

Project Update (Dec.1st)

What I have done since Nov.17th:

Algorithm design:

Residual vs norm

```
jupyter project Last Checkpoint: 3 minutes ago (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [30]: A = X_train
         d = y_train
         la_array = np.logspace(-6, np.log10(100),50)
         n = len(la_array)

In [31]: U,s,VT = np.linalg.svd(X_train,full_matrices=False)

In [32]: Lambda = np.logspace(-10,10,num = 50,base = 2)
         Lambda

Out[32]: array([9.76562500e-04, 1.29589385e-03, 1.71964506e-03, 2.28196091e-03,
                3.02815142e-03, 4.01834271e-03, 5.33232190e-03, 7.07596613e-03,
                9.38977382e-03, 1.24601859e-02, 1.65346080e-02, 2.19413472e-02,
                2.91168647e-02, 3.8636811e-02, 5.12799598e-02, 6.80363228e-02,
                9.02838809e-02, 1.19806286e-01, 1.58982379e-01, 2.10968870e-01,
                2.79954700e-01, 3.71498572e-01, 4.92976861e-01, 6.54177979e-01,
                8.68091107e-01, 1.15195282e+00, 1.52863599e+00, 2.02849277e+00,
                2.69180039e+00, 3.57200647e+00, 4.74803581e+00, 6.28000526e+00,
                8.34680745e+00, 1.10761743e+01, 1.46980313e+01, 1.95042185e+01,
                2.58820063e+01, 3.43453008e+01, 4.55760529e+01, 6.04792082e+01,
                8.0256253e+01, 1.06498839e+02, 1.41323458e+02, 1.87535565e+02,
                2.48958814e+02, 3.30234477e+02, 4.38219601e+02, 5.81515352e+02,
                7.71668141e+02, 1.02400000e+03])

In [33]: def ista_solve_hot(A, d, la_array):
         max_iter = 10**4
         tol = 10**(-3)
         tau = 1/np.linalg.norm(A,2)**2
         n = A.shape[1]
         w = np.zeros((n, 1))
         num_lam = len(la_array)
```

```
jupyter project Last Checkpoint: 4 minutes ago (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [34]: # At = X_train
         # bt = y_train
         # Av = X_test
         # bv = y_test
         # lam_vals = np.logspace(-10,10,num = 50,base = 2)
         # number = lam_vals.shape[0]
         # W = ista_solve_hot(At,bt,la_array);
         # err = []
         # res = []
         # norm = []
         # nonz = []
         # errv = []
         # resv = []
         # for i in range(number):
         #     err.append(np.mean(np.sign(At@w[:,i:i+1])!=bt))
         #     res.append(np.linalg.norm(At@w[:,i:i+1]-bt)**2)
         #     norm.append(np.linalg.norm(w[:,i], 1))
         #     nonz.append(np.sum(abs(w[:,i]))>1e-8)
         #     errv.append(np.mean(np.sign(Av@w[:,i:i+1])!=bv))
         #     resv.append(np.linalg.norm(Av@w[:,i:i+1]-bv)**2)

In [35]: # plt.figure()
         # plt.plot(norm, res, 'b.-')
         # plt.xlabel('$l_1$-norm');
         # plt.ylabel('Squared Error');
         # plt.title('Training Data')
         # plt.show()
         # plt.figure()
         # plt.plot(nonz, err, 'b.-')
         # plt.xlim([0,150])
         # plt.xlabel('Non Zero Elements')
         # plt.ylabel('Error Rate')
         # plt.show()
         # plt.figure()
         # plt.plot(norm, resv, 'b.-')
         # plt.xlabel('$l_1$-norm');
```

I am facing some problem in the last part of linear regression, which is a bug of array size, this bug will be fixed in the following days.

KNN

```
jupyter project Last Checkpoint a few seconds ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Checkpoint created: 23:25:03 Trusted Python 3

KNN

In [36]: y_train = y_train.astype('int')
         y_test = y_test.astype('int')

In [37]: from sklearn.neighbors import KNeighborsClassifier
         classifier = KNeighborsClassifier(n_neighbors=5)
         classifier.fit(X_train, y_train)

Out[37]: KNeighborsClassifier()

In [38]: y_pred = classifier.predict(X_test)

In [39]: from sklearn.metrics import classification_report, confusion_matrix
         print(confusion_matrix(y_test, y_pred))
         print(classification_report(y_test, y_pred))

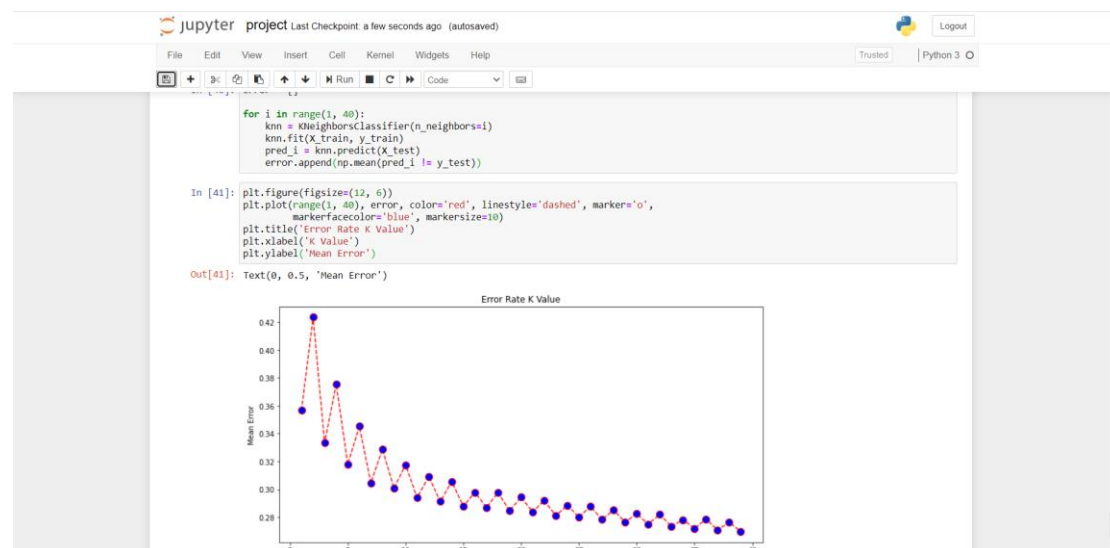
[[ 5363  4852]
 [ 4301 14265]]
      precision    recall  f1-score   support

     -1       0.55      0.53      0.54      10215
     1       0.75      0.77      0.76      18566

 accuracy          0.68      28781
 macro avg          0.65      0.65      0.65      28781
 weighted avg          0.68      0.68      0.68      28781

In [40]: error = []

         for i in range(1, 40):
             knn = KNeighborsClassifier(n_neighbors=i)
             knn.fit(X_train, y_train)
             pred_i = knn.predict(X_test)
```



First of all I pick a K value of 5 to start and get an accuracy of 68%.

Next I compare the error rate with the K value and as you can see with the increase of K the error rate is getting smaller and smaller, so the next step is to set a better K value for this algorithm.

My plan ahead

Dec.1st – Dec.6th

1. Fix bug of linear regression
2. Change the K value to get better result in KNN
3. Neural networks(still learning from activity)

Dec.7th – Dec.12th

Report