IE613:Online Learning
Assignment-2

Vinay Chourasiya (173190011)

April 2, 2018

Question 1

- **Proof of FoReL and WMA equivalence**
- The update rule for FoRel is

$$W_t = argmin_W \sum_{s=1}^{t-1} c_s(W) + R(W)$$

$$R(W) = W^T \log(W), \qquad c_s(W) = \langle W, V_t \rangle$$

where V_t is loss vector generated in round t.

- now for optimal W_t, :

$$min\{\sum_{s=1}^{t-1}\sum_{j=1}^{d} W_j v_{j,s} + \frac{1}{\eta}\sum_{j=1}^{d} W_j \log W_j\}$$

s.t.

$$\sum_{j=1}^{d} W_j = 1$$

- Using KKT-condition :

$$f := \sum_{s=1}^{t-1}\sum_{j=1}^{d} W_j v_{j,s} + \frac{1}{\eta}\sum_{j=1}^{d} W_j \log W_j + \lambda(\sum_{j=1}^{d} W_j - 1) = 0 \tag{1}$$

$$\sum_{j=1}^{d} W_j = 1 \tag{2}$$

Now differentiate equation (1) with respect to W_j :

$$\frac{df}{dW_j} = 0$$

$$\sum_{s=1}^{t-1} v_{j,s} + \frac{1}{\eta}(1 + \log(W_j)) + \lambda = 0 \qquad \forall j$$

$$\log(W_j) = \eta(-\lambda - \sum_{s=1}^{t-1} v_{j,s}) - 1$$

$$W_j = \exp\left(\eta(-\lambda - \sum_{s=1}^{t-1} v_{j,s}) - 1\right) \tag{3}$$

Now we will proceed with differentiating equation (1) with respect to λ :

$$\frac{d}{d\lambda} = 0$$

From equation (2), we know

$$\sum_{j}^{d} W_j = 1$$

substitute value of W_j from equation (3), we get

$$\sum_{j=1}^{d} \exp\left(\eta(-\lambda - \sum_{s=1}^{t-1} v_{j,s}) - 1 \right) = 1$$

$$\left(\sum_{j-1}^{d} \exp(-\eta \sum_{s=1}^{t-1} v_{j,s}) \right) \exp(-\eta\lambda) = e$$

$$\exp(-\eta\lambda) = \frac{e}{\sum_{j}^{d} \exp(-\eta \sum_{s=1}^{t-1} v_{j,s})}$$

$$\lambda = \frac{-1}{\eta}\left[1 - \log \sum_{j=1}^{d} \exp(-\eta \sum_{s=1}^{t-1} v_{j,s}) \right]$$

now substitute value of λ in equation (3), we get

$$W_j = \exp(-\log \sum_{j=1}^{d} \exp(v_{j,s})) \exp(-\eta \sum_{s=1}^{t-1} v_{j,s})$$

further simplifying above equation will give

$$W_j = \frac{\exp(-\eta v_{j,s})}{\sum_{j=1}^{d} \exp(v_{j,s})}$$

The obtained weight update is equivalent to the weight update of Weighted Majority algorithm.
Hence,

$$w_j^f = w_j^{wm}$$

Hence η^* will also be the same as that of WM i.e.

$$\eta^* = \sqrt{\frac{2\log(d)}{T}}$$

Question 2

- **Follow-The-Leader (FTL)**
- Program file for both FTL and FoReL – *Q2.py*, The graph plotted on Logarithmic scale

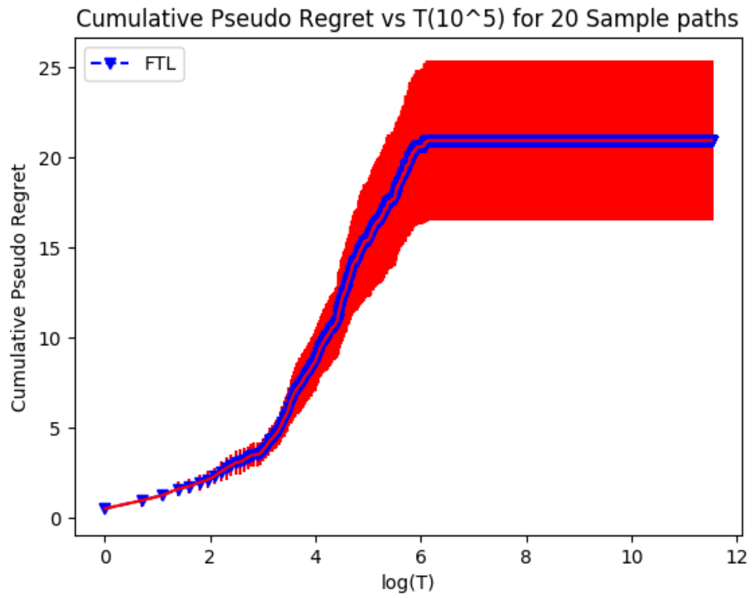

Figure 1: For Follow The Leader:Cumulative Pseudo Regret vs T

Follow-the-Regularized-Leader (FoReL)

$$\eta^* = \sqrt{\frac{2\log(d)}{T}}$$

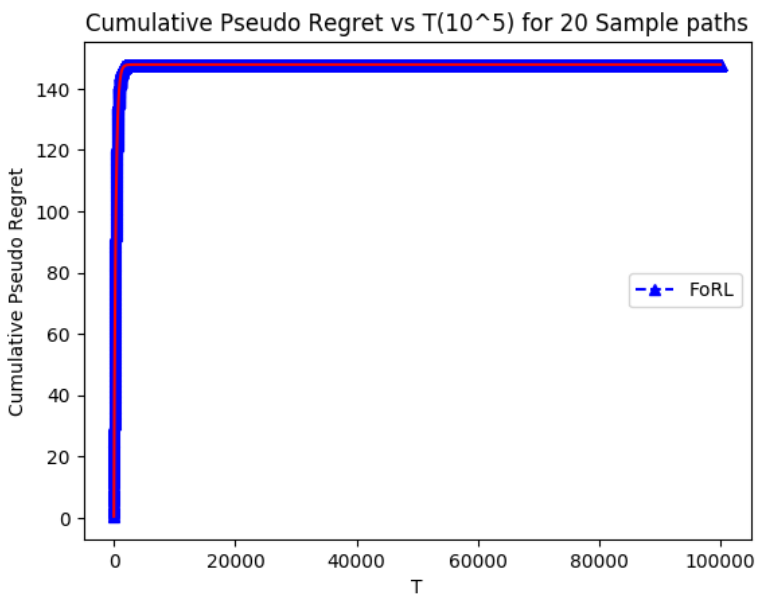

Figure 2: Cumulative Pseudo Regret vs T

Question 3

1 program file – *Q3.py*

2 learning rate $(\eta) = c\eta^{\star}$ and $c = [.1, 2.2]$.

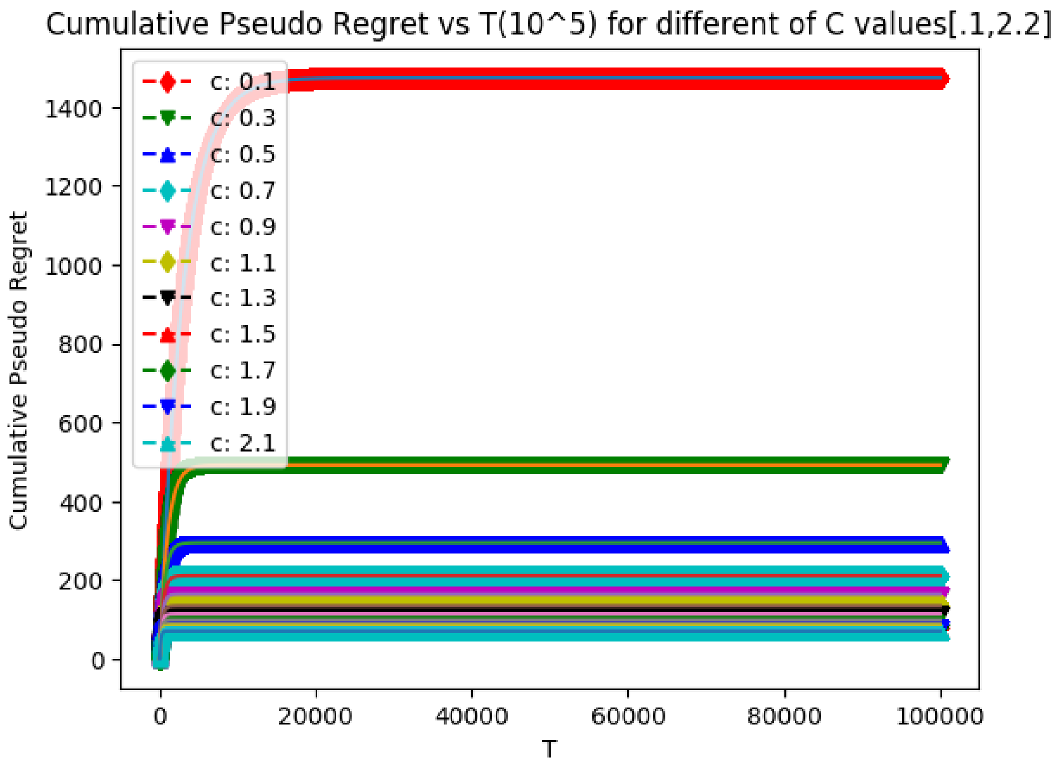

Figure 3: Cumulative Pseudo Regret vs T with different c values

Question 4

- Graph for Skin Data
- program file – *Q41.py*
- We run both algorithm on given Data as on course website
- the graph for skin data has only one spike,because this is sorted data.

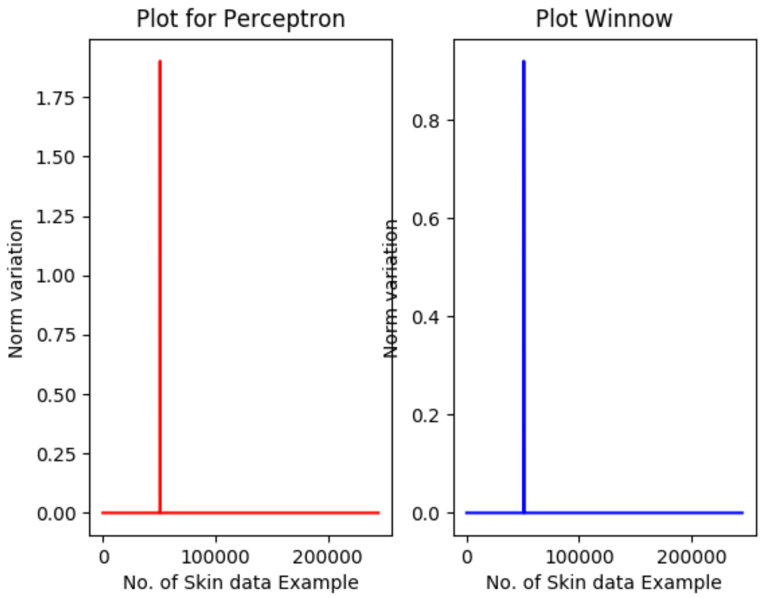

Figure 4: Perceptron and Winnow for Skin Data

- Graph for News Data
- program file – *Q42.py*
- the given Data of News normalized, because original data set this is not working with winnow update(some update lead to infinity value.).

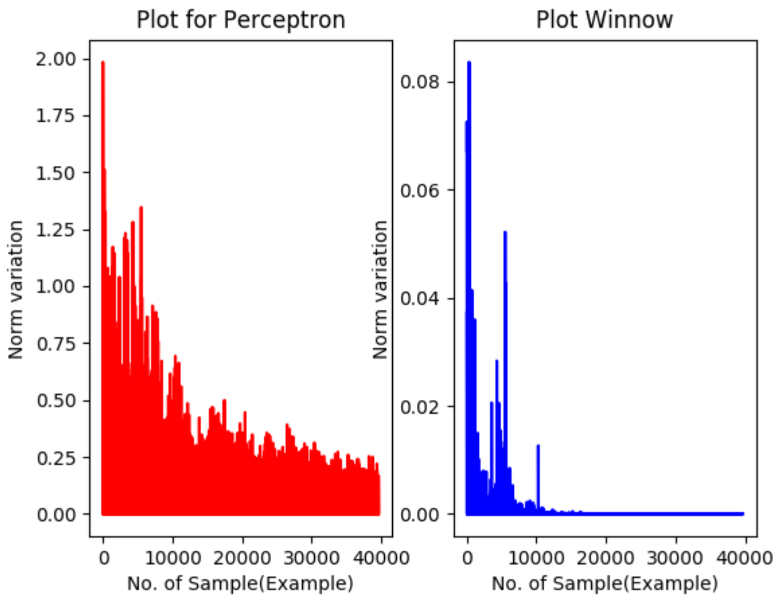

Figure 5: Graph for Perceptron an Winnow

- Graph for Wine Data
- program file – *Q43.py*

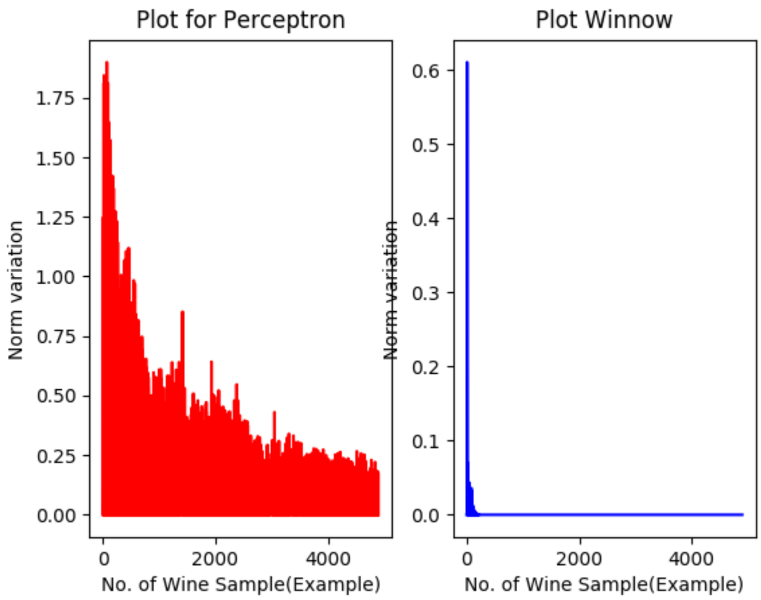

Figure 6: Graph for Perceptron an Winnow

Question 6

- program file – *Q6.py*
- we use 1 extra feature in x_t as bias term to more efficiently learn the data set, this will reduce mistakes significantly .
- estimated margin(γ) = 3.2
- for perceptron algorithm mistake count = 23
- mistake bound for Perceptron = 176.23
  Mistake Bound

$$\frac{R^2\|W^\star\|^2}{\gamma^2} \qquad R = sup\{\|x_t\|_2\}$$

- Mistake count for Winnow Algorithm with $\eta \leq .5$

γ	mistakes
.05	1277
.1	1269
.15	1246
.2	1247
.25	1207
.3	1196
.35	1172
.4	1150
.45	1145

Question 7

- program file – *Q7.py*

Given Cost function $c_t(w) = max\{0, 1 - \eta * x_t * y_t\}$
for Online gradient descent(OGD) update rule is $w_{t+1} \leftarrow proj_k(w_t - \eta \nabla c_t)$
and $\nabla c_t = -y_t * x_t$ (when there is mistake, otherwise zero) update rule $w_t + 1 \leftarrow w_t - (-y_t * x_t)$
thus we can conclude that given cost function *OGD* work same as *Perceptron*

For Online Mirror Descent(OMD) regularizer $R(w) = (w^T log(w))$ and with same $c_t(w)$. update rule for OMD is

$$\nabla R(w_{t+1}) \leftarrow \nabla R(w_t) - \eta \nabla c_t$$

$\nabla R(w) = 1 + log(w)$,after substituting the $\nabla R(W_{t+1}), R(w_t) and \nabla c_t$ we get:

$$w_{t+1} = w_t exp\{\eta y_t x_t\}$$

- **online gradient descent and online mirror descent algorithm**

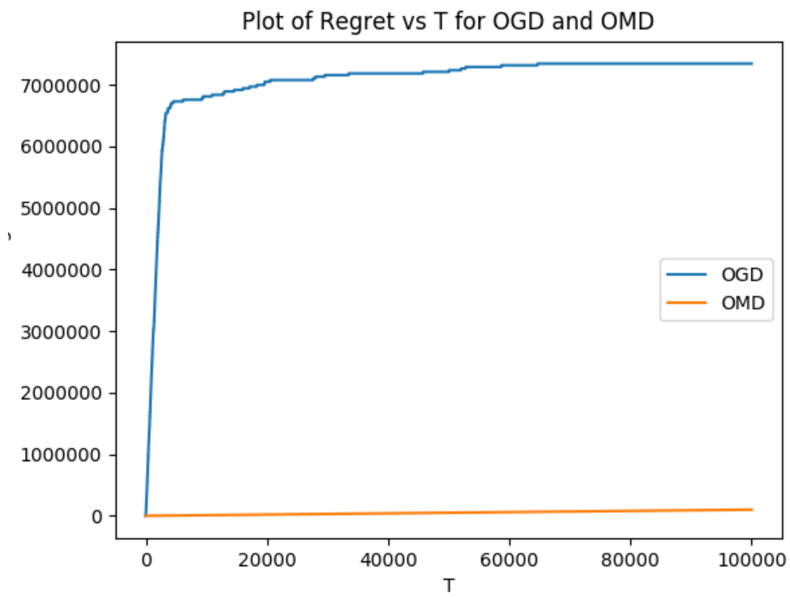

Figure 7: regret vs round t for both algorithms

- from the Figure 7 this is clear that OGD Regret is more than the OMD
- Due to multiplicative update of OMD run with lower regret and also learn faster than OGD.