Dacon Exercise Classification

Private 2nd Code

Analysis by Yeseo

https://github.com/newave986

Review

1. 주어진 데이터의 특성을 파악하고, 그에 맞는 방법 적용해 보는 것이 중요함을 느낌. - 푸리에 변환

2. 지금까지 배운 것이 헛된 것 아니라는 것 느낌.

CNN, GAP, K-Fold 모두 배운 적 있는 것이었음.
하지만 어떻게 같이 사용해야 할지,
어떻게 실전에서 써야 할지 막연한 상태였는데,
이러한 분석을 통해서 감을 잡고
다음 번에는 직접 스스로 해 볼 수 있겠다는 자신감을 얻음.

3. 이곳저곳 레퍼런스 활용 유익함

Data Loading & Feature Engineering

```
train['acc_Energy']=(train['acc_x']**2+train['acc_y']**2+train['acc_z']**2)**(1/3)

train['gy_acc_Energy']=((train['gy_x']-train['acc_x'])**2+(train['gy_y']-train['acc_y'])**2+(train['gy_z']-train['acc_z'])**2)**(1/3)
```

4.1 SVM(Signal Vector Magnitude)

3축 가속도 센서의 출력 값에는 회전성분이 포함되므로 이를 고려하지 않고 하나의 대표값으로 처리하기 위하여 식 (2)와 같이 SVM을 적용하여 하나의 에너지 값(E)으로 변환하였다. 여기서 x, y, z 값은 3축 가속도 센서의 출력 가속도 값이다.

$$E = \sqrt{x^2 + y^2 + z^2} \qquad (2)$$

E : 에너지 값
x, y, z: 3축 가속도 센서 출력 값

(그림 2)의 (a)는 시간에 따른 입력신호의 3축 가속도 센서의 출력 값이며 (b)는 식 2를 적용하여 3축 가속도 센서의 x, y, z값을 에너지(E)값으로 변환하여 얻어진 데이터의 그래프이다.

3축 가속도 센서를 이용한 실시간 걸음 수 검출 알고리즘☆

Real-Time Step Count Detection Algorithm Using a Tri-Axial Accelerometer

도대체 무슨 에너지??

Feature Engineering

jerk_signal

https://www.kaggle.com/c/2020mltermprojecthar

Jerk: 속도 변화량
데이터에 대하여 각각 미분 수행,
각 센서들의 변화량을 피처로 추가

test_dt[]에 jerk_signal 함수로 만든 피처를 추가함

```
dt=0.02

def jerk_signal(signal):
        return np.array([(signal[i+1]-signal[i])/dt for i in
range(len(signal)-1)])
```

Feature Engineering **Fourier Transform**

```
from scipy import fftpack
from numpy.fft import * def fourier_transform_one_signal(t_signal):
        complex_f_signal= fftpack.fft(t_signal)
        amplitude_f_signal=np.abs(complex_f_signal)
        return amplitude_f_signal
```

푸리에 급수(Fourier Series)는 아무리 복잡한 신호라도,
기본적인 주기함수인 사인과 코사인 함수의 조합으로 전개하는 것.

푸리에 함수 변환 후에 나타나는 시간과 주파수의 관계는
일정한 주기로 나타낼 수 있어 분석하기 편리함.

Fourier Transforms (scipy.fftpack)

https://docs.scipy.org/doc/scipy-1.1.0/reference/tutorial/fftpack.html

Feature Engineering **Standard Scaling**

```
from sklearn.preprocessing import StandardScaler,MinMaxScaler

scaler = StandardScaler()

train_s.iloc[:,2:]= scaler.fit_transform(train_s.iloc[:,2:])
train_sc = pd.DataFrame(data = train_s,columns =col)
test_s.iloc[:,2:]= scaler.transform(test_s.iloc[:,2:])
test_sc = pd.DataFrame(data = test_s,columns =col)
```

StandardScaler
각 feature의 평균을 0, 표준 편차를 1로 변경.
모든 특성들이 같은 스케일을 가지게 됨.

평균 제거, 데이터를 단위 분산 조정
단점: 이상치가 있는 경우 균형 잡힌 척도 보장 불가
하지만 우리 데이터에서는 이상치 없음

유용한 참고: "어떤 스케일러를 쓸 것인가?"
https://mkjjo.github.io/python/2019/01/10/scaler.html

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

Modeling

CNN,

Global Average

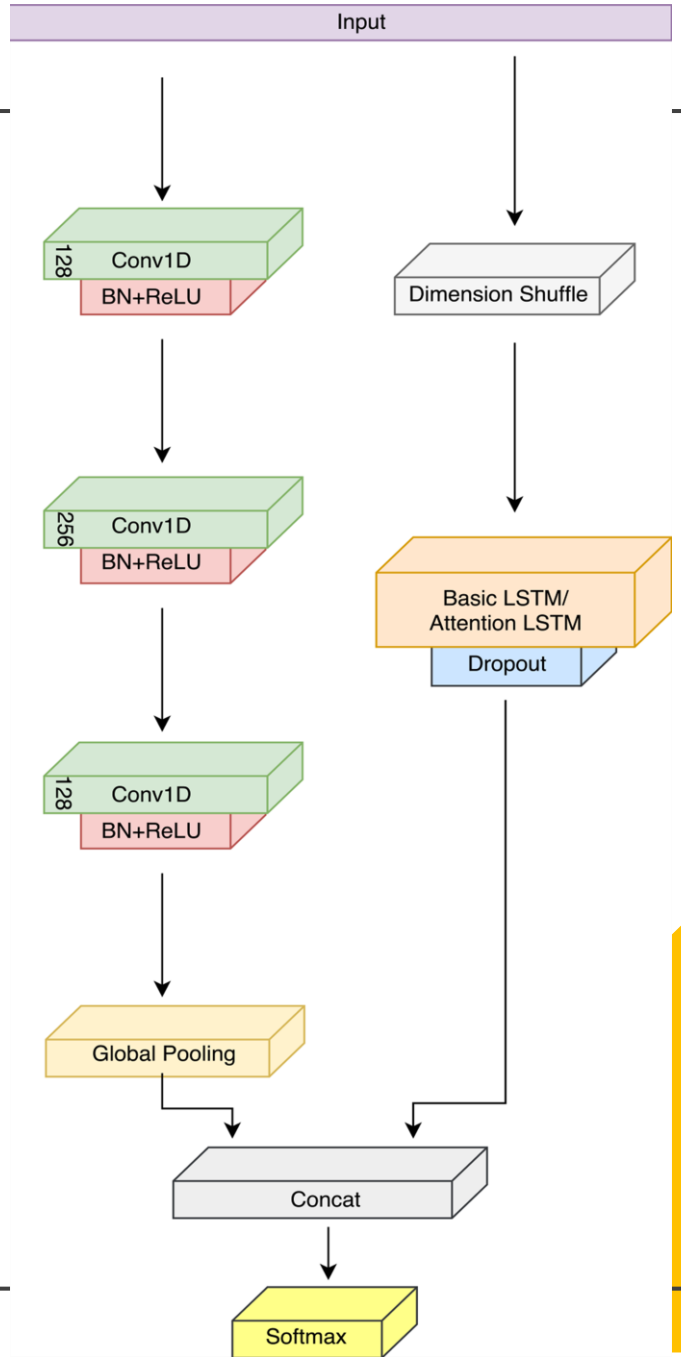
Pooling

without Flatten

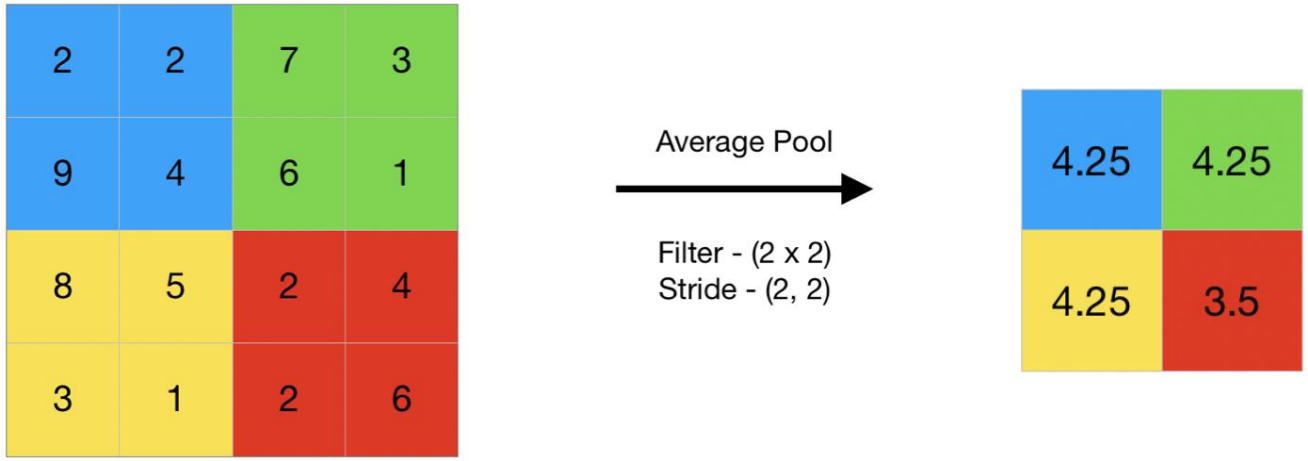
층이 3개인 CNN 모델 사용
1 dim CNN을 세 번 겹침

Conv1d + BN + relu + Dropout 세 번 반복.
Dropout의 rate만 0.3 -> 0.4 -> 0.5로 증가시킴
마지막 계층은 **softmax activation 함수** 이용

https://dzlab.github.io/timeseries/2018/11/25/LSTM-FCN-pytorch-part-1/



Modeling **CNN, Global Average Pooling** without Flatten



GAP

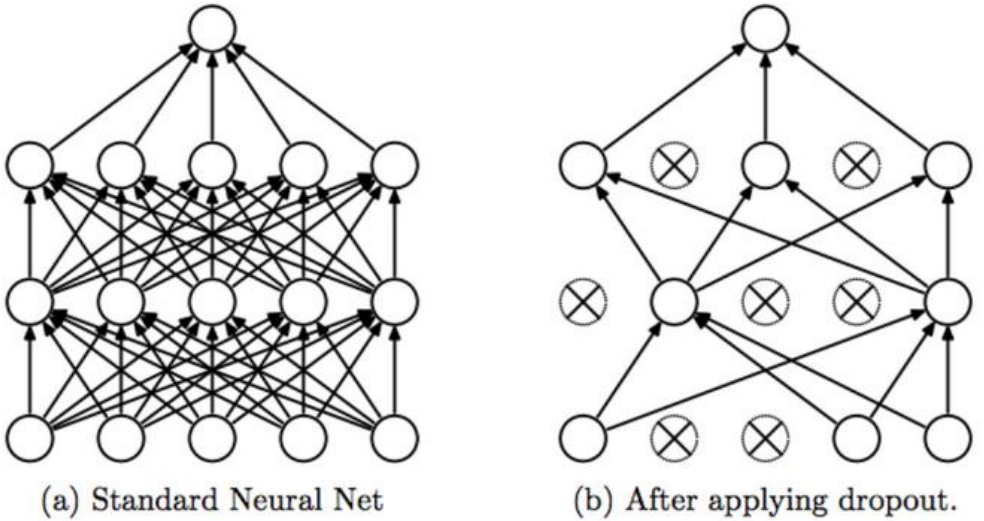
- CNN + FC에서 FC 없애기 위한 방법으로 도입
- Feature Map 모두 pooling하여
- 1x1xN 같은 형태로 mapping시킴. 하나의 neuron이 됨.
- 기존 FC의 단점 없앰:
  오래 걸림, 입력 숫자 개수 고정, feature 위치 정보 사라짐

https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

Modeling

CNN Model

Dropout



(a) Standard Neural Net (b) After applying dropout.

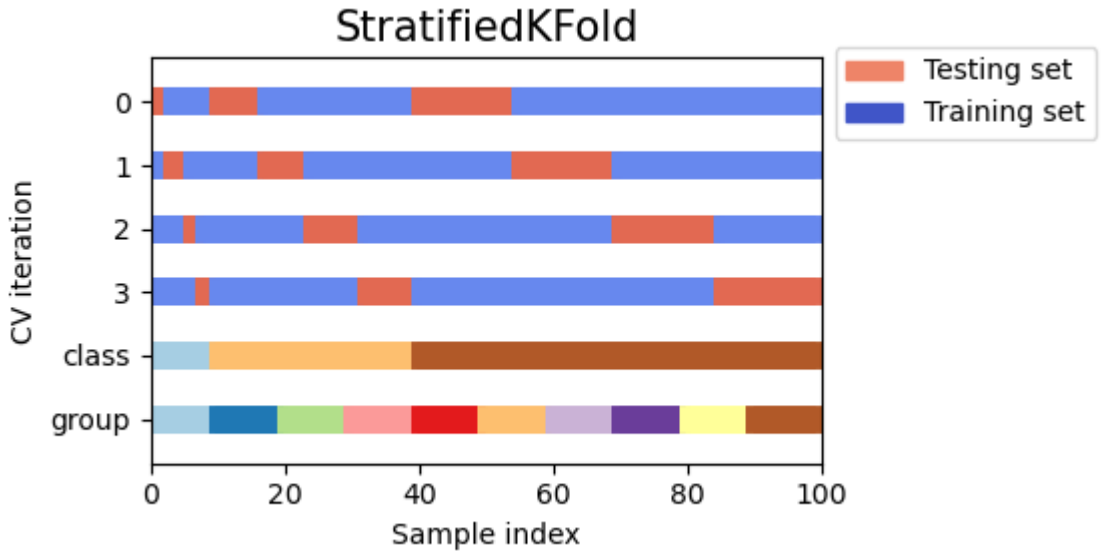
Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

Dropout
- 정규화 기법
- 가중치에 덜 민감함
- 과적합(Overfitting) 방지

Modeling

10-fold

StratifiedKFold



Stratified K Fold

- 일반적인 k-fold의 label이 고른 분포를 갖지 않는다는 단점을 해결
- 과적합(Overfitting) 방지

https://stackoverflow.com/questions/45969390/difference-between-stratifiedkfold-and-stratifiedshufflesplit-in-sklearn

Tensorflow to Pytorch

https://pytorch.org/docs/stable/index.html

	Tensorflow	Pytorch
Conv1D	keras.layers.Conv1D	torch.nn.Conv1d
Batch Normalization	keras.layers.BatchNormalization()(conv1)	torch.nn.BatchNorm1d
Global Average Pooling 1D	keras.layers.GlobalAveragePooling1D()	torch.nn.AvgPool2d
Dropout	keras.layers.Dropout(rate=0.3)(conv1)	torch.nn.Dropout
Dense Activation Softmax	keras.layers.Dense(classes, activation='softmax')(gap)	torch.nn.Softmax(dim=None)
Activation Relu	keras.layers.Activation(activation='relu')(conv1)	torch.nn.ReLU(inplace=False)