I dedicate this to my parents.
ACKNOWLEDGMENTS

First and foremost I offer my sincerest gratitude to my supervisor, Professor Baba C. Vemuri, who has supported my PhD study all the way. He encouraged me with his patience, his creativity and his first class leading knowledge.

I also thanks my other committee members including Professor Anand Rangarajan, Professor Jeffery Ho, Professor Arunava Banerjee and Professor William Hager. Professor Anand Rangarajan with his great knowledge and explanation, taught me many machine learning techniques which are very important building blocks for my research. Professor Jeffery Ho taught me a lot of ideas about convex optimization which was fundamental for my study in other topics. Professor Arunava Banerjee gave me a lot of suggestions during the talks which motivated my study a lot. Professor William Hager taught me many numerical methods that gave me the necessary background to solve several optimization problems encountered in this thesis efficiently. Special thanks to Professor Shun-Ichi Amari and Professor Frank Nielsen for their collaboration and their great help for my research. Also, special thanks to Professor Rachid Deriche for his supervision before and during my internship at INRIA. His support encouraged me immensely.

My friendly and knowledgeable colleagues Bing, Fei, Santhosh, Angelos, Adrian, Ajit, Ozlem, Ritwik, O’Neil, Ting, Yuchen, Wenxing, Sile, Guang, Yan, Yuanxiang, Dohyung, Hesam, discussed with me a lot about my research and other courses. I would like to express my sincere thanks to one and all.

Finally I offer my greatest appreciation to my parents for bringing me to the world, feed me, educate me, support me and their constant help in my daily life. I can not expect better parents than them. I offer my appreciation to my grand parents, who always made nice dishes for me and told interesting stories to me when I was a child. I also offer my great thanks to my sister and my brothers for growing up together with me, giving me support and bringing me great happiness.
The research in this dissertation was in part supported by NIH grants EB007082, NS066340 to Professor Baba C. Vemuri. I also received the University of Florida Alumni Fellowship, many travel grants from NSF, the CISE department, the Graduate Student Council at the University of Florida, and the INRIA Internships Program. I gratefully acknowledge the permission granted by IEEE and Springer for me to reuse materials from my prior publications in this dissertation.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>4</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>8</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>9</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>10</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>11</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>11</td>
</tr>
<tr>
<td>1.2 $\ell_1$-norm Center of TBD</td>
<td>12</td>
</tr>
<tr>
<td>1.2.1 Outline</td>
<td>13</td>
</tr>
<tr>
<td>2 TOTAL BREGMAN DIVERGENCE</td>
<td>14</td>
</tr>
<tr>
<td>2.1 Definition of TBD and Examples</td>
<td>14</td>
</tr>
<tr>
<td>2.2 Total Bregman Divergence Centers</td>
<td>17</td>
</tr>
<tr>
<td>2.2.1 $\ell_1$-norm $t$-center</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Properties of $t$-center</td>
<td>19</td>
</tr>
<tr>
<td>2.3.1 $t$-center Uniquely Exists</td>
<td>19</td>
</tr>
<tr>
<td>2.3.2 $t$-center is Statistically Robust to Outliers</td>
<td>23</td>
</tr>
<tr>
<td>2.3.3 Properties of TBD</td>
<td>25</td>
</tr>
<tr>
<td>3 APPLICATION OF TBD TO DTI ANALYSIS</td>
<td>26</td>
</tr>
<tr>
<td>3.1 SPD Tensor Interpolation Applications</td>
<td>26</td>
</tr>
<tr>
<td>3.2 Piecewise Constant DTI Segmentation</td>
<td>29</td>
</tr>
<tr>
<td>3.3 Piecewise Smooth DTI Segmentation</td>
<td>30</td>
</tr>
<tr>
<td>3.4 Experimental Results</td>
<td>31</td>
</tr>
<tr>
<td>3.5 Tensor Interpolation Experiments</td>
<td>31</td>
</tr>
<tr>
<td>3.6 Tensor Field Segmentation Experiments</td>
<td>33</td>
</tr>
<tr>
<td>3.6.1 Segmentation of Synthetic Tensor Fields</td>
<td>34</td>
</tr>
<tr>
<td>3.6.2 Segmentation of DTI Images</td>
<td>34</td>
</tr>
<tr>
<td>3.7 Discussions</td>
<td>38</td>
</tr>
<tr>
<td>4 HIERARCHICAL SHAPE RETRIEVAL USING $t$-CENTER BASED SOFT CLUSTERING</td>
<td>39</td>
</tr>
<tr>
<td>4.1 Literature Review for Shape Retrieval</td>
<td>39</td>
</tr>
<tr>
<td>4.2 Total Bregman Divergence Clustering</td>
<td>41</td>
</tr>
<tr>
<td>4.2.1 Total Bregman Divergence Hard Clustering</td>
<td>41</td>
</tr>
<tr>
<td>4.2.2 Total Bregman Divergence Soft Clustering</td>
<td>43</td>
</tr>
<tr>
<td>4.2.2.1 Synthetic experiments for soft clustering</td>
<td>45</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table | page
--- | ---
2-1 TBD $\delta_f$ corresponding to $f$. $\bar{x} = 1 - x$, $\bar{y} = 1 - y$, $x^t$ is the transpose of $x$. $\Delta^d$ is $d$-simplex. | 17
2-2 TBD $\delta_f$ and the corresponding $t$-center. $\bar{x} = 1 - x$, $\bar{y} = 1 - y$. $c$ is the normalization constant to make it a pdf, $w_i = \frac{1}{\sum_j 1/\sqrt{1+||\nabla f(x_j)||^2}}$ | 22
3-1 The difference between the various means/medians, and the ground truth. | 33
3-2 Time (seconds) spent in finding the mean/median using different divergences. | 34
3-3 Time (seconds) comparison for segmenting the rat spinal cord, corpus callosum and hippocampus using different divergences. | 37
4-1 The clustering results for the 1D, 2D and 5D data sets by applying the Bregman and TBD soft clustering algorithms. | 46
4-2 NMI beween the original clusters and the clusters got from TBD and BD soft clustering algorithms. $\bar{c}$ is the predicted number of clusters. | 47
4-3 Recognition rates for shape retrieval in the MPEG-7 database. | 55
4-4 Recognition rates for shape retrieval from the Brown database. | 56
4-5 Recognition rates for shape retrieval from the Swedish leaf database. | 56
4-6 Retrieval comparison with other methods (CRSP [92], DSR [2], DBF [2], DBI [126], Sil [126], D2 [89]) on PSB. | 57
5-1 Description of the UCI datasets that we use. | 72
5-2 Classification accuracy (mean ± deviation) for different boosting algorithms. | 72
5-3 Classification accuracy of different methods on the OASIS dataset. | 74
5-4 Classification accuracy for different methods on the Epilepsy, CNS, Colon tumor and Leukemia datasets. | 75
6-1 Error in estimated DTI and $S_0$, using different methods, from synthetic DWI with different levels of noise. | 85
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>$d_f(x, y)$ is BD, $\delta_f(x, y)$ is TBD. This shows the $d_f(x, y)$ and $\delta_f(x, y)$ before and after rotating the coordinate system.</td>
</tr>
<tr>
<td>3-1</td>
<td>The isosurfaces of $d_r(P, I) = r$, $d_R(P, I) = r$, $KL_s(P, I) = r$ and $tKL(P, I) = r$ shown from left to right. The three axes are eigenvalues of $P$.</td>
</tr>
<tr>
<td>3-2</td>
<td>From left to right are initialization, intermediate step and final segmentation.</td>
</tr>
<tr>
<td>3-3</td>
<td>Dice coefficient comparison for $tKL$, $KL_s$, $d_R$, $d_M$ and $LE$ segmentation of synthetic tensor field with increasing level (x-axis) of noise.</td>
</tr>
<tr>
<td>3-4</td>
<td>Dice coefficient comparison for $tKL$, $KL_s$, $d_R$, $d_M$ and $LE$ segmentation of synthetic tensor field with increasing percentage (x-axis) of outliers.</td>
</tr>
<tr>
<td>3-5</td>
<td>The segmentation results using $tKL$, $KL_s$, $d_R$, $d_M$ and $LE$.</td>
</tr>
<tr>
<td>3-6</td>
<td>tKL segmentation of a 3D rat corpus callosum.</td>
</tr>
<tr>
<td>4-1</td>
<td>$k$-tree diagram. Every key is a GMM. Each key in the inner nodes is the $t$-center of all keys in its children nodes.</td>
</tr>
<tr>
<td>4-2</td>
<td>Left to right: original shapes; aligned boundaries; GMM with 10 components.</td>
</tr>
<tr>
<td>4-3</td>
<td>Comparison of clustering accuracy of $tSL$, $\chi^2$ and $SL$, versus average number of shapes per cluster.</td>
</tr>
<tr>
<td>4-4</td>
<td>Retrieval results using our proposed method.</td>
</tr>
<tr>
<td>5-1</td>
<td>An illustration of choosing a weak classifier using a decision stump. $x$ is a sample, $a$ is the threshold and $h$ is the weak classifier.</td>
</tr>
<tr>
<td>5-2</td>
<td>Classification using TBRLPBoost and ELPBoost on the training and testing sets of the pima, spambase, iris and spectf datasets.</td>
</tr>
<tr>
<td>6-1</td>
<td>Ground truth synthetic DTI field, the original DWI, the Rician noise affected DWI, estimation using MRE, VF, NLMt, NLM, and the proposed method.</td>
</tr>
<tr>
<td>6-2</td>
<td>The figures are the one slice of the estimated tensor fields using MRE, NLM, and the proposed method respectively.</td>
</tr>
<tr>
<td>6-3</td>
<td>The FA of the estimated tensor fields using MRE, NLM, the proposed method, and the principal eigenvectors of the ROIs.</td>
</tr>
</tbody>
</table>
Divergence measures provide a means to measure the pairwise dissimilarity between “objects”, e.g., vectors and probability density functions (pdfs). Kullback-Leibler (KL) divergence and the square loss (SL) function are two examples of commonly used dissimilarity measures which along with others belong to the family of Bregman divergences (BD). In this thesis, we present a novel divergence dubbed the Total Bregman divergence (TBD), which is inherently very robust to outliers, a very desirable property in many applications. Further, we derive the TBD center, called the $t$-center (using the $\ell_1$-norm), for a population of positive definite matrices is in closed form and show that it is invariant to transformations from the special linear group. This $t$-center, which is also robust to outliers, is then used in shape retrieval, diffusion tensor imaging (DTI) estimation, interpolation and segmentation. Furthermore, TBD is used to regularize the conventional boosting algorithms, which have been applied to applications in pattern classification.
CHAPTER 1
INTRODUCTION

1.1 Motivation

In applications that involve measuring the dissimilarity between two objects (numbers, vectors, matrices, functions, images and so on) the definition of a distance or divergence becomes essential. The state of the art has many widely used divergences. The square loss (SL) function has been used widely for regression analysis; Kullback-Leibler (KL) divergence [61], has been applied to compare two probability density functions (pdfs); the Mahalanobis distance is used to measure the dissimilarity between two random vectors from the same distribution. All the aforementioned divergences are special cases of the Bregman divergence which was introduced by Bregman in 1967 [18], and of late has been widely researched both from a theoretical and practical viewpoint [4, 10, 47, 84, 140]. At this juncture, it would be worth inquiring, why does one need yet another divergence? The answer would be that none of the existing
Divergences are statistically robust and one would need to use M-estimators from robust statistics literature to achieve robustness. This robustness however comes at a price, which is, computational cost and accuracy. Moreover, some of the divergences lack invariance to transformations such as similarity, affine etc. Such invariance becomes important when dealing with for example, segmentation, it is desirable to achieve invariance to similarity or affine transformations that two different scans of the same patient might be related by. More recently, several methods have adopted the square root density representation for representing shapes which then can be treated as points on a hypersphere and one can use the metric on the sphere to compare the shapes efficiently since the metric on the sphere is in closed form. We refer the reader to [96, 111] for more details. In this work, we propose a new class of divergences which allow us to perform pairwise comparison of objects that are invariant to rigid motions (translations and rotations) applied to the graph of the convex function used in defining the divergence. This divergence measures the orthogonal distance between the value of a convex and differentiable function at the first argument and its tangent at the second argument. We dub this divergence the total Bregman divergence (TBD).

1.2 \( \ell_1 \)-norm Center of TBD

Bregman divergence has been widely used in clustering, where cluster centers are defined using the divergence. We will also define a cluster center using the TBD in conjunction with the \( \ell_1 \)-norm that is termed the \( t \)-center. The \( t \)-center can be viewed as the cluster representative that minimizes the \( \ell_1 \)-norm TBD between itself and the members of a given population. We derive an analytic expression for the \( t \)-center which affords it an advantage over its rivals (for example, the \( \chi^2 \) distance based median of a population of densities). The key property of the \( t \)-center is that it is a weighted mean

\[ d(p, q) = \frac{\int (p-q)^2}{\int (p+q)^2} \]
and the weight is inversely proportional to the magnitude of the gradient of the convex function used in defining the divergence. And since noisy data and outliers have greater gradient magnitude their influence is underplayed. In other words, $t$-center puts more weight on the normal data and less weight on the “extraordinary” data. In this sense, $t$-center is a robust and stable representative and this property makes the $t$-center attractive in many applications. Since, the TBD can be viewed as a weighted BD with the weight being inversely proportional to the magnitude of the gradient of the convex function, the resulting $t$-center that we obtain can be viewed as a weighted median of sorts. This weighting scheme makes the $t$-center robust to noise and outliers, since it is inversely dependent on the gradient of the convex function. Another salient feature of the $t$-center is that it can be computed very efficiently due to its analytic form and this leads to efficient clustering.

1.2.1 Outline

The rest of the dissertation is organized as follows:

Chapter 2 introduces the definition of TBD and $t$-center as well as their properties. Chapter 3 introduces the application of TBD and $t$-center to DTI interpolation and segmentation. Chapter 4 utilizes TBD in shape retrieval applications on various datasets. Chapter 5 uses TBD to regularize the conventional linear programming based boosting (LPBoost) and uses it for classification. Chapter 6 proposes a variational framework for DTI estimation. Finally we present conclusions in Chapter 7.
CHAPTER 2
TOTAL BREGMAN DIVERGENCE

In this chapter, we will first define the TBD, and then give the definition of its $\ell_1$-norm based $t$-center. Finally, We will explore the properties of TBD and $t$-center.

2.1 Definition of TBD and Examples

We will first recall the definition of the conventional Bregman divergence [10] and then define the TBD. Both divergences are dependent on the corresponding convex and differentiable function $f : X \rightarrow \mathbb{R}$ that induces the divergences. It is worth pointing out that if $f$ is not differentiable, one can mimic the definition and proofs of properties with gradient substituted by any of its subdifferentials [140].

A geometrical illustration of the difference between TBD and BD is given in Figure 2-1. $d_f(x, y)$ (the dotted line) is Bregman divergence between $x$ and $y$ based on a convex and differentiable function $f$ where as $\delta_f(x, y)$ (bold line) is the TBD between $x$ and $y$ based on a convex and differentiable function $f$. The two arrows indicate the coordinate system. We can observe that $d_f(x, y)$ will change if we apply a rotation to the coordinate system, while $\delta_f(x, y)$ will not.

Figure 2-1. $d_f(x, y)$ is BD, $\delta_f(x, y)$ is TBD. This shows the $d_f(x, y)$ and $\delta_f(x, y)$ before and after rotating the coordinate system.
**Definition 1.** [10] The Bregman divergence $d$ associated with a real valued strictly convex and differentiable function $f$ defined on a convex set $X$ between points $x, y \in X$ is given by,

$$d_f(x, y) = f(x) - f(y) - \langle x - y, \nabla f(y) \rangle,$$  

(2-1)

where $\nabla f(y)$ is the gradient of $f$ at $y$ and $\langle \cdot, \cdot \rangle$ is the inner product determined by the space on which the inner product is being taken.

$d_f(\cdot, y)$ can be seen as the distance between the first order Taylor approximation to $f$ at $y$ and the function evaluated at $x$. Bregman divergence $d_f$ is non-negative definite, but it is not symmetric and does not satisfy the triangular inequality thus making it a divergence. As shown in Figure 2-1, Bregman divergence measures the ordinate distance. Additionally, it is not invariant to rigid transformations applied to $(x, f(x))$, such an invariance is however desirable in many applications. This motivated the development of an alternative that we call the total Bregman divergence (TBD).

**Definition 2.** The total Bregman divergence (TBD) $\delta$ associated with a real valued strictly convex and differentiable function $f$ defined on a convex set $X$ between points $x, y \in X$ is defined as,

$$\delta_f(x, y) = \frac{f(x) - f(y) - \langle x - y, \nabla f(y) \rangle}{\sqrt{1 + \|\nabla f(y)\|^2}},$$  

(2-2)

$\langle \cdot, \cdot \rangle$ is the inner product as in definition 1, and $\|\nabla f(y)\|^2 = \langle \nabla f(y), \nabla f(y) \rangle$ generally.

As shown in Figure 2-1, $d_f(\cdot, y)$ measures the ordinate distance, and $\delta_f(\cdot, y)$ measures the orthogonal distance. $\delta_f(\cdot, y)$ can be seen as a higher order “Taylor” approximation to $f$ at $y$ and the function evaluated at $x$. Since

$$\frac{1}{\sqrt{1 + \|\nabla f(y)\|^2}} = 1 - \frac{\|\nabla f(y)\|^2}{2} + O(\|\nabla f(y)\|^4)$$  

(2-3)

then

$$\delta_f(x, y) = d_f(x, y) - \frac{\|\nabla f(y)\|^2}{2} d_f(x, y) + O(\|\nabla f(y)\|^4)$$  

(2-4)
where $O(\cdot)$ is the Big O notation, which is usually small compared to the first term and thus one can ignore it without worrying about the accuracy of the result. Also, we can choose the higher order “Taylor” expansion if necessary.

Consider the Bregman divergence $d_f(\eta, \zeta) = f(\eta) - f(\zeta) - \langle \eta - \zeta, \nabla f(\zeta) \rangle$ obtained for strictly convex and differentiable generator $f$. Let $\hat{x}$ denote the point $(x, z = f(x))$ of $X \times \mathbb{R}$ lying on the graph $\mathcal{F} = \{ \hat{x} = (x, z = f(x)) \mid x \in X \}$. We visualize the Bregman divergence as the $z$-vertical difference between the hyperplane $H_\zeta$ tangent at $\hat{\zeta}$ and the hyperplane parallel to $H_\zeta$ and passing through $\hat{\eta}$:

$$d_f(\eta, \zeta) = f(\eta) - H_\zeta(\eta). \quad (2-5)$$

with $H_\zeta(x) = f(\zeta) + \langle x - \zeta, \nabla f(\zeta) \rangle$. If instead of taking the vertical distance, we choose the minimum distance between those two hyperplanes $H_\eta$ and $H_\zeta$, we obtain the total Bregman divergence (by analogy to total least square fitting where the projection is orthogonal).

Since the distance between two parallel hyperplanes $\langle x, a \rangle + b_1 = 0$ and $\langle x, a \rangle + b_2 = 0$ is $\frac{|b_1 - b_2|}{\|a\|}$, letting $a = (\nabla f(\zeta), -1)$, $b_1 = f(\eta) - \langle \eta, \nabla f(\zeta) \rangle$, and $b_2 = f(\zeta) - \langle \zeta, \nabla f(\zeta) \rangle$, we deduce that the total Bregman divergence is

$$\delta_f(\eta, \zeta) = \frac{d_f(\eta, \zeta)}{\sqrt{1 + \|\nabla f(\zeta)\|^2}} \quad (2-6)$$

Compared to the BD, TBD contains a weight factor (the denominator) which complicates the computations. However, this structure brings up many new and interesting properties and makes TBD an “adaptive” divergence measure in many applications. Note that, in practice, $X$ can be an interval, the Euclidean space, a $d$-simplex, the space of non-singular matrices or the space of functions. For instance, in the application to shape representation, we let $p$ and $q$ be two pdfs, and $f(p) = \int p \log p$, then $\delta_f(p, q)$ becomes what we will call the total Kullback-Leibler
<table>
<thead>
<tr>
<th>$X$</th>
<th>$f(x)$</th>
<th>$\delta_f(x, y)$</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{R}$</td>
<td>$x^2$</td>
<td>$\frac{(x-y)^2}{1+4y^2}$</td>
<td>Total square loss</td>
</tr>
<tr>
<td>$[0, 1]$</td>
<td>$x \log x + \bar{x} \log \bar{x}$</td>
<td>$\frac{x \log \frac{x}{\bar{x}} + \bar{x} \log \frac{x}{\bar{x}}}{1+y(1+\log y)^2+y(1+\log y)^2}$</td>
<td>Total logistic loss</td>
</tr>
<tr>
<td>$\mathbb{R}_+$</td>
<td>$-\log x$</td>
<td>$\frac{-\log \frac{x}{1+y^2}}{1+y^2}$</td>
<td>Total Itakura-Saito distance</td>
</tr>
<tr>
<td>$\mathbb{R}$</td>
<td>$e^x$</td>
<td>$\frac{e^x-e^y-(x-y)e^y}{1+e^y}$</td>
<td>Total squared Euclidean distance</td>
</tr>
<tr>
<td>$\mathbb{R}^d$</td>
<td>$|x|^2$</td>
<td>$\frac{\sqrt{1+4|y|^2}}{(x-y)^T A (x-y)}$</td>
<td>Total Mahalanobis distance</td>
</tr>
<tr>
<td>$\mathbb{R}^d$</td>
<td>$x^T A x$</td>
<td>$\frac{1+4|y|^2}{\sqrt{1+4|A|_F^2}}$</td>
<td>Total KL divergence</td>
</tr>
<tr>
<td>$\Delta^d$</td>
<td>$\sum_{j=1}^d x_j \log x_j$</td>
<td>$\frac{\sum_{j=1}^d x_j \log x_j}{\sqrt{1+4|y|_F^2}}$</td>
<td>Total squared Frobenius norm</td>
</tr>
<tr>
<td>$\mathbb{C}^{m \times n}$</td>
<td>$|x|_F^2$</td>
<td>$\frac{\sum_{j=1}^d x_j^2 (1+\log y_j)^2}{\sqrt{1+4|y|_F^2}}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 2-1. TBD $\delta_f$ corresponding to $f$. $\bar{x} = 1 - x$, $\bar{y} = 1 - y$, $x^T$ is the transpose of $x$. $\Delta^d$ is $d$-simplex.

...divergence (tKL). Note that for tKL, we define $\|\nabla f(q)\|^2 = \int (1 + \log q)^2 q$ specifically to make it integrable. Table 2-1 lists some TBDs with various associated convex functions.

### 2.2 Total Bregman Divergence Centers

In many applications of computer vision and machine learning such as image and shape retrieval, clustering and classification etc., it is common to seek a representative or template for a set of objects having similar features. This representative normally is a cluster center, thus, it is desirable to seek a center that is intrinsically representative and easy to compute. In this section, we will introduce the tBD-based centers, including the $\ell_p$-norm mean, the geometric and harmonic means respectively. Specifically, we will focus on the $\ell_1$-norm cluster center that we call the total center ($t$-center for short) and explore its properties.

**Definition 3.** Let $f : X \to \mathbb{R}$ be a convex and differentiable function and $E = \{x_1, x_2, \ldots, x_n\}$ be a set of $n$ points in $X$, then, the $\ell_p$-norm distance based on tBD, $\mathcal{A}_p^f$, between a point $x \in X$ and $E$ with associated $f$ and the $\ell_p$-norm is defined as

$$\mathcal{A}_p^f(x, E) = \left( \frac{1}{n} \sum_{i=1}^n (\delta_f(x, x_i))^p \right)^{1/p} \quad (2-7)$$
The $\ell_p$-norm mean $x_p$ of $E$ is defined as

$$x_p = \arg \min_x A_p(x, E). \quad (2-8)$$

It is well known that the conventional geometric, harmonic and arithmetic means (in the Euclidean case have a strong relationship. This is also the case for the tBD centers. When $p = 1$, the tBD center is the arithmetic mean of $\delta_f$, and when $p = -1$, the tBD mean becomes the harmonic mean, and when $p \to 0$, the mean becomes the geometric mean [5].

These means also bear the name of circumcenter ($p \to \infty$), centroid ($p = 2$) and median ($p = 1$) respectively. In this chapter, we call the median ($p = 1$) the $t$-center and we will derive an analytic form for the $t$-center and focus on its applications to shape retrieval.

The circumcenter $x_c$ in the limiting case amounts to solving for $x_c = \arg \min_x \max_i \delta_f(x, x_i)$. $x_c$ does not have an analytic form. In the case of BD, the circumcenter was shown to be combinatorially solvable in randomized linear-time using the linear property of Bregman bisectors [84]. Similarly results hold for total Bregman divergences. However, the main drawback of the circumcenter is the fact that it is highly vulnerable to outliers in the data, a well-known fact and one would have to use preprocessing techniques to get rid of these outliers.

The $\ell_2$-norm center is obtained by solving the $\ell_2$-norm minimization $x_m = \arg \min_x \sqrt{\frac{1}{n} \sum_{i=1}^n (\delta_f(x, x_i))^2}$. Similar to the circumcenter, $\ell_2$-norm mean does not have an analytic form and is not easy to compute. The mean $x_m$ is also sensitive to outliers in the data but not to the same degree as the circumcenter. In the following we will define the $\ell_1$-norm mean, which we call $t$-center.
2.2.1 $\ell_1$-norm t-center

Given a set $E = \{x_1, x_2, \cdots, x_n\}$, we can obtain the $\ell_1$-norm t-center $\bar{x}$ of $E$ by solving the following minimization problem

$$
\bar{x} = \arg\min_x \delta_1^t(x, E) = \arg\min_x \sum_{i=1}^n \delta_f(x, x_i)
$$

Using the $\ell_1$-norm t-center $\bar{x}$ has advantages over other centers since it has a closed form which makes its computationally attractive. The advantage is evident in the experiments presented subsequently.

The $t$-center is closely related to other kinds of tBD-based centers, like the geometric mean and harmonic mean. We will show in the next section that, based on tKL, the $t$-center of a set of pdfs is a weighted geometric mean of all pdfs, and the $t$-center of a set of symmetric positive definite matrices is the weighted harmonic mean of all matrices.

2.3 Properties of $t$-center

2.3.1 $t$-center Uniquely Exists

In [123], we showed that $t$-center exists, is unique and can be written in an explicit form for most commonly used cases. The proof made use of the convexity of $f$ and the Legendre dual space of TBD.

A real-valued function $f$ defined on an interval or on any convex subset of vector spaces is said to be convex if the following condition holds

$$
f(tx + (1-t)y) \leq tf(x) + (1-t)f(y),
$$

$x, y \in \text{dom}(f), \forall t \in [0, 1]$. $f$ is strictly convex if

$$
f(tx + (1-t)y) < tf(x) + (1-t)f(y), x, y \in \text{dom}(f),
$$

$\forall t \in (0, 1)$ and $x \neq y$. If $f$ is strictly convex and differentiable, then the slope will be monotonic and the second derivative if it exists will be positive.
Definition 4. Let \( x \in X \), \( X \) can be \( \mathbb{R}^n \), \( \mathbb{R}^n_+ \), or the set of probability distributions \( \mathbb{R}^n_+ \) \( (\sum_{i=1}^{n} x_i = 1) \) and \( f(x) \) be a convex function. We then have the dual coordinates through the Legendre transformation

\[
x^* = \nabla f(x),
\]

and the dual convex function

\[
f^*(x^*) = \sup_x \{\langle x, x^* \rangle - f(x)\}. \tag{2–13}
\]

For the Legendre transformation, the derivative of the function \( f \) becomes the argument to the function \( f^* \). In addition, if \( f \) is convex, then \( f^* \) satisfies the functional equation

\[
f^*(\nabla f(x)) = \langle x, \nabla f(x) \rangle - f(x). \tag{2–14}
\]

The Legendre transform is its own inverse, i.e. \( f^{**} = f \). Like the familiar Fourier transform, the Legendre transform takes a function \( f(x) \) and produces a function of a different variable \( x^* \). However, while the Fourier transform consists of an integration with a kernel, the Legendre transform uses maximization as the transformation procedure. The transform is especially well behaved if \( f(x) \) is a convex function. If \( f \) is a closed (lower-continuous) convex function, then \( f^* \) is also closed and convex.

We already know that the gradient at the \( \ell_1 \)-norm \( t \)-center \( \bar{x} \) is a weighted Euclidean average of the gradient of all the elements in the set \( E = \{x_1, x_2, \cdots, x_n\} \) \[123\], as given by

\[
\nabla f(\bar{x}) = \left( \sum_{i=1}^{n} w_i \nabla f(x_i) \right) / \left( \sum_{i=1}^{n} w_i \right), \tag{2–15}
\]

The weight \( w_i = (1 + \|\nabla f(x_i)\|^2)^{-1/2} \). Utilizing the Legendre dual transform, and let \( g \) be the Legendre dual function of \( f \), i.e.

\[
g(y) = \sup_x \{\langle y, x \rangle - f(x)\}, \tag{2–16}
\]
then
\[ \bar{x} = \nabla g(y_0), \] (2-17)
and \[ y_0 = \left( \sum_{i=1}^{n} w_i \nabla f(x_i) \right) / \left( \sum_{i=1}^{n} w_i \right), \]
which is a constant.

We will prove more properties of \( t \)-center, which are summarized in the following theorem.

**Theorem 2.1.** \([123]\) The \( t \)-center of a population of objects (densities, vectors etc.) for a given divergence exists, and is unique.

**Proof.** Suppose \( g \) is the Legendre dual function for the convex and differentiable function \( f \) in the space \( X \), and the dual space is denoted as \( Y \). Then \( \forall x \in X, \exists y \in Y: \)
\[ g(y) = \sup_x \{ \langle y, x \rangle - f(x) \}, \quad \text{and} \quad f(x) = \sup_y \{ \langle x, y \rangle - g(y) \}, \] (2-18)
where \( x \) and \( y \) satisfy \( y = \nabla f(x) \) and \( x = \nabla g(y) \). Since
\[ \nabla f(\bar{x}) = \left( \sum_{i=1}^{n} w_i \nabla f(x_i) \right) / \left( \sum_{i=1}^{n} w_i \right), \] (2-19)
which is to say
\[ \bar{y} = \left( \sum_{i=1}^{n} w_i \nabla f(x_i) \right) / \left( \sum_{i=1}^{n} w_i \right), \] (2-20)
and
\[ \bar{x} = \nabla g(\bar{y}). \] (2-21)

\[ \square \]

Actually, if \( f \) is given, then we will get the explicit form for \( \bar{y} \) and \( g \), and consequently \( \bar{x} \). \( \bar{x} \) does not have a uniform expression for all convex functions (this is unlike the Bregman divergence center, which is always the mean whatever the generating function \( f \) is \([10]\)), instead, \( \bar{x} \) is dependent on the convex functions (e.g., for tSL, \( \bar{x} \) is the weighted mean; for tKL, \( \bar{x} \) is the weighted geometric mean), which is reasonable since specific functions have specific meanings and thus result in specific representatives, i.e. \( t \)-centers.
Table 2-2. TBD $\delta_f$ and the corresponding $t$-center. $\hat{x} = 1 - x$, $\hat{y} = 1 - y$. $c$ is the normalization constant to make it a pdf, $w_i = \frac{1}{\sqrt{1 + \|\nabla f(x_i)\|_2^2}}$.

Theorem (2.1) reveals that the $t$-center uniquely exists, the gradient at the $t$-center has a closed form expression, which is a weighted average, and the weight is inversely proportional to the magnitude of the gradient of $f$ at the corresponding element. Table 2-2 lists the $t$-centers corresponding with various associated convex functions. For better illustration, we provide three concrete examples of TBD with their $t$-centers in explicit form.

- **tSL**: $f(x) = x^2$, the $t$-center

$$\hat{x} = \left(\frac{\sum w_i x_i}{\sum w_i}\right), w_i = \frac{1}{\sqrt{1 + 4x_i^2}}; \tag{2-22}$$

- **Exponentials**: $f(x) = e^x$, the $t$-center $\hat{x} = \left(\sum w_i x_i\right) / \left(\sum w_i\right)$, where $w_i = \left(1 + e^{2x_i}\right)^{-1/2}$;

- **tKL**: Let $f(q) = \int q \log q$, which is the negative entropy [116], and $E = \{q_1, q_2, \ldots, q_n\}$ be a set of probability density functions, the $t$-center is then given by Also, as an $\ell_1$-norm median, $t$-center is closely related with geometric center and harmonic center. The relationship is obvious when using the tKL between two pdfs. Let $f(q) = \int q \log q$, which is the negative entropy [116], and
\[ E = \{ q_1, q_2, \ldots, q_n \} \] be a set of pdfs, the \( t \)-center is then given by

\[ \bar{q} = c \prod_i q_i^{w_i / \sum_j w_j}, \quad w_i = \frac{1}{\sqrt{1 + \int \log q_i^2 dx}}, \quad (2-23) \]

where \( c \) is a normalization constant to make \( \bar{q} \) a pdf, i.e. \( \int x \bar{q}(x) dx = 1 \). \( \bar{q} \) is a weighted geometric mean of \( \{ q_i \}_{i=1}^n \). This is very useful in tensor interpolation, where a tensor is a symmetric positive definite (SPD) matrix. The TBD between two tensors \( Q_i \) and \( Q_j \) can be taken as the tKL between two normal distributions

\[ p(x; Q_i) = \frac{1}{(2\pi)^d \det Q_i} \exp \left( -\frac{1}{2} x^T Q_i^{-1} x \right), \quad (2-24) \]

\[ q(x; Q_j) = \frac{1}{(2\pi)^d \det Q_j} \exp \left( -\frac{1}{2} x^T Q_j^{-1} x \right), \quad (2-25) \]

and

\[ tKL(Q_i, Q_j) = tKL(p, q) = \frac{\log(\det(Q_i^{-1} Q_j)) + tr(Q_j^{-1} Q_i) - d}{2\sqrt{c + (\log(\det Q_i))^2} - d(1 + \log 2 \pi)^2 \log(\det Q_j)}, \quad (2-26) \]

where \( c = \frac{3d^2}{4} + \frac{d^2 \log 2 \pi}{2} + \left( \frac{d \log 2 \pi}{4} \right)^2 \), and \( d \) is the number of rows/columns of \( Q_i \). The \( t \)-center \( \bar{Q} \) for \( \{ Q_i \}_{i=1}^n \) is weighted harmonic mean:

\[ \bar{Q} = \left( \sum_i \frac{w_i^{-1}}{Q_i} \right)^{-1}, \quad (2-26) \]

where

\[ w_i = \frac{2 \sqrt{c + (\log(\det Q_i))^2} - d(1 + \log 2 \pi) \log(\det Q_i)}{\sum_j \left( 2 \sqrt{c + (\log(\det Q_j))^2} - d(1 + \log 2 \pi) \log(\det Q_j) \right)^{-1}}. \]

### 2.3.2 \( t \)-center is Statistically Robust to Outliers

Besides closed form expression, another fundamental property of \( t \)-center is that it is robust to outliers. We will present its theoretical robustness here and prove its practical robustness in the experimental section.

**Theorem 2.2.** [123] The \( t \)-center is statistically robust to outliers. The influence function of the \( t \)-center from outliers is upper bounded.
Proof. The robustness of t-center is analyzed by the influence function of an outlier $y$. Let $\bar{x}$ be the $t$-center of $E = \{x_1, \ldots, x_n\}$. When $\epsilon\% (\epsilon$ small) of outlier $y$ is mixed with $E$, $\bar{x}$ is influenced by the outliers, and the new center becomes to $\tilde{\bar{x}} = \bar{x} + \epsilon z(y)$. We call $z(y)$ the influence function. The center is robust when $z(y)$ does not grow even when $y$ is very large. The influence curve is given explicitly in the case of $\ell_1$-norm. $\tilde{\bar{x}}$ is the minimizer of

$$
(1 - \epsilon) \frac{1}{n} \sum_{i=1}^{n} \delta(x, x_i) + \delta(x, y),
$$

(2–27)

Hence, by differentiating the above function, setting it equal to zero at $\tilde{\bar{x}} = \bar{x} + \epsilon z$ and using the Taylor expansion, we have

$$
z(y) = \frac{G^{-1}}{w(y)} (\nabla f(y) - \nabla f(\bar{x})),
$$

(2–28)

where $G = \frac{1}{n} \sum_{i=1}^{n} \nabla \nabla f(\bar{x}) w(x_i)$, and $w(y) = \sqrt{1 + \|\nabla f(y)\|^2}$. Hence, the $t$-center is robust

$$
|z(y)| < \frac{G^{-1}}{w(y)} (\|\nabla f(y)\| + \|\nabla f(\bar{x})\|).
$$

(2–29)

Since $w(y) \geq \|\nabla f(y)\|$ and $w(y) \geq 1$, therefore,

$$
|z(y)| < G^{-1}(1 + \|\nabla f(\bar{x})\|) = c.
$$

(2–30)

To make this proof more understandable, we give a simple example of the Euclidean case when $f(x) = \frac{1}{2} \|x\|^2, x \in \mathbb{R}^d$. Then,

$$
G = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sqrt{1 + \|x_i\|^2}} I,
$$

(2–31)

and $z(y) = G^{-1} \frac{y}{\sqrt{1 + \|y\|^2}}$, when $\|y\|$ is large, this is approximated by $z(y) = G^{-1} \frac{y}{\|y\|}$, which implies $z$ is bounded, and the $t$-center is robust for large $\|y\|$.

Note that the influence function for the ordinary Bregman divergence is $z = y$, and hence is not robust.
2.3.3 Properties of TBD

TBD has many good properties which make it an appropriate divergence in many applications. For example, TBD is invariant to special linear group (SL(n)) transformations on the domain of diffusion tensor fields. This will be proved in Chapter 3. These properties will be proved and explained at length in the following chapters.
CHAPTER 3
APPLICATION OF TBD TO DTI ANALYSIS

This chapter is about using TBD, mainly total Kullback-Leibler divergence, and t-center to interpolate diffusion tensors in DT-MRI data and segmentation of tensor fields, specifically DT-MRI.

3.1 SPD Tensor Interpolation Applications

Define $tKL$, the total Kullback-Leibler divergence between symmetric positive definite (SPD) rank-2 tensors (SPD matrices), and show that it is invariant to transformations belonging to the special linear group $SL(n)$. Further, we compute the $t$-center using $tKL$ for SPD matrices, which has a closed form expression, as the weighted harmonic mean of the population of the tensors.

$tKL$ between order two SPD tensors/matrices $P$ and $Q$ is derived using the negative entropy of the zero mean Gaussian density functions they correspond to. Note that order two SPD tensors can be seen as covariance matrices of zero mean Gaussian densities. Suppose,

$$p(x; P) = \frac{1}{\sqrt{(2\pi)^n \det P}} \exp\left(-\frac{1}{2} x^t P^{-1} x\right),$$

$$q(x; Q) = \frac{1}{\sqrt{(2\pi)^n \det Q}} \exp\left(-\frac{1}{2} x^t Q^{-1} x\right),$$

then,

$$tKL(P, Q) = \frac{\int p \log \frac{p}{q} dx}{\sqrt{\int (1 + \log q)^2 q dx}} = \log(\det(P^{-1} Q)) + tr(Q^{-1} P) - n + \frac{\log(\det Q)}{2} - \frac{n(1+\log 2\pi)}{4} \log(\det Q),$$

where $c = \frac{3n}{4} + \frac{n^2 \log 2\pi}{2} + \frac{(n \log 2\pi)^2}{4}$. When an $SL(n)$ transformation is applied on $x$, i.e., $x \mapsto Ax$, then $P \mapsto A^t PA$ and $Q \mapsto A^t QA$. It is easy to see that

$$tKL(P, Q) = tKL(A^t PA, A^t QA), \quad \forall A \in SL(n).$$
which means that $tKL$ between SPD tensors is invariant under the group action, when the group member belongs to $SL(n)$. Given an SPD tensor set $\{Q_i\}_{i=1}^m$, its $t$-center $P^*$ can be obtained from (2–26) and

$$P^* = \left(\sum_i w_i Q_i^{-1}\right)^{-1}, \quad w_i = \frac{\mu_i}{\sum_j \mu_j}.$$  \hfill (3–5)

where $\mu_i = \left(2\sqrt{c + \frac{(\log(\det Q_i))^2}{4} - \frac{n(1 + \log 2\pi)}{2} \log(\det Q_i)}\right)^{-1}$. It can be seen that $w_i(Q_i) = w_i(A^t Q_i A), \forall A \in SL(n)$. If a transformation $A \in SL(n)$ is applied, the new $t$-center will be $\hat{P}^* = \left(\sum_i w_i (A^t Q_i A)^{-1}\right)^{-1} = A^t P^* A$, which means that if $\{Q_i\}_{i=1}^m$ are transformed by some member of $SL(n)$, then the $t$-center will undergo the same transformation.

Also we can compute the $tSL$ between $P$ and $Q$ from the density functions $p$, $q$ and $f(p) = \int p^2 dx$, and the result is

$$tSL(P, Q) = \frac{\int (p - q)^2 dx}{\sqrt{1 + \int (2q)^2 dx}} = \frac{1/\sqrt{\det(2P)} + 1/\sqrt{\det(2Q)} - 2/\sqrt{\det(P + Q)}}{(2\pi)^n + 4(2\pi)^n/\sqrt{\det(3Q)},}$$

also, $tSL(P, Q) = tSL(A^t PA, A^t QA), \forall A \in SL(n)$, which means that $tSL$ is also invariant under $SL(n)$ transformations. Similarly, we can also prove that the total Itakura-Saito distance and total squared Euclidean distance between SPD matrices are invariant under $SL(n)$ transformations. For the rest of this chapter, we will focus on $tKL$.

There are several ways to define the distance between SPD matrices, e.g. using the Frobenius norm [127], Riemannian metric [57, 64, 79, 94], symmetrized KL divergence [79, 85, 128] and the log-Euclidean distance [103], respectively defined as

$$d_F(P, Q) = \|P - Q\|_F = \sqrt{\text{tr}((P - Q)^t(P - Q))},$$  \hfill (3–6)

$$d_R(P, Q) = \sqrt{\sum_{i=1}^n \log^2 \lambda_i},$$  \hfill (3–7)

27
where $\lambda_i, i=1, \cdots, n$, are eigenvalues of $P^{-1}Q$,

$$KL_s(P, Q) = \frac{1}{4} tr(Q^{-1}P + P^{-1}Q - 2I). \quad (3-8)$$

$$LE(P, Q) = \| \log(P) - \log(Q) \|_F. \quad (3-9)$$

$d_F(., .)$ is not invariant to transformations in $SL(n)$, $d_R(., .)$, $KL_s(., .)$ and $LE(., .)$ are invariant to transformations in $GL(n)$, but none of them are robust to outliers encountered in the data for e.g., in tensor interpolation. For $d_R(., .)$, neither its mean nor its median is in closed form, which makes it computationally very expensive as the population size and the dimensionality of the space increases. Here, the Karcher mean denoted by $P_M^*$ is defined as the minimizer of the sum of squared Riemannian distances and the median $P_R^*$ is minimizer of the sum of Riemannian distances [41]

$$P_M^* = \arg \min_P \sum_{i=1}^m d_R^2(P, Q_i),$$

and

$$P_R^* = \arg \min_P \sum_{i=1}^m d_R(P, Q_i).$$

For simplicity in notation, we denote $d_R^2(P, Q_i)$ by $d_M(P, Q_i)$, and $P_M^* = \arg \min_P \sum_{i=1}^m d_M(P, Q_i)$.

Even though there are many algorithms (for example [40, 41, 64, 94]) to solve the geodesic mean and median, most of them adopt an iterative (gradient descent) method. Performing gradient descent on matrix manifolds can be tricky and rather complicated in the context of convergence issues (see [1]) and hence is not preferred over a closed form computation.

$KL_s$ in (3–8) has a closed form mean [79, 128], which is $G(A(Q_1, \cdots, Q_m), H(Q_1, \cdots, Q_m))$, where $A$ is the arithmetic mean of $\{Q_i\}_{i=1}^m$, $H$ is the harmonic mean of $\{Q_i\}_{i=1}^m$, and $G$ is the geometric mean of $A$ and $H$. However, it can be shown that the mean computed using sum of squared $KL_s$ divergences is not statistically robust since all tensors are treated equally. This is because, neither $KL_s$ nor the sum of squared $KL_s$s are robust functions. This is demonstrated in the experimental results.
to follow. First we will observe the visual difference between the aforementioned divergences/distances. Figure 3-1 shows the isosurfaces centered at the identity matrix with radii $r = 0.1, 0.5$ and $1$ respectively. From left to right are $d_F(P,I) = r$, $d_R(P,I) = r$, $KL_s(P,I) = r$ and $tKL(P,I) = r$. These figures indicate the degree of anisotropy in the divergences/distances.

![Figure 3-1](image.png)

**Figure 3-1.** The isosurfaces of $d_F(P,I) = r$,

$d_R(P,I) = r$, $KL_s(P,I) = r$ and $tKL(P,I) = r$ shown from left to right. The three axes are eigenvalues of $P$.

### 3.2 Piecewise Constant DTI Segmentation

Given a noisy diffusion tensor image (DTI) $T_0(x)$ – a field of positive definite matrices, our model for DTI segmentation is based on the Mumford-Shah functional [83],

$$E(C, T) = \alpha \int_\Omega \delta(T_0(x), T(x))dx + \int_{\Omega-C} |\nabla T(x)| dx + \beta |C|, \quad (3-10)$$

where $\alpha$ and $\beta$ are control parameters, $\delta$ is the $tKL$ divergence same as (3-3). $\Omega$ is the region of the tensor field, $T(x)$ is an approximation to $T_0(x)$, which can be discontinuous only along $C$. However, in a simplified segmentation model, a field $T_0(x)$ can be represented by piecewise constant regions [21, 83]. Therefore, we consider the following binary segmentation model for DTI,

$$E(C, T_1, T_2) = \int_R \delta(T_0(x), T_1)dx + \int_{R^2} \delta(T_0(x), T_2)dx + \beta |C|, \quad (3-11)$$
$T_1$ is the $t$-center of DTI for the region $R$ inside the curve $C$ and $T_2$ is the $t$-center of the DTI for the region $R^c$ outside $C$,

$$ T_1 = \left( \sum_{Q_i \in R} w_i Q_i^{-1} \right)^{-1}, \quad w_i = \frac{\mu_i}{\sum_{Q_j \in R} \mu_j}. \quad (3-12) $$

$$ T_2 = \left( \sum_{Q_i \in R^c} w_i Q_i^{-1} \right)^{-1}, \quad w_i = \frac{\mu_i}{\sum_{Q_j \in R^c} \mu_j}. \quad (3-13) $$

The Euler Lagrange equation of (3–11) is

$$ (\delta(T_0(x), T_1) - \delta(T_0(x), T_2) + \beta \kappa) \mathbf{N} = 0, \quad (3-14) $$

where $\kappa$ is the curvature, $\kappa = \nabla \cdot \frac{\nabla C}{|\nabla C|}$, and $\mathbf{N}$ is the normal of $C$, $\mathbf{N} = \nabla C$. $C$ can be updated iteratively according to the following equation

$$ \frac{\partial C}{\partial t} = -(\delta(T_0(x), T_1) - \delta(T_0(x), T_2) + \beta \kappa) \mathbf{N}. \quad (3-15) $$

At each iteration, we will fix $C$, update $T_1$ and $T_2$ according to (3–12) and (3–13), and then freeze $T_1$ and $T_2$ to update $C$.

In the level set formulation of the active contour [73], let $\phi$ be the signed distance function of $C$ and choose it to be negative inside and positive outside. Then the curve evolution equation (3–15) can be reformulated using the level set framework

$$ \frac{\partial \phi}{\partial t} = (\delta(T_0(x), T_1) - \delta(T_0(x), T_2) + \beta \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|}) \nabla \phi. \quad (3-16) $$

### 3.3 Piecewise Smooth DTI Segmentation

For complicated DTI images, the piecewise constant assumption does not hold. Therefore, we have to turn to the more general model, the piecewise smooth segmentation. In this chapter, we follow Wang et al’s [128] model but replace their
divergence \((KL_s)\) with \(tKL\), resulting in the following functional

\[
E(C, T) = \alpha \int_R \delta(T_0(x), T_R(x)) dx + \int_{R_c} \delta(T_0(x), T_{R_c}(x)) dx + \int_{\Omega - C} \rho(T(x)) dx + \beta |C|,
\]

the third term measures the lack of smoothness of the field using the Dirichlet integral \([52]\), giving us the following curve evolution equation:

\[
\frac{\partial C}{\partial t} = \left( \alpha \delta(T_0(x), T_{R_c}) + \sum_{y \in N_{R_c}(x)} \delta(T_{R_c}, T_{R_c}(y)) \right) \mathbf{N} - \left( \alpha \delta(T_0(x), T_R) + \sum_{y \in N_R(x)} \delta(T_R, T_R(y)) - \beta \kappa \right) \mathbf{N},
\]

where, \(N_R\) and \(N_{R_c}\) are the sets of x’s neighboring pixels inside and outside of the region \(R\) respectively. In the discrete case, one can use appropriate neighborhoods for 2D and 3D. We apply the two stage piecewise smooth segmentation algorithm in \([128]\) to numerically solve this evolution.

### 3.4 Experimental Results

We performed two sets of experiments here, (i) tensor interpolation and (ii) tensor field segmentation. The experimental results are compared with those obtained by using other divergence measures discussed above.

### 3.5 Tensor Interpolation Experiments

SPD tensor interpolation is a crucial component of DT-MRI analysis involving segmentation, registration and atlas construction \([12]\), where a robust distance/divergence measure is very desirable. First, we perform tensor interpolation on a set of tensors with noise and outliers. We fix an SPD tensor \(\tilde{G}\) as the ground truth tensor and generate noisy tensors \(G\) from it by using a Monte Carlo simulation method as was done in \([90, 93]\). This entails, given a \(b\)-value (we used \(1200 \text{s/mm}^2\)) a zero gradient baseline image \(\tilde{S}_0\), and six vectors \(\{q_i\}_{i=1}^6\), the magnitude of the noise free complex-valued
diffusion weighted signal is given by, $S_i = S_0 \exp(-b q_i^T G q_i)$. We add Gaussian distributed (mean zero and variance $\sigma^2 = 0.1$) noise to the real and imaginary channels of this complex valued diffusion weighted signal and take its magnitude to get $S_i = S_0 \exp(-b q_i^T G q_i)$. $S_i$ then has a Rician distribution with parameters $(\sigma, \bar{S}_i)$. $G$ is obtained by fitting the $\{S_i\}_{i=1}^6$ using Log-Euclidean tensor fitting. To generate outliers, we first generate a $3 \times 3$ matrix $Z$, with each entry drawn from the normal distribution, and the outlier tensor $G$ is given by $G = \exp(ZZ^t)$. We compute the eigen-vectors $U$ and eigen-values $V$ of $G$, i.e. $G = U \text{diag}(V) U^t$, and then rotate $U$ by a rotation matrix $r(\alpha, \beta, \gamma)$ in 3D, to get $U_o$, where $\alpha$, $\beta$ and $\gamma$ are uniformly distributed in $[0, \pi/2]$. Then the outlier is given by $G_o = U_o \text{diag(randperm}(V))) U_o^t$. To generate an image, the same process is repeated at every voxel.

The tKL $t$-center, the geometric median, the geometric mean, the $KL_s$ mean and the log-Euclidean mean $M$ for 21 SPD tensors along with 0, 5, 10, 15, and 20 outliers are then computed. The difference between the various means/medians, and the ground truth $\bar{G}$ are measured in four ways, using the Frobenius norm $\|\bar{G} - M\|_F$, the $\ell_\infty$ distance $\|\bar{G} - M\|_\infty$, the angle between the principal eigenvectors of $M$ and $\bar{G}$, and the difference of fractional anisotropy index (FA) [91, 97] for $M$ and $\bar{G}$, i.e., $|FA(M) - FA(\bar{G})|$. The results are shown in Table 3-1 from left to right, top to bottom.

From the tables, we can see that the $t$-center yields the best approximation to the ground truth, and faithfully represents the directionality and anisotropy. The robustness of $t$-center over others such as the Riemannian mean and median, $KL_s$ mean, and log-Euclidean mean is quite evident from this table. Even though geometric median seems to be competitive to the $t$-center obtained using $tKL$ in the case of lower percentage of outliers, the geometric median computation however is much slower than that of $t$-center computation. This is because the $t$-center for $tKL$ has a closed form while the geometric median does not. Table 3-2 shows the CPU time to find the tensor mean/median using different divergences. We use 1000 SPD tensors, along with 0, 10,
Table 3-1. The difference between the various means/medians, and the ground truth.

<table>
<thead>
<tr>
<th># of outliers</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$tKL$</td>
<td>0.8857</td>
<td>1.0969</td>
<td>1.1962</td>
<td>1.3112</td>
<td>1.4455</td>
</tr>
<tr>
<td>$d_R$</td>
<td>1.3657</td>
<td>1.4387</td>
<td>1.5348</td>
<td>1.6271</td>
<td>1.7277</td>
</tr>
<tr>
<td>$d_M$</td>
<td>1.4291</td>
<td>1.5671</td>
<td>1.8131</td>
<td>2.1560</td>
<td>2.5402</td>
</tr>
<tr>
<td>$KL_s$</td>
<td>3.4518</td>
<td>3.5744</td>
<td>3.7122</td>
<td>4.1040</td>
<td>4.3235</td>
</tr>
<tr>
<td>$LE$</td>
<td>1.0638</td>
<td>1.5721</td>
<td>1.6249</td>
<td>1.6701</td>
<td>1.8227</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># of outliers</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>$tKL$</td>
<td>0.1270</td>
<td>0.3875</td>
<td>0.6324</td>
<td>1.8575</td>
<td>2.9572</td>
</tr>
<tr>
<td>$d_R$</td>
<td>0.4218</td>
<td>0.9058</td>
<td>8.9649</td>
<td>17.4854</td>
<td>44.9157</td>
</tr>
<tr>
<td>$d_M$</td>
<td>0.8147</td>
<td>1.1985</td>
<td>10.1576</td>
<td>21.8003</td>
<td>43.7922</td>
</tr>
<tr>
<td>$KL_s$</td>
<td>0.8069</td>
<td>0.9134</td>
<td>14.9706</td>
<td>26.1419</td>
<td>44.9595</td>
</tr>
<tr>
<td>$LE$</td>
<td>0.6456</td>
<td>0.9373</td>
<td>8.9937</td>
<td>17.0636</td>
<td>44.7635</td>
</tr>
</tbody>
</table>

100, 500 and 800 outliers respectively. All tensors are generated in the same way as described in the first experiment. The time is averaged by repeating the experiment 10 times on a PC, with Intel(R) Core(TM) 2 Duo CPU P7370, 2GHz, 4GB RAM, on 32-bit Windows Vista OS. Table 3-1 and 3-2 depict the superior robustness to outliers and the computational efficiency in estimating the $tKL$ $t$-center for SPD tensor interpolation in comparison to its rivals.

3.6 Tensor Field Segmentation Experiments

We now describe experiments on segmentation of synthetic and real DTI images.
Table 3-2. Time (seconds) spent in finding the mean/median using different divergences.

<table>
<thead>
<tr>
<th>Divergences % of outliers</th>
<th>0</th>
<th>10</th>
<th>100</th>
<th>500</th>
<th>800</th>
</tr>
</thead>
<tbody>
<tr>
<td>( tKL )</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>( KL_s )</td>
<td>0.03</td>
<td>0.03</td>
<td>0.03</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>( d_R )</td>
<td>0.58</td>
<td>1.67</td>
<td>4.56</td>
<td>93.11</td>
<td>132.21</td>
</tr>
<tr>
<td>( d_M )</td>
<td>0.46</td>
<td>1.42</td>
<td>3.13</td>
<td>72.74</td>
<td>118.05</td>
</tr>
<tr>
<td>( LE )</td>
<td>0.02</td>
<td>0.02</td>
<td>0.03</td>
<td>0.03</td>
<td>0.03</td>
</tr>
</tbody>
</table>

3.6.1 Segmentation of Synthetic Tensor Fields

The first synthetic tensor field is composed of two types of homogeneous tensors. Figure 3-2 depicts the synthetic data and the segmentation results. We added different levels of noise to the tensor field using the method described in section 3.5, segmented it using the aforementioned divergences and compared the results using the dice coefficient [82]. We also added different percentages of outliers to the tensor fields and segmented the resulting tensor fields. Figure 3-3 depicts the comparison of segmentation results from different methods using the dice coefficient, for varying noise levels with \( \sigma \) varying from \((0, 0.02, 0.04, \ldots, 0.2)\). Figure 3-4 displays the comparison of dice coefficient with different percentage \((0, 5, 10, \ldots, 50)\) of outliers. The results show that even in the presence of large amounts of noise and outliers, \( tKL \) yields very good segmentation results in comparison to rivals. However, in our experiments, we observed that the segmentation accuracy is inversely proportional to the variance of the outlier distribution.

3.6.2 Segmentation of DTI Images

In this section, we present segmentation results on real DTI images from a rat spinal cord, an isolated rat hippocampus and a rat brain. The data were acquired using a PGSE with \( TR=1.5s, TE=28.3ms, \text{bandwidth}=35Khz, 21 \text{diffusion weighted images with a } b\text{-value of } 1250s/mm^2 \text{ were collected. A } 3 \times 3 \text{ diffusion tensor in each DT image is illustrated as an ellipsoid [128], whose axes’ directions, and lengths correspond to its eigen-vectors, and eigen-values respectively. The same initialization is used for each} \)
Figure 3-2. From left to right are initialization, intermediate step and final segmentation.

Figure 3-3. Dice coefficient comparison for $tKL$, $KL_s$, $d_R$, $d_M$ and $LE$ segmentation of synthetic tensor field with increasing level (x-axis) of noise.

Figure 3-4. Dice coefficient comparison for $tKL$, $KL_s$, $d_R$, $d_M$ and $LE$ segmentation of synthetic tensor field with increasing percentage (x-axis) of outliers.
Figure 3-5. The segmentation results using \( tKL, KL_s, d_R, d_M \) and \( LE \).

We apply the piecewise constant segmentation model on a single slice (108 × 108) of the rat spinal cord, and apply the piecewise smooth segmentation model on the molecular layer from single slices of size (114 × 108) for rat corpus callosum (CC) and (90 × 90) for the rat hippocampus respectively. Figure 3-5ABC show the initialization, Figure 3-5D shows the segmentation results, and Table 3-3 records their execution time. The results confirm that when compared to other divergences, \( tKL \) yields a more accurate segmentation in a significantly shorter amount of CPU time.
Table 3-3. Time (seconds) comparison for segmenting the rat spinal cord, corpus callosum and hippocampus using different divergences.

<table>
<thead>
<tr>
<th>Divergences</th>
<th>tKL</th>
<th>KLs</th>
<th>dR</th>
<th>dM</th>
<th>LE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time for Cord</td>
<td>33</td>
<td>68</td>
<td>72</td>
<td>81</td>
<td>67</td>
</tr>
<tr>
<td>Time for CC</td>
<td>87</td>
<td>183</td>
<td>218</td>
<td>252</td>
<td>190</td>
</tr>
<tr>
<td>Time for hippocampus</td>
<td>159</td>
<td>358</td>
<td>545</td>
<td>563</td>
<td>324</td>
</tr>
</tbody>
</table>

Apart from segmentation in 2D slices, we also demonstrate 3D DTI image segmentation using the proposed divergence. Figure 3-6 depicts the process of segmenting rat corpus callosum (114 × 108 × 11) using the piecewise constant segmentation model. Figure 3-6A-B-C-D is a 2D slice of the corresponding evolving surface, from left to right are initialization, intermediate steps and final segmentation. Figure 3-6E is a 3D view of the segmentation result. The result demonstrates that $tKL$ can segment this white matter bundle quite well.

Figure 3-6. $tKL$ segmentation of a 3D rat corpus callosum.
3.7 Discussions

In this chapter, we developed an application of using TBD for DTI interpolation and segmentation. Specifically, we derived an explicit formula for the $t$-center which is the TBD-based median, that is robust to outliers. In the case of SPD tensors, the $t$-center was shown to be $SL(n)$ invariant. However, the story is not yet complete and further investigations are currently underway.

The robustness (to outliers) property of TBD was demonstrated here via applications to SPD tensor field interpolation and segmentation. The results favorably demonstrate the competitiveness of our newly defined divergence in comparison to existing methods not only in terms of robustness, but also in terms of computational efficiency and accuracy as well.
In this chapter, we consider the family of total Bregman divergences (tBDs) as an efficient and accurate “distance” measure to quantify the (dis)similarity between shapes. We use the $t$-center as the representative of a set of shapes and propose a new clustering technique namely, the total Bregman soft clustering algorithm. We evaluate the tBD, $t$-center and the soft clustering algorithm on shape retrieval applications.

The shape retrieval framework is composed of three steps: (1) extraction of the shape boundary points (2) affine alignment of the shapes and use of a Gaussian mixture model (GMM) \cite{24, 54} to represent the aligned boundaries, and (3) comparison of the GMMs using tBD to find the best matches given a query shape. To further speed up the shape retrieval algorithm, we perform hierarchical clustering of the shapes using our total Bregman soft clustering algorithm. This enables to compare the query with a small subset of shapes which are chosen to be the cluster $t$-centers. The proposed method is evaluated on various public domain 2D and 3D databases, and demonstrate comparable or better results than state-of-the-art retrieval techniques.

4.1 Literature Review for Shape Retrieval

As the number of images on the Internet, in public databases and in biometric systems grows larger and larger, efficient and accurate search algorithms for retrieval of the best matches have become crucial for a variety of tasks. Therefore, image retrieval becomes more and more fundamental in computer vision and plays an indispensable role in many potential applications. In contemporary literature, there are mainly two types of algorithms for image retrieval, key-words based and content based. Key-words are an important and easy to use features for representation and retrieval of images. However, though efficient, key-words are very subjective, since different people may use different key-words to index the same image. Therefore, the accuracy of key-words based retrieval is very limited. Hence there is interest in the idea of retrieval based
on image features [8, 66, 67, 76, 96] such as texture, color, shape, and so on. Of these, shape is considered more generic and is one of the best for recognition as studies [16] have shown. Shape comparison and classification is very often used in the areas of object detection [48, 71] and action recognition [132]. Therefore many researchers [106, 115, 121, 136, 137] have been developing algorithms for improving the performance of shape retrieval. An efficient modern shape retrieval scheme has the following two components: an accessible and accurate shape representation, and an efficient as well as robust distance/divergence measure. There are many ways to represent shapes, for example, axial representation [65, 109], primitive-based representation [42], constructive representation [36], reference points and projection based representation [28], cover-based representation [98], histograms of oriented gradients [72]. Of these, contour based representation in object recognition methods [23, 35, 56, 81, 88, 108, 138] have shown great performance. The probability density function (pdf) has emerged as a successful representation for shape contours [32, 70, 96]. It is known to be mathematically convenient and robust to rigid transformations, noise, occlusions and missing data. Bearing this in mind, we choose to represent shapes as pdfs.

Again as the number of images in the website is increasing, the requirement of retrieving speed becomes stronger (e.g. TinEye reverse image search and Google image retrieval, both require realtime response). An accurate and fast retrieving method will lead to great convenience for shape retrieval in real life. in this chapter, we will bring up a fast and accurate shape retrieval method, which represents shapes using mixture of Gaussians (GMM), and divide the shapes into smaller groups using total Bregman divergence soft clustering algorithm, each cluster having a representative which is TBD based $\ell_1$-norm center, noted as $t$-center [70]. The $t$-center is a weighted combination of all elements in the cluster, has a closed form expression, and is robust to noise and outliers. We store the shapes using $k$-tree structure, with the inner nodes $t$-centers, and
the leaf nodes GMMs of shapes, which we will explain at length later in the experimental part. During retrieval, we only need to compare the query with the $t$-centers, and once the best match representative is obtained, we compare the query with the $t$-centers of the relative sub-clusters, recursively doing so, until the required number of best matches are found.

### 4.2 Total Bregman Divergence Clustering

When retrieving in a small database, it is possible to apply the brute-force search method by comparing the query shape with each shape in the database one by one, however, in the case of retrieving in a large database, it becomes impractical to use this brute-force search method because of the extremely high computational cost. To make real time retrieval in a large database, we turn to a far more efficient strategy, namely a divide and conquer strategy. First, utilizing the top down approach, we split the whole database into subclusters, and repeat the same approach on the subclusters. The divergence between shapes from the same cluster should be less than the divergence between images from different clusters, then choose a representative for each cluster, and assign each shape to the nearest cluster. There are two ways of assigning a shape to a cluster, assign it to a cluster completely, or assign the shape to a cluster according to some probability. The former corresponds to hard clustering, while the later case is soft clustering.

#### 4.2.1 Total Bregman Divergence Hard Clustering

Total Bregman divergence hard clustering assigns one object to a cluster whose center is the closest to the object. The assignment process is as following. We compare the object with the cluster centers, and pick the center that has smallest divergence to the object, and then assign the object to the relative cluster. After all the assignment, we recompute the cluster center, and then repeat the aforementioned steps.

When retrieving, we only need to compare the query with each cluster's representative, if the divergence is larger than some threshold, we will prune this whole cluster, thus
reducing the number of unnecessary comparisons and consequently speeding up
the retrieval. Once the representative that best matches the query is obtained, we
will recursively split the corresponding cluster into smaller clusters, and repeat the
aforementioned process on each subcluster, and in this way, successfully seeking out
the best matches. More conveniently, the step of splitting can be done off line, which
saves a lot of computation time. To partition the database into smaller clusters efficiently
and accurately, we utilize an idea similar to that of k-tree (a hybrid of the B-Trees and
k-means [19, 62]), by first dividing the database into k clusters, calculating the t-center
as the representative of each cluster according to the total Bregman hard clustering
algorithm (Algorithm 1), and repeat the above process on the resulting clusters to
get k sub-clusters, and accordingly get a hierarchy of clusters. The t-centers for the
hierarchical clusters form the k-tree. To be explicit, in the k-tree, every key is a mixture
of Gaussians, every inner node (including the root) has 1 to k keys, each of which is
the t-center of all keys in its children nodes, and the key for a leaf node is a mixture of
Gaussians for an individual shape. The k-tree illustration is shown in Figure 4-1.

Figure 4-1. k-tree diagram. Every key is a GMM. Each key in the inner nodes is the
t-center of all keys in its children nodes.
During retrieval, one only needs to compare the query with the representatives, and once the best match representative is obtained, we compare the query with the \( t \)-centers of the relative sub-clusters, recursively doing so, until the required number of best matches are found. Furthermore, the steps of clustering can be parallelized using multi-core, multi-thread level parallelism.

**Algorithm 1 Total Bregman Hard Clustering Algorithm**

**Input**: \( X = \{x_i\}_{i=1}^n \).

**Output**: \( \{C_j, \bar{m}_j\}_{j=1}^k \), \( C_j \) is the \( j^{th} \) cluster with cluster center \( \bar{m}_j \)

**Initialization**: Randomly choose \( k \) elements from \( P \) as the \( t \)-centers \( \{\bar{m}_j\}_{j=1}^k \) and set \( C_j = \emptyset \)

repeat
   for \( i = 1 \) to \( n \) do
      {Compute \( x_i \) belongs to which cluster}
      \( C_j \leftarrow C_j \cup \{x_i\} \), where \( j = \arg \min_{\tilde{j}} \delta_F(x_i, \bar{m}_{\tilde{j}}) \)
      if the radius of cluster \( C_j \) is too large, then \( C_j \) will be randomly divided into two clusters, and goto repeat.
   end for
   for \( j = 1 \) to \( k \) do
      {Update cluster centers}
      \( \bar{m}_j \leftarrow t \)-center for cluster \( C_j \) (Equation (2–17))
   end for
until Some convergence criterion is met (e.g., unchanged assignment.)

The number of clusters \( c \) is determined by the radius of the cluster, i.e. the maximum divergence from all elements to the cluster center. If the radius of the cluster is too large, then the cluster will be divided into two clusters.

### 4.2.2 Total Bregman Divergence Soft Clustering

Total Bregman divergence soft clustering is another clustering strategy, which assigns the object to more than one clusters according to some probability that is related with TBD.

Soft clustering assigns one element to more than one clusters and every element has fractional membership in several clusters. Soft Clustering is very useful in numerous areas, like parameter estimation for mixture of exponential family distributions (e.g. Gaussian mixture models, Binomial mixture models and hidden Markov models). Also
in many cases, soft clustering is more reliable and accurate than hard clustering. One popular soft clustering algorithm is the expectation-maximization (EM) algorithm, which has been used for finding maximum likelihood estimations of parameters in statistical models. EM is conceptually very simple, easy to implement and works well when the missing information is small and the dimension of data is low. However, the convergence rate can be excruciatingly slow as approaching a local optima and it will fail when the amount of missing data is large or when the dimension of data is high.

Total Bregman soft clustering is a special kind of soft clustering which assigns an element to several clusters according to the probability determined by TBD. The steps of total Bregman soft clustering are very similar to those of EM algorithm, which, basically speaking, is composed of two parts. The first is to assign every element to the clusters by the probability that maximizes the intra cluster similarity and minimizes the inner cluster similarity; the second is to update the cluster centers and relative probabilities of each cluster. The algorithm for total Bregman soft clustering is shown in Algorithm 2.

**Algorithm 2** Total Bregman Soft Clustering Algorithm

**Input:** $X = \{x_i\}_{i=1}^{N}$, number of clusters $c$.

**Output:** $M = \{m_j\}_{j=1}^{c}$ and $Q = \{q_j\}_{j=1}^{c}$, $m_j$ is the cluster center for the $j^{th}$ cluster with probability $q_j$.

**Initialization:** Randomly choose $c$ elements from $X$ as $M$ and their corresponding probabilities as $Q$.

**repeat**

  {assign $x_i$ to clusters}

  **for** $i = 1$ to $N$ **do**

    **for** $j = 1$ to $c$ **do**

    $q(j | x_i) \leftarrow \frac{q_j \exp(-\delta_t(m_j, x_i))}{\sum_{j=1}^{c} q_j \exp(-\delta_t(m_j, x_i))}$

    **end for**

  **end for**

  {update cluster centers}

  **for** $j = 1$ to $c$ **do**

    $q_j \leftarrow \frac{1}{N} \sum_{i=1}^{N} q(j | x_i)$

    $m_j \leftarrow t$-center for cluster $j$ (Equation (2–17))

  **end for**

**until** The change of the results between two consecutive iterations is below some sensitivity threshold.
TBD soft clustering converges very fast, and can do very well in both low and high dimensional space. We compared our TBD soft clustering algorithm with Bregman soft clustering algorithm [10] using synthetic examples and by real applications of shape retrieval.

4.2.2.1 Synthetic experiments for soft clustering

We did four experiments using the same data sets as Banerjee et al. used in [10]. The first one is based on several 1D data sets of 300 samples each, generated from mixtures of Gaussian and Binomial models respectively. Both mixture models had three components with equal priors centered at 10, 20 and 40. The standard deviation of the Gaussian distribution was set to 5 and the number of trials of the Binomial distribution was set to 300 so as to make the two models somewhat similar to each other, in terms that the variance is almost the same for all the models. We also use the same method to generate 2D and 5D data sets and compare the algorithms on them.

The accuracy of clustering was measured by using the normalized mutual information (NMI) [113] between the predicted clusters (the predicted cluster number is 5) and original clusters that generating the samples, and the results were averaged over 30 trials.

Table 4-1 lists the NMI resulted from soft clustering using BD and TBD. Gaussian mixture and Binomial mixture represent the models that generated the data sets. $d_{\text{Gaussian}}$ and $d_{\text{Binomial}}$ represent the Bregman divergence for Gaussian and Binomial distributions while $\delta_{\text{Gaussian}}$ and $\delta_{\text{Binomial}}$ represent the TBD for Gaussian and Binomial distributions. In detail,

$$\delta_{\text{Gaussian}}(x_1, x_2) = \frac{f(x_1) - f(x_2) - (x_1 - x_2, \nabla f(x_2))}{\sqrt{1 + \|\nabla f(x_2)\|^2}}$$  (4–1)

$f(x) = q(x) \log q(x)$ and $q(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$. $\delta_{\text{Binomial}}$ is in the same format as (4–1) but $q(x) = \binom{n}{x} \rho^x (1 - \rho)^{n-x}$, and $\rho$ is the probability for a single success.

For Table 4-1, in ((a), (b)), and (c)), rows 1 and 2 correspond to the NMI between the original and the predicted clusters obtained by applying the Bregman clustering
(a)

<table>
<thead>
<tr>
<th>Measure</th>
<th>Gaussian Mixture</th>
<th>Binomial Mixture</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{\text{Gaussian}}$</td>
<td>0.73660±0.00142</td>
<td>0.54998±0.00035</td>
</tr>
<tr>
<td>$d_{\text{Binomial}}$</td>
<td>0.54307±0.00066</td>
<td>0.72982±0.00379</td>
</tr>
<tr>
<td>$\delta_{\text{Gaussian}}$</td>
<td>0.75076±0.00193</td>
<td>0.55089±0.00018</td>
</tr>
<tr>
<td>$\delta_{\text{Binomial}}$</td>
<td>0.57899±0.00773</td>
<td>0.74450±0.00218</td>
</tr>
</tbody>
</table>

(b)

<table>
<thead>
<tr>
<th>Measure</th>
<th>Gaussian Mixture</th>
<th>Binomial Mixture</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{\text{Gaussian}}$</td>
<td>0.43922±0.03577</td>
<td>0.38246±0.05020</td>
</tr>
<tr>
<td>$d_{\text{Binomial}}$</td>
<td>0.36805±0.05513</td>
<td>0.42987±0.04747</td>
</tr>
<tr>
<td>$\delta_{\text{Gaussian}}$</td>
<td>0.59385±0.11910</td>
<td>0.50518±0.07280</td>
</tr>
<tr>
<td>$\delta_{\text{Binomial}}$</td>
<td>0.52136±0.09187</td>
<td>0.57063±0.05173</td>
</tr>
</tbody>
</table>

(c)

<table>
<thead>
<tr>
<th>Measure</th>
<th>Gaussian Mixture</th>
<th>Binomial Mixture</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{\text{Gaussian}}$</td>
<td>0.39869±0.00049</td>
<td>0.38246±0.05020</td>
</tr>
<tr>
<td>$d_{\text{Binomial}}$</td>
<td>0.19765±0.00025</td>
<td>0.14205±0.05619</td>
</tr>
<tr>
<td>$\delta_{\text{Gaussian}}$</td>
<td>0.52360±0.00018</td>
<td>0.41516±0.05320</td>
</tr>
<tr>
<td>$\delta_{\text{Binomial}}$</td>
<td>0.36883±0.00035</td>
<td>0.52164±0.04173</td>
</tr>
</tbody>
</table>

Table 4-1. The clustering results for the 1D, 2D and 5D data sets by applying the Bregman and TBD soft clustering algorithms. The algorithm using the Bregman divergences $d_{\text{Gaussian}}$ and $d_{\text{Binomial}}$ [10] respectively. Rows 3 and 4 correspond to the NMI yielded by the TBD clustering algorithm using $\delta_{\text{Gaussian}}$ and $\delta_{\text{Binomial}}$ respectively. The numbers in Table 4-1 illustrate that using $\delta_{\text{Gaussian}}$ to cluster data sets generated by Gaussian mixture and using $\delta_{\text{Binomial}}$ to cluster data sets generated by Binomial mixture gives better NMI than using $d_{\text{Gaussian}}$ and $d_{\text{Binomial}}$. More importantly, using $\delta_{\text{Gaussian}}$ to measure the data sets generated by Binomial mixture gives much better NMI than using $d_{\text{Gaussian}}$ to do the same job. This is also true with $\delta_{\text{Binomial}}$ and $d_{\text{Binomial}}$. This is very useful in the real life, because often times, we don’t know the model that generates the data, instead we have to blindly choose some divergence measure. But one thing that we are sure now is that TBD is always better than Bregman divergence when using the same generating functions.
Table 4-2. NMI beween the original clusters and the clusters got from TBD and BD soft clustering algorithms. $\tilde{c}$ is the predicted number of clusters.

<table>
<thead>
<tr>
<th>$\tilde{c}$</th>
<th>$d_{\text{Gaussian}}$</th>
<th>$\delta_{\text{Gaussian}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.55627±0.00020</td>
<td>0.556364±0.00034</td>
</tr>
<tr>
<td>8</td>
<td>0.58417±0.00122</td>
<td>0.58554±0.00109</td>
</tr>
<tr>
<td>6</td>
<td>0.59958±0.00112</td>
<td>0.70202±0.00154</td>
</tr>
<tr>
<td>5</td>
<td>0.73660±0.00142</td>
<td>0.75076±0.00193</td>
</tr>
<tr>
<td>3</td>
<td>0.86115±0.00141</td>
<td><strong>0.98658 ± 0.00120</strong></td>
</tr>
</tbody>
</table>

Remarks: From Table 4-1, we can see that with the dimension increasing, TBD soft clustering becomes more and more accurate than BD clustering, and the performance difference between TBD clustering and BD clustering becomes larger.

Also, fix the dimension of data and the original number of clusters $c$, and let the predicted cluster number $\tilde{c}$ approximate to $c$, the NMI of TBD clustering increases faster than that of BD clustering. It can be seen from Table 4-2 using Gaussian generative model where the original number of clusters $c = 3$. This point is very fundamental in segmentation in the process of partitioning an image into multiple regions or sets and also can typically be used to locate objects and boundaries.

From Table 4-2, we can see TBD soft clustering behaves very well even the predicted number of clusters is incorrect.

4.3 TBD Applications–Shape Retrieval

The task of shape retrieval is to find the best match from a database of shapes to the query shape. In this section, we propose an efficient and accurate method for shape retrieval that includes an easy to use shape representation, and an analytical shape dissimilarity divergence measure. Also, we present an efficient scheme to solve the computationally expensive problem encountered when retrieving from a large database. The scheme is composed of clustering and efficient pruning, which will be elaborated on in Section 4.3.3.
4.3.1 Shape Representation

A time and space efficient shape representation is fundamental to shape retrieval. Given a segmented shape (or a binary image), we use a mixture of Gaussians (GMM) [24, 54] to represent it. The procedure for obtaining the GMM from a shape is composed of three steps. First, we extract the points on the shape boundary or surface (to make it robust, for each point on the boundary, we also picked its closest 2 neighbors off the boundary), since MPEG-7 shapes are binary, the points that have nonzero gradient lie on the boundary (this step uses one line MATLAB code). After getting the boundary points for every shape, we use the affine alignment proposed by Ho et al. [53] to align these points to remove the effect of rigid transformations, e.g., given two sets of points \( \{x_i\}_{i=1}^m \) and \( \{y_j\}_{j=1}^n \), we can find affine alignment \((A, b), A \in \text{GL}(2)\) , \(b \in \mathbb{R}^2\), such that \( g(A, b) = \sum_i \min_j \{(Ax_i + b - y_j)^2\} \) achieves minimum, and then we use the aligned \( \{\bar{x}_i | \bar{y}_i = Ax_i + b\}_{i=1}^m \) to represent the original point set \( \{x_i\}_{i=1}^m \). This step is also very simple due to the explicit solution of \((A, b)\), and it only takes several lines of MATLAB code to implement. Finally, we compute the GMM from the aligned boundary points. A parametric GMM is a weighted combination of Gaussian kernels, which can be written as

\[
p(x) = \sum_{i=1}^{m} a_i \mathcal{N}(x; \mu_i, \Sigma_i), 0 \leq a_i \leq 1, \sum_{i=1}^{m} a_i = 1,
\]

\(m\) is the number of components; \(\mathcal{N}(x; \mu_i, \Sigma_i)\) is the Gaussian density function with mean \(\mu_i\), variance \(\Sigma_i\), and weight \(a_i\) in the mixture model. The mixture model is obtained through applying the EM algorithm and iteratively optimizing the centers and widths of the Gaussian kernels. \(m\) should be as small as possible, but make the determinant of the covariance for each component not large (we found that \(m=10\) is a good compromise for MPEG-7 database). The above process is portrayed using the flow

\(^1\text{GL}(2):\) The set of \(2 \times 2\) invertible matrices.
chart shown below.

```
Shape \rightarrow Boundary points extraction and alignment \rightarrow GMM
```

Some concrete examples of the application of the flow chart are shown in Figure 4-2. The columns from left to right in Figure 4-2 are original shapes, aligned boundaries, and GMM with 10 components. The dot inside each ellipse is the mean of the corresponding Gaussian density function, and the transverse and conjugate diameter correspond to the eigen values of the covariance matrix.

### 4.3.2 Shape Dissimilarity Comparison Using tSL

After getting the GMM representation of each shape, we use tSL to compare two GMMs, and take the difference as the dissimilarity between the corresponding shapes. Note that the $tSL$ for two GMMs is in closed form (see below). Suppose two shapes have the following GMMs $p_1$ and $p_2$,

\[ p_1(x) = \sum_{i=1}^{m} a_i^{(1)} N(x; \mu_i^{(1)}, \Sigma_i^{(1)}) , \quad (4-3) \]
\[ p_2(x) = \sum_{i=1}^{m} a_i^{(2)} N(x; \mu_i^{(2)}, \Sigma_i^{(2)}) . \quad (4-4) \]

Since

\[ \int N(x; \mu_1, \Sigma_1) N(x; \mu_2, \Sigma_2) dx = N(0; \mu_1 - \mu_2, \Sigma_1 + \Sigma_2) , \]

we can arrive at

\[ tSL(p_1, p_2) = \frac{\int (p_1 - p_2)^2 dx}{\sqrt{1 + \int 4p_2^2 dx}} = \frac{d_1 + d_2 - d_{1,2}}{\sqrt{1 + 4d_2}} , \quad (4-5) \]
Figure 4-2. Left to right: original shapes; aligned boundaries; GMM with 10 components.
where

\[
d_1 = \sum_{i,j=1}^{m} a_i^{(1)} a_j^{(1)} N(0; \mu_i^{(1)} - \mu_j^{(1)}, \Sigma_i^{(1)} + \Sigma_j^{(1)}),
\]

(4–6)

\[
d_2 = \sum_{i,j=1}^{m} a_i^{(2)} a_j^{(2)} N(0; \mu_i^{(2)} - \mu_j^{(2)}, \Sigma_i^{(2)} + \Sigma_j^{(2)}),
\]

(4–7)

\[
d_{1,2} = 2 \sum_{i,j=1}^{m} a_i^{(1)} a_j^{(2)} N(0; \mu_i^{(1)} - \mu_j^{(2)}, \Sigma_i^{(1)} + \Sigma_j^{(2)}),
\]

(4–8)

where \( N(0; \mu, \Sigma) = \frac{1}{(\sqrt{2\pi})^p \sqrt{\det(\Sigma)}} \), and \( e \) is the dimension of \( \mu \). Given a set of GMMs \( \{ p_i \}_{i=1}^n \), \( p_i = \sum_{j=1}^{m} a_j^{(l)} N(x; \mu_i^{(l)}, \Sigma_i^{(l)}) \), its \( t \)-center can be obtained from equation (2–17), which is

\[
\bar{p} = \frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i}, \quad w_i = (1 + 4d_i)^{-1/2},
\]

(4–9)

\[
d_i = \sum_{i,j=1}^{m} a_i^{(l)} a_j^{(l)} N(0; \mu_i^{(l)} - \mu_j^{(l)}, \Sigma_i^{(l)} + \Sigma_j^{(l)}).
\]

(4–10)

We evaluate the dissimilarity between the GMM of the query shape and the GMMs of the shapes in the database using \( tSL \), and the smallest dissimilarities correspond to the best matches.

### 4.3.3 Shape Retrieval in MPEG-7 Database

The proposed divergence is evaluated on the shape retrieval problem using the MPEG-7 database [63], which consists of 70 different objects with 20 shapes per object, for a total of 1400 shapes. This is a fairly difficult database to perform shape retrieval because of its large intraclass variability, and, for many classes, small interclass dissimilarity, and furthermore, there are missing parts and occlusions in many shapes.

We cluster the database into hierarchical clusters, calculate their \( t \)-centers and compare the query shape with the \( t \)-centers hierarchically. For the clustering part, we applied both hard clustering and soft clustering. For hard clustering, we apply a variation of \( k \)-tree method by setting \( k = 10 \) at the first level of clustering, \( 7 \) at the second level, \( 5 \) at the third level and \( 2 \) at all following levels, so the average number of shapes in each
cluster is 140, 20, 4, 2, and 1. For soft clustering, we append a semi-hard assignment to the soft clustering algorithm, i.e., after the soft clustering converges, we will assign the shape \( x_i \) to cluster \( C_j \), if \( p(C_j | x_i) \geq a \). We use \( a = 1/2 \), so that one shape can be assigned to at most 2 clusters, but the cluster center may be dependent on \( x_i \) even though \( x_i \) is not assigned to that cluster finally.

The clustering process is a coarse to fine procedure, which greatly enhances efficiency while guaranteeing accuracy. Also, we compare the clustering accuracy of \( tSL \), \( \chi^2 \) and \( SL \) soft and hard clustering by a reasonable measure, which is the optimal number of categories per cluster (denoted by \( |C|^* \), \( |C| \) represents the cardinality of \( C \), i.e., the number of categories in \( C \)) divided by the average number of categories in each cluster (denoted by \( \text{Avg}(|C|) \)). For example, at the first level clustering, there are 10 clusters \( \{ C_i \}_{i=1}^{10} \), with an average of 140 shapes per cluster, and thus, \( |C|^* = 140/20 = 7 \); \( \text{Avg}(|C|) = \frac{\sum_{i=1}^{10} |C_i|}{10} \). The smaller number of categories per cluster, the higher the clustering accuracy is, and the more accurately will the different categories be separated. The optimal clustering accuracy is 1. Figure 4-3 compares the clustering accuracy of \( tSL \), \( \chi^2 \) and \( SL \) soft and hard clustering, which shows that \( tSL \) soft clustering has a striking clustering accuracy, implying substantial capability to detect outliers, occlusion, missing parts, and strong ability to distinguish shapes from different categories.

We put here several groups of retrieval results in Figure 4-4, the first shape in each figure is the query, and the other shapes are shown from left to right, up to down according to the ascending order of the divergence to the query. The results show that our method can deal very well with scale, rotation, pose, occlusion, missing parts, great intraclass dissimilarity and large interclass similarity.

The evaluation of accuracy for retrieving in the whole MPEG-7 database is based on the well recognized criterion, recognition rate [34, 63, 76, 96]. Each shape is used as a query and the top 40 matches are retrieved from all 1400 shapes. The maximum
Figure 4-3. Comparison of clustering accuracy of $tSL$, $\chi^2$ and $SL$, versus average number of shapes per cluster.

possible number of correct retrievals for each query is 20, and hence there are a total of 28,000 possible matches with the recognition rate reflecting the number of correct matches divided by this total.

Table 4-3 lists the recognition rate we obtained and comparison with some other existing techniques. Note that our method gives very high recognition rate, even though it is not as good as [7, 115], but our method does not need any preprocessing of the shapes or any postprocessing of the similarities. The simplicity and speed of our method are incomparable.
Figure 4-4. Retrieval results using our proposed method.
<table>
<thead>
<tr>
<th>Technique</th>
<th>Recognition rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMM+soft clustering + tSL</td>
<td>93.41</td>
</tr>
<tr>
<td>GMM+hard clustering + tSL [70]</td>
<td>89.1</td>
</tr>
<tr>
<td>Shape-tree [34]</td>
<td>87.7</td>
</tr>
<tr>
<td>IDSC + DP + EMD [67]</td>
<td>86.56</td>
</tr>
<tr>
<td>Hierarchical Procrustes [76]</td>
<td>86.35</td>
</tr>
<tr>
<td>IDSC + DP [66]</td>
<td>85.4</td>
</tr>
<tr>
<td>Shape L’Anne Rouge [96]</td>
<td>85.25</td>
</tr>
<tr>
<td>Generative Models [121]</td>
<td>80.03</td>
</tr>
<tr>
<td>Curve Edit [105]</td>
<td>78.14</td>
</tr>
<tr>
<td>SC + TPS [15]</td>
<td>76.51</td>
</tr>
<tr>
<td>Visual Parts [63]</td>
<td>76.45</td>
</tr>
<tr>
<td>CSS [80]</td>
<td>75.44</td>
</tr>
<tr>
<td>Perceptual S. + IDSC + LCDP [115]</td>
<td>95.60</td>
</tr>
<tr>
<td>IDSC + Mutual Graph [60]</td>
<td>93.4</td>
</tr>
<tr>
<td>IDSC + LCDP + unsupervised GP [137]</td>
<td>93.32</td>
</tr>
<tr>
<td>IDSC + LCDP [137]</td>
<td>92.36</td>
</tr>
<tr>
<td>IDSC + LP [8]</td>
<td>91.61</td>
</tr>
<tr>
<td>Contour Flexibility [136]</td>
<td>89.31</td>
</tr>
<tr>
<td>Perceptual + IDSC [115]</td>
<td>88.39</td>
</tr>
<tr>
<td>SC + IDSC + Co-Transduction [7]</td>
<td>97.72</td>
</tr>
</tbody>
</table>

Table 4-3. Recognition rates for shape retrieval in the MPEG-7 database.

4.3.4 Brown Database

Additionally, we apply our proposed method to the Brown database [106], which contains 9 shape categories, where each shape category has 11 different segmented binary shapes and 99 shapes in total. We use GMM to represent each shape, the number of components for each GMM is decided using the same way as in the MPEG-7 experiment, and compare the difference of shapes using the tSL between their corresponding GMMs. We tested our method using the criteria as in [8, 34, 106, 115, 121]: every shape is taken as the query, and compare it with all the shapes in the database, find the best 10 matches, and check the number of correct matches, i.e., the number of shapes which belongs to the same category as the query shape. This process is repeated by taking each one of the 99 shapes in the whole data set as the query shape. Then we check the total correct matches for the $i^{th}$ shape,
Table 4-4. Recognition rates for shape retrieval from the Brown database.

\[ i = 1, 2, \ldots, 10 \] (the maximum number of correct matches is 99), which are shown in Table 4-4. We can see that our method gives perfect result.

### 4.3.5 Swedish Leaf Data Set

The Swedish leaf data set [110] contains isolated leaves from 15 different Swedish tree species, with 75 leaves per species, with 1125 shapes in total. We use the classification criteria as in [34, 67, 115, 137], which used the 1-nearest-neighbor approach to measure the classification performance. For each leaf species, 25 samples are selected as a template and the other 50 are selected as targets. We use GMM to represent each shape, and use TBD soft clustering algorithm to cluster the shapes into different clusters. Shape classification results on this data set are shown in Table 4-5, from which we can see that our accessible shape representation plus TBD soft clustering algorithm gives the best results.

<table>
<thead>
<tr>
<th>Technique</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
<th>7th</th>
<th>8th</th>
<th>9th</th>
<th>10th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape-tree [34]</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>97</td>
<td>93</td>
<td>86</td>
</tr>
<tr>
<td>IDSC [67]</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>98</td>
<td>98</td>
<td>97</td>
<td>97</td>
<td>98</td>
<td>94</td>
<td>79</td>
</tr>
<tr>
<td>Shock-Graph Edit [106]</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>98</td>
<td>98</td>
<td>97</td>
<td>96</td>
<td>95</td>
<td>93</td>
<td>82</td>
</tr>
<tr>
<td>Generative Models [121]</td>
<td>99</td>
<td>97</td>
<td>99</td>
<td>98</td>
<td>96</td>
<td>94</td>
<td>83</td>
<td>75</td>
<td>48</td>
<td></td>
</tr>
</tbody>
</table>

Table 4-5. Recognition rates for shape retrieval from the Swedish leaf database.
4.3.6 3D Princeton Shape Benchmark

Our method performed very well in the domain of 2D shape retrieval and it can be extended very easily to higher dimensional space. We evaluate our method on the Princeton Shape Benchmark (PSB) [107] containing 1814 3D models, which is divided into the training set (907 models in 90 classes) and the testing set (907 models in 92 classes). We evaluate our method on the testing set, and compare our results with others in three ways, Nearest Neighbor (NN), Discounted Cumulative Gain (DCG) and Normalized DCG (NDCG) using the software provided in PSB [107]. Our method outperforms all other methods when using NN criteria, and can find the first closest matches that belong to the query class more accurately.

<table>
<thead>
<tr>
<th>Technique</th>
<th>ours</th>
<th>CRSP</th>
<th>DSR</th>
<th>DBF</th>
<th>DBI</th>
<th>SIL</th>
<th>D2</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>72.3</td>
<td>67.9</td>
<td>66.5</td>
<td>68.6</td>
<td>60.9</td>
<td>55.7</td>
<td>31.1</td>
</tr>
<tr>
<td>DCG</td>
<td>66.7</td>
<td>66.8</td>
<td>66.5</td>
<td>65.9</td>
<td>61.4</td>
<td>59.7</td>
<td>43.4</td>
</tr>
<tr>
<td>NDCG</td>
<td>16.1</td>
<td>16.4</td>
<td>15.9</td>
<td>14.9</td>
<td>7</td>
<td>4.1</td>
<td>-24.4</td>
</tr>
</tbody>
</table>

Table 4-6. Retrieval comparison with other methods (CRSP [92], DSR [2], DBF [2], DBI [126], SIL [126], D2 [89]) on PSB.

4.4 Discussions

This chapter presents TBD hard and soft clustering algorithms. We use Gaussian mixture models (GMMs) to represent shapes, cluster the shapes into subclusters and store the shapes using $k$-tree. The clustering is quite fast because the cluster center is in closed form and is only dependent on the GMM means and variances. The $k$-tree makes it efficient to retrieval with logarithmic comparisons, and furthermore, each comparison is very -fast because the TBD between two GMMs also has explicit form. Our method is fast and space efficient, can be applied to low and high dimensional spaces, no need to do any preprocessing about the shapes, or any postprocessing on the dissimilarities, and is robust to rigid transformations, outliers, pose, occlusion, missing parts, and has better or similar results compared with those from the state-of-the-art techniques.
Boosting is a well known machine learning technique used to improve the performance of weak learners and has been successfully applied to computer vision, medical image analysis, computational biology and other fields. Several boosting methods have been proposed in the literature, such as AdaBoost, GentleBoost, SoftBoost, BrownBoost, LPBoost, RBoost and their variations. A critical step in boosting algorithms involves update of the data sample distribution, however, most existing boosting algorithms use updating mechanisms that lead to overfitting and instabilities during evolution of the distribution which in turn results in classification inaccuracies. Regularized boosting has been proposed in literature as a means to overcome these difficulties.

In this chapter, we propose a novel total Bregman divergence (tBD) regularized LPBoost, termed tBRLPBoost [69]. tBD was developed in Chapter 2, which is statistically robust and we prove that tBRLPBoost requires a constant number of iterations to learn a strong classifier and hence is computationally more efficient compared to other regularized Boosting algorithms in literature. Also, unlike other boosting methods that are only effective on a handful of datasets, tBRLPBoost works well on a variety of datasets. We present results of testing our algorithm on many public domain databases including the UCI machine learning repository, the OASIS MRI brain database, the CNS embryonal tumor dataset, the Colon tumor dataset, the Leukemia cancer dataset and the Epilepsy dataset. We also present comparisons to several other state-of-the-art methods. Numerical results show that the proposed algorithm has much improved performance in efficiency and accuracy over other methods.

5.1 Introduction to Boosting

Classification is a very important task in numerous areas including but not limited to computer vision, pattern recognition and image processing. Ensemble classifiers have
been in vogue for hard classification problems where use of single classifiers have not been very successful. Boosting is such an ensemble classification tool. It generates a strong classifier through a linear convex combination of weak classifiers. A weak classifier is usually only required to be better than random guessing. However, the strong classifier greatly boosts the performance of the weak classifiers and performs quite well on the whole dataset. Therefore, Boosting has become a very popular method to improve the accuracy of classification and has been widely used in computer vision, pattern recognition and other areas, for example, in image and object categorization [33, 68, 95, 102], retrieval [117], tracking [87, 139], face detection [125], and object recognition [11, 88, 118] etc.

The main idea behind the boosting algorithm is that at each iteration, it will learn a weak classifier from the training samples that follow a distribution. The weighted weak classifier is then added to the strong classifier. This weight is typically related to the weak classifier’s accuracy. The higher the accuracy, the larger the weight, and vise versa. After the weak classifier is added to the strong classifier, the distribution of the samples is updated. The data is reweighted following the rule that samples that are misclassified tend to gain weight and samples that are classified correctly tend to lose weight. Thus the weight will be very large if the sample has been misclassified by many previously learned weak classifiers, and the weight will be very small if the sample has been classified correctly by many previously learned weak classifiers. Therefore, the future weak classifiers to be learned will be focused more on the samples that many previous weak classifiers misclassified. This method has been proven to be effective in various classification problems and thus motivated a lot of research in the machine learning community. Schapire and Singer [104] proposed AdaBoost minimizing the exponential hinge loss, followed by the inception of LPBoost [29] which maximizes the minimum margin between classes, and others proposed many variations and improvements to AdaBoost, namely TotalBoost, SoftBoost, LogitBoost, GentleBoost.
[46], and entropy regularized LPBoost. These aforementioned boosting algorithms are mainly for binary classification, i.e., the database contains only two classes. To solve the multiclass classification problems, several researchers extended the binary boosting algorithms to the multiclass case, see for example, [45, 141]. As the demand for classifying large scale datasets, dynamic environments and multimedia datasets increased, various online boosting techniques were proposed [6, 75, 101] and used on face recognition and tracking among other applications.

In this chapter, we present a regularized LPBoost that is based on a recently introduced robust divergence measure, for binary classification that can easily be generalized to the multiclass case. This divergence measure is called total Bregman divergence (tBD) which is based on the orthogonal distance between the convex generating function of the divergence and its tangent approximation at the second argument of the divergence. tBD is naturally robust and leads to efficient algorithms for soft and hard clustering. For more details, we refer the reader to [123].

Based on earlier work on an upper bound for the number of update iterations in the entropy regularized LPBoost algorithm (ELPBoost), by Warmuth et al. [131], we present a constant bound on the number of iterations for our tBD-regularized boosting algorithm (tBRLPBoost). We also show empirical results that depict the efficiency of our proposed classification algorithm in comparison to ELPBoost and others. Finally, due to its computational efficiency, our boosting method is a promising candidate for online classification.

The rest of the chapter is organized as follows. In Section 5.2, we briefly review the boosting literature. In section 5.3, we introduce our algorithm, the total Bregman divergence regularized LPBoost, dubbed as tBRLPBoost, and investigate its properties, like robustness and efficiency. In section 5.4, we investigate the same properties experimentally. In the same section, we also validate/test our algorithm on a number of datasets and compare the results with state-of-the-art boosting algorithms and other
competitive classifiers. Finally, we conclude the chapter and discuss possible future work in section 5.5.

5.2 Previous Work

In the literature of boosting, numerous techniques exist which solve various optimization problems. AdaBoost is the origin for the modern boosting algorithms. It sets up a framework for all the following boosting techniques. AdaBoost performs very well in many cases, however, it does not do well on noisy datasets [31].

BrownBoost [44] however overcomes this noise problem. Because in BrownBoost, it is assumed that only noisy examples will be misclassified frequently, thus the samples that are repeatedly misclassified are taken as noisy data and are removed. In this way, the final classifier is learned from the noise-free samples, and therefore, the generalization error of the final classifier may be much smaller than that learned from noisy and non-noisy samples. BrownBoost works well on a variety of problems, but it does not maximize the margin between different classes which limit its performance in many cases.

Another version of boosting is the LPBoost [29] which, with a soft margin is a linear programming algorithm that can maximize the minimum soft margin between different classes, and perform very well on natural data. However, it does not have any iteration upper bound and sometimes, it requires linear time ($O(N)$, $N$ is the size of the training dataset) to get a good strong classifier, which is computationally expensive when the dataset is large. SoftBoost [130] on the other hand can maximize the minimum margin up to $\epsilon$ accuracy, with $O(\log N/\epsilon^2)$ number of iterations. This is a great improvement except it suffers from the common annoying problem of slow start, which makes SoftBoost rather unsatisfactory. More recently, the entropy regularized LPBoost (ELPBoost) was introduced in [131], which is a regularized version of LPBoost. It uses relative entropy, the Kullback-Leibler divergence (KL) between the updated distribution and the original distribution to regularize the conventional LPBoost.
ELPBoost overcomes the slow start issue and it performs as well as LPBoost on natural datasets with an iteration bound of $O(\log N/\epsilon^2)$. Nevertheless, it is not robust to noisy datasets and $O(\log N/\epsilon^2)$ is still an expensive computational requirement.

To avoid all the aforementioned problems, we present a robust and efficient boosting algorithm in this chapter. We show that it has constant upper bound for the number of iterations required to achieve a good strong classifier. The algorithm uses the total Bregman divergence (tBD) to regularize LPBoost. tBD was recently proposed in [70, 123] and it has three salient features. First, it is invariant to rigid transformations of the coordinate system used to specify the convex generating function. Secondly, it is intrinsically robust to noise and outliers. Finally, the representative of a set of objects (whether they are scalars, vectors or functions) has a closed form expression, which makes it computationally attractive. These properties have been verified in the applications of image clustering, retrieval [70] and medical image segmentation [123].

The tBD $\delta$ associated with a real valued strictly convex and differentiable function $f$ defined on a convex set $\mathcal{X}$ between points $d, \tilde{d} \in \mathcal{X}$ is defined by,

$$\delta_f(d, \tilde{d}) = \frac{f(d) - f(\tilde{d}) - \langle d - \tilde{d}, \nabla f(\tilde{d}) \rangle}{\sqrt{1 + \|\nabla f(\tilde{d})\|^2}},$$  \(5-1\)

where, $\langle \cdot, \cdot \rangle$ denotes the standard inner product, $\nabla f(y)$ is the gradient of $f$ at $y$, and $\|\nabla f(y)\|^2 = \langle \nabla f(y), \nabla f(y) \rangle$. tBD is a class of divergences which has the specific format as in (5–1). In this chapter, we will focus on the total Kullback-Leibler (tKL) divergence and use it to regularize the LPBoost. We show that the maximum number of iterations for this tBRLPBoost to learn a strong classifier is a constant. In other words, the iteration upper bound is independent of the number of the training samples, which makes it easily scalable and computationally efficient in comparison to the existing methods. Also, this algorithm can perform very well on noisy datasets due to the intrinsic robustness of tBD.
5.3 tBRLPBoost: Total Bregman Divergence Regularized LPBoost

TtBRLPBoost uses tBD to regularize the conventional LPBoost. The purpose of regularization is to make the boosting algorithm converge quickly and smoothly, and increase its robustness to noise and outliers. We now briefly present a mathematical description of the conventional boosting. Given the input $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the domain and $\mathcal{Y}$ is the range, the goal is to learn a function $H: \mathcal{X} \mapsto \mathcal{Y}$. In the binary classification problem, $\mathcal{X}$ is a set of feature vectors, and $\mathcal{Y} = \{1, -1\}$. The training samples are $\{(x_n, y_n)\}_{n=1}^N$, $x_n \in \mathcal{X}$ is the feature vector, and $y_n \in \{1, -1\}$ is the class label for $x_n$. Given the training samples, the task of boosting is to learn the strong classifier $H(x) = \text{sign}(\sum_{t=1}^T w_t h_t(x))$ to approximate $\tilde{H}$. Here, $h_t$ belongs to the weak classifier space $\mathcal{H}$, and $h_t: \mathcal{X} \mapsto \mathbb{R}$. In addition, $h_t$ is learned at the $t$th iteration with respect to the distribution $d_{t-1}$. Furthermore, $\text{sign}(h_t(x_n))$ predicts the class label for $x_n$, and $|h_t(x_n)|$ is the confidence factor in the prediction. $w_t$ is the weight for $h_t$, and $T$ is the number of weak classifiers. Given the distribution of the training samples $d_{t-1}$, the accuracy of the weak classifier $h_t$ is given by $\rho_t = \sum_{n=1}^N d_{t-1}^n \text{sign}(y_n h_t(x_n))$. For ease of exposition and avoid notation clutter, we introduce a vector $u_t$ and call it the “edge”. We let

$$u_t^n = \text{sign}(h_t(x_n)y_n), \quad (5-2)$$

i.e., $u_t$ measures the accuracy of the weak classifier $h_t$, $u_t^n \in \{-1, 1\}$, and $u_t = 1$ implies $h_t$ is a perfect classifier while $u_t = -1$ implies $h_t$ is a very poor classifier. Now we can rewrite the accuracy $\rho_t$ of $h_t$ as

$$\rho_t = \sum_{n=1}^N d_{t-1}^n u_t^n = d_{t-1} \cdot u_t. \quad (5-3)$$

During boosting, if the samples are linearly separable, we will maximize the hard margin between different classes. The hard margin is the width of the area separating the positive from the negative samples. By maximizing the hard margin, the classification accuracy of the strong classifier on the testing dataset will probably be
maximized. The LPBoost formulation to maximize the hard margin at iteration $t$ is given by,

$$
\max_{\mathbf{w}, \rho} \rho \\
\text{s.t.} \sum_{i=1}^{t} u_{n_i}^{t} \omega_i \geq \rho, \ n = 1, \cdots, N, \quad (5-4)
$$

where, $\rho$ is the minimum hard margin between the two classes. $\Delta_t$ is the $t$-simplex and $\mathbf{w} \in \Delta_t$ implies that $\sum_{j=1}^{t} w_j = 1$ and $w_j \geq 0$, for $j = 1, \cdots, t$. The effect of constraining $\mathbf{w} \in \Delta_t$ is to prevent $\mathbf{w}$ from scaling. In other words, if we remove the constraint $\mathbf{w} \in \Delta_t$, then $\mathbf{w}$ will scale to $\infty$ making $\rho$ tend to $\infty$.

When the samples can not be linearly separated, we can not find a hard margin that clearly separates the positive from the negative classes. In this case, soft boosting is an alternative choice, where we maximize the soft margin at each iteration and allow samples to fall below the margin. Samples can fall below the margin up to some slack factors, but we can levy a penalty for falling below the margin to make sure the slack factors don’t become extremely large. LPBoost to maximize the soft margin can be expressed by the following function,

$$
\max_{\mathbf{w}, \rho, \zeta} \rho - D \sum_{n=1}^{N} \zeta_n \\
\text{s.t.} \sum_{i=1}^{t} u_{n_i}^{t} w_i \geq \rho - \zeta_n, \ n = 1, \cdots, N, \quad (5-5)
$$

where, $\zeta$ is the slack variable vector, and $D$ is the constant factor which penalizes the slack variables. If $D$ is very large, say $\infty$, then $(5-5)$ becomes the hard margin maximization problem $(5-4)$; if $D$ is small enough, then it will always leads to feasible solutions for $(5-5)$. 

64
The dual problem of (5–5) at step $t$ minimizes the maximum accuracy resulting from the weak classifiers learned thus far, and is given by,

$$\min_{d \in \Delta_N, d \leq D1} \max_{i=1,\ldots,t} u^i \cdot d$$

(5–6)

Note that adding the weight $D$ to the slack variables of the primal (5–5) results in $d \leq D1$. To make such a $d$ exist, we should require $D \geq 1/N$. It was shown in [131] that $D = 1/s$, and $s \in \{1, \ldots, N\}$ is a favorable choice. LPBoost works well and has been extensively used, however, it converges very slowly, and the number of iterations for the algorithm is at best $O(\log N)$ with a large constant factor. When $N$ is large, it will be computationally expensive to use LPBoost. Also, the evolution of $d$ might have serious instabilities, which reduces the efficiency significantly and might result in overfitting. To overcome these downsides, we add a regularization term based on the total Kullback-Leibler (tKL) divergence to (5–6). This regularization makes the evolution of $d$ smooth, and more interestingly the number of iterations will be reduced to a constant. At the same time, the soft margin can be maximized without loss. The regularized LPBoost (tBRLPBoost) is

$$\min_{d \in \Delta_N, d \leq D1} ( \max_{i=1,\ldots,t} u^i \cdot d + \lambda \delta(d, d^0))$$

(5–7)

$\delta(., .)$ is the tKL and

$$\delta(d, d^0) = \left( \sum_{n=1}^{N} d_n \log \frac{d_n}{d_0^n} \right) / \sqrt{1 + \sum_{n=1}^{N} d_0^n (1 + \log d_0^n)^2}$$

(5–8)

where, $d^0$ is the initialized distribution, which is set to a uniform distribution, i.e., $d_0^n = 1/N, n = 1, \ldots, N$. $\lambda > 0$ is the regularization parameter for the tKL compared with the maximum accuracy. When $\lambda = 0$, the regularization term will vanish, and (5–7) will become (5–5) maximizing the soft margin of LPBoost.
At each iteration, tBRLPBoost will compute the weak classifier \( h_t \), along with its corresponding weight \( w_t \) in building the strong classifier, and based on the performance of the weaker classifier, tBRLPBoost will update the distribution of the training samples. The distribution is updated from \( d^{t-1} \) to \( d^t \). The algorithm for tBRLPBoost is depicted in Algorithm 3.

### Algorithm 3 Total Bregman divergence regularized LPBoost

**Input:** \( \{(x_n, y_n)\}_{n=1}^N \), \( x_n \in \mathcal{X} \), and \( y_n \in \{1, -1\} \).

**Output:** \( H(x) = \text{sign}(\sum_{t=1}^T w_t h_t(x)) \), \( w_t > 0 \), \( \{h_t(x)\}_{t=1}^T \) are the weak classifiers to be learned, \( T \) is the number of iterations/weak classifiers.

**Initialization:** \( d_0^n = 1/N, n = 1, \ldots, N \)

**for** \( t = 1 \) to \( T \) **do**

- (Find the weak classifier)
  \( h_t \leftarrow \arg \max_h \sum_{n=1}^N d_{n}^{t-1} \text{sign}(y_n h(x_n)) \).

- Update the distribution from \( d^{t-1} \) to \( d^t \) according to (5–7) and the weight \( w \) for the weak classifiers according to (5–5)

**end for**

Return \( H(x) = \text{sign}(\sum_{t=1}^T w_t h_t(x)) \)

To make the boosting algorithm evolve to a good strong classifier more quickly, some researchers impose the constraint that the weak classifiers \( \{h_t\}_{t=1}^T \) should be linearly independent to remove redundancy, i.e., \( h_t \cdot h_i = 0, i = 1, \ldots, t - 1 \). This will guarantee that the redundancy factor is the least among the weak classifiers and thus leading to high efficiency. However, this is rather complicated to implement, and in addition, our tBRLPBoost is already able to achieve very high efficiency, so we do not impose this condition.

#### 5.3.1 Computation of \( d^t \) and \( w \)

To directly compute \( d^t \) from (5–7) is complicated, instead, we will first get the Lagrangian dual of (5–7) and accordingly compute \( d^t \). To find the Lagrangian dual, we
rewrite (5–7) into the following form

\[
\min_{\beta, d} \beta + \lambda \delta(d, d^0)
\]

s.t. \( u^i \cdot d \leq \beta, \ i = 1, \ldots, t \)  
\( d \in \Delta_N, \)  
\( d \leq D1. \)  

(5–9)

Now, the Lagrangian \( \Phi \) of (5–7) is easy to see, and given by,

\[
\Phi(d, \beta, w, \xi, \gamma) = \beta + \lambda \delta(d, d^0) + \sum_{i=1}^{t} w_i (u^i \cdot d - \beta) 
\]

\[
+ \sum_{n=1}^{N} \xi_n (d_n - D) + \gamma (d \cdot 1 - 1). 
\]

(5–10)

where, \( w_i, i = 1, \ldots, t, \xi_n, n = 1, \ldots, N \) and \( \gamma \) are non-negative regularizers.

Differentiating \( \Phi \) with respect to \( \beta \), we get

\[
\frac{\partial \Phi}{\partial \beta} = 1 - \sum_{i=1}^{t} w_i = 0, \quad (5–11)
\]

and by enforcing \( \sum_{i=1}^{t} w_i = 1 \) manually (done by normalizing \( w \)), we can remove \( \beta \) from the Lagrangian dual function (5–10). Also since,

\[
\frac{\partial \Phi}{\partial \gamma} = d \cdot 1 - 1 = 0, \quad (5–12)
\]

by enforcing \( d \cdot 1 = 1 \), we can eliminate \( \gamma \) from (5–10). Moreover, according to the KKT condition [17], \( \xi_n (d_n - D) = 0 \). Therefore, we can simplify (5–10) and get the partial Lagragian

\[
\Phi(d, w) = \lambda \delta(d, d^0) + \sum_{i=1}^{t} w_i u^i \cdot d. 
\]

(5–13)

Now differentiating \( \Phi \) with respect to \( d \), setting it to 0, and normalizing \( d \), we can get,

\[
d^t_n = \frac{d^0_n \exp \left( -c \sum_{i=1}^{t} u^i_n w_i \right)}{Z_t}, \quad (5–14)
\]
where, \( c = \frac{1}{\lambda} \sqrt{1 + \sum_{n=1}^{N} d_n^0 (1 + \log d_n^0)^2} \), and \( Z_t \) is the normalization parameter to make \( \sum_{n=1}^{N} d_n^t = 1 \).

The weight vector \( w \) for the weak classifiers should satisfy the linear programming problem \((5–5)\), and this can be solved using column generation \([29]\) or a gradient based method. We used the delayed column generation technique to solve \( w \). Delayed column generation \([29]\) is an efficient algorithm to solve linear programming problems. It utilizes the divide and conquer strategy by splitting the goal problem into smaller problems. By solving the smaller problems through their duals, it finds the solution to the bigger problem. For a detailed explanation about delayed column generation the reader is referred to \([29]\).

5.3.2 Bounding the Number of Iterations

Let the tolerance be \( \epsilon \), i.e., we allow the maximum soft margin resulting from \( \text{tBRLPBoost} \) to be different from the optimal soft margin by \( \epsilon \). If the regularization parameter \( \lambda \) in \((5–7)\) is set to \( \frac{\epsilon \sqrt{1+(\log N-1)^2}}{2 \log(ND)} \), then the number of iterations for \( \text{tBRLPBoost} \) is at most \( b = 32 \frac{\sqrt{2}}{\epsilon^2} + 1 \).

**Theorem 5.1.** Let \( \lambda \) in \((5–7)\) be \( \frac{\epsilon \sqrt{1+(\log N-1)^2}}{2 \log(ND)} \), then the \( \text{tBRLPBoost} \) will converge in constant \( b \)’ number of iterations given by \( b = 32 \frac{\sqrt{2}}{\epsilon^2} + 1 \).

**Proof.** Since the initialized distribution \( d^0 \) is set to \( d_n^0 = 1/N, n = 1, \cdots, N \), therefore, \( \sqrt{1 + \sum_{n=1}^{N} d_n^0 (1 + \log d_n^0)^2} = \sqrt{1 + (\log N - 1)^2} \). Also according to \((5–7)\), \( d_n \leq D, n = 1, \cdots, N \), thus

\[
\delta(d, d^0) \leq \frac{\log(ND)}{\sqrt{1 + (\log N - 1)^2}}. \tag{5–15}
\]

Consequently, if

\[
\lambda = \frac{\epsilon \sqrt{1 + (\log N - 1)^2}}{2 \log(ND)}, \tag{5–16}
\]

we have

\[
\lambda \delta(d, d^0) \leq \frac{\epsilon}{2}. \tag{5–17}
\]
This means that the difference between the optimal margin of the regularized problem (5–7) is at most $\epsilon$ different from the margin of the unregularized problem (5–5). Using similar proof as in [131], we can say that the algorithm will converge in

$$T \leq \frac{16}{\lambda \epsilon} + 1 = \frac{32 \log(ND)}{\epsilon^2 \sqrt{1 + \log N}} + 1.$$  

(5–18)

Since $\sqrt{1 + \log N} \geq \log N$, and $D \leq 1$, thus, setting $D = 1$, we get

$$T \leq 32 \sqrt{2}/\epsilon^2 + 1.$$  

(5–19)

From (5–7), $D = 1$ means there is no constraint on the distribution, in other words, $d$ can get to any point in the $N$-simplex, so the algorithm can achieve the optimum. 

5.3.3 Weak Classifiers

The type of weak classifiers is very important in determining the accuracy of the final strong classifier. In the boosting community, there are many different kinds of popular weak classifiers, such as decision stump, RIPPER [25], SLIPPER [26]. To emphasize the performance strength of tBRLPBoost, we use the simplest type of weak classifier, namely, the decision stump, which is a two level binary decision tree. An illustration of choosing a weak classifier using the decision stump is shown in Figure 5–1.

The weak classifiers are computed in the following way. We randomly select one feature from the feature vector, and put together the values of this feature for all training
samples, sort them, and get the mean of every two consequent values. These means and the smallest, as well as the largest values will constitute the candidate threshold set. Given a test sample, if the value of its feature is above the threshold, we will assign this sample to one class, otherwise assign it to the other class. For example, if we selected the kth feature of the training samples, and the values of all the training samples for the kth feature are \( \{x_1^k, x_2^k, \ldots, x_N^k\} \). We sort \( \{x_1^k, x_2^k, \ldots, x_N^k\} \) in non-decreasing order, and if the sorted values are denoted by \( \{\hat{x}_1^k, \hat{x}_2^k, \ldots, \hat{x}_N^k\} \), then the candidate threshold set is \( \{\hat{x}_1^k, (\hat{x}_1^k + \hat{x}_2^k)/2, \ldots, (\hat{x}_{N-1}^k + \hat{x}_N^k)/2, \hat{x}_N^k\} \). We will choose the threshold which gives us the highest classification accuracy and take it as our weak classifier.

5.4 Experiments

We evaluated our algorithm on numerous public domain datasets, including the UCI machine learning repository [43], the OASIS MRI brain database [74], the Epilepsy dataset [59], the CNS dataset [99], the Colon tumor dataset [3] and Leukemia cancer dataset [50]. We compared our method with several state-of-the-art boosting techniques and classifiers.

5.4.1 UCI Datasets

The UCI repository [43] is a collection of databases that have been extensively used for analyzing machine learning techniques. The repository contains very noisy data (e.g. waveform) as well as relatively clean data, which is optimal for testing classification algorithms. We selected 13 datasets from the UCI repository. The selected datasets include noisy and clean datasets, cover small size to large size
datasets in terms of number of samples in the datasets, and range from low dimension to high dimension in terms of number of attributes per sample of the datasets. The description of the selected datasets is shown in Table 5-1. We compared our results with those generated from other techniques in the literature, including AdaBoost, LPBoost, SoftBoost, BrownBoost and ELPBoost. For the implementation of most methods, we used the existing codes on the website (http://sourceforge.net/projects/jboost/, http://www.kyb.mpg.de/bs/people/nowozin/gboost/#pubs), and for ELPBoost, we adapted it from LPBoost (http://www.kyb.mpg.de/bs/people/nowozin/gboost/#pubs). We use the same experimental settings as in [131]. Each dataset is separated into 100 predefined splits as training and testing sets. For each of the splits, we use 5-fold cross-validation, i.e. 80% of the samples are for training and validation, and 20% of the samples are for testing. We determine the parameters for each of the boosting algorithms during the training and validation period, and the parameters are set to be those maximizing the classification accuracy of the training samples. The results obtained are an average taken over 100 runs of the estimation of the classification accuracy for each algorithm and dataset. The means and standard deviations are reported in Table 5-2, from which it is evident that tBRLPBoost has up to 13% higher accuracy than other boosting algorithms on various kinds of datasets.

We also report the change of the classification accuracy as a function of the number of iterations on the pima, spambase, iris and spectf datasets, and compare our results with the results from an application of ELPBoost [131]. The results are obtained using 5-fold cross-validation and we show the average classification accuracy vs. the number of iterations in Figure 5-2. The figure shows that the accuracy of tBRLPBoost increases much faster during the first few iterations and gets to satisfactory accuracy in far less number of iterations in comparison to ELPBoost.
Table 5-1. Description of the UCI datasets that we use.

<table>
<thead>
<tr>
<th>dataset</th>
<th>% instance</th>
<th>% attribute</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>breast cancer</td>
<td>699</td>
<td>9</td>
<td>b-cancer diagnosis</td>
</tr>
<tr>
<td>diabetes</td>
<td>768</td>
<td>8</td>
<td>diabetes diagnosis</td>
</tr>
<tr>
<td>german credit</td>
<td>1000</td>
<td>24</td>
<td>good/bad credit</td>
</tr>
<tr>
<td>heart disease</td>
<td>303</td>
<td>74</td>
<td>heart diseases</td>
</tr>
<tr>
<td>ionosphere</td>
<td>351</td>
<td>33</td>
<td>classify radar data</td>
</tr>
<tr>
<td>liver disorders</td>
<td>345</td>
<td>6</td>
<td>blood test results</td>
</tr>
<tr>
<td>sonar</td>
<td>208</td>
<td>60</td>
<td>sonar signals</td>
</tr>
<tr>
<td>pendigits</td>
<td>1364</td>
<td>16</td>
<td>handwritten digits</td>
</tr>
<tr>
<td>waveform</td>
<td>5000</td>
<td>21</td>
<td>waveforms</td>
</tr>
<tr>
<td>pima</td>
<td>768</td>
<td>8</td>
<td>signs of diabetes</td>
</tr>
<tr>
<td>spambase</td>
<td>4601</td>
<td>51</td>
<td>classify spam email</td>
</tr>
<tr>
<td>iris</td>
<td>150</td>
<td>4</td>
<td>flower classification</td>
</tr>
<tr>
<td>spectf</td>
<td>267</td>
<td>44</td>
<td>(ab)normal heart</td>
</tr>
</tbody>
</table>

Table 5-2. Classification accuracy (mean ± deviation) for different boosting algorithms.

<table>
<thead>
<tr>
<th>dataset</th>
<th>AdaBoost</th>
<th>LPBoost</th>
<th>BrownBoost</th>
<th>ELPBoost</th>
<th>tBRLPBoost</th>
</tr>
</thead>
<tbody>
<tr>
<td>breast cancer</td>
<td>0.702 ± 0.039</td>
<td>0.734 ± 0.037</td>
<td>0.699 ± 0.037</td>
<td>0.728 ± 0.042</td>
<td>0.864 ± 0.008</td>
</tr>
<tr>
<td>diabetes</td>
<td>0.722 ± 0.015</td>
<td>0.765 ± 0.018</td>
<td>0.728 ± 0.014</td>
<td>0.756 ± 0.018</td>
<td>0.783 ± 0.015</td>
</tr>
<tr>
<td>german credit</td>
<td>0.730 ± 0.021</td>
<td>0.754 ± 0.023</td>
<td>0.752 ± 0.019</td>
<td>0.758 ± 0.223</td>
<td>0.805 ± 0.102</td>
</tr>
<tr>
<td>heart disease</td>
<td>0.798 ± 0.025</td>
<td>0.812 ± 0.033</td>
<td>0.793 ± 0.027</td>
<td>0.827 ± 0.031</td>
<td>0.895 ± 0.016</td>
</tr>
<tr>
<td>ionosphere</td>
<td>0.879 ± 0.016</td>
<td>0.843 ± 0.011</td>
<td>0.875 ± 0.015</td>
<td>0.838 ± 0.014</td>
<td>0.957 ± 0.006</td>
</tr>
<tr>
<td>liver disorders</td>
<td>0.753 ± 0.026</td>
<td>0.765 ± 0.032</td>
<td>0.802 ± 0.036</td>
<td>0.783 ± 0.027</td>
<td>0.875 ± 0.031</td>
</tr>
<tr>
<td>sonar</td>
<td>0.861 ± 0.053</td>
<td>0.857 ± 0.060</td>
<td>0.861 ± 0.042</td>
<td>0.873 ± 0.071</td>
<td>0.902 ± 0.018</td>
</tr>
<tr>
<td>pendigits</td>
<td>0.921 ± 0.010</td>
<td>0.942 ± 0.012</td>
<td>0.903 ± 0.012</td>
<td>0.921 ± 0.008</td>
<td>0.985 ± 0.003</td>
</tr>
<tr>
<td>waveform</td>
<td>0.892 ± 0.004</td>
<td>0.899 ± 0.005</td>
<td>0.900 ± 0.004</td>
<td>0.895 ± 0.006</td>
<td>0.923 ± 0.003</td>
</tr>
</tbody>
</table>

5.4.2 OASIS Datasets

We also evaluated our algorithm on the OASIS MRI brain database [74]. The OASIS database contains a cross-sectional collection of 416 subjects aged 18 to 96. Out of the 416 subjects, 175 subjects are younger than 40 that we designated as young (Y), and 195 subjects are above 60 that we designated as old (O). The other 66 subjects are designated as middle aged (M). Each subject is represented using a 3D histogram describing the non-rigid registration required to co-register an emerging atlas (in a groupwise registration process) to a subject MR brain scan. This groupwise registration was accomplished by using the method described in [55] on the OASIS dataset. The
Figure 5-2. Classification using tBRLPBoost and ELPBoost on the training and testing sets of the pima, spambase, iris and spectf datasets.

The number of bins in each direction was set to \((6 \times 6 \times 6)\) for constructing the histograms of the displacement vectors describing the non-rigid registration. Further, among the old aged people, we took 70 subjects, and 35 of them were diagnosed with very mild to moderate Alzheimer disease (AD) while the rest were controls. We did four groups of experiments on this dataset. First classified the age groups (Y vs. M, M vs. O, and O vs. Y), and then classified the healthy and the very mild to moderate AD patients (AD vs. Control). For each experiment, we use 5-fold cross validation and report the average classification accuracy. The parameters of each of the algorithms are set to be those...
maximizing the accuracy of the training dataset. We compared our results with those from AdaBoost, LPBoost and ELPBoost. The results are shown in Table 5-3, from which it is evident that our method (tBRLPBoost) outperforms the competitive methods. Also, tBRLPBoost converges much faster than the other algorithms. All algorithms are run on a laptop with Intel(R) Core(TM)2 CPU L7500 @1.6GHz, 4GB memory, GNU Linux and MATLAB (Version 2010a). The average CPU time taken to converge for our algorithm is 0.2778s, while AdaBoost takes 1.8082s, LPBoost takes 2.1430s and ELPBoost takes 2.1164s.

5.4.3 Other Datasets

The Epilepsy dataset [59] consists of 3D histograms $(6 \times 6 \times 6)$ of displacement vector fields representing the registration deformation field between the left and right hippocampi in 3D. The goal is to distinguish between left and right anterior temporal lobe (L/RATL) epileptics. The CNS dataset [99] contains the treatment outcomes of 60 subjects for central nervous system embryonal tumor, which includes 21 survivors and 39 failures. The Colon tumor dataset [3] consists of 22 normal and 40 tumor colon tissue features. The Leukemia cancer dataset [50] contains 25 acute myeloid leukemia (AML) and 47 acute lymphoblastic leukemia (ALL) samples. The goal is to distinguish from AML to ALL. We compared our method with the recently published competitive classifiers, including conic section classifier (CSC) [9], CSC with margin pursuit (CSC-M) [59], kernel Fisher Discriminants (KFD) [39, 77], and kernel SVMs (SVM) [122]. The results are shown in Table 5-4, which depicts that our algorithm performs better than the competing classifiers.

<table>
<thead>
<tr>
<th>dataset</th>
<th>AdaBoost</th>
<th>LPBoost</th>
<th>ELPBoost</th>
<th>tBRLPBoost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y vs. M</td>
<td>0.960</td>
<td>0.980</td>
<td>0.973</td>
<td>0.985</td>
</tr>
<tr>
<td>M vs. O</td>
<td>0.962</td>
<td>0.972</td>
<td>0.974</td>
<td>0.996</td>
</tr>
<tr>
<td>O vs. Y</td>
<td>0.987</td>
<td>0.988</td>
<td>0.988</td>
<td>1.000</td>
</tr>
<tr>
<td>AD vs. Con.</td>
<td>0.923</td>
<td>0.967</td>
<td>0.968</td>
<td>0.977</td>
</tr>
</tbody>
</table>

Table 5-3. Classification accuracy of different methods on the OASIS dataset.
<table>
<thead>
<tr>
<th>dataset</th>
<th>CSC-M</th>
<th>CSC</th>
<th>KFD</th>
<th>SVM</th>
<th>tBRLPBoost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epilepsy</td>
<td>0.932</td>
<td>0.886</td>
<td>0.864</td>
<td>0.864</td>
<td>0.941</td>
</tr>
<tr>
<td>CNS</td>
<td>0.700</td>
<td>0.733</td>
<td>0.650</td>
<td>0.683</td>
<td>0.757</td>
</tr>
<tr>
<td>Colon</td>
<td>0.871</td>
<td>0.871</td>
<td>0.758</td>
<td>0.823</td>
<td>0.892</td>
</tr>
<tr>
<td>Leukemia</td>
<td>0.972</td>
<td>0.986</td>
<td>0.986</td>
<td>0.972</td>
<td>0.990</td>
</tr>
</tbody>
</table>

Table 5-4. Classification accuracy for different methods on the Epilepsy, CNS, Colon tumor and Leukemia datasets.

### 5.5 Discussions

In this chapter, we proposed a new boosting algorithm dubbed tBRLPBoost, which is total Bregman divergence (tBD) regularized LPBoost. tBRLPBoost is robust to noise and outliers due to the intrinsic robustness property of tBD. tBRLPBoost is able to maximize the soft margin for linearly inseparable dataset. We showed that tBRLPBoost requires only a constant number of iterations to converge and hence is independent of the size of the training set, which makes it very efficient. In comparison, most efficient boosting algorithm in the published literature cost logarithmic number of iterations. We showed several comparisons to other existing boosting methods and depicted better performance of our algorithm in comparison to the state-of-the-art methods in literature. Even though in this chapter we focus on tKL regularized LPBoost, we can easily extend this and use other classes of tBD to regularize the LPBoost. Further, since tBRLPBoost is highly efficient, this makes it a promising candidate for online boosting applied to very large scale datasets, dynamic environments and multimedia datasets, which will be the focus of our future work. Finally, tBRLPBoost can be very easily generalized to multiclass input as well to the confidence-related input, where the label is not binary but takes values in the interval $[-1, 1]$. 
Chapter 6
Simultaneous Smoothing and Estimation of DTI via Robust Variational Non-Local Means

Regularized diffusion tensor estimation is an essential step in DTI analysis. There are many methods proposed in literature for this task but most of them are neither statistically robust nor feature preserving denoising techniques that can simultaneously estimate symmetric positive definite (SPD) diffusion tensors from diffusion MRI. One of the most popular techniques in recent times for feature preserving scalar-valued image denoising is the non-local means filtering method that has recently been generalized to the case of diffusion MRI denoising. However, these techniques denoise the multi-gradient volumes first and then estimate the tensors rather than achieving it simultaneously in a unified approach. Moreover, some of them do not guarantee the positive definiteness of the estimated diffusion tensors. In this chapter, we propose a novel and robust variational framework for the simultaneous smoothing and estimation of diffusion tensors from diffusion MRI. Our variational principle makes use of a recently introduced total Kullback-Leibler (tKL) divergence, which is a statistically robust similarity measure between diffusion tensors, weighted by a non-local factor adapted from the traditional non-local means filters. For the data fidelity, we use the nonlinear least-squares term derived from the Stejskal-Tanner model. We present experimental results depicting the positive performance of our method in comparison to competing methods on synthetic and real data examples.

6.1 Literature Review for DTI Estimation

Diffusion MRI is a technique that uses diffusion sensitizing gradients to non-invasively image anisotropic properties of tissue. Diffusion tensor imaging (DTI) introduced by Basser et al. [13], approximates the diffusivity function by a symmetric positive definite tensor of order two. There is abundant literature on DTI analysis including but not limited to denoising and tensor field estimation [22, 78, 100, 114, 119, 120, 124, 129], DTI
registration, fiber tractography etc and all of these latter tasks will benefit from denoising and estimation of smooth diffusion tensors.

In most of the existing methods, the diffusion tensors (DTs) are estimated using the raw diffusion weighted echo intensity image (DWI). At each voxel of the 3D image lattice, the diffusion signal intensity $S$ is related with its diffusion tensor $D \in \text{SPD}(3)^1$ via the Stejskal-Tanner equation \[112\]

$$S = S_0 \exp(-bg^T Dg),$$

(6–1)

where $S_0$ is the signal intensity without diffusion sensitizing gradient, $b$ is the diffusion weighting and $g$ is the direction of the diffusion sensitizing gradient.

Estimating the DTs from DWI is a challenging problem, since the DWI is normally corrupted with noise \[100, 114, 119\]. Therefore, a statistically robust DTI estimation method which is able to perform feature preserving denoising is desired. There are various methods \[14, 37, 51, 94, 103, 114, 119, 129\] that exist in the literature to achieve this goal of estimating $D$ from $S$. A very early one is direct tensor estimation \[133\]. Though time efficient, it is sensitive to noise because only 7 gradient directions are used to estimate $D$ and $S_0$. Another method is the minimum recovery error (MRE) estimation or least squares fitting \[13\] which minimizes the error when recovering the DTs from the DWI. MRE is better than direct estimation but it does not enforce spatial regularization or the SPD constraint resulting in possible inaccuracies.

Bearing these deficiencies in mind, researchers developed variational framework (VF) based estimation \[22, 120, 129\]. These approaches take into account the SPD constraint on the diffusion tensors. The smoothing in all these approaches involves some kind of weighted averaging over neighborhoods which define the smoothing

\[1\] $\text{SPD}(3)$ represents the space of $3 \times 3$ symmetric positive definite matrices
operators resulting from the variational principles. These smoothing operators are locally defined and do not capture global geometric structure present in the image.

More recently, some denoising frameworks have been proposed according to the statistical properties of the noise. They assume the noise follows the Rician distribution \cite{30, 58}, and the DWI is denoised using maximum likelihood estimation. After denoising the DWI, one can use other techniques to estimate the DTI. Besides, the popular NLM based method has been adapted by many to denoise DTI data sets \cite{27, 134, 135}. In the NLM based approaches, one first needs to denoise the DWI field and then estimate DTI from the denoised DWI. Alternatively, the diffusion tensors are first estimated and then denoised using Riemannian approach \cite{20} or the NLM framework incorporating a Log-Euclidean metric \cite{38}. The drawback of such two-stage processes is that the errors might accumulate from one stage to the other.

To overcome the aforementioned problems, we propose a novel statistically robust variational non-local approach for simultaneous smoothing and tensor estimation from the raw DWI data. This approach combines the VF, NLM and an intrinsically robust regularizer on the tensor field. The main contributions of this approach are three-fold. First, we use total Bregman divergence (specifically, the tKL) as a measure to regularize the tensor field. Combined with the Cholesky decomposition of the diffusion tensors, this automatically ensures the positive definiteness of the estimated diffusion tensors, which overcomes the common problem for many techniques \cite{129} that need to manually force the tensor to be positive definite. Second, it preserves the structure of the tensor field while denoising via an adaptation of the NLM framework. Finally, it allows for simultaneous denoising and DT estimation, avoiding the error propagation of a two stage approach described earlier. Besides, this method can be easily extended to higher order tensor estimation. We will explain these points at length in the rest of the chapter.
The rest of the chapter is organized as follows. In Section 6.2, we introduce our proposed method and explore its properties, followed by the empirical validation in Section 6.3. Finally we conclude in Section 6.4.

6.2 Proposed Method

The simultaneous denoising and estimation of the DTI is achieved by minimizing the following energy function:

$$\min_{S_0, D \in \text{SPD}} E(S_0, D) = \lambda \sum_{x \in \Omega} \sum_{i=1}^{n} (S_i(x) - S_0(x) \exp\{-b g_i^T D(x) g_i\})^2$$

$$+ (1 - \lambda) \sum_{x \in \Omega} \sum_{y \in V(x)} w(x, y) \left[(S_0(x) - S_0(y))^2 + \delta(D(x), D(y))\right],$$

(6–2)

where $\Omega$ is the domain of the image, $w(x, y)$ is the similarity between voxels $x$ and $y$, $V(x)$ is the user specified search window at $x$, and $\delta(D(x), D(y))$ is the total Kullback-Leibler (tKL) divergence proposed in [70, 123] between tensors $D(x)$ and $D(y)$. tKL is defined in [123] and has been used in DTI segmentation [123] and classification [69]. We will redefine it later in Section 6.2.2 for the sake of completeness. The first term of (6–2) minimizes the non-linear fitting error, the second term enforces smoothness constraints on $S_0$ and $D$ via a non-local means regularizer. $\lambda$ is the regularization constant balancing the fitting error and the smoothness. Note that $S_i, S_0$ and $D$ by default represent the values at voxel $x$, unless specified otherwise.

6.2.1 Computation of the Weight $w(x, y)$

The weight $w(x, y)$ is regularizes the similarity between $S_0(x)$ and $S_0(y)$, as well as $D(x)$ and $D(y)$. Usually, one requires the similarity to be consistent with the similarity between their corresponding diffusion signals. Therefore, we define $w(x, y)$ based on the diffusion signal intensities of the two voxels’ neighborhoods.

Let $N(x)$ and $N(y)$ denote the neighborhoods of $x$ and $y$ respectively. If $y \in V(x)$, then $w(x, y)$ is defined as,

$$w(x, y) = \frac{1}{Z(x)} \exp \left(-\|S(N(x)) - S(N(y))\|^2/h^2\right),$$

(6–3)
where \( h \) is the user specified filtering parameter and \( Z \) is the normalization constant. 

\[
S(N(y)) = \left( \sum_j^m \| S(\mu_j) - S(\nu_j) \|^2 \right)^{\frac{1}{2}},
\]

where \( \mu_j \) and \( \nu_j \) are the \( j \)th voxels in the neighborhoods respectively, and \( m \) is the number of voxels in each neighborhood.

From (6–2), we can see that when \( w(x, y) \) is large, then \( S_0(x) \) and \( S_0(y) \) as well as \( D(x) \) and \( D(y) \) respectively are similar. In other words, if the signal intensities for the neighborhoods of two voxels are similar, their corresponding \( Ds \) and \( S_0s \) should also be similar. Though having very good accuracy, NLM is known for its high time complexity. To reduce the time cost, we use two tricks. One is to decrease the number of computations by selecting only those voxels whose signal intensity is similar to that of the voxel under consideration. This is specified by

\[
w(x, y) = \begin{cases} 
\frac{1}{Z(x)} \exp \left( -\frac{\| S(N(x)) - S(N(y)) \|^2}{h^2} \right), & \text{if } \frac{\|S(N(x))\|^2}{\|S(N(y))\|^2} \in [\tau_1, \tau_2] \\
0, & \text{Otherwise}
\end{cases}
\]

We choose \( \tau_1 = 0.5 \) and \( \tau_2 = 2 \) in our experiments. This prefiltering process greatly decreases the number of computations.

The other trick is using parallel computing, where we divide the computations into several parts, and assign the computation parts to several processors. In our case, we divide the volumes into 8 subvolumes, and assign each subvolume to one processor, and a desktop with 8 processors is used. This multi-threading technique greatly enhances the efficiency.

6.2.2 Computation of the tKL Divergence

Motivated by earlier use of KL divergence as a similarity measure between DTs in literature [129], we use the recently introduced tKL [123] to measure the similarity between tensors. tKL has the property of being intrinsically robust to noise and outliers, yields a closed form formula for computing the median (an \( \ell_1 \)-norm average) for a set
of tensors, and is invariant to special linear group transformations (denoted as SL($n$)) – transformations that have determinant one [123].

Note that order-2 SPD tensors can be seen as covariance matrices of zero mean Gaussian probability density functions (pdf) [129]. Let $P, Q \in \text{SPD}(I)$, then their corresponding pdf are

$$p(t; P) = \frac{1}{\sqrt{(2\pi)^l \det P}} \exp \left( -\frac{t^TP^{-1}t}{2} \right),$$

$$q(t; Q) = \frac{1}{\sqrt{(2\pi)^l \det Q}} \exp \left( -\frac{t^TQ^{-1}t}{2} \right),$$

and the tKL between them (in closed form) is given by,

$$\delta(P, Q) = \frac{\int p \log \frac{p}{q} dt}{\sqrt{1 + \int (1 + \log q)^2 q dt}} = \frac{\log(\det(P^{-1}Q)) + tr(Q^{-1}P) - m}{2\sqrt{c_1 + \frac{(\log(\det Q))^2}{4} - c_2 \log(\det Q)},}$$

where $c_1 = \frac{3l}{4} + \frac{\log 2\pi}{2} + \frac{9(\log 2\pi)^2}{4}$ and $c_2 = \frac{(1+\log 2\pi)}{4}$.

Moreover, the minimization of the third term in (6–2)

$$\min_{D(x)} \sum_{x \in \Omega} \sum_{y \in V(x)} \delta(D(x), D(y))$$

leads to an $\ell_1$-norm average which was shown to have closed form expression [123]. In [123], this was called the t-center and was shown to be invariant to transformations from the $SL(n)$ group, i.e., $\delta(P, Q) = \delta(A^TPA, A^TQA)$, where $\det A = 1$.

Furthermore, given a set of SPD tensors $\{Q_i\}_{i=1}^m$, its t-center $P^*$ is given by [123]

$$P^* = \arg \min_P \sum_{i=1}^m \delta(P, Q_i),$$

and $P^*$ is explicitly expressed as

$$P^* = \left( \sum_i \frac{a_i}{M} Q_i^{-1} \right)^{-1}, a_i = \left( 2M \sqrt{c_1 + \frac{(\log(\det Q_i))^2}{4} - c_2 \log(\det Q_i)} \right)^{-1}. \quad (6-5)$$

The t-center for a set of DTs is the weighted harmonic mean, which is in closed form. Moreover, the weight is invariant to $SL(n)$ transformations, i.e., $a_i(Q_i) = a_i(A^TQ_iA), \forall A \in \text{SL}(n)$ for all $i$. 

81
SL(n). The $t$-center after the transformation becomes $\hat{P}^* = (\sum a_i (A^T Q_i A)^{-1})^{-1} = A^T P^* A$. This means that if $\{Q_i\}_{i=1}^m$ are transformed by some member of $SL(n)$, the $t$-center will undergo the same transformation. It is also found that the $t$-center is statistically robust to noise in that the weight is smaller if the tensor has more noise [123].

6.2.3 The SPD Constraint

We now show how to guarantee the positive definiteness constraint on the diffusion tensors to be estimated from the DWI data. It is known that if a matrix $D \in SPD$, their exists a unique lower diagonal matrix $L$ with its diagonal values all positive, and $D = LL^T$ [49]. This is the Cholesky factorization theorem. Many researchers [129] have used Cholesky factorization to ensure the positive definiteness. They first compute $L$, enforcing the diagonal values of $L$ to be positive, and consequently, $LL^T$ will be positive. Unlike this technique, we use Cholesky decomposition and tKL divergence to regularize the tensor field, and this automatically ensures the diagonal values of $L$ to be positive. The points can be validated as follows.

Substituting $D = LL^T$ into (6–4), we get

$$\delta(L(x), L(y)) = \frac{\sum_{i=1}^{3} (\log L_{ii}(y) - \log L_{ii}(x)) + tr(L^{-T}(y)L^{-1}(y)L(x)L^T(x)) - 1.5}{\sqrt{c_1 + (\sum_{i=1}^{3} \log L_{ii}(y))^2} - c_2 \sum_{i=1}^{3} \log L_{ii}(y)}.$$ 

(6–6)

Because of using the log computation, Eq. (6–6) automatically ensures $L_{ii}$s to be positive, therefore we do not need to add the SPD constraint manually.

6.2.4 Numerical Solution

In this section, we present the numerical solution to the variational principle (6–2).

The partial derivative equations of (6–2) with respect to $S_0$ and $L$ can be computed
explicitly and are,

\[
\frac{\partial E}{\partial S_0(x)} = -2\lambda \sum_{i=1}^{n} (S_i - S_0 \exp\{-bg_i^T LL^T g_i\}) \exp\{-bg_i^T LL^T g_i\} \\
- 2(1 - \lambda) \sum_{y \in V(x)} w(x, y)(S_0(x) - S_0(y)).
\]

\[
\frac{\partial E}{\partial L(x)} = 4\lambda \sum_{i=1}^{n} (S_i - S_0 \exp\{-bg_i^T LL^T g_i\}) S_0 \exp\{-bg_i^T LL^T g_i\} bL^T g_i g_i^T \\
- 2(1 - \lambda) \sum_{y \in V(x)} \frac{w(x, y)(L^{-1}(x) - L^T(x)L^{-T}(y)L^{-1}(y))}{c_1 + \frac{(\sum_{i=1}^{n} \log L_{ii}(y))^2}{4} - c_2 \sum_{i=1}^{n} \log L_{ii}(y)}.
\]

(6–7)

To solve (6–7), we use the limited memory quasi-Newton method described in [86]. This method is useful for solving large problems with a lot of variables, as is in our case. This method maintains simple and compact approximations of Hessian matrices making them require, as the name suggests, modest storage, besides yielding linear rate of convergence. Specifically, we use L-BFGS [86] to construct the Hessian approximation.

### 6.3 Experimental Results

We evaluate our method on both synthetic datasets with various levels of noise, and on real datasets. We compared our method with other state-of-the-art techniques including the techniques VF [120], NLMt [135] and NLM [135] respectively. We also present the MRE method for comparison since several software packages in vogue use this technique due to its simplicity. We implemented VF and NLMt by ourselves since we did not find any open source versions on the web. For the NLM, we used existing code \(^2\) for DWI denoising and used our own implementation of the least squares fitting to estimate DTI from the denoised DWI. To ensure fairness, we tuned all the parameters of each method for every experiment, and chose the set of parameters yielding the best results. The visual and numerical results show that our method yields better results than competing methods.

\(^2\) https://www.irisa.fr/visages/benchmarks/
6.3.1 DTI Estimation on Synthetic Datasets

The synthetic data is a $16 \times 16$ tensor field with two homogeneous regions as shown in Figure 6-1A. We let $S_0 = 5$, $b = 1500s/mm^2$, and $g$ be 22 uniformly-spaced directions on the unit sphere starting from $(1, 0, 0)$. Substituting the DTs, $S_0$, $b$, $g$ into the Stejskal-Tanner equation, we generate a $16 \times 16 \times 22$ DWI $S$. One representative slice of $S$ is shown in Figure 6-1B. Then following the method proposed in [58], we add Rician noise to $S$ and get $\hat{S}$, using the formula, $\hat{S}(x) = \sqrt{(S(x) + n_r)^2 + n_i^2}$, where $n_r$ and $n_i$ $\sim N(0, \sigma)$. Figure 6-1C shows the slice in Figure 6-1B after adding noise (SNR=10). By varying $\sigma$, we get different levels of noise and therefore a wide range of signal to noise ratio (SNR). The estimated DTI from using MRE, VF, NLMt, NLM, and the proposed method are shown in Figure 6-1. The figure shows that our method can estimate the tensor field more accurately.

To quantitatively evaluate the proposed model, we compared the average of the angle difference $\epsilon_\theta$ between the principle directions of the estimated tensor field and the ground truth tensor field, and the difference $\epsilon_{S_0}$ between the estimated and ground truth $S_0$. The results are shown in Table 6-1, from which it is evident that our method outperforms others and the significance in performance is more evident at higher noise levels. The average CPU time taken to converge for our method on a desktop computer with Intel 8 Core 2.8GHz, 24GB of memory, GNU Linux and MATLAB (Version 2010a) is 7.03s, whereas, NLM requires 10.52s (note both methods are executed using parallel computing).

6.3.2 DTI Estimation on Real Datasets

We also did DTI estimation on a $124 \times 96 \times 40$ 3D rat spinal cord DWI. The data was acquired using a PGSE technique with TR=1.5s, TE=28.3ms, bandwidth=35Khz, 22 diffusion weighted images with $b$-value about $1014s/mm^2$. The estimated tensor field is shown in Figure 6-2. We compared our method with all aforementioned methods, however due to space limit, we only present the results of MRE and NLM. From these
Figure 6-1. Ground truth synthetic DTI field, the original DWI, the Rician noise affected DWI, estimation using MRE, VF, NLMt, NLM, and the proposed method.

Table 6-1. Error in estimated DTI and $S_0$, using different methods, from synthetic DWI with different levels of noise.

<table>
<thead>
<tr>
<th>SNR</th>
<th>error</th>
<th>MRE</th>
<th>VF</th>
<th>NLMt</th>
<th>NLM</th>
<th>proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>$\epsilon_\theta$</td>
<td>20.1 ± 12.1</td>
<td>8.4 ± 10.2</td>
<td>8.9 ± 10.7</td>
<td>6.0 ± 10.1</td>
<td>5.8 ± 7.3</td>
</tr>
<tr>
<td></td>
<td>$\epsilon_{S_0}$</td>
<td>0.54 ± 0.09</td>
<td>0.66 ± 0.05</td>
<td>0.64 ± 0.08</td>
<td>0.31 ± 0.04</td>
<td>0.28 ± 0.01</td>
</tr>
<tr>
<td>40</td>
<td>$\epsilon_\theta$</td>
<td>22.1 ± 12.5</td>
<td>12.1 ± 13.2</td>
<td>15.7 ± 14.2</td>
<td>7.2 ± 12.5</td>
<td>6.1 ± 8.6</td>
</tr>
<tr>
<td></td>
<td>$\epsilon_{S_0}$</td>
<td>0.75 ± 0.17</td>
<td>0.75 ± 0.31</td>
<td>0.94 ± 0.41</td>
<td>0.64 ± 0.30</td>
<td>0.53 ± 0.27</td>
</tr>
<tr>
<td>30</td>
<td>$\epsilon_\theta$</td>
<td>22.3 ± 12.9</td>
<td>19.5 ± 13.9</td>
<td>18.3 ± 14.7</td>
<td>7.6 ± 12.7</td>
<td>6.8 ± 9.7</td>
</tr>
<tr>
<td></td>
<td>$\epsilon_{S_0}$</td>
<td>2.24 ± 2.16</td>
<td>1.03 ± 1.22</td>
<td>1.03 ± 1.31</td>
<td>1.02 ± 1.21</td>
<td>0.81 ± 0.69</td>
</tr>
<tr>
<td>15</td>
<td>$\epsilon_\theta$</td>
<td>28.3 ± 17.1</td>
<td>27.2 ± 15.1</td>
<td>25.6 ± 16.2</td>
<td>14.7 ± 16.1</td>
<td>8.2 ± 10.3</td>
</tr>
<tr>
<td></td>
<td>$\epsilon_{S_0}$</td>
<td>3.81 ± 2.24</td>
<td>1.91 ± 2.02</td>
<td>1.86 ± 1.87</td>
<td>1.85 ± 1.77</td>
<td>1.02 ± 0.87</td>
</tr>
<tr>
<td>8</td>
<td>$\epsilon_\theta$</td>
<td>43.2 ± 23.4</td>
<td>32.9 ± 25.8</td>
<td>28.2 ± 20.6</td>
<td>20.2 ± 18.5</td>
<td>8.7 ± 11.0</td>
</tr>
<tr>
<td></td>
<td>$\epsilon_{S_0}$</td>
<td>5.29 ± 4.36</td>
<td>2.48 ± 2.72</td>
<td>2.29 ± 2.32</td>
<td>2.24 ± 2.19</td>
<td>1.09 ± 0.92</td>
</tr>
</tbody>
</table>

figures, we can see that our proposed method can estimate a smoother tensor field which preserves the structure much better compared with other methods.

We also did DTI estimation on a $100 \times 80 \times 32$ 3D rat brain DWI. The data was acquired using a Bruker scanner under the Spin Echo technique with TR=2s, TE=28ms, 52 diffusion weighted images with a $b$-value of $1334 s/mm^2$. The FA and the principal
Figure 6-2. The figures are the one slice of the estimated tensor fields using MRE, NLM, and the proposed method respectively.

eigenvectors of the estimated tensor field are shown in Figure 6-3, which showed the comparison of our method with MRE and NLM. The results illustrate that our proposed method can estimate the tensor field more accurately compared with others.

6.4 Discussions

We proposed a robust variational non-local means approach for simultaneous denoising and DTI estimation. The proposed method combines the variational framework, non-local means and an intrinsically robust smoothness constraint. In the variational principle, we used non-linear diffusion tensor fitting term, along with a combination of non-local means and the tKL based smoothness for denoising. To speed up the NLM method, we prefiltered the voxels in the search window to reduce the number of computations and made use of parallel computing to decrease the computational load. This variational non-local approach was validated with synthetic and real data and shown to be more accurate than competing methods in the literature. We did not however compare with many methods in literature that were already compared to the NLM technique in [135]. For future work, we plan to develop a GPU-based implementation to better the computation time. We will also explore other quantitative measures of validation such as method noise defined for tensors. After getting a more
Figure 6-3. The FA of the estimated tensor fields using MRE, NLM, the proposed method, and the principal eigenvectors of the ROIs.

A comprehensive tensor estimation technique, we will utilize it as a preprocessing step for the applications of fiber tracking and DTI segmentation.
In this thesis, we developed a novel robust divergence dubbed the \textit{total Bregman divergence (TBD)}, which is intrinsically robust to noise and outliers. The goal here was not simply to develop yet another divergence measure but to develop an intrinsically robust divergence. This was achieved via alteration of the basic notion of divergence which measures the ordinate distance between its convex generator and its tangent approximation to the orthogonal distance between the same. This basic idea parallels the relationship between least squares and total least squares but the implications are far more significant in the field of information geometry and its applications since the entire class of the well known Bregman divergences have been redefined and some of their theoretical properties studied. Specifically, we derived an explicit form formula for the $t$-center which is the TBD-based median, that is robust to outliers. In the case of SPD tensors, the $t$-center was shown to be $SL(n)$ transformation invariant.

The robustness property of TBD was demonstrated here via applications to SPD tensor field segmentation, shape retrieval, regularized boosting, and DTI estimation. The results favorably demonstrate the competitiveness of our newly defined divergence in comparison to existing methods not only in terms of robustness, but also in terms of computational efficiency and accuracy as well.

However, the story is not yet complete and further investigations are currently underway. Our future research will focus on applications of TBD to fiber tracking and fiber clustering.
REFERENCES


BIOGRAPHICAL SKETCH

Meizhu Liu was born in Kaifeng, Henan, P. R. China. She received her Bachelor of Science degree from the Computer Science Department at the University of Science and Technology of China, P. R. China, in 2007. She earned her Master of Science degree from Department of Computer and Information Science and Engineering at the University of Florida, Gainesville, in May 2011. She will receive her Doctor of Philosophy degree from Department of Computer and Information Science and Engineering at the University of Florida, Gainesville, in December 2011. Her research interests include computer vision, machine learning, image processing and medical image analysis, information theory, and computational geometry.