파이토치로 배우는 딥러닝 기초

DNN 퀴즈 정답

edwith
boostcourse

Quiz 1 해설

Quiz

정답 Q 1 (단일선택 , 10점)

다음 중 Python의 Library 및 Package에 대한 설명으로 옳지 않은 것은?

- numpy : Scientific computing과 관련된 여러 편리한 기능들을 제공해주는 라이브러리이다.
- **torch : 최근 2.0버전으로 업데이트 되었으며, 내장되어 있는 Keras 를 이용하여 High-level 구현이 가능하다.**
- torch.utils.data : Mini-batch 학습을 위한 패키지이다.
- torchvision : PyTorch 에서 이미지 데이터 로드와 관련된 여러가지 편리한 함수들을 제공하는 라이브러리이다.
- matplotlib.pyplot : 데이터 시각화를 위한 다양한 기능을 제공하는 패키지이다.

1 2 3 4 5 6 7 8 9 10

2번은 Tensorflow 에 관한 설명으로, PyTorch 는 2019년 5월 기준 1.1 버전이 최신버전입니다.

edwith boostcourse

Quiz 2 해설

Quiz

정답 Q 2 (단일선택 , 10점)

torch.Tensor 에 대한 설명으로 다음 중 옳지 않은 것은?

- ○ 다차원 배열(Multi-dimensional Matrix)을 처리하기 위한 가장 기본이 되는 자료형(Data Type)이다.
- ○ NumPy의 ndarray와 거의 비슷하지만, GPU 연산을 함으로써 computing power를 극대화 시킬 수 있다.
- ○ 기본텐서 타입(Default Tensor Type)으로 32비트의 부동소수점(Float) torch.FloatTensor으로 되어있다.
- ● **torch.Tensor의 사칙연산은 torch.Tensor 간이나 torch.Tensor와 Python의 Scala 값 뿐만 아니라 Numpy의 ndarray와도 연산이 가능하다.**
- ○ NumPy의 ndarray 와 마찬가지로 브로드캐스팅(Broadcasting) 적용이 가능하다.

1 2 3 4 5 6 7 8 9 10

torch.Tensor는 torch.Tensor 간이나 torch.Tensor와 Python의 Scala 값과만 연산이 된다. 따라서, NumPy의 ndarray 및 torch.cuda.Tensor와의 연산은 불가능하여 자료형을 일치시켜준 뒤 연산하여야 합니다.

자료형 변경은 다음과 같습니다.

```
[1]  import torch
     import numpy as np

     a = np.random.rand(2,3)
     b = torch.rand(2,3)
```

```
[2]  a = torch.tensor(a)
     b = b.numpy()
     a
     b
```

```
array([[0.46221453, 0.8301099 , 0.6369593 ],
       [0.19132781, 0.06042129, 0.7270087 ]], dtype=float32)
```

edwith boostcourse

Quiz 2 해설

Quiz

정답 Q 2 (단일선택 , 10점)

torch.Tensor 에 대한 설명으로 다음 중 옳지 않은 것은?

- 다차원 배열(Multi-dimensional Matrix)을 처리하기 위한 가장 기본이 되는 자료형(Data Type)이다.
- NumPy의 ndarray와 거의 비슷하지만, GPU 연산을 함으로써 computing power를 극대화 시킬 수 있다.
- 기본텐서 타입(Default Tensor Type)으로 32비트의 부동소수점(Float) torch.FloatTensor으로 되어있다.
- **torch.Tensor의 사칙연산은 torch.Tensor 간이나 torch.Tensor와 Python의 Scala 값 뿐만 아니라 Numpy의 ndarray와도 연산이 가능하다.**
- NumPy의 ndarray 와 마찬가지로 브로드캐스팅(Broadcasting) 적용이 가능하다.

이전 페이지 이어서, 자료형을 변경하는 방법입니다.

```
[3]  a = a.to('cuda')
     b = torch.tensor(b, device='cuda')
     a
     b
```

```
tensor([[0.4622, 0.8301, 0.6370],
        [0.1913, 0.0604, 0.7270]], device='cuda:0')
```

```
[4]  a = a.to('cpu')
     a
```

```
tensor([[0.0328, 0.1121, 0.6612],
        [0.7770, 0.8825, 0.2265]], dtype=torch.float64)
```

Quiz 3 해설

Quiz

1 2 3 4 5 6 7 8 9 10

정답 Q 3 (단일선택 , 10점)

아래 코드는 간단한 Linear Regression을 구현한 것이다.

```
import torch
import torch.nn as nn
import torch.nn.functional as F

# 데이터
x_train = torch.FloatTensor([[1], [2], [3]])
y_train = torch.FloatTensor([[1], [2], [3]])

# 모델 초기화
W = torch.zeros(1, requires_grad=True)  # ------------------1
b = torch.zeros(1, requires_grad=True)

# optimizer 설정
optimizer = torch.optim.SGD([W, b], lr=0.01)

nb_epochs = 1000

# 모델 학습
for epoch in range(nb_epochs + 1):

    hypothesis = x_train * W + b

    cost = torch.mean((hypothesis - y_train) ** 2)  # ------2

    optimizer.zero_grad()   # ------------------------------3
    cost.backward()  # -------------------------------------4
    optimizer.step()  # ------------------------------------5
```

다음페이지로 이어집니다.

edwith boostcourse

Quiz 3 해설

Quiz

다음 중 **옳지 않은** 것은?

#1(1번 주석, 10행)에서 requires_grad=True 는 학습할 것이라고 명시함으로써, gradient를 자동적으로 계산하라는 뜻이다.

#2(2번 주석, 23행)는 cost를 계산하는 과정으로, cost = F.mse_loss(hypothesis, y_train) 또는 cost = nn.MESLoss (hypothesis, y_train)으로 대체될 수 있다.

#3(3번 주석, 25행)는 PyTorch에서 backpropragation 계산을 할 때마다 gradient 값을 누적시키기 때문에, gradient를 0으로 초기화 해주기 위한 것이다.

#4(4번 주석, 26행)는 gradeint를 계산하겠다는 의미이다.

#5(5번 주석, 27행)는 다음 epoch으로 넘어가라는 뜻이다.

- ○ #1
- ○ #2
- ○ #3
- ○ #4
- ● #5

1 2 **3** 4 5 6 7 8 9 10

5번은 학습에 필요한 매개변수인 parameter(W, b)들을 업데이트하며 가설인 H(x) 를 개선하겠다는 뜻입니다.

edwith boostcourse

Quiz 4 해설

Quiz

1 2 3 4 5 6 7 8 9 10

정답 Q 4 (단일선택 , 10점)

로지스틱 회귀 모델(Logistic Regression Model)을 이용하여 이진분류모델(A Binary Classifier)를 만들려고 한다.

```
1   import torch
2   import torch.nn as nn
3
4   .
5   .
6   # 로지스틱 회귀 모델
7   class LogisticRegression(nn.Module):
8
9       def __init__(self):
10          super(LogisticRegression, self).__init__()
11          self.linear = nn.Linear(input_size, 1)
12
13      def forward(self, x):
14          ##### 활성화 함수 #####
15          y_pred = None
16          ######################
17          return y_pred
18
19  BinaryClassifierModel = LogisticRegression()
20
21  ##### 손실 함수 ######
22  criterion = None
23  ####################
24
25  .
26  .
27
```

다음페이지로 이어집니다.

edwith boostcourse

Quiz 4 해설

Quiz

```
27
28   hypothesis = BinaryClassifierModel(x_train)
29   loss = crieterion(hypothesis, y_train)
30
31   .
32   .
```

다음 중 모델의 손실 함수(Loss Function)와 출력 층(Output Layer)의 활성화 함수(Activation Function)의 최적의 조합으로 알맞은 것은?

	활성화 함수	손실함수
1	`y_pred= nn.ReLU(self.linear(x))`	`criterion = nn.BCELoss()`
2	`y_pred= nn.ReLU(self.linear(x))`	`criterion = nn.MSELoss()`
3	`y_pred= nn.Sigmoid(self.linear(x))`	`criterion = nn.MSELoss()`
4	`y_pred= nn.Sigmoid(self.linear(x))`	`criterion = nn.BCELoss()`
5	`y_pred = nn.Softmax(self.linear(x))`	`criterion = nn.BCELoss()`

- ○ 1
- ○ 2
- ○ 3
- ● 4
- ○ 5

1 2 3 **4** 5 6 7 8 9 10

일반적으로 분류모델을 만들 때, Cross-Entropy Loss와 함께 Softmax 함수를 활성화 함수(activation function)로 마지막 단에 사용합니다. 그 이유는 모든 출력 노드에 대해서 확률로 만들어 주기 때문입니다. 하지만, 이진분류모델을 만들 때에는 하나의 출력 노드와 Sigmoid 함수 조합을 사용할 수도 있는데, 이는 연산이 간단하고, 출력 노드 개수가 2인 Softmax와 Cross-Entropy Loss 조합과 같은 기능을 하기 때문입니다.

edwith boostcourse

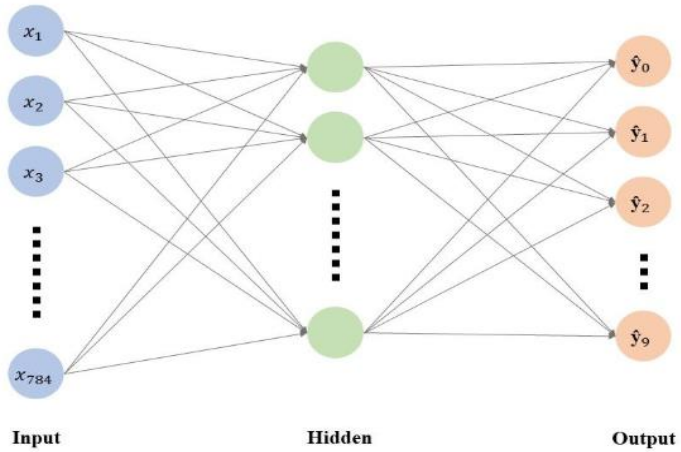
Quiz 5 해설

Quiz

1 2 3 4 5 6 7 8 9 10

정답 Q 5 (단일선택 , 10점)

다음과 같이 512개의 hidden layer 뉴런으로 구성된 Multilayer Perceptron(MLP) 모델을 이용하여, 28*28의 MNIST 데이터를 10개의 class로 구분하는 모델을 만드려고 한다.



Multilayer Perceptron (MLP)

```
N = None
M = None
H = None

class MLP(nn.Module):
    def __init__(self, N):
```

다음페이지로 이어집니다.

edwith boostcourse

Quiz 5 해설

Quiz

```
        self.layer1 = nn.Sequential(
            nn.Linear(M, H),
            nn.BatchNorm1d(H),
            nn.ReLU()
        )
        self.layer2 = nn.Sequential(
            nn.Linear(H, N)
        )

    def forward(self, x):
        x = x.view(x.size(0), -1)
        x_out = self.layer1(x)
        x_out = self.layer2(x_out)
        return nn.softmax(x_out)
```

위 코드에서 정수 N, M, H에 들어갈 숫자의 조합으로 알맞은 것은?

- ○ N: 10, M: 512, H: 28*28
- ● **N: 10, M: 28*28, H: 512**
- ○ N: 28*28, M: 10, H: 512
- ○ N: 28*28, M: 512, H: 10

1 2 3 4 **5** 6 7 8 9 10

- **N** : Class의 개수로 10 입니다.
- **M** : Input의 크기로 28*28 MNIST 이미지 크기가 들어갑니다.
- **H** : Hidden Layer 뉴런의 개수로, 512가 들어갑니다.

edwith boostcourse

Quiz 6 해설

Quiz

정답 Q 6 (단일선택 , 10점)

신경망 모델링을 위한 Hyperparameter 에 대한 설명으로 옳은 것은?

- ○ Epoch : 하나의 Mini-batch를 한 번 훈련시켰을 때, 1 Epoch을 돌았다고 한다.
- ● **Batch Size : 하나의 Mini-batch의 크기(Mini-batch의 데이터 개수). 즉, 전체 Dataset 크기를 Mini-batch의 개수로 나눈 것.**
- ○ Iteration : 전체 Training Dataset을 한 번 훈련시켰을 때, 1 Iteration을 돌았다고 한다.
- ○ Learning Rate : Learning Rate가 클수록 Local Minimum에 빠질 위험이 크다

1 2 3 4 5 **6** 7 8 9 10

Epoch 와 Iteration 설명이 바뀌었습니다.

- **Epoch :** 전체 Training Dataset을 한 번 훈련시켰을 때, 1 Epoch을 돌았다고 합니다.
- **Iteration :** 하나의 Mini-batch를 한 번 훈련시켰을 때, 1 Iteration을 돌았다고 합니다.

Learning rate가 작을수록 Local Minima에 빠질 위험이 커 적당한 조정이 필요합니다.

edwith boostcourse

Quiz 7 해설

Quiz

정답 Q 7 (단일선택 , 10점)

다음 문구의 진위 여부를 판별하세요.
"우리가 모델을 학습시킬 때, 데이터가 많을수록 더 좋은 모델이 된다. 하지만, 많은 양의 데이터를 가져와 한 번에 학습시키기엔 너무 느리고 저장하는 데에 문제가 있다. 따라서, 이를 효율적으로 학습시키기 위해 전체 데이터를 균일하게 Mini-Batch로 나누어서 학습하게 되고, 이 때 overfitting을 막아주기 위해 Epoch 마다 데이터가 학습되는 순서를 바꾸어 준다."

- ● True
- ○ False

1 2 3 4 5 6 **7** 8 9 10

학습을 위해 DataLoader로 data를 가져올 때 Shuffle=True로 설정해주어 dataset을 섞어줍니다.

edwith boostcourse

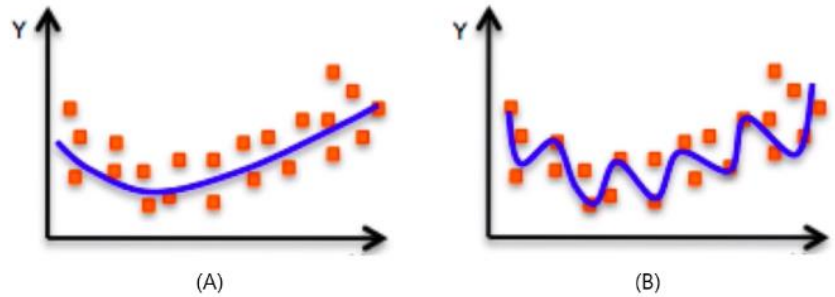
Quiz 8 해설

Quiz

1 2 3 4 5 6 7 **8** 9 10

정답 Q 8 (다중선택 , 10점)

예측 모델에 대한 결과로, (B)와 같은 결과를 가져왔다. 이를 해결하여 (A)와 같이 더 일반화된 예측을 위해 취할 수 있는 방법으로 알맞은 것을 모두 고르시오.



(A) (B)

- ☐ Data 양을 더 줄인다.
- ☐ Feature의 숫자를 늘린다.
- ☑ Early Stopping: Validation Loss가 더 이상 낮아지지 않을 때 학습을 중단한다.
- ☑ 활성화 함수(Activation Function) 앞에 Batch Normalization을 적용시켜준다.
- ☑ Dropout 방법을 적용시켜준다.

Early Stopping, Batch Normalization, Dropout 기법들은 모델 일반화에 도움이 됩니다.

edwith boostcourse

Quiz 9 해설

Quiz

정답 Q 9 (단일선택 , 10점)

다음 문구의 진위 여부를 판별하세요.
"비선형의(Non-linear) 활성화 함수(Activation Function)를 쓰는 이유는, 활성화 함수가 선형(Linear)일 경우 퍼셉트론(Perceptron)이 여러 개라도, 즉, 층(Layer)을 깊게 쌓더라도 층이 한 개인 경우와 차이가 없기 때문이다."

- (●) True
- ( ) False

1 2 3 4 5 6 7 8 **9** 10

선형함수인 $h(x) = cx$ 를 활성화 함수로 사용한 3층 네트워크를 떠올려 보세요. 이를 식으로 나타내면 다음과 같습니다.

$$y(x) = h(h(h(x)))$$

이는 실은 $y(x) = ax$ 와 똑같은 식입니다. $a = c^3$이라고만 하면 끝이죠. 즉, 은닉층이 없는 네트워크로 표현할 수 있습니다. 뉴럴 네트워크에서 층을 쌓는 혜택을 얻고 싶다면 활성화 함수로는 반드시 비선형 함수를 사용해야 합니다.

- 해당문구는 "<밑바닥부터 시작하는 딥러닝> - 사이토 고키" 에서 발췌했습니다.

edwith boostcourse

Quiz 10 해설

Quiz

정답 Q 10 (단일선택 , 10점)

다음 문구의 진위 여부를 판별하세요.
"학습된 모델의 성능을 Test할 때, torch.Tensor의 requires_grad를 True로 만들어 gradient를 계산하여 업데이트를 해주어야 한다."

- ○ True
- ● False

1 2 3 4 5 6 7 8 9 10

검증 혹은 테스트 단계에서는 오차 역전파(backpropagation)를 통해 경사(gradient)를 계산할 필요가 없기 때문에, torch.no_grad()를 이용하여 requires_grad를 False로 만들어주어 메모리를 낭비하지 않도록 합니다.

이제 프로젝트를 진행하세요!