83. Mini-batch Training, GPU Training

```
#for 80
import re
import string

#for 81
import nltk
nltk.download('punkt')
from nltk import stem
import itertools
import csv
import numpy as np
import tensorflow as tf
from tensorflow import keras
import pandas as pd
#import sys

#for 82
import matplotlib.pyplot as plt

# for 83
import time

print('80. turing words into numeric IDs')
#st.header('80. Turning words into numeric IDs')

TRAINING_NUM = 0

#@st.code
def preprocessing(text): # same function in problem 50
    text = text.lower()
    text = re.sub('[0-9]+', '', text)
    text = "".join([i for i in text if i not in string.punctuation])

    tokens = nltk.word_tokenize(text)
    stemmer = stem.PorterStemmer()
    stem_tokens = [stemmer.stem(token) for token in tokens]
    return " ".join(stem_tokens)

#@st.code
def texts_to_id(texts):
    word_counter: dict[str, int] = {}
    for title in texts:
        for word in title.split():
            word_counter.setdefault(word, 0)
            word_counter[word] += 1
    map = {
        k: i + 1 for i, (k, v) in # 順番に取り出したとき、keyにはi+1を与えてね
        enumerate([
        (k, v) for (k, v) in #keyとvalueについて
        sorted([(k, v) for (k, v) in word_counter.items()], key=lambda x:x[1], reverse=True) #出現頻度でsortされたkeyとvalueの中の
        if v >= 1
        ])
    }

    def mapper(title: str) -> list[int]:
        return[
            map.get(word, 0) #これは何をしている？
            for word in title.split()
        ]
    ids = texts.apply(mapper)
    return ids

titles = pd.read_csv('./train.txt', sep='\t', quoting=csv.QUOTE_NONE) #quoteを無視する
titles = titles['title'][:10]
print(titles)

titles = titles.apply(preprocessing)
titles_id = texts_to_id(titles)
print(titles_id)
#st.write(title_1_id)
```

```
print('81. Prediction with an RNN')
def get_data(training_num: int):

    ## load the data
    train = pd.read_csv('./train.txt', sep='\t', quoting=csv.QUOTE_NONE)
    if training_num:
        print(f'現在テスト用にTraining Dataは {training_num} 個のみ使用しています')
        train = train[:training_num]
    X_train = train['title']

    valid = pd.read_csv('./valid.txt', sep='\t', quoting=csv.QUOTE_NONE) #validation data for problem 82
    X_valid = valid['title']

    category_dict = {'b': 0, 't': 1, 'e': 2, 'm': 3}
    y_train = train['category'].map(category_dict)
    y_valid = valid['category'].map(category_dict)

    ## convert texts to the sequence of ids
    X_train = X_train.apply(preprocessing)
    X_valid = X_valid.apply(preprocessing)

    ## use a part of texts_to_ids()
    word_counter: dict[str, int] = {}
    for title in X_train:
        for word in title.split():
            word_counter.setdefault(word, 0)
            word_counter[word] += 1

    map = {
        k: i + 1 for i, (k, v) in
        enumerate([
        (k, v) for (k, v) in
        sorted([(k, v) for (k, v) in word_counter.items()], key=lambda x : x[1])
        if v >= 1
        ])
    }
    def mapper(title: str) -> list[int]:
        return [
            map.get(word, 0)
            for word in title.split()
        ]
    X_train = X_train.apply(mapper)
    X_valid = X_valid.apply(mapper)

    return X_train.tolist(), X_valid.tolist(), y_train.tolist(), y_valid.tolist(), len(map) + 1

X_train, X_valid, y_train, y_valid, id_count = get_data(TRAINING_NUM)
print(f'2回以上出現した単語は {id_count}種類')

def get_onehots(training_num: int, id_count: int):
    max_len = max(list(map(len, X_train)) + list(map(len, X_valid)))
    for ids in X_train:
        ids += [0 for _ in range(max_len - len(ids))]

    for ids in X_valid:
        ids += [0 for _ in range(max_len - len(ids))]

    X_train_onehot = tf.one_hot(X_train, depth=id_count)
    X_valid_onehot = tf.one_hot(X_valid, depth=id_count)
    return X_train_onehot, X_valid_onehot

X_train_onehot, X_valid_onehot = get_onehots(TRAINING_NUM, id_count)

def get_model() -> keras.Model:
    model = keras.models.Sequential([
        keras.layers.SimpleRNN(4, input_shape=[None, id_count]),
        keras.layers.Dense(4, activation="softmax"),
    ])
        # units(dimentionality of output space), input_shape(optional)
        # dimention of d_w = len(word_id_dic), d_n = 4
    #
    model.summary()
    return model

model = get_model()

print('学習前の予測')
res = model.predict(X_train_onehot[:3])
print(res)
```

```
# 82. Training with Stochastic Gradient Descent
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

## training
start = time.time()

history = model.fit(
    X_train_onehot,
    np.array(y_train),
    epochs=10,
    validation_data=(X_valid_onehot, np.array(y_valid)),
    batch_size=32 #problem 83
)

elapsed = time.time() -start
print(f'GPUでは{elapsed: .4f}秒')

print('学習後の予測!')
res = model.predict(X_train_onehot[:3])
print(res)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
80. turing words into numeric IDs
0    UPDATE 1-Ousted American Apparel CEO Charney r...
1    'Mad Men' premiere draws 2.3 mln, lowest seaso...
2    E-cigarettes CAN help people kick the habit: S...
3    PRECIOUS-Gold rebounds on bargain hunting afte...
4    Why so shy? Lea Michele keeps her head down af...
5    Europe Stocks Rise With Emerging Markets as Bo...
6    "Angelina Jolie: Honorary Damehood Means ""A G...
7                    The Momentum of Freedom (Passover)
8    Ebersman Departs Facebook on Top After Post-IP...
9    Former Anglo Irish Bank Executives Guilty on 1...
Name: title, dtype: object
0                          [6, 7, 8, 9, 10, 11, 12, 13]
1                 [14, 15, 16, 17, 18, 19, 20, 21, 22]
2      [23, 24, 25, 26, 27, 3, 28, 29, 30, 31, 32, 33]
3                        [34, 35, 1, 36, 37, 2, 38, 39]
4    [40, 41, 42, 43, 44, 45, 4, 46, 47, 2, 48, 4, ...
5                  [51, 5, 52, 53, 54, 55, 56, 57, 58]
6             [59, 60, 61, 62, 63, 64, 65, 66, 67, 68]
7                                  [3, 69, 70, 71, 72]
8                    [73, 74, 75, 1, 76, 2, 77, 5, 78]
9                  [79, 80, 81, 82, 83, 84, 1, 85, 86]
Name: title, dtype: object
81. Prediction with an RNN
2回以上出現した単語は 10159種類
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 4)                 40656

 dense (Dense)               (None, 4)                 20

=================================================================
Total params: 40,676
Trainable params: 40,676
Non-trainable params: 0
_________________________________________________________________
学習前の予測
1/1 [==============================] - 3s 3s/step
[[0.2207805  0.28435072 0.25577584 0.23909295]
 [0.22761717 0.27145353 0.25486842 0.24606086]
 [0.20329218 0.28316936 0.2686422  0.24489626]]
Epoch 1/10
334/334 [==============================] - 15s 36ms/step - loss: 1.0335 - accuracy: 0.6091 - val_loss: 0.8049 - val_accuracy: 0.7451
Epoch 2/10
334/334 [==============================] - 10s 31ms/step - loss: 0.6169 - accuracy: 0.8131 - val_loss: 0.7087 - val_accuracy: 0.7549
Epoch 3/10
334/334 [==============================] - 11s 33ms/step - loss: 0.4880 - accuracy: 0.8519 - val_loss: 0.6825 - val_accuracy: 0.7796
Epoch 4/10
334/334 [==============================] - 11s 34ms/step - loss: 0.4096 - accuracy: 0.8721 - val_loss: 0.7352 - val_accuracy: 0.7751
Epoch 5/10
```

```
334/334 [==============================] - 12s 35ms/step - loss: 0.3627 - accuracy: 0.8840 - val_loss: 0.7054 - val_accuracy: 0.7856
Epoch 6/10
334/334 [==============================] - 12s 35ms/step - loss: 0.3191 - accuracy: 0.8933 - val_loss: 0.7304 - val_accuracy: 0.7789
Epoch 7/10
```

Double-click (or enter) to edit