Author: Joseph D Ciarvino
Assignment: Homework 5
Class: CSE6740

1. Conceptual Questions

(a) (15 points) Consider the mutual information based feature selection. Suppose we have the following table (the entries in table indicate counts) for the spam versus and non-spam emails:

	"prize" = 1	"prize" = 0
"spam" = 1	150	10
"spam" = 0	1000	15000

	"hello" = 1	"hello" = 0
"spam" = 1	155	5
"spam" = 0	14000	2000

Given the two tables above, calculate the mutual information for the two keywords, "prize" and "hello" respectively. Which keyword is more informative for deciding whether or not the email is a spam?

See attachment(picture) below. Calculating the Mutual information, "prize" comes out significantly ahead of "hello". In fact, "hello" has a very small mutual information score which aligns with our intuition of "hello" being a generic term.

"Prize" on the other hand has a score of 0.0329 indicating it is much more useful for deciding whether an email is spam or not. It is not a perfect predictor, but per the instructions on assignment, it is better than "hello" in making a decision on spam.

```
In [9]:
import numpy as np
import pandas as pd
import math

In [10]:
# Prize/Spam MI
n11 = 150
n01 = 1000
n10 = 10
n00 = 15000
n = n11 + n01 + n10 + n00
print(n)

16160

In [11]:
def func_MI(n11,n01,n10,n00):
    n = n11 + n01 + n10 + n00
    n1_ = n11+n10
    n_1 = n11+n01
    n0_ = n01 + n00
    n_0 = n10 + n00
    t1 = (n11/n)*math.log2((n*n11)/(n1_*n_1))
    t2 = (n01/n)*math.log2((n*n01)/(n0_*n_1))
    t3 = (n10/n)*math.log2((n*n10)/(n1_*n_0))
    t4 = (n00/n)*math.log2((n*n00)/(n0_*n_0))
    return t1+t2+t3+t4

In [12]:
func_MI(n11,n01,n10,n00)

Out[12]: 0.03296011876395397

In [13]:
# Prize/Spam MI
n11 = 155
n01 = 14000
n10 = 5
n00 = 2000
n = n11 + n01 + n10 + n00
print(n)

16160

In [14]:
func_MI(n11,n01,n10,n00)

Out[14]: 0.0007839352232340266
```

1 (b)

a span.

(b) (15 points) Given two distributions, $f_0 = \mathcal{N}(0,1)$, $f_1 = \mathcal{N}(0.5,1)$ (meaning that we are interested in detecting a mean shift of minimum size 3), derive what should be the CUSUM statistic (i.e., write down the CUSM detection statistic). Plot the CUSUM statistic for a sequence of randomly generated samples, $x_1, \ldots, x_{100}$ are are i.i.d. (independent and identically distributed) according to f_0 and $x_{101}, \ldots, x_{200}$ that are i.i.d. accordign to f_1.

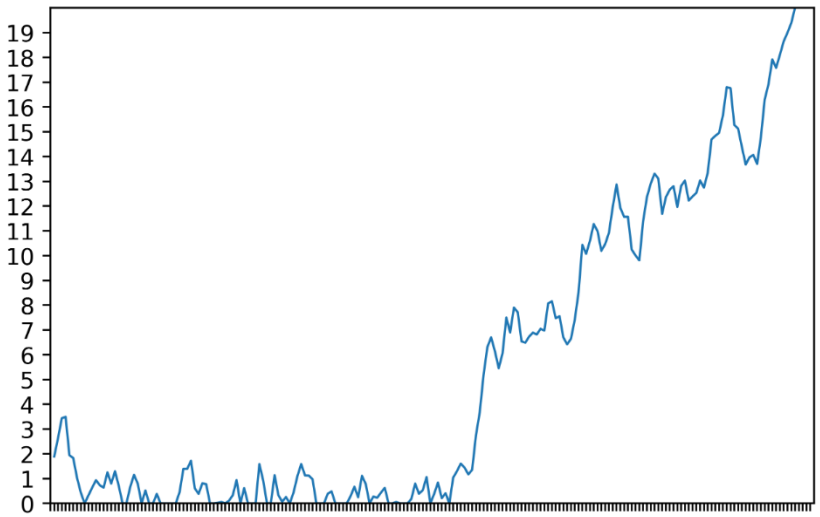
$$W_t = \max\left(W_{t-1} + \log\frac{f_1(X_t)}{f_0(X_t)}, 0\right)$$

Derive CUSUM statistic:

f1 and f0 represent the pdf of the two distributions in the problem. Each data point the probability of that data point belong to each distribution is calculated then f1/f0 and the log is taken. When the probability starts to adhere closer to N(0.5,1) then the probability increases compared to N(0,1) and the log likelihood ratio increases, pushing the CUSUM statistic higher for each sequential data point. See screenshot of code for closed form pseudo code and derivation of calculation.

```
w = 0
w_list = []
for i in full_set:
    f0 = multivariate_normal.pdf(i,mean=0,cov=1)
    f1 = multivariate_normal.pdf(i,mean=0.5,cov=1)
    log_ratio = math.log2(f1/f0)
    w = max(w+log_ratio,0)
    w_list.append(w)
    #w = max(w+math.log2())
```

CUSUM Plot



2

(a) (20 points) Now use lasso to recover the image and select λ using 10-fold cross-validation. Plot the cross-validation error curves, and show the recovered image.

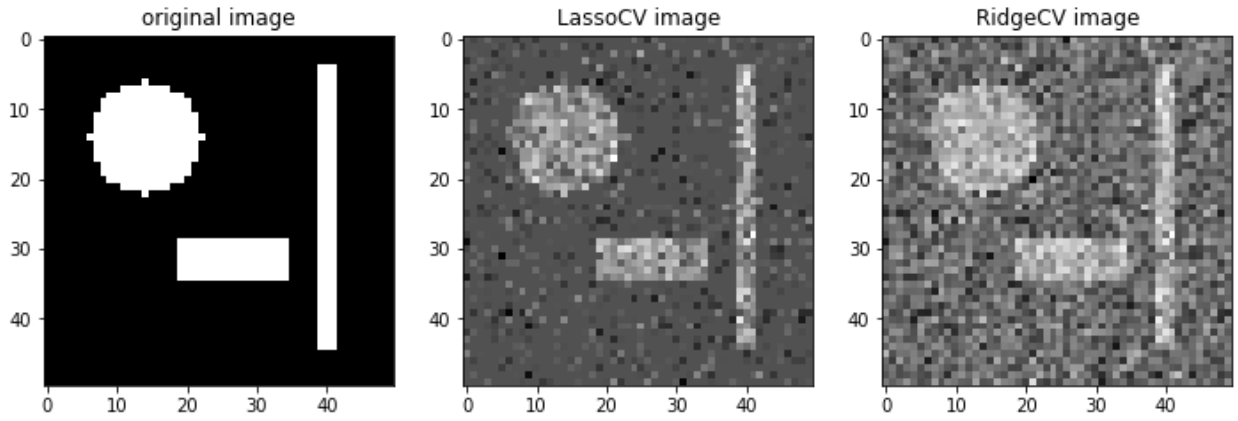
(a)

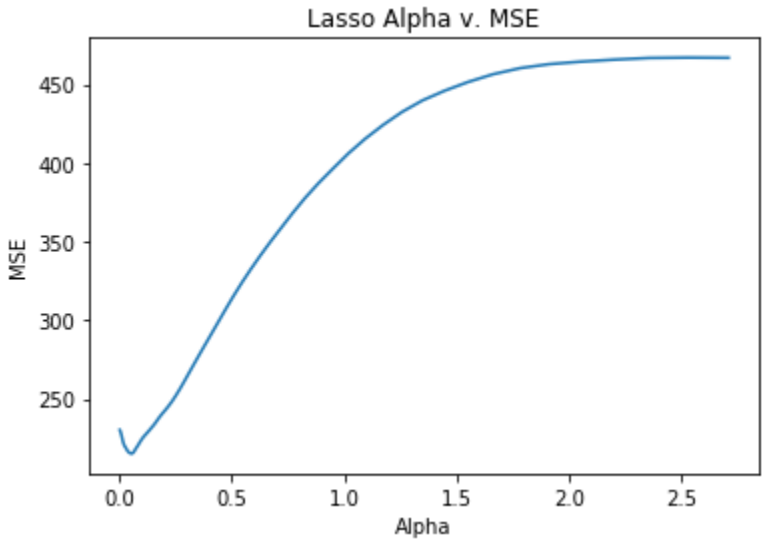
(b) (20 points) To compare, also use ridge regression to recover the image:

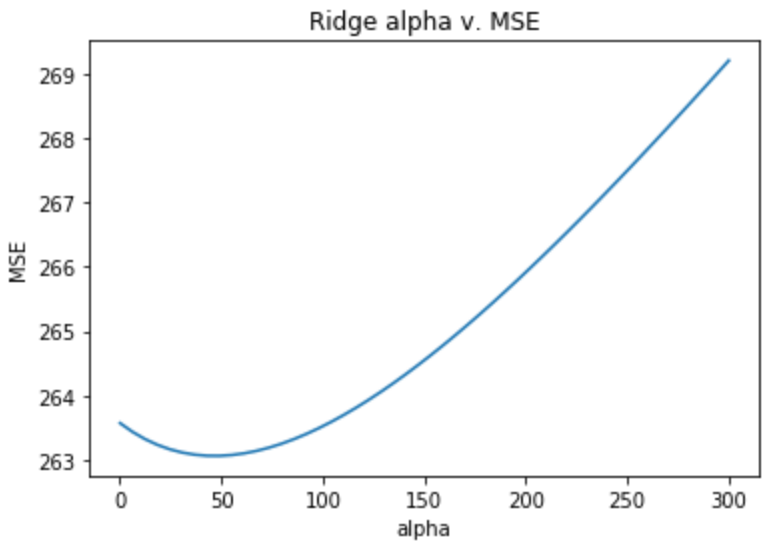
$$\min_x \|y - Ax\|_2^2 + \lambda \|x\|_2^2.$$

Select λ using 10-fold cross-validation. Plot the cross-validation error curves, and show the recovered image. Which approaches give a better recovered image?

(b)







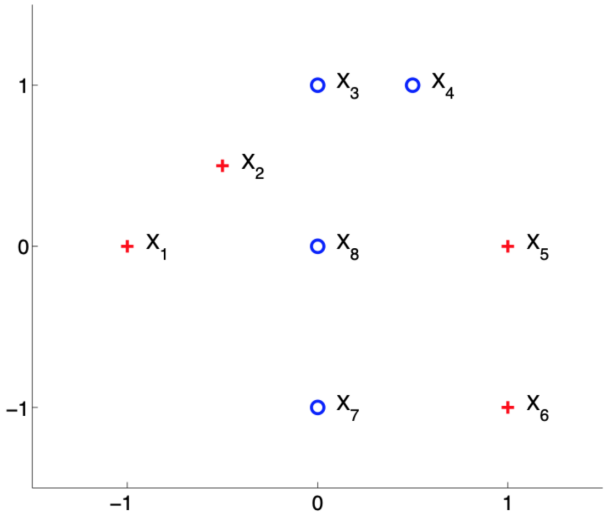
Generally, Lasso performs better in generating recovered images. The Ridge image contains alot of noise. The reason for this is that Ridge does not perform variable selection, it only shrinks coefficients. Since the original image is sparse(only a few pixels - are non-black) Ridge is not an ideal method for reconstructing the image. Lasso maintains sparsity because it performs variable selection forcing some features to zero.

3.

(a) (15 points) For each iteration $t = 1, 2, 3$, compute ϵ_t, α_t, Z_t, D_t by hand (i.e., show the calculation steps) and draw the decision stumps on the figure (you can draw this by hand).

(b) (15 points) What is the training error of this AdaBoost? Give a short explanation for why AdaBoost outperforms a single decision stump.

See attachment for calculation steps



Step 1: $D_1(i) = \frac{1}{m} = \frac{1}{8}$

① 1st Decision stump

$h_1(x) = \begin{cases} 1 & \text{if } y\text{-axis} \leq .80 \\ -1 & \end{cases}$

result: misclassify 7, 8

$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right)$

≈ 0.549306

$\epsilon_1 = \sum_{i=1}^{8} D_t(i)\,\mathbb{I}\{y \neq h_t(x^i)\} = \frac{1}{8} + \frac{1}{8} = \frac{2}{8} = \frac{1}{4}$

$Z_1 = \sum_{i}^{8} D_1(i) e^{-\alpha y_i h_1(x_i)} = \frac{6}{8}e^{-\alpha} + \frac{2}{8}e^{\alpha} \approx 0.866025$

---

② 2nd Decision stump

1, 2, 3, 4, 5, 6 = 0.083

7, 8 = 0.25

update $D_2(i)$

1	2	3	4	5	6	7	8
.083	.083	.083	.083	.083	.083	.25	.25

$h_2(x) = \begin{cases} 1 & \text{if } x \text{ is less than } 0 \\ -1 & \end{cases}$

misclassify 5, 6

$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\frac{1}{6}}{\frac{1}{6}}\right)$

≈ 0.804718

$Z_2 = \frac{10}{12}e^{-\alpha_2} + \frac{2}{12}e^{\alpha_2} = 0.745356$

$\epsilon_2 = \frac{2}{12} = \frac{1}{6}$

---

③ 3rd Decision step

update $D_3(i)$

1	2	3	4	5	6	7	8
.05	.05	.05	.05	0.25	0.25	0.15	0.15

$h_3(x) = \begin{cases} 1 & \text{if } x > .75 \\ -1 & \end{cases}$

misclassify 1, 2

$\epsilon_3 = \frac{1}{20} + \frac{1}{20} = \frac{2}{20} = \frac{1}{10}$

$\alpha_3 = \frac{1}{2}\ln\left(\frac{.9}{.1}\right) = 1.09861$

$Z_3 = \frac{9}{10}e^{-\alpha_3} + \frac{1}{10}e^{\alpha_3} = 0.6$

Final Classifier $= H(x) = \text{Sign}\left(0.5495\,\text{sign}(-x_y + 0.75) + 0.8047\,\text{sign}(-x_x + 0) + 1.098\,\text{Sign}(x_x - 0.75)\right)$

Part b - What is training error?

The training error is zero for this model (shown in above attachment). Adaboost outperforms simple single decision stumps, because the individual stumps are weak learners. They significantly underfit the data and have high bias. Aggregated however, the weightings of misclassified data points are adjusted to help the algorithm learn and calculate more accurate predictions/classifications.