In [1]:

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

import numpy as np
import matplotlib.pyplot as plt
seed = 1234
np.random.seed(seed)
```

In [2]:

```
def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict
```

Use smaller dataset to train model by neural network

We use 5000 matrixs as training data and 5000 matrixs as validating data to test our model.

In [17]:

```
# read file get X, y, testing X, testing y
original = unpickle("cifar-10-batches-py/data_batch_1")
Label_names = unpickle("cifar-10-batches-py/batches.meta")
y = original[b'labels']
X = original[b'data']
test_data = unpickle("cifar-10-batches-py/test_batch")
X_te = test_data[b'data']
y_te = test_data[b'labels']
# print(X_te)
```

In [18]:

```
# regulate X and testing X
scaler = StandardScaler()
scaler.fit(X)

X = scaler.transform(X)
X_te= scaler.transform(X_te)
```

In [19]:

```
# seperate X to training data X_tr, y_tr and validating data X_val, y_val
X_tr, y_tr = X[:5000], y[:5000]
X_val, y_val = X[5000:10000], y[5000:10000]
```

In [20]:

```
# checking accuracy score of hidden layers on mlp
hidden_sizes = [(100, 100,100,100), (100, 100, 100, 100, 100)]
tr_scores = []
val_scores = []
for i in hidden_sizes:
    mlp = MLPClassifier(hidden_layer_sizes=i, alpha=0.001, activation="relu")
    mlp.fit(X_tr, y_tr)
    tr_score = mlp.score(X_tr, y_tr)
    val_score = mlp.score(X_val, y_val)
    tr_scores.append(tr_score)
    val_scores.append(val_score)
    print("training score is", tr_score,"; validating score is", val_score,
              "hidden layer size is", i)
```

```
training score is 1.0 ; validating score is 0.3964 hidden layer size is (100, 100, 100, 100)
training score is 1.0 ; validating score is 0.4046 hidden layer size is (100, 100, 100, 100, 100)
```

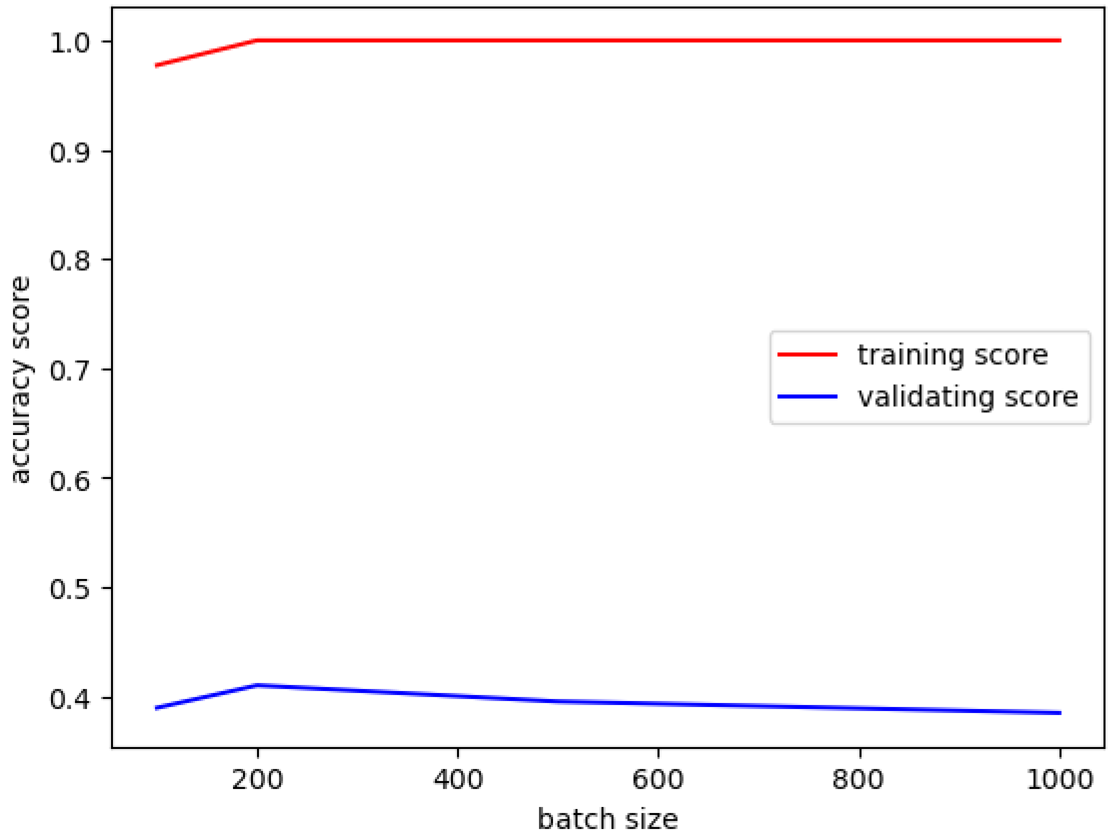
In [24]:

```
# find best accuracy score of batch size on mlp
batch_sizes = [100, 200, 500, 1000]
tr_scores = []
val_scores = []
for i in batch_sizes:
    mlp = MLPClassifier(hidden_layer_sizes=(100, 100, 100, 100, 100), alpha=0.001, batch_size = i, activation='relu')
    mlp.fit(X_tr, y_tr)
    tr_score = mlp.score(X_tr, y_tr)
    val_score = mlp.score(X_val, y_val)
    tr_scores.append(tr_score)
    val_scores.append(val_score)
    print("training score is", tr_score,"; validating score is", val_score,
              "hidden layer size is (100, 100, 100, 100, 100); alpha is 0.001; batch_size is", i)
#draw graph
fig, axes = plt.subplots()
axes.plot(batch_sizes, tr_scores, color='red', label='training score')
axes.plot(batch_sizes, val_scores, color='blue', label ='validating score')
axes.set_xlabel("batch size")
axes.set_ylabel('accuracy score')
axes.legend()
```

```
training score is 0.9772 ; validating score is 0.3892 hidden layer size is (100, 100, 100, 100, 100); alpha is 0.001;
batch_size is 100
training score is 1.0 ; validating score is 0.4098 hidden layer size is (100, 100, 100, 100, 100); alpha is 0.001; ba
tch_size is 200
training score is 1.0 ; validating score is 0.395 hidden layer size is (100, 100, 100, 100, 100); alpha is 0.001; bat
ch_size is 500
training score is 1.0 ; validating score is 0.3846 hidden layer size is (100, 100, 100, 100, 100); alpha is 0.001; ba
tch_size is 1000
```

Out[24]:

```
<matplotlib.legend.Legend at 0x7fba90091930>
```



In [25]:

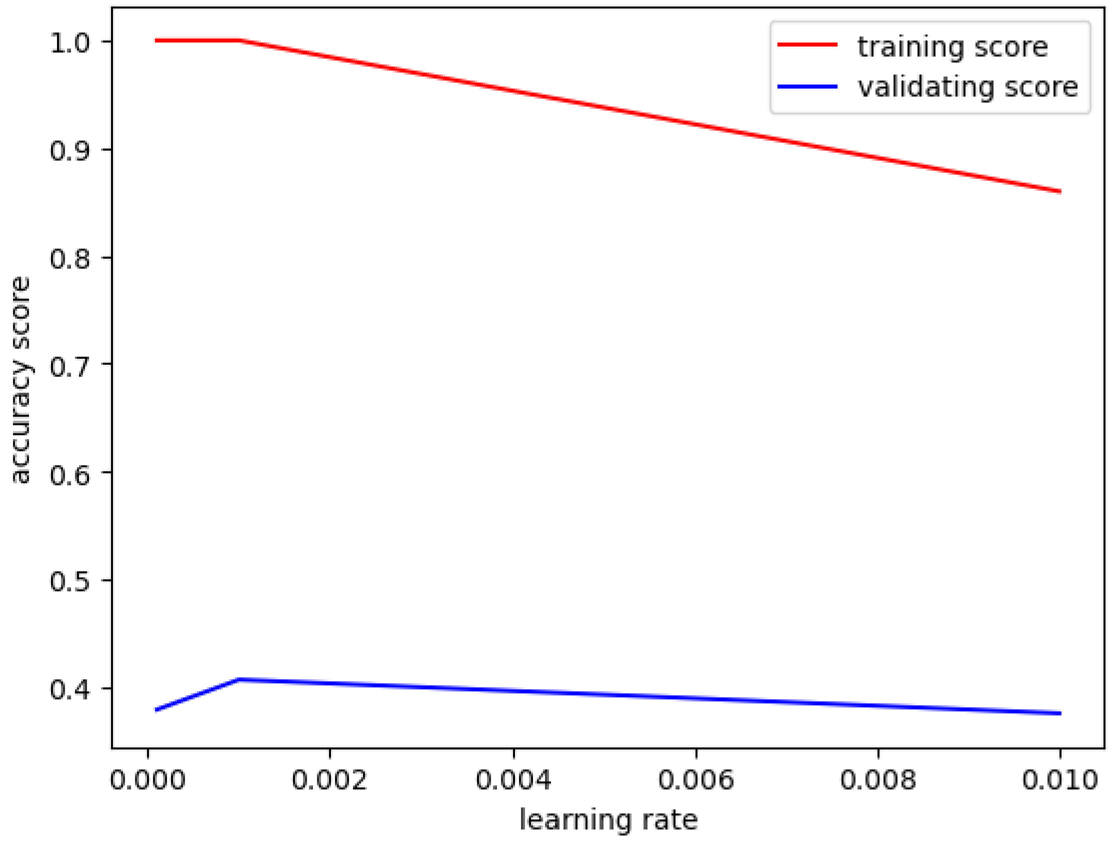
```
# find best accuracy score of learning rate on mlp
learning_rate = [0.0001, 0.001, 0.01]
tr_scores = []
val_scores = []
for i in learning_rate:
    mlp = MLPClassifier(hidden_layer_sizes=(100, 100, 100, 100, 100), alpha=0.001, learning_rate_init = i, activation=
                        max_iter = 400)
    mlp.fit(X_tr, y_tr)
    tr_score = mlp.score(X_tr, y_tr)
    val_score = mlp.score(X_val, y_val)
    tr_scores.append(tr_score)
    val_scores.append(val_score)
    print("training score is", tr_score,"; validating score is", val_score,
              "hidden layer size is (100, 100, 100, 100, 100); alpha is 0.001; learning rate is", i)

#draw graph
fig, axes = plt.subplots()
axes.plot(learning_rate, tr_scores, color='red', label='training score')
axes.plot(learning_rate, val_scores, color='blue', label ='validating score')
axes.set_xlabel("learning rate")
axes.set_ylabel('accuracy score')
axes.legend()
```

```
training score is 1.0 ; validating score is 0.379 hidden layer size is (100, 100, 100, 100, 100); alpha is 0.001; lea
rning rate is 0.0001
training score is 1.0 ; validating score is 0.4068 hidden layer size is (100, 100, 100, 100, 100); alpha is 0.001; le
arning rate is 0.001
training score is 0.8598 ; validating score is 0.3756 hidden layer size is (100, 100, 100, 100, 100); alpha is 0.001;
learning rate is 0.01
```

Out[25]:

```
<matplotlib.legend.Legend at 0x7fba900de4a0>
```



In [26]:
```
# testing data accuracy score
mlp = MLPClassifier(hidden_layer_sizes=(100, 100, 100, 100, 100), alpha=0.001, activation='relu',random_state=seed)
mlp.fit(X_tr, y_tr)
print(mlp.score(X_te, y_te))
```

```
0.3879
```

Use larger training data to check the accuracy rate increase or not

We use 10000 matrixs as training data and 10000 matrixs as validating data to train the model by neural network

In [15]:
```
# use large training data and validating data
# training data num:10000, validating data: 10000
# read file get X, y, testing X, testing y, validating X, validating y
X1 = unpickle("cifar-10-batches-py/data_batch_1")
Label_names = unpickle("cifar-10-batches-py/batches.meta")
y_tr = X1[b'labels']
X_tr = X1[b'data']

X2 = unpickle("cifar-10-batches-py/data_batch_2")
X_val = X2[b'data']
y_val = X2[b'labels']

test_data = unpickle("cifar-10-batches-py/test_batch")
X_te = test_data[b'data']
y_te = test_data[b'labels']
```

In [17]:
```
print(X_tr.shape, len(y_tr))
print(X_val.shape, len(y_val))
```

```
(10000, 3072) 10000
(10000, 3072) 10000
```

In [16]:
```
# regulate training X, validating X, and testing X
scaler = StandardScaler()
scaler.fit(X_tr)

X_tr = scaler.transform(X_tr)
X_val = scaler.transform(X_val)
X_te= scaler.transform(X_te)
```

In [18]:
```
# checking accuracy score of hidden layers on mlp
hidden_sizes = [(100, 100,100,100), (100, 100, 100)]
tr_scores = []
val_scores = []
for i in hidden_sizes:
    mlp = MLPClassifier(hidden_layer_sizes=i, alpha=0.001, activation="relu")
    mlp.fit(X_tr, y_tr)
    tr_score = mlp.score(X_tr, y_tr)
    val_score = mlp.score(X_val, y_val)
    tr_scores.append(tr_score)
    val_scores.append(val_score)
    print("training score is", tr_score,"; validating score is", val_score,
          "hidden layer size is", i)
```

```
training score is 1.0 ; validating score is 0.4234 hidden layer size is (100, 100, 100, 100)
training score is 1.0 ; validating score is 0.4198 hidden layer size is (100, 100, 100)
```

In [19]:
```
# get the testing accuracy score
mlp = MLPClassifier(hidden_layer_sizes=(100, 100, 100, 100), alpha=0.001, activation="relu")
```

```
mlp.fit(X_tr, y_tr)
print(mlp.score(X_te, y_te))
```

```
0.4308
```