Assignment2_p2

In this Assignment I use SVM and ELM algorithm to train this EEG Dataset

First I use fixed-elm algorithm to train EEG , in order to see In order to compare the influence of the number of neurons on the network, I set up a for loop of 100, 200, 500, 800, 1000 neurons, recording accuracy.

In [1]:

```python
import load_data

# return DataSet class
data = load_data.read_data_sets(one_hot=False)
# get sample number
n_samples = data.train.num_examples
# get train data and labels by batch size
train_x, train_label = data.train.next_batch(n_samples)

# get test data
test_x = data.test.data

# get test labels
test_labels = data.test.labels
```

```
/Users/wangxiang/Library/Python/3.6/lib/python/site-packages/tensorflow/p
ython/framework/dtypes.py:516: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it w
ill be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/wangxiang/Library/Python/3.6/lib/python/site-packages/tensorflow/p
ython/framework/dtypes.py:517: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it w
ill be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/wangxiang/Library/Python/3.6/lib/python/site-packages/tensorflow/p
ython/framework/dtypes.py:518: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it w
ill be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/wangxiang/Library/Python/3.6/lib/python/site-packages/tensorflow/p
ython/framework/dtypes.py:519: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it w
ill be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/wangxiang/Library/Python/3.6/lib/python/site-packages/tensorflow/p
ython/framework/dtypes.py:520: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it w
ill be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/wangxiang/Library/Python/3.6/lib/python/site-packages/tensorflow/p
ython/framework/dtypes.py:525: FutureWarning: Passing (type, 1) or '1typ
e' as a synonym of type is deprecated; in a future version of numpy, it w
ill be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])

WARNING:tensorflow:From /Users/wangxiang/Code/EEG_Practice/load_data.py:1
0: The name tf.gfile.Exists is deprecated. Please use tf.io.gfile.exists
instead.

WARNING:tensorflow:From /Users/wangxiang/Code/EEG_Practice/load_data.py:1
1: The name tf.gfile.MakeDirs is deprecated. Please use tf.io.gfile.maked
irs instead.
```

```
/Users/wangxiang/Library/Python/3.6/lib/python/site-packages/tensorboard/
compat/tensorflow_stub/dtypes.py:541: FutureWarning: Passing (type, 1) or
'1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/wangxiang/Library/Python/3.6/lib/python/site-packages/tensorboard/
compat/tensorflow_stub/dtypes.py:542: FutureWarning: Passing (type, 1) or
'1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/wangxiang/Library/Python/3.6/lib/python/site-packages/tensorboard/
compat/tensorflow_stub/dtypes.py:543: FutureWarning: Passing (type, 1) or
'1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/wangxiang/Library/Python/3.6/lib/python/site-packages/tensorboard/
compat/tensorflow_stub/dtypes.py:544: FutureWarning: Passing (type, 1) or
'1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/wangxiang/Library/Python/3.6/lib/python/site-packages/tensorboard/
compat/tensorflow_stub/dtypes.py:545: FutureWarning: Passing (type, 1) or
'1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/wangxiang/Library/Python/3.6/lib/python/site-packages/tensorboard/
compat/tensorflow_stub/dtypes.py:550: FutureWarning: Passing (type, 1) or
'1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])

Start downloading dataset...
WARNING:tensorflow:From /Users/wangxiang/Code/EEG_Practice/load_data.py:1
6: The name tf.gfile.GFile is deprecated. Please use tf.io.gfile.GFile in
stead.

Successfully downloaded train 209530726 bytes.
Start downloading dataset...
Successfully downloaded test 144273978 bytes.
```

In [9]:

```
# -*- coding:utf-8 -*-
import elm

import load_data

number_neruon= [100,200,500,800,1000]
for i in number_neruon:
    elmc = elm.ELMClassifier(n_hidden=i,activation_func='tanh')
    data = load_data.read_data_sets(one_hot=True)

#     elmc.fit(data.train.data,data.train.labels)
#     Accuracy = elmc.score(data.test.data,data.test.labels)
    elmc.fit(train_x,train_label)
    Accuracy = elmc.score(test_x,test_labels)

    print("The number of ",i ,"neurons","Testing Accuracy is ",Accuracy)
```

```
The number of  100 neurons Testing Accuracy is  0.4585225708780622
The number of  200 neurons Testing Accuracy is  0.4837255711533168
The number of  500 neurons Testing Accuracy is  0.5622247453894853
The number of  800 neurons Testing Accuracy is  0.5580443159922929
The number of  1000 neurons Testing Accuracy is  0.5932252958987063
```

Second , beacuse the performance of ELM is not good, so I use SVM Algorithm, And I use SVM with different kernel to see performance

In [6]:

```
import pandas as pd
import numpy as np
```

First I use 'linear' kernel to test this dataset

In [7]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
```

In [8]:

```
svclassifier.fit(train_x,train_label)
```

Out[8]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In [10]:

```
y_pred = svclassifier.predict(test_x)
```

In [11]:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(test_labels,y_pred))
print(classification_report(test_labels,y_pred))
```

```
[[ 9097  7792  1549]
 [ 3720 14622  1398]
 [ 2302  1133 16515]]
              precision    recall  f1-score   support

           0       0.60      0.49      0.54     18438
           1       0.62      0.74      0.68     19740
           2       0.85      0.83      0.84     19950

    accuracy                           0.69     58128
   macro avg       0.69      0.69      0.69     58128
weighted avg       0.69      0.69      0.69     58128
```

Using SVM with 'linear' kernel performace is : 69%

Second I use another kernel 'poly' to test this dataset

In [12]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='poly',degree=8)
```

In [42]:

```
# import load_data

# # return DataSet class
# data = load_data.read_data_sets(one_hot=False)

# # get train data and labels by batch size
# train_x, train_label = data.train.next_batch(n_samples)

# # get test data
# test_x = data.test.data

# # get test labels
# test_labels = data.test.labels

# # get sample number
# n_samples = data.train.num_examples
```

In [13]:

```
svclassifier.fit(train_x,train_label)
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pack
ages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma w
ill change from 'auto' to 'scale' in version 0.22 to account better for u
nscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this
warning.
  "avoid this warning.", FutureWarning)
```

Out[13]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=8, gamma='auto_deprecated',
    kernel='poly', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In [63]:

```
y_pred = svclassifier.predict(test_x)
```

In [64]:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(test_labels,y_pred))
print(classification_report(test_labels,y_pred))
```

```
[[ 8690  8269  1479]
 [ 2495 15894  1351]
 [ 2458   816 16676]]
              precision    recall  f1-score   support

           0       0.64      0.47      0.54     18438
           1       0.64      0.81      0.71     19740
           2       0.85      0.84      0.85     19950

    accuracy                           0.71     58128
   macro avg       0.71      0.70      0.70     58128
weighted avg       0.71      0.71      0.70     58128
```

Using SVM with 'poly' kernel performace is : 70%

Third i use another kernel 'rbf' to test this dataset

In [14]:

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
```

In [15]:

```
svclassifier.fit(train_x,train_label)
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pack
ages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma w
ill change from 'auto' to 'scale' in version 0.22 to account better for u
nscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this
warning.
  "avoid this warning.", FutureWarning)
```

Out[15]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In [16]:

```
y_pred = svclassifier.predict(test_x)
```

In [17]:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(test_labels,y_pred))
print(classification_report(test_labels,y_pred))
```

```
[[10925  5714  1799]
 [ 1474 17657   609]
 [  685  1060 18205]]
              precision    recall  f1-score   support

           0       0.83      0.59      0.69     18438
           1       0.72      0.89      0.80     19740
           2       0.88      0.91      0.90     19950

    accuracy                           0.80     58128
   macro avg       0.81      0.80      0.80     58128
weighted avg       0.81      0.80      0.80     58128
```

Using SVM with 'rbf' kernel performace is : 80%

Conclusion

In this assigment , I use two algorithm to test this EEG , and especially,in terms of SVM I use several kernel to compare this performace

For Fixed-ELM, the performace is just 59% for 1K neruons,so we can see that is not good and it is not my expection. Then, I use SVM ,for SVM I use several kernels : 'linear' 'poly' 'rbf'

so compare those performace,I found that Using SVM with 'rbf' kernel the performace is best,it is 80% ,which is much better than ELM and other algorithms

In [ ]:

In [ ]: