

Formalization of Induction Logic in Biomedical Research

Marina Sapir

Aureon Biosciences, 28 Wells Ave, Yonkers, NY 10701

marina@sapir.us

I. INTRODUCTION

Induction of dependencies given a set of research data is an important part of biomedical research. The research data contain records of observations which are associated with different classes or concepts. The observations are characterized by chosen features. The task is to generalize the observations to produce the definitions of the concepts in terms on conditions of the features.

In computational analysis, the problem is often reduced to an approximation of the dependency between the description of the observation and its class. Yet, the properties of the research data and requirements for the solutions make this problem different from the usual “approximation” problems. Biomedical research data are noisy, redundant and insufficient. The primary goal of the decision is to improve medical understanding of the problem. The solutions should imitate human logic both in tolerance of exceptions and in their interpretability.

The task invites a “soft computing” approach, formulated by Lofti Zadeh. The idea is to exploit the tolerance for imprecision and uncertainty to achieve tractability, lower cost, higher level Machine Intelligence and economy of communication.

On the other hand, the problem of induction is a logical problem which was studied for centuries by philosophers, starting with David Hume (1711-1776). Here we want to introduce formal induction logic as a tool in the soft computing arsenal for biomedical research. We demonstrate both theoretical and applied results achieved with this logic.

In our formalization of induction logic we employ the idea of “granulation” [1]. To implement this general approach, we propose two levels of the learning process. On the first level, some combinations of conditions, which are specific, mostly, for only one class (concept) are to be found. The predicates of this type, which describe homogeneous granules in the attribute space, are important for the understanding of the studied phenomena. The second level, aggregation, complements the granulation. On this level, the granules are assembled to define the concepts in a logical comprehensive form.

As mentioned in [1], two level learning was implemented in various learning approaches, starting with perceptrons. A similar two step process can be found in such concept learning methods as LAD (Logical Analysis of Data) [10]. However, in the first step, LAD uses “crisp” predicates, which do not

allow for contradicting observations. In our view, research data require probabilistic criterion for acceptance of the hypotheses. We do not impose a single acceptability criterion for all induction problems. Instead, we postulate some general statistical properties of the confirmation function.

The dataset with the records of observations is a random sample from a general population of all possible observations with the unknown probability laws. The goal of learning is to express in the logical language the probabilistic nature of the studied concepts.

Introduction of the apparatus of our induction logic allows us to confront the most important problem of the learning: the possibility to infer general rules having only a finite and random set of observations. We demonstrate that if the confirmation function satisfies the proposed axioms, almost all hypotheses are stable, meaning that for all sufficiently large datasets almost any hypothesis is always acceptable or always unacceptable.

We compare this induction logic with other models of reasoning with facts and rules after the section of formal definitions.

II. FORMAL INDUCTION LOGIC (IL): DEFINITIONS

The granulation level is intended to simplify the problem, and it allows simpler language for the aggregation level of induction. We start with the definition of the granulation level language.

A. Granulation Level Language.

The language on this level needs to safely handle features of different types. Generally, different features allow for different operations and predicates. For example, the value of temperature can be above or below some threshold, while the diagnosis may take only one of few meanings in the study. To take into account these distinctions, we propose building the language based on a many-sorted signature.

Definition 1: Input Signature is a many-sorted signature $\mathcal{A} = \langle T, V, F, P, C \rangle$, where

- $T = \langle T_1, \dots, T_k \rangle$ are sets of constants, called *sorts*; k is a natural number;
- $V = \langle v_1, \dots, v_k \rangle$ is a set of individual variables

$$\text{dom}(v_i) \subseteq T_i, i = 1, \dots, k;$$

- F and P are sets of constant functional and predicates symbols on these variables;

- $C = \langle c_1, \dots, c_m \rangle$ is a finite set of predicate variables, called **concepts**. Concepts are predicates of all the variables.

Each element from the sets F, P is a user defined function or predicate.

Consider a simple example. Suppose the goal is to distinguish the common cold from a strep throat. The signature may include the following sets of constants:

- 1) T_1 is the set of all different values of temperature which can be found on a medical thermometer;
- 2) T_2 has two values "stuffy", "not stuffy";
- 3) T_3 has two values "sore", "not sore";
- 4) etc ...

The signature may then contain following variables: *temperature* with values in T_1 , *nose*, with values in T_2 , *throat* with values in T_3 and some other. The set of predicates may contain:

- 1) $Fever(temperature) = (temperature > 98)$;
- 2) $HighFever(temperature) = (temperature > 102)$;
- 3) $SoreThroat(throat) = (throat = sore)$;
- 4) etc ...

The signature has two concepts: "cold" and "strep".

The sorts and sets of the predicates and functions may be infinite and indexed by parameters. For example, the set of predicates may include comparisons with constants parameters $\{p_a(x) = (x \geq a)\}$, while the set of functions may include multiplication by a constant parameters $\{f_a(x) = ax\}$, with the parameter a ranging over some sort T_i .

To illustrate the importance of the sorts of variables in user-defined functions, let us consider another example. Suppose, one needs to study how the difference in day and night body temperature of a patient affects a diagnosis. The researcher may introduce the function: $F(x_i, x_j) = x_i - x_j$, where the variables x_i and x_j are the night and day temperature of a patient, respectively. The language with this function may contain conditions on the difference of day and night temperature. However, the language will not allow meaningless formulas with conditions on the difference between temperature and blood pressure, for example.

The atomic formulas and literals are defined in the usual way: if p is a predicate symbol, $p(X)$ is an atomic formula, and $p(X)$ and $\neg p(X)$ are literals.

The legal expressions in the IL include formulas and *observations*, which will be defined latter. We distinguish two types of formulas in IL: *constant formulas* and *productions*.

Literals with the constant predicates from P are called *constant formulas*. If $a(V), b(V)$ are constant formulas, the expression $(a(V) \& b(V))$ is also a constant formula in IL.

Definition 2: Production in an input signature $\mathcal{A} = \langle T, V, F, P, C \rangle$ is a universal statement of the form

$$\forall v_1, \dots, \forall v_k H(v_1, \dots, v_k) \Rightarrow c(v_1, \dots, v_k),$$

where $H(v_1, \dots, v_k)$ is a constant formula, not necessarily dependent on all the variables, and c is a concept.

In our sample signature, an example of a constant formula is

$$SoreThroat(throat) \& HighFever(temperature).$$

An example of a production is

$$SoreThroat(throat) \& HighFever(temperature) \\ \Rightarrow strep(throat, nose, temperature).$$

For production $h = (\forall v_1, \dots, \forall v_k H(v_1, \dots, v_k) \Rightarrow c(v_1, \dots, v_k))$, we will use the short notation $h = (H \Rightarrow c)$. The left part of the production, the constant formula H is called *premise*, the right part is called *conclusion*. We do not allow any other type of formulas on the granulation level of induction.

Definition 3: An **observation** in the input signature Σ is a pair $\langle t, c \rangle$, where constant vector t , called an **argument of the observation**, is a sequence of values of some (different) variables in the signature Σ , and c is a concept predicate.

An observation is an expression which assigns a concept to a combination of values of some variables. In our example, the observations are records about patients. Each patient is characterized by his temperature, stuffiness of his nose, absence - presence of the pain in the throat, the confirmed diagnosis (strep throat or common cold).

Observation and formulas play different roles in the semantics of IL and in the induction problem, as we will see. Formulas are subjective, they can be formulated by the researcher. The observations are given, they constitute the input of nature in the dialogue of learning.

B. Semantics of the Granulation Level language.

In this subsection, we will be talking about statistical interpretation of the introduced language.

Since the variables are defined on sets of constants, all models in IL have the same base, called *attribute space*. For the input signature $\Sigma = \langle T, V, F, P, C \rangle$ with k sorts and m concepts, the *attribute space* $X(\Sigma)$ is the direct product of the sets of constants $T_i : X(\Sigma) = T_1 \times \dots \times T_k$. Denote the extended space $X'(\Sigma) = X(\Sigma) \times c_1 \times \dots \times c_m$.

In contrast to logics of deduction, semantics of IL requires the notion of an object of the research, a stochastic entity with unknown law. We call this object the *datasource*.

Definition 4: Given an input signature Σ with m concepts, **datasource** \mathcal{D} is pair of independent probability distributions over the attribute space $X(\Sigma)$:

- **distributions of probabilities of concepts** $Q = \langle Q_1, \dots, Q_m \rangle, Q_i(X) = P(c_i | X), X \in X(\Sigma)$;
- **the distribution of probabilities of data** $P(X), X \in X(\Sigma)$.

In our "strep throat" example, the datasource will have the probability of strep and probability of common cold for each possible combination of values of the parameters (*temperature, nose, throat*, etc.) and a probability of each such combination as well.

The distributions of the datasource are latent, i.e. not known to the researcher. They are properties of his opponent in the dialog of learning. Existence of such distributions is a formalization of the “principle of uniformity of nature” [9], which is a necessary assumption for meaningful induction. If we do not know that the probabilities will stay the same in the future, we can not extract any useful information from the experience.

Let us stress that, according to this definition, various concepts can have non zero probability on the same element of the attribute space. This formalizes the fact that the concepts may not be strictly distinguished in the language selected by the researcher. It follows that some observations with the same argument may have different concepts. In our “strep” example, some patients may have inconclusive combinations of features insufficient for a doctor to make a certain diagnosis.

Before we can define models in IL, let us introduce some other terms and notations. Call the pair (Σ, \mathcal{D}) of the input signature and the datasource the *context*. For a set of observations D , denote $B(D)$ the set of all its arguments, and $C_i(D) \subseteq B(D)$ arguments of observations with the concept c_i .

Definition 5: A finite set of observations D is **dataset** in the context $(\mathcal{A}, \mathcal{D})$ with m concepts, if

- the set $B(D)$ is a random sample from the distribution $P(X)$ of the datasource \mathcal{D} ;
- the subset $C_i(D)$ is a random sample from the distribution $Q_i(X)$ of the datasource \mathcal{D} , $i = 1, \dots, m$.

In the “strep” example, the dataset is the set of available records about patients with one of the two confirmed diagnoses (strep throat or common cold).

In IL, the pair of attribute space and a dataset $\langle X(\mathcal{A}), D \rangle$ is a model of the context. Note that the variety of possible models of the context is not restricted by a theory, as in traditional logic. Instead, the probability distributions of the datasource determine the probability of the models.

C. Acceptability of Productions

In IL, the truthfulness of the hypothesis (production) can not be known due to the limited number of observations. Instead, the hypotheses can be characterized as acceptable or not. A hypotheses is acceptable if the dataset supports it in some statistical sense. Next we define the statistical characteristics, which can affect acceptability of the production given the dataset.

Definition 6: For a production $h = (H \Rightarrow c)$, a dataset E , an observation $e = \langle x, d \rangle \in E$ is **covered** by h , iff the premise H are true on $x : H(x)$; e **confirms** h , iff $H(x)$ and $c = d$. The **support** $S_E(h)$ is a fraction of all observations in E , which are covered by h , **intensity** $I_E(h)$ is a fraction of observations in E , confirming h , among all the observations, which are covered by h ; **base** $B_E(h)$ is a fraction of all observations in E , which belong to the concept c .

Let us look at some conditions making a hypothesis acceptable in an example of a new drug test. Suppose, a blind test is conducted on two groups of patients. One group takes

the drug; another group takes placebo. Assuming the drug does not causes any harm, when the statement “drug helps” is considered acceptable? For example, if the test was conducted in groups of 3 or 4 people, it would not be convincing, even if all the patients given drug and none of the patients given placebo had an improvement. Thus, acceptability of the productions depends on the **size of the datasets**.

Now let us assume the groups of patients are large enough according to industry standards. If only 3 people in the drug group and none in the placebo group had an improvement, the difference will not be convincing either. In our terminology, the **support** of the production is not large enough in this case.

Yet, if improvement is noticed in 50% of the drug patients and in 3% of the placebo patients, the drug will be considered helpful, despite the fact that the drug does not have any affect on half of the recipients. In this case, the **intensity** 0.5 of the production may be sufficient for a positive conclusion.

Suppose, the drug group has an adverse reaction in 20% of patients, while in the placebo group there is only 1% of patients with this complication. In this case, the drug may be considered harmful. The frequency of the side effect (**base** of the production “drug \Rightarrow side effect”) is so small that even the 20% intensity may be sufficient for the conclusion.

Based on such examples, we consider acceptability $\gamma(h, E)$ of a production h on the dataset E , $|E| = n$, to be a binary function of the support, intensity, base of the production and size of the dataset:

$$\gamma(h, E) = g_n(S_E(h), I_E(h), B_E(h)),$$

where the function $g_n : \mathcal{N} \times [0, 1] \times [0, 1] \times [0, 1] \rightarrow \{0, 1\}$ is called a *confirmation function*.

The axioms below define general properties of the confirmation function.

Definition 7: The confirmation function $g_m(x, y, z) : \mathcal{N} \times [0, 1] \times [0, 1] \times [0, 1] \rightarrow [0, 1]$ satisfies the next requirements:

- 1) $g_m(x, y, z)$ monotonously increases with x, y, m and monotonously decreases with z ;
- 2) $g_m(0, y, z) = g_m(x, 0, z) = 0$;
- 3) for all $c > 0$ there exist $a, b, m : g_m(a, b, c) = 1$.

The first axiom formalizes the monotonicity of confirmation we illustrated on the drug test example. The second axiom postulates that any production without support or with zero intensity is not confirmed. The axiom makes hypotheses which can not be confirmed by observations unacceptable. The third axiom guarantees that notwithstanding the probability of a concept, some productions for this concept can be confirmed in large datasets.

D. Granular Induction Problem

The goal of the granular level of induction is generalization of the observations. Hence, to formulate the problem, we need to characterize not only acceptability, but the generality of the productions as well.

Definition 8: The production $h_1 = (H_1 \Rightarrow c_1)$ is more general than the production $h_2 = (H_2 \Rightarrow c_2)$, $h_1 \succeq h_2$, if the

conclusions c_1, c_2 are the same and in FOL $\forall X(H_2(X) \Rightarrow H_1(X))$.

Notice that the formula $H_2(X) \Rightarrow H_1(X)$ does not exist in IL.

If the productions $h_1 = (H_1 \Rightarrow c), h_2 = (H_2 \Rightarrow c)$ are both acceptable, and all the atomic formulas of the premise H_1 are included in the premise H_2 , the production h_1 is not only shorter, it is more general.

If h_1, h_2 are acceptable productions, and $h_1 \succeq h_2$, the production h_2 is not a satisfactory result of the induction process. We formalize this idea in the next statement of the problem.

Definition 9: The Granular Induction Problem (GrP): Given an input signature Σ , a dataset D and a confirmation function $g_m(x, y, z)$, to find all most general acceptable productions.

The productions which constitute the solution of the GrP will be referred as **goal productions**. As the most general, the goal productions include only necessary terms. Thus, the solution of the GrP elicits the most essential terms for the understanding of the studied phenomena.

One can see the GrP as a formalization of the goal of soft computing: to achieve economy of communication with controlled tolerance for the contradictions. On another hand, this is a formal introduction of the famous Occam's parsimony principle as a main criterion of learning.

E. Aggregation Level of Induction

By the definition of the GrP, the premise of goal productions describe maximal homogeneous "granules" of the attribute space. Each observation either belongs to the granule, i.e. the production covers the observation, or it does not. Thus, the granules may be used to build new binary features, associated with the concepts. We call these new binary features aggregate features. Since each production is associated with certain concept by its conclusion, the same is true about the aggregate features.

Each disjunction R of the aggregate features of a concept i can be associated with numbers of "errors" of the first order and of the second order e_1, e_2 on the dataset. The number e_1 is the number of observations of the concept i which do not satisfy the disjunction R . The number e_2 is the number of observations of other concepts which satisfy the disjunction R .

The general purpose of the aggregation level of induction is to find short disjunctions of the aggregate features for a certain concept, which make few of the first and second type errors. The problem allows various formalizations, depending on the importance of each type of errors and complexity of the disjunctions.

III. COMPARISON OF IL WITH OTHER MODELS OF REASONING WITH FACTS AND RULES

There are several approaches to the formalization of reasoning about rules and observations, examples. Below, we trace the parallels and distinctions between the inductive logic and other methods.

A. Induction Logic vs Logic Programming

The production of IL has a parallel in the Horn rule clause of logic programming (LP) [3]. However, LP uses crisp FOL for reasoning. Thus, in LP, a production is "true" if and only if it is true on all the observations it covers. In IL, this condition is neither necessary nor sufficient for acceptability, as we saw in our drug test example. Even if a production is confirmed by all observations it covers, it can be still unacceptable, if the sample size or support were too small. LP is concerned with deduction of facts and rules from logic programs rather than with induction of logic programs from the sets of observations. The task of IL is opposite.

B. Induction Logic vs Inductive Concept Learning

The task of the inductive concept learning is given sets of examples of two different concepts (positive and negative) and background knowledge, to find a hypothesis, which is true on all the positive examples and false on all negative ones [4]. The inductive concept learning uses FOL for reasoning and for evaluation of the acceptability of the hypotheses. The truthfulness of the hypotheses does not allow for exceptions or contradictions. As our illustration about the drug testing demonstrated, the relaxed requirement on the confirmation of hypotheses in IL is much closer to real medical logic in many cases. In medical classification problems, contradictions are both inevitable and tolerable. Unlike IL, the language of the inductive concept learning does not include user-defined functions or predicates.

C. Induction Logic vs Attributional Calculus

We refer to attributional calculus (AC) as it is presented in [2]. The idea of AC is closer to deduction, than induction because it uses the rules produced by some machine learning program on the database and the database itself to produce new, rewritten rules. The rules use both disjunction and conjunction in their language. More exactly, the language uses conjunctions of special type of disjunctions, which contain only the literals with the same variable. However, the language does not allow user-defined functions and predicates. The acceptability of the rules in AC is defined by their statistical characteristics on the database. The rule, with the statistics above the user defined threshold, is considered acceptable. This makes this calculus similar with the IL.

D. Induction Logic and PAC Learning

Valiant's [7] theory of probably approximately correct learning (PAC) explores learning from examples which tolerates errors in the learning concept. Classical PAC learning assumes the concept can be expressed in the selected language. More application-oriented agnostic PAC learning [8] does not make such an assumption. Instead, it approximates not the real concept but the most-close hypothesis in the selected class of hypotheses. The main question in both cases is an existence of an algorithm which approximates the concept almost certainly with low error for any distribution using a polynomial (on the error and probability) size dataset.

It is easy to see the differences between IL and PAC learning and other theoretical machine learning approaches [11]. IL rejects the notion of learning as error minimization and accepts a more opportunistic view. By definition, the confirmation function tolerates certain level of contradictions. Thus, in IL, the degree of confirmation makes a constrain on the acceptability of hypotheses. The main criterion of choosing the best hypotheses among the acceptable ones is generality which translates into simplicity and communication economy of the hypotheses.

IV. JUSTIFICATION OF INDUCTION

Inductive conclusions require a “leap of faith”: when some dependence is found on a given dataset, one needs to believe that the dependence is a law of nature rather than a property of the given data. Since this “leap of faith” makes the whole meaning of the induction, it is critical to find how justified is this generalization. We show that the assumed properties of the confirmation function warrant this possibility of induction.

First, we restrict the class of productions, which may be of interest.

Definition 10: A production $h = (H \Rightarrow c)$ is **singular**, if the probability of $H(V)$ equals zero in the datasource.

Say, if the watch always shows 7:00:00 AM of April 29 of 2004, it is still possible to find an exact day and time with it, by looking at it in a proper moment. But the probability that somebody who does not know date and time looks at the watch at exactly 7:00:00 on that day is zero. The production: “observation time is 7:00:00 AM of 04/29/04 \Rightarrow watch is exact” is singular. Usually, the purpose of research is general, not singular, productions.

Since the concept of singularity is defined by the unknown probabilities of the datasource, we can only estimate the singularity of a production by the dataset. To check if a production h in the dataset E with the support $S_E(h)$ may be singular, one can estimate a confidence interval of the proportion $S_E(h)$ based on the Central Limit Theorem. For datasets with more than 30 observations, for a significance level α , if

$$S_E(h) - z_{\frac{\alpha}{2}} \sqrt{\frac{S_E(h)(1 - S_E(h))}{n}} \leq 0,$$

there is at least $\alpha \cdot 100\%$ chance that the production is singular. For the confidence level $\alpha = 0.05$, $z_{\frac{\alpha}{2}} = 1.96$. For example, if the dataset has 100 observations, all the productions which cover not more than three observations are under suspicion of being singular. The same number three goes as the threshold for the datasets with 10000 observations for this confidence level.

For any nonsingular production $h = (H \Rightarrow c)$ we can define support, intensity and base in the datasource:

$$S(h) = P(H), I(h) = P(c | H) = \frac{P(c \& H)}{P(H)}, B(h) = P(c),$$

where $P(H), P(c | H), P(c)$ are probabilities in the distributions Z and Q of the datasource.

Call an increasing sequence of datasets, $\langle E_i \rangle : |E_{i+1}| > |E_i|$, an *experience*.

The next theorem shows that support, intensity and base of a nonsingular production converge on any experience to its datasource support, intensity and base, which is the whole basis of the induction and makes learning possible.

Theorem 4.1: For any nonsingular production h , for any experience $E = \langle E_n \rangle$,

$$\begin{aligned} \lim_{n \rightarrow \infty} (S_{E_n}(h)) &= S(h) \\ \lim_{n \rightarrow \infty} (I_{E_n}(h)) &= I(h) \\ \lim_{n \rightarrow \infty} (B_{E_n}(h)) &= P(c). \end{aligned}$$

Proof. Suppose, $h = (H \Rightarrow c)$ and $P(H) > \alpha > 0$. According with the Law of Large Numbers, the $S_{E_n}(h)$ converges to $S(h)$; the frequency of the concept c in a dataset E , $F(c, E)$, converges to the $P(c)$. The frequency of observations satisfying both H and c , $F(H \& c, E_n)$, converges to $P(H \& c)$. Then,

$$\begin{aligned} \lim_{n \rightarrow \infty} (I_{E_n}(h)) &= \lim_{n \rightarrow \infty} \left(\frac{F(H \& c, E_n)}{S_{E_n}(h)} \right) = \\ &= \frac{\lim_{n \rightarrow \infty} (F(H \& c, E_n))}{\lim_{n \rightarrow \infty} (S_{E_n}(h))} = \frac{P(H \& c)}{P(H)} = I(h) \end{aligned}$$

QED.

Denote $SIB(h)$ ($SIB_E(h)$) the point $\langle S(h), I(h), B(h) \rangle$ ($\langle S_E(h), I_E(h), B_E(h) \rangle$) in the unite cube. The ε -neighborhood of the $SIB(h)$ will be abbreviated as $\Omega_\varepsilon(h)$.

If, in the context R , the acceptability of a production h converges in every experience and has the same limit $\Gamma(h)$, the production h is called **stable** in R .

Since the confirmation function depends on the size of the dataset, the above theorem does not mean yet that each nonsingular production is stable. Our goal is to show that almost every nonsingular production is stable. We start with answering the question of when the acceptability of a production does not converge in an experience.

Theorem 4.2: For a production h , if there exists an experience E , where the acceptability of the production h does not converge, $SIB(h)$ is a point of discontinuity of the confirmation function for all large enough datasets.

Proof. Denote (m) the size of the dataset E_m in an experience E , $X_m = SIB_{E_m}(h)$. Consider the case, when the acceptability of the production h does not converge on the experience E . In this case, it is possible to find two datasets in E with arbitrarily large indices where acceptability of the production h is different:

$$\forall N \exists m > n > N : g_{(m)}(X_m) \neq g_{(n)}(X_n). \quad (1)$$

According to the theorem 4.1, for any ε , for all n larger than some $N(\varepsilon)$, $X_n \in \Omega_\varepsilon(h)$. For any ε , we can take N in (1) such that $(N) > N(\varepsilon)$. In this case, both X_m and X_n belong to the $\Omega_\varepsilon(h)$.

Let us show that in this case, for some large enough fixed n , the function $g_n(X)$ is not constant in $\Omega_\varepsilon(h)$.

If $g_{(m)}(X)$ is constant in $\Omega_\varepsilon(h)$ and equals 0, then $g_{(m)}(X_m) \neq g_{(n)}(X_n)$ means that $g_{(n)}(X_n) = 1$, while $g_{(m)}(X_n) = 0$. Since $m > n$, it is impossible due to monotonicity of the function g over the index n . If $g_{(m)}(X)$ equals 1 in $\Omega_\varepsilon(h)$, due to the monotonicity over index n the confirmation function is equal 1 on all $k > (m)$ in the $\Omega_\varepsilon(h)$. It contradicts (1). Therefore, for any N there exists $m > N$ such that $g_{(m)}(X)$ is not constant in $\Omega_\varepsilon(h)$.

If the function $g_n(X)$ is not constant on the $\Omega_\varepsilon(h)$ for the indices $p, q : p < q$, it is not constant on the same neighborhood for every $k : p < k < q$. Suppose it is not true and $g_k(X)$ is constant in $\Omega_\varepsilon(h)$. If $g_k(X) = 1$ on $\Omega_\varepsilon(h)$, then $g_q(X) = 1$ on $\Omega_\varepsilon(h)$ due to the monotonicity of $g_k(X)$ on k . If $g_k(X) = 0$ on $\Omega_\varepsilon(h)$, then $g_p(X) = 0$ on $\Omega_\varepsilon(h)$. In both cases, it contradicts our assumptions. Together with (1), it means that, for any given ε , the confirmation function $g_n(X)$ has different values in the neighborhood $\Omega_\varepsilon(h)$ for all large enough values of n .

Since $\Omega_\varepsilon(h)$ has the point of discontinuity of $g_n(X)$ for all large n for arbitrary ε , it means that the center of this neighborhood, the point $SIB(h)$, is a point of discontinuity for the confirmation function.

QED.

Prior to answering the question about stability, we need to prove the next lemma.

Lemma 4.1: The set of the points of discontinuity of a confirmation function has volume 0 in the unit cube.

Proof. Take an arbitrary value $Z > 0$ of the third argument of the confirmation function. By the axioms of the validity function, for any n , $g_n(0, 0, Z) = 0$, for large enough n $g_n(1, 1, Z) = 1$. Take any ray $R(\alpha)$ with origin in $(1, 1)$ and angle α of the unit square $z = Z$ in the unite cube. Suppose, T_1, T_2 are two points on the ray $R(\alpha)$, such that the point T_2 is closer to $(1, 1)$ than T_1 . If $g_n(T_1, Z) = 1$, then $g_n(T_2, Z) = 1$ due to monotonicity of the confirmation function by the first two arguments. Then, the ray $R(\alpha)$ may have only one point of discontinuity of the function $g_n(x, y, Z)$. The size of area of the points of discontinuity on the ray $R(\alpha)$ equals length of one point: $\mu(R(\alpha), Z) = 0$. The size of the area of all the points of discontinuity in the unit square can be calculated as an integral by all the angles of rays from 0 to 90^0 :

$$\mu(Z) = \int \mu(X, R(\alpha)) d\alpha = 0.$$

The volume V of the surface in the unit cube, which separates the area $g_n(x, y, z) = 1$, can be calculated as an integral

$$V = \int_0^1 \mu(Z) dZ = 0.$$

QED.

The Lemma does not use the fact that the confirmation function $g_n(x, y, z)$ is monotonous on z .

Theorem 4.3: Justification of Induction. For any nonsingular production h , for any context R , probability that the production h is stable in R equals 1.

Proof. Take a nonsingular production h . The production is not stable, if its acceptability does not converge on some experience, or there are two experiences, where the acceptability have different limits.

Suppose, there exist two experiences $E = \langle E_i \rangle$, $T = \langle T_i \rangle$ such that

$$\lim_n(\gamma(h, E_n)) \neq \lim_n(\gamma(h, T_n)).$$

Then, there exists another experience $G = \langle G_i \rangle : G_{2n} \in E, G_{2n+1} \in T$. The acceptability of the production h does not have any limit on the experience G .

Therefore, we need only to evaluate the probability, that the acceptability of the production does not converge in some experience. By the theorem 4.2, in this case, the point $SIB(h)$ is the point of discontinuity of the confirmation function for all large enough datasets. According to the Lemma, the volume of the points of discontinuity of the validity function equals 0. Then, the probability that the acceptability of a production does not converge in some experience equals 0 as well.

Q.E.D.

V. SOME SPECIAL CASES OF INDUCTION PROBLEMS AND INDUCTION METHODS FOR THEM

A. Plain GrP

Let us consider a special case of the Granular induction problem (GrP). The GrP with an input signature $\Sigma(T, V, F, P, C)$ and a confirmation function $\gamma(h, E)$ will be called *plain*, if

- 1) the input signature does not have constant functions, $F = \emptyset$, and
- 2) the set of constant predicates P has the only type of constant predicates:

$$p_{a,b}(x) \equiv (a \leq x \leq b),$$

where x is an individual variable $x \in V$, $a, b \in \text{dom}(V)$.

- 3) the confirmation function $\gamma(h, E)$ is defined by the thresholds on the support and the intensity of a production:

$$\gamma(h, E) = (S_E(h) > st) \& (I_E(h) > it),$$

where st, it are positive constants.

In the plain problem, a constant predicate $p_{a,b}(x)$ of a nominal variable x have the parameters a, b identical: $a = b$, and $p_{a,b}(x) \equiv (x = a)$. For other types of variables, a and b may be either equal or unequal. Thus, plain GrP allows handling of variables of different types.

Let us notice that, for the purposes of the solution of the plain GrP, it is sufficient to restrict a sort T_i to the values of the variable v_i , which are present in the dataset, $i = 1, \dots, k$. Hence, we can always consider only finite sorts. For every $i = 1, \dots, k$, if $\min T_i, \max T_i$ are the minimal and maximal values of the variable v_i in the dataset, the inequality $\min T_i \leq v_i, \leq \max T_i$ will be called *universal* and it is always true.

A constant formula in the input signature of the plain GrP is a conjunction of double inequalities $a \leq x \leq b$. If a formula does not have a inequality with a variable v_i , it can be equivalently rewritten with addition of the universal inequality by the variable v_i . If a formula has more than one inequality by some variable, it can be equivalently rewritten with not more than one inequality by each variable.

Thus, every constant formula in the input signature of the plain GrP with k variables is equivalent with some *block formula*:

$$f \equiv (a_1 \leq v_1 \leq b_1) \& \dots \& (a_k \leq v_k \leq b_k),$$

where there is one double inequality for each variable, and the inequalities are written in the order of variables.

It is easy to see that no two block formulas are equivalent. Therefore, every constant formula in plain GrP has exactly one equivalent block formula. It enables us to restrict the set of constant formulas under consideration to the block formulas. Also, we can consider only productions which have a block formula as its premise.

A block formula describes an area in the attribute space which can be viewed as a hyper-block, with axes-parallel edges. It can be conveniently presented as a sequence of k pairs of borders of its inequalities: $\langle\langle a_1, b_1 \rangle\rangle, \dots, \langle\langle a_k, b_k \rangle\rangle$.

B. Solution of the Plain GrP

Block formulas can be partially ordered by inclusion of the hyper-blocks they describe. If

$$\begin{aligned} f_1 &\equiv \langle\langle a_1, b_1 \rangle\rangle, \dots, \langle\langle a_k, b_k \rangle\rangle, \\ f_2 &\equiv \langle\langle c_1, d_1 \rangle\rangle, \dots, \langle\langle c_k, d_k \rangle\rangle \end{aligned}$$

are two block formulas, $f_1 \leq f_2$ iff

$$(a_1 \leq c_1) \& (b_1 \geq d_1) \& \dots \& (a_k \leq c_k) \& (b_k \geq d_k).$$

In another words, for two block formulas f_1, f_2 , $f_1 \leq f_2$, iff the hyper-block described by the formula f_2 is included in the hyper-block described by the formula f_1 .

To organize the search of the premises of goal productions, we will need a linear order on the set block formulas, which is in an agreement with this partial order. First, we order atomic formulas of the same variable:

$$(a \leq v \leq b) \preceq (c \leq v \leq d),$$

iff

$$(a \leq c) \bigvee ((a = c) \& (b \geq d)).$$

It is easy to see that for two atomic formulas of the same variable f_1, f_2 , either $f_1 \preceq f_2$, or $f_2 \preceq f_1$.

For the block formulas $f_1 = (t_1 \& \dots \& t_k)$, $f_2 = (g_1 \& \dots \& g_k)$, where t_m, g_m are atomic formulas of the variable $v_m, m = 1, \dots, k$, $f_1 \prec f_2$ iff there exists an index $i : 1 \leq i \leq k$,

$$(t_1 = g_1) \& \dots \& (t_{i-1} = g_{i-1})$$

and

$$t_i \prec g_i.$$

Again, it is easy to see that for two block formulas f_1, f_2 either $f_1 \prec f_2$ or $f_2 \prec f_1$.

For a block formula f , f' denotes the block formula next in order \prec . Since we consider only finite sorts in this problem, there are only finite number of block formulas, and such an operation is uniquely defined for every block formula f , except for the last block formula in the order. Denote \bar{f} the closest block formula next to f in the order \prec such that $\neg(\bar{f} \geq f)$.

In [5], we proposed an algorithm of a plain GrP. We will present it here in a different view. The algorithm searches for all premises of the goal productions of a certain concept c . It looks over all block formulas in the order \prec , starting from the formula of the maximal hyper-block. In this search, we avoid testing acceptability of the block formulas, which are greater (\geq) than the found premises of goal production, since these formulas can not be premises of the goal productions themselves. Also, we avoid testing formulas which can not provide enough support, based on the previous testing of comparable formulas. It improves greatly the speed of the search.

The Block algorithm

The algorithm uses for storage of block formulas two sets: M, N . The set M is used to store the premises of the goal productions, the set N is used for storage of the block formulas which do not provide enough support. From the beginning, these sets are empty. The formula H is minimal in the order \prec , it is a conjunction of the universal inequalities by all variables.

Step 1. If there exists $Z : Z \in M \cup N$, such that $Z \leq H$, then $H := \bar{H}$. Return to the step 1. Otherwise, go to the step 2.

Step 2. If the support of the production ($H \Rightarrow c$) is too small, then $N := N \cup H, H := \bar{H}$, return to the step 1. Otherwise go to the step 3.

Step 3. If the production $H \Rightarrow c$ is acceptable, then $M := M \cup H, H := \bar{H}$, return to the Step 1. Otherwise, go to the Step 4.

Step 4. $H := H'$, return to the Step 1.

The algorithm stops, when the required operation H' or \bar{H} is impossible.

When the algorithm stops, M is a set of antecedents of all the hypotheses with the concept c .

The algorithm uses monotonicity of the validity function over support of the productions to skip testing lots of productions when it jumps from a current block H to the \bar{H} . The algorithm uses the the sets M, N to avoid testing any less general production than stored there. In the first case, the production will be less general than some hypothesis, in the second case, its support will be less than the low limit s for a valid production.

Still, experiments show that the average time of the algorithm is proportional to $m_1 \times \dots \times m_k$, where m_i is the number of different values of the i -th variable in the dataset. Hence, one of the effective ways to accelerate the algorithm is to group some values of the continuous variables.

Another way to improve performance of the algorithm on large datasets is to minimize redundancy of the attributes. Our analysis shows, that attributes are strongly correlated in many medical research problems. In [6], we proposed a fuzzy clustering algorithm Heavy Centers which can be used to find a maximal subset of uncorrelated variables.

In the same work, we introduced an aggregation level procedure to build decision rules with the hypotheses, obtained on the granulation level of induction. The combination of these algorithms were applied on the Wisconsin Breast Cancer dataset with 9 discrete attributes. The resulting decision rules were simpler and more accurate, comparing with other methods.

VI. CONCLUSION

We introduced logic formalization of induction learning in the framework of the soft computing. The formalization includes two “players”: the learner and nature. Nature is represented by statistical laws, which determine probabilities of the observations. The learner can be characterized by his language and by his “confirmation function”: the conditions on the statistical characteristics of a dependence in data, which make the learner to believe it. Implementing the idea of granulation, we proposed to separate learning on two stages: granulation and aggregation. The purpose of the granulation process is to find maximal homogeneous granules in the attribute space, which are described by the most general acceptable hypotheses. The aggregation level of induction builds the decision rules using combinations of granules. We introduced distinct languages for two stages of learning.

The confirmation function is defined here by some reasonable properties, rather than by an explicit formula. Based on these properties, we were able to justify the “leap of faith” of induction, which allows a learner to make general conclusions based on a finite set of observations. More exactly, we were able to demonstrate that, almost certainly, any conclusion is always acceptable or unacceptable on large enough datasets. This confirms productivity of our formalization as a theory.

For the practical application of the IL, we described a particular type of the granulation level problem and proposed a method for its solution. Since the problem and its solution have clear value for the automatic learning, this method opens possibilities of effective use of the IL approach in medical applications.

REFERENCES

- [1] L. Zadeh. Some reflections on Information Granulation. In: Data mining, rough sets and granular computing. T.Y. Lin, Y.Y. Yao, L.Zadeh, eds. Springer-Verlag, 2002.
- [2] C. Glowinski, R. Michalski. Discovering multihead attributional rules in large databases. Tenth International Symposium on Intelligent Information Systems, Zakopane, Poland, June, 2001. <http://www.mli.gmu.edu/papers/01-5.pdf>
- [3] V. Sperschneider, G. Antoniou. A logic foundation for computer science. Addison-Wesley, 1991.
- [4] F. Divina, E. Marchiori Evolutionary Concept Learning. In Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, pp. 343-350. New York, NY, USA 2002

- [5] Sapir M. A method of constructing plausible hypotheses for attributes of various types. Automatics and Remote Control, N11, 1993. (in Russian)
- [6] Sapir M, Sherman S. A Toolkit for the Search of the Most General Interpretable Hypotheses. International Conference on Integration of Knowledge Intensive Multi-Agent Systems. KIMAS'03: 318 - 323.
- [7] Valiant L. A theory of learnable. Communications of ACM, 27 (11) pp 1134-1142, 1984.
- [8] Auer, P., R.C. Holte, and W. Maass (1995), "Theory and Applications of Agnostic PAC-Learning with Small Decision Trees," Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, CA, pp. 21–29 (Morgan Kaufmann, San Francisco, CA)
- [9] Ballantine, W.G. "Inductive logic". Ginn & Company. Boston and London, 1896.
- [10] Triantaphylow, E., Kovalerchuk, B., Deshpande A.S. Some recent developments of using logical analysis for inferring a Boolean function with few clauses. Interfaces in Computer Science and Operation Research. Barr, E.S., Helgason, R.V., Kennington, J.L., eds. Kluwer. pp 217 - 240, 1997.
- [11] D. Wolpert. The status of supervised learning science circa 1994: the search for consensus //The mathematics of generalization. Ed. D. Wolpert