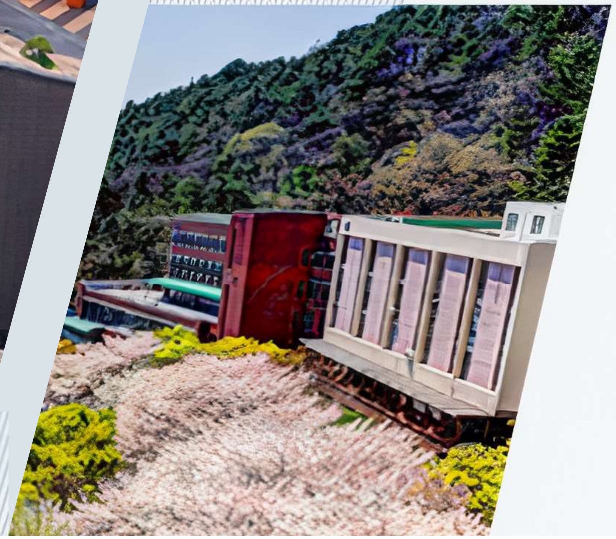
Dong-A Univ. (ISPL)

동아대학교
DONG-A UNIVERSITY

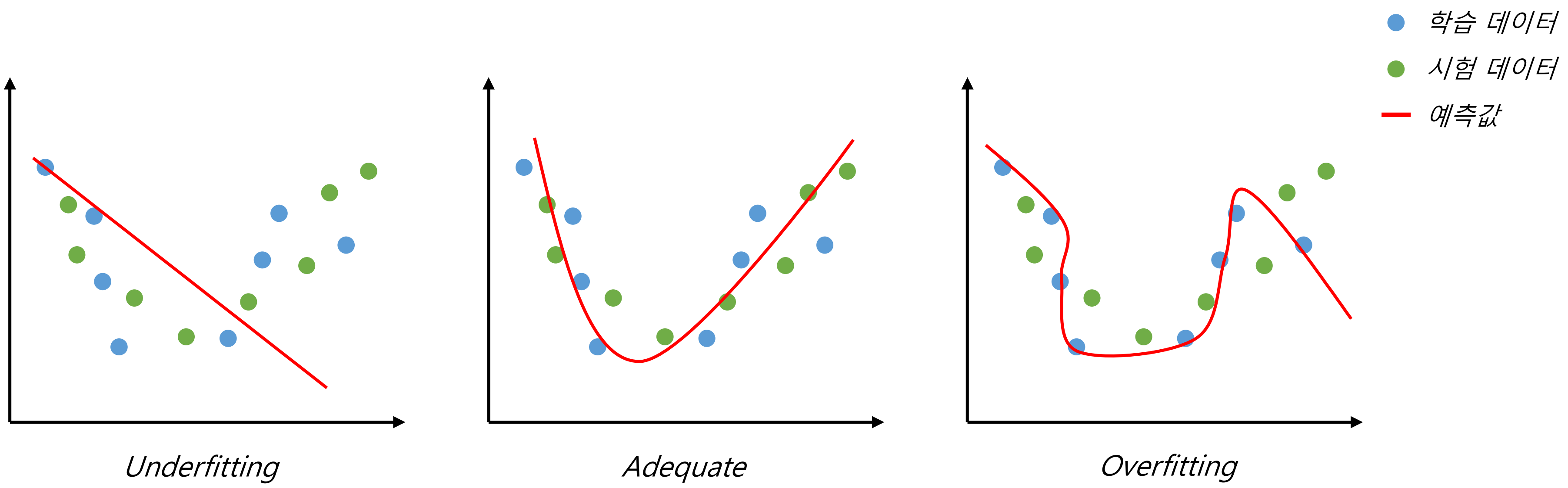
Backpropagation (역전파)
– 실습: Overfitting



컴퓨터AI공학부
2024년 1학기 인공지능

Overfitting

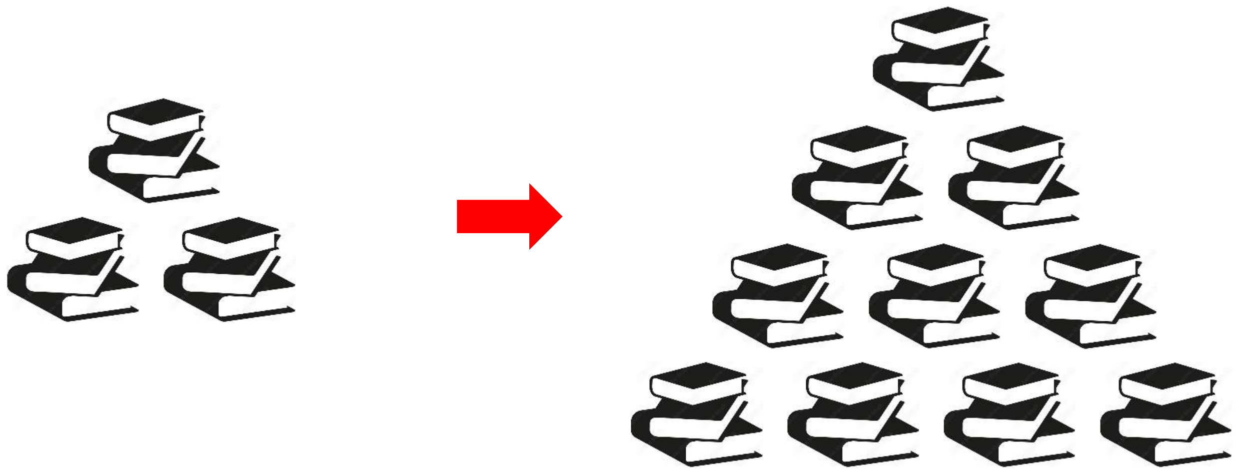
- **오버피팅(Overfitting): 학습데이터를 과하게 학습하여 그 외의 데이터에는 대응하지 못하는 상태**
- **오버피팅이 주로 일어나는 경우**
  - 매개변수가 많은 모델
  - 학습데이터가 적음



동아대학교 DONG-A UNIVERSITY

Overfitting

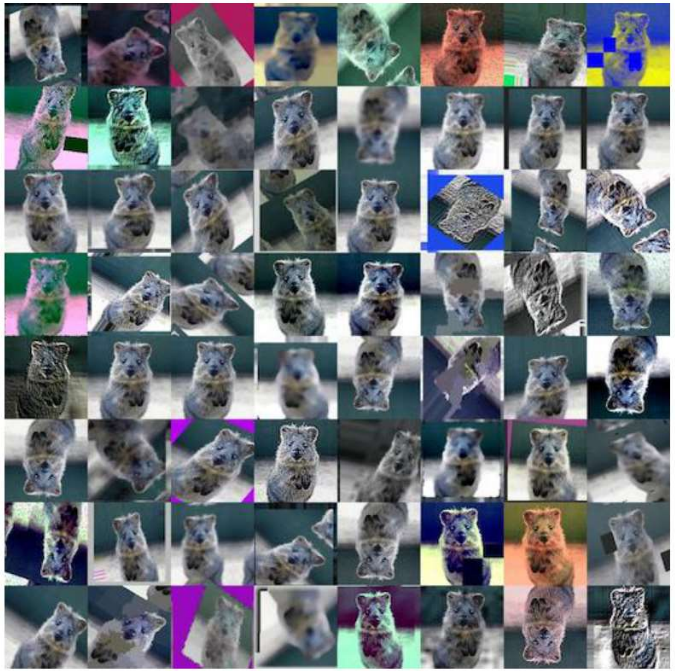
- **해결방법 1** → **데이터 확보**



동아대학교 DONG-A UNIVERSITY

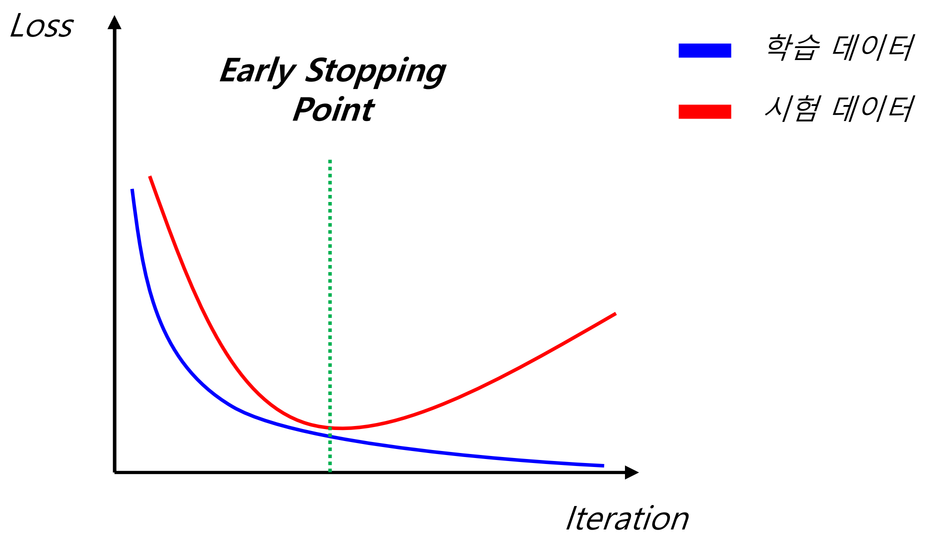
Overfitting

- **해결방법 ➔ 데이터 증식(Data Augmentation)**
  - 입력 이미지(학습 이미지)를 '인위적'으로 확장
  - 회전(Rotation), 이동(Move), 자르기(Crop), 대칭(Symmetry) 등



Overfitting

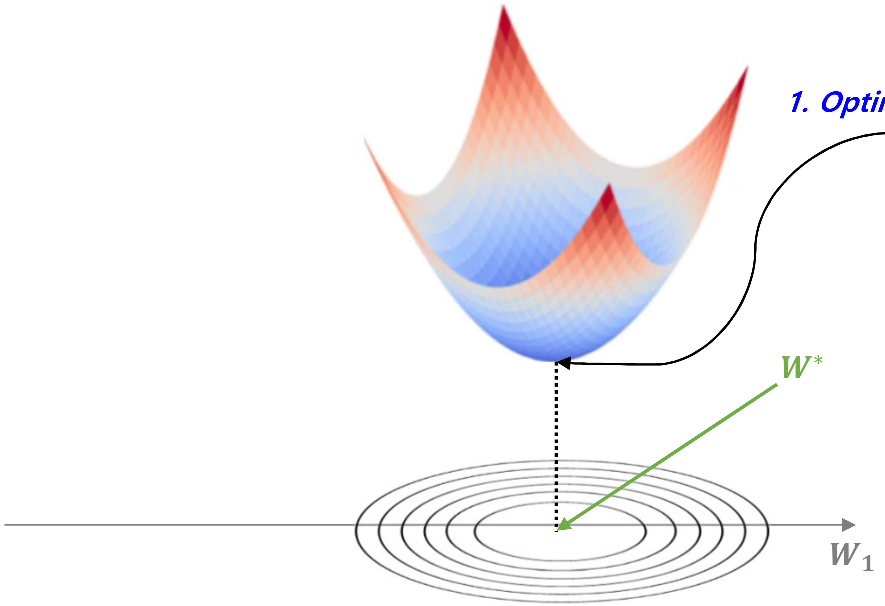
- **해결방법 2 ➔ 조기 종료(Early Stopping)**
  - Epoch, Iteration을 많이 돌린 후, 특정 시점에서 멈추는 것



Overfitting

- **해결방법 3 ➔ L1, L2 정규화(L1, L2 Regularization)**

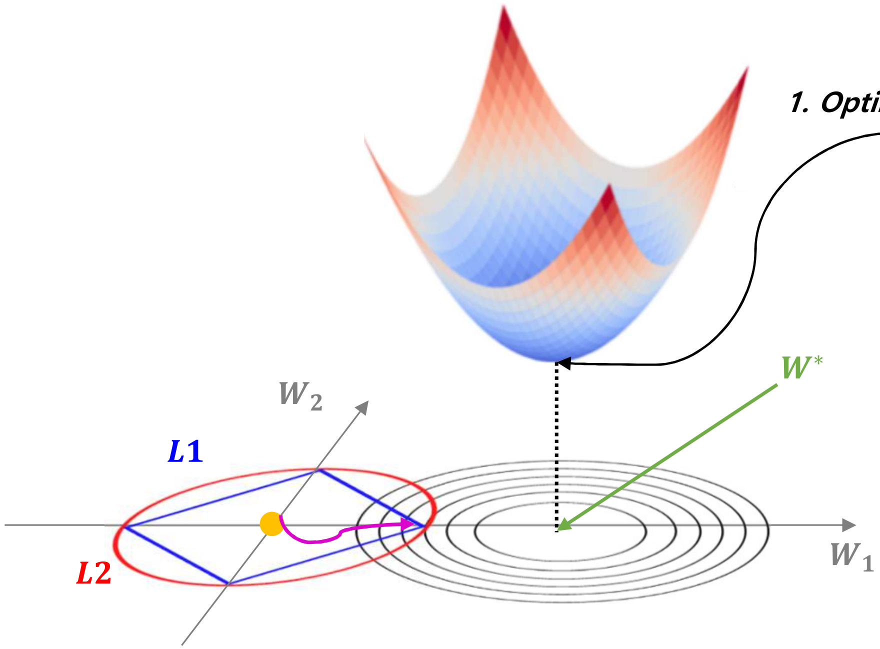
1. Optimization을 통해 Training dataset에 최적화된 상태



동아대학교 DONG-A UNIVERSITY

Overfitting

- **해결방법 ➔ L1, L2 정규화(L1, L2 Regularization)**

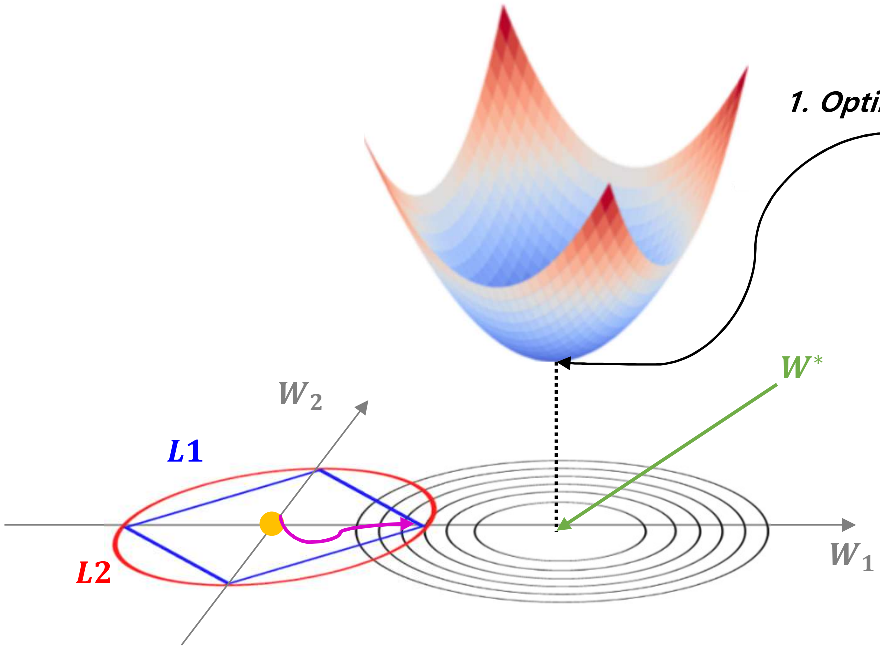


1. Optimization을 통해 Training dataset에 최적화된 상태

2. Loss Function을 이용하여 Optimization을 못하게 설정 ➔ Loss Function Boundary 추가

Overfitting

- **해결방법 ➔ L1, L2 정규화(L1, L2 Regularization)**



1. Optimization을 통해 Training dataset에 최적화된 상태

2. Loss Function을 이용하여 Optimization을 못하게 설정 ➔ Loss Function Boundary 추가

3. Loss Function뒤에 정규화 Term(Loss Function Boundary)을 추가

$$\tilde{L} = L(y, \hat{y}) + \boxed{\lambda\Omega(w)}, \lambda \geq 0$$

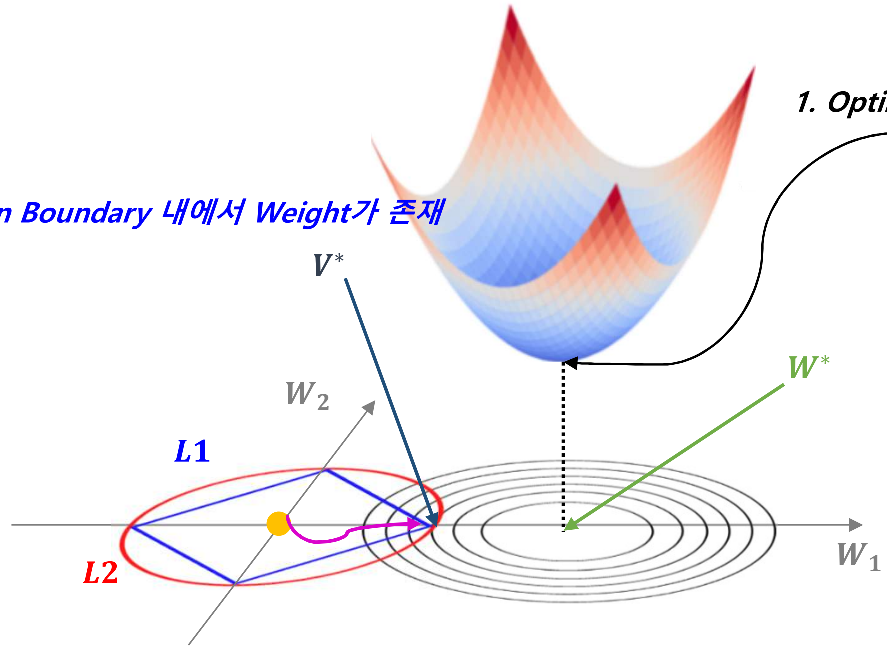
$$L1:\ \Omega(w) = \sum_{i=1}^{n} |w_i|$$

$$L2:\ \Omega(w) = \sum_{i=1}^{n} w_i^2$$

동아대학교 DONG-A UNIVERSITY

Overfitting

- **해결방법 ➔ L1, L2 정규화(L1, L2 Regularization)**



1. Optimization을 통해 Training dataset에 최적화된 상태

2. Loss Function을 이용하여 Optimization을 못하게 설정 ➔ Loss Function Boundary 추가

3. Loss Function뒤에 정규화 Term(Loss Function Boundary)을 추가

$$\tilde{L} = L(y, \hat{y}) + \boxed{\lambda\Omega(w)}, \lambda \geq 0$$

4. Loss Function Boundary 내에서 Weight가 존재

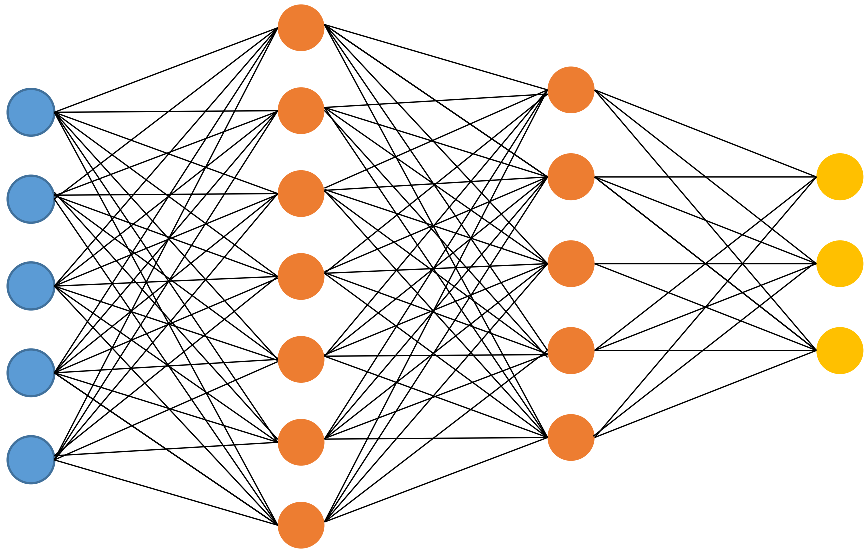
$$L1:\ \Omega(w) = \sum_{i=1}^{n} |w_i|$$

$$L2:\ \Omega(w) = \sum_{i=1}^{n} w_i^2$$

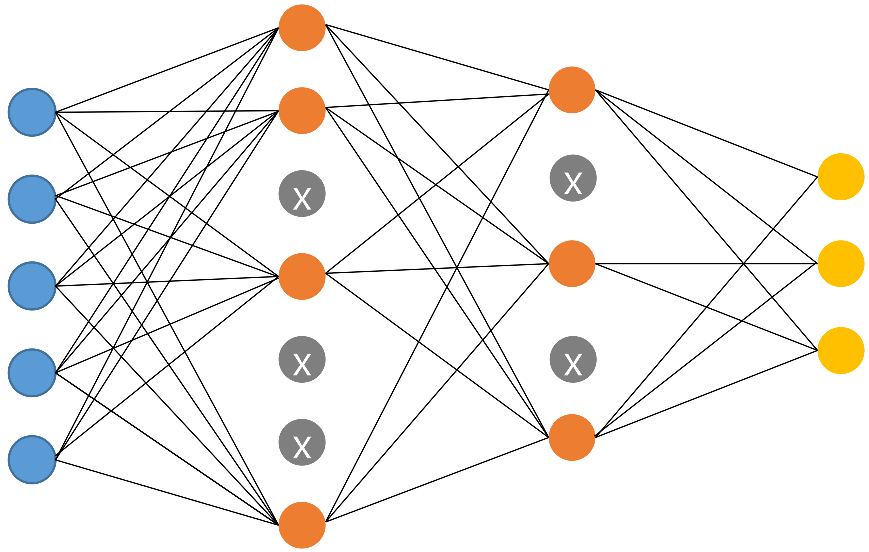
동아대학교 DONG-A UNIVERSITY

Overfitting

- **해결방법 4 ➔ 드롭아웃(Dropout)**
  - Backpropagation시 모든 Weight가 업데이트 되는 것을 방지
  - 일부 Node를 랜덤하게 제거



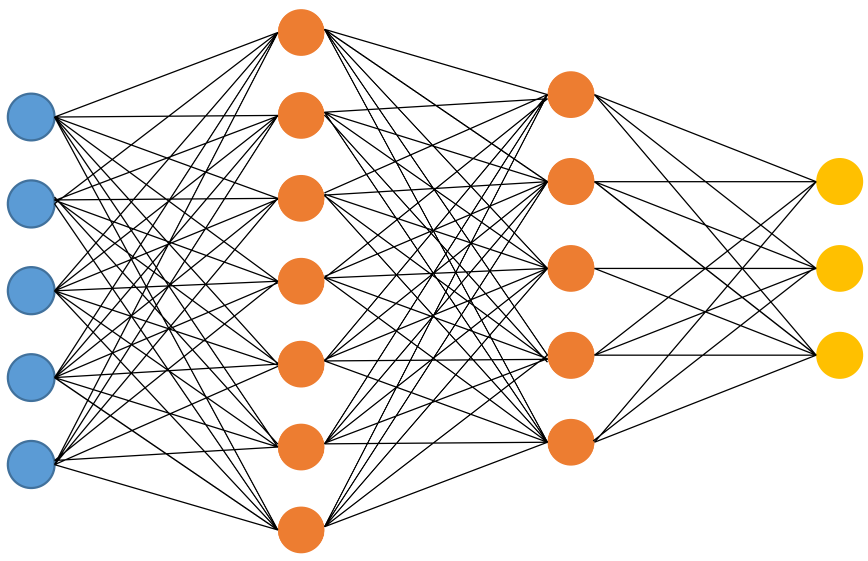
Dropout 적용 전



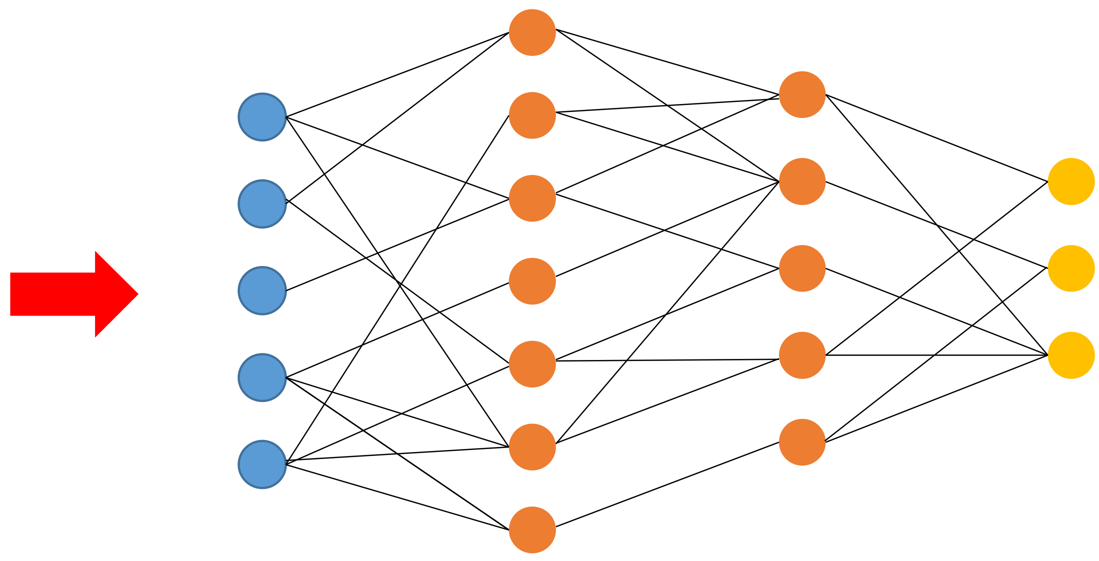
Dropout 적용 후

동아대학교 DONG-A UNIVERSITY

- **해결방법 5 ➔ 드롭 커넥트(Drop Connect)**
  - Backpropagation시 모든 Weight가 업데이트 되는 것을 방지
  - 일부 Weight를 랜덤하게 제거



Drop Connect 적용 전



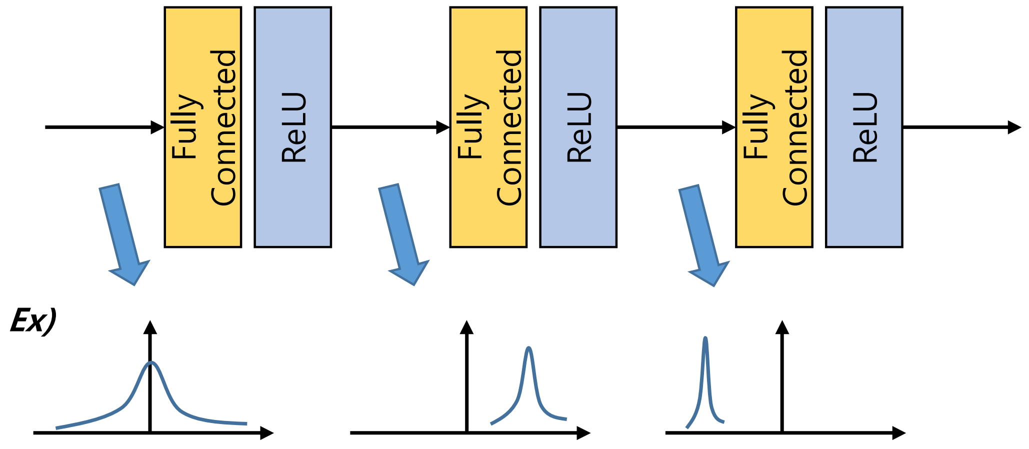
Drop Connect 적용 후

동아대학교 DONG-A UNIVERSITY

Overfitting

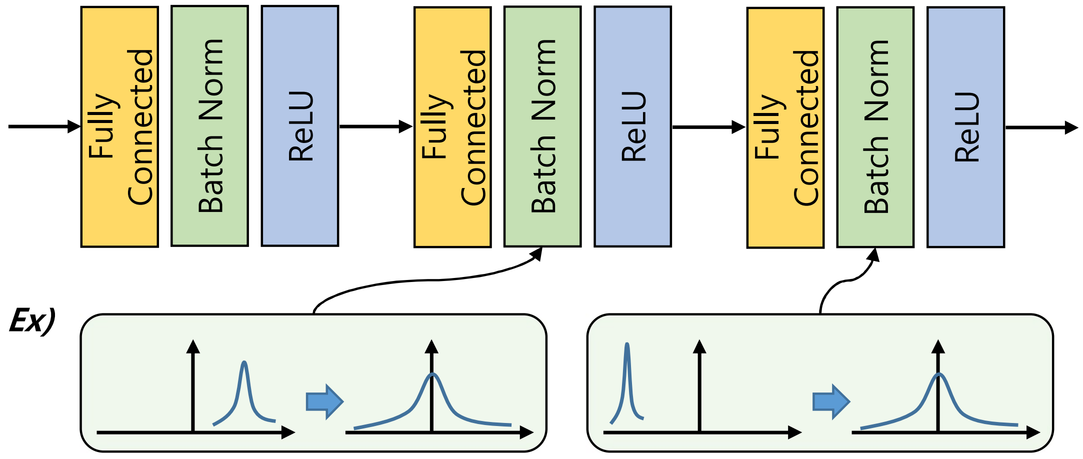
- **해결방법 6 ➔ 배치 정규화(Batch Normalization)**
  - 출력값을 정규화하는 작업

<배치 정규화 적용 전>



학습 과정에서 계층별로 입력의 데이터 분포가 달라지는 현상을
내부 공변량 변화(Internal Covariate Shift)라고 함

<배치 정규화 적용 후>



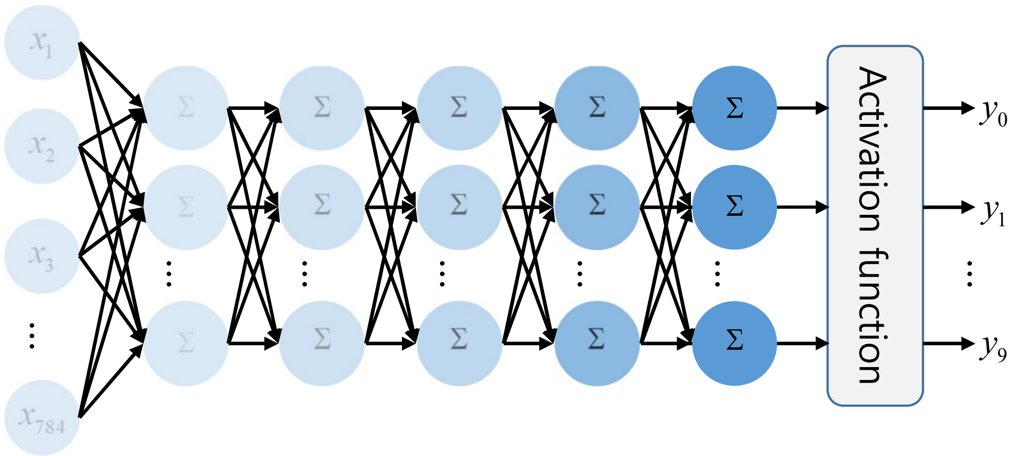
학습 과정에서 배치별로 평균과 분산을 이용해 정규화하는 계층을
배치 정규화 계층이라 함

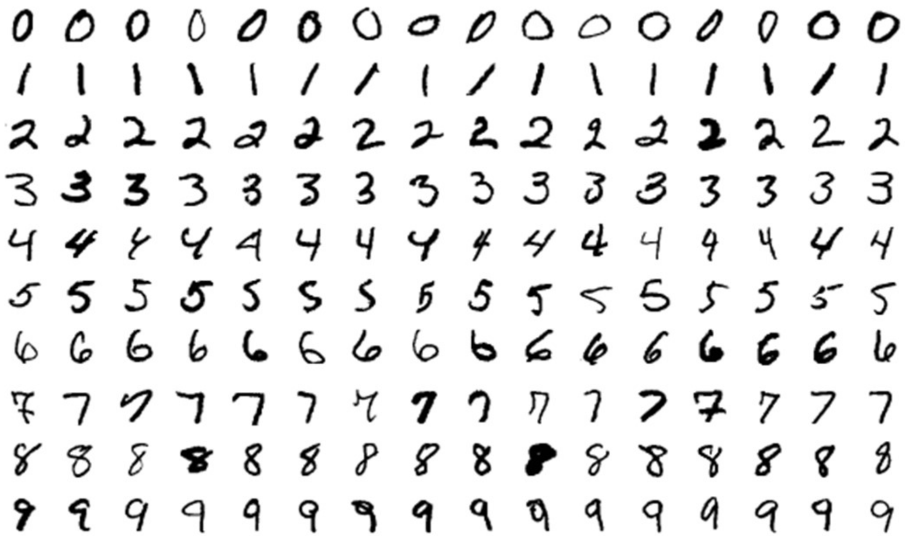
실 습

동아대학교
DONG-A UNIVERSITY

Overfitting 문제 실습

- **Overfitting 이 주로 일어나는 경우**
  - 매개변수가 많은 표현력이 높은 모델
  - 훈련데이터가 적음





동아대학교 DONG-A UNIVERSITY

Overfitting 문제 실습

- **Overfitting 문제 확인**
  - (1) MNIST train dataset 개수 변경 60000 ➔ 300

MNIST train dataset

Image	img0	img1	img2	…	img 59999
Label	5	0	4	…	8

➔

MNIST train dataset

Image	img0	img1	img2	…	img 300
Label	5	0	4	…	6

Overfitting 문제 실습

- **Overfitting 문제 확인**
  - (1) MNIST train dataset 개수 변경 60000 ➔ 300

▾ MNIST train dataset 개수 변경

[59]
```python
mnist_train.data = mnist_train.data[:300]
mnist_train.targets = mnist_train.targets[:300]
mnist_train
```

→ Image 데이터 개수 300으로 설정
→ Label 데이터 개수 300으로 설정

```
Dataset MNIST
    Number of datapoints: 300
    Root location: ./drive/MyDrive/dataset
    Split: Train
    StandardTransform
Transform: ToTensor()
```

데이터 개수 확인

Overfitting 문제 실습

- **Overfitting 문제 확인**
  - (2) MLP 모델 정의

Multi-layer Perceptron 모델 정의: 5-layer

```
[ ] class MLP(nn.Module):
      def __init__(self):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(in_features=784, out_features=100)
        self.fc2 = nn.Linear(in_features=100, out_features=100)
        self.fc3 = nn.Linear(in_features=100, out_features=100)
        self.fc4 = nn.Linear(in_features=100, out_features=100)
        self.fc5 = nn.Linear(in_features=100, out_features=10)
        self.relu = nn.ReLU()


      def forward(self, x):
        x = x.view(-1, 28*28)
        y = self.relu(self.fc1(x))
        y = self.relu(self.fc2(y))
        y = self.relu(self.fc3(y))
        y = self.relu(self.fc4(y))
        y = self.fc5(y)
        return y
```

Overfitting 문제 실습

- **Overfitting 문제 확인**
  - (3) Hyper-parameter 지정

Hyper-parameter 지정

```
batch_size = 10
learning_rate = 0.1
training_epochs = 100

loss_function = nn.CrossEntropyLoss()
network = MLP()
optimizer = torch.optim.SGD(network.parameters(), lr = learning_rate)

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

→ Epoch 100 설정

→ Network 이름 확인

Overfitting 문제 실습

- **Overfitting 문제 확인**
  - (4) MLP 학습을 위한 반복문 선언

Network Training 진행

[15]

```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 Loss = %f' %(epoch+1, avg_cost))
print('Learning finished')
```

Overfitting 문제 실습

- **Overfitting 문제 확인**
  - (4) MLP 학습을 위한 반복문 선언

Network Training 진행

[15]
```
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 Loss = %f' %(epoch+1, avg_cost))
print('Learning finished')
```

```
Epoch: 85 Loss = 0.000251
Epoch: 86 Loss = 0.000246
Epoch: 87 Loss = 0.000242
Epoch: 88 Loss = 0.000238
Epoch: 89 Loss = 0.000235
Epoch: 90 Loss = 0.000231
Epoch: 91 Loss = 0.000227
Epoch: 92 Loss = 0.000224
Epoch: 93 Loss = 0.000221
Epoch: 94 Loss = 0.000217
Epoch: 95 Loss = 0.000214
Epoch: 96 Loss = 0.000211
Epoch: 97 Loss = 0.000208
Epoch: 98 Loss = 0.000205
Epoch: 99 Loss = 0.000202
Epoch: 100 Loss = 0.000200
Learning finished
```


동아대학교 DONG-A UNIVERSITY

Overfitting 문제 실습

- **Overfitting 문제 확인**
  - (5) MNIST Train dataset, MNIST Test dataset 분류 성능 확인

```
[24] with torch.no_grad():
       img_test = mnist_train.data.float()
       label_test = mnist_train.targets

       prediction = network(img_test)

       correct_prediction = torch.argmax(prediction, 1) == label_test
       accuracy = correct_prediction.float().mean()

       print('Accuracy: ',accuracy.item())

     Accuracy:  1.0
```

→ Train Data 적용

정답률: 100%

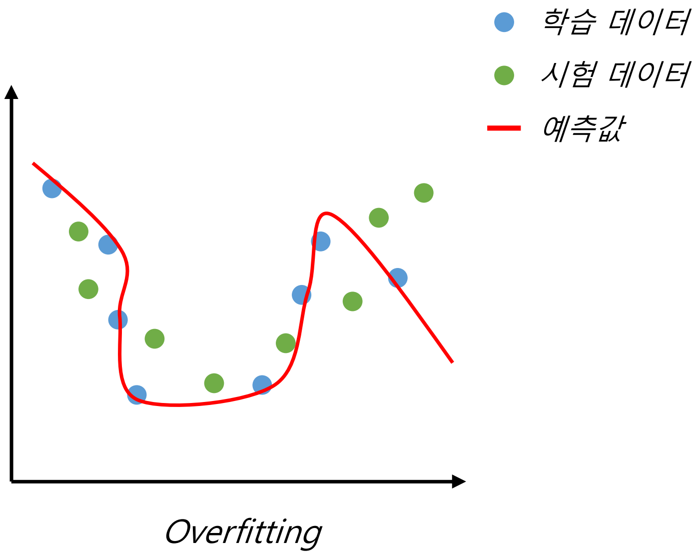
```
[22] with torch.no_grad():
       img_test = mnist_test.data.float()
       label_test = mnist_test.targets

       prediction = network(img_test)

       correct_prediction = torch.argmax(prediction, 1) == label_test
       accuracy = correct_prediction.float().mean()

       print('Accuracy: ',accuracy.item())

     Accuracy:  0.763700008392334
```

정답률: 76.3%

Overfitting 문제 실습

- **Overfitting 문제 확인**
  - (5) MNIST Train dataset, MNIST Test dataset 분류 성능 확인

학습 데이터 정답률: 100%



시험 데이터 정답률: 76.3%

Overfitting 문제 발생!

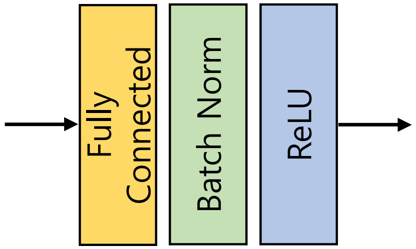
Overfitting 문제 실습

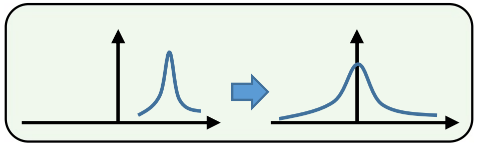
- **Overfitting 문제 해결(1): Batch Normalization**
  - (1) MLP 모델 재정의
    - Batch Normalization 선언 및 적용

```
[25] class MLP(nn.Module):
       def __init__(self):
         super(MLP, self).__init__()
         self.fc1 = nn.Linear(in_features=784, out_features=100)
         self.fc2 = nn.Linear(in_features=100, out_features=100)
         self.fc3 = nn.Linear(in_features=100, out_features=100)
         self.fc4 = nn.Linear(in_features=100, out_features=100)
         self.fc5 = nn.Linear(in_features=100, out_features=10)
         self.relu = nn.ReLU()
         self.bn = nn.BatchNorm1d(100)
       def forward(self, x):
         x = x.view(-1, 28*28)
         y = self.relu(self.bn(self.fc1(x)))
         y = self.relu(self.bn(self.fc2(y)))
         y = self.relu(self.bn(self.fc3(y)))
         y = self.relu(self.bn(self.fc4(y)))
         y = self.fc5(y)
         return y
```

→ Batch Normalization 수행 선언 (100 features)

→ Batch Normalization 적용





Overfitting 문제 실습

- **Overfitting 문제 해결(1): Batch Normalization**
  - (2) Hyper-parameter 지정 및 Training 진행

```
[9]  batch_size = 10
     learning_rate = 0.1
     training_epochs = 100

     loss_function = nn.CrossEntropyLoss()
     network = MLP()
     optimizer = torch.optim.SGD(network.parameters(), lr = learning_rate)

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

```
[24]
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 Loss = %f' %(epoch+1, avg_cost))
print('Learning finished')
```

Overfitting 문제 실습

- **Overfitting 문제 해결(1): Batch Normalization**
  - (3) MNIST Test dataset 분류 성능 확인

```
[29] with torch.no_grad():
       img_test = mnist_test.data.float()
       label_test = mnist_test.targets

       prediction = network(img_test)

       correct_prediction = torch.argmax(prediction, 1) == label_test
       accuracy = correct_prediction.float().mean()

       print('Accuracy: ',accuracy.item())

     Accuracy:  0.7935000061988831
```

정답률: 79.3%

- **Overfitting 문제 해결(2): Dropout**
  - (1) MLP 모델 재정의
    - Dropout 선언 및 적용

```
[31] class MLP(nn.Module):
       def __init__(self):
         super(MLP, self).__init__()
         self.fc1 = nn.Linear(in_features=784, out_features=100)
         self.fc2 = nn.Linear(in_features=100, out_features=100)
         self.fc3 = nn.Linear(in_features=100, out_features=100)
         self.fc4 = nn.Linear(in_features=100, out_features=100)
         self.fc5 = nn.Linear(in_features=100, out_features=10)
         self.relu = nn.ReLU()
         self.dropout = nn.Dropout(0.2)

       def forward(self, x):
         x = x.view(-1, 28*28)
         y = self.relu(self.fc1(x))
         y = self.dropout(y)
         y = self.relu(self.fc2(y))
         y = self.dropout(y)
         y = self.relu(self.fc3(y))
         y = self.dropout(y)
         y = self.relu(self.fc4(y))
         y = self.dropout(y)
         y = self.fc5(y)
         return y
```

→ Dropout 선언 (0.2 비율)

→ Dropout 적용

Overfitting 문제 실습

- **Overfitting 문제 해결(2): Dropout**
  - (2) Hyper-parameter 지정 및 Training 진행

[23]
```
batch_size = 100
learning_rate = 0.1
training_epochs = 15
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)
```

[24]
```
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 Loss = %f' %(epoch+1, avg_cost))
print('Learning finished')
```

Overfitting 문제 실습

- **Overfitting 문제 해결(2): Dropout**
  - (3) MNIST Test dataset 분류 성능 확인

```
[120] with torch.no_grad():
        network.eval()  → Network Dropout 비활성
        img_test = mnist_test.data.float()
        label_test = mnist_test.targets

        prediction = network(img_test)

        correct_prediction = torch.argmax(prediction, 1) == label_test
        accuracy = correct_prediction.float().mean()

        print('Accuracy: ',accuracy.item())

    Accuracy:  0.7954000234603882
```

정답률: 79.3%

동아대학교 DONG-A UNIVERSITY

- **Overfitting 문제 해결(3): Data Augmentation**
  - (1) MNIST Train Data Rotation 변환

```
[121] import torchvision.transforms as transform

      trans_rotation_left_15=transform.RandomRotation((-15,-15))
      rotation_data_left_15=trans_rotation_left_15(mnist_train.data)

      trans_rotation_left_30=transform.RandomRotation((-30,-30))
      rotation_data_left_30=trans_rotation_left_30(mnist_train.data)

      trans_rotation_right_15=transform.RandomRotation((15,15))
      rotation_data_right_15=trans_rotation_right_15(mnist_train.data)

      trans_rotation_right_30=transform.RandomRotation((30,30))
      rotation_data_right_30=trans_rotation_right_30(mnist_train.data)
```

→ 반 시계 방향 15도 Rotation 수행

→ 반 시계 방향 30도 Rotation 수행

→ 시계 방향 15도 Rotation 수행

→ 시계 방향 30도 Rotation 수행

Overfitting 문제 실습

- **Overfitting 문제 해결(3): Data Augmentation**
  - (2) rotation_data 형태확인
  - (3) rotation 수행된 데이터 MNIST Train Dataset 에 합치기

2

```
[149] print(rotation_data_left_15.shape)
      print(rotation_data_left_30.shape)
      print(rotation_data_right_15.shape)
      print(rotation_data_right_30.shape)

      torch.Size([300, 28, 28])
      torch.Size([300, 28, 28])
      torch.Size([300, 28, 28])
      torch.Size([300, 28, 28])
```

3

```
[129] mnist_train.data = torch.cat((mnist_train.data, rotation_data_left_15, rotation_data_left_30, rotation_data_right_15, rotation_data_right_30),0)
      mnist_train.data.shape

      torch.Size([1500, 28, 28])
```

Overfitting 문제 실습

- **Overfitting 문제 해결(3): Data Augmentation**
  - (4) Train Dataset label 늘리기

```
[150] mnist_train.targets = mnist_train.targets.repeat(5)
      mnist_train.targets.shape

      torch.Size([1500])
```

→ 같은 Tensor 데이터 5배 증가

```
[159] print(mnist_train.targets[0])
      print(mnist_train.targets[300])
      print(mnist_train.targets[600])
      print(mnist_train.targets[900])
      print(mnist_train.targets[1200])

      tensor(5)
      tensor(5)
      tensor(5)
      tensor(5)
      tensor(5)
```

```
[160] print(mnist_train.targets[1])
      print(mnist_train.targets[301])
      print(mnist_train.targets[601])
      print(mnist_train.targets[901])
      print(mnist_train.targets[1201])

      tensor(0)
      tensor(0)
      tensor(0)
      tensor(0)
      tensor(0)
```

Overfitting 문제 실습

- **Overfitting 문제 해결(3): Data Augmentation**
  - **300 Train Dataset ➔ 1500 Train Dataset**

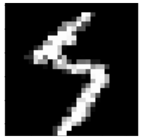
MNIST train dataset

Image	img0	img1	img2	…	img 300	…	img 600	…	img 900	…	img 1200	…	img 1499
Label	5	0	4	…	5	…	5	…	5	…	5	…	6

Overfitting 문제 실습

- **Overfitting 문제 해결(3): Data Augmentation**
  - **300 Train Dataset ➔ 1500 Train Dataset**

<u>MNIST train dataset</u>

Image		img1	img2	…		…		…		…		…	img 1499
Label	5	0	4	…	5	…	5	…	5	…	5	…	6

동아대학교 DONG-A UNIVERSITY

Overfitting 문제 실습

- **Overfitting 문제 해결(3): Data Augmentation**
  - (5) MLP 모델 재정의

Multi-layer Perceptron 모델 정의: 5-layer

```
[ ] class MLP(nn.Module):
      def __init__(self):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(in_features=784, out_features=100)
        self.fc2 = nn.Linear(in_features=100, out_features=100)
        self.fc3 = nn.Linear(in_features=100, out_features=100)
        self.fc4 = nn.Linear(in_features=100, out_features=100)
        self.fc5 = nn.Linear(in_features=100, out_features=10)
        self.relu = nn.ReLU()


      def forward(self, x):
        x = x.view(-1, 28*28)
        y = self.relu(self.fc1(x))
        y = self.relu(self.fc2(y))
        y = self.relu(self.fc3(y))
        y = self.relu(self.fc4(y))
        y = self.fc5(y)
        return y
```

Overfitting 문제 실습

- **Overfitting 문제 해결(3): Data Augmentation**
  - (6) Hyper-parameter 지정 및 Training 진행

```
[9]  batch_size = 10
     learning_rate = 0.1
     training_epochs = 100

     loss_function = nn.CrossEntropyLoss()
     network = MLP()
     optimizer = torch.optim.SGD(network.parameters(), lr = learning_rate)

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

```
[24]
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 Loss = %f' %(epoch+1, avg_cost))
print('Learning finished')
```

Overfitting 문제 실습

- **Overfitting 문제 해결(3): Data Augmentation**
  - (7) MNIST Test dataset 분류 성능 확인

```
[173] with torch.no_grad():
        img_test = mnist_test.data.float()
        label_test = mnist_test.targets

        prediction = network(img_test)

        correct_prediction = torch.argmax(prediction, 1) == label_test
        accuracy = correct_prediction.float().mean()

        print('Accuracy: ',accuracy.item())

    Accuracy:  0.8073999881744385
```

정답률: 80.73%

동아대학교 DONG-A UNIVERSITY

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