An Associative Memory Applied to Anatomical Analysis of CT Images

by

Daniel Peter Vasilaky

A DISSERTATION

Submitted to

The Graduate Faculty of

The University of Medicine and Dentistry of New Jersey

In Partial Fulfillment

Of the Requirements for the Degree of

Doctor of Philosophy

In Biomedical Informatics

January, 2013

UMDNJ
UNIVERSITY OF MEDICINE & DENTISTRY OF NEW JERSEY

Final Dissertation Approval Form

An Associative Memory Network for Analysis of Anatomical Images

BY

Daniel Vasilaky

Dissertation Committee:

Dinesh P. Mital, Ph.D., Professor and Chairman of the Committee

Syed Haque, Ph.D. Professor and Chair, Department of Health Informatics

Shankar Srinivasan, Ph.D., Associate Professor

Approved by the Dissertation Committee:

[illegible]	Date 1/24/2013
[illegible]	Date 1/24/2013
[illegible]	Date 1/24/13
	Date
	Date
	Date

ABSTRACT

We generalize Kohonen's Optimal Linear Associative Memory (OLAM) neural network model and apply it to the analysis of medical images. More specifically, we apply an OLAM to the identification of anatomical structures and abnormalities in CT images.

OLAM is one of the simplest unsupervised learning models, making it a popular alternative to more complicated neural networks models such as back-propagation or the Hebbian model. However, OLAM has several drawbacks. One is that Kohonen’s solution to the classical OLAM formulation is unstable, in that small changes in the data accompany large changes in the solution. Another drawback of OLAM is its limited memory capacity, as the dimension of training data vectors is not sufficiently large enough. Still another is its computational complexity, as it requires a large number of floating point operations.

In this thesis, we resolve these problems and apply our new algorithms to the detection of anatomical structures in CT images. We address the stability problem by developing an algorithm for a more stable “nearby” problem, drawing on Bellman's theory of dynamic programming.

We remedy the problems of noisy data and memory capacity of OLAM by choosing training data that is not only less noisy but at the same time increases the memory capacity of OLAM.

OLAM relies on the orthogonalization of data vectors. Unfortunately, the classical Gram-Schmidt orthogonalization process, used by Kohonen, performs poorly when the data vectors are nearly linearly dependent, and completely fails when they are dependent. Our algorithm for the solution to the modified OLAM formulation generalizes the Gram-Schmidt process, producing approximately orthogonal vectors even when the data vectors are nearly dependent or dependent.

We conduct recognition experiments on artificially generated CT images, known as the Shepp-Logan phantoms, of cross sections of the human head. The experiments support our model and perform as expected.

There are several advantages in using the modified OLAM model to complement the diagnostic abilities of clinicians. One is that the model captures the collective experience of many experts, i.e. it can store more diagnostic experience than a single physician, as it does not compromise its abilities due to fatigue or stress. It also quantifies the diagnostic results.

ACKNOWLEDGEMENTS

Writing a thesis is a major collaborative effort, and I cannot pay homage to all of the individuals that have contributed to this work. Firstly, however, I would like to thank Dr. Mital for his guidance and patience throughout this process. I would also like to thank Dr. Haque and Dr. Srinivasan for serving on the dissertation committee.

A special thank you is in order to my computer science professors at Long Island University, Florida State University, and the City College of New York, for introducing me to neural networks, and, in particular to Professor Devi for exposing me to dynamic programming, which plays an enormous role in this thesis.

I would also like to thank the professors at Stanford University, especially Dr. Ng, for his courses on machine learning at coursera.org.

Finally, I would like to thank my family for their emotional and financial support.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

1. INTRODUCTION

1.1 Title

An Associative Memory Applied to Anatomical Analysis of CT Images

1.2 Goals and Objectives

Artificial neural networks (ANNs) have been used in a variety of contexts, including medicine.[28–41] For instance, ANNs have been applied to medical image registration, computer-aided detection, and computed-aided diagnoses.[22–47] The effectiveness of ANNs in medical applications is due to their adaptive learning ability. With a suitable learning algorithm, an ANN's ability to learn improves with the variety and the change in the input data.

We focus on an ANN known as associative memory, a content-addressable memory structure. An associative memory models the recollective capabilities of the human brain. It recalls data based on the degree of similarity between the input patterns and the patterns stored in its memory. An advantage of associative memories over other storage algorithms, such as look-up tables, is that they are fault tolerant and robust with respect to variations in the input patterns. For example, an output pattern that is closest to the corresponding correct output pattern may be recalled from a given input pattern, instead of returning "not found", as location addressable memories do.

Kohonen formulated his OLAM model to model the associative memory recall of the brain. The basic form of associative linear recall is a signal transfer operation in a physical network, where a spatial input pattern, the key, transforms into a corresponding output pattern, the recollection. The recall problem is to produce a matrix operator M, by which a pattern $y_k \varepsilon\ R^p$, for every k=1,...,m, is obtained from the pattern $x_k \varepsilon\ R^n$, by Mx_k= y_k, for all k ε {1,2,...,m}. When X and Y known matrices, a formal solution for the unknown M follows from the Penrose pseudoinverse method, the approximate solution $\widehat{M}$ = YX^+, where X^+ is the pseudoinverse of X.[1,7] For an arbitrary matrix X, it is well known that this $\widehat{M}$ is the minimum norm solution to the least-squares problem $\overset{min}{M} ||MX - Y||_2^2$. However, what makes $\widehat{M}$ unsuitable in most applications is that YX^+ is not a continuous function of the data when X is not of full rank.[11] And, even if X is near-rank deficient, that is, its columns are nearly dependent, the solution is sensitive to slight changes in the data, where small perturbations of X and Y may induce large changes in YX^+, making the solution unsuitable in practical applications. To address this problem, we develop an algorithm for the solution to a "nearby" perturbed least-square problem. We then show that the pseudoinverse solution is a special case of our more general solution.

We use Bellman's dynamic programming method to develop our algorithm for the solution to the perturbed least-squares problem.[12] One of our objectives is to be able to compute an $\widehat{M}$ when X is not of full rank or when X is near-rank deficient.

Our first goal is to develop a stable OLAM algorithm that maps input signals into output signals. Recall that unstable systems are very sensitive to noisy data, as small deviations in the input and/or output data accompany very large errors in the solution. To address this problem, we use regularization, where an unstable system gets replaced by a "nearby," more stable system.[19−22] Our algorithm solves the nearby system, yielding a solution that is nearest to the solution of the unstable system.

Regularization methods have been proposed for OLAM models, but they lack efficient, iterative algorithms that solve them.[23−27,49−55] We derive our algorithm using Bellman's principle of optimality.[12] We modify his algorithm, simplify it, and make it more computationally efficient. Later, we show that our modified algorithm generalizes the classical Gram-Schmidt (CGS) orthogonalization process. These results are important not only to our applications, but have far reaching implications in other areas where unstable systems arise.

There are several reasons our algorithms could perform better than physicians in analyzing medical images. Firstly, our OLAM stores the collective memory of many experts, not just the experience of a single clinician. Secondly, our OLAM assigns a numerical measure quantifying the degree of dissimilarity between a given anatomical structure and the structures stored in its memory, a task that often proves difficult for a clinician because of the subjectivity involved.

Our second goal is to provide a data encoding that is relatively noiseless. We do not make use of feature vectors, as is traditionally done with ANNs that analyze medical images. The feature vector approach consists of extracting features from segmented regions in an image, then encoding properties of image segments, such as color, texture, shape, and positional composition, as feature vectors. Instead of this approach, we use the computed Radon transforms of an image to train our OLAM. Recall that a Radon transform of a discretized image is a set of values obtained by summing the products of the intensity of the gray scale times the length of the line segment for each pixel along a ray that that passes through the image. In a CT scan, pencil rays are passed through an object from several angles. Sensors measure the energy lost as each ray exits the object. The X-ray energy decays exponentially, and the exponential coefficient is the Radon transform value for that ray. It is then possible to construct an interior slice of the object, by computing the inverse of the Radon transform.

Our choice of encoding data as Radon transforms has several advantages over feature vectors. We discuss them in no order of importance. Firstly, we reiterate that we may recover an image from a Radon transform by taking its inverse radon transform. Secondly, as Radon transforms are path integrals, or weighted averages that act as smoothing operators, they are far less noisy than feature vectors. Thirdly, Radon transforms do not change under image translations and rotations, as they are invariant with respect to those operations. Fourthly, Radon transforms do not demand the computational complexity of extracting features from each segment and then normalizing the values, since properties such as color and texture require different units of measure. Furthermore, as we discuss later, an OLAM's memory capacity increases with the dimension of the encoded data vectors.[1] As each entry in a vector represents a neuron, increasing the number of neurons increases the memory capacity of OLAM. Radon vectors are typically quite large in terms of dimension. Depending on the number of views and the number of rays in each view of a CT scan, the number of vector components is correspondingly large. In our CT recognition simulation experiments, we use 18 views and 185 rays per view, so our input vectors have a dimension of 18 views times 185 rays = 3,330 components. Our choice of the data encoding is significant not only to reducing noise, but to increasing the memory capacity as well. Naturally, increasing the number of views will produce sharper images, as with the 180 views typically used with CTs, but this requires much more computing power than we have available.

Our third goal is to identify and display anatomical structures in CT images of human head sections. Our heteroassociative method consists of choosing a sufficient and appropriate set of input and output vectors, so our OLAM may recognize a structure when presented with a CT image. Training an OLAM consists of selecting a set of training images and their corresponding anatomical structures, chosen by clinical experts. We choose a training set of Radon transforms of images as the input vectors, and the Radon transforms of anatomical structures as the output vectors. When presented with an image, we apply our algorithm to the Radon transform of the image, obtaining the corresponding Radon transform of the anatomical structure. We then take the inverse of the produced Radon transform to generate the image of the anatomical structure. Of course, the success of this approach depends on the clinical experts' choice of training sets. As experts improve in selecting training sets, our algorithm improves in recognizing anatomical structures in CT images.

1.3 Statement of Problem, Background, Significance, and Hypotheses

The significance of a computer-based system's ability to identify, locate, and display anatomical structures in CT images lies in its applicability to applications such as diagnosis, surgery, and laser treatments. Since the location and appearance of an anatomical structure vary among patients, a computer-based system that identifies, locates, and displays a given individual's anatomical structure is essential. We later perform a set of simulation experiments testing the various assumptions made with regard to our choice of training data.

Our first hypothesis is that our generalized OLAM consistently identifies, retrieves, and measures the abnormality of anatomical structures in CT images. Recall that a system capable of doing this must be able to process noisy measurement data and large numerical data sets. Our model has the ability to orthogonalize large sets of vectors that may be nearly linearly dependent or dependent. As stated by Kohonen et al. in [10]:

> "In the theory of associative memory and associative information processing, the projection principle and subspaces are used in explaining the optimality of associative mappings and novelty filters..... To facilitate operations connected with subspaces, it is most economical to choose the basis vectors orthonormal. If only a non-orthonormal set is available in the first place, they can be normalized by the Gram-Schmidt orthogonalization method."

However, the Gram–Schmidt method typically displays severe loss of orthogonality for large sets of vectors, failing when the input and/or output vectors are nearly dependent or dependent. A projection operator is our principle tool. The projection of any vector x onto a subspace is the closest point in the subspace to the vector x. Kohonen proposed using the pseudoinverse to compute the projection operator. However, the pseudoinverse is unstable when the input and output matrices are made up of either nearly dependent vectors or dependent vectors.[11] We use the projection operator to project a signal vector onto a subspace of training set vectors. The most efficient way to store a subspace of vectors in a computer memory is to store its orthonormal basis vectors. If the orthonormal basis is u$_1$,u$_2$, ... , u$_n$, we may compute the projection operator P that projects any vector x onto the subspace spanned by the orthonormal vectors: $Px = \widehat{x} = \sum_{i=1}^{n}(u_i^T x)u_i = (\sum_{i=1}^{n}u_i u_i^T)x.$

We show that $\sum_{i=1}^{n}u_i u_i^T$ is a projection operator that projects any vector x onto the subspace spanned by the orthonormal basis u$_1$,u$_2$, ...,u$_n$. A problem arises in obtaining an orthonormal basis for a subspace. Our second hypothesis is that our generalized orthogonalization algorithm yields a desired basis.

An alternative to using the above projection operator is to approximate the projection operator XX$^+$, with X$^+$ the pseudoinverse of a matrix of input vectors X. As stated earlier, Kohonen's pseudoinverse method is unstable. We remedy the instability with an algorithm that computes a stable approximation of XX$^+$.

Our third hypothesis is that our algorithm computes a projection operator that is a stable approximation of our desired projection operator. Furthermore, we show the approximation can be computed arbitrarily close to the desired projection operator.

Our fourth hypothesis is that our algorithm correctly identifies, locates, and displays anatomical parts in a CT computation of image.

We emphasize that our algorithms are designed to detect and retrieve global features, such as retrieving anatomical structures, or determining the probability of a structure's abnormality. Our algorithms are not capable of identifying image properties such as color or texture. Researchers typically apply segmentation to images, then extract features from each segment to encode local details.[23–43] However, it is nonetheless useful to have relatively simple algorithms that can identify, retrieve, and display anatomical structures without resorting to segmentation and feature extraction. To support our fourth hypothesis, we perform some preliminary recognition experiments on the Shepp-Logan phantoms, getting encouraging results. However, we need to undertake more experimentation to evaluate the performance of our algorithms.

Our fifth hypothesis is that our algorithm measures the probability that an anatomical structure in a CT image is abnormal. This requires we have a sufficiently large set of encoded vectors of normal structures. The training set is continuously updated as our OLAM encounters more normal versions of the structure, as bias is introduced by the choices made by experts as to which structures to include in the training set. The most convincing demonstration that our algorithm correctly distinguishes normal from abnormal structures is our experimentation on cases where the correct answers are known in advance.

Our training set consists of Radon transforms of "normal" anatomical structures. Clinical experts choose a sufficient set of images representing normal anatomical structures in CT images. We compute the Radon transforms of each anatomical structure in the image to form a subspace of "normal" Radon vectors. We use our algorithm to project the Radon transform vector of a given structure onto the "normal" subspace. We then apply our abnormality measure on the projected vector to quantify to what extent it is normal.

Furthermore, we show that the discrete Radon transform vectors are multivariate Gaussian random vectors. This enables us to employ statistical methods to derive the threshold for a given false-positive probability, so that we can determine that an anatomical structure is abnormal if the abnormality measure falls below the threshold value.

Chapter 2

2. LITERATURE REVIEW

2.1 An Overview of Neural Networks

Neural networks are adaptive models, formally developed in the 1960s. They originated through the work of McCullough and Pitts, who developed mathematical models based on observational studies of neurons in human brains.[61] Neural networks "learn" or adapt via two means, supervised and unsupervised learning. They provide an alternative to statistical models, where a functional form or an underlying distribution is assumed about the data. If the assumption is incorrect, the model gives errors in the results, such as in prediction, classification, and functional relation estimation. Supervised learning employs an external supervisor, where each output unit is compared with the supervisor's response to the inputs. Synaptic weights are adjusted to reduce the error based on the difference between the network's and supervisor's responses. Backpropagation remains a popular, yet not always practical, means of supervised learning.[74] However, it has some limitations that we list in no order of importance. Firstly, the convergence obtained from computation of backpropagation is very slow. Secondly, the convergence in backpropagation learning is not guaranteed. Thirdly, backpropagation learning requires input scaling or normalization. Fourthly, convergence may occur only at local minima.

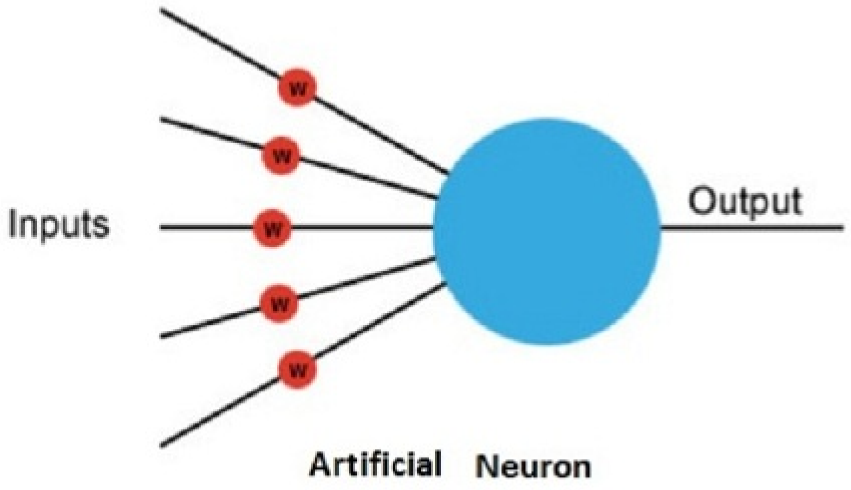


Fig. 1: A symbolic representation of a neuron.

Recall that the human brain consists of about 100 billion units called neurons. Each neuron is connected to thousands of others, communicating via electrochemical signals. Signals coming into the neuron are received via junctions called synapses, located at the end of branches of the neuron cell, the dendrites. A neuron continuously receives signals from these inputs, summing a fan-in of inputs, then firing if the sum exceeds some threshold value. Firing generates a voltage that outputs a signal along an axon.

Every input into the neuron has a weight associated with it, a scalar that can change during network training. The weights may be positive or negative, providing excitatory or inhibitory influences to each input. As each input enters the nucleus, it is multiplied by the weight. The nucleus sums all these new input values, giving an activation, a scalar that may be negative or positive. If the activation is greater than a threshold value, say, for example 1, the neuron outputs a signal. If the activation is less than 1, the neuron outputs nothing.

Associative recall consists of a signal transfer operation in a physical network, where an input pattern, the key, transforms into a corresponding output pattern, the recollection[1]. The recall problem is to produce a matrix M, where a pattern vector $y_k \varepsilon\ R^m$, is recalled from the pattern $x_k \varepsilon\ R^n$, via Mx_k= y_k, for all k ε {1,2,...,m}. The matrix M$^{m \times n}$ contains synapses between two fields of neurons, F$_X$ and F$_Y$. An element of M is μ_{ij}, the synaptic connection from the ith neuron in F$_X$ to the jth neuron in F$_Y$. The vector x_i is the state of F$_X$, while the vector y_i is the state of F$_Y$. The sign of μ_{ij} determines whether the synaptic connection is excitatory, with $\mu_{ij} > 0$, or inhibitory, with $\mu_{ij} < 0$. The magnitude of μ_{ij} records the strength of the connection. The m-vector Mx_i is a fan-in vector of input sums to the neurons in F$_Y$. The input to y_j is I$_{y_j}$=$\mu_{j1}\xi_{j1}$ +$\mu_{j2}\xi_{j2}$ +...+$\mu_{jn}\xi_{jn}$, where x_j=[ξ_{j1} ,ξ_{j2},...,ξ_{jn}]^T is the jth input vector. Neuron j processes input I$_{y_j}$to produce the output signal S(I$_{y_j}$). In general, the signal function S is nonlinear, usually sigmoidal or S-shaped.

Recall that unsupervised learning uses no external supervisor. Hence, unsupervised neural networks do not assume that the training data has some functional relationship or underlying distribution. Thus, training does not involve altering the synaptic weights that are based on the distance of the output from an expected output. An unsupervised ANN learns to group or cluster patterns in the inputs, based on common

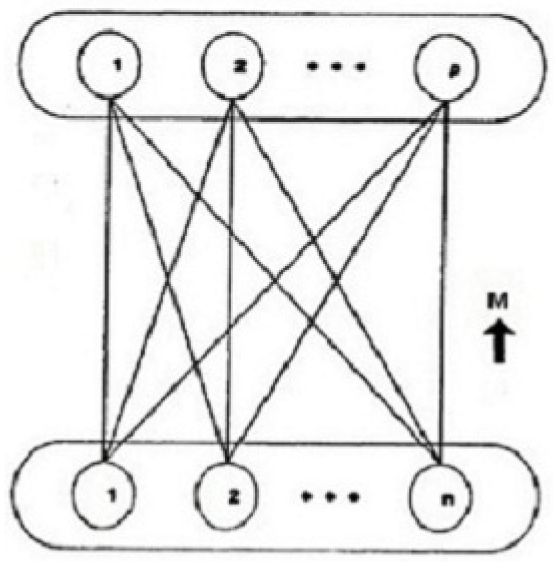


Fig. 2: A single layer feed forward network.

features, similar to statistical methods such as factor analysis and principal component analysis (PCA)[62–63]. In general, unsupervised learning is used when there is no way of determining how dissimilar the output is from an expected output.

2.2 The Use of Neural Networks in Health Care

Some applications of ANN's to medicine have been to clinical diagnoses, image analysis and interpretation, signal analysis and interpretation, and drug development.[75–80] The FDA has also approved several diagnostic tests using neural networks. For instance, Papnet is a commercial neural network based computer program that assists in the screening of Pap (cervical) smears[86–87]. Several studies show that it performs better than doctors. Open Clinical, an international organization that promotes awareness and use of decision support, clinical work-flow, and other advanced knowledge management technologies for patient care and clinical research, reports at the URL http://www.openclinical.org, that

> "Relying on manual inspection alone makes it inevitable that some abnormal Pap smears will be missed, no matter how careful the laboratory is. In fact, even the best laboratories can miss from 10% - 30% abnormal cases. Papnet-assisted reviews of cervical smears result in a more accurate screening process than the current practice, leading to an earlier and more effective detection of pre-cancerous and cancerous cells in the cervix".

Another study reveals that

> "A research group at University Hospital, Lund, Sweden tested whether neural networks trained to detect acute myocardial infarction could lower this error rate. They trained a network using ECG measurements from 1120 patients who suffered a heart attack, and 10,452 healthy persons with no history of heart attack. The performance of the neural networks was then compared with that of a widely used ECG interpretation program and that of an experienced cardiologist. Neural networks were 15.5% more sensitive than the interpretation program and 10.5% more sensitive than the cardiologist in diagnosing any abnormalities. But the

> cardiologist was slightly better at recognizing ECGs with very clear-cut acute myocardial infarction changes."[82]

A considerable body of literature surveys the use of neural networks in image processing, for instance, Egmon-Petersen et al.[81] Most of the cited papers use feature vectors as training data, as opposed to our use of discrete Radon transforms as training data.

One of the most popular neural network models for image reconstruction is the Hopfield.[69–73] It solves an optimization problem by letting the network converge to a stable state while minimizing an energy function. A drawback of Hopfield networks is that they may become trapped in non-optimal, local minima. To circumvent this problem, researchers must use other methods, such as genetic programming, adding to the complexity of the Hopfield network's algorithm. We choose Kohonen's OLAM over models such as the Hopfield for its simplicity, as OLAM is a linear model.[1]

Kohonen et al.[27] train an OLAM to identify abnormal brain images. However, their results are not always consistent, as the CGS method they use does not always give reliable results. As is well-known, the CGS process is unstable or fails completely when the input vectors are near-rank or rank deficient. We remedy this problem by developing a more stable orthogonalization process than the CGS. We apply our generalized CGS process to identify abnormal anatomical structures in CT images. Next, we develop a probability measure quantifying the degree of abnormality.

2.3 Associative Memory

Several associative neural memory models have been proposed over the last two decades.[59,99–101,102] The aim of associative memory networks is to retrieve a previously learned pattern from an example that resembles the previously presented patterns. These patterns are typically represented as real-valued vectors. Associative memory networks are able to retrieve a previously learned pattern from an example similar to a pattern in the data. We may classify these models as static or recurrent, and, by the nature of the stored associations, as either autoassociative or heteroassociative. Our model can be classified as static, and is capable of autoassociative and heteroassociative stored associations. Most models of associative memory are recurrent types such as the Hopfield model. The Hopfield model is different from OLAM in that it computes its output recursively until the system becomes stable. Unlike the OLAM model, which consists of two layers of processing units, one as the input and the other output, the Hopfield model consists of a single layer of processing elements, where each unit other than itself is connected to every other unit in the network. The connection weight matrix W of this type of network is square symmetric, i.e., $w_{ij} = w_{ji}$ for i, j = 1, 2, ..., m. A Hopfield model performs a recursive search for a stored pattern when given an initial pattern. The process is iterated until the network settles on a pattern, so that further computations do not change the output. However, the models based on the Hopfield have the drawback that the solution may be spurious when the cost function ends up in a local minimum. Hopfield shows that such networks are characterized by an energy function by which stored memories correspond to local energy minima.[59] He demonstrates that the maximum number of patterns that can be stored with a single layer feed-forward network model of n nodes, before the error in the retrieved pattern becomes severe, is around 0.15n.

Another associative memory model is the Bidirectional Associative Memory (BAM).[7,101] It is similar to that of the associative memory model, but the connections are bidirectional, i.e., $w_{ij} = w_{ji}$, for i = 1, 2, ..., m, and j = 1, 2, ..., n. The memory capacity of BAM is about 0.1998n. Since BAM uses binary-valued associations, it is not useful for our applications.[105] proposes a BAM model based on OLAM. Unlike many existing BAM algorithms, the presented BAM uses an optimal associative memory matrix, determined only by using correlation learning, and requiring no pseudoinverse calculation. However, this BAM also assumes only binary-valued associations.

Our modified OLAM model is static, linear, and much simpler than the Hopfield type models. However, OLAM performs well only when the data is highly dimensional relative to the sample size.[1] shows that for linearly independent input vectors or even approximately independent ones, the OLAM reduces the input noise when the number of observations is small, relative to the dimension of the input data.

2.4 The Problem of Noisy Data

We reiterate that Kohonen's generalized pseudoinverse solution for OLAM is extremely sensitive to noisy data.[1] Researchers have proposed several models to remedy this problem, most of which perturb the least-squares formulations associated with the generalized pseudoinverse inverse solution.[23–26] Each of the cited studies proposes a slightly different perturbation. In each study, the authors show their model is more stable than the generalized inverse model. However, none of the studies offers an iterative algorithm that computes the solution to the corresponding perturbed problem. A regularization method worth mentioning is the Truncated Singular Value Decomposition (TSVD).[16,18–19] In TSVD, any input matrix X may be written in SVD form, namely $X = U\sum V^T$. The matrix U is orthogonal and has as many rows as X. The matrix V is orthogonal and has as many columns as X. The matrix $\sum$ is the same size as X, containing nonzero elements on its main diagonal. These diagonal elements are the singular values of $\sum$. The columns of U and V are the left and right singular vectors, respectively.

We may expand an ill-conditioned matrix X as a sum of rank-1 matrices, or $X = \sum_{k=1}^{n} \sigma_k E_k$. E_k is an outer product of the kth left and right singular vectors, so $E_k = u_k v_k^T$. The singular values of X decrease, that is, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$. If X's columns are Radon transform vectors, we can truncate the terms in the above sum after r terms, getting a rank-r approximation to the original matrix with less noise. The error in the approximation depends upon the magnitude of the neglected singular values.

The Tikhonov and TSVD regularizations yield similar results.[72] In Section 4.5, we perform experiments using TSVD, getting almost the same results as our Tikhonov approach. However, our algorithm is computationally more efficient than TSVD. For $n \approx m$, our algorithm is at least twice as fast as the TSVD-based algorithms of [18] and [19] that use $3n^3$ to $5n^5$ flops, as our algorithm needs only $2n^3$ flops. Our algorithm also generalizes the CGS process. We may apply it to compute the novelty filter to identify anatomical features in medical images.

In "real world" scenarios, the input training matrices are extremely large and are likely ill-conditioned. Our algorithm more stably computes the solution to the perturbed least-square problem. Furthermore, it is more efficient than the cited algorithms.

2.5 Dynamic Programming

Dynamic programming is applicable to problems that can be formulated recursively in terms of nested subproblems. Often, these subproblems are really the same, but of lower dimension. Dynamic programming is based on Richard Bellman's Principle of Optimality[14,12] :

> "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."

The book of Bellman et al. in[12] plays a central role in this work. The recursive equations for the Tikhonov regularization problem in[15–22] were formulated and solved in[12]. However, these equations and their solution are so complex that they attracted little notice. We not only simplify the recursive equations but also show that the simplified equations generalize the Gram-Schmidt process equations for rank deficient and near rank deficient sets of vectors. The derived generalized Gram-Schmidt process can be written in terms of simple expressions consisting of quotients, with an outer product in the numerator, and an inner product plus the regularization constant in the denominator. The simplified equations reduce to the Gram-Schmidt method when the vectors are of full rank and the regularization constant is set to zero. We use our pseudo-orthogonalization method to approximate the transition matrix for the associative memory. This differs from Kohonen's pseudoinverse computations of the transition matrix for an associative memory[1,2–6,8,9,25,26,51–55] .

2.6 Algorithms for Ill-Conditioned Problems

Tikhonov regularization is often used to model ill-conditioned problems. In statistics, this is known as solving the ridge regression problem. Much literature has been devoted to it.[6-14] In its standard form, Tikhonov regularization is

$$\underset{x}{min}\,(||Ax - b||_2^2 + \delta||x||_2^2, \delta > 0. \qquad (1)$$

The optimal regularization parameter δ is typically not known, and is often determined by best fitting the data. However, doing this in our applications may result in over-fitting the data. We must determine our parameter δ through cross-validation, a technique that assesses how well a model generalizes an independent data set. We discuss cross-validation, as it pertains to analyzing CT images, in Section 4.9.

Bellman suggests how to derive an iterative algorithm solving (1).[12] But, he does not provide an explicit derivation. We give an exact derivation in the next section.

Chapter 3

3. RESEARCH METHODOLOGY

3.1 Modified OLAM Model

We wish to modify Kohonen's optimal associative mappings in the way we compute the matrix M. We apply an estimate of M to heteroassociative associative memory, that is, when F_X and F_Y are distinct, as well as to autoassociative associative memory, when F_X and F_Y are the same.

Using OLAM, we estimate the transfer matrix M as $\widehat{M} = YX^+$.[1]

The m×n matrix X^+denotes the pseudoinverse of X. If the matrix X is of full rank, then $X^+=X^T(XX^T)^{-1}$, and the transition matrix $\widehat{M}$ minimizes the LS problem of forward recall, namely

$$\widehat{M}=\underset{M}{min}\,||MX-Y||_{2.}^2 \tag{2}$$

On the other hand, we do not assume the matrix X is of full rank. We compute an estimate of the transfer matrix M as the solution to the regularized LS problem $\underset{M}{min}(||MX-Y||_2^2+\delta||M||_2^2,\text{for } \delta > 0.$

The regularization assures us of a stable solution in providing an estimate of $\widehat{M} = YX^+$.

Recall that with autoassociative recall, the input matrix X and output matrix Y are the same, i.e. X = Y. Let x_i be a vector formed by concatenating the rows of an image's matrix. We may construct a projection operator $\widehat{M}$ for the corresponding class of inputs, so that any pattern in the class is a projection onto the subspace spanned by the vectors $x_1, x_2, ..., x_n$.

That is, for an image x, $\hat{x} = \widehat{M}x$. Since the Radon transform of an image recovers the image, instead of using the ith feature vector of an image, we use the Radon transform of the image as the input signal x_i. In working with the Radon transforms of an image instead of the image itself, or with feature vectors extracted from the image, we have a simpler model, with less noisy data. Recall that a Radon transform is a path integral. It is a weighted average in the discrete case, and is a smoothing operator.

There are many ways to construct $\widehat{M}$. The simplest is the Hebbian approach that sets the connection weights as the sum of the outer-product matrices of the input vectors $x_1, x_2..., x_n$,

$$\widehat{M} = \sum_{i=1}^{n} x_i x_i^T. \tag{3}$$

If x_i is a vector formed by concatenating rows of a face image, for instance, the matrix $\widehat{M}$ contains the covariance of possible pairs of pixels in the set of learned faces.[13] Because the cross-product connection weight matrix is semidefinite, it may be written as a linear combination of its eigenvectors as

$$\widehat{M} = \sum_{i=1}^{r} \lambda_i u_i u_i^T = U \Lambda U^T, \tag{4}$$

where u_i and λ_i denote the ith eigenvector, and the corresponding eigenvalue of $\widehat{M}$, respectively. In the matrix U, the ith column is u_i, and Λ is the diagonal matrix of eigenvalues, of rank r. The eigenvectors of the weight matrix can be thought of as a set of “global features” or “macro-features” from which a face is built. A significant drawback of Hebbian autoassociative memory or PCA is that the memory may commit too many errors, as the feature data is noisy.[13] However, our algorithm, which approximately orthogonalizes the input vectors, eliminates most of this cross talk. So we may apply the Hebbian-type memory by approximately orthonormalizing the Radon transform input vectors $x_1, x_2 ..., x_n$. Next, we compute the sum of the outer-products $\widehat{M} = \sum_{i=1}^{n} q_i q_i^T$, where q$_i$ are the orthonormalized input vectors. The vector $\widehat{M}$x $= (\sum_{i=1}^{n} q_i q_i^T) x = \sum_{i=1}^{n} (q_i^T x) q_i$ is an expansion of x in terms of the basis q$_i$. $\widehat{M}$x is a projection of x onto the subspace spanned by q$_1, ..., q_n$. We may also compute the transition matrix $\widehat{M}$ for autoassociative memory by approximating the projection matrix M=XX$^+$.

We derive and simplify Bellman’s algorithm, showing our simplified algorithm generalizes the CGS process. We also show that our solution to (1) generalizes the QR factorization solution to the LS problem. Finally, we determine the best choice of the regularization parameter by the cross-validation method discussed in Section 4.9.

3.2 The Algorithms

Consider the least-squares (LS) problem.

Let $A^{m\times n}$=$(\zeta_{ij})^{m\times n}$, and the vector b=$(\gamma_i\ )\varepsilon R^m$. Find the vector x=$(\xi_j)\ \varepsilon R^n$ that satisfies (1). We rewrite the LS problem as

$$f_n(b) = \underset{\xi_1,\xi_2,\ldots,\xi_n}{min} (\sum_{i=1}^{m}(\sum_{j=1}^{n}\zeta_{ij}\xi_j - \gamma_i)^2 + \delta\sum_{j=1}^{n}\xi_j^2). \tag{5}$$

Now, for k=1,2,...,n let

$$f_k(b) = \underset{\xi_1,\xi_2,\ldots,\xi_k}{min} (\sum_{i=1}^{m}(\sum_{j=1}^{k}\zeta_{ij}\xi_j - \gamma_i)^2 + \delta\sum_{i=1}^{k}\xi_j^2). \tag{6}$$

By the principle of optimality of [12], we may rewrite our recurrence relation, for $k \geq 2$, as

$$f_k(b) = \min_{\xi_k}(f_{k-1}(b - \xi_k A(:,k)) + \delta\xi_k^2),\ \delta > 0. \tag{7}$$

Above, we replace the vector y with $b - \xi_k A(:,k)$ in f_{k-1}, minimizing over the set $\{\xi_1,\xi_2,\ldots,\xi_{k-1}\}$. A(:,k) is the kth column vector of A. We make use of the following lemma.

Lemma 1. $f_k(b)$ is a quadratic form in b, having the form $b^T V_k b$, where V_k is a square matrix independent of b.

Proof: Denote by $A^{m\times(n-k)}$ the A matrix with the last n-k columns replaced by zeros. The vector x that minimizes (7) is a linear function of y, namely

$$\hat{x} = (A^{m\times(n-k)T} A^{m\times(n-k)} + \delta I)^{-1} A^{m\times(n-k)T} y. \tag{8}$$

Denote

$$(A^{m\times(n-k)T} A^{m\times(n-k)} + \delta I)^{-1} A^{m\times(n-k)T}$$

by matrix B$^{m\times m}$=(β_{ij}). Then (1) becomes

$$\hat{x} = ||A^{m\times(n-k)}\hat{x} - y||^2 + \delta||\hat{x}||^2$$

$$= ||A^{m\times(n-k)}By - y||^2 + \delta||By||^2$$

$$= ||(A^{m\times(n-k)}B - I)y||^2 + \delta||By||^2$$

$$= ((A^{m\times(n-k)}B - I)y)^T (A^{m\times(n-k)}B - I)y) + \delta(By)^T(By)$$

$$= y^T((A^{m\times(n-k)}B - I)^T(A^{m\times(n-k)}B - I) + \delta B^T B)y. \tag{9}$$

The last expression is a quadratic form $y^T V_k$y, where V_k is an $m \times m$ matrix independent of y.

We introduce the following theorem.

Theorem 1. The following recurrence relation holds for the square matrix V_k, for k=1,2,...,n:

$$V_k = V_{k-1} - \frac{(V_{k-1}A(:,k))(A(:,k)V_{k-1})^T}{\delta + A(:,k)^T V_{k-1} A(:,k)}, k = 1, ..., n. \tag{10}$$

Bellman et al. state but do not derive Theorem 1 in chapter 5 of [12]. We derive a new recurrence relation below. The numerator in the above expression is an outer product of the column vector $V_{k-1}A(:,k)$ and the row vector $(V_{k-1}A(:,k))^T$. The denominator is a scalar. So, the rightmost term above is a projection matrix.

We derive our recurrence relation by replacing $f_{k-1}(b - \xi_k A(:,k))$ with

$$(b - \xi_k A(:,k))^T V_{k-1}(b - \xi_k A(:,k)), \tag{11}$$

getting

$$f_k(b) = \min_{\xi_k}((b - \xi_k A(:,k))^T V_{k-1}(b - \xi_k A(:,k)) + \ \delta\xi_k^2). \tag{12}$$

We take the derivative with respect to ξ_k, then set it equal to zero, getting $\delta\xi_k^* -$

$(b - \xi_k^* A(:,k))^T V_{k-1} A(:,k) = 0$, where ξ_k^* is the minimal value. Solving for ξ_k^*, we get

$$\xi_k^* = \frac{b^T V_{k-1} A(:,k)}{\delta + A(:,k)^T V_{k-1} A(:,k)}. \tag{13}$$

Now

$$\begin{aligned}
f_k(b) &= b^T V_k b \\
&= (b - \xi_k^* A(:,k))^T V_{k-1}(b - \xi_k^* A(:,k)) + \ \delta \xi_k^{*2} \\
&= (b - \xi_k^* A(:,k))^T V_{k-1} b - \xi_k^* (b - \xi_k^* A(:,k))^T V_{k-1} A(:,k) + \delta \xi_k^{*2} \\
&= (b - \xi_k^* A(:,k))^T V_{k-1} b + \xi_k^* (\delta \xi_k^* - (b - \xi_k^* A(:,k))^T V_{k-1}) A(:,k)) \\
&= (b - \xi_k^* A(:,k))^T V_{k-1} b \\
&= (b^T - \xi_k^* A^T(:,k)) V_{k-1} b \\
&= b^T V_{k-1} b - \xi_k^* A^T(:,k) V_{k-1} b \\
&= b^T V_{k-1} b - \frac{b^T V_{k-1} A(:,k)}{\delta + A^T(:,k) V_{k-1} A(:,k)} A^T(:,k) V_{k-1} b \\
&= b^T V_{k-1} b - \frac{b^T (V_{k-1} A(:,k))(V_{k-1}\text{A(:,k)})^T b}{\delta + A^T(:,k) V_{k-1} A(:,k)} \\
&= b^T (V_{k-1} - \frac{(V_{k-1} A(:,k))(V_{k-1} A(:,k))^T}{\delta + A^T(:,k) V_{k-1} A(:,k)}) b
\end{aligned}$$

Note that $\delta \xi_k^* - (b - \xi_k^* A^T(:,k))^T V_{k-1} A(:,k) = 0$ in the fourth line, which we substitute into the seventh line.

We have shown that our recurrence relation for the square matrix $\mathrm{V_k}$ is the same as that of Bellman et al. in[12]. However, we have gone a step further by making a change of variables, and showing that our recurrence relation generalizes the CGS. The generalization of CGS is a significant discovery and is an important contribution to matrix computation.

Theorem 2. Let the matrix V_0= I, and the column vector q$_k$(δ) $= V_{k-1}$(δ)A(:,k), for k=1,2,...,n. Using the recurrence relation for the matrix V$_k$ gives

$$q_1 = IA(:,1), \tag{14}$$

,

$$q_2(\delta) = V_1(\delta)A(:,2) = (I - \frac{q_1 q_1^T}{\delta + A^T(:,1)q_1})A(:,2), \tag{15}$$

$$q_3(\delta) = V_2(\delta)A(:,3)$$

$$= (I - \frac{q_1 q_1^T}{\delta + A^T(:,1)q_1} - \frac{q_2(\delta)q_2^T(\delta)}{\delta + A^T(:,2)q_2(\delta)})A(:,3), \tag{16}$$

In general,

$$q_k(\delta) = V_{k-1}(\delta)A(:,k)$$

$$= (I - \sum_{j=1}^{k-1} \frac{q_j(\delta)q_j^T(\delta)}{\delta + A^T(:,j)q_j(\delta)})A(:,k), k = 1, ..., n. \tag{17}$$

.

3.2.1 The Generalized Gram-Schmidt Process

Theorem 3. Let $A^{m\times n}$=$(\zeta_{ij})^{m\times n}$ be a matrix with n independent columns, and consider the derivation above of the column vector $q_k(\delta)$, k=1,2,...,n. If we set δ=0, the derivation of $q_k(0)$ is equivalent to the CGS.

Proof. Denote $q_k(0)$ by q_k, with δ=0. We show that $A^T(:,k)q_k = q_k^T q_k$. It is fairly easy to see that the column vector

$$q_k = (I - \sum_{j=1}^{k-1} \frac{q_j q_j^T}{q_j^T q_j})A(:,k)$$

$$= A(:,k) - \sum_{j=1}^{k-1} \frac{q_j^T A(:,k)}{||q_j||} \frac{q_j}{||q_j||},\ for\, k = 1, ..., n, \tag{18}$$

is identical to that of the CGS form. A proof follows.

$A^T(:,k)q_k = q_k^T q_k$ is trivially true for k=1, as $q_1 = IA(:,1)$.

So, it is true for k = 2 also, as $q_2^T q_2 = A^T(:,2)(I - \frac{q_1 q_1^T}{q_1^T q_1})(I - \frac{q_1 q_1^T}{q_1^T q_1})A(:,2)$.

But, $(I - \frac{q_1 q_1^T}{q_1^T q_1})$ is a projection matrix, and is idempotent, so

$$q_2^T q_2 = A^T(:,2)(I - \frac{q_1 q_1^T}{q_1^T q_1})A(:,2) = A^T(:,2)q_2.$$

For 1 $\leq$ k $\leq$ n, $(I - \sum_{j=1}^{k} \frac{q_j q_j^T}{q_j^T q_j})$ is an idempotent matrix. So

$$q_k^T q_k = A^T(:,k)(I - \sum_{j=1}^{k-1} \frac{q_j q_j^T}{q_j^T q_j})(I - \sum_{j=1}^{k-1} \frac{q_j q_j^T}{q_j^T q_j})A(:,k)$$

$$= A^T(:,k)(I - \sum_{j=1}^{k-1} \frac{q_j q_j^T}{q_j^T q_j})A(:,k) = A^T(:,k)q_k. \tag{19}$$

Therefore,

$$q_k = (I - \sum_{j=1}^{k-1} \frac{q_j q_j^T}{q_j^T q_j})A(:,k), k = 1, 2, ..., n \tag{20}$$

is an orthogonalized column of the square matrix A.

Note that for an ill-conditioned matrix A, it is tempting to use the CGS on $\left[\begin{smallmatrix} A \\ \sqrt{\delta} I \end{smallmatrix}\right]$ to get an approximate orthogonalization of A. Since the matrix $\left[\begin{smallmatrix} A \\ \sqrt{\delta} I \end{smallmatrix}\right]$ is an (m+n)-by-n matrix, it takes 2(m+n)n^2 flops to do this, or roughly 2n^3 flops. There is also the n^2 memory needed for the scalar $\sqrt{\delta} I$. Finally, it is not clear how to obtain the solution to (1) when orthogonalizing the columns of $\left[\begin{smallmatrix} A \\ \sqrt{\delta} I \end{smallmatrix}\right]$.

Furthermore, if we no longer assume the columns of A are linearly independent, the columns $q_k(\delta)$ are still nearly orthogonal, in the sense that $q_l^T(\delta) q_k(\delta) \approx 0$, for relatively small $\delta > 0$.

We introduce the following theorem.

Theorem 4. Let $A^{m\times n}$=$(\zeta_{ij})^{m\times n}$, b=(γ_i) ε R^m , x=(ξ_j) ε R^n, and $q_k(\delta)$ be as above, for k=1,...,n. The vector that minimizes $\underset{x}{min}(||Ax-b||^2+\delta||x||^2|),\ \delta>0$ is

$$\hat{x}(\delta)=[\hat{\xi}_1(\delta),\hat{\xi}_2(\delta),...,\hat{\xi}_n(\delta)]^T, \tag{21}$$

where

$$\begin{aligned}\hat{\xi}_n(\delta) &= \frac{q_n^T(\delta)}{\delta+A^T(:,n)q_n(\delta)}b,\\ \hat{\xi}_{n-1}(\delta) &= \frac{q_{n-1}^T(\delta)}{\delta+A^T(:,n-1)q_{n-1}(\delta)}(b-\hat{\xi}_n(\delta)A(:,n)),\end{aligned}$$

$$...$$

$$\hat{\xi}_{n-k}(\delta)=\frac{q_{n-k}^T(\delta)}{\delta+A^T(:,n-k)q_{n-k}(\delta)}(b-\sum_{i=k}^{n-(k-1)}\hat{\xi}_i(\delta)A(:,i)),k=0,...,n-1. \tag{22}$$

Proof: Substituting the following expression for $\mathrm{f}_{n-1}(y)$ into (7) gives

$$(b-\xi_nA(:,n))^TV_{n-1}(b-\xi_nA(:,n)). \tag{23}$$

Then, for k = n,

$$\underset{\xi_n}{min}((b-\xi_n A(:,n))^T V_{n-1}(b-\xi_n A(:,n))+\delta\xi_n^2). \tag{24}$$

Since this is a minimization of a quadratic function with respect to ξ_n, we may it differentiate with respect to ξ_n. We set the derivative equal to zero, getting

$$\hat{\xi}_n(\delta)=\frac{q_n^T(\delta)}{\delta+A^T(:,n)q_n(\delta)}b. \tag{25}$$

Recall that the column vectors $q_k(\delta) = V_{k-1}(\delta)A(:,k)$ and the matrix $V_k(\delta)$ are symmetric, for k=1,2,...,n. Thus, the column vector $A^T(:,n)V_{n-1} = A^T(:,n)V_{n-1}^T = (V_{k-1}(\delta)A(:,k))^T = q_n^T(\delta)$. To compute $\hat{\xi}_{n-1}(\delta)$, we minimize the following quadratic function:

$$(b-\hat{\xi}_n A(:,n)-\xi_{n-1}A(:,n-1))^T V_{n-2}(b-\hat{\xi}_n A(:,n)-\xi_{n-1}A(:,n-1)), \tag{26}$$

Minimizing with respect to ξ_{n-1} gives

$$\hat{\xi}_{n-1}(\delta)=\frac{q_{n-1}^T(\delta)}{\delta+A^T(:,n-1)q_{n-1}(\delta)}(b-\hat{\xi}_n(\delta)A(:,n)). \tag{27}$$

We continue this way to obtain $\hat{\xi}_{n-k}(\delta)$. Namely,

$$\hat{\xi}_{n-k}(\delta) = \frac{q_{n-k}^T(\delta)}{\delta + A^T(:, n-k)q_{n-k}(\delta)}(b - \sum_{i=n}^{n-(k-1)} \hat{\xi}_i(\delta)A(:, i)), k = 2, ..., n-1. \tag{28}$$

Note we never compute the matrix V_k to obtain the solution to the Tikhonov regularization problem, as in[12]. Instead, we compute the column vectors, $q_k(\delta)$, for k=1,...,n. This is a more efficient method than that of [12], revealing an orthogonalization process.

The computation of the column vectors $q_k(\delta)$, that is, the approximation of the orthogonalization of the matrix A's columns, is our algorithm's most costly procedure, requiring at least $2\mathrm{m}^2 n$ flops. The remaining computations are of the order $m \times n$, which we did not include in estimating the number of flops.

It is useful to compare the QR factorization solution to the LS problem, as well as our solution to the Tikhonov problem. CGS orthogonalizes the columns $q_k(0) = q_k$, for k=1,...n .

Thus $q_k(0) \perp \mathcal{L}(q_1(0), ..., q_{k-1}(0)\ )$. Let $\frac{q_k(0)}{||q_k(0)||}$ be the columns of matrix Q. Then Q is an orthogonal matrix. QA is an upper triangular matrix, of the form

$$\frac{q_i^T}{||q_i||}A(:, j), i \leq j. \tag{29}$$

and 0 for i>j. We reduce the LS problem to solving a triangular system. We use back substitution to obtain the solution, as in [12]:

$$\hat{\xi}_{n-k}(\delta) = \frac{q_{n-k}^T}{q_{n-k}^T q_{n-k}}(b - \sum_{i=k}^{n-(k-1)} \hat{\xi}_i(\delta) A(:,i)), k = 0, ..., n-1. \tag{30}$$

The above is a special case of the solution of Theorem 4, with δ=0, and $A^T(:, n-k)q_{n-k}(0) = q_{n-k}^T q_{n-k}$. We may view Theorem 4's solution as a generalization of the QR solution to the LS problem.

Also, as

$$A^+ = \lim_{\delta \to 0}(A^T A + \delta I)^{-1} A^T = \lim_{\delta \to 0} A^T (AA^T + \delta I)^{-1}, \delta > 0, \tag{31}$$

A^+ always exists. So, for m $\geq$ n, $\hat{x}(\delta) = (A^T A + \delta I)^{-1} A^T y$. Then $\overset{lim}{\delta \to 0} \hat{x}(\delta) = A^+ y$.

We now produce a sequence of solutions $\hat{x}_n(\delta)$ to the general Tikhonov problem. The sequence converges to the solution x of $A^{m \times n}$ x = b, for m = n, as well as to the solution of $A^T A$x= $A^T b$, or x =$(A^T A)^{-1} A^T b$, for m > n. The matrix $A^{m \times n}$ is of rank n and possibly ill-conditioned. The sequence $\hat{x}_n(\delta)$ converges independently of our choice of δ > 0 or x_0. Numerical computations reveal the choice of δ influences the rate of convergence. We now derive our algorithm for the solution to the general regularization problem

$$\underset{x}{min}(||Ax-b||_2^2+\delta||x-x_0||_2^2, \delta>0, \tag{32}$$

where x_0 is the initial estimate of the solution. Next, we show more generally that the sequence

$$\hat{x}_n = \underset{x}{min}(||Ax-b||_2^2+\delta||x-x_{n-1}||_2^2,\ \delta>0 \tag{33}$$

converges to the solution above.

3.2.2 Generalized Tikhonov Regularization

We rewrite the LS problem of [12] as

$$f_n(b) = \underset{\xi_1,\xi_2,...,\xi_n}{min}(\sum_{i=1}^{m}(\sum_{j=1}^{n}\zeta_{ij}\xi_j-\gamma_i)^2+\delta\sum_{j=1}^{n}(\xi_j-\xi_j^0)^2), \tag{34}$$

where ξ_j^0 is the jthcomponent of x_0. For k=1,2,...,n,

$$f_k(b) = \underset{\xi_1,\xi_2,...,\xi_k}{min}(\sum_{i=1}^{m}(\sum_{j=1}^{k}\zeta_{ij}\xi_j-\gamma_i)^2+\delta\sum_{j=1}^{k}(\xi_j-\xi_j^0)^2). \tag{35}$$

By a similar argument of Section 3.2, it is not hard to see that the above is a quadratic form, $f_k(b)$= b^TV_kb + $2{w_k}^Tb$ + c_k. V_k is a symmetric matrix independent of the vector b, w_k a vector independent of b, and c_k a constant term, independent of b.

We insert the coefficient 2 as the middle term to simplify the following computation. We note similarities to the derivations of section 3.2, but caution that the derivation which follows is more complicated.

The recurrence equation of [12] is

$$f_k(b) = \min_{\xi_k} f_{k-1}(b - \xi_k A(:,k)) + \ \delta(\xi_k - \xi_k^0)^2)$$

$$= \min_{\xi_k}((b - \xi_k A(:,k))^T V_{k-1}(b - \xi_k A(:,k))$$

$$+2w_{k-1}^T(b\text{-}\xi_k A(:,k) + \ \delta(\xi_k - \xi_k^0)^2 + c_k). \tag{36}$$

Taking the derivative with respect to ξ_k, and setting it equal to zero, gives

$$\delta(\xi_k^* - \xi_k^0) - (b - \xi_k^* A(:,k))^T V_{k-1} A(:,k) - w_k^T A(:,k) = 0, \tag{37}$$

where ξ_k^* is the minimizing value. Solving for ξ_k^*, we get

$$\xi_k^* = \frac{b^T V_{k-1} A(:,k) + \ \delta\xi_k^0 + w_k^T \mathrm{A(:,k)}}{\delta + A(:,k)^T V_{k-1} A(:,k)}. \tag{38}$$

Now

$$f_k(b) = b^T V_k b + 2w_k^T b + c_k$$

$$= f_{k-1}(b - \xi_k^* A(:,k)) + \ \delta(\xi_k^* - \xi_k^0)^2)$$

$$= ((b - \xi_k^* A(:,k))^T V_{k-1}(b - \xi_k^* A(:,k))$$

$$+2w_{k-1}^T(b\text{-}\xi_k^* A(:,k) + \ \delta(\xi_k^* - \xi_k^0)^2 + c_k). \tag{39}$$

We expand the above expression, rearranging the terms so that we can eliminate some terms by (37). This simplifies things a bit. To obtain the recurrence relations for V_k and w_k^T, we collect the terms quadratic in b and linear in b. We square $(\xi_k^* - \xi_k^0)^2$, then multiply through by $(b - \xi_k^* A(:,k))^T V_{k-1})$ above, getting

$$(b - \xi_k^* A(:,k))^T V_{k-1} b - \xi_k^*(b - \xi_k^* A(:,k))^T V_{k-1} A(:,k) + 2w_{k-1}^T(b\text{-}\xi_k^* A(:,k))$$

$$+ \ \delta({\xi_k^*}^2 - 2\xi_k^*\xi_k^0 + {\xi_k^0}^2) + c_k$$

$$= (b - \xi_k^* A(:,k))^T V_{k-1} b - \xi_k^*(b - \xi_k^* A(:,k))^T V_{k-1} A(:,k) + 2w_{k-1}^T b$$

$$\text{-}2w_{k-1}^T \xi_k^* A(:,k)) + \delta\xi_k^*(\xi_k^* - \xi_k^0) - \delta\xi_k^*\xi_k^0 + \delta{\xi_k^0}^2 + c_k. \tag{40}$$

By (37), we eliminate the following expression, embedded in the expression above, so that

$$\delta\xi_k^*(\xi_k^* - \xi_k^0) - \xi_k^*(b - \xi_k^* A(:,k))^T V_{k-1} A(:,k) - \xi_k^* w_k^T A(:,k) = 0, \tag{41}$$

obtaining

$$(b - \xi_k^* A(:,k))^T V_{k-1} b + 2w_{k-1}^T b - \xi_k^* w_k^T A(:,k) \text{-}\ \delta\xi_k^*\xi_k^0\ \text{-}\ \delta\xi_k^{0^2} + c_k. \tag{42}$$

Next, we substitute the above expression of (38) in ξ_k^*, obtaining

$$(b - (\frac{b^T V_{k-1} A(:,k) +\ \delta\xi_k^0 + w_k^T \mathrm{A(:,k)}}{\delta + A(:,k)^T V_{k-1} A(:,k)}) A(:,k))^T V_{k-1} b + 2w_{k-1}^T b$$

$$-(\frac{b^T V_{k-1} A(:,k) +\ \delta\xi_k^0 + w_k^T \mathrm{A(:,k)}}{\delta + A(:,k)^T V_{k-1} A(:,k)}) w_k^T A(:,k)$$

$$-\delta(\frac{b^T V_{k-1} A(:,k) +\ \delta\xi_k^0 + w_k^T \mathrm{A(:,k)}}{\delta + A(:,k)^T V_{k-1} A(:,k)})\xi_k^0\ \text{-}\ \delta\xi_k^{0^2} + c_k. \tag{43}$$

We multiply through the expression, then collect the terms quadratic and linear in b. Next, we absorb the remaining terms into a constant, getting

$$b(V_{k-1} - \frac{(V_{k-1}A(:,k))(V_{k-1}A(:,k))^T}{\delta + A^T(:,k) V_{k-1} A(:,k)})b$$

$$+2w_{k-1}^T b - \frac{2\delta\xi_k^0 A(:,k))^T V_{k-1} b}{\delta + A^T(:,k) V_{k-1} A(:,k)}$$

$$-\frac{2w_k^T A(:,k) A(:,k))^T V_{k-1} b}{\delta + A^T(:,k) V_{k-1} A(:,k)} + Constant. \tag{44}$$

Recall that $f_k(b) = b^T V_k b + 2w_k^T b + c_k$. By equating the quadratic and linear terms in b, we obtain

$$b^T V_k b = b^T (V_{k-1} - \frac{(V_{k-1} A(:,k))(V_{k-1} A(:,k))^T}{\delta + A^T(:,k) V_{k-1} A(:,k)}) b, \tag{45}$$

and

$$2w_k^T b = 2w_{k-1}^T b - \frac{2\delta\xi_k^0 A(:,k))^T V_{k-1} b}{\delta + A^T(:,k) V_{k-1} A(:,k)} - \frac{2w_{k-1}^T A(:,k) A(:,k))^T V_{k-1} b}{\delta + A^T(:,k) V_{k-1} A(:,k)}. \tag{46}$$

We now have the following recurrence equations, for matrices V_k, and vectors w_k, respectively:

$$V_k = V_{k-1} - \frac{(V_{k-1} A(:,k))(V_{k-1} A(:,k))^T}{\delta + A^T(:,k) V_{k-1} A(:,k)}$$

$$= V_{k-1} - \frac{q_k(\delta) q_k(\delta)^T}{\delta + A^T(:,k) q_k(\delta)}, \tag{47}$$

and

$$w_k = w_{k-1} - \frac{\delta\xi_k^0 V_{k-1} A(:,k)) + w_{k-1}^T A(:,k) A(:,k))^T V_{k-1}}{\delta + A^T(:,k) V_{k-1} A(:,k)}$$

$$= w_{k-1} - \frac{\delta\xi_k^0 q_k(\delta) + V_{k-1} A(:,k) A(:,k)^T w_{k-1}}{\delta + A^T(:,k) q_k(\delta)}$$

$$= w_{k-1} - \frac{(\delta\xi_k^0 + A(:,k)^T w_{k-1})}{\delta + A^T(:,k)q_k(\delta)} q_k(\delta). \tag{48}$$

In arriving at the final steps, we used the fact that the matrix V_k is symmetric. By definition, $q_k(\delta) = V_{k-1}(\delta)A(:,k)$. Thus, in the denominator, we have $A^T(:,k)V_{k-1}A(:,k) = A^T(:,k)q_k(\delta)$, for k=1,2,...,n,... Note that $\frac{(\delta\xi_k^0+A(:,k)^T w_{k-1})}{\delta+A^T(:,k)q_k(\delta)}$ is a scalar times the vector $q_k(\delta)$.

We now derive the algorithm for the generalized Tikhonov regularization.

Theorem 5. Let $A^{m\times n}$=$(\zeta_{ij})^{m\times n}$, b=(γ_i)ε R^m , x=(ξ_j) ε R^n,and $q_k(\delta)$, as in Theorem 4. The vector that minimizes $\overset{min}{x}(||Ax-b||_2^2 + \delta||x-x_0||_2^2, \delta > 0$ is

$$\hat{x}(\delta) = [\hat{\xi}_1(\delta), \hat{\xi}_2(\delta), ..., \hat{\xi}_n(\delta)]^T, \tag{49}$$

with the scalars

$$\begin{aligned}
\hat{\xi}_n(\delta) &= \frac{\delta\xi_n^0 + A(:,n)^T w_{n-1} + q_n^T(\delta)b}{\delta + A^T(:,n)q_n(\delta)}, \\
\hat{\xi}_{n-1}(\delta) &= \frac{\delta\xi_{n-1}^0 + A(:,n-1)^T w_{n-2} + q_{n-1}^T(\delta)(b - \hat{\xi}_n(\delta)A(:,n))}{\delta + A^T(:,n-1)q_{n-1}(\delta)},
\end{aligned}$$

$$\ldots$$

$$\hat{\xi}_{n-k}(\delta) = \frac{\delta\xi^0_{n-k} + A(:,n-k)^T w_{n-(k+1)} + q^T_{n-k}(\delta)b_k}{\delta + A^T(:,n-k)q_{n-k}(\delta)}, k = 0, ..., n-1. \tag{50}$$

We set the vectors b$_0$=b, and b$_k$=$b - \sum_{i=k}^{n-(k-1)} \hat{\xi}_i(\delta)A(:,i)$. Each b$_k$ is the adjusted right hand side of an m-by-(n-k) system of equations. For k = 0, we have the original m-by-n system with the right hand side b$_0$=b.

Proof. This proof is similar to that of Theorem 4. Substituting for f$_{n-1}(b-\xi_n A(:,n))$ into the recurrence equation, that is,

$$f_n(b) = \min_{\xi_n} f_{n-1}(b - \xi_n A(:,n)) + \delta(\xi_n - \xi^0_n)^2), \tag{51}$$

gives

$$(b - \xi_n A(:,n))^T V_{n-1}(b - \xi_n A(:,n)) +$$

$$2w^T_{n-1}(b\text{-}\xi_k A(:,n) + \delta(\xi_n - \xi^0_n)^2 + c_n. \tag{52}$$

Since this is a minimization of a quadratic function with respect to ξ_n, we differentiate the above expression with respect to ξ_n, and set the derivative equal to zero, getting

$$\hat{\xi}_n(\delta) = \frac{\delta\xi^0_n + A(:,n)^T w_{n-1} + q^T_n(\delta)b}{\delta + A^T(:,n)q_n(\delta)}. \tag{53}$$

Now that we know $\hat{\xi}_n(\delta)$, we may minimize with respect to ξ_{n-1}, getting $\hat{\xi}_{n-1}(\delta)$. Aside from the additional term $\delta\xi^0_{n-k} + A(:,n-k)^T w_{n-(k+1)}$, in the numerator, the steps taken to obtain $\hat{\xi}_{n-k}(\delta)$ k=0,...,n-1 are the same as of Theorem 4. We reiterate that the computation of $q_k(\delta)$, for k=1,...,n, is the most expensive procedure in the algorithm, requiring at least $2\mathrm{m}^2 n$ flops. We ignore the remaining computations of order $m \times n$. Except for different notation, this is exactly the solution given in chapter 5 of [12].

Theorem 6. Assume matrix A is a matrix of full rank. By the algorithm of Theorem 5, the successive solutions to converge to the solution of $A^T A$x= $A^T b$. That is, they converge to the vector $\hat{x}$ =$(A^T A)^{-1} A^T b$= A^+b, for m > n. For m = n, they converge to the vector $A^{n\times n}x = b^{n\times 1}$, or $\hat{x} = (A^{n\times n})^{-1} b^{n\times 1}$.

This has been proved for for square matrices, where n = m, as in[12]. We extend the proof for m > n.

Proof. The normal equations are $(A^T A + \delta I)x_i = A^T b + \delta x_{i-1}$. The solution is

$$\hat{\mathrm{x}}_i = (A^T A + \delta I)^{-1} A^T b + \delta (A^T A + \delta I)^{-1} \hat{\mathrm{x}}_{i-1}, \tag{54}$$

where the vector $\hat{\mathrm{x}}_0$ is an initial guess. We show that the eigenvalues of the square matrix $\delta(A^T A + \delta I)^{-1}$ are all less than 1. Thus, the sequence $\hat{\mathrm{x}}_i$ converges at a geometric rate. Substituting the SVD of A=U$\sum$V^T, in $(A^T A + \delta I)^{-1}$, where $\mathrm{U}^{m\times m}$ and $\mathrm{V}^{n\times n}$ are orthonormal matrices, and where $\sum$=diag(σ_1,...,σ_n), a matrix of singular values of A, gives

$$((U\sum V^T)^T(U\sum V^T) + \delta I_n)^{-1} = (V\sum U^T U\sum V^T + \delta I_n)^{-1}$$

$$= (V{\sum}^2 V^T + \delta I_n)^{-1}$$

$$= (V({\sum}^2 + \delta I_n)V^T)^{-1} = V({\sum}^2 + \delta I_n)^{-1}V^T. \tag{55}$$

Hence the eigenvalues of the square matrix $\delta(A^T A + \delta I)^{-1}$are $\frac{\delta}{\sigma_1^2+\delta}$, . . . , $\frac{\delta}{\sigma_n^2+\delta}$. Since $\frac{\delta}{\sigma_k^2+\delta} < 1$, for k=1, . . , n, the matrix $\delta(A^T A + \delta I)^{-1}$ contracts. Thus, the sequence $\hat{x}_i$ converges. Let $\hat{x}$ be the limiting vector. Then $(A^T A + \delta I)\hat{x} = A^T b + \delta\hat{x}I$, or $A^T A\hat{x} = A^T b$, and the vector$\hat{x} = (A^T A)^{-1} A^T b$, or $\hat{x} = A^+ b$. If A is a square matrix of full rank, $A^T A\hat{x} = A^T b$ reduces to $A\hat{x} = b$. We have proved that the algorithm of Theorem 5 computes successive approximations that converge to the pseudoinverse solution of an overdetermined linear system. They also converge to the solution of a square linear system, independent of the parameter δ and initial guess of vector $\hat{x}_0$. Nevertheless, the choice of δ and $\hat{x}_0$ are important, as they determine the computational rate of convergence. The number of flops taken for each successive approximation is of order $m \times n$. Even with successive approximations, the algorithm is of order $2\mathrm{m}^2 n$. We now consider some numerical experiments, as well as a stopping rule for the successive approximations.

In computing the transition matrix $\widehat{M}$, the halting rule for computing successive approximations is "stop as soon as the relative residual error is greater than the prior residual error," that is, stop if $\frac{||A\hat{x}_i - b||_2}{||b||_2} > \frac{||A\hat{x}_{i-1} - b||_2}{||b||_2}$.

We have chosen the relative residual error, instead of the residual error $||A\hat{x}_i - b||_2$. We may convert the residual error to any value by multiplying $A\hat{x}_i$ and b with a constant, and then canceling out the relative residual error. The reason for choosing relative residual error is, that as we continue computing successive approximations, the round-off error eventually becomes so large that continuing the algorithm cannot improve the accuracy of the solution. In other words, our stopping rule states that the solution accuracy is based more on the number of significant figures in the computation than on the condition.

3.3 Computation Example

We offer a more intuitive demonstration of our algorithm of Theorem 4. Let the matrix A be square and non-singular, so A^+=A^{-1}. We set δ=0.

Now, solve Ax=b, where $A = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix}$, $x = \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix}$, $and\, y = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$.

Let the matrix A=$(\alpha_{kl})^{3x3}$, the vector x=$(\zeta_l)^{3x1}$, and the vector b=(β_k). We may write the algorithm symbolically as well as computationally. We orthogonalize the columns of A, so

$$q_1 = A(:,1) = \begin{bmatrix} \alpha_{11} \\ \alpha_{21} \\ \alpha_{31} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}, \tag{56}$$

$$q_2 = (I - \frac{q_1 q_1^T}{q_1^T q_1})A(:,2) = (I - \frac{q_1 q_1^T}{q_1^T q_1}) \begin{bmatrix} \alpha_{12} \\ \alpha_{22} \\ \alpha_{32} \end{bmatrix} \tag{57}$$

$$= \left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{6} \begin{bmatrix} 1 & 1 & 2 \\ 1 & 1 & 2 \\ 2 & 2 & 4 \end{bmatrix} \right) \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{5}{6} & -\frac{1}{6} & -\frac{2}{6} \\ -\frac{1}{6} & \frac{5}{6} & -\frac{2}{6} \\ -\frac{2}{6} & -\frac{2}{6} & \frac{2}{6} \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{7}{6} \\ \frac{1}{6} \\ -\frac{4}{6} \end{bmatrix},$$

$$q_3 = (I - \frac{q_1 q_1^T}{q_1^T q_1} - \frac{q_2 q_2^T}{q_2^T q_2})A(:,3) = (I - \frac{q_1 q_1^T}{q_1^T q_1} - \frac{q_2 q_2^T}{q_2^T q_2}) \begin{bmatrix} \alpha_{13} \\ \alpha_{13} \\ \alpha_{13} \end{bmatrix} \tag{58}$$

$$= \left(\begin{bmatrix} -\frac{5}{6} & -\frac{1}{6} & -\frac{2}{6} \\ -\frac{1}{6} & \frac{5}{6} & -\frac{2}{6} \\ -\frac{2}{6} & -\frac{2}{6} & \frac{2}{6} \end{bmatrix} - \frac{q_2 q_2^T}{q_2^T q_2} \right) \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= \left(\begin{bmatrix} -\frac{5}{6} & -\frac{1}{6} & -\frac{2}{6} \\ -\frac{1}{6} & \frac{5}{6} & -\frac{2}{6} \\ -\frac{2}{6} & -\frac{2}{6} & \frac{2}{6} \end{bmatrix} - \begin{bmatrix} \frac{49}{66} & \frac{7}{66} & -\frac{28}{66} \\ \frac{7}{66} & \frac{1}{66} & -\frac{4}{66} \\ -\frac{28}{66} & -\frac{4}{66} & \frac{16}{66} \end{bmatrix} \right) \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{11} & -\frac{3}{11} & \frac{1}{11} \\ -\frac{3}{11} & \frac{9}{11} & -\frac{3}{11} \\ \frac{1}{11} & -\frac{3}{11} & -\frac{3}{11} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{11} \\ +\frac{3}{11} \\ -\frac{1}{11} \end{bmatrix}.$$

Note the column vectors $q_1 = \begin{bmatrix} \alpha_{11} \\ \alpha_{21} \\ \alpha_{31} \end{bmatrix}$, $q_2 \perp \mathcal{L}(q_1)$ and $q_3 \perp \mathcal{L}(q_1,q_2)$. Thus, Ax = y may be written as $\zeta_1 q_1 + \zeta_2(q_2 + c_1 q_1) + \zeta_3(q_3 + c_1 q_1 + c_2 q_2) =$ y. We multiply both sides of $\zeta_1 q_1 + \zeta_2(q_2 + c_1 q_1) + \zeta_3(q_3 + c_1 q_1 + c_2 q_2) = y$ by $\frac{q_3^T}{q_3^T q_3}$, so that the left side of the equation becomes ζ_3. The right side is

$$\hat{\xi}_3 = \frac{q_3^T}{q_3^T q_3} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}, \tag{59}$$

where c_1 and c_2 are constants.

This is because

$$\frac{q_3^T}{q_3^T q_3} q_1 = \frac{q_3^T}{q_3^T q_3} \begin{bmatrix} \alpha_{11} \\ \alpha_{21} \\ \alpha_{31} \end{bmatrix} = 0, \tag{60}$$

and

$$\frac{q_3^T}{q_3^T q_3}(q_2 + c_1 q_1) = \frac{q_3^T}{q_3^T q_3}(\begin{bmatrix} \alpha_{12} \\ \alpha_{22} \\ \alpha_{32} \end{bmatrix} + c_1 \begin{bmatrix} \alpha_{11} \\ \alpha_{21} \\ \alpha_{31} \end{bmatrix}) = 0, \tag{61}$$

and

$$\frac{q_3^T}{q_3^T q_3}(q_3 + c_1 q_1 + c_2 q_2) = \frac{q_3^T}{q_3^T q_3}(\begin{bmatrix} \alpha_{13} \\ \alpha_{13} \\ \alpha_{13} \end{bmatrix} + c_2 \begin{bmatrix} \alpha_{12} \\ \alpha_{22} \\ \alpha_{32} \end{bmatrix} + c_1 \begin{bmatrix} \alpha_{11} \\ \alpha_{21} \\ \alpha_{31} \end{bmatrix}) = 1. \tag{62}$$

The scalar $\hat{\xi}_3$ is the projected length of vector y onto the subspace $\mathcal{L}(q_3)$.

Now that we know $\hat{\xi}_3$, we may reduce the problem's dimension, so

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \\ \alpha_{31} & \alpha_{32} \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \end{bmatrix} = \left(\begin{bmatrix} \Upsilon_1 \\ \Upsilon_2 \\ \Upsilon_3 \end{bmatrix} - \hat{\xi}_3 \begin{bmatrix} \alpha_{13} \\ \alpha_{13} \\ \alpha_{13} \end{bmatrix} \right). \tag{63}$$

We multiply both sides of this two dimensional equation by $\frac{q_2^T}{q_2^T q_2}$, obtaining

$$\hat{\xi}_2 = \frac{q_2^T}{q_2^T q_2} \left(\begin{bmatrix} \Upsilon_1 \\ \Upsilon_2 \\ \Upsilon_3 \end{bmatrix} - \hat{\xi}_3 \begin{bmatrix} \alpha_{13} \\ \alpha_{13} \\ \alpha_{13} \end{bmatrix} \right). \tag{64}$$

For the two dimensional case, $\hat{\xi}_2$ is the projected length of

$$\left(\begin{bmatrix} \Upsilon_1 \\ \Upsilon_2 \\ \Upsilon_3 \end{bmatrix} - \zeta_3 \begin{bmatrix} \alpha_{13} \\ \alpha_{23} \\ \alpha_{33} \end{bmatrix} \right), \tag{65}$$

onto the subspace $\mathcal{L}(q_2)$. Finally, we reduce the problem to one dimension, so

$$\begin{bmatrix} \alpha_{11} \\ \alpha_{21} \\ \alpha_{31} \end{bmatrix} \zeta_1 = \left(\begin{bmatrix} \Upsilon_1 \\ \Upsilon_2 \\ \Upsilon_3 \end{bmatrix} - \hat{\xi}_2 \begin{bmatrix} \alpha_{12} \\ \alpha_{22} \\ \alpha_{32} \end{bmatrix} - \hat{\xi}_3 \begin{bmatrix} \alpha_{13} \\ \alpha_{23} \\ \alpha_{33} \end{bmatrix} \right). \tag{66}$$

Multiplying both sides of the above by $\frac{q_1^T}{q_1^T q_1}$ gives

$$\hat{\xi}_1 = \frac{q_1^T}{q_1^T q_1} \left(\begin{bmatrix} \Upsilon_1 \\ \Upsilon_2 \\ \Upsilon_3 \end{bmatrix} - \hat{\xi}_2 \begin{bmatrix} \alpha_{12} \\ \alpha_{22} \\ \alpha_{32} \end{bmatrix} - \hat{\xi}_3 \begin{bmatrix} \alpha_{13} \\ \alpha_{23} \\ \alpha_{33} \end{bmatrix} \right). \tag{67}$$

For one dimension, $\hat{\xi}_1$is the projected length of the new right hand side of

$$\left(\begin{bmatrix} \Upsilon_1 \\ \Upsilon_2 \\ \Upsilon_3 \end{bmatrix} - \hat{\xi}_2 \begin{bmatrix} \alpha_{12} \\ \alpha_{22} \\ \alpha_{32} \end{bmatrix} - \hat{\xi}_3 \begin{bmatrix} \alpha_{13} \\ \alpha_{23} \\ \alpha_{33} \end{bmatrix}\right), \tag{68}$$

onto the subspace $\mathcal{L}(q_1)$.

Next, we compute the lengths of the projected right hand sides on the orthogonalized vectors,

$$\hat{\xi}_3 = \frac{q_3^T}{q_3^T q_3} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} \tag{69}$$

$$= \begin{bmatrix} -1 & 3 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = 4,$$

$$\hat{\xi}_2 = \frac{q_2^T}{q_2^T q_2} (\begin{bmatrix} \Upsilon_1 \\ \Upsilon_2 \\ \Upsilon_3 \end{bmatrix} - \hat{\xi}_3 \begin{bmatrix} \alpha_{13} \\ \alpha_{23} \\ \alpha_{33} \end{bmatrix}) \tag{70}$$

$$= \begin{bmatrix} \frac{7}{11} & \frac{1}{11} & -\frac{4}{11} \end{bmatrix} \left(\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} - 4 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} \frac{7}{11} & \frac{1}{11} & -\frac{4}{11} \end{bmatrix} \begin{bmatrix} -3 \\ -2 \\ -3 \end{bmatrix} = -1,$$

$$\hat{\xi}_1 = \frac{q_1^T}{q_1^T q_1} \left(\begin{bmatrix} \Upsilon_1 \\ \Upsilon_2 \\ \Upsilon_3 \end{bmatrix} - \hat{\xi}_2 \begin{bmatrix} \alpha_{12} \\ \alpha_{22} \\ \alpha_{32} \end{bmatrix} - \hat{\xi}_3 \begin{bmatrix} \alpha_{13} \\ \alpha_{23} \\ \alpha_{33} \end{bmatrix} \right) \tag{71}$$

$$= \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{2}{6} \end{bmatrix} \left(\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} - (-1) \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} - 4 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{2}{6} \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -2 \end{bmatrix} = -1.$$

The solution is

$$\begin{bmatrix} \hat{\xi}_1 \\ \hat{\xi}_2 \\ \hat{\xi}_3 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ +4 \end{bmatrix}. \tag{72}$$

We have now solved Ax=b. Namely, $\begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ +4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}.$

The QR factorization of A solves a LS problem, by orthonormalizing the matrix A to obtain the matrix Q. In the equation QA = R, QA is an upper triangular matrix.

It is not hard to show, as[11] does, that the LS problem reduces to solving x in the equation Rx = Qb. We make use of back substitution, also called back projection.

Our example provides insights as to how the algorithm of Theorem 4 works in general, for $\delta > 0$. In the general form, we pseudo-orthogonalize the columns of matrix A. We then compute the lengths of the projections of the right hand sides onto the pseudo-orthogonalized columns of A.

For a non-singular, square matrix, our algorithm is computationally more stable than the common reduced row echelon form method found in many textbooks. Our algorithm is stable even when a system is ill-conditioned, overdetermined, or rank deficient.

3.4 A Hilbert Matrix Example

Typically, Hilbert matrices are used to test algorithms that model ill-conditioned systems. We test our algorithm using a 12-by-12 Hilbert matrix. To generate a 12-by-12 Hilbert matrix in MatLab, the command is: H = hilb(12), where H is the Hilbert matrix. To compute its condition number, the command is cond(H), which gives $1.7\text{x}10^{16}$. To test our algorithm, we set the vector x equal to a column of 1's, then multiply x by the matrix H, obtaining the right hand side, the vector b. Hence, we know the exact solution to the equation Hx = b. The Euclidean distance of the MatLab solution to the actual solution is 0.010663758566179, while the distance of our solution to the actual solution, with $\delta = 0.000001$, is 0.014060974264627. However, when perturbing the right hand side, the vector b, by adding 0.00001 or -0.00001 to the components of b, the distance of the MatLab solution to the original solution of 1s is $3.138807922868325\text{x}10^8$. On the other hand, the distance of our solution is 0.033601846692835. The MatLab pseudoinverse solution is off by more than ten million. Using the Hilbert matrix test, our example demonstrates the instability of the solution, whereas the pseudoinverse solution of Kohonen does not.

3.5 Golub's Example

The following example appears in[11]. If the rank(A) $<$ n, the solution $\hat{x}= A^{+}$b is not a continuous function of the data[11]. Small changes in the matrix A and/or vector b may accompany arbitrary changes in the vector $\hat{x}= A^{+}$b .

The proof of the last statement appears in [11]. To illustrate the instability of the equation $\hat{x}= A^{+}$b, where A is a singular matrix, we cite the following example:

Let A $= \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$, ΔA$= \begin{bmatrix} 0 & 0 \\ 0 & \varepsilon \\ 0 & 0 \end{bmatrix}$, then $A^{+}= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$,

and $(A+\Delta A)^{+}= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\varepsilon} & 0 \end{bmatrix}$.

But $||A^{+}$ - $(A+\Delta A)^{+}|| = \frac{1}{\varepsilon}$. To verify that A^{+} and $(A+\Delta A)^{+}$ are the matrices that are the pseudoinverses of A and (A+ΔA), respectively, we may use one of the alternate definitions of pseudoinverse in Appendix A. It is easy to check that the solution satisfies the four Moore-Penrose conditions.

We compare Penrose's solution to our algorithm's. Consider the above matrices, and the least-square problems

$$\underset{x}{min} ||Ax - y|| \tag{73}$$

$$\underset{x}{min} ||(A + \Delta A)x - y||, \tag{74}$$

where the matrices A $= \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$, ΔA$= \begin{bmatrix} 1 & 0 \\ 0 & 10^{-6} \\ 0 & 0 \end{bmatrix}$, and the vector y$= \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$. Solve

$\underset{\zeta_1,\zeta_2}{min} || \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} \zeta_1 \\ \zeta_2 \end{pmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} ||.$

Since the pseudoinverse $A^+$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, the Penrose solution, a vector, is $A^+y =$

$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$

since $(A + \Delta A)^{+} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 10^6 & 0 \end{bmatrix}$. The Penrose solution to the perturbed problem is $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 10^6 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 10^6 \end{bmatrix}$.

On the other hand, using the algorithm of theorem 4, with $\delta = 0.001$, gives the solution $\begin{bmatrix} 0.9990099 \\ 0.0010000 \end{bmatrix}$, a vast improvement over $\begin{bmatrix} 1 \\ 10^6 \end{bmatrix}$.

Chapter 4

4. RESULTS

4.1 Data Encoding

We derive the Radon transform from Lambert and Beer's X-ray decay law, i.e.

$$I = I_0 e^{-_{L_i} \int u(x,y)ds}, \tag{75}$$

or

$$_{L_i} \int u(x,y)ds = -ln(\frac{I}{I_0}), \tag{76}$$

where u(x,y) is the image, L_i is the i^{th} X-ray, I_0 the initial intensity of the i^{th} ray, I the final intensity of the i^{th} ray, and $-ln(\frac{I}{I_0})$ the observed i^{th} datum on a detector. The Radon transform is the path integral $_{L_i} \int u(x,y)ds$, a line integral taken over the image area. Detectors record values $\frac{I}{I_0}$, then $-ln(\frac{I}{I_0})$ is computed. Finally, the inverse radon transform recovers the image.

Typically, in medical imaging, feature vectors are extracted from segmented regions. They are used to train an associative memory mapping patterns in the image into the feature vectors. Various pattern algorithms then analyze the feature vectors. We will work instead with the computed Radon transforms. In our approach, the pattern space is the n-dimensional Euclidean vector space of Radon transforms of an image. A vector is formed by concatenating the columns of the Radon transform matrix. Training consists of taking images, generating the Radon transform vector of each image, removing an object of interest from the image, recomputing the Radon transform of the corresponding modified image, and finally, taking the difference of the corresponding Radon vectors. The difference vector is the Radon transform of the object of interest.

Figure 3, on the next page, from left to right, shows the original Shepp-Logan phantom, the same phantom with an object removed (the white dot), and the object itself. These phantoms were generated by taking the inverse of the difference of two Radon transforms. The barely visible rays emanating from the object are computational artifacts. These computational artifacts can be diminished by increasing the number of rays in the Radon transform, but at the cost of increasing the number of computations.

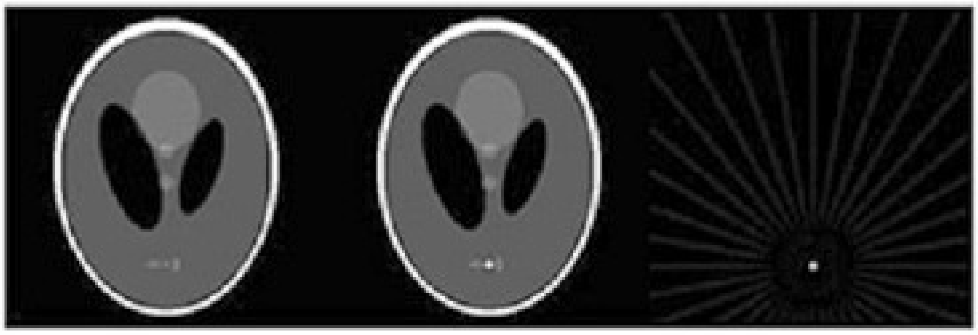


Fig. 3: An example of an image feature.

We now apply OLAM to two different goals. The first is to train the modified OLAM to distinguish normal anatomical structures from abnormal ones. The second is to train our OLAM to recognize anatomical structures in an image. To train our OLAM to recognize abnormal structures, we project the Radon transform of the structure in question onto the subspace spanned by the Radon transforms of normal structures. Next, we use a metric to measure the probability that the projected vector is normal. The second goal is to train our modified OLAM to display anatomical structures in a CT image. For this task, we train a heteroassociative memory by taking input signals as Radon transforms of images, and output signals as Radon transforms of anatomical structures in the images. When given a new image, we compute its Radon vector x, then invert the Radon transform obtained from the vector $\widehat{M}x$ to produce the image. Our experiments produce the desired results.

4.2 Simulation Experiments

The Shepp-Logan phantom is an artificial representation of a human head cross-section. It consists of ten ellipses, having six parameter values for each ellipse that may be altered. The parameters of each ellipse are:

1. The additive intensity value of the ellipse.

2. The length of the horizontal semi-axis of the ellipse.

3. The length of the vertical semi-axis of the ellipse.

4. The x-coordinate of the center of the ellipse. increasing computation

5. The y-coordinate of the center of the ellipse.

6. The angle (in degrees) between the horizontal semi-axis of the ellipse and the x-axis of the image.

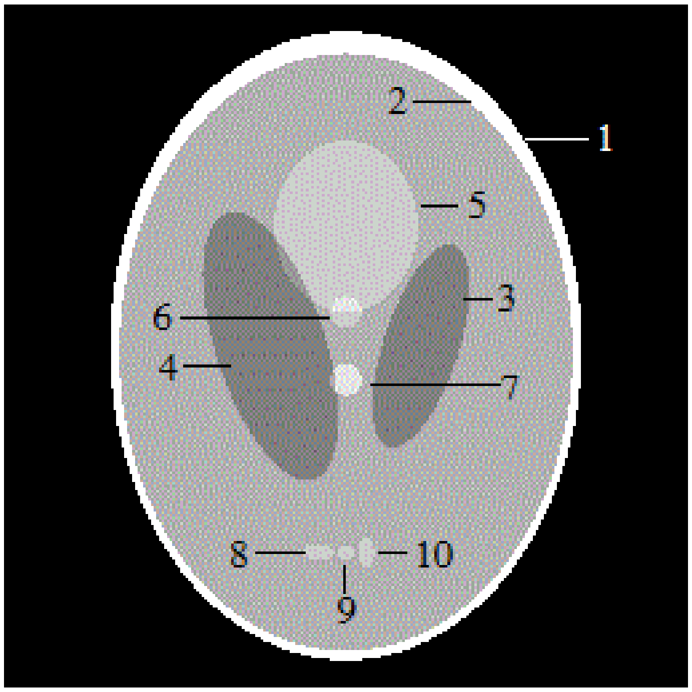


Fig. 4: The 10 ellipses of the Shepp-Logan phantom.

We test how well our algorithms detect abnormalities in images. Using 18 angles and 128 by 128 pixels produces a Radon vector with 3330 entries. Each entry is a neuron in the OLAM model. Of course, increasing the number of angles and pixes would produce better pictures, but would probably overwhelm any laptop computer, as it did in this author's case.

We conduct simulation experiments on Shepp-Logan phantoms. Data are generated by changing the six parameter values of the ten ellipses making up a phantom. We use a random number generator to slightly change the parameter values of each ellipse, generating different images of a head section. We include the MatLab command that generates the discrete Random transform of each image within the loop generating the images. We then concatenate the rows of the transform into a column vector to produce the Radon vector of each image. A collection of the Radon vectors makes up our training data. In order to generate the Radon vectors of an anatomical structure, for instance, one of the interior ellipses in the phantom, we set the gray-scale parameter of an ellipse-in-question to a value representing no density, the density of air. We then take the Radon transform of this modified phantom and construct its Radon vector. Finally, we take the difference of the two Radon vectors, the original image and the image with an ellipse whose gray scale is set to the density of air, producing the Radon transform vector of the ellipse or anatomical structure in question. Taking the difference of the two Radon vectors cancels out all the values except those of the structure in question. This roundabout method of generating the Radon vector of the anatomical structure allows us to reconstruct the anatomical structure of a new image in the correct location in the new image. All of these operations are performed within the same loop, to produce Radon vectors of each image and the anatomical structures. From the generated data, we compute the transition matrix, or projection operator. Each anatomical structure requires its own transition matrix in order to identify and display it. Once we have the projection operators for each structure, we generate the Radon vector of a new image and apply the computed projection matrix on that Radon vector to obtain the Radon vector of its anatomical structure. We compute the Radon inverse of the computed Radon vector to identify and display an anatomical structure. With our experiments, the structure produced closely resembled the structure in the image, as well as the correct location relative to other parts in the image.

We also conduct simulation experiments to measure the abnormality of a given anatomical structure. The training data consists only of Radon vectors of the normal anatomical structures. The Radon vectors of these structures are produced as stated in the previous paragraph. We then generate a new image whose structure in question was modified considerably more than those structures in the training set. Next, we apply our developed probability measure on the Radon vector of the new structure to obtain the probability of the structure's normality. When we modify the structure in question only slightly, the probability that the structure is normal is greater than 0.9. Subsequently, when we modify the structure considerably more, moving it to a new location, for example, the computed probability measure that the structure is normal is less than 0.1. These results show that our algorithm produces the desired results. We discuss these experiments in more detail in Section 4.3. The MatLab code appears in Appendix B.

4.3 Abnormality Measure

When applying our computed projection operator on the Radon transform of an anatomical structure, it is possible to distinguish a normal anatomical structure from an abnormal one. If the structure is normal, it may be written as a linear combination of a basis of normal-structure Radon vectors forming a subspace. In "real life" scenarios, clinicians select hundreds of images and then decide if the anatomical structure in each image is normal. A training set of vectors is generated by computing the Radon transforms of the structure in each image as the input and output training signals. In autoassociative recall, the estimated projection operator $\widehat{M}$ encodes only the known signal patterns $x_k \varepsilon\ R^n$, k ε {1,2,...,m}, where Y = X. So $\widehat{M}$ reduces to $\widehat{M} = XX^+$.

Now suppose a vector x is a corrupted signal of one of the input patterns x_k, so that $x = x_k + x_{noise}$.

The signal x_k is estimated by projecting x onto $\mathcal{L}(x_1,x_2,...,x_n)$.[1] shows that if x_{noise} distribution is radially symmetric, such as in the multivariate Gaussian probability distribution, the variance of the random variable estimator norm $||\hat{x}$-$x_k||$ is

$$Var[||\hat{x} - x_k||] = \frac{n}{m}||\hat{x} - x_k||^2 = \frac{m}{n}||x_{noise}||^2. \tag{77}$$

The above expression shows that the projection operator $\widehat{M} = XX^{+}$ suppresses noise if n < m, that is, if the number of independent input patterns is less than the dimension of the input pattern. Noise is amplified if n > m, as shown above. We may interpret this as the fact that more neurons allow for a more reliable storage and recollection of more patterns. How the dimensionality m of input patterns limits memory capacity in associative networks appears in[1]. Now, suppose X=(x_1 ,..., x_n) are the input signal vectors spanning the subspace $\mathcal{L}$(x_1 ,..., x_n) . Let the matrix P= XX^+. Then P is a projection operator projecting an arbitrary vector onto $\mathcal{L}(x_1, ..., x_n)$. To demonstrate this, we show P's idempotence, that is,

$$P^2 = XX^+XX^+ = (XX^+X)X^+ = XX^+ = P. \tag{78}$$

We used a property of pseudoinverses, that the matrix X=XX^+X . Similarly, if we expand the matrix I - P, we get

$$(I - XX^+)^2 = I^2 - 2XX^+ + XX^+XX^+$$

$$= I - 2XX^+ + XX^+ = I - XX^+ = I - P, \tag{79}$$

and see that I - P is also an idempotent matrix. The matrix P projects the vector space R^n onto $\mathcal{L}$(x_1 ,..., x_n), while the matrix I - P projects R^n onto the orthogonal subspace $\mathcal{L}^{\perp}$.[9] The following metric, proposed by Kohonen in [9], measures how near a vector x is to subspace $\mathcal{L}(\mathrm{x}_1 ,..., \mathrm{x}_n)$:

$$d(x, \mathcal{L}) = \frac{||\hat{x}||_2}{||x||_2}, \tag{80}$$

where $\hat{x}$ = Px, and P=XX^{+} is a projection matrix.

The vector $\tilde{x} = (I - P)x$ is the projection of x onto the orthogonal subspace $\mathcal{L}^{\perp}$. The vector x $= \hat{x}+\tilde{x}$. $\tilde{x}$ is the novelty not explained by $\mathcal{L}$. However, instead of using the metric $d(x, \mathcal{L}) = \frac{||\hat{x}||_2}{||x||_2}$, as in [9], we use $d(x, \mathcal{L}) = \frac{||\hat{x}||_2^2}{||x||_2^2}$. We square the norm so that $d(x, \mathcal{L})$ $+d(x, \mathcal{L}^{\perp}) = 1$. Thus, we have a measure capturing the percentage of a vector x explained by the subspace $\mathcal{L}$, and the percentage of x explained by the subspace $\mathcal{L}^{\perp}$. Another advantage to using the norm squared is, if x is a random vector, we may more easily obtain the probability distribution of $d(x, \mathcal{L})$. This permits hypothesis testing and the computing of confidence intervals for $d(x, \mathcal{L})$.

Clinicians identify "normal" anatomical structures from a set of calibrated and normalized medical images. We test our algorithms on Shepp-Logan phantoms deemed normal. To simulate a set of images with normal anatomical structures, we slightly, randomly alter the ellipse parameters in each of the phantoms. There are 10 ellipses and 6 parameters for each ellipse, allowing up to 60 possible changes. The images produced make up the training set. The MatLab code in Appendix B generates the phantoms with slight, random parameter changes to ellipse number 9, the ellipse near the bottom in the center, as well as to the same phantom with that ellipse removed. When we take the difference of those Radon vectors, we get the Radon transform vector of only ellipse 9. We are left with Radon transforms of ellipse number nine that consist of our input signals. Suppose we take an image that is deemed abnormal. Our metrics $d(x,\mathcal{L}) = \frac{||\hat{x}||_2^2}{||x||_2^2}$ and $d(x,\mathcal{L}^{\perp}) = \frac{||\tilde{x}||_2^2}{||x||_2^2}$ tell us, as a percentage, by how much a new ellipse is normal and abnormal when compared to the set of normal ellipses. So, when given an image, we obtain the Radon vector x of the anatomical structure in question and project it onto $\mathcal{L}$ to obtain $\hat{x}$. We apply the metric

$$d(x,\mathcal{L}) = \frac{||\hat{x}||_2^2}{||x||_2^2} \tag{81}$$

that gives the probability that vector x is normal. Recall that the vector $\tilde{x} = (I - P)x$ measures the novelty not explained by the subspace $\mathcal{L}$ of Radon vectors of the “normal” object.

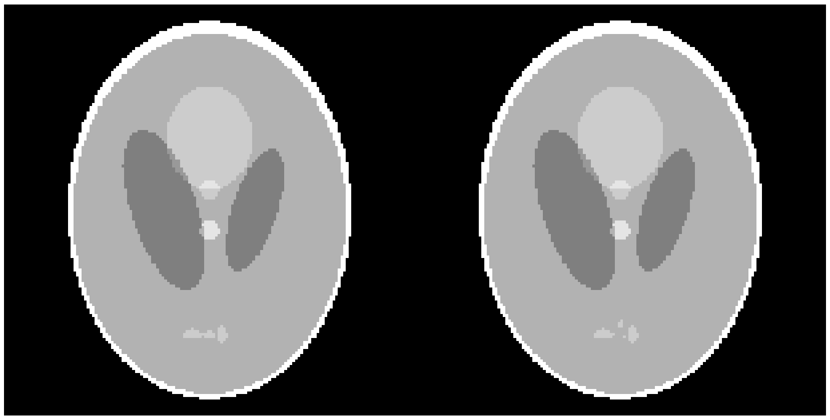
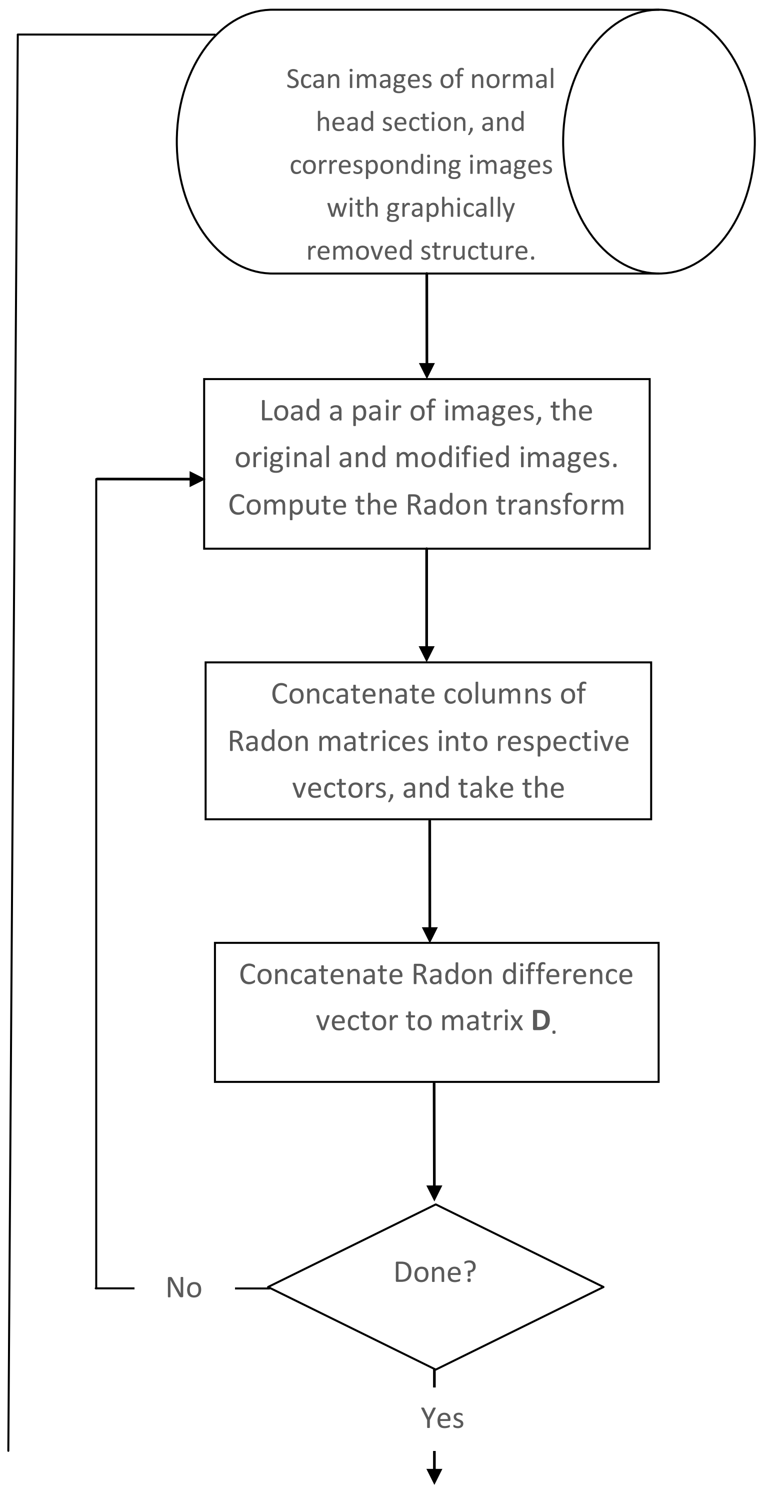


Fig. 5: A normal image on the left and a distorted image on the right.

In the above figure, $d(x,\mathcal{L}) = \frac{||\hat{x}||_2^2}{||x||_2^2} = 0.0003$ and $d(x,\mathcal{L}^{\perp}) = \frac{||\tilde{x}||_2^2}{||x||_2^2} = 0.9997$ of the image on the right. The subspace $\mathcal{L}$ is spanned by the normal Radon vectors of images such as the image on the left. In the image on the right, ellipse 9 has been modified more than the ellipses in the training set. We may interpret $d(x,\mathcal{L})$=0.0003 as the probability that the new image on the right is normal and $d(x,\mathcal{L}^{\perp}) = 0.9997$ that it is abnormal.

The MatLab code in Appendix B generates phantoms with the images slightly altered each time. When a new phantom is generated, one with substantially different parameter values, for instance, gray scale, location, or angle, the MatLab algorithm below produces $d(x,\mathcal{L}) = \frac{||\hat{x}||_2^2}{||x||_2^2} = 0.0003$ and $d(x,\mathcal{L}^{\perp}) = \frac{||\tilde{x}||_2^2}{||x||_2^2} = 0.9997$. We may interpret this result as the probability that the new image is much different from the previously generated ones. 0.9997 implies that is it highly unlikely that the ellipse does not belong to the "normal" set of ellipses.

An Application of Autoassociative Memory to the Identification of Anatomical Structures in a Head Section:


Scan images of normal head section, and corresponding images with graphically removed structure.
Load a pair of images, the original and modified images. Compute the Radon transform
Concatenate columns of Radon matrices into respective vectors, and take the
Concatenate Radon difference vector to matrix **D**.
Done?
No
Yes

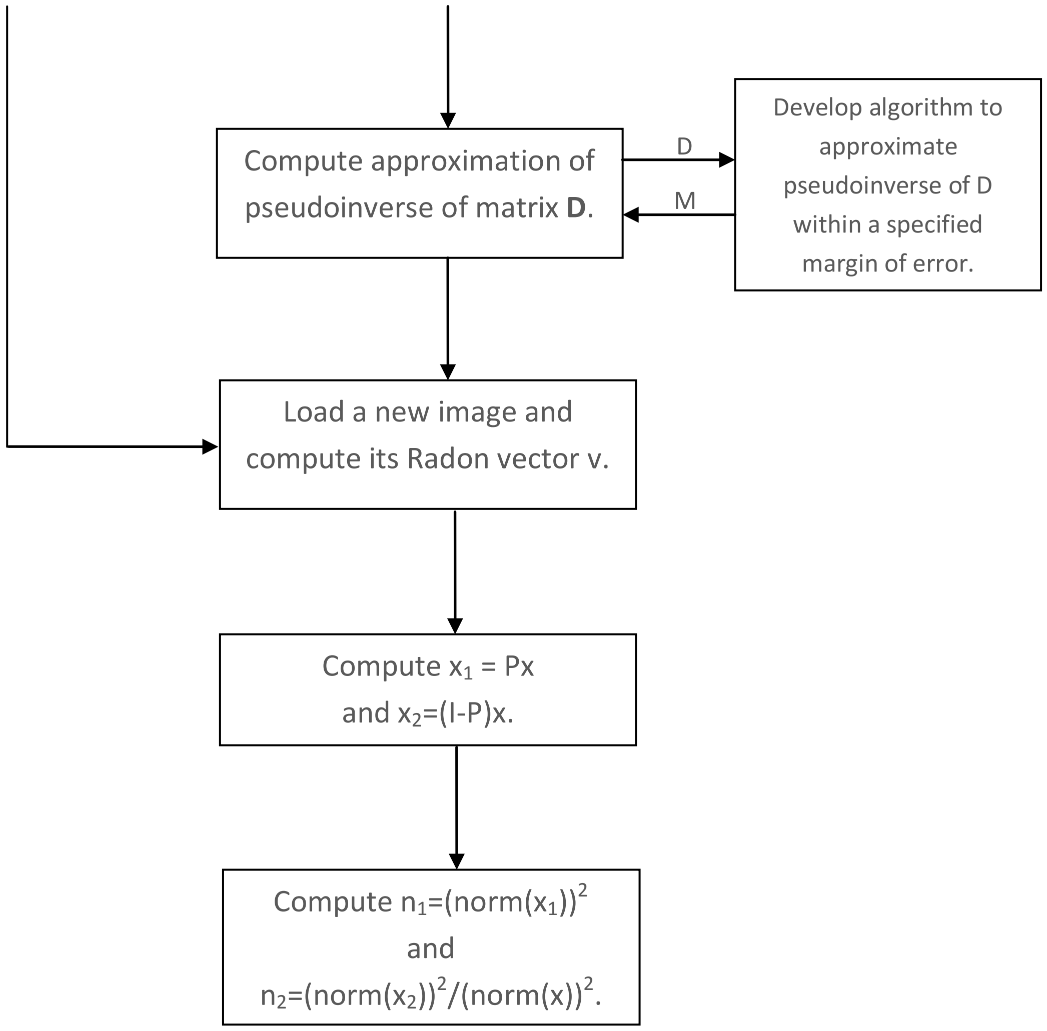

Compute approximation of pseudoinverse of matrix **D**.
D
M
Develop algorithm to approximate pseudoinverse of D within a specified margin of error.
Load a new image and compute its Radon vector v.
Compute $x_1 = Px$ and $x_2=(I-P)x$.
Compute $n_1=(\text{norm}(x_1))^2$ and $n_2=(\text{norm}(x_2))^2/(\text{norm}(x))^2$.

Fig. 6: Abnormality measure algorithm

In the flow chart above, n_1 is the probability that the anatomical structure in question is normal, or the distance of the vector Px from the subspace S, generated by the columns of the matrix D, consisting of the Radon vectors of normal anatomical structures. n_2 is the probability that the anatomical structure in question is abnormal. It is the novelty in the anatomical structure unexplained by S. Note that $n_1+n_2=1$.

4.4 The Hebbian Approach

In this section, we compare our construction of an associative memory to that of the Hebbian approach. Recall that a Hebbian projection operator sets up the connection weights as the sum of outer-product matrices of the input vectors $x_1, x_2..., x_n$:

$$\widehat{M} = \sum_{i=1}^{n} x_i x_i^T. \tag{82}$$

This approach does not produce any meaningful results when the input vectors are Radon transforms. However, we may orthonormalize the input vectors with our generalized Gram-Schmidt process, obtaining an approximately orthonormal set of vectors $q_1, q_2..., q_k$. We form the sum of the outer-product matrices, producing the projection operator$\widehat{P} = \sum_{i=1}^{k} q_i q_i^T$, where k ≤ n. In our method, we discard any q_i if its norm is too close to zero. Each matrix $q_i q_i^T$ is an elementary projection, so that for any vector x, the operation ($q_i q_i^T$)x projects x onto the unit vector q_i. Consequently, $\widehat{P}$x projects x onto the subspace $\mathcal{L}$ spanned by vectors $q_1, q_2..., q_k$. By Lemma 1 in Appendix A, $\widehat{P}x = \sum_{i=1}^{k} q_i q_i^T x = \sum_{i=1}^{k} (q_i^T x) q_i$, and so the two expansions are equivalent. We may view $\widehat{P}x$ as an expansion of a vector x in terms of bases vectors q_i. So we may compute $\widehat{P}x$ using either elementary projections or expanding in terms of basis vectors. The first approach has better numerical properties, while the second is more efficient. We use our metric

$$d(x, \mathcal{L}) = \frac{||\widehat{P}x||_2^2}{||x||_2^2} \tag{83}$$

to determine what percentage of x belongs to the subspace $\mathcal{L}$ of normal vectors. Our simulation experiments produce similar results to that of the previous section, when using the projection operator

$$\widehat{P}x = \sum_{i=1}^{k} q_i q_i^T x. \tag{84}$$

Appendix B shows the MatLab Code for the Hebbian approach.

4.5 Singular Value Decomposition (SVD)

Instead of orthonormalizing the columns of input matrix X to obtain the projection operator$\widehat{P} = \sum_{i=1}^{k} q_i q_i^T$, we may write X as a product of three matrices, so that

$$X = U \sum V^T. \tag{85}$$

The matrix U is orthogonal and has as many rows as X. The matrix V is orthogonal and has as many columns as X. The matrix $\sum$ is the same size as X, having nonzero elements on its main diagonal. The diagonal elements of $\sum$ are the singular values, and the columns of U and V are the left and right singular vectors.

The SVD says that for any linear transformation it is possible to choose an orthonormal basis for the domain, and a possibly different orthonormal basis for the range.

Orthogonal transforms preserve linear independence. Thus, the rank of any matrix is the number of nonzero singular values.

Let E$_k$ be the outer product of the kth left and right singular vectors

$$E_k = u_k v_k^T. \tag{86}$$

By [11], the matrix X may be expressed as a sum of rank-1 matrices

$$X = \sum_{k=1}^{n} \sigma_k E_k, \tag{87}$$

where the singular values decrease, i.e. $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$. If the columns of X are Radon transform vectors, and we truncate the above sum after r terms, our result is a rank-r approximation with less noise than the original matrix. The error in the approximation depends upon the magnitude of the neglected singular values. We keep in mind that $X = \sum_{k=1}^{n} \sigma_k E_k \approx X_{truk} = \sum_{k=1}^{r} \sigma_k E_k$, if $\sigma_{r+1}$is small where $\sigma_{r+1} \geq \sigma_{r+2} \geq ... \geq \sigma_n$. For instance, we may choose the smallest r so that $\frac{\sum_{k=1}^{r} \sigma_k}{\sum_{k=1}^{n} \sigma_k} > 0.95$, where the first r components capture at least 95% of the variance. The projection operator P $= X_{truk} X_{truk}^{+}$ is more stable than XX$^+$, as [16] shows. For a full rank matrix X, simulation results obtained using the projection operator $X_{truk} X_{truk}^{+}$ are close to the results obtained in Section 3.2. However, this TSVD approach only works for a full rank matrix X, whereas for our algorithm, as shown in Section 3.2, X need not be of full rank. More precisely, in[16] a comparative study on TSVD and Tikhonov regularization was done where the matrices with well-determined and ill-determined numerical ranks were defined. Loosely speaking, a well-determined rank means there is an index k such that σ_k is the smallest singular value sufficiently greater than zero, so that σ_{k+1} is so tiny that it is indistinguishable from zero. [16] goes on to show that for well-determined matrices, the TSVD and Tikhonov approaches produce the same results. The results of using TSVD with ill-determined matrices are typically unreliable. In particular,[16] states that for matrices with ill-determined numerical ranks, obtained from underlying ill-posed problems, the concept of rank has no intuitive interpretation. Examples of such problems are digital image restoration, solutions to integral equations in solid state physics, and the inverse Radon and Laplace transformations.

Appendix B shows the MatLab algorithm.

4.6 Principal Component Analysis (PCA)

The principal component analysis (PCA) approach is useful when data consists of several different units, requiring standardization. Since our data consists of the same unit of measure, we do not need to use this approach, and discuss it only to compare it to previous methods. The PCA approach is equivalent to the SVD approach of the previous section in that after standardizing, the analysis can be performed by the SVD approach of Section 4.5. Standardization consists of subtracting the mean of each column from the entire column to center the data in each column. Next, each value in a column is divided by the standard deviation of that column. In other words, we center the data at the mean and scale it by the standard deviation for each column. As in Section 4.5, we expand the standardized matrix as a sum of rank-1 matrices. The right singular vectors v_k's are the components, and the scaled left singular vectors, $\sigma_k \mathrm{u}_k$'s, the scores. Typically, PCAs are described in terms of the eigenvalues and eigenvectors of a covariance matrix XX^T. The idea is that the variance captured by the least important principal components is noise that may be rejected. Assuming the variables bear a linear relationship, they will lie in a line, plane, or hyperplane, and any noise items will pull them away from that respective line, plane, or hyperplane. Dropping the last principal components means flattening the data in a geometric sense, so we may eliminate some of the noise. As in the the previous section, if we choose the smallest r so that $\frac{\sum_{k=1}^{r} \sigma_k}{\sum_{k=1}^{n} \sigma_k}$ >0.95, the first r components capture at least 95% of the variance. So PCA "squeezes" as much information, or variance, as possible into the first principal components. In this case, the number of principal components needed to store the majority of the variance is significantly small. As in chapter 4.5, we compute the projection operator

$$\widehat{P} = X_{truk} X_{truk}^{+}. \tag{88}$$

For a standardized vector x, we compute the projection $\widehat{P}x$, then reverse the standardization to get back to the original units. We use our metric, namely $d(x, \mathcal{L}) = \frac{||\widehat{P}x||_2^2}{||x||_2^2}$, to measure the probability that x is normal. As a SVD approach, PCA requires that data matrix X be of full rank, whereas our approach does not make that assumption.

4.7 Identification of Anatomical Structures

We employ heteroassociative associative memory to compute the identification of anatomical structures, where the input and output matrices X and Y are distinct. The matrix X consists of Radon vectors of scanned images. The matrix Y consists of Radon vectors of the chosen anatomical structures in the corresponding image. Training the neural network consists of mapping the Radon vector of an image into the Radon vector of a chosen structure of that image. In a "real case" scenario, clinicians identify the structures in the images. The Radon transforms of the chosen structures are computed as in the prior section. We take the difference of a Radon vector of an image and the Radon vector of the same image, with a structure removed. When presented with an image, our projection operator maps the Radon vector of the image into the Radon vector of the structure in the image. An image of the structure is produced via the inverse Radon transform of the projected vector. We compute a projection operator for every anatomical structure of an image that we wish to identify. Geometrically, we may interpret this approach as taking a vector in $x \varepsilon R^m$, projecting that vector onto a subspace R_i via its corresponding projection matrix M_i, and producing an image of the i^{th} anatomical structure, i=1,...,p, with p the p^{th} anatomical structure in the image. We now may locate and produce each part of an image for which we have a corresponding projection matrix M_i, for i=1,...,p.

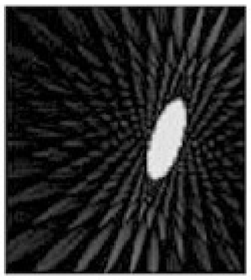
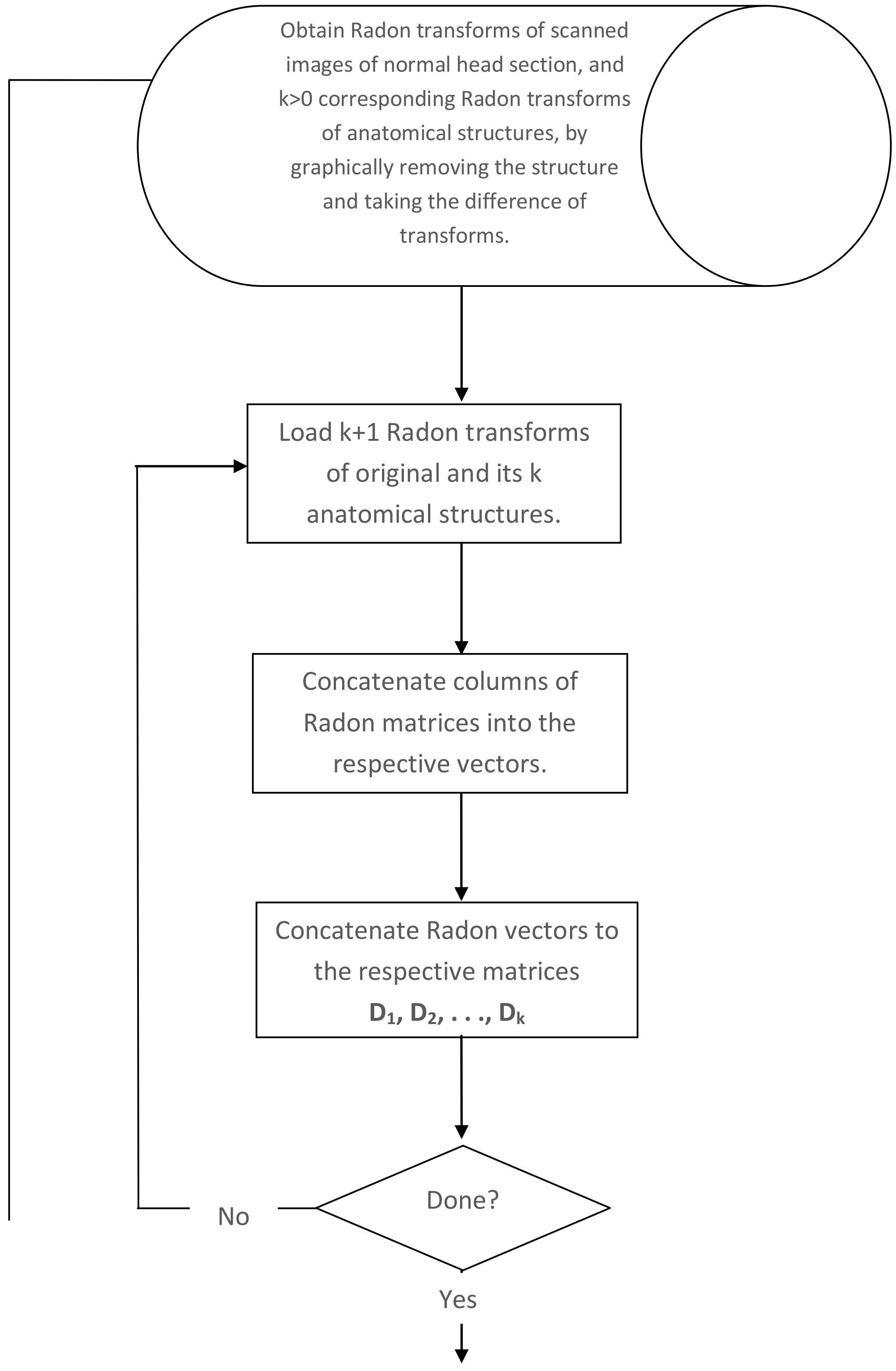


Fig. 7: Identification of ellipse number 3.

The above figure shows ellipse number 3, identified by projection matrix M_3, that was computed by our algorithm, with input $X^{3330\times 500}$ as 500 Radon transform vectors of 500 images, and 500 Radon transforms $Y^{3330\times 500}$of the corresponding ellipse number 3. The image above was produced by the Radon inverse of M_3x, where M_3 is the projection matrix for the training set of 500 images, and x is the Radon vector of a Shepp-Logan phantom that is not in the training set. In other words, if we are presented with a new image and we want an automated identification of anatomical structure number 3, we apply operator M_3 on the Radon vector x of the new image, then take the inverse Radon transform of M_3x to produce the anatomical structure 3 of the new image.

An Application of Heteroassociative Memory to the Identification of Anatomical Structures in a Head Section:


Obtain Radon transforms of scanned images of normal head section, and k>0 corresponding Radon transforms of anatomical structures, by graphically removing the structure and taking the difference of transforms.
Load k+1 Radon transforms of original and its k anatomical structures.
Concatenate columns of Radon matrices into the respective vectors.
Concatenate Radon vectors to the respective matrices D₁, D₂, . . ., Dₖ
Done?
No
Yes

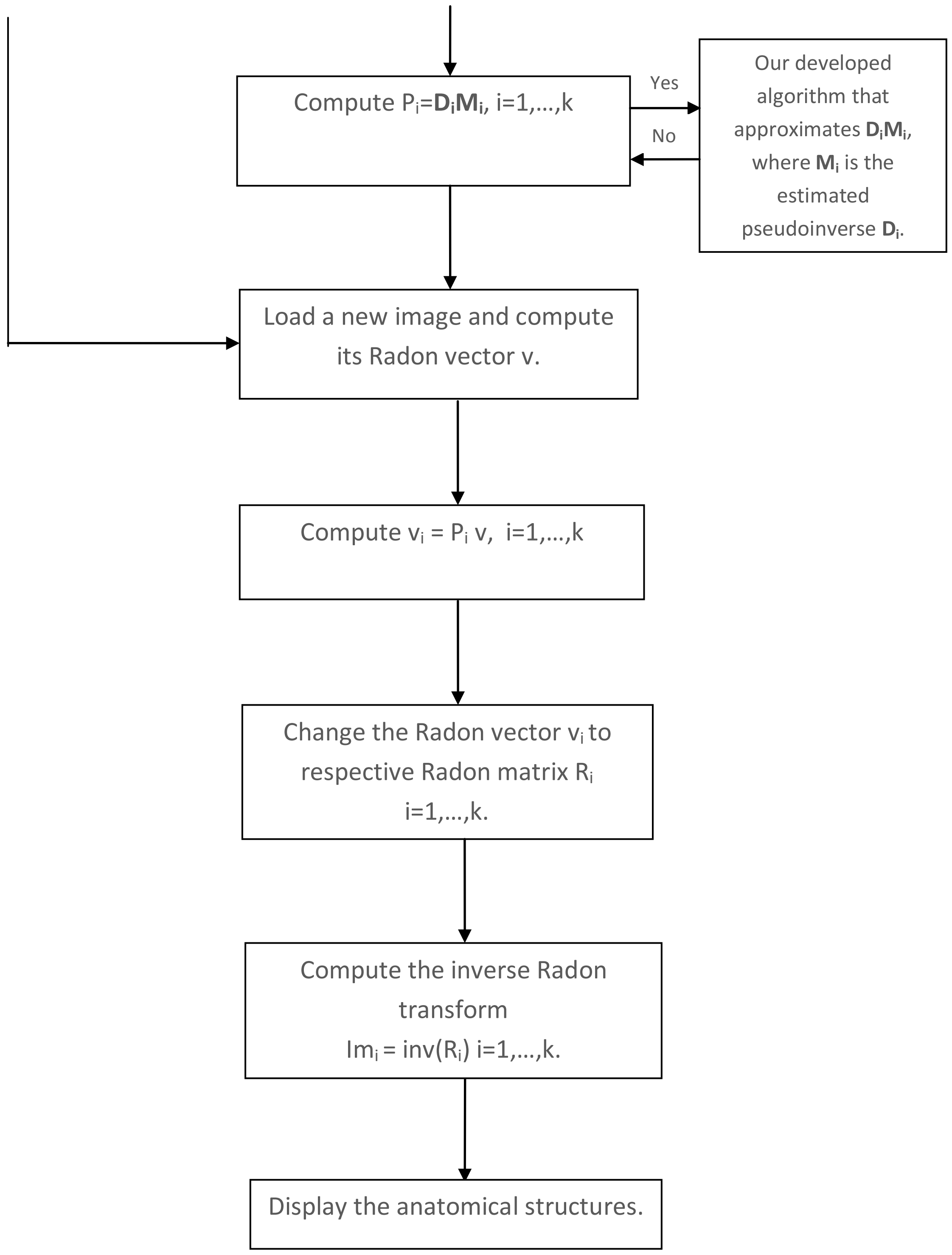

Compute $P_i = \mathbf{D_i M_i}$, i=1,…,k
Yes
No
Our developed algorithm that approximates $\mathbf{D_i M_i}$, where $\mathbf{M_i}$ is the estimated pseudoinverse $\mathbf{D_i}$.
Load a new image and compute its Radon vector v.
Compute $v_i = P_i v$, i=1,…,k
Change the Radon vector v_i to respective Radon matrix R_i i=1,…,k.
Compute the inverse Radon transform $Im_i = inv(R_i)$ i=1,…,k.
Display the anatomical structures.

Fig. 8: Identification of anatomical structures algorithm.

4.8 Anomaly Detection Algorithm

In this section, we develop an anomaly detection algorithm as an extension and generalization of the measure of abnormality algorithm of chapter 4.3. We also address the problem of how to choose the regularization parameter δ. Our extended anomaly detection algorithm is likely to be useful in scenarios where there are a very small number of abnormal instances and a large number of normal instances. In these scenarios, it is difficult to learn from normal examples to recognize anomalies. Our approach is based on the modified OLAM developed in the previous sections. It works best in situations where the number of features is much larger than the number of examples in the training set. Other uses of the algorithm, aside from detecting abnormal anatomical structures in CT scans, could be detecting anomalies in images where the number of features is large, detecting manufacturing flaws, detecting fraud, and monitoring machines in a data center. In all of these examples, we assume that the anomalies are rare, and that the number of features is large.

Consider a data set $X = (x_1, ..., x_m)$, $x_i \varepsilon R^n$ that represents a collection of observations, considered to be normal. Given a new example, $x_{test} \varepsilon R^n$, we want to know whether this new example is abnormal with a predetermined probability of a false-positive identification. In the previous section, our observed data set consists of Radon vectors of "normal" CT scans of an anatomical structure in a head section. We then compute the approximation of the projection operator P= XX$^+$and project x$_{\text{test}}$ onto subspace $\mathcal{L}$(x$_1$,..., x$_m$), obtaining $\hat{x}$ = Px$_{\text{test}}$ ε $\mathcal{L}$ and $\tilde{x} = (I - P)x_{\text{test}}$ ε $\mathcal{L}^{\perp}$. We use the following measure of normality $d(x, \mathcal{L}) = \frac{||\hat{x}||_2^2}{||x||_2^2}$. Since $d(x, \mathcal{L})$ $+d(x, \mathcal{L}^{\perp}) = 1$, we may interpret $d(x, \mathcal{L})$ as a probability that x is normal. However, in reality the data set is a sample from a population of random vectors in R^n, and so the measure $d(x, \mathcal{L})$ is a statistic with its own distribution. In order to determine if the value of $d(x, \mathcal{L})$ has a low probability, we would like to know the probability distribution of $d(x, \mathcal{L})$. For normal examples, the probability should be large, and small for abnormal examples. Without knowing the probability distribution of $d(x, \mathcal{L})$, we have no way of computing of how unusual the value of $d(x_{\text{test}}, \mathcal{L})$ is.

In this section, we determine a threshold value ρ such that Pr($d(x, \mathcal{L}) <$ ρ) < ε, for a chosen ε > 0. We then predict that x_{test} is abnormal if $d(x, \mathcal{L}) <$ ρ, and normal if $d(x, \mathcal{L}) \geq$ ρ with a predetermined probability ε of false-positive identification. Knowing how to compute Pr($d(x, \mathcal{L})$< ρ) is essential to detecting abnormalities. A difficulty arises in deriving the probability distribution for $d(x, \mathcal{L})$=$\frac{||\hat{x}||_2^2}{||x||_2^2}$, in that the numerator and the denominator are not statistically independent. Nevertheless, we will show that for x$_i$, i=1,. . . ,m, x$_i$ ~ N(μ$_i$,Iσ$_i$), and for a given ε > 0, we can compute ρ and the probability Pr($d(x, \mathcal{L})$< ρ) < ε.

Recall that each entry in the Radon vector is a weighted average of pixel gray scale values weighted by the ray segment lengths through the pixel segments. Hence, if we normalize each Radon vector entry by dividing by the length of the ray through the object, we can invoke the Central Limit Theorem and assert that each entry in the Radon vector is normally distributed. In what follows, we compute the z-score of the normalized Radon vector entry values by estimating the mean and standard deviation for each Radon vector entry i across the sample data. This justifies our assumption that z-score data is a multivalued standard normal distribution of vectors. that is, $x_i \sim N_m(0 ; I_m)$, for i=1,. . . ,m.

We make use of the following statistical proposition.

Proposition : If $x \sim N_m(0 ; I_m)$, and P is a projection matrix, then

$x^T P x \sim \chi^2_{rank(P)} = \chi^2_{tr(P)}$.

Proof: Since P is a projection matrix, P is symmetric and idempotent, that is, $P=P^T$ and $P^2 = P$. Hence $x^T P x = x^T P^T P x = (Px)^T (Px) = ||Px||_2^2$. But, it well-known that know that $||Px||_2^2 \sim \chi^2_{tr(P)}$

The following theorem is due to Cochran.

If $x \sim N_m(0 ; I_m)$ and if $P_1, \ldots, P_k$ are such that $\sum_{j=1}^{k} P_j = I_m$, and if $r_i = rank(P_i)$ i=1,. . . ,k, then any one of the following conditions implies the other two:

- $\sum_{j=1}^{k} r_j$= m

- $||Px||_2^2$ ~ $\chi^2_{tr(P)}$

- x^{T}P$_j$x is statistically independent of x^{T}P$_i$x for i $\neq$ j .

If P is a projection, then so is (I$_m$- P). Since P + (I$_m$ - P) = I$_m$ and tr(P) + tr(I$_m$ − P) = m, then by Cochran's theorem and Proposition 1, $||Px||_2^2$ and $||(I-P)x||_2^2$ are statistically independent and are $\chi^2_{tr(P)}$ and $\chi^2_{m-tr(P)}$ distributed, respectively. Hence, the ratio

$$\frac{\frac{||Px||_2^2}{tr(P)}}{\frac{||(I-P)x||_2^2}{m-tr(P)}} = \frac{(m-tr(P))||Px||_2^2}{(tr(P))||(I-P)x||_2^2} \tag{89}$$

is F-distributed, with parameters tr(P) and m-tr(P).

Given an ε >0, let ρ be the value such that Pr($d(x, \mathcal{L})$< ρ) < ε . We define ρ as the threshold such that if $d(x_{\text{test}}, \mathcal{L})$< ρ, we predict that x_{test} is abnormal, with the probability ε that the prediction is false positive.

For a given ε we will now derive the value of ρ.

$$Pr(d(x_{\text{test}}, \mathcal{L}) < \rho) = Pr(\frac{||\hat{x}||_2^2}{||x||_2^2} < \rho)$$

$$= Pr(\frac{||\hat{x}||_2^2}{||\hat{x}||_2^2 + ||\tilde{x}||_2^2} < \rho) = Pr(\frac{1}{1 + \frac{||\tilde{x}||_2^2}{||\hat{x}||_2^2}} < \rho)$$

$$= Pr(1 < (1 + \frac{||\tilde{x}||_2^2}{||\hat{x}||_2^2})\rho) = Pr((1 - \rho < \frac{||\tilde{x}||_2^2}{||\hat{x}||_2^{24.8}}\rho)$$

$$= Pr(\frac{1-\rho}{\rho} < \frac{||\tilde{x}||_2^2}{||\hat{x}||_2^2}) = 1 - Pr(\frac{||\tilde{x}||_2^2}{||\hat{x}||_2^2} \leq \frac{1-\rho}{\rho})$$

$$= 1 - Pr(\frac{(tr(P))||\tilde{x}||_2^2}{(m - tr(P))||\hat{x}||_2^2} \leq \frac{(tr(P))(1-\rho)}{(\text{m-tr(P)})\rho})$$

$$= 1 - F(\frac{(tr(P))(1-\rho)}{(\text{m-tr(P)})\rho}, m - tr(P), tr(P)) = \varepsilon.$$

Hence

$$F(\frac{(tr(P))(1-\rho)}{(\text{m-tr(P)})\rho}, m - tr(P), tr(P)) = 1 - \varepsilon \tag{90}$$

Let τ = F^{-1}(1- ε, tr(I-P), tr(P)), then $\tau = \frac{(tr(P))(1-\rho)}{(\text{m-tr(P)})\rho}$. We can now solve for ρ to obtain $\rho = \frac{tr(P)}{tr(P)+\tau(m-tr(P))}$. Hence, for a given $\varepsilon > 0$, we now know how to compute the threshold ρ. Also, for a given x_{test} if $\frac{||\hat{x}_{test}||_2^2}{||x_{test}||_2^2} < \rho$, we predict that x_{test} is abnormal with the probability ε that the prediction is false positive, a "false alarm."

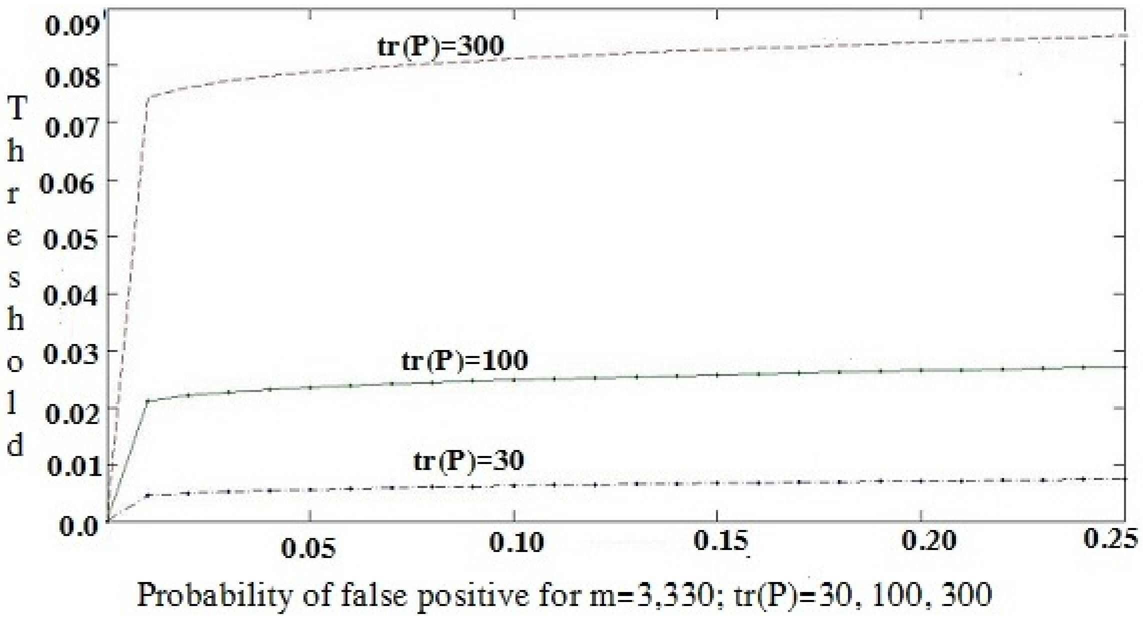


Fig. 9: Graph of false positive probability vs. the threshold.

Figure 9 illustrates the false positive probability error ε vs. threshold ρ, for n=3330, and three different ranks of projection operator P, 30, 100, and 300.

Example 1: Choose ε=0.1, and suppose m=3330 and tr(P) = 300, so τ= $F^{-1}(0.99,3030,300) = 1.1202$. Hence $\rho = \frac{tr(P)}{tr(P)+\tau(m-tr(P))} = \frac{300}{300+1.1202*3030} = 0.0812$. We conclude that if $d(x_{\text{test}}, \mathcal{L}) < 0.0812$, we declare x_{test} to be abnormal, with a probability of 0.1 of a false positive identification.

In other applications, if the random vector x ε R^n is not $N_n(0 ; I_n)$, then after standardization, it may be possible to try various transformations on non-Gaussian distributions to make the data Gaussian, for example log(x), log(x+1), log(x+c), $\sqrt{x}$, etc. We note that for a random vector x ε R^n with large n, it is possible to approximate the distribution of x by the Gaussian Mixture Model (GMM), a weighted sum of Gaussian distributions.[88] Here the means and diagonal covariance matrices as well as the weights can be found using the Expectation-Maximization algorithm, the EM algorithm.[89] For large n, the algorithm still applies, even if the distribution of x ε R^n is unknown.

A question still remains, namely how to choose the regularization parameter δ, and how to evaluate our anomaly detection system's performance. When detecting anomalies in CT images, we assume we have a very small number of abnormal examples and a large number normal training examples. Such training sets are known as skewed classes. In these cases, when we may have many different types of anomalies, it is difficult for any algorithm to learn from mostly normal examples what the anomalies look like. The optimal regularization is often determined by choosing the parameter that best fits the data. However, this choice in our applications may result in over-fitting the data. Over-fitting is likely to occur when the size of the training data set is small and the number of parameters in the model is large, which is what we have with Radon transform data.

4.9 Choice of Regularization Parameter

We determine our choice of regularization parameter by cross-validation, a technique that shows how the results of a statistical analysis generalize to an independent data set. It is mainly used in settings where the goal is prediction, as it gives an estimate as to how accurately a predictive model performs.

Specifically, we take the normal set of examples and split it into three disjoint subsets. The first subset consists of 60% of the sample reserved for training. The second 20%, the cross-validation set, is used to choose a parameter. The third, remaining 20%, the test set, is used to test the performance of the algorithm. We assume that most of the training sample is normal, and only a small percentage is known to be abnormal. We then split the known abnormal examples equally between the cross-validated sample and the test sample.

Our procedure for computing the regularization parameter value follows.

1. Estimate the projection matrix $P_{\hat{\delta}}$ using the first set of the normal examples.

2. Apply the estimated $P_{\hat{\delta}}$ on the cross-validation set.

3. For the cross-validation set, record the prediction results vs. the actual:

   - True positive: predicted abnormal and actual abnormal.

- True negative: predicted normal and actual normal.
- False negative: predicted normal and actual abnormal.
- False positive: predicted abnormal and actual normal.

4. Define precision, recall, and the F1 score, as follows:

$$Precision = \frac{TruePos.}{TruePos. + FalsePos.} \tag{91}$$

Precision is the fraction out of all the anomalies detected correctly.

$$Recall = \frac{TruePos.}{TruePos. + FalseNeg.} \tag{92}$$

Of all the true anomalies, the recall is the fraction detected correctly.

$$F_1\,score = 2\frac{Recall*Precision}{Recall + Precision} \tag{93}$$

5. Repeat steps 1 - 4 for several possible values of parameter δ, and choose the value that produces the highest F1 score for the cross-verified sample.

6. Once we choose the value of δ based on the cross-validation set sample, we can compute all three metrics by using the test set to evaluate the performance of the algorithm.

To understand the definition of the Recall , consider a training sample where only 0.1% are abnormal and suppose our prediction algorithm always predicts a normal case. We may claim that our algorithm is 99.9% correct. But this is a useless algorithm, as it would never predict an abnormal example, because the Recall shows that

$$Recall = \frac{TruePos.}{TruePos. + FalseNeg.} = \frac{0}{0 + FalseNeg.} = 0. \tag{94}$$

Chapter 5

5. CONCLUSION AND DISCUSSION

5.1 Conclusion

The first attempts applying artificial intelligence to capture the experience of experts were rule-based systems, relying on a predefined set of rules to capture the knowledge of human experts. However, a drawback of such systems is that they are not able to compress a large number of rules, possibly infinite, into a small, easily manageable set of rules. We demonstrated how we could capture the memory of experts using the computed orthonormal basis and projection matrix. Instead of running an instance through a large set of rules to reach a conclusion, we projected a vector onto a subspace composed of normal vectors. We then used a measure to determine how unusual the vector was from our subspace of normal vectors. A significant advantage over rule-based systems was our neural network's ability to better identify abnormal structures as the associative memory was updated with better data.

We used associative memory as our ANN, a content-addressable memory structure. Our content-addressable memory imitated the recollective abilities of the human brain, retrieving data based on how similar the input patterns were to the patterns in the memory. An advantage of this approach was that our associative memory was fault tolerant with respect to variations in the input patterns. For example, an output pattern resembling a stored pattern was obtained from a given input pattern. This contrasted with location addressable memories, returning "not found" when an exact match was not in the memory.

Kohonen formulated his OLAM to model the associative memory recall of the human brain. In its basic form, it consisted of a signal transfer in a physical network, where a spatial input pattern, the key, transformed into a corresponding output pattern, the recollection. The recall problem was to produce a matrix operator M, by which a pattern $y_k \varepsilon\ R^p$, for every k=1,...,m, was obtained from the pattern $x_k \varepsilon\ R^n$, via Mx_k= y_k, for all k ε {1,2,...,m}. Recall that what made $\widehat{M}$ unsuitable was that $YX^+$was not a continuous function of the data when X was not of full rank.[11] Also, if X was near-rank deficient, the solution would be sensitive to slight changes in the data. We developed an algorithm for the solution to a "nearby" perturbed least-squares problem. We demonstrated that the pseudoinverse solution was a special case of a more general solution, using Bellman's dynamic programming method. One of our accomplishments was producing an $\widehat{M}$ when X was not of full rank or was near-rank deficient.

Our first goal was to develop a stable OLAM algorithm that mapped input signals into output signals. Recall that unstable systems are very sensitive to noisy data, as small deviations in the input and/or output data produce very large errors in the solution. We used regularization to offset this problem, where an unstable system was replaced by a nearer, stabler system. Our algorithm solved the nearby system, giving a solution nearest to the solution of the unstable system.

Regularization methods had been proposed for OLAM models, but they lacked efficient algorithms that could solve them. We derived our algorithm using Bellman's principle of optimality. More specifically, we modified his algorithm, simplified it, and made it more computationally efficient. Later, we showed that our modified algorithm generalized the CGS process. These results were important not only to our applications, but had far reaching implications in other areas where unstable systems arise.

Our algorithms posed several advantages over physicians in interpreting medical images. Firstly, our OLAM could store the collective experience of many experts, and not just a single clinician's knowledge. Secondly, our OLAM provided a numerical measure that quantified the degree of dissimilarity between a given anatomical structure and the structures stored in the memory, a task difficult for a clinician to perform.

Our second goal was to provide a relatively noiseless encoding of data. We did not make use of feature vectors, as researchers often do. In the feature vector approach, features are obtained from segmented regions in an image. Properties of image segments, such as color, texture, shape, and positional composition, are then converted into feature vectors. Instead of using feature vectors, we trained our OLAM by using the Radon transforms of images.

Our third goal was to identify and display anatomical structures in CT images of human head sections. Our method was to choose an appropriate set of input and output vectors to train our OLAM to identify structures in a CT image. Training consisted of selecting a set of training images and their corresponding anatomical structures in the images. We used a set of Radon transforms of images as the input vectors, and the set of Radon transforms of anatomical structures in the images as the output vectors. We then applied our algorithm to the Radon transforms of an image, getting the Radon transforms of anatomical structures in the images. We then took the inverse of the Radon transform to obtain the image of an anatomical structure. The success of this method hinged upon on the clinicians' choice of training sets.

The significance of having a computerized system identifying normal and abnormal anatomical structures was self-evident. Significant questions to ponder are how well it would perform and how cost effective it would be. While our simulation experiments indicated that our algorithms work relatively well, independent experiments would still need to be done with real images selected by clinicians. Except for obtaining data that is accurate and representative of normal anatomical structures, the cost of doing this would be small since there would be no need for expensive equipment. The servers running the software are inexpensive, and their cost will decrease.

5.2 Discussion

Anomaly detection is often critical to a wide range of applications. For example, an anomalous CT scan of the brain may indicate the presence of a malignant tumor. Anomalies are patterns in data that do not conform to data which is considered to be a normal. A straightforward anomaly detection approach, therefore, is to define a region and declare any observation that is outside this region as abnormal or an anomaly. What makes this approach challenging is that it is difficult to represent a normal region. In addition, most anomaly detection algorithms determine the boundary between normal and anomalous data so that an observation that lies close to the boundary may produce a false-positive or a false-negative prediction. In this thesis, we apply autoassociative memory to anomaly detection. Generally, autoassociative memory has been used for retrieving the complete image when presentation with only partial image. To the best of our knowledge, this is the first time that autoassociative memory has been applied to anomaly detection. We address the problem of defining a normal data region and propose a method for identifying normal and anomalous instances with regions that do not necessarily share a boundary. This is a departure from most current anomaly detection techniques which usually separate normal and anomalous regions by a shared boundary.

We have developed a method for representing a normal region with the use of the generalized Gram-Schmidt procedure to obtain an approximate orthonormal basis for the subspace spanned by vectors deemed normal. The same procedure can be used to obtain an orthonormal basis for the anomalous region as well. The orthonormal basis for the normal subspace is then used to construct a projection operator on the normal subspace. When presented with a vector that represents a new image instance, we project the vector onto the normal subspace, and then apply a measure that determines the probability that a given instance is not a member of the normal region. This approach could be thought of as novelty detection, which aims at detecting previously unobserved instances. In addition to developing a measure of abnormality, we have imposed a probability distribution on this measure, so that it is possible to compute the boundary of the normal region for a given probability of false-positive identification. It is important to understand how to interpret the probability that an anatomical structure is abnormal. For example, if the algorithm determines that the probability that a structure is normal is 0.99, this means that 99% of the structure may be reproduced by the orthonormal basis of the normal subspace. If, for a given probability error of a false-positive decision, we use the corresponding threshold value to predict an abnormal structure, this does not necessarily imply that the structure is abnormal. It means that the structure cannot be reproduced by the orthonormal bases of the observed normal subspace for a given probability of a false positive decision. In other words, that structure does not resemble anything in the associative memory's subspace, and does not necessarily mean that it is abnormal.

When labeled data is available, normal and anomalous regions can be separated by the region that represents the as yet unobserved instances. So, should an observation fall outside both regions, we declare it to be an observation that requires further studies by the experts. In other words, our algorithm is capable of saying that it does not know if the given instance is normal or abnormal because it is something that cannot be explained by the orthonormal bases of either region. This is a novel departure from most existing anomaly detection techniques, where the normal and anomalous regions share a boundary. It should be noted, however, that obtaining labeled data that is accurate as well as representative of all normal and abnormal types, is often expensive due to the fact that labeling requires the work of experts. Moreover, the training data is only as good as the experts and is never complete.

Most anomaly detection techniques are determined by the nature and availability of the data. Our method is no exception, and is only applicable to situations where the observation vectors are of high dimensionality relative to the number of observations. But even a limited number of labeled data is very useful. We can use the available labeled data to fine tune our regularization parameter and to assess the performance of our anomaly detection algorithms, as shown in Section 4.9. But when there is no labeled data available, the modified OLAM model is still capable of identifying anomalous instances for a given probability of a false-positive identification, known in hypothesis testing as a type I error.

We have discussed two other approaches that could be used to detect abnormal structures in CT images, namely the truncated SVD and PCA. We compared the simulation results using these methods with our algorithms. Our experiments indicated that all of our procedures produced nearly the same results when the input data matrix X is of full rank. However, our algorithms do not make the assumption of full rank, which is more applicable to "real world" scenarios.

We have also applied heteroassociative memory to the identification and display of anatomical structures in CT images. The associative mapping is between data that represents different type of instances. In our case, the input data are Radon transform vectors of CT scans of a head section, and the output data are Radon transforms of a specific anatomical part in the CT scan. For a given CT image, the algorithms identify the anatomical parts in the image and then display them. This time, for a given input matrix X and output matrix Y, we approximate the projection operator YX^+, instead of XX^+, as is done with autoassociative memory. The identification and display of anatomical parts of a CT image can be useful for training purposes. We point out that the identified and displayed parts are approximations of the parts in a specific CT scan, and not just look-ups of some stored images. Hence the identified and displayed parts may be useful for diagnostic purposes as well.

5.3 Future Research

A topic we would like to further explore is to compare supervised learning to our method. We would need to train a neural network by a supervised learning algorithm such as backpropagation. However, a drawback of supervised learning is that it always gives an answer, even when the answer is “I don’t know.” This conflicts with our approach. In our approach, we generate two subspaces, rather than one, or a subspace made up of normal structures, and a subspace made up of abnormal structures. For each of the two subspaces, we compute their orthonormal bases and projection matrices. Note, as shown in the Venn diagram below, if a vector does not belong to the normal subspace, this does not imply that it resides in the abnormal subspace. The idea is that if the projection operators for the normal and abnormal subspaces complement each other, in the sense that if the normal projection operator applied to a vector is not in the normal subspace, and the abnormal projection operator applied to the vector is in the abnormal subspace, we may conclude with confidence that the vector belongs to the abnormal subspace. In the reverse case, we conclude that the vector is normal. The only time we are fairly certain of the result is when the probability of normality is high at the same time that the probability of abnormality is low, or when normality is low and abnormality is high. In all other cases, further investigation by clinicians is required, as the measures for normal and abnormal do not complement each other. If the probability is low for normal and abnormal case, the structure is a good potential addition to either the normal or abnormal sets, depending on the experts' decisions. It is a good addition because there is no other vector or linear combination of the basis that can reproduce it, and, as a result, it expands the subspace to another dimension. If the probability is high for both normal and abnormal cases, this implies that some of the training data had been included in both subspaces by mistake.

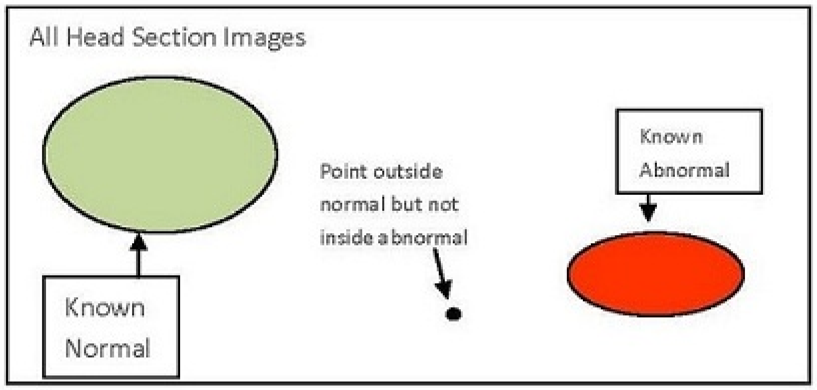


Fig. 10: A Venn diagram of normal, abnormal, and unknown instances.

We wish to further explore this approach of using two subspaces, one consisting of vectors known to be normal, and the other consisting of vectors known to be abnormal. As both subspaces grow with new data, the system produces fewer undecided results. We would then be able to compare the results to the backpropagation approach.

Another topic we wish to explore is to apply the anomaly detection algorithm of Section 4.8 to the detection of anomalies in images where the number of features is relatively large. This could be used to detect manufacturing flaws and fraud.

Finally, we would like to test our proposed methods on real images. However, that would require substantial resources in the form of clinical experts, real images, and computing facilities, which we postpone to a future study.

5.4 Closing Statement

In medicine, ANN's are actively being used in locating previously undetected patterns in enormous collections of research data used for detecting characteristics in medical images. We have developed a simple unsupervised learning model that can detect anomalies as well as identify anatomical structures. Our method of identifying abnormal anatomical structures in CT images for given probability of a false positive identification represents a new addition to the analysis of medical images. We have also proposed an encoding of data for CT images that is relatively noiseless. Another novel feature of our model is that it can conclude that no decision can be made based on the available data. This contrasts with many ANN models, in particular, supervised learning models, that always make decisions, even when they lack sufficient data.

REFERENCES

1. Kohonen,T. Self-organization and Associated Memory. Berlin: Heidelberg. 1988.

2. Fan Y., Kalaba R. "Dynamic programming and pseudoinverses." Applied Mathematics and Computation. 2003. p. 139.

3. Wang J. "Networks for computing pseudoinverses of rank deficient matrices." SIAM Journal on Scientific Computing. 1997. pp. 1479 -1493.

4. R. Kalaba, R., Rasakhoo. "Algorithms for generalized inverses." Journal of Optimization Theory and Applications. 1986. pp. 427-435.

5. Udwadia, Kalaba, R. "An alternative proof of the Greville formula." Journal of Optimization Theory and Applications. 1997. 94: 23-28.

6. Albert, A. Regression and the Moore-Penrose Pseudoinverse. Academic Press. 1972.

7. Kostko, B. Neural Networks and Fuzzy Systems. Prentice Hall. 1992.

8. Kalaba, R., Kim, M., Moore, J.E. "Linear programming and simple associative memories." Applied Mathematics and Computation. 1990. 40: 203-214.

9. Ojal, E., Kohonen, T. "The subspace learning algorithm as a formalism for pattern recognition and neural networks." Neural Networks. 1988. 1: 277-284.

10. Riihimaki E., Hall L., Kohonen T., Epistola P. "Application of computerized pattern recognition technique to the identification of abnormal brain images." Nuclear Medicine. 1975.

11. Golub, G. H., Van Loan, C.F. Matrix Computations. The John Hopkins University Press. 2004.

12. Bellman, R., Kalaba, E., eds. Modern Analytic and Computational Methods in Science and Mathematics. American Elsevier Publishing Company. 1966.

13. Zhang, B.-L., Zhang, H., and Ge, S. "Face recognition by applying wavelet subband representation and kernel associative memory." IEEE Transactions On Neural Networks. 2004. 15.

14. Bellman, R.E. 1957. Dynamic Programming. Princeton University Press. Princeton, NJ. 2003.

15. Engl, H., Hanke, M., Neubauer A. Regularization of Inverse Problems. Kluwer Academic Publishers. 2000.

16. Hansen, P.C. "Rank-deficient and discrete ill-posed problems." SIAM. Philadelphia, PA. 1997.

17. Vogel, C.R. Computational Methods for Inverse Problems. SIAM: Philadelphia, PA. 2002.

18. Golub, G. H. "Some modified eigenvalue problems." SIAM Review. 1973. 15: 318-334.

19. Eldén, L. "Algorithms for regularization of ill-conditioned least squares problems." BIT. 1977. 17: 134-145.

20. Tychonoff, A., Arsenin, V. Solution of Ill-posed Problems. Winston and Sons. 1997.

21. Hoerl, A. "Application of ridge analysis to regression problems." Chemical Engineering Progress. 58: 54-59.

22. Hoerl, A., Kennard, R. "Ridge regression: Biased estimation for non-orthogonal problems." Technometrics. 42: 1.

23. Zou L.-H., Jun, L. "Optimal linear associative memories for noise rejection". Circuits and Systems. 1992.

24. Zou L.-H., Jun, L. "Linear associative memories with optimal rejection to colored input noise." Circuits and Systems II: Analog and Digital Signal Processing. 1997. 44: 990-1000.

25. Cherkassky V., Fassett, K., Vassilas, N. "Linear algebra approach to neural associative memories and noise performance of neural classifiers." IEEE Transactions on Computers. 1991. 40.

26. Murakami K., Aibara T. "Least squares associative memory and a theoretical comparison of its performance." IEEE Transactions on Systems, Machines, and Cybernetics. 1989. 19: 5.

27. Riihimaki E., Hall L., Kohonen T., Epistola P. "Application of a computerized pattern recognition technique to identification of abnormal brain images." 4th International Symposium of Nuclear Medicine. 1975.

28. Ge J., Sahiner B, Hadjiiski L.M., et al. "Computer aided detection of clusters of microcalcifications on full field digital mammograms." Medical Physics 2006. 33: 2975–88.

29. Lo S.C.B., Chan H.P., Lin J.S., Li H, Freedman M.T., Mun S.K.. "Artificial convolution neural network for medical image pattern recognition." 1995. 8: 1201–1214.

30. Nagel R.H., Nishikawa R.M., Papaioannou J., Doi K. "Analysis of methods for reducing false positives in the automated detection of clustered microcalcifications in mammograms." Medical Physics. 1998. 25: 1502–1506.

31. Papadopoulossa D.I., Fotiadisb A., Likasb. "An automatic microcalcification detection system based on a hybrid neural network classifier." Artificial Intelligence in Medicine. 2002. 25: 149–167.

32. Christoyiani E., Dermatas G., Kokkinakis. "Fast detection of masses in computer aided mammography." IEEE Signal Processing Magazine. 2000. 17: 54–64.

33. Patrocinio A.C., Schiabel H., Benatti R.H., Goes C.E., Nunes F.L.S. "Investigation of clustered microcalcification features for an automated classifier as part of a mammography CAD scheme." Proceedings of the 22nd Annual EMBS International Conference. 2000. pp. 1203–1205.

34. Setiono R. "Extracting rules from pruned neural networks for breast cancer diagnosis." Artificial Intelligence in Medicine. 1996. 8: 37–51.

35. Xu X.W., Doi K., Kobayashi T., MacMahon H., Giger M.L.. "Development of an improved CAD scheme for automated detection of lung nodules in digital chest images." Medical Physics. 1997. 24: 1395–1403.

36. Zhou Z.H., Jiang Y., Yang Y.B., Chen S.F. "Lung cancer cell identification based on artificial neural network ensembles." Artificial Intelligence in Medicine. 2002. 24: 25–36.

37. Keserci B., Yoshida H. "Computerized detection of pulmonary nodules in chest radiographs based on morphological features and wavelet snake model." Medical Image Analysis. 2002. 6: 431–47.

38. Qian W., Zhukov T., Song D.S., Tockman M.S. "Computerized analysis of cellular features and biomarkers for cytologic diagnosis of early lung cancer." Analytical and Quantitative Cytology and Histology. 2007. 29: 103–111.

39. Coppini G, Diciotti S, Falchini M, Villari N, Valli G. "Neural networks for computer-aided diagnosis: detection of lung nodules in chest radiograms." IEEE Transactions on Information Technology in Biomedicine. 2003. 7: 344–57.

40. Mohamed E.I., Maiolo C., Linder R., Poppl S.J., De Lorenzo A. "Artificial neural network analysis: a novel application for predicting site-specific bone mineral density." Acta Diabetologica. 2003. 40: 19–22.

41. Scott J.A. “The lung scan and the abnormal chest X-ray: Difficult diagnoses.” Nuclear Medicine Communications. 2004. 25: 1137–41.

42. Hopfield, J.J. "Neural networks and physical systems with emergent collective computational abilities." Proceedings of the National Academy of Sciences of the USA. 1982. 79: 2554-2558.

43. Hebb, D.O. Organization of Behavior. New York: Wiley Hertz, J. 1949.

44. Krogh, A., Palmer, R.G. Introduction to the Theory of Neural Computation. Redwood City, CA: Addison-Wesley. 1991.

45. McCullough, W.S., Pitts, W.H. “A logical calculus of the ideas immanent in nervous activity.” Bulletin of Mathematical Biophysics. 1943. 5: 115-133.

46. Polyn, S.M., Kahana, M.J. “Memory search and the neural representation of context.” Trends in Cognitive Sciences. 2008. 12: 24-30.

47. Rizzuto, D.S., Kahana, M.J. “An autoassociative neural network model of paired-associate learning.” Neural Computation. 2001. 13: 2075-2092.

48. Sinha, U., Hooshang, K. “Principal Component Analysis for content based image retrieval.” RadioGraphics. 2002. 22: 1271–1289

49. S. W. Zhang, A. G. Constantinides, A.G. and L. H. Zou, “Further noise rejection in linear associative memories.” Neural Networks. 1992: 163–168.

50. Stiles, G.S., Deng, D. “On the effect of noise on the Moore–Penrose generalized inverse associative memory.” IEEE Transactions on Systems, Machines, and Cybernetics. 1988. 18: 358–360.

51. Murakami, K., Aibara, T. “Optimal association with partly missing key vectors.” Biological Cybernetics. 1982. 44: 154–155.

52. Murakami, K.; Aibara, T. “An improvement on the Moore–Penrose generalized inverse associative memory.” IEEE Transactions on Systems, Machines, and Cybernetics. 1987. 17: 699–707.

53. Murakami, K.; Aibara, T.; “Least squares associative memory and theoretical comparison of its performance.” IEEE Transactions on Systems, Machines, and Cybernetics. 1989. 19: 1230–1234.

54. Olivier, P.D. “Optimal noise rejection in linear associative memories.” IEEE Transactions on Systems, Machines, and Cybernetics. 1988. 18: 814–815.

55. Kenji, M., Aibara, T. “Least squares associative memory and a theoretical comparison of its performance.” IEEE Transactions on Systems, Machines, and Cybernatics. 1989. 19:1230 - 1234.

56. Chan H.P., Sahiner B., Wagner R.F., et al. “Classifier design for computer aided diagnosis: Effects of finite sample size on the mean performance of classical and neural network classifiers.” Medical Physics. 1999. 26: 2654– 2668.

57. Sahiner B., Chan H. P, Petrich N., et al. “Feature selection and classifier performance in computer aided diagnosis: The effect of finite sample size.” Medical Physics. 2000. 27: 1509-1522.Rasakhoo

58. Minsky, M. L., Papert, S. A. Perceptrons. Cambridge, MA: MIT Press. 1969.

59. Hopfield, J. J. "Neural networks and physical systems with emergent collective computational abilities." Proceedings of the National Academy of Sciences USA. 1982. 81: 2554-2558.

60. Hopfield, J. J., Tank, D. W. "Neural computation of decisions in optimization problems." Biological Cybernetics. 1985. 52: 141-152.

61. McCulloch, W. S., and Pitts, W. "A logical calculus of ideas imminent in nervous activity." Bulletin of Mathematical Biophysics. 1943. 5: 115-133

62. Harman, H. H. Modern Factor Analysis. Chicago: University of Chicago Press. 1976.

63. Morrison, D. F. Multivariate Statistical Methods. New York: McGraw-Hill. 1976.

64. Kohonen, T. “Correlation matrix memories,” IEEE Trans. Comput.. 1972. 21: 353–359.

65. Abdi, H., Valentin, D. and O’Toole, A. J. “A generalized autoassociative model for face processing and sex categorization: From principal components to multivariate analysis.” Optimality in Biological and Artificial Networks. 1997. pp. 317-337.

66. J. A. Anderson, “A simple neural network generating an interactive memory.” Mathematical Bioscience. 1972. 14: 197–220.

67. Hebb D. The Organization of Behavior. New York: Wiley. 1949.

68. Zhenghao S., Lifeng H., “Application of neural networks in medical image processing.” Proceedings of the Second International Symposium on Networking and Network Security. 2010. pp. 23-26.

69. Cichocki, A., Unbehauen, R., Lendl, M., Weinzierl, K. "Neural networks for linear inverse problems with incomplete data especially in applications to signal and image reconstruction." Neurocomputing. 1995. 8: 7-41.

70. Warsito, W., Fan, L.S. "Neural network based multi-criterion optimization image reconstruction technique for imaging two and three phase flow systems using electrical capacitance tomography." Chemical Engineering and Processing. 2001. 42: 2198-2210.

71. Su, B., Zhang, Y., Peng, L., Yao, D., Zhang, B. "The qualitative use of simultaneous iterative reconstruction technique for electrical capacitance tomography." Chemical Engineering Journal. 2000. 77: 37-41.

72. C. Hansen. "The truncated SVD as a method for regularization." BIT. 1987. 27: 534-553.

72. Wang, Y. , Wahl, F.M. "A vector-entropy optimization-based neural network approach to image reconstruction from projections." IEEE Trans. Neural Networks. 1997. 8: 1008-1014.

73. Wang, Y., Heng, P. , Wahl, F.M. "Image reconstructions from two orthogonal projections." International Journal of Imaging Systems and Technology. 2003: 141-145.

74. Russell, S., Norvig, P. Artificial Intelligence: A Modern Approach. 578.

75. Lisboa, P. J. G. "Neural networks in medical journals: Current trends and implications for BioPattern." Proceedings of the First European Workshop on Assessment of Diagnostic Performance (EWADP). 2004. 99-112.

76. Lisboa P. J. G. "A review of evidence of health benefit from artificial neural networks in medical intervention." Neural Networks. 2002. 15:11-39.

77. Baxt, W. G. (1995). "Application of artificial neural networks to clinical medicine." The Lancet. 1995. 1135-1138.

78. Miller, A., Blott, B., and Hames, T. "Review of neural network applications in medical imaging and signal processing." Medical and Biological Engineering and Computing. 1992. 30: 449- 464.

79. Weinstein, J., Kohn, K., Grever, M., et al. "Neural computing in cancer drug development: Predicting mechanism of action." Science. 1992. 258: 447-451.

81. Egmont-Petersena, M., de Ridderb, D., Handelsc, H. Pattern Recognition. 2002. 35: 2279–2301.

82. Fricker, J. “Artificial neural networks improve diagnosis of acute myocardial infarction.” The Lancet. 1997. 96: 1798-1802.

83. Hungl, M.S., Shanker, M., and Hu, M.Y. “Estimating breast cancer risks using neural networks.” Journal of the Operational Research Society. 2002. 53: 222-231.

84. Egmont-Petersena, M., de Ridderb, D., Handelsc, H. “Image processing with neural networks-a review.” Pattern Recognition. 2002. 35: 2279–2301.

85. Brause, R.W. “Medical analysis and diagnosis by neural networks.” Lecture Notes in Computer Science. 2001.

86. Miller, A., Blott, B., and Hames, T. “Review of neural network applications in medical imaging and signal processing.” Medical and Biological Engineering and Computing. 1992. 30: 449-464.

87. Weinstein, J., Kohn, K., Grever, M., et al. “Neural computing in cancer drug development: Predicting mechanism of action.” Science. 258: 447-451.

88. Silverman B. W. (1986) Density Estimation for Statistics and Data Analysis. Chapman and Hall.

89. Dempster, A., N. Laird, and D. Rubin (1977). “Maximum likelihood from incomplete data via the EM algorithm (with discussion).” Journal of the Royal Statistical Society, Series B 39. 1: 1–38.

99. Kohonen, T. (1972). "Correlation matrix memories." IEEE Transactions on Computers, C-21. pp. 353-359.

100. Kohonen, T. and Ruohonen, M. (1973). "Representation of associated data by matrix operators," IEEE Transactions on Computers, C-22. pp. 701-702.

101. Kosko, B. (1987). "Adaptive bidirectional associative memories," Applied Optics. 26. pp. 4947-4960.

102. Okajima, K., Tanaka, S., and Fujiwara, S. (1987). "A heteroassociative memory network with feedback connection," in Proceedings of the IEEE First International Conference on Neural Networks. 2. pp. 711-718.

103. Chiueh, T. D. and Goodman, R. M. (1988). "High capacity exponential associative memory," in Proceedings of the IEEE First International Conference on Neural Networks. pp. 153-160.

104. Hassoun, M. H. (1995). Fundamentals of Artificial Neural Networks. MIT Press.

Appendix A: Mathematical Background

The following is a brief review of linear algebra topics necessary to understand this work. Much of the notations, definitions, and matrix properties below are provided in [1,11]. Examples of the matrix properties are given in [11]. Near the end of this section, we prove that the CGS method is the same as using elementary projections.

Scalars are lower-case Greek letters, and vectors lower-case Roman letters.

A column vector x in the vector space R^n, i.e. x εR^n, is x^T=$(\zeta_1, \zeta_2, ...\zeta_n)$. x^T stands for the transpose of vector x.

Rectangular matrices are upper-case Roman letters of the form A$^{m\times n}$=(α_{kl}) , with k=1,2,...,m, and l=1,2,...,n. α_{kl} is the scalar entry of A.

The l^{th} column vector of A is A(:,l).

The Euclidean norm of a vector x εR^n is

$$||x|| = \sqrt{(x,x)} = \sqrt{x^T x} = \sqrt{\zeta_1^2 + \zeta_2^2 + ... + \zeta_n^2}. \tag{95}$$

The Euclidean norm of a matrix A^{mxn} =$(\zeta_{ij})^{mxn}$ is

$$||A|| = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\zeta_{ij}^2} = \sqrt{tr(A^T A)} \tag{96}$$

$tr(A^TA)$ is the sum of all the diagonal elements of A^TA, called the trace of A^TA.

The inverse of a square matrix A is the square matrixA^{-1}, such that AA^{-1} =A^{-1} A= I, where I is an identity matrix.

The Euclidean norm is sometimes written $||.||_2$. We may refer to it the L^2 norm.

The angle ϑ between two vectors x εR^n and y εR^n is $\cos(\vartheta) = \frac{x^Ty}{||x||||y||}$, $with$ $|x|\neq$||y|$\neq$|0.

Two vectors x εR^n and y εR^n are orthogonal, x$\perp$ y, iff (x,y)=x^Ty=0. (x,y) and x^Ty are the inner products of x and y.

The vectors x_1,x_2...x_n are linearly independent iff their linear combination $\alpha_1x_1+\alpha_2x_2+...+\alpha_nx_n$ cannot be the zero vector, unless α_1=α_2=...=α_n=0.

x_1,x_2...x_k are the vectors given in the previous definition. All of the linear combinations of the vectors form the subspace of R^n, $\mathcal{L}(x_1,x_2...x_k)$. The subspace is of dimension k.

The linearly independent vectors x_1,x_2...x_n, spanning $\mathcal{L}$ in R^n, are the bases of $\mathcal{L}(x_1,x_2...x_k)$.

A vector x is orthogonal to subspace $\mathcal{L}$ iff it is orthogonal to every vector in $\mathcal{L}$.

Two subspaces, $\mathcal{L}_1$and $\mathcal{L}_2$, are orthogonal to each other, i.e. $\mathcal{L}_1\perp$ $\mathcal{L}_2$, iff every vector in $\mathcal{L}_1$ is orthogonal to every vector in $\mathcal{L}_2$.

If $\mathcal{L}$ is a subspace of the vector space R^n , the subspace$\mathcal{L}^{\perp}$ is the orthogonal complement of $\mathcal{L}$. $\mathcal{L}^{\perp}$ consists of all the vectors in R^n orthogonal to $\mathcal{L}$.

If $\mathcal{L}$ is a subspace of R^n, an arbitrary vector x εR^n may be written uniquely as the sum x=$\hat{x}$+$\tilde{x}$, of two vectors, $\hat{x}\varepsilon\mathcal{L}$, and $\tilde{x}\varepsilon\mathcal{L}^{\perp}$. $\hat{x}$ is the orthogonal projection of x on $\mathcal{L}$. $\tilde{x}$ is the orthogonal projection of x on $\mathcal{L}^{\perp}$.

C = xy^T is the outer product of two vectors, the matrix C. The outer product plays a major role later, prompting the example:

If x =$[{}^1_3]$ and y^T=[1,2,3] then C=xy^T $= \begin{bmatrix} 1\times 1 & 1\times 2 & 1\times 3 \\ 3\times 1 & 3\times 2 & 3\times 3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 6 & 9 \end{bmatrix}$.

The range of a matrix A is the set of vectors Ax, for all vectors x εR^n,$\mathcal{R}$(A).

If the columns of matrix A are linearly independent, they span $\mathcal{R}$(A).

The rank of a matrix A, rank(A), is the dimension of $\mathcal{R}$(A) .

The rectangular matrix $A^{m\times n}$ is of full rank iff rank(A) = min(m,n).

The rank(A) = rank(A^T) = rank(A^TA) = rank(AA^T). This implies that, if the columns of matrix A are linearly independent, A^TA is a matrix of full rank.

Let A be the square matrix $A^{n\times n}$. If A's range is R^n, and the null space the zero vector, A is a non-singular matrix. Otherwise, A is a singular matrix.

Let A be as above, now with Ax = λx, and λ a non-zero real scalar. Then x is an eigenvector of A, and λ is an eigenvalue of A.

If Ax $=$ λx or (A - λI)x=0 have solutions aside from x=0, the determinant |A - λI| $=$ 0 is an n-degree polynomial in λ, called the characteristic equation of A, whose roots are the eigenvalues of A.

The condition of a rectangular matrix A of full rank, is k_2(A) $= \frac{\sigma_{max}(A)}{\sigma_{min}(A)}$, $\sigma_{min}(A) \neq 0$.$\sigma_{max}(A)$ is the largest eigenvalue of A and $\sigma_{min}(A)$ is the smallest of eigenvalue of A.

A large k_2(A) implies that the LS problem is ill-conditioned.[11] That is, the solution is sensitive to small perturbations in the input and/or output data.

We present the example given in [11]:

Suppose A $= \begin{bmatrix} 1 & 0 \\ 0 & 10^{-6} \\ 0 & 0 \end{bmatrix}$, b$= \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, then k_2 (A)$=10^6$. The LS solution is $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

If we perturb A, that is, make

A+ΔA $= \begin{bmatrix} 1 & 0 \\ 0 & 10^{-6} \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 10^{-8} \end{bmatrix}$, the new LS solution is $\begin{bmatrix} 1 \\ 9999 \end{bmatrix}$! A nearly 10^4 fold increase in the LS solution occurs with a 10^{-8}change in the input matrix A. Clearly, a small perturbation in the matrix A accompanies a much greater perturbation in the LS solution.

If P is a matrix and P^2=P, P is an idempotent matrix. This implies that I - P is an idempotent matrix.

Matrices of the form P = I - αuv^T,$\alpha\epsilon$R, are elementary matrices. uv^T is an outer product of two vectors u and v, of the same dimension.

P = I - $\frac{uu^T}{u^Tu}$ and $\frac{uu^T}{u^Tu}$ are projection matrices. It is shown, that for the vector x εR^{n}[11],

$$(I - \frac{uu^T}{u^Tu})x \tag{97}$$

projects x on $\mathcal{L}^{\perp}(u)$. And

$$\frac{uu^T}{u^Tu}x \tag{98}$$

projects x on $\mathcal{L}$ (u).

$\frac{uu^T}{u^Tu}$ and I - $\frac{uu^T}{u^Tu}$ are elementary projections.

The CGS method may be expressed in terms of elementary projections.

The proof is:

Let x_1,x_2...x_p , p$\geq$k, be vectors spanning R^k. The CGS generates p orthogonal vectors, h_1,h_2...h_p, spanning R^k, as follows:

$$h_1 = x_1$$

$$h_2 = x_2 - \frac{x_2^T h_1}{h_1^T h_1} h_1$$

$$h_p = x_p - \sum_{j=1}^{p-1} \frac{x_p^T h_j}{h_j^T h_j} h_j \tag{99}$$

The sum over terms such as $h_j^T h_j \neq 0$ appears above.

For any two vectors u and v, of the same dimension, we may write $(u^T \mathrm{v})\mathrm{v} = (v^T \mathrm{u})\mathrm{v}$ $= \mathrm{v}(v^T u) = (\mathrm{v}v^T)\mathrm{u}$. $\mathrm{v}v^T$ is an outer product.

In the CGS process, we let $u^T = x_i^T$, and $\mathrm{v} = \mathrm{h_j}$.

$\frac{x_i^T h_j}{h_j^T h_j} h_j$ may be replaced by $\frac{h_j h_j^T}{h_j^T h_j}$. For $h_j^T h_j \neq 0$,

$$h_i = x_i - \sum_{j=1}^{i-1} \frac{h_j h_j^T}{h_j^T h_j} x_i = (I - \sum_{j=1}^{i-1} \frac{h_j h_j^T}{h_j^T h_j}) x_i, i = 1...p. \tag{100}$$

$$(I - \sum_{j=1}^{i-1} \frac{h_j h_j^T}{h_j^T h_j}) \tag{101}$$

projects x_i on $\mathcal{L}^{\perp}(h_1, h_2 ... h_{i-1})$, where $\frac{h_j h_j^T}{h_j^T h_j}$, for j=1,...i-1, are elementary projections.

We use Lemma 1 later, to show that the CGS method is a special case of our more general method. A disadvantage of the CGS is it often displays a severe loss of orthogonality among the computed h_i. See, for instance,[11]. The elementary projections approach does much better in producing orthogonal h_i. The CGS error size is approximately the round-off-error times k_2^2(A). However, when using elementary projections, as in [11], the error size is just the round-off-error. The CGS's advantage over using projections is its greater computational efficiency. In choosing to use the projections approach, we emphasize accuracy over speed.

There are several equivalent definitions of the pseudoinverse. A definition of [6] is

A^+ is the pseudoinverse of A iff

$$A^+ = \lim_{\delta \to 0}(A^T A + \delta\ I)^{-1} A^T = \lim_{\delta \to 0} A^T (AA^T + \delta\ I)^{-1}, \delta\ > 0 \qquad (102)$$

The matrices $(A^T A + \delta\ I)^{-1}$ and $(AA^T + \delta\ I)^{-1}$ exist, even if the matrices $(A^T A)^{-1}$ and $(AA^T)^{-1}$ do not. The eigenvalues of $(A^T A + \delta\ I)^{-1}$ and $(AA^T + \delta\ I)^{-1}$ are $\geq\delta$ $>$0.

An equivalent definition, given in[6], is:

Matrix B is the pseudoinverse of matrix A iff A and B satisfy the four Moore-Penrose conditions :

(i) ABA=A (ii) BAB=B (iii) $(AB)^T$=AB (iv) $(BA)^T$=BA

Now, suppose $A^{m\times n}$ is a general matrix. Consider the equations Ax = b. A is an m×n matrix, and x εR^n and b εR^m are column vectors. The following cases may arise:

1. If n=m, rank(A)=n, A^{-1} exists, x =A^{-1} b, and so A^{+}= A^{-1}.

2. If rank(A) = m $\leq$n , there are more unknowns than equations. The system is underdetermined. It is easy to check that

$$x = A^T(AA^T)^{-1}b + (I - A^T(AA^T)^{-1})y \tag{103}$$

satisfies the equations, with y an arbitrary vector. Of course, $A^T(AA^T)^{-1}$b is a solution, with y=0, and A^{+}=$A^T(AA^T)^{-1}$.

3. If rank(A) = n$\leq$m, $(A^TA)^{-1}A^T$b is a solution to the equation, with A^{+}=$(A^TA)^{-1}A^T$.

4. If A is of full rank, but ill-conditioned, the LS solution is sensitive to small perturbations of matrix A and/or vector b.

5. If A is not of full rank, the system is overdetermined, and so $(AA^T)^{-1}$ and $(A^TA)^{-1}$ do not exist. A^{+} always exists, but is unstable.

We provide algorithms for solutions to cases 4 and 5. We also derive a new algorithm to solve cases 1, 2 and 3.

[1]and [11] show that that if rank(A) = n $\leq$ m, as in case 3, $A^{+} = A^T(AA^T)^{-1}$ satisfies the definition of the pseudoinverse of A. If rank(A) = n $\leq$ m, as in case 2, $A^{+} = (A^TA)^{-1}A^T$is the pseudoinverse of A. Our algorithm yields the same solutions to cases 1,2, and 3, as do other well-known methods.

[11]says that when rank(A)=n, as in case 4, the accuracy of the solution computed by the LS problem depends directly on the square of the condition of the matrix A, k_2(A) $= \frac{\sigma_{max}(A)}{\sigma_{min}(A)}, \sigma_{min}(A) \neq 0$.

[11]also says that if A is not of full rank, the formal Penrose solution is unstable. Our hypothesis is that the instability of the Penrose solution makes it unsuitable to many applications. We address this instability by providing a more stable solution for cases 4 and 5. We also produce a new algorithm, solving cases 1, 2, and 3.

Appendix B: MatLab Code

We discuss testing the abnormality of an anatomical structure with MatLab.

Lines 2 - 7 generate a Radon vector of a standard Shepp-Logan phantom. Lines 9 - 13 generate a Radon vector of the same phantom with ellipse 9 set to the gray scale density of air. In line 14, we take the difference of the two Radon vectors to subtract everything except ellipse 9 of the original Radon vector. Lines 18 - 42 provide a loop that generates an additional n=20 Radon vectors, as in line 14, where the parameters of the phantom were perturbed by random distortions to create a training set of vectors. Lines 44 - 63 provide our algorithm that computes the projection operator M, with the perturbation parameter set to cst = 0.00001. In lines 51 - 54, we show the approximate orthogonalization. In lines 55 - 63, we show the back projections. In line 66, we compute the projection operator that approximates M = XX^{+}. Lines 71 - 79 generate a Radon vector of ellipse 9 distorted more than in the 21 training images. In line 80, we apply our computed projection operator M to the new Radon vector in order to project it onto the subspace generated by the 21 training images. In line 81, we apply the orthogonal projection on the new Radon vector. In lines 82 and 83, we compute the probability that the new structure is normal and abnormal respectively. In lines 84 and 85, we display the computed values of lines 82 and 83.

```
% Computation of Projection Operator by the Approximate
  Pseudoinverse method.
clear; %above line converges to the solution x of  x
theta=0:10:170; %use 18 angles of view.
[p,E]=phantom(128); %generate phantom of 128 by 128 pixels.
[Rp,xp]=radon(p,theta); %generate the Radon transform for the
  phantom.
rp=Rp(:); %concatenate all the columns of Rb matrix into one column.
X(1,:)=rp'; %take the transpose to make it a row vector.
%the next 6 lines generate the phantom with hole at ellipse 9.
E(9,1)=-1; %set the 9th ellipse gray scale to black (air density).
ph=phantom(E,128); %generate a phantom with a black hole at ellipse
  9.
[Rh,xp]=radon(ph,theta); %generate the Radon transform.
rh=Rh(:); %concatenate all the columns of Rh into one column.
h=rh'; %Take transpose of rh.
D=X(1,:) - h ; %subtract out everything except ellipse 9.
% D=[D;D]; %add the same column to make it rank deficient.
%The following code generates the input vectors, that converge to
  the solution x of  x
%generate 15 variations of image pb with random gray scale.
for k=1:20, % loop 20 times.
    [p,E]=phantom(128);
    rn=random('unif',-.05,.05); %generate a random number.
```

```
E(9,1)=.1 ; %assign the generated gray scale.
rn=random('unif',-.001, .001);
E(9,2)= 0.0230 + rn; %change horizontal axis by rn.
rn=random('unif',-.06, .06);
E(9,3)= 0.0230 + rn; %change vertical axis by rn.
E(9,4)= 0.0 + rn; %change x coordinate for center of ellipse.
rn=random('unif',-.02, .02);
E(9,4)= -0.6060 + rn; %change y coordinate for center of ellipse.
E(9,6)= 30; %change the angle.
pv=phantom(E,128); %generate image with the new grayscale.
[Rv,xp]=radon(pv,theta); %generate a Radon transform of the new
  image.
rv=Rv(:); %make a vector out of the transfom matrix.
x=rv'; % take the transpose to make it a row vector.
X=[X;x]; %concatenate the vector.
E(9,1)=-1; %set the 9th ellipse gray scale to black (air density)
  .
ph=phantom(E,128); %generate a phantom with a black hole at
  ellipse 9.
[Rh,xp]=radon(ph,theta); %generate the Radon transform for the
  phantom.
rh=Rh(:); %concatenate all the columns of Rab into one column.
  converges to the solution x of  x
h=rh'; %Take the traspose or rh.
```

```
   d=x-h; % generate the radon vector for the 9th ellipse of the
     generated image.
   D=[D;d]; %concatenate the radon vectors of the difference radon.
end
%***********************  compute our Inv of D  ******************
cst=.00001; %set pertubation parameter.
[r,c]=size(D);  %get the number of rows and columns.
Y=eye(r);
[rr,cr]=size(Y);
Q(:,1)=D(:,1); %converges to the solution x of  x
I=eye(r);  %generate identity matrix of dim rxr.
sum=zeros([r,r]); %initialize sum with a rxc matrix with zeros.
for k=1:c-1,
   sum=sum + Q(:,k)*Q(:,k)'/(cst + D(:,k)'*Q(:,k));
   Q(:,k+1)= (I-sum)*D(:,k+1);
end
for j=1:r,
   S=zeros([r,1]);
   M(c,j)=((Q(:,c)'*Y(:,j)))/(cst + D(:,c)'*Q(:,c));
   for i=1:c-1,
      S=S + M(c-i+1,j)*D(:,c-i+1);
      dm= 1/(cst + D(:,c-i)'*Q(:,c-i));
      M(c-i,j)=(dm*Q(:,c-i)')*(Y(:,j) - S);
   end
%endconverges to the solution x of  x
```

```
%****End of Pseudoinverse Computation****
Inv=M; %get pseudoinverse of training matrix, 16 by 3330 matrix.
M=Inv*D;
[r,c]=size(M); %D is a 3330 by 16 matrix.
I=eye(c); %generate identity matrix 3330 by 3330.
Po=I-M; %P is a projection operator.
%generate a new image.
[pn,E]=phantom(128);
E(10,1)=.8; %set the gray sacale of ellipse 9.
[Rn,xp]=radon(pn,theta); %get its Radon transform that converges to
  the solution x of  x.
rn=Rn(:); %make a vector out of Rn.
E(10,1)=-1;
pnn=phantom(E,128);
[Rpn,xp]=radon(pnn,theta); %generate the Radon transform for the
  phantom.
rpn=Rpn(:);
rd=rpn-rn; %subtract from original image to isolate the Radon vector
   of object.
pd=M*rd; %project onto the subspace generated by the difference of
  Radon vectors.
pdn=Po*rd; %project rd onto the orthoganal subspace.
dif=norm(pd)^2/norm(rd)^2; %display the probability.
difi=norm(pdn)^2/norm(rd)^2;
disp(dif);
disp(difi); %converges to the solution x of  x
```

We discuss our code for computing $\widehat{P}x = (\sum_{i=1}^{k} q_i q_i^T)x$.

Lines 2 - 42 generate the training Radon vectors using the same code as in the above code. In lines 44 - 53, we orthogonalize in the same way as in lines 51 - 54 above. In lines 56 - 61, we normalize the orthogonalized vectors. In lines 64 - 72, we generate the Radon vector of a structure in a new image. In lines 74 - 78, we expand the new Radon vector in terms of the basis generated in lines 56 - 61, to obtain the projection $\widehat{P}x$ of the new Radon vector onto the normal subspace spanned by the input vectors of normal structure. In line 80, we compute the vector orthogonal to the projected vector. In lines 81 and 83, we compute the probability of the new structure being normal and abnormal, respectively. In lines 84 and 85, we display the two computed probabilities. In lines 86 and 87, we display the outputs, with probability of being normal as 0.0997, and probability of being abnormal as 0.9003. So the probability that the structure is normal is 0.09, and that it is abnormal is 0.9002.

```
1 %Represent the vector x in terms of a subspace basis vectors.
2 theta=0:10:170; %use 18 angles of view.
3 [p,E]=phantom(128); %generate phantom 128 by 128 pixels.
4 [Rp,xp]=radon(p,theta); %generate the Radon transform for the
    phantom.
5 rp=Rp(:); %concatenate all the columns of Rb matrix into one column.
6 X(1,:)=rp'; %take the transpose to make it a row vector.
7 %the next 6 lines generate the phantom with hole at ellipse 9.
8 E(9,1)=-1; %set the 9th ellipse gray scale to black (air density.)
```

```
ph=phantom(E,128); %generate a phantom with a black hole at ellipse
  9.
[Rh,xp]=radon(ph,theta); %generate the Radon transform for the
  phantom.
rh=Rh(:); %concatenate all the columns of Rab into one column.
h=rh'; %Take traspose of rh.
D=X(1,:) - h ; %subtract out everything except ellipse 9.
% D=[D;D]; %add the same column to make rank defficient.
%following code generates the input vectors.
% generate variatins of image pb with random gray scale.
for k=1:20, % loop 20 times.
    [p,E]=phantom(128);
    rn=random('unif',-.05,.05); %generate a random number.
    E(9,1)=.1 ; %assign the generated gray scale to ellipse 9.
    rn=random('unif',-.001, .001);
    E(9,2)= 0.0230 + rn; %change horizontal axis by rn.
    rn=random('unif',-.06, .06);
    E(9,3)= 0.0230 + rn; %change vertical axis by rn.
    E(9,4)= 0.0 + rn; %change x coordinate for center of ellipse.
      converges to the solution x of  x
    rn=random('unif',-.02, .02);
    E(9,4)= -0.6060 + rn; %change y coordinate for center of ellipse
      .
    E(9,6)= 30; %change the angle.
   pv=phantom(E,128); %generate image with the new gray scale of ellipse 9.
   [Rv,xp]=radon(pv,theta); %generate the Radon transform of the new image.
   rv=Rv(:); %make a vector out of the transfom matrix.
   x=rv'; %take the transpose to make it a row vector.
   X=[X;x]; %concatenate the vector.
   E(9,1)=-1; %set the 9th ellipse gray scale to black (air density.)
   ph=phantom(E,128); % generate a phantom with a black hole.
   [Rh,xp]=radon(ph,theta); %generate the Radon transform.
   rh=Rh(:); %concatenate all the columns of Rh into one column.
   h=rh'; %Take the traspose of rh.
   d=x-h; %generate the Radon vector for the 9th ellipse of the generated image.
   D=[D;d]; %concatenate the Radon vectors of the difference Radon vector.
end
D=D';
%************************ Orthogonalize  ******************
cst=.00001;
[r,c]=size(D);  %get the number of rows and columns of X.
Y=eye(r);
Q(:,1)=D(:,1);
I=eye(r);  %generate identity matrix of dim rxr.
```

```
sum=zeros([r,r]);  %initialize sum with an rxc matrix with zeros.
for k=1:c-1,
   sum=sum + Q(:,k)*Q(:,k)'/(cst + D(:,k)'*Q(:,k));
   Q(:,k+1)=(I-sum)*D(:,k+1);
end
%******************End of orthogonalize************************
%***************normalize***************
[r,c]=size(Q);
for j=1:c
   if (norm(Q(:,j))>.0001)
      Q(:,j)=(1/norm(Q(:,j)))*Q(:,j);
   end
end
%************end normalize**************

[pn,E]=phantom(128);%generate a new image.
E(10,1)=.8; %set the gray scale of ellipse 9.
[Rn,xp]=radon(pn,theta); %get Radon transform of the abovel line.
rn=Rn(:); %make a vector out of Rn.
E(10,1)=-1;
pnn=phantom(E,128);
[Rpn,xp]=radon(pnn,theta); %generate the Radon transform for the
  phantom.
rpn=Rpn(:);
```

```
72 rd=rpn-rn; %subtract from original image to isolate the Radon vector
      of object.
73 %*************expand rd in Qs***************
74 pd=zeros(r); Algorithm
75 pd=pd(:,1);
76 for i=1:c,
77   pd=pd + (Q(:,i)'*rd)*Q(:,i);
78 end
79 %**************expand**********************
80 pp=rd-pd;
81 dif=norm(pd)^2/norm(rd)^2; %display the probability.
82 %The closer to 1, the greater novelty.
83 difi=norm(pp)^2/norm(rd)^2;
84 disp(dif); Algorithm
85 disp(difi);
86 % result: probability of normal is 0.0997
87 % result: probability of abnormal is 0.9003
```

In lines 2 - 39, we generate the training vectors as we did in the two previous examples. In line 40, we invoke the singular value decomposition function, SVD(D), of the training matrix D, consisting of the vectors generated in lines 2 - 39.

The matrix U is orthogonal and has as many rows as D. The matrix V is orthogonal and has as many columns as D. The matrix S is the same size as D, and its nonzero elements lie on its main diagonal. The diagonal elements of S are the singular values, and the columns of U and V are the left and right singular vectors, respectively. In lines 43 - 45, we expand D as a sum of rank-1 matrices, that is, $D = \sum_{k=1}^{n} \sigma_k E_k$, where E_k= U(:,k)V^T(:,k), but only up to 20 terms instead of 50. In other words, we have dropped the last 30 terms in the rank-1 expansion of D. So we have an approximation of the original D, where the noisy terms with low singular values have been dropped. In line 46, we take the pseudoinverse of the approximation of D, which is more stable than the original D. In line 47, we obtain the projection operator $D^{+}{}_{truk}$ D_{truk} using the truncated D computed in lines 43 - 45. Lines 52 - 60 generate the Radon vector of the structure in a new image. In line 61, we project the new Radon vector using the computed projection matrix M = $D^{+}{}_{truk}$ D_{truk} onto the subspace spanned by the training vectors. In line 62, we project the new Radon vector on the orthogonal subspace. In lines 63 and 64, we compute the probability that the new Radon vector is normal and abnormal, respectively. In lines 65 and 66, we display the computed probabilities.

```
1 %Autoassociative OLAM as a Novelty Filter with SVD dimension
    reduction.
2 theta=0:10:170;              %use 18 angles of view.
3 [p,E]=phantom(128);          %generate phantom 128 by 128 pixels.
```

```
[Rp,xp]=radon(p,theta);    %generate the Radon transform for the
  phantom.
rp=Rp(:);                  %concatenate all the columns of Rb.
X(1,:)=rp';                %take the transpose to make it a row
  vector.
   %***the next 6 lines generate the phantom with hole at ellipse
     9***
E(9,1)=-1;                 %set the 9th ellipse gray scale to black (
  air density.)
ph=phantom(E,128);         %generate a phantom with a black hole.
[Rh,xp]=radon(ph,theta);   %generate the Radon transform.
rh=Rh(:);                  %concatenate all the columns of Rh into
  one column.
h=rh';                     %Take the traspose or rh.
D=X(1,:) - h ;             %Subtract out everything except ellipse 9.
%****following code generate the input vectors****
for k=1:50,               % Generate 15 variations of image pb.
       [p,E]=phantom(128);
       rn=random('unif',-.05,.05);    %generate a random number
         between -.1 and .1.
      E(9,1)=.1 + rn;       %assign the generated gray scale.
      rn=random('unif',-.001, .001);
      E(9,2)= 0.0230 + rn;         %change the horizontal axis by rn.
      rn=random('unif',-.06, .06);
      E(9,3)= 0.0230 + rn;         %change the vertical axis by rn.
      E(9,4)= 0.0 + rn;            %change the x coordinate for
        center of ellipse.
     rn=random('unif',-.02, .02);
     E(9,4)= -0.6060 + rn;        %change the y coordinate for center
        of ellipse.
     E(9,6)= 30;                  %change the angle.
    pv=phantom(E,128);            %generate an image with the new
      gray scale.
    [Rv,xp]=radon(pv,theta);      %generate a Radon transform of the
      new image.
     rv=Rv(:);                    %make a vector out of the transfom
       matrix.
     x=rv';                       %take the transpose to make it a
       row vector.
     X=[X;x];                     %concatenate the training matrix.
    E(9,1)=-1;                    %set the 9th ellipse gray scale to
      air density.
   ph=phantom(E,128);             %generate a phantom with a black
     hole at ellipse 9.
   [Rh,xp]=radon(ph,theta);       %generate the Radon transform for
     the above line.
    rh=Rh(:);                      %concatenate all the columns of
      Rab into one column.
    h=rh';                         %Take traspose or rh.
    d=x-h;                        %generate the Radon vector for the
       9th ellipse.
    D=[D;d];                      %concatenate the Radon vectors of
      the difference Radon vector.
end
[U,S,V]=svd(D);
[r,c]=size(D);
Do=zeros(r,c);
for i=1:20,
    Do=Do + S(i,i)*U(:,i)*V(:,i)';
end
Inv=pinv(Do);
M=Inv*Do;
[r,c]=size(M); %D is a 3330 by 16 matrix, where Inv. M is the
  transition matrix.
I=eye(c);      %generate a density matrix of size 3330 by 3330.
Po=I-M;        %P is a projection operator.
%generate a new image
[pn,E]=phantom(128);
E(10,1)=.8;     %set the gray scale of ellipse 9.
[Rn,xp]=radon(pn,theta);       %get its Radon transform.
rn=Rn(:);                      %make a vector out of Rn.
E(10,1)=-1;
pnn=phantom(E,128);
```

```
58 [Rpn,xp]=radon(pnn,theta);       %generate the Radon transform for
     the phantom.
59 rpn=Rpn(:);
60 rd=rpn-rn;                         %subtract from original image to
     isolate the radon of object.
61 pd=M*rd;                           %project onto subspace generated by
      Radon difference vectors.
62 pdn=Po*rd;                         %project rd onto the orthogonal
     subspace.
63 dif=norm(pd)^2/norm(rd)^2;         %It is 1 if x normal to L and 0 if
     x is in L.
64 difi=norm(pdn)^2/norm(rd)^2;
65 disp(dif);
66 disp(difi);
```

We discuss the MatLab code for identifying and displaying an anatomical structure.

In this algorithm, we have to compute a projection operator for each anatomical structure of interest. In lines 2 - 44, we generate the training Radon vectors the same way as in the above previous examples, except this time we save the Radon vectors of each of the image, as the matrix X, as well as the corresponding Radon vectors of the anatomical structure, in matrix D. In lines 47 - 65, we compute the transition operator that maps the image Radon vector to the anatomical structure Radon vector. In lines 48 -57, we orthogonalize the vectors of matrix X. In lines

58 - 65, we compute the approximate pseudoinverse of X. In line 68, we compute the approximate transition operator $X^{+}D$. In lines 70 - 72, we generate the Radon vector of a new image. In line 73, the transition operator maps the Radon vector of the new image into the corresponding Radon vector of the anatomical structure. In lines 77 - 80, we transform the Radon vector into a Radon matrix. In line 81, we take the inverse Radon transform of the mapped anatomical structure vector. In line 82, we display the image of the computed anatomical structure.

```
1  %Heteroassociative OLAM, M maps from image Radon vector to anatomical structure Radon vector.
2  %If we apply M on the Radon vector of a new image, we should get the correponding structure Radon vector.
3  %Take the inverse of this Radon vector and show image of structure.
4  %generate the standard phantom and get its Radon trasform vector.
5  theta=0:10:170; %use 18 angles of view.
6  [p,E]=phantom(128); %generate a phantom 128 by 128 pixels.
7  [Rp,xp]=radon(p,theta); %generate the Radon transform for the phantom.
8  rp=Rp(:); %concatenate all the columns of Rb matrix into one column.
9  X(1,:)=rp'; %take the transpose to make it a row vector.
10 %The next 6 lines generate the phantom with hole at ellipse 4.
11 E(4,1)=-1; %set the 4th ellipse gray scale to black (air density).
12 ph=phantom(E,128); %generate a phantom with a black hole at ellipse 4.
13 [Rh,xp]=radon(ph,theta); %generate the radon transform.
```

```
rh=Rh(:); %concatenate all the columns of Rh into one column.
h=rh'; %Take the transpose of rh.
D=X(1,:) - h ; %subtract out everything except ellipse 4.

%The following code generates the input vectors that train the
  associative memory.
for k=1:60, %generate 60 variations of image pb.
   [p,E]=phantom(128);
   rn=random('unif',-.01,.01); %generate a random number between -.1
      and .1.
   E(4,1)=-.2 + rn; %assign the generated gray scale to ellipse 4.
   rn=random('unif',-.001, .001);
   E(4,2)= 0.16 + rn; %change the horizontal axis by rn.
   rn=random('unif',-.001, .001);
   E(4,3)= 0.41 + rn; %change the vertical axis by rn.
   rn=random('unif',-.001, .001);
   E(4,4)= -0.22 + rn; %change the y coordinate for center of
     ellipse.
   E(4,5)= -0.0 + rn;
   rn=random('unif',-1, 1);
   E(4,6)= 18+rn; %change the angle.
   pv=phantom(E,128); %generate image with the new  gray scale of
     ellipse 4.
   [Rv,xp]=radon(pv,theta); %generate a Radon transform of the new
     image.
```

```
    rv=Rv(:); %make a vector out of transform matrix.
    x=rv'; %take transpose to make it a row vector.
    X=[X;x];%concatenate the vector.
    E(4,1)=-1; %set the 4th ellipse gray scale to black (air density)
      .
    ph=phantom(E,128); %generate a phantom with a black hole at
      ellipse 4.
    [Rh,xp]=radon(ph,theta); %generate the Radon transform.
    rh=Rh(:); %concatenate all the columns of Rab into one column.
    h=rh'; %Take transpose or rh.
    d=x-h; %generate the Radon vector for the 4th ellipse of the
      generated image.
    D=[D;d]; %concatenate the Radon vectors of the difference radon.
end
Y=D;
%**************** compute our solution MX=D *****
cst=.001; %set parameter
[r,c]=size(X); %get the number of rows and columns of X.
Y=eye(r);
[rr,cr]=size(D);
Q(:,1)=X(:,1);
I=eye(r); %generate an identity matrix of dim rxr.
sum=zeros([r,r]); %initialize sum with an rxc matrix with zeros.
%for k=1:c-1,converges to the solution x of  x
    sum=sum + Q(:,k)*Q(:,k)'/(cst + X(:,k)'*Q(:,k));
    Q(:,k+1)= (I-sum)*X(:,k+1);
end
for j=1:r,
    S=zeros([r,1]);
    M(c,j)=((Q(:,c)'*Y(:,j)))/(cst + X(:,c)'*Q(:,c));
    for i=1:c-1, S=S + M(c-i+1,j)*X(:,c-i+1);
        dm= 1/(cst + X(:,c-i)'*Q(:,c-i));
        M(c-i,j)=(dm*Q(:,c-i)')*(Y(:,j) - S);
    end
end
%***********End of Pseudoinverse Computation*****
Inv=M;
M=Inv*D;
%generate a new image.
[pn,E]=phantom(128);
[Rn,xp]=radon(pn,theta); %get its Radon transform.
rn=Rn(:); %make a vector out of Rn.
rx=rn'*M;
rx=rx';
%*****Convert a vector of size=Rr*Rc to a matrix of size Rr by Rc
  ****
[Rr,Rc]=size(Rn); %Rn is the Radon transform matrix and rd is a
  Radon vector.
R=rx(1:Rr);
for i=1:Rc-1,
    R=[R rx(i*Rr+1:(i+1)*Rr)];
end %*********end of convert*******
Im=iradon(R,theta);
imshow(Im);
```