STAT 607 - Assignment 1

Name: Zhen Qin, Uniqname: qinzhen

1.1

When debugging, calculate the number of correct answers. In the k-means project, `cluster_ids` is the clustering labels, and `labels` is the provided labels.

```
adj=[v+1 for v in cluster_ids]
answ=[adj[i]-labels[i] for i in range(210)]
len([v for v in answ if v==0])
```

```
188
```

The result is 188, that is to say they disagree on 22 instances. The accuracy is $188/210 = 0.895$.

1.2

Similarly in the k-means++ project, calculate the accuracy.

```
adj=[3-v for v in cluster_ids]
answ=[adj[i]-labels[i] for i in range(210)]
len([v for v in answ if v==0])
```

```
187
```

The result is 187, that is to say they disagree on 23 instances. The accuracy is $187/210 = 0.890$.

In my project, there is no distinct differences between k-means and k-means++.

2.1

```
Gradient Descent Test...

Optimization took 0.720000 seconds.

Coordinate Descent Test...

Optimization took 2.320000 seconds.

Gradient Descent Test (logistic regression)...

Optimization took 4.728000 seconds.

Coordinate Descent Test (logistic regression)...

Optimization took 13.480000 seconds.
```

Gradient descent method took less time. The answer does not depend on the loss function type.

2.2

These optimization methods return reasonable solutions. The errors are shown below.

For gradient descent test:

```
[beta_star[i]-final_point[i] for i in range(10)]
```

> [0.080521654040286, 0.04030281147395742, -0.03473371696250571, 0.01520615631054445, 0.018958109646398436, 0.0244376703380757, 0.038891220375221136, 0.031037263128651793, -0.015646476291780204, -0.02685347415330336]

For coordinate descent test:

```
[beta_star[i]-final_point[i] for i in range(10)]
```

> [0.08052165154761515, 0.040302809936592965, -0.03473371788739965, 0.015206151885877928, 0.018958108698054588, 0.02443766542224507, 0.03889121633546244, 0.03103726671912621, -0.01564647789650689, -0.026853472076627277]

For gradient descent test(logistic regression):

```
[beta_star[i]-final_point[i] for i in range(10)]
```

> [-0.02890565921009447, 0.027113619883121554, 0.007207307476731595, 0.0430107674855823, 0.06053636734145684, 0.05818783393383847, -0.0835165478371343, 0.03607754709022509, -0.045082219428547315, 7.724501673338213e-05]

For coordinate descent test(logistic regression):

```
[beta_star[i]-final_point[i] for i in range(10)]
```

> [-0.027937933391402936, 0.024631765798382443, 0.05244422005989802, -0.0031054491557297448, 0.022133557442850504, -0.04224588876843016, 0.13362558300186056, -0.03336008512700267, -0.03695946382360449, 0.03880929878475082]

For each model, they converge to the same points. However they are not the real points. That means there are local minimum, so it will not work to run the optimization to more and more iterations.

3.1

```
min(cv_error)
```

> 0.12380952380952381

```
cv_error.index(0.12380952380952381)
```

> 24

When k was 24, the 10-fold CV error is the lowest.

Use the file "seeds dataset.txt", calculate the lowest error.

```
min(cv_error)
```

> 0.1619047619047619

Basically, the errors are higher because when calculating cross-validation errors shuffled dataset leads to similar training and test data that have lower errors, while the original dataset produces training and test data with distinct differences.