ALLPAD: Approximate Learning of Logic Programs with Annotated Disjunctions

Fabrizio Riguzzi

Dipartimento di Ingegneria, Università di Ferrara, Via Saragat 1
44100 Ferrara, Italy, fabrizio.riguzzi@unife.it

Abstract. In this paper we present the system ALLPAD for learning Logic Programs with Annotated Disjunctions (LPADs). ALLPAD modifies the previous system LLPAD in order to tackle real world learning problems more effectively. This is achieved by looking for an approximate solution rather than a perfect one. ALLPAD has been tested on the problem of classifying proteins according to their tertiary structures and the results compare favorably with most other approaches.

1 Introduction

Logic Programs with Annotated Disjunctions [1] are a relatively new formalism for representing probabilistic information in logic programming. They have been recognized as one of the simplest and most expressive languages that combine logic and probability [2].

In [3] the definition of a learning problem for LPADs has been proposed together with an algorithm for solving it called LLPAD. However, LLPAD does not work well on non-toy problems because it relies on the exact solution of a large constraint satisfaction problem. On real world problems such a solution may not exist or may be too expensive to find. Therefore in this paper we propose the system ALLPAD (Approximate Learning of Logic Programs with Annotated Disjunctions) that modifies LLPAD in order to be able to solve real world problems by looking for a solution that "approximately" satisfies the learning problem.

2 ALLPAD

ALLPAD learns ground LPADs in five phases. The first and the third are the same as those of LLPAD. The second and the fourth modify those of LLPAD and the fifth one is new.

In the first and second phases ALLPAD looks respectively for definite and disjunctive clauses that satisfy a number of constraints reported in [3]. In the third phase the disjunctive clauses found in the second phase are annotated with probabilities by exploiting theorem 1 of [3].

In the fourth phase ALLPAD solves an optimization problem in which a subset of the found disjunctive clauses is selected so that the resulting program

assigns to the input interpretations a probability that is as close as possible to the one given. This is done by exploiting theorem 2 of [3]. Since the optimization problem can be expressed as a linear problem, we can use mixed-integer programming (MIP) techniques. If no perfect solution exist, a non zero optimum will be found.

However, the optimization problem is NP-hard and thus solvable only for small instances. To overcome this problem, we exploit the possibility of setting a time limit offered by many MIP packages. In this way, ALLPAD looks for the best solution given the available time.

To make sure that an admissible solution will be found within the time limits, the complete search in the space of bodies performed by LLPAD in the second phase is given up for an incomplete search strategy, beam search. The heuristic to be used for ranking bodies is the sum of the probabilities of the interpretations where the body is true. This heuristic ensures that the clauses that apply only to a small number of improbable interpretations are discarded and the dimension of the optimization problem is reduced.

In the fifth phase, the definite clauses not mutually exclusive with the selected disjunctive clauses are removed.

3 Experiments

ALLPAD was applied to the problem of predicting the tertiary structure of proteins by classifying them into one of the SCOP classes [4]. Each protein is described by a sequence of secondary structure elements. The elements are either of the form *he(Type,Length,Position)* or of the form *st(Orientation,Length,Position)*. The last argument is an ordinal number indicating the position in the sequence.

The dataset available [5] (kindly provided by Kristian Kersting) has the following distribution of examples into classes (class,examples): (fold1, 721), (fold2, 360), (fold23, 274), (fold37, 441), (fold55, 290).

ALLPAD can be used for classification as other probabilistic model learners: a model is learned for each class and an example is assigned the class whose model gives the highest probability to the example.

In order to learn an LPAD that describes a class, the interpretations given as input to the system are annotated each with the same probability given by 1 over the total number of interpretations in the training set.

Proteins are modeled with LPADs as stochastic processes: the structure at position p is predicted on the basis of the structures in a number of previous positions. To this purpose, ALLPAD learns programs containing rules having all the possible structures with position equal to p in the head and a conjunction of structures in the body with positions belonging to the set $S(p,k) = \{p-1, p-2, \ldots, p-k\}$ for a given k.

Since the constraint solving phase finds only an approximate solution, the theories learned are tested in an approximate way: if for a sequence position no learned rule is applicable, the marginal probability of the atom in the class is used.

The accuracy of the learned LPAD is compared with the accuracy of a naïve Bayes classifier obtained in the following way: the approximate testing procedure is applied by using for all positions the marginal probability in the class.

Two experiments were performed using 10-fold cross validation.

In both experiments we used Xpress-Optimizer by Dash Optimization for solving the MIP problem. The time limit has been set to 1 hour for each class in the first experiment and to 100 minutes for each class in the second experiment.

The other important parameters are: the value of k (the number of previous positions to consider), set to 4; the size of the beam, set to 100, and the maximum number of bodies to be explored for each clause template, which has been set to 100 in the first experiment and to 125 in the second experiment. The experiments have been performed on a PC with an Athlon XP 2600+ processor at 2138 Mhz, 1GB of RAM and Windows 2000.

In the two experiments ALLPAD reached respectively an average accuracy of 85.14% and of 85.67%, while the naive Bayes approach reached an average accuracy of 82.79%. A cross-validated paired two-tailed t test was performed for comparing the accuracy of ALLPAD to that of naïve Bayes and the null hypothesis of equivalence can be rejected with 98.3% probability for the first experiment and with 98.6% probability for the second experiment.

The results available in the literature regarding accuracy on datasets in the same domain are: 74% in [5], 83.6% in [6], 76% and 73% in [7] and 92.96% in [8]. The results of ALLPAD compare favorably with all results apart from the last one, even if the system is not specifically tailored to learning sequences.

References

1. Vennekens, J., Verbaeten, S., Bruynooghe, M.: Logic programs with annotated disjunctions. In: The 20th International Conference on Logic Programming (ICLP 2004). (2004)
2. Blockeel, H.: Probabilistic logical models for mendel's experiments: An exercise. In: Inductive Logic Programming (ILP 2004), Work in Progress Track. (2004)
3. Riguzzi, F.: Learning logic programs with annotated disjunctions. In: Inductive Logic Programming, (ILP 2004). Number 3194 in LNCS, Springer (2004) 270–287
4. Turcotte, M., Muggleton, S., Sternberg, M.J.E.: The effect of relational background knowledge on learning of protein three-dimensional fold signatures. Machine Learning **43**(1/2) (2001) 81–95
5. Kersting, K., Raiko, T., Kramer, S., Raedt, L.D.: Towards discovering structural signatures of protein folds based on logical hidden markov models. In: Pacific Symposium on Biocomputing. (2003) 192–203
6. Kersting, K., Gärtner, T.: Fisher kernels for logical sequences. In: Machine Learning, (ECML 2004). Number 3201 in LNCS, Springer (2004) 205–216
7. Kersting, K., Raedt, L.D., Raik, T.: Logical hidden markov models. Journal of Artificial Intelligence Research **25** (2006) 425–456
8. Gutmann, B., Kersting, K.: TildeCRF: Conditional random fields for logical sequences. In: Machine Learning, (ECML 2006). LNCS, Springer (2006)