Example: A Classical Algorithm Configuration Approach to Dynamic Learning Rate Configuration

Steven Adriaensen[1] **Göktuğ Karakaşlı**[1]

[1]University of Freiburg

Here we solve DAC **?** by reduction to a classical (static) algorithm configuration problem. Specifically, we statically tune the parameters of hand-crafted dynamic learning rate schedulers using SMAC **?**. We illustrate this approach for four different schedulers (e.g., see `schedulers.py`):

`ConstantLRPolicy`: A scheduler using the same learning rate (configurable parameter) throughout the training process.

`CosineAnnealingLRPolicy`: A cosine annealing scheduler **?**. Starting from an initial learning rate (configurable parameter) reduces the learning rate to 0 following a re-scaled half cosine curve.

`SimpleReactivePolicy`: A novel simple scheduler using a constant learning rate (configurable parameter) for the first $2n$ steps (configurable parameter). After every $(2 + i)n^{\text{th}}$ step ($i \in \mathbb{N}$), if the average mini-batch loss during the last n steps was lower than that in the previous $2n$ steps, it *increases* the learning rate with a factor a (configurable parameter). If the loss is higher, it *decreases* the learning rate with a factor b (configurable parameter).

`ReduceLROnPlateauPolicy`: A scheduler emulating the ReduceLROnPlateau pytorch scheduler. It adjusts the learning rate per epoch, based on the validation loss (observed at the end of every epoch). In particular, it reduces the learning rate with a 'factor' if the validation loss has stagnated for 'patience' epochs. This scheduler has many parameters, but we only optimized the initial learning rate, 'patience', 'factor', and used the pytorch defaults for the others.

We provide the code to configure these schedulers using SMAC (see `train.py`) as well as the resulting optimized configurations (see `tmp/saved_configs`) on 1000 training instances (environment seed was 42) after 5000 target algorithm evaluations (smac seed was 42). Note that the training code requires installing SMAC3 (e.g., calling `pip install -r requirements_train.txt`).

