Optimizers

BaseSGDOptimizer ¶

class paddle.trainer_config_helpers.optimizers.**BaseSGDOptimizer**

SGD Optimizer.

SGD is an optimization method, trying to find a neural network that minimize the "cost/error" of it by iteration. In paddle's implementation SGD Optimizer is synchronized, which means all gradients will be wait to calculate and reduced into one gradient, then do optimize operation.

The neural network consider the learning problem of minimizing an objective function, that has the form of a sum

$$Q(w) = \sum_{i}^{n} Q_i(w)$$

The value of function Q sometimes is the cost of neural network (Mean Square Error between prediction and label for example). The function Q is parametrised by w, the weight/bias of neural network. And weights is what to be learned. The i is the i-th observation in (trainning) data.

So, the SGD method will optimize the weight by

$$w = w - \eta \nabla Q(w) = w - \eta \sum_{i}^{n} \nabla Q_i(w)$$

where η is learning rate. And n is batch size.

MomentumOptimizer

class paddle.trainer_config_helpers.optimizers.**MomentumOptimizer**(*momentum=None*, *sparse=False*)

MomentumOptimizer.

When sparse=True, the update scheme:

$$\begin{aligned}
\alpha_t &= \alpha_{t-1} / k \\
\beta_t &= \beta_{t-1} / (1 + \lambda \gamma_t) \\
u_t &= u_{t-1} - \alpha_t \gamma_t g_t \\
v_t &= v_{t-1} + \tau_{t-1} \alpha_t \gamma_t g_t \\
\tau_t &= \tau_{t-1} + \beta_t / \alpha_t
\end{aligned}$$

where k is momentum, λ is decay rate, γ_t is learning rate at the t'th step.

Parameters: **sparse** (*bool*) — with sparse support or not.

AdamOptimizer

class paddle.trainer_config_helpers.optimizers.**AdamOptimizer**(*beta1=0.9*, *beta2=0.999*, *epsilon=1e-08*)

Adam optimizer. The details of please refer Adam: A Method for Stochastic Optimization

$$m(w, t) = \beta_1 m(w, t-1) + (1 - \beta_1)\nabla Q_i(w)$$
$$v(w, t) = \beta_2 v(w, t-1) + (1 - \beta_2)(\nabla Q_i(w))^2$$
$$w = w - \frac{\eta}{\sqrt{v(w, t) + \epsilon}}$$

Parameters:
- **beta1** (*float*) — the β_1 in equation.
- **beta2** (*float*) — the β_2 in equation.
- **epsilon** (*float*) — the ϵ in equation. It is used to prevent divided by zero.

AdamaxOptimizer

class paddle.trainer_config_helpers.optimizers.**AdamaxOptimizer**(*beta1*, *beta2*)

Adamax optimizer.

The details of please refer this Adam: A Method for Stochastic Optimization

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \nabla Q_i(w)$$
$$u_t = max(\beta_2 * u_{t-1}, abs(\nabla Q_i(w)))$$
$$w_t = w_{t-1} - (\eta/(1 - \beta_1^t)) * m_t/u_t$$

Parameters:
- **beta1** (*float*) — the β_1 in the equation.
- **beta2** (*float*) — the β_2 in the equation.

AdaGradOptimizer

class paddle.trainer_config_helpers.optimizers.**AdaGradOptimizer**

Adagrad(for ADAptive GRAdient algorithm) optimizer.

For details please refer this Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.

$$G = \sum_{\tau=1}^{t} g_\tau g_\tau^T$$
$$w = w - \eta diag(G)^{-\frac{1}{2}} \circ g$$

DecayedAdaGradOptimizer

class paddle.trainer_config_helpers.optimizers.**DecayedAdaGradOptimizer**(*rho=0.95*, *epsilon=1e-06*)

AdaGrad method with decayed sum gradients. The equations of this method show as follow.

$$E(g_t^2) = \rho * E(g_{t-1}^2) + (1 - \rho) * g^2$$
$$learning\_rate = 1/sqrt((E(g_t^2) + \epsilon)$$

Parameters:
- **rho** (*float*) — The ρ parameter in that equation
- **epsilon** (*float*) — The ϵ parameter in that equation.

AdaDeltaOptimizer

class paddle.trainer_config_helpers.optimizers.**AdaDeltaOptimizer**(*rho=0.95*, *epsilon=1e-06*)

AdaDelta method. The details of adadelta please refer to this ADADELTA: AN ADAPTIVE LEARNING RATE METHOD.

$$E(g_t^2) = \rho * E(g_{t-1}^2) + (1 - \rho) * g^2$$
$$learning\_rate = sqrt((E(dx_{t-1}^2) + \epsilon)/(E(g_t^2) + \epsilon))$$
$$E(dx_t^2) = \rho * E(dx_{t-1}^2) + (1 - \rho) * (-g * learning\_rate)^2$$

Parameters:
- **rho** (*float*) — ρ in equation
- **epsilon** (*float*) — ρ in equation

RMSPropOptimizer

class paddle.trainer_config_helpers.optimizers.**RMSPropOptimizer**(*rho=0.95*, *epsilon=1e-06*)

RMSProp(for Root Mean Square Propagation) optimizer. For details please refer this slide.

The equations of this method as follows:

$$v(w, t) = \rho v(w, t-1) + (1 - \rho)(\nabla Q_i(w))^2$$
$$w = w - \frac{\eta}{\sqrt{v(w,t) + \epsilon}} \nabla Q_i(w)$$

Parameters:
- **rho** (*float*) — the ρ in the equation. The forgetting factor.
- **epsilon** (*float*) — the ϵ in the equation.

settings

paddle.trainer_config_helpers.optimizers.**settings**(**args*, ***kwargs*)

Set the optimization method, learning rate, batch size, and other training settings. The currently supported algorithms are SGD and Async-SGD.

Warning: Note that the 'batch_size' in PaddlePaddle is not equal to global training batch size. It represents the single training process's batch size. If you use N processes to train one model, for example use three GPU machines, the global batch size is N*'batch_size'.

Parameters:

- **batch_size** (*int*) — batch size for one training process.
- **learning_rate** (*float*) — learning rate for SGD
- **learning_method** (*BaseSGDOptimizer*) — The extension optimization algorithms of gradient descent, such as momentum, adagrad, rmsprop, etc. Note that it should be instance with base type BaseSGDOptimizer.
- **regularization** (*BaseRegularization*) — The regularization method.
- **is_async** (*bool*) — Is Async-SGD or not. Default value is False.
- **model_average** (*ModelAverage*) — Model Average Settings.
- **gradient_clipping_threshold** (*float*) — gradient clipping threshold. If gradient value larger than some value, will be clipped.