Concept learner
Assignment 1

Oliver Rosander
Acronym: olra13
Email: rosanderoliver@gmail.com

Keywords—Concept learning, machine learning, classification, logical model, logic.

I. Hypothesis space and conjunctive concepts

The hypothesis space is the combination of all possible concepts in the expression to be generalised. In this task there are five features with a finite number of possible of values that can be assigned. [1]

$$3 * 3 * 3 * 2 * 2 = 108$$

By multiplying the number of possible values that can be assigned to each feature we get get number of possible combinations, which is the hypothesis space.

Using the same features we can calculate the number of possible conjunctive concepts by including the empty value to the set of feature values [1]. If we do that we get that the number of possible conjunctive concepts are:

$$4 * 4 * 4 * 3 * 3 = 576$$

II. Implementation

The assignment is written in Python 2.7. It is implemented using two classes. The class Attribute represents the features and the values they can assume, the second class Instance represents a set of Attributes.

The calculation of the least general conjunctive generalisation is done with the algorithm LGG-Set(D) (Algorithm 4.1) [1]. The algorithm takes data as an input and outputs a model, which is the logical model of the input data[1]. In the process LGG-Set(D) utilizes LGG-Conj(x,y) (Algorithm 4.2) to calculate the conjunction of the literals x and y. This is done repeatedly until there is no more training data to generalise from.

After reading the data from a file named instances.txt the class structure is built. This class structure is then used by LGG-Set(D) that loops over all instances and calls LGG-Conj(x,y) to compare and return the conjunctions of the input x and y.

```
expression = instances[0]
instances.pop(0)

for ins in instances:
    expression = LGGConj(expression, ins)
```

The algorithm **LGG-Conj(x,y)** the compare all the instances in x and y and removes the ones that have the same feature but not the same value from the instance x.

```
def LGGConj(gen, ins):

    for x in ins.attributes:
        for g in gen.attributes:
            if (g.expression == x.expression)
            and (g.value != x.value):
                    #remove expression from g
                    gen.remove(g)
    return gen
```

III. Result

When running the program over the data given in the assignment the output is:

> Least general generalisation of set using LGG-Conj algorithm (4.2): Creative=Yes AND Meticulous=Yes

This result is the most conservative generalisation that can be learned from the data. This means that every other generalisation is at least as general as the result. If we were to generalise the result more we would probably end up with an over-generalisation that covers negative examples to a greater extent than the LGG does with this sparse data set. To avoid this the data should include negative cases that remove the more general generalisations from the possible hypotheses. [1]

References

[1] Peter Flach, *Machine Learning The Art And Science of Algorithms that Make Sense of Data*, 1st ed. Cambridge, England: Cambridge university press, 2012.