COMBINING MACHINE LEARNING TECHNIQUES WITH STATISTICAL SHAPE MODELS IN MEDICAL IMAGE SEGMENTATION

BY
EUSTACE EBHOTEMHEN

A THESIS

SUBMITTED TO THE AFRICAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

ABUJA – NIGERIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF SCIENCE IN COMPUTER SCIENCE

SUPERVISORS:
Dr. KOLA BABALOLA

Prof. LEHEL CSATO

African University of Science and Technology
www.aust-abuja.org
P.M.B 681,Garki,Abuja F.C.T
Nigeria.

May, 2013

COMBINING MACHINE LEARNING TECHNIQUES WITH STATISTICAL SHAPE MODELS IN MEDICAL IMAGE SEGMENTATION

A THESIS APPROVED

BY

SUPERVISORS _____

Dr. KOLA BABALOLA / Prof. LEHEL CSATO

MEMBER _____

Prof. M.K. TRAORE

MEMBER _____

Dr. KOLA BABALOLA

ABSTRACT

In this thesis, we implemented Point Distribution Model and basic Active Shape Model algorithm and contributed this to the AUST Computer Vision and Machine Learning code library. We applied the Active Shape Model to segmenting lateral ventricles of 2D brain images and used machine learning – specifically K-Nearest Neighbour algorithm- to improve segmentation results. A statistical shape model is created from a training dataset which is used to search for an object of interest in an image. Active shape model has shown over time to be a reliable image segmentation methodology but its segmentation accuracy is hindered especially by poor initialization which can't be guaranteed to always be perfect. In our methodology, we extract features for each landmark using Haar filters. We train a classifier with these features and use the classifier to classify points around the final points of an Active shape model search. The aim of this approach is to better place points that might have been wrongly placed from the ASM search. We have used the simple, yet effective K-Nearest Neighbour machine learning algorithm, and have demonstrated the ability of this method to improve segmentation accuracy by segmenting lateral ventricles of the brain.

DEDICATION

I dedicate this thesis to God Almighty, to whom I owe my every breadth, and to my parents for their all-round support.

ACKNOWLEDGEMENTS

First, I want to thank God almighty for making this programme a huge success. For all he did for me, I couldn't ask for more.

I also want to thank my parents and my family members for standing by me in all of this.

To my supervisors, Dr. Kola Babalola, and Prof. Lehel Csato, I can't thank you enough for your guidance, encouragement and support throughout this thesis.

A big thank you to the President of AUST, Prof. Wole Soboyejo, for his support to students especially during our thesis.

To our one and only Prof. Mamadou Kaba Traore, the Head of Computer Science, I lack words to describe you, and I don't know enough vocabulary to correctly and appropriately express my gratitude for everything you are to the Computer Science stream. A thank you will not suffice, but I hope you understand my "predicament". Thank you Prof.

To all my friends, too numerous to mention, A very big thanks to you all.

TABLE OF CONTENT

CHAPTER SIX

LIST OF FIGURES

CHAPTER ONE

1.1 INTRODUCTION

In this thesis, our aim is to segment images, specifically, medical images. We aim to implement the popular Active Shape Model algorithm [16] and demonstrate its usefulness in segmenting 2d medical images. Furthermore, we explore improving the results of the Active Shape Model segmentation using machine learning techniques.

Predominantly, the aim of medical image segmentation is to label each pixel in an image to indicate the anatomical structure it belongs to and delineate such structures of interest for the purposes of visualization, diagnosis or medical research. Segmentation is often a crucial first step in patient diagnosis especially when qualitative and quantitative information about appearance, size, or shape of patient anatomy is desired. Results of medical image segmentation are useful for many purposes including image guided surgery, detection of anatomical changes over time, detection of pathological diseases, volumetric measurement, visualization and research. With the increasing importance of the segmentation process in diagnosis, accuracy of the process is important, as this may impact diagnostic accuracy, treatment planning and subsequently treatment.

The segmentation process is unfortunately as difficult as it is important, and the reasons are easy to comprehend. Computers are not half as good as humans when it comes to ill-defined problems such as object recognition, and when these images contain noise, it makes the process even more difficult for a computer. More often than not, images will be noisy, altering the intensity values of some pixels. This could make anatomical structures difficult to separate from their surroundings, and strong edges may not be present around its borders. Sometimes, the intensity

level of a single tissue class varies gradually over the image –a phenomenon known as intensity inhomogeneity or non-uniformity – and this doesn't make the segmentation task any easier. Other times, an individual pixel may contain mixture of tissue classes such that intensity of a pixel in the image may not be consistent with one class. The gray levels of different tissues, if too close would increase the difficulty of the process. These problems and the variability in the tissue distribution among individuals in the human population means that some degree of uncertainty must be attached to all segmentation results.

Segmentation can be done manually, semi-automatically or can be a fully automated process. Manual segmentation is a time consuming task and with the volume of medical image data needed to be processed, it is highly unlikely to be the ideal method considering the fact that results of such a process would depend on operator variability and thus would be difficult to reproduce. The level of confidence ascribed to manual processes suffers accordingly. Automatic methods overcome these drawbacks and are preferred especially with the computing resources available today and the amount of data needed to be processed. However, accurate automatic segmentation is by no means an easy feat and remains an active area of research in computer vision.

There are more segmentation methods than can be mentioned in this thesis, with abundant literature on most of them. Segmentation methods range from earlier intensity based approaches such as thresholding and region growing to pattern recognition approaches such as neural networks and model based approaches such as Active Shape and Appearance models. Thresholding approaches segment images by creating a binary partitioning of the image intensities. A typical thresholding approach attempts to determine an intensity value called the threshold which can separate the image into desired classes. This method of segmentation is

simple yet effective when the image that is being segmented contains structures with contrasting intensities. Region growing extracts an image region that is connected to a point called the seed point – usually manually selected – based on some predefined criteria which can be based on intensity and/or edge information [22].

Pattern recognition techniques seek to partition a feature space derived from an image by using data with known labels. These techniques are supervised methods since they require training data that are manually segmented and then used as references for segmenting new data. Clustering algorithms perform segmentation without training data and are termed unsupervised methods.

Deformable models are model based techniques for segmenting images by using closed parametric curves that deform under the influence of internal and external forces. To delineate an object boundary in an image, a closed curve must first be placed near the desired boundary and allowed to undergo an iterative "deformation" process. Deformable models include the Active contour model (snakes) and the Active Shape Model (Smart snakes). In this thesis however, our focus will be on Active Shape Model.

1.2 PROBLEM DEFINITION

In this thesis, we deal with the problem of medical image segmentation. The aim is to accurately delineate a particular structure of interest from an image for the purpose of diagnosis, or research. In this thesis we consider the use of statistical shape models for automatic medical image segmentation. Specifically, we aim to extend the popular Active Shape Model with machine learning techniques to improve the segmentation accuracy of medical images. At the end of this thesis, we aim to:

1. Build a repository of code (Matlab) for the Computer Vision and Machine Learning group at AUST to allow future AUST students and researchers alike build on what has been done previously. These sorts of code repository are available in many world class Universities.

2. Demonstrate the application of our implementation of ASM to 2D segmentation problems.

3. Explore improving results with machine learning techniques.

1.3 OUTLINE OF THESIS

The outline of this thesis is as follows: Chapter 2 discusses literature on medical image segmentation. Chapter 3 reviews Machine learning techniques and usage of machine learning in medical image segmentation. Chapter 4 presents the statistical shape model framework and chapter 5 describes the proposed algorithm for the extended ASM with experiments, results and discussions. In chapter 6, conclusions are drawn and future work outlined.

CHAPTER TWO

LITERATURE REVIEW

2.1 IMAGE SEGMENTATION

In this section, we review the basic concepts of segmentation especially in medical imaging and also a host of image segmentation methods. We discuss the earlier approaches using intensity based methods and current model based approaches.

2.1.1 Basic Concepts.

Segmentation is the process of partitioning images into constituent sub regions or delineating areas of interest in an image. With the volume of medical data needed to be processed it is simply inadvisable to use manual methods for segmentation since this can be time consuming and it may be difficult to reproduce results since those results are subject to operator variability.

With the amount of data and the increasing use of Computed topography (CT) and Magnetic resonance (MR) imaging for diagnosis, there is an increasing need to use computers to assist the process of medical image segmentation. This is necessary to achieve fast and accurate results and to ensure that a large number of cases are handled with the same precision and accuracy eliminating differences that would result from operator inconsistencies.

According to Sharma *et al*. [9], medical image segmentation aims to:

- Study anatomical structure
- Identify regions of Interest i.e. locate tumors, lesions and other abnormalities
- Measure tissue volume to measure growth of tumors (also decrease in size of tumor with treatment)

- Help in treatment planning prior to radiation therapy; in radiation dose calculation

Automatic segmentation of medical images is a difficult task as medical images are complex in nature and rarely have any simple linear feature. Further, the outputs of segmentation algorithms are affected by:

- partial volume effect.
- intensity inhomogeneity
- presence of artifacts
- closeness in gray level of different soft tissue

Medical image segmentation literature can be divided into three generations [3], with each generation representing a new level of algorithmic development. The earliest segmentation methods which were mostly based on intensities of different image regions and requiring little or no prior knowledge occupy the first generation. The second generation of segmentation methods used image models, optimization methods and uncertainty models while the third generation methods are capable of incorporating knowledge into image models for segmentation.

2.2 First Generation Segmentation Methods.

The first generation includes low-level techniques where little, if any, prior information is included. For example, the application of intensity thresholds, region growing, and heuristic edge tracing [3]. These methods often require a high degree of human interaction, thus making results probably irreproducible and susceptible to the human operator inconsistencies. In this sub section we discuss these first generation methods: Thresholding, Region growing and Edge tracing.

2.2.1 Thresholding

Thresholding is an intensity based approach for segmenting images by creating a binary partitioning of the image intensity. It is the simplest method for image segmentation where a certain intensity value known as the threshold is determined somehow and used to separate image pixels into desired classes. This segmentation procedure groups all procedures whose values are below this threshold into one class and those whose values are higher than the determined threshold into another class (see Figure 2.1 and Figure 2.2).

$$g(x) = \begin{cases} 1 & \text{if } x_{i,j} > T \\ 0 & \text{if } x_{ij} \leq T \end{cases} \quad (2.1)$$

Thresholding is a very simple, yet effective means of image segmentation in images where structures have very contrasting intensities. A huge disadvantage of this method is that it can only classify into two classes and cannot segment multi-channel images.

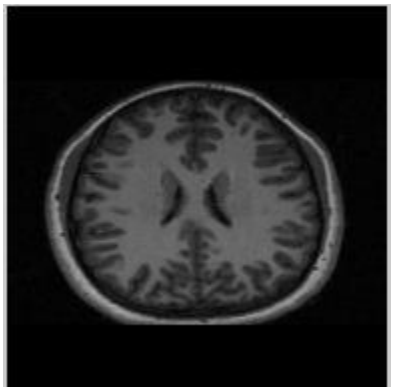


Figure 2.1:Original Image

Figure 2.2:Threholded Image (T = 75).

Thresholding methods are largely grouped as global or local. Global thresholding techniques uses a “global” threshold T to segment the entire image. Local thresholding techniques divides the image into different parts segmenting individual parts with a different threshold value.

2.2.2 Region Growing

Region growing is an approach to image segmentation that starts from some pixel called seed (see Figure 2.3) and grows throughout the desired region based on some homogeneity criteria. We need a rule describing a growth mechanism and a rule checking the homogeneity of the regions after each growth step

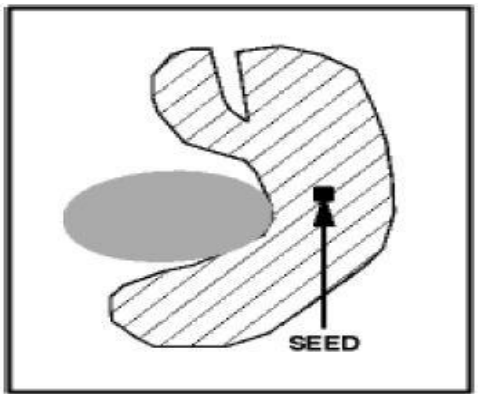


Figure 2.3:Region Growing

The region growing technique is one of the simplest region-based segmentation methods. It performs a segmentation of an image by examining the neighboring pixels of the seed points, and determines whether the pixels could be classified to the cluster of seed point or not. The algorithm procedure is as follows.

Step1. We start with a number of seed points which have been clustered into n clusters, called $C_1, C_2, \dots, C_n$, and the positions of initial seed points is set as $P_1, P_2, \dots, P_n$.

Step2. Compute the difference of pixel value of the initial seed point P_i and its neighboring points, if the difference is smaller than the threshold (criterion) we define, the neighboring point could be classified into C_i , where i = 1, 2, …,n.

Step3. Recompute the boundary of $C_i$and set those boundary points as new seed points P_i (s). In addition, the mean pixel values of $C_i$have to be recomputed, respectively.

Step4. Repeat Step2 and 3 until all pixels in image have been allocated to a suitable cluster.

The threshold is user-defined and it usually based on intensity, gray level, or color values. The regions are chosen to be as uniform as possible. There is no doubt that each of the segmentation regions of Region Growing has high color similarity and no fragmentary problem. However, it still has two drawbacks, seed-points and time-consuming problems

2.2.3 Edge tracing

Edge detection is more common for detecting discontinuities in gray level than detecting isolated points and thin lines because isolated points and thin lines do not occur frequently in most practical images. The edge is the boundary between two regions with relatively distinct gray level properties. It is assumed here that the transition between two regions can be determined on the basis of gray level discontinuities alone. Edge detection forms an edge image after which edge pixels which are adjacent neighbors are followed sequentially and collected into a list to represent an object boundary. The search for neighbouring pixels is heuristic in nature.

2.2.4 Split and Merge

Splitting and merging attempts to divide an image into uniform regions. Usually the algorithm starts from the initial assumption that the entire image is a single region, then computes the homogeneity criterion to see if it is TRUE. If FALSE, then the square region is split into the four smaller regions. This process is then repeated on each of the sub-regions until no further splitting is necessary. These small square regions are then merged if they are similar to give larger irregular regions. The problem (at least from a programming point of view) is that any two regions may be merged if adjacent and if the larger region satisfies the homogeneity criteria, but regions which are adjacent in image space may have different parents or be at different levels (i.e. different in size) in the pyramidal structure. The process terminates when no further merges are possible.

2.3 Second Generation Segmentation Methods

In the second generation of algorithms, efforts are made to overcome the problems posed by the first generation algorithms including their over reliance on human operation. The second generation algorithms introduce uncertainty models and optimization methods as well as attempting to avoid heuristics.

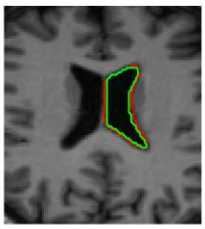
2.3.1 K-means Clustering

K-means clustering algorithm – so called because it finds K unique clusters, and the center of each cluster is the mean of the values in that cluster – is an unsupervised learning algorithm that attempts to classify or group objects into K number of groups based on the similarity or dissimilarity of these objects. The notion of similarity is dependent on a similarity measurement.

K-means require the user to specify the number of clusters K and the distance metric. The most critical choice is K. Although there is no mathematical criterion for selecting K, there are a number of heuristics for choosing K. Typically, k-means is run independently for different values of k and the partition that appears the most meaningful to domain experts is selected. The performance of k-means also depends on the initial positions of the cluster centers (centroids). K-means can be sensitive to noise and intensity inhomogeneity

k-means Algorithm:

- Create k points for starting centroids (often randomly)
- For every point in our image:
- For every centroid:
    - Calculate the distance from centroid to point
    - Assign the point to the cluster with the lowest distance
- For every cluster calculate the mean points in the cluster
- Repeat until little or no change occurs



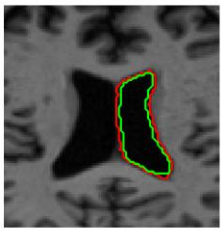


FIGURE 2.4: Results of K-means clustering of the Brain's lateral ventricles

2.3.2 Fuzzy C-means Clustering

Fuzzy c-means is a method of clustering which allows a data point to belong to more than one cluster albeit to varying degrees. The algorithm assigns these memberships on the basis of distance between the cluster and data point. The more the data is near the cluster center, the more its membership towards the particular cluster center. Clearly, summation of membership of each data point is equal to one.

Fuzzy c-means Algorithm:

Let X be the set of data points and V the set of centers.

- Randomly select 'c' clusters.
- Calculate the fuzzy membership 'μ_{ij}' using

$$\mu_{ij} = 1/\ \sum_{k=1}^{c}(d_{ij}/d_{ik})^{(\frac{2}{m}-1)} \quad (2.2)$$

- Compute fuzzy centers 'v_j' using

$$v_{ij} = \frac{\sum_{i=1}^{n}(\mu_{ij})^{m}\, x_i}{\sum_{i=1}^{n}(\mu_{ij})^{m}}, \forall_j = 1,2,\ldots.,c \quad (2.3)$$

- Repeat step 2 and 3 until the minimum J is achieved or $||U^{(k+1)} - U^{(k)}|| < \beta$.

Where:

'n' is the number of data points.

'v_j' represents the jth cluster center.

'm' is the fuzziness index m $[1, \infty]$.

'c' represents the number of cluster center.

'μ_{ij}' represents the membership of ith data to jth cluster center.

'd_{ij}' represents the Euclidean distance between ith data and jth cluster center.

The main objective of fuzzy c-means algorithm is to minimize:

$$J(U,V) = \sum_{i=1}^{n}\sum_{j=1}^{c}(\mu_{ij})^{m}\left\|x_i - v_j\right\|^2 \quad (2.4)$$

2.3.3 ACTIVE CONTOUR MODEL (SNAKES)

According to Kass *et al*. in [23], a snake is an energy-minimizing spline guided by external constraint forces and influenced by image forces that pull it toward features such as lines and edges. Snakes are active contour models: they lock onto nearby edges, localizing them accurately. Snakes have been used successfully in image interpretation, where a user imposes constraints that guide the snake to features of interest. A Snake exhibits dynamic behavior by continually minimizing its energy function. Snakes rely on some other mechanisms to place them near the desired contour. However, even in cases where no satisfactory automatic starting mechanism exists, snakes can still be used for semiautomatic image interpretation. If an expert user pushes a snake close to an intended contour, its energy minimization will carry it the rest of the way. The minimization provides a 'power assist' for a person pointing to a contour feature.

Snakes are an example of a more general technique of matching a deformable model to an image by means of energy minimization.

Representing the position of a snake parametrically by $v(s) = \big(x(s), y(s)\big)$, we can write its energy function as

$$E^*_{snake} = \int_0^1 E_{snake}(v(s))ds$$

$$= \int_0^1 E_{int}(v(s)) + E_{image}(v(s)) + E_{con}(v(s))ds \quad (2.5)$$

Where E_{int} represent the internal energy of the spline due to bending, E_{image} gives rise to the image forces, and E_{con} gives rise to external constraints forces.

The internal energy can be written as

$$E_{int} = (\alpha(s)|v_s(s)|^2 + \beta(s)|v_{ss}(s)|^2)/2 \quad (2.6)$$

The spline energy is composed of a first order term controlled by $\alpha(s)$ and a second order term controlled by $\beta(s)$.

The total image energy can be expressed as a weighted combination of the three energy Functionals which attract a snake to lines edges and terminations.

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term} \quad (2.7)$$

By adjusting the weights, a wide range of snake behavior can be created.

By combining E_{term} and E_{edge} we can create a snake that is attracted to edges or terminations. Snakes are autonomous and self-adapting in their search for a minimal energy state and they can be easily manipulated using external image forces. But they can often get stuck in local minima states and often overlook minute features in the process of minimizing the energy over the entire path of their contours.

2.4 Third Generation Segmentation Methods

The second generation of segmentation methods which introduced uncertainty models and optimization techniques are still very useful in practice but can hardly segment images

automatically and accurately enough. The third generation of segmentation methods incorporated higher level knowledge such as shape model of the desired object, appearance model (shape + texture model) of the desired object etc. In this subsection we discuss segmentation methods that fall in this category.

2.4.1 Active Shape Models

The Active Shape model was inspired by Active contour models, with the added intention of limiting the extent to which the contour model (snake) deformed. This method uses a statistical shape model to locate an object of interest in a complex image. But in order to locate such an object of interest, a model – the Point Distribution model - must first be built. To build this model, we require a set of such images. We must then introduce suitable points called landmark points which best describe the target shape and can be found on every training image. Good choices for such points are points at clear corners of object boundaries, 'T' junctions between boundaries or easily located biological landmarks which when not enough (as is always the case) are augmented with equally spaced points along the boundary [18]. The landmarks for a particular example are used to form a shape vector. After marking the landmark points, those points are aligned since the shape of an object is considered independent of its position, orientation and scale. The aim of the alignment is to translate, rotate and scale each shape so that the sum of distances of each shape to the mean ($D = \sum |x_i - \bar{x}|^2$) is minimized.

After alignment, a statistical model of these shapes is created from which new examples can be created and decide whether these new shapes are plausible. The first step in creating this shape model is to reduce the dimensions using Principal Components Analysis (PCA). After applying PCA to the data, we can now approximate any of the training set x using

$$\mathbf{x} \approx \bar{\mathbf{x}} + \boldsymbol{Pb} \tag{2.8}$$

Where $\mathbf{P} = (p_1|p_2|\ldots..|p_t)$ contains t eigenvectors of the covariance matrix and p is a t dimensional vector given by

$$\boldsymbol{b} = \boldsymbol{P}^T(\mathbf{x} - \bar{\mathbf{x}}) \tag{2.9}$$

By varying the elements of b, we can vary the shape x. This statistical shape model can now be used to search for objects of interests in various images using an iterative scheme shown below:

- Initialize the shape parameters, b, to zero
- REPEAT UNTIL CONVERGED
    - Generate the model instance $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Pb}$
    - Find the pose parameters (X_t, Y_t, s, θ) which best maps **x** to the new points Y.
    - Invert the pose parameters and use to project Y into the model co-ordinate frame, $y = T^{-1}_{X_t,Y_t,s,\theta}(Y)$.
    - Project y into the tangent plane to $\bar{\mathbf{x}}$ by scaling 1/(y. $\bar{\mathbf{x}}$**).**
    - Update model parameters to match y,$\mathbf{b} = \mathbf{P}^T(y - \bar{\mathbf{x}})$
    - Apply constraints on **b**.

2.4.2 Active Appearance Models

Active Appearance Models [16] are generated by combining a model of shape variation with a model of appearance variation. Just like the ASM, we require a set of labeled images where appropriate landmark points are labeled on each example image.

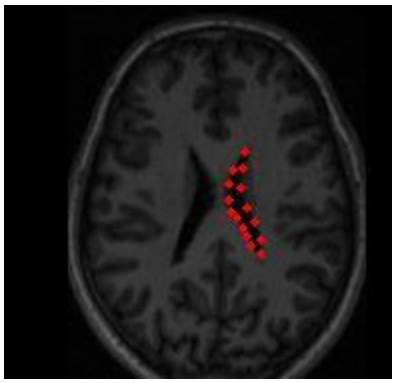


FIGURE 2.5: Lateral ventricle of the brain labeled with 15 points

Given a set of such points, we can generate a statistical shape model as described in the ASM section.

To build a statistical model of grey level appearance, we warp each example image so that each control point matches the mean shape. We then sample the grey level information from the shape normalized image over the region covered by the mean shape. To minimize the effect of global lighting variation, the example samples are normalized by applying a scaling, $\alpha, and\ offset, \beta,$

$$\mathbf{g} = (\mathbf{g}_{im} - \beta 1)/\alpha \qquad (2.10)$$

The values of α and β are chosen to best match the vector to the normalized mean. Let $\bar{g}$ be the mean of the normalized data, scaled and offset, so that the sum of the elements is zero and the variance of the elements is unity. The values of α and β required to normalize g_{im} are given by

$$\alpha = \mathbf{g}_{im}.\bar{\mathbf{g}}, \qquad \beta = (\mathbf{g}_{\mathrm{im}}.\mathbf{1})/n \qquad (2.11)$$

Where n is the number of elements in the vector.

By applying PCA to the normalized data, we obtain a linear model:

$$\mathbf{g} = \bar{\mathbf{g}} + \boldsymbol{P}_{\mathrm{g}}\boldsymbol{b}_{\mathrm{g}} \tag{2.12}$$

Where $\bar{\boldsymbol{g}}$ is the mean of the mean normalized gray level vector and $\boldsymbol{P}_{\mathrm{g}}$ is the set of orthogonal modes of variation and $\boldsymbol{b}_g$ is the set of gray level parameters.

The shape and appearance of any example can be summarized by the vectors $\boldsymbol{b}_s$ and $\boldsymbol{b}_{\mathrm{g}}$. Since there are many correlations between shape and grey level variations, we apply a further PCA to the data. For each example we generate the concatenated vector

$$b = \begin{pmatrix} \boldsymbol{W}_s\boldsymbol{b}_s \\ \boldsymbol{b}_{\mathrm{g}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{W}_s\boldsymbol{P}_s^T(\mathbf{x} - \bar{\mathbf{x}}) \\ \boldsymbol{P}_g^T(\mathbf{g} - \bar{\mathbf{g}}) \end{pmatrix} \tag{2.13}$$

Where $\boldsymbol{W}_s$ is a diagonal matrix of weights for each shape parameter allowing the difference in units between the shape and grey models. We then apply PCA on these vectors to get a further model

$$\boldsymbol{b} = \boldsymbol{Q}\boldsymbol{c} \tag{2.14}$$

Where **Q** are the Eigen vectors and **c** is a vector of appearance parameters controlling both shape and grey levels of the model. We can now express the shape and grey levels directly as functions of c.

$$\mathbf{x} = \bar{\mathbf{x}} + \boldsymbol{P}_s\boldsymbol{W}_s\boldsymbol{Q}_s\boldsymbol{c}, \mathbf{g} = \bar{\mathbf{g}} + \boldsymbol{P}_{\mathrm{g}}\boldsymbol{Q}_{\mathrm{g}}\boldsymbol{c} \tag{2.15}$$

Where

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{Q}_s \\ \boldsymbol{Q}_{\mathrm{g}} \end{pmatrix} \tag{2.16}$$

An example image can then be synthesized for a given **c** by generating the shape-free grey level image from the vector g and warping it using the control points described by **x**. We can now construct an iterative method for solving our optimization problem and fit an appearance model to a region of interest in a new image thus:

Given the current estimate of model parameters, $\mathbf{c}_0$, and the normalized image sample at the current estimate, $\mathbf{g}_s$, one step of the iterative procedure is as follows:

- Evaluate the error vector $\delta\mathbf{g}_0 = \mathbf{g}_s - \mathbf{g}_m$
- Evaluate the current error $E_0 = |\delta\mathbf{g}_0|^2$
- Compute the predicted displacement, $\delta\mathbf{c} = \mathbf{A}\delta\mathbf{g}_0$
- Set $k = 1$
- Let $\mathbf{c}_1 = \mathbf{c}_0 - k\,\delta\mathbf{c}$
- Sample the image at this new prediction, and calculate a new error vector, $\delta\mathbf{g}_1$
- If $|\delta\mathbf{g}_1|^2 < E_0$ then accept the new estimate, $\mathbf{c}_1$,
- Otherwise try at $k = 1.5$, $k = 0.5$, $k = 0.25$, etc.

This procedure is repeated until no improvement is made to the error, $|\delta\mathbf{g}|^2$, and convergence is declared.

2.4.3 Atlas based segmentation

There are many applications where there is no well-defined relationship between an image pixel's value(s) and the label that should be assigned to it. This is usually the case when we seek to label anatomical structures rather than tissues. For example, that different structures composed of the same tissue (e.g., different bones) cannot be distinguished from one another by looking at their intensity values in an image. What distinguishes these structures instead is their location

and their spatial relationship to other structures. In such cases, spatial information (e.g., neighborhood relationships) needs to be taken into consideration and included in the segmentation process. An atlas incorporates the locations and shapes of anatomical structures, and the spatial relationships between them. An atlas can, for example, be generated by manually segmenting a selected image. It can also be obtained by integrating information from multiple segmented images, for example from different individuals.

Given an atlas, an image can be segmented by mapping its coordinate space to that of the atlas in an anatomically correct way, a process commonly referred to as registration. Labeling an image by mapping it to an atlas is consequently known as atlas-based segmentation, or registration-based segmentation. The idea is that, given an accurate coordinate mapping from the image to the atlas, the label for each image pixel can be determined by looking up the structure at the corresponding location in the atlas under that mapping. Obviously, computing the coordinate mapping between the image and atlas is the critical step in any such method.

In atlas-based segmentation, an expert manually segments several images. These manually segmented images can be used to segment new images by extrapolating from the labeled images using image registration techniques.

CHAPTER THREE

3.1 MACHINE LEARNING

A machine learns if it becomes better at a Task *T* with some Experience *E*. There are many applications for Machine Learning, the most significant of which is data mining. People are often prone to making mistakes during analyses or, possibly, when trying to establish relationships between multiple features. This makes it difficult for them to find solutions to certain problems. Machine learning can often be successfully applied to these problems, improving the efficiency of systems and the designs of machines. Every instance in any dataset used by machine learning algorithms is represented using the same set of features. The features may be continuous, categorical or binary. If instances are given with known labels (the corresponding correct outputs) then the learning is called supervised, in contrast to unsupervised learning, where instances are unlabeled. By applying these unsupervised (clustering) algorithms, researchers hope to discover unknown, but useful, classes of items. Another kind of machine learning is reinforcement learning. The training information provided to the learning system by the environment (external trainer) is in the form of a scalar reinforcement signal that constitutes a measure of how well the system operates. The learner is not told which actions to take, but rather must discover which actions yield the best reward, by trying each action in turn.

As computer technology continues to advance, we possess an increasing capacity to store and process large amounts of data and to access it from physically distant locations. This generation is witnessing digital information explosion as the internet provides a seemingly endless amount of data that is constantly growing. This poses tremendous challenges regarding the intelligent organization, semantic search, and deep analysis of the data. This concerns not just the data and knowledge on the Web, but also in databases, social media, digital libraries, and scientific data

repositories. The science of learning plays a key role in the fields of statistics, data mining and artificial intelligence, computer vision, intersecting with areas of engineering and other disciplines. With machine learning we can gain insight from a dataset; we are going to ask the computer to make some sense from data.

There are three main classes of machine learning algorithms:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

3.1.1 Supervised learning

The defining characteristic of supervised learning is the availability of annotated or labeled training data. The name invokes the idea of a 'supervisor' that instructs the learning system on the labels to associate with training examples. Typically these labels are class labels in classification problems. Supervised methods are methods that attempt to discover the relationship between input attributes (sometimes called independent variables) and a target attribute (sometimes referred to as a dependent variable). The relationship discovered is represented in a structure referred to as a *model*. Supervised learning entails learning a mapping between a set of *input* variables *X* and an *output* variable *Y* and applying this mapping to predict the outputs for unseen data.

It is useful to distinguish between two main supervised models: *classification models* (*classifiers)* and *Regression Models*. Regression models map the input space into a real-value domain. For instance, a regressor can predict the demand for a certain product given its characteristics. On the other hand, classifiers map the input space into pre-defined classes. For

instance, classifiers can be used to classify mortgage consumers as good (fully payback the mortgage on time) and bad (delayed payback). There are many alternatives for representing classifiers, for example, support vector machines, decision trees, probabilistic summaries, algebraic function, etc.

Examples of supervised machine learning algorithms include K-Nearest Neighbour, Support-Vector Machines, Neural Networks, Decision Trees, Random Forest, Naïve Bayes etc.

3.1.2 Unsupervised learning

In contrast to supervised learning algorithms, the dataset provided for unsupervised learning are not annotated. Hence the task of such algorithms is to find hidden structures in unlabeled data. Data points in these dataset are grouped into a number of clusters or categories such that data points in the same category are more similar than data points in different categories.

Approaches to unsupervised learning include: clustering (e.g., k-means, mixture models, hierarchical clustering), blind signal separation using feature extraction techniques for dimensionality reduction (e.g., Principal component analysis, Independent component analysis, Non-negative matrix factorization, Singular value decomposition).

Among neural network models, the self-organizing map (SOM) and adaptive resonance theory (ART) are commonly used unsupervised learning algorithms. The SOM is a topographic organization in which nearby locations in the map represent inputs with similar properties. The ART model allows the number of clusters to vary with problem size and lets the user control the degree of similarity between members of the same clusters by means of a user-defined constant called the vigilance parameter.

3.1.3 Reinforcement learning

Reinforcement Learning allows the machine or software agent to learn its behaviour based on feedback from the environment. This behaviour can be learnt once and for all, or keep on adapting as time goes by. If the problem is modeled with care, some Reinforcement Learning algorithms can converge to the global optimum; this is the ideal behaviour that maximizes the reward.

This automated learning scheme implies that there is little need for a human expert who knows about the domain of application. Much less time will be spent designing a solution, since there is no need for hand-crafting complex sets of rules as with Expert Systems, and all that is required is someone familiar with Reinforcement Learning. There are many challenges in current Reinforcement Learning research. Firstly, it is often too memory expensive to store values of each state, since the problems can be pretty complex. Solving this involves looking into value approximation techniques, such as Decision Trees or Neural Networks. There are many consequence of introducing these imperfect value estimations, and research tries to minimize their impact on the quality of the solution.

3.2 Machine Learning Techniques

In this section, we describe some machine learning algorithms, including supervised and unsupervised techniques.

3.2.1 K-Nearest Neighbour Algorithm

KNN is a non-parametric (does not make assumptions about the distribution of the data) learning algorithm. It does not use training data points for generalization, rather on receipt of a new data

instance, it computes it K nearest neighbours from the dataset and makes a decision based on those. Characteristics of KNN include:

- Memory-based, no explicit training or model, "lazy learning".
- In its basic form one of the most simple machine learning methods
- Gives the maximum likelihood estimation of the class posterior probabilities
- Can be used as a baseline method

To perform a KNN classification, we represent our dataset as an MxN matrix with M training examples with each example consisting of N features. A vector O of length M representing output values for each training example.

Given a query input q, KNN can classify q with the following steps.

- For each instance in the dataset
    - Compute the distance (say Euclidean) between q and datapoint in dataset
- Select K closest neighbours
- Calculate majority.

The selected closest neighbours simply cast a vote each. The query data q is said to belong to the class with the most votes.

3.2.2 Naïve Bayes Classification

A Naive Bayes classifier assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 4 inches in diameter. A naive Bayes

classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of the presence or absence of the other features. Bayes theorem provides a way of calculating the posterior probability, P(C|x), from P(C), P(x), and P(x|C). Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.

$$P(C|x) = \frac{P(x|C)P(C)}{P(x)} \quad (3.1)$$

$$P(C|X) = P(x_1|C) \times P(x_2|C) \times \dots\dots \times P(x_n|C) \times P(C) \quad (3.2)$$

- $P(C|X)$ is the posterior probability of the class target given the predictor attribute.
- P(C) is the prior probability of class
- $P(x|C)$ is the likelihood which is the probability of the predictor given class
- $P(x)$ is the prior probability of the predictor.

3.3.3 Decision Trees

Decision Tree learning is a supervised learning algorithm whose goal is to create a model that predicts the value of a target variable based on several input variables. Given a training data, we can induce a decision tree. From a decision tree we can easily create rules about the data. Using decision tree, we can easily predict the classification of unseen records. There are several most popular decision tree algorithms such as ID3, C4.5 and CART (classification and regression trees).

A Decision tree works by continuously finding the best attribute which can be used to split the entire dataset based on the information gain of those attributes. A measure of impurity in the dataset (Entropy) is used to compute the information gain.

Entropy = $\sum_j(-p_j log_2 p_j)$ (3.3)
where p_j is the probability mass function of the j^{th} outcome.

The measure to compare the difference of impurity degrees is called information gain. We would like to know what our gain is if we split the data table based on some attribute values. Information gain is computed as impurity degrees of the parent table and weighted summation of impurity degrees of the subset table. The weight is based on the number of records for each attribute values.

3.3.4 Bagging, Boosting and Random Forests

Classification trees are adaptive and robust, but do not generalize well. The techniques discussed here enhance their performance considerably. Ensemble methods are usually divided into bagging & boosting, depending on whether the models are trained independently (bagging) or if previous models are somehow allowed to influence the training of subsequent models (boosting), usually by making the new model compensate for errors of previous models. Random forests are a kind of bagged tree. The trees are trained on (usually bootstrap) samples of the training data, but at each split in a tree only a random subset of the available features are considered for splitting on.

Training a Bagged Classifier:

Given a dataset S, at each iteration i, a training set Si is sampled with replacement from S (i.e. bootstraping)

A classifier Ci is learned for each Si

Classification: given an unseen sample X,

Each classifier Ci returns its class prediction

The bagged classifier H counts the votes and assigns the class with the most votes to X

The basic idea behind ensemble classifiers is to build different "experts" and let them vote. This greatly improves predictive performance as other types of classifiers can be included. Bagging works because it reduces the variance by voting or averaging and it is especially useful if the data is noisy.

Boosting is a general method of improving the prediction accuracy of a learning algorithm usually known as a "base" or "weak" learner. A weak learner is one whose prediction accuracy is just slightly better than random guessing. Boosting iteratively calls a given weak learner in a series of rounds $t = 1 \ldots\ldots T$. One of the main ideas of the algorithm is to maintain a distribution or set of weights over the training set. The weight of this distribution on training example i during round t is denoted $D_t(i)$ Initially, all weights are set equally, but on each round, the weights of incorrectly classified examples are increased so that the weak learner is forced to focus on the hard examples in the training set. The job of the weak learner is to find a weak hypothesis appropriate for the distribution D_t.

A typical Boosting Algorithm (AdaBoost)[14] is given below:

Given: $(x_1, y_1), \ldots\ldots, (x_n, y_n)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$

For $t = 1 \ldots\ldots T$

- Train weak learner using Distribution D_t
- Get weak hypothesis $h_t: X \to \{-1, +1\}$with error $\epsilon_t = Pr_{i\sim D_t}[h_t(x_i) \neq y_i]$

- Choose $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$
- Update:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

$$= \frac{D_t(i)\exp(-\alpha_i y_i h_t(x_i))}{Z_t}$$

Where Z_t is a normalization factor chosen so that D_{t+1} will be a distribution.

Our final output hypothesis is

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$$

A **Random Forest** is a combination of tree predictors where each tree depends on the values of a random vector sampled independently. The generalization error depends on the strength of the individual trees and the correlation between them. Random Forest was proposed by Breiman in 2001, adding an additional layer of randomness to bagging. In addition to constructing each tree using a different bootstrap sample of the data, random forests change how the classification or regression trees are constructed. In standard trees, each node is split using the best split among all variables. In a random forest, each node is split using the best among a subset of predictors randomly chosen at that node. This somewhat counterintuitive strategy turns out to perform very well compared to many other classifiers, including discriminant analysis, support vector machines and neural networks, and is robust against overfitting.

The random forests algorithm (for both classification and regression) is as follows:

- Draw n bootstrap samples from the original data.
- For each of the bootstrap samples, grow an unpruned classification or regression tree, with the following modification: at each node, rather than choosing the best split among

all predictors, randomly sample m of the predictors and choose the best split from among those variables. (Bagging can be thought of as the special case of random forests obtained when m = p, the number of predictors.)

- Predict new data by aggregating the predictions of the n trees (i.e., majority votes for classification, average for regression).

3.3.5 CLUSTERING

Clustering is an unsupervised learning technique which aims to separate data into various categories or "clusters" such that data points are closer or more similar to data points in the same cluster than they are to data points in other clusters. Among clustering formulations that are based on minimizing a formal objective function, perhaps the most widely used and studied is k-means clustering. Given a set of n data points in real $d - dimensional$ space, R^d, and an integer k, the problem is to determine a set of k points in R^d, called centers, so as to minimize the mean squared distance from each data point to its nearest center.

The K-means algorithm is a method to automatically cluster similar data examples together. Concretely, you are given a training set$\{x^{(1)}, \dots, x^{(m)}\}$ where $(x^{(i)} \in \mathbb{R}^n)$, and want to group the data into a few cohesive "clusters". The intuition behind K-means is an iterative procedure that starts by guessing the initial centroids, and then refines this guess by repeatedly assigning examples to their closest centroids and then re-computing the centroids based on the assignments. The algorithm consists of a simple re-estimation procedure as follows. Initially, the data points are assigned at random to the sets. For step 1, the centroid is computed for each set. In step 2, every point is assigned to the cluster whose centroid is closest to that point. These two

steps are alternated until a stopping criterion is met, i.e., when there is no further change in the assignment of the data points.

3.3.6 NEURAL NETWORKS

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest.

3.3.7 SUPPORT VECTOR MACHINES

A support vector machine (SVM) is a supervised learning algorithm that learns to assign labels to objects from examples. It is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. The question is; In which case is the hyperplane considered optimal? Consider a linearly separable set of 2D-points which belong to one of two classes as shown below:

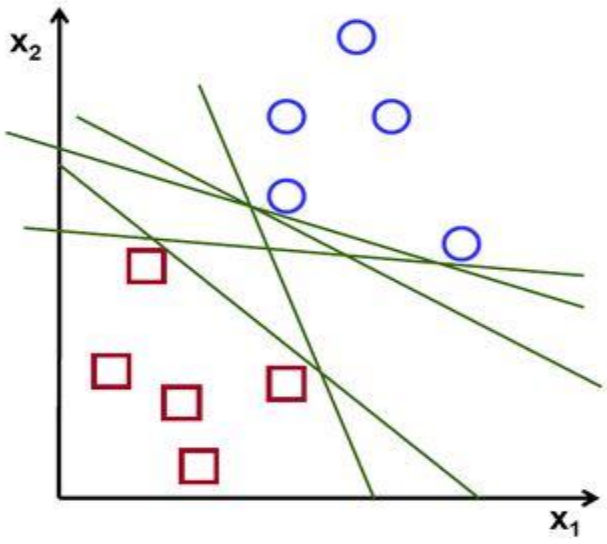


FIGURE 3.1 Hyperplanes separating data

In the above figure you can see that there exists multiple lines that offer a solution to the problem. Is any of them better than the others? Can we intuitively define a criterion to estimate the worth of the lines?

A line is bad if it passes too close to the points because it will be noise sensitive and it will not generalize correctly. Therefore, our goal should be to find the line passing as far as possible from all points. Then, the operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important name of margin within SVM's theory. Therefore, the optimal separating hyperplane maximizes the margin of the training data.

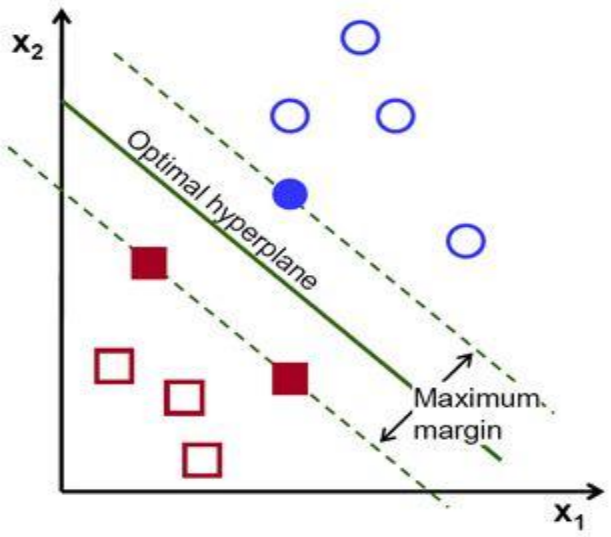


FIGURE 3.2 Optimal Hyperplane

The separating hyperplane has the form $\boldsymbol{w}^T\boldsymbol{X} + b$, where $\boldsymbol{w}$ is the weight vector, $\boldsymbol{X}$ is the input vector and b is the bias. To find the distance from a support vector to the separating hyperplane, we must measure normal or perpendicular to the line, given by $|\boldsymbol{w}^T\boldsymbol{X} + b|/\ \|\boldsymbol{w}\|$. The goal is now to find the **w** and b values that will define our classifier as:

$$\arg\max_{w,b}\left\{\min_n\left(label * (\boldsymbol{w}^T\boldsymbol{X} + b)\right) * \frac{1}{\|\boldsymbol{w}\|}\right\} \quad (3.4)$$

3.4 MACHINE LEARNING IN MEDICAL IMAGE SEGMENTATION

Machine learning has been successfully applied in speech recognition, Mining medical as well as business data, computer vision (including image segmentation), Bio-surveillance, Robot control as well as accelerating empirical sciences. In this section, we focus on applications in image segmentation.

Research in machine learning in image segmentation initially used the naive metric of the pixel error. A classifier was trained to perform the task of boundary detection by measuring its pixel

error relative to a dataset of true boundary labelings. Even with this primitive metric, machine learning delivered superior performance at segmenting natural images as compared to the classic Canny edge detector and a second-moment matrix based detector related to corner detection techniques.

Plath *etal* [10], proposed a machine learning method for multi-class image segmentation in which local and global evidences are coupled. They first formulate a Conditional Random Field that couples local segmentation labels in a scale hierarchy. Then, a global image classification information is used to decide prior to the segmentation process which segment labels are considered possible in a given new image.

Plath's segmentation algorithm follows a number of steps which is briefly summarized here:

1. First, an unsupervised segmentation procedure is used to fragment the whole image into a number of patches at multiple scales.
2. Color, texture and SIFT features are computed on each patch level
3. A global image feature vector is computed from all patch features
4. Trained SVMs are used to predict a class member- ship for each patch and for the whole image.
5. Depending on the image structure and the global classification, they build a CRF model, a dependency tree between patch labels and use the output of SVMs as local evidences.
6. Thresholding the posterior labels for the nest patches produces the final segmentation

Turaga *et al* [15], provided a machine learning algorithm for image segmentation which consists of a parametrized classifier that predicts the weights of a nearest neighbor affinity graph over

image pixels, followed by a graph partitioner that thresholds the affinity graph and finds its connected components. The classifier is used to generate the weights of an affinity graph. The nodes of the graph are image pixels, and the edges are between nearest neighbor pairs of pixels. The weights of the edges are called affinities. A high affinity means that the two pixels tend to belong to the same segment. The classifier computes the affinity of each edge based on an image patch surrounding the edge.

The graph partitioner first thresholds the affinity graph by removing all edges with weights less than some threshold value q. The connected components of this thresholded affinity graph are the segments of the image.

Etyngier *etal* [11], used machine learning techniques in an attempt to improve Active Contour-based image segmentation. In their work, they model a category of shapes as a smooth finite-dimensional sub-manifold of the infinite dimensional shape space, termed the shape prior manifold. This manifold which cannot be represented explicitly is approximated from a collection of shape samples using a recent manifold learning technique called Diffusion maps.

[8] proposed and evaluated several Fuzzy and Neural Network based clustering techniques: Fuzzy C Means Algorithm (FCM), Possibilistic C Means Algorithm, Hierarchical C Means Algorithm, C-mean based Fuzzy Hopfield Neural Network, Adaline Neural Network and Regression Neural Network. These algorithms were applied to the dermoscopic image and compared with the expected lesion segmentation. [21] proposed an algorithm for segmenting white matter lesion using machine learning with weakly labeled MR images. The algorithm only requires the user to provide a few regions of interest (ROI's) containing lesions. An unsupervised clustering algorithm is applied to segment these ROI's into areas. The initial lesion label is used

to further refine the probability distribution estimation for the final lesion segmentation. The advantages of the algorithm are as follows: 1. By using the weak labels, the dependency of the segmentation performance on the expert discrimination of lesion voxels in the training samples is reduced; 2. The training can be done using labels generated by users with only general knowledge of brain anatomy and image characteristics of WM lesion, instead of these carefully labeled by experienced radiologists; 3. The algorithm is fast enough to make interactive segmentation possible.

Xudong Cao etal in [20] presented an efficient and accurate "Explicit Shape Regression" approach to image segmentation specifically for face alignment. In their work, they directly learn a vectorial regression function to infer the entire facial shape from the image and minimize the alignment errors over the training data. They also used a two-level boosted regression, shape-indexed features and a correlation based feature selection to make the regression more effective. In aligning the face, a boosted regression is used to combine T weak regressors ($R^1, \dots R^t, \dots, R^T$) in an additive manner. Given a new image I and an initial face shape S^0, each regressor computes a shape increment δS from the image features and updates the shape thus:

$$S^t = S^{t-1} + R^t(I, S^{t-1}), t = 1, .., T$$

Where the t*th* weak regressor R^t updates the previous shape S^{t-1} to the new shape S^t.

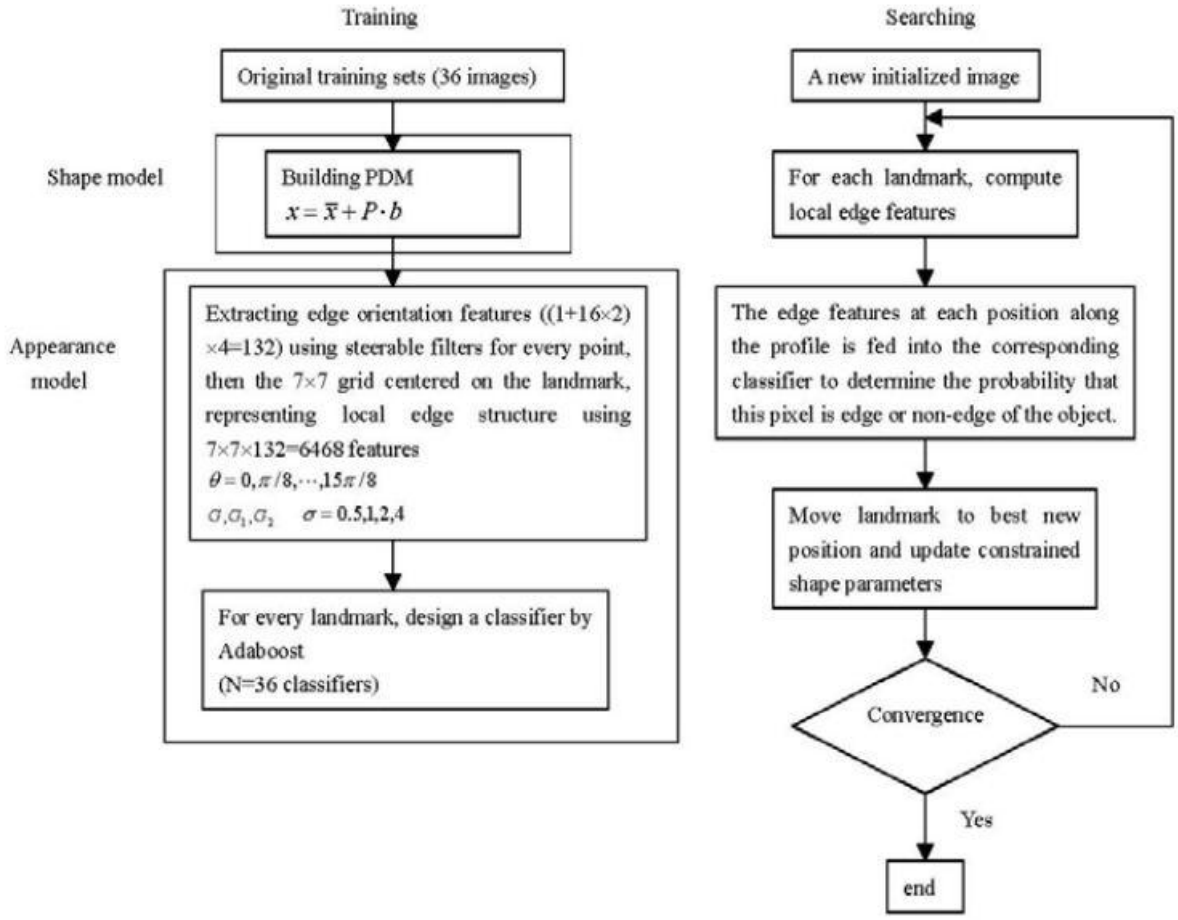
3.5 Statistical Shape Models with Machine Learning

Machine learning algorithms have also been used to improve the performance of known image segmentation algorithms. Li *et al*[14] describes a machine learning approach for improving active shape model segmentation, which can achieve high detection rates. Steerable filters are used to extract local edge features for each landmark point across the training set. An AdaBoost

based machine learning algorithm is used to select a small number of critical features from a larger set and yields extremely efficient classifiers. The aim is to move the landmark points to better locations during optimization along a profile perpendicular to the object contour. The best location is that whose local edge structure is the most similar to the corresponding landmark. First, a point distribution model is also constructed as in the original ASM.

To describe the local edge structure, a 7x7 grid with the landmark point at the center was used to represent the structure of each landmark and 6468 features were computed from the 7x7 grid for each landmark. When the model is fitted to a test image, the scheme starts by computing the 6468 features for each searching point. Instead of sampling the normalized derivative profiles, the feature set at each position along the profile perpendicular to the object contour is fed into a trained classifier to determine the position that this pixel is moved to. The output 1 represents that the probability that the pixel is on the edge is large, 0 means that the probability that the pixel is not on the edge is large. The index along the profile is oriented from non-edge position to the edge position. 36 images with 36 landmark points each was used for the experiments.

Below is a flow chart describing the machine learning approach for improving ASM presented by Li *et al*.



COPYRIGHT: Shuyu Li, Litao Zhu, Tianzi Jiang

FIGURE 3.3: AdaBoost-ASM algorithm

Cristinacce *et al* [2] also described a machine learning approach of fitting a set of local feature models to an image. A boosted regression predictor which learns the relationship between the local neighbourhood appearance and the displacement from the true feature location was used. A point distribution model is created as in the ASM. This model is then used to search for objects of interest in new image instances. In their work, they used boosted feature detectors similar to those of the Viola and Jones Face detector and used GentleBoost in their training. The aim of the GentleBoost algorithm is to learn a discrimination function between a set of positive and

negative examples. Where positive examples are image patches centered on the correct feature locations and negative examples are nearby examples displaced from the true locations.

The summary of the search procedure is as follows:

1. Find initial feature points - for example using a global detection method

2. Iterate the following:-

    (a) Search around the current feature location with a feature detector - Or alternatively predict the improved feature location using boosted regression

    (b) Fit the shape model to the current set of feature locations to remove outliers

Until Converged.

CHAPTER FOUR

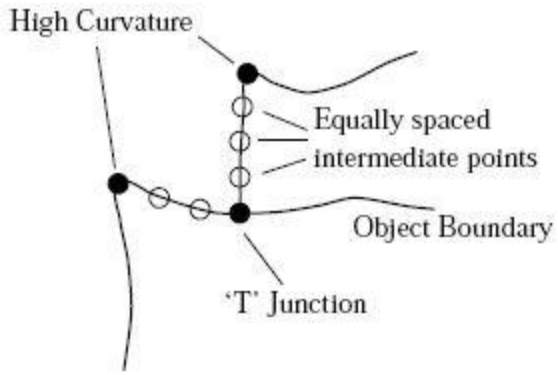
4.1 ACTIVE SHAPE MODELS

A shape can be defined as the quality of a configuration of points which is invariant under some transformation [18]. In this section, we describe how to build models of shape and how those models can be used to search for an object in an image. Our aim is to build models, which are flexible enough to allow us analyse new shapes so as to be able to delineate structures of interest from a new image. To build a statistical shape model, a set of landmarked points is given which are then aligned to remove differences due to translation, rotation and scaling before estimating the shape distribution.

4.1.1 Landmarks

The first step in building a statistical shape model is to mark up or annotate each of a series of images with a set of corresponding points. This can be done manually by a human expert or automatically. This part of the process is very time consuming and automatic methods are currently being developed to aid this process. The choice of landmark points is imperative to realizing good models (models that generalize well) and these are points that can be consistently located from one image to another.

Good choices are clear corners of object boundaries, ‘T’ junctions between boundaries or easily located biological landmarks. However, not all images have enough of such points to describe the shape in its entirety. To get points that describe the shape entirely, the corners and ‘T’ junctions would have to be augmented with other points along boundaries which are arranged to be equally spaced between well-defined points. Figure 4.1 shows examples of suitable landmark points.



COPYRIGHT: T. COOTES [18]

FIGURE 4.1: Good choices for landmark points

In a 2-D image, landmark points can be represented by a set of coordinate points in the plane $\{(x_i, y_i)\}$. A single example in the training set can then be represented as a 2n element vector, x, where x = $(x_1, \dots\dots, x_n, y_1, \dots\dots, y_n)^T$. See Figure 4.2 for images with marled points.

If we have *m* training examples, we generate *m* such vectors x_j. We then wish to eliminate differences attributable to location, rotation and scaling before performing statistical analysis.

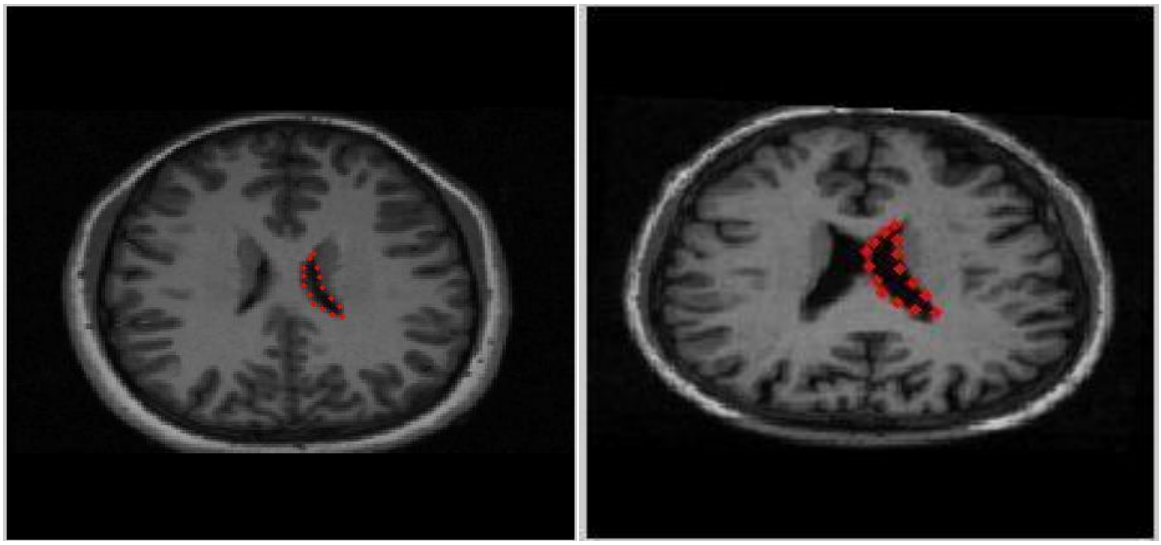


FIGURE 4.2:Images with landmark points.

4.1.2 Aligning the Training Set

The aim of this step is to align each shape so that the sum of distances of each shape to the mean is minimized. This removes differences due to translation, rotation and scaling. See Figure 4.3a for unaligned points and Figure 4.3b for aligned points. This process is poorly defined unless constraints are placed on the alignment of the mean (for example, ensuring it is centered on the origin, has unit scale and some fixed but arbitrary orientation). The alignment algorithm is as follows:

- Translate each shape instance so that its centre of mass is at the origin
- Align every shape in the training set with the first shape (or any other shape) in the training set by rotating, scaling and translating

REPEAT UNTIL CONVERGENCE

- Calculate the mean from the aligned shapes
- Normalize the orientation, scale and origin of the mean
- Re-align every shape with the current mean

The different operations during the alignment process will affect the final shape distribution. First, each shape is centered around the mean so that |x| = 1 and then an orientation is chosen that minimizes ($D = \sum|x_i - \bar{x}|^2$). The scale constraint means that all the corners lie on a circle about the origin.

Another approach is to transform each shape into the tangent space to the mean so as to minimize D or allow both scaling and orientation to vary when minimizing D.

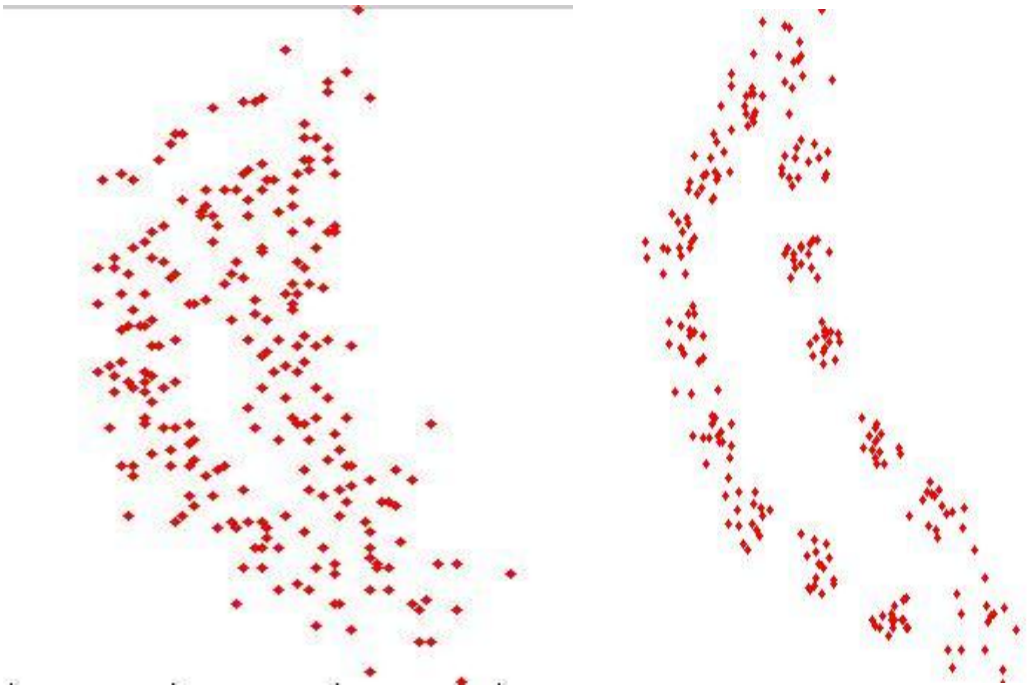


Figure 4.3: A set of unaligned points and aligned points respectively

4.1.3 Modeling Shape Variation

At this point, we now have a set of points which are aligned into a common co-ordinate frame as shown in Figure 4.3b. Our aim is to build a model from which we can generate shapes similar to those in the training set. First, we have to reduce the dimensionality of the data from n-dimension to something smaller using Principal Component Analysis. The data form a cloud of points in the n-dimensional space. PCA computes the main axes of this cloud, allowing one to approximate any of the original points using a model with fewer than n parameters. Modeling shape variation is as follows:

Compute the mean of the data,

$$\bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m} x_i \qquad (4.1)$$

Compute the covariance of the data,

$$\boldsymbol{S} = \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{x})(x_i - \bar{x})^T \quad (4.2)$$

Compute the eigenvectors, p_i and the corresponding *t* eigenvalues λ_i of $\boldsymbol{S}$ (sorted).

Any point in our allowable shape domain can be reached by taking the mean and adding a linear combination of the eigenvectors. Any shape in the training set can be approximated using the mean shape and a weighted sum of these deviations *t* modes:

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Pb}, \quad (4.3)$$

Where $\mathbf{P} = (\mathbf{p}_1\ \mathbf{p}_2 \ldots \mathbf{p}_t)$is the matrix of the first *t* eigenvectors and $\mathbf{b} = (\mathrm{b}_1\ \mathrm{b}_2 \ldots \mathrm{b}_t)^T$ is a *t* dimensional vector of weights given by

$$\mathbf{b} = \mathbf{P}^{\boldsymbol{T}}(\mathrm{x} - \bar{\mathrm{x}}) \quad (4.4)$$

The vector b defines a set of parameters of a deformable model and by varying its values within suitable limits, we can generate new examples of plausible shapes. The limits for the values of the vector b are derived by examining the distributions of the parameter values required to generate the training set. Suitable limits for b_k are of the order

$$-3\sqrt{\lambda_k} \leq b_k \leq 3\sqrt{\lambda_k} \quad (4.5)$$

Alternatively, $\{b_1 \ldots \ldots b_t\}$ can be chosen such that the Mahalanobis distance D_m from the mean is less than a suitable value, D_{max}:

$$D_m^2 = \sum_{k=1}^{t} \left(\frac{b_k^2}{\lambda_k}\right) \leq D_{max}^2 \quad (4.6)$$

The number of modes *t* can be chosen in one of several ways. The simplest way is probably to choose *t* that accounts for a given proportion (e.g. 98%) of the variance exhibited in the training data.

4.1.4 Using Shape Models in Image Search

At this point, we have a Point Distribution model, which we would like to use in image search. We must then find the set of parameters which best match the model to the image. Given a rough starting approximation, a model instance can be fit to an image. For each point in the rough starting approximation, it searches for better points around that point, usually along the normal. The initial approximation takes the form of estimates for the position at which the model should be placed, its orientation, scale and shape parameters required to fit the model to the image. After the initialization, we now wish to find the best pose and shape parameters to match a model instance to a new set of image points Y.

An iterative approach for achieving this is as follows:

- Initialize the shape parameters, b, to zero
- REPEAT UNTIL CONVERGED
    - Generate the model instance $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Pb}$
    - Find the pose parameters (X_t, Y_t, s, θ) which best maps **x** to the new points Y.

- Invert the pose parameters and use to project Y into the model co-ordinate frame:
- $y = T^{-1}_{X_t,Y_t,s,\theta}(Y)$
- Project y into the tangent plane to $\bar{\mathbf{x}}$ by scaling 1/(y. $\bar{\mathbf{x}}$**).**
- Update model parameters to match y

$$\mathbf{b} = \mathbf{P}^T(y - \bar{\mathbf{x}})$$

Apply constraints on **b**.

The Active shape model has shown in many applications to be a fast, simple and accurate method of segmentation, but sparsely uses image information. The search for better points is mostly along the normal and when the correct points are not perpendicular to the normal of the initial points, the resulting segmentation leaves much to be desired. In the next section, we describe a new way of overcoming some of the short comings of the Active Shape model segmentation to realize better segmentation accuracy.

CHAPTER FIVE

5.1 ASM WITH KNN AND HAAR FEATURES

In this section, we describe a new way of moving landmark points to better positions in ASM image segmentation. Haar filters are used to extract local image edge features and K-Nearest Neighbour algorithm is used to learn better positions for each landmark. The best position is the position whose local edge structure is closest to a corresponding landmark. First, points are marked manually and those points are used to create a point distribution model as described in the preceding section. The model is then used to search for an object of interest in a new image as in the original ASM. The KNN-ASM then examines a small grid around each point for better points.

5.1.1 Haar Feature Extraction

Haar features measure vertical, horizontal, central, and diagonal variations of pixel intensities. They are defined as the difference between the sum of image values on two, three and four rectangles. Our algorithm classifies points as good or bad points based on values of simple haar features. Using haar features was preferred to using pixel intensities directly because features can encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data.

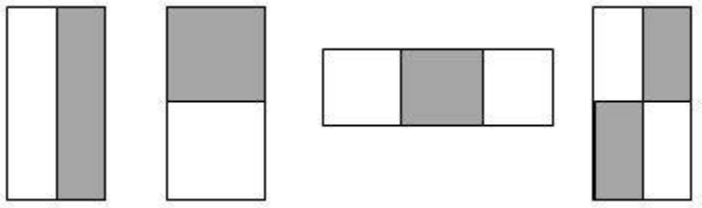


FIGURE 5.1:Haar-based rectangular features.

The value of a two-rectangle feature is the difference between the sum of the pixels within two rectangular regions. A three=rectangle feature computes the sum within the two outside rectangles subtracted from the sum in a center rectangle. A four-rectangle feature computes the difference between diagonal pairs of rectangles.

Haar features are computed very rapidly using an intermediate representation called integral image [12]. The integral image at location (x, y) is the sum of the pixels above and to the left.

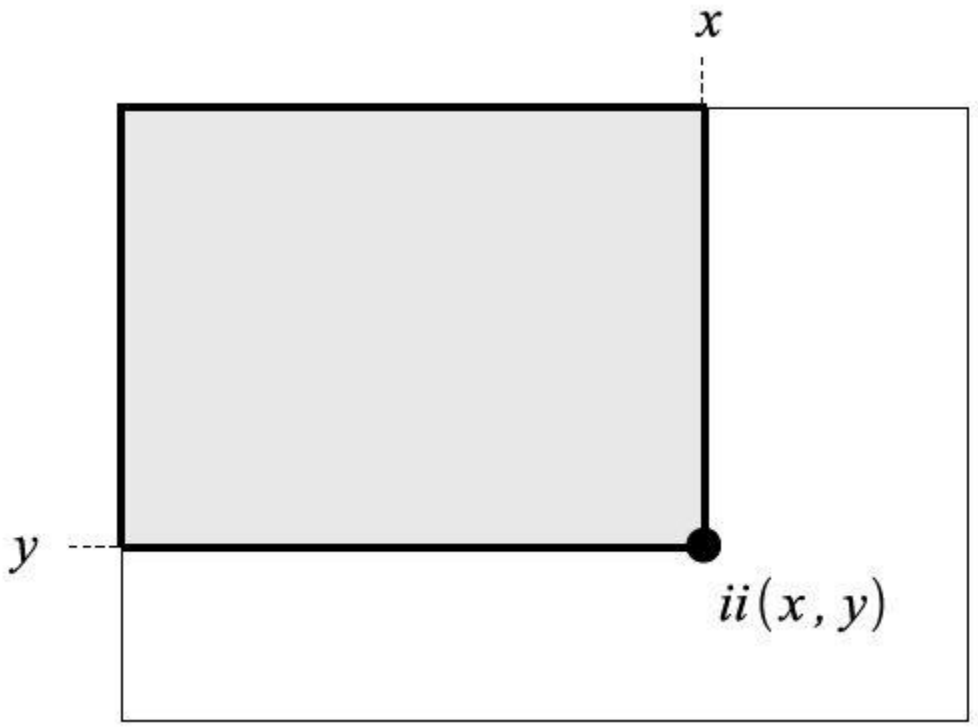


Figure 5.2: The value of integral image at point (x, y) is the sum of all pixels above and to the left.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (5.1)$$

Where $ii(x, y)$ is the integral image and $i(x, y)$ is the original image. The integral image is computed in one pass over the image using the recurrence

$$s(x,y) = s(x,y-1) + i(x,y) \quad (5.2)$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \quad (5.3)$$

Where $s(x,y)$ is the cumulative row sum, $s(x,-1) = 0$, and $ii(-1,y) = 0$

Using the integral image, Haar features can be computed efficiently.

5.2 KNN-ASM

For each landmark point, Haar features were computed to represent the local edge structure. A point in an image with its corresponding points in other training images formed the positive examples, while two and three pixels away were used as negative examples for that particular point.

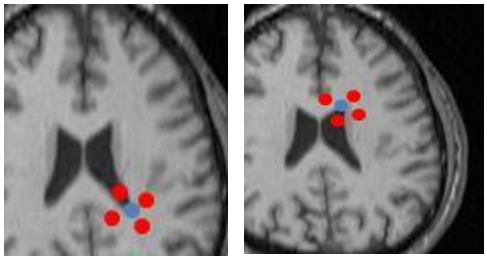


FIGURE 5.3: Positive and Negative Examples

Given a training set of positive and negative examples, any machine learning could be used to classify unknown points. In this thesis, we use KNN to find the closest points to the positive examples.

A point distribution model is created as in the original ASM and used to search for an object of interest in a new image moving points to better points by iteratively checking points along the normal.

A small region around each final point of the ASM search is examined using the KNN algorithm for better positions. This improves the segmentation by examining points which although were good points, but were not along the normal of the initial position of the model. When this process completes, the shape parameters are constrained to realize the closest plausible shape to the final shape.

5.3 KNN-ASM FRAMEWORK

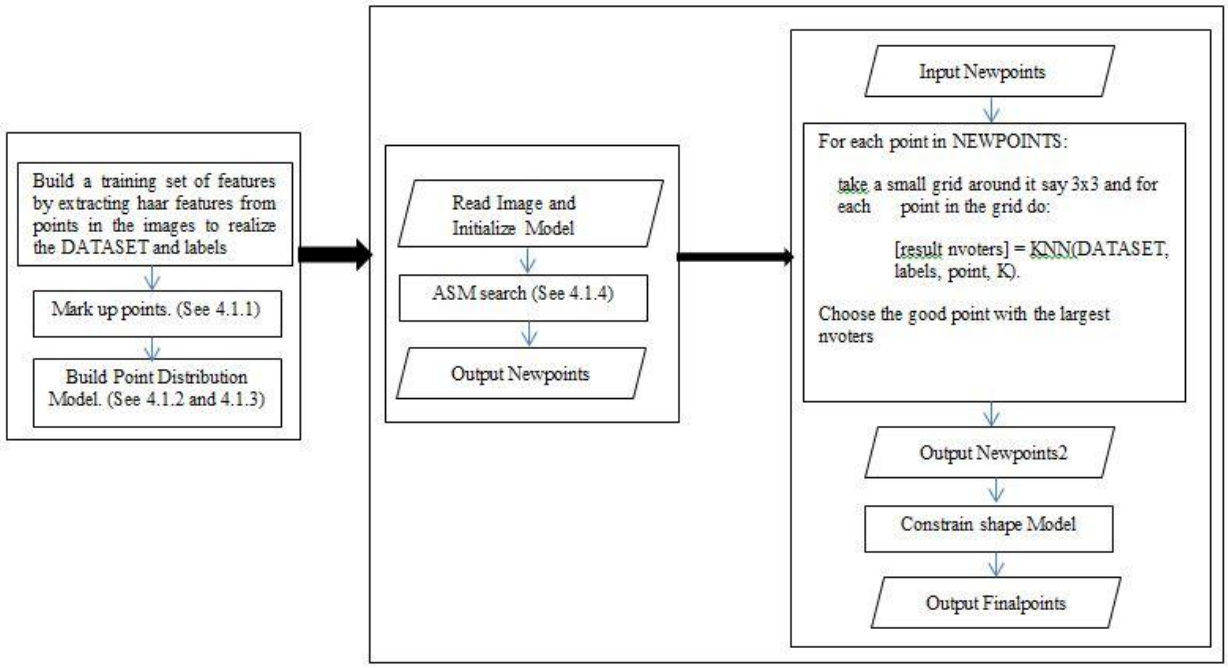
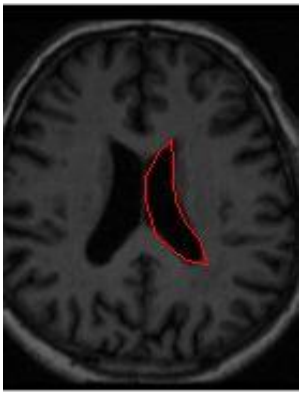


Figure 5.4: KNN-ASM Framework

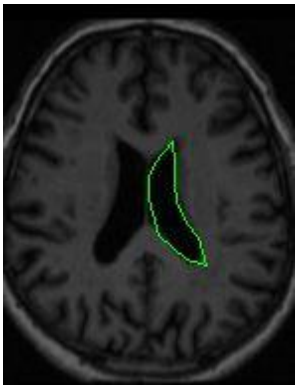
5.4 RESULTS AND DISCUSSIONS

A total of 109 2D slices of MRI images of the brain was used for the experiment. 86 of those were used to build the Point Distribution Model and to train the K-Nearest Neighbour classifier. The lateral ventricles of the other 23 images were segmented first using the Active shape Model

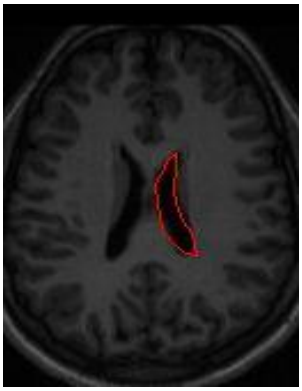
algorithm, and then the KNN-ASM algorithm. The results from both the ASM and the KNN-ASM were compared with available ground truth for the images using the Dice overlap metric. Our KNN-ASM was trained and built with 86 images and has been tested with 23 images so far achieving improved segmentation accuracy on the average of 81.22% $\pm$ 3.12% (standard error on mean) over the ASM's 80.37% $\pm$ 0.0295% (standard error on mean) using the dice overlap metric. The standard deviation from the mean with the KNN-ASM segmentation gave 14.95%. With the ASM segmentation, the standard deviation from the mean was 14.16%. The figures below show some of the segmentation results given alongside their dice overlap values with correctly segmented images.



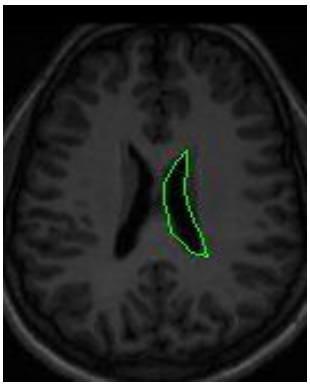
ASM segmentation (87%)



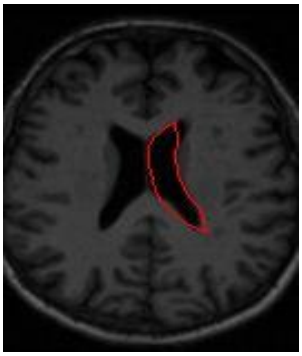
KNN-ASM segmentation(88%)



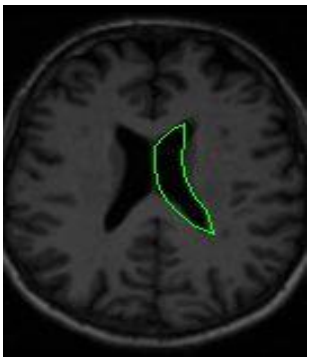
ASM segmentation (82%)



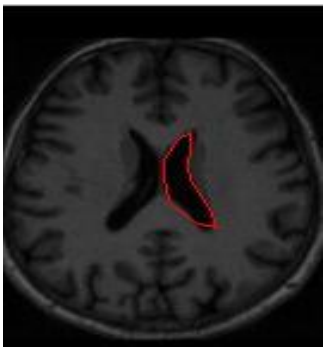
KNN-ASM segmentation (84%)



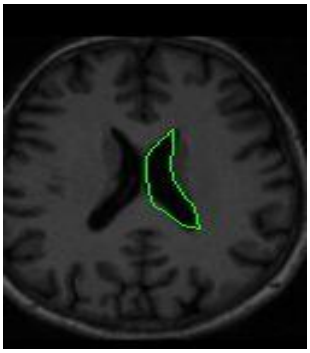
ASM segmentation (87%)



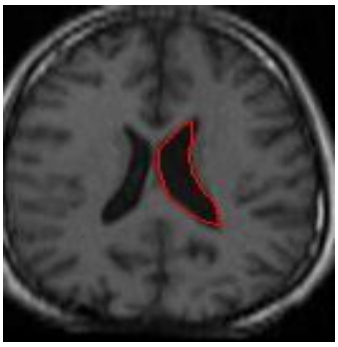
KNN-ASM segmentation (89%)



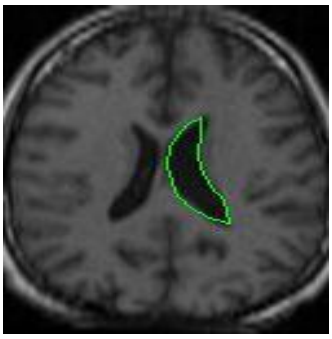
ASM segmentation (89%)



KNN-ASM segmentation (90%)



ASM segmentation (80%)



KNN-ASM segmentation (81%)

Figure 5.5: Segmentation Results

From our experiments and results, we achieved improved segmentation accuracy on the average with the 23 images used for testing. But the 23 images only represent a small sample of brain images. A t-test on the results however shows that the improvement was not as significant as expected. This could have been due to the number of training examples and the sophistication of the machine learning algorithm used.

K-Nearest Neighbour algorithm – the machine learning algorithm that was used in this work – is a simple but effective algorithm but falls short in performance (predictive accuracy) when compared with more sophisticated machine learning algorithms like support vector machines, neural networks and random forests. Also when the training data is very large, KNN will not be the ideal algorithm since it keeps all the data in memory.

Results from our KNN-ASM however indicate that machine learning can be used to improve automatic segmentation accuracy of medical images in the way that has been described in this work.

CHAPTER SIX

6.1 CONCLUSION

We have proposed an algorithm which combines the popular Active Shape Model with a machine learning algorithm to improve the segmentation accuracy of medical images. In the proposed algorithm, final points that resulted from the ASM search are used as starting points for the KNN-ASM, with a small neighbourhood of each point being examined for better positions for the points. We validated the effectiveness of the proposed algorithm using a good number of images of the brain as we segmented the lateral ventricles of the brain. Results show that machine learning techniques can improve the performance of Active Shape Model in medical image segmentation.

6.2 FUTURE WORK

For future work, we propose the exploration of other machine learning algorithms like random forest and also the extension of this idea to segmenting other medical images apart from brain images.

REFERENCES

1. B. Bhanu , S. Lee and S. Das "Adaptive image segmentaton using multi-objective evaluation and hybrid search methods", AAAI Fall Symp. Machine Learning in Computer Vision, 1993.

2. D. Cristinacce and T.F. Cootes, "Boosted Regression Active Shape Models", Proc. British Machine Vision Conference, Vol. 2, 2007, pp.880-889.

3. D.J Withey and Z.J Koles, "Medical Image segmentation: Method and Software". Proceedings of NFSI8 ICFBI, October 12 -14, 2007.

4. D.J Withey and Z.J Koles, "Three generations of medical image segmentation: Methods and Available Software". International Journal of Bioelectromagnetism. Vol 9, No 2, 2007.

5. P. Etyngier, F. Ségonne, R. Keriven, "Active-contour-based image segmentation using machine learning techniques", Med Image Comput Assist Interv. 2007;10(Pt1):891-9.

6. Fuzzy c-means clustering. Available at: "sites.google.com/site/dataclusteringalgorithms/fuzzy-c-means-clustering-algorithm", Accessed March 23, 2013.

7. V. Jain, H.S. Seung, S.C. Turaga, "Machines that learn to segment images: a crucial technology for connectomics", Curr Opin Neurobiol. 2010 Oct;20(5):653-66.

8. L. P. Suresh, K.L. Shunmuganathan and S.H. K. Veni, "Dermoscopic Image Segmentation using Machine Learning Algorithm", American Journal of Applied Sciences 8 (11): 1159-1168, 2011.

9. N. Sharma and L. M. Aggarwal, “Automatic medical image segmentation techniques”. Journal of medical physics. Jmedphys 2010, Jan-Mar, 35(1) 3-14.

10. N. Plath, M. Toussaint, S. Nakajima, “Multi-class image segmentation using conditional random fields and global classification”. Proceedings of the 26th Annual International Conference on Machine Learning, 2009,pp.817-824.

11. P. Etyngier , F. Segonne and R. Keriven N. Ayache , S. Ourselin and A. Maeder "Active-Contour-Based Image Segmentation Using Machine Learning Techniques", Proc. Medical Image Computing and Computer-Assisted Intervention, pp.891 -899 2007.

12. P. Viola and M. Jones. “Robust real-time object detection”. International Journal of Computer Vision, 57(2):137-154, 2004.

13. S. Almari, N.V. Kalyankar and S.D. Khamitkar, “Image segmentation by using Thresholding techniques”. Journal of Computing, vol 2, issue 5, May 2010, ISSN 2151-9617.

14. S. Li, L. Zhu, T. Jiang, “Active Shape Model Segmentation Using Local Edge Structures and AdaBoost”. Medical Imaging and Augmented Reality Lecture Notes in Computer Science Volume 3150, 2004, pp 121-128.

15. S. C. Turaga, K. L. Briggman, M. Helmstaedter, W. Denk, H. S. Seung, “Maximin affinity learning of image segmentation”, In Advances in Neural Information Processing Systems 22 (2009), pp. 1865-1873.

16. T.F. Cootes, D. Cooper, C.J. Taylor and J. Graham, "Active Shape Models - Their Training and Application." Computer Vision and Image Understanding. Vol. 61, No. 1, Jan. 1995, pp. 38-59.

17. T.F.Cootes, G.J. Edwards and C.J.Taylor. "Active Appearance Models", IEEE PAMI, Vol.23, No.6, pp.681-685, 2001

18. T.F. Cootes, "An Introduction to Active Shape Models". Available at: http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/Papers/asm_overview.pdf, Accessed April 19 2013.

19. T. M. Mitchell, "The Discipline of Machine Learning". Available at: http://www.cs.cmu.edu/~tom/pubs/MachineLearning.pdf, Accessed April 1 2013.

20. X. Cao, Y. Wei, F. Wen, J. Sun, "Face alignment by Explicit Shape Regression". Proceedings of the 2012 IEEE Conference on Computer Vision Pattern Recognition (CVPR), pp.2887 – 2894.

21. Y. Xie and X. Tao, "White matter lesion segmentation using machine learning and weakly labeled MR images", Proc. SPIE 7962, Medical Imaging 2011: Image Processing, 79622G (); doi:10.1117/12.878237; http://dx.doi.org/10.1117/12.878237.

22. D. L. Pham, C. Xu, and J. L. Prince, "CURRENT METHODS IN MEDICAL IMAGE SEGMENTATION", Annual. Rev. Biomed. Eng. 2000.2:315-337.

23. M. Kass, A. Witkin, D. Terzopoulos. International Journal of Computer Vision In International Journal of Computer Vision, Vol. 1, No. 4. (1 January 1988), pp. 321-331