Deep learning

자주 발생하는 error와 설계 방법

https://youtu.be/C3XUYQLaVtc

자주 발생하는 에러 top 5 (주관적...)

설계 팁 (주관적..)

자주 발생하는 에러 1- Colab error

- **경로**

  : drive 마운트해서 사용할 때, 경로 복사 사용 (그냥 입력해서 하면 에러날 때 있음)

- **GPU 할당**

  : Colab pro 먼저 좋은 GPU 할당해서 일반 계정으로는 GPU 할당 끊길 때 있음

- **Import error**

  : !pip install ~ 해서 설치

- **keras, tf.keras import error**

  optimizer를 예로 들면,

  from keras.optimizers import Adam 에러 뜸

  from tensorflow.keras.optimizers import Adam 는 가능

  → tf 2.0에서는 tf.keras 를 사용 권장

```
ImportError: cannot import name 'Adam' from 'keras.optimizers' (/usr/local/lib/python3.7/dist-packages/keras/optimizers.py)
```

자주 발생하는 에러 2 – Shape / Type error

- Data shape 안 맞춰줄 때
  레이어에 입력되는 데이터 차원 안 맞을 때 발생

```
ValueError: Input 0 of layer sequential_43 is incompatible with the layer: expected axis -1 of input shape to have value 3 but received input with shape (None, 101)
```

```
ValueError: Input 0 of layer sequential_3 is incompatible with the layer: : expected min_ndim=4, found ndim=2. Full shape received: (None, 101)
```

- 신경망 Input data로 float type의 array 사용해야 하는데, list 그냥 넣을 때 type error
  list는 shape 못 씀 (이 외에도 자주 발생)
  → np.asarray(list)

```
ValueError: Failed to find data adapter that can handle input: (<class 'list'> containing values of types {"<class 'float'>"}), <class 'numpy.ndarray'>
```

```
AttributeError: 'list' object has no attribute 'shape'
```

자주 발생하는 에러 3 – Nan, Inf error

inf : 무한대
NaN : 실수나 복소수가 아닌 값

- **Nan, Inf**

  1. data에 None type 있을 경우

  → assert not np.any(np.isnan(x))으로 확인 가능

  2. categorical_crossentropy

  → 1) 예측 값에 매우 작은 값 (1e-8)을 더해주기

  2) 학습률을 낮춰야 함 (수렴하는 속도가 빠른 손실 함수라서 학습률이 너무 크면 안 줄어들고 발산해버림)

  3) sparse categorical crossentropy 로 바꿔서 해결한 적 있음 ([1,2,3,…1])

  3. learning rate 너무 큰 경우

  → lr이 너무 크면, 가중치가 크게 변하면서 발산해버림

  → 0.001 이하로 낮춰보면 해결될 때 있음

  4. Numerical Exception : division 0, log(0), sqrt(0) 등이 있는 경우

  → 해당 연산 부분에 작은 값을 더해주거나, 웬만하면 라이브러리 사용..

  5. 데이터 범위

  → -1~1이면 0~1 로 범위 변경해보면 해결될 때 있음

자주 발생하는 에러 4 - data, label, loss

- **네트워크가 동작하더라도 데이터와 레이블 간의 관계가 없을 경우에도 학습이 되지 않음**
- 예를 들어 강아지와 고양이 분류 문제에서 label을 0,1로 한 후, loss를 mean squared error를 사용하면 안 됨

  → categorical crossentropy이 적합

  → 회귀 문제는 label이 수치적으로 관계가 있어야 함 (수치형 데이터 o, class x)
- 데이터 의미와 해결하고자 하는 문제를 고려

자주 발생하는 에러 5 – oom error

- **GPU 터지면 oom (Out of memory) error 발생**

  → 사이즈 큰 배열 사용하거나, 배치크기가 너무 클 경우 발생

  → 더 큰 GPU 사용하거나 해당 부분 줄이면 됨

설계 팁 1

- **일단 동작하는지 실행**
  1. 적은 data 사용해서 빨리 실행해 봄
  2. data shape, layer shape 확인
  3. 수치 불안정 방지 위해 라이브러리 사용 (tf, keras, np …)

     → exp(x) 같은 경우 지수적으로 증가해서 라이브러리 안 쓸 경우 값이 커지면 overflow 생김

     (수식 그대로 구현해도 안 되는 경우가 있음)
  4. optimizer, loss function 잘 선택된 건지 확인 – 학습 동작하는지
  5. model.summary()로 전체 구조 확인

설계 팁 1

- **작은 모델부터 실행한 후, 하나씩 추가 / 수정**

  1. 기본 Layer 사용

  : Dense (keras), Linear(Pytorch)

  2. Activation

  : ReLu 사용 (빠르고 성능 좋은 기본 활성화 함수)

  3. Optimizer (Adam(lr : 3e-4))

  : 일반화 성능이 가장 좋은 Adam optimizer, learning rate는 3e-4가 좋다고 유명

  4. Regularization x : 과적합 방지에 중요하지만 일단 빼고 시작

Hyperparameter가 모델에 영향을 미치는 정도

- **학습률, 손실함수, 레이어 사이즈가 가장 큰 영향**
- 최적화 함수 및 배치사이즈의 경우는 영향이 적음
- 영향이 큰 hyperparameter부터 수정하면 좋을 것 같음
- 실제로 튜닝할 때,
  학습률, 손실 함수, 레이어 유닛 수를 주로 바꾸고
  그 다음 레이어 추가/제거, 정규화 등을 수정했던 것 같음

Hyperparameter	Approximate sensitivity
Learning rate	High
Learning rate schedule	High
Optimizer choice	Low
Other optimizer params (e.g., Adam beta1)	Low
Batch size	Low
Weight initialization	Medium
Loss function	High
Model depth	Medium
Layer size	High
Layer params (e.g., kernel size)	Medium
Weight of regularization	Medium
Nonlinearity	Low

설계 팁 2

- **Colab 사용 시, 최대한 셀 단위, 함수로 분할해서 작성**

  만약 데이터 로드부터 학습 및 추론까지 한 셀에 돌리면

  1. 학습 후 데이터 변경, 학습 과정 중간에 오류, 추론 과정에서의 오류 등이 발생할 때 다시 처음부터 실행
  2. 하이퍼 파라미터 튜닝을 통한 최적화 → 일반화 성능 향상

     따라서 모델 구조 바꾸는 일이 많음

  → 그냥 한 셀에 돌릴 경우 학습을 계속 해야해서 시간이 오래 걸림

설계 팁 3

- **Data 확인을 가장 먼저**

  1. data 내용

     : 어떤 feature들을 사용하는지, 어떤 데이터로 무슨 작업을 수행하려는지, 값 범위 …

     → 어떤 네트워크를 사용할 지 고려, 데이터 정규화 (-1 ~ 1)

  2. data shape

     : 데이터 차원, 개수 등의 shape 중요

     → 데이터 차원 (이미지의 channel, csv 데이터의 경우 column)

     차원은 너무 많으면 복잡하고 학습도 잘 안 됨 (PCA, AE 등을 통해 차원 축소 가능)

     → 모든 데이터는 형태가 동일해야 함

     → 사용하려는 네트워크에 맞지 않을 경우 reshape (Reshape, Lambda 레이어도 있음)

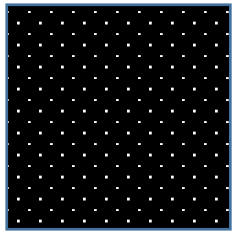
  3. type

     : data int, float 형 확인

     : list는 array로 변경 ; (arr = np.asarray(list))

feature

	바퀴 수	제조사
data1	2	A
data2	4	B



Channel → Gray scale : 1, RGB : 3

설계 팁 4

- **네트워크 구성**

  여러 모델로 하이퍼 파라미터 수정을 계속 반복하다보면 변수랑 값이랑 위치랑 헷갈려서 꼬이는 경우가 많았음..

  1. data shape 사용

  2. 하이퍼 파라미터 한 군데에 모아서 바꾸기

  → params dictionary에 하이퍼 파라미터 설정

  3. 반복되는 레이어 구조는 for문으로 작성

  → 보기 쉬움

https://colab.research.google.com/drive/1hTViarQWekmMEiDsZG46vaosS37wsnqQ#scrollTo=mRsG1WuQG6AW

설계 팁 4

- **4. Reshape, Lambda layer 사용**

  → 네트워크 내부에서 텐서 형태나 값 수정

5. for문으로 학습 반복 가능

→ 하이퍼 파라미터도 바꿀 수 있음

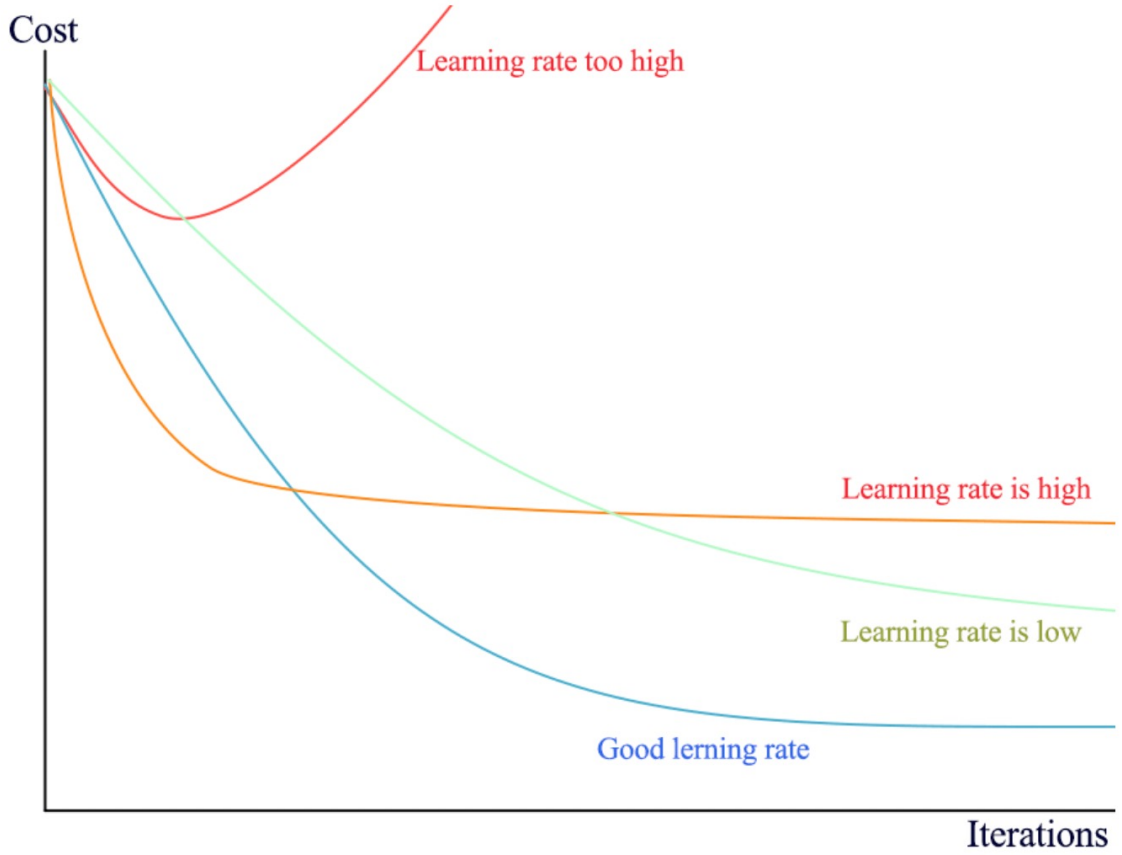
: 예를 들어, AE의 latent vector를 1~10차원 중에 뭐가 가장 정확도가 높은지 확인하고 싶은 경우

```
for i in range(1,10):
    latent = i
    acc_test(latent, ~)
```

로 해서 해당 레이어 유닛 설정하여 확인 가능 (Dense(latent,activation – ‘relu’))

Learning rate

- 학습률은 3~10배 단위로 낮추면서 실험
- 손실 **증가**할 경우, 손실이 **지그재그 패턴**일 경우 (중간에 치솟음)
  → lr이 너무 높음
- 손실이 **충분히 감소하지 않을 경우**
  → lr이 낮음
- 보통 1 ~ 1e-7 사이 (0.0001~0.001 정도 쓰는 듯)
- 좋은 기본값은 3e-4



감사합니다