Content

1. Tensor Logic Programming

This results in this section are from [1]

We consider a language $\mathcal{L}$ that contains a finite set of propositional variables and the logical connectives $\neg$, $\wedge$, $\vee$ and $\leftarrow$. $\mathcal{L}$ contains $\top$ and $\bot$ representing true and false, respectively.

Given a logic program P, the set of all propositional variables appearing in P is the *Herbrand base* of P (written B_P)

1.1 Horn Logic Program

a) Definition:

A *Horn (logic) program* is a finite set of *rules* of the form:

$$h \leftarrow b_1 \wedge \ldots \wedge b_m \ (m \geq 0) \qquad (1)$$

Where h and b_i are propositional variables (also called atoms) in L. In particular, the rule (1) is a *fact* if "$h \leftarrow \top$" and (1) is a constraint if "$\bot \leftarrow b_1 \wedge \ldots \wedge b_m$". The fact and the constraint are simply written as "$h \leftarrow$" and "$\leftarrow b_1 \wedge \ldots \wedge b_m$", respectively.

For each rule r of the form (1), the left-hand side of ← is the *head* and the right-hand side is the *body*:

$$head(r) = h \text{ and } body(r) = \{b_1, \ldots, b_m\}$$

A *definite program* is a Horn program that contains no constraints.

Let P be a Horn program.

A set $I \subseteq B_P$ is an *interpretation* of P.

An interpretation I is a *model* of P if $\{b_1,\ldots, b_m\} \subseteq I$ implies $h \in I$ for every rule (1) in P and $\perp \notin I$.

A model I is the *least model* of P if $I \subseteq J$ for any model J of P.

P is *inconsistent* if it has no model.

A mapping $T_P : 2^{B_P} \to 2^{B_P}$ is defined as $T_P(I) = \{h \mid h \leftarrow b_1 \wedge \ldots \wedge b_m \in P \text{ and } \{b_1,\ldots b_m\} \subseteq I\}$

The powers of T_P are defined as:

$$T_P^{k+1}(I) = T_P\left(T_P^k(I)\right) \text{ and } T_P^0(I) = I$$

Given $I \subseteq B_P$, there is a *fixpoint* $T_P^{n+1}(I) = T_P^n(I) \quad (n \geq 0)$

For a definite program *P*, the fixpoint $T_P^n(\varnothing)$ coincides with the least model of P.

We consider a Horn program P such that for any two rules r_1 and r_2 in P:

$head(r_1) = head(r_2)$ implies $|body(r_1)| \leq 1$ and $|body(r_2)| \leq 1$

(called the *multiple definitions (MD) condition*)

It means: if two different rules have the same head, those rules contain at most one atom in their bodies.

Lemma: Every Horn program is converted to this class of programs by a simple program transformation.

Proof:

Given a Horn program P, define a set $Q_P \subseteq P$ such that:

$$Q_P = \{r \in P \mid head(r) = p \text{ and } |body(r)| > 1\}$$

Let $Q_P = \{p \leftarrow \Gamma_1,\ldots, p \leftarrow \Gamma_k\}$ where Γ_i is the conjunction in the body of each rule.

Then Q_P is transformed to:

$$Q_p' = \{p_1 \leftarrow \Gamma_1,\ldots, p \leftarrow \Gamma_k\} \cup \{p \leftarrow p_1,\ldots, p \leftarrow p_k\}$$

Where p_i are new propositional variables associated with p.

Let P' be the program obtained from P by replacing $Q_P \subseteq P$ with Q_p' for every $p \in B_P$ satisfying the condition:

$$|\{r \in P \mid head(r) = p\}| > 1 \text{ and } Q_P \neq \varnothing$$

Then, P has the least model M iff P' has the least model M' such that $M' \cap B_P = M$

b) Represent interpretations and programs in vector spaces:

Definition 1.1 (vector representation of interpretations)

Let P be a Horn program and $B_P = \{p_1,\ldots, p_n\}$. Then an interpretation I of P is represented by a vector $\boldsymbol{v} = (a_1, \ldots, a_n)^T$ where each element a_i represents the truth value of the proposition p_i such that $ai = 1$ if $p_i \in I$ $(1 \leq i \leq n)$ or $p_i = \top$; otherwise, $ai = 0$. In particular, $I = \varnothing$ is represented by $\boldsymbol{v}0$.

We write $raw_i(\boldsymbol{v}) = p_i$.

Definition 1.2 ($\leq$)

Let *P* be a Horn program and $B_P = \{p_1,\ldots, p_n\}$. Suppose two vectors $\boldsymbol{v} = (a_1,., a_n)^T$ and $\boldsymbol{w} = (b_1, \ldots, b_n)^T$ representing interpretations *I* and *J*, respectively.

Then $\boldsymbol{v} \leq \boldsymbol{w}$ if $ai \leq bi$ for every $1 \leq i \leq n$.

A vector $\boldsymbol{v} = (a_1, \ldots, a_n)^T$ representing an interpretation $I \subseteq B_P$ is *minimal* if $\boldsymbol{w} \leq \boldsymbol{v}$ implies $\boldsymbol{v} \leq \boldsymbol{w}$ for any $\boldsymbol{w}$ representing $J \subseteq B_P$.

Proposition 1.1: For $\boldsymbol{v} = (a_1,\ldots, a_n)^T$ and $\boldsymbol{w} = (b_1, \ldots, b_n)^T$,

$\mathbf{v} \leq \mathbf{w}$ iff $a_i = a_i \,.\, b_i$ for i = 1,…, n.

Definition 1.3 (matrix representation of Horn programs)

Let *P* be a Horn program and $B_P = \{p_1,\ldots, p_n\}$. Then P is represented by a matrix $M_P \in \mathbb{R}^{n\times n}$ such that for each element *aij* $(1 \leq i, j \leq n)$ in M_P,

1. $aij = 1$ if $pi = \top$ or $pj = \bot$
2. $a_{ij_k} = \frac{1}{m}$ $(1 \leq k \leq m;\ 1 \leq i, j_k \leq n)$ if $p_i \leftarrow p_{j_1} \wedge \ldots \wedge p_{j_m}$ is in P.
3. Otherwise, $aij = 0$.

We write $raw_i(M_P) = p_i$ and $col_j(M_P) = p_j$.

In M_P, the *i*-th row corresponds to the atom *pi* appearing in the head of a rule, and the *j*-th column corresponds to the atom *pj* appearing in the body of a rule:

- The first condition says that every element in the *i*-th raw is 1 if $raw_i(M_P) = \top$ and every element in the *j*-th column is 1 if $col_j(M_P) = \bot$.
- The second condition says if there is a rule $p_i \leftarrow p_{j_1} \wedge \ldots \wedge p_{j_m}$ in *P*, then each a_{ij_k} in the *i*-th row and the *jk*-th column has the value 1/m.
- The remaining elements are set to $aij = 0$.

Example 1.1

(1) Consider $P_1 = \{p \leftarrow q, p \leftarrow r, q \leftarrow r\}$ with $B_{P1} = \{p, q, r\}$. Then $M_{P1} \in R^{3\times3}$ is the matrix where

The 1st raw "011" represents two rules $p \leftarrow q$ and $p \leftarrow r$,

The 2nd raw "001" represents the rule $q \leftarrow r$

	p	q	R
p	0	1	1
q	0	0	1
r	0	0	0

(2) Consider P2 = { p ← q ∧ r, q ← , ← p } with B_{P2} = {p, q, r, ⊤ , ⊥}. Then $\mathbf{M}_{P2} \in R^{5\times5}$ is the matrix.

The 1st raw "0 ½ ½ 0 1" represents p ← q ∧ r and p ← ⊥(≡ ⊤).

The 2nd raw "00011" represents q ← ⊤ (≡ q ←) and q←⊥ (≡ ⊤).

The 4th raw "11111" represents ⊤ ←p, ⊤ ← q, ⊤ ← r, ⊤ ← ⊤, and ⊤ ← ⊥, which are all equivalent to ⊤.

The 5th raw "10001" represents ⊥ ←p (≡← p) and ⊥←⊥ (≡ ⊤).

	P	q	r	⊤	⊥
p	0	½	½	0	1
q	0	0	0	1	1
r	0	0	0	0	1
⊤	1	1	1	1	1
⊥	1	0	0	0	1

Suppose a matrix $M_P \in \mathbb{R}^{n\times n}$ representing a Horn program. Given a vector $v \in \mathbb{R}^n$ representing an interpretation I ⊆ B_P, let $M_P \bullet v = (a_1,...,a_n)^T$. We transform $M_P \bullet v$ to a vector $w = (a_1',...,a_n')^T$ where a'i = 1 ($1 \le i \le n$) if ai ≥ 1; otherwise, a'i = 0. We write $w = M_p \circ v$.

Proposition 1.2: Let P be a Horn program and $M_P \in \mathbb{R}^{n\times n}$ its matrix representation. Let $v \in \mathbb{R}^n$ be a vector representing an interpretation I ⊆ B_P. Then $w \in \mathbb{R}^n$ is a vector representing J = T_P(I) iff $w = M_p \circ v$

Proof:

Let $M_P = (a_{ij})$ and $v = (y_1,...,y_n)^T$ then $M_P \bullet v = (x_1,...,x_n)^T$ with $x_k = a_{k1}y_1 + ... + a_{kn}y_n$

+ Suppose $w = M_p \circ v = (x_1',...,x_n')^T$ with $x_k' = 1 \Leftrightarrow x_k \ge 1$.

We have: $x_k = a_{k1}y_1 + ... + a_{kn}y_n \ge 1$

If $raw_j(M_P)$ = ⊤ then a_{kj} = 1.

Let $\{b_1,...,b_m\} \subseteq \{a_{k1},...,a_{kn}\}$ such that $b_i \ne 0$. We have two cases:

i/ b_i = 1 ($1 \le i \le m$) and $b_1y_{j1} + ... + b_my_{jm} \ge 1$

Then there are rules: $p_k \leftarrow p_i$ in P such that $p_i = col_j(M_P)$ for $b_i = a_{kj}$ ($1 \le i \le m$) and $\{p_1,...,p_m\} \subseteq I$ implies $p_k \in T_P(I)$ where $p_k = raw_k(w)$.

ii/ b_i = 1/m ($1 \le i \le m$) and $b_1y_{j1} + ... + b_my_{jm} = 1$

Then there is a rule: $p_k \leftarrow p_1 \wedge \ldots \wedge p_m$ in P such that $p_i = col_j(M_P)$ for $b_i = a_{kj}$ $(1 \le i \le m)$ and $\{p_1,...,p_m\} \subseteq I$ implies $p_k \in T_P(I)$ where $p_k = raw_k(w)$.

Putting $J = \{raw_k(w) \mid x_k' = 1\}$, then $J = T_P(I)$

+ Suppose $J = T_P(I)$:

Define $w = (x_1',...,x_n')^T$ represents $J = T_P(I)$.

We have: $M_P \bullet v = (x_1,...,x_n)^T$ is a vector such that:

$$x_k \ge 1 \Leftrightarrow raw_k(M_P \bullet v) = \top \text{ or } raw_k(M_P \bullet v) \in T_P(I)$$

Then w such that: $x_k' = 1 \Leftrightarrow x_k \ge 1$ in $(M_P \bullet v)$

Hence $w = M_P \circ v$

Proposition 1.3: Let P be a definite program and $M_P \in \mathbb{R}^{n \times n}$ its matrix representation. Let $v \in \mathbb{R}^n$ be a vector representing an interpretation $I \subseteq B_P$.

Then I is a model of P iff $w = M_p \circ v$ and $w \le v$.

Give a matrix $M_P \in \mathbb{R}^{n \times n}$ and vector $v \in \mathbb{R}^n$. Define:

$$M_p \circ^{k+1} v = M_P \circ \left(M_p \circ^k v\right) \text{ with } M_p \circ^1 v = M_P \circ v$$

If $M_p \circ^{k+1} v = M_p \circ^k v$ for some $k \ge 1$, we write $FP\left(M_p \circ v\right) = M_p \circ^k v$

Theorem 1.1: Let P be a definite program and $M_P \in \mathbb{R}^{n \times n}$ its matrix representation. Then $m \in \mathbb{R}^n$ is a vector representing the least model of P iff $m = FP\left(M_p \circ v_0\right)$ where v_0 is the vector representing $I = \emptyset$.

Corollary: Let P be a Horn program and $M_P \in \mathbb{R}^{n \times n}$ its matrix representation. Then P is inconsistent iff a vector $m = M_p \circ^k v_0$ $(k \ge 1)$ has an element $a_i = 1$ $(1 \le i \le n)$ such that $raw_i(\mathbf{w}) = \bot$.

1.2 Disjunctive Logic Program:

a) Definition:

A *disjunctive (logic) program* is a finite set of *rules* of the form:

$$h_1 \vee ... \vee h_l \leftarrow b_1 \wedge ... \wedge b_m \quad (l, m \ge 0) \qquad (2)$$

Where h_i and b_j are propositional variables (also called atoms) in L. The rule (2) is called a *disjunctive* if $l > 1$.

In particular, the rule is fact if the body of (2) is $\top$ and it is a constraint if the head of (2) is $\bot$. The disjunctive fact is simply written as "$h_1 \vee \ldots \vee h_l \leftarrow$ ".

For each rule r of the form (2), the left-hand side of $\leftarrow$ is the *head* and the right-hand side is the *body*:

$$head(r) = \{ h_1 , \ldots , h_l \} \text{ and } body(r) = \{b_1, \ldots, b_m\}$$

A set I is an *interpretation* of P.

An interpretation $I \subseteq B_P$ is a *model* of a disjunctive program P if $body(r) \subseteq I$ implies $head(r) \cap I \neq \emptyset$ for every rule r in P and $\bot \notin I$.

A model I is a *minimal model* of P if there is no model J of P such that $J \subset I$.

Consider a disjunctive program P satisfying the MD condition: For any two rules r_1 and r_2 in P:

$$head(r_1) \cap head(r_2) \neq \emptyset \text{ implies } |body(r_1)| \leq 1 \text{ and } |body(r_2)| \leq 1$$

That is, if two different rules share the same atom in their heads, those rules contain at most one atom in their bodies.

Lemma: Every disjunctive program is converted to this class of programs by a simple program transformation as the case of Horn programs.

b) Tensor representation of disjunctive programs

Given a disjunctive program *P*, its *split program* is a Horn program obtained from *P* by replacing each disjunctive rule of the form (2) in *P* with a Horn rule:

$$h_i \leftarrow b_1 \wedge \ldots \wedge b_m \quad (1 \leq i \leq l)$$

By definition, P has multiple split programs in general. When P has k split programs, it is written as $SP_1, \ldots, SP_k$

For a set $\mathfrak{I}$ of interpretations, define the set of minimal elements as:

$$\min(\mathfrak{I}) = \{ I \mid I \in \mathfrak{I} \text{ and there is no } J \in \mathfrak{I} \text{ such that } J \subset I \}$$

Proposition 1.4: Let P be a disjunctive program and $SP_1, \ldots, SP_k$ its split programs. Also let LM be the set of least models of $SP_1, \ldots, SP_k$. Then min(LM) coincides with the set MM of minimal models of P.

Proof:

Let M be the least model of some split program SP_j.

Then for each rule r: $h_i \leftarrow b_1 \wedge \ldots \wedge b_m$ in SPj , there is a rule r': $h_1 \vee \ldots \vee h_l \leftarrow b_1 \wedge \ldots \wedge b_m$ in P such that $h_i \in head(r')$. Since M satisfies r, M satisfies r'. Thus, M is a model of P.

Then a minimal set M in min(LM) is a minimal model of P.

Hence, $\min(LM) \subseteq MM$.

Conversely, let $M \in MM$.

Then for each rule r : $h_1 \vee \ldots \vee h_l \leftarrow b_1 \wedge \ldots \wedge b_m$ in P, $\{b_1, \ldots, b_m\} \subseteq M$

implies $h_i \in M$ for some i ($1 \le i \le l$).

In this case, there is a split program SPj of P in which r is replaced by $h_i \leftarrow b_1 \wedge ... \wedge b_m$.

Then M is the least model of SPj .

Since M is a minimal among models in LM, MM $\subseteq$ min(LM).

Example 1.2: P = { p ∨ r ← s, q ∨ r ←, s ←} has the four split programs:

SP1 = { p ← s, q ←, s ←}

SP2 = { p ← s, r ←, s ←}

SP3 = { r ← s, q ←, s ←}

SP4 = { r ← s, r ←, s ←}

The set of least models of split programs is:

LM = {{ p, q, s }, { p, r, s }, { q, r, s }, { r, s }}.

Then min(LM) = {{p, q, s}, {r, s}}, which is the set of minimal models of P.

Definition 1.4 (tensor representation of disjunctive programs)

Let P be a disjunctive program that is split into $SP_1,\ldots, SP_k$ and $B_P = \{p_1,\ldots, p_n\}$. Then P is represented by a third order tensor $U_P \in \mathbb{R}^{n\times n\times k}$ as follows:

1. Each slice $U_{::h}$ $(1 \le h \le k)$ of U_P is a matrix $M_h \in \mathbb{R}^{n\times n}$ representing a split program SP_h.
2. Each matrix $M_h \in \mathbb{R}^{n\times n}$ has an element aij ($1 \le i, j \le n$) such that:
   (a) aij = 1 if pi = ⊤ or pj = ⊥.
   (b) $a_{ij_l} = \frac{1}{m}$ $(1 \le k \le m;\ 1 \le i, j_l \le n)$ if $p_i \leftarrow p_{j_1} \wedge ... \wedge p_{j_m}$ is in SP_h.
   (c) Otherwise, aij = 0.

Suppose a third-order tensor $U_P \in \mathbb{R}^{n\times n\times k}$. Given a vector $v \in \mathbb{R}^n$ representing an interpretation $I \subseteq B_P$, the (2-mode) product $U_P \bullet v$ produces a matrix $\left(a_{ij}\right) \in \mathbb{R}^{n\times k}$.

We transform $U_P \bullet v$ to a vector $W = \left(a'_{ij}\right) \in \mathbb{R}^{n\times k}$ where a'ij = 1 ($1 \le i \le n$, $1 \le j \le k$) if aij ≥ 1; otherwise, a'ij = 0.

We write $W_p = U_p \circ v$.

Example 1.3: The program P in example 2 is represented by tensor U_P:

$$U_{::1} = \begin{array}{c} \begin{matrix} p & q & r & s & \top \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array} \begin{matrix} p \\ q \\ r \\ s \\ \top \end{matrix} \qquad U_{::2} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$U_{::3} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad U_{::4} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Given the vector $\boldsymbol{v}_0 = (0, 0, 0, 0, 1)^T$ representing $I = \varnothing$ then $U_P \bullet v_0 \in \mathbb{R}^{5\times 4}$ and $W_p = U_p \circ v = U_p \bullet v$. We have W_P is a matrix as followed:

p	0	0	0	0
q	1	0	1	0
r	0	1	0	1
s	1	1	1	1
⊤	1	1	1	1

The 1st column $(0, 1, 0, 1, 1)^T$ represents the interpretation $T_{SP1}(\emptyset) = \{q, s\}$,
The 2nd column $(0, 0, 1, 1, 1)^T$ represents $T_{SP2}(\emptyset) = \{r, s\}$,
The 3rd column $(0, 1, 0, 1, 1)^T$ represents $T_{SP3}(\emptyset) = \{q, s\}$,
The 4th column $(0, 0, 1, 1, 1)^T$ represents $T_{SP4}(\emptyset) = \{r, s\}$.

Then two vectors $(1, 1, 0, 1, 1)^T$ and $(0, 0, 1, 1, 1)^T$ are minimal in M_P, which represents two minimal models {p, q, s} and {r, s} of *P*.

1.3 Normal Logic Program

A *normal (logic) program* is a finite set of *rules* of the form:

$$h \leftarrow b_1 \wedge \ldots \wedge b_m \wedge \neg b_{m+1} \wedge \ldots \wedge \neg b_n \quad (n \geq m \geq 0) \qquad (3)$$

Where h and b_i are propositional variables in L. h and bi ($1 \leq i \leq m$) are positive literals, and ¬bj ($m + 1 \leq j \leq n$) are negative literals.

A rule is a fact if the body of (3) is ⊤ and it is a constraint if the head of (3) is ⊥.
For each rule r of the form (3), we write:

head(r) = h, body+(r) ={ $b_1,\ldots, b_m$} and body−(r) = { $b_{m+1},\ldots, b_n$}

A normal program reduces to a Horn program if it contains no negative literals.

An interpretation I ⊆ BP is a *model* of a normal program P if:

body+(r) ⊆ I and body−(r) ∩ I = ∅
imply head(r) ∈ I for every rule r in P and ⊥ ∉ I.

Given a normal program *P*, an interpretation *I* is a *stable model* of P if it coincides with the least model of the Horn program:

$P^I = \{ h \leftarrow b_1 \wedge ... \wedge b_m \mid h \leftarrow b_1 \wedge ... \wedge b_m \wedge \neg b_{m+1} \wedge ... \wedge \neg b_n \in P$ and $\{b_{m+1}, \ldots, b_n\} \cap I = \emptyset\}$.

A stable model coincides with the least model if *P* is a Horn program.

A normal program may have no, one, or multiple stable models in general.

A program is *consistent* if it has at least one stable model; otherwise, the program is *inconsistent*.

Consider a disjunctive program P satisfying the MD condition: For any two rules r_1 and r_2 in P:

- head(r_1) = head(r_2) implies |body+(r_1)| ≤ 1 and |body+(r_2)| ≤ 1
- body-(r_1) ∩ body-(r_2) ≠ ∅ implies |body+(r_1)| ≤ 1 and |body+(r_2)| ≤ 1

This condition reduces to the one of Horn programs when *P* contains no negative literals.

Lemma: Every normal program is converted to this class of programs by a simple program transformation.

Example 1.4: Consider P = { p ← q ∧ r ∧ not s, p ← r ∧ t ∧ not s, q ← t, r ←, t ←}.

First, it is converted to:

P' = { p_1 ← q ∧ r ∧ not s, p_2 ← r ∧ t ∧ not s, q ← t,
r ←, t ←, p ← p1, p ← p2 }.

Next, it is converted to

P'' = { p1 ← q ∧ r ∧ not s1, p2 ← r ∧ t ∧ not s2, q ← t,
r ←, t ←, p ← p1, p ← p2, s1 ← s, s2 ← s }.

Then P'' has the single stable model M' = {p, p1, p2, q, r, t} which corresponds to the stable model M = {p, q, r, t} of P.

2. Deep learning in Logical Tensor Network

This results in this section are from [2]

+ Set of objects $O = \{o_1, o_2, \ldots\}$.

Some of the objects are associated with a set of quantitative attributes, represented by an n-tuple of real values $G(o_i) \in \mathbb{R}^n$, called *grounding*.

+ Set of relations $R = \{R_1, R_2, \ldots, R_k\}$ with $R_i \subseteq O^{\alpha(R_i)}$ with $\alpha(R_i)$ denotes the arity of relation R_i

An agent is required to:

(i) infer new knowledge about the relational structure on the objects of O;

(ii) predict the numerical properties or the class of the objects in O.

Classes and relations are not normally independent.

For example, it may be the case that if an object x is of class C, C(x), and it is related to another object y through relation R(x, y) then this other object y should be in the same class C(y). In logic: $\forall x \exists y((C(x) \wedge R(x, y)) \rightarrow C(y))$.

Whether or not C(y) holds will depend on the application: through reasoning, one may derive C(y) where otherwise there might not have been evidence of C(y) from training examples only; through learning, one may need to revise such a conclusion once examples to the contrary become available.

The vectorial representation permits both reasoning and learning as exemplified above.

2.1 Real logic

First order language $\mathcal{L}$: + Set C of constant symbols,
+ Set F of functional symbols
+ Set P of predicate symbols

Skolemised normal form: removing existential quantifiers from formal logic statements

Such as: $\forall x(A(x) \rightarrow \exists y R(x, y)) \Leftrightarrow \neg A(x) \vee R(x, f(x))$, where f is a new function that map x to y.

– G associates an n-tuple of real numbers G(t) to any closed term t of L; intuitively G(t) is the set of numeric features of the object denoted by t.

– G associates a real number in the interval [0, 1] to each clause ϕ of L. Intuitively, $G(\phi)$ represents one's confidence in the truth of ϕ; the higher the value, the higher the confidence.

Definition 2.1: Let n > 0. A grounding G for a first order language L is a function from the signature of L to the real numbers that satisfies the following conditions:

1. $G(c) \in \mathbb{R}^n$ for every constant symbol c ∈ C;
2. $G(f) \in \mathbb{R}^{n.m} \to \mathbb{R}^n$ for every f ∈ F and m is the arity of F;
3. $G(P) \in \mathbb{R}^{n.m} \to [0,1]$ for every P∈ P and m is the arity of P

A grounding G is inductively extended to all the closed terms and clauses, as follows:

$$G(f(t_1,...,t_m)) = G(f)(G(t_1),....,G(t_m))$$
$$G(P(t_1,...,t_m)) = G(P)(G(t_1),....,G(t_m))$$
$$G(\neg P(t_1,...,t_m)) = 1 - G(P)(G(t_1),....,G(t_m))$$
$$G(\phi_1 \vee ... \vee \phi_k) = \mu(G(\phi_1),...,G(\phi_k))$$

Where μ is an s-norm operator (the co-norm operator associated some t-norm operator).

The semantics of universally quantified formulas are defined w.r.t. an *aggregation operator* and a set of possible interpretations of the variables. More formally:

Definition 2.2: Let φ(**x**) be a clause of L that contains a k-tuple **x** = <x_1,…, x_k> of k distinct variables. Let T be the set of all k-tuples **t** = <t_1,…, t_k> of closed terms of L. G(φ(**x**)) = $\alpha_{t \in T}$ G(φ(**t**)), where α is an aggregation operator from $[0, 1]^{|T|} \to [0, 1]$.

<u>*Examples of aggregation operator*</u>: that can be adopted in real logic are the minimum, the arithmetic mean, the geometric and the harmonic mean. At this stage we stay general and we simply suppose that α is a generic function from $[0, 1]^{|T|}$ to [0, 1]

<u>Example 2.1:</u> Suppose that O = {o_1,o_2,o_3} is a set of documents defined on a finite dictionary D = {w_1, …, w_n} of n words.

Let $\mathcal{L}$ be the language that contains the binary function symbol concat(x, y) denoting the document resulting from the concatenation of documents x with y.

Let $\mathcal{L}$ contain also the binary predicate Sim which is supposed to be *true* if document x is deemed to be similar to document y.

As a consequence, a natural grounding of the *concat* function would be the sum of the vectors, and of the *Sim* predicate, the cosine similarity between the vectors. More formally:

- $G(o_i) = \langle n_{w_1}^{o_i},...,n_{w_n}^{o_i} \rangle$ where n_w^d is the number of occurrences of word w in document d;
- if $u,v \in \mathbb{R}^n$, $G(concat)(u,v) = u + v$;
- if $u,v \in \mathbb{R}^n$, $G(Sim)(u,v) = \frac{u.v}{\|u\|.\|v\|}$;

– if $u, v \in \mathbb{R}^n$, $G(Contains)(u, v)$ is a value in [0, 1] that provides a confidence measure that the content of a document associated with the bag of words **u** is included in the document associated with the bag of words **v**;

The grounding of the predicate *Contains* need to be "learned" from a set of examples.

For instance:

o_1 = *"John studies logic and plays football"*,
o_2 = *"Mary plays football and logic games"*,
o_3 = *"John and Mary play football and study logic together"*
W = {*John, Mary, and, football, game, logic, play, study, together*}
Then $G(o_1)$ = <1, 0, 1, 1, 0, 1, 1, 1, 0>
$G(o_2)$ = <0, 1, 1, 1, 1, 1, 1, 0, 0>
$G(o_3)$ = <1, 1, 2, 1, 0, 1, 1, 1, 1>
$G(concat(o_1,o_2)) = G(o_1) + G(o_2)$ = <1, 1, 2, 2, 1, 2, 2, 1, 0>

$$G\left(Sim\left(concat\left(o_1, o_2\right), o_3\right)\right) = \frac{G\left(concat\left(o_1, o_2\right)\right).G\left(o_3\right)}{\left\|G\left(concat\left(o_1, o_2\right)\right)\right\|.\left\|G\left(o_3\right)\right\|} \approx \frac{13}{14.83} \approx 0.88$$

$$G\left(Sim\left(o_1, o_3\right) \vee Sim\left(o_2, o_3\right)\right) = \mu_{\max}\left(G\left(Sim\left(o_1, o_3\right)\right), G\left(Sim\left(o_2, o_3\right)\right)\right) \approx \max(0.86, 0.73) = 0.86$$

$G(Contains)$ cannot be defined directly. It should be learned from a set of *examples*, as the following:

– $G(Contains(o_1, o_2))$ should be equal or close to 0
– $G(Contains(o_1, o_3))$ should be equal or close to 1
– $G(Contains(concat(o_i, o_j), o_i))$ for any o_i, o_j should be equal or close to 1
– as well as $G(Contains(o_i, o_i))$.

2.2 Learning as approximate satisfiability

Definition 2.3: (Satisfiability of Clause)

Let ϕ be a closed clause in L, G a grounding, and $v \leq w \in [0, 1]$. We say that G satisfies ϕ in the confidence interval [v, w], written $G \models_w^v \phi$, if $G(\phi) \in [v, w]$

A partial grounding, denoted by G, is a grounding that is defined on a subset of the signature of L. A grounded theory is a set of clauses in the language of L and partial grounding G.

Definition 2.4: (Grounded Theory)

A grounded theory (GT) is a pair $\langle K, G \rangle$ where K is a set of pairs $\langle [v, w], \phi(x) \rangle$, where $\phi(x)$ is a clause of L containing the set x of free variables, and $[v,w] \subseteq [0, 1]$ is an interval contained in [0, 1], and G is a partial grounding.

Definition 2.5: (Satisfiability of a Grounded Theory)

A GT $\langle K, G \rangle$ is satisfiabile if there exists a grounding G, which extends G such that for all $\langle [v, w], \phi(x) \rangle \in K$, $G \models_w^v \phi(x)$.

From the previous definition it follows that checking if a GT < K, G > is satisfiable amounts to searching for an extension of the partial grounding G in the space of *all possible groundings*, such that *all* the instantiations of the clauses in K are satisfied w.r.t. the specified interval.

Clearly a direct approach that follows the definition is unfeasible from a practical point of view. We therefore look for a form of partial satisfiability, which is computationally sustainable. There are three dimensions along which we can approximate, namely:

(i) the form of G,

(ii) the set of instantiations of the free variables **x** for each clause that need to be considered to compute the aggregation function associated to the semantics of the universal quantifier

(iii) the closeness of the truth value of the clause to the indicated interval.

For any $\langle [v, w], \phi(x) \rangle \in K$ we want to find a grounding G that minimizes the *satisfiability error*. An error occurs when a grounding G assigns a value G(ϕ) to a clause ϕ which is outside the interval [v, w] prescribed by K. The measure of this error can be defined as the minimal distance between the points in the interval [v, w] and G(ϕ):

$$Loss(G, <[v, w], \phi>) = \min_{v \le x \le w} |x - G(\phi)|$$

Notice that if G(ϕ)∈[v,w], Loss(G, ϕ) = 0.

For open clauses we have also to identify a subset of terms in which the variables need to be interpreted. We therefore define:

$$Loss(G, <[v, w], \phi>, T_{\phi(x)}) = \min_{v \le x \le w} |x - \alpha_{t \in T_{\phi(x)}} G(\phi(t))|$$

Where $T_{\varphi(\mathbf{x})}$ is a finite set of *k*-tuples of grounded terms of *L*, with $|\mathbf{x}| = k$, and α is the aggregation function used for interpreting the universally quantified variables.

The above gives rise to the following definition of approximate satisfiability w.r.t. a family 𝔾 grounding functions on the language L.

Definition 2.6: (Best satisfiability problem).

Let < K, G > be a grounded theory

Let 𝔾 be a family of grounding functions. For every $\langle [v, w], \phi(x) \rangle \in K$, let $T_{\varphi(\mathbf{x})}$ be a subset of all k-tuples of ground terms of L. We define the best satisfiability problem as the

problem of finding an extensions G* of G in G that minimizes the satisfiability error on the set clause of K interpreted w.r.t the relative domain:

$$G^* = \arg\min_{G \subseteq G \in \mathbb{G}} \sum_{\langle [v,w], \phi(x) \rangle \in K} Loss\left(G, \langle [v,w], \phi(x) \rangle, T_{\phi(x)}\right)$$

2.3 Implementing Real Logic in Tensor Networks:

$\mathbb{G}$ is the space of real tensor transformations of order k (where k is a parameter).

If f is a function symbol of arity m and $v_1, ..., v_m \in \mathbb{R}^n$ are real vectors corresponding to the grounding of m terms then $G(f)(v_1, ..., v_m)$ can be written as:

$$G(f)(v_1, ..., v_m) = M_f v + N_f$$

for n × mn matrix M_f and n-vector N_f where v = <v_1,…,v_m>

The grounding of m-ary predicate P, G(P), is defined as a generalization of the neural tensor network (which has been shown effective at knowledge compilation in the presence of simple logical constraints), as a function from $\mathbb{R}^{mn}$ to [0,1], as follows:

$$G(P) = \sigma\left(u_P^T \tanh\left(v^T W_P^{[1:k]} v + V_P v + B_p\right)\right)$$

Where $W_P^{[1:k]}$ is a 3-tensor in $\mathbb{R}^{mn \times mn \times k}$, V_P is a matrix in $\mathbb{R}^{k \times mn}$ and B_P is a vector in $\mathbb{R}^k$, and σ is the sigmoid function.

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \qquad x \in \mathbb{R}, \quad \tanh(u) = <\tanh(u_1), ... \tanh(u_k)> \qquad u = <u_1, ..., u_k> \in \mathbb{R}^k$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad x \in \mathbb{R}$$

With this encoding, the grounding of a clause can be determined by a neural network which first computes the grounding of the literals contained in the clause, and then combines them using the specific s-norm.

Example 2.2: The tensor network for $\neg P(x, y) \rightarrow A(y)$ as the figure:

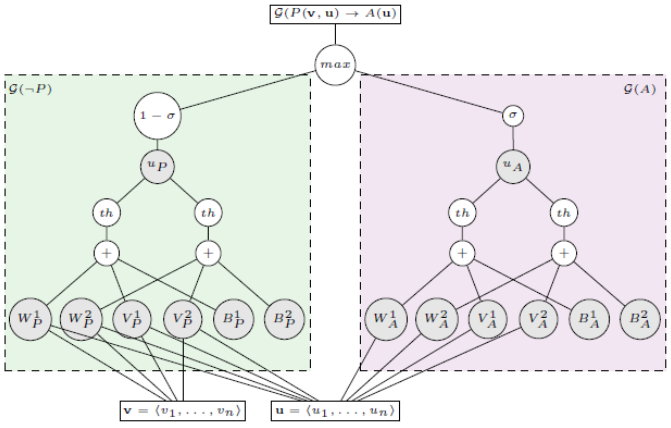

Fig. 1: Tensor net for $\neg P(x, y) \rightarrow A(y)$ with G(x) = v, G(y) = u and k = 2.

Example 2.3: Use the *friends and smokers* example, such as:

Table 1 Example of a first-order knowledge base and MLN. Fr() is short for Friends(), Sm() for Smokes(), and Ca() for Cancer()

English	First-order logic	Clausal form	Weight
Friends of friends are friends	$\forall x \forall y \forall z\ \mathrm{Fr}(x,y) \wedge \mathrm{Fr}(y,z) \Rightarrow \mathrm{Fr}(x,z)$	$\neg\mathrm{Fr}(x,y) \vee \neg\mathrm{Fr}(y,z) \vee \mathrm{Fr}(x,z)$	0.7
Friendless people smoke	$\forall x\ (\neg(\exists y\ \mathrm{Fr}(x,y)) \Rightarrow \mathrm{Sm}(x))$	$\mathrm{Fr}(x,g(x)) \vee \mathrm{Sm}(x)$	2.3
Smoking causes cancer	$\forall x\ \mathrm{Sm}(x) \Rightarrow \mathrm{Ca}(x)$	$\neg\mathrm{Sm}(x) \vee \mathrm{Ca}(x)$	1.5
If two people are friends, either both smoke or neither does	$\forall x \forall y\ \mathrm{Fr}(x,y) \Rightarrow (\mathrm{Sm}(x) \Leftrightarrow \mathrm{Sm}(y))$	$\neg\mathrm{Fr}(x,y) \vee \mathrm{Sm}(x) \vee \neg\mathrm{Sm}(y)$, $\neg\mathrm{Fr}(x,y) \vee \neg\mathrm{Sm}(x) \vee \mathrm{Sm}(y)$	1.1 1.1

There are 14 people divided into two groups {a,b,…,h} and {i,j,…,n}. Within each group of people we have complete knowledge of their smoking habits. In the first group, we have complete knowledge of who has and does not have cancer. In the second group, this is not known for any of the persons. Knowledge about the friendship relation is complete within each group only if symmetry of friendship is assumed.

Otherwise, it is incomplete in that it may be known that, e.g., a is a friend of b, but not known whether b is a friend of a. Finally, there is also general knowledge about smoking, friendship and cancer, namely, that smoking causes cancer, friendship is normally a symmetric and anti-reflexive relation, everyone has a friend, and that smoking propagates (either actively or passively) among friends.

$\mathcal{K}^{SFC}_{a\ldots h}$

$S(a),\ S(e),\ S(f),\ S(g),$
$\neg S(b),\ \neg S(c),\ \neg S(d),\ \neg S(g),\ \neg S(h),$
$F(a,b),\ F(a,e),\ F(a,f),\ F(a,g),\ F(b,c),$
$F(c,d),\ F(e,f),\ F(g,h),$
$\neg F(a,c),\ \neg F(a,d),\ \neg F(a,h),\ \neg F(b,d),\ \neg F(b,e),$
$\neg F(b,f),\ \neg F(b,g),\ \neg F(b,h),\ \neg F(c,e),\ \neg F(c,f),$
$\neg F(c,g),\ \neg F(c,h),\ \neg F(d,e),\ \neg F(d,f),\ \neg F(d,g),$
$\neg F(d,h),\ \neg F(e,g),\ \neg F(e,h),\ \neg F(f,g),\ \neg F(f,h),$
$C(a),\ C(e),$
$\neg C(b),\ \neg C(c),\ \neg C(d),\ \neg C(f),\ \neg C(g),\ \neg C(h)$

$\mathcal{K}^{SF}_{i\ldots n}$

$S(i),\ S(n),$
$\neg S(j),\ \neg S(k),$
$\neg S(l),\ \neg S(m),$
$F(i,j),\ F(i,m),$
$F(k,l),\ F(m,n),$
$\neg F(i,k),\ \neg F(i,l),$
$\neg F(i,n),\ \neg F(j,k),$
$\neg F(j,l),\ \neg F(j,m),$
$\neg F(j,n),\ \neg F(l,n),$
$\neg F(k,m),\ \neg F(l,m)$

$\mathcal{K}^{SFC}$

$\forall x \neg F(x,x),\quad \forall xy(S(x) \wedge F(x,y) \rightarrow S(y)),$
$\forall xy(F(x,y) \rightarrow F(y,x)),\quad \forall x(S(x) \rightarrow C(x))$
$\forall x \exists y F(x,y),$

Fig. 1

If we adopt classical FOL semantics the knowledge base $K^{SFC} = K^{SFC}_{x,y} \cup K^{SFC}_{a,\ldots,h} \cup K^{SFC}_{i,\ldots,n}$ is inconsistent. Indeed the axiom $\forall x\big(S(x) \rightarrow C(x)\big)$ contained in $K^{SFC}_{x,y}$, the facts $S(f)$ and $\neg C(f)$ contained in $K^{SFC}_{a,\ldots,h}$

If instead we admit the possibility for facts in K^{SFC} to be true "to a certain extent", i.e., they are associated to an (unknown) degree of truth smaller than 1, then K^{SFC} is consistent and admits a grounding. Our main tasks are:

(i) Find the maximum degree of truth of the facts contained so that K^{SFC} is consistent
(ii) Find a truth-value for all the ground atomic facts not explicitly mentioned in K^{SFC}
(iii) Find the grounding of each constant symbol a, ..., n.

To answer (i)-(iii), we use LTN to find a grounding that best approximates the complete KB. To show the role of background knowledge in the learning-inference process, we run two experiments:

In the first (exp1), we seek to complete a KB consisting of only factual knowledge: $K^{SFC}_{\text{exp1}} = K^{SFC}_{a...h} \cup K^{SFC}_{i...n}$.

In the second (exp2), we also include background knowledge, that is: $K^{SFC}_{\text{exp2}} = K^{SFC}_{\text{exp1}} \cup K^{SFC}_{x,y}$.

We configure the network as follows: each constant (i.e. person) can have up to 30 real-valued features. We set the number of layers k in the tensor network to 10, and the regularization parameter $\lambda = 10^{-10}$. For the purpose of illustration, we use the Lukasiewicz t-norm with s-norm μ(a, b) = min(1, a + b), and use the harmonic mean as aggregation operator. An estimation of the optimal grounding is obtained after 5,000 runs of the RMSProp learning algorithm available in TENSORFLOW.

The results of the two experiments are reported in Table 2.

	S	C	F							
			a	b	c	d	e	f	g	h
a	**1.00**	**1.00**	0.00	**1.00**	0.00	0.00	**1.00**	**1.00**	**1.00**	0.00
b	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00
c	0.00	0.00	0.00	**0.82**	0.00	**1.00**	0.00	0.00	0.00	0.00
d	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00
e	**1.00**	**1.00**	0.00	0.33	0.21	0.00	0.00	**1.00**	0.00	0.00
f	**1.00**	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00
g	**1.00**	0.00	0.03	**1.00**	**1.00**	**1.00**	0.11	**1.00**	0.00	**1.00**
h	0.00	0.00	0.00	0.23	0.01	0.14	0.00	0.02	0.00	0.00

	S	C	F					
			i	j	k	l	m	n
i	**1.00**	0.00	0.00	**1.00**	0.00	0.00	**1.00**	0.00
j	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
k	0.00	0.00	0.10	**1.00**	0.00	**1.00**	0.00	0.00
l	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00
m	0.00	0.03	**1.00**	**1.00**	0.12	**1.00**	0.00	**1.00**
n	**1.00**	0.01	0.00	0.98	0.00	0.01	0.02	0.00

Learning and reasoning on $\mathcal{K}_{exp1} = \mathcal{K}^{SFC}_{a...h} \cup \mathcal{K}^{SF}_{i...n}$

	S	C	F							
			a	b	c	d	e	f	g	h
a	**0.84**	**0.87**	0.02	**0.95**	0.01	0.03	**0.93**	**0.97**	**0.98**	0.01
b	0.13	0.16	0.45	0.01	**0.97**	0.04	0.02	0.03	0.06	0.03
c	0.13	0.15	0.02	**0.94**	0.11	**0.99**	0.03	0.16	0.15	0.15
d	0.14	0.15	0.01	0.06	**0.88**	0.08	0.01	0.03	0.07	0.02
e	**0.84**	**0.85**	0.32	0.06	0.05	0.03	0.04	**0.97**	0.07	0.06
f	**0.81**	0.19	0.34	0.11	0.08	0.04	0.42	0.08	0.06	0.05
g	**0.82**	0.19	**0.81**	0.26	0.19	0.30	0.06	0.28	0.00	**0.94**
h	0.14	0.17	0.05	0.25	0.26	0.16	0.20	0.14	**0.72**	0.01

	S	C	F							
			i	j	k	l	m	n		
i	**0.83**	**0.86**	0.02	**0.91**	0.01	0.03	**0.97**	0.01		
j	0.19	0.22	**0.73**	0.03	0.00	0.04	0.02	0.05		
k	0.14	0.34	0.17	0.07	0.04	**0.97**	0.04	0.02		
l	0.16	0.19	0.11	0.12	0.15	0.06	0.05	0.03		
m	0.14	0.17	**0.96**	0.07	0.02	0.11	0.00	**0.92**		
n	**0.84**	**0.86**	0.13	0.28	0.01	0.24	**0.69**	0.02		

	$a, \ldots, h, i, \ldots, n$	
$\forall x \neg F(x,x)$	**0.98**	
$\forall xy(F(x,y) \rightarrow F(y,x))$	**0.90**	**0.90**
$\forall x(S(x) \rightarrow C(x))$	**0.77**	
$\forall x(S(x) \wedge F(x,y) \rightarrow S(y))$	**0.96**	**0.92**
$\forall x \exists y(F(x,y))$	**1.0**	

Learning and reasoning on $\mathcal{K}_{exp2} = \mathcal{K}^{SFC}_{a...h} \cup \mathcal{K}^{SF}_{i...n} \cup \mathcal{K}^{SFC}$

Table 2

For readability, we use boldface for truth-values greater than 0.5. The truth-values of the facts listed in a knowledge-base are highlighted with the same background color of the knowledgebase in Figure 1. The values with white background are the result of the knowledge completion produced by the LTN learning-inference procedure.

To evaluate the quality of the results, one has to check whether:

(i) the truth-values of the facts listed in a KB are indeed close to 1.0, and

(ii) the truth-values associated with knowledge completion correspond to expectation.

An initial analysis shows that the LTN associated with K^{SFC}_{exp1} produces the same facts as K^{SFC}_{exp1} itself. In other words, the LTN fits the data.

However, the LTN also learns to infer additional positive and negative facts about F and C not derivable from K^{SFC}_{exp1} by pure logical reasoning; for example: F(c, b), F(g, b) and ¬F(b, a). These facts are derived by exploiting similarities between the groundings of the constants generated by the LTN. For instance, G(c) and G(g) happen to present a high cosine similarity measure. As a result, facts about the friendship relations of c affect the friendship relations of g and vice-versa, for instance F(c, b) and F(g, b). The level of satisfiability associated with $K^{SFC}_{\text{exp1}} \approx 1$, which indicates that K^{SFC}_{exp1} is classically satisfiable.

The results of the second experiment show that more facts can be learned with the inclusion of background knowledge. For example, the LTN now predicts that C(i) and C(n) are true. Similarly, from the symmetry of the friendship relation, the LTN concludes that m is a friend of i, as expected.

In fact, all the axioms in the generic background knowledge K^{SFC} are satisfied with a degree of satisfiability higher than 90%, apart from the *smoking causes cancer* axiom - which is responsible for the classical inconsistency since in the data f and g smoke and do not have cancer -, which has a degree of satisfiability of 77%.

References

[1] *Linear Algebraic Characterization of Logic Programs*, ID: 1978.

[2] Luciano Serafini, Artur S. d'Avila Garcez, *Learning and Reasoning with Logic Tensor Network*, AI*IA 2016, LNAI 10037, pp. 334–348, 2016.