A Semantic Loss Function for Deep Learning with Symbolic Knowledge

Jingyi Xu [1] **Zilu Zhang** [2] **Tal Friedman** [1] **Yitao Liang** [1] **Guy Van den Broeck** [1]

Abstract

This paper develops a novel methodology for using symbolic knowledge in deep learning. From first principles, we derive a semantic loss function that bridges between neural output vectors and logical constraints. This loss function captures how close the neural network is to satisfying the constraints on its output. An experimental evaluation shows that it effectively guides the learner to achieve (near-)state-of-the-art results on semi-supervised multi-class classification. Moreover, it significantly increases the ability of the neural network to predict structured objects, such as rankings and paths. These discrete concepts are tremendously difficult to learn, and benefit from a tight integration of deep learning and symbolic reasoning methods.

1. Introduction

The widespread success of representation learning raises the question of which AI tasks are amenable to deep learning, which tasks require classical model-based symbolic reasoning, and whether we can benefit from a tighter integration of both approaches. In recent years, significant effort has gone towards various ways of using representation learning to solve tasks that were previously tackled by symbolic methods. Such efforts include neural computers, Turing machines, and differentiable programming (e.g. Weston et al. (2014); Reed & De Freitas (2015); Graves et al. (2016); Riedel et al. (2016)), relational embeddings (e.g. Yang et al. (2014); Lin et al. (2015)), deep learning for graph data, neural theorem proving (e.g.Bordes et al. (2013); Neelakantan et al. (2015); Duvenaud et al. (2015); Niepert et al. (2016)), and many more.

[1]Department of Computer Science, University of California Los Angeles, Los Angeles, CA, USA [2]Peking University, Beijing, China. Correspondence to: Jingyi Xu <jixu@cs.ucla.edu>, Zilu Zhang <zhangzilu@pku.edu.cn>, Tal Friedman <tal@cs.ucla.edu>, Yitao Liang <yliang@cs.ucla.edu>, Guy Van den Broeck <guyvdb@cs.ucla.edu>.

Submitted to 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

2. Background and Notation

A logical sentence (α or β) is constructed in the usual way, from variables and logical connectives ($\wedge$, $\vee$, etc.), and is also called a formula or constraint. A state or world $\mathbf{x}$ is an instantiation to all variables $\mathbf{X}$. A state $\mathbf{x}$ satisfies a sentence α, denoted $\mathbf{x} \models \alpha$, if the sentence evaluates to be true in that world, as defined in the usual way. A sentence α entails another sentence β, denoted $\alpha \models \beta$ if all worlds that satisfy α also satisfy β. A sentence α is logically equivalent to sentence β, denoted $\alpha \equiv \beta$, if both $\alpha \models \beta$ and $\beta \models \alpha$.

The output row vector of a neural net is denoted p. Each value in p represents the probability of an output and falls in $[0, 1]$. We use both softmax and sigmoid units for our output activation functions. The notation for states $\mathbf{x}$ is used to refer the an assignment, the logical sentence enforcing the assignment, or the binary vector capturing that same assignment, as these are all equivalent notions.

3. Semantic Loss

In this section, we formally introduce semantic loss. We also show that semantic loss is not just an arbitrary definition, but rather is defined uniquely by a set of intuitive properties. Stating these properties formally, we then provide a rigorous, axiomatic proof of the uniqueness of semantic loss in satisfying these.

3.1. Definition

The formal definition of this is as follows:

Definition 1 (Semantic Loss)**.** Let p be a vector of probabilities, one for each variable in $\mathbf{X}$, and let α be a sentence over $\mathbf{X}$. The semantic loss between α and p is

$$\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \propto -\log \sum_{\mathbf{x} \models \alpha} \prod_{i:\mathbf{x} \models X_i} \mathsf{p}_i \prod_{i:\mathbf{x} \models \neg X_i} (1 - \mathsf{p}_i).$$

Intuitively, the semantic loss is proportionate to a negative logarithm of the probability of generating a state that satisfies the constraint, when sampling values according to p. Hence, it is the self-information (or "surprise") of obtaining an assignment that satisfies the constraint (Jones, 1979).

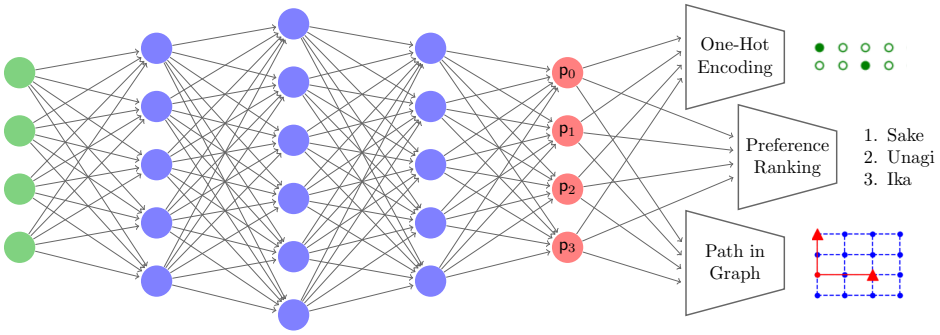
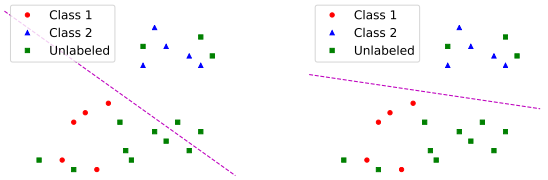


Figure 1: Outputs of a neural network feed into semantic loss functions for constraints representing a one-hot encoding, a total ranking of preferences, and paths in a grid graph.

Table 1: CIFAR. Test accuracy comparison between CNN with Semantic Loss and ladder nets.

Accuracy % with # of used labels	4000	ALL
CNN Baseline in Ladder Net	76.67 ($\pm$ 0.61)	90.73
Ladder Net (Rasmus et al., 2015)	79.60 ($\pm$0.47)	
Baseline: CNN, Whitening, Cropping	77.13	90.96
CNN with Semantic Loss	**81.79**	90.92



(a) Trained w/o semantic loss (b) Trained with semantic loss

Figure 2: Binary classification toy example: a linear classifier without and with semantic loss.

4. Conclusions

Both reasoning and semi-supervised learning are often identified as key challenges for deep learning going forward. In this paper, we developed a principled way of combining automated reasoning for propositional logic with existing deep learning architectures. Moreover, we showed that semantic loss provides significant benefits during semi-supervised classification, as well as deep structured prediction for highly complex output spaces.

References

Bordes, Antoine, Usunier, Nicolas, Garcia-Duran, Alberto, Weston, Jason, and Yakhnenko, Oksana. Translating embeddings for modeling multi-relational data. In *NIPS*, pp. 2787–2795, 2013.

Duvenaud, David K, Maclaurin, Dougal, Iparraguirre, Jorge, Bombarell, Rafael, Hirzel, Timothy, Aspuru-Guzik, Alán, and Adams, Ryan P. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, pp. 2224–2232, 2015.

Graves, Alex, Wayne, Greg, Reynolds, Malcolm, Harley, Tim, Danihelka, Ivo, Grabska-Barwińska, Agnieszka, Colmenarejo, Sergio Gómez, Grefenstette, Edward, Ramalho, Tiago, Agapiou, John, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

Jones, Douglas Samuel. *Elementary information theory*. Clarendon Press, 1979.

Kingma, Diederik P and Ba, Jimmy Lei. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Lin, Yankai, Liu, Zhiyuan, Sun, Maosong, Liu, Yang, and Zhu, Xuan. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.

Neelakantan, Arvind, Roth, Benjamin, and McCallum, Andrew. Compositional vector space models for knowledge base inference. In *ACL-IJCNLP*, pp. 156–166, 2015.

Niepert, Mathias, Ahmed, Mohamed, and Kutzkov, Konstantin. Learning convolutional neural networks for graphs. In *ICML*, pp. 2014–2023, 2016.

Nishino, Masaaki, Yasuda, Norihito, Minato, Shin-ichi, and Nagata, Masaaki. Compiling graph substructures into sentential decision diagrams. In *AAAI*, 2017.

Rasmus, Antti, Berglund, Mathias, Honkala, Mikko, Valpola, Harri, and Raiko, Tapani. Semi-supervised learning with ladder networks. In *NIPS*, 2015.

Reed, Scott and De Freitas, Nando. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

Riedel, Sebastian, Bosnjak, Matko, and Rocktäschel, Tim. Programming with a differentiable forth interpreter. *CoRR, abs/1605.06640*, 2016.

Weston, Jason, Chopra, Sumit, and Bordes, Antoine. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

Yang, Min-Chul, Duan, Nan, Zhou, Ming, and Rim, Hae-Chang. Joint relational embeddings for knowledge-based question answering. In *EMNLP*, 2014.

A. Axiomatization of Semantic Loss: Details

This appendix provides further details on our axiomatization of semantic loss.

The first axiom says that there is no loss when the logical constraint α is always true (it is a logical tautology), independent of the predicted probabilities p.

Axiom 1 (Truth)**.** The semantic loss of a true sentence is zero: $\forall \mathsf{p}, \mathrm{L}^{\mathrm{s}}(\mathit{true}, \mathsf{p}) = 0$.

Next, when enforcing two constraints on disjoint sets of variables, we want the ability to compute semantic loss for the two constraints separately, and sum the results for their joint semantic loss.

Axiom 2 (Additive Independence)**.** Let α be a sentence over $\mathbf{X}$ with probabilities p. Let β be a sentence over $\mathbf{Y}$ disjoint from $\mathbf{X}$ with probabilities q. The semantic loss between sentence $\alpha \wedge \beta$ and the joint probability vector $[\mathsf{p}\,\mathsf{q}]$ decomposes additively: $\mathrm{L}^{\mathrm{s}}(\alpha \wedge \beta, [\mathsf{p}\,\mathsf{q}]) = \mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) + \mathrm{L}^{\mathrm{s}}(\beta, \mathsf{q})$.

It directly follows from Axioms 1 and 2 that the probabilities of variables that are not used on the constraint do not affect the semantic loss.

Proposition 1 formalizes this intuition.

Proposition 1 (Locality)**.** *Let α be a sentence over $\mathbf{X}$ with probabilities p. For any $\mathbf{Y}$ disjoint from $\mathbf{X}$ with probabilities q, the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, [\mathsf{p}\,\mathsf{q}]) = \mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})$.*

Proof. Follows from the additive independence and truth axioms. Set $\beta = \mathit{true}$ in the additive independence axiom, and observe that this sets $\mathrm{L}^{\mathrm{s}}(\beta, \mathsf{q}) = 0$ because of the truth axiom. □

To maintain logical meaning, we postulate that semantic loss is monotone in the order of implication.

Axiom 3 (Monotonicity)**.** If $\alpha \models \beta$, then the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \geq \mathrm{L}^{\mathrm{s}}(\beta, \mathsf{p})$ for any vector p.

Intuitively, as we add stricter requirements to the logical constraint, going from β to α and making it harder to satisfy, semantic loss cannot decrease. For example, when β enforces the output of an neural network to encode a subtree of a graph, and we tighten that requirement in α to be a path, semantic loss cannot decrease. Every path is also a tree and any solution to α is a solution to β.

A first consequence following the monotonicity axiom is that logically equivalent sentences must incur an identical semantic loss for the same probability vector p. Hence, the semantic loss is indeed a semantic property of the logical sentence, and *does not depend on the syntax* of the sentence.

Proposition 2. *If $\alpha \equiv \beta$, then the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) = \mathrm{L}^{\mathrm{s}}(\beta, \mathsf{p})$ for any vector p.*

A second consequence is that semantic loss must be non-negative.

Proposition 3 (Non-Negativity)**.** *Semantic loss is non-negative.*

Proof. Because $\alpha \models \mathit{true}$ for all α, the monotonicity axiom implies that $\forall \mathsf{p}, \mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \geq \mathrm{L}^{\mathrm{s}}(\mathit{true}, \mathsf{p})$. By the truth axiom, $\mathrm{L}^{\mathrm{s}}(\mathit{true}, \mathsf{p}) = 0$, and therefore $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \geq 0$ for all choices of α and p. □

A state $\mathbf{x}$ is equivalently represented as a data vector, as well as a logical constraint that enforces a value for every variable in $\mathbf{X}$. When both the constraint and the predicted vector represent the same state (for example, $X_1 \wedge \neg X_2 \wedge X_3$ vs. $[1\,0\,1]$), there should be no semantic loss.

Axiom 4 (Identity)**.** For any state $\mathbf{x}$, there is zero semantic loss between its representation as a sentence, and its representation as a deterministic vector: $\forall \mathbf{x}, \mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathbf{x}) = 0$.

The axioms above together imply that any vector satisfying the constraint must incur zero loss. For example, when our constraint α requires that the output vector encodes an arbitrary total ranking, and the vector $\mathbf{x}$ correctly represents a single specific total ranking, there is no semantic loss.

Proposition 4 (Satisfaction)**.** *If $\mathbf{x} \models \alpha$, then the semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathbf{x}) = 0$.*

Proof of Proposition 4. The monotonicity axiom specializes to say that if $\mathbf{x} \models \alpha$, we have that $\forall \mathsf{p}, \mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathsf{p}) \geq \mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})$. By choosing p to be $\mathbf{x}$, this implies $\mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathbf{x}) \geq \mathrm{L}^{\mathrm{s}}(\alpha, \mathbf{x})$. From the identity axiom, $\mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathbf{x}) = 0$, and therefore $0 \geq \mathrm{L}^{\mathrm{s}}(\alpha, \mathbf{x})$. Proposition 3 bounds the loss from below as $\mathrm{L}^{\mathrm{s}}(\alpha, \mathbf{x}) \geq 0$. □

As a special case, logical literals (x or $\neg x$) constrain a single variable to take on a single value, and thus play a role similar to the labels used in supervised learning. Such constraints require an even tighter correspondence: semantic loss must act like a classical loss function (i.e., cross entropy).

Axiom 5 (Label-Literal Correspondence)**.** The semantic loss of a single literal is proportionate to the cross-entropy loss for the equivalent data label: $\mathrm{L}^{\mathrm{s}}(x, p) \propto -\log(p)$ and $\mathrm{L}^{\mathrm{s}}(\neg x, p) \propto -\log(1-p)$.

Next, we have the symmetry axioms.

Axiom 6 (Value Symmetry)**.** For all p and α, we have that $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) = \mathrm{L}^{\mathrm{s}}(\bar{\alpha}, 1 - \mathsf{p})$ where $\bar{\alpha}$ replaces every variable in α by its negation.

Axiom 7 (Variable Symmetry)**.** Let α be a sentence over $\mathbf{X}$ with probabilities p. Let π be a permutation of the variables $\mathbf{X}$, let $\pi(\alpha)$ be the sentence obtained by replacing variables x by $\pi(x)$, and let $\pi(\mathsf{p})$ be the corresponding permuted vector of probabilities. Then, $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) = \mathrm{L}^{\mathrm{s}}(\pi(\alpha), \pi(\mathsf{p}))$.

The value and variable symmetry axioms together imply the equality of the multiplicative constants in the label-literal duality axiom for all literals.

Lemma 5. *There exists a single constant K such that $\mathrm{L}^{\mathrm{s}}(X, p) = -K \log(p)$ and $\mathrm{L}^{\mathrm{s}}(\neg X, p) = -K \log(1-p)$ for any literal x.*

Proof. Value symmetry implies that $\mathrm{L}^{\mathrm{s}}(X_i, \mathsf{p}) = \mathrm{L}^{\mathrm{s}}(\neg X_i, 1 - \mathsf{p})$. Using label-literal correspondence, this implies $K_1 \log(p_i) = K_2 \log(1 - (1 - p_i))$ for the multiplicative constants K_1 and K_2 that are left unspecified by that axiom. This implies that the constants are identical. A similar argument based on variable symmetry proves equality between the multiplicative constants for different i. □

Finally, this allows us to prove the following form of semantic loss for a state $\mathbf{x}$.

Lemma 6. *For state $\mathbf{x}$ and vector p, we have $\mathrm{L}^{\mathrm{s}}(\mathbf{x}, \mathsf{p}) \propto -\sum_{i:\mathbf{x}\models X_i} \log \mathsf{p}_i - \sum_{i:\mathbf{x}\models\neg X_i} \log(1 - \mathsf{p}_i)$.*

Proof of Lemma 6. A state $\mathbf{x}$ is a conjunction of independent literals, and therefore subject to the additive independence axiom. Each literal's loss in this sum is defined by Lemma 5. □

The following and final axiom requires that semantic loss is proportionate to the logarithm of a function that is additive for mutually exclusive sentences.

Axiom 8 (Exponential Additivity)**.** Let α and β be mutually exclusive sentences (i.e., $\alpha \wedge \beta$ is unsatisfiable), and let $f^s(K, \alpha, \mathsf{p}) = K^{-\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})}$. Then, there exists a positive constant K such that $f^s(K, \alpha \vee \beta, \mathsf{p}) = f^s(K, \alpha, \mathsf{p}) + f^s(K, \beta, \mathsf{p})$.

We are now able to prove the main uniqueness theorem.

Proof of Theorem **??**. The truth axiom states that $\forall \mathsf{p}, f^s(K, \mathit{true}, \mathsf{p}) = 1$ for all positive constants K. This is the first Kolmogorov axiom of probability. The second Kolmogorov axiom for $f^s(K, ., \mathsf{p})$ follows from the additive independence axiom of semantic loss. The third Kolmogorov axiom (for the finite discrete case) is given by the exponential additivity axiom of semantic loss. Hence, $f^s(K, ., \mathsf{p})$ is a probability distribution for some choice of K, which implies the definition up to a multiplicative constant. □

B. Specification of the Convolutional Neural Network Model

Table 2 shows the slight architectural difference between the CNN used in ladder nets and ours. The major difference lies in the choice of ReLu. Note we add standard padded cropping to preprocess images and an additional fully connected layer at the end of the model, neither is used in ladder nets. We only make those slight modification so that the baseline performance reported by Rasmus et al. (2015) can be reproduced.

C. Hyper-parameter Tuning Details

Validation sets are used for tuning the weight associated with semantic loss, the only hyper-parameter that causes noticeable difference in performance for our method. For our semi-supervised classification experiments, we perform a grid search over $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ to find the optimal value. Empirically, 0.005 always gives the best or nearly the best results and we report its results on all experiments.

For the FASHION dataset specifically, because MNIST and FASHION share the same image size and structure, methods developed in MNIST should be able to directly perform on FASHION without heavy modifications. Because of this, we use the same hyper-parameters when evaluating our method. However, for the sake of fairness, we subject ladder nets to a small-scale parameter tuning in case its performance is more volatile.

For the grids experiment, the only hyper parameter that needed to be tuned was again the weight given to semantic loss, which after trying $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ was selected to be 0.5 based on validation results. For the preference learning experiment, we initially chose the semantic loss weight from $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ to be 0.1 based on validation, and then further tuned the weight to 0.25.

D. Specification of Complex Constraint Models

Grids To compile our grid constraint, we first use Nishino et al. (2017) to generate a constraint for each source destination pair. Then, we conjoin each of these with indicators specifying which source and destination pair must be used, and finally we disjoin all of these together to form our constraint.

To generate the data, we begin by randomly removing one third of edges. We then filter out connected components with fewer than 5 nodes to reduce degenerate cases, and proceed with randomly selecting pairs of point to create data points.

The predictive model we employ as our baseline is a 5 layer MLP with 50 hidden sigmoid units per layer. It is trained using Adam Optimizer, with full data batches (Kingma & Ba, 2015). Early stopping with respect to validation loss is used as a regularizer.

Preference Learning We split each user's ordering into their ordering over sushis 1,2,3,5,7,8, which we use as the features, and their ordering over 4,6,9,10 which are the labels we predict. The constraint is compiled directly from logic, as this can be done in a straightforward manner for an n-item ordering.

The predictive model we use here is a 3 layer MLP with 25 hidden sigmoid units per layer. It is trained using Adam Optimizer with full data batches (Kingma & Ba, 2015). Early stopping with respect to validation loss is used as a regularizer.

E. Probabilistic Soft Logic Encodings

We here give both encodings on the exactly-one constraint over three x_1, x_2, x_3. The first encoding is:

$$(\neg x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \neg x_3)$$

The second encoding is:

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3)$$

Following the pattern presented, readers should be able to extend both encodings to cases whether the number of variables is 10 and beyond.

Table 2: Specifications of CNNs in Ladder Net and our proposed method.

CNN in Ladder Net	CNN in this paper
Input 32×32 RGB image	
	Resizing to 36 × 36 with padding Cropping Back
Whitening Contrast Normalization Gaussian Noise with std. of 0.3	
3×3 conv. 96 BN LeakyReLU	3×3 conv. 96 BN ReLU
3×3 conv. 96 BN LeakyReLU	3×3 conv. 96 BN ReLU
3×3 conv. 96 BN LeakyReLU	3×3 conv. 96 BN ReLU
2×2 max-pooling stride 2 BN	
3×3 conv. 192 BN LeakyReLU	3×3 conv. 192 BN ReLU
3×3 conv. 192 BN LeakyReLU	3×3 conv. 192 BN ReLU
3×3 conv. 192 BN LeakyReLU	3×3 conv. 192 BN ReLU
2×2 max-pooling stride 2 BN	
3×3 conv. 192 BN LeakyReLU	3×3 conv. 192 BN ReLU
1×1 conv. 192 BN LeakyReLU	3×3 conv. 192 BN ReLU
1×1 conv. 10 BN LeakyReLU	1×1 conv. 10 BN ReLu
global meanpool BN	
	fully connected BN
10-way softmax	