Chapter 1

Introduction

1.1 Motivation

The rapid growth of the World Wide Web (WWW) over the past two decades has brought with it a phenomenal increase in the amount of image, video and text based data being collected, stored and shared across the world. This phenomenon has been fuelled by the popularity of social media networks, cheap disk storage and the wide availability of Internet-enabled smartphones. For example it has been estimated that Facebook has in the order of 300 million images uploaded per day[1], YouTube receives 10 years worth of content per day[2] and there are now estimated to be well over 1 trillion web pages[3] in existence (Murphy (2012)). Figure 1.1 illustrates the explosive growth of images being uploaded onto popular social media websites and applications during the period 2005-2014. The trend towards real-time video sharing over the Internet with applications such as Periscope, involving a medium that is many times the size of individual images or documents, will severely exacerbate this torrent of data. In the near-term future the emergence of the Internet-of-Things (IoT) and Smart Cities promise to add further fuel to this fire, hinting at a connected society in which Internet linked sensors embedded in everyday objects, such as CCTV cameras and thermostats, produce an abundance of data that is automatically captured, stored and analysed so as to produce actionable insights for interested citizens and government stakeholders (Albakour et al. (2015)). The sheer scale of the data being produced around the world brings with it a need for computationally efficient algorithms that ensure the storage

[1] Velocity 2012: Jay Parikh, "Building for a Billion Users"

[2] http://www.youtube.com/yt/press/en-GB/statistics.html

[3] http://googleblog.blogspot.co.uk/2008/07/we-knew-web-was-big.html

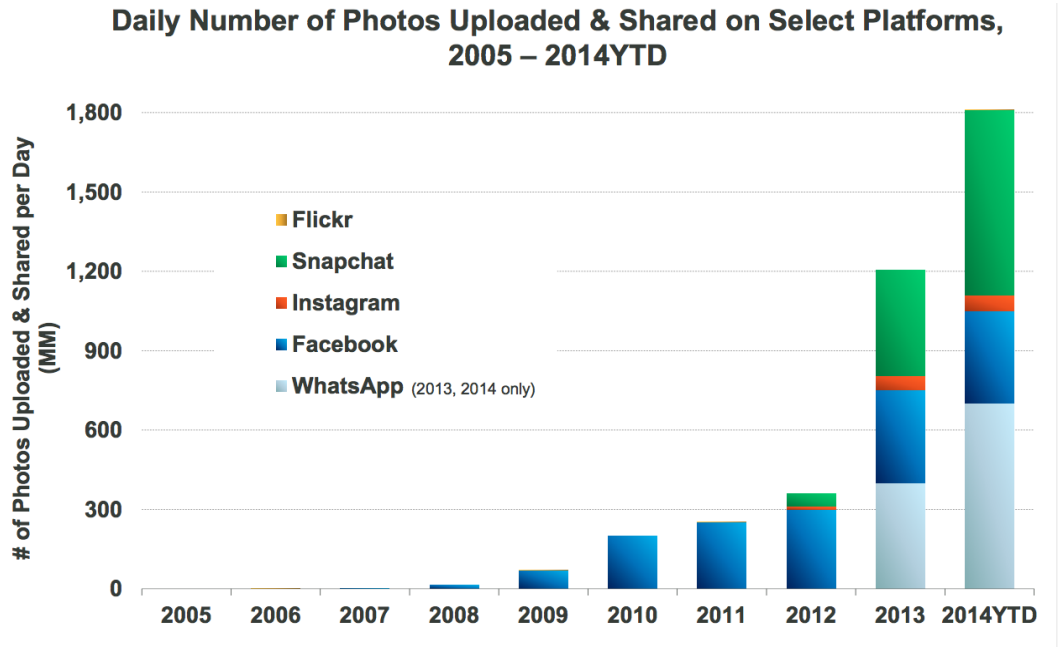


Figure 1.1: The amount of images being uploaded to popular social media websites (Facebook, Flickr) and mobile applications (Instagram, SnapChat, WhatsApp) has undergone a dramatic growth since 2005. Efficient algorithms for searching through such large image datasets are needed now more than ever. This chart has been copied directly from slide 62 of the talk *"Internet Trends 2014 - Code Conference"* given by the venture capitalist Mary Meeker of Kleiner Perkins Caufield Byers (KPCB): `http://www.kpcb.com/blog/2014-internet-trends` (URL accessed on 16/12/15).

requirements and processing overhead do not grow with the quantity of the data being produced.

In this thesis I focus on the problem of nearest neighbour (NN) search where the goal is to find the most similar data-point(s) to a query in a large database. Similarity is typically judged by representing the data-points as fixed dimensional vectors in a vector space and computing a distance metric such as the Euclidean or cosine distance. In this case data-points with a sufficiently low distance are judged to be nearest neighbours. Due to its generality and usefulness nearest-neighbour search finds application in many areas of Science, ranging from the field of Information Retrieval (IR) where we wish to find documents relevant to a query, to the problem of Genomic assembly in the field of Bioinformatics. The näive way to solve the nearest-neighbour search problem would be to compare the query to every data-point in the database, a method known as *brute-force search*. Brute-force search is only feasible in relatively small databases where performing the number of required comparisons between the

data-points remains computationally tractable. Given the linear scaling of the query time with respect to the dataset size it is impossible to exhaustively search large-scale datasets consisting of millions to billions of data-points for nearest neighbours in a reasonable amount of time. This problem is compounded in the streaming data scenario where data-points need to be processed sequentially in real-time with potentially no end to the amount of incoming data. To efficiently find nearest-neighbours in large-scale datasets, algorithms are required that offer a query time that is independent of the dataset size.

Hashing-based approximate nearest neighbour (ANN) search methods are a popular class of algorithms that permit the nearest neighbours to a query data-point to be retrieved in constant time, independent of the dataset size. Hashing has proved to be an extremely useful method for ANN search over high-dimensional, large-scale datasets that are prevalent in the modern data-rich world. 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. Crucially, unlike cryptographic hash functions such as MD5 or SHA-1, the data-points need not be identical to receive matching hashcodes. Rather the degree of similarity between the hashcodes is a direct function of the similarity between the feature representation of the data-points. This property is ideal for my chosen task of image retrieval where we rarely wish to find only those images that are identical down to the pixel level. Most people would deem two images to be related even if the semantically equivalent objects (e.g. a tiger) depicted in both images are in widely different poses, and therefore the images have a completely different pixel consistency.

The aforementioned similarity preserving property enables the hashcodes to be used as the keys into the buckets of hashtables so that similar, but not necessarily identical, images will collide in the same buckets (Figure 1.2). This is a rather different use-case to the typical application of hashtables in Computer Science in which it is imperative to avoid collisions between non-identical data-points. In hashing-based ANN search we are actively encouraging collisions between similar data-points. The bucketing of the data-points drastically reduces the computational overhead of nearest neighbour search by reducing the number of comparisons that are required between the data-points: at query time we need only compare our query to those data-points colliding in the same buckets. There is no free lunch however as we pay for the reduced query time with a non-zero probability of failing to retrieve the closest nearest

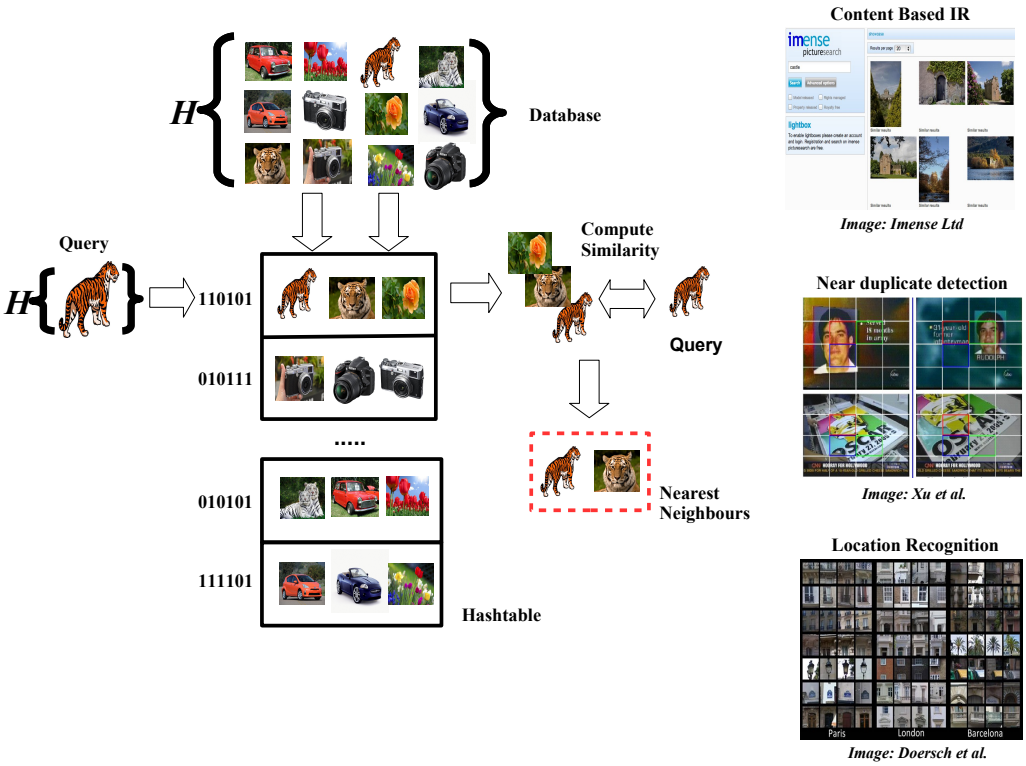


Figure 1.2: Nearest neighbour search with hashcodes. Similarity preserving binary codes generated by a hash function $\mathcal{H}$ can be used as the indices into the buckets of a hashtable for constant time search. Only those images that are in the same bucket as the query need be compared thereby reducing the size of the search space. The focus of this thesis is learning the hash function $\mathcal{H}$ to maximise the similarity of hashcodes for similar data-points. On the right-hand side I present examples of tasks for which nearest neighbour search has proved to be fundamental: from content-based information retrieval (IR) to near duplicate detection and location recognition. The three images on the right have been taken from Imense Ltd (`http://www.imense.com`) and Doersch et al. (2012); Xu et al. (2010); Grauman and Fergus (2013).

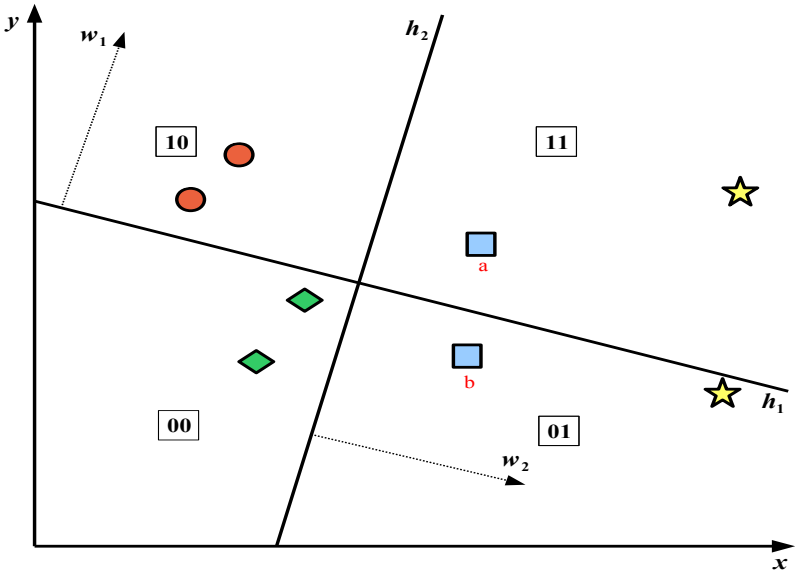
neighbours in the case where they happen to fall in different buckets. Nevertheless this quantifiable non-zero false negative probability turns out to be an acceptable trade-off in many application areas in which sub-optimal nearest neighbours can be almost as good as finding the exact nearest neighbour (Dean et al. (2013); Petrović et al. (2010)).

This thesis makes fundamental contributions to increasing the retrieval effectiveness of the core algorithmic processes underlying a well-known and widely applied method for hashing-based ANN search. I evaluate the effectiveness of these contributions on the task of content-based image retrieval, a signature problem encountered within the fields of IR and Computer Vision that is characterised by an abundance

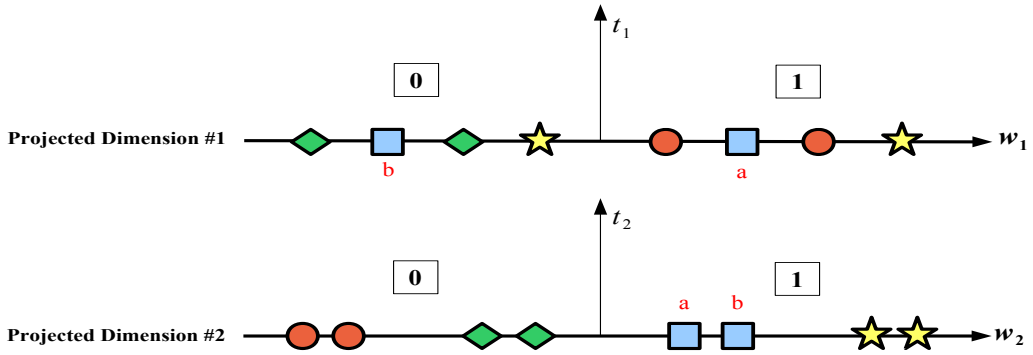
of data and the need for accurate and efficient search. Hashing-based ANN has also shown great promise in terms of efficient query processing and data storage reduction across a wide range of other interesting application areas within IR and Computer Vision. For example Petrović et al. (2010) present an efficient method for event detection in Twitter that scales to unbounded streams through a novel application of Locality Sensitive Hashing (LSH), a seminal randomised approach for ANN search (Indyk and Motwani (1998)). In the streaming data scenario of Petrović et al. (2010) the $O(N)$ worst case complexity of inverted indexing is undesirable, motivating the use of LSH to maintain a constant $O(1)$ query time[4]. Hashing-based ANN has also proved particularly useful for search over dense and much lower dimensional (compared to text) feature vectors, such as GIST (Oliva and Torralba (2001)), that are commonly employed in the field Computer Vision. In one recent application of LSH within Computer Vision, similarity preserving hashcodes have been successfully used for fast and accurate detection of 100,000 object classes on just a single machine (Dean et al. (2013)).

The ANN search hashing models I will examine and build upon in this thesis all partition the input feature space into disjoint regions with a set of hypersurfaces, either linear (hyperplanes) or non-linear. In the case of linear hypersurfaces the polytope-shaped regions formed by the intersecting hyperplanes constitute the hashtable buckets (Figure 1.3). 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. Intuitively, the hashcode can be thought of as an identifier that captures the geometric position of the data-points within the input feature space with each bit encoding the position of the data-point with respect to a given hyperplane. Algorithmically this hashcode generation procedure can be accomplished in two separate steps performed in a pipeline: *projection* followed by *quantisation*. This procedure is illustrated with a toy example in Figure 1.3. Projection involves a dot product of the feature vector representation of a data-point onto

[4]This is only true if we ignore the hashing cost (cost of generating the hashcode) and assume that each database data-point goes into its own hashtable bucket. In practice the LSH computational cost for a single hashtable and a single data-point is a sum of the hashing cost ($O(KD)$), lookup cost ($O(1)$) and the candidate test cost ($O(ND/2^K)$), where K is the hashcode length and assuming a uniform distribution of data-points to buckets. Observe that there is a trade-off between the hashing cost and the candidate test cost, both of which are dependent on K. For example, in the situation where the data-points are evenly distributed into their own hashtable bucket ($N = 2^K$), the total computational cost for LSH is actually sub-linear ($O(D \log N)$).



(a) **Projection**



(b) **Quantisation**

Figure 1.3: The projection and quantisation operations. In Figure (a) a 2D space is partitioned with two hyperplanes $\mathbf{h}_1$ and $\mathbf{h}_2$ with normal vectors $\mathbf{w}_1, \mathbf{w}_2$ creating four buckets. Data-points are shown as coloured shapes, with similar data-points having the same colour and shape. The hashcode for each data-point is found by taking the dot-product of the feature representation onto the normal vectors ($\mathbf{w}_1$, $\mathbf{w}_2$) of each hyperplane. The resulting projected dimensions are binarised by thresholding at zero (Figure (b)) with two thresholds t_1, t_2. Concatenating the resulting bits yields a 2-bit hashcode for each data-point (indicated by the unfilled squares). For example the projection of data-point a is greater than threshold t_1 and so a '1' is appended to its hashcode. Data-point a's projection onto normal vector $\mathbf{w}_2$ is also greater than t_2 and so a '1' is further appended to its hashcode. The hashcode for data-point a is therefore '11' which is also the label for the top-right region of the feature space in Figure (a).

the hyperplane normal vectors positioned either randomly or in data-aware positions in the feature 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[5] typically with a single threshold placed at zero for mean centered data.

Despite the success and wide application of algorithms for hashing-based ANN search there still remains considerable downsides to the manner in which the projection and quantisation steps are typically performed. Locality Sensitive Hashing (LSH), one of the arguably most well-known and widely applied methods for hashing-based ANN search, sets the hashing hyperplanes and the quantisation thresholds in a manner that is *independent* of the distribution of the data. For example, in the variant of LSH for preserving the cosine similarity, the normal vectors of the hashing hyperplanes are randomly sampled from a zero mean unit variance multidimensional Gaussian distribution. This data-oblivious mechanism for generating the hashing hypersurfaces runs a high risk of separating dense areas of the feature space and therefore partitioning related data-points into different hashtable buckets (e.g. points a and b in Figure 1.3). To ameliorate this low recall problem a typical LSH deployment involves partitioning the data with multiple independent hashtables and presenting the union of all the data-points in the colliding hashtable buckets as candidate nearest neighbours. Unfortunately, the greater the number of hashtables the higher the memory requirements needed for an LSH deployment. The quantisation thresholds are also set in a data-independent manner, typically by thresholding at zero along a projected dimension. In this context a projected dimension is formed from collating the projections from all data-points onto the normal vector to a hyperplane. Unfortunately, the region around zero on a projected dimension is usually the area of highest point density which means that there is a high chance of related data-points falling on opposite sides of the threshold and therefore being assigned different bits.

There is clearly a wide scope for improving the retrieval effectiveness of LSH and many other influential but data-oblivious algorithms for hashing-based approximate nearest neighbour search by tackling both of these issues head on: it is this task that forms the focus of this thesis. Specifically I am interested in maximising the *neighbourhood preservation* of both of these steps in the pipeline, that is the preservation of

[5] I 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.

the relative distances in the input feature space in the corresponding Hamming space, as this will directly translate into compact binary codes that are more similar for similar data points, yielding a corresponding increase in retrieval effectiveness. Furthermore, there is the added demand that this criterion be met with the shortest possible length of hashcode to conserve storage space and computation time.

1.2 Thesis Contributions

In this thesis I undertake a detailed study of the projection and quantisation operations as they relate to hashing-based ANN search. As identified in Section 1.1, both steps are crucial components in the process which many existing hashing-based ANN search models use to generate similarity preserving hashcodes for data-points. The central hypothesis examined in this thesis can be compactly stated as follows:

The retrieval effectiveness of existing hashing-based ANN search methods can be significantly improved by learning the quantisation thresholds and hashing hyperplanes in a manner that is directly influenced by the distribution of the data.

This thesis statement forms the backbone of the dissertation and all contributions and experiments are focused on gathering evidence to demonstrate its validity. Recall from Section 1.1 that a popular algorithm for solving the ANN search problem, Locality Sensitive Hashing (LSH), firstly picks a *random* dimension in the input feature space and then projects a data-point onto this dimension. This projection step is then followed by a quantisation operation that converts the projection into binary (0 or 1) by *thresholding at zero* with a *single threshold*. The projection and quantisation operations are repeated K times to build up a K-bit hashcode for a given data-point. This hashcode generation pipeline effectively relies on the following three underlying assumptions[6]:

- A_1: Single static threshold placed at zero (for mean centered data)
- A_2: Uniform allocation of thresholds across each dimension

[6]These three specific assumptions were chosen as they are the simplest assumptions that are frequently made in the literature, and whose relaxation I thought would result in the largest gain in retrieval effectiveness. This list is not exhaustive and other limiting assumptions exist such as learning the hyperplanes independently of each other. Examination of these assumptions is left to future work (Chapter 8).

- A_3: Linear hypersurfaces (hyperplanes) positioned randomly

These three assumptions permeate the learning to hash literature and can be found, in some form or another, in most existing work in the field (Indyk and Motwani (1998); Weiss et al. (2008); Liu et al. (2012); Gong and Lazebnik (2011); Raginsky and Lazebnik (2009); Kulis and Darrell (2009)). In this dissertation, for each identified assumption, I introduce a novel data-driven algorithm that relaxes that assumption. In each case I evaluate the proposed algorithm with respect to state-of-the-art baselines on standard image retrieval datasets and demonstrate statistically significant increases in retrieval effectiveness. There are three advantages to the algorithms presented in this thesis: firstly, they can be used to improve the retrieval effectiveness of almost *any* existing model for hashing-based ANN search not just LSH. Secondly a simple extension permits cross-domain applicability, such as retrieving images using a textual query. Thirdly, as a further contribution, I will show in Chapter 7 that the algorithms can be *combined* in a synergistic manner to increase the retrieval effectiveness over what is possible using either algorithm in isolation, at the expense of additional training time.

The specific contributions of this thesis to relaxing these three assumptions are:

A_1) **Single static threshold placed at zero (for mean centered data)**: I address assumption A_1 by formulating a new quantisation algorithm that assigns *multiple thresholds* for each projected dimension, rather than a single threshold. I show that retrieval effectiveness depends heavily on the correct positioning of the threshold(s) along a projected dimension and that a static positioning is sub-optimal. To learn the optimal threshold positions I introduce a novel semi-supervised clustering algorithm that directly optimises an F_1-measure based objective function computed from a data-point adjacency matrix. This objective function seeks to maximise the number of true nearest neighbours assigned identical bits while minimising the number of unrelated data-points receiving the same bits. Under the same bit budget constraint I demonstrate an improved retrieval effectiveness from a multi-threshold quantisation versus a single threshold quantisation. This work is presented in Chapter 4.

A_2) **Uniform allocation of thresholds across each dimension**: It is usually the case that a collection of hyperplanes are not equally informative about the structure of the input feature space. For example, hyperplanes generated by solving an eigenvalue problem tend to capture the majority of the variance of the data in a small subset of the eigenvectors with the largest associated eigenvalues (Gong and Lazebnik (2011)). This

means that the lower-variance hyperplanes are unreliable and typically do not capture any meaningful structure in the input space. Intuitively we would like to maximally exploit the additional structure captured by the more informative hyperplanes so as to generate more discriminative hashcodes. I show that this is possible by learning a *variable threshold allocation* across hyperplanes. I relax assumption A_2 and allocate more thresholds (bits) from a fixed threshold budget to the more informative hyperplanes and less thresholds (bits) to the less informative hyperplanes. I propose two novel algorithms for computing a non-uniform allocation of thresholds across projected dimensions. For certain classes of projection functions both algorithms demonstrate significantly improved effectiveness over a uniform threshold allocation. This work is presented in Chapter 5.

A_3) **Linear hypersurfaces placed at random in the feature space**: I introduce a novel three-step iterative hashing algorithm that *learns linear or non-linear hypersurfaces* based on the distribution of the data. Hashcodes initialised from an existing hashing scheme such as LSH, are regularised in step A over an adjacency matrix derived from the training dataset. This step has the effect of setting the hashcode for a given data-point to be the average of the hashcodes of its nearest neighbours (as specified by the adjacency matrix). In the second step (B) a set of binary classifiers are learnt to predict the regularised hashcodes with maximum margin. The regularised hashcodes are then re-labelled using the learned classifiers in Step C and these re-labelled bits are then fed into Step A, with the three steps repeating for a fixed number of iterations. I report significant increases in retrieval effectiveness for hashing with hyperplanes that are specifically adapted to the distribution of the data. I obtain a further boost in retrieval effectiveness through learning non-linear hypersurfaces induced by a radial-basis function (RBF) kernel. This work is presented in Chapter 6.

This hypersurface learning algorithm is then extended to the *cross-modal* hashing scenario in which the query and database points are now in *two* different feature spaces (e.g. text and image descriptors). This extension requires a straightforward change to the unimodal hypersurface learning algorithm: rather than learning K hypersurfaces in a single feature space, I now learn $2K$ hypersurfaces, one set of K hypersurfaces in each of the two feature spaces. Within the textual modality the algorithm is identical to the unimodal model: the annotation hashcode bits are regularised over the data affinity graph and K hypersurfaces are learnt using the textual features and the textual hashcode bits as labels. To form the cross-modal bridge I simply use the regularised

hashcode bits in the textual modality as the labels to position K hypersurfaces in the visual feature space. This latter step encourages the hypersurfaces in the two modalities to form buckets that are consistent in both feature spaces. Experimental evaluation of the multimodal hashing scheme demonstrates state-of-the-art cross-modal retrieval effectiveness in comparison to strong multimodal hashing baselines from the literature.

Finally I bring together all the contributions of this thesis by learning both the hashing hyperplanes *and* quantisation thresholds together in the same model, overcoming the main limitation of previous work where either the projection function or quantisation function, or both, remain uninformed by the data distribution. To achieve this learning objective I combine the multi-threshold quantisation algorithms introduced as a means of relaxing assumptions A_1-A_2 with the projection function introduced to relax assumption A_3. The result is a new hashing model for ANN search that is fully flexible, adapting the positioning of the hyperplanes and the quantisation thresholds to the statistics of a given dataset. In the experimental evaluation it is shown conclusively that this combination of models exhibits a retrieval effectiveness greater than using either component in isolation. This result neatly unifies the main contributions of this thesis while also revealing a potentially fruitful new research direction in which the entire hashing model becomes data-adaptive. This work is presented in Chapter 7.

1.3 Thesis Outline

The remainder of this document is structured as follows:

Chapter 2: Background, provides background on the problem of hashing-based ANN search and a review of existing relevant research, placing my contributions in the context of prior-art. The learning to hash research field is at a point where a consolidated review of previous existing work is required. I therefore contribute one of the first thorough reviews of the field encompassing the important functions of quantisation and projection.

Chapter 3: Experimental Methodology, introduces the standard datasets, evaluation paradigms and evaluation metrics used in the literature. I also identify and suggest corrections to certain flaws in the way existing work is evaluated.

Chapter 4: Learning Multiple Quantisation Thresholds, outlines my quantisation

algorithm for positioning multiple thresholds along each projected dimension. I show how the model optimises threshold positions based on maximisation of an F_β-measure objective computed on a data-point adjacency matrix. This objective function encourages a positioning of the thresholds so that more related data-points fall within the same quantised regions and thereby are assigned similar bits. I describe how a brute-force optimisation of the threshold positions is computationally intractable and introduce an efficient stochastic search algorithm that rapidly finds a good local optima in the F_β-measure objective function.

Chapter 5: Learning Variable Quantisation Thresholds, relaxes the uniform threshold allocation assumption introduced by the quantisation model presented in Chapter 4. Specifically I examine the benefits of *varying* the number of thresholds allocated per projected dimension. The two proposed adaptive threshold learning algorithms are found to be more effective for image retrieval than the model presented in Chapter 4.

Chapter 6: Learning the Hashing Hypersurfaces, departs from Chapters 4-5 and focuses on the complementary problem of data-dependent projection function learning. I show how this problem reduces to the optimisation of a set of hypersurfaces in the input feature space guided by must-link and cannot-link constraints on the data-points pairs. I introduce a novel three-step iterative algorithm for learning hashing hyperplanes and show that it can be readily extended to learn non-linear hypersurfaces induced by the radial basis function kernel. I discuss how the model exhibits a number of considerable advantages over previous work, most notably the absence of a computationally expensive matrix factorisation. I then further extend the model to tackle the task of cross-modal retrieval where the query and database data-points are in two different feature spaces (for example, image and textual descriptors).

Chapter 7: Learning Hypersurfaces and Quantisation Thresholds, provides a preliminary exploration into the effects of combining multiple complementary relaxations in the same hashing model. The multi-threshold optimisation algorithms introduced in Chapters 4-5 are used to quantise the projections resulting from the hypersurface learning algorithm introduced in Chapter 6. This chapter unifies my main contributions to hypersurface and quantisation threshold learning in a single model for hashing-based ANN search. I show the important result that relaxing multiple assumptions as part of the same model can have an additive benefit on retrieval effectiveness.

Chapter 8: Conclusions and Future Work, summarises the main contributions in

this thesis and reviews the central claim set out in Chapter 1 in the context of the results presented in Chapters 4-7. Potential fruitful avenues for future research are also proposed.

1.4 Published Work

Chapter 4 expands on the work previously published in SIGIR'13 by providing more detail on the stochastic search algorithms used for threshold learning and giving additional experimental results and analysis.

- Moran, S. and Lavrenko, V. and Osborne, M. (2013). Neighbourhood Preserving Quantisation for LSH. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. Dublin, Ireland.

Chapter 5 provides more detail on the work previously published in ACL'13. Specifically, I introduce a second greedy algorithm for solving the variable threshold allocation problem and provide an expanded set of experiments.

- Moran, S. and Lavrenko, V. and Osborne, M. (2013). Variable Bit Quantisation for LSH. In *Association for Computational Linguistics (ACL)*. Sofia, Bulgaria.

The unimodal part of Chapter 6 was previously published as follows:

- Moran, S. and Lavrenko, V. (2015). Graph Regularised Hashing. In *European Conference on Information Retrieval (ECIR)*. Vienna, Austria.

The cross-modal part of Chapter 6 was previously published in SIGIR'15:

- Moran, S. and Lavrenko (2015). Regularised Cross Modal Hashing. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. Santiago, Chile.

Chapter 7 was previously published in SIGIR'16:

- Moran, S. (2016). Learning to Project and Binarise for Hashing Based Approximate Nearest Neighbour Search. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. Pisa, Italy.