

CH.1 SGD

Def 1.1 SGD

$$\underline{W \leftarrow W - \alpha \frac{\partial L}{\partial W}}$$

Thm 1.2 실행 흐름

optimizer = SGD() # 다른 알고리즘 쓸 때 이것만 업데이트!

for \bar{t} in 10,000

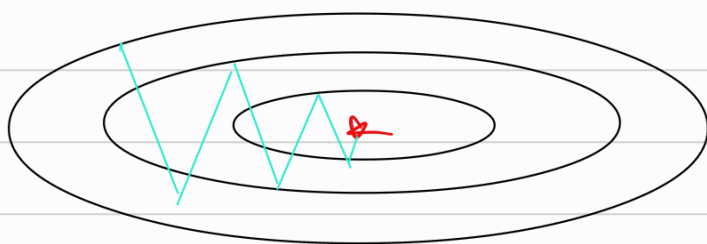
① X_batch, t_batch 추출 & forward

② grad 계산

③ Params 계산

✱ ④ optimizer.update(Params, grad) → for key in Params.keys():
Params[key] -= self.lr * grads[key]

Note 1.3 SGD의 한계.



CH. 2 Momentum

Def 2.1 momentum

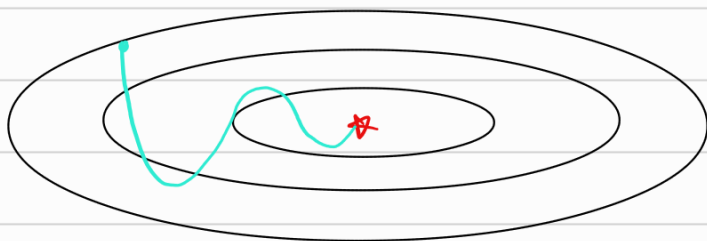
보통 0.9 자연에서 아날 & 공의 저항 등.

$$V \leftarrow \alpha V - \beta \frac{\partial L}{\partial w}$$

$$W \leftarrow W + V$$



Thm 2.2 momentum의 개선?



why?

속도의 의미는 무엇인지
전혀 이해 X.

CH. 3 AdaGrad

Note 3.1 학습률 감소

basic 한 방법은 매번 전체 학습률을 낮추는 것.

여기서 AdaGrad는 이를 타로 뺀 사자고 했다. 각각의 매개 변수에 맞춤형

Def 3.2 Ada Grad

$$(1) \dots \beta \leftarrow \beta + \frac{\partial L}{\partial w} \odot \frac{\partial L}{\partial w} \quad \checkmark \text{ 매개변수마다 학습률을 조정!}$$
$$W \leftarrow W - \alpha \left(\frac{1}{\sqrt{\beta}} \right) \frac{\partial L}{\partial w}$$

Rmk 3.3 (1)에서 계속 과거의 차이가 쌓여 있을 것 (제곱 스케일링)

\Rightarrow 이는 학습을 진행할수록 개선 정도 \downarrow , 실제로 어느 순간부터 개선량 0

이로부터 RMSProp은 과거의 모든 차이

새로운 차이를 더 새로! (EDA라고 함)

CH. 4 Adam (AdaGrad + Momentum)


\star 나의 회상. SGD와 AdaGrad, Momentum의 움직임을 보고 Loss fun의 개형, convex 정도를 파악할 수 있는가?

CH.5 weight init.

Note 5.1 가중치를 전부 0 (혹은 똑같은 값으로 채워진 경우)으로 설정하면 Backpropagation의 원리에 의해 학습 X

Prnk. 5.2 활성화 함수 층의 히스토그램 분포가 극도로 편향된 것이 문제! ↑

Def 5.3 Xavier 초기값



$\frac{1}{\sqrt{n}}$ 을 std로 설정

+ tanh랑 균형이 맞고 한다.
(원점 부근에서 선형성이 필요)

Def 5.4 He 초기값

ReLU랑 잘 맞음.

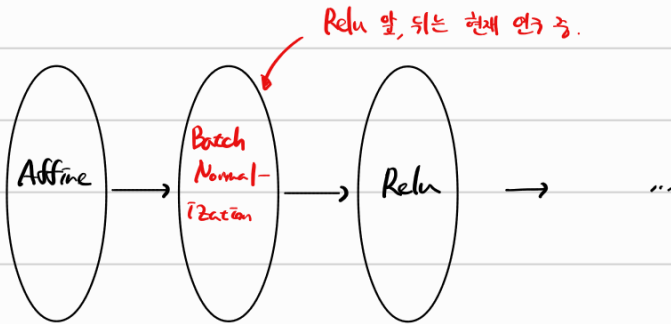
$\frac{2}{\sqrt{n}}$ 을 사용하라!

CH. 6 Batch Normalization

Note 6.1 good

- * ① 학습 속도 개선
- * ② 초깃값 크게 의존 x
- ③ 오버피팅 방지 (드롭아웃 등 필요성 ↓)

Ex 6.2



Def 6.3

미니배치 단위의 정규화 진행

$$\mu_B \leftarrow \frac{1}{n} \sum_{i=1}^n x_i$$

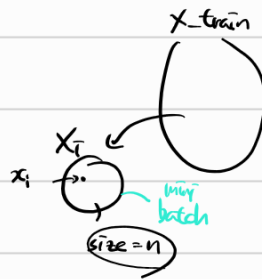
$$\sigma_B^2 \leftarrow \frac{1}{n} \sum_{i=1}^n (x_i - \mu_B)^2$$

$$z_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

exception 방지

$$y_i \leftarrow \underbrace{\sigma}_{\text{scale}} z_i + \underbrace{\beta}_{\text{shift}}$$

초기값 1, 0



CH. 7 Overfitting & Weight decay

Note 7.1 overfitting은 주로 특정 가중치 매개변수와 과하게 커지기 때문에 발생

⇒ Loss function 이 $\frac{1}{2}\lambda W^2$ 이라는 패널티 항을 도입해볼까?

Def 7.2 weight decay

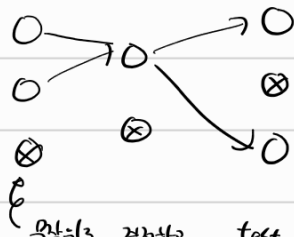
(Penalty 항)

$$\frac{\partial L}{\partial W_{ij}} \leftarrow \frac{\partial L}{\partial W_{ij}} + \lambda W_{ij}$$

Def 7.3 dropout

(양성분의 효과를 가친다고 볼수 있다.)

Weight decay 만으로는 부족, 신경망이 복잡해질 때 "dropout" 을 해보자.



무작위로 제거하고 test 단계에서는 전방 다 살린 후 delete ratio를 곱한다.