Optimizer (옵티마이저)

Stochastic gradient descent. (SGD)
(확률적 경사 하강법)

$$W = W - \alpha \cdot \frac{dE}{dW}$$

gradient.

W: weight, α: learning rate

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J(\theta)$$

Momentum
(모멘텀)

$$W = W + \beta V - \alpha \cdot \frac{dE}{dW}$$

아무런 힘을 받지 않을때 서서히 하강시키는 역할

β (0.9) (상수), V: velocity

AdaGrad

$$h = h + \frac{dE}{dW} \odot \frac{dE}{dW}$$

원소별 곱셈. 기울기의 제곱

$$W = W - \alpha \frac{1}{\sqrt{h}} \cdot \frac{dE}{dW}$$

매개변수 W 중에 가장 많이 움직인 원소는 학습률을 낮아지게.

- 학습률 η을 정하기 위한 방법
  → η를 처음에 낮춘다 모두에게
  → 모두보단 개별 맞춤으로!
- 과거의 기울기를 제곱해서 더하기 때문에 어느 순간 갱신 량이 0에 가까워지는 현상
  (RMSprop에 해결.

Adam (RMSprop + Momentum)

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) \cdot g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) \cdot g_t^2$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t$$

$\beta_1 = 0.9$
$\beta_2 = 0.999$
$\epsilon = 10^{-8}$

- 2015년
- 하이퍼 파라미터의 편향 보정
- 지수평균 이동

Adam

과거값을 줄이는가.. How? ⇒ 지수 이동 평균을 통해!

Momentum + Rms

adagrad + Rms

$m_t = \beta_1 m_{t-1} + (1-\beta_1)\cdot g_t$

$v_t = \beta_2 v_{t-1} + (1-\beta_2)\cdot g_t^2$

$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$

$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$

초기 값의 속도가 작은경우 ($\beta \doteq 1$) $M(0), V(0) = 0$으로 초기화 되는데 이때 이동평균을 구하면 0으로 편향되는 값추정이 나오게에 보정함

(β_1^t, β_2^t는 각각 t번 곱한것)

t>>1 일때 $\hat{v}_t = \frac{v_t}{1-\beta_2^t} = \frac{v_t}{1-0}$

$(0.9)^{100} = 0$

t = step

$\alpha = 0.001$ 최적 default 값

$\beta_1 = 0.9$ = momentum decay rate

$\beta_2 = 0.999$ = adaptive term decay rate

$\epsilon = 10^{-8}$

$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}$

momentum / momentum decay rate / adaptive term decay rate

기존의 momentum과 달리 지수이동평균을 사용하게에 $(1-\beta_1)$가 상대적 작은 값을 갖음

$a_{n+1} = \beta_1 \cdot a_n + (1-\beta_1)\cdot \nabla f(x_n)$, $a_0 = 0$

$b_{n+1} = \beta_2 \cdot b_n + (1-\beta_2)\cdot \nabla f(x_n) \odot \nabla f(x_n)$, $b_0 = 0$

RMSprop

$\hat{\alpha}_{n+1} = \alpha \cdot \frac{\sqrt{1-(\beta_2)^{n+1}}}{\sqrt{1-(\beta_1)^{n+1}}}$, $\alpha = 0.1$

$x_{n+1} = x_n - \frac{\hat{\alpha}_{n+1}}{\sqrt{b_{n+1}+\epsilon}} \odot a_{n+1}$, $\epsilon = 1e-8$

기존 Momentum 과 달리 Rmsprop의 지수 이동평균 사용 → $(1-\beta_1)$이 작은 값을 가짐

Adam optimizer 최초 update 속도를 보정(올리기) 위해 다음과 같이 정의함

→ $\hat{\alpha}_{n+1} = \alpha \cdot \frac{\sqrt{1-(\beta_2)^{n+1}}}{\sqrt{1-(\beta_1)^{n+1}}}$

이때 $n >> 1$ 일때 $\hat{\alpha}_n$은 α로 수렴

ex) $n = \infty$

$\hat{\alpha}_{n+1} = \alpha \cdot \frac{\infty}{\infty}$

$= \alpha \cdot 1$

$\therefore \hat{\alpha} \doteq \alpha$

learning rate

ex) $f(x) = (2x-10)^2$ → w 한개에 대해서

init $x_0 = 0$, lr 또는 $\alpha = 0.1$, momentum decay rate $\beta_1 = 0.9$

adaptive term decay rate $\beta_2 = 0.999$

$\nabla f(x) = 2(2x-10)\cdot 2 = 4(2x-10)$

- $a_1 = (0.9)\cdot 0 + (1-0.9)\cdot 4(2x_0-10)$, $(x_0=0)$, $(a_0=0)$

  $= -4$

- $b_1 = (0.999)\cdot 0 + (1-0.999)\cdot 16(2x_0-10)^2$, $(x_0=0)$ $(b_0=0)$

  $= 0.001 \cdot 1600$

  $= 1.6$

- $\hat{\alpha}_1 = (0.1)\cdot \frac{\sqrt{1-(0.999)^1}}{\sqrt{1-(0.9)^1}}$, $\alpha = 0.1$

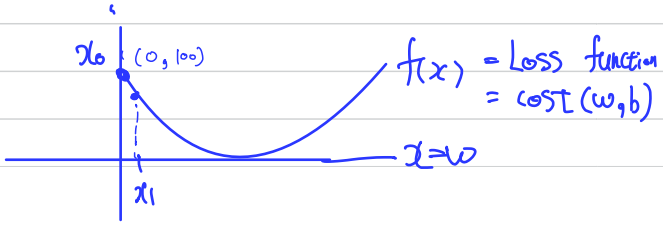
  $= 0.0158..$ ($n=\infty$ 이면 $\frac{\infty}{\infty}=1$로 $\hat{\alpha}=\alpha\cdot 1$로 수렴) $\frac{\sqrt{0.001}}{\sqrt{0.1}} = \sqrt{\frac{\frac{1}{1000}}{\frac{1}{10}}} = \sqrt{\frac{1}{100}} = 0.1$

- $x_1 = x_0 - \frac{\hat{\alpha}_1}{\sqrt{b_1+\epsilon}} \odot a_1$, $\epsilon = 1e-8$

  $= 0 - \frac{0.0158}{\sqrt{1.6+\epsilon}}\cdot(-4)$

  $= \frac{0.0632}{\sqrt{1.6}}$

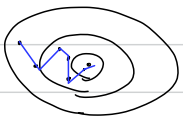
  $= 0.0499..$



- 초기 update 속도가 GD보다 느리지만 global minimum을 지나치지 않고 잘 도착함.

Momentum

weight를 업데이트 해줄때 이전 것들도 포함해 계산 → 아 그래서 부드럽게 가는거구나 Stochastic gradient descent는 그래서 w가 옳은 방향으로 가지만 그 부근에 확률적으로 찍히니까 당연히 지그재그로 가지!



- gradient가 0인 부분이 와도 이전 gradient가 남아있게에 움직이지.

- $V_t = \gamma \cdot V_{t-1} + \eta \cdot \nabla_{\theta_t} J(\theta_t)$
- $\theta_{t+1} = \theta_t - V_t$ ← 이전 w들을 저장

γ = momentum decay rate (≒ 0.9)

$$\theta_{t+1} = \theta_t - \gamma \cdot V_{t-1} - \eta \cdot \nabla_{\theta_t} \cdot J(\theta_t)$$

θ_{t+1}: w′ / $\frac{\partial L}{\partial t}$

ex) $\eta \cdot \nabla_{\theta_t} J(\theta_t) = f_t$

$V_1 = f_1$

$V_2 = \gamma \cdot f_1 + f_2$

$V_3 = \gamma^2 \cdot f_1 + \gamma \cdot f_2 + f_3$

$V_4 = \gamma^3 f_1 + \gamma^2 f_2 + \gamma \cdot f_3 + f_4$

($\gamma = 0.9$ = momentum decay rate)

오래된 gradient의 영향력이 줄어든다.

- gradient가 0인, local minimum에서 벗어나 loss function을 더 탐험할수 있도록 해준다.

Adagrad

- Momentum, SGD와 같은 optimizer는 η이 고정되기에 global minimum을 지나칠 수도 있다.

→ 모든 파라미터에 대해 같은 learning rate(η)를 천천히 낮춘다

→ 이 와중에도 sparse한 node가 있을수도 있기에 개별 맞춤으로 (η)를 낮춘다.

Sparse 하다.
⇒ 특정 공간 안에 굉장히 상대적으로 적은 양의 값만 존재하는 경우
(드문)(희박한)

ex) i번째 input node 값을 $a_1, a_2, a_3, \cdots, a_n$ 이라고 할때
$i+1$ 번째 layer의 입력값은

$$w_0 + w_1 a_1 + w_2 a_2 + \cdots + w_n \cdot a_n \quad (w_i = \theta_i)$$

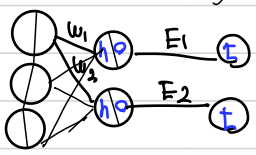
(w_0: bias, w_1: weight, a_1: input)

이때 유독 a_2는 sparse해서 주로 0값이 많이 나올때.
학습할 데이터의 특징이 그 부분만 0값일때

$$w_0 + w_1 a_1 + 0 + \cdots + w_n a_n$$

이기에 w_2 term이 없어서 w_2는 update 되지않음!

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$$



그러다 다음 데이터에서 오랜만에 $a_2 \neq 0$ 인 값이 나와봤자 이미 w_2 제외 w_n들은 global minimum에 도달했지만 w_2는 저~ 처음 위치에서 update 시작임.

$$\frac{dE}{dw_2} = \frac{dE}{do}\frac{do}{dh}\frac{dh}{dw}, \quad (h = w_0 + w_1 a_1 + 0 + w_n a_n)$$

$$\frac{d\,(w_0 + w_1 a_1 + 0 + w_n a_n)}{d\,w_2} = 0$$

따라서 w_2는 다른 파라미터(w_n)를 더 크게 update 해줘야 global minimum에 조금더라도 빨리 들어감!

- t번째 step의 i 번째 파라미터 $\theta_{t,i}$
  (epoch, iterable) (weight) ($w_{t,i}$)
- t 시점의 $\theta_{t,i}$ 에 대한 gradient vector

$$\nabla_{\theta_t} J(\theta_{t,i}) = g_{t,i}$$

$$\frac{\partial\, Loss(w_{ti})}{\partial\, w_{ti}} = g_{t,i}$$ ⇒ Loss 함수의 w_{ti}에 대한 기울기

SGD $\quad \theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}$

Adagrad $\quad \theta_{t+1,i} = \theta_{t,i} - \dfrac{\eta}{\sqrt{G_{t,ii} + \varepsilon}}$

Adagrad vector form $\quad \theta_{t+1} = \theta_t - \dfrac{\eta}{\sqrt{G_t + \varepsilon}} \odot g_t$

(η: learning rate)
(ε: $1e-8$)
($G_{t,ii}$: $g_{1,i}^2 + g_{2,i}^2 + g_{3,i}^2 + \cdots + g_{n,i}^2$)

$\Longleftrightarrow$

$$h = h + \frac{dE}{dw} \odot \frac{dE}{dw}$$ (원소별 곱셈, 기울기의 제곱)

$$w = w - \alpha \frac{1}{\sqrt{h}} \cdot \frac{dE}{dw}$$

매개변수 w 중에 가장 많이 움직인 원소는 학습률을 낮아지게.

RMSProp (Root Mean Square Propagation)

Adagrad는 t가 증가할때마다 $G_{t,ii}$이 커져 learning rate가 소실되는 문제점 (step, epoch, iterate) $\left(\frac{\partial L}{\partial w}, (\nabla_{\theta_t} J(\theta))^2\right)$ (G_t가 분모에 있으니까)

↳ 나거 모든 기울기를 균일하게 더하지 않고 새로운 기울기의 정보만 반영하도록 해서 학습률이 0에 가까워지는 것을 방지

i) ∘ $G_{t,ii}$를 저장하지 않고 지수평균이동.을 저장 (제곱의 합)

ii) ∘ 과거 gradient의 영향력을 감소시키기 위해서 지수이동 평균을 사용.

$$\chi_k = \alpha P_k + (1-\alpha)\chi_{k-1} \quad \text{where } \alpha = \frac{2}{N+1}$$

χ = 이동 평균 값
P = 현재 값 (w의)
α = 영향력 (w에 대한 가중치)
k = step, epoch
N = 값의 개수

ex) 처음부터 지금까지 계산 해보자

$$\begin{aligned}
\chi_k &= \alpha \cdot P_k + (1-\alpha)\chi_{k-1} \\
&= \alpha \cdot P_k + (1-\alpha)(\alpha P_{k-1} + (1-\alpha)\chi_{k-2}) \\
&= \alpha \cdot P_k + \alpha(1-\alpha)P_{k-1} + (1-\alpha)^2\chi_{k-2} \\
&= \alpha \cdot P_k + \alpha(1-\alpha)P_{k-1} + (1-\alpha)^2(\alpha P_{k-2} + (1-\alpha)\chi_{k-3}) \\
&= \alpha \cdot P_k + \alpha(1-\alpha)P_{k-1} + \alpha(1-\alpha)^2 P_{k-2} + (1-\alpha)^3 \cdot \chi_{k-3} \\
&= \cdots \\
&= \alpha\left(P_k + (1-\alpha)P_{k-1} + (1-\alpha)^2 P_{k-2} + (1-\alpha)^3 P_{k-3} + \cdots (1-\alpha)^{N-1} \cdot P_{N-1}\right) + (1-\alpha)^N \cdot \chi_{k-N} \\
&= \alpha \cdot \sum_{t=0}^{N-1} (1-\alpha)^t \cdot P_{k-t} + (1-\alpha)^N \cdot \chi_{k-N}
\end{aligned}$$

이때 $(1-\alpha)$의 주기때문 곱해지고 $(1-\alpha) < 1$ 이기에 시간이 지나면 영향력이 작아진다.
저장된 이전 weight 들이

↳ 필터이론에서는 이런 α(가중치 영향력)을 forgetting factor or decaying factor 라고 불림.

RMS prop

$$G_t = \gamma \cdot G_{t-1} + (1-\gamma)\left(\frac{\partial Cost(w_i)^2}{\partial w_i}\right)\left(\frac{\partial L}{\partial w_{ij}^t}\right)^2$$

$$W_{t+1} = W(t) - \eta \frac{1}{\sqrt{G(t) + \varepsilon}} \cdot \nabla_{\theta_t} J(\theta_{ti})$$

, $\left(\nabla_{\theta_t} J(\theta_{ti}) = \frac{\partial E_{total}}{\partial w_{ti}} = \frac{\partial Loss(w_{ti})}{\partial w_{ti}}\right)$

, ($\gamma = \alpha$ = 영향력 (전과 현 가중치 비율))

∘ 이전 변화량과 현재의 변화량의 어느정도 평균으로 learning rate가 급격히 감소되는 걸 막을수있다
∘ G_t로 인해 최근 변화량에 높은 영향력을 준다 (γ)

ex) $\gamma = 0.9$

$$\begin{aligned}
g_{ij}^{(t)} &= (0.9)\cdot g_{ij}^{(t-1)} + (0.1)\left(\frac{\partial L}{\partial w_{ij}^t}\right)^2 \\
&= (0.9)\left((0.9)\cdot g_{ij}^{(t-2)} + (0.1)\left(\frac{\partial L}{\partial w_{ij}^{t-1}}\right)^2\right) + (0.1)\left(\frac{\partial L}{\partial w_{ij}^t}\right)^2 \\
&= \underbrace{(0.81)\, g_{ij}^{(t-2)}}_{\text{전전 항}} + \underbrace{(0.09)\left(\frac{\partial L}{\partial w_{ij}^{t-1}}\right)^2}_{\text{전 항}} + \underbrace{(0.1)\left(\frac{\partial L}{\partial w_{ij}^t}\right)^2}_{\text{현재}}
\end{aligned}$$