Internal Insensitive Loss for Ordinal Classification

Structure of Code:

The following steps include the basic structure of code --

- 1. Read the entire data from the file.
- 2. Calculate the total number of classes or bins based on the provided bin size and size of data. number_of_classes = data_size / bin_size
- 3. Define the ranges (left extreme and right extreme) for each bin.
- 4. Segregate the data into files as "train" and "test" using Kfold algorithm.
- 5. Calculate values of different variables in the dual form and provide it to QP solver to find the Lagrangian constants.
- 6. Find weight vector using Lagrangian constants.
- 7. Classify the input samples using the obtained weight vector and determine accuarcy and mean absolute error loss.
- 8. Plot the graph based on the above results.

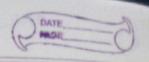
There are two files namely, binning.py and computation.py

binning.py: steps 1 to 4 computation.py: steps 5 to 8

Running the code:

python computation.py bin_size Example: python computation.py 50

Conversion of given primal to dual:



$$f = \frac{\lambda}{2} \| \| \omega \|^2 + \frac{c}{S} \cdot \mathcal{E}_{ii} + \mathcal{E}_{i}^*$$

S.t.

$$\begin{cases}
\xi_{i} \geq y_{i} - y + \langle x_{i}, w \rangle (y - y_{i}) + b_{y} - b_{y_{i}} \\
\xi_{i} \geq y - y_{i} + \langle x_{i}, w \rangle (y - y_{i}) + b_{y} - b_{y_{i}} \\
\xi_{i} \approx y - y_{i} + \langle x_{i}, w \rangle (y - y_{i}) + b_{y} - b_{y_{i}}
\end{cases}$$

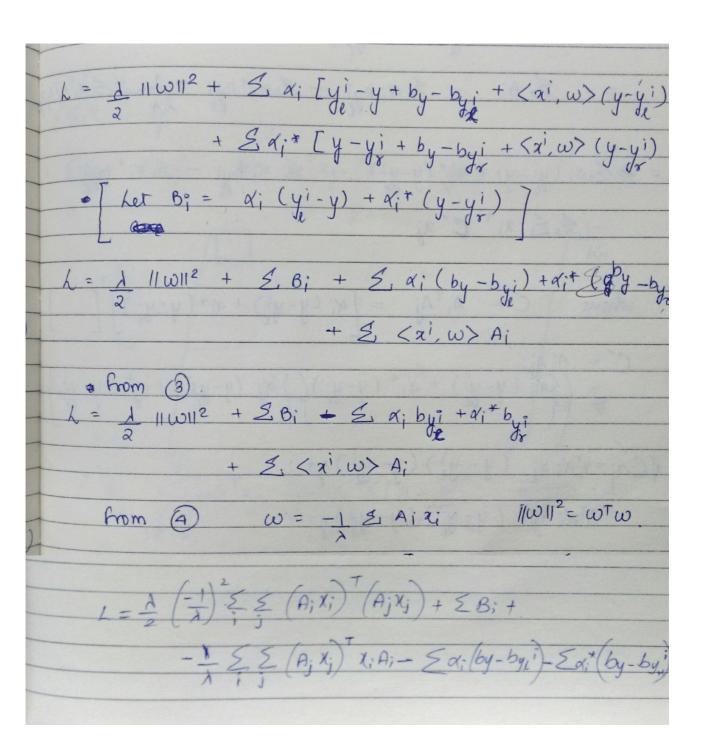
=>

$$L = \frac{1}{2} \|w\|^2 + \frac{1}{2} \frac{3}{m} \frac{8i}{i-1} \frac{8i}{m} + 8i^*$$

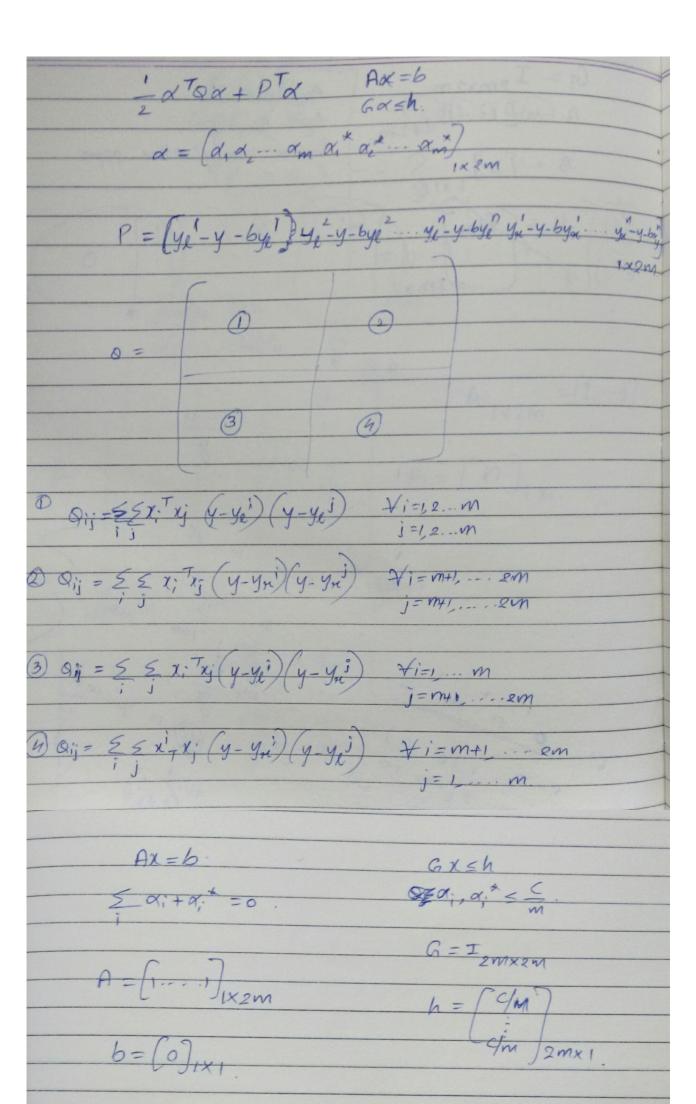
$$\frac{\delta L}{\delta \epsilon_{ii}} = \frac{C}{M} - \alpha_{i} - \delta_{i} = 0 \qquad -0$$

$$\frac{8L}{8E_i^*} = \frac{C}{M} - x_i^* - S_i^* = 0 \qquad \qquad \boxed{2}.$$

$$\frac{\delta L}{\delta \omega} = \lambda \omega + \sum \alpha_i \cdot \chi_i^{\dagger} \left(y - y_i^{\dagger} \right) + \alpha_i^{\star} \times \chi_i^{\dagger} \left(y - y_{\star i}^{\dagger} \right)$$



L= 1 & & A X; TA; TA; X; -1 & & & TA; TX;A; - 2 (di byi + di*byi) + 3, Bi = 3 Bi - 3 x; byi - 3 x; byi - 1 3 5 x 1 A 144 L= = & x; (yi-y) + x;* (y-yi) - & x; byi - 5, x; byi -122 xiT C' xj where C = AiTAj = [\arr (y-yi) + \arr (y-yi)] [C' = Ai Aj = (Sai (y-yi) + ai* (y-yi) (Saj (y-yi) + aj* (y-yi)) at Cz = zta; aj (y-yi) (y-yi) x;+) 3, xi xi (y-yi) (y-yi) xi+ 2Tx; * dj (y-yi) (y-yi) x; + 2, x; * x; * (y-yi) (y-yi) 2;



List of files written on our own: binning.py and computation.py