect relevant attributes in $\mathcal{VS}$ in $R'$
    $BBR = BBR \cup (R' \backslash R)$
    $R = R \cup R'$
  collect irrelevant attributes in $\mathcal{VS}$ in $I'$
    $BBI = BBI \cup (I' \backslash I)$
    $I = I \cup I'$
  specialize $\mathcal{VS}$ over all attributes in $R \backslash BBR$
  generalize $\mathcal{VS}$ over all attributes in $I \backslash BBI$
RETURN $\mathcal{VS}$

The RCCE algorithm does not make assumptions on the representation language for hypotheses. The consequences of enriching the hypothesis language will be discussed Section 6.

# 5 Convergence

In this section we discuss convergence of the version spaces and of the relevance information. Haussler [4] gives a bound of the *true error* of a hypothesis in a version space after learning database $D$ containing $m$ examples. The *true error* is the probability that the hypothesis will make an incorrect prediction for an example. If $\mu$ is a distribution defined over the example space, from which the examples are drawn and $t$ is a target hypothesis then the true error of hypothesis $h$ is equal to

$$\mu\{x \in \mathbf{X} | h(x) \neq t(x)\}. \tag{5}$$

In general, the exact true error of a hypothesis is not known, since it depends on the unknown $\mu$ and $t$. But one can make a statement, after observing database $D$ of size $m$, on the probability that the true error of an arbitrary hypothesis from version space $\mathcal{VS}(D)$ is within an user-defined range. Haussler made it possible to state that the true error of each hypothesis $h \in \mathcal{VS}(D)$ is smaller than $\epsilon$, with confidence $(1 - \delta)$. The three variables ($m$, $\epsilon$ and $\delta$) are related through the following equation:

$$m < \frac{1}{\epsilon}\left(\ln\frac{1}{\delta} + \ln|\mathbf{H}|\right), \tag{6}$$

with $|\mathbf{H}|$ the size of the hypothesis space. A hypothesis space $\mathbf{H}$ is equal to $\mathcal{VS}(\emptyset)$. The notion of $\epsilon$-exhausting can be used as a stopping criterion for The algorithm. By giving an acceptable probability of an error, the number of training examples can be derived. The RCCE algorithm terminates when the version space is $\epsilon$-exhausted (or when the version space has converged, in this case the version space has surely met the $\epsilon$-exhaustedness threshold), with confidence $(1 - \delta)$. But terminating does not guarantee that all attributes are known to be relevant or irrelevant. In this section we will show that the probability that one will find an example that proves the irrelevance of attribute $a_i$ is a decreasing function in the number of learners $n$. Let $s$ be the most specific conjunctive hypothesis, in the $\epsilon$-exhausted version space $\mathcal{VS}$, with respect to confidence parameter $\delta$. And let $a_i$ be an attribute that is relevant nor irrelevant in $\mathcal{VS}$. To proof that $a_i$ is actually irrelevant to the target hypothesis $t$, one should draw an example $x$ for which holds that:

$$(x)_i \neq (s)_i \ \wedge \ t(x) = +, \tag{7}$$

with $(z)_j$ denoting the $j$-the value of example $z$. The probability that an example satisfying the Constraint 7 is drawn depends on the unknown distribution $\mu$. Let us use $\mu_i^+$ to refer to the probability equal to

$$\mu\{x|(x)_i \neq (s)_i \ \wedge \ t(x) = +\}. \tag{8}$$

However, after $m$ examples attribute $a_i$ still was not found to be irrelevant. The probability that $m$ times no example like $x$ is drawn is equal to $(1 - \mu_i^+)^m$. If we recycle the confidence parameter $\delta$ one can give a bound for $\mu_i^+$, as follows:

$$(1 - \mu_i^+)^m < \delta, \tag{9}$$

which is necessarily the case when the following equation holds:

$$e^{-\mu_i^+ m} < \delta. \tag{10}$$

Solving this equation, while remembering that in the worst-case

$$m = \frac{1}{\epsilon}\left(\ln\frac{1}{\delta} + \ln|\mathcal{VS}(\emptyset)|\right), \tag{11}$$

shows that $\mu_i^+$ is proportional to $\epsilon$, because:

$$\mu_i^+ \;>\; \frac{1}{m}\ln\frac{1}{\delta} \;=\; \left(\frac{\ln\frac{1}{\delta}}{\ln\frac{1}{\delta} + \ln|\mathcal{VS}(\emptyset)|}\right)\epsilon. \tag{12}$$

But now we forget that none of the other $n-1$ learners did find an example proving the irrelevance of attribute $a_i$. So instead of $m$ examples $nm$ examples do not satisfy Constraint 7. So the probability that one draws an example that proves the irrelevance of an attribute is maximally

$$\frac{1}{nm}\ln\frac{1}{\delta}. \tag{13}$$

In the above we have made clear that, although the target hypotheses differ, stronger bounds can be given for the probability that the irrelevance will be proven.

Beside faster convergence of individual learners, the main reason for RCCE algorithms is saving memory space. Memory-space is saved by the RCCE algorithm, because relevance information reduces the hypothesis space dramatically and therefore the version space, although that depends on the precise content of the database. If attribute $a_i$ is known to be irrelevant, all hypotheses mentioning $a_i$ can be removed, which reduces the hypothesis space (in the worst-case, when $a_i$ is a binary attribute) with a factor $\frac{1}{3}$. Therefore an immediate transfer of relevance information limits the memory-space of the version space.

In general this will reduce the memory load but it is not trivial how precisely. It is clear, however, that when $n$ learners are learning, they can find the irrelevance of attributes more quickly compared to a single learner

# 6 Generalizing RCCE

The restriction to conjunctive hypothesis languages, noise-free domains and strictly shared relevance are very strong and make the method only of theoretical interest. In this section we discuss relaxing these restrictions.

## 6.1 More Expressive Languages

We have only considered the conjunctive hypothesis language-case but the RCCE algorithm can also be applied to more expressive languages. Consider for instance the $k$-DNF hypothesis language. In this case the space needed to represent the $S$ and $G$-set is larger than for conjunctive hypotheses but the gain in

memory space by co-learning is also larger. Finding (ir)relevant attributes is not fundamentally harder in a $k$-DNF version space, than in a conjunctive version space. Checking if attribute $a_i$ is irrelevant in the $k$-DNF version space $\mathcal{VS}$, for instance, boils down to checking if every disjunct in a $k$-DNF hypothesis $h$ from the $S$-set of $\mathcal{VS}$ of the form $h_1 \vee h_2 \vee \ldots \vee h_k$ does not mention attribute $a_i$. It is straightforward to detect relevance and irrelevance in attributes with more than two values and for intervals for numerical attributes in the hypothesis language.

Let us now consider the difference between single CE and distributed RCCE algorithms. We can make two comparisons. The first is to assume that in each cycle each learner observes one new training example, writes new relevance information on the blackboard, reads from the blackboard and uses the new relevance information to update its version space. In this case we can compare the number of cycles of a single learner with the number of cycles for $n$ learners. The number of cycles will clearly be smaller. The precise effect clearly depends on the hypothesis language, the distribution of training examples and the proportion of relevant attributes. As long as there is no direct use for version spaces we will not try to clarify this relation. The results should be seen as a framework of communicative concept-learning. Another comparison can be made in a slightly different setting. Suppose that in each cycle only one learner observes a new example, updates its version space, reads from the blackboard, updates its version space, finds relevant and irrelevant attributes and writes these on the blackboard. The learners take turns in this. Will the learning curves of individual learners differ from the setting in which they learn individually? Here again, the effect will be that RCCE learning is faster.

At least we expect that the RCCE algorithm will perform well in environments in which the distributions differ highly. It might be the case that in some environment an attribute is constant. As a consequence the learner learning under this attribute can never make a statement on the (ir)relevance of this attribute.

## 6.2 Uncertain Hypotheses and Relevance

A drawback of version spaces is given by the fact that they can not handle noise. The presented data should be totally free of noisy examples. An example $x$ is noisy if it is associated with a distribution of classes instead of a single class. E.g. the approach in [8] can be used for a probabilistic version of the CE algorithm. A threshold is used to decide when there is enough evidence to shift the boundaries of the version space. This method can be extended to the decision that an attribute is relevant or irrelevant.

### 6.3 Discovering Shared Relevance

If we relax the assumption that all criteria have the same relevant attributes then we must resort to a kind of bootstrapping approach in which observed shared relevances are used for grouping version spaces that appear to share relevances. Specifying the details of this and interleaving it with the RCCE method is a problem that we hope to address in the future.

## 7 Discussion

The RCCE method combines ideas that were introduced separately before. The notion of explicit, "declarative" bias was explored by a number of authors, e.g. Russell studied the notion of "determinations", e.g. [10, 9], Morik and others [6] use declarative "meta-rules", [12] and [7] studied language bias in the context of Inductive Logic Programming and [2] in the context of classification. Although the ideas in this paper grew out of this earlier work and the literature on the Candidate Elimination algorithm, the multi-agent setting makes these ideas much more useful than they were before, in the setting of single learners acquiring their own bias or bias acquired manually from a human domain expert. We showed that the method can be extended to more realistic hypothesis languages. Most studies of multi-agent learning focus on how multiple learners can complement each other in learning about a single domain. For example, [1] describes a form of multi-agent learning in which learning agents learn about different parts of the attribute space and, unlike our setting, combine their results.

## References


[1] W. Davies and P. Edwards. Dagger: Using instance selection to combine multiple models learned from disjoint subsets.

[2] J. G. Ganascia. Improvement and refinement of the learning bias semantic. In *Proceedings of the European Conference on Artificial Intelligence*, pages 384–389, Munich, Germany, August 1–5 1988.

[3] M. R. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos, California, 1987.

[4] D. Haussler. Quantifying inductive bias: Ai learning algorithm's and valiant's framework. *Artificial Intelligence*, 36(2):177–221, 1988.

[5] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1983.

[6] K. Morik, S. Wrobel, J.-U. Kietz, and W. Emde. *Knowledge acquisition and machine learning*. Academic Press, London, 1993.

[7] C. Nédellec, C. Rouveirol, H. Adé, F. Bergadano, and B. Tausend. Declarative bias in ILP. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 82–103. IOS, 1996.

[8] S. W. Norton and H. Hirsh. Learning dnf via probabilistic evidence combination. In *Proceedings of the Tenth International Conference on Machine Learning (ICML93)*, pages 220–227. Morgan Kaufmann Publishers, 1993.

[9] S. Russell and B. Grosof. A sketch of autonomous learning using declarative bias. In P. Brazdil and K. Konolige, editors, *Machine learning, meta–reasoning and logics*, pages 19–54, Dordrecht, The Netherlands, 1990. Kluwer.

[10] S. J. Russell. *The use of knowledge in analogy and deduction.* Pitman, London, 1989.

[11] E. Smirnov. *Conjunctive and Disjunctive Version Spaces with Instance Based Boundary Sets.* Shaker Publishing, Maastricht, the Netherlands, 2001.

[12] B. Tausend. A guided tour through hypothesis spaces in ilp. In N. Lavrac and S. Wrobel, editors, *Machine Learning: ECML-95*, pages 245–259, Berlin, 1995. Springer.