Reproducibility Report for "DropMax: Adaptive Stochastic Softmax"

James Liu* Takashi Nagata* John English*

December 16, 2017

1 Introduction

1.1 Motivation for choosing DropMax: Adaptive Stochastic Softmax

Our group was interested in DropMax[1] because of its novel use of dropout regularization and variational inference. The combination of these two techniques, if proven effective, seemed like a valuable addition tool for deep learning. We saw DropMax as an opportunity to learn how to apply dropout regularization and variational inference and in what situations they would be useful. The following information entails our results when attempting to reproduce the results claimed by the OpenReview paper "DropMax: Adaptive Stochastic Softmax". We based our report on the metrics explained in the presentation by Joelle Pineau[2].

2 Method of Implementation

The formula for z_t is given by:

$$z_t = Sigm\left\{\frac{1}{\tau}\left(log\frac{\rho_t(x;\nu)}{1-\rho_t(x;\nu)} + log\frac{u}{1-u}\right)\right\} \tag{1}$$

ρ is only used in the loss function for the variational and regularization terms, so in our implementation we use

$$z_t = Sigm\left\{\frac{1}{\tau}\left(\rho_t^h(x;\nu) + log\frac{u}{1-u}\right)\right\} \tag{2}$$

where

$$\rho^h(x;\nu) = V^{\mathrm{T}} h_{L-1}(x) + c \tag{3}$$

*Department of Computer Science, University of California, Irvine

as inverting the sigmoid can cause numerical instability.
The authors used the formula

$$p(y|x,z;\omega) = \frac{exp(z_y o_y(x;\omega))}{\sum_t exp(z_t o_t(x;\omega))} \tag{4}$$

as their DropMax layer (dropping softmax logits). This contradicts their stated goal of dropping classes from consideration, since it assigns non-zero probability to all classes with $z_i = 0$. Our preferred change is to drop classes instead of class logits. This is done by using the formula

$$p(y|x,z;\omega) = \frac{z_y exp(o_y(x;\omega))}{\sum_t z_t exp(o_t(x;\omega))} + \epsilon \tag{5}$$

where ϵ is a small constant used to avoid infinite loss when the target class is dropped.
When dropping logits, the authors mention the necessity of regularizing ρ_t in order to keep z from being identically 1. This occurs because if o_t is negative for some non-target class, then $\frac{\partial \mathcal{L}}{\partial \rho_t}$ is positive, since dropping the logit for that class actually increases the predicted probability for that class. We confirm that this is the cause of failure experimentally by applying ReLU activation to o_t, and successfully avoid $z = 1$ without having to use regularization on ρ. However, our validation experiments show that regularization of ρ can still be helpful, even when dropping classes.
A minor point of ambiguity in their approximation equation

$$p(y|x^*,z;\omega) = \frac{exp(\rho_y(x^*;\theta) o_y(x^*;\omega))}{\sum_t exp(\rho_t(x^*;\theta) o_t(x^*;\omega))} \tag{6}$$

is that ρ_t is used in place of z_t, with the idea of setting $u = .5$, but the correct substitution should presumably be

$$\rho_t^* = Sigm(\frac{1}{\tau}\rho_t^h) \tag{7}$$

It wasn't clear to us how Q(y | x) was learned in sampled softmax, as the paper by Jean et. al[3] didn't seem to mention strategies for non-uniform Q.
We didn't implement the test-time sampling versions of DropMax, as they didn't seem to have any benefit over the approximate version.
We ran into some issues with the TensorFlow Sparsemax implementation and got poor results, which we weren't able to investigate.

3 Reproducibility metrics

3.1 Availability of datasets, partitioning information.

The authors of DropMax used four easily accessible datasets in the form of MNIST, CIFAR-10, CIFAR-100, and AwA. When describing MNIST the authors were very clear on how to partition the data when reproducing their experiments. They were also clear on how to partition CIFAR-10 and CIFAR-100. AwA, as of December 15, 2017, is no longer available due to a copyright dispute. This very negatively affects reproducibility, but the authors of DropMax were clear on the shape of AwA.

3.2 Availability of code, names and version numbers of dependencies.

In their first paragraph they stated they would make their code available after the paper was accepted. The paper itself only contains mathematical descriptions of their functions. They mentioned the fact that they use Keras[4] and TensorFlow[5], but they do not mention any dependencies or version numbers of libraries.

3.3 Availability of random seed and all hyperparameters.

In the experiments setup portion of the paper they gave inconsistent amounts of detail about their hyperparameters. There was more information about the setup for MNIST than for the other networks. Our main difficulty with reproducing their setup was interpreting the statement, "All other hyper-parameters are found with separate validation set." It wasn't clear if they were using the same validation set for all tuned hyperparameters, or what the other validation set was if they were using a different one. There were also a number of hyperparameters that they kept fixed, such as the temperature parameter τ and the sample sizes S used in training and test time. They also mentioned that S was set to 30 or 100 during test time, but didn't say which value they used for their benchmarks. For CIFAR-10, CIFAR-100, and AwA they did not give much detail beyond which base architecture was used. There was some guesswork necessary when we tried to replicate their results.

3.4 Alignment between the paper and the code.

For the DropMax paper no code or pseudo-code was given. However, we attempt to reproduce their experimental results using our own implementation and we compare the original DropMax results with our results below.

Table 1 (from paper): Test classification error (%). The three columns are results on MNIST dataset. The reported number is the median of 3 runs.

Models	1K	5K	55K
Base Network	7.17	2.19	0.84
Sampled Softmax (uniform Q)	7.48	2.17	0.91
Sampled Softmax (learned Q)	7.62	2.45	0.85
Sparsemax	6.84	2.28	0.82
Random-Dropmax (sampled)	7.25	2.45	0.80
Random-Dropmax (approx.)	7.26	2.51	0.78
Adaptive-Dropmax (sampled)	6.70	1.99	0.78
Adaptive-Dropmax (approx.)	6.66	1.99	0.78

Table 2 (our reproduction): Test classification error (%). The three columns are results on MNIST dataset. The reported number is the median of 3 runs.

Models	1k	5k	55k
Base Network	4.00	1.52	1.06
Sparsemax	23.02	1.75	1.29
Random-Dropmax (Drop Class)	3.57	1.73	0.77
Random-Dropmax (Drop Logit)	3.54	1.61	0.79
Adaptive-Dropmax (Drop Class)	3.04	1.36	0.56
Adaptive-Dropmax (Drop Logit)	2.97	1.44	0.63

3.5 Clarity of code & paper (ease of understanding, language).

The paper had some several grammatical errors but was generally coherent. One way they could have improved clarity would have been to include a network diagram of the various softmax alternatives.
The convergence plot was difficult to interpret because the base network test curve was mostly occluded by the adaptive-DropMax test curve.

3.6 Details of computing infrastructure used.

The paper did not mention the computing infrastructure that was used.

3.7 Computation requirements (time, memory, number/type of machines).

The MNIST hyperparameter search and benchmarks ran in about 24 hours on a GTX Titan. The other datasets could take significantly longer, depending on whether or not pre-trained models are used.

3.8 Reimplementation effort (time, expertise).

Our total time spent on this project was about 30 hours. Our team consists of three 1st-year graduate students.

3.9 Number and complexity of interactions with the authors.

We did not interact with the authors, as the submission was made anonymously, and we did not find any drafts on arXiv.

4 Conclusion

4.1 Experiment Design

The test classification error table appears to justify their conclusion that DropMax is especially beneficial in situations where the dataset is relatively small compared to the number of classes.
When reproducing the results, we didn't see a significant difference in performance when adding the variational term to the loss function. The paper would benefit from including a clear demonstration that the variational term benefitted the training process in some way or another. It's also likely that there should be a hyperparameter controlling the strength of variational loss.
Lastly, a section could be added analyzing the computational overhead needed for DropMax, in order to demonstrate that it's more beneficial than increasing the complexity of another component of the network. This is especially true for Adaptive DropMax, since it contains extra parameters, and requires the computation of additional loss functions, versus some of the other networks.

4.2 Our Findings

DropMax with dropped classes performs similarly as DropMax with dropped logits. We would still recommend the class-dropping version because it avoids the learning pathology described earlier. During validation we discovered that regularizing ρ can still be beneficial when dropping classes.
Our results seem to be in line with the original results, at least in terms of relative model performance.
Our code can be found at https://github.com/jamesal1/DropMax

4.3 Overall Opinion On Reproducibility

Overall the DropMax paper seemed to reasonably comply with the standards of reproducibility set out for this challenge. The data was very easily obtainable and its partitions were well defined. The DropMax paper mentioned what frameworks were used but did not give any code or pseudo-code. The DropMax paper's hyperparameter selection for the adaptive DropMax model was well stated, but was non-existent for the other models. Despite this lack of documentation, we were able to show the distinct improvement DropMax had over the base networks. The language of the DropMax paper was clear, but could have been made drastically more clear by including a diagram of the network the paper was proposing. The DropMax paper did not mention any of the computing hardware used. The runtime of the experiments was significant but reasonable for an academic research setting. Based on the above compliance with the criteria of reproducibility set forth by Joelle Pineau we believe that DropMax is adequately reproducible. We give it a 7/10 overall on reproducibility.

References

[1] Anonymous. (2017) Dropmax: Adaptive stochastic softmax. [Online]. Available: https://openreview.net/forum?id=Sy4c-3xRW&noteId=Sy4c-3xRW

[2] J. Pinaue. (2017) Reproducability in machine learning. [Online]. Available: https://d1b10bmlvqabco.cloudfront.net/attach/j6to9oa3hss262/irm4i83u8ap2hl/jalxtxbel8s2/ICLR2018NotesForClass.pptx

[3] R. M. Y. B. Sébastien Jean, Kyunghyun Cho. (2015, mar) On using very large target vocabulary for neural machine translation. [Online]. Available: https://arxiv.org/pdf/1412.2007.pdf

[4] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[5] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/