Pruning Neural Networks with Lottery Tickets in a MDP Approach

Andrey de Aguiar Salvi
Machine Intelligence and Robotics Research Group
School of Technology
Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS
Porto Alegre, RS, Brazil

Abstract

With the growing size from neural networks, the work of prune these models emerged in the literature to create reduced models in order to allow its computation quickly or possible in devices with less computational power as mobile devices. In this proposal, we aim to improve the Lottery Tickets Hypothesis (LTH), a compression method that works on the trade-off between compression and accuracy, by the use of Markov Decision Process (MDP).

Artificial neural networks are the state of the art on computer vision tasks, despite the fact of creating giant models with many parameters. For example, the work of Simonyan and Zisserman creates a model that wins the ImageNet Challenge 2014 with 144 million of weights. The various possible ways of combine convolutions filters with different sizes and many other details allowed to create models with fewer weights and less computation, i.e the works (Szegedy et al. 2015), (He et al. 2016), (Szegedy et al. 2017), etc. Even so, none of the models we present have less than 10 million of weights.

This high computational cost makes it necessary to use GPUs for real-time computer vision and makes it impracticable to use these models on CPUs or mobile devices. With these problems in mind, many authors have worked to find ways to compress these models for some purposes such as reducing the number of operations per second, or just reducing the size of storage required, or even reducing the energy required to use the model on mobile or embedded devices, depending on the need for the problem to be attacked.

Consequently, many different forms of model compression with different purposes were created in the state of the art. For example, Han, Mao, and Dally has created a method for compressing convolutional networks with the main purpose of reducing the model's disk storage. Han et al. has created a compression technique to reduce energy consumption to run the model on mobile devices. There are also in the literature some techniques that use automated planning for model compression. For example, Ashok et al. use policy gradient reinforcement learning to compress models. Another example is the work of Zhang et al., which creates a reinforcement learning framework with Markov Decision Process (MDP) to remove layers from a DenseNet model.

One work of the state of the art on model compression aimed at maximum compression with minimal loss in accuracy is the work of Frankle and Carbin. The method employed also utilizes an iterative process of training and pruning connections.

The objective of this work is to investigate the possibility of generating a model compression algorithm based on Frankle and Carbin using an MDP. The rest of our work goes as follows: in the section Technical Approach we will explain in more detail about MDP, about the work from Frankle and Carbin and our proposed method. In the section Proposed Method, we will describe the current phase of our work and present a schedule of future work and finally, we will present the results we hope to achieve in section Conclusions.

Technical Approach

Here we will describe the main concepts, the Lottery Tickets Hypothesis and the Markov Decision Process, and after the proposed method.

Lottery Tickets Hypothesis

A layer of a fully connected neural network can be given by the equation below

$$\sigma(x) \leftarrow f(W^T x) \tag{1}$$

where x is the input vector, W is the weight matrix, $f()$ is an activation function and σ is the output computed. In the lottery thickets (Frankle and Carbin 2018), the forward computation is given by

$$\sigma(x) \leftarrow f((W' \odot W)^T x) \tag{2}$$

where W' is a binary mask (initialized with ones) that holds or prunes the weights, having the same shape as W and $\odot$ is the Haddamart Product. Given a pruning rate p, in the LTH, the $p\%$ remaining weights with lower magnitude are pruned, this means $W'_{i,j} = 0$.

The model is trained with the weight matrix W in the dataset composed by inputs X and labels Y for some epochs. After this, in an iterative process, the model is pruned, the weights from W are rewarded to the initial state and the model has trained again, until only $p\%$ of the weights remain on the model.

Markov Decision Process

Markov Decision Process (MDP) is a non-deterministic state-transition system where the probability distribution of the next state depends only on the current state (Ghallab, Nau, and Traverso 2016). Given a set of states S, a set of actions A, the state transition function $\gamma : S \times A$, we have $P_a(s, s')$ as the probability that action a in state s will lead to state s' and $R_a(s, s')$ as the immediate reward received after transitioning from state s s s to state s' with the action a.

In the cases where the probabilities or rewards are unknown, we have a problem of Reinforcement Learning, and the MDP must be solved with Q-Learning. In Q-Learning we have a table called Q, where rows are indexed by states and columns are indexed by actions. Q table is initialized with zeros and is incrementally updated on $Q(s, a)$ by the equation

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a) + \max_{a'} \{Q(s', a')\} - Q(s,a)] \tag{3}$$

where the rewards $r(s, a)$ are unknown and the value of Q in s' is observed in the step of the trial (Ghallab, Nau, and Traverso 2016). $\alpha \in [0, 1]$ is the learning rate and measure how important are the actual rewards over the old ones. With an state s, the action with higher value in Q table is choose to create the new state s' (sometimes with a little probability, a random action is chosen to perform exploration) and the equation 3 is performed iteratively until some goal is reached.

The Proposed Model

We propose in this work to modify the method from Frankle and Carbin, changing the way of choosing the weights to be pruned by a pretrained MDP with Q-Learning. This means that in method from the subsection Markov Decision Process, s will represent a mask W' used in LTH, the action a will delete a row from the weight matrix W that has not already been removed and the reward will be the accuracy obtained by the pruned neural network after the action.

Project Management

We already have an implementation of the LTH (in Python using Pytorch) and the implementation of the MDP is in the final step, missing the lasts steps of debugging. We make a schedule with weakly goals for accomplishing this work, that we describe below:

- Week 1 (24/10): the MDP will be completed and all the tests will be performed.
- Weeks 2, 3 and 4 (31/10, 07/11 and 14/11): we will execute the MDP to find the best way of prune a LeNet 300-100 trained on MNIST dataset because this is the same configuration created on the work of Frankle and Carbin. We will perform some tests to learn the influence of the hyperparameters and find the best combination, as the learning rate α, the discount factor γ, the probability of the agent to randomly choose an action and also the way to compute the reward function.
- Week 5 (21/11): we will analyze the results obtained, write the second version of the paper and create slides for the last presentation.

Conclusions

In this work, we aim to investigate the influence in use MDP to find best weights to prune instead of the traditional way from LTH. An improvement in the LTH could be possible since in the LTH the choice of weights to be pruned does not take into account the temporal influence of the removed weights, ie, the method does not make decisions now aiming at possible future rewards. Learning about current actions that optimize future rewards is the core of an MDP.

Moreover, MDPs are automatic planning algorithms, which means that we are trying to find a policy that is a universal problem solver (Ghallab, Nau, and Traverso 2016). With this in mind, perhaps the pruning mask found by MDP can be reused on another model with the same architecture but trained on a different dataset. If this will be confirmed, the pruning process can be reused, drastically reducing the high cost required for the pruning process.

References

Ashok, A.; Rhinehart, N.; Beainy, F.; and Kitani, K. M. 2017. N2n learning: Network to network compression via policy gradient reinforcement learning.

Frankle, J., and Carbin, M. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks.

Ghallab, M.; Nau, D.; and Traverso, P. 2016. *Automated planning and acting*. Cambridge University Press.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc. 1135–1143.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning.

Zhang, X.; liu, H.; Zhu, Z.; and Xu, Z. 2019. Learning to search efficient densenet with layer-wise pruning.