Deep Learning Lab Course
Machine Learning and Computer Vision Track
Exercise 6

Aaron Klein
(send the report to kleinaa@cs.uni-freiburg.de)

Due: January 9, 2018

In this exercise we will optimize the hyperparameters and the architecture choices of a fully connected neural network for MNIST. Send us a small report (1 or 2 pages) with the below described plots. As in the previous exercise, instead of optimizing the true objective function we optimize a surrogate function to save computation time during the developing phase. You can find the surrogate together with a code template that shows how to run SMAC and Hyperband here

1 Configuration Space

The first part of this exercise is to implement the configuration space in Table 1 that defines the input search space for our optimizer by providing the bounds and type of each hyperparameter. Compared to the configuration space from the previous exercise which was purely continuous, we now also have discrete and categorical hyperparameters . Additionally, our configuration space has hyperparameters that are conditionally dependent on other hyperparameters, which means they are only active if the other hyperparameter is set to a specific value. For instance the dropout rate of layer 2 is only active if num_layer is $\geq$ 2.

We will use the ConfigSpace python package to implement our configuration space. Check out the example python script in the zip file to see how you can use it.

Note: The hyperparameters 'loss_function' and 'output_activation' are constant hyperparameters, and would actually be dropped from the configuration space. The only reason we include them here is because they were part of the training data for the surrogate, thus we have to provide them for the surrogate.

Name	Range	Default	log scale	Type	conditioned on
Adam_final_lr_fraction	$[10^{-4}, 1.0]$	10^{-2}	✓	float	optimizer = Adam
Adam_initial_lr	$[10^{-4}, 10^{-2}]$	10^{-3}	✓	float	optimizer = Adam
SGD_final_lr_fraction	$[10^{-4}, 1.0]$	10^{-2}	✓	float	optimizer = SGD
SGD_initial_lr	$[10^{-3}, 0.5]$	10^{-1}	✓	float	optimizer = SGD
SGD_momentum	$[0.0, 0.99]$	0.9	-	float	optimizer = SGD
StepDecay_epochs_per_step	$[1, 128]$	16	✓	int	learning_rate_schedule = StepDecay
activation	{relu, tanh}	relu	-	cat	-
batch_size	$[8, 256]$	16	✓	int	-
dropout_0	$[0.0, 0.5]$	0.0	-	float	-
dropout_1	$[0.0, 0.5]$	0.0	-	float	num_layer ≥ 2
dropout_2	$[0.0, 0.5]$	0.0	-	float	num_layer ≥ 3
dropout_3	$[0.0, 0.5]$	0.0	-	float	num_layer = 4
l2_reg_0	$[10^{-6}, 10^{-2}]$	10^{-4}	✓	float	-
l2_reg_1	$[10^{-6}, 10^{-2}]$	10^{-4}	✓	float	num_layer ≥ 2
l2_reg_2	$[10^{-6}, 10^{-2}]$	10^{-4}	✓	float	num_layer ≥ 3
l2_reg_3	$[10^{-6}, 10^{-2}]$	10^{-4}	✓	float	num_layer = 4
learning_rate_schedule	{ExponentialDecay, StepDecay}	ExponentialDecay	-	cat	-
loss_function	{categorical_crossentropy}	categorical_crossentropy	-	cat	-
num_layers	$[1, 4]$	2	-	int	-
num_units_0	$[16, 256]$	32	✓	int	-
num_units_1	$[16, 256]$	32	✓	int	num_layer ≥ 2
num_units_2	$[16, 256]$	32	✓	int	num_layer ≥ 3
num_units_3	$[16, 256]$	32	✓	int	num_layer = 4
optimizer	{Adam, SGD }	Adam	-	cat	-
output_activation	{softmax}	softmax	-	cat	-

2 SMAC

After implementing the configuration space we will use SMAC to find a good hyperparameter configuration. SMAC is an Bayesian optimization method that uses a random forest to model the objective function such that it is able to model mixed continuous, categorical and discrete configuration spaces. Have a look at its documentation on the github repository to see how you can install it.

After running SMAC, plot the validation error of the incumbent on the y-axis and the estimated wall-clock time on the x-axis. The python code already contains the functionality to extract the incumbent trajectory from SMAC.

3 Hyperband

In the last part of the exercise we will speed up the optimization procedure by using Hyperband. We will use our implementation (HpBandSter) of Hyperband. Have a look on the github documentation to see how you can install it.

Run Hyperband with the python code template and plot the incumbent trajectory to compare it with SMAC. How does it compare to SMAC? How much faster is it by using early stopping of learning curves?

In a second figure plot all the learning curves SMAC evaluated in one subfigure and all the learning curves Hyperband evaluated in another subfigure.