# Inductive Logic Programming

Andrii Zakharchenko

## 1 Agent's behaviour

The agents start with an empty hypothesis and transforms each observation to a clause using a representation that will be discussed later. The policy of the agent can be described as follows:

$$y_k = \pi(h_k, o_k) = \begin{cases} 0, & \text{if } h_k \text{ is empty} \\ 1, & \text{if } h_k \subseteq_\theta o_k \\ 0, & \text{otherwise} \end{cases}$$

Returning a 0 when the hypothesis is empty allows agent to start learning from positive examples. And then agent continues to learn in a following way:

$$h_k = \begin{cases} h_{k-1}, & \text{if } r_k = 0 \\ lgg(h_{k-1}, o_{k-1}), & \text{otherwise} \end{cases}$$

## 2 Representation

The world of the agent is a set of cards $V \times S$, where $S$ is a set of shapes and $V = \{2, 3, 4, ..., 14\}$ is a set of values.

I use next predicates:

- **Greater**/2 is a predicate that for two variables $x, y \in V$ defines greater relation.
- **Equals**/2 is a predicate that defines equal relation.
- **CardsP0**/n predicate of arity n, where n is number of cards for one Oracle configuration. This predicate is true when cards belong to the player 0.
- **CardsP1**/n same as above, but for player 1.

I use next function symbols:

- **max**/n a function max : $V^n \to V$ returns maximal input element
- **sum**/n a function sum : $V^n \to V$ returns sum of input elements

- **val**/1 a function val : $V \times S \to V$ returns a value of a card
- **shape**/1 a function shape : $V \times S \to S$ returns a shape of a card.

This representation seems to me quite general, I don't use some very specific predicates (e.g. SumPlayer1IsGreater/0) but only very general relations such as greater and equals, also function symbols are quite general as well. Only **val** and **shape** are problem specific and their purpose is to distinguish statements such as Equals(val(x1), val(x2)) from Equals(shape(x1), shape(x2)), that is when we want to use the same relation whether on the value of a card or its shape.

Also, I use predicates CaprdsP0/1 to "bind" variables. Let me explain, let's say we have to observation Greater(sum(x1,x2), sum(y1,y2)) and next one is Greater(sum(y1,y2), sum(x1,x2)) these are two different observations but during generalizations this can be generalized to Greater(sum(v0,v1), sum(v2,v3)) but we want to be able to distnguish and we can do so by adding CardsP0(x1,x2), CardsP1(y1, y2).

The disadvantage of such representation is that to compare multiple cards by value and by shape I will need to add a lot of Equals predicates, which slows down a process of finding LGG.

# 3 New concept description

I implemented two new Rules:

- **HighCard**: according to this rules the player 0 wins if his highest cards if greater or euqal that player's 1 highest card.
- **AtLeastOnePair**: player 0 wins if he has at least one pair (by value) of cards, otherwise player 1 wins.

# 4 Experiments and discussion

## 4.1 Experiment 1

- **Oracle settings**: handSize=3, cards=5, shapes=['BELLS', 'HEARTS'], samples=500, rule=Flush()
- **Learned hypothesis**: !Equals(shape(v0), shape(v12)) , !Equals(val(v0), val(v9)) , !Equals(val(v20), val(v2)) , CardsP0(v0, v1, v2) , CardsP1(v3, v4, v5) , Equals(v16, v17)

Here we can see that agent learned that in order for player 1 to win player 0 should have his first card and any other card from his hand not equal, the agent learned two more rules regarding the euality of values of cards, which are redundant for Flush rule but are always true for this setting of Oracle due to limited number of card shapes.

### 4.2 Experiment 2

- **Oracle settings**: handSize=3, cards=5, shapes=['BELLS', 'HEARTS', 'CLUBS'], samples=500, rule=Flush()
- **Learned hypothesis**: Equals(shape(v3), shape(v6)) , CardsP0(v3, v4, v5) , CardsP1(v0, v1, v2)

In this experiment we added one more shape and we can see now that the agent's hypothesis does not contain additional rules, besides those which will satisfy the Flush rule.

### 4.3 Experiment 3

- **Oracle settings**: handSize=2, cards=13, shapes=['BELLS', 'HEARTS', 'ClUBS'], samples=samples, rule=HighCard()
- **Learned hypothesis**: !Equals(v6, v7) , CardsP0(v4, v5) , CardsP1(v2, v3) , Greater(max(val(v2), val(v3)), max(val(v4), val(v5)))

Here we can see that agent learnead that for player one to win his highest card should be strictly higher than the highest card of his opponent. Also, interesting fact that agent obtained a literal that states that no two cards should be the same, however I only constructed observations with (!)Equals(shape(x1), shape(x1)), (!)Equals(val(x1), val(x1)).

### 4.4 Experiment 4

- **Oracle settings**: handSize=3, cards=13, shapes=['BELLS', 'HEARTS', 'ClUBS'], samples=samples, rule=Sum()
- **Learned hypothesis**: !Equals(v6, v7) , CardsP0(v0, v1, v2) , CardsP1(v3, v4, v5) , Greater(sum(val(v3), val(v4), val(v5)), sum(val(v0), val(v1), val(v2)))

And again we see that the agent learned rule of sum and that two cards should not be the same.

### 4.5 Experiment 5

- **Oracle settings**: handSize=3, cards=13, shapes=['BELLS', 'HEARTS', 'ClUBS'], samples=samples, rule=AtLeastOnePair()
- **Learned hypothesis**: !Equals(val(v0), val(v1)) , !Equals(val(v0), val(v2)) , !Equals(val(v1), val(v2)) , CardsP0(v0, v1, v2) , CardsP1(v3, v4, v5)

So here we can see that agent one learned that in order for player 1 to win player should not have a pair of cards with same value.