Learning to Project and Binarise for Hashing Based Approximate Nearest Neighbour Search

Sean Moran
sean.moran@glasgow.ac.uk
University of Glasgow, UK

ABSTRACT

In this paper we focus on improving the effectiveness of hashing-based approximate nearest neighbour search. Generating similarity preserving hashcodes for images has been shown to be an effective and efficient method for searching through large datasets. Hashcode generation generally involves two steps: bucketing the input feature space with a set of hyperplanes, followed by quantising the projection of the data-points onto the normal vectors to those hyperplanes. This procedure results in the makeup of the hashcodes depending on the positions of the data-points with respect to the hyperplanes in the feature space, allowing a degree of locality to be encoded into the hashcodes. In this paper we study the effect of learning both the hyperplanes and the thresholds as part of the same model. Most previous research either learn the hyperplanes assuming a fixed set of thresholds, or vice-versa. In our experiments over two standard image datasets we find statistically significant increases in retrieval effectiveness versus a host of state-of-the-art data-dependent and independent hashing models.

1. INTRODUCTION

Nearest neighbour search is the problem of finding the most similar data-points to a query in a database, and is a fundamental operation that has found wide applicability in many fields of study, ranging from Information Retrieval (IR) to Bioinformatics. Hashing-based approximate nearest neighbour (ANN) search methods permit the nearest neighbours to a query data-point to be retrieved in constant time [5]. Hashing permits constant time search per query by condensing both the database and the query into fixed-length compact binary hashcodes or fingerprints. The hashcodes exhibit the neighbourhood preserving property that similar data-points will be assigned similar (low Hamming distance) hashcodes. To compute these hashcodes, many hashing models partition the input feature space into disjoint regions with hyperplanes [1, 6, 9]. In the case of hyperplanes the polytope-shaped regions formed by the intersecting hyperplanes constitute the hashtable buckets. The hashtable key for a data-point is generated by simply determining which side of the hyperplanes the data-point lies. Depending on which side it falls a '0' or a '1' is appended to the hashcode for that data-point. By repeating this procedure for each hyperplane we can build up a hashcode for each data-point that is the same length as the number of hyperplanes partitioning the space.

Algorithmically this hashcode generation procedure can be accomplished in two separate steps performed in a pipeline: *projection* followed by *quantisation*. Projection involves a dot product of the feature vector representation of a data-point (e.g. bag of visual words) onto the hyperplane normal vectors positioned either randomly or in data-aware positions in the data-space. The hyperplanes should ideally partition the space in a manner that gives a higher likelihood that similar data points will fall within the same region, and therefore assigned the same hashcode. In the second step the real-valued projections are quantised into binary ('0' or '1') by thresholding the corresponding projected dimensions[1] typically with a single threshold placed at zero for mean centered data.

In our contribution we depart from most previous work and study the effect of learning both of the hyperplanes and the quantisation thresholds as part of the same hashing model. On image retrieval experiments over standard collections we demonstrate that learning both parameters can achieve a significant increase in retrieval effectiveness versus models that learn either parameter in isolation.

2. RELATED WORK

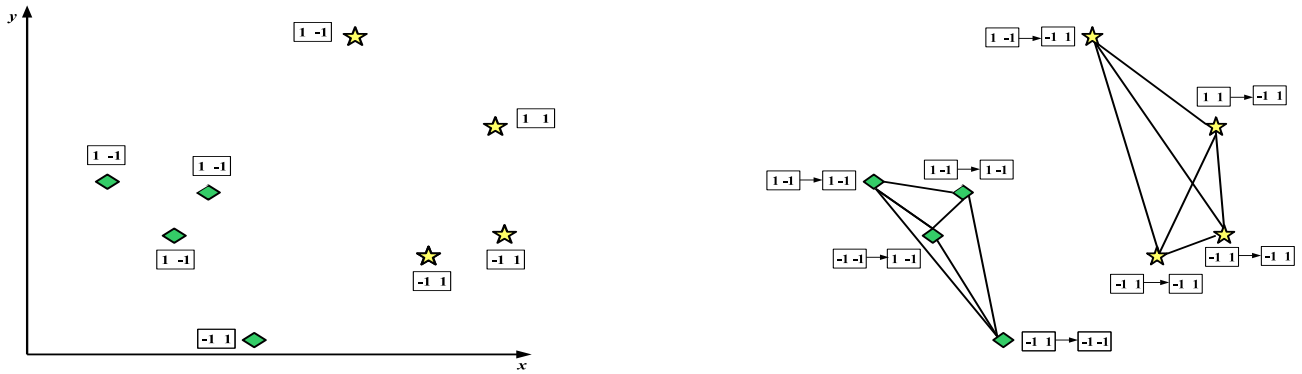
In the most closely related past research authors have generally focused on either learning the hashing hyperplanes [1, 4, 6] or the quantisation thresholds [2, 3, 7, 8] based on the distribution of the data. Seminal approaches for data-dependent hyperplane learning either solving an eigenvalue problem to generate a set of orthogonal hyperplanes, for example using Principal Components Analysis (PCA) [9], or frame a custom objective functions that uses pairwise labels to appropriately position the hyperplanes within the feature space [1]. Many of these models for hyperplane learning use a single threshold placed directly at zero (for mean centered data) to quantise the projections into binary. This approach is commonly known as Single Bit Quantisation (SBQ). Recently there has been significant interest in improving on

[1]We define a projected dimension as the collection of the real-valued projections (dot products) of all data-points onto the normal vector to a hyperplane.

SIGIR '16, July 17 - 21, 2016, Pisa, Italy

ISBN 978-1-4503-4069-4/16/07…$15.00

DOI: http://dx.doi.org/10.1145/2911451.2914766



(a) **Initialisation** (b) **Regularisation**

Figure 1: **Initialisation**: 2-bit hashcodes initialised by LSH. Coloured shapes indicated data-points, and those with the same shape and colour are 1-NNs. **Regularisation** (Step 1): The hashcodes are regularised over the adjacency matrix. Lines between points indicate a nearest neighbour relationship. The hashcodes are updated as shown by the directed arrows.

SBQ by learning one or more thresholds to quantise projections [2, 3, 8, 7]. These quantisation models use either an unsupervised objective such as k-means or squared error minimisation to learn a good set of thresholds for each projected dimension [7, 8], or propose a semi-supervised objective that takes into consideration the neighbourhood structure between the data-points in the input feature space [2, 3].

3. LEARNING TO PROJECT AND BINARISE

3.1 Overview of the Approach

We couple the graph regularised projection model (GRH) of [1] and the semi-supervised quantisation model (NPQ) of [2]. The hyperplanes are learnt using GRH and the projections quantised using NPQ. We aim to answer the following research question:

RQ-1: Can learning the hashing hyperplanes and *multiple* quantisation thresholds as part of the same model give a higher retrieval effectiveness than a model which learns either the hyperplanes or thresholds?

Initially, the model is first run for M iterations, in which the following three steps (1-3) are applied per iteration:

- **1) Regularisation:** Regularise the hashcodes using Equation (1):

$$\mathbf{B}_m \leftarrow \text{sgn}\left(\alpha\ \mathbf{S}\mathbf{D}^{-1}\mathbf{B}_{m-1} + (1-\alpha)\mathbf{B}_0\right) \quad (1)$$

  $\mathbf{B}_m \in \{-1,1\}^{N_{trd}\times K}$ are the hashcodes for the N_{trd} training data-points ($N_{trd} \ll N$) at iteration m, $\alpha \in [0,1]$ is the GRH interpolation parameter, $\mathbf{S}$ is the $N_{trd} \times N_{trd}$ adjacency matrix, which has S_{ij}=1 if $\mathbf{x}_i$ $\mathbf{x}_j$ are true nearest neighbours and 0 otherwise, $\mathbf{D}$ is a diagonal matrix containing the degree of each node in the graph. $\mathbf{B}_0$ is initialised using any existing projection function e.g. PCA.

- **2) Partitioning:** Solve K constrained optimisation problems in Equation (2), where K is the length of the hashcodes:

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

  $\mathbf{w}_k \in \mathbb{R}^D$ is the hyperplane normal vector, $\xi_{ik} \in \mathbb{R}_+$ are slack variables that allow points $\mathbf{x}_i$ to fall on the wrong side of hyperplane $\mathbf{h}_k$ and $C \in \mathbb{R}_+$ is the flexibility of margin. We use `liblinear` [10] as our solver.

- **3) Prediction:** Compute the N_{trd} dot products in Equation (3):

$$y_i^k = \mathbf{w}_k^\intercal \mathbf{x}_i \ \text{ for } i{=}\{1\ldots N_{trd}\} \text{ and } k{=}\{1\ldots K\} \quad (3)$$

  SBQ is applied to generate the updated hashcode matrix $\mathbf{B}_m$. We repeat steps 1-3 for M iterations. This is the GRH model of [1].

Having learnt the projections of our data-points in steps 1-3, we then quantise the learnt projections using NPQ [2]:

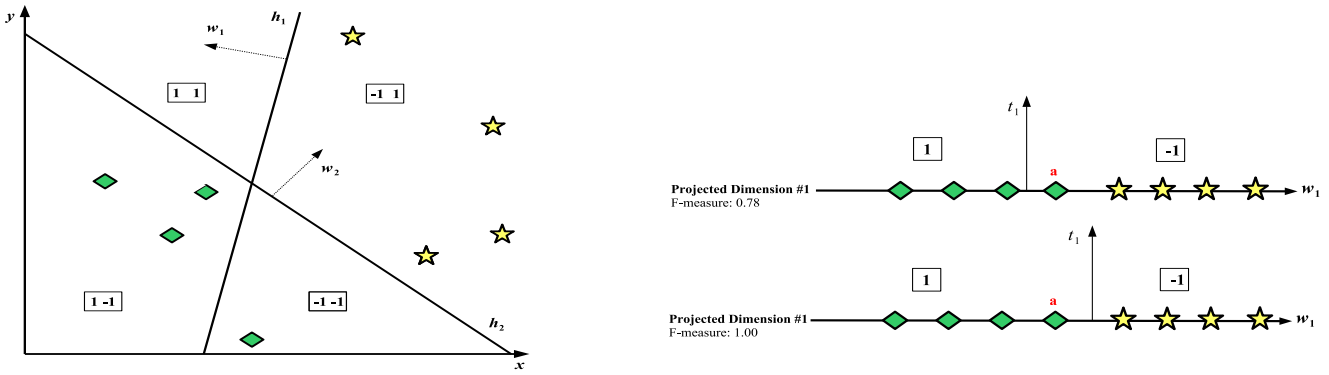
- **4) Quantisation:** Quantise $\mathbf{y}^k$ with thresholds $\mathbf{t}_k = [t_{k1}, t_{k2}, \ldots, t_{kT}]$ learnt by optimising $\mathcal{J}_{npq}(\mathbf{t}_k)$ in Equation (4):

$$\mathcal{J}_{npq}(\mathbf{t}_k) = F_1(\mathbf{t}_k) \quad (4)$$

  Here $T \in [1, 3, 7, 15]$ is the number of thresholds per projected dimension and $F_1(\mathbf{t}_k)$ is a supervised term that leverages the neighbourhood structure encoded in $\mathbf{S}$. For a fixed set of thresholds $\mathbf{t}_k = [t_{k1} \ldots t_{kT}]$ we define a per-projected dimension indicator matrix $\mathbf{P}^k \in \{0,1\}^{N_{trd}\times N_{trd}}$ with the property given in Equation (5):

$$P_{ij}^k = \begin{cases} 1, & \text{if} \quad \exists_\gamma \quad s.t. \quad t_{k\gamma} \leq (y_i^k, y_j^k) < t_{k(\gamma+1)} \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

  The index $\gamma \in \mathbb{Z}$ spans the range: $0 \leq \gamma \leq T$, where the scalar quantity T denotes the total number of thresholds for a projected dimension. Intuitively, matrix $\mathbf{P}^k$ indicates whether or not the projections (y_i^k, y_j^k) of any pair of data-points $(\mathbf{x}_i, \mathbf{x}_j)$ fall within the same thresholded region of the projected dimension $\mathbf{y}^k \in \mathbb{R}^{N_{trd}}$. Given thresholds $[t_{k1} \ldots t_{kT}]$, the algorithm counts *true positives* (TP), *false negatives* (FN) and *false positives* (FP) (Equations (6)-(8)).



(a) **Partitioning** (b) **Quantisation**

Figure 2: **Partitioning step** (Step 2): hyperplanes are positioned to separate opposing bits (-1,1) with maximum margin. **Quantisation step** (Step 4) using the projections for hyperplane $\mathbf{h}_1$ as an example. The top-most diagram shows the quantisation obtained with a threshold placed at zero. Point a is on the wrong side of the threshold. The bottom diagram shows the result obtained by optimising the threshold t_1 to maximise pairwise F_1. In this case the threshold is shifted so that point a receives the same bits as its neighbours.

$$TP = \frac{1}{2}\sum_{ij} P_{ij}S_{ij} = \frac{1}{2}\|\mathbf{P} \circ \mathbf{S}\|_1 \quad (6)$$

$$FN = \frac{1}{2}\sum_{ij} S_{ij} - TP = \frac{1}{2}\|\mathbf{S}\|_1 - TP \quad (7)$$

$$FP = \frac{1}{2}\sum_{ij} P_{ij} - TP = \frac{1}{2}\|\mathbf{P}\|_1 - TP \quad (8)$$

where $\circ$ denotes the Hadamard product and $\|.\|_1$ is the L_1 matrix norm defined as $\|\mathbf{X}\|_1 = \sum_{ij}|X_{ij}|$. TP is the number of positive pairs that are found within the same thresholded region, FP is the negative pairs found within the same region, and FN are the positive pairs found in different regions. The counts are combined using the familiar F_1-measure (Equation (9)):

$$F_1(\mathbf{t}_k) = \frac{2\|\mathbf{P} \circ \mathbf{S}\|_1}{\|\mathbf{S}\|_1 + \|\mathbf{P}\|_1} \quad (9)$$

Equation (9) is directly maximised using Evolutionary Algorithms [2], and encourages a clustering of the projected dimension that respects the constraints in $\mathbf{S}$.

Figures 1-2 provide an intuitive overview of steps 1,2 and 4.

4. EXPERIMENTAL EVALUATION

4.1 Experimental Setup

Groundtruth: We define the groundtruth nearest neighbours using the ϵ-NN paradigm[2], that is if a data-point is within a radius of ϵ to the query then it is deemed to be true nearest neighbour for the purposes of evaluation ([2, 6, 8, 7]). To evaluate the quality of the hashcodes, database images are ranked based on the Hamming distance to the hashcodes of the queries. The resulting ranked lists are used to compute the area under the precision recall curve (AUPRC). The higher the AUPRC, the better the quality of the hashcodes.

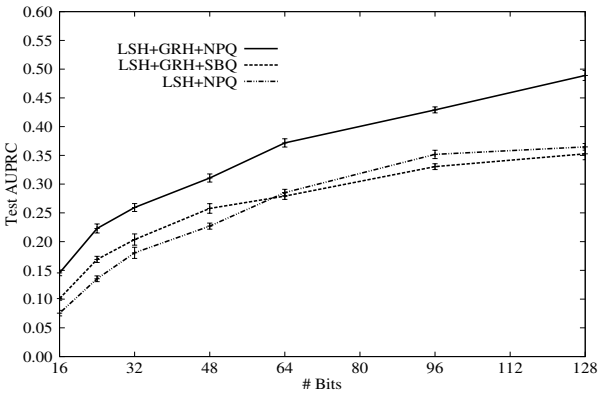
Parameter Optimisation: We use three thresholds ($T = 3$) per projected dimension for NPQ and use the Manhattan hashing codebook and ranking strategy [8]. This corresponds to a 2-bit per projected dimension encoding, which means that $K/2$ hyperplanes are learnt for a K-bit hashcode. For GRH we use a held out validation dataset to find the number of iterations $M \in \mathbb{Z}_+$, the interpolation parameter $\alpha \in [0, 1]$ and the flexibility of margin $C \in \mathbb{R}_+$ for the linear SVM. Our parameter tuning strategy for the GRH model is as follows: firstly, we set $C = 1$ and perform a grid search over $M \in \{1 \ldots 20\}$ and $\alpha \in \{0.1, \ldots, 0.9, 1.0\}$, selecting the setting that gives the highest validation AUPRC. To find M we stop the sweep when the validation dataset AUPRC falls for the first time, and set M to be the number of the penultimate iteration. Finally, we then find the optimal value for the flexibility of margin $C \in \{0.01, 0.1, 1.0, 10, 100\}$.

Datasets: CIFAR-10[3] and Tiny 100K image [8] datasets both encoded with GIST features. To define the test queries we randomly sample N_{teq}=1,000 data points with the remaining points forming the database for retrieval and the training dataset for learning the hyperplanes and quantisation thresholds. We set N_{trd}=2,000 for learning. Reported AUPRC figures are the average over 10 independent runs.

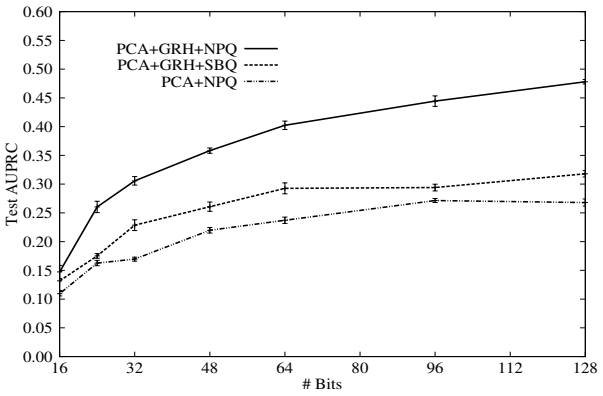
Initialisation Methods: LSH [5], Shift-invariant Kernel-based Locality-Sensitive Hashing (SIKH) [6], PCA Hashing (PCAH) [4] and Spectral Hashing (SH) [9].

[2]ϵ is set to be the average L_2 distance to the 50^{th} nearest neighbour. This parameter setting follows previous work [7, 8].

[3]https://www.cs.toronto.edu/~kriz/cifar.html



(a) **LSH projections**



(b) **PCA projections**

Figure 3: The effect of learning the hashing hyperplanes and quantisation thresholds as part of the same model **(GRH+NPQ)** versus independently **(GRH+SBQ, SBQ)** over a haschode length of 16-128 bits on the CIFAR-10 dataset for LSH and PCA. Bars show standard error of the mean.

	Tiny 100K				
	SBQ	**NPQ**	**GRH**	**VBQ**	**GRH+NPQ**
LSH	0.3623	0.4970	0.4791	0.4796	**0.5680▲▲**
SH	0.1431	0.4519	0.4837	0.5014	**0.5827▲▲**
PCA	0.0439	0.3785	0.4749	0.5116	**0.5791▲▲**
SKLSH	0.1585	0.4357	0.4337	0.5198	**0.5309**

Table 1: AUPRC on the **Tiny 100K** dataset with a hashcode length of 32 bits. ▲▲ indicates statistical significance (Wilcoxon signed rank test, $p < 0.01$) when comparing **GRH+NPQ** to the next best model.

	CIFAR-10				
	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▲▲**

Table 2: AUPRC on the **CIFAR-10** dataset with a hashcode length of 32 bits.

4.2 Experimental Results

The Hamming ranking retrieval results are shown in Tables 1-2 and Figures 3(a)-(b). We observe in Tables 1-2 that learning the hyperplanes and thresholds as part of the same combined model (GRH+NPQ) yields the highest retrieval effectiveness compared to learning the hyperplanes (GRH) or the thresholds (NPQ, VBQ) independently. For example, for LSH projections GRH+NPQ gains a relative increase in AUPRC of 60% over NPQ and 28% over GRH on CIFAR-10. Furthermore, the combination of GRH+NPQ outperforms the adaptive thresholds allocation model (VBQ) of [3] by a relative margin of 27%. Each of these increases are found to be statistically significant using a Wilcoxon signed rank test (p-value < 0.01). We find that the superior retrieval effectiveness of GRH+NPQ is maintained when the hashcode length is varied between 16-128 bits for both LSH and PCA projections (Figure 3(a)-(b)) on CIFAR-10. In most cases, significant increases in effectiveness are found for other popular projection functions including SH and SKLSH across both datasets (Tables 1-2). Based on these experimental results we can answer **RQ-1** in the affirmative.

5. CONCLUSIONS

In this paper we have explored the benefits of learning the hashing hyperplanes and multiple quantisation thresholds as part of the same hashing model. We introduce a simple coupling of two state-of-the-art models for projection function and quantisation threshold learning [1, 2] and find statistically significant increases in retrieval effectiveness. Future research will explore a tighter coupling between the learning of the hyperplanes and quantisation thresholds by creating a unified objective function encompassing both parameters that could be jointly learnt using gradient-based optimisers.

6. ACKNOWLEDGEMENTS

The author would like to thank Victor Lavrenko for his valuable feedback on this research.

7. REFERENCES

[1] S. Moran and V. Lavrenko. Graph Regularised Hashing. In *Proc. ECIR*, 2015.
[2] S. Moran, V. Lavrenko, and M. Osborne. Neighbourhood Preserving Quantisation for LSH. In *Proc. SIGIR*, 2013.
[3] S. Moran, V. Lavrenko, and M. Osborne. Variable Bit Quantisation for LSH. In *Proc. ACL*, 2013.
[4] J. Wang, S. Kumar, and SF. Chang. Semi-Supervised Hashing for Large-Scale Search. In *Proc. PAMI*, 2012.
[5] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. STOC*, 1998.
[6] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Proc. NIPS*, 2009.
[7] W. Kong and W. Li. Double-Bit Quantisation for Hashing. In *Proc. AAAI*, 2012.
[8] W. Kong, W. Li, and M. Guo. Manhattan hashing for large-scale image retrieval. In *Proc. SIGIR*, 2012.
[9] Y. Weiss, A. Torralba, and R. Fergus. Spectral Hashing. In *Proc. NIPS*, 2008.
[10] Rong-En Fan et al. LIBLINEAR: A Library for Large Linear Classification. In *JMLR, 9*, 2008.