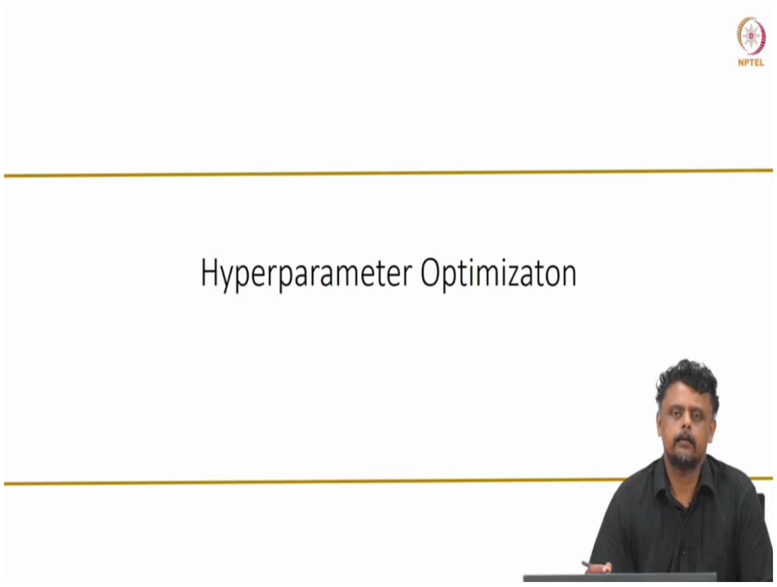
Machine Learning for Engineering and Science Applications

Doctor Ganapathy Krishnamurthy
Department of Engineering Design
Indian Institute of Technology Madras

Hyper parameter Optimization

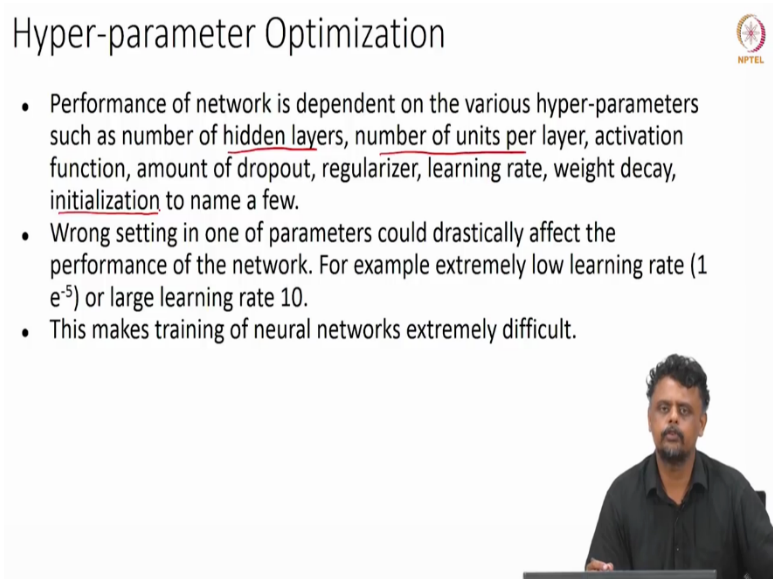
(Refer Slide Time: 0:16)



Hyperparameter Optimizaton

In this video we will look at hyper parameter optimization which is one of the important aspects of training a deep neural network.

(Refer Slide Time: 0:25)



Hyper-parameter Optimization

- Performance of network is dependent on the various hyper-parameters such as number of hidden layers, number of units per layer, activation function, amount of dropout, regularizer, learning rate, weight decay, initialization to name a few.
- Wrong setting in one of parameters could drastically affect the performance of the network. For example extremely low learning rate (1 e^{-5}) or large learning rate 10.
- This makes training of neural networks extremely difficult.

So in general the performance of the network depends on how well you train it. And one aspect of the training is to various hyper parameters involved in constructing a deep neural network, okay. So this would involve parameters like number of hidden layers, number of units per layers or number of filters per player if you can think of it that way.

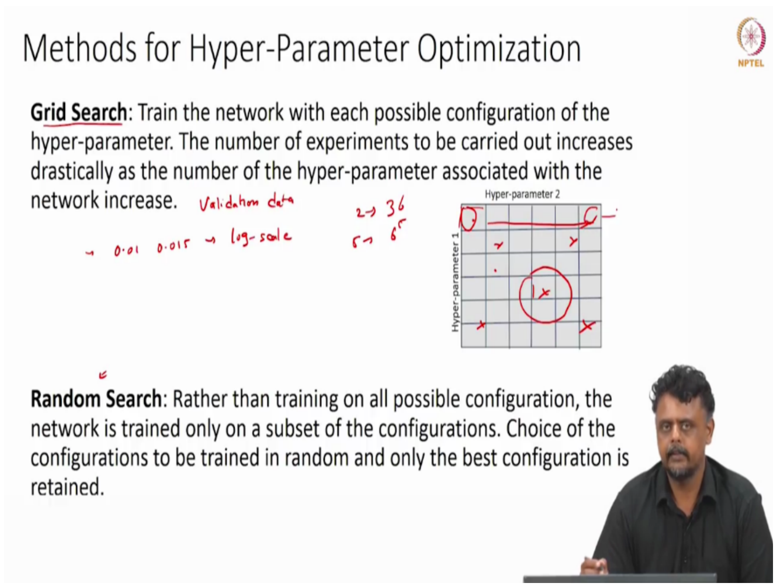
What kind of activation functions they use? If you are using some kind of dropout regularization, how much do you use that? What kind of regularizer to use L1 or L2? And what is the strength of the regularizer? What learning rates I would typically do? The weight decay? And of course initialization we will look at this as a separate topic. We will not touch this here but we will look at how to do weight initialization later on. In the later video.

Wrong setting. In this case wrong setting is, I would say it is more like not an optimal setting, it could actually affect the performance of a network okay. In many ways, in the sense, it would div, it would converge to appoint where it's not giving you a very high accuracy in terms of the classification or the last function does not decrease beyond a certain point.

Or another case is, very slow convergence if you don't have optimal values, so this is a portion, this comment is part of the network optimization is what makes the training of a network very hard, okay. So in the process if you are doing a much simpler network, let's say a couple of layers or 2 to 3 layers in general that is easier to optimize you can do that by hand per se.

But as the number of players become large and size of the dataset grows, in general it becomes very difficult to determine what exactly you have to choose as these hyper parameters.

(Refer Slide Time: 2:17)



Methods for Hyper-Parameter Optimization

Grid Search: Train the network with each possible configuration of the hyper-parameter. The number of experiments to be carried out increases drastically as the number of the hyper-parameter associated with the network increase.

Random Search: Rather than training on all possible configuration, the network is trained only on a subset of the configurations. Choice of the configurations to be trained in random and only the best configuration is retained.

So typically there are 2 ways, methods that are recommended, say it is more like heuristic than method. You look at one of them it is called Grid Search. The idea is to train the network with each possible combination of a hyper parameter, okay. And less to say as your number of parameters increases then your, the competitions and all will become much higher.

Remember that if you have a slightly deeper or larger network then you actually have to rein the network with every choice of these parameters, okay. And of course you would have a hold out dataset or called validation set data set. Validation dataset which you would use to see which parameter setting is receive the best results in terms of you know, depending on your task that the neutral network was set or accomplished.

Now this we are doing it, this hyper parameter search and we will search in a very systematic way. So in this case if you see if the columns are hyper parameter to then for a fixed value of hyper parameter one you would change hyper parameter 2 and that will be the only variable and that's one way of doing it and then you would fix that and change the hyper parameter one.

So one way is you would hold all other, it's Grid search paradigms where in you hold, you need to randomly initialize all the parameters that you are trying to optimize and just very one systematically see which gives the best validation result and you take that value of that hyper parameter, keep that fixed and then go to the next one change that and so on and so forth, okay. So this kind of grid search is possible in a grid search scenario.

However as the number of parameters increases then the number of times that you have to do is experiment goes up exponentially, very quickly. So it's very easy to see here, right? You have 2 parameters, you have 6 values for every one of them, so that's like you have to do 36 and if you have 3 or 4 or 5 and each of them has 6 combinations then you see that's 6 raise to 5, right?

So it will be for 2 parameters you get, you have 5 then it is pretty much 6 raise to 5, okay. So number of operation increases. Number of operation increases. Now another thing to look at here is the scale of the grid search. So what we meant by grid search was, so if you have certain range of your parameters, most of the parameters except for a number of layers or the number of activations per player or the number of filter kernels or continuous.

So for instance learning rates or some weight decay parameter, so that you have to discretize in a, let's a say very meaningful fashion. So for instance if you are looking at let's say learning rates just for the sake of argument then you can set the learning rate to 0.1 and then you can set it to 0.015 so on and so forth but then if you see these are of the same order of magnitude on the same scale.

So one trick is to set all of these in a log scale, okay. That's one way of splitting the grid. So this grid search method typically works very well for smaller networks where you don't have as many parameters to optimize and it's also cheaper to do these computations in a brute force manner, okay. So as your search space increases exponentially as because you have more and more number of hyper parameters to optimize. The more the better method would be to do Random search.

Here instead of training on all possible configuration, you just choose "Set Radom", so that is to the same grid, you can make the similar grid and then you would try to sample, you know different areas of the grid, okay. Just to get an idea of, what kind of results you get for each of these points that you have to. Now once amongst these 5 points, let's say that I have clicked.

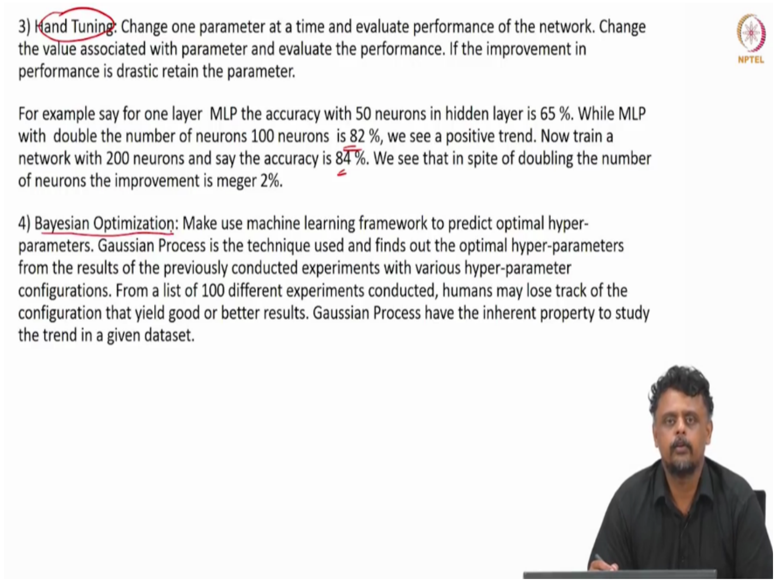
Let's say this one gives you the best result, right? What you can they do is, search much closer to this point, see for a better option that is possible. One the way this computation savings happen is that if you recall from the grid search argument, you have to hold all the parameters the same. Let's say you have 6 parameters, you keep constant 5 parameters and you have to change 6 continuously depending on a good discretization.

That means that you are repeating calculations for the same parameter value numerous times that doesn't happen in a random search. However it is important that you sample your search space intelligently, so that you are able to cover most of the space in a reasonable fashion. Other thing to look out for is that, when your best result corresponds to something at the edge.

So this is the edge of your hyper parameter grid then it is also better to actually increase the grid to the left or in that case if it is over here to the right and search a little bit more because it is possible that there are better values of a settings of the hyper parameters available beyond that value, okay. So the 2 important takeaways are that discretize your hyper parameter values on a log scale.

And you construct a grid and this grid again can be made based on what kind of values you would expect to be meaningful depending on the context of the problem or you can look at some literature survey and see what typical values are used in setup shown good performance, so the ImageNet architectures are a pretty good pointers that way and most of them do a random search in order to get to a point where they can initialize to the correct hyper parameters.

(Refer Slide Time 9:11)


3) Hand Tuning: Change one parameter at a time and evaluate performance of the network. Change the value associated with parameter and evaluate the performance. If the improvement in performance is drastic retain the parameter.

For example say for one layer MLP the accuracy with 50 neurons in hidden layer is 65 %. While MLP with double the number of neurons 100 neurons is 82 %, we see a positive trend. Now train a network with 200 neurons and say the accuracy is 84 %. We see that in spite of doubling the number of neurons the improvement is meger 2%.

4) Bayesian Optimization: Make use machine learning framework to predict optimal hyper-parameters. Gaussian Process is the technique used and finds out the optimal hyper-parameters from the results of the previously conducted experiments with various hyper-parameter configurations. From a list of 100 different experiments conducted, humans may lose track of the configuration that yield good or better results. Gaussian Process have the inherent property to study the trend in a given dataset.

So there is another way which is called as Hand Tuning which is not highly recommended which is again very similar to the grid search but you don't do it in a very systematic fashion.

We just do this to see if there is set trend, okay. So for example is given that, so if you have an MLP and if you have the neurons in all the layers and you get accuracy 65 percent and then you double the number and you get an accuracy of 82 percent that’s a positive trend.

Then maybe you can just be a little bit more adventuress and Private 200 neutrons and the accuracy of 84 percent, maybe that’s a diminishing returns there. So then maybe you see okay this is, maybe then it doesn’t make sense to go beyond let’s say h100 or 150 neurons, okay. So you can use this kind of strategy to figure out the limits of your grids, okay. See where you can start from, so that you can get a reasonable result and then maybe a grade at that time and either do a random or systematic grid search.

So there is one more advanced technique we won’t go into this in detail which makes use of a machine learning framework. So you have a learning program or learning strategy to figure out the optimal hyper parameters. Here people use by the Gaussian processes to figure out the space of hyper parameters what your validation accuracy looks like and what the accuracy on a validation data looks like.

So that’s the function you are trying to fit, since that the loss of the validation performance or the performance on the validation data as a function of the hyper parameters. So if you estimate that function using Gaussian processes that you can find out the optimal value of the hyper parameters which gives you the best possible result on the validation data.