Advanced techniques

Practice Session 6

Changho Suh

January 23, 2024

Outline

Will learn how to implement **cross validation**.

Cross validation via sklearn

```
from sklearn.model_selection import KFold

kfold = KFold(n_splits=4, shuffle=True)
```

4-fold cross validation

val	train	train	train	test
train	val	train	train	test
train	train	val	train	test
train	train	train	val	test

How to use kFold?

```
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split

X1  = np.array([10, 20, 30, 40, 50])
y1  = np.array([60, 70, 80, 90, 100])

kfold = KFold(n_splits=4,shuffle=True)

X1_,X1_test,y1_,y1_test = train_test_split(X1,y1,test_size=1/5)

print(kfold.get_n_splits())
print(X1_)
print(y1_)
```

```
4
[20 30 50 10]
[ 70  80 100  60]
```

How to split into [train,val] and [test]?

```
print(kfold.get_n_splits())
print(X1_)
print(y1_)
```

```
4
[20 30 50 10]
[ 70  80 100  60]
```

```
for train, val in kfold.split(X1_,y1_):
    print("Train indices: ",train)
    X1_train = X1_[train]
    print("Train datasets: ",X1_train)
```

```
Train indices:  [0 1 2]
Train datasets:  [20 30 50]
Train indices:  [1 2 3]
Train datasets:  [30 50 10]
Train indices:  [0 1 3]
Train datasets:  [20 30 10]
Train indices:  [0 2 3]
Train datasets:  [20 50 10]
```

Apply to adult income dataset

```
from sklearn.model_selection import KFold

kfold = KFold(n_splits=4, shuffle=True)

X_,X_test,y_,y_test = train_test_split(X,y,test_size=1/10,stratify=y)
```

Dataset construction

```
from sklearn.model_selection import KFold

kfold = KFold(n_splits=4, shuffle=True)

X_,X_test,y_,y_test = train_test_split(X,y,test_size=1/10,stratify=y)

for train, val in kfold.split(X_,y_):

    # Train and val datasets
    X_train, X_val = X[train], X[val]
    y_train, y_val = y[train], y[val]
```

Model construction

```
for train, val in kfold.split(X_,y_):
    # Train and val datasets
    X_train, X_val = X[train], X[val]
    y_train, y_val = y[train], y[val]

    # Construct a model
    init = HeNormal()
    model = Sequential()
    model.add(Dense(128,kernel_regularizer=l2(0.01),
              bias_regularizer=l2(0.01),
              kernel_initializer=init))
    model.add(BatchNormalization())
    model.add(ReLU())
    model.add(Dropout(0.5))
    model.add(Dense(1,activation='sigmoid'))
```

Compile

```
for train, val in kfold.split(X_,y_):

                    :

    # Compile
    opt = Adam(learning_rate=0.01,beta_1 = 0.9,beta_2 = 0.999)
    model.compile(optimizer=opt,
                  loss='binary_crossentropy',
                  metrics=['acc'])
```

Early stopping & learning rate decaying

```
for train, val in kfold.split(X_,y_):

                    :

    # Early stopping & learning rate decaying
    es_callback = EarlyStopping(monitor='val_acc',patience=15)
    ls_callback = LearningRateScheduler(scheduler)
```

Training & evaluation

```
for train, val in kfold.split(X_,y_):

                    :

    # Early stopping & learning rate decaying
    es_callback = EarlyStopping(monitor='val_acc',patience=15)
    ls_callback = LearningRateScheduler(scheduler)

    # Training
    hist = model.fit(X_train, y_train, epochs=100,
                     validation_data=(X_val,y_val),
                     callbacks=[es_callback,ls_callback])
    # Valiation performance
    print(model.evaluate(X_val, y_val))
```

Saving a keras model

```
model.save("2layerDNN")
```

filename

Loading a saved model

```
model.save("2layerDNN")
```

```
from tensorflow.keras.models import load_model

loaded_model = load_model('2layerDNN')

loaded_model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 128)               6528

 batch_normalization_3 (Batc  (None, 128)              512
 hNormalization)

 re_lu_3 (ReLU)              (None, 128)               0

 dropout_3 (Dropout)         (None, 128)               0

 dense_7 (Dense)             (None, 1)                 129

=================================================================
Total params: 7,169
Trainable params: 6,913
Non-trainable params: 256
_________________________________________________________________
```