Learning to Reason

PhD Thesis Proposal

Wojciech Zaremba WOJ.ZAREMBA@GMAIL.COM

Advisors:
Rob Fergus FERGUS@CS.NYU.EDU
Yann LeCun YANN@CS.NYU.EDU
New York University

Abstract

Neural networks proven to be a very powerful models for object recognition (Krizhevsky et al., 2012), natural language processing (Mikolov, 2012), speech recognition (Graves et al., 2013), and many others (Sutskever et al., 2014). However, there is still a huge gap between them, and an intelligent systems. I identify several potential unaddressed skills, which intelligent systems should possess: (1) reasoning abilities, (2) capability to integrate with external interfaces and (3) small sample complexity. My research focuses on tackling this problems.

1. Introduction

It's clear that by improving performance of current statistical learning systems, we won't be able to make them intelligent. Even if our object recognition system would yield 0% of prediction error, they wouldn't be intelligent. Same applies to speech recognition systems, machine translation and others. This work asks what skills are necessary for statistical learning system to become "intelligent". Moreover, it attempts to address this remaining unaddressed skills.

I believe, that crucial, poorly addressed skills that intelligent system has to poses are (1) reasoning abilities, (2) capability to integrate with external interfaces and (3) small sample complexity. I would like to address all this problems within a seamless system. The same system should be used across different tasks to verify its empirical universality.

I have partially addressed some of proposed problems. I will make clear over the further part of this proposal, which parts have been addressed, and which are future goals.

2. Reasoning abilities

Reasoning - "the process of forming conclusions, judgments, or inferences from facts or premises"*.

System that can reason should be able to understand high level concepts like

- scope
- being able to instantly memorize
- conditioning
- nesting
- repetitions
- and many more

Each of this skills could be solved in particular domain. However, solving problem in a single domain limits use of such system, and assumes that we know a priori all possible scenarios. Moreover, I think that, intelligent reasoning system cannot be based only on predefined rules. Intelligent system has to be based on pattern matching, and application of learnt heuristic algorithms.

There are many domains where we can test if our system can reason, and if it's able to learn postulated concepts. Eventually, I would like to use the same tools for all domains. Domains of my interest are learning about computer programs, and proving mathematical theorems. This domains are rich in scoping, branching, nesting, pattern repetition, and so on. Drawing high level conclusions in such domains requires sophisticated reasoning skills.

*Definition from http://dictionary.reference.com/browse/reasoning

Input:

```
f=(8794 if 8887<9713 else (3*8334))
print((f+574))
```

Target: 9368.
Model prediction: 9368.

Figure 1. An exemplary program that we take as an input. Task is to predict program evaluation. We have achieved high performance on this task (Zaremba & Sutskever, 2014).

2.1. Reasoning in computer programs

The first reasoning task that I am interest in is to train statistical models meaning of computer programs. I propose one instance of program understanding. Task is to take a program as an input (in our case character-by-character), and predict program evaluation. Prediction of program evaluation requires understanding every single operand of a program. For instance, in case of addition it involves bit shifts, and memorization of operations on digits. Moreover, programs contain variable assignment, if-statements, and so on. We were able partially to address postulated problem (Zaremba & Sutskever, 2014). Figure 1 shows an exemplary program, target for such program, and prediction that we obtained with recurrent neural network.

Current results are promising, however they are very limited. We are able to deal with programs that can be evaluated by reading them once from left-to-right. Generic programs are not like that. Our reasoning system has to be able to evaluate for arbitrary long time, if task requires it. Moreover, it shouldn't be limited by finite memory size. Memory should be available as an interface (Section 3).

2.2. Reasoning in mathematics

It's known that theorem proving is an intractable task in computational sense. However, humans are able to prove theorems. They have to employ a-prior knowledge. They fit known mathematical "tricks" to the new problem.

More formally, we can think about proofs as a multiple axioms application that starts with hypothesis that we want to prove. This way mathematical skills can be perceived as learning prior over trees of axioms. This prior "suggests" us which sequence of axioms is more likely to lead to the theorem proof.

My interest lies in training statistical machine learning system to learn such priors. We have started with very constrained mathematical domain. We learn here prior over mathematical identities. It is not even a prior over proofs.

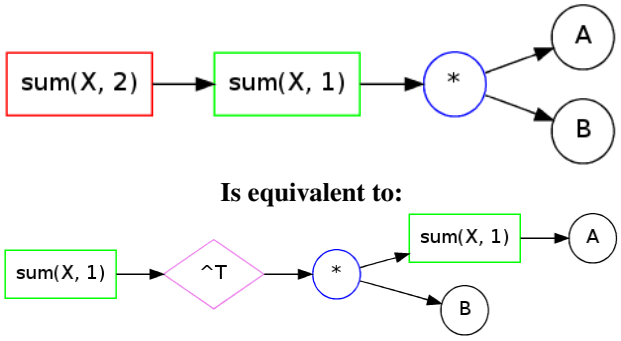

Figure 2. Example of two equivalent mathematical expressions. It states that $\sum_{i,j}\sum_k A_{i,k}B_{k,j} = \sum_j\sum_k(\sum_i A_{i,k})B_{k,j})$ (Zaremba et al., 2014). This is a toy example, and cited paper contains more sophisticated examples.

Our recent work (Zaremba et al., 2014) concerns identities over polynomials over matrices. Figure 2 shows example of a simple identity over matrix polynomials. It states that, $\sum_{i,j}\sum_k A_{i,k}B_{k,j} = \sum_j\sum_k(\sum_i A_{i,k})B_{k,j})$. This an example of toy identity. Our system is able to derive much more sophisticated identities.

3. Interface learning

Neural networks are a very powerful statistical models, however some concepts might be very difficult for them to express. For instance, nesting seems to be a very common operation, that allows to deal with (1) object hierarchy, (2) natural language parsing, and (3) compositionality. It would be best to provide stack as an external interface to the neural network. Neural network should be able to learn how to use such interface.

External interfaces could enhance neural networks capabilities, and also provide access to external resources, and datasets. There are few other interfaces that could be very useful:

- stack
- FIFO
- an external memory (Weston et al., 2014; Graves et al., 2014)
- search engine
- database
- execution environment like python interpreter

State space of many interfaces is massive (e.g. all possible search queries). Moreover, external interfaces are not-differentiable, This obstacles could potentially be addressed by learning a differentiable model that describes

them (e.g. neural network). Neural network would simulate such external interface for purpose of being differentiable. Some of work in area of interface learning is under progress. It is one of my major projects now.

4. Small sample complexity

Current deep learning systems suffer of large sample complexity. Such high sample complexity hinders potential use of the systems in online learning setting (e.g. robots). Its expected that during the initial phase of learning any system without prior knowledge would need to consume a large number of samples. We hope that over the time of training, sample complexity should drop. However, this is not observed in a current systems. I propose several approaches how sample complexity can be decreased. Meta-optimizer (Subsection 4.4) is the target solution. Moreover, I propose some intermediate solutions, that can potentially decrease sample complexity.

4.1. Automatic derivation of better models

Neural networks are designed ad-hoc. People have only some partial intuition regarding how to build high prediction performance systems. One way of addressing this issue would be to fully understand why, and how models work. However, it is unclear if it is even possible (maybe best explanation is a model itself). I would like to automatically explore large space of models, and evaluate them on various tasks. This involves connectivity, activation functions, and hyper-parameters. I would optimize it with genetic programming. This work is under progress. It is one of my major projects now.

4.2. Augmentation marginalization

For simplicity of explanation, we focus on computer vision tasks.

We often train neural networks for multiple epochs, while the same samples are presented multiple times. The best performance is achieved when samples are slightly modified over the course of training. In case of computer vision, it is done by jittering images, rotating them, and so on. This is known as data-augmentation. Theoretically, a full image contains information about all updates obtained with all possible augmented samples. More formally, let x be an image, y its label and $x_1, x_2 \dots x_k$, are its augmented copies. L is a loss function, and θ_i are parameters at step i. I would like to predict with x sum of derivatives for all data augmentations:

$$f(x, \theta_1) = \sum_{i=1}^{k} \partial_\theta L(x_k, y, \theta_i) \tag{1}$$

Instead of following gradient of $\partial_\theta L(x, y, \theta_1)$, we could follow $f(x, \theta_1)$. This could improve learning speed, as a single sample would provide information about it's all augmented copies.

Function f might have its own parameters, and I would like to model it as a neural network. We could train it in supervised fashion. The main issue is how to avoid explosion of parameters in f.

4.3. One-shot learning objective

Human is able to recognize a new object by seeing them just once. Contemporary deep learning systems are far from it. They need hundreds or even thousands of instances to make use out of it. We propose a new formulation of neural network training that focus on training toward the one-shot learning objective.

We achieve it by training neural network to predict parameters of a neural network. More formally, let $\phi(x, W_{in}, \theta) = (y, W_y)$ be a neural network parametrized by θ. Weights $W_{in} \in \mathbb{R}^{f \times l}$ are a top layer weights, where f is a top layer feature size, and l is a number of labels. $W_y \in \mathbb{R}^f$ is a single weight vector used to predict class y.

For a given new class x', y', we compute $, W_{y'} = \phi(x', None, \theta)$ to produce weights $W_{y'}$. We use this weights to classify other objects belonging to the class y'.

This work is under progress. It is supervised by my collaborators Elias Bingham, and Brenden Lake.

4.4. Meta-optimizer

I would like to build a meta-optimizer that would be used to train neural network. Such optimizer would consume gradients of a neural network, and would decide on the next update step. Optimizer itself could be parameterized with a neural network. Proper weights of meta-optimizer should be able to simulate any first order, gradient-based, learning algorithm like SGD, momentum, LBFGs etc. This implies that meta-optimizer should subsume all first order, gradient-based optimization techniques. Trained meta-optimizer could update the network in a much more clever way, and a single sample could provide enough knowledge. This work is under progress. It is supervised by my collaborator Rahul Krishnan, and Anna Choromanska.

5. Discussion

Tackling aforementioned problems would take us much closer to the real intelligent systems, and defines for me the three main pillars of artificial intelligence. However, there are many other problems, which would need to be solved and integrate to achieve fully intelligent system, e.g. navigation, learning by imitation, cooperation, and many oth-

ers. However, I think that, all that skills can be integrated by means of external interface, and don't have to be modeled in any special way. For instance, navigation skill could emerge as an use two interfaces (1) GPS location interface, and (2) an external memory.

6. Disclaimer

This is my personal opinion, and it shouldn't be judged in a scientific way.

I strongly believe that creation of artificial intelligence is potentially dangerous. However, I think, that more dangerous is avoiding to create it. We exhaust resources of our planet in rapid fashion, and lack of resources leads to wars. The only way to prevent it is to have abundance of resources. Artificial intelligence could provide abundance of all resources.

References

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 6645–6649. IEEE, 2013.

Graves, Alex, Wayne, Greg, and Danihelka, Ivo. Neural turing machines, 2014.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Mikolov, Tomáš. *Statistical language models based on neural networks*. PhD thesis, Ph. D. thesis, Brno University of Technology, 2012.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

Weston, Jason, Chopra, Sumit, and Bordes, Antoine. Memory networks, 2014.

Zaremba, Wojciech and Sutskever, Ilya. Learning to execute, 2014.

Zaremba, Wojciech, Kurach, Karol, and Fergus, Rob. Learning to discover efficient mathematical identities. *arXiv preprint arXiv:1406.1584*, 2014.