Week 3 Assignment Preview

미래연구소 12기 3주차

본격적으로 NN 구현하는 과제여서 어려우실까봐 준비했습니다.

참고하며 과제하시면 수월하실 것입니다.

모르는 부분이 있으면 꼭 바로 질문 남겨주시기 바랍니다.

0. Intro

1) 문제점	2) 해결	3) 명심할 사항
1> 딥러닝은 어느 정도 큰 틀을 배워야 코드로 구현할 수 있는데 벌써부터 과제로 하나의 모델을 만들게 강요한다.	**1> 그래서 빈칸 문제이다.**	**1> 빈칸만 채울 수 있으면 충분합니다.** (빈칸 이외의 중요한 부분은 제가 preview와 review 때 안내합니다.)
2> 어렵다. (전체 코드를 이해하는 사람 1명도 없을 것이고 다 풀어 오는 사람도 절반 정도밖에 안 될 것입니다.)	**2> 그래서 문제 내에 설명과 힌트가 자세하게 서술되어 있습니다.**	**2> 잘 안 풀릴 때, 문제 부분을 자세하게 읽어보면 풀리는 경우가 절반 가량 됩니다. 문제 잘 읽어 봅시다.**
3> 실제 딥러닝 코드와 다르다. (원래 keras, tensorflow, pytorch 등을 사용하여 딥러닝을 구현한다.)	**3> 과제에 부담 갖지 말고 배운 내용을 구현하며 복습하는 관점과 python을 익힌다는 관점에서만 이 과제를 접근하기 바랍니다.**	**3> 못 푼다고 좌절할 것 없고 최대한 질문하면서 해보시기 바랍니다.** (다 못 풀어도 됩니다.) (답지 필요하시면 드립니다.)

0. Intro 4) 과제를 잘 해결하는 법

1) 문제와 함수 내의 주석 부분을 천천히 정독하시는 게 좋습니다. 정독만으로 힌트를 찾아서 풀 수 있는 문제들이 많습니다. (답을 대놓고 적어드린 경우도 있습니다.)

2) 모르는 건 빈칸에 내지 않습니다. 혹시 모르는 게 나온 것 같다면 문제 혹은 주석에서 충분히 설명을 합니다.

3) 실행(**shift + enter**)버튼을 처음부터 빠짐없이 누르지 않아서 **error**가 나는 경우가 있습니다. **error** 발생 시 위에서부터 다시 한 번 빠짐없이 실행버튼을 눌러주세요. 의외로 많은 문제들이 해결됩니다.

4) 문제 내에 함수들이 많습니다. **python** 함수 정의에 대해서만 다시 한 번 공부를 하시는 걸 추천 드립니다. 또한 리스트 자료형과 딕셔너리 자료형도 공부 더 해주세요.

5) 더불어 함수가 많기 때문에 잘 모르는 코드가 보인다면 그것이 위에서 정의한 함수인지 찾아보는 것도 도움이 됩니다. (위에서 정의한 함수를 아래에서 사용하는 **case**가 빈번하게 등장합니다.)

0. Intro 5) Cat classification

1> 수업시간에 배운 **binary classification** 입니다. **x? y?**

2> **data preprocess (Week 2 1**강에서 어떤 작업을 했었는지?)

3> **y hat**이 어떤 값이 나오면 얼마로 예측?

1. Import Packages

1) 중요한 라이브러리: pandas, matplotlib, numpy

1> pandas: csv 파일을 다룸, data 전처리에 가장 많이 쓰임 (8~9주차에 등장) (당분간 과제에는 등장X)

2> matplotlib: jupyter에 그림과 그래프를 그려주며 2가지 case에 많이 쓰임 (2가지 case 기억하기)

3> numpy: 실제 딥러닝에 많이 쓰이지는 않지만 많은 라이브러리의 기초가 된다.

2) 나머지: scipy, PIL, h5py

1> 특정 case에만 쓰이며 실제 프로젝트에도 위 3개보다 덜 쓰입니다.

2. Data 1) 업로드

```
# Loading the data (cat/non-cat)
train_set_x_orig, train_set_y, test_set_x_orig, test_set_y, classes = load_dataset()
```

1> x? y? classes?

2> train set: 실제 딥러닝 모델이 학습(gradient descent)하는데 사용되는 data set

3> test set: 학습한 data의 성능을 test 하는데 사용되는 data set
(우리가 그동안 배운 data는 train set이며 train set에 대해서만 생각하시면 됩니다.)

4> train_set_x_orig에는 training data(shape=num_px, num_px, 3)가 m개 있다.
=> train_set_x_orig.shape = (m, num_px, num_px, 3)

2. Data 1) 업로드

```
# Loading the data (cat/non-cat)
train_set_x_orig, train_set_y, test_set_x_orig, test_set_y, classes = load_dataset()
```

5> _orig의 의미:
문제를 만든 사람이 이미지가 딥러닝 모델에 들어가기 위해
픽셀들을 일렬로 작업하지 않은 **original image**인 상태라는 의미를 주려고
변수명을 이렇게 부여함.

=> 그 말은 **data preprocess** 과정이 필요하다는 얘기
(하지만 자세히 배운 내용이 아니어서 빈칸에 많은 힌트를 주고 있는 걸 문제에서 확인할 수 있습니다.**)**

2. Data 2) data 확인

```
# Example of a picture
index = 25
plt.imshow(train_set_x_orig[index])
print ("y = " + str(train_set_y[:, inc
```

training data가 총 209개이므로
index를 0~208까지 넣어보며 셀 실행해보시면 data들이 많이 있는 걸 볼 수 있습니다.

1> matplotlib.pyplot(plt)의 첫 번째 용도: data가 잘 업로드 되었나 확인하는 과정

(더불어 data를 소개하는 과정이기도 합니다.)

(프로젝트 하실 때 꼭 이 과정을 넣어주시기 바랍니다.)

(plt는 당장 몰라도 됩니다. 9주차에 설명합니다.)

Question 1

문제: 아래의 값에 적당한 값을 할당하시오:

- m_train (number of training examples)
- m_test (number of test examples)
- num_px (= height = width of a training image)

hint: `train_set_x_orig.shape` = (m_train, num_px, num_px, 3). For instance, you can access `m_train` by writing `train_set_x_orig.shape[0]`.

```
### START CODE HERE ### (≈ 3 lines of code)
m_train = train_set_x_orig.shape[0]
m_test = test_set_x_orig.shape[0]
num_px = train_set_x_orig.shape[1]
### END CODE HERE ###
```

1) 이미지(train_set_x_orig, test_set_x_orig)는 shape이 저런 꼴이라고 합니다.
강의에 나온 내용이지만 지금 당장 몰라도 되고 암기하는 것이 이 문제의 의도가 아닙니다.

2) numpy에서 맨 앞의 index를 줄 수 있느냐만 묻는 것이며
푸는 사람 입장에서 어려울 수도 있을까봐 답을 아예 **hint** 부분에 적어놓았습니다.

3) index는 앞에 있는 것부터 0, 1, 2, … 입니다. m_train은 0을 num_px는 1을 의미

2. Data

Data preprocess

딥러닝이라는 거대한 함수에
x와 **y**를 매칭시키기 적합하게 만드는 모든 과정을 의미하며

이 과제에서는 **1) flatten 2) normalize**를 볼 것이며

1) 는 **Course 1 week 2**에서 가볍게 보았고
2) 는 **Course 2 week 1**에서 배울 예정입니다.(정확한 용어도 다시 정리합니다.)

2. Data 3) data preprocess

Question 2

문제: training and test data sets을 reshape해서 images of size (num_px, num_px, 3)가 (num_px ∗ num_px ∗ 3, 1) 형태로 되게 한다.

hint: A trick when you want to flatten a matrix X of shape (a,b,c,d) to a matrix X_flatten of shape (b∗c∗d, a) is to use:

```
X_flatten = X.reshape(X.shape[0], -1).T      # X.T is the transpose of X
```

```
# Reshape the training and test examples

### START CODE HERE ### (≈ 2 lines of code)
train_set_x_flatten =
test_set_x_flatten =
### END CODE HERE ###
```

1> reshape은 제가 가르쳐드린 내용입니다. 강의 자료를 확인해보시거나 아래 사이트를 참고하시기 바랍니다. (https://rfriend.tistory.com/345)

2> 이 또한 어려울 수 있어 답을 거의 공개하다시피 했습니다.
(물론 저런 힌트없이 제가 가르쳐드린 내용과 문제 상황을 잘 인지해서 스스로 풀어내면 더 좋습니다.)

2. Data 3) data preprocess

```
train_set_x = train_set_x_flatten/255.
test_set_x = test_set_x_flatten/255.
```

1> 픽셀의 값은 **0~255**사이이며 **255**로 나눠서 **0~1**사이가 되게 하는 작업입니다.

2> Course 2 week 1에서 배울 내용입니다.

3> 배우지 않은 내용은 빈칸에 잘 안 냅니다.
그래서 빈칸만 채워넣으실 수만 있으면 됩니다.

3. 기타 풀이에 관한 detail tip 1) 기본

1> 문제 푸실 때 logistic regression 5가지 과정을 상기하며 풀면 도움이 됩니다.

(다음 과정에서 어떤 코드를 써야하는지 힌트를 얻을 수 있기 때문입니다.)

3. 기타 풀이에 관한 detail tip 2) 수식에 대한 얘기

1> 수식 같은 경우 문제에 출제될 때 문제에 다 일일이 서술합니다.

=> 수식을 암기하는 것은 앤드류 응 교수님의 의도가 아닙니다.

Question 5

문제: cost function과 gradient를 계산하고 있는 `propagate()` 함수를 완성하시오.

hints: 아래 식들을 numpy를 통해 구현하시오. (이 식을 외우지 않아도 됩니다. 구현할 줄만 알면 됩니다.)

- (forward propagation) $A = \sigma(w^T X + b) = (a^{(1)}, a^{(2)}, \ldots, a^{(m-1)}, a^{(m)})$
- (compute cost function) $J = -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log(a^{(i)}) + (1 - y^{(i)}) \log(1 - a^{(i)})$
- Here are the two formulas you will be using:

$$\frac{\partial J}{\partial w} = \frac{1}{m} X(A - Y)^T$$
$$\frac{\partial J}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} (a^{(i)} - y^{(i)})$$

3. 기타 풀이에 관한 detail tip 3) 함수의 빈번한 등장

```
# GRADED FUNCTION: propagate

def propagate(w, b, X, Y):
```

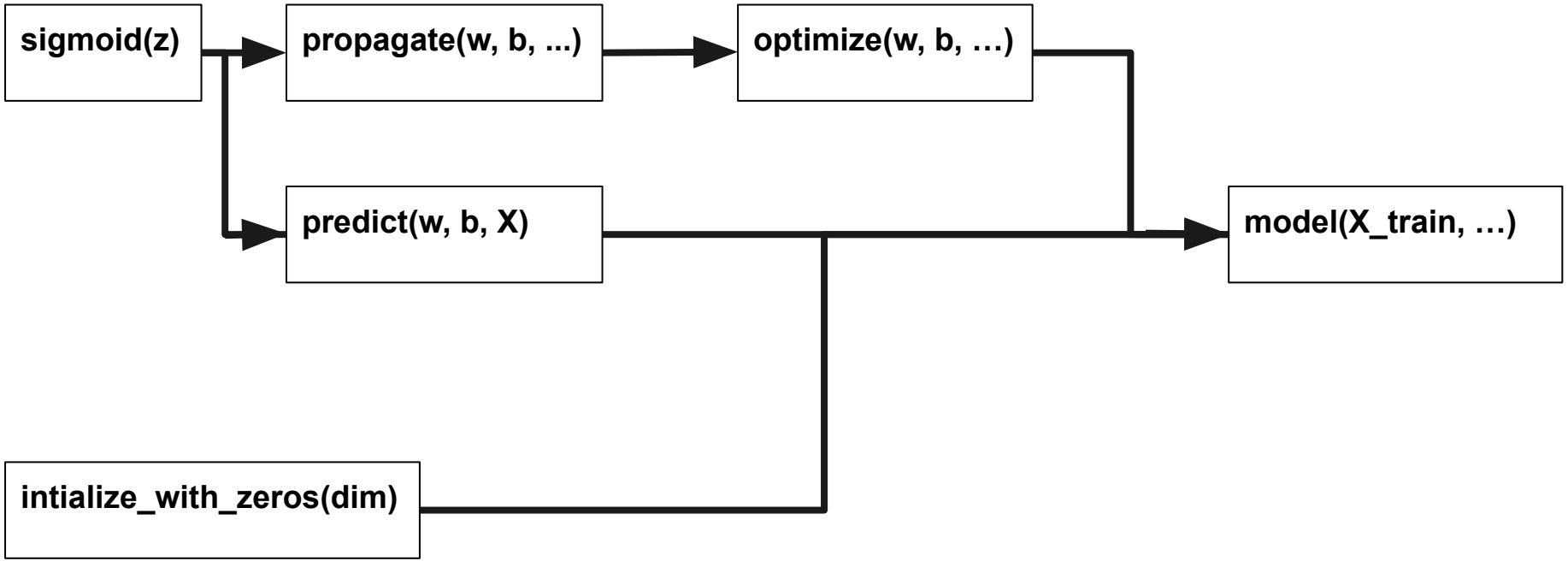
```
    assert(dw.shape == w.shape)
    assert(db.dtype == float)
    cost = np.squeeze(cost)
    assert(cost.shape == ())

    grads = {"dw": dw,
             "db": db}

    return grads, cost
```

```
# GRADED FUNCTION: optimize

def optimize(w, b, X, Y, num_iterations, learning_rate, print_cost = False):
    """
```

```
    for i in range(num_iterations):


        # Cost and gradient calculation (≈ 1-4 lines of code)
        ### START CODE HERE ###
        grads, cost = None
        ### END CODE HERE ###
```

답이 뭘까요?

3. 기타 풀이에 관한 detail tip 3) 함수의 빈번한 등장



4. Accuracy 99.5%? 66%?

1) 나쁜 것인가?

2) 왜 나쁜가?

3) 해결책?

스스로 고민해보시고 다음시간에 뵙겠습니다.
(다음 시간에 답을 말씀드리겠습니다.)

Hint: Course 1 week 1 lec 4: 딥러닝이 발전할 수 있었던 이유