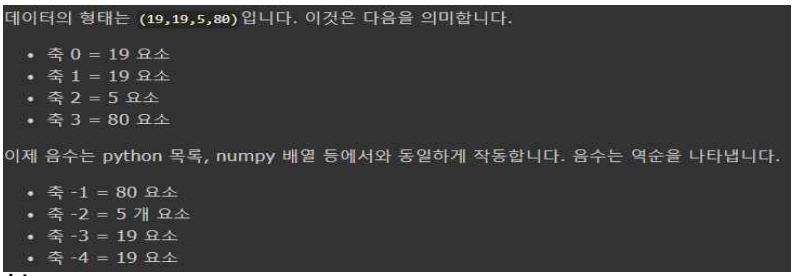
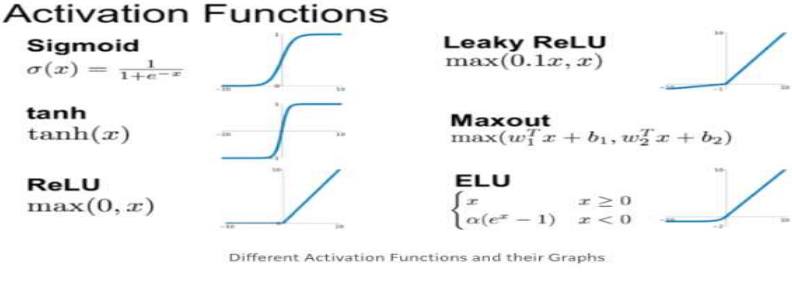
〈 TensorFlow 튜토리얼 〉

1. 데이터 로드 및 사전 처리

- CSV 데이터 로드
(https://www.tensorflow.org/tutorials/load_data/csv?hl=ko)

해당 튜토리얼은 파일에서 csv 데이터를 로드하는 방법에 대해 설명하는 예제. 데이터는 타이타닉 승객 목록에서 가져온 것으로 연령, 성별, 티켓등급 등의 특성을 기반으로 승객의 생존 가능성을 예측함.	
① 필요 라이브러리 설치	해당 없음(tensorflow와 numpy 설치되어 있으면 됨)
참 고	**numpy set_printoptions() 함수 사용법?** precision 옵션은 소숫점 자리 설정 suppress 옵션은 지수 같은 형태를 제거할지 말지 결정 글로벌 적용이기 때문에 한 번 호출하면 쭉 적용됨 (참고 : https://sevity.tistory.com/32) **tf.keras.utils.get_file 함수의 파일 저장위치?** 인터넷의 파일을 로컬 PC의 ~/.keras/datasets에 다운로드 **jupyter notebook에서 shell 사용하기?** linux 명령어를 그대로 쓰되, 맨 앞에 **!** 입력해 사용 (참고 : https://greeksharifa.github.io/references/2019/01/26/Jupyter-usage/) **axis=-1 의미?** 반환할 데이터를 인덱스의 마지막 축에서 가져오도록  참고[illegible]) **Sigmoid 함수?(딥러닝에서 Activation Function의 한 종류)** S자와 유사한 완만한 커브 형태를 보이는 대표적인 Logistic 함수 모든 실수 입력 값을 0~1 사이의 미분 가능한 변수로 변환(비선형) binary classification에 적절한 함수(일정 값 기준 0,1 여부 구분)  (참고 : https://reniew.github.io/12/ & http://taewan.kim/post/sigmoid_diff/)

2. 데이터 로드 및 사전 처리

- Numpy 데이터 로드
(https://www.tensorflow.org/tutorials/load_data/numpy?hl=ko)

<table>
<tr><td colspan="2">해당 튜토리얼은 tf.data.Dataset을 통해 NumPy 배열에서 데이터를 로드하는 예제.</td></tr>
<tr><td>① 필요 라이브러리 설치</td><td>해당 없음(tensorflow와 numpy 설치되어 있으면 됨)</td></tr>
<tr><td>참 고</td><td>
.npz 파일이란?
여러 개의 list(numpy array)를 한 번에 저장하기 위한 포맷
.npy 파일은 하나의 numpy array를 저장하기 위한 포맷
(참고 : https://jangjy.tistory.com/330)

dataset suffle과 batch?
suffle = 한 번 epoch가 끝나고 고정된 버퍼 크기로 데이터 섞기
overfitting(과적합)을 피하기 위해 매우 중요함.
(완전 랜덤하게 섞이려면 입력된 데이터 크기보다 큰 수를 입력해야 함)
batch = 데이터를 읽어 올 개수 지정
<pre>BATCH_SIZE = 64
SHUFFLE_BUFFER_SIZE = 100

train_dataset = train_dataset.shuffle(SHUFFLE_BUFFER_SIZE).batch(BATCH_SIZE)
test_dataset = test_dataset.batch(BATCH_SIZE)</pre>
참고 : https://hiseon.me/data-analytics/tensorflow-dataset/)

ReLU 함수?(딥러닝에서 Activation Function의 한 종류)
0보다 작은 값이 나온 경우 0을 반환하고,
0보다 큰 값이 나온 경우 그 값을 그대로 반환하는 함수
sigmoid, tanh 함수와 비교 시 학습 속도가 훨씬 빠름(구현 간단)
hidden layer에는 relu를 적용하고, 마지막 output layer에서만
sigmoid 함수를 적용하면 정확도가 더 많이 ↑
요즘 추세는 sigmoid(치명적 오류 2가지) 보다 ReLU를 많이 사용
참고
</td></tr>
</table>

3. 데이터 로드 및 사전 처리

- pandas.DataFrame 데이터 로드 (https://www.tensorflow.org/tutorials/load_data/pandas_dataframe?hl=ko)

<table>
<tr><td colspan="2">해당 튜토리얼은 tf.data.Dataset을 통해 pandas.DataFrame에 데이터를 로드하는 예제. 데이터는 Cleveland Clinic Foundation(https://archive.ics.uci.edu/ml/datasets/heart+Disease)에서 가져온 것으로(csv파일) 환자의 심장 질환 여부를 예측함.(이진 분류작업)</td></tr>
<tr><td>① 필요 라이브러리 설치</td><td>해당 없음(tensorflow와 pandas 설치되어 있으면 됨)</td></tr>
<tr><td>② dataset 다운로드</td><td>csv_file = tf.keras.utils.get_file('heart.csv', 'https://storage.googleapis.com/applied-dl/heart.csv')

위 코드를 통해 다운로드 받은 파일은 내 서버의 ~/.keras/datasets/heart.csv 에 위치함.</td></tr>
<tr><td>참 고</td><td>numpy, pandas, matplotlib 비교?
numpy = 리스트, 배열, 매트릭스 연산(같은 데이터 형식)을 위한 라이브러리
pandas = 데이터 프레임(다른 데이터 형식)을 위한 라이브러리
matplotlib = 데이터 시각화 라이브러리(곡선, 원, 막대 등)
(참고 : https://deancode-ws.tistory.com/13, https://software-creator.tistory.com/22)

Tensorflow에서 dataset을 사용하는 방법?
모델에 데이터를 제대로 공급하려면 입력 파이프라인을 만들어 GPU로 들어올 데이터를 멈추지 않게 해야 함.
Tensorflow는 Dataset이라는 build-in-API(내장 API) 제공하며 이를 통해 최적화된 입력 파이프라인을 만들어 모델을 학습, 평가, 테스트 할 수 있는 빠르고 강력한 방법을 사용 할 수 있음.

< dataset의 사용 3단계 >
1) 데이터 불러오기 : 사용하려는 data로부터 instance 생성
- numpy, csv, tensor, placeholder 에서 불러오기 등
- from_tensor_slices(features, labels) 함수 등 다양
2) Iterator(반복자) 생성하기
- one shot, initializable, reinitializable, feedable
3) 데이터 사용하기 : 생성된 iterator를 통해 dataset 사용

< 이 외 데이터 변형에 쓰이는 함수들 >
· batch : 주어진 크기로 dataset을 자동 처리 가능
· repeat : dataset을 몇 번 반복 사용할지 지정(epoch로 제어)
· shuffle : 설정된 epoch마다 dataset 섞기(overfitting 피할 수 있음)
· map : dataset의 각 멤버에 사용자 지정 함수 적용 가능

(참고 : https://cyc1am3n.github.io/2018/09/13/how-to-use-dataset-in-tensorflow.html
https://datascienceschool.net/view-notebook/5771410 3a75c43ed9a7c95f96135f0adf/)</td></tr>
</table>

4. 데이터 로드 및 사전 처리

- 이미지 데이터 로드
(https://www.tensorflow.org/tutorials/load_data/images?hl=ko)

해당 튜토리얼은 tf.data를 사용해 image dataset을 로드하는 예제. 이 예제의 dataset은 디렉토리 당 하나의 이미지 클래스를 사용.	
① 필요 라이브러리 설치	기본(tensorflow==2.1.0, numpy, matplotlib) PIL = python에서 이미지를 처리하고 핸들링 하기 위한 패키지 ! pip3 install Pillow 참고:
② dataset 다운로드	data_dir = tf.keras.utils.get_file(origin='https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz', fname='flower_photos', untar=True) 위 코드를 통해 다운로드 받은 파일은 내 서버의 *~/.keras/datasets/flower_photos* 에 위치함.
참 고	**image 파일을 로드하는 방법은 크게 두 가지로 나뉨** ***1) keras.preprocessing 사용*** - 쉽게 쓸 수 있음 - 느림 - 세밀한 제어가 힘듦 - 다른 tensorflow와 잘 통합되지 않음 ***2) tf.data 사용*** - cache를 사용해 keras.preprocessing보다 빠름 - 디테일한 제어가 가능 - 쓰기 복잡함(함수 많이 만들어야 함)

5. 데이터 로드 및 사전 처리

- 텍스트 데이터 로드
(https://www.tensorflow.org/tutorials/load_data/text?hl=ko)

<table>
<tr><td colspan="2">해당 튜토리얼은 tf.data.TextLineDataset를 사용해 text file을 로드하는 예제.
TextLineDataset 텍스트 파일에서 dataset을 만들도록 디자인 되어 각 예제는 원본 파일의 텍스트를 줄이며, line 기반 text data에서 유용.
이 예제에서는 동일한 작업에 대한 3가지 다른 번역인 Homer's Illiad를 사용하고 한 줄의 텍스트로 번역기를 식별하는 모델 학습.</td></tr>
<tr><td>① 필요 라이브러리 설치</td><td>기본(tensorflow==2.0.0 설치되어 있으면 됨)
$ pip3 install tf-nightly</td></tr>
<tr><td>참 고</td><td>tf-nightly란?
tensorflow의 일일 빌드 버전으로 개발 중인 버전.
tensorflow가 stable한 버전이라면,
tf-nightly는 unstable 하지만 최신 기능과 버그 패치가 포함됨.
tensorflow나 tf-nightly 중 마지막에 설치된 버전이 호출됨!
(참고 : https://devlog.jwgo.kr/2019/10/25/what-is-gpg-file-extension/)

lambda 함수?
한 줄 함수, 익명함수 라고 부르기도 함.
쓰고 버리는 일시적인 함수로 생성된 곳에서만 필요한 경우.
return이 포함되어 있지 않으며, map(), filter(), reduce() 등의 함수와 매우 유용하게 쓰일 수 있음.
lambda 인자 : 표현식 (참고 : https://wikidocs.net/64)

LSTM 네트워크?
Long Short-Term Memory의 약자로 장-단기 메모리를 의미
인간의 뇌 구조처럼 한 번 본 내용은 기억하고 학습하는 네트워크
(참고 : http://colah.github.io/posts/2015-08-Understanding-LSTMs/)</td></tr>
</table>

6. 데이터 로드 및 사전 처리

- unicode 데이터 로드
(https://www.tensorflow.org/tutorials/load_data/unicode?hl=ko)

<table>
<tr><td colspan="2">해당 튜토리얼은 tensorflow에서 유니코드 문자열을 표현하고, 조작하는 방법에 대한 예제
자연어 처리모델은 다른 문자 집합을 갖는 다양한 언어를 다루게 되는데,
유니코드는 거의 모든 언어의 문자를 표현 할 수 있는 표준 인코딩 시스템으로 쓰임
Unicode는 전 세계의 모든 문자를 컴퓨터에서 일관되게 표현하고 다룰 수 있도록 설계된 산업 표준
전 세계의 문자를 key와 value로 1:1 매칭한 코드로 16년 기준 $2^{20}+2^{16}$개 만큼 존재
(참고 사이트 : https://miaow-miaow.tistory.com/37)</td></tr>
<tr><td>① 필요 라이브러리 설치</td><td>해당 없음(tensorflow만 설치되어 있으면 됨)</td></tr>
<tr><td>참 고</td><td>이모티콘 unicode 테이블
😊 이모티콘이 '₩xf0₩x9f₩x98₩x8a'과 매칭됨.
https://apps.timwhitlock.info/emoji/tables/unicode

유니코드 표현
1) scalar(string) = 문자열을 사용해 인코딩
2) vector(int32) = 위치마다 개별 코드 포인트를 의미

표현 간의 변환
1) tf.strings.unicode_decode = 인코딩된 string 스칼라를 코드 포인트의 벡터로 변환
2) tf.strings.unicode_encode = 코드 포인트의 벡터를 인코드된 string 스칼라로 변환
3) tf.strings.unicode_transcode = 인코드된 string 스칼라를 다른 인코딩으로 변환

오프셋(offset)이란?
동일 오브젝트 안에서 오브젝트 처음부터 주어진 요소나 지점까지의 변위차를 나타내는 정수형으로 상대주소 의미
ex) A = 'abcdef' 일 때 'c' 문자는 A의 시작점에서 2 offset
참고:

unicode script
하나의 코드포인트 집합을 의미
ex) 'Б'가 키릴(Cyrillic) 스크립트라는 것을 알고 있으면 이 문자가 포함된 텍스트는 아마도 (러시아어나 우크라이나어 같은) 슬라브 언어라는 것을 알 수 있음
script는 ICU(International Components for Unicode)의 UScriptCode 값과 일치하는 int32 값</td></tr>
</table>

7. 데이터 로드 및 사전 처리

- TF.Text 사용법
(https://www.tensorflow.org/tutorials/tensorflow_text/intro?hl=ko)

해당 튜토리얼은 Tensorflow Text의 유용한 기능(text 관련 class 및 기능)에 대한 예제.	
① 필요 라이브러리 설치	기본(tensorflow==2.0.0 설치되어 있으면 됨) $ pip3 install tensorflow-text
참 고	5~6번을 좀 더 쉽게 쓸 수 있도록 tensorflow에서 제공하는 Text API(6번 예제의 연장선 개념으로 생각) **tokenizer?** 문자열을 공백, 탭, 구분자를 기준으로 분리 (한글로 따지면 형태소 단위로 분리하는 느낌) **Wordshape?** 특정 속성이 있는지 확인하는 것 (대문자, 대문자로 시작, 온점, 숫자여부 등) **N-grams & Sliding Window?** n개의 연속적인 단어 나열을 의미 text.Reduction.STRING_JOIN, STRING_SUM, STRING_MEAN 위 3가지 종류의 축소 방법이 가장 많이 사용됨.

8. 데이터 로드 및 사전 처리

- TF.Record 및 tf.Example
(https://www.tensorflow.org/tutorials/load_data/tfrecord?hl=ko)

해당 튜토리얼은 TFRecord(일련의 이진 레코드를 저장하기 위한 간단한 형식)와 tf.Example(또는 protobuf) 메시지 작성 및 구문 분석 사용법에 대한 예제. 데이터를 효율적으로 읽기 위해 직렬화(serialization) 및 파일 set으로 만들기 및 데이터 전처리 캐싱 필요. protobuf는 구조화된 데이터의 효율적인 직렬화를 위한 라이브러리. tf.Example은 TFX(TensorFlow Extended)와 같은 상위 레벨 API에서 사용됨.	
① 필요 라이브러리 설치	기본(tensorflow, numpy, IPython 설치) $ pip install -q tf-nightly
참 고	번역된 블로그 (http://solarisailab.com/archives/2603)