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:
  \[
  \theta^{(t+1)} = \arg \min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \sigma_\theta(x_n))
  \]

- After:
  \[
  \theta^{(t+1)} = \arg \min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} (1 - \pi) \ell(y_n, \sigma_\theta(x_n)) + \pi \ell(s_n^{(t)}, \sigma_\theta(x_n))
  \]

\((\pi) is exponentially decreasing hyperparameter e.g. 1-0.9^t\)
Rules

- Rules: $\{(R_t, \lambda_t)\}_{t=1}^{L} \times \{r_{lg}(X, Y)\}_{g=1}^{G}$
- Each rule grounding $r_{lg}$ is composed of soft boolean values
- $A \& B = \max\{A + B - 1, 0\}$
  $A \lor B = \min\{A + B, 1\}$
  $A_1 \land \cdots \land A_N = \sum_i A_i / N$
  $\neg A = 1 - A$
- However, each rule seems to be any arbitrary function
- Each rule $R_i$ has a confidence level $\lambda_i$ (value of $\infty$ when hard constraint)
Teacher Network: $q(Y|X)$

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

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

- Closed-form:
  \[
  q^*(Y|X) \propto p_\theta(Y|X) \exp \left\{ -\sum_l C\lambda_l(1 - r_l(X,Y)) \right\}
  \]

(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 $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$
1. The rule set $\mathcal{R} = \{(R_l, \lambda_l)\}_{l=1}^L$
2. Parameters: $\pi$ – imitation parameter
3. $C$ – regularization strength

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

Output: Distill student network $p_\theta$ and teacher network $q$
Results - Sentiment Classification

- Sentiment Classification using a CNN
- Logic Rule: \[ \text{has-`A-but-B'-structure}(S) \Rightarrow (1(y = +) \Rightarrow \sigma_\theta(B)_+ \land \sigma_\theta(B)_+ \Rightarrow 1(y = +)) \]

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

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.

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) “-l2-reg” imposes a regularization term \( \gamma \| \sigma_\theta(S) - \sigma_\theta(Y) \|_2 \) to the CNN objective, with the strength \( \gamma \) 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 “-l2-reg” are ad-hoc methods applicable specifically to the “but”-rule.
Results - Named Entity Recognition

- Uses BLSTM-CNN
- Logic Rule:

\[\text{is-counterpart}(X, A) \Rightarrow 1 - \|c(e_y) - c(\sigma_\theta(A))\|_2,\]
\[\text{equal}(y_{i-1}, \text{I-ORG}) \Rightarrow \neg \text{equal}(y_i, \text{B-PER})\]

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  BLSTM</td>
<td>89.55</td>
</tr>
<tr>
<td>2  BLSTM-Rule-trans</td>
<td>p: 89.80, q: 91.11</td>
</tr>
<tr>
<td>3  BLSTM-Rules</td>
<td>p: 89.93, q: 91.18</td>
</tr>
<tr>
<td>4  NN-lex (Collobert et al., 2011)</td>
<td>89.59</td>
</tr>
<tr>
<td>5  S-LSTM (Lample et al., 2016)</td>
<td>90.33</td>
</tr>
<tr>
<td>6  BLSTM-lex (Chiu and Nichols, 2015)</td>
<td>90.77</td>
</tr>
<tr>
<td>7  BLSTM-CRF₁ (Lample et al., 2016)</td>
<td>90.94</td>
</tr>
<tr>
<td>8  Joint-NER-EL (Luo et al., 2015)</td>
<td>91.20</td>
</tr>
<tr>
<td>9  BLSTM-CRF₂ (Ma and Hovy, 2016)</td>
<td>91.21</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Data size</th>
<th>5%</th>
<th>10%</th>
<th>30%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CNN</td>
<td>79.9</td>
<td>81.6</td>
<td>83.6</td>
<td>87.2</td>
</tr>
<tr>
<td>2 -Rule-p</td>
<td>81.5</td>
<td>83.2</td>
<td>84.5</td>
<td>88.8</td>
</tr>
<tr>
<td>3 -Rule-q</td>
<td>82.5</td>
<td>83.9</td>
<td>85.6</td>
<td><strong>89.3</strong></td>
</tr>
<tr>
<td>4 -semi-PR</td>
<td>81.5</td>
<td>83.1</td>
<td>84.6</td>
<td>-</td>
</tr>
<tr>
<td>5 -semi-Rule-p</td>
<td>81.7</td>
<td>83.3</td>
<td>84.7</td>
<td>-</td>
</tr>
<tr>
<td>6 -semi-Rule-q</td>
<td><strong>82.7</strong></td>
<td><strong>84.2</strong></td>
<td><strong>85.7</strong></td>
<td>-</td>
</tr>
</tbody>
</table>

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 $\lambda$

⇒ dog

⇒ #legs=4