Universal Learning of Classes From Sparse and Non-uniform Evidence*

C. Ferri-Ramírez J. Hernández-Orallo M.J. Ramírez-Quintana

DSIC, UPV, Camino de Vera s/n, 46020 Valencia, Spain.
{cferri,jorallo,mramirez}@dsic.upv.es

Abstract. We present a framework for the Inductive Functional Logic Programming paradigm from positive data and from non-uniform classes. It extends the Bayesian approach in [18] to consider classes in the definition of hypothesis generality, now called *class unevenness*, which is used for the evaluation of hypotheses. This new measure takes into account how the distributions of classes of the hypothesis and the target theory are. The target theory class distribution is approximated by using the class distribution of the evidence and the initial distribution. Any problem with different classes with any proportion between 0% and 100% can be addressed by this new measure. As a result, negative examples are no longer necessary for learning Boolean functions since they should be considered as the positive examples which define a second $false$ class. Finally, we discuss an implementation of sample generators for a given or unknown probability distribution of terms. The usefulness of the generators is shown for approximating the value of unevenness of a hypothesis.
Keywords: Machine Learning, Inductive Logic Programming (ILP), Inductive Functional Logic Programming (IFLP), Classification Problems, Hypothesis Evaluation.

1 Introduction

The Inductive Inference framework in general and Inductive Logic Programming (ILP) in particular usually deal with the learning of a target concept from positive and negative evidence. Nevertheless, learning from only positive data is desirable for several applications. These applications have many positive examples available while few or no negative examples are given (such as the analysis of DNA and RNA sequences [10], data mining applications, software engineering, grammatical inference [17] and natural language, to cite some of them). The main difficulty of inferring from only positive examples is that the learning mechanism has to avoid overgeneralisation. This is due to the fact that the simplest and most general possible hypothesis for a concept p, i.e. $\forall X\ p(X)$, will always be consistent. However, is it neither an acceptable hypothesis in most of the applications [17] nor can it be specialised unless negative evidence is provided. Gold [6] showed that even the class of regular languages is not inferable

* This work has been partially supported by CICYT under grant TIC 98-0445-C03-C1.

from positive data, and, thus, neither are (the whole class of) Prolog programs. A lot of work has been done to discover restricted hypothesis languages which are inferable in this way. In particular, ILP has identified constrained hypothesis languages for which learning from positive data is possible ([20],[1], [14],[19]). Another approach known as U-learnability, which is independent from the hypothesis representation, has been presented by Muggleton in [17, 18]. It is based on the development of a Bayes function for maximising posterior probability. Under this Bayes framework, a trade-off between size and generality of a hypothesis is defined which allows for estimating the probability of a hypothesis.

We present a framework for the Inductive Functional Logic Programming (IFLP) paradigm [8, 9] from positive data and from non-uniform classes following this proposal. As in ILP, the induction of some IFLP programs from only positive evidence presents the same problem of overgeneralisation. The following example demonstrates this fact.

Example 1. Consider the following evidence $e_1 - e_{10}$:

$$E = \left\{ \begin{array}{llll} e_1: & e(4) = true, & e_2: & e(12) = true, \\ e_3: & e(3) = false, & e_4: & e(2) = true, \\ e_5: & e(7) = false, & e_6: & e(7) = false, \\ e_7: & e(20) = true, & e_8: & e(0) = true, \\ e_9: & e(3) = true, & e_{10}: & e(2) = false \end{array} \right\}$$

where natural numbers are represented by using the functor s as the symbol for succesor, e.g. $s(s(s(0)))$ means 3.

If only positive examples are given, the most general theory $e(X) = true$ would be optimal and would cover the evidence. As has been shown for ILP, this problem can be avoided by weighing the generality of the program. Nevertheless, it is important to note that this problem only occurs in for Boolean problems. Consider the case of learning the function of addition ($sum/2$). In this case, no negative evidence is strictly required, since programs must be confluent and terminating. $sum(X, Y) = Z$ (the most general program) is not confluent nor terminating.

In functional programming [2] (and in functional logic programming, FLP [7]), the evaluation of a functional term w.r.t. a program P is the computation of its normal form[1]. The set of all possible different normal forms represents the meaning of the program. In fact, the ADT defined by the theory P consists of normal forms. Moreover, from this algebraic semantic point of view, they represent the elements or congruence classes of the initial algebra $\mathcal{T}(\Sigma)/\equiv_P$ [4]. For this reason, we will refer to them as classes throughout the paper. This situation is different from that in ILP since the meaning of a logic program is the set of ground atoms that are proven *true*, i.e. that belong to its Herbrand minimal model. Hence, using the notation of functional programming, we can say that *true* is the unique class defined by a logic program. In this paper, we extend the Bayesian approach in [18] in order to take classes into consideration within the

[1] The normal form of a term is unique if the program is confluent and terminating.

definition of the generality of a hypothesis. For the learning of Boolean functions, if we work with more than one class, negative examples are no longer necessary since they should be considered as the positive examples which define the $false$ class. From this point of view, learning a concept from positive examples in ILP can be seen as the problem of learning a *partially* defined concept (since the evidence only contains examples for the $true$ class). Learning the same Boolean target concept in IFLP can be seen as the problem of learning a *totally* defined concept if positive evidence also contains examples for the $false$ class, as we will show in this paper. This makes the problem more general. Now, it is not a question of positive-only and positive and negative examples, but on what the distribution of classes is. We want a measure that considers any case between 0% and 100% for every class. If no correction is made, for a general problem (not necessarily Boolean), examples for all classes must appear in the evidence. However, in the case of learning from sparse and non-uniform evidence, a good idea could be to preserve the proportion of examples of each kind according to a prior probability distribution if an adequate evidence is generated.

On the other hand, our approach is also useful for classification problems in the case of more than two classes[2]. Firstly, these problems can be better and more naturally formulated as functional programs which define several classes. Secondly, the programs that solve problems of this kind are suitable to be learned from only positive examples due to the fact that negative evidence does not seem to actually be neccessary, as we will show in Section 6. Of course, when there are no examples in the evidence for one class, the learning task cannot be made from only positive data even if a correction is not included. This correction is a generalisation for more than one class of the Muggleton generalisation degree, now called *class unevenness*. Once its theoretical properties are shown, this value must be approximated by the use of an evidence generator. This generator also allows the practical application of the new evaluation measure in the system FLIP[3] [5].

The paper is organised as follows. Section 2 includes general notation and terminology. In Section 3, we review the Bayesian formulation of [18]. Our approach for evaluating the unevenness of a hypothesis in IFLP is defined in Section 4. Also, in Section 5, we show that Boolean problems are transferable to two classes in this framework. Classification problems of more than two classes are analysed in Section 6. In Section 7, we discuss an implementation defining a generator of samples given a certain probability distribution of terms which is used for approximating the value of unevenness of a hypothesis. Some experimental results are also given. Finally, Section 8 concludes the paper.

[2] Classification problems which define two classes can be considered as Boolean problems.

[3] The FLIP system is a learner of functional logic programs which is based on narrowing (the best known operational semantic for FLP).

2 Preliminaries

Let $\mathcal{V}$ be a countably infinite set of variables and let Σ be a set of *function symbols* (or functors) together with their arity, where f/n denotes the function symbol f of arity n. Then $\mathcal{T}(\Sigma, \mathcal{V})$ denotes the set of *terms* built from Σ and $\mathcal{V}$, and $\mathcal{T}(\Sigma)$ denotes the set of *ground terms* built from Σ. We consider that the signature Σ is partitioned as $\Sigma = \mathcal{C} \biguplus \mathcal{F}$, where $\mathcal{C}$ is the set of irreducible symbols and $\mathcal{F}$ is the set of defined function symbols. The length of a term t, $l(t)$, is defined as

$$l(t) = \left\{ \begin{array}{ll} 1 & \text{if } t \text{ is a constant or a variable} \\ 1 + \sum_{i=1}^{n} t_i & \text{if } t \text{ is of the form } f(t_1, \ldots, t_n) \end{array} \right\}$$

An equation is an expression of the form $l = r$ where l and r are terms. l is called the left hand side (lhs) of the equation and r is the right hand side (rhs). By $card(A)$ we denote the cardinality of a set A. Given a set of equations A and given a class c, the cardinality of A over c is defined as $card_c(A) = card(\{l = c \in A\})$. An equational theory $\mathcal{E}$ (which we call *program*) is a finite set of equational clauses of the form $l = r \Leftarrow e_1, \ldots, e_n.$ with $n \geq 0$ where e_i is an equation, $1 \leq i \leq n$. The theory (and the clauses) are called *conditional* if $n > 0$ and *unconditional* if $n = 0$. An equational theory can also be viewed as a (Conditional) Term Rewriting System (CTRS) since the equation in the head is implicitely oriented from left to right and the literals e_i in the body are ordinary non-oriented equations. Given a (C)TRS $\mathcal{R}$, $t \rightarrow_{\mathcal{R}} s$ is a rewrite step if there exists an ocurrence u of t, a rule $l = r \in \mathcal{R}$ and a substitution θ with $t_{|u} = \theta(l)$ and $s = t[\theta(r)]_u$. A term t is said to be in *normal form* w.r.t. $\mathcal{R}$ if there is no term t' with $t \rightarrow_{\mathcal{R}} t'$. $\mathcal{R}$ is said to be *canonical* if the binary one-step rewriting relation $\rightarrow_{\mathcal{R}}$ is terminating (there is no infinite chain $s_1 \rightarrow_{\mathcal{R}} s_2 \rightarrow_{\mathcal{R}} s_3 \rightarrow_{\mathcal{R}} \ldots$) and confluent ($\forall\, s_1, s_2, s_3 \in \mathcal{T}(\Sigma, \mathcal{X})$ such that $s_1 \rightarrow^*_{\mathcal{R}} s_2$ and $s_1 \rightarrow^*_{\mathcal{R}} s_3, \exists\, s \in \mathcal{T}(\Sigma, \mathcal{X})$ such that $s_2 \rightarrow^*_{\mathcal{R}} s$ and $s_3 \rightarrow^*_{\mathcal{R}} s$). Narrowing is a sound and complete method for solving equations w.r.t. canonical programs. Given a program P, a term t *narrows* into a term t' (in symbols $t \overset{u, l=r, \theta}{\hookrightarrow}_P t'$[4]) iff $u \in \bar{O}(t)$, $l = r$ is a new variant of a rule from P, $\theta = mgu(t_{|u}, l)$ and $t' = \theta(t[r]_u)$. We write $t \hookrightarrow^n_P t'$ if t narrows into t' in n narrowing steps.

The Inductive Functional Logic Programming, IFLP, has been defined as the functional extension to ILP [15]. The goal is the inference of a functional logic program P from a set of positive and optionally negative equations E using a background knowledge theory B (another functional logic program). The evidence is composed of positive E^+ and negative E^- equations such that their lhs are ground terms of the form $f(t_1, \ldots, t_n)$ where $f \in \mathcal{F}$ and $t_i \in \mathcal{T}(\mathcal{C})$ and their rhs are normalized w.r.t. the background theory B and the theory P which is meant to be discovered (hypothesis), with $B \cup P$ being canonical.

[4] Or simply $t \overset{l=r, \theta}{\hookrightarrow}_P t'$ or $t \overset{\theta}{\hookrightarrow}_P t'$ if the occurrence or the rule is clear from the context. Also, the subscript P will usually be dropped when clear from the context.

We use the following notation to denote a vector $\overrightarrow{v}$ of n components: $[v_i]_{i=1..n}$. Given a vector $\overrightarrow{v}$, its *normalisation* is

$$\overrightarrow{v}^{norm} = \frac{1}{\sum_{i=1}^{n} v_i} \cdot \overrightarrow{v}$$

Obviously, after normalisation, $\sum_{i=1}^{n} v_i = 1$. Two vectors $\overrightarrow{v}$ and $\overrightarrow{w}$ of n components form an angle between them, $\widehat{\overrightarrow{v}\,\overrightarrow{w}}$, which is defined as

$$\widehat{\overrightarrow{v}\,\overrightarrow{w}} = \arccos\left(\frac{\overrightarrow{v} \bullet \overrightarrow{w}}{|\overrightarrow{v}||\overrightarrow{w}|}\right)$$

where $\bullet$ denotes the scalar product, that is $\overrightarrow{v} \bullet \overrightarrow{w} = v_1 w_1 + \ldots + v_n w_n$, and $|\overrightarrow{a}|$ denotes $\sqrt{a_1^2 + \ldots + a_n^2}$. With $\overrightarrow{v}/\overrightarrow{w}$ we denote $(v_1/w_1, v_2/w_2, \ldots, v_n/w_n)$.

3 The Bayesian framework

Following the U-learnability framework defined in [16], Muggleton [18] derived a Bayes' function for maximising posterior probability of hypotheses when learning from positive data. We briefly review that formulation in this Section.

X is taken to be a countable class of instances and $\mathcal{H} \subseteq 2^X$ to be a countable class of concepts. Let D_X and $D_{\mathcal{H}}$ be probability distributions over X and $\mathcal{H}$ respectively. For $H \in \mathcal{H}$, $D_X(H)$ is defined as $\Sigma_{x \in H} D_X(x)$. Size and generality of a hypothesis H are defined as follows:

$$sz(H) = \text{-ln } D_{\mathcal{H}}(H) \quad (1)$$
$$g(H) \;\; = D_X(H) \qquad (2)$$

where formula (1) is justified by Occam's Razor [12]. The Bayes theorem allows us to relate size and generality. Let $p(H \mid E)$ be interpreted as the probability that the hypothesis chosen by the learner, H, would be the target theory T given the example sequence E. Hence,

$$p(H \mid E) \quad = D_{\mathcal{H}}(H) \left(\frac{1}{D_X(H)}\right)^m c_m \qquad (3)$$

$$\ln\, p(H \mid E) = m \ln \left(\frac{1}{g(H)}\right) - sz(H) + d_m \quad (4)$$

where $d_m = \ln c_m$ with c_m being a normalising constant. Formula (4) shows that the probability that a hypothesis H would be the target theory decreases when either its size and generality increase.

As we have already mentioned in Section 1, in the above rationale it is assumed that all the evidence defines one value or class: the *true* value.

In what follows, we extend this formulation to take classes into consideration. In our proposal for IFLP, X is the set $\mathcal{T}(\mathcal{F},\mathcal{C})$ whereas the evidence E is composed by equations of the form $l = r$ such that $l \in X$ and r identifies a class.

4 A Framework for the Evaluation of Hypotheses in IFLP

In this section, we define our method for the evaluation of hypotheses. The basic idea is to define the generality of a hypothesis (the above formula (2)) w.r.t. all the classes defined by the target theory (i.e. by the problem) and the relation between a probability distribution over H and T.

In what follows, the (possibly infinite) collection of classes C determined by a target program T is defined as $C = \bigcup_{i=1}^{n} c_i$, where c_i denotes each one of the n classes. Also, X_{c_i} denotes the instances of X which belong to the c_i class according to T. Note that X can be infinite. When it is the case, we will assume that X is approximated by a finite subset which is large enough to be meaningful. In Section 7 we will introduce a procedure to generate finite approximations to both X and C.

The following definition shows that the probability distribution over a collection C of n classes according to a hypothesis H is a vector such that each one of its components represents the probability of each class in H.

Definition 1. *Given a hypothesis H and given a collection of classes C, the probability distribution over C for H, $\overrightarrow{D_C}(H)$, is:*

$$\overrightarrow{D_C}(H) = \left[\frac{card(X_{c_i}(H))}{card(X_C(H))}\right]_{i=1..n}$$

where

$$X_{c_i}(H) = \{x \in X \mid H \models x = c_i\} \text{ and } X_C(H) = \sum_{i=1..n} X_{c_i}(H)$$

Example 2. For instance, if H defines the function $mod3/1$ (which computes $true$ if its argument is divisible by 3 and $false$ otherwise), then $\overrightarrow{D_C}(H) = (\frac{1}{3}, \frac{2}{3})$.

Definition 2. *Given a hypothesis H, the probability distribution of H w.r.t. X and C, $\overrightarrow{D_{C,X}}(H)$ is:*

$$\overrightarrow{D_{C,X}}(H) = \left[\sum_{x \in X_{c_i}(H)} D_X(x)\right]_{i=1..n}$$

Example 3. Consider again the target theory $T = mod3$ and suppose that the terms from X follow the distribution $D_X(x) = 2^{-l(x)}$, then:

$$\overrightarrow{D_{C,X}}(T) = \left[\begin{array}{cc} \sum_{\substack{i=0..n \\ mod3(s^i(0)) =_T \\ true}} 2^{-l(s^i(0))} & , \sum_{\substack{i=0..n \\ mod3(s^i(0)) =_T \\ false}} 2^{-l(s^i(0))} \end{array}\right] \simeq [0.6, 0.4]$$

where $s^i(0)$ denotes the natural number i.

Our aim is to estimate whether a hypothesis H is close or even to the target theory T. The goodness of H can be obtained by analysing what the probability of each class of C in H is and what the probability in T is. Since these probabilities are components of two vectors, we can compare H with T by means of the angle that its corresponding vectors form in the space. In a certain way, this angle is a measure of the closeness of $\overrightarrow{D_C}(T)$ with $\overrightarrow{D_C}(H)$[5]. This difference is called *unevenness* and is defined as follows:

Definition 3. *Class Unevenness*
Given a hypothesis H and a collection C of n classes, the class unevenness of H is

$$u(H) = \frac{\tan\alpha}{1+\tan\alpha} \cdot \sum_{i=1..n} (D_C(H))_i$$

where α represents the angle $\widehat{\overrightarrow{D_C}(H)\overrightarrow{D_C}(T)}$.

The above definition shows that the smaller the angle is, the better the hypothesis is because unevenness is lower.

Example 4. Consider $H_1 = mod2$ and $H_2 = mod3$. We have $D_C(H_1) = [0.5, 0.5]$ and $D_C(H_2) \simeq [0.33, 0.67]$. Consider a T such that $D_C(T) = \left[\frac{1}{3}, \frac{2}{3}\right]$. Comparing the two hypotheses with T:

$$\alpha_1 = \arccos(\frac{0.5}{0.5270}) = 0.3217 \quad \alpha_2 = 0$$

$$u(H_1) = 0.3332 \quad u(H_2) = 0$$

As we could expect, the correct hypothesis obtains a 0 *rad* angle.

Although from a theoretical point of view, the probability distribution over a collection of classes has been defined considering the set of terms X, in practice, the learning task is done from a partial evidence E. This also means that we must take into consideration that E can contain one example $l = r$ more than once since this evidence is generated according to the probability distribution $D_X(l)$. Note that, on the contrary, X does not contain any term twice.

Definition 4. *The probability distribution function over an evidence E, $\overrightarrow{D_E}$, can be computed as:*

$$\overrightarrow{D_E} = \left[\sum_{x \in E} \frac{card_{c_i}(E)}{card(E)} \right]_{i=1..n}$$

[5] This has usually been done by a statistical χ^2 test. This test cannot be used here, because, for it to be useful, the absolute frequency of each class must be ≥ 5 and the initial distribution must be closer to the uniform distribution. There are other methods to estimate the discrepancy such as some corrections of the χ^2 test. However, these can only be applied if all the expected frequencies are greater than 0. Example 6, for instance, shows a case where this test cannot be used.

The following results show that the probability distribution over classes w.r.t. an evidence E with infinite examples converges to the theoretical one (which considers X instead of E).

Lemma 1. *Given an evidence E such that lhs of equations of E are generated following a distribution D_X, then*

$$\lim_{card(E)\to\infty} \overrightarrow{D_E} = \overrightarrow{D_{C,X}}(T)$$

The following result is established when $\overrightarrow{D_X}$ is the uniform distribution:

Corollary 1. *If $\overrightarrow{D_X}$ is the uniform distribution, then*

$$\lim_{card(E)\to\infty} \overrightarrow{D_E} = \overrightarrow{D_C}(T)$$

Proof. Trivial, since if $\overrightarrow{D_X}$ is the uniform distribution then

$$\forall i \in \{1..n\}\ \forall x \in X_{c_i}(T) : D_X(x) = \frac{1}{card(X_C(T))}$$

and by Definition 2

$$(D_{C,X}(T))_i = \sum_{x \in X_{c_i}(T)} D_X(x) = \sum_{x \in X_{c_i}(T)} \frac{1}{card(X_C(T))} =$$

$$= \frac{card(X_{c_i}(T))}{card(X_C(T))} = (D_C(T))_i$$

Therefore, the corollary holds by Lemma 1. □

Example 4 illustrates how we can choose among a set of hypotheses by comparing them with the target theory. However, this is not a very real situation because $\overrightarrow{D_C}(T)$ is usually not known, and without this distribution it is impossible to know the angle between a hypothesis and T. We solve this problem by introducing a method that computes an approximation of $\overrightarrow{D_C}(T)$. Our approximation, which we denote as $\widetilde{\overrightarrow{D_C}}(T)$, is expressed by the following definitions:

Definition 5. *Given an evidence E and a collection of classes C, $\overrightarrow{\mu}(E)$ is:*

$$\overrightarrow{\mu}(E) = \left[\sum_{x \in E} D_X(lhs(x)) \cdot \frac{card(E)}{card_{c_i}(E)} \right]_{i=1..n}$$

Definition 6. *The distribution $\overrightarrow{D_C}(T)$ is approximated from an evidence E and a distribution D_X by:*

$$\widetilde{\overrightarrow{D_C}}(T) = \frac{\overrightarrow{D_E}}{\overrightarrow{\mu}(E)}$$

Example 5. Let foo be a theory such that:

$$foo(x) = \begin{cases} if\ 1 \leq x \leq 33 \quad then\ 1 \\ if\ 34 \leq x \leq 99\ then\ 0 \end{cases}$$

then $\overrightarrow{D_C}(T) = (\frac{1}{3}, \frac{2}{3})$. And consider

$$\overrightarrow{D_x} = \begin{cases} 1 \leq x \leq 33 \quad \rightarrow 0.0203 \\ 34 \leq x \leq 99 \rightarrow 0.005 \end{cases}$$

If we take an evidence E_1 formed by 10 examples, 5 from each class, then we have:

$$\overrightarrow{D_{E_1}} = (0.5, 0.5) \quad \overrightarrow{\mu}(E_1) = (0.066, 1.939) \quad \widetilde{\overrightarrow{D_C}}(T)^{norm} = (0.967, 0.033)$$

If we take a second evidence, E_2, built ideally from D_X, we would get 1000 examples where 33 are from the first class, and 967 are from the second class.

$$\overrightarrow{D_{E_2}} = (0.033, 0.967) \quad \overrightarrow{\mu}(E_2) = (1, 14.665) \quad \widetilde{\overrightarrow{D_C}}(T)^{norm} = (0.33, 0.67)$$

As we see, as long as the evidence is generated according to D_X and T, $\widetilde{\overrightarrow{D_C}}(T)^{norm}$ is close to $\overrightarrow{D_C}(T)$.

This last example shows how important a good evidence is, when $\overrightarrow{D_C}(T)$ is not given. Therefore, the approximation to $\overrightarrow{D_C}(T)$ greatly depends on the quality and quantity of the evidence. If the evidence goes to infinite, then both the theoretical value and its approximation agree, as the following theorem shows.

Theorem 1. *If an evidence E is generated following the distribution D_X and a theory T then*

$$\lim_{card(E)\to\infty} \widetilde{\overrightarrow{D_C}}(T) = \overrightarrow{D_C}(T)$$

Proof. By Definition 6, we have that $\widetilde{\overrightarrow{D_C}}(T) = \frac{\overrightarrow{D_E}}{\overrightarrow{\mu}(E)}$. Hence,

$$\lim_{card(E)\to\infty} \widetilde{\overrightarrow{D_C}}(T) = \lim_{card(E)\to\infty} \frac{\overrightarrow{D_E}}{\overrightarrow{\mu}(E)} = \frac{\lim_{card(E)\to\infty} \overrightarrow{D_E}}{\lim_{card(E)\to\infty} \overrightarrow{\mu}(E)}$$

Now, by Lemma 1, $\lim_{card(E)\to\infty} \overrightarrow{D_E} = \overrightarrow{D_{C,X}}(T)$. Therefore,

$$\frac{\lim_{card(E)\to\infty} \overrightarrow{D_E}}{\lim_{card(E)\to\infty} \overrightarrow{\mu}(E)} = \frac{\overrightarrow{D_{C,X}}(T)}{\lim_{card(E)\to\infty} \overrightarrow{\mu}(E)}$$

and by Definitions 2 and 5

$$\frac{\overrightarrow{D_{C,X}}(T)}{\lim_{card(E)\to\infty} \overrightarrow{\mu}(E)} = \frac{\left[\sum_{x\in X_{c_i}(H)} D_X(x)\right]_{i=1..n}}{\lim_{card(E)\to\infty} \left[\sum_{x\in E} D_X(x) \frac{card(E)}{card_{c_i}(E)}\right]_{i=1..n}}$$

On the other hand, it holds that

$$\lim_{card(E)\to\infty} card(E) = K \cdot card(X_C(T))$$

$$\lim_{card(E)\to\infty} card_{c_i}(E) = K \cdot card(X_{c_i}(T))$$

where K is a constant. Finally,

$$\frac{\left[\sum_{x\in X_{c_i}(H)} D_X(x)\right]_{i=1..n}}{\lim_{card(E)\to\infty}\left[\sum_{x\in E} D_X(x)\frac{card(E)}{card_{c_i}(E)}\right]_{i=1..n}} =$$

$$\frac{\left[\sum_{x\in X_{c_i}(H)} D_X(x)\right]_{i=1..n}}{\left[\sum_{x\in X_{c_i}(T)} D_X(x)\frac{\cancel{K}\cdot card(X_C(T))}{\cancel{K}\cdot card(X_{c_i}(T))}\right]_{i=1..n}} = \overrightarrow{D_C}(T)$$

□

5 The Function $u(H)$ as a Generalisation of $g(H)$

We have introduced the previous measures in order to generalise the results of learning from only positive data to any proportion of instances of each of the classes. Now we will show that this extension is coherent with the results which are obtained for one class. We will also establish the equivalence between Boolean problems with one class and the same problem presented with two classes (*false* and *true*). In most problems of learning from only positive data, the proportion of possible constructable positive examples over negative examples is small. For instance, in natural languages, there are many more combinations of characters that mean nothing (or are incorrect) than there are well-formed ones. Moreover, as any problem from only positive data, the general (everything is valid) hypothesis is suggested by the data and one must 'correct' the hypothesis towards more specific ones. If we consider two classes (positive and negative) it holds that the 'ideal' vector to consider is $\overrightarrow{D_C}(T) = (0,1)$, for the previous two reasons. This is finally what is implicit in many works of learning from positive data only [18].

Theorem 2. *Given a learning problem A with only one class c_1 and a partial hypothesis H_a and a second problem B with two classes c_1 and c_2 where there exists a total hypothesis H_b such that:*

$$\forall x \in X \begin{cases} \textit{if} \quad x =_{H_a} c_1 \textit{ then} & x =_{H_b} c_1 \\ & \textit{otherwise } x =_{H_b} c_2 \end{cases}$$

If we consider the optimal vector for positive evidence $\overrightarrow{D_C}(T) = (0,1)$, i.e., there are infinitely more negative cases than positive ones, then

$$g(H_a) = u(H_b)$$

and this holds for any distribution of examples of the hypothesis.

Proof. Consider a hypothesis H_a with a distribution of examples such that $D_E = (1/a, (a-1)/a)$ where $1/a$ corresponds to class c_1. Consequently, $g(H_a) = 1/a$. On the other hand, from the definition of $u(H)$ we have:

$$u(H_b) = \frac{tan\alpha}{1 + tan\alpha} \sum_{i=0..2} (D_C(H_b))_i$$

which is equal to:

$$u(H_b) = \frac{tan\alpha}{1 + tan\alpha}$$

since H_b is total. It is easy to show that, from the angle between $\overrightarrow{D_C}(T) = (0,1)$ and $D_E = (1/a, (a-1)/a)$, it follows that $tan\alpha = 1/(a-1)$. From here,

$$u(H_b) = \frac{\frac{1}{a-1}}{1 + \frac{1}{a-1}} = \frac{1}{a} = g(H_a)$$

□

It is important to note that the previous theorem shows the equivalence between considering the problem with two classes and a total function, and considering the problem with one class and a partial function in the traditional NAF (Negation As Failure) sense.

Example 6. Consider a complete hypothesis with $\overrightarrow{D_C}(H) = (1/3, 2/3)$. The angle with $\overrightarrow{D_C}(T) = (0,1)$ is $\arccos\frac{2/3}{\sqrt{5/9}} = \arccos 2/\sqrt{5} = 0.4636$ rad. From here $u(H) = \frac{tan0.4636}{1+tan0.4636} = \frac{0.5}{1.5} = 0.3333$ which matches with the corresponding one-class partial hypothesis $g(H_a) = 1/3$.

However, function $u(H)$ is not equal to $g(H)$ for one class, since for one class, angles are always 0, and $u(H)$ will always be 0. Consequently, if one is going to consider partial hypothesis, it is better to add a "default class" to which unclassified examples are going to be assigned.

Corollary 2. *Consider a hypothesis H for n classes where only 2 of them (c_i and c_j) follow that $D_i(H) + D_j(H) = 1$, and the expected vector $\overrightarrow{D_C}(T)$ is of the form $(0, 0, \cdots, 0, 1, 0, 0, \cdots, 0)$ where $D_j = 1$. If we construct a hypothesis H' such that:*

$$\forall x \in X \begin{cases} if \quad x =_H c_i \ then \ x =_{H'} c_i \\ if \quad x =_H c_j \ then \ x =_{H'} c_j \end{cases}$$

then $u(H) = u(H')$.

The previous corollary affirms that empty classes can be added without affecting the evaluation measure.

The advantage is that we can now consider any distribution of positive and negative examples, either in the initial distribution D_X which generates the examples or in the expected distribution of classes of the theory which has assigned classes to the examples.

6 General Classification Problems and Negative Evidence

In the case of classification problems with more than two classes, our approach extends the formulation of Muggleton [18] since we take into consideration not only classes in the formulae but the probability distribution of every class.

Also, we have shown that for Boolean problems, it is better to work with two classes, that is, without negative examples. However, in the case of more than two classes, negative examples are less useful than positive ones. Consider, for instance, a problem which defines four classes $\{c_1, c_2, c_3, c_4\}$. A negative example $f(a) \neq c_3$, is equivalent to $f(a) = c_1 \vee f(a) = c_2 \vee f(a) = c_4$. This last formula is less informative than $f(a) = c_2$, for instance. Hence, positive examples carry more information than negative examples if $n > 2$. This situation is extreme when the number of classes is infinite. In fact, in this case, a negative example as $sum(3, 4) \neq 6$ has no information, since it is equivalent to

$$sum(3, 4) = 0 \vee sum(3, 4) = 1 \vee sum(3, 4) = 2 \vee sum(3, 4) = 3 \ldots$$

The FLIP system is able to handle negative examples, although, as we have shown, they are not necessary for classification problems.

7 Implementation and Experiments

In the previous sections we have seen a method for comparing $\overrightarrow{D_C}(T)$ with $\overrightarrow{D_C}(H)$, resulting in the extension of the notion of generality. We have also found an estimation for $\overrightarrow{D_C}(T)$ that converges in the limit. For this estimation, we must know D_X, as in [18]. For many problems of positive data, this distribution can be estimated. For instance, in grammar learning, one may suppose that written corpora or spoken examples are constituted by mostly positive examples. In other words, if we consider two classes $\{w, b\}$ where w represents a well-formed sentence and b a badly-formed sentence, then we can assume $D_b \approx 0$ if wrong examples are not given. In other cases, if this distribution is not known, we will assume the universal distribution.

Another question with regard to implementation is the handling of zero values for some classes. This can cause division by zero in some of the previous definitions. In order to solve this problem, it is sufficient to redefine $card'_{c_i}(E) = card_{c_i}(E) + c$. In other words, c fictitious examples of each class is initially added, in order to avoid zero values. This constant c $(0 \leq c \leq 1)$ should be as small as possible in order to preserve the classes probabilities. Note that for two classes and $c = 1$, this corresponds to the Laplace corrected estimate of $g(H) = (positiveE + 1)/(possibleE + 2)$ [18].

Example 7. Consider $D_E = (100, 0)$ and that D_X assigns nine times more probability to positive examples than to negative ones, i.e. $\overrightarrow{D_{C,X}}(T) = (0.9, 0.1)$. From here, we have that:

$$\widetilde{\overrightarrow{D_C}}(T) = \frac{\overrightarrow{D_E}}{\overrightarrow{\mu}(E)} = \frac{(101, 1)}{(90.9 \times \frac{102}{101}, 0.1 \times \frac{102}{1})} = \frac{(101, 1)}{(91.8, 10.2)} = (1.1, 0.098)$$

The other question is how to compute $D_C(H)$ because X is usually infinite or very large. In these cases, as in [18], we must generate examples according to a distribution, in order to obtain $D_{C,X}(H)$ and, from here, $D_C(H)$. Finally, for an infinite number of classes, the vectors are of infinite dimension. In these cases, only the finite number of classes which appeared in the evidence are considered.

7.1 Evidence Generators

In this subsection, we discuss how to generate random samples in order to generate evidences that will permit us to evaluate the hypothesis.

The easiest case is when there exists a limited number of classes in the domain. This first method produces examples randomly from a given distribution function D_X which can be arbitrarily selected (or given). One particular instance of this case is when D_X is the universal distribution. Even if the number of classes n is finite, a generator can be useful because this number could be large and $D_C(H)$ can also be approximated.

The second case is when there are infinitely many classes in the domain. In this case, D_X must necessarily be approximated, because the infinite positive inputs to the hypothesis cannot all be evaluated. This situation is more common than one might expect, because any function over the natural, integer or real numbers has an infinite number of classes. At first glance, it seems that we cannot define a distribution to generate examples for testing to which class they belong, because probabilities, if uniformly distributed, would tend to 0. In order to solve this problem, we use an approach which is based on the universal distribution: $D_X = 2^{-Kt(x)}$. Specifically, we will use Levin's variant [11]. This variant is derived from the notion of "age" of a string, where "age is dominated by the total time needed for a string to appear out of nothing, enumerated by a constant-size program". The Levin variant is defined as $Kt(X) = \log age(x)$. More formally:

Definition 7. *The Levin Length-Time Complexity of an object x given y on a descriptional mechanism β is:*

$$Kt_\beta(x|y) = min\{LT_\beta(p|y) : \phi_\beta(\langle p, y\rangle) = x)\}$$

where $LT_\beta(p|y) = l(p) + log\ \tau_\beta(\langle p, y\rangle)$ and $\phi_\beta(a)$ represents the output given by ϕ_β on input a.

The term y represents the background knowledge and x represents the evidence. LT weights the length of the program with the logarithm of the temporal cost τ that the program takes to generate the evidence.

For the logical functional case, we have the additional advantage that if we have a confluent program, each term that we can ever construct over the program will have only a normal form. Consequently, we simply have to randomly generate lhs of equations but not complete equations. For instance, consider the *sum* example. We have the signature $\Sigma = sum, 0, s$. If we consider types, the signature is divided into $\sigma_0 = \{sum_{\{1,1\}}\}$ and $\sigma_1 = \{0_{\{\}}, s_{\{1\}}\}$, where subindexes represent the number of arguments of each function symbol and which signature they are from. In the remainder of the paper, by *type* we denote a natural number

as the index of each subsignature. We denote the constructors with σ_c. In this case, $\sigma = \{0, s\}$. We denote the set of these subsignatures (all the σ_i and the σ_c) by σ_0.

The generator can easily be constructed from this point:

function generate(n, $\sigma_\circ$): t;
  e:= select with prob. $1/card(\sigma_n)$ **an element from** σ_n ;
  let m:= **number of arguments of** e;
  for i:= **1 to** m **do begin**
    let s_i:= ***type*** **of the** i**th argument of** e;
    t_i:= **generate**(s_i, $\sigma_\circ$);
  endfor;
  construct the term t:= $e(t_1, t_2, \ldots, t_m)$;
  return t;
endfunction;

If executed with the call $generate(0, \sigma_\circ)$, this simple algorithm generates ground terms t according to a good approximation to $P(t) = 2^{-l(t)}$.

Finally, consider a function $narrows(t, P,$ var: $steps, maxsteps, \sigma_c) : u$ that narrows a term t wrt. a program P returning in u the normal form of the term (a term which is different from t such that all its function symbols are in σ_c)[6] and sets *steps* as the number of narrowing steps (including failed branches) that were necessary for deriving u. This function returns $\emptyset$ if the term t does not narrow to a normal form or if *maxsteps* are used.

From here, m positive examples can be generated in the following way:

function positive(m, P, *maxsteps*, $\sigma_\circ$): **set of examples**;
  let S:= $\emptyset$;
  while $m \neq 0$ **do**
    t:= **generate**(**0**, $\sigma_\circ$);
    u:= **narrows**(t, P, *steps*, *maxsteps*, σ_c);
    if $u \neq \emptyset$ **and TempCorrection**(*steps*) **then**
      add $\{t = u\}$ **to** S; $m := m - 1$;
    endif;
  endwhile;
  return S;
endfunction;

TempCorrection(*steps*) is a function that computes the following correction to approximate function Kt: $TemporalCorrection(steps) = (rand \cdot \log(steps)) < 1/2$ where *rand* gives a random value between 0 and 1.

In summary, the previous approach generates examples according to an approximation of the universal distribution $P(x) = 2^{-Kt(x)}$. This approach is directly applicable to logic programs (they are special cases of conditional functional logic programs) and easily adaptable to other languages. If TemporalCorrection is ignored, it gives the distribution $2^{-l(t)}$, which is an approximation to

[6] In FLIP [5] we do this by constructing an equation $t = X$ with X being a fresh variable. The normal form is given by the substitution of X.

the universal distribution $P(x) = 2^{-K(x)}$ with $K(x)$ being the absolute Kolmogorov complexity.

Finally, by using the generators we can approximate $D_{C,X}(H)$ by generating k examples into an evidence E^k. If D_X is uniform, by corollary 1, then $D_C^k(H) \simeq D_{C,X}^k(H) = E^k$ when k is large. In general,

$$D_C^k(H) = \left[\frac{D_{C,X}^k(H)}{\left[\sum_{\substack{x \in E_k \\ rhs(x) = c_i}} D_X(x) \right]_{i=1..n}} \right]^{norm}$$

Example 8. Let us recover example 4. Consider the following evidence E formed by 100 examples ($k = 100$) and generated by the universal distribution $2^{-l(x)}$:

$Example$	$Probability$	$Number\ of$	$H_2 = mod3$
$mod(0)$	0.5	52	$True$
$mod(1)$	0.25	26	$False$
$mod(2)$	0.125	12	$False$
$mod(3)$	0.0625	6	$True$
$mod(4)$	0.0312	3	$False$
$mod(5)$	0.015	1	$False$

We have

$$D_C^k(H) = \left[\frac{D_{C,X}^k(H)}{\left[\sum_{\substack{x \in E_k \\ rhs(x) = c_i}} D_X(x) \right]_{i=1..n}} \right]^{norm} = \left[\frac{(58, 42)}{(26.38, 8.109)} \right]^{norm} =$$

$$[(2.199, 5.18)]^{norm} = (0.298, 0.702) \simeq (0.33, 0.66)$$

With all this, we can retake the entire definition of the probability $P(H|E)$:

Definition 8. *The optimality of a hypothesis is*

$$Opt(H) = ln\ p(H|E) - d_m = m\ ln(\frac{1}{u(H)}) - sz(H)$$

Apart from being useful for approximating $D_{C,X}(H)$ and $D_C(H)$, the generators, are useful for generating sets of evidence and essaying the robustness of learning systems. However, we do not deal with this question in this paper.

7.2 Experiments

Let us present an example that illustrates the use of the approximations. This example has been implemented in the FLIP system and describes a case where D_X is known (a uniform distribution).

Example 9. Consider Table 1 which represents an evidence of size 100 based on a uniform D_X of a simplified database for fitting contact lenses which originally appeared in [3]. [7]

Eyeglass Prescription	*Astigmatism*	*Tear Production Rate*	*contact lenses*	Examples
myopia	no	reduced	no	13
myopia	no	normal	soft	15
myopia	yes	reduced	no	11
myopia	yes	normal	hard	11
hypermetropia	no	reduced	no	13
hypermetropia	no	normal	soft	12
hypermetropia	yes	reduced	no	14
hypermetropia	yes	normal	hard	11

Table 1. Databases for fitting contact lenses

This database can be interpreted as a problem that determines whether the patient needs to use contact lenses, and if this is the case, what type of lenses s/he must use. Therefore, this case is a classification problem with three target classes: $C = \{no, soft, hard\}$ with $\overrightarrow{D_C} = (1/2, 1/4, 1/4)$.

With our method we can approximate $\widetilde{\overrightarrow{D_{C_1}}}(T)$ as:

$$\widetilde{\overrightarrow{D_C}}(T) = \frac{\overrightarrow{D_E}}{\overrightarrow{\mu_E}} = (0.51, 0.27, 0, 22)$$

Consider these possible hypotheses with their respective D_C and the angle they form with $\widetilde{\overrightarrow{D_C}}(T)$.

$$H_1 = \{ lens(X, Y, r) = no \qquad D_C = [0.5, 0, 0] \qquad \alpha_1 = 0.599\ rad$$

$$H_2 = \begin{cases} lens(X, Y, r) = no \\ lens(X, Y, n) = soft \end{cases} \quad D_C = [0.5, 0.5, 0] \quad \alpha_2 = 0.464\ rad$$

$$H_3 = \begin{cases} lens(X, Y, r) = no \\ lens(X, no, n) = soft \end{cases} \quad D_C = [0.5, 0.25, 0] \quad \alpha_3 = 0.365\ rad$$

$$H_4 = \begin{cases} lens(X, Y, r) = no \\ lens(X, no, n) = soft \\ lens(X, yes, n) = hard \end{cases} \quad D_C = [0.5, 0.25, 0.25] \quad \alpha_3 = 0.174\ rad$$

[7] The simplification consists in assuming that a fourth attribute (age) is always *young*.

As expected, the angle with respect to $\widetilde{\overrightarrow{D_C}}(T)$ gets lower and lower as the hypothesis becomes more accurate. Moreover, the corresponding $u(H)$ and optimalities are:

$$\begin{array}{lll} H_1 : l = 4, & u = \frac{tan0.599}{1+tan0.599} \times 0.5 = 0.2028, & Opt = 100ln(\frac{1}{0.2028}) - 4 = 155.55 \\ H_2 : l = 8, & u = 0.3333 \times 1 = 0.333, & Opt = 100ln(\frac{1}{0.3333}) - 8 = 101.86 \\ H_3 : l = 8, & u = 0.276 \times 0.75 = 0.207, & Opt = 100ln(\frac{1}{0.207}) - 8 = 149.4 \\ H_4 : l = 12, & u = 0.15 \times 1 = 0.15, & Opt = 100ln(\frac{1}{0.15}) - 12 = 177.71 \end{array}$$

This and other experiments can be found in the FLIP system web page:

`http://www.dsic.upv.es/~jorallo/flip/` .

8 Conclusions

This paper has converted the notion of hypothesis generality into a new notion of class unevenness. This new view of the problem places the learning of Boolean functions as a special case: when the number of classes equals 2 and where any proportion of examples of class true and class false can be used.

Moreover, it clarifies the problem when sparse or non-uniform evidence is given for one class. Consider, for instance, the *lens* example in a purely logical way, as a predicate $lens(X, Y, Z, C)$ where C is the argument that represents the classes to be predicted (usually done by a mode declaration in ILP systems). The learnability of the problem and the evaluation of hypotheses do not only depend on the number of positive or negative examples, but also depends on the proportion of classes in the evidence, hypothesis and theory.

In conclusion, the approach presented in this paper neglects the use of negative evidence and supports the conversion of Boolean problems into classification problems. Although there is important literature on handling class distribution, it does not deal with universal representation languages, such as functional logic programming. In this regard, the IFLP framework is the most natural step for handling classes from the ILP point of view.

In the presence of noise, our work could be extended following the model presented in [13].

Acknowledgments

We would like to thank Stephen Muggleton for suggesting the implications and naturalness of IFLP for classification problems. We also thank the anonymous reviewers for their useful comments.

References

1. H. Arimura and T. Shinohara. Inductive inference of prolog programs with linear data dependency from positive data. In T. Kitahashi H. Jaakkola, H. Kangassalo and A. Markus, editors, *Proc. Information Modelling and Knowledge Bases V*, pages 365–375. IOS Press, 1994.

2. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
3. J. Cendrowska. PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machines Studies*, 27:349–370, 1987.
4. H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1985.
5. C. Ferri, J. Hernández, and M.J. Ramírez. The flip system: From theory to implementation. *submitted to Machine Learning*, 2000.
6. E.M. Gold. Language indentification in the limit. *Information and Control*, 10:447–474, 1967.
7. M. Hanus. The Integration of Functions into Logic Programming: From Theory to Practice. *Journal of Logic Programming*, 19-20:583–628, 1994.
8. J. Hernández and M.J. Ramírez. Inverse Narrowing for the Induction of Functional Logic Programs. In *Proc. Joint Conference on Declarative Programming, APPIA-GULP-PRODE'98*, pages 379–393, 1998.
9. J. Hernández and M.J. Ramírez. A Strong Complete Schema for Inductive Functional Logic Programming. In *Proc. of the Ninth International Workshop on Inductive Logic Programming, ILP'99*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 116–127, 1999.
10. T. Koshiba, E. Mäkinen, and Y. Takada. Learning deterministic even linear languages from positive examples. *Theoretical Computer Science*, 185:63–79, 1997.
11. L.A. Levin. Universal search problems. *Probs. Inform. Transm.*, 9:265–266, 1973.
12. M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and its Applications*. 2nd Ed. Springer-Verlag, 1997.
13. E. McCreath and A. Sharma *ILP with Noise and Fixed Example Size: A Bayesian Approach* 15th Intl. Joint Conference on Artificial Intelligence, pp. 1310-1315, 1997.
14. E. Martin and A. Sharma. On sufficient conditions for learnability of logic programs from positive data. In S. Dzeroski and P. Flach, editors, *Proc. of the 9th International Workshop on Inductive Logic Programming, ILP'98*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 198–209. Springer-Verlag, 1999.
15. S. Muggleton. Inductive Logic Programming. *New Generation Computing*, 8(4):295–318, 1991.
16. S. Muggleton. Predicate invention and utilisation. *Journal of Experimental and Theoretical Artificial Intelligence*, 6(1):127–130, 1994.
17. S. Muggleton. Inverse entailment and progol. *New Generation Computing Journal*, 13:245–286, 1995.
18. S. Muggleton. Learning from positive data. *Machine Learning (accepted subject to revision)*, 1999.
19. M. Krishna Rao. A framework for incremental learning of logic programs. *Theoretical Computer Science*, 185:191–213, 1997.
20. T. Shinohara. Inductive inference of monotonic formal systems from positive data. *New Generation Computing*, 8:371–384, 1991.