UNIDIRECTIONAL KL DIVERGENCE MAY BE BETTER FOR R-DROP *

Haoran Ye
2027407057
hrye@stu.suda.edu.cn

ABSTRACT

Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. R-Drop [1], one of the variations of Dropout, shows its promise for alleviating the inconsistency between training and inference phase. R-Drop minimizes the bidirectional KL-divergence between the output distributions of two submodels sampled by dropout, which seems a natural way to match two equally-treated distributions. However, in this report we[2] (1) **theoretically** show that unidirectional and bidirectional KL divergence loss are equivalent for R-Drop, and (2) **empirically** show that using unidirectional KL divergence loss **may** be better. Empirical results on IWSLT'14 German to English dataset show that implementing unidirectional KL divergence loss obtains significantly higher BLEU score at a 90% confidence level.

***K*eywords** Dropout · R-Drop · Regularization · KL-divergence · Neural translation · PyTorch

1 Introduction

Overfitting can occur when a model becomes too complex and starts to fit the noise in the data, rather than the underlying signal. This can lead to poor performance on unseen data. Regularization is a technique used in neural network training to prevent overfitting. Among a variety of regularization techniques [2, 3, 4], dropout [5] is known for its easy implementation, universal applicability and effectiveness. It addresses overfitting by randomly dropping out (i.e., setting to zero) output features of a layer during training. Therefore, it prevents complex co-adaptations in the network since the model cannot rely too heavily on any particular feature.

However, a potential issue that can arise when implementing dropout is an inconsistency between model training and inference [1, 6]. That is, the randomly sampled submodel during training phase is inconsistent with the full model used during inference phase. To eliminate this inconsistency, Liang et al. [1] propose R-Drop, forcing the distributions for the same data sample outputted by two submodels to be consistent. To this end, R-Drop minimizes the bidirectional Kullback-Leibler (KL) divergence between the two distributions.

KL divergence is non-symmetric. Hence, the bidirectional KL divergence seems a natural choice when two distributions outputted by two submodels are equally treated. However, we are motivated by the following observations and intuitions to test unidirectional KL divergence.

Motivations. Our motivations are as follows:

- Please see the training process from the perspective of *Law of Large Number*. The process repeatedly samples distributions for approximation purpose. Consider that we sample distribution p and q in this iteration, we could also sample q and p (note the order) in a future iteration. Singling out these two iterations, implementing R-Drop with unidirectional or bidirectional KL divergence loss is roughly equivalent.
- Implementing unidirectional KL divergence loss can be marginally more efficient.

*The third course project of *Application Practice of Deep Learning.*

[2]This report uses "we" by convention.

- To my knowledge, there is no relevant ablation study (specifically targeting the bidirectional design) in the original literature [1].

Contributions. We conclude our contributions as follows:

- We theoretically prove that implementing R-Drop with unidirectional or bidirectional KL divergence loss is equivalent.
- We empirically show that implementing R-Drop with unidirectional KL divergence loss is statistically significantly better (with 90% of confidence rate) on IWSLT'14 German to English dataset.

2 Preliminary

Dropout [5]. Dropout is a regularization technique for reducing overfitting in neural networks. It works by randomly dropping out, or setting to zero, a number of output features of the layer during training. This has the effect of making the model more robust and less sensitive to the specific weights of neurons. Dropout can be applied to input layers as well as hidden layers. It is most commonly used during training, but can also be used during inference to improve performance. Dropout is a simple and effective technique for improving the performance of neural networks on a wide range of tasks.

KL divergence. Kullback-Leibler (KL) divergence is a measure of the difference between two probability distributions. It is commonly used in machine learning and statistics to compare the difference between two probability distributions, such as the difference between the predicted distribution and the actual distribution. KL divergence is a non-symmetric measure, which means that the KL divergence between two distributions is not necessarily the same as the KL divergence between those two distributions in the opposite order. This measure is also known as relative entropy or information divergence.

R-Drop [1]. In R-Drop, each data sample in a mini-batch goes through a forward pass twice, with each pass being processed by a different submodel that has randomly dropped out some hidden units. Then R-Drop forces the two probability distributions for the same data sample output by the two submodels to be consistent with each other by minimizing the bidirectional KL divergence between them. This regularizes the outputs of the two submodels and helps to alleviate the inconsistency between the training and inference phases. R-Drop only adds a bidirectional KL divergence loss without any structural modifications compared to conventional dropout.

3 Theoretical Analysis

We analyze the equivalence of using unidirectional and bidirectional KL divergence loss in R-Drop. Following the orginal literature, we let $\xi^l \in \mathbb{R}^d$ denote a random vector. Each dimension of of ξ^l is independently sampled from a Bernoulli distribution $B(p)$:

$$\xi_i^l = \begin{cases} 1, & \text{with probability } p \\ 0, & \text{with probability } 1-p. \end{cases} \tag{1}$$

Then the dropout operation on $h^l(x)$ is represented by $h^l_{\xi^l}(x) = \frac{1}{p}\xi^l \odot h^l(x)$, where $\odot$ denotes the element-wise product. After applying dropout, the output distribution of a neural network w is denoted as $\mathcal{P}^w_\xi(y \mid x) := softmax\left(linear\left(h^L_{\xi^L}\left(\cdots\left(h^1_{\xi^1}\left(x_{\xi^0}\right)\right)\right)\right)\right)$, where $\xi = \left(\xi^L, \cdots, \xi^0\right)$. Implementing bidirectional KL divergence loss in R-Drop enforces the following constraint:

$$\frac{1}{2n}\sum_{i=1}^{n}\mathbb{E}_{\xi^{(1)},\xi^{(2)}}[\mathcal{D}_{KL}(\mathcal{P}^w_{\xi^{(1)}}(y_i \mid x_i)\,\|\mathcal{P}^w_{\xi^{(2)}}(y_i \mid x_i)) + \mathcal{D}_{KL}(\mathcal{P}^w_{\xi^{(2)}}(y_i \mid x_i)\,\|\mathcal{P}^w_{\xi^{(1)}}(y_i \mid x_i))] \le \epsilon, \tag{2}$$

where n is the size of training dataset; x_i and y_i are the ith training sample and ground truth label respectively; $D_{KL}(P_1\|P_2)$ is the KL divergence between two distributions P_1 and P_2; $\xi^{(1)}$ and $\xi^{(2)}$ are two i.i.d. ξ for two submodels respectively; ϵ is a very small real value.

Due to the additivity of the expectation, we have:

$$\frac{1}{2n}\sum_{i=1}^{n}\mathbb{E}_{\xi^{(1)},\xi^{(2)}}[\mathcal{D}_{KL}(\mathcal{P}^w_{\xi^{(1)}}(y_i \mid x_i)\,\|\mathcal{P}^w_{\xi^{(2)}}(y_i \mid x_i))] + \frac{1}{2n}\sum_{i=1}^{n}\mathbb{E}_{\xi^{(1)},\xi^{(2)}}[\mathcal{D}_{KL}(\mathcal{P}^w_{\xi^{(2)}}(y_i \mid x_i)\,\|\mathcal{P}^w_{\xi^{(1)}}(y_i \mid x_i))] \le \epsilon. \tag{3}$$

Now two terms are separate. Since $\xi^{(1)}$ and $\xi^{(2)}$ are i.i.d., we can reassign two symbols $\xi^{(1)}$ and $\xi^{(2)}$ to replace $\xi^{(2)}$ and $\xi^{(1)}$ in the second term. Hence:

$$\frac{1}{2n}\sum_{i=1}^{n}\mathbb{E}_{\xi^{(1)},\xi^{(2)}}[\mathcal{D}_{KL}(\mathcal{P}^{w}_{\xi^{(1)}}\,(y_i \mid x_i)\,\|\mathcal{P}^{w}_{\xi^{(2)}}(y_i \mid x_i))] + \frac{1}{2n}\sum_{i=1}^{n}\mathbb{E}_{\xi^{(1)},\xi^{(2)}}[\mathcal{D}_{KL}(\mathcal{P}^{w}_{\xi^{(1)}}\,(y_i \mid x_i)\,\|\mathcal{P}^{w}_{\xi^{(2)}}(y_i \mid x_i))] \leq \epsilon. \tag{4}$$

Two terms are now identical, therefore the enforced constraint can be rewritten as:

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\xi^{(1)},\xi^{(2)}}[\mathcal{D}_{KL}(\mathcal{P}^{w}_{\xi^{(1)}}\,(y_i \mid x_i)\,\|\mathcal{P}^{w}_{\xi^{(2)}}(y_i \mid x_i))] \leq \epsilon. \tag{5}$$

The above constraint is identical to that enforced by unidirectional KL divergence loss.

4 Empirical Results

Experimental setup. With unidirectional KL divergence loss and bidirectional KL divergence loss respectively, we conduct three trials each with a different random seed on IWSLT'14 German to English dataset. All other setup follows the original implementation[3].

Experimental environment. An NVIDIA RTX 3090 Graphics Card with 24GB graphics memory.

Logs. The training and inference logs can be found at: `https://github.com/YeHaoran2001/APDL/tree/main/Project3_logs`.

Results. The test results are shown in table 1. Implementing unidirectional KL divergence loss is marginally more efficient.

Statistical test. Non-parametric Wilcoxon signed rank test shows that p-value is 0.10, Cohen's d value is 1.489. The p-value indicates that implementing unidirectional KL divergence loss obtains significantly higher BLEU score at a 90% confidence level. The Cohen's d value indicates a large difference.

Table 1: Empirical results on IWSLT'14 German to English dataset. Uni: using R-Drop with unidirectional KL divergence loss. Bi: using R-Drop with bidirectional KL divergence loss. Avg. BLEU: average BLEU score of three trials. Avg. Duration: average training duration of three trials.

	Seed=63	Seed=64	Seed=65	Avg. BLEU	Avg. Duration
Uni	37.34	37.29	37.32	37.32	41735s
Bi	37.19	37.28	36.97	37.15	42719s

5 Conclusion

In this report, we theoretically and empirically show that implementing unidirectional KL divergence loss in R-Drop may be preferable to its bidirectional counterpart.

Limitation. Our experiments are limited by time and computation resources. More trials and testing on more datasets / tasks can make our proposition more reliable.

Future work. We look forward to addressing the limitations and expanding on this research with the authors initially propose R-Drop [1].

Acknowledgements

I am grateful to Prof. Li and TA Xia for their thoughtful suggestions, helpful guidance, and hopefully, upcoming high scores.

[3] `https://github.com/dropreg/R-Drop`

Appendix

A Typo in the Original R-Drop Paper

Eq. (6) in the orignal paper [1] is:

$$s.t. \quad \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\xi^{(1)},\xi^{(2)}}\left[\mathcal{D}_{KL}\left(\mathcal{P}^{w}_{\xi^{(1)}}\left(y_i \mid x_i\right)||\mathcal{P}^{w}_{\xi^{(2)}}\left(y_i \mid x_i\right)\right)\right] \leq \epsilon, \tag{6}$$

with an extra right parenthesis.

References

[1] Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al. R-drop: Regularized dropout for neural networks. *Advances in Neural Information Processing Systems*, 34:10890–10905, 2021.

[2] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722, 2019.

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[4] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066. PMLR, 2013.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[6] Xuezhe Ma, Yingkai Gao, Zhiting Hu, Yaoliang Yu, Yuntian Deng, and Eduard Hovy. Dropout with expectation-linear regularization. *arXiv preprint arXiv:1609.08017*, 2016.