ELE888 Intelligent Systems

Lab 2: Linear Discriminant Functions

Author: Bethany Santos

# Introduction

The objective of this lab is to determine the Linear Discriminant Function (LDF) and create a Linear Discriminant Classifier (LDC) from an iris data set obtained from the following link:

https://archive.ics.uci.edu/ml/datasets/Iris

In order to determine the decision boundary, an iterative process will be performed to determine the *weight vector* using the descent algorithm approach.

# Theory

## Linear Discriminant Function

A linear discriminator can easily be built using LDFs. LDFs do not require knowledge on the underlying probability densities of the given data, instead it can be computed using a training data sample and a criterion function.

A linear discriminant function g(**x**) with feature vector **x** can be expressed as:

(1)

where **w** is the weight vector andis the bias or threshold. By observing the output of the LDF, we can classify a dataset to either class 1 if or class 2 if .

A generalized form of LSD can be given as:

(2)

where the *augmented vector y* and *augmented vector a* are given by:

(3)

(4)

The goal here is to obtain the solution vector from a training set to create the LDF stated in (1). The solution vector can be obtained by performing an iterative process approach using perceptron criterion with the gradient descent algorithm described as follows:

1. begin initialise: a, criterion θ, η (.), k = 0

2. do k ← k + 1

3. a ← a − η(k) J(a)

4. until |η(k) J(a)| < θ

5. return a

6. end

# Results

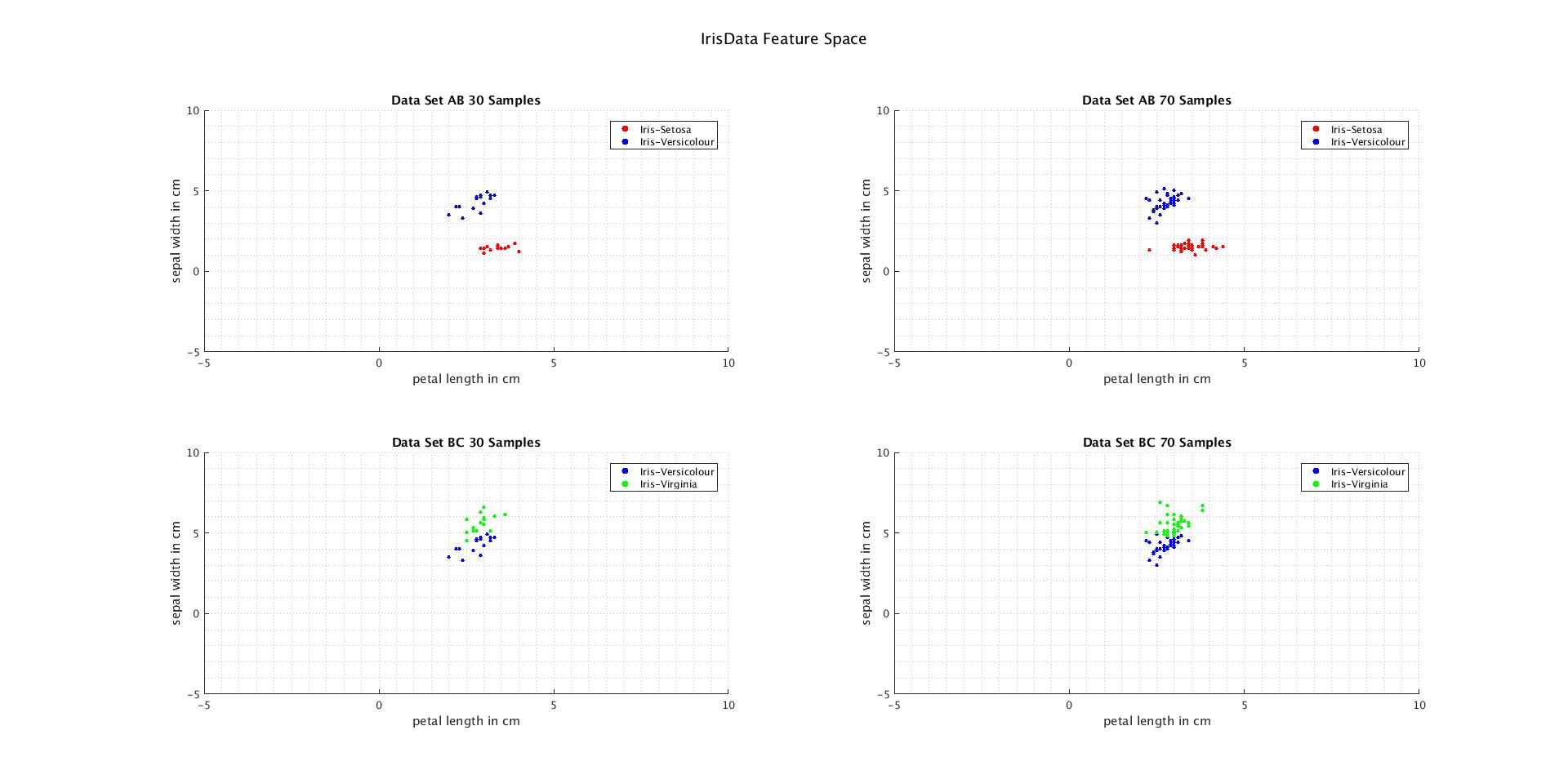
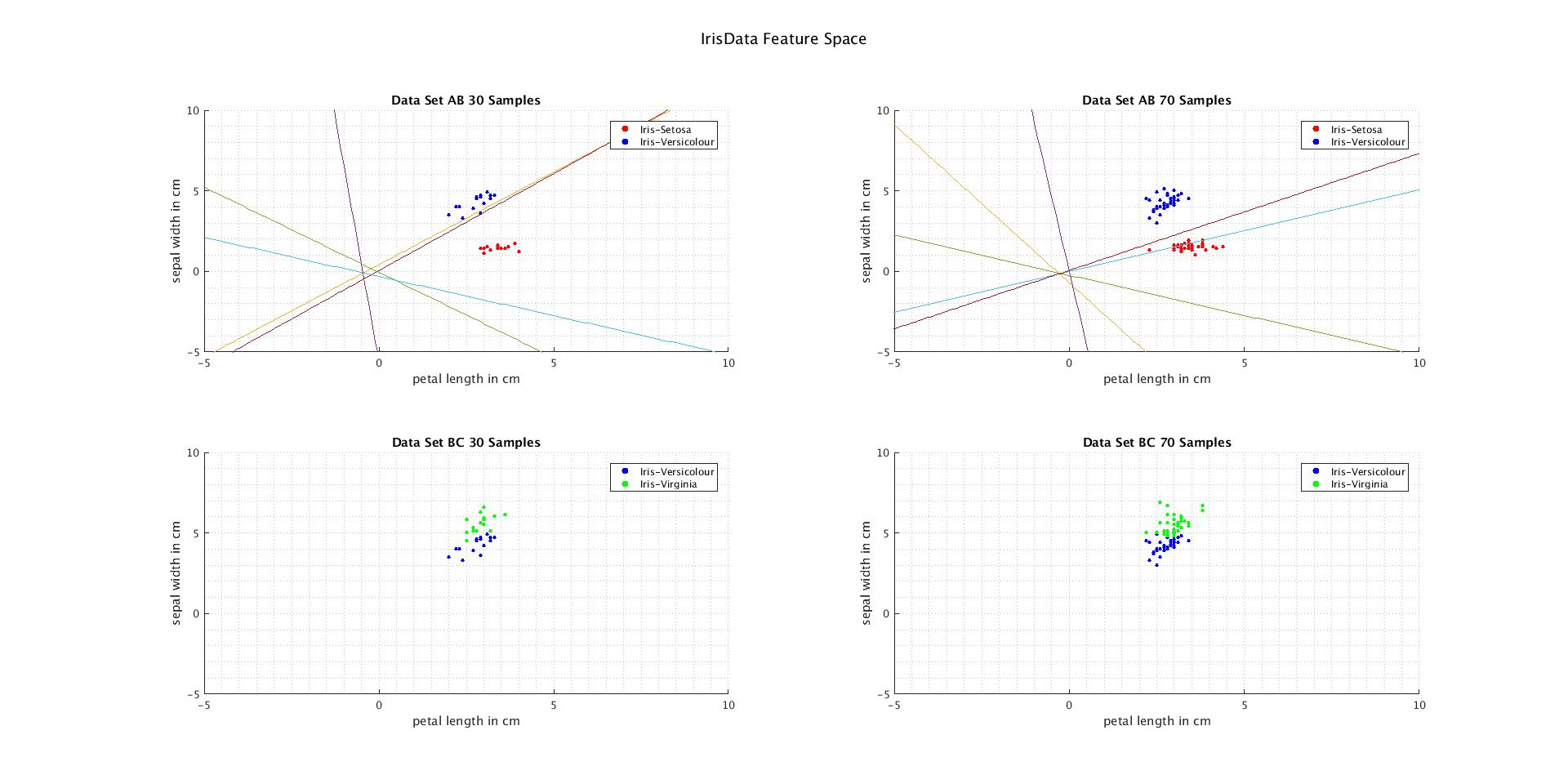


Figure 1: Data Set AB 30 samples feature space

Figure 2: Data Set AB 30 samples feature space and decision boundary iterations

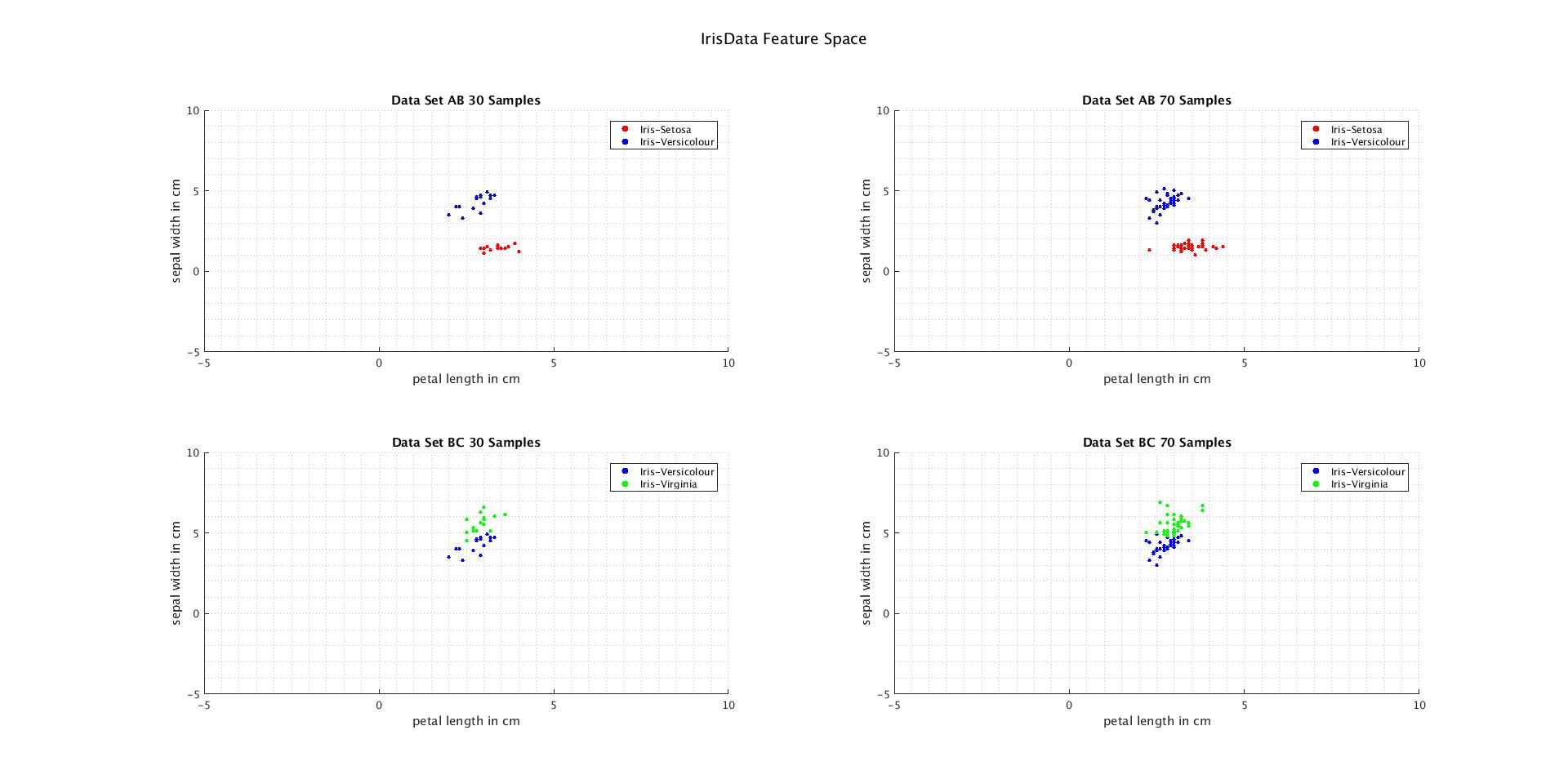


Figure 3: Data Set AB 70 samples feature space

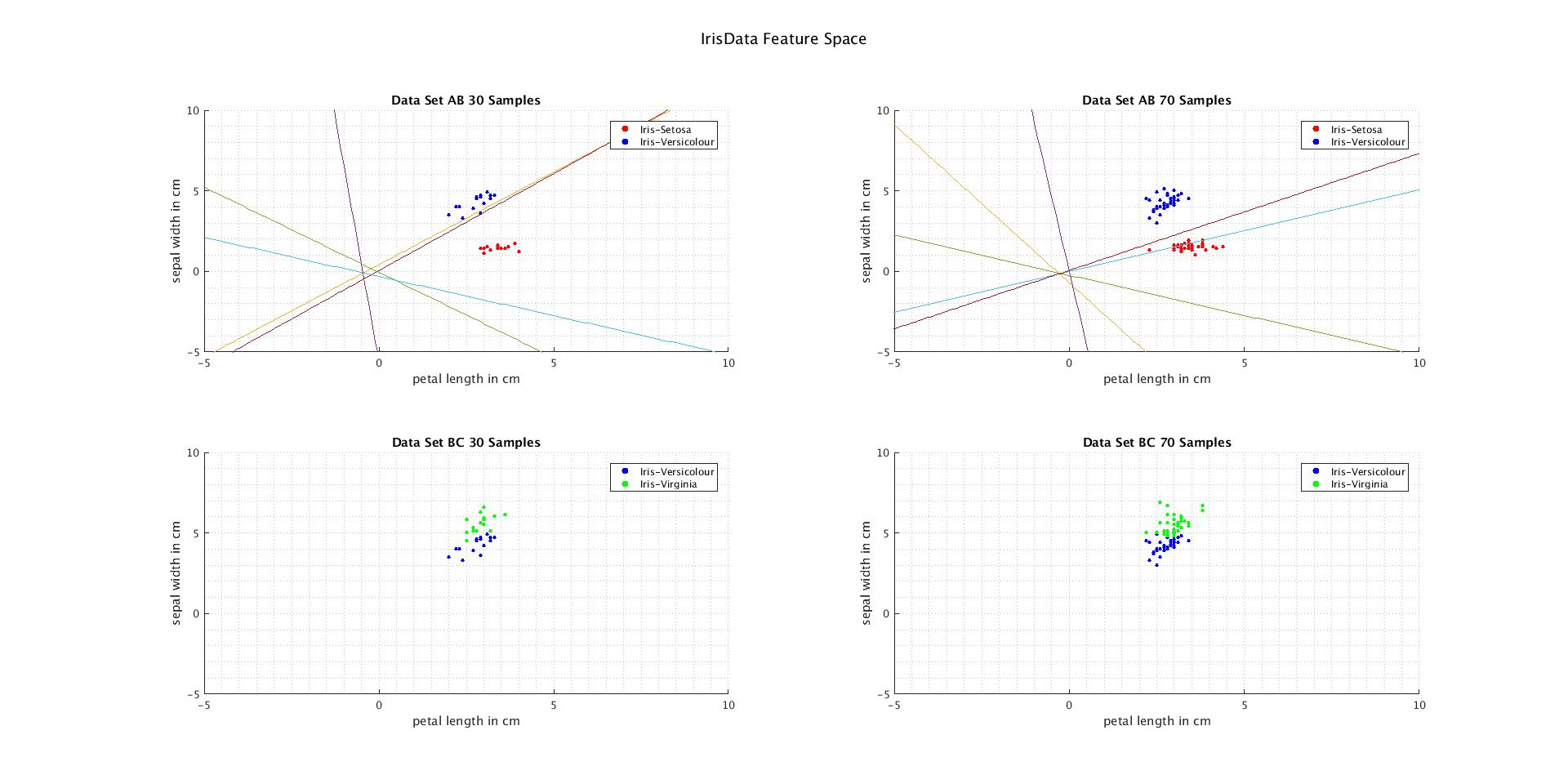


Figure 4: Data Set AB 70 samples feature space and decision boundary iterations

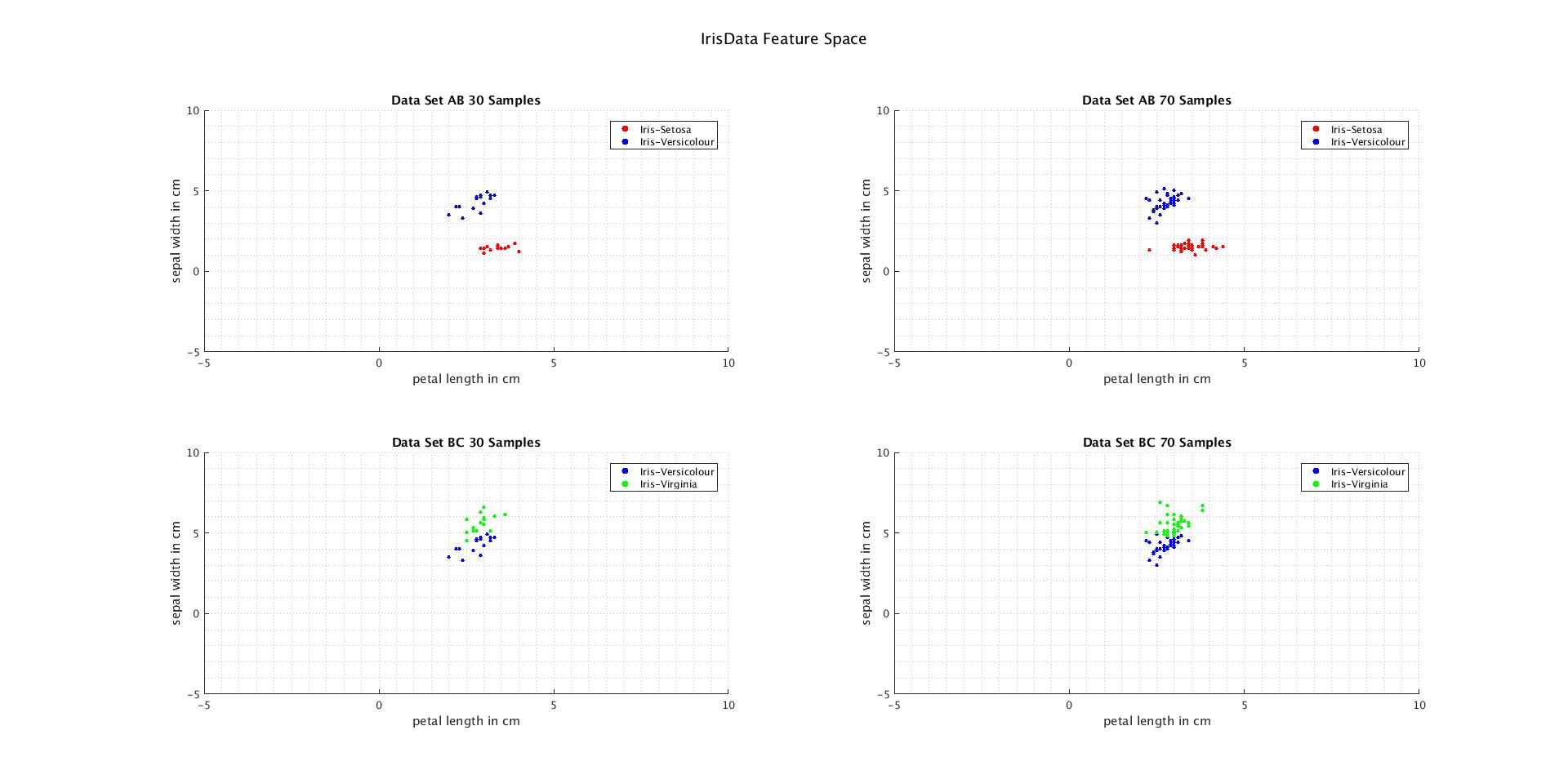


Figure 5: Data Set BC 30 samples feature space

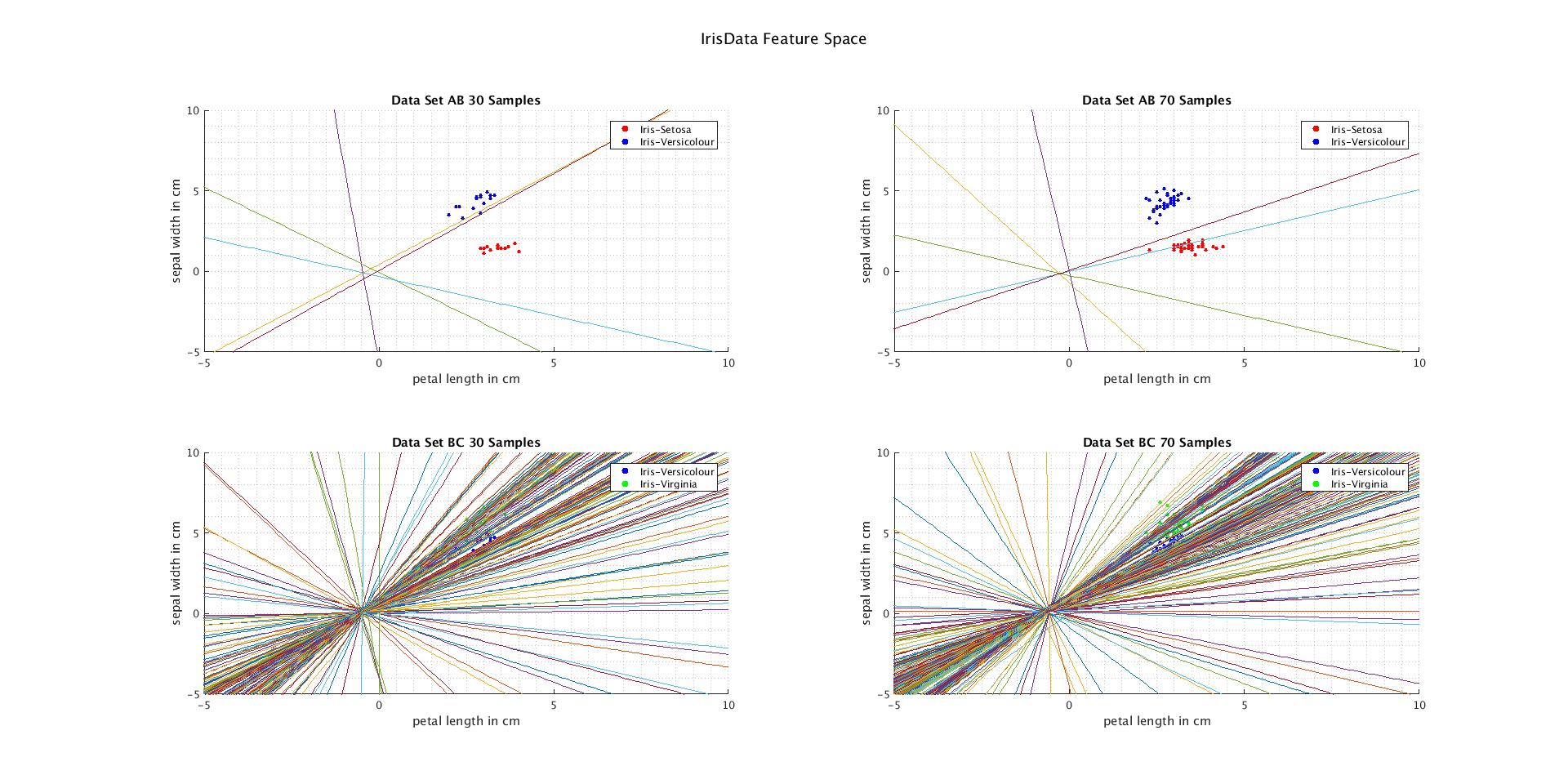


Figure 6: Data Set BC30 samples feature space and decision boundary iterations (300 = max)

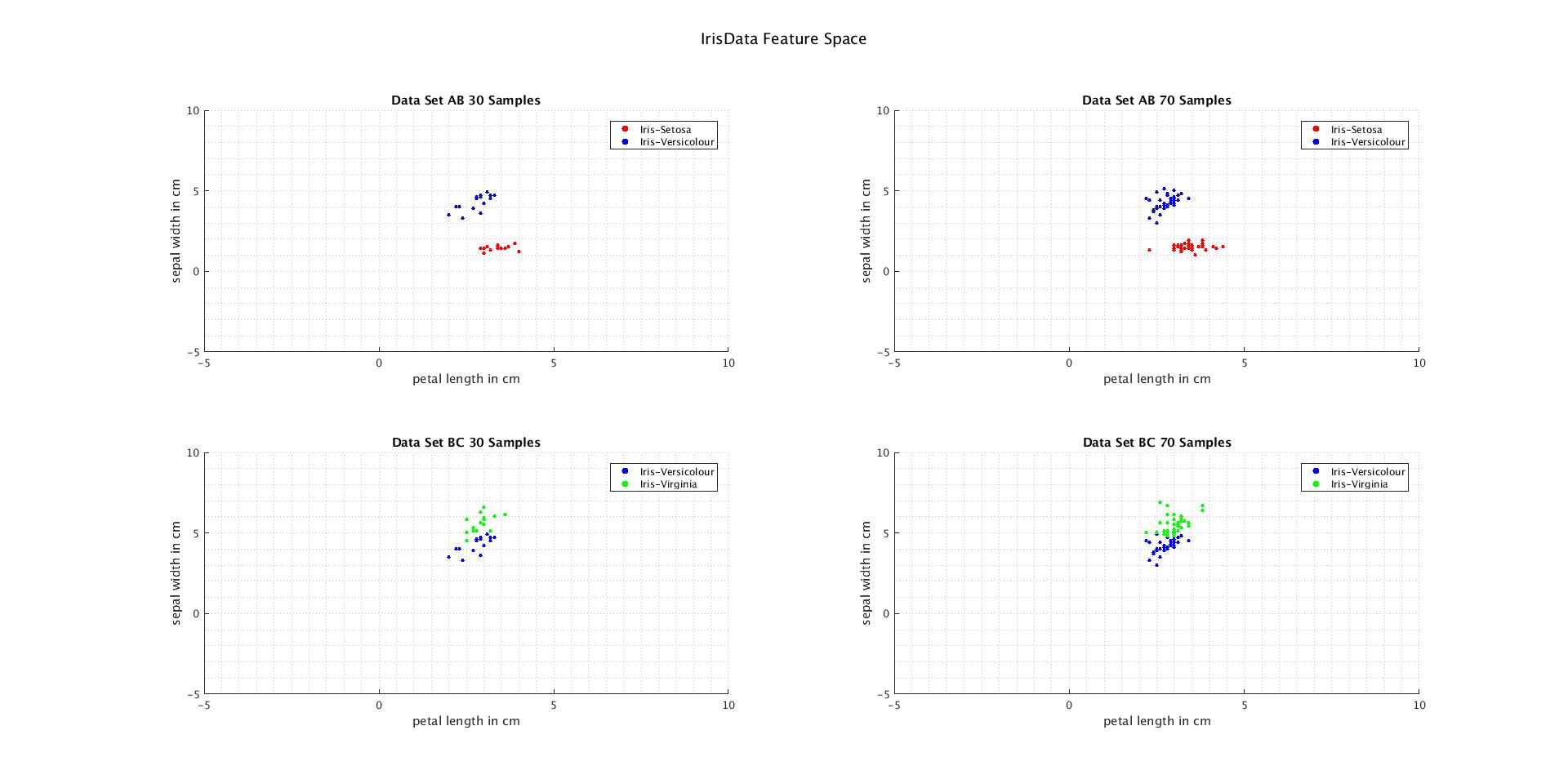


Figure 7: Data Set BC 70 samples feature space

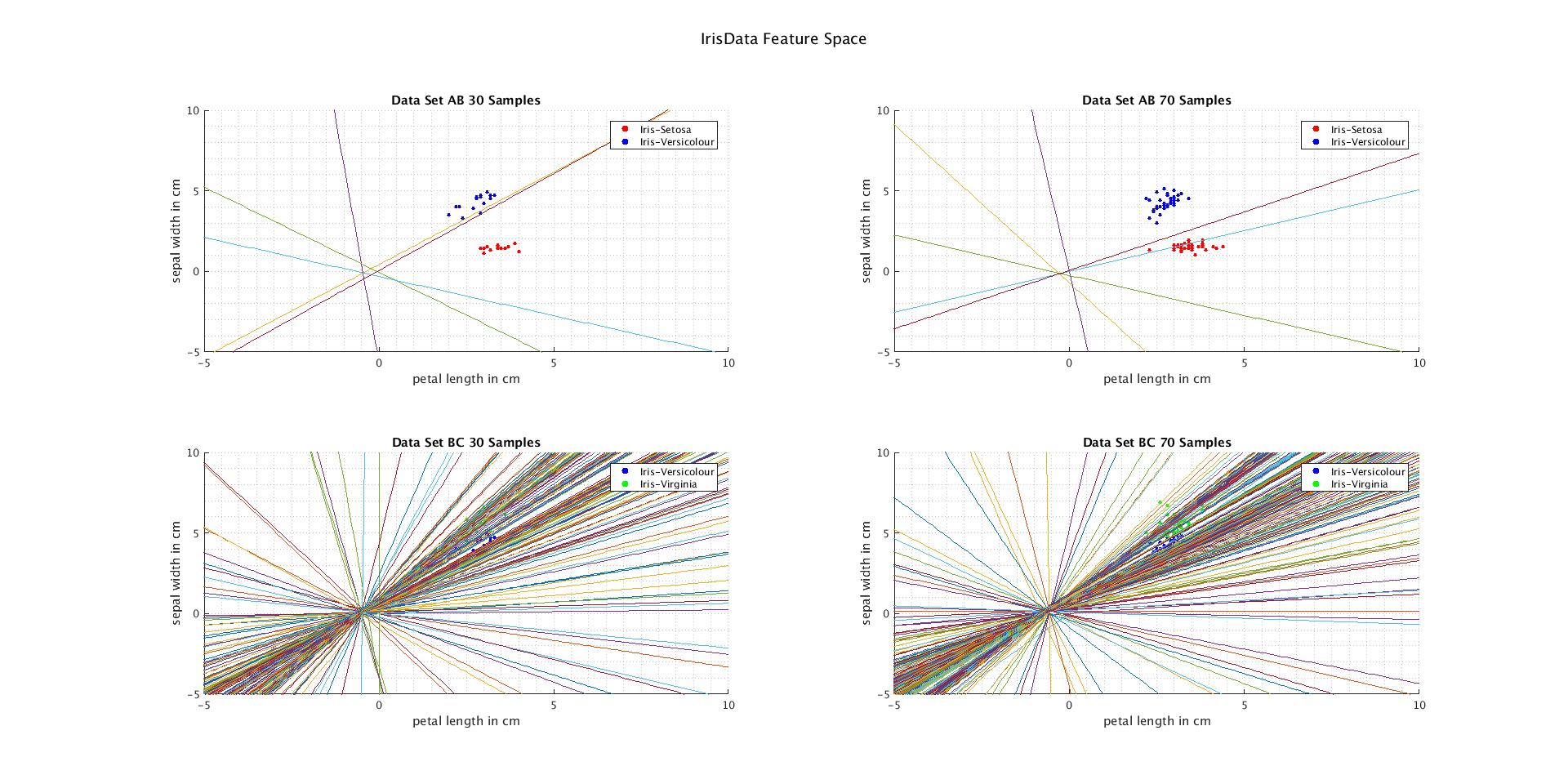


Figure 8: Data Set BC70 samples feature space and decision boundary iterations (300 = max)

|  |  |  |  |
| --- | --- | --- | --- |
| Training Set | Solution Vector a | # of Iterations | Accuracy (%) |
| AB30 | [0.010 0.284 -0.236] | 5 | 98.5 |
| AB70 | [0.05 0.629 -0.867] | 5 | 100 |
| BC30 | [2.309 3.228 -2.435] | 300 | 84.3 |
| BC70 | [6.920 7.374 -5.549] | 300 | 93.3 |

Table 1: Column 1:Training Set used. Column 2: solution vector where [. Column 3: number of iterations it took to obtain solution vector a. Column 4: accuracy of the LDF computed from the respective training set.

# Discussion

*1. Effects of varying training data and testing data set sizes.*

Looking at Table 1, there is a clear distinction that using a training data set with larger number of samples, we are able to obtain a better LDC with a higher accuracy than when using a training set with a smaller number of samples. This behaviour is analogous to human behaviour when learning. As we gain more information and experience about something we can predict and analyse similar experiences and react faster to them.

*2. Effects of different learning rates, threshold, initial weight values.*

The learning rate describes how well the descending algorithm corrects itself every iteration to arrive at the solution vector. The better the learning rate the less iterations the algorithm needs to reach a solution vector. There seems to be a converging value for the number of iteration as the eta value increases (looking at Table 2, this value seems to be 5), also with small values of eta, it takes a considerable number of iterations to reach a solution vector. This can be explained by looking at the step 3 in the descent algorithm. Eta or the learning rate is multiplied by a non changing normalized y vector to create a new weight vector for the next iteration. If the changes in each iteration is small then it will require more repetitions to reach solution (takes smaller steps to reach destination). Since the solution vector is directly proportional to the learning rate, as the learning rate increases, so does the values of the weight vector.

Furthermore, if our initial weight vector has values too far off causing an overshoot, then this will cause confusion through the iteration making it difficult to settle to a solution vector. Table 3 shows that when the initial weight vector is [100 100 100] it requires 247 iterations while the other smaller initial weight vectors requires substantially less. This means that weight vector [100 100 100] is not most suited to start with.

|  |  |  |
| --- | --- | --- |
| η | Solution Vector | Iterations |
| 0.0001 | [-0.0498, 0.0191, -0.0034] | 248 |
| 0.01 | [0.0100, 0.2840, -0.2360] | 5 |
| 1 | [15.00, 67.20, -62.50] | 5 |
| 10 | [150.0, 672.0, -634.0] | 5 |
| 10000 | [1.500, 6.720, -6.350] e4 | 5 |

Table 2: Resulting solution vector and number of iterations for different eta values for AB30 training set with initial weight vector = [0 0 1].

|  |  |  |
| --- | --- | --- |
| Initial weight vector | Solution Vector | Iterations |
| [0.001 0.001 0.001] | [0.1510, 0.6730, -0.6340] | 5 |
| [1 1 1] | [0.650, 0.0360, -0.4590] | 3 |
| [100 100 100] | [68.95, 13.91, -31.02] | 247 |

Table 3: Resulting solution vector and number of iterations for initial vector values for AB30 training set with eta = 0.01.

*3. Criterion function over the iteration.*

The criterion function used in this lab is the perceptron criterion which uses a set of misclassified data to iterate through the process of the descent algorithm. When the criterion presents no negative values (meaning no miss classified data), then the a vector used in the criterion is the solution vector. The perceptron criterion is a linear piecewise criterion which works well when data set is linearly separable. For example, it only took 5 iterations to reach a 100% accurate decision boundary using AB70 as the training dataset as seen in Figure 4. This criterion does not work well when the data set is not linearly separable as seen Figure 6 and Figure 8 where it reached the maximum iteration count of 300 (manually set in the code) because it was unable to settle on a criterion with no miss classified data. It is clearly seen from Figure 5 and Figure 7 that the classes are not linearly separable.

# Conclusion

In conclusion, we are able to create an LDC with a higher accuracy when there is a larger sample data set to train with and that the learning rate and initial weight vector are critical parameters in which can help us optimize iterations when determining the solution vector. It was also determined that the perceptron criterion is most useful when the training data set is linearly separable or else it will take longer for the algorithm to settle to a solution vector.