딥러닝 Optimization 방법론

└> Neural Net을 학습시키는 training 방법.

① Gradient Descent => 어떤 함수  에 대해 최소값 찾기

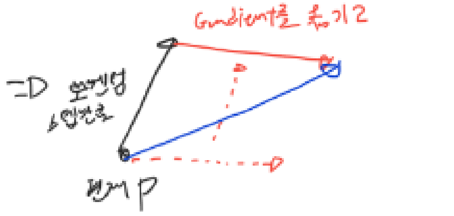
└> parameter의 기울기를 알고 cost를 최소화할 하는 기울기를 찾자

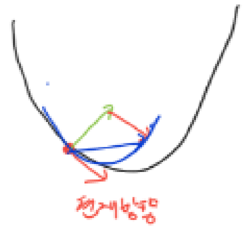
- Batch gradient descent
  - => 모든 data 기울기 구하고 평균 (모두)
- Stochastic " "
  - => 한번에 하나만 기울기 구하기 (1개만)
- mini-batch " "
  - => 중간 갯수정도의 기울기들을 구한다. (2 사이 적절히)
  - ex) 50~256개 정도 $2^{\wedge}$단위로 구한다.

* 적절한 learning rate을 정한다.

기울기 구하는 방법에

* ① Momentum => 이전 momentum + 현재 Gradient를 다음 Gradient에 반영.





• 이전 모멘텀을 현재 gradient에 활용

② Nesterov accelerated gradient

* Adagrad = learning rate 스케줄링인데 => 모든 parameter 마다 learning rate을 바꿔준다.

(Adagrad └> exponential moving average 를 쓰지않음)

para 1000개] ? 바뀌어) => 많이 변한 para는 조금 변하기

1 r 1개 ; 조금 " " 많이 "

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii}+\epsilon}} g_{t,i}$$

└> Gradient의 제곱 => G가 크면 $\theta_{t+1,i}$는 작아짐 (분모니까).

* 문제점 G가 계속 커지면 $\theta_{t+1,i}$가 계속 작아짐

=> 학습 더이상 안 됨.

||

식 해설

Adadelta

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma) \Delta\theta_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\sqrt{E[\Delta\theta^2]_t + \epsilon}}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

ϵ → 입실론 존재 : 초기값시 g가 0이므로 ε를 주어 작은 대용 값을 설정

learning rate 가 없는 버전

↕

* RMSprop = learning rate 가 있는 버전

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

η : global learning rate

Adagrad : Exponential moving average 'η'를 사용하지 않고 Gradient 제곱으로 θ_{t+1}을 조절
⇒ adaptive learning rate은 가능하지만 learning rate이 계속 감소
→ 이를 막기위해 adadelta에서 exponential moving avg는 사용하여 최근 Gradient가 변하는 것을 check!

↕ 차이 ①

RMSprop : Exponential moving average를 사용하여
⇒ global lr을 가짐 ... 'η'

RMSprop과 Adadelta의 차이 ②
⇒ global learning rate이 없다 있다.
'η' 에이라

* Adam ⇒ learning rate과 momentum을 합침

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

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

m_t : 모멘텀
v_t : g의 변화량
v_{t-1} : g의 변화량
β_1 → 모멘텀을 구할 때 얼마나 expo moving avg할지
β_2 → adaptive lr을 구할 때 얼마나 G를 expo mvg할지
보통 0.9 이상값 사용

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} \times \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} m_t$$

m_t → 모멘텀
v_t → g 변화량
$\frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$ → unbias based를 보정하기 위한 식
→ 0에 가까워짐.

ϵ → 입실론 → 초기 g가 0 이 되는 것을 막기위한 값

adam에서 ε의 초기값이 e^{-8} 정도인데 e^{-4} 정도가 학습이 잘됨 (실제로는)
↓
웬만한 경우에는 Adam을 통해 optimize하면 됨

gradient는

모멘텀은 사용 (이전 기울기를 사용)

m t을 사용하기 싫으면 RMSprop => 직전 기울기 사용 X -> global exponential moving avg 사용 => 'g' 베이타

결론 : optimizer에 어떤 방법을 쓰는 것이 좋은가?

=> Adaptive learning rate을 사용하는 것이 좋다.

(Adagrad, Adadelta, RMSprop, Adam)

Adagrad => learning rate이 계속 감소하는 특징 why?) G (기울기 제곱)으로 나누므로

RMSprop => Adagrad의 lr 감소 해결 (Adadelta도 마찬가지)

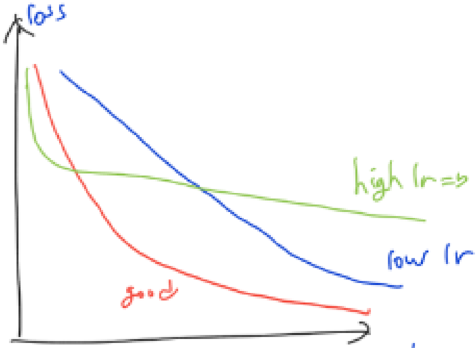
Adam은 bias-correction 또 momentum을 RMSprop에 추가한 것

이전 기울기를 사용 g_t

$\frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$ => bias 보정을 위함

β => 모멘텀 구할 때 사용

* Learning rate



* Restricted Boltzmann Machine (RBM)

① Energy-based models = 대표적 unsupervised model (img만 주고 어떻게 잘 표현할지 학습)
특징을 모음

* Supervised learning 이란 (지도 학습) 입력 -> [f] -> 출력.

$$p(x) = \frac{e^{-E(x)}}{Z}$$

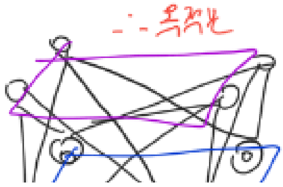
x는 입력, E(x)는 입력이 존재할 확률이고

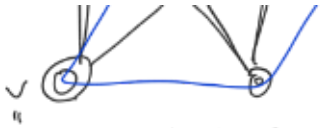
e : energy이고 E(x)는 반비례이므로 '-'

Z : normal parameter

hidden = h

∴ 목적은 e를 학습하는 것.





visible = 우리가 갖고 있는 Image

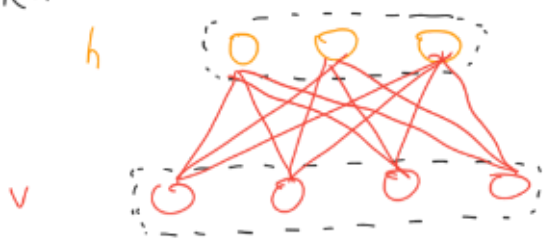
V visible layer

Restricted Boltzman Machine은

= Restricts connection between visible and hidden

즉, RBM은 { v끼리 연결 X / h끼리 연결 X } v와 h만 서로연결

RBM



* Energy (RBM에서는 에너지를 확률식에 씀)

$$E(v,h|\theta) = -\sum_{i=1}^{D}\sum_{j=1}^{F} w_{ij} v_i h_j - \sum_{i=1}^{D} b_i v_i - \sum_{j=1}^{F} a_j h_j$$

w_{ij}, b_i, a_j를 구하는 것.

* Probability (확률)

에너지

$$P(v,h|\theta) = \frac{1}{Z(\theta)} \exp(-E(v,h|\theta))$$

normalization constant

$$Z(\theta) = \sum_{v,h} \exp(-E(v,h|\theta))$$ (모든가능한 v, h (argument)에 대해 에너지를 exp 한다.)

$P(x|y)$ ⇒ y가주어졌을때 x가나올확률

$$= \frac{P(x,y)}{\sum_x P(x,y)}$$

분모 x에대해서 = y의확률 = $P(y)$

* v가 주어졌을 때 v를 최대로하는 parameter 찾기

⇒ w_{ij}, b_i, a_j

$$\hat{\theta} = \operatorname{argmax} \log P(v|\theta)$$

$$\text{FD} \quad \frac{\partial}{\partial\theta}(-\log P(v)) = \frac{\partial}{\partial\theta}\left(-\log \sum_h P(v,h)\right)$$

Sum rule

$$= \frac{\partial}{\partial\theta}\left(-\log \sum_h \frac{1}{Z}\exp(-E(v,h))\right)$$

$f(\theta)$

$\frac{d}{d\theta}\log(f(\theta)) = \frac{1}{f(\theta)}\cdot\frac{d}{d\theta}f(\theta)$

분모의 합수, 분모 θ의 함수으로 θ에대한 합수를 미분

$\frac{\partial}{\partial\theta}(f(\theta)g(\theta)) = f'(\theta)g(\theta) + f(\theta)g'(\theta)$

$$= -\frac{Z}{\left(\sum_h \frac{1}{Z}\exp(-E(v,h))\right)}\left(\sum_h \frac{1}{Z}\frac{\partial}{\partial\theta}\exp(-E(v,h)) - \sum_h \frac{1}{Z^2}\frac{\partial Z}{\partial\theta}\exp(-E(v,h))\right)$$

$\sum_h \exp(-E(v,h))$ ... $f(\theta)g'(\theta)$ ... $f'(\theta)g(\theta)$

$\frac{1}{f(\theta)}$ ↓ 소거

$$= \frac{\partial}{\partial\theta}(-\log p(v)) = \sum_h \left(\frac{\exp(-E(v,h))}{\sum_{h'} \exp(-E(v,h'))} \cdot \frac{\partial}{\partial\theta} E(v,h) \right) + \frac{1}{Z}\frac{\partial Z}{\partial\theta}$$

$* \; Z(\theta) = \sum_{v,h} \exp(-E(v,h))$

$= p(v|h)$: h가 주어졌을 때 v가 나올 확률

$\frac{p(x,y)}{\sum_x p(x,y)}$ = y가 주어졌을 때 x가 나올 확률 $p(x|y)$ 이므로

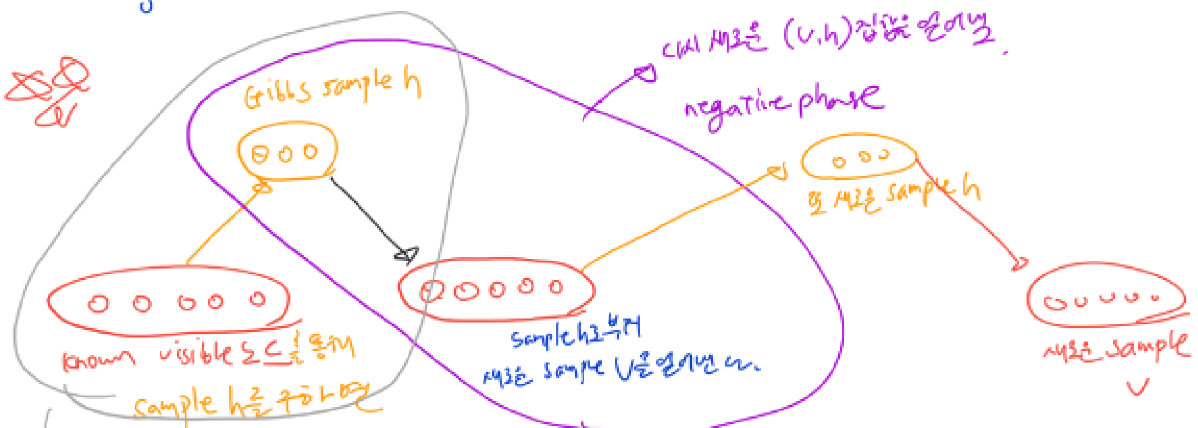
$$= \sum_h p(v|h) \frac{\partial}{\partial\theta} E(v,h) - \frac{1}{Z} \sum_{v,h} \exp(-E(v,h)) \frac{\partial}{\partial\theta} E(v,h)$$

$= p(v,h)$ 모든 v,h에 대한 확률

$$= E\left[\frac{\partial}{\partial\theta} E(v,h) \,\middle|\, v\right] - E\left[\frac{\partial}{\partial\theta} E(v,h)\right]$$

positive phase
||
어떤 v가 주어졌을 때 모든 가능한 hidden의 sum을 구함
Try to lower energy of v

negative phase
h와 v 모두에 대해
Try to increase the energy of all (v,h)



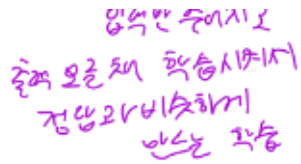
$E\left[\frac{\partial}{\partial\theta} E(v,h) \,\middle|\, v\right]$를 구할 수 있다 (know v)

positive phase

$E\left[\frac{\partial}{\partial\theta} E(v,h)\right]$ 계산 가능 (sample 한 v,h 쌍)

* Deep Belief Network (DBN) = RBM을 쌓는 것. (v,h) layer를 차곡차곡

||
이렇게 해서 비지도 학습 가능



마지막 수정: 오후 5:56