Statistical Machine Learning

3주차

담당 : 11기 명재성

Review

- Regression

$$Y_i \overset{ind}{\sim} N(\mu_i(\mathbf{X}_i),\ \sigma) \qquad \text{where} \quad E[Y_i] = \mu_i(\mathbf{X}_i)$$

$$\mu_i(\mathbf{X}_i) = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_p X_{pi}$$

$$= \boldsymbol{\beta}^T \mathbf{X}_i$$

Review

- Logistic Regression

$$Y_i \overset{ind}{\sim} \text{Bernoulli}(\pi_i(\mathbf{X}_i)) \quad \text{where} \quad E[Y_i] = \pi_i(\mathbf{X}_i)$$

$$logit(\pi_i(\mathbf{X}_i)) = \log\left(\frac{\pi_i(\mathbf{X}_i)}{1-\pi_i(\mathbf{X}_i)}\right) = \beta_0 + \beta_1 X_{1\mathrm{i}} + \beta_2 X_{2\mathrm{i}} + \cdots + \beta_p X_{p\mathrm{i}}$$

$$= \boldsymbol{\beta}^T \mathbf{X}_i$$

Review

- Estimation

$$\underset{\boldsymbol{\beta}}{argmin}\, L[\tau(\theta), T(X)] \Leftrightarrow \underset{\boldsymbol{\beta}}{argmax}\, L(\boldsymbol{\beta}, \sigma)$$

- Regression → SSE

- Logistic Regression → Cross Entropy

Review

- Cross Entropy

$$CE_i = -\sum_{k=1}^{C} y_{ik} \log \pi_i(k)$$

y_{ik} : the k^{th} value in y_i

$\pi_i(k)$: the probability for the i^{th} observation to belonging to Class k

Review

- For C = 3 (number of Class)

Class 1: y_i =(1,0,0) Class 2: y_i =(0,1,0) Class 3: y_i =(0,0,1)

⇒ One-Hot encoding

$$\sum_{i=1}^{n} CE_i = -\sum_{k=1}^{C} [y_{i1} \log \pi_i(1) + y_{i2} \log \pi_i(2) + y_{i3} \log \pi_i(3)]$$

Review

- IF Class 2,

Class 1: y_i =(1,0,0)　　Class 2: y_i =(0,1,0)　　Class 3: y_i =(0,0,1)

$$CE_i = -[0 * \log \pi_i(1) + 1 * \log \pi_i(2) + 0 * \log \pi_i(3)]$$

$$= -\log \pi_i(2)$$

$\Rightarrow$ IF $\pi_i(2) = 1$, then $CE_i = -\log 1 = 0$ (minimum Loss)

Review

- Categorical Cross Entropy

$$CE_i = -\sum_{k=1}^{C} y_{ik} \log \pi_i(k)$$

- Likelihood of ?

$$Y_i \overset{ind}{\sim} Multi(\pi_1, \cdots, \pi_C)$$

Review

- For C = 2 (number of Class)

$$P(Y_i = 1|\mathbf{X}_i) = \pi_i(\mathbf{X}_i) = \frac{e^{\boldsymbol{\beta}^T\mathbf{X}_i}}{1 + e^{\boldsymbol{\beta}^T\mathbf{X}_i}} = \frac{1}{1 + e^{-\boldsymbol{\beta}^T\mathbf{X}_i}}$$ (sigmoid function)

Review

- For C = 2 (number of Class)

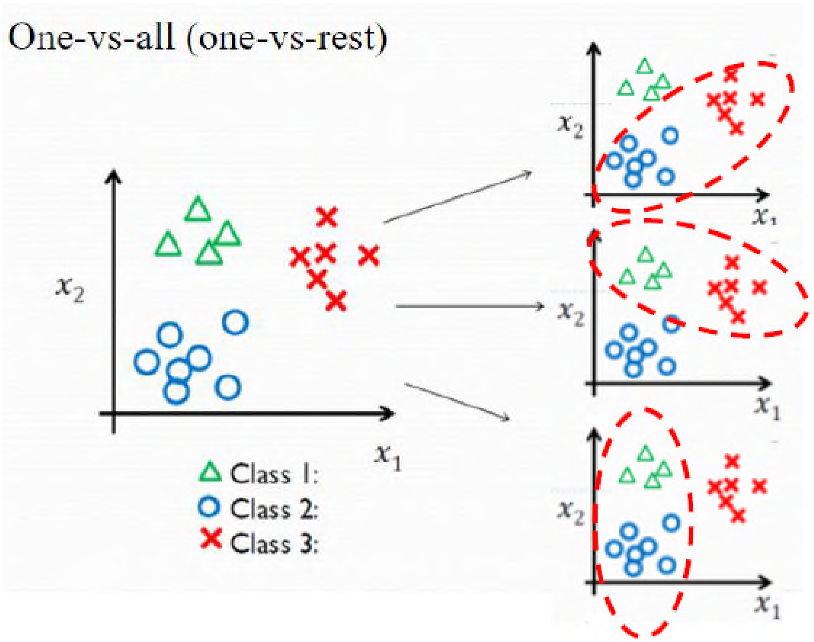
$$P(Y_i = 1|\mathbf{X}_i) = \pi(\mathbf{X}_i) = \frac{e^{\boldsymbol{\beta}^T\mathbf{X}_i}}{1 + e^{\boldsymbol{\beta}^T\mathbf{X}_i}} = \frac{1}{1 + e^{-\boldsymbol{\beta}^T\mathbf{X}_i}}$$ (sigmoid function)

- For C = K > 2 (number of Class)

$$P(Y_i = l|\mathbf{X}_i) = \pi_l(\mathbf{X}_i) = \frac{e^{\boldsymbol{\beta}_l^T\mathbf{X}_i}}{\sum_{c=1}^{K} e^{\boldsymbol{\beta}_c^T\mathbf{X}_i}}$$ (softmax function)

Review

- One-Vs-Rest



Review

- One-Vs-Rest

```
LogisticRegression(solver='sag', multi_class='multinomial')
```

```
'ovr'
```

Review

- Naïve Bayes Classifier

$$P(Y_i = k|\mathbf{X}_i) = \frac{P(\mathbf{X}_i|k)P(k)}{\sum_k P\,(\mathbf{X}_i|k)P(k)} \qquad Bayes'\ Theorem$$

$$\text{where} \quad P(\mathbf{X}_i|k) = \prod_j^p P\,(X_{ij}|k)$$

Review

- LDA

$$P(Y_i = k|\mathbf{X}_i) = \frac{P(\mathbf{X}_i|k)P(k)}{\sum_k P\,(\mathbf{X}_i|k)P(k)} \qquad Bayes'\ Theorem$$

$$\text{where} \quad P(\mathbf{X}_i|k) \sim N_p(\boldsymbol{\mu}_k, \Sigma)$$

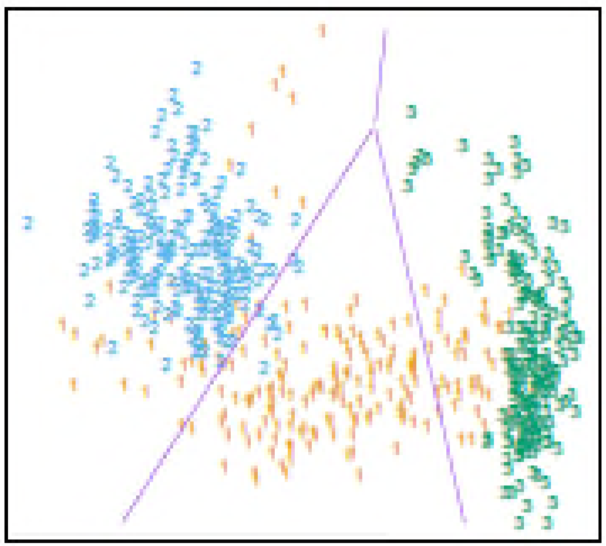
Review

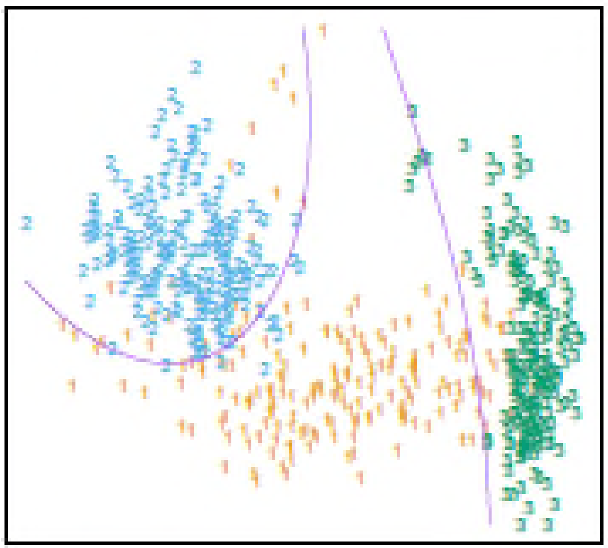
- QDA

$$P(Y_i = k|\mathbf{X}_i) = \frac{P(\mathbf{X}_i|k)P(k)}{\sum_k P\,(\mathbf{X}_i|k)P(k)} \qquad Bayes'\ Theorem$$

$$\text{where} \quad P(\mathbf{X}_i|k) \sim N_p(\boldsymbol{\mu}_k, \Sigma_k)$$

Review





Review

- 0-1 Loss

$$L[\tau(\theta), T(X)] = \sum I(Y_i \neq \hat{Y}_i)$$

- The Bayes decision rule for minimizing the loss ($\mathrm{P}\left(Y_i \neq \hat{Y}_i\right)$) is

$$\underset{\mathrm{k}}{argmax}\ P(Y = k|\mathbf{X})$$

Optimization

- Loss function

$$L[\tau(\theta), T(X)] = L[Y,\ \hat{Y}]$$

$$Regression \quad \Rightarrow \quad L[Y,\ \hat{Y}] = \sum_{i}^{n} (Y_i - \hat{Y}_i)^2$$

$$Classification \quad \Rightarrow \quad L[Y,\ \hat{Y}] = -\sum_{i}^{n} \sum_{k}^{C} Y_i \ log\hat{\pi}_i(k)$$

Optimization

- Machine “Learning”

$$\underset{\boldsymbol{\theta}}{argmin}\ L\left[Y,\ \hat{Y}\right] = \widehat{\boldsymbol{\theta}}$$

$$\Rightarrow\ Optimization$$

Optimization

- Logistic Regression

$$L[Y,\ \hat{Y}] = -\sum_{i=1}^{n}[\,y_i \text{log}\pi_i + (1-y_i)\text{log}(1-\pi_i)]$$

$$= -\sum_{i=1}^{n}[\,y_i(\boldsymbol{\beta}^T\mathbf{X}_i) - \text{log}(1+\text{exp}(\boldsymbol{\beta}^T\mathbf{X}_i)]$$

Optimization

- Logistic Regression

$$L[Y,\ \hat{Y}] = -\sum_{i=1}^{n}[\,y_i(\boldsymbol{\beta}^T\mathbf{X}_i) - \log(1 + \exp(\boldsymbol{\beta}^T\mathbf{X}_i)]$$

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{argmin}\ L[Y,\ \hat{Y}]$$

⇒ Can you solve it?

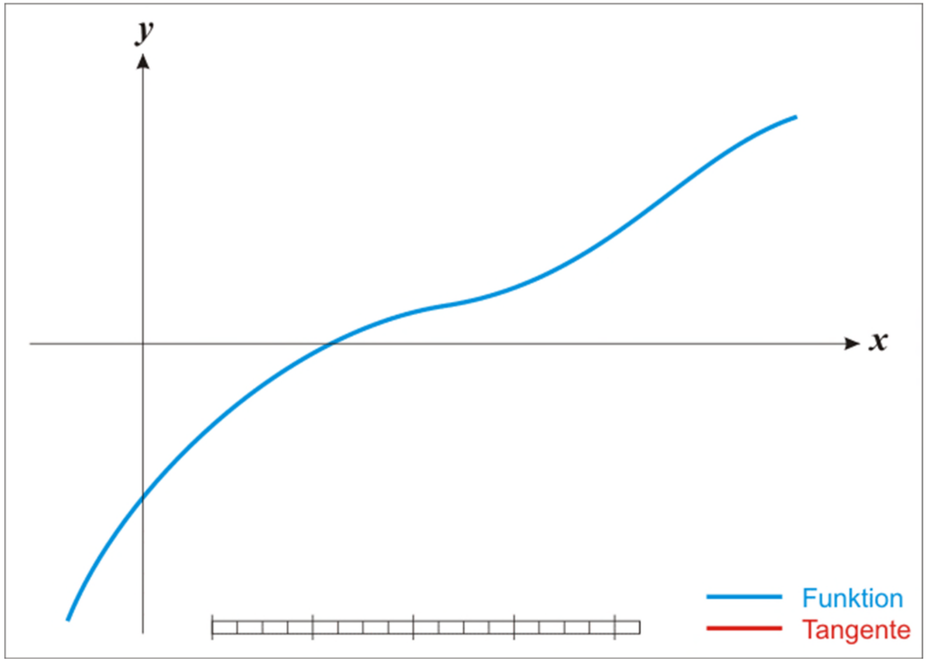
Optimization

- Optimization often can be rewritten as solving equations.

  ex) Normal equation

- Some problems do not have an explicit solution and a numerical approach should be exploited.

Newton-Raphson Method



Newton-Raphson Method

- Linear approximation (1st order Taylor Expansion)

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) = 0$$

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$\Rightarrow \quad \theta^{(\mathrm{t}+1)} = \theta^{(\mathrm{t})} - \frac{f(\theta^{(\mathrm{t})})}{f'(\theta^{(\mathrm{t})})}$$

Newton-Raphson Method

1. Initialize $\theta^{(0)} = \theta_0$ which can be arbitrary on the domain of the function

2. Update for $\mathrm{t} = 0, 1, 2, 3, \cdots$

$$\theta^{(\mathrm{t}+1)} = \theta^{(\mathrm{t})} - \frac{f\left(\theta^{(\mathrm{t})}\right)}{f'\left(\theta^{(\mathrm{t})}\right)}$$

until

$$\left|\,\theta^{(\mathrm{t}+1)} - \theta^{(\mathrm{t})}\,\right| < \epsilon$$

for small $\epsilon > 0$

Newton-Raphson Method

- Quadratic approximation (2nd order Taylor Expansion)

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(x_0)(x - x_0)^2$$

$$\frac{\partial}{\partial x} f(x) \approx f'(x_0) + f''(x_0)(x - x_0) = 0$$

$$x = x_0 - \frac{f'(x_0)}{f''(x_0)} \quad \Rightarrow \quad \theta^{(\mathrm{t}+1)} = \theta^{(\mathrm{t})} - \frac{f'(\theta^{(\mathrm{t})})}{f''(\theta^{(\mathrm{t})})}$$

Newton-Raphson Method

- Quadratic approximation (2nd order Taylor Expansion)

$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}_0) + L'(\boldsymbol{\theta}_0)^T(\boldsymbol{\theta} - \boldsymbol{\theta}_0) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T L''(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

Newton-Raphson Method

- Quadratic approximation (2nd order Taylor Expansion)

$$L(\boldsymbol{\theta}) \approx L(\boldsymbol{\theta}_0) + \nabla L(\boldsymbol{\theta}_0)^T(\boldsymbol{\theta} - \boldsymbol{\theta}_0) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \mathbf{H}(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

where

$$\nabla L(\boldsymbol{\theta}_0) = \frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta}) \Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}$$

$$\mathbf{H}(\boldsymbol{\theta}_0) = \frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} L(\boldsymbol{\theta}) \Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}$$

Newton-Raphson Method

- Updating equation is

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{(t)})\nabla L(\boldsymbol{\theta}^{(t)})$$

$$= \boldsymbol{\theta}^{(t)} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{(t)})\frac{\partial}{\partial \boldsymbol{\theta}^{(t)}} L(\boldsymbol{\theta}^{(t)})$$

$$cf. \quad \theta^{(\mathrm{t}+1)} = \theta^{(\mathrm{t})} - \frac{f'(\theta^{(\mathrm{t})})}{f''(\theta^{(\mathrm{t})})}$$

Newton-Raphson Method

$$Loss[\boldsymbol{\beta}] = -\sum_{i=1}^{n}[\,y_i(\boldsymbol{\beta}^T\mathbf{X}_i) - \log(1+\exp(\boldsymbol{\beta}^T\mathbf{X}_i)]$$

$$\nabla L(\boldsymbol{\beta}^{(t)}) = \frac{\partial}{\partial\boldsymbol{\beta}^{(t)}}L(\boldsymbol{\beta}^{(t)}) = -\sum_{i=1}^{n}\left[y_i\mathbf{X}_i - \frac{\exp({\boldsymbol{\beta}^{(t)}}^T\mathbf{X}_i)}{1+\exp({\boldsymbol{\beta}^{(t)}}^T\mathbf{X}_i)}\mathbf{X}_i\right]$$

$$\mathbf{H}(\boldsymbol{\beta}^{(t)}) = \frac{\partial^2}{\partial\boldsymbol{\beta}^{(t)}\partial{\boldsymbol{\beta}^{(t)}}^T}L(\boldsymbol{\beta}^{(t)}) = \sum_{i=1}^{n}\left[\left(\frac{\exp({\boldsymbol{\beta}^{(t)}}^T\mathbf{X}_i)}{1+\exp({\boldsymbol{\beta}^{(t)}}^T\mathbf{X}_i)}\right)\left(\frac{1}{1+\exp({\boldsymbol{\beta}^{(t)}}^T\mathbf{X}_i)}\right)\mathbf{X}_i\mathbf{X}_i^T\right]$$

Newton-Raphson Method

$$Loss[\boldsymbol{\beta}] = -\sum_{i=1}^{n}[y_i(\boldsymbol{\beta}^T\mathbf{X}_i) - \log(1 + \exp(\boldsymbol{\beta}^T\mathbf{X}_i)]$$

Update

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \mathbf{H}^{-1}\left(\boldsymbol{\beta}^{(t)}\right)\nabla L\left(\boldsymbol{\beta}^{(t)}\right)$$

until

$$||\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^{(t)}|| < \epsilon \qquad \text{for small } \epsilon > 0$$

Newton-Raphson Method

					Solvers
Penalties	**'liblinear'**	**'lbfgs'**	**'newton-cg'**	**'sag'**	**'saga'**
Multinomial + L2 penalty	no	yes	yes	yes	yes
OVR + L2 penalty	yes	yes	yes	yes	yes
Multinomial + L1 penalty	no	no	no	no	yes
OVR + L1 penalty	yes	no	no	no	yes
Behaviors					
Penalize the intercept (bad)	yes	no	no	no	no
Faster for large datasets	no	no	no	yes	yes
Robust to unscaled datasets	yes	yes	yes	no	no

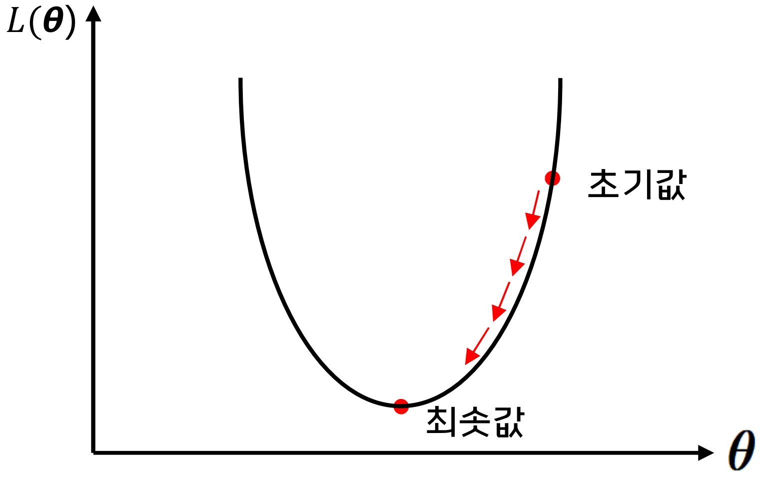
Gradient Descent

- Newton-Raphson is expensive to compute due to the computation of the inverse of Hessian.

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \mathbf{H}^{-1}(\boldsymbol{\theta}^{(t)})\nabla L(\boldsymbol{\theta}^{(t)})$$

$$\Rightarrow \quad \boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta^{(t)}\, \nabla L(\boldsymbol{\theta}^{(t)})$$

Gradient Descent



Gradient Descent

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta^{(t)}\, \nabla L(\boldsymbol{\theta}^{(t)})$$

or

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta\, \nabla L(\boldsymbol{\theta}^{(t)})$$

```
t0, t1 = 5, 50  # learning schedule hyperparameters

def learning_schedule(t):
    return t0 / (t + t1)
```

```
eta = learning_schedule(epoch * m + i)
theta = theta - eta * gradients
```

Batch Gradient Descent

- Regression → SSE

$$\nabla L(\boldsymbol{\beta}) = \frac{\partial}{\partial \boldsymbol{\beta}} L(\boldsymbol{\beta}) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

- Logistic Regression → Cross Entropy

$$\nabla L(\boldsymbol{\beta}) = \frac{\partial}{\partial \boldsymbol{\beta}} L(\boldsymbol{\beta}) = -\sum_{i=1}^{n}\left[y_i \mathbf{X}_i - \frac{\exp(\boldsymbol{\beta}^T \mathbf{X}_i)}{1 + \exp(\boldsymbol{\beta}^T \mathbf{X}_i)} \mathbf{X}_i\right]$$

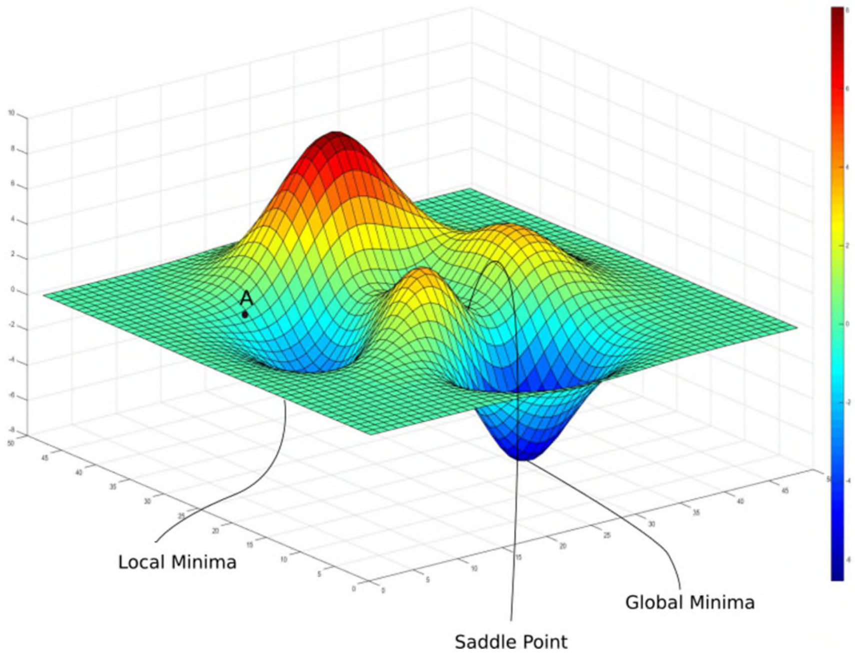
Steepest Descent

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \frac{\partial}{\partial \boldsymbol{\theta}^{(t)}} L(\boldsymbol{\theta}^{(t)}) \quad \Leftrightarrow \quad (\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}) \propto -\frac{\partial L(\boldsymbol{\theta}^{(t)})}{\partial \boldsymbol{\theta}^{(t)}}$$

$$\frac{L(\boldsymbol{\theta}^{(t+1)})}{\partial \eta} = [\frac{\partial L(\boldsymbol{\theta}^{(t+1)})}{\partial \boldsymbol{\theta}^{(t+1)}}]^T \frac{\partial \boldsymbol{\theta}^{(t+1)}}{\partial \eta} = -[\frac{\partial L(\boldsymbol{\theta}^{(t+1)})}{\partial \boldsymbol{\theta}^{(t+1)}}]^T \frac{\partial L(\boldsymbol{\theta}^{(t)})}{\partial \boldsymbol{\theta}^{(t)}} \overset{\text{set}}{=} 0$$

$\Rightarrow$ $\boldsymbol{\theta}^{(t+2)}$ and $\boldsymbol{\theta}^{(t+1)}$ are orthogonal

Stochastic Gradient Descent



Stochastic Gradient Descent

$$\underbrace{\boxed{\theta^{(1)}}\boxed{\theta^{(2)}}\boxed{\theta^{(3)}}\boxed{\quad\cdots\cdots\quad}\boxed{\theta^{(n-1)}}\boxed{\theta^{(n)}}}_{\text{1 epoch}}$$

- Stochastic (Randomness) $\rightarrow$ Shuffle the data

Stochastic Gradient Descent

```python
from sklearn.linear_model import SGDRegressor

sgd_reg = SGDRegressor(loss='squared_loss',
                       max_iter=50, tol=-np.infty, penalty=None, eta0=0.1, random_state=42)
sgd_reg.fit(X, y.ravel())

sgd_reg.intercept_, sgd_reg.coef_
```

```
(array([4.16782089]), array([2.72603052]))
```

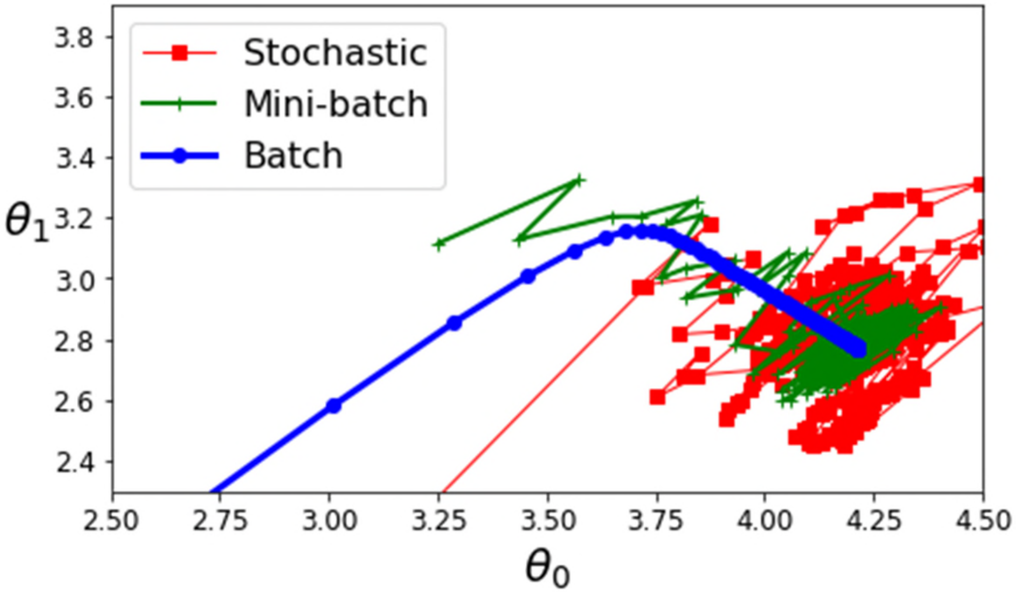
```python
from sklearn.linear_model import SGDClassifier

sdg_cls = SGDClassifier(loss='log',
                        max_iter=50, tol=-np.infty, penalty=None, eta0=0.1, random_state=42))
```

Mini-Batch Gradient Descent

$$\underbrace{\boxed{\theta^{(1)}}\boxed{\theta^{(2)}}\boxed{\cdots\cdots}\boxed{\theta^{(k)}}}_{\text{1 epoch}}$$

- k batches have p data → n = k x p

Mini-Batch Gradient Descent



Gradient Descent

- What about $\eta^{(t)}$? $\rightarrow$ in Deep Learning

reference

자료

19-2 STAT424 통계적 머신러닝 - 박유성 교수님

교재

파이썬을 이용한 통계적 머신러닝 (2020) – 박유성

ISLR (2013) - G. James, D. Witten, T. Hastie, R. Tibshirani

The elements of Statistical Learning (2001) - J. Friedman, T. Hastie, R. Tibshirani

Hands on Machine Learning (2017) - Aurelien Geron