Blog about optimizers

SGD: $w_{t+1} = w_t - \eta \frac{dL}{dw_t}$. This primitive weight update equation show that optimizers can only work around learning rate and gradient.

Momentum: $w_{t+1} = w_t - \eta * m_t$, $m_t = \beta m_{t-1} + (1-\beta)\frac{dL}{dw_t}$. Here we change gradient to exponential moving average. Beta is between 0 and 1. A large beta value indicates favor on previous gradient. Note that if η and m_t are both large and we are reaching global minimum, this might result to a saddle point. (Default value for β is 0.9)

Adagrad: $w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t+\varepsilon}} * \frac{dL}{dw_t}$, $v_t = v_{t-1} + (\frac{dL}{dw_t})^2$. In Adagrad, it keeps gradient the same as in SGD, and decay the learning rate as t increase. Since v_t will only increase. Note that if η is small at beginning, $\frac{\eta}{\sqrt{v_t+\varepsilon}}$ will soon reach to 0, and learning stops. Hence for Adagrad, a large η should be assigned. Also for features that occurs more frequent, their learning rate would be small and vice versa. Hence Adagrad works well with sparse data.

RMSProp: $w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t+\varepsilon}} * \frac{dL}{dw_t}$, $v_t = \beta v_{t-1} + (1-\beta)(\frac{dL}{dw_t})^2$. RMSProp uses exponential moving average, instead of sum in Adagrad. So for RNN, RMSProp should perform well. Adadelta works like RMSProp, but it does not need initial learning rate.

Adam: $w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t}+\epsilon} m_t$, where v_t and m_t are similar as in momentum and RMSProp. Hence Adam is the first optimizer that is adaptive on both learning rate and gradient. And has the most freedom.

AdaMax: Similar to Adam, but v_t is the maximum between βv_{t-1} and absolute value of gradient. So not much difference between Adam.

However, several papers in 2017 suggest that we should stay away from adaptive methods, and returns back to SGD with momentum due to the bad generalization of all adaptive methods.

Here are some reasons:
1. In **The Marginal Value of Adaptive Gradient Methods in Machine Learning**, it shows an example where Adagrad will always label unseen data as positive. But SGD makes no error.
2. AdamW is introduce in 2019, where it adds weight decay and keeps that from learning rate optimization separately. And shows huge improvement for generalization.

In conclusion: A SGD + momentum with finetuned hyperparameter will definitely outperform adaptive method. But the grid search for optimized learning rate, step size, batch size, decay function is time consuming. Hence most people will choose adaptive methods, and let those hyperparameters correct themselves during training.