Brief Announcement: Gradual Learning of Deep Recurrent Neural Network

Ziv Aharoni(✉), Gal Rattner, and Haim Permuter

Ben-Gurion University, 8410501 Beer-Sheva, Israel
{zivah,rattner}@post.bgu.ac.il, haimp@bgu.ac.il

Abstract. Deep Recurrent Neural Networks (RNNs) achieve state-of-the-art results in many sequence-to-sequence modeling tasks. However, deep RNNs are difficult to train and tend to suffer from overfitting. Motivated by the Data Processing Inequality (DPI) we formulate the multi-layered network as a Markov chain, introducing a training method that comprises training the network gradually and using layer-wise gradient clipping. In total, we have found that applying our methods combined with previously introduced regularization and optimization methods resulted in improvement to the state-of-the-art architectures operating in language modeling tasks.

Keywords: Data-processing-inequality · Machine-learning
Recurrent-neural-networks · Regularization · Training-methods

1 Introduction

Several forms of deep Recurrent Neural Network (RNN) architectures, such as LSTM [7] and GRU [2], have achieved state-of-the-art results in many sequential classification tasks [3,5,6,14,15,17] during the past few years. The number of stacked RNN layers, i.e. the network depth, has key importance in extending the ability of the architecture to express more complex dynamic systems [1,12]. However, training deeper networks poses problems that are yet to be solved.

In this paper, we suggest an approach that breaks the optimization process into several learning phases. Each learning phase includes training an increasingly deeper architecture than the previous ones. In this way, we gradually train and extend the network depth, reducing the deleterious effects of degradation and backpropagation problems. Additionally, by adjusting the appropriate training scheme (mainly the regularization) at every learning phase, we are able to maximize the network performance even further.

2 Gradual Learning

2.1 Notation

Let us represent a network with l layers as a mapping from an input sequence $X \in \mathcal{X}$ to an output sequence $\hat{Y}_l \in \mathcal{Y}$ by $\hat{Y}_l = S_l \circ f_l \circ f_{l-1} \circ \cdots \circ f_1(X; \Theta_l)$,

I. Dinur et al. (Eds.): CSCML 2018, LNCS 10879, pp. 274–277, 2018.
https://doi.org/10.1007/978-3-319-94147-9_21

where the term $\Theta_l = \{\theta_1, \ldots, \theta_l, \theta_{S_l}\}$ denotes the network parameters, such that θ_k are the parameters of the k^{th} layer. We also define the l^{th} layer cost function by $J(\Theta_l) = \mathrm{cost}(\hat{Y}_l, Y)$, where $\theta^l = \{\theta_1, \ldots, \theta_l\}$. Next, we define the gradient vector with respect to $J(\Theta)$ by $\mathbf{g} = \frac{\partial}{\partial \Theta} J(\Theta)$, and the gradient vector of the k^{th} layer parameters with respect to $J(\Theta)$ by $\mathbf{g}_k = \frac{\partial}{\partial \theta_k} J(\Theta)$.

2.2 Theoretical Motivation

The structure of a neural network comprises a sequential processing scheme of its input. This structure constitutes the Markov chain $Y - X - T_1 - T_2 - \cdots - T_L$. The goal is to estimate $P_{Y|T_L}(y|t)$ by $Q^{\Theta}_{Y|T_L}(y|t)$. Driven by the Markov relation we state two theorems (without proofs due to space constraints).

Theorem 1 (Maximum Likelihood Estimator (MLE) and minimal negative log-likelihood). *Given a training set of N examples $S = \{(x_i, y_i)\}_{i=1}^{N}$ drawn i.i.d from an unknown distribution $P_{X,Y} = P_X P_{Y|X}$, the MLE of $Q^{\Theta}_{Y|T_L}$ is given by $P_{Y|X}$ and the optimal value of the criteria is $H(Y|X)$.*

Theorem 2. *If $Q^{\Theta}_{Y|T_L}$ satisfies the optimality conditions of Theorem 1, then $I(X;Y) = I(T_l;Y) \quad \forall l = 1, \ldots, L$.*

We show that by satisfying the optimality criteria of Theorem 1 we necessarily did not lose relevant information of Y by processing X to T_L. In particular, we show that a necessary condition to achieve the MLE is that the network states, namely $\{T_l\}_{l=1}^{L}$, will satisfy $I(Y;X) = I(Y;T_l)$.

2.3 Implementation

Due to the fact that shallow networks are easier to train, we propose a greedy training scheme, where we break the optimization process into L phases (as the number of layers), optimizing $J(\Theta_l)$ sequentially as l increases from 1 to L. The training scheme is depicted in Fig. 1.

3 Layer-Wise Gradient Clipping (LWGC)

Previous studies [4,8,13] have shown that *covariate shift* has a negative effect on the training process among deep neural architectures. *Covariate shift* is the change in a layer's input distribution during training, also manifested as *internal covariate shift*. We suggest that treating each layer weights' gradient vector individually and clipping the gradients vector layer-wise can reduce *internal covariate shift* significantly. LWGC for a network with L different layers is formulated as

$$\left[\hat{\mathbf{g}}_1^T, \ldots, \hat{\mathbf{g}}_L^T\right]^T := \left[\frac{\mu_1}{\max(\mu_1, \|\mathbf{g}_1\|)}\mathbf{g}_1^T, \ldots, \frac{\mu_N}{\max(\mu_N, \|\mathbf{g}_N\|)}\mathbf{g}_N^T\right]^T. \quad (1)$$

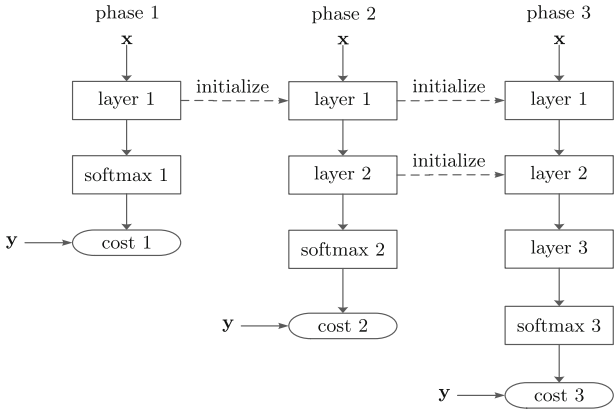

Fig. 1. Depiction of our training scheme for a 3 layered network. At phase 1 we optimize the parameters of layer 1 according to cost 1. At phase 2, we add layer 2 to the network, and then we optimize the parameters of layers 1, 2, when layer 1 is copied from phase 1 and layer 2 is initialized randomly. At phase 3, we add layer 3 to the network, and then we optimize all of the network's parameters, when layers 1, 2 are copied from phase 2 and layer 3 is initialized randomly.

4 Experiments

We present results on a dataset from the field of natural language processing, the PTB, conducted as a word-level dataset.

We conducted two models in our experiments, a *reference* model and a *GL-LWGC LSTM* model that was used to check the performance of our methods. Our GL-LWGC LSTM model compared the state-of-the-art results with only two layers and 19M parameters, and achieved state-of-the-art results with the third layer phase. Results of the *reference* model and *GL-LWGC LSTM* model are shown in Table 1.

Table 1. Single model validation and test perplexity of the PTB dataset

Model	Size	Valid	Test
Zoph and Le [18] - NAS	25M	-	64.0
Melis et al. [10] - 2-layer skip connection LSTM	24M	60.9	58.3
Merity et al. [11] - AWD-LSTM	24M	60.0	57.3
Yang et al. [16] - AWD-LSTM-MoS + finetune	22M	56.54	54.44
Ours - 2-layers GL-LWGC-AWD-MoS-LSTM + finetune	19M	55.18	53.54
Ours - GL-LWGC-AWD-MoS-LSTM + finetune	26M	**54.24**	**52.57**
Krause et al. [9] AWD-LSTM + dynamic evaluation	24M	51.6	51.1
Yang et al. [16] AWD-LSTM-MoS + dynamic evaluation	22M	48.33	47.69
Ours - GL-LWGC-AWD-MoS-LSTM + dynamic evaluation	26M	**46.64**	**46.34**

References

1. Bianchini, M., Scarselli, F.: On the complexity of neural network classifiers: a comparison between shallow and deep architectures. IEEE Trans. Neural Netw. Learn. Syst. (2014)
2. Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: encoder-decoder approaches (2014). arXiv preprint arXiv:1409.1259
3. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN Encoder-Decoder for statistical machine translation (2014). arXiv preprint arXiv:1406.107
4. Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, Ç., Courville, A.: Recurrent batch normalization (2016). arXiv preprint arXiv:1603.09025
5. Ha, D., Dai, A., Le, Q.V.: Hypernetworks (2016). arXiv preprint arXiv:1609.09106
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015). arXiv preprint arXiv:1512.03385
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
8. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift (2015). arXiv preprint arXiv:1502.03167
9. Krause, B., Kahembwe, E., Murray, I., Renals, S.: Dynamic evaluation of neural sequence models (2017). arXiv preprint arXiv:1709.07432
10. Melis, G., Dyer, C., Blunsom, P.: On the State of the Art of Evaluation in Neural Language Models. ArXiv e-prints, July 2017
11. Merity, S., Shirish Keskar, N., Socher, R.: Regularizing and Optimizing LSTM Language Models. ArXiv e-prints, August 2017
12. Montufar, G., Pascanu, R., Cho, K., Bengio, Y.: On the number of linear regions of deep neural networks (2014). arXiv preprint arXiv:1402.1869
13. Shimodaira, H.: Improving predictive inference under covariate shift by weighting the log-likelihood function. J. Stat. Plann. Infer. **90**(2), 227–244 (2000)
14. Smith, L.N., Hand, E.M., Doster, T.: Gradual dropin of layers to train very deep neural networks (2015). arXiv preprint arXiv:1511.06951
15. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on International Conference on Machine Learning, ICML 2013, vol. 28, pp. III-1139–III-1147 (2013). JMLR.org
16. Yang, Z., Dai, Z., Salakhutdinov, R., Cohen, W.W.: Breaking the softmax bottleneck: a high-rank RNN language model (2017). arXiv preprint arXiv:1711.03953
17. Zilly, J.G., Srivastava, R.K., Koutník, J., Schmidhuber, J.: Recurrent highway networks (2016). arXiv preprint arXiv:1607.03474
18. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning (2016). arXiv preprint arXiv:1611.01578