Improved Sampling Strategy for Representative Set Construction

by

Debangsha Kusum Sarkar,

MDS., The University of British Columbia, 2019

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE COLLEGE OF GRADUATE STUDIES

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Okanagan)

January 2022

© Debangsha Kusum Sarkar, 2022

The following individuals certify that they have read, and recommend to the College of Graduate Studies for acceptance, a thesis/dissertation entitled:

Improved Sampling Strategy for Representative Set Construction
submitted by Debangsha Kusum Sarkar in partial fulfillment of the requirements of the degree of Master of Science.

Dr. Apurva Narayan, Irving K. Barber Faculty of Science
Supervisor

Dr. Ramon Lawrence, Irving K. Barber Faculty of Science
Supervisory Committee Member

Dr. Jeffrey Andrews, Irving K. Barber Faculty of Science
Supervisory Committee Member

Dr. Mahmudur Fatmi, School of Engineering
University Examiner

"The great thing about predictions is that you can be wrong."
— Dr. Chris Wiggins (Columbia University)

Abstract

Active learning solves machine learning problems where acquiring labels for the data is costly. A representative subset helps active learning by selecting the most useful subset of a complete data set and querying the labels of the examples present in it. Various sampling strategies exist for the construction of this type of representative subset. However, the state-of-the-art approaches for subset sampling aim to construct a representative subset where the cost of optimization on the subset is comparable to that of the complete data set. They do not attempt to capture the underlying distribution of the data, which we believe is the most important part of representative sampling. This thesis proposes an adaptation of the sigma point sampling (SPS) technique from unscented transformation (UT) for subset sampling. Unscented transformation (UT) has shown to be very effective in non-linear transformation modeling in object tracking and robotics.

We show in this thesis that when combined with the Gaussian mixture model, sigma points can estimate the true statistics of an unknown distribution of data with very few samples. The sigma point sampling approach being parameterized, gives better control over the sampling process than the non-parameterized sampling techniques. We use the representative subset for active learning on an existing data set. As a practical application of this novel sampling technique, we use it for Active Transfer Learning (ATL) on autoclave processing data to optimize the manufacturing of high-quality aerospace composite parts. The sampling technique is tested on both classification and regression problems in a pool-based active learning scenario.

After comparing our approach to the state-of-the-art sampling techniques, We show that sigma point sampling either outperforms or matches their performance. We also test the sensitivity of the parameters in the sigma point sampling. The success of this sampling technique is very significant for future developments of representative sampling on big data sets. The application of our representative subset is not limited to active learning and transfer learning. There is further potential to extend this approach to parallelizing computation on big data sets among multiple clusters.

Lay Summary

Traditional machine learning models are data-hungry. Training supervised models require a massive amount of labeled data. However, in some industries acquiring labels might be costly even though unlabeled data is cheaply available. Active learning solves this problem by querying the labels of the most *informative* unlabeled training examples and training a supervised model with a small amount of data. In this thesis, we study the construction of representative subsets which can be used to train an active learning model. A representative subset is a summarized version of the complete data set. We propose a novel approach for representative subset construction that preserves the local data distribution in a data set. Our approach generates sigma points, and they are calculated based on the local statistics of the data set, which is acquired through Gaussian mixtures. We train the active learner on the sigma points instead of the data points.

Preface

The experiments in the thesis outlined in Chapter 5 were conducted by the author Debangsha Sarkar. The experiments in section 5.1 were designed by the author. The data used in this experimental setup was prepared by the author.

The experimental design described in section 5.2 was adopted from *Ramezankhani et al.* [43]. Milad Ramezankhani helped the author in the preparation of the composite manufacturing data set used in this section. A detailed description of the data set is in Chapter 4.

Finally, all the experimental plots and figures, manuscript composition were done by the author under the careful supervision of the thesis supervisor Dr. Apurva Narayan.

Table of Contents

Abstract . . . . . **iii**

Lay Summary . . . . . **iv**

Preface . . . . . **v**

Table of Contents . . . . . **vi**

List of Tables . . . . . **viii**

List of Figures . . . . . **ix**

List of Equations . . . . . **xi**

List of Abbreviations . . . . . **xii**

Acknowledgements . . . . . **xiv**

Dedication . . . . . **xv**

1 Introduction **1**
- 1.1 Passive vs Active Machine Learning 2
  - 1.1.1 Passive Learning 2
  - 1.1.2 Active Learning 4
- 1.2 Scalability in Unsupervised Learning 5
- 1.3 Thesis Proposal 5
- 1.4 Thesis Outline 8

2 Background **9**
- 2.1 Problem Statement in Pool-based Active Learning 9
- 2.2 Uncertainty Sampling 9
- 2.3 Query-By-Committee 10

2.4 Representative Sampling 11
2.4.1 Coreset 11
2.4.2 K-means Based Coreset 12
2.5 Unscented Transform 13

3 Sigma Point Sampling, a Strategy for Improved Active Learning 15
3.1 Variational Autoencoder (VAE) 15
3.2 Sigma Point Sampling on Latent Space 17

4 Applications of Sigma Point Sampling in Industry 4.0 . . . 21
4.1 Transfer Learning on Composites Manufacturing Data 22
4.1.1 Transfer Learning with Sigma Point Sampling 24
4.1.2 Composite Material data set 25
4.2 Regression with Sigma Point Sampling on Composites Manufacturing Data 27

5 Experiments 28
5.1 Sigma Point Sampling on MNIST data set 28
5.1.1 Experimental Setup 28
5.1.2 Experimental Results 33
5.1.3 Comparison of different sampling techniques 33
5.1.4 Sensitivity Analysis 34
5.2 Transfer Learning with Sigma Point Sampling on Composite Manufacturing data set 41
5.2.1 Experimental results 42
5.3 Regression Learning with Sigma Point Sampling on Composite Manufacturing data set 47
5.3.1 Experimental results 48

6 Conclusion and Future Work 49

Bibliography 51

List of Tables

4.1 Observed variables used in predictive modeling of composite materials data set. 26

5.1 Labels used for classification problems with their corresponding data set sizes used for training and testing. 29

5.2 Labels used for classification problems with their corresponding data set sizes used for training and testing. 47

List of Figures

1.1 Sub-figure (a) shows unlabeled data. (b) shows a rough estimate of the groups present in the data achieved through a very efficient algorithm e.g. initialization step of k-means++. (c) shows the selected samples from each group 6

2.1 Efficient prediction of unknown distribution of a non-linear function $y = f(x)$ using Unscented Transform. For 2-dimensional data it uses $(2 \times 2 + 1) = 5$ sigma points in x [63]. 14

3.1 The data set distribution D lies in a high dimensional x-space while the latent space is relatively simpler and low dimensional (figure shows a spherical latent space to emphasize the simplicity of latent space and the complexity of the observed variable space). The generator model estimates the joint distribution $p_\theta(x, z)$, which for a isotropic Gaussian distribution can be factorized into $p_\theta(x, z) = p_\theta(z)p_\theta(x|z)$. $p_\theta(z)$ is the prior distribution which we assume to be $p_\theta(\mathbf{z}) \sim \mathrm{N}(\mathbf{z}; 0, \mathbf{I})$ and $p_\theta(x|z)$ is the stochastic decoder. The stochastic encoder $q_\phi(z|x)$ approximates the true posterior distribution $p\theta(z|x)$. [26] 16

3.2 Representative querying using sigma point sampling. Sigma points $\mathcal{X}$ are highlighted in red. $x_j \in \mathcal{U}$ are unlabeled data in the encoded latent space. A Sigma point takes label from their nearest neighbor.The $(\mathcal{X}_j, y_j)$ pair is added to the labeled pool for training active learner. 18

4.1 Autoclave curing process in aerospace manufacturing. [43] . . 22

4.2 Transfer Learning with Sigma Point Sampling. 24

4.3 Two-step autoclave cure cycle.[45] 26

5.1 Accuracy comparison of sigma point sampling vs. other representative sampling methods 30

5.2 Accuracy comparison of sigma point sampling vs. maximum uncertainty sampling, margin sampling and entropy sampling 31
5.3 Accuracy comparison of sigma point sampling vs. consensus entropy sampling, maximum disagreement sampling and vote entropy sampling of Query by committee. 32
5.4 Effect of α on the spread of sigma points. As the value of α increases the sigma points spread more across a Gaussian distribution. The spread of the sigma points affects their representative quality. 35
5.5 Spread of the sigma points in the 2 dimensional latent space of MNIST data set (with digits 0,3, and 9) for different α values. 36
5.6 Spread of the sigma points inside each Gaussian mixture component and throughout the 2 dimensional latent space of MNIST data set (with digits 0,3, and 9) for different α values. In the regions where the latent space data is sparse the Gaussian components have higher covariance. 37
5.7 Representation of the 2 dimensional latent space of MNIST data set (with digits 0,3, and 9) with the sigma points for different α values. In the regions where the latent space data is sparse, the higher covariance of the Gaussian components causes the sigma points to be more spread out than the dense regions. By changing the value of α, which is a scaled covariance parameter for sigma points, we can control the spread of the sigma points inside the Gaussian components even further to get a satisfactory representation. 38
5.8 Effect of α on multi-class classification accuracy 39
5.9 Comparison of performance metrics with increasing number of training examples in AS4/8552 for SPS vs. random sampling 42
5.10 Comparison of accuracy in Random vs Transfer Learning (TL) vs Active Learning (AL) vs ATL on 20 points 44
5.11 Comparison of accuracy in Random vs TL vs AL vs ATL on 100 points 45
5.12 Comparison of accuracy in Random vs TL vs AL vs ATL on 200 points 46
5.13 Performance metrics Mean Absolute Error (MAE), Mean Squared Error (MSE) and R^2 Score on test data set for Regression with Sigma Point Sampling (SPS) 48

List of Equations

1.1 Lower Bound in PAC Learning 3
2.1 Least Confident Sampling 10
2.2 Margin Sampling 10
2.3 Entropy Sampling 10
2.4 Voting Entropy in Query by Committee (QBC) 10
2.5 Cost of Optimization 11
2.6 Coreset Criteria 11
2.7 Cost of Optimization in Uniform Sampling on Complete data set . 12
2.8 Cost of Optimization in Uniform Sampling on Coreset 12
2.9 Sigma Point Calculation 13
3.1 Variational Lower Bound in Variational Auto Encoder (VAE) . . . 17
3.2 Mixture of Gaussians 18
3.3 Sigma Point Calculation with Gaussian Mixture Models 19

List of Abbreviations

AL Active Learning. x, xi, 4, 44–47, 49

ATL Active Transfer Learning. iii, x, xi, 42–47, 49, 50

ELBO Evidence Lower Bound. xi, 15

GMM Gaussian Mixture Model. xi, 5, 7, 17, 30, 34, 49, 50

HTC Heat Transfer Co-efficient. xi, 25, 26

KL Kullback Leibler. xi, 10

KLD Kullback Leibler Divergence. xi

MAE Mean Absolute Error. x, xi, 27, 47, 48

ML Machine Learning. xi, 1, 50

MRCC Manufacturer's Recommended Curing Cycle. xi, 21, 27

MSE Mean Squared Error. x, xi, 27, 47, 48

OVR One Versus Rest. xi, 28

PAC Probably Approximately Correct. xi, 3, 12, 50

PDF Probability Density Function. xi, 15

QBC Query by Committee. xi, 10, 32, 33

RBF Radial Basis Function. xi, 28

ReLU Rectified Linear Unit. xi, 29

SELU Scaled Exponential Linear Unit. xi, 41

SPKF Sigma Point Kalman Filters. xi, 13

SPS Sigma Point Sampling. x, xi, 2, 28, 42, 43, 48–50

SPSATL Sigma Point Sampling based Active Transfer Learning. xi, 49

SVM Support Vector Machine. xi, 28, 29, 31

TL Transfer Learning. x, xi, 22, 24, 25, 44–47, 49, 50

UKF Unscented Kalman Filters. xi, 7, 13, 49

UT Unscented Transformation. xi, 13

VAE Variational Auto Encoder. xi, 12, 15, 17, 18, 29, 50

Acknowledgements

I would like to sincerely acknowledge the endless patience of my supervisor, Dr. Apurva Narayan. I am grateful to him for always motivating me to work smarter especially when I made clumsy mistakes and providing me with numerous research opportunities. I would like to thank Dr. Ramon Lawrence for giving me the opportunity to be a part of this wonderful academic journey.

I enjoyed working with other students on many projects and was able to learn a lot from them. I would like to thank Milad Ramezankhani from the Mechanical Engineering department, for providing me with the industrial data set for my research and guiding me through the composite material manufacturing process. I also appreciate Pradeep Mahato for helping me with many aspects of research.

All the experiments in my research was made possible by the computational resources provided by UBC ARC and Compute Canada.

Dedication

I would like to dedicate this work to my parents who were extremely understanding and always motivated me to pursue my dreams.

Chapter 1

Introduction

Machine Learning (ML) in computer science is the study of computer programs that automatically learn from examples and use their learnt experience to improve their performance in specific tasks. In the last decade, we have seen an explosion of machine learning applications in different problems of computer science and other disciplines of research. Machine learning has been successful in risk analysis in financial sectors [9, 15, 19], disease diagnosis in health-care [32, 55], quality control in manufacturing [39, 69], and many more. There are three distinct categories of machine learning methods, namely supervised learning, unsupervised learning and reinforcement learning.

- **Supervised learning:** In this learning category, d-dimensional data $x \in \mathcal{R}^d$ and its corresponding labels y is fed to the learner as pair $(x_i, y_i)_{i=1}^N$. The pairs are used to train the learner and predict the labels for new data. If the labels y are real numbers the learning task is called regression and if the labels y are ordinal the task is called classification.

- **Unsupervised learning:** In this learning category, only d-dimensional data $x \in \mathcal{R}^d$ is fed to the learner without the labels. The unlabeled data is used to find the groups in the data. Clustering is one application of unsupervised learning.

- **Reinforcement learning:** In this machine learning category, an intelligent agent receives a sequence of acceptable states and its corresponding rewards (s_t, r_t) for each time step t. The learner predicts a policy $\pi(a, s) = P(a_t|s_t)$ that moves the state to s_{t+1} by an action a_t and a reward r_{t+1} associated with the transition (s_t, a_t, s_{t+1}). The goal of the learner is to predict a policy that maximizes the cumulative reward over time [22]. Reinforcement learning does not require labeled data.

The machine learning systems are set to optimize one or more specific objective(s) by learning from data ingested into it. Hence, these systems

are called data-driven systems [42, 59]. In recent years, as opposed to hand-crafted rule-based expert systems, machine learning models have shown to be more effective and more efficient in most cases [37, 49]. However, one caveat is the amount of data machine learning models require to achieve such performance.

In the current times, data collection is effortless in most industries, and data storage is very cheap. However, data labelling is still costly and time-consuming. For example, labelling medical images for diagnosis requires medical professionals to hand label images. Labeling industrial sensor data for fault detection requires passing the data through costly software [58]. This restricts the availability of the labelled data for machine learning applications. Apart from the cost associated with labeling the data, training both supervised and unsupervised models on massive data sets is time-consuming and computationally expensive. It means that neither supervised nor unsupervised learning algorithms are scalable when applied on a huge number of examples, even though a considerably large data set is necessary for better learning. To solve this problem, in this thesis, we propose a novel data summarization technique called the SPS, which can help in training the machine learners with very few examples and still achieve significant performance.

1.1 Passive vs Active Machine Learning

1.1.1 Passive Learning

Computation learning theory deals with the general principles that dictate the final hypothesis space proposed by the machine learning models. It theorizes the characteristics of machine learners to measure the quality of learnt experience from examples. These characteristics include the convergence of the learner to the correct hypothesis, the sample complexity (number of examples) and computation complexity required to learn a concept class (the distribution function associated with a particular class). If we consider a traditional supervised learning task, labeled samples $(x_i, y_i)_{i=1}^{N}$ with iid $x_i \sim D$ and $y_i = c^*(x_i)$ are given, where D is the unknown distribution of the data and C^* is the unknown concept class which maps all the y_is to the x_is. We can frame this task as a computation learning theoretic problem where the goal of the learner is to approximate the unknown distribution D and output a hypothesis $h : X \rightarrow y$ for a collection of unseen data X that approximates the concept class c^*. The error of hypothesis is the mismatch between the output hypothesis and the concept class when they are applied on the input data x_i. For a predicted hypothesis $h \in H$, where

H is all the possible hypotheses, the error of hypothesis is the probability $P(h(x_i) \neq c^*(x_i)|i \in N)$. The objective of the learner is to minimize the error of hypothesis h for all examples i.e. $\min_x L = P(h(x) \neq c^*(x)|i \in N)$. Since the learner approximates the concept class and error of hypothesis for the unseen data, the learner is called **approximately correct** if $L_i \leq \epsilon$ on an example for some $\epsilon > 0$. One assumption here is that the train and test examples are chosen from the same independent and identical distribution (iid). Considering this assumption that the examples are randomly chosen from iid, there might be some cases where the hypothesis will fail to satisfy the condition of being approximately correct in other words in some cases the hypothesis will fail to be consistent with the examples. For a parameter $0 < \delta < 1$, if the hypothesis is approximately correct (the hypothesis error $L < \epsilon$) at least $1 - \delta$ fraction of times (or approximately incorrect at most δ fraction of times), the hypothesis is called **probably approximately correct**. In Probably Approximately Correct (PAC) learning [62] approach, the machine learner requires a lot of training examples to output a consistent hypothesis. In fact, the total number of training examples required for a hypothesis to be consistent with an error ϵ and probability $(1 - \delta)$ is expressed in equation 1.1 [21]:

$$m \geq \frac{1}{\epsilon}\left(ln|H| + ln\frac{1}{\delta}\right) \tag{1.1}$$

The number of examples required (m) to guarantee consistent hypothesis is called the **sample complexity**. *Haussler et al.* [21] explains several problems with PAC learning. For example, a PAC learning model to converge to a correct hypothesis requires the examples to be noise-free. The addition of noisy data hinders the capability of the learner to discriminate between correct hypotheses and incorrect hypotheses. The author also mentions that calculating the sample complexity for even a relatively simple learning algorithm is very difficult because the total number of hypotheses $|H|$ is difficult to estimate. In such cases, we are forced to insert the upper bound of hypothesis space in equation (1.1) which inflates the sampling complexity of the learner. We can underline two significant drawbacks of PAC learning concerning sampling complexity then,

- For most learners, the calculated sampling complexity overestimates the error of hypothesis [21].
- In real data sets, noisy data and randomly choosing noisy examples for training hinder learning.

This type of learning is called *passive learning* because the machine learner is ignorant of the *informativeness* [51] of the examples. *Settles et al.* [50] point out that we can find redundancies within each class in massive data sets, and due to the presence of noisy data, all examples are not representative of a particular class. It is important to sample only the most representative and informative examples from each class.

1.1.2 Active Learning

AL [8, 10, 20, 46, 50, 57] proposes a machine learning technique where the learner intelligently enquires the usefulness of the examples for learning. In active learning, the learner asks intelligent questions to sample the most useful examples from the complete data set. By removing noisy examples from the training process and including only the informative examples the machine learner can learn as good as in passive learning but with less training data. AL is advantageous in scenarios where labeling the data is costly, either due to financial restrictions or lack of access to the experts or resources.

The setup of active learning is based upon the ability of a machine learner to query the labels of examples it decides to include in the training set. The assumption is that the unlabeled examples are either cost free or very easy to acquire. The labeling system of the selected samples take the form of an annotator (either human or computer) also known as an *Oracle*, providing labels for the unlabeled data.

Different settings of active learning can be formed to allow the learner to ask intelligent questions. The two most common settings are stream-based selective sampling and pool-based active learning. In stream-based selective sampling one unlabeled example is sampled at each iteration and the machine learner decides either to query its label or to reject it. The decision is made based on query strategies like setting a threshold on the *informativeness measure*. In pool-based active learning, all the unlabeled instances are collected at the beginning of the task. The pool is assumed to be static that is new unlabeled data are not introduced to the pool after the active learning begins. Although this is not a necessary restriction. In pool-based sampling, the learner ranks all the unlabeled instances according to some *informativeness measure* and samples the ones with the best scores. In this thesis, all the experiments are formed in a pool-based active learning setup. The crux of active learning is the query strategy or sampling method that helps to train a machine learner with very few examples. Various sampling techniques have been proposed for active learning. We discuss them in de-

tail in Chapter 2. In this thesis, we use use a novel representative sampling technique called *Sigma point sampling* for extracting the informative examples from the unlabeled pool. we later use the extracted subset for training the active learner.

1.2 Scalability in Unsupervised Learning

Unsupervised learning such as clustering, Gaussian Mixture Model (GMM) are notoriously unscalable for high dimensional massive data sets. The clustering algorithms are mostly $\mathcal{O}(N^2)$ on the size of the data set. GMM has a time complexity of $\mathcal{O}(NKD^3)$ [40], where N is the size of the data set, K is the number of components and D is the dimension of the data. The cubic complexity on the dimension comes from the co-variance matrix inversion in GMM. Due to this polynomial dependency on either the dimension of the data or size of the data set, these algorithms are very inefficient. One of the proposed methods to solve this problem is extracting a meaningful subset of the complete data set that summarizes it [3, 5–7, 16, 17, 35]. The representative subset is called a coreset if it approximates the loss on the complete data set. Coresets have been proven to be very effective in scaling unsupervised learning algorithms [7].

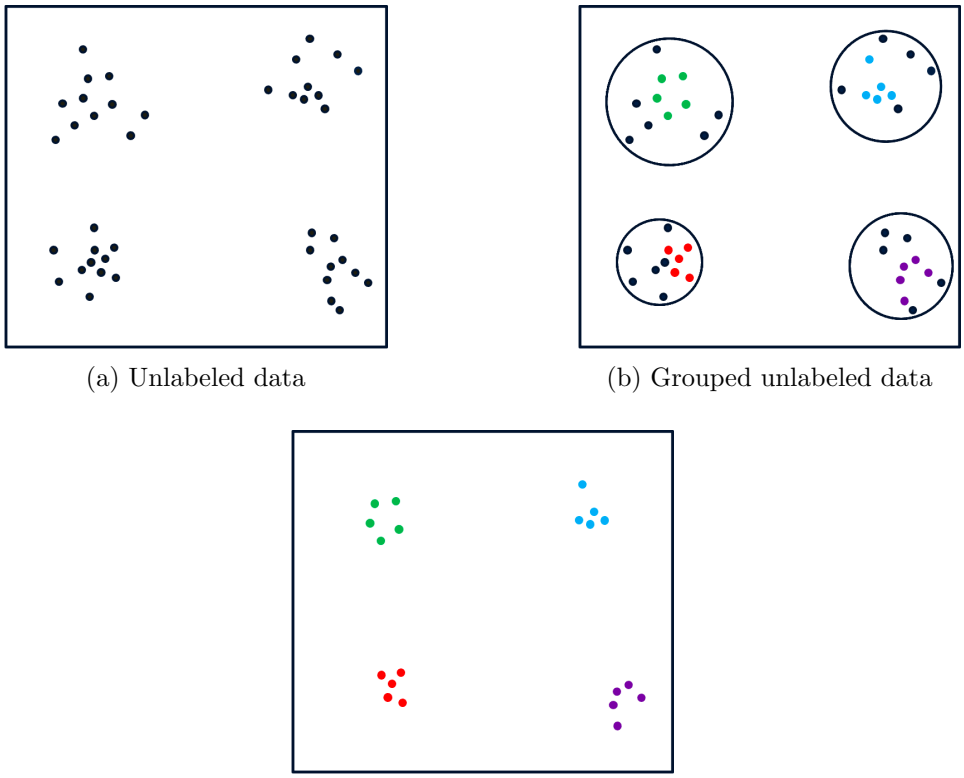
In this thesis we solve this problem as follows:

- We propose a novel technique for representative subset construction that captures the underlying distribution of the data set from a latent space and can be scaled on massive high-dimensional data sets.
- The proposed sampling technique is very easy to implement and the sampled subset is a very good representative of the complete data set which can be used for further machine learning.

1.3 Thesis Proposal

In this thesis, we dive deeper into the existing research and propose a potentially better approach for representative subsetting. We also discuss how the representative subset constructed using our approach improves active learning. The objective of active learning is to train a machine learning model using a subset of the most *informative* labeled data points from a data set that comprises of a very high number of examples. It addresses the issue that a budget constraint might limit acquiring labels for the data. Also, even if collecting unlabeled data is easy, the unlabeled data set might consist of

examples that are either redundant or noisy. Active learning model should be able to avoid those examples by using an effective query algorithm.



(a) Unlabeled data

(b) Grouped unlabeled data

(c) Selected samples

Figure 1.1: Sub-figure (a) shows unlabeled data. (b) shows a rough estimate of the groups present in the data achieved through a very efficient algorithm e.g. initialization step of k-means++. (c) shows the selected samples from each group

The goal of representative subset is to represent a complete data set using a smaller set (in some cases a subset) of points that summarizes the complete data set. A naïve approach to constructing a representative subset is randomly selecting the members. Noisy training data in the randomly sampled subset is a bottleneck for achieving optimum performance for a machine learner. For random selection, the minimum coreset size required to achieve satisfactory result can be calculated as equation (1.1). Figure 1.1 shows a better approach to this problem. In this sub-sampling technique the complete data set is partitioned into clusters and some of the examples are

sampled from each cluster based on pre-defined sampling strategy. *Pourkamali et al.* [41] use a similar approach to prove that the subset summarizes the complete data set well and can be used for active learning. *Bachem et al.* [6] on the other hand propose another approach of representative sampling called coreset constriction that summarizes massive unlabeled data sets for scaling unsupervised learning which sometimes can be computationally expensive (Section 1.2).

Even though the existing representative sampling techniques are useful for effective representative subset generation, none of them aim to capture the underlying distribution of the data set in the representative subset. [41] uses K-means to distribute the representative subset candidates throughout the data set proportionally. [5],[6] constructs the coreset (representative subset) in such a way that it approximates the cost of optimization of the complete unlabeled data set in the subset. These representative subset construction techniques query the best candidates and add them to a subset. They do not take into consideration that adding data directly from the unlabeled pool might make the representative subset susceptible to noise.

This thesis proposes a very effective sub-sampling technique that approximates the underlying data distribution of the complete data set. Sigma points selection is a very popular tool used in Unscented Kalman Filters (UKF) [63]. It has shown to be very effective in capturing the statistics of the distribution that goes through non-linear transformations in estimation theory in different applications, such as object tracking and navigation [61, 65]. [33, 60, 66] use sigma points in UKF for the estimation of the unknown state-space in deep learning. UKF, which is a derivative-free approach, is very easy to compute and it is more effective than other Kalman filters.

In this thesis, we combined sigma point sampling with GMM to estimate the priors of the data set and divide the data into soft clusters. We use the soft-clusters to estimate the density and distribution of the data and use the distribution parameters to compute the sigma points. In our research, the collection of sigma points act as the representative subset of the data set. Finally, we use the coreset in a pool-based active learning scenario. The benefits of using sigma point sampling over other representative sampling approaches are as follows:

- Firstly, in our approach, the representative subset approximates the distribution of the complete data set with very few data points. We empirically show that approximating the distribution is a better approach than the existing techniques and leads to better accuracy.

- Secondly, we use the sigma points instead of the actual data to train our active learner, which reduces the risk of including the inherent noise from the data points themselves.
- Thirdly, the sigma point sampling approach is parameterized which adds more control to the user while constructing the representative subset.

In the upcoming chapters, this thesis discusses in detail the following hypotheses:

- Sigma point sampling is a novel framework for representative subset construction that estimates the underlying distribution of the complete data set.
- Sigma point representative subset outperforms the other representative sampling approaches by measuring and comparing their performance in active learning.
- Sigma point representative subset can be used along with a dimensionality reduction tool to implement it on massive high dimensional data sets.
- Sigma point sampling can be used in both classification and regression, and it overcomes some of the challenges of the existing active learning frameworks.

1.4 Thesis Outline

This thesis is organized as follows: Chapter 2 describes the background of sigma point sampling and active learning. The first section of Chapter 3 discusses the application of sigma point sampling in classification of handwritten digits. The second section of Chapter 3 shows an application in regression. Chapter 4 delves into the details of using sigma point sampling with transfer learning. Chapter 5 discusses the experiments to show the validity of the hypotheses. Finally, Chapter 6 summarizes the findings of my research and concludes the thesis.

Chapter 2

Background

In this chapter, we first discuss the different sampling techniques in active learning. Then we discuss the working of unscented transformation. We compare the sigma point sampling technique with the sampling strategies discussed in this chapter. Chapter 5 discusses the comparison of different sampling techniques with our sigma point sampling graphically. Because our representative sampling strategy is not coreset based sampling, we compared it with other representative sampling strategies as well as uncertainty sampling and query-by-committee which are good choices of sampling in active learning.

2.1 Problem Statement in Pool-based Active Learning

Settles et al. [51] summarize the pool-based active learning setup, where collecting unlabeled data is easy, but labeling them is costly. Pool-based active learning starts with collecting all the unlabeled input data to create the unlabeled pool $\mathcal{U} = \{x_i\}_{i=1}^{u}$ and a labeled pool $\mathcal{L} = \{x_i, y_i\}_{i=1}^{l}$ where $(u + l)$ is the total number of input instances, and y_i is the label for x_i. The learner outputs a hypothesis $h_0 : \mathbf{X} \rightarrow y$ by training on the latest $\mathcal{L}$ at the first iteration. In the next step, the learner asks the label for the most *informative* instance $x^* \in \mathcal{U}$ and adds the (x^*, y^*) pair to $\mathcal{L}$. The learner trains on the new labeled space $\mathcal{L} + (x^*, y^*)$ and outputs a new hypothesis $h_1 : \mathbf{X} \rightarrow y$ in the second iteration. This iterative process continues until the learner exhausts the labeling budget $\mathcal{B}$.

2.2 Uncertainty Sampling

[11, 30, 31] use uncertainty sampling to choose points closest to the decision boundary of a binary classifier carrying the lowest confidence from the unlabeled pool. In this pool-based query strategy the learner ranks the examples based on the class confidence and the examples with confidence closest to

0.5 are queried. Equation (2.1) shows the sampling strategy.

$$x^* = \underset{x}{\operatorname{argmax}}\big(1 - P_\theta(\hat{y}|x)\big) \tag{2.1}$$

where, $\hat{y} = \underset{y}{\operatorname{argmax}} P_\theta(y|x)$ and x^* is the query strategy.

Scheffer et al. [48] propose a multi-class variant of least-confident sampling called *margin sampling*, which selects points with minimum probability difference between the most and second most probable classes($\hat{y}_1$ and $\hat{y}_2$ respectively) to train the classifier. Equation (2.2) shows the mathematical format of margin sampling.

$$x^* = \underset{x}{\operatorname{argmin}}\big(P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x)\big) \tag{2.2}$$

Entropy sampling [52, 53] is a generalization of uncertainty sampling strategies. In this sampling approach, entropy is calculated for each example in the unlabeled pool in each iteration for all possible labels, and the example with the highest entropy is added to the labeled pool. Equation (2.3) shows the entropy sampling.

$$x^* = \underset{x}{\operatorname{argmax}} - \sum_i P_\theta(y_i|x) \log P_\theta(y_i|x) \tag{2.3}$$

2.3 Query-By-Committee

Seung et al. [54] propose QBC which works by creating a committee of models $\mathcal{C} = \{\theta^{(1)}, \theta^{(2)}, \cdot, \theta^{(\mathrm{k})}\}$ and chooses samples from $\mathcal{U}$ on which the committee members disagree the most. Different strategies can be used to quantify the degree of disagreement, e.g., *Dagan et al.* [12] propose *vote entropy* sampling to train a Hidden Markov Model classifier for parts-of-speech tagging. *Mccallumzy et al.* [36] use Kullback Leibler (KL) divergence to measure member disagreements for text classification. For a samples x, all possible labels $y_i = \{y_1, y_2, \cdot, y_N\}$ and a total of k committee members the query strategy can be expressed as equation (2.4).

$$VE = \underset{x}{\operatorname{argmax}} - \sum_i \frac{V(y_i)}{\mathrm{k}} \log \frac{V(y_i)}{\mathrm{k}} \tag{2.4}$$

where, $V(y_i)$ is the total votes for label y_i.

2.4 Representative Sampling

In representative sampling, a subset of the complete data set is used for active learning. The subset approximates the complete data set based on some criteria. Usually, the general goal of representative sampling is to include as much versatile examples as possible from the complete data set without being redundant. Several representative sampling exists in the literature and the next subsections discuss two of them.

2.4.1 Coreset

A coreset is a computational geometric approach [1] that was introduced as an approximation technique to extract a small most useful (core) point-set $\mathcal{C}$, from a collection of points $\mathcal{D}$ such that the result from an expensive algorithm applied on $\mathcal{C}$ can be translated to $\mathcal{D}$. Most of the applications of coreset sampling in machine learning are either for scaling clustering algorithms on large data sets or parallelizing the computation of clustering algorithms [3–5, 7, 16, 18, 34]. In that sense, coreset is a representative subset or a summary of a data set that can be further used for inference. However, the interpretation of a coreset and the techniques used for coreset sampling vary from one task to another. *Bachem et al.*[6] formulate the creation of a coreset in machine learning as an optimization problem. They assumed that the input data $x \in X$ where $X \in \mathcal{R}^n$ and their corresponding weights are given. For all possible solutions $\theta \in \Theta$, the cost of optimization on the complete data set is,

$$cost(X, \theta) = \sum_{x \in X} w(x) l(x, \theta) \tag{2.5}$$

where, $l(x, \theta)$ is a additive loss function used in any machine learning objective such as logistic regression, support vector machine, linear regression, k-means clustering etc.

A weighted subset $\mathcal{C}$ of the original data set is called a ϵ-coreset of X if the following holds:

$$|cost(X, \theta) - cost(\mathcal{C}, \theta)| \leq \varepsilon cost(X, \theta) \tag{2.6}$$

Since, achieving the optimal solution is difficult, any solution $\theta \in \Theta$ is queried whose cost of optimization $cost(C, \theta)$ falls within the $(1 \pm \varepsilon) \times cost(X, \theta)$ range is used as a coreset. The simplest approach for coreset

construction is assigning equal weights to all the examples in the data set. For a data set of size M the weights of all examples are $w_M(x) = \frac{1}{M}$. The cost of optimization on complete data set for a query $\theta \in \Theta$ is,

$$cost(X, \theta) = \sum_{x \in X} \frac{1}{M} l(x, \theta) = \mathbb{E}_x[l(x, \theta)] \tag{2.7}$$

where, $x \in X$ is chosen uniformly at random. If a weighted subset $\mathcal{C}$, of size m is chosen uniformly at random from X and the weights of the points in the subset is equal to $w_{\mathcal{C}}(x) = \frac{1}{m}$, then the cost function $cost(C, \Theta)$ is an unbiased estimator of $cost(X, \Theta)$ for any $\theta in \Theta$ if the following holds,

$$\mathbb{E}_{\mathcal{C}}[cost(\mathcal{C}, \Theta)] = \sum_{x \in \mathcal{C}} \frac{1}{m} \mathbb{E}_x[l(x, \theta)] = cost(X, \Theta) \tag{2.8}$$

It can be shown that the variance of $cost(\mathcal{C}, \Theta)$ is finite, which means that when $m \to \infty$ it converges to $cost(X, \Theta)$. But this convergence makes m impractically large and not any better than the size of training data set (m) in PAC learning (1.1). In order to make the sampling process better the authors propose *Importance sampling* which assigns more weights to the more important examples. This assignment of non-uniform weights to examples causes a subset $\mathcal{C}$ to estimate the cost of complete data set with less examples. This algorithm of coreset construction takes into consideration only the multiplicative error $(\varepsilon \times cost(X, \theta))$ as shown in equation (2.6). *Bachem et al.* [7] improves this approach of coreset construction to include both multiplicative and additive error.

2.4.2 K-means Based Coreset

Another method to form a representative subset is to add different variations of samples from the data distribution without being redundant. *Pourkamali et al.* [41] shows an example of this method. The authors propose a k-means based sub-sampling technique on the latent space of an image data set for active learning. The authors show that features extracted with VAE can be used for active learning. Their geometric approach is easier to implement in the latent space and scalable on large data sets. The authors map the real data with dimension D into a lower dimensional space d for all n points in the data set $\mathcal{Z} = \{z_1, z_2, \cdots, z_n\} \forall z_i \in \mathbb{R}^d$. Then they partition the latent variables into K clusters. Given that the sampling budget for active learning is set to $\mathcal{B}$, $m = \frac{\mathcal{B}}{K}$ points are sampled uniformly at random from

each cluster. This process of coreset construction, minimizes the maximum Euclidean distance between the coreset $z_j \in \mathcal{C}$ and the rest of the examples $z_i \in \mathcal{Z} \backslash \mathcal{C}$. The authors achieve better test accuracy with the latent variables compared to the observed variables in active learning.

2.5 Unscented Transform

Unscented Transformation (UT) [24] is a popular method for estimating the parameters of a system that goes through a non-linear transformation, such as in integrated navigation and robotics. *Sigma points* in UT capture the Gaussian approximation of the current state and help estimate the distribution parameters of the new state after non-linear transformation. Contrary to traditional probabilistic sampling algorithms such as the Monte Carlo sampling method [14], UT estimates an unknown distribution with only $(2 \times N + 1)$ sigma points for an N-dimensional data set (Figure 2.1). *Wan et al.* [65] use the unscented kalman fiter UKF, a use case of UT, to estimate the prior distribution (mean and covariance) of the current state, and after non-linear transformation, approximate the posterior mean and covariance accurately for both Gaussian and non-Gaussian input.

Merwe et al. [63] generalize derivativeless and deterministic sampling-based Gaussian approximation filters under Sigma Point Kalman Filters (SPKF). The proposed sigma point calculation in SPKF is a modification of *scaled unscented transform* [23] to remove the scaling errors over increasing dimensions. Consider a random variable $\mathbf{x}$ of dimension N with mean $\mu_{\mathbf{x}}$ and covariance $\Sigma_{\mathbf{x}}$, which propagates through a non-linear function $\mathbf{y} = f(\mathbf{x})$. A sigma point matrix $\mathcal{X}$ is formed with dimension $(2N + 1 \times N)$ to estimate the statistics of $\mathbf{y}$. Equation (2.9) shows the calculation of sigma points.

$$
\begin{aligned}
&\mathcal{X}_0 = \mu \\
&\mathcal{X}_i = \mu + \left(\sqrt{(N+\lambda)\Sigma}\right)_i \quad i = 1, \cdots, N \\
&\mathcal{X}_i = \mu - \left(\sqrt{(N+\lambda)\Sigma}\right)_{i-N} \quad i = (N+1), \cdots, 2N \\
&\\
&\mathcal{W}_0^{(m)} = \frac{\lambda}{(N+\lambda)} \\
&\mathcal{W}_0^{(c)} = \frac{\lambda}{(N+\lambda)} + (1 - \alpha^2 + \beta) \\
&\mathcal{W}_i^{(m)} = \mathcal{W}_0^{(c)} = \frac{1}{2(N+\lambda)} \quad i = 1, \cdots, 2N
\end{aligned}
\tag{2.9}
$$

One of the sigma points is the mean μ itself, and the rest of the $2N$ sigma points spread out around the mean symmetrically using the square root of scaled covariance term $\sqrt{(N+\lambda)\Sigma}$, where $\lambda = \alpha^2(N+\kappa) - N$. The calculation of sigma points is parameterized with α, β, and κ. α is the primary scaling parameter, and κ is the secondary scaling parameter that controls the spread of the sigma points. In our experiments we vary α for scaling the sigma points. β is responsible for capturing prior distribution information. For Gaussian prior $\beta = 2$ is recommended by [63].

A higher value of α weights the mean μ more than the other sigma points and spreads the points further from μ than in the case of smaller α.

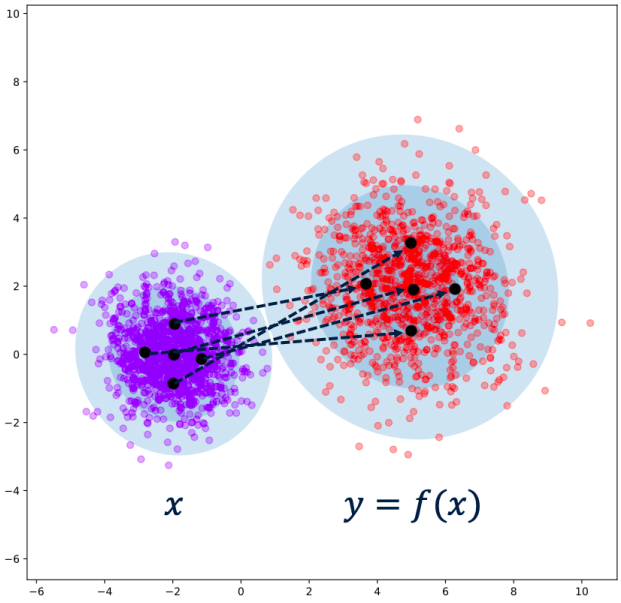


Figure 2.1: Efficient prediction of unknown distribution of a non-linear function $y = f(x)$ using Unscented Transform. For 2-dimensional data it uses $(2 \times 2 + 1) = 5$ sigma points in x [63].

This chapter outlines the concepts of active learning and the sampling strategies used in active learning. We delved into the details of representative sampling and its usage in active learning. We also briefly discussed the background of sigma point sampling. In the next chapter, we will discuss usage of sigma point sampling in active learning and build a pipeline to implement it on high dimensional massive data sets.

Chapter 3

Sigma Point Sampling, a Strategy for Improved Active Learning

In this chapter, we explain the integration of the novel representative sampling approach in active learning. *Pourkamali et al.* [41] show that geometric approach for sampling works better on the latent space than the observed variable space. To that end, in my experiments (Chapter 5), we use sigma point sampling on the latent space. To generate the latent space, on the MNIST data set [29] we use Variational Autoencoder (VAE) which we explain briefly in the first section of this chapter and on the other data set (Section 4.1.2) we use multi layer perceptron as an encoder.

3.1 Variational Autoencoder (VAE)

We use VAE to create latent space of image data set because of its ability to learn non-linear representations of the data. *Kingma et al.* [25] introduced VAE as a directed probabilistic graphical model that replaces the deterministic nature of Auto-encoders with stochastic data generation, which is more robust as a generative model. They solve the intractability problem of the marginal likelihood of data $p(x)$ while calculating the posterior distribution $p_\theta(z|x)$ by approximating it with a neural network architecture *(inference model)* with Probability Density Function (PDF) $q_\phi(z|x)$. Reconstructed data is generated from z with PDF $p_\theta(x|z)$ from another neural net architecture *(generator model)*. We can treat z as a hidden state representation of the original data, making the whole architecture autoencoder-like with $q_\phi(z|x)$ being stochastic encoder and $p_\theta(x|z)$ being stochastic decoder. *Kingma et al.* [26] describes VAE with figure 3.1. Evidence Lower Bound (ELBO) on the marginal likelihood of data is the objective function of VAE and is given by (3.1).

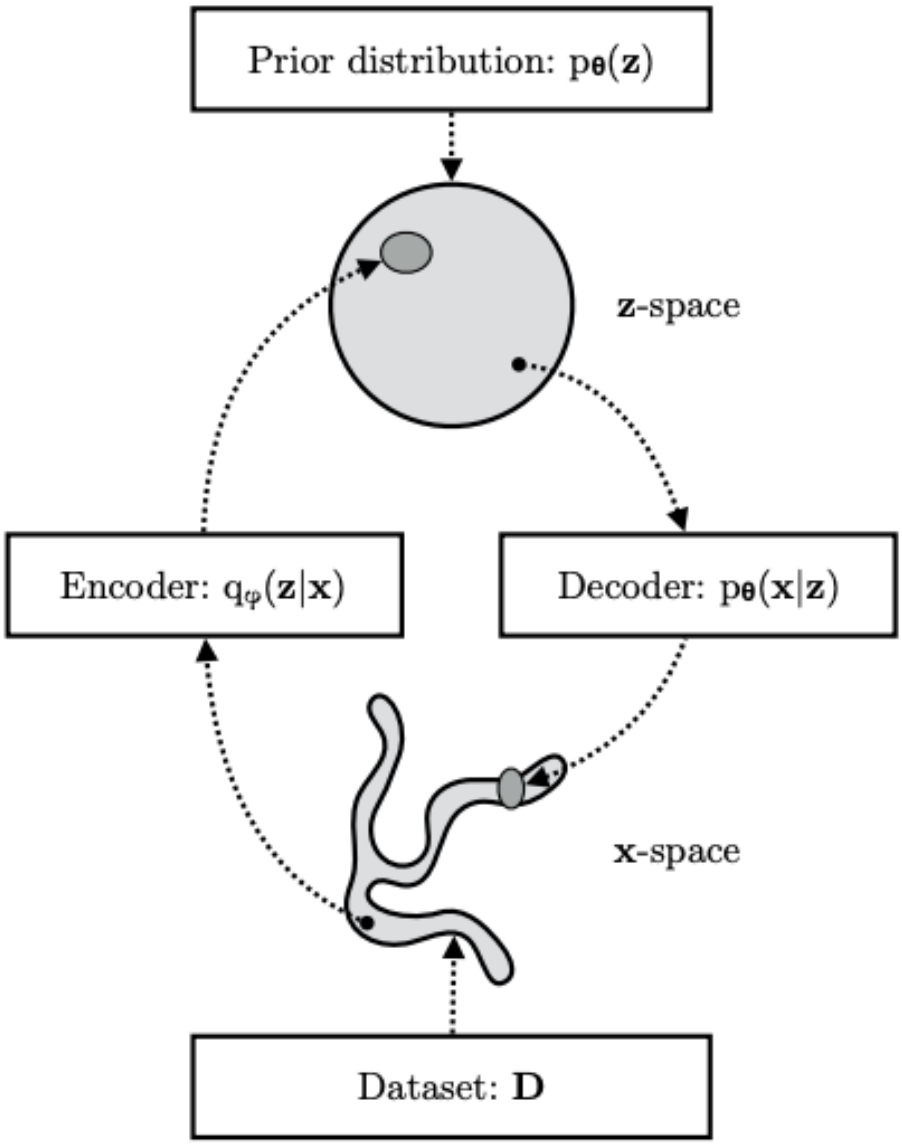


Figure 3.1: The data set distribution D lies in a high dimensional x-space while the latent space is relatively simpler and low dimensional (figure shows a spherical latent space to emphasize the simplicity of latent space and the complexity of the observed variable space). The generator model estimates the joint distribution $p_\theta(x, z)$, which for a isotropic Gaussian distribution can be factorized into $p_\theta(x, z) = p_\theta(z)p_\theta(x|z)$. $p_\theta(z)$ is the prior distribution which we assume to be $p_\theta(\mathbf{z}) \sim \mathrm{N}(\mathbf{z}; 0, \mathbf{I})$ and $p_\theta(x|z)$ is the stochastic decoder. The stochastic encoder $q_\phi(z|x)$ approximates the true posterior distribution $p\theta(z|x)$. [26]

$$\begin{aligned} &\mathrm{L}(\theta, \phi; \mathbf{x}^{(i)}) = \\ &- D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})\big) + \mathrm{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})] \end{aligned} \tag{3.1}$$

The first term on the right-hand side of equation (3.1) is a regularizer term that checks whether the approximate posterior distribution of latent variable $\mathbf{z}$ is the same as its prior distribution, i.e., $p_\theta(\mathbf{z})$. The common practice is to draw prior from isotropic Gaussian, $p_\theta(\mathbf{z}) \sim \mathrm{N}(\mathbf{z}; 0, \mathbf{I})$ and the output of the encoder $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ is $\mu^{(i)}$ and $\sigma^{(i)}$. The second expression in equation (3.1) is the reconstruction term and expressed as the log-likelihood of $p_\theta(\mathbf{x}^{(i)}|\mathbf{z})$ over the posterior distribution of latent variable $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$. By optimizing $\mathrm{L}(\theta, \phi; \mathbf{x}^{(i)})$ with respect to θ and ϕ both the generator and the inference model are optimized.

Because of the effectiveness to learn any complex function of neural networks, it makes the probabilistic encoder-decoder architecture an excellent tool to learn non-linear representations of the hidden variables.

3.2 Sigma Point Sampling on Latent Space

We use the sigma point sampling to construct a representative subset and train a learner in a pool-based active learning scenario. Sigma points can capture the mean and covariance of a completely unknown distribution with decent accuracy with much fewer samples than Monte Carlo-type sampling techniques used in particle filters [27]. We show that sigma point sampling along with GMM can summarize a large data set with unknown manifolds effectively. Figure 3.2 shows our coreset construction approach using sigma points.

The objective of the representative subset is to find a collection of points that capture the underlying distribution of the complete data. To tackle the high dimensionality of our data, we use a VAE. VAE can stochastically fit to any random distribution and approximate the hidden variables in a low dimensional latent space better than traditional dimensionality reduction algorithms [13]. Even after the feature space is formed using VAE, the latent manifolds have complex properties that are difficult to capture with a single isotropic Gaussian assumption of VAE. To resolve that, we apply the GMM [47] on the latent space to approximate the latent manifolds with a mixture of known distributions and create soft clusters. GMM represents data as a weighted sum of Gaussian distributions. Equation (3.2) shows the

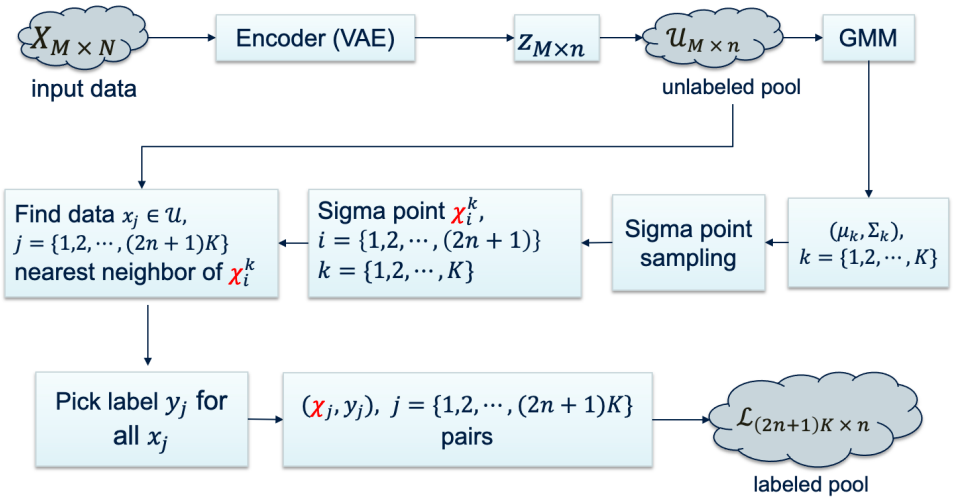


Figure 3.2: Representative querying using sigma point sampling. Sigma points $\mathcal{X}$ are highlighted in red. $x_j \in \mathcal{U}$ are unlabeled data in the encoded latent space. A Sigma point takes label from their nearest neighbor.The $(\mathcal{X}_j, y_j)$ pair is added to the labeled pool for training active learner.

representation of data x as a mixture of Gaussian.

$$p(x|w,\mu,\Sigma) = \sum_{k=1}^{K} \frac{w_k}{(2\pi)^{n/2}|\Sigma_k|^{\frac{1}{2}}} exp\Big\{ -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\Big\} \quad (3.2)$$

where K is the total number of Gaussian components used to represent an n-dimensional data set x and $\sum_{k=1}^{K} w_k = 1$. Given the mean vectors and covariance matrices of the soft clusters, (μ_k, Σ_k), $k = \{1, 2, \cdots, K\}$ parameters for soft cluster k, we use equation (2.9) to calculate the sigma points for that cluster. We sample $2n+1$ sigma points from each cluster for the n-dimensional latent space acquired from the VAE. Given that we use K Gaussian mixtures to represent the latent space, we get $\sum_{k=1}^{K}(2n+1) = K(2n+1)$ sigma points in total. Equation (3.3) shows the calculation of sigma points from $k = \{1, 2, \cdots, K\}$ clusters of the mixture of Gaussians.

$$\begin{aligned}
\mathcal{X}_0^k &= \mu_{\mathbf{k}}, \quad k = \{1, 2, \cdots, K\} \\
\mathcal{X}_i^k &= \mu_{\mathbf{k}} + \left(\sqrt{(n+\lambda)\Sigma_{\mathbf{k}}}\right)_i, \quad i = 1, \cdots, n \quad \text{and} \quad k = \{1, 2, \cdots, K\} \\
\mathcal{X}_i^k &= \mu_{\mathbf{k}} - \left(\sqrt{(n+\lambda)\Sigma_{\mathbf{k}}}\right)_{i-n}, \quad i = (n+1), \cdots, 2n \quad \text{and} \quad k = \{1, 2, \cdots, K\}
\end{aligned} \tag{3.3}$$

After we acquire the sigma points, we find their nearest neighbors to estimate their labels from the real data set. Because we run this operation on the latent space, the time complexity is $\mathcal{O}(n \times K(2n+1) \times M) \approx \mathcal{O}(M)$. The final coreset comprises of $(\mathcal{X}_j, y_j)$ pairs where $j = 1, 2, \cdots, K \times (2n+1)$. We use support vector machine (SVM) with grid search cross validation to train on the coreset.

Algorithm 1: Active learning with sigma point sampling

1: **Inputs:** Input data $x \in \mathbf{X}_{(M \times N)}$, Query budget $\mathcal{B}$
2: VAE on x to create a latent space $\mathbf{z}$, with dimension $(M \times n)$, where $n \ll N$
3: Unlabeled pool $\mathcal{U} = \{z_i\}_{i=1}^{M}$, where $z_i \in \mathrm{R}^n$
4: Apply GMM on $\mathcal{U}$ with K components to create soft clusters
5: Obtain (μ_k, Σ_k), $k \in 1, \cdots, K$ for each cluster
6: Sample sigma points $\mathcal{X}_i^k$ from each cluster using the (μ_k, Σ_k), where $i = \{1, \cdots, (2n+1)\}$, $k = \{1, \cdots, K\}$
7: For each $\mathcal{X}_i^k$ find nearest neighbor $z_{j=1}^{\mathcal{B}} \in \mathcal{U}$ and query labels $y_j^{\mathcal{B}}$
8: **if** $\mathcal{B} \leq K \times (2n+1)$ **then**
9: $\quad \mathcal{L} = \{\mathcal{X}_l, y_l\}_{l=1}^{K \times (2n+1)}$
10: $\quad$ Train learner on $\mathcal{L}$
11: $\quad$ output hypothesis $h : \mathbf{z} \to y$
12: **else**
13: $\quad \alpha = \alpha$ - 0.1
14: $\quad$ Repeat step 6 to 11 with updated α
15: **end if**

Algorithm 1 shows our proposed approach. We start with a high dimensional data set $x \in \mathrm{R}^N$ and a query budget $\mathcal{B}$. We use an encoder (in our case, convolution Variational Autoencoder) to create a latent space $z \in \mathrm{R}^n$, which we treat as the unlabeled pool. We apply GMM on the unlabeled pool to retrieve the soft clusters, which we later use to sample sigma points $\mathcal{X}_i^k$. We find the nearest neighbors of the sigma points in the latent space and attach their labels to form the $(\mathcal{X}_i, y_i)$ pairs. We use the pairs as labeled data set in our active learning. Finding the nearest neighbors and using

the labels is the query step in our representative sampling algorithm. Since we only query one nearest for each sigma point and we have $K \times (2n + 1)$ sigma points in total, we can check for the query budget with the condition $\mathcal{B} \leq K \times (2n + 1)$. If it fails, we reduce α and rerun the steps from sigma point generation to retrain the model.

We choose to train the active learner on sigma points instead of the data for the following reasons. Firstly, a representative subset constructed with sigma points is a good summarization because it captures the true distribution of the data. Secondly, the sigma points are less affected by noise in the data because we can control the spread of sigma points by tuning α. Usually, making α too large causes generation of sigma points that represent noisy data. Thirdly, to query the labels for the sigma points, we use the labels of their nearest neighbour in the latent space. Because in this experiment, the query budget ($\mathcal{B}$) for active learning is set on the latent data space $x_j \in \mathcal{U}$, we can query label y_j for at most $\mathcal{B}$ examples (considering a unit cost for each query). We can adjust α to sample points with a fixed budget of $\mathcal{B}$ to satisfy $\mathcal{B} \leq K \times (2n + 1)$. We also find in our experiments that sigma points in latent space perform better than latent space data.

This chapter detailed the procedure to implement the proposed sigma point sampling to generate representative subset of a data set and use the subset for active learning. We saw the application of the pipeline on the data available to us.

To summarize,

- The proposed pipeline is applied on the latent space of the data in order to save computation by using the lower dimensional latent space.
- The latent space also captures rich features about the data extracted with an encoder which makes machine learning more effective.
- The active learning setup in the pipeline also saves computation and reduces labeling costs by applying machine learning only on a representative subset of the data.

The next chapter explains the integration of sigma point sampling with transfer learning and active learning. It also details the application of the novel pipeline on a new data set.

Chapter 4

Applications of Sigma Point Sampling in Industry 4.0

In this chapter, we show two applications of the sigma point sampling on industrial data sets for aerospace composite parts manufacturing. With the advent of advanced computing platforms and intelligent systems, manufacturing has become automated. An integral part of this automated manufacturing is data driven predictive models at different stages of the process e.g. supply, data monitoring and product quality. In this study, we focus only on designing a predictive model to maintain the quality of manufacturing on aerospace material data. In aerospace industry, the manufacturing processes take very long to be able to generate sufficient amount of product quality data, which means that such data sets are usually sparse.

Even though simulated data can be generated by using novel softwares, they are costly and time consuming to generate. Another problem in this manufacturing process is that every raw material acts differently in the manufacturing processes and models trained on one data set usually perform poorly on another data set that revolves around using a different material. This brings a possibility to design a new predictive system that is agnostic to the varying data distribution caused by different materials in the same manufacturing process. The naive approach of training a new model on every data set is not optimal to the dynamic and fast evolving needs of smart manufacturing (Industry 4.0) [28].

Manufacturing composite parts requires high levels of precision and dependency on the manufacturing tools. Constant monitoring of each step of the manufacturing process is very crucial for producing cured parts that show expected material properties. In this experiment, we show that transfer learning with sigma point sampling can help to automate the manufacturing process of cured parts. Figure 4.1 shows the curing process that we are trying to automate.

In aerospace manufacturing, raw materials are put inside an autoclave and through a curing cycle the composite parts are produced. A manufacturer usually decides the presets of the curing process, it is called Manu-

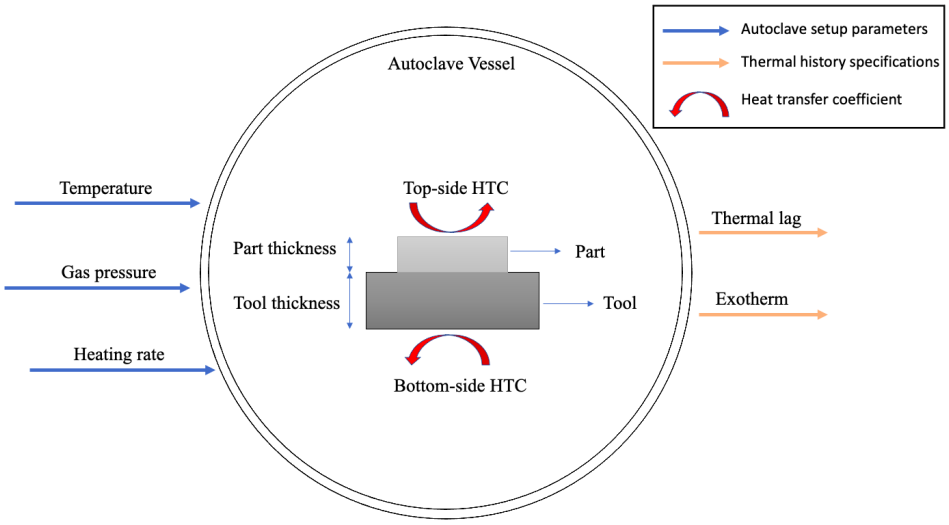


Figure 4.1: Autoclave curing process in aerospace manufacturing. [43]

facturer's Recommended Curing Cycle (MRCC). *Thermal lag* and *exotherm* are the thermal history specifications during the curing process that are the primary quality metrics to evaluate the mechanical, chemical and physical properties of the composite parts. *Thermal lag* is the deviation between the part temperature and the autoclave temperature during the cure cycle. *Exotherm* denotes the maximum temperature achieved by the part material during the cure cycle. In this study, we setup thresholds for these two metrics. Parts that exceed these thresholds are considered failed products. In the experiment, we use a binary classifier to detect whether a curing process configuration will lead to failed product or acceptable composite parts. Then we transfer the knowledge of this classifier from one resin material to another to show that our transfer learning process with sigma points sampling can detect successful or failed configuration on the new raw material with high accuracy using very few samples.

4.1 Transfer Learning on Composites Manufacturing Data

TL [67] focuses on training on a data set and transferring the knowledge to a similar data set with a different distribution. TL has proven to be very ef-

fective in manufacturing where acquiring new data distribution is common. *Xu et al.* [70] use deep transfer learning to design a fault detection system in manufacturing process by training a model on simulated data and transferring the knowledge on physical system data. *Sun et al.* [56] transfer learnt features and weights from historical failure data to new objects. The goal of this chapter is to present a novel variant of inductive transfer learning [73] that combines sigma point sampling to increase the effectiveness of transferred knowledge from one data set to another in aerospace manufacturing. The results show that we achieve significant accuracy on the target data set with very few training examples by using sigma point sampling.

Transfer learning (TL) is usually defined with domains and tasks. These terms are defined below.

Definition 1 *A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, x_2, \cdots, x_n\} \in \mathcal{X}$. Therefore, $\mathcal{D} = \{\mathcal{X}, P(X)\}$.*

Definition 2 *A task $\mathcal{T}$ consists of a label space $\mathcal{Y}$ and an objective function $f : \mathcal{X} \rightarrow \mathcal{Y}$. Therefore, given a specific domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ the task is defined as $\mathcal{T} = \{\mathcal{Y}, f\}$. f is learnt from sample data.*

Usually the source domain $\mathcal{D}_S$ for a particular task $\mathcal{T}_S$ is observed with instance-label pairs $\mathcal{D}_S = \{(x, y)|x_i \in \mathcal{X}_S, y_i \in \mathcal{Y}_S, i = 1, 2, \cdots, n_S\}$. The objective function f_S is learnt from the instance-label pairs.

Definition 3 *Given observations corresponding to a source domain and source task $(\mathcal{D}_S, \mathcal{T}_S)$ and some observations corresponding to target domain and task $(\mathcal{D}_T, \mathcal{T}_T)$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$, the goal of transfer learning is to improve the performance of target predictive function f_T on $\mathcal{D}_T$ using the knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$.*

In this definition of transfer learning we use only a single source for learning and a single target task for transferring the knowledge. A common scenario for transfer learning is where source domain consists of abundant labeled instances but labeled target domain instances are limited. Since both source and target domains have labels, the transfer learning is called *inductive transfer learning.*

In our data set on manufacturing parameters, the source domain and target domain are different ($\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$)), which makes this learning *heterogeneous transfer learning.* In smart composites manufacturing, changing domains can arise from changing the raw materials in the

manufacturing process which changes the target task ($\mathcal{T}_T$). We use transfer learning to reduce the effort and computational cost for training new models on new target data sets. TL has proven to be very effective in getting high accuracy on different but similar domains by transferring knowledge from the source domain [56, 70, 71].

4.1.1 Transfer Learning with Sigma Point Sampling

We propose a modified approach to transfer learning so that we can apply the transferred model trained on the source domain to a small subset of the target domain. Figure 4.2 shows our proposed transfer learning pipeline with sigma point sampling. At the initial step we consider that the observed instances from the target domain are unlabeled. We use feature-based transfer learning [38] on the first step of target domain. We learn latent features ($\mathcal{Z}_S$) of the source data ($\mathcal{X}_{source}$) with a fully-connected encoder (g) and apply the same encoder on the unlabeled target domain to generate similar latent space ($\mathcal{Z}_T$).

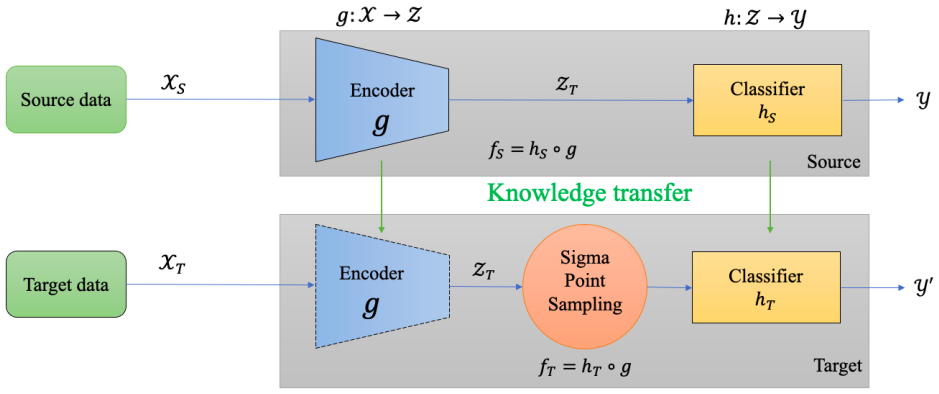


Figure 4.2: Transfer Learning with Sigma Point Sampling.

We apply sigma point sampling on the latent space ($\mathcal{Z}_T$) instead of the observed feature space for the following reasons. Firstly, [41] show that active learning with representative sampling performs better in the latent space because an encoder captures rich features from the observed variable space. In chapter 3 we apply sigma point sampling on latent space. In order to keep our experiments comparable we follow the same pipeline in this chapter. Secondly, the latent space representations in our experiments

have reduced dimension which helps implementing the sigma point sampling algorithm with reduced computation.

In the second step we use parameter-based transfer learning [38] where we transfer the source classifier model (h_S) to the target domain without freezing. Since the target task ($\mathcal{T}_T$) is different from source task ($\mathcal{T}_S$) we fine tune the classifier (h_T) here. We trained the final output layer for target domain from scratch without transferring weights from source.

We implement our TL with sigma point sampling pipeline in two stages. First, we use an encoder $g : \mathcal{X} \to \mathcal{Z}$ to generate a target latent space that is similar to the source latent space. Then we use sigma point sampling on the latent space of the target domain to generate sigma points to train the classifier (h_T). In the final step, we use a classifier $h : \mathcal{Z} \to \mathcal{Y}$. We fine-tune the weights from classifier (h_S) to fit the classification task of the target data set. The sampling strategy is similar to the process described in Section 3.2. The results of our novel technique show that active learning with representative sampling can be successfully integrated with transfer learning. We achieve very high accuracy and F1 score for detecting the defective curing process for different raw composite materials. Section 5.2 describe the experimental setup on the composites manufacturing and the results of the experiments.

4.1.2 Composite Material data set

In this experiment, we use a data set consisting of curing process configurations for aerospace composite material manufacturing. Figure 4.3 shows the two-hold autoclave curing process that include the observed variables for the curing configurations. First step is a heating process followed by an isothermal hold. This step is repeated twice until the autoclave achieves a temperature of 180°. The next step is cool-down process. The temperature of the autoclave is brought down from 180° to 20° at a rate of $3.5°/minute$. As shown in Figure 4.1, apart from the current temperature and the heating rate inside the autoclave, gas pressure also affects the quality of the composite material produced in the process. This is mostly because the physical properties of the tool material along with the gas pressure and gas velocity inside the autoclave determine the thermal history around the part material. Hence, we also keep track of the Heat Transfer Co-efficient (HTC) above and below the part material.

For our application in this experiment, we choose two aerospace-grade composite material data sets namely, AS4/8552 as the source data set and AS4/8551 with a different resin system as the target data set. The observed

variables for both the data sets are described in the Table 4.1.

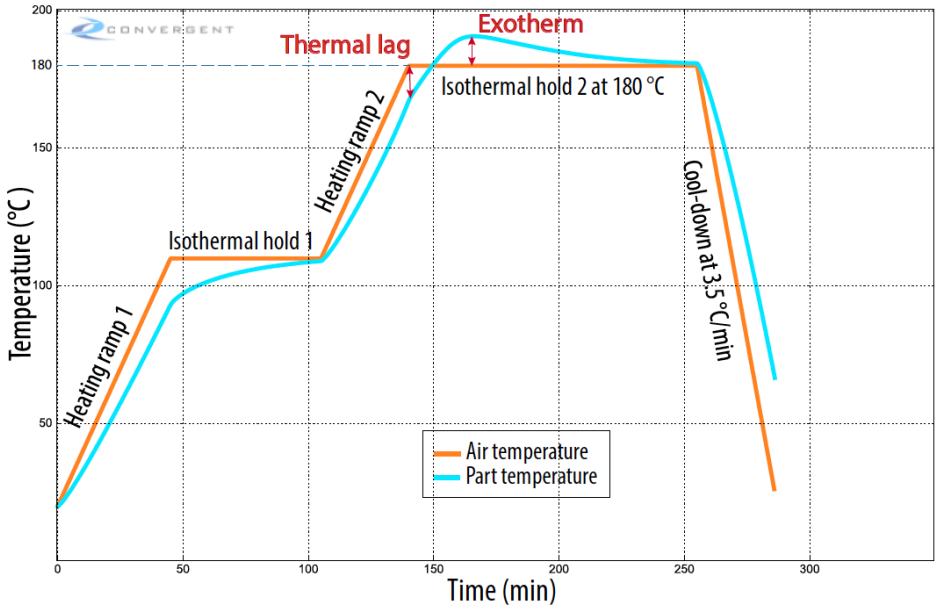


Figure 4.3: Two-step autoclave cure cycle.[45]

Table 4.1: Observed variables used in predictive modeling of composite materials data set.

Variable	**Min value**	**Max value**
Tool thickness (mm)	2.5	20
Part thickness (mm)	2.5	20
Heating rate-ramp 1 $(°C/min)$	1	5
Isothermal hold 1 temp $(°C)$	100	130
Isothermal hold 1 duration $(seconds)$	30	90
Heating rate-ramp 2 $(°C/min)$	1	5
Isothermal hold 2 temp $(°C)$	175	185
Isothermal hold 2 duration $(seconds)$	30	90
Top-side HTC (W/m^2K)	10	200
Bottom-side HTC (W/m^2K)	5	150

The data for the source and target data set were generated using RAVEN composites processing simulation software [45]. The output from RAVEN of the observed variables mentioned in Table 4.1 are two quality metrics for the composite parts namely, **thermal lag** and **exotherm**. The definitions of these two terms has been described in Chapter 4. Even though these metrics are quantitative, we frame the problem of fault detection as a binary classification problem. As per the MRCC we choose $Exotherm \leq 5°C$ and $Thermal \quad lag \leq 20°C$ as successful cases and any value above these thresholds are considered as failed cases. In all our experiments, we keep the tool material fixed to Aluminium.

4.2 Regression with Sigma Point Sampling on Composites Manufacturing Data

In this section of the thesis, we describe how easily our sigma point sampling can be adopted for linear regression instead of classification described in the previous sections. To demonstrate the usability, we use the source data set from the transfer learning study (AS4/8552). The data set consists of 1000 examples and we used 80% data for training and active learning and 20% data for testing. The explanatory variables are described in table 4.1. As shown in figure 4.1 the dependent variables are **thermal lag** and **exotherm**. To measure the performance of the linear regression model we used three different metrics namely, R^2, MAE and MSE. Whereas MAE and MSE help to understand the scatter of the residuals points around the linear regression line, R^2 helps to understand the quality of fit of the model. We achieved an upper bound R^2 value of 0.85 which shows a good fit of the multi-output linear regression model. Because the linear regression model outputs two dependent variables we used variance weighted R^2 value which takes into account all output variances individually. To further prove that the linear model fit is good, we applied threshold on both the real and predicted values of **thermal lag** and **exotherm** (dependent variables) in the test data set. The process of thresholding is described in Section 4.1 ($Exotherm \leq 5°C$ and $Thermal \quad lag \leq 20°C$ for success). Thresholding generates binary values for outputs where 0s are successful cases and 1s are failed cases.

Chapter 5

Experiments

In this chapter, we perform experiments on two data sets to empirically prove that SPS is a valid representative sampling strategy for active learning. We show that the sigma points themselves can be used for further machine learning and the subset created with sigma points are representative of the complete data set. We also perform a sensitivity analysis on the effect of the value of α on the active learning accuracy. The sensitivity analysis experiment shows the justification for choosing α.

5.1 Sigma Point Sampling on MNIST data set

In this section, we first describe the setup of our experiment. We discuss the data sets used in our experiments and the neural network architecture used in variational autoencoder for dimensionality reduction. Later we discuss the experimental results on multi-class classification tasks.

5.1.1 Experimental Setup

We compare my proposed sampling method with others using the MNIST [29] data set on three, four, five, and ten digits (labels). Increasing the number of labels in our experiment increases the size of the data set, changes the distribution of the data, and changes the manifolds of the latent space. We choose four different classification tasks on the same data set by increasing the number of labels in the experiment to demonstrate the robustness and scalability of our model. We compare sigma point sampling approach with coreset based representative sampling techniques [3, 41], least confident uncertainty sampling [30, 31], entropy sampling [48], and margin sampling, query by committee [12, 36, 54]. We report the accuracy comparisons on the test set and show that our method works either better or equally well when compared with the existing approaches. We use Support Vector Machine (SVM) [64] with grid search cross validation as the active learning classifier. Since we are using SVM for multiclass classification we use One Versus Rest (OVR) settings. We use five fold cross validation with linear and Radial

Basis Function (RBF) kernel in SVM to remove the constraint of a fixed hypothesis on the active learner.

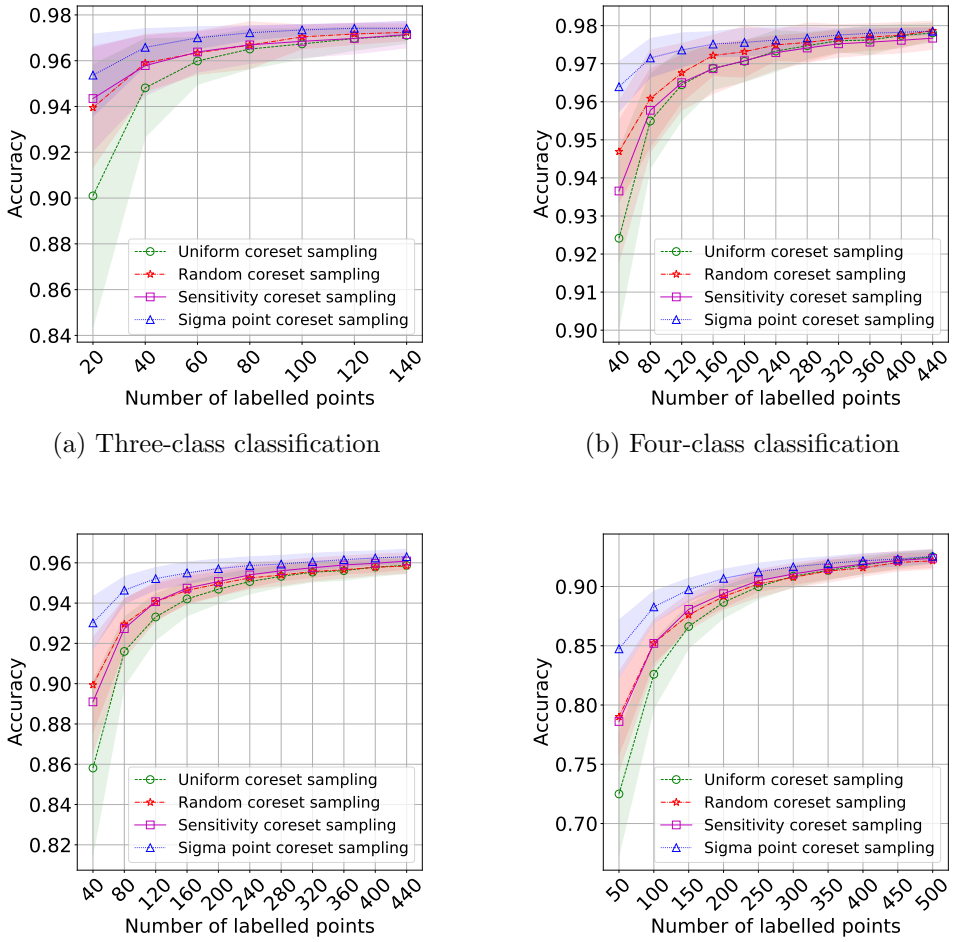
Table 5.1 shows the labels of MNIST used in our study and the size of the train and test set for each classification task. The MNIST data set consists

Table 5.1: Labels used for classification problems with their corresponding data set sizes used for training and testing.

Number of labels	Labels	Training set	Test set
3	**0,3,9**	18003	2999
4	**0,3,7,9**	24745	4134
5	**0,2,3,7,9**	30226	5059
10	**0-9**	60000	10000

of handwritten digits of dimension (28×28) [29]. We use VAE for dimensionality reduction and feature learning. We use the **Keras** framework to implement the VAE. The convolution VAE used in our experiment has four convolution layers (except the final layer) in the encoder and the decoder with 32, 64, 64 and 64 channels with Rectified Linear Unit (ReLU) activation. Maxpooling function is used in the encoder along with 25 percent dropout. Similarly, in the decoder upsampling function is used along with 25 percent dropout. The last layer of the encoder is a dense layer with 128 output channels with ReLU activation. The bottleneck layer has two consecutive dense layer without any activation[1]to estimate the parameters (mean and variance) of the latent space with a isotropic Gaussian assumption and sample the latent variable z from them. The final layer of the decoder is a convolution layer with a single output channel with sigmoid activation. This is to bound the output pixel values of the reconstructed images by VAE within zero and one. We run 100 epochs for three, four, five, and ten classes with early stopping (10 epoch patience). We use *Adam* optimizer to optimize our loss function. We use *Binary Cross-entropy* to calculate the reconstruction loss between real and reconstructed images. For three, four, five, and ten class classification we choose a latent space of dimension 3,5,5, and 7 respectively.

[1] `https://www.tensorflow.org/tutorials/generative/cvae`



(a) Three-class classification

(b) Four-class classification

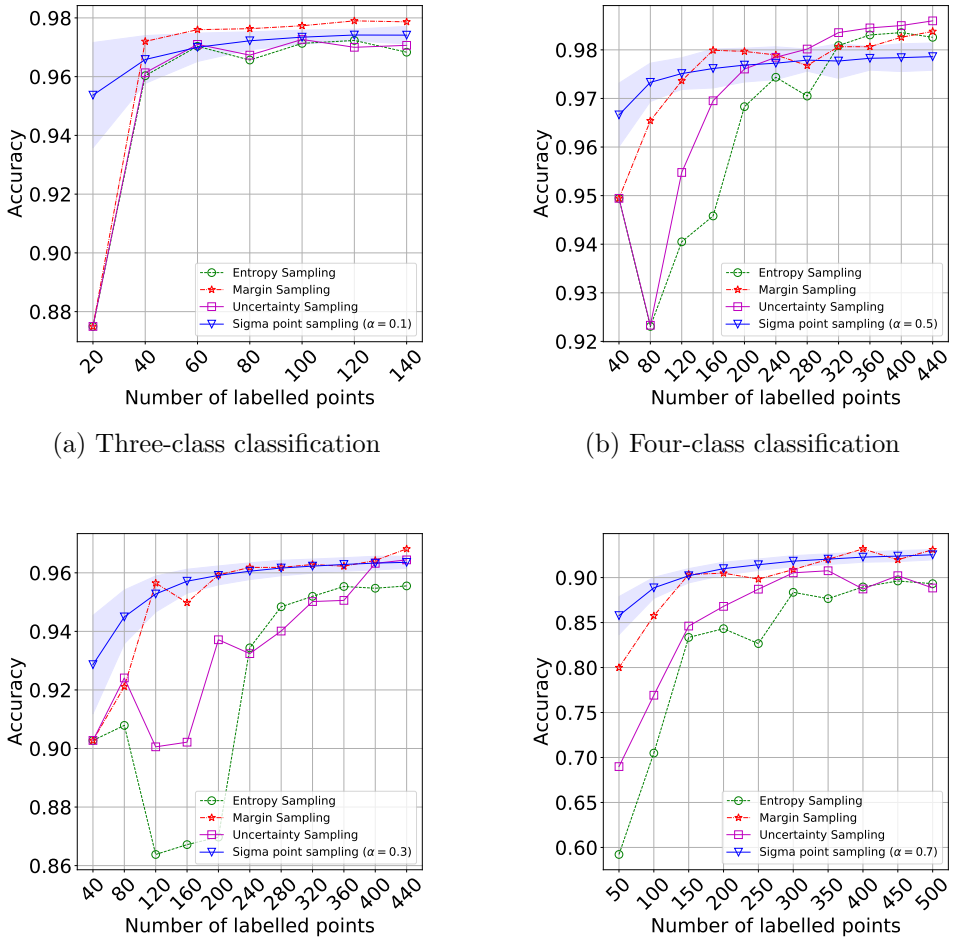
(c) Five-class classification

(d) Ten-class classification

Figure 5.1: Accuracy comparison of sigma point sampling vs. other representative sampling methods

We represent the latent space with a mixture of Gaussian distributions with K components. For three, four, and five-class classification, we use 80 components in GMM, and for ten-class classification, we use 100 components. We collect the $(\mu_k,\ \Sigma_k)$ parameters from each soft cluster, $k = \{1, 2, \cdots, K\}$. We treat these parameter pairs as input to the sigma point sampling algorithm. We know from equation (3.3) that for an n-dimensional latent space, the sigma point sampling algorithm generates $(2n + 1)$ sigma points from each soft cluster. We collect $K(2n + 1)$ sigma points from all

clusters. We run *KNeighborsClassifier* (with argument n_neighbors=1) from the *scikit-learn* package in python to find the nearest neighbor of each sigma point and predict their labels. We generate $(\mathcal{X}_i, y_i)$ pairs for all $K(2n+1)$ sigma points and construct the final representative subset $\mathcal{C} = (\mathcal{X}_i, y_i)$, for $i = \{1, 2, \cdots, K(2n+1)\}$.



(a) Three-class classification

(b) Four-class classification

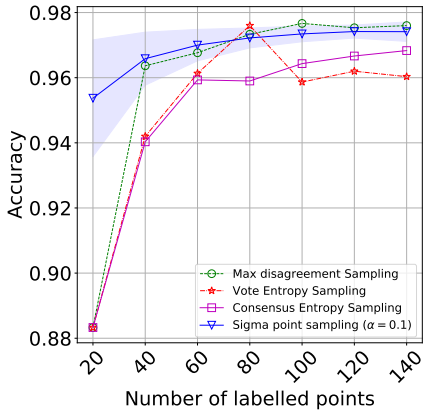
(c) Five-class classification

(d) Ten-class classification

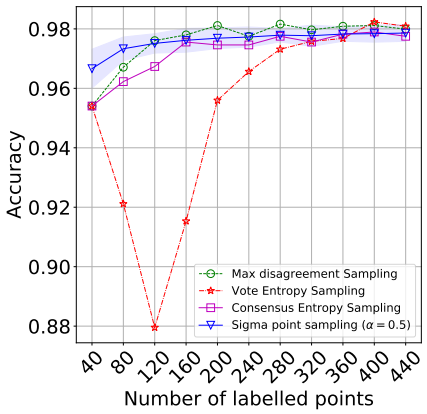
Figure 5.2: Accuracy comparison of sigma point sampling vs. maximum uncertainty sampling, margin sampling and entropy sampling

We train the active learner on this representative subset. To train the SVM we iteratively add 20, 40, 40, and 50 pairs of $(\mathcal{X}_i, y_i)$ from $\mathcal{C}$ to $\mathcal{L}$ for three, four, five, and ten-class classification, respectively. For representative

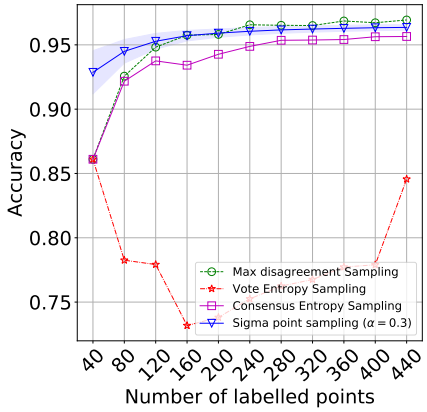
sampling we run each trial 500 times and report the average test accuracy with standard deviation for each classification task. For uncertainty sampling methods and QBC we run each of the experiments once because running them 500 times is time-consuming and we did not observe too much variation in each run. To form the committee for QBC we use three different classifiers namely, support vector machine with linear kernel, random forest classifier and K-nearest neighbour classifier.



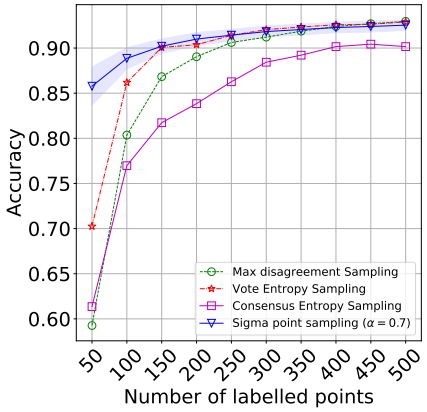
(a) Three-class classification



(b) Four-class classification



(c) Five-class classification



(d) Ten-class classification

Figure 5.3: Accuracy comparison of sigma point sampling vs. consensus entropy sampling, maximum disagreement sampling and vote entropy sampling of Query by committee.

The codes and results for our experiments can be found here. [2]

5.1.2 Experimental Results

In this section we show that sigma points can be treated as members of a representative subset, and we can use them for training learners in a pool-based active learning setup. We first compare the test accuracy of sigma point sampling with uniform sampling, k-means based random coreset sampling [41], and sensitivity coreset sampling [6] (Figure 5.1). Secondly, we compare our approach with maximum uncertainty sampling, entropy sampling, margin sampling (Figure 5.2). Then we compare the test accuracy of our approach with consensus entropy sampling [36], maximum disagreement sampling, and vote entropy sampling [12] of QBC (Figure 5.3).

5.1.3 Comparison of different sampling techniques

From Figure 5.1, we report that the test accuracy for the sigma point sampling outperforms all other representative sampling methods in all four classification tasks. We can observe from Figure 5.2 that margin sampling performs the best in most cases but sigma point sampling is really effective when we have very less training points. The test accuracy for 20 training examples in three-class classification, 40 training examples in four-class classification and five-class classification, and 50 training examples in ten-class classification are highest for sigma point sampling. We observe similar phenomena in Figure 5.3 as well. Compared to sampling techniques used in QBC, sigma point sampling either performs better or equally well on very small amount of training data.

We can summarize the drawbacks of other sampling techniques and advantages of sigma point sampling in three folds. Firstly, all sampling techniques other than representative sampling techniques query one example at a time from the unlabeled pool and calculate informativeness for every example in the unlabeled pool. Hence, they are computationally very expensive. Secondly, the committee based querying strategies are very inconsistent across our experiments. For example, in three, four, and five-class classification maximum disagreement sampling performs best (after sigma point sampling) but in ten-class classification vote entropy sampling outperforms the other committee based querying. Vote entropy sampling performs poorly in the other experiments which shows the inconsistency whereas sigma point sampling consistently either outperforms or works equally well as compared

[2] `https://github.com/anonymous-moniker/repo1`

to all other sampling techniques. Thirdly, [68] describes the necessary qualities of an ideal active learning setup which include removing redundancy of data in the labeled pool, including all variations of data distribution in the labeled pool. Even though random coreset sampling [41] attempts to include different data distributions in the labeled pool it fails to reduce redundancy and some of the examples in the labeled pool might be outliers. Sensitivity sampling however focuses on sampling examples that best optimizes the cost function.

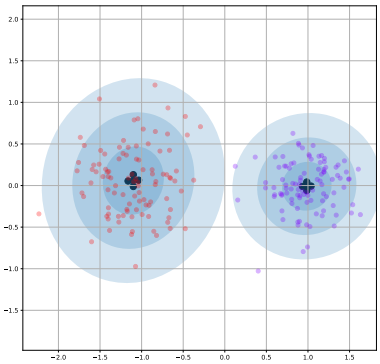
Sigma Point Sampling on the other hand, captures all variations of data distribution through GMM and we include at least one member from each Gaussian component. Because we use sigma points for training our model we reduce redundancy of data distribution as a higher value α spreads the sigma points within the data distribution. When compared to the other representative sampling strategies, sigma point sampling outperforms them.

5.1.4 Sensitivity Analysis

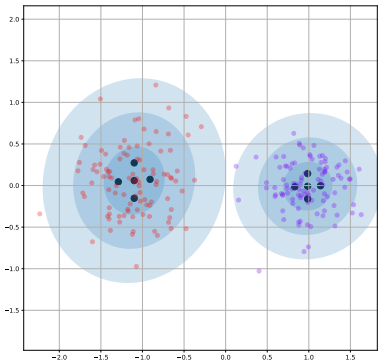
In Section 3.2, we argue that the representative subset can be constructed with a maximum of $\mathcal{B}$ queried labels (considering unit cost for each query) from real data space to accommodate the query budget $\mathcal{B}$ for active learning. Algorithm 1 shows that, we tune the value of α to accommodate the query budget. In this section, we discuss the effect of the primary scaling parameter α of sigma point sampling on the test accuracy of the active learner. We show that the test accuracy is optimum for a certain range of α, for a specific classification task. Figure 5.4 shows the variation of the spread of sigma points across two Gaussian distributions for a two-dimensional toy data with increasing α. The black dots represent the sigma points. It is evident from the figure that as the value of α increases the sigma points spread more.

In our experiments, we vary α between 0.1 and 0.9, as recommended in the original paper [63]. Higher value of α produces sigma points with higher scaled covariance and the sigma points spread further, which is validated in equation 3.3. Figures, 5.5, 5.6 and 5.7 show the spread of sigma point on the 2 dimensional latent space of MNIST data set (with digits 0,3,9) and how the sigma points represent the latent space for varying values of α. 5.5 shows the overlap of the sigma points (shown using black dot markers) on latent space data (shown with coloured square markers). We can see that the density of the data points in the latent space is not homogeneous. We use 80 Gaussian components to estimate the density of this latent space and the Gaussian components in the sparse regions have higher covariance

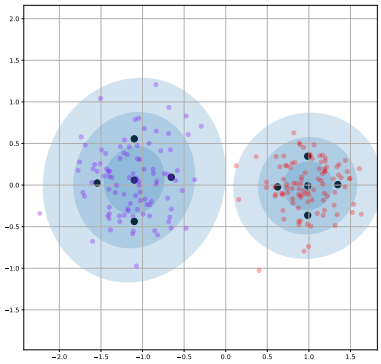
than those in the dense regions. Figure 5.6 shows the spread of sigma points inside each Gaussian component. We can use the primary scaling parameter α (which uses scaled covariance) to further control the scaling of the sigma points inside the Gaussian components.



(a) $\alpha = 0.1$

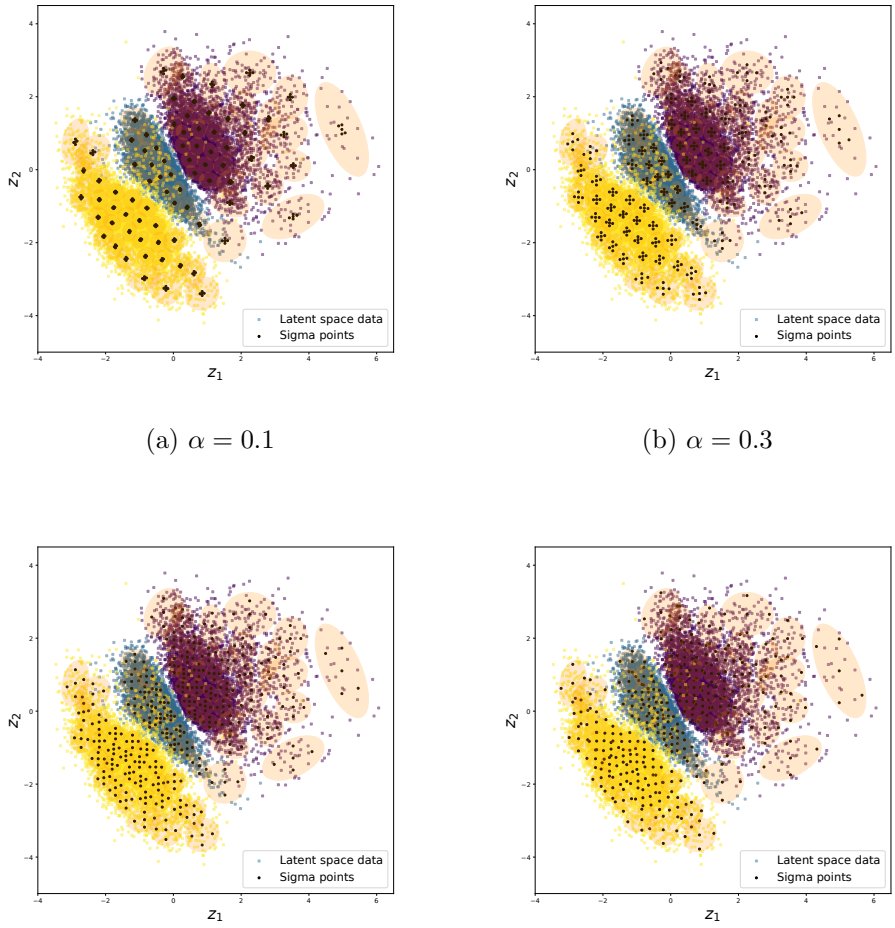


(b) $\alpha = 0.3$



(c) $\alpha = 0.7$

Figure 5.4: Effect of α on the spread of sigma points. As the value of α increases the sigma points spread more across a Gaussian distribution. The spread of the sigma points affects their representative quality.



(a) $\alpha = 0.1$

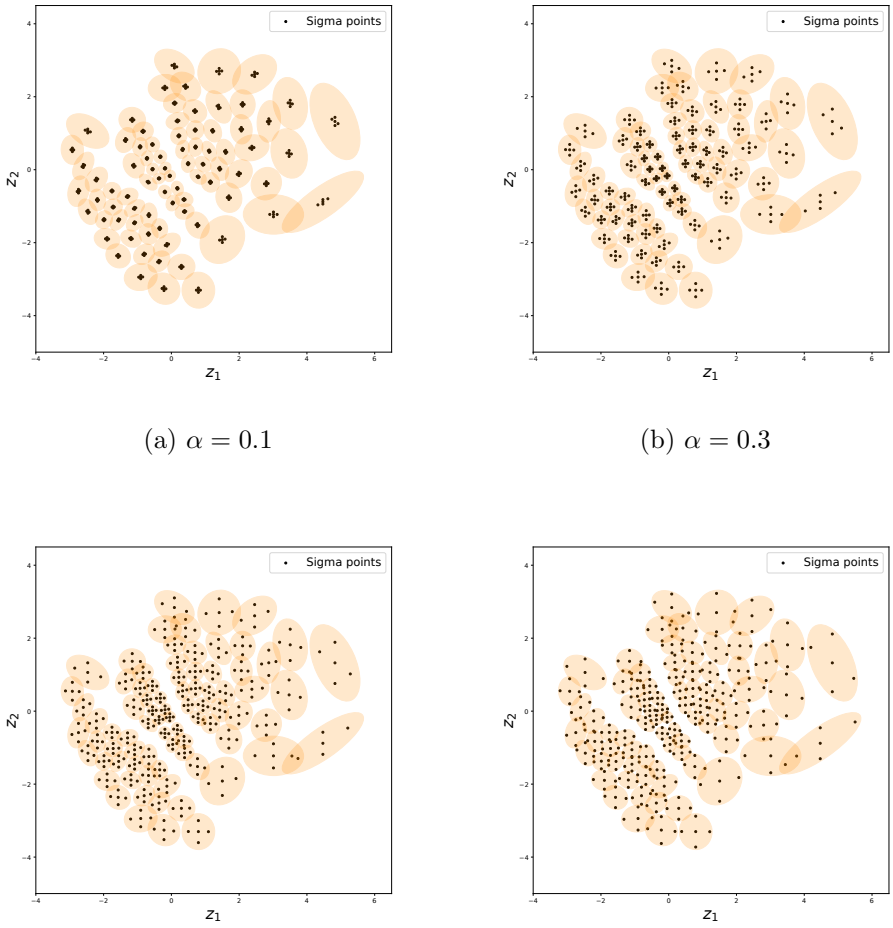
(b) $\alpha = 0.3$

(c) $\alpha = 0.5$

(d) $\alpha = 0.7$

Figure 5.5: Spread of the sigma points in the 2 dimensional latent space of MNIST data set (with digits 0,3, and 9) for different α values.

Finally, the figure 5.7 shows the collection of $80 \times (2 \times 2 + 1) = 400$ points (where $K = 80$ and $N = 2$) as a representative subset of the latent space data set with 18003 data points. Varying α generates different representative subsets.



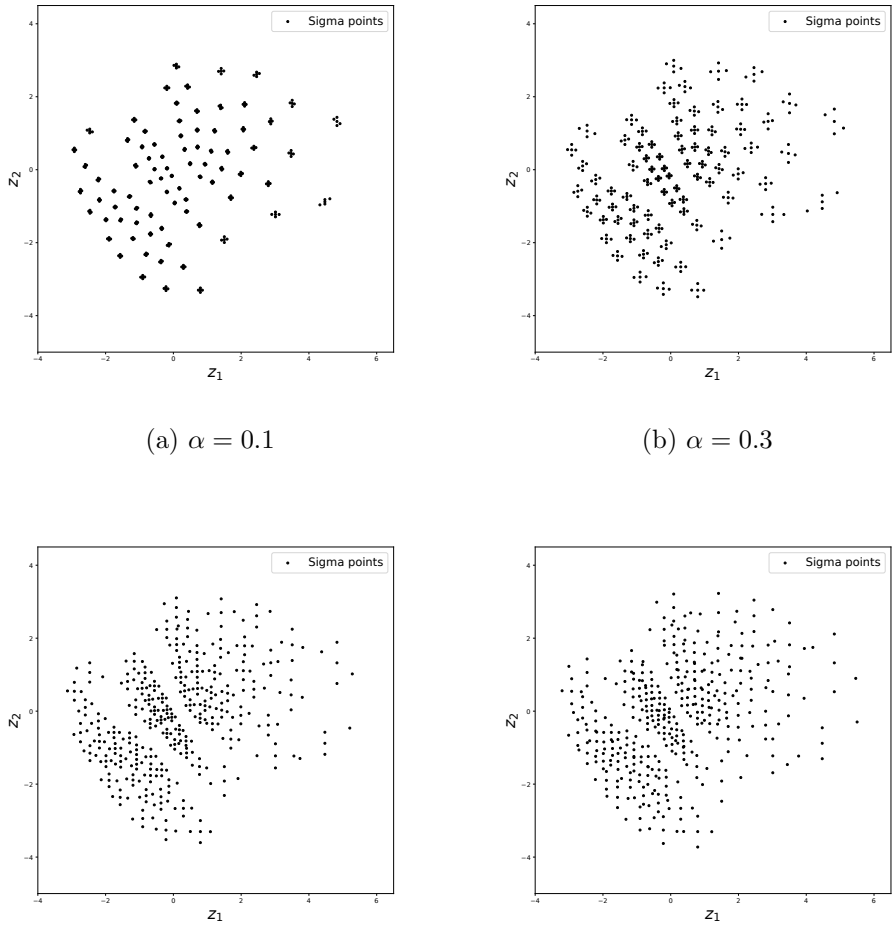
(a) $\alpha = 0.1$

(b) $\alpha = 0.3$

(c) $\alpha = 0.5$

(d) $\alpha = 0.7$

Figure 5.6: Spread of the sigma points inside each Gaussian mixture component and throughout the 2 dimensional latent space of MNIST data set (with digits 0,3, and 9) for different α values. In the regions where the latent space data is sparse the Gaussian components have higher covariance.



(a) $\alpha = 0.1$

(b) $\alpha = 0.3$

(c) $\alpha = 0.5$

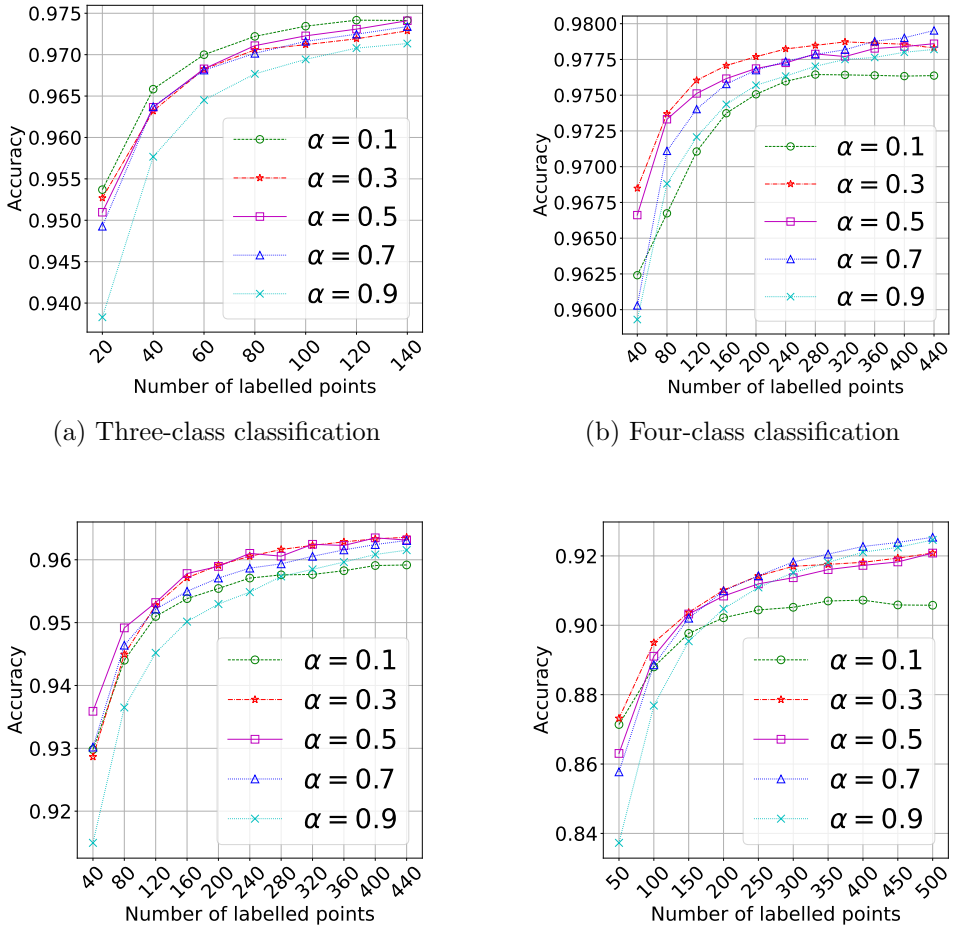
(d) $\alpha = 0.7$

Figure 5.7: Representation of the 2 dimensional latent space of MNIST data set (with digits 0,3, and 9) with the sigma points for different α values. In the regions where the latent space data is sparse, the higher covariance of the Gaussian components causes the sigma points to be more spread out than the dense regions. By changing the value of α, which is a scaled covariance parameter for sigma points, we can control the spread of the sigma points inside the Gaussian components even further to get a satisfactory representation.

Figure 5.8 shows the variation of test accuracy with varying α for three,

four, five, and ten-class classification. In these experiments, we keep the number of Gaussian components (K) at 80, 80, 80 and 100, respectively, for three, four, five, and ten-class classification.

Our experiment shows that, for a certain value of K, the best test accuracy depends on α. As expected, for a small data set, a small α gives the best test accuracy, and for a bigger data set, a high value of α spreads the sigma points enough to represent the data set best. Figure 5.8 shows that for three, four, five, and ten-class classification, the best test accuracy are achieved at α equal to 0.1, 0.3, 0.5, and 0.7, respectively.



(a) Three-class classification

(b) Four-class classification

(c) Five-class classification

(d) Ten-class classification

Figure 5.8: Effect of α on multi-class classification accuracy

This is expected because for three-class classification the unlabeled data set size is small which means a small value of α should be sufficient to generate the sigma points that are representative of the complete data set. For four and five-class classification the value of K is the same as that of the three-class but the unlabeled data set size increases with the number of classes (Table 5.1). So it is expected that higher values of α are needed as the number of classes in the experiment increases. Our experiment shows that, for a certain value of K, the best test accuracy depends on α. As expected, for a small data set, a small α gives the best test accuracy, and for a bigger data set, a high value of α spreads the sigma points enough to represent the data set best. Figure 5.8 shows that for three, four, five, and ten-class classification, the best test accuracy are achieved at α equal to 0.1, 0.3, 0.5, and 0.7, respectively. This is expected because for three-class classification the unlabeled data set size is small which means a small value of α should be sufficient to generate the sigma points that are representative of the complete data set. For four and five-class classification the value of K is the same as that of the three-class but the unlabeled data set size increases with the number of classes (Table 5.1). So it is expected that higher values of α are needed as the number of classes in the experiment increases.

It is visible in Figure 5.1b and Figure 5.1c that $\alpha = 0.1$ under-represents the data sets and hence produces the worst accuracy in both cases. It is also visible that sigma points with $\alpha = 0.7$ performs very good when the training labeled data set size is over 400 examples in four and five-class classification. This is because, for a higher number of sampled points we need versatile examples in the labeled set ($\mathcal{L}$) representative of the complete data set. Higher values of α generate more versatile sigma points because when they spread more across a Gaussian, the distance between them increases. It is safe to assume that in a latent space, with Euclidean assumption, sigma points that have higher intra-cluster distance are more versatile. For ten-class classification, $\alpha = 0.7$ performs better after 150 training samples and performs better than the other α values, which is why we declare it the best α value this task. This is expected given that the unlabeled pool in this task has 60000 examples. It is also visible that $\alpha = 0.9$ spreads the sigma points far from the most representative data in all cases.

5.2 Transfer Learning with Sigma Point Sampling on Composite Manufacturing data set

In this section, we discuss the experimental setup and results on the composite manufacturing process data set.

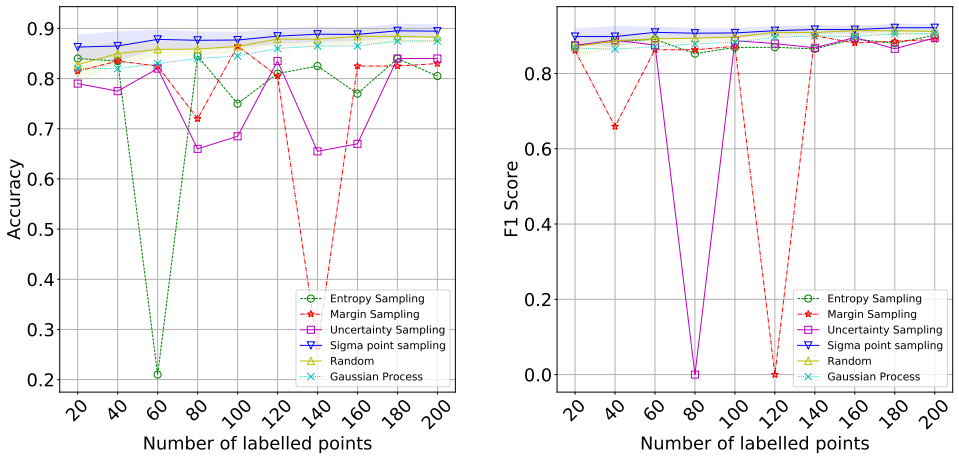
For the experiment on transfer learning with composite materials data set, we assumed the source data set (AS4/8552) has 1000 examples and we used 80% data to train the predictive modeling and 20% data was set aside for validation. In our experiment, the target data set (AS4/8551) has 1000 examples. For the target data set (AS4/8551), we set aside 200 points for testing and evaluation of our transfer learning model and apply sigma point sampling (and random sampling for comparison) on the remaining 800 examples. We vary the training set for target data from 20 examples to 200 examples. This setup is realistic for both transfer learning and active learning.

To develop the transfer learning portion of the experiment, we design the neural network (NN) architecture on the **Keras** platform of Python. Usually, the initial layers of a NNs act as feature extractors that is useful to learn the low-level features of data. On the other hand, the last few layers of the NNs are more specific to the tasks [72]. Hence, in our experiment the NN architecture has two stages namely an encoder and a classifier. The classifier from source data is less suitable for transfer hence we unfreeze the weights and retrain it on the target data. However, we use the same encoder from the source data set to extract features from the target data. The encoder has one input layer with ten channels and three hidden layers. The first two hidden layers have ten output channels, whereas the final layer of the encoder has five output channels. The reason for less number of channels in the last layer is to use the encoder as a dimensionality reduction tool as well as a feature extractor. The classifier has two hidden layers with five and ten output channels respectively while the last output layer of the classifier has one layer (for binary classification) with **sigmoid** activation. All hidden layers have Scaled Exponential Linear Unit (SELU) activation because it performed relatively better than other activation functions. We kept the learning rate fixed at 0.01 throughout the training with **Adam** optimizer. To prevent overfitting on the training data we used early stopping on 100 epochs.

In the next section, we show that the knowledge learnt from the source data set can be used in the target data set for fault detection with very few training examples.

5.2.1 Experimental results

In this section, we share results of the transfer learning study with sigma point sampling. To demonstrate the effectiveness of our sigma point sampling in an ATL scenario we compare it with various other sampling techniques namely entropy sampling, margin sampling, uncertainty sampling, random sampling and Gaussian processes [44]. Before jumping into ATL, we show in figure 5.9 that even for simple active learning (without transfer learning) on the target data set (AS4/8552) sigma point sampling (SPS) performs better than all other sampling strategies. In this figure, we start with 20 training points and we compare the test accuracy and test F1-score (on the test data set with 200 examples) for up to 200 training samples. For all the experiments, we keep a test set of 200 examples from the target data set.



(a) Accuracy comparison of SPS vs. random

(b) F1-score comparison of SPS vs. random

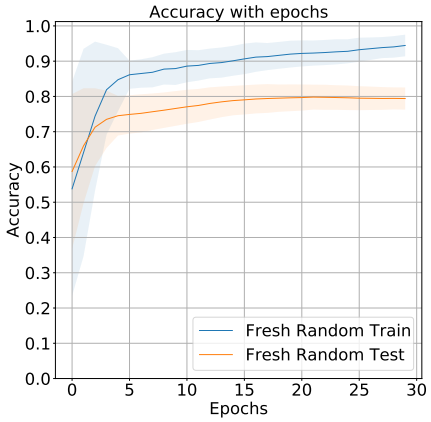
Figure 5.9: Comparison of performance metrics with increasing number of training examples in AS4/8552 for SPS vs. random sampling

For this experiment, we use the pre-trained encoder from the source data set to generate the latent space and apply the sampling techniques on the latent space to choose training examples. We train the classifier with initialized weights on the training examples. For a particular number of training points (say 20 training examples), we run 100 trials (sampling and training the classifier). Figure 5.9 shows standard deviation (SD) band for SPS and random sampling because each trial we get slightly different

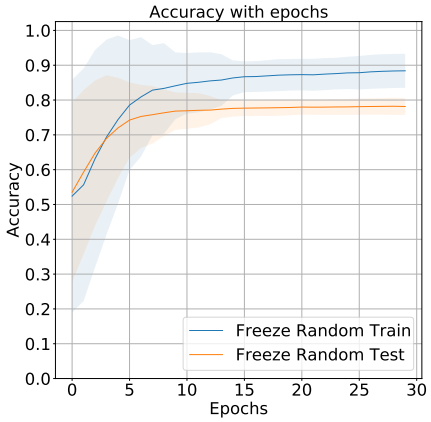
samples. For all other sampling techniques we do not have SD band because all the 100 trials in those sampling strategies led to almost the same accuracy (and F1 score).

For a fair comparison, we should also consider the upper limit of the accuracy on the target data set. We achieved a test accuracy of 92.59% when trained on the complete data set. The figure 5.9a shows that with 20 randomly selected training examples 83% test accuracy was achieved whereas training with 20 sigma points achieved a little over 86% test accuracy. The test accuracy of the other sampling strategies are very close to that of random sampling. We see the same phenomena while comparing the F1 score (Figure 5.9a). As expected, the test accuracy increases with increasing number of training examples. With 200 randomly chosen training examples we achieved a test accuracy of 88% whereas training with 200 sigma points achieved a 89.5% test accuracy. Other sampling techniques perform inferior to SPS well. The same can be said for the F1-score on the test data set (Figure 5.9a). Apart from lacking in accuracy entropy, margin and uncertainty sampling are very unreliable on this data set. We can conclude that from the sudden drops in the accuracy and f1-score in figures 5.9a and 5.9b.

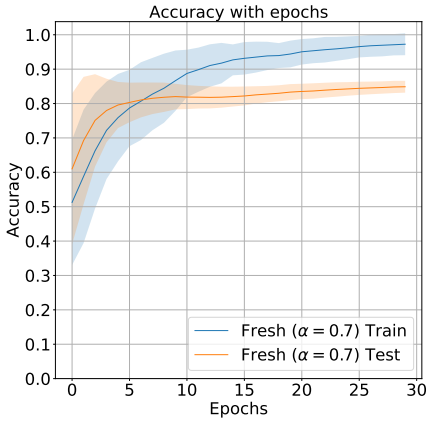
For the experiments of active transfer learning (ATL), we compare SPS with random sampling because among all other sampling strategies (except SPS) it performed the best (Figure 5.9).



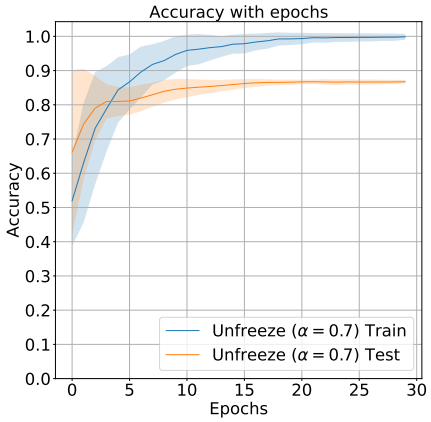
(a) Random sampling (RS)



(b) Random sampling with TL (TL)



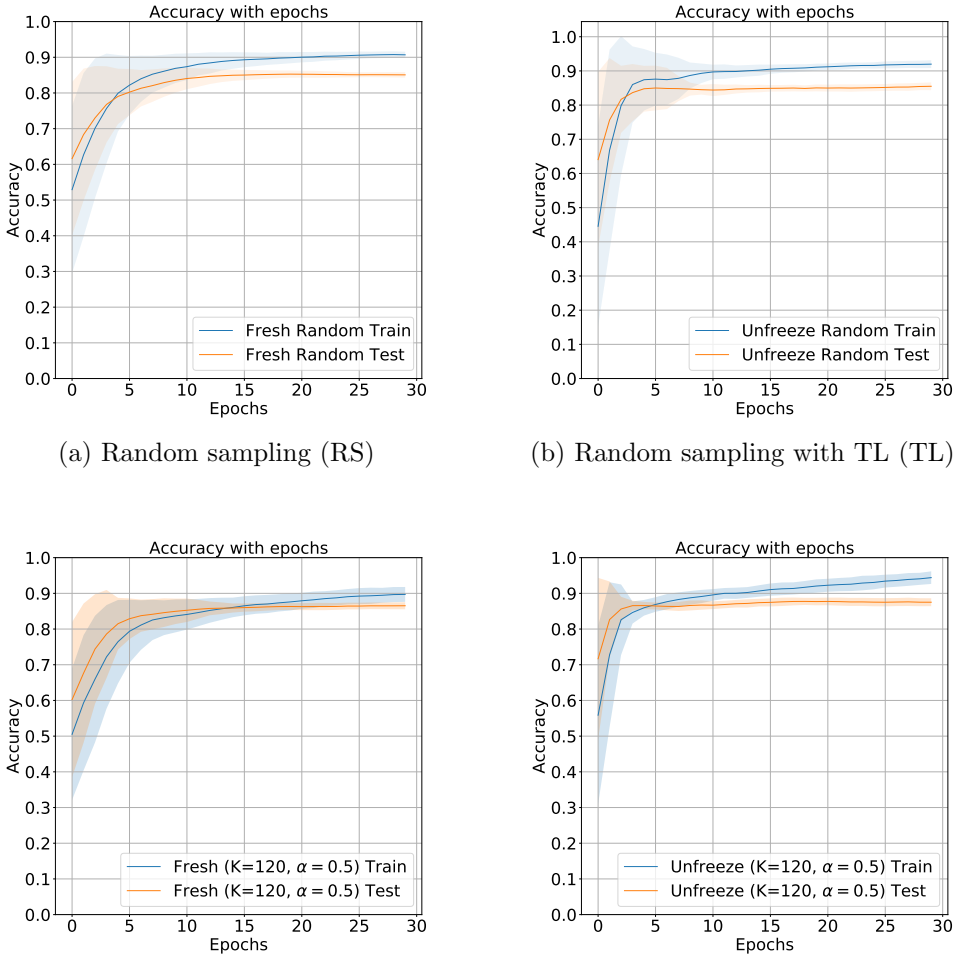
(c) SPS (AL)



(d) SPS with TL (ATL)

Figure 5.10: Comparison of accuracy in Random vs TL vs AL vs ATL on 20 points

Figures 5.10,5.11,5.12 show the train and test curves for random sampling with fresh NN (RS), random sampling with TL, sigma point sampling with fresh NN (AL), sigma point sampling with transfer learning (ATL) for up to 30 epochs. In order to make the number of passes over data for NN equal for all trials in every experiment we chose 30 epochs without early stopping. In my experiment, we found that running more than 30 epochs on 20 training samples without early stopping results in overfitting in most cases.



(a) Random sampling (RS)

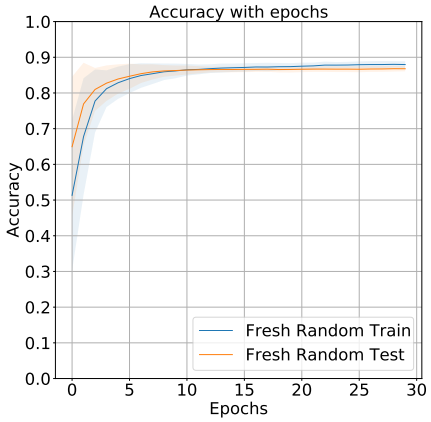
(b) Random sampling with TL (TL)

(c) SPS (AL)

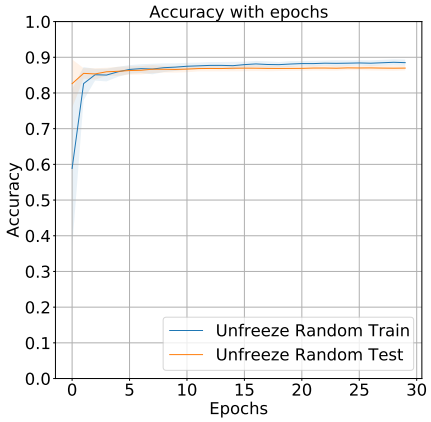
(d) SPS with TL (ATL)

Figure 5.11: Comparison of accuracy in Random vs TL vs AL vs ATL on 100 points

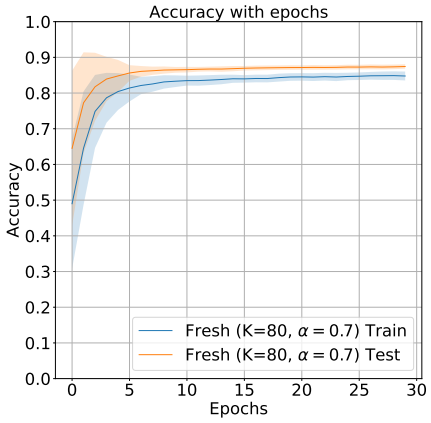
Figure 5.10 shows the train and test curves of the classifier NN with 20 training examples and 200 test examples that we set aside at the very beginning of the transfer learning study from the target data set. In figure 5.10 ATL outperforms RS, TL, AL in train and test accuracy. The confidence intervals (standard deviation fringes) of the train and test curves are the narrowest in ATL which shows that adding transferred knowledge from source data set not only increases the test accuracy of the model but also increases the confidence of performance of the model.



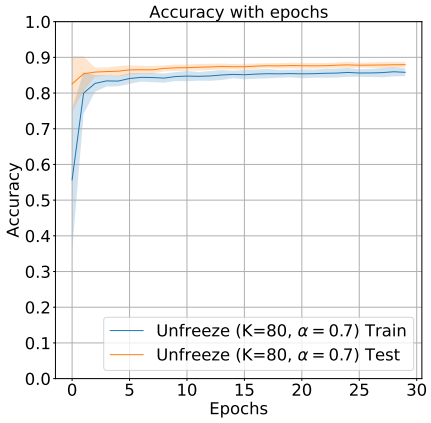
(a) Random sampling (RS)



(b) Random sampling with TL (TL)



(c) SPS (AL)



(d) SPS with TL (ATL)

Figure 5.12: Comparison of accuracy in Random vs TL vs AL vs ATL on 200 points

Figure 5.11 shows the train and test curves with 100 training examples and 200 test examples (test set with 200 examples is same for all the experiments across the transfer learning study). ATL outperforms RS, TL and AL in train and test accuracy on every epoch proving that adding transferred knowledge improves the model training. The confidence intervals (CI) in figure 5.11 are narrower when compared to the CI in figure 5.10 which shows that adding more training samples improves the confidence of the model.

Finally, figure 5.12 shows train and test curves of the classifier with 200

points. Even though we do not get a significant improvement in the final train and test accuracy, ATL slightly outperforms RS, TL and AL. ATL also has the narrowest CI of all the experiments, solidifying the claim that adding more sigma points for training with transferred weights from source data set improves test accuracy and model confidence.

5.3 Regression Learning with Sigma Point Sampling on Composite Manufacturing data set

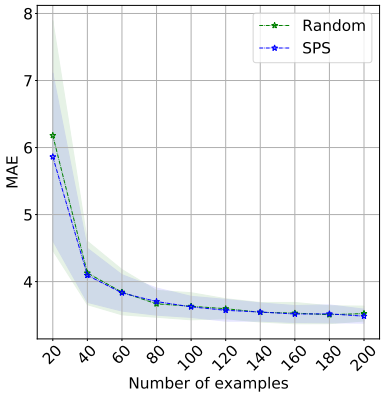
In this section, we discuss my findings on the linear regression study on composite manufacturing data set. For linear regression study, we used the AS4/8552 data set with 1000 examples. The data set has ten independent (observed) variables listed in table 4.1 and two dependent variables namely **thermal lag** and **exotherm** from the autoclave. We set aside 200 examples for testing and evaluating the performance of the linear regression model. We kept 800 examples for training and active learning. To set the upper limit performance of the linear model we used all 800 examples for training and measured the R^2 score, MAE and MSE on the test data set. MAE and MSE are good performance metrics for a linear regression model and they give a good idea about how far the residuals are from the linear regression line. Whereas R^2 score measures the amount of variance explained by the linear regression model and hence it measures the quality of fit of the linear regression model.

Table 5.2: Labels used for classification problems with their corresponding data set sizes used for training and testing.

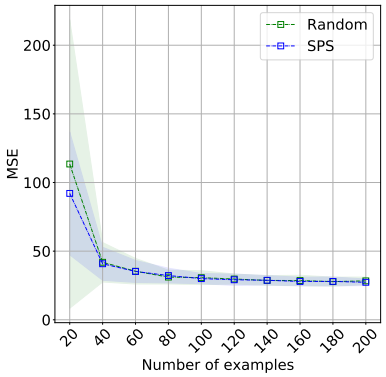
Performance metric	Value
$R^2 score$	**0.85**
MAE	**3.44**
MSE	**26.56**

For linear regression, we use **sklearn** module in Python, with 5-fold cross validation. The submodule **LinearRegression** from **sklearn** supports multiple outputs for a linear regression model which is essential for our data set. To measure the performance metrics (R^2, MAE, MSE), we used the

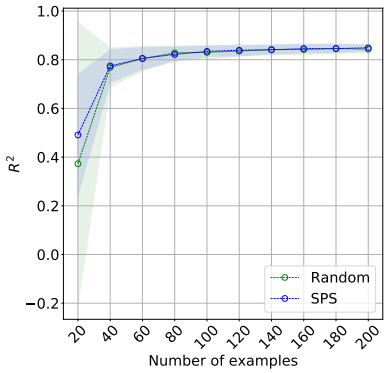
variance weighted argument for multiple outputs. Table 5.2 shows the upper limit performance metrics.



(a) Mean Absolute Error (MAE)



(b) Mean Squared Error MSE



(c) R^2 Score

Figure 5.13: Performance metrics MAE, MSE and R^2 Score on test data set for Regression with SPS

5.3.1 Experimental results

Figures 5.13 show the comparison of random sampling with sigma point sampling on the performance metrics MAE, MSE and R^2 score. Sigma point sampling performs slightly better than random sampling in all the metrics. It also has a better confidence interval especially when the training data set is very small. By using 200 examples for training, we achieved almost similar performance as the upper limit (Table 5.2).

Chapter 6

Conclusion and Future Work

In this thesis, we discussed representative subset construction and its use in active learning. We explored the theory and application of a new representative sampling approach. To our knowledge, it is the only representative sampling technique that approximates the distribution of the complete data set with a much smaller subset of points. Our novel representative subset construction approach uses sigma points (used in UKF) along with GMM to estimate the density and distribution of the data set. We use the representative subset constructed with sigma points to train the active learner compared to all other existing approaches that use real data points to do so.

In the first experiment, we saw how we can use the latent space of a data set to apply sigma point sampling. The latent space of the data set has a smaller dimension which significantly reduces the cost of implementing the soft clusters (GMM) on it compared to the observed variable space. Our experiments showed that the active learner trained with sigma points from the latent space outperforms all the representative sampling techniques used for active learning. We also showed that our approach almost always performs better than any other query strategy used for pool-based active learning.

On the second set of experiments, we showed an application of the sigma point sampling approach on the composite material manufacturing process in *Industry 4.0*. In this experiment, we proposed a new approach to active transfer learning ATL [43]. We showed that SPS helps in integrating active learning (AL) with TL, resulting in Sigma Point Sampling based Active Transfer Learning (SPSATL). We show that SPS helps in training on a new data set with transferred weights much more efficiently without compromising the accuracy. We also show that the proposed sampling strategy can be integrated with both classification and regression learning. To improve upon the existing ATL approach for optimizing the manufacturing process, we applied our *SPS with TL* (ATL) technique on the feature space of the data set. We found that SPS outperformed all other query strategies on this data set. We also performed a series of experiments to check the effect of our sampling

approach on the neural networks used for transfer learning. We found that *SPS with TL* (ATL) results in faster convergence of the neural networks. The final mean accuracy in the train and test phase is slightly higher with smaller variance in *SPS with TL* (ATL) when compared to simple random sampling (RS), random sampling with TL, and simple SPS without transferring. We replicated this experiment in three different settings by varying the size of the training set. The three experimental settings used 20, 100 and 200 points respectively for training the neural network. We found that increasing the training data reduces the variance of accuracy in both training and testing. We also found that increasing the training set brings the train and test accuracy over each epoch closer to each other, which indicates an absence of over-fitting of the model.

Labeling cost associated with data sets can be a hindrance while applying traditional machine learning methodologies in cases where labeling cost is high. Traditional PAC ML techniques are data hungry and active learning solves that problem by smartly querying the useful data. In this thesis, we saw an application of active learning where labeling the data might be costly because it needs to through a proprietary software. Our batch-mode active learning pipeline includes a novel query strategy (sigma point representative subset) which captures the distribution of the complete data set with very few data points. Our active learning pipeline works with dimensionality reduction tools such as VAE and principal component analysis (PCA).

One limitation of this research is that our pipeline requires us to apply GMM on the latent space first and then use sigma point sampling for representative subset construction. We will create an end-to-end active learning on latent space pipeline with sigma points in the future. Our pipeline assumes that the latent space follows the Euclidean geometry which is not always correct. Even though ideally we should ask the labels of the sigma points to an oracle we use *k-nearest neighbor* for querying the label of the nearest data point because it is readily available to us. While querying the labels of the sigma points, the nearest neighbor might change depending on the geometry of the latent space. In future, we will also take into consideration the Reimann geometry of the latent space [2] while querying the labels of the sigma points.

Bibliography

[1] Pankaj K Agarwal, Sariel Har-Peled, Kasturi R Varadarajan, et al. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.

[2] Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*, 2017.

[3] Olivier Bachem, INF ETHZ, Mario Lucic, and ETHZ CH. Scalable and distributed clustering via lightweight coresets. *stat*, 1050:27, 2017.

[4] Olivier Bachem, Mario Lucic, S Hamed Hassani, and Andreas Krause. Approximate k-means++ in sublinear time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[5] Olivier Bachem, Mario Lucic, and Andreas Krause. Coresets for nonparametric estimation-the case of dp-means. In *International Conference on Machine Learning*, pages 209–217. PMLR, 2015.

[6] Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017.

[7] Olivier Bachem, Mario Lucic, and Andreas Krause. Scalable k-means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1119–1127, 2018.

[8] Sima Behpour, Anqi Liu, and Brian Ziebart. Active learning for probabilistic structured prediction of cuts and matchings. In *International Conference on Machine Learning*, pages 563–572. PMLR, 2019.

[9] Philippe Bracke, Anupam Datta, Carsten Jung, and Shayak Sen. Machine learning explainability in finance: an application to default risk analysis. 2019.

[10] Corinna Cortes, Giulia DeSalvo, Mehryar Mohri, Ningshan Zhang, and Claudio Gentile. Active learning with disagreement graphs. In *International Conference on Machine Learning*, pages 1379–1387. PMLR, 2019.

[11] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751, 2005.

[12] Ido Dagan and Sean P Engelson. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, pages 150–157. Elsevier, 1995.

[13] Bin Dai, Yu Wang, John Aston, Gang Hua, and David Wipf. Connections with robust pca and the role of emergent sparsity in variational autoencoder models. *The Journal of Machine Learning Research*, 19(1):1573–1614, 2018.

[14] JFG De Freitas, Mahesan Niranjan, AH Gee, and Arnaud Doucet. Sequential monte carlo methods for optimisation of neural network models. *Cambridge University Engineering Department, Cambridge, England, Technical Report TR-328*, 1998.

[15] Matthew F Dixon, Igor Halperin, and Paul Bilokon. *Machine Learning in Finance*. Springer, 2020.

[16] Dan Feldman, Matthew Faulkner, and Andreas Krause. Scalable training of mixture models via coresets. In *NIPS*, pages 2142–2150, 2011.

[17] Dan Feldman, Zahi Kfir, and Xuan Wu. Coresets for gaussian mixture models of any shape. *arXiv preprint arXiv:1906.04895*, 2019.

[18] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering. *SIAM Journal on Computing*, 49(3):601–657, 2020.

[19] Lirong Gan, Huamao Wang, and Zhaojun Yang. Machine learning solutions to challenges in finance: An application to the pricing of financial products. *Technological Forecasting and Social Change*, 153:119928, 2020.

[20] Zahra Ghafoori, James C Bezdek, Christopher Leckie, and Shanika Karunasekera. Unsupervised and active learning using maximin-based anomaly detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 90–106. Springer, 2019.

[21] David Haussler. *Probably approximately correct learning*. University of California, Santa Cruz, Computer Research Laboratory, 1990.

[22] Junyan Hu, Hanlin Niu, Joaquin Carrasco, Barry Lennox, and Farshad Arvin. Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12):14413–14423, 2020.

[23] Simon J Julier. The scaled unscented transformation. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, volume 6, pages 4555–4559. IEEE, 2002.

[24] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–193. International Society for Optics and Photonics, 1997.

[25] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[26] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

[27] Mike Klaas, Nando De Freitas, and Arnaud Doucet. Toward practical n2 monte carlo: the marginal particle filter. *arXiv preprint arXiv:1207.1396*, 2012.

[28] Andrew Kusiak. Smart manufacturing. *International Journal of Production Research*, 56(1-2):508–517, 2018.

[29] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[30] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier, 1994.

[31] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

[32] Wei Li, Yuanbo Chai, Fazlullah Khan, Syed Rooh Ullah Jan, Sahil Verma, Varun G Menon, Xingwang Li, et al. A comprehensive survey on machine learning-based big data analytics for iot-enabled smart healthcare system. *Mobile Networks and Applications*, pages 1–19, 2021.

[33] Zhencai Li, Yang Wang, and Zhen Liu. Unscented kalman filter-trained neural networks for slip model prediction. *PloS one*, 11(7):e0158492, 2016.

[34] Mario Lucic, Olivier Bachem, and Andreas Krause. Strong coresets for hard and soft bregman clustering with applications to exponential family mixtures. In *Artificial intelligence and statistics*, pages 1–9. PMLR, 2016.

[35] Mario Lucic, Matthew Faulkner, Andreas Krause, and Dan Feldman. Training gaussian mixture models at scale via coresets. *The Journal of Machine Learning Research*, 18(1):5885–5909, 2017.

[36] Andrew Kachites McCallumzy and Kamal Nigamy. Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer, 1998.

[37] Ziad Obermeyer and Ezekiel J Emanuel. Predicting the future—big data, machine learning, and clinical medicine. *The New England journal of medicine*, 375(13):1216, 2016.

[38] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[39] Duc T Pham and Ashraf A Afify. Machine-learning techniques and their applications in manufacturing. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 219(5):395–412, 2005.

[40] Rafael Coimbra Pinto and Paulo Martins Engel. A fast incremental gaussian mixture model. *PloS one*, 10(10):e0139931, 2015.

[41] Farhad Pourkamali-Anaraki and Michael B Wakin. The effectiveness of variational autoencoders for active learning. *arXiv preprint arXiv:1911.07716*, 2019.

[42] Foster Provost and Tom Fawcett. Data science and its relationship to big data and data-driven decision making. *Big data*, 1(1):51–59, 2013.

[43] Milad Ramezankhani, Bryn Crawford, Apurva Narayan, Heinz Voggenreiter, Rudolf Seethaler, and Abbas S Milani. Making costly manufacturing smart with transfer learning under limited data: A case study

on composites autoclave processing. *Journal of Manufacturing Systems*, 59:345–354, 2021.

[44] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[45] RAVEN. simulation software. `http://www.convergent.ca/products/raven-simulation-software`.

[46] Florence Regol, Soumyasundar Pal, Yingxue Zhang, and Mark Coates. Active learning on attributed graphs via graph cognizant logistic regression and preemptive query generation. In *International Conference on Machine Learning*, pages 8041–8050. PMLR, 2020.

[47] Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741, 2009.

[48] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer, 2001.

[49] Aditya Kumar Sehgal, Sanmay Das, Keith Noto, Milton Saier, and Charles Elkan. Identifying relevant data for a biological database: Handcrafted rules versus machine learning. *IEEE/ACM transactions on computational biology and bioinformatics*, 8(3):851–857, 2010.

[50] Burr Settles. *Curious machines: Active learning with structured instances*. PhD thesis, University of Wisconsin–Madison, 2008.

[51] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[52] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, 2008.

[53] Burr Settles, Mark Craven, and Lewis Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, volume 1. Vancouver, CA:, 2008.

[54] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[55] M Sughasiny and J Rajeshwari. Application of machine learning techniques, big data analytics in health care sector–a literature survey. In *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2018 2nd International Conference on*, pages 741–749. IEEE, 2018.

[56] Chuang Sun, Meng Ma, Zhibin Zhao, Shaohua Tian, Ruqiang Yan, and Xuefeng Chen. Deep transfer learning based on sparse autoencoder for remaining useful life prediction of tool in manufacturing. *IEEE Transactions on Industrial Informatics*, 15(4):2416–2425, 2018.

[57] Iiris Sundin, Peter Schulam, Eero Siivola, Aki Vehtari, Suchi Saria, and Samuel Kaski. Active learning for decision-making from imbalanced observational data. In *International Conference on Machine Learning*, pages 6046–6055. PMLR, 2019.

[58] Fei Tao, Qinglin Qi, Ang Liu, and Andrew Kusiak. Data-driven smart manufacturing. *Journal of Manufacturing Systems*, 48:157–169, 2018.

[59] R Andrew Taylor, Joseph R Pare, Arjun K Venkatesh, Hani Mowafi, Edward R Melnick, William Fleischman, and M Kennedy Hall. Prediction of in-hospital mortality in emergency department patients with sepsis: a local big data–driven, machine learning approach. *Academic emergency medicine*, 23(3):269–278, 2016.

[60] Ryan Turner and Carl Edward Rasmussen. Model based learning of sigma points in unscented kalman filtering. *Neurocomputing*, 80:47–53, 2012.

[61] Inam Ullah, Yu Shen, Xin Su, Christian Esposito, and Chang Choi. A localization based on unscented kalman filter and particle filter localization algorithms. *IEEE Access*, 8:2233–2246, 2019.

[62] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[63] Rudolph Van Der Merwe et al. *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. PhD thesis, OGI School of Science & Engineering at OHSU, 2004.

[64] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

[65] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[66] Yuanyuan Wang, Shuhong Chai, Faisal Khan, and Hung Duc Nguyen. Unscented kalman filter trained neural networks based rudder roll stabilization system for ship in waves. *Applied Ocean Research*, 68:26–38, 2017.

[67] Jeremy West, Dan Ventura, and Sean Warnick. Spring research presentation: A theoretical foundation for inductive transfer. *Brigham Young University, College of Physical and Mathematical Sciences*, 1(08), 2007.

[68] Dongrui Wu, Vernon J Lawhern, Stephen Gordon, Brent J Lance, and Chin-Teng Lin. Offline eeg-based driver drowsiness estimation using enhanced batch-mode active learning (ebmal) for regression. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 000730–000736. IEEE, 2016.

[69] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1):23–45, 2016.

[70] Yan Xu, Yanming Sun, Xiaolong Liu, and Yonghua Zheng. A digital-twin-assisted fault diagnosis using deep transfer learning. *IEEE Access*, 7:19990–19999, 2019.

[71] Bin Yang, Yaguo Lei, Feng Jia, and Saibo Xing. An intelligent fault diagnosis approach based on transfer learning from laboratory bearings to locomotive bearings. *Mechanical Systems and Signal Processing*, 122:692–706, 2019.

[72] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014.

[73] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.