DISSERTATION

ADAPTABLE TEXT AND IMAGE RETRIEVAL SYSTEMS USING RELEVANCE FEEDBACK

Submitted by

Jaime Salazar Tamez

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2006

COLORADO STATE UNIVERSITY

July 17, 2006

WE HEREBY RECOMMEND THAT THE **DISSERTATION** PREPARED UNDER OUR SUPERVISION BY **JAIME SALAZAR TAMEZ** ENTITLED **ADAPTABLE TEXT AND IMAGE RETRIEVAL SYSTEMS USING RELEVANCE FEEDBACK** BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

Prof. Anura P. Jayasumana

Prof. Edwin K. Chong

Prof. Yashwant K. Malaiya

Prof. Mahmood R. Azimi-Sadjadi
Adviser

Prof. Anthony A. Maciejewski
Department Head/Director

ABSTRACT OF DISSERTATION

ADAPTABLE TEXT AND IMAGE RETRIEVAL SYSTEMS USING RELEVANCE FEEDBACK

The purpose of this work is to develop adaptable and robust search and retrieval systems for both text and image databases. The proposed systems referred to as "model-reference text retrieval system (MRTRS)", and "model-reference image retrieval system (MRIRS)", are inspired from the well-known theory of model-reference adaptive control systems. A learning methodology that incorporates users' information and expertise via relevance feedback to improve the relevancy of solutions is presented. This methodology relies on a limited number of single-term as well as multi-term queries for text databases and on a limited number of training samples for image databases.

The learning in MRTRS involves three phases that are: (i) initial model-reference learning, (ii) model-reference following, and (iii) relevance feedback learning from expert users. The initial model-reference learning involves capturing the behavior of a reference text retrieval model, when this is available, or simply the results of an indexing system. The model-reference following is implemented in dynamic conditions, when documents are added, deleted or updated. New relevance feedback learning methods are developed for single-term and multi-term queries from multiple users using either score-based or click-through selection feedback. Additionally, as a by-product of our system, we account for the ability to cluster queries according to their content. This feature allows the system to provide suggestion feedback to the users to enhance their original query. The developed MRTRS system is tested on

a text database that encompasses several HP-products with over 60,000 documents and 130,000 terms.

The learning in MRIRS involves two phases that are: (i) model-reference learning, and (ii) relevance feedback learning from expert users. Again, the model-reference learning involves capturing the behavior of a reference image retrieval model, when class information of the images is available. Relevance feedback learning is implemented using the information on the relative positions of some relevant images. We propose two different MRIRS retrieval systems that can operate in an online fashion or in a batch mode. The first retrieval system uses several regulators working independently from each other, though they are influenced by the users' feedback. Each regulator transforms the original query image into a mapped version with the goal of driving the error signal between the output of the retrieval system and that of the expert users to zero, hence meeting the users requirements. The second system uses a single regulator to deliver a single mapped version of the submitted query image. Although, structurally more complex, the multiple regulator image retrieval system involves lesser number of parameters to fine-tune in response to either model reference or relevance feedback learning. The implementation of each system via a structurally adaptable neural network in which relevance feedback learning from multiple expert users optimally maps the original query is presented. The learning algorithms are thoroughly tested on a domain-specific image database, which encompasses a wide range of underwater mine-like and non-mine-like objects captured with an electro-optical sensor.

Jaime Salazar Tamez
Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, Colorado 80523
Fall 2006

ACKNOWLEDGEMENTS

First and foremost, I want to express my profound gratitude to Dr. Mahmood R. Azimi-Sadjadi, for his support during my stay at CSU. Thanks to his guidance, I have learnt that for doing research one needs perseverance and patience to pursue an idea until fruition. I consider him a friend, who is always willing to listen.

I am indebt with my committee members: Dr. Anura P. Jayasumana, Dr. Edwin K. Chong, and Dr. Yashwant K. Malaiya who were willing to dedicate part of their valuable time reviewing this dissertation and attending my presentation.

I would like to thank Dr. Sassan Sheedvash at Hewlett Packard in Boise-ID who helped me during the course of research and experimentation on the developed text retrieval and gave me invaluable recommendations. The research on text retrieval was supported by HP-Boise, Idaho management and business teams under contract # 50B000553.

I express my gratitude to my colleagues and friends at the 'Instituto Tecnologico y de Estudios Superiores de Monterrey' (ITESM) in Toluca, Mexico, in particular to Dr. Roberto Rueda, Mrs. Sandra Ortiz, Dr. Jose Carlos Miranda, Dr. Luciano Chirinos, and Dr. Jesús Gutiérrez, who have been very helpful and cooperative during my studies. I thank them for their continuous support they provided me.

I also want to thank my fellow students, Ali Pezeshki, Marc Robinson, Arta Jamshidi, Amanda Falcone, Gordon Wichern, and Kumar Srinivasan, with whom I spent part of my life at CSU.

To my wife Sonia Patricia, my Sons: Daniel and Luis, and my parents

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Statement and Motivations

The focus of most general-purpose text retrieval systems (TRS) is to apply robust search and content matching to deal effectively and consistently with an overwhelmingly large volume of information. In these systems, the user typically modifies and enhances the query text in a subjective manner in order to narrow the domain of the search. The search process typically culminates at a list of the documents from which the user identifies, either implicitly or explicitly, the most relevant ones after navigating or browsing through the list in the order of the documents' "retrieval status values" or relative scores [1]. This subjective query modification does not allow for the incorporation of the user expertise or feedback to influence the suggested solutions. Moreover, identification of an optimum query that carries the required concept is difficult or sometimes impossible, even for expert users. We propose to develop adaptable text [2–4] and image search/retrieval systems for special-purpose applications allowing only expert users to incorporate their suggested solutions via relevance feedback.

Special-purpose text and image databases are commonly found in a variety of areas such as medicine [5,6] and military. These databases typically contain information in the form of text, images, or text and images combined in the same document. Due to the development of efficient technologies to transmit and store information,

these databases may contain thousand or even hundreds of thousands of documents. Querying these databases is performed regularly by an expert user who has the possibility to search through the list of retrieved documents but cannot typically modify the retrieval system when the list is unsatisfactory. Although there exist methods [7] that aid a particular user to search through a list of retrieved documents in an efficient way, few of them keep what has been learnt from the expert user to enhance the performance of the retrieval process for subsequent sessions by the same or other expert users.

Relevance feedback was originally introduced in the vector space model for TRS by Rocchio [8]. In this model, documents are considered as vectors in an n-dimensional space where each element represents the weight or importance of a term or a stemmed word in the document. Relevance feedback is a mechanism through which an original query is automatically modified in order to improve retrieval efficiency based on information extracted from labeled lists of relevant and non-relevant documents identified by a user. Two commonly used methods that implement relevance feedback are query expansion [9] and query point movement [10]. In query expansion, a new query is formed by augmenting the original query using relevant terms extracted from the set of relevant documents; while, in the query point movement, a new query is formed by inducing a little perturbation to the original query. This perturbation is obtained from the set of relevant and non-relevant documents. Most often, these two methods rely on Rocchio's formula [11] and work fairly well for general (not special-purpose) text or image databases where a reasonable solution is also acceptable. But, they do not offer good solutions for special-purpose databases where an accurate result list is expected, unless several rounds of query modification is performed. Moreover, in general-purpose databases, a user may be willing to spend several rounds of querying until he/she gets the desired information. Whereas, in a special-purpose database, an expert user expects to receive an accurate list of documents immediately after he/she

submits the initial query. Therefore, learning methods used in the retrieval systems for special-purpose databases must incorporate relevant information efficiently and with minimum involvement from expert users.

In summary, current TRS tools are typically designed for general purpose search and retrieval applications and are not suited for special-purpose scenarios where the domain of applications is limited to specific databases with expert end-users. Examples of such systems are in customer support of various corporations like that in our study, educational institution databases, hospital databases [12–15], homeland security and other similar applications where the accuracy and relevancy of search results and system adaptability are of utmost importance. In these environments, TRS must continuously learn from the users under hugely under-specified relevance feedback. In other words, in contrast to the general-purpose TRS where abundant number of feedback in terms of click-through selection are typically available for every query, it is important to accurately meet the user requirements when queries normally do not receive numerous feedback. Additionally, it is crucial to capture the expertise of all the users by adapting the system parameters for more refined future searches using an appropriated relevance feedback mechanism.

The need to develop a well-established framework for the problem of learning from expert users via relevance feedback was the incentive that drove us to do this research. In particular, there are definite needs for rigorous multi-term multi-user relevance feedback learning for TRS that can accurately capture a set of input/output relations even in contradictory and incomplete scenarios and for linear and nonlinear query mapping cases. In what follows, we provide a survey of related previous work on text and image retrieval systems.

1.2 Previous Work

1.2.1 Survey on Text Retrieval Systems

Text retrieval systems that allow for user contribution typically utilize "relevance feedback" [16–19] from the users to modify the original query in order to meet the users' requirements and improve the retrieval efficiency. It is expected that the new query would deliver a more refined list of documents than that delivered by the original query. As mentioned before, a mechanism to implement relevance feedback was originally introduced by Rocchio [8]. In this algorithm, the query is modified selectively using relevant documents to achieve improved retrieval. Ide [20] showed that by incorporating non-relevant documents as well as the relevant ones, retrieval accuracy could be improved even further. In [11] and [21], query modification using Rocchio's formula and query expansion schemes are used, while others rely on support vector machines (SVM) [22–24], on learning from a committee of agents [25], or on boosting algorithms [26].

The learning capabilities of a neural network provides an ideal framework around which an adaptive TRS could be built [27, 28]. Kwok [29] devised a probabilistic document retrieval system implemented using a feedforward neural network. The search results are ranked in the order of conditional probabilities that are estimated based on a sample of relevant documents to the query. In [30], [31], a back-propagation neural network (BPNN) was used as a retrieval system. The retrieved documents for a query are compared against the corresponding relevant document set and if any non-relevant document is also retrieved the network relearns to remove the document from the list. The documents are listed as relevant or non-relevant and are not ordered in accordance with their relevancy to the query. The results in [30], however, indicated poor performance of the network when the training was done only with several relevant documents. Boughanem *et al.* [32] used an unsupervised network with two fully interconnected layers where the neurons in the first layer represent

terms and those in the second layer represent documents. Hebb's learning rule [33] was used to modify the connection weights. Recently, Bouchachia [34] proposed a hierarchical fuzzy neural network architecture for document retrieval. The documents and queries are represented as fuzzy sets and a two-layer neural network is used to learn the implicit relationship among the documents.

In [22], Tong and Koller developed an active learning scheme using SVM, called SVM_{Active}, to quickly and effectively learn the boundary that separates samples that satisfy the user's query concept from the rest of the text database. To solicit relevance feedback, the user is asked to label a small set of documents as relevant or non-relevant classes. Using these initially labelled samples, the system finds the separating hyperplane and performs a series of querying rounds. At each round, the hyperplane parameters are adjusted based upon user votes on the unlabelled samples closest to the hyperplane. At the completion, the SVM_{Active} returns the top k-most relevant samples which are farthest from the hyperplane on the query concept side (i.e. relevant samples). Comparison of the SVM_{Active} results with those obtained using Query by Committee (QC) algorithm [25] indicated the superiority of SVM_{Active} regardless of the initial number of labelled samples.

In [35], using the risk minimization framework of SVM and the description-oriented class of ranking functions [36], a learning method for linear retrieval function using click-through data is presented. Discordance pairs between the ranked documents and the click-through data are used to create a set of training samples. The goal is to learn a ranking function with the minimum number of discordance pairs. This is equivalent to maximizing the Kendall's τ [37] factor, which measures the degree of correspondence between two ranking schemes. A suboptimal solution is suggested by formulating the SVM problem with a penalizing factor that accounts for the errors of the discordance pairs.

1.2.2 Survey on Image Retrieval Systems

During the last decade extensive research on content-based image retrieval (CBIR) has been conducted [38–40] to incorporate user's expertise via relevance feedback. Relevance feedback, which is a well-known [26, 41–43] mechanism created for text retrieval systems, has found acceptance in image retrieval applications as well. CBIR systems utilize relevance feedback to dynamically modify the original submitted image query [44] or some similarity measure [45] in order to meet users' requirements.

Algorithms such as query modification and query point movement for relevance feedback in the image retrieval area have been adopted from their counterparts in the text retrieval area. In query modification, the query is modified selectively using relevant and non-relevant documents to achieve improved retrieval. An example of an image retrieval system that utilizes relevance feedback using this scheme is the Multimedia Analysis and Retrieval System (MARS) [46]. In most retrieval systems, the process of query modification via relevance feedback is typically interactive involving several querying rounds until the refinements eventually result in a list of images that most closely carry the required concept. During this challenging and time consuming process the end-user may loose his/her patience in voting and scoring the relevant images, which may in turn lead to inaccurate or inconsistent association. Moreover, as pointed out before these systems are designed for general-purpose search and retrieval applications and are not suited for scenarios where the domain of applications is limited to specific databases with expert end-users. Also, relevance feedback is applied only on temporarily basis for each user during a particular session. Additionally, most of the present algorithms assume binary (positive or negative cases) relevance feedback from the users.

More recently, user feedback has found numerous applications in machine learning using both positive and negative samples. In this framework, methods such as

Support Vector Machines (SVM) [47], Bayesian inference [48], K-nearest neighbor classification, boosting, and bagging [49] are among the popular ones. The system developed by Chen *et al.* [50] used a learning scheme based upon SVM to classify positive feedback samples from the negative ones given that only the positive samples are known. The assumption is that positive examples cluster in a certain way but negative ones do not, since they can potentially belong to any class. They used a function to map each positive sample from the input space to the feature space, where most of the positive samples are placed inside a hyper-sphere, with unknown center and radius subject to certain constraints. An unknown sample is then classified as positive or negative depending on whether it lies inside or outside the hyper-sphere. They reported excellent results considering that they used a small number of positive training samples. However, the performance of the method relies on the choice of a free parameter. There is also no control on the relevance (or position) of the positive samples as all of them are treated equally.

Another method that modifies a similarity metric was proposed by Guo *et. al* [26]. They applied a boosting algorithm in order to learn the boundary between positive samples and the negative ones. The input to their algorithm consists of the training samples and their corresponding labels. Adaboost iterative algorithm [49] is used starting from a set of weak hypothesis to arrive at the best final hypothesis after several iterations. The final hypothesis is used to form the decision boundary. The results are compared with the boundary region generated by a SVM. Positive and negative samples for boundary learning are provided by the users through the relevance feedback. Although this algorithm is appropriate for some photo collections, its effectiveness remains in doubt when the amount of noise in images increase [51].

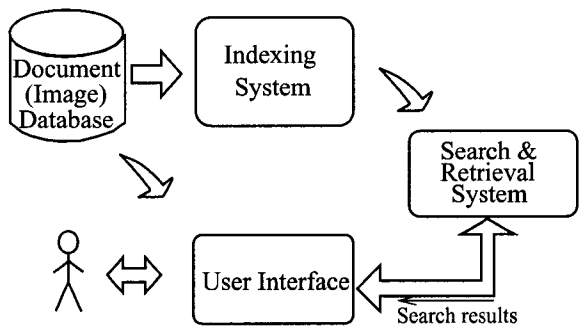
Among the amply variety of methods that incorporate relevance feedback into the learning process for image retrieval [26, 43, 52–54], the approach in [55] is perhaps

the most related one to that presented here. In [55], the authors created a RBF-alike scoring function using a sum of univariate Gaussian-shaped functions centered at individual features of the input query. After the user is asked to label some displayed images as relevant or non-relevant, the standard deviation for each feature is estimated. In addition, the input query suffers a series of modifications in order drive the query towards the relevant images and against the non-relevant images. To this aim, they use the linear vector quantization (LVQ) algorithm. After convergency of the algorithm, the final query is substituted in the RBF-alike scoring function to compute the score of each image in the database. The main advantage of the scoring function is that it is easy to compute. However, the scoring function does not represent a RBF structure but rather an overwhelming simplification where the good properties of RBF approach as function approximation are not fully exploited. In our work, we circumvent this problem by using the kernel functions along with appropriate learning rules to find the corresponding parameters.

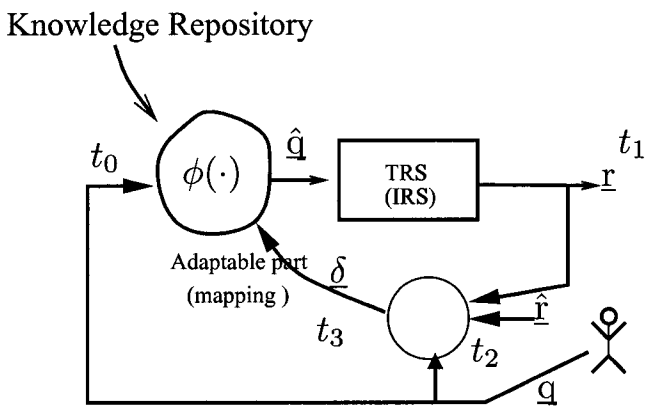
1.2.3 Overview of a Typical Text/Image Retrieval System

A typical TRS/IRS consists of several components or subsystems namely storage, document (image) indexing system, user interface and search/retrieval system (see Figure 1.1 (a)). The storage system maintains the documents (or images) in the database in their original format, which for the case of text could be a simple text format in ASCII, a more sophisticated format such as hyper-text-markup-language 'HTML', or Standard Generalized Markup Language (SGML) and for the case of images, it could be the Joint Photographic Experts Group (JPEG) format, the windows bitmap (bmp) format, or another useful representation. The indexing system processes each document (or image) in the database and generates an indexed file (or feature vector) based upon certain attributes in the documents (images). These attributes represent the importance of different terms (or features) contained in the

document (image). These attributes (features) form a vector $\underline{d}_j = [d_{1j}, d_{2j}, \ldots, d_{Mj}]^T$, that represents j^{th} document, $j \in [1, N]$, where N is the total number of documents (images) in the database and $'T'$ represents the transposition operation. The component, $d_{ij}, i \in [1, M]$ where $M \gg N$ is the total number of terms (number of features) in the entire corpus, gives the weight or importance of the term (feature) t_i in document (image) $\underline{d}_j$. When a specific term is not present in the document the corresponding entry in the vector is 0 for the case of text (for images, regularly all the elements have a value different of zero).



(a) Typical (non-adaptable) TRS (IRS).



(b) Relevance-Feedback Adaptable Retrieval System.

Figure 1.1: Typical and Adaptable Text (Image) Retrieval Systems.

The retrieval and search system performs, upon the user request or query, a similarity measure $s(\underline{q}, \underline{d}_j)$ between the submitted (text or image) query $\underline{q}$ and each document (image) vector $\underline{d}_j$ in the entire database and delivers $n \leq N$ closest matches. In the simplest case, this similarity measure could be $s(\underline{q}, \underline{d}_j) = \underline{d}_j{}^T \underline{q}$. The search and matching processes result in a list of the relevant and non-relevant documents arranged in order of their relevancy (or match) to the submitted query. The "retrieved status value" or relative score of each listed document is clearly a function of the adopted similarity measure and the model used by the particular TRS (IRS) system [56].

If the search results and the retrieved status values are not arranged in their relevancy to the items of interest, the user may need to interactively modify the query until the refinements lead to results that most closely carry the required query concept. Since the identification of an optimum query is difficult and sometimes an impossible task (especially for images), the user must navigate through the list of documents and identify the most relevant document(s), which could be unfavorable for large databases.

The navigation process is illustrated in the upper part of Figure 1.1 (b). For simplification purposes, only one round of querying divided in four consecutive time steps is presented. Let us describe this process in detail for our text retrieval case. At time t_0, query $\underline{q}$ is submitted to the TRS. This query is an M-dimensional binary vector $\underline{q} = [1 \ldots 0 \ 1 \ 0 \ldots 1 \ldots 0 \ 1]^T$, where the i^{th} element is one if the i^{th} term in the database is in the query, otherwise it is zero. At time t_1, the list of scored documents $\underline{r}$ is delivered. Then at t_2, the user, evaluates the listed documents and provides information about some relevant and non-relevant documents in order to form a desired list of scored documents $\underline{\hat{r}}$. At time t_3, an error signal $\underline{\delta}$ is formed that is the difference between the actual output of the retrieval system and the desired output, i.e. $\underline{\delta} = \underline{r} - \underline{\hat{r}}$. This error is then used to adjust the internal parameters of

the adjustable mechanism in order to meet user's requirements.

We define relevance feedback learning as the process of finding a multivariate mapping function $\underline{\phi} : \mathbb{R}^M \rightarrow \mathbb{R}^M$ that transforms an input query $\underline{q}$ into an optimal one, $\hat{\underline{q}}$, given a set of K training samples $\left\{\underline{q}_i, \underline{d}_j, \hat{r}_{ij}\right\}_{i=1,j\in[1,N]}^{K}$, where $\hat{r}_{ij}$ is the desired score of document $\underline{d}_j$ for query $\underline{q}_i$, and N is the number of documents in the collection. Thus, the problem can be cast as follows. Find $\underline{\phi}(\cdot)$ such that

$$f(\underline{\phi}) = \sum_{i=1}^{K}\sum_{j=1}^{N} ||\hat{r}_{ij} - \underline{\phi}^T(\underline{q}_i)\underline{d}_j||^2 \tag{1.1}$$

attains its minimum. Note that this must be solved for a known class of $\underline{\phi}(\cdot)$ functions.

In the case of text retrieval, one of the goals of the proposed system in this work is to incorporate relevance feedback information via a linear mapping function, i.e. $\underline{\phi}(\underline{q}_i) = Z\underline{q}$, where Z is an $M \times M$ matrix. Other two goals are (1) to follow the behavior of a reference model and (2) to make the system robust to environmental changes resulting from removing, adding, or updating documents (images) in the database. In the case of image retrieval, the goal is to incorporate relevance feedback via a non-linear mapping function. The goals of the proposed retrieval systems as well as the approach taken here are discussed in the next section.

1.3 Goals and Contributions of the Present Work

The goal of the present work is to develop, under a unified approach, a theoretical and practical framework related to the creation, maintenance, and learning of adaptable text/image retrieval systems for special-purpose text and image databases with expert end users. The approach is inspired from the well-known model-reference adaptive control theory, and hence our proposed text and image retrieval systems are subsequently referred to as "model-reference text retrieval system (MRTRS)" and "model-reference text retrieval system (MRIRS)". The proposed systems advance the state-of-the-art in adaptable text and image retrieval systems that can continuously

learn from multiple users while maintaining the stability of the previously learnt information.

The propose of our text and image retrieval systems it to capture both the input-output behavior of a "reference retrieval model" and the knowledge from a set of experts users via relevance feedback. The novelty of our approach lies in the rigorous formulation of the problem of learning from users in both text and image retrieval cases where usually only a limited number of training samples are available. The proposed adaptable systems typically contain four components: a plant, which embeds the information of the documents (images) in the database; one or several regulators, that embed relevance feedback repositories; a reference-model, which is represented by a log-file in the image case or an external model-reference as in the case of text retrieval; and an adjustment mechanism, which provides the learning rules to adjust the parameters of the regulator(s).

Three learning modes are devised for the adaptable text retrieval system. These are: initial model-reference learning, model-reference following, and relevance feedback learning. Initial model-reference learning phase is used to capture the input-output mapping of a reference retrieval model for an ensemble set of queries and their corresponding listed documents. This could be done either in a regression mode using a score-based matching or in a classification mode using the support vector machines (SVM)-based framework [33]. The reference model could be either a typical non-adaptable (to user feedback) TRS, or an initial retrieval system. In the latter case, only the document vectors are needed to initially train the system, without the need to have queries and their associated listed documents. In the former case, a typical retrieval model is initially used and then subsequently enhanced using a limited amount of multi-user feedback.

Once the initial model-reference learning is completed, it is crucial that the regulator adapts itself to the changes in the model or the environment (in this case the

database). These changes can be brought about as a result of document re-indexing, adding new documents that may contain new terms, and deleting the obsolete ones. The key requirement is that these changes must be incorporated into the system without impacting the performance or sacrificing the stability of the previously established learning. The model-reference following is accomplished efficiently in the regulator of the proposed system using a recursive (online or batch) learning. This learning mode enables the retrieval system to account for all such changes in a fast and efficient way while keeping all the previously learnt information intact.

Relevance feedback learning mode aims at meeting multiple users' requirements and at the same time preserving the previous learning. In our proposed text (or image) retrieval system, relevance feedback information is incorporated by updating the parameters of the regulator(s). The user relevance feedback information to the adjustment mechanism could be either in form of desired scores of the most relevant document(s) or click-through selection results. These specific scenarios can be implemented using regression or classification-based learning algorithms. When dealing with incorporating relevance feedback from a group of expert users, a classical classification framework is not the most appropriate choice as different documents might receive different votes, even for the same query. In this regard, a regression framework or a modified classification framework which consider partial order relations is more appropriate. Most of the submitted queries to a TRS are short queries containing three or less terms. The proposed methodology for relevance feedback learning accounts for this types of queries as well as for the single-term queries that constitute almost 30% of the total queries that can be encountered.

In summary, the proposed adaptable text and image retrieval systems hold certain unique attributes such as adaptability to environmental changes (database) and their effectiveness in incorporating multiple experts users' feedback under conflicting conditions. Moreover, the ability to incorporate all the new solutions without erasing

the old information is one of the greatest attributes of the proposed retrieval systems.

1.4 Organization of the Thesis

The organization of the thesis is as follows. In Part I of the thesis, Chapter 2 introduces our model-reference text retrieval framework. A general overview of the proposed text retrieval system framed in the model-reference theory is presented. Chapter 3 addresses the problem of setting up the parameters of our system as the problem of learning using a limited set of input-output relationships of an external model retrieval system. Both single-term and multi-term queries cases area considered. The experimental results for the initial model-reference training based on single terms queries and testing on single and multiple terms queries are also presented in this chapter. In Chapter 4, the algorithms for adapting the parameters of the system as well as its structure under the introduction of new documents or the deletion of old ones are presented. Numerous tests are conducted to evaluate the proposed retrieval system when it faces environmental changes resultant from document addition, deletion, or updating. Chapter 5 introduces the problem of learning from expert users via relevance feedback in the proposed adaptable text retrieval system for both single-term and multi-term queries. The problem of relevance feedback learning is cast as regression or a classification problem, depending on the nature of the received relevance feedback. To test the learning algorithms a representative set of queries collected by HP corporation during a six month period is used in the experiments. Chapter 6 introduces a new clustering algorithm that can provide feedback to the users on the relevant query concepts. Comparative results of this algorithm versus the two dimensional self organized memory (2D-SOM) and results from experts users are presented.

In Part II of this thesis, Chapter 7 presents our model-reference image retrieval framework and gives general overviews of the two image retrieval systems proposed in

this work. Chapter 8 covers the detailed description of the image databases and feature extraction methods, namely the shape-dependent features given by the Zernike moments and the textural-dependent features given by Co-Occurrence Matrices, used in the subsequent experiments. Experimental results showing the characteristics of the texture and shape dependent features for some of the 40 different types of objects are presented. In Chapter 9, a new multiple-regulator-based model-reference image retrieval system is presented. The associated learning algorithms for learning from a reference model and users are also developed. Results of the multi-regulator MRIRS are also presented for various modes of learning on an underwater target database. Chapter 10, presents our preliminary results on the development of the single-regulator image retrieval system and some of the learning algorithms. Finally, Chapter 11 presents the conclusions, observations on this work as well as some items to be studied in future work.

PART I: TEXT RETRIEVAL

CHAPTER 2

A MODEL REFERENCE TEXT RETRIEVAL FRAMEWORK

2.1 Introduction

The purpose of this chapter is to present the fundamental ideas behind the creation of an adaptable and robust text search and retrieval engine that can incorporate the users' information and expertise to improve the relevancy of solutions. The proposed system is inspired from the well-known framework of model-reference adaptive control systems [57, 58], and hence is referred to as "model-reference text retrieval system (MRTRS)". The novelty of our approach lies in the development of rigorous formulations for the problem of learning from users in text retrieval systems.

The proposed (MRTRS) system captures users' expertise to enhance its performance based upon a limited number of single-term or multi-term queries. The learning involves three phases that are: (i) initial model-reference learning, (ii) model-reference following, and (iii) relevance feedback learning from expert users. Initial model-reference learning involves capturing the behavior of a reference text retrieval model. The initial model-reference learning uses a set of input-output relationships generated by an external non-adaptable retrieval system to initialize the internal parameters of our MRTRS. In absence of a reference text retrieval model the indexed documents can be used to initialize our system. The model-reference following is

implemented in dynamic situations, when documents are added, deleted or updated without jeopardizing the accuracy of the system for previous solutions. A new relevance feedback learning method will also be developed for single-term and multi-term queries from multiple users using either score-based or click-through selection. Additionally, as a by-product of our system is the ability to cluster queries. This feature allows the system to provide suggestions to the users on how to enhance their original query.

The MRTRS contains five components. These are: a retrieval system or a 'plant', a mapping mechanism or 'regulator', an adjustment mechanism, an external text retrieval system (TRS) or reference model, and a user relevance feedback repository. The plant contains the information of the documents in the text database needed for the retrieval process. In the plant, each document is as a feature vector containing the weights of the terms in the corpus. The plant is in charge of normal search and retrieval operations as in most text retrieval systems. The plant is not adaptable under users' relevance feedback, however, it is adaptable to changes caused by adding or deleting documents. The regulator takes the original query, transforms it using the accumulated feedback, and resubmits it to the plant. When there is no accumulated feedback, provisions are taken to adequately initialize the MRTRS, as we shall show later. The adjustment mechanism modifies the parameters of the regulator based upon the information received from either expert users or another retrieval system (model-reference). The learning is carried out by the adjustment mechanism while what has been learnt is stored in the regulator. The external TRS generates a set of input-output relationships that will be used to set up the regulator. The user feedback repository contains relevance information captured from expert users. This information is then used to enhance the accuracy of the retrieval process not only for solutions previously found but also for new ones that share some common information (generalization capability).

In special-purpose TRS, where mainly expert users conduct search and retrieval operations, it is crucial that the system exactly meets the specific requirements while maintaining the previous learning. The main benefit of the proposed approach is the ability for the expert users to contribute to the decision-making capability of the system and to enhance the relevancy of the suggested solutions, without the slow and expensive process of authoring or modifying the information content within the query directly. The efficiency of the system improves over time as expert users actively provide relevance information in the context of their needs. The learning eventually culminates at the optimal association for mapping queries to documents that will be captured in the system for future use. The MRTRS system offers high retrieval accuracy needed in critical applications and more importantly preserves stability of the stored information, while offering plasticity needed in these applications.

Section 2.2 describes an overview of the adaptable MRTRS along with its components. Section 2.3 presents brief description of the different model-reference learning modes, namely, the initial model-reference, model reference following, and relevance feedback learning covered in more details in the next chapters. Finally, conclusions and observations are given in Section 2.4

2.2 Overview of the Proposed MRTRS

To capture user's expertise via relevance feedback, we propose a model-reference text retrieval system (MRTRS) framework. The overall MRTRS retrieval system architecture is presented in Figure 2.1. The purpose of the system MRTRS is to provide an adaptable text retrieval system that enhances the capabilities of a typical text retrieval system. Under this framework the system might work either in a retrieval mode or in an active learning mode, modifying its internal parameters. The retrieval mode involves computing the score list $\underline{r}$ of the documents following the direct path

between the submitted query $\underline{q}$ and $\underline{r}$, as indicated by the solid lines in the feed-forward path of Figure 2.1, i.e. $\underline{r} = D^T Z \underline{q}$, where $D = [\underline{d}_1 \underline{d}_2 \cdots \underline{d}_N]$ is the matrix of documents of the database $\underline{d}_j = [d_{1j}\ d_{2j} \cdots d_{Mj}]^T$, $j \in [1, N]$, with d_{ij} being the weight of term i in j^{th} document, and Z is an appropriated $M \times M$ mapping matrix that transforms the original query into the 'optimal' one. The learning mode involves capturing either the user expertise via relevance feedback, as indicated in the lower part of the figure, or the behavior of another retrieval system, as indicated by the upper two blocks. In addition, the leaning mode must take into account dynamic databases where documents could be modified, added, or removed.

As can be seen, the proposed MRTRS contains five components. These are: (1) a retrieval system or plant (in analogy to adaptive control); (2) a mapping mechanism (or a regulator) in the form of a mapping matrix; (3) an adjustment mechanism (learning); (4) a reference model; and (5) user's relevance feedback repository, which could be a log file (not shown in Figure 2.1).

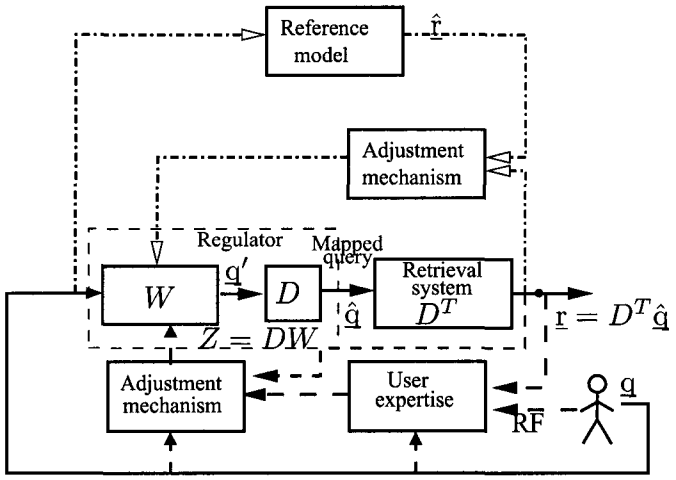


Figure 2.1: Model Reference Text Retrieval System.

In the MRTRS, when the user submits a query, the system first transforms the query into a new or modified version and then resubmits this modified query to the retrieval system to get the score of the documents. The retrieval system could be a TRS as in Figure 1.1 (b). The modified query is generated by the regulator, which

captures the user expertise through the interaction with the user via the relevance feedback process (see Figure 2.1). The initial mapping matrix in the regulator could be an identity matrix, which implies that no expertise is built into the regulator yet. Through time, however, the regulator gains experience from the expert user's feedback (lower path in Figure 2.1) or from the TRS reference model (upper path), if the latter is available. Normally, the relevance feedback information is binary, because users normally click on at the most relevant document(s). When an external reference model or search engine provides the relevant documents for an input query, which happens in our system during the initial-model reference learning mode, the feedback may not be binary. That is, the desired scores of the listed documents (by the model TRS) are used in conjunction with an adjustment mechanism to update the parameters of the regulator. We shall see in the next chapters that if the score for each document is a continuous value then a regression framework is needed to derive the appropriate rules for updating the parameters. On the other hand, if the output of the reference model is given as an ordered list of relevant and non-relevant documents then a classification framework is more appropriate. Thus, the adjustment mechanism associated with the reference model provides the learning rules to update or find (in case that a complete input-output relations were available) the parameters in the regulator. The adjustment mechanism (lower path) associated with the user feedback, can also be implemented either in classification mode (click-through selection) or in regression mode (score-based voting).

2.3 Model Reference Learning Modes

We devise three different learning modes for our MRTRS. These are: (i) initial model-reference learning mode which enables the system to follow accurately a limited number of training samples provided by the reference model; (ii) reference following learning mode which makes the system to adapt itself to the environmental changes brought

about from the addition, deletion or updating of the database documents; and (iii) relevance feedback learning mode which helps the system to incorporate adequately the information from one or more expert users. These modes will be briefly described next.

2.3.1 Initial Model-Reference Learning

The goal of the initial model-reference learning is to capture the input-output mapping of a reference model for an ensemble set of queries and their corresponding listed documents. The reference model could be a typical non-adaptable TRS either with *known* retrieval function, where abundant input-output mapping for a specific set of queries can be generated and then utilized to initially train the system, or a TRS with *unknown* retrieval function, where the input-output mapping can be limited and/or vague. It is vague when, for instance, the scores are not known but instead an ordered list of documents is provided. In absence of a reference model, the results of an indexing system can be used to initially set up the MRTRS. In this case, the parameters in the regulator are obtained exclusively based upon document vectors $\underline{d}_j$'s produced by an indexer (See Remark 3.1 in Chapter 3).

The initial model-reference learning will be implemented in either *regression* or *classification* modes. The regression mode is used when the model retrieval function is known or the output of system is not vague, as defined before. The classification mode is used when the mapping of the model reference system is vague and hence only the class labels (i.e. relevance or non-relevance) are considered. In both regression and classification modes, the formulations can be extended to the kernel case in order to capture non-linearities inherent in the retrieval function. It must be pointed that a linear scoring system with known retrieval function can be easily captured using our framework while the degree on the approximation of a nonlinear scoring function depends in part on the knowledge we have about the type of retrieval function and

on the basis functions we use to approximate it, e.g. it is known [59] that polynomial functions cannot be approximated using Gaussian radial basis functions .

Now, let us consider a reference model that exploits the linear retrieval function proposed by Robertson and Jones [11]. This system uses the classical probabilistic retrieval model. The function for computing a similarity measure between document $\underline{d}$ and query $\underline{q}$ in this model is given by

$$r(\underline{d},\underline{q}) \sim \sum_{l=1}^{M} w_{lq} w_{ld} \left(log \frac{P(t_l|R)}{1-P(t_l|R)} + log \frac{1-P(t_l|\overline{R})}{P(t_l|\overline{R})} \right), \tag{2.1}$$

where w_{lq} and $w_{ld} \in \{0,1\}$ are the weights for term t_l in query $\underline{q}$ and document $\underline{d}$, respectively, and $P(t_l|R)$ and $P(t_l|\overline{R})$ are the estimated probabilities that term t_l is present in a relevant or in a non-relevant document, respectively. If we define $d_l = w_{ld}(log(P(t_l|R)/(1-P(t_l|R))) + log((1-P(t_l|\overline{R}))/P(t_l|\overline{R})))$, $l \in [1, M]$, the document vector becomes $\underline{d} = [d_1 d_2 \dots d_M]^T$, and the query vector is $\underline{q} = [w_{1q} w_{2q} \dots w_{Mq}]^T$. Then, the retrieval function in (2.1) can be expressed as $r(\underline{d},\underline{q}) = \underline{d}^T \underline{q}$. This formulation implies that the plant is represented by matrix D^T in our proposed system (See Figure 2.1).

In the initial model-reference learning, the query mapping subsystem generates a modified query (like control signal in an adaptive control system), $\hat{\underline{q}}$, that yields the desired response or the document list, $\hat{\underline{r}}$, for the original submitted query. The desired response, $\hat{\underline{r}}$, for a submitted query is generated by the reference model. Note that the dimension of the original query space $\underline{q}$ is the same as that of the mapped query $\hat{\underline{q}}$. The process is shown using the dotted-dashed lines in the upper loop of the block diagram in Figure 2.1. The static retrieval system in the feedforward path, which plays a similar role as a plant in an adaptive control system, is a linear system described by matrix D^T where $D = [\underline{d}_1 \underline{d}_2 \dots \underline{d}_j \dots \underline{d}_N]$ is the document matrix generated by an indexing system, $\underline{d}_j$ is the j^{th} document vector of size $M \times 1$.

Initial model-reference learning using the regression method based upon single-term queries is already developed in [2] and successfully tested on the HP product family databases. In this research, we develop linear learning algorithms for both multi-term and single-term query combinations in both regression or classification modes. In Chapter 3, we introduce the ideas and methodologies behind our proposed approaches in more details.

2.3.2 Model Reference Following

Once the initial model-reference learning is completed and the system in the feed-forward path of Figure 2.1 captures the underlying input-output relationship of the reference model, it is crucial that the regulator can adapt itself to the changes in the model or the environment (database). These changes can be brought about as a result of document re-indexing, adding new documents that may contain new terms, and deleting the obsolete ones. The key requirement is that these changes must be incorporated into the system without impacting the performance or sacrificing the stability of the previously established learning.

This model-reference following can be accomplished efficiently in the regulator of the proposed MRTRS using a recursive (online or batch) learning. As in the initial learning, the upper loop in Figure 2.1 together with an appropriate adjustment mechanism is used in this phase. In [2], a batch method for document addition and deletion in linear regression mode and for single-term queries has been developed and tested on the HP product family database. In this research, we have developed on-line iterative learning schemes for both the regression and classification modes when the queries contain multiple terms. We shall explore how to incrementally add and/or delete documents without requiring to retrain the system. The connection of our learning methodology with the classification framework and in particular with the SVM paradigm will be presented in Chapter 3.

2.3.3 Relevance Feedback Learning

Relevance feedback learning refers to the interaction of a retrieval system with a user with the aim of incorporating user's expertise into the retrieval process. Before developing a learning algorithm capable of efficiently incorporating relevance feedback it is indispensable to understand how a typical user-retrieval system iteration takes place. The term 'typical' denotes an interaction between a user and the system in which relevance feedback is not involved. A typical interaction starts when a user submitting a query and the system displaying a list of documents along possibly with their respective scores and a brief description of the document or short passages of text where the terms in the query are highlighted. If the user finds one of these documents of interest then he/she may open it and start searching for the desired concept. In case that the concept is not present the user might decide to open another document or resubmit a modified version of the original query by adding or deleting some query terms. Even in the case that the concept is present in the document the user might still want to conduct another search and look for additional documents that could contain the same concept. The interaction ends satisfactorily when the user finds the concept of interest and decides to terminate the retrieval process.

Often the original submitted query does not meet the specific user requirements in terms of the listed documents, their relevancy and relative scores. To meet the users requirement and at the same time preserve the previous learning, in our proposed MRTRS system, the relevance feedback information can be incorporated using two different learning frameworks, depending on the nature of the user feedback. This can be accomplished by updating the parameters of the regulator or the weights of the block, namely W matrix. In the MRTRS, the lower feedback loop (relevance feedback loop) of the system in Figure 2.1 provides expert users' votes on relevant and non-relevant documents to the adjustment mechanism, which in turn updates the parameters of the regulator to meet the user requirements by imposing relevance

feedback. The user relevance feedback information to the adjustment mechanism could be either in form of desired scores of the most relevant document(s) or click-through selection results. These specific scenarios can be implemented using regression or classification-based learning. Clearly, this phase of learning captures certain user-based information and expertise that cannot be learnt from the reference model alone.

To enhance the behavior of the MRTRS for better capturing the hidden associations between the queries and the relevant documents, relevance feedback learning from multiple expert users can be implemented in our system even when the votes are contradictory. In our proposed system, relevance feedback captures this high-level information from the expert users via the same adaptation mechanism for the regulator parameter. This is done without jeopardizing the previous learning. The multi-user voting process (click-through) results in a log file of queries and the voted documents together with other useful information such as frequency of votes, expertise level of the users, date in which voting takes place, date in which the document is last modified or any other factors. These can then be used in conjunction with some specific heuristic rules to arrive at the desired score vector for every query in the log file. These scores are then provided to the adjustment mechanism for relevance feedback learning.

2.4 Conclusions

This chapter presented a novel adaptable text retrieval system that is based on concepts of adaptive model reference control. The system is capable of incorporating knowledge collected from a pool of expert users via the relevance feedback information or from a reference model via a limited number of training samples. Our proposed model-reference text retrieval (MRTR) system consists primordially of a retrieval system that stores the information of the collection of documents and a regulator that

captures the behavior of a model retrieval system or a group of expert users or the combination. The main function of the regulator is to transform the original query into a new one that meets user requirements in terms of the positions of the relevant documents in the displayed list and/or their retrieval scores. This transformation somehow resembles the addition and/or deletion of terms that a user performs on the original query in order to get the desired list of documents. The learning modes are: (i) initial model-reference learning used to emulate the behavior of an external model text retrieval system; (ii) model reference following used to make the system robust to environmental changes resulting from the introduction of new documents, deletion, and updating of old ones; and (iii) relevance feedback learning used to capture, without erasing previously captured learning, knowledge from a group of expert users. These learning modes provide a fully flexible search engine that can accurately and promptly respond to the needs of the expert users of any other expert agent. A general overview of different possible learning modes were presented, leaving specific details for the subsequent chapters.

CHAPTER 3

INITIAL MODEL REFERENCE LEARNING FOR TEXT RETRIEVAL

3.1 Introduction

As mentioned in the previous chapters, the goal of the initial model-reference learning phase is to capture the input-output mapping of a reference model using an ensemble set of queries and their corresponding listed documents. This could be done either in regression mode using a score-based matching or in classification mode using the SVM framework using only partial information about the listed documents as will be shown later in this chapter. The reference model could be either a typical non-adaptable (to user feedback) TRS, or a simple document indexing system (see Remark 3.1). In the latter case, only the document vectors $\underline{d}_j$'s are needed to initially train the system, without the need to have queries and their associated listed documents.

During this initial model-reference learning the role of the query mapping subsystem is similar to a regulator in an adaptive control system that generates a modified query (control signal), $\hat{\underline{q}}$, that yields the desired response or the document list, $\hat{\underline{r}}$, for the original submitted query. Note that the dimension of the original query space $\underline{q}$ is the same as that of the mapped query $\hat{\underline{q}}$. The process is shown by the dotted-dashed lines in the upper loop of the block diagram in Figure 2.1. The desired response, $\hat{\underline{r}}$, for a submitted query is generated by the reference model. The static retrieval

system, which plays a similar role as a 'plant' in an adaptive control system, is a linear mapping system described by matrix D^T where $D = [\underline{d}_1 \underline{d}_2 \dots \underline{d}_j \dots \underline{d}_N]$ is the document matrix for the entire collection, $\underline{d}_j$ is the j^{th} document vector of size $M \times 1$ as defined before. The document matrix is generated either by the indexing system within the TRS or any other basic indexing process. In the former case, the matrix is generated using the input-output relationships of a selected set of single-term queries, while in the latter case, the matrix is generated using a particular indexing scheme. In both cases, the creation of the 'plant' (represented by the matrix of documents) is a straightforward process that does not require a lot of computations. However, we cannot make the same statement for the creation of the regulator, for which, an appropriated learning scheme for adjusting or updating its parameters must be derived.

The organization of this chapter is as follows. Section 3.2 presents our novel initial-reference learning methodology casting a minimization problem in a regression framework. This is done using either single term or multiple terms queries. Section 3.3 presents the minimization problem in a classification framework, used when only partial information of the listed documents is known. Section 3.4 gives a connectionist system that effectively implements the learning rules derived in previous sections. Section 3.5 presents a study on the computational complexity of the algorithms that generate the mapping matrix of the regulator. Section 3.6 presents the experimental results on a text database of the Hewlett-Packard (HP) corpus. Finally, Section 3.7 presents the conclusions of our initial model reference learning framework.

3.2 Learning as a Regression Problem

3.2.1 Single-Term Queries

The goal of this initial model-reference learning is that having submitted the original i^{th} query $\underline{q}_i$, we would like to find the optimal mapped query, $\underline{\hat{q}}_i$, that yields the desired

response, $\hat{\underline{r}}_i$, where $\underline{q}_i$ is a single-term query consisting of term t_i and represented by input vector $\underline{q}_i = \underline{e}_i$, with $\underline{e}_i = [0 \cdots 1 \cdots 0]^T$ being a unit norm vector with the i^{th} component being 1. Since typically $M \gg N$, this parameter estimation problem is under-determined. Thus, the problem can be cast as a minimum-norm least square (LS) [60] where it is desirable to find a mapped query $\hat{\underline{q}}_i$ with minimum distance from the origin (i.e. small number of terms) subject to constraint $D^T \hat{\underline{q}}_i = \hat{\underline{r}}_i$, where $\hat{\underline{r}}_i = [\hat{r}_{i1}, \hat{r}_{i2}, \ldots, \hat{r}_{iN}]^T$ is the desired score vector for the i^{th} submitted query. Accordingly, we can construct the Lagrangian function

$$J(\hat{\underline{q}}_i, \underline{w}_i) = \frac{1}{2} \hat{\underline{q}}_i^T \hat{\underline{q}}_i + \sum_{j=1}^{N} w_{ij} (\hat{r}_{ij} - \underline{d}_j^T \hat{\underline{q}}_i) \tag{3.1}$$

where $\underline{w}_i = [w_{i1} \cdots w_{iN}]^T$ and w_{ij}'s are Lagrangian multipliers. Differentiating $J(\hat{\underline{q}}_i, \underline{w}_i)$ wrt $\hat{\underline{q}}_i$ and setting the result to zero yields

$$\hat{\underline{q}}_i = \sum_{j=1}^{N} w_{ij} \underline{d}_j = D \underline{w}_i \tag{3.2}$$

Now, taking the derivative of $J(\hat{\underline{q}}_i, \underline{w}_i)$ wrt $\underline{w}_i$ and setting the result to zero yields $D^T \hat{\underline{q}}_i = \hat{\underline{r}}_i$. Combining with (3.2) gives the solution for the optimal $\underline{w}_i$,

$$\underline{w}_i = (D^T D)^{-1} \hat{\underline{r}}_i \tag{3.3}$$

which yields the desired result $\hat{\underline{r}}_i$ at the output of the retrieval system. Thus, the LS solution for $\hat{\underline{q}}_i$ lies in the space spanned by the documents. This can be viewed as a generalization of the Rocchio's formula [11] where all the documents are included and their associated weights are obtained using the learning mechanism in this section.

Note that from Figure 2.1 in Chapter 2, we have the relationship $\hat{\underline{q}}_i = Z \underline{q}_i = DW \underline{q}_i$ between the original query and the mapped query $\hat{\underline{q}}_i$. Clearly, for the single-term queries $\underline{q}_i = \underline{e}_i$, this relationship becomes

$$\hat{\underline{q}}_i = DW \underline{e}_i = D \underline{w}_i, \tag{3.4}$$

where $\underline{w}_i$ is the i^{th} column of weight matrix W. That is, in this case it is only essential to solve for $\underline{w}_i$ to meet the desired score vector $\hat{\underline{r}}_i$. Thus, single-term queries can only identify one column of W at a time.

The objective function $J(\hat{\underline{q}}_i, \underline{w}_i)$ can equivalently be represented in terms of documents $\underline{d}_i$'s and the Lagrangian multipliers leading to the following "dual problem",

$$\xi(\underline{w}_i) = \sum_{j=1}^{N} w_{ij} r_{ij} - \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} w_{ij} w_{ik} \underline{d}_j^T \underline{d}_k \tag{3.5}$$

$$= \underline{w}_i^T \underline{r}_i - \frac{1}{2} \underline{w}_i^T D^T D \underline{w}_i \tag{3.6}$$

which should be maximized wrt $\underline{w}_i$. This cost function is only represented in terms of weights and dot product of documents $\underline{d}_j^T \underline{d}_k$. This implies that there is a close link between the proposed query mapping approach and SVM in the original linear space. This is shown in Section 3.3.

3.2.2 Multiple-Term Queries

The developed algorithms for single-term queries can be extended to multi-term query learning. To see this, let us divide the problem of finding mapping matrix $Z = DW$ (see Figure 2.1) given an ensemble of training samples into the problem of finding certain columns of mapping matrix $Z = [\underline{z}_1 \ \underline{z}_2 \dots \underline{z}_M]$ by using only specific samples that convey the necessary information. For instance, queries that contain one or more terms t_{j_1}, t_{j_2}, and t_{j_3}, $j_1, j_2, j_3 \in [1, M]$, can be used to find columns $\underline{z}_{j_1}$, $\underline{z}_{j_2}$ and $\underline{z}_{j_3}$, respectively.

Now, the problem of finding T columns $\underline{z}_{j_1}$, $\underline{z}_{j_2}$, ... $\underline{z}_{j_T}$ given a set of T-term queries $\underline{q}_i = \sum_{l=1}^{T} q_{ij_l} \underline{e}_{j_l}$, $i \in [1, K]$ with $K \geq T$, with their respective outputs $\underline{r}_i$'s, where $\underline{e}_{j_l}$ is the vector containing 1 at the j_l^{th} position and zero elsewhere, can be cast in an optimization framework. The goal here is to find $\underline{z}_{j_1}, \dots, \underline{z}_{j_T}$ and $\underline{v}_i$'s that minimize the Lagrangian function,

$$J(\underline{z}_{j_1}, \dots, \underline{z}_{j_T}, \underline{v}_1, \dots, \underline{v}_K) = \frac{1}{2} \sum_{i=1}^{K} \hat{\underline{q}}_i^T \hat{\underline{q}}_i + \sum_{i=1}^{K} \underline{v}_i^T (\hat{\underline{r}}_i - D^T \hat{\underline{q}}_i) \tag{3.7}$$

Since we require linear optimal mapping of the form $\hat{\underline{q}}_i = Z\underline{q}_i = \sum_{l=1}^{T} q_{ij_l}\underline{z}_{j_l}; \quad i \in [1, K]$, we set the derivative of J wrt $\underline{z}_{j_l}$ to $\underline{0}$. This yields

$$\sum_{i=1}^{K} \hat{\underline{q}}_i q_{ij_l} - \sum_{i=1}^{K} q_{ij_l} D\underline{v}_i = \underline{0}; \qquad l \in [1, T] \tag{3.8}$$

To transfer the information of the ensemble of optimal queries $\hat{\underline{q}}_i$'s into the mapping matrix, let us plug the i^{th} optimal query $\hat{\underline{q}}_i = \sum_{p=1}^{T} q_{ij_p}\underline{z}_{j_p}$ into (3.8). This gives,

$$\sum_{p=1}^{T}\left(\sum_{i=1}^{K} q_{ij_p} q_{ij_l}\right)\underline{z}_{j_p} = D\sum_{i=1}^{K} q_{ij_l}\underline{v}_i = D\hat{\underline{w}}(l); \qquad l \in [1, T] \tag{3.9}$$

which must hold for the query terms $t_{j_l}, \ldots, t_{j_T}$. Here, $\hat{\underline{w}}(l) = \sum_{i=1}^{K} q_{ij_l}\underline{v}_i; \; l \in [1, T]$, and the term in the bracket in (3.9) represents a correlation measure between the terms in the set of queries. To find a solution for $\underline{z}_{jl}$'s let us define the matrix $H = [h_{pl}]$, $p, l = 1, 2, \ldots, T$ with elements $h_{pl} = \sum_{i=1}^{K} q_{ij_p} q_{ij_l}$ and matrix $\widehat{W} = [\hat{\underline{w}}(1)\; \hat{\underline{w}}(2) \ldots \hat{\underline{w}}(T)]$. Then, (3.9) can be rewritten in matrix form $\widehat{Z}H = D\widehat{W}$, where matrix $\widehat{Z} = [\underline{z}_{j_1}\underline{z}_{j_2} \ldots \underline{z}_{j_T}]$ contains columns $j_1, j_2, \ldots, j_T$ of Z. If we solve for $\widehat{Z}$, we get

$$\widehat{Z} = D\widehat{W}H^{-1} \tag{3.10}$$

As can be seen, to solve for $\widehat{Z}$ we need to compute $\widehat{W}$ and H^{-1}, where H^{-1} (inverse of a $T \times T$ correlation matrix) exists and is easy to compute as T is usually small ($T < K$) and queries with few terms are abundant.

A recursive equation for $\widehat{Z}$ can be found using the corresponding recursive equations for $\widehat{W}$ and H^{-1}. Since $H(k)$, the correlation matrix for query 'k', can be written as a function of the new query sample $\underline{q}_k$ and $H(k-1)$ as $H(k) = H(k-1) + \underline{q}_k\underline{q}_k^T$, $k \in [1, K]$, its inverse can easily be computed using the matrix inversion lemma [49],

$$H^{-1}(k) = H^{-1}(k-1) - \frac{H^{-1}(k-1)\underline{q}_k\underline{q}_k^T H^{-1}(k-1)}{1 + \underline{q}_k^T H^{-1}(k-1)\underline{q}_k} \tag{3.11}$$

Now, since the constrains in (3.7) is given by $\hat{\underline{r}}_i = D^T\hat{\underline{q}}_i \,; i = 1.2 \ldots K$, this implies that (3.8) can be rewritten as $\sum_{i=1}^{K}\left(\hat{\underline{r}}_i - D^T D\underline{v}_i\right) q_{ij_l} = \underline{0}; \; j_l \in [1, T]$. From here, the

Lagrange multipliers $\underline{v}_i = (D^T D)^{-1}\hat{\underline{r}}_i$, $i \in [1, K]$ are found and then used to form the columns of $\widehat{W}(k)$ to obtain $\widehat{W}(k) = (D^T D)^{-1}\sum_{i=1}^{k}\hat{\underline{r}}_i\underline{q}_i^T$. Consequently, the recursive equation for $\widehat{W}(k)$ is

$$\widehat{W}(k) = \widehat{W}(k-1) + (D^T D)^{-1}\hat{\underline{r}}_k\underline{q}_k^T. \tag{3.12}$$

Having found $\widehat{W}(k)$ and $H^{-1}(k)$, $\widehat{Z}(k)$ can be computed using $\widehat{Z}(k) = D\widehat{W}(k)H^{-1}(k)$.

3.3 Learning as a Classification Problem

The initial model reference learning and relevance feedback (Chapter 5) in our framework can also be implemented in classification mode, if desired. To see this, let us compare the dual cost function [24] of the SVM to that in (3.5) or (3.6). If we change $\underline{w}_i \Rightarrow \Lambda_i\underline{w}_i$ where Λ_i is a diagonal matrix with elements 1 (class 1) or -1 (class 2), and further $\Lambda_i\underline{r}_i = \underline{1}$ with $\underline{1}$ being the one vector, then the cost function in (3.6) becomes exactly the same as that of SVM. Note that since $\Lambda_i^2 = I$ then $\underline{r}_i = \Lambda_i\underline{1}$ which implies that the score vector consists of elements 1 (relevant documents) or -1 (non-relevant documents) as in a two-class problem. Now, taking the partial derivative of the modified $\xi(\underline{w}_i)$ wrt $\underline{w}_i$ and setting the result to zero yields,

$$\underline{w}_i = \Lambda_i(D^T D)^{-1}\hat{\underline{r}}_i \tag{3.13}$$

Moreover, from the modified primal problem in (3.1),

$$J(\hat{\underline{q}}_i, \underline{w}_i) = \frac{1}{2}\hat{\underline{q}}_i^T\hat{\underline{q}}_i + \underline{w}_i^T\Lambda_i(\underline{r}_i - D^T\hat{\underline{q}}_i) \tag{3.14}$$

$$= \frac{1}{2}\hat{\underline{q}}_i^T\hat{\underline{q}}_i + \underline{w}_i^T(\underline{1} - \Lambda_i D^T\hat{\underline{q}}_i) \tag{3.15}$$

the solution for the optimal query for this two-class classification problem becomes,

$$\hat{\underline{q}}_i = D\Lambda_i\underline{w}_i \tag{3.16}$$

Clearly, this optimal query yields the desired output of $D^T\hat{\underline{q}}_i = \underline{r}_i$. These results show that the proposed learning can be implemented in either regression mode or

classification mode linked strongly to the SVM framework. This offers the potential for development of kernel-based search machines using this framework.

3.4 Connectionist Network Implementation

The learning capabilities of a neural network provides an ideal framework around which an adaptive TRS could be built [27,28,61,62]. Kwok [29] devised a probabilistic document retrieval system implemented using a feedforward neural network. The search results are ranked in the order of conditional probabilities that are estimated based on a sample of relevant documents to the query. In [30], [31], a back-propagation neural network (BPNN) was used as a retrieval system. The retrieved documents for a query are compared against the corresponding relevant document set and if any non-relevant document is also retrieved the network relearns to remove the document from the list. The documents are listed as relevant or non-relevant and are not ordered in accordance with their relevancy to the query. The results in [30], however, indicated poor performance of the network when the training was done only with several relevant documents.

Boughanem *et al.* [32] used an unsupervised network with two fully interconnected layers where the neurons in the first layer represent terms and those in the second layer represent documents. Hebb's learning rule [33] was used to modify the connection weights. Recently, Bouchachia [34] proposed a hierarchical fuzzy neural network architecture for document retrieval. The documents and queries are represented as fuzzy sets and a two-layer neural network is used to learn the implicit relationship among the documents.

The query mapping and retrieval processes in the feed-forward path of our MRTRS system in Figure 2.1 can be implemented using a simple three-layer network as illustrated in Figure 3.1. Each node, j, in the first and third layers represents a document $\underline{d}_j$, $j \in [1, N]$; whereas each node i in the second layer represents a term t_i, $i \in [1, M]$.

The connection weight from node j in the first layer to node i in the second layer is d_{ij}. Similarly, the connection weight between node j in the third layer and node i in the second layer is the same weight d_{ij}. Thus, the weight matrices for the second and third layers are D and D^T, respectively. These weights remain unchanged unless the documents are re-indexed or updated. The weight matrix for the first (flexible) layer, however, is $W = [\underline{w}_1 \underline{w}_2 \cdots \underline{w}_M]$. As shown before, in the initial model-reference learning the objective is to find W to capture input/output behavior of the reference system for every query in the ensemble set. The first and second layers combined form the mapped query at the term layer, i.e. $\underline{\hat{q}}_i = DW\underline{q}_i$, which in turn yields the retrieved documents and their desired score vector, $\underline{\hat{r}}_i = D^T DW\underline{q}_i$ at the output of the retrieval layer for optimal W. Consequently, the first and second layers combined function like the regulator in Figure 2.1. Note that the space that spans the original input query is the same as the mapped term space (second layer) that forms the optimal mapped query.

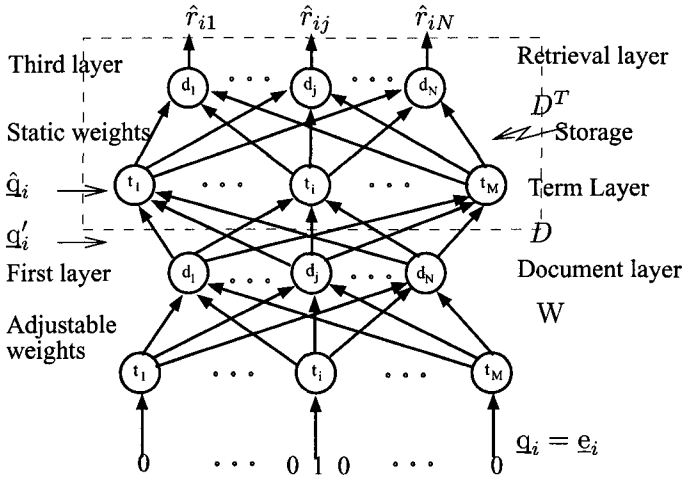


Figure 3.1: Proposed flexible network structure.

In this network, the inputs are indexed representing different possible terms in the submitted query. Each input can take either 0 or 1 values depending on the absence or presence of the corresponding term in the query. As mentioned before, if a

single-term query is applied to the network, the output of the first layer extracts the i^{th} column of weight matrix W, i.e. $W\underline{e}_i = \underline{w}_i$, which in turn generates the mapped query $\hat{\underline{q}}_i = D\underline{w}_i$ at the output of the second layer. Thus, $\underline{w}_i$ is the weight vector of Lagrangian parameters that connects the term t_i to all the document layer nodes. This implies that learning for each single-term query can be performed independently by only updating $\underline{w}_i$ to meet the desired scores at the output layer. This interesting feature of this network guarantees the stability of the weights for other queries while offering flexibility that is needed to accommodate new model-based or user-based information.

For the initial training phase, (3.3) can be rewritten as,

$$\underline{w}_i^{(0)} = {A_0}^{-1}\hat{\underline{r}}_i^{(0)} \tag{3.17}$$

where $\underline{w}_i^{(0)}$ is the initial weight vector, $\hat{\underline{r}}_i^{(0)}$ is the initial document score vector provided either by the TRS or the indexing system (see Remark 3.1) and $A_0 = D^T D$ is a symmetric positive definite (PD) Gram matrix with element $[A_0]_{i,j} = \underline{d}_i^T \underline{d}_j$. The superscript '0' is used to represent the initial training phase. Equation (3.17) is solved once for all the queries in the ensemble set. The Gram matrix $A_0 \in \mathbb{R}^{N\times N}$ can be expressed as $A_0 = G_0 G_0^T$ where $G_0 \in \mathbb{R}^{N\times N}$ is a lower triangular matrix with positive diagonal entries. Fast algorithms using Cholesky decomposition and triangular matrix inversion [63] can be applied to solve for the weight vector $\underline{w}_i^{(0)}$ for each query i.e., $\underline{w}_i^{(0)} = G_0^{-T} G_0^{-1} \hat{\underline{r}}_i^{(0)}$.

Once the initial model reference learning is completed the weights of the first layer can be updated in response to the relevance feedback from expert users. Relevance feedback learning will only impact those weight vectors corresponding to the terms in the submitted query. This process will be discussed in Chapter 5.

Remarks

3.1 From the definition of W weight matrix and the result in (3.3), it can easily be shown that $W = (D^T D)^{-1} R$, where $R = [\underline{r}_1, \cdots \underline{r}_M]$ is the score matrix for all

the queries in the corpus. It is interesting to note that when a reference TRS is not present, the results of an indexing system can be directly used to initially train the network. In this case, we get $W = (D^T D)^{-1} D^T$ which yields the regulator mapping matrix of $P_D = D(D^T D)^{-1} D^T$, i.e. the projection matrix associated with document space D. This implies that the regulator projects the original query onto a space spanned by the documents to generate the mapped query. Clearly, for the i^{th} query, $\underline{q}_i = \underline{e}_i$, this gives the retrieved score vector $\underline{r}_i = [d_{i1}, \cdots d_{iN}]^T$ that contains the attributes or the weights for the i^{th} term in all documents. Thus, the document that has the highest weight attribute for term t_i will have the highest retrieved score. Although, these scores are not directly representative of relevancy, subsequent learning based upon numerous user-based votes (in general-purpose TRS) for every query will gradually improve the relevancy scores of the documents. This will be shown in Section 3.6.

3.2 If two document vectors are identical, the Gram matrix A_0 becomes singular. This situation can be avoided, if for each document its ID, which is unique to each document, is also added as a representative term. If α is the weight assigned to the term that represents the document name, and $\underline{d}'_i$ is the document vector after augmenting the original document vector $\underline{d}_i$ with the document name, then we have

$$\underline{d}'^T_i \underline{d}'_j = \underline{d}^t_i \underline{d}_j \qquad i \neq j \text{ and } \forall\, i, j \in [1, N] \tag{3.18}$$

$$\underline{d}'^T_i \underline{d}'_i = \underline{d}^t_i \underline{d}_i + \alpha^2 \qquad i \in [1, N] \tag{3.19}$$

Therefore, the new Gram matrix is $A'_0 = A_0 + \alpha^2 I$, which is regularized version of A_0. In the sequel, it is assumed that the Gram matrix is regularized when needed, and for simplicity the notation superscript " $'$ " is dropped.

3.5 Computational Complexity

The computational complexity of an algorithm is commonly measured by the number of elementary operations such as additions/subtractions, multiplications/divisions, and other operations that have a direct impact on the computational time. Since multiplications and divisions are generally more costly, in terms of computational time, than the additions and subtractions, they are primarily the measure of the complexity of an algorithm. In this section, we present computational complexity analysis of the two learning algorithms for setting up the parameters of our MRTRS, the first uses a regression approach and the second one uses a classification approach.

3.5.1 Regression Mode

In the regression learning mode, we are interested in knowing the number of operations needed to compute each column $\underline{w}_i$, $i \in [1, M]$, of matrix W using the weight learning rule given in (3.3). Finding $\underline{w}_i$ involves computing one matrix transposition, one matrix-matrix multiplication, one matrix-vector multiplication, and one matrix inversion. When $N > M$ the bulk of operations is concentrated on the inverse of the Gram matrix $A_0 = D^T D$, which, in general[1], requires roughly $N^3/2$ multiplications, for $N \gg 1$. However, if we decompose the Gram matrix using the Cholesky factorization $A_0 = G_0 G_0^T$ with G_0 being lower triangular the number of operations for the computations reduces to $N^3/3$. Thus, the number of operations is approximately reduced by 23% with the aid of Cholesky factorization. Note that this inverse operation needs to be carried out only once for the given document database.

Now, the computational complexity for setting up the whole matrix W is obtained using the ensemble of M different single-term queries, where each one, most often, produces a score vector $\underline{r}_i$ with many zeros. Thus, the computational complexity for

[1]Without using some properties of the matrix that could help us reduce the number of computations.

computing weight matrix W for a repository of N documents and a vocabulary of M terms is given by

$$C(N, M) \approx \frac{N^3}{3} + N^2 ME, \tag{3.20}$$

where E is the average number of non-zero elements in the vectors of scored documents when single-term queries are submitted. Note that, the second term in (3.20) is obtained as follows. The matrix-vector operation $G_0^{-1}\hat{\underline{r}}_i$ requires on average N^2E multiplications. Now, if we take into account that the training must be done for all M single-term queries then the second term gives the number of multiplications for these queries. Because typically $M \gg N$ and $E > 1$, the second term in (3.20) dominates, i.e.

$$C(N, M) \approx N^2 ME. \tag{3.21}$$

3.5.2 Classification Learning Mode

Following a similar approach as in the regression case, the computational complexity for setting up matrix W in classification learning mode is expressed as

$$C(N, M) \approx \frac{N^3}{3} + N^2 + N^2 ME, \tag{3.22}$$

where the term N^2 is the number of multiplications needed to obtain $\Lambda_i G_0^{-1}$. The number of multiplications needed to obtain $[\Lambda_i G_0^{-1}]\hat{\underline{r}}_i$ is N^2E, if we assume that $\Lambda_i G_0^{-1}$ has already been computed and that $\hat{\underline{r}}_i$ has on average E non-zero elements. Thus, the third term is the number of multiplication for all M single-term queries. Again, the term N^2ME in (3.22) dominates and thus the computational complexity is roughly

$$C(N, M) \approx N^2 ME. \tag{3.23}$$

Note that if we assume that the number of different terms is greater that the size of the document collection ($M > N$) and that $E > 1$ then the number of multiplications is the same for both the regression and classification learning modes.

To put into perspective the computational complexity of these algorithms let us consider two databases: the Hewlett-Packard (HP) text database, subject of our study, and the text retrieval conference (TREC) database [64], used as a testbed in the text retrieval area. The computational complexity of either the regression or the classification learning algorithm using one of the HP-collections that contains 5,000 documents, 30,000 terms, and E around 20, is of order of 1.5×10^{12} multiplications. Thus, the learning algorithm for this collection could take around 4 hours on a machine that executes 100 millions of multiplications per second. Now for one of the TREC collections containing 60,000 documents, assuming 100,000 terms, and $E = 20$, the computational complexity is of 7.2×10^{15} multiplications. Thus, for TREC collection the initial reference learning model could take 833 days on the same machine. This implies that the proposed learning schemes are meant primarily for small to medium size special-purpose text retrieval problems with expert end users. Alternatively, large collections could be partitioned into smaller ones, depending on the content, that can be more efficiently handled by our MRTRS.

3.6 Experimental Results

3.6.1 HP-Text Database

The entire knowledge base for the TRS consists of several major collections of about 60K documents and about $130K$ distinct terms. These collections provide a wide range of information for support, diagnostics, and specifications for consumers and commercial suites of HP products. The documents contain both unstructured and structured information on various product types. The majority of content is represented in text format, while the collections also contain a mixed graphical and multimedia formats. The major collections are based on product types. The results presented in this work are obtained based upon seventeen product collections containing over 32K documents and about 108K distinct terms.

The document survey feature available within the TRS helps users to log their votes on one or more documents for the query submitted. The voting process is influenced by the users' perception on the desired solution documents. The influence comes in the form of a vote, either a positive or a negative one, that the user assigns to a particular document. In order to incorporate user expertise level and add some dynamics into the retrieval system, the vote is weighted by a factor that depends on the users' expertise level and the elapsed time since the document was created or modified. Using the log file of queries along with their respective feedback information, we recreate the voting process used by the users to generate a set of prototype pairs $(\underline{q}, \underline{\hat{r}})$'s with the goal of training our MRTRS. Then, the relevance feedback learning can be applied either iteratively or in the batch mode. It is important to mention that although a large number of queries were available, only a subset of approximately 5900 most commonly used queries are used to form the prototypes. The prototype query set consists of 2386 1-term, 1664 2-term, and 1846 3^+-term queries.

3.6.2 Performance Measures

To assess the performance of the learning algorithms, the rank order correlation measure based on Kendall's τ [37] is used. This nonparametric measure is useful when the underlying distribution that generates the scores cannot easily be estimated. The performance measure based on Kendall's τ compares two ranked or unranked lists and generates a coefficient ($\tau \in [-1, 1]$) that represents the closeness of the two lists. The coefficient τ [37] is given by

$$\tau = \frac{P - Q}{\frac{1}{2}N(N-1)} \tag{3.24}$$

where P and Q are the number of concordant and discordant pairs, respectively. The denominator in (3.24) is the number of combinations of taking two elements out of N and is equal to the sum of the concordant pairs P and discordant pairs Q found in the two lists when they do not contain ties. However, when there are ties, the

measure in (3.24) is not accurate. In this case, a modification can be made to yield Kendall's τ_b as defined by

$$\tau_b = \frac{P - Q}{\sqrt{(\frac{1}{2}N(N-1) - T_1)(\frac{1}{2}N(N-1) - T_2)}} \tag{3.25}$$

where T_1 and T_2 are the number of tied pairs for lists 1 and 2, respectively. Clearly, (3.25) reduces to (3.24) when there are no ties.

Another performance measure used in this study is the standard 'recall' [11] defined as the ratio of the number of relevant documents retrieved to the total number of relevant documents.

3.6.3 Initial Model Reference Learning Results

As pointed out before, the goal of the initial model-reference learning is to capture the input/output behavior of a reference TRS system or to use the results of the indexing system in absence of a model TRS. Thus, three scenarios are considered here: the first and second use the response of the TRS to all the single-term queries in the log file to set up the weights of the network (initial training) through the regression or classification learning modes; while the third one uses the projection matrix P_D, i.e. the indexing results. The regression mode is useful when the scores of the documents are available from the TRS, while the classification mode is useful when a set of relevant document(s) for each submitted query in the training set is known. However, the initial training based upon the indexer does not rely on any querying results.

The networks are trained in the batch mode and the weights of the networks are obtained using (3.3), (3.13) and the projection matrix P_D for the regression, classification, and indexer-based learning modes, respectively. The Kendall's τ_b measure is then generated based on the top twenty documents listed by the initially trained networks evaluated against the 'benchmark', i.e. the results of the TRS augmented by the users' votes. The purpose is to determine how close each initially trained

system can get to the ultimate benchmark. The results are obtained for the most commonly used 1-term, 2-term and 3^{+}-term queries in the log file. Table 3.1 gives the values of τ_b for these most commonly used prototype queries. In the regression and classification learning modes the value of τ_b gets close to 1 (perfect match) for all the single and multi-term queries.

Table 3.1: Performance measure, τ_b, for different learning modes.

Learning Modes	1-Term	2-Terms	3+ Terms
Regression mode	0.935	0.892	0.935
Classification mode	0.927	0.880	0.918
Indexer mode	0.613	0.033	-0.123

The results obtained based upon regression mode learning are slightly better than those of the classification mode learning. This is due to the fact that unlike the regression mode, in the classification mode document scores are ignored and only their binary relevance information is considered. However, the high value of τ_b indicates that the learning through both the regression and classification modes is able to capture the reference model behavior very closely. The results obtained based upon the initial learning from the document indexer indicate that though for single-term queries the value of τ_b is reasonable, the corresponding values for the multi-term queries become unacceptable. This is due to the fact that in this case the querying information and the response of the reference model are not used in the initial training. Nonetheless, we shall show in Chapter 5 that even with this coarse initial training, the proposed relevance feedback learning leads to excellent final results.

Next, the recall capability of the TRS initially trained using the three different initial learning modes is evaluated. Figures 3.2 (a), (b) and (c) show the recall plots for various thresholds of the retrieved list for the regression, classification, and indexer-based learning modes, respectively and for the most commonly used 1-term,

2-term and 3^{+}-term queries in the log file. As observed from these figures, the results obtained based on the regression mode are significantly better than those of the classification and indexer learning modes. This is due to the fact that in the classification mode learning, for a given query, the binary relevance information of only top twenty documents are considered without considering the scores of the relevant documents. The results of the indexer-mode, which requires the least amount of prior information, are better than those of the classification mode. Thus, in Chapter 5 this initially trained system is used as the starting point for the subsequent learning via relevance feedback.

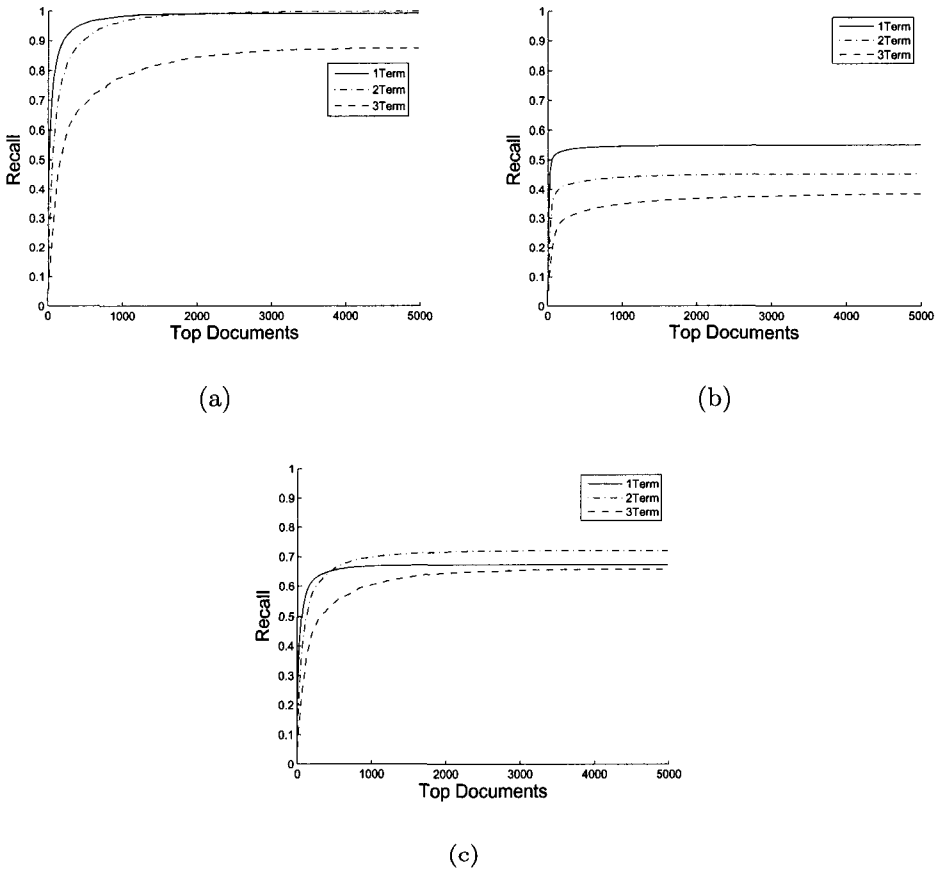


Figure 3.2: Recall plots for learning in (a) regression mode (b) classification mode and (c) indexer mode.

3.7 Conclusions

Two learning algorithms that implement initial-model reference learning for our new adaptive MRTRS are proposed in this chapter. The first algorithm is obtained from the minimization of a cost function cast as a regression problem, while the second algorithm is obtained from the minimization of a cost function cast as a classification problem. The choice of learning algorithm used depends on the type of information provided by a model-reference model, which could be the scores of certain documents in a list of relevant documents. In absence of a model reference TRS another mode of initialization was proposed that only uses information available from the indexed documents in the database. Clearly, the first two modes aim at capturing the behavior of the reference model while the "indexer mode" aims at setting up the parameters of our MRTRS in the absence of a model-reference. The learning is assimilated in a mapping matrix of a simple three-layer connectionist network structure.

The excellent results on both the Kendall's τ correlation measure used to compare the ranked list of our retrieval system against that of a model retrieval system, and the 'recall' measure show the effectiveness of our learning algorithms to initially set up the MRTRS based on a limited number of training samples corresponding to single-term queries. Among the three learning modes, namely regression, classification, and indexer presented in this chapter, the results of the first two show that in fact it is possible to learn and initially set up the system using a limited number of samples; situations that commonly arise when learning from typical search and retrieval engines. The advantage of the indexer-based learning mode is that it does not rely on an external retrieval system and hence does not require document scores/labels. Although the results on the indexer initial learning mode indicated poorer performance when compared to the regression and classification modes, the performance significantly improves as a result of the incorporation of user relevance feedback. This will be shown in Chapter 5.

CHAPTER 4

MODEL REFERENCE FOLLOWING FOR TEXT RETRIEVAL

4.1 Introduction

In practice, most text retrieval systems operate under dynamic environments where the arrival of new documents and the deletion or updating of old ones are events that occur frequently and typically asynchronously. The majority of the systems incorporate these changes of the databases by re-indexing [65–69] the set of documents. It is fundamental that our retrieval system can easily incorporate these changes without impacting the performance or sacrificing the stability of the previously established learning. In order of not interfering with the running retrieval system, the changes can be made in batch mode using a replica of the information of the retrieval system that includes the parameters of the regulator and the documents of the plant.

We define model reference following in the MRTRS as the methodology that makes the regulator or the plant to respond adequately to environmental changes caused by frequent database operations such as addition, deletion, and updating of documents. Since the changes are related to the reference model (rather than the expert users), after the model reference following is carried out, the outputs of the retrieval system for a selected set of training queries should be close enough to those of the reference model. Given that the list of queries could be large it is very important to ensure

that the retrieval system follow exactly the reference-model for all the queries.

In our MRTRS, the model-reference following can be accomplished efficiently in the regulator of the system using a recursive (online or batch) learning. As in the initial learning, the upper loop in Figure 2.1 together with an appropriate adjustment mechanism is used in this phase. In the proposed three-layer network these changes can easily be implemented via structural as well as weight adaptation mechanisms. Structural adaptation involves node addition and deletion. The following sections describe the adaptation rules for these scenarios for the case where the learning is carried out in regression mode. Note that the results in Chapter 3.1 can be used to derive similar adaptation rules for the classification mode of model-reference following.

The organization of this chapter is as follows. Section 4.2 develops the document addition learning using basically two modes, namely an iterative online mode for the case when documents are added one by one, or in a batch mode where documents are first collected and then added to the database. Section 4.3 develops the document deletion learning, covering again the online and the batch modes. Section 4.4 presets the document updating learning for documents that undergo modifications. Section 4.5 presents the results of the proposed model-reference following. Finally, Section 4.6 presents the conclusion of this chapter.

4.2 Document Addition

To incorporate L new documents into the three-layer network in Figure 4.1, additional nodes with new connection weights must be added while the old weights of the network must be updated. Let $\underline{d}_{(N+1)}$, $\underline{d}_{(N+2)}$, and $\underline{d}_{(N+L)}$ be the new document vectors to be added into the system, and m be the total number of new terms introduced as a result of adding these documents. To accommodate these documents, L new nodes must be inserted into the first and the third (output) layers. The weight vectors corresponding to connections emanating from the added nodes in the first hidden

layer and the weight vectors for the incoming connections to the added nodes in the third layer are $\underline{d}_{N+l}, \forall l \in [1, L]$. Additionally, to account for the addition of m newly introduced terms, m new nodes must be added in the second hidden layer as well as in the input layer. The network has to be updated in such a way that even after the insertion of the new documents with the corresponding new terms, the system retains the original training. Figure 4.1 shows the modified version of network in Figure 3.1 after the insertion of the new documents and terms. The newly added nodes are shown inside the dashed boxes. For all the already existing terms i.e., $t_i, i \in [1, M]$, the old connections weights $w_{ij}, j \in [1, N]$ have to be updated; while the new connection weights $w_{i(N+l)}, \forall l \in [1, L]$ have to be found. For those newly introduced terms $t_i, i \in [M+1, M+m]$ all the connection weights must be computed.

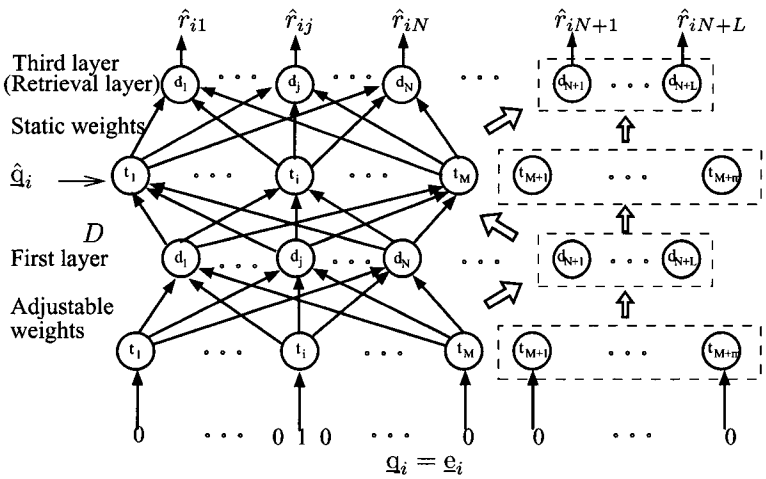


Figure 4.1: Network after the insertion of L new documents.

Assume that the system is initially trained for N documents and that data matrix D contains these documents. Let D_L be the data matrix of L newly introduced document vectors. These new documents are added to the column space of matrix D. Now, using (3.2) in Chapter 3 the optimal query for the single-term query, t_i, in this augmented space is given by,

$$\underline{\hat{q}}_i^{(1)} = \sum_{j=1}^{N+L} w_{ij}^{(1)} \underline{d}_j \tag{4.1}$$

The desired score vector, $\hat{\underline{r}}_{iL} = [\hat{r}^{(1)}_{i\,N+1} \ldots \hat{r}^{(1)}_{i\,N+L}]^T$, of the L newly added documents is assumed to be available from the reference model. Since the score of the already existing documents should not change, the score vector of all the documents for the mapped query in (4.1) is $\hat{\underline{r}}_i^{(1)} = [\hat{\underline{r}}_i^{(0)^T} \hat{\underline{r}}_{iL}^T]^T$. We redefine $\underline{w}_i^{(1)} = [w_{i1}^{(1)}, w_{i2}^{(1)}, \ldots w_{i\,N+L}^{(1)}]^T$ as the new weight vector after inserting these L new documents. Clearly, we still have $\underline{w}_i^{(1)} = A_1^{-1} \hat{\underline{r}}_i^{(1)}$ where matrix A_1 is given as,

$$A_1 = \begin{pmatrix} A_0 & B \\ B^T & C \end{pmatrix}. \tag{4.2}$$

Here A_0 is the same as before, $B = D^T D_L$ is a matrix of dot products between old and new documents, and $C = D_L^T D_L$ is a matrix formed with the dot products of the new documents only. If the inverse of matrix A_1 is expressed as,

$$A_1{}^{-1} = \begin{pmatrix} F & V \\ V^T & U \end{pmatrix} \tag{4.3}$$

then

$$\begin{aligned} U &= (C - B^T A_0^{-1} B)^{-1} \\ V &= -A_0^{-1} B U \\ F &= A_0^{-1}(I - B V^T) \end{aligned} \tag{4.4}$$

Using the expansion for matrix $A_1{}^{-1}$, the updating equation for the weight vector $\underline{w}_i^{(1)}$ can be given as,

$$\underline{w}_i^{(1)} = \begin{pmatrix} \underline{w}_i^{(0)} + V(\hat{\underline{r}}_{iL} - B^T \underline{w}_i^{(0)}) \\ U(\hat{\underline{r}}_{iL} - B^T \underline{w}_i^{(0)}) \end{pmatrix} \tag{4.5}$$

This weight update equation is performed for all nodes $i \in [1, M]$ or single-term queries in the input layer. The first and second parts of the vector in the right hand side of (4.5) correspond to updating the old weights and computing the new

added weights, respectively. For those newly introduced terms, since $\underline{w}_i^{(0)} = \underline{0}$, $\forall\ i \in [M+1, M+m]$, we have

$$\underline{w}_i^{(1)} = \begin{pmatrix} V\underline{\hat{r}}_{iL} \\ U\underline{\hat{r}}_{iL} \end{pmatrix} \tag{4.6}$$

From (4.5) and (4.6), it can be seen that only U and V are needed to update the weights. Computing U involves inverting a matrix of dimension L where $L \ll N$.

Remark

This batch learning reduces to an iterative on-line learning if every time a new document is added the weight updating is implemented. This leads to the following recursive equations

$$\underline{w}_i^{(1)} = \begin{pmatrix} \underline{w}_i^{(0)} - \beta^{-2} A_0^{-1} \underline{b}(\hat{r}_{i(N+1)} - \underline{b}^t \underline{w}_i^{(0)}) \\ \beta^{-2}(\hat{r}_{i(N+1)} - \underline{b}^t \underline{w}_i^{(0)}) \end{pmatrix}, \quad \forall\ i \in [1, M] \tag{4.7}$$

and

$$\underline{w}_i^{(1)} = \begin{pmatrix} -\beta^{-2} A_0^{-1} \underline{b} \hat{r}_{i(N+1)} \\ \beta^{-2} \hat{r}_{i(N+1)} \end{pmatrix}, \qquad \forall\ i \in [M+1, M+m] \tag{4.8}$$

where A_1 matrix in this case is given by,

$$A_1 = \begin{pmatrix} A_0 & \underline{b} \\ \underline{b}^T & c \end{pmatrix}, \tag{4.9}$$

matrix A_0 is defined before, $\underline{b} = D^T \underline{d}_{(N+1)}$, and $c = \underline{d}_{(N+1)}^T \underline{d}_{(N+1)}$. Also, it can easily be shown that

$$\beta^2 = \underline{d}_{(N+1)}^T P_D^{\perp} \underline{d}_{(N+1)} \tag{4.10}$$

where $P_D^{\perp} = I - P_D$ and the projection matrix P_D was defined before.

4.3 Document Deletion

When multiple (e.g. L) documents are to be simultaneously removed, they could be at any position within the network structure. To facilitate the deletion of these

documents (or nodes) it is desirable to move them to the last L columns of matrix D. Let A_0' be the Gram matrix after moving the document vectors to be deleted to the far right side of the data matrix D. Thus, A_0' can be defined in terms of the original Gram matrix A_0 via permutations i.e. we have, $A_0' = (DP_1P_2\cdots P_L)^T(DP_1P_2\cdots P_L) = (P_1P_2\cdots P_L)^T A_0(P_1P_2\cdots P_L)$ where P_i's, $i \in [1, L]$ are the appropriate permutation matrices. Each permutation matrix P_i is formed such that it moves a document vector to be deleted to the far right side of the data matrix so that it can easily be removed. Additionally, we partition the weight and score vectors such that the connection weights corresponding to the documents to be deleted are at the lower end hence we have $\underline{w}_i^{\prime(0)} = [\underline{w}_{i(N-L)}^{\prime(0)^T} \;\; \underline{w}_{iL}^{\prime(0)^T}]^T$ and similarly for the score vector, $\hat{\underline{r}}_i' = [\hat{\underline{r}}_{i(N-L)}^T \;\; \hat{\underline{r}}_{iL}^T]^T$. Thus, the weight vector solution can be written as,

$$\underline{w}_i^{\prime(0)} = A_0'^{-1}\hat{\underline{r}}_i' \tag{4.11}$$

Rewriting A_0' in the block matrix form we have,

$$A_0' = \begin{pmatrix} A_1 & B' \\ B'^T & C' \end{pmatrix} \tag{4.12}$$

The inverse of matrix A_0' can be computed using $A_0'^{-1} = P^T A_0^{-1} P$. Now, rewriting $A_0'^{-1}$ in the block matrix form,

$$A_0'^{-1} = \begin{pmatrix} F_0 & V_0 \\ V_0^T & U_0 \end{pmatrix} \tag{4.13}$$

and using (4.11) gives,

$$\begin{pmatrix} \underline{w}_{i(N-L)}^{\prime(0)} \\ \underline{w}_{iL}^{\prime(0)} \end{pmatrix} = \begin{pmatrix} F_0\hat{\underline{r}}_{i(N-L)} + V_0\hat{\underline{r}}_{iL} \\ V_0^T\hat{\underline{r}}_{i(N-L)} + U_0\hat{\underline{r}}_{iL} \end{pmatrix} \tag{4.14}$$

where F_0, U_0 and V_0 are defined like before. Now, the solution for the connection weights of the network after deleting L documents can be written as,

$$\underline{w}_{i(N-L)}^{(1)} = A_1^{-1}\hat{\underline{r}}_{i(N-L)} \tag{4.15}$$

Note that (4.15) is obtained by deleting the last L columns and L rows of matrix A_0' in (4.12) and then solving for $\underline{w}_{i(N-L)}^{(1)}$. From (4.12) and (4.13), we can express $A_1^{-1} = F_0(I - B'V_0^T)^{-1}$. Using the matrix inversion lemma [63, 70] we have,

$$A_1^{-1} = F_0(I + B'(I - V_0^T B')^{-1}V_0^T) = F_0 - V_0 U_0^{-1} V_0^T \tag{4.16}$$

Substituting for A_1^{-1} in (4.15) yields,

$$\underline{w}_{i(N-L)}^{(1)} = F_0 \hat{\underline{r}}_{i(N-L)} - V_0 U_0^{-1} V_0^T \hat{\underline{r}}_{i(N-L)} \tag{4.17}$$

From (4.14), we get the following updating equation for document deletion

$$\underline{w}_{i(N-L)}^{(1)} = \underline{w}_{i(N-L)}^{\prime(0)} - V_0 U_0^{-1} \underline{w}_{iL}^{\prime(0)} \tag{4.18}$$

where weight vector $\underline{w}_{i(N-L)}^{(1)}$ after the deletion is expressed in terms of the weight vectors $\underline{w}_{i(N-L)}^{\prime(0)}$ and $\underline{w}_{iL}^{\prime(0)}$ before the deletion. As can be observed, the inverse of matrix U_0 (dimension $L \ll N$) needs to be computed.

4.4 Document Updating

When a document is re-indexed or updated care must be taken to include it into the system because of possibly new document terms. Although the weights in the second and third layers can easily be updated, additional steps should be carried out to modify the first layer connections in order to retain the previously stored information. Suppose that document $\underline{d}_j$ is updated and also new terms are introduced. Let T^0 be the set of remaining terms in the system after the document to be updated is removed from the system. The process of finding the weights can be accomplished in two steps, where the document to be updated is first removed and then added into the system with new attributes and terms. However, the equations for removal and addition of a document can be simplified using a sequential updating. If document j is to be updated and $\underline{w}_{i(N-1)} = [w_{i1} \cdots w_{i\,j-1} w_{i\,j+1} \cdots w_{iN}]^T$ represents the weight vector emanating from term $t_i \in T^0$ in the input to the remaining documents, i.e.

excluding the j^{th} one, then using (4.18) with $L = 1$ the weight vector $w^{(1)}_{i(N-1)}$ after deletion will be

$$\underline{w}^{(1)}_{i(N-1)} = \underline{w}^{(0)}_{i(N-1)} - u_0^{-1}\underline{v}_0 w^{(0)}_{ij}, \quad \forall\, t_i \in T^0 \tag{4.19}$$

where $w^{(0)}_{i1} \rightarrow w^{(0)}_{ij}$ represents the weight connection of the document to be removed (i.e. document j), $U_0 \rightarrow u_0$ is a scalar, and $V_0 \rightarrow \underline{v}_0$ is a column vector. Now, we use (4.5) to add the updated document along with its new terms and its new score $\hat{r}_{ij}$. In this case $V \rightarrow \underline{v}_1$ and $B \rightarrow \underline{b}_1$ will be column vectors and $U \rightarrow u_1$ is a scalar. Therefore, we have

$$\underline{w}^{(2)}_i = \begin{pmatrix} \delta(i)\underline{w}^{(1)}_{i(N-1)} + \underline{v}_1(\hat{r}_{ij} - \delta(i)\underline{b}_1{}^T\underline{w}^{(1)}_{i(N-1)}) \\ u_1(\hat{r}_{ij} - \delta(i)\underline{b}_1{}^T\underline{w}^{(1)}_{i(N-1)}) \end{pmatrix} \tag{4.20}$$

where

$$\delta(i) = \begin{cases} 1 & \text{if } t_i \in T^0 \\ 0 & \text{otherwise} \end{cases} \tag{4.21}$$

Note that $\delta(i)$ is needed to account for the remaining terms after document j is deleted. Combining (4.19) and (4.20), the final equations to modify the weights of the system, when document j is updated, are

$$\underline{w}^{(2)}_i = \begin{pmatrix} \delta(i)\underline{w}^{(0)}_{i(N-1)} - \delta(i)\underline{v}_1\underline{b}_1{}^T\underline{w}^{(0)}_{i(N-1)} - \delta(i)u_0^{-1}\underline{v}_0 w^{(0)}_{ij} + \delta(i)u_0^{-1}\underline{v}_1\underline{b}_1{}^T\underline{v}_0 w^{(0)}_{ij} + \underline{v}_1\hat{r}_{ij} \\ u_1\hat{r}_{ij} - \delta(i)u_1\underline{b}_1{}^T w^{(0)}_{i(N-1)} + \delta(i)u_0^{-1}u_1\underline{b}_1{}^T\underline{v}_0 w^{(0)}_{ij} \end{pmatrix} \tag{4.22}$$

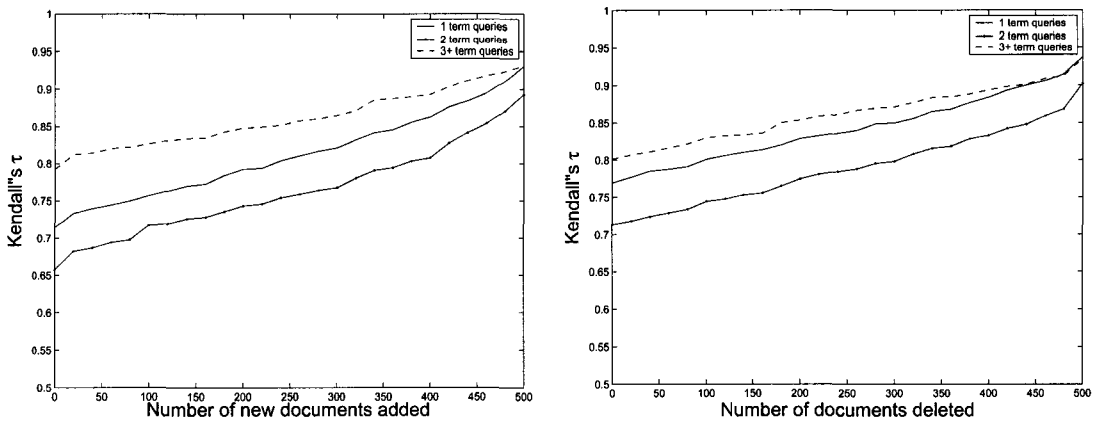
where the last element of $\underline{w}^{(2)}_i$ is the weight between term t_i and document j.

4.5 Experimental Results

The model-reference following developed for learning in dynamic environments is tested in this section when new documents with new terms are introduced or obsolete ones are removed from the initially trained system in Chapter 3. The experiments in

this section are performed on only one collection consisting of about 3700 documents and the results are obtained for 800, 724, and 1037 1-term, 2-term and 3^+-term queries, respectively. Two experiments are conducted here. In the first experiment, 500 new documents with new terms are introduced into the initially trained system in steps of 20. The algorithm in Section 4.2 was then applied to update the weights of the initially trained network using the regression-based learning in Section 3.2. Every time 20 new documents are added, the value of τ_b is computed for the most commonly used 1-term, 2-term and 3^+-term queries. Figure 4.2(a) shows the plots of the averaged τ_b for these three cases during the course of the model-reference following. Again, the Kendall's τ_b measure is generated based on the top twenty documents listed by the updated system against the benchmark. Note that in this phase, the benchmark corresponds to the log file of users' votes for the enlarged document set that included the additional 500 documents. As can be seen, for the 1-term, 2-term and 3^+-term queries the values of τ_b increased from 0.714, 0.659 and 0.792 to 0.93, 0.89 and 0.93, respectively. This increasing trend of τ_b in Figure 4.2(a) implies that the retrieved results of the system after adding the new documents approach those of the initially trained system based upon the enlarged document set. It is interesting to note that the retrieved results for the 3^+-term queries are much closer to the benchmark. This is also consistent with the results of initial model reference learning modes shown in Table 3.1. This may be attributed to the fact that incorporating more than 3 relevant terms in queries closely captures the user concepts and requirements.

Another experiment was conducted when 500 documents were deleted from the initially trained system. The benchmark for this study was the log file of the users' votes and TRS results for the reduced set of documents that excluded the 500 documents. Documents were deleted in steps of 20 and the value of τ_b was generated at every step for the most commonly used 1-term, 2-term and 3^+-term queries in the log file. Figure 4.2(b) shows the plots of the averaged τ_b for these queries. As can be



(a) (b)

Figure 4.2: Model-reference following: (a) τ_b when 500 new documents are added and (b) τ_b when 500 documents are deleted.

observed from these plots, the value of τ_b for these queries increased from 0.77, 0.71 and 0.80 to 0.938, 0.902 and 0.935, respectively. Again, these results attest to the fact that the system after the model-reference following closely captures the underlying dynamic behavior of the model when documents are added or deleted.

4.6 Conclusions

In this chapter we presented three model-reference following learning algorithms for the operations of document addition, deletion, and updating. The algorithms adjust the exiting parameters in the regulator and modify dynamically the structure of our connectionist retrieval system in order to respond adequately to such changes in the database environment where terms or documents are constantly introduced or removed.

For dynamic databases where at any moment new documents might arrive, obsolete ones should be removed or modified, it is important to devise algorithms that can incorporate these changes in the retrieval system efficiently. Since new documents might arrive asynchronously it is impractical and perhaps inefficient to make changes

to the weight connections in the system in an on-line fashion. However, it is possible to make them off-line and in batch mode, e.g. every 2-3 days, when the activity-level of the retrieval system is low. More importantly, these changes must be incorporated without impacting the previously stored information, i.e. offering flexibility while guaranteeing the long-term stability of the system.

For document addition when a new document is introduced along with its new and old concepts (terms) the structure of our connectionist system can be adjusted to incorporate this new information. The output and the first layer associated with the documents and the input layer and second layers associated with the concepts are modified. One neuron is created in both the first and the output layers to accommodate every new document and several neurons are created in the input and third layers to accommodate the new terms in the new document added. For document deletion it was found that this process was not the exact inverse process of the document addition process because the documents to be deleted could be at any place of the network structure rather than at the far right end of the network structure, place of the added documents. However, moving the documents to be deleted at the far right end the deletion process could be seen as the inverse operation (in mathematical terms) of the document addition process. When a document is deleted all the layers of the network are narrowed; the output and the first layer are reduced in one neuron while the second and third layers are reduced in one or more neurons depending on the terms that uniquely belong to the deleted document.

Batch and recursive versions of the algorithms were developed and tested on a HP-collection containing around 3700 documents. The test results showed an increasing trend of the correlation measure (τ_b) used in the experiments for document addition. This implies that the retrieval system adjusts its performance after adding the new documents. It was noticed that the retrieval results for the 3^+-term queries were closer to the benchmark. This may be attributed to the fact that incorporating more

than 3 relevant terms in queries closely captures the user concepts and requirements.

The high value of the correlation measure τ_b between document lists produced by our MRTRS system and those of the model text retrieval indicated the good capability of our system in following the time-varying (slow) model reference system when new documents are introduced, deleted, or updated. Although τ_b did not reach a perfect match of $\tau_b = 1$ it got very close to it (around 0.93), attesting the fact that our system can follow very closely the external reference model when new documents are added or some documents are deleted or modified in context.

CHAPTER 5

RELEVANCE FEEDBACK LEARNING FOR TEXT RETRIEVAL

5.1 Introduction

Often the original submitted query does not meet the specific user requirements in terms of the listed documents, their relevancy and relative scores. To meet the users requirement and at the same time preserve the previous learning we propose a new learning mechanism to capture relevance feedback information. In our proposed MRTRS system, the relevance feedback information can be incorporated using two different mechanisms, depending on the nature of the user feedback, which could be binary or score based. The user's feedback is binary when he/she specifies, via a click-through selection scheme, the relevancy of some documents without assigning to them specific scores. Whereas in the score-based scheme the user specifies either the desired scores of the selected documents or their relative positions from which their respective scores can be estimated. Relevance feedback learning can be implemented using either regression learning mechanism for the scored-based case, or the classification-based learning mechanism for the click-through selection case. For both cases, relevance feedback learning is ultimately accomplished by updating the parameters of the regulator or the weights of the first layer in the connectionist network. In the MRTRS framework, the lower feedback loop (relevance feedback loop)

of the system in Figure 2.1 provides expert users' votes on relevant and non-relevant documents to the adjustment mechanism, which in turn updates the parameters of the regulator to meet the user requirements by imposing relevance feedback.

The relevance feedback learning should not only be applied to single-term queries but also to various combinations of multi-term queries. This requirement is crucial in light of the fact that most text queries for special-purpose databases are long [12] while most queries for general-purpose databases are short [71–73]. Thus, the development of learning algorithms (for both single-term and multi-term queries) capable of incorporating users' expertise via relevance feedback is the main purpose of this chapter.

The organization of this chapter is as follows. Section 5.2 covers the details to casting the relevance feedback learning in a regression framework. Section 5.3 uses the classification approach for relevance feedback learning for cases where only partial information is provided, i.e. when only the relevancy or non-relevancy of some documents are known. Section 5.4 develops the algorithm for relevance feedback learning using multi-term queries and presents a convergence analysis of the algorithms involved. Section 5.5 presents the experimental results to evaluate the performance of our relevance feedback algorithms, both for the single-term and the multi-term queries cases. Finally, Section 5.6 presents the conclusions on the relevance feedback learning algorithms.

5.2 Relevance Feedback Learning as a Regression Problem

As in Section 3.2, the problem of relevance feedback learning can be cast in a constrained optimization framework. In this case, the main objective is to transform the previously mapped query (obtained in phase 1), to a new optimal query $\hat{\underline{q}}_i$ that is the closest to the old one (in the Euclidian norm) and further satisfies the new constraint

$D^T \hat{\underline{q}}_i = \hat{\underline{r}}_i$. Thus, the Lagrangian function in this regression-based learning should be modified to

$$J(\hat{\underline{q}}_i, \Delta \underline{w}_i) = \frac{1}{2}(\hat{\underline{q}}_i - \underline{q}_i)^T(\hat{\underline{q}}_i - \underline{q}_i) + \sum_{j=1}^{N} \Delta w_{ij}(\hat{r}_{ij} - \underline{d}_j^T \hat{\underline{q}}_i) \tag{5.1}$$

where Δw_{ij} represents the incremental change in the Lagrangian multiplier. Then, the LS solution for $\hat{\underline{q}}_i$ becomes

$$\hat{\underline{q}}_i = \underline{q}_i + \sum_{j=1}^{N} \Delta w_{ij} \underline{d}_j = \underline{q}_i + D\Delta \underline{w}_i \tag{5.2}$$

and the optimal solution for $\Delta \underline{w}_i$ is,

$$\Delta \underline{w}_i = (D^T D)^{-1} \underline{\delta}_i \tag{5.3}$$

where $\underline{\delta}_i = \hat{\underline{r}}_i - \underline{r}_i$ with $D^T \underline{q}_i = \underline{r}_i$ i.e. score vector for the old query. Thus, the weight vector $\underline{w}_i$ in the first layer can be updated to $\hat{\underline{w}}_i$ using $\hat{\underline{w}}_i = \underline{w}_i + \Delta \underline{w}_i$, where $\hat{\underline{w}}_i = (D^T D)^{-1} \hat{\underline{r}}_i$ meets the new score requirements $\hat{\underline{r}}_i$.

If there are K voted documents whose scores need to be modified in the regression mode, the new score vector becomes,

$$\hat{\underline{r}}_i = \underline{r}_i + \sum_{k \in S} \underline{e}_k \Delta r_k \tag{5.4}$$

where $\underline{e}_k$ is as defined before, Δr_k is the incremental change corresponding to user-specified score of the voted documents $\underline{d}_k$, and S represents the set of voted documents with cardinality $|S| = K$. In this case, the parameters of the regulator or the weights of the first layer must be updated using the weight increment vector,

$$\Delta \underline{w}_i = \sum_{k \in S} \underline{b}_k \Delta r_k \tag{5.5}$$

where $\underline{b}_k = {A_0}^{-1} \underline{e}_k$ is the k^{th} column of the matrix ${A_0}^{-1} = (D^T D)^{-1}$.

From the above equation it can be observed that document voting corresponds to linearly adjusting the weights to incorporate user information. Since this weight

updating is only implemented for the weights associated with the term t_i, it is ensured that information learnt in the previous learning is not lost while at the same time allowing for adaptation of new associations.

Remarks

5.1 The relevance feedback learning in this section can be implemented either on-line for every user (not typical) or in batch mode after collecting all the users' votes on various queries and forming a log file. In the former case, the score for a click-through selection can be specified internally using a particular scoring scheme [35] and without any user involvement. In the latter case, frequency of votes, expertise level of the user, date in which voting takes place, date in which the document is last modified or any other meaningful criterion can be used in conjunction with some specific heuristic rules to arrive at the desired score vector $\underline{\hat{r}}_i$ for every query in the log file. This is used in Section 5.5, though the same updating rules can also be applied for on-line iterative-based learning.

5.2 Although during the initial model-reference learning (phase 1) the weights of the first layer of Figure 3.1 are computed for all the single-term queries, due to the linearity of the network one can perform weight adaptation during relevance feedback for multi-term queries as well. For instance if a two-term query containing terms t_i and t_j is applied, the weights associated with these terms i.e. $\underline{w}_i$ and $\underline{w}_j$ will undergo adaptation. Since the global corpus weights of these terms are known, the contribution of each term toward the desired response will be determined based upon these weights. However, this fine-tuning may slightly change the response for other queries that contain the same terms. To remedy this problem, a new learning algorithm is developed in Section 5.4, which starts from the indexing results, i.e. projection matrix P_D for the initial training of the regulator, and then applies relevance feedback learning based upon a set of single and multi-term queries and their associated votes in the log file. Section 5.5 gives the results of this method and their benchmarking

with the original single-term relevance feedback learning method.

5.3 An alternative relevance feedback mechanism [74] that provides flexibility in the position of the documents (typical in general-purpose TRS) can be devised using our network structure. In this scheme, the terms in the voted documents can be modified to elevate the relevant documents while demoting the non-relevant ones. This can be accomplished in our system by updating the corresponding document terms in the retrieval layer. The learning is based upon Gram-Schmidt orthogonalization [75] in conjunction with a node creation strategy to incorporate the user feedback.

5.3 Relevance Feedback Learning as a Classification Problem

If the user identifies the most relevant document(s) via click-through selection, he/she may not be interested in assigning scores, rather contend in specifying whether or not a document is relevant to a particular query. Then, the new score vector can still be found using (5.4) for this classification-based learning (two class) with the minor difference that $\Delta r_k = \pm 2$, where $+$ is used when the k^{th} document status should be changed from non-relevant to relevant and vice versa. Additionally, it is easy to show that the new diagonal matrix $\widehat{\Lambda}_i$ should be changed to

$$\widehat{\Lambda}_i = \Lambda_i + \sum_{k \in S} \underline{e}_k \underline{e}_k^T \Delta r_k \tag{5.6}$$

in order to satisfy the requirement $\widehat{\Lambda}_i \underline{\hat{r}}_i = \underline{1}$. With these minor modifications, the weight updating increment $\Delta \underline{w}_i$ becomes

$$\Delta \underline{w}_i = \widehat{\Lambda}_i \sum_{k \in S} \underline{b}_k \Delta r_k + \sum_{k \in S} \underline{e}_k \underline{b}_k^T \Delta r_k \underline{r}_i \tag{5.7}$$

where $\underline{b}_k = {A_0}^{-1} \underline{e}_k$ is the k^{th} column of the matrix ${A_0}^{-1} = (D^T D)^{-1}$ and S represents the set of voted documents (See Section 3.3 in Chapter 3).

To implement relevance feedback learning in a classification framework we only need a small set of relevant (positive feedback) and non-relevant (negative feedback);

without soliciting the scores of the documents as in the regression framework. This characteristic make the classification learning scheme very appealing for environments where is difficult to gather this information.

5.4 Relevance Feedback Learning for Multi-Term Queries

As mentioned in the introduction, the majority of the queries that arrive at most retrieval engines contains two or more terms. Therefore, it is important to consider this fact into the learning mechanism. The purpose of this section is to extend relevance feedback learning to account for the multi-term as well as the single-term query learning. Relevance feedback information for a particular set $\{\underline{q}_i\}_{i=1}^{k}$ of queries is used to update specific columns of mapping matrix Z; the learning is somewhat similar to that for initial training with the difference that here our goal is to update the mapping matrix instead of setting it up.

Suppose that the number of different terms among the training queries $\{\underline{q}_i\}_{i=1}^{k}$ is T. The problem of finding the updates $\Delta\underline{z}_{j_1}$, $\Delta\underline{z}_{j_2}$, ... $\Delta\underline{z}_{j_T}$ for T columns $\underline{z}_{j_1}$, $\underline{z}_{j_2}$, ... $\underline{z}_{j_T}$ of mapping matrix Z (where j_h corresponds to term t_{j_h} for $h \in [1,T]$), given a set of T-term queries $\underline{q}_i = \sum_{l=1}^{T} q_{ij_l}\underline{e}_{j_l}$, $i \in [1,k]$ with $k \geq T$, with their respective outputs $\hat{\underline{r}}_i$'s, where $\underline{e}_{j_l}$ is the vector containing 1 at the j_l^{th} position and zero elsewhere, can be cast in an optimization framework. The goal here is to find $\Delta\underline{z}_{j_1}$,..., $\Delta\underline{z}_{j_T}$ specific columns of $\Delta Z = Z - Z^{(0)}$ and $\Delta\underline{v}_i$'s that minimize the Lagrangian function,

$$J(\Delta\underline{z}_{j_1},\ldots,\Delta\underline{z}_{j_T},\Delta\underline{v}_1,\ldots,\Delta\underline{v}_k) = \frac{1}{2}\sum_{i=1}^{k}(\hat{\underline{q}}_i - \underline{q}_i^{(0)})^T(\hat{\underline{q}}_i - \underline{q}_i^{(0)}) + \sum_{i=1}^{k}\Delta\underline{v}_i^T(\hat{\underline{r}}_i - D^T\hat{\underline{q}}_i) \tag{5.8}$$

where $\underline{q}_i^{(0)} = Z^{(0)}\underline{q}_i$ and $\hat{\underline{q}}_i = Z\underline{q}_i$ are the mapped queries at the output of the regulator of our MRTRS system when the old $Z^{(0)}$ and the new Z mapping matrices are applied to the submitted query $\underline{q}_i$. Since we require linear optimal mapping of the form $\hat{\underline{q}}_i = Z\underline{q}_i = \sum_{l=1}^{T} q_{ij_l}\underline{z}_{j_l}$; $\quad i \in [1,k]$, we set the derivative of J wrt $\Delta\underline{z}_{j_l}$ to $\underline{0}$.

This yields

$$\sum_{i=1}^{k}(\hat{\underline{q}}_i - \underline{q}_i^{(0)})q_{ij_l} - \sum_{i=1}^{k} q_{ij_l} D\Delta\underline{v}_i = \underline{0}; \qquad l \in [1,T] \tag{5.9}$$

Given that the difference between the optimal query $\hat{\underline{q}}_i$ and $\underline{q}_i^{(0)}$ can be expressed as $\hat{\underline{q}}_i - \underline{q}_i^{(0)} = \sum_{p=1}^{T} q_{ij_p}\Delta\underline{z}_{j_p}$ we can transform (5.9) into

$$\sum_{p=1}^{T}\left(\sum_{i=1}^{k} q_{ij_p}q_{ij_l}\right)\Delta\underline{z}_{j_p} = D\sum_{i=1}^{k} q_{ij_l}\Delta\underline{v}_i = D\Delta\hat{\underline{w}}_l(k); \qquad l \in [1,T] \tag{5.10}$$

which must hold for the query terms $t_{j_1},\ldots,t_{j_T}$. Here, $\Delta\hat{\underline{w}}_l(k) = \sum_{i=1}^{k} q_{ij_l}\Delta\underline{v}_i$; $l \in [1,T]$, and the term in the bracket in (5.10) represents a correlation measure between the terms in the set of queries.

Now, since the constrains in (5.8) are given by $\hat{\underline{r}}_i = D^T\hat{\underline{q}}_i\,; i = 1.2\ldots k$, and $\underline{r}_i^{(0)} = D^T\underline{q}_i^{(0)}$ this implies that (5.9), after premultiplying by D^T, can be rewritten as $\sum_{i=1}^{k}\left(\delta\hat{\underline{r}}_i - D^TD\Delta\underline{v}_i\right)q_{ij_l} = \underline{0}$; $j_l \in [1,T]$, where $\delta\hat{\underline{r}}_i = \hat{\underline{r}}_i - \underline{r}_i^{(0)}$. From here, the Lagrange multipliers $\Delta\underline{v}_i = (D^TD)^{-1}\delta\hat{\underline{r}}_i$, $i \in [1,k]$ are found and then used to form the columns of $\Delta\widehat{W}(k) = [\Delta\hat{\underline{w}}_1(k)\ \Delta\hat{\underline{w}}_2(k)\ldots\Delta\hat{\underline{w}}_T(k)]^T$ to obtain $\Delta\widehat{W}(k) = (D^TD)^{-1}\sum_{i=1}^{k}\delta\hat{\underline{r}}_i\underline{q}_i^T = \sum_{i=1}^{k}\Delta\underline{v}_i\underline{q}_i^T$.

To find a solution for $\underline{z}_{jl}$'s let us define the matrix $H(k) = [h_{pl}(k)]$, $p,l = 1,2,\ldots,T$ with elements $h_{pl}(k) = \sum_{i=1}^{k} q_{ij_p}q_{ij_l}$. Then, (5.10) can be rewritten in matrix form $\Delta Z(k)H(k) = D\Delta\widehat{W}(k)$, where matrix $\Delta Z(k) = [\Delta\underline{z}_{j_1}\Delta\underline{z}_{j_2}\ldots\Delta\underline{z}_{j_T}]$ contains columns $j_1, j_2, \ldots, j_T$ of $Z - Z^{(0)}$, where $Z^{(0)} = D\Theta(0)$ and $\Theta(0) = \widehat{W}(0)H^{-1}(0)$. If we solve for $\Delta Z(k)$, we get

$$\Delta Z(k) = D\Delta\widehat{W}(k)H^{-1}(k) \tag{5.11}$$

$$= D\Delta\Theta(k), \tag{5.12}$$

where $\Delta\Theta(k) = \Theta(k) - \Theta(0) = \Delta\widehat{W}(k)H^{-1}(k)$. As can be seen in (5.11), solving for $\Delta\Theta(k)$ requires computing $\Delta\widehat{W}(k)$ and $H^{-1}(k)$, where $H^{-1}(k)$ exists and is easy

to compute using the matrix inversion lemma as in Section 3.2 in Chapter 3. The recursive equations for $\Delta\widehat{W}(k) = \sum_{i=1}^{k} \Delta\underline{v}_i\underline{q}_i^T$ and $H^{-1}(k)$ are

$$\Delta\widehat{W}(k) = \Delta\widehat{W}(k-1) + \Delta\underline{v}_k\underline{q}_k^T. \tag{5.13}$$

$$H^{-1}(k) = H^{-1}(k-1) - \frac{H^{-1}(k-1)\underline{q}_k\underline{q}_k^T H^{-1}(k-1)}{1+\underline{q}_k^T H^{-1}(k-1)\underline{q}_k} \tag{5.14}$$

Now, the recursive equations for $\Delta\Theta(k) = \Delta\widehat{W}(k)H^{-1}(k)$ can be found using the recursive equation for $\Delta\widehat{W}(k)$ in (5.13) as follows:

$$\begin{aligned}
\Delta\Theta(k) &= (\Delta\widehat{W}(k-1) + \Delta\underline{v}_k\underline{q}_k^T)H^{-1}(k) \\
&= (\Delta\Theta(k-1)H(k-1) + \Delta\underline{v}_k\underline{q}_k^T)H^{-1}(k) \\
&= (\Delta\Theta(k-1)[H(k) - \underline{q}_k\underline{q}_k^T] + \Delta\underline{v}_k\underline{q}_k^T)H^{-1}(k) \\
&= \Delta\Theta(k-1) + [\Delta\underline{v}_k - \Delta\Theta(k-1)\underline{q}_k]\underline{q}_k^T H^{-1}(k)
\end{aligned} \tag{5.15}$$

It is convenient to express this updating equation in terms of the feedback it receives from both the user and the retrieval system. In one hand, it receives the query and the desired scores from the user, and on the other hand, it receives the output of the retrieval system for the submitted query. The goal of the updating mechanism of our MRTRS is to provide an error signal used to adjust the parameters of the regulator. Thus, (5.15) can be expressed as

$$\begin{aligned}
\Delta\Theta(k) &= \Delta\Theta(k-1) + [\Delta\underline{v}_k - \Delta\Theta(k-1)\underline{q}_k]\underline{q}_k^T H^{-1}(k) \\
&= \Delta\Theta(k-1) + (D^T D)^{-1}[\delta\hat{\underline{r}}_k - (D^T D)\Delta\Theta(k-1)\underline{q}_k]\underline{q}_k^T H^{-1}(k)
\end{aligned} \tag{5.16}$$

From the definition of $\Delta\Theta(k-1) = \Theta(k-1) - \Theta(0)$, the term $(D^T D)\Delta\Theta(k-1)\underline{q}_k$ can be simplified to

$$\begin{aligned}
(D^T D)\Delta\Theta(k-1)\underline{q}_k &= (D^T D)\Theta(k-1)\underline{q}_k - (D^T D)\Theta(0)\underline{q}_k \\
&= \underline{r}_k - \underline{r}_k^{(0)}
\end{aligned} \tag{5.17}$$

Thus, the term in the brackets on the right hand side of (5.16) becomes $\delta\underline{\hat{r}}_k - (\underline{r}_k - \underline{r}_k^{(0)}) = \underline{\hat{r}}_k - \underline{r}_k$, where $\underline{r}_k$ is the output of the retrieval system for query $\underline{q}_k$ when mapping matrix $\Theta(k-1)$ is used and $\underline{\hat{r}}_k$ is the desired score output for query $\underline{q}_k$. Now, let $\Delta\underline{\hat{r}}_k$ be the difference between the desired score vector $\underline{\hat{r}}_k$ and $\underline{r}_k$, i.e. $\Delta\underline{\hat{r}}_k = \underline{\hat{r}}_k - \underline{r}_k$. If we plug this value in (5.16) then the updating equation for $\Delta\Theta(k)$ can be expressed as

$$\Delta\Theta(k) = \Delta\Theta(k-1) + (D^T D)^{-1}\Delta\underline{\hat{r}}_k \underline{q}_k^T H^{-1}(k) = \Delta\Theta(k-1) + (D^T D)^{-1}\Delta\underline{\hat{r}}_k \boldsymbol{K}^T(k) \tag{5.18}$$

where $\boldsymbol{K}(k) = H^{-1}(k)\underline{q}_k$ is the gain. If we replace $\Delta\Theta(k)$ by $[\Theta(k) - \Theta(0)]$ and $\Delta\Theta(k-1)$ by $[\Theta(k-1) - \Theta(0)]$ in (5.18) then the updating equation for $\Theta(k)$ becomes

$$\Theta(k) = \Theta(k-1) + (D^T D)^{-1}\Delta\underline{\hat{r}}_k \boldsymbol{K}^T(k) \tag{5.19}$$

Now, if we define $P(k) = H^{-1}(k)$ and plug it into (5.14) we arrive at the following recursion [76]:

$$P(k) = P(k-1) - \frac{P(k-1)\underline{q}_k \underline{q}_k^T P(k-1)}{1 + \underline{q}_k^T P(k-1)\underline{q}_k} \tag{5.20}$$

The updating equation (5.19) for finding the mapping matrix $\Theta(k)$ together with the expression for the gain $\boldsymbol{K}(k)$, and the recursion formula in (5.20) constitute the multi-variable recursive (multi-output case) least-squares (RLS) [57, 77] algorithm:

$$\begin{aligned} \Theta(k) &= \Theta(k-1) + (D^T D)^{-1}\Delta\underline{\hat{r}}_k \boldsymbol{K}^T(k) \\ \boldsymbol{K}(k) &= P(k)\underline{q}_k = \frac{P(k-1)\underline{q}_k}{1 + \underline{q}_k^T P(k-1)\underline{q}_k} \\ P(k) &= P(k-1) - \frac{P(k-1)\underline{q}_k \underline{q}_k^T P(k-1)}{1 + \underline{q}_k^T P(k-1)\underline{q}_k} \end{aligned} \tag{5.21}$$

with initial conditions $\Theta(0) = (D^T D)^{-1} D^T$ and $P(0) = I$.

5.4.1 Special Case: Single-Term Queries

Although the recursive equations for finding mapping matrix $\Theta(k)$ are simple in structure, the number of computations involved can be an impediment for an online

implementation. Potentially, $P(k)$ could be a full $M \times M$ matrix, where M is the number of distinct terms found in the documents, hence the number of operations to update P in (5.20) is of order $O(M^2)$. Though improbable, this case might happen in the long run after the system has processed a lot of queries. As mentioned throughout this work queries with few terms, especially those with only one term, are abundant. Therefore, algorithms that exploit these types of queries and use less operations in the algorithm (5.21) for finding mapping matrix $\Theta(k)$ are desired.

Let us consider the case where we rely only on single-term queries to update both the recursion in (5.21) and the mapping matrix Θ. Thus, matrix $P(k)$ becomes a diagonal matrix with element

$$P_{ii}(k) = \frac{1}{1+n_i} \tag{5.22}$$

where $n_i \leq k$ is the number of single-term queries $\underline{q}_i = \underline{e}_i$ submitted prior to query $\underline{q}_{k+1}$. In addition, it can be shown that $\sum_{i=1}^{M} n_i = k$. The proof that $P(k)$ is a diagonal matrix with elements given by (5.22) can be done by the method of induction. That is, we know that $P(0) = I$ is diagonal now assure that $P(k)$ is also diagonal for the submitted single-term query $\underline{q}_k = \underline{e}_i$. We must prove that $P(k+1)$ is also a diagonal matrix whose i^{th} diagonal is given by $P_{ii}(k+1) = \frac{1}{1+(n_i+1)}$ and the other elements remain unchanged. From (5.21), the recursion for $P(k+1)$ and query $\underline{q}_{k+1} = \underline{e}_i$ is expressed as

$$P(k+1) = P(k) - \frac{P(k)\underline{e}_i\underline{e}_i^T P(k)}{1+\underline{e}_i^T P(k)\underline{e}_i} \tag{5.23}$$

Now, since $P(k)$ is a diagonal matrix the vector $P(k)\underline{e}_i$ contains only one non-zero element, i.e. its i^{th} element that is $P_{ii}(k)$. Then, from (5.23), it follows that

$$P_{ii}(k+1) = \frac{P_{ii}(k)}{1+P_{ii}(k)} \tag{5.24}$$

$$P_{lp}(k+1) = P_{lp}(k) \quad \text{for} \quad l \neq i \text{ or } p \neq i \tag{5.25}$$

Plugging $P_{ii}(k)$ from (5.22) into the right-hand side of (5.24) yields $P_{ii}(k+1) = 1/(1+(n_i+1))$, which is the desired result.

Furthermore for this special case, the gain $\boldsymbol{K}(k) = P(k)\underline{q}_k$ for the submitted query $\underline{q}_k = \underline{e}_i$ is a column vector whose i^{th} element is $P_{ii}(k) = 1/(1+n_i)$ while the others are zero. Now, using the value for the gain into (5.21) and assuming that the submitted query at time k is single term, i.e. $\underline{q}_k = \underline{e}_i$, the RLS version for the single-term query case becomes:

$$\Theta_i(k) = \Theta_i(k-1) + \gamma_i(k)(D^TD)^{-1}\Delta\hat{\underline{r}}_k \tag{5.26}$$

and

$$\Theta_j(k) = \Theta_j(k-1); \quad j \neq i$$

where $\gamma_i(k) = 1/(1+n_i)$ and $\Theta_i(k)$ is the i^{th} column of $\Theta(k)$. The initial condition $\Theta(0) = (D^TD)^{-1}D^T$. Equation (5.26) resembles the Robbins-Monro algorithm [78]. It is known [77] that if $\gamma_i(k)$ is a sequence of positive numbers satisfying

$$\sum_{k=1}^{\infty}\gamma_i(k) = \infty, \quad \lim_{k\to\infty}\gamma_i(k) = 0 \tag{5.27}$$

then the recursion in (5.26) converges. Note that not all query samples $\underline{q}_k$ modify Θ_i, but only those queries that are single-term queries $\underline{e}_i$, as can be seen in (5.26). Let $\chi = k_1k_2k_3\ldots$ be the sequence of values of k that participate during the modification of Θ_i. Using this sequence, updating equation (5.26) yields

$$\Theta_i(k_t) = \Theta_i(k_{t-1}) + \gamma_i(k_t)(D^TD)^{-1}\Delta\hat{\underline{r}}_k \tag{5.28}$$

Then, for the convergence of learning algorithm (5.26) we must prove that

$$\sum_{t=1}^{\infty}\gamma_i(k_t) = \infty, \quad \lim_{k_t\to\infty}\gamma_i(k_t) = 0 \tag{5.29}$$

where $\gamma_i(k_t) = \frac{1}{1+n_{k_t}}$. Since the variable n_{k_t} accounts for the number of times that query $\underline{e}_i$ is submitted until time k_t then $n_{k_t} = t$. In consequence $\gamma_i(k_t) = \frac{1}{1+t}$ satisfies (5.29) and column Θ_i converges with probability 1 to the solution [78].

5.5 Experimental Results

The goal of the relevance feedback is to promote/demote one or more documents to the required positions based upon the users' votes and their characteristics such as expertise level of the users, frequency of votes and document publish dates. The relevance feedback can be performed either in an iterative on-line mode for every expert user, or in a batch mode based upon votes collected over certain period of time. In the former case, it is important to guarantee that the system retains the previous learning while incorporating the user feedback. Although, the iterative relevance feedback training in Section 5.2 ensures this stability versus flexibility requirement, the effectiveness of the proposed algorithms is demonstrated for the batch mode relevance feedback based upon the votes collected in the log file for single-term as well as multi-term queries. It will also be shown that voting for single-term queries improves the accuracy of the retrieval system for multi-term queries.

5.5.1 Training Based Upon Single-Term Queries

In this study, the votes collected in the log file for 1-term queries are used for relevance feedback training in Section 5.2 and the updated system is then evaluated on the multi-term (testing) queries. The system was initially trained based on the document indexer with the projection matrix P_D as explained in the results for the initial training in Section 3.6 of Chapter 3. Relevance feedback learning is performed in both the regression and classification modes by incrementally adjusting the weights of the first layer based on (5.5) and (5.7). After the relevant feedback learning for every 1-term query in the log file, the Kendall's τ_b is computed for the training and testing queries. Figures 5.1(a) and (b) show the plots of τ_b for the entire 2386 iterations of relevance feedback for these two learning modes.

In the regression mode, the initial values of Kendall's τ_b were 0.611, 0.026, and -0.124;

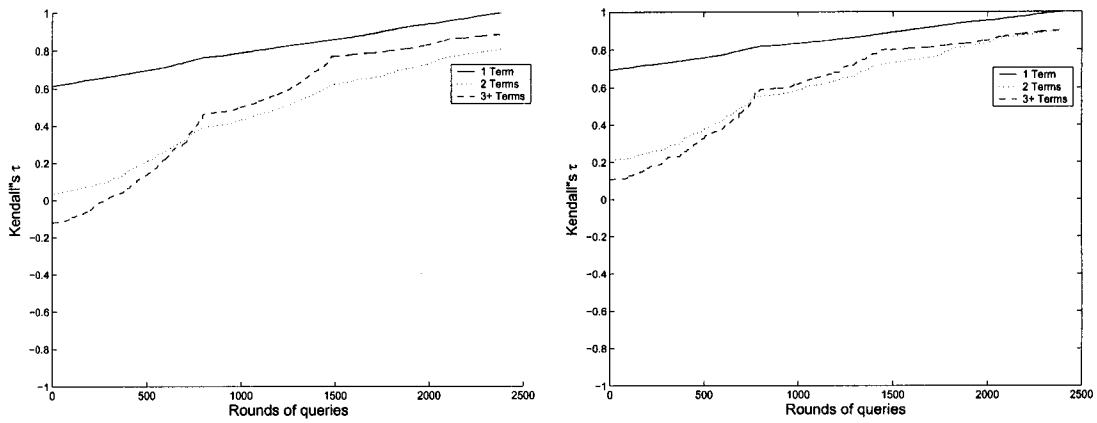
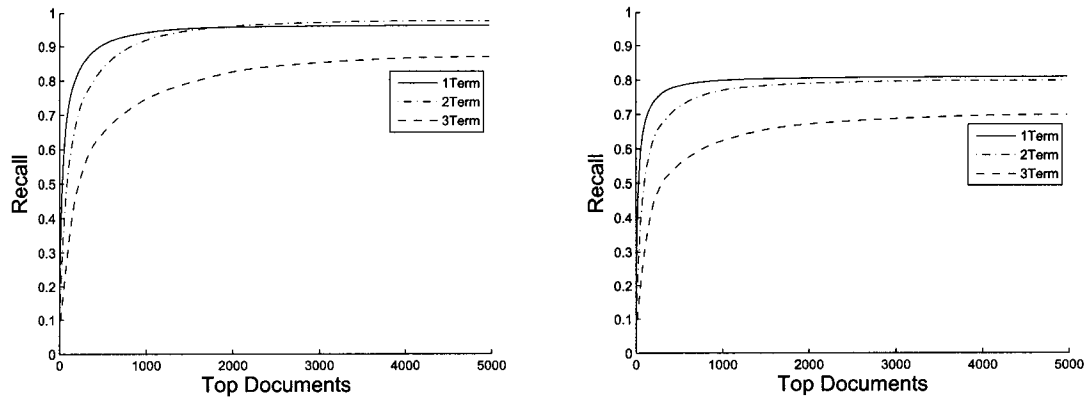


Figure 5.1: Relevance feedback learning: (a) weight update in regression mode (b) weight update in classification mode.

while the final values after relevance feedback learning became 1, 0.805 and 0.89 for 1-term, 2-term and 3^+-term queries, respectively. Similarly, in the classification mode, initial values of τ_b were 0.692, 0.210, and 0.109; whereas the final values became 1, 0.902, 0.906 for 1-term, 2-term and 3^+-term queries, respectively. This shows that although relevance feedback learning is exclusively applied for 1-term queries and the value of $\tau_b = 1$ for these queries indeed indicates perfect match, the substantial increase in the values of τ_b for the multi-term queries is indicative of the generalization capability of the system. This is especially true when there are terms in the multi-term queries that did not receive any relevance feedback training. Note that this relevance feedback adaptability is achieved without jeopardizing the stability of the previous trained system.

Figures 5.2 (a) and (b) show the plots of recall measure for various choices of top documents after the relevance feedback using the regression and classification learning modes, respectively. These plots are generated for the most commonly used 1-term, 2-term and 3^+-term queries in the log file by taking the average of the recall values. As mentioned before the system was initially trained using the indexer results.



(a) (b)

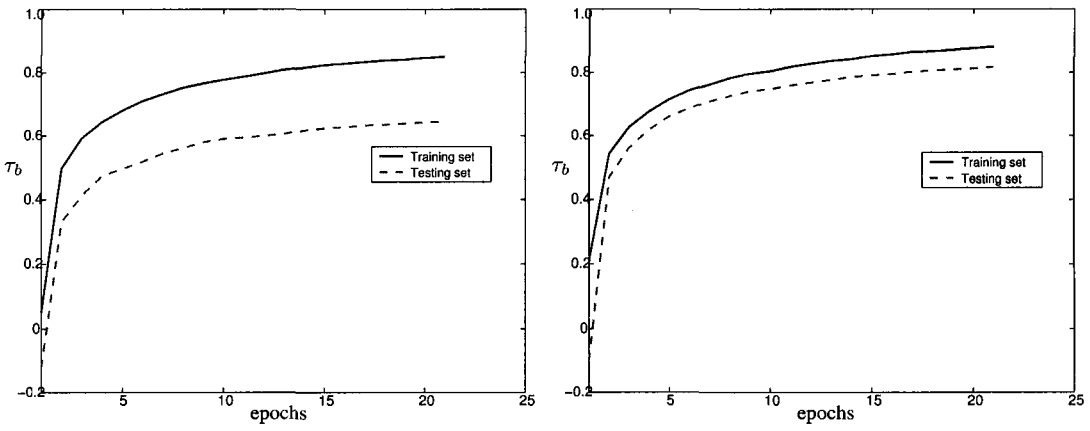
Figure 5.2: Recall plot after relevance feedback learning in (a) regression mode and (b) classification mode.

As can be seen from Figures 3.2 and 5.2 the maximum achieved recall increased considerably after the relevance feedback learning. In the case of regression-based relevance feedback learning, the maximum achieved recall increased from 0.673, 0.720 and 0.657 (refer Figure 3.2 (a)) to 0.961, 0.980 and 0.869 (refer Figure 5.2 (a)) for the 1-term, 2-term and 3^+-term queries, respectively. Similarly for the classification mode, the maximum recall achieved increased from 0.551, 0.453 and 0.385 (refer Figure 3.2 (b)) to 0.808, 0.797 and 0.698 (refer Figure 5.2 (b)) for the 1-term, 2-term and 3^+-term queries, respectively. The above results indicate the effectiveness of our system in achieving the relevance feedback learning without jeopardizing the stability of the previously trained system.

5.5.2 Training Based Upon Multi-Term Queries

In this study, the multi-term query learning method developed in Section 5.4 is tested and analyzed. Three different experiments were conducted. In the first experiment, we split the queries in the log file into a training set consisting of 2386 1-term queries, 1664 2-term queries, and 1661 3^+-term queries randomly drawn from the set of 1846 3^+-term queries, and a testing set consisting of the remaining 185 3^+-term queries.

Note that the purpose of this experiment is to evaluate the system performance when new 3^+-term queries are encountered. In the second experiment, we split the queries in the log file into a training set consisting of 2386 1-term queries and 1664 2-term queries, and a testing set consisting of 3^+-term queries that have common terms with the 1-term, or 2-term queries or both. That is, only those 3^+-term queries that contained terms that participated in the training are used for testing and performance evaluation. The third experiment aims at showing the convergency of the multi-variable RLS algorithm (5.21). The queries for this experiment were obtained as follows. From the 1664 2-term queries, we chose those queries whose terms alone formed single-term queries found in the set of 2386 1-term queries. We found 724 two-term queries and 230 single-term queries satisfying this criterion. The 230 single-term queries form the training set and the 724 two-term queries form the testing set.



(a) Kendall's τ_b vs epochs (first experiment)

(b) Kendall's τ_b vs epochs (second experiment)

Figure 5.3: Kendall's τ_b for two experiments

Figures 5.3(a) and (b) show the plots of Kendall's τ_b versus the number of epochs for the training (solid line) and testing sets (dashed line) for the first and second experiments, respectively. For any submitted query, only the top 20 documents were

considered to generate Kendall's τ_b measure. The training process consisted of a series of epochs in which all the samples in the training set were submitted to the system and pertinent adjustments to Θ using (5.21) were made. Furthermore, at each epoch, the Kendall's τ_b performance measures for all the samples in the training and testing sets were generated and plotted. As can be seen, in both figures, τ_b starts from very low values, indicating that the document lists from the initially trained system using the projection matrix P_D and those of the benchmark are not highly correlated. However, as relevance feedback training progresses, the τ_b values for the training set ends up at high values of approximately 0.85 and 0.90 for the first and second experiment, respectively. This indicates that the retrieval system closely captures the information content in the benchmark. Moreover, in the second experiment τ_b values for the testing set follow more closely those of the training set. This behavior is expected since the testing queries in the second experiment have common terms with the training queries. The performance of the system on the testing set, as shown in Figure 5.3(b) for the second experiment illustrates the generalization ability of the learning algorithm on multi-term queries. It is important to point out that Kendall's τ_b plots resemble most learning curves in the sense that the learning rate is high during the early stages of learning and decreases gradually as learning progresses.

Figure 5.4 shows the learning curve of the multi-variable recursive learning algorithm (5.21) using the training set of 230 single-term queries for the third experiment. The mean squared error $E(\underline{\xi}_k^T \underline{\xi}_k)$ for the training + testing sets is plotted versus the rounds of relevance feedback. During each round, all training queries are processed, the mapping matrix Θ is updated according to (5.21). At the end of each round, the mean squared error $E(\underline{\xi}_k^T \underline{\xi}_k)$ is computed and plotted.

As can be seen in Figure 5.4, the learning rate is very fast during the early rounds of feedback and decreases gradually at later stages of learning. After 30 rounds of feedback there is an improvement of six orders of magnitude in the mean square error

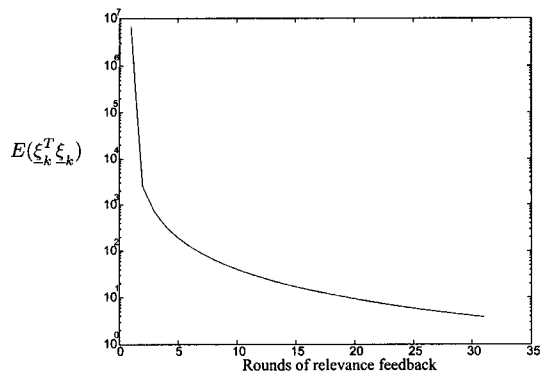


Figure 5.4: Learning curve: $E(\xi_k^T, \xi_k)$ mean squared error versus round of relevance feedback.

as the initial mean squared error at round '0' is around 10^7, while, it reaches almost 10^1 after 30 rounds.

5.6 Conclusions

In this chapter we proposed two novel methods to solve the problem of capturing users' expertise and knowledge via relevance feedback. One method uses the scored documents and hence is cast in a regression mode while the other is based on a click-trough selection and hence is cast in a classification framework. The learning can be implemented for either single-term or multi-term queries by adapting the parameters in the first layer of our connectionist network or in the regulator of the MRTRS. The user feedback is employed to administer the expert user voting based upon frequency of votes, users' expertise and other criteria. The second and third layers perform document-to-term mapping and search/retrieval tasks. The effectiveness of the proposed algorithms is demonstrated on a relatively large domain-specific text database containing various HP products.

It was found that our MRTRS captures new information without impairing its performance for the already stored information. We used the Kendal's τ correlation

and the recall measures for comparing the list of documents provided by our system against the benchmark list of documents provided by experts users for a small but representative set of queries. According to the results both the regression and classification modes show similar performance because for both the correlation measure approaches '1' (maximum value) in very few rounds of feedback. Surprisingly, the performance of our MRTRS system for three-term queries was better than for two-term queries, though the performance of both systems for the two-term queries was better that for the three-term queries at early stages of learning. This, implies that the system learned better the concepts and possible associations among them for queries with more than two terms.

The experimental results using the recall measure for comparing the retrieved list of our MRTRS against the experts users' list for the same set of queries show that indeed both the regression and classification learning modes can capture the relevance information from the users. The recall plots for these learning modes show that the measure increases as more documents are considered. For instance, in the regression mode, the recall increases from 0.673, 0.720, and 0.657 to 0.961, 0.980, and 0.869 for the 1-term, 2-term, and 3^{+}-term queries, respectively, while in the classification mode, increments from 0.551, 0.453, and 0.385 to 0.808, 0.797, and 0.698 were observed for the same queries. The recall plots indicate the effectiveness of our system in achieving the relevance feedback learning without jeopardizing the stability of the previously trained system.

Evaluation of our MRTRS based upon several collection of the HP database indicates that the proposed model MRTRS is indeed very effective in finding relevant information from a small set of training queries, typical scenario of most retrieval systems. These results suggest that our system can also be used in other similar special-purpose databases where the need for accurate retrieval solutions is of great importance.

CHAPTER 6

QUERY CLUSTERING IN TEXT RETRIEVAL SYSTEMS

6.1 Introduction

The great difficulty in querying is that the user has to specify the right query in order to retrieve the desired results. Researchers [79–82] have often focused on automatic query expansion to help users better define the queries. Although, automatic query expansion improves the retrieval efficiency, there is no guarantee that the retrieved documents meet the user's requirements. As a result, it is convenient that the retrieval system suggests some relevant terms from which the user can pick one or more terms and augment or modify the original query. The purpose of this chapter is to develop a query clustering algorithm based upon our MRTRS system, that will help in providing these additional terms.

We regard query clustering as the process of partitioning the most frequent asked questions into different and disjoint clusters with the aim of providing additional information to the user. This information is usually found in the terms of the already submitted queries collected in a log file. To suggest these terms, the submitted query is evaluated against the different clusters and the terms from the cluster that are more related (in concept) to the query retrieved and suggested to the user for query refinement.

Our method for extracting information from a set of users' queries basically involves three steps: (1) mapping the original query into a space where the correlations between different terms in the same query are more 'meaningful', (2) clustering the already mapped queries using an appropriate algorithm that takes into account the possible high dimensionality of the new space, and (3) using a navigation algorithm to extract information for new queries by using the set of clusters already created. Although there have been many studies related to clustering documents [83–87], the methodology for query clustering has not been well-addressed.

Three different query domains and clustering algorithms have been tested in this chapter. The first and second domains correspond to the original and weight query domains while the third one is given in the space of documents retrieved by the search engine. In addition, we use different unsupervised methods for clustering, mainly the 2-D self-organized map (SOM) [33] and an agglomerative clustering [88] method.

The organization of this chapter is as follows. Section 6.2 presents an overview of some clustering methods presented here to exploit different query domains. Section 6.3 presents our agglomerative-based clustering method and compare it against the 2D-SOM. Test results on the the HP database are presented in this section. Section 6.4 gives the conclusions for the methods and results presented in this chapter.

6.2 Overview of Clustering Methods

An agglomerate algorithm [89–91], which is a variant of a hierarchical clustering algorithm [49], is employed here for query clustering. However, the main differences are the selection of a suitable query domain and an appropriate 'goodness' measure for a cluster and for the set of cluster formed at each iteration. The two major steps of this algorithm involves finding the 'best' pair of clusters that would maximize the 'goodness' measure and merging this pair into a single cluster. All the steps in this algorithm are described next.

(1) Preprocess each query

- Remove stop words.

- Find the stem version of each term in the query.

(2) Selection of an 'appropriate' query domain

- Possible domains:

(a) original query domain: i^{th} query is expressed as a vector in the form $\underline{q}_i = [q_{1i}, q_{2i}, \ldots, q_{Mi}]^T$, where q_{im} is '1' when the m^{th} term is present and is '0' otherwise.

(b) query weight domain: the mapped query $\underline{q}'_i$ for the i^{th} submitted query is $\underline{q}'_i = W\underline{q}_i$ (see Figure 3.1) is formed and used for clustering, where W is the mapping matrix discussed in Chapter 5.

(c) document-score domain: the original query $\underline{q}_i$ is submitted to the retrieval system and from the scores of the documents delivered a new vector $\underline{r}_i = [r_{i1}, r_{i2}, \ldots, r_{iN}]^T$ is formed, where r_{ij} is the score of the j^{th} document for the i^{th} query submitted. We can regard $\underline{r}_i$ as the effect of the query $\underline{q}_i$ in the document-score domain.

(3) Define a similarity measure $s(\underline{q}_i, \underline{q}_j)$ between queries i and j. In our case, the measure $s(\underline{q}_i, \underline{q}_j)$ is close to one when $\underline{q}_i \approx \underline{q}_j$ and is zero when the queries are not 'related'.

(4) Define 'goodness' measures for each cluster and for the overall set of clusters.

(5) Find the 'best' pair of clusters.

(6) Merge the 'best' pair of clusters into a single cluster.

(7) Repeat Steps 5 and 6 until a stop criterion is met. In our case the stop criterion was the final number of clusters.

In what follows, different query domains and our new agglomerative clustering algorithm will be explained in more detail. An evaluation of the clusters generated by the algorithm as well as comparisons with the clusters generated by the 2D-SOM expert users on the HP database will be presented.

6.2.1 Clustering Based on Original Query Domain

The original query domain representation does not take into account either the semantics of the original request or the relative position of the terms in the request (which could be expressed in natural language). Thus, the discrimination of queries based on the context is very difficult. Although this is an apparent weakness many TRS use this representation for retrieval purposes (not for clustering) due to its simplicity, ease of implementation, and the good results it has shown until today.

Any cluster algorithm needs some type of distance measure between two queries. Let $\underline{q}_i$ and $\underline{q}_j$ be the queries i and j in the log file of queries, then a similarity measure $s(\underline{q}_i, \underline{q}_j)$ is used for comparing the closeness of two queries. The frequently used similarity measure is the cosine of the angle between the two vectors i.e. $s(\underline{q}_i, \underline{q}_j) = cos(\alpha_{ij}) = \underline{q}_i^T \underline{q}_j / (||\underline{q}_i|| ||\underline{q}_j||)$.

The cohesion $c(\underline{q}_i, \underline{q}_j)$ between queries $\underline{q}_i$ and $\underline{q}_j$ is defined as the number of elements they have in common. The cohesion of the set of queries is then the expected value $E[c(\underline{q}_i, \underline{q}_j)]$ or average cohesion over this set. Clearly, when all the queries in a set $Q = \{\underline{q}_1, \underline{q}_2, \ldots, \underline{q}_n\}$ are represented in the original query domain, the cohesion between query i and query j is given by

$$c(\underline{q}_i, \underline{q}_j) = \underline{q}_i{}^T \underline{q}_j$$

For a set Q of queries, let $y(x)$ be the number of different pairs of queries that have x elements in common. That is

$$y(x) = ||\{(\underline{u}, \underline{v})|\ c(\underline{u}, \underline{v}) = x;\ \underline{u} \neq \underline{v}\ and\ \underline{u}, \underline{v} \in Q\}|| \tag{6.1}$$

where ||.|| stands for the cardinality operation.

Before clustering a set of queries it is recommended to determine their cohesion. If all the queries show high cohesion because they share the same terms then they do not convey anything new. In contrast, if in the set there are not two queries that share at least one term then each query should form a cluster and nothing new is learnt from the clustering. Even though there is no need for clustering in the latter case, some correlation between terms of the same query can be obtained.

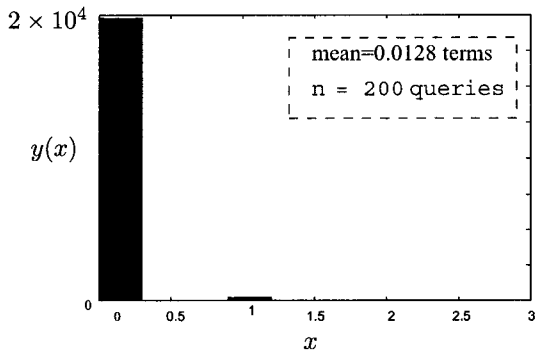


Figure 6.1: $y(x)$: number of different query-query pairs that have x elements in common.

Figure 6.1 shows the number $y(x)$ of different query pairs that contain x terms in common for the set of queries that corresponds to the most often submitted queries 200 queries provided by HP. For this set, the maximum number query-query combinations is 20,000, therefore $y(x) < 20000$, for any x. As can be seen in the plot in Figure 6.1, most queries share 0 terms. The mean value for the cohesion in the set of queries is 0.0128 terms. This value indicates that if we randomly choose two queries then they share on average 0.0128 terms. The value is rather low, so two original queries share very little information.

By clustering the queries in the original domain, the correlations between terms will somehow reflect the preference of the external users for choosing them. The clustering in this domain does not incorporate the high level expertise from either expert

users or our retrieval system that might be fine-tuned via relevance feedback learning (see Chapter 5). This shortcoming as well as the difficulty of finding some cross-related information between original queries led us to pursue other representation domains that are discussed next.

6.2.2 Clustering Based on Query Weight Domain

In this section, we show how the weights learned by the query mapping mechanism can be used for document term associations. For testing, the suggested document terms are the ones that were previously queried. These queries are stored in a query log in a database. To test the document term association, we have used 200 most frequently submitted queries taken from this query log. Now, when a new query is submitted, the query terms are evaluated against these 200 most frequently submitted queries and those queries that are more related (in concept) to the submitted query are retrieved and suggested to the user for query refinement. The selection of the related terms is based upon determining the amount of match between the weight vectors of the query terms, extracted from the first layer of the connectionist network, and those associated with the queries in the log file. If the match is above some pre-specified threshold then they are selected and suggested to the user for query refinement. The weight vectors captured by the network for single-term queries have to be mean corrected and normalized prior to the matching process. Consequently, term-matching via dot product operation corresponds to finding the cosine of the angle between the two weight vectors.

In our testing, ten different queries are submitted and for each query, its match in the query weight domain with those 200 pre-selected queries are evaluated. Queries with matching above 0.4 are selected and suggested to the user. The suggested query terms are then evaluated by the expert users for their relevance to the submitted query terms. Our experimental results revealed that on average 84.4% of the suggested

terms are indeed relevant to the query term. This shows the usefulness of first layer mapping weights of the network in Figure 3.1 for term association based upon their captured concept. It should be noted that this approach of query refinement using term association, which is one of by-products of our proposed system MRTRS, is very fast and amenable for real-time implementation. The results of query refinement are shown for two user submitted queries in the Tables 6.1 and 6.2. Columns 1-2 in these tables show the suggested terms and their corresponding match index, respectively. The terms on the first row of these tables, which have a match value of 1, are the actual user-submitted query terms while the rest are the suggested ones. Clearly, one can attest to the similarity in their concept and relevance to the original submitted queries. These results point to this interesting observation that the mapping matrix captured during the initial model reference training of the MRTRS and embedded in the connectionist network indeed contain useful information for query association and clustering applications. However, there are some clusters that contain terms with no similar information. Tables 6.3 and 6.4 show the results of such cases where most of the suggested terms are not related to the query term. In Table 6.3, only 2 out of the 9 suggested terms are relevant to the query term 'LINUX' and in Table 6.4, none of the suggested terms are related to the query term 'TROI'. Hence, in the next section, we developed a different clustering method based on the document domain idea.

6.2.3 Clustering Based on Document Score Domain

This domain lies in the space of the scores of the retrieved documents. Here, $\underline{r}_i = [r_{i1}, r_{i2}, \ldots, r_{iN}]^T$ is the normalized vector of document retrieved scores when query i is submitted. To find the cohesion between vectors in this domain and to compute the average cohesion for a set of queries we follow two steps: (1) for each vector $\underline{r}_i$ form its binary version $\underline{p}_i = [p_{i1}, p_{i2}, \ldots, p_{iM}]^T$, where $p_{im} = 1$ if $r_{im} > 0$ and is zero if $r_{im} \leq 0$, and (2) the cohesion between binary vectors $\underline{p}_i$ and $\underline{p}_j$ is $c(\underline{p}_i, \underline{p}_j)$ which

Table 6.1: Term association for the query 'PCLXL ERROR'

Associated relevant queries	Matching value
PCLXL ERROR	1
ERROR	0.8325
79 ERROR	0.6520
ERROR CODE	0.6288
NUMER ERROR	0.5949
SEVER ERROR	0.4833
PCL XL ERROR	0.4444
SPOOL 32 ERROR	0.4331

Table 6.2: Term association for the query 'USB CHIPSET'

Associated relevant queries	Matching value
USB CHIPSET	1
USB CHIPSET ISSU	0.9165
LJ1000 USB CHIPSET ISSU	0.9043
LJ1200 USB CHIPSET ISSU	0.9041
USB	0.7965
CHIPSET	0.6242
USB TROUBLESHOOT	0.4331
IPD	0.4124

Table 6.3: Term association for the query 'LINUX'

Associated relevant queries	Matching value
LINUX	1
IPD	0.5474
BPL07282	0.5378
SPOOLSVEX	0.5226
CITRIX	0.5079
BPL90092	0.5028
BPL10582	0.4728
CPLOCK	0.4681
QUICKBOOK	0.4613
DOA	0.4522

Table 6.4: Term association for the query 'TROI'

Associated relevant queries	Matching value
TROI	1
IPD	0.6540
BPL07282	0.6410
SPOOLSVEX	0.6155
BPL90092	0.6078
CITRIX	0.5865
BPL10582	0.5739
CPLOCK	0.5678
8993	0.5609
QUICKBOOK	0.5429

gives the number of documents in common for any pair (i, j) of submitted queries.

Figure 6.2 shows the number $y(x)$ of number of vector-vector pairs that contain x documents (elements) in common for the set of 200 most commonly used queries. These queries were submitted to the retrieval system and mapped to a binary output (document score) vector as described earlier. As can be seen, the queries in this domain indeed share many documents. The mean value for the cohesion in this case is 200 elements. This value indicates that if we randomly choose two queries from

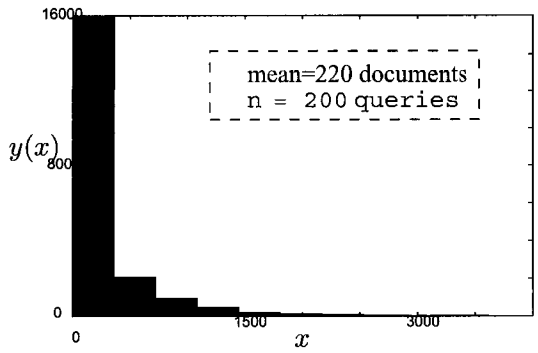


Figure 6.2: Number of different query-query pairs in the document score domain versus the number of elements in common.

the set then they share 200 elements on average. This value is better now than that found before using the original query domain in Figure 6.1.

6.3 Agglomerative Clustering Algorithm

Starting with one query per cluster, our agglomerative clustering algorithm merges, at each iteration, the 'best' two clusters according to the 'overall cohesion' computed on the resulting set of clusters until a predetermined number of clusters is reached. Let us define the cohesion c of cluster $P = \{\underline{p}_1, \underline{p}_2, \ldots, \underline{p}_K\}$ as the average 'similarity' of the normalized vectors in this set by

$$c(P) = \begin{cases} \dfrac{\sum\limits_{\underline{p}_i, \underline{p}_j \in P, j>i} \underline{p}_i^T \underline{p}_j}{N(P)} & |P| > 1 \\ 0 & |P| \leq 1 \end{cases} \tag{6.2}$$

where $N(P)$ stands for the cardinality of set P, $\underline{p}_i^T \underline{p}_j$ gives our similarity measure between the i^{th} and the j^{th} associated binary vectors, and $N(P)$ is the number of combinations or number of inter-relations when selecting two different vectors from P.

Now, using the average cohesion for each cluster as well as its number of inter-relations, let us define the overall cohesion OC of a set of H clusters $P_1, P_2, \ldots, P_H$

by

$$E[c(P)] = OC = \frac{\sum_{i=1}^{H} c(P_i)N(P_i)}{\sum_{i=1}^{H} N(P_i)} \tag{6.3}$$

When two prospect clusters k and l, are merged the overall cohesion $OC_{k,l}$

$$OC_{k,l} = \frac{\sum_{n=1,n\neq k,n\neq l}^{H} c(P_n)N(P_n) + c(P_{kl})N(P_{kl})}{\sum_{n=1,n\neq k,n\neq l}^{H} N(P_n) + N(P_{kl})} \tag{6.4}$$

will change as a result of this process. Here, P_{kl} is the resulting cluster after the union of clusters P_k and P_l, i.e. $P_{kl} = P_k \bigcup P_l$. The goal of our clustering algorithm is to merge the two clusters that would yield the maximum value of the new overall cohesion. This is expressed as

$$i, j = arg \max_{k,l} OC_{k,l} \tag{6.5}$$

The steps of this algorithm are listed below.

Agglomerative Algorithm

Let $\{\underline{p}_1, \underline{p}_2, \ldots, \underline{p}_L\}$ be the initial set of L normalized output vectors and F be the desired number of clusters. Then, we follow the following steps.

1. Initialize L clusters and set $P_i = \{\underline{p}_i\}$ for $i = 1, 2, \ldots, L$, i.e., each cluster contains only one normalized vector.
2. Set $n \leftarrow L$.
3. Using the cohesion measure in (6.4) find the best two clusters according to (6.5).
4. Merge clusters i and j into one cluster.
5. Set $n \leftarrow n - 1$.
6. Repeat steps 3 through 5 until $n = F$, ie., the number of clusters n reaches the pre-specified value F.

6.3.1 Experimental Results of Agglomerative Algorithm and Comparison with Expert Users Clustering

The agglomerate algorithm is used to generate thirty (F=30) clusters from a set of the 200 most commonly used queries. Each query is first submitted to the retrieval system and from the list of the scored documents, the normalized output vector in the document domain was formed using the procedure explained in the previous section. However, for each vector only those scores that were above a certain threshold (a factor of the score of the top document) were considered, the score of the rest were set to zero. The overall cohesion for the set of clusters was used to measure the performance of our algorithm and to compare it against the 2D-SOM algorithm [33] as well as the set of clusters generated by expert users at HP.

Figure 6.3 (a) shows the plot of cohesion for each of the 30 clusters produced with the agglomerative algorithm. In this figure, P_i represents the set of normalized score vectors for cluster i and $c(P_i)$ is the cohesion measure for that set. Also shown is the overall cohesion for the set of clusters. As can be seen, the overall cohesion is 0.33, which is very good comparing to the cohesion measure found for the sets of clusters provided by the expert users (OC=0.23) or by the 2D-SOM algorithm (Figures 6.4 (b) and (c)). This implies that the vectors in a cluster formed using our agglomerative algorithm share more information (they are more similar) that those vectors in the clusters formed by expert users or the 2D-SOM algorithm.

Another possible indicator that the algorithm produces a 'good' set of clusters could be computed by using the distribution of the vectors within the clusters; a 'good' algorithm should balance, in some way, the number of vectors per cluster, i.e., producing a set of clusters with small variance according to number of vectors per cluster. This implies that there could be a relationship between our criterion for merging two clusters based on our cohesion measure and another criterion based on the number of vectors per cluster. Figures 6.4 (a), (b), and (c) show the number

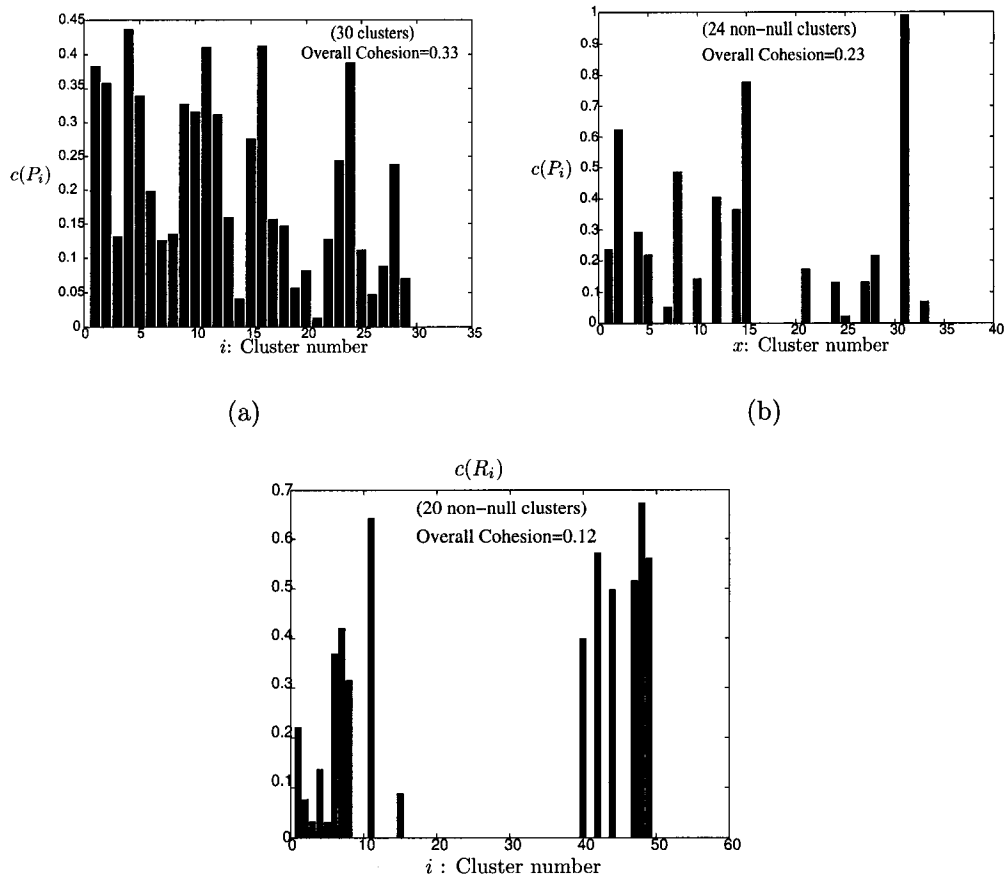


Figure 6.3: Cohesion measure per cluster using normalized outputs for the clusters generated by (a) agglomerative algorithm, (b) expert users, and (c) 2D-SOM algorithm.

of vectors for each of the clusters generated by the agglomerative algorithm using the document-score domain, the expert users, and the 2D-SOM algorithm. As can be seen, the variance of the number of vectors per cluster for the agglomerative algorithm (var = 16) is smaller than those generated by using the clusters provided by the expert users (var= 242) or the 2D-SOM algorithm (var=200). These results clearly indicates a good performance of the proposed algorithm for query clustering.

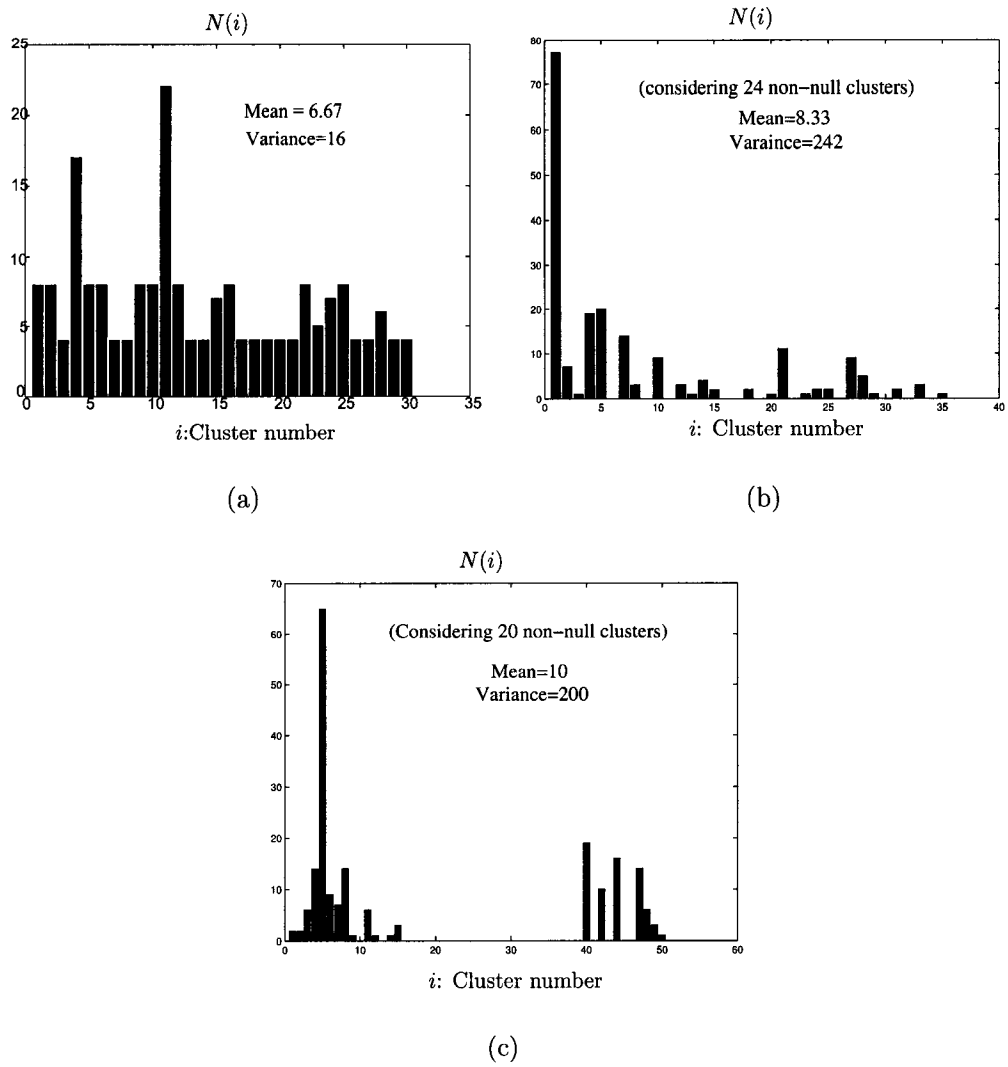


Figure 6.4: Number of queries per cluster using normalized output vectors (a) agglomerative algorithm, (b) expert users, and (c) 2D-SOM algorithm

6.4 Conclusions

In this chapter we developed an algorithm for query clustering based upon both a cohesion measure between two transformed vectors and a overall cohesion measure for a set of queries. Original queries are transformed to using the first layer of our connectionist network system to the optimal mapped query with weights that can be used for clustering. Alternatively, the last layer results that correspond to the document-score domain could be used. These transformed vectors are then clustered

using our agglomerative clustering algorithm and the created clusters are then analyzed to provide further information to the users. Thus, when a user submit a query the our MRTRS system suggests additional words that can be used for the user to modify the query.

The optimal query representation proposed in this work captures the documents' content not only from the reference model TRS system but also from expert users via relevance feedback. This in turn help the clustering process because the weight or the document-score queries domain captures more information that the original-query domain. Different query vector domains were also proposed, mainly the query weight vector in the mapping layer of the connectionist network and the normalized output (document score) vector. It was shown that the proposed agglomerative algorithm provides clusters with much better cohesion measure than those generated by the expert users or the 2D-SOM algorithm when either the weight or the document score domain was used. The effectiveness of the proposed algorithms has been successfully demonstrated on a text database that contains HP LaserJet and other product families.

A study to measure the number of terms between a pair of queries showed that two submitted queries share on average 0.0128 terms. The study was based on a set of the most frequently submitted 200 queries related to HP-LaserJet-product family and submitted by expert users at HP-Corporation in a six-month period. Given that the number of terms shared by two submitted queries is small and the dimension of the query vectors is very large a standard clustering algorithm based upon these queries could lead to relatively poor results. Thus, the transformed queries could provide more information for clustering purposes as they have more terms in common.

The cohesion measure to evaluate the closeness of a set of queries in a cluster is introduced. A value of '1' indicates that all queries share the same terms and a value of '0' indicates that no query in the set share any term with another query.

The cohesion measure were 0.33, 0.23, and 0.12 for the clusters generated with the agglomerative clustering algorithm, the expert users, and the 2D-SOM algorithm, respectively. This implies that the clusters generated by the proposed agglomerative clustering algorithm are more compact than those generated by the expert users or the 2D-SOM algorithm. This leads to the discover of some similarity between queries that produce approximately the same output in some specific domain.

PART II: IMAGE RETRIEVAL

CHAPTER 7

A MODEL REFERENCE IMAGE RETRIEVAL FRAMEWORK

7.1 Introduction

The focus of most content-based image retrieval (CBIR) [41, 92, 93] systems is to apply robust search, content matching and retrieval to deal effectively and consistently with a large volume of information. These systems usually provide mechanisms that allow the user to navigate through the listed images. A search process typically culminates at a list of images from which the user identifies the most relevant ones after navigating or browsing in the order of the images' "retrieval status values" or relative scores. In these systems, the ambiguity of the initial query image makes it difficult to guarantee a retrieval with successful outcome. A methodology that allows for an efficient navigation through the images and automatic incorporation of user's expertise to influence the suggested solutions is essential.

In this chapter, we present two new image retrieval systems based upon the theory of model-reference adaptive control. They are referred to as multiple-regulator and single-regulator model reference image retrieval systems. The systems can incorporate users' expertise in an online fashion or model-reference information in batch mode. The backbone of the systems is the regulator, which implements query modification by mapping the original query into an 'optimal' one using linear transformations in

a high dimensional space. The multiple-regulator system is structurally adaptable under user's feedback and captures the information in only a few parameters, making it suitable for on-line implementation. In the single-regulator, relevance feedback learning is incorporated into a single mapping matrix, in contrast to the multiple regulator case, which contains several mapping matrices. This leads to a simpler structure and algorithms but it requires more parameters to be identified. For both systems, relevance feedback learning is cast in a constraint optimization framework. Subsequent sections present overviews of the proposed image retrieval systems; leaving the details to the next two chapters.

7.1.1 Definitions

Some important definitions used throughout this and the next two chapters will be presented next.

- Database: A set of images (generally features vectors) that can be searched and retrieval from our image retrieval systems. Database typically does not contain class information of the images. A database containing N images is represented by matrix $X = [\underline{x}_1 \underline{x}_2 \cdots \underline{x}_N]$, where $\underline{x}_j$ is the j^{th} feature vector. Such a database is used for the start up process of our system.

- Model-Reference Database: The model-reference database is formed with the database together with class label information for every constituent image.

- Log-file: Contains information of query images and their corresponding scores generated by the expert users when applying relevance feedback learning. Both the query image and the listed images belong to the database.

- Training query: A query image that is in the database.

- Testing query: A novel query image that is not in the database.

The organization of the chapter is as follows. Section 7.2 presents a general overview of the architecture of multiple-regulator model-reference image retrieval system. Section 7.3 describes the constituent parts of our single-regulator image retrieval system as well as the mechanisms to incorporate users' feedback. Section 7.4 presents the conclusions and observations on the proposed systems.

7.2 Multiple Regulator Image Retrieval System

The idea behind learning in model-reference adaptive control systems (MRAS) [57,94] is adopted here to develop a new model-reference image retrieval system (MRIRS) referred to as Multiple Regulator Image Retrieval System (MRIRS). The system architecture is shown in Figure 7.1. The retrieval system contains four components: a 'plant', representing a non-adaptable image retrieval system; a series of regulators, each represented by a mapping matrix; a model reference image retrieval system, represented by a model-reference database; an adjustment mechanism in charge of setting or updating the parameters of the regulator in order to follow the model reference or the users' requirements.

The proposed MRIRS as shown in Figure 7.1 basically operates in three modes (i) search and retrieval, (ii) model reference learning, and (iii) relevance feedback learning. These modes along with the initial start up will be briefly described in the subsequent subsections.

7.2.1 Search and Retrieval Mode

The search and retrieval mode is the most basic of the three. Nonetheless, it deserves special attention because search and retrieval operations should be performed in almost real-time. After an image query $\underline{q}_k$ is submitted it is mapped nonlinearly into vector $\underline{q}'_k = \Phi(\underline{q}_k)$ using a particular kernel producing nonlinear function $\Phi(\cdot)$. The formed vector is then mapped into N different 'optimal' queries, $\{\hat{\underline{q}}_{jk}\}_{j=1}^{N}$, using the mapping matrices of the regulators, hence $\hat{\underline{q}}_{jk} = A_j \underline{q}'_k$, $j \in [1, N]$. The next step is to

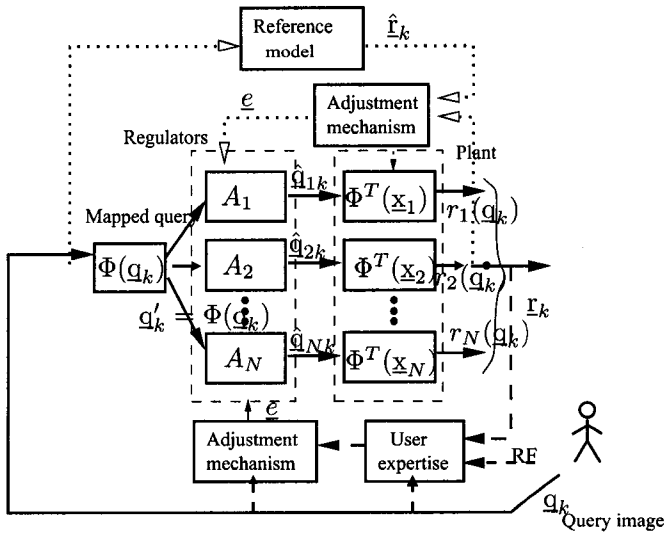


Figure 7.1: Multiple Regulator Image Retrieval System (MRIRS)

obtain inner products between each 'optimal' query and its associated mapped image in the database. These products will be used as retrieval scores, i.e.

$$r_j(\underline{q}_k) = \Phi^T(\underline{x}_j)\hat{\underline{q}}_{jk} = \Phi^T(\underline{x}_j)A_j\Phi(\underline{q}_k) \tag{7.1}$$

is the score for the j^{th} image for the original query $\underline{q}_k$. The search and retrieval process ends when the most relevant images are listed in order of their scores. The solid lines in Figure 7.1 show the paths of the search and retrieval process from the moment the user submits the query until the creation of the score vector.

7.2.2 Initial Start Up

Any adaptable image retrieval system should be able to adequately initialize its internal parameters. For the multiple regulator system, the goal of the initial start up is to adequately select each one of the mapping matrices and to compute the underlying operations with minimum computational resources. Consider the multiple regulator retrieval system shown in Figure 7.1 and a database with N images, the goal of the initial start up in this system is to find matrices $A_1, A_2, \ldots, A_N$ for the N regulators (i.e. one regulator for every image in the database) and an appropriate mapping

function $\Phi(\cdot)$ so that the initial scoring function between an image in the database and a query is given by the simple directional cosine in the high dimensional space. Therefore, we require that

$$\Phi^T(\underline{x}_j)A_j\Phi(\underline{q}_k) = \Phi^T(\underline{x}_j)\Phi(\underline{x}_j) \tag{7.2}$$

holds for any query $\underline{q}_k$.

Clearly, one choice for A_j that complies with (7.2) is $A_j = I$, and another is the projection matrix, i.e. $A_j = \Phi(\underline{x}_j)\Phi^T(\underline{x}_j)/k(\underline{x}_j, \underline{x}_j)$, where $k(\cdot,\cdot)$ is an appropriated kernel function. Choosing the projection matrix as the mapping matrix has the benefit that its trace (which is 1) is less than that of the identity matrix (This is discussed in more details in Chapter 9). Furthermore, using the projection matrix, most of our learning algorithms for image retrieval, including the initial start up process, find their counterparts in the text retrieval area, already presented in previous chapters.

7.2.3 Model-Reference Learning Mode

Model-reference learning exploits the information from an external image retrieval system or in its absence from the class label information for the images in the database. The first step when applying model-reference learning is to form a model-reference training database which contains query-score list relationships using the images of the database and their known class labels. To form this set every image in the database is submitted as a query and the 'desired' scores of the relevant retrieved images are computed. If the score of a relevant image is below certain threshold then its score is set to the threshold and this query-score relationship is stored in the model-reference training database. Having found the model-reference set the information needed for each regulator is separated from the rest and used to set up its parameters. In absence of class information, the parameters may be updated using relevance feedback learning, which will be discussed in the next section. The main benefit of the model-reference learning mode in the multiple regulator IRS is that the number of

parameters to be updated are only a fraction of those in the single regulator IRS. The dashed lines in the feedforward path of retrieval system in Figure 7.1 show the flow of information and the subsystems involved in the model-reference learning. The details of this learning mode are presented in Chapter 9.

7.2.4 Relevance Feedback Learning

Relevance feedback learning is activated when the user selects one or more relevant images from the retrieved list and assigns new positions and possibly scores to them. If the user only assigns positions then the system computes the desired scores internally for the relevant images. The desired score vector and the output score vector are used by the adjustment mechanism to form an error vector used to update the parameters of the regulators in such a way that if the same query image is encountered again on the system will respond to it more accurately. During relevance feedback learning the mapping matrices of the regulators are continuously accumulating users' knowledge into their elements. However, not all matrices are incorporating information at the same time but only those associated to the most relevant images selected for a given query. Because the relevancy information is only influencing certain mapping matrices in the overall retrieval system, the incorporation of information is highly localized, contrary to the single regulator system.

The process goes like this. At time t_0 the j^{th} regulator is assume to have already captured the information of 'L' queries and its adjustable parameter vector at that moment is $\underline{w}_j^{(0)}$. At time t_0 a new query image arrives and the user selects, via relevance feedback, the image associated with this regulator as the most relevant one. In order to incorporate the new information the regulator modifies its parameters by updating $\underline{w}_j^{(0)}$ using $\underline{w}_j^{(1)} = \underline{w}_j^{(0)} + \Delta\underline{w}_j$, where $\Delta\underline{w}_j$ is obtained by solving a constrained optimization problem that relies on the users' relevance feedback. The details of the learning rule derivations are provided in Chapter 9. We shall see that what drives the

updating equation for $\underline{w}_j$ is the difference between the desired scores and the actual output of the retrieval system for the old stored queries and the newly submitted one.

Clearly, the apparent drawback for the proposed multiple regulator image retrieval system is its structural complexity, especially for large image databases. However, in spite of its complexity the updating process requires only the updating of parameters of those regulators that participate in the learning via the relevance feedback.

7.3 Single Regulator Image Retrieval System

The single regulator image retrieval system referred to as (SRIRS) is inspired from the model reference text retrieval system described in previous chapters. The proposed SRIRS also contains four components and its structure is shown in Figure 7.2. The structure is represented in a possible high dimensional space where both images and queries are mapped into this space using some kernel producing function. The components are: (i) a plant represented by a matrix with columns that are mapped feature vectors (in the high dimensional space) of the images in the database, (ii) a single regulator represented by a single mapping matrix A with adaptable parameters, (iii) a model reference image retrieval system, and (iv) an adjustment mechanism in charge of imposing the learning rules. The proposed SRIRS can perform ordinary search and retrieval operations and address the problem of learning from users via a log-file if they are off-line or directly from them if they are online. These are briefly introduced in the following subsections.

7.3.1 Search and Retrieval Mode

The search and retrieval process goes like this. When a query $\underline{q}_k$ is submitted, it is transformed into query $\underline{q}' = \Phi(\underline{q})$ using a particular kernel producing function $\Phi(\cdot)$. The formed vector $\underline{q}'$ is then mapped into a single 'optimal' query, $\hat{\underline{q}}$, using mapping matrix A of the regulator,i.e. $\hat{\underline{q}} = A\underline{q}'$. Then, this optimal query, which might be a feature vector in a high dimensional space, is used to effect each one of the images in

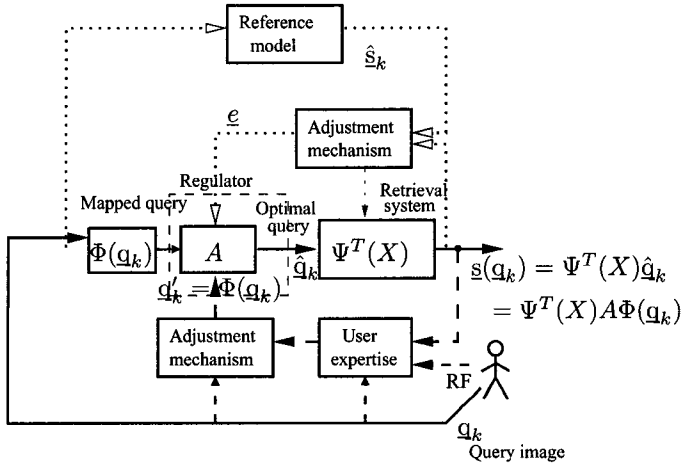


Figure 7.2: Single Regulator Image Retrieval System (SRIRS)

the plant matrix, yielding a score vector. The score vector $\underline{s}$ is simply computed by a dot product, i.e. $\underline{s} = \Psi^T(X)\hat{\underline{q}}$, where matrix X contains as its rows the images in the database. The solid lines in the feedforward path in Figure 7.2 show the paths of the search and retrieval process.

7.3.2 Initial Start Up

In the proposed SRIRS system the score vector $\underline{s}(\underline{q}_k)$ for query $\underline{q}_k$ is given by

$$\underline{s}(\underline{q}_k) = \Psi^T(X)A\Phi(\underline{q}_k). \tag{7.3}$$

If we impose the constraint that $\underline{s}(\underline{q}_k) = \Psi^T(X)\Phi(\underline{q}_k)$ for any given query $\underline{q}_k$ then we might want to choose one of these two choices: $A = I$ or $A = \Psi(X)[\Psi^T(X)\Psi(X)]^{-1}$ $\Psi^T(X)$. The latter choice is better as the trace of the matrix is N, rather than a possibly large value for the former case because of large dimension of A corresponding to high dimensional feature space. Furthermore, if we consider a linear mapping, i.e. $\Psi(X) = X$ the projection matrix becomes $P_X = X[X^TX]^{-1}X^T$, which resembles the projection matrix used for text retrieval but here the feature vectors are images

instead of documents. Further details for the initial start up are given in Chapter 10.

7.3.3 Model-Reference Learning Mode

Over time a set of input-output (query-desired score list) relationships can be captured in the log-file. This information can be used to set up the parameters of the regulator and thus making it respond better to new concepts that share some similarity with already stored concepts. The problem of model-reference learning can also be cast in a constrained optimization framework when for a set of $(\underline{q}_i, \underline{\hat{s}}(\underline{q}_i))$, $i \in [1\ L]$, training pairs. The details are given in Section 10.3 in Chapter 10.

7.3.4 Relevance feedback Learning Mode

Preliminary theoretical and experimental results indicate that the single regulator retrieval system is not adequate to conduct relevance feedback learning in an online real-time fashion, which is primordial requirement for any practical and interactive image retrieval system. Further research is required to improve the performance of this system.

7.4 Conclusions

In this chapter, two different MRIRS retrieval systems have been proposed, namely the multiple regulator and the single-regulator retrieval systems. In the former case each regulator is assigned to a particular image of the database. Therefore, the acquisition of concepts via relevance feedback is focused towards a particular regulator, which can be seen as a memory repository. In contrast, in the single regulator the concepts are captured in a single-matrix, making the information visible to all images as they share the same regulator.

Although the multiple regulator system is structurally more complicated, its computational requirements for storing relevant information are less than those of the

single regulator; due in part to the less number of free parameters that require updating. The initial start up of the two systems corresponds to initializing the regulator mapping matrices to the projection matrices. This projection matrices are generally expressed in a high dimensional space. However, in our kernel-based retrieval systems like in most kernel-based mechanisms the operations are not carried out explicitly in the high dimensional space but rather in a low dimensional space where many operations include dot products that can easily be computed using the kernel-based functions.

The model-reference and relevance feedback learning rules used for the adjustment mechanism for the two systems are derived (see Chapters 9 and 10) from various constrained optimization problems. Although the form of the cost function for both systems is somewhat similar, the solution of the multiple regulator case leads to a much simpler algorithm when compared with that of the single regulator. In the multiple regulator system, relevance feedback learning can easily be implemented, as the relevancy information only affects only a few regulators and their parameters, while in the single regulator case the information affects all the free parameters in the regulator which are of order of $O(N^2)$, hence making the single-regulator system impractical for realistic applications. Thus, new research in this area is needed to develop efficient learning algorithms for this structurally simple system.

CHAPTER 8

DATA DESCRIPTION AND FEATURE EXTRACTION

8.1 Introduction

The major challenge in any target classification system is in extracting pertinent sets of shape, texture, and color dependent features with high discriminatory ability from the segmented images. This process not only leads to a set of salient features but also reduces the dimension of the data to a manageable size. A set of robust shape-dependent features that are invariant to rotation, translation and scaling is needed to classify the objects based upon their 2-D or 3-D shape characteristics. In this work, we studied different shape-dependent feature extraction schemes for rotation, translation and scaling invariant pattern identification.

Some results on the STIL imagery and on the handwritten digits are presented in this chapter to discuss the properties and shortcomings of these feature extraction algorithms.

8.2 Texture and Shape-Dependent Features

Zernike moments [95, 96] were found to provide certain benefits for our particular target identification problem such as immunity to additive noise and to some modest amount of distortion in the segmented images. However, In order to identify an

object with high degree of confidence it is almost imperative to use multiple sets of independent features.

Target identification can be greatly enhanced, especially for STIL imagery data for which range as well as contrast maps are available, by using the texture information of the segmented objects. This information will allow better discrimination based upon target's surface and reflectivity characteristics. Textural features, among the other features, are often exploited by humans to describe different characteristics such as smoothness, fineness and coarseness associated with an object in the image. They reflect the local spatial distribution property in a small region on the object. The spectral and textural features are widely used in target recognition problems particularly in multi-spectral and hyper-spectral imagery data. To improve the identification of the targets in the STIL imagery, we used the Gray Level Co-occurrence Matrices (GLCM) method [97] that computes several statistical/textural features namely contrast, correlation, entropy, and homogeneity.

8.2.1 Shape-Dependent Feature Extraction using Zernike Moments

Various feature extraction schemes are available that can be used to extract shape-dependent features for a wide variety of pattern recognition problems. However, moment-based schemes [96], [98–101] are among the most widely used methods as they provide translation, rotation and scaling invariant features ideal for 2-D as well as 3-D pattern recognition applications. In [99], a comparison was made among several types of moments including regular moments, Legendre moments, Zernike moments, and complex moments. These methods were compared in terms of their image representation ability, noise sensitivity, and information redundancy on several character recognition examples. Owing to the fact that the regular moments do not provide an orthogonal representation, the extracted features using this scheme lack

optimality in representation. This is in contrast to the orthogonal moments, e.g. Legnedre and Zernike moments. The experiments conducted in this reference indicated that the classification results of the Zernike moments are substantially less sensitive to additive noise effects in the images when compared to the other types. In [100], a similar study was carried out where the regular moments and Zernike moments were used for feature extraction and a back-propagation neural network (BPNN) was employed as a classifier. The system was tested for classifying 26 uppercase characters (A to Z) in the English alphabets. The silhouettes were allowed to have varying scale, translation and orientation forming 24 sets of images. In addition, random noise with varying SNR from 5 to 50 dB was added to the patterns. The simulation results once again showed the noise immunity of the Zernike moments particularly when used in conjunction with a BPNN classifier. In another study [96] Zernike moments were used for recognition and pose estimation of 3-D objects from the 2-D perspective views. The scheme utilizes multiple BPNN's with different parameters and structures. The decisions of these networks were fused together using a majority voting scheme. It was observed that combining the decisions of these parallel networks can minimize the occurrence of erroneous decisions. Due to the use of Zernike moments the performance was invariant to viewing angle, location and orientation of the objects in the image. The effectiveness of the system was demonstrated on several clean and noisy patterns of military ground targets. Finally, the two pose parameters, namely elevation and aspect angles, were estimated using a two-stage neural network structure. In [101], a pattern classification scheme for classifying buried landmines of wood and nylon compositions from microwave imagery data. Two-dimensional (2-D) Karhonen Loeve (KL) transform and Zernike moments were used to extract energy and shape-dependent features of the segmented landmine regions. A neural network was then trained to discriminate the targets from the non-target anomalies. The comparison of the results indicated that the Zernike moments gave much better discrimination

of wooden type mines that are generally very difficult to identify due to their weak response. This is due to the property that the dielectric constant of wood is closer to that of soil than the nylon. Additionally, it was observed that the uncorrelated property of these feature extraction schemes substantially improved the training of the neural network classifier. Due to all the useful properties of the Zernike moments we have adopted this method for shape-dependent feature extraction.

Zernike moments are obtained using a complete set of complex polynomials defined in the interior of the unit circle. For an image $f(x,y)$, the Zernike moments of order n with repetition m where $|m| \leq n$ and $n - |m|$ constrained to an even number, are given by

$$A_{nm} = \frac{n+1}{\pi} \sum_x \sum_y f(x,y) V^*_{nm}(\rho, \theta) \tag{8.1}$$

where $x^2 + y^2 \leq 1$ i.e. confined to the interior of the unit circle, ρ is the length of a vector from the origin of the unit circle to (x,y) point, θ is the corresponding phase angle, and $V_{nm}(\rho,\theta)$ form a complete set of orthogonal complex polynomials over the unit circle. These are defined by

$$V_{nm}(\rho,\theta) = R_{nm}(\rho) e^{jm\theta} \tag{8.2}$$

where the radial polynomial $R_{nm}(\rho)$ is given by

$$R_{nm}(\rho) = \sum_{s=0}^{n-|m|/2} \frac{(-1)^s [(n-s)!] \rho^{n-2s}}{s!(\frac{n+|m|}{2} - s)!(\frac{n-|m|}{2} - s)!} \tag{8.3}$$

Note that we have $R_{n,-m}(\rho) = R_{nm}(\rho)$ and also $A^*_{nm} = A_{n,-m}$.The magnitude of the Zernike moments can be viewed as rotation invariant features. To make the moments translation invariant, the image is transformed to a new coordinate system by moving the origin to the centroid prior to the moment calculation. Then, the first-order moments of the new image becomes zero. To achieve scale invariance, the image is resized by changing its zeroth-order moment to a predetermined value. A general mapping of type $f(x,y) = g(\overline{x} + \frac{x}{a}, \overline{y} + \frac{y}{a})$ where $(\overline{x}, \overline{y})$ are coordinates of

the centroid of the original image $g(x, y)$ and a is a scaling parameter, is applied prior to computing the Zernike moments. The scale and translation normalization processes affect two of the Zernike features namely $|A_{00}|$ and $|A_{11}|$. However, $|A_{00}|$ will be the same for all the images and $|A_{11}|$ will be zero. As a result, these moments will not be included in the extracted feature set and the selected features start from the second-order moments. The great benefits of these moments are the orthogonal property which is considerably useful when these are used for object classification. In addition, it has been shown that the Zernike moments [96, 100] are more immune to noise and distortion than the regular moments.

8.2.2 Texture Feature Extraction using Co-Occurrence Matrices

Several textural feature extraction schemes have been studied for a wide variety of classification problems [97], [102–104]. Among the most widely used approaches are: the Gray Level Co-occurrence Matrices (GLCM)-based approach [97], Singular Value Decomposition (SVD) [102], Gabor filters [105] and wavelet transform [103, 104]. Gabor filtering [105] uses a bank of filters that nearly uniformly covers the spatial frequency range. A non-linear transformation is then applied to some of the selected filter outputs to compute energy-dependent features. These are then integrated using a clustering algorithm. One difficulty with the texture feature extraction is the effect of scaling on the features. This problem can be circumvented by using the multi-resolution characteristics of the wavelet transform. Wavelet packets [104] provide an image-dependent sub-band decomposition using a filter bank with a rich menu of orthonormal basis functions. Each sub-band extracts certain spatial-spectral information depending on the frequency range and content of the image. Generally, a criterion such as energy or entropy is selected and used to construct the best tree structure for the decomposition of a particular class of textures.

One of the defining qualities of texture is the spatial distribution of gray levels.

The use of statistical features is therefore is one of the earliest methods proposed in this context. Haralick [97] suggested the use of gray level co-occurrence matrices (GLCM) which have become one of the most well-known and widely used texture feature extraction methods. The GLCM method assumes that the texture information in an image is contained in the overall or average spatial relationships that gray levels have with each other. Spatial gray level co-occurrence estimates image properties related to the second order statistics that define the likelihood of observing a pair of gray values when they are connected by a certain displacement vector. When the entry values of these matrices are normalized they account for probability estimates (frequencies). Experiments performed in the areas of psychology and biology suggested that these spatial gray-level matrices may play an important role in human texture identification. Several comparative studies have indicated that these co-occurrence measures are superior to other basic textural measures in their classification performance. Next, a brief description of the GLCM method is presented. The reader is referred to [106] for more detailed description.

Let $I(x, y)$ denote an $N_1 \times N_2$ image with L gray levels. The size of gray-level co-occurrence matrix denoted by $P_{\mathbf{d}}$ for a displacement $\mathbf{d} = (d_x, d_y)$ is $L \times L$. The entry (i, j) of matrix $P_{\mathbf{d}}$ is the number of occurrences of the pair of gray levels i and j which have a displacement $\mathbf{d}$. This is given by

$$P_{\mathbf{d}}(i, j) = |\{(r, s), (t, v) : I(r, s) = i, I(t, v) = j\}|$$

where $(r, s), (t, v) \in N_1 \times N_2$, $(t, v) = (r + d_x, s + d_y)$, and $|.|$ stands for the cardinality of the set.

As an example, consider the following 4×4 image containing three different gray levels (L=3), i.e., 0,1 and 2:

$$I(x,y) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \end{bmatrix} \tag{8.4}$$

The 3×3 gray level co-occurrence matrix for this image for a displacement vector $\mathbf{d} = (1,0)$ is given by

$$P_{\mathbf{d}} = \begin{bmatrix} 4 & 0 & 2 \\ 2 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

For example, the entry (0,0) of $P_{\mathbf{d}}$ is 4 because there are four pixel pairs of $(0,0)$ which are offset by $(1,0)$ amount. Examples of $P_{\mathbf{d}}$ for other displacements are

$$\text{For } \mathbf{d} = (0,1) \Rightarrow \qquad P_{\mathbf{d}} = \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 0 \\ 2 & 0 & 2 \end{bmatrix}$$

and

$$\text{For } \mathbf{d} = (1,1) \Rightarrow \qquad P_{\mathbf{d}} = \begin{bmatrix} 3 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

The co-occurrence matrix reveals certain properties about the spatial distributions of the gray levels in the image. For example, if most entries in the co-occurrence matrix are concentred along the diagonal, then the texture is coarse with respect to the choice of displacement vector $\mathbf{d}$. Haralick [97] proposed a number of useful textural features that can be computed from the co-occurrence matrix. A small subset of these features namely contrast, homogeneity, energy or angular second moment, correlation,

and entropy are typically employed in most texture discrimination problems, but before extracting these set of features is recommendable to normalized each element of $P_{\mathbf{d}}$ by dividing it by the summation $\sum_{i,j} P_{\mathbf{d}}(i,j)$ of all the elements of matrix $P_{\mathbf{d}}$. Hereafter, we assume that $P_{\mathbf{d}}$ is a normalized matrix. Table 8.1 lists some of the formulae for calculating these features. Here μ_1 and μ_2 are the means and σ_1 and σ_2 are the standard deviations based upon $P_{\mathbf{d}}(i)$ and $P_{\mathbf{d}}(j)$ as density functions, respectively , where $P_{\mathbf{d}}(i)$ and $P_{\mathbf{d}}(j)$ are vectors of dimensions equal to the number of rows and columns of $P_{\mathbf{d}}$ and consist of the sum of the elements of columns and rows of $P_{\mathbf{d}}$. Thus, we can write

$$P_{\mathbf{d}}(i) = \sum_j P_{\mathbf{d}}(i,j) \qquad and \qquad P_{\mathbf{d}}(j) = \sum_i P_{\mathbf{d}}(i,j)$$

Table 8.1: Definition of some textural features of the co-occurrence matrix

Texture Feature	Formula
Contrast	$\sum_i \sum_j (i-j)^2 P_{\mathbf{d}}(i,j)$
Correlation	$\frac{\sum_i \sum_j (i-\mu_1)(j-\mu_2) P_{\mathbf{d}}(i,j)}{\sigma_1 \sigma_2}$
Entropy	$-\sum_i \sum_j P_{\mathbf{d}}(i,j) log P_{\mathbf{d}}(i,j)$
Homogeneity	$\sum_i \sum_j \frac{P_{\mathbf{d}}(i,j)}{1+\lvert i-j \rvert}$
Energy	$\sum_i \sum_j P_{\mathbf{d}}^2(i,j)$

In what follows, we describe these typical GLCM texture features used in most of the texture discrimination problems.

GLCM Textural Features

1. ***Contrast*** is a measure of local variation between pixels separated by displacement **d**. It represents the degree of spread of the gray levels at a fixed displacement **d**. A small value of contrast indicates high concentration of occurrences

on the main diagonal, which in turn represents a coarse texture. A large value of contrast describes the occurrences spread out of the main diagonal, hence representing a fine texture at the observed displacement **d**.

2. ***Correlation*** is a measure of gray level linear dependencies in the scene at pixel pairs separated by **d**.

3. ***Homogeneity*** is the inverse difference moment of the co-occurrence matrix and represents a measure of the amount of similarity in the scene. Homogeneity works opposite to that of contrast. That is, a coarse texture has a large value of local homogeneity than does a fine texture. Regions of low contrast are those of high local homogeneity and conversely regions of high contrast correspond to those of low local homogeneity.

4. ***Energy*** or angular second moment is a direct measure of textural homogeneity. This measure reaches its maximum value of 1 when the image is homogeneous either because the same gray level appears throughout the image (contrast equal to 0) or because it occurs at a displacement **d** whose $||\mathbf{d}|| \neq 1$, i.e., contrast value different than 0. The minimum value is reached when all the elements of $P^2_{\mathbf{d}}(i,j)$ are equal. In this case, the energy is close to 0.

5. ***Entropy*** is a measure if disorder in the scene. The entropy of the image is a descriptor of texture giving an indication of the randomness of the pixels, achieving high values when the pixels are highly different, i.e. the image is not uniform. Its minimum value is 0, which is reached when all the gray levels are the same (i.e. no disorder). For this value the energy reaches its maximum of 1. The maximum entropy value is found when $P_{\mathbf{d}}(i,j) = c$ for all possible values of i and j. In this case, the energy measure reaches its minimum. Thus, energy and entropy are inversely related. However, energy measure is preferred

because it has a limited range between 0 and 1, while the range of values for entropy is not limited; it could go from 0 to a relatively high value.

The GLCM method suffers from a number of deficiencies. Perhaps the main one is the large memory and computational requirement. Additionally, there is not a well-established method of selecting the displacement vector $\mathbf{d}$, though tests based on independence of two random variables can be useful to select a particular GLCM. Two of these are the χ^2-test and the Kolmogorov-Smirnov goodness of fit test [107]. The former was used for textural analysis by Zucker and Terzopoulos [108]. The hypothesis test proposed by Zucker is based on checking the independence of rows and columns of the co-occurrence matrix $P_{\mathbf{d}}(i,j)$. The purpose is to find the displacement $\mathbf{d}$ for which the likelihood of finding independence between rows and columns is as low as possible, i.e., rejecting the null hypotheses

$$H_0 : P_{\mathbf{d}}(i,j) = P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)$$

and accepting the true hypotheses $H_1 : P_{\mathbf{d}}(i,j) \neq P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)$. Rejecting the null hypotheses assures some degree of 'uncertainty' among features. In contrast, if the null hypotheses were accepted, then the correlation will be zero automatically and the knowledge of some other features could affect the knowledge of some others, e.g., if we knew μ_1, μ_2, σ_1 and σ_2 then the contrast would be $\sigma_1^2 + \sigma_2^2 + (\mu_1 - \mu_2)^2$. Now, the χ^2-test can be checked by forming

$$\tau_{\mathbf{d}} = \sum_{i,j} \frac{P_{\mathbf{d}}^2(i,j)}{P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)} - 1 \tag{8.5}$$

which can be used to accept or to reject the hypotheses. If the test reaches a value greater than a certain threshold, then it is rejected (null hypotheses). Since the value for the threshold is unknown, a better scheme is to find the displacement $\mathbf{d}$ which yields the maximum value of the right hand side in (8.5). In this way, there is a higher

possibility of failing the test for this displacement than for others, i.e., assuring some dependency between rows and columns of $P_{\mathbf{d}}(i,j)$.

This test is sensitive to the gray level resolution. The optimal choice of the displacement $\mathbf{d}$ can be found using

$$\mathbf{d}_{opt} = \underset{\mathbf{d}}{arg\ max}\ \tau_{\mathbf{d}}$$

Let us define the operator $E_{\mathbf{d}}[.]$ over a function $f(i,j)$ by

$$E_{\mathbf{d}}[f(i,j)] = \sum_{i,j} f(i,j) P_{\mathbf{d}}(i,j) \tag{8.6}$$

which is similar to expectation operation if $P_{\mathbf{d}}(i,j)$ is normalized. Then, the following relations hold $\mu_1 = E_{\mathbf{d}}[i]$, $\sigma_1^2 = E_{\mathbf{d}}[(i-\mu_1)^2]$, $\mu_2 = E_{\mathbf{d}}[j]$, $\sigma_2^2 = E_{\mathbf{d}}[(j-\mu_2)^2]$.

The value obtained in (8.5) can be viewed as $E_{\mathbf{d}}[r(i,j)]$ for the relative error $r(i,j) = \frac{P_{\mathbf{d}}(i,j) - P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)}{P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)}$, i.e.

$$\tau_{\mathbf{d}} = E_{\mathbf{d}}\left[\frac{P_{\mathbf{d}}(i,j) - P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)}{P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)}\right]$$

This can easily be shown from (8.5) using

$$\begin{aligned}
\tau_{\mathbf{d}} &= \sum_{i,j} \frac{P_{\mathbf{d}}^2(i,j)}{P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)} - 1 \\
&= \sum_{i,j} \left(\frac{P_{\mathbf{d}}^2(i,j)}{P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)} - P_{\mathbf{d}}(i,j) \right) \\
&= \sum_{i,j} P_{\mathbf{d}}(i,j) \left[\frac{P_{\mathbf{d}}(i,j) - P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)}{P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)} \right] \\
&= E_{\mathbf{d}}\left[\frac{P_{\mathbf{d}}(i,j) - P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)}{P_{\mathbf{d}}(i)P_{\mathbf{d}}(j)} \right]
\end{aligned} \tag{8.7}$$

Thus, the optimal displacement, $\mathbf{d}_{opt}$ corresponds to the value of $\mathbf{d}$ for which $E_{\mathbf{d}}[r(i,j)]$ in (8.7) is maximum. By choosing maximum values of other textural

features, besides the relative error, one could provide other options for selecting $\mathbf{d}_{opt}$, which are given in Table 8.2.

Table 8.2: Selection of the displacement

Feature selected	$\mathbf{d}_{opt} =$
Energy	$\arg \max_{\mathbf{d}} E_{\mathbf{d}}[P_{\mathbf{d}}(i,j)]$
Contrast	$\arg \max_{\mathbf{d}} E_{\mathbf{d}}[(i-j)^2]$
Entropy	$\arg \max_{\mathbf{d}} E_{\mathbf{d}}[-logP_{\mathbf{d}}(i,j)]$
Homogeneity	$\arg \min_{\mathbf{d}} E_{\mathbf{d}}[\frac{1}{1+\lvert i-j\rvert}]$

In this particular study, an alternative approach was adopted for selecting the direction of displacement vector $\mathbf{d}$ whose magnitude was set to an arbitrary value. The angle was found by treating the object as a set of scattered points. Then, the ellipse that encompasses this object is determined using the eigenvectors and eigenvalues of the data correlation matrix associated with the scattered points. The angle chosen for the displacement vector corresponds to the angle of the eigenvector associated with the largest eigenvalue. Our experimental results reveal the fact that this method provides very good results for this particular application.

8.3 Underwater Imagery: STIL Data Description

STIL sensor produces high-resolution 3-D images of underwater objects. The STIL sensor scanned, line by line, a rectangular area of a target field layout stretched approximately 320 feet long. The data analyzed here was collected in shallow water, approximately 60 feet deep, of the Gulf of Mexico in Panama City, FL on August 14 and 15, 2001. Ten runs were conducted in the first day and three runs on the second day. The results of the present study are based upon only six runs (three runs for each of these days). The collected raw STIL data was then rendered to

produce pairs of contrast (gray-level) and range (distance) maps. Conversion of raw data to rendered and enhanced contrast and range maps is accomplished using an automated routine [109] developed by Coastal Systems Station (CSS) in Panama City, FL. Twelve distinct types of mine-like objects and 32 different types of non-mine-like objects formed the collection. The whole list of objects is presented in Table 8.3. Due to limited size of the database, new images were generated using the geometric transformation on available images. This procedure is explained in Section 8.3.2.

8.3.1 Test Objects & Experimental Setup

Targets used in the data collection exercise were a selection of mine-like objects and some typical non-mine-like objects that could be found on the sea floor with characteristics that resemble mines. Additionally, there are several "technical panels" that are primarily designed and used to quantify certain physical phenomena in STIL imagery and validate the models [110], [111]. Some of these panels are flat made of thick aluminum and painted with specifically designed patterns using paint with specified optical properties. In addition to these panels, a number of raised panels with different patterns were also deployed to quantify range accuracy for the STIL and aid in the development of performance metrics for identification. Contrast images of some technical panels used during testing are shown in Fig. 8.3. Note that for identification purposes, all the objects in the database are tagged (See Table 8.3). The tags of the rectangular shape panels as well as those that correspond to the circular shape panels are also shown in this table.

The mine-like objects included several (twelve) types of bullet shape, cylindrical shape, truncated cone shape, and trapezoidal shape targets. More specifically, there were two truncated cone shape (tags 28 and 29), four trapezoidal shape (tags 30-33), two bullet shape (tags 34 and 35) and four cylindrical shape (tags 36-39) targets.

Some examples of contrast maps of these targets are shown in Fig.8.4.

The non-mike-like objects (man-made clutter) included items that could typically be found in coastal areas, such as a tire (tag 40), a crab/fish trap (tag 42), a concrete pipe (tag 44), a Christmas tree (tag 44) and a 55-gallon drum (tag 41). Some of these non-targets might be confused as targets to a classification system, especially in difficult operating conditions. There was also a 12-foot stepladder that was installed upright on the bottom. This object was only present for a portion of data collection and was intended to help measure the depth of field. Five examples of contrast images for non-mine-like objects are shown in Fig. 8.5.

The complete list of 40 objects, which included mine-like, non-mine-like, and technical panels is presented in Table 8.3. This list includes twelve mine-like targets (four types), five non-mine-like (man-made clutter) objects, and twenty three technical panels. The list of the objects for each of six runs are also presented in [112]. The list was provided by CSS and it includes the frame where each object appears as well as its coordinates within the frame.

The field was made up of all the above-mentioned objects placed along a 500-foot length transect of 1/4-inch wire rope, stretched between two 1600-pound concrete clumps in 60 feet of water. For navigational guidance, four marker buoys were placed 150 feet perpendicular to the centerline at each clump and a reference buoy was placed 120 feet off the center. The result was a 500 x 300 box with the mines and clutter placed along its centerline. The technical panels were deployed by attaching them to the 1/4-inch wire rope with brass clips at predetermined points along the wire rope. This placed the centerline of the panels 2 feet to one side of the wire rope. The panels were spaced 2 feet apart enough distance to avoid interference effects between panels during imaging, but close enough to minimize the target field length. The mine-like and non-mine-like objects (tag 28-43) were placed consecutively after the panels with their centerline aligned with the centerline of the panels. The result

was a line of targets 320 feet long. Two environmental sensor packages were placed along the centerline, one at either end of the technical panels. A detailed layout of the target field is give in [111].

During the course of the field test, a broad range of environmental conditions was experienced [111]. Water clarity ranged from very good to poor, with a suspension layer hovering as much as 10 feet above the bottom. Both environmental and sensor data were collected primarily during daylight hours with some collected at night to serve as a control on the measure of ambient light impact. The TB and sensor packages were operated for a total of seven days and nights of testing. These runs consisted of 76 runs over the primary target field and 23 runs over "targets of opportunity". Altitudes of the data runs were varied over a prescribed range so that the quality of the data varied from poor to very good. Every attempt was made to obtain imagery on both ends of the performance envelope as well as points in between (i.e., from approximately 2.5 beam attenuation lengths to greater than 5 beam attenuation lengths).

8.3.2 Generation of Additional Synthetic Data

Owing to the relatively small number of mine-like and non-mine-like images in our database, any training and testing performed on this limited data set will not be statistically reliable. Additionally, the data for certain objects are scarce while others appear abundantly. The smallest number of occurrence for a mine-like object corresponds to Target 37 as it appears in only four images out of six possible instances while the other mine-like objects appear at least in five images from the six runs taking during two days (see the tables in the report [112]). Another problem is that some objects (either mine-like or non-mine-like) are almost completely out of range or are highly corrupted by noise/distortion, and hence are discarded from further analysis. These issues motivated us to generate additional images in order to guarantee the

validity and statistical significance of the test results. Several geometric transformations can be applied to the original images to generate additional ones that represent viewing the objects at different range and grazing angle. Figure 8.1 illustrates the basic idea behind transforming images on different coordinate systems.

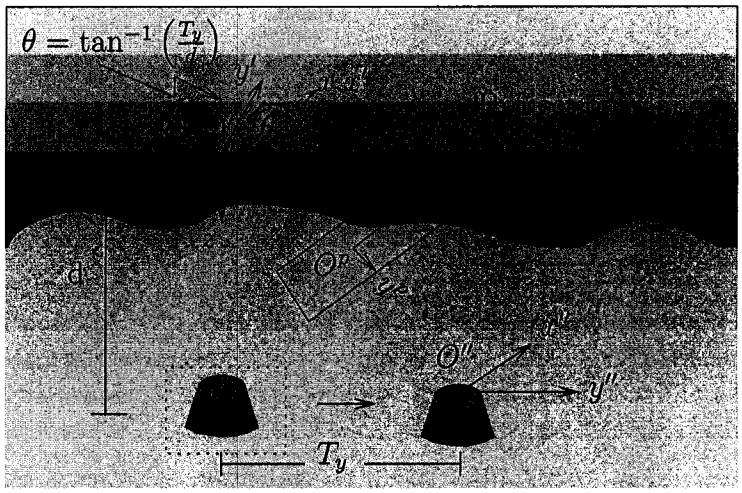


Figure 8.1: Geometric transformation for generating synthetic images.

The initial coordinate system denoted by O has each one of its axes oriented as follows: x-axis is perpendicular to the plane of the picture shown in (Fig. 8.1) and is entering to it; y-axis is horizontal; and the z-axis is vertical with downward direction as shown. Coordinate system O' is a rotated version of O w.r.t x-axis as a pivot. The angle is positive when the rotation is performed as indicated in the figure, otherwise it is negative. This angle is not arbitrary as it follows the movement of an object located at system O''. Coordinate system O'' is the translated version of system O, i.e., (x'', y'', z'')-axis are parallel to (x, y, z)-axis. The translation takes place along the z and y axes with the origin at O'' located at coordinates $(0, T_y, d)$ with respect to system O. The plane of the perspective system O^p is perpendicular to axis z' and is located at a focal distance f from the origin of O'. Figure 8.1 shows these four coordinate systems denoted by O, O', O'' and O^p.

Rendering an object involves basically three steps: first, a translation along the

horizontal, y axis, maps the origin of the object from the point $(x, y, z) = (0, 0, d)$ to point $(x, y, z) = (0, T_y, d)$; second, this translated object is projected onto a plane that is perpendicular to the z'-axis and is located at a distance f from its origin; and finally, for visualization purposes, the projected image on this plane is mapped to within a window. This last transformation involves basically scaling and translating operations.

Given that coordinate system O' is a rotated version of O around the x-axis and is positive in the direction shown in the figure we can write

$$x' = x \tag{8.8}$$

$$y' = y\cos\theta + z\sin\theta \tag{8.9}$$

$$z' = -y\sin\theta + z\cos\theta \tag{8.10}$$

where the angle of rotation θ can be found easily noting that the axis z' points to the origin O'', i.e. we have $\theta = \tan^{-1}\left(\frac{T_y}{d}\right)$.

The coordinate systems O and O'' are parallel and related using

$$x = x'' \tag{8.11}$$

$$y = y'' + T_y \tag{8.12}$$

$$z = z'' + d \tag{8.13}$$

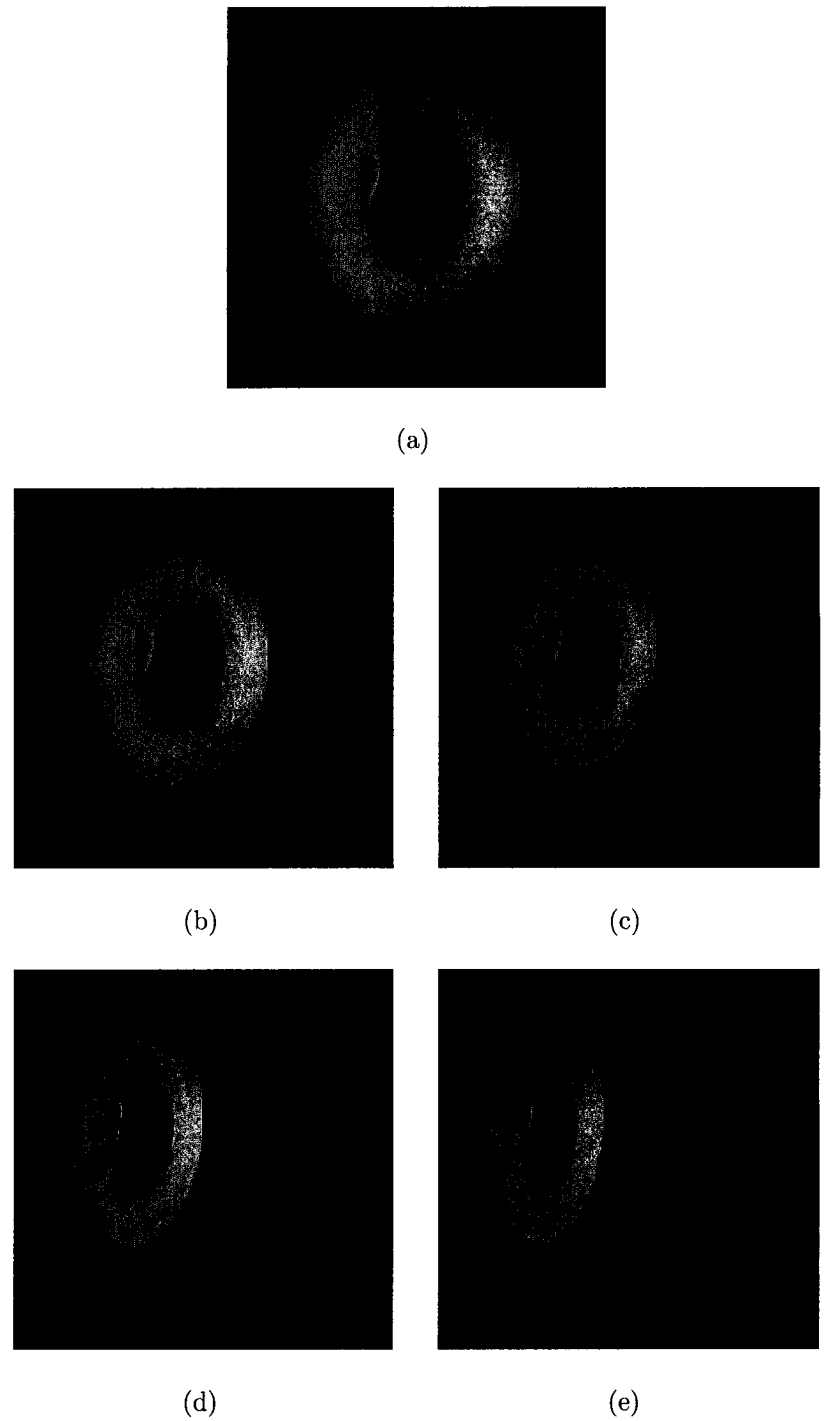
where the origin O'' lies along the z' axis. Since a planar image is used for which the depth coordinate is zero, we set $z'' = 0$. The coordinate systems O' and O^p are parallel with the origin of O^p lying along the z' axis at a focal distance of f from the origin of O. A perspective transformation is involved to convert one point from system O' to O^p. Since the transformation projects point (x', y', z') into the plane located at $(0, 0, f)$, the relation between the two systems can be expressed by

$$x^p = \frac{f}{z'}x' \tag{8.14}$$

$$y^p = \frac{f}{z'}y' \tag{8.15}$$

Now, given that every point in our image is represented in coordinates system O'' and its projection into the system O^p needs to be determined, a sequence of mappings $O'' \longrightarrow O$, $O \longrightarrow O'$ and $O' \longrightarrow O^p$ must be applied in order. The resultant image is then properly scaled and translated to fit within a window for visualization purposes. Figures 8.2(b)-(e) show the results of coordinate transformation applied to the original image in Figure 8.2(a). These new synthetic images that mimic data collection at different range T_y and viewing angle are obtained for different translation values $T_y = 100,\ 200,\ 300,$ and 400, respectively. For all the images the depth d value was set to 400 and the focal distance f was chosen to be 100.

The coordinate transformation method is applied to synthetically generate a relatively large STIL image database (containing 585 pairs of images) that is not only representative of the type of imagery that can be obtained during field test but also can be used to produce statistically significant classification and identification rates. This large data set is available to all researcher for the purpose of validation of the underwater electro-optical sensor models as well as the development of Automatic Target Recognition (ATR) algorithms. Moreover, this data can be used to provide additional insights into the performance envelope of the STIL sensor that could be of great important to the fleet exercises. In this research project, this large data set is used to develop: (a) dedicated filtering and segmentation methods, (b) robust feature extraction schemes that capture shape and texture-dependent characteristics of the objects and (c) efficient and accurate identification algorithms to classify different types of mine-like and non-mine-like objects. These will be discussed in the subsequent sections of this report.



(a)

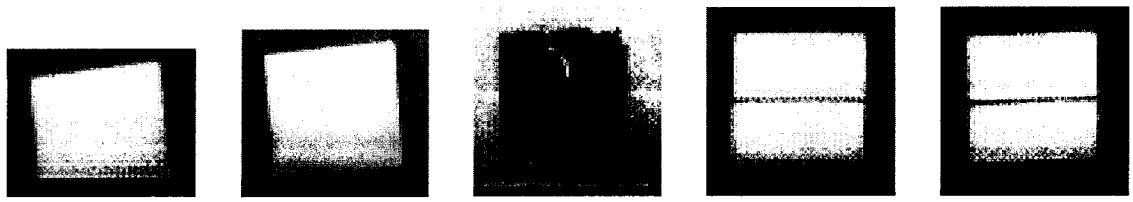
(b)

(c)

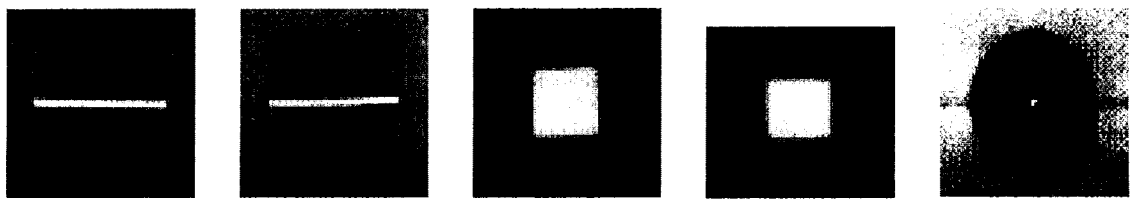
(d)

(e)

Figure 8.2: Coordinate mappings applied to an image. (a) Original image, (b)-(e) four synthetically generated images for $T_y = 100$, 200, 300, and 400, respectively and $d = 400$ and $f = 100$.

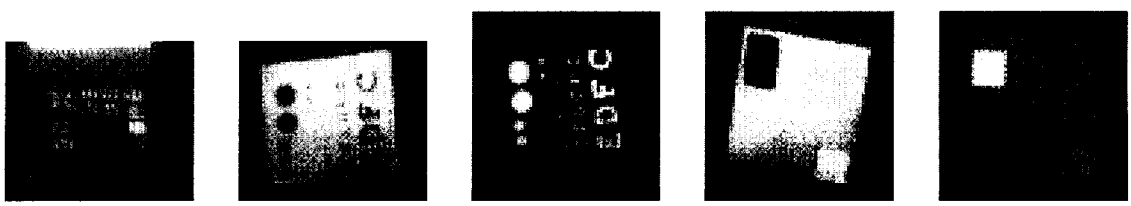


(a) Tag 1 (b) Tag 2 (c) Tag 3 (d) Tag 4 (e) Tag 5

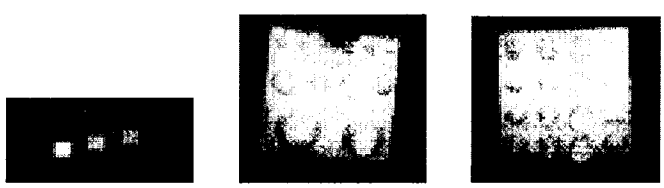


(f) Tag 6 (g) Tag 7 (h) Tag 8 (i) Tag 9 (j) Tag 10

(k) Tag 11 (l) Tag 12 (m) Tag 13 (n) Tag 14 (o) Tag 15

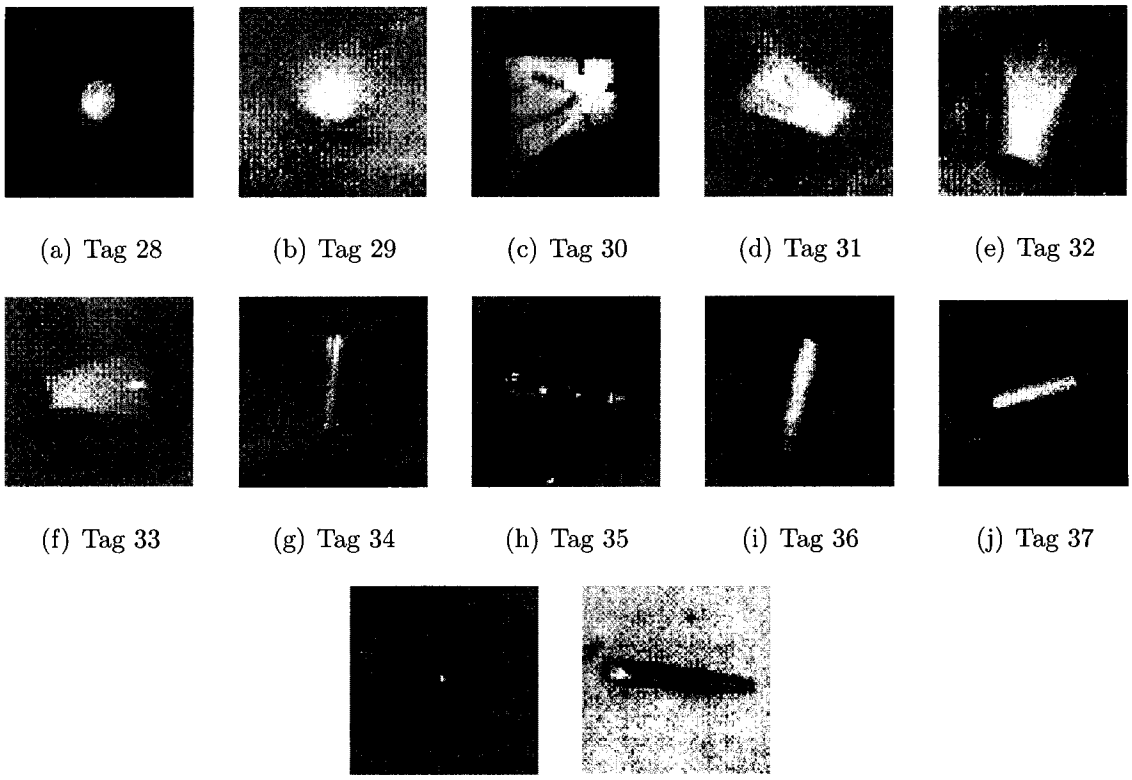


(p) Tag 16 (q) Tag 18 (r) Tag 19 (s) Tag 20 (t) Tag 21



(u) Tag 22 (v) Tag 23 (w) Tag 24

Figure 8.3: Examples of technical panels.



(a) Tag 28 (b) Tag 29 (c) Tag 30 (d) Tag 31 (e) Tag 32

(f) Tag 33 (g) Tag 34 (h) Tag 35 (i) Tag 36 (j) Tag 37

(k) Tag 38 (l) Tag 39

Figure 8.4: Examples of mine-like objects. (a) and (b) Truncated cone shape. (c)-(f) Trapezoidal shape. (g) and (h) Bullet shape. (i)-(l) Cylindrical shape

(a) Tag 27 (b) Tag 40 (c) Tag 41 (d) Tag 42 (e) Tag 44

Figure 8.5: Examples of non-mine-like objects (man-made clutter) (a) A Christmas tree. (b) A tire. (c) A 55-gallon drum. (d) A crab trap. (e) A concrete pipe

Table 8.3: A complete list of objects: mine-like and non-mine-like

Tag #	Name	Comments
1	Solid white rectangle	Technical panel
2	Solid gray rectangle	Technical panel
3	Solid black rectangle	Technical panel
4	Black bar on white rect.	Technical panel
5	Black bar on grey rect.	Technical panel
6	White bar on black rect.	Technical panel
7	Gray bar on black rect.	Technical panel
8	Black/white bar (Horiz.)	Technical panel
9	Black/white bar (Vert.)	Technical panel
10	Black circle, small hole	Technical panel
11	Black circle, large hole	Technical panel
12	White circle	Technical panel
13	Black resolution (left)	Technical panel
14	White resolution (right)	Technical panel
15	Black resolution (center)	Technical panel
16	Black resolution (right)	Technical panel
18	White letter panel	Technical panel
19	Black letter panel	Technical panel
20	3-D boxes # 2 (black)	Technical panel
21	3-D boxes # 1 (white)	Technical panel
22	3-D step	Technical panel
23	3-D raised panel #2 (large)	Technical panel
24	3-D raised panel #1 (small)	Technical panel
27	Christmas tree (pipe)	Clutter
28	Target (mine)	Truncated cone shape
29	Target (mine)	Truncated cone shape
30	Target (mine)	Trapezoidal shape
31	Target (mine)	Trapezoidal shape
32	Target (mine)	Trapezoidal shape
33	Target (mine)	Trapezoidal shape
34	Target (mine)	Bullet shape
35	Target (mine)	Bullet shape
36	Target (mine)	Cylindrical shape
37	Target (mine)	Cylindrical shape
38	Target (mine)	Cylindrical shape
39	Target (mine)	Cylindrical shape
40	Tire	man-clutter
41	55-gallon drum	man-clutter
42	Crab trap	man-clutter
44	Concrete pipe	man-clutter

8.3.3 Experimental Results on Zernike Moments

It is known that Zernike moments are rotation, translation and scaling invariant. However, this is only true if they are applied to continuous image functions. That is, the summations in (8.1) represent approximation to the double integral

$$A_{nm} = \frac{n+1}{\pi} \int \int_{x^2+y^2 \leq 1} f(x,y) V^*_{nm}(\rho, \theta) dx\, dy \tag{8.16}$$

for continuous function $f(x,y)$. This implies that due to this approximation, there will be some error that could adversely affect the invariant characteristic of the Zernike moments. The amount of error depends on the sampling resolution and should be measured on real data in order to determine the impact on the features.

To assess the rotation invariance of several Zernike moments, the magnitudes of four Zernike moments for two mine-like objects were plotted against the rotation angle for values from 0 to 180 degrees in increments of 5 degrees. The Zernike moments chosen were two 2^{nd} order moments namely A_{20} and A_{22}, and two 4^{th} order moments namely A_{40} and A_{42}. Figure 8.6 illustrates the changes as a function of rotation angle for each moment and each object. As can be seen, variations are very minimal. To measure the deviation from the mean (horizontal line) the standard deviation σ was computed. This value is then divided by the mean value μ to provide a new measure $I = \frac{\sigma}{\mu}$, which can then be used to compare the variations of all the moments. Looking at all the plots in Figure 8.6, one can see that this index is rather low, indicating that each Zernike moment indeed remains invariant under rotation, regardless of the angle.

The discrimination capability of the Zernike moments to differentiate these two mine-like objects is remarkable since using a simple horizontal line separator one can classify each object based upon any of the selected moments. Clearly, the discrimination capability of the Zernike moments does not change by rotation of the objects. However, we shall demonstrate that these moments are not invariant to some affine

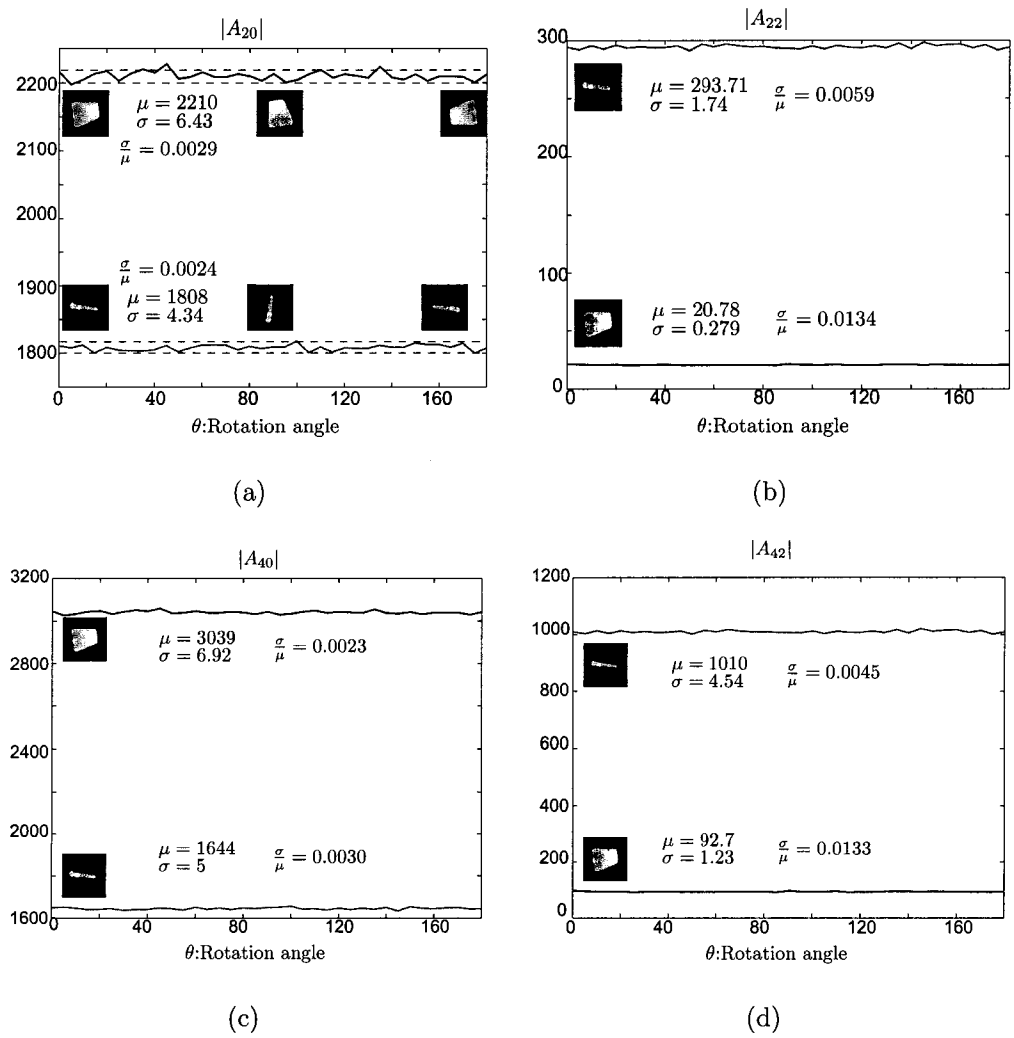


Figure 8.6: Magnitude of several Zernike moments versus the rotation angle for two mine-like objects, a bullet-shape target and a trapezoidal mine.

transformation, i.e., they indeed vary rather significantly depending on the grazing (look) angle of a perspective transformation. Experimental results conducted on two mine-like objects (see Figures 8.7 and 8.8) and the four chosen moments clearly reveal the dependency of the Zernike moments on the grazing angle. Although the rest of the moments were not analyzed, we anticipate that a similar characteristic will still be valid. Figure 8.7 shows plots of $|A_{22}|$ (solid lines) for the two targets at different grazing angles. As can be seen, this moment is indeed dependent on the grazing angle. Furthermore, this dependency can be modelled somewhat accurately by a second

order polynomial (dotted line 8.7)). Thus, although the moments are not invariant, they follow cetain trend, which can be useful if a model is needed for this correction. If $A(\theta)$ represents one of the Zernike moments and θ is the grazing angle, then we can write

$$A(\theta) = a\theta^2 + b\theta + c + \eta \tag{8.17}$$

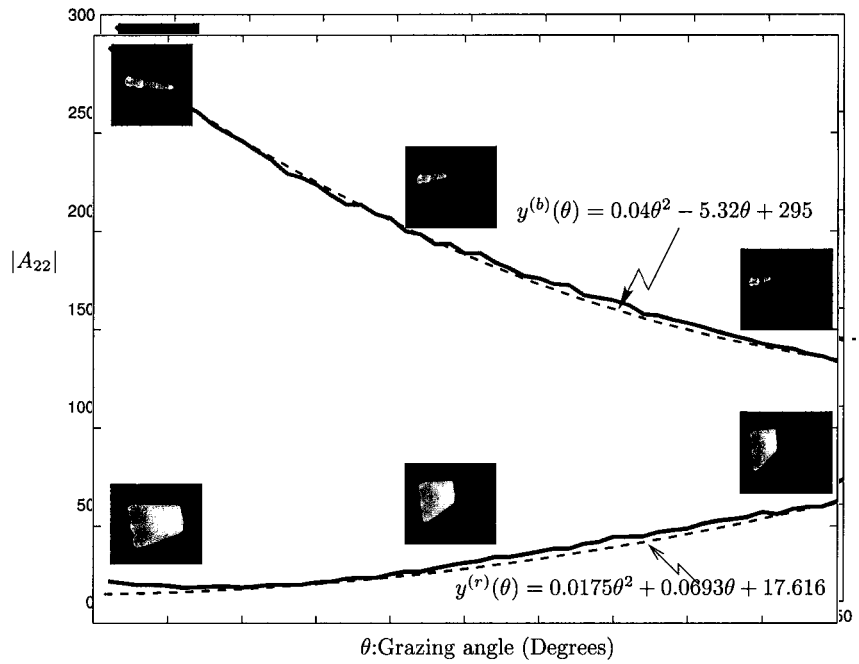


Figure 8.7: Magnitude of the second order momentum $|A_{22}|$ versus the grazing angle for two mine-like targets, a bullet shape target (upper plot) and a trapezoidal target (lower plot).

where a, b and c are some parameters that need to be identified using least squares method, and η represents noise or inaccuracy in the modelling. The properties of the noise for each moment are different. While the noise variance for $|A_{20}|$ is relatively high, this variance for $|A_{42}|$ is rather low.

Similar tests have to be carried out on the other moments to verify whether similar

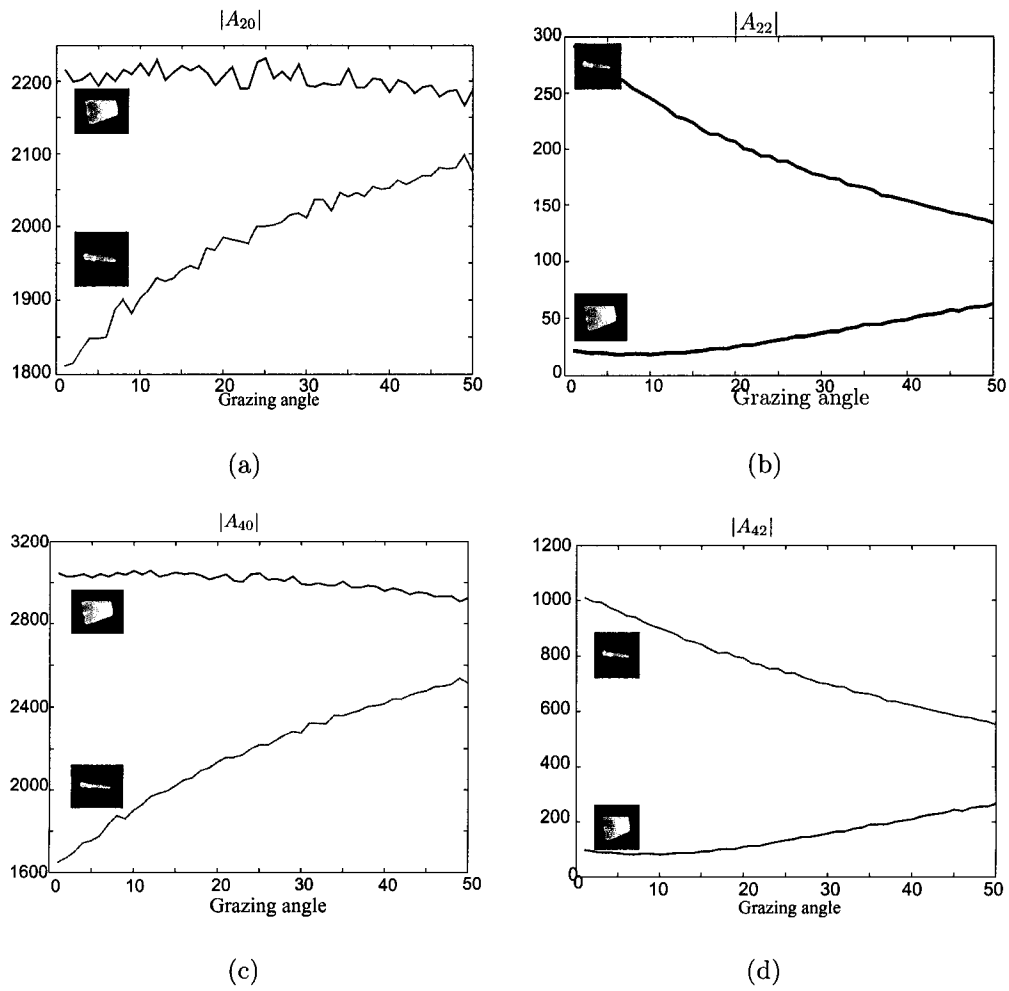


Figure 8.8: Zernike moments for a bullet-shape and a trapezoidal target (tag 30) vs. the grazing angle.

observations hold. Nevertheless, we can conclude that the Zernike moment features are not invariant to affine transformations, particularly to changes in the grazing angle.

8.3.4 Experimental Results on Textural Features

Our experimental results demonstrated that although Zernike moments are invariant to some affine transformations, e.g. rotation, translation and scaling, they indeed change with respect to grazing angles changes. Thus, it is important to complement these features with another set of features that remain invariant to the grazing angle

changes.

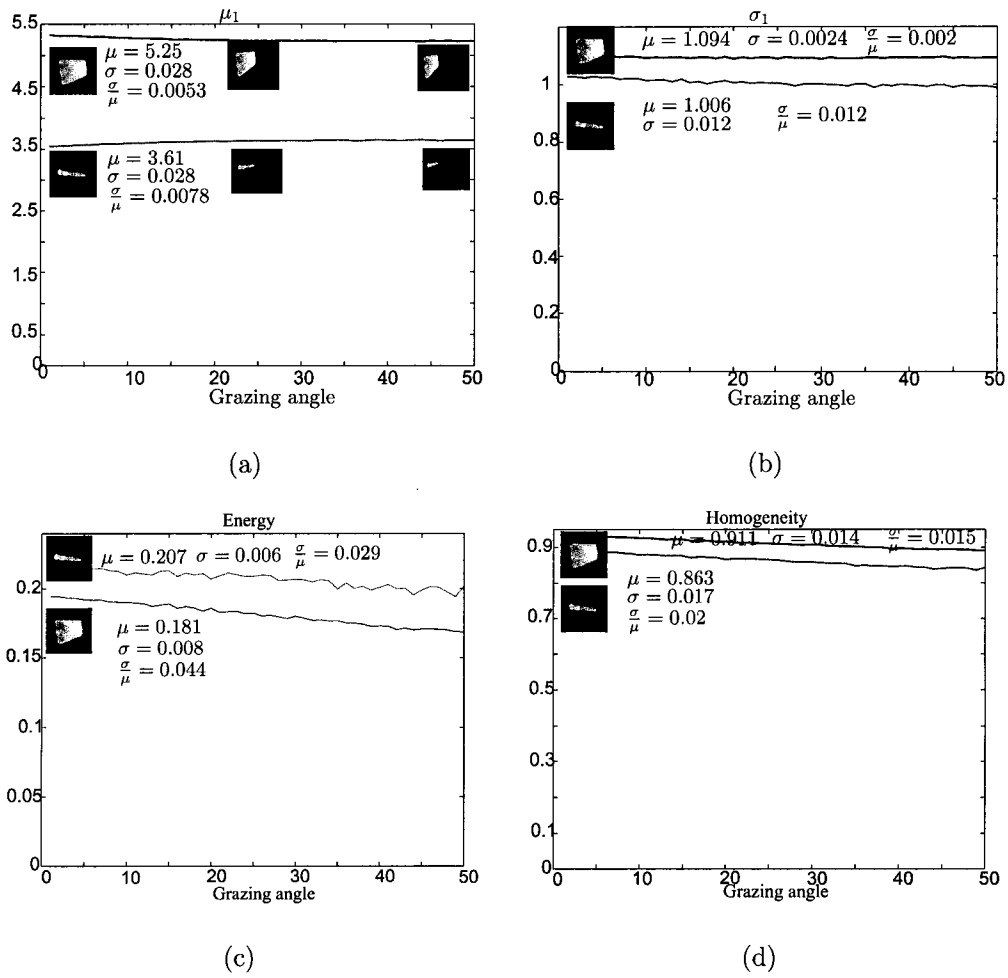


Figure 8.9: Several texture features versus the grazing angle for two mine-like objects, a bullet-shape target and a trapezoidal mine.

To show that GLCM textural features are invariant to changes in the grazing angle, we repeated the experiment in Section 8.3.3 for four GLCM features, namely μ_1, σ_1, energy and homogeneity. Figure 8.9 (a)-(d) show the plots of these features for the same two targets versus grazing angle. As evident from these results, grazing angle variations has little or no effect on these features.

8.4 Conclusions

Zernike moments provide a good set of features useful to represent shape-dependent features of mine-like and non-mine-like objects. Although these moments are invariant to some transformations such as rotation, translation and scaling, we showed that they are not invariant to the grazing angle changes. However, the trend that each moment follows could be modelled by a polynomial expansion. From the analysis of some of the low order moments, it appears that a 2^{nd} order polynomials could successfully represent this trend. This assumption should be verified on other Zernike moments. Such simple models can be used to undo the effects of transformation using an inverse modelling approach in order to yield invariance with respect to the grazing angle. This topic could be pursued in future research.

Owing to the variance of the Zernike moments to grazing angle changes, it is expected that the shape-dependent features alone cannot provide good discrimination ability to separate different classes of mine-like and non-mine-like objects from each other. This deficiency of the Zernike moments will be demonstrated in Section 5 of this report. Nevertheless, we will show that augmenting the shape-dependent features (Zernike moments) with textural features obtained using GLCM method can substantially improve the classification results as we showed that the latter features are somewhat independent on the grazing angle. The independence of the GLCM features, namely $\mu_1, \mu_2, \sigma_1, \sigma_2$, to grazing angle changes points to the important conclusion that these features play an important role for successful underwater target/non-target identification.

CHAPTER 9

A MULTIPLE REGULATOR IMAGE RETRIEVAL SYSTEM

9.1 Introduction

This chapter presents an adaptable image retrieval system developed using adaptive control theory and kernel-based machines. The image retrieval system referred to as a multiple regulator image retrieval system (MRIRS) consists of a retrieval subsystem (plant) that carries certain image-dependant information, multiple query mapping subsystems that drive the error between the outputs of the retrieval and the reference model to zero, a relevance feedback mechanism that incorporates user expertise, and a reference model retrieval system based upon the image database together with the class labels. The proposed system can incorporate expertise via relevance feedback from multiple expert users in online fashion and with minimum user involvement. The main characteristic of the proposed system is that it can easily integrate new concepts with old ones already stored in the regulators. In absence of the model reference the internal parameters of the regulator are set up appropriately using a rank one projection matrix.

The fundamental parts of our multiple regulator image retrieval system are its regulators, represented by a set of matrices as shown in Figure 7.1. The purpose of these regulators is not only to provide a good list of images but also to gather

users's feedback, acting like knowledge repositories. We have devised three different learning phases for updating their parameters, namely initial start up, model-reference learning, and relevance feedback learning as in the proposed text retrieval system. The initial start up refers to setting up the parameters by knowing solely the images in the database without relying on information from users. The start up is essential given that the system can operate even without receiving feedback from the users, which is a typical case in most image retrieval systems where most users do not provide this information. Model-reference learning is used to incorporate other prior information from an external model reference (when it is available) or from the image database augmented with their corresponding class labels. This is performed off-line. Relevance feedback learning, on the other hand, aims at capturing user's expertise in an online fashion. In the next sections, these three phases of learning for the multiple regulator system in Figure 7.1 will be described in detail.

9.2 Initial Start Up

Any realistic image retrieval system should provide provisions for the initial start up or the initialization of the parameters of the system. Moreover, for the system to be of practical use it is important to implement the underlying operations in real-time, if possible, and with minimum usage of computational resources. Consider the multiple regulator retrieval system shown in Figure 7.1. For a database with N images, $\underline{x}_j$, $j \in [1, N]$, the initial start up in this system involves finding matrices $A_1, A_2, \ldots, A_N$ for the N regulators and an appropriate mapping function $\Phi(.)$ that yields an adequate scoring function. Then, the retrieval status function for the j^{th} image upon submitting new query image $\underline{q}$ is

$$r_j(\underline{q}) = \Phi^T(\underline{x}_j)\hat{\underline{q}}_j = \Phi^T(\underline{x}_j)A_j\Phi(\underline{q}). \tag{9.1}$$

That is, the purpose of the mapped query $\hat{\underline{q}}_j$ supplied by regulator j is to provide a specific $r_j(\underline{q})$ for image $\underline{x}_j$. Let us assume that the mapping function is linear, i.e.

$\Phi(\underline{q}) = \underline{q}$ and $\Phi(\underline{x}_j) = \underline{x}_j$. Under this assumption, the scoring function becomes $r_j(\underline{q}) = \underline{x}_j^T A_j \underline{q}$. The mapping matrix A_j can initially be chosen as the identity matrix $I_{M\times M}$ or the projection matrix $P_{\underline{x}_j} = \underline{x}_j(\underline{x}_j^T\underline{x}_j)^{-1}\underline{x}_j^T$. In either case, we arrive at the directional cosine scoring function[1] i.e. $r_j(\underline{q}) = \underline{x}_j^T\underline{q}$, commonly used in most retrieval systems. However, the projection matrix is a preferred choice as the energy ξ defined as $\xi = tr(P_{\underline{x}_j}^T P_{\underline{x}_j}) = tr(P_{\underline{x}_j}^2)$, where $tr(.)$ is the trace operator, is concentrated in one eigenvector $\underline{x}_j$, while the energy of the identity matrix $I_{M\times M}$ is scattered among M eigenvectors as will be shown later. Furthermore, for the projection matrix $\xi = 1$, while for the identity matrix $\xi = M$, i.e. M times the energy of the projection matrix. Although the two matrices produce the same scoring function $r_j(\underline{q}) = \underline{x}_j^T\underline{q}$ the projection matrix is preferred because of this smaller energy.

Remark 9.1 In the multiple regulator retrieval system, the projection matrix $P_{\underline{x}_j}$ associated with image $\underline{x}_j$ has a single eigenvalue of '1' and the single eigenvector $\underline{x}_j$. Furthermore, this projection matrix is positive definite (PD) as $\underline{v}^T P_{\underline{x}_j}\underline{v} > 0$ holds for all $\underline{v} \in \mathbb{R}^M$.

Using the projection matrix for the non-linear case, matrix A_j associated with the j^{th} regulator can be expressed as $A_j = \Phi(\underline{x}_j)[\Phi^T(\underline{x}_j)\ \Phi(\underline{x}_j)]^{-1}\Phi^T(\underline{x}_j) = \Phi(\underline{x}_j)\ \Phi^T(\underline{x}_j)/k(\underline{x}_j, \underline{x}_j)$, where $k(\underline{x}_j, \underline{x}_j) = \Phi^T(\underline{x}_j)\Phi(\underline{x}_j)$ is a scalar kernel function and matrix $\Phi(\underline{x}_j)\Phi^T(\underline{x}_j)$ is a rank-one estimate of the correlation matrix for the transformed image $\Phi(\underline{x}_j)$. Note that the requirement for choosing $\Phi(\cdot)$ will be discussed in Section 9.3.

9.3 Model Reference Learning

The goal of the model-reference learning phase is to capture the input-output mapping of a reference model for an ensemble set of query images, their corresponding listed images and scores. The initial model-reference learning developed in this section as well as the relevance feedback learning presented in Section 9.4 do not rely on the

[1]It is assumed that the feature vectors are normalized.

class membership information of the retrieved images as this information might not be available.

This section also addresses the steps for the creation of N practical kernel-based network retrieval systems associated with the multiple regulator system shown in Figure 7.1. The three-layer "conceptual" network that forms the j^{th} scoring function assigns a score to the j^{th} image $\underline{x}_j$ in the database (see Chapter 7 for definitions) for a submitted query image, $\underline{q}$, as shown in Figure 9.1. The term "conceptual" is referred to a system that does not exist in practice and it is only for the purpose of illustrating some mathematical concepts involved in the development of a "practical" retrieval system as shown in Figure 9.2. The first layer (static) maps the original submitted query $\underline{q}$ into a possibly high-dimensional mapped feature vector $\underline{q}'$ using the transformation $\Phi(.) : \mathbb{R}^M \rightarrow \mathbb{R}^{N_f}$ of the form

$$\underline{q}' = \Phi(\underline{q}) = [\phi_1(\underline{q}) \;\; \phi_2(\underline{q}) \;\; \ldots \;\; \phi_{N_f}(\underline{q})]^T, \tag{9.2}$$

where $\phi_j(.) : \mathbb{R}^M \rightarrow \mathbb{R}$, $j = 1, 2, \ldots, N_f$, are related to the kernel producing function $\Phi(.)$. These possible large number of functions are not explicitly computed; instead they are used to find the scoring functions $r_j(\underline{q})$'s, $i \in [1, N]$ in (9.1), that assign appropriate score values to the listed images for input query $\underline{q}$. The requirement of having a large number of functions $\phi_j(.)$'s is needed to guarantee that the information gathered from the users or from the model reference is assimilated without jeopardizing the performance of the retrieval system for previously learnt queries. The second layer is adaptable and can incorporate, in an online fashion, the expert users' relevance feedback or in batch mode the information that is preestablished in a set of input-output relations imposed by the reference model (see Figure 7.1). The output layer contains N neurons, each representing an image in the database.

In the model-reference learning the process of generating the scoring function $r_j(.)$, for the j^{th} neuron, $j \in [1, N]$, involves finding new mapping matrix $\hat{A}_j$ and the

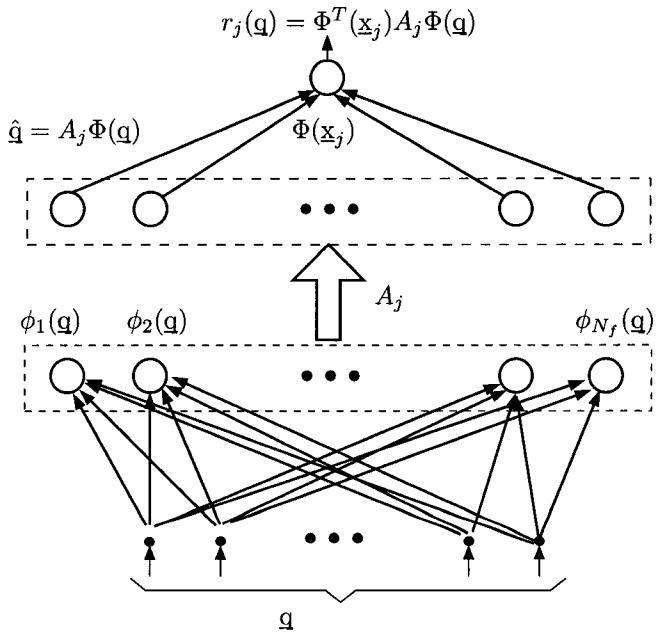


Figure 9.1: Retrieval function for the j^{th} Output Neuron.

mapping function $\Phi(.)$ given a set of L model-reference training pairs $\left\{\underline{q}_i, \hat{r}_j^{(i)}\right\}_{i=1}^{L}$ created from the augmented database with class label (referred to as model-reference database) information used by the model reference, where $\underline{q}_i$ is the i^{th} query and $\hat{r}_j^{(i)}$ is its desired score. The problem can be stated as follows. Find $\hat{A}_j$ and $\Phi(.)$ such that:

$$\Phi(\underline{x}_j)^T\hat{A}_j\Phi(\underline{q}_i) = \hat{r}_j^{(i)}, \quad i \in [1, L]. \tag{9.3}$$

Clearly, the problem is ill-posed because many choices for the mapping matrices $\hat{A}_j$ and functions $\Phi(.)$ can potentially satisfy (9.3) requirement. Furthermore, due to the high dimensionality of $\Phi(.)$, the problem will remain ill-posed even though a particular class of functions is chosen. Let us decouple the problem of finding matrix $\hat{A}_j$ and the function $\Phi(.)$ that 'best' satisfy (9.3) into two problems: (a) finding a set of $N_f \leq \infty$ basis functions $\{\phi_l(.)\}_{l=1}^{N_f}$ with "good" properties that can be used to approximate an underlying scoring function, and (b) finding the matrix $\hat{A}_j$ using the chosen basis functions.

Let us confine the basis eigenfunctions $\phi_l(.)$'s to those generated by some positive and symmetric (i.e. $k(\underline{s},\underline{t}) = k(\underline{t},\underline{s})$) kernel $k(\underline{s},\underline{t})$ [113], and

$$\phi_l(\underline{s}) = \lambda_l \int k(\underline{s},\underline{t})\phi_l(\underline{t})d\underline{t}, \tag{9.4}$$

where λ_l is the l^{th} eigenvalue of $k(\underline{s},\underline{t})$. According to Mercer's theorem [113, 114], to guarantee that a continuous symmetric function $k(\underline{s},\underline{t})$ from $L_2(\mathcal{C})$ ($\mathcal{C}$ is a compact subset of $\mathbb{R}^n$) can be expanded as

$$k(\underline{s},\underline{t}) = \sum_{l=1}^{N_f} \lambda_l \phi_l(\underline{s})\phi_l(\underline{t}), \tag{9.5}$$

it is necessary and sufficient that $\iint k(\underline{s},\underline{t})f(\underline{s})f(\underline{t})d\underline{s}d\underline{t} \geq 0$ holds for all $f \in L_2(\mathcal{C})$. Given that all eigenvalues of a positive and symmetric kernel are positive, we can define $\phi_l(.) \leftarrow \sqrt{\lambda_l}\phi_l(.)$ and plug this value in (9.5). Thus, the kernel $k(\underline{s},\underline{t})$ can be expressed as

$$k(\underline{s},\underline{t}) = \sum_{l=1}^{N_f} \phi_l(\underline{s})\phi_l(\underline{t}) = \Phi^T(\underline{s})\Phi(\underline{t}), \tag{9.6}$$

where $\Phi(.)$ is defined in (9.2). When the kernel is expanded using a finite number of eigenfunctions, it is said that the kernel is degenerate [113].

Having chosen a kernel that generates a specific set of basis functions, we can concentrate on solving the ill-posed problem in (9.3). It is ill-posed because if matrix $\hat{A}_j$ is a solution to (9.3) so is matrix $\hat{A}_j + B_j$, where each column of B_j is orthogonal to $\Phi(\underline{x}_j)$. Therefore, a solution $\hat{A}_j$ with minimum Frobenius norm requires that each column is a multiple of $\Phi(\underline{x}_j)$ leading to a matrix in the form

$$\hat{A}_j = \Phi(\underline{x}_j)\underline{w}_j^T, \tag{9.7}$$

with some weight vector $\underline{w}_j$. Moreover, the information will be concentrated in a few elements as $\hat{A}_j$ will be rank 1. Now, let us transform the ill-posed problem into a

well-posed one by introducing the cost function $\mathcal{T}$ [115,116]

$$\begin{aligned}\mathcal{T}(\underline{w}_j, \lambda) &= \frac{1}{2}\sum_{i=1}^{L}(\hat{r}_j^{(i)} - \Phi^T(\underline{x}_j)\Phi(\underline{x}_j)\underline{w}_j^T\Phi(\underline{q}_i))^2 + \frac{1}{2}\lambda||\underline{w}_j||^2 \\ &= \frac{1}{2}||\hat{\underline{r}}_j - \Psi^T\underline{w}_j\Phi^T(\underline{x}_j)\Phi(\underline{x}_j)||^2 + \frac{1}{2}\lambda||\underline{w}_j||^2 \qquad (9.8) \\ &= \frac{1}{2}||\hat{\underline{r}}_j - c_j\Psi^T\underline{w}_j||^2 + \frac{1}{2}\lambda||\underline{w}_j||^2, \qquad (9.9)\end{aligned}$$

where $\hat{\underline{r}}_j = [\hat{r}_j^{(1)}\ \hat{r}_j^{(2)}\ \ldots\ \hat{r}_j^{(L)}]^T$ is the vector of desired scores for queries $\underline{q}_i$, $i \in [1, L]$, $c_j = k(\underline{x}_j, \underline{x}_j) = \Phi^T(\underline{x}_j)\Phi(\underline{x}_j)$ and $\Psi = [\Phi(\underline{q}_1)\ \Phi(\underline{q}_2)\ \ldots\ \Phi(\underline{q}_L)]$ is the matrix containing all the mapped queries.

The first term in (9.8) is the error term induced when trying to reproduce the scores $\hat{r}_j^{(i)}$, $i \in [1, L]$, while the second term reflects the a priori information that the solution for $\underline{w}_j$ should be close to the origin. Using this function, the problem of finding optimal $\underline{w}_j$ can be cast in an optimization framework where we would like to find $\underline{w}_j^*$ such that

$$\underline{w}_j^* = arg\ \min_{\underline{w}_j} \mathcal{T}(\underline{w}_j, \lambda). \qquad (9.10)$$

Now, let us assume that $\underline{w}_j$ can be expressed as a sum of the optimal vector $\underline{w}_j^*$ and a perturbation vector $\underline{\eta}$, i.e. $\underline{w}_j = \underline{w}_j^* + \alpha\underline{\eta}$, where α is a scalar. Then, (9.9) can be expressed as a function of α and $\underline{\eta}$:

$$\mathcal{T}(\alpha, \underline{\eta}) = \frac{1}{2}||\hat{\underline{r}}_j - c_j\Psi^T(\underline{w}_j^* + \alpha\underline{\eta})||^2 + \frac{1}{2}\lambda(\underline{w}_j^* + \alpha\underline{\eta})^T(\underline{w}_j^* + \alpha\underline{\eta}). \qquad (9.11)$$

Now, take the partial derivative of $\mathcal{T}$ w.r.t α and set it to zero. At $\alpha = 0$, that corresponds to the optimal solution ($\underline{w}_j = \underline{w}_j^*$), we have

$$\left.\frac{\partial\mathcal{T}(\alpha, \underline{\eta})}{\partial\alpha}\right|_{\alpha=0} \Rightarrow\ -[\hat{\underline{r}}_j - c_j\Psi^T\underline{w}_j^*]^T c_j\Psi^T + \lambda\underline{w}_j^{*T} = 0. \qquad (9.12)$$

As can be seen, the optimal weight vector $\underline{w}_j^*$ depends on the penalizing parameter λ. In order to reduce the error term in (9.8) to its minimum value, thus ensuring the maximum score reproduction, we want the optimal vector $\underline{w}_j^*$ for λ approaching zero. That is

$$\underline{w}_j^* = \lim_{\lambda\to 0}[\lambda I + c_j^2\Psi\Psi^T]^{-1}c_j\Psi\hat{\underline{r}}_j. \qquad (9.13)$$

Thus, the optimal vector for this case is

$$\underline{w}_j^* = \frac{1}{c_j}\Psi[\Psi^T\Psi]^{-1}\hat{\underline{r}}_j. \tag{9.14}$$

As can be seen from (9.14), the solution for the optimal vector is dependent on the existence of the inverse of the $L \times L$ Gram matrix $G = [\Psi^T\Psi]$ with elements $g_{l,m} = k(\underline{q}_l, \underline{q}_m)$. To guarantee its invertibility we confine the set of kernel functions to those that are symmetric, PD and non-degenerate [113]. An example of a non-degenerate kernel is the Gaussian kernel defined as $k(\underline{s}, \underline{t}) = e^{-a\|\underline{s}-\underline{t}\|^2}$, where a is a constant and positive real number.

Once the model reference learning is completed for all the image queries in the model-reference training database the corresponding $\underline{w}_j^*$'s (or $\hat{A}_j$'s) are identified to produce the desired scores for the chosen images. Now, one can compute the scoring function $r_j(\underline{q})$ in (9.1) for a query $\underline{q}$ by plugging the optimal vector $\underline{w}_j^*$ in (9.14) into (9.7). This yields

$$\begin{aligned} r_j(\underline{q}) &= \Phi^T(\underline{q})\Psi[\Psi^T\Psi]^{-1}\hat{\underline{r}}_j & (9.15)\\ &= [k(\underline{q}, \underline{q}_1) \ \ldots \ k(\underline{q}, \underline{q}_L)][\Psi^T\Psi]^{-1}\hat{\underline{r}}_j. & (9.16) \end{aligned}$$

Alternatively, we can write

$$\begin{aligned} r_j(\underline{q}) &= [k(\underline{q}, \underline{q}_1)\ k(\underline{q}, \underline{q}_2) \ \ldots \ k(\underline{q}, \underline{q}_L)]\ \underline{b}_j \\ &= \sum_{i=1}^{L} b_{ji}k(\underline{q}, \underline{q}_i) = \underline{b}_j^T\mathcal{K}, & (9.17) \end{aligned}$$

where $\underline{b}_j := [\Psi^T\Psi]^{-1}\hat{\underline{r}}_j$ and $\mathcal{K} := [k(\underline{q}, \underline{q}_1) \ \ldots \ k(\underline{q}, \underline{q}_L)]^T$. We use the latter formulation to create the "practical" two-layer retrieval system illustrated in Figure 9.2, and then adjust the weight vectors that connect the pools to the output layer using the information in the model-reference training database for this model-reference learning.

Figure 9.2 shows the practical retrieval network used to score the new query image,

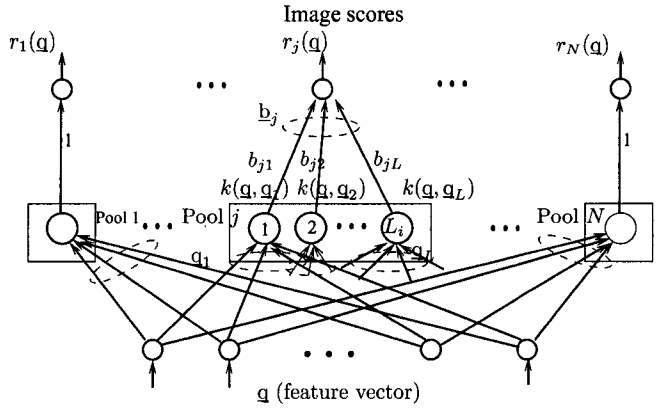


Figure 9.2: Practical Retrieval Network associated with the Multiple Regulator Retrieval System.

$\underline{q}$. Each neuron in the output layer represents one of the query images in the model-reference database and provides a multivariate scoring function, $r_j(\underline{q})$, that computes a score to a submitted query, $\underline{q}$, by using a linear combination in (9.17) of the outputs of the neurons within pool j in the first layer, which contains a set of neuron pools. Each pool of neurons is assigned to one of the images in the database and contains the set of neurons representing queries that have been learnt for this image. Each neuron in a pool is connected to the output neuron associated with the same image. Note that here we assume that several queries $\underline{q}_i$'s, $i \in [1, L]$, are captured by pool j leading to L neurons.

Thus, during the initial training we set up the network by forming the pools and choosing an appropriate kernel function. A pool is initially formed of one neuron and the feature vector $\underline{x}_j$ corresponding to the j^{th} image in the original database is incorporated into the pool by setting $\underline{q}_1 = \underline{x}_j$. This is repeated for all the pools associated with each one of the images in the database. The weight connection of the neuron of a pool to the corresponding neuron in the output layer is set to 1. Thus, the initial scoring function computed by the j^{th} output neuron becomes $r_j(\underline{q}) = k(\underline{q}, \underline{x}_j)$ if $A_j = \Phi(\underline{x}_j)\Phi^T(\underline{x}_j)/k(\underline{q}, \underline{x}_j)$ is the rank one correlation matrix for the initial start

up.

Remark 9.2 An appropriate kernel can be chosen as a monotonically decreasing function of the distance between image and query, i.e. if $||\underline{q} - \underline{x}_j|| < ||\underline{q} - \underline{x}_k||$ then the kernel should satisfy $k(\underline{q}, \underline{x}_j) > k(\underline{q}, \underline{x}_k)$. This guarantees that the scoring function does not alter the order of the relevant and non-relevant displayed images. Both the Gaussian kernel and the inverse-multi-quadrics kernel $k(\underline{s}, \underline{t}) = 1/(||\underline{s} - \underline{t}||^2 + a)^{1/2}$ satisfy this requirement; while the multi-quadrics kernel[2] $k(\underline{s}, \underline{t}) = (||\underline{s} - \underline{t}||^2 + a)^{1/2}$ does not. For these kernels, a is a positive scalar quantity.

In absence of an external image retrieval system, model reference learning aims at capturing some a priori information, such as class membership of the objects, shared by the images in the database. Although for general-purpose databases this information may not be readily available, for some specific-purpose databases, like the one used in this study, the information is available. To describe the model reference learning process, let us assume that the object classes of the images in the database are known. Clearly, in this case, the goal of the learning process is to make sure that for a submitted query, i.e. an image in the database, all the images of the same object type as that of the query appear at the top. To do this, a model-reference training database is generated using the information of the images in the database and their class labels. Every image in the database $\{\underline{x}_j\}_{j=1}^N$ is submitted as a query image, i.e $\underline{q} = \underline{x}_j$ for $j \in [1, N]$, displayed images are automatically evaluated, and if a relevant image score falls bellow certain threshold then its 'desired' score is set to the threshold and stored in the model-reference training database along with the submitted query. Once all the query images have been submitted, the information in the model-reference training database is used to train, in batch mode, each one of the pools of the network.

[2]The multi-quadrics kernel is non-positive [59]

9.4 Relevance Feedback via Structural Adaptation

Relevance feedback learning is a methodology to incorporate users' feedback into the retrieval system as soon as it becomes available, and hence it calls for an online implementation. For each query image the list of retrieved relevant and non-relevant images is evaluated and the user can either promote one or more images by raising their relative scores (positive relevance feedback) or demote other images by lowering their (negative relevance feedback). These scores of the affected images are then used to generate an error signal to stimulate the appropriate regulator(s). As can be seen, the learning process depends exclusively on the visual evaluation of the retrieved images, which may not consider the knowledge of the classes. In general, requiring the user to assign classes could be a tedious or a very difficult task, especially when the categorization of the images is not-well defined, e.g. an image may belong to two or more classes.

The relevance feedback learning process goes like this. When a user submits a query image, the retrieval system assigns a score to each one of the N pools in Figure 9.2 and displays the images along with their respective scores. The score of each image is computed using the linear combination of the outputs of the neurons within a pool as given by (9.17). Depending on the retrieved results, the expert user may decide to (a) change the score of a displayed image that he/she considers to be more relevant than other highly scored images or (b) do not alter the retrieval list. The former case occurs when the new query image is close enough to the previously stored images and the user is satisfied with the listed retrieved images but not necessarily with their scores. In case that the user has assigned new relevance information (new scores), the system computes an error signal based upon the new score and the previously stored one. The regulator of the voted image takes the error signal and according to an appropriate learning rule adjusts its parameters. In our system, relevance feedback from expert users is captured through feature adaptation, which relies on structural

adaptation and more specifically node expansion in the corresponding neuron pool in the first layer. The goal there is to capture an underlying function that assigns a score to a retrieved image. This process ensures that the previous learning is retained along with the newly learnt information provided by the expert users. This procedure is described in details below.

Assume that at time t_0, L queries have already been learnt and stored in neuron pool j, as shown in Figure 9.2, and now at time t_1 a new query $\underline{q} = \underline{q}_{L+1}$ arrives. For this query, suppose the expert user reviews the retrieved list of scored images and selects image j in the log-file and wants to assign a new score, say $\hat{r}_j^{(L+1)}$ to it. Then the procedure is as follows. Let $\underline{w}_j^{(0)}$ and $\hat{\underline{r}}_j^{(0)}$ be the weight and score vectors for the previously learnt queries respectively, and let $\underline{w}_j^{(1)}$ and $\hat{\underline{r}}_j^{(1)} = [\hat{\underline{r}}_j^{(0)^T} r_j^{(L+1)}]^T$ be the new weight vector and the new desired score vector at time t_1. It is desirable to generate the new weight vector by slightly modifying vector $\underline{w}_j^{(0)}$ with the aim of producing the new score and at the same time maintaining the previous scores of the stored images $\underline{q}_1, \underline{q}_2, \ldots, \underline{q}_L$. That is, we require $\underline{w}_j^{(1)} = \underline{w}_j^{(0)} + \Delta\underline{w}_j$ and the problem can be cast by the function in (9.9) as

$$\mathcal{T}(\Delta\underline{w}_j, \lambda) = \frac{1}{2}||\hat{\underline{r}}_j^{(1)} - c_j \Psi^{(1)^T}(\underline{w}_j^{(0)} + \Delta\underline{w}_j)||^2 + \frac{1}{2}\lambda||\Delta\underline{w}_j||^2. \tag{9.18}$$

Solving for optimal $\Delta\underline{w}_j^*$, and similar to the result for $\underline{w}_j^*$ we get

$$\Delta\underline{w}_j^* = \frac{1}{c_j}\Psi^{(1)}[\Psi^{(1)^T}\Psi^{(1)}]^{-1}[\hat{\underline{r}}_j^{(1)} - c_j\Psi^{(1)^T}\underline{w}_j^{(0)}]. \tag{9.19}$$

Now, using (9.15) with $\underline{w}_j^* = \underline{w}_j^{(0)} + \Delta\underline{w}_j^*$, the new scoring function for the j^{th} image and for query $\underline{q}$ becomes

$$r_j^{(1)}(\underline{q}) = r_j^{(0)}(\underline{q}) + \mathcal{K}^{(1)^T}[\Psi^{(1)^T}\Psi^{(1)}]^{-1}[\hat{\underline{r}}_j^{(1)} - c_j\Psi^{(1)^T}\underline{w}_j^{(0)}], \tag{9.20}$$

where $\mathcal{K}^{(1)} = [k(\underline{q}, \underline{q}_1) \ \ldots \ k(\underline{q}, \underline{q}_{L+1})]^T = [\mathcal{K}^{(0)^T} k(\underline{q}, \underline{q}_{L+1})]^T$ and $r_j^{(0)}(\underline{q})$ is the old scoring defined as: $r_j^{(0)}(\underline{q}) = c_j\Phi(\underline{q})^T\underline{w}_j^{(0)}$. Equation (9.20) can be simplified significantly

to yield a scalar scoring function that depends on the previously computed values. Let us start by reducing the vector $[\underline{\hat{r}}_j^{(1)} - c_j \Psi^{(1)^T} \underline{w}_j^{(0)}]$ using $\underline{\hat{r}}_j^{(1)} = [\underline{\hat{r}}_j^{(0)^T} \ \hat{r}_j^{(L+1)}]^T$, where $\hat{r}_j^{(L+1)}$ is the desired score for image j given the input query $\underline{q}_{L+1}$; $c_j \Psi^{(1)^T} \underline{w}_j^{(0)} = [c_j (\Psi^{(0)^T} \underline{w}_j^{(0)})^T \ \ c_j \Phi^T(\underline{q}_{L+1}) \underline{w}_j^{(0)}]^T$ and $\underline{\hat{r}}_j^{(0)} = c_j \Psi^{(0)^T} \underline{w}_j^{(0)}$. This produces

$$[\underline{\hat{r}}_j^{(1)} - c_j \Psi^{(1)^T} \underline{w}_j^{(0)}] = [\underline{0}^T \ (\hat{r}_j^{(L+1)} - c_j \Phi^T(\underline{q}_{L+1}) \underline{w}_j^{(0)})]^T \tag{9.21}$$

The inverse of the Gram matrix $G_1 = [\Psi^{(1)^T} \Psi^{(1)}]$ at time t_1 can also be computed in terms of inverse of the Gram matrix $G_0 = [\Psi^{(0)^T} \Psi^{(0)}]$ at time t_0 and the new image $\underline{q}_{L+1}$ introduced at time t_1 as follows

$$\left[\Psi^{(1)^T} \Psi^{(1)}\right]^{-1} = \begin{pmatrix} \Psi^{(0)^T} \Psi^{(0)} & \Psi^{(0)^T} \Phi(\underline{q}_{L+1}) \\ \Phi^T(\underline{q}_{L+1}) \Psi^{(0)} & \Phi^T(\underline{q}_{L+1}) \Phi(\underline{q}_{L+1}) \end{pmatrix}^{-1} = \begin{pmatrix} G_0 & \underline{v} \\ \underline{v}^T & \beta \end{pmatrix}^{-1} \tag{9.22}$$

where vector $\underline{v}$ is defined as

$$\underline{v} = \Psi^{(0)^T} \Phi(\underline{q}_{L+1}) = [k(\underline{q}_1, \underline{q}_{L+1}) \ k(\underline{q}_2, \underline{q}_{L+1}) \ldots k(\underline{q}_L, \underline{q}_{L+1})]^T, \tag{9.23}$$

and scalar $\beta = \Phi^T(\underline{q}_{L+1}) \Phi(\underline{q}_{L+1}) = k(\underline{q}_{L+1}, \underline{q}_{L+1})$. Now, the inverse of the symmetric matrix expressed in the right hand side in (9.22) can be put in the form

$$\begin{pmatrix} G_0 & \underline{v} \\ \underline{v}^T & \beta \end{pmatrix}^{-1} = \frac{1}{\beta - \underline{v}^T G_0^{-1} \underline{v}} \begin{pmatrix} G_0^{-1}((\beta - \underline{v}^T G_0^{-1} \underline{v}) I + \underline{v}\underline{v}^T G_0^{-1}) & -G_0^{-1} \underline{v} \\ -\underline{v}^T G_0^{-1} & 1 \end{pmatrix} \tag{9.24}$$

Plugging (9.24) and (9.21) into (9.20) and making the necessary simplifications, the new scoring function $r_j^{(1)}(\underline{q})$ becomes

$$r_j^{(1)}(\underline{q}) = r_j^{(0)}(\underline{q}) - (\underline{v}^T G_0^{-1} \underline{v} - \beta)^{-1} \delta_j \left(k(\underline{q}, \underline{q}_{L+1}) - \sum_{i=1}^{L} k(\underline{q}, \underline{q}_i) z_i \right) \tag{9.25}$$

$$= r_j^{(0)}(\underline{q}) + \delta_j \frac{k(\underline{q}, \underline{q}_{L+1}) - \sum_{i=1}^{L} k(\underline{q}, \underline{q}_i) z_i}{k(\underline{q}_{L+1}, \underline{q}_{L+1}) - \sum_{i=1}^{L} k(\underline{q}, \underline{q}_i) z_i} \tag{9.26}$$

where $\delta_j = \hat{r}_j^{(L+1)} - r_j^{(0)}(\underline{q}_{L+1})$, with $r_j^{(0)}(\underline{q}_{L+1}) = c_j \Phi^T(\underline{q}_{L+1})\underline{w}_j^{(0)}$, represents the prediction error between the new score $\hat{r}_j^{(L+1)}$ assigned by the user and the predicted value $r_j^{(0)}(\underline{q}_{L+1})$ by the retrieval system and z_i is the i^{th} element of $\underline{z} = G_0^{-1}\underline{v}$, which represents the i^{th} coefficient when $k(\underline{q}, \underline{q}_{L+1})$ is approximated as a finite sum using kernel functions $\{k(\underline{q}, \underline{q}_1)\, k(\underline{q}, \underline{q}_2) \ldots k(\underline{q}, \underline{q}_L)\}$. To see this let us approximate $k(\underline{q}, \underline{q}_{L+1})$ as

$$k(\underline{q}, \underline{q}_{L+1}) \approx \sum_{i=1}^{L} k(\underline{q}, \underline{q}_i) z_i \tag{9.27}$$

and solve for z_i's. It can easily be shown that the solution for $\underline{z}$ is $G_0^{-1}\underline{v}$. Hence, the change in scoring function given by $r_j^{(1)}(\underline{q}) - r_j^{(0)}(\underline{q})$, see (9.25), is controlled by the product of the prediction error δ_j and the function-expansion error $(k(\underline{q}, \underline{q}_{L+1}) - \sum_{i=1}^{L} k(\underline{q}, \underline{q}_i) z_i)$ for the newly introduced kernel function $k(\underline{q}, \underline{q}_{L+1})$. Again, as can be seen, all the operations in this updating are carried out in the kernel domain.

Substituting $r_j^{(1)}(\underline{q})$ by $\underline{b}_j^{(1)}\mathcal{K}^{(1)}$ and $r_j^{(0)}(\underline{q})$ by $\underline{b}_j^{(0)}\mathcal{K}^{(0)}$ in (9.25) and solving for $\underline{b}_j$ from the resultant equation we arrive at the updating equation

$$\underline{b}_j^{(1)} = \begin{bmatrix} \underline{b}_j^{(0)} + (\underline{v}^T G_0^{-1}\underline{v} - \beta)^{-1}\delta_j G_0^{-1}\underline{v} \\ -(\underline{v}^T G_0^{-1}\underline{v} - \beta)^{-1}\delta_j \end{bmatrix} \tag{9.28}$$

$$\underline{b}_j^{(1)} = \begin{bmatrix} \underline{b}_j^{(0)} - \frac{1}{\beta - \underline{v}^T G_0^{-1}\underline{v}}\delta_j \underline{z} \\ \frac{1}{\beta - \underline{v}^T G_0^{-1}\underline{v}}\delta_j \end{bmatrix} \tag{9.29}$$

which expresses the new updated connection weight vector $\underline{b}_j^{(1)}$ for the j^{th} scoring function (see Figure 9.2) in terms of the old vector $\underline{b}_j^{(0)}$, which contains the old information, and the new sample $\underline{q}_{L+1}$, which carries the new information.

Note that for the initial network set up, where there is a single neuron in each pool j, the connection weight $b_j^{(0)} = 1$ and $G_0 = k(\underline{q}_1, \underline{q}_1)$ (see Fig. 9.2), i.e. scalars. As learning progresses, the inverse of the new Gram matrix can be computed using (9.24) without actually performing any matrix inversion process. This implies that (9.25) can be carried out without using computationally demanding matrix operations.

The next two remarks give some insight into the formulations on relevance feedback learning introduced throughout this section. The first remark studies the behavior of the scoring function for large values of the Gaussian kernel parameter a. Care must be taken if we try to apply this remark to other types of kernel functions as they may not be localized. The second remark is introduced to analyze the stability properties of our multiple regulator image retrieval system. This is an important issue, especially if our image retrieval system receives abundant feedback and we have to guarantee the good performance of our system for the previously learnt queries.

Remark 9.3 Using a Gaussian kernel $k(\underline{s},\underline{t}) = exp(-\frac{||\underline{s}-\underline{t}||^2}{2\sigma^2})$ and making parameter σ(standard deviation) approach zero, the kernel function reduces to the Kronecker delta, i.e. $k(\underline{s},\underline{t}) \rightarrow \delta(\underline{s}-\underline{t})$ and thus the scoring function (9.25) reduces to

$$r_j^{(1)}(\underline{q}) \approx \sum_{i=1}^{L+1} \hat{r}_j^{(i)} \delta(\underline{q}-\underline{q}_i) \tag{9.30}$$

The reasoning being when $\sigma \rightarrow 0$, $G_0 \rightarrow I$, $\underline{v} \rightarrow \underline{0}$, $\delta_j \rightarrow \hat{r}_j^{(L+1)}$, the predicted value $r_j^{(0)}(\underline{q}_{L+1}) \rightarrow 0$, $\beta \rightarrow 1$, and $\underline{z} \rightarrow \underline{0}$. Although simple, scoring function (9.30) seems to be attractive. However, its low prediction capability makes it of limited usage unless certain properties are satisfied. Consider the case when we have a large number of one dimensional and equally spaced queries, i.e. $\underline{q}_i = iT$ and $\underline{q} = t$, where T is a sampling period. Then, the output score $r_j^{(1)}(\underline{q})$ in (9.30) reduces to

$$r_j^{(1)}(t) \approx \sum_{i=-\infty}^{\infty} \hat{r}_j(iT)\delta(t-iT); \quad \hat{r}_j(iT) = \hat{r}_j^{(i)} \tag{9.31}$$

The score $r_j^{(1)}(t)$ in (9.31) is the well known sampled version [117] of the desired signal $\hat{r}_j(t)$.

Remark 9.4 To show stability property of our image retrieval system for maintaining previous knowledge while accumulating new information via relevance feedback learning, let us assume that the last learnt image query was $\underline{q}_{L+1}$. Then, according to (9.26) the new score for this query is $r_j^{(1)}(\underline{q}_{L+1}) = r_j^{(0)}(\underline{q}) + \delta_j \cdot 1$. However, given that

$\delta_j = \hat{r}_j^{(L+1)} - r_j^{(0)}(\underline{q}_{L+1})$ therefore $r_j^{(1)}(\underline{q}_{L+1}) = \hat{r}_j^{(L+1)}$. That is, the new information is incorporated into the scoring function. Now, to complete the proof we have to show that the previous information is retained for all previous queries $\underline{q}_k$, $k \leq L$. For simplicity let us modify the superscripts of (9.26) to yield

$$r_j^{(L+1)}(\underline{q}) = r_j^{(L)}(\underline{q}) + \delta_j \frac{k(\underline{q}, \underline{q}_{L+1}) - \sum_{i=1}^{L} k(\underline{q}, \underline{q}_i) z_i}{k(\underline{q}_{L+1}, \underline{q}_{L+1}) - \sum_{i=1}^{L} k(\underline{q}, \underline{q}_i) z_i} \tag{9.32}$$

where $r_j^{(L+1)}(\cdot)$ is the j^{th} scoring function at time $L+1$. If we substitute $\underline{q}$ by $\underline{q}_k$, $k \leq L$, (i.e. any previously applied queries) in (9.32) the numerator of the second term on the right-hand side becomes zero because $k(\underline{q}_k, \underline{q}_{L+1})$ is equal to $\sum_{i=1}^{L} k(\underline{q}_k, \underline{q}_i) z_i$ for queries $\underline{q} = \underline{q}_k$, when $k \leq L$ (this can be seen by comparing the k element of $\underline{v}$ with the corresponding element of $G_0 \underline{z}$). Therefore, $r_j^{(L+1)}(\underline{q}_k) = r_j^{(L)}(\underline{q}_k)$. If we proceed in this fashion then we arrive at the sequence $r_j^{(L)}(\underline{q}_k) = r_j^{(L-1)}(\underline{q}_k) = r_j^{(L-2)}(\underline{q}_k), \cdots,$ $= r_j^{(k)}(\underline{q}_k)$. Now, given that at time k the desired score $\underline{\hat{r}}^{(k)}$ is solicited to the user and our relevance feedback formulation enforces the reproduction of the score, then at time k we have $r_j^{(k)}(\underline{q}_k) = \hat{r}_j^{(k)}$. Now, as shown above this also holds at time L, i.e. $r_j^{(L)}(\underline{q}_k) = \hat{r}_j^{(k)}$ (the output score of the retrieval system at time L is that assigned by the user at time k).

In the case that kernel parameter a becomes large then the stability property can be proved directly by using (9.30) of Remark 9.3.

9.5 Selective Sampling using Fisher's Information Matrix

The idea behind selective sampling is to control the expansion of the pools by incrementally choosing those query samples that will bring the greatest information to the already established units in the pools. The introduced query will form a new unit with a kernel function centered at the query. This new unit brings old and new

information to the set of neurons in the pool. The 'old information' refers to part of the information in the new neuron that can be deduced from the other old neurons while the 'new information' cannot be extracted from them.

It is known [118] that the best linear estimator $\underline{\theta}$ for the linear model

$$y(x) = \underline{\theta}^T \underline{\mathrm{f}}(x) + \epsilon(x) \tag{9.33}$$

with basis functions $\underline{\mathrm{f}}(x) = [f_1(x), f_2(x), \cdots, f_L(x)]^T$ given the observations $y(x_i) = y_i$, and errors $\epsilon(x_i)$, $i \in [1, L]$, with variances $\sigma_1^2, \sigma_2^2, \cdots, \sigma_L^2$, can be expressed as [118]

$$\underline{\theta} = M^{-1} \underline{\mathrm{y}} \tag{9.34}$$

where M is the Fisher information matrix defined as

$$M = \sum_{i=1}^{L} \underline{\mathrm{f}}(x_i) \underline{\mathrm{f}}^T(x_i) / \sigma_i^2 \tag{9.35}$$

and $\underline{\mathrm{y}} = \sum_{i=1}^{L} y_i \underline{\mathrm{f}}(x_i)/\sigma_i^2$. If we assume equal standard deviations, ie. $\sigma = \sigma_1 = \sigma_2 = \cdots = \sigma_L$, then the best estimator does not depend on σ_i's and the information matrix M can be written as $M = \sum_{i=1}^{L} \underline{\mathrm{f}}(x_i) \underline{\mathrm{f}}^T(x_i)/\sigma^2$.

Comparing the deterministic part of the linear model (9.33) to the scoring function (9.17) we see that the associated Fisher matrix M for the L neurons in pool j is given as

$$M = \frac{1}{\sigma^2} G_0^T G_0 \tag{9.36}$$

$$= \frac{1}{\sigma^2} G_0^2 \tag{9.37}$$

where $G_0 = \Psi^{(0)^T} \Psi^{(0)}$ is the symmetric Gram matrix defined as in Section 9.4.

It is also known [118] that the covariance matrix associated with the best linear estimator is the inverse of the Fisher information matrix. In the design of experiments it is convenient to minimize the determinant or the norm of the covariance matrix

[119]. This is equivalent to maximizing the logarithm of the determinant of the Fisher information matrix. Let p_L be the logarithm of the determinant of M_0. That is

$$p_L = logM_0 = 2L \log|\sigma| + 2 \log det(G_0) \tag{9.38}$$

where $|\cdot|$ denotes the absolute value and L is the number of neurons in pool j.

Suppose that at time t_1 a new sample $\underline{q}_{L+1}$ is introduced to the pool. Then p_{L+1}, which is the logarithm of the new information matrix M_1, is

$$p_{L+1} = \log M_1 = 2(L+1) \log|\sigma| + 2 \log det(G_1) \tag{9.39}$$

The difference between p_{L+1} and p_L is given by $p_{L+1} - p_L = 2log|\sigma| + 2(\log det(G_1) - \log det(G_0))$. We can see that what controls the gain in information is the difference $(\log det(G_1) - \log det(G_0))$ because σ is assumed to be a constant. If we define g_{L+1} as this difference, i.e.

$$g_{L+1} = \log det(G_1) - \log det(G_0) \tag{9.40}$$

we can find some relationship between this gain and the expression for the updating equation (9.25) for the scoring function using relevance feedback. Let us first reduce (9.40) using the expression for G_1 and some of its properties.

$$\begin{aligned} \begin{bmatrix} G_0 & \underline{v} \\ \underline{v}^T & \beta \end{bmatrix} &= \begin{bmatrix} G_0^{1/2} & \underline{0} \\ \left[G_0^{-1/2}\underline{v}\right]^T & [\beta - \underline{v}^T G_0^{-1}\underline{v}]^{1/2} \end{bmatrix} \begin{bmatrix} G_0^{1/2} & G_0^{-1/2}\underline{v} \\ \underline{0}^T & [\beta - \underline{v}^T G_0^{-1}\underline{v}]^{1/2} \end{bmatrix} \\ &= [\beta - \underline{v}^T G_0^{-1}\underline{v}][G_0] \end{aligned} \tag{9.41}$$

where $[\cdot]$ stands for the determinant. Then, the information gain g_{L+1} when introducing sample $\underline{q}_{L+1}$ can be simplified to

$$\begin{aligned} g_{L+1} &= \log|\beta - \underline{v}^T G_0^{-1}\underline{v}| \\ &= \log\left|k(\underline{q}_{L+1}, \underline{q}_{L+1}) - \sum_{i=1}^{L} k(\underline{q}_{L+1}, \underline{q}_i) z_i\right| \end{aligned} \tag{9.42}$$

As can be seen in the scoring function (9.25) the factor $\beta - \underline{v}^T G_0^{-1}\underline{v}$ is the denominator of the adjustment term added to the previous scoring function. Having a large value for this term ensures a small increment for the scoring function.

A geometrical interpretation of the gain $k(\underline{q}_{L+1}, \underline{q}_{L+1}) - \sum_{i=1}^{L} k(\underline{q}_{L+1}, \underline{q}_i) z_i$ can be obtained by using the mapping functions $\Phi(\cdot)$'s. For illustrative purposes, let us assume that the number of neurons in a pool is $L = 2$ and that the new query to be introduced into the pool is $\underline{q}_3$.

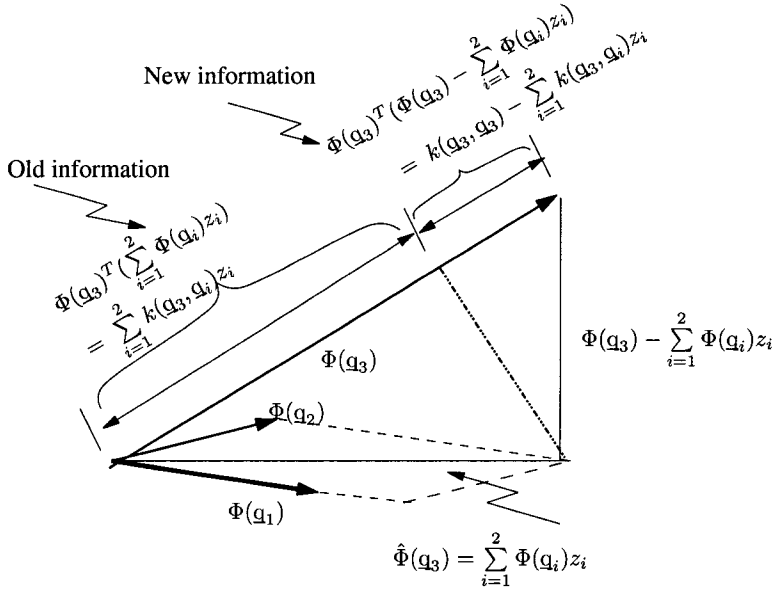


Figure 9.3: Old and new information of a new query sample.

Figure 9.3 shows a geometrical interpretation for the information that a new query carries. The information of the new query $\underline{q}_3$ consists of old information $(\sum_{i=1}^{2} k(\underline{q}_3, \underline{q}_i)z_i)$, which can be deduced from the other queries, and the new information $(k(\underline{q}_3, \underline{q}_3) - \sum_{i=1}^{2} k(\underline{q}_3, \underline{q}_i)z_i)$. Now, careful investigation of the updating equation (9.29) for the weights connections reveals that the inverse of the new information has a penalizing effect for the pool. That is, when the new information is little, the pool receives more penalty, which is reflected by a large value of the new weight connection and large increments for the already established connections. Though at first it may seem illogical, this could be understandable. Clearly, if the sample carries little new information i.e. most of the information of the sample can be deduced from the

other neurons then the penalty for having received relevance feedback (not having predicted the desired score) and expanding the pool should be greater. Also, from (9.28) we can see that the penalty is distributed amongst the preexisting units in the pool. That is, the units that are more responsible for predicting the old information of the new query are penalized more. We can see that what controls the increment for the existing connections of the neurons in the pool is the ratio of the old information to the new information.

The methodology of selective sampling is simple: (i) select from the training set the unmarked query sample that carries the most new information and apply this (most informative) query as a submitted query; (ii) apply relevance feedback, if needed, to the displayed images; (iii) mark the query sample; and (iv) finish the process if there is no more unmarked query image, otherwise repeat the process from step "i".

9.6 Experimental Results

To assess the performance of the proposed multiple regulator retrieval system several experiments are conducted on a domain-specific image database, namely the underwater target described in Chapter 8, where high retrieval accuracy is of utmost importance. The database consists of a relatively large electro-optical imagery collected from 12 different underwater mine-like and 28 non-mine-like objects [111], i.e. the different number of potential categories in this database is 40. The training and testing sets consist of 351 and 234 feature vectors in a 18-dimension space containing textural and shape related information. The experiments to collect the images and the description of the types of underwater objects are described in details in [111]. A detail description of the image database and the feature extraction methods used to form the feature vectors can be found in Chapter 8. For all our experiments a Gaussian kernel function with parameter $a = 1$ is used.

9.6.1 User Interface

We first briefly describe the user interface (UI) that is specifically designed for the multiple regulator image retrieval system as shown in Figure 9.4. In designing the user interface it is of great importance to offer the user with versatile and user friendly interface under which he/she can provide relevance feedback if needed. In our designed UI, the users can browse, select some query image, and apply, at will, relevance feedback on the displayed images. The browsing process in our UI is easily implemented by moving the scrolling bar that appears at the small window located in the upper left part of the interface as shown in 9.4. The selected query image of a bullet-shape mine-like object from the database described in Chapter 8 is seen in the upper left side of the UI. Figure 9.4(a) shows the top nine retrieved images and their associated scores for this submitted query using the initially trained system with one neuron in each pool. As can be seen, the top three images correspond to the same object type as that of the submitted image. However, there are also images corresponding to different object types, namely two images of another bullet-shape object shown on the first and third columns in the second row and two images of a non-mine like object shown on the bottom row. Clearly, it is desirable to have all the relevant images (same object type) at the top of the list, followed by the less relevant images. To this aim the relevance feedback methodology developed in this chapter will be used, as we will shown later.

9.6.2 Relevance Feedback Learning

In this section the relevance feedback learning methodology developed in Section 9.4 will be tested in conjunction with the selective sampling method described in Section 9.5. Selective sampling is first applied to select from the training queries the "best" image query that will bring the maximum new information to one of the pools. Relevance feedback learning is then applied to adjust the connections of the old

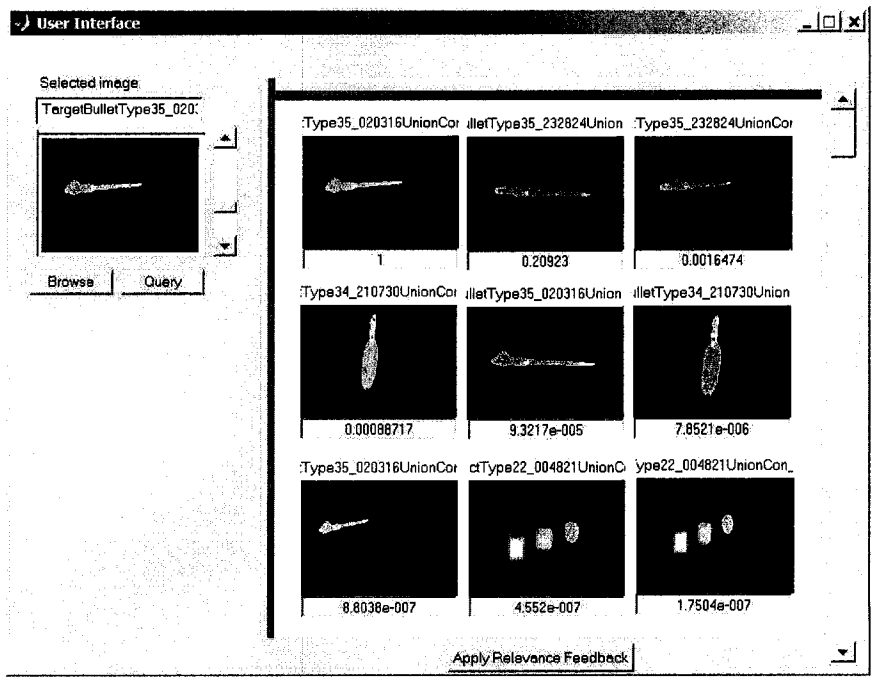


Figure 9.4: User Interface for the Multiple Regulator Image Retrieval System and Retrieval Results

neurons and set up the new connections of the recently introduced neuron. Several experiments are conducted that are described next.

Experiment 1: The purpose of this experiment is to asses the performance of relevance feedback learning and the complexity of the model reference image retrieval system in terms of the size of the pools when the 'desired' scores of the misplaced relevant images are computed from the scores of other images. A two-layer neural network is initially constructed using the initial start up procedure mentioned in Section 9.2. For every image query in the 351-sample training database, an output neuron provides a score. The training phase uses the information embedded in the training queries with the purpose of avoiding a second or possibly more rounds of relevance feedback learning where all training queries should be presented again.

The training phase basically involves these four major steps: (i) choose from the training set, an unsubmitted query image that will bring the maximum information to

one of the pools using the methodology derived from the Fisher information matrix (see Section 9.5). (ii) Submit this query image is submitted to the MRIRS and relevance feedback is subsequently applied to the misplaced relevant images. To estimate the 'desired' scores, from the displayed images find the relevant image that has lowest score among all the relevant images that are not misplaced, i.e. there is no non-relevant images above the image. Let r_1 be the score of this relevant image and r_0 be the score of the non-relevant image that is immediately below it. The specified desired scores of the misplaced images are chosen to be between r_0 and r_1. (iii) Once the 'desired' scores are found each one of the pools corresponding to the misplaced images are updated using the error between the desired score and the actual value. This will lead to the expansion of the pool and the updating of the old connections as explained in Section 9.4. (iv) mark the submitted query and if there is a non-marked query then repeats the steps from (i) otherwise the training process is completed.

To evaluate the performance of the relevance feedback learning methodology the recall plots (see Figure 9.5) of the training and testing sets are generated during the relevance feedback learning using every image query (351 queries). A recall value of '1' indicates that all the relevant images appear at the top of the retrieved list while a value of '0' is representative of the fact only non-relevant images are on the top.

As can be seen from Figure 9.5, after starting from a rather low value around 0.45 corresponding to initial start up using the projection matrix as in Section 9.2, the recall measure for both the training set or testing sets steadily increases reaching almost '1' and 0.9, for the training and testing sets, respectively. This demonstrates two aspects of the proposed relevance feedback methodology for the MRIRS. These are: (i) the good performance of the MRIRS system for the training set which is attested by the fact that in one pass the recall plot almost reaches '1' (0.99); and (ii) the generalization capability of the system to respond adequately to unseen queries, which is illustrated by the high value (0.90) of the recall measure on the testing set

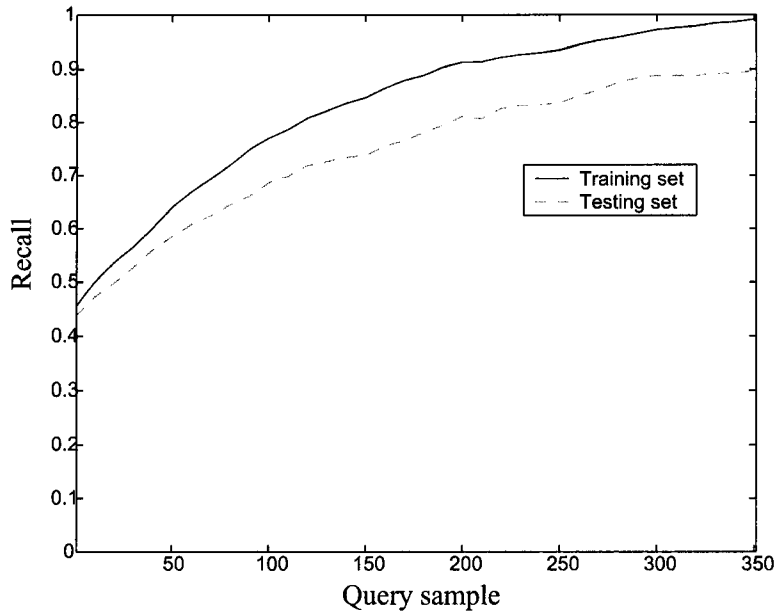


Figure 9.5: Recall measure versus the training query sample for Experiment 1.

after learning is completed.

Next, we study the complexity of our MRIRS system in terms of the size of the pools and also examine the possibility of implementing our retrieval system on larger image databases. Figure 9.6 shows the histogram of the number of pools (vertical axis) versus the pool size (horizontal axis). As can be seen from this histogram plot, the minimum pool size is 1 corresponding to the the start up of the system while the maximum size is 11. There are 9 pools of size 1 that never received relevance feedback, implying that their predicted scores for the set of training queries were closer to the desired values and hence the pools were not expanded. On the other hand, the pools of size 11 (3 of them) were the least reliable and the most stimulated ones during the relevance feedback learning as they did not predicted the desired score almost every time that an image query of the same category was submitted. As a result, it was necessary to apply relevance feedback to these pools. In Figure 9.6, the average size of a pool is found to be 5, i.e. on overage a pool receives relevance feedback 4 times during the training process.

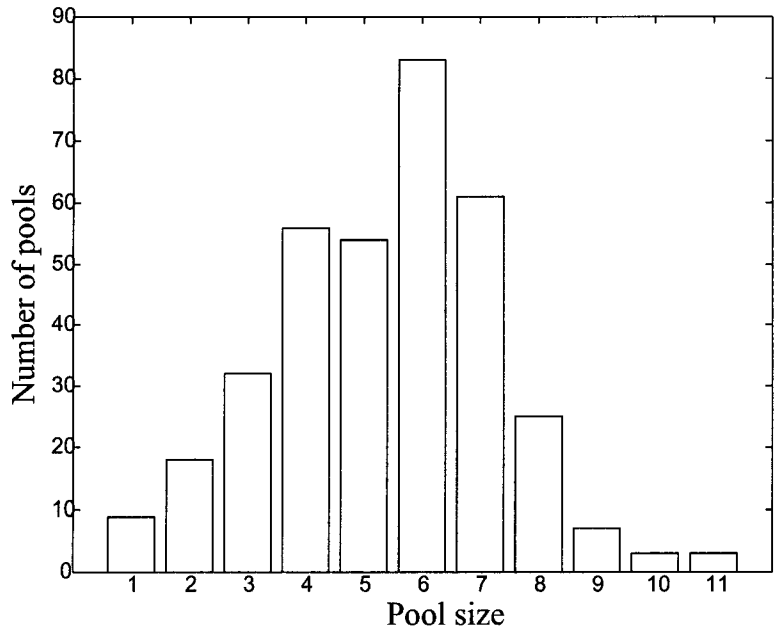


Figure 9.6: Histogram of size of the pools. The vertical axis represents the number of pools of a particular size (horizontal axis).

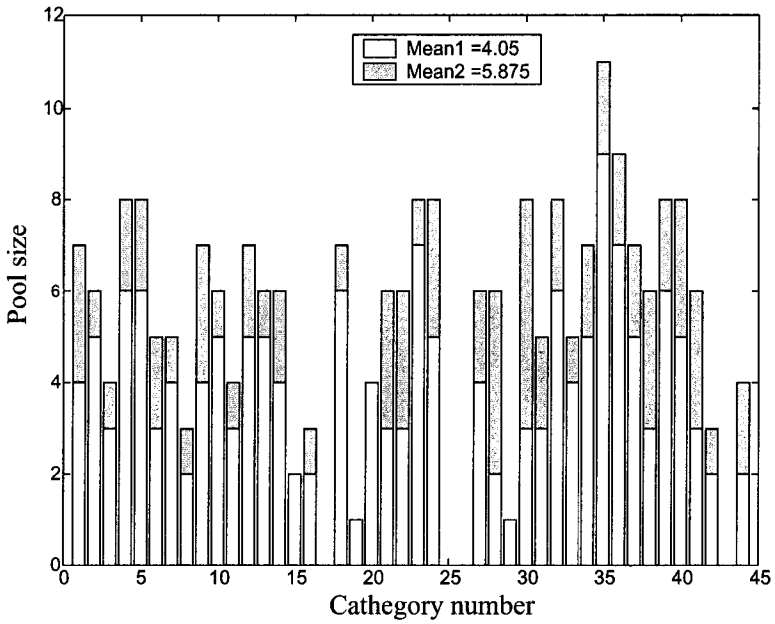
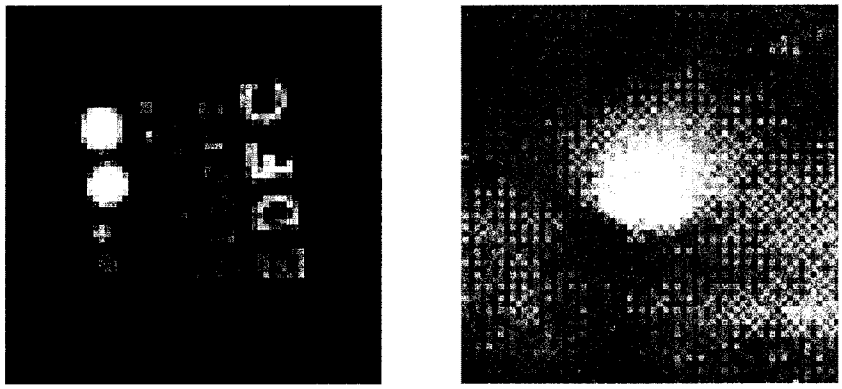


Figure 9.7: Minimum and maximum pool sizes for the different object categories.

Our image database consists of 40 different categories of mine-like and non-mine like objects (see Chapter 8 for the description of these categories). There are images

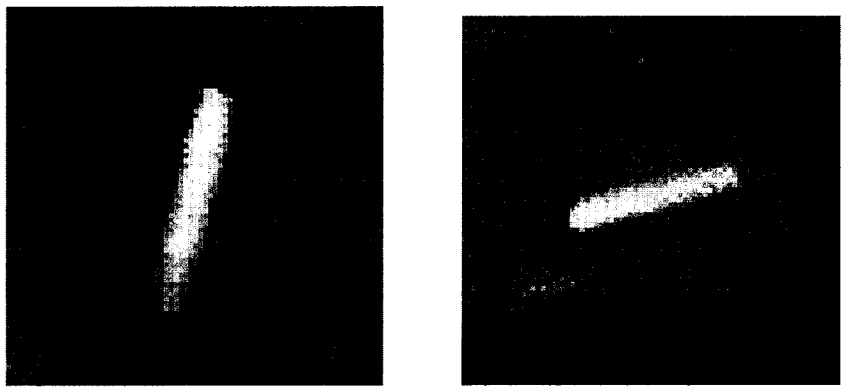
in some categories that look very similar to images in other categories, while there are also images in some categories that look very different to other images of any other category. Consequently, one would expect that those pools associated with the images that are visually similar but belong to different category will receive more relevance feedback than those of the images that do not possess similar visual appearance. Figure 9.7 shows the minimum and maximum pool sizes versus the category number. For instance, for Category 21 (a three dimensional panel) the minimum pool size is 3 and the maximum pool size is 6. Categories that do not contain a vertical bar are not considered in our study, leaving the total number of non-empty categories 40. The mine-like objects comprise categories 28-39 while the rest are non mine-like objects. As can be seen, the pools corresponding to Categories 19 and 29 are very reliable as they do not receive relevance feedback. It can also be argued that the textural and shape dependent features used here are more suitable to distinguish these types of images from the rest. This suggests that if we choose a good set of features then relevance feedback learning could impact the formed pools less. Figures 9.8(a) and (d) show a non-mine like (panel) object corresponding to Category 19 and a mine-like object corresponding to Category 29, respectively. While it is somehow clear why the well-formed panel object with white letters and defined solid circles shown in Figure 9.8(a) is a distinguishable object, it is somewhat baffling why the fuzzy mine-like object shown in Figure 9.8(b) is distinguishable!. However, the reason may be the fuzzy nature of this object that is ultimately embedded in the textural features given that there is no other object in the database with such fuzziness. Now, we shall present two categories not very reliable for retrieval purposes. Figures 9.9(a) and (b) show two images corresponding to Categories 36 and 37, respectively. These two categories correspond to cylindrical shape objects having slight differences at their tips. They are so difficult to distinguish, even for an expert working with this image database for some time. Thus, it is not a surprise that the maximum pool sizes (as

shown in Figure 9.7) for the Categories 36 and 37 are 9 and 7, respectively.



(a) Category 19 (b) Category 29

Figure 9.8: Two categories more reliable for retrieval purposes.



(a) Category 36 (b) Category 37

Figure 9.9: Two categories not very reliable for retrieval purposes.

As mentioned before, the average size of a pool is 5 which implies that the average number of applications of relevance feedback is 4. To close this experiment we will present an object category that needs precisely 4 rounds of relevance feedback in order to incorporate the users' expertise. Figure 9.10 shows the user interface after 4 rounds of relevance feedback have been applied to the initially trained system in Figure 9.4. During each round of feedback the user selects a misplaced relevant

image and chooses a desired score, then the relevance feedback learning methodology is applied to adjust the old connections and set up the new connection. As can be seen from Figure 9.10, all other relevant and non-relevant images remain at their places while relevance feedback is being applied, demonstrating experimentally the stability of our multiple regulator image retrieval system in incorporating new information while keeping the old one.

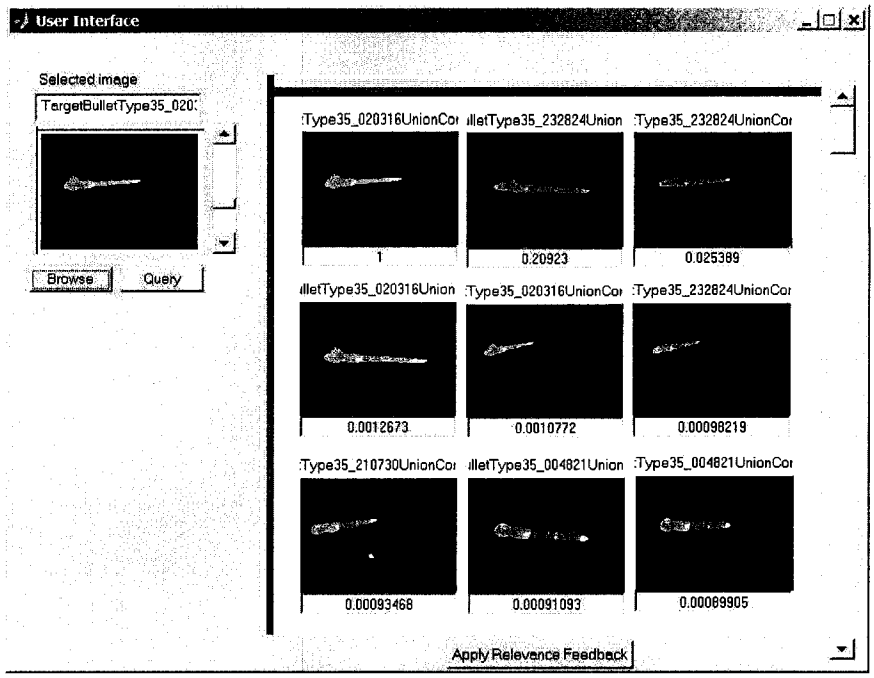


Figure 9.10: Retrieval results of Figure 9.5 after 4 Rounds of Relevance Feedback.

Experiment 2: The purpose of this experiment is to bring the recall measure to the maximum even though the complexity of the model reference image retrieval system might be increased. This experiment might be suitable for special-purpose image database where some kind of information to categorize the images can be intuitively used by the expert users, contrary to Experiment 1 suitable for general-purpose databases where this information might not be available. Relevance feedback learning is applied based upon the criterion that the relevant images should have scores above a pre-specified threshold (0.9 in our experiment). As in the previous experiment the

neural network is initially constructed using the initial start up procedure in Section 9.2 and similarly relevance feedback learning is carried out using all training queries in one pass fashion where the most informative query (according to the methodology presented at the end of Section 9.5) is submitted first, followed by the second most informative, and so on. If the score of a relevant image falls below the threshold then an error signal that is the difference between the threshold and the actual score of the relevant image is generated to stimulate the appropriate pool, update the old connections, and assign a weight to the new connection.

Figures 9.11 and 9.12 show the recall measure reported during the relevance feedback learning process and the histogram of the number of pools versus their size reported at the end of the training, respectively. Immediately, one can identify three major differences between the recall plot for this experiment and that of Experiment 1. First, the recall plot of the testing test reaches a higher value (≈ 0.98) when the learning is completed. Second, the recall plot on the testing set continues increasing even after the recall on the training set has reached its maximum value of '1' around query sample number 250. The reason is obvious, since for this experiment relevance feedback is applied based upon the score of the relevant images rather than the relative positions with respect to the non-relevant images. Moreover, the recall plot at the initial stages of learning shows a higher slope than that for Experiment '1', implying intuitively that the prediction capabilities of the pools of neurons are better for this experiment than for the previous one. The high recall values reported in this experiment suggests a methodology that could be exploited when class information is available for the image database and high performance is of great requirement, even though this implies sacrificing the overall complexity as we shall see next.

Figure 9.12 shows the histogram of the number of pools for Experiment 2, where this more "aggressive policy" of relevance feedback takes place. As can be seen the average size of the pools is substantially increased (compared to that in Experiment

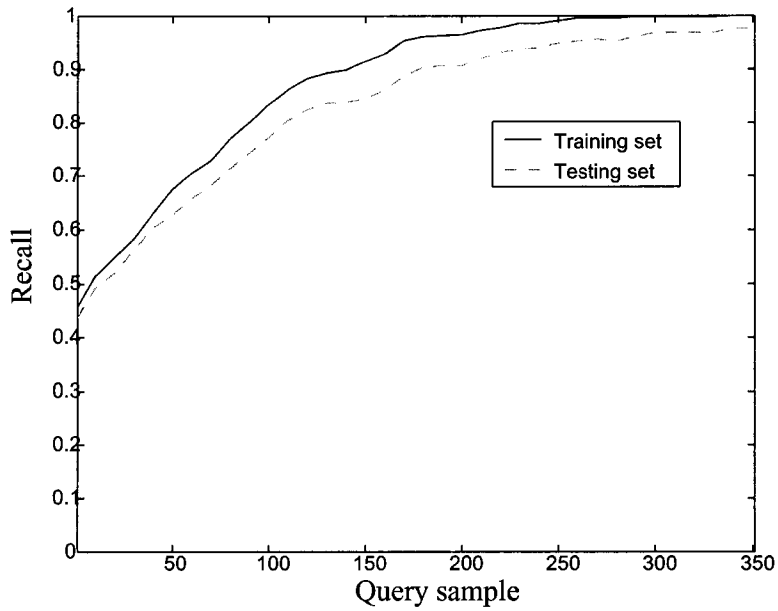


Figure 9.11: Recall measure versus the training query sample for Experiment 2.

1) to almost 9 neurons per pool, which is the maximum capacity of the pools. This is expected in light of the fact that the relevance feedback policy assigns scores greater that 0.9 to those relevant images that are below 0.9, regardless of their visual appearance to other images as in Experiment 1. Clearly, there are still pools with few neurons, e.g. 6 pools of size 1, 5 pools of size 2, and 11 pools of size 3, that are very reliable and could easily represent the respective category.

Experiment 3: Effect of the parameter a of the kernel function $k(\underline{x}, \underline{y}) = e^{-a\|\underline{x}-\underline{y}\|^2}$ on the stability and the generalization capability of our MRIRS system is studied in this experiment. The purpose is to highlight the importance of a good selection of parameter a in the design of the kernel function. In essence this experiment is a recreation of Experiment 2 but with the minor difference that the recall plots for both the training and testing sets are presented for various choices of parameter a of the kernel function.

The recall plots for the training and testing sets for four values of parameter a are shown in Figures 9.13(a)-(d). Two important properties are readily noticeable

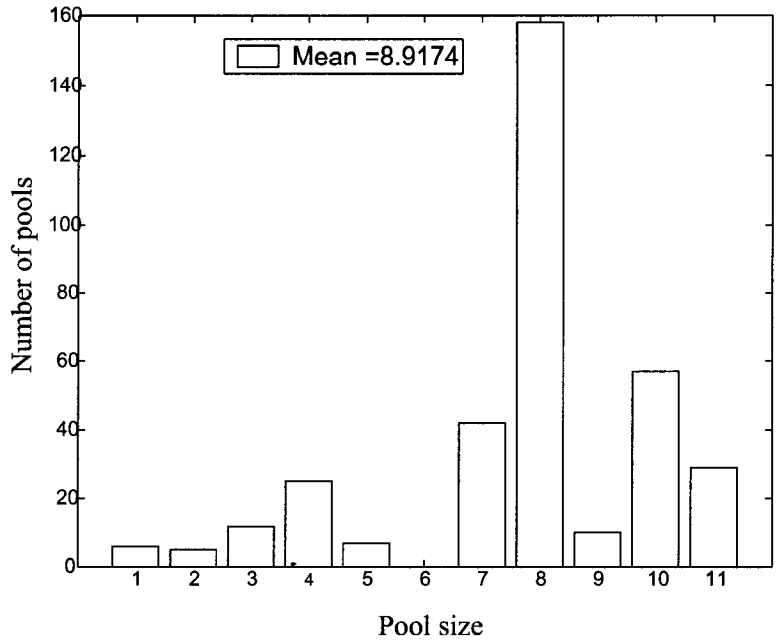
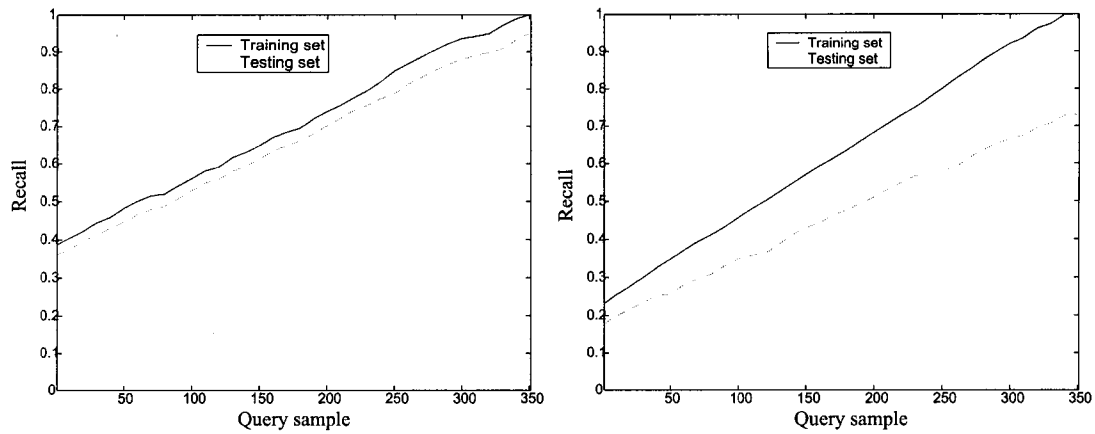


Figure 9.12: Histogram of size of the pools for Experiment 2.

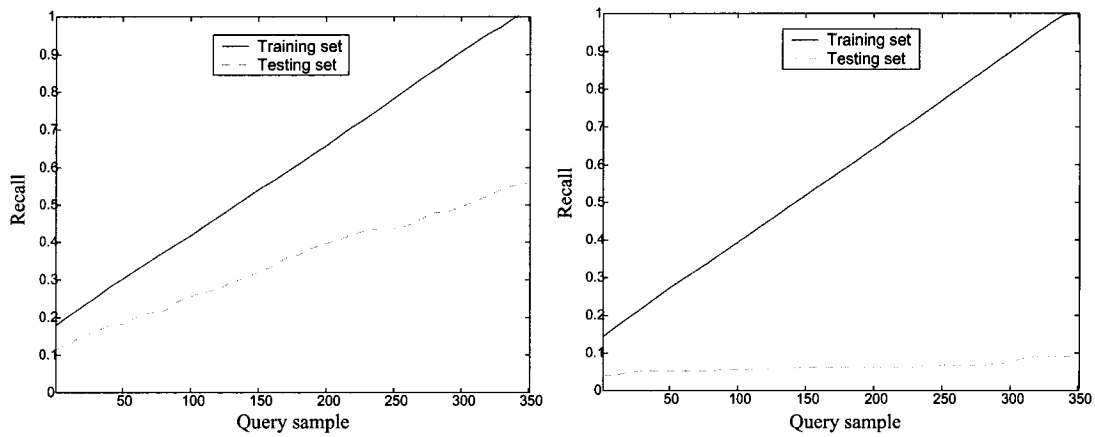
in these plots. First, the recall plot for the training set is almost a straight line that always reaches the perfect recall value of '1' at the end of the training process. This, experimentally validates the derivations in Remark 9.3 and proves the stability of our image retrieval system even for large values of parameter a, e.g. 1000 and 5000. Second, the recall capacity of the image retrieval system for the initial start up (at the start of the plot) for either the training or testing set is deteriorated as parameter a increases. For $a = 5000$, the initial recall for the testing set approaches to a value around 0.034 and the recall measure almost remains at that value throughout the entire training process, confirming somewhat the statement made in Remark 9.3 about the low prediction capability (which is related to generalization capability) of the system for large values of parameter a.

9.6.3 Model Reference Learning

In this section we will focus our attention in one experiment to assess the model reference learning methodology described in Section 9.3. In certain form, this experiment



(a) $a = 100$ (b) $a = 500$



(c) $a = 1000$ (d) $a = 5000$

Figure 9.13: Recall measure for large values of kernel's parameter a.

is the counterpart of Experiment 2 of the previous section but for the model reference learning. The training and testing sets are those of the previous experiments, the kernel parameter is again $a = 1$, and the performance measure is the recall. The differences between this experiment and Experiment 2 are basically two things: (i) model reference learning is implemented in batch mode after all the input-output relationships of the training queries have been captured while relevance feedback learning is implemented online by processing each training query as soon as it arrives; (ii) the most informative query sample derived from the Fisher information matrix is not used in this learning mode given the batch nature of the learning.

Experiment: Here we study the model reference learning introduced in Section 9.3 for the case when the class labels of the training samples are known and used to estimate the desired score. The purpose of model reference learning is to train each one of the pools independently by first collecting all the input-output relationships for a set of training queries. To this aim, all the training queries should be submitted to the retrieval system operating in a search and retrieval mode. At the time of submitting a query image if the score of some relevant image is less than the threshold 0.9 then its new score is set to 0.9 otherwise its score is kept unchanged. This input-output relationship is stored in model-reference database until all the training image queries have been processed. Then, the input-output information for each pool is obtained and used to set up the connections of that particular pool.

The recall plot for the model reference learning mode is shown in Figure 9.14. As can be observed, the recall measure for the training set reaches a perfect one while the recall plot for the testing set reaches a high value of 0.95. Though this final value is excellent it is somewhat inferior to that of Experiment 2 using relevance feedback learning. This might be attributed to the Fisher information matrix used in the latter case, which presents the most informative query samples during the relevance feedback learning mode in Experiment 2. Also note that rate of learning is fixed throughout the training as evident by almost a straight line curve.

The histogram of size of the pools after the model reference learning process is completed is shown in Figure 9.15. As can be seen the average size is substantially higher than that reported for the Experiment 2 on relevance feedback learning. In addition there are only a few smaller pools, in contrast to Experiment 2 where there were a lot of pools with few neurons. However, even with this increase in complexity the retrieval system for this experiment does not beat the performance of the retrieval system of Experiment 2 in terms of generalization capability, as evident from the recall plots for the testing set of the two experiments.

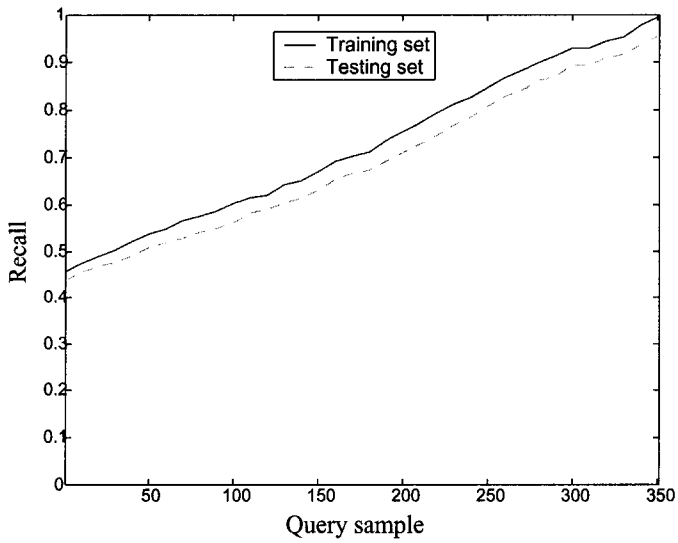


Figure 9.14: Recall plot for the model reference learning mode.

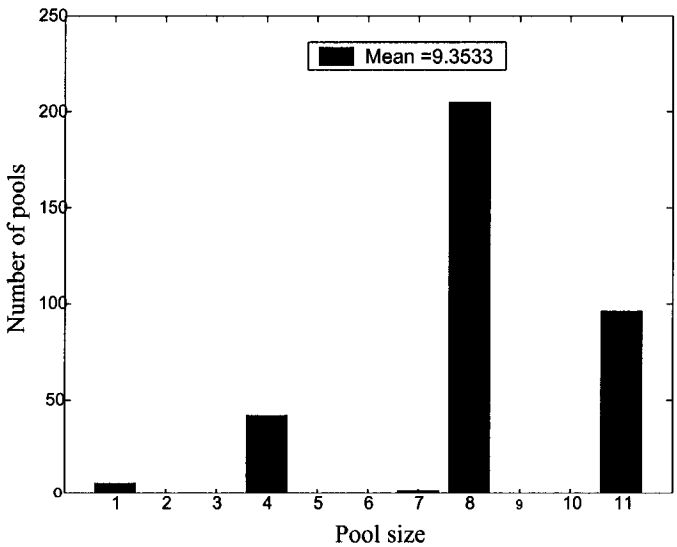


Figure 9.15: Histogram of size of the pools for the model reference learning mode.

9.7 Conclusions

In this chapter, we introduced a multiple regulator image retrieval system (MRIRS) that contains four components, namely a plant, multiple regulators, an adjustment mechanism, and a reference model (if available). A practical kernel machine (neural network) for simple implementation of this system is also proposed. In this two layer network, the first layer, which is structurally dynamic, contains a set of neuron pools while the second layer (structurally static) contains a set of output neurons, each associated with a particular image in the database.

Using constrained optimization framework we developed two learning mechanisms to incorporate information from either the expert users or a model reference. The model reference could be an external image retrieval system or an artificially created one using some a prior information about the class membership of the images in the database. Both model reference and relevance feedback learning mechanisms lead to an expansion of the pools of the first layer. The expansion of the pool is achieved by adding a neuron and creating a connection between this neuron and the corresponding neuron in the second (retrieval) layer. The incorporation of users' expertise in the relevance feedback learning is carried out in an online fashion while the incorporation of the model reference information is performed in batch mode. As in our text retrieval system the projection matrix can be used to set up the initial system.

To facilitate the incorporation of user's information during relevance feedback learning a new scheme based upon the Fisher information matrix was developed. Using this scheme the most informative image query is suggested to the user, and finally introduced into one or more pools. To assess the assimilation of users expertise into the retrieval system, three experiments on relevance feedback were conducted. In the first experiment a "relaxed policy" is used for estimating the 'desired scores' of the relevant images based upon their relative positions assigned by the user. The recall

plot of the training set attests to the performance of the system as it reaches a perfect 1 at the end of the relevance feedback learning. The recall plot on the testing set also indicates a high generalization capability of the system as this measure reaches 0.95 at the end of the training. The second experiment uses a more "strict policy" for estimating the "desired scores" based upon assigning a high score to all relevant images. This experiment leads to the best relevance feedback learning results in terms of generalization capability given that at the end of the learning process the recall plot for the proposed image retrieval system reached a value of 0.98. The third experiment was created to study the generalization capability and the performance of the image retrieval system as a function of different values of the Gaussian kernel parameter. It was found that the generalization capability is deteriorated for large values of the parameter. As far as final complexity of the system is concerned the best retrieval system was that of Experiment 1, which had a more relaxed policy to assigned the "desired scores", given that the average size of the pools at the end of the learning was 5, while for Experiment 2 the average size was around 9 neurons per pool.

An experiment on model reference learning was also conducted to evaluate the complexity, generalization capability, and response of the multiple regulator image retrieval system to already learnt query images. This experiment revealed that the image retrieval system exhibited a good generalization at the end of the model-reference learning as the recall measure on the testing set approached 0.95. However, the experiment also showed that most of the pools were almost at their maximum capacity (at 9 neurons/pool), hence increasing the overall complexity for practical implementation. Nonetheless, having an average of 9 neurons per pool does not represent a burden neither in terms of computational time nor in terms of memory requirements.

CHAPTER 10

A SINGLE-REGULATOR IMAGE RETRIEVAL SYSTEM

10.1 Introduction

As in the multiple-regulator system in the previous chapter, the fundamental parts of our proposed single-regulator model-reference image retrieval system as shown in Figure 7.2 are essentially the same. It generates an optimal query that will enhance the retrieval process. The optimal query is viewed with the same perspective by all elements in the plant (by all images), in contrast to the multiple-regulator system where a mapped query is only viewed by a particular image. The single-regulator system has three phases of learning. These are: (1) initial start up, (2) model-reference, and (3) relevance feedback learning. All these phases aim at finding or adjusting the parameters of the regulator. The initial start up deals with the methodology for setting up the parameters under the assumption that we only know the set of images. An interesting result that will be shown latter in this chapter is that the matrix associated to the regulator is the projection matrix of the set of images.

Model reference learning adjusts the parameters of the regulator using the knowledge from an external model reference retrieval system or via a model-reference training database as defined in Chapter 7. On the other hand, relevance feedback learning

is the methodology to incorporate user's expertise either online or in batch mode. Provisions should be taken if relevance feedback learning is to be applied online, because a large computational complexity could be an impediment for its implementation.

The organization of this chapter is as follows. Section 10.2 presents the initial start up process for the proposed single regulator IRS. Section 10.3 develops the concepts for learning from either an external model reference or an artificially generated one via a model-reference training set. Section 10.4 presents the relevance feedback learning methodology that captures user's expertise based upon a set of desired scores for the relevant images. Section 10.5 shows the experimental results on relevance feedback learning. Finally, Section 10.6 presents the conclusions and remarks.

10.2 Initial Start Up

For the single-regulator retrieval system shown in Figure 7.2, the retrieval status vector

$$\underline{s}(\underline{q}) = \Psi^T(X)A\Phi(\underline{q}). \tag{10.1}$$

Assuming that mapping functions are linear, the scoring function (10.1) is simplified to $\underline{s}(\underline{q}) = X^T A\underline{q}$. Again, as discussed before, for setting an initial value for A we have two options: (1) make $A = I_{M\times M}$, where M is the dimension of the space on which the input query $\underline{q}$ and the feature vectors in the database lie, or (2) make $A = P_X = X(X^TX)^{-1}X^T$. The best option depends on the dimension M and the number N of feature vectors in the database. In a typical case, in which $M \ll N$, only the first option is practical because the inverse of X^TX may not exist.

However, for the nonlinearly mapped feature space generated through some PD kernel induced function [113], setting A to the identity matrix will produce a high dimensional matrix. In contrast, $A = P_X = \Psi(X)[\Psi^T(X)\Psi(X)]^{-1}\Psi^T(X)$, will produce a matrix with rank $N \ll M'$, where M' is the dimension of the mapped feature space. Clearly, in this case the first choice is not a viable one.

Remark 10.1 In this single-regulator image retrieval system, the projection matrix $P_X = \Psi(X)\,[\Psi^T(X)\,\Psi(X)\,]^{-1}\Psi^T(X)$ associated with the set of images has eigenvalues of '1' with multiplicity N and N eigenvectors $\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_N$, each one is equal to a mapped image, i.e. $\underline{v}_i = \Phi(\underline{x}_j)$, $j \in [1, N]$. We assume that the kernel is PD and the mapped images are linearly independent. Therefore, the ranks of $\Psi^T(X)\Psi(X)$ and the projection matrix P_X are both equal to N. If we express the i^{th} mapped image vector in terms of the mapping matrix $\Psi(X)$ and the basis vector $\underline{e}_j = [0\,0 \ldots 0\,1\,0 \ldots 0]^T$, where its j^{th} element is '1' and the rest are '0', i.e. $\Phi(\underline{x}_j) = \Psi(X)\underline{e}_j$, it is easy to verify that the vector $\underline{v}_j = \Psi(X)\underline{e}_j$ is an eigenvector of P_X and its corresponding eigenvalue λ_i is '1'. This can be shown by substituting $\underline{v}_j$ and λ_j into $P_X\underline{v}_j = \lambda_j\underline{v}_j$ and verifying that the equation holds.

10.3 Model Reference Learning

The model-reference learning mechanism involves finding the parameters in the regulator in Figure 7.2 given a set of samples captured in a model-reference training database. In each iteration, all the samples are submitted as queries and an error signal is generated by comparing the actual output of the retrieval system against that of the reference model. This error signal is used to create our model-reference learning framework for the single-regulator system. This will be described next.

The problem of finding parameter matrix A in Figure 7.2 given a set of $(\underline{q}_i, \hat{\underline{s}}_i)$, $i \in [1\ L]$, training pairs can be cast as an optimization framework where we would like to find A^* such that

$$A^* = arg\ \min_A \mathcal{T}(A),$$

where $\mathcal{T}(A)$ is the Tikhonov function [116] defined as follows.

$$\mathcal{T}(A) = \frac{1}{2L}\sum_{i=1}^{L} ||\hat{\underline{s}}_i - \Psi^T(X)A\Phi(\underline{q}_i)||^2 + \tag{10.2}$$

$$+ \lambda\frac{1}{2L}\sum_{i=1}^{L} ||A\Phi(\underline{q}_i)||^2. \tag{10.3}$$

The first term in (10.2) is the error term induced when we try to reproduce the scores $\underline{\hat{s}}_i$'s for the training vectors $\underline{q}_i$'s. The set of training vectors could be the feature vectors in the image database, or in general, any set of vectors for which we want a specific output. The second term is a penalizing term that favors solutions for which the mapped query $\underline{\hat{q}}_i = A\Phi(\underline{q}_i)$ is of minimum norm.

Due to the high dimensionality of A, the problem expressed in (10.2) will be ill-posed, unless some kind of a constraint or property is imposed on A. From (10.2), we can deduce that rows of A must be lie in the space spanned by the mapped training queries. That is, A can be expressed as

$$A = W\Psi^T(X). \tag{10.4}$$

Now, using (10.4) and the kernel trick, we can express $\underline{\hat{q}}_i = A\Phi(\underline{q}_i)$ as

$$\begin{aligned} \underline{\hat{q}}_i = W\Psi^T(X)\Phi(\underline{q}_i) &= W\mathcal{K}(X, \underline{q}_i) && (10.5) \\ &= W[k(\underline{x}_1, \underline{q}_i) \ \ldots k(\underline{x}_N, \underline{q}_i)]^T, && (10.6) \end{aligned}$$

with an appropriate kernel function $k(\cdot,\cdot)$. Thus, we can rewrite (10.2) as

$$\begin{aligned} \mathcal{T}(W) &= \frac{1}{2L}\sum_{i=1}^{L} ||\underline{\hat{s}}_i - \Psi^T(X)W\mathcal{K}(X, \underline{q}_i)||^2 + && (10.7) \\ &\quad + \lambda\frac{1}{2L}\sum_{i=1}^{L} ||W\mathcal{K}(X, \underline{q}_i)||^2. && (10.8) \end{aligned}$$

Matrix W that minimizes (10.7) can be found by solving the Euler-Lagrange equations that results from this minimization problem [33]. Furthermore, if λ approaches zero, it can be shown that

$$W = [\underline{\hat{w}}_1 \ \underline{\hat{w}}_2 \ldots \underline{\hat{w}}_N]H^{-1}, \tag{10.9}$$

where

$$\underline{\hat{w}}_j = \sum_{i=1}^{L} k(\underline{x}_j, \underline{q}_i)\Psi(X)\left(\Psi^T(X)\Psi(X)\right)^{-1}\underline{\hat{s}}_i, \quad j \in [1, N], \tag{10.10}$$

with $H = [\underline{h}_1\ \underline{h}_2 \dots\ \underline{h}_N]$, and

$$\underline{h}_j = \sum_{i=1}^{L} k(\underline{x}_j, \underline{q}_i)\mathcal{K}(X, \underline{q}_i), \quad j \in [1, N]. \tag{10.11}$$

Note that all the elements of matrix H are represented in terms of the kernel functions values. Now, it is easy to show that the multivariate scoring function $\underline{s}(\underline{q})$ can be written as

$$\underline{s}(\underline{q}) = \Psi^T(X)W\Psi^T(X)\Phi(\underline{q}) = \Psi^T(X)W\mathcal{K}(X, \underline{q}) \tag{10.12}$$

To show that all the operations in (10.12) can be performed in the original low dimensional space instead of the high dimensional feature space, we expand

$$\begin{aligned} \Psi^T(X)W &= \Psi^T(X)[\underline{\widehat{w}}_1\ \underline{\widehat{w}}_2 \dots \underline{\widehat{w}}_N]H^{-1} \\ &= [\underline{p}_1\ \underline{p}_2\ \cdots\ \underline{p}_N]H^{-1} = PH^{-1} = B, \end{aligned} \tag{10.13}$$

where $\underline{p}_j = \sum_{i=1}^{L} k(\underline{x}_j, \underline{q}_i)\hat{\underline{s}}_i$, $j \in [1, N]$. Now, plugging (10.13) into (10.12) yields

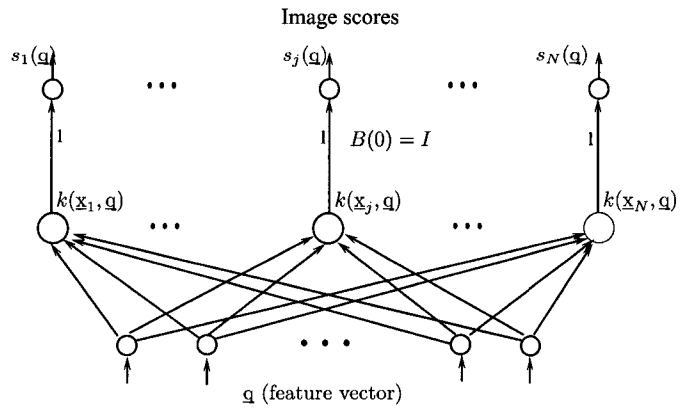
$$\underline{s}(\underline{q}) = B\ \mathcal{K}(X, \underline{q}) \tag{10.14}$$

As can be seen, all the terms in (10.13) and (10.14) can be computed in the original space without using the eigenfunctions themselves. Since the knowledge of matrices P and H is sufficient to compute score vector $\underline{s}(\underline{q})$ and the elements of these matrices are expressed in the kernel domain, instead of matrix A, matrices P and H are stored in the regulator.

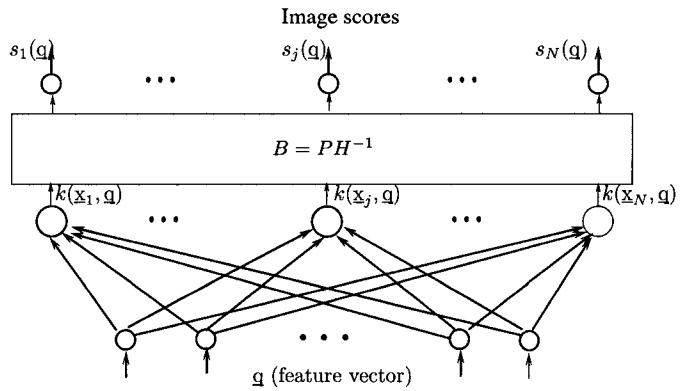
The mapping matrix A of the regulator can be computed by finding W explicitly from (10.9), plugging it into (10.4) and substituting PH^{-1} by B. Thus, we get

$$A = \Psi(X)(\Psi^T(X)\Psi(X))^{-1}B\Psi^T(X) \tag{10.15}$$

Note that the expression for A in (10.15) is in harmony with the assumption that the projection matrix was the best option for the initial start up. The reason being when $B(0) = I$ we arrive at $A = P_X$.



(a) Initial Set up



(b) At the end of model reference learning

Figure 10.1: Practical Network for the Single-regulator Retrieval System: (a) Initial Set up, (b) At the end of model reference learning.

Figure 10.1 shows the practical network for the single-regulator retrieval system. Note that the network is similar to that of the multiple regulator retrieval system presented in Chapter 9 with the difference that learning is carried out not through the structural adaptation. Matrix $B = PH^{-1}$ implements the mapping from the output of the first layer to the output or retrieval layer. Each unit in the first layer has an activation function given by the kernel and a particular stored pattern. The initial value for mapping matrix B is the identity matrix. Does that mean that P and H should also be set to identity matrices as well?

To get initial values for matrices P and H as indicated in (10.13) and (10.11) we

submit each one of the images in the database as queries, i.e. $\underline{q}_i = \underline{x}_i$, $i \in [1, N]$. Since the retrieval system is initially set up using matrix $B(0) = I$, query $\underline{q} = \underline{x}_i$ yields $\underline{s}_i = \mathcal{K}(X, \underline{x}_i)$ and using this value as an artificially 'desired' score vector in $\underline{p}_j = \sum_{i=1}^{L} k(\underline{x}_j, \underline{q}_i)\hat{\underline{s}}_i$ produces the initial value for P as

$$P(0) = [\underline{p}_1\ \underline{p}_2 \dots \underline{p}_j \dots \underline{p}_N]; \quad \underline{p}_j = \sum_{i=1}^{N} k(\underline{x}_j, \underline{x}_i)\mathcal{K}(X, \underline{x}_i);\ j \in [1, N] \tag{10.16}$$

Thus, we can see that $P(0) = G^2$, where G is the Gram matrix defined as $G = [k(\underline{x}_j, \underline{x}_j)]_{i,j=1}^{N}$, assuming that the kernel is symmetric. The initial value for H, $H(0)$, is then easily obtained by solving for $H(0)$ in $I = P(0)H^{-1}(0)$, hence $H(0) = G^2$.

Remark 10.2 Setting up the initial values $H(0) = G^2$ and $P(0) = G^2$, is equivalent to setting up the initial mapping matrix A to the projection matrix $P_X = \Psi(X)(\Psi^T(X)\Psi(X))^{-1}\Psi^T(X)$, which has certain benefits as mentioned in Remark 10.1. From previous results, we know that submitting $\underline{q}_i = \underline{x}_i$, $i \in [1, N]$, as queries yields to $\underline{s}_i = \mathcal{K}(X, \underline{x}_i)$ and $H^{-1}(0) = G^{-2}$. Now, if we plug these values in (10.10), (10.9), and (10.4) in that order then the desired relationship between A and $P_X = \Psi(X)(\Psi^T(X)\Psi(X))^{-1}\Psi^T(X)$ is reveal.

10.4 Relevance Feedback Learning

Recursive equations for $P(L)$ and $H^{-1}(L)$ can be derived to incorporate users relevance information into the image retrieval system. Assume that at time step $L-1$ we already have matrices $P(L-1)$ and $H^{-1}(L-1)$ and at time L a new training pair $(\underline{q}_L, \hat{\underline{s}}_L)$ arrives, then to incorporate the information of this new query we need to compute the new updated matrices $P(L)$ and $H(L)$ as functions of the old matrices $P(L-1)$ and $H(L-1)$, and finally adjust the scoring function. The initial choices of the matrices $P(0)$ and $H(0)$ were described before. After, the initial set up, a series of iterations are performed until the desired system's performance accuracy is achieved. The recursive equation for $P(L)$ can be derived as follows:

$$P(L) = P(L-1) + \hat{\underline{s}}_L \, \mathcal{K}^T(X, \underline{q}_L). \tag{10.17}$$

Similarly, a recursive equation for $H(L)$ can be found using (10.11) as follows:

$$H(L) = H(L-1) + \mathcal{K}(X, \underline{q}_L)\mathcal{K}^T(X, \underline{q}_L) \tag{10.18}$$

Since we need $H^{-1}(L)$ to compute matrix B a recursive equation for finding $H^{-1}(L)$ is also needed. To this aim, the matrix inversion lemma [49] is applied to (10.18) to obtain a recursive equation

$$\begin{aligned} H^{-1}(L) &= \left(H(L-1) + \mathcal{K}(X, \underline{q}_L)\mathcal{K}^T(X, \underline{q}_L)\right)^{-1} \\ &= H^{-1}(L-1) - \frac{H^{-1}(L-1)\mathcal{K}(X, \underline{q}_L)\mathcal{K}^T(X, \underline{q}_L)H^{-1}(L-1)}{1 + \mathcal{K}^T(X, \underline{q}_L)H^{-1}(L-1)\mathcal{K}(X, \underline{q}_L)} \end{aligned} \tag{10.19}$$

which requires only matrix multiplications and additions.

To emphasize the importance of current training sample over the previous ones, it is convenient to include a forgetting factor $\lambda < 1$ into the recursive equations for $P(L)$ and $H^{-1}(L)$ as follows:

$$\begin{aligned} P(L) &= \lambda P(L-1) + \hat{\underline{s}}_L \, \mathcal{K}^T(X, \underline{q}_L) \\ H^{-1}(L) &= \frac{1}{\lambda}\left[H^{-1}(L-1) - \frac{H^{-1}(L-1)\mathcal{K}(X, \underline{q}_L)\mathcal{K}^T(X, \underline{q}_L)H^{-1}(L-1)}{\lambda + \mathcal{K}^T(X, \underline{q}_L)H^{-1}(L-1)\mathcal{K}(X, \underline{q}_L)}\right] \end{aligned} \tag{10.20}$$

Looking at (10.20), it is evident that some elements of matrices $H^{-1}(L)$ might be very large as learning progresses. To circumvent this problem we can impose bounds on the norm or the trace of this matrix. The reader is referred to Chapter 11 in [120] for the details of some procedures that address this problem.

10.5 Experimental Results

An experiment on relevance feedback learning using the recursive equations for $P(L)$ and $H^{-1}(L)$ was conducted on the same image database used in the multiple regulator

system (see Chapter 9 for the details on the training and testing sets). The initial configuration or initial set up of the single regulator is shown in Figure 10.1(a). For this initial set up, matrices $P(0)$ and $H^{-1}(0)$ were set to G^2 and G^{-2}, respectively (see Remark 10.2). Now, relevance feedback learning is apply to adjust the internal parameters of the regulator by using the training set.

Basically, relevance feedback learning is carried out in three steps: (i) randomly choose an unmarked query image from the training set and submit it to the single regulator retrieval system; (ii) set the 'desired' scores to 0.9 for all the relevant images whose scores are less than 0.9 while keeping actual scores for the rest of the images; (iii) update the matrices P and H^{-1} using (10.17) and (10.19), and mark the query image; and (iv) repeat the previous steps if there is an unmarked query image left in the training set, otherwise stop.

To briefly analyze the performance and generalization properties of the single regulator image retrieval system, the recall plots for the training and testing samples are shown in Figure 10.2 there are two important observations. First: the linear trend of the recall plot for the training set is clearly evident due to the nature of this mode of learning. Second: the behavior of the recall plot for the testing set is somewhat strange, because as can be seen, at some points the recall measure is better for the testing set than that for the training set. Nonetheless, at later stages of learning the recall measure starts dropping as expected until it reaches a final value of 0.95 at the end of relevance feedback learning. Comparison with the results in Experiment 2 indicates the superiority of the multiple regulator image retrieval system. Moreover, relevance feedback learning for this experiment lasted more than 6 hours on a Pentium IV computer running at 2.4 GHz.

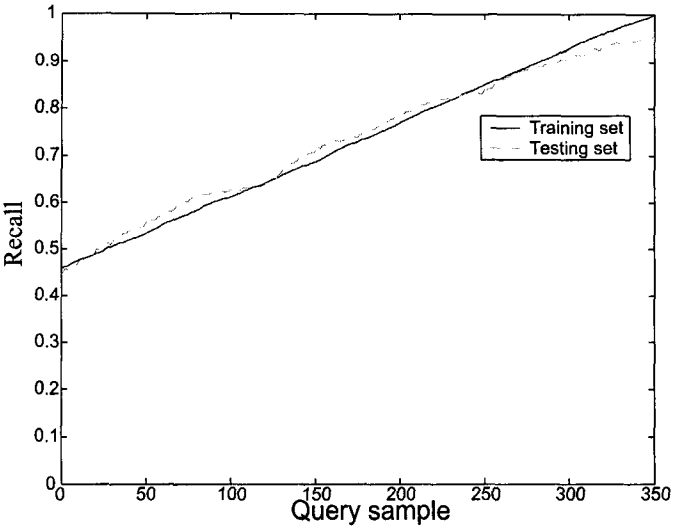


Figure 10.2: Recall plot for training and testing sets using the single-regulator image retrieval system.

10.6 Conclusions

This chapter presents the proposed single regulator image retrieval system. The retrieval system contains four major components. A regulator transforms the submitted query image into a single 'optimal' query image. This operation is carried out in a high dimensional space induced by a kernel producing function. That is, the optimal query does not have some perceivable significate. Further research could be focused on finding the pre-image of this 'optimal' query in order to explore some of its possible properties in a low-dimensional space. A plant stores some information of the image database in a matrix, where each column is a transformed feature vector of an image in the database into a high dimensional space. A model reference provides a series of input-output relationships using an augmented database that contains the original image database together with class label information. An adjustable mechanism is utilized to provide the learning rules for the model reference and the relevance feedback learning modes. The learning rules for the two modes are derived from a constrained optimization problem using mapped image and query features vectors in

a high dimensional space. An experiment was conducted for the relevance feedback learning to demonstrate the generalization capability of the single regulator image retrieval system for learning from expert users. Although the recall plots show good generalization capability of this system, they are clearly inferior to those reported for Experiment 2 on relevance feedback for the multiple regulator described in Chapter 9.

CHAPTER 11

CONCLUSIONS AND FUTURE WORK

11.1 Conclusions and Observations

The development of adaptable text and image retrieval systems that can incorporate the solutions proposed by expert users via relevance feedback is the main goal of the present work. The proposed systems use the well-known model-reference adaptive control framework with some major differences and characteristics.

The proposed adaptable text retrieval system is capable of incorporating knowledge from both an external retrieval system (model-reference) and from multiple expert users. The first characteristic allows the system to mimic the input/output behavior of an external retrieval system (when exists) while the second characteristic provides the means to incorporate expert users solutions, hence improving the responses of the external system. Two learning mechanisms, one that captures the input/output behavior of the model-reference and the other that captures the users' knowledge, are employed. In each case, two different versions of the algorithms are proposed. In the first version, a regression framework is used to incorporate feedback based upon the scores of some relevant and non-relevant documents. The second algorithm uses a classification framework to incorporate feedback as in a two-class problem. Independent of the choice of the framework (regression or classification), the aim of the two learning mechanisms is to map the original query into an 'optimal' query that would provide the expected solutions. This transformation somehow

resembles the addition and/or deletion of terms that a user needs to perform on the original query in order to get the desired list of documents. However, in our scheme this is done without involving the user directly with the query modification process. Clearly, the regression framework is preferred if the scores are available. It was found that the optimal query lies in the space spanned by the documents of the database, regardless of the chosen framework. This could be seen as a generalization of the popular Rocchio's formula [11] but with the difference that here the weights for the documents are found by solving an optimization problem.

To set up (or initialize) our proposed text retrieval system a learning mode referred to as initial model-reference learning is used to reproduce the behavior of an external model text retrieval system. The initial model-reference learning uses a limited set of single-term queries to set up the internal parameters of the regulator. To make the trained system robust to environmental (database) changes resulting from document addition or deletion operations a learning mode referred to as model-reference following learning mode is created. This learning mode makes our text retrieval system adapts itself by changing its structure to accommodate the new information or erase the old one. Finally, to incorporate users' expertise via relevance feedback without erasing the previously captured learning from other users a learning mode referred to as relevance feedback learning is developed.

The proposed model reference text retrieval system (MRTRS) contains: a static retrieval system or a 'plant' that stores the information of the documents in the database; a mapping mechanism or a 'regulator' that transforms the original query into an optimal one; an adjustment mechanism that provides the adequate adjustments to the parameters in the regulator; an external text retrieval system (TRS) or reference model that provides a limited set of training samples used for the initial model-reference and model following learning modes; and a user relevance feedback repository that provides a limited set of training samples to fine-tune the response

of our text retrieval system for future solutions. In our proposed system, relevance feedback captures the information from the expert users via an adaptation mechanism that adjusts the regulator parameters.

The search and retrieval subsystem of MRTRS can be represented by a structurally adaptable three-layer connectionist network with two static and one dynamic layers. There are two benefits immediately exploited for this connectionist representation; (i) the operations involved are basic sums and multiplications which are easier to carry out than the non-linear activation functions of general neural network representations, (ii) the sparseness of the weight connections in the static layers representing the documents is fully exploited by keeping, in the computer memory, only those connections that are not zero, allowing the search and retrieval processes to be performed almost in real-time. In the connectionist network, the first and third layer are associated to the documents and the input layer and the second layer are associated to the terms (concepts). If a new document is introduced along with its new and old concepts (terms) the structure of our connectionist system is adjusted modifying all layers to incorporate this new information. One neuron is created in both the first and the output layers and several neurons are created in the input and third layers to account for the new terms. If a document is deleted the inverse process of document addition takes place. The incorporation of users' expertise is accomplished via a relevance feedback mechanism where the users are able to modify the response of the system if the listed solutions are not adequate. Two main benefits of the relevance feedback mechanism worth to highlight are:(i) the incorporation of new information is carried out without scarifying the stability of the retrieval system for previous solutions for the same or other users, and (ii) relevance feedback learning affects only a few adjustable parameters of the connectionist system, mainly those associated with the terms in the submitted query, making it suitable for online implementation. These adjustable parameters correspond to the connections of the input

to the first layer of our retrieval system, all other connections in the other layers are unchanged. This feature allows a fast implementation of our proposed relevance feedback learning mechanism because the adjustable connections represent only a fraction of the total number of connections in the network.

The experiments conducted on the HP-text database revealed the proposed MRTRS can indeed capture the relevance information from the users using either a regression or a classification learning frameworks mentioned before. This is attested by the large values of the non-parametric Kendal's τ_b correlation measure generated during the last stages of learning. The measure almost reached one (perfect match) for some queries in the training and testing sets, especially for the single and two-term queries. The recall measure also indicated that the response of our MRTRS system approaches that of the users after only a few rounds of relevance feedback learning. The recall plots also attest to the fact that the proposed MRTRS can incorporate users' expertise into the retrieval process by using either the regression or the classification modes. In the regression mode, the recall plots increases from 0.673, 0.720, and 0.657 to 0.961, 0.980, and 0.869 for the 1-term, 2-term, and 3^{+}-term queries, respectively, while in the classification mode, it increases from 0.551, 0.453, and 0.385 to 0.808, 0.797, and 0.698 for the same queries. The recall plots indicate the effectiveness of our system in incorporating relevance feedback learning without jeopardizing the performance of the system to the previously learnt information.

The optimal query mapping mechanism proposed in this work captures the documents' content from both TRS system and expert users via relevance feedback. The weights of the optimal query domain can be used to cluster the queries with similar concepts. This is clearly beneficial for providing clues to the expert users. A different clustering method was tried, namely the two dimensional self-organizing feature map (2D-SOM). It was shown that the proposed agglomerative algorithm provides clusters with much better cohesion (measure used for closeness of queries in a cluster) that

those generated by the expert users or the 2D-SOM.

The proposed MRTRS was then extended to image databases. Two different image retrieval systems (IRS) were proposed. One referred to as the multiple regulator IRS and the other referred to as the single regulator IRS. The multiple regulator retrieval system contains N regulators, each associated to a particular image in the database; while the single regulator system contains only one regulator associated to all images in the database. Though it may seem complicated, this apparent drawback of the multiple regulator IRS leads to a more efficient retrieval system that only needs to update a small number of parameters to incorporate user's relevance feedback. Although the structure of the single regulator IRS is simpler than that of the multiple regulator IRS, the computational complexity of the single regulator is by far greater. This leads to greater time to run the same rounds of feedback in the single regulator system compared to the multiple regulator case.

The two systems contains four components namely: a plant that store the information of the images in the database; a regulator (single-regulator system) or several regulators in charge of transforming the submitted query image into the 'optimal' one(s) that will generate the required solutions; an adjustment mechanism that provides the necessary adjustments to the regulator(s); and a model-reference model that gives a series of pairs as query image-score vector relationships to train the image retrieval system. The plant and the regulator are in a high dimensional space where basic linear matrix operations are computed. However, for creating a practical retrieval system some of the formulations are taken to the original low dimensional space. For incorporating users' expertise, a rigorous relevance feedback learning was developed; while for assimilating the information of an external image retrieval system (when it is available) a specific model-reference learning was introduced. It was found that relevance feedback learning was not appropriate for the single regulator system (in its present form) due to the long time it takes to process a query image,

which precludes any real-time relevance feedback learning.

Either the multiple regulator or the single regulator retrieval systems can be represented by two simple layers of neurons. In the multiple regulator case, the first layer is conceptually divided in a set of pools, each assigned to an image in the database. Also, each pool contains one or more neurons that are connected to the output or scoring layer. Each neuron represents a kernel function centered in a particular query image. A weighed summation of the kernel functions in a particular pool provides a similarity measure between the associated image to the pool and the submitted query image. In the single regulator case there is no conceptual division of the first layer because the learning operations are not localized.

For the multiple regulator image retrieval system (MRIRS) it was found that the relevance feedback mechanism causes the expansion of the first layer, which maintains a set of pools associated with an image in the database. The expansion is stopped when the pool responds adequately to the users requirements. Every time that a pool does not respond adequately to a submitted query and it receives relevance feedback the pool expands itself by including another kernel function centered in the submitted query image, assimilating in this form the users' expertise into the retrieval process. As can be seen, the pool expansion process is highly localized in the multiple regulator case. For the single regulator image retrieval system (SRIRS), it was found that the model-reference and relevance feedback learning modes affected all the parameters of the regulator. This undesirable characteristic made us to abandon the idea of a practical implementation of the single regulator retrieval system, even for a small image repository like the underwater image database used in the present research. Future research should investigate how the single regulator system can easily be parameterized for different learning phases.

11.2 Future work

There are many avenues in which the proposed text retrieval systems can further be researched. The following is just a brief list of a few that we believe are possible extensions.

- Creation of taxonomies of knowledge: Presently, the amount of unstructured information is presenting a challenge in management and utilization of relevant knowledge to the consumers of such information. While the lack of information is no longer an issue, the actual identification of relevancy and categorization of knowledge within a specific domain remains to be a crucial issue in research and practical application. It is desirable to create soft categories of knowledge that can be updated dynamically and adopt to new users queries and interest in time without the expensive and exhaustive need to read and/or tag the documents explicitly. This will additionally help the less familiar users of the knowledge domain to navigate through taxonomies of knowledge that was created based on prior queries and knowledge of the more experts users.

- Other optimization methods: Modify the formulations and learning methods by using other optimization methods such as in the Lasso [121]. This might lead to more meaningful optimal queries that could better capture users' requirements by maintaining a minimum number of terms in the optimal query. At present these optimization methods work under the concept of linear programming which does not lead to an explicit expression similar to the one we found in our formulation. Nevertheless, this methodology could be studied in more detail to ponder its benefits.

- Kernel-Based TRS: Extend the TRS algorithms to the kernel domain where high-order correlations between the terms in the query could be better exploited.

This is a real challenge given the large number of terms and documents even for special-purpose databases.

The development of image retrieval systems that respond adequately to relevance feedback from expert users is a real challenge. In contrast to the text retrieval area where the concepts (terms) are well-defined and easy to extract, the concepts in an image retrieval system are not easy to extract. We believe some issues for the image retrieval area that deserve further studies are:

- Efficient Multiple Regulator IRS: Devise new structures and algorithms for the multiple-regulator image retrieval system to minimize the number of regulators especially for large image databases. This idea could be linked to the that of the discovery of a taxonomy discussed for the text retrieval system. If the pools were associated to a particular category of knowledge rather than to a particular image we could reduce the number of regulators and if the categories were structurally grouped hierarchically then even a further reduction could be achieved.

- Combine Text and Image Retrieval Systems: Developing under a unified approach, information retrieval systems for Multimedia databases where text and images sources are blended together without explicit text annotations is a promising area of study. Most of the search process in these databases could be performed via a text query, contradicting in some way the old saying that says "an image worths a thousand words" or alternatively via image querying for the most part. The problem of a query formulation for these databases involves adequate combining of the text part and the image parts of the query. We believe the development of a retrieval system that is capable of combining or simultaneously treating text and images would be of great interest for a variety of multimedia and educational applications.

- IRS with semantics: Include the semantic information into the feature vectors and query images. At present, image retrieval systems that include the semantics of an image into the search process are at their infancy. Semantic relationships such as nearby, inside, and outside together with Gestalt factors [122] such as similarity and symmetry could help to conceptually form an object from its disjoint parts. Several psychological studies on visual perception have focused on Gestalt factors to understand visual recognition on humans. Since our relevance feedback learning is based upon users expertise, it could be of interest to reformulate our relevance feedback algorithms to include these factors when training the flexible image retrieval system.

APPENDIX A

TIKHONOV FUNCTIONAL

The analytic approach to regularization problems relies in minimizing the *Tikhonov functional* $\mathcal{T}$ [115]

$$\mathcal{T} = \frac{1}{2}\sum_{i=1}^{N}[y_i - f(\underline{X}_i)]^2 + \lambda\psi(f) \tag{A.1}$$

to find an optimal function f close enough to data points in the training set and smooth enough to reduce the second term in (A.1). Here, the functional $\psi(f)$ or stabilizer defines the smoothness of the function f. A usual stabilizer is of the form $\psi(f) = ||Df||^2$, where the reproductive kernel function Df is in the Hilbert space.

Before we define each of the parameters and terms involved in this functional let us state the regularization problem as follows: Given a set of training examples in the form $\{\underline{X}_i, y_i\}$, where $X_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$, we want a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that reduces the error between for the training points as indicated by the first term in (A.1) and, at the same time, yields the highest smoothness (values close to zero indicates high smoothness) as indicated by the second term in (A.1). Here, λ is a positive real number. The parameter λ provides a trade-off between pure interpolation ($\lambda = 0$) and pure smoothness ($\lambda \rightarrow \infty$). Note, that if λ were set to zero in (A.1), then the problem is an ill-posed one because there are many functions that make the first term in (A.1) zero. However, if we solve the regularization problem for $\lambda > 0$ and then make $\lambda \rightarrow 0$, then the solution can be unique (ill-posed regularized) or not (ill-posed non-regularized), depending on the dimension of the input space and on the number of data samples. We are going to see this problem in the example in this section.

Different stabilizer functionals yield a variety of function approximations families.

For instance the stabilizer

$$\psi(f) = \int_{\mathbb{R}^n} e^{a\|\underline{w}\|^2} |\tilde{f}(\underline{w})| d\underline{w},$$

where a is a parameter and $\tilde{f}(\underline{w})$ is the Fourier transform of $f(\underline{x})$ satisfying Dirichlet conditions, produces Gaussian radial basis function approximations [123]. Other stabilizers that yield a spectrum of radial basis functions are also presented in [123].

Example: Let us consider the *Tikhonov Functional*

$$\mathcal{T}(f) = \frac{1}{2}\sum_{i=1}^{N}[y_i - f(x_i)]^2 + \frac{1}{2}\lambda \int_{x_1}^{x_N} \left|\frac{d^2 f}{d\zeta^2}\right|^2 d\zeta \tag{A.2}$$

for the univariate function f and suppose that f^* is the function that minimizes the functional. Then, $\mathcal{T}(f^*) \leq \mathcal{T}(f)$ holds for any $f : \mathbb{R} \rightarrow \mathbb{R}$. This is also true for $f(x) = f^*(x) + \alpha\eta(x)$, where α is a scalar. Let us assume that $\eta(x)$ and $\eta'(x)$ vanish at extreme points x_1 and x_N, i.e. $\eta(x_1) = \eta(x_N) = 0$ and $\eta'(x_1) = \eta'(x_N) = 0$. Thus, we can write

$$\mathcal{T}(f^* + \alpha\eta(x)) = \frac{1}{2}\sum_{i=1}^{N}[y_i - (f^*(x_i) + \alpha\eta(x_i))]^2 + \frac{1}{2}\lambda \int_{x_1}^{x_N} \left|\frac{d^2 f^*}{dx^2} + \alpha\frac{d^2\eta}{dx^2}\right|^2 dx \tag{A.3}$$

Now, given that $\mathcal{T}(f^*) \leq \mathcal{T}(f^* + \alpha\eta(x))$ holds $\forall\alpha \in \mathbb{R}$ and $lim_{\alpha\rightarrow 0}\mathcal{T}(f^* + \alpha\eta(x)) = \mathcal{T}(f^*)$, it follows

$$\left.\frac{\partial\mathcal{T}(f^* + \alpha\eta(x))}{\partial\alpha}\right|_{\alpha\rightarrow 0} = 0 \tag{A.4}$$

Moreover, it is easy to show that if we apply (A.4) to $\mathcal{T}$ we arrive at

$$\sum_{i=1}^{N}[y_i - f^*(x_i)]\eta(x_i) = \lambda \int_{x_1}^{x_N} \frac{d^2 f^*}{dx^2}\frac{d^2\eta}{dx^2} dx \tag{A.5}$$

The left part in (A.5) is converted to the integral form

$$\int_{x_1}^{x_N} \sum_{i=1}^{N}[y_i - f^*(x)]\eta(x)\delta(x - x_i)dx \tag{A.6}$$

by using the delta function $\delta(x)$ which satisfies the properties

$$\begin{aligned} &\text{a)} \quad \delta(x) = 0, \quad \text{when } x \neq 0; \\ &\text{b)} \quad \int_{-\infty}^{\infty} \delta(x)dx. = 1, \end{aligned} \tag{A.7}$$

while, by using the identity $\int udv \equiv uv - \int vdu$ and the vanishing properties of $\eta(x)$ on points x_1 and x_2, the right part in (A.5) is converted to

$$\lambda \int_{x_1}^{x_N} \frac{d^4 f^*}{dx^4} \eta(x)dx \tag{A.8}$$

Thus, (A.5) can be rewritten as,

$$\int_{x_1}^{x_N} \sum_{i=1}^{N} [y_i - f^*(x)]\eta(x)\delta(x - x_i)dx = \lambda \int_{x_1}^{x_N} \frac{d^4 f^*}{dx^4} \eta(x)dx \tag{A.9}$$

Now, because (A.9) must hold for any function $\eta(x)$ the Euler-Lagrange differential equation

$$\sum_{i=1}^{N} [y_i - f^*(x)]\delta(x - x_i) = \lambda \frac{d^4 f^*}{dx^4} \tag{A.10}$$

also holds.

It is clear from (A.10) that f^* must be a polynomial of degree three or less and that it must be defined by intervals (otherwise, it would not comply with the left part in (A.10)). This polynomial is a cubic spline. Another variational approach which arrives at a cubic spline can be found in [124].

Previous example shows us that the smoothest function for scattered data interpolation other than a hyperplane is a cubic spline. Hence, if we want an univariate function that passes through all points, this function should be a cubic spline.

REFERENCES

[1] K. K. L., G. L., and L. D. D., "Trec-3 ad-hoc, routing retrieval and thresholding experiments using pircs," *Proceedings of TREC-3*, pp. 247–255, 1995.

[2] M. R. Azimi-Sadjadi, J. Salazar, S. Srinivasan, and S. Sheedvash, "An adaptable connectionist text retrieval system with relevance feedback." Submitted to the IEEE Transactions on Neural Networks, July 2004.

[3] M. Azimi-Sadjadi and J. Salazar, "Relevance feedback using neural networks," tech. rep., HP Corporation, Nov 2002. Final Report to HP Corporation.

[4] M. R. Azimi-Sadjadi, J. Salazar, S. Srinivasan, and S. Sheedvash, "An adaptable connectionist text retrieval system with relevance feedback." Submitted, IEEE Transactions on Neural Networks.

[5] G. Thoma and D. Le, "Medical database input using integrated ocr and document analysis and labeling technology," *Proceedings 1997 Symposium on Document Image Understanding Technology*, pp. 180–181, 1997.

[6] X. Qian *et al.*, "Shape indexing in medical image databases using pre-shape embedding," *Proc. Distributed Databases and processing in Medical Image Computing, MICCAI*, pp. 36–44, Sept 2004.

[7] T. Onoda, H. Murata, and S. Yamada, "Relevance feedback with active learning for document retrieval," *Proceedings of the International Joint Conference on Neural Networks*, vol. 3, pp. 1757 – 1762, 2003.

[8] J. Rocchio, *Relevance Feedback in Information Retrieval in The Smart System - Experiments in Automatic document processing.* Englewood Cliffs, NJ: Prentice Hall Inc., 1971.

[9] L. Wu, C. Faloutsos, K. P. Sycara, and T. R. Payne, "Falcon: Feedback adaptive loop for content-based retrieval," in *Proceedings of the 26th International Conference on Very Large Data Bases*, pp. 297–306, Morgan Kaufmann Publishers Inc., 2000.

[10] M. Ortega, Y. Rui, K. Chakrabarti, A. Warshavsky, S. Mehrotra, and T. Huang, "Supporting ranked boolean similarity queries in mars," *IEEE Trans. on Knowledge and Data Engr.*, vol. 10, pp. 905–925, December 1999.

[11] R. Baeza, *Modern Information Retrieval.* Addison Wesley, 1999.

[12] S. Feinglos, *MEDLINE: A Basic Guide to Searching.* Chicago:Medical Library Association, 1985.

[13] American College of Physicians, *Annals of Internal Medicine:1993-1998.* Majors Publishing, 1 ed., 1998.

[14] R. Cleary *et al.*, "Comparative hospital databases: value for management and quality," *Qual Health Care PMID: 10136257 [PubMed - indexed for MEDLINE]*, vol. 3(1), pp. 3–10, 1994.

[15] N. Allee, K. Alpi, K. Cogdill, C. Selden, and M. Youngkin, "Public health information and data: a training manual [internet]," *National Library of Medicine (US)*, 2004.

[16] Karen Sparck Jones and Peter Willett, *Readings in Information Retrieval.* Morgan Kaufmann Multimedia Information And Systems Series, 1997.

[17] D. Harman, *Relevance feedback and other query modification techniques*, pp. 241–263. Prentice-Hall, Inc., 1992.

[18] Gerard Salton, *Automatic Information Organization and Retrieval.* McGraw-Hill, 1968.

[19] G. Salton, C. Buckley, and E. A. Fox, "Automatic query formulations in information retrieval," tech. rep., Cornell University, Ithaca, NY, USA, 1982.

[20] G. Salton, *The Smart retrieval system: experiments in automatic document processing.* Englewood Cliffs, 1971.

[21] C. Claudio and R. Giovanni, "Order-theoretical ranking," *Journal of the American Society for Information Science*, vol. 51, no. 7, pp. 587–601, 2000.

[22] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, 2002.

[23] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[24] V. Vapnik, *The Nature of Statistical Learning Theory.* Springer, 1995.

[25] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Computational Learning Theory*, pp. 287–294, 1992.

[26] G. Guo-Dong, A. Jain, M. Wei-Ying, and Z. Hong-Jiang, "Learning similarity measure for natural image retrieval with relevance feedback," *IEEE Transactions on Neural Networks*, vol. 13, pp. 811–820, Jul 2002.

[27] S. Wong and Y. Yao, "Query formulation in linear retrieval models," *Journal of the American Society for information Science*, vol. 41, pp. 334–341, July 1990.

[28] Y. Ouyang and W. Jermann, "Neural network based retrieval issue on prototype database systems," *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, pp. 1493 – 1497, 1991.

[29] K. L. Kwok, "A network approach to probabilistic information retrieval," *ACM Transactions on Information Retrieval*, vol. 13, pp. 324–353, 1995.

[30] F. Crestani, "Comparing neural and probabilistic relevance feedback in an interactive information retrieval system," *IEEE International Conference on Neural Networks*, vol. 5, pp. 3426–3430, 1994.

[31] R. C. Muniyandi., "Neural network: an exploration in document retrieval system," *TENCON, Proceedings*, vol. 1, pp. 156 – 160, Sept 2000.

[32] M. Boughamen, A. Caron, and R. Layaida, "A neural network model for documentary self-organezing and querying," *Proceedings ICCI'93 Fift International Conference on Computing and Information*, pp. 512–518, May 1993.

[33] S. Haykin, Neural Networks: A Comprehensive Foundation. Prentice-Hall, 2 ed., 1999.

[34] R. Bouchachia, A; Mittermeir, "A neural cascade architecture for document retrieval," *Neural Networks, Proceedings of the International Joint Conference*, vol. 3, pp. 1915 – 1920, July 2003.

[35] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, 2002.

[36] N. Fuhr, "Optimum polynomial retrieval functions based on the probability ranking principle," *ACM Transactions on Information Systems*, vol. 7, no. 3, pp. 183–204, 1989.

[37] M. G. Kendall, *Rank Correlation Methods.* Hafner Pusblishing Company, 1962.

[38] M. Kherfi and D. Ziou, "Relevance feedback for cbir: a new approach based on probabilistic feature weighting with positive and negative examples," *IEEE Transactions on Image Processing*, vol. 15, pp. 1017–1030, April 2006.

[39] I. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yianilos, "Bayesian relevance feedback for image retrieval," *Proc. Inter. Conf. Pattern Recogn.*, vol. 3, pp. 362–369, 1996.

[40] B. Li and S. Yuan, "A novel relevance feedback method in content-based image retrieval," *International Conference on Information Technology: Coding and Computing (ITCC'04)*, vol. 2, 2004.

[41] Y. Li, X. Wan, and C. Kuo, *Image Databases: Search and Retrieval of Digital Imagery*, ch. 10: Introduction to content-based image retrieval - Overview of key techniques, pp. 261–284. Wiley Inter-Science, 2002. V. Castelli and L.D. Bergman (Eds) - ISBN: 0-471-32116-8.

[42] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1349–1380, December 2000.

[43] C. Meilhac, M. Mitschke, and C. Nastar, "Relevance feedback in surfimage," in *Proceedings of Fourth IEEE Workshop on Applications of Computer Vision*, pp. 266–267, Oct 1998.

[44] G. Aggarwal, S. Ghosal, and P. Dubey, "Efficient query modification for image retrieval," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 255–262, 2000.

[45] P. Muneesawang and L. Guan, "A neural network approach for learning image similarity in adaptive cbir," *IEEE Fourth Workshop on Multimedia Signal Processing*, pp. 257–262, 2001.

[46] K. Porkaew, M. Ortega, and S. Mehrota, "Query reformulation for content based multimedia retrieval in mars," *IEEE International Conference on Multimedia Computing and Systems*, vol. 2, pp. 747–751, June 1999.

[47] O. Chapelle, P. Haffner, and V. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Transactions on Neural Networks*, 1999.

[48] N. Vasconcelos and A. Lippman, "A bayesian framework for content-based indexing and retrieval," in *Data Compression Conference*, p. 580, 1998.

[49] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification.* John Wiley and Sons,INC, 2 ed., 2001.

[50] Y. Chen, X. S. Zhou, and T. Huang, "One-class SVM for learning in image retrieval," in *Proceedings. 2001 International Conference on Image Processing*, vol. 1, (Thessaloniki , Greece), pp. 34–37, 2001.

[51] E. Bauer, "An empirical comparison of voting classification algorithms:bagging, boosting and variants," *Machine Learning*, vol. 36, pp. 105–139, August 1999.

[52] S. Zhong, H. Zhang, S. Li, and S. Ma, "Relevance feedback in content-based image retrieval: Bayesian framework, features subspaces, and progressive learning," *IEEE Transactions on Image Processing*, vol. 12, pp. 924–937, August 2003.

[53] Y. Wu and A. Zhang, "A feature re-weighting approach for relevance feedback in image retrieval," *Proc. International Conference on Image Processing*, vol. 2, pp. 581–584, Sept. 2002.

[54] S. Tong and E. Chang, "Support vector machine active learning for image retrieval," in ACM Inter. Conf. on Multimedia, pp. 107–118, October 2001.

[55] P. Muneesawang and L. Guan, "An interative approach for CBIR using a network of radial basis functions," *IEEE Transactions on Multimedia*, vol. 6, pp. 703–716, October 2004.

[56] R. Manmatha, T. Rath, and F. Feng, "Modeling score distributions for combining the outputs of search engines," in *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 267–275, ACM Press, 2001.

[57] Karl Jophan Åström and Björn Wittenmark, *Adaptive Control.* Adisson-Wesley, 1989.

[58] S. Sastry and M. Bodson, *Adaptive Control: Stability, Convergence, and Robustness.* Prentice-Hall, 1994.

[59] M. D. Buhmann, *Radial Basis Functions.* Cambridge University Press, 2003.

[60] L. L. Scharf, *Statistical Signal Processsing: Detection, Estimation, and time series analysis.* Addison-Wesley, 1991.

[61] W. Jermann and Y. Ouyang, "Neural network based retrieval issue on prototype database systems," in *Decision Aided for Complex Systems*, vol. 3, pp. 1493–1497, IEEE International Conference on Systems Man and Cybernetics, Oct 1991.

[62] H. Kordylewsky, D. Graupe, and K. Liu, "A novel large memory neural network as an aid in medical diagnosis applications," *IEEE Transactions on Information Tecnology in Biomedicine*, vol. 5, pp. 202–209, Sep 2001.

[63] G. H. Golub and C. Van Loan, Matrix Computations. Johns Hopkins Press, 3rd edition ed., 1996.

[64] J. C. French, A. L. Powell, C. L. Viles, T. Emmitt, and K. J. Prey, "Evaluating database selection techniques: A testbed and experiment," in *Research and Development in Information Retrieval*, pp. 121–129, 1998.

[65] E. Brown, J. Callan, and W. Croft, "Fast incremental indexing for full-text information retrieval," in *Proceedings of the 20th International Conference on Very Large Databases (VLDB)*, (Santiago, Chille), pp. 192 – 202, September 1994.

[66] A. Tomasic, H. García-Molina, and K. Shoens, "Incremental updates of inverted lists for text document retrieval," in *In Proceedings of 1994 ACM International Conference on management of Data (SIGMOD94)*, pp. 289–300, 1994.

[67] N. Shivakumar and H. García-Molina, "Wave-indices: indexing evolving databases," in *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 381–392, ACM Press, 1997.

[68] S. Lawrence, K. Bollacker, and C. L. Giles, "Indexing and retrieval of scientific literature," in *CIKM '99: Proceedings of the eighth international conference on Information and knowledge management*, (New York, NY, USA), pp. 139–146, ACM Press, 1999.

[69] F. Scholer, H. E. Williams, J. Yiannis, and J. Zobel, "Compression of inverted indexes for fast query evaluation," in *SIGIR '02: Proceedings of the 25th annual*

international ACM SIGIR conference on Research and development in information retrieval, (New York, NY, USA), pp. 222–229, ACM Press, 2002.

[70] Brian D.O Anderson and John B. Moore, *Optimal Filtering.* Dover Publications, 2005.

[71] B. J. Jansen, J. Bateman, A. Spink, and T. Saracevic, "Real life information retrieval: A study of user queries on the web," *ACM SIGIR Forum*, vol. 32, no. 1, pp. 5–17, 1998.

[72] N. C. M. Ross, "End user searching on the internet: an analysis of term pair topics submitted to the excite search engine," *J. Am. Soc. Inf. Sci.*, vol. 51, no. 10, pp. 949–958, 2000.

[73] A. Spink, D. Wolfram, M. B. J. Jansen, and T. Saracevic, "Searching the web: the public and their queries," *J. Am. Soc. Inf. Sci. Technol.*, vol. 52, no. 3, pp. 226–234, 2001.

[74] S. Srinivasan and M. R. Azimi-Sadjadi, "An iterative relevance feedback learning algorithm for image retrieval system," in *To appear in Proceedings of IEEE Intl. Joint Conf. on Neural Networks*, August, 2005.

[75] John G. Proakis and Charles M. Rader and Fuyun Ling, *Algorithms for Statistical Signal Processing.* Prentice Hall, 2001.

[76] Simon Haykin and Bernard Widrow, *Least-Mean-Square Adaptive Filters.* Wiley-Interscience, 2003.

[77] Goodwin Graham C. and Kwai Sang Sin, *Adaptive Filtering Prediction and Control.* Prentice-Hall, 1984.

[78] Herbert Robbins and Sutton Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.

[79] J. Xu and W. B. Croft, "Query expansion using local and global document analysis," in *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, NY, USA), pp. 4–11, ACM Press, 1996.

[80] E. M. Voorhees, "Query expansion using lexical-semantic relations," in *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, NY, USA), pp. 61–69, Springer-Verlag New York, Inc., 1994.

[81] E. N. Efthimiadis and P. V. Biron, "UCLA-okapi at TREC-2: Query expansion experiments," in *Text REtrieval Conference*, pp. 278–290, 1993.

[82] N. E. Efthimis, "Interactive query expansion: A user-based evaluation in a relevance feedback environment," *Journal of the American Society for Information Science*, vol. 51, no. 11, pp. 989–1003, 2000.

[83] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 269–274, ACM Press, 2001.

[84] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Mach. Learn.*, vol. 42, no. 1/2, pp. 143–175, 2001.

[85] Y. Zhao and G. Karypis, "Soft clustering criterion functions for partitional document clustering: a summary of results," in *CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management*, (New York, NY, USA), pp. 246–247, ACM Press, 2004.

[86] K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram, "A hierarchical monothetic document clustering algorithm for summarization and

browsing search results," in *WWW '04: Proceedings of the 13th international conference on World Wide Web*, (New York, NY, USA), pp. 658–665, ACM Press, 2004.

[87] Y. Zhao and G. Karypis, "Empirical and theoretical comparisons of selected criterion functions for document clustering," *Mach. Learn.*, vol. 55, no. 3, pp. 311–331, 2004.

[88] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," in *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 407–416, ACM Press, 2000.

[89] D. Beeferman and A. Berger, "Agglomerative clustering of a search engine query log," *In Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 407–415, 2000.

[90] S.-L. Chuang and L.-F. Chien, "Towards automatic generation of query taxonomy: a hierarchical query clustering approach," *Proceedings. 2002 IEEE International Conference on Data Mining ICDM 2002*, pp. 75–82, December 2002.

[91] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, "Clustering user queries of a search engine," in *World Wide Web*, pp. 162–168, 2001.

[92] J. R. Smith and S.-F. Chang, "Visualseek: a fully automated content-based image query system," *ACM Multimedia 96*, Nov 1996.

[93] A. Kak, C. Pavlopoulou, A. Kak, and C. Pavlopoulou, "Content-based image retrieval from large medical databases," in *Proceedings. First International Symposium on Data Processing Visualization and Transmission*, pp. 138–147, 2002.

[94] G. Tao, *Adaptive Control Design and Analysis (Adaptive and Learning Systems for Signal Processing, Communications and Control).* IEEE Computer Society Press, 2003.

[95] S. Liao and M. Pawlak, "On the accuracy of Zernike moments for image analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1358–1364, Dec 1998.

[96] A. Khotanzad and J. Liou, "Recognition and pose estimation of unoccluded three-dimensional objects from a two-dimensional perspective view by banks of neural networks," *IEEE Trans. on Neural Networks*, vol. 7, pp. 897–906, July 1996.

[97] R. H. *et. al.*, "Textural features for image classification," *IEEE Trans. on Syst. Man and Cybern.*, vol. 3, pp. 610–621, March 1973.

[98] M. R. Teague, "Image analysis via the general theory of moments," *Journal of Optical Soc. of Amer.*, vol. 70, pp. 920–930, August 1980.

[99] C. Teh and R. Chin, "On image analysis by the methods of moments," *IEEE Trans. on Pattern Anaysis and Machine Intelligence*, vol. 10, pp. 496–513, July 1988.

[100] A. Khotanzad and J. Lu, "Classification of invariant image representation using a neural network," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 38, pp. 1028–1038, June 1990.

[101] M. Azimi-Sadjadi and S. Stricker, "Detection and classification of buried dielectric anomalies using neural networks–further results," *IEEE Trans. on Instrumentation and Measurement*, vol. 43, pp. 34–39, February 1994.

[102] B. Tian, M. Azimi-Sadjadi, M. Shaike, T. Vonder-Haar, and D. Reinke, "A study of cloud classification with neural networks cloud classification using textural and spectral features," *IEEE Trans. on Neural Networks*, vol. 10, pp. 138–151, January 1999.

[103] T. Chang and C. Kuo, "Texture analysis and classification with tree-structured wavelet tarnsform," *IEEE Trans. on Image Processing*, vol. 2, pp. 429–441, October 1993.

[104] A. Laine and J. Fan, "Texture classification by wavelet packet signatures," *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 15, pp. 1186–1191, November 1993.

[105] A. K. Jain and F. Farroknia, "Unsupervised Texture Segmentation using Gabor Filters," *Pattern Recognition*, vol. 24, pp. 1167–1186, 1991.

[106] L.-K. Soh and C. Tsatsoulis, "Texture Analysis of SAR Sea Ice Imagery Using Gray Level Co-Occurrence Matrices," *IEEE TRANSACTIONS on GEOSCIENCE AND REMOTE SENSING*, vol. 37, March 1999.

[107] R. Agostino and M. A. Stephens, Goodness-of-fit Thechniques. Marcel Dekker, Inc., 1986.

[108] S. W. Zucker and D. Terzopoulus, "Finding structure in co-occurrence matrices for texture analysis," *Comput. Graph. Image Processing*, vol. 12, pp. 286–308, 1980.

[109] A. J. Nevis, "Image enhancement for mine identification with laser line scan sensors," *Third International Symposium on Technology and the Mine Problem, Monterey, CA*, April 1998.

[110] J. S. Taylor and B. Cordes, "Process for the development of image quality metrics for underwater electro-optic sensors," *Proc. of 2002 MTS/IEEE Oceans Conference, Biloxi*, vol. 2, pp. 1003–1009, October 2002.

[111] J. S. Taylor and M. C. Hulgan, "Electo-optic identification research program," *Proc. of 2002 MTS/IEEE Oceans Conference, Biloxi*, vol. 2, pp. 994–1002, October 2002.

[112] M. Azimi-Sadjadi, "Preliminary results on target identication system for stil data set," tech. rep., Submitted to CSS, 2002.

[113] R. Courant and D. Hilbert, *Methods of Mathematical Physics.* Interscience Publishers, 3 ed., 1961.

[114] B. Schölkopf, C. J. Burgues, and A. J. Somola, *Advances in Kernel Methods.* The MIT press, 1999.

[115] A. N. Tikhonov, A. V. Goncharsky, V. V. Stepanov, and A. G. Yagola, *Numerical Methods for the Solution of Ill-Posed Problems (Mathematics and Its Applications, Vol 328).* Kluwer Academic Publishers, August 1995.

[116] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481–1497, 1990.

[117] D. K. Linder, *Introduction to Signals and Systems.* Mc Graw-Hill, 1999.

[118] V. Fedorov, *Theory of optimal experiments.* Academic Press, Inc, 1972.

[119] S. Amari, *Methods of Information Geometry.* American Mathematical Society, 2000.

[120] K. J. Astrom and B. Wittenmark, *Adaptive control.* Addison-Wesley publishing company, 1962.

[121] R. Tibshirani, "Optimal Reinsertion:Regression shrinkage and selection via the lasso," *J.R.Statist. Soc.*, vol. 58, no. 1, pp. 267–288, 1996.

[122] L. Ye and L. Ze-Nian, "Object Extraction and Reconstruction in Active Video," *The 3rd Canadian Conference on Computer and Robot Vision*, no. 1, 2006.

[123] F. Girosi, M. Jones, , and T. Poggio, "Priors, stabilizers and basis functions: from regularization to radial, tensor and additive splines," tech. rep., AI Memo 1430, MIT, 1993.

[124] M. Powell, Approximation'Theory and Methods, ch. 23, pp. 283–298. Cambridge University Press, 1981.