**Part 1: Supervised Data-space Partitioning**

- **Step A:** Use LSH [1] to initialise hashcode bits $B \in \{-1, 1\}^{N_{i,d} \times K}$, $N_{i,d}$: # training data-points, $K$: # bits

- **Repeat for $M$ iterations:**
  - **Step B:** Graph regularisation, update the bits of each data-point to be the average of its nearest neighbours
    
    
    
    $B \leftarrow \text{sgn} \left( \alpha SD^{-1}B + (1-\alpha)B \right)$

    $S \in \{0,1\}^{N_{i,d} \times N_{i,d}}$: adjacency matrix, $D \in \mathbb{Z}_{+}^{N_{i,d} \times N_{i,d}}$ diagonal degree matrix, $B \in \{-1,1\}^{N_{i,d} \times K}$ bits, $\alpha \in [0,1]$: interpolation parameter, $\text{sgn}$: sign function

  - **Step C:** Data-space partitioning, learn hyperplanes that predict the $K$ bits with maximum margin
    
    
    
    for $k = 1 \ldots K$:
    
    $\min ||w_{k}||^2 + C \sum_{i=1}^{N_{i,d}} \xi_{ik}$
    
    s.t. $B_{ik}(w_{k}^T x_i) \geq 1 - \xi_{ik}$ for $i = 1 \ldots N_{i,d}$

    $w_{k} \in \mathbb{R}^{D}$: hyperplane, $x_i$: image descriptor, $B_{ik}$: bit $k$ for data-point $x_i$, $\xi_{ik}$: slack variable

  - Use the learnt hyperplanes $\{w_{k} \in \mathbb{R}^{D} \}_{k=1}^{K}$ to generate $K$ projected dimensions: $\{y_{k} \in \mathbb{R}^{N_{i,d}} \}_{k=1}^{K}$ for quantisation.

**Part 2: Supervised Quantisation Threshold Learning**

- Thresholds $t_k = \{t_{k,1}, t_{k,2}, \ldots, t_{k,A}\}$ are learnt to quantise projected dimension $y_{k}$, where $T \in [1, 3, 7, 15]$ is the number of thresholds.

  - We formulate an $F_1$-measure objective function that seeks a quantisation respecting the constraints in $S$. Define $P_{k} \in \{0,1\}^{N_{i,d} \times N_{i,d}}$:
    
    
    
    $p_{k}(i,j) = \begin{cases} 1, & \text{if } y_{k,i} < y_{k,j} \leq t_{k,(\gamma+1)} \\ 0, & \text{otherwise.} \end{cases}$

  - $\gamma \in \mathbb{Z} \geq 0 \leq \gamma \leq T$. $P_{k}$ indicates whether or not the projections $(y_{k,i}, y_{k,j})$ fall within the same thresholded region. The algorithm counts true positives (TP), false negatives (FN) and false positives (FP):
    
    
    
    $TP = \frac{1}{2} || P \circ S ||_{1}$, $FN = \frac{1}{2} || S ||_{1} - TP$, $FP = \frac{1}{2} || P \| _{1} - TP$

  - $\circ$ is the Hadamard product, $|| \cdot ||_{1}$ is the $L_1$ matrix norm. TP is the number of +ve pairs in the same thresholded region, FP is the -ve pairs, and FN are the +ve pairs in different regions. Counts combined using $F_1$-measure optimised by Evolutionary Algorithms [3]:

  $F_1(t_k) = \frac{2||P \circ S||_{1}}{||S||_{1} + ||P||_{1}}$

**ILLUSTRATING THE KEY ALGORITHMIC STEPS**

(a) Initialisation

(b) Regularisation

(c) Partitioning

(d) Quantisation

**EXPERIMENTAL RESULTS (HAMMING RANKING AUPRC)**

- Retrieval evaluation on CIFAR-10. Baselines: single static threshold (SBQ) [1], multiple threshold optimisation (NMQ) [3], supervised projection (GRH) [2], and variable threshold learning (VBQ) [4].

  - LSH [1], PCA, SKLH [5], SH [6] are used to initialise bits in $B$

- Learning hyperplanes and thresholds (GRH-NPQ) most effective.

**CONCLUSIONS AND REFERENCES**

- Hashing model that learns the hyperplanes and thresholds. Found to have highest retrieval effectiveness versus competing models.

  - Future work: closer integration of both steps in a unified objective.