USING DOWNHILL SIMPLEX METHOD FOR OPTIMIZING MACHINE LEARNING TRAINING RUNNING TIME

Lev Faivishevsky
Advanced Analytics
Intel
lev.faivishevsky@intel.com

Amitai Armon
Advanced Analytics
Intel
amitai.armon@intel.com

Abstract

Many modern machine learning algorithms rely on a set of configuration parameters, such that a proper setting of their values influences the accuracy and the running time of the algorithm. We propose an automated approach based on the Downhill Simplex optimization method for calculating the optimal parameter set in terms of training time. We demonstrated 5X-30X speedups for training the text analytics algorithm word2vec, just through using better configuration parameters.

1 Introduction

This work addresses the problem of computational performance optimization of machine learning algorithms for running fast during the train time. Each machine learning algorithm includes a set of configuration parameters (or hyperparameters), such that a proper setting of their values influences the accuracy and the running time of the algorithm. From the mathematical perspective, the current methods for selecting optimal parameters for machine learning algorithms include Sequential Model-Based Global Optimization (SMBO), Tree-structured Parzen Estimator Approach (TPE), Random Search for Hyper-Parameter Optimization in DBN and others, see the review in [4]. These approaches are designed to optimize the accuracy of a machine learning algorithm, but none of them consider its running time.

2 Speeding up training time

The proposed method solves the problem of setting the algorithm's parameters for the fastest running time while maintaining its accuracy, through performing multivariate optimization. The parameters are considered to be the variables, and the target function to be minimized is the running time of the algorithm required to achieve a fixed value of accuracy. The goal is minimizing the target. To the best of our knowledge, this constrained formulation of hyperparameter optimization was not considered before.

The field of mathematical optimization contains a number of techniques for minimizing a function of several variables. Most of the methods adopt the gradient based iterative approach, in which an initial set of parameters is updated in each step with better values. These values are calculated based on the gradient of the target function. The gradient itself has to be evaluated analytically or by means of numerical approximations of partial derivatives of the target function [8].

In the considered field of algorithm parameter optimization, the gradient of the target function (which is the running time of the algorithm) cannot be evaluated analytically. Therefore, in order to apply gradient based optimization techniques the gradient should be estimated by means of numerical approximations to partial derivatives. This numerical procedure is inferior for the problem of running time minimization, because in this special case the running time itself is the target function, and if we perturb a value of a parameter in both directions, we have to compute the target function for the worse direction of value parameter significant amount of times. This negatively affects the optimization running time. Therefore, the gradient based methods do not lend themselves well to this problem.

3 Adaptation of Downhill Simplex method

In order to construct an efficient optimization scheme we employ gradient-free method of numerical optimizations of the target function. In particular, we use the Downhill Simplex optimization method [7] for the multivariate optimization. The Downhill Simplex optimization method is an iterative approach, which keeps track of $n + 1$ points in n dimensions, where n is the number of parameters to be set for the machine learning algorithm. These points are considered as vertices of a simplex (e. g. a triangle in 2D, a tetrahedron in 3D).

Each iteration updates a vertex with the worst value of the target function. The simplex can then move into a new position in different ways according to the new computed value of the target function. We propose a new specific tuning of the Downhill Simplex method to ensure that the simplex method will run sufficient time to allow the algorithm to attain target accuracy without wasting additional running time after the accuracy is achieved. The Downhill Simplex method is configured to make a fixed amount of iterations that limits its running time. In the current case we may also force the stop of the evaluation of the target function if its running time exceeded the current worst time achieved far. The Downhill Simplex methods terminates when the last iteration finishes. The final values of variables are used as parameter values for the Machine Learning algorithm.

4 Application to speeding up NLP analytics for Intel Architecture

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing where words or phrases from the vocabulary are mapped to vectors of real numbers in a low-dimensional space relative to the vocabulary size ("continuous space"). Such word and phrase embeddings, when used as the underlying input representation, have been shown to boost the performance in NLP tasks such as syntactic parsing and sentiment analysis [9].

Word2vec models [6] are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words: the network is shown a word, and must guess which words occurred in adjacent positions in an input text. After training, word2vec models can be used to map each word to a vector of typically several hundred elements, which represent that word's relation to other words. This vector is the neural network's hidden layer.

In the explored case the goal was to accelerate the running time of word2vec training. We considered 8 main parameters of the word2vec algorithm. We studied the TensorFlow word2vec implementation [1], which has default values of these parameters optimized for best accuracy. We applied our method for the running time minimization for achieving the same accuracy. Each iteration of Downhill Simplex method ran the word2vec training as much time as required to achieve needed testing accuracy. We tested the proposed approach on two sets of initial parameters values, corresponding to different possible scenarios of the word2vec embeddings usage: embedding sizes of 200 and 300, see [2] for examples of embedding sizes used in different domain applications. Training was performed on the popular text8 data set [3], that consists of 17 million words.

In these use cases the running time of the algorithms was accelerated 30X times from 276.9 minutes to 8.5 minutes only for embedding size 200, and 5.5X times from 80.2 minutes to 14.3 minutes, see Table 1. Our use case was performed on Xeon E5 4699 V3 dual socket 18 cores processor.

Finally, the optimal parameters found by our procedure were applied to training the word2vec algorithm on the much larger one billion word data set [5]. Previous default parameters made the word2vec training on the data set computationally difficult on a single node machine, because it would take about a week to achieve an optimal accuracy. With the optimal set of parameters, found on the 60 times smaller text8 dataset, the word2vec training took only 4 hours to achieve the same accuracy on the Intel Xeon E5 4699 V3 machine.

Table 1: Results for hyperparameter optimization of Word2vec on Intel Architecture

Parameter	Default	Optimal	Default	Optimal
Embedding size	200	200	300	300
Initial learning rate	0.02	0.0268	0.025	0.0269
Negative samples per training example	100	24	25	24
Numbers of training examples each step processes	16	520	500	496
The number of concurrent training steps	12	36	12	36
The number of words to predict to the left and right	5	5	15	5
The minimum number of word occurrences	5	7	5	7
Target				
Running time to accuracy, minutes	276.9	**8.5**	80.2	**14.3**

References

[1] `https://www.tensorflow.org/versions/r0.11/tutorials/word2vec/index.html`.

[2] `https://github.com/3Top/word2vec-api`.

[3] text 8 dataset. http://mattmahoney.net/dc/textdata.

[4] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl. Algorithms for hyper parameter optimization. In *Advances in Neural Information Processing Systems 23*. 2011.

[5] C. C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. One billion word benchmark for measuring progress in statistical language modeling. *INTERSPEECH*, page 2635–2639, 2014.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 25*. 2013.

[7] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[8] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[9] R. Sochera, A. Perelygin, J. J. Wu, C. Jason, C. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. *EMNLP*, 2013.