M. Tech. (Computer Science) Dissertation Series

Estimation of Image Features Representing Facial Emotions for Emotion Synthesis

A dissertation submitted in partial fulfilment of the requirements of
M.Tech (Computer Science)
degree of Indian Statistical Institute, Kolkata.

by Archan Ray

Under the supervision of

Prof. Dipti Prasad Mukherjee

Electronics and Communication Sciences Unit
Indian Statistical Institute, Kolkata
India
August 6, 2015

Abstract

We develop a method to estimate emotion-specific features on human face. Application of such a method include characterizing an emotion class and synthesis of emotions. The emotion-specific features can also be used to study the statistical differences between two clusters, one facial expression images with no expressions and two facial expression images with some or maximum emotional content. Once the feature vectors are extracted from the input data, we classify the data and use the normal to the classifier to trace the changes that a facial expression image may undergo in different stages of an emotion. We use Support Vector Machines learning algorithm to construct an optimal classifier. In the result section we show that we are able to reduce the number of features by 66.36% as compared to the total number of pixels. We show that using these features and state-of-the-art methods to synthesize images, we improved SNR of the synthesized image by 13.20% and also improved the cluster measures between a cluster of no-expression images and a cluster of with-expression images.

Declaration

I, **Archan Ray** (**CS1318**), registered as a student of **M. Tech** program in **Computer Science**, **Indian Statistical Institute**, **Kolkata** do hereby submit my Dissertation Report entitled "**Estimation of Image Features Representing Facial Emotions for Emotion Synthesis**". I certify

1. The work contained in this Dissertation Report is original and has been done by me under the guidance of my supervisor.
2. The material contained in this Dissertation Report has not been submitted to any University or Institute for the award of any degree.
3. I followed by guidelines provided by the Institute in preparing the report.
4. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of report and giving their details in the bibliography.

.....

Place : ISI, Kolkata
Date : July, 2015

Archan Ray
(CS1318)

Acknowledgments

First of all I'd like to thank Dipti Prasad Mukherjee, Professor & Head, ECS Unit, ISI Kolkata for being the guiding light of my last one and half years. He has been truly inspiring, at times criticising and seldom punishing to bring out the best of me. He will be a significant reason for what I am today and I will be in coming years, and I am highly indebted to him for that. Thank you sir, thank you for all the patience you had with me.

A dissertation involves influences from all of one's surrounding. Fortunately I shared my surroundings with people of most amiable nature who would rush to aid you in however simple way they can. I would like to thank Professor C. A. Murthy (MI Unit, ISI Kolkata) for those moments of doubt clearances and mathematical enlightments. I would also like to thank Swagatam Das (Assistant Professor, ECS Unit, ISI Kolkata) for helping out whenever I was stuck in my work, with whatever mathematical problems I faced. The two courses under him were the most I had enjoyed being a part of here in ISI Kolkata. I would also like to thank Sarbani Palit (Associate Professor, CVPR Unit, ISI Kolkata) for teaching me digital signal processing and advanced digital signal processing and allowing me to work with her last summer. I am deeply inspired by the teachings of Palash Sarkar (Professor, AS Unit, ISI Kolkata) and Mandar Mitra (Associate Professor, CVPR Unit, ISI Kolkata). I hope, I could hold on to the motivations you provided us with during the very short span of interactions we had. I'd also like to thank Sandip Das (Professor, ACM Unit, ISI Kolkata) for teaching us data and file structures. I'm grateful to Dr. Hiranmay Ghosh for the summer I spent in TCS Innovation Labs (Gurgaon) and also for introducing me to the world of emotion recognition and analysis.

I would like to specially thank Tamal Batabyal (University of Virginia) for being the source of every tiny bit of inspiration one can psuh into someone. He has been a friend, a brother and at times a teacher. All that I have learnt in mathematics is for the suggestions this person put forward in the last summer. Thank you for introducing me to Funtional Analysis. I thank him for all the bestowings he endowed me with love. I would also equally like to thank Kaushik Chakraborty (INRIA, France) for everything similar. These are the two men who stood by me always, as friends and brothers, and I shall be deeply indebted to them for that.

I would also like to thank Dhrubajit Chowdhury (will be pursuing PhD at Michigan Tech), for his constant support from the undergraduate days. I sincerely thank Sanjay Bhattacharjee (ENS-Lyon) for his guidance in literatures on mathematics. I would also like to acknowledge Debleena Thacker (SM Unit, ISI Kolkata). I sincerely thank Arindam, Kushal, Sourya, Shankha, Tanmoy, Amit, Ravindra, Ranjan, Chirag, Harmender, Ashwin, Durgesh, Debjyoti, Dnyaneshwar, Ansuman, Mayur, Sanjana, Kunal Ray, Arkadeep, Sebati, Ekta, Swapna Agarwal, Bikash Santra, Sanjoy Mazumder, Ansuman Das, Sanchayan Santra, Soumen Nandi, and many more.

I sincerely thank Indian Statistical Institute, Kolkata for bestowing us with unmatched facilities of libraries and labs. I would also thank everyone whoever impacted my life in these two years of my study here.

I would thank Ayan Bharadwaj (CMC Ltd.), Alok Saha (Wipro), and Abanti Deb Chowdhury (City University of New York) for being my best buddies all my life, and for all of their support.

Finally I would like to thank Ma, Baba and Dada. It is for your constant support in all thicks and thins that I exist, and all of me is you. This dissertation is dedicated to you. It has been emotional.

Archan Ray
Kolkata, West Bengal
July 2015

Ceritificate of Approval

This is to certify that this thesis titled **Estimation of Image Features Representing Facial Emotions for Emotion Synthesis** submitted by Archan Ray, embodies the work done under my supervision.

Prof. Dipti Prasad Mukherjee
Electronics and Communication Sciences Unit,
Indian Statistical Institute,
Kolkata

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human emotions are mostly surfaced on the face. There are six basic human emotions considered as universal [9]. They are anger, joy, happiness, sorrow, disgust and fear.

The question that we would like to address is: from the images of face showing any one expressions (say, anger, joy, disgust, sorrow etc.), would it be possible to capture the emotion-specific features? In more lucid terms, given a video of face image showing a particular emotion (say, anger), can we identify a feature vector that represents that specific emotion (anger)? Of course the more challenging issue is whether this feature vector characterizing an emotion is independent of the structure of face. That is, whether feature vector extracted from the face of person A is relevant for the person B.

To understand the more general case, assume that we are successful in extracting this emotion-specific feature vector from a data set containing video data of facial emotion expressions of five persons. The challenge is can this feature vector be equally true or relevant for a sixth person whose face image was not included in extracting the original feature vector.

In case we are successful in extracting such a feature vector specific to an emotion, we can use this feature vector to classify emotions. Given a set of training and test image sequences, we can possibly extract feature vectors from training images and then use these feature vectors to classify or label test video frames into any one of six basic emotion classes.

Such an emotion-specific feature vector can be utilized in another scenario. We can think of a model to synthesize an emotion. If an expression-neutral face image is available as a shape vector ($\mathbf{x}_i$) and a feature vector ($\mathbf{f}_i$) for a particular emotion is derivable as noted earlier from the training image sequences, then an emotion can be synthesized as some function of $\mathbf{x}_i$ and $\mathbf{f}_i$. If our modelling is successful and the quality of $\mathbf{f}_i$ is reliable then an emotion and even degree or extent of emotion

can also be imparted on the test (emotion-neutral) face. Hamm *et.al.* in [16] tried to separate pose and expression of human face. A solution was proposed by decomposing the appearance manifold into invariant substructure and variant substructure, and then clustering of similar expressions were done using factorized isometric embedding. But the problem of imparting the expressions on a test face was not addressed. The scenario can be further explained as follows.

Consider there are two clusters of face images. One cluster, say C_1, represents faces with no or little emotions while the other cluster, say C_2, represents faces with (maximum degree of) emotions. In reality, however, there will be a transition from no emotion to full or maximum degree of emotion (for example, face showing no anger to maximum anger). So, one utility of our synthesis model could be to generate the intermediate faces with emotions leading to full-blown emotion. This is a valid application of *in-betweening* of emotions used in animation [27]. The term in-betweening refers that end frames showing no or full emotions are available while the model tries to develop frames between the end frames (between the extent of emotions).

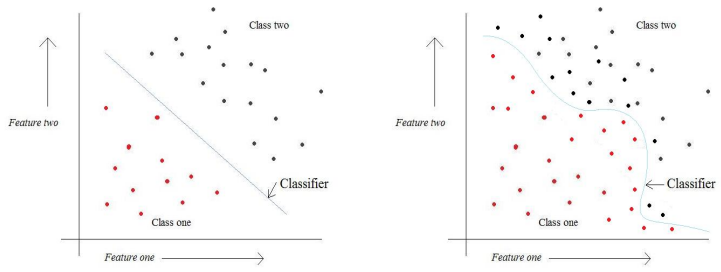
The emotion in-betweening can be visualized as a transition between two clusters C_1 and C_2 when C_1 and C_2 are separated by a classifier hyper-plane, say $\mathbf{y}$. If we assume that there is a hyper-plane, say $\mathbf{h}$, leading from C_1 to C_2, and $\mathbf{h}$ is orthogonal to $\mathbf{y}$, then the in-betweening emotions are ways to find points on $\mathbf{h}$.

Note that for any of the universal emotional expressions that we are discussing in this report, for example, C_1 and C_2 as mentioned above, cannot be separated by linear separating planes. In the next section, we show that non-linear classifiers will be better to separate clusters like C_1 and C_2. In section 3, we show a method with which we can extract emotion-specific-features from dataset containing video data of facial expressions of, let, k persons, and use it for some $(k+1)^{th}$ person. In section 4, we use the emotion-specific-features of section 3 to synthesize in-between frames as one test frame with little or no-expression make its transition to a frame with maximum expression. In section 5, we present the results of our methods, and in section 6, we discuss future prospects of our work, and in section 7 we conclude.

Chapter 2

Inter-cluster Separability

Consider two clusters δC_1 and δC_2 as mention in the previous chapter. To repeat, let face images with no expression represent one cluster δC_1 and face images with some or maximum emotional content of any one type (*e.g.*, anger) be the other cluster δC_2. Let $N \in \mathbb{R}$, where N is the total size of the two clusters, and the populations be defined as a set $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$. To differentiate between the two classes, let us assign labels to each population as $Y \in \{-1, 1\}$. A linear classifier can be constructed in linear space (*i.e.*, the input space) if the populations are linearly separable. A set of population is said to be non-linearly separable if there does not exist any linear classifier that may separate the two classes in input space. Ideally an example of a linearly separable class of population with two features can be shown in Fig 2.1(a), and non-linearly separable class of population with two features can be shown in Fig 2.1(b).



(a) (b)

Figure 2.1: (a) *Linear classification of two classes*, (b) *Non-linear classification of two classes*

Now we return to the problem of separability of clusters δC_1 and δC_2. As a basic step to capture expression we need to understand the distribution of the data in the input space or some higher dimensional space where they are lineary

separable. So we study two statistical procedures to be implemented for better understanding of the separability of the populations.

2.1 Principal component analysis

Principal component analysis (PCA) [19] is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called *principal components*. The principal components are the eigenvectors of the covarince matrix of the given population. The transformation is done in such a way so that, the first principal component has the largest variances, and each of the succeeding principal component have maximum variance among the the components orthogoal to the preceding component. So let $\bar{\mathbf{x}}$ be the mean of the input set X, $P = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N\}^T$ be a matrix of principal components of X, and $\mathbf{b}_i$ (a column vector) be the set of associated coefficients of P for the i^{th} input vector $\mathbf{x}_i$, then $\mathbf{x}_i$ can be written as,

$$\mathbf{x}_i = \bar{\mathbf{x}} + P\mathbf{b}_i. \tag{2.1}$$

Now consider that the principal components are arranged in decreasing order of variance in P. Then the associated coefficients $\mathbf{b}_i$ are also arranged accordingly. We call these coefficients as associated weight values.

2.2 Kernel principal component analysis

Kernel PCA (KPCA) [29] is an extension of PCA using kernel methods to analyze multivariate data. Let there be N samples in X and $X \subseteq \mathbb{R}^n$. It is natural that if we project the population X to a space of dimension d such that $d \in \mathbb{N}$ and $d \geq N$, then the data would almost always be linearly separable. Let the such a d-dimensional space be $\mathbb{F}$. Let ϕ be a map such that $\phi : X \mapsto \mathbb{F}$, and inner product $\langle \cdot, \cdot \rangle$, be defined in $\mathbb{F}$. Then the input population is transformed to $\mathbf{\Phi} = \{\phi(x_1), \phi(x_2), ..., \phi(x_n)\}$. Let an eigenvector of the covariance matrix of $\mathbf{\Phi}$ be $\mathbf{v}$. Let there exist coefficients α_i such that $(i = 1, 2, ..., N)$, then by KPCA,

$$\mathbf{v} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i). \tag{2.2}$$

So let for k^{th} eigenvector $\mathbf{v}^k$, the set of coefficients be α^k. Then if x_j be test point from the given population then the projection of $\phi(x_j)$ on $\mathbf{v}^k$ is given as,

$$\langle \mathbf{v}^k, \phi(x_j) \rangle = \sum_{i=1}^{N} \alpha_i^k \langle \phi(x_j), \phi(x) \rangle. \tag{2.3}$$

So α^k are the associated weight vectors in KPCA, analogous to vectors $\mathbf{b}_i$ in PCA. Here we consider that $\mathbf{v}^k$'s are arranged in the decreasing order of variance, and as such α^k's are also ordered in decreasing order of variance.

2.3 Separability of face images with and without emotions

So now let us see what constitutes the two clusters δC_1 and δC_2. Let there be r sequence of images for any one type of emotion (say, anger). In each sequence, let an image with no expression be the *reference frame*. Thus for r sequences we have r reference frames. If we subtract any frame from one sequence from its reference frame, the result is a difference image. Let the population consist of difference images from all r sequences. In such a case consider two clusters of images, one, a cluster of difference images from faces with no expression, and two, a cluster of difference images from faces with emotional content. We call these clusters as δC_1 and δC_2 repectively. To under cluster separability, PCA and KPCA was performed on the population of difference images. We consider the first three principal components only. Let $\mathbf{b}_1, \mathbf{b}_2$, and $\mathbf{b}_3$ be the weights corresponding to the first three principal components in PCA and α^1, α^2 and α^3 to be the weights corresponding to the first three principal components in KPCA.

We validate our results on the benchmark Cohn Kanade dataset [20]. From this database we select 44 sequences of 44 subjects displaying anger (expression labels are known). Each sequence contains at least 7 frames. Each sequence starts from an image with almost no expression and in subsequent frames is deformed to an image showing maximum expression. The images are digitized in 241×321 pixel arrays with 8-bit precision for gray scale [7]. We label initial three frames to be a member of the cluster δC_1 showing no expression and the rest of the frames to be class δC_2 showing some or maximum expression. From each sequence we consider one reference frame and find out the difference image for rest of the images in the sequence.

On this database PCA and KPCA are performed, and we plot $\mathbf{b}_1, \mathbf{b}_2$, and $\mathbf{b}_3$ and . The results are shown in Fig. 2.2. We can clearly see that separability is more in case of KPCA. This visual analysis is further supported when we consider the cluster analysis of the two clusters formed. The results as shown in Tab. 2.1, clealy show that the cluster measures of KPCA is much better than PCA.

Table 2.1: *Cluster Measures after PCA and KPCA of clusters δC_1 and δC_2*

A	DaviesBouldinIndex	DunnIndex	F-Measure
PCA	.9832	.3329	.8421
KPCA	.6302	.4037	.8960

Thus we have shown that the cluster separability is more in case of KPCA as

(a) PCA (b) KPCA

Figure 2.2: *Principal component space of the first three weight space of clusters* δC_1 *and* δC_2 [20]

compared to PCA. Hence we can conlcude that a non-linear clustering method may perform better in separating the clusters of facial expression images with no expression and facial expression images with some or maximum expression. This helps in deciding for a classifier for the facial expression data if required. In the next section, we try to answer the question that whether we will be able to find any emotion-specific features, that may help in representing a specific emotion.

Chapter 3

Emotion specific features

Having established that the facial expression images are better classified using non-linear classifier, we return to our question on the possibilty of a method to identify emotion specific features from the two clusters of images, one containing facial expression images with no emtion and the other containing maximum emotional content. So let the two clusters be C_1 and C_2 respectively. In this section we first elucidate on a method of extracting an emotion specific feature from C_1 and C_2 and then show that this identified feature is generalizable to any image not present in the clusters considered.

Consider we have a set of three images at different time stages of a man smiling. Let the time stages be *t1*, *t2* and *t3*, where *t1* be an image with no expression, and *t3* be an image with maximum expression. *t2* being some intermediary state. Now if we want to see the different stages or steps of changes the face underwent as it went from *t1* to *t2* to *t3*, we need to find the exact path of its change. Image differencing will not be much of use as simple subtraction and interpolation may not be enough to generate the intermediate frames. This is also due to the fact that the subsequent frames in a sequence may not be linearly aligned. Thus we need proper understanding of the direction of the path of changes that occur at each stage and also we need to be able to represent and visualize the intermediate frames.

Before we begin analyzing the changes that occur to human face as it goes through the stages from no expression to a state of maximum emotional content, we need to identify a set of features that shall help in the analysis. Naturally these set of features should minimize intra-class variation while maximizing inter-class variation. The features selected should again be such that we can represent the features in terms of facial images. This will help in visualization of any changes made to the feature vectors. Thus the changing pixel values across a particular sequence can be of importance, in two ways, one, it will help to identify the

direction of change and two it will help to represent the features in terms of input space. Before we delve further in the matter we would like to look back at methods that can help in identifying features that gives the information of the direction of change and change in pixel values.

3.1 Existing approaches related to feature selection

Optical flow analysis helps to identify motion fields in one or several frames of an image sequence and have been studied by many authors, most notably in [18], [24] and [34]. The method though depends on various external elements like lighting conditions and non-rigid motions, which was generalized in [4]. Facial geometry analysis, is used to represent the face using various shapes and location of various fiducial points. Active Shape Models (ASM) [7] is one of the most popular methods used in this category. Facial Action Coding System (FACS) [11], introduced the concept of using facial action units (AUs) being used for describing any facial activities. Notable works in AU detection include [31], [33] and [3]. More recently a fully automated AU detection algorithm [32] is used to represent the facial feature points in the initial video frames, thereby recognizing other temporal features using *AdaBoost* [12].

3.2 Criteria for feature selection

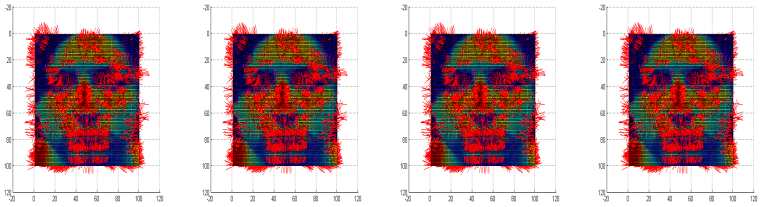
We are trying to identify the changes that occur on the surface of a human face as it goes from a state of no expression to a state of maximum expression. We intend to generalize the change captured for any facial expression image. To identify these generalized changes we need to understand what deformations occurs on the surface of a human face as it goes through the transition. To model this kind of deformation causing transforms (here, change of expression), we need a feature selection criteria that can model, one, any motion on the surface of the face with time and two, capture the change in geometry of the surface of face. We also need to understand that any changes captured on the surface of a human face has be local to the region of change. None of the methods described in the previous section can achieve the two requirements we mentioned few lines ago.

To make the situation more clear, consider a sequence of human facial expressions from a state of no expression to a state of maximum expression as shown in Fig 3.1. We can see that the surface of face changes as it goes from a state of no expression to maximum expression. So consider surface normals, as we progress from the no-expression state to maximum expression state, the normals also change as seen in Fig. 3.2. It is this change on the surface of the face that we are interested

to capture.

(a) (b) (c) (d)

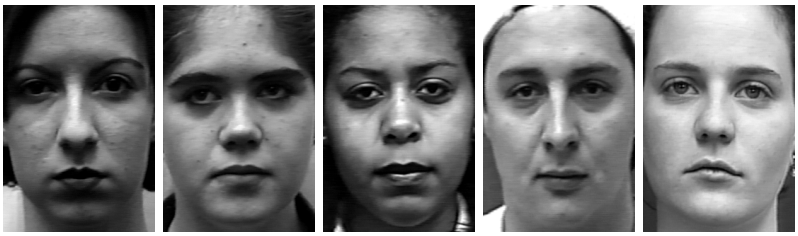
Figure 3.1: *Variation on the surface of face with change in expression from no expression to maximum expression*



(a) (b) (c) (d)

Figure 3.2: *Variation of the surface normals of faces in Fig. 3.1. The red lines are the surface normals*

Consider again the set of human facial images as shown in Fig 3.3. Here we see a great variation in the structure of each face. To make a generalized emotion-specific feature we need to reduce the information of the structure of a face.



(a) (b) (c) (d) (e)

Figure 3.3: *Variation on the structure of faces in humans*

In the next section we show that the information from these surface normals are extracted and a method to identify the regions on the surface of the face where

most activty occurs is described.

3.3 Localized feature extraction

We begin our feature extraction first by collecting all input images, and constructing surface traingulations. We consider each pixels position as points on which triangulation is to be done. We perform Delaunay triangulation [25] on the given input image to obtain the surface mesh. To each vertex of a triangle in the surface mesh we add the pixel values. This gives a 3D surface mesh for each image. Now at each vertex we calculate the unit surface normals. This is done by using concepts of Gouraud shading [14]. If more than two triangles shares the same vertex, we average the normals of each the triangles and assign the resultant vector to the pixel. This defines the surface normal at each pixel position in every image. We henceforth shall name these normal vectors as *Gouraud shading vectors*. We then try to find the pixel positions where these Gouraud shading vectors vary the most in an image sequence. The pixel positions thus obtained are called *active pixels*. To obtain active pixels in an image sequence, we first find a mean representation of each Gouraud shading vectors in a sequence of images and then the standard deviation.
Let there be r sequence of images and p pixels in an image in each of the r sequences. Each of the images is converted to vector of size p. Let $\mathcal{I}$ be the input space, then $\mathcal{I} \subseteq \mathbb{R}^p$. Let there be q images in each sequence. Let the Gouraud shading vectors of the i^{th} pixel be $\mathbf{Y}_i$, such that $\mathbf{Y}_i = \{\mathbf{y}_{i1}, \mathbf{y}_{i2}, ..., \mathbf{y}_{iq}\}$ where $\mathbf{y}_{ij}$ is the Gouraud shading vector of the i^{th} pixel of the j^{th} frame. Let μ_i be the mean and σ_i be the standard deviation of the Gouraud shading vectors of the i^{th} pixel in a sequence of images. Let k be any number such that $k \in \mathbb{R}$. For each $\mathbf{Y}_i$, we calculate the cosine of the angle between $\mu_i + k\sigma_i$ and $\mu_i - k\sigma_i$. Let the cosine of the vectors of i^{th} pixel be θ_i. θ_i would give an estimate of the angle between $\mu_i \pm k\sigma_i$. This would help to understand how much the Gouraud shading vector varies in a given sequence. The smaller the cosine value, the greater is the angle between the vector, hence the greater is the variance. Therefore, we rank the pixels on the basis of increasing θ_i value. Let m be such that $m \in \mathbb{N}$ and $m \leq p$. We select m pixel positions from each of the sequence of images. For the l^{th} sequence, we store the selected m pixel positions in a vector, $\mathbf{V}_l$. Thus, for r sequences, we have a set of r $\mathbf{V}_l$ vectors. We select the pixel positions that occur most frequently in each of the $\mathbf{V}_l$ vectors. The selection is done as follows. For each occurance of j^{th} pixel in l^{th} sequence, we assign a single vote to the j^{th} pixel. Therefore, each j^{th} pixel can get a minimum of 0 votes and a maximum of r votes. We find the mean, μ, and standard deviation, σ, of the votes. Let t be any number such that $t \in \mathbb{R}$. We select all those pixels which have at least $\mu + t\sigma$ votes. We call these pixels as

active pixels. A example of the active pixels identified from the surface mesh can be seen in Fig. 3.4 and in Fig. 3.5(c).

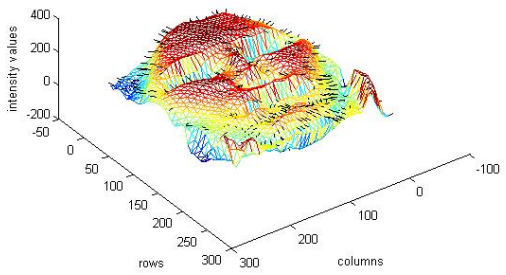
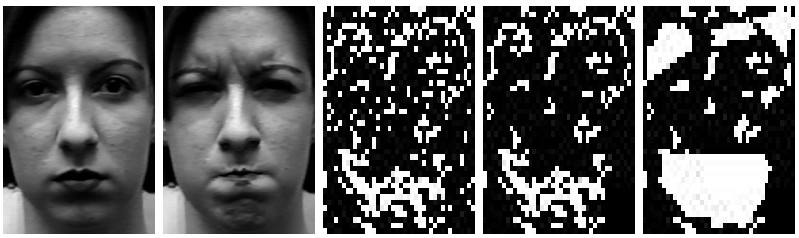


Figure 3.4: *The colored lines form the surface mesh of a given shape, and the black vectors form the unit Gouraud shading normals from the active pixels on the face (this is before active region identification)*

To ensure that the changes made as a result of altering the pixel values have a smooth effect on a region of an image, we segment the image on the basis of density of the active pixel. The segmented regions are called *active regions*, and are identified with the help of the active pixels identified in the previous step. We find 4-connectivity of each of the active pixels. We then remove all the active pixels which have no 4-connected neighbours. This leads to removal of lone pixels with no neighbours, as seen in Fig. 3.5(d). We say two pixels are connected if, one, they are 4-way connected, and two, one pixel's 4-connected neighbour is 4-connected to another pixel. We thus have different sets of connected pixels. We take each set and construct its convex hull [10]. Let the region identifed by the v^{th} convex hull be $\boldsymbol{\kappa}_v$. We consider every pixel inside the convex hull $\boldsymbol{\kappa}_v$ to be the new set of active pixels. The regions thus identified are called active regions, as can be seen in Fig. 3.5(e).

We construct the feature vectors as pixel values taken from the active pixels identified in the previous step. The feature vectors do not form a linear vector space, and lie on a space of higher dimensional manifold. The active regions identified in the previous step ensures that we have information of the pixels where changes due to expressions occur the most. In other words, they help in idenfing *local* deformations and minimizes the effect of any *global* change. Let the number of pixels selected as active pixels be n. Let $k \in \mathbb{N}$, and $k \leq n$, such that for all values of k, $\mathbf{x}_k \in \mathbb{R}^n$ are the feature vectors. We denote the space spanned by the



(a) (b) (c) (d) (e)

Figure 3.5: *The identification of active pixels and active regions from images (a) Image with no expression, (b) image with maximum emotional content (here, anger) (c) the active pixels after removing pixels with no 4-connected neighbours, (d) the active regions after creating convex hull of the 4-connected components, and (e) the white pixels are the identified active pixels from the image sequence*

feature vectors as $\mathcal{X}$. An obvious conclusion is $\mathcal{X} \subseteq \mathbb{R}^n$.

In the next section we use the active regions identified in this section as feature vectors specific to one emotion, and use this to understand the differences between the two class of no expression and with expression facial images. We also show that understanding the differences shall help in synthesizing emotions.

Chapter 4

Emotion Synthesis

Once we extract the feature vectors from the input images, we return to the question of what we intend to do with these features. We want to use these emotion specific features to deform a non-expression image to an image with some-or maximum emotional content. As such we want to generate the in-between frames. To do this we need to know the differences between the two cluster of images, where one cluster, say C_1, consist of features extracted from images with no-expression and the other cluster, say C_2, consist of images with some or maximum emotion content pertaining to one specific expression (say, anger). To do this, we need to statistically analyze the clusters. Here we construct a classifier to distinguish between two classes, one, face images with no expression, and two, face images with maximum emotional content. There are many classifers that can used to classify the two clusters. We use SVM learning algorithm [5], [6] to estimate an optimal classifier. The classifier function constructed as a result implicitly encodes the differences between the two classes. It is these differences that we are trying to understand in our work. If the difference can be interpreted in terms of the input space, then we shall be able to visualize them. So before we delve further into how this can be done, we study some basic theories of SVM.

4.1 Background: Support Vector Machine

Let $\mathbf{x}_k \in \mathcal{X}$ where $\mathcal{X} \subseteq \mathbb{R}^n$, be obeservation vectors (can also be referred to as feature vectors), and $y_k \in \{-1, 1\}$ be corresponding class labels. Let Euclidean inner product be defined in $\mathcal{X}$. Let there exist l such observation vectors. SVM learning algorithms helps to estimate a linear optimal classifier from the given observation vectors. Let $\mathbf{w}$ be the projection vector that maximizes the distance between two classes. Let $\frac{b}{\|\mathbf{w}\|}$ be the offset of the classifier from the origin, and

$\mathbf{x} \in \mathcal{X}$. The classifier can then be represented as,

$$f(\mathbf{x}) = \langle \mathbf{x} \ . \ \mathbf{w} \rangle + \ b. \tag{4.1}$$

It maximizes the margin between the two classes with respect to the separating hyperplane. It has been shown that the projection vector that maximizes the distance between the two classes can be represented as a linear combination of the vectors in the feature space. Let $m \in \mathbb{N}$, such that $m \leq l$. Let $\alpha_m \in [0, \infty)$ be coefficients constructed as a result of constrained quadriatic optimization to solve for $\mathbf{w}$, and $\mathbf{w}^*$ be the optimized value of $\mathbf{w}$, then $\mathbf{w}^*$ can be represented as,

$$\mathbf{w}^* = \sum_{m=1}^{l} \alpha_m y_m \mathbf{x}_k. \tag{4.2}$$

If any m^{th} vector has a non-zero α_m, it is termed as *supprot vector*, as it is these vectors that define the boundary of each class. Thus the classifier function becomes,

$$f(\mathbf{x}) = \sum_{m=1}^{l} \alpha_m y_m \langle \mathbf{x} \ . \ \mathbf{x}_m \rangle + \ b. \tag{4.3}$$

A visual depiction of an SVM can be seen in Fig. 4.1.

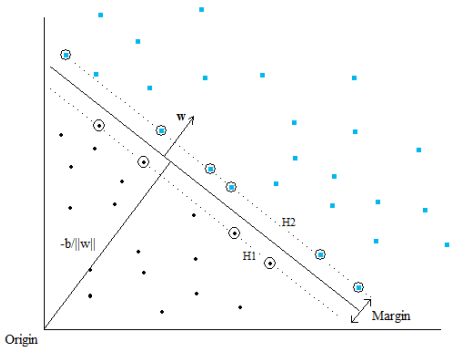


Figure 4.1: *The figure shows a typical case of a linear SVM. The black dots represent one class and the blue dots represent another class. The black bold line separating the two classes is the classifier.* $\mathbf{w}$ *is the normal to the classifier.* $\frac{-b}{\|\mathbf{w}\|}$ *is the distance of classifier from the origin. The points on the dotted line represent the support vectors of each class. The distance between the two doted lines is the margin of separation between the two classes*

Naturally this method is not applicable to all problems, as there exists non-linear problems. To solve these non-linear problems we need a special trick.

4.1.1 Non-linear analysis

Let $\mathbf{x}_i$, $\mathbf{x}_j \in \mathbb{R}^p$ and k be a similarity measure defined as,

$$k : \mathcal{I} \times \mathcal{I} \mapsto \mathbb{R}, (\mathbf{x}_i, \mathbf{x}_j) \mapsto k(\mathbf{x}_i, \mathbf{x}_j). \tag{4.4}$$

The function k is called a kernel [23]. Let K be a $p \times p$ matrix, such that for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{I}$, $K \coloneqq (k(\mathbf{x}_i, \mathbf{x}_j))_{ij}$. A $p \times p$ matrix K over a set of complex numbers defined as $\{c_1, c_2, ..., c_p\} \in \mathbb{C}$ satisfying

$$\sum_{i=1}^{p} \sum_{j=1}^{p} c_i \hat{c_j} K_{ij} \geq 0, \tag{4.5}$$

where $\hat{c_j}$ is the complex conjugate of c_j, is called a positive definite matrix.

Let $\mathcal{I}$ be a non-empty set. Any function k, defined as $k : \mathcal{I} \times \mathcal{I} \mapsto \mathbb{C}$ for which, all $m \in \mathbb{N}$, $\{\mathbf{x}_1, ..., \mathbf{x}_m\} \in \mathcal{I}$, results in a positive definite gram matrix is called a positive definite kernel. Real valued kernels are included in the previous definition as a special case. But c_i necessarily need not be real. To obtain real coefficients c_i, K needs to be a symmetric matrix, *i. e.*, $K_{ij} = K_{ji}$.

The Moore-Aronszajn theorem [1] states that for every positive definite kernel, there exists a unique reproducing kernel Hilbert space (RKHS), and vice-versa. Let H be the RKHS associated withy k, such that H is endowed with the inner product.

Let a function be defined in H as:

$$\phi : \mathcal{I} \mapsto H, \qquad x \mapsto k(\cdot, \mathbf{x}). \tag{4.6}$$

Thus $\phi(\mathbf{x}) = k(\cdot, \mathbf{x})$ is the function that assigns the value $k(\mathbf{x}_i, \mathbf{x})$ to $\mathbf{x}_i \in \mathcal{I}$. In other words, the function ϕ helps in representing $\mathbf{x}$ in terms of $\mathbf{x}$'s similarity to *all* other points in the input space $\mathcal{I}$. Since inner product is defined in H, any fucntion $\psi(\cdot)$ in H can be evaluated at any $\mathbf{x}_i \in \mathcal{I}$ with,

$$\psi(\mathbf{x}_i) = \langle \psi(\cdot), k(\cdot, \mathbf{x}_i) \rangle, \tag{4.7}$$

i. e., k is the *representer of evaluation*. The rich representation of the input space provided by ϕ, comes with the cost of high computational requirements. To counter this, the definition of eq. 4.7 provides for an efficient kernel trick if $\psi(\cdot)$ is replaced with $k(\cdot, \mathbf{x}_i)$. This is given as,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle k(\cdot, \mathbf{x}_i), k(\cdot, \mathbf{x}_j) \rangle, \tag{4.8}$$

for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{I}$. This is called the *reproducing kernel property*. Thus $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i) \rangle$. This is the *kernel trick*. Following the kernel trick, representer theorem [21] provides powerful theoritical foundations for non-linear problems in SVMs and kernel-PCA. It was later generalized in [28].

4.1.2 Solving the non-linear problem

Let us assume the observation vectors $\mathbf{x}_k$ are not linearly separable. So we project $\mathbf{x}_k$ on a higher dimension space $\mathbb{F}$, with the help of a kernel function ϕ. Therefore ϕ is defined as:

$$\phi : \mathcal{I} \mapsto \mathbb{F}, \qquad x \mapsto k(\cdot, \mathbf{x}). \tag{4.9}$$

Since ϕ is a kernel function, we can construct a kernel matrix $\mathbf{K}$ defined as:

$$\mathbf{K}(\mathbf{x}_i,\ \mathbf{x}_j) = \ \langle \phi(\mathbf{x}_i)\ .\ \phi(\mathbf{x}_j) \rangle \quad \forall \mathbf{x}_i, \mathbf{x}_j\ \in\ \mathbb{R}^n \tag{4.10}$$

Let f_K be the non-linear classifier function analogous to f of eq. 4.1. The equation of the classifier for a kernel mapping to a higher dimensional space, is

$$f_K(\mathbf{x}) = \sum_{k=1}^{l} \alpha_k y_k \mathbf{K}(\mathbf{x}\ ,\ \mathbf{x}_k) +\ b \tag{4.11}$$

Thus the projection vector becomes

$$\mathbf{w} = \sum_{k=1}^{l} \alpha_k y_k \phi(\mathbf{x}_k). \tag{4.12}$$

Before we proceed further we also state how the kernel function behaves when the function is i) Linear, and ii) RBF. For linear kernels $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i . \mathbf{x}_j \rangle$, that is the mapping ϕ is identity. For RBF kernels we have $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_j - \mathbf{x}_j\|^2/\gamma}$ where γ determines the width of the kernel.

4.2 Works related to detection of local deformation

We have shown in chapter two, that the clusters formed by facial expression images are better seprated using non-linear methods. But as in all non-linear problems we are presented with the complication of solving the pre-image problem. This arises due to the fact that any small deformation done to a face with no-expression will project the feature vector in the direction of the cluster of feature vectors from images with some expression. In fact in a linear situation an ideal direction will be the normal to the classifier. Thus we need to first understand the differences between the two classes and then identify the normal to the classifier and then use it to project a feature vector form one class to another.

In this section we identify the statistical differences between the two classes by Golland *et. al.* [13]. SVM's learning algorithm is used to estimate the optimal classifier. The classifier characteristics allow us the opportunity to identify differences between the populations. Thus, as we move the vectors from one class

to another we are able to identify changes which when expressed in terms of the original image gives us better insight to these differences. The concept of *discriminative direction* is introduced here, which identifies a specific direction to move from every point in one class in the feature space $\mathcal{I}$ to another class. In other words, this discriminative direction helps us change any input sample to make it look like a member of the other class.

4.2.1 Discriminative direction

The discriminative direction principally gives us that direction in which the least change in a feature vector leads it to be identified as a member of the other class. Ideally this change is along the direction of the normal to the classifier constructed due to the support vector machine. If we consider kernel mapping for non-linear classifiers, we cannot always move along this normal in the feature space. This is caused since we do not have any access to the kernel space. Thus we search in the feature space a direction across the classifier which minimizes the divergence of the projection from the classifier.

We now return to determining the discriminative direction in the feature space with respect to the discriminative direction in the higher dimensional space. We note that ideally to move from one class to the another class we need to move in the direction of the normal to the classifier, *i.e.*, $\mathbf{w}$. So we take the projection of the feature vector along $\mathbf{w}$.

Let $\mathbf{x}$ be a point in the feature space $\mathcal{I}$. Thus formally consider $\mathbf{x}$ to be moved to $\mathbf{x} + \mathrm{d}\mathbf{x}$ in the feature space, the image in $\mathbb{F}$ changes by,

$$d\mathbf{z} = \phi(\mathbf{x} + d\mathbf{x}) - \phi(\mathbf{x}). \tag{4.13}$$

Thus the deviation form $\mathbf{w}$ can be computed as

$$\mathbf{e} = d\mathbf{z} - \frac{\langle d\mathbf{z} \,.\, \mathbf{w}\rangle}{\|\mathbf{w}\|}\frac{\mathbf{w}}{\|\mathbf{w}\|} = d\mathbf{z} - \frac{\langle d\mathbf{z} \,.\, \mathbf{w}\rangle}{\langle \mathbf{w}.\mathbf{w}\rangle}\mathbf{w}, \tag{4.14}$$

Squaring this error, we get

$$\|\mathbf{e}\|^2 = \langle d\mathbf{z} \,.\, d\mathbf{z}\rangle - \frac{\langle d\mathbf{z} \,.\, \mathbf{w}\rangle^2}{\langle \mathbf{w} \,.\, \mathbf{w}\rangle} \tag{4.15}$$

$\mathbf{e}^2$ is expressed as a function of $\mathrm{d}\mathbf{x}$, where $\mathbf{e}^2$ is the error in the kernel space and the mapping is thus defined as $E(d\mathbf{x}) = \mathbf{e}^2$. We thus minimize this error in the kernel space and thus obtain the minimum possible error in the feature space. Thus the optimization problem can be now defined as:

$$\text{minimize } E(d\mathbf{x}) = \|\mathbf{e}\|^2 = \langle d\mathbf{z} \,.\, d\mathbf{z}\rangle - \frac{\langle d\mathbf{z} \,.\, \mathbf{w}\rangle^2}{\langle \mathbf{w} \,.\, \mathbf{w}\rangle} \tag{4.16}$$

$$s.t. \;\; \|d\mathbf{x}\|^2 \;=\; \epsilon \tag{4.17}$$

Now if we look at the optimization problem and consider the definition of the kernel function, we immediately realize how the kernel trick is going to work for us. Thus computing the dot productcs in the kernel space we have:

$$\langle d\mathbf{z} \;.\; \mathbf{w}\rangle \;=\; \langle\{\phi(\mathbf{x} \;+\; d\mathbf{x}) \;-\; \phi(\mathbf{x})\}.\{\mathbf{w}\}\rangle$$

$$\text{or,} \;\; \langle d\mathbf{z} \;.\; \mathbf{w}\rangle \;=\; \langle\phi(\mathbf{x} \;+\; d\mathbf{x}) \;.\; \mathbf{w}\rangle \;-\; \langle\phi(\mathbf{x}) \;.\; \mathbf{w}\rangle$$

$$\text{or,} \; \langle d\mathbf{z}.\mathbf{w}\rangle = \sum_{k=1}^{l} \alpha_k y_k \{\langle\phi(\mathbf{x} \;+\; d\mathbf{x}).\phi(\mathbf{x}_k)\rangle \;-\; \langle\phi(\mathbf{x}).\phi(\mathbf{x}_k)\rangle\}$$

$$\text{or,} \;\; \langle d\mathbf{z} \;.\; \mathbf{w}\rangle \;=\; \sum_{k=1}^{l} \alpha_k y_k (\mathbf{K}(\mathbf{x} \;+\; d\mathbf{x},\; \mathbf{x}_k) \;-\; \mathbf{K}(\mathbf{x} \;,\; \mathbf{x}_k))$$

$$\text{or,} \;\; \langle d\mathbf{z} \;.\; \mathbf{w}\rangle \;=\; \sum_{k=1}^{l} \alpha_k y_k \sum_i \left.\frac{\partial \mathbf{K}(\mathbf{u},\; \mathbf{v})}{\partial \mathbf{u}_i}\right|_{\substack{\mathbf{u}=\mathbf{x} \\ \mathbf{v}=\mathbf{x}_k}} d\mathbf{x}_i$$

$$\text{or,} \;\; \langle d\mathbf{z} \;.\; \mathbf{w}\rangle \;=\; \nabla f_K(\mathbf{x}) \;.\; d\mathbf{x} \tag{4.18}$$

where d$\mathbf{x}$ is the gradient of the feature vector presently considered and $\nabla f_K(\mathbf{x})$ is the gradient of the classifier function evaluated at $\mathbf{x}$. If we try to geometrically analyze the dot product we find that $\nabla f_K(\mathbf{x})$ helps us get the direction in which the feature vector needs to be ideally displaced in the feature space to produce the instantaneous change required for it to be classified as a member of the other class.

Computing $\langle d\mathbf{z} \;.\; d\mathbf{z}\rangle$ in present feature space we have:

$$\langle d\mathbf{z} \;.\; d\mathbf{z}\rangle = \langle(\phi(\mathbf{x} \;+\; d\mathbf{x}) \;-\; \phi(\mathbf{x})).(\phi(\mathbf{x} \;+\; d\mathbf{x}) \;-\; \phi(\mathbf{x}))\rangle$$

$$\text{or,} \langle d\mathbf{z} \;.\; d\mathbf{z}\rangle = \langle(\phi(\mathbf{x}+d\mathbf{x}).\phi(\mathbf{x}+d\mathbf{x})\rangle - 2.\langle\phi(\mathbf{x}+d\mathbf{x}).\phi(\mathbf{x})\rangle \;+\; \langle\phi(\mathbf{x}) \;.\; \phi(\mathbf{x})\rangle$$

$$\text{or,} \; \langle d\mathbf{z} \;.\; d\mathbf{z}\rangle = \mathbf{K}(\mathbf{x}+d\mathbf{x}, \mathbf{x}+d\mathbf{x}) \;-\; 2\mathbf{K}(\mathbf{x}+d\mathbf{x}, \mathbf{x}) \;+\; \mathbf{K}(\mathbf{x}, \mathbf{x})$$

$$\text{or,} \quad \langle d\mathbf{z} \;.\; d\mathbf{z}\rangle \;=\; \left.\frac{\partial^2 \mathbf{K}(\mathbf{u} \;,\; \mathbf{v})}{\partial \mathbf{u}_i \partial \mathbf{v}_i}\right|_{\substack{\mathbf{u}=\mathbf{x} \\ \mathbf{v}=\mathbf{x}_k}} d\mathbf{x}_i d\mathbf{x}_j$$

$$\text{or,} \quad \langle d\mathbf{z} \;.\; d\mathbf{z}\rangle \;=\; d\mathbf{x}^T H_{\mathbf{K}}(\mathbf{x}) d\mathbf{x} \tag{4.19}$$

The Hessian taken on a small (local) region on the surface of a manifold gives the information about the direction in which the normal to the surface change at fastest rate. The fastest changing normal is termed as the principal direction of that local region. In other words, the Hessian gives us the information of the pricipal direction of a local region. Therefore equating the Hessian of the classifier

function at $\mathbf{x}$ we get the direction in which the normal from the surface of the manifold, in which the feature vectors are defined, changes at the fastest rate. Any movement in this direction shall cause that change which causes the feature vector to be classified as a member of the other class. Hence naturally any vector in the feature space tends to move in this direction towards the classifier, and thereby causing the error.

The optimization problem now becomes:

$$\text{minimize } E(d\mathbf{x}) = d\mathbf{x}^T(H_K(\mathbf{x}) - \|\mathbf{w}\|^{-2}\nabla f_K^T(\mathbf{x})\nabla f_K(\mathbf{x}))d\mathbf{x} \tag{4.20}$$

$$\text{s.t. } \|d\mathbf{x}\|^2 = \epsilon \tag{4.21}$$

The solution to the above problem is the smallest eigenvector of the matrix

$$Q_K(\mathbf{x}) = H_K(\mathbf{x}) - \|\mathbf{w}\|^{-2}\nabla f_K^T(\mathbf{x})\nabla f_K(\mathbf{x}) \tag{4.22}$$

If $H_K(\mathbf{x}) = cI$, then the smallest eigenvector of the matrix $Q_K(\mathbf{x})$, which is same as the largest eigenvector of $\nabla f_K^T(\mathbf{x})\nabla f_K(\mathbf{x})$. The largest eigenvector of $\nabla f_K^T(\mathbf{x})\nabla f_K(\mathbf{x})$ is $\nabla f_K^T(\mathbf{x})$, which is nothing but the gradient of the classifier function. We note that the rank of $\|\mathbf{w}\|^{-2}\nabla f_K^T(\mathbf{x})\nabla f_K(\mathbf{x})$ is unity and the non-zero eigenvalue is equal to $\|\mathbf{w}\|^{-2}\|\nabla f_K(\mathbf{x})\|^2$ corresponding to the eigenvector $\nabla f_K^T(\mathbf{x})$. We would now see that such a form of $H_K(\mathbf{x})$ would help us achieve an analytical solution of the above mentioned minimization problem.

The linear kernel is a dot product kernel, *i.e.*, $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = k(\langle \mathbf{x}_i, \mathbf{x}_j \rangle)$. Thus

$$\left.\frac{\partial^2 \mathbf{K}(\mathbf{u}\ ,\ \mathbf{v})}{\partial \mathbf{u}_i \partial \mathbf{v}_i}\right|_{\substack{\mathbf{u}=\mathbf{x} \\ \mathbf{v}=\mathbf{x}_k}} = k'(\|\mathbf{x}\|^2)\delta_{ij} + k''(\|\mathbf{x}\|^2)\mathbf{x}_i\mathbf{x}_j \tag{4.23}$$

Since we consider linear kernels here $k''(\|\mathbf{x}\|^2) = 0$, we have $H_K(\mathbf{x}) = cI$ for all $\mathbf{x}$. For linear kernel classifiers, $\|\nabla f_K(\mathbf{x})\| = \|\mathbf{w}\|$, and thus $c = 1$. Hence we have, $H_K(\mathbf{x}) = I$ and the discriminative direction is defined as

$$d\mathbf{x}^* = \nabla f_K^T(\mathbf{x}) = \sum_k \alpha_k y_k \mathbf{x}_k = \mathbf{w} \tag{4.24}$$

$$E(d\mathbf{x}^*) = H_K(\mathbf{x}) - \|\mathbf{w}\|^{-2}\|\nabla f_K(\mathbf{x})\|^2 = 0 \tag{4.25}$$

The above result conforms to our intuition that we shall be able to move ideally along the direction of the normal to the separating hyperplane for linear kernels.

The Gaussian kernel is a distance kernel, *i.e.*, $\mathbf{K}(\mathbf{u},\mathbf{v}) = \mathrm{k}(\|\mathbf{u} - \mathbf{v}\|^2)$. For distance kernels,

$$\left.\frac{\partial^2 \mathbf{K}(\mathbf{u}\ ,\ \mathbf{v})}{\partial \mathbf{u}_i \partial \mathbf{v}_i}\right|_{\substack{\mathbf{u}=\mathbf{x} \\ \mathbf{v}=\mathbf{x}_k}} = -2k'(0)\delta_{i,j} \tag{4.26}$$

For Gaussian kernels, $k'(0) = -\frac{1}{\gamma}$. The discriminative direction is thus,

$$d\mathbf{x}^* = \nabla f_K^T(\mathbf{x}) = -\frac{2}{\gamma}\sum_k \alpha_k y_k \mathbf{K}(\mathbf{x}, \mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) \tag{4.27}$$

$$E(d\mathbf{x}^*) = \frac{2}{\gamma} - \|\mathbf{w}\|^{-2}\|\nabla f_K(\mathbf{x})\|^2 \tag{4.28}$$

It should be observed that the error here is non-zero, since all vectors in the kernel space masy not have a pre-image in the feature space. This restricts us to move ideally in the direction of the discriminative direction.

The method though suffers from the fact that the feature space is not differentiable. As such the generated pre-images of the deformed vector fails to comply to the distribution of the feature set. To counter this short coming L. Zhou *et.al.* in their work in [35] added the constrained that the pre-image constructed should comply with the distribution of the feature vector to be deformed in the input space $\mathcal{X}$. In the next subsection we study the methods of doing this.

4.2.2 Regularized discriminative direction

L.Zhou *et.al.* [35], built on the short comings in the previous theorem. They constructed the discriminative direction using information of the local region around a point in the feature space. The idea was to deform the input features in such a way so that the reconstructed image conformed to the distribution of the feature space. The direction in which the featurue points were deformed is called *regularizing discriminative direction*, and it is constructed using the distribution of the neighborhood of a point in the feature space. In this section we shall elaborate on the methods involved in the development of the theory.

Considering the same feature space as given in the previous section, let, $\mathbf{w}$ be the projection vector that maximizes the distance between two classes. Let $\hat{\mathbf{x}} \in \mathcal{X}$ be the vector which is to be deformed so that it will be classified as a member of the other class and $\phi(\cdot)$ be the function that projects $\hat{\mathbf{x}}$ to a higher dimensional kernel space. So the higher dimensional vector corresponding to $\hat{\mathbf{x}}$ is $\phi(\hat{\mathbf{x}})$. Moving $\phi(\hat{\mathbf{x}})$ along $\mathbf{w}$ for $\mathbf{s}$ steps leads to $\phi(\hat{\mathbf{x}}) + \mathbf{sw}$.

Let $\mathbf{z} \in \mathcal{X}$ such that $\mathbf{z} \xleftarrow{\phi^{-1}} \phi(\hat{\mathbf{x}}) + \mathbf{sw}$ where $\mathbf{z}$ is the best estimate of the pre-image of $\phi(\hat{\mathbf{x}})+\mathbf{sw}$. Therefore, the *discriminative direction* is $\mathbf{z}-\hat{\mathbf{x}}$. To estimate the pre-image of the discriminative direction, the error due to reconstruction should be minimized. We call this error as the *residual error*, and is represented by $\rho(\mathbf{z})$. From definition $\rho(\mathbf{z})$ can be represented as,

$$\rho(\mathbf{z}) = \|\phi(\hat{\mathbf{x}}) + \mathbf{sw} - \phi(\mathbf{z})\|^2. \tag{4.29}$$

The eq. 4.29 should be minimized with respect to $\mathbf{z}$ to get an optimal representation of $\mathbf{z}$. We name this optimal representation of $\mathbf{z}$ as $\mathbf{z}^*$, and it can be defined as,

$$\mathbf{z}^* = arg \min_{\mathbf{z}\in\mathbb{R}^d} \rho(\mathbf{z}),$$

$$or, \quad \mathbf{z}^* = arg \min_{\mathbf{z}\in\mathbb{R}^d} \|\phi(\hat{\mathbf{x}}) + \mathbf{sw} - \phi(\mathbf{z})\|^2. \tag{4.30}$$

Now

$$\rho(\mathbf{z}) = \langle \phi(\hat{\mathbf{x}}) + \mathbf{sw}, \phi(\hat{\mathbf{x}}) + \mathbf{sw} \rangle + \langle \phi(\mathbf{z}), \phi(\mathbf{z}) \rangle - 2\langle \phi(\hat{\mathbf{x}}) + \mathbf{sw}, \phi(\mathbf{z}) \rangle, \tag{4.31}$$

where $\langle \phi(\hat{\mathbf{x}}) + \mathbf{sw}, \phi(\hat{\mathbf{x}}) + \mathbf{sw} \rangle$ is constant with respect to $\mathbf{z}$. Considering Gaussian kernels $\langle \phi(\mathbf{z}), \phi(\mathbf{z}) \rangle$ is also constant. So, eq. 4.31 can be reduced to:

$$\rho(\mathbf{z}) = -2\langle \phi(\hat{\mathbf{x}}) + \mathbf{sw}, \phi(\mathbf{z}) \rangle + c \tag{4.32}$$

where c is any constant in $\mathbb{R}$. Thus the minimization function of eq. 4.30 can now be reformularized as:

$$\mathbf{z}^* = arg \max_{\mathbf{z}\in\mathbb{R}^d} 2\langle \phi(\hat{\mathbf{x}}) + \mathbf{sw}, \phi(\mathbf{z}) \rangle \tag{4.33}$$

Before we move further, we would like to look into the works of Rathi [26] and Kwok [22].

Rathi [26] Review

Let the distance between two vectors in the feature space be d, and the distance between their corresponding representations in the kernel space be $d_{\mathcal{F}}$. We already know $\mathbf{w} = \sum_i \alpha_i \phi(\mathbf{x}_i)$ and lies in the higher dimensional kernel space spanned by the training points. To minimize $\rho(\mathbf{z})$, we set the derivative with respect to $\mathbf{z}$ to zero and rearrange. For RBF kernel:

$$\mathbf{z} = \frac{\mathbf{K}(\hat{\mathbf{x}}, \mathbf{z})\hat{\mathbf{x}} + \mathbf{s}\sum_{i=1}^{n} \alpha_i \mathbf{K}(\mathbf{x}_i, \mathbf{z})\mathbf{x}_i}{\mathbf{K}(\hat{\mathbf{x}}, \mathbf{z}) + \mathbf{s}\sum_{i=1}^{n} \alpha_i \mathbf{K}(\mathbf{x}_i, \mathbf{z})} \tag{4.34}$$

Assuming $\langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle = 1 \ \ \forall \mathbf{x}$, we have

$$\mathbf{K}(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \ = \ 1 - \frac{1}{2}\|\phi(\mathbf{x}) - \phi(\mathbf{y})\|^2$$

Therefore:

$$\mathbf{z} = \frac{(2 - \|\phi(\hat{\mathbf{x}}) - \phi(\mathbf{z})\|^2)\hat{\mathbf{x}} + \mathbf{s}\sum_{i=1}^{n} \alpha_i (2 - \|\phi(\mathbf{x}_i) - \phi(\mathbf{z})\|^2)\mathbf{x}_i}{(2 - \|\phi(\hat{\mathbf{x}}) - \phi(\mathbf{z})\|^2) + \mathbf{s}\sum_{i=1}^{n} \alpha_i (2 - \|\phi(\mathbf{x}_i) - \phi(\mathbf{z})\|^2} \tag{4.35}$$

for $\rho(\mathbf{z}) = 0$, $\phi(\mathbf{z}) = \phi(\hat{\mathbf{x}}) + \mathbf{sw}$, and $\|\mathbf{w}\| = 1$. Therefore eq. 4.35 reduces to:

$$\mathbf{z}^* = \frac{(2-\mathbf{s}^2)\hat{\mathbf{x}} + \mathbf{s}\sum_{i=1}^{n}\alpha_i(2 - \|\phi(\mathbf{x}_i) - (\phi(\hat{\mathbf{x}}) + \mathbf{sw})\|^2)\mathbf{x}_i}{(2-\mathbf{s}^2) + \mathbf{s}\sum_{i=1}^{n}\alpha_i(2 - \|\phi(\mathbf{x}_i) - (\phi(\hat{\mathbf{x}}) + \mathbf{sw})\|^2)} \tag{4.36}$$

Now we move on to express $\|\phi(\mathbf{x}_i) - (\phi(\hat{\mathbf{x}}) + \mathbf{sw})\|^2$ in terms of the kernel matrix.

$$\begin{aligned}
&\|\phi(\mathbf{x}_i) - (\phi(\hat{\mathbf{x}}) + \mathbf{sw})\|^2 \\
&= \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_i)\rangle + \langle\phi(\hat{\mathbf{x}}), \phi(\hat{\mathbf{x}})\rangle + \mathbf{s}^2\langle\mathbf{w}, \mathbf{w}\rangle + 2\mathbf{s}^2\langle\phi(\hat{\mathbf{x}}), \mathbf{w}\rangle \\
&\quad -2\langle\phi(\mathbf{x}_i), \phi(\hat{\mathbf{x}})\rangle - 2\langle\phi(\mathbf{x}_i), \mathbf{w}\rangle \\
&= 1 + 1 + \mathbf{s}^2 + 2\mathbf{s}\sum_{j=1}^{n}\alpha_j\mathbf{K}(\mathbf{x}_j, \hat{\mathbf{x}}) - 2\mathbf{K}(\mathbf{x}_i, \hat{\mathbf{x}}) - 2\mathbf{s}\sum_{j=1}^{2}\alpha_j\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) \\
&= 2 + \mathbf{s}^2 + 2\mathbf{s}\boldsymbol{\alpha}^T\mathbf{K}_{\hat{\mathbf{x}}} - 2\mathbf{K}(\mathbf{x}_i, \hat{\mathbf{x}}) - 2\mathbf{s}\boldsymbol{\alpha}^T K_{\mathbf{x}_i}
\end{aligned} \tag{4.37}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^T$, $\mathbf{K}_{\hat{\mathbf{x}}} = (\mathbf{K}(\hat{\mathbf{x}}, \mathbf{x}_1), \ldots, \mathbf{K}(\hat{\mathbf{x}}, \mathbf{x}_n))^T$ and $\mathbf{K}_{\mathbf{x}_i} = (\mathbf{K}(\mathbf{x}_i, \mathbf{x}_1), \ldots, \mathbf{K}(\mathbf{x}_i, \mathbf{x}_n))^T$.

Kwok's [22] Method

This method tries to find a relationship between distances in the feature space $\mathcal{X}$ and the kernel space. To do so, let us define $\phi(\hat{\mathbf{x}}) + \mathbf{sw} = \mathbf{z}_{\mathcal{F}}$. Now, we try to find $\mathbf{x}_i$ in feature space, such that $\phi(\mathbf{x}_i)$ and $\mathbf{z}_{\mathcal{F}}$ are k nearest neighbors. Let $\mathbf{z}$ be the pre-image of $\mathbf{z}_{\mathcal{F}}$. Also define

$$d_{\mathcal{F}} = \|\phi(\mathbf{x}_i) - (\phi(\hat{\mathbf{x}}) + \mathbf{sw})\| \tag{4.38}$$

be the distance between $\phi(\mathbf{x}_i)$ and $(\phi(\hat{\mathbf{x}}) + \mathbf{sw})$ in the kernel space.

We thus need to establish a relationship between $d_{\mathcal{F}}$ and $d = \|\mathbf{x}_i - \mathbf{z}\| \in \mathbb{R}^n$ (n-dimensional real space). If ϕ maps $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ to the Gaussian kernel space, then

$$\begin{aligned}
d^2_{\mathcal{F}}(\mathbf{x}, \mathbf{y}) = \|\phi(\mathbf{x}) - \phi(\mathbf{y})\|^2 &= 2 - \mathbf{K}(\phi(\mathbf{x}) - \phi(\mathbf{y})) \\
&= 2(1 - e^{-\frac{d^2(\mathbf{x}-\mathbf{y})}{2\sigma^2}}) \\
d^2 &= -2\sigma^2 log(1 - \frac{d^2_{\mathcal{F}}}{2})
\end{aligned} \tag{4.39}$$

The pre-image $\mathbf{x}$ is computed using

$$\mathbf{z} = -U \wedge^{-1} V^T \frac{(d^2 - d_0^2)}{2} + \bar{\mathbf{x}} \tag{4.40}$$

where, $d^2 = (d_1, \ldots, d_m^2)^T$, d_0 is a vector with the $\mathbf{i}$-th entry equal to $\|\mathbf{x}_i - \bar{\mathbf{x}}\|^2$. We need to observe that $\mathrm{X} = [\mathbf{x}_1 - \bar{\mathbf{x}}, \ldots, \mathbf{x}_m - \bar{\mathbf{x}}]$, where $\bar{\mathbf{x}}$ is the mean given by $\bar{\mathbf{x}} = \frac{1}{m}\sum_i \mathbf{x}_i$ and the singular value decomposition of X is given as $X = U \wedge V^T$.

Regularized discriminative direction

Returning to the problem, we already know that

$$\mathbf{z}^* = arg \min_{\mathbf{z}\in\mathbb{R}^d} \rho(\mathbf{z})$$

To find a solution to the above minimization problem, we try to do a regularization of the output term using the local distribution. So let, $\hat{\mathbf{x}}$ be the feature vector which is to be deformed, leading to it's classification as a member of the opposite class. To do this we move $\hat{\mathbf{x}}$ for $\mathbf{s}$ steps in the direction of $\mathbf{w}$. Therefore we have

$$\mathbf{z} \xrightarrow{\phi} \phi(\hat{\mathbf{x}}) + \mathbf{sw} \tag{4.41}$$

We now try to define the neighborhood of $\hat{\mathbf{x}}$ in the feature space. Let us construct a closed ball of ϵ radius around $\hat{\mathbf{x}}$. This shall be termed as the neighborhood of $\hat{\mathbf{x}}$ and $\mathrm{N}_\epsilon(\hat{\mathbf{x}})$ be the set of vectors in the neighborhood. Therefore:

$$\mathrm{N}_\epsilon(\hat{\mathbf{x}}) = \{\mathbf{x} \mid \|\mathbf{x} - \hat{\mathbf{x}}\| \leq \epsilon\} \tag{4.42}$$

Let $\rho(\mathbf{x} \mid \mathbf{x} \in \mathrm{N}_\epsilon(\hat{\mathbf{x}}))$ be the empirical pdf of $\mathbf{x}$ in $\mathrm{N}_\epsilon(\hat{\mathbf{x}})$ estimated from n training samples in $\mathrm{N}_\epsilon(\hat{\mathbf{x}})$. We model this pdf as a normal distribution with mean $\mu = \hat{\mathbf{x}}$ and $\sigma^2 = \sum = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\mathbf{x}})(\mathbf{x}_i - \hat{\mathbf{x}})^T$, where $\mathbf{x}_i$'s are the training samples in $\mathrm{N}_\epsilon(\hat{\mathbf{x}})$. Provided that $\mathbf{s}$ is small, it is enough to ensure that $\mathbf{z}$ stays in $\mathrm{N}_\epsilon(\hat{\mathbf{x}})$. Since $\mathbf{z}$ complies with the pdf of $\hat{\mathbf{x}}$, we require that $\rho(\mathbf{z})$ to be large or, similarly $(\mathbf{z}-\mu)^T\sum^{-1}(\mathbf{z}-\mu)$ be small for $\sum$ to be of full rank. Thus the optimization problem is

$$\mathbf{z}^* = arg \min_{\mathbf{z}\in\mathbb{R}^d} \rho(\mathbf{z}) + 2\eta(\mathbf{z}-\mu)^T\sum^{-1}(\mathbf{z}-\mu) \tag{4.43}$$

where η is the regularization term. Observe that $\sum$ is the co-variance matrix of the vector in the neighborhood of $\hat{\mathbf{x}}$. Therefore, the covariance matrix can be written as:

$$\sum = \Gamma \wedge \Gamma^T$$

where Γ is the matrix of eigenvectors and $\wedge$ is the diagonal matrix of the eigenvalues. Therefore, the optimal $\mathbf{z}^*$ should be of the form

$$\mathbf{z}^* \in \{\mathbf{z} \mid \mathbf{z} = \mu + \Gamma \wedge^{\frac{1}{2}} \mathbf{u}\} \tag{4.44}$$

Combining eq. 4.43 and eq. 4.44, we can get an optimized $\mathbf{z}$ from an optimized $\mathbf{u}$, which can be derived as

$$\mathbf{u}^* = arg \min_{\mathbf{z}\in\mathbb{R}^d} \rho(\mu + \Gamma \wedge^{\frac{1}{2}} \mathbf{u}) + 2\eta\langle\mathbf{u},\mathbf{u}\rangle \tag{4.45}$$

$\mathbf{z}^*$ can be computed using eq. 4.44. Since the dimensions of the feature space is large optimizing $\mathbf{u}$ is cumbersome. A differential equation based method is thus used to directly work out $\mathbf{z}^*$ and $\mathbf{u}^*$ for a given step size $\mathbf{s}$.

Using eq. **??** and eq. 4.45, eq. 4.44 is equivalent to maximizing $\langle \phi(\hat{\mathbf{x}}) + \mathbf{sw}, \phi(\mu + \mathit{\Gamma} \wedge^{\frac{1}{2}} \mathbf{u}) \rangle - \eta \langle \mathbf{u}, \mathbf{u} \rangle$ provided $\langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle$ which is the case for Gaussian kernels as shown before. So we can rewrite the optimization function as

$$\begin{aligned} f(\mathbf{s}, \mathbf{u}) &= \langle \phi(\hat{\mathbf{x}}) + \mathbf{sw}, \phi(\mu + \mathit{\Gamma} \wedge^{\frac{1}{2}} \mathbf{u}) \rangle - \eta \langle \mathbf{u}, \mathbf{u} \rangle \\ &= K(\hat{\mathbf{x}}, \mu + \mathit{\Gamma} \wedge^{\frac{1}{2}} \mathbf{u}) + \mathbf{s} \sum_i \alpha_i K(\mathbf{x}_i, \mu + \mathit{\Gamma} \wedge^{\frac{1}{2}} \mathbf{u}) - \eta \langle \mathbf{u}, \mathbf{u} \rangle \\ &\triangleq g(\mathbf{u}) + \mathbf{s} h(\mathbf{u}) - \eta l(\mathbf{u}) \end{aligned} \tag{4.46}$$

where

$$g(\mathbf{u}) = K(\hat{\mathbf{x}}, \mu + \mathit{\Gamma} \wedge^{\frac{1}{2}} \mathbf{u})$$

$$h(\mathbf{u}) = \sum_i \alpha_i K(\mathbf{x}_i, \mu + \mathit{\Gamma} \wedge^{\frac{1}{2}} \mathbf{u})$$

and

$$l(\mathbf{u}) = \langle \mathbf{u}, \mathbf{u} \rangle$$

Thus for each $\mathbf{s}$ there exists a $\mathbf{u}^*$ for which $f(\mathbf{s}, \mathbf{u})$ is maximized. We note that if $\mathbf{s} = 0$, then $f(\mathbf{s}, \mathbf{u})$ is maximized when $\mathbf{u}^* = 0$. The optimization problem given by eq. 4.46, is not convex and thus have multiple local maxima. Thus for each $\mathbf{s}$ we try to find $\mathbf{u}$ which is a solution for the optimization problem for a given $\mathbf{s}$ (static optimization problem). We thus consider the function $\mathbf{u}^*(\mathbf{s})$ in $\mathbb{R}^d$ which tries to identify the changes occuring to $\mathbf{u}^*$ with $\mathbf{s}$ such that $\mathbf{u}^*(0) = 0$. As long as $\mathbf{u}^*(\mathbf{s})$ is continuous and differentiable we can get at least a local maxima. Approximating $f(\mathbf{s}, \mathbf{u})$ by a second order Taylor series expansion, we have:

$$\begin{aligned} f(\mathbf{s}, \mathbf{u}) \approx\ & g(\mathbf{u}_0) + \mathbf{s} h(\mathbf{u}_0) - \eta l(\mathbf{u}_0) + (\mathrm{J}_g - \mathbf{s}\mathrm{J}_h - \eta \mathrm{J}_l)(\mathbf{u} - \mathbf{u}_0) \\ & + \frac{1}{2}(\mathbf{u} - \mathbf{u}_0)^T (\mathrm{H}_g + \mathbf{s}\mathrm{H}_h - \eta \mathrm{H}_l)(\mathbf{u} - \mathbf{u}_0) \end{aligned} \tag{4.47}$$

where J and H are Jacobian and Hessian of g, h and l evaluated at $\mathbf{u}_0$, and $\mathbf{u}_0$ maximizes $f(\mathbf{s},\mathbf{u})$ when $\mathbf{s} = \mathbf{s}_0$. Thus first order derivative of f with respect to $\mathbf{u}$ vanishes at $\mathbf{u}_0$ and any other extremum $\mathbf{u}^*$.

We have $\frac{\partial f}{\partial u}|_{(\mathbf{u}_0, \mathbf{s}_0)} = 0$. Thus we get from eq. (39) $\mathrm{J}_g - \eta \mathrm{J}_l = -\mathbf{s}_0 \mathrm{J}_h$. Using this in the equation of $\frac{\partial f}{\partial u}|_{(\mathbf{u}^*, \mathbf{s})} = 0$ we have:

$$\begin{aligned} & \mathbf{s}\mathrm{J}_h + \mathrm{J}_g - \eta \mathrm{J}_l + (\mathrm{H}_g + \mathbf{s}\mathrm{H}_h - \eta \mathrm{H}_l)(\mathbf{u}^* - \mathbf{u}_0) \\ &= (\mathbf{s} - \mathbf{s}_0)\mathrm{J}_h + (\mathrm{H}_g + \mathbf{s}\mathrm{H}_h - \eta \mathrm{H}_l)(\mathbf{u}^* - \mathbf{u}_0) = 0 \end{aligned}$$

Thus we have

$$\left.\frac{d\mathbf{u}^*}{d\mathbf{s}}\right|_{\mathbf{s}=\mathbf{s}_0} = -(\mathrm{H}_g + \mathbf{s}\mathrm{H}_h - \eta\mathrm{H}_l)^{-1}\mathrm{J}_h \tag{4.48}$$

Therefore the tangent of $\mathbf{u}^*(\mathbf{s})$ at $\mathbf{s} = 0$ is:

$$\left.\frac{d\mathbf{u}^*}{d\mathbf{s}}\right|_{\mathbf{s}=0} = -(\mathrm{H}_g - \eta\mathrm{H}_l)^{-1}\mathrm{J}_h \tag{4.49}$$

It is very clear from the above that once we get the slope (tangent) of $\mathbf{u}^*(\mathbf{s})$ at different $\mathbf{s}$ values, we can iterate to obtain $\mathbf{u}^*$ for every $\mathbf{s}$ as :

$$\mathbf{u}^{*\ (t)} = \mathbf{u}^{*\ (t-1)} + \left.\frac{d\mathbf{u}^*}{d\mathbf{s}}\right|_{\mathbf{s}=0} (\mathbf{s}_t - \mathbf{s}_{t-1}) \tag{4.50}$$

We can also use this formulation to obtain the pre-image using the regularized discriminative direction as:

$$\hat{\mathbf{x}}^{(t)} = \mathbf{z}^* = \hat{\mathbf{x}}^{(t-1)} + \mathit{\Gamma}^{t-1}[\wedge^{t-1}]^{1/2}\mathbf{u}^{*\ (t)} \tag{4.51}$$

where $\mathbf{u}^{*\ (0)} = 0$, $\mathit{\Gamma}^{t-1}$ and $\wedge^{t-1}$ are estimated from $\hat{\mathbf{x}}^{(t-1)}$ and it's neighborhood. We should also note that $\hat{\mathbf{x}}^{(0)} = \hat{\mathbf{x}}$. The mean $\mu^{(t-1)}$ and covariance $\sum^{t-1}$ are estimated from the training points close to $\hat{\mathbf{x}}^{(t-1)}$ in $N_\epsilon(\hat{\mathbf{x}})$.

The iterative solution for each $\mathbf{u}^{(t)}$ is estimated using the four stage Runge-Kutta Method. This helps in supressing lower order error terms. Given any intial value problem in ordinary differential equations $y' = f(x, y)$ and $y_0 = f(x_0)$, the four stage Runge-Kutta Method attains the solution by:

$$\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\
k_3 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) \\
k_4 &= hf(x_n + h, y_n + k_3) \\
y_{n+1} &= y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 + \mathcal{O}(k^5)
\end{aligned} \tag{4.52}$$

where h denotes the small step interval.

In the next section we explain on how we intend to use discriminative direction as a solution to the problem of synthesizing emotions on the facial images.

4.2.3 Use of discriminative direction for emotion synthesis

We now look at how all the analysis we did in the previous section is going to help us. We need to remember the fact that we are trying to synthesize emotions on the surface of a facial image with no-expression. We also do not have the information of the neighbourhood of such an image. Keeping these things in mind, we look at the feature vectors we extracted in the previous chapter. These feature vectors capture local information on the surface of a human face, in the form of regions where typically most changes occur as the face goes from a state of no expression to maximum expression. In other words the local regions identified can be generalized for any face.

We can extract the information about how changes on the surface of human face occur in $\mathcal{X}$, with the help of discriminative direction as discussed in the last section. If we use the active regions as features, we aid to the classfier by giving it information only regarding the local changes in the surface of the face as it goes through different stages of any one expression. As such we use the feature vectors identified in the last chapter as features for modelling the deformations. Since they contain pixel values, they can also be used directly for emotion synthesis on facial expression images once we have the pre-images. We show in the next chapter that these discriminative direction from each feature vector can be used as a representaive of each class. We add the deformed vectors to the input images, after some pre-processing and hence are successful in synthesizing the emotions. We also show that discriminative direction helps us to achieve a powerful tool to represent the data in $\mathcal{X}$.

Chapter 5

Experimental Results

5.1 Dataset

In this section we analyze the results due to discriminative direction and regularized discriminative direction using the feature vectors constructed as a result of active region segmentation. The study includes capturing the changes identified by both algorithms on the same feature set. We aim at benchmarking the results due to discriminative direction and regularized discriminative direction using features recognized by active region segmentation against features taken from all pixels in a population. For this we use the Cohn-Kanade facial expression database [20]. From the database, we have selected 27 sequences of 27 subjects displaying anger (labels for each sequence is known). In each sequence, there are at least 7 frames. Each sequence of frames start with a facial expression image with almost no expression and end with maximum expression, the intermediate frames representing the gradual change of each face. Images are digitized to 640×480 or 490 pixels with 8-bit precision for gray-level shading.

Each image in the database contain at most 50% of face regions. Since we work on finding active pixels which are valid for any image, we need a proper standardization techniqueto find and normalize the orientation and position of the face. We did not consider any sequence of images where any rotation along horizontal and vertical images exist. We then used ASM [7] to find faces and then segment them. We then resize all the segmented image to 100×100. We then use TV-L2 decomposition [2] to decompose some of the facial structure and texture from all the images in the database. All experiments are done on both kind of images separately. Finally during synthesis we combine by joining respective structure and texture images.

5.2 Experiments

To study the pre-images generated, we separate the dataset into two classes. Class labels used are: *-1* for facial expresson images with no expression (we refer to these as normal class henceforth) and *1* for facial expresson images with some or maximum expression of any one type, *e.g.* anger (we refer to these as abnormal class henceforth). The face images are converted into m-dimensional vectors, where m is the product of the number of rows and columns of the image matrix. For each person's images in the dataset, we denote the first four frames as normal class and last four frames as abnormal class. The step size $\Delta\mathbf{s} = \mathbf{s}_t - \mathbf{s}_{t-1}$ in the direction of the normal to the classifier of SVM is taken as *0.1* in each algorithm. A total of *10* steps are considered during reconstruction. We perform two experiments principally, a) construction of facial expression images using all pixel values as feature for the two methods, one, using Golland's discriminative direction and two using Zhou's regularized discriminative direction and b) construction of facial expression images using emotion specific features as features for the two methods, one, using Golland's discriminative direction and two using Zhou's regularized discriminative direction. We then compare the results on the basis of SNR with respect to ground truth. We also demonstrate the cluster indices by using discriminative directions as a representaive of each image in a cluster. We also finally do a complexity analysis of our methods.

5.3 Tables and figures

To illustrate the ability to capture the minimum deformations required for a vector of the normal class to be classified as a member of the abnormal class, as a result of the two methods, we construct the respective pre-images after deformations in the kernel space using all the pixel values. Four reconstructed frames for each methods are shown in figures 5.1, 5.2, 5.3, and 5.4. While figures 5.1 and 5.3 show the outputs of the respective pre-image methods, figures 5.2 and 5.4 show the changes that are identified across time. Blue marks those pixels whose values increases with time, and red marks those pixels whose value decreases with time, taking the initial image as the frame of reference. In Fig. 5.5 and 5.6 we show the result of using active regions as features on both the methods. Again, blue marks those pixels whose values increases with time, and red marks those pixels whose value decreases with time, taking the initial image as the frame of reference.

In table 5.3, we show the size of feature vectors used in case of our feature extraction method against pixel values considered for each image. In table 5.1, we compare the clusters of no expression image against images with some or maximum expression using the features identified by our method and the features identified

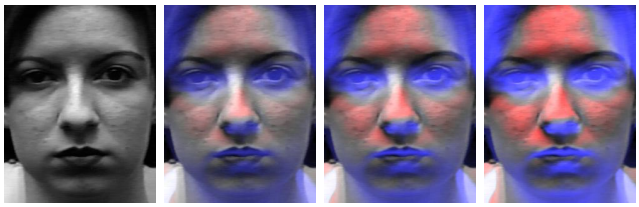
by using pixel values. In table 5.2 we compare the results of SNR of the synthesized images, using each of the methods, and ground truth frames.

5.4 Images

In this section we present the images that shows the synthesis methods. In Fig. 5.2, 5.4, 5.6, and 5.8, the red region represent the regions where the pixel values have increased and blue represent the region where the pixel values have decreased.

(a) (b) (c) (d)

Figure 5.1: *Reconstructed images using discriminative direction*

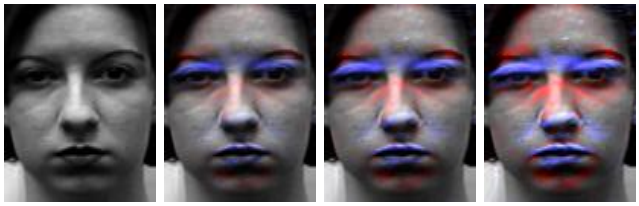


(a) (b) (c) (d)

Figure 5.2: *Regions of changes identified using discriminative direction*

(a) (b) (c) (d)

Figure 5.3: *Reconstructed images using regularized discriminative direction*

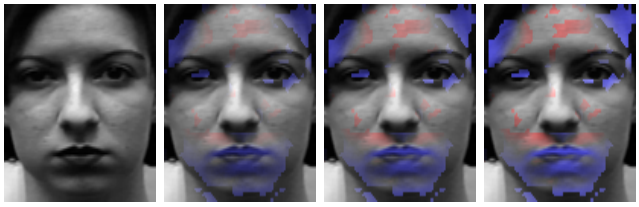


(a) (b) (c) (d)

Figure 5.4: *Regions of changes identified using regularized discriminative direction*

(a) (b) (c) (d)

Figure 5.5: *Reconstructed images using discriminative direction and emotion-specific features*

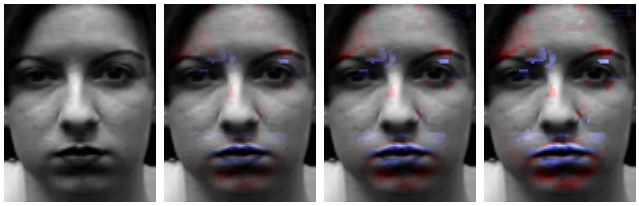


(a) (b) (c) (d)

Figure 5.6: *Regions of changes identified using discriminative direction and emotion-specific features*

(a) (b) (c) (d)

Figure 5.7: *Reconstructed images using regularized discriminative direction and emotion-specific features*



(a) (b) (c) (d)

Figure 5.8: *Regions of changes identified using discriminative direction and emotion-specific features*

5.5 Comparison

5.5.1 Cluster Analysis

As mentioned before we represent each vector in each class using their respective discriminative direction. Thus the clusters C_1 and C_2 represent discriminative direction of no-expression images and with expression images. The cluster measures of these two clusters is shown in table 5.1

Table 5.1: *Cluster Measures of different methods on the dataset*

Method	DaviesBouldinIndex	DunnIndex	F-Measure
Discriminative direction (using all pixel values)	.1391	2.6430	.9577
Regularized discriminative direction (using all pixel values)	.1678	3.1422	.8423
Discriminative direction (using active regions)	.1343	2.7202	0.5516
Regularized discriminative direction (using active regions)	.1769	3.2952	.9938

5.5.2 Quality of synthesis

We calculate SNR of each synthesized image with the ground truth and present the results in table 5.2. We can clearly see that implementation using our features yeild better results.

Table 5.2: *SNR of the synthesized image with the ground truth*

Method	SNR
Discriminative direction (using all pixel values)	14.9151
Regularized discriminative direction (using all pixel values)	18.1130
Discriminative direction (using active regions)	18.3793
Regularized discriminative direction (using active regions)	20.5395

5.5.3 Number of features

We present a comparision of the number of features required to represent each vector in table 5.3. This is important becuase of the fact that the decrement in features is infact beneficiary for faster execution and practical usability of the method. This also helps in decreasing the complexity of each algorithm used to construct discriminative direction.

Table 5.3: *Comparision of the size of features using active regions and using all pixel values*

Method	Number of features
Using active regions	3364
Using all pixels	10000

5.6 Complexity Analysis

We detail here the time complexity of our method. Let there be n pixels in an image. Finding the Gouraud shading vectors can be done in $\mathcal{O}(n)$ time. The surface triangulation can also be done in $\mathcal{O}(nlogn)$ [30]. Thus for r sequence of images a total of rn vectors are to be found. For each of the r sequence of Gouraud shading vectors we need to find a mean and a standard deviation. Both of these can be found in $\mathcal{O}(rn)$ time. To find which pixels are active we calculate the angle between the interval of mean and sigma. This can be done in $2\mathcal{O}(n)$. Selection of active pixels are done using first ranking of the angle variables in decreasing order of angles, and then selecting k number of features where k can be any number less than n. This can be done in $\mathcal{O}(nlogn)$. From these active pixels, each pixel's 4-connectivity can be calculated in $\mathcal{O}(n)$ time. Then finding the connected components can be done in $\mathcal{O}(n)$ time using depth first search techniques [8]. We then for each connected components calculate the convex hull. Thus can be done in $\mathcal{O}(nlogn)$ time [15]. Thus the overall complexity never goes beyond $\mathcal{O}(rn)$. In ideal situations of facial expression identification $r << n$. Thus the complexity of our algorithm never goes beyond $\mathcal{O}(nlogn)$. However this helps in significantly decreasing the size of the features. As such, computations for both the methods, discriminative direction and regularized discriminative direction is hastened.

Chapter 6

Conclusion

In this thesis we study the problem of image-based feature detection to represent emotion. This can help in better clustering of facial expression images and also emotion synthesis. Our study reveals, that extracting such features are of greater importance for reasons of complexity in preventing facial structure to influence the feature vectors. We have used a novel feature extraction technique to minimize the effect of facial structure by extracting localized informatio from the surface of human face. We then use this feature to study the differences in images of no facial expressions and with facial expressions. We demonstrate our results on facial expression images. We believe that such localized information extraction techniques can be used in broader aspects to identify local deformities (as in medical image analysis). The informations can then be used to understand the discriminative features between two class. The method can be extended to generative models to understand differences in populations. The analysis generates a detailed description of deformations on the surface of any shape and can facilitate understanding the cause of such deformations.

Chapter 7

Future Direction

Classification using discriminative directions has not been done during the course of work. This is due to the fact that parameter estimation was taking up a lot of time, and thus it became difficult to estimate an optimal classifier.

We also tried to find a closed form solution to the problem. A closed solution for pre-image in KPCA exists in [17]. Using generalized representer theorem [28] this can be achieved.

There is one last probblem that has not been addressed here. Defining a manifold on which any discriminative direction can be generalized. This is valid due to the fact that all face images has been studied to follow certain deformation pattern as it goes from a state of no expression to maximum expression. If this direction can be modelled using a vector space, the work of estimation any changes on the surface of the face due to a single expression shall be easy. It has to be noted that combining information from all expression we can never create a vector space.

Bibliography

[1] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, pages 337–404, 1950.

[2] Jean-François Aujol, Guy Gilboa, Tony Chan, and Stanley Osher. Structure-texture image decompositionmodeling, algorithms, and parameter selection. *International Journal of Computer Vision*, 67(1):111–136, 2006.

[3] Marian Stewart Bartlett, Gwen Littlewort, Claudia Lainscsek, Ian Fasel, and Javier Movellan. Machine learning methods for fully automatic recognition of facial expressions and facial actions. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 1, pages 592–597. IEEE, 2004.

[4] Dominique Béréziat, Isabelle Herlin, and Laurent Younes. A generalized optical flow constraint and its physical interpretation. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 487–492. IEEE, 2000.

[5] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[6] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[7] Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[8] Thomas H Cormen. *Introduction to algorithms*, volume 2. MIT press, 2009.

[9] Charles Darwin, Paul Ekman, and Phillip Prodger. *The expression of the emotions in man and animals*. Oxford University Press, USA, 1998.

[10] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. *Computational geometry*. Springer, 2000.

[11] Paul Ekman and Wallace V Friesen. Facial action coding system. *Consulting Psychologists Press, Stanford University, Palo Alto*, 1977.

[12] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[13] Polina Golland, W Eric L Grimson, Martha E Shenton, and Ron Kikinis. Detection and analysis of statistical differences in anatomical shape. *Medical image analysis*, 9(1):69–86, 2005.

[14] Henri Gouraud. Continuous shading of curved surfaces. *Computers, IEEE Transactions on*, 100(6):623–629, 1971.

[15] Ronald L. Graham. An efficient algorith for determining the convex hull of a finite planar set. *Information processing letters*, 1(4):132–133, 1972.

[16] Jihun Ham and Daniel D Lee. Separating pose and expression in face images: a manifold learning approach. *Neural Information Processing-Letters and Reviews*, 11(4/6):91–100, 2007.

[17] Paul Honeine and Cédric Richard. A closed-form solution for the pre-image problem in kernel-based machines. *Journal of Signal Processing Systems*, 65(3):289–299, 2011.

[18] Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics, 1981.

[19] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[20] Takeo Kanade, Jeffrey F Cohn, and Yingli Tian. Comprehensive database for facial expression analysis. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 46–53. IEEE, 2000.

[21] George Kimeldorf and Grace Wahba. Some results on tchebycheffian spline functions. *Journal of mathematical analysis and applications*, 33(1):82–95, 1971.

[22] James Tin-Yau Kwok and Ivor Wai-Hung Tsang. The pre-image problem in kernel methods. *Neural Networks, IEEE Transactions on*, 15(6):1517–1525, 2004.

[23] James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, pages 415–446, 1909.

[24] Hans-Hellmut Nagel. Displacement vectors derived from second-order intensity variations in image sequences. *Computer Vision, Graphics, and Image Processing*, 21(1):85–117, 1983.

[25] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*, volume 501. John Wiley & Sons, 2009.

[26] Yogesh Rathi, Samuel Dambreville, and Allen Tannenbaum. Statistical shape analysis using kernel pca. In *Electronic Imaging 2006*, pages 60641B–60641B. International Society for Optics and Photonics, 2006.

[27] William T Reeves. Inbetweening for computer animation utilizing moving point constraints. *ACM SIGGRAPH Computer Graphics*, 15(3):263–269, 1981.

[28] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *Computational learning theory*, pages 416–426. Springer, 2001.

[29] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[30] SW Sloan. A fast algorithm for generating constrained delaunay triangulations. *Computers & Structures*, 47(3):441–450, 1993.

[31] Ying-li Tian, Takeo Kanade, and Jeffrey F Cohn. Recognizing action units for facial expression analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(2):97–115, 2001.

[32] Michel Valstar and Maja Pantic. Fully automatic facial action unit detection and temporal analysis. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 149–149. IEEE, 2006.

[33] Michel Valstar, Maja Pantic, and Ioannis Patras. Motion history for facial action detection in video. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 1, pages 635–640. IEEE, 2004.

[34] Alessandro Verri, Federico Girosi, and Vincent Torre. Differential techniques for optical flow. *JOSA A*, 7(5):912–922, 1990.

[35] Luping Zhou, Richard Hartley, Lei Wang, Paulette Lieby, and Nick Barnes. Regularized discriminative direction for shape difference analysis. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2008*, pages 628–635. Springer, 2008.