CS680 – Assignment 2

Ali ElSaid (20745892)

Q1.1:

Q1. 1

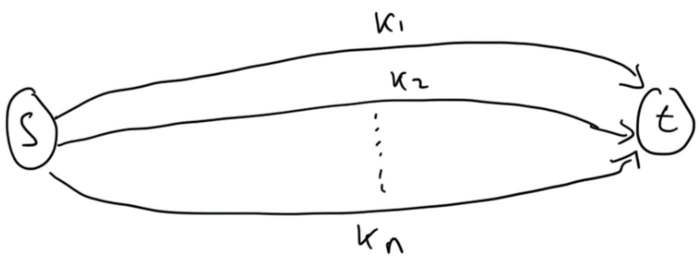
- Since product of Kernels is a Kernel then, K_P is a Kernel For any Path P.
- & since the Sum of two kernels is a Kernel So
  $\sum K_P$ is also a Kernel

  $\therefore K_G$ is a Kernel

Q1.2:

Q1.2

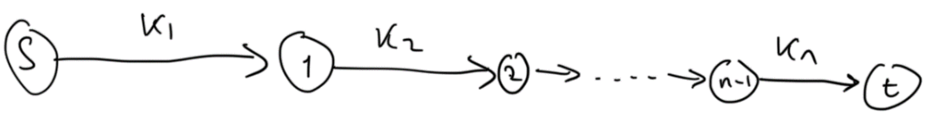
(1)



$\therefore K_G = \sum_{P \in Path(S \to t)} K_P$ & every Path is one edge

So $\sum_{i=1}^{n} k_i = K_G$

(2)



Only one Path from $S \to t$ $\therefore K_G = \prod_{i=1}^{n} k_i$

Q1.3:

Q1.3

(1) Since only one path between S & C

$$K_{G_1} = xz \cdot (xz-1)^3$$

(2) 2 paths between S & d, So

$$K_{G_2} = xz \cdot (xz-1)^3 \cdot e^{-(x-z)^2/2} + xz \cdot e^{-|x-z|} \cdot \tanh(xz+1)$$

Q1.4:

Q1.4

Assuming there is one source (s) & one sink (t), & there's a path between them.
atleast one

- then we can just start backwards from t, & store the result of each node & multiply it with the nodes that have edges going into it. This way we compute every nodes result once & pass by every edge once. (This is a form of dynamic programing, since we store the results of the nodes & reuse them later)
- we can do this because the graph is a DAG & we can go backwards in the reverse topological sort ordering

since for a node u, all edges coming into u will multiply with its "result", i.e. the result of the back propagation.

- Algo:

we can use dfs and memo-ize the result of every node u after computing it & then reusing it when it is later called potentially from another path. With the base case being result(t) = 1 the sink node. (This way we traverse each node & edge once, i.e. $O(|V|+|E|)$)

there's also the bottom up approach which starts from t & goes backwards, but the top down one with memo-ization is more intuitive.

Q1.5:

Q1.5:

if K is a kernel it is also positive semi definite

function. $\therefore$ $K(s+\delta s, t+\delta t) > K(s, t+\delta t)$

& $K(s+\delta s, t+\delta t) > K(s+\delta s, t)$

$\therefore$ the numerator is Positive Semi Definite too

- & Since addition of Kernels is a Kernel,

  the numerator is a Kernel

- & since cK for a constant c is a kernel if K is a Kernel

  then $\underbrace{\left(\frac{1}{\delta s \delta t}\right)}_{c} \cdot$ (numerator) is also a Kernel

- finally the limit of a Kernel is also a Kernel

$\therefore$ $\lim\limits_{\delta t \to 0, \delta s \to 0} \left(\frac{1}{\delta s \delta t}\right) \cdot \left(K(s+\delta s, t+\delta t) - K(s+\delta s, t) - K(s, t+\delta t) + K(s,t)\right)$

is a Kernel as a whole

& $\therefore$ the mixed derivative is also a Kernel

Q2.1:

Q2.1

- $a = \cos(s)$, • $b = \exp(a) = \exp(\cos(s))$
- $c = \dfrac{1}{a^2+1} = \dfrac{1}{\cos(s)^2+1}$, • $d = \sin(b) \cdot \ln c$

$$= \sin(\exp(\cos(s))) \cdot \ln\left(\frac{1}{\cos(s)^2+1}\right)$$

- $t = d^2 = \left(\sin(\exp(\cos(s))) \cdot \ln\left(\dfrac{1}{\cos(s)^2+1}\right)\right)^2$

Q2.2

Q2.2:

Using Productrule & Chain rule we get

$$\frac{\partial t}{\partial s} = 2(d) \cdot \underbrace{\frac{\partial d}{\partial s}} = 2d \cdot \left(\sin b \cdot \frac{\partial \ln c}{\partial s} + \frac{\partial \sin b}{\partial s} \cdot \ln c \right)$$

$$= 2d \left(\sin b \left(\frac{1}{c}\right) \cdot \underbrace{\frac{\partial c}{\partial s}} \right) + \cos b \cdot \underbrace{\frac{\partial b}{\partial s}} \cdot \ln c)$$

- $\frac{\partial c}{\partial s} = \frac{\partial \frac{1}{a^2+1}}{\partial a} \cdot \frac{\partial a}{\partial c} = -(a^2+1) \cdot 2a \cdot \underbrace{\frac{\partial a}{\partial c}}$

- $\frac{\partial b}{\partial s} = \frac{\partial \exp(a)}{\partial a} \cdot \frac{\partial a}{\partial c} = e^a \cdot \underbrace{\frac{\partial a}{\partial c}}$

- $\frac{\partial a}{\partial c} = -\sin(s)$

Plugging everything back, we get

- $\frac{\partial b}{\partial s} = \exp(\cos(s)) \cdot -\sin(s)$

- $\frac{\partial c}{\partial s} = -\left(\cos(s)^2+1\right)^{-2} \cdot 2\cos(s) \cdot -\sin(s)$

- $\frac{\partial d}{\partial s} = \sin\left(\exp(\cos(s))\right) \cdot \left(\cos(s)^2+1\right) \cdot (\quad)$
  $+ \cos\left(\exp(\cos(s))\right) \cdot \ln\left(\frac{1}{\cos(s)^2+1}\right) \cdot (\quad)$

So $\rightarrow \frac{\partial t}{\partial s} = \left[2\left(\sin(\exp(\cos(s))) \cdot \ln\left(\frac{1}{\cos(s)^2+1}\right)\right) \right]$

$\cdot \Big[\sin(\exp(\cos(s))) \cdot (\cos(s)^2+1) \cdot \left(\cos(s)^2+1\right)^{-2} \cdot 2\cos(s) \cdot \sin(s)$

$+ \cos(\exp(\cos(s))) \cdot \ln\left(\frac{1}{\cos(s)^2+1}\right) \cdot \left(\exp(\cos(s)) \cdot -\sin(s)\right) \Big]$

Q2.3:

Q 2.3:

Using the results from 2.2

$$\frac{\partial v_0}{\partial s} = 1, \quad \frac{\partial v_1}{\partial s} = -\sin(v_0) \cdot \frac{\partial v_0}{\partial s}$$

$$\frac{\partial v_2}{\partial s} = \exp(v_1) \cdot \frac{\partial v_1}{\partial s}, \quad \frac{\partial v_3}{\partial s} = -2v_1(v_1 + 1)^{-2} \cdot \frac{\partial v_1}{\partial s}$$

$$\frac{\partial v_4}{\partial s} = \sin(v_2) \cdot \frac{1}{v_3} \cdot \frac{\partial v_3}{\partial s} + \cos(v_2) \cdot \frac{\partial v_2}{\partial s} \cdot \ln(v_3)$$

$$\frac{\partial v_5}{\partial s} = \frac{\partial t}{\partial s} = 2 v_4 \frac{\partial v_4}{\partial s}$$

, We can traverse once. Since we go from source to sink and Conpute along the way.

Q2.4:

Q2.4

$$\frac{\partial t}{\partial v_5} = 1, \quad \frac{\partial t}{\partial v_4} = \frac{\partial t}{\partial v_5} \cdot \underbrace{\frac{\partial v_5}{\partial v_4}}_{\to 2v_4 \text{ (from Q1.3)}}$$

$$\frac{\partial t}{\partial v_3} = \frac{\partial t}{\partial v_4} \cdot \underbrace{\frac{\partial v_4}{\partial v_3}}_{\to \sin(v_2) \cdot \frac{1}{v_3}}, \quad \frac{\partial t}{\partial v_2} = \frac{\partial t}{\partial v_4} \cdot \underbrace{\frac{\partial v_4}{\partial v_2}}_{\to \cos(v_2) \cdot \ln(v_3)}$$

$$\frac{\partial t}{\partial v_1} = \frac{\partial t}{\partial v_2} \cdot \underbrace{\frac{\partial v_2}{\partial v_1}}_{-2v_1(v_1^2+1)^{-2}} + \frac{\partial t}{\partial v_3} \cdot \underbrace{\frac{\partial v_3}{\partial v_1}}_{\exp(v_1)}$$

$$\frac{\partial t}{\partial v_0} = \frac{\partial t}{\partial s} = \frac{\partial t}{\partial v_1} \cdot \underbrace{\frac{\partial v_1}{\partial v_0}}_{\to -\sin(v_1)}$$

$\to \frac{\partial t}{\partial s} = \frac{\partial t}{\partial v_1} \cdot -\sin(s)$, we need to traverse two times, one to represent the final result in terms of intermediate ones & one to compute final result from them by substituting the calculations in the first iteration.

Q3.1:

Q3.1.a

Prove: $\epsilon_t = 1 - \gamma_t \longleftrightarrow \underset{d\times 1}{\epsilon_t} + \underset{d\times 1}{\gamma_t} = \underset{d\times 1}{\mathbb{1}}$

$$\epsilon_t + \gamma_t = M_-^T P_t + M_+^T P_t = (M_-^T + M_+^T) P_t$$

$$= |M|^T \cdot P_t = \underset{d\times n}{\mathbb{1}} \cdot \underset{n\times 1}{P_t} = \underset{d\times 1}{\mathbb{1}}$$

Since all the elements of M are either -1 or 1

Q3.1.b

Lets create a mapping from the Adaboost algorithm in class (ADA) to the one presented in the assignment (ASN). For the sake of notation, put ~ above a symbol to indicate that it is from the assignment algorithm.

First: Mapping between classifiers. Let each weak classifier in ADA be one of the d weak classifiers in ASN.
& we map the output as follows: (And we let $T = \text{max\_pass}$)

$$\underbrace{h_t^*(x_i)}_{\text{from ASN}} = 2 \cdot \underbrace{h_t(x_i)}_{\text{from ADA}} - 1$$

, this maps the output:

$0 \longrightarrow -1$

$1 \longrightarrow 1$

(ADA) (ASN)

Second: Show that the weight assigned to each classifier in both algorithms is the same.

• ADA:

weighted prediction per classifier: $\ln\left(\frac{1}{\beta_t}\right)\left(h_t(x_i) - \frac{1}{2}\right)$

transforming the prediction to be similar to ASN we get

$$\rightarrow \frac{1}{2}\ln\left(\frac{1}{\beta_t}\right) \cdot \underbrace{\left(2\,h_t(x_i) - 1\right)}_{h_t^*(x_i)}$$

$\therefore$ the weight assigned to each classifiers prediction in ADA if we transform their prediction to $\tilde{h}_t(x_i)$ is

$$\frac{1}{2}\ln\left(\frac{1}{\beta_t}\right)$$

· **ASN:**

Since $\tilde{w}_0$ is initialized to be zero, & we choose one classifier ~~let us say~~ j_t each iteration. Then the final weight for a specific classifier $\tilde{h}_j$ is $\rightarrow$ $\tilde{w}_{mp,j}$ = sum of $\tilde{\beta}_{t,j}$ for all the times $\tilde{h}_j$ was chosen

(since only one entry of α_t is 1 at a time, ∴ only one entry of w_t is updated at a time)

mp = max-pass

to map to ADA, lets consider each time $\tilde{h}_j$ is chosen as a "different" classifier with a different weight, so each "version" at iteration t of $\tilde{h}_j$ which we call $\tilde{h}_{t,j}$ has weight

$$\tilde{\beta}_{t,j} = \tfrac{1}{2}\left(\ln \tilde{\gamma}_{t,j} - \ln \tilde{\epsilon}_{t,j}\right)$$
$$= \tfrac{1}{2}\left(\ln(1-\tilde{\epsilon}_{t,j}) - \ln(\tilde{\epsilon_t})\right) \quad \text{since } \epsilon_t + \gamma_t = 1$$
$$= \tfrac{1}{2}\ln\left(\left(\frac{\tilde{\epsilon}_t}{1-\tilde{\epsilon}_t}\right)^{-1}\right)$$

- where $\tilde{\epsilon}_{t,j}$ is the expected (weighted) loss for classifier j at iteration t & likewise for $\gamma_{t,j}$

- Note that $\tilde{\epsilon}_{t,j} = \epsilon_t$, i.e. the expected loss for the classifier at time t is the same in both ADA and ASN

- if $\tilde{\epsilon}_{t,j} = \epsilon_t$, then $\frac{\tilde{\epsilon}_{t,j}}{1-\tilde{\epsilon}_{t,j}} = \frac{\epsilon_t}{1-\epsilon_t} = \beta_t$, making ASN's weights $\tilde{\beta}_{t,j}$ be also $\rightarrow \frac{1}{2}\ln\left(\frac{1}{\beta_t}\right)$

  So we have, weights of ADA = ASN for each classifier

- We prove this in the third part by proving that the updates to the weights of observations in both algorithms are essentially the same. ($\tilde{p}_t$ & p_t).

Third: updates

ADA: $w_{t+1} \leftarrow w_t \odot \beta_t^{\ell_t}$

if we multiply by a constant $\beta_t^{-1/2}$ for all w_t, we change nothing because of the normalization in the next iteration. $\left(p_t = \frac{w_t}{1^T w_t}\right)$

But the update becomes:
$$\begin{cases} \beta_t^1 \cdot \beta_t^{-1/2} = \beta_t^{1/2}, & \text{if } \ell_t = 1 \text{ (i.e. correct pred.)} \\ \beta_t^0 \cdot \beta_t^{-1/2} = \beta_t^{-1/2}, & \text{if } \ell_t = 0 \text{ (i.e. incorrect pred.)} \end{cases}$$

where $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

$1-\epsilon_t$

ASN:

$$\tilde{P}_{t+1} \leftarrow \tilde{P}_t \odot \exp\left(-M(\tilde{\alpha}_t \odot \tilde{\beta}_t)\right)$$

we focus only on the j^{th} classifier (the one chosen at time t, since $\tilde{\alpha}_t$ is zero at all entries corresponding to other classifiers & so no update happens)

$$\text{So, } \tilde{\alpha}_t \odot \tilde{\beta}_t = \left[\ldots 0 \ldots, \tilde{\beta}_{t,j}, \ldots 0 \ldots\right]$$

$$\& -M(\tilde{\alpha}_t \odot \tilde{\beta}_t) = -\underbrace{\begin{bmatrix} 0+0+\ldots+y_1 \tilde{h}_j(x_1)\cdot\tilde{\beta}_{t,j}+\ldots+0+0 \\ \vdots \\ 0+0+\ldots+y_n \tilde{h}_j(x_n)\tilde{\beta}_{t,j}+\ldots+0+0 \end{bmatrix}}$$

, i.e $-M(\tilde{\alpha}_t \odot \tilde{\beta}_t)$ will only have entries corresponding to

inspecting one element in the update corresponding to point(i)

$$P_{t+1,i} = P_{t,i} \odot \underbrace{e^{-y_i \tilde{h}_j(x_i)\tilde{\beta}_{t,j}}}$$

$$= e^{-\frac{1}{2}\ln\left(\frac{\tilde{\epsilon}_t}{1-\tilde{\epsilon}_t}\right)\cdot y_i h_j(x_i)} = \left(\frac{\tilde{\epsilon}_t}{1-\tilde{\epsilon}_t}\right)^{\frac{1}{2}(y_i h_j(x_i))}$$

$$\text{Which is } \begin{cases} \left(\frac{\tilde{\epsilon}_t}{1-\tilde{\epsilon}_t}\right)^{\frac{1}{2}} & \text{, if correct prediction} \\ \left(\frac{\tilde{\epsilon}_t}{1-\tilde{\epsilon}_t}\right)^{-\frac{1}{2}} & \text{, if incorrect prediction} \end{cases}$$

Inspecting both algorithms updates, we see that they are exactly the same due to the normalization & since both $\tilde{\epsilon}_t$ & ϵ_t are the respective weighted losses.

$$\therefore\ \tilde{\epsilon}_t = \epsilon_t \quad \& \quad \tilde{\beta}_{t,j} = \frac{1}{2}\ln\frac{1}{\beta_t}$$

Finally: That proves that both algorithms are equivalent.

Q3.2:

Q3.2

Algorithm: Since b_j is a threshold for x_j, we only need to try the $n+1$ points between the values of x_j. At each of those points we also flip (the sign of) s_j & b_j to try flipping the current labeling & calculate the new loss as $1-$ the current loss, since we flipped all the predictions and $\sum_{i=1}^{n} p_i = 1$.

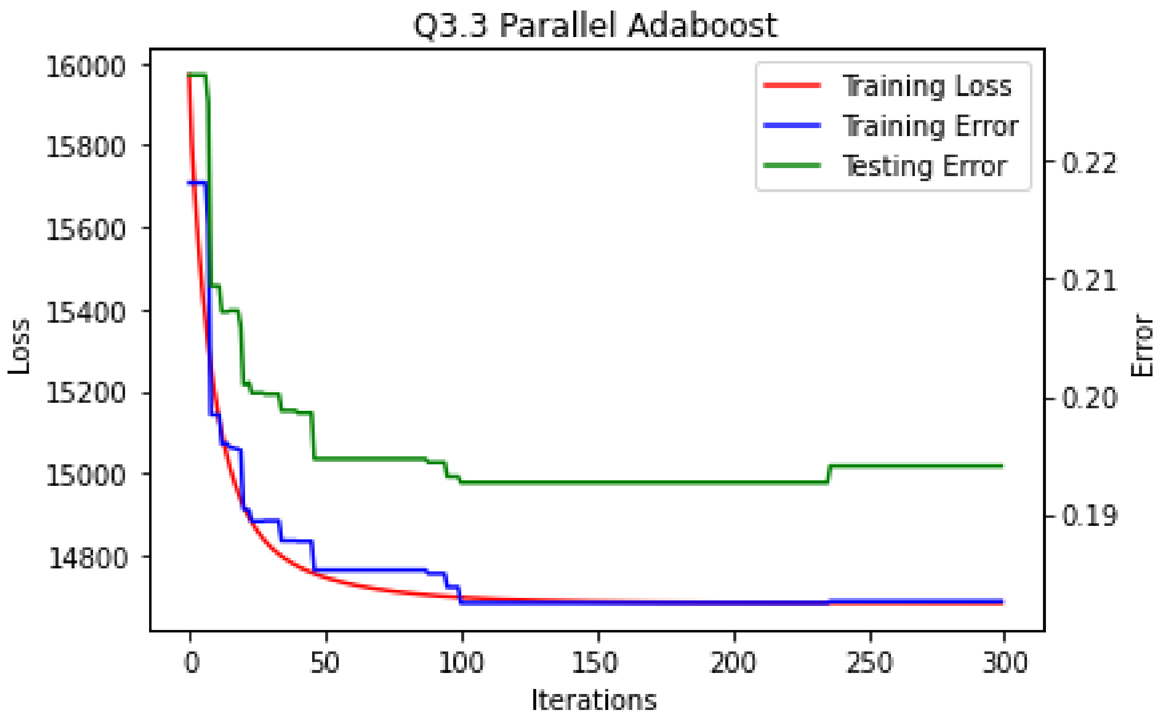
To efficiently compute the loss we do it as we go from the bottom most threshold to the top most, adding or subtracting the new p_i from the loss if the new point is missclassified or classified correctly respectively according to moving the threshold up once.

Finally, we report s_j & b_j corresponding to the minimum loss.

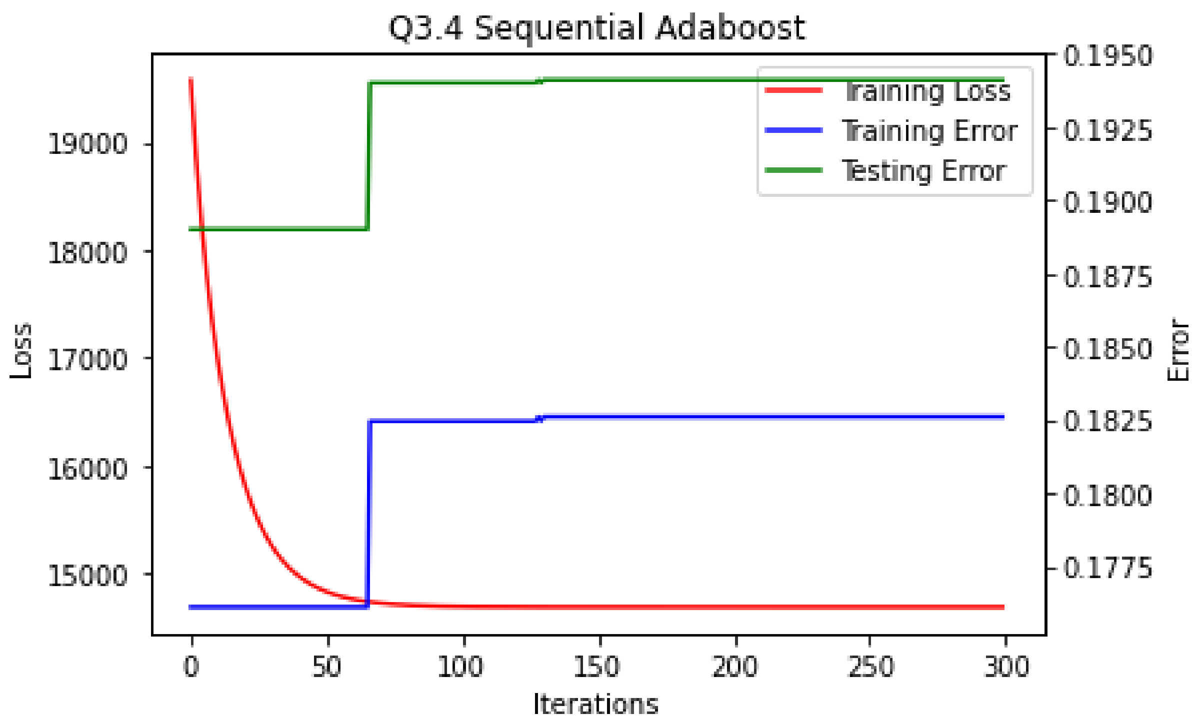
(we also take into consideration that the points can have duplicates & aren't sorted)

```
Best Feature: [6]
Test Error: [0.189]
Training Error: [0.1761]
sj: [1.]
bj: [-1.5]
Worst Feature: [14]
Test Error: [0.2276]
Training Error: [0.218]
sj: [-1.]
bj: [-19844.75]
```

Q3.3:


Q3.3 Parallel Adaboost
Training Loss
Training Error
Testing Error
16000
15800
15600
15400
15200
15000
14800
Loss
0.22
0.21
0.20
0.19
Error
0
50
100
150
200
250
300
Iterations

Q3.4:



Q3.4 Sequential Adaboost
Training Loss
Training Error
Testing Error
19000
18000
17000
16000
15000
Loss
0.1950
0.1925
0.1900
0.1875
0.1850
0.1825
0.1800
0.1775
Error
0
50
100
150
200
250
300
Iterations