中国科学技术大学

University of Science and Technology of China

本科毕业论文

A Dissertation Submitted for the Bachelor's Degree

若干基于高性能计算平台的机器学习算法研究

Study of Several Machine Learning Algorithms Based on High Performance Computing Platform

姓　　名	陈 凯
B.S. Candidate	Kai Chen
导　　师	霍强 研究员
Supervisor	Dr. Qiang Huo

2011 年 6 月

June 2011

致　　谢

在本文即将完成之际，向所有关心、帮助和支持过我的人致以最诚挚的感谢。

首先感谢霍强老师，霍老师在百忙之中抽出时间对我进行了悉心指导，为我的工作提供了宝贵的建议，向我推荐了大量有益的书籍和资料，一步步教会我如何做研究。在微软亚洲研究院的短短七个月时间里，霍老师的言传身教让我获益匪浅，在这里再次表达我衷心的感谢。同时还要感谢鄢志杰师兄、高腾师兄和杜俊师兄，他们在工作上对我的指导让我不断进步；感谢孙雷师兄、张羽师兄和韩玮师兄，他们的帮助和建议让我少走了许多弯路，而他们对待研究的认真态度非常值得我学习。

感谢中科大 –微软亚洲研究院联合培养班和语音组的同学们，他们让我度过了开心而又充实的七个月，感谢所有为我们上过课的老师，他们的讲述开拓了我的视野，激发了我的兴趣，尤其要感谢邹欣老师，他对待工作的认真负责是我学习的榜样。

四年的大学转眼即过，科大给我留下了太多美好的回忆。感谢四年来所有教过我的老师，他们的教诲让我终生受益；感谢少年班学院和自动化系的同学，他们的陪伴让我的生活充满欢笑；感谢辩论队的师兄师姐师弟师妹还有各位同仁，他们的出现让我第一次对一个集体产生归属感；感谢四年来为我提供帮助的各位知名和不知名的师兄师姐，他们的无私让我免去了前进途中的彷徨。最后我要感谢科大，这个让我无悔挥洒青春的地方，“永远科大人” 会是我一生的信仰。

最后感谢我的父母和姐姐。

目　录

摘　要

当今时代，得益于信息技术的快速发展，研究人员可以使用越来越多的数据训练机器学习模型以达到理想效果。数据的增加对计算能力的要求相应提高，使用高性能计算机集群成为其中一个选择。在本文中，我使用由微软亚洲研究院员工搭建的高性能计算平台研究了若干机器学习算法。该平台为开发人员使用高性能计算机集群高效地处理海量数据文件提供了便利。在此平台上，我实现了 LBG 聚类算法和基于期望最大化算法的似然率最大化高斯混合模型训练算法，并利用二维空间数据和语音 PLP 特征数据对两种算法的性能进行了验证。

我还采用改进型 Quickprop 算法和 4 种不同的 Rprop 算法对基于样本分离边距和最小分类误差的多原型手写识别模型进行了优化。与改进型 Quickprop 相比，Rprop 是一种非常简单的优化算法。一系列实验表明，Rprop 也是一种鲁棒的算法，而经过其优化的模型在简体中文数据集上可以获得比改进型 Quickprop 更高的识别准确率。

关键字:　机器学习，高性能计算，LBG 聚类，期望最大化算法，高斯混合模型，样本分离边距，最小分类误差，Rprop

Abstract

Nowadays, thank to the development of information technology, more and more data is available for researchers to train machine learning models. In order to handle larger number of data, using high performance computer cluster becomes a good choice. In this dissertation, I use a high performance computing platform developed by MSRA to implement several machine learning algorithms. This platform allows developers to use high performance computer cluster to handle huge amount of data files more efficiently and conveniently. With its help, I implemented an LBG clustering algorithm and an EM algorithm for maximum likelihood training of Gaussian Mixture Model. After testing on 2 dimensional data and PLP features of speech, the efficiency of my implementations of these two algorithms are verified.

I also use modified Quickprop and four kinds of Rprop algorithms to optimize Minimum Classification Error and Sample Separation Margin based Multi-Prototype classifiers for handwritten character recognition. Compared to modified Quickprop, Rprop is simpler. After a series of experiments, Rprop is confirmed to be robust and can achieve higher recognition accuracy than modified Quickprop on a simplified Chinese handwritten character data set.

Keywords: Machine Learning, High Performance Computing, LBG clustering algorithm, Expectation Maximization algorithm, Gaussian Mixture Model, Sample Separation Margin, Minimum Classification Error, Rprop

第 1 章　绪论

1.1　研究背景及意义

机器学习是一门以计算机系统为平台，以数据为研究对象，结合了概率论、统计学、运筹学、计算机科学等领域的交叉学科，它的目的是构建模型并利用模型使计算机系统更加智能化，或使计算机的某些性能得到提高，恰如人工智能大师 H. A. Simon 对“学习”给出的定义：“如果一个系统能够通过执行某个过程改进它的性能，这就是学习”。机器学习是人工智能研究的重要领域，在语音识别、手写识别、计算机视觉、生物特征识别、搜索引擎、医学诊断、DNA 序列测序等领域都得到了广泛应用。

机器学习的特点决定了其对数据的大量需求，海量的训练数据是获得性能优异的模型的基础。近年来，随着信息技术特别是互联网的高速发展，人类社会的信息总量呈爆炸式增长。以中国为例，自 2006 年开始，中国的网页规模基本保持翻番增长，2010 年网页数量达到 600 亿个，年增长率 78.6%[1]，如图1.1所示。 而早在 2008 年的时候，谷歌索引的网页总数就已经达到一万亿[1]，

图 1.1: 2006-2010 年中国网页规模变化

信息爆炸之迅猛可见一斑。

信息总量的增长为机器学习的模型训练提供了海量数据，计算机硬件发展

[1]http://tech.sina.com.cn/i/2008-07-26/08552352750.shtml

带来的运算速度提升为处理大规模数据提供了有利条件，并行计算理论和技术的研究为充分发挥高性能计算机的运算能力提供了强大手段。不难看出，当下是机器学习研究的大好契机。而掌握使用高性能计算机处理海量数据的能力可以大大缩短程序运行时间，加速新技术的研究与开发，显著提高科研工作效率。本文的工作，就是在搭建好的高性能计算平台上实现若干机器学习算法，检验平台性能，并尝试新的模型训练算法。

1.2 实验平台简介

本文所有的实验均在基于高性能计算机集群的平台上进行。 该平台由微软

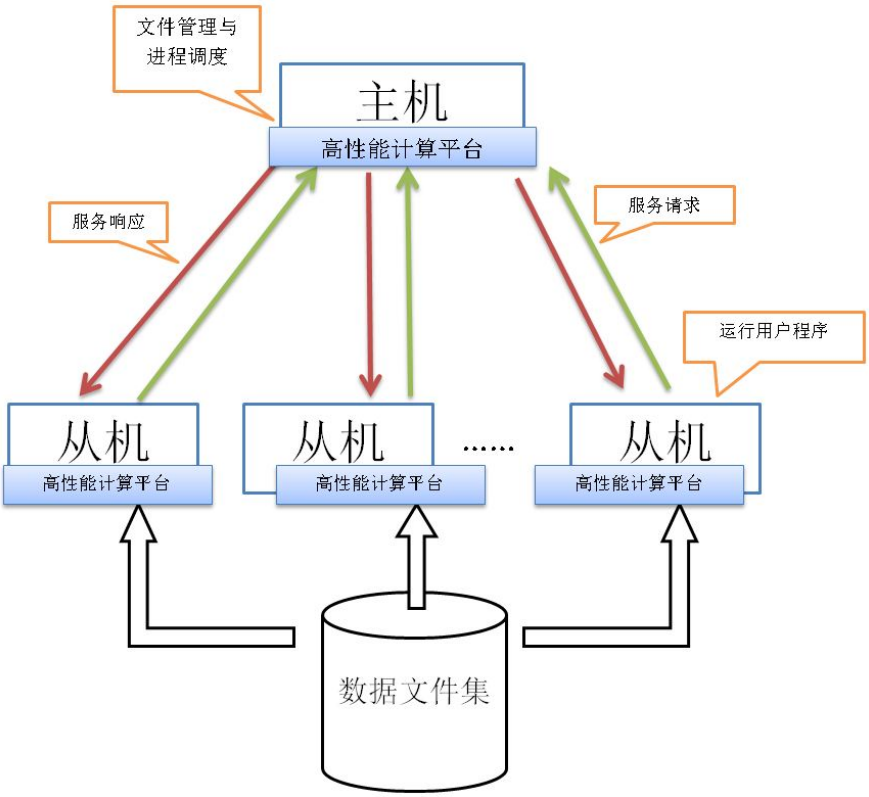

图 1.2: 高性能计算平台结构

亚洲研究院语音组的员工搭建，在高性能计算机集群上运行，与微软公司的高性能计算作业管理软件 HPC[2] Job Manager 配合使用。平台搭建的目的就是为

[2]High Performance Computing

了将用户从繁杂的任务管理工作中解放出来，集中注意于算法的研究，更好的发挥高性能计算机集群强大的运算能力。

平台的基本结构如图1.2所示。平台程序运行在 $N+1$ 台计算机上，其中 1 台为主机， N 台为从机。主机负责文件管理和进程调度，从机负责数据文件的处理，主机的管理有效地协调了各从机的工作，而从机在主机的控制之下，专注于对数据的复杂运算，两者分工名确，各司其职，有助于提高程序执行效率。用户程序在从机上运行，编程人员无需了解平台内部实现，只需调用平台提供的应用程序接口就能编写适于在该平台上运行的代码。代码编译通过之后，利用 HPC Job Manager 向高性能计算机集群提交任务。

利用高性能计算平台，用户可以方便地实现需要处理大量数据文件、可并行的机器学习算法。本文接下来要介绍的几种算法均属于这一类型。

1.3 主要工作概述

本文的工作主要包括如下三个方面：

1. 实现了文献 [2] 中的 LBG 聚类算法，并对大规模的语音 PLP 特征数据文件进行了处理，为后续工作打下基础。

2. 利用期望最大化算法实现了似然率最大化的高斯混合模型的训练。

3. 实现了基于样本分离边距和最小分类误差的多原型手写识别模型的训练算法，并使用多种算法对模型进行了优化。

上述三项工作的共同特点是都需要处理大量数据文件，且可以进行并行计算，因此适合在高性能计算平台上实现。

本文的内容组织如下：

- 第二章讲述 LBG 聚类算法的实现及实验结果，重点介绍在不同失真测度下，聚类中心的计算方法。

- 第三章介绍高斯混合模型的训练方法及在语音 PLP 特征数据文件上的训练结果，重点讲述基于期望最大化算法的似然率最大化高斯混合模型训练方法的推导过程。

- 第四章介绍基于样本分离边距和最小分类误差的多原型手写识别模型，重点介绍最小分类误差方法的训练框架并比较不同优化算法的优化效果。
- 第五章总结和展望，总结本文所做的工作，并对未来工作进行展望。

第 2 章　聚类算法

2.1　数据聚类概述

2.1.1　数据聚类

数据聚类（Data Clustering）是一种重要的非监督学习方法，它是分析静态数据的一门技术，通过一定的手段把对象集合分成不同的类别，使得属于同一类的成员对象彼此相似，不同类的成员彼此相异。与分类不同的是，聚类所要划分的类是未知的，这也是聚类被划入非监督学习的原因。

“数据聚类” 一词最早出现在一篇发表于 1954 年的处理人类学研究数据的文章标题中 [3]，它的发展要归因于多学科研究人员的共同努力，分类学家、社会学家、心理学家、生物学家、统计学家、数学家、工程师、计算机科学家等均为其发展贡献了力量。经过 50 多年的发展，数据聚类在机器学习、数据挖掘、模式识别、图像分析等领域都得到了广泛应用。

数据聚类算法主要有两类：层次聚类（hierarchical clustering）和划分聚类（partitional clustering）。层次聚类的主要思想是自上而下分解数据集或者自下而上合并数据集以达到聚类目的，而划分聚类则是直接将数据集聚成所要求的类数，没有上述的分解合并过程。

2.1.2　K-means 算法

在模式识别领域，最常用的是划分聚类方法，而划分聚类方法中最常用的是 50 多年前提出的 K-means 算法 [4]。本文将要用到的 LBG 算法就是一种类似于 K-means 的聚类算法。除此之外，还出现了大量的 K-means 算法的改进算法，比如 fuzzy C-means，bisecting K-means，X-means，K-medoid 等 [3]。为此有必要对 K-means 算法进行简单介绍。

假设有 D 维样本数据集 $\mathcal{X} = \{\boldsymbol{x}_n \in \mathcal{R}^D | n = 1, \cdots, N\}$，若要将其聚为 K 类（$K << N$），聚类中心为 $\mathcal{C} = \{\boldsymbol{c}_k \in \mathcal{R}^D | k = 1, \cdots, K\}$，具体做法 [5][6] 是：

1. 选择 K 个初始聚类中心点。
2. 将数据集 $\mathcal{X}$ 中的点聚到 K 个类中。聚类规则是如果 $\boldsymbol{x}_n$ 与 $\boldsymbol{c}_k$ 最相似，则将 $\boldsymbol{x}_n$ 聚到类 k 中，相似性的度量根据聚类目的的不同会有差别。

3. 根据聚类的结果更新聚类中心点的值。

4. 重复步骤第2步和第3步，直到聚类中心趋于稳定。

K-means 算法中，初始聚类中心点的选择对最终聚类结果有较大影响，一旦初始值选择的不好，可能无法得到有效的聚类结果，这也成为 K-means 算法的一个主要问题，而 LBG 算法正好给出了一种确定初始聚类中心点的方法。

2.2 LBG 算法简介

LBG[2] 算法由 Yoseph Linde, Andres Buzo, Robert M. Gray 三人于 1980 年提出，故而得名。LBG 算法所要解决的问题与 K-means 算法相同，不再赘述。为实现聚类的目的，首先应该定义失真测度。 $\boldsymbol{x}_n$ 与 $\boldsymbol{c}_k$ 的失真测度记为 $d(\boldsymbol{x}_n, \boldsymbol{c}_k)$。本文中使用了三种失真测度:

- 平方欧几里得距离（以下简称欧氏距离）， $d(\boldsymbol{x}_n, \boldsymbol{c}_k) = \|\boldsymbol{x}_n - \boldsymbol{c}_k\|^2$。
- 马哈拉诺比斯距离（以下简称马氏距离），$d(\boldsymbol{x}_n, \boldsymbol{c}_k) = (\boldsymbol{x}_n - \boldsymbol{c}_k)^T {\boldsymbol{\Sigma}_k}^{-1} (\boldsymbol{x}_n - \boldsymbol{c}_k)$， $\boldsymbol{\Sigma}_k$ 为聚类 k 中所有样本的协方差矩阵，为降低运算复杂度，我们取为对角阵。
- 余弦相似度[1] ， $d(\boldsymbol{x}_n, \boldsymbol{c}_k) = \cos < \boldsymbol{x}_n, \boldsymbol{c}_k >= \frac{\boldsymbol{x}_n \cdot \boldsymbol{c}_k}{\|\boldsymbol{x}_n\| \cdot \|\boldsymbol{c}_k\|}$。

如果样本 $\boldsymbol{x}_n$ 被聚到 j 类中，在欧氏距离和马氏距离下有：

$$j = \arg\min_k d(\boldsymbol{x}_n, \boldsymbol{c}_k) \tag{2-1}$$

在余弦相似度下有:

$$j = \arg\max_k d(\boldsymbol{x}_n, \boldsymbol{c}_k) \tag{2-2}$$

LBG 算法的主要流程是：

1. 计算全部样本的质心 c_0，作为初始的聚类中心点。

2. 如果当前聚类数量不大于 K，转下一步。

[1]严格的讲，余弦相似度并不能叫失真测度，为叙述方便，故沿用此说法

3. 用当前的聚类中心对所有样本进行聚类操作，聚类结束之后，计算新的聚类中心点，聚类中心点的计算方法将在2.3给出。该步骤与 K-means 算法的第2步和第3步相同。

4. 如果全局失真测度 $d_{global} = \frac{1}{N}\sum_{n=1}^{N} d(\boldsymbol{x}_n, \boldsymbol{c}_{\bar{k}})^2$ 已经收敛，即在欧氏和马氏距离下：

$$0 \leq \frac{d_{global}^{(t-1)} - d_{global}^{(t)}}{d_{global}^{(t-1)}} \leq \varepsilon \tag{2-3}$$

在余弦相似度下:

$$0 \leq \frac{d_{global}^{(t)} - d_{global}^{(t-1)}}{d_{global}^{(t-1)}} \leq \varepsilon \tag{2-4}$$

（$d_{global}^{(t)}$ 为当前全局失真测度， $d_{global}^{(t-1)}$ 为前一次迭代的全局失真测度， ε 为人为设定的参数）转下一步，否则转第3步。

5. 如果当前聚类数目等于 K，程序终止，否则按一定规则选择现有的聚类进行分裂操作，转第2步。

在上述的步骤3完成之后，有可能会出现 “空” 聚类，此处的 “空” 聚类定义为聚到其中的样本总数小于 $D+1$，这时应当选择非空聚类进行分裂以替换 “空” 聚类，而在步骤5中如果聚类总数小于目标数目，同样需要进行分裂操作。分裂策略有多种，我们选择所含样本数目最多的聚类进行分裂。分裂方法是：

$$\boldsymbol{c}_k^{(1,2)} = \boldsymbol{c}_k \pm \alpha \cdot \boldsymbol{\sigma}_k \tag{2-5}$$

$\boldsymbol{\sigma}_k = (\sigma_{k1}, \cdots, \sigma_{kD})$ 为 $\boldsymbol{\Sigma}_k$ 中的对角元素组成的列向量， α 是需要人工设定的系数。

分析上述 LBG 算法流程可以看出，LBG 算法从一个聚类中心点开始，不断重复 “聚类 -更新 -分裂” 的过程，最终得到一个各类中样本数目比较均衡的聚类模型。这种通过现有聚类中心点的分裂增加中心点数目的方法正可以用于 K-means 算法的初始化。实际上，在 [2] 对该算法的描述中，聚类中心点达到 K 之前的操作都被作者归为了初始化过程。

[2] $\bar{k}$ 为 $\boldsymbol{x}_n$ 被聚到的类

2.3 聚类中心点求算公式推导

在聚类操作完成之后，需要对聚类中心点的值执行更新操作。假设分到聚类 k 的样本集合为 $\mathcal{X}_k = \{\boldsymbol{x}_n | n = 1, \cdots, N_k\}$，对聚类中心点进行更新的目的就是使聚类中的样本到聚类中心点的平均失真测度最小（对欧氏距离和马氏距离）或最大（对余弦相似度）。下面对三种测度下的更新方法一一进行推导。

2.3.1 欧几里得距离

在欧氏距离下，中心点更新的目标函数是：

$$\min f(\boldsymbol{c}_k) = \frac{1}{N_k} \sum_{n=1}^{N_k} \|\boldsymbol{x}_n - \boldsymbol{c}_k\|^2 \tag{2-6}$$

计算其一阶导数得：

$$\begin{aligned} \frac{\partial f(\boldsymbol{c}_k)}{\partial \boldsymbol{c}_k} =& -\frac{2}{N_k} \sum_{n=1}^{N_k} (\boldsymbol{x}_n - \boldsymbol{c}_k) \\ =& 2 \left[\boldsymbol{c}_k - \frac{1}{N_k} \sum_{n=1}^{N_k} \boldsymbol{x}_n \right] \end{aligned} \tag{2-7}$$

因此更新后的中心点的值为：

$$\boldsymbol{c}_k = \frac{1}{N_k} \sum_{n=1}^{N_k} \boldsymbol{x}_n \tag{2-8}$$

2.3.2 马哈拉诺比斯距离

马氏距离下目标函数为：

$$\begin{aligned} \min f(\boldsymbol{c}_k) =& \frac{1}{N_k} \sum_{n=1}^{N_k} \sum_{i=1}^{D} \left(\frac{x_{ni} - c_{ki}}{\sigma_{ki}} \right)^2 \\ =& \sum_{i=1}^{D} \frac{1}{{\sigma_{ki}}^2} \sum_{n=1}^{N_k} (x_{ni} - c_{ki})^2 \end{aligned} \tag{2-9}$$

其一阶导数值为:

$$\frac{\partial f(\boldsymbol{c}_k)}{\partial \boldsymbol{c}_k} = 2 \cdot \left(\frac{1}{{\sigma_{k1}}^2} \left(c_{k1} - \frac{1}{N_k} \sum_{n=1}^{N_k} x_{n1} \right), \cdots, \frac{1}{{\sigma_{kD}}^2} \left(c_{kD} - \frac{1}{N_k} \sum_{n=1}^{N_k} x_{nD} \right) \right)^T \tag{2-10}$$

所以更新后的中心点的值与欧氏距离下相同，为 $\frac{1}{N_k}\sum_{n=1}^{N_k}\boldsymbol{x}_n$。

2.3.3 余弦相似度

余弦相似度情况下的目标函数为：

$$\begin{aligned}\max f(\boldsymbol{c}_k) =& \frac{1}{N_k}\sum_{n=1}^{N_k}\cos<\boldsymbol{x}_n,\boldsymbol{c}_k> \\ =& \frac{1}{N_k}\sum_{n=1}^{N_k}\frac{\boldsymbol{x}_n\cdot\boldsymbol{c}_k}{\|\boldsymbol{x}_n\|\cdot\|\boldsymbol{c}_k\|} \\ =& \frac{1}{N_k}\sum_{n=1}^{N_k}\widehat{\boldsymbol{x}}_n\cdot\widehat{\boldsymbol{c}}_k \\ =& \widehat{\boldsymbol{c}}_k\cdot\left(\frac{1}{N_k}\sum_{n=1}^{N_k}\widehat{\boldsymbol{x}}_n\right) \\ =& \left\|\frac{1}{N_k}\sum_{n=1}^{N_k}\widehat{\boldsymbol{x}}_n\right\|\cos<\widehat{\boldsymbol{c}}_k,\frac{1}{N_k}\sum_{n=1}^{N_k}\widehat{\boldsymbol{x}}_n>\end{aligned} \tag{2-11}$$

$\widehat{\boldsymbol{c}}_k$ 和 $\widehat{\boldsymbol{x}}_n$ 均为单位矢量，为了简化计算，设 $\boldsymbol{c}_k=\widehat{\boldsymbol{c}}_k$。当 $\widehat{\boldsymbol{c}}_k$ 与 $\frac{1}{N_k}\sum_{k=1}^{N_k}\widehat{\boldsymbol{x}}_n$ 共线时，目标函数取得最大值，因此有：

$$\boldsymbol{c}_k=\frac{\sum_{n=1}^{N_k}\widehat{\boldsymbol{x}}_n}{\|\sum_{n=1}^{N_k}\widehat{\boldsymbol{x}}_n\|} \tag{2-12}$$

2.3.4 小结

从上述的推导可以发现，在欧氏距离和马氏距离为失真测度时，聚类中心点是聚类中所有样本的质心。这并不是一个偶然现象，除了这两种失真测度外，Itakura-Saito 距离，相对熵、Kullback-Leibler 距离等也经常用于各种不同目标的聚类任务中，这些失真测度都属于 Bregman divergence，它们的聚类中心点均为聚类中所有样本的质心 [7]。由于这不是本文的主要工作，不再赘述。

在本文使用的三种不同失真测度下，进行聚类操作时均需要多次计算样本与聚类中心点之间的失真测度值，当数据规模很大时，运算开销会相当可观，

同时各样本在进行聚类时相互之间没有影响，也就是可以并行，于是我们可以利用高性能计算平台实现该算法，以加快运行速度。

在进行聚类中心点更新和分裂时需要计算聚类中样本的期望和方差，而二者的计算需要计算聚类中所有样本的和与平方和，数据规模增大时，直接进行计算可能会导致溢出发生，因此我们采取了一定手段以避免这种情况发生，以 i 维为例，由于

$$\mu_{ki} = \frac{1}{N_k} \sum_{n=1}^{N_k} x_{ni} \tag{2-13}$$

$$\sigma_{ki}^2 = \frac{1}{N_k} \sum_{n=1}^{N_k} x_{ni}^2 - \mu_{ki}^2 \tag{2-14}$$

所以我们通过计算

$$\sum_{n=0}^{N_k} \left(x_{ni} - \mu_{ki}^{(t-1)} \right) \tag{2-15}$$

代替

$$\sum_{n=1}^{N_k} x_{ni} \tag{2-16}$$

通过计算

$$\sum_{n=0}^{N_k} \left[x_{ni}^2 - \left(\mu_{ki}^{(t-1)} \right)^2 - \left(\sigma_{ki}^{(t-1)} \right)^2 \right] \tag{2-17}$$

代替

$$\sum_{n=1}^{N_k} x_{ni}^2 \tag{2-18}$$

从而使求和得到的值的期望在 0 附近，避免了溢出的危险。上面的式子中 $\mu_{ki}^{(t-1)}$ 和 $\sigma_{ki}^{(t-1)}$ 为上一轮迭代后聚类 k 第 i 维的均值和方差。

2.4 实验结果

2.4.1 二维空间数据聚类效果

为检验 LBG 算法在三种失真测度下的聚类效果，用 Matlab 生成 9000 组二维随机数据进行测试。这些数据总共由八个二维高斯分布函数生成，如

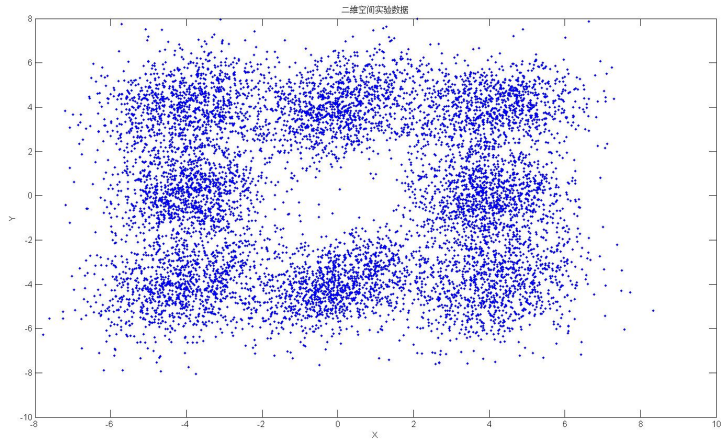

图 2.1: 二维空间原始数据

图2.1所示。 接下来的三次实验中，数据均被聚为 8 类。在图中用不同颜色和形状表示不同的聚类。

利用欧氏距离做失真测度，得到的聚类结果如图2.2所示。 图中的黄色圆

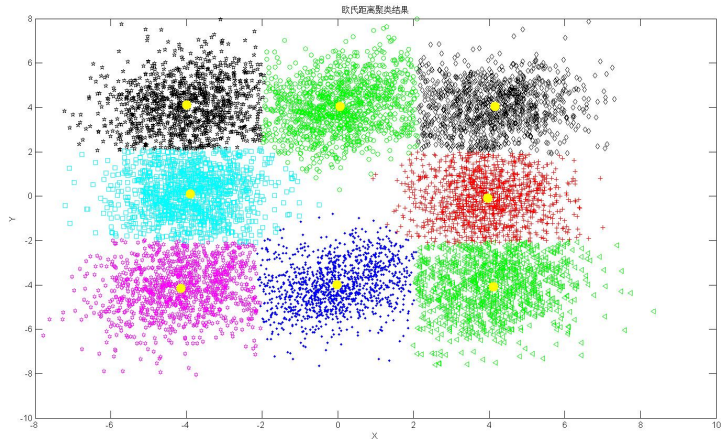

图 2.2: 二维空间欧氏距离聚类结果

点表示聚类中心点，从图上可以看出，聚类之间的边界线正是两聚类中心点连线的垂直平分线。这与数学计算的结果相一致。

用马氏距离做失真测度时，聚类结果如图2.3所示。 图中黄色圆点同样代

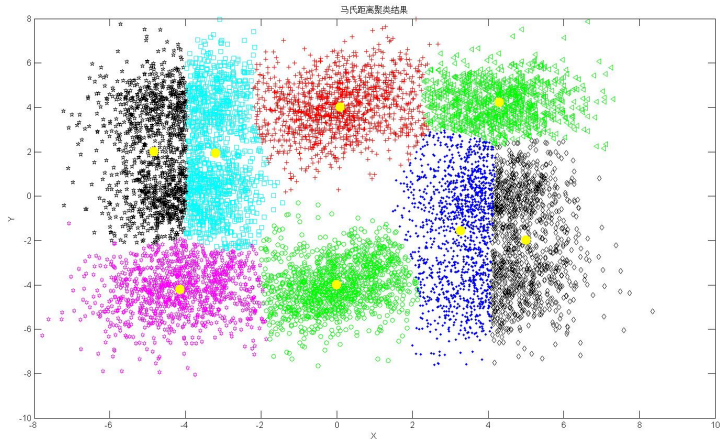

图 2.3: 二维空间马氏距离聚类结果

表聚类中心点。由聚类结果可以看出，当某个点到两个聚类中心点的欧氏距离相等时，该点会被聚到数据比较松散即方差比较大的类中，这与用马氏距离进行聚类的预期效果相符。

用余弦相似度做失真测度时，结果如图2.4所示。 图中的红线是原点与各

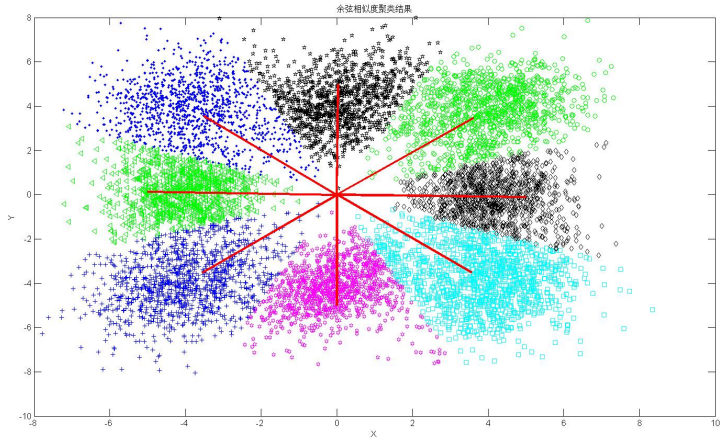

图 2.4: 二维空间余弦相似度聚类结果

聚类中心点的连线。由于余弦相似度只利用了训练数据的方向信息，方向相近的数据点会被聚到一起，相邻聚类的分界线是两聚类中心点与原点连线夹角的

角平分线，与实验结果相符。

综合上述实验结果可以发现，通过失真测度的选取，LBG 算法确实可以实现不同的聚类目的。

2.4.2 语音 PLP 特征数据聚类效果

本节使用的是 39 维的语音 PLP 特征数据，该数据集总计有 5,455,682 个训练样本，为 14.6 小时的数据，总共 811MB，是 SwitchBoard 数据集的一个子集。分别采用欧氏距离和余弦相似度对数据进行聚类操作，两次均是把数据聚为 8 类。

采用欧氏距离进行聚类时，方程2-5中的 α 取为 0.02，聚类结果如图2.5所示。经过 36 次迭代，聚类任务完成。 聚类中心点发生分裂时，平均距离会略

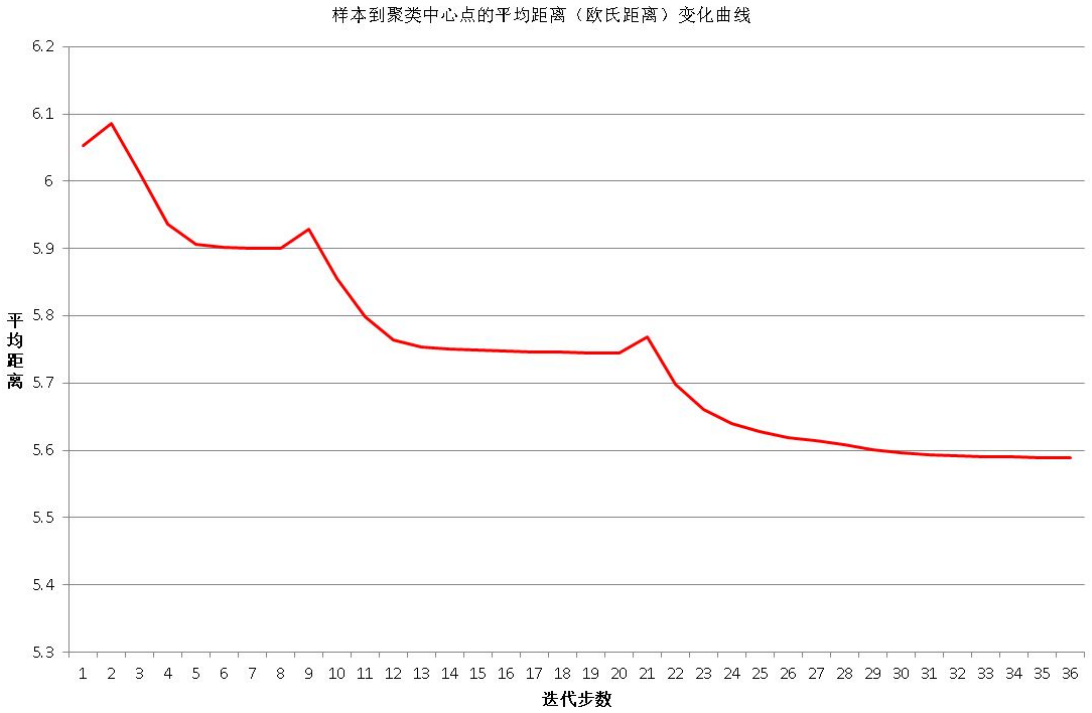

图 2.5: 欧氏平均距离变化图

有上升，这是因为此时的中心点并不是通过2.3 中的方法得到，不能保证样本到该点的距离最小，之后随着 LBG 算法的运行，样本到聚类中心点的平均距离逐渐下降直至收敛。该实验在 3 台 8 核机器上完成，用时 2 分钟。

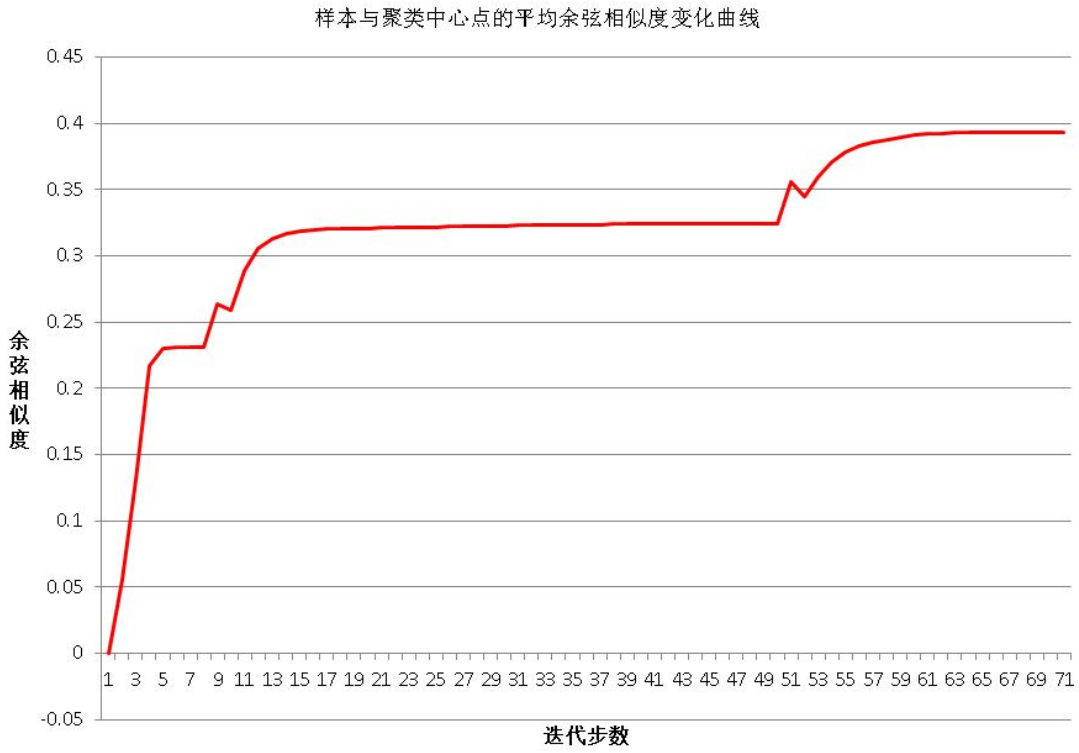

图 2.6: 平均余弦相似度变化图

采用余弦相似度聚类时，方程2-5中的 α 取为 0.04，聚类结果如图2.6所示。经 71 次迭代，聚类完成。余弦相似度从开始的 -0.00035 提高到 0.39337，有了显著提高。

该实验也是在 3 台 8 核机器上完成，总计用时 5 分钟。综合两次实验，LBG 算法在大规模数据上同样有很好的性能，且借助于高性能计算平台，运行速度可以达到非常快。正因为此，在接下来的两章中，我们会使用 LBG 算法对要训练的模型进行初始参数估计。

第 3 章　高斯混合模型

3.1　高斯混合模型简介

3.1.1　混合模型

在统计学中，有大量经典的概率（密度）分布函数，比如二项分布、高斯分布、指数分布、泊松分布、贝塔分布，t 分布等，它们在处理一些特定问题时可以达到很好的效果，但在处理现实中的许多复杂问题时，数据的概率分布形式通常是未知的、复杂的，这时单纯用这些分布函数对数据分布进行描述可能出现比较大的偏差。为了解决这一问题，我们通过多个已知分布函数的线性组合对未知分布函数进行近似往往可以得到比较好的效果, 由这种方法得到的混合分布模型称为混合模型[1]，数学描述如下: 假设我们有一族概率分布函数 $p_1(\boldsymbol{x}), p_2(\boldsymbol{x}), \cdots, p_K(\boldsymbol{x})$，混合分布函数为：

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k \cdot p_k(\boldsymbol{x}) \tag{3-1}$$

其中

$$\begin{aligned} &\omega_k \geq 0, 1 \leq k \leq K \\ &\sum_{k=1}^{K} \omega_k = 1 \end{aligned} \tag{3-2}$$

ω_k 为 $p_k(\boldsymbol{x})$ 在 $p(\boldsymbol{x})$ 中所占的权重，其意义是某个样本由分布 $p_k(\boldsymbol{x})$ 生成的概率是 ω_k。

根据组成混合分布的概率分布函数的不同，混合模型可以分为许多种，我们将要介绍的高斯混合模型就是混合模型的一种，在金融、语音识别、手写识别等领域都得到了广泛应用。

3.1.2　高斯混合模型

顾名思义，高斯混合模型是高斯分布的线性组合。图3.1给出了一个简单的一维空间上的例子。 图中蓝线表示单个的高斯分布，红线表示它们的和。由此

[1]本文所指的混合模型均为线性混合模型

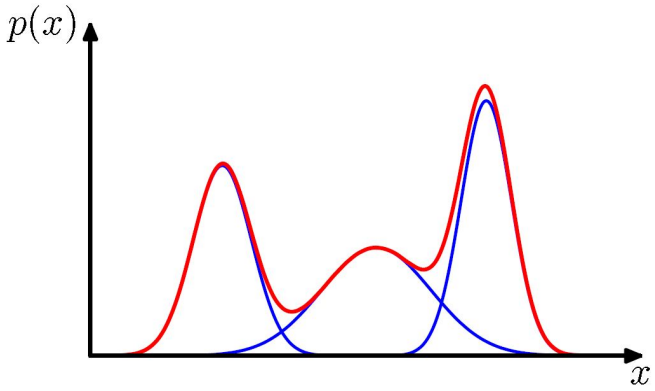

图 3.1: 一维空间高斯混合分布模型示例

可以看出仅三个高斯分布函数的组合就能近似复杂的分布函数，实际上通过调整高斯混合模型的各项参数，该模型几乎可以将任意的连续概率分布模型近似到任意精度 [6]。

在 D 维空间上的高斯分布函数为：

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T {\boldsymbol{\Sigma}_k}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right] \tag{3-3}$$

$\boldsymbol{\mu}_k$ 和 $\boldsymbol{\Sigma}_k$ 分别为高斯分布的期望和协方差矩阵。含 K 个 D 维高斯分布的混合模型为

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Omega}) = \sum_{k=1}^{K} \omega_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3-4}$$

$\boldsymbol{\mu}$ 和 $\boldsymbol{\Sigma}$ 分别为 K 个高斯分布的期望集合与协方差矩阵集合，$\Omega = (\omega_1, \cdots, \omega_K)^T$ 为各高斯分布的权重矢量。

混合模型的训练方法有许多种，如期望最大化算法（Expectation Maximization,EM）、马尔可夫链蒙特卡罗方法（Markov chain Monte Carlo）、矩匹配方法（Moment Matching）、谱方法（Spectral method）等。本文中高斯混合模型训练的目的是在给定样本集合 $\mathbf{X} = \{\boldsymbol{x}_n \in \mathcal{R}^D | n = 1, \cdots, N\}$ 和高斯分布个数 K 的情况下利用一定的方法确定高斯混合模型的参数集 $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Omega}\}$，以使下述对数似然概率最大：

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Omega}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \omega_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \tag{3-5}$$

解决这一问题最常用的是期望最大化算法。

3.2 期望最大化算法

在统计计算中，期望最大化算法是在概率模型中寻找最大似然（Maximum Likelihood,ML）估计或者最大后验（Maximum a Posterior,MAP）估计的算法，其中的概率模型依赖于隐藏变量（latent variables）。

在3.1.1的混合模型中，记模型参数为 $\boldsymbol{\theta} = \{\boldsymbol{\theta}_k | k = 1, \cdots, K\}$，训练数据集为 $\mathbf{X} = \{\boldsymbol{x}_n \in \mathcal{R}^D | n = 1, \cdots, N\}$，隐藏变量为 $\mathbf{Z} = \{\boldsymbol{z}_n \in \mathcal{R}^K | n = 1, \cdots, N\}$。对 $\boldsymbol{z}_n$ 有

$$\sum_{k=1}^{K} z_{nk} = 1, z_{nk} \in \{0, 1\} \tag{3-6}$$

对混合模型我们定义一个条件概率分布 $p(\boldsymbol{x}|\boldsymbol{z})$ 和边缘分布 $p(\boldsymbol{z})$，其中

$$p(z_k = 1) = \omega_k \tag{3-7}$$

则

$$p(\boldsymbol{z}) = \prod_{k=1}^{K} {\omega_k}^{z_k} \tag{3-8}$$

因此

$$p(\boldsymbol{x}|z_k = 1) = p_k(\boldsymbol{x}) \tag{3-9}$$

也可以写成

$$p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{k=1}^{K} p_k(\boldsymbol{x})^{z_k} \tag{3-10}$$

所以有

$$p(\boldsymbol{x}, \boldsymbol{z}) = \prod_{k=1}^{K} {\omega_k}^{z_k} p_k(\boldsymbol{x})^{z_k} \tag{3-11}$$

根据上一节的论述，我们要最大化的是

$$p(\mathbf{X}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \sum_{k=1}^{K} \omega_k \cdot p_k(\boldsymbol{x}_n) \tag{3-12}$$

也即：

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \tag{3-13}$$

根据贝叶斯公式，我们有

$$\begin{aligned}\ln p(\mathbf{X}|\boldsymbol{\theta}) &= \ln p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta}) - \ln p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}) \\ &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})} - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta})}{q(\mathbf{Z})} \\ &= \mathcal{L}(q,\boldsymbol{\theta}) + \mathrm{KL}(q \parallel p)\end{aligned} \tag{3-14}$$

其中 $q(\mathbf{Z})$ 为一概率分布函数， $\mathrm{KL}(q \parallel p)$ 为概率分布 $q(\mathbf{Z})$ 与 $p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta})$ 的 Kullback-Leibler 距离，由于

$$\mathrm{KL}(q \parallel p) \geq 0 \tag{3-15}$$

当且仅当 $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta})$ 时取等号，因此 $\mathcal{L}(q,\boldsymbol{\theta}) \leq \ln p(\mathbf{X}|\boldsymbol{\theta})$，如图3.2所示。期望最大化算法是一种迭代算法，分为 E 步和 M 步。就该目标函数而言在 E

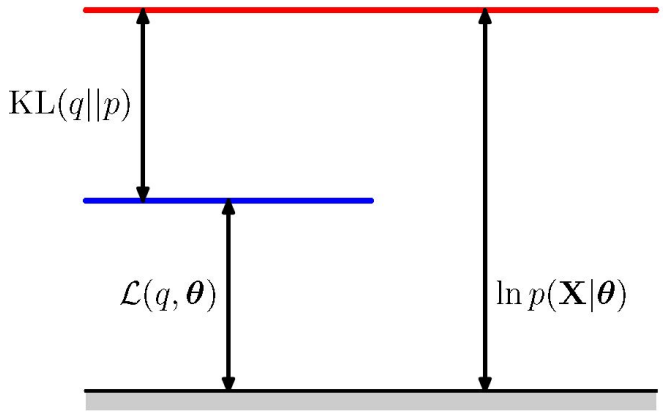

图 3.2: $\ln p(\mathbf{X}|\boldsymbol{\theta})$ 分解图 [6]

步，参数 $\boldsymbol{\theta}$ 并没有改变，记为 $\boldsymbol{\theta}^{\mathrm{old}}$，使 $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}_{\mathrm{old}})$，如图3.3 此步骤中经过 E 步之后，有：

$$\begin{aligned}\mathcal{L}(q,\boldsymbol{\theta}) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{\mathrm{old}}) \ln p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta}) - \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{\mathrm{old}}) \ln p(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{\mathrm{old}}) \\ &= \mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^{\mathrm{old}}) + \mathbf{const} \\ &= \ln p(\mathbf{X}|\boldsymbol{\theta})\end{aligned} \tag{3-16}$$

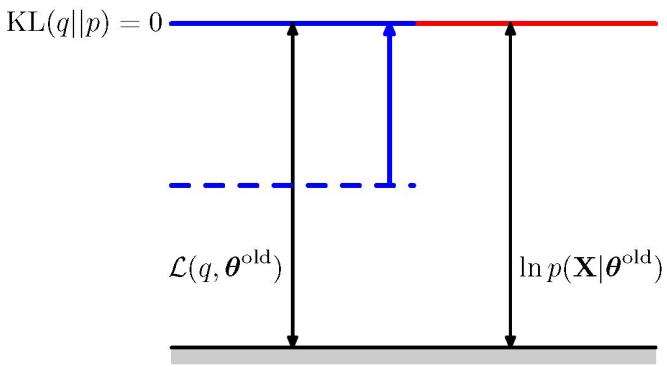

图 3.3: E 步示意图 [6]

因此，在 M 步我们要最大化 $\ln p(\mathbf{X}|\boldsymbol{\theta})$，只需最大化 $\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ 即可，也即应该计算新的参数值 $\boldsymbol{\theta}^{new}$：

$$\boldsymbol{\theta}^{\text{new}} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) \tag{3-17}$$

由于 $\boldsymbol{\theta}$ 发生了变化，此时 $q(\mathbf{Z}) \neq p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{new}})$，所以 $\text{KL}(q \parallel p) > 0$，而 $\mathcal{L}(q, \boldsymbol{\theta})$ 的值也增大了，所以 $\ln p(\mathbf{X}|\boldsymbol{\theta})$ 的值增大，如图3.4所示。

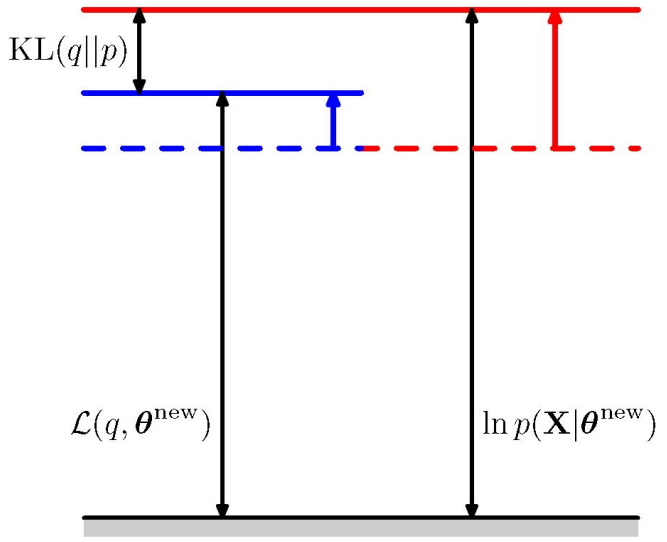

图 3.4: M 步示意图 [6]

由上述分析可以看出，期望最大化算法确实可以使最大似然概率增加，能

够被用来训练混合模型。在训练过程中，要交替执行 E 步和 M 步，直到满足迭代终止条件，模型训练结束。

3.3 似然率最大化高斯混合模型训练方法

3.3.1 计算公式推导

在高斯混合模型中，要通过优化参数集 $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Omega}\}$ 使模型在训练集上的似然概率最大，从上一节的分析知道，参数的计算是根据方程3-17，由于

$$\begin{aligned} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \\ &= \mathbb{E}_{\mathbf{Z}}[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})] \end{aligned} \tag{3-18}$$

所以方程3-17实际上是完整数据 $\{\mathbf{X}, \mathbf{Z}\}$ 的对数似然期望，对高斯混合分布有：

$$p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Omega}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \omega_k^{z_{nk}} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{nk}} \tag{3-19}$$

$$\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Omega}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \{\ln \omega_k + \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\} \tag{3-20}$$

所以，我们有下式：

$$\mathbb{E}_Z[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Omega})] = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{E}[z_{nk}] \{\ln \omega_k + \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\} \tag{3-21}$$

接下来的关键是求算 $\mathbb{E}[z_{nk}]$。因为

$$p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Omega}) = \frac{p(\mathbf{Z}, \mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Omega})}{p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Omega})} \tag{3-22}$$

故而

$$p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Omega}) \propto \prod_{n=1}^{N} \prod_{k=1}^{K} \omega_k^{z_{nk}} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{nk}} \tag{3-23}$$

所以有

$$\begin{aligned}\mathbb{E}[z_{nk}] &= \frac{\sum\limits_{z_{nk}} z_{nk}[\omega_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{z_{nk}}}{\sum\limits_{z_{nj}} [\omega_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)]^{z_{nj}}} \\ &= \frac{\omega_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum\limits_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \\ &= \gamma(z_{nk})\end{aligned} \tag{3-24}$$

故我们要最大化的函数为

$$\mathbb{E}_Z[\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Omega})] = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma(z_{nk})\{\ln \omega_k + \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\} \tag{3-25}$$

对上式求导，求导过程此处省略，我们可以得到，在 M 步得到的更新后的参数为

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}) \tag{3-26}$$

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{3-27}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T \tag{3-28}$$

$$\omega_k^{\text{new}} = \frac{N_k}{N} \tag{3-29}$$

3.3.2 模型训练步骤

我们已经知道了在利用期望最大化算法训练似然率最大化高斯混合模型的求算公式，现在面临的问题是如何初始化高斯混合模型。

从上述的整个分析过程可以看出，高斯混合模型其实是一种“软”聚类模型，$\gamma(z_{nk})$ 实际上可以看作样本 $\boldsymbol{x}_n$ 由第 k 个高斯分布生成的概率，由此我们可以利用第二章提到的 LBG 算法对其进行初始化。具体做法是，将训练数据集 $\boldsymbol{X}$ 通过 LBG 算法聚为 K 类，其中聚到第 k 类的样本集合为 $\boldsymbol{X}_k = \{\boldsymbol{x}_{n_k}|n_k = 1, \cdots, N_k\}$，用 $\boldsymbol{X}_k$ 的期望和协方差矩阵初始化 $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$，用 $\frac{N_k}{N}$ 初始化 ω_k。

模型训练的完整步骤为：

1. 初始化：利用 LBG 算法初始化 $\boldsymbol{\mu}^{\text{old}}, \boldsymbol{\Sigma}^{\text{old}}, \boldsymbol{\Omega}^{\text{old}}$
2. E 步：根据 $\boldsymbol{\mu}^{\text{old}}, \boldsymbol{\Sigma}^{\text{old}}, \boldsymbol{\Omega}^{\text{old}}$ 计算 $\gamma(z_{nk})$
3. M 步：根据式3-26、3-27、3-28和3-29计算 $\boldsymbol{\mu}^{\text{new}}, \boldsymbol{\Sigma}^{\text{new}}, \boldsymbol{\Omega}^{\text{new}}$
4. 检查：将 $\boldsymbol{\mu}^{\text{new}}, \boldsymbol{\Sigma}^{\text{new}}, \boldsymbol{\Omega}^{\text{new}}$ 赋给 $\boldsymbol{\mu}^{\text{old}}, \boldsymbol{\Sigma}^{\text{old}}, \boldsymbol{\Omega}^{\text{old}}$，检查对数似然概率是否已经收敛或者循环结束条件是否已经满足，如果是，退出，如果否，转第2步。

3.3.3 小结

分析高斯混合模型的训练步骤，我们可以发现，运算的开销主要在计算 $\gamma(z_{nk})$ 上，对于不同的训练样本， $\gamma(z_{nk})$ 的求算是相互独立的，这为并行计算创造了条件。因此可以利用高性能计算平台实现该算法。

为了降低运算复杂度，我们假设训练数据各维之间是相互独立的，也就是说，混合模型中的每个协方差矩阵均为对角阵。求算每个 $\gamma(z_{nk})$ 时，都要计算 K 个概率密度函数，而高斯分布函数是指数函数，对此，我们可以通过求算概率密度函数的自然对数来简化运算。当要计算各概率密度的和时，即已知 $\ln p_1$、 $\ln p_2$ 求算 $p_1 + p_2$ 时，通过计算 $\ln p_1 + \ln\left(1 + \exp(\ln p_2)/\exp(\ln p_1)\right)$ 得到 $\ln(p_1 + p_2)$ 最后进行指数运算达到目的。

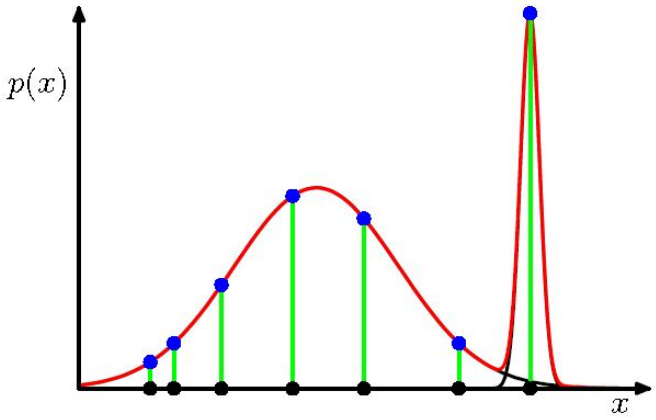

图 3.5: 病态高斯分布示意图 [6]

高斯混合模型训练过程中要解决的一个重要问题是如何避免病态高斯分布的出现，如图3.5所示就是一个出现了病态高斯分布的例子。图中右侧的高斯分布方差趋近于 0，使得整个模型的似然概率非常大，这是一种过拟合的情况，会导致模型在测试集上的性能下降。为避免这种情况的发生，我们为每个高斯分布协方差矩阵的元素设置了一个下界，当矩阵某个元素小于该值时，以该值替代之。在本文中，下界值是全部训练样本协方差矩阵值的 1%。

与 LBG 算法相同，高斯混合模型的训练也需要求算期望和协方差矩阵，所以数据规模增大时同样面临着溢出的风险，由于本文中的协方差矩阵为对角阵，所以我们采取类似于2-15和2-17的方案解决这一问题。

3.4 实验结果

3.4.1 二维空间数据训练结果

为了检验基于期望最大化算法的似然率最大化高斯混合模型训练方法是否有效，我们采用二维空间上的数据进行了实验。

利用 Matlab 生成 4000 组符合高斯混合分布的随机数据，如图3.6所示。该高斯混合分布由四个高斯分布函数构成，相应参数见表3.1。 首先利用 LBG

表 3.1: 高斯混合分布参数

高斯分布	1	2	3	4
ω	0.2	0.3	0.3	0.2
$\boldsymbol{\mu}$	$\begin{pmatrix} 1 \\ 2 \end{pmatrix}$	$\begin{pmatrix} -3 \\ -5 \end{pmatrix}$	$\begin{pmatrix} -2 \\ 2 \end{pmatrix}$	$\begin{pmatrix} 3 \\ -4 \end{pmatrix}$
$\boldsymbol{\Sigma}$	$\begin{pmatrix} 2 & 0 \\ 0 & 0.5 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0.5 & 0 \\ 0 & 0.8 \end{pmatrix}$	$\begin{pmatrix} 1.2 & 0 \\ 0 & 0.6 \end{pmatrix}$

算法将训练数据聚为 4 类，利用聚类结果作为初始参数估计进行训练。在训练中，将期望最大化算法的迭代次数设为 200。迭代结束时，得到一组高斯混合模型的参数，见表3.2。

对比表3.1和3.2，考虑到数据生成的随机性和数据总量的有限性，可以认为期望最大化算法能够对高斯混合模型的参数进行很好的近似。

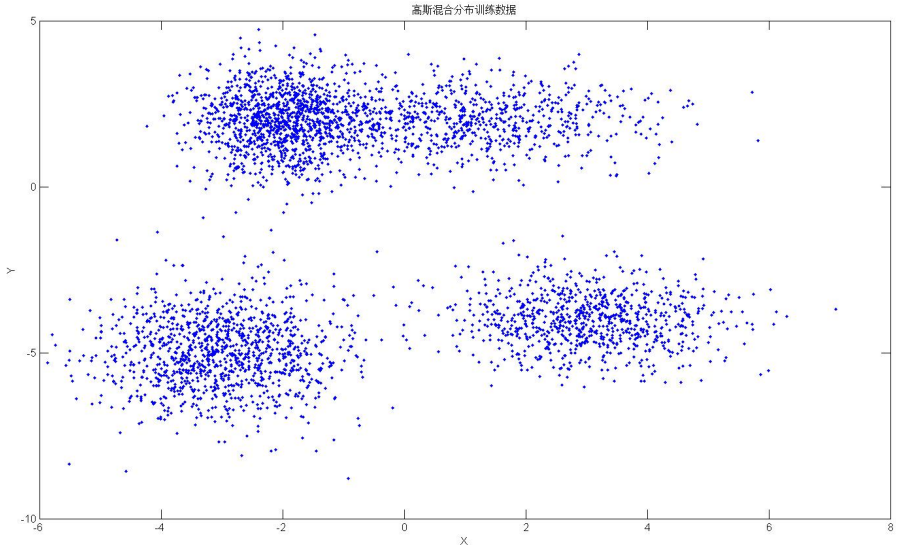

图 3.6: 高斯混合分布训练数据

表 3.2: 训练得到的高斯混合分布参数

高斯分布	1	2	3	4
ω	0.202	0.294	0.290	0.214
$\boldsymbol{\mu}$	$\begin{pmatrix} 0.946 \\ 1.995 \end{pmatrix}$	$\begin{pmatrix} -3.000 \\ -5.001 \end{pmatrix}$	$\begin{pmatrix} -2.036 \\ 1.995 \end{pmatrix}$	$\begin{pmatrix} 3.022 \\ -4.021 \end{pmatrix}$
$\boldsymbol{\Sigma}$	$\begin{pmatrix} 2.309 & 0 \\ 0 & 0.496 \end{pmatrix}$	$\begin{pmatrix} 1.000 & 0 \\ 0 & 1.072 \end{pmatrix}$	$\begin{pmatrix} 0.480 & 0 \\ 0 & 0.761 \end{pmatrix}$	$\begin{pmatrix} 1.241 & 0 \\ 0 & 0.581 \end{pmatrix}$

3.4.2 语音 PLP 特征数据训练结果

本节使用的数据与2.4.2相同。

在本实验中，训练数据首先由 LBG 算法聚为 8 类，以其结果作为对模型的初始参数估计。为训练高斯混合模型，总共进行了 30 次迭代。模型训练过程中，平均对数似然概率变化曲线如图3.7所示。

从图上可以看出，在迭代过程中，平均对数似然概率不断增大，在第 13 次迭代之后收敛，基本不再发生变化。该实验在 3 台 8 核机器上完成，总计用

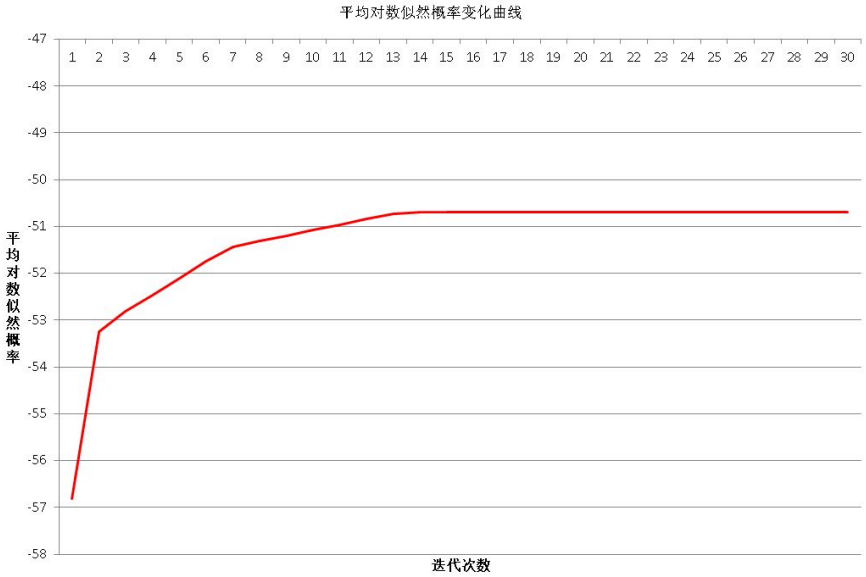

图 3.7: 平均对数似然概率变化曲线

时 30 分钟。

综合上述的实验结果，应用期望最大化算法确实能够增大高斯混合模型在训练集上的似然概率，与理论分析结果相符。

第 4 章　基于样本分离边距和最小分类误差的多原型手写识别模型

4.1　最小分类误差方法概述

4.1.1　问题描述

在前面两章，我们讨论的都是非监督式学习，本章要讲述的多原型手写识别模型则属于监督式学习的范畴。监督式学习是机器学习的一个重要组成部分，它通过训练样本和期望输出构建一个模型，通过此模型可以对未知输入样本的性质进行预测。模型的输出可以是一个连续的值（称为回归分析），或是一个分类标签（称作分类）。显然我们要讨论的手写识别模型属于后者。

假设我们有训练样本 $\mathcal{X} = \{\boldsymbol{x}_r \in \mathcal{R}^D | r = 1, \cdots, R\}$，样本标签 $\mathcal{I} = \{i_r | r = 1, \cdots, R\}$， N 个样本有 M 种标签，也就是说所有样本属于 M 个类，记 M 个类的集合为 $\mathcal{C} = \{C_i | i = 1, \cdots, M\}$。记类 C_i 的参数为 $\boldsymbol{\lambda}_i$，整个分类器的参数集合为 $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}_i | i = 1, \cdots, M\}$。分类器训练的目的就是通过确定分类器参数集和分类决策规则（decision rule），使得样本被误分类的概率最小，误分类的概率通常被经验性地近似为在训练样本集上的分类误差，即目标函数为：

$$\min_{\boldsymbol{\Lambda}} \frac{R_{\text{error}}}{R} \tag{4-1}$$

R_{error} 为误分类的样本个数。如果每次误分类都对应一个惩罚项或者损失函数，那么目标函数则变为最小化损失函数的期望。

4.1.2　贝叶斯决策理论

贝叶斯决策理论是一种基本的统计学的分类决策方法，也是众多模式分类技术的基础。贝叶斯决策理论假设数据集与所属分类器的联合概率分布 $P_{\boldsymbol{\Lambda}}(\boldsymbol{x}, C_i)$ 是知道的，于是就可以得到边缘分布和条件概率分布。

为衡量分类器的性能，我们为每两个类 (C_i, C_j) 定义一个损失函数 e_{ji}，它表示将属于类 C_i 的样本分到类 C_j 中的损失，其中 $e_{ii} = 0$，表示分类正确。

对任意一个样本 $\boldsymbol{x}$ 将其类 C_i 中的条件损失函数为

$$L(C_i | \boldsymbol{x}) = \sum_{j=1}^{M} e_{ji} P_{\boldsymbol{\Lambda}}(C_j | \boldsymbol{x}) \tag{4-2}$$

$P_{\boldsymbol{\Lambda}}(C_j|\boldsymbol{x})$ 为后验概率分布。由此求得分类器的期望损失函数表达式为

$$\mathcal{L} = \int L(C(\boldsymbol{x})|\boldsymbol{x}) P_{\boldsymbol{\Lambda}}(\boldsymbol{x}) d\boldsymbol{x} \tag{4-3}$$

$C(\bullet)$ 代表分类操作，若要最小化 $\mathcal{L}$，则

$$L(C(\boldsymbol{x})|\boldsymbol{x}) = \min_i L(C_i|\boldsymbol{x}) \tag{4-4}$$

一般的

$$e_{ji} = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } i \neq j \end{cases} \tag{4-5}$$

因此

$$L(C_i|\boldsymbol{x}) = \sum_{i \neq j} P_{\boldsymbol{\Lambda}}(C_j|\boldsymbol{x}) = 1 - P_{\boldsymbol{\Lambda}}(C_i|\boldsymbol{x}) \tag{4-6}$$

所以我们得到贝叶斯决策规则为

$$C(\boldsymbol{x}) = C_i \ \text{ if } \ P_{\boldsymbol{\Lambda}}(C_i|\boldsymbol{x}) = \max_j P_{\boldsymbol{\Lambda}}(C_j|\boldsymbol{x}) \tag{4-7}$$

该决策规则还常常写成先验概率和联合概率分布的形式

$$C(\boldsymbol{x}) = C_i \ \text{ if } \ P_{\boldsymbol{\Lambda}}(\boldsymbol{x}|C_i) P_{\boldsymbol{\Lambda}}(C_i) = \max_j P_{\boldsymbol{\Lambda}}(\boldsymbol{x}|C_j) P_{\boldsymbol{\Lambda}}(C_j) \tag{4-8}$$

贝叶斯决策又称最大后验概率决策，该决策下获得的最小分类误差又称贝叶斯风险 [8]。

根据上面的分析，我们发现贝叶斯决策要求分类器设计者完全知晓数据集概率分布的形式和相关参数的值，而这些往往是未知的。在概率分布形式已知的情况下，可以利用数据集去估计相关参数，而在概率分布形式未知的情况下，设计者只能对其分布形式进行假设，然后利用数据集去估计参数的值。也就是说贝叶斯决策方法并不是直接去最小化与分类误差相关的损失函数，而是去拟合数据集的概率分布。当假设的概率分布与真实情况不同时，最优的概率分布估计并不能带来性能最优的分类器，且概率分布的估计需要大量的数据，而数据的收集与标注代价是高昂的。诸多条件的限制导致贝叶斯决策方法得到的分类器很可能得不到最优的性能。

4.1.3 最小分类误差方法一般形式

与贝叶斯决策方法不同，最小分类误差方法直接去最小化损失函数，而非拟合数据集的概率分布。

最小分类误差方法最先由 Shunichi Amari[9] 提出，由 Biing-Hwang Juang 和 Shigeru Katagiri[10] 进行了扩展和完善。最小分类误差方法的关键在于目标函数的定义，定义最小分类误差函数有三个主要步骤 [11]。

第一步，定义判别函数。为每个类 C_i 定义一个判别函数 $g_i(\boldsymbol{x};\boldsymbol{\Lambda})$，该判别函数不必是概率分布函数，根据分类器结构等的不同，判别函数可以是任意合理的测度，比如距离、相似度等。本文用到的判别函数将在4.1.4给出。假设要处理的样本为 $\boldsymbol{x}$，在使用判别函数方法时，拥有最大判别函数值的类被认为是样本 $\boldsymbol{x}$ 被识别为的类，记为 $r(\boldsymbol{x};\boldsymbol{\Lambda})$ 即

$$r(\boldsymbol{x};\boldsymbol{\Lambda}) = \arg\max_j g_j(\boldsymbol{x};\boldsymbol{\Lambda}) \tag{4-9}$$

第二步，对每个训练样本 $\boldsymbol{x}_r$ 定义一个近似的误分类测度 $d(\boldsymbol{x}_r, i_r;\boldsymbol{\Lambda})$。$d(\boldsymbol{x}_r, i_r;\boldsymbol{\Lambda})$ 的符号如果为负表示分类正确，如果为正，表示分类错误，$|d(\boldsymbol{x}_r, i_r;\boldsymbol{\Lambda})|$ 的大小则代表分类结果的可信程度。一个常用的误分类测度为

$$d(\boldsymbol{x}_r, i_r;\boldsymbol{\Lambda}) = -g_{i_r}(\boldsymbol{x}_r;\boldsymbol{\Lambda}) + G_{i_r}(\boldsymbol{x}_r;\boldsymbol{\Lambda}) \tag{4-10}$$

其中

$$G_{i_r}(\boldsymbol{x};\boldsymbol{\Lambda}) = \frac{1}{\eta}\ln\left[\frac{1}{|\mathcal{M}(\boldsymbol{x};j)|}\sum_{n\in\mathcal{M}(\boldsymbol{x};j)}\exp\left[\eta g_n(\boldsymbol{x};\boldsymbol{\Lambda})\right]\right] \tag{4-11}$$

$\mathcal{M}(\boldsymbol{x};j)$ 表示样本 $\boldsymbol{x}$ 可能被识别为的所有标签的集合，其中不包括 $\boldsymbol{x}$ 的真实标签，$|\mathcal{S}|$ 表示集合 $\mathcal{S}$ 中元素的个数。本文使用的基于样本分离边距的误分类测度将在4.1.5进行介绍。

第三步，定义一个平滑可微的函数用于近似样本 x 被识别为 $r(\boldsymbol{x};\boldsymbol{\Lambda})$ 的 0-1 损失函数（正确分类为 0，错误分类为 1）。一个常用的近似函数为 S 函数。

如图4.1所示为 S 函数 $f(x) = \frac{1}{1+\exp(-7x)}$ 的曲线，由图可见，S 函数确实可以很好的近似 0-1 损失函数。所以我们采用如下的 S 函数对损失函数进行近似

$$l(\boldsymbol{x}_r, i_r;\boldsymbol{\Lambda}) = \frac{1}{1+\exp[-\alpha d(\boldsymbol{x}_r, i_r;\boldsymbol{\Lambda}) + \beta]} \tag{4-12}$$

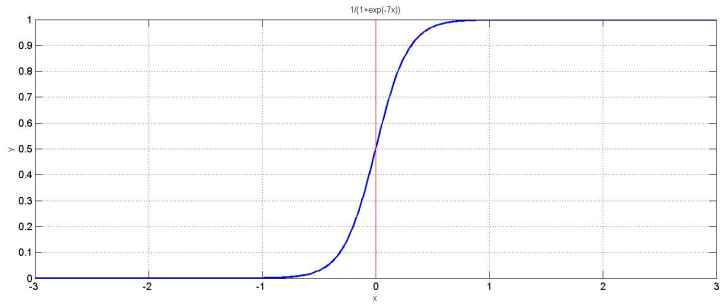

图 4.1: S 函数曲线示意图

其中 $\alpha > 0$，这样分类正确时由于 $d(\boldsymbol{x}_r, i_r; \boldsymbol{\Lambda})$ 为负，使得4-12接近于 0。故在整个数据集上要最小化的目标函数为

$$\ell(\mathcal{X}, \mathcal{I}; \boldsymbol{\Lambda}) = \frac{1}{R} \sum_{r=1}^{R} \frac{1}{1 + \exp[-\alpha d(\boldsymbol{x}_r, i_r; \boldsymbol{\Lambda}) + \beta]} \tag{4-13}$$

通过适当选择判别函数、误分类测度以及 S 型 0-1 损失函数的参数 α 和 β， 4-13可以无限近似训练集样本的错误分类率即方程4-1。因此对4-13的最小化可以达到最小化分类误差的目的。

4.1.4 多原型分类器模型判别函数

在多原型分类器模型中，每个类 C_i 的参数集为 $\boldsymbol{\lambda}_i = \{\boldsymbol{m}_{ik} | k = 1, \cdots, K_i\}$，其中 $\boldsymbol{m}_{ik}$ 为 D 维向量，表示类 C_i 的一个原型。在本文中我们使用欧氏距离定义判别函数如下：

$$g_i(\boldsymbol{x}; \boldsymbol{\Lambda}) = \frac{1}{\eta} \left[\frac{1}{K_i} \sum_{k=1}^{K_i} \exp\left[-\eta \cdot \|\boldsymbol{x} - \boldsymbol{m}_{ik}\|^2 \right] \right] \tag{4-14}$$

方程4-14是连续可微的，当 $\eta \to +\infty$ 时，判别函数为：

$$g_i(\boldsymbol{x}; \boldsymbol{\Lambda}) = -\min_k \|\boldsymbol{x} - \boldsymbol{m}_{ik}\|^2 \tag{4-15}$$

即与类 C_i 中的原型越接近，判别函数越大。分类决策规则为：

$$\boldsymbol{x} \in C_k \quad \text{if} \quad k = \arg\max_i g_i(\boldsymbol{x}; \boldsymbol{\Lambda}) \tag{4-16}$$

4.1.5 基于样本分离边距的误分类测度

样本分离边距的思想最先由 Amari[9] 提出，在 [11][12][13] 中被用于训练手写识别模型。

给定标签为 i_r 的训练样本 $\boldsymbol{x}$，样本 $\boldsymbol{x}$ 可能被识别为除 i_r 之外所有标签的集合为 $\mathcal{M}(\boldsymbol{x};j)$，对 $q \in \mathcal{M}(\boldsymbol{x};j)$，类 C_{i_r} 与类 C_q 的决策边界为

$$\mathcal{B}_{i_r,q} = \{\boldsymbol{z} | g_{i_r}(\boldsymbol{z};\boldsymbol{\Lambda}) - g_q(\boldsymbol{z};\boldsymbol{\Lambda}) = 0\} \tag{4-17}$$

$\boldsymbol{x}$ 到 $\mathcal{B}_{i_r,q}$ 的距离记为 $|\mathrm{dis}(\boldsymbol{x},\mathcal{B}_{i_r,q})|$，当 $\boldsymbol{x}$ 被正确分类，即 $r(\boldsymbol{x};\boldsymbol{\Lambda}) = i_r$ 时，$\mathrm{dis}(\boldsymbol{x},\mathcal{B}_{i_r,q})$ 为正，否则为负。样本分离边距定义为

$$\rho(\boldsymbol{x}) = \frac{1}{\eta} \ln \left[\frac{1}{|\mathcal{M}(\boldsymbol{x};j)|} \sum_{q \in \mathcal{M}(\boldsymbol{x};j)} \exp\left[\eta \mathrm{dis}(\boldsymbol{x},\mathcal{B}_{i_r,q})\right] \right] \tag{4-18}$$

当 $\eta \to +\infty$ 时

$$\rho(\boldsymbol{x}) = \min_{q \in \mathcal{M}(\boldsymbol{x};j)} \mathrm{dis}(\boldsymbol{x},\mathcal{B}_{i_r,q}) \tag{4-19}$$

本文中我们采用的基于多原型分类器的判别函数为4-15，此时4-19的计算公式 [12] 为

$$\rho(\boldsymbol{x}) = \frac{g_{i_r}(\boldsymbol{x};\boldsymbol{\lambda}_{i_r}) - g_q(\boldsymbol{x};\boldsymbol{\lambda}_q)}{2\|\boldsymbol{m}_{i_r\widehat{k}} - \boldsymbol{m}_{q\bar{k}}\|} \tag{4-20}$$

其中

$$\begin{aligned} \widehat{k} &= \arg\min_k \|\boldsymbol{x} - \boldsymbol{m}_{i_r k}\|^2 \\ q &= \arg\max_{i \in \mathcal{M}} g_i(\boldsymbol{x}_r;\boldsymbol{\lambda}_i) \\ \bar{k} &= \arg\min_k \|\boldsymbol{x}_r - \boldsymbol{m}_{qk}\|^2 \end{aligned} \tag{4-21}$$

如图4.2所示。 此时的样本分离边距表示训练样本到它真实所属的类和除此之外最有可能被分到的类的边界 $\mathcal{B}_{i_r,q}$ 的距离，分类正确时，该值为正，错误时为负，因此我们定义基于样本分离边距的误分类测度为：

$$d(\boldsymbol{x}_r, i_r;\boldsymbol{\Lambda}) = -\rho(\boldsymbol{x}_r) \tag{4-22}$$

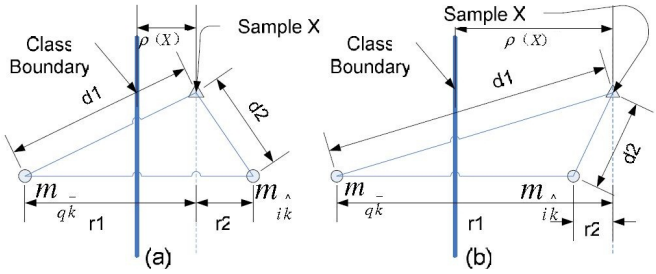

图 4.2: 基于多原型分类器的样本分离边距示意图

使用基于样本分离边距的最小分类误差方法可以在最小化分类误差的同时使训练样本远离决策边界，因而能够获得比较好的分类效果。[12] 中的对比实验证明了这一点。

4.2 多原型手写识别模型训练步骤

4.2.1 模型初始化

本章中，我们要优化的是手写识别模型，因此前面所说的类的集合 $\mathcal{C}$ 为字符集合。每个字符要有若干训练样本，记字符 C_j 的训练样本集合为 $\mathcal{X}_j = \{\boldsymbol{x}_r \in \mathcal{R}^D | 1 \leq r \leq R, i_r = j\}$。为获得模型的初始参数集 $\boldsymbol{\Lambda}^{(0)}$，我们对每个样本集合 $\mathcal{X}_j$ 使用 LBG 算法进行聚类，聚类的个数为 K_j，聚类得到的中心点的值即为字符 C_j 的初始原型参数值 $\boldsymbol{\lambda}_j$。

4.2.2 训练数据预处理

本章手写识别模型训练的数据集是汉字。作为一种语素文字，与拼音文字相比，字符的总数异常庞大，1994 年冷玉龙等的《中华字海》，收录的汉字多达 85000 字，仅仅考虑常用文字，也有数千之多。本章模型训练采用的分类决策规则为4-16，若要计算训练样本与每个字符的判别函数，运算量会非常巨大，所以在对模型进行优化之前，有必要对训练样本进行预处理。

由于 LBG 算法的良好性能，由其初始化的参数是模型最优参数的良好估计。因此，我们可以使用由 LBG 算法初始化的模型，得到每个训练样本被误分入的可能性最大的若干字符的集合，在后续的模型优化过程中，只需计算该

集合中的字符对应的判别函数以及样本所属字符的判别函数即可。具体做法是，对训练样本 $\boldsymbol{x}$，计算其对应的所有判别函数 $\mathcal{G}(\boldsymbol{x}) = \{g_i(\boldsymbol{x};\boldsymbol{\Lambda})|i = 1,\cdots,M\}$，从其中选择最大的 N 个判别函数对应的字符标签（真实标签除外），组成样本 $\boldsymbol{x}$ 的最易误分类标签集合，即 $\mathcal{M}(\boldsymbol{x};j)$。因为初始化的参数是最优参数的良好估计，所以在后续的优化过程中 $\mathcal{M}(\boldsymbol{x};j)$ 基本保持稳定，保证了优化过程的准确性。

预处理的使用，大大加快了程序执行速度。在4.4的实验中，要训练 9282 个字符的手写识别模型，不对数据进行预处理，一轮迭代用时为 3.5h，对数据进行预处理后，一轮迭代只需 1 分钟（$N = 50$）。

4.2.3 模型优化

在本模型中，对样本 $\boldsymbol{x}_r$ 我们使用的判别函数为：

$$g_i(\boldsymbol{x}_r;\boldsymbol{\Lambda}) = -\min_k \|\boldsymbol{x}_r - \boldsymbol{m}_{ik}\|^2 \tag{4-23}$$

误分类测度为：

$$\begin{aligned} d(\boldsymbol{x}_r, i_r;\boldsymbol{\Lambda}) &= \frac{-g_{i_r}(\boldsymbol{x}_r;\boldsymbol{\lambda}_{i_r}) + g_q(\boldsymbol{x};\boldsymbol{\lambda}_q)}{2\|\boldsymbol{m}_{i_r\widehat{k}} - \boldsymbol{m}_{q\bar{k}}\|} \\ &= \frac{\|\boldsymbol{x}_r - \boldsymbol{m}_{i_r\widehat{k}}\|^2 - \|\boldsymbol{x}_r - \boldsymbol{m}_{q\bar{k}}\|^2}{2\|\boldsymbol{m}_{i_r\widehat{k}} - \boldsymbol{m}_{q\bar{k}}\|} \end{aligned} \tag{4-24}$$

相关参数的意义见4-21。损失函数为4-12，最小化的目标函数4-13。最小化该函数的方法有许多种，如序列化广义概率下降（Generalized Probabilistic Descent,GPD）算法 [12][14][15]、Quickprop 算法、Rprop 算法等。这些算法都是基于梯度的，为此，对于每个参数 $\boldsymbol{m}_{ik}$ 我们需要计算其相对于目标函数的梯度。

对

$$l(\boldsymbol{x}_r, i_r;\boldsymbol{\Lambda}) = \frac{1}{1 + \exp[-\alpha d(\boldsymbol{x}_r, i_r;\boldsymbol{\Lambda}) + \beta]} \tag{4-25}$$

我们有

$$\frac{\partial l(\boldsymbol{x}_r, i_r;\boldsymbol{\Lambda})}{\partial \boldsymbol{m}_{i_r\widehat{k}}} = \alpha l(1-l)\left[-\frac{\boldsymbol{x}_r - \boldsymbol{m}_{i_r\widehat{k}}}{\|\boldsymbol{m}_{i_r\widehat{k}} - \boldsymbol{m}_{q\bar{k}}\|} - d(\boldsymbol{x}_r, i_r;\boldsymbol{\Lambda})\frac{\boldsymbol{m}_{i_r\widehat{k}} - \boldsymbol{m}_{q\bar{k}}}{\|\boldsymbol{m}_{i_r\widehat{k}} - \boldsymbol{m}_{q\bar{k}}\|^2}\right] \tag{4-26}$$

$$\frac{\partial l(\boldsymbol{x}_r, i_r; \boldsymbol{\Lambda})}{\partial \boldsymbol{m}_{q\bar{k}}} = \alpha l(1-l)\left[\frac{\boldsymbol{x}_r - \boldsymbol{m}_{q\bar{k}}}{\|\boldsymbol{m}_{i_r\widehat{k}} - \boldsymbol{m}_{q\bar{k}}\|} - d(\boldsymbol{x}_r, i_r; \Lambda)\frac{\boldsymbol{m}_{q\bar{k}} - \boldsymbol{m}_{i_r\widehat{k}}}{\|\boldsymbol{m}_{q\bar{k}} - \boldsymbol{m}_{i_r\widehat{k}}\|^2}\right] \tag{4-27}$$

本章中模型的训练采用的是改进型的 Quickprop 算法 [13] 和 Rprop 算法。这两种方法的共同点是所有的训练样本处理完毕求得 $\frac{\partial \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda})}{\partial \boldsymbol{m}_{ik}}$ 后再对模型进行优化。通过分析可以发现，不同样本的偏导数求算过程互不影响，因此可以使用高性能计算平台在不同机器上对不同样本的导数进行求算，求算完毕后，调用平台的应用程序接口得到 $\frac{\partial \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda})}{\partial \boldsymbol{m}_{ik}}$。

4.3 优化算法

4.3.1 改进型 Quickprop 算法

Quickprop 算法在 [16] 中提出，是一种并不严格基于牛顿法的方法，它利用了二阶偏导和对损失函数曲面的估计信息，因此能够达到比较快的学习速度。本章的模型训练采用是一种引入了重启机制的改进型 Quickprop 算法，流程如下：

1. 令 t=0，计算 $\ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda})$ 关于 m_{ikd} 的偏导，然后用下式更新 m_{ikd} 的值

$$m_{ikd}^{(t+1)} = m_{ikd}^{(t)} - \varepsilon_0 \frac{\partial \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda}^{(t)})}{\partial m_{ikd}} \tag{4-28}$$

   m_{ikd} 为 $\boldsymbol{m}_{ik}$ 的第 d 维元素，$\frac{\partial \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda}^{(t)})}{\partial m_{ikd}} \triangleq \frac{\partial \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda})}{\partial m_{ikd}}|_{\boldsymbol{\Lambda}=\boldsymbol{\Lambda}^{(t)}}$，$\varepsilon_0$ 是初始的学习速率，根据经验设定。

2. 令 $t \leftarrow t+1$，计算 $\ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda})$ 关于 m_{ikd} 的二阶近似偏导：

$$\frac{\partial^2 \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda}^{(t)})}{\partial m_{ikd}^2} \approx \frac{\frac{\partial \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda}^{(t)})}{\partial m_{ikd}} - \frac{\partial \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda}^{(t-1)})}{\partial m_{ikd}}}{m_{ikd}^{(t)} - m_{ikd}^{(t-1)}} \tag{4-29}$$

3. 计算各参数的更新步长：

   - 如果 $\frac{\partial^2 \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda}^{(t)})}{\partial m_{ikd}^2} > 0$ 且 $\frac{\partial \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda}^{(t)})}{\partial m_{ikd}}$ 与 $\frac{\partial \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda}^{(t-1)})}{\partial m_{ikd}}$ 异号，采用牛顿法求得更新步长为：

$$\delta_t m_{ikd} = -\frac{\partial \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda}^{(t)})}{\partial m_{ikd}} \Big/ \frac{\partial^2 \ell(\mathcal{X},\mathcal{I};\boldsymbol{\Lambda}^{(t)})}{\partial m_{ikd}^2} \tag{4-30}$$

- 如果 $\frac{\partial^2 \ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda}^{(t)})}{\partial m_{ikd}^2} > 0$ 且 $\frac{\partial \ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda}^{(t)})}{\partial m_{ikd}}$ 与 $\frac{\partial \ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda}^{(t-1)})}{\partial m_{ikd}}$ 同号，则采用改进型的牛顿法计算更新步长:

$$\delta_t m_{ikd} = -\left(1/\frac{\partial^2 \ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda}^{(t)})}{\partial m_{ikd}^2} + \varepsilon_t\right)\frac{\partial \ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda}^{(t)})}{\partial m_{ikd}} \tag{4-31}$$

  其中 $\varepsilon_t = \varepsilon_0(1 - t/T_Q)$,$T_Q$ 为 Quickprop 的迭代步数。

- 如果 $\frac{\partial^2 \ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda}^{(t)})}{\partial m_{ikd}^2} < 0$ 或者 $\delta_t m_{ikd}$ 过小，直接采用梯度下降法：

$$\delta_t m_{ikd} = -\varepsilon_t \frac{\partial \ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda}^{(t)})}{\partial m_{ikd}} \tag{4-32}$$

4. 如果 $|\delta_t m_{ikd}| > |\text{limit} \times \delta_{t-1} m_{ikd}|$，将 $\delta_t m_{ikd}$ 重新赋值为 $\text{sign}(\delta_t m_{ikd}) \times \text{limit} \times |\delta_{t-1} m_{ikd}|$， limit 是一个控制参数，在本章的实验中设置为 1.75。本步骤的目的是为了保证4-29 的近似与实际情况差别较小。

5. 更新 m_{ikd}，即 $m_{ikd}^{(t+1)} = m_{ikd}^{(t)} + \delta_t m_{ikd}$

6. 重复步骤2到5 $T_Q - 2$ 次。

7. 重复步骤1到6 $T_R - 1$ 次

由算法流程可以看出，第1步为梯度下降方法，第2步到第5步为 Quickprop 过程。1 次梯度下降加 $T_Q - 1$ 次 Quickprop 过程称为一个循环，在重启新的循环之前随着 Quickprop 的进行，学习速率逐渐下降，当 T_Q 次迭代结束时，损失函数可能尚未收敛，所以引入重启机制以期获得更优的模型。

4.3.2 Rprop 算法

Rprop 是一种十分简单的优化算法，由 Martin Riedmiller 和 Heinrich Braun 在 1993 年提出 [17]，之后又出现了多种改进型 Rprop 算法 [18, 19, 20, 21, 22]。与传统的梯度下降方法不同，该方法只利用导数值的符号进行参数更新步长的调整。

记参数 m_{ikd} 在第 t 次迭代更新步长的大小为 $\Delta_{ikd}^{(t)}$，对所有 $\Delta_{ikd}^{(0)} = \Delta_0$，$\Delta_0$ 为一根据经验设定的正数。利用 Rprop 更新 $\Delta_{ikd}^{(t)}$ 的学习规则 [17] 为:

$$\Delta_{ikd}^{(t)} = \begin{cases} \eta^+ \Delta_{ikd}^{(t-1)} & \text{if } \frac{\partial \ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda}^{(t-1)})}{\partial m_{ikd}} \cdot \frac{\partial \ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda}^{(t)})}{\partial m_{ikd}} > 0 \\ \eta^- \Delta_{ikd}^{(t-1)} & \text{if } \frac{\partial \ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda}^{(t-1)})}{\partial m_{ikd}} \cdot \frac{\partial \ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda}^{(t)})}{\partial m_{ikd}} < 0 \\ \Delta_{ikd}^{(t-1)} & \text{else} \end{cases} \tag{4-33}$$

其中 $\eta_+ > 1$， $0 < \eta_- < 1$。为控制更新过程中的步长大小，还设定了两个额外的参数 $\Delta_{\max}$ 和 $\Delta_{\min}$。 m_{ikd} 的更新规则为

$$m_{ikd}^{(t+1)} = m_{ikd}^{(t)} - \text{sign}\left(\frac{\partial \ell(\mathcal{X}, \mathcal{I}; \mathbf{\Lambda}^{(t)})}{\partial m_{ikd}}\right) \cdot \Delta_{ikd}^{(t)} \tag{4-34}$$

各种改进型 Rprop 算法均是基于4-33和4-34，这些改进算法有的使用了损失函数的全局信息，有的使用了回退策略。本章总共使用了四种 Rprop 算法：Rprop-，Rprop+，iRprop-，iRprop+[19]，下面将一一介绍。

Rprop 算法流程如下：

1. 令 $t = 0$，设定 Δ_0、 $\Delta_{\max}$ 和 $\Delta_{\min}$, 此时 $\Delta_{ikd}^{(t)} = \Delta_0$，计算 $\ell(\mathcal{X}, \mathcal{I}; \mathbf{\Lambda})$ 关于 m_{ikd} 的偏导，然后用下式更新 m_{ikd} 的值: $m_{ikd}^{(t+1)} = m_{ikd}^{(t)} - \text{sign}\left(\frac{\partial \ell(\mathcal{X}, \mathcal{I}; \mathbf{\Lambda}^{(t)})}{\partial m_{ikd}}\right) \cdot \Delta_{ikd}^{(t)}$

2. 令 $t \leftarrow t + 1$，记 $S = \frac{\partial \ell(\mathcal{X}, \mathcal{I}; \mathbf{\Lambda})}{\partial m_{ikd}}^{(t-1)} \cdot \frac{\partial \ell(\mathcal{X}, \mathcal{I}; \mathbf{\Lambda})}{\partial m_{ikd}}^{(t)}$ 通过下面的方法更新 m_{ikd} 的值。

   - 采用 Rprop-算法

     如果 $S > 0$，

     $$\Delta_{ikd}^{(t)} = \min(\Delta_{ikd}^{(t)} \cdot \eta+, \Delta_{max})$$

     如果 $S < 0$，

     $$\Delta_{ikd}^{(t)} = \max(\Delta_{ikd}^{(t)} \cdot \eta-, \Delta_{min})$$

     否则

     $$\Delta_{ikd}^{(t)} = \Delta_{ikd}^{(t-1)}$$

     最后

     $$m_{ikd}^{(t+1)} = m_{ikd}^{(t)} - \text{sign}\left(\frac{\partial \ell(\mathcal{X}, \mathcal{I}; \mathbf{\Lambda}^{(t)})}{\partial m_{ikd}}\right) \cdot \Delta_{ikd}^{(t)}$$

   - 采用 Rprop+ 算法

     如果 $S > 0$，

     $$\Delta_{ikd}^{(t)} = \min(\Delta_{ikd}^{(t)} \cdot \eta+, \Delta_{max})$$

     $$\Delta m_{ikd}^{(t)} = -\text{sign}\left(\frac{\partial \ell(\mathcal{X}, \mathcal{I}; \mathbf{\Lambda})}{\partial m_{ikd}}^{(t)}\right) \cdot \Delta_{ikd}^{(t)}$$

     $$m_{ikd}^{(t+1)} = m_{ikd}^{(t)} + \Delta \boldsymbol{m}_{ikd}^{(t)}$$

如果 $S < 0$，

$$\Delta_{ikd}^{(t)} = \max(\Delta_{ikd}^{(t-1)} \cdot \eta-, \Delta_{min})$$
$$m_{ikd}^{(t+1)} = m_{ikd}^{(t)} - \Delta m_{ikd}^{(t-1)}$$
$$\frac{\partial \ell(\mathcal{X}, \mathcal{I}; \mathbf{\Lambda})}{\partial m_{ikd}}^{(t)} = 0$$

否则

$$\Delta_{ikd}^{(t)} = \Delta_{ikd}^{(t-1)}$$
$$\Delta m_{ikd}^{(t)} = -\text{sign}\left(\frac{\partial \ell(\mathcal{X}, \mathcal{I}; \mathbf{\Lambda})}{\partial m_{ikd}}^{(t)}\right) \cdot \Delta_{ikd}^{(t)}$$
$$m_{ikd}^{(t+1)} = m_{ikd}^{(t)} + \Delta m_{ikd}^{(t)}$$

- 采用 iRprop-算法

  如果 $S > 0$，

  $$\Delta_{ikd}^{(t)} = \min(\Delta_{ikd}^{(t)} \cdot \eta+, \Delta_{max})$$

  如果 $S < 0$，

  $$\Delta_{ikd}^{(t)} = \max(\Delta_{ikd}^{(t)} \cdot \eta-, \Delta_{min})$$
  $$\frac{\partial \ell(\mathcal{X}, \mathcal{I}; \mathbf{\Lambda})}{\partial m_{ikd}}^{(t)} = 0$$

  否则

  $$\Delta_{ikd}^{(t)} = \Delta_{ikd}^{(t-1)}$$

  最后

  $$m_{ikd}^{(t+1)} = m_{ikd}^{(t)} - \text{sign}\left(\frac{\partial \ell(\mathcal{X}, \mathcal{I}; \mathbf{\Lambda}^{(t)})}{\partial m_{ikd}}\right) \cdot \Delta_{ikd}^{(t)}$$

- 采用 iRprop+ 算法

  如果 $S > 0$，

  $$\Delta_{ikd}^{(t)} = \min(\Delta_{ikd}^{(t)} \cdot \eta+, \Delta_{max})$$
  $$\Delta m_{ikd}^{(t)} = -\text{sign}\left(\frac{\partial \ell(\mathcal{X}, \mathcal{I}; \mathbf{\Lambda})}{\partial m_{ikd}}^{(t)}\right) \cdot \Delta_{ikd}^{(t)}$$
  $$m_{ikd}^{(t+1)} = m_{ikd}^{(t)} + \Delta \boldsymbol{m}_{ikd}^{(t)}$$

如果 $S<0$，

$$\Delta_{ikd}^{(t)}=\max(\Delta_{ikd}^{(t-1)}\cdot\eta-,\Delta_{min})$$

$$\frac{\partial\ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda})}{\partial m_{ikd}}^{(t)}=0$$

如果$\ell^{(t)}>\ell^{(t-1)}$

$$m_{ikd}^{(t+1)}=m_{ikd}^{(t)}-\Delta m_{ikd}^{(t-1)}$$

否则

$$\Delta_{ikd}^{(t)}=\Delta_{ikd}^{(t-1)}$$

$$\Delta m_{ikd}^{(t)}=-\text{sign}\left(\frac{\partial\ell(\mathcal{X},\mathcal{I};\mathbf{\Lambda})}{\partial m_{ikd}}^{(t)}\right)\cdot\Delta_{ikd}^{(t)}$$

$$m_{ikd}^{(t+1)}=m_{ikd}^{(t)}+\Delta m_{ikd}^{(t)}$$

3. 重复第2步 $T-2$ 次。 T 为总的迭代次数（含第1步）。

从算法流程可以看出，Rprop 算法简单易实现，4.4将通过实验检验 Rprop 的性能。

4.4 实验结果

4.4.1 实验数据集介绍

本章实验全部在简体中文数据集上完成。该数据集包含 9,282 个字符，其中 9,119 个为简体中文字符，62 个为字母和数字，101 个为标点符号等其他字符。

训练数据集的样本总数为 9,795,394, 这些样本是基于 8 方向信息 [23] 的 128 维向量。由于该数据集上书写正规的字比书写潦草的字样本要多，为了防止训练得到的模型只在书写正规的字上有好的识别结果，我们将部分书写潦草的字样本数量加倍。这样处理过后，训练样本总数达 12,353,363。在全部的 9,282 个字符中，常用的有 3755 个，在实验中，这些字符的原型个数设为 2，其余设为 1。

实验中使用的测试集有六个，具体信息见表4.1。 现对测试集的名称简单说明：

表 4.1: 测试集概况

测试集	CurBoxNat	CurBoxUni1234	PriBoxNat	PriBoxRare	PriBoxUni	PrintRareBoxed2
字符总数	3,323	3,755	2,652	2,355	6,890	6,503
样本总数	51,238	383,064	103,104	84,549	96,904	36,441

- Cur：书写潦草的字符样本；
- Pri、Print: 书写正规的字符样本；
- Uni：样本平均地选取；
- Nat：样本自然地选取；
- Rare：生僻字符样本；
- Box、Boxed: 样本字符通过手写输入框取得。

此外在实验中，损失函数4-25的参数 α 设为 7.0， β 设为 0。Rprop 算法的部分参数取值为 $\Delta_{\min} = 1\mathrm{e}-6$， $\Delta_{\max} = 50$， $\eta+ = 1.2$， $\eta- = 0.5$。

4.4.2 四种 Rprop 算法的训练结果

本组实验中，取 $\Delta_0 = 0.05$，迭代步数设为 100 ，比较 4 种 Rprop 算法的性能。随着实验的进行，训练集上样本的识别准确率变化如图4.3所示。

由图4.3可以看出，iRprop-算法在收敛速率和识别准确率上略优于其他三种算法。经过 100 轮迭代得到的模型在测试集上的识别结果见表4.2。

表中的 LBG 指的是由 LBG 算法得到的初始模型。由表4.2可以看出，在 5 个测试集上，iRprop-的表现要优于其他 Rprop 算法。注意到在 PriBoxRare 测试集上，四种 Rprop 算法得到的模型均比 LBG 初始化的模型性能要差，原因在于 PriBoxRare 为生僻字集，这类字符的训练样本较少，导致对其参数的优化不足，而对其他训练样本多的字符所做的较多优化影响到了模型在生僻字集上的识别效果。

从训练集和测试集上的结果来看，iRprop-算法总体上要优于其他三种，接下来的实验将围绕 iRprop-展开。

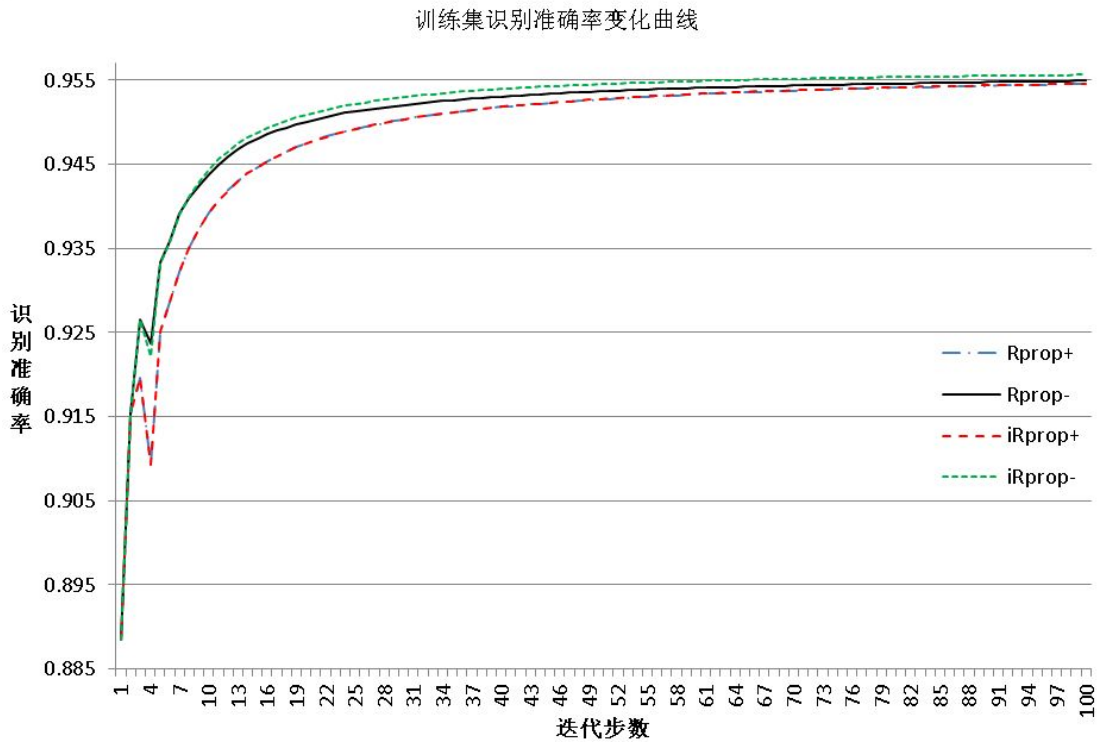

图 4.3: 四种 Rprop 算法训练集识别准确率变化曲线

表 4.2: 四种 Rprop 算法优化的模型在测试集上的识别准确率（%）

测试集	CurBoxNat	CurBoxUni1234	PriBoxNat	PriBoxRare	PriBoxUni	PrintRareBoxed2
LBG	84.35	87.62	88.36	**96.61**	94.22	82.84
Rprop-	89.73	92.86	92.11	96.22	97.30	87.93
Rprop+	89.72	92.82	**92.17**	96.22	97.27	87.91
iRprop-	**89.81**	**92.89**	92.11	**96.24**	**97.32**	**87.95**
iRprop+	89.64	92.81	91.83	96.20	97.26	87.93

4.4.3 初始步长对 iRprop-算法的影响

在本节的实验中，我们改变 Δ_0 的值，观察 iRprop-训练得到的模型在训练集和测试集上表现的变化情况。

在训练集上，100 轮迭代过程中，识别准确率变化曲线如图4.4所示。

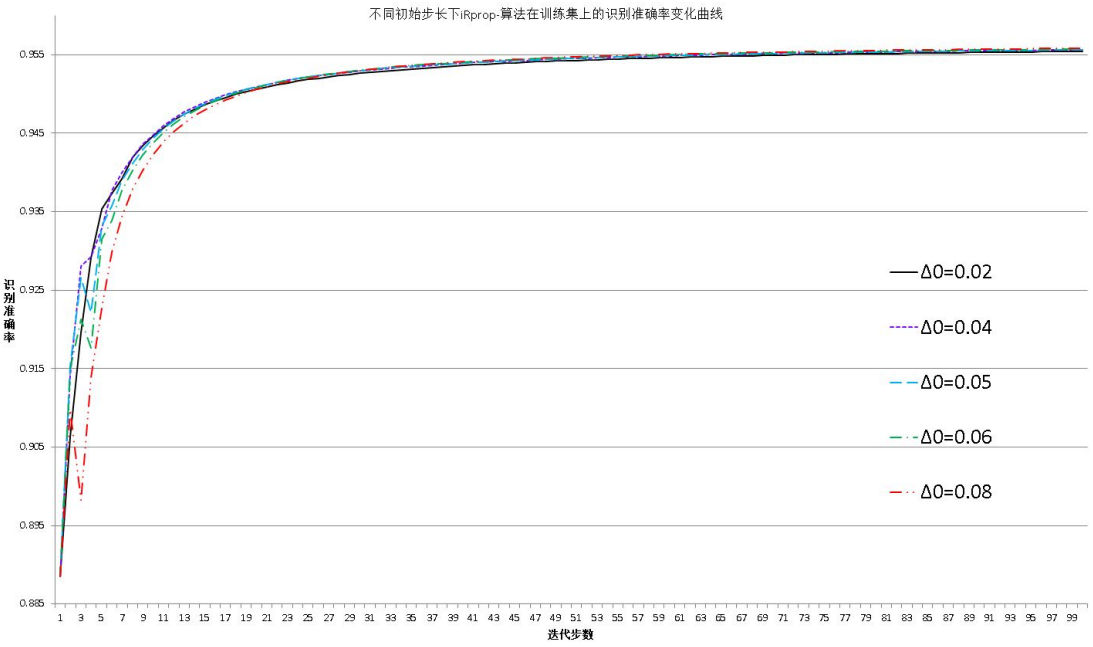

图 4.4: 不同 Δ_0 对 iRprop-算法影响示意图

由图可以看出， Δ_0 的不同除影响最初几步的准确率变化，对收敛速度和最终的识别准确率影响不大。

训练结束后，在测试集上对模型性能测试的结果见表4.3。 从表4.3可以看

表 4.3: 在不同 Δ_0 下 iRprop-算法在测试集上的识别准确率（%）

测试集	CurBoxNat	CurBoxUni1234	PriBoxNat	PriBoxRare	PriBoxUni	PrintRareBoxed2
LBG	84.35	87.62	88.36	96.61	94.22	82.84
$\Delta_0 = 0.02$	**89.84**	92.90	91.94	96.22	97.31	**88.02**
$\Delta_0 = 0.04$	89.80	92.88	92.21	96.22	97.32	87.97
$\Delta_0 = 0.05$	89.81	92.89	92.11	**96.24**	97.32	**88.02**
$\Delta_0 = 0.06$	89.73	92.91	**92.25**	96.23	**97.36**	**88.02**
$\Delta_0 = 0.08$	89.91	**92.93**	91.91	96.21	97.33	87.91

出，在 5 种不同的 Δ_0 下，模型的性能差别不大。综合训练集和测试集上的结果，我们可以得出，iRprop- 是一种鲁棒性很好的优化算法。

4.4.4 iRprop-与改进型 Quickprop 训练结果对比

在 [13] 中，改进型 Quickprop 算法在简体中文数据集上取得了理想的识别效果。现将 iRprop-方法与其进行简单比较。

iRprop-的 Δ_0 取 0.05，迭代步数为 100，改进型 Quickprop 算法 $T_Q = 20$，$T_R = 5$， $\varepsilon_0 = 0.1$，在训练过程中，训练集上的识别准确率变化如图4.5所示。

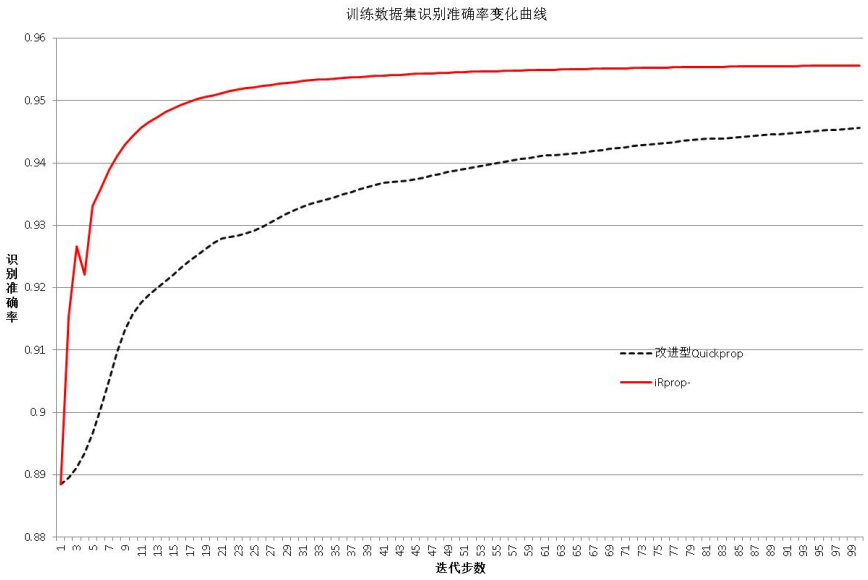

图 4.5: iRprop-算法与改进型 Quickprop 算法训练效果对比

从图4.5可以看出，iRprop-的准确率和收敛速度明显快于改进型 Quickprop。经过训练得到的模型在测试集上的识别结果见表4.4。 iRprop-在测试集

表 4.4: iRprop-算法与改进型 Quickprop 算法测试集识别准确率对比（%）

测试集	CurBoxNat	CurBoxUni1234	PriBoxNat	PriBoxRare	PriBoxUni	PrintRareBoxed2
LBG	84.35	87.62	88.36	96.61	94.22	82.84
改进型 Quickprop	88.98	92.11	91.47	96.08	97.01	87.17
iRprop-	89.81	92.89	92.11	96.24	97.32	88.02

上的表现也是优于改进型 Quickprop 的。

在图4.5中，经过 5×20 次迭代，改进型 Quickprop 尚未收敛，这在一定程度上影响了结果的可信度。为得到更为令人信服的结果，令 $T_Q = 100$，$T_R = 3$，重新进行模型训练。3 次循环的初始学习速率 ε_0 分别设为 0.1，0.08 和 0.05，得到的在训练集上的识别准确率变化曲线如图4.6所示。从曲线可以看出，即使经过了 300 次参数优化改进型 Quickprop 算法在训练集上的表现依然不及 iRprop-算法。

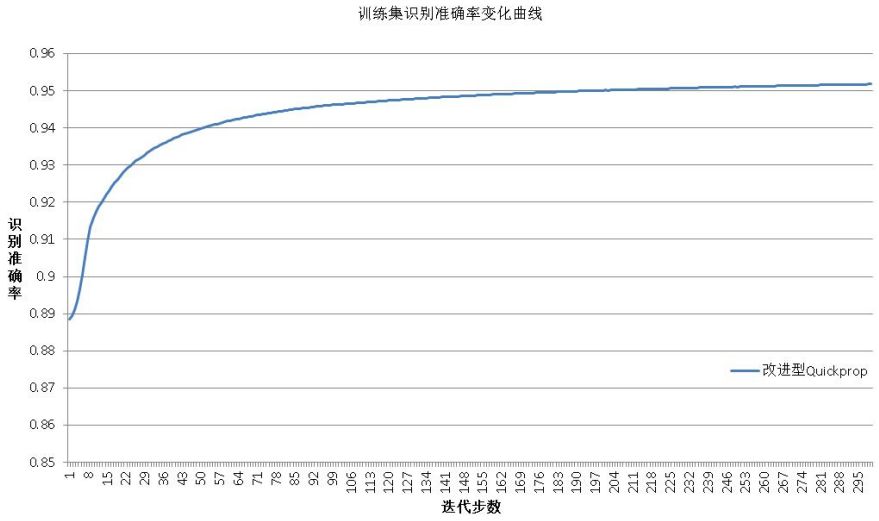

图 4.6: 改进型 Quickprop 算法训练效果示意图

利用由此得到的模型在测试集上进行测试，得到的结果见表4.5。 对比

表 4.5: 改进型 Quickprop 算法测试集识别准确率（%）

测试集	CurBoxNat	CurBoxUni1234	PriBoxNat	PriBoxRare	PriBoxUni	PrintRareBoxed2
改进型 Quickprop	89.47	92.59	91.79	96.19	97.20	87.71

表4.5与4.4，iRprop-在测试集上的性能也优于改进型 Quickprop[1]。

综合上述一系列的描述及实验可以发现，Rprop 算法是一种简单、鲁棒、性能出色的模型优化算法。在今后解决类似问题时，可以考虑采用这种方法。

[1]利用文章 [13] 中实验得到的模型进行测试，改进型 Quickprop 性能要好于本章的实验，但与 iRprop-相比，结果依然稍差。

上述每次实验均使用 3 台 8 核机器，迭代一轮只需 1 分钟，大大提高了工作效率。

第 5 章　总结与展望

本次毕业设计主要研究了若干机器学习算法在高性能计算平台上的实现和性能，总结起来我主要的工作和收获如下：

- 掌握了使用高性能计算平台实现机器学习算法的技巧。
- 通过查阅资料，学习了 LBG 算法、基于期望最大化算法的似然率最大化高斯混合模型训练算法、最小分类误差训练方法和样本分离边距的基本思想，掌握了基于梯度的模型优化算法 Rprop 和改进型 Quickprop。
- 通过编程实践，检验了上述机器学习算法在大规模数据集上的性能，同时提高了自己编写 C++ 代码的能力。
- 通过一系列对比实验，验证了 Rprop 算法应用于多原型手写识别模型训练的优越性能，为后续的研究工作提供了参考。

由于本人能力和时间的限制，本次毕业设计尚有诸多可以改进的地方，在未来的工作中可以尝试实现，比如:

- 改进 LBG 算法中聚类中心点的分裂策略，以减弱中心点分裂后对聚类性能的不良影响。
- 在高斯混合模型训练过程中，尝试改进协方差矩阵取值下界的设定和取值越界后的处理方法，以提高模型性能。
- 在多原型手写识别模型训练过程中，尝试将多种不同的优化算法相结合，并在数据处理时引入随机扰动，以期得到识别准确率更高的模型。

经过这次毕业设计工作的锻炼，我对科学研究从一知半解到略窥门径，初步掌握了做研究的方法，在理论学习和工程实践方面都有了很大的提高。这段时间学到的东西为我未来的学习和工作打下了坚实的基础。

参考文献

[1] 中国互联网络信息中心, “中国互联网络发展状况统计报告,” 2011.

[2] Y. Linde, A. Buzo, and R. Gray, “An algorithm for vector quantizer design,” *IEEE Trans. Comunications*, vol. 28, no. 1, pp. 84–95, 1980.

[3] A. K. Jain, “Data clustering: 50 years beyond k-means,” *Pattern Recognition Letters*, vol. 31, pp. 651–666, 2009.

[4] J. MacQueen, “Some methods for classification and analysis of multivariate obeservations,” in *Proc. the 5th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, 1967.

[5] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. John Wiley & Sons,Inc., 2001.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[7] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, “Clustering with bregman divergences,” in *SIAM Int. Conf. Data Mining*, 2005.

[8] B.-H. Juang, W. Chou, and C.-H. Lee, “Minimum classification error rate methods for speech recognition,” *IEEE Trans. Speech and Audio Processing*, vol. 5, no. 3, pp. 257–265, 1997.

[9] S. Amari, “A Theory of Adaptive Pattern Classifiers,” *IEEE Trans. Electronic Computers*, no. 3, pp. 299–307, 1967.

[10] B.-H. Juang and S. Katagiri, “Discriminative learning for minimum error classification,” *IEEE Trans. Signal Processing*, vol. 40, no. 12, pp. 3043–3054, 1992.

[11] Y. Wang and Q. Huo, “Sample-separation-margin based minimum classification error training of pattern classifiers with quadratic discriminant functions,” in *Proc. IEEE Int. Conf. Acoustics Speech and Signal Processing (ICASSP)*, pp. 1866–1869, 2010.

[12] T. He and Q. Huo, "A study of a new misclassification measure for minimum classification error training of prototype-based pattern classifiers," in *Proc. 19th Int. Conf. Pattern Recognition ICPR 2008*, pp. 1–4, 2008.

[13] Y. Wang and Q. Huo, "A Study of Designing Compact Recognizers of Handwritten Chinese Characters Using Multiple-Prototype Based Classifiers," in *Proc. 20th Int. Conf. Pattern Recognition (ICPR)*, pp. 1872–1875, 2010.

[14] S. Katagiri, B.-H. Juang, and C.-H. Lee, "Pattern recognition using a family of design algorithms based upon the generalized probabilistic descent method," *Proc. the IEEE*, vol. 86, no. 11, pp. 2345–2373, 1998.

[15] W. Chou, B.-H. Juang, and C.-H. Lee, "Segmental gpd training of hmm based speech recognizer," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing ICASSP-92.*, vol. 1, pp. 473–476, 1992.

[16] S. E. Fahlman, "An empirical study of learning speed in back-propragation networks," Tech. Rep. CMU-CS-88-162, Carnegie Mellon University, 1988.

[17] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm," in *Proc. IEEE Int. Conf. Neural Networks*, pp. 586–591, 1993.

[18] C. Igel and M. Hüsken, "Empirical evaluation of the improved Rprop learning algorithms," *Neurocomputing*, vol. 50, pp. 105–123, 2003.

[19] C. Igel and M. Hüsken, "Improving the Rprop Learning Algorithm," in *ICSC Symposium on Neural Computation*, pp. 115–121, ICSC Academic Press, 2000.

[20] A. D. Anastasiadis, G. D. Magoulas, and M. N. Vrahatis, "Sign-based learning schemes for pattern classification," *Pattern Recognition Letters*, vol. 26, pp. 1926–1936.

[21] A. D. Anastasiadis, G. D. Magoulas, and M. N. Vrahatis, "An Efficient Improvement of the Rprop Algorithm," in *Proc. the 1st International Work-*

shop on Artificial Neural Networks in Pattern Recognition (IAPR 2003), pp. 197–201, 2003.

[22] A. D. Anastasiadis, G. D. Magoulas, and M. N. Vrahatis, "Improved sign-based learning algorithm derived by the composite nonlinear Jacobi process," *Journal of Computational and Applied Mathematics*, vol. 191, pp. 166–178, 2006.

[23] Z.-L. Bai and Q. Huo, "A study on the use of 8-directional features for online handwritten chinese character recognition," in *Proc. 8th Int. Conf. Document Analysis and Recognition*, pp. 262–266, 2005.

[24] R. Rojas, *Neural Networks: A Systematic Introduction*. Springer, 1996.