A Semantic Loss Function for Deep Learning with Symbolic Knowledge

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, Guy Van den Broeck,

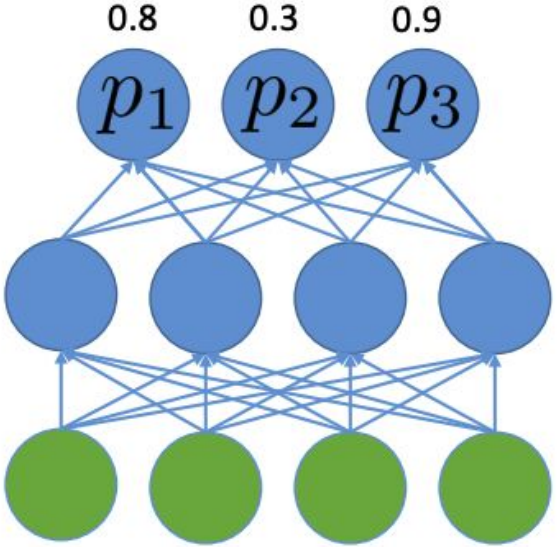
ICLR 2018

Presenter: Qiao Han (hanqiao@seas)
03/25/20

Penn Engineering

Problem

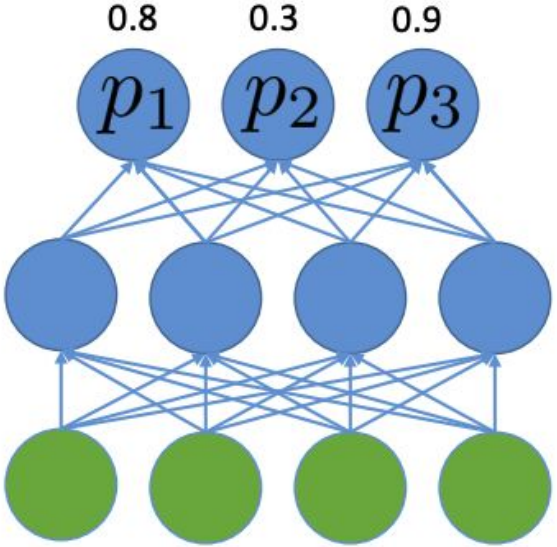
- Neural networks cannot integrate logic (or in the field of NLU semantic structure).
- Simple solution: using probabilistic representation of the output to form a loss function that represents how close outputs are to satisfying the (logical/structured) constraints



Example -- Multiclass Classification

- Multiclass Classification
  - Want exactly one class

$$\begin{cases} x_1 \neg x_2 \neg x_3 \\ \quad \vee \\ \neg x_1 x_2 \neg x_3 \\ \quad \vee \\ \neg x_1 \neg x_2 x_3 \end{cases}$$



Semantic Loss: Definition

- p: probabilities for variables in X
- alpha: a sentence over X

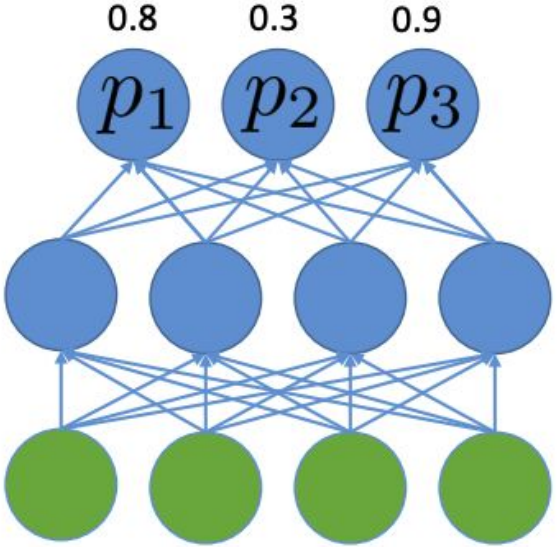
$$\mathrm{L}^{\mathrm{s}}(\alpha, \mathrm{p}) \propto -\log \sum_{\mathbf{x} \models \alpha} \prod_{i:\mathbf{x} \models X_i} \mathrm{p}_i \prod_{i:\mathbf{x} \models \neg X_i} (1 - \mathrm{p}_i)$$

Example -- Multiclass Classification

- Multiclass Classification
  - Want exactly one class

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

$$\begin{aligned} &\boldsymbol{x_1(1-x_2)(1-x_3)} \\ &\boldsymbol{+(1-x_1)x_2(1-x_3)} \\ &\boldsymbol{+(1-x_1)(1-x_2)x_3} \\ &\boldsymbol{= 0.188} \end{aligned}$$

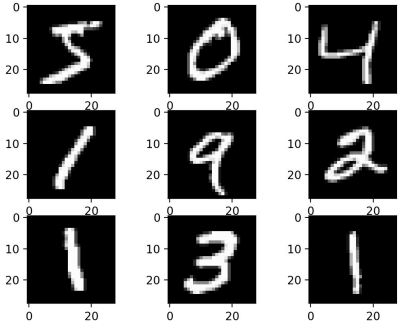


Semantic Loss: Properties

- Uniqueness
- Semantic Equivalence
- Satisfaction
- Differentiability

Application: Supervised Learning

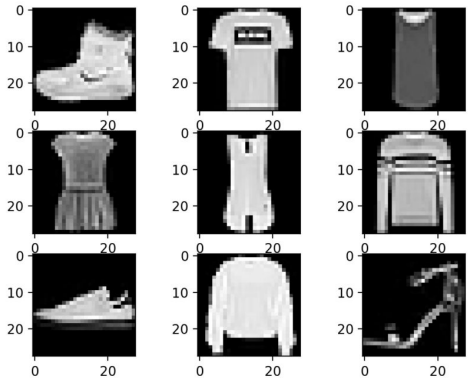
- MNIST



Accuracy % with # of used labels	100	1000	ALL
AtlasRBF (Pitelis et al., 2014)	91.9 (±0.95)	96.32 (±0.12)	98.69
Deep Generative (Kingma et al., 2014)	96.67(±0.14)	97.60 (±0.02)	99.04
Virtual Adversarial (Miyato et al., 2016)	97.67	98.64	99.36
Ladder Net (Rasmus et al., 2015)	**98.94** (±0.37)	**99.16** (±0.08)	99.43 (±0.02)
Baseline: MLP, Gaussian Noise	78.46 (±1.94)	94.26 (±0.31)	99.34 (±0.08)
Baseline: Self-Training	72.55 (±4.21)	87.43 (±3.07)	
Baseline: MLP with Entropy Regularizer	96.27 (±0.64)	98.32 (±0.34)	99.37 (±0.12)
MLP with Semantic Loss	98.38 (±0.51)	98.78 (±0.17)	99.36 (±0.02)

Application: Supervised Learning

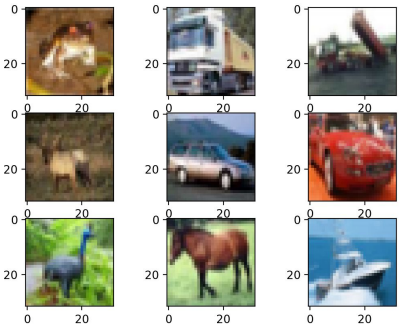
- Fashion-MNIST



Accuracy % with # of used labels	100	500	1000	ALL
Ladder Net (Rasmus et al., 2015)	81.46 (±0.64)	85.18 (±0.27)	86.48 (±0.15)	90.46
Baseline: MLP, Gaussian Noise	69.45 (±2.03)	78.12 (±1.41)	80.94 (±0.84)	89.87
MLP with Semantic Loss	**86.74** (±0.71)	**89.49** (±0.24)	**89.67** (±0.09)	89.81

Application: Supervised Learning

- CIFAR-10



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

Application: Experimental Problem Solving

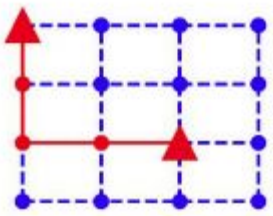
- Shortest Path with DNN
  - Encode logically plausible path using semantic loss

Test accuracy %	Coherent	Incoherent	Constraint
5-layer MLP	5.62	**85.91**	6.99
Semantic loss	**28.51**	83.14	**69.89**

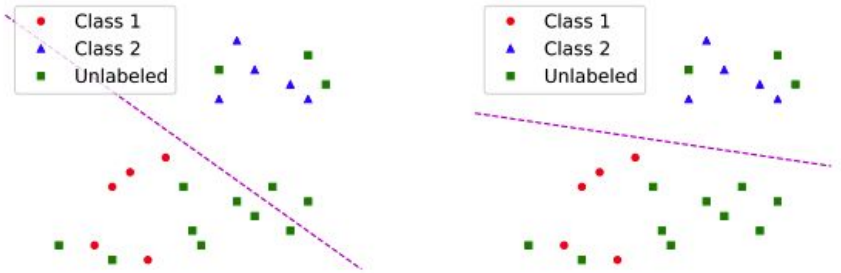
Is output the true shortest path?

Does output have true edges?

Is output a path?



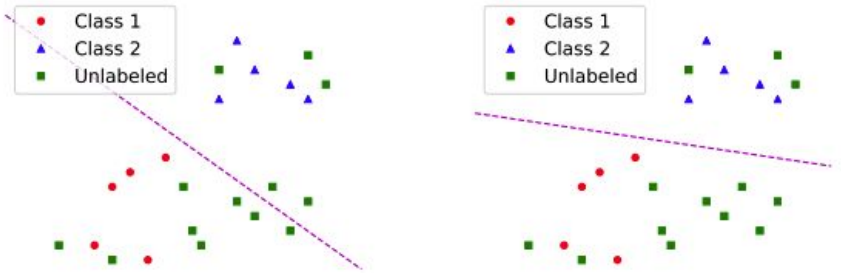
Application: Semi-supervised Learning



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

- Consider a binary classification task. Ignoring the unlabeled examples, a simple linear classifier learns to distinguish the two classes by separating the labeled examples. However, the unlabeled examples are also informative, as they must carry some properties that give them a particular label. This is the crux of semantic loss for semi-supervised learning: a model must confidently assign a consistent class even to unlabeled data

Application: Semi-supervised Learning



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

- Loss = existing loss + w * semantic loss

$$\mathrm{L}^{\mathrm{s}}(\text{exactly-one}, \mathrm{p}) \propto -\log \sum_{i=1}^{n} \mathrm{p}_i \prod_{j=1, j\neq i}^{n} (1 - \mathrm{p}_j)$$

Caveat

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

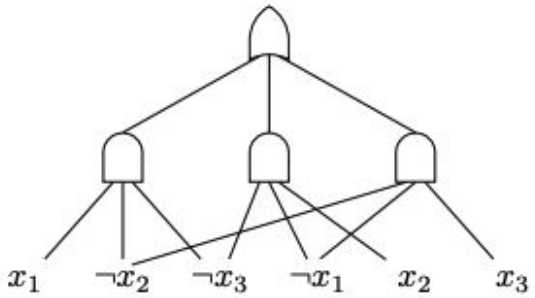
- In general, for arbitrary constraints α, semantic loss is not efficient to compute using the above definition, and more advanced automated reasoning is required. For example, using automated reasoning can reduce the time complexity to compute semantic loss for the exactly-one constraint from O(n^2), to O(n).

Caveat

- Tractability of Semantic Loss

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

AND gates as *, OR gates as +



Conclusions, Shortcomings and Future Work

- Supervised learning:
  - Pros:
    - Reaches a near SOTA accuracy on provided dataset without the use of special-purpose architecture or customized learner
  - Cons:
    - No mention of overfitting compared to baseline/SOTA

Conclusions, Shortcomings and Future Work

- Pros:
  - Integrate DNN with logical using probability as a general paradigm (e.g. shortest path)
  - Improve prediction of semi-supervised training
- Cons:
  - Potentially computationally intensive
  - Pre-tuned loss based on semantics of problems, changes the nature of the learning process
- Not addressed:
  - The more interesting case is where the loss needs to decompose over the parts of a structural decision, where symbolic knowledge can help constrain the output space

Questions?