19MCEC08_ANN

1 Artificial Neural Networks

[3]:

```
import numpy as np
import sklearn.datasets as datasets
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

iris=pd.read_csv('irisd.csv')
print(iris.head())
y=iris['variety']
x=iris.drop(['variety'],axis=1)

grid_values={
             'hidden_layer_sizes':[(14,14)],
             'solver':['lbfgs','sgd','adam'],
             'activation':['logistic','tanh','relu'],
             'batch_size':[5,10,20]}



x_train, x_test, y_train, y_test = train_test_split(x,y, test_size= 0.25)
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
feature_scaler = StandardScaler()
x_train = feature_scaler.fit_transform(x_train)
x_test = feature_scaler.transform(x_test)

clf=MLPClassifier(hidden_layer_sizes=(10,10),␣
 ↪max_iter=5000,solver='sgd',random_state=1)
clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
```

```
grid_clf_acc=GridSearchCV(clf,param_grid=grid_values,cv=3,scoring='accuracy')
grid_result=grid_clf_acc.fit(x_train,y_train)
print(grid_result)
print(accuracy_score(y_test,y_pred))
print(grid_result.best_params_)
```

```
   sepal.length  sepal.width  petal.length  petal.width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa

D:\Users\kshitij\Anaconda3\lib\site-
packages\sklearn\model_selection\_search.py:813: DeprecationWarning: The default
of the `iid` parameter will change from True to False in version 0.22 and will
be removed in 0.24. This will change numeric results when test-set sizes are
unequal.
  DeprecationWarning)

GridSearchCV(cv=3, error_score='raise-deprecating',
             estimator=MLPClassifier(activation='relu', alpha=0.0001,
                                     batch_size='auto', beta_1=0.9,
                                     beta_2=0.999, early_stopping=False,
                                     epsilon=1e-08, hidden_layer_sizes=(10, 10),
                                     learning_rate='constant',
                                     learning_rate_init=0.001, max_iter=5000,
                                     momentum=0.9, n_iter_no_change=10,
                                     nesterovs_momentum=True, power_t=0.5,
                                     random_state=1, shuffle=True, solver='sgd',
                                     tol=0.0001, validation_fraction=0.1,
                                     verbose=False, warm_start=False),
             iid='warn', n_jobs=None,
             param_grid={'activation': ['logistic', 'tanh', 'relu'],
                         'batch_size': [5, 10, 20],
                         'hidden_layer_sizes': [(14, 14)],
                         'solver': ['lbfgs', 'sgd', 'adam']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='accuracy', verbose=0)
0.9736842105263158
{'activation': 'relu', 'batch_size': 20, 'hidden_layer_sizes': (14, 14),
'solver': 'sgd'}
```

[5]:

```
from sklearn.metrics import␣
 ↪classification_report,confusion_matrix,accuracy_score,recall_score,precision_score,f1_score
```

```
print("\nConfusion Matrix : ")
print(confusion_matrix(y_test,y_pred))
print("\nClassification Report : ")
print(classification_report(y_test,y_pred))
print("\nAccuracy Score : {0}".format(accuracy_score(y_pred,y_test)))
print("\nRecall Score : {0}".format(recall_score(y_pred,y_test,average=None)))
print("\nPrecision Score : {0}".
 ↪format(precision_score(y_pred,y_test,average=None)))
print("\nF1 Score : {0}".format(f1_score(y_pred,y_test,average=None)))
```

```
Confusion Matrix :
[[15  0  0]
 [ 0 13  1]
 [ 0  0  9]]

Classification Report :
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        15
  Versicolor       1.00      0.93      0.96        14
   Virginica       0.90      1.00      0.95         9

    accuracy                           0.97        38
   macro avg       0.97      0.98      0.97        38
weighted avg       0.98      0.97      0.97        38


Accuracy Score : 0.9736842105263158
Recall Score : [1.  1.  0.9]
Precision Score : [1.         0.92857143 1.        ]
F1 Score : [1.         0.96296296 0.94736842]
```

[10]:

```
combinations=[]
accuracies=[]
hidden_layer_sizes = [(11,),(12,),(13,),(14,),(15,)]
activation = ['logistic','tanh','relu']
batch_size = [5,10]
for i in hidden_layer_sizes:
    for j in activation:
        for k in batch_size:
            mlp =␣
 ↪MLPClassifier(hidden_layer_sizes=i,activation=j,batch_size=k,max_iter=2000)
            mlp.fit(x_train, y_train)
            y_pred_train = mlp.predict(x_train)
            combinations.append([i,j,k,accuracy_score(y_train,y_pred_train)])
            accuracies.append(accuracy_score(y_train,y_pred_train))
```

```
print("\n\n")
print("Hidden Layer Size \t Activation Function \t Batch Size \t \t Accuracy␣
 ↪\n")
for i in combinations:
    if i[1]=='tanh' or i[1]=='relu':
        print(" {0} \t \t {1} \t \t \t {2} \t \t {3} ".
 ↪format(i[0],i[1],i[2],i[3]))
    else:
        print(" {0} \t \t {1} \t \t {2} \t \t {3} ".format(i[0],i[1],i[2],i[3]))
print("\n\n")
print("Lowest Accuracy is : {0}".format(min(accuracies)))
print("\n\n")
index=-1
for i in range(0,len(accuracies)):
    if accuracies[i]==min(accuracies):
        index=i
print("Worst Parameters : \nHidden Layer Size : {0} , Activation Function : {1}␣
 ↪, Batch Size : {2} ".
 ↪format(combinations[index][0],combinations[index][1],combinations[index][2]))
```

```
Hidden Layer Size        Activation Function     Batch Size
Accuracy

 (11,)           logistic                5               0.9821428571428571
 (11,)           logistic                10              0.9642857142857143
 (11,)           tanh                    5               0.9821428571428571
 (11,)           tanh                    10              0.9732142857142857
 (11,)           relu                    5               0.9821428571428571
 (11,)           relu                    10              0.9821428571428571
 (12,)           logistic                5               0.9821428571428571
 (12,)           logistic                10              0.9642857142857143
 (12,)           tanh                    5               0.9821428571428571
 (12,)           tanh                    10              0.9732142857142857
 (12,)           relu                    5               0.9732142857142857
 (12,)           relu                    10              0.9910714285714286
 (13,)           logistic                5               0.9732142857142857
 (13,)           logistic                10              0.9642857142857143
 (13,)           tanh                    5               0.9821428571428571
 (13,)           tanh                    10              0.9821428571428571
 (13,)           relu                    5               0.9910714285714286
 (13,)           relu                    10              0.9821428571428571
 (14,)           logistic                5               0.9642857142857143
 (14,)           logistic                10              0.9642857142857143
 (14,)           tanh                    5               0.9821428571428571
 (14,)          tanh                    10              0.9821428571428571
 (14,)          relu                    5               0.9821428571428571
 (14,)          relu                    10              0.9821428571428571
 (15,)          logistic                5               0.9821428571428571
 (15,)          logistic                10              0.9732142857142857
 (15,)          tanh                    5               0.9821428571428571
 (15,)          tanh                    10              0.9821428571428571
 (15,)          relu                    5               0.9821428571428571
 (15,)          relu                    10              0.9821428571428571
```

```
Lowest Accuracy is : 0.9642857142857143
```

```
Worst Parameters :
Hidden Layer Size : (14,) , Activation Function : logistic , Batch Size : 10
```

Analysis:-Using Grid search we obtain the best set of optimal parameters like number of hidden layers,batch size,activation function and solver.The best parameters are returned along with accuracy.Also the worst parameters are analyzed