Harnessing Deep Neural Networks with Logic Rules

By Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric P. Xing

Presentation by Rishub Jain

Motivation

- Deep NNs are:
  - **Hard to encode domain knowledge into**
  - Uninterpretable
  - Rely on labeled data

- Humans learn from:
  - Concrete examples (data)
  - **General knowledge (rules)**
    - Past tense words mostly end in -d/-ed

Related Work

- Several previous attempts
- All have issues, including:
  - Need for specific NN architecture
  - Only applicable to specialized knowledge (similarity tuples)
  - Not applicable to using NNs (instead using graphical models)
  - Poor performance

Their work - Iterative Rule Knowledge Distillation

- Usable on any NN architecture (including CNNs, RNNs, etc)
- General types of knowledge representations
- Good performance

Method

- Before:

at iteration t:

$$\boldsymbol{\theta}^{(t+1)} = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} \ell(\underbrace{\boldsymbol{y}_n}_{\text{true hard label}}, \underbrace{\boldsymbol{\sigma}_\theta(\boldsymbol{x}_n)}_{\text{soft prediction of } p_\theta}))$$

- After:

at iteration t:

$$\boldsymbol{\theta}^{(t+1)} = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} (1-\pi)\ell(\underbrace{\boldsymbol{y}_n}_{\text{true hard label}}, \underbrace{\boldsymbol{\sigma}_\theta(\boldsymbol{x}_n)}_{\text{soft prediction of } p_\theta}))$$
$$+ \underbrace{\pi}_{\text{balancing parameter}}\ell(\underbrace{\boldsymbol{s}_n^{(t)}}_{\text{soft prediction of the teacher network}}, \boldsymbol{\sigma}_\theta(\boldsymbol{x}_n)),$$

(π is exponentially decreasing hyperparameter e.g. $1-0.9^t$)

Rules

- Rules: $\{(R_l, \lambda_l)\}_{l=1}^{L}$ $\{r_{lg}(\boldsymbol{X}, \boldsymbol{Y})\}_{g=1}^{G_l}$
- Each rule grounding r_{lg} is composed of soft boolean values
- $$A\&B = \max\{A + B - 1, 0\}$$
  $$A \vee B = \min\{A + B, 1\}$$
  $$A_1 \wedge \cdots \wedge A_N = \sum_i A_i / N$$
  $$\neg A = 1 - A$$

- However, each rule seems to be any arbitrary function
- Each rule R_l has a confidence level λ_l (value of ∞ when hard constraint)

Teacher Network: *q(Y|X)*

- Models p with the constraint: $E_q[r(X,Y)] = 1$, with confidence λ

- Optimization Problem:

$$\min_{q,\boldsymbol{\xi}\geq 0} \ \mathrm{KL}(q \| p_\theta(\boldsymbol{Y}|\boldsymbol{X})) + C\sum\nolimits_l \xi_l$$
$$\text{s.t.} \ \ \lambda_l(1 - \mathbb{E}_q[r_l(\boldsymbol{X},\boldsymbol{Y})]) \leq \xi_l$$
$$l = 1, \ldots, L$$

slack variable

rule constraints

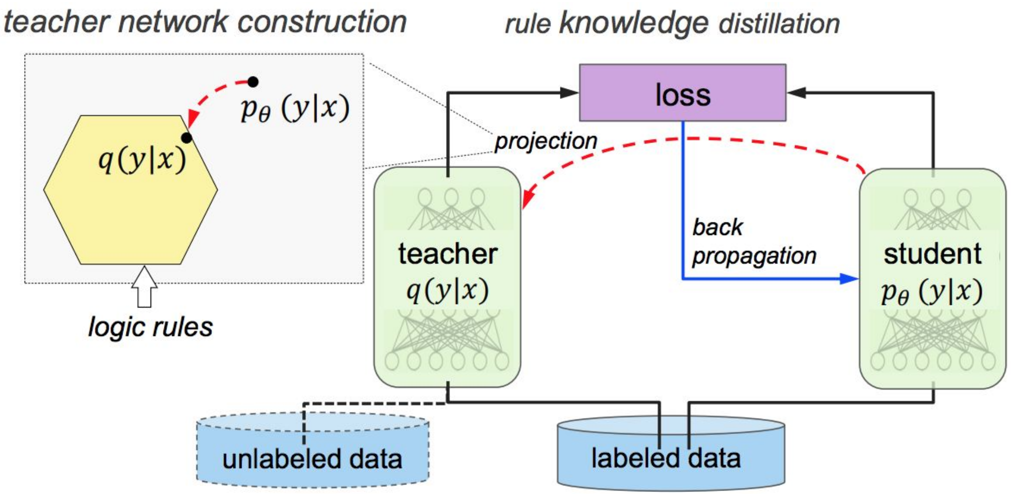
- Closed-form:

$$q^*(\boldsymbol{Y}|\boldsymbol{X}) \propto p_\theta(\boldsymbol{Y}|\boldsymbol{X}) \exp\left\{ -\sum_l C\lambda_l(1 - r_l(\boldsymbol{X},\boldsymbol{Y})) \right\}$$

(C is a fixed hyperparameter e.g. 400)

Method

- For each mini-batch:
  - Compute student: $p_\theta(Y|X)$
  - Compute projection on logic rule space (teacher): $q(Y|X)$
  - Run backprop to update weights of $p_\theta(Y|X)$ based on weighted average of teacher and student



At Test time

- Can use student $p_\theta(Y|X)$ or teacher $q(Y|X)$ for inference
- Tradeoff:
  - Teacher performs better on average
  - Student can be faster if the rule computation is expensive or unavailable at test time

Final Algorithm

Algorithm 1 Harnessing NN with Rules

Input: The training data $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$,
The rule set $\mathcal{R} = \{(R_l, \lambda_l)\}_{l=1}^{L}$,
Parameters: π – imitation parameter
C – regularization strength

1: Initialize neural network parameter $\boldsymbol{\theta}$
2: **repeat**
3: Sample a minibatch $(\boldsymbol{X}, \boldsymbol{Y}) \subset \mathcal{D}$
4: Construct teacher network q with Eq.(4)
5: Transfer knowledge into p_θ by updating $\boldsymbol{\theta}$ with Eq.(2)
6: **until** convergence

Output: Distill student network p_θ and teacher network q

Results - Sentiment Classification

- Sentiment Classification using a CNN
- Logic Rule: has-'A-but-B'-structure$(S) \Rightarrow$ $(\mathbf{1}(y=+) \Rightarrow \boldsymbol{\sigma}_\theta(B)_+ \ \wedge \ \boldsymbol{\sigma}_\theta(B)_+ \Rightarrow \mathbf{1}(y=+))$,

	Model	Accuracy (%)
1	CNN (Kim, 2014)	87.2
2	-but-clause	87.3
3	-ℓ_2-reg	87.5
4	-project	87.9
5	-opt-project	88.3
6	-pipeline	87.9
7	-Rule-p	88.8
8	-Rule-q	**89.3**

Table 2: Performance of different rule integration methods on SST2. 1) CNN is the base network; 2) "-but-clause" takes the clause after "but" as input; 3) "-ℓ_2-reg" imposes a regularization term $\gamma\|\boldsymbol{\sigma}_\theta(S) - \boldsymbol{\sigma}_\theta(Y)\|_2$ to the CNN objective, with the strength γ selected on dev set; 4) "-project" projects the trained base CNN to the rule-regularized subspace with Eq.(3); 5) "-opt-project" directly optimizes the projected CNN; 6) "-pipeline" distills the pre-trained "-opt-project" to a plain CNN; 7-8) "-Rule-p" and "-Rule-q" are our models with p being the distilled student network and q the teacher network. Note that "-but-clause" and "-ℓ_2-reg" are ad-hoc methods applicable specifically to the "but"-rule.

	Model	SST2	MR	CR
1	CNN (Kim, 2014)	87.2	81.3±0.1	84.3±0.2
2	CNN-Rule-p	88.8	81.6±0.1	85.0±0.3
3	CNN-Rule-q	89.3	**81.7±0.1**	**85.3±0.3**
4	MGNC-CNN (Zhang et al., 2016)	88.4	–	–
5	MVCNN (Yin and Schutze, 2015)	**89.4**	–	–
6	CNN-multichannel (Kim, 2014)	88.1	81.1	85.0
7	Paragraph-Vec (Le and Mikolov, 2014)	87.8	–	–
8	CRF-PR (Yang and Cardie, 2014)	–	–	82.7
9	RNTN (Socher et al., 2013)	85.4	–	–
10	G-Dropout (Wang and Manning, 2013)	–	79.0	82.1

Table 1: Accuracy (%) of Sentiment Classification. Row 1, CNN (Kim, 2014) is the base network corresponding to the "CNN-non-static" model in (Kim, 2014). Rows 2-3 are the networks enhanced by our framework: CNN-Rule-p is the student network and CNN-Rule-q is the teacher network. For MR and CR, we report the average accuracy±one standard deviation using 10-fold cross validation.

Results - Named Entity Recognition

- Uses BLSTM-CNN
- Logic Rule:

$$\text{is-counterpart}(X, A) \Rightarrow 1 - \|c(\mathbf{e}_y) - c(\boldsymbol{\sigma}_\theta(A))\|_2,$$

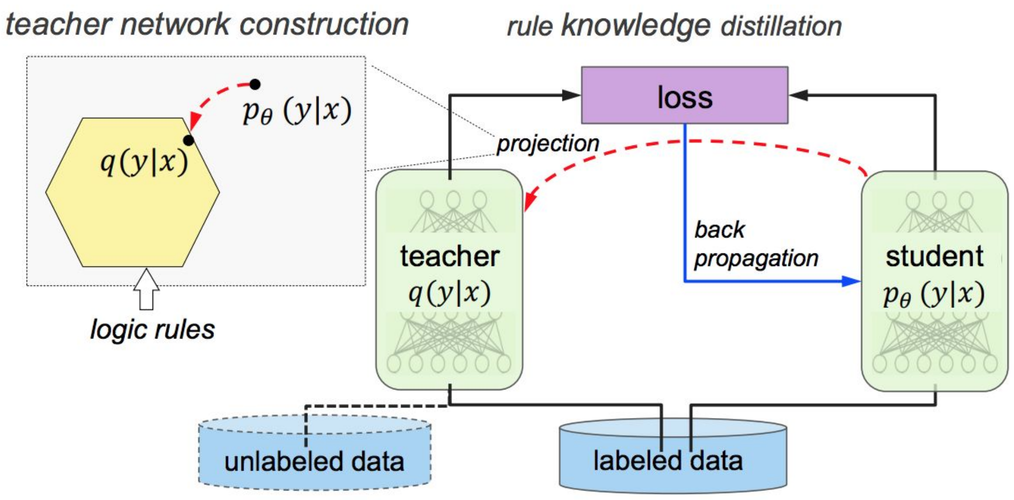
$$\text{equal}(y_{i-1}, \text{I-ORG}) \Rightarrow \neg\ \text{equal}(y_i, \text{B-PER})$$

	Model	F1
1	BLSTM	89.55
2	BLSTM-Rule-trans	p: 89.80, q: 91.11
3	BLSTM-Rules	p: 89.93, q: **91.18**
4	NN-lex (Collobert et al., 2011)	89.59
5	S-LSTM (Lample et al., 2016)	90.33
6	BLSTM-lex (Chiu and Nichols, 2015)	90.77
7	BLSTM-CRF$_1$ (Lample et al., 2016)	90.94
8	Joint-NER-EL (Luo et al., 2015)	91.20
9	BLSTM-CRF$_2$ (Ma and Hovy, 2016)	**91.21**

Table 4: Performance of NER on CoNLL-2003. Row 2, BLSTM-Rule-trans imposes the transition rules (Eq.(6)) on the base BLSTM. Row 3, BLSTM-Rules further incorporates the list rule (Eq.(7)). We report the performance of both the student model p and the teacher model q.

Semi-supervised Learning

- Can use unlabeled data to incorporate rule structure in student
- Loss during semi-supervised phase just becomes difference between student and teacher



Semi-supervised Results - Sentiment Classification

	Data size	5%	10%	30%	100%
1	CNN	79.9	81.6	83.6	87.2
2	-Rule-p	81.5	83.2	84.5	88.8
3	-Rule-q	82.5	83.9	85.6	**89.3**
4	-semi-PR	81.5	83.1	84.6	–
5	-semi-Rule-p	81.7	83.3	84.7	–
6	-semi-Rule-q	**82.7**	**84.2**	**85.7**	–

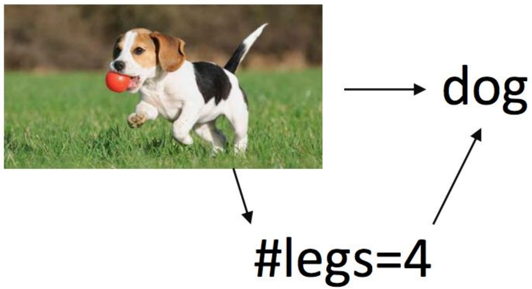
Table 3: Accuracy (%) on SST2 with varying sizes of labeled data and semi-supervised learning. The header row is the percentage of labeled examples for training. Rows 1-3 use only the supervised data. Rows 4-6 use semi-supervised learning where the remaining training data are used as unlabeled examples. For "-semi-PR" we only report its projected solution (in analogous to q) which performs better than the non-projected one (in analogous to p).

Their Contributions

- Incorporated domain knowledge into NN model
- Usable on any NN architecture
- General types of knowledge representations
- Good performance

Future Work

- Incorporate intermediate representations



- Learn confidence level λ