Lab 8 - ML Programming - Task 1

January 14, 2022

1 EXERCISE 1

1.1 Optical Character Recognition via Neural Networks

[11]:
```
## Use the Sklearn library for implementing solution

import numpy as np
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import RandomizedSearchCV
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
```

[12]:
```
## Load the MNIST digits dataset via sklearn provided built-in utility␣
↪function(s)

X, y = load_digits(return_X_y=True)
```

[13]:
```
## Import the necessary classes to do k-cross fold validation
## Choose k depending upon your computational budget and task complexity but␣
↪for most purposes 'k=5' is fine
## Set aside 20% of the images for testing

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=3116)
k_fold = KFold(n_splits=5, shuffle=True, random_state=3116)
```

[14]:
```
## Define a hyperparameter grid for the 'MLPClassifier' that is the Neural␣
↪Network model implementation from Sklearn

param_grid  = {
    'hidden_layer_sizes': [(32,16,8,4,2),(8,4,2),(16,8,4),(32,)],
    'activation': ['logistic','tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
```

```
    'max_iter': [500, 750, 1000]
}

neural_net = MLPClassifier(random_state=3116)
```

[17]:
```
## Define a Random Search procedure over ranges chosen above
## Then train the model by calling the '.fit' method for the search object
## Report the best hyperparameters found

search = RandomizedSearchCV(neural_net, param_grid, cv=k_fold,␣
↪random_state=3116, n_jobs=-1).fit(X_train, y_train)
search.best_params_
```

[17]:
```
{'solver': 'adam',
 'max_iter': 500,
 'learning_rate': 'constant',
 'hidden_layer_sizes': (32,),
 'alpha': 0.05,
 'activation': 'tanh'}
```

[18]:
```
## Report one test accuracy

search.best_estimator_.fit(X_train, y_train)
y_pred = search.best_estimator_.predict(X_test)
y_true = y_test
print("Test accuracy:", np.round(accuracy_score(y_true, y_pred), 2))
```

```
Test accuracy: 0.97
```

[ ]:
```

```