**Weekly report of lessons**

**Name:** Mayank Kumar
**Roll No:** 19CS30029
**The week:** 16-08-2021 to 22-08-2021

## The topics covered:

Problems with Find-S Algorithm, Consistent Hypotheses, Agnostic Hypothesis, Version Space, List-Then-Eliminate Algorithm, Compact Representation of Version Space, Candidate-Elimination Algorithm

## Summary topic wise:

Problems with Find-S Algorithm:

- it throws away negative examples from the training set information
- can't determine if it has learned the concept as it picks a specific hypothesis h from the set H and leaves all others
- can't identity error in the training data as negative examples are ignored

Consistent Hypotheses:

Hypothesis h consistent with training examples set D of target concept c should satisfy the notation:

Consistent (h, D) $\equiv \forall$ <x, c(x)> $\in$ D :: h(x) = c(x)

Agnostic Hypothesis: May label erroneously a training sample

*Notation:* Agnostic (h, D) $\equiv \exists$ <x, c(x)> $\in$ D :: h(x) $\neq$ c(x)

Version Space:

Version Space ($VS_{H,D}$) is the subset of hypotheses from hypothesis space set H which are consistent with set of training examples D

*Notation:* $VS_{H,D} = \{h \mid h \in H \wedge \text{Consistent } (h, D)\}$

List-Then-Eliminate Algorithm:

A brute-force method to find the Version Space $VS_{H,D}$ where all the hypotheses in H are checked against training example. For each training example <x, c(x)>, hypothesis h is removed from $VS_{H,D}$ if h(x) $\neq$ c(x)

Compact Representation of Version Space:

Can be represented in terms of General Boundary (G) and Specific Boundary (S), where G is set of maximally general members consistent with D and S is set of maximally specific members consistent with D.

Every member of $VS_{H,D}$ lies between the S, G boundary

$VS_{H,D} = \{h \mid (h \in H) \wedge (\exists\ s \in S) (\exists\ g \in G) (g \geq_g h \geq_g s)\}$

Candidate Elimination Algorithm:

This algorithm computes version space containing all hypotheses from H consistent with an observed sequence of training examples. It finds the Version Space in terms of G *(initialized with maximally general hypothesis)* and S *(initialized with maximally specific hypothesis)*.

G is made more specific if a negative training example is encountered, while S is made more general if a positive training example is encountered.

## Concepts challenging to comprehend:

It was not trivial to understand why the List-Then-Eliminate algorithm is a brute-force procedure, but it became very clear when I realized that the algorithm requires exhaustive enumeration of all hypotheses, which is impractical and will cost a lot of time.
Another concept difficult to comprehend was how the specific boundary deals with positive examples while the general boundary deals with negative examples.

## Interesting and exciting concepts:

It is exciting to note that we are using operations like computing minimal generalizations and specializations of given hypotheses to define Candidate-Elimination Algorithm, so that it can always be applied given that those operations are well-defined.

## Concepts not understood:

None, all the concepts were clear to me.

## Any novel idea of yours out of the lessons:

Even the Candidate-Elimination algorithm converges on the assumptions that there are no errors in training set and there exist some hypothesis h in H describing target concept c. In case of error in training example, true target concept might get eliminated. So, there is need for a system to look into this issue. An idea for the same is that the learner should be allowed to conduct experiments in which it chooses next instance and obtains the correct classification for this instance from an external system. This might help the learner to converge towards the correct hypothesis if it exists.

---