UNIVERSITY of CALIFORNIA
Santa Barbara

Fusion Coding, Search and Retrieval of Correlated Data

A Dissertation submitted in partial satisfaction of the
requirements for the degree

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Sharadh Ramaswamy

Committee in charge:

Professor Kenneth Rose, Chair
Professor B. S. Manjunath
Professor Jerry Gibson
Professor Amr El Abbadi

March 2009

The dissertation of Sharadh Ramaswamy is approved.

Professor B. S. Manjunath

Professor Jerry Gibson

Professor Amr El Abbadi

Professor Kenneth Rose, Committee Chair

December 2008

Fusion Coding, Search and Retrieval of Correlated Data

by

Sharadh Ramaswamy

To my parents,

M.N. Ramaswamy and Vasantha Ramaswamy,

who sacrificed their dreams

so that we could have better lives.

Acknowledgements

This thesis would not have been possible but for the constant encouragement and guidance provided by my adviser, Prof. Kenneth Rose. Under him, I had the opportunity to develop and trust my intuition. What I most appreciated in his guidance was that he allowed me the freedom to tackle the set of problems in my own fashion. When I made mistakes, he corrected me, and when things clicked, he shared my joy of discovery. A Ph.D. in a great school like UCSB is an expensive process and therefore, I am grateful for the financial support from NSF, under grants IIS-0329267 and CCF-0728986, the University of California MICRO program, Applied Signal Technology Inc., CISCO Systems Inc., Dolby Laboratories Inc., and Qualcomm Inc.

Santa Barbara is a small town but I hung out with great people. I am very thankful for having worked with Jayanth Nayak, who was practically a second adviser to me. Life in the Signal Compression Lab would have been boring but for the companionship provided by Ankur, Christian, Japsreet, Vinay, Emre, Emrah, Sang, Pakpoom, Jaewoo and Hua. A great set of seniors were there to show me the ropes when I landed in Santa Barbara. This includes Sumit Paliwal, Vinay Dwivedi, Navneet Panda and Nisheet Shrivastava. Over five years, I have had several fun, wonderful and understanding roommates, who have stood by me during all my fortunes and misfortunes. For that I am grateful to Balki, Nilesh Modi, Karthik Perumalsamy, Vivek Padi, Ashish Sharma and Terry Vy Ly for their friendship and camaraderie. Given that many of us are on a quest for scholarship and fame, in a land far away from home, I realize how much of a difference a supportive atmosphere, the quick joke or the pat on the back, can make. In the five years I have been here, I often reminisce over my time spent with Susmit Biswas, Ajay Raman, Ramya Ragahvendra, Shilpa Kolhar, Roopa Kannan-Iyer, Anindya Sarkar, Vikram Siddavaram, Abhyudai Singh, Rajesh Kumar, Mohit Tiwari, Poornima Kolhar, Gaurav Soni, Sreekar, Chandrasekhar, Naveen Shrivastava and several fun, quirky fellow “internationals” including Yuri Malikov, Nick Nasella, “Jose”, Anup

Hosangadi, Massimo, Marc Pidoux, Anurag Tyagi, Sheena Menezes, David Rosenberg, Evelyn Wade and other TAs in the German department. And finally, I had friends from my IIT days - Ajit "Q" Narayanan and Swaroop "Noddy" Venkatesh - whom I could share with my experiences at UCSB and depend on for honest opinions and advice. Thank you all, for all the coffee and chai, fun and laughter.

When research was "slow", or in between paper submissions, I had the opportunity to try my hand at music. The hours spent jamming on my guitar or singing with Mohit, Susmit, Shivakumar Vasanth and Shaunak Bopardikar were a refreshing and stimulating break from the rigors of a Ph.D. By the side, I learnt a bit about acting from Amit Pathak and realized how useful this could be in real life,as well! Performing on stage was a different challenge, and without these people, I would not have had my "fifteen minutes of fame". It is in this regard I came to know of the superb organization skills of the office bearers of the IASB - Sumit Singh, Uttam Singisetty, Banit Agrawal, Prashant Acharya, Ketan Savla, Sriram Venkateswaran, Karthik Jayaraman, Seshadri and others.

Among my non-academic pursuits while at UCSB, I take pride in having organized Indian classical music concerts for Raagmala and later, charity fund-raisers for ASHA . During these events, I met a bunch of hard-working, committed and inspirational individuals. I recall all the effort and support put in by Susmit, Ashwin and myself, in reinvigorating Raagmala. In addition to Susmit and Ashwin, it was great fun working alongside people like Swaroop Jagadish, Sunita, Karthik Jayaraman, Sansaptak, Nidhi, Ramya and the ASHA liaisons, Anand Ramaswamy, Seshadhri Kiran Kolluri and Nagender Bandi. I also had the time to try and develop new hobbies including surfing with Jagdish Balam and Kaviyesh Doshi, RJing on KCSB with Ashwin, Gargi Shah, Niranjan Shetty and Amit Jardosh and hiking with Balki, Esteban Gonzalez, Vineet Birman, Brendon Hall, Prof. Meiburg in the Grand Canyon and Zion Canyon. Hiking with Amitabh and later, in the Himalayas, with Vivek Padi and Rajkumar Shilwal was equally fun. So were all the trysts in Sipsey, Seven Falls, and Big Sur with my undergrad

buddies Ajit “Q” Narayanan and Swaroop “Noddy” Venkatesh.

I am grateful to both the Kumar families for their hospitality. Your warm welcome certainly went a long way in combatting home-sickness and I shan’t forget this. Nor can I forget the timely advice and concern shown by Kalpana and the constant support from my Sunila Athe. It was the dream of my father and mother, Madurai Narayanan Ramaswamy and Vasantha Ramaswamy, that I attain high scholarship. They have sacrificed a lot of their own time, energy and happiness, for me to get here. I owe my success to my parents and I hope I can continue to make them proud in the years to come.

Curriculum Vitæ

Sharadh Ramaswamy

Education

2003	Bachelor of Technology (Electrical Engineering), Indian Institute of Technology, Madras.
2003	Master of Technology (Communications Systems), Indian Institute of Technology, Madras.

Research Experience

2004–2008	Graduate Student Researcher, Signal Compression Lab, University of California, Santa Barbara.

Teaching Experience

2003–2004	Teaching Assistant, University of California, Santa Barbara.

Professional Experience

Summer 2006	Summer Intern, Qualcomm Inc., Campbell, CA, USA
Summer 2007	Summer Intern, Invention Labs Engineering Products Pvt. Ltd., Chennai, India

Selected Publications

- J. Nayak, S. Ramaswamy and K. Rose. "Correlated Source Coding for Fusion Storage and Selective Retrieval of Correlated Sources". In *IEEE International Symposium on Information Theory*. pp. 5403-5408, 2005.

- S. Ramaswamy, J. Nayak and K. Rose. "Code Design for Fast Selective Retrieval of Fusion Stored Time-Series/Sensor Network Data". In *IEEE International Conference on Acoustics, Speech and Signal Processing.* volume 2, pp. 1005-1008, 2007.
- S. Ramaswamy, and K. Rose. "Adaptive Cluster Distance Bounding for Similarity Search in Image Databases". In *IEEE International Conference on Image Processing.* volume 6, pp. 381-384, 2007.
- S. Ramaswamy, and K. Rose. "Shared Description Fusion Coding for Storage and Selective Retrieval of Correlated Sources". In *the IEEE Data Compression Conference.* pp. 262-271, 2008.
- S. Ramaswamy, and K. Rose. "Predictive Fusion Coding of Spatio-temporally Correlated Sources". In *IEEE International Conference on Acoustics, Speech and Signal Processing.* pp. 2509-2512, 2008.
- S. Ramaswamy, and K. Rose. "Fast, Adaptive Mahalanobis Distance-based Search in Image Databases". In *IEEE International Conference on Image Processing*, 2008.
- S. Ramaswamy, J. Nayak and K. Rose. "Fusion Coding of Correlated Sources for Storage and Selective Retrieval". to be submitted to *IEEE Transactions on Signal Processing.*
- S. Ramaswamy and K. Rose. "Adaptive Cluster Distance Bounding for High-Dimensional Indexing". to be submitted to *IEEE Transactions on Knowledge and Data Engineering.*
- S. Ramaswamy and K. Rose. "Towards Optimal Indexing for Relevance Feedback in Image Databases". to be submitted to *IEEE Transactions on Image Processing.*
- S. Ramaswamy and K. Rose. "Shared Descriptions Fusion Coding of Correlated Sources for Storage and Selective Retrieval". to be submitted to *IEEE Transactions on Signal Processing.*

- S. Ramaswamy and K. Rose. "Predictive Fusion Coding of Correlated Sources". to be submitted to *IEEE Transactions on Signal Processing*.

Abstract

Fusion Coding, Search and Retrieval of Correlated Data

by

Sharadh Ramaswamy

This work is a collection of results on novel uses of quantizers for two problems in database management, namely, similarity search in high-dimensional databases, and fusion-compression for selective retrieval of correlated data sources (data-streams). The results can be grouped into three main categories, each of which occupies a paragraph below.

Similarity search for nearest neighbors in databases where entries are represented by high-dimensional, correlated features, is challenging and compression is a reliable means to speed up search time. It is shown how to efficiently use the vector quantization scheme for exact similarity search. A new cluster distance bounding technique, based on separating hyperplanes and cluster supports, is presented and shown to achieve exact similarity search while requiring the retrieval of only a few relevant clusters. Next, a property of point-to-hyperplane distance is derived that enables extending the query-cluster distance bound to an adaptively changing Mahalanobis distance matrix. The resulting cluster indexing efficiently adapts to relevance feedback from the user. Finally, these bounds are further tightened by recasting the problem of cluster-distance bounding as norm minimization with linear constraints. The resulting convex optimization problem complexity is polynomial in the number of dimensions.

The second focus is on fusion compression for selective retrieval of data from correlated source (sensor) networks. A fusion coding paradigm is presented where, given a fixed storage rate (space), the retrieval rate (time) is traded against distortion (qual-

ity). Only statistics of future queries are assumed to be apriori known. The optimality properties of the fusion coder are derived, and an iterative algorithm for design is presented. Lastly, these properties are exploited to reduce growth of design complexity with query-set size and designs are adapted to queries unknown during design phase.

The proposed fusion coder's design complexity scales exponentially with storage rate and number of blocks. Two techniques are presented to handle the growth in design complexity. In the first, by imposing "clever" constraints on fusion coder modules, a "Shared Descriptions" framework is obtained wherein, it is possible to trade storage rate, retrieval rate, distortion and complexity of design against each other. This allows design at high storage rates and provides gains over naive quantization schemes that have not been optimized for fusion coding. On the other hand, time correlations are exploited by a predictive coder, which is a low-complexity alternative to fusion coding over long blocks. The algorithm for predictive fusion coder design is based on the asymptotic closed loop principle and thereby, stable. The resulting predictive coder is matched to the closed-loop error statistics and provides substantial gains over memoryless fusion coding and joint compression schemes.

Contents

List of Figures

List of Acronyms

ACL	: Asymptotic closed loop
CAD	: Computer aided design
CAM	: Computer aided manufacturing
CD	: Compact disc
DA	: Deterministic annealing
dB	: Decibel
DFT	: Discrete Fourier transform
DVD	: Digital video/versatile disc
FC	: Fusion coder
GLA	: Generalized Lloyd algorithm
IO	: Input-output operation
kB	: Kilobyte
KLT	: Karhunen-Loève Transform
kNN	: k nearest neighbor
LDC	: Local Digital Coding
MBR	: Minimum Bounding Rectangle
MBS	: Minimum Bounding Sphere
MSE	: Mean squared error
NN	: Nearest neighbor
PFC	: Predictive fusion coder
PVQ	: Predictive vector quantizer
RAM	: Random access memory
SDFC	: Shared descriptions fusion coder
SNR	: Signal to noise ratio
VA	: Vector approximation
VQ	: Vector quantizer

Chapter 1

Introduction

The development of high-speed electronics has resulted in proliferation of multimedia devices such as CD players, MP3 players, digital cameras, imaging devices, which generate huge volumes of multimedia data. At the same time, there has been a dense deployment of low-cost sensing devices to monitor the environment, such as temperature sensors, where due to their geographical proximity, there are dependencies in the observations of the sensors. Combined with the availability of cheap storage, this has led vast data archives as well as several applications to mine these data. Data mining in such repositories requires the extraction of rich, high-dimensional, correlated features. This thesis focusses on how these correlations might be exploited through source coding to minimize required storage and to accelerate retrieval in databases.

We are concerned with two important database applications. The first deals with efficient nearest neighbor search over databases of high-dimensional features extracted from images etc. The other is a fundamental problem in efficient storage and fast retrieval of correlated data-streams whose solution is critical to the usefulness and practicality of sensor network databases. Compression has so far been used only to reduce storage space/bandwidth requirements. But in this thesis, we show the extensive applications of compression in both the above problems.

The work can be grouped into three categories, each of which is described below. In Section 1.1, we outline the compression approach to similarity search, including extensions to incorporate user feedback and retrieve perceptually relevant results. In section 1.2, we describe the problem of fusion coding of correlated sources, as well as a framework for optimizing the storage rate-retrieval rate-distortion tradeoff. In Section 1.3, we consider the scalability of optimal fusion coding systems and note the exponential growth in complexity. By revising the method to impose constraints on complexity, we achieve scalable design algorithms that handle large sensor networks, and signal sources with memory.

1.1 Indexing for Exact k-Nearest Neighbor (kNN) Search

In data mining, the data objects are abstracted by a set of representative features. Given the complexity of human perception, a rich set of feature vectors needs to be extracted from multimedia data for mining. The high-dimensionality of the features and the large databases handled necessitate offline storage, on a hard disk device. Since hard disk devices are slower than computing devices, the disk access times dominate query-search times. However, the search for nearest neighbors in such high-dimensional databases is challenging due to the "curse of dimensionality". Several, state-of-the-art indexing schemes rely on some form of compression to speed up search time. The well known Vector Approximation-File (VA-File) performs scalar quantization of the data-set, but ignores feature correlations. Vector quantization (VQ) on the other hand can exploit correlations and dependencies across feature dimensions and thereby produce a compact representation of the data-set. While the use of clustering/VQ for fast, approximate search is well known, we now show how to efficiently use the same VQ scheme for exact search.

Central to our indexing approach, is the need to estimate query-cluster distance bounds. Conventional distance bounds based on Minimum Bounding Rectangles (MBRs) and Minimum Bounding Spheres (MBSs) are loose. This results in weak spatial filtering and restricts applicability of the VQ scheme to approximate similarity search. We present a new cluster distance bounding technique, where we project the query onto separating hyperplanes and complement with cluster-hyperplane distance (cluster support). This "hyperplane bound" produces tight, lower bounds on query-cluster distances, resulting in retrieval of only a few relevant clusters. We note that Voronoi cluster boundaries are linear hyperplanes and it is these that are used in cluster-distance bounding. We also present a reduced storage complexity hyperplane bound, that is almost equally effective. Substantial gains in IO performance, computations and preprocessing storage over known indexes are noticed on several large data-sets.

While indexing multimedia data, it is equally important to retrieve perceptually relevant results. In order to do so, a popular approach is to use relevance/user feedback to update the distance measure, typically the weight matrix of a Mahalanobis distance measure. However, this complicates indexing, as usually knowledge of the weight matrix is needed for index construction. Consequently, when relevance feedback updates the weight matrix, several indexing schemes are rendered ineffective. We derive an invariance property of point-to-hyperplane distance that allows recalculation of cluster supports as the weight matrix changes, without accessing the cluster. This allows for tight bounding of query-cluster distance using the proposed hyperplane bound, under a changing Mahalanobis distance matrix. Thus, cluster indexing for relevance feedback is possible.

Finally, we improve upon the hyperplane bound by recasting cluster-distance estimation as norm minimization under linear constraints, imposed by separating hyperplane boundaries. This is equivalent to bounding the cluster with a convex polytope and estimating the distance from the query to the polytope. In this fashion, given multiple separating hyperplanes, we search over all hyperplanes and all points of intersection.

We note that, given K hyperplanes in a d dimensional space, the number of points of intersection are $O(K^d)$. This is a consequence of "curse of dimensionality", which incapacitates naive search over all possible intersections. However, since norms are convex functions, the distance estimation is a convex optimization problem and can be solved in time polynomial in the number of dimensions. We note that Mahalanobis distances are also norms and hence, such cluster distance bounding technique adapts to changing Mahalanobis distances as well. The reader is directed to Chapter 3 for details and results from experiments on several real, high-dimensional data-sets.

1.2 Fusion Coding of Correlated Sources

Our second focus is on a new application of source coding in the compression of data collected from correlated source (sensor) networks, which we term as the *fusion coding of correlated sources*. Given huge volumes of data from multiple sources (or data-streams), it is necessary to compress and save storage space. Yet, at the same time, only a select a-priori unknown subset of sources could be queried. Such database design introduces fundamentally new and interesting challenges: On the one hand, inter-source correlations may be exploited via joint coding to reduce the overall storage requirement and to potentially reduce the retrieval time. On the other hand, a future query may select only few of the sources for retrieval, and it would be wasteful to have to retrieve the entire (jointly) compressed data only to reconstruct a small subset.

Thus, at the heart of the problem, lies a trade-off between storage rate (space) versus retrieval rate (time), (both measured in terms of bits stored or retrieved). An example application of the proposed fusion coding of correlated sources is in the arena of sensor networks, which has been the focus of extensive research in recent years. Much of the effort in sensor network design has been dedicated to the development of device and communication technologies. But in order to fully realize the potential of most such systems, it is necessary to efficiently store the vast volumes of data generated by the

network for future retrieval, as needed for analysis or other uses.

We assume that only statistics of future queries is known and are interested in designs for lossy compression. In Chapter 4, we present a Fusion Coder (FC) for the joint compression and selective retrieval, or *fusion coding*, of correlated sources, where given a fixed storage rate (space), the retrieval rate(time) is traded against distortion (quality). The fusion coder is composed of three modules: an encoder, a bit-selector and a decoder. The crucial components is the bit-selector which allows retrieval of only subsets of stored bits for each query. We derive optimality properties of each module and present an iterative algorithm for FC design. We study the design complexity of FC and note growth with query-set size. We exploit the properties of the optimal FC to reduce complexity of design. Lastly, we present techniques to adapt FC designs to queries unknown during design phase.

1.3 Efficient Design of Fusion Coders

By increasing the storage rate, better trade-offs between retrieval rate and distortion are possible. Secondly, in order to exploit temporal correlations in data, coding over larger block lengths would be necessary. However, the proposed FC's design, storage and operational complexity scales exponentially with storage rate and number of blocks coded. We consider two approaches to handle the growth in complexity. In the first scenario, by imposing "clever" constraints on FC modules, we obtain a framework wherein it is possible to trade storage rate, retrieval rate, distortion and complexity of design. Specifically, the bit-selector is constrained to retrieve only from disjoints groups of bits. Consequently, the encoding of such bits can be independently performed, which in turn reduces storage complexity. Simultaneously, an estimate of complexity becomes available. Each such group of bits is termed a "Shared Description" and the corresponding coder is termed the Shared Descriptions Fusion Coder (SDFC). The proposed SDFC allows design at high storage rates and provides gains over naive quantization schemes

that have not been optimized for fusion coding. For details, the reader is directed to Chapter 5.

On the other hand, we propose to exploit time correlations by a Predictive Fusion Coder (PFC). However, PFC design is complicated, not only by the feedback loop, but also by the need to account for an (exponentially) large set of queries at the encoder. The optimal encoder would have as many prediction loops as there are queries and could be of impractical complexity (even for moderate sized networks). We impose complexity constraints on the encoder where queries and sources share prediction loops. We derive an iterative algorithm for predictive fusion coder design, which is based on the "Asymptotic Closed Loop" framework and hence, circumvents optimality and stability issues that plague traditional predictive quantizer design. The proposed PFCs directly optimize the distortion-retrieval rate tradeoff, given a fixed storage capacity, and hence, provide significant gains over storage schemes that perform only joint compression or memoryless fusion coding of all sources.

Prior to discussion of our methods and approaches to organizing such correlated databases, we present background, discuss other prior, relevant works and develop some of our notations in Chapter 2. A summary of the main contributions of this dissertation and some directions for future research are presented in Chapter 6.

Chapter 2

Background

2.1 Multimedia Database Management

Advancements in magnetic storage and semiconductor technology has led to a proliferation of personal digital multi-media devices, such as digital cameras, video recorders etc. and archival of multimedia data. This, in turn, has spawned new database applications that handle these data, such as image search engines [1], digital libraries (such as the Alexandria Digital Library project [2]), personal digital albums [3], Multimedia Information Systems (see [4][5] [6] and more recently, the Photo-tourism project [7]), Computer Aided Design/Manufacturing (CAD/CAM), Geographical Information systems (GIS) [8], medical image mining [9][10], social networking [11] etc. However, the nature of these databases is such that their organization based on the well known relational database model is insufficient. For example, while searches based on keywords is the current paradigm in many search engines, keywords are not necessarily the most effective representatives of multimedia information. It would be ineffective to mine databases of medical images based on keywords or "metadata" if the goal is to discover hidden correlations that are unknown and hence have not been quantified

through metadata. Clearly, content-based database management is a more appropriate paradigm.

In the last decade or so, we have moved towards an "Information Age", where we handle more complicated tasks for which known analytical methods/models are poor fits. It has become inevitable to learn the models and predict from data i.e. to perform "machine learning". For example, spam is of such a nature that only machine learning approaches such as Bayesian prediction [12] are appropriate for spam filtering. However, the impact of such tools necessitates large training sets and hence the need to collect and archive data. Another example is efficient bug detection in large program code i.e. automated software support (see [13]). Yet another example is extensive weather sensing and archiving for building better climate models. In some cases, the statistics and nature of data change with time. For example, spammers respond to Bayesian filtering by changing the structure of their emails [14] and hence, even spam filters would need to evolve. Another case would be mathematical modelling and algorithmic trading strategies in finance. As better models for financial data are derived, better trading strategies are devised. However, the market "evolves" and changes to these approaches, which necessitates continued study and improved modelling. Even then, all new models are evaluated against (archived) historical data, before acceptance.

Thus, there is a strong rationale for archiving and mining multimedia data. The variety of data that need to be collected, handled and mined is huge, varying from text data to streaming time-series data; geospatial data to source/executable program data (in code repositories) and multimedia data. Data organization for content-base retrieval/use would involve feature extraction, clustering, indexing, tagging, filtering (e.g. spam/ham filtering), pattern recognition, mining etc. Therefore, there is need for efficient multimedia database management.

2.2 Storage Technology

Historically, storage technology has always lagged processor technology in speed and available storage has almost always lagged applications i.e. as with case of computational units, applications outstrip available storage. In all storage media, data is stored and accessed in units of *pages*. A page is a block of bytes and modern hard disks have a page size of around 4kB. Typically, there is a limit on the page size. This is necessary to reduce disk fragmentation and to reduce susceptibility to power failure (on hard disks). However, page sizes themselves are application/technology dependent [15]. With the exception of tape drives, on most storage media, data can be accessed by serial access (or serial IO) or by a random access. This includes hard disks, random access memory and compact discs (optical) media. Depending upon the kind of storage technology, the costs of sequential and random page IOs vary.

2.2.1 Hard Disk Technology

Hard-disk technology consists of storage on cylindrical disks where a head writes (creates) or retrieves a magnetic polarization pattern. The magnetic medium can be easily erased and rewritten, and it will "remember" the magnetic flux patterns stored onto the medium for many years. Combined with a very low price/density ratio, this makes the hard disk the preferred storage medium. The magnetic read/write technology is similar to how storage on tapes and tape drives takes place but retrieval on hard disks can be through random IOs as well. The disks rotate at high speed (currently reaching 20,000 rpm), while the head can move across the surface of the disk. Some hard disks have multiple read/write heads.

The time taken to retrieve data from the disk is a sum of *seek time*, *rotational latency* and *transfer time*. Seek-time is the time taken to position the disk head over the track and varies from 0-20msec. Rotational time is the time taken for the corresponding block

to rotate under the head and takes about 0-10msec. Transfer time is the time taken to actually move data to/from the disk surface and is about 1msec per 4KB page. Thus, we can see that seek times dominate access times. Hence, random IOs are more expensive on hard disks. On modern disks, the page access time for a single random IO is 100 times that for a single serial page access. Random IOs are better in retrieving pages spaced far apart, while sequential IOs are better for reading closely spaced pages. Each new disk access would involve one random IO and if more data need to be retrieved, a few additional sequential IOs[1]. If all the pages to be accessed are known before hand, the optimal sequence of sequential and random IOs is determined as the shortest path over a suitably defined graph [16].

2.2.2 Semiconductor Memory

Semiconductor or random access memory (RAM) have faster access times than magnetic media. The access times on RAM are limited by electronics (or corresponding quantum mechanical limits), unlike hard disks, where access speed is limited by mechanical components. Sequential and random page IOs have roughly the same cost. Historically, RAM has been more expensive and only possible for short term storage. But recent technology developments have resulted in better price-performance ratios and allowed long term retention of stored data. It is expected that in the near future, RAM would replace hard disks, at least, in personal computing devices.

2.2.3 Other Storage Media

Optical (such as CD/DVD) disks, unlike hard disks and RAM are typically "write once, read many" devices, which restricts their applications, even though they have high density of storage and are durable for several years. Optical media are typically suited

[1]We note that this is a relatively simple model, as we ignore the effects of the cache and other physical hardware processes on disk access. True estimates of access times would depend on these factors as well, but for our purposes, this level of abstraction is sufficient.

for video/audio streaming applications.

2.3 Data Representation and Feature Extraction

Data stored in a database often contain correlations and dependencies. By exploiting correlations, we can compress a database efficiently i.e. have a more compact representation for same quality or vice versa. In other words, a compressed representation allows for faster retrieval. More importantly, this could be used to improve kNN search speed by searching over the compressed version of the database prior to access of the actual database. In this thesis, we deal with signal databases of two types - either static databases of vector data or dynamic databases of streaming data. The former would consist of high-dimensional features extracted from images etc. and maintained for similarity indexing. The latter consist several individual, correlated streams of data (signals) consisting of measurements from sensor networks or time-series such as stock market streams.

Sometimes raw data are used directly as features, for example in a database of employee payrolls or in repositories of sensor data [17]. In the case of signal databases, often, it is more useful to represent the database objects based on features extracted from the data, rather than the raw data. A feature representation would be chosen based on an application or database at hand. For example, Fourier transforms are found to be compact representations of time-series and are often used in feature extraction [18]. A useful property of the Discrete Fourier Transform (DFT) is that moving average (or any linear operation) in time is a scaling of the Fourier feature space (when the DFT is properly evaluated) [19]. Additionally, it has been found to be easier to monitor correlations among streams in the Fourier domain [20]. Wavelet representations have been found to be useful in interpolating missing values [21] and comparable to Fourier representations [22]. Piecewise constant representations [23] have been found to return perceptually relevant results, when used for similarity indexing.

In the case of images, other kinds of features are more useful. Popular descriptors are the color histogram [4], the color layout descriptor (CLD) [24], scale-invariant feature transforms (SIFT) [25], shape descriptors [26] etc. Recently, a combination of texture features (extracted through Gabor filters) and color features (histograms) have been found to be efficient descriptors for a broad class of images and form a part of the MPEG-7 multimedia standard (see [27]). Useful feature vectors are often high dimensional, such as the 60 dimensional texture descriptors of [27]. For video, in addition to color, shape and texture features, motion intensity descriptors are known to be useful [24]. In the case text mining, term frequency-inverse document frequency (tf-idf) features are commonly used [28], while inverted lists [29] are often preferred for text document indexing.

2.3.1 Feature/Data Storage

With large amounts of data being collected and with a rich set of high-dimensional features being extracted, storage on the relatively limited and expensive primary memory (RAM) is impractical. On the other hand, magnetic hardware offer high storage density-cost ratios and preferred media for feature or data storage. However, such secondary storage devices are significantly slower and hence, IO access time dominates query processing time.

2.4 The Concept of a Query

A query is a request for information. However, the nature of the query would depend on the database and application at hand. In this thesis, we restrict attention to two kinds of queries, which are described in the following subsections.

2.4.1 Similarity queries

The first class of queries that shall be considered are similarity queries. Here, the user provides the system with an object, from which a query vector is abstracted. The system is requested to return all objects similar to the query or perform *similarity search*. The search and retrieval is based on the content of the query and objects in the database, i.e. it is a *content-based retrieval*. We note that in the context of similarity queries, the query vectors and feature vectors lie in the same vector space. A common model is the query-by-example where the user selects an object from the database most relevant to the query.

The similarity of two objects is assumed to be proportional to the distance between their feature vectors. A distance function $d(p, q)$ estimates the distance between vectors p and q in the feature space. Commonly used distance functions are the Euclidean (l_2 norm), Mahalanobis distances and l_p norm ($p \geq 1$) (for real vectors), the Edit distance (for strings/text) and the Hamming distance (for finite alphabet vectors). Usually, good distance functions are also metrics [30].

Some popular similarity queries are listed in the following table.

Table 2.1. Types of Similarity Queries

Type of query	Search results in ...
Single Nearest Neighbor (1NN)	The closest (nearest) neighbor to the query
k Nearest Neighbor (kNN)	The 'k' Nearest Neighbors (NN) to the query
range-query or ϵNN	All feature vectors (data elements) within ϵ distance from the query
$(1+\epsilon)$-query	All feature vectors within $(1+\epsilon)$ of the distance to the 1NN
ϵkNN query	k feature vectors with utmost ϵ error in distance from the query

We note that kNN queries are widely applicable to many databases, whereas range queries are more appropriate for geospatial or alphanumeric (such as payroll) data. This is because when dealing with databases of features such as image feature sets, users would

find it difficult to define the radius of search, whereas in geospatial databases, this is quite intuitive. It is for of this reason, in experimental sections, we limit discussion to kNN queries as our similarity queries.

2.4.2 Queries in Correlated Source Databases

We also consider another class of queries. Given a database of sources $X_1, ..., X_M$, this query is a request for a *subset* of these sources. For example, when a database accumulates several stock market streams, a user could request data from only the technology stocks (a subset of these streams).

Employing binary variables $q_i \in \{0, 1\}$ to denote whether source X_i is requested or not, we represent queries by M-tuples of the form

$$\mathbf{q} = (q_1, ..., q_M) \in \mathcal{Q} \tag{2.1}$$

where $\mathcal{Q} \subseteq \{0, 1\}^M$ represents the domain-set of queries. We next introduce notation for the query distribution, or the probability mass function (pmf),

$$P : \mathcal{Q} \to [0, 1] \tag{2.2}$$

It is to be noted that there are conceivably 2^M possible queries and $\sum_{\mathbf{q} \in \mathcal{Q}} P(\mathbf{q}) = 1$. Without loss of generality, we assume that each source is requested with positive probability (i.e., there exists some query with positive probability whose requested subset includes the source) and that a query always asks for a non-empty subset of sources, i.e.,

$$P(\mathbf{0}) = 0 \tag{2.3}$$

Clearly, queries and data lie in different spaces.

2.5 Need For Similarity Indexing

As described before, a host of new database applications such as Multimedia Information Systems, Geographical Information systems (GIS), time-series analysis (in stock markets and sensors) etc., need to be supported, that store large amounts of data for future retrieval (see [10] for more examples and applications). The size of these databases can range from the relatively small (a few 100 GB) to the very large (several 100 TB, or more). A very important application is to find the most similar objects within the database to a query object. When the search is based on the content of the (query and database) objects, the search is termed a similarity search. A typical application is in image databases i.e. to search for the database images most similar to a query image.

Similarity search serves an important role in knowledge discovery and is hence extremely useful. For example, by similarity searches over biomedical images, hidden correlations between different medical conditions might be discovered. In the future, large organizations will have to retrieve and process petabytes of data, for various purposes such as data mining and decision support. Given the volume of data being handled, the only cost effective solution (to date) is storage on secondary hard-disk devices. But since secondary devices are extremely slow, compared to modern processing units such as CPUs, the time spent in data access (IO time) overwhelmingly dominates query processing time. Hence, the need to organize the database for quick similarity searches i.e. efficient similarity indexing.

More generally, indexes perform information filtering, reduce search and access times and are used in several disparate applications. Examples include but are not limited to dictionaries (inherent usage), file systems, payroll indexing [31], Web search engines, scientific document and co-citation indexes such as Google Scholar, CiteSeer, Microsoft Libra (also see [32] [33]), inverted lists [29] and Latent Semantic Indexing (LSI) [34] for text retrieval, Computer Aided Design (CAD) systems, Geographical Information Systems (GIS) systems (e.g. Google maps) etc. In fact, the R-tree [35] was originally

devised for a CAD application (see [36]) i.e. to reduce search and processing time in determining whether an area of the chip was used or not. The first portion of this thesis focuses on multidimensional indexes for similarity searching in real, vector spaces.

2.5.1 Traditional Tree-based Similarity Indexing

Several index structures exist that facilitate search and retrieval of multi-dimensional data. In low dimensional spaces, *recursive* partitioning of the space with hyper-rectangles (R-trees [35], R*-trees [37], k-d trees [38]), hyper-spheres (SS-Tree [39]) or a combination of hyper-spheres and hyper-rectangles (SR-Tree [40]), have been found to be effective for nearest neighbor search and retrieval. While the preceding methods specialize to Euclidean distance (l_2 norm), M-trees [41] have been found to be effective for metric spaces with arbitrary distance functions (which are metrics).

2.5.2 Curse of dimensionality

The phrase '*curse of dimensionality*' actually refers to the exponential growth of hyper-volume with dimensionality of the space and was initially introduced by Bellman [42], in the context of multidimensional optimization. As an example, for the simplest space partitioning scheme (such as described in the [43]), where the data space in each dimension is partitioned into two halves, with d-dimensions, there are 2^d partitions. Also, note that each such d-dimensional partition is a rectangle with 2^d vertices. With d$\leq$10 and with around 10^6 elements in the data space, such a partitioning strategy for database search makes sense. However, if d is larger, say d=100, there are around 10^{30} partitions for only 10^6 points i.e. the overwhelming majority of partitions are empty.

Several multidimensional indexing strategies can be thought of as maps of the data (input) space. Loosely speaking, such maps try to somehow cover or represent every part of the input space. Covering the entire input space takes up a lot of resources,

and, in the most general case, the amount of resources needed is proportional to the hyper-volume of the input space. The resources in this case are the size of the index tree (storage cost) and the number of IO accesses (the retrieval cost). Because of the curse of dimensionality, a very-large portion of the space is actually empty and hence, almost all resources are used on irrelevant/empty portions of the data space. So, similarity search by searching all space partitions, would lead to a large number of needless disk accesses, making it slower than the simple sequential scan.

2.5.3 Failure of Tree-based Indexing

Conventional multi-dimensional indexes such as R-tree [35] work well in low dimensional spaces, where they outperform sequential scan. But it has been observed that the performance of many such multidimensional index structures, degrades with increase in feature dimensions and, after a certain dimension threshold, becomes inferior to sequential scan. In a celebrated result, Weber et. al. [43] have shown that whenever the dimensionality is above 10, these methods are outperformed by simple sequential scan. Such performance degradation is typically attributed only to Bellman's '*curse of dimensionality*' [42]

A more careful analysis of such indexing schemes reveals other reasons for their failure. These are

- **Recursive Partitioning and Fragmentation:** A notable characteristic of such indexes is recursive space partitioning. This is done till each node contains utmost only as many vectors as allowed by the page size and dimensionality. While the goal is to retrieve these pages with a single random IO, we note since page capacity reduces with dimensionality, the "fanout" of these trees reduces with dimensionality and consequently, height of such trees increases. This leads to (exponentially) more storage ($O(a^h)$, for some constant a and tree height h) and unnecessary fragmentation of the data-set. For example, consider total index storage costs reported

in [44], which dominate the size of the original data-set. Similar feature vectors now lie on different pages and need to be accessed by separate random IOs.

- **Weak Adaptation to Data-set Statistics :** Few of these indexes are created by processing the whole data-set in batch mode. In indexing terminology, few indexes are *bulk-loaded*. Often, indexes are created and updated by sequential access of elements, assuming that data elements are available only one-at-a-time. This unnecessary restriction leads to weak adaptation to data statistics. Index creation is dependent on the order of data access and hence can trapped in poor local minima.

- **Distance Estimation with Bounding Rectangles and Spheres :** Typically, distance bounds to regions of the database are calculated through Minimum Bounding Rectangles (MBRs) and Minimum Bounding Spheres (MBSs). As $d \to \infty$, the useful (or *used*) volume inside MBRs and MBSs $\to 0$ (a true consequence of the 'curse of dimensionality'). An attempt to compensate for this by partitioning through both MBRs and MBSs was performed with minimal success [40].

- **The Law of Large Numbers Effect :** Some data-sets are just not indexable, typically synthetic data-sets for e.g. Uniform iid data-sets. If $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^d$, X_i, Y_i are iid i.e. $\forall i$ and $X_i \perp Y_j, \forall i, j$, then for "large" d
  $\sum_{m=1}^{d}(X_m - Y_m)^2 \approx d.E((X_k - Y_k))^2)$ =constant
  This result (see [45], [46]) would imply that any two elements in the data-set are almost at the same distance away from each other. We note that this is a consequence of the Law of Large Numbers [47]. This has led to questioning notions of similarity [48] and the meaningfulness of conventional distance measures [49].

Often, researchers have mistakenly concluded that the nearest-neighbor search, with Euclidean distance metric, is hopeless in *all* high dimensional data-sets, due to the notorious "curse of dimensionality". This is has been shown to be an overpessimistic

inference. Specifically, the authors of [50] have shown that what determines the search performance (at least for R-tree-like structures) is the *intrinsic* dimensionality of the data set and not the dimensionality of the address space (or the *embedding* dimensionality).

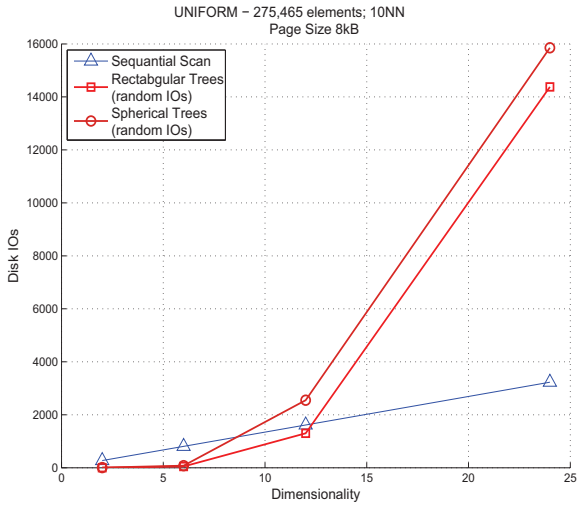

Figure 2.1. Rectangular Trees vs. Spherical Trees vs. Scan: UNIFORM Data-set

Consider the following results from experiments conducted on a UNIFORM (synthetic) data-set and the AERIAL (real) data-set. They both have 275,465 elements and are stored on a hard drive with 8kB page size. We vary the dimensionality and extract the 10NN for 100 queries randomly extracted from each data-set. We compare the average performance of a typical rectangular and spherical trees and contrast with sequential scan. In the UNIFORM data-set (Figure 2.1), as predicted by theory [45], once dimensionality crosses 10, it becomes more meaningful to perform sequential scan. However, on the real data-set AERIAL (Figure 2.2), performance the tree based methods still access less data pages than sequential scan. But sequential scan could still be preferable, because of the relatively less cost of sequential page access versus random page access.

The typical (and often implicit) assumption in many previous studies is that the data is uniformly distributed, with independence between attributes. However, real data sets overwhelmingly disobey these assumptions; rather, they typically are skewed

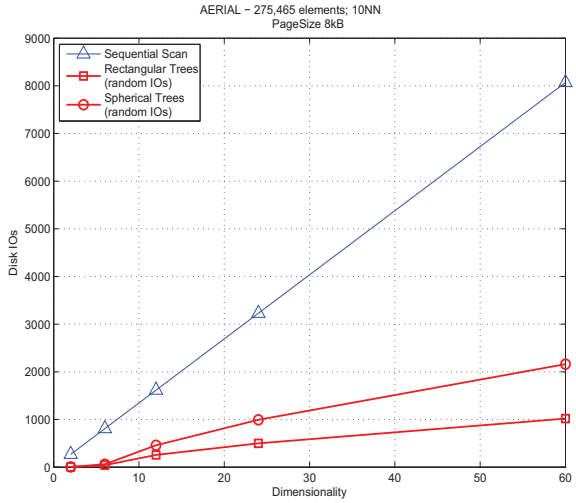

Figure 2.2. Rectangular Trees vs. Spherical Trees vs. Scan: AERIAL Data-set

and exhibit intrinsic (*fractal*) dimensionalities that are much lower than their embedding dimension, e.g., due to subtle dependencies between attributes. Hence, real data-sets are demonstrably indexable and that too with Euclidean distances. Whether Euclidean distances are *perceptually* acceptable is an entirely different matter, and it is for this reason, significant research activity (in content-based image retrieval) has been directed toward Mahalanobis distances (see [51],[52]).

2.6 Search Quality

The quality of search results could be viewed as the analog of distortion in compression. However, as show in subsequent sections, search quality has different meanings in different contexts.

2.6.1 Exact vs. Approximate Neighbors

In the context of indexing, when one speaks of quality, one refers to how accurately or what fraction of the true nearest neighbors (as determined by the features and distance

measure used) were returned. One could easily ensure perfect quality by reading the entire database i.e. by sequential scan, but this would result in high search times. On the other hand, by sacrificing some accuracy in search, search time could be reduced. The quality of the search is typically measured through precision/recall [30], distance-sensitive recall [53] or distance ratio [54] [55]. In an information theoretic analysis of approximate search [56], the use of a three-way distortion measure (which depends on the query, data-element and a compressed version of the data element) for evaluating, and optimizing for, search quality has been proposed.

2.6.2 User Perception/Feedback

When dealing with multimedia data, there is a need to retrieved perceptually relevant results. Hence, feature extraction is performed with user perceptions in mind. However, there is an inherent approximation of user perceptions and hence search results are always approximate. So even a "best" result is undefined. In order to retrieve the required data objects, the search happens over several iterations till the user is satisfied with the quality of the results returned. At each step, the user gives feedback and this is used to improve the quality of results in the next round. In this context, since feature vectors are already approximations of perceptions, one may consider further compression of feature vectors, to reduce storage space and possibly ease indexing.

In order to return perceptually relevant results, two strategies are possible. We either

1. fix the distance measure, return approximate results and possibly search again with a new query vector. It is possible to train, and periodically update, the index [53]. Or

2. assume errors in search results are due to the wrong distance measure. The system gradually tunes the distance measure and but performs exact search i.e. fixes the index, trains distance measure and returns the exact nearest neighbors.

Quality is normally measured through precision and recall. In this thesis, we focus on the impact of the second strategy on indexing (see Chapter 3, Section 3.4).

2.7 Compression for Similarity Search

The early explicit role of compression in indexing was a "passive" role i.e. just data object compression. With careful scrutiny, one may view indexes themselves as compressed versions of the database. For example, recursive data-set partitioning reveals structure and clustering of the data. Different levels of the index structure represent a compressed version of the data-set at a different resolution i.e. multi-resolution (compressed) representations. However, today compression plays an active role. Modern indexing techniques such as the VA-File[43] explicitly employ compression. In high dimensions, compressed representations of the database allow for quick browsing and search of the database. Such indexes have only a single layer and directly trade index size for retrieval speed (time)[2] .

2.7.1 Information-Theoretic Bounds on Search and Retrieval

Indexing has also been characterized within an information theoretic setting. The search time is dominated by the IO time, which in turn is proportional to the number of bits read from the storage device (typically, the hard-disk drives). This depends on index structure but even the optimal index cannot retrieve less information than a certain limit (which depends on the data-set). In the context of exact search, this was related to the capacity of identification systems [58]. In the context of approximate search, by relating search quality to distortion, the search quality-search time tradeoff was characterized in [56].

[2]VQ has also been applied to image feature extraction. By performing compression, redundancies are eliminated and only dominant patterns appear, for e.g. in dominant color descriptor extraction [24] and the Keyblock method [57].

2.7.2 State-of-the-art of Exact Indexing

Recent trends in database indexing involves explicit use of compression. We next describe two classes of such indexing techniques that resort to compression.

Vector Approximation Files

A popular and effective technique to overcome the curse of dimensionality is the vector approximation file (VA-File) [43]. VA-File partitions the space into hyper-rectangular cells, to obtained a quantized approximation for the data that reside inside the cells. Non-empty cell locations are encoded into bit strings and stored in a separate *approximation file*, on the hard-disk.

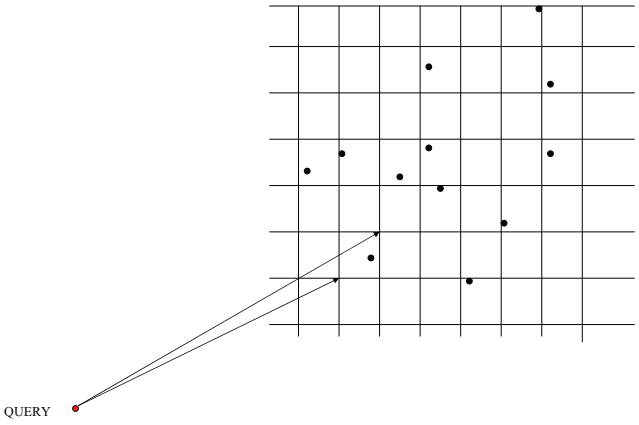

Figure 2.3. VA-File: Uniform Quantization

Distance Bound Estimation in the VA-File

Consider a query $\mathbf{q}$ and a rectangle $\mathcal{R}$ in a d-dimensional space. The rectangle is defined by the co-ordinates of its main diagonal $\mathbf{l} = [l_1, ..., l_d]^T$ and $\mathbf{u} = [u_1, ..., u_d]^T$ i.e.

$$\forall \mathbf{x} \in \mathcal{R}, l_m \leq x_m \leq u_m, \forall m = 1, 2, ..., d \tag{2.4}$$

To lower bound the distance from the query to the rectangle, we evaluate

$$\begin{aligned} d(\mathbf{q}, \mathcal{R}) &= \min_{\mathbf{x} \in \mathcal{R}} d(\mathbf{q}, \mathbf{x}) \\ &= \min_{\mathbf{x} \in \mathcal{R}} \sqrt{\sum_{m=1}^{d} (x_m - q_m)^2} \\ &= \sqrt{\min_{\mathbf{x} \in \mathcal{R}} \sum_{m=1}^{d} (x_m - q_m)^2} \\ &= \sqrt{\sum_{m=1}^{d} \min_{l_m \leq x_m \leq u_m} (x_m - q_m)^2} \end{aligned}$$

Now, each minimization within the summation can be easily performed, without complex calculations. In a similar fashion, an upper bound on the distance to the rectangle can be easily evaluated. During a nearest neighbor search, the vector approximation file is sequentially scanned and upper and lower bounds on the distance from the query vector to each cell are estimated [3]. The bounds are used to prune irrelevant cells. The final set of candidate vectors are then read from the hard-disk and the exact nearest neighbors are determined. At this point, we note that the terminology "Vector Approximation" is somewhat confusing, since what is actually being performed is *scalar quantization*, where each component of the feature vector is *separately and uniformly quantized* (in contradistinction with vector quantization in the signal compression literature).

The VA-File was followed by several more recent techniques to overcome the curse of dimensionality. Methods such as LDR [59] and the recently proposed non-linear approximations [60] aim to outperform sequential scan by a combination of clustering and dimensionality reduction. Some other indexes, such as VA$^+$-File [61] build on and extend the principles of VA-File, in that a suitable approximation file is created, and during query processing, this is used to prune the data-set of irrelevant objects.

In the VA$^+$-File, the data set is rotated into a new set of uncorrelated dimensions, prior to vector approximation. The rotation is chosen to be the data-set de-correlating

[3] As a separate contribution of this thesis, we show how the calculation of upper bounds can be avoided. See Appendix A for details.

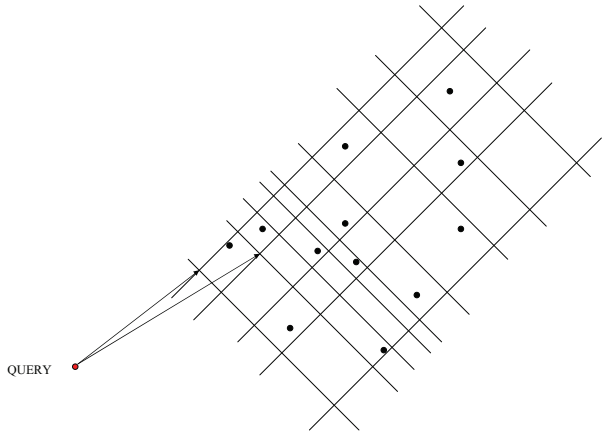

Figure 2.4. VA$^+$-File: Transform Quantization

transform i.e. the Karhunen-Loève Transform (KLT). Evaluating the KLT (or equivalently the principal components) also returns the variances of the new dimensions. Bit distribution among the new dimensions is performed in a non-uniform fashion, with more bits for dimensions with higher energies (variances). The quantizer in each new dimension is designed through the Lloyd algorithm [62] [63]. In other words, the data-set undergoes transform coding. Each new dimension is quantized at the appropriate resolution, and the approximation file is created. During query processing, distances are evaluated to the bounding boxes around each data point and the nearest elements are retrieved. There also exist a few hybrid methods, such as the A-Tree [44], and IQ-Tree [64], which combine VA-style approximations within a tree based index.

Clustering and Cluster Pruning

Since clustering performs vector quantization, it can exploit correlations across feature dimensions and reduce preprocessing storage. In some methods, such as in [65] [66], query-cluster distance is bounded by using bounded spheres and the clusters are retrieved in order of distance bounds till the kNNs are identified. Other techniques, such as LDC [67], iDistance [68], and Pyramid Tree [69], are based on local dimensionality reducing transformations. The data-set is clustered and partitioned. In each partition,

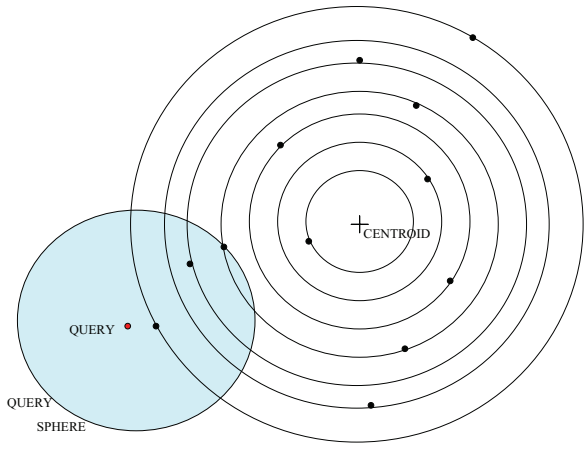

Figure 2.5. iDistance: Indexing the Distance

the distances of the resident vectors to some reference point, typically the centroid, are evaluated. The feature vectors in a partition are now indexed by their centroid-distance, using ubiquitous one-dimensional indexes such as the B^+-tree [10]. During query processing, spheres of gradually increasing radii are drawn around the query, until they intersect a cluster sphere. Now, the *relevant* elements in the partition, identified by centroid-distances which lie in the intersecting region, are retrieved for finer scrutiny. The search radius is set to such a value that on the average high-precision *approximate* NNs are returned.

It is to be noted that co-ordinate hyperplanes translated to the centroid divide the feature space into 2^d boxes, where d is the space dimensionality. In LDC [67], another approximation layer is created, by generating a box identification code for each resident point. Once an initial set of candidates have been identified with the B^+-tree, the corresponding approximations are scanned to further filter out irrelevant points within the partition. The surviving elements are finally retrieved from the hard drive to determine the nearest neighbors. In order to reduce disk IO, care is taken to control the maximal fraction of space searched. In this form, LDC can provide significant speed-ups and generate high precision results but cannot guarantee that the exact nearest neighbors

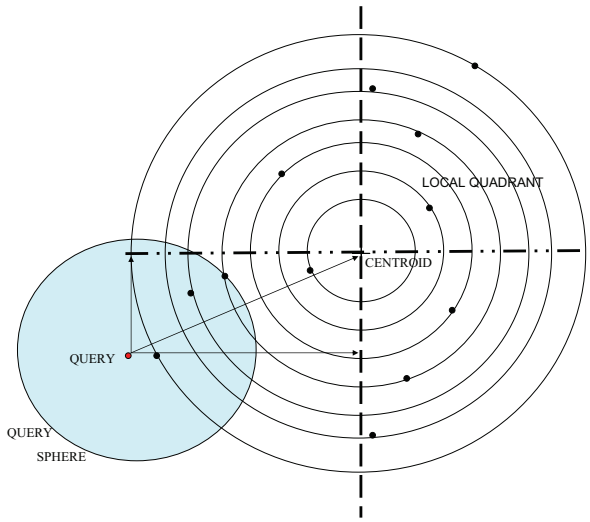

Figure 2.6. LDC: Local Digital Coding

will be found (much like the iDistance [68]).

2.7.3 Approximate Nearest Neighbor Indexing

It has often been argued that the feature vectors and distance functions are often only approximations of user perception of similarity. Hence, even "exact" similarity search is inevitably approximate. Conversely, by performing an approximate search, considerable savings in query processing time would be possible. Examples of such search strategies are clustering techniques such as KLT-Clustering [55], Clindex [70], Local dimensionality reduction [59] and CSVD [71] as well as scalar quantization (compression) approaches such as VA-LOW [72]. We note that even the famous locality sensitive hashing (LSH) approach ([73] [74] and more recent [75]) performs randomized binning of the data-set (a suboptimal clustering of the data-set). Other methods include multi-level dimensionality reduction [76], probabilistic searches (PAC-NN) [77] etc. The reader is directed to [30] for a more detailed survey of approximate similarity search.

Retrieved Set Reduction vs. Retrieved Information Reduction

In practice, there are two popular lines of attack for approximate similarity search [30]. In the first approach, a modified distance function is used over feature space or the space itself, is modified. Generally, a smaller number bits or dimensions are used to represent the feature vectors. The approximated feature vectors are stored in a sequential file, which is scanned for NN queries. This compression-based approach is called *retrieved information reduction* [55]. Search algorithms suggested in [59], [43] and [61] fall under this category. This method of approximate NN search is crucial in high-dimensional data sets, as comparing all dimensions at the same time dramatically degrades the efficiency of the query processing.

The second approach uses the exact distance and the exact feature vectors to compare objects. Here, the data set is partitioned in to several clusters, each of which is stored into a separate sequential file. Each cluster has a representative, which is compared with the query. The elements of a cluster are accessed sequentially, only if the representative is "close" to the query. In this way, the search space is pruned down to the set of objects most likely to contain the nearest neighbors of the query. Several rounds of pruning are possible. This clustering-based approach is called *retrieved set reduction* [55]. The search algorithms suggested in [70] and [53] fall under this category. This approach is important, especially in large data sets. Clearly, it can outperform any approximate NN query processing technique that tries to access the entire data set.

It is also possible to formulate a search procedure that has the "best of both worlds". For example, Clustering with Singular Value Decomposition (CSVD) [71] and KLT-clustering [55] group data into clusters but each cluster is represented only by a reduced set of dimensions. In VQ-Index [54], the data-set is partitioned into several regions using the query statistics, but thereafter, each partition is represented by a smaller *codebook*.

2.8 Compression in Correlated Source (Sensor) Databases

We now consider the problem of storing correlated sources in a database for future retrieval of any subset of them as queried by users. This problem, which we term as the *fusion coding of correlated sources*, differs from the well known distributed source coding setting [78, 79] in that all information about the sources is centrally available during encoding for storage in the database. However, only statistical information about future queries is available. Such database design introduces fundamentally new and interesting challenges: On the one hand, inter-source correlations may be exploited via joint coding to reduce the overall storage requirement and to potentially reduce the retrieval time. On the other hand, a future query may select only few of the sources for retrieval, and it would be wasteful to have to retrieve the entire (jointly) compressed data only to reconstruct a small subset.

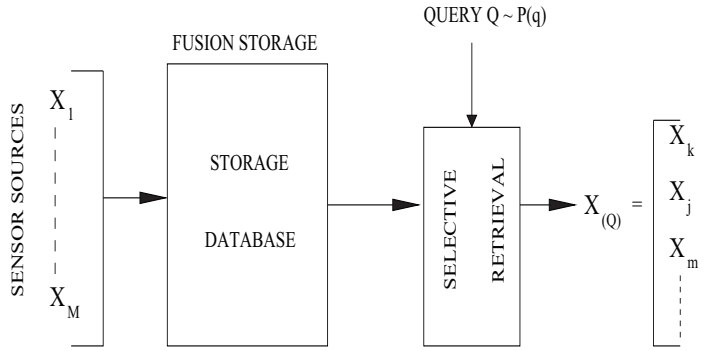

Figure 2.7. Fusion storage and selective retrieval or Fusion coding of correlated sources

Thus, at the heart of the problem, lies a trade-off between storage rate (space) versus retrieval rate (time), (both measured in terms of bits stored or retrieved). An example application of the proposed fusion coding of correlated sources is in the arena of sensor networks, which has been the focus of extensive research in recent years. Much of the effort in sensor network design has been dedicated to the development of device and communication technologies [80]. But in order to fully realize the potential of most such systems, it is necessary to efficiently store the vast volumes of data generated by the

network for future retrieval, as needed for analysis or other uses.

Consider a network of surveillance cameras covering a scene. The video signals generated by these cameras are expected to be highly correlated, since they are covering the same scene. This data is sent to a fusion center to be stored for possible future analysis. We note in passing the more generalized setting of *multiple* storage centers, each of which store video data from one or more cameras, i.e. *distributed storage*, but for the sake of simplicity, we assume herein that all video signals are stored in a single fusion center. When the data from the fusion center is eventually accessed by a user, it is very likely that the views from only a small subset, and not all, of the cameras will be requested at any given time. Note also that fusion storage of correlated sources has applications even in areas that are far removed from traditional signal processing and communications, such as storage and indexing of stock market data streams [81].

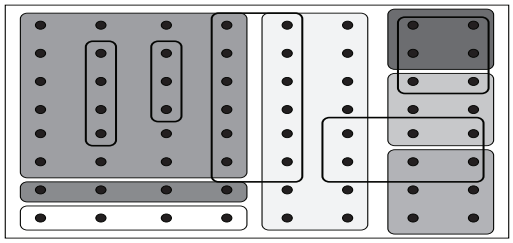

Figure 2.8. A 2D sensor field: dots represent sensors and boxes represent regions of interest (queries)

Figures 2.7 and 2.8 depict the setting we consider. The fusion coding problem was first identified in [82], where an information-theoretic characterization of the achievable rate region for lossless coding, via reformulation as a multi-terminal source coding problem [83], was derived.

2.8.1 Why Compress for Storage?

The rapid development of low-cost, high-density storage devices motivates one to consider why compressed storage is even considered. The issue is that stored data are

often going to be requested by users and without (joint) compression of the requested sources, unnecessarily additional bits would need to be retrieved which could significantly increase the retrieval time. A secondary issue is that we are considering storage of data from many correlated sources, the number of which could run from hundreds to several thousands. Hence the volume of data encountered could overwhelm storage systems, unless compression is performed. Compression would also be intricately linked with any high-dimensional index for nearest neighbor search over such databases [56], [43].

2.8.2 Information Theoretic Bounds on Lossless Storage and Retrieval

We continue with the notation developed in Section 2.4.2. To compress any source of information X, the number of bits required for lossless representation should be at least the Shannon entropy (*entropy-rate*, for sources with memory) of the source, $H(X)$. Information theory offers straightforward bounds on the performance of naive compression techniques when applied to the problem of fusion coding.

Minimal Storage Rate :

It follows from Shannon's basic result that the minimum number of bits required to store M sources $X_1, \ldots, X_M$, i.e. the minimal *storage rate* is

$$R_{s,min} = H(X_1, ..., X_M) \tag{2.5}$$

Since $H(X_1, ..., X_M) \leq \sum_m H(X_m)$ joint compression of correlated sources *never* requires more bits for storage than separate compression (by exploiting inter-source redundancies). The retrieval rate for this method is similarly

$$R_r = H(X_1, ..., X_M) \tag{2.6}$$

since *the entire compressed description needs to be retrieved for any query.*

Minimal Retrieval Rate :

If we denote the set of sources queried as,

$$X_{(\mathbf{q})} \quad = \quad \{X_m, \forall m : q_m = 1\}$$

the minimum number of bits required to reconstruct the sources requested in query $\mathbf{q}$ is $H(X_{(\mathbf{q})})$ and hence, the minimum retrieval rate averaged over the query distribution is

$$R_{r,min} = \sum_{\mathbf{q}} P(\mathbf{q}) H(X_{(\mathbf{q})}) \leq H(X_1, ..., X_M)$$

This implies that joint compression is suboptimal in retrieval speed. In order to achieve the fastest retrieval speed, we need to *separately compress and store each subset* of sources that may be requested,. However, unless M is very small or the set of queries $\mathcal{Q}$ is severely restricted, the storage requirement would be impractically high as it would have to individually accommodate a very large (possibly an exponential) number of queries, i.e.

$$R_s = \sum_{\mathbf{q} \in \mathcal{Q}} H(X_{(\mathbf{q})}) \gg H(X_1, ..., X_M) = R_{s,min} \tag{2.7}$$

Thus, it is clear that the optimum storage technique is wasteful in retrieval speed, and the optimal retrieval technique is wasteful in storage.

Scalar/Separate Compression :

Lastly, we consider separate/scalar coding of each source. The storage rate for lossless separate compression is

$$R_{s,sep} = \sum_{m} H(X_m) \gg R_{s,min} \tag{2.8}$$

while the retrieval rate

$$R_{r,sep} = \sum_{\mathbf{q}} P(\mathbf{q}) \sum_{m} q_m H(X_m) \gg R_{r,min} \tag{2.9}$$

Clearly, separate/scalar coding(quantization) would be severely sub-optimal both in storage efficiency and retrieval speed. It ignores the correlations that exist across sources, thereby incurring larger storage costs and higher retrieval time than necessary.

2.8.3 Fusion Coding Objectives in Other Forms

In fusion coding, we compress database of sources but want access to and retrieve only relevant information. There are several other applications that have objectives similar to that of fusion coding, i.e. require "compression with random access". Some notable examples are

- **Browsing Compressed Video:** When video is compressed, prediction (or inter-frame coding) is used to exploit the significant temporal correlations that may be present, in addition to intra-frame coding. However, when viewed there is a need to fast-forward or rewind or jump to a scene of interest. If purely predictive coding were performed, then the whole video would need to be uncompressed to the point where streaming can be resumed, causing significant delay. Typically, I-frames (i.e.) purely intra-coded frames are available every few milliseconds. Predictive coding is restricted to sections between I-frames. The codec "jumps" to the closest I-frame and decodes the video stream from that section, thereby avoiding excessive delay.

- **Multi-file Compression :** Often, in an archive for e.g on news servers, several files are jointly, losslessly coded (through a Lempel-Ziv like algorithm). However, the need to extract specific files from a huge archive requires careful decompression.

- **Executable Data Compression :** Depending upon the complexity of the task to be performed, program data are stored in compressed format. In embedded systems, since on-chip cache is limited, there is a need to retrieve only small, relevant sections of the code, without decompressing the whole program.

- **kNN Search ?** It is possible to even view kNN search in the light of fusion coding. The optimal search for kNNs accesses only relevant portions of database i.e. accesses only relevant images/feature vectors. By creating an index, irrelevant portions of the database may be eliminated and only relevant portions are retrieved.

Sequential scan could be viewed as the analog to joint compression, wherein all the data are retrieved.

Chapter 3

Indexing by Cluster Distance Bounding

We consider a clustering approach for kNN search as an alternative to the Vector Approximation (VA) Files. The data set is clustered using a standard clustering or vector Quantization (VQ) technique, e.g., K-means or Lloyd's algorithm and during query processing, the "nearest" clusters are loaded into the main memory. Clustering exploits correlations and dependencies across feature dimensions. In fact, clustering algorithms are the methods of choice for the design of vector quantizers [62]. As a result, a more compact representation of the data-set is possible, for the same representation quality. Alternatively, better representation quality is possible when allowed the same number of codevectors. This facilitates nearest neighbor search since the search can now be performed over a very compact and accurate representation of the data-set prior to actual access of the database.

Additionally, clustering results in several similar feature vectors sharing the same representative. In other words, in clustering techniques the mapping is from the code-vectors to the many database entries (or feature vectors) they represent, while compression-based methods such as VA-File employ the reverse mapping. Thus each cluster retrieved re-

turns several promising candidates. Next, we note that each cluster retrieved results in only one more random IO and several sequential IOs. Finally, observe that the sequential scan is a special case of clustering i.e. with one cluster. Thus, a clustering approach would use costly random disk IOs more efficiently.

In order to adapt clustering/VQ to exact NN search, tight estimates (lower bounds) of query-cluster distance, *without accessing the clusters*, are necessary. This would lead to retrieval of clusters in decreasing probability of containing entries relevant to the query. Once the true kNNs have been found, clusters with cluster-distances that exceed the k^{th}-NN distance can be eliminated from the search procedure. While such a clustering approach to search has been studied in the image database and mainstream database community (see [66, 65, 71], [68] [67]), the earlier approaches have been applicable only for *approximate* nearest neighbor search. The distance bounds (based on bounding hyperspheres) are loose and hence the search strategy performs poorly when adapted to exact nearest neighbor search. We next present an effective cluster distance bound that complements our branch-and-bound search algorithm.

3.1 The Hyperplane Bound

Let $d(\mathbf{x}, \mathbf{y})$ be a distance function that estimates the distance between vectors $\mathbf{x}$ and $\mathbf{y}$ in the feature space.

$$d : \mathcal{R}^n \times \mathcal{R}^n \rightarrow [0, \infty) \tag{3.1}$$

In subsequent discussion, we shall specialize to the Euclidean distance over (real vector spaces) as the feature similarity measure i.e. $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$. We define the distance from query $\mathbf{q}$ and a cluster $\mathcal{X}_m$ as

$$d(\mathbf{q}, \mathcal{X}_m) = \min_{\mathbf{x} \in \mathcal{X}_m} d(\mathbf{q}, \mathbf{x}) \tag{3.2}$$

The distance of vector $\mathbf{q}$ to a hyper plane $H(\mathbf{n}, p) = \{\mathbf{y} : \mathbf{y}^T\mathbf{n} + p = 0\}$ is defined in the normal fashion as

$$d(\mathbf{q}, H) = \frac{|\mathbf{q}^T\mathbf{n} + p|}{\|\mathbf{n}\|_2} \tag{3.3}$$

Given a cluster $\mathcal{X}_m$, the query $\mathbf{q}$ and a hyperplane H that lies between the cluster and the query (a "*separating hyperplane*", see Figure 3.1), by simple geometry it is easy to see that for any $\mathbf{x} \in \mathcal{X}_m$

$$\begin{aligned} d(\mathbf{q}, \mathbf{x}) &\geq d(\mathbf{q}, H) + d(\mathbf{x}, H) \\ &\geq d(\mathbf{q}, H) + \min_{\mathbf{x} \in \mathcal{X}_m} d(\mathbf{x}, H) \\ &= d(\mathbf{q}, H) + d(\mathcal{X}_m, H) \\ \Rightarrow d(\mathbf{q}, \mathcal{X}_m) &\geq d(\mathbf{q}, H) + d(\mathcal{X}_m, H) \end{aligned} \tag{3.4}$$

3.1.1 Cluster Distance Bounding

If $\mathcal{H}_{sep}$ represents a countably finite set of separating hyperplanes (that lie-between the query $\mathbf{q}$ and the cluster $\mathcal{X}_m$),

$$\Rightarrow d(\mathbf{q}, \mathcal{X}_m) \geq \max_{H \in \mathcal{H}_{sep}} \{d(\mathbf{q}, H) + d(\mathcal{X}_m, H)\} \tag{3.5}$$

The second lower bound presented in (3.5) can be used to tighten the lower bound on $d(\mathbf{q}, \mathcal{X}_m)$. Next, we note that the *boundaries* between clusters generated by the K-means algorithm are *linear hyperplanes*. If $\mathbf{c}_1$ and $\mathbf{c}_2$ are centroids of two clusters $\mathcal{X}_1$ and $\mathcal{X}_2$, and $\mathcal{Y}_{12}$ the boundary between them, then $\forall \mathbf{y} \in \mathcal{Y}_{12}$

$$\begin{aligned} d(\mathbf{c}_1, \mathbf{y}) &= d(\mathbf{c}_2, \mathbf{y}) \\ \Rightarrow \|\mathbf{c}_1\|_2^2 - \|\mathbf{c}_2\|_2^2 - 2(\mathbf{c}_1 - \mathbf{c}_2)^T\mathbf{y} &= 0 \end{aligned}$$

Therefore, the hyperplane $H_{12} = H(-2(\mathbf{c}_1 - \mathbf{c}_2),\ \|\mathbf{c}_1\|_2^2 - \|\mathbf{c}_2\|_2^2)$ is the boundary between the clusters $\mathcal{X}_1$ and $\mathcal{X}_2$. What is to be noted is that these hyperplane boundaries **need not be stored**, rather they can be **generated during run-time from** just **the**

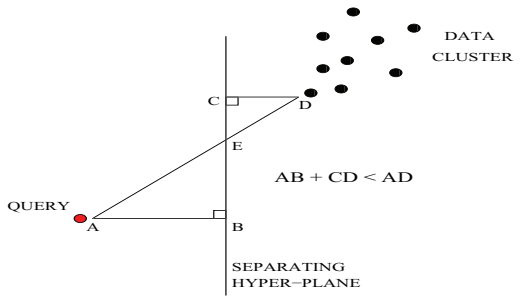

Figure 3.1. The Hyperplane Bound

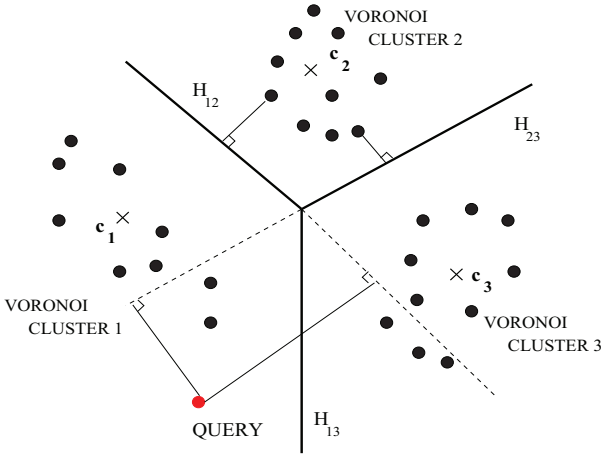

Figure 3.2. Cluster Distance Bounding

centroids $\{\mathbf{c}_m\}_1^K$ themselves. It is straightforward to show that: Given a query $\mathbf{q}$ and a hyperplane H_{mn} that separates clusters $\mathcal{X}_m$ and $\mathcal{X}_n$, it lies between the query and cluster $\mathbf{X}_m$ ***if and only if*** $d(\mathbf{q}, \mathbf{c}_m) \geq d(\mathbf{q}, \mathbf{c}_n)$.

3.1.2 Reduced Complexity Hyperplane Bound

For evaluation of the lower-bound presented in (3.4) and (3.5), we would need to pre-calculate and store $d(H_{mn}, \mathcal{X}_m)$ for all cluster pairs (m, n). With K clusters, there are $K(K-1)$ distances that need to be pre-calculated and stored, in addition to the cluster centroids themselves. The total storage for all clusters would be $O(K^2 + Kd)$, where d is the dimensionality. This heavy storage overhead makes the hyperplane bound, in this

form, impractical for a large number of clusters. We can loosen the bound in (3.5) as follows:

$$\begin{aligned} d(\mathbf{q}, \mathcal{X}_m) &\geq \max_{H \in \mathcal{H}_{sep}} \{d(\mathbf{q}, H) + d(H, \mathcal{X}_m)\} \\ &\geq \max_{H \in \mathcal{H}_{sep}} d(\mathbf{q}, H) + \min_{H \in \mathcal{H}_{sep}} d(H, \mathcal{X}_m) \end{aligned}$$

This means that for every cluster $\mathcal{X}_m$ we would only need to store one distance term

$$d_m = \min_{1 \leq n \leq K, n \neq m} d(H_{mn}, \mathcal{X}_m)$$

Figure 3.3 is representative of our index.

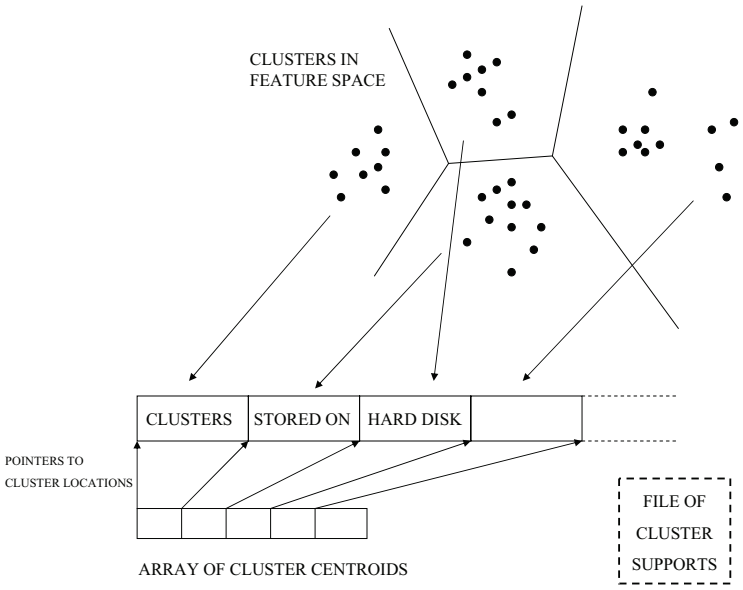

Figure 3.3. Index Structure

3.2 Data-set Clustering

The first step in index construction is the creation of Nearest Neighbor/Voronoi clusters. There exist several techniques of clustering the data-set, from the fast K-means algorithm [84] (which requires multiple scans of the data-set) and Generalized

Lloyd Algorithm (GLA) [62] to methods such as BIRCH [85], which require only a single scan of the data-set. The output of any of these algorithms can be a starting point. From each of the K clusters detected by a generic clustering algorithm, a pivot is chosen i.e. K pivot points in all. Then the entire data-set is scanned and each data-element is mapped to the nearest pivot. Lastly, data mapping to the same pivot are grouped together to form Voronoi clusters (see Algorithm 1). This would lead to slight re-arrangement of clusters, but this is necessary to retain piecewise linear hyperplane boundaries between clusters. We believe the centroid is a good choice as a pivot. Thus, quick Voronoi clustering, with possibly only a single scan of the entire data-set, can be achieved using any generic clustering algorithm.

Algorithm 1 VORONOI-CLUSTERS($\mathcal{X}$,K)

1: //Generic clustering algorithm returns
   //K cluster centroids
   $\{\mathbf{c}_m\}_{=1}^{K} \leftarrow$GenericCluster($\mathcal{X}$,$K$)
2: set $l = 0$, $\mathcal{X}_1 = \phi, \mathcal{X}_2 = \phi, \ldots, \mathcal{X}_K = \phi$
3: **while** $l < |\mathcal{X}|$ **do**
4: $l = l + 1$
5: //Find the centroid nearest to data element $\mathbf{x}_l$
   $k = \arg\min_m d(\mathbf{x}_l, \mathbf{c}_m)$
6: //Move $\mathbf{x}_l$ to the corresponding Voronoi partition
   $\mathcal{X}_k = \mathcal{X}_k \bigcup \{\mathbf{x}_l\}$
7: **end while**
8: return $\{\mathcal{X}_m\}_{=1}^{K}, \{\mathbf{c}_m\}_{=1}^{K}$

We note that the K-means, GLA and BIRCH algorithms are fast and can generate reliable estimates of cluster centroids, from sub-samples of the data-set. Typically, for K clusters, even a sub-sample of size $100K$ is sufficient. As we shall see, for the range of clusters we are considering, this would be overwhelmingly smaller than the data-set.

Faster index construction would be possible by allowing for hierarchical and multi-stage clustering. However, only the clusters at the leaf level are returned.

We tested several clustering techniques including GLA and BIRCH, and the results were largely similar. While it is possible to also optimize the clustering itself, that is not our goal in these experiments.

3.3 Experimental Results

We have conducted extensive tests on real data-sets to characterize and compare the performance of our index structure with other popular schemes. By varying the number of clusters (or some suitable parameter), a range of solutions was obtained for each index, which are subsequently compared.

3.3.1 Data-sets and Experimental Set-up

We tested our index on 5 different data-sets - HISTOGRAM, SENSORS, AERIAL, BIO-RETINA and CORTINA. See Table 3.1 for details on size and dimensionality. The HISTOGRAM [1] data-set consisted of a color histogram extracted from images on a CD. The second real data-set, SENSORS, was generated by the Intel Berkeley Research Lab[2] . Data were collected from 54 sensors deployed in the Intel Berkeley Research lab between February 28 and April 5, 2004. Each sensor measures humidity, temperature, light and voltage values once every 31 seconds. We retain data from those sensors that generated in excess of 50,000 readings. This corresponds to temperature, light, humidity and voltage readings from 15 sensors which is equivalent to 60 sources.

The next two data-sets, AERIAL and BIO-RETINA, were MPEG-7 texture feature descriptors extracted from 64×64 tiles of the images. AERIAL[3] was extracted from

[1]Download from http:/scl.ece.ucsb.edu/datasets/Histogram.mat
[2]Download from http://db.csail.mit.edu/labdata/labdata.html
[3]Download from http://vision.ece.ucsb.edu/data-sets

Table 3.1. Data-Sets Used

Name	Dimensionality	No. of Vectors	Size (Pages)
HISTOGRAM	64	12,103	379
SENSORS	60	50,000	1471
AERIAL	60	275,465	8300
BIO-RETINA	62	208,506	6200
CORTINA	74	1,088,864	40,329

40 large aerial photographs while BIO-RETINA [4] was generated from images of tissue sections of feline retinas as a part of an ongoing project at the Center for Bio-Image Informatics, UCSB. On the other hand, the CORTINA[5] data-set consists of a combination of homogenous texture features, edge histogram descriptors and dominant color descriptors extracted from images crawled from the World Wide Web. In our experiments, we assumed a *page size* of 8kB. The query sets were 100 randomly chosen elements of the relevant data-sets. In all subsequent sections, we report results from experiments where the 10 nearest neighbors (10NN) were mined, unless otherwise stated (see section 3.3.9, where the number of kNNs retrieved was varied).

3.3.2 Performance Measure and Parameter Variation

The common performance metrics for exact nearest neighbor search have been to count page accesses or the response time. However, page accesses may involve both serial disk accesses and random IOs, which have different costs. On the other hand, response time (*seek times* and *latencies*) is tied to the hardware being used and therefore, the performance gain/loss would be platform dependent. We propose to *separately* track the *average* number of sequential and random accesses incurred to retrieve the 10NNs. In a typical search procedure, by varying the tunable parameters θ (where applicable), a range of solutions with different sequential and random accesses would be possible i.e. a *performance graph*. Two competing search procedures are compared by comparing the number of random seeks R given roughly the same S or vice-versa.

[4]Download from http://scl.ece.ucsb.edu/datasets/BIORETINA_features.txt
[5]http://scl.ece.ucsb.edu/datasets/CORTINA_all_feat1089K.mat

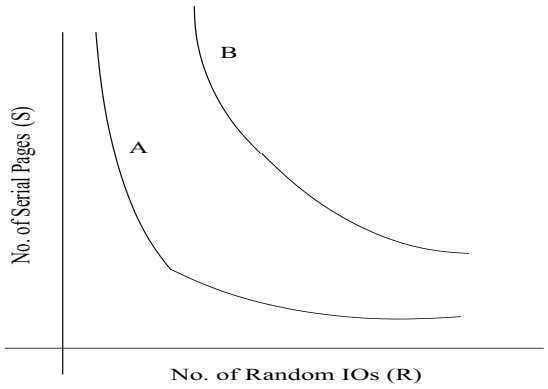

Figure 3.4. Performance Measure for Hypothetical Indexes A and B

We note that given the performance in terms of the number of sequential (S) pages and number of random IOs (R) i.e. the (S,R) pair, it is relatively easy to estimate total number of disk accesses or response times on different hardware models. The number of page accesses is evaluated as $S + R$. If T_{seq} represented the average time required for one sequential IO and T_{rand} for one random IOs, the average response time would be $T_{res} = T_{seq}S + T_{rand}R$ [6].

Table 3.2. Tunable Parameter Variation in Experiments

Indexing Method	Tunable parameter	Range Tested
VA-File	Bits per dimension	3 - 12 bits
iDistance	No. of clusters	10 - 400
LDC	No. of clusters	10 - 4000
Clustering + Cluster-distance bounding	No. of clusters	10 - 400

Table 3.2 lists the tunable parameter for each indexing technique and the range of values considered. The performance of each indexing technique is evaluated at all values of its tunable parameter within the ranges considered. In the VA-File, the number of quantizing bits is varied from 3 to 12 bits per dimension. In LDC, the number of clusters is varied from 10 - 4000 clusters, whereas in the iDistance and our and our proposed technique, the number of clusters was varied from 10-400. The performance is evaluated

[6] Assuming T_{rand} and T_{seq} are known constants for a given hardware, the optimal operating point(s) on the $S - R$ curve are where a line of slope $\frac{T_{rand}}{T_{seq}}$ is tangential to the curve.

and plotted at each setting.

3.3.3 Comparison with Bounding Rectangles and Spheres

Traditionally, spatial filtering of irrelevant clusters has been done by constructing minimum bounding hyperrectangles (MBR) and hyperspheres (MBS) around clusters, and evaluating distance bounds to these minimum bounding surfaces (see Figure 3.5). We, on the other hand, evaluate this distance bound from the separating hyperplane boundaries that lie between the query and the cluster.

In Figures 3.6 and 3.7, we present MBR, MBS and Hyperplane distance bounds along with the true (golden) query-cluster distances for a sample query, on the BIO-RETINA data-set. The distance of the query to every cluster is evaluated and these distances are sorted in ascending order. We note that his sorting order or ranking is also the order in which the clusters are read. Now, the distances bounds are evaluated to these *ranked* clusters and are compared with the true distances. The goal of this study is to compare the relative tightness of the bounds and also to observe how well they imitate the correct ranking.

We immediately note that hyperplane bounds are very close to the true distances. For the 30 cluster case, we note that the sphere (MBS) bound is *almost zero* for the first 6-8 clusters. This is because for large cluster sizes (small number of clusters) the volume occupied by bounding spheres is significantly more than the cluster volume. Hence, the ability to filter out clusters is significantly diminished. Once the number of clusters has been increased to around 300, the performance of the sphere bound improves slightly. On the other hand, though the MBR bound is looser than the hyperplane bounds, it is tighter than the sphere bound. This is a reflection on the structure of the data-clusters, suggesting that they are more rectangular than spherical in shape. We noticed similar behavior in the AERIAL data-set.

Lastly, we note that, the reduced complexity hyperplane bound is almost as good as

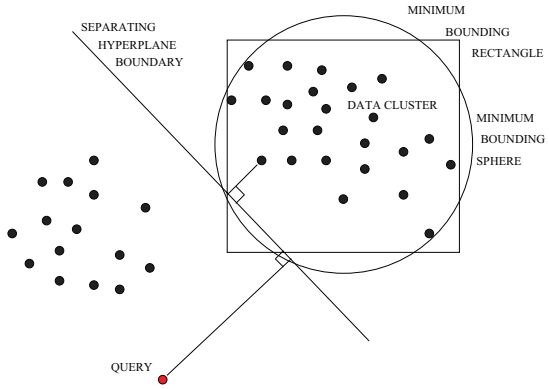

Figure 3.5. Separating Hyperplane vs. Minimum Bounding Hypersphere and Rectangle

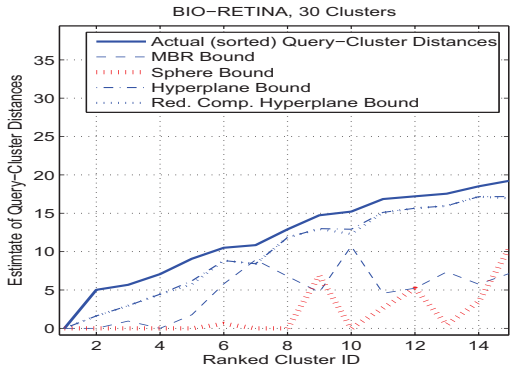

Figure 3.6. Comparison of Distance Bounds, 30 Clusters

the full hyperplane bound, while at the same time enjoying a smaller storage overhead. We also note that the *distance profiles* generated by the hyperplane bounds has almost the same (non-decreasing) nature of the true, sorted query-cluster distances. This means that even if the distance estimates are a little off, *the clusters are still searched in approximately the same order*. However, this is not so for the MBR and MBS bounds. We note in Figures 3.6 for the MBR bound and 3.7 for the MBS bound, the distance profile is very jittery. This, as we shall see, leads to a large number of needless disk accesses, further compromising the IO performance.

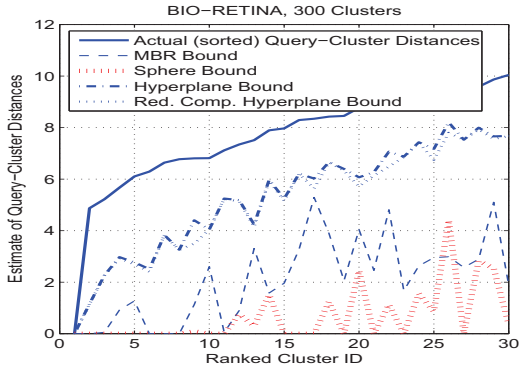

Figure 3.7. Comparison of Distance Bounds, 300 Clusters

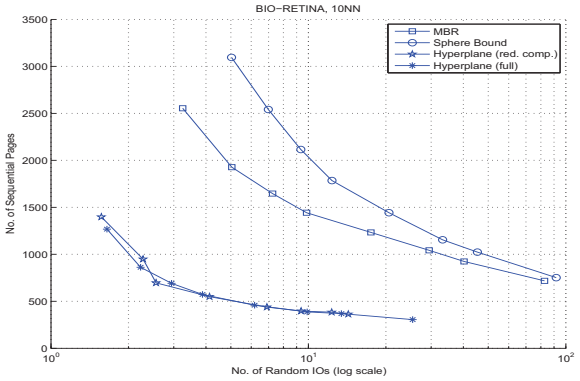

Figure 3.8. IO Performance of Distance Bounds - BIO-RETINA

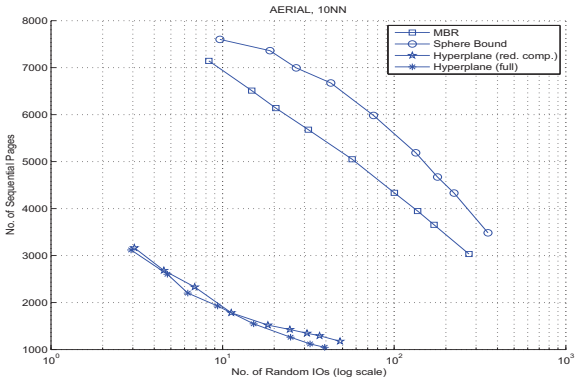

Figure 3.9. IO Performance of Distance Bounds - AERIAL

IO Performance Comparison

We conclude this study with comparison of the IO performance of the different distance bounds on the different data-sets (see Figures 3.8, 3.9, 3.10, 3.11, 3.12). We

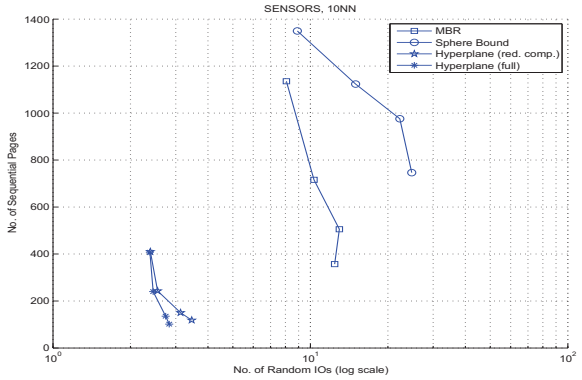

Figure 3.10. IO Performance of Distance Bounds - SENSORS

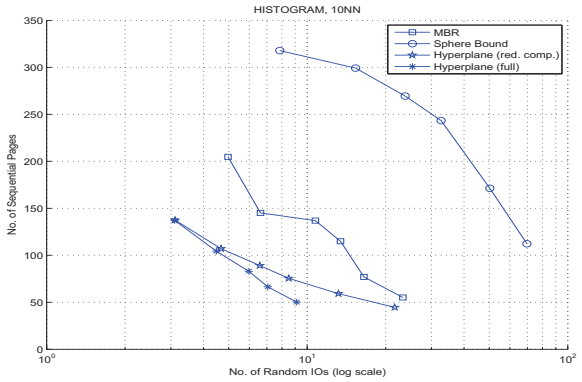

Figure 3.11. IO Performance of Distance Bounds - HISTOGRAM

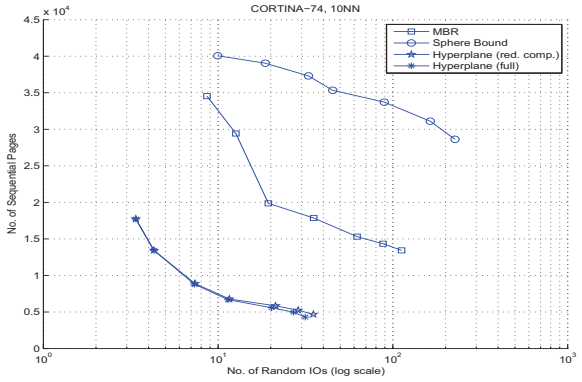

Figure 3.12. IO Performance of Distance Bounds - CORTINA

note that the sphere (MBS) and the MBR bounds are comprehensively outperformed by the two proposed hyperplane bounds in all data-sets. For the BIO-RETINA data set,

when allowed roughly 1000 sequential IO pages, the MBR and sphere bounds generate nearly 20X *more* random accesses than the hyperplane bounds. When allowed roughly 10 random disk accesses, the MBR and sphere require nearly 4X-5X more sequential disk reads. Similar trends are noticed in other data-sets, with the performance gains higher in bigger data-sets.

3.3.4 Comparison with Popular Indexes

Next, we compare the performance of our proposed clustering plus hyperplane bounding ('Hyperplane (full)' or 'Hyperplane (red. comp.)') framework with the VA-File [43], iDistance [68] and Local Digital Coding (LDC) [67]- recently proposed multidimensional index structures that have been successful in overcoming the curse of dimensionality.

Description

We note that in the VA-File initially the approximation files are read into the main memory and then the nearest elements are retrieved for finer scrutiny. As more bits are allocated in quantizing the dimensions, the size of the approximation file, which is proportional to the no. of sequential IOs, increases. At the same time, the number of vectors visited in the second stage, which determines the no. of random IOs, are reduced. Hence, the vector approximation files were created at different resolutions, from 12 bits per dimension to 3 bits per dimension, and the IO performance was evaluated at each level of compression.

For the proposed index structure, as well as the iDistance and the LDC, the performance would depend on the number of clusters created. The number of clusters was varied from 10-4000 for LDC and 10-400 for iDistance and our technique. Each multi-dimensional index was searched till the 10 *exact* nearest neighbors (10NNs) were *guaranteed* to be found.

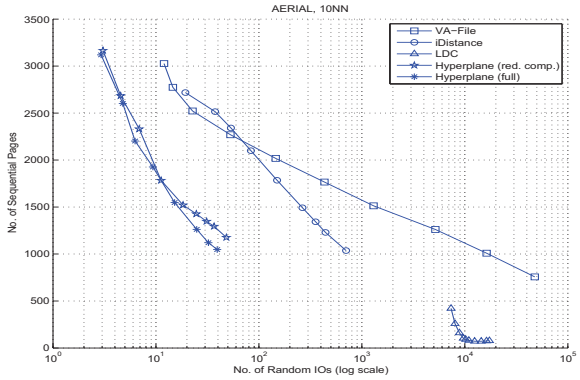

Figure 3.13. Clustering vs. VA-File vs. iDistance vs. LDC - AERIAL

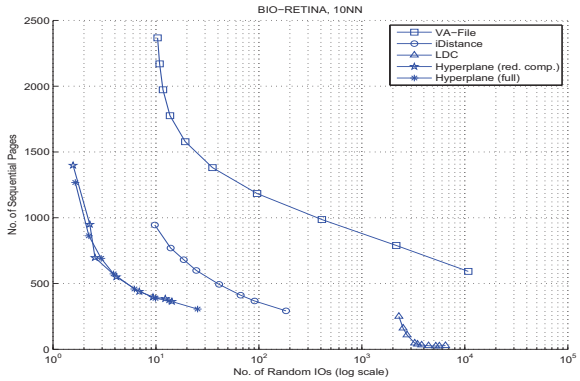

Figure 3.14. Clustering vs. VA-File vs. iDistance vs. LDC - BIORETINA

In all data-sets (Figure 3.13, 3.14, 3.15, 3.16 and 3.17), the proposed index (clustering plus hyperplane bounds) outperforms all popular indexes. For the CORTINA data-set, when allowed (roughly) the same number of sequential IOs as the VA-File, speed-ups in costly random IOs ranging from 3000X-5X, are possible. In the AERIAL data-set, when allowed (roughly) the same number of sequential IOs as the VA-File, our index structure has random IOs reduced by factors ranging from 500X-5X. When allowed the same number of sequential page accesses, random disk IOs reductions ranging from 40X-90X over the LDC and 3x-16x over the iDistance were observed.

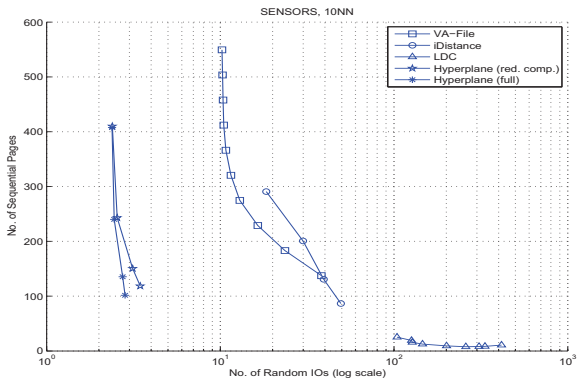

Figure 3.15. Clustering vs. VA-File vs. iDistance vs. LDC - SENSORS

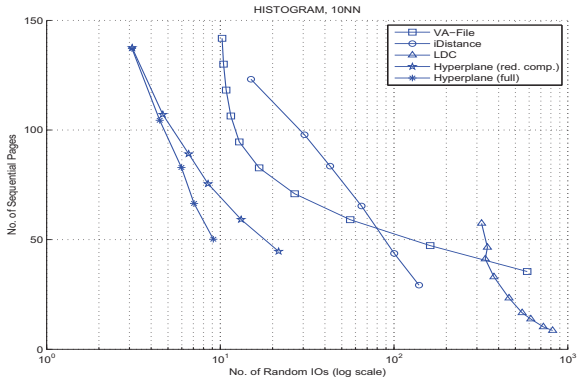

Figure 3.16. Clustering vs. VA-File vs. iDistance vs. LDC - HISTOGRAM

3.3.5 Computational Costs

We also evaluated the computational costs involved for the different indexes, in terms of the number of distance evaluations. Since the VA-File maintains an approximation for every element of the data-set and evaluates the distances to these approximation cells, the number of distance calculations are $O(|\mathcal{X}|)$, the size of the data-set. However, we note that the other indexes are based on clustering, which can exploit spatial dependencies across dimensions. Initial distances are evaluated to the cluster centroids and *relevant clusters* alone are inspected. This results in efficient filtering and substantial reductions in the number of distance computations required.

We only present results from the CORTINA data-set, as similar trends were observed

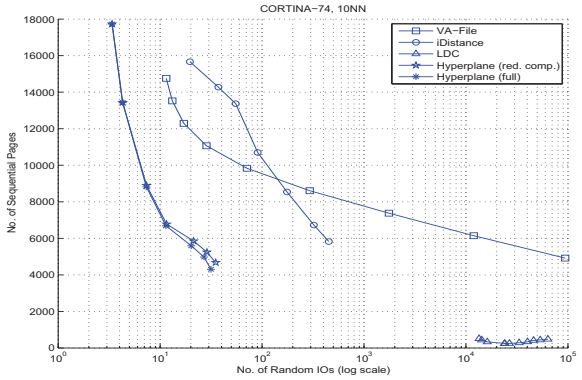

Figure 3.17. Clustering vs. VA-File vs. iDistance vs. LDC - CORTINA

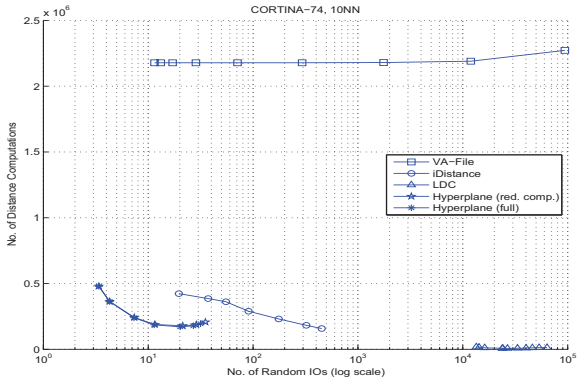

Figure 3.18. Computational Costs - CORTINA

on other data-sets as well. Our index structure requires roughly ≈10X less distance calculations than the VA-File on the average. also less than the iDistance and compare favorably with the LDC. We note specifically in the LDC, with a combination of dimension ranking arrays and partial distance searches, explicit distance evaluations to the second layer of approximations are replaced by faster binary operations. However, since processor speeds are much faster than IO speeds, out index structure still maintains its advantage in total response time over the LDC.

3.3.6 Preprocessing Storage

We compared the preprocessing storage induced by the different indexing schemes and notice substantial gains over competing indexes (Figure 3.19). This is because on the one hand the approximation file size grows with data-set size, dimensions and the number of approximation/quantization bits per dimension. On the other hand, both iDistance and LDC store one distance term for every data-set element (in addition to the centroids), thus incurring enormous preprocessing storage costs. For roughly the same number of random IOs, our proposed index requires ≈100X less storage than the VA-File.

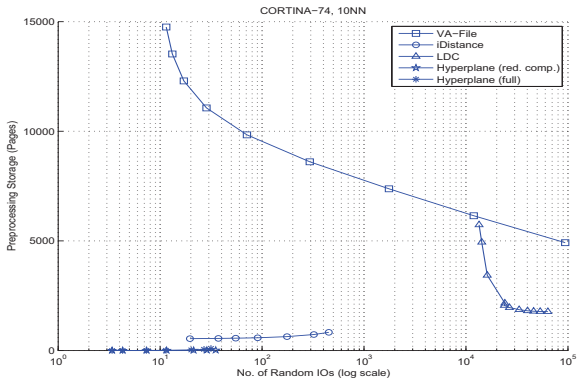

Figure 3.19. Preprocessing Storage - CORTINA

3.3.7 Scalability with Data-set Size

Figures 3.20 and 3.21 present performance comparisons of the proposed indexes with sub-sampled versions of the CORTINA data-set. Figure 3.20 represents the results of varying the data-set size for the full complexity hyperplane bound. Figure 3.21 pertains to the reduced complexity hyperplane bound. The performance variation is nearly linear in the number of sequential accesses and almost insensitive in the number of random IOs. Hence, our clustering based index scale well with data-set size.

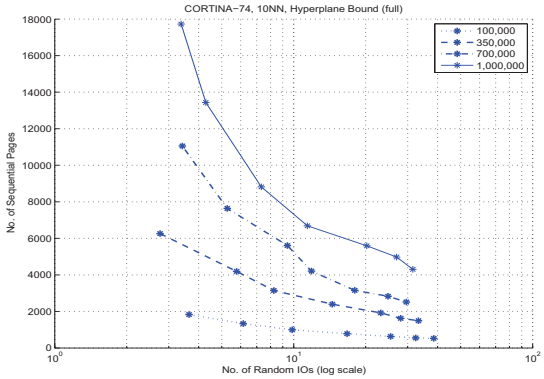

Figure 3.20. Data-set Size vs. Full Complexity Hyperplane Bound

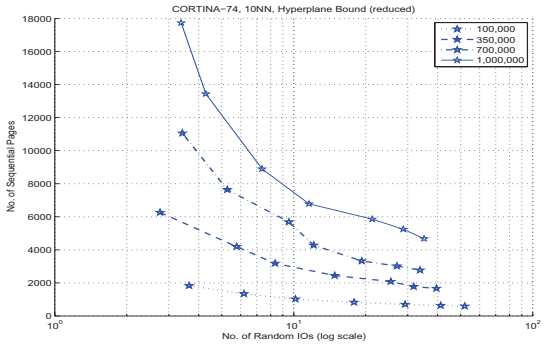

Figure 3.21. Data-set Size vs. Reduced Complexity Hyperplane Bound

3.3.8 Scalability with Dimensionality

We also evaluated the scalability of the proposed indexes with dimensions by retaining only 48, 24, 10 and 1 dimensions of the original CORTINA data-set (Figures 3.22, and 3.23 respectively). Both methods display a graceful degradation in performance with dimensionality. We also note that the VA-File degrades at very low number of dimensions while naive indexes degrade at high dimensionality.

3.3.9 Scalability with Number of Nearest Neighbors

We also vary the number of nearest neighbors with 10NNs, 20NNs and 50NNs being mined (see Figures 3.24 and 3.25). We note very slight variation in performance. This

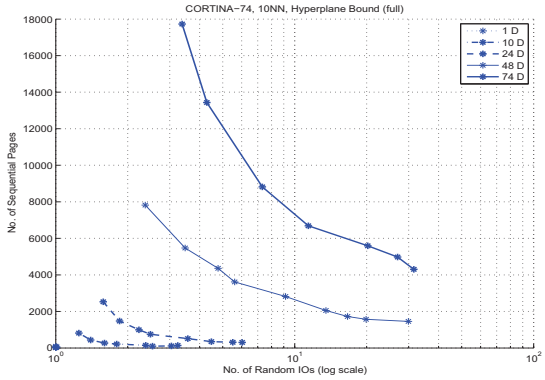

Figure 3.22. Dimensionality vs. Full Complexity Hyperplane Bound

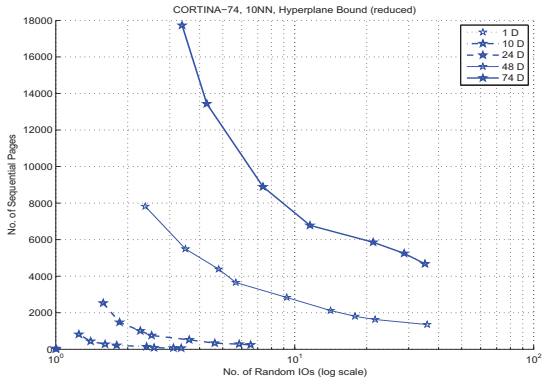

Figure 3.23. Dimensionality vs. Reduced Complexity Hyperplane Bound

is because when each cluster is retrieved, several promising candidates are available. Hence, even if we search for more NNs, we don't notice any substantial increase in disk accesses.

3.3.10 Extension to Approximate Search

Since feature vectors are imperfect representations of the corresponding objects, it could be argued that even exact search is unavoidably approximate. Since each disk IO retrieves a cluster, it may be sufficient to stop the search after the first few disk accesses to extract an approximate result, that could be further refined with user feedback. For brevity of presentation, we describe results for only the reduced complexity hyperplane

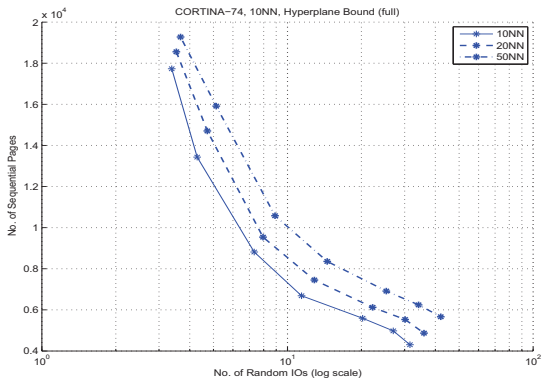

Figure 3.24. No. of kNNs vs. Full Complexity Hyperplane Bound

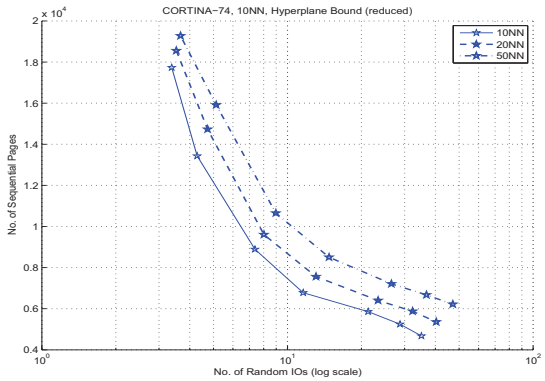

Figure 3.25. No. of kNNs vs. Reduced Complexity Hyperplane Bound

bound. Similar results are observed with the full complexity hyperplane bound.

In approximate similarity search, the quality of the retrieved is typically measured through precision or recall metrics. If $\mathcal{A}(\mathbf{q})$ and $\mathcal{G}(\mathbf{q})$ represent the approximate and golden (true) answer sets for query $\mathbf{q}$, we define

$$\begin{aligned} Precision &= \frac{|\mathcal{A}(\mathbf{q}) \cap \mathcal{G}(\mathbf{q})|}{|\mathcal{A}(\mathbf{q})|} \\ Recall &= \frac{|\mathcal{A}(\mathbf{q}) \cap \mathcal{G}(\mathbf{q})|}{|\mathcal{G}(\mathbf{q})|} \end{aligned}$$

For kNN queries, $|\mathcal{A}(\mathbf{q})|=|\mathcal{G}(\mathbf{q})|$ and hence precision equals recall.

It has also been argued that precision or recall are hard metrics that improperly measure the quality of results [54] [53] and that softer metrics such as the *distance ratio* metric proposed in [54] [55] would be more appropriate.

$$D = \frac{\sum_{\mathbf{x} \in \mathcal{A}(\mathbf{q})} d(\mathbf{x}, \mathbf{q})}{\sum_{\mathbf{x} \in \mathcal{G}(\mathbf{q})} d(\mathbf{x}, \mathbf{q})} \tag{3.6}$$

We compare against VA-LOW [72], a variant of the VA-Files that can return approximate NNs. In the VA-LOW, the search in the second phase is stopped once sufficient vectors have been visited to assure a certain precision. Figure 3.27 shows the performance of our clustering + reduced complexity hyperplane bound retrieval with precision as quality metric, while Figure 3.26 pertains to distance ratio.

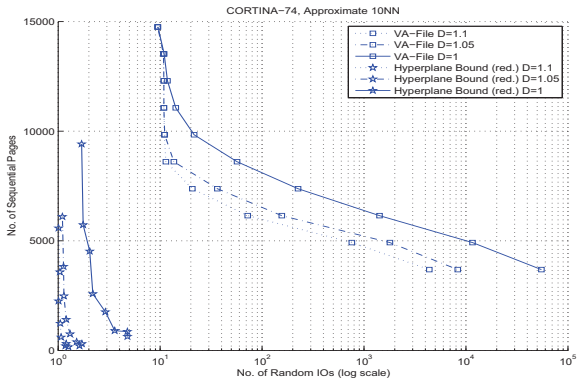

Figure 3.26. Clustering Retrieval vs. VA-LOW, Distance Ratio (D) Metric

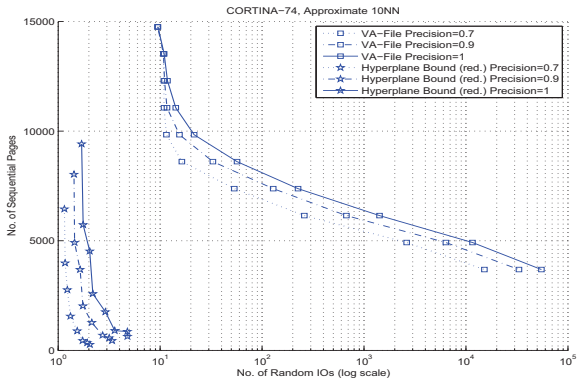

Figure 3.27. Cluster Retrieval vs. VA-LOW, Precision Metric

We first note that even the very first cluster returns high-precision results. In order to reduce the number of sequential IOs, it is necessary to allow ≈300 clusters, which results

in a few additional random disk IOs. On the other hand, in VA-LOW, several random and sequential disk access are necessary. Moreover, we observe that 100% precision results i.e. the retrieval of *exact* nearest neighbors, is possible with just the first few disk IOs. Of course, in this approach, there is no guarantee that the exact NNs have been found, even though experimentally we notice 100 % precision.

3.4 Relevance Feedback in Image Databases

While the Euclidean distance metric is popular within the multimedia indexing community, it is by no means the perceptually "correct" distance measure. Hence, significant research activity (in content-based image retrieval) has been directed toward Mahalanobis (or weighted Euclidean) distances (see [51]). The Mahalanobis distance measure has more degrees of freedom than the Euclidean distance and by proper updating (or *relevance feedback*), has been found to be a much better estimator of user perceptions (see [86, 87, 51]).

The goal in relevance feedback is to adapt the distance measure to match user expectations, by making the search an interactive process. Here, in each iteration a set of results is retrieved and user provides feedback on the relevance of each result. If Mahalanobis distance is employed, this is used to update the weight matrix for the next iteration. Sometimes, the query vector is also modified [86]. The process stops when the user is satisfied with the results.

3.4.1 Mahalanobis Weight Adaptation

The Mahalanobis distance measure has more degrees of freedom than the Euclidean distance and by proper updating (or *relevance feedback*), has been found to be a much better estimator of user perceptions (see [86, 87, 51]). The goal in relevance feedback is to adapt the distance measure to match user expectations, by making the search an

interactive process. Here, in each iteration a set of results is retrieved and the user provides feedback on the relevance of each result. If Mahalanobis distance is employed, this is used to update the weight matrix for the next iteration. There are two popular methods for weight adaptation - MARS [88] and MindReader [86]. In MARS, the weight matrix is constrained to be a diagonal matrix. If σ_{ii} is the i^{th} diagonal entry and $\mathcal{Q}^+ = \{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_l\}$ is the set of positive samples returned from the earlier iteration, in the next iteration

$$\sigma_{ii} = \frac{1}{\sqrt{VAR[Q_i^+]}} \tag{3.7}$$

The MindReader approach attempts to estimate the full weight matrix W by minimizing the average distance from the query $\mathbf{q}$ to the elements of $\mathcal{Q}^+$, with the determinant $|W|$constrained to be unity. At the same time, the query vector is also modified. The "optimal" W is found to be

$$W = \alpha C^{-1} \tag{3.8}$$

where C is the sample correlation matrix of $\mathcal{Q}^+$.

However, the MindReader approach can be affected by the singularity of C because of the relatively high dimensionality of the space involved and the low cardinality of $\mathcal{Q}^+$. While the authors of [86] propose to use the Moore-Penrose inverse instead of C^{-1}, perceptual quality in experiments conducted by the authors of [87] have been observed to be poor. Hence, the authors of [87], attempt to combine the MARS and MindReader approaches within the same framework, with the weight matric W constrained to be diagonal if C is singular. If the weights are to be updated in batch fashion (after accumulating several queries) rather than the continuous online fashion, approaches such as the information theoretic approach to learn the new weight matrix [52] or the optimization based methods presented in [89] [90] can be considered. The process of weight adaptation stops when the users are satisfied with the results.

3.4.2 Efficiency of Indexing with Relevance Feedback

Multidimensional search indexes are typically designed assuming a fixed Mahalanobis distance measure that is known in advance. The weight matrix is diagonalized and the data are correspondingly rotated and scaled into a new set of dimensions, prior to indexing. However, in relevance feedback applications, the weight matrix changes with time and renders most standard indexes ineffective and very slow. Clearly, a truly effective relevance feedback application requires a new indexing approach.

The popular VA-File [43] approach partitions the space into hyper-rectangular cells, aligned with the co-ordinate axes. Each dimension is quantized uniformly and the quantization indices are stored of each feature vector in the so called *approximation file*, on the hard-disk. A change in the Mahalanobis weight matrix is equivalent to rotating

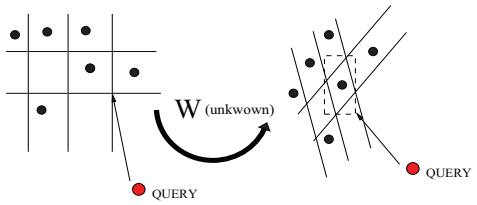

Figure 3.28. VA-File under an Unknown Linear Transformation

and skewing the bounding rectangles into uniform hyper-parallelograms. Since distance bounding to these parallelograms is more complicated and involves $O(d^3)$ computations [91] [92] [93], for dimensionality d, the total number of computations for all points scales as $O(Nd^3)$, where N is the database size. The method of [91] fits minimum bounding rectangles that contain these parallelograms. A new set of distance bounds to these rectangles are evaluated and used in spatial filtering (see Figure 3.28). We note that distance bounds to these rectangles can be efficiently performed and computational complexity only scales as $O(Nd)$. Note that these hyper-rectangles are larger and overlapping, which results in weaker distance bounds and consequently, weaker spatial filtering.

Since we focus on the clustering approach, a natural step would be to investigate

any properties of the hyperplane bound that could allow adaptability to a change in the Mahalanobis weight matrix. We begin by evaluating the point-to-hyperplane distance for an arbitrary Mahalanobis distance measure.

3.5 Generalized Point-to-Hyperplane Distance

Let $d_W(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T W(\mathbf{x}-\mathbf{y})}$ be the distance between any two feature vectors $\mathbf{x}$ and $\mathbf{y}$. Without loss of generality, we assume W is symmetric and positive definite i.e. $d_W(\cdot,\cdot)$ is a metric. Let $H(\mathbf{a}, b) = \{\mathbf{x} : \mathbf{a}^T\mathbf{x} + b = 0\}$ be a hyperplane and $\mathbf{y}$ a point in the space outside of it. Then,

$$d_W(\mathbf{y}, H) = \min_{\mathbf{x}\in H} d_W(\mathbf{x}, \mathbf{y}) = \sqrt{\min_{\mathbf{x}\in H} d_W(\mathbf{x}, \mathbf{y})^2}$$

Using Lagrange multiplier λ, let $J = (\mathbf{x}-\mathbf{y})^T W(\mathbf{x}-\mathbf{y}) + \lambda(\mathbf{a}^T\mathbf{x} + b)$.

$$\begin{aligned}
\frac{\partial J}{\partial \mathbf{x}} = 0 \Rightarrow \mathbf{x}^* - \mathbf{y} &= -\frac{1}{2}\lambda W^{-1}\mathbf{a} \\
\frac{\partial J}{\partial \lambda} = 0 \Rightarrow \mathbf{a}^T\mathbf{x}^* + b &= 0 \\
\Rightarrow \lambda = \frac{2(\mathbf{a}^T\mathbf{y} + b)}{\mathbf{a}^T W^{-1}\mathbf{a}}, \mathbf{x}^* - \mathbf{y} &= \frac{-(\mathbf{a}^T\mathbf{y} + b)W^{-1}\mathbf{a}}{\mathbf{a}^T W^{-1}\mathbf{a}} \\
\Rightarrow (\mathbf{x}^* - \mathbf{y})^T W(\mathbf{x}^* - \mathbf{y}) &= \frac{(\mathbf{a}^T\mathbf{y} + b)^2}{\mathbf{a}^T W^{-1}\mathbf{a}} \\
\Rightarrow d_W(\mathbf{y}, H) = d_W(\mathbf{x}^*, \mathbf{y}) &= \frac{|\mathbf{a}^T\mathbf{y} + b|}{\sqrt{\mathbf{a}^T W^{-1}\mathbf{a}}}
\end{aligned}$$

We note that if W were the identity matrix, then the formula reduces to the known version for Euclidean distance. Next consider two weight matrices W_1 and W_2, it is easy to note that

$$\frac{d_{W_1}(\mathbf{y}, H)}{d_{W_2}(\mathbf{y}, H)} = \sqrt{\frac{\mathbf{a}^T W_2^{-1}\mathbf{a}}{\mathbf{a}^T W_1^{-1}\mathbf{a}}} \tag{3.9}$$

In other words, the ratio of point-to-hyperplane distances under differing weight matrices is *independent* of the point $\mathbf{y}$ (as well as the fixed translation b).

3.6 The Hyperplane Bound Revisited

It is easy to show that for any positive definite W, the shortest path between two points is along the straight line passing through the two points. Now, given a cluster $\mathcal{X}_m$, the query $\mathbf{q}$ and a hyperplane H that lies between the cluster and the query (a "*separating hyperplane*", see Figure 3.2), by simple geometry it is easy to see that for any $\mathbf{x} \in \mathcal{X}_m$

$$\begin{aligned} d_W(\mathbf{q},\mathbf{x}) &\geq d_W(\mathbf{q},H) + d_W(\mathbf{x},H) \\ &\geq d_W(\mathbf{q},H) + \min_{\mathbf{x}\in\mathcal{X}_m} d_W(\mathbf{x},H) \\ &= d_W(\mathbf{q},H) + d_W(\mathcal{X}_m,H) \\ \Rightarrow d_W(\mathbf{q},\mathcal{X}_m) &\geq d_W(\mathbf{q},H) + d_W(\mathcal{X}_m,H) \end{aligned} \tag{3.10}$$

We focus on the second term, $d_W(\mathcal{X}_m, H)$, the "*support*". Had W been known in advance, this could have been evaluated offline and stored. Instead, let us denote the weight matrix used during clustering as W_0. Then, (3.9) implies

$$d_W(\mathcal{X}_m,H) = \sqrt{\frac{\mathbf{a}^T W_0^{-1}\mathbf{a}}{\mathbf{a}^T W^{-1}\mathbf{a}}}\, d_{W_0}(\mathcal{X}_m,H) \tag{3.11}$$

which demonstrates that it is *unnecessary* to reevaluate the support due to change in weight matrix after the clustering phase. Without loss of generality, in subsequent discussion, we will assume that $d_{W0}(\cdot,\cdot)$ is the Euclidean distance, and drop the suffix W_0.

3.6.1 Cluster Distance Bounding

As described in section 3.1.1, if $\mathcal{H}_{sep}$ represents a countably finite set of separating hyperplanes (that lie-between the query $\mathbf{q}$ and the cluster $\mathcal{X}_m$),

$$d_W(\mathbf{q},\mathcal{X}_m) \geq \max_{H\in\mathcal{H}_{sep}} \{d_W(\mathbf{q},H) + d_W(\mathcal{X}_m,H)\} \tag{3.12}$$

The conditions for a hyperplane boundary to be a separating hyperplane can be similarly derived.

3.6.2 Reduced Complexity Hyperplane Bound

For evaluation of the lower-bound of (3.10) and (3.12), we would need to pre-calculate and store $d(H_{mn}, \mathcal{X}_m)$ for all cluster pairs (m, n). With K clusters, there are $K(K-1)$ distances that need to be pre-calculated and stored, in addition to the cluster centroids themselves. The total storage for all clusters would be $O(K^2 + Kd)$. This heavy storage overhead makes the hyperplane bound, in this form, impractical for a very large number of clusters. However, we can loosen the bound in (3.12) as follows:

$$\begin{aligned} d_W(H, \mathcal{X}_m) &= \sqrt{\frac{\|\mathbf{a}\|_2^2}{\mathbf{a}^T W^{-1} \mathbf{a}}} d(\mathcal{X}_m, H)\} \\ &\geq \sqrt{\frac{\|\mathbf{a}\|_2^2}{\mathbf{a}^T W^{-1} \mathbf{a}}} \min_{H \in \mathcal{H}_{sep}} d(\mathcal{X}_m, H)\} \\ \Rightarrow d_W(\mathbf{q}, \mathcal{X}_m) &\geq \max_{\mathcal{H}_{sep}} \{d_W(\mathbf{q}, H) + \sqrt{\frac{\|\mathbf{a}\|_2^2}{\mathbf{a}^T W^{-1} \mathbf{a}}} d_{sep}\} \end{aligned}$$

where $d_{sep} = \min_{H \in \mathcal{H}_{sep}} d(\mathcal{X}_m, H)$. This means that for every cluster $\mathcal{X}_m$ we would only need to store one distance term d_{sep}, thus reducing the total storage to $O(K(d+1))$. For the special case when $d_W(\cdot, \cdot)$ is itself Euclidean, i.e., no weight adaptation, see [94]. For small K, even $\|\mathbf{a}\|_2$, for all cluster boundaries $\mathbf{a}$, can be calculated offline and stored. Even otherwise, we note that it is IO time (and not processor time) which is the bottleneck in query processing.

3.7 Experimental Results

We compared the performance of our index (henceforth referred to as 'VQ-Hyperplane (full)' for full complexity and 'VQ-Hyperplane (reduced)' for reduced storage complexity bounds resp.) with a well-known variant of VA-File [91] that is adapted to leverage relevance feedback. We evaluated the performance of VA-File at various quantization levels (3-12 bits per dimension) and the VQ method for varying numbers of clusters (10-400 clusters). We note that this expands upon and subsumes results presented in [95].

3.7.1 Data-set: CORTINA-Caltech101

This data-set consists of a 1,103,271 element sub-sample of the CORTINA image data-set [7] and consists of 48-dimensional MPEG-7 texture features [8]. Within this we embed 733 images extracted from 11 classes of the CalTech 101 data-set [96] which have strong texture signatures. The classes considered are "accordion", "barrel", "bass", "brain", "beaver", "cougar body", "Leopards", "wild cat", "hedgehog", "platypus", and "soccer ball". These images classes are a-priori unknown to the system and the images are used as queries for performance evaluation. For each query, the 10 nearest neighbors (10NN) were mined. We also assumed a *page size* of 8kB.

3.7.2 Data-set: BIO-RETINA

The data-set BIO-RETINA[9] consists of MPEG-7 texture feature descriptors extracted from 64×64 blocks generated from images of tissue sections of feline retinas as a part of an ongoing project at the Center for Bio-Image Informatics, UCSB. It is 208,506 elements long and 62 dimensional. We also assumed a *page size* of 8kB. The query sets themselves were generated by randomly selecting 100 elements from the relevant data-sets. For each query, the 10 nearest neighbors (10NN) were mined.

3.7.3 CORTINA-CalTech 101 + MARS

In this set of experiments, we evaluated the (diagonal) weight matrix for each query-class according to principles and heuristics established in MARS [88]. We note that with the MARS weights there is *no rotation* of the feature space, since the weight matrix is diagonal.

We notice moderate gains for the VQ-hyperplane methods in the IO performance

[7] See http://cortina.ece.ucsb.edu/
[8] http://scl.ece.ucsb.edu/datasets/CORTINA_HTD_1Million.bin
[9] Download from http://scl.ece.ucsb.edu/datasets/BIORETINA_features.txt

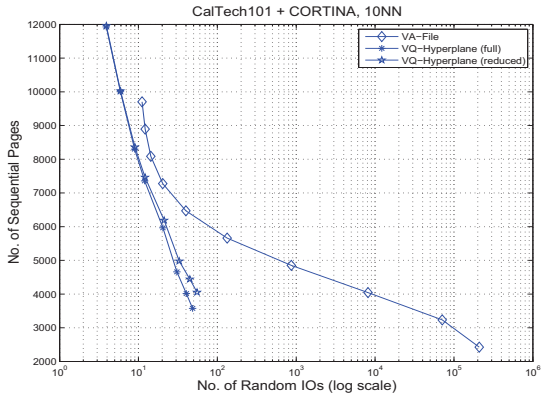

Figure 3.29. IO Performance with MARS (diagonal weight matrix)

(Figure 3.29). For the same number of sequential accesses, random disk IOs are reduced by factors ranging from 1.5X to 200X. But we note that the VA-File requires $\approx$ 100X more pre-processing storage (Figure 3.30) and also $\approx$ 10X higher computational costs (Figure 3.31). This is because the VA-File maintains a separate compressed representation for *each element* of the database, as a result of which $O(N)$ distance computations and storage of $.15N - 0.3N$ are needed, where N is the size of the database. In order to reduce the number of costly random access reads in the VA-File, the quantization resolution in each dimension needs to be increased, which again results in larger approximation files. In contradistinction, the VQ method reduces random IO reads by *reducing* the number of clusters.

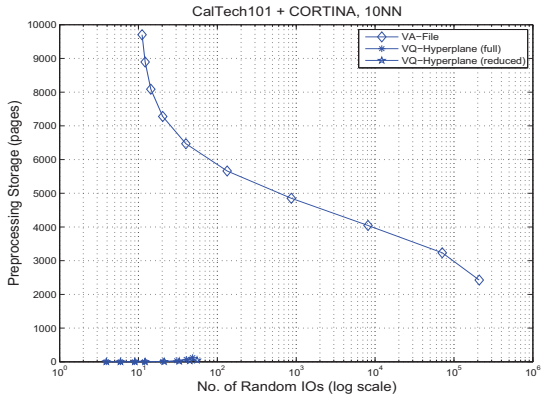

Figure 3.30. Preprocessing Storage with MARS

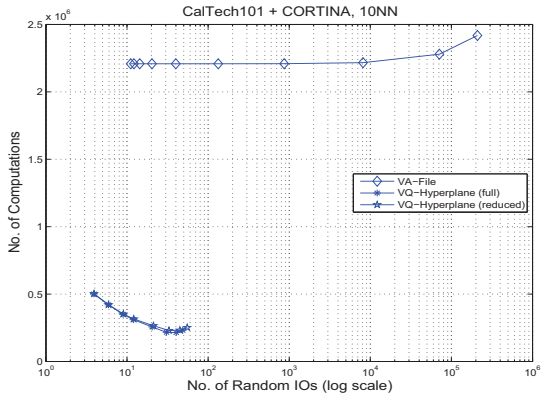

Figure 3.31. Computational Costs with MARS

3.7.4 CORTINA-CalTech 101 + Random Weight Matrix

Here, the weight matrix was modelled as $W = U^T \Lambda U$. The orthonormal matrix U was generated randomly and the eigenvalues were uniformly distributed between 0 and 10. We present results from one such realization of W, that is representative of general performance.

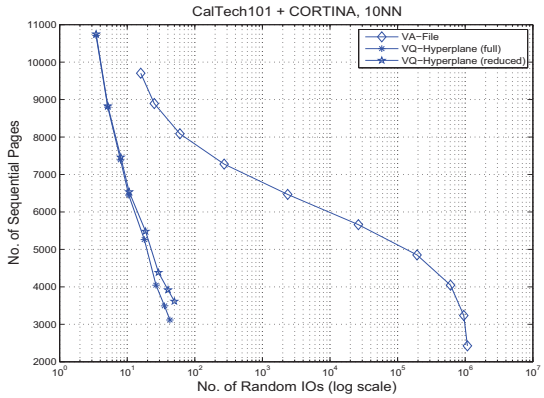

Figure 3.32. IO Performance with Random Weight Matrix

For the random weight matrix, our indexes are able to consistently reduce the number of random IO reads as compared with VA-File, when allowed (roughly) the same number of sequential disk accesses (see Figure 3.32). This is because of the rotation and scaling of the space, which significantly loosens the distance bounds of the VA-File. At 5-bit quantization for the VA-File and 300 clusters for our indexes, we note an $\approx 10^5$X

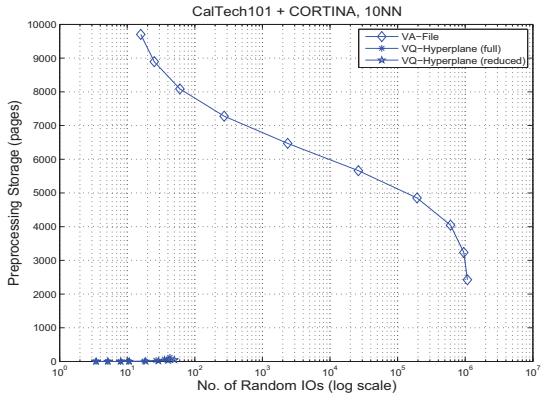

Figure 3.33. Preprocessing Storage with Random Weight Matrix

reduction in costly random IO reads. Clearly, at this quantization level, efficiency of the VA-File in spatial filtering is almost nil i.e. it almost underperforms the sequential scan. The performance degradation from the full complexity to the reduced storage complexity hyperplane bounds is also minimal. Large gains in storage and computational costs were also observed (Figures 3.33 and3.34).

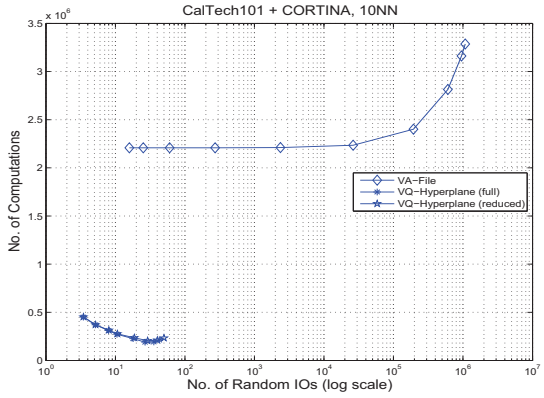

Figure 3.34. Computational Costs with Random Weight Matrix

3.7.5 BIORETINA + Random Weight Matrix

Similar to the previous set of experiments, the weight matrix, typically a correlation matrix [86], was modelled as $W = U^T \Lambda U$. The orthonormal matrix U was generated randomly and the eigenvalues were uniformly distributed between 0 and 10. We present

results from one such realization of W, that is representative of general performance. Here, we evaluated the performance of VA-File at various quantization levels (3-12 bits per dimension) and the VQ method for varying numbers of clusters (10-600 clusters).

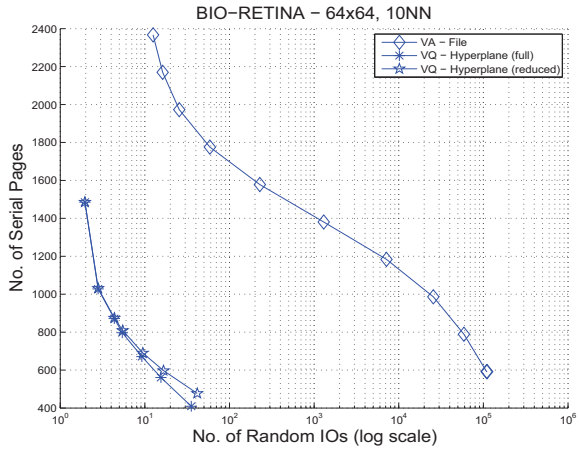

Figure 3.35. IO Performance

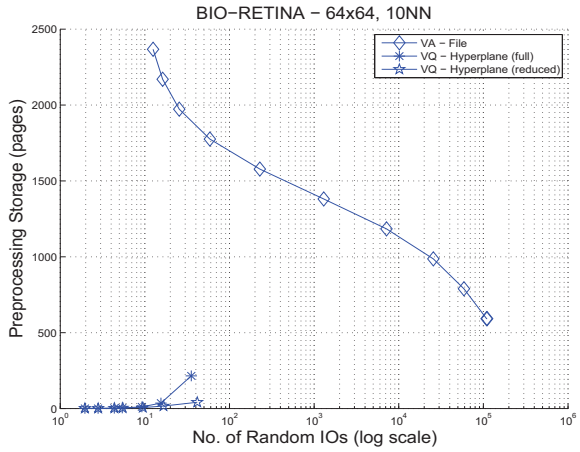

Figure 3.36. Preprocessing Storage

We note that our indexes are able to consistently reduce the number of random IO reads as compared with VA-File, when allowed (roughly) the same number of sequential disk accesses. For BIO-RETINA (Figure 3.35), at 6 bit quantization for VA-File, a nearly $3000X$ reduction in costly random disk accesses is achieved by the vector quantization/clustering approach with 15 clusters. We note that at $K = 600$ clusters (the

last/rightmost operating point), clustering was done in a multi-stage fashion, which explains the sudden "plateau" in storage/computation.

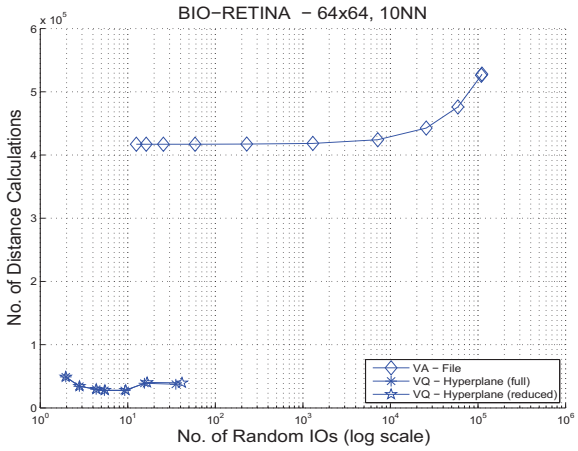

Figure 3.37. Computational Cost

We also note significant ($\approx 10X - 100X$) lower storage and lower computational costs incurred by the VQ method (Figure 3.36 and 3.37).

3.7.6 Approximate Nearest Neighbors: BIO-RETINA

For brevity of presentation, we describe results for only the reduced complexity hyperplane bound with the BIO-RETINA data-set. Similar results are observed with the full complexity hyperplane bound and with the CORTINA-CalTech101 data-set. Our performance metrics are precision (which equals recall for kNN queries) and distance ratio.

Comparison with VA-LOW

Figures 3.38 and 3.39 show the performance of our clustering + red. complexity hyperplane bound retrieval and VA-LOW [72], a variant of the VA-Files that can return approximate NNs. In the VA-LOW, the search in the second phase is stopped once sufficient vectors have been visited to assure a certain precision.

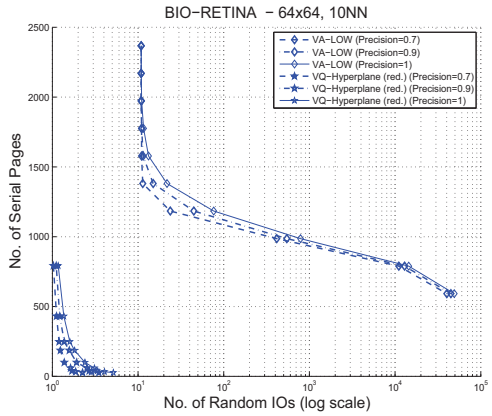

Figure 3.38. IO Performance under Precision Metric

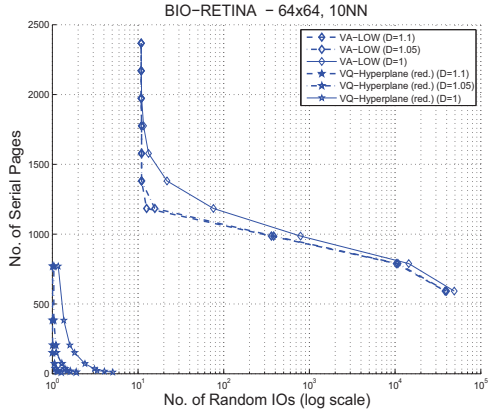

Figure 3.39. IO Performance under Distance Ratio (D) Metric

We note substantial gains over VA-LOW. Even the very first cluster returns high-precision results and by creating 300-500 clusters, the number of sequential IOs is reduced. Moreover, we observe that 100 % precision results i.e. the retrieval of *exact* nearest neighbors, is possible with just the first few disk IOs (even though there is no guarantee that the exact NNs have been found). On the other hand, in VA-LOW several random and sequential disk access are necessary.

Comparison with Naive Cluster Retrieval

Next, we compare performance against naive cluster retrieval i.e. retrieval of clusters in order of distances to the centroids. The search is stopped at various stages and the precision (or distance ratio) is measured. This was performed at two levels of clustering with $K = 600$ clusters and with $K = 1200$ clusters.

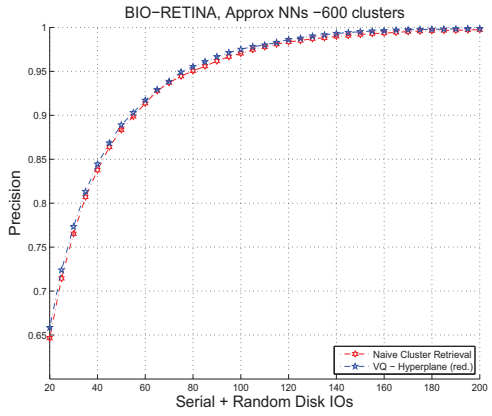

Figure 3.40. Precision, 600 clusters

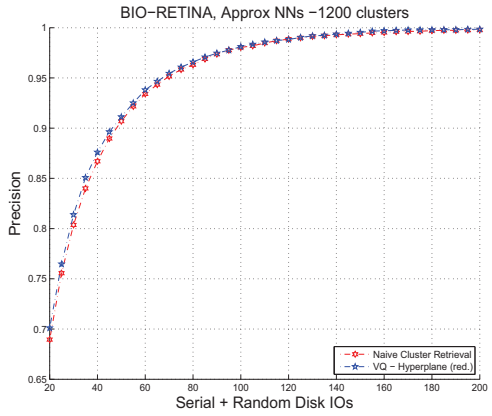

Figure 3.41. Precision, 1200 clusters

We note that when the performance measure is precision, the retrieval performance of both schemes is very similar, with minor gains for hyperplane based cluster retrieval. However, if the performance measure is the distance ratio, we note that the naive cluster retrieval is worse with disk accesses increased by as much as 20%, when compared with

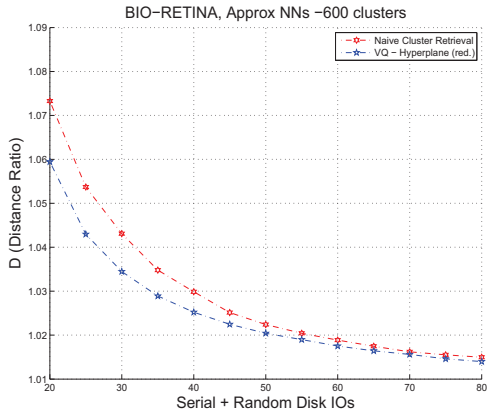

Figure 3.42. Distance Ratio (D) Metric, 600 clusters

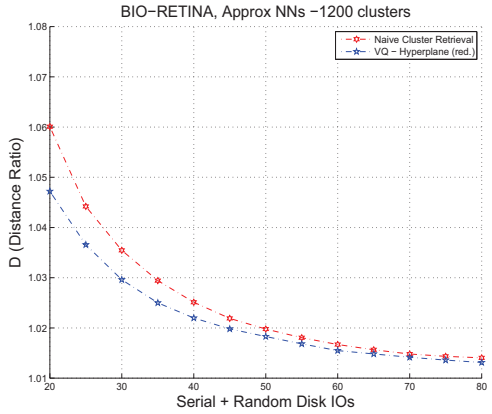

Figure 3.43. Distance Ratio (D) Metric, 1200 clusters

hyperplane bound based retrieval for roughly same accuracy (D). This is because, with the hyperplane bound, better selection of clusters is possible.

3.8 Enhanced Cluster Distance Bounds

In this section, we present a method that improves upon the hyperplane based cluster-distance bounds. We first note that both Euclidean and (positive definite) Mahalanobis distance measures are metrics i.e. they are symmetric, non-negative, obey the triangle inequality and two vectors are equal if their Euclidean or Mahalanobis distance is zero.

Now, for query $\mathbf{q}$ consider the function

$$f_{\mathbf{q}}(\mathbf{x}) = \|\mathbf{x} - \mathbf{q}\|_2 \tag{3.13}$$

By applying the triangle inequality,

$$\begin{aligned} f_{\mathbf{q}}((1-\lambda)\mathbf{x} + \lambda\mathbf{y}) &= \|(1-\lambda)\mathbf{x} + \lambda\mathbf{y} - \mathbf{q}\| \\ &= \|(1-\lambda)(\mathbf{x} - \mathbf{q}) + \lambda(\mathbf{y} - \mathbf{q})\| \\ &\leq (1-\lambda)\|\mathbf{x} - \mathbf{q}\| + \lambda\|\mathbf{y} - \mathbf{q}\| \\ &= (1-\lambda).f_{\mathbf{q}}(\mathbf{x}) + \lambda.f_{\mathbf{q}}(\mathbf{y}) \end{aligned}$$

i.e. $f_{\mathbf{q}}$ is a convex function. Convex functions have the useful property that when optimized over convex sets, if a local minimum exists, then it is a global minimum [97]. If the function is strictly convex, then there exists at most one minimum. The optimization of convex functions over convex sets has been well studied and under some conditions shown to be of complexity polynomial in d, the number of dimensions. In the case of l_2 norm minimization with linear constraints, it is of typically $O(d^3)$ complexity [93] [92].

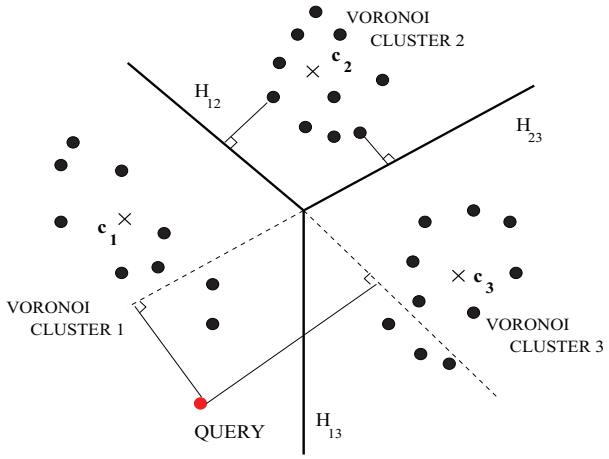

Figure 3.44. Cluster Bounded by Multiple Hyperplanes

Now, when considers the cluster procedure, the cluster is bounded by multiple hyperplanes (the cluster boundaries, see Figure 3.44). Let the rows of matrix A represent

the hyperplane normals and while the rows of column vector $\mathbf{b}$ represent the distance of the hyperplanes from the origin. Hence, if $\mathcal{X}_m$ is a cluster, then it is easy to see that

$$\forall \mathbf{x} \in \mathcal{X}_m, A\mathbf{x} \leq \mathbf{b} \tag{3.14}$$

The cluster supports can be viewed as the distance the hyperplanes need to be "moved" till they just touch the cluster. Hence, by suitable additions/subtractions to the rows of $\mathbf{b}$, it is also possible to incorporate information about the cluster supports.

Define $\mathcal{A} = \{\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\}$. Clearly,

$$\mathcal{X}_m \subseteq \mathcal{A} \tag{3.15}$$

and $\mathcal{A}$ is a convex (intersection of half-spaces) set (see [97], Chapter 2). Now, for query $\mathbf{q}$, the true query-cluster distance is

$$d(\mathbf{q}, \mathcal{X}_m) = \min_{\mathbf{x} \in \mathcal{X}_m} d(\mathbf{x}, \mathbf{q}) \tag{3.16}$$

By combining with (3.15),

$$d(\mathbf{q}, \mathcal{X}_m) \geq \min_{\mathbf{x} \in \mathcal{A}} d(\mathbf{x}, \mathbf{q}) = \min_{\mathbf{x} \in \mathcal{A}} f_{\mathbf{q}}(\mathbf{x}) \tag{3.17}$$

This minimization of $f_{\mathbf{q}}$ over $\mathcal{A}$ provides another lower bound on query-cluster distance.

3.8.1 Interpretation

The minimization of $f_{\mathbf{q}}$ over the convex set $\mathcal{A}$ is equivalent to bounding the cluster with a convex polytope (i.e. $\mathcal{A}$) and searching for the point closest to the query on the polytope i.e. we now search over all (separating) hyperplane surfaces *and their intersections*. Previously in the hyperplane bound, we only estimate distances to the separating hyperplanes surfaces and hence, this new bound improves upon the hyperplane bounds.

Since $f_{\mathbf{q}}$ is a convex function, its minimization over a convex set can be done very efficiently (see [93] [92]). The computational costs do not grow with size of the database but only with the dimensionality. It is important to note that in a d-dimensional space, K hyperplanes in general position have $O(K^d)$ intersections and hence, naively searching over all intersections in a high dimensional space is not feasible.

Mahalanobis distances, with a positive definite Mahalanobis matrix W, are also metrics. Therefore, by suitably modifying $f_{\mathbf{q}}$, the convexity property of $f_{\mathbf{q}}$ can be similarly shown to hold for Mahalanobis distances. We notice that the clustering procedure, and hence, the hyperplanes, can be defined independent of W, which allows cluster distance-bound calculation even as W changes. Hence, we improve upon the hyperplane bounds even for the relevance feedback application.

3.8.2 Experimental Results

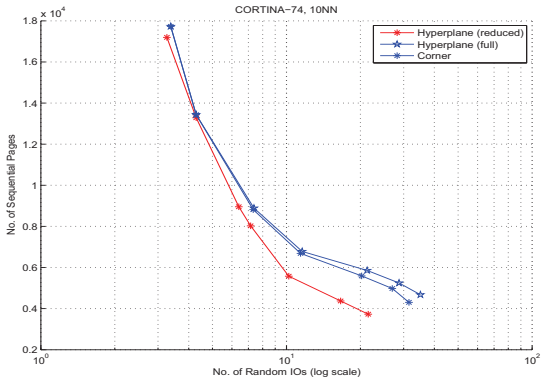

Figure 3.45. Euclidean Distance Indexing

We show results from both Euclidean distance and relevance feedback indexing (see Figures 3.45, 3.46, 3.47), when tested on the two CORTINA feature sets. We refer to the convex optimization based approach as the 'Corner' and hyperplane methods as 'Hyperplane (full)' and 'Hyperplane (reduced)' for the full and reduced complexity hyperplane bounds.

In the case of the Euclidean distance indexing, we notice a further reduction of ≈2X in costly random IOs, given roughly the same amount of sequential IOs, over the full complexity hyperplane bound and ≈3X over the reduced complexity hyperplane bound.

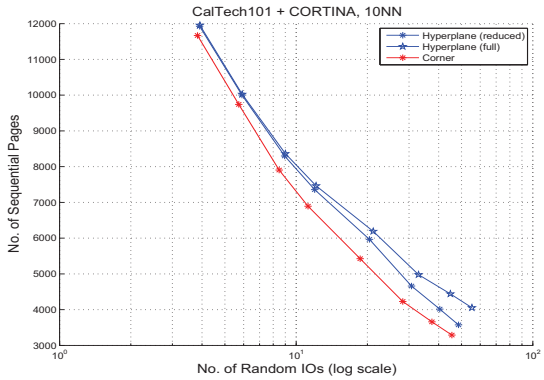

Figure 3.46. Relevance Feedback Indexing: MARS

For relevance feedback indexing with a diagonal weight matrix (the MARS approach [88]), the reductions in random IOs range from 1.3X-1.6X (see Figure 3.46). With a random weight matrix (Figure 3.47), the random IO reductions are in the range 2.8X-4X.

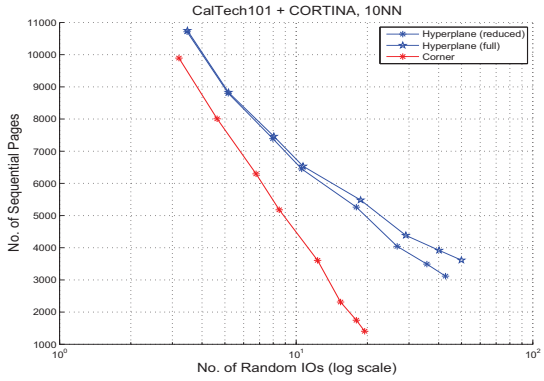

Figure 3.47. Relevance Feedback Indexing: Random Weight Matrix

Chapter 4

Fusion Coding of Correlated Sources

This chapter develops a practical framework for the fusion coding of correlated sources in a sensor network-database. The goal is to design a system that can exploit correlations and compress multiple data sources (data-streams) efficiently, and yet have the ability to retrieve information (bits) that are relevant only to a queried subset of the sources (data-streams). Future queries are unknown, but we assume access to a query distribution (or a training set of sample queries). The typical application is for handling data generated in sensor networks, which generate huge amounts of correlated data and where users would be interested in specific regions of the sensor field.

In subsequent discussion, we shall focus attention on real-valued i.i.d sources $X_m, \forall m$. Any practical signal storage scheme would need to quantize and compress the data before storage, hence leading to some error or *distortion*. Given query $\mathbf{q}$, the *reconstruction* distortion is measured as

$$d_{\mathbf{q}}(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{m=1}^{M} q_m d_m(x_m, \hat{x}_m), \tag{4.1}$$

where

$$d_m : \mathbb{R} \times \mathbb{R} \to [0, \infty). \tag{4.2}$$

Hereafter, we will specialize to the *squared error distortion* measure, i.e. distortion

measur

(4.3)

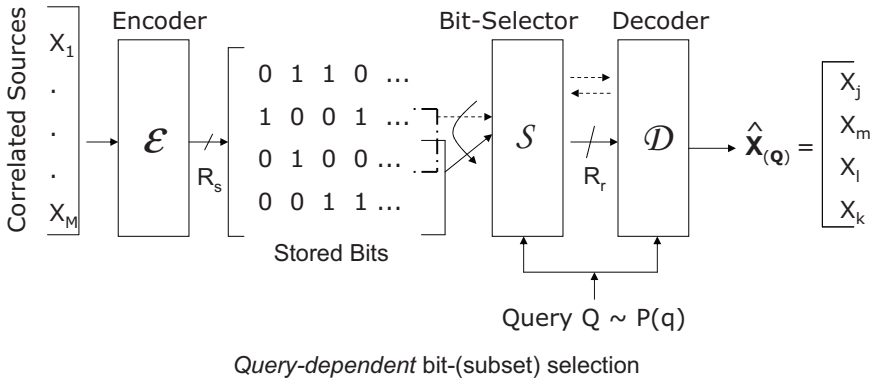

Figure 4.1. Proposed Fusion Coder

4.1 Fusion Coder Formulation

We propose a *fusion coding* framework to optimize the fusion storage-selective retrieval of correlated sources. A block diagram, representative of our Fusion Coder (FC), is given in Figure 4.1. The fusion coder is composed of three modules: encoder, bit (subset)-selector and decoder. We define the encoder by the function

$$\mathcal{E} : \mathbb{R}^M \rightarrow \mathcal{I} = \{0,1\}^{R_s} \tag{4.4}$$

which compresses the M-dimensional input vector $\mathbf{x}$, representing the M sources, to R_s bits at each instant.

The bit (subset) selector is the mapping

$$\mathcal{S} : \mathcal{Q} \rightarrow \mathcal{B} = 2^{\{1,...,R_s\}} \tag{4.5}$$

where $\mathcal{Q} \subseteq \{0,1\}^M$ represents the domain-set of queries and $\mathcal{B}$ is power set (set of all subsets) of the set $\{1,...,R_s\}$. This mapping determines which of the stored bits to retrieve for a given query $\mathbf{q}$. Clearly, $\mathcal{S}(\mathbf{q}) \subseteq \{1,\ldots,R_s\}, \forall \mathbf{q}$.

For each subset of bits $\mathbf{e}$ that can be retrieved, an estimate of all the sources is formed by the decoder

$$\mathcal{D} : \mathcal{I} \times \mathcal{B} \rightarrow \hat{\mathcal{X}} \tag{4.6}$$

where $\hat{\mathcal{X}} \subset \mathbb{R}^M$ is the corresponding codebook. The average distortion for a specific query $\mathbf{q}$ is

$$D_{\mathbf{q}} = E[d_{\mathbf{q}}(\mathbf{X}, \mathcal{D}(\mathcal{E}(\mathbf{X}), \mathcal{S}(\mathbf{q})))] \tag{4.7}$$

where $E[\ldots]$ denotes statistical expectation, and the distortion averaged across all queries is $D = \sum_{\mathbf{q}\in\mathcal{Q}} P(\mathbf{q}) D_{\mathbf{q}}$. In practice, since we have access to the database $\mathcal{X}$ itself, we use it as a training set and replace the expectation operator $E[\ldots]$ by a simple average, evaluated across the database $\mathcal{X}$. Hence, the distortion is evaluated as

$$D = \sum_{\mathbf{q}\in\mathcal{Q}} P(\mathbf{q}) \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}\in\mathcal{X}} d_{\mathbf{q}}(\mathbf{x}, \hat{\mathbf{x}}), \tag{4.8}$$

Noting that $R_{\mathbf{q}} = R_{\mathcal{S}(\mathbf{q})} = |\mathcal{S}(\mathbf{q})|$, the average retrieval rate is,

$$R_r \;\; = \;\; \sum_{\mathbf{q}} P(\mathbf{q}) R_{\mathbf{q}} = \sum_{\mathbf{q}} P(\mathbf{q}) |\mathcal{S}(\mathbf{q})| \tag{4.9}$$

Given M correlated sources and a storage constraint R_s, the designer aims to minimize distortion (or maximize quality) and at the same time minimize retrieval rate. Hence, we are interested in optimizing the trade-off between distortion and retrieval rate, given a fixed amount of storage. This is equivalent to minimizing the Lagrange sum of distortion and retrieval rate

$$\min_{\mathcal{E},\mathcal{S},\mathcal{D}} J = \min_{\mathcal{E},\mathcal{S},\mathcal{D}} D(R_s) + \lambda R_r(R_s), \lambda \geq 0 \tag{4.10}$$

for different λ.

4.2 Necessary Conditions for Optimality

The Lagrangian cost J can be rewritten as

$$J = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}} \sum_{\mathbf{q}} P(\mathbf{q}) d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathcal{E}(\mathbf{x}), \mathcal{S}(\mathbf{q}))) + \lambda \sum_{\mathbf{q}} P(\mathbf{q}) |\mathcal{S}(\mathbf{q})|$$

Optimal Encoder : From the above equation, the optimal encoding index $\mathcal{E}(\mathbf{x})$ for input vector $\mathbf{x}$ is

$$\mathcal{E}(\mathbf{x}) = \arg\min_{\mathbf{i}\in\mathcal{I}} \sum_{\mathbf{q}} P(\mathbf{q}) d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathbf{i}, \mathcal{S}(\mathbf{q}))), \forall \mathbf{x}$$

It is easy to see the resemblance to the "nearest neighbor" condition in quantizer design [62].

Optimal Bit-Selector : Similarly, the best set of bits to be retrieved for a particular query should be the one that minimizes the Lagrangian sum of the distortion measure $d_{\mathbf{q}}(.,.)$, averaged across the training set, and the query-specific retrieval rate $R_{\mathbf{q}} = |\mathcal{S}(\mathbf{q})|$ i.e.

$$\mathcal{S}(\mathbf{q}) = \arg\min_{\mathbf{e}\in\mathcal{B}} \{\frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}} d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathcal{E}(\mathbf{x}), \mathbf{e})) + \lambda|\mathbf{e}|\}, \forall \mathbf{q} \tag{4.11}$$

Optimal Decoder : Let $\mathbf{i} \in \mathcal{I}$ be a typical encoding index and $\mathbf{e} \in \mathcal{B}$ represent some subset of bits. We use $\mathbf{i_e}$, to denote the sub-index extracted from $\mathbf{i}$ by retrieving the bits in the positions indicated by $\mathbf{e}$ and $\mathcal{D}(\mathbf{i}, \mathbf{e})$ to denote the corresponding codevector. Clearly, the choice of codevector does not affect the rate component of the Lagrangian cost J.

Let $F_{\mathbf{i},\mathbf{e}} = \{\mathbf{x} : (\mathcal{E}(\mathbf{x}))_{\mathbf{e}} = \mathbf{i_e}\}$ and $E_{\mathbf{e}} = \{\mathbf{q} : (\mathcal{S}(\mathbf{q})) = \mathbf{e}\}$. We can rewrite the distortion as

$$\begin{aligned} D &= \frac{1}{|\mathcal{X}|} \sum_{\mathbf{i},\mathbf{e}} \sum_{m} \sum_{\mathbf{q}\in E_{\mathbf{e}}} \sum_{\mathbf{x}\in F_{\mathbf{i},\mathbf{e}}} P(\mathbf{q}) q_m (x_m - \mathcal{D}(\mathbf{i}, \mathbf{e})_m)^2 \\ &= \frac{1}{|\mathcal{X}|} \sum_{\mathbf{i},\mathbf{e}} \sum_{m} w_m \sum_{\mathbf{x}\in F_{\mathbf{i},\mathbf{e}}} (x_m - \mathcal{D}(\mathbf{i}, \mathbf{e})_m)^2 \end{aligned} \tag{4.12}$$

where $w_m = \sum_{\mathbf{q}\in E_{\mathbf{e}}} P(\mathbf{q}) q_m$.

By setting to zero, the partial derivatives of J w.r.t $\mathcal{D}(\mathbf{i}, \mathbf{e}), \forall \mathbf{i}, \mathbf{e}$, we find the optimal decoder to be

$$\mathcal{D}(\mathbf{i}, \mathbf{e}) = \frac{1}{|F_{\mathbf{i},\mathbf{e}}|} \sum_{\mathbf{x}\in F_{\mathbf{i},\mathbf{e}}} \mathbf{x}, \forall \mathbf{i}, \mathbf{e} \tag{4.13}$$

We note in passing that this is analogous to the "centroid" rule in quantizer design [62].

4.2.1 Algorithm for Fusion Coder Design

Since we are considering the storage and retrieval of signals from a database, the signals from all sources are already available. Hence, while one would use the entire database as a training set, it is equally important to note that the database is also the *test set*. A natural design algorithm is to iteratively enforce each of the necessary conditions for optimality (derived in the preceding sections), till a convergence condition is satisfied. In effect, the algorithm just partitions the elements of the training set and the storage bits into different groups, and for a finite sized training set and a finite storage rate, there exist only a finite number of set partitions. At each step of the optimization, a set of parameters are chosen to minimize the Lagrangian cost and hence, with every iteration, the cost is non-increasing. Therefore, the algorithm is guaranteed to converge in a finite number of iterations. It is to be noted that since the Lagrangian cost surface is non-convex and has multiple local optima, iterative design would be initialization dependent and may *not* lead to a *globally optimal* solution.

4.3 Complexity of Design

At the encoder, the search for the optimal encoding index for each element $\mathbf{x}$ in the training set/database $\mathcal{X}$, involves 2^{R_s} distance evaluations of the form $\sum_{\mathbf{q}} P(\mathbf{q}) d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathbf{i}, \mathcal{S}(\mathbf{q})))$. This implies a net complexity cost $O(2^{R_s}|\mathcal{X}||\mathcal{Q}|M)$ additions and multiplications in encoder optimization. Codebook optimization, on the other hand, involves computation of averages of different subsets of the training set and this complexity grows as $O(|\mathcal{X}|)$, for each codevector. Additionally, the total number of codevectors that need to be maintained is $\sum_{k=1}^{R_s} \binom{R_s}{k} 2^k = 3^{R_s} - 1$ and hence codebook update involves $O(3^{R_s}|\mathcal{X}|M)$ operations. The optimization of the bit(-subset) selector is effectively a search for the best subset among the set of $2^{R_s} - 1$ possible subsets, for each query $\mathbf{q}$. This implies that the complexity of this step grows as $O(2^{R_s}|\mathcal{X}||\mathcal{Q}|M)$. We also note that the storage

complexity of the bit-selector, which is just a look-up table, grows as $O(|\mathcal{Q}|)$. Clearly, the complexity of FC design scales linearly with the size of the database, the query set and the number of sources but exponentially with the R_s.

4.3.1 Complexity of Deployment

Once the system has been designed, the encoding of the database has been completed. Hence, during usage/deployment, only the decoding is performed. For each query, the optimal subsets of encoded bits are retrieved and the relevant sources are reconstructed. However, if only a small part of the database was used to train the system, either to speed up training or because these entries were unavailable during the FC design phase, any remaining/new entries would of course need to be encoded (entailing the corresponding encoder usage complexity).

4.3.2 Complexity Reduction Strategies

We note that the query-set $\mathcal{Q}$ could be very large as could be the number of sources considered and the database itself. A first attempt at reducing complexity would be to limit the training set to be a statistically representative subset, and not the entire database. But the design complexity may still be un acceptable as it is a product of the sizes of the training set, query set and the number of sources. In this section, we exploit properties of the optimal FC to further reduce the design complexity.

Encoding with an Average Codebook

At a first glance, the assignment of the optimal encoding index to each input vector $\mathbf{x}$ (see (5.2.1)) involves $O(|\mathcal{Q}|2^{R_s})$ operations. We try to exploit properties of the squared-error distortion measure to reduce encoding complexity. Noting that $d_{\mathbf{q}}(\mathbf{x}, \mathbf{0}) =$

$\sum_m q_m x_m^2$, it is easy to see

$$\begin{aligned} d_{\mathbf{q}}(\mathbf{x},\mathbf{y}) &= d_{\mathbf{q}}(\mathbf{x},\mathbf{0}) - 2\sum_m q_m x_m y_m + d_{\mathbf{q}}(\mathbf{y},\mathbf{0}) \\ &= d_{\mathbf{q}}(\mathbf{x},\mathbf{0}) - 2\mathbf{x}^T W_{\mathbf{q}} \mathbf{y} + d_{\mathbf{q}}(\mathbf{y},\mathbf{0}) \end{aligned}$$

where the matrix $W_{\mathbf{q}}$ is diagonal, with $W_{\mathbf{q},m,m} = q_m$ This implies that

$$\begin{aligned} \sum_{\mathbf{q}} P(\mathbf{q}) d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathbf{i}, \mathcal{S}(\mathbf{q}))) &= \sum_{\mathbf{q}} P(\mathbf{q}) d_{\mathbf{q}}(\mathbf{x},\mathbf{0}) - 2\mathbf{x}^T \sum_{\mathbf{q}} P(\mathbf{q}) W_{\mathbf{q}} \mathcal{D}(\mathbf{i}, \mathcal{S}(\mathbf{q})) \\ &\quad + \sum_{\mathbf{q}} P(\mathbf{q}) d_{\mathbf{q}}(\mathcal{D}(\mathbf{i}, \mathcal{S}(\mathbf{q})), \mathbf{0}) \\ &= \sum_{\mathbf{q}} P(\mathbf{q}) d_{\mathbf{q}}(\mathbf{x},\mathbf{0}) - 2\mathbf{x}^T \bar{\mathbf{c}}(\mathbf{i}) + \alpha_{\mathbf{i}} \end{aligned}$$

The first term is common to all encoding indices and hence need not be computed. The second term could be viewed as an inner (dot) product of $\mathbf{x}$ with a *query-averaged* codevector $\bar{\mathbf{c}}(\mathbf{i}) = \sum_{\mathbf{q}} P(\mathbf{q}) W_{\mathbf{q}} \mathcal{D}(\mathbf{i}, \mathcal{S}(\mathbf{q}))$, where and the last term a constant, $\alpha_{\mathbf{i}}$, independent of $\mathbf{x}$. $\alpha_{\mathbf{i}}$ and $\bar{\mathbf{c}}(\mathbf{i})$ can be computed in a separate step prior to the encoding of all $\mathbf{x}$. Thus we obtain the faster encoding rule

$$\mathcal{E}(\mathbf{x}) = \arg\min_{\mathbf{i}\in\mathcal{I}} \alpha_{\mathbf{i}} - 2\bar{\mathbf{c}}(\mathbf{i})^T \mathbf{x}, \forall \mathbf{x} \tag{4.14}$$

The new encoder design/operation is now of $O(2^{R_s}|\mathcal{X}|M + 2^{R_s}|\mathcal{Q}|M) \approx O(2^{R_s}|\mathcal{X}|M)$ complexity, since the computation of the average codebook is a one-time affair prior to encoding the entire database (training set).

Recursive Codevector Update

Given R_s storage bits, there are $2^{R_s} - 1$ codebooks, one for each non-empty subset of bits. We denote the codevector based on all R_s encoded bits for index $\mathbf{j}$ as $\hat{\mathbf{x}}_{\mathbf{j}} = \mathcal{D}(\mathbf{j}, \{1, ..., R_s\})$,

Let $F_{\mathbf{i},\mathbf{e}} = \{\mathbf{x} : (\mathcal{E}(\mathbf{x}))_{\mathbf{e}} = (\mathbf{i})_{\mathbf{e}}\}$, $G_{\mathbf{j}} = \{\mathbf{x} : \mathcal{E}(\mathbf{x}) = \mathbf{j}\}$ and $H_{\mathbf{i},\mathbf{e}} = \{\mathbf{j} \in \mathcal{I} : \mathbf{j}_{\mathbf{e}} = \mathbf{i}_{\mathbf{e}}\}$. It is easy to see that

$$F_{\mathbf{i},\mathbf{e}} = \bigcup_{\mathbf{j}\in H_{\mathbf{i},\mathbf{e}}} G_{\mathbf{j}} \tag{4.15}$$

and that $G_{\mathbf{j}} \bigcap G_{\mathbf{k}} = \phi, \forall \mathbf{j} \neq \mathbf{k}$. Consequently

$$\sum_{\mathbf{x} \in F_{\mathbf{i},\mathbf{e}}} \mathbf{x} = \sum_{\mathbf{j} \in H_{\mathbf{i},\mathbf{e}}} \sum_{\mathbf{x} \in G_{\mathbf{j}}} \mathbf{x} \tag{4.16}$$

$$|F_{\mathbf{i},\mathbf{e}}| = \sum_{\mathbf{j} \in H_{\mathbf{i},\mathbf{e}}} |G_{\mathbf{j}}| \tag{4.17}$$

But (4.13) also implies

$$\hat{\mathbf{x}}_{\mathbf{j}} = \frac{1}{|G_{\mathbf{j}}|} \sum_{\mathbf{x} \in G_{\mathbf{j}}} \mathbf{x} \tag{4.18}$$

Hence,

$$\mathcal{D}(\mathbf{i}, \mathbf{e}) = \frac{1}{|F_{\mathbf{i},\mathbf{e}}|} \sum_{\mathbf{x} \in F_{\mathbf{i},\mathbf{e}}} \mathbf{x} = \frac{1}{|F_{\mathbf{i},\mathbf{e}}|} \sum_{\mathbf{j} \in H_{\mathbf{i},\mathbf{e}}} |G_{\mathbf{j}}| \hat{\mathbf{x}}_{\mathbf{j}}$$

In other words, codevector update is effectively the *weighted (vector) average* of the corresponding codevectors obtained by extracting all the stored bits. Hence, codebook update can be performed recursively starting off from the updates of $\mathcal{D}(\mathbf{j}, \{1, ..., R_s\}), \forall \mathbf{j}$ and is now of $O(2^{R_s}|\mathcal{X}|M + (3^{R_s} - 2^{R_s})M) \approx O(2^{R_s}|\mathcal{X}|M)$ complexity. Note, that the storage complexity is reduced from $O(3^{R_s})$ to $O(2^{R_s})$ codevectors. We store $\hat{\mathbf{x}}_{\mathbf{j}}$ and $|G_{\mathbf{j}}|, \forall \mathbf{j}$ and extract all other codevectors by appropriate averaging.

Reduced Complexity Bit-Selector Optimization

The optimal bit-selector satisfies

$$\mathcal{S}(\mathbf{q}) = \arg\min_{\mathbf{e} \in \mathcal{B}} \{ \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}} d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathcal{E}(\mathbf{x}), \mathbf{e})) + \lambda |\mathbf{e}| \}, \forall \mathbf{q}$$

Let $D_m(\mathbf{e}) = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}} (x_m - \mathcal{D}(\mathcal{E}(\mathbf{x}), \mathbf{e})_m)^2, \forall m, \forall \mathbf{e} \subseteq \{1, ..., M\}$. Now, by interchanging the order of summation

$$\mathcal{S}(\mathbf{q}) = \arg\min_{\mathbf{e} \in \mathcal{B}} \sum_{m} q_m D_m(\mathbf{e}) + \lambda |\mathbf{e}|, \forall \mathbf{q} \tag{4.19}$$

which implies that we could compute $D_m(\mathbf{e}), \forall m, \forall \mathbf{e}$ in a separate step and use this result in finding the optimal $\mathcal{S}(\mathbf{q}), \forall \mathbf{q}$. Hence, the complexity of bit-selector optimization reduces to $O(2^{R_s}|\mathcal{X}|M + |\mathcal{Q}|M) \approx O(2^{R_s}|\mathcal{X}|M)$.

4.4 Initialization Strategies

We note that there are two mappings that require initialization - the codebook $\hat{\mathcal{X}}$ and the bit-selector $\mathcal{S}$. As described in section 4.2.1, the FC design is iterative and hence, dependent on initialization. Additionally, the cost surface is non-convex and riddled with local minima. Therefore, for any given λ, FC design should be performed with different (possibly random) initializations, in order to avoid poor local minima. In this we describe some initialization heuristics that would help avoid some poor minima.

4.4.1 Computation of Operational Retrieval Rate- Distortion Curve

Suppose for some λ, a good codebook and bit-selector are known. This can be a good initialization point for other (R_r, D) points. For an incrementally different value of lambda, we start off with the same codebook, iteratively optimize the bit-selector, the encoder and decoder (in that order) till convergence. Alternatively, we could retain the same bit-selection and iteratively optimize the encoder, decoder and bit-selector (in that order). This process could be used to gradually compute the entire rate distortion curve.

Now, for an arbitrary λ, it is unclear how best to initialize the bit-selection and codebooks. However, there are two settings where good heuristics for bit-selector initialization are possible. This would alleviate initialization issues to some extent, even though multiple runs with random codebook initialization could still be necessary to obtain good results.

4.4.2 Retrieve All Bits or $\lambda = 0$

We first consider the special case when $\lambda = 0$. This implies Lagrangian cost is solely composed of distortion and that the penalty for bit retrieval is zero. In other words, the optimal bit-selector setting is to *retrieve all compressed bits*. This initialization setting would work for *all* query sets.

4.4.3 Retrieve Only One Bit or $\lambda = \infty$

Next, we consider the case when $\lambda = \infty$ (or in practice, a very large value). This implies that the Lagrangian cost is dominate by R_r, while distortion plays almost no role. Since the FC is constrained to retrieve at least one bit, the bit-selector setting must do exactly that i.e. retrieve exactly one bit for any query. Since there are R_s encoding bits, it is necessary to partition the query set into R_s groups, where each group maps to the same compressed bit. This partitioning of the query set is necessary so that all allowed R_s bits are used during encoding. We note that such partitioning of the query set would clearly be possible in some query distributions, such as those with multiple modes, while for others, this partitioning may not be very clear.

4.5 Adapting to Large/Incomplete Query Training Sets

While the fusion coding framework is optimal, we note that the bit-selector is a look-up table that grows with the size of the query-set. In principle, the query-set might be very large. For example, if $M = 100$ sources and suppose any query of size 20 can be requested, then $|\mathcal{Q}| = \binom{100}{20} \approx 10^{20}$, which would impose an unbearable storage requirement. In an extreme case, the query-set may be the entire distribution i.e. $|\mathcal{Q}| = 2^M - 1$. Even otherwise, the query-set may change after training i.e. a different

set of queries (nevertheless drawn from the same distribution) might be encountered during operation. In either case, the queries need to be classified (grouped), where all queries in a group share the same bit-selection (combination of bits).

Let the allowed number of groupings be L. We define the query-classifier as the mapping

$$\mathcal{C} : \mathcal{Q} \rightarrow \mathcal{L} = \{1, \ldots, L\} \tag{4.20}$$

$\mathcal{C}$ defines L disjoint partitions of the query-space $\{B_l\}_{l=1}^{L}$ such that

$$B_l = \{\mathbf{q} : \mathcal{C}(\mathbf{q}) = l\}, \forall l = 1, \ldots, L \tag{4.21}$$

$$\Rightarrow \bigcup_{l=1}^{L} B_l = \mathcal{Q}, B_l \bigcap B_m = \phi, \forall l \neq m \tag{4.22}$$

The bit-selector is modified to be the mapping

$$\mathcal{S} : \mathcal{L} \rightarrow \mathcal{B} = 2^{\{1,\ldots,R_s\}} \tag{4.23}$$

where the notations have the usual meaning. The Lagrangian cost to be optimized is

$$J = \sum_{l} \sum_{\mathbf{q} \in B_l} P(\mathbf{q}) \{ \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}} d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathcal{E}(\mathbf{x}), \mathcal{S}(l))) + \lambda |\mathcal{S}(l)| \}$$

Now, $\mathcal{C}$ must be learnt from the available (training) set of queries and must classify accurately an unseen query test-set. This might require some structural constraint to be imposed on $\mathcal{C}$, such as the nearest neighbor classifier, nearest prototype classifier, decision tree etc. This structure can be either learnt during the optimization of all mappings $\mathcal{E}, \mathcal{D}, \mathcal{S}$ or learnt offline (after which $\mathcal{E}, \mathcal{D}, \mathcal{S}$ would need to be re-optimized). In subsequent sections, we shall confine discussion to the latter option.

4.5.1 Necessary Conditions for Optimality

We now present the optimality conditions for this fusion coder (avoiding lengthy derivations).

Optimal Encoder : From the above equation, the optimal encoding index $\mathcal{E}(\mathbf{x})$ for input vector $\mathbf{x}$ is

$$\mathcal{E}(\mathbf{x}) = \arg\min_{\mathbf{i}\in\mathcal{I}} \sum_{l=1}^{L} \sum_{\mathbf{q}\in B_l} P(\mathbf{q}) d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathbf{i}, \mathcal{S}(l))), \forall \mathbf{x}$$

Optimal Bit-Selector : Similarly, the best set of bits to be retrieved for a particular label (query-region/partition) is the one that minimizes the Lagrangian sum of the distortion measure $\sum_{\mathbf{q}\in B_l} P(\mathbf{q}) d_{\mathbf{q}}(.,.)$, averaged over the training set, and the retrieval rate i.e.

$$\mathcal{S}(l) \quad = \quad \arg\min_{\mathbf{e}\in\mathcal{B}} \sum_{\mathbf{q}\in B_l} P(\mathbf{q}) \{ \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}} d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathcal{E}(\mathbf{x}), \mathbf{e})) + \lambda |\mathbf{e}| \}, \forall l$$

Optimal Query-classification : The optimal labelling (grouping) of the queries would be

$$\mathcal{C}(\mathbf{q}) \quad = \quad \arg\min_{1\le l\le L} \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}} d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathcal{E}(\mathbf{x}), \mathcal{S}(l))) + \lambda |\mathcal{S}(l)|$$

Optimal Decoder : We use $\mathbf{i}_{\mathbf{e}}$, to denote the sub-index extracted from $\mathbf{i}$ by retrieving the bits in the positions indicated by $\mathbf{e}$ and $\mathcal{D}(\mathbf{i}, \mathbf{e})$ to denote the corresponding code-vector. By setting to zero, the partial derivatives of J w.r.t $\mathcal{D}(\mathbf{i}, \mathbf{e}) \forall \mathbf{e} \in \mathcal{B}$, we find the optimal decoder to be

$$\mathcal{D}(\mathbf{i}, \mathbf{e}) = \frac{1}{|F_{\mathbf{i},\mathbf{e}}|} \sum_{\mathbf{x}\in F_{\mathbf{i},\mathbf{e}}} \mathbf{x}, \forall \mathbf{e}, \mathbf{i}$$

where $F_{\mathbf{i},\mathbf{e}} = \{\mathbf{x} : (\mathcal{E}(\mathbf{x}))_{\mathbf{e}} = (\mathbf{i})_{\mathbf{e}}\}$.

4.5.2 Algorithm for Design

We design the fusion coder by iteratively applying the conditions for optimality. Upon convergence, we learn the parameters of the structure imposed on $\mathcal{C}$ such as the centroids, prototypes, the nodes to be split etc. Once the structure of $\mathcal{C}$ has been learnt, we re-optimize $\mathcal{E}, \mathcal{D}, \mathcal{S}$.

4.6 Experimental Results

4.6.1 Data-sets

We tested our algorithm extensively on both synthetic and real data-sets, where we evaluated the operational (retrieval) rate (R_r) vs. distortion D curves, for different settings of storage complexity. A brief description of our data-sets follows.

SYNTH: Synthetic data

For the synthetic data-set SYNTH, the sensor sources were modelled as correlated Gaussian sources (of unit variance i.e. $\sigma^2 = 1$), with the correlation between sources modelled as falling exponentially with distance. Specifically, if ρ_{ij} represents the correlation between sources X_i and X_j,

$$\rho_{ij} = \rho^{|i-j|} \tag{4.24}$$

where $-1 \leq \rho \leq 1$. This correlation model can be expected when spatio-temporal sensor fields are uniformly sampled [98]. We created synthetic data-sets with $\rho = 0.3$ and $\rho = 0.8$, corresponding to low and moderately-high correlation data-sets, with $M = 50$ sources, each having 6000 training samples and 1,000,000 test samples. The performance on the test-set is reported.

STOCKS: Real Data

The first real data-set, the STOCKS data-set, is available in the University of California, Riverside (UCR) Time-Series Data Mining Archive [1]. It consists of $M = 93$ stocks, each having 3000 samples.

[1]The authors would like to thank Dr. Eamonn Keogh of the University of California, Riverside for kindly providing the STOCKS data-set.

Intel Berkeley Sensor Data: Real Data

The second real data-set used was the one generated by the Intel Berkeley Research Lab[2]. Data were collected from 54 sensors deployed in the Intel Berkeley Research lab between February 28 and April 5, 2004. Each sensor measures humidity, temperature, light and voltage values once every 31 seconds. We retain data from those sensors that generated in excess of 50,000 readings. This corresponds to temperature, light, humidity and voltage readings from 15 sensors which is equivalent to 60 sources.

4.6.2 Query Distribution

We tested the performance of the fusion coder on several query distributions that model real user behavior. Even though in theory, there are $2^M - 1$ possible queries, typically, only a smaller subset of sources (say n) will normally requested at any time. There are $\binom{M}{n}$ ways of selecting n out of M objects and for moderate values of M, even this might be very large. For example, if $M = 30$ and $n = 4$, $2^{30} - 1 \simeq 10^9$and $\binom{30}{4} = 27405$.

We describe *exponential queries* ("EXP"), which we believe are reasonable models for real-user behavior. Queries request for contiguous *neighborhoods* of sources. We impose a shifted exponential distribution on neighborhood size. In our sample distribution, on an average 9 are requested and it is representative of 345 queries, that were randomly generated. Even though the queries were chosen randomly, we ensured that each source is requested by at least one query. Figure 4.3 is representative of this query distribution. The probability is plotted versus the *size* of the query $|\mathbf{q}|$. It is also to be noted that even a query set of size 345 *cannot be handled* with the naive storage technique presented in 2.8.2, i.e. compressing and storing every subset of sources separately, without paying an enormous price in total storage.

[2]Download from http://db.csail.mit.edu/labdata/labdata.html

Figure 4.2. "Neighborhoods" of sources (on a 1-D sensor array) of varying size

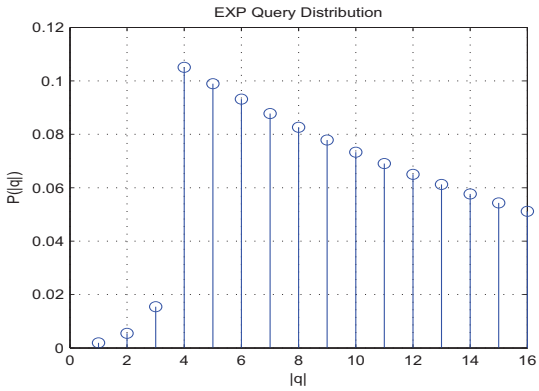

Figure 4.3. EXP: Exponential Distribution on "neighborhood" (query) sizes for $M = 50$ sources

4.6.3 Fusion Coding (FC) vs. Joint Compression (VQ)

We compared the performance of joint compression and selective bit-retrieval for both the synthetic and real data-sets (see Figures 4.4, 4.5 and 4.6). The joint compression of the data set was performed by a Vector Quantizer (VQ) designed with the standard Generalized Lloyd Algorithm (GLA) [62]. The performance of our proposed FC was evaluated at two settings of storage, $R_s = 4$ and $R_s = 6$ bits. Since both systems are designed iteratively, they are initialization dependent. We performed 20 different runs with random codebook initializations (for both VQ and FC) and present the best performance of each method.

For the synthetic data-set with $\rho = 0.3$ (Figure 4.4), FC is able to provide a speed-up of nearly 3X at a distortion level of 9.1dB. For the synthetic data-set with $\rho = 0.8$ (Figure 4.5), there is a 3X speed-up over the joint compression technique, with average distortion of 8.5dB. Additionally, there is also a distortion gain of nearly 1dB at a retrieval rate of 1 bit per sample.

In the real data-set STOCKS (Figure 4.6), FC provides a $\approx$1.6X speed-up with

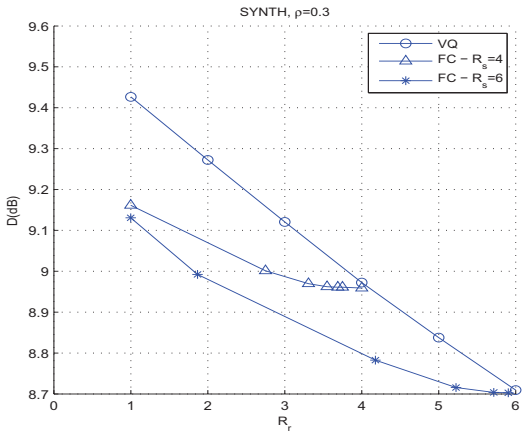

Figure 4.4. Data-set SYNTH, $\rho = 0.3$

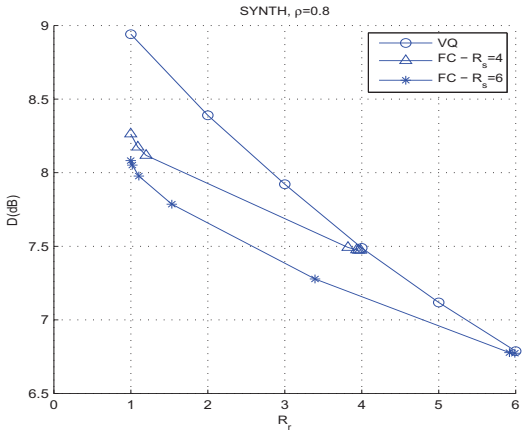

Figure 4.5. Data-set SYNTH, $\rho = 0.8$

distortion 28dB and nearly 3.5dB less distortion at an average retrieval rate of 3 bits.

In the Intel Berkeley Lab data-set (Figure 4.7), we notice gains in the range of 2.6dB at an average retrieval rate of 2 bits and 3.5dB at an average retrieval rate of 3 bits. FC also provides $\approx$ 2X reduction in retrieval rate at a distortion of 26.2dB.

We also note that increasing R_s results in better performance of the selective retrieval technique. This is possible since increasing storage allows more freedom in the design of the bit-selector. However, this increase in gain is relatively marginal in the STOCKS data-set. This is because the design algorithm gets trapped in local minima that riddle

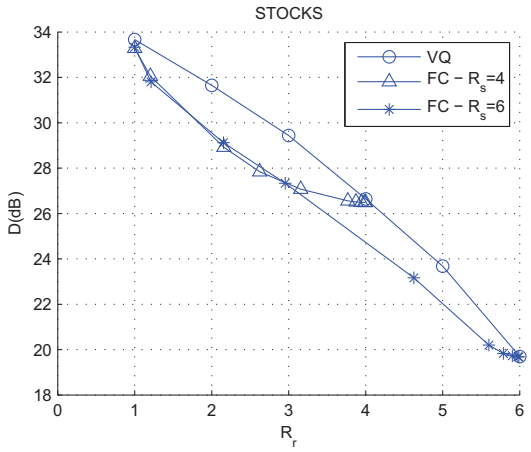

Figure 4.6. Data-set STOCKS

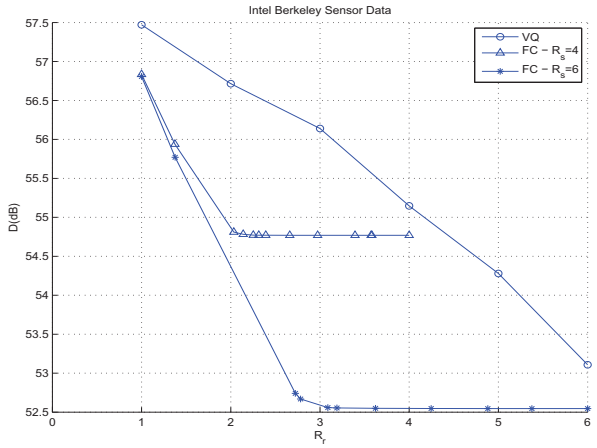

Figure 4.7. Data-set Intel Berkeley Sensor Data

the cost surface.

4.6.4 Quantization of the Query-space

Once the queries were grouped as described in section 4.5.1, we partitioned the query space with a decision tree [99] [100]. We preferred the decision tree over the nearest neighbor/nearest prototype classifiers since the performance of the latter would be dependent on meaningful distance/distortion metrics for the discrete query space $\{0,1\}^M$. For example, if two sources are highly correlated, the request for either or

both of them should have the same retrieval cost (subset of bits) and hence it would make sense that these queries be handled together. It might be necessary to employ a (linear/non-linear) transformation to a secondary feature space for the Euclidean norm to make sense. On the other hand decision trees can operate on discrete data and are known to be efficient classifiers in this setting. Hence, we expect the decision tree paradigm to handle (unknown) queries.

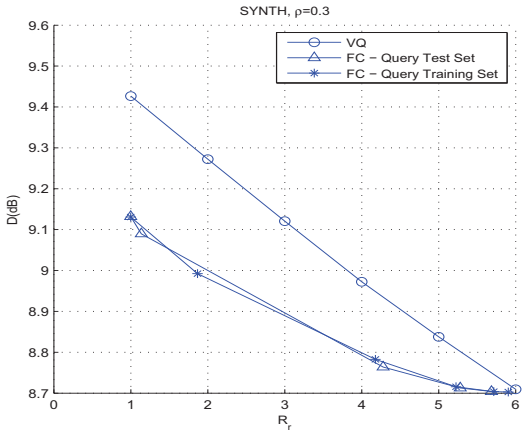

Figure 4.8. Data-set SYNTH, $\rho = 0.3$ with Query Quantization

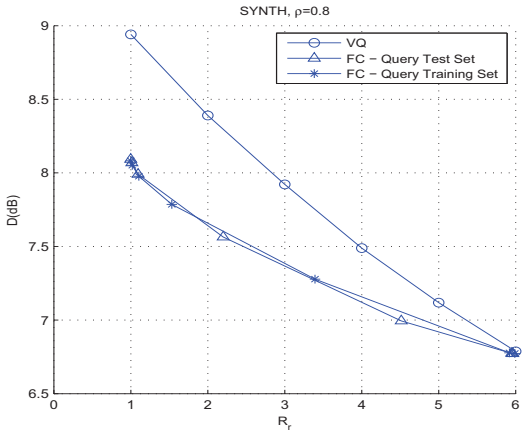

Figure 4.9. Data-set SYNTH, $\rho = 0.8$ with Query Quantization

During the training phase, the training set of queries were partitioned into $L = 6$ groups during the fusion coder design (as explained in section 4.5). Subsequently,

a decision tree that performs this classification was constructed and the fusion coder components were re-optimized. Next, a test set of 345 queries was extracted from the same distribution. This new set of queries was classified by the decision tree and the resulting distortion-retrieval rate performance of the system was evaluated.

We note a very small loss in performance of the system on the new set of queries as compared (see Figures 4.4, 4.5, 4.8 and 4.9) and the performance advantages over the joint compression (VQ) scheme are maintained.

Chapter 5

Efficient Fusion Coder Design

In this chapter we describe approaches to reduce design complexity in fusion coding. We first consider scalability of the (memoryless) fusion coder and note the exponential growth in complexity of design with storage rate.

5.1 Fusion Coder Performance and Scalability

Consider an experimental example of memoryless correlated Gaussian sources of unit variance $X_m, 1 \leq m \leq M$. The correlation between sources X_i and X_j is $\rho_{ij} = \rho^{|i-j|}$, where $-1 \leq \rho \leq 1$. This correlation model is consistent with uniform sampling of a linear sensor field [98]. Queries were assumed to be uniformly distributed over contiguous "neighborhoods" of n sensors (see Figure 5.1). In our experiments, $M = 50$ sources and any $n = 10$ contiguous sources are queried, which implies that $|\mathcal{Q}| = 41$. We chose $\rho = 0.8$ and generated a database of 40,000 vectors.

Figure 5.1. Neighborhood queries on a linear sensor array

The fusion coder was designed at two storage rate constraints, $R_s = 4$ and $R_s = 8$. The competing joint compression technique employed a vector quantizer (VQ), where the compression rate $R_s(= R_r)$ was varied from 1 to 8 bits per vector. Figure 5.2 provides the performance evaluation (ignore the "shared descriptions" results which will be discussed in the next subsection). It is clear from the figure that the fusion coder provides significant selective retrieval gains over naive joint compression (vector quantization) of sources, and these gains increase with the allowed storage rate. This is because at higher storage rates, there are more degrees of freedom in the design of the bit-subset selector.

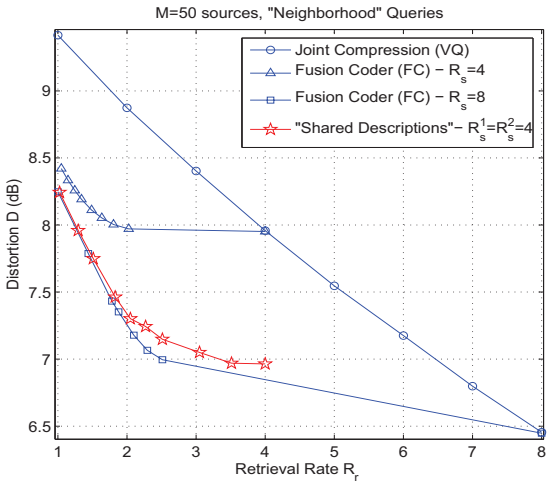

Figure 5.2. Performance comparison of fusion coding for selective retrieval

It is, however, of considerable practical importance to note that the overall design complexity of the optimal solution is $O(2^{R_s})$. Given storage rate R_s, the encoding operation involves searching for the best index out of 2^{R_s}, i.e. the index that minimizes the query averaged distortion. Note further that this encoding complexity also arises during operation of the fusion coder, not only during offline design. The design of the bit-subset selector searches for the best subset of R_s bits, out of $2^{R_s} - 1$ candidates. The codebook update operation and the codevector storage are also of $O(2^{R_s})$ complexity.

Thus, the system complexity (design, operational and codevector storage) grows exponentially with storage rate. This implies that system design and operation scale poorly with the allowed storage rate, which itself grows with the number of sources.

5.1.1 Implications

This represents a major practical concern. In order to design for practical sized sensor networks, large storage rates are necessary. Secondly, real life signals exhibit temporal correlations that are equally as significant as spatial correlations. The optimal approach to exploit spatio-temporal correlations would be to fusion code over larger block lengths. Even if R_s were small, we note the overall complexity growth is $O(2^{nR_s})$, where n is the number of blocks. Clearly, fusion coding over several blocks is unfeasible and hence, more efficient approaches are necessary. In subsequent sections, each of these practical issues is tackled in a different manner. The Shared Descriptions idea, described in section 5.2, enables scalability to large storage rates, while Predictive Fusion Coding (section 5.3) is a low complexity technique to exploit spatio-temporal correlations over long blocks.

5.2 The Shared Descriptions Approach

We now describe the "Shared Descriptions" reformulation of the fusion coding problem (originally presented in [101]) such that it enables explicit control of the complexity. A structure needs to be imposed to constrain the complexity in a controlled way so as to optimize tradeoff with performance. For example, in classical vector quantizer design, the split VQ structure might be preferred over full-search VQ because of its lower codevector search complexity [62]. In an analogous fashion, we "split" the storage and spread the complexity over a number of smaller (lower complexity) encoders. Since the complexity is exponential in the storage rate this entails considerable complexity gains.

Each encoder now operates independently and we refer to the compressed bits produced by each encoder as a *shared description* for reasons that will shortly become obvious.

We constrain the bit-selection module to select the subset of bits for a query from one of the shared descriptions (see Figure 5.3). This restriction implies that each description is in fact shared by a group of queries. Since each query is mapped to a particular description and no query retrieves bits from two or more different descriptions, it follows that the different encoders can operate independently of each other. If we employ two encoders (two descriptions) which encode at rates R_s^1 and R_s^2, then the net encoding complexity is $2^{R_s^1} + 2^{R_s^2}$ rather than $2^{R_s^1+R_s^2}$.

To illustrate this property, let use $\mathbf{i}_1 \in \mathcal{I}_1$ and $\mathbf{i}_2 \in \mathcal{I}_2$ to denote the two partitions of a typical encoding index $\mathbf{i} \in \mathcal{I}$, where $\mathcal{I} = \mathcal{I}_1 \times \mathcal{I}_2$ and $\mathbf{i} = (\mathbf{i}_1, \mathbf{i}_2)$. The space of queries is now partitioned into groups of queries, one per shared description, though each query in the group may still use a different subset of bits from their shared description. Let A_1 and A_2 denote the two groups, where $A_1 \cup A_2 = \mathcal{Q}$ and $A_1 \cap A_2 = \phi$. Then under the chosen constraint, the optimal encoding rule transforms to $\forall \mathbf{x} \in \mathcal{X}$

$$\begin{aligned}
\mathcal{E}(\mathbf{x}) &= \arg\min_{\mathbf{i}} \sum_{\mathbf{q}} P(\mathbf{q}) d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathbf{i}, \mathcal{S}(\mathbf{q}))) \\
&= \arg\min_{\mathbf{i}_1,\mathbf{i}_2} \{ \sum_{\mathbf{q}\in A_1} P(\mathbf{q}) d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}((\mathbf{i}_1, \mathbf{i}_2), \mathcal{S}(\mathbf{q}))) \\
&\qquad\qquad + \sum_{\mathbf{q}\in A_2} P(\mathbf{q}) d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}((\mathbf{i}_1, \mathbf{i}_2), \mathcal{S}(\mathbf{q})))\} \\
&= \arg \min_{\mathbf{i}_1\in\mathcal{I}_1,\mathbf{i}_2\in\mathcal{I}_2} (f(\mathbf{i}_1, A_1) + g(\mathbf{i}_2, A_2)) \\
&= (\arg \min_{\mathbf{i}_1\in\mathcal{I}_1} f(\mathbf{i}_1, A_1)), \arg \min_{\mathbf{i}_2\in\mathcal{I}_2} g(\mathbf{i}_2, A_2))
\end{aligned}$$

where f and g are suitably defined functions.

In a similar fashion it can be argued that the design complexity for the bit and description selection is $2^{R_s^1} + 2^{R_s^2}$ and the net codevector update and storage complexity is $2^{R_s^1} + 2^{R_s^2}$. Thus the net system complexity is $O(2^{R_s^1} + 2^{R_s^2}) << O(2^{R_s^1+R_s^2}) = O(2^{R_s})$. The performance of this "split encoder"/"shared description" formulation is presented in Figure 5.2. There is a small performance loss relative to the unconstrained fusion coder,

of about 0.2dB. However, the "shared description" setup considerably reduces system complexity from $O(256)$ to $O(32)$. It also has significant performance advantages over joint compression.

In general, let K be the number of descriptions/encoders. The k^{th} encoder compresses the M-dimensional input vector $\mathbf{x}$ to R_s^k storage bits at each instant. The total storage would be $R_s = \sum_k R_s^k$. Correspondingly, we introduce notation

$$\mathcal{E}_k : \mathbb{R}^M \rightarrow \mathcal{I}_k = \{0,1\}^{R_s^k}, \forall k = 1, ..., K \tag{5.1}$$

for the K encoders.

For the k^{th} description, we have the corresponding bit-selector as

$$\mathcal{S}_k : \mathcal{Q} \rightarrow \mathcal{B}_k = 2^{\{1,...,R_s^k\}} \tag{5.2}$$

Next, we use $\beta : \mathcal{Q} \rightarrow \mathcal{K} = \{1, ..., K\}$ to denote the description selector i.e. for query $\mathbf{q}$, bits are retrieved from description $\beta(\mathbf{q})$.

The decoder is now modified to be the map

$$\mathcal{D} : \bigcup_{k=1}^{K} \mathcal{I}_k \times \mathcal{B}_k \times \{k\} \rightarrow \hat{\mathcal{X}} \tag{5.3}$$

Given $\beta(\mathbf{q}) = k$ for some $\mathbf{q}$, the decoder accesses the bits specified by $S_k(\mathbf{q})$ in description k and estimates $\hat{X} = \mathcal{D}(\mathcal{E}_k(X), S_k(\mathbf{q}), k)$. (Consequently, reconstruction of the relevant sources is $\hat{X}_{(\mathbf{q})}$, where we use the subscript $\mathbf{q}$ to indicate the relevant sources.) If we denote the set of queries that are mapped to the k^{th} description as $A_k = \{q : \beta(q) = k\}$, the average distortion is evaluated as

$$D = \sum_{k=1}^{K} \sum_{\mathbf{q} \in A_k} P(\mathbf{q}) \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathcal{E}_k(\mathbf{x}), \mathcal{S}_k(\mathbf{q}), k)) \tag{5.4}$$

where $\mathcal{X}$ is the training set. Likewise, the average retrieval rate is evaluated as

$$R_r = \sum_{k=1}^{K} \sum_{\mathbf{q} \in A_k} P(\mathbf{q}) |\mathcal{S}_k(\mathbf{q})| \tag{5.5}$$

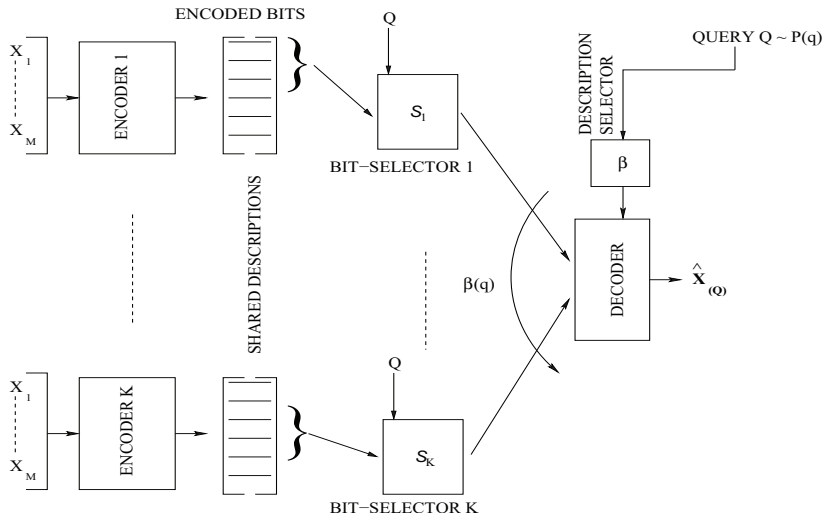

Figure 5.3. Shared Description Fusion Coder for Memoryless Sources

Let E_k represents the bits within description k that are actually used. Clearly,

$$E_k = \bigcup_{A_k} S_k(\mathbf{q}) \tag{5.6}$$

and

$$R^k_{s,util} = |E_k| = |\bigcup_{\mathbf{q}\in A_k} \mathcal{S}_k(\mathbf{q})| \tag{5.7}$$

This implies that the *true* complexity of the k^{th} encoder is $O(2^{R^k_{s,util}})$. Hence, the total storage and system complexity are evaluated to be

$$R_{s,net} = \sum_{k=1}^{K} R^k_{s,util} \qquad C_{net} = \sum_{k=1}^{K} 2^{R^k_{s,util}}$$

Given a total storage capacity R_s, allowed average retrieval rate R_{ret}, allowed system complexity C and K shared descriptions, the optimal shared description fusion coder is the solution to

$$\arg\min_{\mathcal{E},\mathcal{D},\mathcal{S}} D \ni R_{s,net} \le R_s, C_{net} \le C, R_r \le R_{ret} \tag{5.8}$$

Equivalently, we seek solutions of

$$\arg\min_{\mathcal{E},\mathcal{D},\mathcal{S}} J = \arg\min_{\mathcal{E},\mathcal{D},\mathcal{S}} D + \lambda R_r \ni C_{net} \le C, R_{s,net} \le R_s \tag{5.9}$$

where $\lambda \ge 0$ is a Lagrange multiplier. Now, it can be clearly seen that fusion coding [102] is actually a special case of shared descriptions fusion coding (when $C = \infty, K = 1$).

5.2.1 Necessary Conditions for Optimality

The Lagrangian cost J can now be written as

$$J = \sum_{k=1}^{K} \sum_{\mathbf{q} \in A_k} \frac{P(\mathbf{q})}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathcal{E}_k(\mathbf{x}), \mathcal{S}_k(\mathbf{q}), k)) + \lambda \sum_{k=1}^{K} \sum_{\mathbf{q} \in A_k} P(\mathbf{q}) |\mathcal{S}_k(\mathbf{q})|$$

Optimal Encoders : Given all the other mappings, it follows from (5.10) that the optimal encoding index produced by encoder k for input vector $\mathbf{x}$ is

$$\mathcal{E}_k(\mathbf{x}) = \arg\min_{\mathbf{i} \in \mathcal{I}_k} \sum_{\mathbf{q} \in A_k} P(\mathbf{q}) d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathbf{i}, \mathcal{S}_k(\mathbf{q}), k)), \forall \mathbf{x}$$

Optimal Codevectors : For $k \in \mathcal{K}, \mathbf{i} \in \mathcal{I}_k, e \in \mathcal{B}_k$, we define $F_k = \{\mathbf{x} : (\mathcal{E}_k(\mathbf{x}))_e = (\mathbf{i})_e\}$, and the optimal codevector is

$$\mathcal{D}(\mathbf{i}, e, k) = \frac{1}{|F_k|} \sum_{\mathbf{x} \in F_k} \mathbf{x}, \forall \mathbf{i} \in \mathcal{I}_k, \forall \mathbf{e} \in \mathcal{B}_k, \forall k \in \mathcal{K}$$

Optimal Bit-subset Selectors : Let $\tilde{\mathcal{B}}_k = \{e \in \mathcal{B}_k : |e \bigcup E_k| + \sum_{k' \neq k} |E_{k'}| \leq R_s, 2^{|e \bigcup E_k|} + \sum_{k' \neq k} 2^{|E_{k'}|} \leq C\}$. $\tilde{\mathcal{B}}_k$ represents the valid set of bit-selections that do not violate the storage and complexity constraints. Hence, the optimal bit-subset selection, given the encoding indices and the codebook, is the rule $\forall \mathbf{q}, k$

$$\mathcal{S}_k(\mathbf{q}) = \arg\min_{e \in \tilde{\mathcal{B}}_k} \lambda |e| + \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}} d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathcal{E}_k(\mathbf{x}), e, k))$$

Optimal Description Selector : By reordering the terms of the Lagrangian, the optimal description for a particular query $\mathbf{q}$, given the encoding indices and the codebook, is the rule $\forall \mathbf{q}$

$$\beta(\mathbf{q}) = \arg\min_{k \in \mathcal{K}} \min_{e \in \tilde{\mathcal{B}}_k} \lambda |e| + \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}} d_{\mathbf{q}}(\mathbf{x}, \mathcal{D}(\mathcal{E}_k(\mathbf{x}), e, k))$$

If $\beta(\mathbf{q}) = k$, we update $E_k = E_k \bigcup \mathcal{S}_k(\mathbf{q})$, before optimizing for the next query. We also note that the E_k update is necessary before the bit-selector optimization for the next query.

5.2.2 Design Algorithm

A natural design algorithm is to iteratively enforce the optimality conditions i.e. optimize each mapping separately, while assuming that the remaining mappings are optimal (and given). There exist only finite number of partitions of a finite training set and there are only finite number of ways to partition bits and queries. As each step of the iteration is monotone non-increasing in the cost, the algorithm must converge to a locally optimal design in a finite number of iterations. However, since the cost is not convex, this simple design approach is dependent on initialization, and multiple runs with different (possibly random) initializations may be necessary to obtain a good solution.

5.2.3 Simulation Results

For the same experimental model we considered before, we performed Shared Descriptions Fusion Coding (SDFC) at a storage rate of $R_s = 24$, with $K = 3$ descriptions. The overall complexity constraint imposed was $C = 768$. The SDFC performance was compared with two naive compression techniques that scale well with storage rate - scalar quantization (at 1 bit per source) and split VQ. Since each query consists of 10 sources, scalar quantization is forced to retrieve 10 bits for every query. Split VQ is a standard scheme in speech coding for scaling VQ to high dimensions (equivalent to a large number of sources in our case) which translate into high rates. Here, we perform split VQ by splitting the $M = 50$ sources into (roughly) equal sized groups and compressing each group separately, where the total storage rate $R_s = 24$ is divided equally among all the groups. The number of groups was varied over 24, 12, 8, 6 and 4. For each such grouping of sources, we obtain a point on the retrieval rate-distortion curve.

From Figure 5.4, we note significant performance gains of SDFC over both scalar quantization and split VQ. At the same retrieval rate, SDFC offers from 0.5dB to 1.6dB decrease in distortion relative to split VQ. At the same level of distortion, SDFC provides

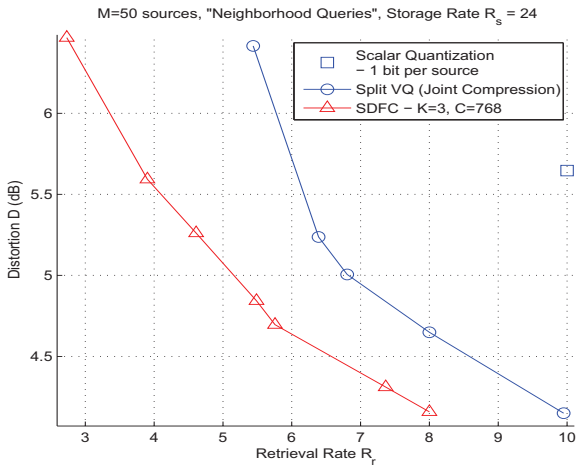

Figure 5.4. SDFC vs. split VQ (joint compression) vs. Scalar Quantization

retrieval rate reduction by factors of 1.25X to 2X over split VQ and 2.5X over scalar quantization. We also note that scalar quantization of sources (at 1 bit per source) requires more than twice the storage of SDFC and split VQ.

5.3 Predictive Fusion Coding

Streaming data such as video/sensor streams exhibit significant temporal correlations and efficient storage and retrieval would have to exploit these correlations as well. But, as we shall show in subsequent sections, the design of optimal predictive fusion coders is compounded both by the presence of the prediction loop and the need to accommodate a (possibly *exponentially*) large query set. We describe a complexity constrained approach, that derives and builds up from our earlier publication [103], and still yields huge gains over naive joint compression (vector quantization (VQ) or predictive VQ) of all sources and memoryless fusion coding.

5.3.1 Optimal Predictive Fusion Coding

While time-correlations could conceivably exploited by fusion coding over larger blocks, this would be a high complexity approach. Linear predictive coding, on the other hand, is attractive as a low-complexity alternative that can yet exploit temporal correlations. We first note that since all the sources X_m are available at the encoder, they can be equivalently replaced by query-specific super sources i.e. vector sources of the form $\mathbf{X}_{(\mathbf{q})} = [..., X_m, ...]^T, \forall m \ni q_m = 1$.

Now in an optimal (linear) predictive fusion coder (PFC), at every instant, each such super source $\mathbf{X}_{(\mathbf{q})}$ is predicted and all the prediction residuals are fusion coded to exploit spatial correlations efficiently (see Figure 5.5). However, this imposes on the encoder the need to accommodate the entire (possibly exponentially large) query set. This would require $|\mathcal{Q}|$ prediction loops and the encoder would need to handle input vectors whose size $\sum_{\mathbf{q}\in\mathcal{Q}} |\mathbf{q}|$ grows with the query set $\mathcal{Q}$ and hence, could be of impracticably high complexity.

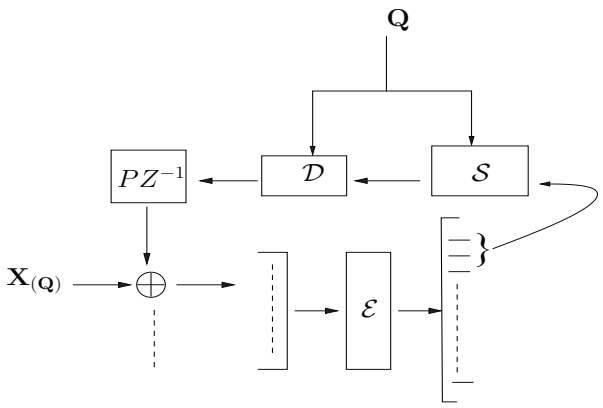

Figure 5.5. Optimal Predictive Fusion Coding: Encoder

5.3.2 Constrained Predictive Fusion Coding

Hence, we propose predictive fusion coding with constraints i.e. where queries and sources share predictors. In the extreme situation, all queries *share a single predictor*. In the general case, we assume K prediction loops are possible. We assume one-step prediction and that the prediction matrix P is estimated from open-loop statistics. Figure 5.6 represents our predictive fusion coding encoder. We continue with the notations developed in section **??** and introduce three new mappings.

$$\begin{aligned} \beta : \mathcal{Q} &\rightarrow \mathcal{K} = \{1, ..., K\} \\ \mathcal{S}_P : \mathcal{K} &\rightarrow \mathcal{B} = 2^{\{1,...,R_s\}} \\ \mathcal{D}_P : \mathcal{I} \times \mathcal{B} &\rightarrow \hat{\mathcal{X}}_P \subset \mathbb{R}^M \end{aligned}$$

The mapping β, which we term the predictor selector, quantizes/partitions the query space into K groups, each of which share a prediction loop. We allow for a predictor bit-selection $\mathcal{S}_P(k), \forall k \in \mathcal{K}$ for each of the K groups. At each instant, predictor k returns an estimate of all sources, based on a subset $\mathcal{S}_P(k)$ of the immediately past encoding bits. We note that $\hat{\mathcal{X}}_P \subseteq \mathbb{R}^{MK}$ and $\hat{\mathcal{X}} \subseteq \mathbb{R}^{MK}$. The prediction errors are fed as input to the encoder $\mathcal{E}$, which is now modified to be

$$\mathcal{E} : \mathbb{R}^{MK} \rightarrow \mathcal{I} \tag{5.10}$$

The error residuals are compressed to the best possible index (set of encoding bits) $\mathcal{E}(\mathbf{e}_1, ..., \mathbf{e}_K)$, that are then stored in the database. The compressed error residual $\hat{\mathbf{e}}_k = \mathcal{D}_P(\mathcal{E}(\mathbf{e}_1, ..., \mathbf{e}_K), \mathcal{S}_P(k))$ is fed as input to the k^{th} prediction loop/filter.

During query processing, the process is reversed. For all queries, the stored bits from the locations given by $\mathcal{S}_P(\beta(\mathbf{q}))$ are extracted and the reconstructed residual $\hat{\mathbf{e}}_{\beta(\mathbf{q})}$ is fed into the prediction filter. For each query $\mathbf{q}$, additional bits are extracted (if necessary) and the output of the prediction filter $\tilde{\mathbf{X}}_{\beta(\mathbf{q})}$ is augmented by $\hat{\mathbf{e}}_{k,\mathbf{q}} = \mathcal{D}(\mathcal{E}(\mathbf{e}_1, ..., \mathbf{e}_K), \mathcal{S}(\mathbf{q}))$, to reconstruct the queried-sources $\hat{\mathbf{X}}_{(\mathbf{q})}$. At this point, we note that the retrieval rate per sample for query $\mathbf{q}$ is $R_{\mathbf{q}} = |\mathcal{S}(\mathbf{q}) \cup \mathcal{S}_P(\beta(\mathbf{q}))|$ bits.

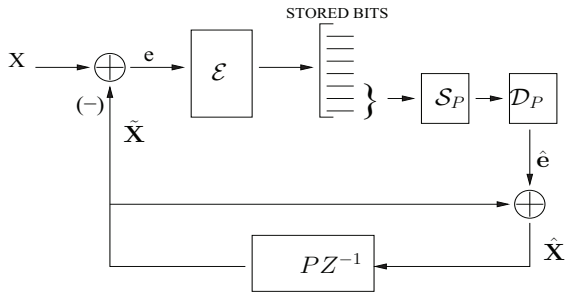

Figure 5.6. Constrained Predictive Fusion Coding: Encoder, K=1 Prediction Loop

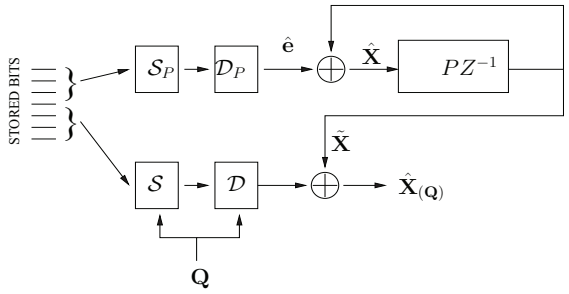

Figure 5.7. Constrained Predictive Fusion Coding: Decoder, K=1 Prediction Loop

In practice, we evaluate average performance over available training sets and assume ergodicity i.e. an average evaluated over the training set is equivalent to ensemble averaging. Let $A_k = \{\mathbf{q} : \beta(\mathbf{q}) = k\}$. We loosely use the notation $\|\mathbf{z}\|^2_{\mathbf{q}}$ to denote $\sum_m q_m z_m^2$. The net cost to be optimized is represented by

$$\begin{aligned} J &= D + \lambda R_r \\ &= \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{A_k} P(\mathbf{q})(\|\mathbf{x}(n) - (\tilde{\mathbf{x}}_k(n) + \hat{\mathbf{e}}_{\mathbf{q}}(n))\|^2_{\mathbf{q}}) \\ &\qquad + \lambda \sum_{k=1}^{K}\sum_{A_k} P(\mathbf{q}) |S(\mathbf{q}) \bigcup S_P(k)| \end{aligned}$$

5.3.3 Design of Predictive Coding Systems

The design of predictive coding systems is complicated by the feedback loop of the prediction filter [62]. It is a well understood fact that the optimal quantizers are matched to the source statistics. However, in predictive coding, the quantizer compresses the

prediction error between the source and an estimate of the source, based on the previous reconstructed (*quantized*) source sample(s). Thus the prediction errors which will be used to train the quantizer, themselves depend upon the quantizer parameters. Hence, designing an optimal quantizer matched to the statistics of the prediction residual is a challenging proposition.

Well known design methodologies for predictive coder design fall into two categories: open-loop design and closed-loop design [62]. In the open loop design, the error residuals are generated from the source i.e. without the quantizer in the loop and these are used to design a quantizer. In closed-loop design, we iterate between generating error residuals, with the quantizer fixed and optimizing the quantizer, given the error residuals. Thus on the one hand quantizer design in open-loop, while stable, is sub-optimal due to mismatch between quantizer and closed-loop error residuals (except at high rates). On the other hand in closed-loop design, the design procedure is always in closed-loop but this is known to be inherently unstable (except at high rates), as the training set and statistics of the training set changes with every iteration in an extremely unpredictable fashion [104].

The asymptotic closed loop (ACL) design principle [104] [105] is a best-of-both worlds approach. It designs the system *always in open loop*, using the same training set within each iteration to design all components and hence, is stable. However, from one iteration to the next, the training set of error residuals is gradually modified till *asymptotically* they match the true residuals generated by the prediction loop. In other words, asymptotically the prediction loop is *closed*. Given a source sequence $\mathcal{X} = \{x(n)\}$ and assuming superscript p to denote iteration number, the philosophy of ACL design can be abstracted in the following sequence of steps (see Algorithm 2).

From step 7, it is clear that each quantizer is applied to error residuals that were used to train it i.e. there is no mismatch between the error residuals and the quantizers. Additionally, we note that in step 4 we are creating the error residuals in one go, *without*

Algorithm 2 VQ-ACL($\mathcal{X}, R_s$)

1: Set $p = 1$, design predictors from open-loop error statistics, initialize quantizer parameters

2: Generate a sequence of source reconstructions $\hat{\mathcal{X}} = \{\hat{x}^p(n)\}$.

3: Increment p.

4: Create the source predictions $\tilde{\mathcal{X}}$ based on $\hat{\mathcal{X}}$ i.e. $\tilde{x}^p(n) = P\hat{x}^{p-1}(n-1)$

5: Create error residuals $e^p(n) = x(n) - \tilde{x}^p(n)$

6: Use error residuals to optimize/update quantizer parameters.

7: Evaluate $\hat{e}^p(n) = Q^p(e^p(n))$

8: Update $\hat{\mathcal{X}}$ such that $\hat{x}^p(n) = \tilde{x}^p(n) + \hat{e}^p(n)$

9: Evaluate cost D^p.

10: If $\frac{|D^{p-1} - D^p|}{D^{p-1}} < \epsilon$, STOP

else GOTO step 3.

going through the prediction loop. Thus, in each iteration it *is* design in open loop. Next, in step 2 we force $\tilde{x}^p(n) = P\hat{x}^{p-1}(n-1)$ and in step 5, we force $\hat{x}^p(n) = \tilde{x}^p(n) + \hat{e}^p(n)$. Equivalently,

$$\hat{x}^p(n) = P\hat{x}^{p-1}(n-1) + Q^p(x(n) - P\hat{x}^{p-1}(n-1)) \tag{5.11}$$

If Q^p was the quantizer at iteration p, as $p \to \infty, Q^p \approx Q^{p-1}$ since the quantizer updates become vanishingly small close to convergence. This implies that $\hat{x}^p(n) \approx \hat{x}^{p-1}(n)$ and $\tilde{x}^p(n) = P\hat{x}^{p-1}(n-1) \approx P\hat{x}^p(n-1)$, i.e. asymptotically, the loop is closed. The reader is directed to [105] for a detailed discussion on convergence criteria and asymptotic behaviour in ACL design.

Application to Predictive Fusion Coding

In the predictive fusion coder, the need to impose complexity constraints on the number of predictions loops possible makes for an important break from the conventional predictive coder design, in that there exists a module ($\mathcal{S}_P$) that lies *within* the prediction loop. $\mathcal{S}_P$ cannot be optimized in open loop form since it exists *in* the prediction loop. Hence, it is not possible to design predictive fusion coders through (stable) open-loop design. For the same reason, a closed loop design of predictive fusion coders would be difficult, since an update of $\mathcal{S}_P$ would not be possible without a proper "handle" on the error residuals. Since the targeted application of predictive fusion coding is for large networks of sources, the coding rate (per source) $\frac{R_s}{M}$ would be very low, leaving ACL as the only viable design methodology.

5.3.4 Predictive Fusion Coder Design by ACL

We extend the asymptotic closed loop (ACL) design principle towards the design of predictive fusion coding systems in the following manner. The ACL predictive fusion decoder is shown in Figure 5.8.

In subsequent discussion, the superscript p denotes the iteration number, the subscript k indexes the prediction loop group, while the parenthesized n denotes time. If

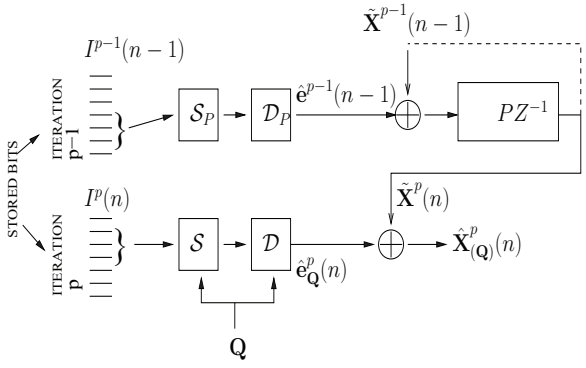

Figure 5.8. Design by Asymptotic Closed Loop: Decoder, K=1 Prediction Loop

$I \in \{0,1\}^{R_s}$ is a typical encoding index and $f \subset \{1,...,R_s\}$, we use the notation I_f to denote the sub-index formed by the bits in I at the locations in f. The error residuals at iteration p, $\forall k \in \mathcal{K}$, $\{\mathbf{e}_k^p(n)\}$ are encoded to the indexes (stored vectors of bits) $\{I^p(n)\}$.

$\{\mathbf{e}_k^p(n)\}$ form the training set at iteration p and are computed as

$$\mathbf{e}_k^p(n) = \mathbf{x}(n) - \tilde{\mathbf{x}}_k^p(n) = \mathbf{x}(n) - P\hat{\mathbf{x}}_k^{p-1}(n-1)$$
$$\Rightarrow \mathbf{e}_k^p(n) = \mathbf{x}(n) - P[\hat{\mathbf{e}}_k^{p-1}(n-1) + \tilde{\mathbf{x}}_k^{p-1}(n-1)]$$

If $\beta(\mathbf{q}) = k$, $\hat{\mathbf{x}}_{\mathbf{q}}^p(n) = \tilde{\mathbf{x}}_k^p(n) + \hat{\mathbf{e}}_{\mathbf{q}}^p(n)$. Note that while $\hat{\mathbf{e}}_k^p(n) = \mathcal{D}_P(I^p(n), \mathcal{S}_P(k))$, $\hat{\mathbf{e}}_{\mathbf{q}}^p(n) = \mathcal{D}(I^p(n), \mathcal{S}(\mathbf{q}))$.

During iteration $p-1$, we seek to minimize the cost at iteration p. Asymptotically this does not matter, but this subterfuge is useful in obtaining effective update rules for $\mathcal{S}_P$ and $\mathcal{D}_P$, since both $\hat{\mathbf{e}}_k^{p-1}(n-1)$ and $\hat{\mathbf{e}}_{\mathbf{q}}^p(n)$ affect $\hat{\mathbf{x}}_{\mathbf{q}}^p(n)$.

The distortion at iteration p is evaluated as

$$D^p = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{A_k} P(\mathbf{q})(\|\mathbf{x}(n) - (\tilde{\mathbf{x}}_k^p(n) + \hat{\mathbf{e}}_{\mathbf{q}}(n))\|_{\mathbf{q}}^2) \tag{5.12}$$

while the retrieval rate at iteration p is evaluated as

$$R_r^p = R_r = \sum_{k=1}^{K}\sum_{A_k} P(\mathbf{q})|S(\mathbf{q})\bigcup S_P(k)| \tag{5.13}$$

Correspondingly, $J^p = D^p + \lambda R_r$. We next present update rules based on ACL for predictive fusion coder design.

Encoder Update : At iteration $p, \forall n,$, we update the encoding as

$$I^p(n) \;=\; \arg\min_{I\in\mathcal{I}} \sum_{k=1}^{K}\sum_{\mathbf{q}\in A_k} P(\mathbf{q})\|\mathbf{e}_k^p(n) - \mathcal{D}(I, \mathcal{S}(\mathbf{q}))\|_{\mathbf{q}}^2$$

Query Bit-subset Selector Update : We update S in the following fashion.

$$\mathcal{S}(\mathbf{q}) \;=\; \arg\min_{f\in\mathcal{B}} \lambda|f\bigcup\mathcal{S}_P(k)| + \frac{1}{N}\sum_{n}\|\mathbf{e}_k^p(n) - \mathcal{D}(I^p(n), f)\|_{\mathbf{q}}^2$$

$\forall \mathbf{q}$, where $k = \beta(\mathbf{q})$.

Query-Codebook Update : Let $F = \{n : \mathcal{E}_f(x(n)) = I_f\}$ and $H_k = \{\mathbf{q} : \mathbf{q} \in A_k, \mathcal{S}(q) = f\}$. $\forall I \in \mathcal{I}, f \in \mathcal{B}$ we set

$$\mathcal{D}(I, f) \quad = \quad \arg\min_{\mathbf{z}} \sum_{n \in F} \sum_{k} \sum_{\mathbf{q} \in H_k} P(\mathbf{q}) \|\mathbf{e}_k^p(n) - \mathbf{z}\|_{\mathbf{q}}^2$$

Predictor Bit-subset Selector Update : On the other hand, the bit-selection for the prediction loop(s) is updated as,

$$\mathcal{S}_P(k) \quad = \quad \arg\min_{f \in \mathcal{B}} \sum_{\mathbf{q} \in A_k} P(\mathbf{q})(\lambda |\mathcal{S}(\mathbf{q}) \bigcup f| + \frac{1}{N} \sum_{n} \|\mathbf{e}_k^p(n) - \mathcal{D}(I^p(n), \mathcal{S}(\mathbf{q}))\|_{\mathbf{q}}^2)$$

$\forall k \in \mathcal{K}$, where $\mathbf{e}_k^p(n) = \mathbf{x}(n) - P[\hat{\mathbf{e}}_k^{p-1}(n-1) + \tilde{\mathbf{x}}_k^{p-1}(n-1)]$ and $\hat{\mathbf{e}}_k^{p-1}(n-1) = \mathcal{D}_P(I^{p-1}(n-1), f), \forall k$.

Predictor-Codebook Update : Let $\tilde{F} = \{n : \mathcal{E}_f(x(n-1)) = I_f\}$, $\forall I \in \mathcal{I}, f \in \mathcal{B}$. Let $G_k = \{\mathbf{q} : \mathbf{q} \in A_k, S_P(\beta(\mathbf{q})) = f\}$ and $\tilde{\mathbf{e}}_{\phi,k}(n) = \mathbf{x}(n) - P[\phi + \tilde{\mathbf{x}}_k^{p-1}(n-1)]$. We update the predictor-codevectors as

$$\mathcal{D}_P(I, f) \quad = \quad \arg\min_{\phi} \sum_{n \in \tilde{F}} \sum_{k=1}^{K} \sum_{G_k} P(\mathbf{q}) \|\tilde{\mathbf{e}}_{\phi,k}(n) - \hat{\mathbf{e}}_{\mathbf{q}}(n)\|_{\mathbf{q}}^2$$

Predictor Selector Update : In every iteration, we partition the query space into K groups, where members of each group share a prediction loop. The rule followed for updating β is

$$\beta(\mathbf{q}) \quad = \quad \arg\min_{k \in \mathcal{K}} \lambda |\mathcal{S}_P(k) \bigcup \mathcal{S}(\mathbf{q})| + \frac{1}{N} \sum_{n} \|\mathbf{e}_k^p(n) - \mathcal{D}(I(n), \mathcal{S}(\mathbf{q}))\|_{\mathbf{q}}^2$$

$\forall \mathbf{q}$, where $\mathbf{e}_k^p(n) = \mathbf{x}(n) - P[\hat{\mathbf{e}}_k^{p-1}(n-1) + \tilde{\mathbf{x}}_k^{p-1}(n-1)]$ and $\hat{\mathbf{e}}_k^{p-1}(n-1) = \mathcal{D}_P(I^{p-1}(n-1), S_P(k)), \forall k$.

ACL algorithm for Predictive Fusion Coder Design

The design algorithm that minimizes $J = D + \lambda R_r$ cost (for a given Lagrange multiplier λ) is presented in Algorithm 3. The design is initialization dependent and several runs with different initializations might be necessary to obtain a good solution.

Algorithm 3 PFC-ACL $(\mathcal{X}, R_s, K, \lambda)$

1: Initialize (e.g. randomly) all query-codebooks and predictor-codebooks, bit-selectors $\mathcal{S}_P$, $\mathcal{S}$, design prediction matrix P, set $p = 0$

2: Increment p

3: Compute the new set of source reconstructions $\{\hat{\mathbf{x}}_k^p(n)\}$, source predictions $\{\tilde{\mathbf{x}}_k^p(n)\}$, error residuals (training set) $\{\mathbf{e}_k^p(n)\}$, $\forall k \in \mathcal{K}$

4: Update **encoding indexes**

5: Update **query bit-subset selection**

6: Update **query-codebooks**

7: Update **predictor bit-subset selection**

8: Update **predictor codebooks**

9: Update **predictor selector**

10: Close the loop and evaluate the Lagrangian $J^p = D^p + \lambda R_r^p$.

11: If $\frac{|J^{p-1} - J^p|}{J^{p-1}} < \epsilon$, STOP

Else go to step 2.

5.3.5 Simulation Results

We used the first order Gauss-Markov source model for our simulations i.e. $X_m(n) = \beta_m X_m(n-1) + W_m(n)$, where $\{W_m(n)\}_1^M$ are i.i.d, zero-mean, unit variance jointly Gaussian random variables with the pairwise correlation coefficient $\rho_{jk} = E[W_j(n), W_k(n)] = \rho^{|j-k|}$. In all our simulations, $\beta_m = 0.8, \forall m$ and $\rho = 0.95$. This model would be representative of a linear sensor array. Our query distribution was a uniform distribution over contiguous "neighborhoods" of n sensors (see Figure 5.9). In our experiments, $M = 100$ sources and any $n = 10$ contiguous sources are queried, which implies that $|\mathcal{Q}| = 91$ and $\sum_{\mathbf{q}} |\mathbf{q}| = 910 >> M = 100$. In all our experiments, the maximum allowed storage capacity was $R_s = 6$ bits and a training set of length 3000 samples was used.

We first consider closed loop design of the PFC for the $\lambda = 0$ setting, where the cost

Figure 5.9. Neighborhood Queries on a Linear Sensor Array

function equals the distortion. At this setting, all bit-selectors are free to retrieve all stored bits and hence, the performance with constrained and unconstrained number of prediction loops is identical. We also note that this design is identical to the design of predictive VQ by closed loop.

From the plot of the true closed loop (distortion) cost variation with iteration (Figure 5.10), it is clear that closed loop design is very unstable. Next, we consider the ACL design of PFC for the $\lambda = 0$ case (Figure 5.11). The true closed loop cost (distortion) gradually decreases with iteration, even if not monotonically. Also, note that the cost estimate by the PFC-ACL algorithm converges with the true cost at a large iteration i.e. asymptotically the loop is closed. Also, note similar convergence in Figure 5.12, where the cost variation is plotted versus iteration for the $\lambda = 20$, $K = 1$ predictor loop case.

We finally compare the closed-loop performance of the predictive fusion coder (designed by Algorithm 3), with three competing techniques - (memoryless) vector quantizer (VQ) and predictive VQ (which compress all sources jointly i.e. they are joint compression techniques) and (memoryless) fusion coder (FC). The joint compression (here, VQ) techniques are compelled to retrieve all the compressed data i.e. $R_r = R_s$ and can reduce

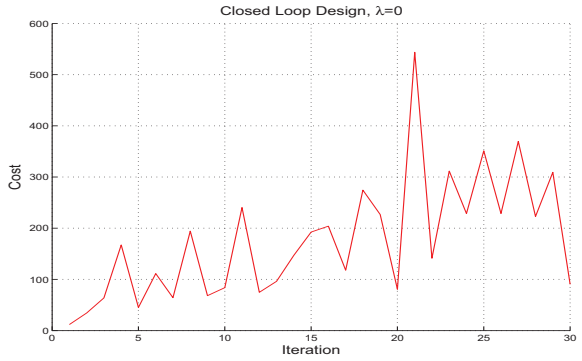

Figure 5.10. PFC Design by Closed Loop, $\lambda = 0$

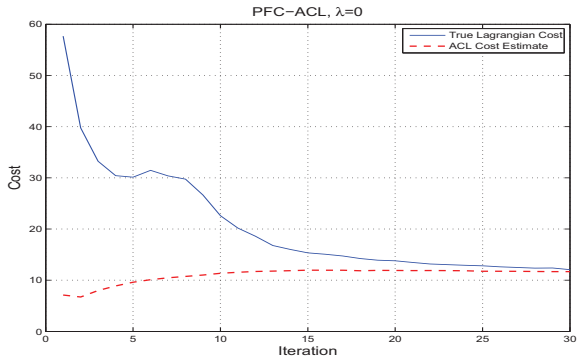

Figure 5.11. PFC Design by ACL, $\lambda = 0$

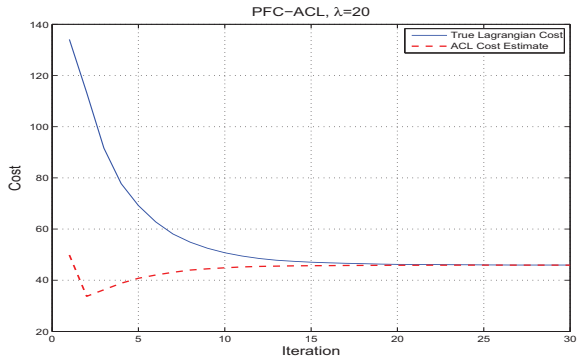

Figure 5.12. PFC Design by ACL, $\lambda = 20$, $K = 1$ prediction loop

retrieval rate (time) only by reducing the compression (storage) rate (see Figure 5.13).

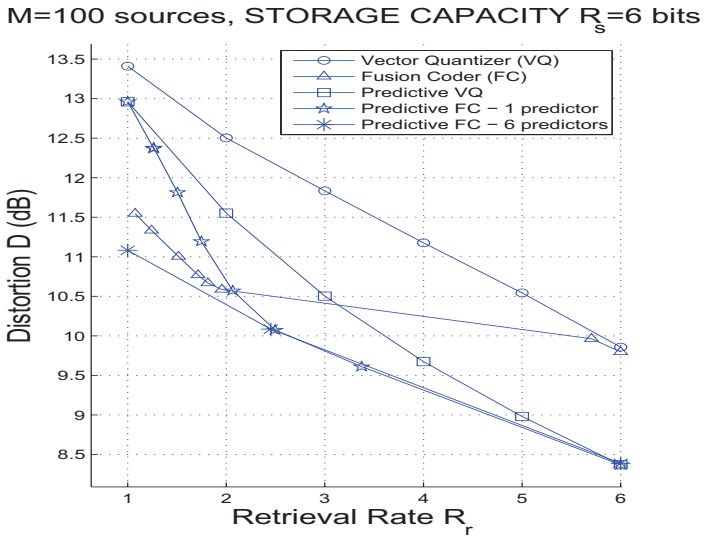

Figure 5.13. Joint Compression (VQ) versus Fusion Coding

We note significant gains of the predictive fusion coding over the three competing methods. We obtain distortion gains about 2 dB over VQ, 1.5 DB over the FC and 1 dB over predictive VQ at $R_r = 2$ bits.The bigger gains over VQ and FC are possible because time-correlations were exploited by the predictive FC. A nearly 1.5X reduction in retrieval rate was achieved over the predictive vector quantizer, given the same distortion of $D = 10.5$ dB. In the very low rate region, we note that the FC has a slightly better performance than predictive FC. Since we constrained the predictive FC to have only prediction loop, in the low rate region $|\mathcal{S}_P \cup \mathcal{S}(\mathbf{q})| = |\mathcal{S}_P| = 1$, $\forall \mathbf{q}$. As we allow more prediction loops, there would be greater freedom in designing the query bit-selector $\mathcal{S}(\mathbf{q})$, and the performance gets better. Note that in Figure 5.13, we show a performance plot with 6 prediction loops as well.

Chapter 6

Conclusion and Future Work

In this thesis, two important applications of source coding in database management were studied, namely fast search in high-dimensional multimedia databases and efficient storage/fast retrieval (fusion coding) in correlated source databases. We now briefly recap the main contributions and indicate directions for future research.

6.1 Main Results

The main results are as follows

- We presented a VQ/clustering based index for exact k-nearest neighbor (kNN) search in high-dimensional databases, which promises to combat the curse of dimensionality. The failure of traditional tree-based indexing has led to compression based indexes, wherein kNN search is performed over a compressed data-set. However, techniques such as VA-File, ignore correlations across feature dimensions. On the other hand, known clustering based methods eliminate irrelevant clusters through (ultimately weak) distance bounds to rectangles and spheres bounding clusters. We proposed a cluster distance bound (the hyperplane bound), where

query-cluster distance is bounded by projecting the query onto a separating hyperplane boundaries and complementing with the cluster-hyperplane distance (cluster support). Experiments conducted on several large real data-sets reveal substantial gains over known indexing methods.

- The Euclidean distance performs a perceptually poor approximation of image similarities. Hence, similarity searching over image databases is often an iterative process where after each round or periodically, user feedback is used to update a Mahalanobis distance measure. We derived a basic property of point-to-hyperplane Mahalanobis distance, which enables efficient recalculation of such query-cluster distance bounds as the Mahalanobis weight matrix is varied. Thus, retrieval of perceptually relevant images is possible, since Mahalanobis weights can be tuned to user perceptions by feedback.

- We improved upon the hyperplane bound, by recasting the estimation of query-cluster distance as the minimization of a norm subject to linear constraints (induced by the separating hyperplanes). By this formulation, we search over all separating hyperplanes and all possible points of intersection between the separating hyperplanes. Norm minimization over linear constraints is a convex optimization problem and can be solved in polynomial time (polynomial in number of dimensions d and constraints M). Moreover, since Mahalanobis distances (with positive definite weight matrices) are also norms, cluster distance estimation is also possible under changing weight matrices. This results in significant improvements over hyperplane bounds for both Euclidean and Mahalanobis distances, with a small (polynomial in d and M) increase in computations.

- We identified an important application of source coding in correlated source databases, wherein correlated sources need to be fusion stored for efficient future retrieval of select subsets. Given only statistics of future queries, a tradeoff between storage rate (space), retrieval rate (time) and distortion (quality) needs to be optimized.

We presented a Fusion Coder (FC) to optimize this tradeoff, derived its optimality properties and presented an iterative algorithm for its design. Additionally, we present initialization heuristics to avoid poor local minima in iterative design. Finally, we suggest methods to reduce design complexity growth with query-set size and adapt to new queries.

- By imposing constraints on FC modules, we obtained a "Shared Descriptions" framework, where tradeoffs between storage rate, retrieval rate, distortion and complexity are possible. The proposed Shared Descriptions Fusion Coder includes FC as a special case and with bounds on design complexity, allows scalability to large storage rates. This allows for better tradeoffs between distortion and retrieval rate and provides significant gains over several naive quantization schemes.

- We developed a low-complexity Predictive Fusion Coder to exploit spatio-temporal correlations in data-streams. Predictive coding avoids the exponentially high design complexity entailed by fusion coding over large blocks of data. However, the design of optimal PFCs is complicated by the feedback loop and the need to accommodate a prediction loop for every element of a (possibly exponentially) large query set at the encoder. We imposed constraints on the number of prediction loops and designed PFCs through the Asymptotic Closed Loop principle. Huge gains over memoryless FC and joint compression were observed.

6.2 Future Directions

We conclude this chapter by discussing open research issues and future work.

6.2.1 Fusion Coding

While a number of FC design issues were tackled, we believe a substantial amount of work lies ahead to make fusion coding more practically applicable. On the theoretical side, while a single letter characterization of an achievable rate region for asymptotically lossless fusion coding was identified in [82], it is unknown if this is also a strict characterization. An equally important aspect is the study and characterization of rate-distortion functions for lossy fusion codes.

On the practical side, a very important objective is to reduce the sensitivity to initialization. While the heuristics for bit-selector initialization were discussed, we note multiple initializations were still necessary. Hence, a globally optimal design methodologies such as deterministic annealing [106], which is initialization independent, would be more suitable. A second open issue is in how variable length fusion coding can be performed. Different indexes occur with unequal probability and could be assigned different code-lengths, and thus save on overall storage. However, if known variable length code designs were directly applied, we notice an immediate conflict, as the freedom for query specific bit-selection is now compromised. Thus, variable length fusion coding would need to optimize the trade-off between storage and retrieval rates. Other open problems are in multi-stage/scalable fusion coder design, i.e. for users with varying bandwidths and fusion coding for error-resilient retrieval i.e. when channel errors are possible.

At very high compression (storage) rates (R_s), there might be a need to resort to transform coding techniques, where a linear transform (followed by scalar quantization) is combined with bit-selection to optimize the storage-retrieval trade-off. Possibly, by combining transform fusion coding with ideas from [91], VA$^+$-File [61], and VA-Stream [81], a framework for similarity search over an arbitrary subset of data-streams might be possible.

6.2.2 Large Scale Distributed Quantization

Distributed quantization [107][108] is often a necessary tool for energy efficient data collection from sensor networks. However, the design of distributed quantizers suffers from the "curse of dimensionality" i.e. exponential growth of storage requirement with the size of the network and transmission rates of the sources. An intuitive approach toward reducing decoder storage complexity would be to identify (select) bits that are relevant to a particular source reconstruction and trade against reconstruction quality. Thus, a bit-selector could be used to trade-off the decoder storage against distortion, much as was performed in Shared Descriptions Fusion Coding [101]. More generally, we could consider a scenario where fusion encoding is performed *in the network*, thereby eliminating a high complexity encoder at the fusion center. The computational complexity of such an encoder is now completely distributed across the network, for a small performance in penalty. However, the search space is exponentially large and additional constraints/heuristics would be necessary to obtain practical designs.

6.2.3 Similarity Search

In indexing, we note the cluster-distance bounding approach is actually a fast database decoding strategy, while the clustering/VQ is the encoding counterpart. Next, note that the optimal cluster distance bounding by norm minimization has a striking similarity with VA-File decoding. While we believe that GLA/K-means optimizes the encoding for several real data-sets, this does not seem to be the case for synthetic data-sets, such as the uniform data-set, where the VA-File provides better performance. It would be of interest to identify why the disparity, if any, and thus unify and optimize the encoding procedure as well.

Often image similarities are modelled through a transformation induced by a kernel. While kernels were initially introduced in the context of support vector machine design [109], their use in modelling image similarity is also known [52]. Extending our

indexing strategy to handle such transformations would be a next step. A promising approach would build on the secondary (Gramm-Schmidt) feature extraction and indexing procedure described for kernel VA-Files [110] [111]. Kernels are also used in modelling similarity in other databases, such as the use of the Fisher kernel in protein databases [112,] and audio classification [113]. Hence, successful indexing of kernels could have wide reaching impact. Finally, we note that while the Gaussian kernel is commonly used for the design of support vector machines, the transformations it induces does not change nearest neighbors to a query vector. However, indexing this transform space, by indexing the secondary features extracted, could have different clustering (encoding) and hence lead to different IO performance from the original feature space. It could be worthwhile to consider if and under what conditions indexing through such similar induced transformations (that do not change the nearest neighbors) leads to better IO performance.

In practice, approximate similarity search is often preferred given its reduced IO complexity as well as the inherent approximation of user perceptions by feature vectors and distances. Several approaches to approximate kNN search based on VQ/clustering are known [53] [54], the index design is often initialization dependent. By suitably combining with a globally optimal design framework, such as deterministic annealing [106], better designs would be possible.

Appendix A

A Contribution Towards Reducing Computations in the VA-File

The first of phase of VA-File based indexing eliminates all those feature vectors whose query-distance lower bound $d_{LB}(\mathbf{q}, \mathbf{x})$ is greater than the k^{th} highest upper bound $d^k_{UB,sort}(\mathbf{q})$. Then, in the second phase we access survivors in increasing order of lower bounds. The search stops when the lower bounds are greater than k^{th} smallest true distance i.e. distance of the query to the k^{th} nearest neighbor.

Let $\mathcal{X}$ be the whole data set. Let $\mathcal{X}_I$ be the elements eliminated after the first round of VA-File filtering. Let $\mathcal{X}_{II}$ be the elements eliminated after the second round of VA-File filtering i.e. assuming only the second phase happens. Let $d_{kNN}(\mathbf{q})$ be the distance of k^{th} nearest neighbor $\mathbf{x}_k$ from the query $\mathbf{q}$.

By definition,

$$\forall \mathbf{x} \in \mathcal{X}_{II}, d(\mathbf{x}, \mathbf{q}) \geq d_{kNN}(\mathbf{q}) \tag{A.1}$$

and

$$\forall \mathbf{x} \in \mathcal{X}_{I}, d(\mathbf{x}, \mathbf{q}) \geq d^k_{UB,sort}(\mathbf{q}) \tag{A.2}$$

Additionally,

$$d_{kNN}(\mathbf{q}) \leq d^k_{UB,sort}(\mathbf{q}) \tag{A.3}$$

This is true since the converse would imply a contradiction, i.e. if $d^k_{UB,sort}(\mathbf{q}) \leq d_{kNN}(\mathbf{q})$, it would mean that there are at least k elements whose distance is smaller than $d_{kNN}(\mathbf{q})$ i.e. $\mathbf{x}_k$ is NOT the k^{th} nearest neighbor.

$$\therefore \mathcal{X}_I \subseteq \mathcal{X}_{II} \tag{A.4}$$

This shows that the phase I of VA-file can be avoided without any performance degradation and consequently upper bound calculation can be eliminated, reducing the total number of computations by almost 50%.

Bibliography

[1] M. L. Kherfi, D. Ziou, and A. Bernardi, "Image retrieval from the world wide web: Issues, techniques, and systems," *ACM Computing Surveys*, vol. 36, no. 1, pp. 35–67, 2004.

[2] T. Smith and J. Frew, "Alexandria digital library," *Communications of the ACM*, vol. 38, no. 4, pp. 61–62, 1995.

[3] A. Graham, H. Garcia-Molina, A. Paepcke, and T. Winograd, "Time as essence for photo browsing through personal digital libraries," in *Proc. of Joint Conf. on Digital Libraries (JCDL)*, 2002, pp. 326–335.

[4] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system," *Computer*, vol. 28, no. 9, pp. 23–32, 1995.

[5] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz, "Efficient and effective querying by image content," *Journal of Intelligent Information Systems*, vol. 3, no. 3/4, pp. 231–262, 1994.

[6] W. Ma and B. Manjunath, "Netra: A toolbox for navigating large image databases," *Multimedia Systems*, vol. 7, no. 3, pp. 184–198.

[7] N. Snavely, S. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3d," in *Proc. of Intl. Conf. on Computer Graphics and Interactive Techniques (CGIT)*, 2006, pp. 835–846.

[8] S. Bhagavathy and B. Manjunath, "Modeling and detection of geospatial objects using texture motifs," *IEEE Trans. On Geoscience And Remote Sensing*, vol. 44, no. 12, p. 3706, 2006.

[9] F. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Protopapas, "Fast nearest neighbor search in medical image databases," in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, 1996, pp. 215–226.

[10] C. Faloutsos, *Searching in Multimedia Databases By Content*. Kluwer Academic Press, 1996.

[11] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. of SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2003, pp. 137–146.

[12] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail," in *Proc. of Workshop on Learning for Text Categorization (WLTC)*, vol. 460, 1998.

[13] J. Ha, C. Rossbach, J. Davis, I. Roy, H. Ramadan, D. Porter, D. Chen, and E. Witchel, "Improved error reporting for software that uses black-box components," in *Proc. of SIGPLAN Conf. on Programming Language Design and Implementation (SIGPLAN)*, 2007, pp. 101–111.

[14] G. Wittel and S. Wu, "On attacking statistical spam filters," in *Proc. of Conf. on Email and Anti-Spam (CEAS)*, 2004.

[15] R. Ramakrishnan and J. Gerhke, *Database Management Systems, Second Edition*. McGraw-Hill Science/Engineering/Math Publishers, 2001.

[16] B. Seeger, P. Larson, and R. McFayden, "Reading a set of disk pages," in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, 1993, pp. 592–603.

[17] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. of Intl. Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002, pp. 88–97.

[18] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *Proc. of Intl. Conf. on Foundations of Data Organization and Algorithms (FODO)*, 1993, pp. 69–84.

[19] D. Rafiei and A. Mendelzon, "Similarity-based queries for time series data," in *Proc. of SIGMOD Intl. Conf. on Management of Data (SIGMOD)*, 1997, pp. 13–25.

[20] Y. Zhu and D. Shasha, "Statstream: Statistical monitoring of thousands of data streams in real time," in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, 2002, pp. 358–369.

[21] S. Papadimitriou, A. Brockwell, and C. Faloutsos, "Adaptive, hands-off stream mining," in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, 2003, pp. 560–571.

[22] Y. Wu, D. Agrawal, and A. E. Abbadi, "A comparison of dft and dwt based similarity search in time-series databases," in *Proc. of Inl. Conf. on Information and Knowledge Management (CIKM)*, 2000, pp. 488–495.

[23] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases," *ACM Trans. on Database Syst.*, vol. 27, no. 2, pp. 188–228, 2002.

[24] T. Sikora, "The MPEG-7 visual standard for content description-an overview," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 696–702, 2001.

[25] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. of Intl. Conf. on Computer Vision (ICCV)*, vol. 2, 1999, pp. 1150–1157.

[26] M. Bober, "MPEG-7 visual shape descriptors," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 716–719, 2001.

[27] B. Manjunath, J.-R. Ohm, V. Vasudevan, and A. Yamada, "Color and texture descriptors." *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 703–715, June 2001.

[28] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc. New York, NY, USA, 1986.

[29] A. Moffat and J. Zobel, "Self-indexing inverted files for fast text retrieval," *ACM Trans. on Information Systems*, vol. 14, no. 4, pp. 349–379, 1996.

[30] P. Ciaccia and M. Patella, "Approximate similarity queries: A survey." *CSITE-08-01 Technical Report*, May 2001.

[31] J. Robinson, "The KDB-tree: a search structure for large multidimensional dynamic indexes," in *Proc. of SIGMOD Intl. Conf. on Management of Data (SIGMOD)*, 1981, pp. 10–18.

[32] S. Lawrence, C. Giles, and K. Bollacker, "Digital libraries and autonomous citation indexing," *Computer*, pp. 67–71, 1999.

[33] D. de Solla Price, "Networks of scientific papers," *Science*, vol. 149, no. 3683, pp. 510–515, 1965.

[34] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[35] A. Guttman, "R-trees: A dynamic index structure for spatial searching." in *Proc. of SIGMOD Intl. Conf. on Management of Data (SIGMOD)*, 1984, pp. 47–57.

[36] Y. Manolopoulos, A. Nanopoulos, A. Papadopoulos, and Y. Theodoridis, "R-trees have grown everywhere," *http://www.rtreeportal.org/ pubs/MNPT03.pdf.*

[37] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles," in *Proc. of SIGMOD Intl. Conf. on Management of Data (SIGMOD)*, 1990, pp. 322–331.

[38] S. Arya and D. M. Mount, "Algorithms for fast vector quantizaton." in *Proc. of Data Compression Conference (DCC)*, 1993, pp. 381–390.

[39] D. A. White and R. Jain, "Similarity indexing with the SS-tree," in *Proc. of Intl. Conf. on Data Engineering (ICDE)*, 1996, pp. 516–523.

[40] N. Katayama and S. Satoh, "The SR-tree: An index structure for high-dimensional nearest neighbor queries." in *Proc. of SIGMOD Intl. Conf. on Management of Data (SIGMOD)*, May 1997, pp. 369–380.

[41] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces." in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, 1997, pp. 426–435.

[42] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ: Princeton University Press, 1961.

[43] R. Weber, H. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces." in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, August 1998, pp. 194–205.

[44] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima, "The A-tree: An index structure for high-dimensional spaces using relative approximation," in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, September 2000, pp. 516–526.

[45] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" pp. 217–235.

[46] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional spaces," pp. 420–434.

[47] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* John Wiley, 1991.

[48] A. Hinneburg, C. C. Aggarwal, and D. A. Keim, "What is the nearest neighbor in high dimensional spaces?" in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, September 2000, pp. 506–515.

[49] C. C. Aggarwal and P. S. Yu, "The iGrid index: Reversing the dimensionality curse for similarity indexing in high dimensional space," in *Proc. of SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2000, pp. 119–129.

[50] B. U. Pagel, F. Korn, and C. Faloutsos, "Deflating the dimensionality curse using multiple fractal dimensions," in *Proc. of Intl. Conf. on Data Engineering (ICDE)*, 2000, pp. 589–598.

[51] T. Huang and X. S. Zhou, "Image retrieval with relevance feedback: From heuristic weight adjustment to optimal learning methods," in *Proc. of Intl. Conf. on Image Processing (ICIP)*, vol. 3, 2001, pp. 2–5.

[52] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon, "Information-theoretic metric learning," in *Proc. of Intl. Conf. on Machine learning (ICML)*, 2007, pp. 209–216.

[53] E. Tuncel and K. Rose, "Towards optimal clustering for approximate similarity searching." in *Proc. of Intl. Conf. on Multimedia and Expo (ICME)*, vol. 2, August 2002, pp. 497–500.

[54] E. Tuncel, H. Ferhatosmanoglu, and K. Rose, "VQ-Index: An index structure for similarity searching in multimedia databases." in *ACM Multimedia*, 2002, pp. 543–552.

[55] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. E. Abbadi, "Approximate nearest neighbor searching in multimedia databases." in *Proc. of Intl. Conf. on Data Engineering (ICDE)*, April 2001, pp. 503–511.

[56] E. Tuncel, P. Koulgi, and K. Rose, "Rate-distortion approach to databases: Storage and content-based retrieval," *IEEE Trans. on Information Theory*, vol. 50, no. 6, pp. 953–967, 2004.

[57] L. Zhu, A. Zhang, A. Rao, and R. Srihari, "Keyblock: an approach for content-based image retrieval," in *ACM Multimedia*, 2000, pp. 157–166.

[58] E. Tuncel, "Capacity/Storage Tradeoff in High-Dimensional Identification Systems," in *Proc. of Intl. Symposium on Information Theory (ISIT)*, 2006, pp. 1929–1933.

[59] K. Chakrabarti and S. Mehrotra, "Local dimensionality reduction: A new approach to indexing high dimensional spaces." in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, September 2000, pp. 89–100.

[60] K. Vu, K. Hua, H. Cheng, and S. Lang, "A non-linear dimensionality-reduction technique for fast similarity search in large databases," in *Proc. of SIGMOD Intl. Conf. on Management of Data (SIGMOD)*, 2006, pp. 527–538.

[61] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. E. Abbadi, "Vector approximation based indexing for non-uniform high dimensional data sets," in *Proc. of Inl. Conf. on Information and Knowledge Management (CIKM)*, 2000, pp. 202–209.

[62] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[63] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[64] S. Berchtold, C. Bohm, H. V. Jagadish, H. P. Kriegel, and J. Sander, "Independent Quantization: An index compression technique for high-dimensional data spaces," in *Proc. of Intl. Conf. on Data Engineering (ICDE)*, 2000, pp. 577–588.

[65] J. Chen, C. Bouman, and J. Allebach, "Fast image database search using tree-structured VQ." in *Proc. of Intl. Conf. on Image Processing (ICIP)*, vol. 2, 1997, pp. 827–8305.

[66] J. Chen, C. Bouman, and J. Dalton, "Hierarchical browsing and search of large image databases." *IEEE Trans. on Image Processing*, vol. 9, no. 3, pp. 442–455, March 2000.

[67] N. Koudas, B. C. Ooi, H. T. Shen, and A. K. H. Tung, "LDC: Enabling search by partial distance in a hyper-dimensional space." in *Proc. of Intl. Conf. on Data Engineering (ICDE)*, 2004, pp. 6–17.

[68] C. Yu, B. C. Ooi, K. L. Tan, and H. V. Jagadish, "Indexing the distance: An efficient method to knn processing." in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, September 2001, pp. 421–430.

[69] S. Berchtold, C. Bohm, and H. Kriegel, "The Pyramid-technique: Towards breaking the curse of dimensionality," in *Proc. of SIGMOD Intl. Conf. on Management of Data (SIGMOD)*, 1998, pp. 142–153.

[70] C. Li, E. Chang, H. Garcia-Molina, and G. Wiederhold, "Clustering for approximate similarity search in high-dimensional spaces," *IEEE Trans. on Knowledge and Data Engineering*, vol. 14, no. 4, pp. 792–808, July–August 2002.

[71] V. Castelli, A. Thomasian, and C. S. Li, "CSVD: Clustering and singular value decomposition for approximate similarity search in high-dimensional spaces." *IEEE Trans. on Knowledge and Data Engineering*, vol. 15, no. 3, pp. 671–685, 2003.

[72] R. Weber and K. Böhm, "Trading quality for time with nearest neighbor search." in *EDBT*, 2000, pp. 21–35.

[73] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing." in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, September 1999, pp. 518–529.

[74] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. of Symposium on Computational Geometry (SOCG)*, 2004, pp. 253–262.

[75] P. Jain, B. Kulis, and K. Graumann, "Fast image search for learned metrics," in *Proc. of Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[76] H. Jin, B. C. Ooi, H. T. Shen, C. Yu, and A. Zhou, "An adaptive and efficient dimensionality reduction algorithm for high-dimensional indexing." in *Proc. of Intl. Conf. on Data Engineering (ICDE)*, March 2003, pp. 87–98.

[77] P. Ciaccia and M. Patella, "PAC nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces," in *Proc. of Intl. Conf. on Data Engineering (ICDE)*, 2000, pp. 244–255.

[78] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. on Information Theory*, vol. 19, no. 4, pp. 471–480, Jul 1973.

[79] A. D. Wyner and J. Ziv, "The rate-distortion function for source coding with side-information at the decoder," *IEEE Trans. on Information Theory*, vol. 22, pp. 1–11, 1976.

[80] S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): Design and construction," in *Proc. of Data Compression Conference (DCC)*, 1999, pp. 158–167.

[81] X. Liu and H. Ferhatosmanoglu, "Efficient k-NN search on streaming data series," in *Proc. of Intl. Symposium on Advances in Spatial and Temporal Database (SSTD)*, 2003, pp. 83–101.

[82] J. Nayak, S. Ramaswamy, and K. Rose, "Correlated source coding for fusion storage and selective retrieval," in *Proc. of Intl. Symposium on Information Theory (ISIT)*, 2005, pp. 92–96.

[83] T. Han and K. Kobayashi, "A unified achievable rate region for a general class of multiterminal source coding systems," *IEEE Tran. on Information Theory*, vol. 26, no. 3, pp. 277–288, May 1980.

[84] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons, 1996.

[85] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases." in *Proc. of SIGMOD Intl. Conf. on Management of Data (SIGMOD)*, 1996, pp. 103–114.

[86] Y. Ishikawa, R. Subramanya, and C. Faloutsos, "Mindreader: Querying databases through multiple examples," in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, August 1998, pp. 218–227.

[87] Y. Rui and T. Huang, "Optimizing learning in image retrieval," *Proc. of Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 1236–1243, 2000.

[88] Y. Rui, T. Huang, and S. Mehrotra, "Content-based image retrieval with relevance feedback in MARS," in *Proc. of Intl. Conf. on Image Processing (ICIP)*, vol. 2, 1997, pp. 815–818.

[89] E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," *Advances in Neural Information Processing Systems*, vol. 15, pp. 505–512, 2003.

[90] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning a mahalanobis metric from equivalence constraints," *Journal of Machine Learning Research*, vol. 6, pp. 937–965, 2005.

[91] Y. Sakurai, M. Yoshikawa, R. Kataoka, and S. Uemura, "Similarity search for adaptive ellipsoid queries using spatial transformation," in *Proc. of Intl. Conf. on Very Large Databases (VLDB)*, 2001, pp. 231–240.

[92] D. Goldfarb and S. Liu, "An O(n^3l) primal interior point algorithm for convex quadratic programming," *Mathematical Programming*, vol. 49, no. 1, pp. 325–340, 1990.

[93] S. Kapoor and P. Vaidya, "Fast algorithms for convex quadratic programming and multicommodity flows," in *Proc. of Symposium on Theory of Computing (STOC)*, 1986, pp. 147–159.

[94] S. Ramaswamy and K. Rose, "Adaptive cluster-distance bounding for similarity search in image databases," in *Proc. of Intl. Conf. on Image Processing (ICIP)*, vol. 6, 2007, pp. 381–384.

[95] ——, "Fast adaptive Mahalanobis-distance based search and retrieval in image databases," in *Proc. of Intl. Conf. on Image Processing (ICIP)*, 2008, pp. 181–184.

[96] CalTech 101 Data-set, *www.vision.caltech.edu/Image_Datasets/Caltech101/*.

[97] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[98] R. Cristescu and M. Vetterli, "On the optimal density for real-time data gathering of spatio-temporal processes in sensor networks," in *Proc. of Intl. Conf. on Information Processing for Sensor Networks (IPSN)*, 2005, pp. 159–164.

[99] R. Duda, P. Hart, and D. G. Stork, *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[100] J. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[101] S. Ramaswamy and K. Rose, "Shared Descriptions Fusion Coding for Storage and Selective Retrieval of Correlated Sources," in *Proc. of Data Compression Conference (DCC)*, 2008, pp. 262–271.

[102] S. Ramaswamy, J. Nayak, and K. Rose, "Code design for fast selective retrieval of fusion stored sensor network/time series data," in *Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, 2007, pp. 1005–1008.

[103] S. Ramaswamy and K. Rose, "Predictive fusion coding of spatio-temporally correlated sources," in *Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 2509–2512.

[104] H. Khalil, K. Rose, and S. L. Regunathan, "The asymptotic closed-loop approach to predictive vector quantizer design with application in video coding," *IEEE Trans. on Image Processing*, vol. 10, no. 1, pp. 15–23, Jan 2001.

[105] H. Khalil and K. Rose, "Predictive vector quantizer design using deterministic annealing," *IEEE Trans. on Signal Processing*, vol. 51, no. 1, pp. 244–254, 2003.

[106] K. Rose, "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems," *Proc. of IEEE*, vol. 86, no. 11, pp. 2210–2239, 1998.

[107] M. Fleming, Q. Zhao, and M. Effros, "Network vector quantization," *IEEE Trans. on Information Theory*, vol. 50, pp. 1584–1604, Aug 2004.

[108] A. Saxena and K. Rose, "Distributed predictive coding for spatio-temporally correlated sources," in *Proc. of Intl. Symposium on Information Theory (ISIT)*, 2007, pp. 1506–1510.

[109] B. Scholkopf and A. Smola, *Learning with kernels*. MIT Press Cambridge, Mass, 2002.

[110] J. Peng and D. Heisterkamp, "Kernel indexing for relevance feedback image retrieval," in *Proc. of Intl. Conf. on Image Processing (ICIP)*, vol. 1, 2003, pp. 733–736.

[111] D. R. Heisterkamp and J. Peng, "Kernel VA-files for nearest neighbor search in large image databases," in *Proc. of Intl. Conf. on Artificial Neural Networks (ICANN)*, June 2003.

[112] T. Jaakkola, M. Diekhans, and D. Haussler, "A discriminative framework for detecting remote protein homologies," *Journal of Computational Biology*, vol. 7, no. 1-2, pp. 95–114, 2000.

[113] P. Moreno and R. Rifkin, "Using the fisher kernel method for web audio classification," in *Proc. of Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 6, 2000, pp. 2417–2420.