### Problem.1



Error rate drops with iteration number. However the error rate drops much faster for the training set than the validation set (i.e. for training set, the error rate drops down to 0 in the end whereas the error rate for validation set only drops down to ~0.225). This is expected because the criterion is based on the training set, and the algorithm keeps going until the training error is 0. There are also fluctuations in the error rate as iteration increases. This is also expected because, depending on the order of the sample data, it is conceivable that sometimes, data near the end of the sample set could alter the b and w in such a way that they become less optimal for the earlier data.

The learning rate should have no effect on the end performance of the algorithm. Since we start with w\_0 = 0, then the final w is just a linear combination of the sample points. Since we’re only comparing with 0, the magnitude of the first part does not matter at all. For example, if we have and , then we can just factor out a from both and compare with 0. This makes no difference.

There should be no effect on accuracy because the two algorithms are essentially doing the same thing (dual computes new w while evaluating the comparison statement, while primal computes new w inside the statement). In terms of efficiency, the dual could be made slightly more efficient. The basics is essentially the same: instead of keeping track of and updating the w vector every iteration, you do it for the vector. However, using the vector allows for optimizations. Since only one chances per iteration, we can essentially get the sum in time instead of time by keeping track of the previous sum.



The smallest validation error rate is achieved at C=8. Using the SVM classifier method, the results on the validation set is slightly better (20.5% error rate vs. 22.5% error rate). For the training set, the error drops as C increases. This is expected because a higher C, by design, reduces the error rate of the training set. However, for validation set, it is slightly different. Initially the error rate drops. After its minimum at C=8, the error rate increases a little. This result is not surprising either because increasing C too much is like over fitting in the decision tree model. By having a high C, we are fitting the model too tightly around the training data, and this is not necessarily correct for the validation set.







Train has the best performance, followed by train1, train2, and lastly train3. This behavior is based on the way that the samples are reordered. Train alternates positive and negative classifications. Train1 alternates every 5. Train2 alternates every 10. And Train3 alternates every 500.

Alternating less in the training samples is bad for the update. The update depends on the label: it updates positively or negatively according to the sign of the label. So, if we provide all the positive samples first, the w vector is just going to get longer and longer, and b will simply increase, until we get to all the negative samples, when the w vector will only get shorter (and then longer in the negative direction) and b will simply decrease. Essentially, this procedure tries to adjust w and b in a much coarser manner than if we alternate the positive and negative samples.

Using C=8 to run SVM on the four files, I found that the validation error rate across all four files are a constant 0.205 and the training error rate across all four files are a constant 0.009. This is expected because the SVM classifier produces only one w value (the optimal linear classifier) regardless of the order. Therefore, it doesn’t matter how the data is ordered, SVM classifier will always produce a consistent result.

1. Using MATLAB, I calculated that the SVM exclusive number of errors is , and the perceptron primal exclusive number of errors is . Assume that the algorithm performs equally well, then we know that and are binomially distributed with .

The null hypothesis, where is binomially distributed, is with , .

We calculate

Since we know , we can reject the null hypothesis with 95% confidence. That is, we can say with 95% confidence that the error rate of SVM is significantly different from that of perceptron primal classifier.

I think the result of the hypothesis test is valid and correct because the SVM is correct almost two times as much as the perceptron primal.

### Problem.2

1. We claim that the new dataset is linearly separable by an unbiased plane with normal vector.

To prove the claim, we look at. By definition of dot product, we have:

For, we know, and for. We also know that by our construction, is same as for the first coordinates. Thus, we know:

We also know that by the indicator construction,  is 0 except when and. Effectively, this means that the second part of our summation is merely the coordinate of multiplied with, which is. That is,

Thus, we have:

We notice that is simply in our original case. We know from question statement that. Thus, we have.

For, we notice that since is 0 except when. Thus, the second summation is 0. Also, since for, we have:

Again, this is also simply in our original case. Thus, we have.

As we have covered all cases for, we can conclude that. In other words, our constructed indeed separates

1. We firstly notice that the margin we calculated for part a is actually a functional margin, as . Thus, the geometric margin can be calculated as:

With that, we can express the mistake bound for as:

Firstly, we could bound with, since we merely swapped one coordinate out with the value k. That is, the maximum of can be no more than the norm of added another coordinate with value, which is square root of plus. That is,

As for the length of, we could bound it by and , where . Since in the problem statement we were given that . We can further simplify to:

Thus, we know that the mistake bound for is simply:

If we notice that , since . The mistake bound could be further loosening to:

### Problem.3

1. The leave-one-out error for the first instance is 0. Since, first instance is not a support vector, and therefore does not affect the optimal hyper-plane if left out.

For instances 2 and 3, we firstly calculate slack variables, and. We also know that the maximum length of feature vector is 1, that is,. We can then calculate the necessary conditions of leave one out error as:;. Thus, instance 2 cannot produce a leave-one-out error, while instance 3 could produce a leave-one-out error.

Thus, the upper bound of leave-one-out error for the three instances is 1.

1. From the dual training algorithm, in each iteration of the algorithm, we will add 1 to if. From the problem statement, we know that any two feature vectors are orthogonal, then we know that the summation,, could be simply expressed as . That is, we are adding 1 to only when. We notice that when starts at 0, the inequality will be satisfied, and if, the inequality will never be satisfied. Therefore we will add 1 to, and never change the values in the later iterations. Thus, the corresponding to is 1, for all.
2. We claim that C should be changed by a factor of so the new solution would define the same linear classifier. We denote the scaled version with \*, then we claim:

We can see this from the dual problem. In the dual problem, we defined the objective function as:

We could multiply a factor of to the objective function, and the optimal solution of should not change. That is, if we change the objective function to the following, the optimal solution should be exactly the same as :

Now consider the modified version of the problem, where we set , the new dual problem can be represented as:

Subject to:

This could be expressed as:

Subject to:

We immediately notice that from (\*), is a solution to this maximization problem with optimal value . We also notice that since we scaled the bound of decision variables by of the original problem, by linearity, one could not get a solution better than . Thus, this indeed generates the optimal solution to the new dual problem.

Then, we could conclude that . Therefore, if we change to , the optimal hyperplane is still the same as before. That is, it is still the same classifier. The resulting