Aditya Bhatkar

Decisions & Notes

1. Performance graphs are mentioned in this document.
2. The code is generic, can be used for different sizes of input and weights.
3. Only first line of main.m needs to be edited to change data size.
4. Two separate loops are mentioned in main.m one for using pseudo inverse and other for nonpseudo inverse. This is done solely for demonstration purpose.
5. Code for plotting results is in the same file.
6. Randomizations is applied at every selection.
7. I think instead of picking random misclassified point, if we select the closest point to previous line then we can converge in fewer iterations. Also it was noticed that sometimes after applying pseudo inverse the number iterations increased instead of decreasing. I think if we make optimum choice as mentioned above instead of random the problem can be solved.

Algorithms

1. [W, iters]=pla(X, Y, W)

Input:	data matrix X (N X 3)
	Label matrix Y (N X 1)
	Weight matrix W (1 X 3)
Output:	learned weight matrix W (1 X 3)
	Number of iterations required for learning

Pseudocode:

```
While all points are not correctly classified

Calculate dot product of X and W

Compare result with given labels

If no misclassified point then exit

Else

Increment number of iters
```

```
Select a random miss classified point
Update weights
```

2. [X,Y]=generateData(N)

Input:	N integer
Output:	X (N X 3) data matrix in [-1, 1] range
	Y (N X 1) label matrix in [1, 0] range

Pseudocode:

```
Generate random number in rage [-1, 1]
Select two random points
Classify other points based on line formed by selected points
```

3. [W]=pseudoinverse(X,Y)

Input:	data matrix X (N X 3)
	Label matrix Y (N X 1)
Output:	weight matrix W (1 X 3)

Pseudocode

```
Use the pseudo inverse formula to generate new set of weights.
```

Aditya Bhatkar

My Observations

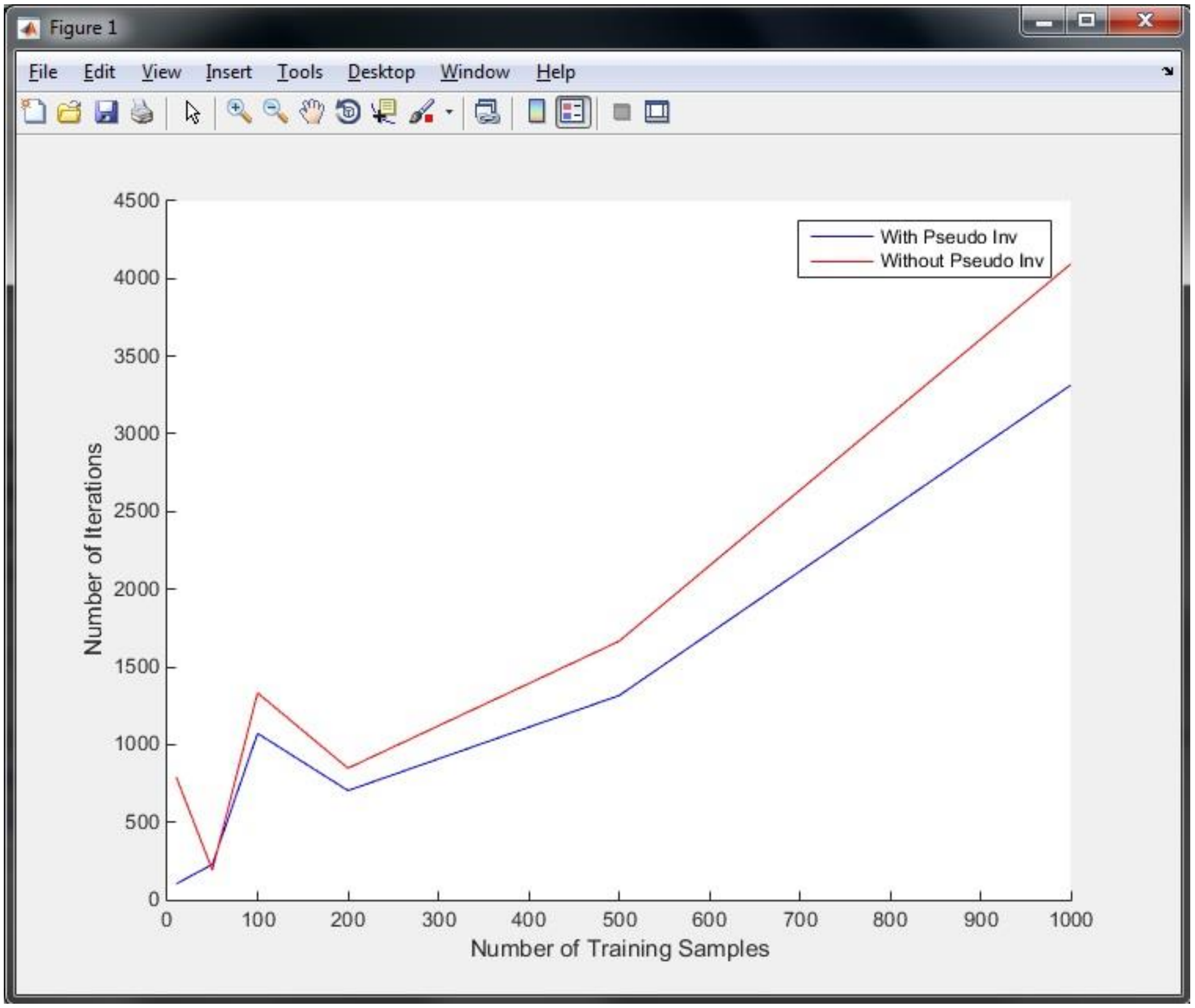
1. In most cases the number of iterations after using pseudo inverse decreased as compared to without using pseudo inverse.
2. Classification results were more consistent for larger data size.
3. Linear classification was achieved.

	With Pseudo Inverse		Without Pseudo Inverse	
N	Iterations	Time	Iterations	Time
10	102.79	0.239524	790.67	2.172468
50	228.42	0.568052	191.08	0.578279
100	1071.6	3.214357	1332.2	3.748959
200	703.54	2.137351	847.68	2.545709
500	1315.1	3.964678	1665.1	5.512501
1000	3314.5	12.37301	4094.7	15.435057

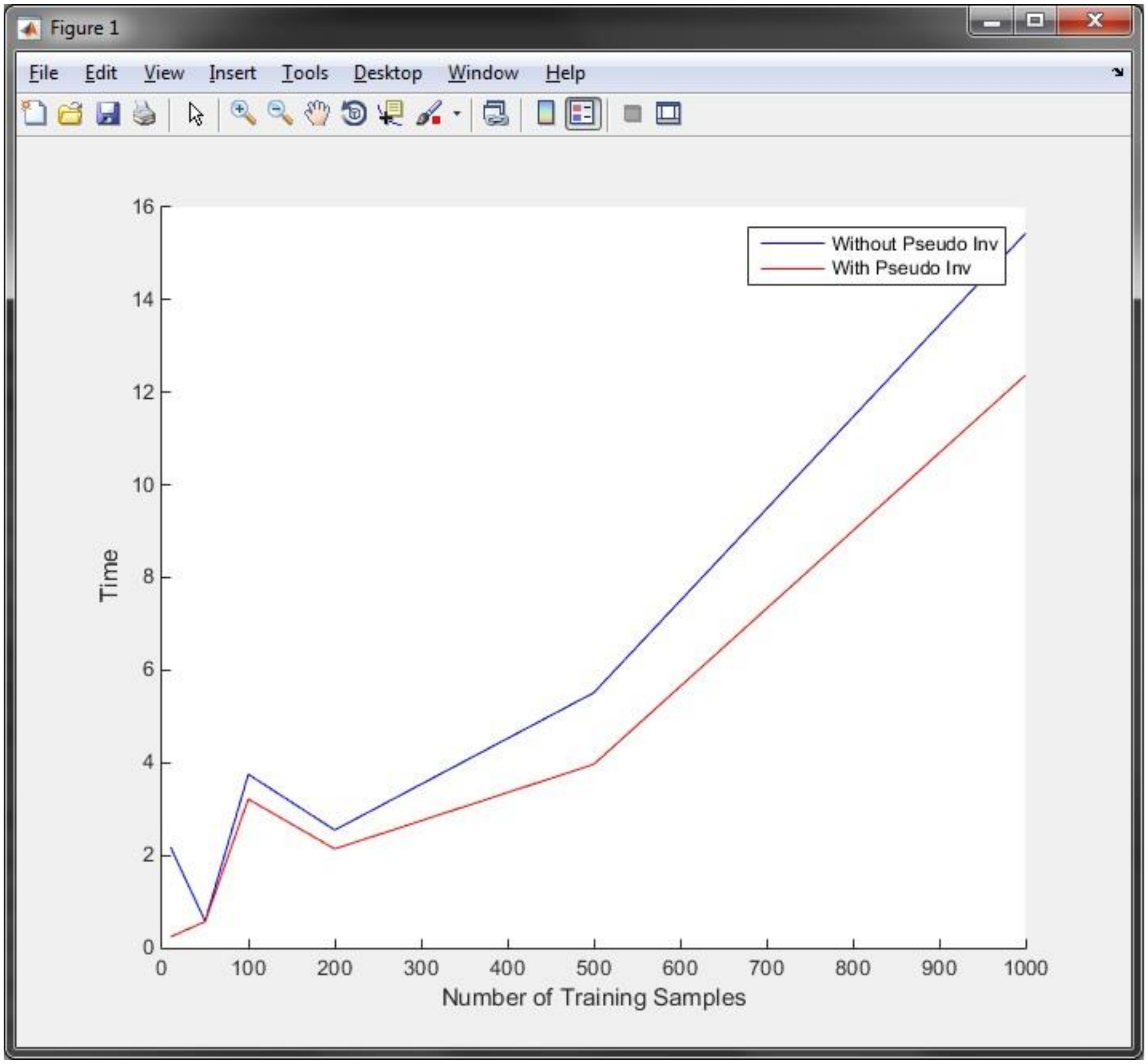
Aditya Bhatkar

Graphical Representation of Observations

1. Number Iterations with and without using pseudo inverse



2. Time consumed to complete operation with and without pseudo inverse.



Conclusion

Pseudo inverse can help reduce the number of iterations.

Using average of number of iteration can help to achieve consistency in results.