PSL 简介

一、 什么是 PSL

概率软逻辑(PSL)是由马里兰大学和加利福尼亚大学圣克鲁斯分校的统计关系学习组 LINQS 开发的机器学习框架。PSL 是一种概率编程语言，并且在自然语言处理，社会网络分析和计算机视觉等许多领域都产生了不错的结果。它可以被应用在很多机器学习相关的问题上，如链路预测、实体对齐等

PSL 结合了两个强大的理论的优势： 一阶逻辑，具有简洁地表示复杂现象的能力；概率图模型，可以捕捉真实世界知识中固有的不确定性和不完备性。细节上，PSL 使用“软”逻辑作为其逻辑组成部分，以马尔可夫网络作为其统计模型。

一阶逻辑：

一阶逻辑（First Order Logic），简称 FOL

- 包含的东西有常量（Constant symbol），谓词符号（Predicate symbol），函数符号（Function symbol），变量（Variable），连词（ ∧∨→↔），量词（Quantifiers, ∃∀），例如：
- Father(Mary) = Bob
- father_of(Mary, Bob)

概率图模型（PGM）：

概率图模型大致可以分为两种，directed graphical model(又称贝叶斯网络）和 undirected graphical model（又称马尔可夫随机场）。贝叶斯网络由 Judea Pearl 教授发明于上世界 80 年代，这项工作获得了 2011 年图灵奖。马尔可夫随机场最早被物理学家用于对原子进行建模，其中的代表作 Ising model 获得过诺贝尔奖。图灵奖+诺贝尔奖，PGM 的重要性可见一斑。另外，PGM 是将人工智能（AI）的研究热点从传统 AI(如逻辑、推理、知识表示）转向机器学习的重要工作（其他起到这一作用的工作有支持向量机、决策树、boosting 等)。概率图模型在实际中（包括工业界）的应用非常广泛与成功。这里举几个例子。隐马尔可夫模型（HMM）是语音识别的支柱模型，高斯混合模型（GMM）及其变种 K-means 是数据聚类的最基本模型，条件随机场（CRF）广泛应用于自然语言处理（如词性标注，命名实体识别)，Ising 模型获得过诺贝尔奖，话题模型在工业界大量使用(如腾讯的推荐系统)。等等。PGM 优雅的理论。机器学习的一个核心任务是从观测到的数据中挖掘隐含的知识，而概率图模型是实现这一任务的一种很 elegant，principled 的手段。PGM 巧妙地结合了图论和概率论。从图论的角度，PGM 是一个图，包含结点与边。结点可以分为两类：隐含结点和观测结点。边可以是有向的或者是无向的。从概率论的角度，PGM 是一个概率分布，图中的结点对应于随机变量，边对应于随机变量的 dependency 或者 correlation 关系。给定一个实际问题，我们通常会观测到一些数据，并且希望能够挖掘出隐含在数据中的知识。怎么用 PGM 实现呢？我们构建一个图，用观测结点表示观测到的数据，用隐含结点表示潜在的知识，用边来描述知识与数据的相互关系，最后获得一个概率分布。给定概率分布之后，通过进行两个任务：inference（给定观测结点，推断隐含结点的后验分布）和 learning(学习这个概率分布的参数)，来获取知识。PGM 的强大之处在于，不管数据和知识多复杂，我们的处理手段是一样的：

建一个图，定义一个概率分布，进行 inference 和 learning。这对于描述复杂的实际问题，构建大型的人工智能系统来说，是非常重要的。

概率软逻辑中的软逻辑指 逻辑结构不需要被严格的限制为 0 或 1，可以是 0-1 之间的某个值，例如下面的逻辑公式：

```
similarName(X, Y) -> sameEntity(X, Y)
```

它表达的逻辑意义可以理解为，如果 X 和 Y 具有相似甚至相同的 name，那么我们可以说 X 和 Y 可能是同一个人，而 similarName(X, Y)的结果是 0-1 之间的某个值，具体的逻辑符号通过以下形式定义：

```
与：A && B = max{A + B - 1, 0}
或：A || B = min{A + B, 1}
非：~A = 1 - A
```

如果将 A、B 的值严格限制为 0 或 1，那上面的公式所表达的结果就和传统的逻辑规则一致。

在 PSL 模型中，这些具体的逻辑公式将成为马尔科夫网络的特征，并且网络中的每个特征都会与一个权重相关联，决定它在特征之间相互作用的重要性。权重可以手动设置或是基于已有真实数据通过学习算法学习得到。PSL 还提供了复杂的推理技术，同时利用软逻辑的特点将知识推理的复杂度优化到多项式时间，而不再是一个 NP-HARD 问题。

二、 PSL 的用途

下面通过一个实体分类的例子来介绍 PSL 的用途，程序的主要功能是根据已有的事实数据去推测每个人的居住地，主要有以下 3 个步骤：

1. 定义一个隐含模型
2. 为模型提供数据
3. 推理发现未知数据

模型的定义：在 PSL 中模型是通过一组逻辑规则去表示的，一般定义在*.psl 文件中

```
10: Knows(P1,P2) & Lives(P1,L) -> Lives(P2,L) ^2
20: Knows(P2,P1) & Lives(P1,L) -> Lives(P2,L) ^2
2: ~Lives(P,L) ^2
```

以上模型直观上表达了，**相互认识的人更有可能居住在相同的地方**。以第一条规则为例，规则开头的整数代表该规则所占权重，规则尾部的平方数是用于平滑权重的。

装载数据：PSL 模型中的规则由一系列谓词构成，而谓词最终将会被对应于该条逻辑的具体数据所替换，通过*.data 文件定义了数据的路径

```
predicates:
  Knows/2: closed
  Lives/2: open

observations:
  Knows : ../data/test/knows_obs.txt
```

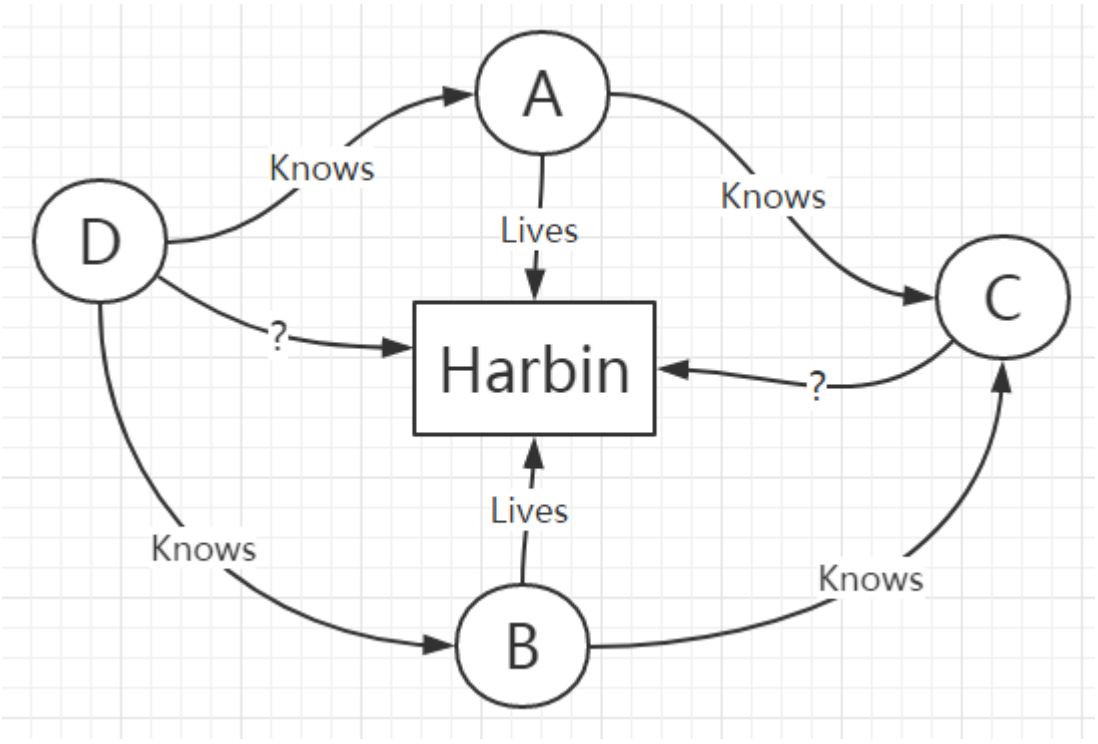
```
  Lives : ../data/test/lives_obs.txt

targets:
  Lives : ../data/test/lives_targets.txt

truth:
  Lives : ../data/test/lives_truth.txt
```

predicates 部分列举出了我们所定义规则中的所有谓词（Knows，Lives），关键词 open 表示该谓词对应的事实数据没有被完全观测到，希望得到推理补全。closed 表示该谓词对应的事实数据是完整的，它将会作为预测推理的依据。在我们的例子中，我们已经充分观察到任意两个人之间知否相互认识。因此，Knows 都是封闭的谓词。但是我们只了解某些人认识（或不认识）某些人，希望来推断其他人之间相互认识的情况。

Predictes 数据位置 /data/lives_obs.txt 、/data/likes_obs.txt
Open 推测关系数据 数据位置 /data/knows_obs.txt
Targets 目标检测数据 数据位置/data/knows_targets.txt
Truths 实际数据关系 数据位置/data/knows_truths.txt



observations 部分列出了每个谓词对应的事实数据文件的路径 knows_obs.txt 文件内容如下：

```
A   C
B   C
D   A
D   B
```

其中每一行表示一条观测数据，第一行表示 A 认识 B，即 A 和 B 存在 Knows

关系；
lives_obs.txt 文件内容如下：

```
A	哈尔滨
B	哈尔滨
```

其中每一行表示一条观测数据，第一行表示 A 居住在哈尔滨，即 A 和哈尔滨存在 lives 关系。

targets 部分指向的文件用于定义模型需要预测的所有可能出现的关系，lives_targets.txt 文件内容如下：

```
C	哈尔滨
D	哈尔滨
```

其中每一行表示一条待预测的关系，需要由 psl 模型根据观测数据以及规则给出关系成立的具体概率值（0-1 之间）。

truth 部分表示对于那些 open 的 predicates 将要进行预测的真实结果，可以用来评价模型的预测结果以及训练参数。

推理预测：基于以上模型和数据我们就可以使用 PSL 进行推理预测，它帮我们完成了所有细节上的工作。

```
zhm@aiiskey-05-win MINGW64 /d/code/psl-examples/collective_classifcati
$ ./run.sh
Jar found cached, skipping download.
0    [main] INFO  org.linqs.psl.cli.Launcher  - Loading data
82   [main] WARN  org.linqs.psl.database.rdbms.DataStoreMetadata  - De
92   [main] INFO  org.linqs.psl.cli.Launcher  - Data loading complete
92   [main] INFO  org.linqs.psl.cli.Launcher  - Loading model
178  [main] INFO  org.linqs.psl.cli.Launcher  - Model loading complete
178  [main] INFO  org.linqs.psl.cli.Launcher  - Starting inference
206  [main] INFO  org.linqs.psl.application.inference.MPEInference  -
238  [main] INFO  org.linqs.psl.application.inference.MPEInference  -
254  [main] INFO  org.linqs.psl.reasoner.admm.ADMMReasoner  - Optimiza
254  [main] INFO  org.linqs.psl.application.inference.MPEInference  -
266  [main] INFO  org.linqs.psl.cli.Launcher  - Inference Complete
LIVES('A', '哈尔▒?') = 1.0
LIVES('B', '哈尔▒?') = 1.0
LIVES('C', '哈尔▒?') = 0.9086945520387539
LIVES('D', '哈尔▒?') = 0.9517787811691728
```

三、 如何使用 PSL

1. 环境配置

JDK 环境：PSL 主要使用 Java 语言开发完成（包含部分 Grovvy 脚本）

Maven 环境：Maven 3.x：PSL 使用 maven 管理构建依赖。

配置文件：配置文件 psl.properties 可以用定义一些具体的属性值，某些内置的一些属性 https://github.com/linqs/psl/wiki/Configuration-Options

2. 使用方式

（1） 命令行窗口：https://github.com/linqs/psl/wiki/Using-the-CLI：

具体步骤：

a. JDK 环境配置，参考 1 上方

b. 下载安装 CLI

```
git clone https://bitbucket.org/linqs/psl-examples.git
cd psl-examples/link_prediction/easy/cli

安装 jar 到 psl-examples/link_prediction/easy/cli
```

c. 运行

```
java -jar psl-cli-CANARY.jar -infer -model simple_lp.psl -data
simple_lp.data
```

（2） PSL 提供了 Grovvy 编程接口：https://github.com/linqs/psl/wiki/Using-the-Groovy-Interface

a. JDK 环境配置，参考 1 上方

b. 模型规则

1)． 调用 add()方法，以参数写入规则、权值、平方

```
model.add(
   rule: ( Likes(A, 'Dogs') & Likes(B, 'Dogs') ) >> Friends(A, B),
   weight: 5.0,
   squared: true
);
```

行语法：根据 Groovy 语法，支持逻辑运算，不支持算术运算；变量命名以大写字母开头。

2).调用 addRules()

```
// Load multiple rules from a single string.
model.addRules("""
   1: ( Likes(A, 'Dogs') & Likes(B, 'Dogs') ) >> Friends(A, B) ^2
   Likes(A, +B) = 1 .
""");

// Load multiple rules from a file.
model.addRules(new FileReader("myRules.txt"));
```

3).限制：对无权重的限制

```
// An unweighted rule (constraint) explicitly specified with a period.
model.add(
   rule: "Likes(A, +B) = 1 ."
);

// An unweighted rule (constraint) implicitly specified by not adding a weight.
model.add(
   rule: "Likes(A, +B) = 1"
);
```

c. 构建数据库

1).连接数据库

```
DataStore data = new RDBMSDataStore(new
H2DatabaseDriver(Type.Disk, "database/path", true),
configBundle);
```

参数说明：DBMSDataStore 有 2 个参数，H2DatabaseDriver 链接数据库，有 3 个参数 ， Type.Disk 数据存储在磁盘 ， path 磁盘路径，boolean ， 是否清除同一路径任何数据库的内容。configBundle 额外的设置

2).读取数据

```
Inserter insert = data.getInserter(<predicateName>, <partition>);
InserterUtils.loadDelimitedData(<filePath>);
```

参数说明：< partition> 数据写入分区、 predicateName 要读取地面原子的谓词的名称、filePath 原始数据路径

（3） Java Application https://github.com/linqs/psl/wiki/Using-the-Java-Interface

a. JDK 环境配置，参考 1 上方

b. Pom.xml 配置

Application pom.xml

```
<dependencies>
    ...
    <dependency>
        <groupId>org.linqs</groupId>
        <artifactId>psl-core</artifactId>
        <version>2.0.0</version>
    </dependency>
    ...
</dependencies>
```

Maven 仓库 pom.xml

```
<repositories>
        <repository>
                <releases>
                        <enabled>true</enabled>
                        <updatePolicy>daily</updatePolicy>
                        <checksumPolicy>fail</checksumPolicy>
                </releases>
                <id>psl-releases</id>
                <name>PSL Releases</name>
                <url>http://maven.linqs.org/maven/repositories/psl-releases/</url>
                <layout>default</layout>
        </repository>
</repositories>
```

4. PSL 在某些领域的应用成果

我们在 LINQS 团队主页上选取了一篇医学研究方向的论文为例（https://linqspub.soe.ucsc.edu/basilic/web/Publications/2014/fakhraei:tcbb14/）。该论文主要将 PSL 模型应用在了药物靶标相互作用识别预测上，该模型的主要逻辑是：相似的药物倾向于作用在相同的靶标上，同时相似的靶标也倾向于被同一种药物影响。以下是该模型在现有数据集的应用结果和目前该领域 state-of-the art 的 Perlman`s 方法的比较。

TABLE 6: Comparison with Perlman's method using ten-fold cross validation

Methods	AUC	AUPR	P@130
Perlman et al. [8]	**0.937±0.018**	0.564±0.050	0.594±0.040
PSL triads $k = 5$	0.920±0.016	**0.617±0.048**	**0.616±0.035**
PSL triads $k = 15$ & excl. tetrads $k = 5$	**0.937±0.012**	0.585±0.056	**0.616±0.039**

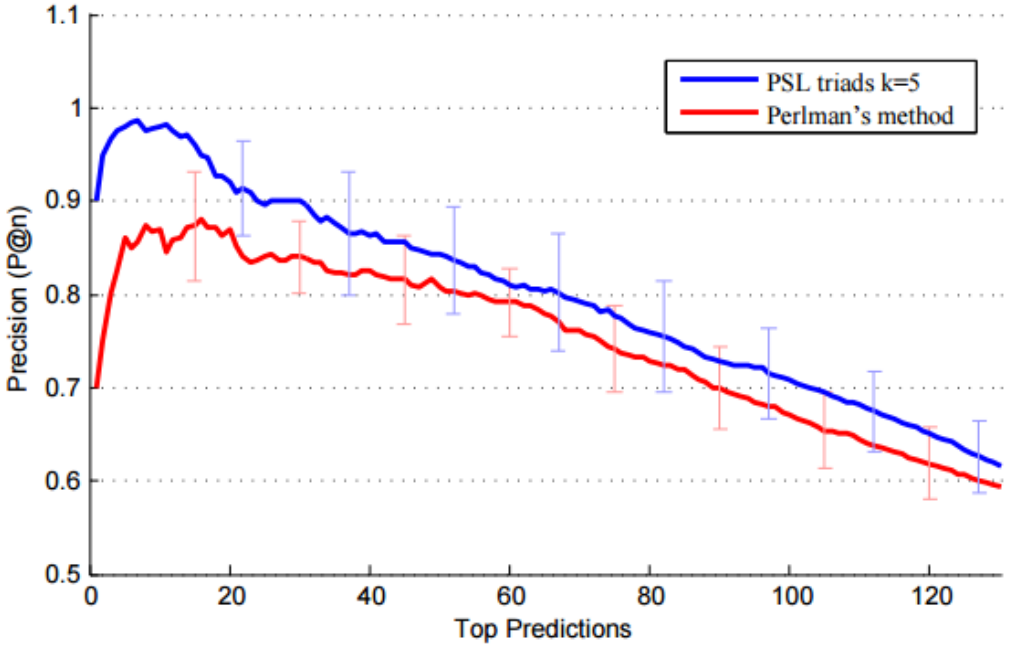


Fig. 11: Comparing Perlman's method with PSL's top 130 predictions using ten-fold cross validation.

QA:

1. `Knows/2: open` 中 2 代表该规则参数个数

2. 能否适用中文?
   存在编码问题，实验后发现对于某些中文字符存在编码不兼容的问题

参考资料：https://github.com/linqs/psl
http://psl.linqs.org/index.html