Paper: RETHINKING THE HYPERPARAMETERS FOR FINE-TUNING

Reviewer: B RAJA NARASIMHAN

Date: 19-8-20

INTRODUCTION

Hyperparameter tunings are one of the most important aspects of making your model perform better. A data scientist mentioned Hyperparameter tuning is like tuning guitar, if done well magic happens. In this paper, researchers have analyzed how we should choose the hyperparameters so we get the best results.

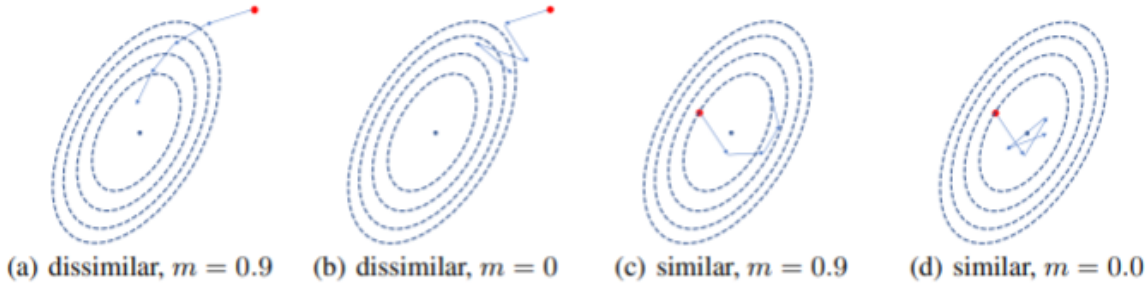
While optimizing any CNN model what most of the people who implement choose default parameters which is m=0.9, $\eta = 0.01$, and weight decay of 0.0001. Here they try various hyperparameter combinations and try to sense and a pattern here. Most of us use pre-trained models like ResNet, densenet, etc which have been trained using ImageNet dataset and they find a pattern on what combination of hyperparameters we have to use based on how similarity of the dataset which will be used by us and ImageNet dataset. They also bring up a concept of coupled learning rate.

EXPERIMENT SETTINGS

From the datasets chosen in the paper, I have tried their methods on Dogs, Caltech, Indoor, and Flowers. They have used resnet101 and Batch Size of 256, but I tried using Batch Size of 16 as that was the limit of my GPU(Google Colab), and also instead of Cal-

tech256, I used Caltech101. Images size is 224X224X3 and have used basic image augmentation.

HYPERPARAMETER BASED ON SIMILARITY OF DATASET



(a) dissimilar, $m = 0.9$ (b) dissimilar, $m = 0$ (c) similar, $m = 0.9$ (d) similar, $m = 0.0$

So I used three combinations of hyperparameter for each dataset. I either put m = 0.9 or 0, η will be 0.01 or 0.001 or 0.005 and regularization or weight decay will be either 0 or 0.0001. So I tried first using datasets that are similar to the ImageNet which were Dogs, Caltech, etc. The similarity is calculated with the help of the paper "Large Scale Fine-Grained Categorization and Domain-Specific Transfer Learning" . So we have to use very small momentum for a similar dataset. So I used Dog, MITIndoor, Caltech101. Here I used m=0 trained the dataset and compared it with m=0.9. As told by the paper I used SGD momentum and I set Nestrov=False in all cases as I couldn't use True when m=0 and I felt setting False for all situations would kind of bring uniformity to the whole process. As said by the paper the I had less validation loss with m=0. So the conclusion was small momentum works with datasets close to ImageNet. The value of similarity between Dogs, Caltech, and Indoor band ImageNet dataset was 0.862, 0.892, and 0.856. What I notice is the similarity value is less than 0.850 then we can take them as less similar datasets and use a different combination of hyperparameter. Since the flower dataset had a similarity value less than 0.850 and as said by the paper I got better results with m=0.9. According to researchers, this happens as Dogs, Caltech

and Indoor are like closer to ImageNet dataset. So huge momentum will push the model from away point of minimum and thus validation loss will be so much higher than m=0 but for datasets which are much far from ImageNet dataset huge momentum will help in better convergence. This is brilliantly illustrated through this diagram

COUPLED HYPERPARAMETERS

Here researchers mention that hyperparameters are coupled and it's not like we can select one hyperparameter, find the best one and make that constant and start changing the other hyperparameter. They talk about a notion called effective learning rate

: η ' $= \eta$ $/(1 -$ m). For particular η ' we can have a different combination of hyperparameter and they produce pretty much the same result. I tried this using the dog dataset for η '= 0.1(m is 0 or 0.9) and η ' = 0.01(m is 0 or 0.9) and the results were very same for a particular η ' and also smaller η ' is better for the similar dataset and I was able to get the same result. Same with optimal effective weight decay λ ' $= \lambda$ $/\eta$. Larger weight decay is preferred when datasets are similar.

CONCLUSION

This paper brings into account how much important hyperparameter is to a model. What I have noticed is most of us during optimization use default parameters and if the model is performing badly we change to more complicated or different models rather than changing hyperparameters. This paper beautifully teaches how we can manipulate different values and get better results and also explains the mathematical concepts well. We can also use this concept and apply it to other optimizers like Adam etc. This concept has huge scope and we can extrapolate this concept into other fields of AI.