# 10주차

| | |
|---|---|
| 날짜 | @2024년 5월 14일 |
| 과제 | 강의 요약 출석 퀴즈 캐글 필사 |
| 세부내용 | [딥러닝 2단계]<br>4. 최적화 알고리즘 |
| 자료 | [Week10] 출석퀴즈 [Week10] 캐글필사 |

## 최적화 알고리즘

### 1️⃣ 미니 배치 경사하강법

Batch vs Mini Batch Gradient Descent

- 벡터화 : m개의 샘플에 대한 계산을 효율적으로 만들어 줌
- $X = [x^{(1)}, x^{(2)}, \cdots, x^{(m)}]$ → $(n_x, m)$ 차원
- $Y = [y^{(1)}, y^{(2)}, \cdots, y^{(m)}]$ → (1, $m$) 차원
- 한계 : m이 매우 크다면 여전히 경사하강법 속도가 느릴 수 있음

  ⇒ m을 전체 다 훈련시키기 전 경사하강법이 진행되도록 하여 더 빠른 알고리즘 얻기
- 미니배치 : 훈련 세트를 더 작은 훈련 세트들로 나눈 것
  - 첫번째 미니배치 $X^{\{1\}} = [x^{(1)}, \cdots, x^{(1000)}]$
  - 두번째 미니배치 $X^{\{2\}} = [x^{(1001)}, \cdots, x^{(2000)}]$
  - m / mini-batch size 만큼의 미니배치 존재
  - Y에 대해서도 동일하게 적용
- 미니배치 t : ($n_x$,1000)차원의 $X^{\{t\}}$와 (1,1000)차원의 $Y^{\{t\}}$로 구성
- 배치 경사 하강법 : 일반적인 경사 하강법, 모든 훈련 세트를 동시에 진행
- 미니배치 경사 하강법 : 전체를 한 번에 진행시키지 않고 미니배치를 동시에 진행

Mini-Batch Gradient Descent

- for t=1, ..., 5000 ← for문 안에 미니배치를 이용한 1-step gradient 구현

  ① forwardprop on $X^{\{t\}}$

  $Z^{[1]} = W^{[1]}X^{\{t\}} + b^{[1]}$ → 벡터화된 구현 (1000개의 mini-sample에 대해서)

  $A^{[1]} = g^{[1]}(Z^{[1]})$

  $\vdots$

  $A^{[L]} = g^{[L]}(Z^{[L]})$

  ② Compute cost function $J^{\{t\}} = \frac{1}{1000}\sum_{i=1}^{l} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2\cdot 1000}\sum_l ||w^{[l]}||_F^2$

  ③ Backprop to compute gradients wrt $J^{\{t\}}$

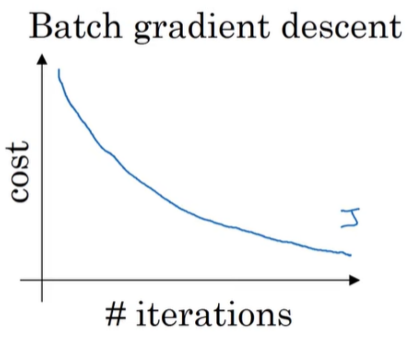
  ④ Update weights : $W^{[l]} := W^{[l]} - \alpha \cdot dW^{[l]}$, $b^{[l]} := b^{[l]} - \alpha \cdot db^{[l]}$

  ⇒ 1 epoch : 훈련 세트를 거치는 한 반복
- 총 5000번의 epoch를 거치게 됨

### 2️⃣ 미니 배치 경사하강법 이해하기

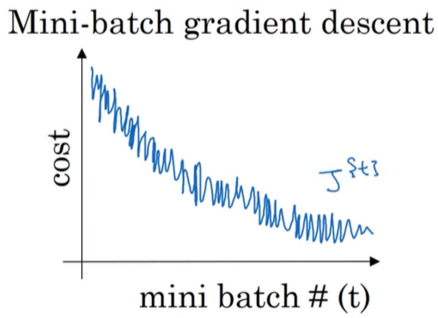
Training with Mini Batch Gradient Descent

- 배치 경사 하강법



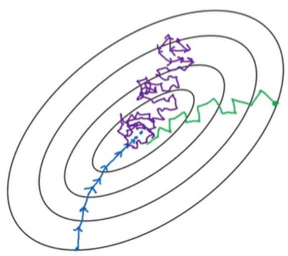

- 모든 반복에서 전체 훈련 세트에 대해 진행
- 각 반복마다 비용이 감소되길 기대함

- 미니배치 경사 하강법




  - 모든 반복에서 다른 훈련 세트(다른 미니배치)에 대해 진행
  - 전체적으로 감소하지만, 많은 노이즈가 발생

Choosing Your Mini Batch Size



- 훈련세트 크기가 m이고, 미니배치 사이즈도 m인 경우 : 배치 경사 하강법
  - $(X^{\{1\}}, Y^{\{1\}})$ = $(X, Y)$
  - 매우 큰 훈련 세트를 모든 반복에 사용 → 한 반복에서 너무 오랜 시간이 걸림
- 훈련세트 크기가 m이고, 미니배치 사이즈는 1인 경우 : 확률적 경사 하강법 (SGD)
  - 각각의 샘플이 하나의 미니배치가 됨
  - $(X^{\{1\}}, Y^{\{1\}})$ =$(X^{(1)}, Y^{(1)})$
  - 벡터화에서 얻을 수 있는 속도 향상을 잃을 수 있음 → 비효율적
- 미니배치 경사 하강법에서 미니배치 크기 : 1과 m 사이 (너무 크거나 작지 않은 것)
  - 미니배치 개수 만큼의 많은 벡터화를 얻을 수 있음
  - 전체 훈련 세트가 진행되기를 기다리지 않고 진행할 수 있음
  - 조금 더 일관되게 전역 최솟값에 도달할 수 있음

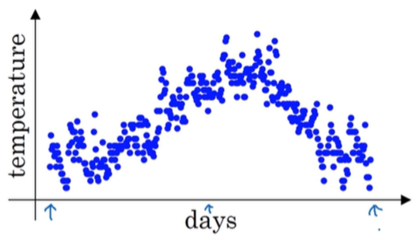
- 미니배치 크기 선택 가이드라인
  - 작은 훈련 세트(m ≤ 2000)라면 배치 경사 하강법 사용
  - 더 큰 훈련 세트라면 일반적으로 64, 128, 256, 512 사용 (2의 제곱)
  - 미니배치에서 모든 $X^{\{t\}}$와 $Y^{\{t\}}$가 CPU와 GPU 메모리에 맞는지 확인

  ⇒ 미니배치 크기는 하이퍼파라미터로, 최적의 값을 찾아내야 함

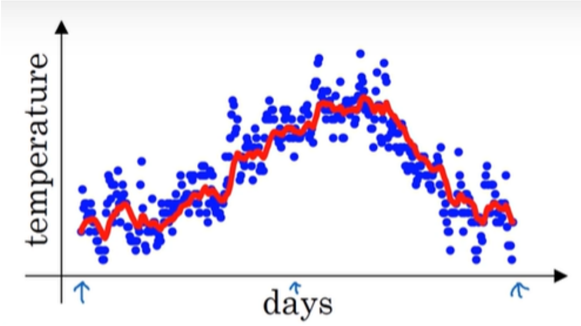
---

### 3️⃣ 지수 가중 이동 평균

Temperature in London (예시)



일별 기온 그래프



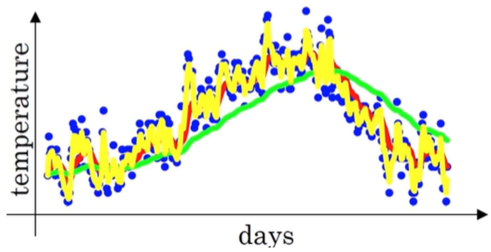
일별 기온의 지수가중평균을 나타낸 그래프

- $\theta_x$ : x번째 날의 온도 → 그래프에 약간의 노이즈가 존재
- $V_0 = 0$ (초기화)

  $V_1 = 0.9V_0 + 0.1\theta_1$

  $V_2 = 0.9V_1 + 0.1\theta_2$
- 일반화 : $V_t = 0.9V_{t-1} + 0.1\theta_t$

Exponentially Weighted (Moving) Averages




- $V_t = \beta V_{t-1} + (1-\beta)\theta_t \approx \frac{1}{1-\beta}$ days' temperature

  → 예) $\beta = 0.9$일 때, $V_t \approx 10$ 일의 기온의 평균
- β=0.98 (1에 가까운 경우), $V_t \approx 50$일의 기온의 평균
  - β값이 클수록 더 많은 데이터의 평균을 이용하기 때문에 선이 더 부드러워짐
  - but, 더 큰 범위에서 데이터를 평균하기 때문에 올바른 값에서 멀어짐
  - 데이터가 바뀔 경우 지수가중평균 공식이 더 느리게 적응

    → 이전 값에 많은 가중치, 현재 값에 작은 가중치를 주기 때문
- β=0.5, $V_t \approx 2$일의 기온의 평균
  - 더 적은 데이터의 평균을 이용하기 때문에 노이즈가 많고 이상치에 민감
  - 데이터 변화에는 더 빠르게 적응

- β값을 변화시키면서 최적의 값(일반적으로 β=0.9)을 찾아야 함

## 4️⃣ 지수 가중 이동 평균 이해하기

Exponentially Weighted Averages

- 공식 : $v_t = \beta v_{t-1} + (1-\beta)\theta_t$ → β = 0.9
- $v_{100} = 0.9v_{99} + 0.1\theta_{100}$

  $v_{99} = 0.9v_{98} + 0.1\theta_{99}$

  $v_{98} = 0.9v_{97} + 0.1\theta_{98}$

  $\ldots$
- $v_{100} = 0.1\theta_{100} + 0.9v_{99}$
  $= 0.1\theta_{100} + 0.9(0.1\theta_{99} + 0.9v_{98})$
  $= 0.1\theta_{100} + 0.9(0.1\theta_{99} + 0.9(0.1\theta_{98} + 0.9v_{97}))$
  $= 0.1\theta_{100} + 0.1 \cdot 0.9\theta_{99} + 0.1 \cdot (0.9)^2\theta_{98} + 0.1 \cdot (0.9)^3\theta_{97} + \cdots$

  ⇒ $\theta_{100}$의 가중치의 평균
- 그림으로 표현하면 지수적으로 감소 0.1 → 0.1*0.9 → ...
- $(1-\varepsilon)^{(1/\varepsilon)} \approx \frac{1}{e} \approx 0.35$ (ε = 1-β)
  - β=0.9 :10일 뒤의 가중치는 현재 날짜의 가중치의 1/3로 줄어듦
  - β=0.98 : 50일이 지나야 현재 날짜의 가중치의 1/3로 줄어듦

Implementing Exponentially Weighted Averages

$$v_0 = 0$$
$$v_1 = \beta v_0 + (1-\beta)\,\theta_1$$
$$v_2 = \beta v_1 + (1-\beta)\,\theta_2$$
$$v_3 = \beta v_2 + (1-\beta)\,\theta_3$$

...

- $v_\theta$=0  # 초기화

  $v_\theta$ := β*$v_\theta$ + (1-β)$\theta_1$

  $v_\theta$ := β*$v_\theta$ + (1-β)$\theta_2$

  $\vdots$
- 반복문으로 구현

  $v_\theta$=0  # 초기화

  repeat {

  get next $\theta_t$

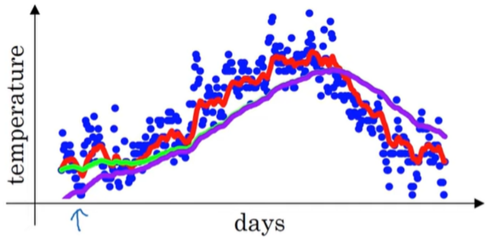
  $v_\theta$ := β*$v_\theta$ + (1-β)$\theta_t$

  }
- 지수평균 장점 : 메모리 효율적

## 5️⃣ 지수 가중 이동 평균의 편향보정

Bias Correction

- β=0.98일 때 → 보라색 라인을 얻게 됨 (약간의 오차)




- $v_t = \beta v_{t-1} + (1-\beta)\theta_t$

  $v_0 = 0$

  $v_1 = 0.98 v_0 + 0.02\theta_1 = 0.02\theta_1$ → 값이 훨씬 낮아져서 첫 번째 날의 온도를 잘 추정할 수 없음

  $v_2 = 0.98 v_1 + 0.02\theta_2 = 0.98 * 0.02 * \theta_1 + 0.02\theta_2 = 0.0196\theta_1 + 0.02\theta_2$

  → $v_2$ 역시 $\theta_1$과 $\theta_2$ 보다 훨씬 작은 값이 되어서 온도를 잘 추정할 수 없음

  ⇒ 추정의 초기 단계에서 정확도가 떨어짐

- 편향 보정

$$\frac{v_t}{1-\beta^t}$$

  - t=2 : $1-\beta^t = 1-(0.98)^2 = 0.0396$ → 둘째날 추정값은 $\frac{v_t}{0.0396} = \frac{0.0196\theta_1 + 0.02\theta_2}{0.0396}$

    → $\theta_1$과 $\theta_2$의 가중평균에서 편향을 없앤 값
  - t가 충분히 크면 $\beta^t$는 0에 가까워지고 따라서 편향 보정의 효과는 거의 없어짐

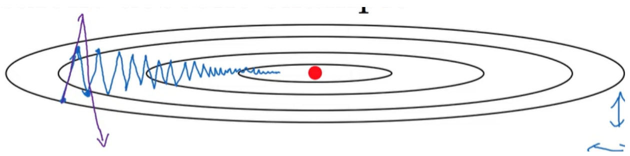
    ⇒ but, 초기 단계에서 편향 보정이 더 나은 추정값을 얻도록 도와줌

---

### 6️⃣ Momentum 최적화 알고리즘

Gradient Descent Example

- 경사하강법 예제



  - 진동은 경사 하강법의 속도를 느리게 하고 큰 학습률을 사용하지 못하게 함
  - 학습률이 너무 크면 오버슈팅 발생 → 발산
  - 수직축 : 진동을 막기 위해 학습이 더 느리게 일어나기를 바람
  - 수평축 : 최솟값을 향해 더 빠른 학습을 원함
- Momentum

  $v_{dw} = 0,\ v_{db} = 0$ → $v_{dw}$는 dw, w와 같은 차원, $v_{db}$는 db, b와 같은 차원

  On iteration t:

    Compute dw, db on current mini-batch

    $v_{dw} = \beta v_{dw} + (1-\beta)dw$ → 이동평균을 w에 대한 도함수로 계산

    $v_{db} = \beta v_{db} + (1-\beta)db$

    $w = w - \alpha v_{dw},\ b = b - \alpha v_{db}$ → 경사하강법의 단계를 부드럽게 만들어 줌

  - 수직 방향에서는 평균이 0 : 훨씬 작은 진동
  - 수평 방향에서는 평균이 큼 : 빠른 학습

  ⇒ dw, db는 최솟값을 향해 내려갈 때 가속을 제공

  ⇒ 모멘텀 항 v는 속도를 나타냄

⇒ β는 1보다 조금 작기 때문에 마찰을 제공해서 제한 없이 빨라지는 것을 방지

Implementation Details

- 두 개의 하이퍼파라미터 존재
  - α : learning rate
  - β (=0.9) : 지수가중평균 제어
- 편향 보정은 거의 하지 않음
  → 10번의 반복 뒤에 이동 평균이 충분히 진행돼서 편향 추정이 더이상 일어나지 않기 때문
- $(1-\beta)$ 항이 삭제되어 있는 경우 존재
  - $v_{dw} = \beta v_{dw} + dw$ → $v_{dw}$가 1/(1-β)에 대한 계수로 스케일링 되는 것
  - α도 1/(1-β)에 대응되는 값으로 바뀌어야 함

---

### 7️⃣ RMSProp 최적화 알고리즘

RMSprop

- 경사하강법의 진행 방향에서 수평축을 w, 수직축을 b라고 가정
  - goal : b 방향의 속도는 느리게, w 방향의 속도는 빠르게
- RMSprop 알고리즘

  On iteration t:

  Compute dw, db on current mini-batch

  $S_{dw} = \beta_2 S_{dw} + (1-\beta_2)dw^2$ (elementwise 제곱 연산) → 도함수의 제곱을 지수가중평균

  $S_{db} = \beta_2 S_{db} + (1-\beta_2)db^2$

  $w := w - \alpha\frac{dw}{\sqrt{S_{dw}}}$, $b := b - \alpha\frac{db}{\sqrt{S_{db}}}$

  - dw$dwS_{dw}$ 값이 작음 → $w$가 커짐 → 수평 방향 속도가 빨라짐
  - db가 상대적으로 큼 → $S_{db}$ 값이큼 → $b$가 작아짐 → 수직 방향 속도가 느려짐
  - 실제로 db가 dw에 비해 큼 → 수직 방향 b에서 기울기가 더 가파르기 때문
- 큰 학습률을 사용해 빠르게 학습 가능하면서도 수직 방향으로 발산하지 않음
- $S_{dw}$가 0이 되지 않도록 안정성을 보장해야 함 → $\varepsilon \geq 10^{-8}$

---

### 8️⃣ Adam 최적화 알고리즘

Adam Optimization Algorithm

- $V_{dw} = 0, S_{dw} = 0, V_{db} = 0, S_{db} = 0$

  On iteration t:

  Compute dw, db using current mini-batch

  $V_{dw} = \beta_1 V_{dw} + (1-\beta_1)dw$, $V_{db} = \beta_1 V_{db} + (1-\beta_1)db$ → momentum

  $S_{dw} = \beta_2 S_{dw} + (1-\beta_2)dw^2$, $S_{db} = \beta_2 S_{db} + (1-\beta_2)db^2$ → RMSprop

  $V_{dw}^{\text{corredcted}} = V_{dw}/(1-\beta_1^t)$, $V_{db}^{\text{corredcted}} = V_{db}/(1-\beta_1^t)$ → momentum에 대한 편향 보정

  $S_{dw}^{\text{corredcted}} = S_{dw}/(1-\beta_2^t)$, $S_{db}^{\text{corredcted}} = S_{db}/(1-\beta_2^t)$ → RMSprop에 대한 편향 보정

  $w := w - \alpha\frac{V_{dw}^{\text{corredcted}}}{\sqrt{S_{dw}^{\text{corredcted}}}+\varepsilon}$, $b := b - \alpha\frac{V_{db}^{\text{corredcted}}}{\sqrt{S_{db}^{\text{corredcted}}}+\varepsilon}$

  ⇒ momentum + RMSprop

Hyperparameters Choice

- α(learning rate) : 중요, 다양한 값을 시도해서 잘 맞는 것을 찾아야 함

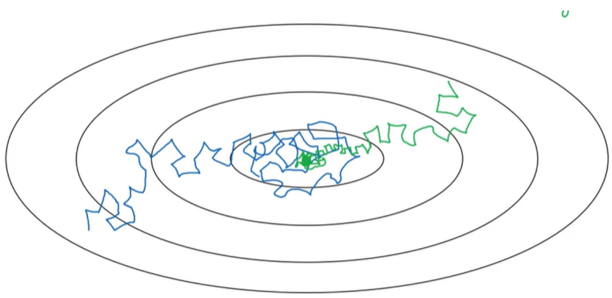
- $\beta_1$(dw의 이동가중평균, 모멘텀) : 일반적으로 0.9 선택
- $\beta_2$($dw^2$의 이동가중평균, RMSprop) : Adam 논문에서 0.999 선택
- $\varepsilon$ : 보정할 필요 없음, Adam 논문에서 $10^{-8}$ 추천
- Adam : Adaptive moment estimation
    - $\beta_1$ : 첫번째 모멘트, 도함수의 평균 계산
    - $\beta_2$ : 두번째 모멘트, 지수가중평균의 제곱을 계산

---

### 9️⃣ 학습률 감쇠

Learning Rate Decay

- 학습률 감쇠가 필요한 이유



    - 상당히 작은 미니배치를 가정 (ex. 64, 128)
    - 기본 : 약간의 노이즈가 있지만 최솟값으로 향하는 경향, 정확하게 수렴하지 않음
    - 학습률 천천히 줄이면 초기 단계에서는 빠른 학습이 가능하나,
      학습률이 작아질수록 작은 스텝으로 최솟값 주변 밀집된 영역에서 진동
- 학습률 감쇠 구현 방법

$$\alpha = \frac{1}{1 + \text{decay.rate} * \text{epoch.num}} \cdot \alpha_0$$

    - 1 epoch = 1 pass through data
    - decay.rate : 감쇠율, 조정 필요한 하이퍼파라미터
    - $\alpha_0 = 0.2, \text{decay.rate} = 1$

| Epoch | α |
| --- | --- |
| 1 | 0.1 |
| 2 | 0.067 |
| 3 | 0.05 |
| 4 | 0.04 |

⇒ $\alpha_0$와 감쇠율을 잘 조정하면서 적절한 값을 찾기

Other Learning Rate Decay Methods

- 지수적 감쇠 : $\alpha = 0.95^{\text{epoch.num}} \cdot \alpha_0$

  → 기하급수적으로 빠르게 학습률 감소
- $\alpha = \frac{k}{\sqrt{\text{epoch.num}}} \cdot \alpha_0$ or $\alpha = \frac{k}{\sqrt{t}} \cdot \alpha_0$
- 이산 계단 방법 : step마다 학습률을 반 씩 줄임