Learning Functions from Imperfect Positive Data

Filip Železný

Center for Applied Cybernetics, Czech Technical University
Prague, Czech Republic

Abstract. The Bayesian framework of learning from positive noise-free examples derived by Muggleton [12] is extended to learning functional hypotheses from positive examples containing normally distributed noise in the outputs. The method subsumes a type of distance based learning as a special case. We also present an effective method of outlier-identification which may significantly improve the predictive accuracy of the final multi-clause hypothesis if it is constructed by a clause-by-clause covering algorithm as e.g. in Progol or Aleph. Our method is implemented in Aleph and tested on two experiments, one of which concerns numeric functions while the other treats non-numeric discrete data where the normal distribution is taken as an approximation of the discrete distribution of noise.

1 Introduction

Most of noise-handling techniques in machine learning are suited for the type of errors caused by wrong classification of training examples into classes, e.g. true or false. In a powerful family of ML methods such as ILP, which uses a Turing-equivalent representation to produce hypotheses and can therefore hypothesise about complicated input-output relations (e.g. functions), the role of *noise in attributes (arguments)* has been recognised [1, 7] but rarely attempted to handle. Moreover, we are not aware of a system which would directly exploit the knowledge of a particular noise-distribution in arguments, despite the fact that Bayesian and distance-based techniques - which have recently been paid a lot of attention in ILP [6, 8, 3, 16, 15] - can very well serve for this purpose.

We want to test the hypothesis that by exploiting the knowledge of a particular noise-distribution in the data (though it may hold only approximately) we may outperform standard noise-handling techniques. In the next section we shall see how to optimally (in the Bayes sense) learn functions with unknown domains and normally-distributed noise in the output arguments. The outstanding role of the normal noise-distribution has been extensively justified in many sources (see e.g. [2]) namely on the basis of the central limit theorem. We implemented the method in the ILP

system Aleph. Section 3 describes an effective outlier-identification technique applicable in the clause-by-clause theory-construction performed by this system, modified as to follow the guideline developed in Section 2.

In the experimental part (Section 4), we first test our method on artificial data. In particular, we learn numeric functions representable by a one-clause Prolog program. This experiment will comply with the conditions of *U-learning* [14] and the noise will be exactly normal. We shall then also try to slightly relax the conditions of U-learning. The second experiment will be based on English verb past tense data. These data have functional, discrete and non-numeric character. The output argument will be damaged by altering a certain number of characters in the word and the continuous normal distribution of noise will only be approximated. This kind of errors simulates the one encountered in literal data digitisation by e.g. OCR systems or human transcription. The predictive accuracy of the resulting multi-clause theory will be significantly improved by the outlier-identification technique described in Section 3. Section 5 concludes.

2 Bayesian Framework

A standard approach to learn functions from positive data in ILP makes use of the closed-world assumption (CWA). Using CWA, we substitute negative examples necessary in the normal ILP setting e.g. by an *integrity constraint* which falsifies all hypotheses which yield the output $\boldsymbol{out_h}$ for an input $\boldsymbol{in}$, such that there exists a positive example $e(\boldsymbol{in}, \boldsymbol{out_e})$ and $\boldsymbol{out_e} \neq \boldsymbol{out_h}$. But CWA clearly cannot be used if the output part of examples contains noise.

Another common drawback of functional learners is that they get no information from the distribution of values in the input parts of the presented positive examples. To get a rough idea how such information could be used, imagine that we are learning scalar functions on the integer (sampling) interval $\langle -10; 10 \rangle$. Assume that the current hypothesis space is $\{equal(in, out), sqrt(in, out)\}$ and we get two positive examples $e(0, 0)$ and $e(1, 1)$. Then both hypotheses are consistent with the examples but $sqrt/2$ has higher posterior probability (in the Bayes sense) since it is less general (defined only for non-negative inputs).

Both of these problems will be treated in the following framework embedded in the Muggleton's U-learning scheme of learning from positive data [12]. For ease of insight we shall formalize it for numeric data to later easily generalize for non-numeric data in the experimental part of the text.

Let I be a finite set, if f and g are (real) functions on a superset of I then the *Euclidean distance* between f and g on I is

$$\mathcal{E}(f(I), g(I)) = \sum_{i \in I} (f(i) - g(i))^2 \tag{1}$$

The normal distribution $N_{\mu,\sigma}(x)$ with *mean* μ and *standard deviation* σ is given as

$$N_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp - \frac{(x-\mu)^2}{2\sigma^2} \tag{2}$$

Let ***bold*** characters denote vectors, their elements being addressed by the lower index. The *instance space* X will be the Cartesian product of the sets of possible inputs I and outputs O. An instance (example)[1] $e \in X$ is then given by the input part $\boldsymbol{in}(e) \in I$ and output part $\boldsymbol{out}(e) \in O$. These parts are in general vectors of $|\boldsymbol{in}|$ and $|\boldsymbol{out}|$ elements, respectively. A (functional) *hypothesis* H on the instance set $X = I \times O$ is a tuple $\langle H_d \subseteq I, h : H_d \to O \rangle$. H_d is the *domain* of H and $H_c = \{e \in E | \boldsymbol{in}(e) \in H_d\}$ is the *coverage* of H. H is said to be *consistent with* $e \in X$ if $\boldsymbol{in}(e) \in H_d$, H is *consistent* (with E) if it is consistent with all $e \in E$. The mapping (such as h) corresponding to a hypothesis (such as H) will be always denoted by lowering the case and by $h_j(.)$ we shall denote the j^{th} element of $h(.)$.

Given a probability distribution D_I on the input space and assuming mutual independence of outputs, we can express the distribution of the conditional probability on the instance space under the condition of validity of a hypothesis H as

$$\begin{gathered} D_{X|H}(e) = D_{X|H}(\boldsymbol{in}(e), \boldsymbol{out}(e)) = D_{I|H}(\boldsymbol{in}(e)) D_{O|H,\boldsymbol{in}(e)}(\boldsymbol{out}(e)) = \\ = D_{I|H}(\boldsymbol{in}(e)) \prod_{j=1}^{|\boldsymbol{out}|} D_{O|H,\boldsymbol{in}(e)}(\boldsymbol{out}_j(e)) \end{gathered} \tag{3}$$

$D_{X|H}(e)$ is zero if e is not consistent with H since then $D_{I|H}(\boldsymbol{in}(e)) = 0$. Otherwise, $D_{I|H}(\boldsymbol{in}(e))$ can be expressed as

$$D_{I|H}(\boldsymbol{in}(e)) = \frac{D_I(\boldsymbol{in}(e))}{D_I(H_d)} \tag{4}$$

and the conditional probability on the outputs will express our assumption of normally distributed error with standard deviation σ_j in the j^{th} output argument

$$D_{O|H,\boldsymbol{in}(e)}(\boldsymbol{out}_j(e)) = N_{h_j(\boldsymbol{in}(e)),\sigma_j}(\boldsymbol{out}_j(e)) \tag{5}$$

[1] We reserve plain characters for vector examples and mappings to improve readability.

Given a *prior probability* distribution on hypotheses D_H, a target hypothesis H^*, a set of examples $E = e_1, e_2, ..., e_m$ selected by m statistically independent choises from $D_{X|H^*}$, the posterior probability of a hypothesis H consistent with E can be found by applying the well-known Bayes rule and Eqs. 3,4,5 as

$$P(H|E) = P(H|e_1, e_2, ..., e_m) = D_H(H)D_X^{-1}(E)D_{X|H}(e_1, e_2, ..., e_m) = \\ = D_H(H)D_X^{-1}(E)\prod_{i=1}^{m}\left[D_I^{-1}(H_d)D_I(\boldsymbol{in}(e))\prod_{j=1}^{|\boldsymbol{out}|}N_{h_j(\boldsymbol{in}(e)),\sigma_j}(\boldsymbol{out}_j(e_i))\right] \tag{6}$$

To choose the most-promising hypothesis, we want to maximise $P(H|E)$ w.r.t H. We shall take logarithms of both sides of this equation (to maximise $\ln P(H|E)$) and for this sake we disassemble the rightmost side into several terms. First, it is argued in [12] that D_H should be expected to obey

$$\ln D_H(H) = -size(H)const_N \tag{7}$$

where $size(H)$ measures the number of bits necessary to encode the hypothesis H and $const_N$ is a normalising constant ensuring that $\sum_H D_H(H)$ sums to one; this constant is neglectable when maximising $\ln P(H|E)$. Following the same source, $\prod_{i=1}^{m} D_I^{-1}(H_d) = D_I^{-m}(H_d)$ can be identified as

$$\ln D_I^{-m}(H_d) = -m \ln gen(H) \tag{8}$$

where $gen(H)$ is the *generality* of H (i.e. the portion of the input space covered by H_d). The term $\ln\left[D_X^{-1}(E)\prod_{i=1}^{m} D_I(\boldsymbol{in}(e_i))\right] = const_1$ is constant for all hypotheses, so it can be neglected when maximising $\ln P(H|E)$. Finally it holds

$$\ln\prod_{i=1}^{m}\prod_{j=1}^{|\boldsymbol{out}|}N_{h_j(\boldsymbol{in}(e_i)),\sigma_j}(\boldsymbol{out}_j(e_i)) = \\ = \ln\prod_{i=1}^{m}\prod_{j=1}^{|\boldsymbol{out}|}\frac{1}{\sigma_j\sqrt{2\pi}}\exp -\frac{(\boldsymbol{out}_j(e_i)-h_j(\boldsymbol{in}(e_i))^2}{2\sigma_j^2} = \\ = -m\sum_{j=1}^{|\boldsymbol{out}|}\ln(\sigma_j\sqrt{2\pi}) - \sum_{j=1}^{|\boldsymbol{out}|}\frac{1}{2\sigma_j^2}\sum_{i=1}^{m}(\boldsymbol{out}_j(e_i) - h_j(\boldsymbol{in}(e_i))^2 \tag{9}$$

The term $-m\sum_{j=1}^{|\boldsymbol{out}|}\ln(\sigma_j\sqrt{2\pi}) = const_2$ does not depend on the hypothesis and can be neglected when maximising $\ln P(H|E)$. Combining Eqs. 7-9 and considering Eq. 1 we arrive to the fact that to maximise $\ln P(H|E)$ we need to maximise the function $f_E(H)$ (w.r.t. consistent hypotheses H)

$$f_E(H) = -m \ln gen(H) - size(H) - \sum_{j=1}^{|\boldsymbol{out}|}\frac{1}{2\sigma_j^2}\mathcal{E}(\boldsymbol{out}_j(E), h_j(\boldsymbol{in}(E))) \tag{10}$$

which can be simplified if there is only one output argument as

$$f'_E(H) = -m \ln gen(H) - size(H) - \frac{1}{2\sigma^2}\mathcal{E}(out(E), h(\boldsymbol{in}(E))) \qquad (11)$$

The first two terms in $f_E(H)$ or $f'_E(H)$ express a generality - size tradeoff derived by Muggleton [12] for the case of learning classification hypotheses from noise-free positive data. In our case of learning functional hypotheses from data with normal output noise, we have instead arrived to a generality - size - Euclidean distance tradeoff, where generality is measured on the input space (function domain) and the output-distance term is weighted by the inverse value of the *variance* σ^2. This is natural: the more noisy (more deviated) are the outputs in the examples, the more it makes sense to decide rather by the input domain data (by measuring the generality on the input domain) and prior hypothesis probability (reflected by the size term) and vice-versa.

In the following we shall concentrate on single-output hypotheses and therefore maximise $f'_E(H)$. Thus the assumption of statistically independent outputs is no longer needed.

3 Outlier Identification

In a hypothesis constructed by an ILP system (ordered set of Prolog clauses $C^1, ..., C^n$), one example may be consistent with more than one clause. Although we are learning functional hypothesis, we do not require consistency with at most one clause, since this would too much constrain the learning algorithm. Instead, we shall *interpret* the Prolog program functionally, i.e. as[2] `once(target_predicate(`$\boldsymbol{inputs}, OUTPUT$`))`. Accordingly, we define the *reduced domain* and *reduced coverage* of a clause C^n as $C^n_{rd} = C^n_d \setminus \cup_{k=1}^{n-1} C^k_d$ and $C_{rc} = \{e \in E | \boldsymbol{in}(e) \in C_{rd}\}$.

To select a hypothesis by maximising $f'_E(H)$ we need to have at hand a set of candidate hypotheses. But in a typical ILP system, hypotheses are constructed clause-by-clause, therefore [12] proposes estimates of the value $gen(H)$ and $size(H)$ based on $|C^n_d|$, $gen(H^n)$, $gen(H^{n-1})$ and $size(C^n)$ where $H^n = \{C^1, ..., C^n\}$ (i.e. H^{n-1} is the already-constructed partial hypothesis and C^n the currently added clause) and H is the final hypothesis. In an analogical spirit, if $|C^i_{rc}|^{-1}\mathcal{E}(out(C^i_{rc}), c^i(\boldsymbol{in}(C^i_{rc})))$ (the average distance of the output of C^i from individual examples on its domain C^i_{rd}), is approximately equal for all clauses C^i in the final hypothesis

[2] The standard Prolog `once/1` predicate returns only the first-found answer whatever may be the number of solutions.

H, we may make the following estimation[3]

$$\mathcal{E}(out(E), h(\boldsymbol{in}(E))) \approx \frac{|E|}{|C^n_{rc}|}\mathcal{E}(out(C^n_{rc}), c^n(\boldsymbol{in}(C^n_{rc}))) \qquad (12)$$

Let the function $f^e_E(C^n)$ denote the estimate of $f'_E(H)$ determined by substituting the size, generality and distance terms by their estimates described in [12][4] and the estimate in Eq. 12, respectively. The clause C^n that maximises $f^e_E(C^n)$ will then be added to the current hypothesis H^{n-1}.

In the clause-by-clause functional hypothesis construction, we are no longer learning optimally (as by Eq. 11). The algorithm maximising $f^e_E(C^n)$ for each added clause has a greedy character and we can use the following heuristic to improve the clause ordering: If there exists a clause with good accuracy on (low output-distance from) a large part of the example set but poor accuracy on a few exceptions (outliers), then this general clause should be preceeded with a more special clause 'handling' these exceptions. Together with the *once*−interpretation, this strategy will produce a form of a specific-to-general decision list, whose advantage to functional representation has been argued in [10].

To attain such clause-ordering, we use the 'degree of freedom' given by the seed-example selection in ILP systems like Aleph [5] and Progol [11]. In these systems, the seed-example is selected randomly or in the presentation order and used for the construction of a *bottom clause* which is then suitably generalised. The idea of our method is that we direct the seed-example selection as to first choose (and cover) those examples that are outliers to some potentially good clause. To protect efficiency, we shall avoid backtracking (deleting previously constructed clauses) .

During the computation of $f^e_E(C^n)$ for each candidate clause C^n, we also evaluate the function $Hope_E(C^n) = max_{O\subset E}(f^e_{E\setminus O}(C^n))$ which yields the highest evaluation potentially reached by C^n if some example subset O (outliers) were avoided, i.e. covered by some previous clause. Evaluating $f^e_{E\setminus O}(C^n)$ for every $O \subset E$ would be intractable, but we can avoid it by first sorting the examples $e \in C^n_{rc}$ decreasingly by the value $(out(e)-c^n(\boldsymbol{in}(e))^2$, i.e. by their contribution to the distance $\mathcal{E}(out(C^n_{rc}), c(\boldsymbol{in}(C^n_{rc})))$ (see Eq. 1). Then outliers are identified by successively replacing examples in this order from C^n_{rc} into the (initially empty) set OL. The set OL which maximises $f^e_{E\setminus OL}(C^n)$ during this cycle is taken as the outlier set and it then holds that $f^e_{E\setminus OL}(C^n) = max_{O\subset E}(f^e_{E\setminus O}(C^n))$. To roughly see

[3] Remind that c^n is the mapping corresponding to the hypothesis C^n.
[4] Where $|C^n_{rc}|$ is taken instead of $|C^n_c|$ denoted as p in [12].

why, note that by exchanging any example e_1 from OL with any example e_2 from $C^n_{rc} \backslash OL$, obtaining $OL_{alt} = \{e_2\} \cup OL \backslash \{e_1\}$, the generality and size estimates in the function f^e maintain the same value and the distance term remains the same or grows as the contribution of e_1 to the Euclidean distance is the same or smaller than that of e_2 (due to the precomputed decreasing order). Therefore $f^e_{E \backslash OL_{alt}}(C^n) \leq f^e_{E \backslash OL}(C^n)$. Fig. 1 shows an example of outlier identification in the English past tense data domain (Section 4.2).

In the learning algorithm, each example in E is assigned a *selection-preference* value, initiated to zero. For every evaluated candidate clause C^n with outliers OL, the current selection-preference value of each example $e \in OL \subset E$ is updated by adding a value increasing with the $Hope$ of C^n. When selecting the seed-example, the example with maximum selection-preference value is chosen[5]. This way, examples that are outliers of high-$Hope$ clauses will be covered in the earlier stages of hypothesis construction. Typical for the described method as implemented in Aleph is that at one stage a clause with high $Hope$ is evaluated, rejected due to outliers, which are then forced to be covered. When the same clause is evaluated newly (as a result of a newly selected seed example), it is accepted since its outliers are already covered. It is the multiple evaluation of one clause intrinsic to the *cover algorithm* of Aleph that enables us to implement the method without backtracking. The only (slightly) superlinear computational overhead introduced by the technique is the sorting of examples by their contribution to the Euclidean distance.

4 Experiments

4.1 Learning Numeric Functions

In the first experiment, we want to identify numeric functions composed of the four elementary functions $\{\ln(x), \sin(x), \cos(x), x+y\}$ by one Prolog clause. The hypothesis bias is limited by the maximum composition depth 4. There are 425 functions in this hypothesis space assuming commutativity of addition [4]. As *background knowledge*, the learning system uses the Prolog definitions of the elementary functions (e.g. `ln(X,Y):-X>0, Y is log(X)`). To comply with the framework of U-learning, we repeatedly perform the learning process with a target hypothesis chosen with a probability exponentially decreasing with the size of its Prolog notatiton. The following table lists the used set of target functions, their input domains

[5] In the first step, before generating any clause, the seed example is chosen randomly.

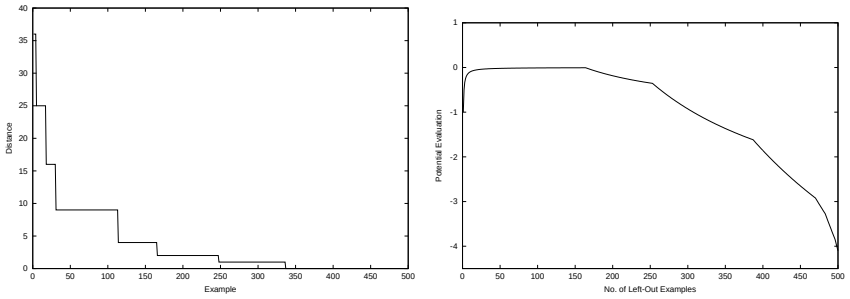

Fig. 1. Outlier Identification in English past tense data with noise variance 0.3. The left diagram shows the decreasing output distance contribution of each of 500 examples w.r.t the clause `past(A,B):-split(B,A,[e,d])`. The right diagram plots for each example e_i the potential evaluation of the clause if $\{e_1, ..., e_i\}$ were avoided from the clause's domain. This potential evauation reaches its maximum for example no. 165. Examples 1-165 are thus considered outliers.

within the chosen sampling interval of integers $\langle -10; 10\rangle$ for example presentation, and their prior probabilities[6].

Target Function	Domain $\in \langle -10; 10\rangle$	Prior Probability
$\ln(x)$	$\langle 1; 10\rangle$	$1/2$ * c_n
$\ln(\sin(x))$	$-10, \langle -6; -4\rangle, \langle 1; 3\rangle, \langle 7; 9\rangle$	$1/4$ * c_n
$\cos(x) + \ln(\cos(x))$	$\langle -7; -5\rangle, \langle -1; 1\rangle, \langle 5; 7\rangle$	$1/8$ * c_n
$\ln(\sin(x) + \cos(x))$	$\langle -7; -4\rangle, \langle 0; 2\rangle, \langle 6; 8\rangle$	$1/16$ * c_n

Examples are presented in the form `e(input,output)` from equal probability distribution on the input domain and the output value is distorted by normal noise. We test three learning methods. BL denotes the Bayesian technique developed in Section 2. DBL is a simpified BL, where size and generality of hypotheses are ignored when maximising $f'_E(H)$, i.e. we ignore the information in the input domain data distribution and in the prior hypothesis probability distribution. We thus reason only on the basis of the output distance and so DBL corresponds to a simple kind of distance based learning. The last tested method is based on a simple classical manner of treating noise in real values in ILP: the standard Aleph (Progol) algorithm of learning from positive data is used, but we introduce a predicate `close/2` as part of the background knowledge, such that `close(A,B)` is true if the values in `A` and `B` differ by less than 10%. The

[6] c_n is a normalising constant

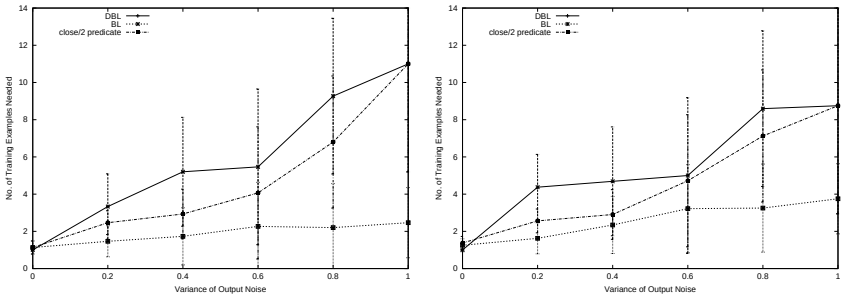

Fig. 2. Learning Numeric Functions. The left diagram shows the minimum number of examples each of the tested methods needed to correctly identify the target function with growing variance in the output noise. For each method and each value of variance the experiment was repeated 20 times, the average result is plotted with standard deviation in the measurement points. The right diagram reflects a similar experiment where, however, the prior hypothesis probabilities were not respected, i.e. the target hypotheses were presented with equal probability.

learner may thus identify e.g. $\ln(x)$ from noisy-output data by the clause `e(A,B):-log(A,C),close(C,B).`

Considering Fig. 2, BL clearly outperforms the other two methods, i.e. the exploitation of the generality and size measures proves useful (compare with DBL) as well as the exploitation of the Eucledian distance measure derived from the normal noise distribution (compare with `close/2`). Relaxing the U-learning conditions by presenting target hypotheses in equal probabilities makes the difference btw. BL and the other methods smaller, but not significantly.

4.2 Learning English Past Tense Rules

The second experiment is based on 1392 tuples of English verbs and their past tenses. Learning rules of English past tense by a multi-clause Prolog program has been studied with noise-free data [9, 13]. The background knowledge contains the predicate `split/3` which splits a word into a prefix and suffix (e.g. `split([m,a,i,l,e,d],[m,a,i,l],[e,d])`; see [9] for typical hypotheses constructed by ILP in this domain. Unlike the noise-free experiments, in our case the output argument is distorted by altering a number of characters in the word such that the probability of n wrong characters decreases exponentially with n^2 to approximate the normal distribution. Following is an example of 5 data with noise.

```
past([m,e,e,t],[m,e,t]).
```

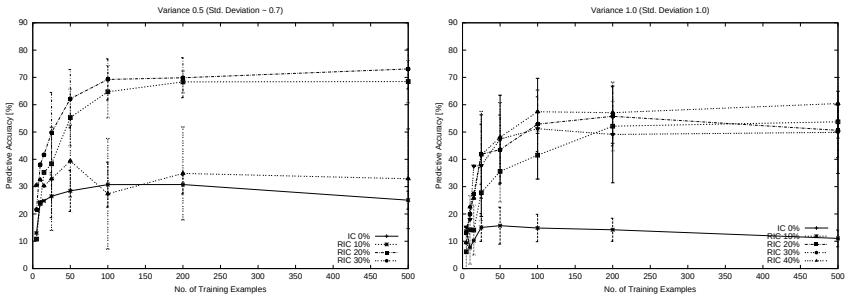

Fig. 3. Learning past tense rules with RIC's for two values of output noise variance. The training sets are selected randomly from the past-tense database and contain successively 5, 10, 15, 20, 50, 100, 200 and 500 examples; the testing set for measuring the predictive accuracy is always composed of 500 examples not including any of the training example. For each training set volume and each tested tuning of RIC, the experiment was repeated 20 times and the average value with its standard deviation is plotted.

```
past([m,i,n,i,s,t,e,r],[m,i,n,i,s,t,w,r,e,d]).
past([n,e,c,e,s,s,i,t,a,t,e],[n,e,c,q,s,s,i,y,a,t,e,d]).
past([o,b,s,e,r,v,e],[o,b,s,e,r,v,e,d]).
past([o,c,c,u,r],[o,c,c,u,r,r,e,f]).
```

The normal probability distribution was discretised in such a way that for $\sigma = 1$, the majority of examples contained at least one error, i.e. in the language of binary classification most of the presented positives were actually negatives.

We compare our method with the standard algorithm of Progol (Aleph) whose performance is good on the noise-free past tense data [13].[7] The integrity constraint (see Section 2) used in [13] to substitute negative examples cannot work in the noisy domain but we may use a *relaxed integrity constraint* (RIC) which falsifies hypotheses giving wrong outputs for a certain minimum percentage of examples. The question which percentage (tolerance) should be allowed for which level of noise (variance) is solved empirically in a preliminary comparative experiment of RIC's tuned to 0%, 10%, 20%, 30% and 40% of tolerance, shown in Fig. 3.

We shall use the Progol (Aleph) algorithm with the best performing RIC for each variance (tolerance 20% for $\sigma^2 = 0.5$, 30% for $\sigma^2 = 1$) to compete with our method, which will first be simplified in the following

[7] Progol was only outperformed by the method of *analogical prediction* whose application scope is rather specialised.

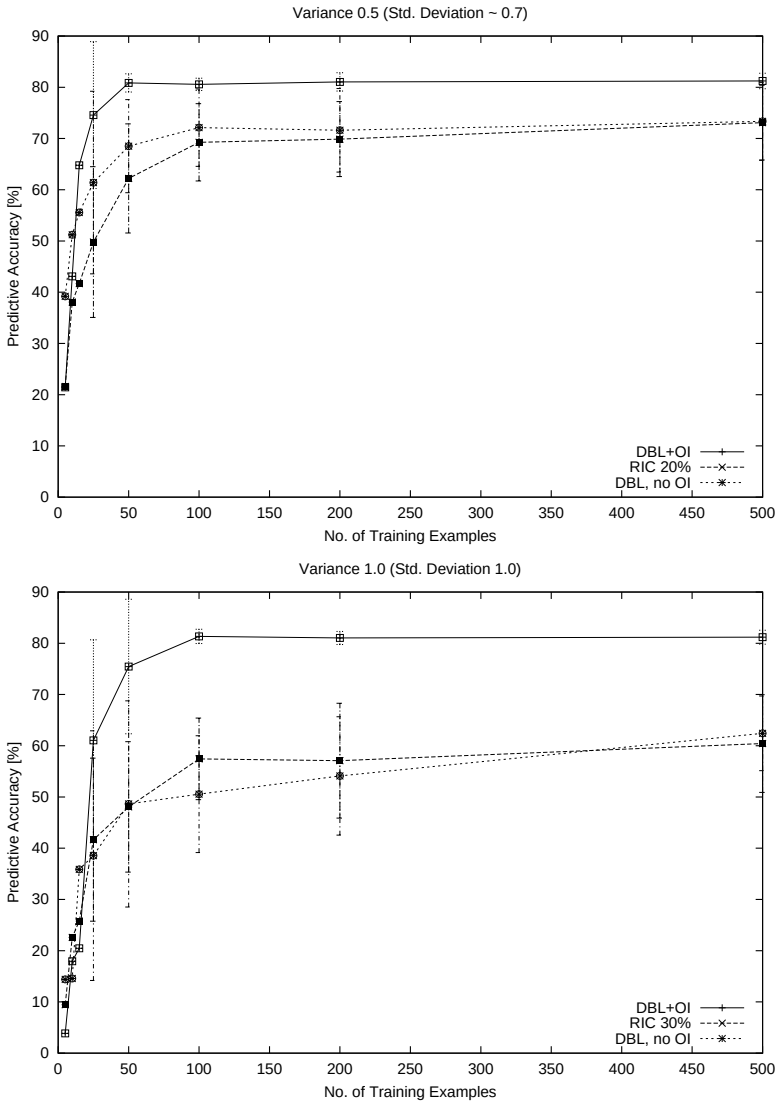

Fig. 4. Learning past tense with RIC, DBL and DBL+OI. The experimental setup is identical to the previous experiment (Fig. 3), from which the best performing RIC was taken for comparision.

ways. First, we require that any resulting hypothesis must yield some output for any input word, i.e. the generality of all acceptable hypotheses is identical. We therefore consider the generality term in $f'_E(H)$ constant. Next, we limit the hypothesis bias by a maximum *variable depth* [11] and within this bias we have no reason to expect that prior hypothesis probability decreases with the hypothesis size, i.e. the size term is also considered constant. Since only the output distance term (measured as squared Hamming distance[8]) is then maximised, we refer to this simplified method as distance-based (DBL).

As we are learning a multi-clause hypothesis, the outlier identification (OI) technique (Section 3) may be used. The performance of the three methods (Aleph with RIC, DBL and DBL+OI) is shown in Fig. 4 for two levels of noise.

We observe that the DBL method alone is comparable with the best-tuned integrity constraint. However, with RIC we need to first determine (e.g. empirically) a good value of tolerance, otherwise the performance may be very poor (Fig. 3). This is not necessary with DBL. Note also that to maximise $f'_E(H)$ the DBL learner does not need to know the value of the noise variance if the size and generality terms are considered constant. We also observe that outlier identification greatly improves the predictive accuracy of the multi-clause hypothesis constructed with DBL and we think this would be the case with any functional data with a high percentage of exception-items, as English past tense. Note also that the integrity constraint method cannot be further improved with OI, since the OI technique is directly based on the distance measure.

5 Conclusions and Future Work

We have illustrated how the exploitation of the knowledge of a particular noise distribution in training data arguments can be utilized to outperform classical noise-handling techniques. Using a Bayesian framework for optimal learning of functional hypothesis in the presence of normal noise, we also exploit the knowledge of the prior hypothesis probability and

[8] E.g. the distance of the hypothesis output [a,b,c] from the example outputs {[a,b,x], [a,x]} would be $1^2 + 2^2 = 5$ because the first example differs from [a,b,c] in one corresponding character and to compare two lists of different lengths we add a suffix to the shorter with characters considered mismatches, i.e. the second example is taken as [a,x,x]. In the normal noise distribution definition we accordingly measure the Hamming distance instead of the subtraction $(x - \mu)$ (see Eq. 2). Such defined distance measure is natural in the experimented domain and different definitions may be suitable in other domains.

its generality on the input domain of the learned function. The advantage of exploiting all these properties was shown in a function-learning experiment.

We implemented the method in the ILP system Aleph and for the clause-by-clause construction of hypotheses guided by this method we proposed a heuristic technique which forces outliers to be covered first so that general clauses can be accepted in the later stage of the clause-by-clause hypothesis construction. This ordering of clauses improves the predictive accuracy of the final hypothesis interpreted functionally, e.g. by the Prolog `once/1` predicate. This was illustrated in an experiment with a high percentage of exceptional examples. The technique does not introduce backtracking into the learning algorithm.

Our future work will focus on proving a bound of expected error related to the deveoped Bayesian learning with noise, similar to the one shown for the noise-free data case in [12]. Next, we want to extend the framework to non-functional hypotheses learning from data with normal noise in arguments.

6 Acknowledgements

The experimental part of this work was conducted during the author's stay in LORIA, France - special thanks go to Amedeo Napoli and Hacene Cherfi. Thanks as well to Ashwin Srinivasan and Steve Moyle for feedback concerning positive-only learning as implemented in Aleph, and to the careful ILP'01 reviewers. The author is supported by the Ministry of Education of the Czech Republic under Project No. LN00B096, and by the project IST-1999-11.495 Sol-Eu-Net.

References

1. W. Emde and D. Wettschereck. Relational instance-based learning. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 122–130. Morgan Kaufmann, 1996.
2. V. V. Fedorov. *Theory of optimal experiments.* Academic Press, 1972.
3. Peter A. Flach and Nicolas Lachiche. Decomposing probability distributions on structured individuals. In Paula Brito, Joaquim Costa, and Donato Malerba, editors, *Proceedings of the ECML2000 workshop on Dealing with Structured Data in Machine Learning and Statistics*, pages 33–43, Barcelona, Spain, May 2000.
4. http://labe.felk.cvut.cz/~zelezny/howmanyfunctions.pl.
5. http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph.html.
6. Kristian Kersting and Luc De Raedt. Bayesian logic programs. In J. Cussens and A. Frisch, editors, *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, pages 138–155, 2000.
7. N. Lavrač, S. Džeroski, and I. Bratko. Handling imperfect data in inductive logic programming. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 48–64. IOS Press, 1996.
8. Eric McCreath and Arun Sharma. ILP with noise and fixed example size: A bayesian approach. In *IJCAI*, pages 1310–1315, 1997.
9. R.J. Mooney and M.E. Califf. Induction of first–order decision lists: Results on learning the past tense of English verbs. *Journal of Artificial Intelligence Research*, 3:1–24, 1995.
10. R.J. Mooney and M.E. Califf. Induction of first-order decision lists: Results on learning the past tense of English verbs. In L. De Raedt, editor, *Proceedings of the 5th International Workshop on Inductive Logic Programming*, pages 145–146. Department of Computer Science, Katholieke Universiteit Leuven, 1995.
11. S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.
12. S. Muggleton. Learning from positive data. In S. Muggleton, editor, *Proceedings of the 6th International Workshop on Inductive Logic Programming*, volume 1314 of *Lecture Notes in Artificial Intelligence*, pages 358–376. Springer-Verlag, 1996.
13. S. Muggleton and M. Bain. Analogical prediction. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 234–244. Springer-Verlag, 1999.
14. S. Muggleton and C.D. Page. A learnability model for universal representations. In S. Wrobel, editor, *Proceedings of the 4th International Workshop on Inductive Logic Programming*, volume 237 of *GMD-Studien*, pages 139–160. Gesellschaft für Mathematik und Datenverarbeitung MBH, 1994.
15. S-H. Nienhuys-Cheng. Distance between herbrand interpretations: A measure for approximations to a target concept. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297 of *Lecture Notes in Artificial Intelligence*, pages 213–226. Springer-Verlag, 1997.
16. J. Ramon and L. De Raedt. Instance based function learning. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 268–278. Springer-Verlag, 1999.