Introduction to Artificial Intelligence Lab Report

Raghuvar Prajapati
(201351003@iiitvadodara.ac.in)

February 17, 2016

Problem No-1 : Compute the value of E_{in}, E_{out} and W for the linear regression ?

Solution : The **Linear Regression** algorithm(Sometimes referred as Ordinary Least Square(OLS)) is based on the minimizing the squared error between h(x) and y.

$$\mathbf{E}_{out}(\mathbf{h}) = \mathbf{E}[(\mathbf{h}(\mathbf{x})\text{-}\mathbf{y})^2]$$

The goal is to find a hypothesis that achieves a small E_{out}(h).

$$\mathbf{E}_{in}(\mathbf{h}) = \frac{1}{N} \sum_{i=1}^{n} (\mathbf{h}(\mathbf{x}_n)\text{-}\mathbf{y}_n)^2$$

In linear regression, h takes the the form of a linear combination of components of x, that is :

$$\mathbf{h}(\mathbf{x}) = \sum_{i=0}^{d} \mathbf{w}_i\mathbf{x}_i = \mathbf{w}^T\mathbf{x}$$

Where x_0=1 and x $\in$ $\{1\}$*R^d as usual and w $\in$ R^{d+1}.

So, for the given dataset If we applied **Linear Regression** model we'd get following result :

- **Error_Measurement** = 0
- **Our new weights for the given dataset are :**
  (0.025827, 0.406082, -0.428349, -0.089038)
- **Accuracy = 1.000000**

Problem No-2 : Shuffle the training data (using a common seed, 1234) and then prepare train-test partitions according to (30-70, 40-60, 50-50, 60-40 and 70-30) percent sizes. Train on train partition and test on both. Report $\mathbf{E}_{in}$ and $\mathbf{E}_{out}$ for each partition.

Solution : For Perceptron If we shuffle the training data for the given IRIS dataset we would get different curve, E_{in} and E_{out} like :

1. First we'll take 70% of the training data and analyze it :

   (A) : $\mathbf{E}_{in}$ = 72 32 9 6 2 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.

   (B) : $\mathbf{E}_{out}$ = 21 8 5 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2.

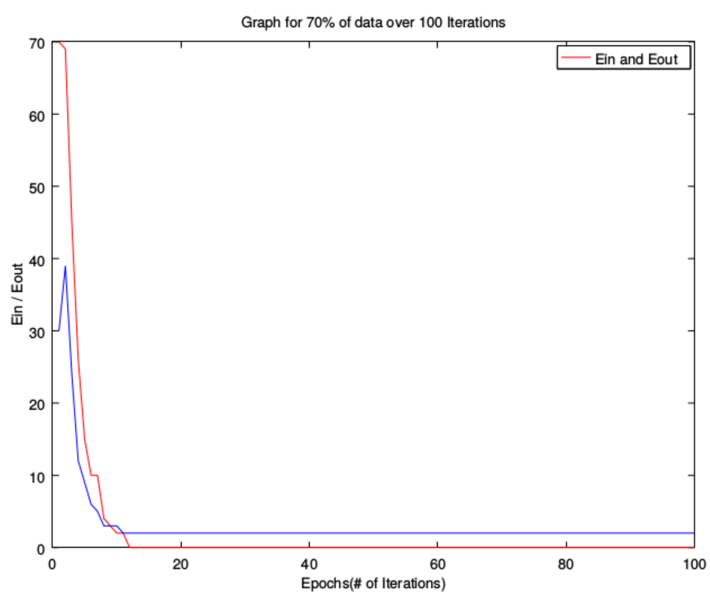

Figure 1: **PLA for 70% of Data**

2. Second we'll take 60% of the training data and analyze it :

   (A) : $\mathbf{E}_{in}$ = 54 23 10 5 5 3 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.

   (B) : $\mathbf{E}_{out}$ = 21 20 14 11 6 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3.

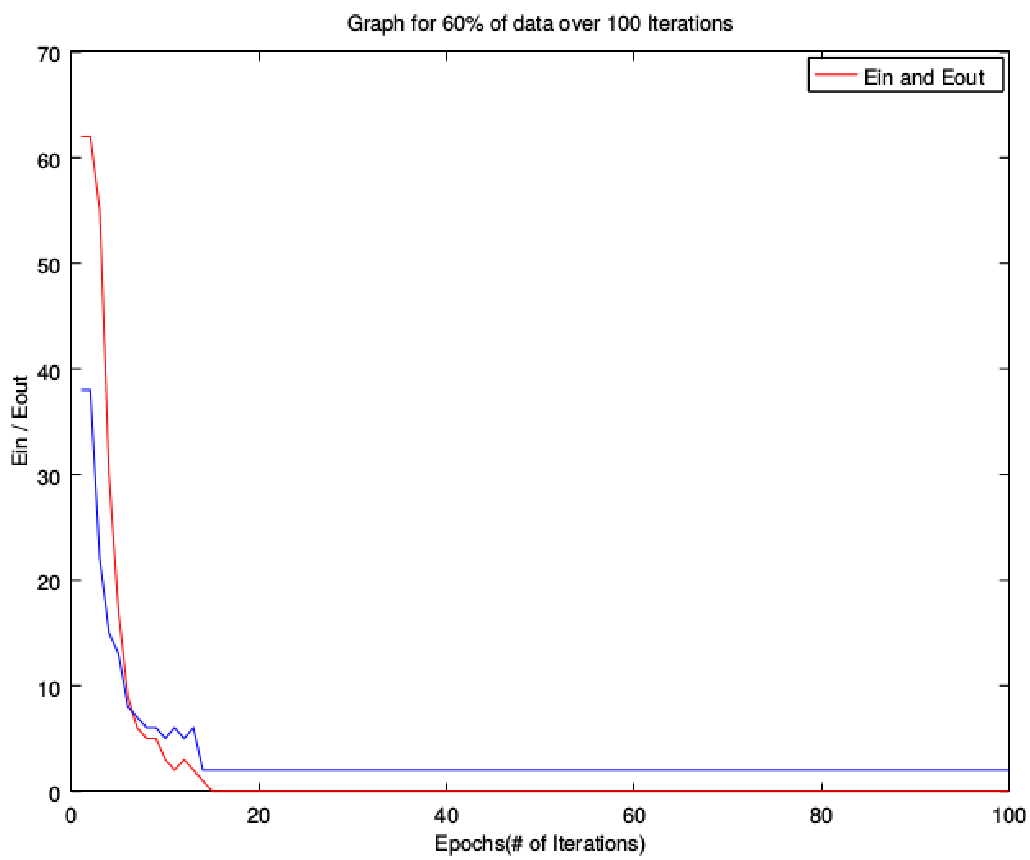

Figure 2: **PLA for 60% of Data**

3. Third we'll take 50% of the training data and analyze it :

(A) : $\mathbf{E}_{in}$ = 39 18 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 00 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

(B) : $\mathbf{E}_{out}$ = 42 8 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2.

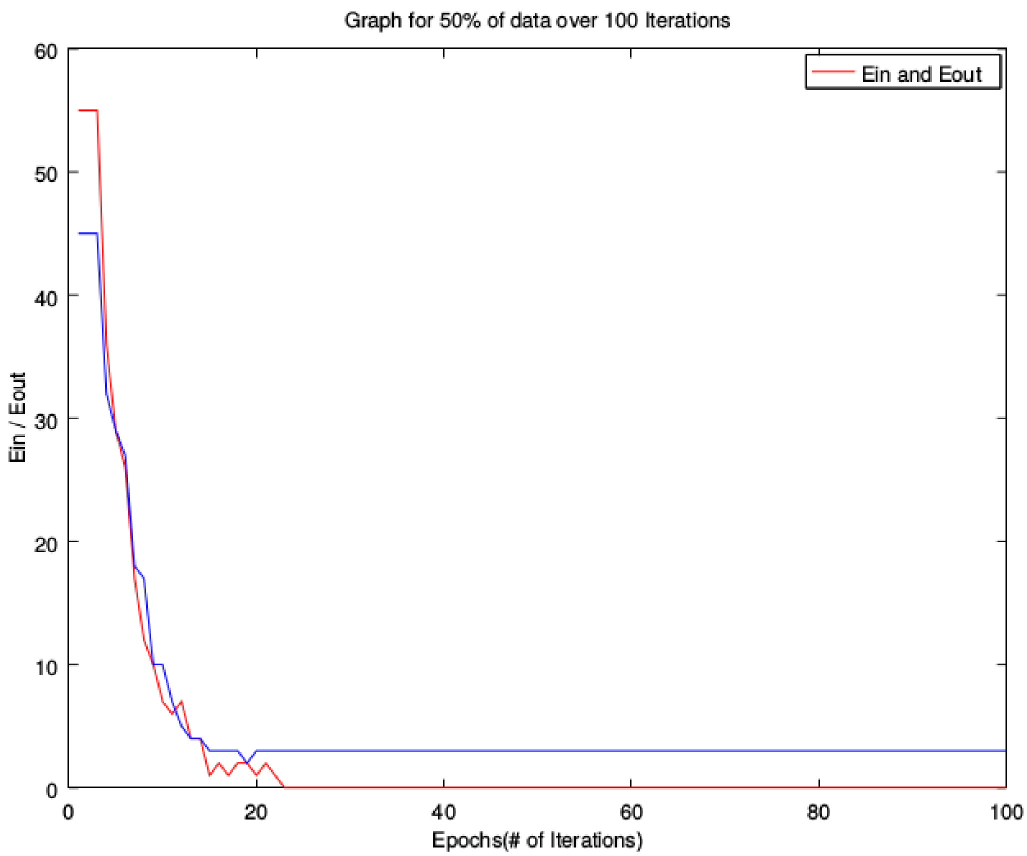

Figure 3: **PLA for 50% of Data**

4. Fourth we'll take 40% of the training data and analyze it :

(A) : $\mathbf{E}_{in}$ = 35 28 11 5 4 3 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

(B) : $\mathbf{E}_{out}$ = 65 65 65 56 47 34 21 16 15 14 13 11 9 8 7 5 3 4 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3.

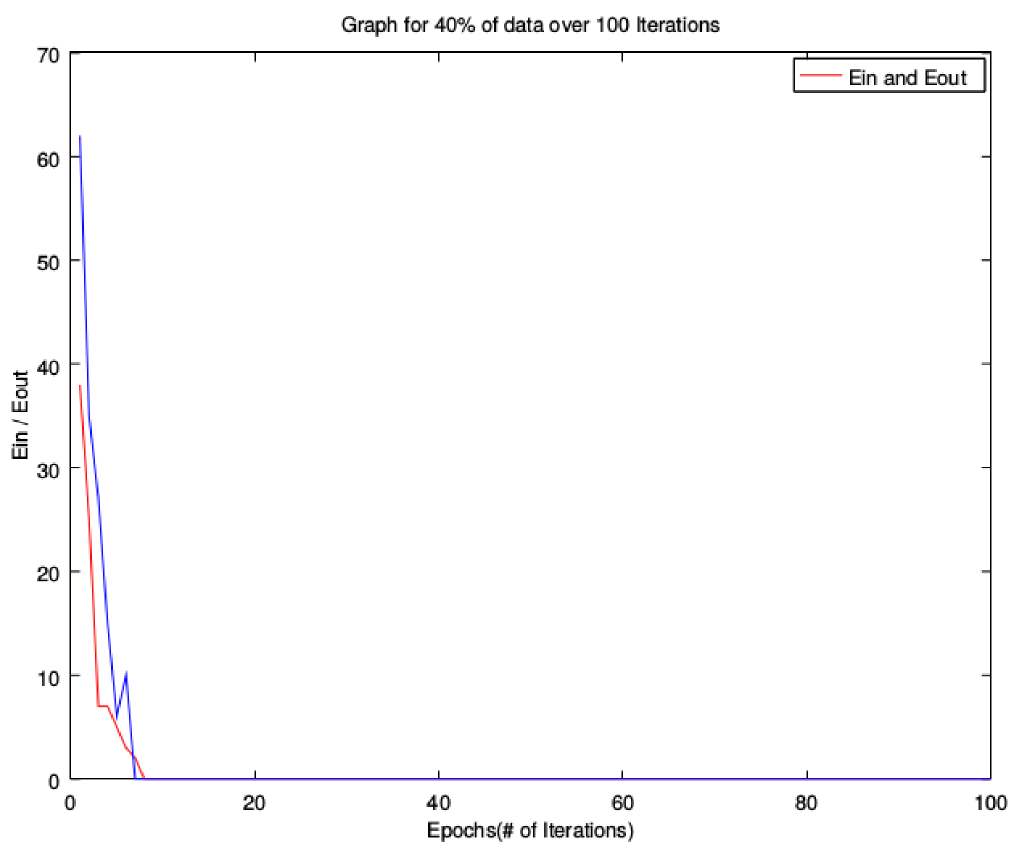

Figure 4: **PLA for 40% of Data**

5. Fifth we'll take 30% of the training data and analyze it :

(A) : $\mathbf{E}_{in}$ = 24 24 21 22 12 5 4 3 3 2 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.

(B) : $\mathbf{E}_{out}$ = 76 76 54 38 25 17 16 14 15 9 8 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11.

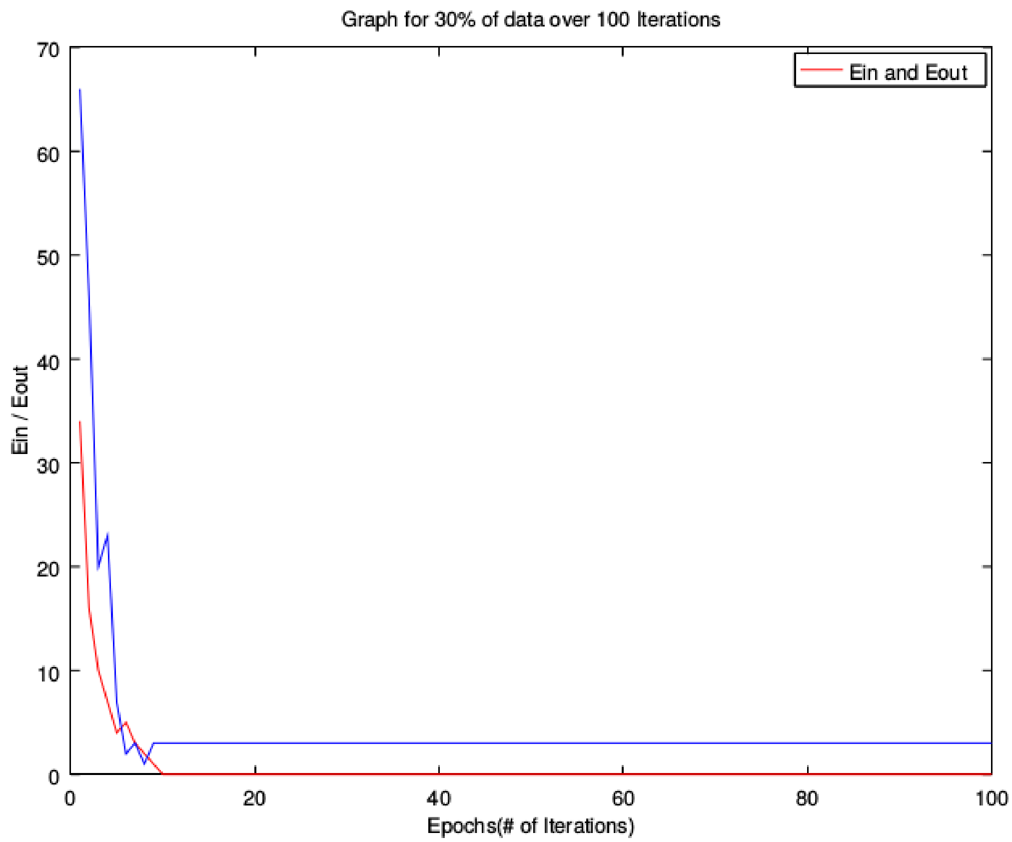

Figure 5: **PLA for 30% of Data**

Problem No-3 : Implement Pocket algorithm for PLA. And plot learning curves for different cases ?

Solution : The pocket algorithm with ratchet (Gallant, 1990) solves the stability problem of perceptron learning by keeping the best solution seen so far "in its pocket". So after implementing pocket algorithm on the IRIS dataset I got following responses :

1. **New Weights are** = **(-0.060007 0.121393 -0.055207 0.133993)**

2. First we'll take 70% data and analyze it :

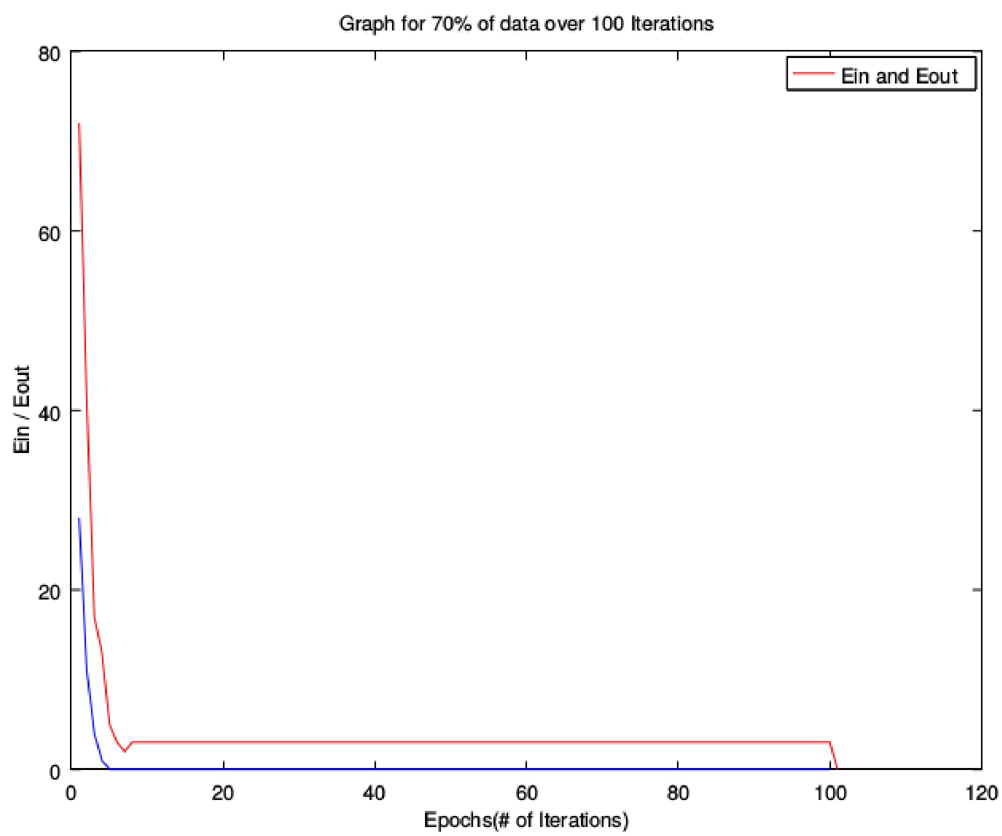

Figure 6: **Pocket algorithm for 70% of Data**

3. Second we'll take 60% data and analyze it :

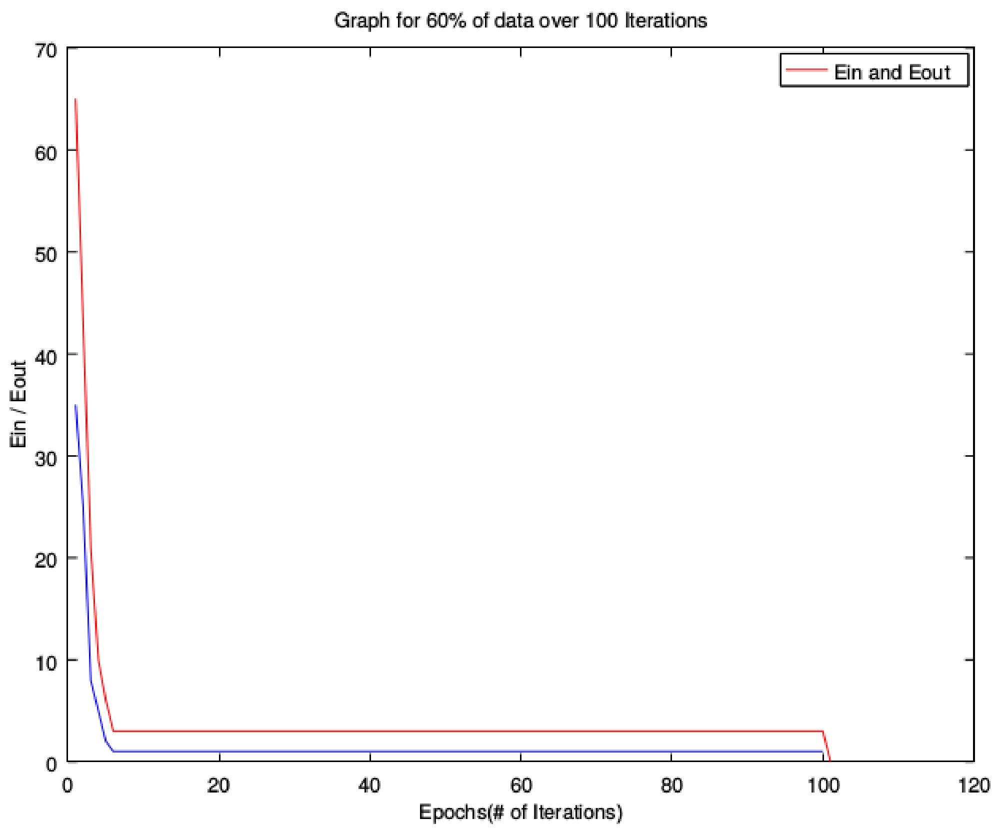

Figure 7: **Pocket algorithm for 60% of Data**

4. Third we'll take 50% data and analyze it :

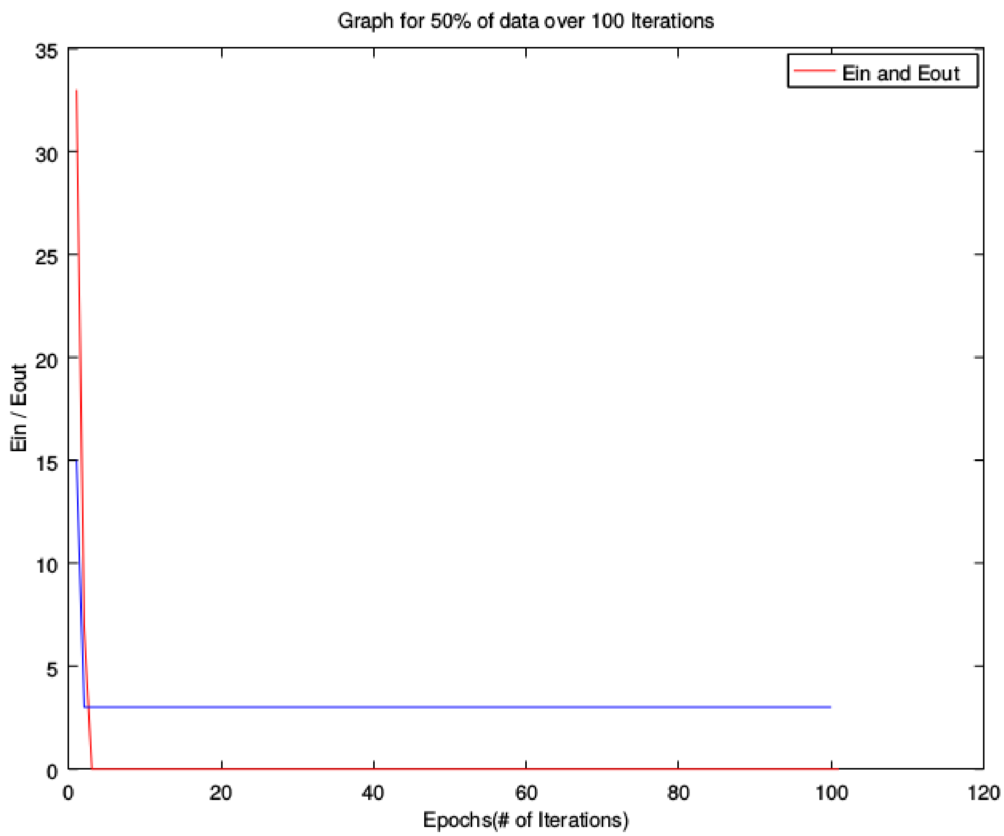

Figure 8: **Pocket algorithm for 50% of Data**

5. Fourth we'll take 40% data and analyze it :

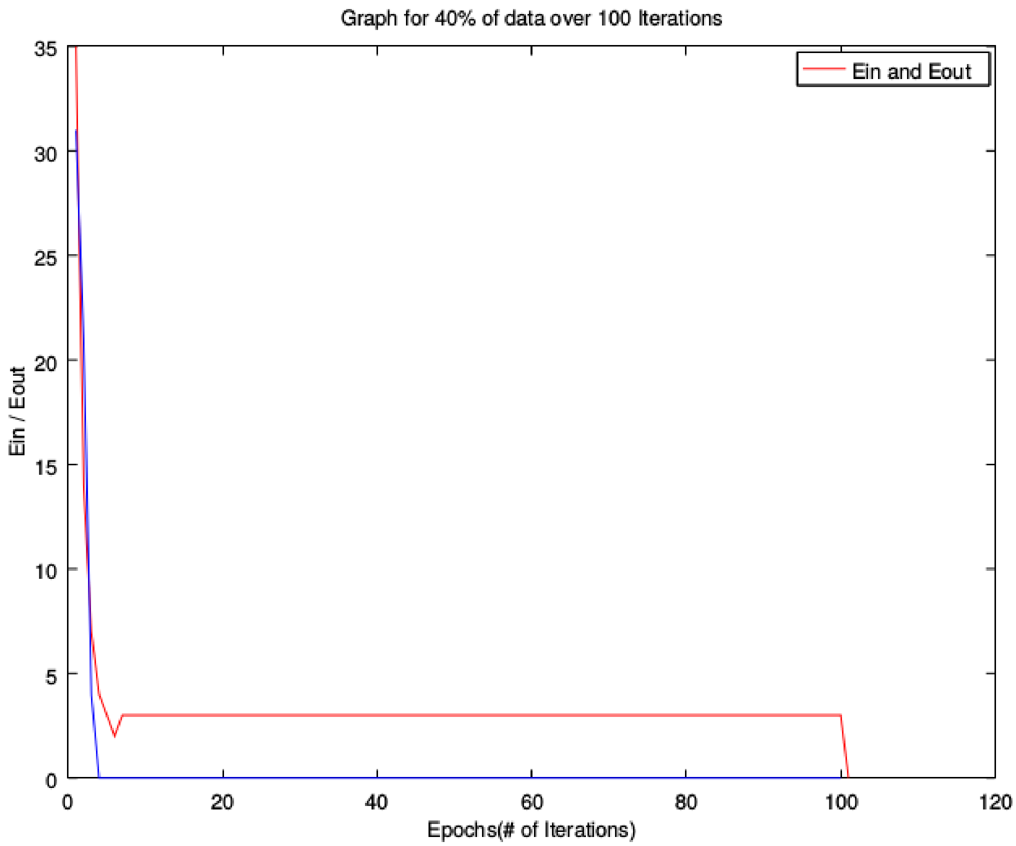

Figure 9: **Pocket algorithm for 40% of Data**

6. Fourth we'll take 30% data and analyze it :

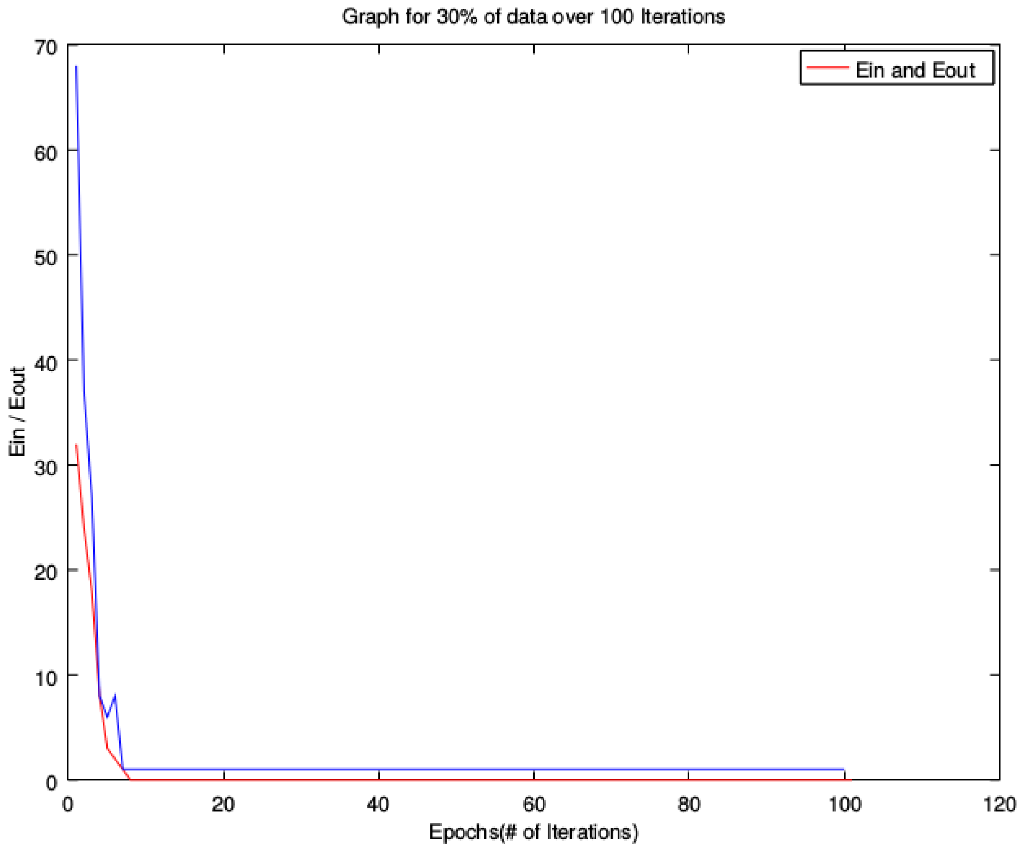

Figure 10: **Pocket algorithm for 30% of Data**

Problem No-4 : Find out the value of E$_{in}$, E$_{out}$ and W for the Non-linear regression ?

Solution : Nonlinear regression is a form of regression analysis in which observational data are modeled by a function which is a nonlinear combination of the model parameters and depends on one or more independent variables. The data are fitted by a method of successive approximations.

After implementing non-linear regression over the IRIS dataset we'll get following results.

- **Error :** = 0
- **New Weights :**

w =
-4.5563166061275213
2.0877495840960867
1.1234298080381084
-1.9278594462558021
0.1163594500289026
-0.1872888566156383
-0.0252081679200517
0.1389208710648461
0.4974216229192964
-0.1413506060501557
0.1489461917851737
-0.2967588527309462
0.1164085530794542
0.0937557436694536
-0.3993001506654886