Learning to Project and Binarise for Hashing-based Approximate Nearest Neighbour Search

Sean Moran†

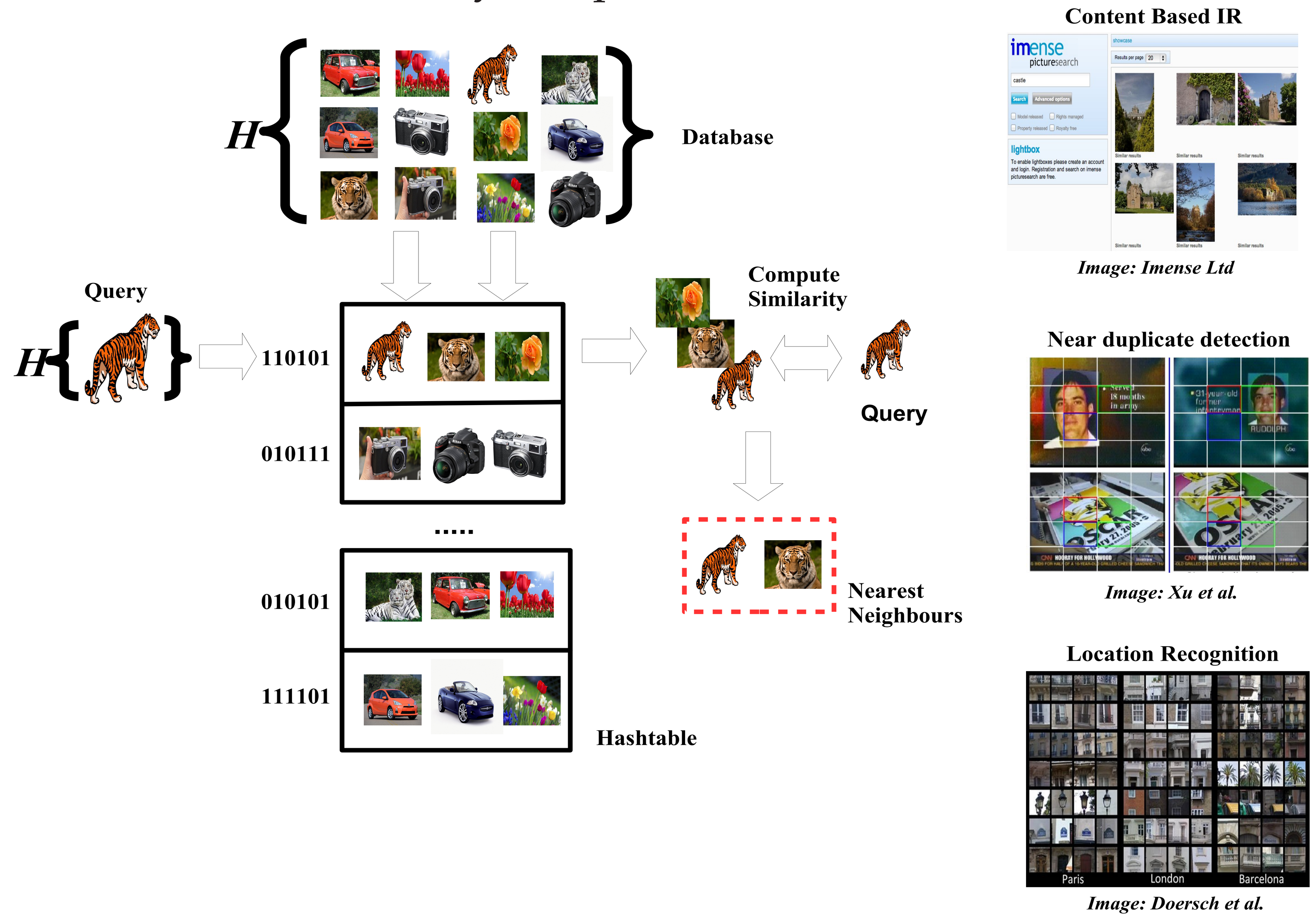
† sean.moran@ed.ac.uk

Learning the Hashing Thresholds and Hyperplanes

- **Research Question:** Can learning the hashing quantisation thresholds and hyperplanes lead to greater retrieval effectiveness than learning either in isolation?
- **Approach:** most hashing methods consist of two steps: data-point projection (hyperplane learning) followed by quantisation of those projections into binary. We show the benefits of explicitly learning the hyperplanes and thresholds based on the data.

Hashing-based approximate nearest neighbour search

- **Problem:** Nearest Neighbour (NN) search in image datasets.
- **Hashing-based approach:**
  - Generate a similarity preserving binary hashcode for query.
  - Use the fingerprint as index into the buckets of a hash table.
  - If collision occurs only compare to items in the same bucket.



- Hashtable buckets are the polytopes formed by intersecting hyperplanes in the image descriptor space. Thresholds partition each bucket into sub-regions, each with a unique hashcode. We want related images to fall within the same sub-region of a bucket.
- **This work:** learn hyperplanes and thresholds to encourage collisions between semantically related images.

Part 1: Supervised Data-space Partitioning

- **Step A:** Use LSH [1] to initialise hashcode bits $\mathbf{B} \in \{-1,1\}^{N_{trd} \times K}$
  N_{trd}: # 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

$$\mathbf{B} \leftarrow sgn\left(\alpha\, \mathbf{S}\mathbf{D}^{-1}\mathbf{B} + (1{-}\alpha)\, \mathbf{B}\right)$$

    * $\mathbf{S} \in \{0,1\}^{N_{trd} \times N_{trd}}$: adjacency matrix, $\mathbf{D} \in \mathbb{Z}_{+}^{N_{trd} \times N_{trd}}$ diagonal degree matrix, $\mathbf{B} \in \{-1,1\}^{N_{trd} \times K}$: bits, $\alpha \in [0,1]$: interpolation parameter, sgn: sign function
  - **Step C:** *Data-space partitioning*, learn hyperplanes that predict the K bits with maximum margin

$$\text{for } k = 1\ldots K: \quad \min\ ||\mathbf{w}_k||^2 + C\sum_{i=1}^{N_{trd}} \xi_{ik}$$
$$\text{s.t.} \quad B_{ik}(\mathbf{w}_k^{\intercal}\mathbf{x}_i) \geq 1 - \xi_{ik} \qquad \text{for } i = 1\ldots N_{trd}$$

    * $\mathbf{w}_k \in \Re^D$: hyperplane, $\mathbf{x}_i$: image descriptor, B_{ik}: bit k for data-point $\mathbf{x}_i$, ξ_{ik}: slack variable
- Use the learnt hyperplanes $\left\{\mathbf{w}_k \in \Re^D\right\}_{k=1}^{K}$ to generate K projected dimensions: $\left\{\mathbf{y}^k \in \Re^{N_{trd}}\right\}_{k=1}^{K}$ for quantisation.

Part 2: Supervised Quantisation Threshold Learning

- Thresholds $\mathbf{t}_k = [t_{k1}, t_{k2}, \ldots, t_{kT}]$ are learnt to quantise projected dimension $\mathbf{y}^k$, 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 contraints in $\mathbf{S}$. Define $\mathbf{P}^k \in \{0,1\}^{N_{trd} \times N_{trd}}$:

$$P_{ij}^k = \begin{cases} 1, & \text{if} \quad \exists_\gamma \quad s.t. \quad t_{k\gamma} \leq (y_i^k, y_j^k) < t_{k(\gamma+1)} \\ 0, & \text{otherwise.} \end{cases}$$

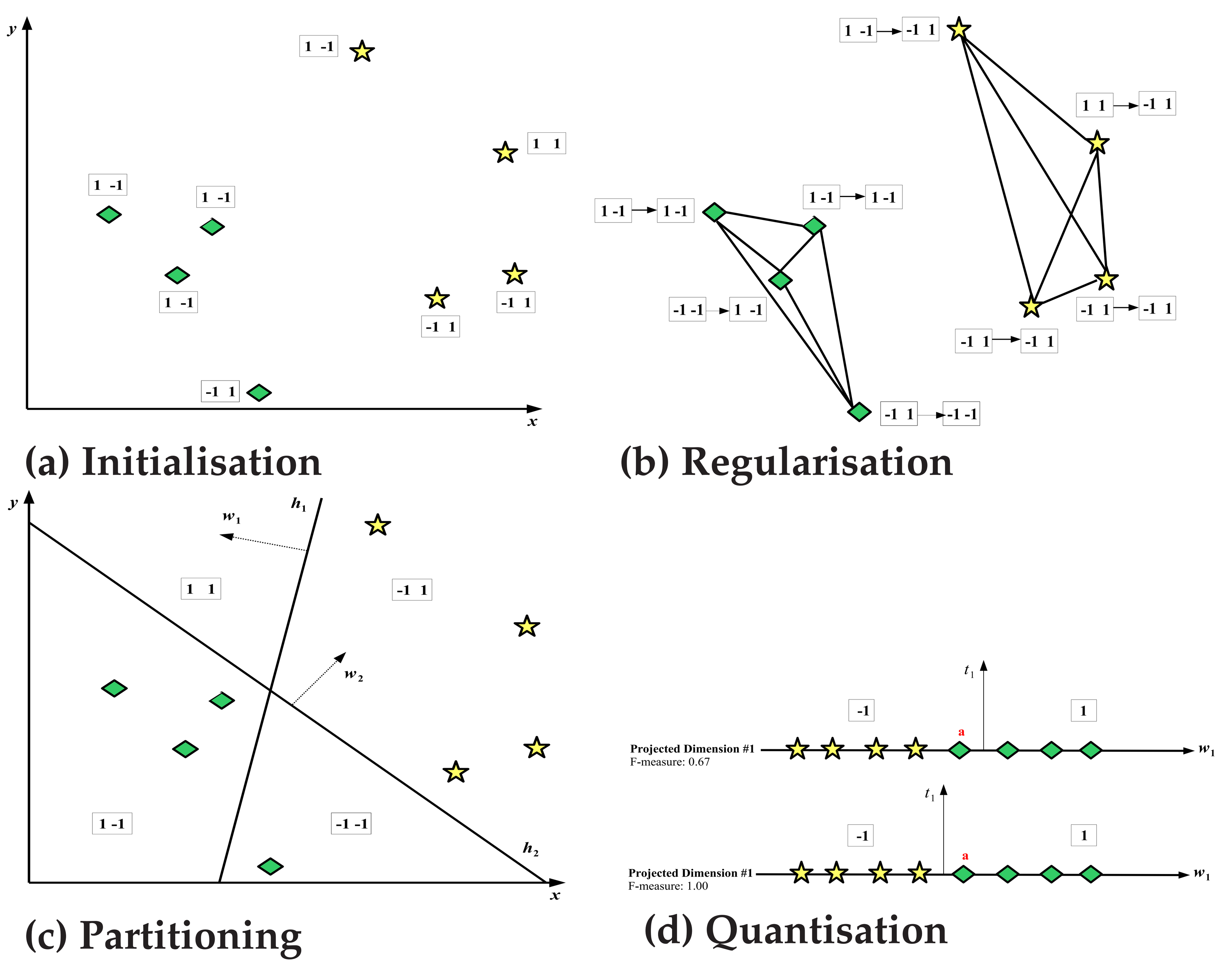
- $\gamma \in \mathbb{Z} \in 0 \leq \gamma \leq T$. $\mathbf{P}^k$ indicates whether or not the projections (y_i^k, y_j^k) fall within the same thresholded region. The algorithm counts *true positives* (TP), *false negatives* (FN) and *false positives* (FP):

$$TP = \frac{1}{2}\|\mathbf{P} \circ \mathbf{S}\|_1 \qquad FN = \frac{1}{2}\|\mathbf{S}\|_1 - TP \qquad FP = \frac{1}{2}\|\mathbf{P}\|_1 - TP$$

- $\circ$ is the Hadamard product, $\|.\|_1$ 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]:

$$F_1(\mathbf{t}_k) = \frac{2\|\mathbf{P} \circ \mathbf{S}\|_1}{\|\mathbf{S}\|_1 + \|\mathbf{P}\|_1}$$

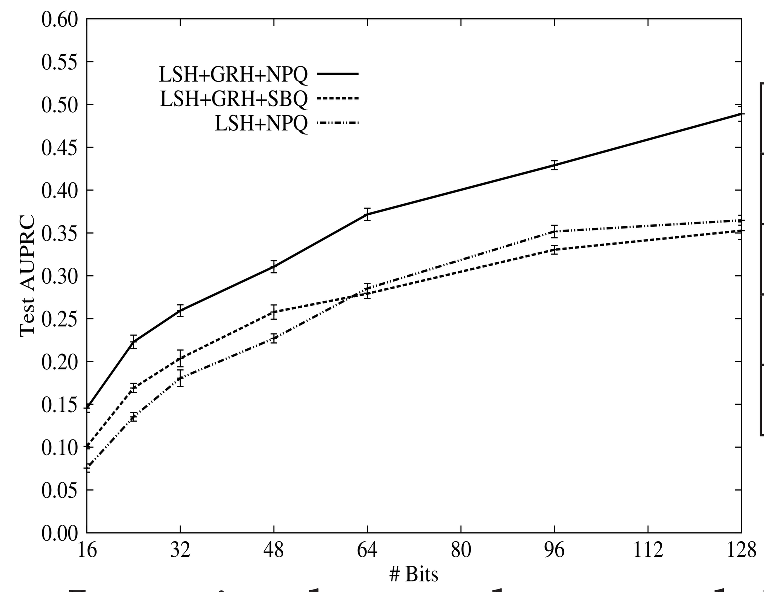
Illustrating the Key Algorithmic Steps



(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 (NPQ) [3], supervised projection (GRH) [2], and variable threshold learning (VBQ) [4].
- LSH [1], PCA, SKLSH [5], SH [6] are used to initialise bits in $\mathbf{B}$



	CIFAR-10 (AUPRC)				
	SBQ	**NPQ**	**GRH**	**VBQ**	**GRH+NPQ**
LSH	0.0954	0.1621	0.2023	0.2035	**0.2593**
SH	0.0626	0.1834	0.2147	0.2380	**0.2958**
PCA	0.0387	0.1660	0.2186	0.2579	**0.2791**
SKLSH	0.0513	0.1063	0.1652	0.2122	**0.2566**

- 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.

References: [1] Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: STOC (1998). [2] Moran S., and Lavrenko, V.: Graph Regularised Hashing. In Proc. ECIR, 2015. [3] Moran S., and Lavrenko, V., Osborne, M. Neighbourhood Preserving Quantisation for LSH. In Proc. SIGIR, 2013. [4] Moran S., and Lavrenko, V., Osborne, M. Variable Bit Quantisation for LSH. In Proc. ACL, 2013. [5] Raginsky M., Lazebnik, S. Locality-sensitive binary codes from shift-invariant kernels. In Proc. NIPS, 2009. [6] Weiss Y., Torralba, A. and Fergus, R. Spectral Hashing. In Proc. NIPS, 2008.