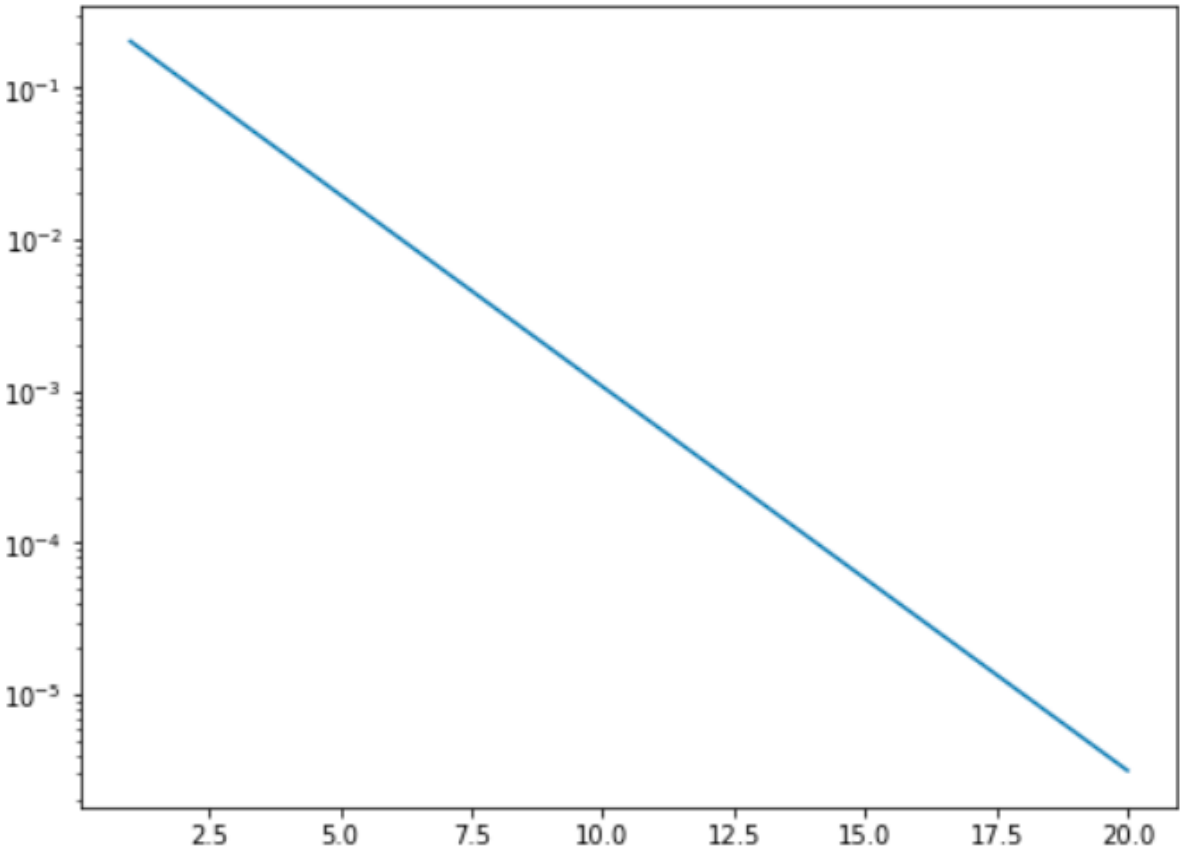
ECE586 MP3: Alternating Projection

Libo Zhang (NetID: lz200)

Exercise 1. (10 pts program + 5 pts solution)

```
(err[-1]) is shown below
3.150207425473006e-06
(v) is shown below
[0.52442269 1.26461352 2.55925049 3.85388314 5.52001383]

[<matplotlib.lines.Line2D at 0x1fef7f66340>]
```

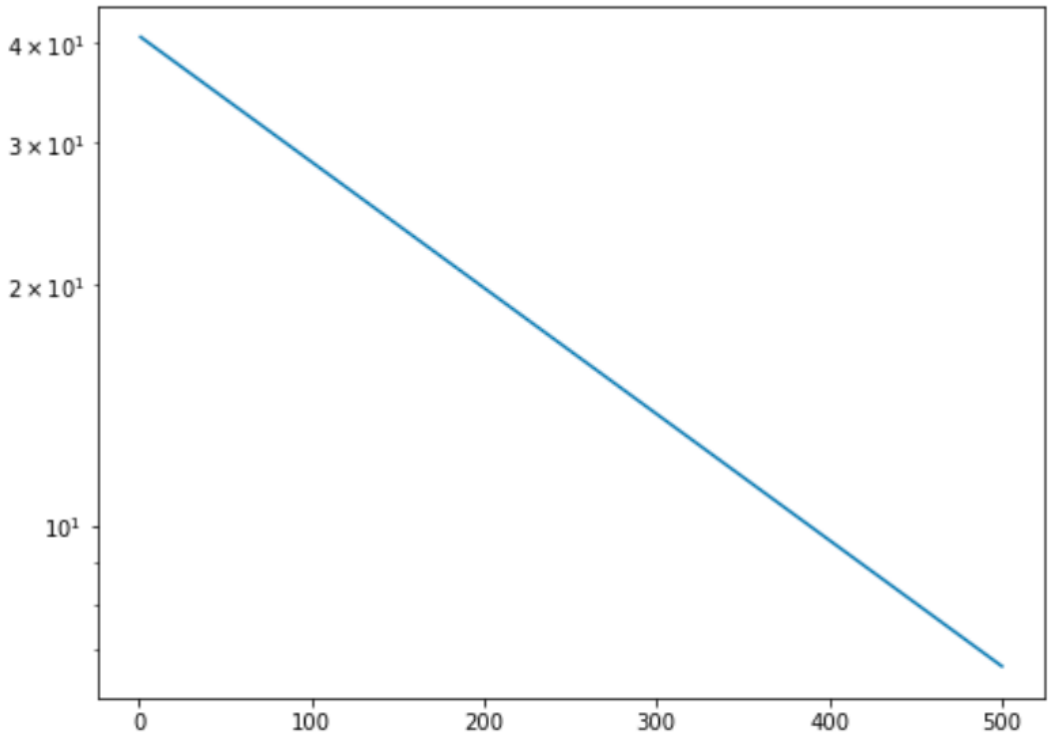


How large should n be chosen so that the projection is correct to 4 decimal places (i.e., $g_{2n} \leq 0.0001$)?

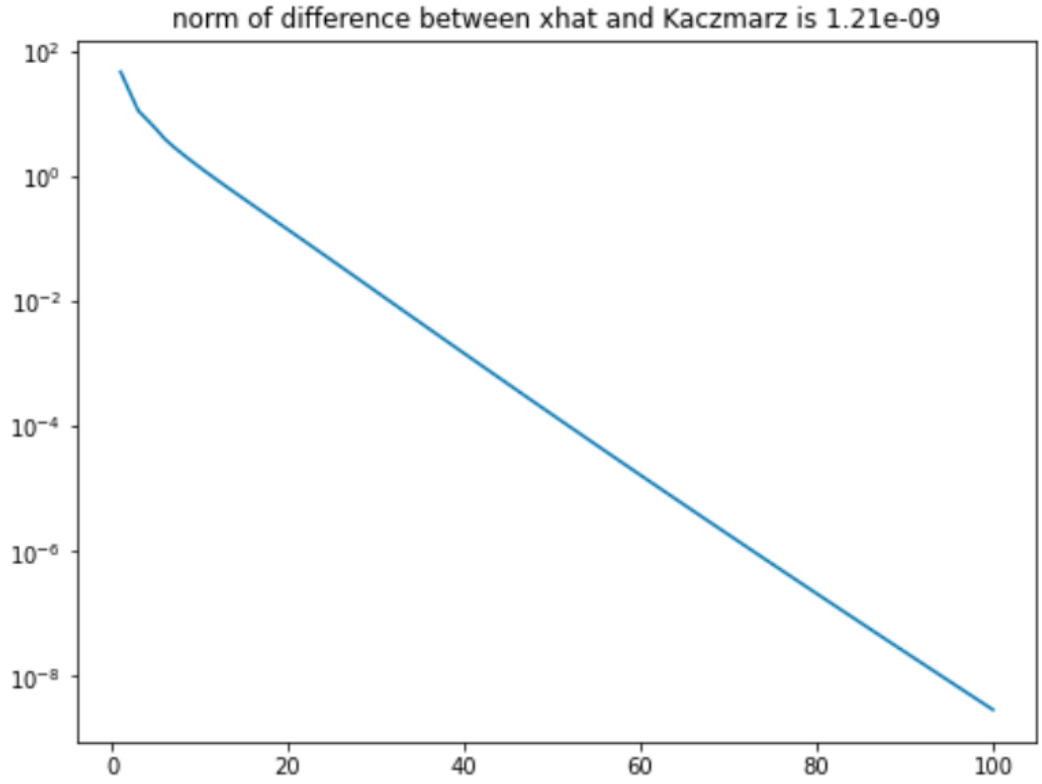
Answer: According to the above figure, we can find that when n is greater than or equal to 14, we can have that the projection is correct to 4 decimal places (10^{-4}).

Please continue to the next page for Exercise 2.

Exercise 2. (10 pts program + 5 pts solution)



Exercise 3. (10 pts)



Compare the iterative solution with the true minimum-norm solution $\hat{x} = A^H(AA^H)^{-1}\underline{b}$.

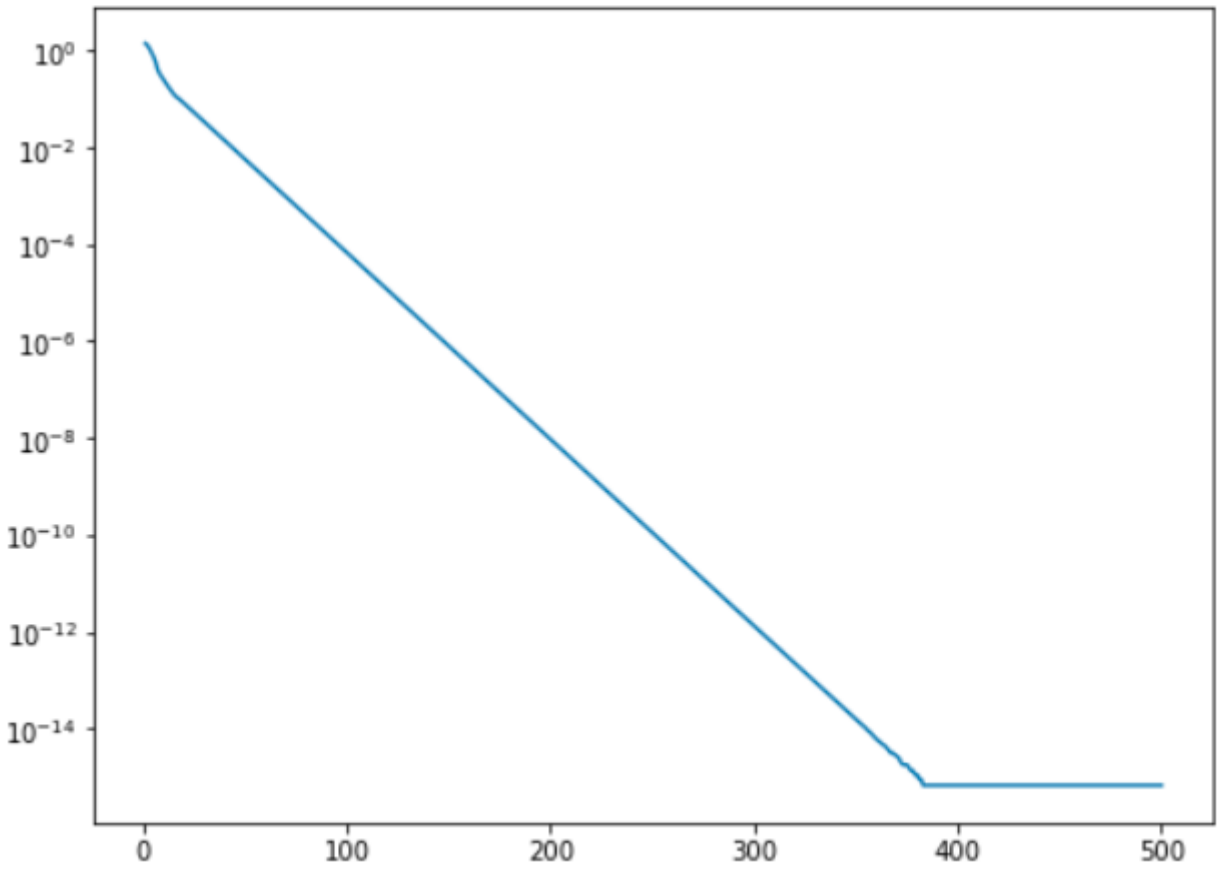
Answer: Given the above two figures from Exercise 2 and Exercise 3, we can find that the sequence generated by the iterative solution converges to the true minimum-norm solution $\hat{x} = A^H(AA^H)^{-1}\underline{b}$.

Exercise 4. (10 pts program + 5 pts solution)

```
     con: array([], dtype=float64)
     fun: 1.8272050539280826e-12
 message: 'Optimization terminated successfully.'
     nit: 5
   slack: array([8.82960371e-13, 3.84137167e-13, 1.00000000e+00])
  status: 0
 success: True
       x: array([1.50428214e-12, 2.00000000e+00, 1.00000000e+00])

True True

: array([0., 2., 1.])
```



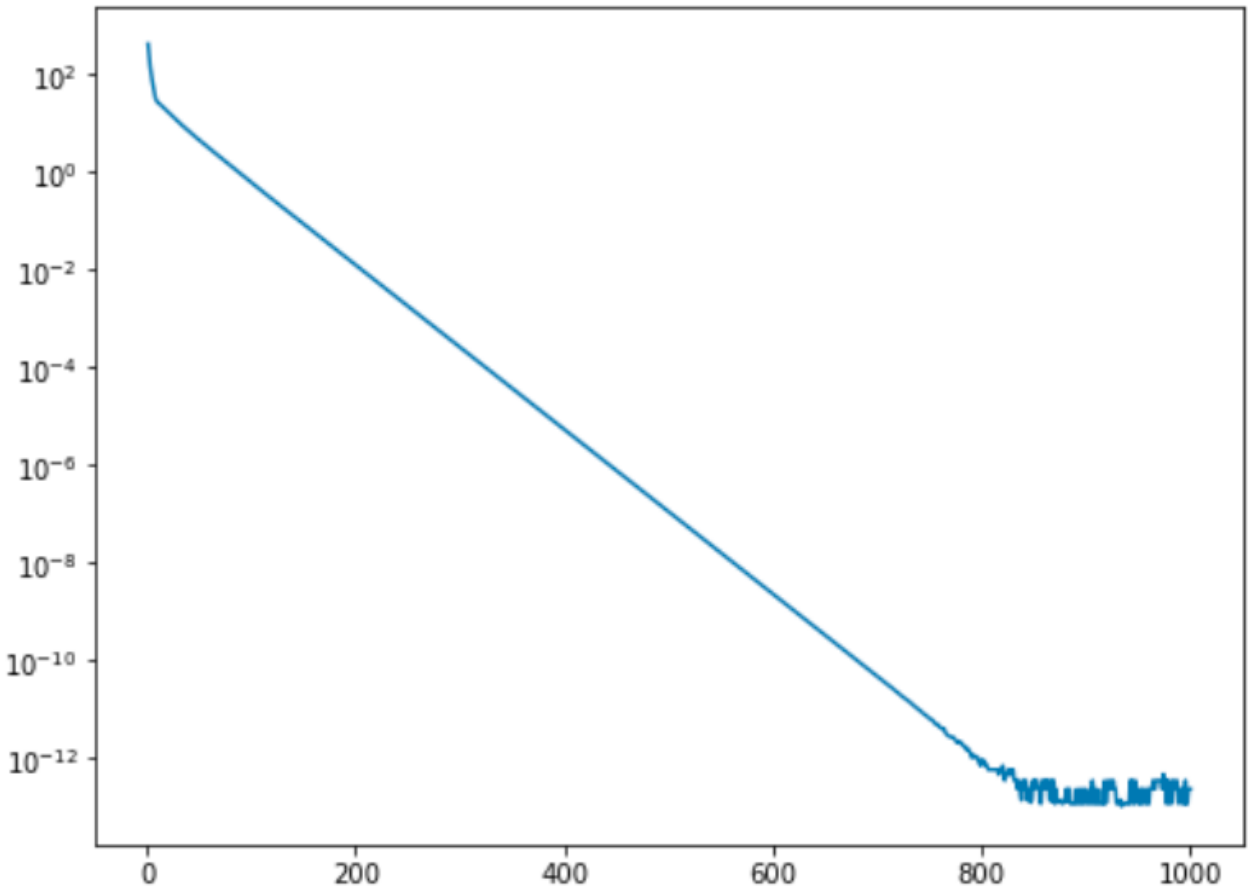
How many full passes are required so that g_k is at most 0.0001?

Answer: According to the above figure, we can see that at least 100 full passes are required so that g_k is at most 0.0001 (10^{-4}). Therefore, the number of full passes should be greater than or equal to 100.

Please continue to the next page for Exercise 5.

Exercise 5. (10 pts + 5 pts for value and strict feasibility)

```
True True [-1000.000001]
-1198.5327823524615
```




```
In [12]: np.all(x>0)
Out[12]: True

In [13]: np.all((A@x-b)>0)
Out[13]: True
```

According to the above two figures, we can conclude that firstly, for some small $\epsilon = 10^{-6} > 0$, we successfully find the strictly feasible point. Secondly, with the help of NumPy "all" method, we can confirm that the resulting $\underline{x}$ will satisfy all constraints. All constraints are satisfied.

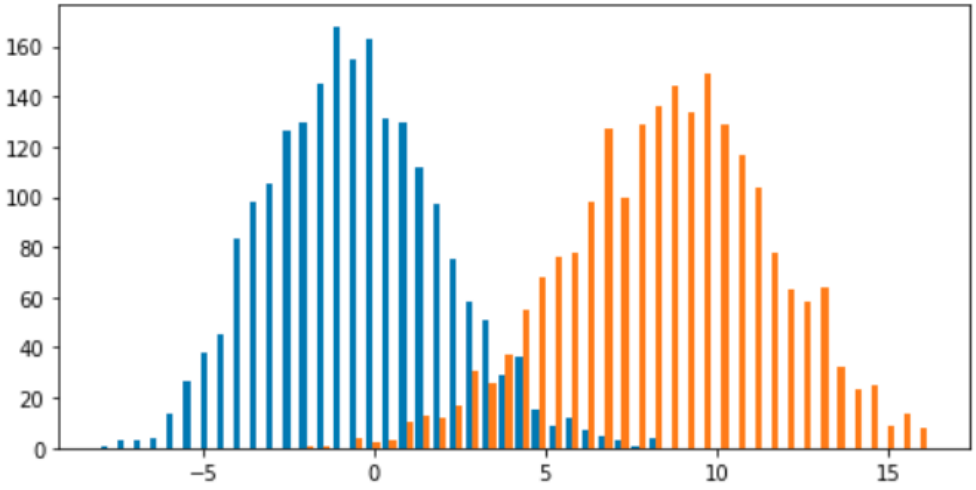
Please continue to the next page for Exercise 6.

Exercise 6. (10 pts)

Note: For this exercise, I implemented multiple experiments with different I values (the number of full passes through the alternating projection) and s values (the step size) to design a linear classifier to separate 2's and 3's. In addition, I also tried to design linear classifiers to separate other pairs of integers.
I record my experimental results below. At the end of Exercise 6, I will try to analyze and answer all Exercise 6's questions according to my recorded results.

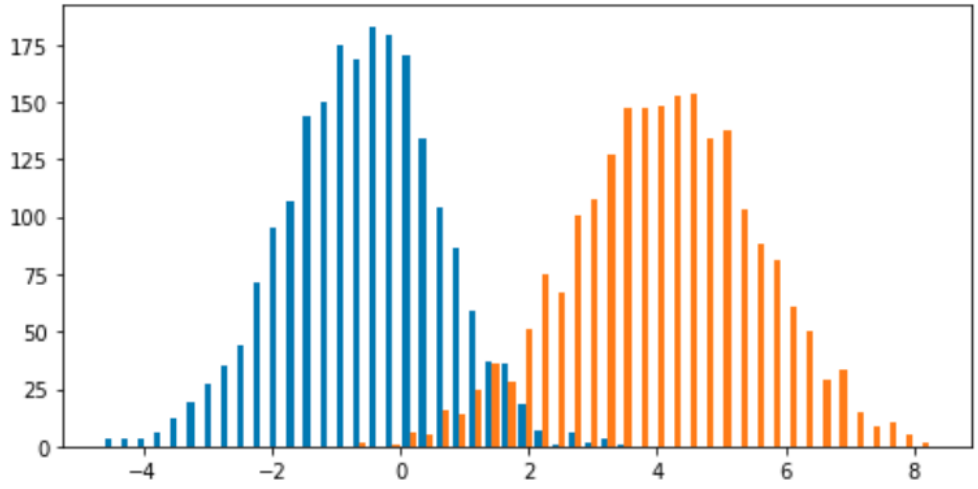
(1) I = 100, s = 1, separate 2 and 3.

```
Pairwise experiment, mapping 2 to -1, mapping 3 to 1
training error = 19.66%, testing error = 20.30%
Confusion matrix for Training Set is:
 [[1250  838]
 [   0 2175]]
Confusion matrix for Test Set is:
 [[1230  859]
 [   7 2169]]
```



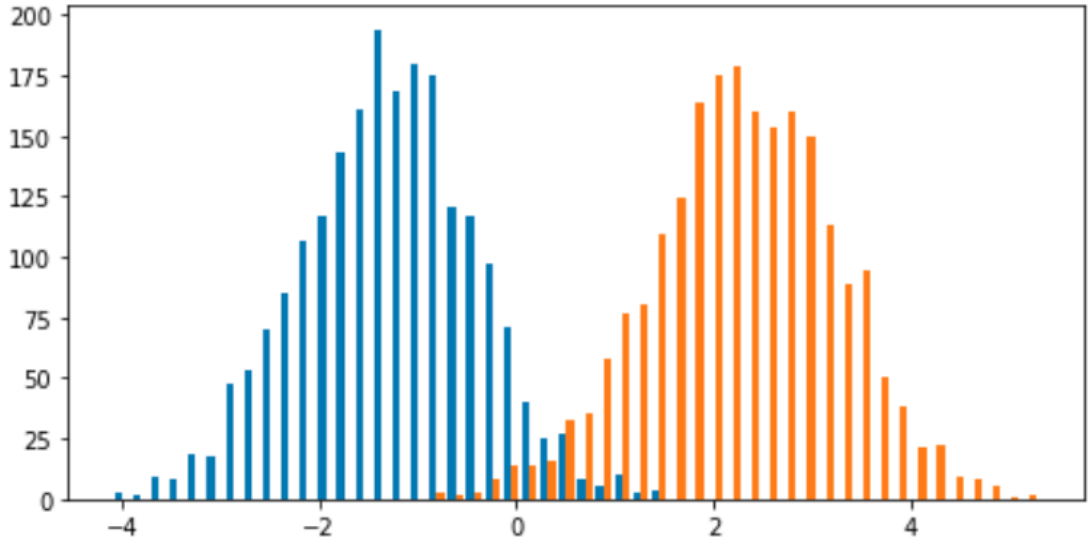
(2) I = 100, s = 0.1, separate 2 and 3.

```
Pairwise experiment, mapping 2 to -1, mapping 3 to 1
training error = 15.13%, testing error = 15.99%
Confusion matrix for Training Set is:
 [[1446  642]
 [   3 2172]]
Confusion matrix for Test Set is:
 [[1416  673]
 [   9 2167]]
```



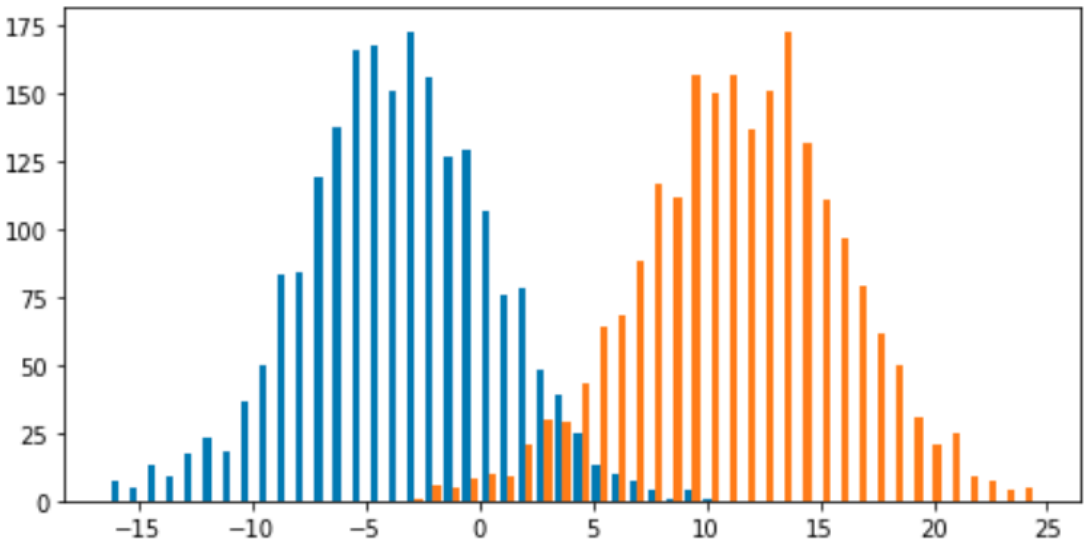
(3) I = 100, s = 0.01, separate 2 and 3.

```
Pairwise experiment, mapping 2 to -1, mapping 3 to 1
training error = 3.40%, testing error = 4.10%
Confusion matrix for Training Set is:
 [[1968  120]
 [  25 2150]]
Confusion matrix for Test Set is:
 [[1949  140]
 [  35 2141]]
```



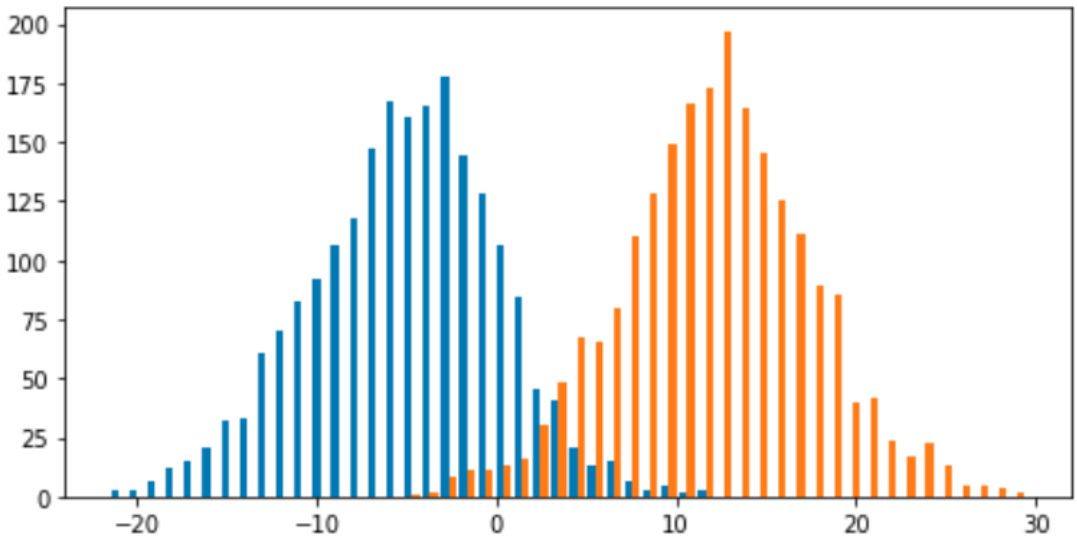
(4) I = 500, s = 1, separate 2 and 3.

```
Pairwise experiment, mapping 2 to -1, mapping 3 to 1
training error = 10.58%, testing error = 10.39%
Confusion matrix for Training Set is:
 [[1637  451]
 [   0 2175]]
Confusion matrix for Test Set is:
 [[1672  417]
 [  26 2150]]
```



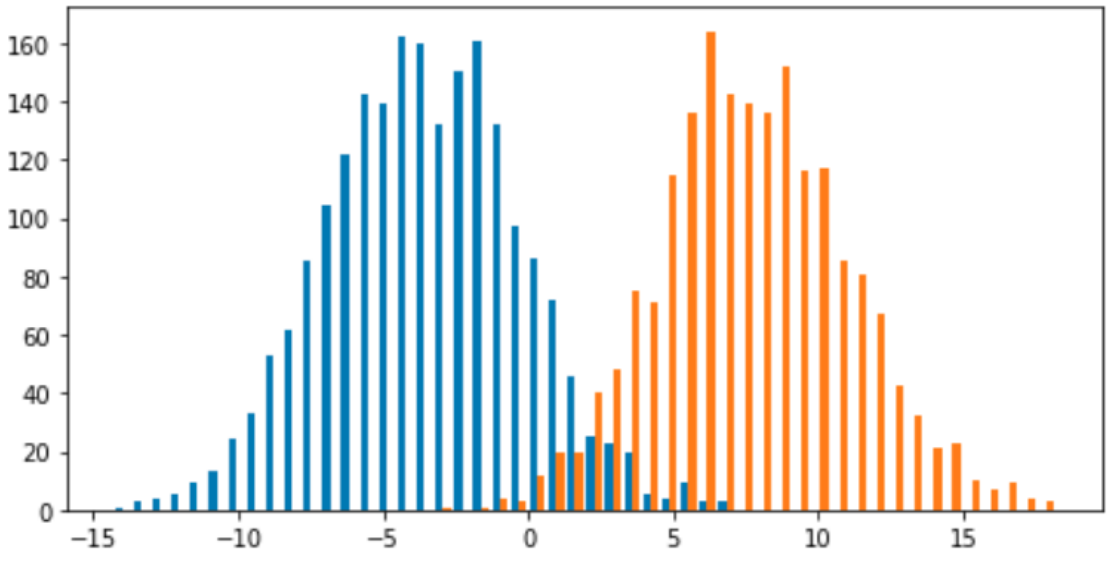
(5) I = 1000, s = 1, separate 2 and 3.

```
Pairwise experiment, mapping 2 to -1, mapping 3 to 1
training error = 6.92%, testing error = 8.56%
Confusion matrix for Training Set is:
 [[1793  295]
 [   0 2175]]
Confusion matrix for Test Set is:
 [[1763  326]
 [  39 2137]]
```



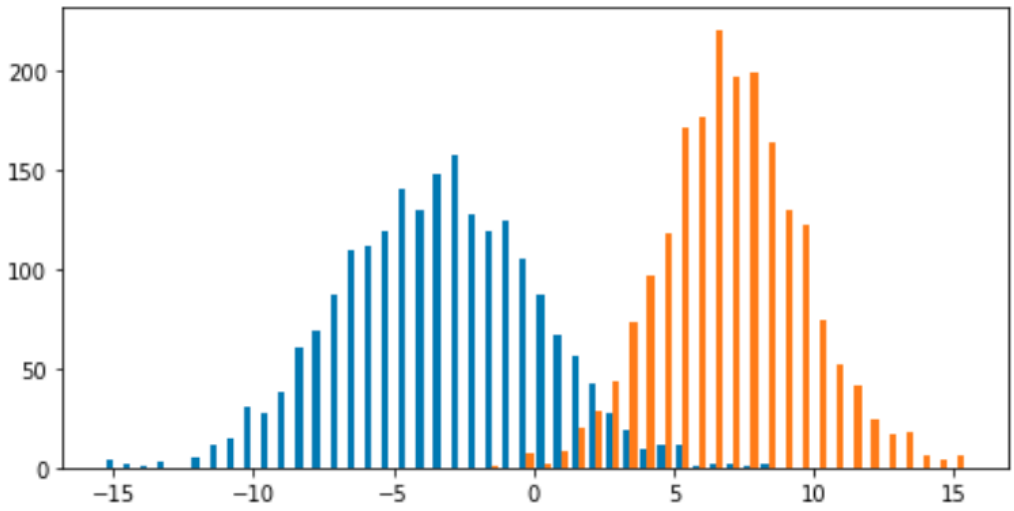
(6) I = 100, s = 1, separate 2 and 5.

```
Pairwise experiment, mapping 2 to -1, mapping 5 to 1
training error = 7.43%, testing error = 7.60%
Confusion matrix for Training Set is:
 [[1792  296]
 [   0 1897]]
Confusion matrix for Test Set is:
 [[1795  294]
 [   9 1889]]
```



(7) I = 100, s = 1, separate 2 and 8.

```
Pairwise experiment, mapping 2 to -1, mapping 8 to 1
training error = 8.98%, testing error = 8.66%
Confusion matrix for Training Set is:
 [[1718  370]
 [   0 2031]]
Confusion matrix for Test Set is:
 [[1743  346]
 [  11 2021]]
```



Is there any benefit to choosing s < 1? If so, why?

Answer: According to the above results of (1), (2), and (3), we can clearly see that if we choose a smaller step size (s), both training error and testing error will noticeably decrease, and I think this can be regarded as benefit to choosing s < 1. To explain why, I think maybe we can try to correlate the step size here with the learning rate when we are doing gradient descent. If our step size/learning rate is too large, then it is very difficult for our algorithm to find/reach the exact location of the optimal solution. If we decrease the step size/learning rate to an appropriate extent, although running the algorithm/model will be a bit slower, it is much easier for our algorithm to find/reach the optimal position.

In addition, according to the results of (1), (4), and (5), if we choose a larger number of full passes through the alternating projection, both training error and testing error will also decrease. And similar to decreasing s, increasing I will also make running the algorithm slower.

According to the results of (1), (6), and (7), it is also very interesting to note that given the same value of I and s, different linear classifiers separating different pairs of integers seem to have significantly different training and testing errors. To explain this, I think integer 2 looks kind of similar to integer 3, while integer 2 looks clearly different from integer 5 and integer 8. Therefore, it is very difficult for the linear classifier to separate 2 and 3, while it is relatively easy for the linear classifier to separate 2 and 5, as well as 2 and 8.

What happens to the test set performance when the error rate converges to zero for the training set?

Answer: When training error converges to zero, the testing error will first decrease because we increase the model complexity or increase the total amount of data samples. Then, however, the testing error will stop decreasing and begin to increase because of model overfitting. Therefore, even if the training error is likely to converge to 0, it is impossible for the testing error to be exactly 0, because of the existence of model overfitting. With some methods, model overfitting can be alleviated to some extent, but can never be eliminated.