Logic Tensor Networks

Lecture 6

Overview of this Lecture

- Real Logic
  - Tensors
  - Groundings
  - Fuzzy operators
- Logic Tensor Networks
  - From Real Logic to LTN
  - LTN Tasks
  - Use Cases
- Bonus: Differentiable Fuzzy Logic

Introduction

- Logic Tensor Networks (LTN) is a recently-developed NSR framework based on **fuzzy differentiable logic**.
- Supports tasks based on manipulating **data** and **knowledge**.
- It has been shown to effectively tackle many tasks that are central to intelligent systems:
  - multi-label classification,
  - relational learning,
  - data clustering,
  - semi-supervised learning,
  - regression,
  - embedding learning, and
  - query answering under uncertainty.

Logic Tensor Networks

Real Logic

Real Logic: Introduction

LTN uses an infinitely-valued fuzzy logical language called **Real Logic** as the underlying formalism:

- Domains are interpreted concretely by **tensors** in the Real field.
- Recall that tensors are **algebraic objects** that include:
  - Scalars: 0-dimensional,
  - Vectors: 1-dimensional,
  - Matrices: 2-dimensional,
  - as well as higher-dimension structures.
- To emphasize this, the authors use the term "**grounding**", denoted with the letter $\mathcal{G}$, instead of "interpretation" (the usual name for this concept in logic).

Source: https://github.com/logictensornetworks/

Real Logic: Introduction (cont.)

LTN uses an infinitely-valued fuzzy logical language called **Real Logic** as the underlying formalism:

- Grounding $\mathcal{G}$ maps:
  - Terms to **tensors** of real numbers, and
  - **Formulas** to **real numbers** in the interval [0,1].
- We commonly use "tensor" to abbreviate the expression "tensor in the Real field".
- As usual, the language allows for logical **connectives** and **quantifiers**.

Source: https://github.com/logictensornetworks/

Real Logic: Language

Constants:

- Denote **individuals** from a space of tensors $\bigcup_{n_1 \ldots nd \in} {}^{*} \mathbb{R}^{n1 \times \cdots \times nd}$ (from now on, we write “tensor of any rank” to denote this expression).
- The individual can be **pre-defined** (data point) or **learnable** (embedding).
- **Intuition**: Each dimension corresponds to a **feature**, and the number corresponds to the value of that feature for that individual.

Variables denote sequences of individuals:

- Sequences represent the **possible values** that the variable can take.
- They can contain more than one instance of the same value.

Source: https://github.com/logictensornetworks/

Real Logic: Language (cont.)

Functions:

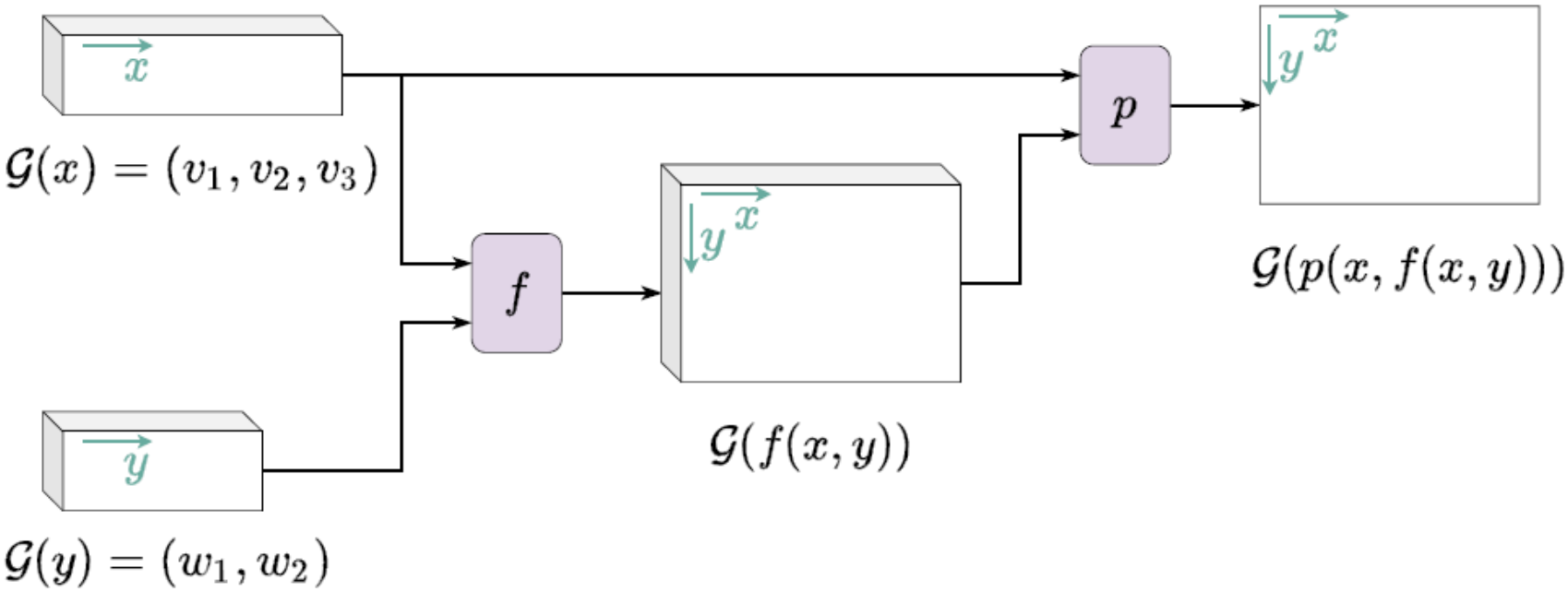
- Can be any mathematical function (either pre-defined or learnable).
- Examples of functions are distance functions, regressors, etc.

Predicates:

- Represented as mathematical functions that map an n-ary domain of individuals to a real in [0,1], interpreted as a **truth degree**.
- Examples of predicates: similarity measures, classifiers, etc.

Source: https://github.com/logictensornetworks/

Real Logic: Grounding of Functions and Predicates



Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Real Logic: Language (cont.)

Connectives are modeled using fuzzy semantics:

- Conjunction ($\wedge$): t-norm T
- Disjunction ($\vee$): t-conorm S
- Implication ($\Rightarrow$): fuzzy implication I
- Negation ($\neg$): fuzzy negation N

$$\text{FuzzyOp} \in \{\text{T}, \text{S}, \text{I}, \text{N}\}$$

We will come back to these operators later when discussing differentiability.

Quantifiers are defined using **aggregators** (symmetric and continuous operators)

- Existential ($\exists$): Generalization of existential quantification in FOL
- Universal ($\forall$): Generalization of universal quantification in FOL

Source: https://github.com/logictensornetworks/

Real Logic: Common Fuzzy Operators

Name	$a \wedge b$	$a \vee b$	$a \rightarrow_R c$	$a \rightarrow_S c$
Goedel	$\min(a, b)$	$\max(a, b)$	$\begin{cases} 1, & \text{if } a \leqslant c \\ c, & \text{otherwise} \end{cases}$	$\max(1 - a, c)$
Goguen/Product	$a \cdot b$	$a + b - a \cdot b$	$\begin{cases} 1, & \text{if } a \leqslant c \\ \frac{c}{a}, & \text{otherwise} \end{cases}$	$1 - a + a \cdot c$
Łukasiewicz	$\max(a + b - 1, 0)$	$\min(a + b, 1)$	$\min(1 - a + c, 1)$	$\min(1 - a + c, 1)$

Name	$I(x, y) =$	S-Implication	R-Implication
Kleene-Dienes I_{KD}	$\max(1 - x, y)$	$S = S_M$ $N = N_S$	-
Goedel I_G	$\begin{cases} 1, & x \leqslant y \\ y, & \text{otherwise} \end{cases}$	-	$T = T_M$
Reichenbach I_R	$1 - x + xy$	$S = S_P$ $N = N_S$	-
Goguen I_P	$\begin{cases} 1, & x \leqslant y \\ y/x, & \text{otherwise} \end{cases}$	-	$T = T_P$
Łukasiewicz I_{Luk}	$\min(1 - x + y, 1)$	$S = S_L$ $N = N_S$	$T = T_L$

Source: https://github.com/logictensornetworks/

Real Logic: Common Fuzzy Operators

Some possible **aggregators** for fuzzy **universal** quantification:

$$A_{T_M}(x_1, \ldots, x_n) = \min(x_1, \ldots, x_n) \qquad \text{(Minimum)}$$

$$A_{T_P}(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i \qquad \text{(Product)}$$

$$A_{T_L}(x_1, \ldots, x_n) = \max(\sum_{i=1}^{n} x_i - n + 1, 0) \qquad \text{(Lukasiewicz)}$$

Some possible **aggregators** for fuzzy **existential** quantification:

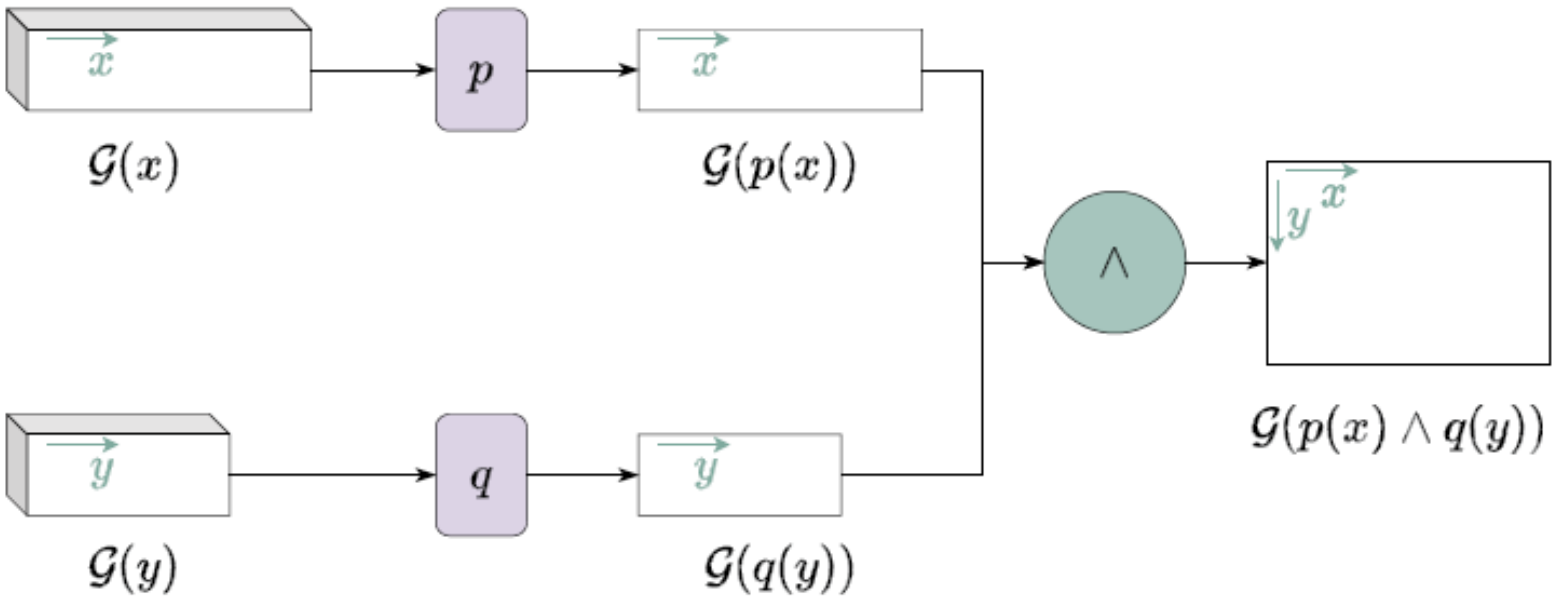
$$A_{S_M}(x_1, \ldots, x_n) = \max(x_1, \ldots, x_n) \qquad \text{(Maximum)}$$

$$A_{S_P}(x_1, \ldots, x_n) = 1 - \prod_{i=1}^{n} (1 - x_i) \qquad \text{(Probabilistic Sum)}$$

$$A_{S_L}(x_1, \ldots, x_n) = \min(\sum_{i=1}^{n} x_i, 1) \qquad \text{(Lukasiewicz)}$$

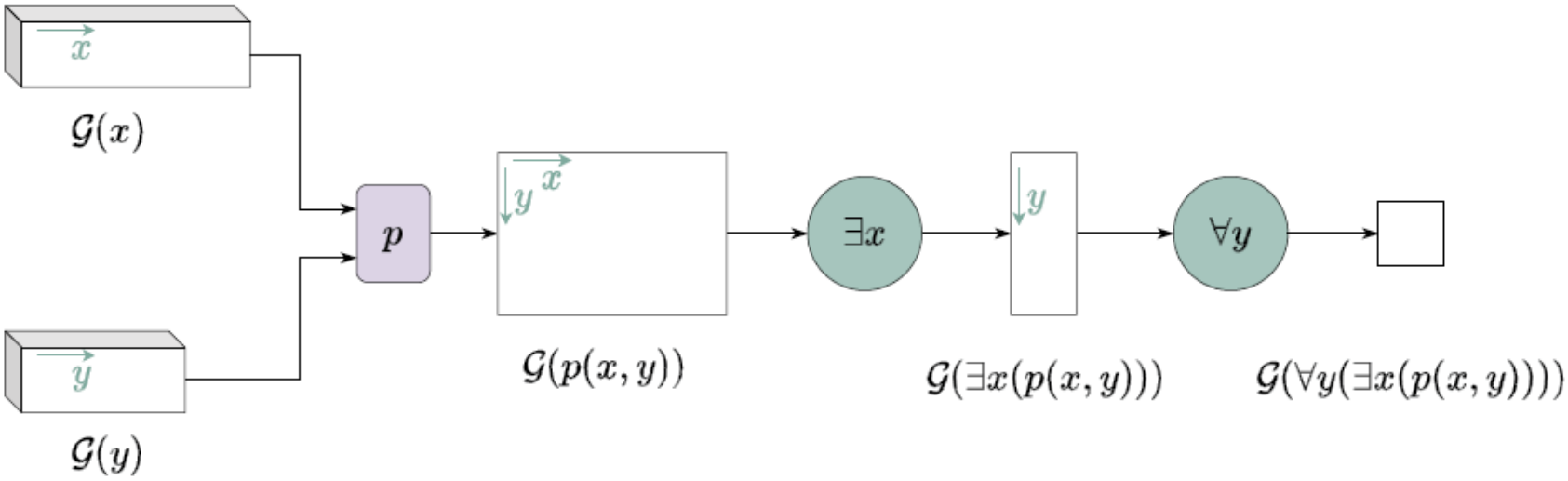
Source: https://github.com/logictensornetworks/

Real Logic: Conjunction



Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Real Logic: Quantification



Note: Depending on the properties that the aggregation operators enjoy, the semantics may or may not adequately generalize that of FOL.

(For instance, commutativity of quantifiers may not be guaranteed.)

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Real Logic: Language (cont.)

Diagonal quantification:

- Diag(x_1, ..., x_h) quantifies over **specific tuples** s.t. the *i*-th tuple contains the *i*-th instance of each of the variables in the argument of Diag.
- Assumes that all variables in the argument are grounded onto sequences with the same number of instances.

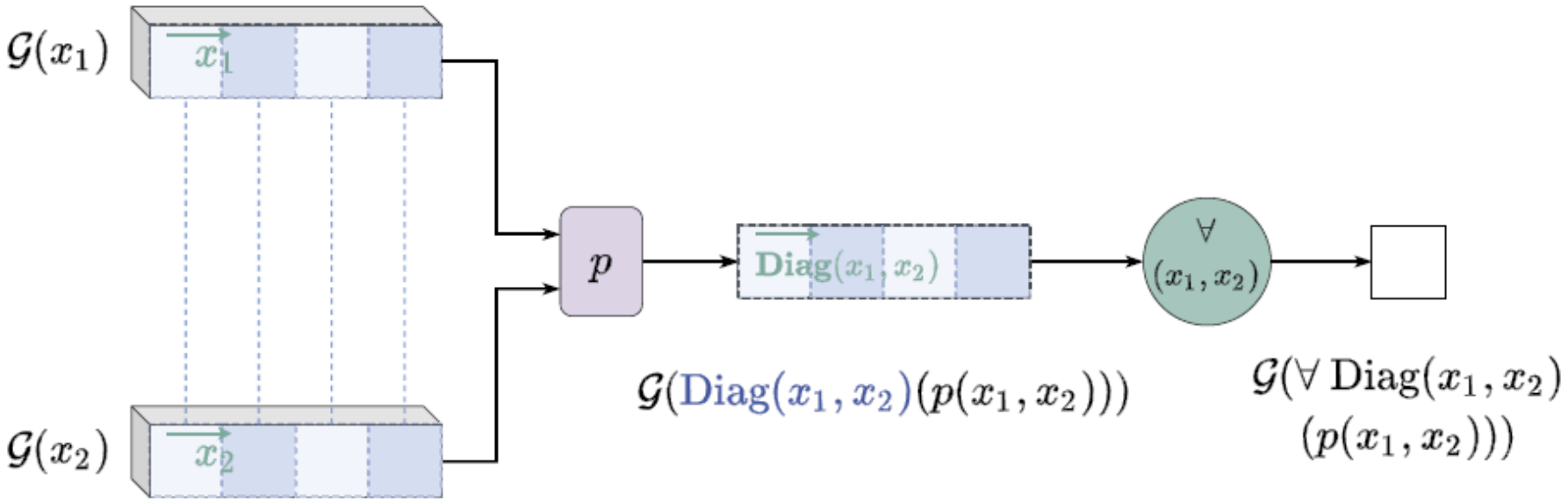
Guarded quantification:

- Quantifies over a **subset** of variables that satisfy a **condition** m (mask)
- Definition:

$$\mathcal{G}(Q\ x_1,\ldots,x_h : m(x_1,\ldots,x_n)(\phi))_{i_{h+1},\ldots,i_n} \overset{\text{def}}{=} \underset{\substack{i_1=1,\ldots,|\mathcal{G}(x_1)| \\ \vdots \\ i_h=1,\ldots,|\mathcal{G}(x_h)| \text{ s.t.} \\ \mathcal{G}(m)(\mathcal{G}(x_1)_{i_1},\ldots,\mathcal{G}(x_n)_{i_n})}}{\text{Agg(Q)}} \mathcal{G}(\phi)_{i_1,\ldots,i_h,i_{h+1},\ldots,i_n}$$

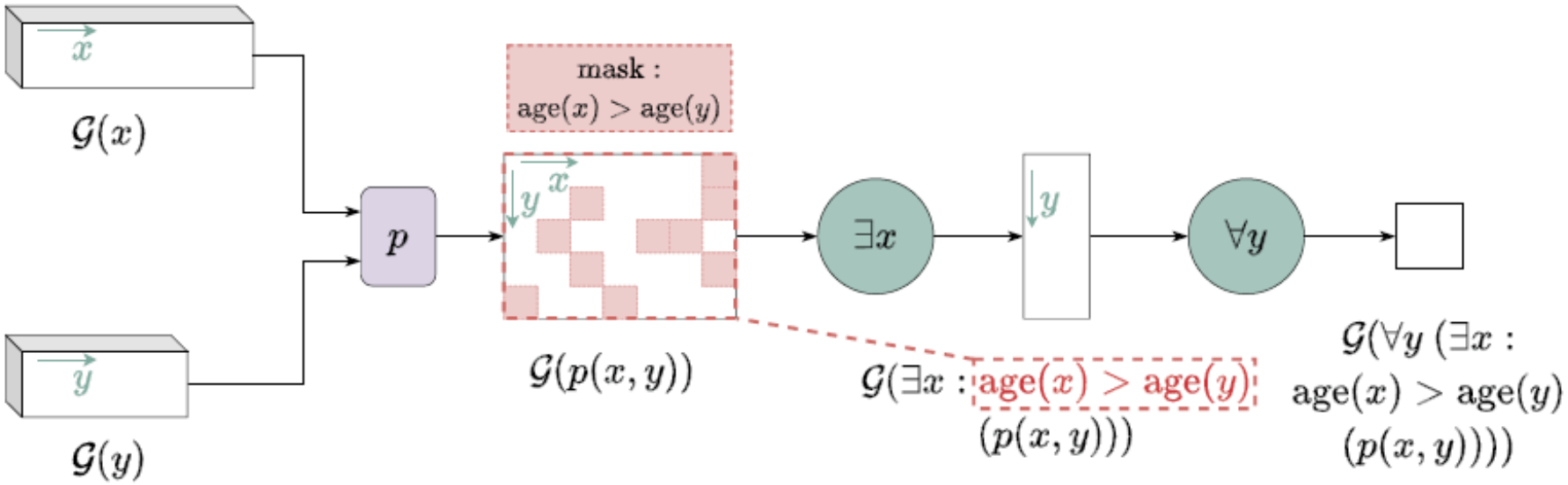
Source: https://github.com/logictensornetworks/

Real Logic: Diagonal Quantification



Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Real Logic: Guarded Quantification



Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Logic Tensor Networks

From Real Logic to LTNs

From Real Logic to Logic Tensor Networks

- Up to now, we have presented a kind of fuzzy logic – where do we go from here?
- Towards a machine learning setup:
  - **Objects** are represented by points in a feature space.
  - Functions and predicates are **learnable**.
- Let's illustrate this with some examples…

From Real Logic to Logic Tensor Networks

$friend(Mary, John)$

Individuals are grounded with real features (e.g. vectors, matrices, ...).

e.g. $\mathcal{G}(Mary) = [6.2, 1.5, \ldots] \in \mathbb{R}^m$

Predicates are grounded with operations (e.g. neural networks, ...)
that project in the interval [0,1].
The output denotes a *satisfaction level*.

e.g. $\mathcal{G}(friend) : \mathbb{R}^m \times \mathbb{R}^m \longrightarrow [0, 1]$

$\forall x(friend(John, x) \rightarrow friend(Mary, x))$

Connectives ($\wedge, \vee, \rightarrow, \neg$) are interpreted using fuzzy semantics

e.g. $0.7 \wedge_{\text{prod}} 0.2 = 0.7 \cdot 0.2 = 0.14$

Variables are grounded as a list of n individuals

e.g. $x \in \mathbb{R}^{n \times m}$

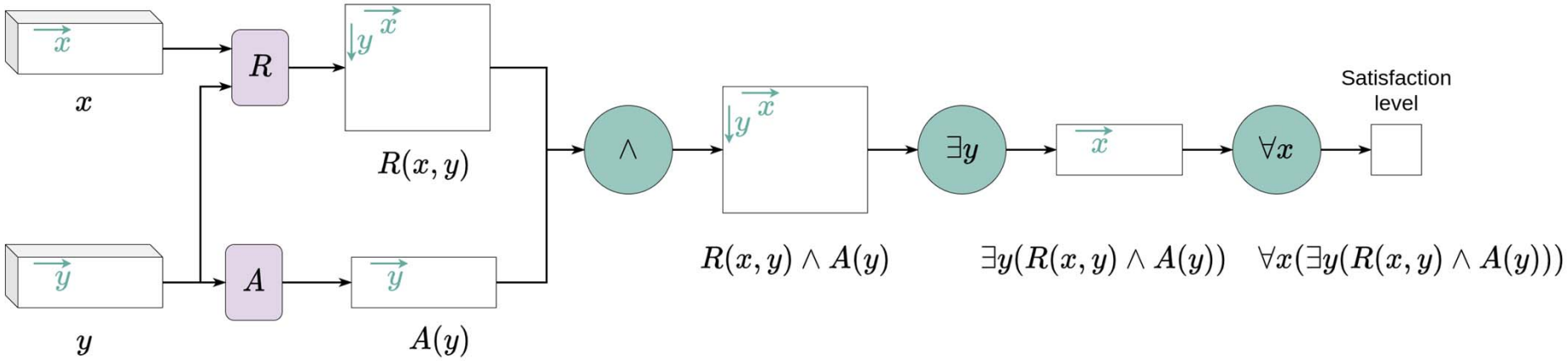
Quantifiers ($\forall, \exists$) are interpreted as aggregators

e.g. $\forall_{\text{mean}}(0.7, 0.2, \ldots) = \frac{1}{n}(0.7 + 0.2 + \ldots)$

Source: https://github.com/logictensornetworks

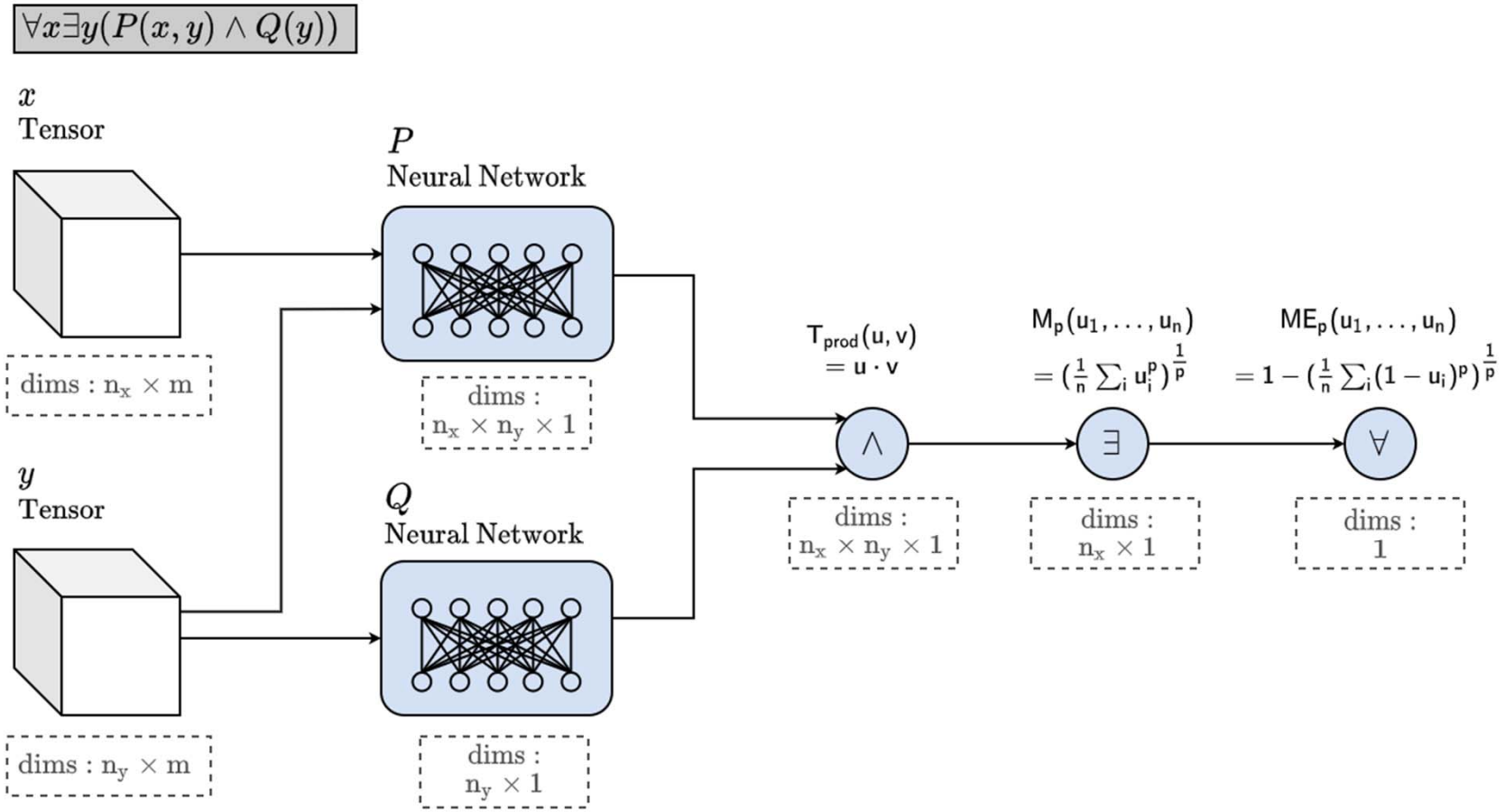
From Real Logic to Logic Tensor Networks

If R denotes the predicate $Friends$, and A denotes the predicate $Italian$, the following computational graph translates the sentence "*everybody has a friend who is Italian*":



Source: https://github.com/logictensornetworks

From Real Logic to Logic Tensor Networks



Source: https://github.com/logictensornetworks

From Real Logic to Logic Tensor Networks

- This is powerful because the NN can be used to learn the **membership** function for the corresponding **concept**.
- The underlying feature space can be based on features **extracted** from training data.
- In summary, the **symbolic-subsymbolic** connection is:
  - Subsymbolic: Weights in a neural network that classifies objects, parameters in regressors, etc.
  - Symbolic: information in rules such as “smoking causes cancer”.

From Real Logic to Logic Tensor Networks

Intuitions:

- When we observe a **new object**, we can use the NNs to classify it, and then reason using the formulas.
- Furthermore, rules can be leveraged for learning NN parameters; for instance:
  - First optimize the weights of the “smoker” network so that it correctly classifies individuals w.r.t. their smoker status.
  - But also take into account fuzzy formula *smokes(x)* → *cancer(x)* so that it is true for all points *x* in the feature space.
  - Rules thus provide additional **constraints**.

LTN: Tasks

In Real Logic, one can define the tasks of:

- **Learning:** The task of making generalizations from specific observations obtained from data (often called inductive inference)
- **Reasoning:** The task of deriving what knowledge follows from the facts that are currently known.
- **Query answering (QA):** The task of evaluating the truth value of a certain logical expression (query), or finding the set of objects in the data that evaluate a certain expression to *true*.

To discuss these tasks, we first need to discuss which types of **knowledge** can be represented in Real Logic.

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Representing Knowledge with Real Logic

- **Groundings** are an integral part of the knowledge represented by Real Logic.
- The **connection** between the symbols and the domain is represented explicitly in the language by a grounding $\mathcal{G}$
- An RL **knowledge base** is thus defined by formulas of the logical language and knowledge about the domain in the form of groundings obtained from data.
- There are several **types** of knowledge that can be represented in Real Logic.

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Knowledge through symbol groundings

Boundaries for domain grounding:

- Constraints specifying that the value of a certain logical expression must be within a certain range.
- For instance:
  - Elements of a domain “color” grounded onto points in $[0,1]^3$ encoding RGB values.
  - The range of a function:

    age(x) as integers between 0 and 100.

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Knowledge through symbol groundings

Explicit definition of grounding for **symbols**:

- Knowledge can be more strictly incorporated by fixing the grounding of some symbols.
- If a constant c denotes an object with known features $v_c \in \mathbb{R}^n$, we can fix its grounding $\mathcal{G}(c) = v_c$.
- For instance:
  - Each element in a training data set.
  - Predicate encoding similarity conditions grounded using a specific function (say, cosine distance).
  - Output layer of the NN associated with a multi-class single-label predicate $\mathrm{P}(\mathrm{x}, class)$ can be a softmax function normalizing the output such that it guarantees exclusive classification.

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Knowledge through symbol groundings

Parametric definition of grounding for symbols:

- Here, the exact grounding of a symbol σ is not known, but it is known that it can be obtained by finding a set of real-valued parameters (i.e., via learning).
- The typical example of parametric grounding for constants is the learning of an **embedding**.
- As an example, consider:
  - A neural network N trained for image classification into n classes: cat, dog, horse, etc.
  - N takes as input a vector v of pixel values and produces as output a vector $\mathbf{y} = (y_{cat}, y_{dog}, y_{horse}, \ldots) \in [0,1]^n$ such that $\mathbf{y} = N(\mathbf{v} \mid \theta_N)$, where y_c is the probability that input image $\mathbf{v}$ is of class c.

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Knowledge through Formulas

Factual propositions:

- Knowledge about the properties of **specific objects** in the domain is represented, as usual, by logical propositions:
  - Suppose that it is known that img_1 is a number eight, img_2 is a number nine, and img_n is a number two.
  - This can be represented by adding the following facts to the knowledge base: $nine(img_1)$, $eight(img_2)$, ..., $two(img_n)$.
- **Semi-supervision** can be specified naturally via propositions containing disjunctions: $eight(img_1) \vee nine(img_1)$, which states that img_1 is either an eight or a nine (or both).

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Knowledge through Formulas

Factual propositions (cont.):

- **Relational learning** can be achieved by logically relating multiple objects (defined as constants or variables, or even as more complex sequences of terms):

  $$nine(img_1) \rightarrow \neg nine(img_2)$$

  "if img_1 is a nine then img_2 is not a nine"

  $$nine(img) \rightarrow \neg eight(img)$$

  "if img is a nine then it is not an eight"

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Knowledge through Formulas

Generalized propositions:

- **General knowledge** about all or some of the objects of some domains can be specified in Real Logic by using first-order logic formulas with quantified variables.
- This type of knowledge allows one to specify **arbitrary** constraints on the groundings independently from the specific data available.
- For example, the formula could be used to encode the **soft constraint** that friends of smokers are normally smokers:

$$\forall xy\ ((smokes(x)\ \wedge friend(x,y))\ \rightarrow smokes(y))$$

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Knowledge through fuzzy semantics: Definition for operators

- The grounding of a formula φ depends on the operators **approximating** the connectives and quantifiers that appear in φ.
- Different operators give different interpretations of the **satisfaction** associated with the formula.
- For instance, the operator $A_{pME}(a_1, \dots, an)$ that approximates universal quantification can be seen as a **smooth minimum**:
  - It depends on a hyperparameter p (the exponent in the gen. mean).
  - If $p = 1$ then $A_{pME}(a_1, \dots, an)$ corresponds to the **arithmetic mean**.
  - As p increases, given the same input, the value of the universally quantified formula will decrease as A_{pME} converges to the **min operator**.

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Knowledge through fuzzy semantics: Definition for operators

- The grounding of a formula φ depends on the operators **approximating** the connectives and quantifiers that appear in φ.
- Different operators give different interpretations of the **satisfaction** associated with the formula.
- For instance, the operator $A_{pME}(a_1, \dots, an)$ that approximates universal quantification can be seen as a **smooth minimum**:(cont.):
  - To define how **strictly** the universal quantification should be interpreted in each proposition, one can use **different values** of p for different propositions of the knowledge base.
  - For instance, a formula $\forall x\ P(x)$ where A_{pME} is used with a low value for p will denote that P holds for **some** x, whereas a formula $\forall x\ Q(x)$ with a higher p may denote that Q holds for **most** x.

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

LTN Satisfiability

- A Real Logic theory $\mathcal{T} = (\mathcal{K}, \mathcal{G}(\,.\,|\,\theta), \Theta)$ has three components:
  - Knowledge about the **grounding of symbols** (domains, constants, variables, functions, and predicate symbols);
  - a set of closed logical **formulas** describing factual propositions and general knowledge;
  - **operators** and the **hyperparameters** used to evaluate each formula.
- Learning and reasoning in a Real Logic theory are both associated with searching for and applying the set of values of parameters θ from the hypothesis space Θ that **maximize** the **satisfaction** of the formulas in $\mathcal{K}$.

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

LTN Satisfiability

- We use the term **grounded theory**, denoted by $\langle \mathcal{K}, \mathcal{G}_\theta \rangle$, to refer to a Real Logic theory with a specific set of **learned** parameter values.
- To define this optimization problem, we aggregate the truth values of all the formulas in $\mathcal{K}$ by selecting a **formula aggregating operator**:

$$SatAgg: [0,1]^* \rightarrow [0,1]$$

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

LTN Learning

- Given a Real Logic theory $\mathcal{T} = (\mathcal{K}, \mathcal{G}(\,.\,|\,\theta), \Theta)$, **learning** is the process of searching for the set of parameter values θ^* that **maximize the satisfiability** of $\mathcal{T}$ w.r.t. a given aggregator:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmax}} \underset{\phi\in\mathcal{K}}{\operatorname{SatAgg}} \mathcal{G}_{\boldsymbol{\theta}}(\phi)$$

- With this general formulation, one can learn the grounding of constants, functions, and predicates:
  - Learning grounding of **constants** corresponds to learning of **embeddings**.
  - Learning grounding of **functions** corresponds to learning **generative models** or a **regression** task.
  - Learning of the grounding of **predicates** corresponds to a **classification** task.

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

LTN Querying

- Given a grounded theory, QA allows one to **check** if a certain **fact** is true (rather, by *how much* it is true).
- Various types of queries can be asked:
  - **Truth queries:** What is the truth value of a formula in the language? If the formula is closed, we get a scalar, if it has n free variables, we get a tensor of order n.
  - **Value queries:** What is the value of a term in the language? Analogous to truth queries.
  - **Generalization truth queries:** What is the truth value associated to a formula evaluated over unseen data?
  - **Generalization value queries:** Analogous

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

LTN Reasoning

- **Reasoning** is the task of verifying if a formula is a logical consequence of a set of formulas.
- A formula φ is a fuzzy **logical consequence** of a finite set of formulas Γ iff every model of Γ is a model of φ.
- In Real Logic, this is **generalized** by defining an interval $[q, 1]$ with $0.5 < q < 1$ and assuming that a formula is true iff its truth-value is in the interval $[q, 1]$.
- Unfortunately, this definition is only useful in theory since it requires inspecting potentially **infinite** sets of groundings.

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

LTN Approximate Reasoning

Option 1: Querying after learning

- Consider only the grounded theories that maximally satisfy the given theory.
- A (likely incomplete) set of such grounded theories can be found via **multiple optimization runs**.
- This is a kind of **brave reasoning**.

Option 2: Proof by refutation

- Search for a **counterexample** to the consequence.
- If no such example is found, the consequence is **assumed** to hold.
- The general formulation cannot be used as an objective function due to null derivatives – see paper for a soft constraint.

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Approx. Reasoning: Option 1 vs. 2 for $A \vee B \models A$?

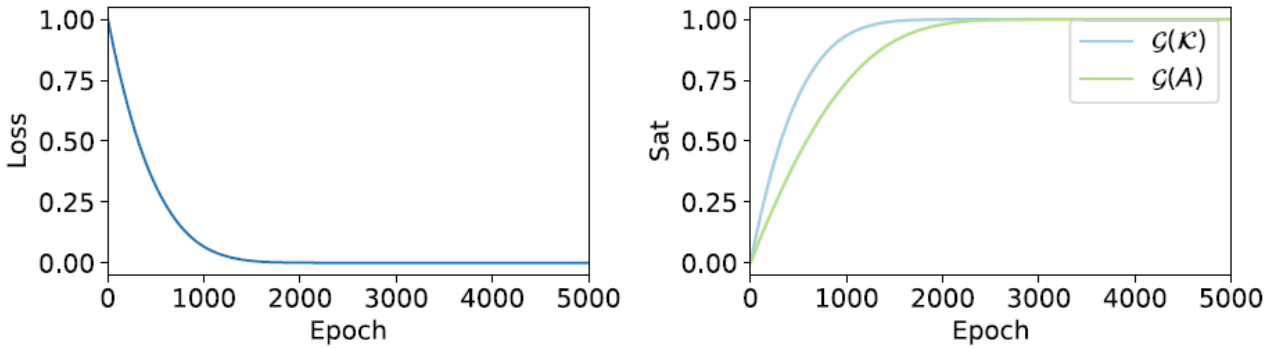


Fig. 22. Querying after learning: 10 runs of the optimizer with objective $\mathcal{G}^* = \text{argmax}_{\mathcal{G}_\theta}(\mathcal{G}_\theta(\mathcal{K}))$. All runs converge to the optimum $\mathcal{G}_1$; the grid search misses the counter-example.

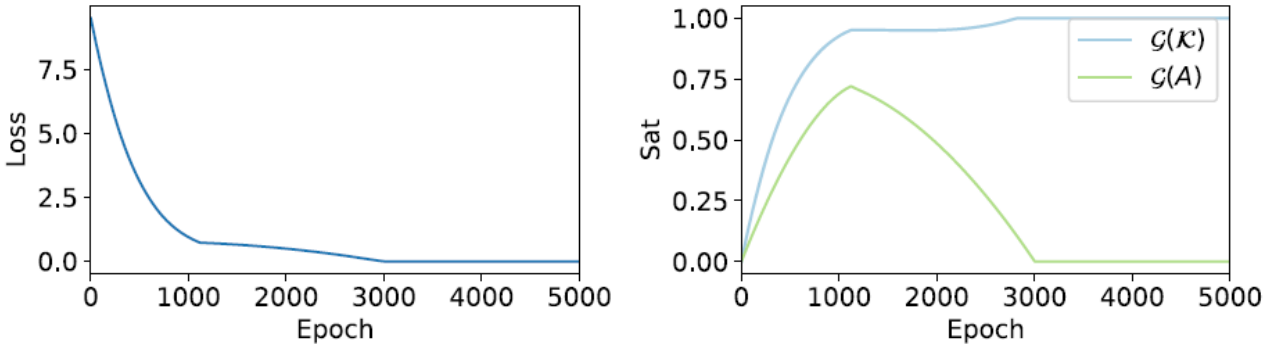


Fig. 23. Reasoning by refutation: one run of the optimizer with objective $\mathcal{G}^* = \text{argmin}_{\mathcal{G}_\theta}(\mathcal{G}_\theta(\phi) + \texttt{elu}(\alpha, \beta(q - \mathcal{G}_\theta(\mathcal{K}))))$, $q = 0.95$, $\alpha = 0.05$, $\beta = 10$. In the first training epochs, the directed search prioritizes the satisfaction of the knowledge base. Then, the minimization of $\mathcal{G}_\theta(\phi)$ starts to weigh in more and the search focuses on finding a counter-example. Eventually, the run converges to the optimum $\mathcal{G}_3$, which refutes the logical consequence.

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Logic Tensor Networks

Use Cases

Use Case: Complex queries to interpret data and models – *Multiclass multilabel classification*

Leptograpsus crabs dataset
- 5 morphological features
- Labeled as Male/Female, Blue/Orange

1. Define symbols

Variable x for the samples

Constants $Male, Female, Blue, Orange$ for the labels

Predicate $C(x, l)$ classifies points into labels using a trained Neural Network

FL: frontal lip of carapace (mm)
RW: rear width of carapace (mm)
CL: length along the midline of carapace (mm)
CW: maximum width of carapace (mm)
BD: body depth (mm)

2. Query the satisfaction level of some formulas

$$\forall x \, (C(x, Blue) \rightarrow \neg C(x, Orange))$$

e.g. satisfaction level of 0.92

$$\forall x \, (C(x, Blue) \rightarrow \neg C(x, Male))$$

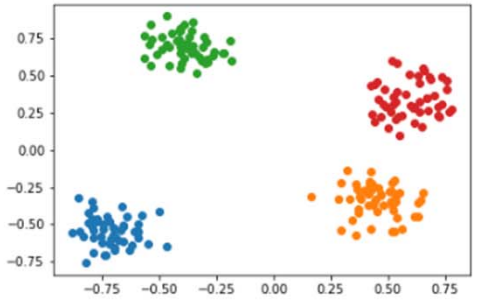
e.g. satisfaction level of 0.38

Source: https://github.com/logictensornetworks

Full example: https://nbviewer.org/github/logictensornetworks/logictensornetworks/blob/master/examples/multiclass_classification/multiclass-multilabel.ipynb

Data: https://www.stats.ox.ac.uk/pub/PRNN/

Use Case: Learning using satisfaction of symbolic rules as an objective – *Clustering*



1. Define symbols

Variable x for the points

Variable c for the cluster labels

Predicate $C(x, c)$ classifies points into clusters using a Neural Network; *untrained*

2. Write axioms

$$\forall x \, \exists c \, C(x, c)$$
$$\forall c \, \exists x \, C(x, c)$$
$$\forall (c, x, y : |x - y| < 0.2 \;\; (C(x, c) \leftrightarrow C(y, c)))$$
$$\forall (c, x, y : |x - y| > 1.0 \;\; \neg (C(x, c) \wedge C(y, c)))$$

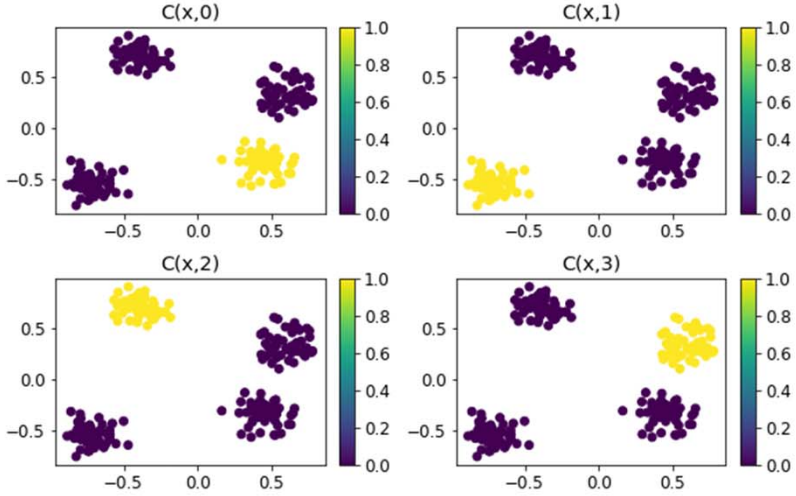
3. Train

Maximize $\mathrm{Agg}_\phi(\mathrm{sat}_{\phi_1}, \mathrm{sat}_{\phi_2}, \ldots)$,
using a "formula aggregator" (e.g. mean, p-mean, ...)

Backpropagate gradients to symbols with untrained groundings: here, the predicate C

```
Epoch 0: Sat Level 0.442
Epoch 100: Sat Level 0.442
Epoch 200: Sat Level 0.765
Epoch 300: Sat Level 0.851
Epoch 400: Sat Level 0.853
Epoch 500: Sat Level 0.853
Epoch 600: Sat Level 0.853
Epoch 700: Sat Level 0.854
Epoch 800: Sat Level 0.854
Epoch 900: Sat Level 0.854
Training finished at Epoch 999 with Sat Level 0.854
```

4. Query the results



Source: https://github.com/logictensornetworks

Full example: https://nbviewer.org/github/logictensornetworks/logictensornetworks/blob/master/examples/clustering/clustering.ipynb

Use Case: Smokers, Friends, Cancer

Let's consider the classic example introduced in the Markov Logic Networks paper (Domingos et al., 2006):

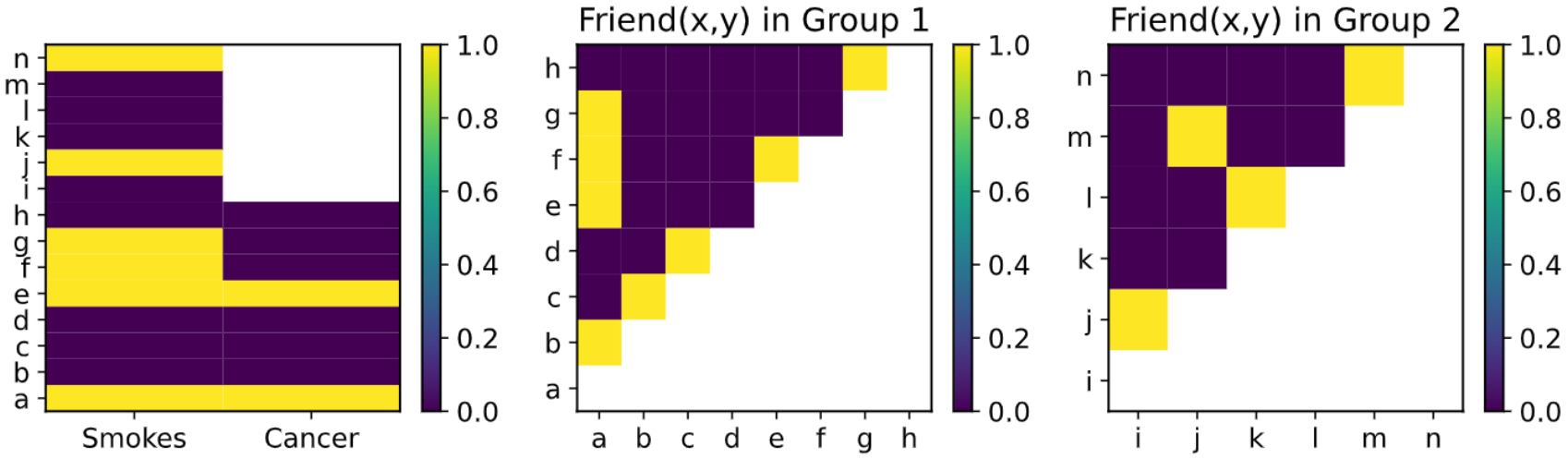
- 14 people divided into two groups: $\{a, b, \ldots, h\}$ and $\{i, j, \ldots, n\}$.
- Within each group, there **is complete knowledge** about **smoking** habits.
- In the first group, there is **complete knowledge** about who has and who does not have **cancer**.
- Knowledge about the **friendship** relation is complete within each group only if symmetry is assumed (i.e., $\forall x, y\ friends(x, y) \rightarrow friends(y, x)$). Otherwise, knowledge about friendship is **incomplete**.
- Finally, general knowledge about smoking, friendship, and cancer:
  - smoking causes cancer,
  - friendship is normally symmetric and anti-reflexive,
  - everyone has a friend, and
  - smoking propagates (actively or passively) among friends.

Source: https://github.com/logictensornetworks

Use Case: Smokers, Friends, Cancer

Language:

- LTN **constants** are used to denote the individuals. Each is grounded as a trainable embedding.
- Smokes, Friends, Cancer **predicates** are grounded as simple MLPs.
- All **rules + facts** are formulated in the knowledgebase.
- **Inconsistency:** for example, person f smokes, doesn't have cancer.
- **Incompleteness:** for example, inter-group friendship, cancer in group 2.

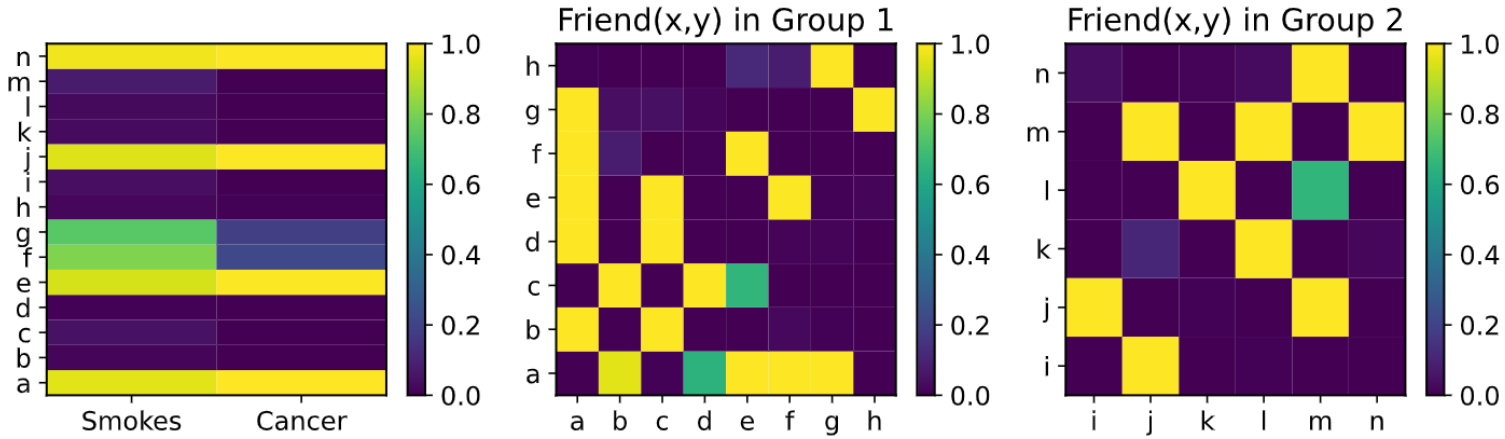


Source: https://github.com/logictensornetworks

Use Case: Smokers, Friends, Cancer

After training, we can:

- **Test satisfiability** of the axioms:



- **Issue queries** with new formulas:

  forall p: Cancer(p) -> Smokes(p): 0.96

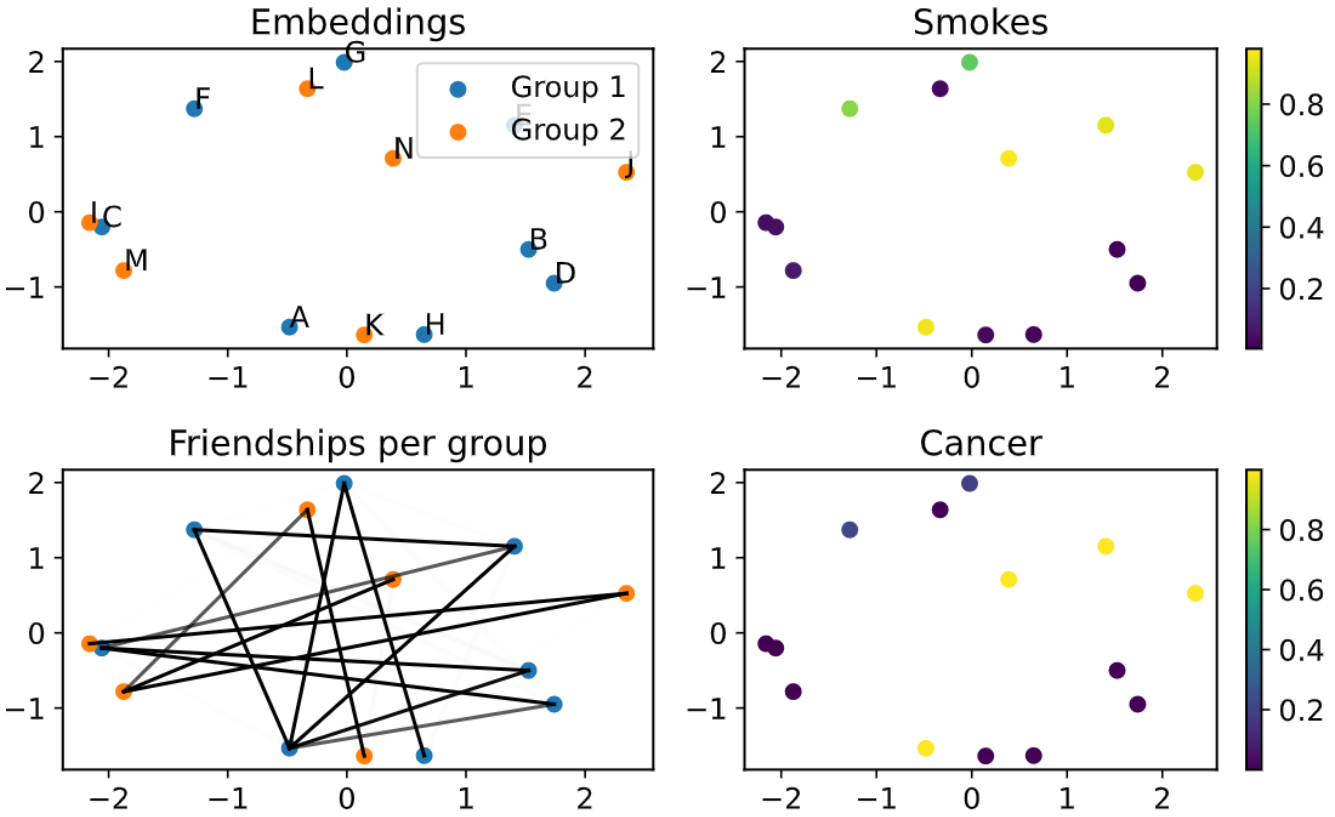
  forall p,q: (Cancer(p) or Cancer(q)) -> Friends(p,q): 0.22

Source: https://github.com/logictensornetworks

Use Case: Smokers, Friends, Cancer

After training, we can (cont.):

- **Visualize** the embeddings:



Source: https://github.com/logictensornetworks

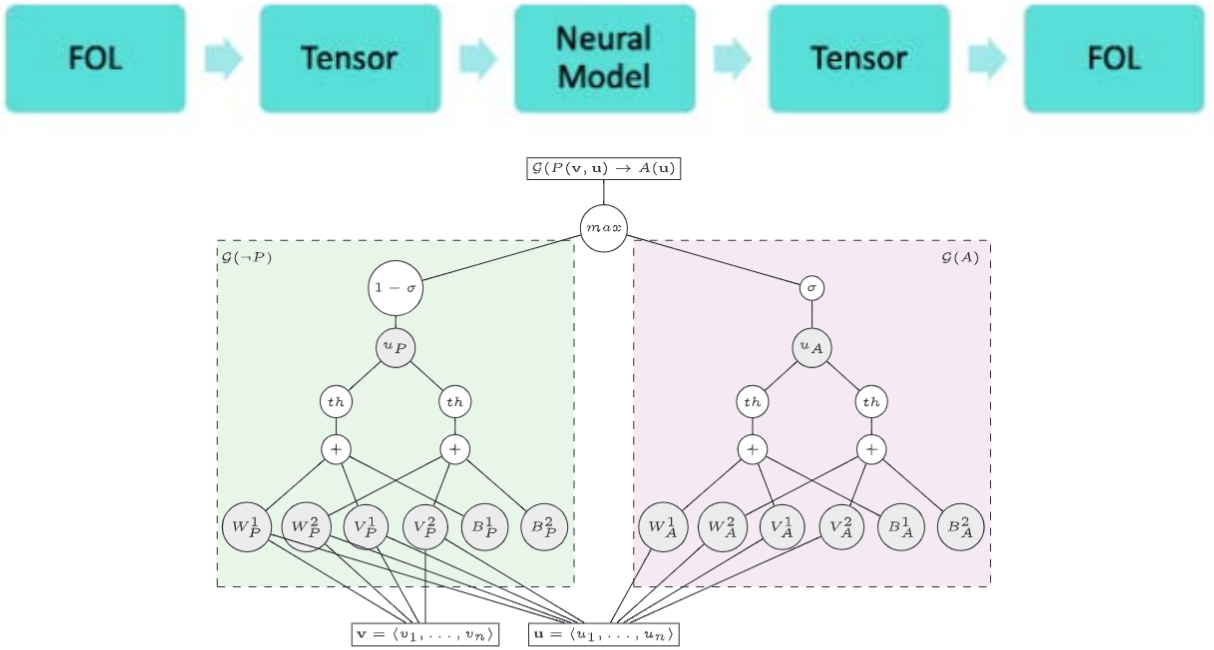
Logic Tensor Networks

Discussion

LTN within the Kautz Taxonomy

According to Kautz, LTN fall within the **Neuro_{Symbolic}** category:

- These architectures **transform** symbolic rules into **templates** for structures **within** the neural network:



Sources:

https://harshakokel.com/posts/neurosymbolic-systems/

Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

LTN within the Kautz Taxonomy

According to Kautz, LTN fall within the **Neuro_{Symbolic}** category:

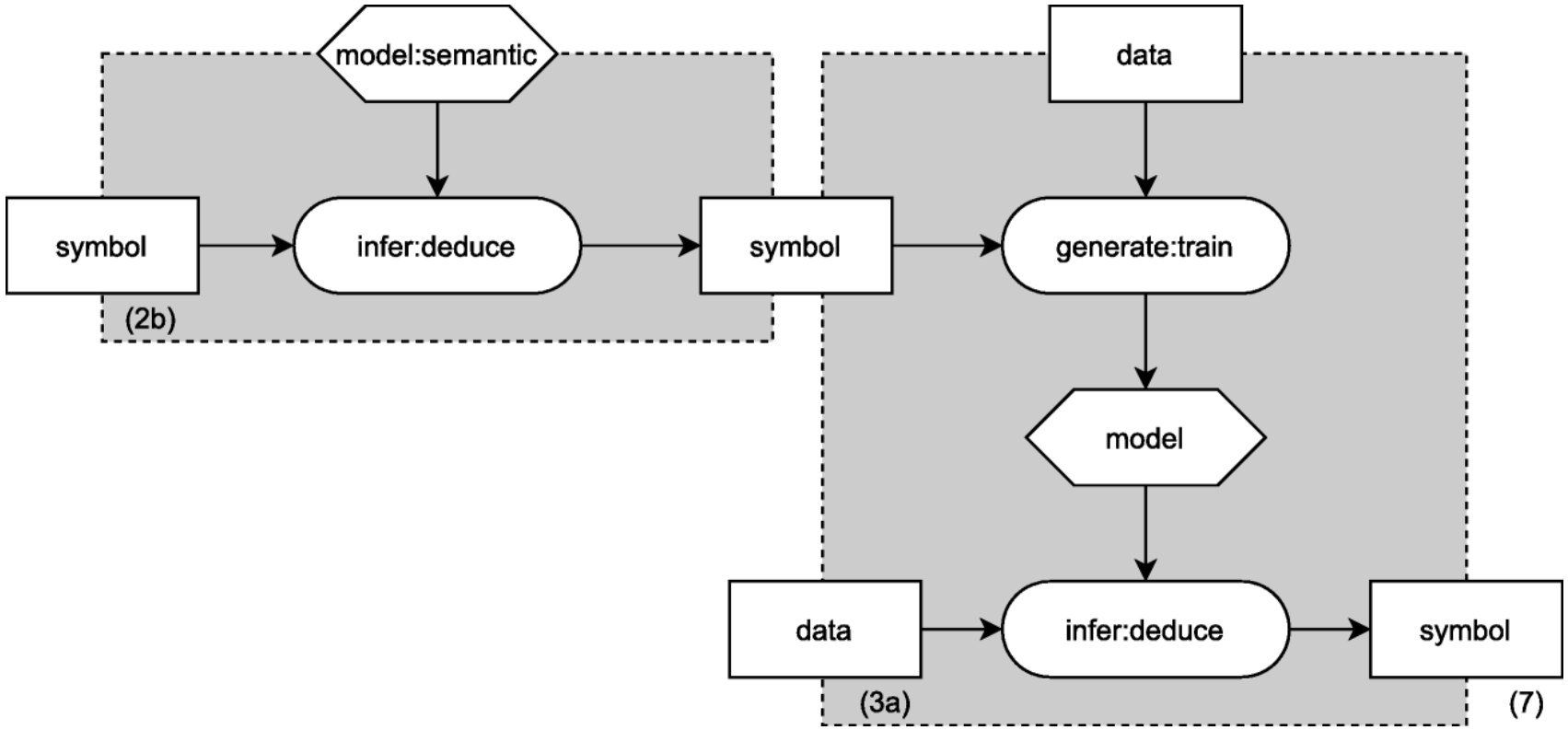
- These architectures **transform** symbolic rules into **templates** for structures **within** the neural network.
- How is this different from category 1 (Symbolic Neuro symbolic)?
  - Only the objects (individuals) are tensorized
  - It's the standard way in which deep learning approaches work: after processing input tensors, output tensors are produced and transformed back into symbols.
  - LTN tensorizes all logical components: objects, predicates, functions, and formulas.

Sources:

https://harshakokel.com/posts/neurosymbolic-systems/

Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

LTN as “Informed learning with prior knowledge”



- LTN jointly min. NN loss functions and max. FOL theory satisfaction.
- This can be regarded as a **semantic loss function**.

Source: van Bekkum, M., de Boer, M., van Harmelen, F. et al. Modular design patterns for hybrid learning and reasoning systems. Appl Intell 51, 6528–6546 (2021).

Logic Tensor Networks

Differentiable Fuzzy Logic

Differentiable Fuzzy Logic

- In presenting Real Logic and LTN, we did not give much thought to the role of **fuzzy operators** in the main **tasks**.
- Gradient descent requires that operators be **differentiable** so that it can smoothly traverse the universe of values.
- Three types of **gradient problems** commonly arise:
  - **Single-Passing:** The derivatives of some operators are non-null for only one argument. The gradients propagate to only one input at a time.
  - **Vanishing Gradients:** Gradients vanish on some part of the domain. Learning does not update inputs that are in the vanishing domain.
  - **Exploding Gradients:** Large error gradients accumulate and result in unstable updates.

Sources:

van Krieken et al. (2022). Analyzing differentiable fuzzy logic operators. Artif. Intell., 302, 103602

Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

https://github.com/logictensornetworks

Problems: Single-Passing Gradients

Some operators have gradients propagating to only one input at a time, meaning that all other inputs will not benefit from learning at this step.

e.g. in $\min(u_1, \ldots, u_n)$.

```
xs = tf.constant([1.,1.,1.,0.5,0.3,0.2,0.2,0.1])

with tf.GradientTape() as tape:
    tape.watch(xs)
    y = forall_min(xs)
res = y.numpy()
gradients = tape.gradient(y,xs).numpy()
print(res)
print(gradients)
```

```
0.1
[0. 0. 0. 0. 0. 0. 0. 1.]
```

Source: https://github.com/logictensornetworks

Problems: Vanishing Gradients

Some operators have vanishing gradients on some part of their domains.

e.g. in $u \wedge_{\text{luk}} v = \max(u + v - 1, 0)$, if $u + v - 1 < 0$, the gradients vanish.

```
x1 = tf.constant(0.3)
x2 = tf.constant(0.5)

with tf.GradientTape() as tape:
    tape.watch(x1)
    tape.watch(x2)
    y = and_luk(x1,x2)
res = y.numpy()
gradients = [v.numpy() for v in tape.gradient(y,[x1,x2])]
print(res)
print(gradients)
```

```
0.0
[0.0, 0.0]
```

Source: https://github.com/logictensornetworks

Problems: Exploding Gradients

Some operators have exploding gradients on some part of their domains.

e.g. in $\mathrm{pME}(u_1, \ldots, u_n) = 1 - \left(\frac{1}{n} \sum_{i=1}^{n} (1 - u_i)^p \right)^{\frac{1}{p}}$, on the edge case where all inputs are 1.0.

```
xs = tf.constant([1.,1.,1.])

with tf.GradientTape() as tape:
    tape.watch(xs)
    y = forall_pME(xs,p=4)
res = y.numpy()
gradients = tape.gradient(y,xs).numpy()
print(res)
print(gradients)
```

```
1.0
[nan nan nan]
```

Source: https://github.com/logictensornetworks

Gradient Problems: Binary Connectives

Table C.7
Gradient problems for some binary connectives. (✗) means that the problem only appears on an edge case.

	Single-Passing	Vanishing	Exploding
Goedel (minimum)			
T_M, S_M	✗		
I_{KD}	✗		
I_G	✗	✗	
Goguen (product)			
T_P, S_P		(✗)	
I_R		(✗)	
I_{KD}		✗	(✗)
Łukasiewicz			
T_L, S_L		✗	
I_{Luk}		✗	

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Gradient Problems: Aggregators

Table C.8
Gradient problems for some aggregators. (✗) means that the problem only appears on an edge case.

	Single-Passing	Vanishing	Exploding
A_{T_M}/A_{S_M}	✗		
A_{T_P}/A_{S_P}		✗	
A_{T_L}/A_{S_L}		✗	
A_{pM}			(✗)
A_{pME}			(✗)

Source: Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

Stable Configuration of Operators

The following is proposed as a stable configuration by Badreddine et al. (2022):

- not: the standard negation $\neg u = 1 - u$,
- and: the product t-norm $u \wedge v = uv$,
- or: the product t-conorm (probabilistic sum) $u \vee v = u + v - uv$,
- implication: the Reichenbach implication $u \to v = 1 - u + uv$,
- existential quantification ("exists"): the generalized mean (p-mean)

$$\mathrm{pM}(u_1, \ldots, u_n) = \left(\frac{1}{n} \sum_{i=1}^{n} u_i^p \right)^{\frac{1}{p}} \qquad p \geq 1,$$

- universal quantification ("for all"): the generalized mean of "the deviations w.r.t. the truth" (p-mean error)

$$\mathrm{pME}(u_1, \ldots, u_n) = 1 - \left(\frac{1}{n} \sum_{i=1}^{n} (1 - u_i)^p \right)^{\frac{1}{p}} \qquad p \geq 1.$$

Sources:

Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

https://github.com/logictensornetworks

Stable Configuration of Operators

Some caveats:

- The product t-norm has vanishing gradients on the edge case $u = v = 0$.
- The product t-conorm has vanishing gradients on the edge case $u = v = 1$.
- The Reichenbach implication has vanishing gradients on the edge case $u = 0, v = 1$.
- p-mean has exploding gradients on the edge case $u_1 = \cdots = un = 0$.
- p-mean error has exploding gradients on the edge case $u_1 = \cdots = un = 1$.

These issues happen on **edge cases** and can be fixed using the following "trick":

- if the edge case happens when an input u is 0, we modify every input with
$$u' = (1 - \epsilon)u + \epsilon$$
- if the edge case happens when an input u is 1, we modify every input with
$$u' = (1 - \epsilon)u$$

  where ϵ is a small positive value (e.g., $1\mathrm{e}{-5}$).

Sources:

Badreddine, S., Garcez, A. D. A., Serafini, L., & Spranger, M. (2022). Logic tensor networks. Artif. Intelli., 303, 103649.

https://github.com/logictensornetworks/LTNtorch/blob/main/tutorials/2b-operators-and-gradients.ipynb


ASU
Ira A. Fulton Schools of
Engineering
Arizona State University