Maximum Entropy Modeling with Clausal Constraints

Luc Dehaspe

Katholieke Universiteit Leuven, Department of Computer Science,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
email : Luc.Dehaspe@cs.kuleuven.ac.be
fax : ++ 32 16 32 79 96; telephone : ++ 32 16 32 75 67

Abstract. We present the learning system MACCENT which addresses the novel task of stochastic MAximum ENTropy modeling with Clausal Constraints. Maximum Entropy method is a Bayesian method based on the principle that the target stochastic model should be as uniform as possible, subject to known constraints. MACCENT incorporates clausal constraints that are based on the evaluation of Prolog clauses in examples represented as Prolog programs. We build on an existing maximum-likelihood approach to maximum entropy modeling, which we upgrade along two dimensions: (1) MACCENT can handle larger search spaces, due to a partial ordering defined on the space of clausal constraints, and (2) uses a richer first-order logic format. In comparison with other inductive logic programming systems, MACCENT seems to be the first that explicitly constructs a conditional probability distribution $p(C|I)$ based on an empirical distribution $\tilde{p}(C|I)$ (where $p(C|I)$ ($\tilde{p}(C|I)$) equals the induced (observed) probability of an instance I belonging to a class C). First experiments indicate MACCENT may be useful for prediction, and for classification in cases where the induced model should be combined with other stochastic information sources.

Keywords: stochastic modeling, inductive logic programming

1 Introduction

Bayesian probability theory permits one to use prior knowledge for making rational inferences about the behavior of a random process. *Maximum Entropy* [15] method is a Bayesian method based on the principle that the target stochastic model should be as uniform as possible, subject to constraints extracted from the training distribution. This principle is not new, it can in fact be traced back to Laplace, and beyond, but relatively recent developments in computer science have boosted its popularity and have enabled its application to real-world problems, mainly in physics (cf. [14]). As a starting point for our research we used the paper by Berger, Della Pietra, and Della Pietra [1], which presents a maximum-likelihood approach to the construction of maximum entropy models of natural language.

We have combined the statistical approach of [1] with techniques from the field of inductive logic programming to build the relational learner MACCENT

that addresses the task of MAximum ENTropy modeling with Clausal Constraints. The adjective "clausal" here refers to the fact that in MACCENT the constraints, which are the building blocks of maximum entropy modeling, are based on the evaluation of Prolog clauses in examples represented as Prolog programs. The integration of probabilistic methods with inductive logic programming has recently become a popular research topic (cf. [5, 16, 19, 21, 22]), but, to the best of our knowledge, MACCENT is the first inductive logic programming algorithm that explicitly constructs a conditional probability distribution $p(C|I)$ based on an empirical distribution $\tilde{p}(C|I)$ (where $p(C|I)$ ($\tilde{p}(C|I)$) gives the induced (observed) probability of an instance I belonging to a class C).

We start our discussion with a brief introduction to the logical and mathematical foundations (Section 2). In the two next parts we then present the task addressed by MACCENT (Section 3), and the algorithm itself (Section 4). Finally, we report on classification and prediction experiments with a prototypical implementation of MACCENT (Section 5), and conclude (Section 6). Though throughout the paper we have restricted the logical notation to a minimum, we do assume some familiarity with the programming language Prolog [3].

2 Logical and statistical background

2.1 Logical paradigm: learning from interpretations

animal	has_covering	has_legs	habitat	homeothermic	class
dog	hair	yes	land	yes	mammal
dolphin	none	no	water	yes	mammal
trout	scales	no	water	no	fish
shark	none	no	water	no	fish
herring	scales	no	water	no	fish
eagle	feathers	yes	air	yes	bird
penguin	feathers	yes	water	yes	bird
lizard	scales	yes	land	no	reptile
snake	scales	no	land	no	reptile
turtle	scales	yes	land	no	reptile

Table 1. The animals domain

Consider the data in Table 1, which presents instances of four classes of animals. Traditional attribute-value learners represent each animal and candidate classification rule as a set of attribute-value pairs, such that simple subset test suffices in order to verify if a rule covers an instance. Alternatively, one can store the information available about a single animal in a Prolog knowledge base, encode candidate rules as Prolog queries, and apply the Prolog execution mechanism to verify whether or not a rule covers an instance. If the query succeeds, the

rule covers the instance. If the query fails[1], the rule does not cover the instance. For example, the logical conjunction $has\_legs \wedge habitat(land)$ covers instance *dog* because the Prolog query $\leftarrow has\_legs \wedge habitat(land)$ succeeds in the Prolog program dog:

```
animal(dog). has_covering(hair). has_legs. habitat(land).
homeothermic. class(mammal).
```

Notice that the attribute-value and Prolog representations are equivalent in standard propositional domains. In Prolog notation however instances can be Prolog programs of deliberate size and complexity, rather than fixed-length vectors of attribute-value pairs. As a consequence, background knowledge can be added in a natural way as Prolog code shared by all instances. For example, with Prolog clause $kids\_love\_it \leftarrow homeothermic \wedge habitat(water)$ added as background knowledge to Table 1, rule $has\_covering(none) \wedge kids\_love\_it$ exclusively covers instance dolphin.

The above alternative and flexible Prolog based representation and verification method fits in the *learning from interpretations* paradigm, introduced by [9] and related to other inductive logic programming settings in [7]. This paradigm provides a convenient environment for upgrading attribute-value learners to first order logic, witness ICL [10] and TILDE [2] which are first order equivalents of respectively CN2 and C4.5.

2.2 Statistical paradigm: maximum entropy

Let us reconsider the data in Table 1, and interpret them as as a sample of an expert biologist's decisions concerning the class of animals. The observed behavior of the biologist can be summarized as an empirical conditional distribution $\tilde{p}(C|I)$ that, given the description of an instance in I in the first five columns, assigns probability 1 to the class specified in the last column, and 0 to all other classes. Sample $\tilde{p}$ can be used to support the construction of a stochastic model p to predict the behavior of the biologist. Given an instance I, this target model will assign to each class C_j an estimate $p(C_j|I)$ of the conditional probability that the biologist would classify I as C_j.

"Relevant" facts (statistics) extracted from the empirical distribution $\tilde{p}$ are the building blocks of the target distribution p. For example, with sample Table 1, $\tilde{p}$ assigns conditional probability 1 to exactly four classes C. If we are convinced this fact captures relevant information about the sample, we can construct our model p such that, for all instances I it meets the following constraint:

$$p(mammal|I) + p(fish|I) + p(bird|I) + p(reptile|I) = 1 \qquad (1)$$

There is of course a whole family of models that meet constraint (1). However, some members of this family seem more justified than others, given the limited

[1] Termination can be guaranteed by employing a depth-bound on the depth of the proof tree.

empirical evidence (1) we have gathered so far. More specifically, the model p_{mfbr} that assigns equal 0.25 probability to all four classes seems intuitively more reasonable than a model p_{mf} that grants probability 0.5 to *mammal* and *fish*, and 0 to the other classes. Though model p_{mf} agrees with what we know, it contains information for which we have – as yet – no evidence. A stochastic model that avoids such bold assumptions, should be as "uniform" as possible, subject to the known constraints.

In the maximum entropy paradigm [15] conditional entropy

$$H(p) \equiv -\frac{1}{N}\sum_{C,I} p(C|I)\log p(C|I) \tag{2}$$

is used to measure uniformity[2], with N the number of instances in the sample. For example, if we apply the conditional entropy metric to the two alternative models introduced in the previous paragraph for the animals domain, we obtain:

$$H(p_{mfbr}) = -\frac{1}{10} * 10 * 4 * 0.25 * \log(0.25) = -\log(0.25) \simeq 1.39$$
$$H(p_{mf}) = -\frac{1}{10} * 10 * 2 * 0.50 * \log(0.50) = -\log(0.50) \simeq 0.69$$

The *Maximum Entropy Principle* then states that, from the family of conditional probability distributions allowed by an empirically compiled set of constraints, we should elect the model with maximum conditional entropy $H(p)$. In the example, $H(p_{mfbr}) > H(p_{mf})$, therefore, in accordance with our intuitions, p_{mfbr} is preferable.

3 Maximum entropy modeling with clausal constraints

3.1 The building blocks: clausal constraints

As indicated in the previous section, the building blocks of a maximum entropy model are constraints such as Equation (1). MACCENT incorporates constraints based on *boolean clausal indicator functions* $f(I,C)$ with general form:

$$f_{j,k}(I,C) = \begin{cases} 1 & \text{if } C = C_j\text{, and} \\ & \quad \text{Prolog query} \leftarrow Q_k \text{ succeeds in instance } I \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

A clausal[3] indicator function $f_{j,k}$ gives rise to a *clausal constraint* of the form

$$CC_{j,k} : p(f_{j,k}) = \tilde{p}(f_{j,k}) \tag{4}$$

[2] Throughout the paper log denotes natural logarithm.

[3] Recall that Prolog queries are a special type of Prolog *clauses* with `false` in the conclusion part.

where $p(f_{j,k})$ is the *expected value* of $f_{j,k}(I,C)$ in the target distribution, and $\tilde{p}(f_{j,k})$ is the expected value of $f_{j,k}(I,C)$ in the empirical distribution. For any $f_{j,k}(I,C)$, the calculation of its expected value with respect to a conditional (target or empirical) distribution proceeds as follows.

First, we count the number of instances in which Prolog query $\leftarrow Q_k$ succeeds. We can obtain this number by taking the sum over all instances $\sum_I f_{j,k}(I,C_j)$. For example, given the data in Table 1, and

$$C_1 = fish$$
$$Q_1 = \neg has\_legs \wedge habitat(water),$$

such that,

$$f_{1,1}(I,C) = \begin{cases} 1 & \text{if } C = fish, \text{ and} \\ & \quad \leftarrow \neg has\_legs \wedge habitat(water) \text{ succeeds in } I \\ 0 & \text{otherwise} \end{cases}$$

we find that $\leftarrow Q_1$ succeeds in four instances. Accordingly, the set of $(instance, class)$ tuples for which $f_{1,1}$ evaluates to one is

$$\{(dolphin, fish), (trout, fish), (shark, fish), (herring, fish)\}.$$

All other combinations evaluate to zero, either because the class is not `fish`, or because query $\leftarrow \neg has\_legs \wedge habitat(water)$ does not succeed in the instance. Hence,

$$\#\{\text{instances in which} \leftarrow Q_1 \text{ succeeds}\} = \sum_I f_{1,1}(I,C_1) = 4.$$

Next, we normalize the summation with constant N, the number of instances in the sample, for example $1/10 \sum_I f_{1,1}(I,C_1) = 0.4$. Finally, to obtain the expected value $p(f_{j,k})$ of $f_{j,k}(I,C)$ in model p, we multiply $f_{j,k}(I,C_j)$ by $p(C_j|I)$. Thus, instead of adding 1 for each instance in which $\leftarrow Q_k$ succeeds, we add the number between 0 and 1 model p assigns to the conditional probability of class C_j given instance I. For the expected value of $f_{j,k}(I,C)$ in empirical distribution $\tilde{p}$, and in target distribution p this yields the equations

$$\tilde{p}(f_{j,k}) \equiv \frac{1}{N} \sum_I \tilde{p}(C_j|I) f_{j,k}(I,C_j) \tag{5}$$

$$p(f_{j,k}) \equiv \frac{1}{N} \sum_I p(C_j|I) f_{j,k}(I,C_j) \tag{6}$$

For example, reconsider the uniform conditional distribution p_{mfbr} introduced above. We know that $p_{mfbr}(fish|I) = 0.25$ for all ten instances in Table 1. As $f_{1,1}(I, fish)$ evaluates to one for the four instances dolphin, trout, shark, herring, and zero for all other instances, the expected value $p_{mfbr}(f_{1,1})$ of indicator function $f_{1,1}(I,C)$ in model p_{mfbr} is

$$p_{mfbr}(f_{1,1}) = \frac{1}{10} * (0.25 + 0.25 + 0.25 + 0.25) = 0.1.$$

On the other hand, in the sample we find that $\tilde{p}(fish|I) = 1$ for the three instances trout, shark, and herring, and $\tilde{p}(fish|I) = 0$ for all other instances, including dolphin. Hence,

$$\tilde{p}(f_{1,1}) = \frac{1}{10} * (1 + 1 + 1) = 0.3.$$

Function $f_{1,1}$, through its expected value, thus reveals a discrepancy between the empirical distribution $\tilde{p}$ and target distribution p_{mfbr}:

$$p_{mfbr}(f_{1,1}) + 0.2 = \tilde{p}(f_{1,1}).$$

This discrepancy disappears if we add the clausal constraint

$$CC_{1,1} : p_{mfbr}(f_{1,1}) = \tilde{p}(f_{1,1}) = 0.3$$

to our initial constraint Equation (1), and construct a new target model that maximizes conditional entropy subject to both constraints.

As a second example, consider $f_{2,2}$ defined by:

$$C_2 = reptile$$
$$Q_2 = \neg has\_covering(hair) \wedge \neg has\_legs$$

The expected values are $p_{mfbr}(f_{2,2}) = 0.125$, and $\tilde{p}(f_{2,2}) = 0.1$. Notice that with respect to the sample the expected value $p_{mfbr}(f_{2,2})$ is 0.025 units too high, whereas $p_{mfbr}(f_{1,1})$ is 0.2 units too low. Later, where we discuss the construction of the new target model, we will see that a model that incorporates $CC_{1,1}$ assigns a positive weight to $f_{1,1}$, whereas a model that incorporates $CC_{2,2}$ assigns a negative weight to $f_{2,2}$. The intuition behind these weights is that Q_1 is a better indication for *fish*, and Q_2 a worse indication for *reptile* than is assumed in model p_{mfbr}.

3.2 Problem specification

Now that we have described the building blocks, we can formulate the task of *maximum entropy modeling with clausal constraints*:

- *Given*: A set $\mathcal{I}$ of N training instances, represented as Prolog programs;
- *Given*: An empirical conditional distribution $\tilde{p}(C|I)$, with $I \in \mathcal{I}$;
- *Given*: A language $\mathcal{L}$ of clausal constraints $CC_{j,k} : p(f_{j,k}) = \tilde{p}(f_{j,k})$;
- *Find*: An optimal set $\mathcal{CC} \subseteq \mathcal{L}$ of clausal constraints;
- *Find*: A stochastic model $p^{\mathcal{CC}}(C|I)$ that maximizes conditional entropy $H(p)$ subject to the clausal constraints in $\mathcal{CC}$.

To restate, algorithms that address this task receive as input training instances, an empirical conditional distribution that relates training instances to classes, and a set of clausal constraints. In response, they must (1) select an "optimal" subset of clausal constraints, and (2) with these constraints build a model for use in classifying new instances.

3.3 Model selection

We come back to the selection of $\mathcal{CC}$ later, and first concentrate on the modeling subtask. Given set $\mathcal{CC} = \{CC_{j_1,k_1}, CC_{j_2,k_2}, \ldots, CC_{j_M,k_M}\}$ of M clausal constraints, the method of Lagrange multipliers from the theory of constrained optimization produces the model p that maximizes conditional entropy $H(p)$. As discussed thoroughly in [12, 1], the model in question has the parametric form

$$p_\Lambda(C|I) = \frac{1}{Z_\Lambda(I)} \exp\left(\sum_{m=1}^{M} \lambda_m f_{j_m,k_m}(I,C)\right). \tag{7}$$

In parameter set $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_M\}$, each of the M parameters (Lagrange multipliers) λ_m corresponds to a clausal constraint $CC_{j_m,k_m} \in \mathcal{CC}$ based on clausal indicator function f_{j_m,k_m}. The optimal parameter set Λ can be determined by maximizing the log-likelihood $L_{\tilde{p}}(p_\Lambda)$ of the empirical distribution $\tilde{p}$, which is defined by

$$L_{\tilde{p}}(p_\Lambda) = -\frac{1}{N} \sum_I \log Z_\Lambda(I) + \sum_{m=1}^{M} \lambda_m \tilde{p}(f_{j_m,k_m}) \tag{8}$$

where $Z_\Lambda(I)$ is a normalization constant:

$$Z_\Lambda(I) = \sum_C \exp\left(\sum_{m=1}^{M} \lambda_m f_{j_m,k_m}(I,C)\right) \tag{9}$$

If we apply these equations to our running example, we obtain, after reduction,

$$L_{\tilde{p}}(p_{\{\lambda_1\}}) = -0.4 * \log(e^{\lambda_1} + 3) - 0.6 * \log(4) + 0.3 * \lambda_1$$
$$L_{\tilde{p}}(p_{\{\lambda_2\}}) = -0.5 * \log(e^{\lambda_2} + 3) - 0.5 * \log(4) + 0.1 * \lambda_2$$

where λ_1 and λ_2 correspond to $f_{1,1}$ and $f_{2,2}$. The graph of these two log-likelihood functions is plotted in Figure 1. On the X axis of this graph, we find the λ values, on the Y axis the log-likelihood of the Table 1 training set. The graph shows that with $\mathcal{CC} = \{CC_{1,1}\}$, the optimal parameter set $\{\lambda_1\}$ is obtained with a positive value close to two for λ_1. This agrees with the earlier finding that in the uniform distribution, $p_{mfbr}(f_{1,1}) < \tilde{p}(f_{1,1})$. The intuition here is that the appropriateness of $CC_{1,1}$ has been underestimated in p, and a positive weight λ_1 is required to correct this. The opposite holds for $\mathcal{CC} = \{CC_{2,2}\}$, where the optimum for λ_2 is reached in a negative value close to zero. Notice both graphs intersect on the Y axis. The point of intersection $(0, -1.39)$ corresponds to the case where no constraints apart from Equation (1) are added (i.e. the weights are set to zero).

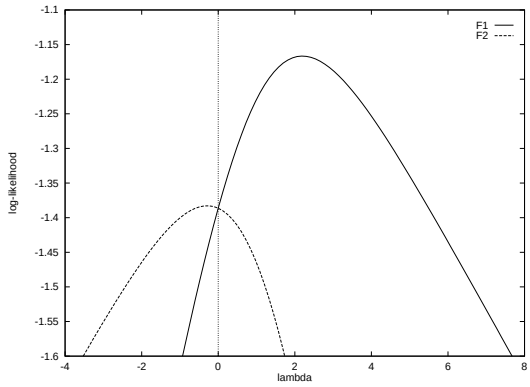

Fig. 1. The log-likelihood functions $F1 = L_{\tilde{p}}(p_{\{\lambda_1\}})$ and $F2 = L_{\tilde{p}}(p_{\{\lambda_2\}})$.

3.4 Feature selection

Let us now return to the selection of an optimal set $\mathcal{CC} \subset \mathcal{L}$. For the time being we assume an exhaustive search through $2^{\mathcal{L}}$ and focus on an evaluation function for $\mathcal{S} \subset \mathcal{L}$. Notice that maximum entropy and maximum likelihood are dual problems and that, of all models of parametric form (7), the maximum entropy model is the one that maximizes the likelihood of the sample. As might be clear from the above discussion of Figure 1, this maximum is function of the set $\mathcal{S}$ of clausal constraints. We are interested in the set $\mathcal{S}$ that produces the highest likelihood of the training sample $\tilde{p}$. Formally, if we let $p^{\mathcal{S}}$ denote the maximum entropy model obtained with a set $\mathcal{S}$ of clausal constraints then

$$\mathcal{CC} \equiv \underset{\mathcal{S} \subset \mathcal{L}}{\operatorname{argmax}}\, L_{\tilde{p}}(p^{\mathcal{S}}) \tag{10}$$

For an illustration of this evaluation function consider again Figure 1. The graph shows that with $\{CC_1\}$ we can obtain log-likelihood around -1.17, whereas with $\{CC_2\}$ we only reach -1.38. Therefore, of these two sets, $\{CC_1\}$ is to be preferred. In Table 2 the conditional probabilities are shown for the two models $p^{\{CC_1\}}$ and $p^{\{CC_2\}}$. Notice $p^{\{CC_1\}}$ is indeed closer to Table 1 than $p^{\{CC_2\}}$.

4 The MACCENT Algorithm

In this section we present the learning system MACCENT which addresses the task of MAximum ENTropy modeling with Clausal Constraints. The task as described in the previous section does not directly translate to a tractable algorithm. Usually, with millions of clausal constraints in $\mathcal{L}$, it is prohibitive to inspect all subsets of $\mathcal{L}$ in search for the optimal one. In our description of MACCENT we will gradually abandon the request for optimality of $\mathcal{CC}$. We first discuss a greedy search method and an approximate gain evaluation function, both due

	$p = p^{\{CC_1\}}$				$p = p^{\{CC_2\}}$			
animal	$p(m\|I)$	$p(f\|I)$	$p(b\|I)$	$p(r\|I)$	$p(m\|I)$	$p(f\|I)$	$p(b\|I)$	$p(r\|I)$
dog	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
dolphin	0.08	0.76	0.08	0.08	0.26	0.26	0.26	0.22
trout	0.08	0.76	0.08	0.08	0.26	0.26	0.26	0.22
shark	0.25	0.25	0.25	0.25	0.26	0.26	0.26	0.22
herring	0.08	0.76	0.08	0.08	0.26	0.26	0.26	0.22
eagle	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
penguin	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
lizard	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
snake	0.25	0.25	0.25	0.25	0.26	0.26	0.26	0.22
turtle	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25

Table 2. Two target distributions for the animals domain

to [1]. In addition, we then propose a structure on $\mathcal{L}$ that one can use to organize the search process.

As shown in Algorithm 1, MACCENT carries out a greedy search, adjoining clausal constraints one at a time, using an evaluation function to select the next constraint at each step. Given a set $\mathcal{S}$ of constraints collected so far, a natural evaluation function for a candidate clausal constraint $\widehat{cc}$ returns the gain $\triangle L_{\tilde{p}}(\mathcal{S}, \widehat{cc})$ in the log-likelihood of the training data obtained by that adding $\widehat{cc}$ to $\mathcal{S}$:

$$\triangle L_{\tilde{p}}(\mathcal{S}, \widehat{cc}) \equiv L_{\tilde{p}}(p^{\mathcal{S}\cup\widehat{cc}}) - L_{\tilde{p}}(p^{\mathcal{S}}). \tag{11}$$

As $L_{\tilde{p}}(p^{\mathcal{S}})$ remains constant, this comes down to adding at each step the constraint CC that maximizes $L_{\tilde{p}}(p^{\mathcal{S}\cup CC})$. We have thus reduced the selection task to the discovery of the best $CC \in \mathcal{L}$. For instance, consider again our running example and the graph in Figure 1. Knowing that $L_{\tilde{p}}(p^{\emptyset}) = -\log(4) \simeq -1.39$, we can compute that the gain obtained by adding $CC_{1,1}$ and $CC_{2,2}$ is something round 0.22 and 0.01 respectively. Therefore, $CC_{1,1}$ is a better initial choice.

Algorithm 1 : MACCENT

Inputs: Language $\mathcal{L}$; empirical distribution $\tilde{p}(C|I)$
Outputs: Set $\mathcal{CC}$ of clausal constraints; model $p^{\mathcal{CC}}$

1. Start with $\mathcal{CC} = \emptyset$; thus $p^{\mathcal{CC}}$ is uniform
2. Select next clausal constraint $\widehat{cc}$ from $\mathcal{L}$ using Algorithm 2
3. Check the termination condition
4. Adjoin $\widehat{cc}$ to $\mathcal{CC}$
5. Compute $p^{\mathcal{CC}}$ using Equation (7)
6. Go to step 2

Still, given the "old" set $\mathcal{S} = \{CC_1, \ldots, CC_n\}$, log-likelihood gain requires

for each candidate $\widehat{cc} \in \mathcal{L}$ the computation of the set of parameters $\Lambda = \{\lambda_1, \ldots, \lambda_n, \alpha\}$ that maximizes $L_{\tilde{p}}(p_\Lambda)$, where α is the parameter that corresponds to $\widehat{cc}$. Various numerical methods have been applied to this problem, including problem specific algorithms such as BROWN [4] or ITERATIVE SCALING [6]. However, none of these methods allow for a sufficiently efficient computation of gain $\triangle L_{\tilde{p}}(\mathcal{S}, \widehat{cc})$. As a remedy, [1] introduce the notion of *approximate gain* which no longer requires the computation of $n+1$ parameters. The "old" parameter set $\{\lambda_1, \ldots, \lambda_n\}$ is retained, and the approximation is made that the addition of constraint $\widehat{cc}$ affects only parameter α. Given candidate constraint $\widehat{cc}$ derived from indicator function $\hat{f}$, Equation (8) then reduces to

$$G_{\mathcal{S},\hat{f}}(\alpha) = -\frac{1}{N} \sum_{I,C} \log \left(p^{\mathcal{S}}(C|I) e^{\alpha \hat{f}(I,C)} \right) + \alpha \tilde{p}(\hat{f}) \tag{12}$$

Equation (12) in turn simplifies the computation of the *approximate gain*

$$\sim \triangle L_{\tilde{p}}(\mathcal{S}, \widehat{cc}) \equiv \max_{\alpha} G_{\mathcal{S},\hat{f}}(\alpha) \tag{13}$$

to a one-dimensional optimization problem which can be solved, for instance, with Newton's method.

Though [1] has shown one can implement a high-performance algorithm on the basis of the above approximations, we still assume an exhaustive search through a flat, unstructured space $\mathcal{L}$ of clausal constraints. To further speed up the search, and enable the exploration of bigger search spaces, MACCENT extends these approximations with classical machine learning techniques. The key observation here is that generality imposes a partial ordering on $\mathcal{L}$:

> $CC_{j,k}$ is more general than $CC_{j',k'}$ if and only if
> $C_j \leftarrow Q_k$ is logically more general than $C_{j'} \leftarrow Q_{k'}$.

One can specify a lattice for clausal constraints with a set of most general descriptions $\{CC_{j,k} \in \mathcal{L} | Q_k = \mathit{true}\}$ at the top, and a specialisation operator for top-down exploration of this lattice. In MACCENT we handle both tasks, language specification and specialisation, with the declarative language bias formalism DLAB [11], which is based on θ subsumption (cf. [20]).

As shown in Algorithm 2, MACCENT exploits the lattice structure of $\mathcal{L}$ to carry out a general-to-specific heuristic beam search for clausal constraints with satisfactory, rather than optimal, approximate gain. At each level of its search, MACCENT considers all specialisations of clausal constraints in the beam, and if necessary updates the current best solution.

We still have to specify a stopping criterion (Algorithm 1, step 3) and a procedure for pruning *Beam* (Algorithm 2, step 5). For our stopping criterion we use Equation (8) to calculate the log-likelihood $L_{\bar{p}}(p^{\mathcal{CC} \cup \widehat{cc}})$ of a separate empirical distribution $\bar{p}$. If the log-likelihood of this witheld portion of the training set decreases[4] with the addition of $\widehat{cc}$, this is taken as an indication of overtraining, and MACCENT terminates.

[4] A delay is built in to allow recovery from a temporary decrease.

Algorithm 2 *:* Select next clausal constraint

Inputs: Language $\mathcal{L}$; empirical distribution $\tilde{p}(C|I)$; current set $\mathcal{CC}$
Outputs: Clausal constraint $\widehat{cc}$

1. Initialize current best approximate gain $Best := -1$
2. Initialize $Beam := \{CC_{j,k} \in \mathcal{L} | Q_k = true\}$
3. For each cc in $Beam$, and for each specialisation $cc' \in \mathcal{L}$ of cc:
   (a) Compute $Gain = \triangle L_{\tilde{p}}(\mathcal{CC}, cc')$ using Equation (13),
   (b) If $Gain > Best$ then update $Best := Gain$ and $\widehat{cc} := cc'$,
   (c) Add cc' to $Beam$
4. Prune $Beam$
5. If $Beam$ is not empty, go to step 3

Given a beam size BS, the default strategy for pruning $Beam$ is to rank the clausal constraints cc' in $Beam$ according to their $Gain$ and keep at most BS of the best scoring constraints. However, in cases where the input empirical distribution assigns only probabilities 1 and 0, as for instance in Table 1, we can apply a more sophisticated heuristic. Thereto, we define the notion of *potential approximate gain* as an upper bound to the maximum $Gain$ that can be obtained via further specialisation of cc'. We cannot go into details here, but under the circumstances mentioned, potential approximate gain can easily be computed. One has to select the instances for which $\tilde{p}(C'|I) = 1$, compute the approximate gain counting only this subset, do the same for the complementary subset, and take the maximum of both computations. The potential approximate gain upper bound allows a branch-and-bound search in which clausal constraints are pruned that after specialisation can never produce an approximate gain above the current best. Apart from that, potential approximate gain can also be used in combination with actual approximate gain to better rank the constraints in $Beam$[5].

5 Experiments

In this section we report on some first classification and prediction experiments with a prototypical Prolog implementation of MACCENT. Both classification and numeric prediction could in fact be considered as special cases of the task of learning from a conditional probability distribution $\tilde{p}(C|I)$: with classification $\tilde{p}(C|I)$ only takes values 0 and 1; with numeric prediction there are only two classes C. We would like to stress that the full power of MACCENT should best be demonstrated on the general case of multi-class datasets where $\tilde{p}$ can take any real value between 0 and 1 (e.g. distribution of votes for more than two

[5] MACCENT mainly employs the average of the actual and the potential approximate gain.

candidates in different districts). At present however, such data do not seem to be readily available.

First, we round off the running example with the application of MACCENT to the multi-class animals domain. With the hold-out stopping criterion suppressed, MACCENT found 13 constraints before the log-likelihood of Table 1 was reduced to 0. The components C_j and Q_k of the indicator functions $f_{j,k}$ on which these clausal constraints are based are shown, in order of discovery, in Table 3. The approximate gain, best λ, and log-likelihood of the sample are listed in columns 3, 4, and 5 respectively.

C_j	Q_k	gain	λ	llh
$reptile$	$has\_covering(scales) \wedge habitat(land)$	0.266	1.59	-1.114
$fish$	$\neg has\_legs \wedge \neg homeothermic$	0.227	1.56	-0.883
$bird$	$has\_covering(feathers)$	0.177	1.59	-0.702
$mammal$	$\neg has\_covering(feathers) \wedge homeothermic$	0.177	1.59	-0.520
$fish$	$\neg has\_covering(none) \wedge \neg habitat(water)$	0.107	-36.98	-0.414
$bird$	$\neg has\_covering(feathers)$	0.118	-36.97	-0.296
$mammal$	$\neg homeothermic$	0.104	-36.86	-0.192
$reptile$	$true$	0.092	-1.56	-0.099
$fish$	$homeothermic$	0.033	-37.10	-0.066
$mammal$	$has\_covering(feathers)$	0.036	-36.09	-0.030
$reptile$	$true$	0.022	-1.46	-0.007
$reptile$	$\neg habitat(land)$	0.005	-37.21	-0.002
$reptile$	$\neg has\_covering(scales)$	0.002	-35.89	0

Table 3. Output of MACCENT on the animals domain

5.1 Classification

When compared to other inductive logic programming classifiers, MACCENT has the advantage that it induces a conditional probability distribution that can directly be combined with other stochastic information sources. To illustrate this point we conducted a relatively small experiment with natural language data extracted from the Wall Street journal corpus [18].

In this corpus the words are tagged with so-called Parts-of-Speech (POS). We grouped the tags into 16 classes, the most important of which are noun, adjective, verb, punctuation, determiner, pronoun, adverb. Next, based on frequency counts in the corpus, we set up a dictionary that, given a word w assigns to each class POS a lexical probability $p_{dict}(POS|w)$. For instance, $p_{dict}(adjective|cross) = 0.03, p_{dict}(noun|cross) = 0.69, p_{dict}(verb|cross) = 0.28$. Then we selected from the corpus the first 1000 occurrences of words, such as *cross*, that are ambiguous between exactly the three classes noun, adjective, and verb. We stored each word, with its class label, and the tagged left and right context in a separate instance. As background knowledge we specified Prolog

predicates $left(Index, Limit, POS)$, and $right(Index, Limit, POS)$ that check whether a certain tag POS appears in the context of a word within distance $Limit$, where $Index$ is the position the word in the sentence. In the language $\mathcal{L}$, with a size of order 10^{15}, we allowed combinations of these two predicates, with $Limit$ taking values 1, 2, 3, 4, 5, and *unlimited*, and with POS instantiated to one of the 16 morphological categories. Finally, we ran MACCENT on this 3-class problem to construct a probability distribution $p_{context}(POS|w)$ based on the context of the word.

Classifying words into the class with highest probability as predicted by the model we ran a 10-fold crossvalidation based on a random partition over 1000 instances, and each time measured the performance in terms of classification accuracy. The model $p_{context}(POS|w)$ induced by MACCENT achieved an average accuracy of $70.4\%(\pm 4.8)$. The model $p_{dict}(POS|w)$ based on the dictionary had a much better score of $81.2(\pm 3.0)$. The best score however was obtained from the combination of the two models $p_{context}(POS|w) * p_{dict}(POS|w)$ which achieved an accuracy of $85.4(\pm 3.0)$.

Though MACCENT by itself is likely to perform better in the presence of more sophisticated background knowledge, this initial experiment with POS-tagging demonstrates the (poor) output of MACCENT can successfully be combined with other stochastic information sources. For a more mature application of maximum entropy method to the general POS-tagging problem, we refer to [23].

5.2 Numeric prediction

Recall that one of the inputs to MACCENT is an empirical distribution $\tilde{p}(C|I)$. In the previous examples, $\tilde{p}$ took only the values 0 and 1. Multi-class datasets where $\tilde{p}$ can take any real value between 0 and 1, do not seem to be readily available. There are however enough tasks available that involve regression, the prediction of a numeric value. These values can always be projected onto a scale from 0 to 1. With some abuse of the intuitions behind a conditional probability model, MACCENT can now interpret the rescaled values as estimates of the probability that the example belongs the single "positive" class. The inverse "unscaling" projection allows one to calculate predictions from the induced model. Like FORS [16], which incorporates linear regression techniques, SRT [17], which builds structural regression trees, and C0.5 [8] which builds first order clusters, MACCENT can then perform first order regression from positive data only.

We applied this technique to two benchmark domains well known in the inductive logic programming community: mutagenicity [25], and mesh design [13].

In the first domain the task is to predict the log mutagenicity of molecules on the basis of structural information on the compounds. We used the full set of available background predicates and defined a language $\mathcal{L}$ with size order 10^{24}. With a 10-fold crossvalidation we obtained an average standard error of $1.38(\pm 0.18)$, with log-mutagenicity ranging from -3 to 5.39.

The second domain requires a prediction about the coarseness of a mesh model used to approximate physical structures. For every edge of the structure one has to decide on the number, ranging from 1 to 17, of subedges on the basis

of properties of the edge and of related edges. We focused our experiments on the 5 structure (A,B,C,D,E) dataset. With a language $\mathcal{L}$, with size of order 10^7, we tried both a 5-fold leave-one-structure-out (mesh-xv5) and a random 10-fold crossvalidation (mesh-xv10). We calculated standard error of the estimate, and accuracy. For accuracy we rounded MACCENT's predictions to the nearest class. Table 4 summarizes results for MACCENT and ICL [10].

	Accuracy		Standard error	
Algorithm	mesh-xv5	mesh-xv10	mesh-xv5	mesh-xv10
ICL	0.50	0.70($\pm$0.05)	2.82($\pm$1.19)	1.84($\pm$1.03)
MACCENT	0.25	0.44($\pm$0.11)	2.83($\pm$0.42)	1.54($\pm$0.33)

Table 4. Performances on the mesh domain

The above table shows roughly the same picture for mesh-xv5 and mesh-xv10: MACCENT scores a lot worse on accuracy, but comparable on standard error. We take this as an indication that the scaling technique indeed makes sense, and that MACCENT can be considered for solving regression tasks.

6 Conclusion and future work

We have presented a framework for combining maximum entropy modeling with inductive logic programming techniques, and a number of approximations that led to the definition and implementation of the MACCENT algorithm. First experiments indicate the first order statistical modeling algorithm MACCENT may be useful for prediction, and for classification in cases where the induced model should be combined with other stochastic information sources.

Further experiments are required on more appropriate multi-class datasets where empirical conditional probability $\tilde{p}$ can take any real value between 0 and 1. For these experiments we intend to make use of the Maximum Entropy Modeling Toolkit [24] to improve our initial prototypical implementation. Future theoretical research should clarify the relationship between MACCENT and (more) standard inductive logic programming approaches to classification and regression.

Acknowledgements

This research is sponsored by the ESPRIT IV project no. 20237 on Inductive Logic Programming II (ILP^2). I would like to thank Walter Daelemans for his crucial pointer to the Maximum Entropy literature; Adam Berger, Vincent Della Pietra, Stephen Della Pietra, and Adwait Ratnaparkhi for their inspiring papers; Bojan Dolsak and Ashwin Srinivasan for the mesh and mutagenicity data; Hendrik Blockeel and Dirk Roose for mathematical assistance; Bart Demoen and

Wim Van Laer for their contributions to the implementation of MACCENT; and finally, Luc De Raedt and Maurice Bruynooghe for the stimulating (logical) context for this research.

References

1. A.L. Berger, V.J. Della Pietra, and S.A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.
2. H. Blockeel and L. De Raedt. Experiments with top-down induction of logical decision trees. Technical Report CW 247, Dept. of Computer Science, K.U.Leuven, January 1997. Also in Periodic Progress Report ESPRIT Project ILP2, January 1997. `http://www.cs.kuleuven.ac.be/publicaties/rapporten/CW1997.html`.
3. I. Bratko. *Prolog Programming for Artificial Intelligence*. Addison-Wesley, 1986.
4. D. Brown. A note on approximations to discrete probability distributions. *Information and Control*, 2:386–392, 1959.
5. J. Cussens. Baysian inductive logic programming with explicit probabilistic bias. Technical Report PRG-TR-24-96, Oxford University Computing Laboratory, 1996.
6. J. N. Darroch and D. Ratcliff. Generalized Iterative Scaling for Log-linear Models. *Annals of Mathematical Statistics*, 43:1470–1480, 1972.
7. L. De Raedt. Induction in logic. In R.S. Michalski and Wnek J., editors, *Proceedings of the 3rd International Workshop on Multistrategy Learning*, pages 29–38, 1996.
8. L. De Raedt and H. Blockeel. Using logical decision trees for clustering. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*. Springer-Verlag, 1997.
9. L. De Raedt and S. Džeroski. First order jk-clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375–392, 1994.
10. L. De Raedt and W. Van Laer. Inductive constraint logic. In *Proceedings of the 5th Workshop on Algorithmic Learning Theory*, volume 997 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1995.
11. L. Dehaspe and L. De Raedt. DLAB: A declarative language bias formalism. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems (ISMIS96)*, volume 1079 of *Lecture Notes in Artificial Intelligence*, pages 613–622. Springer-Verlag, 1996.
12. S.A. Della Pietra, V.D. Della Pietra, and J. Lafferty. Inducing features of random fields. Technical Report CMU-CS-95-144, Carnegie-Mellon University, Pittsburgh, PA, 1995.
13. B. Dolšak and S. Muggleton. The application of Inductive Logic Programming to finite element mesh design. In S. Muggleton, editor, *Inductive logic programming*, pages 453–472. Academic Press, 1992.
14. S.F. Gull and G.J. Daniell. Image Reconstruction from Incomplete and Noisy Data. *Nature*, 272:686, 1978.
15. E.T. Jaynes. Notes on present status and future prospects. In W.T Grandy and L.H. Schick, editors, *Maximum Entropy and Bayesian Methods*, pages 1–13. Kluwer Academic Publishers, 1990.
16. A. Karalič and I. Bratko. First order regression. *Machine Learning*, 1997. To appear.

17. S. Kramer. Structural regression trees. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, 1996.
18. M.P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.
19. S. Muggleton. Stochastic logic programs. *Journal of Logic Programming*, 1997. submitted.
20. G. Plotkin. A note on inductive generalization. In *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.
21. U. Pompe and I. Kononenko. Naive bayesian classifier within ILP-R. In L. De Raedt, editor, *Proceedings of the 5th International Workshop on Inductive Logic Programming*, pages 417–436, 1995.
22. U. Pompe and I. Kononenko. Probabilistic first-order classification. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*. Springer-Verlag, 1997.
23. A. Ratnaparkhi. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*. University of Pennsylvania, 1996.
24. E.S. Ristad. Maximum entropy toolkit, release 1.5 beta. Technical report, Princeton Univ., January 1997. `ftp://ftp.cs.princeton.edu/pub/packages/memt`.
25. A. Srinivasan, S.H. Muggleton, M.J.E. Sternberg, and R.D. King. Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, 85, 1996.

This article was processed using the LaTeX macro package with LLNCS style