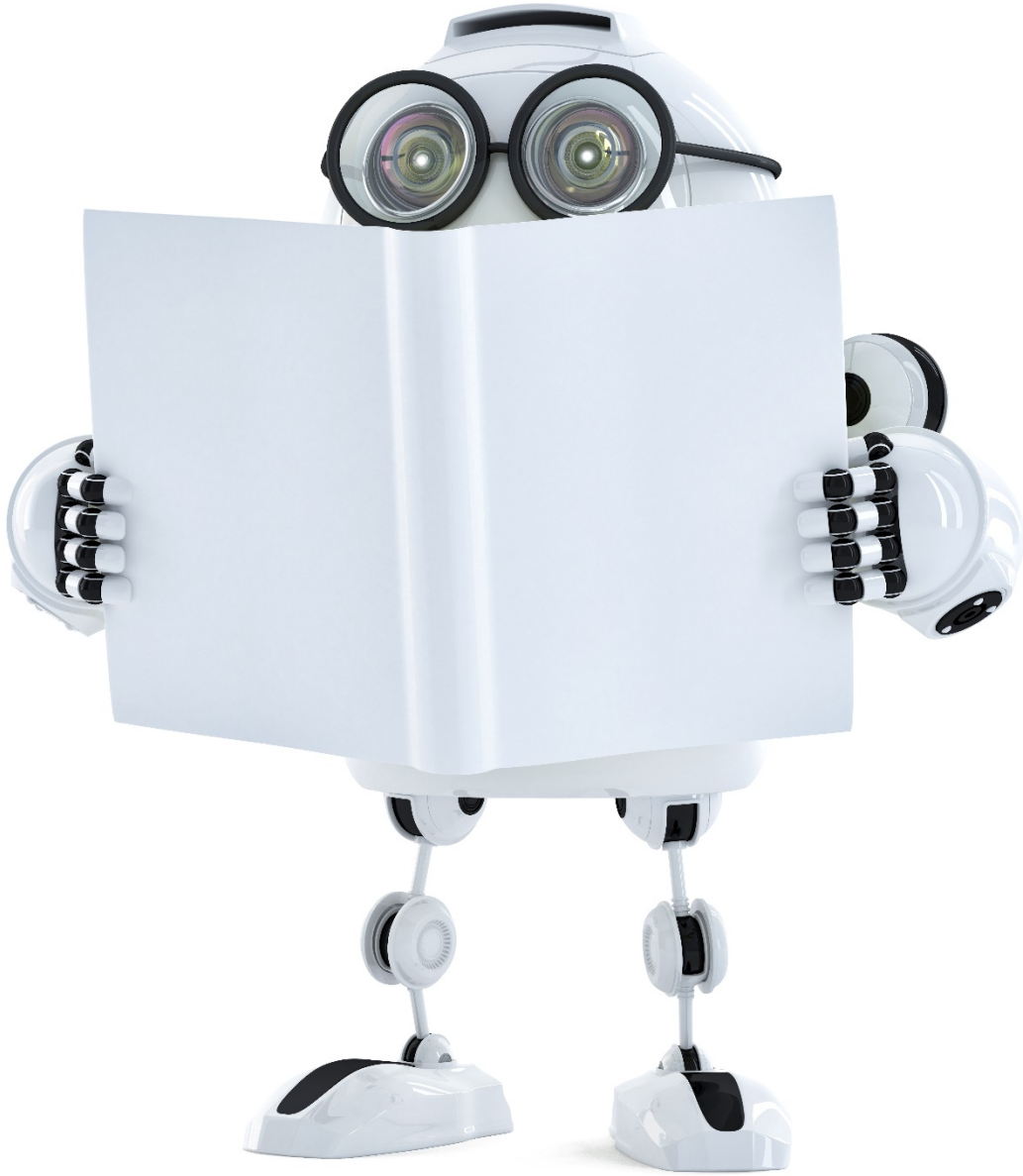
SGD implementation issues

Linear Regression

Director of TEAMLAB
Sungchul Choi



SGD를 구현할 때 생각해봐야 할 일 들

Mini-Batch SGD

```
for epoch in range(epoches):
    X_copy = np.copy(X)
    if is_SGD:
        np.random.shuffle(X_copy)
    batch = len(X_copy) // BATCH_SIZE
    for batch_count in range(batch):
        X_batch = np.copy(
                X_copy[batch_count*BATCH_SIZE : (batch_count+1)*BATCH_SIZE])
        # Do weight Update
    print("Number of epoch : %d" % epoch)
```

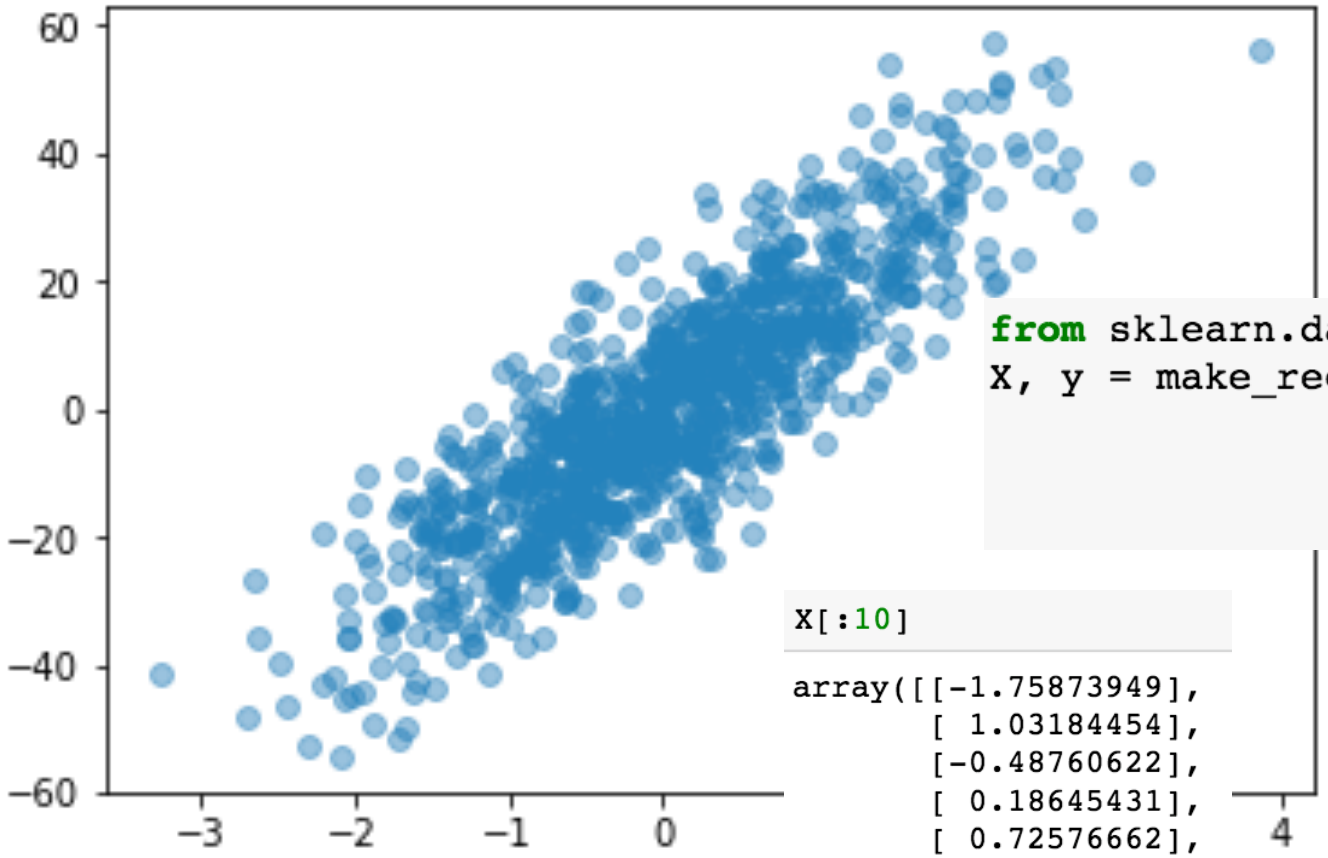
전체 Epoch이 iteration 되는 횟수

SGD 여부 → SGD일 경우 shuffle

한번에 처리하는 BATCH_SIZE

BATCH_SIZE 크기 만큼 X_batch 생성

Convergence process




```
from sklearn.datasets.samples_generator import make_regression
X, y = make_regression(n_samples = 1000,
                       n_features=1,
                       noise=10,
                       random_state=42)
```

```
X[:10]
```

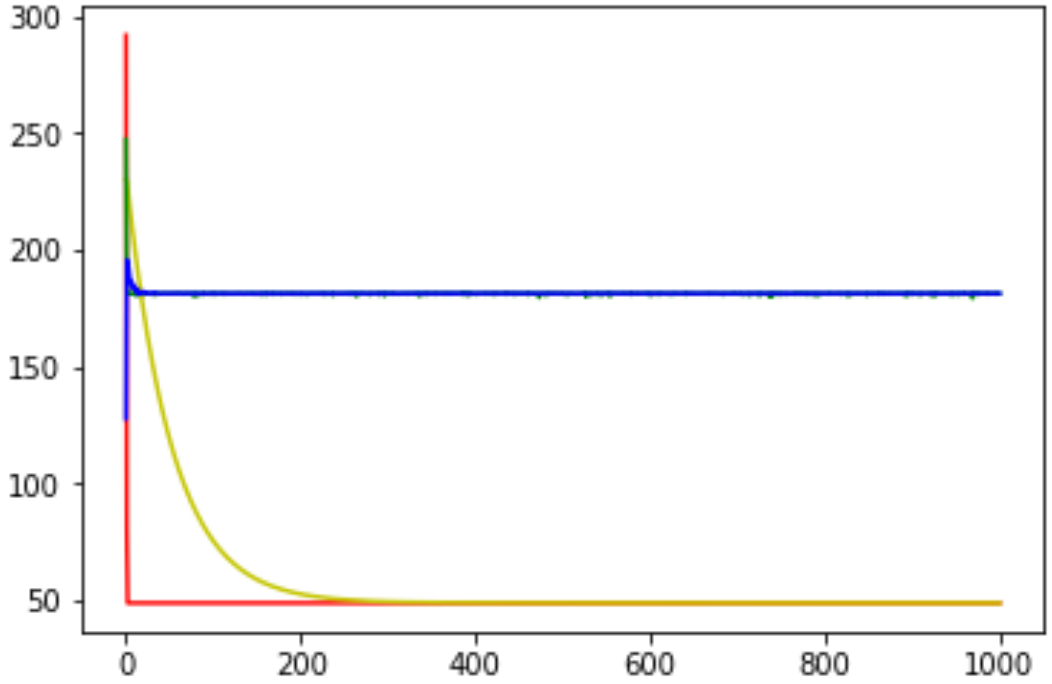
```
array([[-1.75873949],
       [ 1.03184454],
       [-0.48760622],
       [ 0.18645431],
       [ 0.72576662],
       [ 0.97255445],
       [ 0.64537595],
       [ 0.68189149],
       [-1.43014138],
       [ 1.06667469]])
```

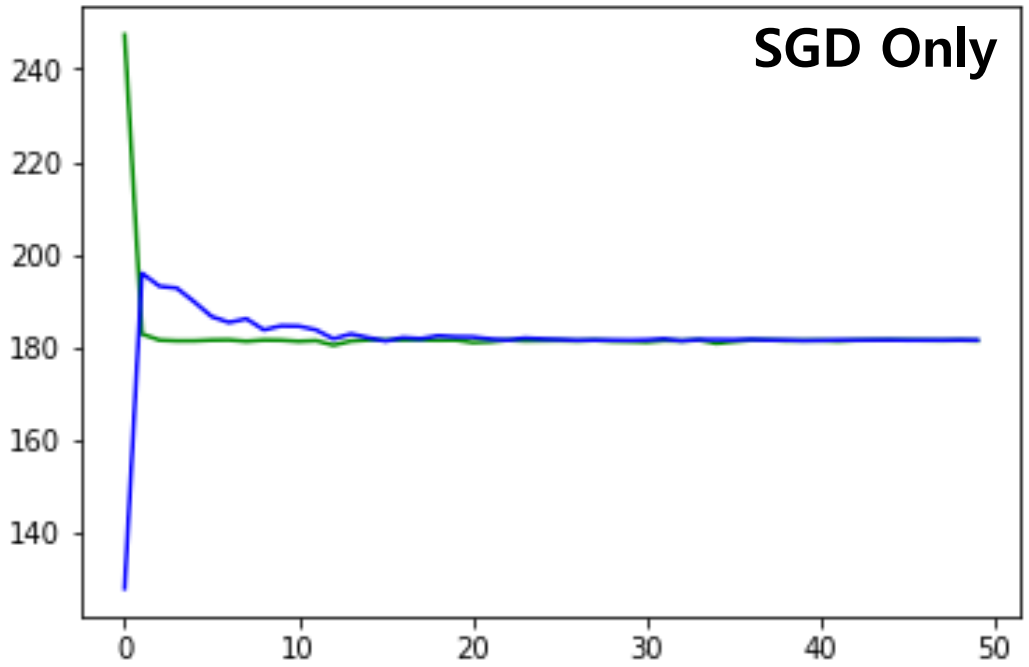
Convergence process

```
gd_lr = linear_model.LinearRegressionGD(eta0=0.001, epochs=10000, batch_size=1, shuffle=False)
bgd_lr = linear_model.LinearRegressionGD(eta0=0.001, epochs=10000, batch_size=len(X), shuffle=False)
sgd_lr = linear_model.LinearRegressionGD(eta0=0.001, epochs=10000, batch_size=1, shuffle=True)
msgd_lr = linear_model.LinearRegressionGD(eta0=0.001, epochs=10000, batch_size=100, shuffle=True)
```

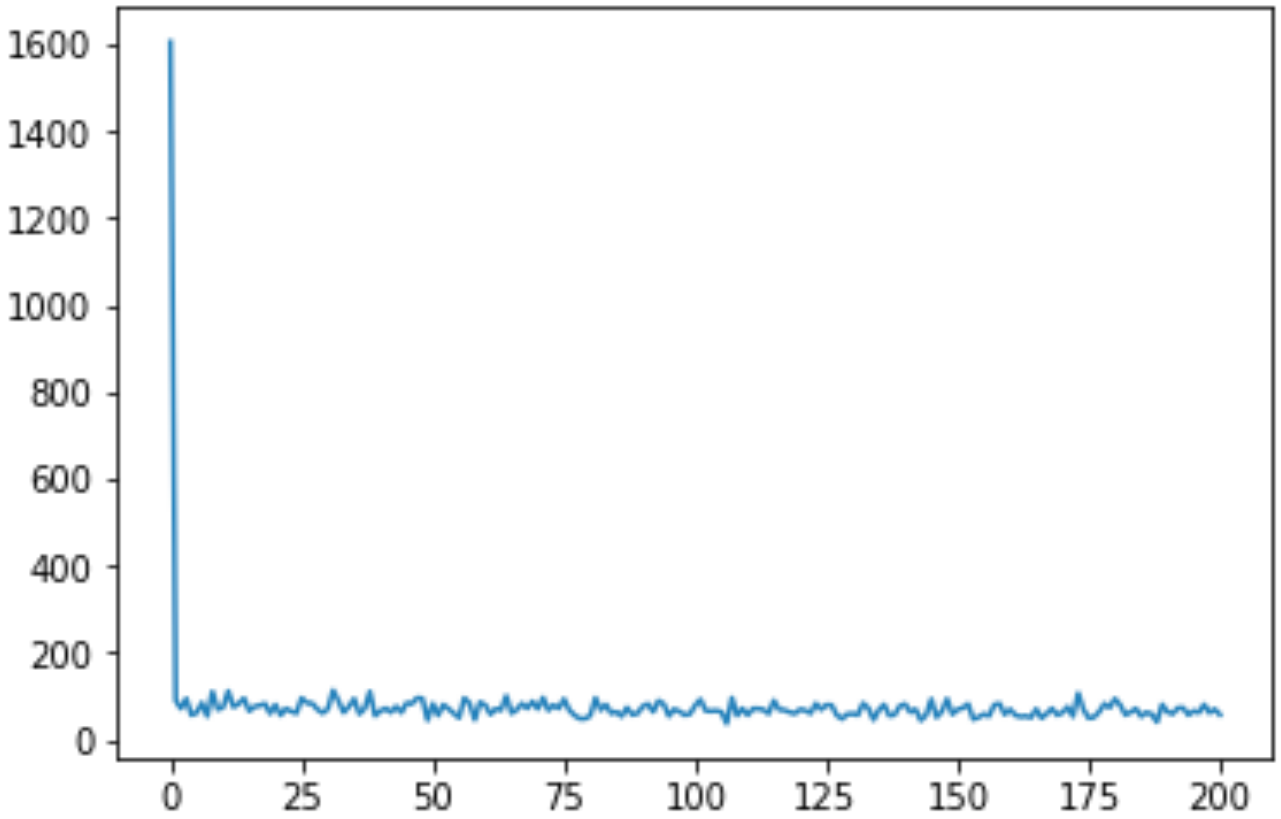
```
for epoch in range(epoches):
    X_copy = np.copy(X)
    if is_SGD:
        np.random.shuffle(X_copy)
    batch = len(X_copy) // BATCH_SIZE
    for batch_count in range(batch):
        X_batch = np.copy(
                X_copy[batch_count*BATCH_SIZE : (batch_count+1)*BATCH_SIZE])
        # Do weight Update
    print("Number of epoch : %d" % epoch)
```

Convergence process





Convergence process



Time-consuming

```
%timeit gd_lr.fit(X,y)
```

Gradient descent

```
1.37 s ± 18.3 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
%timeit bgd_lr.fit(X,y)
```

Full-batch Gradient descent

```
3.74 ms ± 121 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

```
%timeit sgd_lr.fit(X,y)
```

Stochstic Gradient descent

```
1.47 s ± 14.5 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

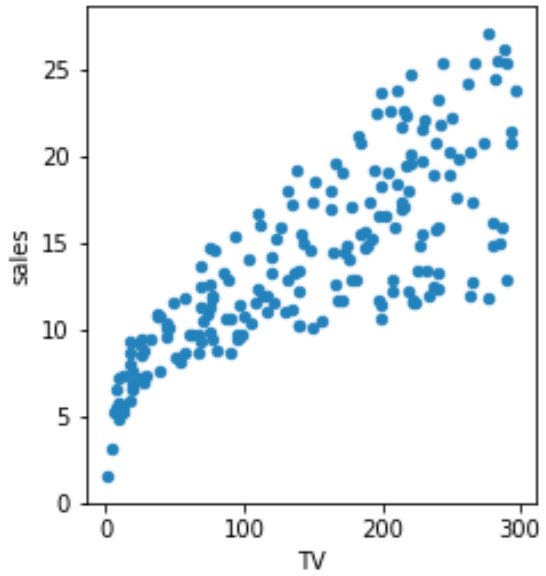
```
%timeit msgd_lr.fit(X,y)
```

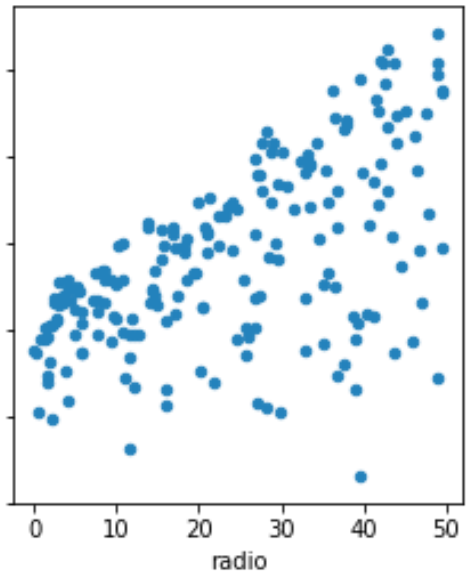
Minibatch-SGD

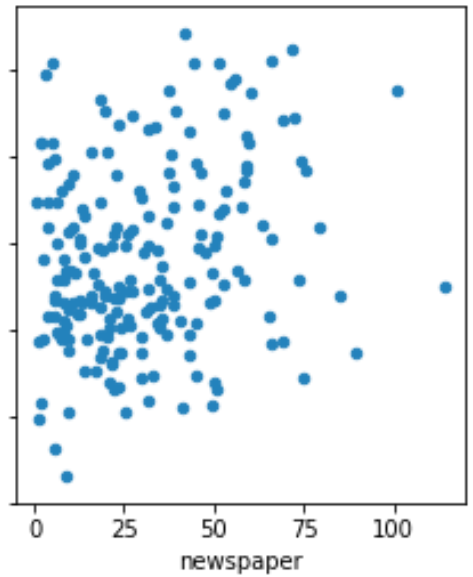
```
122 ms ± 1.07 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Multivariate

	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9



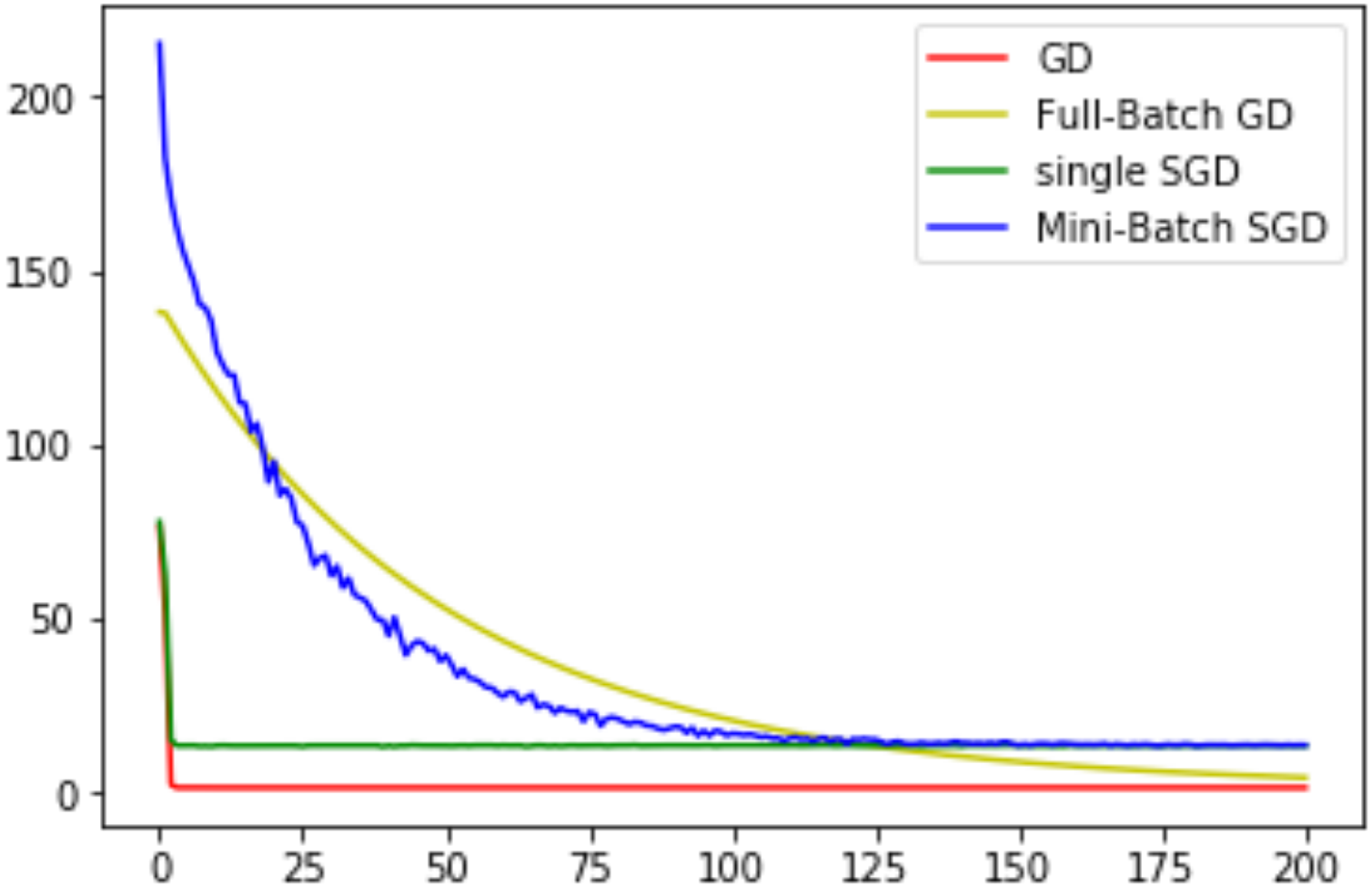




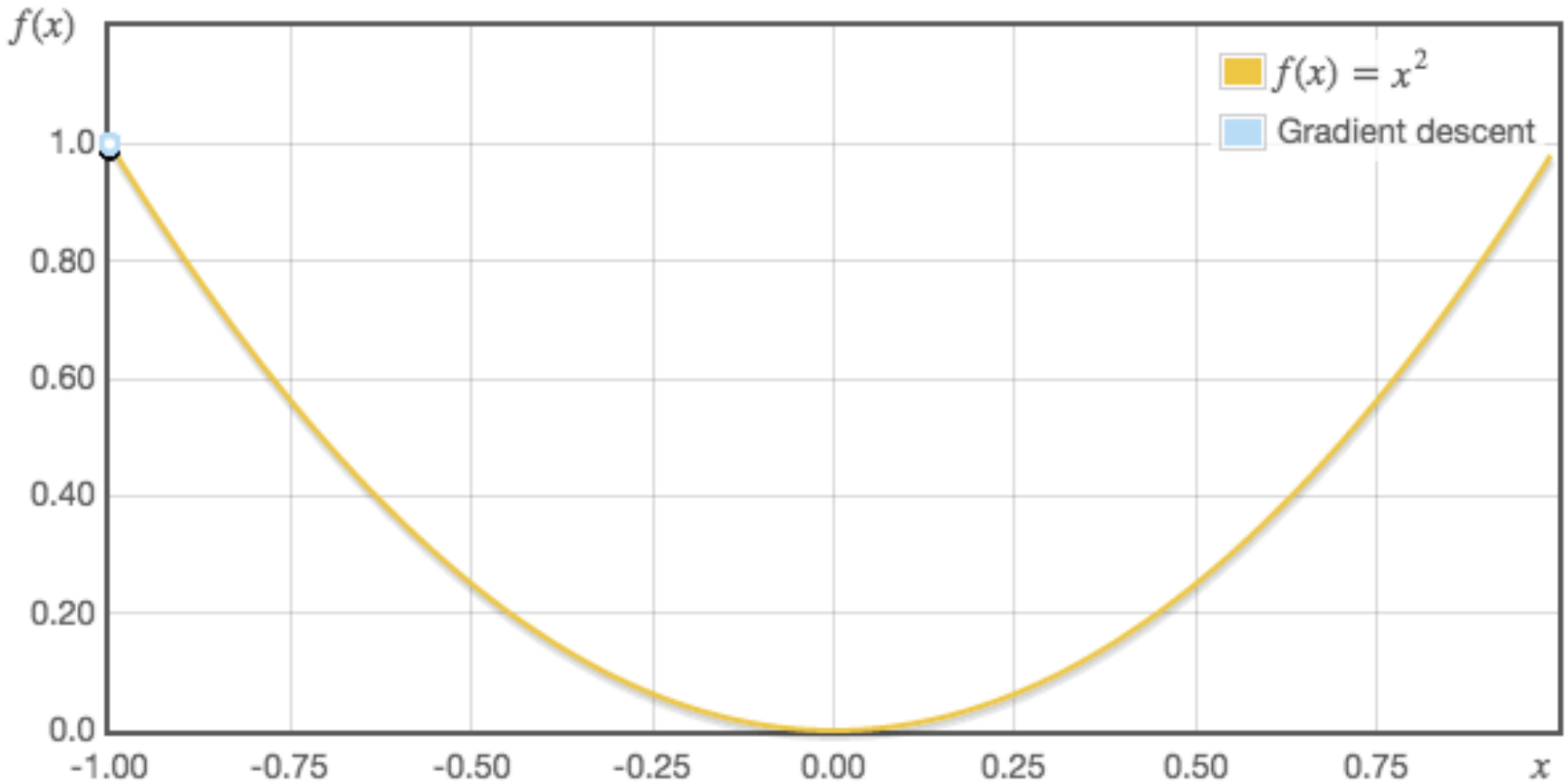

```
from sklearn import preprocessing
X_scaled = preprocessing.scale(ad_cost)
y = sales
X_scaled [:5]
```

```
array([[ 0.96985227,  0.98152247,  1.77894547],
       [-1.19737623,  1.08280781,  0.66957876],
       [-1.51615499,  1.52846331,  1.78354865],
       [ 0.05204968,  1.21785493,  1.28640506],
       [ 0.3941822 , -0.84161366,  1.28180188]])
```

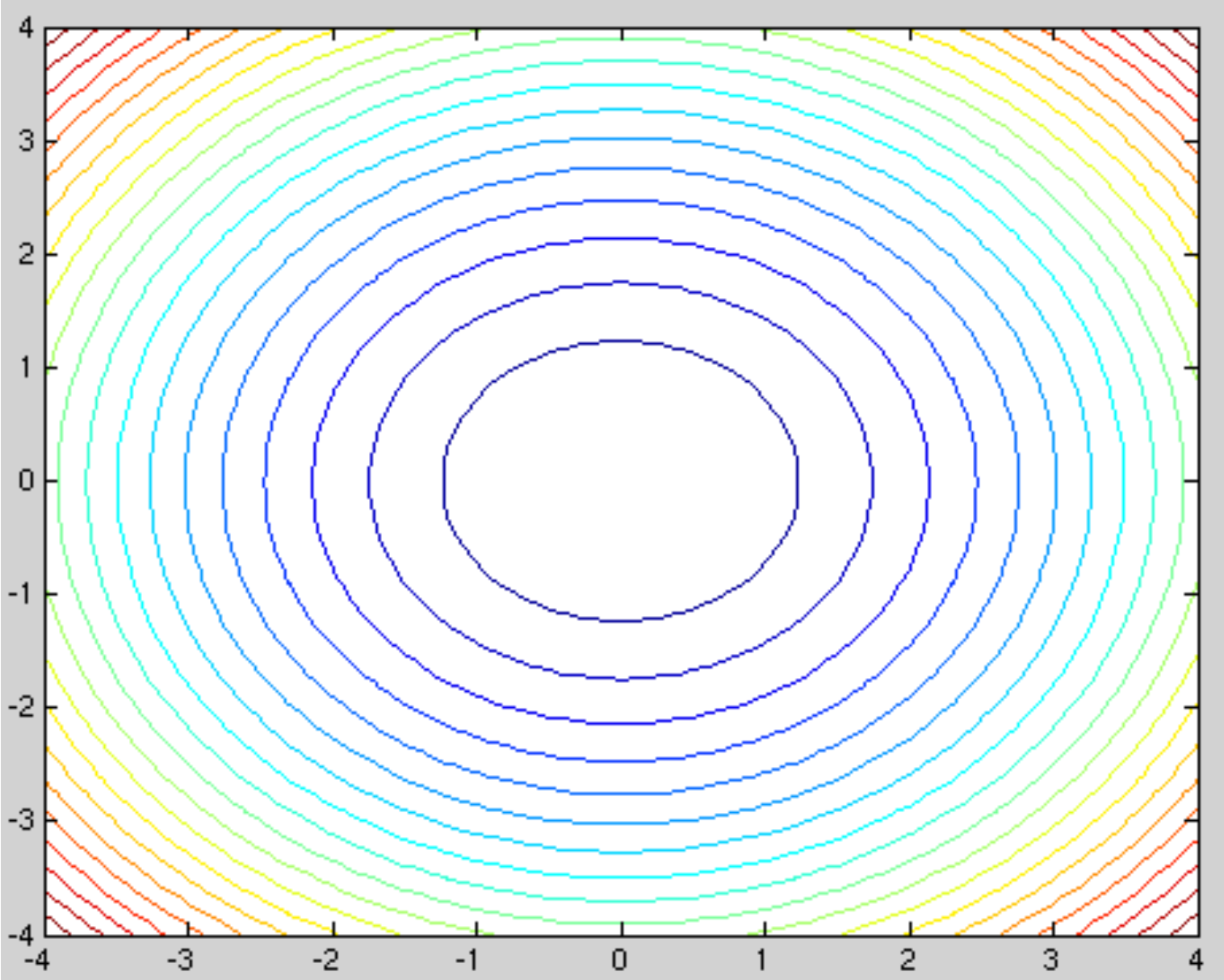
Multivariate



Learning rate는 일정해야 하는가?


$f(x)$
1.0
0.80
0.60
0.40
0.20
0.0
-1.00
-0.75
-0.50
-0.25
0.00
0.25
0.50
0.75
x
$f(x) = x^2$
Gradient descent

contour graph



Learning-rate decay

- 일정한 주기로 Learning rate을 감소시키는 방법
- 특정 epoch 마다 Learning rate를 감소

```
self._eta0 = self._eta0 * self._learning_rate_decay
```

- Hyper-parameter 설정의 어려움
- 지수감소 $\alpha = \alpha_0 e^{-kt}$, 1/t 감소 $\alpha = \frac{\alpha_0}{(1+kt)}$

종료조건 설정

- SGD과정에서 특정 값이하로 cost function이 줄어들지 않을 경우 GD를 멈추는 방법
- 성능이 좋아지지 않는/필요없는 연산을 방지함
- 종료조건을 설정 - tol > loss - previous_loss
- tol은 hyperparameter로 사람 설정함

Human knowledge belongs to the world.