Neighbourhood Preserving Quantisation for LSH

Sean Moran
School of Informatics
The University of Edinburgh
EH8 9AB, Edinburgh, UK
sean.moran@ed.ac.uk

Victor Lavrenko
School of Informatics
The University of Edinburgh
EH8 9AB, Edinburgh, UK
vlavrenk@inf.ed.ac.uk

Miles Osborne
School of Informatics
The University of Edinburgh
EH8 9AB, Edinburgh, UK
miles@inf.ed.ac.uk

ABSTRACT

We introduce a scheme for optimally allocating multiple bits per hyperplane for Locality Sensitive Hashing (LSH). Existing approaches binarise LSH projections by thresholding at zero yielding a single bit per dimension. We demonstrate that this is a sub-optimal bit allocation approach that can easily destroy the neighbourhood structure in the original feature space. Our proposed method, dubbed Neighbourhood Preserving Quantization (NPQ), assigns multiple bits per hyperplane based upon adaptively learned thresholds. NPQ exploits a pairwise affinity matrix to discretise each dimension such that nearest neighbours in the original feature space fall within the same quantisation thresholds and are therefore assigned identical bits. NPQ is not only applicable to LSH, but can also be applied to any low-dimensional projection scheme. Despite using half the number of hyperplanes, NPQ is shown to improve LSH-based retrieval accuracy by up to 65% compared to the state-of-the-art.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Measurement

Keywords

Locality Sensitive Hashing, Image Retrieval, Approximate Nearest Neighbour Search, Hamming Distance, Manhattan Distance

1. INTRODUCTION

Approximate nearest neighbour (ANN) search using hashing techniques is widely used to find objects of interest within large-scale datasets. In this scenario similarity preserving hash functions map nearby points in the original space into similar and substantially more compact binary codes. Using binary codes as surrogates for the original data points reaps two important computational benefits: firstly storage requirements are vastly reduced given the more compact nature of the binary codes; and secondly ANN search can be performed in sublinear time by treating the binary codes as hash keys.

The primary requirement for an effective hashing scheme is the generation of codes that use as few bits as possible while also mapping similar data points to binary codes. There are two distinct stages to most existing hashing schemes: *projection* and *quantisation*. A common first step in many binary encoding methods is to project the data into a lower dimensional space, for example, by using principal component analysis (PCA) or by using a Gaussian random matrix (LSH). The projected data points are then subsequently encoded into binary using a quantisation step. Both steps should ideally preserve the neighbourhood structure between the points in the original feature space.

There has been a significant amount of prior research that has sought to improve the projection stage. For example, by introducing machine learning methods to *learn* compact codes [11][2]. In contrast, the *quantisation* stage has attracted substantially less attention from the research community, but is arguably equally as important. Binarising real-valued features can lead to significant information loss if not performed judiciously. In this paper we focus on improving quantisation performance. Our primary hypothesis in this paper is that a pairwise affinity matrix (for example, based upon the Euclidean distance between points in the original feature space) can be an effective signal for quantizing projected dimensions[1] in a manner that explicitly maintains the affinity between the data points.

2. RELATED WORK

The seminal work on Locality Sensitive Hashing (LSH) demonstrated how random projections could be applied in the generation of binary codes that approximately maintain the distance between points in the original feature space [3]. LSH function families have the property that objects that are close in the original feature space have a higher probability of being allocated identical binary codes than objects that are further apart. The LSH hash codes permit sublinear retrieval time by acting as an index into the bucket of a hash table. Many LSH hash function families have been

[1]In this paper we define a *projected dimension* as the normal vector of the hyperplane.

SIGIR'13, July 28–August 1, 2013, Dublin, Ireland.

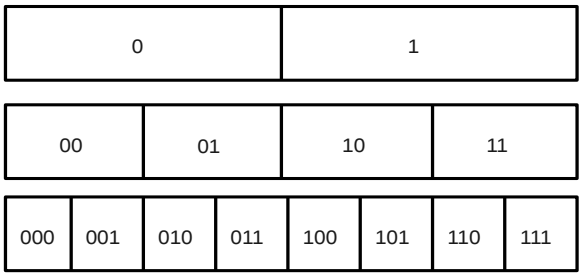

Figure 1: Partitioning of a projected dimension based on thresholds (solid vertical lines) and the associated binary encoding of each region. Top: bit allocation for single bit quantisation. Middle: NPQ with 2 bits per dimension and three thresholds. Bottom: NPQ with 3 bits per dimension and seven thresholds. This is the same bit encoding as in [5].

developed. For example, LSH for cosine similarity [1] intersects the space with random hyperplanes, with each resulting subspace forming a bucket of the hashtable. The number of LSH hyperplanes directly equate to the number of bits in the binary code for a data point. More precisely, if $\mathbf{x}.\mathbf{n_i} < 0$, $i \in \{1 \dots k\}$ for data point $\mathbf{x}$ and hyperplane normal vector $\mathbf{n_i}$, then the i-th bit is set to 0, and 1 otherwise. This effectively thresholds the projected dimension at zero. The k bits generated from k hyperplanes are concatenated to form the hash key for use in ANN search.

LSH uses *data-independent random projections*, which despite having some attractive theoretical guarantees, is not generally conducive to the production of compact hash codes. This problem has recently been circumvented by using machine learning techniques to find appropriate hash functions through optimization of an underlying hashing objective function for compact binary codes. Many of the more recent methods in this area, including Spectral Hashing [11] and Iterative Quantization [2] have demonstrated improvements on LSH in terms of the number of bits required to find good approximate nearest neighbours. These approaches are also based upon the idea of thresholding a projected dimension at zero to generate bits.

3. NEIGHBOURHOOD PRESERVING QUANTISATION

Our approach, dubbed *Neighbourhood Preserving Quantisation* (NPQ), allocates multiple bits per hyperplane based upon a novel adaptive thresholding scheme. In contrast, vanilla LSH thresholds at zero and assigns a single bit per hyperplane. Existing multiple-bit quantisation schemes seek to position the thresholds based solely on information within the projected space, for example by using a k-means clustering on each projected dimension [5][4]. In this work we show that the affinity between the data points in the original space can be a valuable signal for optimal threshold positioning.

We use $X \in \Re^{n \times m}$, to denote the data matrix, where n is the number of instances and m is the number of features. We are given n data points $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, $x_i \in \Re^m$ which form the rows of the data matrix X. The objective is to learn a binary code matrix $B \in \{0,1\}^{n \times b}$, where b is the total number of bits in our binary representation. In previous work the binary encoding function for bits $k = \{1 \dots b\}$ is given by $h_k(\mathbf{x}) = sgn(\mathbf{n}_k \mathbf{x})$, where $\mathbf{n}_k$ is a vector normal to the hyperplane and $sgn(v) = 1$ if $v \geq 0$ and 0 otherwise.

As argued by previous authors [5] [4], thresholding at zero in this manner is suboptimal given that the area of highest data density typically occurs in the region around zero. There is therefore a high chance that neighbouring points may fall on different sides of the threshold, and therefore be assigned different bits in our encoding. NPQ seeks to overcome this problem by using multiple thresholds to partition the dimension in such a way that points close in the original feature space are more likely to fall between the same thresholds, and therefore are assigned identical bit codes.

Let $T = \{t_1, t_2, \dots, t_u\}$ be a set of thresholds, where $u \geq 1$, $t_i \in \Re$ with $t_1 \leq t_2 \dots \leq t_u$. We denote by t_0 and t_{u+1} the leftmost and rightmost ends of the sorted dimension[2], respectively. NPQ directly leverages a *pairwise affinity matrix*, $S \in \Re^{n \times n}$, where $S_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are ϵ-nearest neighbours in the original feature space, and 0 otherwise, to guide the positioning of the thresholds in T along the projected dimension. In our terminology a pair of data points $\mathbf{x}_i$ and $\mathbf{x}_j$ with $S_{ij} = 1$ are deemed a *positive pair*, and a pair of points with $S_{ij} = 0$ a *negative pair*. Intuitively, the pairwise affinity matrix specifies which points should fall between the same thresholds, and therefore be assigned the same bit codes. NPQ seeks to exploit this valuable signal.

We define a *region* r_a as all the projected values y_i between thresholds t_a and t_{a+1}, where $a \in \{0, \dots, u\}$. For u thresholds, there are $u+1$ regions. NPQ counts the number of *true positives* (TP), *false positives* (FP) and *false negatives* (FN) across all regions, r_a, of the projected dimension.

$$TP = \sum_{a=0}^{u} \sum_{ij:\{y_i,y_j\} \in r_a} S_{ij} \qquad FN = \sum_{ij} S_{ij} - TP$$

$$FP = \sum_{a=0}^{u} \sum_{ij:\{y_i,y_j\} \in r_a} (1 - S_{ij})$$

Intuitively, TP are the number of positive pairs that are found within the same region, FP is the number of negative pairs found within the same region, and FN are the number of positive pairs found in different regions of the threshold partitioned projected dimension. We combine the TP, FP and FN counts by computing the F_1 score. The overall NPQ objective function Z_{npq} is a convex combination of the F_1 score and the regularization term $\Omega(T_{1:u})$:

$$Z_{npq} = \alpha F_1 + (1 - \alpha)(1 - \Omega(T_{1:u})) \tag{1}$$

Here we define $\Omega(T_{1:u})$ as:

$$\Omega(T_{1:u}) = \frac{1}{\sigma} \sum_{a=0}^{u} \sum_{i:y_i \in r_a} \{y_i - \mu_{r_a}\}^2$$

Where $\sigma = \sum_{i=1}^{n} \{y_i - \mu_d\}^2$, μ_d denotes the mean of the dimension and μ_{r_a} is the mean of the points in region r_a, that is, between thresholds t_a and t_{a+1}. NPQ varies the position of the thresholds along the projected dimension until a maximum of Z_{npq} is attained. The optimisation approach is shown in Figure 2. We use random restarts for the threshold positions to ensure that the optimisation remains linear in the number of data-points rather than $O(n^u)$.

[2]We refer to a projected dimension as the set of real-valued projections of all data-points for one particular hyperplane

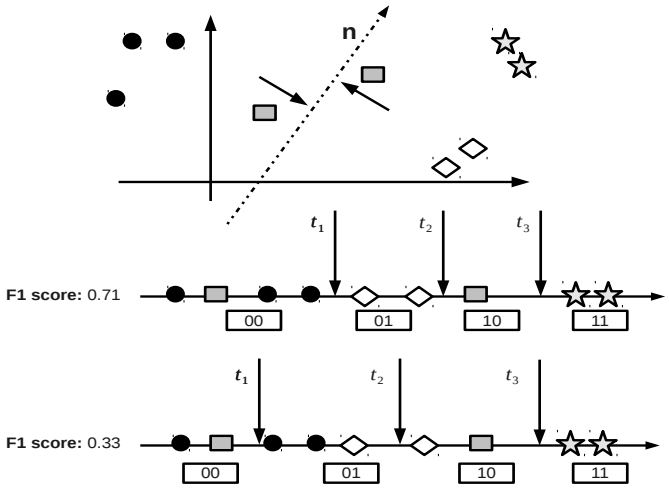

Figure 2: Top: data points in the original 2 dimensional space. Middle: Projection of points onto the normal vector n of a hyperplane. This illustrates a good positioning of the thresholds. In this case most pairs receive similar binary bits. Bottom: a poor positioning of the thresholds. Here the thresholds divide neighbouring pairs into different regions thereby causing differences in their bit assignments. NPQ favours the threshold positioning that maximises the F1 score.

4. EXPERIMENTS

4.1 Datasets

We evaluate the effectiveness of NPQ in the domain of image retrieval, although our approach is general and can be used for other types of data (for example, text, video). To do so, we test against three publicly available image datasets: *22k Labelme* consisting of 22,019 images represented as 512 dimensional Gist descriptors [8]; *CIFAR-10* a dataset of 60,000 images represented as 512 dimensional Gist descriptors; and *100k TinyImages* a collection consisting of 100,000 images, represented by 384 dimensional Gist descriptors, randomly sub-sampled from the original 80 million tiny images dataset. The datasets and and associated features are identical to that used in previous related work [5] [4]. This ensures that our results are directly comparable to previously published figures.

4.2 Projection Methods

We evaluate NPQ quantisation performance with *five* projection schemes: LSH-based projections [3], Shift-invariant Kernel-based Locality-Sensitive Hashing (SIKH) [9], Principal Components Analysis Hashing (PCAH) [10], Spectral Hashing (SH) [11] and Iterative Quantisation (ITQ) [2]. All five projections are assigned 2 bits per dimension from the encoding scheme in Figure 1.

4.3 Baselines

NPQ quantisation performance is compared against *four* state-of-the-art quantisation schemes in addition to the standard threshold at zero technique: single bit quantisation (SBQ) [3], Manhattan Hashing (MQ) [5], Double Bit Quantisation (DBQ) [4] and Hierarchical Quantisation (HQ) [6].

4.4 Evaluation Protocol

In all experiments we follow previously accepted procedure [5] and randomly select 1000 data points as queries, with the remaining points being used to train the hash functions. All experimental results are averaged over 10 random training and testing partitions, and the average result reported.

We define the ϵ-*neighbours* of each data-point based upon a data-dependent threshold ϵ which is computed by sampling 100 training data-points at random from the training dataset. The threshold ϵ is set to the Euclidean distance at which these training points have 50 nearest neighbours on average. This threshold is used to determine the nearest neighbours for training and testing. To evaluate the quality of retrieval we use the Euclidean ground truth to compute precision, recall and the area under the precision-recall curve (AUPRC)[3].

The NPQ affinity matrix is computed by thresholding the Euclidean distance matrix by the average distance to the 50th nearest neighbour. The affinity matrix is computed using the training dataset only. The parameter α (Equation 1) is set based on a held-out validation dataset[4]. As the core contribution of NPQ lies in our novel objective function we adopt both the binary bit encoding method (Figure 1) and Manhattan distance metric used by MQ [5].

4.5 Results

Experimental results are presented in Table 1 and Figure 3. Table 1 demonstrates that NPQ outperforms all state-of-the-art quantisation schemes at 32-bits across *five* different projection methods, and *three* different datasets. We find a 33% performance gain over MQ for LSH-based projections for 22k Labelme. This result is statistically significant based upon a paired t-test across 10 random training/testing partitions of the dataset (*p-value*: $\leq 1.7 \times 10^{-5}$). We also find statistically significant gains in performance on the larger CIFAR-10 and 100k TinyImages datasets. For CIFAR-10, NPQ obtains a substantial 65% increase in AUPRC over MQ for LSH at 32 bits (*p-value*: $\leq 1.0 \times 10^{-7}$). For 100k TinyImages NPQ achieves a 37% increase in AUPRC over MQ with LSH at 32 bits (*p-value*: $\leq 2.5 \times 10^{-12}$). Figure 3 presents the *precision-recall* (PR) curves (at 32 bits) and *AUPRC vs number of bits* for all three datasets. NPQ obtains improved precision at all recall levels, in addition to decisively dominating both SBQ and MQ in terms of AUPRC between 8 to 128 bits. We confirm our primary hypothesis that a pairwise affinity matrix is a beneficial signal for guiding threshold-based quantisation.

5. CONCLUSIONS

This paper presents the neighbourhood preserving quantization (NPQ) method for approximate similarity search. NPQ leverages adaptively learned thresholds to quantize LSH projections into multiple bits based upon a pairwise affinity matrix. Our results show that retrieval performance can be significantly enhanced by using the neighbourhood information in the original feature space to inform the place-

[3]The authors in [5] [4] use AUPRC and not Mean Average Precision (MAP).

[4]We find a dependence of α on bit length: $\alpha = 1.0$ is effective for bit lengths ≤ 128 bits, with settings of $\alpha < 1.0$ effective for higher bit lengths.

Dataset	22k LabelMe					CIFAR-10					100k TinyImages				
	SBQ	HQ	MQ	DBQ	NPQ	SBQ	HQ	MQ	DBQ	NPQ	SBQ	HQ	MQ	DBQ	NPQ
ITQ	0.277	0.316	0.354	0.308	**0.408**	0.272	0.224	0.235	0.222	**0.407**	0.494	0.396	0.428	0.410	**0.660**
SIKH	0.049	0.051	0.072	0.077	**0.107**	0.042	0.036	0.063	0.047	**0.090**	0.135	0.139	0.221	0.182	**0.365**
LSH	0.156	0.121	0.138	0.123	**0.184**	0.119	0.067	0.093	0.066	**0.153**	0.361	0.301	0.340	0.285	**0.464**
SH	0.080	0.154	0.221	0.182	**0.250**	0.051	0.074	0.135	0.111	**0.167**	0.117	0.187	0.237	0.136	**0.356**
PCAH	0.050	0.141	0.191	0.156	**0.220**	0.036	0.065	0.137	0.107	**0.153**	0.046	0.212	0.257	0.295	**0.312**

Table 1: Area under the Precision Recall curve (AUPRC) at 32 bits. The best score is shown in bold face.

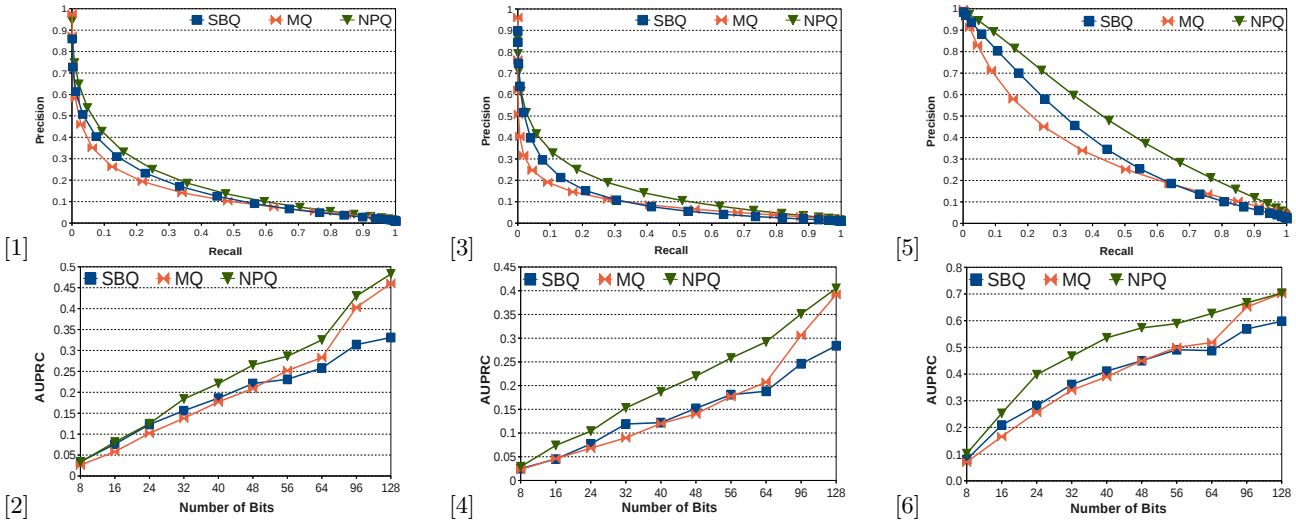

Figure 3: [1] LSH PR curve for 22k Labelme [2] LSH AUPRC on 22k Labelme [3] LSH PR curve for CIFAR-10 [4] LSH AUPRC for CIFAR-10 [5] LSH PR curve for 100k TinyImages [6] LSH AUPRC for 100k TinyImages

ment of quantisation thresholds. NPQ is orthogonal to existing approaches for improving the accuracy of LSH, for example multi-probe LSH [7], and can be applied alongside these techniques to further improve retrieval performance. An interesting avenue for future work would be the development of a principled method for selecting a variable number of bits per dimension that does not rely on either a projection-specific measure of hyperplane informativeness (e.g. magnitude of the eigenvalues) or computationally expensive cross-validation.

Acknowledgements

We are grateful to Weihao Kong for making available his feature sets for the CIFAR-10 and 100k Tiny Image datasets.

6. REFERENCES

[1] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 380–388, New York, NY, USA, 2002. ACM.

[2] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 817–824, Washington, DC, USA, 2011. IEEE Computer Society.

[3] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[4] W. Kong and W.-J. Li. Double-bit quantization for hashing. In *AAAI*, 2012.

[5] W. Kong, W.-J. Li, and M. Guo. Manhattan hashing for large-scale image retrieval. In *SIGIR*, pages 45–54, 2012.

[6] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.

[7] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 950–961. VLDB Endowment, 2007.

[8] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vision*, 42(3):145–175, May 2001.

[9] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1509–1517, 2009.

[10] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(12):2393–2406, Dec. 2012.

[11] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.