HW_4

2016707072 전자통신공학과 신민정

*고찰은 Tool로 Hyperparameter를 조정한**3.Classification**의 code에 대한 고찰으로 작성하였습니다.*

(2.Reggresion $trainY = 3 * trainX + 4 + randomnoise$ *과*

1. 자유낙하공식 : 이제 여러분은 뉴턴의 공식을 알지 못해도 위치를 추정할 수 있습니다.
$$y = 9.8x^2 + randomnoise$$
2. 삼각함수공식 : 대표적인 nonlinear 함수인 사인 함수를 추정해보세요.
$$y = sin(x)$$

2.5. 에 대한 고찰은 notebook에 텍스트로 남겼습니다.)

Keras.dataset에 있는 fashion MNIST data로 진행하였습니다.

10개의 범주(category)와 70,000개의 흑백 이미지로 구성된 패션 MNIST 데이터셋을 사용하겠습니다. 이미지는 해상도(28x28 픽셀)가 낮고 다음처럼 개별 옷 품목을 나타냅니다.

Image인 만큼 CNN이 보다 효과적일 것이라고 예상하지만 MLP로 model을 구성하여보았습니다.

-GridSearch

BaseLine은 튜토리얼에 있는 FC-3 으로 진행하였습니다.

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])
```

변경해야 할 Hyper parameter는 optimizer, epoch, learning rate,optimizer 이 외에도 batch_size, batch_norm 여부, drop out등이 있습니다.

직접 Hyper parameter를 조정하며 loss를 비교해 보는 방법도 있지만, 너무 많은 경우의 수를 고려해야하는 불편함이 있습니다.

Base line이 있고 Hyper parameter를 조정해야하는 이러한 task에서 저는 주로 GridSearch를 사용합니다.

GridSearch란 우리가 지정해준 몇 가지 잠재적 Parameter들의 후보군들의 조합 중에서 가장 Best 조합을 찾아줍니다. 우리가 하나하나 대입해 가면서 loss를 확인하는 작업을 GridSearch는 대신 해준다고 볼 수 있습니다. (sklearn 패키지에서 제공해주고 있기때문에 손쉽게 사용할 수 있습니다.)

```
def create_model(_optimizer, _lr, _batch_norm, _activation, _dropout):
    """
    기법 간 비교를 위한 모델 생성 (FC 3-Layer)
    """

    model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.BatchNormalization()  if _batch_norm else '',
    keras.layers.Dropout(_dropout),
    keras.layers.Dense(128, activation=_activation),
    keras.layers.Dense(10, activation='softmax')])

    optimizer = getattr(tf.keras.optimizers, _optimizer)(learning_rate=_lr)

    model.compile(loss='sparse_categorical_crossentropy',
              optimizer=optimizer,
              metrics=['accuracy'])

    return model

# KerasClassifier로 모델 생성기 설정
model = KerasClassifier(build_fn=create_model, verbose=3)
```

GridSearch를 진행하기 위해서 model을 캡슐화 해줍니다.

GridSearch로 비교하며 찾을 parameter list입니다.

```
param_grid
```

```
{'_activation': ['relu', 'selu', 'swish'],
 '_batch_norm': [1, 0],
 '_dropout': [0.2, 0.4],
 '_lr': [0.001, 0.002, 0.01],
 '_optimizer': ['RMSprop', 'Adam'],
 'batch_size': [512, 1024],
 'epochs': [15]}
```

캡슐화 된 model에 parameter를 바꾸어가며 학습합니다.

```
grid = GridSearchCV(estimator=model, param_grid=param_grid, cv=3)
grid_result = grid.fit(train_images, train_labels) #Grid Search 학습
grid_result
```

결과

GridSearc로 찾은 best parameter입니다.

```
9] grid_result.best_params_
```

```
{'_activation': 'swish',
 '_batch_norm': 1,
 '_dropout': 0.2,
 '_lr': 0.01,
 '_optimizer': 'Adam',
 'batch_size': 512,
 'epochs': 15}
```

Best score입니다. **0.890**

```
grid_result.best_score_
```

```
0.8903666734695435
```

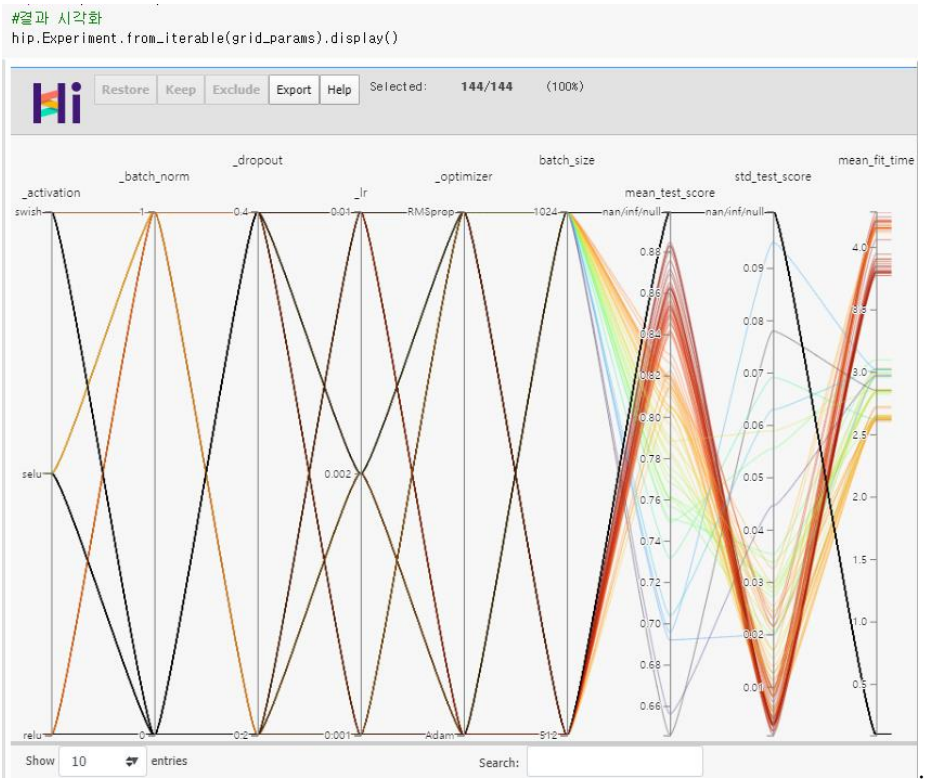
튜토리얼의 score(0.86)보다 0.03정도 향상되었지만, 기존 튜토리얼의 BaseLine을 사용하였기 때문에 큰 성과를 보지 못하였습니다.

Dense를 어느정도 깊게 쌓았다면 높은 score가 나왔을 것이라고 예상됩니다.

시각화

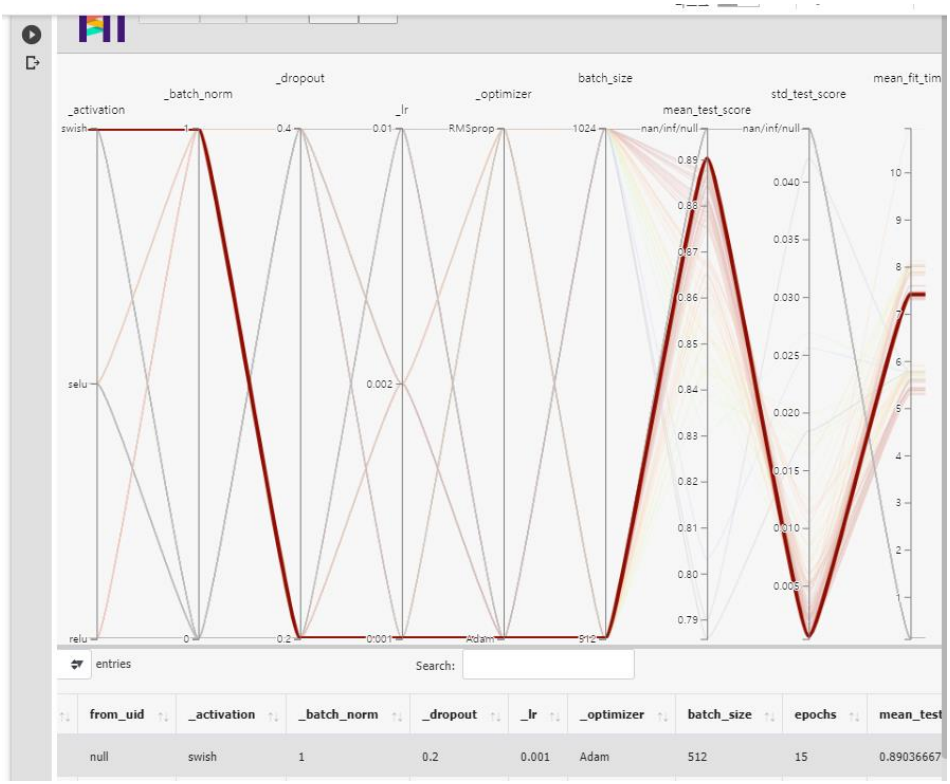
시각화로 Facebook Research의 Hiplot(https://github.com/facebookresearch/hiplot)을 사용하였습니다. 특히 고차원의 데이터 간의 패턴을 보기에 용이합니다

```
#결과 시각화
hip.Experiment.from_iterable(grid_params).display()
```



마우스 커서로 parameter별 그래프를 볼 수 있습니다.

최대점수를 낸 parameter들의 그래프



단점

하지만, 가장 큰 단점은 우리가 지정해 준 hyperparameter 후보군의 갯수만큼 비례하여 시간이 늘어나기 때문에 최적의 조합을 찾을 때까지 시간이 매우 오래 걸린다는 단점이 있습니다. 또한 MLP에서 model의 layer수나 종류는 GridSearch로 지정해 줄 수 없어 GridSearch만으로는 data에 대해 100% 최적을 찾을 수는 없습니다.

-AutoML

(Colab에서 계속 오류가 나서 jupyter notebook 환경에서 돌렸습니다.)

Base Line부터 parameter까지 알아서 짜주는 AutoML을 이용하여 classification을 진행해 보았습니다. AutoML은 data만 input으로 지정해주면 알아서 model을 짜주는 편리한 tool입니다.

단점

AutoML 시간이 오래걸린다는 단점이 있습니다. GridSearch와는 비교도 안되게 오래걸림

(제출전까지 학습이 완료되지 않아 추가제출하겠습니다)

최종 결론

Model별, layer별 튜닝해야하는 Hyper parameter의 특성을 잘 알아야 한다는 것을 알 수 있는 과제였습니다. 숫자로 입력하는 epoch, learning rate, hidden unit수들도 있고 직접 종류를 지정해주는 activation function도 있습니다. Activation function의 경우 장단점이 분명하기 때문에 data의 특성에 따라 잘 선택해 주어야 합니다. 또한 epoch, learning rate, hidden unit과 같이 숫자로 지정해주는 parameter들은 그 수가 너무 많은것도, 적은것도 좋지 않고 model에 최적화 된 parameter를 찾는 것이 중요합니다. 2.5에서 처럼, data양과 복잡도에 비해 dense와 hidden unit을 너무 많이 쌓으면 overffiting이 일어날 수 있음에 주의해야합니다.

Hyper parameter간의 상관관계를 찾는 것은 아주 어려운 일이고 data마다,model마다 성능에 영향을 주는 parameter가 다를 수 있다는 점에서 Grid Search나 Auto ML과 같이 자동화 된 Tool을 활용하여 parameter를 조정하는 것도 좋은 방법이라고 생각합니다.

(개인적으로, 시간이 많다면 AutoML로 base line을 짜고 Drop out, Batch norm등을 추가하여 실험하는 것이 가장 효율적이라고 생각합니다.)