Assessing Inductive Logic Programming Classification Quality by Image Synthesis

Bart Lamiroy, Jean-Philippe Ropers

Assessing Inductive Logic Programming Classification Quality by Image Synthesis

Bart Lamiroy Jean-Philippe Ropers
Nancy-Université – INPL – LORIA
Campus Scientifique – BP 239
54506 Vandoeuvre-lès-Nancy Cedex – France

Abstract

One of the major difficulties in classification is the assessment of classification quality. This paper builds on a recently developed classification technique, based on *Inductive Logic Programming* to propose a way of visually reconstructing the learnt classes in order to give a feedback to the user. The system translates positioning constraints into a linear programming problem, which can be solved with standard state-of-the-art approaches. The result can then be used for generating a synthetic image, representative of the class.

1 Introduction

We have recently [1, 2] introduced a new approach to graphical symbol representation and classification by means of a restricted visual vocabulary and the use of *Inductive Logic Programming* (ILP) [3]. Our current investigations have shown that ILP is a very powerful framework and that it allows for types of symbol characterization of symbols that are not possible with numerical classification techniques. Since one of the advantages of ILP is to be able to express image properties in a close-to natural language, it is necessary to have an easy, but objective way to assess the image properties that are used for classification.

In this paper we develop a method that allows us to generate a synthetic image from a set of first order logic expressions that are characterizing an image class. This will allow the user to validate if the learning process has been coherent with the context semantics of the images under consideration.

2 Visual Vocabulary and ILP

The approach developed in [1, 2] and recently extended, produces a description of a symbol using first order logic predicates. These predicates express relations between previously extracted visual cues from the image (composing the *visual vocabulary*). These cues are extracted using classical image treatment algorithms and consist of: circles [4], loose endpoints, thick components and corners.

These cues are then linked together with relative positioning relations (`north`, `north-west`, `west` ...), `bigger`, `smaller`, `near`, `far`, `inside`, `superimposed`. As a result, the image description can be expressed with first order logic predicates that are quite close to a natural language description like: “Image A contains two circles, one upper-right corner and two lower-left corners; the first circle lies left (*east*) to the second, and above (*north*) the upper-right corner ...”. More prosaically speaking, this gives something like:

```
type(prim_A,circle). type(prim_B,cornerne). type(prim_C,cornersw).
type(prim_D,cornersw). type(prim_E,circle). has_element(img,prim_A).
has_element(img,prim_B). has_element(img,prim_C).
has_element(img,prim_D). w(prim_A,prim_E). n(prim_B,prim_A). ...
```

Inductive Logic Programming allows to apply supervised learning on these data, provided that it’s given

1. a set of known vocabulary, rules, axioms or predicates, describing the domain knowledge base $\mathcal{K}$;

2. a set of positive examples $\mathcal{E}^+$ the system is supposed to describe or characterize with the set of predicates of $\mathcal{K}$;

3. a set of negative examples $\mathcal{E}^-$ that should be excluded from the deducted description or characterization.

The ILP solver is then able to find the set of properties $\mathcal{P}$, expressed with the predicates and terminal vocabulary of $\mathcal{K}$ such that the largest possible subset of $\mathcal{E}^+$ verifies $\mathcal{P}$, and such that the largest possible subset of $\mathcal{E}^-$ does not verify $\mathcal{P}$. This approach has already been successfully used for extraction of semantics from written text [5] or in document and image analysis structures [6, 7].

This approach allows for learning common properties within classes of symbols such as to express non-trivial knowledge of visual representation of more semantic concepts.

The conclusions of the previous studies have shown that ILP is very useful for describing complex and non trivial symbols and taking into account non-geometric transformations. However, the approach remains tributary of the initially defined vocabulary $\mathcal{K}$. More specifically the adjunction of new relationships influences on the classification "quality" (provided there is such a notion of quality). In this paper we propose a subjective classification assessment tool, based on image synthesis.

The paper is organized as follows. First we give a brief overview of the classification differences induces by variations in the description vocabulary (section 2.2). We then provide an approach to retrieve image geometry from an first order logic image description in section 3, before concluding and opening new perspectives in section 2.4.

2.1 Vocabulary Description

In order to give a clear example of what can vary in the classification results, depending on the used vocabulary, we introduce different types of predicates. Relations like proximity, size *etc.*. significantly modify classification results. In this section we describe the semantics of the new relations we are adding to $\mathcal{K}$. These new relations concern: connectedness, qualitative distance and relative size [8]. We shall then combine these in order to see how they influence on the final classification.

2.1.1 Inclusion and Connectedness

In order to distinguish whether two elements in the image are connected we construct the relation `superimposed` and set up a new algorithm to evaluate if two elements are bound. To do that, we compute the connected component tree of each individual element and the connected component tree of synthetic image summing both. Since, by construction of the different vocabulary-related extraction algorithms, each element has a unique black component at the root of this tree, it is quite straightforward to determine if two components are connected, disjoint or included one into another by simply counting the number of components of the fusion of both elements. Indeed, if the number of connected components of the fusion is equal to the sum of connected components of each individual element, then both are disconnected. On the contrary, if the depth of the resulting component tree is equal to the sum of individual depths, then both are `included`. Any other situation means both elements are connected (and therefore `superimposed`).

2.1.2 Dimensional Relations

The implementation of the dimensional relations are also based on the connected component trees. We proceed by comparing the area of the first child of the root of both elements. All elements are linked pairwise by a size relation : `bigger`, `smaller` or `samesize`. Here again, we used the hypothesis that the connected component tree of a descriptor is a root with only one child.

This algorithm is based on a pixel-wise estimation of the area covered by the element, and is therefore a very reliable measure.

2.1.3 Proximity

While the two previous sets of relations were quite obvious, proximity is dealt with in a more subtle manner. `near` and `far` are subjective (or at least non-absolute) concepts. On our case, these relations are considered to be "*object-centric*", meaning that `B` is close *with respect to* `A` if the distance between both is comparable to the size of `A`. This induces, however, that the proximity relation is asymmetric.

As a practical point of view, we compute first a dilation of the element `A`, then if `A` and `B` are superimposed (according to relation above) `B` is close to `A`. The size of the structural element is $\sqrt{Area(A)}$.

Actual positioning
a

`near(A,B)`
b

`far(B,A)`
c

Figure 1: relations far and near

2.2 Influences on Classification

In what follows we are going to compare different configurations and see how changes in the vocabulary affect learned classification rules. We distinguish five configurations:

- the *basic* configuration, including directional relations with `inside` as in [2];
- the *proximity* configuration, directional relations, `near` and `far` as described in section 2.1.3;
- the *superimpose* configuration: directional relations, `superimposed` and `inside` described in section 2.1.1;
- the *size* configuration: directional relations, `bigger`, `smaller`, `samesize` as described in section 2.1.2;
- *all* the relations: directional relations, `near`, `far`, `superimposed`, `inside`, `bigger`, `smaller` and `samesize`.

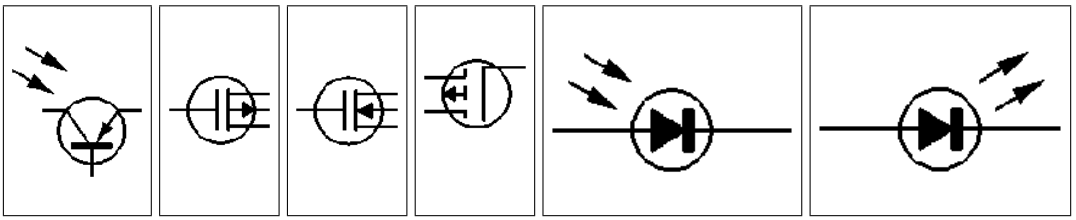

Figure 2: Image Set of Positive Examples.

2.2.1 Introducing Ambiguity

We use the set of symbols from Figure 2 and use our method to learn a description of these symbols with respect to a set of counter-examples. The different configurations (*i.e.* allowing more or less predicates to be used, as mentioned above) we have tested give the following, results:

- *basic*: `has_element(A,B), type(B,circle), has_element(A,C), inside(C,B).`
- *proximity*: `has_element(A,B), type(B,circle), has_element(A,C), near(B,C).`
- *superimpose*: `has_element(A,B), type(B,circle), has_element(A,C), superimposed(C,B).`
- *size*: `has_element(A,B), type(B,circle), has_element(A,C), bigger(B,C).`
- *all*: `has_element(A,B), type(B,circle), has_element(A,C), bigger(B,C).`

Although this might seem normal at a first glance (actually it is quite comforting as a result) it raises a fundamental problem: the one of the uniqueness of the symbol representation. All configurations give perfect classifications, and use the same number of predicates and of primitives. Although one can argue that the first four situations cannot be compared *per se*, the last one creates the issue. What causes our system to select `bigger` as distinguishable predicate, compared to `superimposed`, `near` or `inside`? This needs to be further investigated, and necessarily depends on the implementation, order of declaration or any other arbitrary rule. It is clear that there will be a strong need to define an order of priority on our predicate ruleset, at least.

Although this is essentially not a problem with, for instance, SVM classifiers [9], it raises an interesting question on classification. Indeed, all classification approaches are evaluated on their capacity of correctly separating instances from one another and affect a class label to them and are tested with respect to a *closed world* of instances. The fact that they are usually trained on a distinct training set, and tested on another set doesn't quite matter that much: there is no real evaluation on the sensitivity of the classification boundaries with respect to the training set, as long as the final classification on the test set is acceptable. The fact that our approach, compared to others (*i.e.* statistical) gives descriptions that are readily interpretable allows for a human-based validation of the classification rules, and can thus be embedded in a relevance feedback process.

2.2.2 Shorter Rules

In most experiments, the fact of adding new predicates results in shorter classification clauses, as shown on the example below. The basic configuration, expresses that the diode symbols represented in Figure 3 contain a `ccblackthick` and a `cornerse` component, as well as an unqualified component `D`. The corner is *east* of `D` and the thick component is `inside D`.

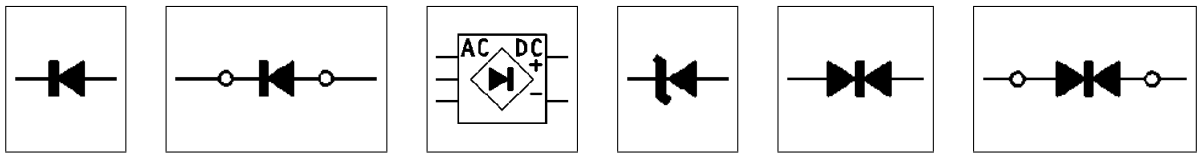

Figure 3: Diode Examples.

```
symbol(A) :-
   has_element(A,B), type(B,ccblackthick), has_element(A,C),
   type(C,cornerse), has_element(A,D), e(D,C), inside(D,B).
```

With the addition of the newly defined relations, the rule produced is one element shorter.

```
symbol(A) :-
   has_element(A,B), type(B,ccblackthick), has_element(A,C),
   has_element(A,D), e(D,C), superimposed(D,B).
```

We can see that `superimposed` replaces `inside` and a `cornerse`.

2.3 Adding Complexity

Our final series of experiments concern situations where the positive example configurations cannot be reduced to a single predicate. The fact of adding predicates does not fundamentally increase the number classification rules. The rules produced this time are a bit different but the structure is almost the same. One of the rule sets produced in the basic case is:

```
[Rule 1] [Pos cover = 1 Neg cover = 0]
symbol(img_180_1).

[Rule 2] [Pos cover = 2 Neg cover = 0]
symbol(A) :-
   has_element(A,B), type(B,circle), has_element(A,C), e(B,C),
   type(C,cornerne).

[Rule 3] [Pos cover = 1 Neg cover = 0]
symbol(img_184_1).

[Rule 4] [Pos cover = 2 Neg cover = 0]
symbol(A) :-
   has_element(A,B), type(B,ccblackthick), has_element(A,C),
   has_element(A,D), e(B,C), type(C,cornerse), inside(D,B).
```

Introducing the new predicates changes the ruleset to three individual rules and one collective covering three examples, this rule is depicted below:

```
symbol(A) :-
   has_element(A,B), type(B,ccblackthick), has_element(A,C),
   has_element(A,D), e(C,D), far(D,C), type(D,cornersw).
```

Here again, as in the previous section, it would be an error to assess this result from a human-centric point of view. The fact that two rules describing the symbol class with respect to the presence of `cornerne` and `cornerse` elements suddenly is described by the presence of a `cornersw` element is only due to the fact that our approach is just another minimization technique, ILP looks for the most generic ruleset. This means containing the smallest number of predicates, covering the largest set of positive examples and avoiding the largest set of negative examples.

2.4 Conclusion on Classification

The conclusion of this work is in line with previous work [2] and with what we expected: the enhanced expressiveness of the description language improves the produced classification rules, by reducing their size. It would be an error to consider that the increase of the vocabulary would induce a richer description of the classified symbols. As mentioned in section 2.3, this would be contrary to the optimization criteria of our approach, which tries to minimize the number of predicates used to describe a symbol.

One important issue has not been covered by this work, is the study of the the effect of the vocabulary increase with respect to the method's inherent sensitivity to the detection of the visual elements and their associated uncertainty. One of the main bottlenecks is the fact that ILP doesn't allow uncertainty to be integrated in its ruleset. One of the extensions of the method presented in this paper will be the use of an uncertainty-aware solver [10, 11].

On the other hand, it is clear that the final results may seem awkward with respect to a human-centric point of view. The rest of this paper will therefore develop a method allowing to represent the obtained first-order logic description into a visual form, in order to visually asses the sense of the obtained set of predicates.

3 Generating Images

Once we have a description of an image in our previously described language, it is quite straightforward to recover an image that is coherent with this description. In order to achieve this, we simply express the relations as a Linear Programming [12] problem.

3.1 Modeling the Problem

As a matter of fact, the set of predicates describing an image can be seen as a set of constraints over the properties of all components composing the image, and described in the previous sections. As we shall show in section 3.2, these constraints can all be approximated by linear inequations. Combining this set of linear inequations with an objective function f expressing that the resulting arrangement of components should be as compact as possible, we obtain a linear constraint set A and a linear objective function f such that, with $\mathbf{x} = (x_1, x_2, \dots x_n)$, $\mathbf{c} = (c_1, c_2, \dots c_n)$ and $f(x_1, x_2, \dots x_n) = c_1x_1 + c_2x_2 + \dots c_nx_n$, we obtain $f(\mathbf{x}) = \mathbf{x}\mathbf{c}^t$ and $A\mathbf{x} \leq b$ for minimization.

3.2 Inequation Set

In order to represent our image description with linear constraints, we are going to model all components by their bounding box. Let $\mathcal{C}$ be a detected component. The centre of its bounding box is given by $(x_\mathcal{C}, y_\mathcal{C})$, its width and height by $w_\mathcal{C}$ and $h_\mathcal{C}$.

Represented in the previously described linear programming framework, this gives

$$\mathbf{x} = (x_{\mathcal{C}_1}, y_{\mathcal{C}_1}, h_{\mathcal{C}_1}, w_{\mathcal{C}_1}, \dots x_{\mathcal{C}_n}, y_{\mathcal{C}_n}, h_{\mathcal{C}_n}, w_{\mathcal{C}_n}, x_I, y_I, h_I, w_I)$$

for n visual components included in an image I.

For each identified type of component, extra constraints apply: for circles, $w_\mathcal{C} = h_\mathcal{C}$; extremities have $w_\mathcal{C} = h_\mathcal{C} = 1$ and corners have $w_\mathcal{C} = h_\mathcal{C} = \mathbf{t}$ where $\mathbf{t}$ is the template size used for detection.

The relationships between any two components $\mathcal{C}_1$ and $\mathcal{C}_2$ give the following constraints. It is to be noted that, due to our deliberate choice of using linear programming, the constraints are reduced to their linear expression:

- `bigger`($\mathcal{C}_1$,$\mathcal{C}_2$) gives $w_{\mathcal{C}_1} - w_{\mathcal{C}_2} \geq 0$ and $h_{\mathcal{C}_1} - h_{\mathcal{C}_2} \geq 0$. `smaller` gives rise to the dual equations.
- `inside`($\mathcal{C}_1$,$\mathcal{C}_2$), needs a little quirk to get linearised. Indeed, the condition for bounding box $\mathcal{C}_1$ being entirely included into $\mathcal{C}_2$ is the following:

$$w_{\mathcal{C}_2} > w_{\mathcal{C}_1} \tag{1}$$

$$|x_{\mathcal{C}_1} - x_{\mathcal{C}_2}| + \frac{w_{\mathcal{C}_1}}{2} < \frac{w_{\mathcal{C}_2}}{2} \tag{2}$$

(and similarly for y and h parameters). Unfortunately, the absolute value function is non linear. However, we can constrain our images to have positive coordinates. Under this assumption, the above constraints

can be reformulated as

$$w_{\mathcal{C}_2} > w_{\mathcal{C}_1} \tag{3}$$

$$x_{\mathcal{C}_1} - x_{\mathcal{C}_2} + \frac{w_{\mathcal{C}_1}}{2} < \frac{w_{\mathcal{C}_2}}{2} \tag{4}$$

$$x_{\mathcal{C}_2} - x_{\mathcal{C}_1} + \frac{w_{\mathcal{C}_1}}{2} < \frac{w_{\mathcal{C}_2}}{2} \tag{5}$$

which are perfectly linear.

- `superimposed`$(\mathcal{C}_1,\mathcal{C}_2)$ is approximated by the overlapping of bounding boxes: $(x_{\mathcal{C}_1}, w_{\mathcal{C}_1})$ and $(x_{\mathcal{C}_2}, w_{\mathcal{C}_2})$ overlap *iff* $|x_{\mathcal{C}_1} - x_{\mathcal{C}_2}| \leq \frac{w_{\mathcal{C}_1}+w_{\mathcal{C}_2}}{2}$. We can however get away with the same linearity trick as before such that `superimposed`$(\mathcal{C}_1,\mathcal{C}_2)$ gives

$$x_{\mathcal{C}_1} - x_{\mathcal{C}_2} \leq \frac{w_{\mathcal{C}_1} + w_{\mathcal{C}_2}}{2} \tag{6}$$

$$x_{\mathcal{C}_2} - x_{\mathcal{C}_1} \leq \frac{w_{\mathcal{C}_1} + w_{\mathcal{C}_2}}{2} \tag{7}$$

(and similarly for y and h parameters).

- `near`$(\mathcal{C}_1,\mathcal{C}_2)$ is more subtle to model linearly, since it is based on a non-linear distance metric. As described in section 2.1.3, proximity is relative to the size of the first argument $\mathcal{C}_1$: $\mathcal{C}_2$ is `near` to $\mathcal{C}_1$, if it is overlaps with a dilation of $\mathcal{C}_1$ with a structural element of size $w_{\mathcal{C}_1} \times h_{\mathcal{C}_1}$. As a consequence, `near` can be expressed using the same constraints as the `superiposed` relation, by using $w'_{\mathcal{C}_1} = 2 \times w_{\mathcal{C}_1}$ and $h'_{\mathcal{C}_1} = 2 \times h_{\mathcal{C}_1}$.

- `far`$(\mathcal{C}_1,\mathcal{C}_2)$ is, unfortunately not appropriately representable in our framework. There is no straightforward, linear way to express the fact that two elements are either `not near`, or explicitly relate the parameters as to model their distance with respect to a given threshold. The linearisation trick used for the `near`, `superimposed` or `inside` predicates cannot be applied here, for the reason that we need to express a $\geq$ relationship, rather than a $\leq$ as it was the case before.

  The solution comes from the objective function we will discuss in the following paragraph.

The inequation set obtained from the set of predicates, combined with the objective function to minimize is then simply given to a linear programming constraint solver [12] making it very straightforward to generate the corresponding image.

In order to express as closely as possible the fact that non-`near` elements should be as far away from each other as possible, we use an objective function that maximizes the dispersion of the elements. Here again, dispersion is a non-linear measure, which we need to approximate.

In our case, the adopted solution is to maximize

$$f(\mathbf{x}) = \left(\sum_{\mathcal{C}_i \neq I} (x_{\mathcal{C}_i} - x_I) + (y_{\mathcal{C}_i} - y_I) \right) - \left(\sum_{\mathcal{C}_i \neq I} w_{\mathcal{C}_i} + h_{\mathcal{C}_i} \right) - w_I - h_I$$

This will tend to minimize the size of the components while "maximizing' their "distance' from the centre of the image.

4 Experimental Results

In order to assess the validity of our approach we first try to synthesize the classification results obtained in section 2.2. The classification rules are given below, followed by the generated, corresponding image.

1. `has_element(A,B), type(B,circle), has_element(A,C),inside(C,B).` 
2. `has_element(A,B), type(B,circle), has_element(A,C), near(B,C).` 
3. `has_element(A,B), type(B,circle), has_element(A,C), superimposed(C,B).` 
4. `has_element(A,B), type(B,circle), has_element(A,C), bigger(B,C).` 
5. `has_element(A,B), type(B,circle), has_element(A,C), bigger(B,C).` 
6. `has_element(A,B), type(B,ccblackthick), has_element(A,C), type(C,cornerse), has_element(A,D), e(D,C), inside(D,B).` 
7. `has_element(A,B), type(B,ccblackthick), has_element(A,C), has_element(A,D), e(D,C), superimposed(D,B).` 
8. `has_element(A,B), type(B,circle), has_element(A,C), e(B,C), type(C,cornerne).` 
9. `has_element(A,B), type(B,ccblackthick), has_element(A,C), has_element(A,D), e(B,C), type(C,cornerse), inside(D,B).` 
10. `has_element(A,B), type(B,ccblackthick), has_element(A,C), has_element(A,D), e(C,D), far(D,C), type(D,cornersw).` 

Overall results are quite encouraging, although we pay the price of excessive linearisation. In item 8, for instance, the system has not been able to produce a sufficient dispersion of the elements to make the `east` relation sufficiently visible.

It is comforting, however, that the issues mentioned in section 2.2 concerning the ambiguity of the classification rules, finally result in identical images.

5 Conclusions

In this paper, the components in an image are represented by their bounding box. This might induce errors with complex shapes. Without loss of generality, components can be approximated by more complex boundaries, while remaining linear. It will have to be established what impact this might have on time complexity, however. Secondly, the relations between components are *approximated* with linear equations, while they are essentially non-linear. This approximation is limiting for more complex shape descriptions. Further work needs to be done to exactly evaluate the impact of this approximation in terms of speed, found optimum or possible singularities and the authors are currently working on a non-linear extension of the minimization algorithm.

Acknowledgements

Part of the experiments described in this paper are due to the Master Project of Karly Langa and Benoît Leoutre, students at ESIAL – Nancy-Université.

References

[1] K.C., S., Lamiroy, B., Ropers, J.P.: Inductive logic programming for symbol recognition. In: 10th International Conference on Document Analysis and Image Recognition, Barcelona, Spain (July 26-29 2009)

[2] K.C., S., Lamiroy, B., Ropers, J.P.: Utilisation de Programmation Logique Inductive pour la reconnaissance de symboles. In: 5ème Atelier ECOI : Extraction de COnnaissance et Images, Strasbourg France, GRCE (2009)

[3] Muggleton, S., Luc, Raedt, D.: Inductive logic programming: Theory and methods. Journal of Logic Programming **19** (1994) 629–679

[4] Lamiroy, B., Gaucher, O., Fritz, L.: Robust Circle Detection. In Flavio Bortolozzi, Robert Sabourin, eds.: 9th International Conference on Document Analysis and Recognition - ICDAR'07, Curitiba Brésil (2007) 526–530

[5] Claveau, V., Sébillot, P.: From efficiency to portability: Acquisition of semantic relations by semi-supervised machine learning. In: 20th International Conference on Computational Linguistics, COLING'04, Geneva, Switzerland (2004) 261–267

[6] Amin, A., Sammut, C., Sum, K.: Learning to recognize hand-printed chinese characters using inductive logic programming. International Journal of Pattern Recognition and Artificial Intelligence **10**(7) (1996) 829–847

[7] Ceci, M., Berardi, M., Malerba, D.: Relational data mining and ilp for document image processing. Applied Artificial Intelligence **21**(8) (2007) 317–342

[8] Mas Romeu, J., Sanchez, G., Llados, J., Lamiroy, B.: An Incremental On-line Parsing Algorithm for Recognizing Sketching Diagrams. In Flavio Bortolozzi, Robert Sabourin, eds.: 9th International Conference on Document Analysis and Recognition - ICDAR'07, Curitiba Brésil (2007) 452–456

[9] Malon, C., Uchida, S., Suzuki, M.: Mathematical symbol recognition with support vector machines. Pattern Recognition Letters **29**(9) (2008) 1326 – 1332

[10] Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62** (2006) 107 – 136

[11] Domingos, P., Kok, S., Lowd, D., Poon, H., Richardson, M., Singla, P.: Markov logic. In: Probabilistic Inductive Logic Programming. (2008) 92–117

[12] Dantzig, G.: Linear Programming and Extensions. Princeton University Press (1998)