demo

January 13, 2024

```
[1]: import numpy as np
     import pandas as pd
     import pickle

     from modules.logistic_regression_cuda import LogisticRegression
     from modules.nn import NNEstimator
     from modules.svm import MultiClassSVM
     from modules.pca import PCA
     from modules.model_selection import CrossValidate, GridSearch, BootStrap
```

```
[2]: # load data
     with open('../data/train_feature.pkl', 'rb') as f:
         dataset_feature = pickle.load(f)
         dataset_feature = np.array(dataset_feature.todense()) # (11314, 10000)
     dataset_label = np.load('../data/train_labels.npy') # (11314,)

     with open('../data/test_feature.pkl', 'rb') as f:
         test_feature = pickle.load(f)
         test_feature = np.array(test_feature.todense()) # (7532, 10000)
```

0.1 SVM

```
[3]: lambda_ = 1 / 1e7
     svm = MultiClassSVM(C=1/lambda_, lr=lambda_)
     svm.fit(dataset_feature, dataset_label, n_jobs=20)
```

```
[4]: svm_pred = svm.predict(test_feature)
     pd.DataFrame({'ID': np.arange(0, len(svm_pred)), 'label': svm_pred}).
      ↪to_csv('submit_SVM.csv', index=False)
```

0.2 Logistic Regression

We use CUDA to accelerate the training process by default. If CUDA is not available, use `from

```
[5]: # if cuda is not available:
     # from modules.logistic_regression import LogisticRegression
```

```
lr = LogisticRegression(lr=100, n_iters=1000)
lr.fit(dataset_feature, dataset_label)
print("train accuracy: ", lr.score(dataset_feature, dataset_label))
```

```
train accuracy:  0.9990277532260916
```

[6]:
```
lr_pred = lr.predict(test_feature)
pd.DataFrame({'ID': np.arange(0, len(lr_pred)), 'label': lr_pred}).
 ↪to_csv('submit_LR.csv', index=False)
```

0.3 Nerual Network

[7]:
```
nn_net = NNEstimator(weight_decay=1e-7,hidden_size=1024,drop_rate=0.
 ↪9,lr=1e-3,epoch_num=20)
nn_net.fit(dataset_feature, dataset_label)
print("train accuracy: ", nn_net.score(dataset_feature, dataset_label))
```

```
train accuracy:  0.9979671203818278
```

[8]:
```
nn_pred = nn_net.predict(test_feature)
pd.DataFrame({'ID': np.arange(0, len(nn_pred)), 'label': nn_pred}).
 ↪to_csv('submit_NN.csv', index=False)
```

0.4 Model Selection

[9]:
```
## Cross Validate

cv = CrossValidate(LogisticRegression(lr=100, n_iters=1000),n_folds=5)
cv.fit(dataset_feature, dataset_label)
print(cv.get_result())
```

```
[0.8926204153778171, 0.9058771542200619, 0.9054352629253204, 0.900574458683164,
0.9022988505747126]
```

[10]:
```
## BootStrap

bs = BootStrap(LogisticRegression(lr=100, n_iters=1000),n_folds=5)
bs.fit(dataset_feature, dataset_label)
print(bs.get_result())
```

```
[0.8847590953785645, 0.8876315267024022, 0.8851113716295428, 0.8818882085719929,
0.8789958815454011]
```

0.5 Model Search

[11]:
```
param_grid = {
    'lr':[1e-4,1e-5,1e-6,1e-7],
    'C':[1e4,1e5,1e6,1e7],
}
gs_svm = GridSearch(MultiClassSVM(), param_grid, 5)
gs_svm.fit(dataset_feature, dataset_label, n_jobs=20)
gs_svm_results_df = pd.DataFrame(columns=list(param_grid.keys())+['score'])
for params, score in gs_svm.results:
    gs_svm_results_df = gs_svm_results_df.append(pd.Series({**params, 'score':␣
 ↪score}), ignore_index=True)
gs_svm_results_df
```

[11]:
```
              lr           C     score
0   1.000000e-04     10000.0  0.773285
1   1.000000e-04    100000.0  0.726169
2   1.000000e-04   1000000.0  0.585468
3   1.000000e-04  10000000.0  0.585295
4   1.000000e-05     10000.0  0.883950
5   1.000000e-05    100000.0  0.865739
6   1.000000e-05   1000000.0  0.851602
7   1.000000e-05  10000000.0  0.844354
8   1.000000e-06     10000.0  0.895173
9   1.000000e-06    100000.0  0.891109
10  1.000000e-06   1000000.0  0.888898
11  1.000000e-06  10000000.0  0.868130
12  1.000000e-07     10000.0  0.861942
13  1.000000e-07    100000.0  0.903836
14  1.000000e-07   1000000.0  0.899417
15  1.000000e-07  10000000.0  0.885539
```

[12]:
```
param_grid = {
    'lr': [1e-3],
    'drop_rate':[0.7,0.8,0.9],
    'hidden_size':[1024, 1024+512, 2048,],
    'epoch_num':[15,20,25],
    'weight_decay':[0,1e-7,1e-6,1e-5]
}
gs_nn = GridSearch(NNEstimator(), param_grid, 5)
gs_nn.fit(dataset_feature, dataset_label)
gs_nn_results_df = pd.DataFrame(columns=list(param_grid.keys())+['score'])
for params, score in gs_nn.results:
    gs_nn_results_df = gs_nn_results_df.append(pd.Series({**params, 'score':␣
 ↪score}), ignore_index=True)
gs_nn_results_df
```

[12]:
```
        lr  drop_rate  hidden_size  epoch_num  weight_decay     score
0    0.001        0.7       1024.0       15.0  0.000000e+00  0.911614
1    0.001        0.7       1024.0       15.0  1.000000e-07  0.912056
2    0.001        0.7       1024.0       15.0  1.000000e-06  0.912675
3    0.001        0.7       1024.0       15.0  1.000000e-05  0.910377
4    0.001        0.7       1024.0       20.0  0.000000e+00  0.913470
..     …          …           …          …             …       …
103  0.001        0.9       2048.0       20.0  1.000000e-05  0.911437
104  0.001        0.9       2048.0       25.0  0.000000e+00  0.911172
105  0.001        0.9       2048.0       25.0  1.000000e-07  0.915768
106  0.001        0.9       2048.0       25.0  1.000000e-06  0.913381
107  0.001        0.9       2048.0       25.0  1.000000e-05  0.908786

[108 rows x 6 columns]
```

[13]:
```python
param_grid = {
    'lr': [0.1, 1, 10, 100, 1000],
    'n_iters': [1000, 2000],
}
gs_lr = GridSearch(LogisticRegression(), param_grid, 5)
gs_lr.fit(dataset_feature, dataset_label)
gs_lr_results_df = pd.DataFrame(columns=list(param_grid.keys())+['score'])
for params, score in gs_lr.results:
    gs_lr_results_df = gs_lr_results_df.append(pd.Series({**params, 'score':␣
 ↪score}), ignore_index=True)
gs_lr_results_df
```

[13]:
```
       lr  n_iters     score
0     0.1   1000.0  0.669521
1     0.1   2000.0  0.733162
2     1.0   1000.0  0.822520
3     1.0   2000.0  0.841701
4    10.0   1000.0  0.885539
5    10.0   2000.0  0.892522
6   100.0   1000.0  0.898886
7   100.0   2000.0  0.902599
8  1000.0   1000.0  0.880414
9  1000.0   2000.0  0.884480
```

0.6 PCA

[14]:
```python
pca = PCA(n_components=5000)
pca.fit(dataset_feature)
```

[15]:
```python
dataset_feature_pca = pca.transform(dataset_feature)
test_feature_pca = pca.transform(test_feature)
```

```
[16]: lr_pca = LogisticRegression(lr=100, n_iters=1000)
      lr_pca.fit(dataset_feature_pca, dataset_label)
      print("train accuracy: ", lr_pca.score(dataset_feature_pca, dataset_label))
      lr_pred_pca = lr_pca.predict(test_feature_pca)
      pd.DataFrame({'ID': np.arange(0, len(lr_pred_pca)), 'label': lr_pred_pca}).
       ↪to_csv('submit_LR_pca.csv', index=False)
```

```
train accuracy:  0.9949619939897472
```

```
[17]: svm_pca = MultiClassSVM(C=1e5, lr=1e-7)
      svm_pca.fit(dataset_feature_pca, dataset_label, n_jobs=20)
      print("train accuracy: ", svm_pca.score(dataset_feature_pca, dataset_label))
      svm_pred_pca = svm_pca.predict(test_feature_pca)
      pd.DataFrame({'ID': np.arange(0, len(svm_pred_pca)), 'label': svm_pred_pca}).
       ↪to_csv('submit_SVM_pca.csv', index=False)
```

```
train accuracy:  0.9642920275764539
```

```
[18]: nn_pca = NNEstimator(weight_decay=1e-7,hidden_size=1024,drop_rate=0.
       ↪9,lr=1e-3,epoch_num=20)
      nn_pca.fit(dataset_feature_pca, dataset_label)
      print("train accuracy: ", nn_pca.score(dataset_feature_pca, dataset_label))
      nn_pred_pca = nn_pca.predict(test_feature_pca)
      pd.DataFrame({'ID': np.arange(0, len(nn_pred_pca)), 'label': nn_pred_pca}).
       ↪to_csv('submit_NN_pca.csv', index=False)
```

```
train accuracy:  0.9950503800601025
```