Overfitting,Dropout,Weight Decay,Batch Normalization

practice 인공지능,머신러닝 · 2024. 10. 16. 06:49

Summary

Dropout : 학습중에 일부 가중치를 확률적으로 0으f로 만드는것
Dropout 쓰는이유: overfitting 막는데 도움
Overfitting: 학습데이터를 모델이 거의 외워버린 상태 => 새로운 테스트 데이터 대해 모델이 성능 낮음
Overfitting 막는 방법: 데이터 늘리기, Early stopping(validation data), Weight Decay(가중치 감쇠), Dropout, Batch Normalization

아래그림) dropout

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

아래그림)

Overfitting: 학습데이터를 모델이 거의 외워버린 상태 => 새로운 테스트 데이터 대해 모델이 성능 낮음

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

아래그림)

train data, validation data, test data

train data만 있으면 언제 멈출지 모르기때문에 validation loss 함수가 꺾이는 지점에서 모델을 선정

test data는 최종적으로 성능평가.

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

아래그림)

데이터 늘리기 - overfitting 방지

기존 overfitting 일때는 loss가 0인 보라색 그래프인데,
데이터를 늘리고 학습을 더 진행하면, train loss가 증가함
그러면 train loss 가 감소되는 방향으로 학습이 일어나서 초록색 그래프처럼 loss가 줄어드는 방향으로 그래프가 변해짐
진짜 함수인 파란색 함수랑 비슷해짐
(ex. 수학문제를 외우면 새로운 문제 못품)

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

Weight Decay

weight 크기 줄여야

출처:https://sooho-kim.tistory.com/85

> l1-norm과 l2-norm에 대하여
> 해당 포스팅은 Dive into Deep Learning과 Mathematics for Machine Learning을 참고하여 정리한 글입니다. 이 영화를 알고 계시다면 저와 비슷한 세대임은 분명하겠군요. (학생 때 영화관에서 봤던 기억이 있었
> sooho-kim.tistory.com

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

L p-norm

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$$

$$\|x\|_1 = \sum |x_i|$$

벡터 크기

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{\frac{1}{2}} = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$

L2-norm

$\|\vec{v}\|_3$

$$\vec{x} = (\underset{x_1}{4}, \underset{x_2}{3})$$

$$\sqrt{4^2+3^2} = 5$$

$$\|\vec{x}\|_2 = \sqrt{\sum |x_i|^2} = \sqrt{4^2+3^2} = \sqrt{25} = 5$$

$$\vec{x} = (x_1\ x_2\ x_3\ x_4 \cdots)$$

$$\sqrt{\sum_{i=1}^{n} |x_i|^2} = \sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2 + \cdots}$$

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

아래그림) 가중치 크기가 적어지면 함수가 복잡해지지 않아서 overfitting이 줄어든다.
L는 새로운 L'으로 커지지만 loss를 줄어드는 방향으로 가서 overfitting이 줄어든다.

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

아래그림)

초기상태는 underfitting,

너무 overfitting(학습이 많이해서 거의 외운상태) 하면 train loss가 거의 0 됨(loss가 적다고 좋은게 아님)

파란색선은 학습이 잘된상태

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

2. l_p-norm

일반적인 식으로 표현하는 l_p-norm은 다음과 같은 식으로 표현됩니다.

$$||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$$

이 식에서 p=1이라면, l_1-norm, p=2라면, l_2-norm이 됩니다.

Batch Normalization

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

normalization

m(평균)= 0 , 표준편차=1로 만들어준다.(원래 표준편차는 1이 같거나 아니었는데 1로만듬)

batch normalization

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

batch normalization

- 데이터의 batch를 normalization한다.
- 정리를 하니 overfitting 이 된다.

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

Code SUMMARY

성씨를 보고 국적을 맞추는 Multi class classification (class 는 국가종류)

mlp모델을 학습시켜서 해결

초기화

```
#Data 다운받는것

# 만약 코랩에서 실행하는 경우 아래 코드를 실행하여 전처리된 라이트 버전의 데이터를 다운로드하세요.
!mkdir data                                                    #Data 다운받는것
```

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

```
!wget https://git.io/JtaFp -O data/download.py
!wget https://git.io/Jtabe -O data/get-all-data.sh
!chmod 755 data/get-all-data.sh
%cd data
!./get-all-data.sh
%cd ..
```

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

4-2_Classifying_Surnames_with_an_MLP.ipynb

```python
if args.reload_from_files:
    # 체크포인트에서 훈련을 다시 시작
    print("로딩!")
    dataset = SurnameDataset.load_dataset_and_load_vectorizer(args.surname_csv,
                                                              args.vectorizer_file)
else:
    # 데이터셋과 Vectorizer 만들기
    print("새로 만들기!") #dataset
    dataset = SurnameDataset.load_dataset_and_make_vectorizer(args.surname_csv)
    dataset.save_vectorizer(args.vectorizer_file)

vectorizer = dataset.get_vectorizer()
classifier = SurnameClassifier(input_dim=len(vectorizer.surname_vocab),
                               hidden_dim=args.hidden_dim,
                               output_dim=len(vectorizer.nationality_vocab))
```

```
새로 만들기!
```

```python
df = pd.read_csv("data/surnames/surnames_with_splits.csv", header=0)
df
```

	nationality	nationality_index	split	surname
0	Arabic	15	train	Totah
1	Arabic	15	train	Abboud

from argparse import Namespace.py 7

from argparse import Namespace.py > SurnameDataset > __init__

```python
class SurnameVectorizer(object):


# a = SurnameDataset(df, vectorizer)
#  dataframe을 불러와서 surname dataset instance를
# 만든다.
# a.surname_df -> surname_df
# self.surname_df = surname_df

class SurnameDataset(Dataset):
    def __init__(self, surname_df, vectorizer):
        """ #init은 class생성할때 호출되는 함수
        매개변수:
            surname_df (pandas.DataFrame): 데이터셋
            vectorizer (SurnameVectorizer): SurnameVectorizer
            객체
        """
        self.surname_df = surname_df #내부에서만
        self._vectorizer = vectorizer

        self.train_df = self.surname_df[self.surname_df.split=='train']
        self.train_size = len(self.train_df)

        self.val_df = self.surname_df[self.surname_df.split=='val']
        self.validation_size = len(self.val_df)

        self.test_df = self.surname_df[self.surname_df.
```

내부(self 맨앞에 무조건 들어가있는것)
참고, cls는 classmethod 에 들어감

4-2_Classifying_Surnames_with_an_MLP.ipynb

```python
if args.reload_from_files:
    # 체크포인트에서 훈련을 다시 시작
    print("로딩!")
    dataset = SurnameDataset.load_dataset_and_load_vectorizer(args.surname_csv,
                                                              args.vectorizer_file)
else:
    # 데이터셋과 Vectorizer 만들기
    print("새로 만들기!") #dataset
    dataset = SurnameDataset.load_dataset_and_make_vectorizer(args.surname_csv)
    dataset.save_vectorizer(args.vectorizer_file)

vectorizer = dataset.get_vectorizer()
classifier = SurnameClassifier(input_dim=len(vectorizer.surname_vocab),
                               hidden_dim=args.hidden_dim,
                               output_dim=len(vectorizer.nationality_vocab))
```

```
새로 만들기!
```

```python
df = pd.read_csv("data/surnames/surnames_with_splits.csv", header=0)
df
```

	nationality	nationality_index	split	surname
0	Arabic	15	train	Totah
1	Arabic	15	train	Abboud


```python
class SurnameDataset(Dataset):
    def __init__(self, surname_df, vectorizer):
        self.validation_size = len(self.val_df)

        self.test_df = self.surname_df[self.surname_df.split=='test']
        self.test_size = len(self.test_df)

        self._lookup_dict = {'train': (self.train_df, self.train_size),
                             'val': (self.val_df, self.validation_size),
                             'test': (self.test_df, self.test_size)}

        self.set_split('train')

        # 클래스 가중치
        class_counts = surname_df.nationality.value_counts().to_dict()
        def sort_key(item):
            return self._vectorizer.nationality_vocab.lookup_token(item[0])
        sorted_counts = sorted(class_counts.items(), key=sort_key)
        frequencies = [count for _, count in sorted_counts]
        self.class_weights = 1.0 / torch.tensor(frequencies, dtype=torch.float32)
# a = SurnameDataset(df, vectorizer)
```

함수 init을 만들어서
이 함수가 실행

google.com/search?q=relu&oq=relu&gs_lcrp=EgZjaHJvbWUyDwgAEEUYO...

Rectified Linear Unit 렉티파
ReLU는 입력값이 0보다 작
그대로 출력하는 유닛인데,

ReLU

relu 함수를 쓸까?
· 1. 계산 효율성 · 2. Gradient(기울기) 소실

ChatGPT

Click here to ask ChatGPT

Ai Tools - Become Superhuman Today!

from argparse import Namespace.py 7

from argparse import Namespace.py > SurnameClassifier > forward

```python
class SurnameClassifier(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_d
        hidden_dim (int): 첫 번째 Linear 층의 출력
        output_dim (int): 두 번째 Linear 층의 출력
        """
        super(SurnameClassifier, self).__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, output_dim)

    def forward(self, x_in, apply_softmax=False):
        """MLP의 정방향 계산

        매개변수:
            x_in (torch.Tensor): 입력 데이터 텐서
                x_in.shape는 (batch, input_dim)입니다.
            apply_softmax (bool): 소프트맥스 활성화 함수를 위한
            플래그
                크로스-엔트로피 손실을 사용하려면 False로
                지정해야 합니다.
        반환값:
            결과 텐서. tensor.shape은 (batch, output_dim)
            입니다.
        """
        intermediate_vector = F.relu(self.fc1(x_in))
        prediction_vector = self.fc2(intermediate_vector)

        if apply_softmax:
            prediction_vector = F.softmax(prediction_vector,
            dim=1)
```

fully connected layer를 지정하고

fully connected layer fc1, fc2지나고
relu activation 사용

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

colab.research.google.com/drive/13lckS_OU47911ewNURTMU9pTrm88MH...

4-2_Classifying_Surnames_with_an_MLP.ipynb

	nationality	nationality_index	split	surname
0	Arabic	15	train	Totah
1	Arabic	15	train	Abboud
2	Arabic	15	train	Fakhoury
3	Arabic	15	train	Srour
4	Arabic	15	train	Sayegh
...	...	...	...	...
10975	Vietnamese	11	test	Dinh
10976	Vietnamese	11	test	Phung
10977	Vietnamese	11	test	Quang
10978	Vietnamese	11	test	Vu
10979	Vietnamese	11	test	Ha

10980 rows × 4 columns

성씨를 보고
나라를 맞추는것

```
df.nationality.unique()
```

```
array(['Arabic', 'Chinese', 'Czech', 'Dutch', 'English', 'French',
       'German', 'Greek', 'Irish', 'Italian', 'Japanese', 'Korean',
       'Polish', 'Portuguese', 'Russian', 'Scottish', 'Spanish',
       'Vietnamese'], dtype=object)
```

```
len(df.nationality.unique())
```

```
class SurnameDataset(Dataset):
    #class를 호출해서 instance 를 만듬

    @classmethod
    def load_dataset_and_load_vectorizer(cls, surname_
    vectorizer_filepath): ···

    @staticmethod
    def load_vectorizer_only(vectorizer_filepath): ···

    def save_vectorizer(self, vectorizer_filepath): ···

    def get_vectorizer(self): ···

    def set_split(self, split="train"):
        """ 데이터프레임에 있는 열을 사용해 분할 세트를 선택합니다

        매개변수:
            split (str): "train", "val", "test" 중 하나
        """
        self._target_split = split
        self._target_df, self._target_size = self._lookup_dic
        [split]

    def __len__(self):
        return self._target_size

    def __getitem__(self, index):
        """ 파이토치 데이터셋의 주요 진입 메서드

        매개변수:
            index (int): 데이터 포인
```

google.com/search?q=relu&oq=relu&gs_lcrp=EgZjaHJvbWUyDwgAEEUYO...

Rectified Linear Unit 렉티파
ReLU는 입력값이 0보다 작
그대로 출력하는 유닛인데,

relu 함수를 쓸까?
· 1. 계산 효율성 · 2. Gradient(기울기) 소실

ChatGPT
Click here to ask ChatGPT
Ai Tools - Become Superhuman Today!

```
class SurnameClassifier(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_d
        hidden_dim (int): 첫 번째 Linear 층의 출력
        output_dim (int): 두 번째 Linear 층의 출력
    """
    super(SurnameClassifier, self).__init__()
    self.fc1 = nn.Linear(input_dim, hidden_dim)
    self.fc2 = nn.Linear(hidden_dim, output_dim)

    def forward(self, x_in, apply_softmax=False):
        """MLP의 정방향 계산

        매개변수:
            x_in (torch.Tensor): 입력 데이터 텐서
                x_in.shape는 (batch, input_dim)입니다.
            apply_softmax (bool): 소프트맥스 활성화 함수를 위한
            플래그
                크로스-엔트로피 손실을 사용하려면 False로
                지정해야 합니다.
        반환값:
            결과 텐서. tensor.shape은 (batch, output_dim)
            입니다.
        """
        intermediate_vector = F.relu(self.fc1(x_in))
        prediction_vector = self.fc2(intermediate_vector)

        if apply_softmax:
            prediction_vector = F.softmax(prediction_vector,
            dim=1)
```

fully connected layer를 지정하고

fully connected layer fc1, fc2지나고
relu activation 사용

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

vectorize는 단어를 vector화

참고로 one hot vector는 하나만 1이고,

multil hot vector는 여러개가 1

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

4-2_Classifying_Surnames_with_an_MLP.ipynb

```python
        # 정확도를 계산합니다
        acc_t = compute_accuracy(y_pred, batch_dic
        running_acc += (acc_t - running_acc) / (batch_index + 1)

        # 진행 바 업데이트
        train_bar.set_postfix(loss=running_loss, acc=running_acc,
                              epoch=epoch_index)
        train_bar.update()

    train_state['train_loss'].append(running_loss)
    train_state['train_acc'].append(running_acc)

    # 검증 세트에 대한 순회

    # 검증 세트와 배치 제너레이터 준비, 손실과 정확도를 0으로 설정
    dataset.set_split('val')
    batch_generator = generate_batches(dataset,
                                       batch_size=args.batch_size,
                                       device=args.device)
    running_loss = 0.
    running_acc = 0.
    classifier.eval()

    for batch_index, batch_dict in enumerate(batch_generator):

        # 단계 1. 출력을 계산합니다
        y_pred =  classifier(batch_dict['x_surname'])

        # 단계 2. 손실을 계산합니다
        loss = loss_func(y_pred, batch_dict['y_nationality'])
        loss_t = loss.to("cpu").item()
        running_loss += (loss_t - running_loss) / (batch_index + 1)
```

from argparse import Namespace.py 7

from argparse import Namespace.py > SurnameDataset > __getitem__

```python
class SurnameDatas

    def __len__(self):
        return self._target_size

    def __getitem__(self, index):
        """ 파이토치 데이터셋의 주요 진입 메서드

        매개변수:
            index (int): 데이터 포인트의 인덱스
        반환값:
            데이터 포인트의 특성(x_surname)과 레이블
            (y_nationality)로 이루어진 딕셔너리
        """
        row = self._target_df.iloc[index]

        surname_vector = \
            self._vectorizer.vectorize(row.surname)

        nationality_index = \
            self._vectorizer.nationality_vocab.lookup_t
            (row.nationality)
            #단어를 넣어서 몇번인덱스인지 알아서
            nationality_index에 넣는다.
        return {'x_surname': surname_vector,
                'y_nationality': nationality_index}

    def get_num_batches(self, batch_size):
        """ 배치 크기가 주어지면 데이터셋으로 만들 수 있는
        개수를 반환합니다

        매개변수:
```

4-2_Classifying_Surnames_with_an_MLP.ipynb

```python
        # 정확도를 계산합니다
        acc_t = compute_accuracy(y_pred, batch_dic
        running_acc += (acc_t - running_acc) / (batch_index + 1)

        # 진행 바 업데이트
        train_bar.set_postfix(loss=running_loss, acc=running_acc,
                              epoch=epoch_index)
        train_bar.update()

    train_state['train_loss'].append(running_loss)
    train_state['train_acc'].append(running_acc)

    # 검증 세트에 대한 순회

    # 검증 세트와 배치 제너레이터 준비, 손실과 정확도를 0으로 설정
    dataset.set_split('val')
    batch_generator = generate_batches(dataset,
                                       batch_size=args.batch_size,
                                       device=args.device)
    running_loss = 0.
    running_acc = 0.
    classifier.eval()

    for batch_index, batch_dict in enumerate(batch_generator):

        # 단계 1. 출력을 계산합니다
        y_pred =  classifier(batch_dict['x_surname'])

        # 단계 2. 손실을 계산합니다
        loss = loss_func(y_pred, batch_dict['y_nationality'])
        loss_t = loss.to("cpu").item()
        running_loss += (loss_t - running_loss) / (batch_index + 1)
```

from argparse import Namespace.py 7

from argparse import Namespace.py > SurnameDataset > __getitem__

```python
class SurnameDatas

    def __len__(self):
        return self._target_size

    def __getitem__(self, index):
        """ 파이토치 데이터셋의 주요 진입 메서드

        매개변수:
            index (int): 데이터 포인트의 인덱스
        반환값:
            데이터 포인트의 특성(x_surname)과 레이블
            (y_nationality)로 이루어진 딕셔너리
        """
        row = self._target_df.iloc[index]

        surname_vector = \
            self._vectorizer.vectorize(row.surname)
            #multihot vector로 surname_vector에 넣는다.
        nationality_index = \
            self._vectorizer.nationality_vocab.lookup_token
            (row.nationality)
            #단어를 넣어서 몇번인덱스인지 알아서
            nationality_index에 넣는다.
        return {'x_surname': surname_vector,
                'y_nationality': nationality_index}

    def get_num_batches(self, batch_size):
        """ 배치 크기가 주어지면 데이터셋으로 만들 수 있는 배치
        개수를 반환합니다

        매개변수:
```

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

4-2_Classifying_Surnames_with_an_MLP.ipynb

```python
            acc_t = compute_accuracy(y_pred, batch_dic
            running_acc += (acc_t - running_acc) / (batch_index + 1)

            # 진행 바 업데이트
            train_bar.set_postfix(loss=running_loss, acc=running_acc,
                                  epoch=epoch_index)
            train_bar.update()

        train_state['train_loss'].append(running_loss)
        train_state['train_acc'].append(running_acc)

        # 검증 세트에 대한 순회

        # 검증 세트와 배치 제너레이터 준비, 손실과 정확도를 0으로 설정
        dataset.set_split('val')
        batch_generator = generate_batches(dataset,
                                           batch_size=args.batch_size,
                                           device=args.device)
        running_loss = 0.
        running_acc = 0.
        classifier.eval()

        for batch_index, batch_dict in enumerate(batch_generator):

            # 단계 1. 출력을 계산합니다
            y_pred =  classifier(batch_dict['x_surname'])

            # 단계 2. 손실을 계산합니다
            loss = loss_func(y_pred, batch_dict['y_nationality'])
            loss_t = loss.to("cpu").item()
            running_loss += (loss_t - running_loss) / (batch_index + 1)
```

```python
class SurnameDataset(Dataset):

    def __len__(self):
        return self._target_size

    def __getitem__(self, index):
        """ 파이토치 데이터셋의 주요 진입 메서드

        매개변수:
            index (int): 데이터 포인트의 인덱스
        반환값:
            데이터 포인트의 특성(x_surname)과 레이블
            (y_nationality)로 이루어진 딕셔너리
        """
        row = self._target_df.loc[index]

        surname_vector = \
            self._vectorizer.vectorize(row.surname)
            #multihot vector로 surname_vector에 넣는다.
        nationality_index = \
            self._vectorizer.nationality_vocab.lookup_token
            (row.nationality)
            #단어를 넣어서 몇번인덱스인지 알아서
            nationality_index에 넣는다.
        return {'x_surname': surname_vector,
                'y_nationality': nationality_index}

    def get_num_batches(self, batch_size):
        """ 배치 크기가 주어지면 데이터셋으로 만들 수 있는 배치
        개수를 반환합니다

        매개변수:
```

이렇게는

이 2 dictionary에 담아서 return해준다.

훈련 반복

4-2_Classifying_Surnames_with_an_MLP.ipynb

훈련 반복

```python
[15] classifier = classifier.to(args.device)
     dataset.class_weights = dataset.class_weights.to(args.device)


     loss_func = nn.CrossEntropyLoss(dataset.class_weights)
     optimizer = optim.Adam(classifier.parameters(), lr=args.learning_rate)#adam은 optimizer,  lr=args.learning_rate->처음 lr
     scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer=optimizer, #lr=learning rate는 학습중에 바꿔주는것
                                                      mode='min', factor=0.5, # lr_scheduler는 학습중에 lr바꿔준다.
                                                      patience=1)

     train_state = make_train_state(args)

     epoch_bar = tqdm.notebook.tqdm(desc='training routine',
                                    total=args.num_epochs,
                                    position=0)

     dataset.set_split('train')
     train_bar = tqdm.notebook.tqdm(desc='split=train',
                                    total=dataset.get_num_batches(args.batch_size),
                                    position=1,
                                    leave=True)
     dataset.set_split('val')
     val_bar = tqdm.notebook.tqdm(desc='split=val',
                                  total=dataset.get_num_batches(args.batch_size),
                                  position=1,
                                  leave=True)
```

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

4-2_Classifying_Surnames_with_an_MLP.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

```
optimizer = optim.Adam(classifier.parameters(), lr=args.learning_rate)#ada
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer=optimizer, #lr=
                                                 mode='min', factor=0.5, #
                                                 patience=1)

train_state = make_train_state(args)

epoch_bar = tqdm.notebook.tqdm(desc='training routine',
                               total=args.num_epochs,
                               position=0)

dataset.set_split('train')
train_bar = tqdm.notebook.tqdm(desc='split=train',
                               total=dataset.get_num_batches(args.b
                               position=1,
                               leave=True)
dataset.set_split('val')
val_bar = tqdm.notebook.tqdm(desc='split=val',
                             total=dataset.get_num_batches(args.bat
                             position=1,
                             leave=True)

try:
    for epoch_index in range(args.num_epochs):
        train_state['epoch_index'] = epoch_index

        # 훈련 세트에 대한 순회

        # 훈련 세트와 배치 제너레이터 준비, 손실과 정확도를 0으로 설정
        dataset.set_split('train')
        batch_generator = generate_batches(dataset,
```

```
class SurnameData
    def __getitem__(self, index):
        매개변수:
            index (int): 데이터 포인트의 인덱스
```

training routine: 100% 100/100 [03:41<00:00, 1.98s/it]

split=train: 99% 119/120 [03:41<00:02, 2.75s/it, acc=51.8, epoc

split=val: 96% 24/25 [03:41<00:00, 21.07it/s, acc=45.4, epoch=9

```
# 가장 좋은 모델을 사용해 테스트 세트의 손실과 정확도를 계산합니다
classifier.load_state_dict(torch.load(train_state['model_filename']))

classifier = classifier.to(args.device)
```

4-2_Classifying_Surnames_with_an_MLP.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

```
        # 훈련 세트와 배치 제너레이터 준비, 손실과 정확도를 0으로 설정
        dataset.set_split('train')
        batch_generator = generate_batches(dataset,
                                           batch_size=args.batch_size,
                                           device=args.device)
        running_loss = 0.0
        running_acc = 0.0
        classifier.train()

        for batch_index, batch_dict in enumerate(batch_generator):
            # 훈련 과정은 5단계로 이루어집니다

            # ----------------------------------------
            # 단계 1. 그레이디언트를 0으로 초기화합니다
            optimizer.zero_grad()

            # 단계 2. 출력을 계산합니다
            y_pred = classifier(batch_dict['x_surname'])

            # 단계 3. 손실을 계산합니다
            loss = loss_func(y_pred, batch_dict['y_nationality'])
            loss_t = loss.item()
            running_loss += (loss_t - running_loss) / (batch_index + 1)

            # 단계 4. 손실을 사용해 그레이디언트를 계산합니다
            loss.backward()

            # 단계 5. 옵티마이저로 가중치를 업데이트합니다
            optimizer.step()
            # ----------------------------------------
```

학습5단계

4-2_Classifying_Surnames_with_an_MLP.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

```
                                           batch_size=args.batch_size,
                                           device=args.device)
        running_loss = 0.0
        running_acc = 0.0
        classifier.train()

        for batch_index, batch_dict in enumerate(batch_generator):
            # 훈련 과정은 5단계로 이루어집니다

            # ----------------------------------------
            # 단계 1. 그레이디언트를 0으로 초기화합니다
            optimizer.zero_grad()

            # 단계 2. 출력을 계산합니다
            y_pred = classifier(batch_dict['x_surname'])

            # 단계 3. 손실을 계산합니다
            loss = loss_func(y_pred, batch_dict['y_nationality'])
            loss_t = loss.item()
            running_loss += (loss_t - running_loss) / (batch_index + 1)

            # 단계 4. 손실을 사용해 그레이디언트를 계산합니다
            loss.backward()

            # 단계 5. 옵티마이저로 가중치를 업데이트합니다
            optimizer.step()
            # ----------------------------------------

            # 정확도를 계산합니다
            acc_t = compute_accuracy(y_pred, batch_dict['y_nationality'])
            running_acc += (acc_t - running_acc) / (batch_index + 1)
```

예측값 나오게하는단계

두개를 비교할때

CrossEntropyLoss사용label은 loss계산시 onehot 나라가하나라서)

모델예측값

label

classifier

모델예측값이 label과 비교해서 loss계산

loss가 적게되도록 classifier weight 가

update되고 예측값이 변경되면서

loss가 점점 적어짐

$W = W - lr \frac{dL}{dW}$

원래weight 미분값은 기울기 (gradient)

learning rate : 얼마만큼 갈지

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

4-2_Classifying_Surnames_with_an_MLP.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

```python
                                    batch_size=args.batch_size,
                                    device=args.device)
running_loss = 0.0
running_acc = 0.0
classifier.train()

for batch_index, batch_dict in enumerate(batch_generator):
    # 훈련 과정은 5단계로 이루어집니다

    # --------------------------------------
    # 단계 1. 그레이디언트를 0으로 초기화합니다
    optimizer.zero_grad()

    # 단계 2. 출력을 계산합니다
    y_pred = classifier(batch_dict['x_surname'])

    # 단계 3. 손실을 계산합니다
    loss = loss_func(y_pred, batch_dict['y_nationality'])
    loss_t = loss.item()
    running_loss += (loss_t - running_loss) / (batch_index + 1)

    # 단계 4. 손실을 사용해 그레이디언트를 계산합니다
    loss.backward()

    # 단계 5. 옵티마이저로 가중치를 업데이트합니다
    optimizer.step()
    # --------------------------------------

    # 정확도를 계산합니다
    acc_t = compute_accuracy(y_pred, batch_dict['y_nationality'])
    running_acc += (acc_t - running_acc) / (batch_index + 1)
```

모델예측값이랑 label과 비교해서
crossEntorpy loss계산

4-2_Classifying_Surnames_with_an_MLP.ipynb

파일 수정 보기 삽입 런타임 도구 도움말

```python
    # --------------------------------------
    # 단계 1. 그레이디언트를 0으로 초기화합니다
    optimizer.zero_grad()

    # 단계 2. 출력을 계산합니다
    y_pred = classifier(batch_dict['x_surname'])

    # 단계 3. 손실을 계산합니다
    loss = loss_func(y_pred, batch_dict['y_nationality'])
    loss_t = loss.item()
    running_loss += (loss_t - running_loss) / (batch_index + 1)

    # 단계 4. 손실을 사용해 그레이디언트를 계산합니다
    loss.backward()

    # 단계 5. 옵티마이저로 가중치를 업데이트합니다
    optimizer.step() #weight업데이트
    # --------------------------------------

    # 정확도를 계산합니다
    acc_t = compute_accuracy(y_pred, batch_dict['y_nationality'])
    running_acc += (acc_t - running_acc) / (batch_index + 1)

    # 진행 바 업데이트
    train_bar.set_postfix(loss=running_loss, acc=running_acc,
                          epoch=epoch_index)
    train_bar.update()

train_state['train_loss'].append(running_loss)
train_state['train_acc'].append(running_acc)
```

4-2_Classifying_Surnames_with_an_MLP.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

```python
    optimizer.step() #weight업데이트
    # --------------------------------------

    # 정확도를 계산합니다 #중간중간 정확도 계산후 weight업데이트반복
    acc_t = compute_accuracy(y_pred, batch_dict['y_nationality'])
    running_acc += (acc_t - running_acc) / (batch_index + 1)

    # 진행 바 업데이트
    train_bar.set_postfix(loss=running_loss, acc=running_acc,
                          epoch=epoch_index)
    train_bar.update()

train_state['train_loss'].append(running_loss)
train_state['train_acc'].append(running_acc)

# 검증 세트에 대한 순회

# 검증 세트와 배치 제너레이터 준비, 손실과 정확도를 0으로 설정
dataset.set_split('val')
batch_generator = generate_batches(dataset,
                                   batch_size=args.batch_size,
                                   device=args.device)
running_loss = 0.
running_acc = 0.
classifier.eval()

for batch_index, batch_dict in enumerate(batch_generator):

    # 단계 1. 출력을 계산합니다
    y_pred =  classifier(batch_dict['x_surname'])
```

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

4-2_Classifying_Surnames_with_an_MLP.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

```python
        # 검증 세트와 배치 제너레이터 준비, 손실과 정확도를 0으로 설정
        dataset.set_split('val')
        batch_generator = generate_batches(dataset,
                                           batch_size=args.batch_size,
                                           device=args.device)
        running_loss = 0.
        running_acc = 0.
        classifier.eval()

        for batch_index, batch_dict in enumerate(batch_generator):

            # 단계 1. 출력을 계산합니다
            y_pred =  classifier(batch_dict['x_surname'])

            # 단계 2. 손실을 계산합니다
            loss = loss_func(y_pred, batch_dict['y_nationality'])
            loss_t = loss.to("cpu").item()
            running_loss += (loss_t - running_loss) / (batch_index + 1)

            # 단계 3. 정확도를 계산합니다
            acc_t = compute_accuracy(y_pred, batch_dict['y_nationality'])
            running_acc += (acc_t - running_acc) / (batch_index + 1)
            val_bar.set_postfix(loss=running_loss, acc=running_acc,
                                epoch=epoch_index)
            val_bar.update()

        train_state['val_loss'].append(running_loss)
        train_state['val_acc'].append(running_acc)

        train_state = update_train_state(args=args, model=classifier,
                                         train_state=train_state)
```

지금부터 validation 쓰겠다.

train set - 실제 모델 학습 -> overfitting우려됨, 기출문제 정답만 외우는상태

validation set- 학습 중 확인, 모델 선정->학습중 정답만 외우는거 아닌거 점검

test - 최종 테스트

model0
model1
model2
...
modeln -> weight다른거로 서로다른모델로 분류,
다저장하면 낭비니까 고를때
validation set에서 성능이 높은것이나 loss가 낮은것을 고름

4-2_Classifying_Surnames_with_an_MLP.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

모델 훈련

헬퍼 함수

```python
def make_train_state(args):
    return {'stop_early': False,
            'early_stopping_step': 0,
            'early_stopping_best_val': 1e8,
            'learning_rate': args.learning_rate,
            'epoch_index': 0,
            'train_loss': [],
            'train_acc': [],
            'val_loss': [],
            'val_acc': [],
            'test_loss': -1,
            'test_acc': -1,
            'model_filename': args.model_state_file}

def update_train_state(args, model, train_state):
    """ 훈련 상태를 업데이트합니다.

    Components:
     - 조기 종료: 과대 적합 방지
     - 모델 체크포인트: 더 나은 모델을 저장합니다

    :param args: 메인 매개변수
    :param model: 훈련할 모델
    :param train_state: 훈련 상태를 담은 딕셔너리
```

train상태 저장용도

4-2_Classifying_Surnames_with_an_MLP.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

```python
    # 적어도 한 번 모델을 저장합니다
    if train_state['epoch_index'] == 0:
        torch.save(model.state_dict(), train_state['model_filename'])
        train_state['stop_early'] = False

    # 성능이 향상되면 모델을 저장합니다
    elif train_state['epoch_index'] >= 1:
        loss_tm1, loss_t = train_state['val_loss'][-2:]

        # 손실이 나빠지면
        if loss_t >= train_state['early_stopping_best_val']:
            # 조기 종료 단계 업데이트
            train_state['early_stopping_step'] += 1
        # 손실이 감소하면
        else:
            # 최상의 모델 저장
            if loss_t < train_state['early_stopping_best_val']: #모델이 좋아질때만 저장하고싶다.loss가 어떤값보다 적으면 저장하겠다
                torch.save(model.state_dict(), train_state['model_filename'])

            # 조기 종료 단계 재설정
            train_state['early_stopping_step'] = 0

        # 조기 종료 여부 확인
        train_state['stop_early'] = W
            train_state['early_stopping_step'] >= args.early_stopping_criteria #early_stopping_criteri = 5, 5번보다 여러번 연속으로 loss가 개선되지 않으면, 종료하겠다.

    return train_state

def compute_accuracy(y_pred, y_target):
    _, y_pred_indices = y_pred.max(dim=1)
```

w는 길어서 한줄내릴려고 쓴거

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

4-2_Classifying_Surnames_with_an_MLP.ipynb

```python
            loss = loss_func(y_pred, batch_dict['y_nationality'])
            loss_t = loss.to("cpu").item()
            running_loss += (loss_t - running_loss) / (batch_index + 1)

            # 단계 3. 정확도를 계산합니다
            acc_t = compute_accuracy(y_pred, batch_dict['y_nationality'])
            running_acc += (acc_t - running_acc) / (batch_index + 1)
            val_bar.set_postfix(loss=running_loss, acc=running_acc,
                                epoch=epoch_index)
            val_bar.update()

        train_state['val_loss'].append(running_loss) #중간중간 loss와 accuracy를 넣는다. 이걸보고 중간에 저장할지,학습을끝낼지 결정
        train_state['val_acc'].append(running_acc)

        train_state = update_train_state(args=args, model=classifier,  #update_train_state는 학습을 끝낼지 결정
                                         train_state=train_state)

        scheduler.step(train_state['val_loss'][-1])

        if train_state['stop_early']:
            break

        train_bar.n = 0
        val_bar.n = 0
        epoch_bar.update()
    except KeyboardInterrupt:
        print("Exiting loop")
```

이값이 true이면 학습을 중단

training routine: 100% 100/100 [03:41<00:00, 1.98s/it]

4-2_Classifying_Surnames_with_an_MLP.ipynb

추론

```python
def predict_nationality(surname, classifier, vectorizer):
    """새로운 성씨로 국적 예측하기

    매개변수:
        surname (str): 분류할 성씨
        classifier (SurnameClassifer): 분류기 객체
        vectorizer (SurnameVectorizer): SurnameVectorizer 객체
    반환값:
        가장 가능성이 높은 국적과 확률로 구성된 딕셔너리
    """
    vectorized_surname = vectorizer.vectorize(surname)
    vectorized_surname = torch.tensor(vectorized_surname).view(1, -1)
    result = classifier(vectorized_surname, apply_softmax=True)

    probability_values, indices = result.max(dim=1)
    index = indices.item()

    predicted_nationality = vectorizer.nationality_vocab.lookup_index(index)
    probability_value = probability_values.item()

    return {'nationality': predicted_nationality, 'probability': probability_value}
```

```python
new_surname = input("분류하려는 성씨를 입력하세요: ")
classifier = classifier.to("cpu")
prediction = predict_nationality(new_surname, classifier, vectorizer)
```

두개를 비교할때

CrossEntropyLoss사용label은 loss계

모델예측값

나라가하나라

모델예측값이 label과 비교해서

loss가 적게되도록 classifier wei

update되고 예측값이 변경되면

4-2_Classifying_Surnames_with_an_MLP.ipynb

[22] 얼마나 많은 예측을 보고 싶나요? 5

```
최상위 5개 예측:
==================
Kim -> Korean (p=0.37)
Kim -> Japanese (p=0.17)
Kim -> Arabic (p=0.11)
Kim -> German (p=0.08)
Kim -> Czech (p=0.07)
```

```python
print(len(dataset.train_df))
print(len(dataset.val_df))
print(len(dataset.test_df))
```

```
7680
1640
1660
```

개수

```python
    def __init__(self, surname_df, vectorizer):
        매개변수:
            surname_df (pandas.DataFrame): 데이터셋
            vectorizer (SurnameVectorizer):
            SurnameVectorizer 객체
        """
        self.surname_df = surname_df #내부에서만
        self._vectorizer = vectorizer

        self.train_df = self.surname_df[self.surname_df.split=='train']
        self.train_size = len(self.train_df)

        self.val_df = self.surname_df[self.surname_df.split=='val']
        self.validation_size = len(self.val_df)

        self.test_df = self.surname_df[self.surname_df.split=='test']
        self.test_size = len(self.test_df)

        self._lookup_dict = {'train': (self.train_df, self.train_size),
                             'val': (self.val_df, self.validation_size),
                             'test': (self.test_df, self.test_
```

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

4-2_Classifying_Surnames_with_an_MLP.ipynb

[16]
```
split=train:  99%|▓▓▓▓▓▓▓▓▓▓| 119/120 [03:28<00:04,  4.11s/it, acc=51.8, epoch=99, loss=
split=val:  96%|▓▓▓▓▓▓▓▓▓▓| 24/25 [03:28<00:03,  3.73s/it, acc=45.4, epoch=99, loss=
```

```
# 가장 좋은 모델을 사용해 테스트 세트의 손실과 정확도를 계산합니다
classifier.load_state_dict(torch.load(train_state['model_filename']))

classifier = classifier.to(args.device)
dataset.class_weights = dataset.class_weights.to(args.device)
loss_func = nn.CrossEntropyLoss(dataset.class_weights)

dataset.set_split('test')
batch_generator = generate_batches(dataset,
                                   batch_size=args.batch_size,
                                   device=args.device)
running_loss = 0.
running_acc = 0.
classifier.eval()

for batch_index, batch_dict in enumerate(batch_generator):
    # 출력을 계산합니다
    y_pred =  classifier(batch_dict['x_surname'])

    # 손실을 계산합니다
    loss = loss_func(y_pred, batch_dict['y_nationality'])
    loss_t = loss.item()
    running_loss += (loss_t - running_loss) / (batch_index + 1)

    # 정확도를 계산합니다
    acc_t = compute_accuracy(y_pred, batch_dict['y_nationality'])
    running_acc += (acc_t - running_acc) / (batch_index + 1)
```

train, validation끝나고 새로운 data로 하는 test

4-2_Classifying_Surnames_with_an_MLP.ipynb

추론

```
def predict_nationality(surname, classifier, vectorizer):
    """새로운 성씨로 국적 예측하기
    매개변수:
        surname (str): 분류할 성씨
        classifier (SurnameClassifer): 분류기 객체
        vectorizer (SurnameVectorizer): SurnameVectorizer 객체
    반환값:
        가장 가능성이 높은 국적과 확률로 구성된 딕셔너리
    """
    vectorized_surname = vectorizer.vectorize(surname)
    vectorized_surname = torch.tensor(vectorized_surname).view(1, -1)
    result = classifier(vectorized_surname, apply_softmax=True)

    probability_values, indices = result.max(dim=1)
    index = indices.item()

    predicted_nationality = vectorizer.nationality_vocab.lookup_index(index)
    probability_value = probability_values.item()

    return {'nationality': predicted_nationality, 'probability': probability_value}
```

하나 data입력시 , lee, kim같은것
모델이 어떻게 예측하는지 보는것

[20]
```
new_surname = input("분류하려는 성씨를 입력하세요: ")
classifier = classifier.to("cpu")
prediction = predict_nationality(new_surname, classifier, vectorizer)
print("{} -> {} (p={:0.2f})".format(new_surname,
                                    prediction['nationality'],
```

4-2_Classifying_Surnames_with_an_MLP.ipynb

[22]
```
k = int(input("얼마나 많은 예측을 보고 싶나요? "))
if k > len(vectorizer.nationality_vocab):
    print("앗! 전체 국적 개수보다 큰 값을 입력했습니다. 모든 국적에 대한 예측을 반환합니다
    k = len(vectorizer.nationality_vocab)

predictions = predict_topk_nationality(new_surname, classifier, vectorizer, k=k)

print("최상위 {}개 예측:".format(k))
print("==================")
for prediction in predictions:
    print("{} -> {} (p={:0.2f})".format(new_surname,
                                        prediction['nationality'],
                                        prediction['probability']))
```

```
분류하려는 성씨를 입력하세요: Kim
얼마나 많은 예측을 보고 싶나요? 5
최상위 5개 예측:
==================
Kim -> Korean (p=0.37)
Kim -> Japanese (p=0.17)
Kim -> Arabic (p=0.11)
Kim -> German (p=0.08)
Kim -> Czech (p=0.07)
```

[23]
```
print(len(dataset.train_df))
print(len(dataset.val_df))
print(len(dataset.test_df))
```

```
7680
1640
1660
```

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

원본 코드

https://colab.research.google.com/drive/13lckS_OU47911ewNURTMU9pTrm88MH6a#scrollTo=gGifMv5ONhSw

출처 파이토치로 배우는 자연어 처리

공감 구독하기

'practice_인공지능,머신러닝' 카테고리의 다른 글

CNN (0)	2024.10.14
감독학습 vs 비지도학습 (Supervised vs Unsupervised): 분류, 군집화 (0)	2024.10.13
데이터 전처리 (Data Pre-processing) (0)	2024.10.13
GPT3 (0)	2024.10.12
Transformer, RNN 차이 (0)	2024.10.12

관련글

관련글 더보기

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

자연어(NLP)

네이쳐2024 님의 블로그입니다.

구독하기 +

댓글 0

이름

비밀번호

내용을 입력하세요.

등록

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF