# Using the Knowledge of a Domain Expert to Train Markov Logic Networks

Tivadar Papai advisor: Shalini Ghosh

September 1, 2011

# Markov Logic - I

- A probabilistic first-order logic (FOL)
- Knowledge Base (KB) is a set of weighted FOL formulas $W = \{\ldots(w_i, F_i)\ldots\}$
- The probability of a truth assignment $x$ to the ground atoms:

$$\Pr(X = x|w) = \frac{1}{Z(w)} \exp(\sum_i w_i n_i(x))$$

where $w_i$ is the weight of $F_i$ (the $i$th formula in the KB) and $n_i(X)$ is the number of true groundings of $F_i$

# Markov Logic - II

- Weights do not have an intuitive meaning (cannot translate them directly into probabilities, only in the simplest cases)
- Weights are usually set to maximize the probability of the training data

# Motivation

- What if there is not sufficient training data available?
- What if there is no training data at all, but we can rely on the knowledge of domain experts?
- A domain expert can tell how likely it is that a randomly chosen instantiation of a formula in the KB holds
- E.g., the domain expert can know the statistics what percentage of the population smokes, how likely it is that smoking causes lung cancer, etc.
- The domain expert has this information available in the form of subjective probabilities of FOL formulas

# Exponential Families of Probability Distributions - I

- The probability distribution defined by an MLN can be written in the form:

$$\Pr(X = x|\theta) = \exp\left(\langle\theta, f(x)\rangle - A(\theta)\right)$$

- $f_i(x) \equiv n_i(x)$, $\theta \equiv w$, $A = \log Z$
- An exponential family of probability distributions

# Exponential Families of Probability Distributions - II

- $\theta$ - natural parameters
- $\mu = \mathbb{E}_\theta[f(x)] = \sum_x f(x) Pr(X = x|\theta)$ - mean parameters
- There is a many-to-one mapping from $\theta$ to $\mu$, let $m$ be this mapping ($m(\theta) = \mu$)
- For every $\theta$ there is a $\mu = m(\theta)$, but it is not true that for every $\mu$ there is a $\theta$ which maps to it ($\mu$ is inconsistent in this case)
- $\Pr(X = x|\theta) = \Pr(X = x|m(\theta))$, i.e., either $\theta$ or $\mu = m(\theta)$ can determine the probability distribution

# Formalization - First Attempt

- Let $\overline{\mu}_i = \frac{\mu_i}{g_i}$, where $g_i$ is the number of groundings of the $i$th formula ($\overline{\mu}_i$ - a randomly chosen grounding of $F_i$ being true)
- If $s$ is the subjective probability vector given by the expert we can try to find a $\theta$ for which $\mu = m(\theta)$ and $\overline{\mu} = s$
- If $\mu$ is inconsistent, we cannot do this
- E.g., $f_1 = P(x)$, $f_2 = P(x) \vee Q(x)$ then for $s_1 = 1.0$, $s_2 = 0.5$ there does not exist any $\theta$ s.t. $\overline{\mu}_1 = s_1$ and $\overline{\mu}_2 = s_2$
- We need to soften the constraint
- When training data is available we have to take that also into account

# Prior on $\mu$

- In MLNs prior has only been put on $\theta$ so far (Gaussian prior with 0 mean)
- Truncated Gaussian ($\mu \in [0, 1]$):

$$\Pr(\mu) \propto \exp\left(-\alpha(\overline{\mu} - s)^T(\overline{\mu} - s)\right) = \exp\left(-\alpha \sum_i (\overline{\mu}_i - s_i)^2\right)$$

# Log-likelihood, Gradient

- The log-likelihood:

$$L = \log Pr(D|\mu) + \log \Pi(\mu) = \sum_{i=1}^{N} \log Pr(D_i|\mu) + \log \Pi(\mu)$$

- The gradient of $L$ w.r.t. $\theta$:

$$\begin{aligned}\frac{\partial L}{\partial \theta} &= \frac{\partial \log Pr(D|\theta)}{\partial \theta} + \frac{\partial \log \Pi(\mu)}{\partial \mu}\frac{\partial \mu}{\partial \theta} \\ &= \sum_{i=1}^{N}\left(f(d_i) - \mu\right) + \Sigma_\theta \frac{\partial \log \Pi(\mu)}{\partial \mu} \\ &= \sum_{i=1}^{N}\left(f(d_i) - \mu\right) + \alpha' \Sigma_\theta \left(s - \overline{\mu}\right)\end{aligned}$$

where $\Sigma_\theta$ is the covariance matrix

# How to Find the Optimum?

- The optimization problem is non-convex in general, however when we have sufficient amount of data then it is
- Simple gradient ascent is slow, can stay in a non-global optimum, can suffer from ill-conditioning
- L-BFGS directly cannot be used, because MC-SAT (slice sampling algorithm for MLNs) provides noisy results, it could also get stuck in local optima

# Demo

# Future Goals

- Improve the existing implementation (all formulae must be given as clauses, gradient ascent's parameters should be part of the user's input, etc.)
- Modify L-BFGS or find another library which could work with noisy MC-SAT results
- Try out different priors
- Give theoretical (probabilistic) guarantees for the usefulness of subjective probabilities (assume expert knows the mean parameters of the features up to some error)
- Apply in the medical and fault tolerance domains

Thank you for your attention!