Problem 1:

10

```
"Normalized" Y =
[[ 1 -2  1]
 [ 1 -5 -4]
 [ 1 -3  1]
 [ 1  0 -3]
 [ 1 -8 -1]
 [-1 -2 -5]
 [-1 -1  0]
 [-1 -5  1]
 [-1  1  3]
 [-1 -6 -1]]

(Y.transpose() * Y).inverse() =
[[ 0.10308142  0.00187679  0.00535757]
 [ 0.00187679  0.00680881 -0.00381904]
 [ 0.00535757 -0.00381904  0.01816773]]

(Y.transpose() * Y).inverse() * Y.transpose() =
[[ 0.10468542  0.0722672   0.10280863  0.08700871  0.08270956 -0.13362285
  -0.10495821 -0.10710778 -0.08513192 -0.11969971]
 [-0.01555987 -0.01689108 -0.02236868  0.01333392 -0.04877463  0.00360081
  -0.00868559 -0.03973987 -0.00652511 -0.03891059]
 [ 0.03116339 -0.04821814  0.03498243 -0.04914563  0.01774218 -0.08855815
  -0.00153853  0.03190538  0.04532658 -0.00061105]]

a =
[[-0.10104096]
 [-0.1805207 ]
 [-0.02695153]]

Y*a
[[ 0.23304891]
 [ 0.90936866]
 [ 0.4135696 ]
 [-0.02018637]
 [ 1.37007616]
 [ 0.59684001]
 [ 0.28156166]
 [ 0.97669293]
 [-0.16033433]
 [ 1.21111669]]

From Y*a we see that [0,-3] and [-1,-3] are misclassified.
```

Problem 2:

By minimizing the norm of the weight vector w, we're maximizing the margins. This will maximize the distance to the closest samples. This is a good objective function b/c you're inspecting the samples where the classes are close together and maximizing the separation.

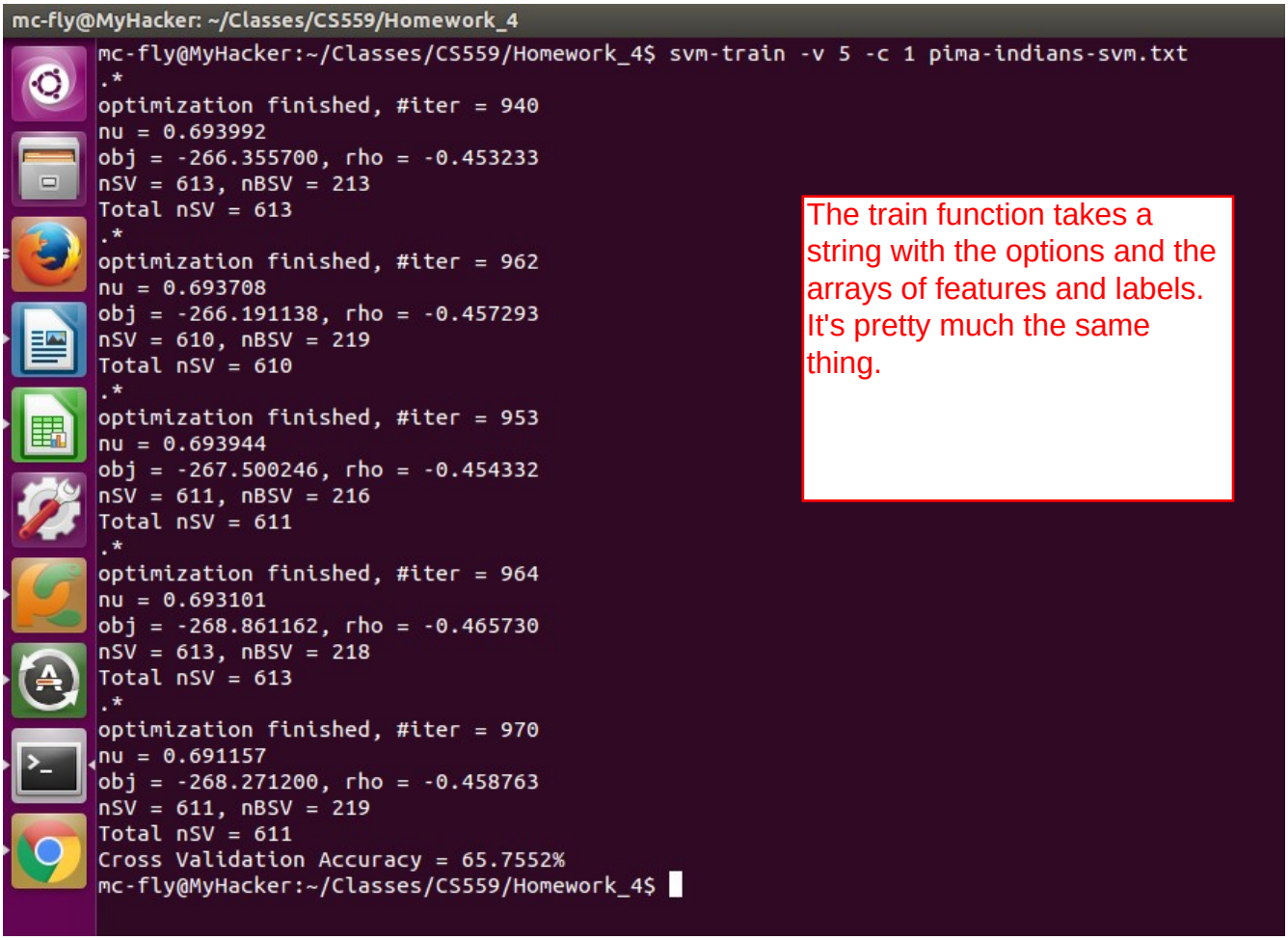
5 talk about generalization, convexity of objective function

Problem 3:

30

I was not able to figure out how to actually code up this problem using liblinear. I was only able to minipulate the original pime-indians csv file into for the format needed, then ran liblinear via the command line.

The accruacy reported is 65.75%



The train function takes a string with the options and the arrays of features and labels. It's pretty much the same thing.

Problem 4:

10

The weak learners test if column 2 > 144, next is column 3 > 107, next is column 4 > 61.
The strong classifier, which is obviously wrong, ends up [0.55, 0.55, 0]
The accuracy after each round of boosting, again obviously wrong is 75% for each round. (code is attached)

Before looking at the code: The first threshold is correct (should have been 143). The others are not. 75% is correct. The weights for the strong classifier are wrong.

==========

How do you get theta[0]= 144? Showing this is a big part of this problem. (-20)

How do you get this range `for i in range(0, 123):`?

I don't get why you check for equality with max_weight (line 52). Accuracy/error after round 1 should take the weights into account. (-5)

Predicted labels, however, should be +/-1. Don't combine with alpha to save a line of code.

==========

Weight updates are correct.

==========

No strong classifier or proper classification (-15)