# randomforest mnist

December 18, 2021

[ ]:

```
import numpy as np
import pickle

#from train_model import train_step, test_step
from models.decisiontree import DecisionTree, RandForest
from utils.load_data import get_data
from utils.make_dict import train_bow, get_bow
```

[ ]:

```
args ={'dataset': 'MNIST',
       'dataroot': './data',
       'model': 'custom_SVM',
       'kernel': 'gaussian',
       'validation': 0.1,
       'dict_size': 100,
       'load_cluster': False
       }
```

[ ]:

```
trainX, trainy = get_data(dataset=args['dataset'], train=True,␣
 ↪dataroot=args['dataroot'])
if args['dataset'] == 'cifar10':
    trainX = trainX.reshape((-1, 32, 32, 3), order='F')

if args['load_cluster']:
    with open("./cluster.dump", "rb") as f:
        cluster = pickle.load(f)
else:
    cluster = train_bow(trainX, num_dict=args['dict_size'], num_select=10000)
    with open("./cluster.dump", "wb") as f:
        pickle.dump(cluster, f)

trainFeature = get_bow(trainX, cluster, num_dict=args['dict_size'])
```

[ ]:

```
model = RandForest(forest=5, bag_size=5000, depth=50)
#model = DecisionTree(depth=150)
```

[ ]:

```
model.fit(trainFeature, trainy)
```

```
100%|          | 5/5 [08:33<00:00, 102.72s/it]
```

```
[ ]: <models.decisiontree.RandForest at 0x1bc38054d88>
```

```
[ ]: prediction, pred = model.predict(trainFeature)
```

```
100%|      | 5/5 [00:04<00:00,  1.19it/s]
```

```
[ ]: trainFeature.shape
```

```
[ ]: (60000, 100)
```

```
[ ]: boolean = (pred == trainy)
     print(np.sum(boolean) / trainy.shape[0] * 100)
```

```
56.08333333333333
```

```
[ ]: testN = trainFeature.shape[0]
     import tqdm
     prediction = np.zeros((testN, model.forestN))

     for idx, tree in enumerate(tqdm.tqdm(model.forest)):
         prediction[:,idx] = tree.predict(trainFeature)

     print(prediction[:,1])
```

```
100%|      | 5/5 [00:04<00:00,  1.12it/s]

[3. 0. 7. … 9. 8. 8.]
```

```
[ ]: with open("./cluster.dump", "rb") as f:
         cluster = pickle.load(f)
     testX, testy = get_data(dataset=args['dataset'], train=False,␣
      ↪dataroot=args['dataroot'])
     if args['dataset'] == 'cifar10':
         testX = testX.reshape((-1, 32, 32, 3), order='F')
     testFeature = get_bow(testX, cluster, num_dict=args['dict_size'])
```

```
[ ]: pred = model.predict(testFeature)
```

```
100%|      | 5/5 [00:00<00:00,  7.14it/s]
```

```
[ ]: boolean = (pred[1] == testy)
```

```
[ ]: print(np.sum(boolean) / testy.shape[0] * 100)
```

```
49.4
```