FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ 2010

INTRODUCTION TO MACHINE LEARNING

HOMEWORK III

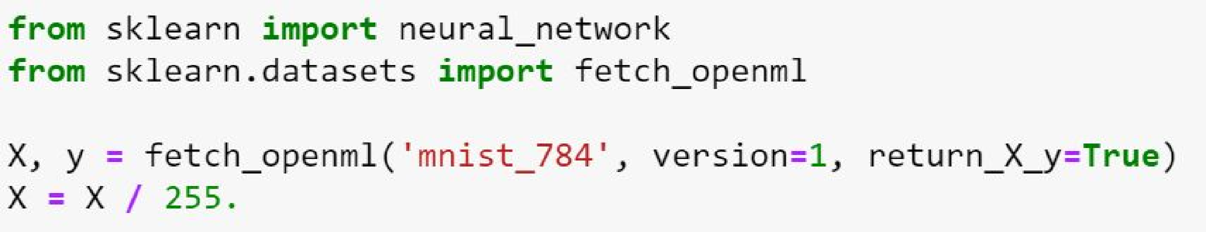
Feyza Nur SAKA

1521221051

Computer Engineering

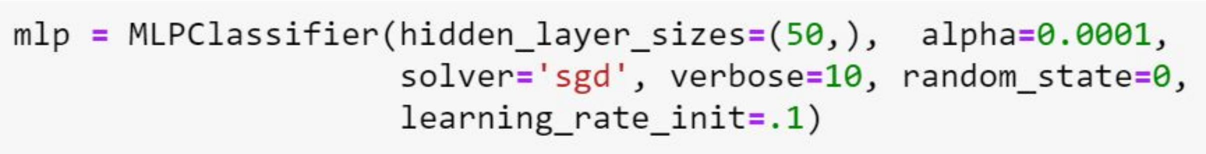
Berna KİRAZ

I used artificial neural network implementation on MNIST data using scikit-learn.



```
from sklearn import neural_network
from sklearn.datasets import fetch_openml

X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
X = X / 255.
```

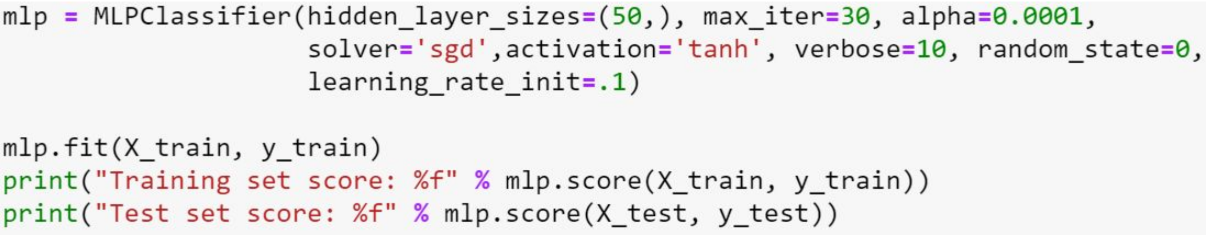
I used the MLPClassifier in the scikit-learn library to study the effect of different values.



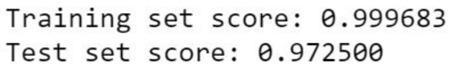
```
mlp = MLPClassifier(hidden_layer_sizes=(50,),  alpha=0.0001,
                    solver='sgd', verbose=10, random_state=0,
                    learning_rate_init=.1)
```

I compared the results using different activation functions

for tanh function

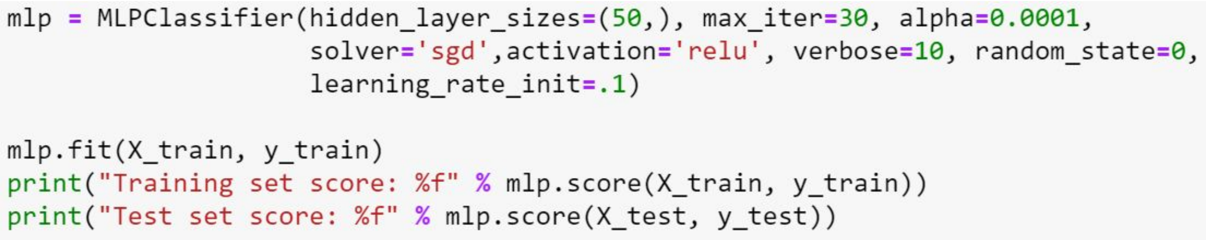


```
mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=30, alpha=0.0001,
                    solver='sgd',activation='tanh', verbose=10, random_state=0,
                    learning_rate_init=.1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```

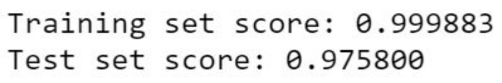


```
Training set score: 0.999683
Test set score: 0.972500
```

for relu function

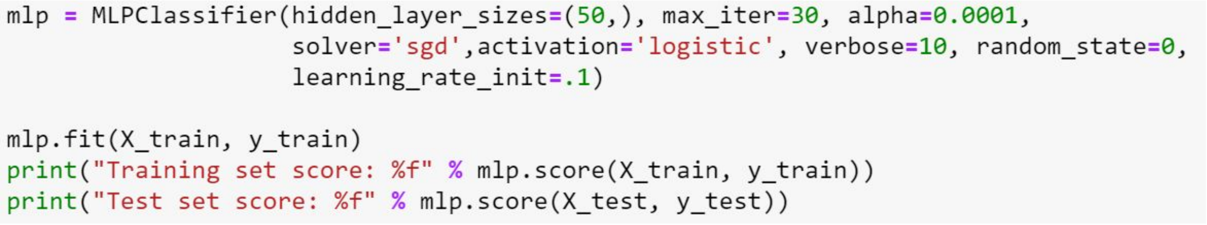


```
mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=30, alpha=0.0001,
                    solver='sgd',activation='relu', verbose=10, random_state=0,
                    learning_rate_init=.1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```

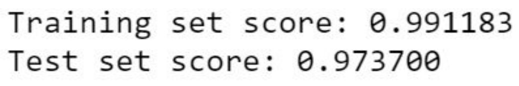


```
Training set score: 0.999883
Test set score: 0.975800
```

for logistic function



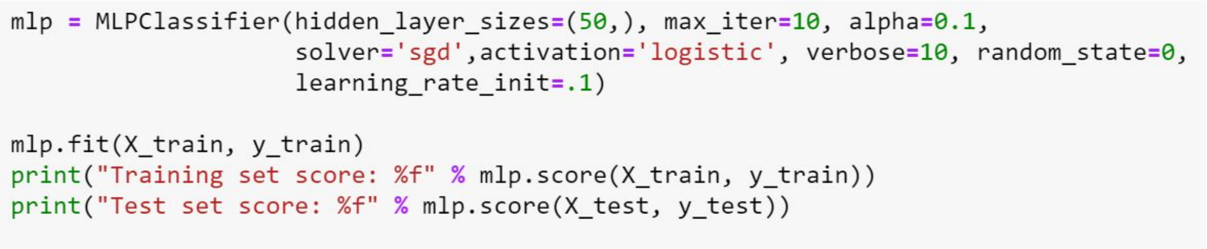
```
mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=30, alpha=0.0001,
                    solver='sgd',activation='logistic', verbose=10, random_state=0,
                    learning_rate_init=.1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```



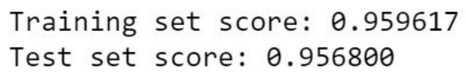
```
Training set score: 0.991183
Test set score: 0.973700
```

I compared the results using different alpha values

alpha=0.1

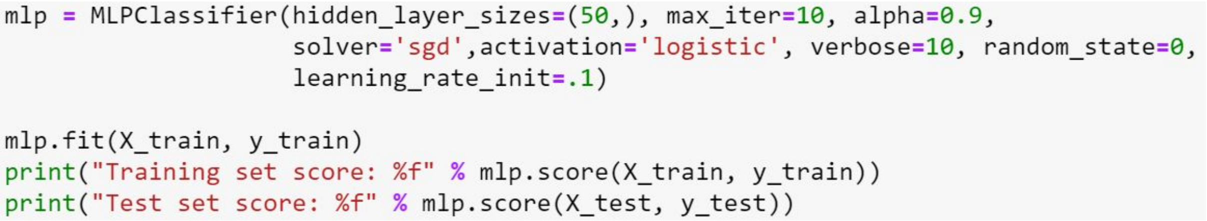


```
mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=10, alpha=0.1,
                    solver='sgd',activation='logistic', verbose=10, random_state=0,
                    learning_rate_init=.1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```

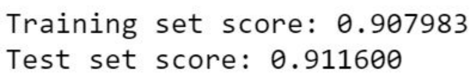


```
Training set score: 0.959617
Test set score: 0.956800
```

alpha=0.9



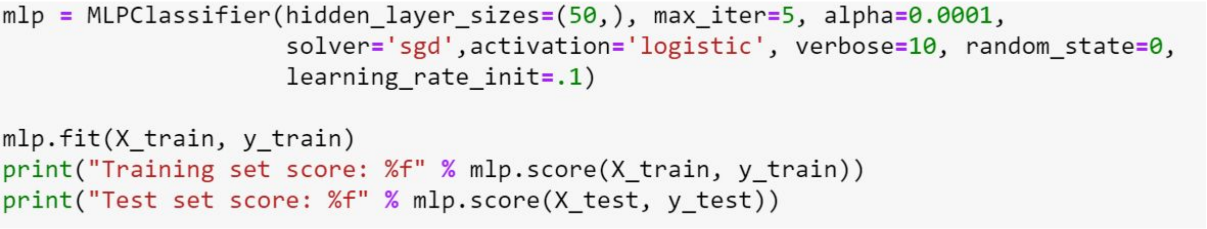
```
mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=10, alpha=0.9,
                    solver='sgd',activation='logistic', verbose=10, random_state=0,
                    learning_rate_init=.1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```



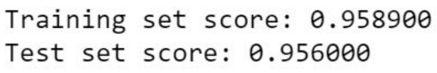
```
Training set score: 0.907983
Test set score: 0.911600
```

I compared the results using different iteration numbers

max_iter=5

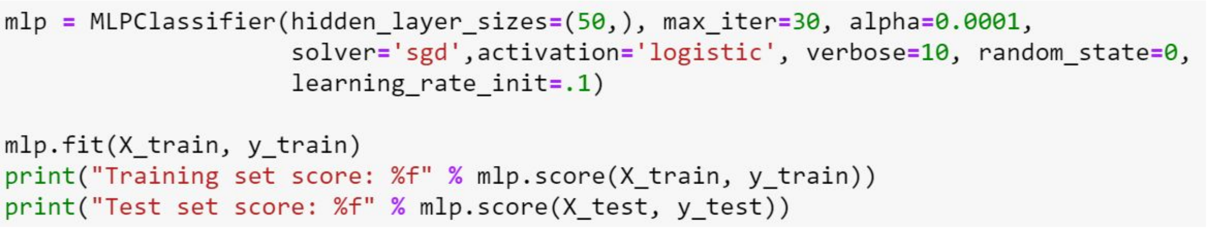


```
mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=5, alpha=0.0001,
                    solver='sgd',activation='logistic', verbose=10, random_state=0,
                    learning_rate_init=.1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```

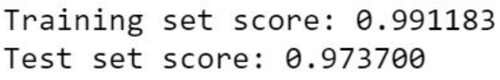


```
Training set score: 0.958900
Test set score: 0.956000
```

max_iter=30



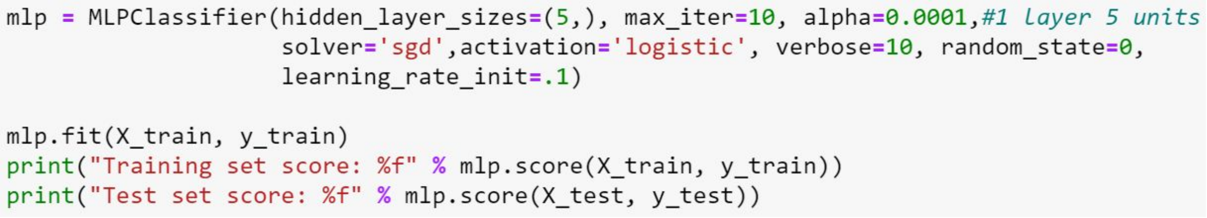
```
mlp = MLPClassifier(hidden_layer_sizes=(50,), max_iter=30, alpha=0.0001,
                    solver='sgd',activation='logistic', verbose=10, random_state=0,
                    learning_rate_init=.1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```



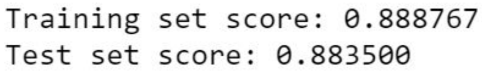
```
Training set score: 0.991183
Test set score: 0.973700
```

I compared 1 layer different layer units numbers

1 layer 5 units

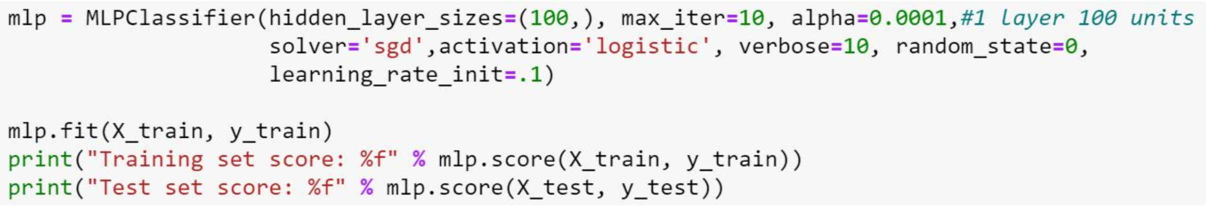


```
mlp = MLPClassifier(hidden_layer_sizes=(5,), max_iter=10, alpha=0.0001,#1 layer 5 units
                    solver='sgd',activation='logistic', verbose=10, random_state=0,
                    learning_rate_init=.1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```

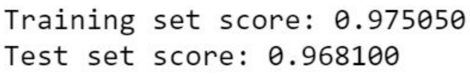


```
Training set score: 0.888767
Test set score: 0.883500
```

1 layer 100 units



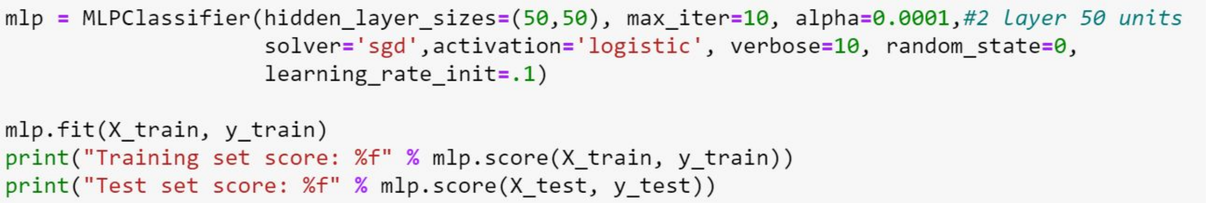
```
mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=10, alpha=0.0001,#1 layer 100 units
                    solver='sgd',activation='logistic', verbose=10, random_state=0,
                    learning_rate_init=.1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```



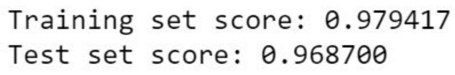
```
Training set score: 0.975050
Test set score: 0.968100
```

I compared the same layer unit numbers with different layer numbers

1 layer 50 units

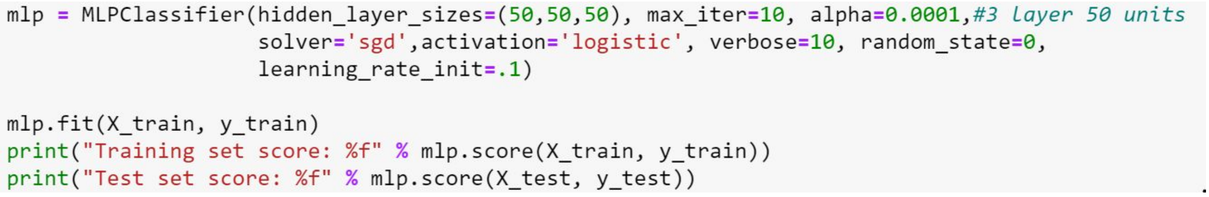


```
mlp = MLPClassifier(hidden_layer_sizes=(50,50), max_iter=10, alpha=0.0001,#2 layer 50 units
                    solver='sgd',activation='logistic', verbose=10, random_state=0,
                    learning_rate_init=.1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```

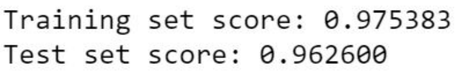


```
Training set score: 0.979417
Test set score: 0.968700
```

2 layer 50 units

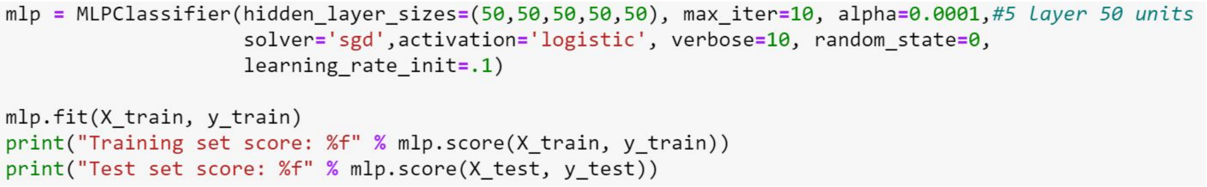


```
mlp = MLPClassifier(hidden_layer_sizes=(50,50,50), max_iter=10, alpha=0.0001,#3 layer 50 units
                    solver='sgd',activation='logistic', verbose=10, random_state=0,
                    learning_rate_init=.1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```

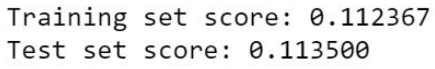


```
Training set score: 0.975383
Test set score: 0.962600
```

5 layer 50 units

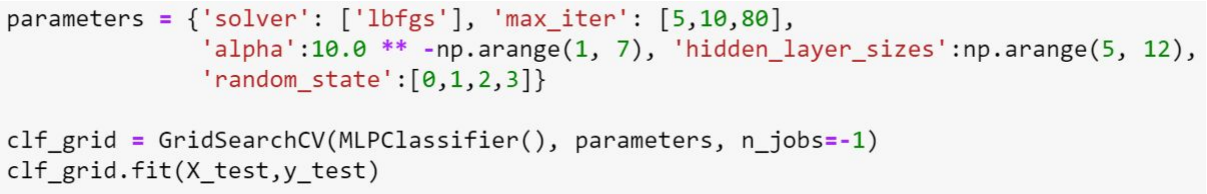


```
mlp = MLPClassifier(hidden_layer_sizes=(50,50,50,50,50), max_iter=10, alpha=0.0001,#5 layer 50 units
                    solver='sgd',activation='logistic', verbose=10, random_state=0,
                    learning_rate_init=.1)

mlp.fit(X_train, y_train)
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```

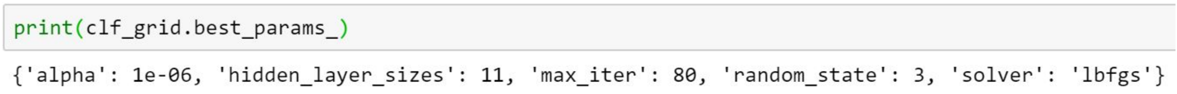


```
Training set score: 0.112367
Test set score: 0.113500
```

Finally, using GridSearchCV, I have determined the best parameter values in artificial neural networks.



```
parameters = {'solver': ['lbfgs'], 'max_iter': [5,10,80],
              'alpha':10.0 ** -np.arange(1, 7), 'hidden_layer_sizes':np.arange(5, 12),
              'random_state':[0,1,2,3]}

clf_grid = GridSearchCV(MLPClassifier(), parameters, n_jobs=-1)
clf_grid.fit(X_test,y_test)
```



```
print(clf_grid.best_params_)
```

```
{'alpha': 1e-06, 'hidden_layer_sizes': 11, 'max_iter': 80, 'random_state': 3, 'solver': 'lbfgs'}
```

Results and Comments

Relu function gave the best result and achieved less iteration.

Training set score and test set score decreased when alpha value increased.

Training set score and test set score increased when the number of iterations increased

Training set score and test set score increased in 1 layer when the number of hidden layer units increased

While the number of layers increased, the number of hidden layer units remained the same, and the training set score and test set score decreased.

KAYNAKÇA

https://github.com/krishnaik06/GRIDSearchCV/blob/master/Gridsearchcv.ipynb

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

https://www.kaggle.com/hhllcks/neural-net-with-gridsearch

https://pypi.org/project/scikit-learn/

https://scikit-learn.org/stable/install.html