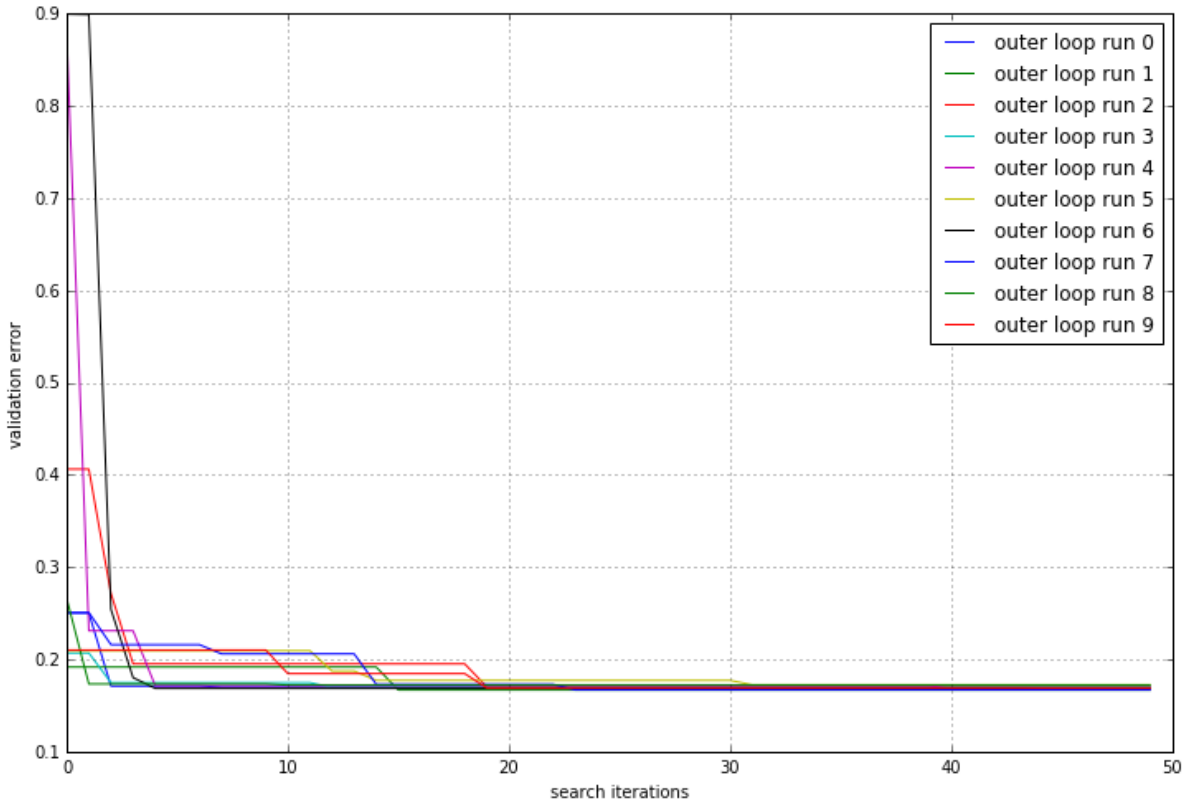
Report Exercise 5, Hyperparameter Optimisation

Task 1, random search

I ran the surrogate function 10 times (50 iterations each) with randomly sampled hyperparameters, and recorded the list of the incumbents (best validation error per run so far).

First I plotted all of these 10 runs, to get a feeling for the variance of these runs of 50 iterations.



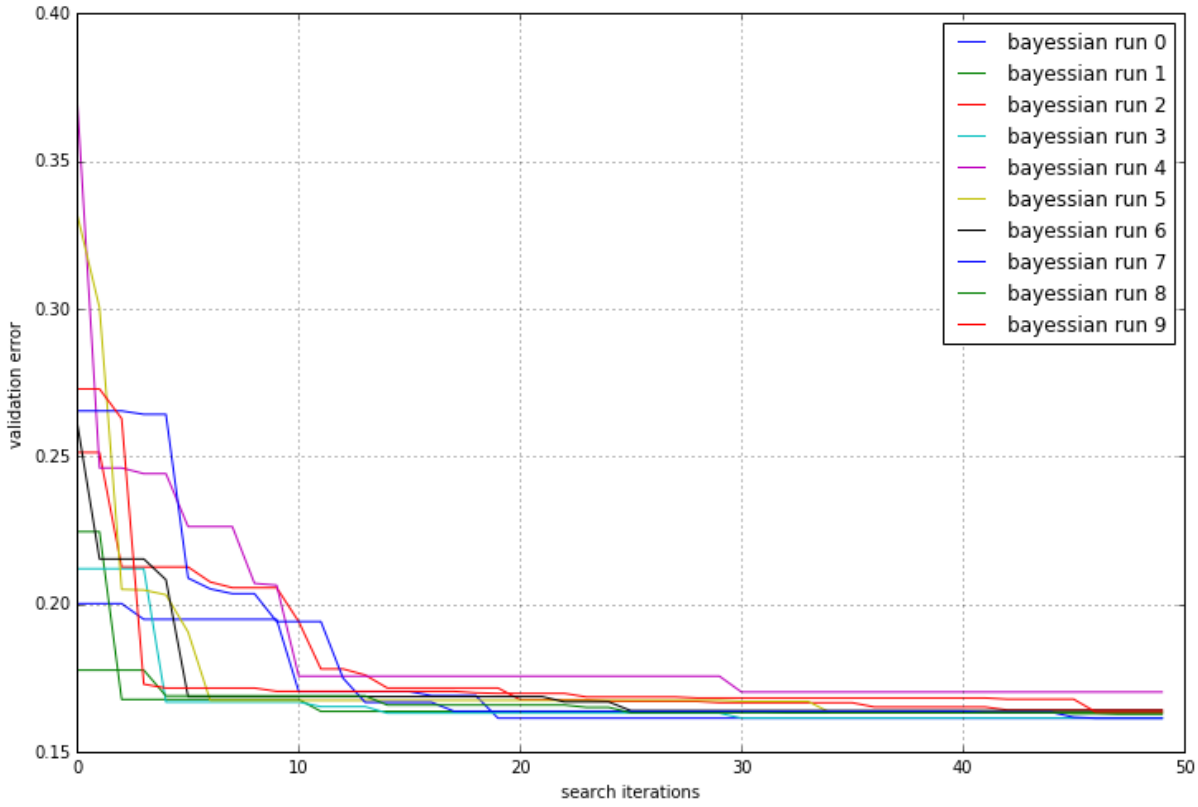
Observations

Obviously there is a large variance in the early loops, while after 20 loops all the runs arrived at a very similar level of ~0.17, which seems already to be close the optimum, as no run reached a significantly lower level.

With regard to the observed best configurations, the observed best learning rates were all in a range between 2.8 and 3.0, Layer 2 and 3 were relatively close to 10 (max possible value). For layer 2 and batch size the observed results varied in a relatively large range, indicating, that the performance as indicated by the surrogate function seems to be less sensitive for these parameters.

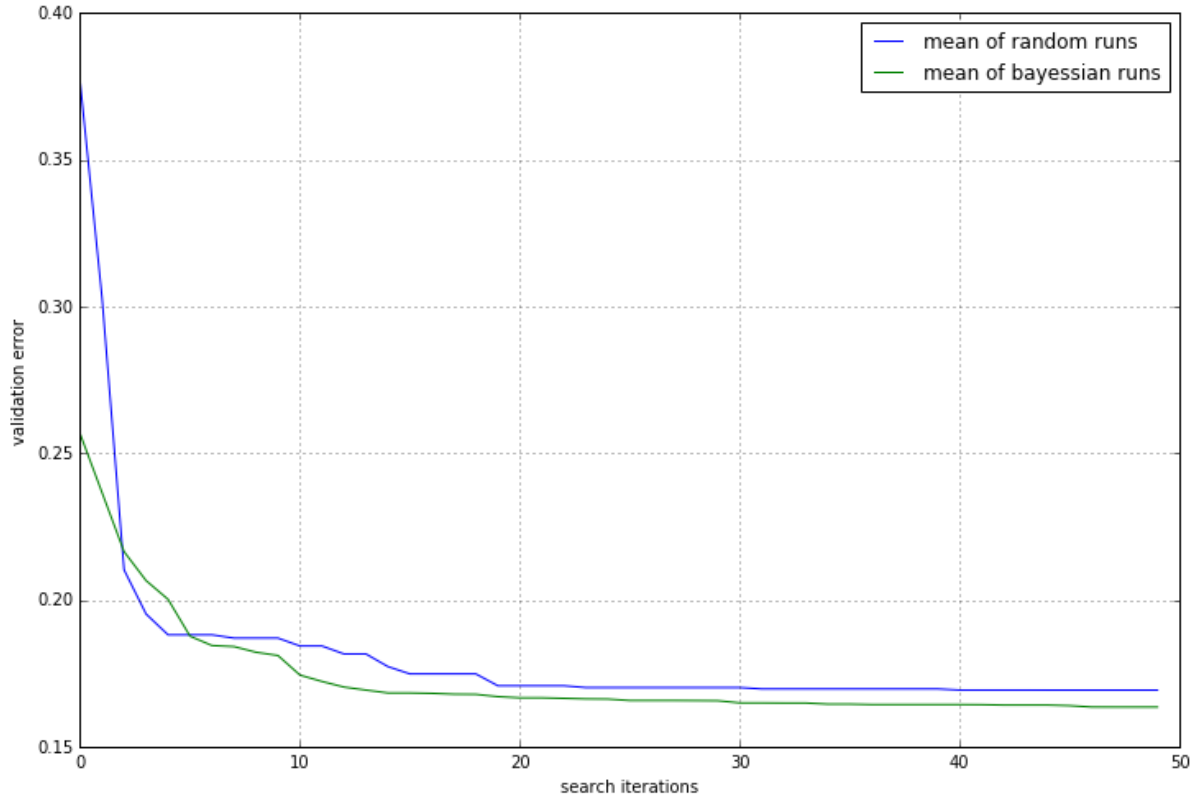
Task 2, Bayesian search

As for the random I ran the Bayesian optimiser on the surrogate objective function 10 times (50 iterations each), and plotted the 10 lists of the incumbents.



Compared to the random search the validation error decreases somewhat faster: after 12 iterations all of the runs reached a level of validation error below ~1,8.

Then I calculated the mean of both the 10 random runs and of the 10 runs with Bayesian optimisation.



Observations

During the first 5 iterations the validation error of the average random search decreases somewhat faster, after more than ~5 runs the error of Bayesian runs decreases faster, and reaches at a slightly lower level. Beyond the random noise in the results there is a clear trend of the Bayesian optimisation outperforming the random algorithm after more than 5 runs.

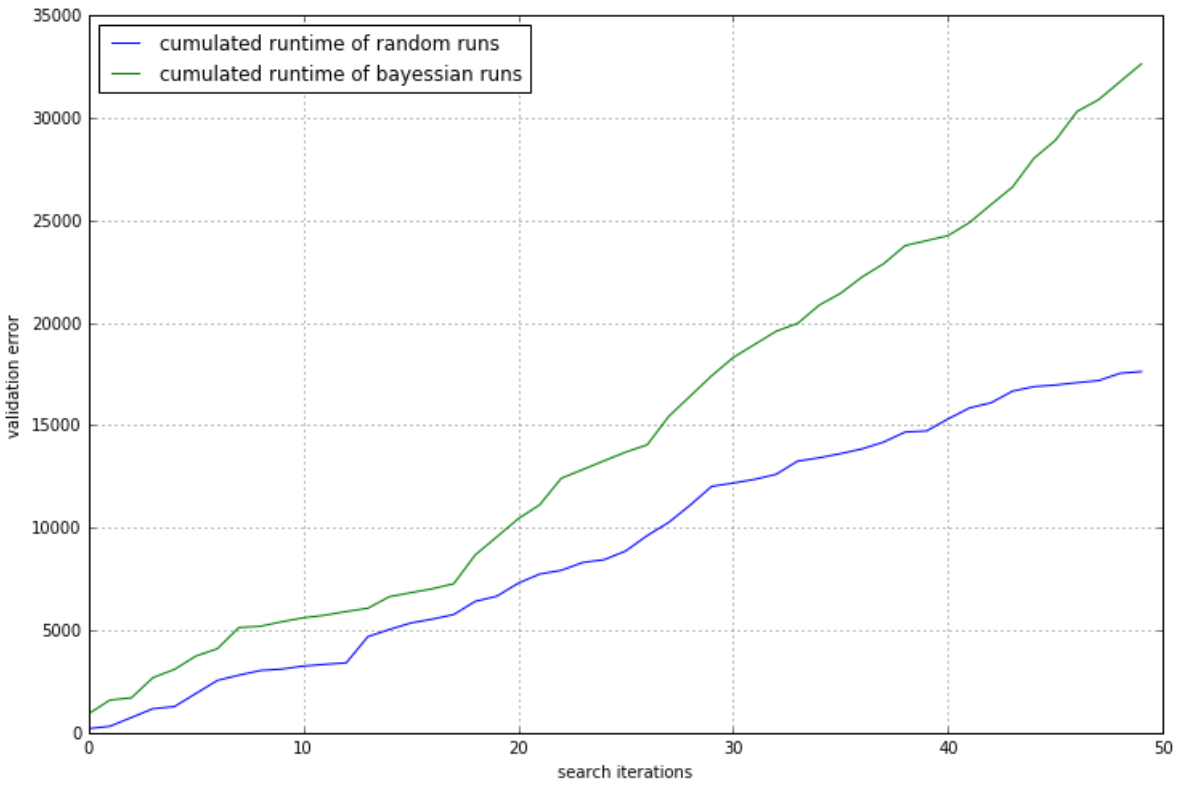
Explanations

The observed behavior might be because the "acquisition function" of the Bayesian optimisation algorithm looks for a trade-off between exploration and optimisation. In the early steps of this process, there might be a high incentive to look into unknown / far-off lying regions of the landscape in order to establish a "helicopter" view before trying to optimise within some best region (~first 5 steps)

After doing so, when putting a focus on the preferred region, it can optmise its search with some guidance from the previous results, while the random search continues to search "blindly" in the total parameter space and waiting for better results to happen accidentally.

Task 2, runtime

I calculated the runtime for both methods for the first of the 10 loops. For the Bayesian optimisation I queried the actually tested configurations “X” and passed them to the runtime surrogate function.



Observations

One loop of 50 iterations would require ~34.000 seconds (9.5 hours) for the Bayesian optimisation, and ~17.500 seconds (~5 hours) for the random search.

The runtime of the Bayesian optimisation is nearly twice as long as for the random optimisation.

The reason might be, that the Bayesian optimisation finds a best configuration with a first hidden layer, which is roughly 7 times larger than the one found by random optimisation (8.09 versus 5.29 on a log2 scale - see printout "best found configurations 2 cells above"). This results in a significantly larger total number of parameters, as can be verified via single test runs with the runtime surrogate function.

One would assume, that an efficient hyperparameter optimisation function would always tend to models with a high number of layers (? overfitting ?), as is the case here, where the Bayesian optimiser places all layer sizes near the maximum of 2^10. The drawback of slowing down the model strongly is not "priced in" here.

It would be interesting, to add the number of layers as a hyperparameter (here fixed to 3), and eventually somehow incentivise sparse models, which are faster to train. One might for example ask this way: given a limited number of weight parameters, what would be the smartest way to arrange them to achieve the best results (variable number and sizes of layers) ?