

✓ Homework #4

Double Click here to edit this cell

- Name : 김현동
- Student Id : 201901208
- Submission date : 2023.12.13

최종 제출 시 모든 셀을 다시 실행한 뒤 제출하세요.

✓ RNN Scratch 구현

Data load & preprocessing

```
!pip install unidecode

Collecting unidecode
  Downloading Unidecode-1.3.7-py3-none-any.whl (235 kB)
  235.5/235.5 kB 2.1 MB/s eta 0:00:00
Installing collected packages: unidecode
Successfully installed unidecode-1.3.7

from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

%cd /content/drive/MyDrive/HW4 RNN 모델 구현

/content/drive/MyDrive/HW4 RNN 모델 구현

# 해당 셀을 실행하세요.
import os
import random
from string import ascii_letters
from IPython.display import Image
import torch
from torch import nn
import torch.nn.functional as F
from unidecode import unidecode

_ = torch.manual_seed(505)

# 경로를 수정 후 해당 셀을 실행하면 됩니다.

# data 형태를 확인하시길 바랍니다.

data_dir = "./data/names" # data file의 경로를 자신의 경로에 맞게 수정하여 사용

lang2label = {
    name.split(".")[0]: torch.tensor([i], dtype=torch.long)
    for i, name in enumerate(sorted(os.listdir(data_dir)))
} # file 이름을 통해 label을 만들고 dictionary로 만들어주는 과정입니다.

unidecode("Ślusàrski")

char2idx = {letter: i for i, letter in enumerate(ascii_letters + " .,:;-")}

num_letters = len(char2idx)
num_langs = len(lang2label)
```

✓ One-hot encoding

- data에 대해 one-hot encoding을 진행합니다.
- 인자로 주어지는 name은 각 언어에 대한 성씨입니다. ex) Kim, Lee, Jang, Cho
- name에 대해 one-hot-encoding을 진행하는 function을 작성하세요.
- pd.get_dummies / sklearn의 library 사용 금지. → 사용시 0점

더블클릭 또는 Enter 키를 눌러 수정

해당 셀을 실행하여 결과를 확인하세요.

```
tensor([[[[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 0., 0., 0., 0., 0., 0., 0.]],  
  
         [[0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 0., 0., 0., 0., 0., 0., 0.]],  
  
         [[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,  
           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 0., 0., 0., 0., 0., 0., 0.]],  
  
         [[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 0., 0., 0., 0., 0., 0., 0.]],  
  
         [[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,  
           0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

Your Output must be

```
# 해당 셀을 실행하세요.
from sklearn.model_selection import train_test_split
```

```
train_idx, test_idx = train_test_split(
    range(len(target_langs)),
    test_size=0.1,
    shuffle=True
)
```

```
train_dataset = [
    (tensor_names[i], target_langs[i])
    for i in train_idx
]
```

```
test_dataset = [
    (tensor_names[i], target_langs[i])
    for i in test_idx
]
```

```
print(f"Train: {len(train_dataset)}")
print(f"Test: {len(test_dataset)}")
```

Train: 18063
Test: 2007

- ✓ RNN Model

- RNN 모델을 scratch로 구현합니다.
- 밑줄친 부분을 지우고 해당 부분에 올바른 코드를 작성하면 됩니다.
- RNN_Scratch에서 주어진 init, forward 외에 추가를 하면 안됩니다.
- 올바른 코드를 작성하였다면 하이퍼파라미터 튜닝을 통해 성능을 향상합니다.
- 하이퍼파라미터를 수정하여 정확도를 비교하는 과정 또한 있어야합니다.
- Inference 전에 셀을 필요한만큼 추가하여 하이퍼파라미터를 변경해가며 결과값을 비교하면 됩니다.(최소 4회 이상의 코드와 결과를 보여주어야합니다.)
- num_epochs, print_interval은 2와 3000으로 고정입니다.
- 학습하며 train_loss의 learning curve를 출력하여야합니다. 해당 부분에 대해서 필요한 코드는 추가하시면 됩니다.

```

import torch
import torch.nn as nn

class RNN_Scratch(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(RNN_Scratch, self).__init__()
        self.input_size = input_size
        self.output_size = output_size
        self.hidden_size = hidden_size

        # 입력에서 은닉층으로의 가중치와 편향
        self.wxh = nn.Parameter(torch.randn(input_size,hidden_size))
        self.bh = nn.Parameter(torch.randn(hidden_size))
        self.whh = nn.Parameter(torch.randn(hidden_size,hidden_size))

        # 은닉층에서 출력층으로의 가중치와 편향
        self.wh_output = nn.Parameter(torch.randn(hidden_size,output_size))
        self.bo = nn.Parameter(torch.randn(output_size))

    def forward(self, x, state = None):
        # 은닉 상태 초기화(0으로 초기화)
        if state == None :
            state = torch.zeros(x.size(0),self.hidden_size)
        else :
            if state is not None and isinstance(state, tuple):
                state, = state

        # RNN forward 연산
        for i in range(x.size(0)):
            state = torch.tanh(torch.matmul(x[i, :],self.wxh)+torch.matmul(state,self.whh)+self.bh)
            out = torch.matmul(state,self.wh_output) + self.bo

        return out, state

input_size = 59
hidden_size = 64
output_size = 18
learning_rate = 0.001

model = RNN_Scratch(input_size,hidden_size,output_size)

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(),lr=learning_rate)

num_epochs = 2
print_interval = 3000
train_losses= []

for epoch in range(num_epochs):
    random.shuffle(train_dataset)
    total_loss = 0
    for i, (name, label) in enumerate(train_dataset):
        hidden = torch.zeros(1,hidden_size) # 은닉 상태 0으로 초기화

        for char in name :
            output, hidden = model(char,hidden)
            loss = criterion(output,label)
            optimizer.zero_grad()
            loss.backward()
            nn.utils.clip_grad_norm_(model.parameters(), 1)
            optimizer.step()

        total_loss += loss.item()

    if (i + 1) % print_interval == 0:
        print(
            f"Epoch [{epoch + 1}/{num_epochs}], "
            f"Step [{i + 1}/{len(train_dataset)}], "
            f"Loss: {loss.item():.4f}"
        )

    train_losses.append(loss.item())

Epoch [1/2], Step [3000/18063], Loss: 3.8659
Epoch [1/2], Step [6000/18063], Loss: 1.0496
Epoch [1/2], Step [9000/18063], Loss: 8.8359
Epoch [1/2], Step [12000/18063], Loss: 1.0851
Epoch [1/2], Step [15000/18063], Loss: 0.2409

```

```
Epoch [1/2], Step [18000/18063], Loss: 6.7238
Epoch [2/2], Step [3000/18063], Loss: 5.5538
Epoch [2/2], Step [6000/18063], Loss: 2.6940
Epoch [2/2], Step [9000/18063], Loss: 0.0020
Epoch [2/2], Step [12000/18063], Loss: 0.0010
Epoch [2/2], Step [15000/18063], Loss: 0.5941
Epoch [2/2], Step [18000/18063], Loss: 7.2466
```

✓ Accuracy

```
"""
input_size = 59
hidden_size = 256
output_size = 18
learning_rate = 0.001
optim : adam
"""

num_correct = 0
num_samples = len(test_dataset)

model.eval()

with torch.no_grad():
    for name, label in test_dataset:
        hidden_state = torch.zeros(1,hidden_size) # 은닉 상태 0으로 초기화
        for char in name:
            output, hidden_state = model(char,hidden_state)
            _, pred = torch.max(output, dim=1)
            num_correct += bool(pred == label)

print(f"Accuracy: {num_correct / num_samples * 100:.4f}%")

Accuracy: 34.0807%

"""
input_size = 59
hidden_size = 128
output_size = 18
learning_rate = 0.001
optim : adam
"""

num_correct = 0
num_samples = len(test_dataset)

model.eval()

with torch.no_grad():
    for name, label in test_dataset:
        hidden_state = torch.zeros(1,hidden_size) # 은닉 상태 0으로 초기화
        for char in name:
            output, hidden_state = model(char,hidden_state)
            _, pred = torch.max(output, dim=1)
            num_correct += bool(pred == label)

print(f"Accuracy: {num_correct / num_samples * 100:.4f}%")

Accuracy: 43.7469%
```

```

"""
input_size = 59
hidden_size = 64
output_size = 18
learning_rate = 0.01
optim : Adam
"""

num_correct = 0
num_samples = len(test_dataset)

model.eval()

with torch.no_grad():
    for name, label in test_dataset:
        hidden_state = torch.zeros(1,hidden_size) # 은닉 상태 0으로 초기화
        for char in name:
            output, hidden_state = model(char,hidden_state)
            _, pred = torch.max(output, dim=1)
            num_correct += bool(pred == label)

print(f"Accuracy: {num_correct / num_samples * 100:.4f}%")

    Accuracy: 48.2312%

"""
Accuracy 51.4200
input_size = 59
hidden_size = 64
output_size = 18
learning_rate = 0.001
optim : Adam
"""

num_correct = 0
num_samples = len(test_dataset)

model.eval()

with torch.no_grad():
    for name, label in test_dataset:
        hidden_state = torch.zeros(1,hidden_size) # 은닉 상태 0으로 초기화
        for char in name:
            output, hidden_state = model(char,hidden_state)
            _, pred = torch.max(output, dim=1)
            num_correct += bool(pred == label)

print(f"Accuracy: {num_correct / num_samples * 100:.4f}%")

    Accuracy: 55.3064%

```

✓ Learning curve

```

# learning curve 출력
import matplotlib.pyplot as plt

plt.plot(train_losses, label='Train Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()

```



▼ Inference

구글 검색 : 'nation' first name으로 검색 후, train data에 없는 이름을 사용해서 예측 진행

해당 셀을 실행하세요.

```
label2lang = {label.item(): lang for lang, label in lang2label.items()}

def myrnn_predict(name):
    model.eval()
    tensor_name = name2tensor(name)
    with torch.no_grad():
        hidden_state = torch.zeros(1,hidden_size) # 은닉 상태 0으로 초기화
        for char in tensor_name:
            output, hidden_state = model(char,hidden_state)
            _, pred = torch.max(output, dim=1)
        model.train()
    return label2lang[pred.item()]
```

myrnn_predict를 활용하여 5개의 예시를 본인이 만들어 결과 확인하기

- 아래에 예시를 본인이 생각하여 5가지 정도 만들어 예측 결과를 출력하시오.

구글 검색 : 'nation' first name으로 검색 후, train data에 없는 이름을 사용해서 예측 진행

```
myrnn_predict('Leyla') #rusian
```