

MACHINE LEARNING

A Focus on Version Space Search

Nikesh Garera

B.E. Computer
Engineering
D1 – 66
Thadomal Shahani
Engineering College,
Bandra

B-8, Sabari, Near BJP
Office, Sec –15, Vashi,
Navi Mumbai 400705
Ph : +91-22-7881511

nikesh@ieee.org

Pejus Das

B.E. Computer
Engineering
D1 – 62
Thadomal Shahani
Engineering College,
Bandra

278/c, Surya mahal, 1st
Floor, Rm no.18,
Girgaum Road,
Thakurdwar.
Mumbai 400004.
Ph : +91-22-3894253

pejus@rediffmail.com

Abhishek Ghildyal

B.E. Computer
Engineering
D3 – 103
Thadomal Shahani
Engineering College,
Bandra

5/11,Prabhakar Kunte
Hsg. Soc. , Adarsh Nagar,
Andheri (W), Mumbai
400102
Ph : +91-22-6335171

tue81@yahoo.com

MACHINE LEARNING

A Focus on Version Space Search

Nikesh Garera

B.E. Computer Engineering
nikesh@ieee.org

Pejus Das

B.E. Computer Engineering
pejus@rediffmail.com

Abhishek Ghildyal

B.E. Computer Engineering
tue81@yahoo.com

ABSTRACT

Herbert Simon defines learning as

“..any change in a system that allows it to perform better the second time on reception of the same task or on another task drawn from the same population”
- (Simon, 1983).

Machine learning is important for practical applications of artificial intelligence. Cost and complexity however, are the major obstacles to its widespread use. In this paper we study the following topics: Inductive learning, concept, version space search. We also propose a technique by which the operation of a query engine can be made more efficient using version space search.

1. INTRODUCTION

- *Symbolic AI and Connectionism*

Artificial Intelligence can be broadly classified into two approaches - symbolic AI, which advocates a computer model of mind and connectionism, argue for a brain model of mind.

Symbolic AI constructs its model of mind using computation as a metaphor. ‘Mental activity is like the execution of a stored program.’

Connectionism bases its model of mind on a nervous system metaphor. ‘Mental activity is like the settling of a network into a stable configuration’. While it is still debatable as to which approach leads us closer, the goal remains the same. That of building an intelligent machine.

- *Machine Learning*

A machine must be able to learn to do new things and to adapt to new situations before it can be considered intelligent. Herbert Simon describes learning as allowing the learner to “perform better the second time”. Induction, which is learning a generalization from a set of examples, is one of the most fundamental learning tasks. Concept learning is a typical inductive learning problem: given examples of some concept, infer a definition that will allow the learner to correctly recognize future instances of that concept.

- *Version space search*

This paper focuses on one algorithm used for concept induction, version space search.

A version space is a representation that enables you to keep track of all the useful information supplied by a sequence of learning examples, without remembering any of the examples. The version space consists of overly general models and an overly specific models. Specialization of the general models and generalization of the specific models ultimately leads to just one, guaranteed-correct model that matches all observed positive examples and does not match any negative examples. At this point the machine has learned.

- *Approach*

We begin by explaining Inductive learning and how it brings about machine learning. Concept space and Version space search are then introduced as techniques to implement inductive learning. A summary of the drawbacks of the version space search and their possible solutions is presented.

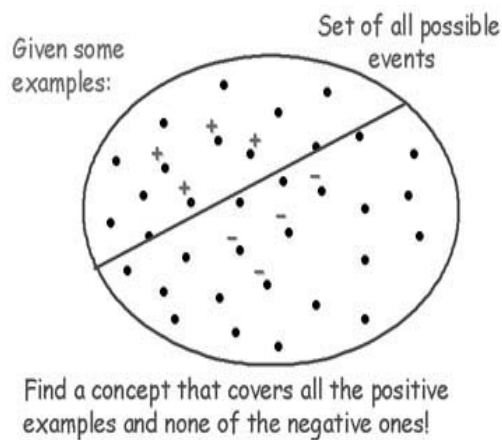
A database query engine must search a large database to find the correct records to return. This presents an excellent area to which version space search can be put to use. In closing, we apply version space search to the field of query optimisation.

2. INDUCTIVE LEARNING

One of the Machine learning techniques is the Inductive learning method. As the name suggests, Induction is learning through examples, a generalization is made from the set of given examples of a concept to be learnt. Arriving at the generalization is the most fundamental learning tasks. Concept learning is a typical inductive learning problem in which a definition is inferred from the given examples of a concept such as a cat, good stock investment, a ball etc. This inferred definition will allow the learner to correctly recognize future instances of that concept. Thus learning takes place.

This can be implemented as a search through the given list of examples of the concept. We will see one such search technique (Version Space search) a little later.

CONCEPT LEARNING AND CONCEPT SPACE



(Fig 2.1)

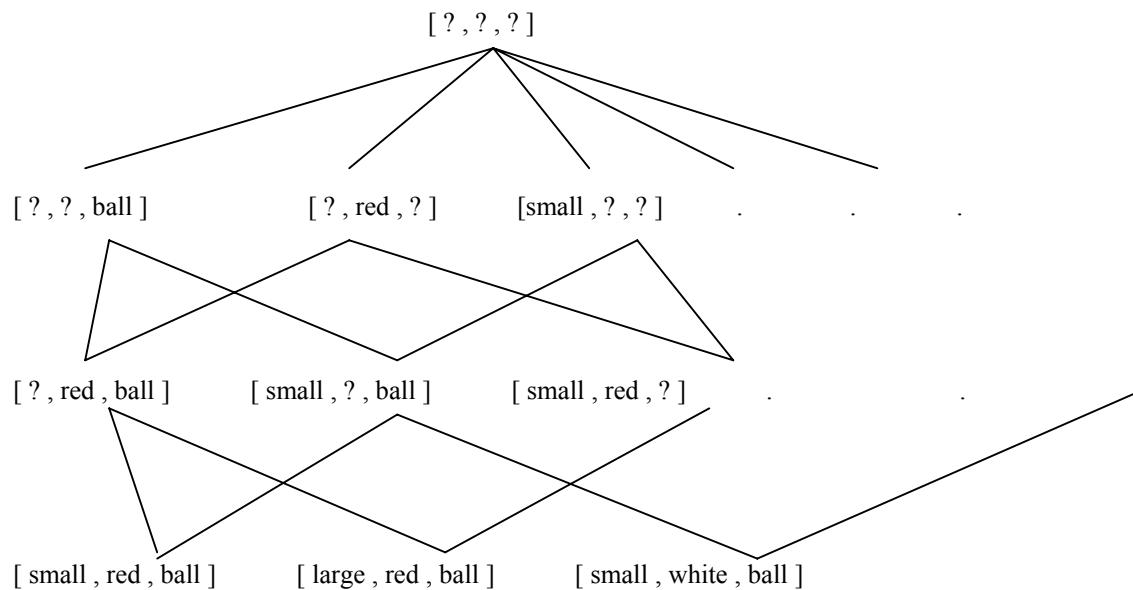
Machine learning programs use representation languages to represent concepts. Simple English sentences containing variables is a simple example of a representation language.
e.g. the concept “ball” may be represented by:

Format: [{size}, {color}, {shape}]
e.g. [small, red, round]

Various operations can be performed on these representations to form heuristic rules or plans that satisfy the goals of the learner. Typical operations include generalizing or specializing these expressions. The above concept “ball” can be further generalized by replacing constants with variables. The example given below is the generalization of the concept “ball” introduced above.

e.g. replacing constants with the symbol “?”:
(? represents most general value, it can be replaced by an possible value)

[small , red , round] can generalise to [? , red , round]



(Fig 2.2) A concept space for the concept 'ball'

The representation language together with the operations defines a space of potential concept definitions. This space is known as the concept space. The complexity of this concept space is a primary measure of the difficulty of a learning problem.

3. VERSION SPACE SEARCHING

We have seen that learning takes place by inferring a definition from the given example of a concept. This can be implemented by searching through the list of given examples.

Version space search is a concept learning technique to search the concept space.

VERSION SPACE

A **version space** is a representation that enables one to keep track of all the useful information supplied by the list of learning examples, without remembering any of the examples. It uses tree structures to represent learning examples. It consists of two trees, one containing nodes connected to overly general models and the other containing nodes connected to overly specific models. Each link between nodes in a version space represents either a specialization or generalization operation between the models connected to the nodes.

Thus a version can be summarized as a representation in which:

- There is a specialization tree and a generalization tree.
- Each node is connected to a model.
- One node in the generalization tree is connected to a model that matches everything.
- One node in the specialization tree is connected to a model that matches only one thing.
- Links between nodes denote generalization and specialization relations between their models.

The key idea in version-space learning is that specialization of the general models and generalization of the specific models ultimately leads to just one, guaranteed-correct model that matches all observed positive training examples and does not match any negative training examples.

Fig 3.1(a) shows a most general model possible and a most specific model. Fig 3.1(b) shows the specialization of the general model so that the models derived fail to match a new negative example. It also shows the generalization of the most specific model so that the models derived match the new positive examples.

The above rules can be summarized as follows.

When a positive example is encountered

- We generalize the specific model.
- We remove all the models in the general models' tree that fail to match the positive example (Fig 3.1(c)).

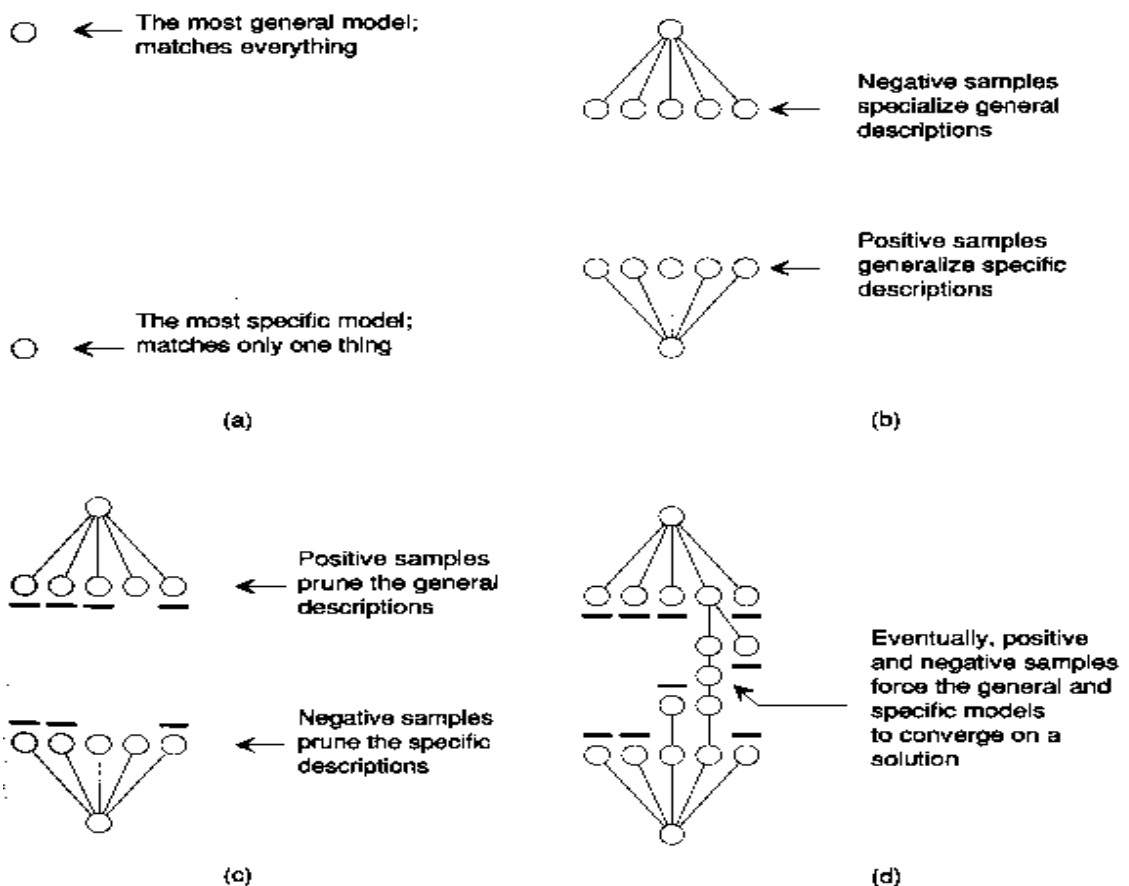
When a negative example is encountered

- We specialize the general model.
- We remove all the models in the specific models' tree that matches with the negative example (Fig 3.1(c)).

Eventually, one general model and one identical specific model may survive. When that happens, no further examples are needed because that sole survivor is sure to be the correct model, assuming that a correct model exists. Thus we see that generalization and specialization leads to the convergence of the version space.

Three more points are to be kept in mind when generalizing or specializing the tree models. Otherwise the two trees may never converge to an identical model. They are:

- Each time a general model is specialized that specialization must be a generalization of one of the specific models.
- Each time a specific model is generalized that generalization must be a specialization of one of the general models.
- Each time a general model is specialized that specialization must not be a specialization of another general model. Otherwise, a needless specific model will be retained in the set of general models.



(Fig 3.1) Learning in a version space.

4. AN EXAMPLE

→ a student has the following observations about having an allergic reaction after meals:

Restaurant	Meal	Day	Cost	Reaction
Alma 3	breakfast	Friday	cheap	Yes +
De Moete	lunch	Friday	expensive	No -
Alma 3	lunch	Saturday	cheap	Yes +
Sedes	breakfast	Sunday	cheap	No -
Alma 3	breakfast	Sunday	expensive	No -

→ **Concept to learn:** under which circumstances do I get an allergic reaction after meals ??

Having learned about version space in abstract terms, let us consider a concrete example (fig). One completely specific model in this food world is a list of values for situation-characterising attributes such as place, meal, day and cost. One example of such a model is [Alma 3, Dinner, Thursday, Expensive].

In contrast, the most general descriptive model does not specify any attribute value. One notation for this general model is a list of question marks [?, ?, ?, ?], with each question mark signaling that any attribute value in the corresponding position is allowed.

The model [Alma 3, ?, ?, Expensive] matches any situation in which your patient eats expensive food at Alma 3 on any day at any meal. Models such as this one lie between those that are completely specific and completely general model.

Generalization:

[Alma 3, ?, ?, Expensive] → [Alma 3, ?, ?, ?]

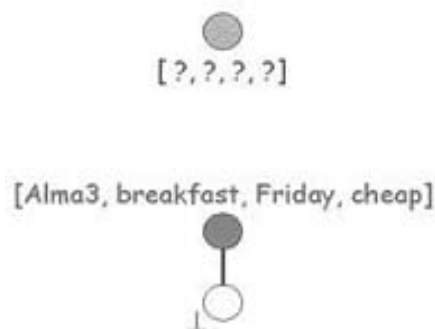
Specialization:

[Alma 3, ?, ?, Expensive] → [Alma 3, Dinner, ?, Expensive]

The following procedure describes the development of version space

- i. *Positive: [Alma 3, breakfast, Friday, Cheap]* (fig 4.1)

The first example, a positive one, initialises the version space.



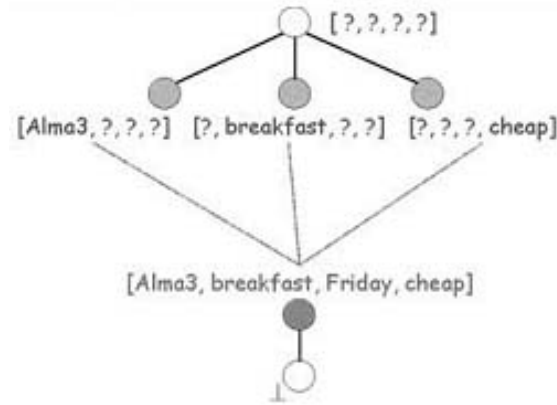
(Fig 4.1)

It contains the most general model [?, ?, ?, ?] along with a completely specific model [Alma 3, breakfast, Friday, cheap].

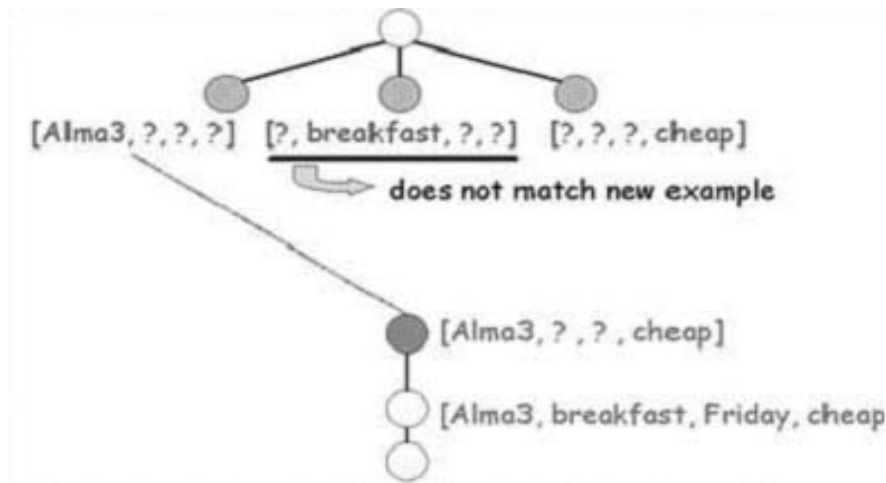
ii. Negative: [De Moete, lunch, Friday, expensive] (fig 4.2)

The second example, a negative one forces specialization of the most general model. While specializing, each ? in the general model is replaced with the corresponding part in the most specific model to ensure that each new specialization is a generalization of the most specific model. This, in turn, ensures that the general and the specific models will finally converge.

Note that [?, Friday, ?, ?] does not appear because it matches with the negative example [De Moete, lunch, Friday, Expensive].



(Fig 4.2)



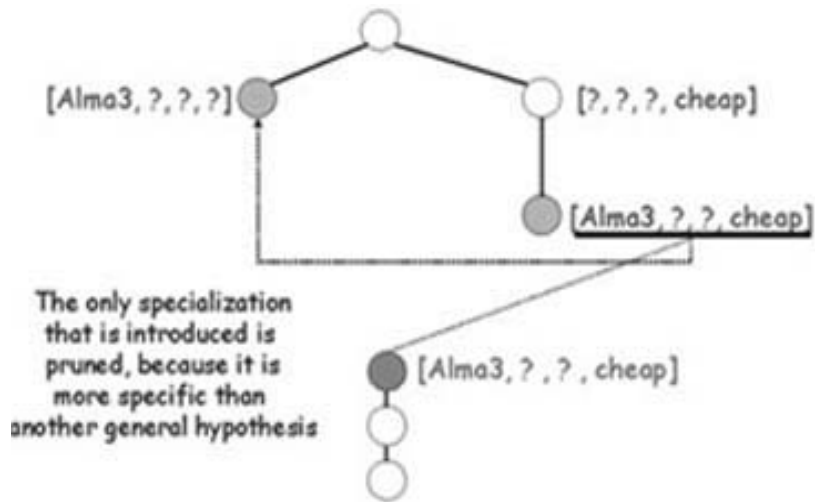
(Fig 4.3)

iii. Positive [Alma 3, lunch, Saturday, Cheap] (Fig 4.3)

The positive example generalises the specific model since we get different values for meal and day so we can generalise them.

[Alma3, breakfast, Friday, cheap] → [Alma3, ?, ?, cheap]

In the general model, [?, breakfast, ?, ?] does not match the new example so it will not lead to the road of convergence and hence it is pruned away.



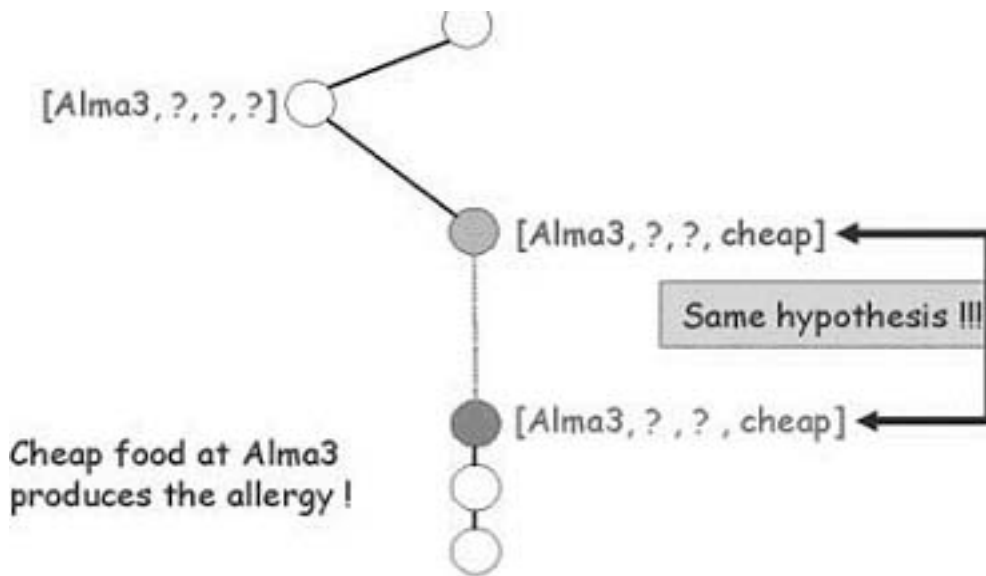
(Fig 4.4)

iv. Negative [Sedes, breakfast, Sunday cheap] (Fig 4.4)

The negative example specializes the general space, since specialization is done on the part which matches the example, i.e. [?, ?, ?, cheap]. As before, the specialization is done in the direction of the specific model which contains only [Alma 3, ?, ?, cheap], thus the only way to specialize is [?, ?, ?, cheap] → [Alma 3, ?, ? cheap].

But this specialization will be pruned since the other part in the general space [Alma 3, ?, ?, ?] is more general than [Alma 3, ?, ?, cheap]. So when the general and specific models converge, we can be sure that the positive and negative examples allow no other model.

At this point, only one general model, [Alma 3, ?, ?, ?] and only one specific model, [Alma3, ?, ?, cheap] remain.



(Fig 4.5)

v. Negative [Alma 3, breakfast, Sunday, expensive] (Fig 4.5)

This negative example brings the general and specific model at one point that they converge. It forces the specialization of the general model [Alma 3, ?, ?, ?] and the only way to specialize it in the direction of the specific model is [Alma 3, ?, ?, cheap] is to construct a new general model that is the same as the existing specific model.

Accordingly learning concludes. Cheap food at Alma 3 produces allergy.

5. EVALUATING VERSION SPACES

The version spaces concept demonstrates the way in which knowledge representation and state space search can be applied to the problem of machine learning. One interesting feature of this mechanism is that it *enables early recognition*. Once one or more negative examples have been analyzed, one can say with certainty that some examples cannot possibly be positive examples.

Although, version space has formed a new benchmark in machine learning, it contains drawbacks, which can be overcome but it may not be always practical to do so.

- **Problem – Complexity** : Search based learning, like all other search problems must take into account the complexity of search space. Because the version space search algorithm uses breadth first search, it could be inefficient while dealing with large spaces.
Solution : If an application is such that the version space might grow large; it may be useful to develop heuristics for pruning states in generalization and specialization spaces.
- **Problem – Noise** : Failure of the algorithm to converge may also be due to some noise or inconsistency in the training data. Version Space search algorithm is not particularly noise resistant. Even a single misclassified training instance can prevent the algorithm converging on the concept.
Solution : One solution to this problem maintains multiple generalization (G) and specialization (S) sets. In addition to the version space derived from all training instances, it maintains additional spaces based on all but one of the training instance, all but two of the training instances, etc. If G and S fail to converge, the algorithm can examine these alternatives to find those that remain consistent. Unfortunately, this approach is too inefficient to be practical in most cases.
- **Problem – Supervised Learning** : The version space technique is based on supervised learning i.e. the training examples provided to the algorithm should be classified as positive or negative in advance by us.
Solution : Unsupervised learning techniques can be used to determine, at a higher level, whether the example is positive or negative. This example along with its evaluated classification can be given to the version space search algorithm thus eliminating the need of predetermined classification of examples.

6. APPLICATION

QUERY OPTIMISATION USING VERSION SPACE TECHNIQUE

Query optimization performed by a query engine involves choosing one of several execution paths so as to minimize the load on the system resources. Query engines do this using techniques like Cost-based algorithm, Rule-based algorithm. The criteria for using the execution path includes memory space required, number of rows returned, the number of joins in the query etc..

Execution of a query requires searching the entire database for records that match the criteria specified. The cost of the search will be determined by the number of attributes in the query, the number of queries to be executed and the number of records to be searched.

In version space search the concept space is reduced as the negative and positive samples are processed terminating at a point where the general and specific models converge which gives us the solution. We can use this technique to optimize queries.

We assume that every query returns a table of records that satisfy the query. The records selected are those whose attributes match the condition(s) specified in the query. By reducing the number of attributes in the query we can optimize the query.

Consider a company manager who wants to select candidates for an interview. There are two attributes – designation, salary, and there are rules governing their selection. These rules are inferred from the positive (accept) and negative (negative) samples. The rules can be specified as -

- Select all candidates having Bachelor of arts degree
- Select all candidates who have no background education
- Reject all people with salary < 700

Given the above rules the version space algorithm can generalize that all candidates with salary ≥ 700 are to be selected. The result is a shorter query. The number of queries executed is also reduced. This not only requires less system resources but also executes faster.

Optimization can also be achieved by reducing the number of records to be returned. We use the same example of selecting candidates. After selecting candidate, if the manager wants to view the details of a particular candidate, the entire table of candidates need not be searched because the candidates matching the rejection criteria have been eliminated and the new search takes place in the reduced table of valid candidates. This dramatically increases the speed of execution.

7. CONCLUSION

Having discussed the version space search algorithm and coming up with an application in which it can be implemented, we can conclude that, version spaces is an important concept in machine learning which can be applied to many fields. We have discussed one such area (query optimisation) where it can be applied. The version space technique demonstrates the use of both positive and negative examples in reducing the concept space, thus in turn searching effectively. Several approaches have come up with modified versions of version spaces (like use of unsupervised learning) and it remains a useful tool in the field of machine learning.

8. REFERENCES

1. T.M Mitchell. Version Spaces: An Approach to Concept Learning. PhD thesis, Stanford University, Dec 1978.
2. "Machine Learning" David W. Aha, A tutorial presented at fifth international workshop on AI & stats, 4 Jan 1995
3. Learning by managing multiple memory models. Patrick Henry Winston, Artificial Intelligence.
4. Structures and Strategies for complex problem solving, Artificial Intelligence. Luger/Stubblefield
5. T.M Mitchell. Generalization as search. Artificial Intelligence, 18(2):203-226, March 1982.
6. The computational complexity of the candidate-elimination algorithm- Haym Hirsh, Rutgers University
7. Learning from examples: Induction. Elaine Rich, Kevin Knight, Artificial Intelligence.