Lifted Discriminative Learning of Probabilistic Logic Programs

Arnaud Nguembang Fadja Fabrizio Riguzzi

Dipartimento di Ingegneria – University of Ferrara

Dipartimento di Matematica e Informatica – University of Ferrara
[fabrizio.riguzzi,arnaud.nguembangfadja]@unife.it

Introduction

- Probabilistic logic programming (PLP) is a powerful tool for reasoning with uncertain relational models.
- Learning probabilistic logic programs is expensive due to the high cost of inference.
- Lifted inference improves the performance by reasoning on whole populations of individuals
- Example [De Raedt, Kimmig, ML15]
  $popular(X) : p :- friends(X, Y), famous(Y).$
  $P(popular(john)) = 1 - (1 - p)^n$ where n is the number of famous friends of *john*.
- Cost logarithmic in n, as computing a^n is $\Theta(\log n)$ with the "square and multiply" algorithm [Gordon, JA98]

Overview

- Liftable PLP: programs contain clauses with a single annotated atom in the head with the same predicate.
- Discriminative parameter learning by using EM or by optimizing the likelihood with Limited-memory BFGS (LBFGS)
- A previous approach for performing lifted learning [Haren et al, ML16] targeted generative learning (no negative examples) for Markov Logic Networks
- LIFTCOVER for "LIFTed slipCOVER" performs discriminative structure learning obtained from SLIPCOVER [Bellodi, Riguzzi, TPLP15] by simplifying structure search and replacing parameter learning

Liftable PLP

- PLP under the distribution semantics
- We allow only clauses of the form

$$C_i = h_i : \Pi_i \ :- \ b_{i1}, \ldots, b_{iu_i}$$

  where all the clauses share the same predicate ($target/a$) for the single atom in the head
- The literals in the body have predicates other than $target/a$ and are defined by facts and rules that are certain, i.e., they have a single atom in the head with probability 1.

Example

- UW-CSE domain, he objective is to predict the "advised by" relation

$$advisedby(A, B) : 0.4 :-$$
$$\quad student(A), professor(B), publication(C, A), publication(C, B).$$
$$advisedby(A, B) : 0.5 :-$$
$$\quad student(A), professor(B), ta(C, A), taughtby(C, B).$$

Inference

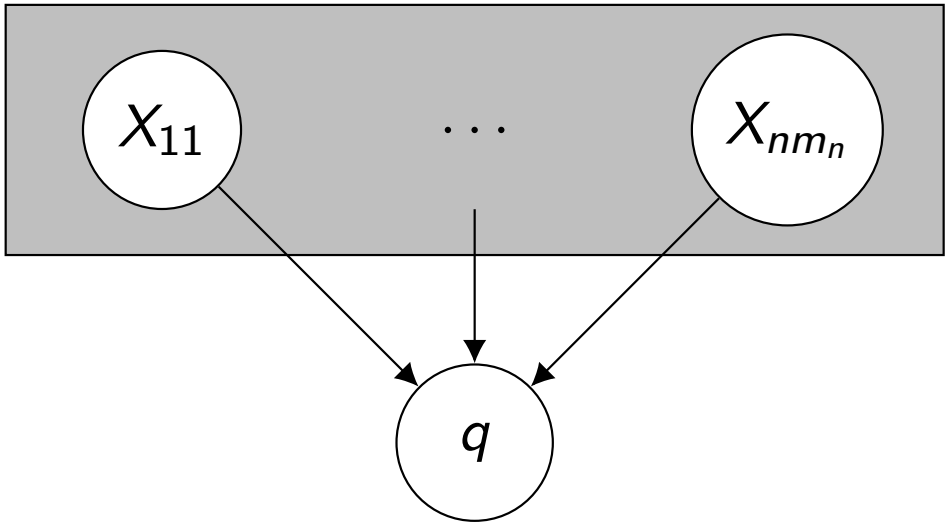
- Computing the probability of a ground atom q of $target/a$
- Find the number of ground instantiations of clauses for $target/a$ such as the body is true and the head is equal to q.
- Suppose there are m_i such instantiations $\{\theta_{i1}, \ldots, \theta_{im_i}\}$, for rule C_i for $i = 1, \ldots, n$.
- Each instantiation θ_{ij} corresponds to a random variable X_{ij} taking values 1 with probability Π_i and 0 with probability $1 - \Pi_i$.
- The query q is true if at least one of the random variables for a rule takes value 1:

$$q = true \Leftrightarrow \bigvee_{i=1}^{n} \bigvee_{j=1}^{m_i} (X_{ij} = 1)$$

- $P(q) = 1 - \prod_{i=1}^{n} (1 - \Pi_i)^{m_i}$.

Inference

- Dependence of the random variable q from the variables of clause groundings:



- The CPT of q is that of an or.
- The CPT of clause variables is $P(X_{ij} = 1) = \Pi_i$ and $P(X_{ij} = 0) = 1 - \Pi_i$.

Example

- UW-CSE domain, he objective is to predict the "advised by" relation
  $advisedby(A, B) : 0.4 :-$
  $\quad student(A), professor(B), publication(C, A), publication(C, B).$
  $advisedby(A, B) : 0.5 :-$
  $\quad student(A), professor(B), ta(C, A), taughtby(C, B).$
- $q = advisedby(harry, ben)$ where *harry* is a student, *ben* is a professor, they have 4 joint publications and *ben* teaches 2 courses where *harry* is a TA. Then
  $P(advisedby(harry, ben)) = 1 - (1 - 0.4)^4(1 - 0.5)^2 = 0.9676.$

Parameter Learning

- Given a liftable PLP T, a set $E^+ = \{e_1, \ldots, e_Q\}$ of positive examples (ground atoms) and a set $E^- = \{e_{Q+1}, \ldots, e_R\}$ of negative examples (ground atoms) and a background knowledge B, find the parameters of T such that the likelihood

$$L = \prod_{q=1}^{Q} P(e_q) \prod_{r=Q+1}^{R} P(\neg e_r)$$

is maximized.

Parameter Learning

- Likelihood expanded

$$L = \prod_{q=1}^{Q} \left(1 - \prod_{l=1}^{n} (1 - \Pi_l)^{m_{lq}}\right) \prod_{r=Q+1}^{R} \prod_{l=1}^{n} (1 - \Pi_l)^{m_{lr}} \quad (1)$$

  where m_{iq} (m_{ir}) is the number of instantiations of C_i whose head is e_q (e_r) and whose body is true and n is the number of clauses.

- We can aggregate the negative examples

$$L = \prod_{l=1}^{n} (1 - \Pi_l)^{m_{l-}} \prod_{q=1}^{Q} \left(1 - \prod_{l=1}^{n} (1 - \Pi_l)^{m_{lq}}\right) \quad (2)$$

  where $m_{l-} = \sum_{r=Q+1}^{R} m_{lr}$.

Expectation Maximization

- We can maximize L using an Expectation Maximization (EM) algorithm since the X_{ij} variables are hidden.
- We need the following conditional probabilities

$$P(X_{ij}=1|e)=\frac{\Pi_i}{1-\prod_{i=1}^{n}(1-\Pi_i)^{m_i}} \quad P(X_{ij}=0|e)=1-\frac{\Pi_i}{1-\prod_{i=1}^{n}(1-\Pi_i)^{m_i}}$$
$$P(X_{ij}=1|\neg e)=0 \qquad P(X_{ij}=0|\neg e)=1$$

- EM alternates between Expectation, where the expectation of the counts are computed, and Maximization, where the parameters are updated.

Gradient Ascent

- The partial derivative of the likelihood with respect to Π_i is:

$$\frac{\partial L}{\partial \Pi_i} = \frac{L}{1-\Pi_i}\left(\sum_{q=1}^{Q} m_{iq}\left(\frac{1}{P(e_q)}-1\right) - m_{i-}\right) \tag{3}$$

- The equation $\frac{\partial L}{\partial \Pi_i} = 0$ does not admit a closed form solution, not even where there is a single clause, so we must use optimization to find the maximum of L.
- Optimization by Limited-memory BFGS (LBFGS) that requires the computation of L and of its derivative at various points.

Structure Learning: LIFTCOVER

- Simplified version of SLIPCOVER [Bellodi, Riguzzi, TPLP15]
- First identify good clauses guided by the log likelihood (LL) of the data.
- Clauses are found by a top-down beam search.
- The refinement operator adds a literal to the body of the clause C taken from a bottom clause built as in Progol
- Each refinement Cl' is scored by performing parameter learning with $T = \{Cl'\}$ and using the resulting LL as the heuristic.
- The scored refinements are inserted back into the beam in order of heuristic.

Structure Learning: LIFTCOVER

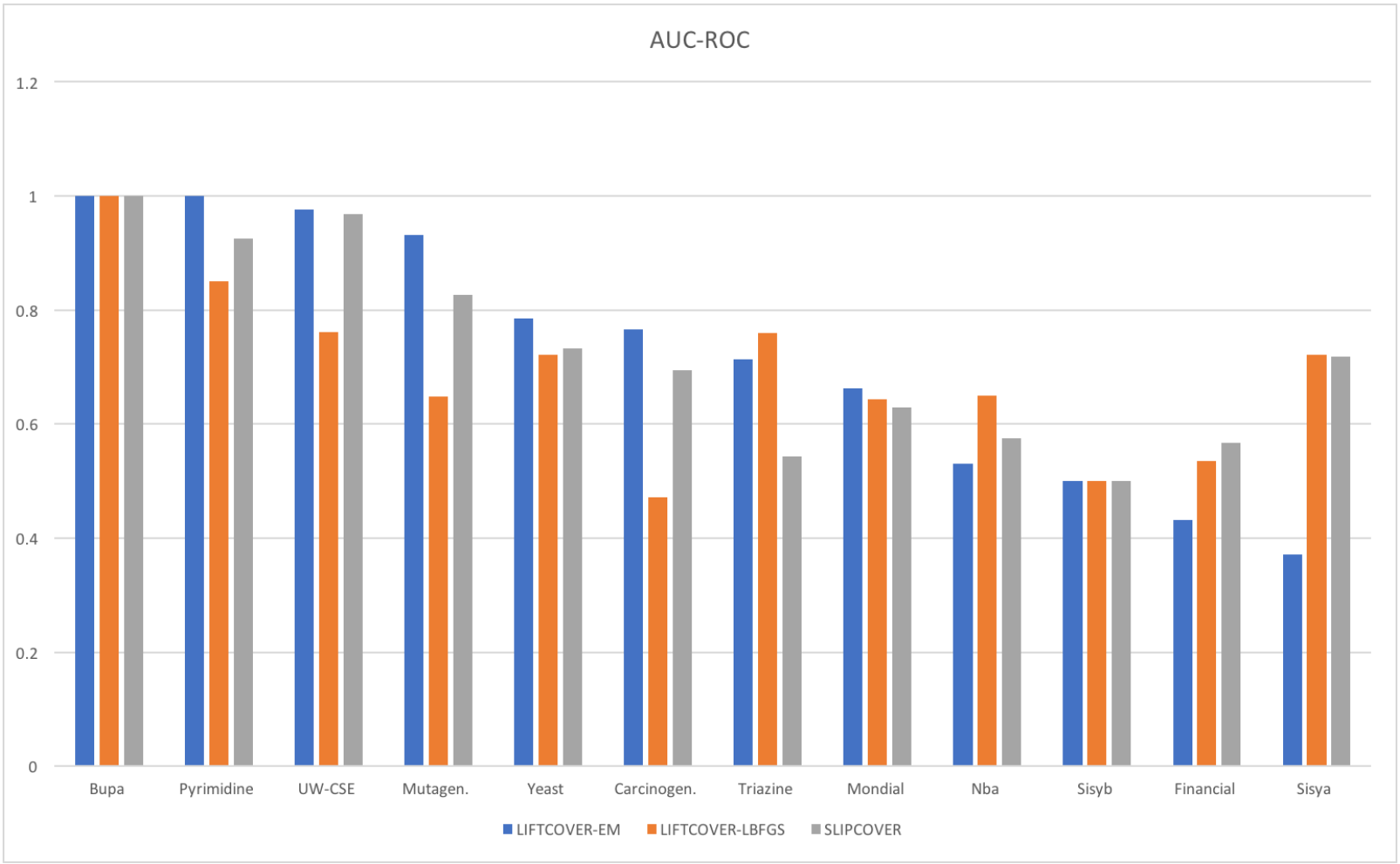
- The refinements are also added to a set of clauses CC.
- Beam search is iterated a user-defined number of times or until the beam becomes empty.
- Then parameter learning is applied to the whole set CC, i.e., $T = CC$.

Experiments

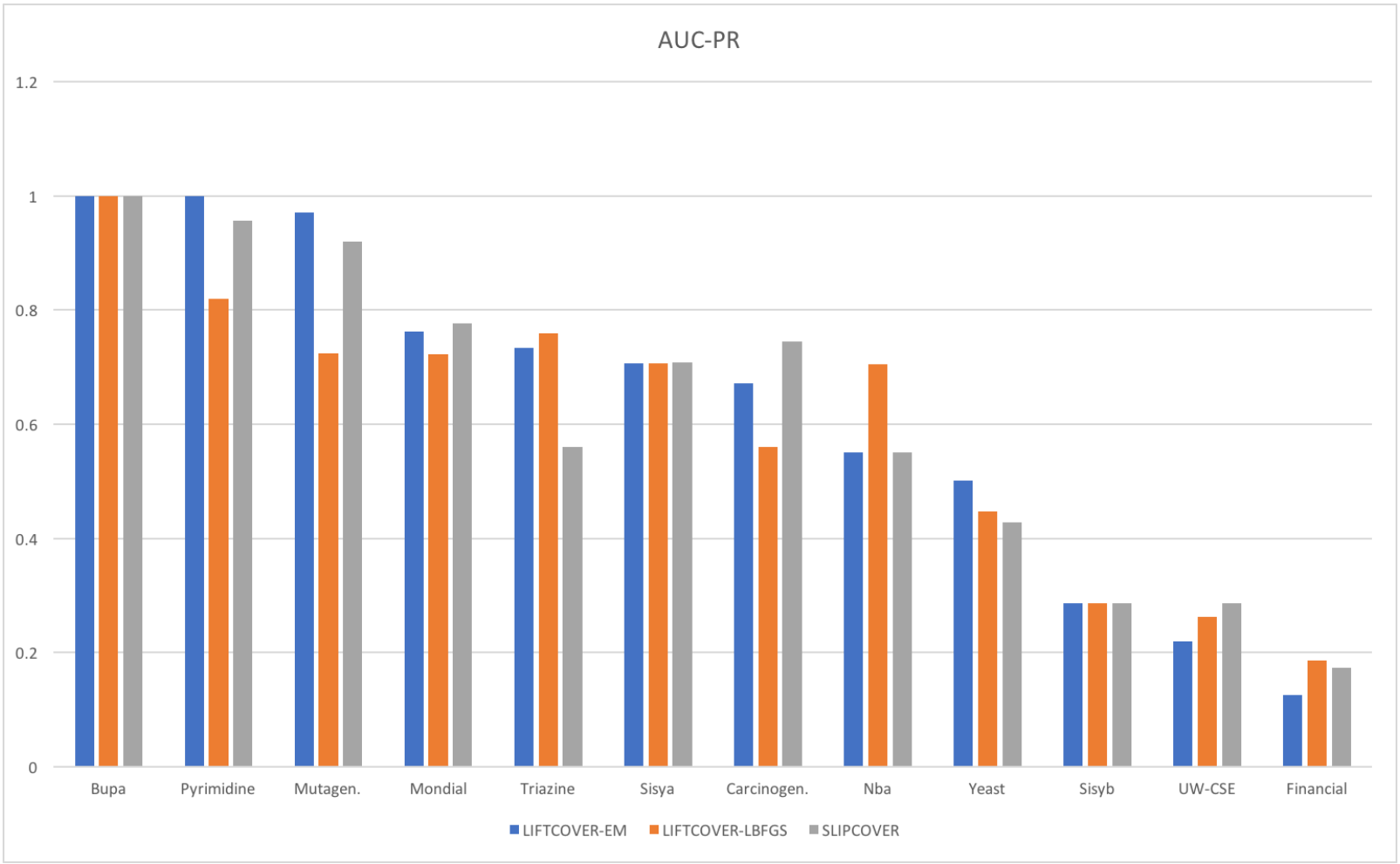
Dataset	P	T	PEx	NEx	F
UW-CSE	15	2673	113	20680	5
Mutagen.	20	15249	125	126	10
Carcinogen.	36	24533	182	155	1
Mondial	11	10985	572	616	5
Bupa	12	2781	145	200	5
Nba	4	1218	15	15	5
Pyrimidine	29	2037	20	20	4
Triazine	62	10079	20	20	4
Financial	9	92658	34	223	10
Sisya	9	358839	10723	6544	10
Sisyb	9	354507	3705	9229	10
Yeast	12	53988	1299	5456	10

Datasets for the experiments: number of predicates (P), of tuples (T) (i.e., ground atoms), of positive (PEx) and negative (NEx) examples for target predicate(s), of folds (F). The number of tuples includes the target positive examples.

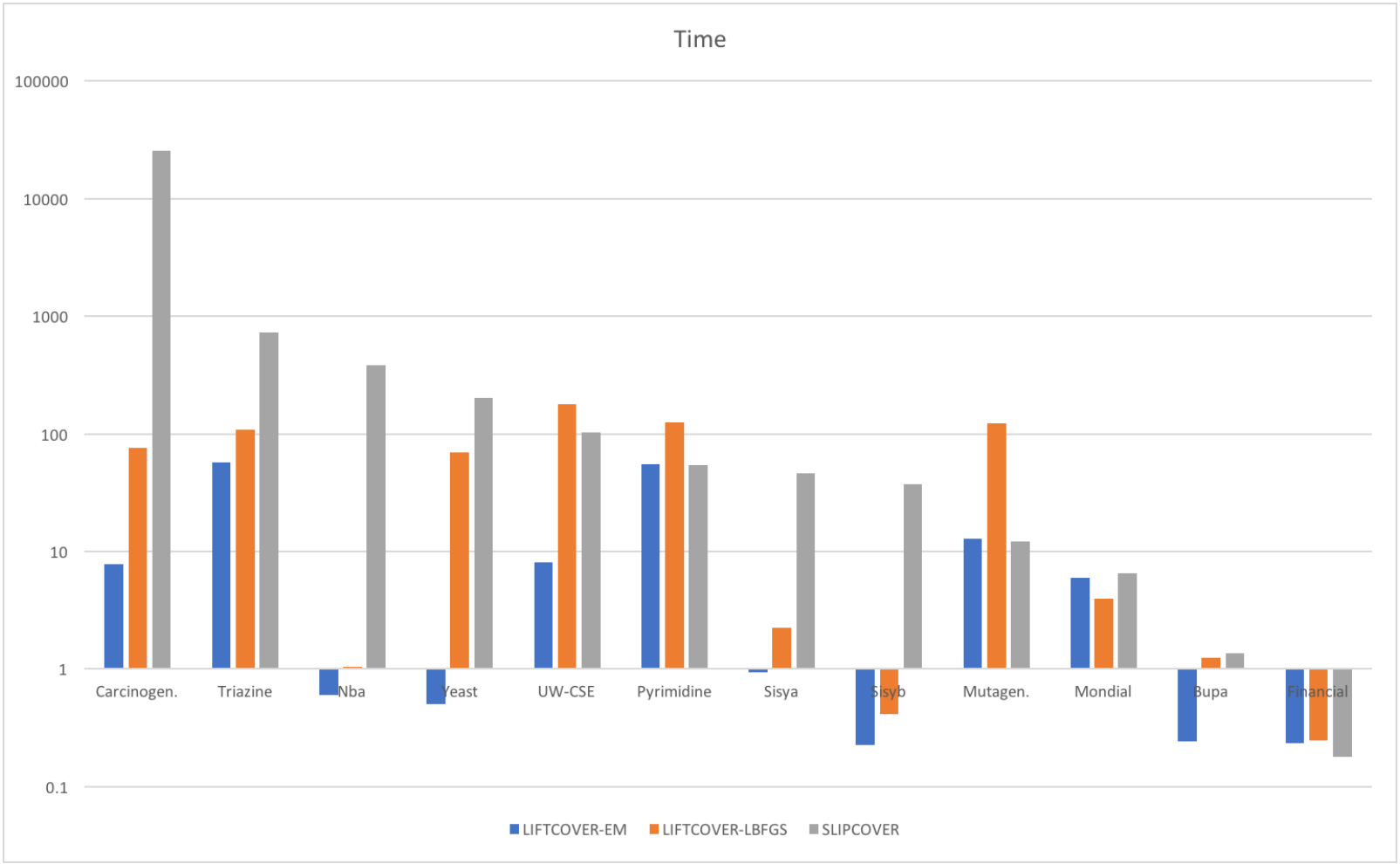
Experiments: AUC-ROC



Experiments: AUC-PR



Experiments: Time



Conclusions and Future Work

- Conclusions
  - LIFTCOVER learns the structure of liftable probabilistic logic programs using either EM or LBFGS for parameter learning.
  - LIFTCOVER-EM is faster than SLIPCOVER and often more accurate,
  - LIFTCOVER-LBFGS is slightly slower and slightly less accurate.
- Future work
  - Test the influence of settings in LBFGS
  - Add regularization to parameter learning in order to avoid overfitting.

THANKS FOR LISTENING AND ANY QUESTIONS ?