Introduction to Logic Tensor Network (LTN)

Luciano Serafini   Michael Spranger

FBK-IRST, Trento, Italy
Sony Computer Science Laboratories Inc., Tokyo, Japan

July 15, 2018

Motivation

- Modelling imprecise concepts: e.g.,

  old man, strong wind, enough food

- Reasoning about imprecise concepts to support decision making: e.g.:

  IF temperature IS very cold THEN stop fan
  IF temperature IS cold THEN turn down fan
  IF temperature IS normal THEN maintain level
  IF temperature IS reasonably hot THEN speed up fan

Fuzzy Logic

Fuzzy logic in the broad sense

serves mainly as apparatus for fuzzy control, analysis of vagueness in natural language and several other application domains. It is one of the techniques of soft-computing, i.e. computational methods tolerant to suboptimality and impreciseness (vagueness) and giving quick, simple and sufficiently good solutions.

Fuzzy logic in the narrow sense

is symbolic logic with a comparative notion of truth, syntax, semantics, axiomatization, truth-preserving deduction, completeness, etc. It is a branch of many-valued logic

Fuzzy logic in LTN

LTN use fuzzy logic in the second definition. Reference Text:

> *Petr. Hájek. Metamathematics of Fuzzy Logic, volume 4 of Trends in Logic- Studia Logica Library. Dordrecht/Boston/London, 1998.*

Fuzzy sets and crisp sets

- In classical mathematics one deals with collections of objects called sets.
- it is convenient to fix some universe U in which every set is assumed to be included.
- It is also useful to think of a set A as a function from U which takes value 1 on objects which belong to A and 0 on all the rest.
- Such functions is called the characteristic function of A, $\chi_A(\cdot)$:

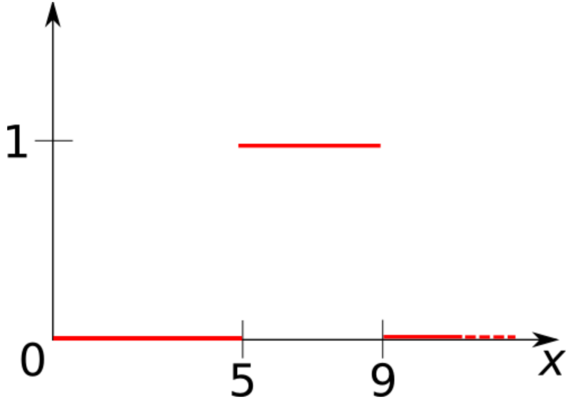
$$\chi_A(x) =_{def} \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

- So there exists a bijective correspondence between characteristic functions and sets

Crisp sets

Example

Let U be the set of all real numbers between 0 and 10 and let $A = [5, 9]$ be the subset of X of real numbers between 5 and 9. This results in the following figure:



Fuzzy sets

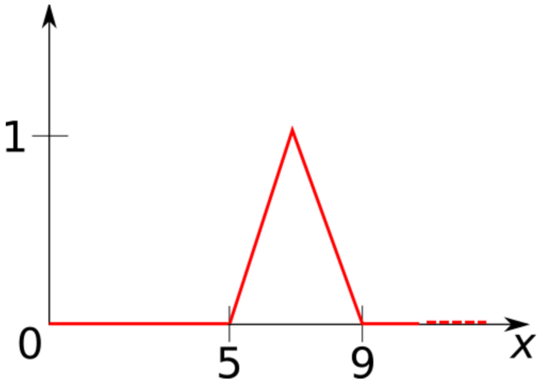
- Fuzzy sets generalise this definition, allowing elements to belong to a given set with a certain degree.
- Instead of considering characteristic functions with value in $\{0, 1\}$ we consider now functions valued in the interval $[0, 1]$.
- A fuzzy subset F of a set U is a function $\mu_F(\cdot)$ assigning to every element $x \in U$ the degree of membership of x to F:

$$\mu_F : U \to [0, 1]$$

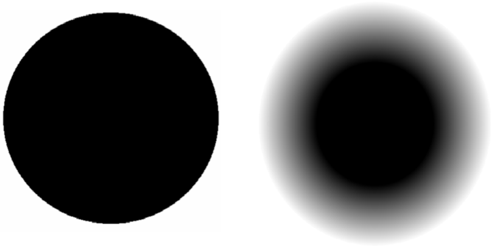
Fuzzy set

Example (Cont.d)

Let, as above, U be the set of real numbers between 1 and 10. A description of the fuzzy set of real numbers close to 7 could be given by the following figure:



Crisp and fuzzy sets



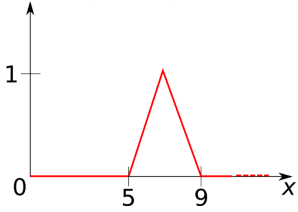
Operations between sets

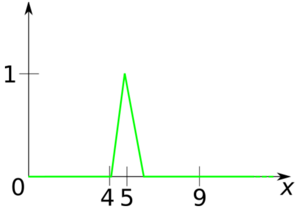
- In classical set theory there are some basic operations defined over sets.
- Let U beasetand 2^U be the set of all subsets of U, or, equivalently, the set of all functions in $U \to \{0, 1\}$.
- The operation of union, intersection and complement are defined in the following ways:

$$
\begin{aligned}
A \cup B &= \{x \mid x \in A \textit{ or } x \in B\} & \chi_{A\cup B}(x) &= max(\chi_A(x), \chi_B(x)) \\
A \cap B &= \{x \mid x \in A \textit{ and } x \in B\} & \chi_{A\cap B}(x) &= min(\chi_A(x), \chi_B(x)) \\
\bar{A} &= \{x \in U \mid x \notin A\} & \chi_{\bar{A}}(x) &= 1 - \chi_A(x)
\end{aligned}
$$

Operations between fuzzy sets

The law $\chi_{A \cup B}(x) = \max(\chi_A(x), \chi_B(x))$ gives us an obvious way to generalise union to fuzzy sets.
Let F and S be fuzzy subsets of U given by membership functions μ_F and μ_S:

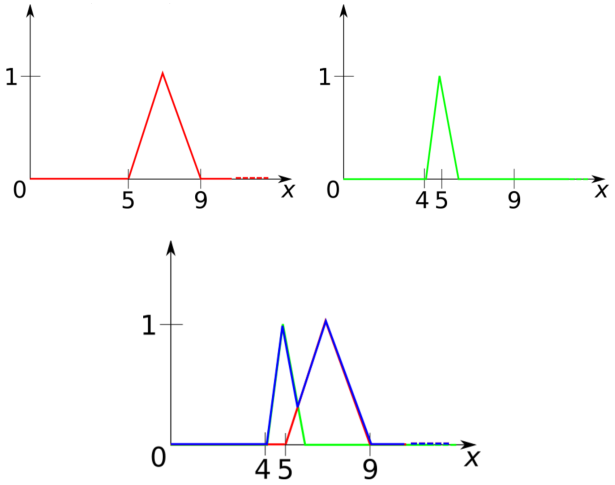




Operations between fuzzy sets: Union

We set

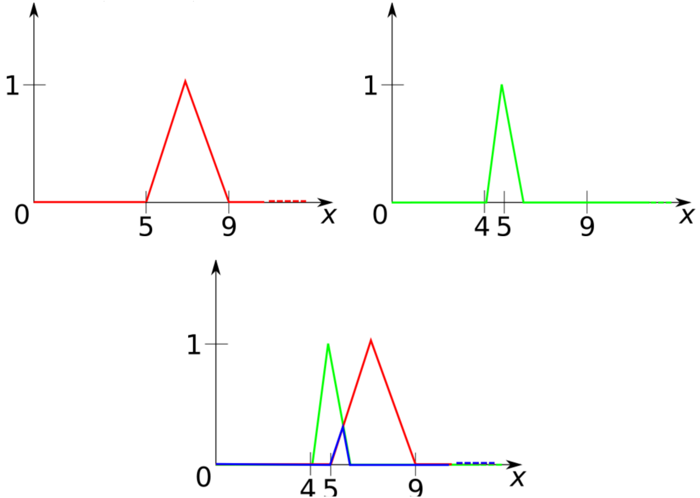
$$\mu_{F \cup S}(x) = \max(\mu_F(x), \mu_S(x))$$



Operations between fuzzy sets: Intersection

Analogously, we set

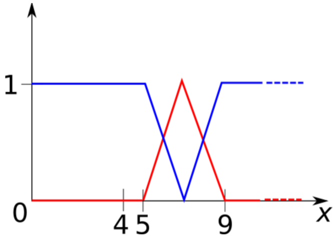
$$\mu_{F \cap S}(x) = \min(\mu_F(x), \mu_S(x))$$



Operations between fuzzy sets: complement

Finally the complement for characteristic functions is defined by,

$$\mu_{\bar{F}}(x) = 1 - \mu_F(x).$$



Fuzzy propositional logic

- We consider propositional languages. I.e., languages that contain only propositions, i.e., statements that can be assigned with some truth value

Fuzzy propositional logic

- We consider propositional languages. I.e., languages that contain only propositions, i.e., statements that can be assigned with some truth value
- Truth values is a subset of the real interval [0,1]

Fuzzy propositional logic

- We consider propositional languages. I.e., languages that contain only propositions, i.e., statements that can be assigned with some truth value
- Truth values is a subset of the real interval [0,1]
- connectives have functional semantics. e.g., a binary connective $\circ$ must be interpreted in a function $f_\circ : [0,1]^2 \rightarrow [0,1]$.

Fuzzy propositional logic

- We consider propositional languages. I.e., languages that contain only propositions, i.e., statements that can be assigned with some truth value
- Truth values is a subset of the real interval [0,1]
- connectives have functional semantics. e.g., a binary connective $\circ$ must be interpreted in a function $f_\circ : [0, 1]^2 \to [0, 1]$.
- Truth values are ordered, i.e., if $x > y$, then x is a stronger truth than y
- Generalization of classical propositional logic:

  0 corresponds to FALSE and

  1 corresponds to TRUE

Fuzzy semantics for propositional connectives

The logical symbols of propositional logics are propositional connectives.
Propositional fuzzy logics have to provide a new semantics for these sysmbols in order to deal with more than two truth values.
We will consider the semantics for $\wedge$, $\rightarrow$, and $\neg$.

Conjunction - Triangular-norm (T-norm)

Conjunction in fuzzy logic

The conjunction connective in fuzzy logic is formalized by a binary operation on truth values, called t-norm, which satisfy a minimal set of properties to capture the intuitive meaning of conjunction.

Definition (t-norm)

A t-norm is a binary operation $\otimes : [0,1]^2 \to [0,1]$ satisfying the following conditions:

Commutative $x \otimes y = y \otimes x$

Associative $x \otimes (y \otimes z) = (x \otimes y) \otimes z$

Non-decreasing $x \leq y \quad \to z \otimes x \leq z \otimes y$

Zero and One $0 \otimes x = 0$ and $1 \otimes x = x$

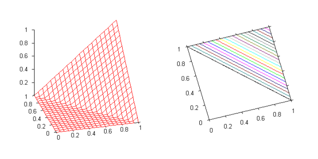
A t-norm $\otimes$ is continuous if the function $\otimes : [0,1]^2 \to [0,1]$ is a continuous function in the usual sense.

T-norm

Example (t-norms)

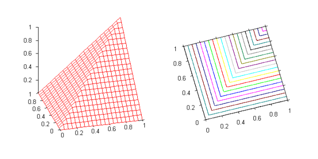
Łukasiewicz t-norm

$x \otimes y = max(0, x + y - 1)$



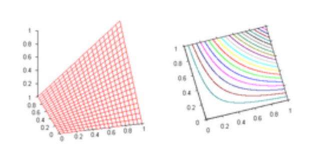
Gödel t-norm

$x \otimes y = min(x, y)$



Product t-norm

$x \otimes y = x \cdot y$



disjunction - t-conorml

Disjunction in fuzzy logic

- Given a t-norm μ, its corresponding t-conorm is defined as

$$x \oplus y = 1 - (1 - x) \otimes (1 - y)$$

- t-conorms are used to provide the semantics of $\vee$, due to their duality w.r.t., t-norm, and the following (properties

Commutative $x \oplus y = y \oplus x$

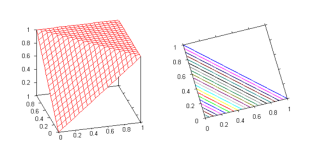
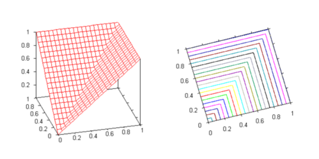
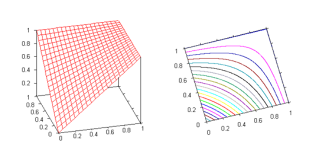
Associative $x \oplus (y \oplus z) = (x \oplus y) \oplus z$

Non-decreasing $x \leq y \quad \rightarrow z \oplus x \leq z \oplus y$

Zero and One $0 \oplus x = x$ and $1 \oplus x = 1$

T-conorms

Example (t-conorms)

Łukasiewicz t-conorm $x \oplus y = \min(x + y, 1)$	
Gödel t-conorm $x \otimes y = \max(x, y)$	
Product t-conorm $x \otimes y = x \cdot y$	

Implication - Residual

Implication in fuzzy logic

- Intuitively the more $\phi \to \psi$ is true, the less additional information is carried by ψ w.r.t., ϕ.
- If x and y are the truth value of ϕ and ψ, let $s \Rightarrow y$ the truth value of $\phi \to \psi$. Then the following property should hold for all $z \in [0, 1]$

$$x * z \leq y \quad \text{iff} \quad z \leq (x \Rightarrow y)$$

- We therefore define the semantics of implication as the maximum truth value to be "added" to x in order to obtain y.

Definition (Residual of a t-norm)

The residual of a t-norm $*$, is a function $\Rightarrow: [0, 1]^2 \to [0, 1]$:

$$x \Rightarrow y \quad = \quad max(\{z \mid x * z \leq y\})$$

Implication - Residual

Example

Residual If $x \leq y$, then $x \Rightarrow y = 1$, if $x > y$ then there are many possible definitions of $x \Rightarrow y$:

- Łukasiewicz residual: $x \Rightarrow y = 1 - x + y$
- Gödel residual: $x \Rightarrow y = y$
- Product residual: $x \Rightarrow y = y/x$ (notice that $x > 0$)

Properties of Residual

1. If $x \leq y$ then $x \Rightarrow y = 1$
2. $(1 \Rightarrow x) = x$
3. $(x \Rightarrow 1) = 1$
4. If $x \leq y$ then $x = y * (y \Rightarrow x)$

Negation - Precomplement

Negation as implication of false

In classical propositional logic $\neg\phi$ can be defined as $\phi \rightarrow \bot$. We can proceed similarly in Fuzzy logic

Definition (Precomplement)

For every residual operator $\Rightarrow$ (and therefore for every t-norm) the precomplement operator denoted by $(-)$, is defined as:

$$(-)x = x \Rightarrow 0$$

Negation - Precomplement

Example (Precomplement)

- Łukasiewicz precomplement: $(-x) = 1 - x$
- Gödel precomplement: $(-x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{Otherwise} \end{cases}$
- Product precomplement: $(-x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{Otherwise} \end{cases}$

Logic Tensor Networks

Logic Tensor Networks

logic tensor networks (LTN) [8] is a framework where the elements of a first order signature are grounded to Neural Networks.

Inspiration

It has been inspired by the work

- Bridging logic and kernel machines by M. Diligenti, M. Gori, M. Maggini, and L. Rigutini (Uni Siena - Italy) [3]
- Reasoning With Neural Tensor Networks by R. Socher, D. Chen, C.D. Manning, and A. Y. Ng. [9]
- Neuro-Symbolic Integration initiative by Artur d'Avila Garces, et. al.

Logic Tensor Networks

Logictensornetwork source

- https://github.com/logictensornetworks/logictensornetworks/
- git clone https://github.com/logictensornetworks/logictensornetworks.git

Tutorial Python Notebook

- https://github.com/logictensornetworks/tutorials/
- git clone https://github.com/logictensornetworks/tutorials.git

The language of LTN

The agent uses First Order Logic (FOL) to represent its knowledge

- FOL signature: constant, function, predicate symbols
- logical symbols: $\wedge, \vee, \rightarrow, \forall, \exists$
- logical symbols are interpreted (grounded) in fuzzy semantic
- what about non logical symbols? In formal logic they are interpreted in abstract algebraic structures, that represent the states-of-affairs of the real world
- in LTN non logical symbols are interpreted in real numbers (i.e, the output of the agent sensors in the world)

FOL Signature

Definition (FOL signature and language)

A *FOL signature* contains:

- a set of constant symbols $c_1, c_2, \dots,$
- a set of function symbols $f_1, f_2, \dots,$ (each f_i has an arity, i.e., the number of arguments it takes)
- a set of relation/predicate symbols $P_1, P_2, \dots,$ (each P_i has an arity, i.e., the number of arguments it takes)

Example (FOL signature)

- CristianoRonaldo, Sergio Ramos, Barca, Juventus, 1, 2, ... are constants and denotes objects of the domain
- number(_) is a unary function symbol
- FootballPlayer(_),Male(_) are unary predicates (class)
- PlaysFor(_,_) is a binary predicate (relation)

FOL Formulas

FOL terms and formulas are inductively defined as usual

Example (Formulas)

$$FootballPlayer(CristianoRonaldo)$$
$$CristianoRonaldo = CR7$$
$$numberOf(CR7) = 7$$
$$PlaysFor(CR7, Barca) \vee PlaysFor(CR7, Juventus)$$
$$\forall x \forall y : PlaysFor(x, y) \rightarrow FootballPlayer(x) \wedge FootballTeam(y)$$
$$\forall x \exists y : FootballPlayer(x) \rightarrow PlaysFor(x, y) \wedge FootballTeam(y)$$
$$\forall y (\forall x : PlaysFor(y, x) \rightarrow Male(x)) \vee (\forall x : PlaysFor(y, x) \rightarrow Female(x))$$

Symbol Grounding (SG)

- SG was originally introduced by Searle [7] and Newell [6],
- SG is a key concept that has beed largely discussed by the AI community [4, 5, 1, 2].

Definition

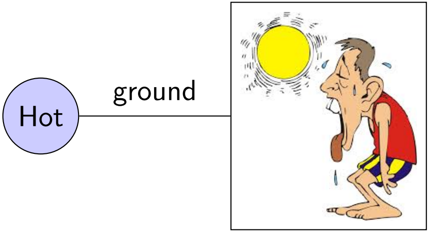
SG is the process of formation and manipulation of correspondences between symbolic tokens used by an agent, and perceptions and actions in the agent's physical environment.

Symbol Grounding (SG)

- SG was originally introduced by Searle [7] and Newell [6],
- SG is a key concept that has beed largely discussed by the AI community [4, 5, 1, 2].

Definition

SG is the process of formation and manipulation of correspondences between symbolic tokens used by an agent, and perceptions and actions in the agent's physical environment.

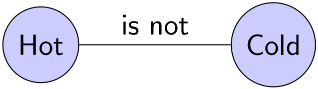


Symbol Grounding (SG)

- SG was originally introduced by Searle [7] and Newell [6],
- SG is a key concept that has beed largely discussed by the AI community [4, 5, 1, 2].

Definition

SG is the process of formation and manipulation of correspondences between symbolic tokens used by an agent, and perceptions and actions in the agent's physical environment.

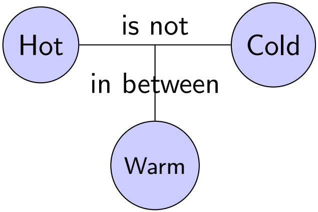


Symbol Grounding (SG)

- SG was originally introduced by Searle [7] and Newell [6],
- SG is a key concept that has beed largely discussed by the AI community [4, 5, 1, 2].

Definition

SG is the process of formation and manipulation of correspondences between symbolic tokens used by an agent, and perceptions and actions in the agent's physical environment.

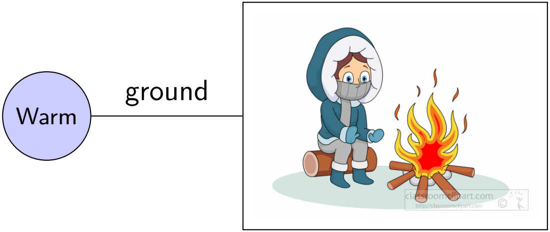


Symbol Grounding (SG)

- SG was originally introduced by Searle [7] and Newell [6],
- SG is a key concept that has beed largely discussed by the AI community [4, 5, 1, 2].

Definition

SG is the process of formation and manipulation of correspondences between symbolic tokens used by an agent, and perceptions and actions in the agent's physical environment.



Symbol grounding in agents

- Suppose that an agent uses a language based on a set of symbols (its *signature*) to represent its knowledge about the world:
- Suppose also that a this agent perceives the world via a set of sensors
- Assume that a sensor measures the current state of the world and returns a real number
- Then, if an agent has k sensors then it perceives the current state of the worlds as a vector in $\mathbb{R}^k$
- Suppose also that symbols are used to express knowledge about the current state of the world (i.e., no dynamic)
- then, agent signature should be grounded in some structure in $\mathbb{R}^k$.

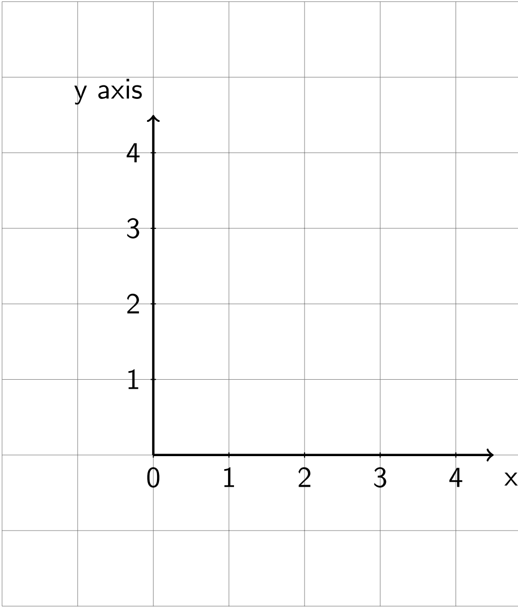
Grounding FOL Signature in the Real Field

Definition (Grounding of FOL signature)

A *grounding* $\mathcal{G}$ of a first order language $\mathcal{L}$ in a k-ary real vector space is a function $\cdot^{\mathcal{G}}$:

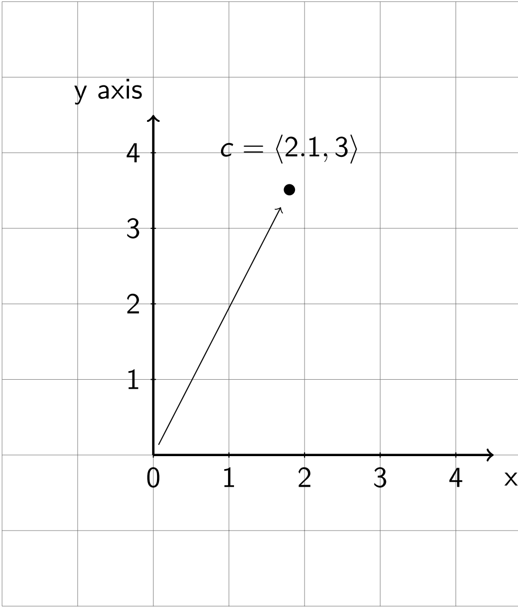
- $c^{\mathcal{G}} \in \mathbb{R}^k$: each constant is mapped in a point in $\mathbb{R}^k$
- $f^{\mathcal{G}} \in \mathbb{R}^{k \cdot n} \to R^k$; each n-ary function symbol is mapped in an n-ary real function;
- $P^{\mathcal{G}} : \mathbb{R}^{k*n} \to [0,1]$: each n-ary relational symbol is mapped in a fuzzy subset of $\mathbb{R}^{k \cdot n}$.

Grounding FOL in $\mathbb{R}^2$



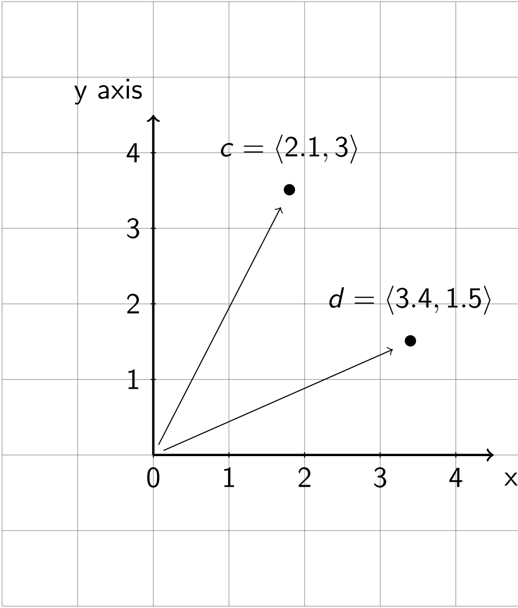
Grounding FOL in $\mathbb{R}^2$

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$



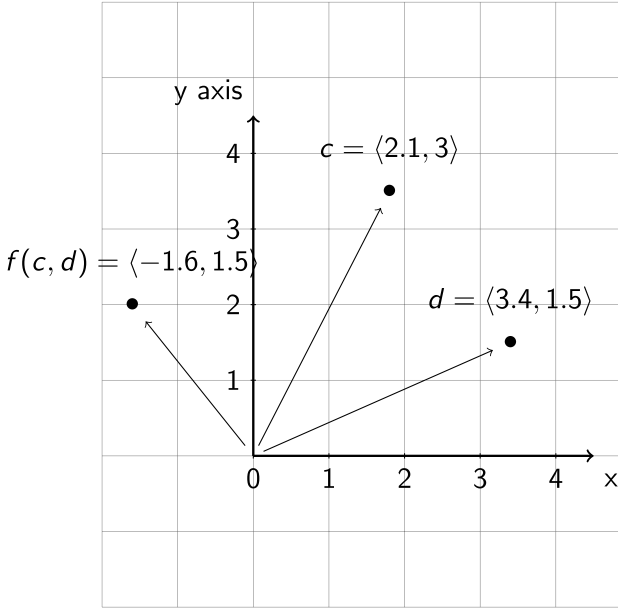
Grounding FOL in $\mathbb{R}^2$

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \langle 3.4, 1.5 \rangle$



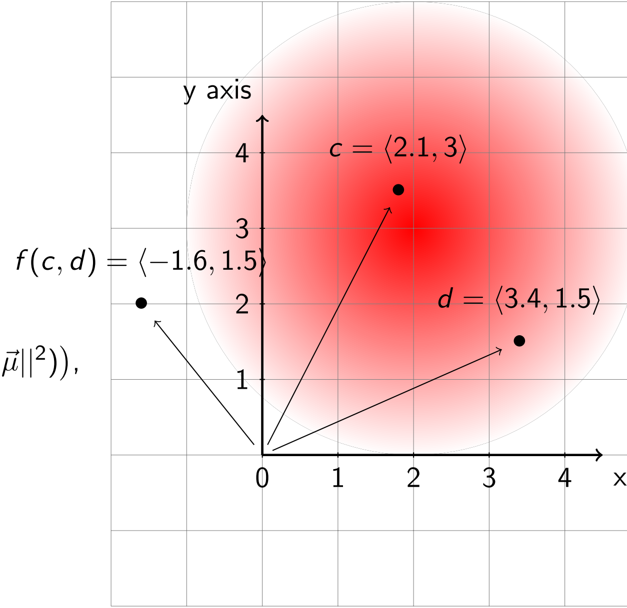
Grounding FOL in $\mathbb{R}^2$

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \langle 3.4, 1.5 \rangle$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \vec{x} - \vec{y}$



Grounding FOL in $\mathbb{R}^2$

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \langle 3.4, 1.5 \rangle$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \vec{x} - \vec{y}$
- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((-||\vec{x} - \vec{\mu}||^2)\right)$, with $\mu = (2, 3)$



LTN coding

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$

Example (LTN code)

```
ltnw.constant("c",[2.1,3])
```

LTN coding

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \langle 3.4, 1.5 \rangle$

Example (LTN code)

```
ltnw.constant("c",[2.1,3])
ltnw.constant("d",[3.4,1.5])
```

LTN coding

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \langle 3.4, 1.5 \rangle$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \vec{x} - \vec{y}$

Example (LTN code)

```
ltnw.constant("c",[2.1,3])
ltnw.constant("d",[3.4,1.5])
ltnw.function("f",4,2,fun_definition=lambda x,y:x-y)
```

LTN coding

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \langle 3.4, 1.5 \rangle$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \vec{x} - \vec{y}$
- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((-||\vec{x} - \vec{\mu}||^2)\right)$, with $\mu = (2, 3)$

Example (LTN code)

```
ltnw.constant("c",[2.1,3])
ltnw.constant("d",[3.4,1.5])
ltnw.function("f",4,2,fun_definition=lambda x,y:x-y)
mu = tf.constant([2.,3.])
ltnw.predicate("P",2,pred_definition=lambda x:
         tf.exp(-tf.reduce_sum(tf.square(x-mu))))
```

Grounding FOL propositional formulas

Definition (Grounding of formulas)

The grounding of formulas is recursively defined according to their structure, and the fuzzy semantics of connectives.

- $P(t_1, \ldots, t_n)^{\mathcal{G}} = P^{\mathcal{G}}(t_1^{\mathcal{G}}, \ldots, t_n^{\mathcal{G}})$

Grounding FOL propositional formulas

Definition (Grounding of formulas)

The grounding of formulas is recursively defined according to their structure, and the fuzzy semantics of connectives.

- $P(t_1, \ldots, t_n)^{\mathcal{G}} = P^{\mathcal{G}}(t_1^{\mathcal{G}}, \ldots, t_n^{\mathcal{G}})$
- $(\neg\phi)^{\mathcal{G}} = 1 - \phi^{\mathcal{G}}$

Grounding FOL propositional formulas

Definition (Grounding of formulas)

The grounding of formulas is recursively defined according to their structure, and the fuzzy semantics of connectives.

- $P(t_1, \ldots, t_n)^{\mathcal{G}} = P^{\mathcal{G}}(t_1^{\mathcal{G}}, \ldots, t_n^{\mathcal{G}})$
- $(\neg\phi)^{\mathcal{G}} = 1 - \phi^{\mathcal{G}}$
- $(\phi \wedge \psi)^{\mathcal{G}} = \max(\phi^{\mathcal{G}} + \psi^{\mathcal{G}} - 1, 0)$

Grounding FOL propositional formulas

Definition (Grounding of formulas)

The grounding of formulas is recursively defined according to their structure, and the fuzzy semantics of connectives.

- $P(t_1, \dots, t_n)^{\mathcal{G}} = P^{\mathcal{G}}(t_1^{\mathcal{G}}, \dots, t_n^{\mathcal{G}})$
- $(\neg\phi)^{\mathcal{G}} = 1 - \phi^{\mathcal{G}}$
- $(\phi \wedge \psi)^{\mathcal{G}} = \max(\phi^{\mathcal{G}} + \psi^{\mathcal{G}} - 1, 0)$
- $(\phi \vee \psi)^{\mathcal{G}} = \min(\phi^{\mathcal{G}} + \psi^{\mathcal{G}}, 1)$

Grounding FOL propositional formulas

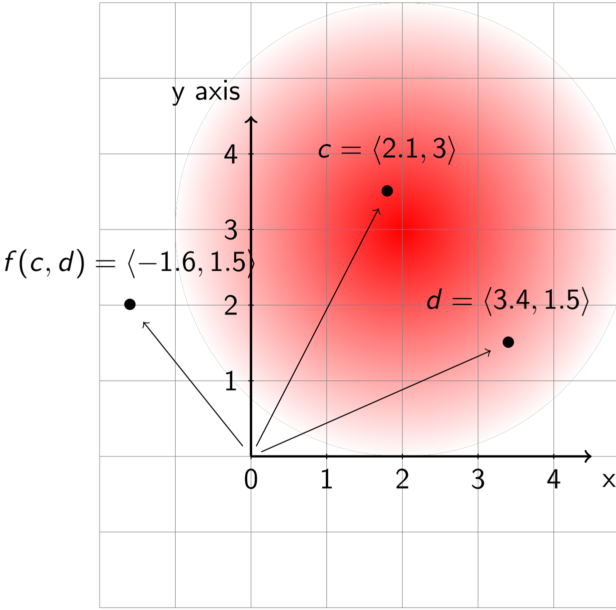
Definition (Grounding of formulas)

The grounding of formulas is recursively defined according to their structure, and the fuzzy semantics of connectives.

- $P(t_1, \ldots, t_n)^{\mathcal{G}} = P^{\mathcal{G}}(t_1^{\mathcal{G}}, \ldots, t_n^{\mathcal{G}})$
- $(\neg\phi)^{\mathcal{G}} = 1 - \phi^{\mathcal{G}}$
- $(\phi \wedge \psi)^{\mathcal{G}} = \max(\phi^{\mathcal{G}} + \psi^{\mathcal{G}} - 1, 0)$
- $(\phi \vee \psi)^{\mathcal{G}} = \min(\phi^{\mathcal{G}} + \psi^{\mathcal{G}}, 1)$
- $(\phi \rightarrow \psi)^{\mathcal{G}} = min(1 - \phi^{\mathcal{G}} + \psi^{\mathcal{G}}, 1)$

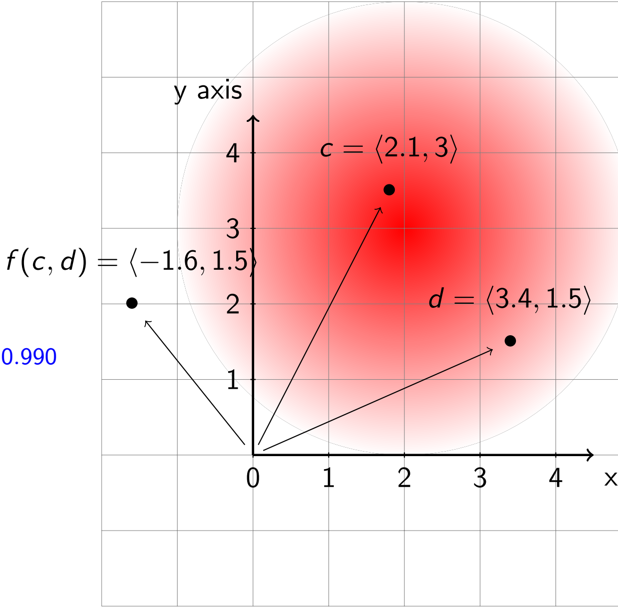
Grounding FOL in $\mathbb{R}^2$

- $c^{\mathcal{G}} = \langle 2.1, 3\rangle$
- $d^{\mathcal{G}} = \langle 3.4, 1.5\rangle$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \vec{x} - \vec{y}$
- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((\vec{x} - \vec{\mu})^2\right)$



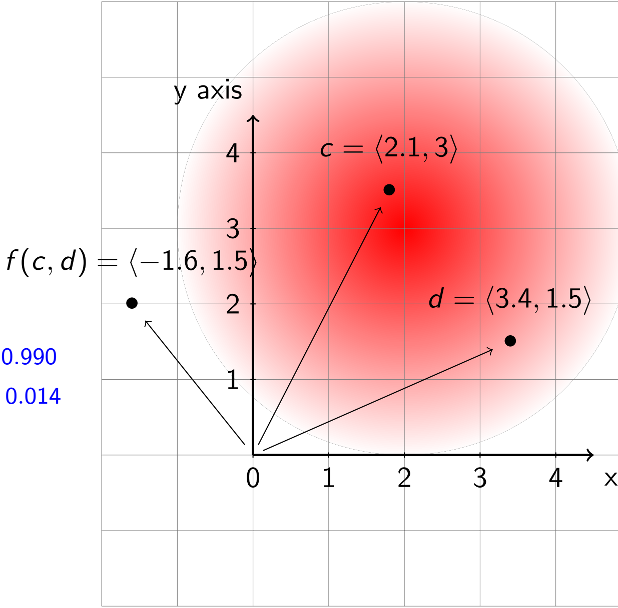
Grounding FOL in $\mathbb{R}^2$

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \langle 3.4, 1.5 \rangle$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \vec{x} - \vec{y}$
- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((\vec{x} - \vec{\mu})^2\right)$
- $P(c)^{\mathcal{G}} = exp(||c^{\mathcal{G}} - \vec{\mu}||^2) = 0.990$



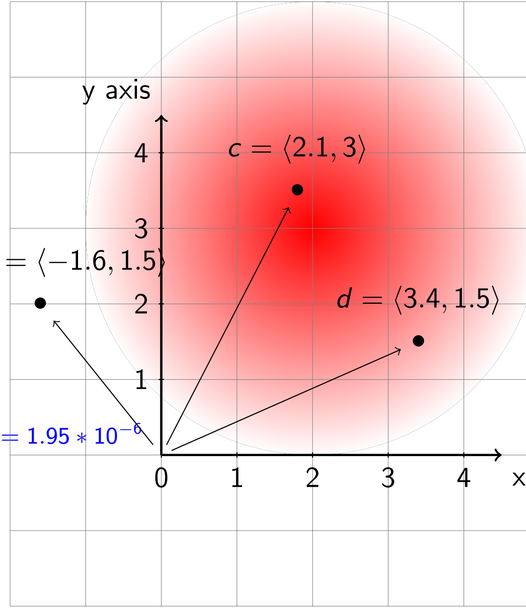
Grounding FOL in $\mathbb{R}^2$

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \langle 3.4, 1.5 \rangle$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \vec{x} - \vec{y}$
- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((\vec{x} - \vec{\mu})^2\right)$
- $P(c)^{\mathcal{G}} = exp(||c^{\mathcal{G}} - \vec{\mu}||^2) = 0.990$
- $P(d)^{\mathcal{G}} = exp(||d^{\mathcal{G}} - \vec{\mu}||^2) = 0.014$



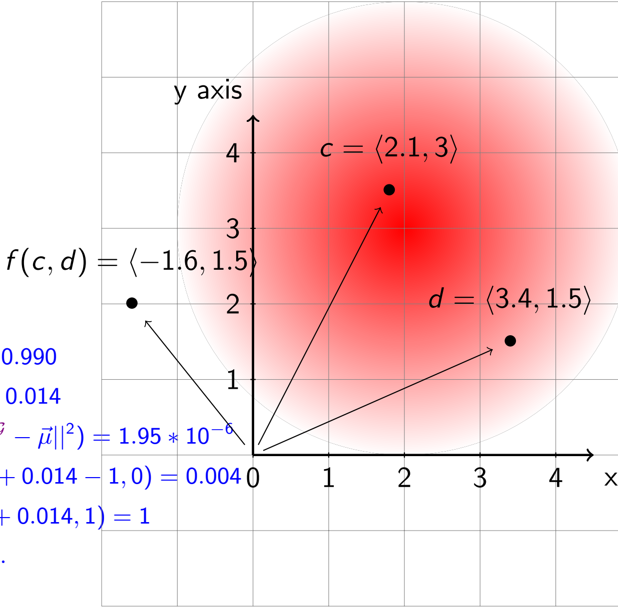
Grounding FOL in $\mathbb{R}^2$

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \langle 3.4, 1.5 \rangle$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \vec{x} - \vec{y}$
- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((\vec{x} - \vec{\mu})^2\right)$
- $P(c)^{\mathcal{G}} = exp(||c^{\mathcal{G}} - \vec{\mu}||^2) = 0.990$
- $P(d)^{\mathcal{G}} = exp(||d^{\mathcal{G}} - \vec{\mu}||^2) = 0.014$
- $P(f(c,d))^{\mathcal{G}} = exp(||c^{\mathcal{G}} - d^{\mathcal{G}} - \vec{\mu}||^2) = 1.95 * 10^{-6}$



Grounding FOL in $\mathbb{R}^2$

- $c^{\mathcal{G}} = \langle 2.1, 3 \rangle$
- $d^{\mathcal{G}} = \langle 3.4, 1.5 \rangle$
- $f^{\mathcal{G}} : \vec{x}, \vec{y} \mapsto \vec{x} - \vec{y}$
- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((\vec{x} - \vec{\mu})^2\right)$
- $P(c)^{\mathcal{G}} = exp(||c^{\mathcal{G}} - \vec{\mu}||^2) = 0.990$
- $P(d)^{\mathcal{G}} = exp(||d^{\mathcal{G}} - \vec{\mu}||^2) = 0.014$
- $P(f(c,d))^{\mathcal{G}} = exp(||c^{\mathcal{G}} - d^{\mathcal{G}} - \vec{\mu}||^2) = 1.95 * 10^{-6}$
- $P(c) \wedge P(d)^{\mathcal{G}} = \max(0.990 + 0.014 - 1, 0) = 0.004$
- $P(c) \vee P(d)^{\mathcal{G}} = \min(0.990 + 0.014, 1) = 1$
- $\neg P(f(c,d)) \rightarrow P(d))^{\mathcal{G}} = \ldots$



LTN code

Example (TODO)

add the code here

Grounding FOL quantifier $\forall$

In theory

Often in fuzzy logic, the semantics of $\forall x \phi(x)$ is given in terms of min aggregation

$$(\forall x \phi(x))^{\mathcal{G}} = \min_{\mathbf{x} \in \mathbb{R}^k} \phi^{\mathcal{G}}(\mathbf{x})$$

However, this involves an uncountably infinite number of instances $\mathbf{x}$ and therefore is difficult to compute directly

Grounding FOL quantifier $\forall$

In theory

Often in fuzzy logic, the semantics of $\forall x \phi(x)$ is given in terms of min aggregation

$$(\forall x \phi(x))^{\mathcal{G}} = \min_{\mathbf{x} \in \mathbb{R}^k} \phi^{\mathcal{G}}(\mathbf{x})$$

However, this involves an uncountably infinite number of instances $\mathbf{x}$ and therefore is difficult to compute directly

In practice

We consider a domain sample, i.e., a finite subset $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of $\mathbb{R}^k$ and define

$$(\forall x \phi(x))^{\mathcal{G}} = \min_{i=1}^{n} \phi^{\mathcal{G}}(\mathbf{x}_i)$$

Grounding FOL quantifiers $\forall, \exists$

All quantifier $\forall$

We consider a domain sample, i.e., a finite subset $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of $\mathbb{R}^k$ and define

$$(\forall x \phi(x))^{\mathcal{G}} = \min_{i=1}^{n} \phi^{\mathcal{G}}(\mathbf{x}_i)$$

Existential quantifier $\exists$

We consider a domain sample, i.e., a finite subset $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of $\mathbb{R}^k$ and define

$$(\exists x \phi(x))^{\mathcal{G}} = \max_{i=1}^{n} \phi^{\mathcal{G}}(\mathbf{x}_i)$$

LTN Variables

- LTN variables corresponds to FOL individual variables
- in LTN to write the formula $\forall x P(x)$ you have to declare x to be an LTN variable
- LTN variables are associated to a **finite** domain sample, which is the set of all values that can be taken by that variable

LTN Variables

- LTN variables corresponds to FOL individual variables
- in LTN to write the formula $\forall x P(x)$ you have to declare x to be an LTN variable
- LTN variables are associated to a **finite** domain sample, which is the set of all values that can be taken by that variable

Example (LTN variables)

- x is an LTN variable associated to the domain samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)} \in \mathbb{R}^k$
- y is an LTN variable associated to the domain samples $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(m)} \in \mathbb{R}^k$
- $(\forall x, P(x))^{\mathcal{G}} = \min_{i=1}^{n}(P(\mathbf{x}^{(i)}))^{\mathcal{G}}$
- $(\forall y, P(y))^{\mathcal{G}} = \min_{i=1}^{m}(P(\mathbf{y}^{(i)}))^{\mathcal{G}}$

LTN Variables

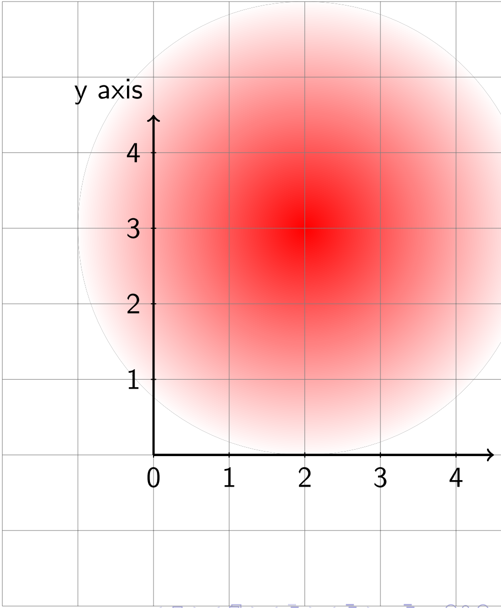
- LTN variables corresponds to FOL individual variables
- in LTN to write the formula $\forall x P(x)$ you have to declare x to be an LTN variable
- LTN variables are associated to a **finite** domain sample, which is the set of all values that can be taken by that variable

Example (LTN variables)

- x is an LTN variable associated to the domain samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)} \in \mathbb{R}^k$
- y is an LTN variable associated to the domain samples $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(m)} \in \mathbb{R}^k$
- $(\forall x, P(x))^{\mathcal{G}} = \min_{i=1}^{n}(P(\mathbf{x}^{(i)}))^{\mathcal{G}}$
- $(\forall y, P(y))^{\mathcal{G}} = \min_{i=1}^{m}(P(\mathbf{y}^{(i)}))^{\mathcal{G}}$
- notice that in general $\forall x P(x)$ is not equivalent to $\forall y P(x)$; it depends on the domain samples associated to x and y

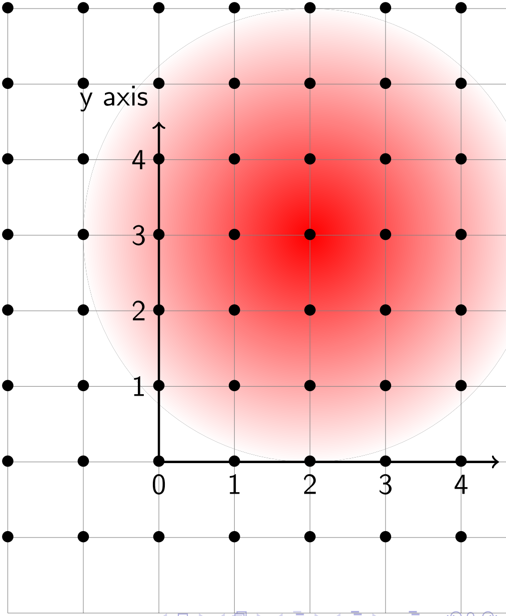
Grounding FOL in $\mathbb{R}^2$

- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((\vec{x} - \vec{\mu})^2\right)$



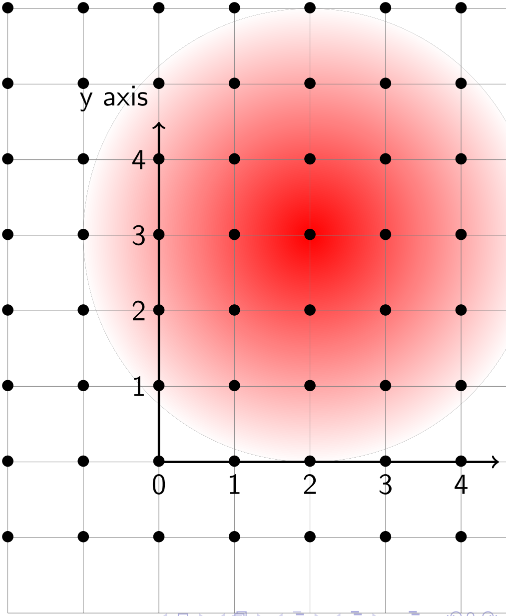
Grounding FOL in $\mathbb{R}^2$

- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((\vec{x} - \vec{\mu})^2\right)$
- x is an LTN variable with domain S_x
- $S_x = \left\{ \langle a, b \rangle \;\middle|\; \begin{array}{l} a = -2, -1, 0, \ldots, 4 \\ b = -1, 0, \ldots, 4, 5, 6 \end{array} \right\}$

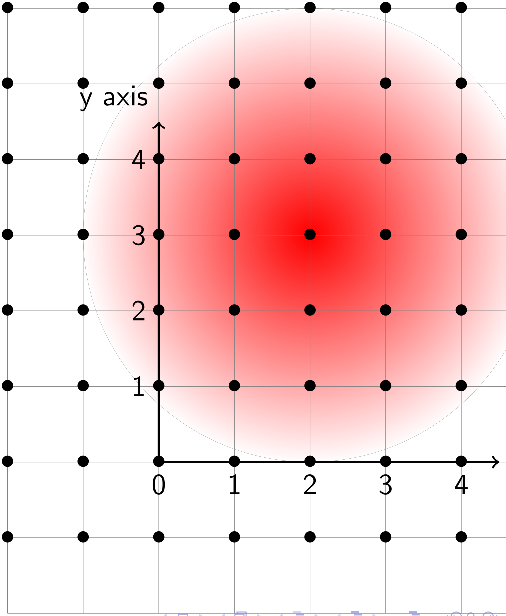


Grounding FOL in $\mathbb{R}^2$

- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((\vec{x} - \vec{\mu})^2\right)$
- x is an LTN variable with domain S_x
- $S_x = \left\{ \langle a, b \rangle \;\middle|\; \begin{array}{l} a = -2, -1, 0, \ldots, 4 \\ b = -1, 0, \ldots, 4, 5, 6 \end{array} \right\}$
- $P(x)^{\mathcal{G}} = \{P(\mathbf{x})^{\mathcal{G}} \mid \mathbf{x} \in S_x\}$



Grounding FOL in $\mathbb{R}^2$



- $P^{\mathcal{G}} : \vec{x} \mapsto \exp\left((\vec{x} - \vec{\mu})^2\right)$
- x is an LTN variable with domain S_x
- $S_x = \left\{ \langle a, b \rangle \;\middle|\; \begin{array}{l} a = -2, -1, 0, \ldots, 4 \\ b = -1, 0, \ldots, 4, 5, 6 \end{array} \right\}$
- $P(x)^{\mathcal{G}} = \{P(\mathbf{x})^{\mathcal{G}} \mid \mathbf{x} \in S_x\}$
- $\forall x : P(x)^{\mathcal{G}} = \min P(x)^{\mathcal{G}} \approx 0$
- $\exists x : P(x)^{\mathcal{G}} = \max P(x)^{\mathcal{G}} = 1$

LTN code

Example (TODO)

add the code here

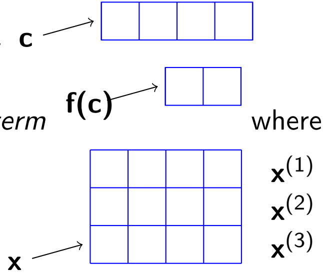
Logic Tensor Networks - representation

- a *constant* **c** 

Logic Tensor Networks - representation

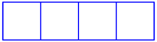
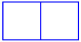
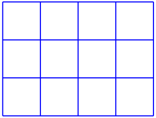
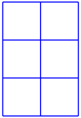
- a *constant* **c**
- a *ground term* **f(c)** where $f : \mathbb{R}^4 \to \mathbb{R}^2$;

Logic Tensor Networks - representation

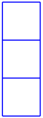


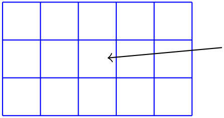
- a *constant*
- a *ground term* where $f : \mathbb{R}^4 \to \mathbb{R}^2$;
- a variable

Logic Tensor Networks - representation

- a *constant* $\mathbf{c}$ $\rightarrow$ 
- a *ground term* $\mathbf{f(c)}$ $\rightarrow$  where $f : \mathbb{R}^4 \rightarrow \mathbb{R}^2$;
- a variable $\mathbf{x}$ $\rightarrow$  $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$
- a term $f(\mathbf{x})$ $\rightarrow$  $f(\mathbf{x}^{(1)})$, $f(\mathbf{x}^{(2)})$, $f(\mathbf{x}^{(3)})$

Logic Tensor Networks - representation

- a ground atom $P(c)$ 
- an open atom $P(\mathbf{x})$ 

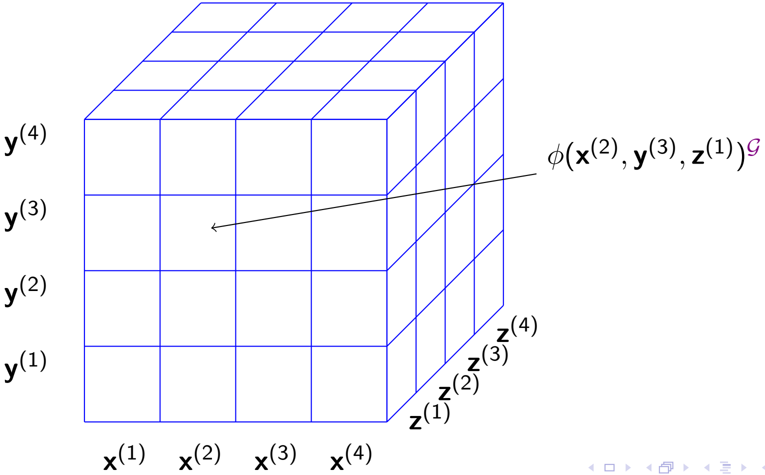
  $P(\mathbf{x}^{(1)})$
  $P(\mathbf{x}^{(2)})$
  $P(\mathbf{x}^{(3)})$
- an open atom $R(\mathbf{x}, \mathbf{y})$ 

  $R(\mathbf{x}^{(2)}, \mathbf{y}^{(3)})$

Logic Tensor Networks - representation

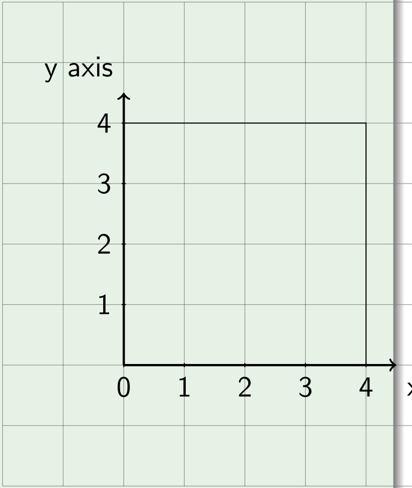
Example: A formula with three free variables $P(x,y) \wedge P(z)$ is represented with a 3D tensor T

$$T_{i,j,k} = \left(\phi\left(\mathbf{x}^{(i)}, \mathbf{y}^{(j)}, \mathbf{z}^{(k)}\right)\right)^{\mathcal{G}}$$



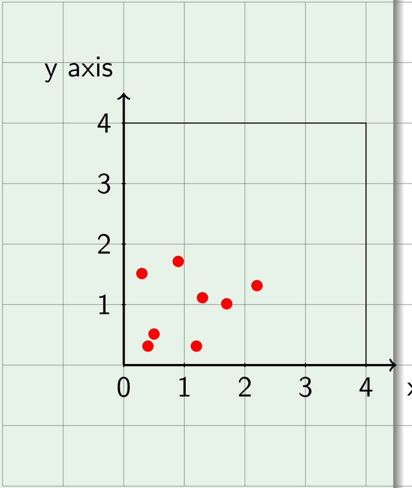
Example

- the domain is the square $[0,4] \times [0,4]$;



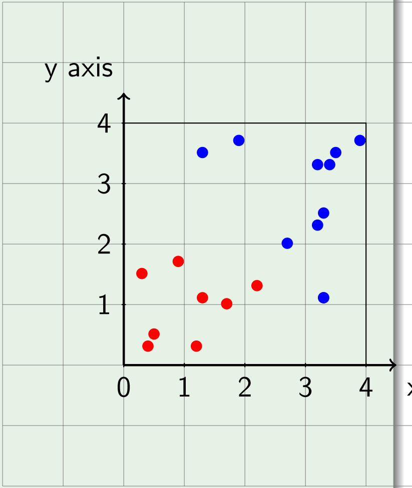
Example

- the domain is the square $[0, 4] \times [0, 4]$;
- we have a set of examples of the class A



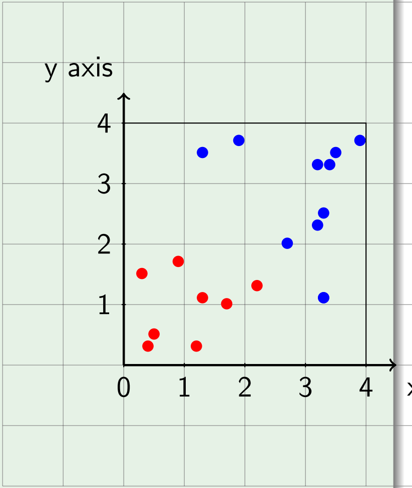
Example

- the domain is the square $[0, 4] \times [0, 4]$;
- we have a set of examples of the class A
- and a set of examples the the class B



Example

- the domain is the square $[0, 4] \times [0, 4]$;
- we have a set of examples of the class A
- and a set of examples the the class B
- we know that A and B are disjoint

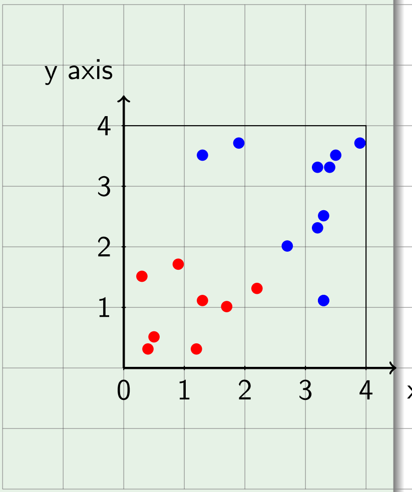


Example

- the domain is the square $[0,4] \times [0,4]$;
- we have a set of examples of the class A
- and a set of examples the the class B
- we know that A and B are disjoint
- and that the shape of the membership function of the classes is

$$\sigma(w_1 \cdot x + w_2 \cdot y + w_3)$$

with $\sigma(x)$ the sigmoid function $\frac{1}{1+e^{-x}}$



Example

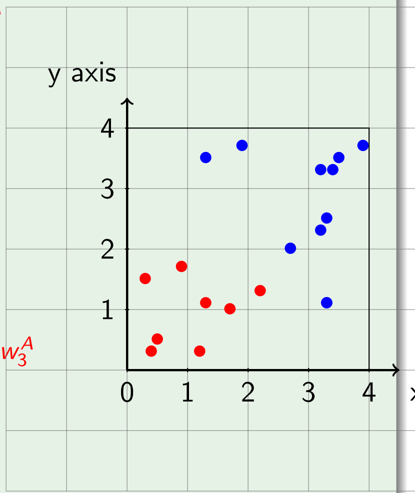
- the domain is the square $[0,4] \times [0,4]$;
- we have a set of examples of the class A
- and a set of examples the the class B
- we know that A and B are disjoint
- and that the shape of the membership function of the classes is

$$\sigma(w_1 \cdot x + w_2 \cdot y + w_3)$$

with $\sigma(x)$ the sigmoid function $\frac{1}{1+e^{-x}}$

- we have to find the parameters w_1^A, w_2^A, w_3^A and w_1^B, w_2^B, w_3^B that maximise the satisfiability of the formulas:

$$A(\mathbf{x}) \wedge B(\mathbf{y}) \wedge \forall x : A(x) \rightarrow \neg B(x)$$



Example

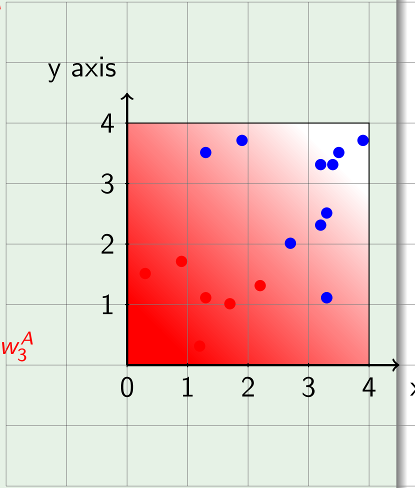
- the domain is the square $[0,4] \times [0,4]$;
- we have a set of examples of the class A
- and a set of examples the the class B
- we know that A and B are disjoint
- and that the shape of the membership function of the classes is

$$\sigma(w_1 \cdot x + w_2 \cdot y + w_3)$$

with $\sigma(x)$ the sigmoid function $\frac{1}{1+e^{-x}}$

- we have to find the parameters w_1^A, w_2^A, w_3^A and w_1^B, w_2^B, w_3^B that maximise the satisfiability of the formulas:

$$A(\mathbf{x}) \wedge B(\mathbf{y}) \wedge \forall x : A(x) \rightarrow \neg B(x)$$



Example

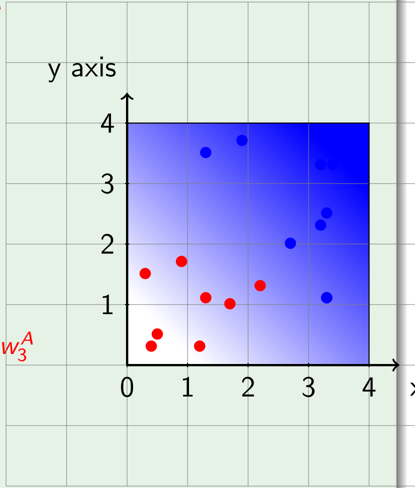
- the domain is the square $[0,4] \times [0,4]$;
- we have a set of examples of the class A
- and a set of examples the the class B
- we know that A and B are disjoint
- and that the shape of the membership function of the classes is

$$\sigma(w_1 \cdot x + w_2 \cdot y + w_3)$$

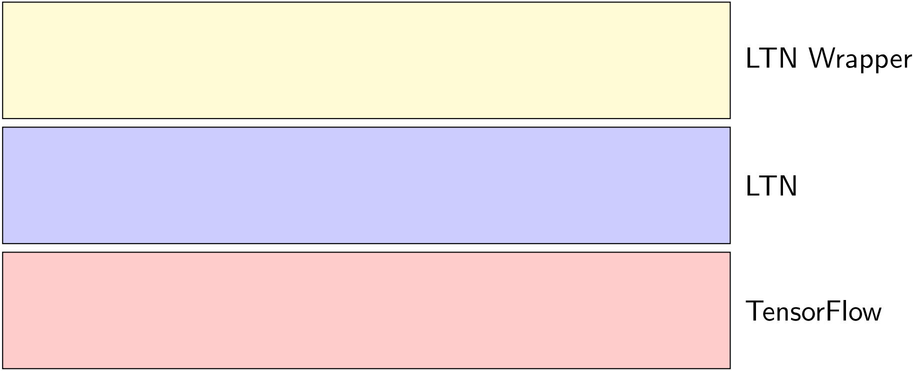
with $\sigma(x)$ the sigmoid function $\frac{1}{1+e^{-x}}$

- we have to find the parameters w_1^A, w_2^A, w_3^A and w_1^B, w_2^B, w_3^B that maximise the satisfiability of the formulas:

$$A(\mathbf{x}) \wedge B(\mathbf{y}) \wedge \forall x : A(x) \rightarrow \neg B(x)$$



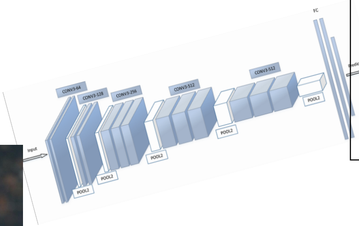
Logic Tensor Networks - Architecture



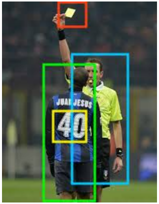
Logic Tensor Networks - Architecture

`"forall x:P(x,y) | A(x)"`	LTN Wrapper
`Forall(x,Or(P(x,y),A(x)))`	LTN
`tf.reduce_min(tf.max(P(x,y),A(x)),axis=0))`	TensorFlow

LTN = Combining Neural Nets with Fuzzy Logic

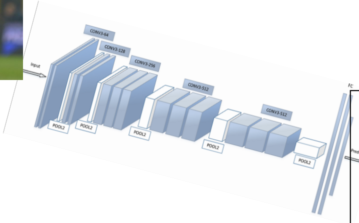


number(x)
player(x)
referee(x)
ball(x)
leg(x)
yellowCard(x)
...



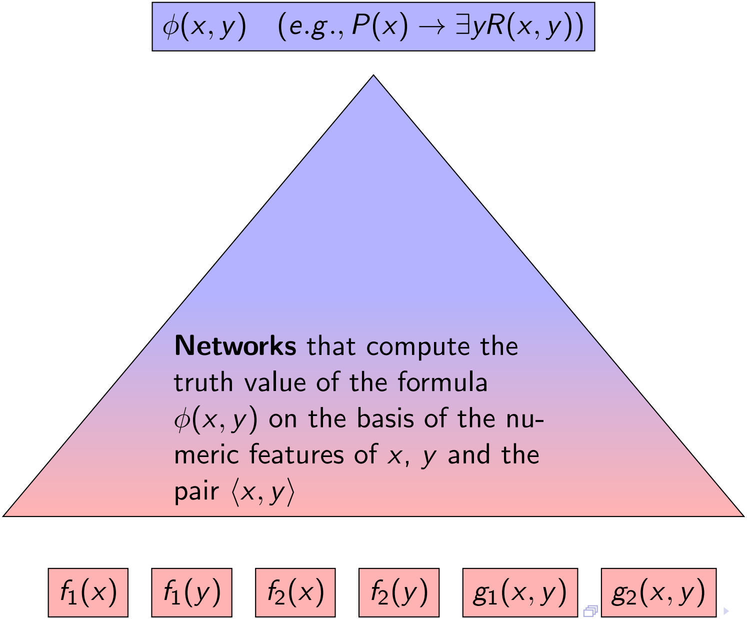
$\forall x, y$
$\quad Number(x) \wedge$
$\quad PartOf(x, y) \rightarrow$
$\qquad Player(x)$

$\forall x, y$
$\quad YellowCard(x) \wedge$
$\quad Holds(x, y) \rightarrow$
$\qquad Referee(x)$

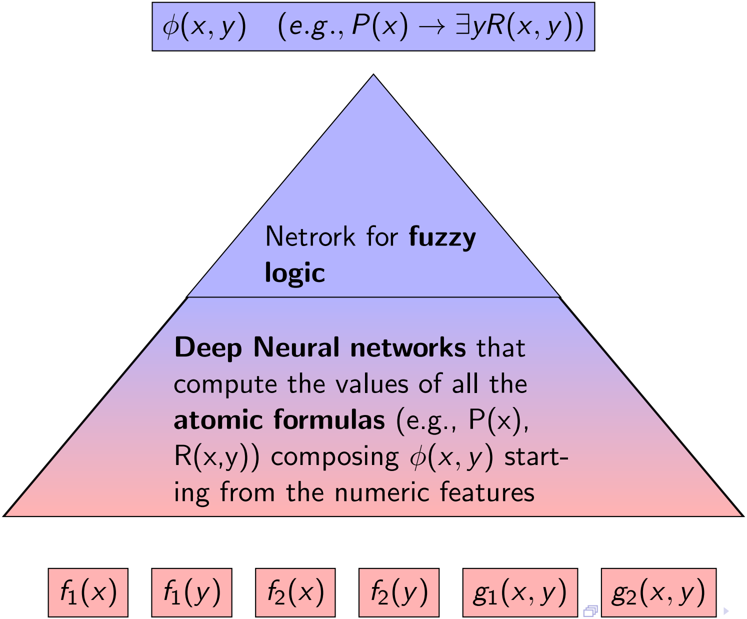


partOf(x,y)
leftOf(x,y)
kicks(x,y)
hold(x,y)
...

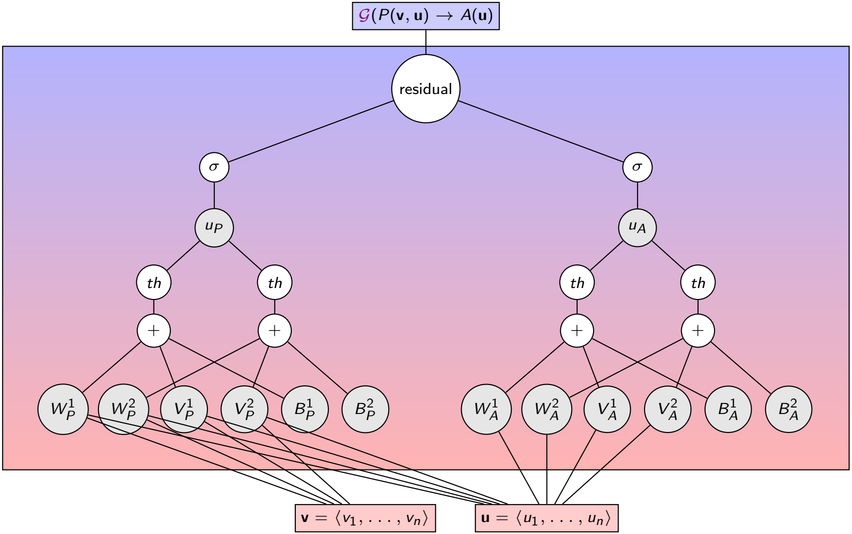
Logic Tensor Network – at a glance



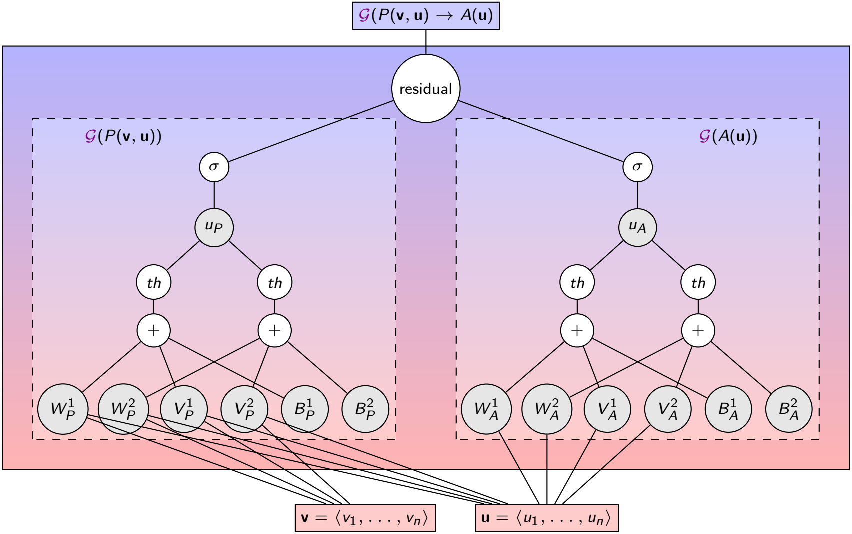
Logic Tensor Network – at a glance



Logic Tensor Network – at a glance



Logic Tensor Network – at a glance



Parameter learning = best satisfiability

Given a FOL theory $\mathcal{T}$ the **best satisfiability problem** as the problem of finding the set of parameters Θ of the LTN, then the problems become $\mathcal{G}^* = LTN(K, \Theta^*)$

$$\Theta^* = \operatorname*{argmax}_{\Theta} \left(\min_{\mathcal{T} \models \phi} LTN(K, \Theta)(\phi) \right)$$

Parameter learning = best satisfiability

Given a FOL theory $\mathcal{T}$ the **best satisfiability problem** as the problem of finding the set of parameters Θ of the LTN with $\mathcal{G}^* = LTN(\mathcal{T}, \Theta^*)$

$$\Theta^* = \operatorname*{argmax}_{\Theta} \left(\min_{\phi \in \mathcal{T}} LTN(\mathcal{T} \mid \Theta)(\phi) \right)$$

Thanks

LTN has been developed thanks to active contributions, feedback, and discussions with the following people:

- Alessandro Daniele (FBK)
- Artur d'Avila Garces (City)
- Francesco Giannini (UniSiena)
- Giuseppe Marra (UniSiena)
- Ivan Donadello (FBK)
- Lucas Brukberger (UniOsnabruck)
- Luciano Serafini (FBK)
- Marco Gori (UniSiena)
- Michael Spranger (Sony CSL)
- Michelangelo Diligenti (UniSiena)

Leon Barrett, Jerome Feldman, and Liam MacDermed.
A (somewhat) new solution to the variable binding problem.
Neural Computation, 20(9):2361–2378, 2008.

Angelo Cangelosi.
Connectionist and robotics approaches to grounding symbols in perceptual and sensorimotor categories.
In *Handbook of Categorization in Cognitive Science (Second Edition)*, pages 795–818. Elsevier, 2017.

Michelangelo Diligenti, Marco Gori, Marco Maggini, and Leonardo Rigutini.
Bridging logic and kernel machines.
Machine Learning, 86(1):57–88, 2012.

Stevan Harnad.
The symbol grounding problem.
Physica D: Nonlinear Phenomena, 42(1-3):335–346, 1990.

Michael J Mayo.
Symbol grounding and its implications for artificial intelligence.

In *Proceedings of the 26th Australasian computer science conference-Volume 16*, pages 55–60. Australian Computer Society, Inc., 2003.

Allen Newell.
Unified theories of cognition.
Harvard University Press, 1994.

John R Searle.
Minds, brains, and programs.
Behavioral and brain sciences, 3(3):417–424, 1980.

Luciano Serafini and Artur S d'Avila Garcez.
Learning and reasoning with logic tensor networks.
In *Conference of the Italian Association for Artificial Intelligence*, pages 334–348. Springer, 2016.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng.
Reasoning with neural tensor networks for knowledge base completion.

In *Advances in neural information processing systems*, pages 926–934, 2013.