Extrapolating Learning Curves of Deep Neural Networks

Tobias Domhan DOMHANT@CS.UNI-FREIBURG.DE
Jost Tobias Springenberg SPRINGJ@CS.UNI-FREIBURG.DE
Frank Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg, Germany

1. Introduction

Experienced human experts in deep learning commonly rely on a large "bag of tricks" to determine model hyperparameters (Bengio, 2012), as well as learning rates for stochastic gradient descent (SGD) (LeCun et al., 1998; Bottou, 2012). Using this acquired knowledge they can often tell after a few SGD iterations whether the training procedure will converge to a model with competitive performance or not. To save time, they then prematurely terminate runs expected to perform poorly. Automating this manual trick would be valuable for speeding up both structure search (Bergstra et al., 2011, 2013; Swersky et al., 2013) and hyperparameter optimization (Snoek et al., 2012; Eggensperger et al., 2013), which are currently often prohibitively expensive (Krizhevsky et al., 2012).

We take a first step towards this goal by studying methods for extrapolating from the first part of a learning curve to its remainder. Preliminary results indicate that such predictions can be quite accurate and enable the early termination of poor runs.

2. Learning Curves of Deep Neural Networks

To create learning curves for a broad range of network structures and hyperparameters, we heavily parameterized the Caffe deep neural network software (Jia, 2013); we considered a total of 81 hyperparameters: 9 network parameters (e.g. learning rate and learning rate schedule, momentum and the number of fully connected layers) and 12 parameters for each of up to 6 layers (e.g. width, weight initialization, and dropout rate). We performed 5 runs of each of the structure search & hyperparameter optimization methods SMAC (Hutter et al., 2011) and TPE (Bergstra et al., 2011) to optimize performance on the CIFAR10 image classification dataset (Krizhevsky and Hinton, 2009).[1] Figure 1 (left) shows SMAC's and TPE's performance over time. Combined, they evaluated 800 instantiations of Caffe, and we recorded the learning curve of each of them. Caffe terminated learning curves once they did not make progress for 25 epochs or exceeded 285 epochs. Figure 1 (middle) shows a random sample of 100 curves.

3. Extrapolation of Learning Curves

Let $y_{1:n}$ denote the observed part of the learning curve for the first n steps. Our basic approach is to fit parametric models M to $y_{1:n}$ and use them to infer y_m, with $m > n$.

From a broad range of parametric models we selected ten that match the shape of learning curves. These are typically increasing, saturating functions, for example functions from the power law or the sigmoidal family.

We first considered averaging predictions of these models via the Bayesian model averaging (BMA) framework (Hoeting et al., 1999). However, in practice BMA often overconfidently selected single imperfect models, suggesting that the true learning curves are not perfectly captured by any single one of these models. To increase representative

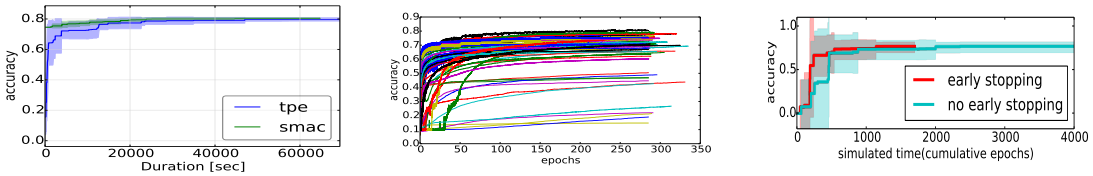

Figure 1: Left: best performance found by hyperparameter optimization over time. Middle: representative sample of learning curves encountered by optimizers. Right: simulation of hyperparameter optimization with early stopping based on extrapolation.

power, we instead linearly combined the $k = 10$ selected models into a single model: $f_{comb}(x) = \sum_{i=1}^{k} w_i f_i(x|\boldsymbol{\theta}_i) + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma^2)$, $\sum_{i=1}^{k} w_i = 1$ and $\forall w_i, w_i \geq 0$.

Let ξ denote our model's parameters $(w_1, \ldots, w_k, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k, \sigma^2)$. To capture model uncertainty, we used MCMC[2] to draw samples $\boldsymbol{\xi}^{(1)}, \ldots, \boldsymbol{\xi}^{(S)}$ from $P(\boldsymbol{\xi} \mid y_{1:n}) = P(y_{1:n} \mid \boldsymbol{\xi})P(\boldsymbol{\xi})/P(y_{1:n})$, with a uniform weight prior and priors capturing monotonically increasing curves for each model. We then predicted y_m as: $P(y_m \mid y_{1:n}) = \frac{1}{S}\sum_{s=1}^{S} P(y_m \mid \boldsymbol{\xi}^{(s)})$.

4. Experiments

Figure 2 (top) shows a typical learning curve and our extrapolations based on observing 10 and 40 epochs. Figure 2 (bottom) shows the same for a somewhat unusual learning curve. Overall, predictions became more certain and more accurate when based on more epochs. On average, the RMSEs for predictions of final performance based on 10, 40, and 60 observed epochs were 0.25, 0.19, and 0.11 respectively.

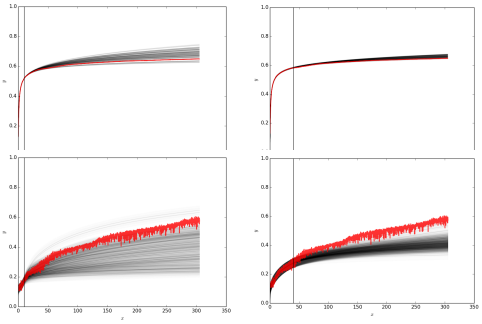

Figure 2: Extrapolations of two learning curves, based on observing 10 (left) and 40 (right) epochs. We plot 100 randomly selected posterior samples semi-transparently.

However, overall, since our prior of monotonically increasing performance only imperfectly captured the flattening towards the end of many learning curves, our models tended to overestimate final performance: 0.11%, 0.30% and 0.59% of the true final performances laid above, within, and below our model's 90% predictive confidence interval, respectively. We accepted this since overpredictions are much less harmful for early stopping than underpredictions.

Figure 1 (right) shows a preliminary experiment to simulate the use of extrapolation-based early stopping in hyperparameter optimization. For this, we simulated optimization trajectories that visit the 800 learning curves from above, terminating a learning experiment once its extrapolated performance at the cutoff point is predicted to be worse than the best seen so far with 99% probability. This simulation resulted in a 2.7-fold reduction of overall runtime. Note that in actual hyperparameter optimization runs, we would also have to predict the return value for stopped runs and poor predictions could mislead the optimizer; we plan to study this in future work.

1. We preprocessed images by extracting k-means features extracted as described by Coates et al. (2011) and also done by Gens and Domingos (2012) and Swersky et al. (2013)).
2. Specifically, we sampled 100 000 parameter instantiations for each model, using the sampler emcee (Foreman-Mackey et al., 2013) with a burn-in of 500, a chain length of 1500 and 100 parallel chains.

References

Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In G. Montavon, G. B. Orr, and K. Müller, editors, *Neural Networks: Tricks of the Trade (2nd ed.)*, pages 437–478. Springer, 2012.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems 24*, 2011.

J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*, pages 115–123, 2013.

L. Bottou. Stochastic gradient tricks. In G. Montavon, G. B. Orr, and K. Müller, editors, *Neural Networks, Tricks of the Trade, Reloaded*, pages 430–445. Springer, 2012.

A. Coates, A. Y Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS Workshop on Bayesian Optimization in Theory and Practice (BayesOpt'13)*, 2013.

D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman. emcee: The MCMC Hammer. *Publications of the Astronomical Society*, 125:306–312, 2013.

R. Gens and P. Domingos. Discriminative learning of sum-product networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 3239–3247. 2012.

J.A. Hoeting, D. Madigan, A.E. Raftery, and C.T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417, 1999.

F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization 2011*, pages 507–523, 2011.

Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012.

Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, chapter 2, page 546. 1998.

J. Snoek, H. Larochelle, and R.P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2960–2968, 2012.

K. Swersky, D. Duvenaud, J. Snoek, F. Hutter, and M.A. Osborne. Raiders of the lost architecture: Kernels for Bayesian optimization in conditional parameter spaces. In *NIPS workshop on Bayesian Optimization in theory and practice (BayesOpt'13)*, 2013.