FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

ENHANCING MORPHOLOGICAL CATEGORIZATION WITH MONTE CARLO METHODS

By

SERDAR CELLAT

A Dissertation submitted to the
Department of Mathematics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Degree Awarded:
Spring Semester, 2018

SERDAR CELLAT defended this dissertation on January 15, 2018.
The members of the supervisory committee were:

Washington Mio
Chair

Giray Okten
Co-chair

Sudhir Aggarwal
Committee Member

Nick Cogan
Committee Member

Harsh Jain
Committee Member

Department of Mathematics has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

The following short list of symbols are used throughout the dissertation.

n	Number of landmarks
k	Dimension of landmark points
m	Number of shapes (P or Q)
Σ	Weight Matrix (Positive Definite & Symmetric)
A	$\Sigma = e^A$ (Symmetric)
P_i	Shapes with size $k \times n$
P^T	Transpose of P
P_c	Centered P
P, Q	Preshapes
$\hat{P}^i$	Aligned Pre-shape P
U	rotation matrix

CHAPTER 1

INTRODUCTION

Study of shapes has been increasingly important in many fields of science including but not limited to evolutionary and developmental biology, biomedical imaging, computer vision, and computer graphics. One of the fundamental problems in the analyis of shapes is to study similarities and dissimilarities between objects. It is also of great interest to analyze the shape variation within a population to discover underlying genotypic mechanism.

The problems of modeling, classifying and recognizing shapes permeate multiple domains of application. To examplify, the investigation of the genetic underpinnings of organismal phenotypic variation at scales ranging from organelles to full organisms often leads to complex problems in shape analysis (cf. [**?**, **?**, **?**]). Complex morphology and structure of pollen grains are of great interest to paleontologists as they form the most abundant and extensive record of plant diversity [**?**, **?**]. In ecology, there is evidence that variation in shape of spatial vegetation patterns might signal critical changes in ecosystems [**?**, **?**]. These are just a few examples in the broad landscape of problems that involve shape quantification and analysis.

1.1 Earlier Works on Shape Quantification

From mathematical point of view, the shape of an object is the geometrical information that is invariant to transformation, scale and rotation []. A shape in our study is defined by locating a finite number of points on the outline, called landmark points. A landmark is a point of correspondence on each object that matches between and within populations [**?**].

Earlier works in shape analysis is attributed to D'Arcy Thompson [] where he was the first scientist to study objects not comparing their features but rather using their geometrical information. Traditional morphometrics consisted of mesuring certain features of species and applying multivariate statistical analysis on collected information. An eminent study to this type of morphometics is R.A Fisher's *iris flower* data. In his data collection process he first measured the sepal and petal's length and width of three flower species, *setosa, versicolor, and virginica.* He then analyzed the species using multivariate statistical techniques with only these four measured information. Summaries obtained using only collected measures such as mass, length, width, and color information generally fall short in analyzing objects. Specifically, as objects get more intricate traditional multivariate methods fail to address the questions that the researchers are after.

The first formal mathematical treatment of shape, due to Kendall, adopts a shape representation based on labeled landmark points [**?**]. This is illustrated in Fig. **??** that shows a fruit fly wing represented by twelve landmark points placed at vein crossings [?]. Kendall constructed a shape space equipped with a metric that quantifies morphological dissimilarity, providing a setting for systematic statistical analysis of shape variation (cf. [?, ?]).

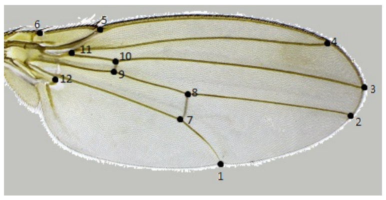


Figure 1.1: Fruit fly wing with twelve landmark points (image courtesy of Houle Lab, Florida State University)

In the original landmark model, all landmark points were treated as equally important in the process of aligning shapes for comparison and quantification of dissimilarity. Oftentimes, however, the morphological differences that most sharply contrast two or more populations, such as different species of fruit flies, are concentrated in particular regions. This suggests the investigation of inhomogeneous variants that highlight regions of interest.

Bookstein [?] proposed a new shape distance which gives different weights to landmark points and C. Goodall [?] introduced the generalized procrustes analysis using Mahalanobis distance of a covariance matrix to give weights to each landmark. An extension of above two models, the *weight matrix model* is investigated in [?]. In the weight matrix model, the aim is to find an optimal weight matrix that enhances the separation of different shape families. The objective is to minimize the distances between shapes that belong to same population while keeping different populations away at a maximum distance.

1.2 Contributions of this Dissertation

This thesis will mainly focus on the practical issues that arises in weight matrix shape model studied in [?]. While proposed shape model is theoretically solid, it is insufficient from computational perspective. We first identify each issue and propose our solutions accordingly.

Weight matrix model develops a shape space formulation that yields a family of shape metrics parameterized by $n \times n$ positive definite, symmetric matrices, where n is the number of landmark points. The matrices encode weights assigned to landmarks or linear combinations of landmarks, thus having the effect of attributing different importance to different parts of a shape.

This family of shape metrics at hand, one now has the problem of selecting a metric that is well suited to an application. For example, we would like to have computationally feasible ways of choosing a shape metric that effectively classify given species of fruit flies. The question is formulated as an optimization problem in the subspace of trace zero symmetric matrices. Here we list computational issues of the original model along with our proposed solutions and results in applications, which we consider contributions of the thesis.

Original model proposes the use of gradient-based optimization methods in the search of optimal weight matrix. However, as the search space (of symmetric matrices) is typically high dimensional, finding such optimal weight matrix with gradient-based methods can be costly and problematic. Since gradient based methods are considered to be local search methods, it is natural to get trapped into a local optima in search of global optimum. Accordingly, the results in classifying shapes may not be satisfactory. To mitigate this problem we propose the use of Monte Carlo optimization methods to learn metrics that enhance shape categorization.

Although the original model may produce acceptably good results for shapes with moderate sized landmark points, say shapes with 10 to 20 landmarks, it fails to maintain the performance in classifying shapes when the number of landmarks is relatively high. We proposed another solution to this problem where we consider to omit some landmarks in the optimization process. After reducing the number of landmarks we applied Monte Carlo methods to find the optimal weight matrix for the shapes with smaller number of landmarks. As a final step in this process, we interpolated the weight matrix obtained with less number of landmark to the full weight matrix that would have been obtained by the full landmark points.

This scheme is mostly useful in two scenarios. First, it is obviously practical for shapes having many landmarks. It is also advantageous for shapes where landmarking is a difficult or redundant task. Landmarking process oftentimes can be challenging mainly due to the correspondence, where correspondence refers to the means of locating same numbered landmarks to the same places on each object. Landmarking can also be redundant with the advance of 3D imaging. Nowadays researchers can capture the images of objects with minuscule efforts. And with the help of softwares, 3D images can serve as point clouds where each point can be considered as a landmark point. It is still possible to apply weight matrix model to problems with such objects if we reduce the number of landmarks. We call this process *landmark sparcification* and it is another main contribution of this dissertation. We give the details in chapter 5.

We applied our proposed solutions along with the idea of weight matrix model in three different experiments. The shapes in the first experiment are synthetic shapes where we generated them. We first wanted to apply our proposed algorithms on synthetic data to investigate the feasiblity of our approach. After achieving fulfilling results we then proceeded to two biological applications. In the first biological application we worked on the shapes of 3D mice skull. We performed two different analysis on these shapes. We first applied our method to 8 parental strains and were able to distinguish them with exceptional classification accuricies. Afterwards, we worked with all the shapes where we had 62 offsprings of 8 parental strains. We again used our methods and algorithms to separate them efficiently. At the end we were able to create a hierarchical organization for the 62 groups.

In the final application we analyzed shapes (of wings) of fruit flies where our goal was to create a taxonomical tree that reveals the closeness of each species to one another.

The organization of the dissertation is as follows. In chapter 2, we briefly visit the basics of statistical shape analysis. Chapter 3 summarizes generalized shape model, namely the weight matrix model, where we present the objective function and address the practical issues that come up with the original model. In chapter 4, we review several Monte Carlo optimization techniques from different class of methods to find a suitable one for our studies. We introduce the idea of landmark sparcification in chapter 5. We also test the effectiveness of the landmark sparcification method on synthetic shapes using different subselection of landmarks. In chapter 6, we present three applications of our method to illustrate the improvement of a given shape classification problem using weight matrix model coupled with Monte Carlo optimization. We conclude with the summary of findings and some discussion.

CHAPTER 2

MATHEMATICAL PRELIMINARIES

In this chapter, we review some concepts of shape analysis and learning shape metrics introduced in [?]. We start by investigating statistical analysis of shapes.

2.1 Statistical Shape Analysis

The field of statistical shape analysis involves methods for the study of the shape of objects and it has been increasingly important with advances in technology. Shape analysis is of great interest in a wide variety of disciplines including biology, medicine, image analysis, archaeology, geography and in many other fields. The main aims of statistical shape analysis are to estimate average shapes and the structure of a population's shape variability, to distinguish different shape classes and carry out inference on population quantities [?]

Earlier works on shape analysis go back at least as far as 1917. The topic was initially developed by D'Arcy Thompson(1917)[?] from a biological point of view. After several decades mathematical and statistical aspects of shape analysis were introduced by D.G Kendall [?],[?],[?] and his works on shape analysis are the foundations of what we call *shape theory.* At about the same time F.L Bookstein began to study shape-theoretic problems in the particular context of zoology. These two authors are considered to be the pioneers in the field and their works provide bases for the work presented here. Dryden and Mardia's work on statistical shape analysis is another great reference for the fundamentals of the field [?],[?].

We now discuss the procrustes analysis of shapes which is the very first step of shape analysis. We then find the mean shape of a group of shapes and finally introduce shape variablilty using principal component analysis- a dimensional reduction technique that is the main component in analyzing shape variation.

2.2 Procrustes Methods

Procrustes Analysis is the name given by Hurley and Cattell(1962) [?] to a statistical method to describe the variability in a data.The method was then used in the context of multidimensional scaling by Schónemann and Carroll (1970)[?] and Gower (1975) [?]. Since then it has been very important method in statistical shape analysis with the contributions of Kendall(1984,1989), C.Goodall(1991) and Dryden and Mardia(1998).

The idea of the method lies in the answer of the following question: how much does one configuration differ from another? In order to give a precise answer to this question we must make sure that we can analyze them in a common ground. This common ground can be obtained only after certain operations: translation/centering, scaling or normalization, and rotation. These are the three main steps in Procrustes Analysis.In the following section we describe the classical procrustes analysis proposed by Kendall.

In Kendall's model, a shape in $\mathbb{R}^k$ is represented by a collection of ordered n landmark points $p_1, p_2, \ldots, p_n$ with each $p_i \in \mathbb{R}^k$. This collection of landmark points is a sparse representation of the original shape. Therefore, each shape can be represented by a $k \times n$ matrix $\begin{bmatrix} p_1 & p_2 & \ldots & p_n \end{bmatrix}$. To get translation and scale-invariant shapes, one can move the shapes so they are centered at the origin and scale the matrix to have **Frobenius** norm 1.

Remark 1. $\|.\|$ *is the **Frobenius** norm associated with the inner product*

$$< P, Q >= \sum_{j=1}^{n} p_j \cdot q_j = \sum_{i=1}^{k} \sum_{j=1}^{n} p_{ij} q_{ij}$$

where p_j,q_j is the jth landmark of the shapes P and Q.

1. **Centering**

   A shape is centered if
   $$p_1 + p_2 + \cdots + p_n = 0.$$
   To do centering we subtract the mean of the landmark points from each landmark. The mean (or centroid) of $p_1, p_2, \ldots p_n \in \mathbb{R}^k$ is the point
   $$C_p = \frac{p_1 + p_2 + \ldots p_n}{n}$$
   here $C_p \in \mathbb{R}^k$. Therefore, centering a shape matrix P is done by
   $$P_c \to P - C_p = \begin{bmatrix} p_1 - C_p & \ldots & p_n - C_p \end{bmatrix}$$
   where P_c represents a centered shape.

2. **Scaling**
   Scaling is the same as normalizing a shape which basically divides a shape by its Frobenius norm so that it has unit length. Scaling sends each nonzero shape P onto the unit sphere by

$$P \leftarrow \frac{P}{\|P\|}$$

again, $\|P\| = (\sum_{j=1}^{n} \|p_j\|^2)^{1/2}$ is the Frobenius norm of P.

A centered and scaled shape is called **preshape** and the space of all possible preshapes is denoted by $\mathbb{S}_n^k$. *From now on we assume all shapes are preshapes.*

Since, by definition, a preshape is invariant under the translation and scaling, if $U \in O(k)$ and $P \in \mathbb{S}_n^k$, then UP represents the same shape as P.

3. **Orthogonal Alignment**

Aligning two shapes with each other by means of orthogonal rotations is to minimize squared distances between corresponding landmarks of each shape. And this can be done by rotating both shapes with two orthogonal matrices. If $P, Q \in \mathbb{S}_n^k$,and $U, V \in O(k)$,we can find the optimal $\hat{U}, \hat{V}$ so that the right hand side of the argument below is minimized

$$\hat{U}, \hat{V} = \arg \min_{U,V \in O(k)} \|UP - VQ\|^2 \tag{2.1}$$

(2.1) can be reduced to finding one orthogonal matrix instead of two in the following sense;

Since $\|UP - VQ\|^2 =< UP - VQ, UP - VQ >$ if we multiply both sides by U^T from the left we get

$$\begin{aligned} < U^T(UP - VQ), U^T(UP - VQ) > &=< U^TUP - U^TVQ, U^TUP - U^TVQ > \\ &=< P - U^TVQ, P - U^TVQ > \quad (U^TU = \mathbb{I}_k) \\ &= \|P - U^TVQ\|^2 \end{aligned} \tag{2.2}$$

Finally, if we just write $U^TV = U$ for the convenience, (2.1) would be equivalent to finding optimal $\hat{U}$ that minimizes (2.3)

$$\hat{U} = \arg \min_{U \in O(k)} \|P - UQ\|^2 \tag{2.3}$$

Now the aim is to find $\hat{U}$ that minimizes (2.3). We can write $\|P - UQ\|^2$ as

$$\begin{aligned} \|P - UQ\|^2 &=< P - UQ, P - UQ > \\ &=< P, P > + < Q, Q > -2 < P, UQ > \end{aligned} \tag{2.4}$$

where $P, Q \in \mathbb{S}_n^k$ and $U \in O(k)$. Since P, Q are both preshapes,
$< P, P >= 1$ and $< Q, Q >= 1$. Therefore equation (2.4) can be rewritten as

$$\|P - UQ\|^2 = 2 - 2 < P, UQ > \tag{2.5}$$

Maximizing $< P, UQ >$ is the same as minimizing (2.5).

$$\begin{aligned} < P, UQ > &=< PQ^T, U > \\ &=< A, U > \end{aligned}$$

where $A = PQ^T$. Now we want to maximize $< A, U >$. Let $U_1 \Sigma V_1^T$ be a singular value decomposition(SVD) of A, then

$$\begin{aligned} < A, U > &=< U_1 \Sigma V_1^T, U > \\ &=< \Sigma, U_1^T U V_1 > \\ &=< \Sigma, V > \end{aligned} \tag{2.6}$$

where $V = U_1^T U V_1$, and Σ is a $k \times k$ diagonal matrix with positive real entries on the diagonal and $V \in O(k)$.

(2.6) can be written as

$$< \Sigma, V >= \sigma_1 V_{11} + \cdots + \sigma_k V_{kk} \tag{2.7}$$

where V_{ii} is the diagonal entries of V. Here $\sigma_i \geq 0, |V_{ii}| \leq 1$ for $i = 1, \ldots, k$.

Thus, (2.7) can be maximized by choosing $V_{ii} = 1$. In this case, $V_{ij} = 0$ for $i \neq j$. This is equivalent to setting $V = \mathbb{I}_k$.

The solution $\hat{U}$ is therefore given by

$$U_1^T \hat{U} V_1 = V = \mathbb{I} \rightarrow \hat{U} = U_1 V_1^T$$

Geodesic distance: is the shape distance on the preshape space between P and $\hat{Q} = \hat{U}Q$, and can be calculated as

$$d(P, Q) = \arccos < P, Q >= \arccos(PQ^T) \tag{2.8}$$

This is simply the shortest arc length of the great circle on preshape space $\mathbb{S}_n^k$ connecting P and Q.

2.2.1 2D Example of Kendall's Shape Model

In fig. **??** we apply the entire procedure of Kendall's shape model on 2D hand shapes. The first figure from left is the original hand shapes, the second is the centered hand shapes, the normalized/scaled shapes are shown in the third and forth illustrates the alignment of blue shape with the red one.

2.3 Mean Shape

Mean shape plays an important role in statistical shape analysis. In this section we will discuss how to find mean shape of a given family of shapes by using attracting fixed point method introduced by Liu et. al(2010) [**?**]. The method is a fast algorithm to compute Fréchet mean shapes and it is a variant of Huckemann&Ziezold's algorithm (2006)[**?**] developed for the study of planar shapes.

Fréchet Mean Shape. In order to understand calculation of Fréchet mean shape, we first look at the definition of mean in Euclidean space. Let $x_1, \ldots, x_m$ be m samples in $\mathbb{R}^N$, and their mean is given by

$$\mu = \frac{1}{m}(x_1 + \cdots + x_m) = \frac{1}{m}\sum_{i=1}^{m} x_i$$

This mean is indeed the minimizer of the Fréchet function $V : \mathbb{R}^N \to \mathbb{R}$ given by

$$V(x) = \sum_{i=1}^{m} \|x - x_i\|^2, \tag{2.9}$$

where $\|\cdot\|$ is the Euclidean norm. Fréchet mean is the point that minimizes the Fréchet function and can be extended to more general metric spaces, in our case it is preshape space.

Let $s_1, \ldots, s_r \in \Sigma$ be a family of shapes represented by the preshapes $P_1, \ldots, P_r \in \mathbb{S}$ where Σ denotes the shape space and $\mathbb{S}$ is the preshape space. Consider $P \in \mathbb{S}$, and let $U_i(P) \in O(N)$ be the orthogonal transformation that optimally aligns P_i with P. We then define the Fréchet mean shape $\bar{s}$ of the family as a shape that minimizes the Fréchet function

$$V(s) = \frac{1}{2}\sum_{i=1}^{r} d^2(s, s_i), \tag{2.10}$$

where d is the shape distance between s and s_i, and $1/2$ is introduced for convenience. In cases of geodesic distance (2.2) and Euclidean distance the scatter function can be expressed respectively as

$$V_1(s) = \frac{1}{2}\sum_{i=1}^{r} \arccos^2 < P, U_i(P) \circ P_i >, \tag{2.11}$$

and

$$V_2(s) = \frac{1}{2}\sum_{i=1}^{r} \|P - U_i(P) \circ P_i\|^2. \tag{2.12}$$

Attracting Fixed Point Method. Attracting fixed point is a much faster method in calculating mean shape than some other optimization methods such as gradient descent. We consider the minimization problem of the scatter function V_1 in which the geodesic distance is used. We compute the solution using the method of Lagrange multipliers with the constraint that P is on the unit sphere so that it has length 1. Now we want to minimize $V_1(P)$ subject to $< P, P >= 1$. The Lagrange multipliers set up for this optimization problem then becomes $\Lambda(P) = V_1(P) + \frac{1}{2}\lambda(< P, P > -1)$. Taking the gradient of $\Lambda(P)$ and setting it 0 we can find the stationary points of Λ as

$d\Lambda_P = dV_1(P) + \lambda < P, P' >=< \nabla V_1(P), P' > + < \lambda P, P' >= 0$ Therefore,

$\nabla V_1(P) + \lambda P = 0$ and $P = -\dfrac{\nabla V_1(P)}{\lambda}$. The gradient of V_1 is given by

$$\nabla V_1 = -\sum_{i=1}^{r} \frac{\arccos < P, \hat{P}_i >}{\sqrt{1- < P, \hat{P}_i >^2}} \hat{P}_i. \tag{2.13}$$

where $\hat{P}_i = U_i(P) \circ P_i$. Finally, at a minimum we obtain

$$P = sign(\lambda) \frac{\nabla V_1(P)}{\|\nabla V_1(P)\|} \tag{2.14}$$

here λ denotes $< P, \nabla V_1 >$, and $\nabla V_1(P)$ is divided by its norm to make sure that the mean shape is also on the unit sphere. Therefore, we are interested in attracting fixed points of the mapping $T : \mathbb{S} \to \mathbb{S}$ defined by

$$T(P) = sign(\lambda) \frac{\nabla V_1(P)}{\|\nabla V_1(P)\|} \tag{2.15}$$

as they lead to a stable minima of V_1. The algorithm to find the mean shape P is then as follows

1. Initialize the process with a random preshape P
2. Calculate $T(P) = sign(\lambda) \dfrac{\nabla V_1(P)}{\|\nabla V_1(P)\|}$
3. Calculate $\|T(P) - P\|$
4. If $\|T(P) - P\| < \epsilon$, stop. Else, update $P = T(P)$ and go to step 2

2.4 Principal Component Analysis(PCA)

Principal Component Analysis is a dimensionality reduction technique and a powerful tool to use in multidimensional scaling. The main idea of PCA is to reduce the dimensionality of a data set, in which a large number of variables are correlated, while keeping as much as possible of the variation present in the data set. This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated and ordered so that the first few contains most of the variation in the original data I.T.Jolliffe(2002)[**?**].

2.4.1 PCA in Euclidean Space

We will first perform PCA in Euclidean space. Let $x_1, x_2, \ldots, x_n$ be n sample points in $\mathbb{R}^k$ and let $X = [x_1 x_2 \ldots x_n]$ be $k \times n$ matrix. Following procedure reveals the PCs of X with their corresponding effects on variation.

1. Subtract the mean of X ,$\bar{x}$, from each column of X. This way we place sample points' centroid at the origin.

2. Calculate the sample covariance matrix of X

$$S = \frac{1}{n-1} X X^T$$

   where S is a $k \times k$ matrix with the rank r.

3. Diagonalize S using singular value decomposition (SVD) so that

$$S = U \Sigma U^T$$

Here Σ is the diagonal matrix with the elements σ_i^2 in a decreasing order so that $\sigma_1^2 > \sigma_2^2 > \ldots \sigma_r^2 > 0 \ldots 0$.Each σ_i^2 is called principal scores and used to determine how much variation occurs in corresponding principal component. U is an orthogonal matrix and the ith column, U_i, is the i th principal component/eigenvector(PC$_i$) of covariance matrix with σ_i^2 variance/eigenvalue. This procedure reveals at most k PCs since U is a $k \times k$ matrix.

When n is smaller then k we perform PCA in a slightly different way in order to reduce computational complexity. In this case we diagonalize $X^T X$ instead of $X X^T$ in the following sense.

Diagonalize $X^T X$

$$X^T X = V \Sigma V^T$$
$$X^T X V = V \Sigma$$

multiplying both sides by X from the left we get

$$X X^T X V = \Sigma X V \tag{2.16}$$

Now 2.16 is an eigenvalue equation of sample covariance matrix $X X^T$ with eigenvectors XV and eigenvalues Σ. Therefore, ith column of XV becomes the ith principal components of $X X^T$ with σ_i^2 variance. This way we obtain at most n PCs since XV is an $k \times n$ matrix. Note that the eigenvectors of the covariance matrix needs to be normalized to get unit eigenvectors/PCs.

2.4.2 PCA at Tangent Space

Since preshape space is non-linear, we cannot apply previous algorithm directly. Instead, we will perform PCA at tangent space. We first project each preshape onto tangent plane at the mean shape via the exponential map to preserve the distance between mean shape and each preshape. Then we apply PCA algorithm that we follow in Euclidean case. Once we find the PCs we project them back into the preshape space. The mathematical expression of what we discussed in this section is as follows

1. First calculate W_i, the projection of preshape P_i onto the tangent space at the mean shape P, using the equation

   $$W_i = arccos < P, \hat{P}_i > \frac{\hat{P}_i - < P, \hat{P}_i > P}{\|\hat{P}_i - < P, \hat{P}_i > P\|^2}$$

   Here $\hat{P}_i$ is the preshape P_i aligned with mean shape P and *arccos* part is added for exponential mapping.

2. Suppose we have m preshapes and let X be our sample matrix with the entries

   $$X = [W_1 W_2 W_3 \ldots W_m]$$

   where X is a $k \times m$ matrix. Assume $m < k$.

3. Diagonalize $X^T X$ $\qquad$ $((X^T X)_{ij} =< W_i, W_j >)$ such that

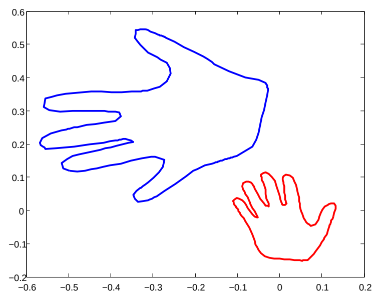
   $X^T X = U \Sigma U^T$ where U and Σ are $m \times m$ matrices. Σ is diagonal matrix with elements in decreasing order $(\sigma_1^2 > \sigma_2^2 > \cdots > \sigma_r^2)$. Each σ_i^2 represents the variations along the ith principal component(PC$_i$) and σ_i is the corresponding standard deviation.

4. The ith principal component on the tangent plane at the mean shape (P) is the ith column of XU. This way we obtain at most m PCs.

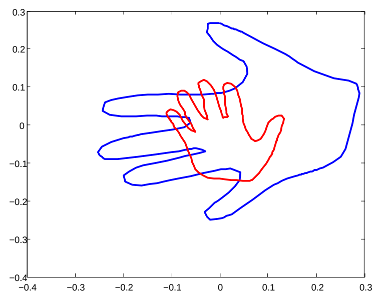
5. Project the PCs back into the preshape space[**?**] by

   $$P_{c\sigma_i} = \cos(c\sigma_i) P + \sin(c\sigma_i) \frac{PC_i}{\|PC_i\|}$$

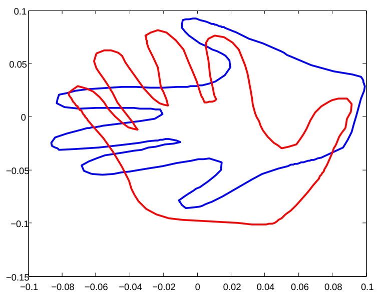
   where c is any real number and $c\sigma_i$ is the geodesic distance between preshape $P_{c\sigma_i}$ and mean shape P. That is, $d_1(P_{c\sigma_i}, P) = c\sigma_i$.



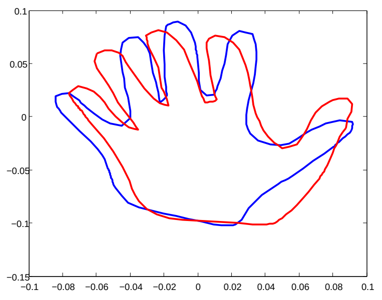
(a) Raw Hand Shapes



(b) Centered Shapes



(c) Scaled(Normalized) Shapes



(d) Aligned Shapes

Figure 2.1:

CHAPTER 3

LEARNING SHAPE METRICS AND CLASSIFICATION

In this chapter, we review an extension of classical procrustes distance, which uses the Euclidean distance metric, to a new family of shape metrics investigated in []. We then present a generalized procrustes analysis associated with any given shape metric. We also introduce a criterion function to find the optimal metric that separates different classes well.

3.1 Introduction to Metric Learning

In classical procrustes analysis every landmark has equal importance. This approach results in shape as equal growth at every point of the organism. However, there are some regions in organisms that can change differently than other regions during development. Change in shape as size changes could be either a result of natural growth or a sign for some diseases. In disease case one would desire to distinguish patients from healthy individuals. In Kendall's shape model the difference/distance between two shapes is basically calculated as the sum of the distances between corresponding landmark points in which each landmark point has equal importance. This approach is unable to locate where the most different region between two shapes is nor to identify the similar parts. Naturally, the following question arises; what would be the best way to separate different group of shapes when they differ from each other at some particular regions? As Yu Fan proposed[**?**] this can be done by attributing different weights to landmark points and developing a classification criterion which maximizes between class distances while keeping the samples belong to same class as close as possible.

In this chapter, we review Yu Fan's weight matrix model[**?**] that generalizes the classical Procrustes analysis by attributing weights to linear combinations of landmarks. In his model, a new family of shape metrics is proposed and an appropriate weight matrix is selected based on selection criterion. This way it becomes more understandable to quantify shape differences during growth and development.

The goal is to find a weight matrix Σ associated with a distance metric that gives the best result for shape classification. Let x and y be vectors in $\mathbb{R}^n$, then any symmetric positive definite matrix Σ defines a distance metric D_Σ in the form of

$$d^2_\Sigma(x,y) = (x-y)^T\Sigma(x-y) =< (x-y)\Sigma, x-y >==< x-y, x-y >_\Sigma \tag{3.1}$$

The general inner product form of our distance metric in shape space is therefore

$$< P, Q >_\Sigma =< P\Sigma, Q >= \text{trace } (P\Sigma Q^T) \tag{3.2}$$

In the implementations there are three cases of the weight matrix. In the first case we give no weights to landmark points and the resulting metric is Euclidean distance metric which we used in classical procrustes analysis. In the second case we give different weights to each landmark but this time we ignore the effect of linear combinations of landmark points. In third case, a full symmetric positive definite matrix is used and it takes linear combinations of landmarks into account as well. In this chapter we adopt the full weight matrix . Mathematical expression of the three cases shown below.

- **Case 1:** When the weight matrix is the identity matrix, i.e. $\Sigma = I_n$, the metric defined in 3.1 reduces to general Euclidean distance metric. We used this distance in classical Procrustes analysis which gives equal importance to each landmark, i.e. there is no weight given to landmarks.

$$d(x,y) = \|x-y\| = \sqrt{<x-y, x-y>} = \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

- **Case 2:** If the weight matrix is a diagonal matrix, that is,

$$\Sigma = \begin{bmatrix} \sigma_{11} & 0 & \dots & 0 \\ 0 & \sigma_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_{nn} \end{bmatrix}$$

  where $\sigma_{ii} = \sigma_i^2$, then the resulting distance measure is called *normalized Euclidean distance* and defined as

$$d(x,y) = \|x-y\|_\Sigma = \sqrt{<x-y, x-y>_\Sigma} = \sqrt{<(x-y)\Sigma, x-y>} = \sqrt{\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{\sigma_i^2}}$$

  This distance measurement gives different weight to each landmark, yet, neglects the effect of linear combinations of landmark points.

- **Case 3:** This is the full case of weight matrix

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \dots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_{nn} \end{bmatrix}$$

  where Σ is an $n \times n$ symmetric positive definite matrix with $\det \Sigma = 1$. We want the determinant of weight matrix to be 1 to have a linear constraint in implementations.

In implementations enforcing Σ to be positive definite might be difficult, instead, we use a logarithmic representation by $\Sigma = \exp A$, with A symmetric and trace $A = 0$ since

$$\det\,(e^A) = 1 \iff \text{trace } A = 0$$

We will also adopt the inner product form introduced in 3.2

3.1.1 Interpretation of Σ-metric

Σ-metric is generally interpreted in two different ways. The first way is to associate it with Euclidean metric in the following sense;

Since Σ is a symmetric positive definite matrix we can decompose it as $\Sigma = LL^T = L^T L = \sqrt{\Sigma}\sqrt{\Sigma}$

$$< P, P^\dagger >_\Sigma = < P\Sigma, P^\dagger > = < PLL^T, P^\dagger > = < PL, P^\dagger L > \tag{3.3}$$

Now the distance metric form can be rewritten in Euclidean distance form equivalently as

$$d^2_\Sigma(P, P^\dagger) = (P - P^\dagger)^T \Sigma (P - P^\dagger) = (P - P^\dagger)^T L^T L (P - P^\dagger) = \|L(P - P^\dagger)\|^2 = \|Q - Q^\dagger\|^2 \tag{3.4}$$

where $q_i = Lp_i$ and $q_i^\dagger = Lp_i^\dagger$

(3.3) and (3.4) simply mean that after changing coordinates the Σ-metric becomes essentially the same as Euclidean metric, and Euclidean metric can be considered as a special case or subspace of Σ-metric. Even though the Σ-metric is fundamentally the same as Euclidean metric, we still need to introduce Σ-metric for two main reasons. First, centering step is quite different in Euclidean metric and we do not want to loose the effect of centering in the generalized procrustes analysis. Second, if we change coordinates each time it increases the computational complexity.

Second way to interpret the Σ-metric is to decompose Σ such that $\Sigma = U\Lambda U^T$ where U is $n \times n$ orthogonal matrix and Λ is $n \times n$ diagonal matrix with positive entries. Since the original form of the inner product is given as

$$< P, Q >_\Sigma = < P\Sigma, Q > \tag{3.5}$$

we can rewrite (3.5) in the following way

$$< P, Q >_\Sigma = < P, Q >_{U\Lambda U^T} = < PU\Lambda U^T, Q > = < PU\Lambda, QU > = < PU, QU >_\Lambda \tag{3.6}$$

The equation (3.6) simply means that if we rotate each shape in the shape space with $U \in O(n)$ then the weight matrix turns into attributing different weights to each landmark of rotated shapes as in the **Case 2**

3.2 Generalized Procrustes Model

Generalized procrustes model is an extension of the classical procrustes methods introduced in the previous chapter. We need to remove the effects of translation, scaling and rotation under the Σ-metric to analyze shapes in a common ground.

3.2.1 Centering under Σ-metric

The idea of centering in Σ-metric is the same as classical procrustes case. We find the mean of landmark points of each shape and then do the centering by subtracting the mean from each landmark. However, finding mean/centroid of landmark points of a shape in Σ-metric is quite different. Let $P = [p_1\ p_2\ \ldots p_n]$be shape with landmark points $p_i \in \mathbb{R}^k$. Assuming x is the mean of landmark points we move shape P to the origin by $P - X$ where $P = [x\ x\ \ldots x]$ is a $k \times n$ matrix and resulting shape can be represented as $P - X = [p_1 - x\ p_2 - x\ \ldots p_n - x]$

Center X of the shape P can be considered as minimizing the distance function with respect to X

$$\min f(X) = \|P - X\|_\Sigma^2, \tag{3.7}$$

where, again, Σ is an $n \times n$ symmetric positive definite matrix. This minimization problem is done earlier by Yu Fan[**?**] and here we will review it. To find the minimizer of f we will take its gradient and set it $\nabla f = 0$. Let us first consider x as a differential map x: $\Omega \to V$ from $\Omega = (-\epsilon, \epsilon)$ to $V = \mathbb{R}^k$. Now we can think of $x(t) \in V$ as a differentiable entity where $t \in \Omega$ and define $x(t)_{t=0} = x(0) = x$ and $\frac{dx(t)}{dt}|_{t=0} = x'(0) = h$.

$$f(X) = \|P - X|_\Sigma^2 =< P - X, P - X >_\Sigma$$

From calculus we know that $\dfrac{d}{dt} < f, g >=< f', g > + < f, g' >$
similarly $\dfrac{d}{dt}\|f\|^2 = \dfrac{d}{dt} < f, f >= 2 < f', f >$. Having P and M both fixed, the directional derivative of f with respect to X is

$$\begin{aligned} df_x(h) &= -2 < [x' \ldots x'], (P - X)\Sigma > \ \ (x' = h) \\ &= -2 < h[1 \ldots 1], (P - X)\Sigma > \\ &= -2 < h, (P - X)\Sigma \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} > \end{aligned} \tag{3.8}$$

Since $df_x(h) =< h, \nabla f >$, from (3.8) we see that the gradient $\nabla f = -2(P - X)\Sigma \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$ setting it to $\bar{0} \in V$ we get

$$P\Sigma \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = X\Sigma \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

that is

$$P\Sigma \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = x[1 \ldots 1]\Sigma \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \tag{3.9}$$

(3.9) can be rewritten as

$$\sum_{i=1}^{n} p_i \sigma_i = x \sum_{i=1}^{n} \sigma_i$$

where σ_i is the sum of the elements in the ith row of Σ and p_i is the ith row of the shape matrix P.

Setting $[1 \ldots 1]^T = e$ and $II = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & 1 & \ldots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \ldots & 1 \end{bmatrix}$

(3.9) now becomes

$$P\Sigma^{-1}e = x < \Sigma, II >$$

Thus, we get

$$x = \frac{P\Sigma e}{< \Sigma, II >}$$

and since $X = [x \ldots x] = x[1 \ldots 1] = xe^T$, the center of P is found as

$$X = \frac{P\Sigma II}{< \Sigma, II >} \tag{3.10}$$

The final expression of generalized centering under Σ-metric is

$$P_c = P - X \tag{3.11}$$

3.2.2 Scaling

We map all centered shapes P_c onto the unit sphere of Σ-metric space as

$$\bar{P} = \frac{P_c}{\|P_c\|_\Sigma} = \frac{P_c}{\|P_c\|_{e^A}}$$

,

where $\|P_c\|_{e^A} = \sqrt{< P - \frac{P\Sigma \text{II}}{< \Sigma, \text{II} >}, P - \frac{P\Sigma \text{II}}{< \Sigma, \text{II} >} >_{e^A}}$

Now all shapes are preshaped under the Σ-metric and from now on we consider all shapes to be preshapes.

3.2.3 Alignment

Similar to orthogonal alignment done in the classical procrustes analysis, we can find the best orthogonal alignment under Σ-metric as well. We find an orthogonal/rotation matrix $U \in O(n)$ that puts two preshapes in a position where corresponding landmarks of each shape is in the closest form. That is, we want to find $\hat{U} \in O(n)$ that minimizes

$$\hat{U} = \arg \min_{U \in O(n)} \|P - UQ\|_{\Sigma}^{2} \tag{3.12}$$

where

$$\|P - UQ\|_{\Sigma}^{2} =< P - UQ, P - UQ >_{\Sigma} \tag{3.13}$$

expanding (3.13) we get

$$< P - UQ, P - UQ >_{\Sigma} =< P, P >_{\Sigma} + < Q, Q >_{\Sigma} -2 < P, UQ >_{\Sigma} = 2 - 2 < P, UQ >_{\Sigma} \tag{3.14}$$

since both P and Q are preshapes $< P, P >_{\Sigma}= 1$ and $< Q, Q >_{\Sigma}= 1$. Minimizing (3.12) now turns into maximizing $< P, UQ >_{\Sigma}$, and $< P, UQ >_{\Sigma}$ can be rewritten as

$$< P, UQ >_{\Sigma} =< P\Sigma, UQ > =< P\Sigma Q^{T}, U > =< X, U > \tag{3.15}$$

where $X = (P\Sigma)Q^{T}$.

Let $X = U_1 \Lambda V_1^{T}$ be a singular value decomposition(SVD) of X, then similar to the result of orthogonal alignment step in classical procrustes analaysis we find $\hat{U}$ that optimally aligns P with Q as $\hat{U} = U_1 V_1^{T}$.

Finding mean shape in Σ-metric is similar to the Euclidean case; we use the attracting fixed point method and only change $\| \cdot \|$ to $\| \cdot \|_{\Sigma}$. In order to obtain the principal components in Σ-metric space, we first project each preshape onto the tangent plane at the mean shape. We then map the feature vectors at tangent space to Euclidean space using the idea of (3.3) and apply PCA algorithm in Euclidean space. Finally, we project the principal components found in Euclidean space back to the Σ-metric space.

After discussing the procedure of generalized Procrustes methods and statistical shape analysis under Σ-metric, we now define the objective function for the shape classification.

3.3 Learning Model of Shape Classification

The idea of shape classification is to keep samples belonging to the same class as close as possible while keeping different classes far away from each other. We will adapt the cost function introduced by Yu Fan (2012)[?] which is built upon the idea of linear discriminant analysis. LDA is a dimensional reduction technique in pattern classification problems, however, it is not exactly a dimensional reduction technique it is rather a representation of the data in a lower dimension which projects the high dimensional samples onto a line,plane or a lower dimensional sphere in which the class separation is in the best form.

We know introduce the components of the cost function S_W and S_B where S_W stands for within class scatter and by definition measures the spread of samples in each class. S_B is the between class scatter and calculates the distances between classes.

As noted before, enforcing Σ to be positive definite is inreasing the computational complexity in implementations, instead, we use a logarithmic representation by $\Sigma = \exp A$ with A symmetric and trace $A = 0$ to have $\det\Sigma = 1$

We want to estimate Σ so that the associated shape metric maximizes S_B while minimizing S_W. Assuming we have m clusters, we define our own **within-class scatter** and **between- class scatter** of shape data as follows:

Within-class Scatter:

$$S_w(A) = \sum_{i=1}^{m}\sum_{j=1}^{n_i} \|\hat{P}_{ij} - u_i\|^2_{e^A} \tag{3.16}$$

where P_{ij} is the jth training shape from ith class, $\hat{P}_{ij}$ is the aligned P_{ij} with u_i, and u_i is the mean shape of the training samples that belong to class i.

Between-class Scatter

$$S_B(A) = \sum_{i=1}^{m} n_i \|\hat{u}_i - u\|^2_{e^A} \tag{3.17}$$

where n_i is the number of training samples in ith class, $\hat{u}_i$ is the optimal alignment of u_i with u, and u is the mean shape of all training set.

We aim to find a weight matrix Σ that can separate different classes very well. That is, we want to find A which maximizes S_B and also minimizes S_W. Thus, the "best" A should be the minimizer of the cost function defined by

$$\frac{S_W(A)}{S_B(A)} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} \|\hat{P}_{ij} - u_i\|^2_{e^A}}{\sum_{i=1}^{m} n_i \|\hat{u}_i - u\|^2_{e^A}} \tag{3.18}$$

where A is symmetric and trace$A = 0$.

We search the optimal A, the minimizer of $S_W(A)/S_B(A)$, on the subspace of symmetric with zero trace matrices. As the function in (3.18) decreases, it is quite possible for the function to go 0 since we can find a symmetric matrix A with zero trace that may set $S_W(A)/S_B(A) = 0$. This happens when A moves off to infinity, i.e. one or more eigenvalues of $\Sigma^{-1} = e^A$ become large as others decay in order to keep the determinant 1. To remove this possibility we add a quadratic regularization term to (3.18) which balances the eigenvalues. The proposed cost function is therefore

$$F(A) = \frac{S_W(A)}{S_B(A)} + c\|A\|^2, \tag{3.19}$$

where $c > 0$ is a constant and we choose c such that it does not dominate S_W/S_B otherwise the only term minimized in (3.19) would be the regularization term.

In the next chapter we introduce the method that we use to find the optimal A by minimizing the cost function (3.19) via simulated annealing method.

CHAPTER 4

MONTE CARLO OPTIMIZATION IN HIGH DIMENSIONS

In this chapter, we will analyze some of commonly used Monte Carlo optimization methods. We will first give the theory and algorithm of each method along with a discussion of their advantages and disadvantages. We will then compare the preformance of methods on some high dimensional test functions.

4.1 Simulated Annealing

Simulated annealing was developed in 1983 by Kirkpatrick et al.[**?**] and can deal with highly nonlinear models, chaotic and noisy data. The method is an adaptation of the Metropolis-Hastings algorithm introduced by N. Metropolis et al. in 1953 [**?**], and improved by W.K.Hastings in 1970 [**?**]. Its main advantages over other local search methods are its flexibility and its ability to approach global optimality. Throughout the section we will give the theory and the practice of the simulated annealing method. We will also introduce the simulated annealing algorithm for the minimization of the cost function(3.19)

4.1.1 Motivation

Simulated annealing is a meta-heuristic local search algorithm capable of escaping from local optima and locating a good approximation to the global optimum of a given function in a large search space. While it is often used when the search space is discrete, it is being used for the continuous spaces as well. The traveling salesmen problem is one example of the discrete space in which a salesman visits a given set of cities with the minimum cost. Depending on the problem, simulated annealing computationally could be more efficient than any other heuristic and deterministic methods if the goal is to find an acceptably good solution while keeping computational complexity low, rather than having the best possible solution.

The name and inspiration come from its analogy to the process of physical annealing in metallurgy, a technique in which a crystalline solid is heated and then cooled in a controlled manner so that it allows a material to increase the size of its crystals and reduce their defects. If the cooling schedule is done very slowly, the final configuration results in a solid superior structure. Heating and cooling the material affects both the temperature and the thermodynamic free energy. Simulated annealing provides the connection between this type of thermo-dynamic behavior and establishes the search for global optimum for an optimization problem.

The method starts with an arbitrary state ω in search space Ω as the initial estimate of the optimal solution. Then a new state ω' is randomly selected from a neighborhood of ω. For a minimization problem, if $f(\omega') \leq f(\omega)$ so that ω' is a better state than ω, then the method accepts the move from ω to ω' and ω' becomes the new estimated optimal solution. Moving from one state to another in case the latter is better than the first is called improving solution since the new state is closer to the optimal solution. However, accepting only improving solutions could lead the optimal solution to one of local optima. Therefore, we sometimes accept non-improving solutions as well in order to escape from local optima in search of global optimum.

The probability of accepting non-improving solutions depends on a temperature parameter, which is typically non-increasing with each iteration of the algorithm in the search space. Accepting non-improving or worse solutions is a fundamental property of meta-heuristics because it allows a more extensive search for the optimal solution.

4.1.2 Definition of Terms

We need a few definitions in order to describe the specific features of simulated annealing algorithm for discrete optimization problems. First, let Ω be the *solution space* (the set of all possible solutions), and $f : \Omega \to \mathbb{R}$ be an *objective function* defined on the solution space. The aim is to find a global minimum/maximum, $\omega^* \in \Omega$ such that $f(\omega) \geq f(\omega^*)$ or $f(\omega) \leq f(\omega^*)$, respectively, for all $\omega \in \Omega$. To ensure the existence of ω^* the objective function must be bounded . Define $N(\omega)$ as the *neighborhood function* for $\omega \in \Omega$. Associated with every solution, $\omega \in \Omega$, $N(\omega)$ gives the neighboring solutions that can be reached in a single iteration of a local search algorithm. From this point on, we assume that the optimization problem is a minimization problem for which we seek the global minimum and this is the case in our applications, too.

The method starts with an initial solution in the search space, $\omega \in \Omega$. A neighboring solution $\omega' \in N(\omega)$ is then generated either randomly or based on some pre-specified rule. The candidate solution, ω', is accepted as the current solution based on the acceptance probability

$$P\{\text{Accept } \omega' \text{ as the next solution}\} = \begin{cases} \exp[-\frac{f(\omega')-f(\omega))}{t_k}] & \text{if } f(\omega') - f(\omega) > 0 \\ 1 & \text{if } f(\omega') - f(\omega) \leq 0 \end{cases} \tag{4.1}$$

where t_k is the temperature parameter at iteration k, such that

$$t_k > 0 \text{ for all } k \text{ and } \lim_{k \to +\infty} t_k = 0. \tag{4.2}$$

The acceptance probability is the basic element of the search mechanism in simulated annealing. If the temperature is reduced sufficiently slowly, then the system can reach an equilibrium (steady state) at each iteration k. Let us consider the objective function values $f(\omega)$ and $f(\omega')$ as the energies associated with states $\omega \in \Omega$ and $\omega' \in N(\omega)$, respectively. This equilibrium follows the

Boltzmann distribution, which gives the probability of the system being in state $\omega \in \Omega$ with energy $f(\omega)$ at temperature T such that

$$P\{\text{ System is in state } \omega \text{ at temperature } T\} = \frac{\exp(-f(\omega)/t_k)}{\sum\limits_{\omega' \in \Omega} \exp(-f(\omega')/t_k)} \tag{4.3}$$

Let us denote the probability of generating a candidate solution ω' from the neighbors of solution $\omega \in \Omega$ as $g_k(\omega, \omega')$. We have

$$\sum_{\omega' \in N(\omega)} g_k(\omega, \omega') = 1, \text{ for all } \omega \in \Omega, k = 1, 2, \ldots, \tag{4.4}$$

Define a non-negative square stochastic matrix $\mathbf{P}_k$ with transition probabilities as

$$\mathbf{P}_k(\omega, \omega') = \begin{cases} g_k(\omega, \omega')\exp(-\Delta_{\omega,\omega'}/t_k) & \omega' \in \mathbf{N}(\omega), \omega' \neq \omega \\ 0 & \omega' \notin \mathbf{N}(\omega), \omega' \neq \omega \\ 1 - \sum\limits_{\omega' \in \mathbf{N}(\omega) \& \omega' \neq \omega} \mathbf{P}_k(\omega, \omega') & \omega' = \omega \end{cases} \tag{4.5}$$

where $\Delta_{\omega,\omega'} \equiv f(\omega') - f(\omega)$ and boldface indicates matrix/vector notation, and all vectors are row vectors. These transition probabilities define a sequence of solutions for all solutions $i \in \Omega$ and all iterations $k = 1, 2, \ldots,$ generated from an inhomogeneous Markov chain [**?**].

4.1.3 Statement of Algorithm

Simulated annealing is outlined in pseudo-code (Eglese,1990)[**?**] as follows.

Algorithm 1: Simulated annealing algorithm

```
1. Select an initial solution $\omega \in \Omega$, initial temperature $T = t_0 > 0$, set two tolerances $\text{tol}_1$ and
   $\text{tol}_2$, and set the temperature counter $k = 0$.

2. Select a temperature cooling schedule $c$, and repetition schedule $M_k$ which defines the number
   of iterations executed at temperature $t_k$

3. **for** $i = 1 : M_k$

     • Generate a solution $\omega' \in \mathbf{N}(\omega)$
     • Compute $\Delta_{\omega,\omega'} = f(\omega) - f(\omega')$
     • **if** $\Delta_{\omega,\omega'} \geq 0$, then $\omega \leftarrow \omega'$
     • **else**, generate a random number $u$ from $U(0,1)$
     • **if** $u \leq \exp(-\Delta_{\omega,\omega'}/t_k)$, then $\omega \leftarrow \omega'$ and/**else** go to step 3-a

4. **if** $t_k < \text{tol}_1$ or $\Delta_{\omega,\omega'} < \text{tol}_2$, **stop**
   **else** update $t_{k+1} \leftarrow ct_k$ and **go to** step 3
```

Detailed expression of the algorithm can be given as

- At each temperature t_k, M_k number of iterations being executed and we find a new optimal solution. The new optimal solution is used as the initial solution in the next M_k loop and another optimal solution is found. For instance, for k=3 we need to execute M_3 number of iterations where M_3 is either given with the repetition schedule or as a constant value, and t_3 is found based on the cooling schedule. We should have an optimal solution from t_2 and we use it as an initial optimal solution for t_3. We then generate neigborhood solutions of the optimal solution and follow the step 3. This goes on until the stopping criteria is met.

- We increase k by one number outside the loop and change t_k based on the cooling schedule that we defined earlier. Cooling schedule c could be an arbitrary constant value in $(0, 1)$ in order to make sure it is cooling the previous temperature or it could be chosen as a function of iteration as well [**?**]. Then we use the new temperature t_k to find a new optimal solution. Note that as the temperature t_k decreases the probability of accepting non-improving solutions also decreases.

- The algorithm stops either when the temperature is too slow or when there is no remarkable difference between current and new optimal solution. Algorithm-1 is given for a minimization problem if the desire is to find the maximum, we only need to change the sign of quantity

$\Delta_{\omega,\omega'} \geq 0$ to $\Delta_{\omega,\omega'} \leq 0$ and the rest of the algorithm holds for the maximization problem as well.

- Selection of initial temperature and cooling schedule is a crucial matter in a simulated annealing algorithm. When they are chosen optimally the algorithm can locate the global minimum no matter what the initial solution is. On the other hand, choosing a non-appropriate initial temperature or a cooling schedule can cause the optimal solution to trap in a local minimum. Further discussion is given in the examples.

Based on the algorithm and the detailed expression, the simulated annealing formulation results in $M_0 + M_1 + \cdots + M_k$ total iterations being executed, where k corresponds to the value for t_k at which stopping criteria is met. In addition, if $M_k = 1$ for all k, then it is obvious that the temperature changes at each iteration.

4.2 Particle Swarm Optimization

Particle Swarm Optimization(PSO) is another stochastic optimization method introduced by Kennedy and Eberhart [?] in 1995. Similar to simulated annealing, it is also a nature inspired algorithm, particularly biologically derived algorithm. The algorithm imitates the social behavior of bird flocking or fish schooling.

PSO is a population based optimization method in which the system is initialized with a population of random solutions where the population is known as *swarm* and each random solution is called *particle*. It is a global optimization method which is well suited to solve high dimensional problems [?].

Since its introduction, particle swarm optimization has become very popular in many different research areas. Its metaheuristic nature, which deals with non-differentiable, noisy, and dynamic problems efficiently, has made the method more and more attractive. Due to its capacity and simplicity in dealing with non-differentiable optimization problems, PSO algorithm has been widely used in calculation of neural networks weights, multi-objective optimization, classification, time series analysis, pattern recognition, business optimization, and in many other fields [?], [?].

As stated, the algorithm mimics the behavior of the animals who act in groups where there is no leader in the group to lead the swarm. These group of animals, such as flocking birds and schooling fish, oftentimes act together in search of food or avoiding the predators. In the case of finding food, animals communicate with each other and change their positions simultaneously with respect to a member of group who has the closest position to a food source. Through this type of communication among its members, the swarm changes its position frequently and finally achieves the closest point to the food source.

Particle swarm optimization is also related to evolutionary computation techniques such as Genetic Algorithms in the sense that the system is initialized with a population of random solutions and each candidate solution is searching for an optimum solution. They also share some other commonalities such as updating the population and using randomization in search for the optimum. Both methods share information among the members of population, although it is done differently in two algorithms.

Compared to genetic algorithms, particle swarm method functions differently as there is no evalutionary steps in the search process such as crossover and mutation. Moreover, unlike a GA, each candidate solution is assigned a randomized velocity and random solutions, particles, freely move in the search space [**?**]. Another difference between the two methods found on the information sharing system of the methods. In GA, all chromosomes share information with each other and the algorithm moves the population to a certain direction based on the information obtained. However, in PSO, only the members who are closest to the optimal area share the information with other members of the group.

4.2.1 Algorithm Formulation

In the algorithm formulation process we adapt the notation used in [**?**] and [**?**]. We first start with introducing the main components of the algorithm; particle, velocity, population, and search mechanism.

Let d be the dimension of the problem in which an optimal solution is to be found. A particle $\mathbf{x}_i$ can be described as a point in $\mathbb{R}^d$ as

$$\mathbf{x}_i = [x_{i1}\ x_{i2}\ \cdots\ x_{id}]$$

.

Let N be the population size, which is the number of particles. Then we can represent the population as

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$$

Each particle moves with its own velocity and the velocity itself is also effected by the information obtained from the member of group who is closest to the optimum solution. This motion of particle can be given in its simplest form as

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$$

where $\mathbf{v}_i$ is the velocity vector of particle $\mathbf{x}_i$ and t and $t+1$ are two consecutive iterations at which the particle $\mathbf{x}_i$ is located.

Movements of particles are determined by the velocity vector and the velocity of ith particle is updated by the following equation

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1(\mathbf{p}_i - \mathbf{x}_i(t))r_1 + c_2(\mathbf{g} - \mathbf{x}_i(t))r_2$$

Now let us look deeper into the each parameter in velocity equation: $\mathbf{p}_i$ is the *personal best*, also called *pbest*, of the ith particle, i.e. best solution obtained by $\mathbf{x}_i$ so far, while $\mathbf{g}$ is the *global best* solution found up until time t. That is, the best solution of the personal bests, also known as *gbest*. c_1 and c_2 are called *cognitive coefficient* and social coefficient, respectively. Cognitive coefficient, c_1, is responsible from keeping the track of particle's own path and its an important component of velocity. Social coefficient, c_2 is in charge of moving the particle in the direction of global best, or *gbest*. r_1 and r_2 are random coefficients, usually generated from uniform distribution in $[0, 1]$, that give the particles flexibility to move around the search space both deterministically and randomly at the same time.

Before we give the pseudo-code for the algorithm we need to take the followings into consideration:

First off, as in any global optimization problem, the search space must be bounded by the lower and upper limits so that the initial starting points fall in this bounded region. That is, each particle must satisfy $LB \leq x_i \leq UB$. In addition to bounded region, it is desired to distribute the initial positions of particals as evenly as possible to guarantee that the search space is adequately covered.

In the algorithm, the number of particles, N, also plays an important role. It it necessary that N is not too large and not too small either. Although when N is large we can expect better optimum solutions, it, however, leads to computational issues. Obviously, when N is small the method may not explore search space sufficiently before the stopping critearia is met.

Final consideration in the algorithm is the stopping criteria. Every researcher can define his/her own stopping criteria and we will list some of the most commonly used criteria in terminating the algorithm here. One criterion is to check whether the algorithm exceeds the maximum number of iterations or function evaluations which is usually user defined. Another common criterion is to check whether any new position improves the function's best value obtained so far. If the best solution is not improved for a certain number of iterations, user can stop the algorithm. In addition to these criteria, one can also check whether each particle's best position is very close to the global best solution. That is, if each particle is converging to the optimum value. This optimum value may be or may not be the global optimum since there is no mechanism to avoid the local optimum[**?**].

Algorithm 2: Particle Swarm Optimization Algorithm

1. Generate N particles with initial positions $x_1(0), x_2(0), \cdots, x_N(0)$.
2. Compute the loss function at each position, that is calculate $f(x_i(0))$ for each i. Set each particle's initial position to its best position, that is, $x_i(0) = p_i$ and also set the smallest function value $f(x_j(0)) = g$, i.e. $f(x_j(0)) < f(x_i(0))$ for $\forall\ i \neq j$.
3. Select c_1 and c_2 and set counter $t = 0$.
4. **DO**
    - Generate two random numbers r_1 and r_2.
    - Update each particle's velocity by
      $\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1(\mathbf{p}_i - \mathbf{x}_i(t))r_1 + c_2(\mathbf{g} - \mathbf{x}_i(t))r_2$
    - Update particles' position with updated velocity
      $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$
    - if $f(x_i(t+1)) > p_i$, then update $p_i \leftarrow f(x_i(t+1))$.
    - if $f(x_i(t+1)) > g$ for some i then update $g \leftarrow f(x_i(t+1))$
5. **If** stopping criteria is met, **stop**.
   Else go to Step 4.

Algorithm above is the basic PSO algorithm that is modified over the years and here we will give a brief discussion on what are the challenges with the basic algorithm and what have been proposed to overcome those challenges.

Speed of Velocity: With the basic algorithm velocity of a particle can get arbitrarily larger and larger values with certain parameter selection. *Velocity clamping* is introduced to limit the speed of particles to a maximum velocity, i.e. v_{max}. If one particle's speed exceeds v_{max} its speed will be replaced by maximum velocity[**?**].

Inertia Weight: It is another mechanism that controls the velocity of particles by penalizing velocity at previous position. The updated velocity formula becomes

$$\mathbf{v}_i(t+1) = \omega\mathbf{v}_i(t) + c_1(\mathbf{p}_i - \mathbf{x}_i(t))r_1 + c_2(\mathbf{g} - \mathbf{x}_i(t))r_2$$

Here ω is the inertia weight and can be selected as a constant or a function of time as well [?], [?]. When selected constant and if $\omega > 1$ then it is evident that the speed of particle will always increase and this will result in exploring the search space utmostly. Yet, it will lead the particles to diverge and the algorithm will produce nonconvergent results. On the other hand, when $0 < \omega < 1$ the velocity of each particle will approach to zero and this will lead an exploitation of the local region by each particle. Due to the nature of selecting constant inertia many researchers proposed a dynamic intertia weight where the weight ω is sometimes greater than 1, so that particles explore the search space widely, and sometimes it is less than 1 to give the particles the flexibility of exploitation.

4.3 Multi-Start Methods

Thus far we have introduced two global optimization methods; simulated annealing, a probabilistic technique, and particle swarm optimization, a population based optimization method, also considered as an evolutionary algorithm. As a final global optimization method in this section, we present a third class of algorithms *multi-start methods*.

Multi start methods, as the name implies, are methods in which an optimization is started at different points in the search space. Therefore multi-start methods have two phases in optimization process: a *global phase* and a *local phase*.

In the global phase the goal is to sample the initial states- from which a local search will be inititated- as evenly as possible. Once the initial states determined then a local search is started from the very same state.

Multi-start methods are well suited for multimodal problems- functions with many local minima. Local searches can be done by deterministic methods such as gradient descent and quasi-newton methods if the function is differentiable. It can also be applied to non-differentiable functions where gradient of the function is no longer exist or computationally expensive. Such derivative-free local search methods are simplex method, pattern search and random search to name a few.

There are many variants of multi-start methods in the literature, [?], [?]. It is yet beyond the scope of this study to go over each method. Instead, we will introduce one popular method in the class of multi-start methods, *multi level single linkage*.

Multi level single linkage method is a well known multi start method and the name 'single linkage' comes from its clustering structure [?],[?]. Let us first introduce the clustering part of multi level single linkage(MLSL). However, in order for the reader to understand the purpose of clustering we need several concepts in advance.

In a global optimization problem for a highly multimodal objective function, the most naive and trivial approach would be pure random search. In pure random search one generates N samples in the design space and calculates the corresponding function value. Although we do not perform any local or global search with this approach we can still reach to many local optima. It is also no surprise that after finite or infinitely many sampling, one would obtain the global optimum which basically corresponds to the best local optimum. We can put the ability of this approach to reach global optimum in mathematical terms as stated in [**?**].

Let $x_N^{(1)}$ be the smallest function value found in the search space with the sample size of N. Then we can prove the following statement although we will skip the proof.

Proposition: If $f(x)$ is continuous on the search space, then $x_N^{(1)}$ converges to the global optimum value x^*, i.e. $f(x^*) \leq f(x_i)$ for $i = 1, \cdots, N$, with probability 1 or almost surely with increasing N, that is

$$Pr\Big[\lim_{N \to \infty} x_N^{(1)} = x^*\Big] = 1$$

This method is overwhelmingly simple yet computationally disturbingly inefficient. It can be considered as the slowest and the weakest optimization method and the reason it is mentioned here is that the idea of multi start methods started from this point and both theoretically and computationally have been extremely improved.

An improvement to the pure random search would be the following simple multi start method[**?**].

Step1. Select a point from the search space Ω

Step2. Start a local search from selected point.

Step3. Repeat steps 1 and 2 until a stopping criteria is met.

The point that gives the smallest function value of all local searches will be x^*. This method is an improved variant of pure random search and proposition above also holds for this method as well.

Even though this method is an improvement, it is still inefficient since it misses several considerations in global search. One such consideration is the possibilty of several local searches that lead the same local minimum. Another concern is the spread of local searches, that is, how the starting point of each local search is selected. Clustering analysis adresses these issues and a detailed discussion is given in [**?**], [**?**]. Here we will only review "single linkage" clustering method.

In single linkage clustering analysis a number of points are sampled in the search space and the function values at each point are calculated. Afterwards, a local search is initiated from w smallest function values where w is a predetermined cluster number. Newly generated points are first checked if they are within a distance to one of the clusters already formed. If not, they are assigned to a new cluster. Here we present the algorithm of a slightly modified variant of single linkage method given in [], []. The modified variant is so-called " simple linkage" method.

Algorithm 3: Multi Level Single Linkage Algorithm

```
1. set counter $k := 1$, select $\sigma > 0$ and $\epsilon > 0$
2. let
   $$r = r_{k,\sigma} = \pi^{-1/2}\Big(\Gamma(1+d/2)\sigma\frac{\log k}{k}\Big)^{1/d}$$
3. Generate a single random point $x_{k+1}$ in the search space $\Omega$
4. Start a local search from $x_{k+1}$ **if** $f(x_j) - f(x_{k+1}) \leq \epsilon$ for $\forall j < k$ **or** $\|x_{k+1} - x_j\| > r$
5. **If** stopping criteria is met, **stop**.
   **Else** let $k = k + 1$ and go to Step 2.
```

Step 3 in the above algorithm corresponds to the global phase where the samples in the search space are generated. Taking step back, we should note that sampling distributions in multi-start methods play an important role. It is shown that low discrepancy sequences, i.e. Quasi-Monte Carlo (QMC) sequences, are well suited for multi start methods since they generate well distributed random points in the search space. It is also argued that for moderate size dimensions samples generated by QMC sequences may provide better results than that of uniform random samples[**?**].

Step 4 checks two conditions: First, it investigates whether newly generated point falls into any cluster that is previously defined. If not, then a new local search is performed from that point and a cluster is formed automatically around that candidate point. It also checks that if the new sample falls into one of the clusters yet its function value is the smallest in that cluster, it starts a local search hoping that it can reach a better local minimum.

After introducing three different class of stochastic global optimization methods, we will next evaluate the performance of each method by applying them to some well-known test functions for global optimization problems.

4.4 Test Functions

In this section we will apply the optimization algorithms introduced in previous sections to some test functions which are considered to be benchmark functions for optimization problems. Since our objective function is a high dimensional function we will focus on the performance of each method in high dimensional test functions and we will pick the one that is both computationally efficient and locates optimum solutions to use in our shape study.

We must note that since the shape of our cost function is unknown, whether it is unimodal or multimodal, we will test on two type of test functions, namely unimodal and multimodal. Modality in this context refers to the function's shape and checks if the function has any local optima other than global optimum. As the name implies multimodal functions have many local optima and these functions are challenging in optimization with some methods since it is very easy to obtain a local optimum and stop the algorithm in search of global optimum.

Ackley Function

$$f(x) = -a\ exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + exp(1)$$

Ackley function is very popular among researchers in optimization as it has many local minima. Due its multimodality, many optimization methods fail to locate global minimum.

Global minimum is located at $\mathbf{x^*} = (0, 0, \cdots, 0)$ and $f(x^*) = 0$.

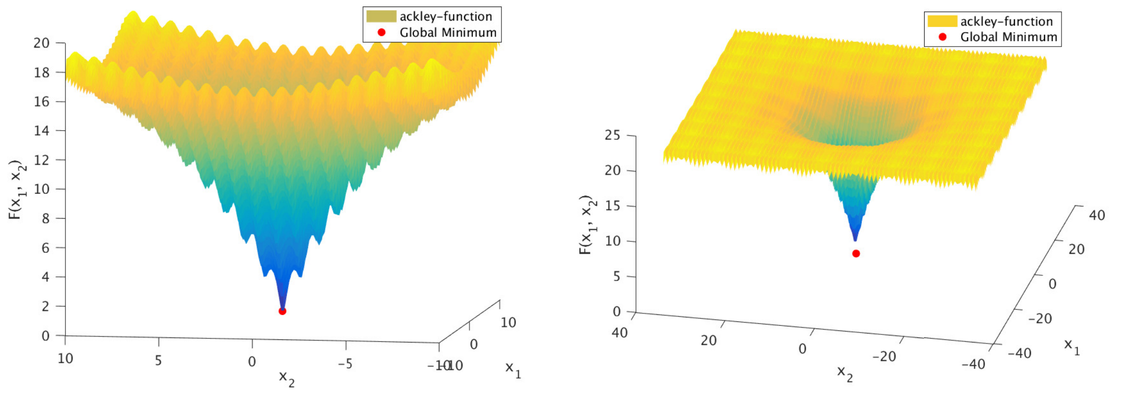
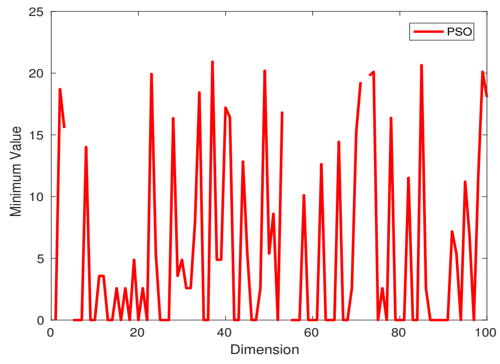
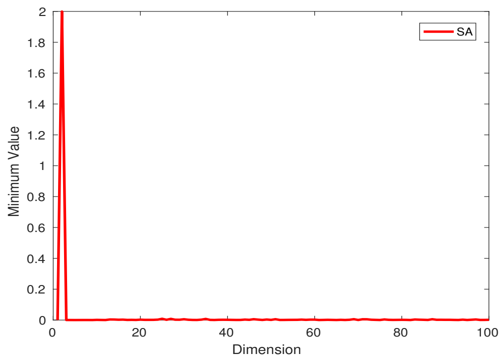


Figure 4.1: 2-D plot of Ackley function in different scales

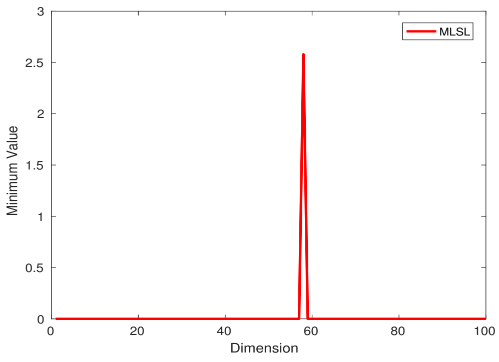
We tested each method on every dimension from 2-dimensional space to 100-dimensional space. It is clear that simulated annealing and multi level single linkage perform quite well on finding global minimum even in very high dimensions. On the other hand, it is evident that particle swarm



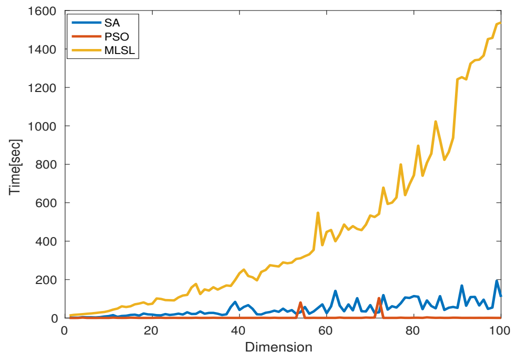
(a) Minimum value obtained by PSO



(b) Minimum value obtained by SA



(c) Minimum value obtained by MLSL



(d) Computing time of 3 methods

Figure 4.2: Comparison of 3 methods- Particle Swarm, Simulated Annealing and Multi Level Single Linkage- on Ackley function

optimization is trapped on one of local minima at many times. This is due to the nature of particle swarm optimization where it has no mechanism to avoid local optimum. However, if we check each methods timing, basically the time it takes for a method to stop algorithm, as shown in Fig.4.2d, we see that MLSL is the worst of all. This is again due to the nature of MLSL where it runs a local search at different points in the space and as space expands it results in more local searches in higher dimensional space which slows down the algorithm even more.

We also observe from Fig.4.2d that best timing results belong to particle swarm optimization as it takes slightly less time for the algorithm converge to a solution compared to simulated annealing. Oftentimes there is a trade off between an algorithm's ability to find an optimum solution and its speed in doing so. In this test function we encounter such a trade off. Although simulated annealing takes longer to reach an optimum solution compared to particle swarm, the time difference is negligible due to its performance on locating global optimum almost all the times where PSO is trapped to a local optimum mostly.

On the contrary, one needs to put a careful consideration into making such trade off between MLSL and PSO where one(MLSL) outperforms in finding global minimum and the other(PSO) outperforms significantly in timing. In these situations it is obvious that one will favor one method over the other based on priorities. That is, if the acceptably good solution is rather good enough than locating the best solution one can proceed with PSO. One must choose MLSL over PSO if timing is not the issue yet the final value matters the most.

Rosenbrock Function

$$f(x) = \sum_{i=1}^{d-1} \left[100(x_{i+1}^2 - x_i^2)^2 + (x_i - 1)^2 \right]$$

The Rosenbrock function is another popular test function. Because of its shape, it is sometimes referred to as Valley or Banana function, too. Its local optimum value is also the global optimum therefore it is in the class of unimodal functions. Although it is valley shaped, it is relatively difficult to reach global optimum.

Global minimum is located at $\mathbf{x^*} = (1, 1, \cdots, 1)$ and $f(x^*) = 0$.

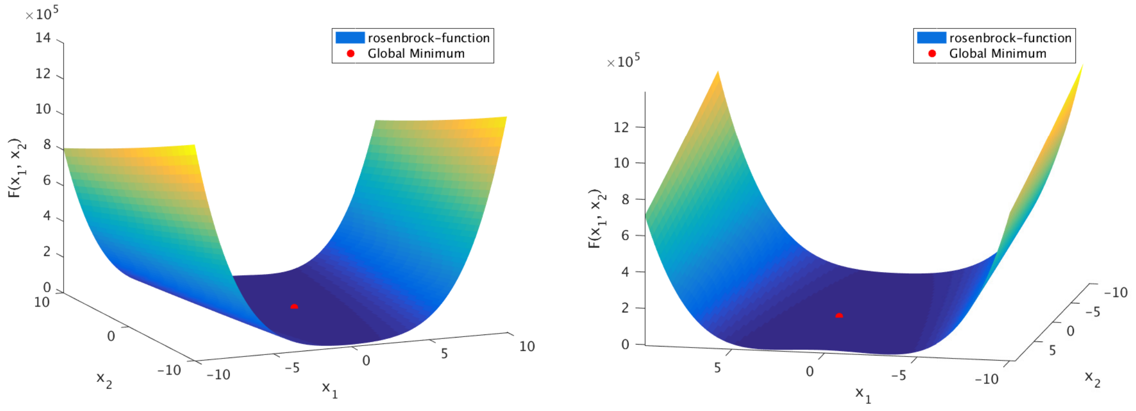
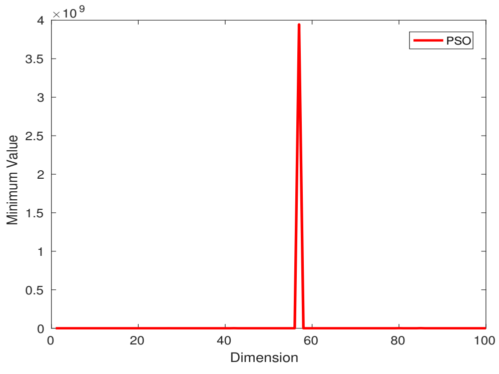


Figure 4.3: 2-D plot of Rosenbrock function

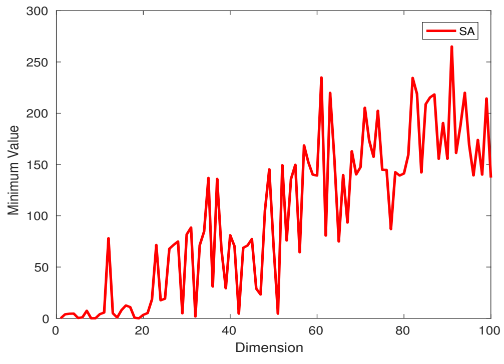
We ran the same experiment with the Rosenbrock function and observed the followings:

In terms of computational time, as shown in Fig.4.7d, the particle swarm is again the winning method and MLSL is taking more and more time as dimension increases. Simulated annealing is very slightly increasing with respect to dimension increments.

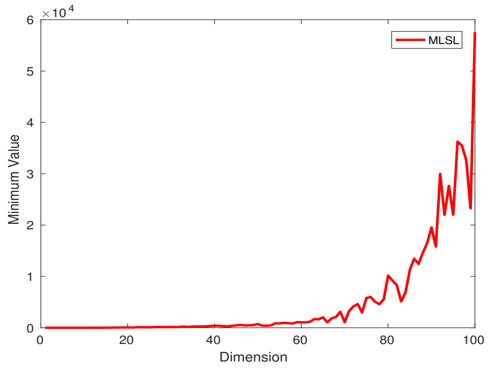
Scale factor in Fig.4.7a may lead a misinterpretation in terms of particle swarm's performance. We would make healthier comparisons if we ignore that single worst outcome obtained at dimension 57. When we replot without that 'outlier' we get the plot in 4.5a. Even though we removed



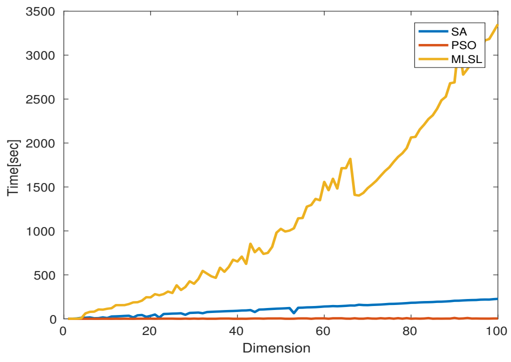
(a) Minimum value obtained by PSO



(b) Minimum value obtained by SA



(c) Minimum value obtained by MLSL



(d) Computing time of 3 methods

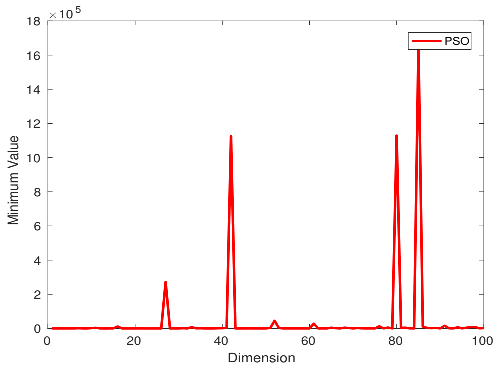
Figure 4.4: Comparison of 3 methods on Rosenbrock function

the biggest ‘outlier’ we still see some worse solutions which exceeds 10^5 and even 10^6. However minimum values obtained by simulated annealing is always less than 250. Average minimum values and average computing time for each method is shown in table4.1.

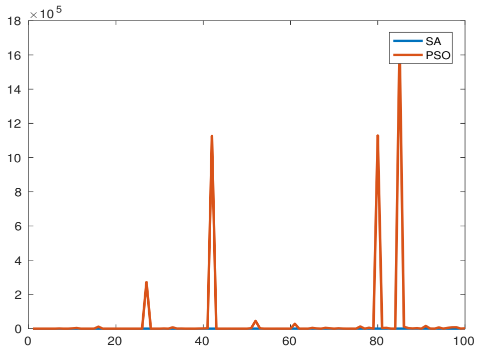
Another observation one can make from these results is that as dimension of the function increases multi level single linkage performs more and more poorly. As in the case of Ackley function simulated annealing is able to locate better optimum values, however, it is slightly slower than PSO.

So far, we have tested performance of each method on one unimodal, Rosenbrock, and one multimodal, Ackley, function. Although in both test functions simulated annealing performed better than the remaining two methods in finding optimum solutions, there is still not enough information to make an argument whether simulated annealing is a “better” method than the others. We must note that it is beyond the scope of our research to conclude that one method is “better” than the other. Our goal is to determine a method that produces good results with these test functions so that we can utilize it in our shape study. We must also note that one can-most of the times- make one method better than the others by changing certain parameters in each method to make sure that proposed method performs better. In our study, selected parameters are default parameters used in MATLAB, and they remain the same and unchanged throughout the experiments.

We will close up this chapter by analyzing an additional unimodal and multimodal test function.



(a) Minimum value obtained by PSO without outlier



(b) SA vs PSO comparison

Figure 4.5: Particle Swarm Optimization vs Simulated Annealing

Table 4.1: Comparison of 3 Methods with Rosenbrock Function

Method	Average Minimum Value	Average Computing Time(sec)
PSO	44207	2.4113
SA	98.4914	112.73
MLSL	5049.2	1190.5

Rastrigin Function

$$f(x) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i) \right]$$

Rastrigin function is a multimodal test function and it is also a very popular function since it has many local minima. The domain of the function is usually bounded by the hypercube $x_i \in [-5.12, 5.12]$

Global minimum is located at $\mathbf{x^*} = (0, 0, \cdots, 0)$ and $f(x^*) = 0$.

One thing that we realize immediately in the performance of particle swarm method on Rastrigin function is that it has an ‘outlier’ similar to Rosenbrock function. Outlier in this context refers

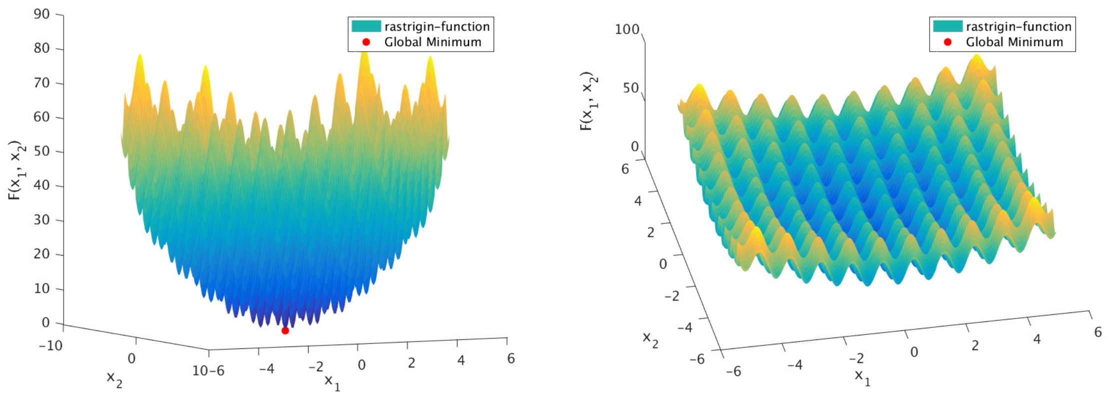


Figure 4.6: 2-D plot of Rastrigin function

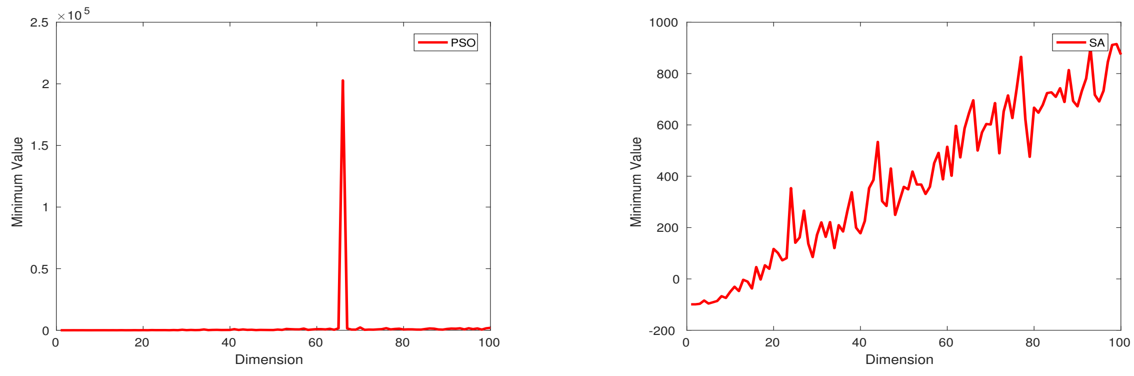
to a solution that is far different than the others. In this case that solution is greater than 20000 and if we replot the same results without this extreme local minima we will get a plot in 5.1a. In this test function we also notice that all 3 methods attain larger and larger local minimum values in search of global minimum as the dimension increases. This is no surprise since Rastrigin function is higly multimodal and it is very normal for a method to get trapped in one of many local minima during the search.

In terms of computational time of each method we see a similar pattern to the functions analyzed earlier,.i.e Ackley and Rosenbrock. MLSL takes the most time to find an optimal solution while PSO is again the fastest method among all. Simulated annealing is not extremely slow like MLSL and it is almost as fast as particle swarm in lower dimensions and as dimension increases it gets slightly slower than PSO.

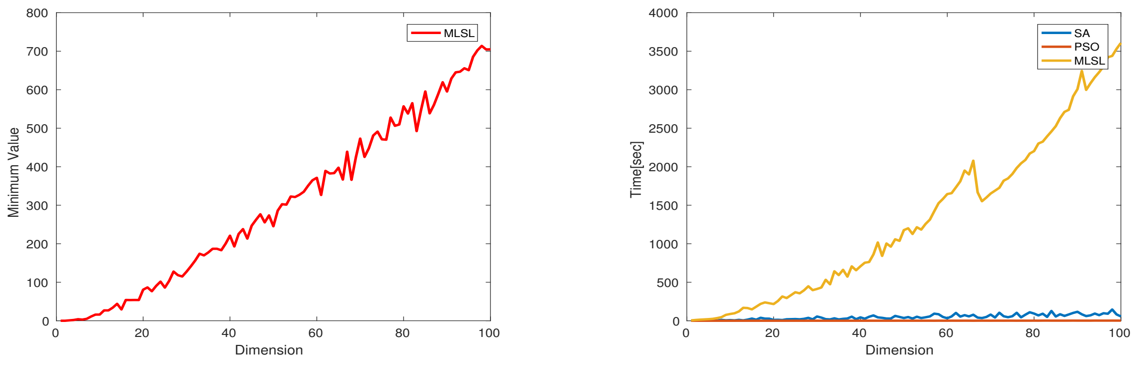
Analyzing final solutions obtained by each method yields the following observation. The optimum solutions obtained by PSO is very volatile and it is most of the times higher, namely worse, than simulated annealing, see Fig.5.1b. Another observation regarding final optimum results is multi level single linkage produces the best results on Rastrigin function. Table 4.2 summarizes these observations.

Table 4.2: Comparison of 3 Methods with Rastrigin Function

Method	Average Minimum Value	Average Computing Time(sec)	Standard Deviation
PSO	556.62	5.93	526.32
SA	378.29	47.77	299.26
MLSL	302.7373	1307.2	217.86



(a) Minimum value obtained by PSO (b) Minimum value obtained by SA



(c) Minimum value obtained by MLSL (d) Computing time of 3 methods

Figure 4.7: Comparison of 3 methods on Rastrigin function

Sphere Function

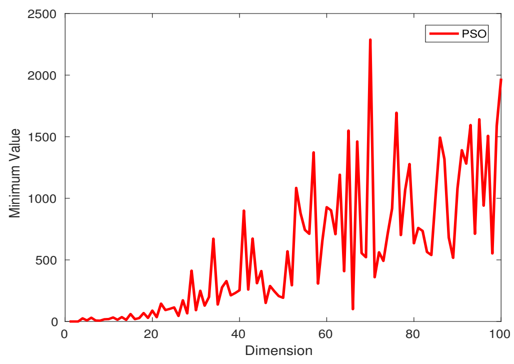
So far we have studied 3 test functions one being unimodal and two were multimodal test function. As a final test function we will analyze another unimodal test function, i.e. *Sphere Function*

Sphere function is a unimodal test function and is strongly convex, see Fig.4.9. It is usually evaluated in the search domain of $x_i \in [-5.12, 5.12]$.

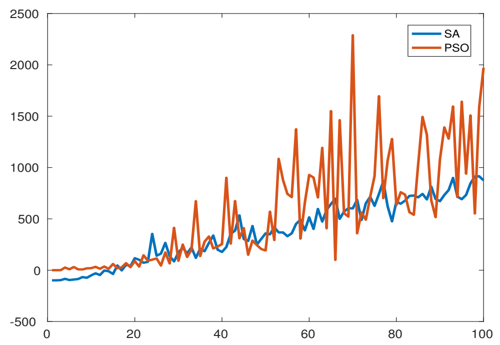
$$f(x) = \sum_{i=1}^{d} x_i^2$$

Global minimum is located at $\mathbf{x^*} = (0, 0, \cdots, 0)$ and $f(x^*) = 0$.

To avoid the repetition we will only summarize the computational results instead of illustrating results with plots. Timing of each method on Sphere function is very similar to that of Rastrigin



(a) Minimum value obtained by PSO without outlier



(b) SA vs PSO comparison

Figure 4.8: Particle Swarm Optimization vs Simulated Annealing

function. Multilevel single linkage takes the most of the time and particle swarm is again the fastest algorithm among the three. Simulated annealing competes with PSO in lower dimensions however as dimension increases it slightly gets slower than particle swarm optimization method.

In this test function, we observe that particle swarm reaches global optimum most of the time while simulated annealing and multi level single linkage locates global minimum up to dimension 20. As we increase the dimension both simulated annealing and multi level method get higher values, the two methods worsen in higher dimensions though not dramatically.

Summary of final optimum solutions obtained by each method is given in table 4.3.

Table 4.3: Comparison of 3 Methods with Sphere Function

Method	Average Minimum Value	Average Computing Time(sec)
PSO	1.64e-12	1.34
SA	9.77	38.73
MLSL	19.97	1192.1

Although we see that simulated annealing and multi level single linkage method is unable to locate the global minimum in average, their performance is very satisfactory as the global optimum values they located range from 0 to 40 and 0 to 50, respectively. These numbers are very small compared to the capabilities of Sphere function since it is the square of elements in each dimension and can get extremely large values in higher dimensions.

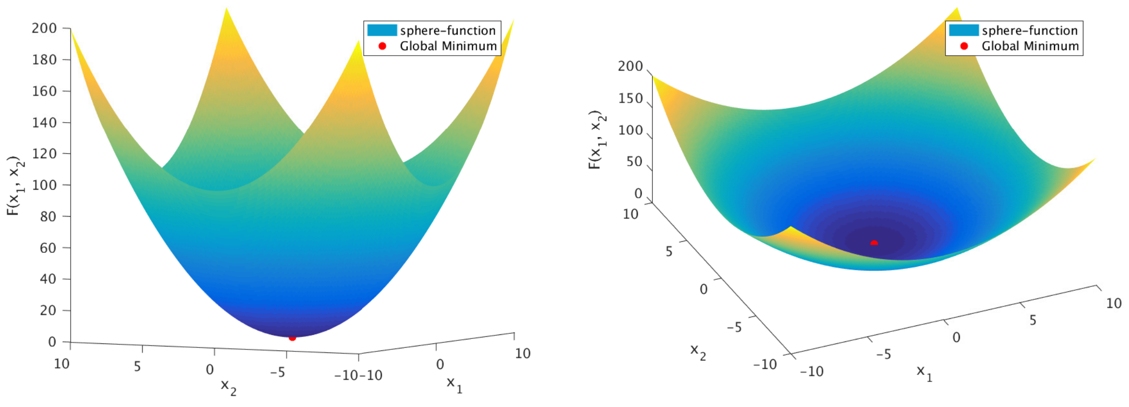


Figure 4.9: 2-D plot of Sphere function

We have tested the performance of 3 methods, namely particle swarm optimization, simulated annealing and multilevel single linkage. We have repeatedly observed that particle swarm optimization method was the fastest of all three in every test function considered here. However, we also witnessed that simulated annealing is approaching global optimum better than particle swarm optimization in all test functions except at Sphere function in which the results were close.

Each of these methods represent a class of algorithms and therefore proceeding with one in shape study is crucial and was ultimate goal of our benchmarking process. Surely one can work on other methods and other test functions and as a result can draw completely different conclusions and therefore can choose a different winning method. We must note that again, the aim in this benchmarking is not to pick the "best" model, it is rather to compare several well known stochastic global optimization methods and select a reasonably good performing method that is to be used in shape study.

After analyzing the test functions with the methods in hand we see that although simulated annealing is slightly slower than particle swarm optimization, it outperforms the latter method in most instances. We believe that this is primarily due to its nature of accepting worse solutions in search of a better solution as this mechanism moves the current solution from a local optimum most of the times.

Moreover, when we also consider the fact that simulated annealing is only running one solution path, that is it starts with a single initial state in high dimensional space and moves to the global optimum, its performance can be considered invaluable. On the other hand, since there is no structure in particle swarm optimization that avoids to get trapped into a local minima, it usually is trapped to one as seen in Ackley function. Particle swarm method's advantage over simulated annealing is that it starts with a population of particles where each particle is just an initial state and searches global optimum.

In our test functions the number of particles was 100. To make an analogy, one can think of the comparison this way: consider a big forest and assume that an item is lost in the forest and no one has any idea where the item is located. Simulated annealing can be depicted as a single person who is searching the entire forest by himself/herself while particle swarm optimization can be thought as a group of 100 people and they search for the same thing. This analogy enhances the importance and power of simulated annealing. We can emulate the same analogy and apply it to any optimization problem whether on a test function or a real world problem.

Based on our observations on the performance of three methods on test functions we choose simulated annealing to be our Monte Carlo optimization method that will be implemented in our shape problem.

Next we give simulated annealing algorithm for our shape metric learning problem.

Algorithm 4: Learning Weight Matrix with Simulated Annealing

```
1. Select an initial zero trace symmetric matrix $A$,i.e. $tr(A) = 0$, initial temperature $t_0 > 0$. Set
   two tolerances $\mathrm{tol}_1$ and $\mathrm{tol}_2$, and set the temperature counter $k = 0$.
2. Select a temperature cooling schedule $c$, and repetition schedule $M_k$ which defines the number
   of iterations executed at temperature $t_k$
3. Turn raw shapes into preshapes using the weight matrix $\Sigma = e^A$ and calculate mean shape
   of each group and global mean shape.
4. Align each preshape with its own group mean shape and also align each group mean shape
   with the global mean shape.
5. Calculate the cost function $F(A) = \frac{S_W(A)}{S_B(A)} + c\|A\|^2$
6. **for** $i = 1 : M_k$
     • Generate a new symmetric matrix $A'$
     • Follow steps 3-5
     • Compute $\Delta_{A,A'} = F(A) - F(A')$
     • **if** $\Delta_{A,A'} \geq 0$, then $A \leftarrow A'$
     • **else**, generate a random number $u$ from $U(0, 1)$
     • **if** $u \leq \exp(-\Delta_{A,A'}/t_k)$, then $A \leftarrow A'$ and/**else** go to step 5-a
7. **if** $t_k < \mathrm{tol}_1$ or $\Delta_{A,A'} < \mathrm{tol}_2$, **stop**
   **else** update $t_{k+1} \leftarrow ct_k$ and **go to** step 5
```

In our problem, the space Ω of solutions consists of $n \times n$ symmetric matrices A that satisfy the growth constraint in (3.19). Given the current state $\omega = A$, the proposed new state A' is obtained by adding a random symmetric matrix of the same size to A. The entries of the random symmetric matrix are generated at random from the uniform distribution on $(-1, 1)$. We should note that A' is not the transpose of A it is rather a neighbor state.

CHAPTER 5

LANDMARK SPARCIFICATION

In this chapter we will adress two additional issues that arise in the implementation of the model introduced in chapter 2. Recall that the objective function to be minimized is

$$F(A) = \frac{S_W(A)}{S_B(A)} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} \|\hat{P}_{ij} - u_i\|_{e^A}^2}{\sum_{i=1}^{m} n_i \|\hat{u}_i - u\|_{e^A}^2} \tag{5.1}$$

where the optimal weight matrix $\Sigma = e^A$ is the minimizer of this function. The search space is all trace zero n x n symmetric positive matrices. Therefore we can consider the dimension of the objective function as $(n^2 + n)/2$. Obviously as n gets larger and larger, that is if we are dealing with shapes that have many landmark points on it, the computational cost will further increase. Essentially we will encounter the curse of dimensionality. In the previous chapter we have observed that even though simulated annealing seems to be a powerful tool in optimizing high dimensional loss functions, computational efficiency slightly decreases as the dimension enlarges.

In our shape study only parameter that effects the dimension of the objective function is the number of landmarks. Suppose that we are working on shapes with 20 landmarks, which is a moderate number for landmarked shapes. Having 20 landmarks, $n = 20$, with 2-dimensional shapes the dimension of the search space turns into $20 * 21 = 420$ dimensional space which is enormous.

One remedy to this problem is to reduce the number of landmarks through landmark sparcificataion and minimize the cost function in lower dimensional space and then approximate the optimal full weight matrix by interpolating the estimated weight matrix.

The second practical issue of the model is the number of shapes that is used to calculate the objective function. In the case of abundant shapes, computational burden of $F(A)$ will increase and number of shapes will have adverse affect in the implementation. One solution to this problem is to use a small number of training shapes in the optimization process and compare the estimated weight matrix with the matrix that is obtained using the entire training shapes. We will start with landmark sparcification and move to the remedy of the second issue afterwards.

5.1 Landmark Sparsification

In landmark sparcification the idea is to use a subset of landmarks in shapes and calculate the weight matrix with the reduced landmarks. Next, interpolate the weight matrix that is obtained with reduced number of landmarks to the full weight matrix where the interaction between every landmark is revealed.

This idea is particularly useful in shapes with many landmarks and with shapes or images where landmarking is a difficult task. Recall that shapes can be landmarked either by hand or using some software techniques. In the former case it is evident that it is not practical and not reproducible either, resulting less landmark points and smaller sized populations. In the latter case, the most difficult task is to register landmarks so that each numbered landmark corresponds to the same point on the shape. There are some techniques developed for this task and it is beyond the scope of the present discussion.

Landmark sparcification is a two-fold approach; in the first step a subset of landmarks are selected from the full landmarked shapes. One must put the followings into consideration in doing so. Subset of landmarks must be not too small to avoid discrepancies between the two weight matrix and also not too large to improve the computational efficiency. In subset selection one can collect a set of landmarks randomly by favoring even distribution or shape collectors can suggest certain regions or landmarks.

Interpolation is the second step where one interpolates the weight matrix obtained with small landmarks to the full weight matrix. Now we express the above discussion from mathematical perspective.

We will first select a subset of landmark points from given shapes in order to find an optimal weight matrix using the learning criteria. Now suppose each shape is outlined by n number of landmark points as a result, we need to find $n \times n$ weight matrix. However, it is impractical in the implementations when n is large. We select m number of landmarks where $m < n$ and using this m landmarks we will find an $m \times m$ weight matrix.

Let P and Q be two shapes where they can be represented as $P = [p_1\ p_2 \cdots p_n]$ and $Q = [q_1\ q_2 \ldots q_n]$. Here p_i and q_i is the ith landmark of the shapes. The goal is to find an $n \times n$ optimal weight matrix Σ such that it gives us the minimum of our cost function 5.1.

By the definition of Σ-metric we have

$$< P, Q >_\Sigma = < P\Sigma, Q > = < P\sqrt{\Sigma}, Q\sqrt{\Sigma} > \tag{5.2}$$

Let us define

$$\sqrt{\Sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \ldots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \ldots & \sigma_{2n} \\ \vdots & \ldots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \ldots & \sigma_{nn} \end{pmatrix}$$

In equation (5.2), $< P, Q >_\Sigma = < P\sqrt{\Sigma}, Q\sqrt{\Sigma} >$, we see that a shape P in Σ-metric space is represented as $P\sqrt{\Sigma}$ in euclidean space and

$$P\sqrt{\Sigma} = [p_1\ p_2 \ldots p_n] \begin{pmatrix} \sigma_{11} & \sigma_{12} & \ldots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \ldots & \sigma_{2n} \\ \vdots & \ldots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \ldots & \sigma_{nn} \end{pmatrix} = [\bar{p}_1\ \bar{p}_2 \ldots \bar{p}_m] \tag{5.3}$$

where $\bar{p}_i = \sum\limits_{j=1}^{n} p_j \sigma_{ji}$ and it is the i th landmark of the weighted shape in euclidean space. Each landmark on the weighted shape is a linear combination of the original shape's landmarks. We will use this knowledge in the interpolation step.

Now let us extract m landmark points from all shapes and represent new shape matrices as $\tilde{P} = [\tilde{p}_1 \tilde{p}_2 \ldots \tilde{p}_m]$, and assume we find an optimal $m \times m$ $\tilde{\Sigma}$ matrix that minimizes the cost function. Similar to (5.3) we have

$$\tilde{P}\sqrt{\tilde{\Sigma}} = [\tilde{p}_1\ \tilde{p}_2 \ldots \tilde{p}_m] \begin{pmatrix} \tilde{\sigma}_{11} & \tilde{\sigma}_{12} & \ldots & \tilde{\sigma}_{1m} \\ \tilde{\sigma}_{21} & \tilde{\sigma}_{22} & \ldots & \tilde{\sigma}_{2m} \\ \vdots & \ldots & \ddots & \vdots \\ \tilde{\sigma}_{m1} & \tilde{\sigma}_{m2} & \ldots & \tilde{\sigma}_{mm} \end{pmatrix} = [\bar{p}_1\ \bar{p}_2 \ldots \bar{p}_m]$$

this time $\bar{p}_i = \sum_{j=1}^{m} \tilde{p}_j \tilde{\sigma}_{ji}$

The interpolation step occurs at this point. With the full shape P and $n \times n$ weight matrix we get

$$\bar{p}_i = \sum_{j=1}^{n} p_j \sigma_{ji} = [p_1\ p_2 \ldots p_n] \begin{bmatrix} \sigma_{11} \\ \sigma_{21} \\ \vdots \\ \sigma_{n1} \end{bmatrix}$$

Now we have a newly generated shape of P, $\tilde{P} = [\tilde{p}_1 \tilde{p}_2 \ldots \tilde{p}_m]$ and a learned $m \times m$ weight matrix, $\tilde{\Sigma}$.

Note that $\tilde{p}_i \in \tilde{P}$ is not necessarily equal to $p_i \in P$ and the indices do not correspond to same point in two shapes. We are rather interested in the location of $\tilde{p}_i$s. Now suppose we write m dimensional $\tilde{P}$ in n dimensional space, $m < n$, we fill the missing values with zeros and resulting $\tilde{P}$ will be

$$\tilde{P} = [\tilde{p}_1\ 0\ 0\ \tilde{p}_2\ 0\ 0\ 0\ \tilde{p}_3\ 0\ \ldots 0\ 0\ \tilde{p}_m]$$

where $\tilde{P}$ has m elements. Here zeros are the elements in P but we did not include them into $\tilde{P}$ so they can be considered as empty values in $\tilde{P}$.

Similarly, the full $n \times n$ weight matrix Σ and $m \times m$ $\tilde{\Sigma}$ can be compared in the same sense.

For simplicity let us compare the first columns of two matrices.

$$\Sigma_1 = \begin{bmatrix} \sigma_{11} \\ \sigma_{21} \\ \sigma_{31} \\ \sigma_{41} \\ \vdots \\ \sigma_{k1} \\ \sigma_{(k+1)1} \\ \vdots \\ \sigma_{n1} \end{bmatrix}, \tilde{\Sigma}_1 = \begin{bmatrix} \tilde{\sigma}_{11} \\ 0 \\ 0 \\ \vdots \\ \tilde{\sigma}_{21} \\ \vdots \\ \tilde{\sigma}_{31} \\ 0 \\ 0 \\ \tilde{\sigma}_{m1} \end{bmatrix}$$

Again, σ_{ij} and $\tilde{\sigma}_{ij}$ values in two matrices do not necessarily have to be equal and the number of non-empty elements in the column of $\tilde{\Sigma}$ is m. Our goal is to find the empty values(zeros) of the weight matrix $\tilde{\Sigma}$ using interpolation techniques so that we can get a good approximation of the original weight matrix $\tilde{\Sigma}$.

Originally we had shape matrix P with all the landmark points and a learned weight matrix Σ and as a result of equation (5.3) we had $\bar{p}_i = \sum_{j=1}^{m} p_j \sigma_{ji}$. Using m landmarks of P we generated another $m \times m$ weight matrix $\tilde{\Sigma}$. We need to interpolate empty values of $\tilde{\Sigma}$ to get the full weight matrix.

We have the information for all landmarks of the original shape $P = [p_1 \; p_2 \; \cdots \; p_n]$ yet we still miss some of the corresponding σ_{ij} values. Thus, using landmarks of original shapes, p_i, we will interpolate $\tilde{\sigma}_{ij}$ of $\tilde{\Sigma}$ to generate entries of $n \times n$ Σ matrix, i.e. σ_{ij}. So we will use an interpolation function to find unknown values of σ_{ij}.

In order for us to understand how this approach works, let us give a simple yet intiutive example. Suppose we have shapes with 4 landmark points, we can represent each shape as $P = [p_1 \; p_2 \; p_3 \; p_4]$ and using the full set of landmarks we can generate a 4×4 full weight matrix Σ. Although shapes with 4 landmarks do not require any interpolation, for the sake of illustration we will apply landmark sparcification. Suppose we select a subset of landmarks and keep only two landmarks p_1 and p_3. Using new shape $\tilde{P} = [\tilde{p_1} \; \tilde{p_2}]$ where $\tilde{p_1} = p_1$ and $\tilde{p_2} = p_3$ the weight matrix we will generate using $\tilde{P}$ will be $\tilde{\Sigma}$ which is only 2×2 and can be described as $\tilde{\Sigma} = \begin{vmatrix} \tilde{\sigma}_{11} & \tilde{\sigma}_{12} \\ \tilde{\sigma}_{21} & \tilde{\sigma}_{22} \end{vmatrix}$

If we, hypothetically, map 2×2 matrix $\tilde{\Sigma}$ to 4×4 Σ it will appear as $\tilde{\Sigma} = \begin{vmatrix} \tilde{\sigma}_{11} & 0 & \tilde{\sigma}_{12} & 0 \\ 0 & 0 & 0 & 0 \\ \tilde{\sigma}_{21} & 0 & \tilde{\sigma}_{22} & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$

Next we set $\tilde{\Sigma} = \Sigma$ so that full weight matrix with missing values will be apt to the interpolation.

$$\tilde{\Sigma} = \begin{vmatrix} \tilde{\sigma}_{11} & 0 & \tilde{\sigma}_{12} & 0 \\ 0 & 0 & 0 & 0 \\ \tilde{\sigma}_{21} & 0 & \tilde{\sigma}_{22} & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix} = \begin{vmatrix} \sigma_{11} & 0 & \sigma_{13} & 0 \\ 0 & 0 & 0 & 0 \\ \sigma_{31} & 0 & \sigma_{33} & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix} = \Sigma \tag{5.4}$$

Our interpolation approach uses landmarks of the original shapes $P = [p_1\ p_2 \cdots p_n]$ as the input values of the interpolation function such that $f : P \to \Sigma_i$ where Σ_i is the ith column of the Σ matrix. This way we can use the interpolation function f as $f(p_i) = \sigma_{ji}$, $i, j = 1, 2, 3, 4$. Following the same procedure for all i and j values we can now fill the full weight matrix Σ.

Keeping the same approach of this simple example, we can interpolate any given weight matrix obtained with desired number of landmarks. The next and big question is : what would be a good interpolation method/function?

One of the most popular interpolation technique used in the study of shapes is *thin plate splines* method, [?], [?], [?], [?], and [?]. Although usage of the method may differ in each application, and in our approach too, the idea remains the same.

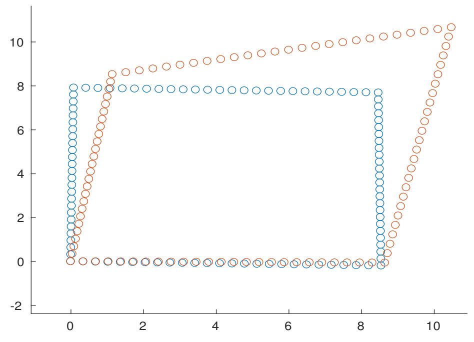
5.2 Explanatory Example

Shapes in our studies have only 12 and 54 landmarks. For the shapes with 12 landmarks, landmark sparcification would be unnecessary. Shapes with 54 landmarks may be a good fit for interpolation, however we first want to test the feasibility of our approach on some randomly shapes. Our randomly generated landmarks originally consists of 100 landmarks on 2 dimensional shape space.

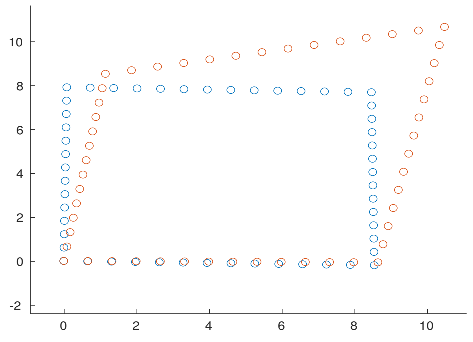
We generated 2 shape families with 100 landmarks and 100 shapes from each family. We then selected 52 landmarks out of 100. Shapes with 100 landmarks and 52 landmarks are shown on Fig. 5.1

We first placed 4 landmarks on the corners. For both groups we set the first landmark at (0,0), while the second and forth landmarks of both groups are spread around the mean (0,8) and (8,0) with 0.5 standard deviation from normal distribution. As it is seen from the figure the main difference between the two groups is the spread of the third landmark, right upper corner. We choose the third landmark to be spread around mean (8,8) in the first group and (10, 10) for the second group with the same standard deviation(0.5) for both groups.

This way we generated 4 landmarks for shapes in each group. We then generated landmarks in between each of these landmarks so that there is an equal distance between each landmark. As we generated more and more points this distance is shrinked. Fig. (5.1) shows the mean landmark points and other shapes are generated from normal distribution with given means and 0.5 standard deviation.



(a) Full Shape: 100 landmarks



(b) Sparcified Shape: 52 landmarks

Figure 5.1: Landmark Sparcification from 100 to 52

We then minimized our objective function with full and sparcified shapes to compare the results and check the performance of our sparsification method. We minimized 5.1 and obtained weight matrices 52×52 $\tilde{\Sigma}$ matrix and 100×100 Σ matrix. Next, we interpolated $\tilde{\Sigma}$ to Σ. Here we present the classification results with the full matrix, interpolated matrix and identity matrix. Recall that when the weight matrix is the identity matrix we are simply performing a classical shape analysis.

We ran the weight matrix interpolation experiments several times selecting different subset of landmarks. However, we implemented the algorithm only once for the identity matrix model and weight matrix model using all 100 landmark points.

Our objective is to investigate the potential of the interpolation method using smaller number of landmarks. Therefore, we analyzed the performance of the model with 20, 40, 52, 60, 76 and 92 different subset of landmark points. Afterwards we interpolated the resulting $m \times m$ matrices, m being the number of landmarks, to 100×100 weight matrix. Finally we executed nearest neighboor classification rule to the test samples, 100 shapes from each family. Table 5.1 summarizes the findings.

Table 5.1: Classification Results(%) of 3 Weight Matrices

Number of Landmarks	$\Sigma = I_n$	Full Weight Matrix (Σ)	Interpolated Weight Matrix($\tilde{\Sigma}$)
20	93.5	97.0	91.5
40	93.5	97.0	95.0
52	93.5	97.0	96.5
60	93.5	97.0	93.0
76	93.5	97.0	97.5
92	93.5	97.0	96.5

Note that the results for the identity weight matrix and full weight matrix are same for all experiments and this is due to the fact that we only changed landmark points for the interpolation purposes. It only effects the results of interpolated weight matrix, thus, the results in the last column are different.

From the experiments we observed that when we select small number of landmarks, i.e. 20, we get poor results compared to the identity matrix yet we get better results as we increase the number of landmarks. We see an exception to this improvement for 60 landmarks. This may be as a consequence of the interpolation process. We also notice that interpolated weight matrix outperforms both identity and full weight matrices in terms of classification results with the selected 76 landmarks. Another conclusion is that the interpolated weight matrix mostly exceeds the results of classical shape model in which there is no weigth matrix, that is the weight matrix is the identity matrix.

A great advantage of the interpolated weight matrix method is its effectiveness from the computational perspective. It produces better results than the classical shape model in most instances yet takes less time compared to the full matrix model.

We should also address an important factor in this method which is the number of landmarks to be selected. Answer to this question lies in the priority; if one's goal is rather to obtain good enough results than perfect results one can hence proceed with the smallest landmark points that exceeds the performance of the classical model. If the goal is to generate as good outcome as the full weight matrix method, then the number of landmark points needs to be tuned to achieve the best results.

CHAPTER 6

APPLICATIONS

We implement our proposed learning method which is summarized in algorithm **??** to three different shape problems in this chaper. Our goal in each application is to learn a weight matrix Σ that separates different classes in the optimal form. We start with a preliminary experiment in which we first introduce the data and give some insights into the idea of shape metric learning problem. We then advance to a real life application in which we employ our learning method in 3-D mice skull data. In the final application we examine fruit fly species and using the shape of their wings we enhance the differentiation of different species.

6.1 Preliminary Experiment

In this experiment we use randomly generated shapes introduced in the previous chapter. We first want to implement our proposed shape metric learning algorithm on synthetic shapes to understand the power of our method. We also wish to shed some light into the capabilities of our method in the classification of two groups differing from each other at certain points.

Let us start by introducing the data. We generated 100 training shapes for each of two different shape groups- red and blue- along with 100 test shapes from each family. Figure 6.1 shows randomly generated shapes for both groups. We set the first landmark for both groups at (0,0), while the second and forth landmarks of both groups are spread around the mean (0,8) and (8,0) with 0.5 standard deviation from normal distribution. As it is seen from the figure the only difference between two groups is the spread of the third landmark. We choose the third landmark to be spread around mean (8,8) in the first group and (10, 10) for the second group with the same standard deviation(0.5) for both groups.

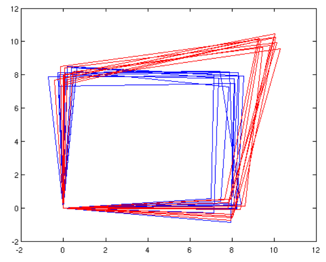


Figure 6.1: Synthetic Shapes

Applying our method to synthetic data we obtain the following weight matrix

$$\Sigma = \begin{bmatrix} 1.20 & -0.91 & 0.56 & 0.24 \\ -0.91 & 2.41 & -1.24 & 0.78 \\ 0.56 & -1.24 & 1.65 & -0.89 \\ 0.24 & 0.78 & -0.89 & 1.38 \end{bmatrix}$$

with eigenvalues $\lambda_1 = 4.1480$, $\lambda_2 = 1.6057$, $\lambda_3 = 0.6834$, and $\lambda_4 = 0.2197$. The eigenvectors of the weight matrix are the colums of the following matrix

$$U = \begin{bmatrix} -0.29 & -0.65 & -0.31 & -0.61 \\ 0.70 & 0.21 & -0.63 & -0.23 \\ -0.54 & 0.18 & -0.69 & 0.42 \\ 0.35 & -0.69 & -0.07 & 0.61 \end{bmatrix}$$

Recall that one way to interpret the weight matrix Σ is accomplished by decomposing Σ such that $\Sigma = U\Lambda U^T$ where U is an $n \times n$ orthogonal matrix, that is $U \in O(n)$ and Λ is an $n \times n$ diagonal matrix where λ_{ii} is the i-th largest eigenvalue of Σ that corresponds to the i-th eigenvector, i.e. i-th column of U.

Let P and Q be two preshapes, then the following transformation can be done

$$< P, Q >_\Sigma = < P\Sigma, Q > = < PU\Lambda U^T, Q > = < PU\Lambda, QU > = < PU, QU >_\Lambda = < \tilde{P}, \tilde{Q} >_\Lambda \tag{6.1}$$

where $\tilde{P} = PU$ and $\tilde{Q} = QU$ represent orthogonally transformed original shapes.

Eq 6.1 simply means that shapes in original Σ-metric space can be orthogonally rotated and resulting shapes are weighted by the elements of diagonal matrix Λ. Fig 6.2 illustrates such rotation. We see that landmarks in the rotated shapes are distributed such that the third landmark still has the largest gap in between the two groups however it is penalized by the amount of third eigenvalue $\lambda_3 = 0.6834$ which is very small compared to other penalizations. On the other hand, we see the second largest gap in the first landmark yet it gets the largest penalization $\lambda_1 = 4.1480$ during the shape alignment process.

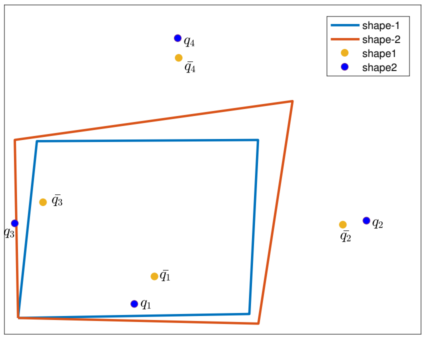
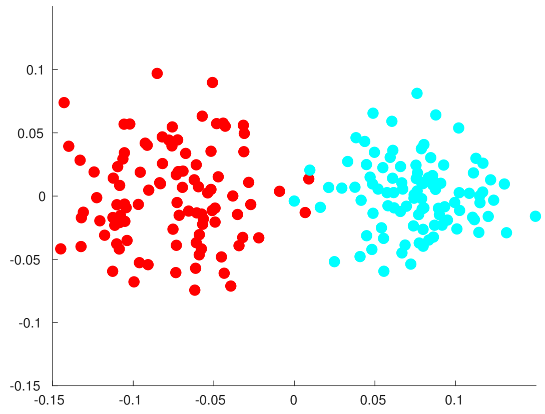
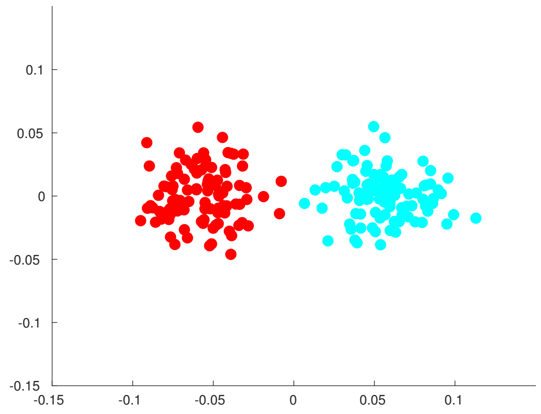


Figure 6.2: Shapes before and after orthogonal rotation



(a) Classical shape model



(b) Weight matrix model

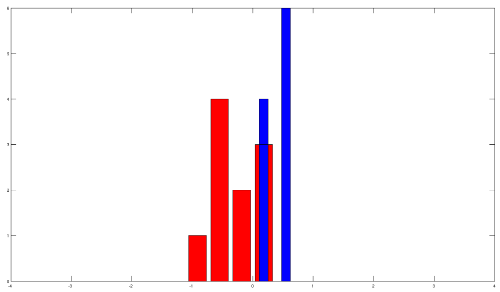
Figure 6.3: Class separation with weight matrix on training data

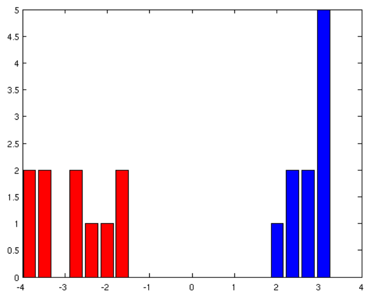
Figure (6.3a) is the multi-dimensional scaling(MDS) plot and it illustrates the separation of two groups with classical procrustes method while figure (6.3b) shows the detachment of two groups with the weight matrix model on. We notice that weight matrix clusters groups so that within class scatter is reduced while keeping the groups away. This observation matches with the theoretical framework along with the motivation and intitution behind the weight matrix model.

After learning the weight matrix from the training data, we applied the nearest neighbor classification rule to 500 test shapes. We used two different distances for the classification, the classical procrustes distance-Euclidean metric and the Σ-metric distance. The classification accuracy for 500 test shapes from both classes is 94.2% in classical shape model while it is 98.3% with generalized shape model.

From these results we can conclude that the weight matrix model not only improves the classification accuracy but it also sheds light onto the different regions between groups of shapes.

We further analyzed the efficiency of weight matrix model with multivariate analysis methods. We first projected shapes from shape space to tangent space followed by implementing principal component analysis(PCA). We then selected first 6 principal components which counted for the 96.23% of variation and reduced the dimension of our shapes into 6. We performed a discriminant analysis in this 6 dimensional space and obtained a projection of the class separation for both Euclidian metric and Σ-metric. Fig 6.4 demonstrates the improvement in shape separation.


(a) Separation when no weight matrix applied



(b) With weight matrix

Figure 6.4: Improvement of class separation with weight matrix

Figure (6.4a) illustrates the separation of two groups with classical procrustes method while figure (6.4b) shows the detachment of two groups with the weight matrix model after projecting data onto the discriminant direction.

Method	Classical Procrustes Distance	Weight Matrix Model
MDS	94.2	98.3
PCA + LDA	93.6	99.1

Table 6.1: Classification of test samples

The following figure shows how our algorithm locates the minimum of cost function for the data in figure(6.4).

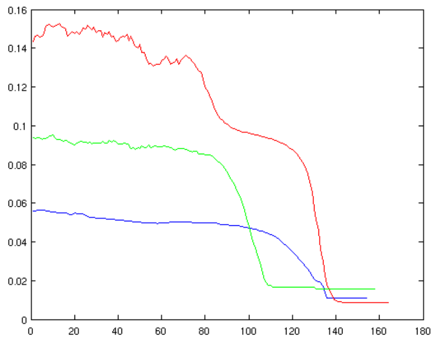


Figure 6.5: Convergence of the minimum of cost function for different initial solutions

6.2 3D Mice Skull Data

We carried out two different experiments with mice skull data. We first worked on 8 parental strains and applied our method to learn an optimal weight matrix to enhance the categorization of parents. We then extended our study to the offsprings of the parental strains.

Let us first introduce the data and then move to the steps we followed in preliminary experiment. The data in the first experiment consists of 8 parent strains as AA,BB,CC,DD,EE,FF,GG,HH and the number of shapes at each group is given at table 6.2.

Group	AA	BB	CC	DD	EE	FF	GG	HH
# of Samples	19	19	18	14	19	19	18	20

Table 6.2: Number of samples at each group

In total we have 146 samples and each sample has 54 landmark points in (x,y,z) coordinates. For instance, the data of first shape in group AA, say, AA_1 is represented as

Coordinate	landmark1	landmark2	...	landmark54
x	9.03	3.955	...	9.31
y	19.95	9.87	...	11.725
z	23.905	13.79	...	17.78

Table 6.3: The landmark points/coordinates of AA_1

Since the number of shapes from each group is very small, 20 at most, we avoided splitting the data into training and the test shapes. Instead, we validated our results using leave-one-out cross validation. Our results in the following experiments are based on the leave-one-out cross validation method.

Figure (6.6) shows the spread of 8 groups with $\Sigma = \mathbb{I}_{54}$, that is, no weight is given to landmark points. Essentially, we are using classical procrustes metric.

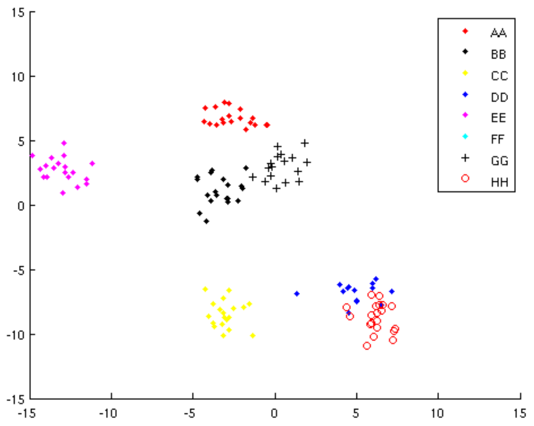


Figure 6.6: Separation of mice data with $\Sigma = \mathbb{I}_{54}$

It seems DD is considerably mixed with HH and as a result many shapes from these two groups will be misclassified. AA, BB and GG seem to be very close to each other and some shapes from these classes would also be misclassified. Applying k-nearest neighbor with k=2 we get 89.04 classification rate when there is no weight given to landmark points.

Intuitively, we would expect from the optimal weight matrix to separate DD from HH and also apart AA,BB and GG from each other. Figure (6.7) shows the spread of 8 groups with the weight matrix Σ obtained using the simulated annealing algorithm.

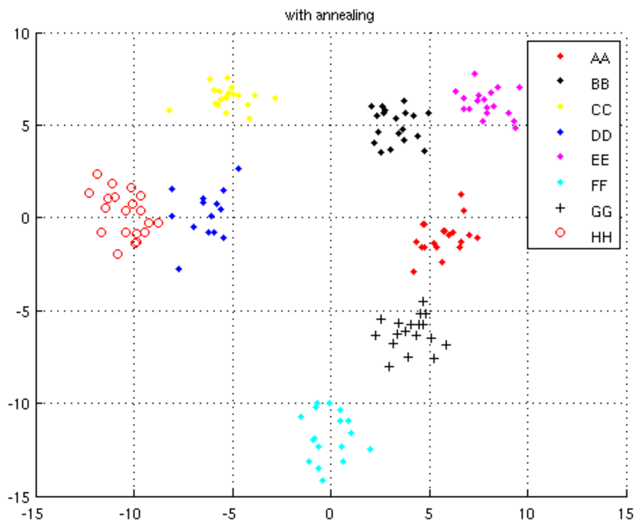


Figure 6.7: Separation of mice data with full weight matrix Σ

In figure (6.7) we see that DD and HH are well separated with one sample exception of DD, which is very close to HH, and AA,BB and GG are also further away from one another. This improvement matches with our intuition as well. With k=2, k-nearest neighbor rule gives us the classification rate as 98.63 in this experiment. The results once again prove that the weight matrix model obtained with the simulated annealing improves the classification of several shape families. Figure (6.8) shows the cost function's convergence over time.

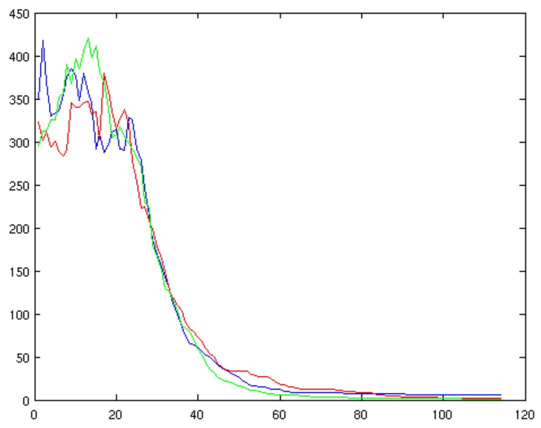


Figure 6.8: Cost function's convergence for different initial solutions

Since all groups are well separated with a single experiment there is no need for further steps. If some of the groups were still mixed at first step, we would then follow a tree-like model in which we would first detach the groups well separated from the others and in the next step we would detach the rest of the groups following the same procedure. We would continue to this until all classes are well separated.

Second experiment in which we include offsprings of parental strains as well will adapt this approach. We have 8 parental strains and their crossproduction yields AB, AC, AD, ... , HH offsprings. In total we have 62 shape families, we are missing EF and EG. We should note that AB and BA are not in the same offspring family since in one, male is from parent A and female is from parent B group and it is the opposite in the other. Considering 62 offsprings, it is very difficult to differentiate them in one experiment compared to the first experiments where we had only 8 parental strains. Thus, we will follow a hierarchical approach.

In hierarchical categorization, we first implement classical shape analysis and using k-means clustering we re-group shape families so that we can put them in k clusters, k is usually chosen 2 or 3 throughout our experiments. Once we cluster all families into 2 or 3 groups we check the separation of the clusters. If they are already well separated with the classical procrustes method we proceed with the current clusters and dive into each cluster afterwards. Note that each cluster has several shape families so we follow the same procedure in each cluster and repeat it until there is no further cluster remains with more than one family in it.

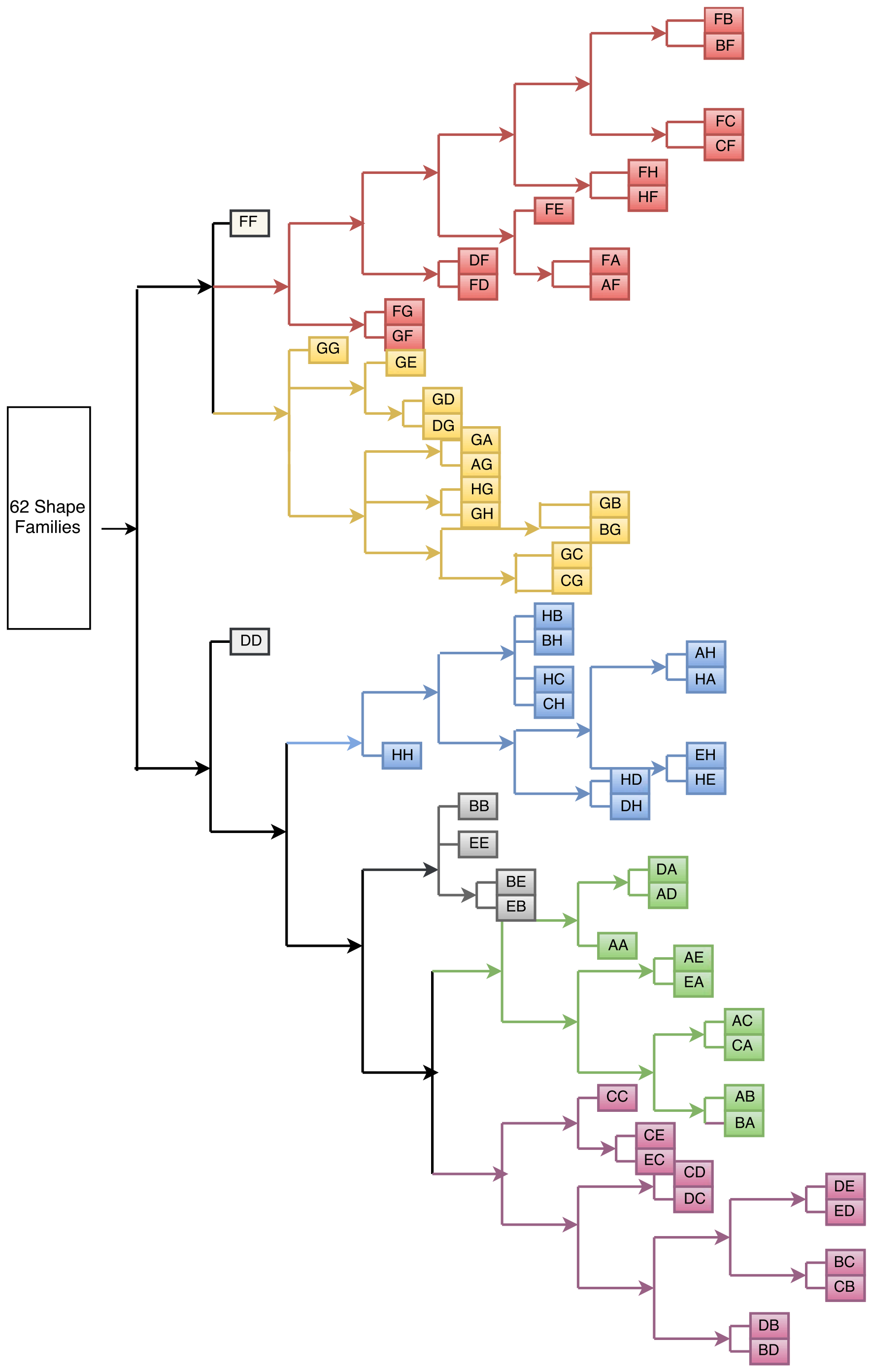


Figure 6.9: Top-down hierarchical organization of 62 offsprings

Following this procedure we obtain a top-down clustering as shown in Fig. 6.9. We first notice that offsprings of F and G parents(26 groups in total) isolated from the remaining offsprings. We also see that FF is well separated from the other offsprings in that 26 groups. Besides, F and G families are well divided as well. This is an indication that some traits of F and G offsprings are significantly different than the other 36 groups.

Recall that the spread of offsprings are formed on discriminant analysis space that is generated using several principal components. One can obtain the traits/landmarks that differ two larger groups by back-tracking the loadings of discriminant line and the loadings of PCA. Although this approach theoretically sounds solid, it is extremely inefficient in the implementations due to the effect of linear combinations of loadings. If the loadings were dominated by only few coefficients we would apply back-tracking method in which we would determine the most important principal components that separate two groups well in discriminant space.

After determining the most discriminatory principal components we then would focus on the coefficients of those particular components. And if those components were also influenced by only few landmarks we would conclude that those most influential landmarks play an important role in the classification process. We should note that in our clustering and classification processes we did not encounter such coefficients. Therefore, here we will only address the separation of different offsprings without diving into what differs them.

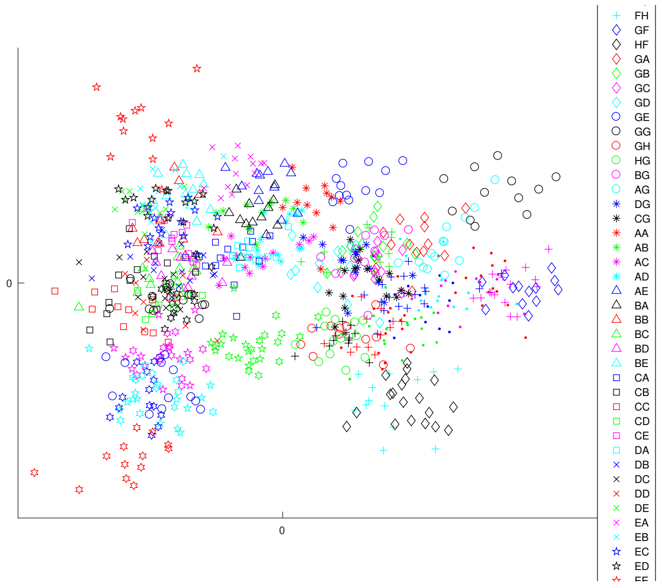


Figure 6.10: Discrimination of 62 Offsprings

Fig. 6.10 shows the very first clustering framework. We notice that all F and G families are on the positive side of the x-axis and the remaining ofsprings fall onto the left of 0. Applying k-means clustering with k=2, 62 shape families are divided to 26(F and G offsprings) and 36(remaining offsprings). We then classify these two groups and obtain a classification rate. If the two larger classes are well segregated, say classification rate is more than 95%-98% with the classical shape model we do not apply any learning algorithm since the classical procrustes method already succeeds. Fig. 6.11 demonstrates such success in discrimination resulting 99.83 classification accuracy on leave one out validation. we proceed to the lower branches of tree.

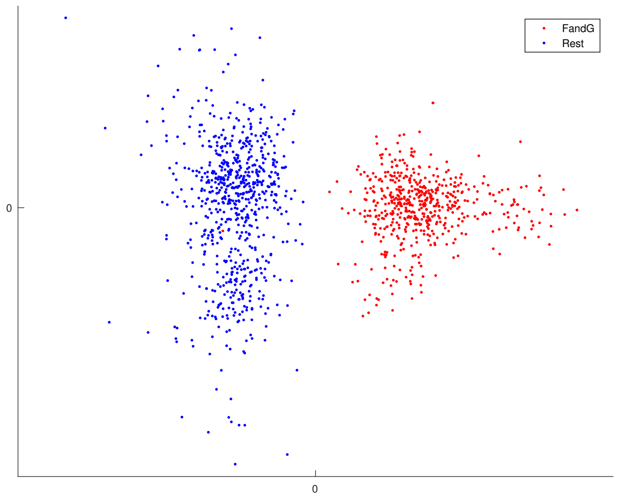
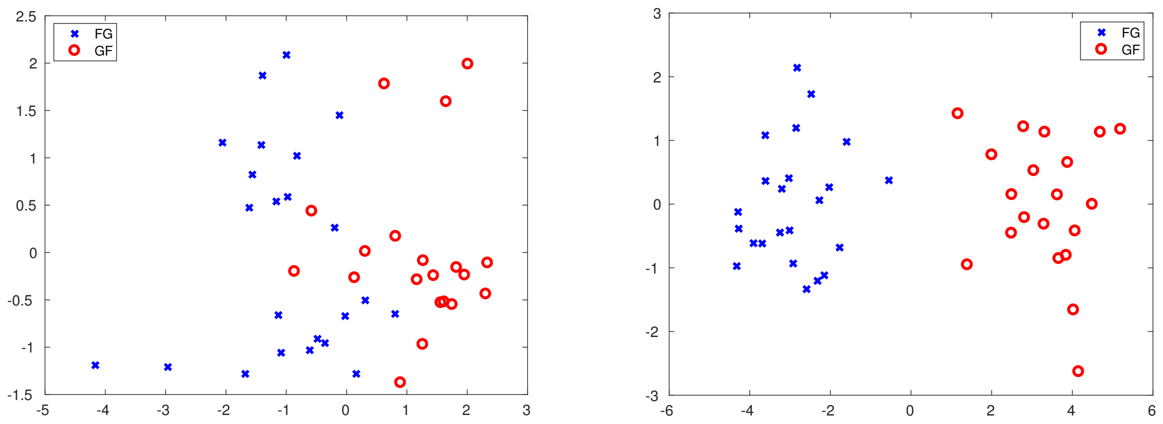


Figure 6.11: First separation: F&G vs other offsprings with classical shape model

As in the case of first branch, most of the times there is no need for learning shape metric in the upper nodes. We applied weight matrix model only to the groups that were poorly separated with the classical shape model. For instance, as seen in Fig. 6.12b, two groups of shapes (FG and GF) are detached unsatisfactorily, therefore we apply our proposed learning model to improve the separation of the two groups. As a result, we see an improvement in discrimination and the classification rate immediately increases from 81.39% to 95.34% for the same groups.

One of the important findings in our study is the distribution of the last branches of hierarchical tree. Genetically and phonetypically we would expect the offsprings from the same parental strains to be closest to each other. Our top-down clustering approach leads us to same conclusion. For instance, FB and BF are the most similar to each other among all 62 families. Note that FB is the offspring when male F and female B is coupled and similarly BF is originated from male B and female F. We see the same pattern in all offsprings from the same parental strains(see Fig. 6.9).

In classification process, as anticipated, most challenging tree branches were the offsprings from the same parental strains. Fig.6.12a illustrates such challenge. We see that FG and GF are poorly separated with the classical shape model resulting in 81.39% classification accuracy with nearest neighbor rule. It is an ideal stage to apply our learning algorithm in which we find an optimal shape metric using simulated annealing method to enhance the detachment of the two groups. We notice such improvement in Fig. 6.12b and consequently we obtain 95.34% classification accuracy.



(a) Classical Procrustes method

(b) Generalized Procrustes method (SA)

Figure 6.12: Distinguishing *FG* and *GF* with classical shape model and generalized shape model using SA

We applied our learning algorithm to almost every last branch of the hierarchical tree and improved the separation of the most similar groups accordingly. Since the partition in upper nodes, such as the node where F and G families are aparted from the rest (see Fig. 6.11), was clear enough with the classical shape model we did not apply weight matrix model in those nodes. We mostly applied our method to the last branches and occasionally to upper branches as well.

Here we present the final results in our study. We tested the efficiency of our hierarchical approach coupled with learned weight matrix using leave one out cross validation. We picked every shape one by one and treated them as an 'unknown' shape though we kept the family information for testing purposes. We then placed each shape to the top node in which we had two larger groups (26-F&G- and 36 families). We subsequently assigned the 'unknown' shape to the group where it was closest to. After the initial assignment the unknown shape is now assigned to one of the 26 or 36 groups. Following the same procedure in the lower nodes we placed the 'unknown' shape to one of the families. We repeated the same scheme for each shape and now every shape has a new assigned family. As a final step we compared the original labels of the shapes with their assigned labels. The accuracy corresponds to the percentage of match of their initial and final labels.

We carried out the same experiment with two different shape metrics. In the first implementation, we used classical shape metric at every node in the classification process. In the second experiment, we applied the shape metric that we learned at certain nodes. Recall that we only learned weight matrix where the detachment was not good enough with the classical approach.

Finally, we obtained overall classification accuracy for classical model to be 79.34% and for generalized shape model the classification accuracy is improved to the 91.83%. Main improvement in this study is the detachment of the most similar offsprings where now we are able increase the separation of those species using weight matrix model in which we use simulated annealing method.

6.3 Classifying Fruit Flies from Wing Shapes

In this section, we present the results of the morphological analysis of fruit fly wings using the weight matrix approach with simulated annealing method discussed in chapter 4. Our dataset contains 24 species of fruit flies where each fly has 12 landmark points on its wing (see Fig. (??)). The objective of the morphological analysis is to determine the species a fruit fly belongs to.

The number of shapes in each species ranges from 181 to 268. Most of the species have more than 200 shapes that gives us the flexibility of splitting data into training and test sets. We used 100 shapes from each family as a training data and remainings as test data. Unlike mice skull analysis, our results will be hinged on the test shapes.

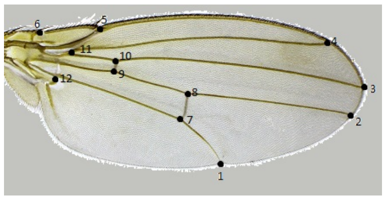


Figure 6.13: Wing of a fruit fly with 12 landmark points on it

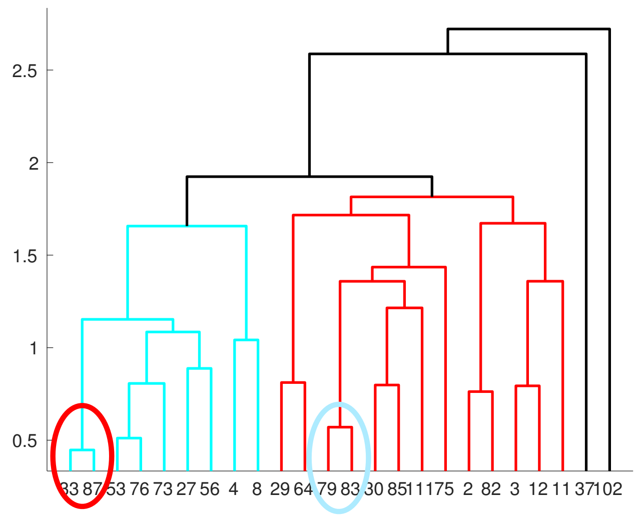
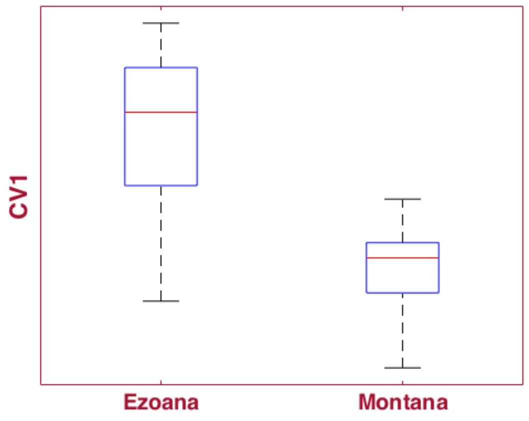
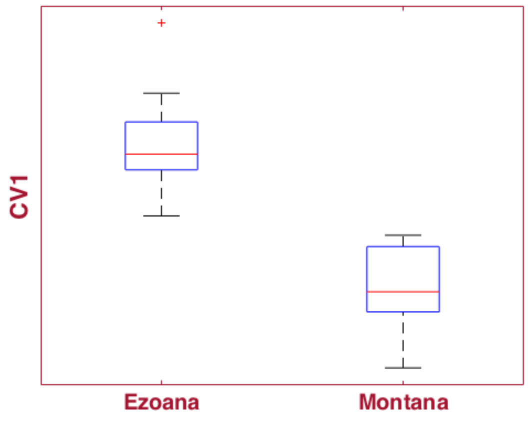


Figure 6.14: Dendrogram plot of 24 fly species

We first determined the similarity of species using the single linkage hierarchical clustering method. The method measures similarity of objects by calculating the pairwise distances between the mean shape of each species. It then builds a dendrogram tree; see Fig. ??. The height of the U-shaped lines in the dendrogram plot represents the distance between the two data points that are connected. For example, *Ezoana* and *Montana* (left-red circle) are connected by a U-shape that has the shortest height in the plot, indicating these two species are the closest to each other, i.e., the most similar species, among all species in the plot. The other two species that are similar are *Erecta* and *Occidentalis* (right-cyan circle).



(a) Classical Procrustes method



(b) Generalized Procrustes method (SA)

Figure 6.15: Distinguishing *Ezoana* and *Montana* with classical Procrustes model and generalized Procrustes model using simulated annealing

Our goal is to correctly distinguish between species that are the most similar; *Ezoana* & *Montana* and *Erecta* & *Occidentalis*. We first applied the classical Procrustes method to distinguish samples from *Ezoana* and *Montana*. Figure (6.15a) is a box-and-whisker plot in the canonical variate direction (CV1). CV1 is the line which is obtained by Fisher's Discriminant analysis. It projects the samples in high dimensional space to a lower dimension in which the samples of different groups are put far away while the samples in the same group are put as close as possible.

There is a clear overlap between the two box-and-whisker plots, which indicates several samples will be misclassified in the classification process. Next we applied the generalized Procrustes model with weight matrix model coupled with simulated annealing. Figure (6.15b) gives the box-and-whisker plot for the generalized method. Now we can see a perfect separation of the two plots, indicating an improved classification accuracy.

We further investigated the effect of the initial guess $\omega \in \Omega$ (step 1 of the algorithm) to the performance of the simulated annealing algorithm. Specifically, we wanted to check if using a low-discrepancy sequence (and thus of low-dispersion) of quasi-Monte Carlo methods, instead of a

Species	Samples	Classical Procrustes Method(%)	Generalized Procrustes Method (SA)(%)
Ezoana	154	62.3	74.6
Montana	162		
Erecta	148	71.2	82.4
Occidentalis	171		

Table 6.4: Classification accuracy of the classical Procrustes method and generalized Procrustes method with simulated annealing

pseudorandom sequence, might improve the convergence of the algorithm. To this end, we generated 1000 initial points from Ω using (i) Mersenne twister, a well known pseudorandom number generator, and (ii) Sobol' sequence, a well known low-discrepancy sequence. In the case of the Sobol' sequence, 1000 points refer to the first 1000 vectors of a 78-dimensional Sobol' sequence (recall that the we have 12 landmarks and the dimension of the symmetric matrix A is $n(n+1)/2$). We then started the simulated annealing algorithm from each initial point. Fig. 2.1b plots the average cost function value of 1000 different simulated annealing algorithms for each iteration, when their initial points are chosen from Mersenne twister and Sobol' sequences. The two plots are visually indistinguishable in the scale of the figure. From this and other numerical results not reported, we have concluded that simulated annealing offers fast convergence to the global minimum and the "quality" of the initial guess does not make much impact.

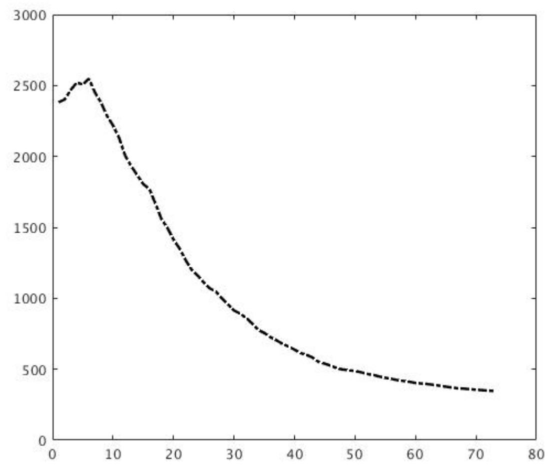


Figure 6.16: Cost function value against simulated annealing iterations

CHAPTER 7

CONCLUSIONS AND DISCUSSION

In this dissertation we applied a Monte Carlo optimization method, the simulated annealing algorithm, to the weight matrix model for which we looked for the optimal weight matrix that separates classes in the optimal form. The idea was to find a weight matrix which distinguishes groups of shapes by attributing weights to different landmark points and their linear combinations as well.

Previous approaches to the weight matrix model were deterministic and as a result computationally expensive. They used to take considerable amount of time in finding an optimal weight matrix. Our approach is novel since it uses a stochastic method in which the optimal weight matrix is estimated. Our results in applications and preliminary experiments show how suitable our approach is as it can separate different classes in the best form rapidly with less computational efforts.

The weight matrix obtained using our algorithm improved the classification rates in each application discussed in our studies including preliminary experiment, 3D mice skull data and fruit flies species. The results are promising for higher dimensional and more complex shape classification problems as well.

We have also introduced the idea of landmark sparcification and produced even better results with using less number of landmarks. This approach is significant in several applications. First, it is very useful with shapes having many landmarks as the number of landmarks decreases the computational efficiency. It is also advantageous for the studies of shapes in which landmarking is a difficult or redundant task. It can be applied to shapes obtained as 3D images and turned into meshes. Since each shape is now represented by a mesh now we can consider the vertices of meshes as landmark points and that helps researchers to avoid landmark correspondence process. Landmark correspondence is one of the most challenging procedure in landmarking processes as it requires researchers to locate same numbered landmarks to the same position at each object. This task can be overwhelming and the data obtained with this methodology usually fall short in the sense that it is difficult to generate as many shapes as desired. It is still possible to apply weight matrix model to problems with such objects if we reduce the number of landmarks. Landmark sparcification process is another main contribution of this dissertation.

Additionally, we have created a hierarchical similarity and dissimilarity measure for the offsprings of 8 parental mice strains. We have observed that offsprings from the same parental strains are closest to each other when we use the weight matrix model as the distance measure. This observation is very intuitive as one would expect to have genetically and phenotypically similar offsprings from the same parents. We have arrived this result from the point of statistical analysis of shapes where we treated each shape as a geometrical information of object.

We have tested the performance of our method on Fruit Fly species and analyzed them using their wing shapes. In this study the goal was to develop a taxonomical tree which represents the most similar species of fruit flies as well as a similarity map for the species. We have created a fruit fly taxonomy based on the shape of their wings. We have also aparted the most similar species and as a result increased the classification rate of those species.

Most common stochastic optimization techniques from different class of methods were also discussed in the dissertation. We analyzed the efficiency of each method on some benchmarking test functions. We have concluded that although each method has its own strength and weaknesses simulated annealing seems to be a good choice of method for our problem at hand. And our results have proven the capabilities of simulated annealing method.