Dynamic Predicate Construction for Learning Relational Concepts

Michael Chiang and David Poole

University of British Columbia
{mchc,poole}@cs.ubc.ca

Abstract. The aim of this work is to enrich the search space of relational rule learning by allowing dynamic construction of predicates. Specifically, the use (resp. non-use) of large predicates lead to hypotheses that are overly specific (resp. general). Without suitable predicates predefined, the space between these two hypotheses is inaccessible to the learner. We seek to address this problem by extensional predicate construction from domain clusters, thus allowing for the kind of intermediate hypotheses of interest here. We show that doing so lead not only to discovery of interesting domain subsets, but also better leveraging of predictive accuracy and overfitting. We develop a dynamic programming method for effectively achieving clusters of individuals, and demonstrate empirical results on on synthetic as well as real-world datasets.

Key words: Relational learning, inductive logic programming, predicate invention.

1 Introduction

Learning relational rules is typically done with a fixed, predefined *vocabulary* and hence a set of fixed features which are assumed sufficient to describe the target domain (e.g. [1–3]). The features in question are typically taken from the input dataset, and one has no *a priori* indication of how useful they may be in generating a descriptive rule.

The process of inducing a relational rule terminates when all predicates in the initial vocabulary have been used, or that a stopping criterion is satisfied, and exploration of the input space ceases[1]. If the input data harbours more information that could be fruitfully exploited, then the current induced hypothesis would be seen as being overly general. However, if we only have large predicates with which to perform clausal refinement, an overly complex hypothesis may result and introduces overfitting. To illustrate, we provide the following example from [6].

[1] This applies not only to logic-based rule learning (e.g. [4]), but also frame-based models such as probabilistic relational models [1], when one wishes to explicate its probabilistic dependency statements (conditional probabilities) by learning its structure (e.g. as a decision tree [5]).

Example 1. A survey of workers in U.S. cities, from which it can be seen that on average approximately 70% of the sampled population prefer to drive to work. However, in the city of New York (NYC) which accounts for one sixth of the survey data, only 25% drives to work. Thus, NYC is clearly a strong exception to the hypothesis that 70% of Americans drive to work. With the sole available feature being $lives(Person)$, whose possible values are the set of cities in the survey, the use and non-use of this predicate yields the following two theories respectively:

$$\forall P, drives(P) \overset{0.7}{\leftarrow} . \tag{1}$$

$$\forall P, drives(P) \overset{p_i}{\leftarrow} lives(P) = c_i. \tag{2}$$

where c_i is the i-th city and p_i the corresponding probability. By inspection, (1) is too general whilst (2) is overly complex even if it is a better fit to data. One could reasonably propose an intermediate theory such as

$$\begin{aligned} &\forall P, drives(P) \overset{0.25}{\leftarrow} lives(P) = nyc. \\ &\forall P, drives(P) \overset{0.73}{\leftarrow} lives(P) \neq nyc. \end{aligned} \tag{3}$$

which appears to achieve a better leverage overfitting.

This example reveals a (large) generality gap between (1) and (2), and hypotheses that lie in between (e.g. (3)) are not learnable when one depends on predefined (and non-existent) features to do so[2].

In this work, we propose to leverage the problem described above by optimally grouping values of large predicates, thereby introducing new extensional predicates in the learning process. The solution we propose utilises dynamic programming to optimise empirical loss on the training data (see Sect. 3). The proposed method can be employed during standard rule learning as an alternative to using large features directly, reducing model complexity and enhancing predictive accuracy.

The remainder of this paper is organised as follows. Sect. 2 covers some standard notations and background into loss functions for probability estimation. This is followed in Sect. 3 by an exposition on the notion of splitting on individuals and derivation of associated methodology. Experimental results on synthetic as well as real-world datasets are included in Sect. 4, where we show the value of our generated models. Discussion of related work and conclusion can be found in Sect. 5.

2 Preliminaries

We will use standard notation where uppercase letters (or words with leading uppercase) denotes random variables as well as logical variables, distinguishable

[2] It may be noted that the above example can be seen as a propositional problem. However, propositionalisation does not circumvent the problem as we are interested not in the choice of representation but achieving an intermediate hypothesis such as (3).

by context or explicitly when required. Lowercase letters denote instantiations, e.g. x is an instantiation of random variable X. If X is a logical variable then x is a constant. We use subscript index notation to represent discrete sets, e.g. $X_{1:n}$ is a set of random variables $X_1 \dots X_n$, and $x_{1:n}$ is a corresponding vector instantiation.

Now we define general loss functions for binary probability estimation, sourced from [7]. Given some predictor data $x_{1:n}$ and response data $y \in \{0,1\}$ (a Bernoulli observation), we are interested in estimating the quantity $\gamma = P(y = 1|x_{1:n})$. Let the discrepancy for estimator $\alpha(x_{1:n}) \in [0,1]$ for $y = 1$ and $y = 0$ be non-negative (e.g. the 1-norm), and based on which we respectively define *partial losses* to be $l_{1,\alpha}(x_{1:n})$ and $l_{0,\alpha}(x_{1:n})$, which are increasing functions. The point-wise empirical loss for estimating the value of y given $x_{1:n}$ (we drop $x_{1:n}$ in the sequel for brevity) is

$$\mathbb{L}(y|\alpha) = yl_{1,\alpha} + (1-y)l_{0,\alpha} \tag{4}$$

The point-wise *expected* loss for estimating γ then takes the form

$$\mathcal{L}(\alpha,\gamma) = \mathbb{E}_Y\mathbb{L}(Y|\alpha) = \gamma l_{1,\alpha} + (1-\gamma)l_{0,\alpha} \tag{5}$$

If $l_{1,\alpha}$ and $l_{0,\alpha}$ are chosen to be convex functions, (5) will also be convex with the minimum at γ, giving Fischer consistency. A common example of this is the *log-loss*, where $l_{1,\alpha} = -\log(\alpha)$ and $l_{0,\alpha} = -\log(1-\alpha)$. Our definitions here are general as our theoretical results in Sect. 3 are for general loss functions. However, for the purposes of evaluation and implementation, we choose *log-loss* as our primary measure for its direct correspondence to likelihood and entropy, which are standard measures in probabilistic machine learning. In this paper we will only discuss the case of binary probability estimation.

3 Domain splitting

In example 1, the domains of individuals are *people* and *cities*, where the latter can be seen as the range of the function *lives.*. We aim to construct new predicates such as $\mathit{lives}(\mathit{Person}, C_1), \dots \mathit{lives}(\mathit{Person}, C_n)$, where $C_{1:n}$ are disjoint sets of cities. The assignment of cities to these sets are based on corresponding statistics (e.g. counts of people who drive in a given city), and a probability estimator is calculated for each set that minimises loss on training data. Splitting on single domains as described here (e.g. cities only) is the focus of the paper, where partitioning in higher-dimensions (such as the space of all person-city pairs) warrants further investigation as discussed in Sect. 5.

Formally, we cast the domain splitting task as the assignment of K probability estimators to $M(> K)$ probabilities $\Omega = \gamma_{1:M}$. Let $\mathcal{A}_i$ be a subset (or partition) of Ω, and let $\bar{\Omega} = \mathcal{A}_{1:K}$. Given K, and a procedure to compute a probability estimate α_i for each $\mathcal{A}_i \in \bar{\Omega}$, the goal is to achieve $\bar{\Omega}$ such that the joint estimation loss incurred by $\alpha_{1:K}$ is minimised. That is, we want to solve

$$\mathcal{A}^*_{1:K} = \underset{\mathcal{A}_{1:K}}{\arg\min} \sum_{i=1}^{K} \mathbb{E}\left[\mathcal{L}(\alpha_i, \mathcal{A}_i)\right] \tag{6}$$

Let $J_x = \mathbb{E}\left[\mathcal{L}(\alpha_x, \mathcal{A}_x)\right]$ be the loss incurred by estimator α_x in partition $\mathcal{A}_x$, then the best estimator α_i is defined as follows:

$$\alpha_i = \arg\min_{\alpha} \mathbb{E}\left[\mathcal{L}(\alpha, \mathcal{A}_i)\right] = \arg\min_{\alpha} \mathbb{E}\left[\sum_{\gamma \in \mathcal{A}_i} \mathcal{L}(\alpha, \gamma)\right] \tag{7}$$

Solving (6) represents the outer-loop of our approach, whereas (7) represents the inner-loop. The general schema of this framework is illustrated in Fig. 1.

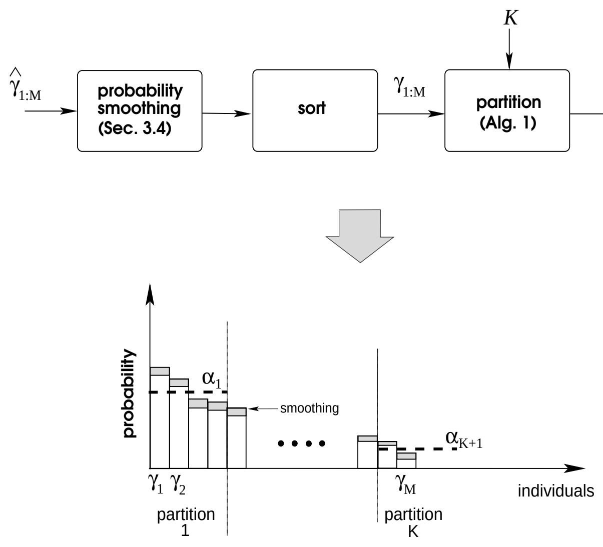


Fig. 1. Proposed schema for generating K partitions and estimators for $M > K$ probabilities.

Since $\gamma_{1:M}$ are empirical probabilities in practice, they are subject to small sample problems (e.g. small towns), which we address in our framework via probability smoothing as a preprocessor (see Sect. 3.4). Partitioning (Sect. 3.2) is done using Alg. 1 after sorting.

3.1 Computing estimators

We first discuss the inner-loop operation of computing α_i for a given partition $\mathcal{A}_i$. Recall that the objective function here is given by (7), and that we choose to use log-loss in this work (i.e. $l_{1,\alpha} = -\log \alpha$ and $l_{0,\alpha} = -\log(1-\alpha)$ in (5)), the loss of α_i for $\mathcal{A}_i$ is then expressed by

$$\begin{aligned} J_i &= \mathbb{E}\left[-\sum_{\gamma\in\mathcal{A}_i}\gamma\log\alpha_i + (1-\gamma)\log(1-\alpha_i)\right] \\ &= -\left(\sum_{\gamma\in\mathcal{A}_i}\mathbb{E}[\gamma]\right)\log\alpha_i - \left(\sum_{\gamma\in\mathcal{A}_i}\mathbb{E}[1-\gamma]\right)\log(1-\alpha_i) \end{aligned} \tag{8}$$

By strict concavity of log functions, J_i is strictly convex, and its stationary point represents our optimal joint estimator for $\mathcal{A}_i$, expressed as (9) below.

$$\alpha_i = \frac{\sum\limits_{\gamma_j\in\mathcal{A}_i}\gamma_j}{|\mathcal{A}_i|} \tag{9}$$

3.2 Monotonic estimator assignment

Before describing our algorithm for computing partitions, we develop results that make it possible. Let α_1, α_2, γ_1, and γ_2 be real numbers in $[0,1]$ such that $\alpha_1 > \alpha_2$ and $\gamma_1 > \gamma_2$. If we use α_i as an estimator for γ_i, an intuition which arises is that if α_1 incurs a smaller loss than α_2 in estimating γ_2, then α_1 is also superior in predicting γ_1. Conversely, if α_2 is the superior estimator for γ_1, then it is also superior for for γ_2. We show this formally in Lemma (1) as follows.

Lemma 1. *Let γ_1 and γ_2 be probabilities, such that $\gamma_1 > \gamma_2$. Also, let α_1 and α_2 be probabilistic estimates of γ_1 and γ_2 respectively, such that $\alpha_1 > \alpha_2$. Applying each estimator to γ_1 and γ_2,*

$$\mathcal{L}(\alpha_2,\gamma_1) \le \mathcal{L}(\alpha_1,\gamma_1) \implies \mathcal{L}(\alpha_2,\gamma_2) < \mathcal{L}(\alpha_1,\gamma_2) \tag{10}$$

Conversely,

$$\mathcal{L}(\alpha_1,\gamma_2) \le \mathcal{L}(\alpha_2,\gamma_2) \implies \mathcal{L}(\alpha_1,\gamma_1) < \mathcal{L}(\alpha_2,\gamma_1) \tag{11}$$

Proof. *Suppose that $\mathcal{L}(\alpha_2,\gamma_1) \le \mathcal{L}(\alpha_1,\gamma_1)$, which expands (using (5)) to the form*

$$\gamma_1 l_{1,\alpha_2} + (1-\gamma_1)l_{0,\alpha_2} \le \gamma_1 l_{1,\alpha_1} + (1-\gamma_1)l_{0,\alpha_1} \tag{12}$$

Let $\gamma_1 = \gamma_2 + \delta$ where $\delta > 0$, then eq. (12) can be expressed as follows

$$\begin{aligned} &(\gamma_2+\delta)l_{1,\alpha_2} + (1-\gamma_2-\delta)l_{0,\alpha_2} \le (\gamma_2+\delta)l_{1,\alpha_1} + (1-\gamma_2-\delta)l_{0,\alpha_1} \\ &\implies \quad \gamma_2 l_{1,\alpha_2} + (1-\gamma_2)l_{0,\alpha_2} + \delta\left(l_{1,\alpha_2} - l_{0,\alpha_2}\right) \\ &\qquad\quad \le \gamma_2 l_{1,\alpha_1} + (1-\gamma_2)l_{0,\alpha_1} + \delta\left(l_{1,\alpha_1} - l_{0,\alpha_1}\right) \end{aligned}$$

$$\therefore \quad \mathcal{L}(\alpha_2,\gamma_2) + \delta\left(l_{1,\alpha_2} - l_{0,\alpha_2}\right) \le \mathcal{L}(\alpha_1,\gamma_2) + \delta\left(l_{1,\alpha_1} - l_{0,\alpha_1}\right)$$

Since $l_{1,\cdot}$ and $l_{0,\cdot}$ are increasing functions, and that $\alpha_1 > \alpha_2$,

$$\begin{gathered} l_{1,\alpha_2} > l_{1,\alpha_1} \\ l_{0,\alpha_2} < l_{0,\alpha_1} \\ \implies l_{1,\alpha_2} - l_{0,\alpha_2} > l_{1,\alpha_1} - l_{0,\alpha_1} \end{gathered}$$

In combination with (12), it must be the case that

$$\mathcal{L}(\alpha_2, \gamma_2) < \mathcal{L}(\alpha_1, \gamma_2) \tag{13}$$

which proves (10). The converse result (11) is proved by noting that it is the contrapositive statement of (10), and is thus logically equivalent.

It follows from Lemma 1 that assigning α_2 to γ_1 and α_1 to γ_2 does not yield optimal loss, and can always be improved upon by reversing the assignment. We state this in the Theorem 1 as follows.

Theorem 1. *Given two probabilities γ_1 and γ_2 such that $\gamma_1 > \gamma_2$, and probabilistic estimates α_1 and α_2 such that $\alpha_1 > \alpha_2$,*

$$\neg \left((\mathcal{L}(\alpha_2, \gamma_1) \leq \mathcal{L}(\alpha_1, \gamma_1)) \wedge (\mathcal{L}(\alpha_1, \gamma_2) \leq \mathcal{L}(\alpha_2, \gamma_2)) \right) \tag{14}$$

Proof. Equation (14) is logically equivalent to

$$\mathcal{L}(\alpha_2, \gamma_1) \leq \mathcal{L}(\alpha_1, \gamma_1) \Rightarrow \mathcal{L}(\alpha_2, \gamma_2) < \mathcal{L}(\alpha_1, \gamma_2)$$

which in turn is true in (10) in Lemma 1.

Theorem 1 essentially states that monotonic value ordering of α_i must be aligned to that of γ_i, and that breaking the ordering (e.g. unordered assignments) always increases loss. The argument extends to the case where we have $\gamma_{1:M}$ and two estimators α_1, α_2. There, every γ_i assigned α_1 must be greater than each of those assigned α_2. In effect, we split the set of γ's into two sets $\mathcal{A}_1$ and $\mathcal{A}_2$ where

$$\forall \gamma_r \in \mathcal{A}_1, \forall \gamma_s \in \mathcal{A}_2 : \gamma_r > \gamma_s$$

and have shown that overall expected loss of the two estimators cannot be improved upon by moving any element of $\mathcal{A}_1$ to $\mathcal{A}_2$ and vice versa. If we now have K estimators $\alpha_{1:K}$ such that $\alpha_1 > \alpha_2 > \ldots > \alpha_K$, we directly apply the above pairwise result to all pairs of estimators. Then, the set of γ's are split into K disjoint subsets $\mathcal{A}_1 \ldots \mathcal{A}_K$, where for all $i < j$

$$\forall \gamma_r \in \mathcal{A}_i, \forall \gamma_s \in \mathcal{A}_j : \gamma_r > \gamma_s \tag{15}$$

This means that to find an optimal bisection of $\gamma_{1:M}$, we need only search linearly in the sorted set described by (15). To make multiple partitions, we apply the same principle in a dynamic programming framework described in Sect. 3.3 that follows.

3.3 Optimal splitting via dynamic programming

Theorem 1 shows that in order to find the best K disjoint subsets of M probabilities $\gamma_{1:M}$ to assign our K estimators, we only need to search in the space where $\gamma_{1:M}$ are sorted. In the simple case of finding two partitions, one simply searches linearly in $\{\gamma_{1:M}\}_{sorted}$ for the best cut-point. Optimal loss can achieved by computing $\alpha_{1,2}$ using (9). To see how $K > 2$ partitions, suppose we are searching for the best j-th cutpoint c_j, which require that all cuts $c_{1:j-1}$ have been made. Note also that the best $c_{1:j-1}$ cuts apply to the set $\gamma_{1:c_j}$. This recursive argument suggests a dynamic programming approach, and indeed is true given that our objective function (implicit in (6)) can be written in additive form $\mathcal{J}_K = \min \sum_{i=1}^{K} J_i$, which yields the recursive form

$$\mathcal{J}_K = \mathcal{J}_{K-1} + \min J_K \tag{16}$$

The recursive form of our minimisation problem directly validates the use of dynamic programming, where we cache scores of all previous partitionings. To perform this, we set up a $M \times (K+1)$ table $\mathcal{D}$, whose elements $d_{m,k}$ are

$$d_{m,k} = \min_{r<m} \left(\mathcal{J}_{k-1}^{r} + J_k^r\right); \quad k = 0 \dots K, m = 1 \dots M \tag{17}$$

which represents the score of the best k-partition model on $\gamma_{1:m}$. $\mathcal{J}_{k-1}^{r}$ denotes the loss for the best $k-1$ partitions on the partial set $\gamma_{1:r-1}$ which is equivalent to $d_{r-1,k-1}$, and J_k^r is the loss of the new partition composed of $\gamma_{r:m}$. Algorithm 1 sweeps through and computes all elements of this matrix, returning $d_{M,K}$ as the final value. The final cutpoints are given by $c_k^* = \arg\min_c [d_{c,k}]$, $k = 1 \dots K$.

Algorithm 1 Finding the best K partitions of the sorted set $\gamma_{1:M}$

```
for $k \in 0 : K$ do
  for $m \in 1 : M$ do
    $\mathcal{H} = \emptyset$
    for $r \in 1 : m$ do
      $\alpha \leftarrow \frac{\sum_{j=r}^{m} \gamma_j}{m-r}$
      $\mathcal{H} \leftarrow \mathcal{H} \cup (d_{r-1,k-1} + J_k^r(\alpha))$
    end for
    $d_{m,k} \leftarrow \min \mathcal{H}$
  end for
end for
```

3.4 Small sample smoothing

As mentioned, $\gamma_{1:M}$ are often empirical probabilities, and therefore subject to small sample problems. Computing an estimator directly from (9) can result in unbounded test loss (e.g. when a test point $y = 1$, and $\alpha = 0$ was computed from the training set, then we have $L(1|0) = -(1)\log(0) - (0)\log(0) = -\infty$). Application of the partitioning algorithm only avoids such problems if one can avoid grouping sets of exclusively extreme probabilities. This necessitates smoothing of the inputs probabilities.

Let sample S be a set of binary data $y_{1:z}$, in which z^+ elements have the value 1. Then, let $\hat{\gamma} = \frac{z^+}{z}$ be an empirical estimate of the probability of finding 1's in S, the smoothed empirical probability γ can be expressed as

$$\bar{\gamma} = \frac{z^+ + a}{z + b} \tag{18}$$

Different forms of smoothing arise from different choices of a and b. In this work we also use the optimisation over choices of a and b via the empirical Bayes method for the Dirichlet-multinomial model (see [8]). The smoothed set of probabilities are then used directly as inputs to Alg. 1. As part of our evaluations, we will also use the well-known Laplace estimate ($a = 1$ and $b = 2$).

4 Experiments

In our experiments, we compare three types of models: (i) models corresponding to no partitioning (e.g. (1)) for which we use a simple maximum likelihood estimate, denoted as *Single*, (ii) those resulting from splitting on a given predicate exhaustively (e.g. (2)), and (iii) those resulting from the application of Algorithm 1. Types (ii) and (iii) have variants depending on the smoothing method used. Namely, type (ii) with no smoothing is denoted as *ML*, and *EmpBayes* with smoothing by empirical Bayes, and *Laplace* with Laplace smoothing. Type (iii) follow a similar system, giving *K-ML*, *K-EmpBayes* and *K-Laplace* methods, where smoothing is done on the initial set of probabilities before being partitioned into K groups. The aim of these experiments are to show that (iii) (splitting optimally) is better than (i) (no splitting) and (ii) full splitting, from the standpoint of classification as well model complexity. The use of specific (relational) rule learners to generate (i) and (ii) is irrelevant to our investigations, as standard variants do not generate type (ii) models.

4.1 Synthetic data

In this experiment we wish to evaluate the average performance over a vast number of wide-ranging datasets. For this purpose we create a generative model to synthesise $M = 30$ bins of binary data, where the size and binomial parameter of each bin are subject to randomisation. Further, we correlate the binomial parameter with bin size in an artificial way, according to a randomly selected sigmoid function. Hidden relations are incorporated (e.g. that which can be described by an unavailable relational feature) by adding strong correlation between select subsets of individuals. We evaluate log-loss for all of the listed methods except *ML* and *K-ML* due to infinite losses on test data from extremal likelihoods. K

is nominally set to 8. Figure 2(a) represents the average 10-fold cross-validated test loss over 100 randomly generated dataset of size N, where $N = 500 \times 2^r$, $r = 1 \ldots 13$. It shows that all methods that partitioning exploits information better than *Single*, resulting in lower loss. Also, whilst probability smoothing helps in avoiding infinite losses, its effect appears to be negligible. However, given that $K = 8$ (compared to $M = 30$), the similarity in performance between full-partition and K-partition methods highlights the advantage of performing K-partitioning. In essence, we show in this trial that exploring the appropriate level of generalisation (i.e. those in the space between those found by type (i) (*Single*) and type (ii) methods (full-partitioning)) leads to much simpler hypotheses without sacrificing accuracy.

4.2 Gene classification data

The goal with the gene classification dataset[3] is to classify the function of a gene given its properties, and possibly known relations to other genes. Of the nine possible gene features, we choose $geneclass(G)$ (which has 22 values, e.g. $M = 22$) for our purposes[4]. There exists 13 functional classes to which a gene can belong, where we treat each as a separate binary classification problem, e.g. gene G is either in class c or not. Here we are interested in seeing how each model exploits the available information, which we illustrate by plotting the relative loss[5] on predicting for each functional class against the entropy inherent in the data for that class. In this experiment, we search for the best K using 10-fold cross-validation. The results are presented in Fig. 2(b), and shows that the advantage of partitioning is less pronounced when the data harbours little information (though it still appears to perform competitively). However, in accordance to our intuition, as the available information increases K-partition models not only outperforms *Single* but also their respective full-partition models. This suggests that the K-partition models effectively achieve good accuracy without overfitting, which is supported by the fact that K range from 6 to 8 (depending on class) compared to $M = 22$.

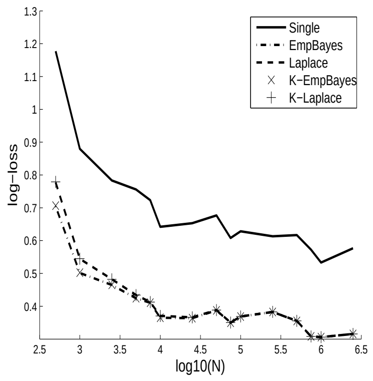
4.3 Journey-to-work data

The journey-to-work surveys conducted by the U.S. census [6] was briefly described in Sect. 1. Counts of people for the four categories of transportation to work (drive, carpool, transit, other) are given, grouped by city of work (there are $M = 50$ cities). Our feature is $lives(X, City)$, using which we carry out binary prediction for each of the four classes. In all there are approximately 18 million data points, and no city carries extreme likelihoods, which allows *ML* to be used here. 10-fold cross-validation tests are carried out, with losses recorded in Table 1. K is computed in the same way as in Sect. 4.2, ranging from 7 to 12.

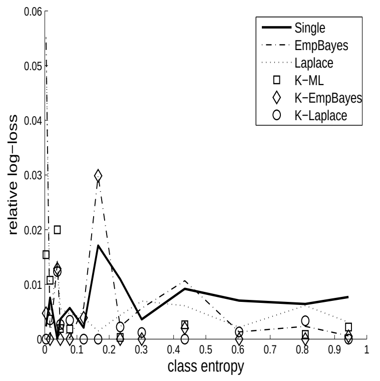
[3] KDD Cup 2001 (http://www.cs.wisc.edu/ dpage/kddcup2001).

[4] Note ”gene class” is not equivalent to the gene’s function.

[5] The difference between log-loss and the minimum log-loss achieved by any method for that function class.



(a) (Synthetic) Log-loss as a function of size of the dataset. K-partitioning achieves similar performance to full-partitioning, with $K = 8$.



(b) (Gene classification) Log-loss as a function of information available in the training data. Points represented as symbols correspond to K-partitioning methods, outperforming both full and non-partitioning methods with increasing information.

Fig. 2. Results for experiments on (a) synthetic data, and (b) gene classification data.

Table 1. Log-loss using the 'travel-to-work' dataset.

	DRIVE	CARPOOL	TRANSIT	OTHER
SINGLE	0.9806	0.5553	0.7007	0.4556
LAPLACE	0.8679	0.5512	0.5522	0.4468
EMPBAYES	0.8679	0.5512	0.5522	0.4468
ML	0.8679	0.5512	0.5522	0.4468
K-ML	0.8679	0.5512	0.5522	0.4469
K-EMPBAYES	0.8679	0.5512	0.5522	0.4468
K-LAPLACE	0.8679	0.5512	0.5522	0.4468

Table 1 shows that whilst there appears to be little overfitting even for the full-partitioning models, the K-partition models achieve similar scores, thereby demonstrating the same benefits as shown in previous experiments. *Single* again over-generalises and scores poorly.

It is also of interest to examine the qualitative models produced by K-partitioning. For the class *drive* and setting $K = 2$, *K-EmpBayes*, *K-Laplace* and *K-ML* all lead to the model shown in Fig. 3 (due to negligible effects of priors). From this model, a rule set that follows (for the $K = 3$ case) is shown

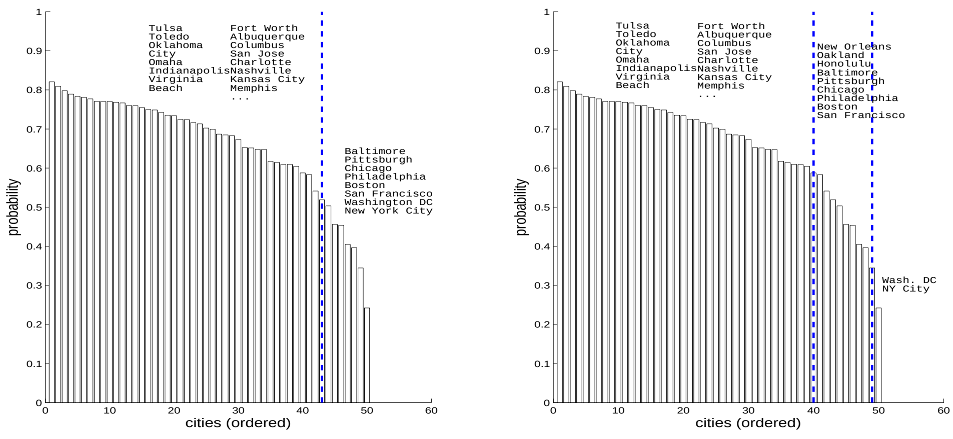


Fig. 3. Models produced by Alg. 1 with $K = 2$ (left) and $K = 3$ (right). Individuals (cities) of each partition are also labelled.

in (19)

$$\begin{aligned}
&\forall P, drives(P) \overset{\theta_i}{\leftarrow} lives(P, C) \wedge C \in L_i.\\
&where \quad i = 1 \ldots 3,\\
&\quad L_1 = \{\text{'Tulsa'}, \text{'Toledo'}, \ldots\}, \theta_1 = 0.714\\
&\quad L_2 = \{\text{'New Orleans'}, \text{'Oakland'}, \ldots\}, \theta_2 = 0.470\\
&\quad L_3 = \{\text{'Washington DC'}, \text{'New York City'}\}, \theta_3 = 0.251.
\end{aligned} \tag{19}$$

It can be seen our method generated models that summarises the journey-to-work habits of people in different cities. It resembles the kind of model shown in (3) in Sect. 1.

5 Conclusions and related work

We have presented an effective method for grouping values of large relational predicates to construct new (extensional) predicates. This results in intermediate hypotheses that reside in the generality gap outlined in Sect. 1, leading to better leveraging of overfitting as demonstrated in Sect. 4, as well as discovery of interesting domain subsets. We argued that standard rule learning mechanisms require predefined predicates to find hypotheses in this gap, whereas we do not.

Existing work in (statistical) *predicate invention* (e.g. [9, 10]) can also, in principle, make intermediate hypotheses from ground examples. However, it is unclear whether it will achieve the same theories and classification performance. Another related work is *(relational) subgroup discovery* [11, 12], which distinguishes individuals by feature descriptions, thus assuming a predefined feature set that we do not require. Along similar lines, recent work in *view learning* attempts to generate new relations (using an existing ILP algorithm, Aleph) from existing ones to optimise predictive performance.

It is noteworthy that some results in [13] for binning bitmap indices capture the same idea as that expressed by Theorem 1. However, these are independent works, and the results in this paper is not attained directly from [13].

The main shortcoming of the method presented here is that one cannot easily partition multi-dimensional domains which exists when higher-arity predicates such as $lives(P, City, Suburb)$ are used. The ability to do so avails an even richer class of relational rules, and is an important extension. The work in this paper presents an initial step in this direction.

References

1. Getoor, L., Friedman, N., Koller, D., Taskar, B.: Learning probabilistic models of link structure. Journal of Machine Learning Research (2002)
2. Muggleton, S.: Learning structure and parameters of stochastic logic programs. In Matwin, S., Sammut, C., eds.: ILP02. Volume 2583 of LNAI., SV (2003) 198–206
3. Blockeel, H., Raedt, L.D.: Top-down induction of first-order logical decision trees. Artificial Intelligence **101**(1-2) (1998) 285–297
4. Quinlan, J.R.: Learning logical definitions from relations. Machine Learning **5** (1990) 239–266
5. Getoor, L.: Learning Statistical Models from Relational Data. PhD thesis, Stanford University (2001)
6. U.S. Census Bureau: Travel to work characteristics for the 50 largest cities by population in the united states: 1990 census (1990)
7. Shen, Y.: Loss functions for binary classification and class probability estimation. PhD thesis, University of Pennsylvania (2005)
8. Minka, T.: Estimating a Dirichlet distribution. Technical report, M.I.T. (2000)
9. Craven, M., Slattery, S.: Relational learning with statistical predicate invention: Better models for hypertext. Machine Learning **43**(1/2) (2001) 97–119
10. Muggleton, S.: Predicate invention and utilisation. Journal of Experimental and Theoretical Artificial Intelligence **6**(1) (1994) 127–130
11. Kloesgen, W. In: Explora: a multipattern and multistrategy discovery assistant. American Association for Artificial Intelligence, Menlo Park, CA, USA (1996) 249–271
12. Zelezný, F., Lavrac, N.: Propositionalization-based relational subgroup discovery with rsd. Machine Learning **62**(1-2) (2006) 33–63
13. Rotem, D., Stockinger, K., Wu, K.: Efficient binning for bitmap indices on high-cardinality attributes. Technical report, Lawrence Berkeley National Laboratory (2004)