Optimization Functions

27 May 2018 16:51

Parameter Update

Vanilla Update

Change the parameters along the negative gradient direction

$w_i = w_i - \lambda \frac{\partial C}{\partial w_i}$ $\quad$ ***where λ is learning Rate***.

A small $\boldsymbol{\lambda}$ guarantess the progress in non – negative progress on loss function, towards minima.
But small learning rate increases the traiing time.

Momentum Update

$\boldsymbol{v = u \cdot v - \lambda \frac{\partial c}{\partial \omega_i}}$ **where u is momentum**

$\boldsymbol{w_i = w_i + v}$

A typical momentum annealing setting is to start with momentum of about 0.5 and anneal it to 0.99 or so over multiple epochs at later stages.

Nestrov Momentum

Stronger theoretical converge guarantees for convex functions

$\boldsymbol{w_{ahead} = w + u * v}$

$\boldsymbol{v = u * v - \lambda * dw_{ahead}}$

$\boldsymbol{w = w + v}$

Rewriting the above equations (update in terms of $\boldsymbol{w_{ahead}}$ instaed $\boldsymbol{w}$)
$\boldsymbol{v_{prev} = v}$

$\boldsymbol{v = u * v - \lambda * dw}$

$\boldsymbol{w = w - u * v_{prev} + (1 + u) * v}$

Per-parameter adaptive learning rate methods

Adagrad

$$C = C + \mathrm{d}w^2$$

$$w = \omega - \lambda * \frac{\mathrm{d}w}{\sqrt{C} + \varepsilon} \qquad \varepsilon \textit{ to avoid division by } 0$$

RMSprop

$$C = \beta * C + (1 - \beta) * \mathrm{d}w^2 \quad \text{where } \beta \textit{ is decay rate}$$

$$w = \mathrm{w} - \frac{\lambda * \mathrm{dw}}{\sqrt{c} + \varepsilon}$$

Hence, RMSProp still modulates the learning rate of each weight based on the magnitudes of its gradients, which has a beneficial equalizing effect, but unlike Adagrad the updates do not get monotonically smaller.

Adam

$$m = \beta_1 * m + (1 - \beta_1) * \mathrm{d}w \quad \# \textit{ smooth version of gradient}$$

$$v = \beta_2 * v + (1 - \beta_2) * \mathrm{d}w^2$$

$$w = \mathrm{w} - \lambda * \frac{\mathrm{m}}{\sqrt{v} + \varepsilon} \qquad \varepsilon \textit{ to avoid division by } 0$$

Recommended Values of
$\varepsilon = 1\mathrm{e} - 8$
$\beta_1 = 0.9$
$\beta_2 = 0.999$

Learning Rate Decay

- **Step decay**: Reduce the learning rate by some factor every few epochs. Typical values might be reducing the learning rate by a half every 5 epochs, or by 0.1 every 20 epochs.
- **Exponential decay.** has the mathematical form $\alpha = \alpha_0 \mathrm{e}^{-kt}$, where α_0,k are hyper parameters and t is the iteration number (but you can also use units of epochs).
- **1/t decay** has the mathematical form $\alpha = \frac{\alpha_0}{(1+kt)}$ where α_0,k are hyper parameters and t is the iteration number.

Source <http://cs231n.github.io/neural-networks-3/>