Master thesis project

Probabilistic Approach to Adaptive Computational Time in Neural Networks

May 2017

Abstract

This master thesis project intends to develop a model that learns a task dependent optimization algorithm and approximates a posterior halting distribution over time steps by means of optimizing a variational lower bound on the log-likelihood. We hope to show that our ACT optimizer acts more *greedy* than the baseline LSTM optimizer from [1] and that the posterior halting distribution can be exploited to dynamically determine the optimization step when computation should stop.

Summary of the project

The underlying idea of the master thesis project briefly described here, originates in the recent work of Alex Graves [3]. The study outlines an approach of *Adaptive Computational Time* (ACT) applied to Recurrent Neural Networks (RNNs) that learn how many computational steps to take between receiving an input and emitting an output. Although it takes additional computational effort during training and inference it is clearly beneficiary to endow a machine learning model with the ability to be parsimonious in its use of computation, ideally limiting itself to the minimum number of steps necessary to solve an optimization problem. This view is also motivated by the assumption that for the most general case in which the model tries to minimize some objective function f, we suppose the evaluation of f for some input x and the computation of the gradients of f w.r.t. the model parameters θ to be expensive operations.

The work of [3] is important because it makes a significant contribution towards gradient-based approaches for learning the number of computational steps in a neural computation graph. The RNN learns a so called *halting distribution* by augmenting the network output with sigmoidal halting units. In addition the objective function is extended with a so called *ponder cost* which acts as a (time) penalty. The stopping policy of the model is based on the cumulative probability of the halting distribution. Although there is some controversy whether the ponder cost is truly differentiable the experimental results especially on the four supervised, synthetic tasks reveal that the approach works.

Another lead of this project is the recent surge of interest in using neural networks to learn optimization procedures, applying a range of innovative meta-learning techniques ([4], [1], [2]). The research conducted in the master thesis project is in particular based on the work of [1]. The approach taken in this study focuses on learning how to utilize gradient observations over time, of the functions to be optimized (e.g. a loss function of an image classification network), in order to achieve fast learning of the underlying model. The authors show that such a model outperforms (w.r.t. convergence speed and final minimum) hand-designed optimizers (e.g. ADAM, RMSprop and SGD) on tasks like simple quadratic function optimization and training a neural network for image classification.

This project combines the ideas of [1] and [3] and has the following objectives: (1) develop a loss function that encourages the optimizer to be more *greedy* than the baseline LSTM optimizer of [1] and therefore takes larger optimization steps towards the optimum; (2) design a training procedure that achieves comparable results as the baseline model using less computational effort[1]; (3) endow the model with the ability to decide dynamically when to stop optimization.

Our work takes the LSTM optimizer from [1] as a baseline model and develops an extension (denoted ACT optimizer) which enables the model to approximate a discrete posterior distribution over time steps t that specifies the probability that computation stops at step t.

In order to train the ACT model we optimize a variational *lower bound* on the log-likelihood

[1] measure needs to be defined.

$$\log p(\mathbf{x}) \geq \mathcal{L}(\phi, \theta, \mathbf{x}) = \sum_{t=0}^{\infty} q_\phi(t|\mathbf{x}) \Big(\log p_\theta(\mathbf{x}|t) - \log \frac{q_\phi(t|\mathbf{x})}{p(t)} \Big) \ , \tag{1}$$

where $\mathbf{x}$ denotes an observed, continuous random variable (e.g. the target values of a regression function), t is a discrete latent random variable that represents the time steps and θ respectively ϕ are the parameters of the model and the RNN inference network. The project drew inspiration from the work of [5] in which the authors develop an algorithm to translate the problem of solving a Markov Decision Problem (MDP) into a problem of likelihood maximization. From their work we extracted the idea of decomposing an infinite-horizon MDP problem into a mixture of finite-time MDPs which is a useful concept in optimization procedures where the number of necessary optimization steps (i.e. horizon T) is not known in advance.

More specific we decompose $q(t|\mathbf{x})$ into

$$q(t|\mathbf{x}) = \sum_{T=0}^{\infty} q(t|\mathbf{x}, T) p(T) \ , \tag{2}$$

and substitute 2 into 1 which results after some algebraic operations in the following formalization of the lower bound

$$\log p(\mathbf{x}) \geq \sum_{T=0}^{\infty} p(T) \sum_{t=0}^{T} q_\phi(t|\mathbf{x}, T) \Big(\log p_\theta(\mathbf{x}|t, T) - \log \frac{q_\phi(t|\mathbf{x}, T)}{p(t|T)} \Big) \ . \tag{3}$$

A more detailed outline of the pursued approach can be found in section 4 of the following document.

The performance of the LSTM and ACT optimizers will be compared by means of the following experiments: (1) optimization of simple regression functions; (2) optimization of a simple neural network trained on MNIST dataset and (3) optimization of a deeper neural network trained on CIFAR-10.

Planning

Please note that *iterative loops* will be necessary for most of the activities pointed out below.

1. February was used for literature study;
2. March was used to start a proof of concept in which both models were evaluated on 2d quadratic function optimization;
3. April, evaluate proof of concept, write introduction, related work & approach section for final paper;
4. May, run experiments for generalized linear regression functions, record results, prepare MNST experiment;
5. June, 01–06–2017 to 18–06–2017 vacation, run MNIST experiments, record results;
6. July, 19–07–2017 to 23–07–2017 vacation, prepare CIFAR-10 experiments, write on final paper;
7. August, run CIFAR-10 experiments, record results, write on final paper;
8. September, write on final paper, wrap up last results, may be re-do short experiment;
9. October, intend to graduate.

References

[1] Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. Learning to learn by gradient descent by gradient descent. *arXiv:1606.04474 [cs]* (June 2016). arXiv: 1606.04474.

[2] Chen, Y., Hoffman, M. W., Colmenarejo, S. G., Denil, M., Lillicrap, T. P., and de Freitas, N. Learning to Learn for Global Optimization of Black Box Functions. *arXiv:1611.03824 [cs, stat]* (Nov. 2016). arXiv: 1611.03824.

[3] Graves, A. Adaptive Computation Time for Recurrent Neural Networks. *arXiv:1603.08983 [cs]* (Mar. 2016). arXiv: 1603.08983.

[4] Li, K., and Malik, J. Learning to Optimize. *arXiv:1606.01885 [cs, math, stat]* (June 2016). arXiv: 1606.01885.

[5] Toussaint, M., and Storkey, A. Probabilistic Inference for Solving Discrete and Continuous State Markov Decision Processes. In *Proceedings of the 23rd International Conference on Machine Learning* (New York, NY, USA, 2006), ICML '06, ACM, pp. 945–952.