손실함수와 경사하강법

수퍼스타를 키우는 곳
전주대학교

경사하강법 최적화알고리즘

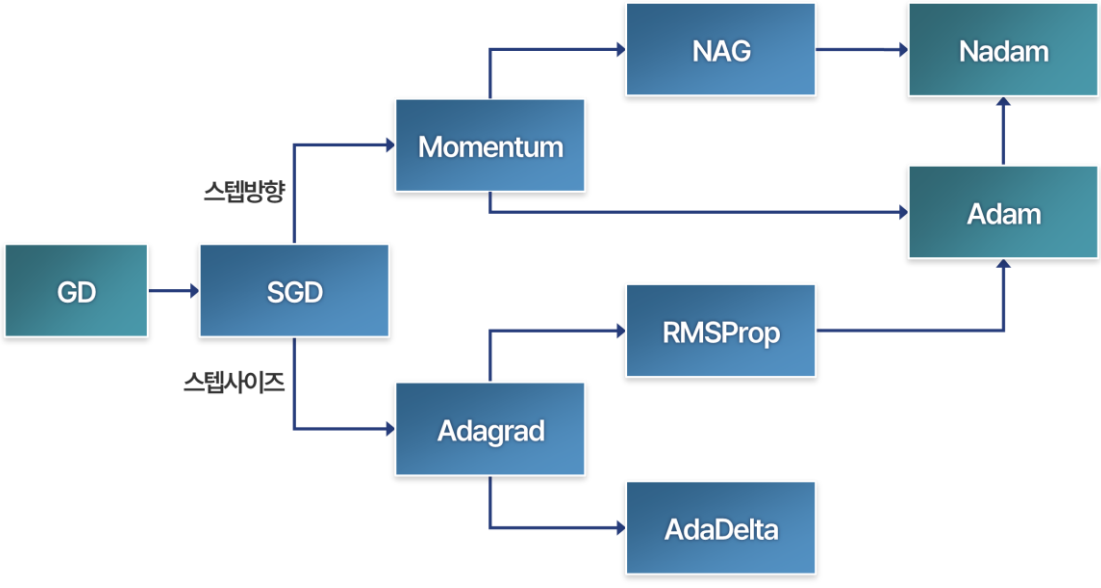
경사하강법 최적화알고리즘 1

경사하강법 최적화알고리즘 1

❖ 경사하강법

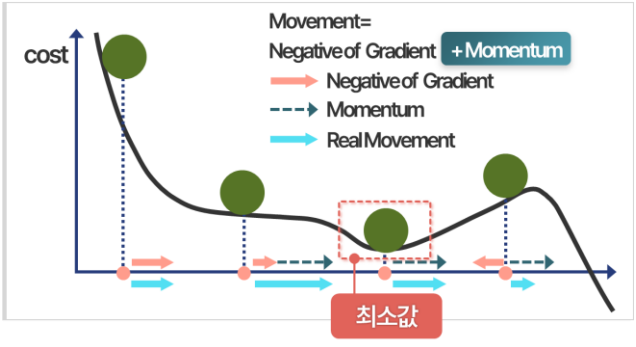
- 모멘텀(Momentum)
- NAG(Nesterov Accelrated Gradient)
- AdaGrad(Adaptive Gradient)
- AdaDeltaAdaptive Delta, 아다델타)
- RMSProp(Root Mean Sqaure Propagation)
- Adaptive Moment Estimation(Adam)

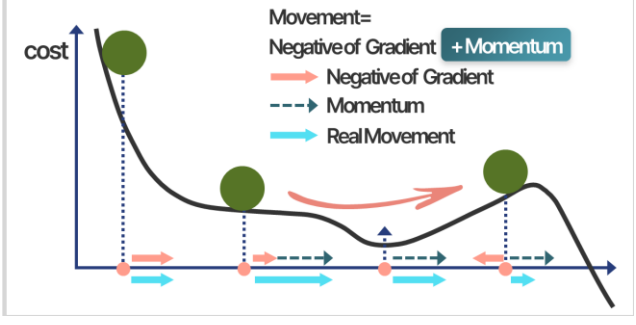
❖ 좋은 방법을 강구한 알고리즘



경사하강법 최적화알고리즘 1

❖ 모멘텀(Momentum)

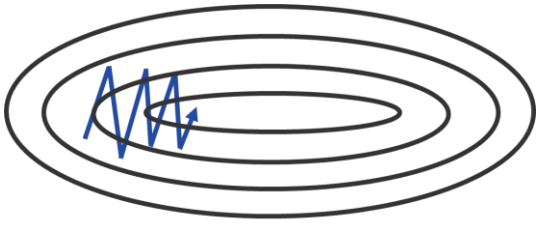




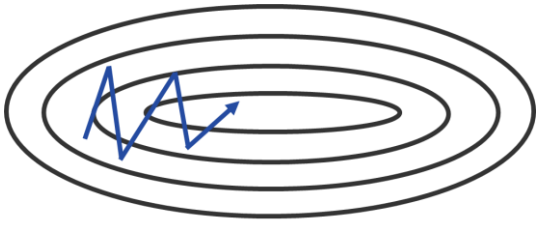
- 외부에서 힘을 받지 않는 한 정지해 있거나 운동 상태를 지속하려는 성질
  - 기울기에 관성을 부과, 작은 기울기는 쉽게 넘어가도록
- 이전에 이동했던 방향을 기억하면서 이전 기울기의 크기를 고려하여 어느 정도 추가로 이동
- 경사하강법으로 손실 함수를 줄여나가다가 Local Minimum에 빠질 수 있는 상황에서 Momentum 기법을 사용하면 이전 기울기의 크기를 고려해 추가로 이동하기 때문에 이를 빠져나갈 수 있음
- Global Minimum이라고 표시한 지점에 도달했을 때는 추가적인 관성을 받아도 더 올라갈 수 없기 때문에 이 지점이 Global Minimum이 됨
- Momentum은 Local Minimum에 빠지는 경우를 대처할 수 있음

경사하강법 최적화알고리즘 1

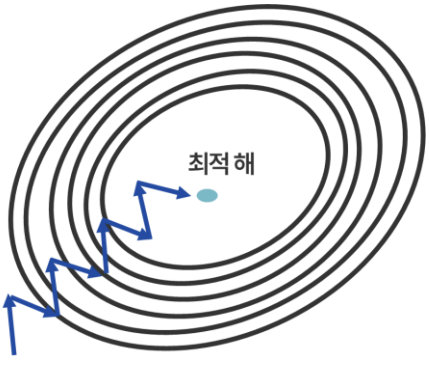
❖ 모멘텀(Momentum)



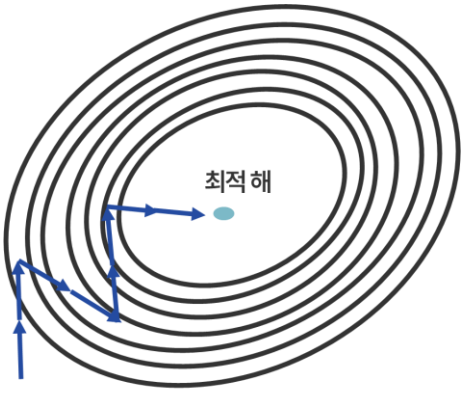
SGD without Momentum



SGD with Momentum



확률적 경사 하강법

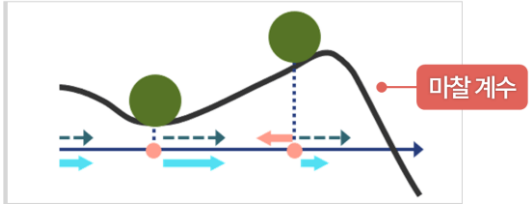


모멘텀

경사하강법 최적화알고리즘 1

❖ 모멘텀(Momentum)

- 이전 방향이 어디였는지가 중요



- 너무 빠르게 내려가지 않도록 일종의 마찰저항을 주는 관성계수 추가
- 모멘텀 항은 같은 방향 증가, 다른 방향 감소

$$v_t = \gamma \cdot v_{t-1} + \eta \cdot \nabla_\theta J(\theta)$$
$$\theta = \theta - v_t$$

v_t = t 번째 time step 에서의 이동벡터

γ = 관성계수(0.9)

```
optimizer = keras.optimizers.SGD
            (lr=0.001, momentum=0.9)
```

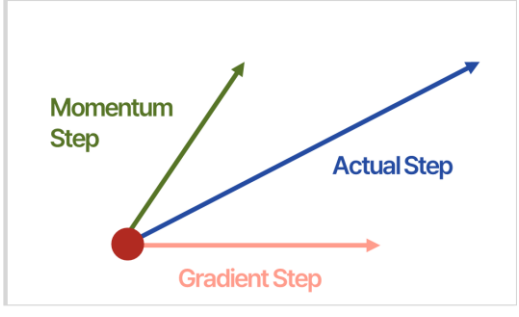
경사하강법 최적화알고리즘

경사하강법 최적화알고리즘 2

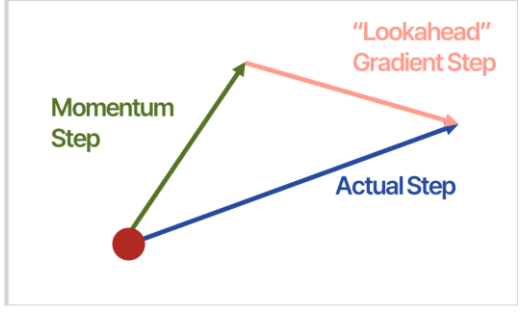
경사하강법 최적화알고리즘2

❖ NAG(Nesterov Accelerated Gradient)

- 네스테로프 모멘텀
- 맹목적으로 경사면을 따라 언덕 아래로 굴러가는 공은 매우 만족스럽지 않음
  - 언덕을 다시 오르기 전에 속도를 늦출 줄 아는 공을 갖고 싶음
- 모멘텀 항에 이런 종류의 선견지명을 부여하는 방법
  - 모멘텀 항을 사용하여 매개 변수 θ를 이동시킬 것임을 알고 있음
  - 따라서 $\theta - \gamma \cdot v_{t-1}$ 을 계산하면 매개 변수의 다음 위치에 대한 근사치를 얻게 됨
  - 현재 매개 변수 θ 대신 매개 변수의 근사 미래 위치에 대한 기울기를 계산함으로써 매개 변수의 이동 방향을 파악할 수 있게 됨



Momentum Update

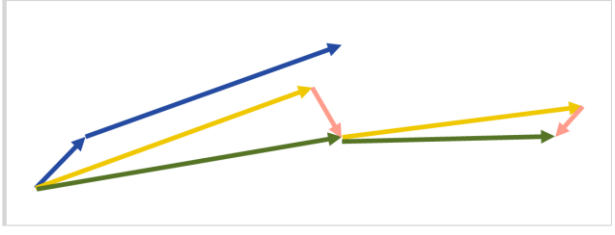


Nesterov Momentum Update

경사하강법 최적화알고리즘2

❖ NAG(Nesterov Accelerated Gradient)

- 방향을 계산하기 전 모멘텀 방법으로 인해 이동될 방향을 미리 예측하고 해당 방향으로 간 다음 Gradient를 계산
  - 즉, 한 단계를 미리 예측함으로써 불필요한 이동을 줄임



Nesterov Update

$$v_t = \gamma \cdot v_{t-1} + \eta \cdot \nabla_\theta J(\theta - \gamma \cdot v_{t-1})$$
$$\theta = \theta - v_t$$

이 수식만 달라짐

v_t = t 번째 time step 에서의 이동벡터

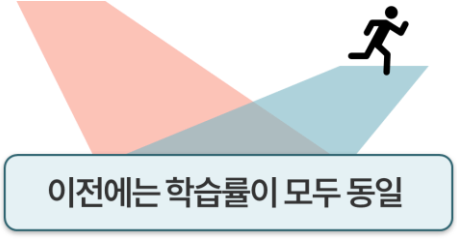
γ = 관성계수(0.9)

optimizer = keras.optimizers.SGD
(lr=0.001, momentum=0.9, nesterov=True)

경사하강법 최적화알고리즘2

❖ AdaGrad(Adaptive Gradient)

- Feature별로 학습률(Learning Rate)을 다르게 조절



$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

$$g_{t,i} = \nabla_{\theta_t} J(\theta_{t,i})$$
$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}$$
$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{(G_{t,ii} + \epsilon)}} \cdot g_{t,i}$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(G_t + \epsilon)}} \odot g_t$$

g_t = 대각 행렬이며 각 대각 성분 i, i는 시간 단계 t까지의 θ_i에 대한 기울기 제곱의 합
ϵ = 분모가 0이 되는 것을 방지하기 위한 작은 값 (10^{-8})
keras.optimizers.Adagrad(lr=0.01, epsilon=1e-8)

- 큰 기울기를 가져 학습이 많이 된 변수는 학습률을 감소
- 학습이 적게 된 변수는 잘 학습되도록 학습률을 높게 설정

경사하강법 최적화알고리즘2

❖ AdaGrad(Adaptive Gradient)

- 기울기 누적크기
  - 기울기 누적크기 값(g_t)은 점차 커지기 때문에 학습이 오래 진행되면 학습률($\frac{\eta}{\sqrt{(G_t+\epsilon)}}$)이 0에 가까워지기 때문에 더 이상 학습이 진행되지 않을 수 있음
- Adagrad의 문제(학습률이 0으로 수렴) 해결
  - 전체 학습률의 항이 0으로 감
- 모든 과거 기울기를 누적하는 대신 웨이트를 다르게 주는 형태
  - 누적돼서 0이 되는 걸 막음

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(G_t + \epsilon)}} \odot g_t$$

$$\Delta\theta_t = -\eta \cdot g_{t,i}$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{(E[g^2]_t + \epsilon)}} \cdot g_t$$

$$\Delta\theta_t = -\frac{\eta}{RMS[g]_t} \cdot g_t$$

경사하강법 최적화알고리즘2

❖ RMSProp(Root Mean Square Propagation)

- AdaGrad는 학습이 진행될 때 학습률이 꾸준히 감소하다 나중에는 0에 수렴
  - 변수별로 학습률을 조절하는 건 동일하지만 기울기 업데이트 방식에 차이
- 지수이동평균(Exponential Moving Average)
  - 멀리 있는 데이터보다는 가까이에 있는 데이터에 웨이트를 주어 구함

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

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

```
tf.train.RMSPropOptimizer(learning_rate=0.01,decay=0.9,
momentum=0.0,epsilon=1e-10)
```

경사하강법 최적화알고리즘2

❖ Adam(Adaptive Moment Estimation)

- Momentum과 RMSProp의 장점을 결합한 알고리즘
  - Momentum
    - 방향
  - RMSProp
    - 스텝

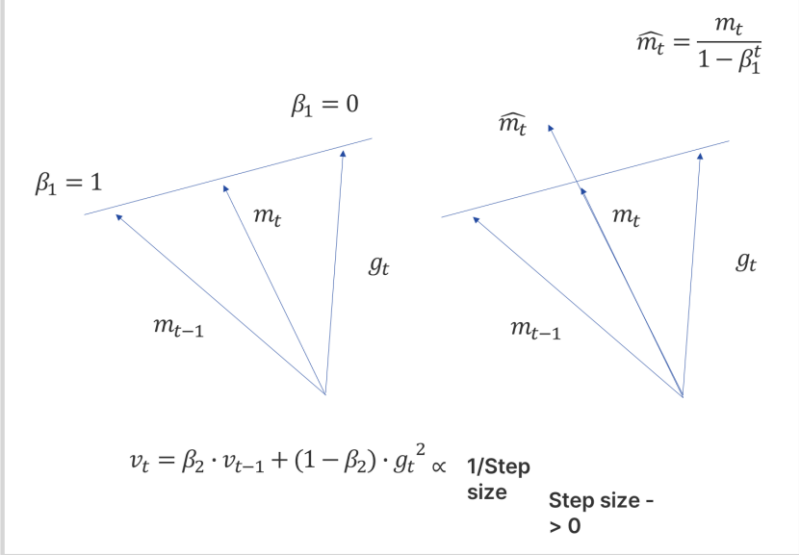
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(\hat{v}_t + \epsilon)}} \cdot \hat{m}_t$$

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

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

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

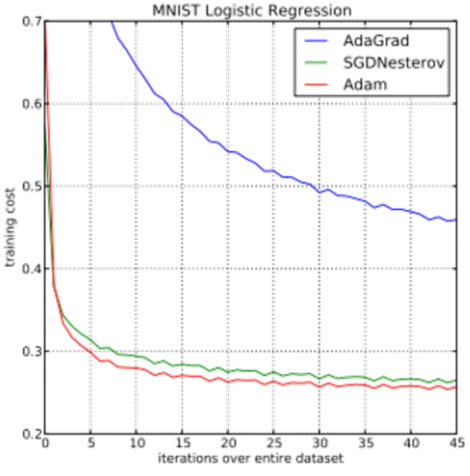
β_1 = momentum 의 지수이동평균(0.9)
β_2 = RMSProp 의 지수이동평균(0.999)
v_t = t 번째 time step 까지의 기울기 누적 크기
ϵ = 분모가 0이 되는 것을 방지하기 위한 작은 값 (10^{-8})
η = 학습률(0.001)

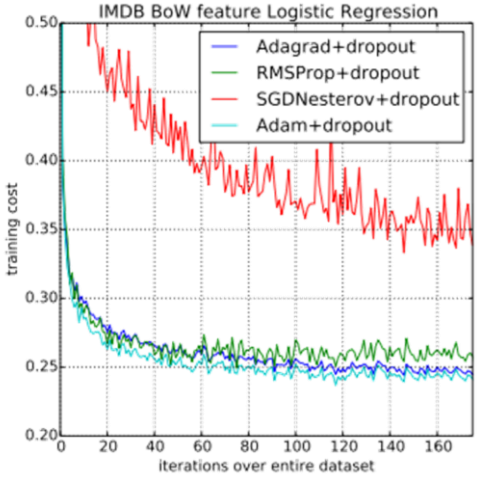


경사하강법 최적화알고리즘2

❖ Adam(Adaptive Moment Estimation)

- 편향 보정(bias correction)
  - M(0)와 V(0)값이 0으로 초기화 되는데 시작값이 0이기 때문에 이동평균을 구하면 0으로 편향된 값추정이 발생할 수 있음
    - 특히 초기 감쇠 속도가 작은 경우(즉, β가 1에 가까울 때)에 발생
    - 이를 방지하기 위해 $1-\beta^t$ 값을 나누어 편향 보정
- 방향과 보폭을 동시에 써서 결과가 좋음





경사하강법 최적화알고리즘2

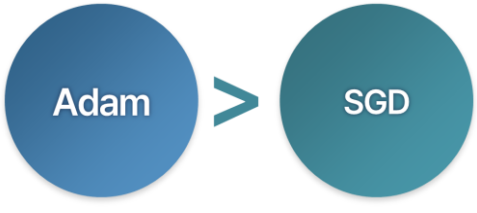
❖ Nadam

- Nesterov-accelerated Adaptive Moment Estimation
- NAG와 Adam을 결합한 알고리즘

$$
\begin{aligned}
m_t &= \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \\
\widehat{m_t} &= \frac{m_t}{1-\beta_1^t} \\
\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{(\widehat{v_t}+\epsilon)}} \cdot \widehat{m_t} \\
&= \theta_t - \frac{\eta}{\sqrt{(\widehat{v_t}+\epsilon)}} \cdot \left(\frac{\beta_1 \cdot m_{t-1}}{1-\beta_1^t} + \frac{(1-\beta_1)\cdot g_t}{1-\beta_1^t}\right) \\
&= \theta_t - \frac{\eta}{\sqrt{(\widehat{v_t}+\epsilon)}} \cdot \left(\beta_1 \widehat{m_t} + \frac{(1-\beta_1)\cdot g_t}{1-\beta_1^t}\right)
\end{aligned}
$$

❖ AdamW

- Decoupled Weight Decay Regularization



- weight decay의 문제
- Regularization과 weight decay를 분리하여 이 문제를 해결

경사하강법 최적화알고리즘2

❖ AdamW

$$\text{Regularized Loss} = f(x) + \frac{\lambda}{2}\sum_i x_i^2$$
$$= f(x) + \frac{\lambda'}{2}\sum_i \|x\|_2^2 , \qquad \lambda' = \frac{\lambda}{\eta}$$

$$x_t = x_{t-1} - \eta\nabla f(x_{t-1}) = x_{t-1} - \eta\left(\frac{\partial f(x)}{\partial x_{t-1}} + \frac{\frac{\lambda'}{2}\sum_i\|x\|_2^2}{\partial x_{t-1}}\right)$$
$$= x_{t-1} - \eta\nabla f(x_{t-1}) - \eta\lambda' x_{t-1} = x_{t-1} - \eta\nabla f(x_{t-1}) - \lambda x_{t-1}$$
$$= (1-\lambda)x_{t-1} - \eta\nabla f(x_{t-1})$$

$$x_t = x_{t-1} - \eta M_t\left(\frac{\partial f(x)}{\partial x_{t-1}} + \frac{\frac{\lambda'}{2}\sum_i\|x\|_2^2}{\partial x_{t-1}}\right)$$
$$= x_{t-1} - \eta M_t(g_t - \lambda' x_{t-1}) = x_{t-1} - \eta M_t g_t - \eta M_t\lambda' x_{t-1}$$
$$= (1 - M_t\lambda)x_{t-1} - \eta M_t\, g_t$$

$g_t = g_{t-1} + \lambda x_{t-1}$ Regularization

보폭 $g_t = \beta_2 \cdot v_{t-1} + (1-\beta_2)\cdot(\nabla f(x_{t-1}))^2$

방향 $m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1)\cdot\nabla f(x_{t-1})$

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t} \qquad \widehat{g_t} = \frac{g_t}{1-\beta_2^t}$$

$$x_t = x_{t-1} - \frac{\eta}{\sqrt{(\widehat{g_t} + \epsilon)}}\cdot\widehat{m_t} - \lambda x_{t-1}$$

경사하강법 최적화알고리즘2

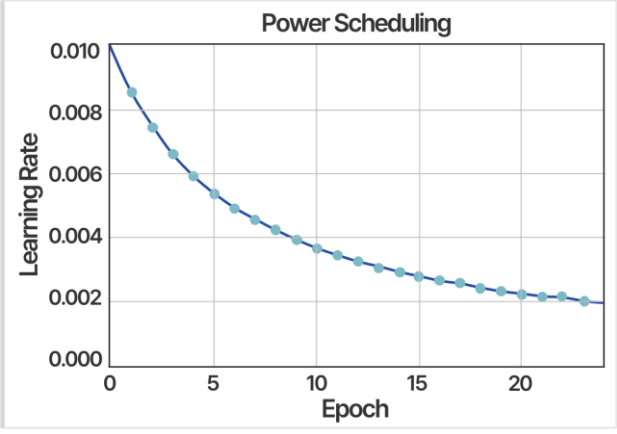
❖ Radam(Rectified Adam)

- Adaptive Learning Rate Method에서 학습 초기에 샘플이 부족하여Adaptive Learning Rate의 분산이 커짐
  - 이로 인해 초기에 Local Minima에 빠지게 될 수 있고 너무 일찍 도달하여 학습이 거의 일어나지 않을 수 있음
- 이를 'Bad Local Optima Convergence Problem'이라 하며, 이 문제를 해결하기 위해 RAdam 개발
  - Adaptive Learning Rate의 분산을 구하고 이를 일정하게 만들어 주기 위한 Rectification Term을 제안

경사하강법 최적화알고리즘2

❖ 학습률

- 거듭제곱 기반 스케줄링(Power Scheduling)
  - 학습이 진행될수록 학습률은 감소

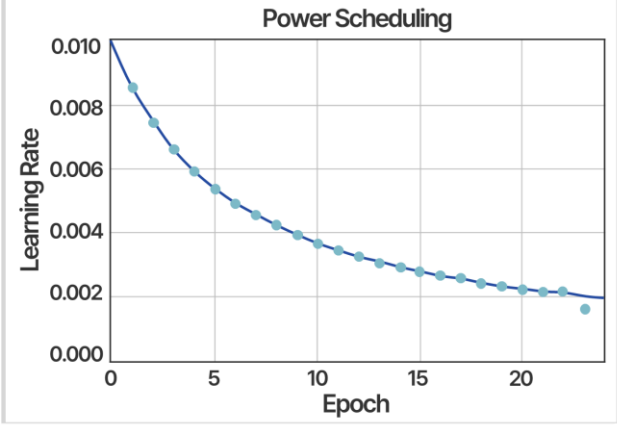


$$\eta_t = \frac{\eta_0}{(1+\frac{t}{s})^c}$$

t = training epoch(반복횟수)
s = 스텝횟수
η_0 = 초기 학습률
c = 상수
decay는 스텝 수 s의 역수(keras는 이를 1로 가정)

```
optimizer = keras.optimizers.SGD(lr=0.01, decay=1e-4)
```

- 지수 기반 스케줄링(Exponential Scheduling)
  - 학습이 진행될수록 학습률은 감소



$$\eta_t = \eta_0\ 0.1^{\frac{t}{s}}$$

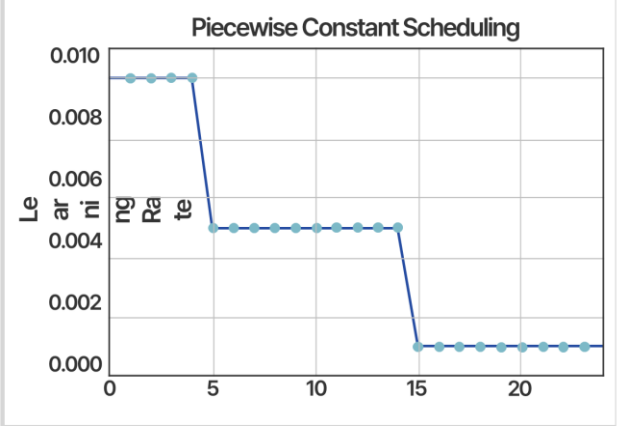
t = training epoch(반복횟수)
s = 스텝횟수
η_0 = 초기 학습률

```
def exponential_decay(lr0, s):
    def exponential_decay_fn(epoch):
    return lr0 * 0.1 **(epoch / s)
 return exponential_decay_fn
exponential_decay_fn = exponential_decay(lr0=0.01, s=20)
```

경사하강법 최적화알고리즘2

❖ 학습률

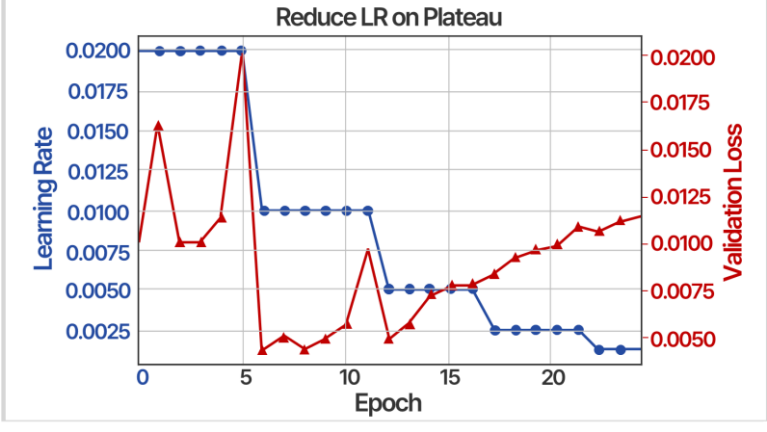
- 구간별 고정 스케줄링(Piecewise Constant Scheduling)
  - 처음에는 학습률을 높게 하다 점점 줄어들게 함




```
def piecewise_constant_fn(epoch):
  if epoch < 5:
    return 0.01
  elif epoch < 15:
    return 0.005
  else: return 0.001

lr_scheduler =
keras.callbacks.LearningRateScheduler(piecewise_constant_fn)
```

- 성능 기반 스케줄링(Perfomance Scheduling)
  - 매 N번의 스텝마다 Validation Loss를 측정하고, 이것이 줄어들지 않으면 l 배만큼 학습률을 감소



경사하강법 최적화알고리즘2

❖ 학습률

- 1사이클 스케줄링(1cycle Scheduling)
  - 훈련 절반 동안 초기에 지정한 학습률 η_0 을 선형적으로 η_1 까지 증가, 그 다음 나머지 절반 동안은 증가되었던 η_1 을 η_0 으로 선형적으로 감소
  - 모멘텀 사용시, 처음에는 높은 모멘텀으로 시작하여 학습률을 제어시켰다가 최대 학습률에 되는 절반 지점까지 점점 모멘텀을 줄임. 그리고 나머지 절반동안 학습률이 다시 초기 학습률로 돌아가는 동안 점진적으로 모멘텀을 높임

경사하강법 최적화알고리즘2

❖ 영양성분 배합 사례

- 영양분을 함유한 원재료를 섞어서 음식을 만들되 재료비를 최소화하고 각 영양소별로 요구되는 조건을 만족해야하는 의사결정
- 비용은 가능한 줄이면서 학생들에게 필요한 영양소를 제공해야 함
  - 1인당 550칼로리에서 750칼로리 사이의 열량을 섭취
  - 단백질은 85그램 이상, 지방은 45그램 이상, 탄수화물은 90그램 이상 섭취
  - 식사의 전체량은 510그램 이상이 되어야 함
- 각 음식의 단위당 영양성분과 비용

음식(단위)	밥 (100g)	콩나물 (100g)	시금치 (50g)	불고기 (100g)	김치 (50g)
단백질(g)	5	10	3	40	5
지방(g)	5	10	2	20	2
탄수화물(g)	40	10	20	10	10
열량(Cal)	140	40	10	200	30
단위당 비용	500원	600원	200원	1000원	400원

- 의사결정변수(식단 구성재료의 양)

$x_1 =$ 밥의 양
$x_2 =$ 콩나물국의 양
$x_3 =$ 시금치의 양
$x_4 =$ 불고기의 양
$x_5 =$ 김치의 양

- 제약조건
  - 필요한 영양소 및 식사량 제공
- 선형계획모형

$$Minimize = 500x_1 + 600x_2 + 200x_3 + 1000x_4 + 400x_5$$
$$s.t$$
$$140x_1 + 40x_2 + 10x_3 + 200x_4 + 30x_5 \geq 550$$
$$140x_1 + 40x_2 + 10x_3 + 200x_4 + 30x_5 \leq 750$$
$$5x_1 + 10x_2 + 3x_3 + 40x_4 + 5x_5 \geq 85$$
$$5x_1 + 10x_2 + 2x_3 + 20x_4 + 2x_5 \geq 45$$
$$40x_1 + 10x_2 + 20x_3 + 10x_4 + 10x_5 \geq 90$$
$$100x_1 + 100x_2 + 50x_3 + 100x_4 + 50x_5 \geq 510$$
$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

경사하강법 최적화알고리즘2

❖ 자투리 최소화 문제

- JJ강철㈜에서는 직경 10cm짜리 파이프를 생산하여 판매하고 있음
- 파이프 수요: 5m, 7m, 9m인 파이프 수요 발생
- 제조공정에서 나온 30m짜리 표준 파이프를 절단하여 고객에게 판매
  - 주문: 5m짜리 120개, 7m짜리 190개, 9m짜리 150개
- 표준 길이 30m 파이프를 자르는 방법에 따라 자투리가 생기는데 이 자투리를 최소화하며 주문을 충족하기 위해서는 어떠한 방법으로 잘라서 공급하여야 하는가?
- 절단 방법과 규격별 산출량, 자투리

요구 규격	절단 방법										필요량
	1	2	3	4	5	6	7	8	9	10	
5m	6	4	3	1	4	2	1	0	2	1	120
7m	0	1	2	3	0	1	2	3	0	1	190
9m	0	0	0	0	1	1	1	1	2	2	150
자투리 길이	0	3	1	4	1	4	2	0	2	0	

- 의사결정변수

x_i = 절단 방법 i를 사용하여 절단한 표준품의 수량 $\quad i = 1, \ldots, 10$

- 목적함수
  - 발생하는 자투리 양의 최소화

자투리의 총량은 2, 3, 4, 5, 6, 7, 9 절단 방법에서 나오는 자투리들의 총합이므로,
$3x_2 + x_3 + 4x_4 + x_5 + 4x_6 + 2x_7 + 2x_9$

- 과잉생산량

$$5\text{m}: 5(6x_1 + 4x_2 + 3x_3 + x_4 + 4x_5 + 2x_6 + x_7 + 2x_9 + x_{10} - 120)$$
$$7\text{m}: 7(x_2 + 2x_3 + 3x_4 + x_6 + 2x_7 + 3x_8 + x_{10} - 190)$$
$$9\text{m}: 9(x_5 + x_6 + x_7 + x_8 + 2x_9 + 2x_{10} - 150)$$

경사하강법 최적화알고리즘2

❖ 자투리 최소화 문제

$$
\begin{aligned}
Min\ \ & 3x_2 + x_3 + 4x_4 + x_5 + 4x_6 + 2x_7 + 2x_9 \\
& + 5(6x_1 + 4x_2 + 3x_3 + x_4 + 4x_5 + 2x_6 + x_7 + 2x_9 + x_{10} - 120) \\
& + 7(x_2 + 2x_3 + 3x_4 + x_6 + 2x_7 + 3x_8 + x_{10} - 190) \\
& + 9(x_5 + x_6 + x_7 + x_8 + 2x_9 + 2x_{10} - 150)
\end{aligned}
$$

$$
\begin{aligned}
&\text{Min } z = \ x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} \\
&\text{s.t.} \\
&\quad 6x_1 + 4x_2 + 3x_3 + x_4 + 4x_5 + 2x_6 + x_7 + 2x_9 + x_{10} \geq 120 \\
&\quad x_2 + 2x_3 + 3x_4 + x_6 + 2x_7 + 3x_8 + x_{10} \geq 190 \\
&\quad x_5 + x_6 + x_7 + x_8 + 2x_9 + 2x_{10} \geq 150 \\
&\qquad\qquad x_i \geq 0
\end{aligned}
$$

❖ 수송문제

- 다수의 공급지(Source)에서 다수의 수요지(Destination)로 최소의 비용으로 제품들을 수송하기 위한 방안을 찾고자 하는 문제