

CoRe Optimizer

CLASS core_optimizer.CoRe(*params*, *lr*=1e-3, *betas*=(0.7375, 0.8125, 250.0, 0.99), *eps*=1e-8, *etas*=(0.7375, 1.2), *step_sizes*=(1e-6, 1e-2), *weight_decay*=0.1, *score_history*=0, *frozen*=0, *, *foreach*=None, *maximize*=False)

Implements the Continual Resilient (CoRe) optimizer.

input : θ (*params*), $f(\theta)$ (objective), $\beta_1^a, \beta_1^b, \beta_1^c, \beta_2$ (*betas*), ϵ (*eps*),
 η_-, η_+ (*etas*), s_{\min}, s_{\max} (step sizes), d (weight decay),
 t_{hist} (score history), n_{frozen} (*frozen*)

initialize : $s_0 \leftarrow \text{lr}$, $g_0 \leftarrow 0$, $h_0 \leftarrow 0$, $S_0 \leftarrow 0$

for $t = 1$ **to** ... **do**

$G_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$

if maximize

$G_t \leftarrow -G_t$

$\beta_{1,t} \leftarrow \beta_1^b + (\beta_1^a - \beta_1^b) \exp\{-(t-1)/\beta_1^c\}$

$g_t \leftarrow \beta_{1,t} g_{t-1} + (1 - \beta_{1,t}) G_t$

if $\beta_2 \geq 0$

$h_t \leftarrow \beta_2 h_{t-1} + (1 - \beta_2) G_t^2$

$P_t \leftarrow 1$

if $t_{\text{hist}} > 0 \wedge t > t_{\text{hist}} \wedge S_{t-1} \text{ top-}n_{\text{frozen}} \text{ in } \mathbf{S}_{t-1}$

$P_t \leftarrow 0$

if $g_{t-1} g_t P_t > 0$

$s_t \leftarrow \min(\eta_+ s_{t-1}, s_{\max})$

else if $g_{t-1} g_t P_t < 0$

$s_t \leftarrow \max(\eta_- s_{t-1}, s_{\min})$

else

$s_t \leftarrow s_{t-1}$

if $\beta_2 \geq 0$

$u_t \leftarrow g_t / (1 - \beta_{1,t}^t) / \{[h_t / (1 - \beta_2^t)]^{0.5} + \epsilon\}$

else

$u_t \leftarrow \text{sgn}(g_t)$

if $t_{\text{hist}} > 0$

if $t \leq t_{\text{hist}}$

$S_t \leftarrow S_{t-1} + t_{\text{hist}}^{-1} g_t u_t P_t s_t$

else

$S_t \leftarrow (1 - t_{\text{hist}}^{-1}) S_{\xi}^{\tau-1} + t_{\text{hist}}^{-1} g_t u_t P_t s_t$

$\theta_t \leftarrow (1 - d|u_t|P_t s_t)\theta_{t-1} - u_t P_t s_t$

return θ_t

For further details regarding the algorithm we refer to the papers [Lifelong Machine Learning Potentials](#) and [CoRe Optimizer: An All-in-One Solution for Machine Learning](#).

Parameters:

- **params** (*iterable*): iterable of parameters to optimize or dicts defining parameter groups
- **lr** (*float, optional*): learning rate (default: 1e-3)
- **betas** (*Tuple[float, float, float, float], optional*): coefficients (beta1a, beta1b, beta1c, beta2) used for computing running averages of gradient and its square (default: (0.7375, 0.8125, 250.0, 0.99))
- **eps** (*float, optional*): term added to the denominator to improve numerical stability (default: 1e-8)
- **etas** (*Tuple[float, float], optional*): pair of (etaminus, etaplus) that are multiplicative increase and decrease factors (default: (0.7375, 1.2))
- **step_sizes** (*Tuple[float, float], optional*): a pair of minimal and maximal allowed step sizes (default: (1e-6, 1e-2))
- **weight_decay** (*float or List[float], optional*): weight decay for all parameters or list of weight decays for parameter groups (default: 0.1)
- **score_history** (*int, optional*): number of epochs to build the score history before applying plasticity factors (default: 0)
- **frozen** (*int or List[int], optional*): number of parameters frozen by the plasticity factors in each parameter group (individual specifications for each parameter group can be provided by a list (default: 0))
- **foreach** (*bool, optional*): whether foreach implementation of optimizer is used. If unspecified by the user (so foreach is None), we will try to use foreach over the for-loop implementation on CUDA, since it is usually significantly more performant. Note that the foreach implementation uses $\sim \text{sizeof}(\text{params})$ more peak memory than the for-loop version due to the intermediates being a tensorlist vs just one tensor. If memory is prohibitive, batch fewer parameters through the optimizer at a time or switch this flag to False (default: None)
- **maximize** (*bool, optional*): maximize the params based on the objective, instead of minimizing (default: False)