An Introduction to Statistical Relational Learning — Part 1

medium.com/@vishy_punditry/an-introduction-to-statistical-relational-learning-part-1-cafc19fb4979

Vishal Bhalla August 16, 2016

> **Statistical Relational Learning** (**SRL**) is an emerging field and one that is taking centre stage in the Data Science age. Big Data has been one of the primary reasons for the continued prominence of this relational learning approach given, the voluminous amount of data available now to learn interesting and unknown patterns from data. Moreover, the tools have also improved their processing prowess especially, in terms of scalability.

This introductory blog is a prelude on SRL and later on I would also touch base on more advanced topics, specifically Markov Logic Networks (MLN). To start off, let's look at how SRL fits into one of the 5 different Machine Learning paradigms.

Five Machine Learning Paradigms

Lets look at the 5 Machine Learning Paradigms: Each of which is inspired by ideas from a different field!

1. Connectionists as they are called and led by Geoffrey Hinton (University of Toronto & Google and one of the major names in the Deep Learning community) think that a learning algorithm should mimic the brain! After all it is the brain that does all the complex actions for us and, this idea stems from Neuroscience.
2. Another group of Evolutionists whose leader is the late John Holland (from the University of Michigan) believed it is not the brain but evolution that was precedent and hence the master algorithm to build anything. And using this approach of having the fittest ones program the future they are currently building 3D prints of future robots.
3. Another thought stems from Philosophy where Analogists like Douglas R. Hofstadter an American writer and author of popular and award winning book — Gödel, Escher, Bach: an Eternal Golden Braid believe that Analogy is the core of Cognition.
4. Symbolists like Stephen Muggleton (Imperial College London) think Psychology is the base and by developing Rules in deductive reasoning they built ***Adam*** — a robot scientist at the University of Manchester!
5. Lastly we have a school of thought which has its foundations rested on Statistics & Logic, which is the ***focal point of interest*** in this blog. This emerging field has started to gain prominence with the invention of Bayesian networks 2011 by Judea Pearl (University of California Los Angeles — UCLA) who was awarded with the Turing award (the highest award in Computer Science). Bayesians as they are called, are the most fanatical of the lot as they think everything can be represented by the Bayes theorem using hypothesis which can be updated based on new evidence.

SRL fits into the last paradigm of Statistics and Logic. As such it offers another alternative to the now booming Deep Learning approach inspired from Neuroscience.

Background

In many real world scenario and use cases, often the underlying data is assumed to be independent and identically distributed (i.i.d.). However, real world data is not and instead consists of many relationships. SRL as such attempts to represent, model, and learn in the relational domain!

There are 4 main Models in SRL

1. Probabilistic Relational Models (PRM)
2. Markov Logic Networks (MLN)
3. Relational Dependency Networks (RDN)
4. Bayesian Logic Programs (BLP)

It is difficult to cover all major models and hence the focus of this blog is only on the emerging field of Markov Logic Networks.
MLN is a powerful framework that combines statistics (i.e. it uses Markov Random Fields) and logical reasoning (first order logic).

Academia

Some of the prominent names in academic and the research community in MLN include:

1. Professor Pedro Domingos from the University of Washington is credited with introducing MLN in his paper from 2006. His group created the tool called Alchemy which was one of the first, First Order Logic tools.
2. Another famous name—Professor Luc De Raedt from the AI group at University of Leuven in Belgium, and their team created the tool ProbLog which also has a Python Wrapper.
3. HAZY Project (Stanford University) led by Prof. Christopher Ré from the InfoLab is doing active research in this field and Tuffy, Felix, Elementary, Deep Dive are some of the tools developed by them. More on it later!
4. Talking about academia close by i.e. in Germany, Prof. Michael Beetz and his entire team moved from TUM to TU Bremen. Their group invented the tool—ProbCog
5. At present, Prof. Volker Tresp from Ludwig Maximilians University (LMU), Munich & Dr. Matthias Nickles at Technical University of Munich (TUM) have research interests in SRL.

Theory & Formulation

A look at some background and theoretical concepts to understand MLN better.

A. Basics—Probabilistic Graphical Models (PGM)

The definition of a PGM goes as such:

A PGM encodes a joint p(x,y) or conditional p(y|x) probability distribution such that given some observations we are provided with a full probability distribution over all feasible solutions.

A PGM helps to encode relationships between a set of random variables. And it achieves this by making use of a graph! These graphs can be either be Directed or Undirected Graphs.

B. Markov Blanket

A Markov Blanket is a Directed Acyclic graph. It is a Bayesian network and as you can see the central node A highlighted in red is dependent on its parents and parents of descendents (moralization) by the circle drawn around it. Thus these nodes are the **only** knowledge needed to predict node A.

C. Markov Random Fields (MRF)

A MRF is an Undirected graphical model. Every node in an MRF satisfies the Local Markov property of Conditional Independence, i.e. a node is conditionally independent of another node, given its neighbours. And now relating it to Markov Blanket as explained previously, a Markov blanket for a node is simply its adjacent nodes!

Intuition

We now that Probability handles uncertainty whereas Logic handles complexity. So why not make use of both of them to model relationships in data that is both uncertain and complex. Markov Logic Networks (MLN) precisely does that for us!

MLN is composed of a set of pairs of <w, F> where F is the formula (written in FO logic) and weights (real numbers identifying the strength of the constraint). MLN basically provides a template to ground a Markov network. Grounding would be explained in detail in the next but one section on “Weight Learning”. It can be defined as a Log linear model

where probability of a world is given by the weighted sum of all true groundings of a formula *i* under an exponential function. It is then divided by Z which is termed as the partition function and used to normalize and get probability values between 0 and 1.

The MLN Template

Rules or Predicates
The relation to be learned is expressed in FO logic. Some of the different possible FO logical connectives and quantifiers are And (^), Or (V), Implication (→), and many more. Plus, Formulas may contain one or more predicates, connected to each other with logical connectives and quantified symbols.

Evidence
Evidence represent known facts i.e. the ground predicates. Each fact is expressed with predicates that contain only constants from their corresponding domains.

Weight Learning
Discover the importance of relations based on grounded evidence.

Inference
Query relations, given partial evidence to infer a probabilistic estimate of the world.

More on Weight Learning and Inference in the next part of this series!

Hope you enjoyed the read. I have deliberately kept the content basic and a mix of non technical and technical so as to highlight first the key players and some background concepts and generate the reader's interest in this topic, the technicalities of which can easily be read in the paper. Any feedback as a comment below or through a message are more than welcome!

Continue reading with An Introduction to Statistical Relational Learning — Part II.

References

- Richardson, Matthew, and Pedro Domingos — Markov logic networks. In Machine learning, vol. 62, no. 1–2, pp. 107–136, 2006.
- Pedro's TEDx video at University of Washington: The Quest for the Master Algorithm
- Hazy project webpage

Note: *Originally published at data-science-blog.com on August 17, 2016.*

An Introduction to Statistical Relational Learning — Part 2

medium.com/@vishy_punditry/an-introduction-to-statistical-relational-learning-part-2-ba6e6dc23644

Vishal Bhalla January 21, 2017

> In the first part of this series on "An Introduction to Statistical Relational Learning", I touched upon the basic Machine Learning paradigms, some background and intuition of the concepts and concluded with how the MLN template looks like. In this blog, we will dive in to get an in depth knowledge on the MLN template; again with the help of sample examples. I would then conclude by highlighting the various toolkit available and some of its differentiating features.

MLN Template — explained

A Markov logic network can be thought of as a group of formulas incorporating first-order logic and also tied with a weight. But what exactly does this weight signify?

Weight Learning

According to the definition, it is the log odds between a world where F is true and a world where F is false,

$$\log(\text{odds}) = \text{logit}(P) = \ln\left(\frac{P}{1-P}\right)$$

and captures the marginal distribution of the corresponding predicate.

Each formula can be associated with some weight value, that is a positive or negative real number. The higher the value of weight, the stronger the constraint represented by the formula. In contrast to classical logic, all worlds (i.e., Herbrand Interpretations) are possible with a certain probability [1]. The main idea behind this is that the probability of a world increases as the number of formulas it violates decreases.

Markov logic networks with its probabilistic approach combined to logic posit that a world is *less likely* if it violates formulas unlike in pure logic where a world is false if it violates even a single formula. Consider the case when a formula with high weight i.e. more significance is violated implying that it is less likely in occurrence.

Another important concept during the first phase of Weight Learning while applying an MLN templateis **"Grounding"**. Groundingmeans to replace each variable/function in predicate with constants from the domain.

Weight Learning — An Example

Note: All examples are highlighted in the Alchemy MLN format.

Let us consider an example where we want to identify the relationship between 2 different types of verb-noun pairs i.e noun subject and direct object.

Command

alchemy-2/bin$ **./learnwts -i** predicateFormula.mln **-o** learnwts.mln **-t** SVO.db **-ne** nsubj, dobj -noAddUnitClauses

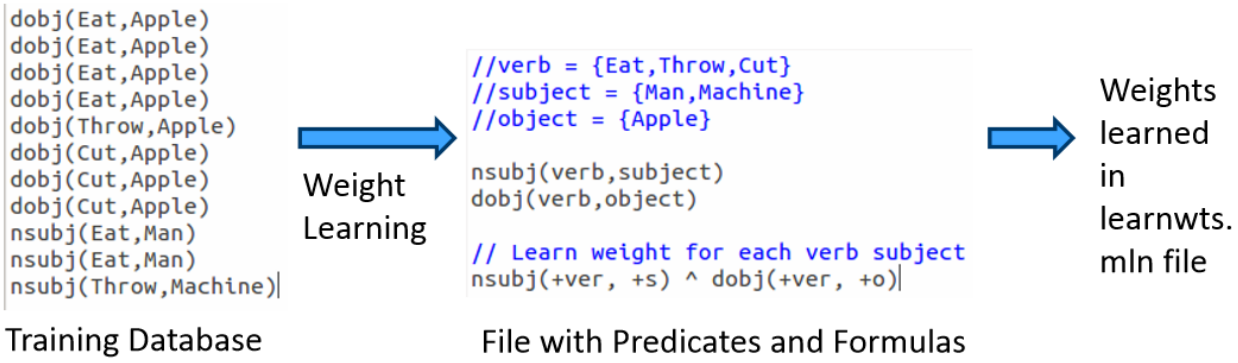
where,

-i Input .mln file containing the predicates and formulas

-o Output .mln file containing formulas with learned weights.

-t Training database containing atom values as true groundings

-ne Non evidence predicates



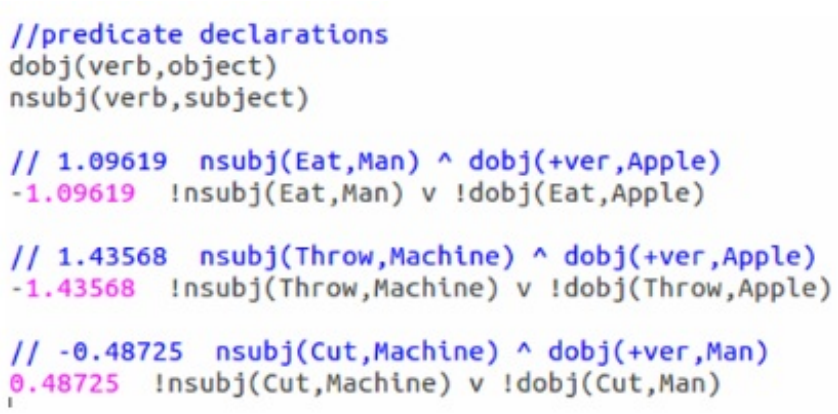
The input *predicateFormula.mln* file contains

a. The predicates nsubj(verb, subject) and dobj(verb, object) and

b. Formula of nsubj(+ver, +s) and dobj(+ver, +o)

These predicates or rules are to learn all possible SVO combinations i.e. what is the probability of a Subject-Verb-Object combination. The + sign ensures a cross product between the domains and learns all combinations. The training database consists of the nsubj and dobj tuples i.e. relations is the evidence used to learn the weights.

When we run the above command for this set of rules against the training evidence, we learn the weights as here:



Weights learned in learnwts.mln file

Note that the formula is now grounded by all occurrences of nsubj and dobj tuples from the training database or evidence and the weights are attached to it at the start of each such combination.

But it should be noted that there is no network yet and this is just a set of weighted first-order logic formulas. The MLN template we created so far will generate Markov networks from all of our ground formulas. Internally, it is represented

as a factor graph.where each ground formula is a factor and all the ground predicates found in the ground formula are linked to the factor.

Inference

The definition goes as follows:

Estimate probability distribution encoded by a graphical model, for a given data (or observation).

Out of the many Inference algorithms, the two major ones are MAP & Marginal Inference. For example, in a MAP Inference we find the most likely state of world given evidence, where y is the query and x is the evidence.

which is in turn equivalent to this formula.

$$y_{MAP} = \arg\max_{y \in Y} P(y \mid x)$$

$$\arg\max_{y} \sum_{i} w_i n_i(x, y)$$

Another is the Marginal Inference which computes the conditional probability of query predicates, given some evidence. Some advanced inference algorithms are Loopy Belief Propagation, Walk-SAT, MC-SAT, etc.

The probability of a world is given by the weighted sum of all true groundings of a formula i under an exponential function, divided by the partition function *Z* i.e. equivalent to the sum of the values of all possible assignments. The partition function acts a normalization constant to get the probability values between 0 and 1.

Probability of a world x:

$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x) \right)$$

Weight of formula *i*

No. of true groundings of formula *i* in *x*

Inference — An Example

Let us draw inference on the the same example as earlier.

Command

alchemy-2/bin$ **./infer -i** learnwts.mln **-r** result.mln **-e** SVO.db **-q** nsubj, dobj

where,

-i Input .mln file containing the learned weights for the predicates

-r Output .mln file with the probability for each predicate

-e Database containing the ground atoms as evidences

-q Query on the predicates

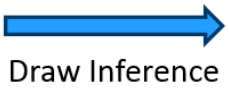
```
//predicate declarations
dobj(verb,object)
nsubj(verb,subject)

// 1.09619  nsubj(Eat,Man) ^ dobj(+ver,Apple)
-1.09619  !nsubj(Eat,Man) v !dobj(Eat,Apple)

// 1.43568  nsubj(Throw,Machine) ^ dobj(+ver,Apple)
-1.43568  !nsubj(Throw,Machine) v !dobj(Throw,Apple)

// -0.48725  nsubj(Cut,Machine) ^ dobj(+ver,Man)
0.48725  !nsubj(Cut,Machine) v !dobj(Cut,Man)
```

Weights learned in learnwts.mln file




```
nsubj(Eat,Machine) 0.194031
nsubj(Throw,Apple) 0.29502
nsubj(Cut,Man) 0.367013
nsubj(Cut,Machine) 0.384012
nsubj(Cut,Apple) 0.363014
```

Inference infer.mln file

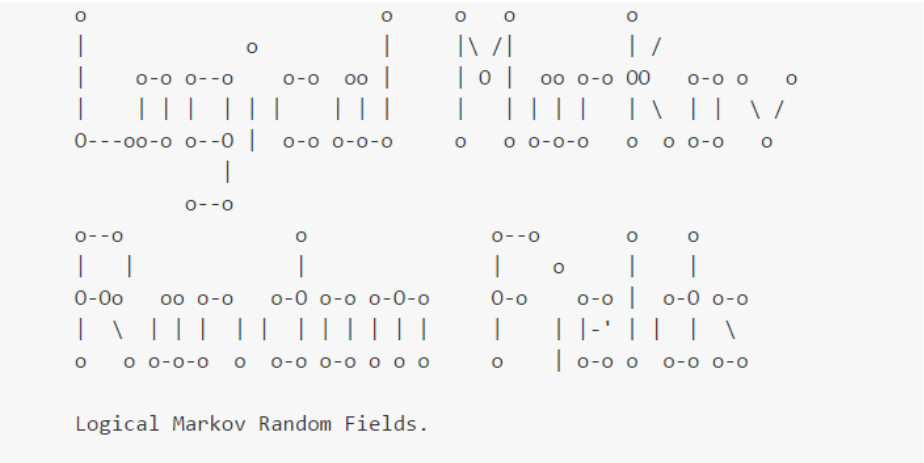
After learning the weights we run inference (with or without partial evidence) and query the relations of interest (nsubj here), to get inferred values.

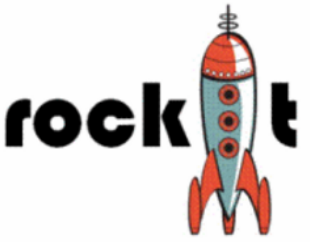
Tool-kits

Let's look at some of the MLN tool-kits at disposal to do learning and large scale inference. I have tried to make an assorted list of all tools here and tried to highlight some of its main features & problems.

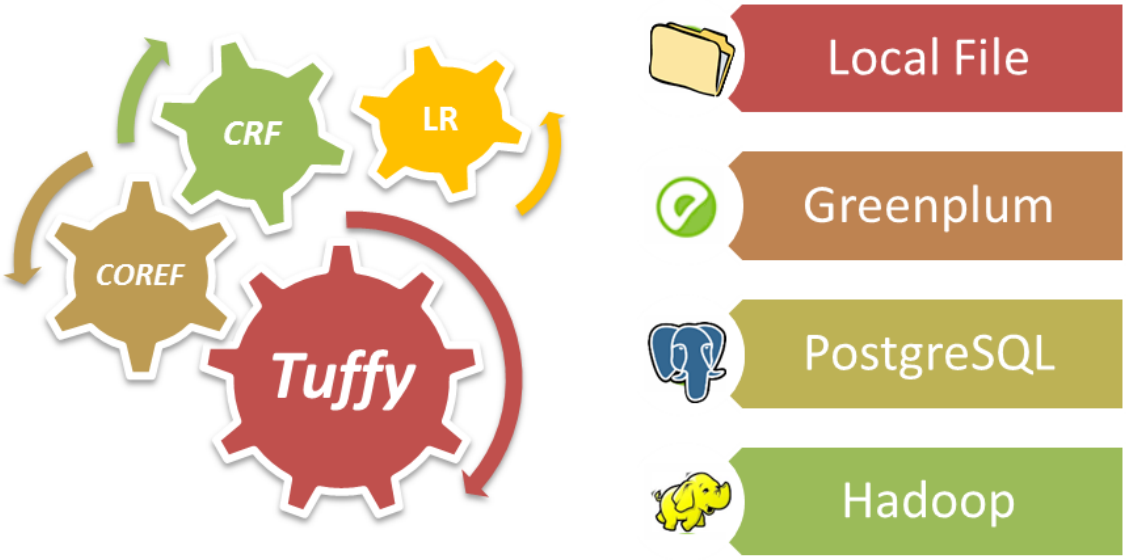
For example, BUGS i.e. **Bayesian Logic** uses a *Swift Compiler* but is **Not relational**! ProbLog has a Python wrapper and is based on *Horn clauses* but has **No Learning** feature. These tools were invented in the initial days, much before the present day MLN looks like.

ProbCog developed at Technical University of Munich (TUM) & the AI Lab at Bremen covers not just MLN but also *Bayesian Logic Networks (BLNs), Bayesian Networks & ProLog*. In fact, it is now GUI based. Thebeast gives a *shell* to analyze & inspect model feature weights & missing features.





LoMRF: Logical Markov Random Fields



Some of the various toolkits

Alchemy from University of Washington (UoW) was the 1st First Order (FO) probabilistic logic toolkit. RockIt from University of Mannheim has an online & rest based interface and uses only *Conjunctive Normal Forms (CNF)* i.e. **And-Or** format in its formulas.

Tuffy scales this up by using a *R*elational *D*ata*b*ase *M*anagement *S*ystem (RDBMS) whereas Felix allows Large Scale inference! Elementary makes use of secondary storage and Deep Dive is the current state of the art. All of these tools are part of the HAZY project group at Stanford University.

Lastly, LoMRF i.e. Logical Markov Random Field (MRF) is *Scala* based and has a feature to analyse different *hypothesis* by comparing the difference in .mln files!

> Hope you enjoyed the read. The content starts from basic concepts and ends up highlighting key tools. In the final part of this 3 part blog series I would explain an application scenario and highlight the active research and industry players. Any feedback as a comment below or through a message is more than welcome!

References and Additional Links:

[1] Knowledge base files in Logical Markov Random Fields (LoMRF)

[2] (still) nothing clever Posts categorized “Machine Learning” — Markov Logic Networks

[3] A gentle introduction to statistical relational learning: maths, code, and examples

Note: *Originally published at data-science-blog.com on January 18, 2017.*

Back to Part I — An Introduction to Statistical Relational Learning