

Idea Factory Intensive Program #2

딥러닝 홀로서기

이론강의/PyTorch실습/코드리뷰

딥러닝(Deep Learning)에 관심이 있는 학생 발굴을 통한
딥러닝의 이론적 배경 강의 및 오픈소스 딥러닝 라이브러리 PyTorch를 활용한 실습

#10

Acknowledgement

Sung Kim's 모두를 위한 머신러닝/딥러닝 강의

- <https://hunkim.github.io/ml/>
- https://www.youtube.com/playlist?list=PLIMkM4tgfjnLSOjrEJN31gZATbcj_MpUm

Andrew Ng's and other ML tutorials

- <https://class.coursera.org/ml-003/lecture>
- <http://www.holehouse.org/mlclass/> (note)
- [Deep Learning Tutorial](#)
- [Andrej Karpathy's Youtube channel](#)

WooYeon Kim & SeongOk Ryu's KAIST CH485 Artificial Intelligence and Chemistry

- <https://github.com/SeongokRyu/CH485---Artificial-Intelligence-and-Chemistry>

SungJu Hwang's KAIST CS492 Deep Learning Course Material

Many insightful articles, blog posts and Youtube channels

Facebook community

- Tensorflow KR (<https://www.facebook.com/groups/TensorFlowKR/>)
- Pytorch KR (<https://www.facebook.com/groups/PyTorchKR/>)

Medium Channel and Writers

- Toward Data Science (<https://towardsdatascience.com/>)

How was Assignment #1?





How was Assignment #1?

1. MLP hidden layer 수가 바뀔 때마다 매번 코드에 직접 쳐야 함(개불편)

How was Assignment #1?

1. MLP hidden layer 수가 바뀔 때마다 매번 코드에 직접 쳐야 함(개불편)
2. 실험 돌리는 거 오래 걸려요(현기증 남)

How was Assignment #1?

1. MLP hidden layer 수가 바뀔 때마다 매번 코드에 직접 쳐야 함(개불편)
2. 실험 돌리는 거 오래 걸려요(현기증 남)
3. 같은 코드인데도 돌릴 때 마다 결과가 달라여?!?!?!

How was Assignment #1?

1. MLP hidden layer 수가 바뀔 때마다 매번 코드에 직접 쳐야 함(개불편)
2. 실험 돌리는 거 오래 걸려요(현기증 남)
3. 같은 코드인데도 돌릴 때 마다 결과가 달라여?!?!?!?
4. Train Loss는 줄어드는데 Validation Loss는 안 줄어들이어요!


How was Assignment #1?

1. MLP hidden layer 수가 바뀔 때마다 매번 코드에 직접 쳐야 함(개불편)
2. 실험 돌리는 거 오래 걸려요(현기증 남)
3. 같은 코드인데도 돌릴 때 마다 결과가 달라여?!?!?!
4. Train Loss는 줄어드는데 Validation Loss는 안 줄어들이어요!
5. 변수들을 어떤 식으로 어떻게 바꿔야 할 지 모르겠어요!


How was Assignment #1?

1. MLP hidden layer 수가 바뀔 때마다 매번 코드에 직접 쳐야 함(개불편)
2. 실험 돌리는 거 오래 걸려요(현기증 남)
3. 같은 코드인데도 돌릴 때 마다 결과가 달라여?!?!?!?
4. Train Loss는 줄어드는데 Validation Loss는 안 줄어들이어요!
5. 변수들을 어떤 식으로 어떻게 바꿔야 할 지 모르겠어요!
6. 그리고 아직도 Train/Validation/Test 어떻게 써야 하는지도 모르겠어요!


How was Assignment #1?

1. MLP hidden layer 수가 바뀔 때마다 매번 코드에 직접 쳐야 함(개불편)  `nn.ModuleList`
`Argparse`
함수화
2. 실험 돌리는 거 오래 걸려요(현기증 남)
3. 같은 코드인데도 돌릴 때 마다 결과가 달라여?!?!?!?
4. Train Loss는 줄어드는데 Validation Loss는 안 줄어들이어요!
5. 변수들을 어떤 식으로 어떻게 바꿔야 할 지 모르겠어요!
6. 그리고 아직도 Train/Validation/Test 어떻게 써야 하는지도 모르겠어요!


How was Assignment #1?

1. MLP hidden layer 수가 바뀔 때마다 매번 코드에 직접 쳐야 함(개불편)
2. 실험 돌리는 거 오래 걸려요(현기증 남)  GPU로 돌려버리기!
3. 같은 코드인데도 돌릴 때 마다 결과가 달라여?!?!?!
4. Train Loss는 줄어드는데 Validation Loss는 안 줄어들이요!
5. 변수들을 어떤 식으로 어떻게 바꿔야 할 지 모르겠어요!
6. 그리고 아직도 Train/Validation/Test 어떻게 써야 하는지도 모르겠어요!


How was Assignment #1?

1. MLP hidden layer 수가 바뀔 때마다 매번 코드에 직접 쳐야 함(개불편)
2. 실험 돌리는 거 오래 걸려요(현기증 남)
3. 같은 코드인데도 돌릴 때 마다 결과가 달라여?!?!?!  Seed 고정
4. Train Loss는 줄어드는데 Validation Loss는 안 줄어들이요!
5. 변수들을 어떤 식으로 어떻게 바꿔야 할 지 모르겠어요!
6. 그리고 아직도 Train/Validation/Test 어떻게 써야 하는지도 모르겠어요!

How was Assignment #1?

1. MLP hidden layer 수가 바뀔 때마다 매번 코드에 직접 쳐야 함(개불편)
2. 실험 돌리는 거 오래 걸려요(현기증 남)
3. 같은 코드인데도 돌릴 때 마다 결과가 달라여?!?!?!
4. Train Loss는 줄어드는데 Validation Loss는 안 줄어들이요!  Overfitting
5. 변수들을 어떤 식으로 어떻게 바꿔야 할 지 모르겠어요!
6. 그리고 아직도 Train/Validation/Test 어떻게 써야 하는지도 모르겠어요!

How was Assignment #1?

1. MLP hidden layer 수가 바뀔 때마다 매번 코드에 직접 쳐야 함(개불편)
2. 실험 돌리는 거 오래 걸려요(현기증 남)
3. 같은 코드인데도 돌릴 때 마다 결과가 달라여?!?!?!
4. Train Loss는 줄어드는데 Validation Loss는 안 줄어 들어요!
5. 변수들을 어떤 식으로 어떻게 바꿔야 할 지 모르겠어요!  Hyperparameter Tuning
6. 그리고 아직도 Train/Validation/Test 어떻게 써야 하는지도 모르겠어요!

How was Assignment #1?

1. MLP hidden layer 수가 바뀔 때마다 매번 코드에 직접 쳐야 함(개불편)
2. 실험 돌리는 거 오래 걸려요(현기증 남)
3. 같은 코드인데도 돌릴 때 마다 결과가 달라여?!?!?!?
4. Train Loss는 줄어드는데 Validation Loss는 안 줄어들이요!
5. 변수들을 어떤 식으로 어떻게 바꿔야 할 지 모르겠어요!
6. 그리고 아직도 Train/Validation/Test 어떻게 써야 하는지도 모르겠어요!



이것도 Hyperparameter Tuning과 관련

Today's Time Schedule

Today's Time Schedule

Assignment #1 Review

How to Parameterize Entire Code

How to Run Code with GPU!

How to Overcome Overfitting

Big Wave: Hyperparameter Tuning

1 hour?

1 hour

2 hour

Today's Time Schedule

Assignment #1 Review

How to Parameterize Entire Code

How to Run Code with GPU!

How to Overcome Overfitting

Big Wave: Hyperparameter Tuning

1 hour?

1 hour

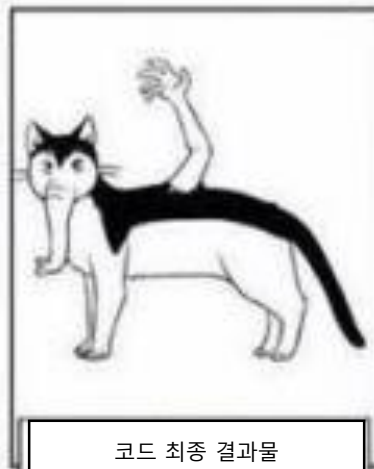
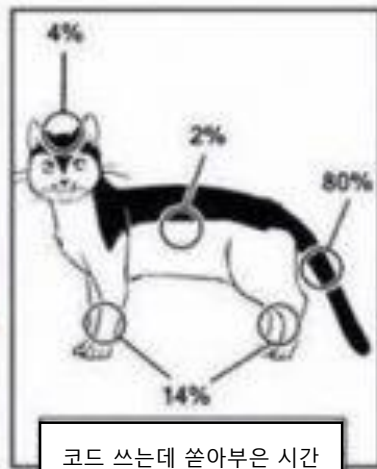
2 hour

Assignment #1 Review

Assignment #1 Review

- Construct MLP model with given parameter (hidden unit, hidden layer)
- Try various combination of MLP model and learning rate
- Organize experiment results

코드의 완성 과정



Assignment #1 Review

```
class MLPModel(nn.Module):
    def __init__(self, hidden_unit, hidden_layer):
        super(MLPModel, self).__init__()
        self.linear1 = nn.Linear(in_features=784, out_features=hidden_unit)
        self.linear2 = nn.Linear(in_features=hidden_unit, out_features=10)
        self.relu = nn.ReLU()
        self.hidden_layer = hidden_layer

    def forward(self, x):
        x = self.linear1(x)
        for i in range(self.hidden_layer):
            x = self.relu(x)
        x = self.linear2(x)
        return x
```

Linear(0.005)	MLP(100, 1, 0.05)	MLP(100, 1, 0.005)	MLP(100, 10, 0.005)	MLP(10, 1, 0.0005)	MLP(100, 1, 0.0005)	MLP(1000, 1, 0.0005)
87%	10%	94%	94.2%	92%	97.5%	98.1%

Assignment #1 Review

```
class MLPModel(nn.Module):
    def __init__(self, hidden_unit, hidden_layer):
        super(MLPModel, self).__init__()
        self.linear1 = nn.Linear(in_features=784, out_features=hidden_unit)
        self.linear2 = nn.Linear(in_features=hidden_unit, out_features=10)
        self.relu = nn.ReLU()
        self.hidden_layer = hidden_layer

    def forward(self, x):
        x = self.linear1(x)
        for i in range(self.hidden_layer):
            x = self.relu(x)
        x = self.linear2(x)
        return x
```

Hidden_layer 수 만큼 linear layer를 생성해야 함



Linear(0.005)	MLP(100, 1, 0.05)	MLP(100, 1, 0.005)	MLP(100, 10, 0.005)	MLP(10, 1, 0.0005)	MLP(100, 1, 0.0005)	MLP(1000, 1, 0.0005)
87%	10%	94%	94.2%	92%	97.5%	98.1%

Assignment #1 Review

```
class MLPModel(nn.Module):
    def __init__(self, hidden_unit, hidden_layer):
        super(MLPModel, self).__init__()
        self.linear1 = nn.Linear(in_features=784, out_features=hidden_unit)
        self.linear2 = nn.Linear(in_features=hidden_unit, out_features=10)
        self.relu = nn.ReLU()
        self.hidden_layer = hidden_layer

    def forward(self, x):
        x = self.linear1(x)
        for i in range(self.hidden_layer):
            x = self.relu(x)
        x = self.linear2(x)
        return x
```

Hidden_layer 수 만큼 linear layer를 생성해야 함

지금은 relu만 hidden_layer 수 만큼 통과되는 중
x = self.relu(self.linear(x)) 이런 식으로 변경할 것

Linear(0.005)	MLP(100, 1, 0.05)	MLP(100, 1, 0.005)	MLP(100, 10, 0.005)	MLP(10, 1, 0.0005)	MLP(100, 1, 0.0005)	MLP(1000, 1, 0.0005)
87%	10%	94%	94.2%	92%	97.5%	98.1%

Assignment #1 Review

2. Model Architecture

```
class LinearModel(nn.Module):
    def __init__(self):
        super(LinearModel, self).__init__()
        self.linear = nn.Linear(in_features=784, out_features=10, bias=True)

    def forward(self, x):
        x = self.linear(x)
        return x

class MLPModel_11(nn.Module):
    def __init__(self, in_dim, out_dim, hid_dim):
        super(MLPModel_11, self).__init__()
        self.linear1 = nn.Linear(in_dim, hid_dim)
        self.linear2 = nn.Linear(hid_dim, out_dim)
        self.act = nn.ReLU()

    def forward(self, x):
        x = self.linear1(x)
        x = self.act(x)
        x = self.linear2(x)
        return x

class MLPModel_12(nn.Module):
    def __init__(self, in_dim, out_dim, hid_dim_1, hid_dim_2):
        super(MLPModel_12, self).__init__()
        self.linear1 = nn.Linear(in_dim, hid_dim_1)
        self.linear2 = nn.Linear(hid_dim_1, hid_dim_2)
        self.linear3 = nn.Linear(hid_dim_2, out_dim)
        self.act = nn.ReLU()

    def forward(self, x):
        x = self.linear1(x)
        x = self.act(x)
        x = self.linear2(x)
        x = self.act(x)
        x = self.linear3(x)
        return x
```

```
class MLPModel_110(nn.Module):
    def __init__(self, in_dim, out_dim, hid_dim):
        super(MLPModel_110, self).__init__()
        self.linear1 = nn.Linear(in_dim, hid_dim)
        self.linear2 = nn.Linear(hid_dim, hid_dim)
        self.linear3 = nn.Linear(hid_dim, hid_dim)
        self.linear4 = nn.Linear(hid_dim, hid_dim)
        self.linear5 = nn.Linear(hid_dim, hid_dim)
        self.linear6 = nn.Linear(hid_dim, hid_dim)
        self.linear7 = nn.Linear(hid_dim, hid_dim)
        self.linear8 = nn.Linear(hid_dim, hid_dim)
        self.linear9 = nn.Linear(hid_dim, hid_dim)
        self.linear10 = nn.Linear(hid_dim, hid_dim)
        self.linear11 = nn.Linear(hid_dim, out_dim)
        self.act = nn.ReLU()

    def forward(self, x):
        x = self.linear1(x)
        x = self.act(x)
        x = self.linear2(x)
        x = self.act(x)
        x = self.linear3(x)
        x = self.act(x)
        x = self.linear4(x)
        x = self.act(x)
        x = self.linear5(x)
        x = self.act(x)
        x = self.linear6(x)
        x = self.act(x)
        x = self.linear7(x)
        x = self.act(x)
        x = self.linear8(x)
        x = self.act(x)
        x = self.linear9(x)
        x = self.act(x)
        x = self.linear10(x)
        x = self.act(x)
        x = self.linear11(x)
        return x
```

Assignment #1 Review

2. Model Architecture

```
class LinearModel(nn.Module):
    def __init__(self):
        super(LinearModel, self).__init__()
        self.linear = nn.Linear(in_features=784, out_features=10, bias=True)

    def forward(self, x):
        x = self.linear(x)
        return x

class MLPModel_11(nn.Module):
    def __init__(self, in_dim, out_dim, hid_dim):
        super(MLPModel_11, self).__init__()
        self.linear1 = nn.Linear(in_dim, hid_dim)
        self.linear2 = nn.Linear(hid_dim, out_dim)
        self.act = nn.ReLU()

    def forward(self, x):
        x = self.linear1(x)
        x = self.act(x)
        x = self.linear2(x)
        return x

class MLPModel_12(nn.Module):
    def __init__(self, in_dim, out_dim, hid_dim_1, hid_dim_2):
        super(MLPModel_12, self).__init__()
        self.linear1 = nn.Linear(in_dim, hid_dim_1)
        self.linear2 = nn.Linear(hid_dim_1, hid_dim_2)
        self.linear3 = nn.Linear(hid_dim_2, out_dim)
        self.act = nn.ReLU()

    def forward(self, x):
        x = self.linear1(x)
        x = self.act(x)
        x = self.linear2(x)
        x = self.act(x)
        x = self.linear3(x)
        return x
```

```
class MLPModel_110(nn.Module):
    def __init__(self, in_dim, out_dim, hid_dim):
        super(MLPModel_110, self).__init__()
        self.linear1 = nn.Linear(in_dim, hid_dim)
        self.linear2 = nn.Linear(hid_dim, hid_dim)
        self.linear3 = nn.Linear(hid_dim, hid_dim)
        self.linear4 = nn.Linear(hid_dim, hid_dim)
        self.linear5 = nn.Linear(hid_dim, hid_dim)
        self.linear6 = nn.Linear(hid_dim, hid_dim)
        self.linear7 = nn.Linear(hid_dim, hid_dim)
        self.linear8 = nn.Linear(hid_dim, hid_dim)
        self.linear9 = nn.Linear(hid_dim, hid_dim)
        self.linear10 = nn.Linear(hid_dim, hid_dim)
        self.linear11 = nn.Linear(hid_dim, out_dim)
        self.act = nn.ReLU()

    def forward(self, x):
        x = self.linear1(x)
        x = self.act(x)
        x = self.linear2(x)
        x = self.act(x)
        x = self.linear3(x)
        x = self.act(x)
        x = self.linear4(x)
        x = self.act(x)
        x = self.linear5(x)
        x = self.act(x)
        x = self.linear6(x)
        x = self.act(x)
        x = self.linear7(x)
        x = self.act(x)
        x = self.linear8(x)
        x = self.act(x)
        x = self.linear9(x)
        x = self.act(x)
        x = self.linear10(x)
        x = self.act(x)
        x = self.linear11(x)
        return x
```

Wow..
만약 100레이어 짜리
를 만들어야 한다면?

Assignment #1 Review

```
class MLPModel(nn.Module):  
    def __init__(self):  
        super(MLPModel, self).__init__()  
        self.linear1 = nn.Linear(in_features=784 , out_features=100)  
        self.linear2 = nn.Linear(100,10)  
        self.relu = nn.ReLU()  
        self.softmax = nn.Softmax()  
  
    def forward(self, x):  
        x = self.linear1(x)  
        x = self.relu(x)  
        x = self.linear2(x)  
        return x
```

괜히 차후에 사용될 수 있으므로
Softmax를 만들지도 말자!!

```
, Test Acc: 97.46%  
, Test Acc: 97.48%  
, Test Acc: 97.6%
```

Assignment #1 Review

```
class MyModel(nn.Module):
    def __init__(self, hidden_nodes):
        super().__init__()
        nodes = (784,) + hidden_nodes + (10,)
        depth = len(nodes)
        linears = [nn.Linear(nodes[i], nodes[i+1]) for i in range(depth-1)]
        self.linears = nn.ModuleList(linears)
        self.relu = nn.ReLU()
        self.depth = depth

    def forward(self, x):
        for linear in self.linears:
            x = linear(x)
            x = self.relu(x)
        return x
```

```
class MLPModel2(nn.Module):
    def __init__(self, in_features, out_features, hid_list):
        super(MLPModel2, self).__init__()

        self.linear1 = nn.Linear(in_features, hid_list[0], bias = True)
        self.hidden = nn.ModuleList()
        for i in range(len(hid_list) - 1):
            self.hidden.append(nn.Linear(hid_list[i], hid_list[i+1], bias = True))
        self.linear2 = nn.Linear(hid_list[-1], out_features, bias = True)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.linear1(x))
        for layer in self.hidden:
            x = self.relu(layer(x))
        x = self.linear2(x)
        return x
```

nn.ModuleList 를 아주 잘 찾아서 사용해주셨습니다 ☺

Assignment #1 Review

```
class MLPModel(nn.Module):
    def __init__(self, input_dim, output_dim, hid_dims):

        super(MLPModel, self).__init__()

        self.relu = nn.ReLU()

        self.hidden = nn.ModuleList()

        if type(hid_dims) == torch.Tensor:

            self.hidden.append(nn.Linear(input_dim, int(hid_dims[0].item())))

            for i in range(len(hid_dims) - 1):
                self.hidden.append(nn.Linear(int(hid_dims[i].item()), int(hid_dims[i+1].item())))

            self.hidden.append(nn.Linear(int(hid_dims[len(hid_dims) - 1].item()), output_dim))

        elif type(hid_dims) == np.ndarray:

            param_list = np.append(np.insert(hid_dims, 0, input_dim), output_dim)

            for i in range(len(param_list) - 1):
                self.hidden.append(nn.Linear(param_list[i], param_list[i+1]))

        elif type(hid_dims) == list:

            hid_dims.append(output_dim)
            hid_dims.insert(0, input_dim)
            for i in range(len(hid_dims) - 1):
                self.hidden.append(nn.Linear(hid_dims[i], hid_dims[i+1]))

        else:
            raise TypeError("hid_dims must be torch.Tensor or numpy.ndarray or list")

    def forward(self, x):
        for layer in self.hidden:
            x = layer(x)
            x = self.relu(x)
        return x
```

Good!

More Safety code!
Use isinstance method?

Assignment #1 Review – Good Question

Loss.backward()를 통해 Back-propagation 계산한 결과가

어떻게 Optimizer.step()을 호출하면 활용되는지?

둘이 딱히 연결되어 있는 곳이 없는거 같은데..?



Optimizer Creation!

```
# ===== Construct Optimizer ===== #  
lr = 0.0001  
optimizer = optim.SGD(model.parameters(), lr=lr)
```


Assignment #1 Review

Model	
Parameter	Parameter Gradient
0.1	0
0.2	0
-1.0	0
0.21	0
0.99	0

```
model = MLPModel(784, 10, [1000])
```

Assignment #1 Review

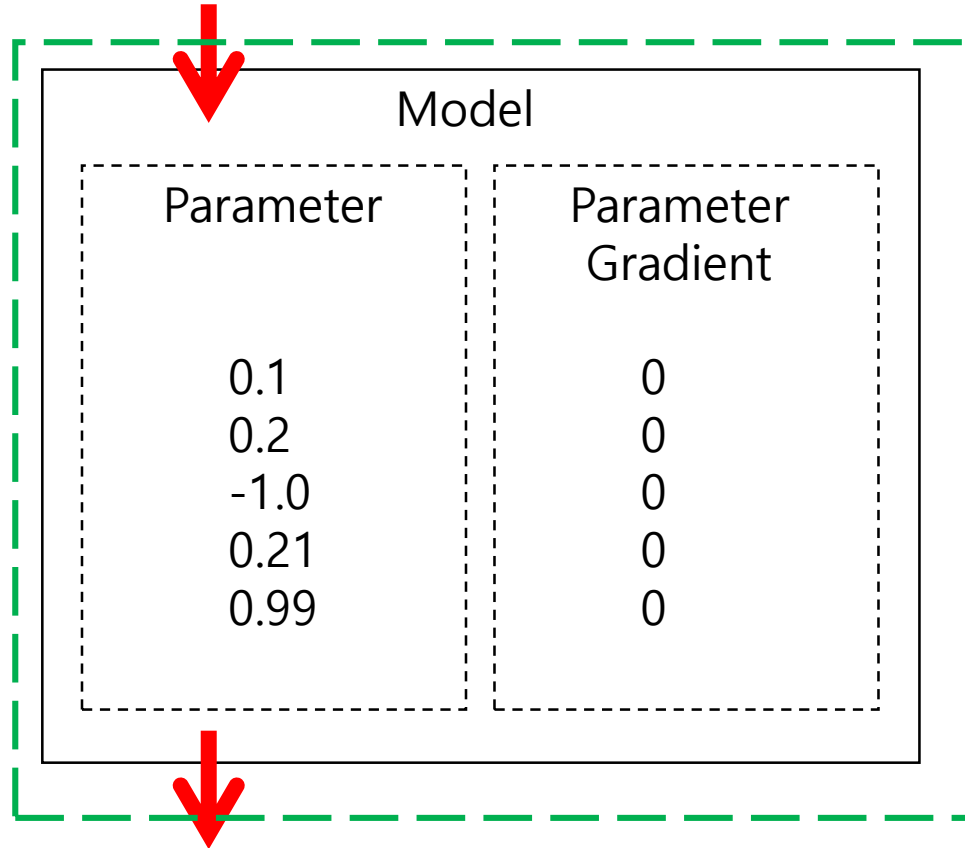
Model	
Parameter	Parameter Gradient
0.1	0
0.2	0
-1.0	0
0.21	0
0.99	0

```
# ===== Construct Optimizer ===== #  
lr = 0.0001  
optimizer = optim.SGD(model.parameters(), lr=lr)
```

Optimizer now takes charge of updating parameters of model

Assignment #1 Review

Train X

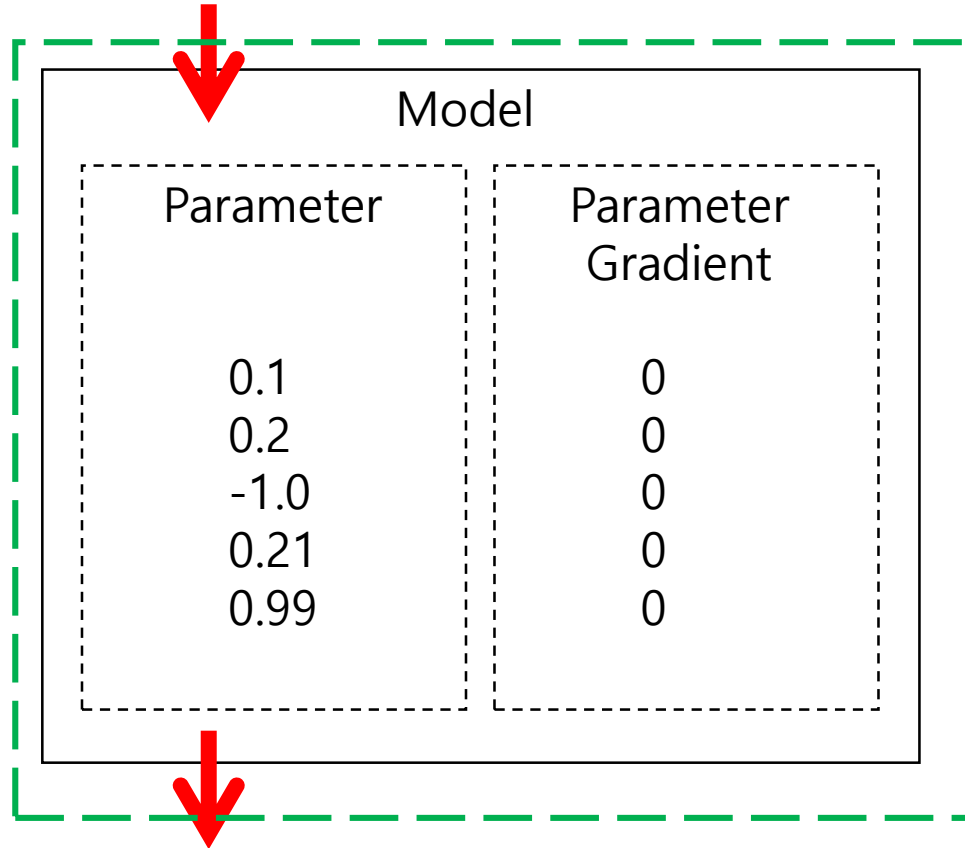


Pred Y

```
pred_y = model(input_X)
```

Assignment #1 Review

Train X



Pred Y

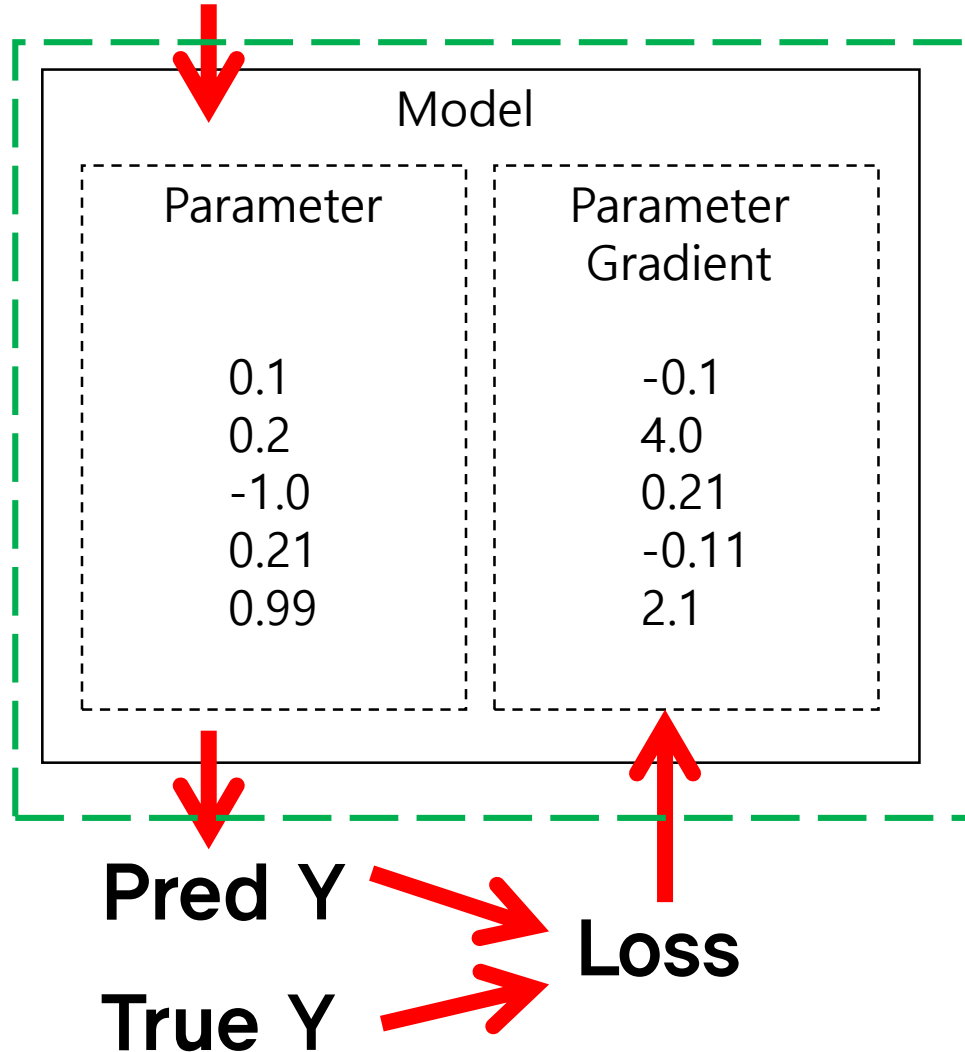
True Y

Loss

```
loss = cls_loss(pred_y.squeeze(), true_y)
```

Assignment #1 Review

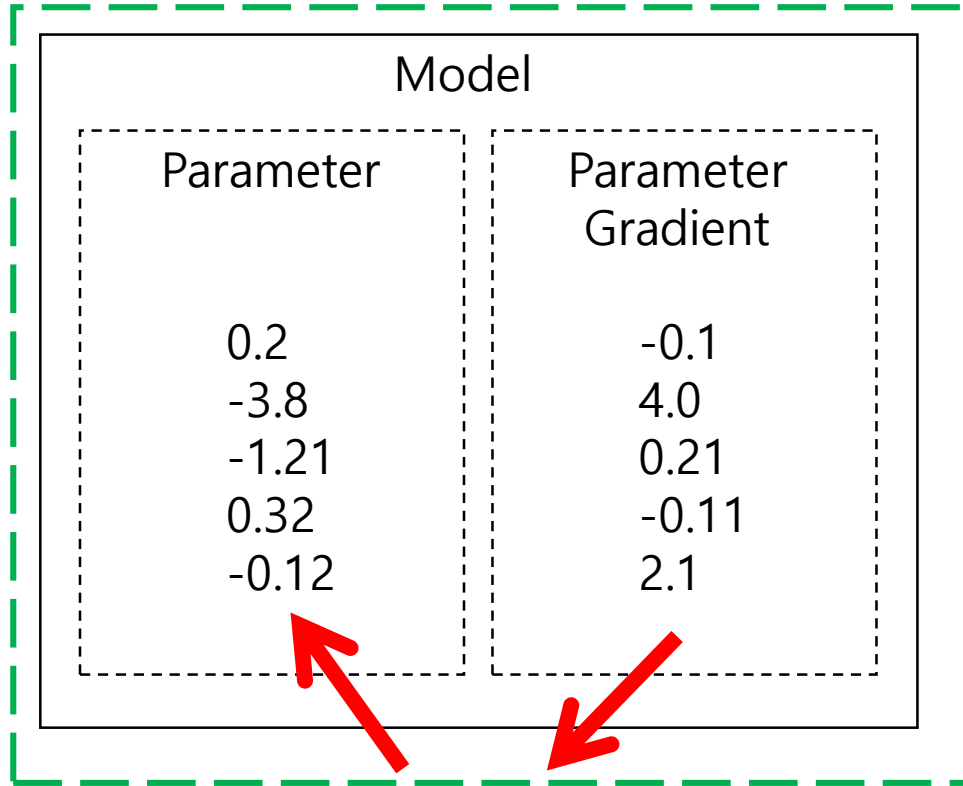
Train X



```
loss.backward()
```

Assignment #1 Review

Model	
Parameter	Parameter Gradient
0.2	-0.1
-3.8	4.0
-1.21	0.21
0.32	-0.11
-0.12	2.1



```
optimizer.step()
```

Assignment #1 Review

Model	
Parameter	Parameter Gradient
0.2	0
-3.8	0
-1.21	0
0.32	0
-0.12	0

```
optimizer.zero_grad()
```

Assignment #1 Review

Why we should use `nn.ModuleList` instead of just python list?

Assignment #1 Review

Why we should use `nn.ModuleList` instead of just python list?



`nn.Module` inside the list will not registered to Optimizer as parameter!

Assignment #1 Review

Why we should use `nn.ModuleList` instead of just python list?



`nn.Module` inside the list will not registered to Optimizer as parameter!



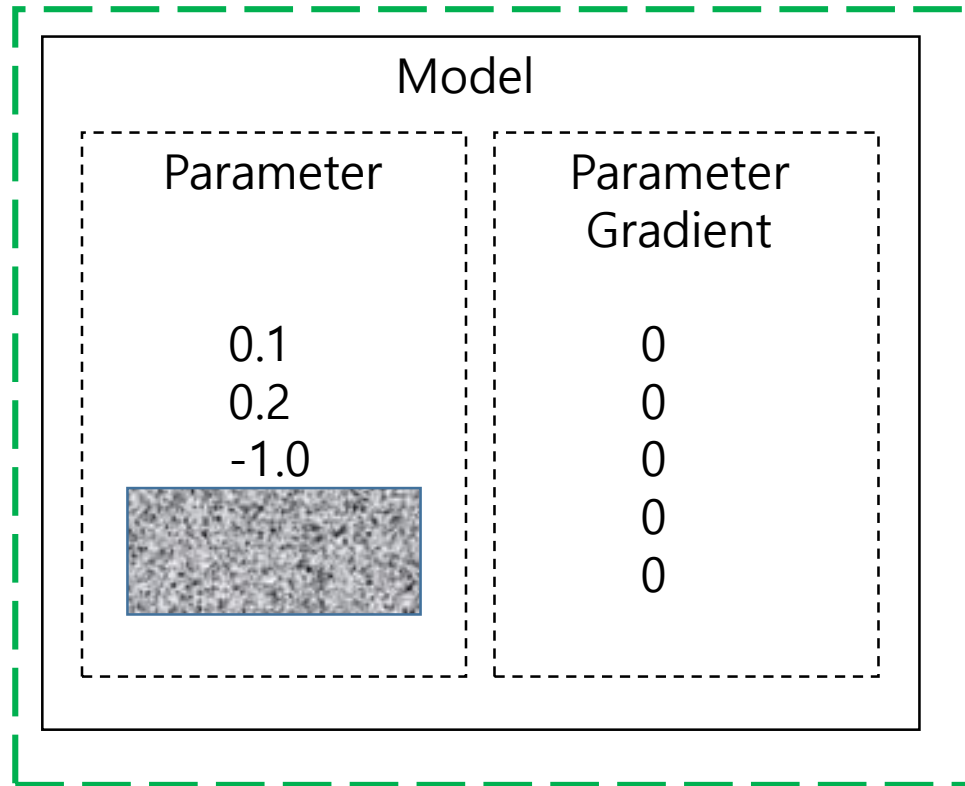
- Number of parameter does not change!
- Accuracy does not change even hyperparameter changed

Assignment #1 Review

Model	
Parameter	Parameter Gradient
0.1	0
0.2	0
-1.0	0
0.21	0
0.99	0

```
# ===== Construct Optimizer ===== #  
lr = 0.0001  
optimizer = optim.SGD(model.parameters(), lr=lr)
```

Assignment #1 Review



```
# ===== Construct Optimizer ===== #  
lr = 0.0001  
optimizer = optim.SGD(model.parameters(), lr=lr)
```

Parameter is not visible to Optimizer without `nn.ModuleList`

Summary

- Nice work everyone!
- Use `nn.ModuleList` when layers are dynamically constructed