Assignment 3

Mushfika Rahman

March 29, 2022

Written Problems

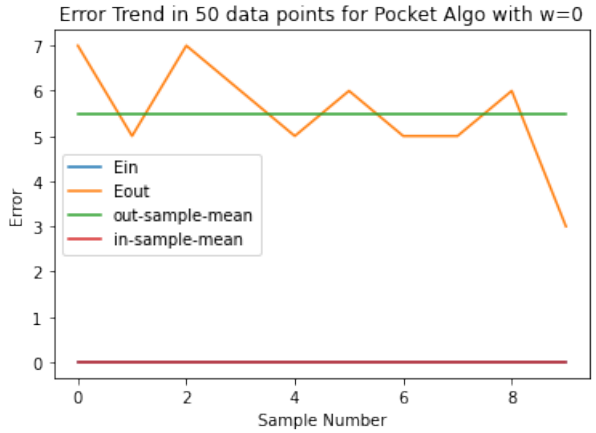
1. Compare two algorithms on a classification task: the Pocket algorithm and linear regression

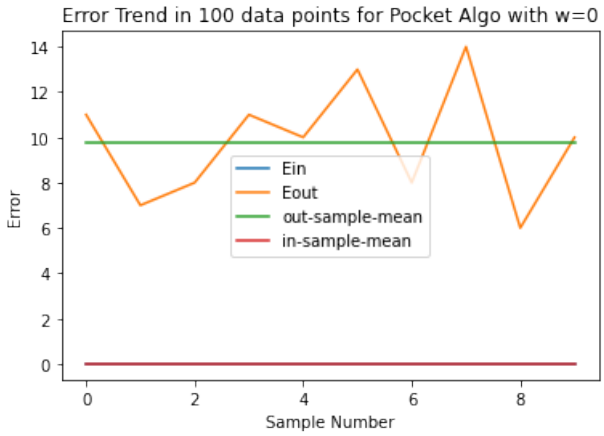
My approach was to observe the error trend in terms of increasing datapoints. I created linearly separable datasets. One dataset contained 50 examples, another contained 100 examples and lastly the dataset contained 1000 datapoints. Each of the datasets was randomly generated from normal distribution, for each number of example the process from generating examples to calculating error was run 10 times. Finally i took the average for observing the E_{out}. For instance, while working with 50 examples, i split the dataset into 80-20 ratio, 80% for training and 20% for the testing purpose.This spliting strategy was applied to all other dataset also. I wanted to take the trainning and testing sample from the same distribution that is why applied this strategy. When the trainning process was done, calculated the number of miss classified examples for the testing that is considered as E_{out}. After finishing the training process, i calculated the number of misclassified points on the training set and i measured that as E_{in}. The whole process ran for 10 times and finally took the average to observe the E_{out} and E_{in}across different sizes datapoints. I took the average because the E_{in} or E_{out} was generating different values every time experiment was conducted, making observation based on experimenting one time would not give better insight.

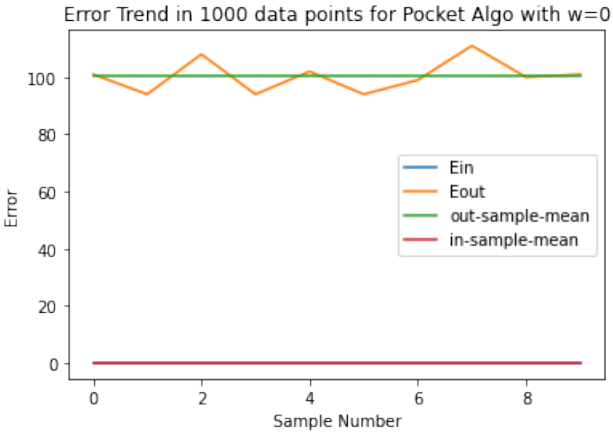
I introduced outlier by randomly taking examples that is $y \in -1$ and making y=1. According to my observation, linear regression is better than other two algorithms because E_out was close to E_in and also it required smaller updates. However, even if the algorithms provided E_in close to zero, but none of them did not provide E_out to be zero.

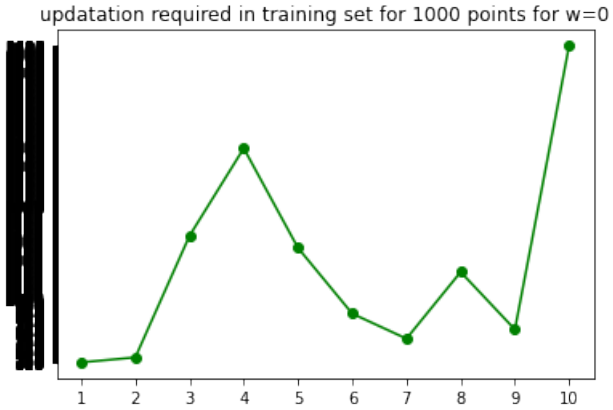
1.1 The Pocket algorithm, starting from w = 0.

→ For the pocket algorithm, for the linearly seperable data the number of misclassified points in training example was 0. I ran the experiment multiple times and sometimes it didn't result in zero. From my analysis as i was creating the dataset based on line position some points were on top of the line specially when i took large number of datapoint. However, the number of misclassified examples for testing test was not always zero. When i ran the algorithm for smaller iteration the number of misclassified points in trainning example was not always zero. With the increasing number of examples the modelś prediction varied a lot. The pocket algorithm required a good number iterations to converge. In case of outlier, the resulting E_in was not zero and E_out was varied from E_in.


Error Trend in 50 data points for Pocket Algo with w=0
Error
Sample Number
Ein
Eout
out-sample-mean
in-sample-mean

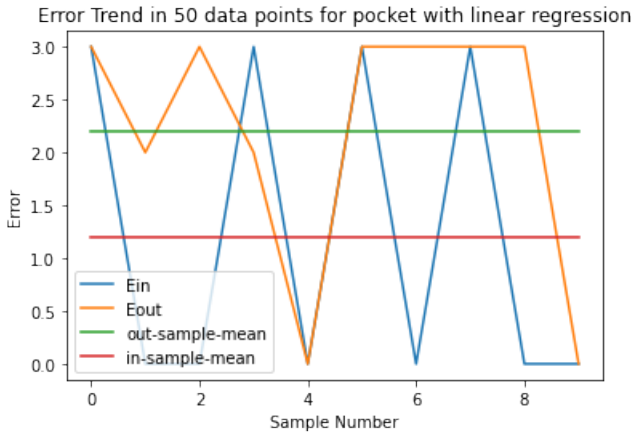

Error Trend in 100 data points for Pocket Algo with w=0
Error
Sample Number
Ein
Eout
out-sample-mean
in-sample-mean

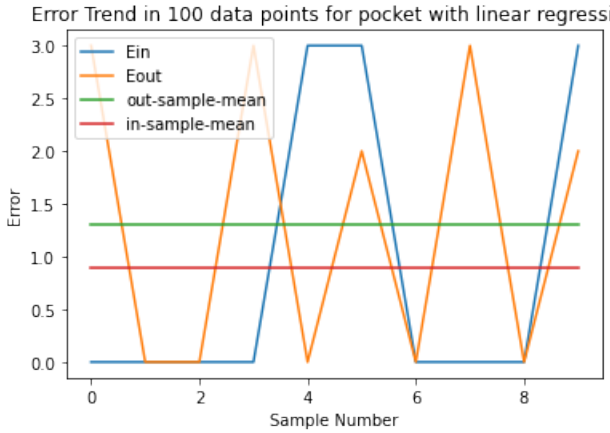

Error Trend in 1000 data points for Pocket Algo with w=0
Error
Sample Number
Ein
Eout
out-sample-mean
in-sample-mean

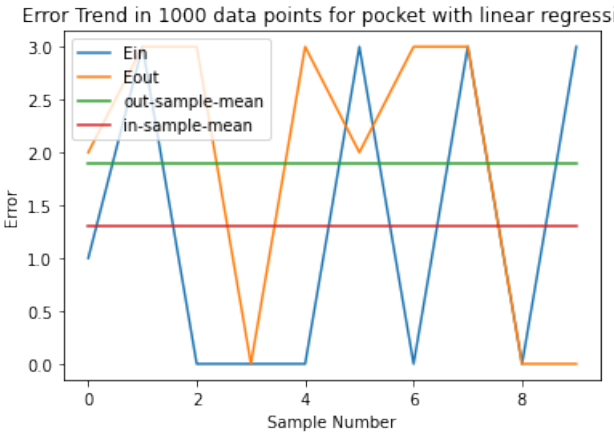


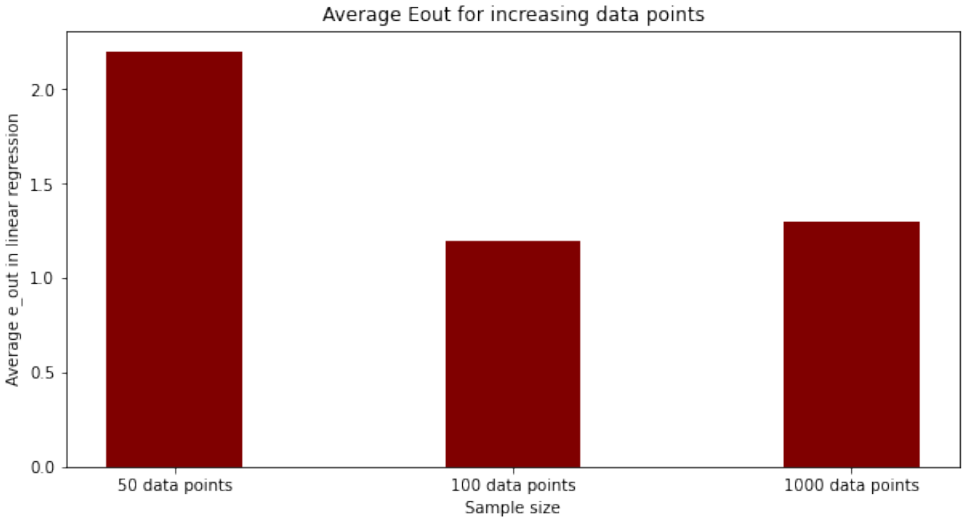
1.2 Linear regression

→ For the linear regression algorithm, implemented the pseudo-inverse algorithm. The number of misclassified example for the trainning set after getting the coeffiecient. The number of misclassified example was counted and measured that as E_{in}. E_{out} was measured in the same way on the testing set, again the testing set was unseen on the data. E_in was sometimes 0, number of misclassified points were 0 and sometimes was not zero. However, the E_out was almost always zero even if it was that was not very far away from the E_in. The variability for the model prediction in testing set was not sparse from the training set. Another interesting observation was with smaller data points the average E_out was a little bit bigger. The model was converged fast for the linearly seperable data as it was calculated in constant step. The gap between average E_in and E_out increased a bit more when dataset was introduced to outliers.



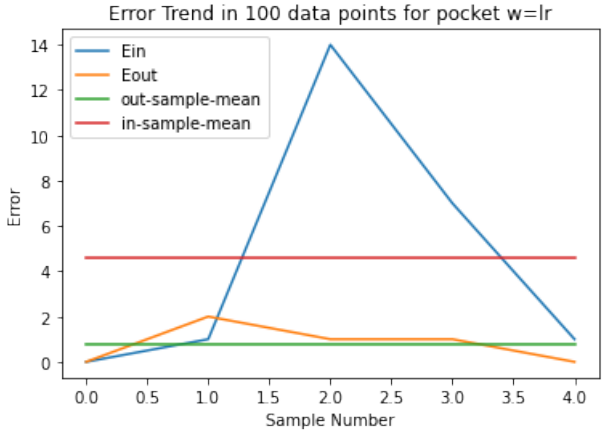

Error Trend in 100 data points for pocket with linear regression
Ein
Eout
out-sample-mean
in-sample-mean
Error
Sample Number

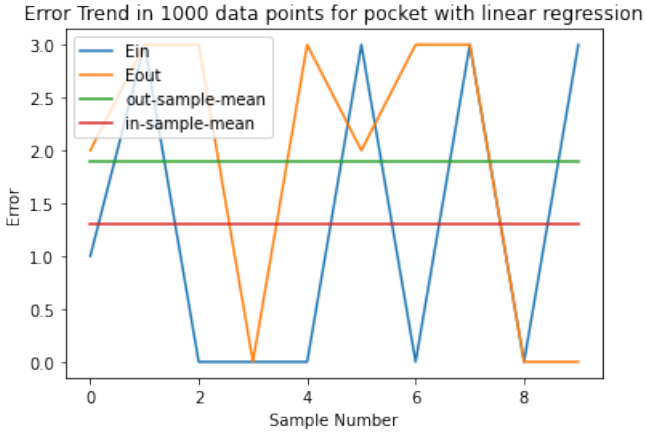

Error Trend in 1000 data points for pocket with linear regression
Ein
Eout
out-sample-mean
in-sample-mean
Error
Sample Number

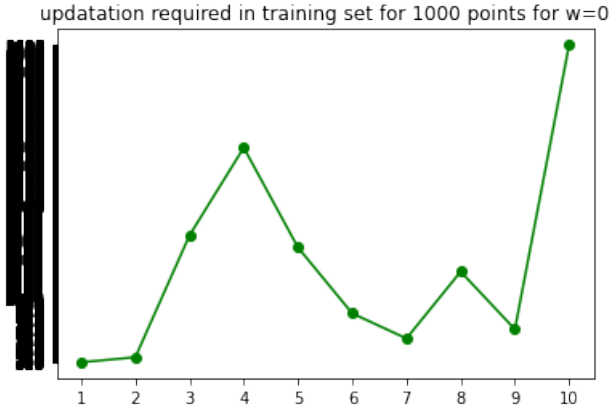

Average Eout for increasing data points
Average e_out in linear regression
50 data points
100 data points
1000 data points
Sample size

1.3 The Pocket algorithm, starting from the solution given by linear regression.

→ For the next experiment instead of w randomly initialized to zero, w was assigned the weights generated by the linear regression. In this case, even though the in sample error was not 0 but in case of out sample error mostly was very close to E_in. That can be concluded as the model generalized well. In this case, the model was converged quickly required less update than pocket algorithm with $w = 0$ initilization.







2. Consider the logistic regression model and its likelihood function:

$$\sigma(\alpha) = \frac{1}{1+\exp(-\alpha)}$$

$$l(w) = \log \prod_{n=1}^{n} \mathcal{P}(y_n|x_n) = \sum_{n=1}^{n} \log \mathcal{P}(y_n|x_n)$$

2.1 show that,

$$\frac{d\sigma}{d\alpha} = \sigma(\alpha)(1-\sigma(\alpha))$$

Given,

$$\sigma(\alpha) = \frac{1}{1+\exp(-\alpha)} = (1+\exp(-\alpha))^{-1}$$

Using reciprocal rule,

$$\begin{aligned}
\frac{d\sigma}{d\alpha} &= -(1+\exp(-\alpha))^{-2}\frac{d\sigma}{d\alpha}(1+\exp(-\alpha)) \\
&= (1+\exp(-\alpha))^{-2}\exp(-\alpha) \\
&= \frac{\exp(-\alpha)}{(1+\exp(-\alpha))(1+\exp(-\alpha))} \\
&= \frac{\exp(-\alpha)}{(1+\exp(-\alpha))}\frac{1}{(1+\exp(-\alpha))} \\
&= \frac{1}{(1+\exp(-\alpha))}\frac{1+\exp(-\alpha)}{(1+\exp(-\alpha))} - \frac{1}{(1+\exp(-\alpha))} \\
&= \sigma(\alpha)(1-\sigma(\alpha))
\end{aligned}$$

2.2 Derive the gradient of the log-likelihood, ∇_wl(w).

$$\nabla_w l(w) = \frac{\partial l(w)}{\partial w} = \frac{\partial \sum_{n=1}^{N}(\log \sigma(y_n w^T x_n))}{\partial w}$$

$$= \sum_{n=1}^{N} \frac{\partial(\log \sigma(y_n w^T x_n))}{\partial w}$$

By applying chain rule,

$$= \sum_{n=1}^{N} \frac{1}{\sigma(y_n w^T x_n)} * \frac{\partial(\sigma(y_n w^T x_n))}{\partial w}$$

$$= \sum_{n=1}^{N} \frac{1}{\sigma(y_n w^T x_n)} * (\sigma(y_n w^T x_n)) * (1-\sigma(y_n w^T x_n)) * y_n x_n$$

$$= \sum_{n=1}^{N} (1-\sigma(y_n w^T x_n)) * y_n x_n$$

$$= \sum_{n=1}^{N} \sigma(-y_n w^T x_n) * y_n x_n$$

$$= \sum_{n=1}^{N} \frac{y_n x_n}{1 + \exp(y_n w^t x_n)}$$

2.3 Write down the update step for gradient ascent of l(w) using the gradient you just derived.

Fixed learning rate for gradient descent:

→ Initialize the weights at time step $t = 0$ to $w(0)$.

→ for t=0,1,2,... do

→ Compute the gradient $g_t = \nabla_w l(w)$.

→ Compute the gradient $v_t = g_t$.

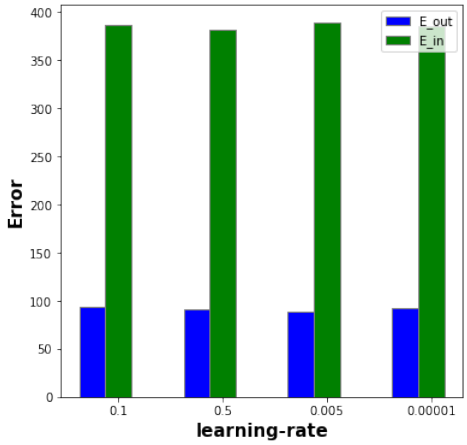
→ Update the weight w(t+1)=w(t)+ηv_t.

→ Iterate to the next step until it is time to stop.

→ Return the final weights

2.4(a) Implement gradient ascent to learn a logistic regression model using the derivations from 2.2. Vary the learning rate.

I varied the learning rate for 1000 datapoints and observed the changes E_{in} and E_{out}. I initialized the $w = 0$. For the stopping criteria, the algorithm will stop after either a certain number of iterations or the in-sample-error didnot get any better for a certain range. The E_{out} was smaller than E_{in} in all cases. The gradient ascent tries to maximize the likelyhood trying find the peek therefore finding the global solution. In the case of linear separability the algorithm required smaller steps when it had smaller datapoints. However, in terms of varying learning rate i was unable to conclude which is best because the E_out was smaller than E_in all cases. One observation was that with larger learning rate, the algorithm requires less steps to converge.



I compared the logistic regression (gradient ascent) with linear regression and pocket algorithm(starting point of w is from the linear regression). The comparison was done on linearly separable data. The pocket algorithm had larger E_{out}, the reason behind it might be i ran the pocket for smaller number of iterations.