On the relation between Loss Functions and T-Norms*

Francesco Giannini[1], Giuseppe Marra[1,2], Michelangelo Diligenti[1], Marco Maggini[1], and Marco Gori[1]

[1] Department of Information Engineering and Mathematical Sciences, University of Siena, ITALY
{fgiannini,diligmic,maggini,marco}@diism.unisi.it
[2] Department of Information Engineering, University of Florence, ITALY
g.marra@unifi.it

Abstract. Deep learning has been shown to achieve impressive results in several domains like computer vision and natural language processing. A key element of this success has been the development of new loss functions, like the popular cross-entropy loss, which has been shown to provide faster convergence and to reduce the vanishing gradient problem in very deep structures. While the cross-entropy loss is usually justified from a probabilistic perspective, this paper shows an alternative and more direct interpretation of this loss in terms of t-norms and their associated generator functions, and derives a general relation between loss functions and t-norms. In particular, the presented work shows intriguing results leading to the development of a novel class of loss functions. These losses can be exploited in any supervised learning task and which could lead to faster convergence rates that the commonly employed cross-entropy loss.

Keywords: Loss functions · Learning from constraints · T-Norms.

1 Introduction

A careful choice of the loss function has been pivotal into the success of deep learning. In particular, the **cross-entropy loss**, or log loss, measures the performance of a classifier and increases when the predicted probability of an assignment diverges from the actual label [7]. In supervised learning, the cross-entropy loss has a clear interpretation as it attempts at minimizing the distribution of the predicted and given pattern labels. From a practical standpoint, the main advantage of this loss is to limit the vanishing gradient issue for networks with sigmoidal or softmax output activations.

Recent advancements in Statistical Relational Learning (SRL) [16] allow to inject prior knowledge, often expressed using a logic formalism, into a learner.

* This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 825619.

One of the most popular lines of research in this community attempts at defining frameworks for performing logic inference in the presence of uncertainty. For example, Markov Logic Networks [18] and Probabilistic Soft Logic [1] integrate First Order Logic (FOL) and graphical models. More recently, many attempts have been focusing on integrating reasoning with uncertainty with deep learning [20]. A common solution, followed by approaches like Semantic Based Regularization [4] and Logic Tensor Networks [5], relies on using deep networks to approximate the FOL predicates, and the overall architecture is optimized end-to-end by relaxing the FOL into a differentiable form, which translates into a set of constraints. For the sake of overall consistency, one question that can naturally arise in this context is how the fitting of the supervised examples can be expressed using logic formalism. Following this starting point, this paper follows an orthogonal approach for the definition of a loss function, by studying the relation between the translation of the prior knowledge using t-norms and the resulting loss function. In particular, the notion of t-norm *generator* plays a fundamental role in the behavior of the corresponding loss. Remarkably, the cross-entropy loss can be naturally derived within this framework. However, the presented theoretical results suggest that there is a larger class of loss functions that correspond to the different possible translations of logic using t-norms, and some loss functions are potentially more effective than the cross-entropy to limit the vanishing gradient issue, therefore proving a faster convergence rate.

The paper is organized as follows: Section 2 presents the basic concepts about t-norms, generators and aggregator functions. Section 3 introduces the learning frameworks used to represent supervised learning in terms of logic rules, while Section 4 presents the experimental results and, finally, Section 5 draws some conclusions.

2 Fuzzy Aggregation Functions

The aggregation takes place on a set of values typically representing preferences or satisfaction degrees restricted to the unit interval $[0, 1]$ to be aggregated. There are several ways to aggregate them into a single value expressing an overall combined score, according to what is expected from such mappings. The purpose of aggregation functions is to combine inputs that are typically interpreted as degrees of membership in fuzzy sets, degrees of preference or strength of evidence. Aggregation functions have been studied by several authors in the literature [2, 3], and they are successfully used in many practical applications, for instance see [8, 19]. Please note that the fuzzy aggregation functions that will be covered in this section can be directly applied to the output of a multi-task classifier, when implemented via a neural network with sigmoidal or softmax output units.

Basic Definitions. Aggregation functions are defined for inputs of any cardinality, however for simplicity the main definitions are provided only for the binary case. A (binary) aggregation function is a non-decreasing function $A : [0, 1]^2 \to [0, 1]$, such that: $A(0, 0) = 0$, $A(1, 1) = 1$. An aggregation function

Gödel	Lukasiewicz	Product
$T_M(x,y) = \min\{x,y\}$	$T_L(x,y) = \max\{0, x+y-1\}$	$T_\Pi(x,y) = x \cdot y$

Table 1: Fundamental t-norms.

A can be categorized according to the pointwise order in Equation 1 as: *conjunctive* when $A \leq \min$, *disjunctive* when $\max \leq A$, *averaging* (a *mean*) when $\min < A < \max$ and *hybrid* otherwise; where min and max are the aggregation functions for the *minimum* and *maximum* respectively.

$$A_1 \leq A_2 \quad \text{iff} \quad A_1(x,y) \leq A_2(x,y), \text{ for all } x,y \in [0,1] \ . \tag{1}$$

Conjunctive and disjunctive type functions combine values as if they were related by a logical AND and OR operations, respectively. On the other hand, averaging type functions have the property that low values can be compensated by high values. Mean computation is the most common way to combine the inputs, since it assumed the total score cannot be above or below any of the inputs, but it depends on all the inputs.

2.1 Archimedean T-Norms

Despite averaging functions have nice properties to aggregate fuzzy values, they are not suitable to represent neither a conjunction nor a disjunction, because they do not generalize their boolean counterpart. This is a reason why, we focus on t-norms and t-conorms [11, 14], that are *associative, commutative* aggregation functions with 1 and 0 as *neutral element*, respectively. Table 1 reports Gödel, Lukasiewicz and Product t-norms, which are referred as the fundamental t-norms because all the continuous t-norms can be obtained as ordinal sums of the two fundamental t-norms [10]. A simple example of a t-norm that is not continuous is given by the Drastic t-norm T_D, that is always returning a zero value, except for $T_D(1,1) = 1$. *Archimedean* t-norms [13] are a class of t-norms that can be constructed by means of unary monotone functions, called *generators*.

Definition 1. *A t-norm T is said to be* Archimedean *if for every $x \in (0,1)$, $T(x,x) < x$. In addition, T is said* strict *if for all $x \in (0,1)$, $0 < T(x,x) < x$ otherwise is said* nilpotent.

For instance, the Lukasiewicz t-norm T_L is nilpotent, the Product t-norm T_Π is strict, while the Gödel one T_M is not archimedean, indeed $T_M(x,x) = x$, for all $x \in [0,1]$. The Lukasiewicz and Product t-norms are enough to represent the whole classes of nilpotent and strict Archimedean t-norms [14].

A fundamental result for the construction of t-norms by *additive* generators is based on the following theorem [12]:

Theorem 1. *Let $g : [0,1] \to [0,+\infty]$ be a strictly decreasing function with $g(1) = 0$ and $g(x) + g(y) \in Range(g) \cup [g(0^+), +\infty]$ for all x, y in $[0,1]$, and*

$g^{(-1)}$ *its pseudo-inverse. Then the function* $T : [0,1] \to [0,1]$ *defined as*

$$T(x,y) = g^{-1}\left(\min\{g(0^+), g(x) + g(y)\}\right) \ . \tag{2}$$

is a t-norm and g *is said an* additive generator *for* T*.*

Any t-norm T with an additive generator g is Archimedean, if g is continuous then T is continuous, T is strict if and only if $g(0) = +\infty$, otherwise it is nilpotent.

Example 1. If we take $g(x) = 1 - x$, then also $g^{-1}(y) = 1 - y$ and we get T_L:

$$T(x,y) = 1 - \min\{1, 1 - x + 1 - y\} = \max\{0, x + y - 1\} \ .$$

Example 2. Taking $g(x) = -\log(x)$, we have $g^{-1}(y) = e^{-y}$ and we get T_Π:

$$T(x,y) = e^{-(\min\{+\infty, -\log(x) - \log(y)\})} = x \cdot y \ .$$

Eq. (2) allows to derive the other fuzzy connectives as function of the generator:

$$\begin{aligned} \text{residuum}: \quad & x \Rightarrow y = g^{-1}\left(\max\{0, g(y) - g(x)\}\right) \\ \text{bi-residuum}: \quad & x \Leftrightarrow y = g^{-1}\left(|g(x) - g(y)|\right) \end{aligned} \tag{3}$$

If g is expressed as a parametric function, it is possible to define families of t-norms, which can be constructed by the generator obtained when setting the parameters to specific values. Several parametric families of t-norms have been introduced [2]. The experimental section of this paper employs the family of Schweizer–Sklar and Frank t-norms, depending on a parameter $\lambda \in (-\infty, +\infty)$ and $\lambda \in [0, +\infty]$ respectively, and whose generators are defined as:

$$g_\lambda^{SS}(x) = \begin{cases} -\log(x) & \text{if } \lambda = 0 \\ \frac{1-x^\lambda}{\lambda} & \text{otherwise} \end{cases} \quad \text{and} \quad g_\lambda^{F}(x) = \begin{cases} -\log(x) & \text{if } \lambda = 1 \\ 1 - x & \text{if } \lambda = +\infty \\ \log\left(\frac{\lambda - 1}{\lambda^x - 1}\right) & \text{otherwise} \end{cases} \tag{4}$$

3 From Formulas to Loss Functions

A learning process can be thought of as a constraint satisfaction problem, where the constraints represent the knowledge about the functions to be learned. In particular, multi-task learning can be expressed via a set of constraints expressing the fitting of the supervised examples, plus any additional abstract knowledge.

Let us consider a set of unknown task functions $\mathbf{P} = \{p_1, \ldots, p_J\}$ defined on $\mathbb{R}^n$, all collected in the vector $\boldsymbol{p} = (p_1, \ldots, p_J)$ and a set of known functions or predicates $\boldsymbol{S}$. Given the set $\mathcal{X} \subseteq \mathbb{R}^n$ of available data, a learning problem can be generally formulated as $\min_{\boldsymbol{p}} \mathcal{L}(\mathcal{X}, \boldsymbol{S}, \boldsymbol{p})$ where $\mathcal{L}$ is a positive-valued functional denoting a certain loss function. Each predicate is approximated by a neural network providing an output value in $[0,1]$. The available knowledge about the task functions consists in a set of FOL formulas $KB = \{\varphi_1, \ldots, \varphi_H\}$ and the

learning process aims at finding a good approximation of each unknown element, so that the estimated values will satisfy the formulas for the input samples. Since any formula is true if it evaluates to 1, in order to satisfy the constraints we may minimize the following loss function:

$$\mathcal{L}(\mathcal{X}, \boldsymbol{S}, \boldsymbol{p}) = \sum_{h=1}^{H} \lambda_h L\big(f_h(\mathcal{X}, \boldsymbol{S}, \boldsymbol{p})\big) \tag{5}$$

where any λ_h is the weight for the h-th logical constraint, which can be selected via cross-validation or jointly learned [15, 21], f_h is the truth-function corresponding to the formula φ_h according to a certain t-norm fuzzy logic and L is a decreasing function denoting the penalty associated to the distance from satisfaction of formulas, so that $L(1) = 0$. In the following, we will study different forms for the L cost function and how it depends on the choice of the t-norm generator. In particular, a *t-norm fuzzy logic* generalizes Boolean logic to variables assuming values in $[0, 1]$ and is defined by its t-norm modeling the logical AND [9]. The connectives can be treated using the fuzzy generalization of first–order logic that was first proposed by Novak [17]. The *universal* and *existential quantifiers* occurring in the formulas in KB allows the aggregation of different evaluations (groundings) of the formulas on the available data. For instance, given a formula $\varphi(x_i)$ depending on a certain variable $x_i \in \mathcal{X}_i$, where $\mathcal{X}_i$ denotes the available samples for the i-th argument of one of the involved predicates in φ, we may convert the quantifiers as the minimum and maximum operations that are common to any t-norm fuzzy logic:

$$\begin{aligned} \forall x_i\, \varphi(x_i) \implies \quad & f_\varphi(X_i, \boldsymbol{S}, \boldsymbol{p}) = \min_{x_i \in \mathcal{X}_i} f_\varphi(x_i, \boldsymbol{S}, \boldsymbol{p}) \\ \exists x_i\, \varphi(x_i) \implies \quad & f_\varphi(X_i, \boldsymbol{S}, \boldsymbol{p}) = \max_{x_i \in \mathcal{X}_i} f_\varphi(x_i, \boldsymbol{S}, \boldsymbol{p}) \end{aligned}$$

3.1 Loss Functions by T-Norms Generators

A quantifier can be seen as a way to aggregate all the possible groundings of a predicate variable that, in turn, are $[0, 1]$-values. Different aggregation functions have also been considered, for example in [5], the authors consider a mean operator to convert the universal quantifier. However this has the drawback that also the existential quantifier has the same semantics conversion and then it is determined by the authors via Skolemization. Even if this choice may yield some learning benefits, it has no direct justification inside a logic theory. Moreover it does not suggest how to map the functional translation of the formula into a constraint. In the following, we investigate the mapping of formulas into constraints by means of generated t-norm fuzzy logics, and we exploited the same additive generator of the t-norm to map the formula into the functional constraints to be minimized, i.e. $L = g$.

Given a certain formula $\varphi(x)$ depending on a variable x that ranges in the set $\mathcal{X}$ and its corresponding functional representation $f_\varphi(x, \boldsymbol{p})$ evaluated on each

$x \in \mathcal{X}$, the conversion of universal and existential quantifiers should have semantics equivalent to the AND and OR of the evaluation of the formula over the groundings, respectively. This can be realized by directly applying the t-norm or t-conorms over the groundings. For instance, for the universal quantifier:

$$\forall x\, \varphi(x) \equiv \bigwedge_x \varphi(x) \quad \Longrightarrow \quad g^{-1}\left(\min\left\{g(0^+), \sum_{x\in\mathcal{X}} g\big(f_\varphi(x, \boldsymbol{S}, \boldsymbol{p})\big)\right\}\right) \ , \tag{6}$$

where g is an additive generator of the t-norm T corresponding to the universal quantifier. Since any generator function is decreasing, in order to maximize the satisfaction of $\forall x\, \varphi(x)$ we can minimize g applied to Equation 6, namely:

$$\min\{g(0^+), \sum_{x\in\mathcal{X}} g(f_\varphi(x, \boldsymbol{S}, \boldsymbol{p}))\} \qquad \text{if } T \text{ is nilpotent} \tag{7}$$

$$\sum_{x\in\mathcal{X}} g(f_\varphi(x, \boldsymbol{S}, \boldsymbol{p})) \qquad \text{if } T \text{ is strict} \tag{8}$$

As a consequence, with respect to the convexity of the expressions in Equations 7-8, we get the following result, that is an immediate consequence of how the convexity is preserved by function composition.

Proposition 1. *If g is a linear function and f_φ is concave, Equation 7 is convex. If g is a convex function and f_φ is linear, Equation 8 is convex.*

Example 3. If $g(x) = 1 - x$ (Lukasiewicz t-norm) from Equation 7 we get:

$$\min(1, \sum_{x\in\mathcal{X}} (1 - (f_\varphi(x, \boldsymbol{S}, \boldsymbol{p}))) \ .$$

Hence, in case f_φ is concave (see [6] for a characterization of the concave fragment of Lukasiewicz logic), this function is convex.
If $g = -\log$ (Product t-norm) from Equation 8 we get the cross-entropy:

$$-\sum_{x\in\mathcal{X}} \log(f_\varphi(x, \boldsymbol{S}, \boldsymbol{p})) \ .$$

As we already pointed out in Section 2, if g is an additive generator for a t-norm T, then the residual implication and the biresidum with respect to T are given by Equation 3. In particular, if p_1, p_2 are two unary predicates functions sharing the same input domain $\mathcal{X}$, and $\boldsymbol{S} = \emptyset$ the following formulas yield the following penalty terms:

$$\forall x\, p_1(x) \Longrightarrow \min\left\{g(0^+), \sum_{x\in\mathcal{X}} g(p_1(x))\right\}$$

$$\forall x\, p_1(x) \Rightarrow p_2(x) \Longrightarrow \min\left\{g(0^+), \sum_{x\in\mathcal{X}} \max(0, g(p_2(x)) - g(p_1(x))\right\}$$

$$\forall x\, p_1(x) \Leftrightarrow p_2(x) \Longrightarrow \min\left\{g(0^+), \sum_{x\in\mathcal{X}} |g(p_1(x)) - g(p_2(x))|\right\} \ .$$

3.2 Redefinition of supervised Learning with Logic

In this section, we study the case of supervised learning w.r.t. the choice of a certain additive generator. Let us consider a multi-task classification problem with predicates $p_j, j = 1, \ldots, J$ defined over the same input domain with a supervised training set $\mathcal{T} = \{(x_i, y_i)\}$ where each $y_i \in \{1, 2, \ldots, J\}$ is the output class for the pattern x_i and $\mathcal{X}$ is the overall set of supervised patterns. Finally, the known predicate S_j is defined for each predicate such that $S_j(x_i) = 1$ iff $y_i = j$, and we indicate as $\mathcal{X}_j = \{x_i \in \mathcal{X} : S_j(x_i) = 1\}$ the set of positive examples for the j-th predicate. Then, we can enforce the supervision constraints for p_j as:

$$\forall x \, S_j(x) \Leftrightarrow p_j(x) \quad \Longrightarrow \quad \mathcal{L}(\mathcal{X}, \boldsymbol{S}, p_j) = \sum_{x \in \mathcal{X}} |g(S_j(x)) - g(p_j(x))|$$

In the special case of the predicates implemented by neural networks and exclusive multi-task classification, where each pattern should be assigned to one and only one class, the exclusivity can be enforced using a softmax output activation. Typically, in this scenario, only the positive supervisions are explicitly listed, and since it holds that $g(S_j(x)) = 0, \forall x \in \mathcal{X}_j$, yields:

$$\mathcal{L}^+(\mathcal{X}, \boldsymbol{S}, p_j) = \sum_{x \in \mathcal{X}_j} g(p_j(x)), \tag{9}$$

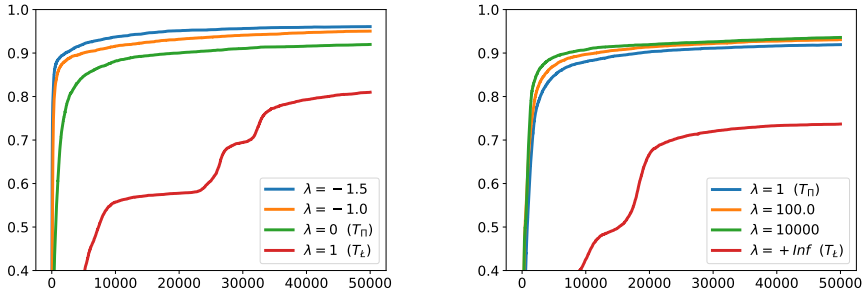
For instance, in the case of Lukasiewicz and Product logic, we have, respectively:

$$\mathcal{L}^+_L(\mathcal{X}_j, p_j) = \sum_{x \in \mathcal{X}_j} (1 - p_j(x)), \quad \mathcal{L}^+_\Pi(\mathcal{X}_j, p_j) = -\sum_{x \in \mathcal{X}_j} \log(p_j(x))$$

corresponding to the L_1 and cross entropy losses, respectively.

4 Experimental Results

The proposed framework allows to recover well-known loss functions by expressing the fitting of the supervision using logic and then carefully selecting the t-norm used to translate the resulting formulas. However, a main strength of the proposed theory is that it becomes possible to derive new principled losses starting from any family of parametric t-norms. Driven by the huge impact that cross-entropy gained w.r.t. to classical loss functions in improving convergence speed and generalization capabilities, we designed a set of experiments to investigate how the choice of a t-norm can lead to a loss function with better performances than the cross-entropy loss. The Schweizer–Sklar and the Frank parametric t-norms defined in Section 2.1 have been selected for this experimental evaluation, given the large spectrum of t-norms that can be generated by varying their λ parameter. The well known MNIST dataset is used as benchmark for all the presented experiments. In order to have a fair comparison, the same neural network architecture is used during all the runs: a 1-hidden layer



(a) The Schweizer–Sklar t-norms

(b) The Frank t-norms

Fig. 1: Convergence speed of multiple generated loss functions on the MNIST classification task for different values of the parameter λ of equation 4. The well-known cross-entropy loss is equivalent to the loss obtained by the T_{Π} generator.

neural network with 50 hidden ReLU units and 10 softmax output units. The softmax activation function allows to express only positive supervisions, like commonly done in mutually exclusive classification using the cross-entropy loss. Optimization is carried on using Vanilla gradient descent with a fixed learning rate of 0.01.

Results are shown in Figure 1, that reports the accuracy on the test set of a neural network trained on the MNIST dataset. Specific choices of the parameter λ recover classical loss functions, like the cross-entropy loss, which is equivalent to the loss obtained using T_{Π}. The results confirm that the cross-entropy loss converges faster than the L_1 obtained when using T_L. However, there is a wide range of possible choices for the parameter λ that brings an even faster convergence and better generalization than the widely adopted used cross-entropy.

5 Conclusions

This paper presents a framework to embed prior knowledge expressed as logic statements into a learning task, showning how the choice of the t-norm used to convert the logic into a differentiable form defines the resulting loss function used during learning. When restricting the attention to supervised learning, the framework recovers popular loss functions like the cross-entropy loss, and allows to define new loss functions corresponding to the choice of the parameters of t-norm parametric forms. The experimental results show that some newly defined losses provide a faster convergence rate that the commonly used cross-entropy loss. Future work will focus on testing the loss functions in more structured learning tasks, like the one commonly addressed with Logic Tensor Networks and Semantic based Regularization. The parametric form of the loss functions allows to define joint learning tasks, where the loss parameters are co-optimized during learning, for example using maximum likelihood estimators.

References

1. Bach, S.H., Broecheler, M., Huang, B., Getoor, L.: Hinge-loss markov random fields and probabilistic soft logic. Journal of Machine Learning Research **18**, 1–67 (2017)
2. Beliakov, G., Pradera, A., Calvo, T.: Aggregation functions: A guide for practitioners, vol. 221. Springer (2007)
3. Calvo, T., Kolesárová, A., Komorníková, M., Mesiar, R.: Aggregation operators: properties, classes and construction methods. In: Aggregation operators, pp. 3–104. Springer (2002)
4. Diligenti, M., Gori, M., Sacca, C.: Semantic-based regularization for learning and inference. Artificial Intelligence **244**, 143–165 (2017)
5. Donadello, I., Serafini, L., d'Avila Garcez, A.: Logic tensor networks for semantic image interpretation. In: IJCAI International Joint Conference on Artificial Intelligence. pp. 1596–1602 (2017)
6. Giannini, F., Diligenti, M., Gori, M., Maggini, M.: On a convex logic fragment for learning and reasoning. IEEE Transactions on Fuzzy Systems (2018)
7. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT press Cambridge (2016)
8. Grabisch, M., Marichal, J.L., Mesiar, R., Pap, E.: Aggregation functions: means. Information Sciences **181**(1), 1–22 (2011)
9. Hájek, P.: Metamathematics of fuzzy logic, vol. 4. Springer Science & Business Media (2013)
10. Jenei, S.: A note on the ordinal sum theorem and its consequence for the construction of triangular norms. Fuzzy Sets and Systems **126**(2), 199–205 (2002)
11. Klement, E.P., Mesiar, R., Pap, E.: Triangular norms. position paper i: basic analytical and algebraic properties. Fuzzy Sets and Systems **143**(1), 5–26 (2004)
12. Klement, E.P., Mesiar, R., Pap, E.: Triangular norms. position paper ii: general constructions and parameterized families. Fuzzy Sets and Systems **145**(3), 411–438 (2004)
13. Klement, E.P., Mesiar, R., Pap, E.: Triangular norms. position paper iii: continuous t-norms. Fuzzy Sets and Systems **145**(3), 439–454 (2004)
14. Klement, E.P., Mesiar, R., Pap, E.: Triangular norms, vol. 8. Springer Science & Business Media (2013)
15. Kolb, S., Teso, S., Passerini, A., De Raedt, L.: Learning smt (lra) constraints using smt solvers. In: IJCAI. pp. 2333–2340 (2018)
16. Koller, D., Friedman, N., Džeroski, S., Sutton, C., McCallum, A., Pfeffer, A., Abbeel, P., Wong, M.F., Heckerman, D., Meek, C., et al.: Introduction to statistical relational learning. MIT press (2007)
17. Novák, V., Perfilieva, I., Mockor, J.: Mathematical principles of fuzzy logic, vol. 517. Springer Science & Business Media (2012)
18. Richardson, M., Domingos, P.: Markov logic networks. Machine learning **62**(1), 107–136 (2006)
19. Torra, V., Narukawa, Y.: Modeling decisions: information fusion and aggregation operators. Springer Science & Business Media (2007)
20. Xu, J., Zhang, Z., Friedman, T., Liang, Y., Broeck, G.V.d.: A semantic loss function for deep learning with symbolic knowledge. arXiv preprint arXiv:1711.11157 (2017)
21. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. In: Advances in Neural Information Processing Systems. pp. 2319–2328 (2017)