HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTER THESIS

Ms.
Tejaswini Devineni

Transfer Learning / Offset-Learning for Learning Vector Quantization

2022

Faculty of **Applied Computer Sciences and Biosciences**

MASTER THESIS

Transfer Learning / Offset-Learning for Learning Vector Quantization

Author:
Tejaswini Devineni

Study Programme:
Applied Mathematics in Networking and DataScience

Seminar Group:
MA18w1-M

First Referee:
Prof. Dr. Thomas Villmann

Second Referee:
Dr. Marika Kaden

Mittweida, January 2022

Acknowledgement

First and foremost I wish to express my sincere gratitude to Prof. Dr. Thomas Villmann and Dr. Marika Kaden at the University of Applied Sciences Mittweida for their tremendous support, indispensable ideas and invaluable advice throughout my thesis.

I am thankful to all the professors that helped me improve myself both technically and socially, so that I could finally be able to successfully finish my studies. I am very thankful to all my friends especially Sunil Kumar Devineni, Sowjanya Yalamanchili and Hanumantha Rao Malapati for their constant support and continuous encouragement during my studies.

And most of all, I must express my very profound gratitude and dedicate my thesis to my parents, brother and family members who have always been a unfailing support throughout my years of study. This accomplishment would not have been possible without them.

Bibliographic Information

Devineni, Tejaswini: Transfer Learning / Offset-Learning for Learning Vector Quantization, 41 pages, 10 figures, Hochschule Mittweida, University of Applied Sciences, Faculty of Applied Computer Sciences and Biosciences

Master Thesis, 2022

Abstract

In Machine Learning, Learning Vector Quantization(LVQ) is well known as supervised learning method. LVQ has been studied to generate optimal reference vectors because of its simple and fast learning algorithm [12]. In many tasks of classification, different variants of LVQ are considered while training a model. In this thesis, the two variants of LVQ, Generalized Matrix Learning Vector Quantization(GMLVQ) and Generalized Tangent Learning Vector Quantization(GTLVQ) have been discussed. And later, transfer learning technique for different variants of LVQ has been implemented, visualized and we have compared the results using different datasets.

I. Contents

II. List of Figures

III. List of Tables

1 Introduction

1.1 Background

Machine Learning (ML) has steadily been acquiring a very important role in industry as well as individual lifestyle. In Machine Learning, there are different type of learning schemes such as supervised Learning, unsupervised learning and reinforcement learning. In this thesis, we will concentrate on prototype-based classification which is a part of supervised learning. Prototype-based classification is playing a vital role to represent the data by a set of prototypes. Particularly, in case of big data or complicate classification problems, it is still a challenging task for the classification. Suppose a model is been trained according to initial training data and then the new data becomes available now for the training, we either have to train all the data including with the new data or we have to train this new data separately. In which it takes so much time and consumes more memory for training all the data again which includes new data. Re-training of data may also cause the problem of information loss and this problem cannot be neglected if the new data has been added differs from the initial data. So, in order to avoid all such problems, we use the concept of transfer learning. The main idea of transfer learning is, the information that has been learned from the training data is kept same and we need to apply some transformation to new data in such a way that the initial model can be applied in order to train or learn the new data (transfer data) [14].

In this thesis, transfer learning for Learning Vector Quantization has been studied which was proposed by Villmann and Saralajew along with the different variants of Learning Vector Quantization models. We have applied transfer learning for both GMLVQ and GTLVQ in this thesis and compared the results. The work has been performed by taking synthetic data and a part of MNIST data. We have trained both the models using these two datasets in the further sections.

1.2 The Structure of the Thesis

The structure of the thesis is organized as follows:

In chapter 2, we discuss about the basic concepts of Machine Learning and Prototype-based classification algorithms such as Learning Vector Quantization. We will use these terms very frequently in this project. Later, we will discuss about the realizations of LVQ such as Kohonen's LVQ algorithms, GLVQ and GMLVQ.

In chapter 3, we concentrate on the explanation of implementing tangent distance for GLVQ which is called as Generalized Tangent Learning Vector Quantization. Initially we

will explain few definitions required for GTLVQ and later we will explain the algorithm of GTLVQ.

In Chapter 4, we first deal with the explanation of transfer learning. Later, we discuss about transfer learning based on concept of Functions and Manifolds. Further, we elaborate how these concept of functions and manifolds are implemented in GMLVQ and GTLVQ.

In chapter 5, we will see the findings of my thesis. We can see the results of GMLVQ and GTLVQ and transfer learning for both GMLVQ and GTLVQ perform on the respective datasets and then we come to the conclusion which techniques performed best in which kind of situation.

2 Learning Vector Quantization

Learning Vector Quantization (LVQ) introduced by Kohonen [8] is a supervised classification algorithm which works on the concept of nearest prototype classifier. To calculate the nearest prototype, dissimilarities has been used. At present, there are many variants of LVQ, each introduced and designed for distinct applications or objectives. We start with the common LVQ concept overview and later, we describe Generalized Learning Vector Quantization (GLVQ) and Generalized Matrix Learning Vector Quantization (GMLVQ). The reason behind many variants is that the dissimilarity have to be selected based on the classification task.

2.1 Basics of Machine Learning

Machine Learning is a subset of Artificial Intelligence (AI) that provides the AI System with the ability to automatically learn from the data and applies that learning to make better decisions. That is, choosing an algorithm depending on the data or the available information to imitate the way that humans learn, gradually applying this learning to improve the accuracy of the model [4]. These algorithms in machine learning are categorized into supervised, unsupervised, reinforcement learning and semi-supervised learning.

Supervised Learning is a machine learning task, where we have input and output pairs. We use an algorithm to learn a function that maps an input to the output and in general, the inputs are in the form of vectors. The purpose of supervised learning model is to predict the correct label for a newly provided unseen data. But in order to predict the correct label, the model must be trained in such a way that the model tries to predict the correct output. The model compares it to the given label in order to calculate the loss. To minimize the loss, it is mandatory to change the parameters of model [9].

Unsupervised Learning is a machine learning task where we have only input data and there will be no corresponding output variables. The main aim of unsupervised learning is about discovering the unknown relationships and structures in the data in order to learn much about the given data. Although unsupervised learning can perform complex task than supervised learning, they can be more unpredictable [5].

Gradient Descent is an iterative method and the objective is to minimize a cost function. It should be possible to compute the partial derivative of the function which is slope or gradient. The coefficients are computed at each iteration by taking batch of training samples and taking the negative of the derivative and by reducing the coefficients at each step by a learning rate multiplied by derivative. So that the local minima can

be achieved after a few iterations. So eventually the iterations are stopped when it converges to minimum value of the cost function after which there is no further reduction in cost function.

The gradient descent that we use in GLVQ, GMLVQ and GTLVQ is **Stochastic Gradient Decent** (SGD). In SGD, error is calculated for each training sample within the dataset and parameters are updated for every training sample. Given a training dataset $\mathcal{T}$ of labeled inputs $(v, c(v))$ where $c(v)$ is the correct label of v and a learning rate $\eta \in \mathbb{R}_{>0}$ and a loss function $l(f(v;\vartheta), c(v))$, the update rule in SGD is given as

$$\vartheta \longleftarrow \vartheta - \eta \nabla_{\vartheta} l(f(v;\vartheta), c(v)) \tag{2.1}$$

2.2 Basics of Learning Vector Quantization

Although there are many variants of LVQ algorithms, the basic concept always remains same, that is, to use the concept of prototypes and dissimilarities. So, we firstly define the concept of dissimilarities and prototypes similar to the explanation given by Sascha Saralajew [12].

2.2.1 Dissimilarities

We use the concepts of distance greatly in our daily life: For example, to illustrate how much distance is present in between two points or for the travel, such as we calculate the average time of travel and distance between two places. Typically, the most fundamental part of prototype-based classification is the concept of mathematical distance (mathematical metric). The distance is calculated in between the arbitrary objects and is restricted to neither scalars nor points.

Definition 2.1 (metric and metric space). The distance (metric) d on a non-empty set S is a function that maps to real numbers as follows:

$$d : S \times S \to \mathbb{R}$$

which should satisfy the below conditions for all $a, b, c \in S$:

- $d(a,b) \geq 0$ (nonnegativity);
- $d(a,b) = 0$ if and only if $a = b$ (identity);
- $d(a,b) = d(b,a)$ (symmetry);
- $d(a,b) \leq d(a,c) + d(c,b)$ (triangle inequality).

The metric space (S,d) is a set S with a metric d defined on S. The function d is also called as distance function or commonly distance(metric).

Definition 2.2 (translation-invariant metric). Let (S,d) be a metric space with an operation

$$+ : S \times S \to S$$

that forms a group $(S,+)$ on S. If we consider the operation $+$ is a vector shift, then the translation-invariant can be explained as the distances remains same although the points a and b are shifted in the same direction. That is, the translation-invariant is true on metric d if

$$d(a,b) = d(a+c,b+c) \tag{2.2}$$

for all $a,b,c \in S$.

A dissimilarity is obtained by dropping basic axiom of a metric(distance), i.e. nonnegativity [11]. The most often used distance measure is the Euclidean distance which was named after the Greek mathematician Euclid. This Euclidean distance calculates the length of the line segment between two considered points in Euclidean space.

Definition 2.4 (Euclidean distance). Let a and b be the vectors of n-dimensional real vector space $\mathbb{R}^n$. The Euclidean distance can be defined as

$$d_E(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

and in terms of vector operations as

$$d_E(a,b) = \sqrt{(a-b)^T(a-b)} \tag{2.3}$$

Euclidean distance is both the metric and translation-invariant metric. In this thesis, the generalization of the Euclidean distance that is used is the Mahalanobis distance. This distance can model the proper distance measure in case if we have the knowledge about the underlying data distribution.

Definition 2.3 (Quasimetric). The quasimetric is a function d that fulfills all the metric axioms except the triangle inequality. The squared Euclidean distance $d_E^2(a,b)$ becomes quasimetric.

Definition 2.5 (Mahalanobis distance). Let the two random vectors be $a,b \in \mathbb{R}^n$ are drawn from a probability distribution along with the full ranked covariance matrix Σ. Then, the Mahalanobis distance can be defined as

$$d_M(a,b) = \sqrt{(a-b)^T\Sigma^{-1}(a-b)} \tag{2.4}$$

where Σ^{-1} is called the precision matrix and the precision matrix is positive definite and

symmetric.

In general, if the covariance matrix is the identity matrix, then the Mahalanobis distance becomes Euclidean distance. The full rank Σ states that the covariance matrix is invertible. If we drop the assumption that the Σ as the full rank, then, we can explain the so-called quadratic-dissimilarity.

Definition 2.6 (quadratic-dissimilarity). Let the two vectors $a, b \in \mathbb{R}^n$ and $Q \in \mathbb{R}^{n \times m}$ be the transition matrix that does a linear mapping. The quadratic-dissimilarity can be explained as

$$d_Q(a,b) = \sqrt{(Q(a-b))^T Q(a-b)} \tag{2.5}$$

where m is a hyperparameter that finds the dimensionality of vector space after the linear mapping. Generally, whenever the rank of Q equals n, then the quadratic-dissimilarity is a metric. Same as Mahalanobis distance, the quadratic-dissimilarity calculates the Euclidean distance for renewed input vectors which can be denoted as

$$d_Q(a,b) = \sqrt{(a-b)^T \Lambda (a-b)} \tag{2.6}$$

where Λ equals $Q^T Q$.

2.2.2 Prototypes

One of the classification scheme is the prototype based classification. In general, prototype based classification uses the concept of dissimilarity by measuring distances. To be specific, in other words, prototypes are like centroids in data points. Prototypes of class c is calculated as the points that are away from other class data points and they should be near to many data points of same class c [7].

Prototype: A vector w_k of the data space $\mathbb{R}^n$ is said to be a prototype(vector) if there exists an element equipped with a class label $c(w_k) \in C$. The set W contains all the collected set of prototypes and M represents the overall number of prototypes.

$$W = \{w_k \in \mathbb{R}^n \,|\, k = 1, 2, ..., M\}, \tag{2.7}$$

Given a data point v, we can find the prototype w_k that is fit to the data point by using the dissimilarity measure d. The distance vector with the dissimilarity measure d with respect to the data point v is given by

$$d(v) = (d(v,w_1), d(v,w_2,), ..., d(v,w_M))^T \tag{2.8}$$

To compute the nearest prototype, we need to consider the smallest dissimilarity, that is, prototypes compete against each other to be the closest prototype w^*. For an input v, we can find the closest prototype using

$$w^*(v) = \underset{w_k \in W}{\arg\min}(d(v, w_k)) \tag{2.9}$$

and is also known as best matching prototype with respect to v.

For a input v, the predicted class is denoted as $c^*(v)$= $c(w^*)$. The main aim of prototype-based learning is to find the proper set of prototypes for a training dataset $\mathcal{T}$. According to Bien and Tibshirani [7], we have to find the set of prototypes such that:

- The prototypes number M should be as limited as possible.
- For a class c, the prototypes are the closest points of as many training data points as possible.
- For a class c, the prototypes are the closest points of as many training points as possible other than class c.

In the following sections, we will describe variants of LVQ. Firstly, we describe about Kohonen's LVQ algorithms. And next GLVQ algorithm that is a version of Kohonen's algorithm that includes differentiable loss function which uses Stochastic Gradient Descent. Later, we describe about the extended version of GLVQ known as GMLVQ, that makes use the concept of matrices.

2.3 Kohonen's Learning Vector Quantization algorithms

We suppose the dataset V = $\{v_i \in \mathbb{R}^n,\ i = 1, ..., m\}$ are the data vectors which has class labels $c(\mathrm{v}_i) \in C$ is the set of classes. Let the prototype set be W = { $w_j \in \mathbb{R}^n$, $j = 1, ..., M\}$ with the class labels $c(w_j)$. The main characteristic property of LVQ is, for each prototype w_k, a class label $c(w_k)$ is allotted in such a way that at least one prototype is exposed for each class. LVQ introduced by Kohonen as a heuristic scheme has different flavours like LVQ1, LVQ2, LVQ3 as described in [8]. In this section, let us concentrate the basic intention behind all the algorithms. We initialize the set of prototypes W randomly. After that, we iterate the below steps:

- From the training dataset, randomly select a data sample $(v, c(v))$ and find the nearest prototype w^* as in equation (2.9).
- If the class label $c(w^*)$ of the nearest prototype is equal to the class label $c(v)$ of the input, the nearest prototype w^* is pushed towards v (attraction) or otherwise prototype is pushed away (repulsion).

More precisely, the update of prototypes is defined as :

$$w^* \longleftarrow w^* - \eta \Delta w^* \tag{2.10}$$

where,

$$\Delta w^* = s(v - w^*) \tag{2.11}$$

with,

$$s = \begin{cases} -1, & c(w^*) = c(v) \\ 1, & c(w^*) \neq c(v) \end{cases} \tag{2.12}$$

and η is the learning rate ($\eta \in \mathbb{R}_{>0}$). This η controls the magnitude of the applied shift. Generally, all the variants mentioned above of LVQ uses the Euclidean distance d_E as in equation (2.2) with respect to prototype is given as

$$\Delta_w d_E(v,w) = \frac{-1}{d_E(v,w)}(v,w) \tag{2.13}$$

if $d_E(v,w) \neq 0$.

2.4 Generalized Learning Vector Quantization

To give generalization of Kohonen's LVQ algorithms, Sato and Yamada [18] has proposed GLVQ. Sato and Yamada have realized this by introducing differentiable cost function which is called as GLVQ loss. But the explanation of GLVQ given in this thesis is inspired from Sascha Saralajew [12]. As before, we suppose that the labeled training data $(v, c(v))$ and a set of labeled prototypes $(w_j, c(w_j))$ with $\{w\}_{j=1}^{M} = \mathrm{W}$. We use the following notations in GLVQ:

- w^+ is denoted as best matching correct prototype for $(v, c(v))$ within all the prototypes w_j with $c(w_j) = c(v)$.
- w^- is denoted as best matching incorrect prototype for $(v, c(v))$ within all the prototypes w_k with $c(w_k) \neq c(v)$.
- $d^+(v) = d(v, w^+)$, is the distance from v to w^+.
- $d^-(v) = d(v, w^-)$, is the distance from v to w^-.

In order to calculate the dissimilarity of a prototype to the correct class $c(v)$ given the classifier function $f(v)$ and a training sample $(v, c(v))$ as

$$d^+(v) = \min\{-f_{c(v)}(v) \,|\, c = c(v)\} \tag{2.14}$$

where $f(v)$ is the classifier function and is calculated as negating the smallest dissimilarity value of each class and mathematically written as

$$f(v) = -\begin{pmatrix} \min\{d_k(v)|\ (w_k) = 1\} \\ \min\{d_k(v)|\ (w_k) = 2\} \\ \cdot \\ \cdot \\ \cdot \\ \min\{d_k(v)|\ c(w_k) = \#C\} \end{pmatrix} \tag{2.15}$$

where $f_c(v)$ is the negative value of smallest dissimilarity of v to a prototype of class c. The smallest dissimilarity of a class to the prototype different than $c(v)$ of v by

$$d^-(v) = \min\{-f_c(v)\,|\,c \neq c(v)\} \tag{2.16}$$

Then, we define the classifier function $\mu(v)$ as

$$\mu(v) = \frac{d^+(v) - d^-(v)}{d^+(v) + d^-(v)} \in [-1, 1] \tag{2.17}$$

$\mu(v) < 0$ if v is correctly assigned to the class whereas $\mu(v) > 0$ informs about the incorrect classification. The function μ is differentiable with respect to w and in general, it has nontrivial gradients and is given as

$$\frac{\partial \mu(v)}{\partial w^+} = \frac{+2d^-(v)}{(d^+(v) + d^-(v))^2} \frac{\partial d(v, w^+)}{\partial w^+} \tag{2.18}$$

$$\frac{\partial \mu(v)}{\partial w^-} = \frac{-2d^+(v)}{(d^+(v) + d^-(v))^2} \frac{\partial d(v, w^-)}{\partial w^-} \tag{2.19}$$

Using the relative distance dissimilarity $\mu(v)$, one can compute classification error for the given training data set ($\mathcal{T}$) as in (2.20). f is parameterized by a parameter vector ϑ of trainable weights.

$$error(f(v, \vartheta), \mathcal{T}) = \frac{1}{\#\mathcal{T}} \sum_{(v, c(v)) \in \mathcal{T}} H(\mu(v)) \tag{2.20}$$

where ϑ is trainable parameter vector(also called weight) and $H(x)$ is a Heaviside step function which is not differentiable and is represented as

$$H(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.21}$$

As Heaviside function is non-differentiable, Sato and Yamada replaced it with monotonically increasing function $\phi : [-1, 1] \to \mathbb{R}$. All these together results as

$$l(f(v; \vartheta), c(v)) = \phi(\mu(v)) \tag{2.22}$$

where $l(f(v;\vartheta),c(v))$ is called as loss function or GLVQ loss function and $\phi(\mu)$ is a monotonically increasing function. Usually, the identity function $f(m)=m$ or the sigmoid function $f(m)=1/(1+e^{m\theta})$ is used. For a given training sample from $\mathcal{T}$ and a learning rate $\eta \in \mathbb{R}_{>0}$, SGD is given as

$$\vartheta \leftarrow \vartheta - \eta \nabla_{\vartheta} l(f(v;\vartheta),c(v)) \tag{2.23}$$

Training

Training sample $(v,c(v))$ have to be picked randomly and we need to perform stochastic gradient descent for $l(f(v;\vartheta),c(v))$ using (2.23). Because of $\mu(v)$, the loss function is dependent on w^+ and w^-, which are prototypes. w^+ and w^- are the closest prototypes of correct class ($d^+(v)$) and incorrect class ($d^-(v)$). We have to calculate the gradient with respect to w^+ and w^- as the derivatives of loss function are nonzero with respect to ϑ. We calculate the gradient for the loss function with respect to w^+ and w^- for given $(v,c(v))$ as

$$\frac{\partial}{\partial w^+}\phi(\mu(v)) = 2\phi'(\mu(v))\frac{d^-(v)}{(d^+(v)+d^-(v))^2}\frac{\partial d(v,w^+)}{\partial w^+}. \tag{2.24}$$

$$\frac{\partial}{\partial w^-}\phi(\mu(v)) = -2\phi'(\mu(v))\frac{d^+(v)}{(d^+(v)+d^-(v))^2}\frac{\partial d(v,w^-)}{\partial w^-}. \tag{2.25}$$

The above equations (2.24) and (2.25) can be also written as

$$\frac{\partial}{\partial w^+}\phi(\mu(v)) = \xi_\mu^+ \Delta w^+ \tag{2.26}$$

$$\frac{\partial}{\partial w^-}\phi(\mu(v)) = \xi_\mu^- \Delta w^- \tag{2.27}$$

where

$$\xi_\mu^+ = 2\phi'(\mu(v))\frac{d^-(v)}{(d^+(v)+d^-(v))^2} \geq 0 \tag{2.28}$$

$$\xi_\mu^- = 2\phi'(\mu(v))\frac{d^+(v)}{(d^+(v)+d^-(v))^2} \geq 0 \tag{2.29}$$

and

$$\Delta w^+ = +\frac{\partial d(v,w^+)}{\partial w^+} \tag{2.30}$$

$$\Delta w^- = -\frac{\partial d(v,w^-)}{\partial w^-} \tag{2.31}$$

The learning rule of the prototypes in Stochastic Gradient Descent(SGD) is denoted as

$$w^+ \longleftarrow w^+ - \eta\,\xi_\mu^+ \Delta w^+ \tag{2.32}$$

$$w^- \longleftarrow w^- - \eta\,\xi_\mu^- \Delta w^- \tag{2.33}$$

GLVQ algorithm implemented by Sato and Yamada makes use of squared Euclidean distance d_E^2 as the dissimilarity d. The above equations (2.30), (2.31) using squared Euclidean distance can be written as

$$\Delta w^+ = -2(v - w^+) \tag{2.34}$$

$$\Delta w^- = +2(v - w^-) \tag{2.35}$$

which are called as gradient and the learning rule(equations (2.32), (2.33)) using squared Euclidean distance is given as

$$w^+ \longleftarrow w^+ + 2\eta\xi_\mu^+(v - w^+) \tag{2.36}$$

$$w^- \longleftarrow w^- - 2\eta\xi_\mu^-(v - w^-) \tag{2.37}$$

Contrary to Kohonen's LVQ algorithms, the elaborated factors ξ^+ and ξ^- makes sure that the repulsion forces are not much stronger than the attraction forces because the prototype w^+ is attracted for a considered sample and prototype w^- is repelled. So, the convergence condition will be fulfilled. As we already got the weight update method for GLVQ, we can observe different types of measures in the below section.

2.5 Generalized Matrix Learning Vector Quantization

Generalized relevance LVQ (GRLVQ) algorithm which was proposed by Hammer and Villmann [3] increase or reduce the dimension of given input data. That is, it only calculates how each dimension affects the input data space for the classification. An additional idea to this is to find pairwise correlations among the data dimensions. So to find unknown correlations, a concept of matrices is needed. Schneider, Biehl, and Hammer [16] introduced GMLVQ as an extension of GRLVQ.

GMLVQ is a method that can be explained by using a $(m \times n)$ matrix, which accommodate to the correlation of different features [16]. The parameter vector ϑ of GLVQ is extended to Q, so that this method GMLVQ tries to learn about the transformation of data points. Instead of using Euclidean distance, GMLVQ uses the squared quadratic-dissimilarity d_Q^2 because it is detected that all the input dimensions of $\mathbb{R}^n$ has similar importance. So to overcome the class discrimination, we extend the learning rules of GLVQ by adapting rules for $Q \in \mathbb{R}^{m \times n}$. Remaining part of GLVQ learning framework will be the same for GMLVQ such as GLVQ loss and derived learning rules of prototypes.

The learning rule that uses quadratic-dissimilarity with the gradient is denoted as

$$\nabla_w d_Q^2(v, w) = -2Q^T Q(v - w) \tag{2.38}$$

where

$$d_Q(v,w) = (v-w)^T Q^T Q(v-w) \tag{2.39}$$

and $\frac{\partial d}{\partial w^+}$, $\frac{\partial d}{\partial w^-}$ in the signed gradient Δw^+, Δw^- (refer to equations (2.34) and (2.35)). The gradient of GMLVQ loss is same as the gradient of GLVQ loss with respect to matrix, that is

$$\nabla_Q \phi(\mu(v)) = \xi_\mu^+ \nabla_Q d_Q^2(v,w^+) - \xi_\mu^- \nabla_Q d_Q^2(v,w^-) \tag{2.40}$$

where ξ^+ and ξ^- are same as in GLVQ and they are given as

$$\xi_\mu^+(v,f,d_{Q^+}) = \frac{2d_{Q^-}(v)\phi(\mu(v))}{(d_{Q^+}(v)+d_{Q^-}(v))^2} \tag{2.41}$$

$$\xi_\mu^-(v,f,d_{Q^-}) = \frac{2d_{Q^+}(v)\phi(\mu(v))}{(d_{Q^+}(v)+d_{Q^-}(v))^2} \tag{2.42}$$

and the gradient of d_Q^2 with respect to Q is

$$\nabla_Q d_Q^2(v,w) = 2Q(v-w)(v-w)^T \tag{2.43}$$

The basic SGD rule to learn Q is defined as

$$Q \longleftarrow Q - \eta(\xi_\mu^+ \nabla_Q d_Q^2(v,w^+) - \xi_\mu^- \nabla_Q d_Q^2(v-w)) \tag{2.44}$$

Whenever w^+ and w^- are updated, the matrix receives the updates and hence matrix Q is adjusted with respect to the class of w^+ and the class of w^-. If we split the equation (2.43), the updated formulas for the matrices Q^+ and Q^- are given as

$$Q^+ = Q^+ - \eta\xi^+ \frac{\partial d_{Q^+}^2(v,w^+)}{\partial Q^+} \tag{2.45}$$

$$Q^- = Q^- + \eta\xi^- \frac{\partial d_{Q^-}^2(v,w^-)}{\partial Q^-} \tag{2.46}$$

3 Generalized Tangent Learning Vector Quantization

If the data in the dataset is having noise, classification learning from such noisy data is still a challenging task in the machine learning which may lead to lower classification performance. In such cases, LVQ is the concept which is able to handle and process such noisy data because LVQ uses the concept of prototypes as explained in the above sections. Not only noise, data can also have some systematic transformations such as drift or different measuring settings. If the model is unable to process these variations correctly, then these variations can also lead to lower classification performance. For example, if we consider the given data are images, then the systematic variations can be in the form of rotations or shifts inside an image. If we know about these systematic variations of an image in advance, corresponding data preprocessing techniques such as scale-invariant transform (SIFT) [10] can be used to reduce their impact on the classification model. If the knowledge of expected invariances are not known before hand, then invariant dissimilarity measures could be used as the substitute of data preprocessing methods.

Systematic variations are also known as transformations, classification of transformed data can be exploited as transformation invariant class detection [13]. However, in general, it is not easy to find suitable transformations. So to overcome such general cases, we can use the popular strategy [13] that can define a parameterized transformation model. Estimation of parameters of the transformation model on a training dataset should be either previous or parallel with classification model's training.

Systematic variations in LVQs can be modeled using distribution of many prototypes to one class or by using the concept of dissimilarity measure. The dissimilarity measure is been used in this thesis and we can see how to train GLVQ using derived dissimilarity measures and this concept is affiliated to the concept of tangent distances introduced by Simard et al [17]. In the below sections, we will see how high amount of data variations can be assigned to a single prototype by using the concept of tangent distance. The explanation of GTLVQ discussed below is inspired from Sascha Saralajew from [12] and [13].

Basics for GTLVQ

Definition 3.1 (Affine subspace): Let θ be a parameter vector and $t \in \mathbb{R}^n$ is a vector which is called as translation. Then the affine subspace (linear manifold) is a set

$$\{t + B\theta \,|\, \theta \in \mathbb{R}^n\} \tag{3.1}$$

where $B \in \mathbb{R}^{n \times n}$ is the basis of n-dimensional linear subspace [12].

Definition 3.2 (Orthogonal projector): For a vector $x \in \mathbb{R}^n$, the orthogonal projection is $P_j x = W_j W_j^T x$ onto the subspace defined by the basis W_j. $H_j x$ is called the projection to its complement and mathematically written as $H_j = (\mathbf{I} - W_j W_j^T)$. The distance $d_{H_j}(v_i, w_j)$ is the dissimilarity between v_i and w_j after projection onto the complement subspace [13].

3.1 Generalized Tangent Learning Vector Quantization

The GTLVQ presented here is proposed by Sascha Saralajew and Thomas Villmann in the paper [13] . Let v_i be a data vector in the parametrized data manifold $\mathcal{V}_i(\gamma)$ with the parameter vector $\gamma \in \mathbb{R}^s$, $s \ll n$ and $v_i = \mathcal{V}_i(0)$. We assume w_j = $\mathcal{W}_j(0)$ with the prototype manifold $\mathcal{W}_j(\theta), \theta \in \mathbb{R}^r$ with $r \ll n$. The distance between data and prototype manifolds can be calculated as

$$\hat{D}^*(\mathcal{V}_i, \mathcal{W}_j) = \min_{\gamma,\theta} d(\mathcal{V}_i(\gamma), \mathcal{W}_j(\theta)) \tag{3.2}$$

where d is the dissimilarity measure of parametrized manifold and prototype manifold. The linear Taylor expansion of the manifold $\mathcal{V}_i(\gamma)$ at the point v_i is known as the tangent subspace (also called as set of tangent vectors at v_i) of $\mathcal{V}_i(\gamma)$ at v_i and tangent subspace basis V_i which can be written as:

$$\mathcal{V}_i(\gamma) \simeq v_i + \mathrm{V}_i \gamma \tag{3.3}$$

with a tangent subspace basis at v_i as:

$$V_i = \left. \frac{\partial \mathcal{V}_i(\gamma)}{\partial \gamma} \right|_{\gamma=0} \tag{3.4}$$

In the same way, we have

$$\mathcal{W}_j(\theta) \simeq w_j + \mathrm{W}_j \theta \tag{3.5}$$

with a tangent subspace basis at w_j as:

$$W_j = \left. \frac{\partial \mathcal{W}_j(\theta)}{\partial \theta} \right|_{\theta=0} \tag{3.6}$$

By using equations (3.4) and (3.6), one-sided tangent distance is calculated as

$$\hat{D}(v_i, w_j, W_j) = \min_{\theta} \{ d(v_i, w_j + W_j \theta) \} \tag{3.7}$$

where d is the dissimilarity measure for vectors. This tangent distance is called as one-sided tangent distance because we are using only one tangent subspace (W_j). If we use two tangent subspaces (V_i and W_j) in the tangent distance, then the tangent distance will be called as two-sided tangent distance and it is given as

$$\hat{D}(v_i, V_i, w_j, W_j) = \min_{\gamma,\theta}\{d(v_i + V_i\gamma, w_j + W_j\theta)\} \tag{3.8}$$

In this paper we discuss only about one-sided tangent distance for GLVQ. In order to incorporate tangent distance for GLVQ, we rewrite the equation (3.7) using the dissimilarity d as the squared Euclidean distance, so that it transforms into

$$D(v_i, w_j, W_j) = \min_{\theta}\{d(v_i - w_j - W_j\theta)^2\} \tag{3.9}$$

But in order to satisfy equation (3.9), we needs to solve the minimum problem. The mandatory condition to solve the minimum is

$$\frac{\partial D(v_i, w_j, W_j)}{\partial \theta} = 0$$

which leads to the specific formula

$$\theta = W_j^T(v_i - w_j) \tag{3.10}$$

assuming that $W_j^T W_j = \mathbf{I}$, that is, an orthonormal basis has been assumed in the tangent subspace basis W_j as explained by Villmann and Saralajew in [13].

We use Gram-Schmidt orthonormalization in this thesis to orthonormalize the basis W_j. From the above assumption of $W_j^T W_j = \mathbf{I}$, we have $\frac{\partial^2 D(v_i, w_j, W_j)}{\partial\theta\partial\theta} = \mathbf{I}$ where $\mathbf{I}$ is the Identity matrix, equation (3.10) provides the minimum for $D(v_i, w_j, W_j)$. If we substitute the value of θ as in (3.10) in equation (3.9), we will have

$$D(v_i, w_j, W_j) = (v_i - w_j - W_j W_j^T(v_i - w_j))^2 \tag{3.11}$$

The equation (3.11) can be also written as

$$D(v_i, w_j, W_j) = (H_j(v_i - w_j))^2$$

$$D(v_i, w_j, W_j) = d_{H_j}(v_i, w_j) \tag{3.12}$$

where

$$H_j = (\mathbf{I} - W_j W_j^T) \tag{3.13}$$

and H_j is called as a orthogonal projector as explained in Definition (3.2). Now, if the distance in equations (2.9) and (2.17) are replaced by the tangent distance explained in equation (3.12), we will get $H^+ = H_j$, $W^+ = W_j$, and $w^+ = w_j$ if $w_j + W_j\theta$ is the best matching prototype with correct class $c(w_j) = c(v_i)$. Similarly, H^-, W^-, w^- are

considered as best matching prototype with incorrect class, that is $c(w_j) \neq c(v_i)$. And later, we can update the translation as

$$\Delta w^+ \propto -2\varepsilon_t \cdot \xi_\mu^+(v_i, f, d_{H^+}) \cdot H^+(v_i - w^+) \tag{3.14}$$

$$\Delta w^- \propto +2\varepsilon_t \cdot \xi_\mu^-(v_i, f, d_{H^-}) \cdot H^-(v_i - w^-) \tag{3.15}$$

with the translation rate $0 < \varepsilon_t \ll 1$ and $\xi_\mu^+(v_i, f, d_{H^+})$ and $0 < \varepsilon_t \ll 1$ and $\xi_\mu^-(v_i, f, d_{H^-})$ are same as equations (2.41) and (2.42) but instead of taking the matrix Q, we replace it with matrix H. Here, we have used the derivative as

$$\frac{\partial d_{H^+}(v_i, w^+)}{\partial w^+} = -2H^+(v_i - w^+) \tag{3.16}$$

$$\frac{\partial d_{H^-}(v_i, w^-)}{\partial w^-} = -2H^-(v_i - w^-) \tag{3.17}$$

So in order to improve the separation of classes in the GLVQ, we now adapt the tangent basis that uses SGD for W_j as

$$\Delta W^+ \propto -2\varepsilon_b \cdot \xi_\mu^+(v_i, f, d_{H^+}) \cdot (v_i - w^+)(v_i - w^+)^T W^+ \tag{3.18}$$

$$\Delta W^- \propto +2\varepsilon_b \cdot \xi_\mu^-(v_i, f, d_{H^-}) \cdot (v_i - w^-)(v_i - w^-)^T W^- \tag{3.19}$$

where $0 < \varepsilon_b \ll 1$ is known as the learning rate of the tangent basis.

Initialization

It is recommended by Sascha and Villmann in [13] to use k-means initialization to each and every class and number of means is considered as number of prototypes in that particular class. We use the same methods that we have used in LVQ methods to determine number of prototypes per class. Further, if we have knowledge on the training dataset, we can use orthonormal bases to initialize the bases. But, if we do not have any knowledge on dataset, then we have to use below steps to initialize the each basis $\mathbf{B}_k$:

- Using dissimilarity, we need to find all the data samples that belongs to correct class and we will consider t_k as the nearest prototype vector. secondly, we have to find all the data samples that belongs to receptive field of t_k using LVQ approach.
- So for these data samples, compute eigenvectors that belongs to largest eingenvalues of estimated covariance matrix.
- Use these eigenvectors as an initialization for basis $\mathbf{B}_k$. If it is necessary, orthonormalize the remaining matrix.

Training

Same as GMLVQ model, we will train GTLVQ model by using GLVQ loss function as in equation (2.22) and also we adapt all the learning rules same as GLVQ. For a training sample $(v, c(v))$, we need to find the nearest correct prototype w^+ and nearest incorrect prototype w^-. Once both the correct and incorrect prototypes are found, we update the translation as given in equations (3.14) and (3.15) using the derivative as in equations (3.16) and (3.17). Later, we have the learning rule of basis as equations (3.18) and (3.19). Succeeding the changes and adjustments of bases, orthonormalizing bases (W^+ and W^-) is important because to make sure that the supposed property $W_j^T W_j = \mathbf{I}$ is necessary for solving the equation (3.10) and we use Gram-Schmidt process for orthonormalization of bases.

The entire learning process of every bases is sort of projected gradient descent. That is, there is a possibility that we apply an update which may lead to violation of the assumed solution. This means, that the matrices are orthonormal bases and therefore we use an appropriate strategy to project the updated bases back to solution space. If we consider the learning rules, see equations (3.14), (3.15) and (3.18), (3.19), we can notice that SGD updates affine subspaces in such a way that the correct affine subspace are attracted towards similar data points of a class and pushes away the incorrect affine subspace.

.

4 Transfer Learning for LVQ

4.1 Transfer Learning

Technologies such as Machine learning and Data mining have been achieving significant success in most of the knowledge engineering areas including classification, regression and clustering. However, many machine learning models work well under a common assumption: the training data and test data are drawn from the same distribution. If the distribution changes, most statistical models need to be rebuilt from the scratch using newly collected sample data. It is impossible or very expensive in the real world applications, to recollect the needed training data and rebuild the models. It would be nice to reduce the need and effort to recollect the training data. In such cases, concept of knowledge transfer or transfer learning between the task domains would be desirable. The study of transfer learning is motivated by the fact that people can intelligently apply knowledge learned previously to solve new problems faster or with better solutions. For example, we may find that learning to recognize apples might help to recognize pears. Transfer learning, in contrast, allows the domains, tasks and distributions used in training and testing to be different.

As a example, consider the problem of sentiment classification, where our task is to automatically classify the reviews on a product, such as brand of camera, into positive and negative views. For this classification task, we need to first collect many reviews of the product and annotate them. We would then train a classifier on the reviews with their corresponding labels. Since the distribution of review data among different types of products can be very different, to maintain good classification performance, we need to collect a large amount of labeled data in order to train the review-classification models for each product. However, this data-labeling process can be very expensive to do. To reduce the effort for annotating reviews for various products, we may want to adapt a classification model that is trained on some products to help learn classification models for some other products. In such cases, transfer learning can save a significant amount of labeling effort [6].

In this thesis, we concentrate only on transfer learning in Supervised learning (LVQ) which was proposed by Sascha Saralajew and Thomas Villmann in [14] as explained in the following sections.

4.1.1 Mathematical description of transfer Learning for Supervised Learning

Let us assume $K_0 \subseteq \mathbb{R}^n$ is the domain which has to be learned using supervised machine learning model and C is the classification co-domain, then the classification function is given as

$$f_0 : K_0 \longrightarrow C \tag{4.1}$$

and the approximation or estimation of f_0 is given as

$$\mathfrak{f}_0 : W \longrightarrow \mathbb{F}(\mathbb{R}^n, C)$$

where W is the set of parameters ω and $\mathbb{F}(\mathbb{R}^n, C)$ is defined as space of functions ($\mathbb{F}(\mathbb{X}, \mathbb{Y})$ is defined as function from $\mathbb{X}$ onto $\mathbb{Y}$). As discussed in the above sections, here also we assume the set of training vectors $V \subseteq K_0$ with cardinality $\#V$ and each element $\mathrm{v}_i \in V$ contains a class label $c(\mathrm{v}_i)$. To find the approximation $\mathfrak{f}_0(\omega) \approx f_0$ for a parameter set ω over W, the learning task have to be figured out depending on a given source training dataset $\mathcal{V} = \{(\mathrm{v}_i, c(\mathrm{v}_i)) | \mathrm{v}_i \in V\}$. In general, the cost function E_0 has to be minimized when learning in dependence with the parameter set ω for a given training data set $\mathcal{V}$, i.e.,

$$\omega = \underset{\omega' \in W}{\arg\min}\, E_0(\mathcal{V}, \mathfrak{f}_0(\omega')) \tag{4.2}$$

is the minimization problem that have to be solved to get the approximation $\mathfrak{f}_0(\omega)$ for a classification function f_0.

In the case of LVQ, we have parameter space W as

$$W_{LVQ} = \{\omega \in P\,(\mathrm{span}(V)) \mid \#\omega = M\} \tag{4.3}$$

where $P(\mathrm{span}(V))$ is the power set of linear span(V) of V. The class label for prototypes vectors $\mathrm{w}_k \in \omega$ are given as $c(\mathrm{w}_k) \in C$.

4.2 Transfer Model based on concept of Functions

The domain K_0 is also called as initial data space or source data space. In this thesis, we assume transfer or target data space as $K_g \subseteq \mathbb{R}^n$ with a continuous transfer function

$$g : K_0 \longrightarrow K_g \tag{4.4}$$

which is generally not known. As we are determining transfer learning for classification, we need to find the mapping and is given as

$$f_g : K_g \longrightarrow C \tag{4.5}$$

with the co-domain C which is based on the knowledge of $\mathfrak{f}_0(\omega)$. But in order to find the above mapping, it is mandatory to use the concept of inverse mapping as explained by Sascha Saralajew and Thomas Villmann in [14]. According to them, inverse mapping is to learn a mapping of the transfer data to the initial data space such that the mapped transfer data can be fed into the initial model. If we are able to identify this inverse mapping, it is easy to transfer new data into intial data space. So that we can apply initial model very easily.

Let the target training dataset be $\mathcal{Y}_g = \{(\mathrm{y}_j, c(\mathrm{y}_j)) \mid \mathrm{y}_j \in \mathrm{Y}_g \subseteq K_g\}$ and a cost function E_h for the transfer learning task (4.5). Now, the inverse mapping function $h = g^{-1}$ is determined by

$$\mathfrak{h} = \underset{\mathfrak{h}' \in \mathbb{F}(\mathbb{R}^n, \mathbb{R}^n)}{\arg\min} \; E_h(\mathcal{Y}_g, \mathfrak{f}_0(\omega) \circ \mathfrak{h}') \tag{4.6}$$

where $\circ$ is the function decomposition given as $(f \circ g)(x) = f(g(x))$ and the function $\mathfrak{f}_g$ can be estimated using $\mathfrak{f}_g = \mathfrak{f}_0(\omega) \circ \mathfrak{h}$. In general, solving the minimization problem (4.6) is very hard. So, the authors have considered the Taylor expansion of $\mathfrak{h}$ at the center $\mathrm{x}_0 \in K_g$ in order to overcome the difficulty

$$\mathcal{T}_{\mathrm{x}}[h(\mathrm{x})]_{\mathrm{x}=\mathrm{x}_0} = \mathcal{A}_{\mathrm{x}}[h(\mathrm{x})]_{\mathrm{x}=\mathrm{x}_0} + \mathcal{O}(\mathrm{x}^2)$$

with the affine part

$$\mathcal{A}_x[h(\mathrm{x})]_{\mathrm{x}=\mathrm{x}_0} = h(\mathrm{x}_0) + \mathrm{H}(\mathrm{x}_0)(\mathrm{x} - \mathrm{x}_0) \tag{4.7}$$

where

$$\mathrm{H}(\mathrm{x}_0) = \left.\frac{\partial h(\mathrm{x})}{\partial \mathrm{x}}\right|_{\mathrm{x}=\mathrm{x}_0} \in \mathbb{R}^{n \times n}$$

is the Jacobi-matrix which describes the linear term and $\mathcal{O}(\mathrm{x}^2)$ is the big $\mathcal{O}$ notation with respect to x that addresses higher order terms.

Affine Function: For a matrix(linear transformation) $\mathrm{H} \in \mathbb{R}^{n \times n}$ and a vector(translation transformation) $\mathrm{h} \in \mathbb{R}^n$, an affine function $a(\mathrm{H}, \mathrm{h}) : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is defined as [14]

$$a(\mathrm{H}, \mathrm{h})(\mathrm{x}) = \mathrm{H}\mathrm{x} + \mathrm{h} \tag{4.8}$$

The affine part $\mathcal{A}_{\mathrm{x}}[h(\mathrm{x})]_{\mathrm{x}=\mathrm{x}_0}$ of Taylor expansion by using above definition of affine function can be written as

$$\mathcal{A}_{\mathrm{x}}[h(\mathrm{x})]_{\mathrm{x}=\mathrm{x}_0} = a(\mathrm{H}, \mathrm{h})(\mathrm{x}_0)$$

Using this new affine part, the transfer learning task (4.6) can be simplified in order to find a pair (H, h) by

$$(\mathrm{H}, \mathrm{h}) = \underset{(\mathrm{H}', \mathrm{h}') \in (\mathbb{R}^{n \times n}, \mathbb{R}^n)}{\arg\min} \; E_h(\mathcal{Y}_g, \mathfrak{f}_0(\omega) \circ a(\mathrm{H}', \mathrm{h}')) \tag{4.9}$$

and estimation of f_g is realized by $\mathfrak{f}_g = \mathfrak{f}_0(\omega) \circ a(\mathrm{H}, \mathrm{h})$.

Another approach to estimate transfer learning function g is to transfer the already trained parameter set ω indirectly, so that task (4.5) is simplified directly. To explain in the mathematical terms, we define the minimization problem as

$$t = \underset{t' \in (\mathbb{F}(W,W))}{\arg\min} \; E_g(\mathcal{Y}_g, \mathfrak{f}_0(t'(\omega))) \tag{4.10}$$

4.3 Transfer model based on the concept of Manifolds

Let us consider a parametric transfer function

$$\gamma : K_0 \times \mathbb{R}^r \longrightarrow K_{\mathcal{M}} \subseteq \mathbb{R}^n \tag{4.11}$$

instead of (4.4) with $\gamma(\mathrm{x}, \theta)|_{\theta=0_r} = \mathrm{x}$ as a restriction where 0_r is the r-dimensional zero vector and $\gamma(\mathrm{x}, \theta)$ is continuous with respect to θ. The restriction says that having zero disturbance of the system does not make any change and the continuity assumption defines that a minor change in θ leads to a minor change in the signal. Additionally, it is important that the transfer function have to be a well-posed problem with respect to θ. So, the authors in [14] have considered the parametric transfer function $\gamma(\mathrm{x}, \theta)$ as a manifold with respect to θ. Hence, in the reference to the transfer model based on the functions, the transfer function $g(\mathrm{x})$ can be written as an instance to the manifold $\gamma(\mathrm{x}, \theta)$ for a fixed $\theta^* \in \mathbb{R}^r$, i.e. $g(\mathrm{x}) = \gamma(\mathrm{x}, \theta^*)$. So, as $K_{\mathcal{M}}$ is the target data space in (4.1), we therefore have, $K_\theta^* = K_g \subseteq K_{\mathcal{M}}$.

In the previous section, we have defined the transfer learning task based on simple functions. In general and in fact, all the description explained based on simple functions remains same for manifold approach if we fix θ in $\gamma(\mathrm{x}, \theta)$. It is versatile for modelling of transfer function for a arbitrary θ by the manifold ($\gamma(\mathrm{x}, \theta)$) as the intrinsic state of this system is dynamic. Everytime the θ changes, we have to calculate transfer function g or its inverse h everytime.

Transfer learning task based on manifolds can be seen as the mapping

$$f_{\mathcal{M}} : K_{\mathcal{M}} \longrightarrow C \tag{4.12}$$

depending on the knowledge of $\mathfrak{f}_0(\omega)$, which is the classification model calculated for $\theta = 0_r$. Therefore, the existence of such a mapping is supposed mathematically.

Let the target training data set be $\mathcal{Y}_{\mathcal{M}} = \{(y_j, c(y_j)) \mid y_j \in \mathrm{Y}_{\mathcal{M}} \subseteq K_{\mathcal{M}}\}$ and a cost function $E_{\mathcal{M}}$ for a transfer learning task based on the classification error, estimating the manifold γ for an arbitrary space of functions $\mathbb{F}(\mathbb{R}^n \times \mathbb{R}^r, \mathbb{R}^n)$ remains hard. So therefore,

authors in [14] used Taylor expansion to calculate and simplify the problem. Additionally, the authors have also assumed that manifold function $\gamma(\mathrm{x},\theta)$ is differentiable in both x and θ. The Taylor expansion of manifold function $\gamma(\mathrm{x},\theta)$ which is centered at x_0 and θ_0 is given as

$$\mathcal{T}_{\mathrm{x},\theta}[\gamma(\mathrm{x},\theta)]_{\mathrm{x}=\mathrm{x}_0,\theta=\theta_0} = \mathcal{A}_{\mathrm{x},\theta}[\gamma(\mathrm{x},\theta)]_{\mathrm{x}=\mathrm{x}_0,\theta=\theta_0} + \mathcal{O}(\mathrm{x}^2,\theta^2)$$

with

$$\mathcal{A}_{\mathrm{x},\theta}[\gamma(\mathrm{x},\theta)]_{\mathrm{x}=\mathrm{x}_0,\theta=\theta_0} = \gamma(\mathrm{x}_0,\theta_0) + \mathrm{G}(\mathrm{x}_0,\theta_0)(\mathrm{x}-\mathrm{x}_0) + \mathrm{W}(\mathrm{x}_0,\theta_0)(\theta-\theta_0) \tag{4.13}$$

with the affine part of the series and

$$\mathrm{G}(\mathrm{x}_0,\theta_0) = \left.\frac{\partial\gamma(\mathrm{x},\theta)}{\partial\mathrm{x}}\right|_{\mathrm{x}=\mathrm{x}_0,\theta=\theta_0} \in \mathbb{R}^{n\times n}$$

$$\mathrm{W}(\mathrm{x}_0,\theta_0) = \left.\frac{\partial\gamma(\mathrm{x},\theta)}{\partial\theta}\right|_{\mathrm{x}=\mathrm{x}_0,\theta=\theta_0} \in \mathbb{R}^{n\times r} \tag{4.14}$$

are the Jacobi-matirces that are describing the linear terms.

Clearly equation(4.13) matches the shape of previously described general affine transformation $a(\mathrm{H},\mathrm{h})$ from (4.8) with $\mathrm{H}=\mathrm{G}(\mathrm{x}_0,\theta_0)$ and $\mathrm{h}=\gamma(\mathrm{x}_0,\theta_0)+\mathrm{G}(\mathrm{x}_0,\theta_0)\mathrm{x}_0+\mathrm{W}(\mathrm{x}_0,\theta_0)(\theta-\theta_0)$ for a fixed θ.

Also, the authors from [14] have assumed that for a given $\delta>0$ there exists an ε-neighborhood $\mathcal{U}_{\varepsilon_\mathrm{x}}(\mathrm{x}_0)\subseteq K_0$ and $\mathcal{U}_{\varepsilon_\theta}(\theta_0)\subseteq\mathbb{R}^r$ with $\varepsilon_\mathrm{x}>0$ and $\varepsilon_\theta>0$, such that the approximation error fulfills the inequality

$$\parallel \gamma(\mathrm{x},\theta) - \mathcal{A}_{\mathrm{x}_0,\theta_0}(\mathrm{x},\theta) \parallel < \delta \tag{4.15}$$

for all points $\mathrm{x}\in\mathcal{U}_{\varepsilon_\mathrm{x}}(\mathrm{x}_0)$ and $\theta\in\mathcal{U}_{\varepsilon_\theta}(\theta_0)$ with respect to an appropriate norm $\parallel . \parallel$. If $\mathcal{U}_{\varepsilon_\mathrm{x}}(\mathrm{x}_0)=K_0$ and $\mathcal{U}_{\varepsilon_\theta}(\theta_0)=\mathbb{R}^r$ are valid, then the approximation of $\gamma(\mathrm{x},\theta)$ by the affine part (4.13) is globally accurate with the error δ. If it is not globally accurate, then it will be locally accurate for $\mathrm{x}\in\mathcal{U}_{\varepsilon_\mathrm{x}}(\mathrm{x}_0)$ and $\theta\in\mathcal{U}_{\varepsilon_\theta}(\theta_0)$, i.e. x and θ are required to belong to vicinities $\mathcal{U}_{\varepsilon_\mathrm{x}}(\mathrm{x}_0)$ and $\mathcal{U}_{\varepsilon_\theta}(\theta_0)$, respectively, for a valid approximation.

The transfer function $t(\omega)$ for the parameter set ω is redefined from (4.10). But in order to realize the approach efficiently for prototype based classifiers, the parameter set ω should be a subset of the data space, i.e. $\omega\subseteq K_0$. So, the parametric transfer function γ from (4.11) is also used to transform the prototype vectors $\mathrm{w}_k\in\omega$ according to

$$\mathrm{w}_k(\theta) = \gamma(\mathrm{w}_k,\theta) \tag{4.16}$$

such that $\mathrm{w}_k(\theta)$ will become the prototype vector in the target data space $K_\mathcal{M}$. We use affine approximation $\mathcal{A}_{\mathrm{x}_0,\theta_0}(\mathrm{x},\theta)$ of the Taylor expansion for $\gamma(\mathrm{w}_k,\theta)$ of particular choice

$\mathrm{x}_0 = \mathrm{w}_k$. Because of trivial fact that $\mathrm{w}_k \in \mathcal{U}_{\varepsilon_\mathrm{x}}(\mathrm{w}_k)$ is valid, we get a valid approximation for $\gamma(\mathrm{w}_k, \theta)$ in the vicinity of the prototype vector w_k. Hence, without loss of generality we take $\theta_0 = 0_r$ such that the affine approximation finally becomes

$$\mathcal{A}_{\mathrm{w}_k,0_r}(\mathrm{w}_k, \theta) = \mathrm{w}_k + \mathrm{W}(\mathrm{w}_k)\theta \tag{4.17}$$

with the Jacobian $\mathrm{W}(\mathrm{w}_k)$ from (4.14). In order to highlight the affine approximation of $\mathrm{w}_k(\theta)$, we will make use of the symbol $\mathfrak{w}_k(\theta)$. And final tasks is to determine the metrices $\mathrm{W}(\mathrm{w}_k)$ to model the approximation and in order to handle the new parameter vector θ, it is important to extend the estimated classifier $\mathfrak{f}_0$.

4.4 Basic LVQ for the Source Data Learning

As explained in the section (4.2), we suppose a source training data set $\mathcal{V}$ for GLVQ and a dissimilarity measure $d : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$ and the distance between data points and prototypes is calculated using the squared Euclidean metric $d_E(\mathrm{v}, \mathrm{w}) = (\mathrm{v} - \mathrm{w})^2$. Later, we suppose M prototype vectors which forms the parameter set ω so that a class C should have at least one prototype.

For a class, given data vector v is assigned in the GLVQ model $\mathfrak{f}_0(\omega) : K_0 \longrightarrow C$ as

$$\mathrm{v} \xmapsto{\mathfrak{f}_0(\omega)} c(w(\mathrm{v}, \omega, \mathrm{C}))$$

where

$$w(\mathrm{v}, \omega, C) = \underset{\mathrm{w}_k \in \omega | c(\mathrm{w}_k) \in C}{\arg\min} \; d_E(\mathrm{v}, \mathrm{w}_k)$$

determines the best matching prototype vector $w(\mathrm{w}, \omega, C)$ for a given data vector v, prototype vector set ω and set of class labels C. The cost function to be minimized in (4.2) is

$$E_{GLVQ}(\mathcal{V}, \mathfrak{f}_0(\omega)) = \sum_{v \in \mathcal{V}} \psi(\mu_E(v)) \tag{4.18}$$

with $\psi(z)$ being a monotonously increasing differentiable squashing function and usually we choose sigmoid function as a squashing function. Using stochastic gradient descent learning, prototype vectors are adapted for $E_{GLVQ}(\mathcal{V}, \mathfrak{f}_0(\omega))$ as

$$\Delta \mathrm{w}^{\pm}(v) \propto \left.\frac{\partial \psi}{\partial \mu_E}\right|_{\mu_E(v)} \cdot \left.\frac{\partial \mu_E}{\partial d_E^{\pm}}\right|_{d_E^{\pm}(v)} \cdot \left.\frac{\partial d_E^{\pm}}{\partial \mathrm{w}^{\pm}}\right|_{\mathrm{w}^{\pm}(v)} \tag{4.19}$$

for the prototype vectors $\mathrm{w}^{\pm}(v)$ if the random training sample $v \in \mathcal{V}$ is given. The remaining algorithm of GLVQ stays same as explained in the previous section.

4.5 Transfer Learning for GMLVQ

Transfer learning for GMLVQ has been proposed by PAASSEN ET AL. in [2]. In that paper, GMLVQ has been trained using the initial data which realizes the classification model $\mathfrak{f}_0(\omega)$. Then, the authors in [2] assumed a transfer function g according to (4.4) in which the inverse transfer function h can be calculated globally by this linear transfer learning model. In such a case, the inverse map h becomes a matrix H so that each point $\mathrm{v}^{'} \in K_g$ is mapped back to the source data space K_0 by $\mathrm{v} = \mathrm{H}\mathrm{v}^{'}$. Villmann and Sascha in [14] have verified by taking the special choice $\mathrm{h} = 0_n$ for the affine transformation $a(\mathrm{H}, 0_n) = \mathrm{H}$ by considering equation (4.8). Hence, it is possible to fed v into already learned classification model $\mathfrak{f}_0(\omega)$.

In transfer learning, if the transformation is substituted into the dissimilarity measure $d_\Omega(\mathrm{v},\mathrm{w}) = (\Omega(\mathrm{v}-\mathrm{w}))^2$, then we have

$$d_{\Omega,\mathrm{H}}(\mathrm{v}^{'},\mathrm{w}) = (\Omega(\mathrm{H}\mathrm{v}^{'}-\mathrm{w}))^2 \tag{4.20}$$

and hence, the classifier function of GMLVQ becomes

$$\mu_{\Omega,\mathrm{H}}(y) = \frac{d^+_{\Omega,\mathrm{H}}(y) - d^-_{\Omega,\mathrm{H}}(y)}{d^+_{\Omega,\mathrm{H}}(y) + d^-_{\Omega,\mathrm{H}}(y)}$$

Therefore, the transfer learning task (4.9) is given as

$$\mathrm{H} = \underset{\mathrm{H}^{'}\in\mathbb{R}^{n\times n}}{\arg\min}\, E_h(\mathcal{Y}_g, \mathfrak{f}_0(\omega)\circ a(\mathrm{H}^{'}, 0_n))$$

$$= \underset{\mathrm{H}^{'}\in\mathbb{R}^{n\times n}}{\arg\min}\left(\sum_{y\in\mathcal{Y}_g}\psi(\mu_{\Omega,\mathrm{H}^{'}}(y)) + \lambda \parallel \mathrm{H}^{'} \parallel^2_F\right)$$

with the parameter set ω of the GMLVQ scheme is fixed. The scaled Frobenius norm $\lambda \parallel \mathrm{H}^{'} \parallel^2_F$ plays as regularization term with $\lambda \in \mathbb{R}^+$. The matrix H is adapted by a SGDL using $\Delta\mathrm{H} \propto \frac{\partial\psi}{\partial\mathrm{H}}$. So, for that reason, the matrix H is defined initially as the identity I_n. The main drawback of transfer learning for GMLVQ is that, it is possible to learn only linear transformations.

4.6 Transfer Learning for GTLVQ

In this section, we will first explain about transfer learning for GLVQ and extend that to GTLVQ. Authors in [14] have assumed that the parameter set ω is the set of prototypes w_k and the affine transformation as $a(\mathrm{G},\mathrm{g}) = \mathrm{G}\mathrm{x} + \mathrm{g}$ according to (4.8) which is used as a approximation of the transfer function $g(\mathrm{x}) = \gamma(\mathrm{x},\theta^*)$ for a fixed manifold parameter θ^*. Same like previous section, if we substitute this affine transformation into the

dissimilarity measure d_E, we obtain

$$d_{\mathrm{G,g}}(\mathrm{v}^{'}, \mathrm{w}_k) = (\mathrm{v}^{'} - (\mathrm{G}\mathrm{w}_k + \mathrm{g}))^2$$

where vectors $\mathrm{v}^{'} \in K_g$ and an arbitrary prototype $\mathrm{w}_k \in \omega$. The cost function $E_g(\mathcal{Y}_g, \mathfrak{f}_0(a(\mathrm{G},\mathrm{g})(\omega)))$ with the target training data set $\mathcal{Y}_g$ to find the pair (G, g) is given as

$$(\mathrm{G},\mathrm{g}) = \underset{(\mathrm{G}^{'},\mathrm{g}^{'})\in(\mathbb{R}^{n\times n},\mathbb{R}^n)}{\arg\min} E_g(\mathcal{Y}_g, \mathfrak{f}_0(a(\mathrm{G}^{'},\mathrm{g}^{'})(\omega)))$$

$$= \underset{(\mathrm{G}^{'},\mathrm{g}^{'})\in(\mathbb{R}^{n\times n},\mathbb{R}^n)}{\arg\min} \left(\sum_{y\in\mathcal{Y}_g} \psi(\mu_{\mathrm{G}^{'},\mathrm{g}^{'}}(y)) \right)$$

and we suppose that affine part $a(\mathrm{G},\mathrm{g})$ is globally accurate approximation.

The matrix G and the vector g in GLVQ can be adapted by a SGDL scheme using $\Delta\mathrm{G} \propto \frac{\partial\psi}{\partial\mathrm{G}}$ and $\Delta\mathrm{g} \propto \frac{\partial\psi}{\partial\mathrm{g}}$ gradients. In order to calculate the first value of G, we have to use the formula $\mathrm{G} = \sum[\mathrm{Y}_g](\sum[\omega])^{-1}$ where $\sum[S]$ denotes the matrix that is generated after sorting the eigen vectors (i.e. normalized to the unit length) of the empirical covariance matrix of a set $S \subset \mathbb{R}^n$ with respect to corresponding eigen values.

In the next step, we illustrate that each prototype $\mathrm{w}_k \in \omega$ of GLVQ model as an instance of the manifold $\mathrm{w}_k(\theta) = \gamma(\mathrm{w}_k, \theta)$. Then, the manifold distance is

$$d_{\mathcal{M}}(\mathrm{v}, \mathrm{w}_k(\theta)) = \min_{\theta\in\mathbb{R}^r} d_E(\mathrm{v}, \gamma(\mathrm{w}_k, \theta))$$

and it is also called as single-sided manifold distance of a data point v to the manifold $\mathrm{w}_k(\theta)$.

We calculate $\mathrm{w}_k(\theta)$ by the affine part $\mathfrak{w}_k(\theta)$, see (4.17), the respective Taylor expansion. We have also discussed in section (4.3) that this approximation $\mathrm{w}_k(\theta)$ is locally accurate for $\theta \in \mathcal{U}^k_{\varepsilon_\theta}(0_r)$. If we substitute affine part into d_E, we have

$$d_{rT}(\mathrm{v}, \mathfrak{w}_k(\theta)) = \min_{\theta\in\mathcal{U}^k_{\varepsilon_\theta}(0_r)} d_E(\mathrm{v}, \mathrm{w}_k + \mathrm{W}_k\theta) \tag{4.21}$$

which is said to be restricted tangent distance [15]. If we insert the Taylor expansion that is globally accurate in (4.21), we can simplify the restricted tangent distance as

$$d_T(\mathrm{v}, \mathfrak{w}_k(\theta)) = \min_{\theta\in\mathbb{R}^r} d_E(\mathrm{v}, \mathrm{w}_k + \mathrm{W}_k\theta) \tag{4.22}$$

which is called as single-sided tangent distance [1]. In order to solve the minimization problem, (4.22) has to be solved and it is given as

$$\theta = \mathrm{W}_k^T(\mathrm{v} - \mathrm{w}_k)$$

having $\mathrm{W}_k^T \mathrm{W}_k = \mathrm{I}_r$ is valid, i.e. for a subspace described by W_k, an orthonormal basis has been assumed. So, if we substitute the above equation in (4.22), we will have

$$d_T(\mathrm{v}, \mathfrak{w}_k(\theta)) = (\mathrm{P}_k(\mathrm{v} - \mathrm{w}_k))^2$$

where $\mathrm{P}_k = \mathrm{I}_n - \mathrm{W}_k \mathrm{W}_k^T$ is the orthogonal projector onto the complement subspace defined by the basis W_k.

Transfer learning task (4.12) with a given source GLVQ model $\mathfrak{f}_0(\omega)$ and the target data set $\mathcal{Y}_\mathcal{M}$ is acquired by inserting d_T into $\mathfrak{f}_0(\omega)$. Consequently, the cost function of GLVQ in transfer learning is

$$E_\mathcal{M}(\mathcal{Y}_\mathcal{M}, \mathfrak{f}_0(\mathfrak{w}(\omega))) = \sum_{y \in \mathcal{Y}_\mathcal{M}} \psi(\mu_T(y))$$

where $\mathfrak{w}(\omega)$ is the notation of transformation of prototype vectors. As in GMLVQ, we again use SGDL to optimize the above cost function $E_\mathcal{M}$ in order to learn the transformation matrix W_k for each prototype according to $\Delta \mathrm{W}_k \propto \frac{\partial \psi}{\partial \mathrm{W}_k}$.

So, as an extension of GLVQ, GTLVQ which was explained in the above section 3.2 and is proposed by Villmann and Saralajew in [13] . For the transfer learning of GTLVQ, the author in [14] have assumed that both the training data $\mathcal{V}$ and transfer data $\mathcal{Y}_\mathcal{M}$ are available at the same time. Hence, the overall training data set is $\mathcal{V}_\mathcal{M} = \mathcal{V} \cup \mathcal{Y}_\mathcal{M}$. The only difference for the transfer learning of GLVQ and transfer learning of GTLVQ is that in GTLVQ method, both the vector w_k and matrix W_k are determined together at the same time, i.e. the transformation $\mathfrak{w}_k(\theta)$ is learned directly.

5 Experimental Results

In this chapter, we will be discussing about the different experiments that are conducted and their results with respect to LVQ models. We will compare the results of transfer learning when applied to GMLVQ model and transfer learning applied to GTLVQ model.

The experiments were performed over two datasets, first is the synthetic dataset and the second is on MNIST dataset. We have considered just a part of MNIST dataset and applied the above models on it.

For implementation, we have considered batch gradient descent learning and our dataset has been splitted using train_test_split with the test_size of 33% in synthetic data and we have considered just 18000 values out of 60000 from MNIST dataset for training and for testing we have considered 3000 values out of 10000 values.

We have used following system requirements to implement GMLVQ, GTLVQ and transfer learning technique:

- Python 3.7
- Matplotlib
- Tensorflow
- Keras

We are doing these experiments in order to check which algorithm works better when transfer learning technique is applied. By applying these models to the two datasets, we have observed that how different values and different datasets affect our results in terms of accuracy.

5.1 Experimental Results

The experimental results which were performed on the two datasets are discussed in the following sections. So, to build the models using LVQ-based approaches, we need data v and class labels $c(v)$. In the below sections, we will first discuss about the results of models applied on synthetic data and later we will discuss about MNIST dataset.

5.1.1 Synthetic Data

We have considered randomly generated two-dimensional multivariate normal data with six clusters and generated 100 data samples for each cluster as shown in Fig.5.1

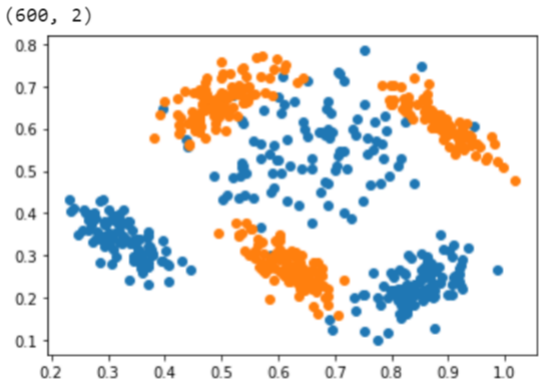

Figure 5.1: Multivariate normal data (Synthetic data)

We have used train_test_split for splitting data and the size of test_data is 33%. For the transfer learning task, we have considered all the above as a source data and for the target data, we have used transformation as rotating the data using some angle for the same data. Further, this dataset is been used for examining the results for GMLVQ and GTLVQ models.

GMLVQ using Synthetic data

As discussed above, we have taken data as a vector v with class labels $c(v)$. So to represent the data, we initialized the set of prototypes w with the class labels $c(w)$. Further, we have used the Euclidean distance to calculate the distance between data and prototypes. We have calculated also the value of cost function and updated the value of cost function by minimizing the error. Fig. 5.2, explains about the training of the GMLVQ algorithm using 1 prototype per class with a learning rate = 0.001 with the accuracy of 93%.

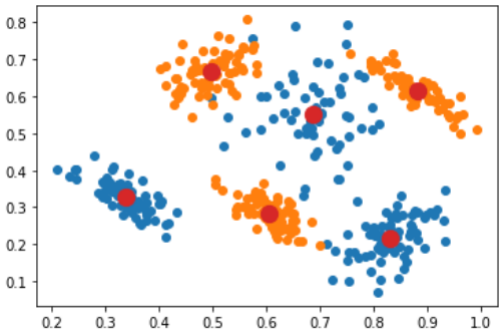

Figure 5.2: Classification of synthetic data using GMLVQ algorithm

GTLVQ using Synthetic data

Similar to GMLVQ algorithm, we will be using the same data, the set of prototypes

and we will calculate the cost fucntion. The Fig.5.3 explains the training of GTLVQ algorithm using 1 prototypes per class with learning rate = 0.001 and we have received the accuracy of 95%.

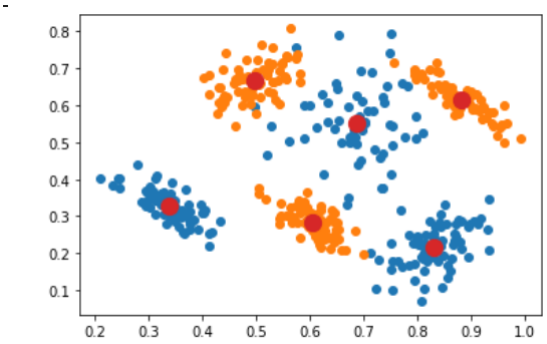

Figure 5.3: Classification of synthetic data using GTLVQ algorithm

Transfer Learning for GMLVQ model

We have considered the 100 data samples for each cluster in source data as D_0 and for the target data, we have considered affine transformation of the source data as $g(\alpha) : K_0 \longrightarrow K_{g(\alpha)}$ as

$$\mathrm{v} \xmapsto{g(\alpha)} \mathrm{R}(\alpha)(\mathrm{v}-\mathrm{t})+\mathrm{t} \tag{5.1}$$

in which $\mathrm{t} = (6,5)^T$ as a fixed translation (vector shift) and $\mathrm{R}(\alpha)$ is the linear mapping is taken as a rotation

$$\mathrm{R}(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}$$

We have considered $\Omega = \mathbf{I}_2$ as defined in (4.20) and it is fixed throughout the training. In this thesis, for the experiments, we have considered $\alpha \in \{78^\circ, 144^\circ, 205^\circ, 294^\circ\}$. The dataset looks like in Fig. 5.4, Fig. 5.5, Fig. 5.6, Fig. 5.7 after the transformations.

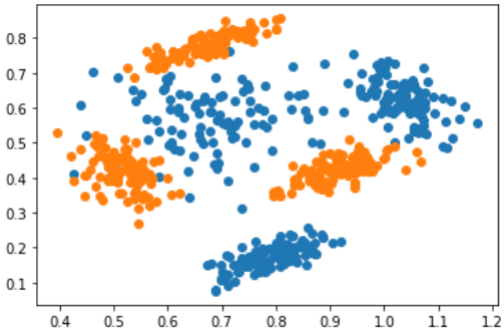

Figure 5.4: Target data for the transfer learning using transformation of 78°

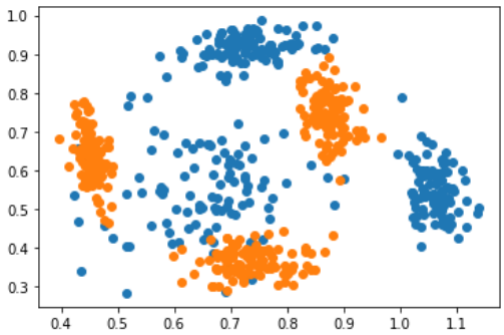

Figure 5.5: Target data for the transfer learning using transformation of $144°$

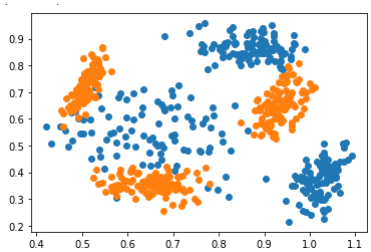

Figure 5.6: Target data for the transfer learning using transformation of $205°$

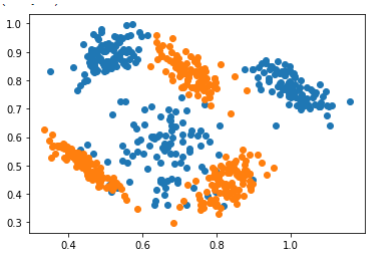

Figure 5.7: Target data for the transfer learning using transformation of $294°$

Now, as explained in section(4.5), we will apply transfer learning technique for GMLVQ. Firstly, we have trained transfer learning for GMLVQ by considering one prototype per class on the training source dataset. The accuracy of all the transformations are provided in the following table 5.1.

	$78°$	$144°$	$205°$	$294°$
GMLVQ	94.9%	93.9%	93.9%	95.4%
GMLVQ for transferred data	75.2%	94.9%	61.6%	78.7%

Table 5.1: Accuracy of GMLVQ and GMLVQ for transferred data using different transformations

Transfer Learning for GTLVQ model

Same as GMLVQ, we have considered 100 samples for each cluster in source data as K_0 and for the target data, we have considered affine transformation of the source data. We have also considered $\Omega = \mathbf{I}_2$ as defined in (4.20) and it is fixed throughout the training. Same as GMLVQ, we have considered $\alpha \in \{78^\circ, 144^\circ, 205^\circ, 294^\circ\}$ and the images are same as shown above. We have trained transfer learning for GTLVQ by considering one prototype per class on the training source dataset. The accuracy of all the transformations are provided in the following table 5.2.

	78°	144°	205°	294°
GTLVQ	94.9%	93.9%	93.9%	95.4%
GTLVQ for transferred data	85.3%	95.9%	85.8%	99.4%

Table 5.2: Accuracy of GTLVQ and GTLVQ for transferred data using different transformations

Comparison of Transfer learning of GMLVQ and GTLVQ

We have trained the two models (GMLVQ and GTLVQ) using transfer learning using the source and training datasets. We can clearly observe that GTLVQ approach performed very good when compared with GMLVQ when transfer learning at $\mathcal{Y}_{g(144^\circ)}$ for GTLVQ reflects the class distribution very well , because the model is able to learn an affine transformation that consists of both linear and a shift term.

As we have considered manifold approximation for prototypes, the structure of manifold is taken as $\gamma(\mathrm{v}, \alpha) = g(\alpha)(\mathrm{v})$ where $\gamma(\mathrm{v}, \alpha)$ is a transfer function and α plays the role of parameter vector θ of the manifold. Manifold here is the rotation $R(\alpha)$ at a center t. In this experiments, we have estimated that the manifold that is based on a prototype vectors $\mathrm{w}_k \in \omega$ of the sources GMLVQ and GTLVQ models ($\mathrm{f}_0(\omega)$) by adapting orthonormal bases $\mathrm{W}_k \in \mathbb{R}^2$. So, in order to train the transfer models, we have considered the overall training data set, i.e. $\mathcal{V}_{\mathcal{M}} = \mathcal{V} \cup \mathcal{Y}_g$. As we can observe from the 5.1 GMLVQ is not able to perform good when learning the affine transformation. For the target data, we have considered transformations as explained in the above paragraph, the accuracy is low when compared to source data because source data does not have any transformations applied.

After training GTLVQ model, we have achieved a classification training error of 7.6% and a classification test error of 8.6% where as for GMLVQ model, we have achieved a classification training error of 7.6% and a classification test error of 31.4%. We can clearly observe the good generalization performance when GTLVQ model is applied.

5.1.2 MNIST Dataset

We will try to test GMLVQ and GTLVQ models and transfer learning technique for GMLVQ and GTLVQ. So, we have considered MNIST dataset of handwritten digits which

contains $70,000$ black and white images representing the digits 0 to 9 having 10 classes and the images were centered in 28×28 pixels for the testing. The entire data has been split into $60,000$ training images and $10,000$ testing images as shown in the below Figure 5.8 and this figure is not having the labels.

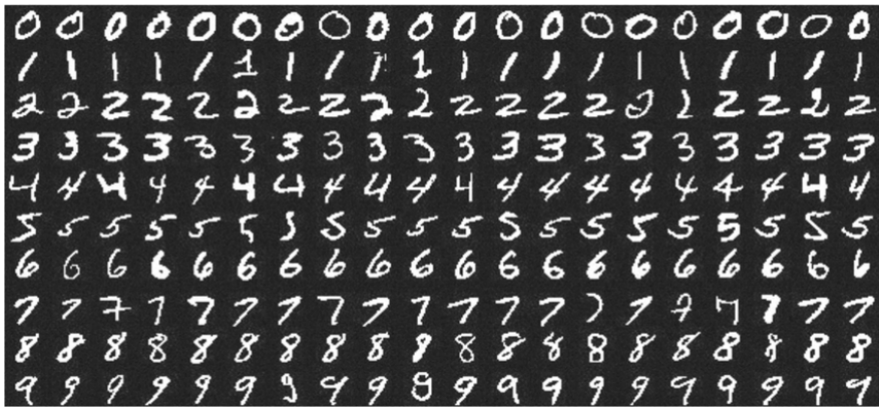

Figure 5.8: Digit images with 28 $\times$ 28 pixels

We have not applied any of the preprocessing techniques to the dataset before training but we have rescaled images which lies in between 0 and 1 and the normalization have been applied only for the data visualization. We have trained GMLVQ and GTLVQ by considering only one prototype per class. The learning rate that we have considered is 0.001 and we have chosen $r = 7$ for the dimension of tangent basis which is maintained equally to all of the prototypes. We have used k-means initialization for initializing the prototypes and we have used stochastic gradient descent to minimize the error in the cost function.

For the source data, we have considered $19,000$ images which are the variants of digits $0, 1, 2$. For the target data we have considered the digits $1, 2, 4$ but with the data noise of 5% and we have rotated the images with the angle of 40° which can be said as the transformation. We have considered the variants of digits $0, 1, 2$ for the source data and the variants of digits $1, 2, 4$ for the target data as shown in below figures 5.9 and 5.10 .

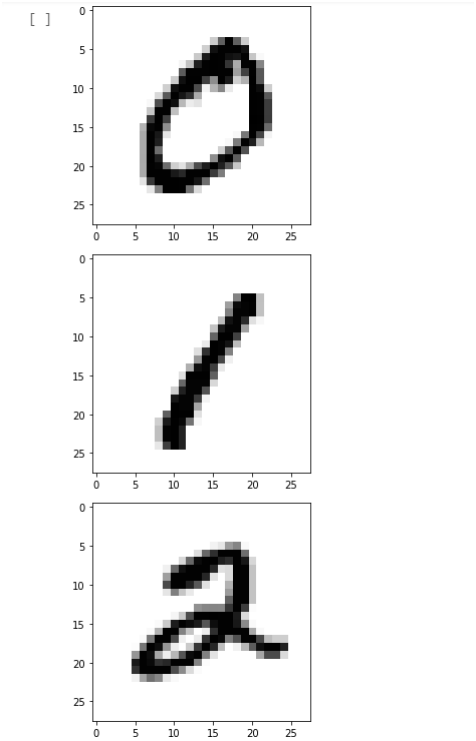

Figure 5.9: Digits considered for source data with 28 $\times$ 28 pixels

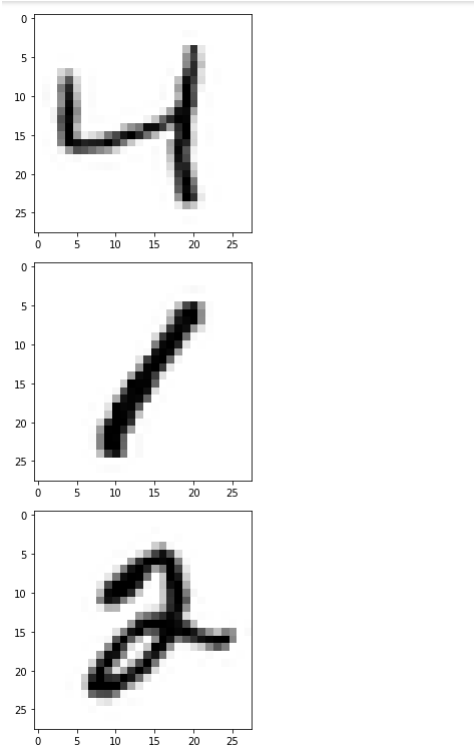

Figure 5.10: Digit considered for target data with 28 $\times$ 28 pixels

The accuracy rate we found was quite impressive. The accuracy of GMLVQ for the

MNIST dataset and the accuracy of GMLVQ after the transfer learning technique applied is 87.5% and 90.5%. The accuracy of GTLVQ for the MNIST dataset and the accuracy of GTLVQ after the transfer learning technique applied is 86.1% and 81.4%. The accuracy of GTLVQ after applying the transfer learning is not that good when compared with GMLVQ and the reasons we have observed are might be because of high dimensional data and as there are many features.

6 Conclusion and Future work

In this master thesis, we have discussed variants of LVQ's and transfer learning for GMLVQ and GTLVQ. These algorithms have been compared using practical results and we have analysed on different datasets. We run the algorithms on the preprocessed dataset(MNIST dataset) and synthetic dataset. Further, we investigated the results from the mentioned algorithms and compared them in order to see which algorithm gives the best result on our dataset.

In the beginning we have created synthetic data and trained GMLVQ algorithm and received the accuracy of 93%. Later, we have trained the same dataset using GTLVQ algorithm and got the accuracy of 95%. We have also trained the transfer learning for GTLVQ and GMLVQ on the datasets and the accuracies were given in the tables 5.1 and 5.2.

Later, we have applied the models on MNIST dataset by considering 3 digits and their variants as source data. On the other side, we have considered again 3 digits and added the data noise and changed the angle of images as transformations for the target data. After successfully running the models on huge MNIST dataset, the accuracy is so impressive and we have observed the results are between $85\% - 90\%$. From the experimental results, we have observed that GTLVQ works better for both small and huge dataset and provides the better results.

One conclusive evidence obtained by these experimental results is that GTLVQ performs much better than GMLVQ. Consequently even after applying the transfer learning technique, we can observe that GTLVQ performs much better than GMLVQ and one reason is that tangent bases are adapted in GTLVQ. The main advantage of transfer learning is that if the data dimension is large, we will have the lower computational complexity and hence we have to transform the prototypes only once.

The work performed in this thesis provides the basis for the future, that is a theoretical and practical comparison between GMLVQ and GTLVQ when the transfer learning technique is applied. Although it takes so much of time to run the huge dataset using the transfer learning technique, it is suggestible that at least it will be easy to train when the new data have been added to the already trained data. GTLVQ has been turned out to be a powerful classifier but due to computational constraints, it could not be deeply explored. This could be fuel for future research.

Bibliography

[1] B. Hammer B. Paassen, A. Schulz. Learning discriminant tangent models for handwritten character recognition. *Internatonal Conference on Artificial Neural Networks (ICANN*95)*, 2:585–590, 1995.

[2] B. Hammer B. Paassen, A. Schulz. Linear supervised transfer learning for generalized matrix lvq. *Proceedings of the Workshop New Challenges in Neural Computation 2016*, pages 11–18, 2016.

[3] Barbara Hammer and Thomas Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8):1059–1068, 2002.

[4] Andreas Holzinger. Introduction to machine learning knowledge extraction (make). *Machine Learning and Knowledge Extraction,*, 1:1–20, 2017.

[5] Eyke Hüllermeierr. Towards analogy-based explanations in machine learning. 2020.

[6] M. Dredze J. Blitzer and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. pages 432–439, 2007.

[7] Robert Tibshirani Jacob Bien. Prototype selection for interpretable classification. *Annals of Applied Statistics 2011*, 7(2):2403–2424, 2016.

[8] Teuvo Kohonen. Learning vector quantization. In *Self-organizing maps*, chapter Learning Vector Quantization, pages 175–189. Springer, 1995.

[9] Qiong Liu and Ying Wu. Supervised learning. *Encyclopedia of the Sciences of Learning*, page 3243–3245, 2012.

[10] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[11] David Nebel, Marika Kaden, Andrea Villmann, and Thomas Villmann. Types of (dis-) similarities and adaptive mixtures thereof for improved classification learning. *Neurocomputing*, 268:42–54, 2017.

[12] Sascha Saralajew. New prototype concepts in classification learning. page 198, 2020.

[13] Sascha Saralajew and Thomas Villmann. Adaptive tangent distances in general-

ized learning vector quantization for transformation and distortion invariant classification learning. *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 2672–2679, 2016.

[14] Sascha Saralajew and Thomas Villmann. Transfer learning in classification based on manifold models and its relation to tangent metric learning. *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1756–1765, 2017.

[15] Sascha Saralajew and Thomas Villmann. Restricted tangent distances for local data dissimilarities. 2018.

[16] Petra Schneider, Michael Biehl, and Barbara Hammer. Adaptive relevance matrices in learning vector quantization. *Neural computation*, 21(12):3532–3561, 2009.

[17] Patrice Simard, Yann LeCun, and John Denker. Efficient pattern recognition using a new transformation distance. *Advances in Neural Information Processing Systems*, 5:50–58, 1993.

[18] Atsushi Sato Keiji Yamada. Generalized learning vector quantization. *Touretzky DS, Mozer MC, Hasselmo ME (eds) Advances in neural information processing systems*, page 423–429, 1996.

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, im January 2022