LEARNING RATE OPTIMIZATION THROUGH STEP SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern machine learning models require selecting hyper-parameters prior to training; important variables that define the way in which the model can learn, but which cannot be learned by the model itself and instead need to be assigned in advance. Of the hyper-parameters that must be selected when configuring a model, arguably the most important is the "learning rate" of the model, the step size that the model uses when learning its parameters with gradient descent. Here we propose a method to deliberately select a learning rate by training a model for a small number of steps over a variety of learning rates and resetting both the model parameters and dataset between each trial. A curve of the log of those rates vs the losses achieved for each is used to select a viable range for an optimal learning rate, and we compare several methods of selecting an optimal point within that range. The performance of the selections from these methods are then evaluated using a full grid search, and in our experiments, they reliably select learning rates that achieve a good accuracy for any given model.

1 INTRODUCTION AND BACKGROUND

The performance of a deep learning model can vary greatly depending on what learning rate is used for training. If a selected learning rate is too low, the model may be unable to learn quickly enough to reach convergence in a reasonable timeframe or may become stuck in a local minimum or saddle point instead of continuing to a global minimum.

High learning rates are also undesirable, as they have a risk of overshooting an ideal model configuration and can potentially even cause the model's loss to explode to infinity. The higher a selected learning rate, the more inconsistent the results of a training session for a model become as well, since the step size is not fine enough to reach any minimum configuration. The ideal learning rate will balance these two risks and should be high enough to allow the model to learn robustly, but low enough that the risk of over-shooting or becoming unstable is reduced.

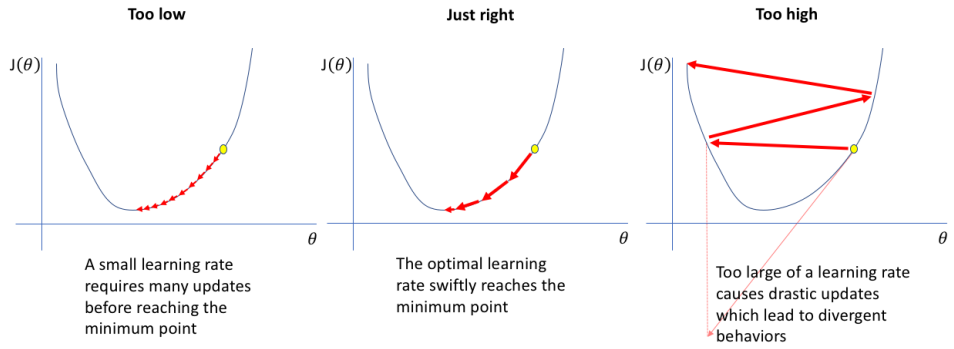


Figure 1: Example of the effects of different learning rates for a simple model. Image Source: (Jordan, 2018)

The selection of an appropriate learning rate is therefore a very important factor in the final performance of a model. Despite this, the traditional method for learning rate selection is either through trial and error based on prior knowledge, or through a more thorough "grid search", whereby a range of learning rates are systematically tested. This methodology is inherently slow and imprecise, and can occupy a significant amount of development time.

2 Prior Work and Our Approach

There have been some suggestions for better ways to approach this problem, and in recent years a few methods have been proposed that work off an idea first presented in Smith (2015). In that paper, learning rates are evaluated by first training a model for several epochs, incrementing the learning rate slightly at each step, until the loss of the model no longer improves. A learning rate window is then selected from where the accuracy of the model first begins to improve, until the point where stops improving. Figure 2 shows the outcome of that learning rate analysis, both raw and smoothed with a moving average.

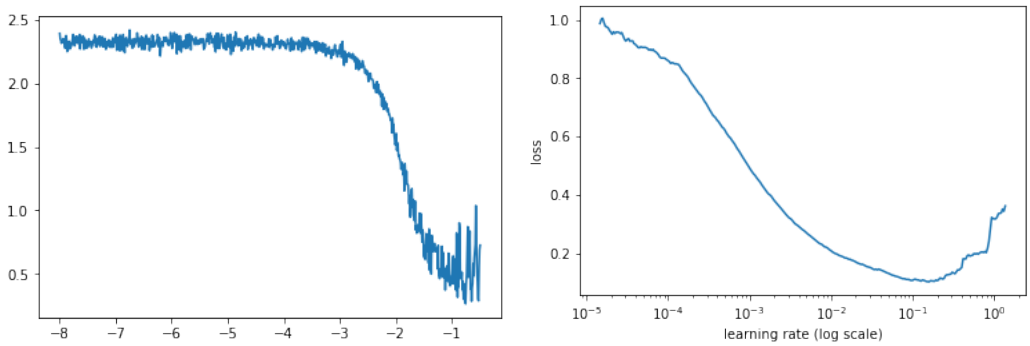


Figure 2: The loss achieved by the method of slowly increasing learning rates over a single training cycle of a model. The figure on the left shows the actual loss achieved by each batch, while the figure on the right shows a weighted average of those weights. Image Source: (Gugger, 2018)

In Smith (2015) this window was used to establish a cyclic learning rate scheduler, but it was later used in Gugger (2018) to select an optimal learning rate. In that approach, they choose the point in that window where the loss stops improving, then select a learning rate an order of magnitude below that point. This is the approach implemented in the popular machine learning library fast-ai (Howard et al.).

In our view, this approach has a few shortcomings. First and most significantly, since learning rates are evaluated over a full training cycle of a model without resetting anything, the data and model configuration being used to test each learning rate are different. This means that a) the results for each learning rate are not directly comparable, and b) the effects of the changing learning rate are difficult to disentangle from the effects of the model training. Additionally, using the minimum loss as a point of reference is problematic since that is the point where the model is no longer learning anything. And since each learning rate is evaluated for a single step, the results are very noisy and must be smoothed, obscuring the precise location of critical events. The creators of this method recognized this, and do subtract an order of magnitude from the point where the loss reaches a minimum to address these issues, but we are interested in a more deliberate choice of learning rate.

Some improvements have been suggested to this, notably Surmenok (2017), where the point of steepest loss decline is selected rather than an order of magnitude below the point of minimum loss, and Wittmann, where learning rates are tested for a number of steps and the dataset and model are reset between each testing cycle, and the optimal learning rate is then chosen as the one that reaches the lowest loss. The first approach retains the comparability issues because of the single training cycle however, and the second approach selects a point right before the loss begins to get worse again, giving it the risk of instability. These unstable results have shown up in our tests.

However, the evaluation procedure from Wittmann where the model and dataset are reset between each trial does get around some of the significant problems with prior approaches, so we choose to retain that in our methods. We therefore start by testing the viability of a range of learning rates by training a small number of steps (we used 20 steps in our tests) for each learning rate, testing the resulting model with a validation set, then resetting the model and dataset after each trial. This ensures an equal comparison between learning rates, since the same initial model parameters and data batches are considered.

From there, our contributions with this paper are as follows:

1. Using a log-scaled graph of those trial outputs, we propose a method for selecting a viable learning rate window, which should contain all learning rates that are worth consideration. The upper limit of this window will be the point at which the learning-rate/loss curve reaches its minimum point, and the lower limit of this window will be the first minimum of the curvature of the learning-rate/loss graph (i.e. the "knee" in the curve).
2. Within that window, we suggest 3 methods for choosing an optimal learning rate: the log-midpoint of that window, the point at which half of the possible loss reduction is achieved, and the point at which the most negative derivative of the learning-rate/loss curve is reached.
3. We also suggest a method of more efficiently selecting which learning rates to evaluate when building the learning rate/loss curve, instead of doing a brute force grid search.

3 METHOD

3.1 VIABLE WINDOW SELECTION

Using the method described in the previous section, whereby learning rates are evaluated by training for n steps, testing the resulting model with an evaluation set, and recording the resulting loss, we can generate a graph such as the one shown in Figure 3. The pattern visible there is common, with lower learning rates learning very little (a "plateau" in the graph), and higher learning rates getting higher losses again as they begin to quickly overshoot ideal configurations. Note that when looking at a plot of learning rates vs loss, it is important to use a log scale for the learning rate, since it's proportional changes in learning rate that matter as opposed to absolute changes.

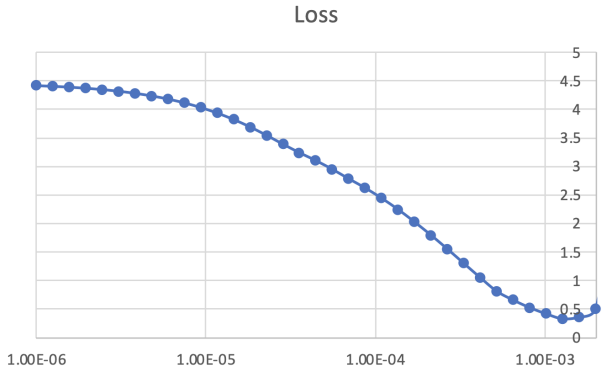


Figure 3: An example result of learning rates vs the loss achieved on a validation set. Learning rate is shown on the x axis, and loss is shown on the y axis.

The upper bound of our viable learning rate window should be the minimum point of the graph in Figure 3. At that point our capability of learning from a freshly initialized model is at a maximum, and if we increase the learning rate past that point we risk instability, without any improvement in the model's ability to learn. We will call that point *max_lr*.

For a lower bound of the learning rate, we want to select a point where loss is just past its plateau and begins to significantly improve for the first time. This can be thought of as the knee in the curve,

which is described mathematically as minimum *curvature* (Kline, 1998). The standard formula for the curvature of a line y is given in equation 1 below.

$$k = \frac{y''}{(1 + (y')^2)^{\frac{3}{2}}} \tag{1}$$

If we plot the curvature of the graph in Figure 3, we get the curve shown in Figure 4. The lower bound of the learning rate will be the first minimum[1] visible on that graph, which we will call *min_lr*. The viable learning rate window is then the range between *min_lr* and *max_lr*.

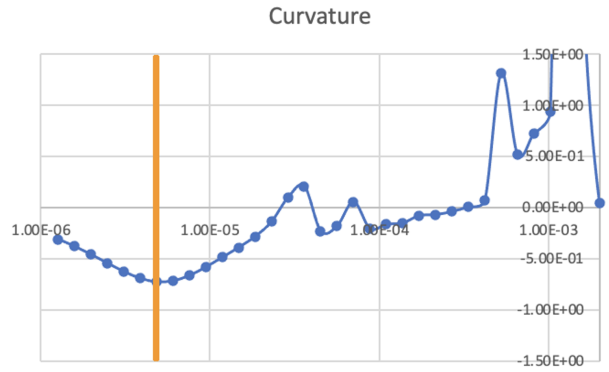


Figure 4: The curvature for the graph shown in Figure 3, with learning rates shown on the x axis, and curvature shown on the y axis. The vertical line is at *min_lr*, the learning rate where first minimum of the curvature occurs.

3.2 Optimal Learning Rate Selection

Within the viable window, there are 3 approaches we propose to select an optimal learning rate, which all work off the intuition that an ideal learning rate is a tradeoff between a range that is too low and risks not learning enough, and a range that is too high and risks overshooting and instability.

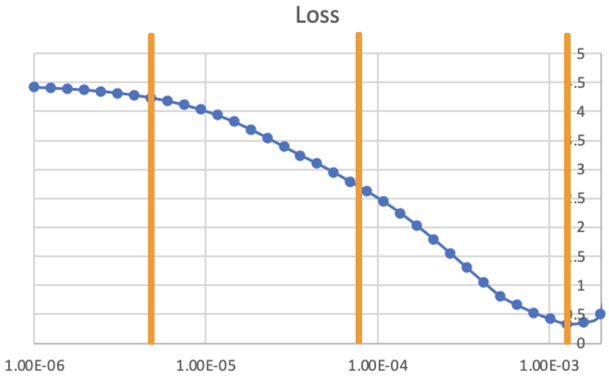


Figure 5: Method 1 for selecting an optimal learning rate, at the log-midpoint of the viable learning rate window.

[1]Note that in practice we cannot simply select the first minimum of the curvature graph as *min_lr* because of small fluctuations in the plateau phase. Instead, it's better to choose the minimum curvature point to the left of the learning rate where half the maximum loss reduction is achieved.

3.2.1 SELECTION METHOD 1

With that in mind, our first method is to take the log-midpoint on the learning rate axis between *min_lr* and *max_lr*. This represents a tradeoff between those 2 extremes and posits that the ideal range is the point furthest from both. This method is shown in Figure 5.

3.2.2 SELECTION METHOD 2

Our second method follows similar logic but considering the loss axis. The maximum loss we will get is with a learning rate of 0, meaning nothing will be learned, and the minimum loss we will get is again at *max_lr*. This method therefore selects a learning rate that achieves a loss halfway between those boundaries. This approach is shown in Figure 6.

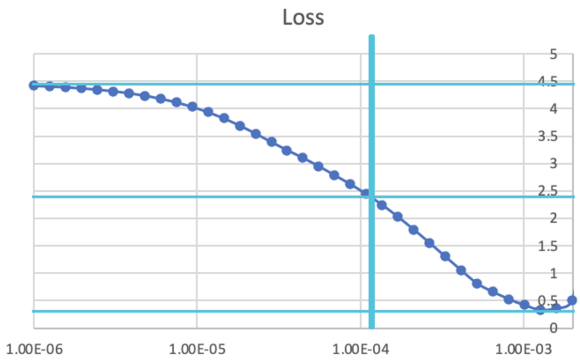


Figure 6: Method 2 for selecting an optimal learning rate, at the point where half of the possible loss reduction is achieved.

3.2.3 SELECTION METHOD 3

Finally, our third method combines several of the previous approaches described in section 1 and selects the point where the log-derivative of the learning rate/loss curve is at a minimum. This is shown in Figure 7.

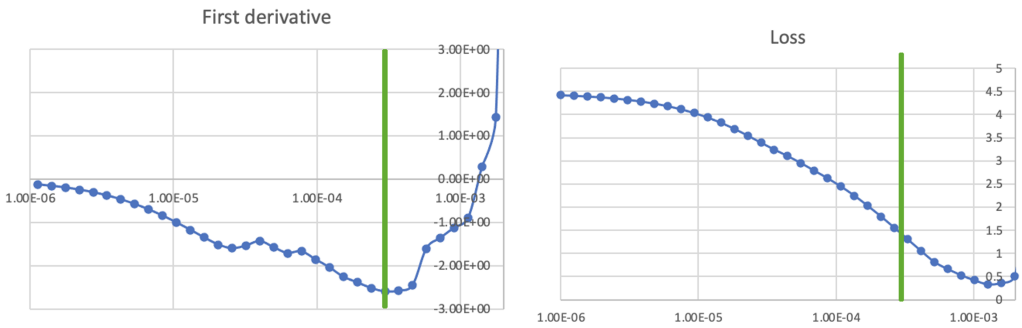


Figure 7: Selection process for method 3, choosing the point where the derivative of the loss is at a minimum. The graph on the left shows the first derivative of the loss curve, and the graph on the right shows where the point with a minimum derivative lies on the original curve.

Those three approaches are the selection methods that will be evaluated in section 4.

3.3 FULL APPROACH

The next optimization that needs to be made to this method is choosing which learning rates to evaluate when building the learning rate/loss curve. The figures that have been shown up until now are all grid search methods, but a lot of the curve does not provide meaningful information.

The approach we will take is to first do a course grid search, to generate some initial data. A step size of a half order of magnitude worked well in our tests. From there, we will systematically calculate each point of interest for each of the 3 selection methods over the known results, then select a learning rate at the log midpoint of each of those points and their neighbors for further training. Those points will be evaluated, and the process repeats until each point of interest is known within a specified threshold.

Using this method, our learning rate/loss curve comes out like the graph in Figure 8. This allows us to get more precise values for these points of interest, without wasting time evaluating learning rates that are not useful.

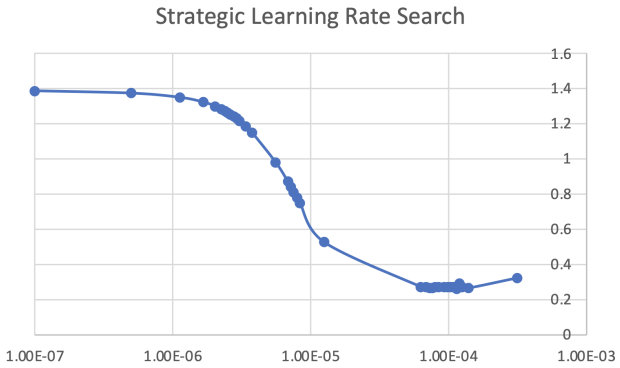


Figure 8: An example of the learning rate/loss curve, when learning rates are evaluated in a strategic way, focusing on our points of interest.

4 EXPERIMENTS

To compare the performance of each selection method, we generated 3 models to evaluate, chosen to test a variety of architectures and dataset types.

First, using an implementation from huggingface (HuggingFace), we used an encoder-only BERT transformer model (Devlin et al., 2018), training over the AG_News dataset (Zhang et al., 2015). Second, we used an encoder/decoder BART transformer model (Lewis et al., 2019), also from huggingface (HuggingFace), training on a proprietary Samsung dataset called ACC. Third, we included an implementation of ResNet18 (He et al., 2015) from torchvision (PyTorch), trained over the hymenoptera_data dataset[2].

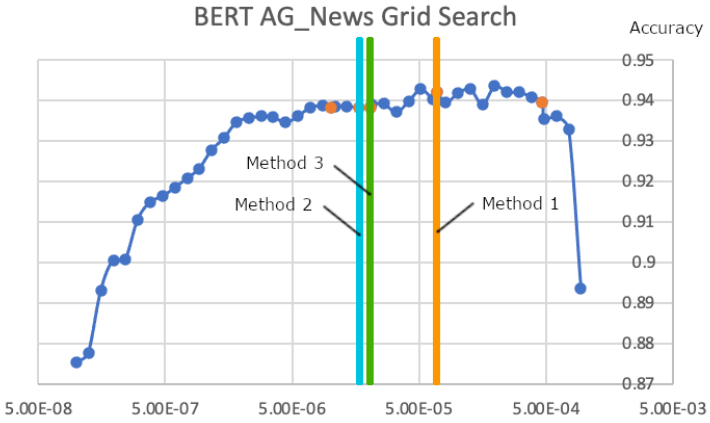
These models were trained until convergence in a full grid search, to validate whether this selection method yields good learning rates. The outcome of all points of interest were also included in these tests.

5 RESULTS

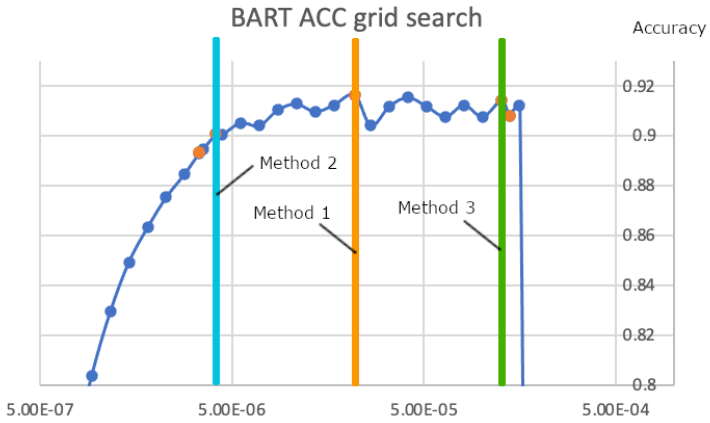
The results of our tests are displayed in Figure 9, showing a plot of final test accuracies after convergence for each model, evaluated using each learning rate in the grid search. The learning rates selected by these methods are shown with vertical bars, with method 1 in orange, method 2 in blue, and method 3 in green.

[2]The hymenoptera_data dataset can be found at `https://download.pytorch.org/tutorial/hymenoptera_data.zip`

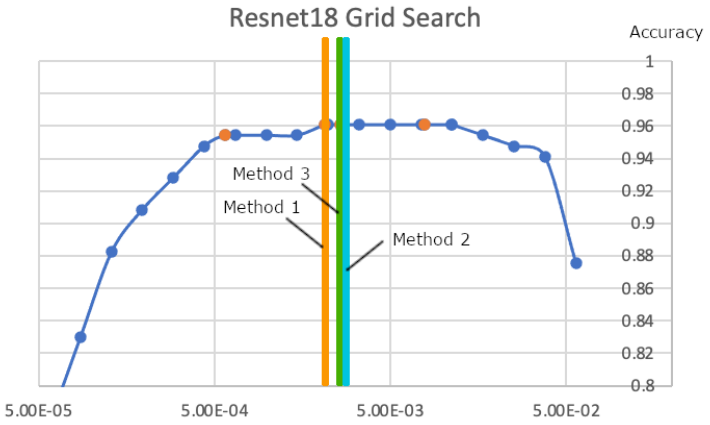
In each of these 3 tests, the learning rate selected using method 1 performed the best overall, followed by method 2 and method 3. As can be seen in Figure 9, method 1 returned learning rates for all 3 tests that are right in the middle of the range that achieves the best accuracy overall.



(a) BERT Model trained over AG News dataset.



(b) BART Model trained over Samsung's ACC dataset.



(c) ResNet18 Model trained over hymenoptera_data dataset.

Figure 9: Results of the learning rate search on 3 models, shown on a grid search graph, training each learning rate to convergence. Method 1 is shown in orange, method 2 in blue, and method 3 in green. The viable window is indicated by the orange dots.

6 Discussion and Future Work

In this overview we have compared several ways of selecting learning rates, by looking at how changes in the learning rate affects the learning capacity of different models over the first n steps in their training cycle. The results are encouraging, and further evaluation with additional models has shown similar effects, indicating that this is a reliable method of learning rate selection.

The biggest advantage of these approaches however is in the time needed for learning rate optimization. Hyper-parameter tuning is typically a major portion of a machine learning researcher's time, so a method to select the most critical of those parameters can make research a more efficient process. In our tests, running a learning rate search in this way took less than 10 minutes to complete, whereas a single full training cycle of a model could take an hour or more. When you consider that finding a learning rate through a grid search or trial and error requires numerous full training cycles, this method represents a time savings of a couple orders of magnitude.

One major limitation of this approach is that it selects for an ideal starting learning rate. This is important, but how that learning rate should be modified during training is an additional challenge. Applying similar methodological approaches to learning rate scheduling strategies is the next logical step in the exploration of efficient learning rate optimization.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Sylvain Gugger. How do you find a good learning rate, March 20 2018. URL https://sgugger.github.io/how-do-you-find-a-good-learning-rate.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Jeremy Howard et al. Fast ai, lr_finder. URL https://fastai1.fast.ai/callbacks.lr_finder.html.

HuggingFace. Transformers. URL https://huggingface.co/transformers/.

Jeremy Jordan. Setting the learning rate of your neural network, March 1 2018. URL https://www.jeremyjordan.me/nn-learning-rate/.

Morris Kline. *Calculus: An intuitive and physical approach*, pp. 457–461. Dover, 2nd edition, 1998.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019. URL http://arxiv.org/abs/1910.13461.

PyTorch. Pytorch vision. URL https://pytorch.org/vision/stable/index.html.

Leslie N. Smith. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186, 2015. URL http://arxiv.org/abs/1506.01186.

Pavel Surmenok. Estimating an optimal learning rate for a deep neural network, November 12 2017. URL https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0.

Fernando Marcos Wittmann. Lr finder. URL https://github.com/WittmannF/LRFinder.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626, 2015. URL http://arxiv.org/abs/1509.01626.