Inclass 08: Gradient Descent

[SCS4049] Machine Learning and Data Science

Seongsik Park (s.park@dgu.edu)

Department of Artificial Intelligence, Dongguk University

Batch gradient descent

Linear regression model $\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$

$$J(\boldsymbol{\theta}) = \mathrm{SSE}(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2 \tag{1}$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = 2\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) \tag{2}$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial}{\partial \theta_1} J(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} J(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} 2\sum_{i=1}^{m} x_1^{(i)} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}\right) \\ 2\sum_{i=1}^{m} x_2^{(i)} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}\right) \\ \vdots \\ 2\sum_{i=1}^{m} x_n^{(i)} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}\right) \end{bmatrix} \tag{3}$$

Gradient descent step

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^t) \tag{4}$$

where iteration number t and $\boldsymbol{\theta}$ arbitrary initial value

Batch gradient descent

Batch gradient descent에서

$$\frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = 2\mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) \tag{5}$$

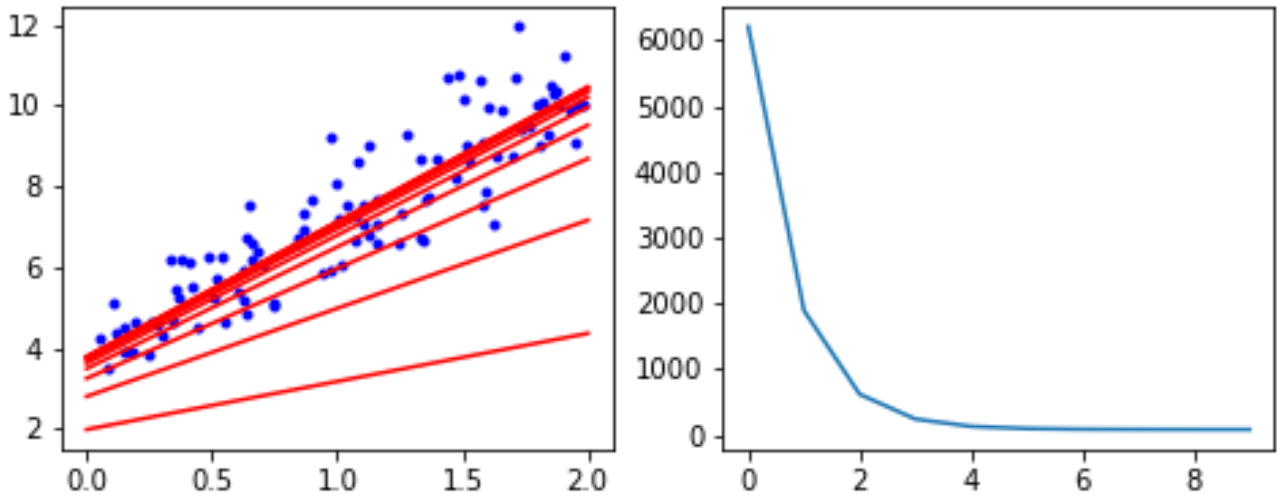
$$= 2 \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \cdots & \mathbf{x}2^{(m)} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta}^T\mathbf{x}^{(1)} - y^{(1)} \\ \boldsymbol{\theta}^T\mathbf{x}^{(2)} - y^{(2)} \\ \vdots \\ \boldsymbol{\theta}^T\mathbf{x}^{(m)} - y^{(m)} \end{bmatrix} \tag{6}$$

$$= 2\sum_{i=1}^{m} \mathbf{x}^{(i)} \left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)} \right) \tag{7}$$

그러므로 이 gradient vector의 j번째 component는

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = 2\sum_{i=1}^{m} x_j^{(i)} \left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)} \right) \tag{8}$$

Batch gradient descent



Batch gradient descent: computational complexity

Batch gradient descent algorithm

- 매 스텝마다 batch 전체에 대한 계산 필요
- 데이터셋이 커지면 속도가 느려짐
- Normal equation: feature 수에 따라 계산 속도가 지수적으로 느려짐
- Gradient descent: feature 수가 늘어도 크게 변하지 않음

Learning rate

- Hyperparameter인 학습률(learning rate) η가 너무 작은 경우 시간이 오래 걸림
- 너무 큰 경우 최적해를 지나쳐 해를 찾지 못할 수 있음

Learning schedule

- Constant learning rate
  - 보통 0.1, 0.01부터 시작하여 여러 가지 값으로 시험해보며 범위를 좁혀 나감
- Time-based decay

$$\eta = \frac{\eta_0}{(1 + kt)} \tag{9}$$

η_0 : 학습률 초기값, k : hyperparameter, t : iteration

- Step decay
  - 정해진 epoch마다 학습률을 줄이는 방법
  - 예: 5 epoch마다 반으로, 20 epoch마다 1/10로
  - Epoch: 훈련 데이터셋 전체를 모두 사용할 때 = 한 epoch
- Exponential decay

$$\eta = \eta_0 e^{-kt} \tag{10}$$

η_0 : 학습률 초기값, k : hyperparameter, t : iteration

Stochastic gradient descent

For our linear regression model $\hat{y} = \boldsymbol{\theta}^T \mathbf{x}$

$$J(\boldsymbol{\theta}) = \mathrm{SSE}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \left(\mathbf{y}^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)} \right)^2 = \sum_{i=1}^{m} J_i(\boldsymbol{\theta}) \tag{11}$$

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - 2\eta \left(y^{(t)} - (\boldsymbol{\theta}^t)^T \mathbf{x}^{(t)} \right) \mathbf{x}^{(t)} \tag{12}$$

- 무작위로 선택한 한 개의 sample에 대해서만 gradient를 계산하여 parameter를 update
- sequential learning or online learning
- 대규모 데이터셋을 처리하는데 유리
- 선택하는 사례의 무작위성으로 움직임이 불규칙
- BGD에 비해 local optimum에서 쉽게 빠져나올 수 있음
- 최적해에 도달하지만 지속적으로 요동
- BGD와 마찬가지로 global optimum이라는 보장이 없음

Mini-batch gradient descent

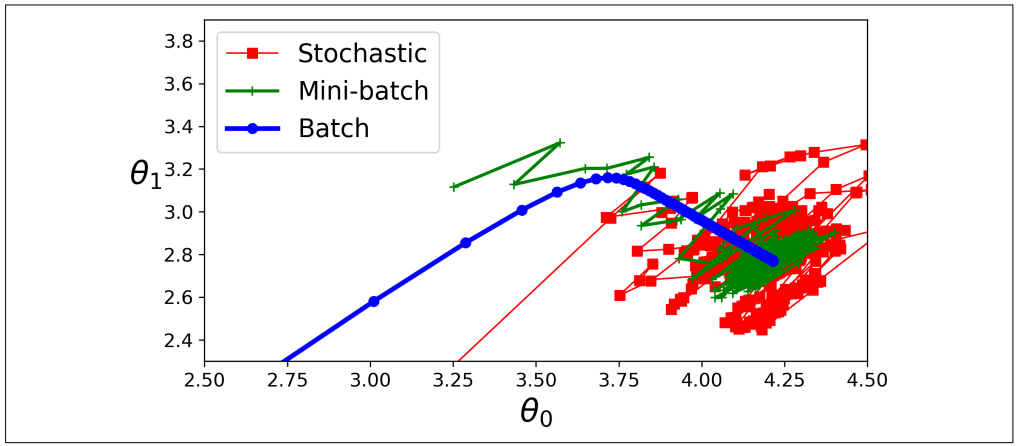


Figure 4-11. Gradient Descent paths in parameter space

- 훈련 데이터셋을 작은 크기의 무작위 부분 집합으로 나누어서 gradient를 구하는 방법
- 예 100,000개의 데이터 = (mini-batch size 100) $\times$ (1,000 mini-batches)
- Batch gradient descent와 stochastic gradient descent(SGD)의 절충
- SGD보다 불규칙한 움직임이 덜함
- SGD보다 local minimum에서 빠져나오기가 상대적으로 더 어려움
- GPU를 통한 매트릭스 연산의 속도를 높일 수 있음

Linear regression comparison

Table 4-1. Comparison of algorithms for Linear Regression

Algorithm	Large *m*	Out-of-core support	Large *n*	Hyperparams	Scaling required	Scikit-Learn
Normal Equation	Fast	No	Slow	0	No	n/a
SVD	Fast	No	Slow	0	No	`LinearRegression`
Batch GD	Slow	No	Fast	2	Yes	`SGDRegressor`
Stochastic GD	Fast	Yes	Fast	≥2	Yes	`SGDRegressor`
Mini-batch GD	Fast	Yes	Fast	≥2	Yes	`SGDRegressor`

There is almost no difference after training: all these algorithms end up with very similar models and make predictions in exactly the same way.