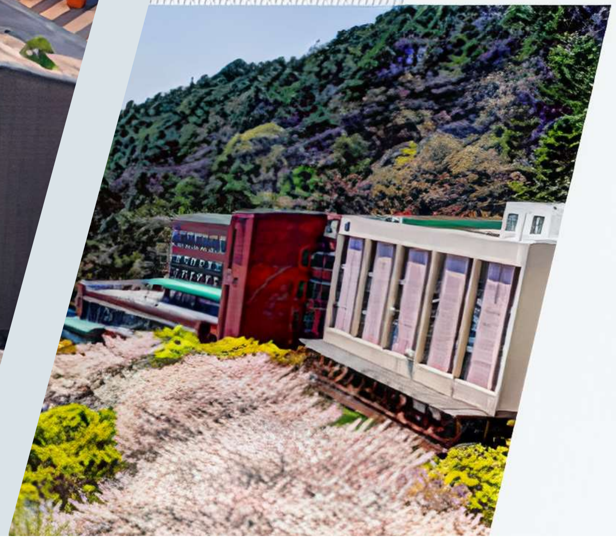
Dong-A Univ. (ISPL)

동아대학교
DONG-A UNIVERSITY

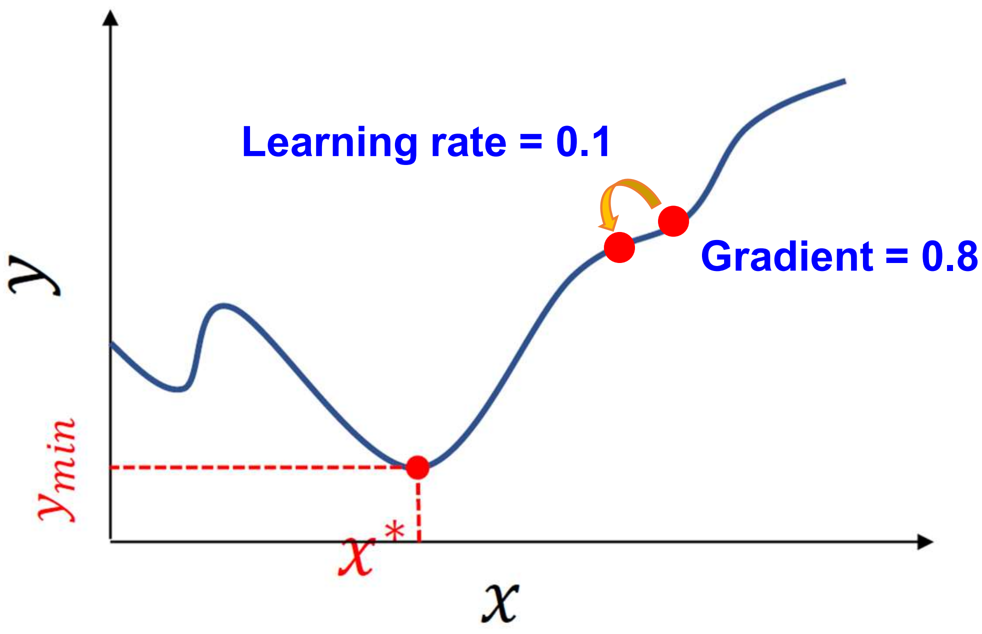
06주차 Learning Rate Control and Optimizers



컴퓨터AI공학부 AI학과
2023년 1학기 인공지능

Learning Rate Control

- **Review: Gradient decent algorithm**



❖ X (W, b): Trainable parameters
❖ Y: Loss function

Gradient decent algorithm

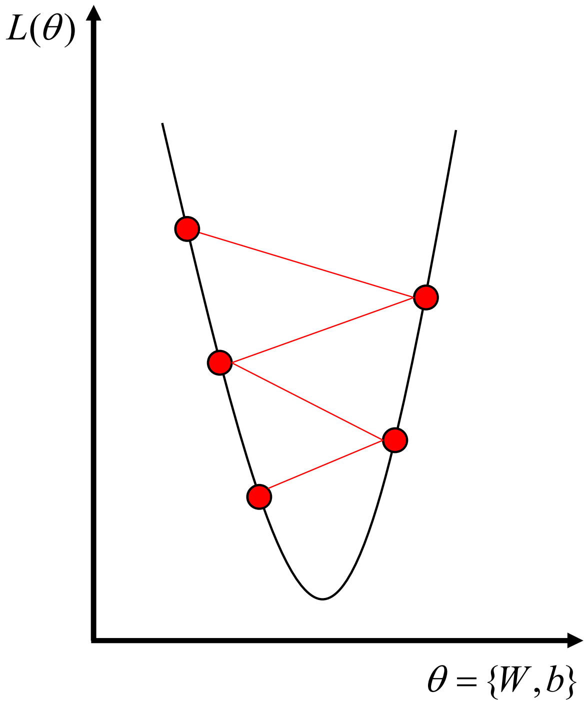
① 현재 지점에서 미분을 이용해 gradient 계산

② Gradient에 learning rate를 곱하고
반대 방향으로 weight update

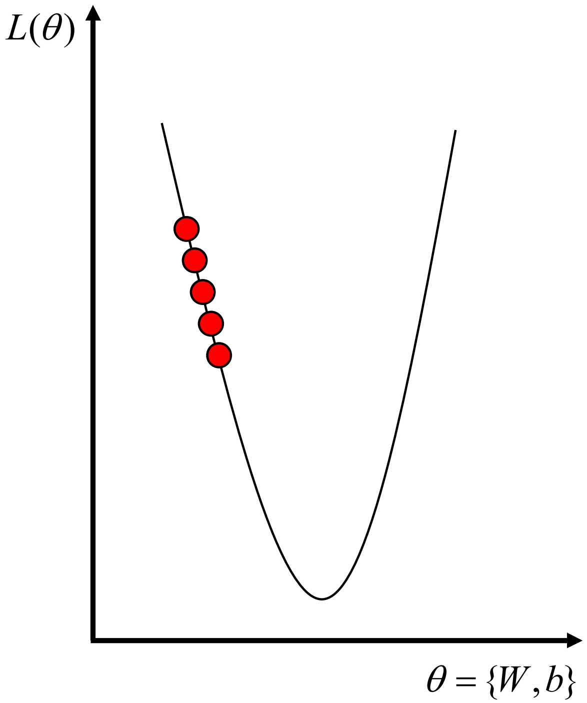
$$W_{new} = W_{prev} - \underbrace{\boxed{\alpha}}_{\text{Learning rate}} \underbrace{\boxed{\frac{\partial L}{\partial W}}}_{\text{Gradient}}$$

Learning Rate Control

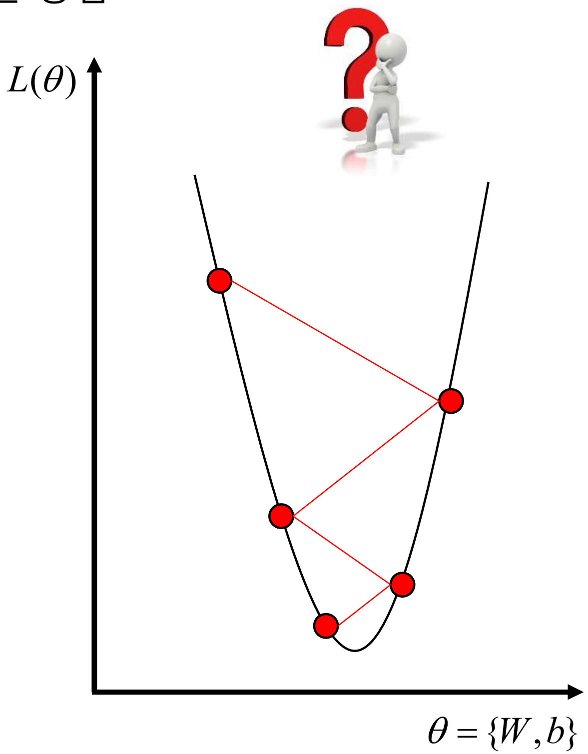
- **Learning rate control (decay)**
  - 빠른 시간내에 정확도 높은 학습 파라미터를 구하기 위해 learning rate를 조절하는 방법



Learning rate가 큰 경우



Learning rate가 작은 경우



동아대학교 DONG-A UNIVERSITY

Learning Rate Control

- 이전 실습까지 **고정된 learning rate를 사용**해 학습 수행 (LR: 0.1)

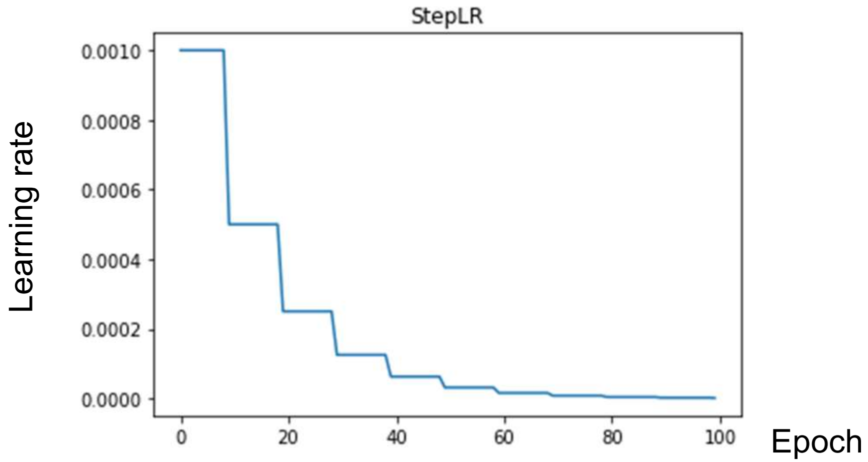
```
[ ]  batch_size = 10
     learning_rate = 0.1
     training_epochs = 100
     loss_function = nn.CrossEntropyLoss()
     network = FMLP()
     optimizer = torch.optim.SGD(network.parameters(), lr = learning_rate)

     data_loader = DataLoader(dataset = mnist_train,
                              batch_size = batch_size,
                              shuffle = True,
                              drop_last = True)
```

5주차 overfitting 실습 자료 중 일부 (Hyper-parameter 지정)

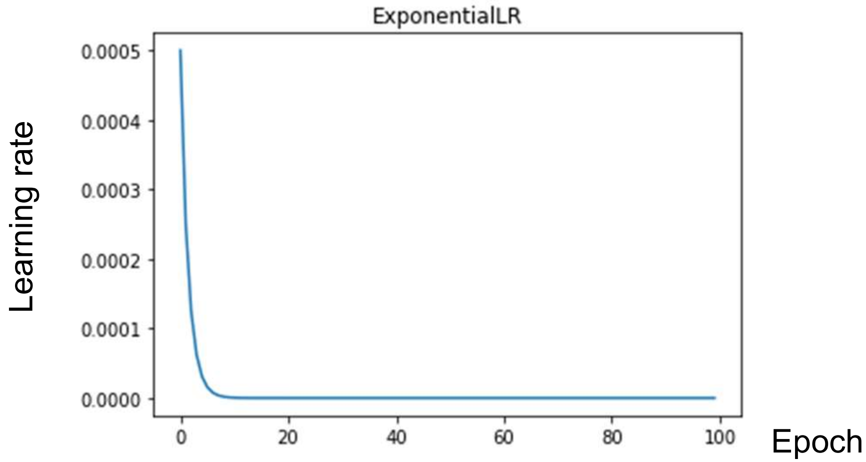
Learning Rate Control

- **Pytorch에서 제공하는 learning rate control 기능**
  - StepLR: 지정된 step (epoch) 단위마다 learning rate 조절
  - ExponentialLR: 매 step (epoch) 단위마다 learning rate 조절
  - etc…



$$lr_{\text{epoch}} = \begin{cases} Gamma * lr_{\text{epoch - 1}}, & \text{if epoch \% step\_size} = 0 \\ lr_{\text{epoch - 1}}, & \text{otherwise} \end{cases}$$

StepLR
(step_size: 10, gamma: 0.5)

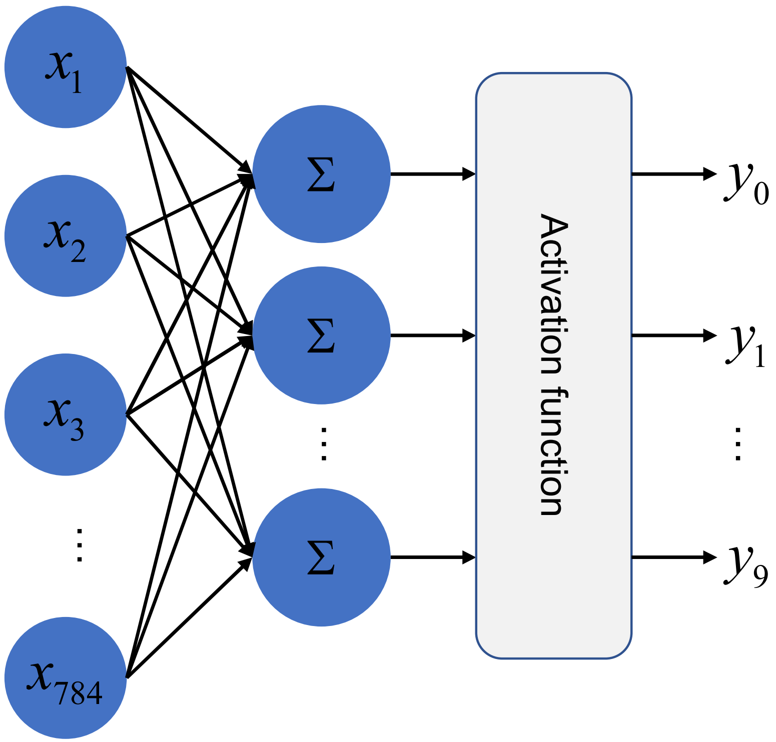


$$lr_{\text{epoch}} = Gamma * lr_{\text{epoch - 1}}$$

ExponentialLR
(step_size: 10, gamma: 0.5)

Learning Rate Control

- 금일은 Single Layer Perceptron (SLP)을 이용해 실습 진행

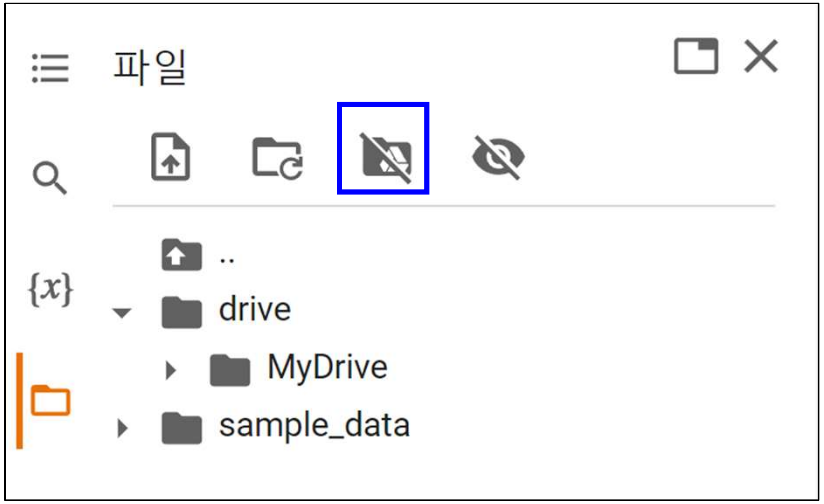



```python
class SLP(nn.Module):
    def __init__(self):
        super(SLP, self).__init__()
        self.fc1 = nn.Linear(in_features=784, out_features=100)

    def forward(self, x):
        x = x.view(-1, 28*28)
        y = self.fc1(x)
        return y
```

Learning Rate Control

- **[실습 1] Learning rate control을 수행하지 않는 SLP 학습**
  1) **6주차 LMS에 업로드 된 기본 소스코드 다운로드**
  2) 구글 드라이브 마운트 (파라미터 및 데이터셋 경로 확인 필요)
  3) 전체 셀 실행



동아대학교 DONG-A UNIVERSITY

Learning Rate Control

- **[실습 1] Learning rate control을 수행하지 않는 SLP 학습**
  1) 6주차 LMS에 업로드 된 기본 소스코드 다운로드
  2) 구글 드라이브 마운트 (파라미터 및 데이터셋 경로 확인 필요)
  3) 전체 셀 실행
  4) **결과 확인**

```
Epoch: 1, LR: 0.100000, Loss: 0.537231
Epoch: 2, LR: 0.100000, Loss: 0.359334
Epoch: 3, LR: 0.100000, Loss: 0.331079
Epoch: 4, LR: 0.100000, Loss: 0.316594
Epoch: 5, LR: 0.100000, Loss: 0.307029
Epoch: 6, LR: 0.100000, Loss: 0.300397
Epoch: 7, LR: 0.100000, Loss: 0.294995
Epoch: 8, LR: 0.100000, Loss: 0.290815
Epoch: 9, LR: 0.100000, Loss: 0.287408
Epoch: 10, LR: 0.100000, Loss: 0.284473
Epoch: 11, LR: 0.100000, Loss: 0.281891
Epoch: 12, LR: 0.100000, Loss: 0.279587
Epoch: 13, LR: 0.100000, Loss: 0.277830
Epoch: 14, LR: 0.100000, Loss: 0.275999
Epoch: 15, LR: 0.100000, Loss: 0.274663
Learning finished
```

```
Accuracy: 0.8845999836921692
```

Learning Rate Control

- **[실습 2] StepLR 기법을 이용한 SLP 학습**
  - [5] Hyper-parameter 및 [6] Training loop 일부 수정

[5] Hyper-parameter 지정

- 이미 학습을 수행한 상태로 다시 학습을 수행할 때 이 셀을 재실행해야함
- Step [4] 모델 구조 선언에 사용한 class 명과 일치한지 반드시 확인

```
[19] batch_size = 100
     learning_rate = 0.1
     training_epochs = 15
     loss_function = nn.CrossEntropyLoss()
     network = SLP()
     optimizer = torch.optim.SGD(network.parameters(), lr = learning_rate)

     scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=5, gamma=0.5)

     data_loader = DataLoader(dataset = mnist_train,
                              batch_size = batch_size,
                              shuffle = True,
                              drop_last = True)
```

[6] Training loop

```
[20] for epoch in range(training_epochs):
         avg_cost = 0
         total_batch = len(data_loader)

         for img, label in data_loader:
             pred = network(img)

             loss = loss_function(pred, label)
             optimizer.zero_grad()
             loss.backward()
             optimizer.step()
             avg_cost += loss / total_batch

         print('Epoch: %d, LR: %f,  Loss: %f' %(epoch+1, optimizer.param_groups[0]['lr'], avg_cost))
         scheduler.step()

     print('Learning finished')
```

StepLR → 매 5epoch 마다 learning rate에 0.5 곱하기

동아대학교 DONG-A UNIVERSITY

Learning Rate Control

- **[실습 2] StepLR 기법을 이용한 SLP 학습**
  - 결과 확인

```
Epoch: 1, LR: 0.100000, Loss: 0.535101
Epoch: 2, LR: 0.100000, Loss: 0.359083
Epoch: 3, LR: 0.100000, Loss: 0.330872
Epoch: 4, LR: 0.100000, Loss: 0.316319
Epoch: 5, LR: 0.100000, Loss: 0.306708
Epoch: 6, LR: 0.050000, Loss: 0.299735
Epoch: 7, LR: 0.050000, Loss: 0.296768
Epoch: 8, LR: 0.050000, Loss: 0.294234
Epoch: 9, LR: 0.050000, Loss: 0.291917
Epoch: 10, LR: 0.050000, Loss: 0.289835
Epoch: 11, LR: 0.025000, Loss: 0.287472
Epoch: 12, LR: 0.025000, Loss: 0.286501
Epoch: 13, LR: 0.025000, Loss: 0.285718
Epoch: 14, LR: 0.025000, Loss: 0.284972
Epoch: 15, LR: 0.025000, Loss: 0.284091
Learning finished
```

❖ 고정된 learning rate 사용 시

```
Accuracy: 0.8845999836921692
```

1.3% 성능 향상

❖ Learning rate control 사용 시

```
Accuracy: 0.8975999951362610
```

Learning Rate Control

- **[실습 3] ExponentialLR 기법을 이용한 SLP 학습**
  - [5] Hyper-parameter 일부 수정

[5] Hyper-parameter 지정

- 이미 학습을 수행한 상태로 다시 학습을 수행할 때 이 셀을 재실행해야함
- Step [4] 모델 구조 선언에 사용한 class 명과 일치한지 반드시 확인

```
[28] batch_size = 100
     learning_rate = 0.1
     training_epochs = 15
     loss_function = nn.CrossEntropyLoss()
     network = SLP()
     optimizer = torch.optim.SGD(network.parameters(), lr = learning_rate)

     scheduler = torch.optim.lr_scheduler.ExponentialLR(optimizer, gamma=0.8)

     data_loader = DataLoader(dataset = mnist_train,
                              batch_size = batch_size,
                              shuffle = True,
                              drop_last = True)
```

[6] Training loop

```
[20] for epoch in range(training_epochs):
         avg_cost = 0
         total_batch = len(data_loader)

         for img, label in data_loader:
             pred = network(img)

             loss = loss_function(pred, label)
             optimizer.zero_grad()
             loss.backward()
             optimizer.step()
             avg_cost += loss / total_batch

         print('Epoch: %d, LR: %f,  Loss: %f' %(epoch+1, optimizer.param_groups[0]['lr'], avg_cost))
         scheduler.step()

     print('Learning finished')
```

Learning Rate Control

- **[실습 3] ExponentialLR 기법을 이용한 SLP 학습**
  - 결과확인

```
Epoch: 1, LR: 0.100000, Loss: 0.536337
Epoch: 2, LR: 0.080000, Loss: 0.361799
Epoch: 3, LR: 0.064000, Loss: 0.337062
Epoch: 4, LR: 0.051200, Loss: 0.325141
Epoch: 5, LR: 0.040960, Loss: 0.317976
Epoch: 6, LR: 0.032768, Loss: 0.313117
Epoch: 7, LR: 0.026214, Loss: 0.309792
Epoch: 8, LR: 0.020972, Loss: 0.307415
Epoch: 9, LR: 0.016777, Loss: 0.305586
Epoch: 10, LR: 0.013422, Loss: 0.304185
Epoch: 11, LR: 0.010737, Loss: 0.303148
Epoch: 12, LR: 0.008590, Loss: 0.302284
Epoch: 13, LR: 0.006872, Loss: 0.301671
Epoch: 14, LR: 0.005498, Loss: 0.301167
Epoch: 15, LR: 0.004398, Loss: 0.300759
Learning finished
```

❖ **고정된 learning rate 사용 (lr: 0.1)**

```
Accuracy: 0.8845999836921692
```

❖ **StepLR (step_size: 5, gamma: 0.5)**

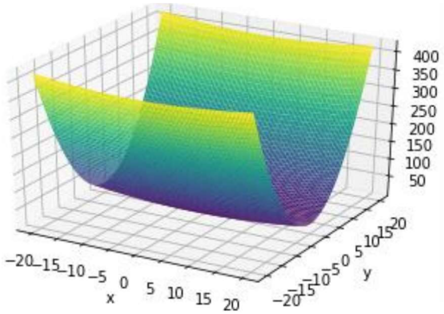
```
Accuracy: 0.897599995136261
```

❖ **ExponentialLR (gamma: 0.8)**

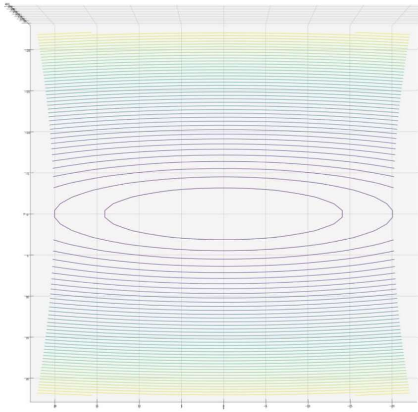
```
Accuracy: 0.9035999774932861
```

Optimizer

- **대표적인 optimizer 기법 비교**

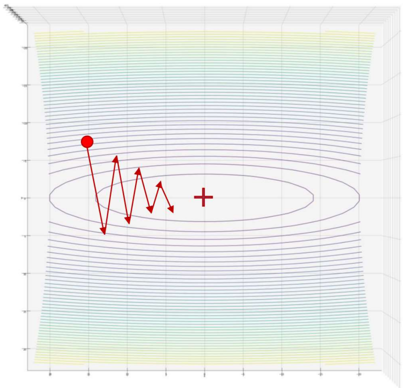


$$f(x,y) = \frac{1}{20}x^2 + y^2$$

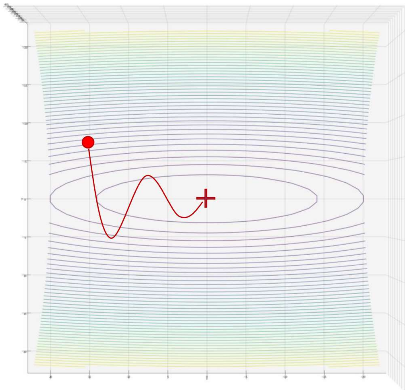


❖ f(x,y): loss function으로 가정

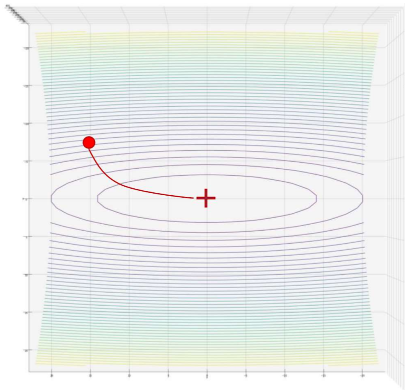
❖ x, y: trainable parameter로 가정



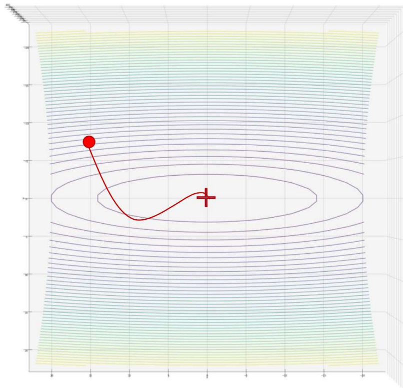
SGD



Momentum



AdaGrad



Adam

● : Starting point

+ : Optimal point

동아대학교 DONG-A UNIVERSITY

Optimizer

- **[실습 1] Momentum 기법을 이용한 SLP 학습**
  - [5] Hyper-parameter 및 [6] Training loop 부분 변경

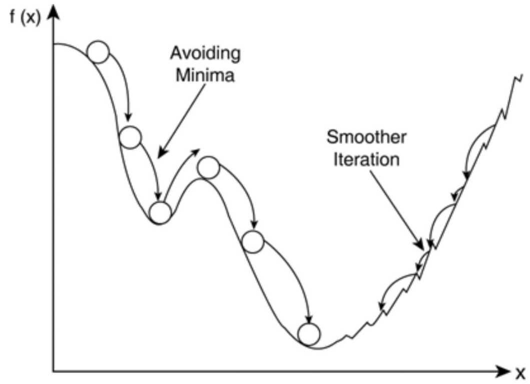
[5] Hyper-parameter 지정

- 이미 학습을 수행한 상태로 다시 학습을 수행할 때 이 셀을 재실행해야함
- Step [4] 모델 구조 선언에 사용한 class 명과 일치한지 반드시 확인

[57]
```
batch_size = 100
learning_rate = 0.1
training_epochs = 15
loss_function = nn.CrossEntropyLoss()
network = SLP()
optimizer = torch.optim.SGD(network.parameters(), lr = learning_rate, momentum = 0.9)

# scheduler = torch.optim.lr_scheduler.ExponentialLR(optimizer, gamma=0.8)

data_loader = DataLoader(dataset = mnist_train,
                         batch_size = batch_size,
                         shuffle = True,
                         drop_last = True)
```



$$W_{t+1} = W_t + v_t$$

$$v_t = \boxed{\alpha} v_{t-1} - \eta \frac{\partial L}{\partial W_t}$$

momentum 계수

동아대학교 DONG-A UNIVERSITY

Optimizer

- **[실습 1] Momentum 기법을 이용한 SLP 학습**
  - [5] Hyper-parameter 및 [6] Training loop 부분 변경

[6] Training loop

```python
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = len(data_loader)

    for img, label in data_loader:
        pred = network(img)

        loss = loss_function(pred, label)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        avg_cost += loss / total_batch

    print('Epoch: %d, LR: %f,  Loss: %f' %(epoch+1, optimizer.param_groups[0]['lr'], avg_cost))
    # scheduler.step()

print('Learning finished')
```

Optimizer

- **[실습 1] Momentum 기법을 이용한 SLP 학습**
  - 결과 확인

```
Epoch: 1, LR: 0.100000,  Loss: 0.382954
Epoch: 2, LR: 0.100000,  Loss: 0.300450
Epoch: 3, LR: 0.100000,  Loss: 0.290717
Epoch: 4, LR: 0.100000,  Loss: 0.285352
Epoch: 5, LR: 0.100000,  Loss: 0.279600
Epoch: 6, LR: 0.100000,  Loss: 0.277274
Epoch: 7, LR: 0.100000,  Loss: 0.273616
Epoch: 8, LR: 0.100000,  Loss: 0.271986
Epoch: 9, LR: 0.100000,  Loss: 0.271436
Epoch: 10, LR: 0.100000,  Loss: 0.270345
Epoch: 11, LR: 0.100000,  Loss: 0.268834
Epoch: 12, LR: 0.100000,  Loss: 0.266973
Epoch: 13, LR: 0.100000,  Loss: 0.264558
Epoch: 14, LR: 0.100000,  Loss: 0.264766
Epoch: 15, LR: 0.100000,  Loss: 0.264690
Learning finished
```

❖ **SGD**

```
Accuracy: 0.8845999836921692
```

❖ **SGD + Momentum**

```
Accuracy: 0.8580999970436096
```

Optimizer

- **[실습 2] Adam 기법을 이용한 SLP 학습**
  - [5] Hyper-parameter 변경

[5] Hyper-parameter 지정

- 이미 학습을 수행한 상태로 다시 학습을 수행할 때 이 셀을 재실행해야함
- Step [4] 모델 구조 선언에 사용한 class 명과 일치한지 반드시 확인

[57]
```
batch_size = 100
learning_rate = 0.1
training_epochs = 15
loss_function = nn.CrossEntropyLoss()
network = SLP()
optimizer = torch.optim.Adam(network.parameters(), lr = learning_rate)

# scheduler = torch.optim.lr_scheduler.ExponentialLR(optimizer, gamma=0.8)

data_loader = DataLoader(dataset = mnist_train,
                         batch_size = batch_size,
                         shuffle = True,
                         drop_last = True)
```

Optimizer

- **[실습 2] Adam 기법을 이용한 SLP 학습**
  - 결과확인

```
Epoch: 1, LR: 0.100000,  Loss: 0.895777
Epoch: 2, LR: 0.100000,  Loss: 0.939572
Epoch: 3, LR: 0.100000,  Loss: 0.984272
Epoch: 4, LR: 0.100000,  Loss: 1.049648
Epoch: 5, LR: 0.100000,  Loss: 1.028599
Epoch: 6, LR: 0.100000,  Loss: 1.010980
Epoch: 7, LR: 0.100000,  Loss: 1.042650
Epoch: 8, LR: 0.100000,  Loss: 1.008793
Epoch: 9, LR: 0.100000,  Loss: 1.028753
Epoch: 10, LR: 0.100000,  Loss: 1.043825
Epoch: 11, LR: 0.100000,  Loss: 0.993260
Epoch: 12, LR: 0.100000,  Loss: 1.024413
Epoch: 13, LR: 0.100000,  Loss: 1.077637
Epoch: 14, LR: 0.100000,  Loss: 1.024739
Epoch: 15, LR: 0.100000,  Loss: 1.040391
Learning finished
```

❖ **SGD**

```
Accuracy: 0.8845999836921692
```

❖ **SGD + Momentum**

```
Accuracy: 0.8580999970436096
```

❖ **Adam**

```
Accuracy: 0.8443999886
```

Questions & Answers

Dongsan Jun (dsjun@dau.ac.kr)
Image Signal Processing Laboratory (www.donga-ispl.kr)
Division of Computer･AI Engineering
Dong-A University, Busan, Rep. of Korea