Improving adversarial synthetic data generation using synaptic plasticity

Diwakar Ravichandran
Amtrak Technologies Pvt. Ltd
diwakar@amtrak.co.in

Kumari Deepshikha
NVIDIA Graphics Pvt. Ltd.
dkumari@nvidia.com

Abstract

Modeled after the human brain, neural networks have motivated researchers to continually mimic their workings. Evolving rapidly, neural networks quickly have become the most effective when dealing with tasks involving learning. The evolution of learning tasks has led to deep reinforcement learning and deep fakes. Policy gradients have been the go-to mechanism for Deep Reinforcement Learning, where the major task is to provide a concrete goal to the agent, to maximize the total rewards. In this regard, there have been numerous methods that constitute such a gradient. Hebbian learning mechanisms, capitalizing on Hebb's law, arrive at a similar crux where the neural chain at the output node is strengthened based on how many similar activations of previous nodes can lead to the bright activations at the corresponding output nodes. Armed with experiential learning, we have synaptic plasticity to act as a complement to the reinforcing agent in an RL based neural network. Further, provision of anticipation is a sought-after module in deep neural networks. With Hebbian learning, we enhance the networks' anticipatory trajectories in its synapse, hence accelerating classification problems. Based on the above principle, applying such neural plasticity to GANs we have an upset in the balance of the two-player min-max game. By unnaturally reinforcing the discriminator(D) of the GAN, we force the generator(G) to improve its performance. Moreover, this also enables us to add more complications to the generator(G) which consequently leads to better synthetic data being generated

1. Introduction

Synthetic data generation akin to real data is of great significance, especially to the unsupervised learning setting. We have had recurrent neural networks(RNNs) with long short-term memory(LSTMs) blocks in it show promising results when it came to sequence generation. The proposed strategy of Scheduled Sampling(SS) by Bengio *et al.* proved to be an insight into feeding synthetic data into your network at the earlier stages, and further down, providing the system with real valued data. This however fails due to its inconsistency and does not solve its intrinsic problem[]. The next proposed step forward was to use a loss function that was built on the entire sequence as opposed to each individual transition. Down the line, generative adversarial networks (GANs) as proposed by Goodfellow *et al.* have the track record of generating images, streams of images and sequences. Other methods that were employed for the task of sequence generation included expert systems *et al.* who based on a partially separated reward function using reinforcement learning (RL) were able to obtain state of the art results when it came to sequence generation. SeqGAN[22] was able to tackle the issues of the GAN being used to generate sequences overcoming the pitfalls of the back-propagation algorithm in context of discrete tokens being used. The work published by Miconi *et al.* [13], demonstrated the use of a novel reinforcement mechanism in which it is observed that the model is tuned and reformed in order to incorporate synaptic plasticity in the model. Mimicking the human brain based on the work by Martin *et al.* [11]and Liu *et al.* [10] and modelling the learning mechanism after Hebb's law (reviewed shortly): if a neuron is repeatedly fired resulting in a similar firing in a subsequent neuron, the connection between the two neurons is strengthened. While there have been previous attempts to recreate a plastic network [4], they don't give a comprehensive enough picture of how to be incorporated into deep learning models. Enabling traditional optimizing algorithms to optimize the plastic layer of the network is a huge step forward in the direction of RL. GANs have shown enormous potential when it has come to generation of synthetic data. In context of sequences it becomes very necessary to note the use of RL in synthetic data generation. With the advent of RL being applied to GANs it provides a great opportunity for a GAN to be incorporated with plastic layers in order for the GAN to take full advantage of the mathematics that it was built on and consequently generate better synthetic data. In this paper we attempt to incorporate Hebbian learning to sequence generation, which resulted in a boost of performance by 37.328% as opposed to the original network.

2. Related work

There has been a wide use of RL for training sequence generators, where policy gradients have proved to be useful[22, 20]. BLEU scores have been used as a task specific reward, and generators attempt to maximize this score. Methods of training RL based models for text generation focus on one of the following two assumptions, that the task specific reward is given or that the task specific reward is totally unavailable. When using RL to train the generator, we have the log likelihood of the reward RNN in the back end and a task specific reward as metrics using MLE. TextGAN[24] pioneering the use of GANs for text generation used RNNs with LSTM[8] as generator and used CNNs[23] as discriminator, and a CNN based discriminator encouraging the model to learn representations which are both informative to the original sentences whilst being discriminative towards synthetic sentences. It capitalizes on the use of Maximum Mean Discrepancy(MMD)[] as a part of the overall loss function. SeqGAN[22] was introduced which provided a better insight into the reaches of GAN based models which could bring competitive results to the ground truth proving the mettle of GANs regarding text generation. Parallel studies have also shown efficacy in context of expert-based systems for the task of text generation coupled with RL[20].

2.1. SeqGAN

SeqGAN inherently solves GANs problem of generating discrete tokens of sequential data[9] by following Bachman and Precup 2015[2] and Bahdanau *et al.* [3] and consider the sequence generation as a sequential decision-making process. The RL agent is the generator model. With RL being applied to the generator, the discriminator is tasked with evaluation of the generation of sequences for the improvement of the generator. A major tackling is introduced in the form a stochastic parameterised policy in order to pass back gradients to the generative model. Policy gradients[19] are employed to assist the RL process, in which Monte Carlo (MC)[18] search is employed to approximate the state-action value and avoids the inherent difficulties of differentiation for discrete data as in a conventional GAN. Starting with pre-training the generator using Maximum Likelihood Estimation (MLE) on the sequence, followed by which the discriminator is pretrained. Following which the model is trained in adversarial fashion until convergence based on the following equation.

$$min_{\varphi} - E_{Y\sim p_{data}}\left[log\, D_{\varphi}(Y)\right] - E_{Y\sim G_{\theta}}\left[log(1-D_{\varphi}(Y))\right] \tag{1}$$

2.2. Fast Associative Memory

In sequential learning where it isn't clear to understand how a real brain would implement the tasks involving the learning, various methods were proposed that progressively showed associative, mapping and other simple functions to understand the workings of the brain[6, 21, 7]. Fast memory association as described by Ba *et al.* [1] was a novel feature addition to the dynamic component of the strength of each synapse of the network. Thus, allowing the network to continually attend to the recent past and hence to ensure a weighted parameterization is included into the model in context of previous outputs from the previous neurons are fed into the subsequent neural layers, there by establishing a sustainable boundary condition. This however limits the network to the recent patterns that are observed by the network and does not add much value when looking for lifelong learning.

2.3. Differentiable plasticity

In the case of RL, we employ agents who are rewarded or penalized based on their performance during training. A lot of work has been put into how one goes about the problem of learning to learn. A basic method would be to employ RNNs which if tuned to take in data with a proper training schedule, the RNN would be able to learn novel information during each episode of information that it is exposed to [6, 17, 14]. A fairly unexplored approach to this learning mechanism is to augment a plastic synaptic component to the already existing mechanism which would grow and decay as a part of the inherent mechanism in itself. When trained, only the non-plastic part of the network model is focussed upon. However, Miconi *et al.* [13, 12] were able to effectuate the trainability of plastic components of the network enabling experiential learning for the model and to pave way into learning using self-modifying networks. Governed by principled equations, augmentation of a plastic synaptic component most likely is in the form of Hebbian learning, where Hebb's rule is mathematically represented as an equation.

$$x_j(t) = \sigma\left\{\sum_{i\in inputs}\left[w_{i,j}x_i(t-1)\right] + \alpha_{i,j}Hebb_{i,j}x_i(t-1)]\right\} \tag{2}$$

$$Hebb_{i,j}(t+1) = \eta x_i(t-1)x_j(t) + (1-\eta)Hebb_{i,j}(t) \tag{3}$$

With the Hebbian trace initialized to 0 at every iteration, and with α being learned over the duration of the training and η being the learning rate, when implemented into a neural network architecture stands to prove the ability of networks to adapt to experiential learning and to enable neural networks to have a different type of reward function adapted to its agent. Governed by equation 2, we have the theoretical backing as explained by Miconi *et al.* [12] which serves as a good headway into this concept of a Hebbain GAN where we try to capitalize on the benefits of both methods.

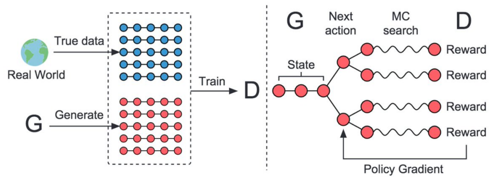

Figure 1. SeqGAN [22]

3. Model details

The model used for experimentation is one that is based out of the original SeqGAN. However, the dynamics of the model differ in their working due to the change in its layers a schematic diagram of the model is as shown. The model is latched with a Hebbian plastic layer at the discriminator, which causes the network to become self modifying in nature. This results in the model begin to perform very well on the the evaluation metric. The model diagram is as shown in Figure1.

3.1. Generator

The generator used comprises of RNNs with LSTMs[8] modelled after Yu *et al.* [22]. The RNN acts as a mapping function to map the input embedding representations to a state of hidden states similar to that of a dense layer. Upon activation after parameterization helps the network map the hidden states to discrete tokens. Further the LSTM is employed to tackle the problem of vanishing and exploding gradients. Gated recurrent units[5] (GRU) are a good alternative to LSTMs.

3.2. Discriminator

Zhang and LeCun [23] have proven the efficacy of CNNs for the task of text classification. However this is limited to complete texts. In our model we follow suite of the discriminator of SeqGAN [22] however taking into consideration the regularization effects of the dropout layer [15] as being a deterrent to the network for it to converge in the presence of the synaptic layer [13] we have deferred from the original model and gone ahead and omitted the dropout layer while retaining the highway layer [16]. In our model the plasticity was introduced at 2 different locations which resulted in different performance boosts. Incorporating plasticity during softmax yeilded best results that are claimed in this paper. However when plasticity was introduced to other parts in the network, does not yield a significant performance boost when compared to the existing model. This change has helped bring about very desirable effects to the network, causing the performance of the native network to drastically improve.

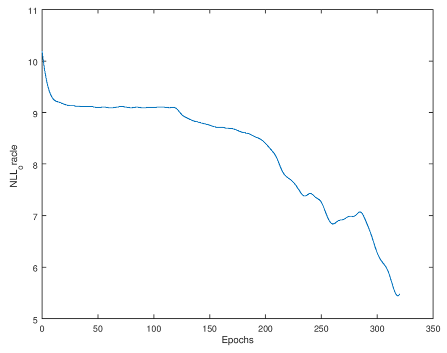

Figure 2. Plot

4. Methodology

4.1. Experiments and Results

Majority of the experiments that were carried out pertained to the use of Hebbian learning and synaptic plasticity in order to improve the performance of GANs and subsequently lead to better synthetic data generation. In this regard the efficacy of the model is tested by conducting a simulated sequence generation where we use a randomly initialized LSTM as the ground truth, referred to as oracle, to generate real data distributions $p(x_t|x_1..., x_{t-1}$. As described by Yu *et al.* [22] the benefit of having such a model is that it serves as an exact performance measuring tool for generative models. Given by the equation shown, the NLL_{oracle} serves as the best evaluation metric when looking at the simplicity and the efficacy of the network. The following results are courtesy [20] where PG refers to the use of policy gradients, S and SL refer to the type of reward function used. Our model's results outperform the results of both expert based text generation mechanism as well as adversarial methods.

$$NLL_{oracle} = -E_{Y_{1:T \sim G_\theta}}[\sum_{t=1}^{T} logG_{oracle}(y_t|Y_{1:t-1})] \quad (4)$$

5. Conclusion and future work

In this paper we propose a novel addition to the existing framework of adversarial data generation, providing a trend setting opportunity to drastically improve the working of a GAN by introducing synaptic plasticity to the network and equipping the discriminator(D) of the GAN to participate in a continuous learning process that complements its already existing ability to classify if a given data point is real or counterfeit. In future, we plan to incorporate Hebbian learning in the generator of the network as well, with the intuition that with enough regularization, the generator would generate synthetic data of higher quality.

Algorithm	Type of Algorithm	NLL_{oracle}
Random	Generic	10.310
MLE	Generic	9.038
SS	Generic	8.985
PG-BLEU	Generic	8.946
SeqGAN (PG_SL)	Adversarial	8.736
PG_S	Adversarial	8.33
PG_SL	Adversarial	8.58
PG_L	Expert_based	7.50
PG_S	Expert_based	6.01
PG_SL	Expert_based	6.69
Ours	**Adversarial**	**5.476**

Table 1. Results

References

[1] Jimmy Ba, Geoffrey Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past, 2016. 2

[2] Philip Bachman and Doina Precup. Data generation as sequential decision making, 2015. 2

[3] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction, 2016. 2

[4] Y. Bengio, S. Bengio, and J. Cloutier. Learning a synaptic learning rule, July 1991. 1

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. 3

[6] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines, 2014. 2

[7] Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory, 2015. 2

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory, Nov. 1997. 2, 3

[9] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary?, 2015. 2

[10] Liu, X. Ramirez, S. Pang, P. Puryear, C. Govindarajan, A. Deisseroth, and K. Tonegawa. Optogenetic stimulation of a hippocampal engram activates fear memory recall, 01 2012. 1

[11] Stephen Martin, P.D. Grimwood, and Richard Morris. Synaptic plasticity and memory: An evaluation of the hypothesis, 02 2000. 1

[12] Thomas Miconi. Learning to learn with backpropagation of hebbian plasticity, 2016. 2

[13] Thomas Miconi, Jeff Clune, and Kenneth O. Stanley. Differentiable plasticity: training plastic neural networks with backpropagation, 2018. 1, 2, 3

[14] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. One-shot learning with memory-augmented neural networks, 2016. 2

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting, 2014. 3

[16] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks, 2015. 3

[17] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks, 2015. 2

[18] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. 2

[19] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation, 1999. 2

[20] Joji Toyama, Yusuke Iwasawa, Kotaro Nakayama, and Yutaka Matsuo. Toward learning better metrics for sequence generation training with policy gradient, 2018. 2, 3

[21] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks, 2014. 2

[22] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient, 2016. 1, 2, 3

[23] Xiang Zhang and Yann LeCun. Text understanding from scratch, 2015. 2, 3

[24] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation, 2017. 2