Report Exercise 6, Hyperparameter Optimization II

Task 1, implementing configuration space

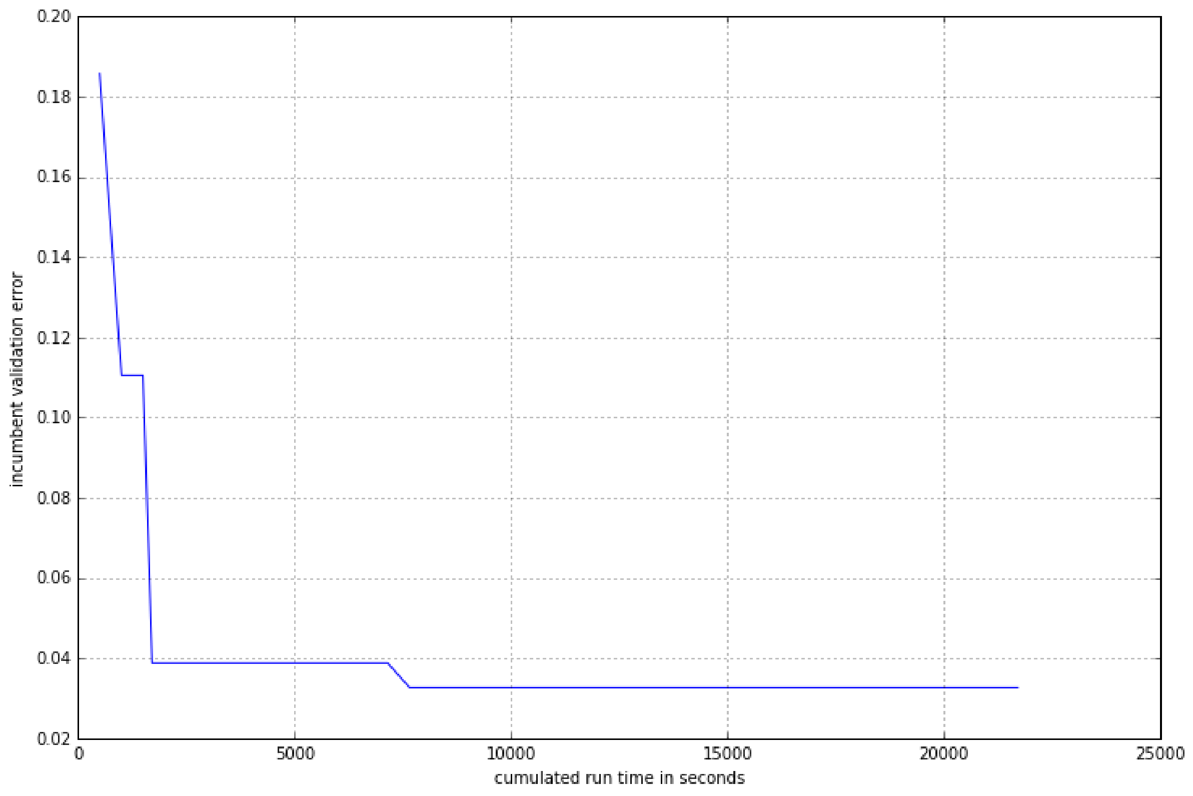
I looked into some examples generated by create_config_space(), and checked if the mechanism for conditional variables works properly. Everything looked reasonable so far.

Task 2, implementing configuration space

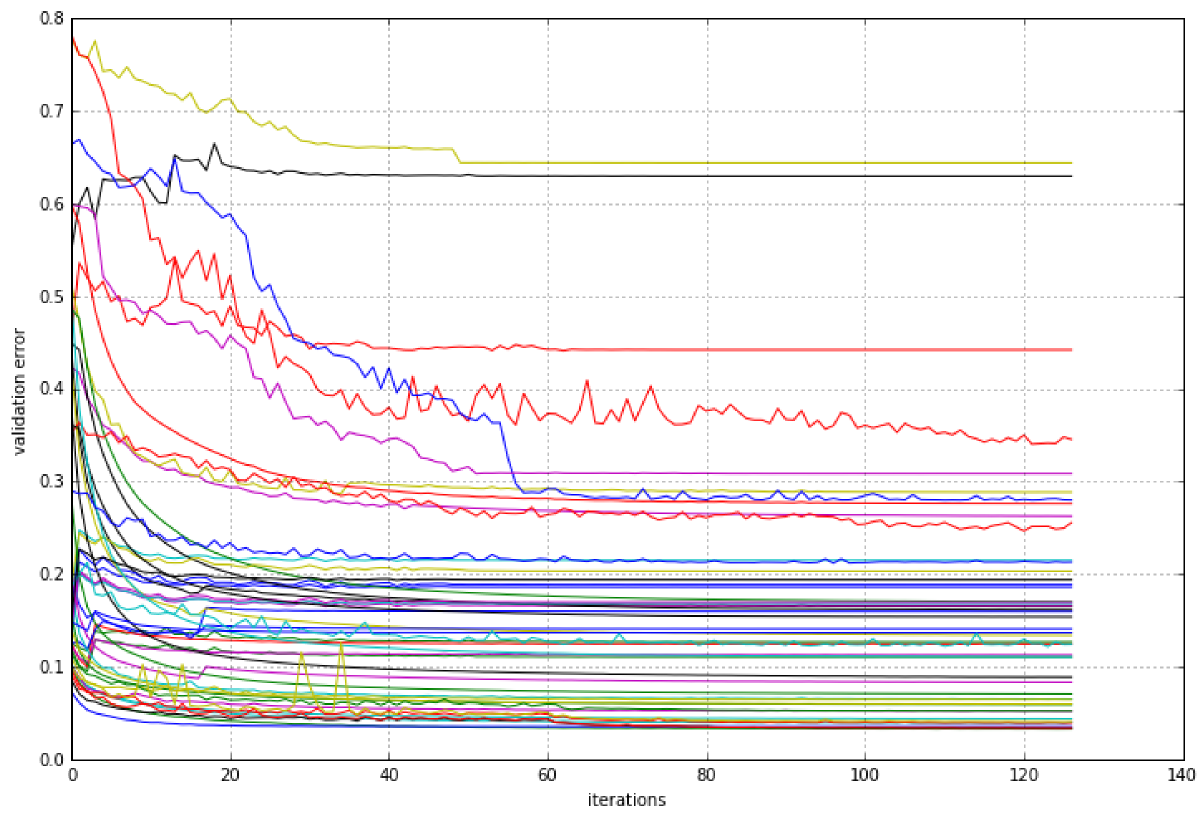
Calling smac.optimize() caused the kernel to halt with a "cannot allocate memory" message. After setting the memory of the virtual machine from 2 to 4 GB it worked, but still required a complete restart of the kernel after each call of smac.optimize(). For convenience I separated the notebook code into different blocks.

Running smac with 50 configurations, at 127 iterations each, yielded the following results:

Best found validation error (incumbent) over cumulated runtime with smac:



The 50 individual learning curves of the smac run:



Observations

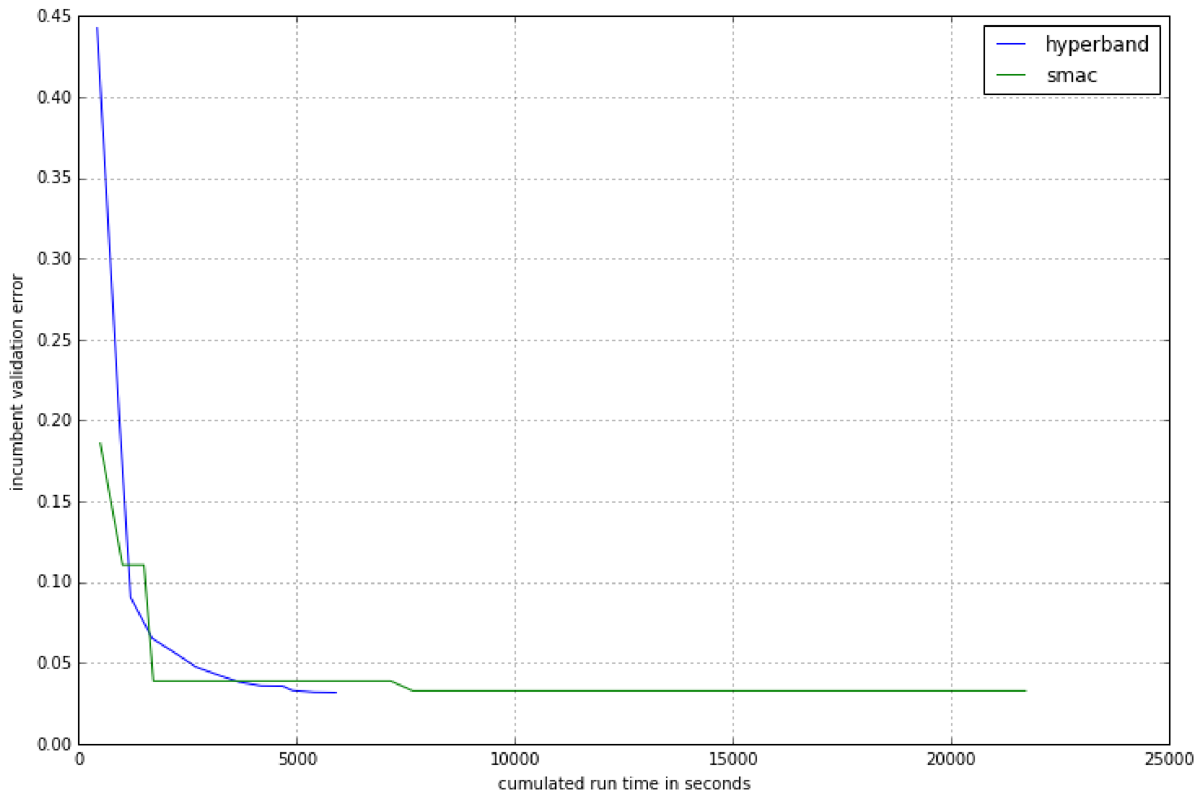
As expected there are large differences between the performances of the different configurations. However the samples are concentrated in the region of the relatively high performers, probably demonstrating the guiding effect of the Bayesian algorithm used by smac.

When looking at the 50 individual learning curves one can see, that the final performance of a configuration is quite strongly related to the performance in the beginning of the run: the best ones are generally the best from start. This behavior is an important assumption made by the Hyperband algorithm.

Task 3, Hyperband

As in the smac implementation Hyperband runs at up to 127 epochs per configuration – but other than with smac, Hyperband uses this as an upper limit, and stops most of the runs much earlier, thus optimising a lot wrt to speed (the plot of the single learning curves above gives an idea of the time wasted with runs, which have no reasonable chance to become the best ones).

Incumbent trajectory of Hyperband compared to smac:



Observations

At the given example the hyperband algorithm does not outperform the smac algorithm significantly with regard to speed – in contrast to the impressive over-performance of the examples on the website of the Berkeley team Lisha Li, Kevin Jamieson et.al.

This might be because of an accidental early “lucky shot” of smac to a low validation error of ~0.04 in the observed case. Anyhow the hyperband algorithm descends very much smoother because after starting it explores a much larger number of configurations in shorter time. It will therefore probably decrease quickly in a more reliable fashion than smac, despite using random search instead of a guided search as smac.

The length of the trajectory, and thus the number of incumbents encountered during ~6.000 seconds, was 13, compared to 4 with smac over the time of ~22.000 seconds.

Hyperband tested a total number 487 of configurations compared to the 50 “n-iters” with smac: the configuration space was thus explored in a much more extensive fashion in much shorter time.

Combining the Bayesian guided search of smac with the methods of hyperband seems to be a promising approach (I could not figure out if “Hyperband on steroids” already does exactly this…).