경사하강법_매운맛

- 경사하강법으로 선형회귀 계수 구하기
- 선형회귀의 목적식: $\| y - X\beta \|_2$

  이를 최소화는 β를 찾기

  - $\nabla_\beta \| y - X\beta \|_2 = (\partial_{\beta_1} \| y - X\beta \|_2, \ldots, \partial_{\beta_d} \| y - X\beta \|_2)$
  - $\partial_{\beta_k} \| y - X\beta \|_2 = \partial_{\beta_k} \left\{ \frac{1}{n} \sum_{i=1}^{n} (yi - \sum_{j=1}^{d} X_{ij}\beta_j)^2 \right\}^{1/2}$
    - RMSE(Root Mean Squared Error): norm은 확작성을 가지고 있고, 관용적으로 RMSE를 L_2-norm처럼 사용한다.

    1. 벡터 간의 거리를 L_2-norm으로 계산
    2. SE(Squared Error)는 각 데이터별 정답과 예측값 간의 차이를 L_2-norm의 제곱으로 계산
    3. MSE(Mean Squared Error)는 SE에서 데이터 숫자만큼 나눔
    4. RMSE(Root Mean Squared Error)는 MSE에서 제곱근을 취함

    $= -\frac{X_k^T(y-X\beta)}{n\|y-X\beta\|_2}$ (행렬 X의 k번째 열벡터 전치)

    => $X\beta$를 계수 β에 대해 미분한 결과인 X^T만 곱해지는 것

- 목적식 초소화하는 β를 구하는 경사하강법 알고리즘

  $\beta^{(t+1)} \leftarrow \beta^{(t)} - \lambda \nabla_\beta \|y - X\beta^{(t)}\|$ (λ: learning rate)

- L_2-norm에 제곱을 해서 식을 단순화하기
  - $\| y - X\beta \|_2$ 대신 $\| y - X\beta \|_2^2$ 최소화

  $\nabla_\beta \| y - X\beta \|_2^2 = (\partial_{\beta_1} \| y - X\beta \|_2^2, \ldots, \partial_{\beta_d} \| y - X\beta \|_2^2)$

  $= -\frac{2}{n} X^T(y - X\beta)$

  $\beta^{(t+1)} \leftarrow \beta^{(t)} + \frac{2\lambda}{n} X^T(y - X\beta^{(t)})$

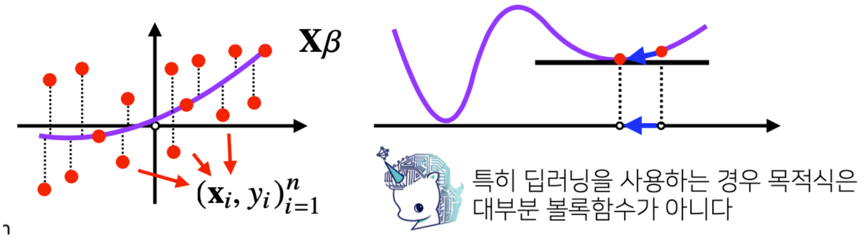
```
norm: L2-노름을 계산하는 함수
lr: 학습률, T = 학습횟수

for t in range(T): # 종료조건을 일정 학습횟수로 변경하는 점을 제외하고는 경사하강법 알고리즘과 같다
  error = y - X @ beta
  grad = - transpose(X) @ error
  beta = beta - lr * grad
```

  - $\nabla_\beta \| y - X\beta \|_2^2$항을 계산해서 β 업데이트
  - **학습률 & 학습횟수** => 중요한 hyperparameter

```python
X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
y = np.dot(X, np.array([1, 2])) + 3

beta_gd = [10.1, 15.1, -6.5] # [1, 2, 3] 이 정답
X_ = np.array([np.append(x,[1]) for x in X]) # intercept 항 추가

for t in range(5000):
    error = y - X_ @ beta_gd
    # error = error / np.linalg.norm(error)
    grad = - np.transpose(X_) @ error
    beta_gd = beta_gd - 0.01 * grad

print(beta_gd)
```

```
[1.00000367 1.99999949 2.99999516]
```

- 이론적으로 경사하강법은 **미분가능하고 볼록(convex)한 함수**에 대한 **적절한 학습률과 학습횟수 선택 시 수렴 보장**
- 특히 **선형회귀**의 경우, 목적식 $\| y - X\beta \|_2$은 **회귀계수 β에 대해 블록함수** => 수렴 보장
- **비선형회귀** 경우, **목적식이 볼록하지 않을 수 있음** => **수렴 항상 보장 X**



- **볼록이 아닌(non-convex) 목적식**: SGD를 통해 **최적화** 가능

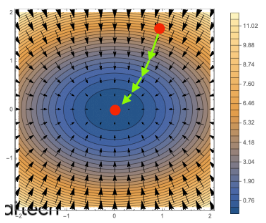
$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \widehat{\nabla_\theta \mathcal{L}}(\theta^{(t)}) \qquad \mathbb{E}[\widehat{\nabla_\theta \mathcal{L}}] \approx \nabla_\theta \mathcal{L}$$

- 딥러닝의 경우 SGD가 BGD보다 실증적으로 더 낫다고 검증됨

$$\beta^{(t+1)} \leftarrow \beta^{(t)} + \frac{2\lambda}{n}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\beta^{(t)}) \xrightarrow{O(d^2n) \to O(d^2b)} \beta^{(t+1)} \leftarrow \beta^{(t)} + \frac{2\lambda}{b}\mathbf{X}_{(b)}^\top(\mathbf{y}_{(b)} - \mathbf{X}_{(b)}\beta^{(t)})$$

전체 데이터 $(\mathbf{X}, \mathbf{y})$ 를 쓰지 않고 미니배치 $(\mathbf{X}_{(b)}, \mathbf{y}_{(b)})$ 를 써서 업데이트 하므로 연산량이 b/n 로 감소한다

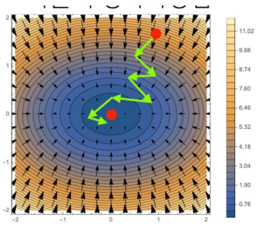
- **배치 경사하강법(Batch Gradient Descent, BGD)**



  - batch: **전체 학습 데이터셋 의미**
  - **전체 학습데이터를 하나의 배치로 묶어 학습**시키는 경사하강법
  - 전체 데이터를 통해 학습 => 모델 파라미터 업데이트 횟수 적음

    (1 Epoch 당 1회 업데이트)

    But 한 스텝에 모든 학습 데이터 사용 => 학습 오래 거림
  - 전체 데이터 모두 한 번에 처리 => 메모리 가장 많이 필요
  - 항상 같은 데이터(전체)에 대해 경사를 구함 => 수렴 안정적
  - local minima 상태가 되면 빠져나오기 힘듬
- **확률적 경사하강법(Stochastic Gradient descent, SGD)**



- **모든 데이터 사용 X, 전체 데이터 중 랜덤하게 선택된 단 하나의 데이터(배치사이즈 = 1)를 이용해 학습**시키는 경사하강법
- BGD보다 적은 데이터 학습 => 속도 빠름
- 연산자원을 좀 더 효율적으로 활용하는데 도움
- 데이터 한 개씩 처리 => GPU 성능 전부 활용 X
- 수렴속도 빠름, but 각 데이터의 손실값 기울기는 약간씩 다름 => shooting 발생
- shooting은 local minima에 빠질 확률 줄어줌
- 보편적, 전체 학습 데이터에 대한 좋은 값을 방향으로 수렴, but global minima에 수렴 어려움

- **미니배치 확률적 경사하강법(Mini-Batch Stochastic Gradient Descent)**
  - BGD, SGD의 절충안(그래프도 그 중간의 모습)
  - **전체데이터를 배치 사이즈(사용자 지정)만큼 나눠 미니배치로 학습**시키는 경사하강법
  - 한 배치의 손실값의 평균으로 학습 => BGD보다 빠르고 SGD보다 낮은 오차율 가짐(shooting 적음)
  - SGD보다 SPU 병렬 연산 유리
  - local minima에 빠질 리스크 적음
  - 배치사이즈 설정 필요: 보통 2의 제곱수 이용, 본인의 GPU에 따른 out of memory 발생 않도록 설정해야함
  - 마지막 남은 배치가 다른 사이즈일 시, 해당 배치의 데이터가 학습에 더 큰 비중을 가짐 => 학습 데이터 갯수에 나누어 떨어지도록 지정 권장

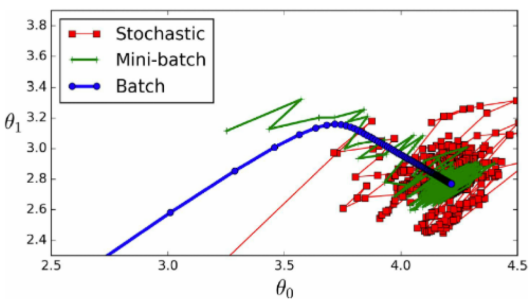


Figure 4-11. Gradient Descent paths in parameter space

- 배치: 경사 하강법 1회에 사용되는 데이터의 묶음
  - (배치 크기) == (전체 학습 데이터) => 배치 경사 하강법 (BGD)
  - (배치 크기) == 1 => 확률적 경사 하강법 (SGD)
  - (배치 크기) == (사용자가 지정) => 미니 배치 확률적 경사 하강법 (MSGD)

> ⚠️ MSGD, SGD는 다른 알고리즘이지만 요즘에는 MSGD를 SGD라고 혼용해서 부름