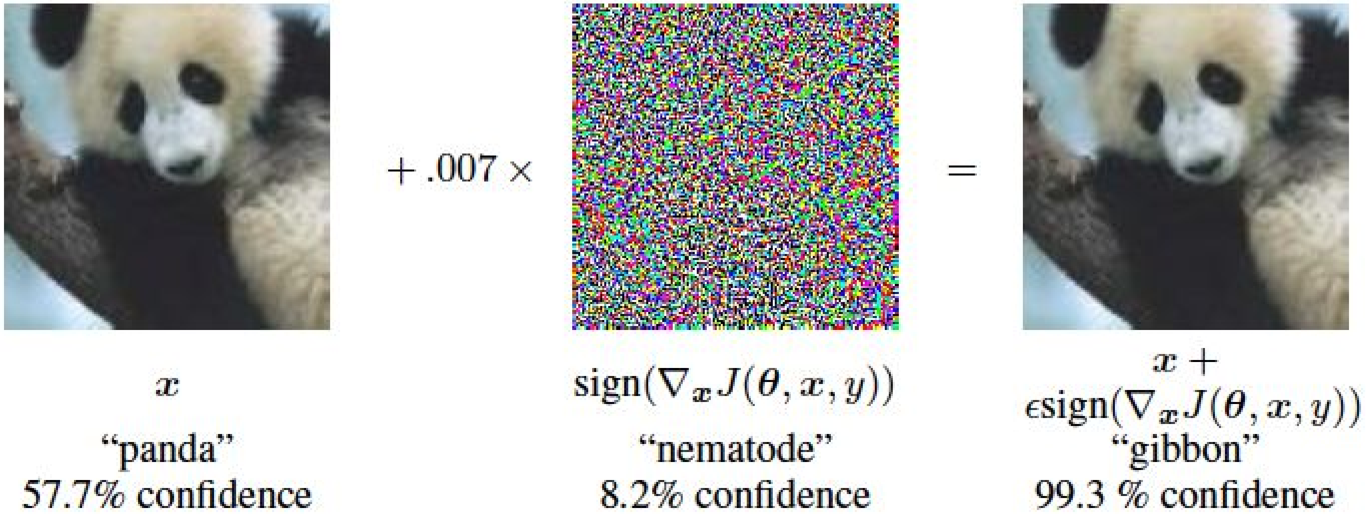
Adversarial training methods for semi-supervised text classification

Takeru Miyato[†] (Kyoto Univ., Google Brain),
Andrew Dai (Google Brain), Ian Goodfellow (OpenAI)

[†]present affiliation is Preferred Networks, Inc.

Adversarial examples



[Goodfellow, et al 2015]

Adversarial training (Goodfellow et al. [2015])

- Additional cost

$$-\min_{\|\boldsymbol{r}\|\leq\epsilon} \log p(y \mid \boldsymbol{x}+\boldsymbol{r}, \boldsymbol{\theta})$$

- $\boldsymbol{r}$ is the perturbation on input $\boldsymbol{x}$.
- Train classifier to be robust to the <u>worst</u> perturbation.
- Only one hyperparameter: ε

Virtual adversarial training (Miyato et al. [2016])

- Additional cost:

$$\max_{||\boldsymbol{r}|| \leq \epsilon} \mathrm{KL}[p(\cdot \mid \boldsymbol{x}, \boldsymbol{\theta}) || p(\cdot \mid \boldsymbol{x} + \boldsymbol{r}, \boldsymbol{\theta})]$$

- Actual label is not required.
    - Virtual adversarial training can be applied to semi-supervised learning.
- Only one hyperparameter: ε.

Virtual adversarial training on text

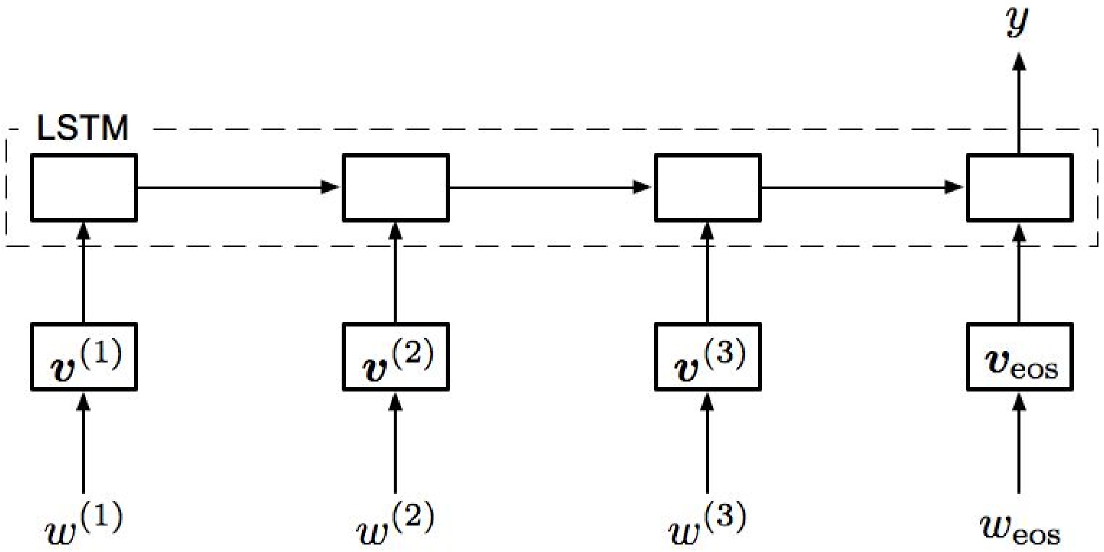
- There are abundant unlabeled examples in text domain.
- Training on text sometimes takes a very long time (e.g. recurrent models)
  - Our method requires little tuning of hyperparameters.
- In our work, we applied virtual adversarial training to semi-supervised text classification.
  - Achieved state of the art performance.

Apply adversarial training to text classification

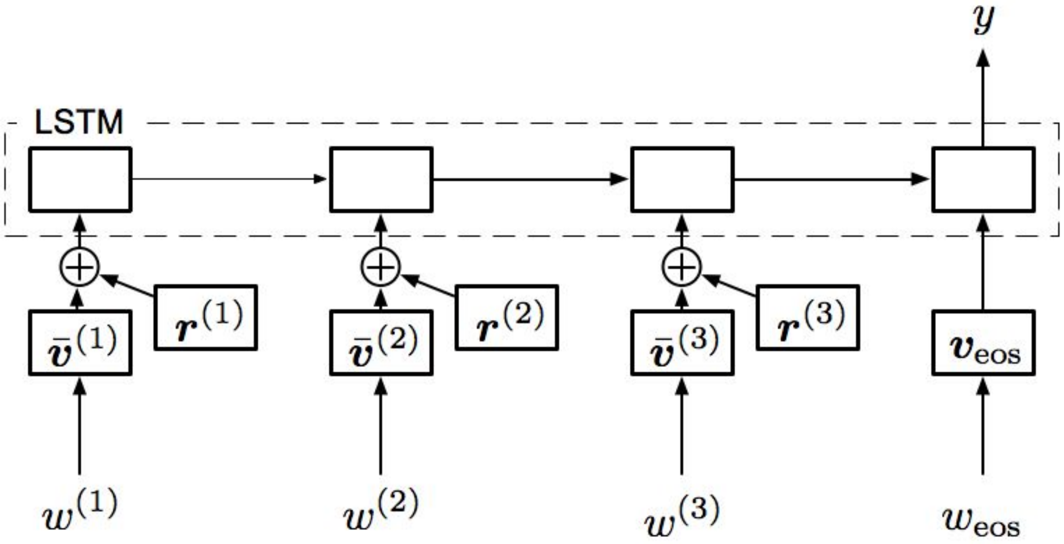
- Text is a sequence of words (discrete input).
  - It is difficult to define adversarial examples on word sequences.
- We define perturbations on continuous word embeddings.

LSTM text classification model

- A sequence of *T* words as $\{w^{(t)} | t = 1, \ldots, T\}$
- Label as *y*
- Embedding matrix $\boldsymbol{V} \in \mathcal{R}^{(K+1) \times D}$



Model with perturbation



- Normalized embeddings

$$\bar{\boldsymbol{v}}_k = \frac{\boldsymbol{v}_k - \mathrm{E}(\boldsymbol{v})}{\sqrt{\mathrm{Var}(\boldsymbol{v})}} \text{ where } \mathrm{E}(\boldsymbol{v}) = \sum_{j=1}^{K} f_j \boldsymbol{v}_j, \mathrm{Var}(\boldsymbol{v}) = \sum_{j=1}^{K} f_j \left(\boldsymbol{v}_j - \mathrm{E}(\boldsymbol{v})\right)^2$$

Adversarial perturbation on embeddings

- Adversarial perturbation

$$\boldsymbol{r}_{\mathrm{adv}} = -\epsilon \boldsymbol{g} / \|\boldsymbol{g}\|_2 \text{ where } \boldsymbol{g} = \nabla_{\boldsymbol{s}} \log p(y \mid \boldsymbol{s}; \hat{\boldsymbol{\theta}})$$

($\hat{\theta}$: a constant set to the current parameters
$\boldsymbol{s}$: a concatenation of a sequence of word embedding vectors $[\bar{\boldsymbol{v}}^{(1)}, \ldots, \bar{\boldsymbol{v}}^{(T)}]$)

- Adversarial loss (regularization term)

$$L_{\mathrm{adv}}(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{n=1}^{N} \log p(y_n \mid \boldsymbol{s}_n + \boldsymbol{r}_{\mathrm{adv},n}, \boldsymbol{\theta})$$

Virtual adversarial perturbation on embeddings

- Virtual adversarial perturbation

$$\boldsymbol{r}_{\text{v-adv}} = \epsilon \boldsymbol{g} / \|\boldsymbol{g}\|_2 \text{ where } \boldsymbol{g} = \nabla_{\boldsymbol{s}+\boldsymbol{d}} \mathrm{KL}\left[p(\cdot \mid \boldsymbol{s}; \hat{\boldsymbol{\theta}}) || p(\cdot \mid \boldsymbol{s} + \boldsymbol{d}; \hat{\boldsymbol{\theta}}) \right]$$

- Virtual adversarial loss

$$L_{\text{v-adv}}(\boldsymbol{\theta}) = \frac{1}{N'} \sum_{n'=1}^{N'} \mathrm{KL}\left[p(\cdot \mid \boldsymbol{s}_{n'}; \hat{\boldsymbol{\theta}}) || p(\cdot \mid \boldsymbol{s}_{n'} + \boldsymbol{r}_{\text{v-adv},n'}; \boldsymbol{\theta}) \right]$$

N’ is the number of both labeled and unlabeled examples

Summary of datasets

- Datasets
  - 4 semi-supervised datasets
  - 1 supervised dataset (DBpedia)

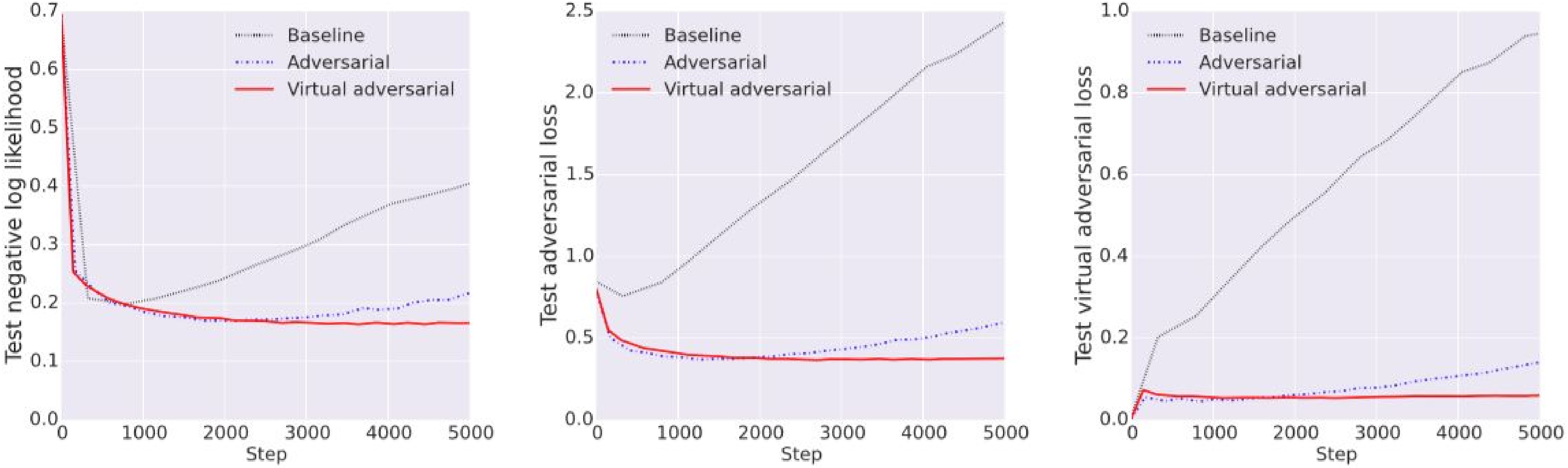
	Classes	Train	Test	Unlabeled	Avg. T	Max T
IMDB	2	25,000	25,000	50,000	239	2,506
Elec	2	24,792	24,897	197,025	110	5,123
Rotten Tomatoes	2	9596	1066	7,911,684	20	54
DBpedia	14	560,000	70,000	–	49	953
RCV1	55	15,564	49,838	668,640	153	9,852

Training

- We first do pretraining following Dai and Le[2015] (recurrent language model).

- We optimized dropout rate on embeddings and norm constraint ε on adversarial and virtual adversarial training.

Learning curves on IMDB (on test sets)



(a) Negative log likelihood (b) $L_{\mathrm{adv}}(\boldsymbol{\theta})$ (c) $L_{\text{v-adv}}(\boldsymbol{\theta})$

Test performance on IMDB

Method	Test error rate
Baseline (without embedding normalization)	7.33%
Baseline	7.39%
Random perturbation with labeled examples	7.20%
Random perturbation with labeled and unlabeled examples	6.78%
Adversarial	6.21%
Virtual Adversarial	**5.91**%
Adversarial + Virtual Adversarial	6.09%
Virtual Adversarial (on bidirectional LSTM)	**5.91**%
Adversarial + Virtual Adversarial (on bidirectional LSTM)	6.02%
Full+Unlabeled+BoW [18]	11.11%
Paragraph Vectors [14]	7.42%
SA-LSTM [4]	7.24%
One-hot bi-LSTM* [11]	5.94%

Nearest neighbors to ***good*** and ***bad***

	'good'				*'bad'*			
	baseline	**random**	**adversarial**	**virtual adversarial**	**baseline**	**random**	**adversarial**	**virtual adversarial**
1	great	great	decent	decent	terrible	terrible	terrible	terrible
2	decent	decent	great	great	awful	awful	awful	awful
3	×bad	excellent	nice	nice	horrible	horrible	horrible	horrible
4	excellent	nice	fine	fine	×good	×good	poor	poor
5	Good	Good	entertaining	entertaining	Bad	poor	BAD	BAD
6	fine	×bad	interesting	interesting	BAD	BAD	stupid	stupid
7	nice	fine	Good	Good	poor	Bad	Bad	Bad
8	interesting	interesting	excellent	cool	stupid	stupid	laughable	laughable
9	solid	entertaining	solid	enjoyable	Horrible	Horrible	lame	lame
10	entertaining	solid	cool	excellent	horrendous	horrendous	Horrible	Horrible

Nearest neighbors to ***great***

	'great'							
	baseline		**random**		**adversarial**		**virtual adversarial**	
1	wonderful	0.244	excellent	0.239	excellent	0.169	wonderful	0.159
2	excellent	0.248	wonderful	0.240	wonderful	0.172	excellent	0.163
3	good	0.285	good	0.288	fantastic	0.213	fantastic	0.208
4	fantastic	0.303	fantastic	0.295	brilliant	0.233	brilliant	0.226
5	terrific	0.308	terrific	0.301	amazing	0.236	amazing	0.227
6	brilliant	0.310	brilliant	0.305	terrific	0.247	terrific	0.234
7	awesome	0.325	awesome	0.309	awesome	0.251	incredible	0.247
8	amazing	0.330	amazing	0.332	incredible	0.263	awesome	0.248
9	fine	0.343	fine	0.347	superb	0.282	superb	0.260
10	incredible	0.350	incredible	0.347	outstanding	0.293	outstanding	0.288
11	superb	0.368	superb	0.355	magnificent	0.310	magnificent	0.299
12	outstanding	0.375	outstanding	0.360	fine	0.314	marvelous	0.312
13	marvelous	0.390	marvelous	0.375	marvelous	0.317	extraordinary	0.315
14	magnificent	0.398	tremendous	0.387	good	0.321	fine	0.321
15	tremendous	0.399	magnificent	0.389	extraordinary	0.333	good	0.331

Test performance on other semi-supervised datasets

Elec:sentiment classification(2 classes)
RCV1:category classification(55 classes)

Method	Test error rate	
	Elec	RCV1
Baseline	6.24%	7.40%
Adversarial	5.61%	7.12%
Virtual Adversarial	5.54%	7.05%
Adversarial + Virtual Adversarial	**5.40**%	6.97%
Virtual Adversarial (on bidirectional LSTM)	5.55%	6.71%
Adversarial + Virtual Adversarial (on bidirectional LSTM)	5.45%	**6.68**%
One-hot CNN* [10]	6.27%	7.71%
One-hot CNN† [11]	5.87%	7.15%
One-hot bi-LSTM† [11]	5.55%	8.52%

Test performance on other semi-supervised datasets

Rotten Tomatoes:sentiment classification(2 classes)

Method	Test error rate
Baseline	17.9%
Adversarial	16.8%
Virtual Adversarial	19.1%
Adversarial + Virtual Adversarial	**16.6%**
NBSVM-bigrams[29]	20.6%
CNN*[12]	18.5%
AdaSent*[32]	16.9%
SA-LSTM † [4]	16.7%

Why VAT is worse than the baseline?

- Virtual adversarial loss on unlabeled examples would overwhelm the supervised loss, and this would cause the “wrong labels” propagation.

Supervised learning task on DBpedia

Category classification (14 classes)

Method	Test error rate
Baseline (without embedding normalization)	0.87%
Baseline	0.90%
Random perturbation	0.85%
Adversarial	0.79%
Virtual Adversarial	**0.76**%
Bag-of-words[31]	3.57%
Large-CNN(character-level) [31]	1.73%
SA-LSTM(word-level)[4]	1.41%
N-grams TFIDF [31]	1.31%
SA-LSTM(character-level)[4]	1.19%

Conclusion

- Adversarial and virtual adversarial training are good regularizers for text classification tasks and achieved good performance.
- With tuning of the additional hyperparameter ε, we can improve over the baseline and achieve state of the art performance.