CoRe Optimizer

CLASS core_optimizer.CoRe(*params, lr=1e-3, step_sizes=(1e-6, 1e-2), etas=(0.7375, 1.2), betas=(0.7375, 0.8125, 250.0, 0.99), eps=1e-8, weight_decay=0.1, score_history=0, frozen=0.0, *, maximize=False, foreach=None*)

Implements the Continual Resilient (CoRe) optimizer.

---

input : θ (params), $f(\theta)$ (objective), $s_{\min}, s_{\max}$ (step sizes),
η_-, η_+ (etas), $\beta_1^{\mathrm{a}}, \beta_1^{\mathrm{b}}, \beta_1^{\mathrm{c}}, \beta_2$ (betas), ϵ (eps),
d (weight decay), t_{hist} (score history), p_{frozen} (frozen)
initialize : $s_0 \leftarrow \mathrm{lr},\ g_0 \leftarrow 0,\ h_0 \leftarrow 0,\ S_0 \leftarrow 0$

---

for $t = 1$ **to** $\ldots$ **do**
  $G_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  if maximize
    $G_t \leftarrow -G_t$
  $\beta_{1,t} \leftarrow \beta_1^{\mathrm{b}} + (\beta_1^{\mathrm{a}} - \beta_1^{\mathrm{b}}) \exp\{-[(t-1)/\beta_1^{\mathrm{c}}]^2\}$
  $g_t \leftarrow \beta_{1,t} g_{t-1} + (1 - \beta_{1,t}) G_t$
  if $\beta_2 \geq 0$
    $h_t \leftarrow \beta_2 h_{t-1} + (1 - \beta_2) G_t^2$
  $P_t \leftarrow 1$
  if $t_{\mathrm{hist}} > 0 \wedge t > t_{\mathrm{hist}} \wedge S_{t-1}$ top-p_{frozen} 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
    $u_t \leftarrow \mathrm{sgn}(g_t)$
  if $t_{\mathrm{hist}} > 0$
    if $t \leq t_{\mathrm{hist}}$
      $S_t \leftarrow S_{t-1} + t_{\mathrm{hist}}^{-1} g_t u_t P_t s_t$
    else
      $S_t \leftarrow (1 - t_{\mathrm{hist}}^{-1}) S_\xi^{\tau-1} + t_{\mathrm{hist}}^{-1} g_t u_t P_t s_t$
  $\theta_t \leftarrow (1 - d|u_t| P_t s_t) \theta_{t-1} - u_t P_t s_t$

---

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 to set initial step size (default: 1e-3)
- **step_sizes** *(Tuple[float, float], optional)*: pair of minimal and maximal allowed step sizes (recommendation: maximal step size of 1e-3 for mini-batch learning, 1.0 for batch learning, and 1e-2 for intermediate cases) (default: (1e-6, 1e-2))
- **etas** *(Tuple[float, float], optional)*: pair of etaminus and etaplus that are multiplicative increase and decrease factors (default: (0.7375, 1.2))
- **betas** *(Tuple[float, float, float, float], optional)*: coefficients beta1a, beta1b, beta1c, and 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)
- **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 optimization steps to build the score history before applying plasticity factors (default: 0)
- **frozen** *(float or List[float], optional)*: fraction of all parameters frozen by the plasticity factors or list of fractions for parameter groups (applies if score_history > 0) (default: 0.0)
- **maximize** *(bool, optional)*: maximize the objective with respect to the params, instead of minimizing (default: False)
- **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 $\sim$ sizeof(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)