Distributional Smoothing with Virtual Adversarial Training,
Adversarial Training Methods for Semi-Supervised Text Classification

Miyato et al., ICLR 2016, 2017
Outline

- Semi-supervised learning
  - Small set of labeled + large set of unlabeled data
  - Supervised + unsupervised learnings
    - using pretrained word embeddings is implicitly semi-supervised learning
  - High performance with few labeled data
    - SOTA on IMDB, BERT-finetune, 200 vs 25000 labeled ~ 95.8%
  - Consistency regularization and data augmentation

- Adversarial examples

- Virtual adversarial examples (this work)

- (Virtual) adversarial embeddings (this work)
  - one of few data augmentation methods for NLP,
  - internal rather external, no changes in context words
  - can improve performances of many NLP tasks, already text classification and relation extraction

- Benefits of (virtual) adversarial embeddings (this work)
  - encouraging generalization
  - Improving semantics of word embeddings while learning the task

- Experimental results on IMDB dataset
Consistency Regularization and Data Augmentation

- Consistency regularization

\[ L(x_l, y_l; \theta) = H(q(y_l | x_l), p(y | x_l; \theta)) \]
\[ L_c(x_{ul}; \theta) = KL(p(y | x_{ul}; \theta), p(y | x_{ul}^+; \theta)) \]

- regularize the model to produce similar output distributions for both original input \(x\) and augmented input \(x^+\) (\(x^+\) has to be “close” to \(x\) in input space)
- labels are not needed for this regularizer term → it suits semi-supervised learning
- intuitively,
  - the model assumes pseudo-labels for unlabeled data and force its augmentation to match its output distribution
  - labels are propagated from labeled to unlabeled examples

Two ways of generating \(x^+\)

- Model’s internal stochasticity: Π-Model, Mean Teacher, ICT
- Data augmentation: Virtual adversarial examples (this work), MixMatch, Unsupervised data augmentation,
Adversarial Examples

- Adversarial examples:

  \[ L(x_l, y_l; \theta) = \mathcal{H}(q(y_l|x_l), p(y|x_l; \theta)) \]
  \[ \delta = \arg\max_{\delta \in S} L(x_l + \delta, y_l; \theta) \]

  - find perturbation \( \delta \) that changes the model’s prediction, i.e., maximize
    the model’s loss on given labeled input \( x_l \)
    - \( x_{adv} = x_l + \delta \) is the resulted adversarial example
  - control perturbation so that the adversaries are “close” to original
    example by a constraint \( S \), e.g., \( l_2 \)-ball
    - adversarial examples fit in the consistency regularization approach

- Approximate solution \((S = l_2)\): gradient descent

  \[ \delta = -\varepsilon \frac{g}{\|g\|_2}, \text{ where } g = \nabla_x L(x_l, y_l; \theta) \]
Virtual Adversarial Examples (this work)

- Adversarial examples
  \[ L(x_l, y_l; \theta) = \mathcal{H}(q(y_l|x_l), p(y|x_l; \theta)) \]
  \[ \delta = \arg\max_{\delta \in \mathcal{S}} L(x_l + \delta, y_l; \theta) \]

- Virtual adversarial examples
  \[ L_c(x_{ul}; \theta) = KL(p(y|x_{ul}; \theta), p(y|x_{ul}^+; \theta)) \]
  \[ \delta = \arg\max_{\delta \in \mathcal{S}} L_c(x_{ul} + \delta; \theta) \]

- similar to adversarial examples, find the perturbation \( \delta \) that maximizes the consistency regularizer \( L_c \) instead
- labels are not needed
- \( x^+ = x_{vadv} = x_{ul} + \delta \) is the resulted virtual adversarial example
(Virtual) Adversarial Embeddings (this work)

- **(Virtual) Adversarial Examples**
  - Adversarial perturbation: small changes to real-valued inputs

- **(Virtual) Adversarial Embeddings for NLP**
  - issue: discrete inputs, series of high-dimensional one-hot vectors → cannot directly compute perturbation
  - solution: compute perturbation on word embeddings instead of discrete word inputs

![Diagram of LSTM-based text classification model.](image)

(a) LSTM-based text classification model.

![Diagram of model with perturbed embeddings.](image)

(b) The model with perturbed embeddings.
Benefits of (Virtual) Adversarial Embeddings

- Encouraging generalization

- test loss consistently stays small
Benefits of (Virtual) Adversarial Embeddings

- Improving semantics of word embeddings during learning the task

<table>
<thead>
<tr>
<th></th>
<th>‘good’</th>
<th>‘bad’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>Random</td>
<td>Adversarial</td>
</tr>
<tr>
<td>1</td>
<td>great</td>
<td>great</td>
</tr>
<tr>
<td>2</td>
<td>decent</td>
<td>decent</td>
</tr>
<tr>
<td>3</td>
<td>×bad</td>
<td>excellent</td>
</tr>
<tr>
<td>4</td>
<td>excellent</td>
<td>nice</td>
</tr>
<tr>
<td>5</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>6</td>
<td>fine</td>
<td>×bad</td>
</tr>
<tr>
<td>7</td>
<td>nice</td>
<td>fine</td>
</tr>
<tr>
<td>8</td>
<td>interesting</td>
<td>interesting</td>
</tr>
<tr>
<td>9</td>
<td>solid</td>
<td>entertaining</td>
</tr>
<tr>
<td>10</td>
<td>entertaining</td>
<td>solid</td>
</tr>
</tbody>
</table>

- baseline is strongly influenced by the grammatical structure of language, but not by the semantics of the task.
  - e.g., “bad” appears in the list of nearest neighbors of “good”
- (virtual) adversarial examples improve the word semantics
  - e.g., “bad” no longer appears in the 10 top nearest neighbors to “good”, falling to the 19th nearest neighbor
  - (virtual) adversarial examples ensures that the meaning of a sentence cannot be inverted via a small change, so these words with similar grammatical role but different meaning become separated.
## Experimental Results on IMDB Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Test error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (without embedding normalization)</td>
<td>7.33%</td>
</tr>
<tr>
<td>Baseline</td>
<td>7.39%</td>
</tr>
<tr>
<td>Random perturbation with labeled examples</td>
<td>7.20%</td>
</tr>
<tr>
<td>Random perturbation with labeled and unlabeled examples</td>
<td>6.78%</td>
</tr>
<tr>
<td>Adversarial</td>
<td>6.21%</td>
</tr>
<tr>
<td>Virtual Adversarial</td>
<td>5.91%</td>
</tr>
<tr>
<td>Adversarial + Virtual Adversarial</td>
<td>6.09%</td>
</tr>
<tr>
<td>Virtual Adversarial (on bidirectional LSTM)</td>
<td>5.91%</td>
</tr>
<tr>
<td>Adversarial + Virtual Adversarial (on bidirectional LSTM)</td>
<td>6.02%</td>
</tr>
<tr>
<td>Full+Unlabeled+BoW (Maas et al., 2011)</td>
<td>11.11%</td>
</tr>
<tr>
<td>Transductive SVM (Johnson &amp; Zhang, 2015b)</td>
<td>9.99%</td>
</tr>
<tr>
<td>NBSVM-bigrams (Wang &amp; Manning, 2012)</td>
<td>8.78%</td>
</tr>
<tr>
<td>Paragraph Vectors (Le &amp; Mikolov, 2014)</td>
<td>7.42%</td>
</tr>
<tr>
<td>SA-LSTM (Dai &amp; Le, 2015)</td>
<td>7.24%</td>
</tr>
<tr>
<td>One-hot bi-LSTM* (Johnson &amp; Zhang, 2016b)</td>
<td>5.94%</td>
</tr>
</tbody>
</table>
Thank you!