UNIVERSITY OF CALIFORNIA, MERCED

Expectation Maximization Algorithm for Optimization of Piecewise-constant Models and Their Applications

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering and Computer Science

by

Pooya Tavallali

Committee in charge:

Professor Mukesh Singhal, Chair
Professor YangQuan Chen
Professor Roummel Marcia

2021

The dissertation of Pooya Tavallali is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

(Professor YangQuan Chen)

(Professor Roummel Marcia)

(Professor Mukesh Singhal, Chair)

University of California, Merced

2021

DEDICATION

To my mom, dad and brothers.

EPIGRAPH

Life is never easy. There is work to be done and obligations to be met –
obligations to truth, to justice, and to liberty.
—John F. Kennedy

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thanks to Dr. Mukesh Singhal for his insight and valuable advice without whom this tough journey would have not been possible.

I also would like to express my gratitude and appreciation to to Dr. YangQuan Chen and Dr. Roummel Marcia for their guidance and advice as committee members of this dissertation.

I also would like to thank Dr. Vahid Behzadan of New Haven University and Dr. Peyman Tavallali of California Institute of Technology who were our collaborators and good technical partners in the projects of this dissertation. I had several stimulating discussions with them.

VITA

2013	B. S. in Electrical Engineering, Shiraz University, Shiraz, Iran
2013-2016	M. Sc. in Electrical Engineering, communication systems, Shiraz University, Shiraz, Iran
2021	Ph. D. in Electrical Engineering and Computer Science, University of California, Merced

PUBLICATIONS

2021 "An EM-Based Optimization of Synthetic Reduced Nearest Neighbor Model towards Multiple Modalities Representation with Human Interpretability", Pooya Tavallali, Peyman Tavallali, Mohammad Reza Khosravi and Mukesh Singhal; Accepted in Multimedia Tools and Applications

2021 "Discrete Linear-Complexity Reinforcement Learning in Continuous Action Spaces For Q-Learning Algorithms" Peyman Tavallali, Pooya Tavallali, Gary B. Doran Jr., Lukas Mandrake, Mukesh Singhal, Under Submission

2021 "Adversarial Poisoning Attacks and Defense for General Multi-Class Models Based On Synthetic Reduced Nearest Neighbors" Pooya Tavallali, Vahid Behzadan, Peyman Tavallali, Mukesh Singhal, arXiv:2102.05867

2021 "K-means tree: an optimal clustering tree for unsupervised learning" Pooya Tavallali, Peyman Tavallali, Mukesh Singhal, The Journal of Supercomputing

2020 "Interpretable Synthetic Reduced Nearest Neighbor: an Expectation Maximization Approach", Pooya Tavallali, Peyman Tavallali, Mohammad Reza Khosravi and Mukesh Singhal; International Conference on Image Processing (ICIP)

2020 "A multi-pollutant model: a method suitable for studying complex relationships in environmental epidemiology", Pooya Tavallali, Hamed Gharibi, Mukesh Singhal and Ricardo Cisneros; Air Quality, Atmosphere and Health

2020 "A Systematic Training Procedure for Viola-Jones Face Detector in Heterogeneous Computing Architecture", Pooya Tavallali, Mehran Yazdi, and Mohammad Reza Khosravi; Journal of Grid Computing, pages 1-16

2019 "Robust Cascaded Skin Detector Based on Adaboost", Pooya Tavallali, Mehran Yazdi, and Mohammad Reza Khosravi; Multimedia Tools and Applications, pages 122

ABSTRACT OF THE DISSERTATION

Expectation Maximization Algorithm for Optimization of Piecewise-constant Models and Their Applications

by

Pooya Tavallali

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California Merced, 2021

Professor Mukesh Singhal, Chair

The Expectation-Maximization (EM) Algorithm is well-known in the literature of machine learning and has been widely used for training of probabilistic and some non-probabilistic models, such as mixture of Gaussians and K-means, respectively. Despite the vast volume of research on application of the EM algorithm for training probabilistic models, there has been little attempt toward usage of the EM algorithm for non-probabilistic models. In this dissertation, various piecewise constant models, and their learning procedures in the literature are reviewed. For each model, the EM-based optimization of reviewed model is proposed. The EM algorithms proposed in this dissertation have the same spirit as the original EM algorithm. For each model, the proposed EM algorithm is properly modified to fit the non-probabilistic nature of the model. The EM algorithm was originally designed to fit the modular structure of any intelligent model, such as neural networks or mixture models. In this dissertation, it is shown how with the EM algorithm it is possible to approach a piecewise constant model as a modular structure and optimize the model based on each module of the structure. The optimization procedure consists of two steps, Expectation/assignment step and Maximization/update step. More specifically, in the EM algorithm, for each module of the structure, a maximization/minimization problem has to be solved. The parameters of optimization problem for each module are provided by the expectation step for that module.

In this dissertation, it is shown that such optimization problems are NP-hard and can often be approximated through a proper surrogate objective function. We proposed novel surrogate functions. The proposed EM-based approach is applied to several piecewise constant models, such as prototype nearest neighbor. Further, the convergence guarantee and computational complexity of the developed EM algorithms are presented for each model. Finally, through extensive experiments we show that the proposed EM-based algorithms have superior or similar performance when compared with several other similar state-of-the-art models and algorithms. Additionally, the proposed approach for optimizing the piecewise constant models provides an in-depth interpretability for training procedures. We specifically applied the proposed optimization algorithm to synthetic reduced nearest neighbor for classification, adversarial label-poisoning, robust synthetic reduced nearest neighbor and synthetic reduced nearest neighbor for regression.

Chapter 1

Introduction

1.1 Problem Statement

Piecewise constant models are a well-known part of the machine learning literature. Here we formally define piecewise constant model as any model that partitions the input space into various disjoint sections and assigns a constant predictor to each section. This definition is similar to definition of piecewise constant function. A piecewise constant model usually represents each partition with a single module or a hierarchy of decision functions such as K-means, prototype nearest neighbor and decision trees [55]. The general optimization problem of a piecewise constant model is represented as [55]:

$$\min_{F} \quad Loss_{\alpha}(F) = \sum_{j} Loss(R_j) + \alpha Cost(R_j) \tag{1.1}$$

Where F represents the model's function, and $Loss_{\alpha}$ represents the cost function. j is the index number of j^{th} region R_j. $Loss$, α, and $Cost(.)$ are the loss function, complexity cost coefficient and complexity cost, respectively. A 2-d visualization of problem (1.1) is presented in figure 1.1 for example of synthetic reduced nearest neighbor. Each region is presented with a centroid. The samples of each class is presented with the color of the class. The problem consists of modifying the regions by changing the centroids such that problem (1.1) is minimized.

The optimization problem presented in (1.1) can be NP-hard [81] or NP-complete [74] depending on the model. As a result, most training procedures

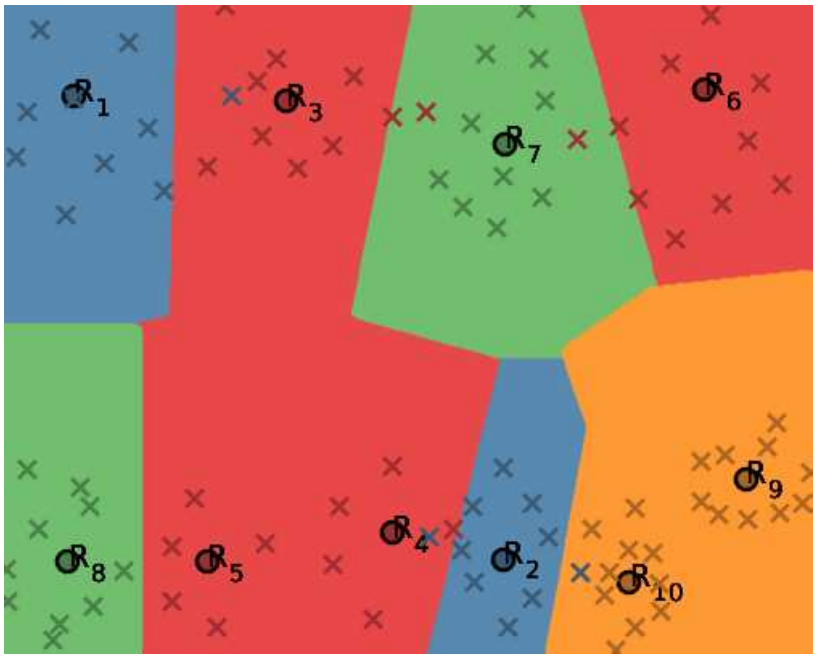


Figure 1.1: 2-D visualization of Synthetic Reduced Nearest Neighbor.

in the literature tend to construct the model either using some greedy approach [21, 96, 81] or by replacing the objective function in (1.1) with a smooth surrogate loss function and 'softening' the rigidness of regions [11, 45]. However, these approaches are either sub-optimal or lack convergence guarantee to a local optimum of problem in (1.1).

A third approach is application of EM algorithm to such models and exploit the modular structure of the model. A well-known example of such approach is the algorithm used for K-means [81]. In K-means, the algorithm consist of assignment step (expectation step) where samples assigned to each centroid is found and minimization step (maximization step) where each centroid is optimized based on the centroid's samples. Although that the third approach is very simple and efficient, it has barely been used for piecewise constant models. In this dissertation, we will expand this approach to several piecewise constant models. An important

aspect of the proposed EM-based algorithms is the guarantee to achieve some local optimum of the original problem. Additionally, it is easy to show that almost all such EM algorithms are efficient algorithms and their computational complexity is linear over trainset.

Expectation Maximization (EM) algorithm is among the oldest meta algorithms of the literature [63] and has widely been used for optimization of various probabilistic models [17, 72] and signal processing tasks [85]. The EM algorithm is a very general method designed to solve maximum likelihood estimation. In an EM algorithm, the aim is to maximize a maximum likelihood function $L(\theta) \equiv P(y|\theta)$ with respect to the parameters of the model θ. It is often the case that maximizing $P(y|\theta)$ is not easy. Therefore, the idea in EM algorithm consist of introducing a latent variable Z whose pdf depends on θ ($P(Z|\theta)$). An observation with EM algorithm is that the optimization of $L(\theta)$ would be easier if only a set of additional variables Z were known. As a result, if Z were known then all that is needed to maximize $P(Z|\theta)$. For example, in case of a K-means algorithm, if the assignment of the samples were known then the optimal place of the centroids could be found by only finding the mean of samples for each cluster. In case of a decision tree, if the rout for each sample was known then the parameters of samples could be calculated easily by only solving the node problems [62]. However, in realty the Z variables are not known and are approximated through an expectation step and then the parameters are updated based on the estimated Z variables. Finally, the algorithm iterates over the expectation step (approximation of Z) and maximization step until no further improvement is possible over the loss of the structure. It is worth mentioning that Z is usually defined such that the optimization procedure fits the modular structure of the model [62, 81, 72]).

1.2 Motivation

As per our extensive search in the literature, most piecewise constant models are optimized indirectly or through some greedy algorithm. The indirect approach consists of smoothening the structure of the model and optimizing a differentiable

surrogate objective function instead of the original objective function. However, such approach lacks any guarantee for the original loss function. The greedy approach consists of greedily setting the modules of the model (e.g., K-means++ [4]) through some criterion. Such approaches might have some guarantee over the global optimum of the problem (1.1) depending on the task. However, they are sub-optimal and can be improved further. As a result, a proper approach that have properties of achieving an optimum in efficient time is needed. In this dissertation we tackle the mentioned issue by introducing EM algorithm for piecewise constant models. This approach provides both guarantee of achieving a local optimum and its time complexity is efficient. For each problem other properties are also extracted.

1.3 Proposed EM-based approach

In this dissertation, the EM algorithm is properly modified to directly optimize the loss function of several piecewise constant models. The modifications depend on the structure of the model and its loss function. The general algorithm consists of applying the following two steps to each module of the structure:

- introducing and approximating latent variables at the module (expectation step)
- optimizing the parameters of the module (maximization step)

The algorithm iterates over the mentioned steps for all the modules until convergence. The latent variables are loss of assigning a sample to a property in the module. The latent variables essentially provide a weighted target label for each sample at every module. The problem at each module is defined based on the latent variables at the module. It is often the case that finding the global optimum of the maximization problem is a NP-hard problem. Therefore, we approximate solution for the maximization problem through either a known surrogate objective function or we propose a novel surrogate objective function. The general pseudo code of the proposed algorithm is presented in figure 1.2.

iterate until convergence:

for each module j:

1-**apply expectation**:

Approximate latent variables

-At the module, calculate the loss of assigning each of the training samples to the module or a property of the module using the current parameters of the model

2-**apply maximization step**:

Approximate the maximization problem

-Approximate the module parameters through a surrogate objective function

Figure 1.2: The general pseudo of EM algorithm for a piecewise constant model.

1.4 Summary of Results

The next chapters consist of the applying the proposed algorithm to various synthetic reduced nearest neighbor. Each chapter consists of four parts. The first part is the introduction and related works of the chapter and the second part is the proposed EM algorithm for the chapter's corresponding model. The third part consists of the experiments. The fourth part is the summary of the chapter.

In the following chapters, the present applications of proposed EM algorithm for piecewise constant models are presented.

In the second chapter, the optimization of synthetic reduced nearest neighbor (SRNN also known as prototype nearest neighbor) based on EM algorithm is presented. The properties of SRNN are explored and it is shown that the EM algorithm proposed at the second chapter is the first that minimizes the original loss function of the model.

In the third chapter, the proposed algorithm is used in the context of robustness and adversarial attacks. The first multi-class adversarial label-poisoning attack is proposed. It is shown that the attack is computationally efficient. Further, a robust defense technique based on EM algorithm is proposed and presented in the

third chapter.

In the fourth chapter, the optimization of regression synthetic reduced nearest neighbor is studied and its properties such as consistency are explored. To the best of our knowledge, the proposed model in the fourth chapter is the first of its kind in the literature.

In each chapter the related work of the problem and its application are presented.

Chapter 2

Interpretable Synthetic Reduced Nearest Neighbor: An Expectation Maximization Approach (EM-SRNN)

A convenient, accurate and well-known way toward any supervised task is using Nearest Neighbor approach or its variants. However, there has been little attempt toward improving interpretability by human and providing a classical optimization of Synthetic Reduced Nearest Neighbor. To tackle these issues, this chapter provides a novel optimization of Synthetic Reduced Nearest Neighbor based on Expectation Maximization (EM-SRNN). Reduced Nearest Neighbor model consists of a subset of the samples from the trainset that has similar accuracy to Nearest Neighbor. Synthetic Reduced Nearest Neighbor relaxes the model to learn K synthetic samples (or prototypes/centroids) in the space of dataset. Therefore, inspired by EM algorithm for K-means, we propose a novel optimization based on EM algorithm to learn EM-SRNN by iterating over the centroids of the model and assignment of training samples to the centroids. The first step consists of optimizing the position of each centroid based on the assignment of the samples to the centroid and the second step consists finding optimal assignments and labels

of the centroids. The EM-SRNN is interpretable since the centroids exist inside the space of training samples. Additionally, the centroids represent the multiple modalities (or sub-clusters) of the classes that are interpretable by human. These properties make this type of interpretability unique, hence, making this model suitable for various studies that are related to interpretability by human, such as image processing and epidemiological studies. In this chapter, analytical aspects of problem are explored and it is shown that computational complexity of proposed optimization is linear over size of the trainset. Finally, EM-SRNN shows superior or similar performance when compared with several other interpretable and similar state-of-the-art models, such as trees and kernel SVMs.

2.1 Introduction and Related Works

One of the fundamental questions in Machine Learning (and various statistical studies such as epidemiological data analysis) is: does there exist any interpretable model that shows the relation of features to each sub-cluster of each class of data? Motivated by this question, this chapter proposes a novel EM-based optimization for SRNN (EM-SRNN) that monotonically decreases the 0-1 loss function until convergence. The optimization is inspired by K-means algorithm [81].

The proposed model, EM-SRNN, is designed such that it can learn multiple modalities of the input data. The learned modalities are represented by centroids that are interpretable by human. The SRNN model is similar to other machine learning models (ensemble and decision tree models [77, 7, 12, 117, 116]) in the sense that it partitions the space to several regions. However, EM-SRNN is capable of learning an accurate model that can properly understand underlying relation between modalities of the data for each region and true classes; hence, making EM-SRNN unique for human interpretability of an accurate multi-classification model.

Nearest Neighbor and Reduced Nearest Neighbor are among the oldest models of Machine Learning [28, 47]. Two of the reasons for popularity of Nearest Neighbor are its simplicity and competitiveness in many machine learning tasks [119] when

combined with the distance metrics [50] [31, 128] or domain knowledge [9, 110]. Recently, machine learning algorithms have become dominant in applications such as recommender systems [105]. Among different machine learning models, kNN has been more attractive because of its interpretable predictions.

However, the main drawback of nearest neighbor methods is its inefficiency at inference time. It is mainly because every input sample has to be compared with all samples from the trainset. Inference particularly takes $\mathcal{O}(ND)$ operations for an input where N and D are number of samples and input dimensionality, respectively. Similarly, the memory complexity of the nearest neighbor is $\mathcal{O}(ND)$. This makes kNN methods very expensive in terms of memory and inference time, making them impractical for time critical applications and large-scale datasets. Design of kNN methods which overcome these deficiencies is currently one of the hot research topics in the area of nearest neighbor models.

One general approach in the literature to reduce inference complexity of Nearest Neighbor is through some data structure such as tree or cover/ball trees [32, 13, 83] or hashing functions [49, 2] which yield high speed ups. However, they are neither interpretable nor storage efficient because they need to store the whole trainset and they also suffer from curse of dimensionality.

The tree methods consist of partitioning the input space into non-overlapping subsets. It is done by traversing through the tree structure. The root node consists of the whole space and each leaf node corresponds to one of the subsets. Split nodes cut the space and samples into two disjoint subsets (left and right children of the split node). Once a sample is entered, the sample is routed through the tree from root node to a leaf node. Trees are different in how they perform split in the space. In [121, 92] samples are partitioned based on their projections over the line between two pivot points. In KD-trees, axis-aligned partitions are used. In spill trees [80, 30] children of a node can spill over into one another and share common data points.

To reduce storage consumption, several studies have aimed at reducing the dimensionality [127]. These methods often use tree structures to improve speed of the model. [26] shrink the trainset to a few cluster centers that are optimized

through multi-phase initialization procedures [34].

Another approach is to reduce the trainset [28, 47, 3] which subsamples the trainset by removing redundant data or through some heuristic methods. Several other studies consist of shrinking the dataset to a few cluster centers [26, 71]. These methods can be optimized with multi-phase initialization procedures [34, 79].

Another approach is based on learning prototypes by 'softening' the Nearest Neighbor rule at inference [11, 45]. The authors in [73] propose a stochastic approach. The work in [54] provides a smoothened objective function as a surrogate for optimizing KNN prediction function; hence, it does not use tree data structures. However, none of these methods has a guarantee of both convergence and monotonically decreasing of the original loss function.

The method in [73] learns synthetic centroids such that the likelihood of a specific class probability model is maximized. [73] uses a surrogate soft objective function, instead of original 0-1 loss function.

Authors in [126] extend the work of [73] to learning a low-dimensional transformation of input dataset, jointly with learning centroids/prototypes. Its main drawback is that it uses a feedforward network to reduce the dimensionality of samples; hence, the model size is larger than that of [73].

In [133], authors propose a binary embedding technique that jointly learns a binary embedding and a set of binary synthetic centroids. However, it does not significantly compresses the size of the model. Additionally, the optimization in [133] is difficult because of the discrete nature of its problem.

As per our extensive search, no EM algorithm is ever applied to learn a SRNN model. EM algorithm and similar iterative approaches have previously been used to optimize several useful models in the literature [81, 62, 35, 23, 85]. These methods are basically based on first introducing a latent variable for assignment of samples to a distribution that generates the data and then iterating over centroids/gating networks/tree nodes/distributions/leaf nodes in the model and updating the latent variables.

This chapter provides the first optimization algorithm for directly optimizing 0-1 loss of SRNN that has provable convergence and monotonically decreasing

the 0-1 loss. It is while other existing methods in the literature use surrogate objective functions instead of 0-1 loss. Therefore, they do not have guarantee on minimizing the original 0-1 loss. Proposed optimization in this chapter is based on EM algorithm and is inspired by K-means algorithm [81]. In this chapter, for the first time, the interpretability of SRNN is explored and visual merits of the model are shown. SRNN is capable of learning the sub-clusters (or modes) of each class of the data in the original space of dataset and it represents them as centroids. This can be interpreted visually and gives an understandable importance of features for each mode of the class. It is shown that the algorithm is efficient because of its linear computational complexity. Through the rest of the chapter, we will first introduce the proposed method and then the experimental results are presented in section 2.3. The discussion will be in section 2.4 and conclusion in section 2.5.

2.2 The Proposed Method

Assume dataset X contains $\{x_1, x_2, ..., x_N\}$ data points and corresponding target points set, say Y, that contains $\{y_1, y_2, ..., y_N\}$. Here, $x_i \in \mathbb{R}^D$ and $y_i \in \{1, 2, ..., M\}$. The SRNN model consists of a set of points $C = \{c_1, c_2, ..., c_K\}$ with their corresponding labels $\hat{Y} = \{\hat{y_1}, \hat{y_2}, ..., \hat{y_K}\}$, where K is the number of centroids. At the test time, prediction of an input is the label of closest centroid to that input. The problem of learning a SRNN is

$$\min_{\{(c_j, \hat{y_j})\}_1^K} \quad \sum_{i=1}^{N} \sum_{j=1}^{K} L(y_i, \hat{y_j}) I_C(x_i, c_j) \tag{2.1}$$

where I_C represents an indicator function that operates over the elements of set C and is defined as,

$$I_C(x_i, c_j) = \begin{cases} 1 & \text{if } j = \underset{\{k\}_1^K}{\operatorname{argmin}} \quad r_{ik} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

In (2.1) j, c and $\hat{y_j}$ represent centroid number, centroid coordinate and label of the j^{th} centroid, respectively. L is a 0-1 loss function that outputs 0 if both of

its arguments are the same, otherwise, L outputs 1. r_{ik} is $||x_i - c_k||_2$ which is the squared Euclidean distance between k^{th} centroid and i^{th} sample. $I_C(x_i, c_j)$ is an indicator function that outputs 1 if j^{th} centroid is the closest centroid from set of $\{c_k\}_1^K$. $\{c_k\}_1^K$ represents set of centroids. Note that in other papers, centroids are also called prototypes. Problem (2.1) resembles K-means [81] problem except that in K-means, the loss function is $||x_i - c_k||_2$. The problem in (2.1) is NP-complete [135]. Inspired by K-means [81], we provide the following iterative approach for optimizing problem (2.1) which includes assignment and update/centroid steps.

Assignment step: The assignment step consists of two parts. First part is assigning the data points to their nearest centroid. This part is essentially calculating values of I_C for all data points. Second part of this step is assigning optimal values to $\{\hat{y}_k\}_1^K$. The problem for this step can be written as

$$\min_{\{\hat{y_j}\}_1^K} \quad \sum_{x_i \in S_j} L(y_i, \hat{y_j}) \tag{2.3}$$

in (2.3), $S_j = \{x_i | I(x_i, c_j) = 1 \quad \forall i = 1, 2, ..., N\}$. Problem (2.3) is in fact a constant predictor for set of S_j; hence, the optimal solution to this problem is majority of labels of samples assigned to the j^{th} cluster. Mathematically, solution is $\hat{y_j}^* = \text{mode}(\{y_i | I(x_i, c_j) = 1 \quad \forall i = 1, 2, ..., N\})$. This step takes at most $\mathcal{O}(N)$.

Update/Centroid step: In K-means algorithm, the centroid step is simply finding the average over the assigned samples to each cluster. However, here the centroid problem is not that simple because the centroid problem is affected by all the samples in the trainset, not just by the assigned samples to the centroid. First, we will describe the centroid problem and then show the centroid problem is affected by all the data points. After that, we show that the centroid problem is a NP-hard binary classification problem and then we propose a novel surrogate to approximate the solution. Approximation of an upper bound is not possible due to NP-hardness of the node problem. We show that the computational complexity of optimization method for the node problem is linear over the trainset; hence, the algorithm is efficient.

Each centroid has to be optimized individually. The optimization of the k^{th}

centroid can be defined as follows:

$$\min_{c_k} \quad \sum_{i=1}^{N} \sum_{j=1, j\neq k}^{K} (L(y_i, \hat{y_j}) I_C(x_i, c_j) + L(y_i, \hat{y_k}) I_C(x_i, c_k)) \tag{2.4}$$

Problem (2.4) is defined over the k^{th} centroid and the rest of centroids are kept constant. The problem is not defined over the samples of k^{th} centroid since the k^{th} centroid can effectively change any of the sample assignments ($I_C(.)$); hence, affecting the loss over the whole model. To further simplify (2.4), we rewrite problem of (2.4) using a step function $U(.)$ and set of $C'_k = C - \{c_k\}$.

$$\begin{aligned} \min_{c_k} \quad \sum_{i=1}^{N} \sum_{j=1, j\neq k}^{K} & (L(y_i, \hat{y_k}) I_{C'_k}(x_i, c_j)(1 - U(r_{ik} - r_{ij}))... \\ & + L(y_i, \hat{y_j}) I_{C'_k}(x_i, c_j) U(r_{ik} - r_{ij})) \end{aligned} \tag{2.5}$$

Note that c_k is hidden inside r_{ik}. Analyzing problem (2.5) gives us the intuition that by increasing and decreasing r_{ik}, the i^{th} sample can be assigned to C'_k and c_k, respectively. The i^{th} sample gets assigned to C'_k if $r_{ik} > r_{ij^*}$, where j^* represents the index of closest centroid in C'_k to i^{th} sample. On the other hand, the i^{th} sample gets to c_k if $r_{ik} \leq r_{ij^*}$. As a result of this change of assignment for i^{th} sample, the sample might get classified correctly or wrongly. Some samples will get classified correctly only if they are assigned to c_k (we call the set of such samples as S_1) and some samples will only get classified correctly if they are assigned to C'_k (we call the set of such samples as S_{-1}). The rest of the samples will not affect the outcome of solving problem (2.5) since they will anyway get classified correctly or wrongly regardless of their assignments. Problem (2.5) is in fact a 0-1 loss binary classification problem; hence, it is NP-hard [89]. Further, due to the nature of the problem, its search space is exponentially large over the number of data points since any combination of samples assigned to k^{th} centroid or rest of the model, can be a solution. Additionally, to verify that a combination is feasible, a feasibility problem that satisfies the following constraints must be solved:

$$\begin{aligned} r_{ik} - r_{ij^*} < 0 \quad \forall i \text{ assigned to } c_k \\ r_{ik} - r_{ij^*} > 0 \quad \forall i \text{ assigned to } C'_k \end{aligned} \tag{2.6}$$

Condition (2.6) is quadratically constrained feasibility problem whose constraints are concave; hence, verifying feasibility of a solution for (2.5) is NP-hard [94]. Therefore, an exhaustive search is not feasible and we introduce an algorithm that can approximate a solution for problem (2.5) using a novel surrogate objective function. Note that many 0-1 loss classification problems are NP-hard but in machine learning, their solutions are approximated using a surrogate objective function [89].

Relation to EM algorithm: The first step represents a part of the expectation step. Although, the latent variables are not explicitly used in this chapter, they are expressing the correctness of assigning a sample to a centroid based on their labels (similar to [62]). Finding the proper assignment of samples to S_1 or S_{-1} in update step is equivalent to calculating the posterior probabilities. Posterior probability is the probability of assigning a sample to a centroid (or set of centroids) while being classified correctly (for k^{th} centroid, it is $(1-L(y_i, \hat{y_k}))(1-U(r_{ik}-r_{ij^*}))$). Problem 2.5 represents the maximization step in which the posterior probabilities are maximized.

Approximating a solution: To approximate problem (2.5), a proper surrogate objective function is needed. Intuitively, problem (2.5) encourages c_k to be close to samples of S_1 and to be away from samples of S_{-1} at least by a distance of r_{ij}. Therefore, we propose solving of the following surrogate problem for $\mu = 0 \rightarrow 1$

$$c_k^*(\mu) = \underset{c_k}{\operatorname{argmin}} \quad \sum_{x_i \in S_1} r_{ik} + \sum_{x_i \in S_{-1}} relu(\mu r_{ij^*} - r_{ik}) \tag{2.7}$$

where $relu(.)$ is a rectified linear unit and μ is a hyperparameter that must be increased from 0 to 1. The intuition behind μ is that the optimum of problem (2.5) does not necessarily classify all the samples correctly and some of them might still get classified wrongly. When value of μ is small, then samples of class S_{-1} are allowed to be classified wrongly and as μ increases, then correct classification of samples in S_{-1} becomes crucial (μ is similar to coefficient of slack variables in objective function of SVM that controls misclassification of samples). The proposed surrogate function is continuous and it is easy to calculate its gradient. However, it is not convex and may have several minima for a constant $\mu > 0$.

Therefore, for each constant value of μ, we find a local optimum ($c_k^*(\mu)$) using Stochastic Gradient Decent (SGD). For $\mu = 0$, problem (2.7) has a global minimum which is average of samples in S_1. For each value of μ, we initialize the SGD algorithm from the optimum found for previous value of μ. The error of problem (2.7) is evaluated for each value of $c_k^*(\mu)$ along the path of $\mu = 0 \rightarrow 1$ and $c_k^*(\mu)$, and a result with the smallest error is picked as the new value for c_k. At $\mu = 0$, the optimization starts with average of S_1 if it has smaller error than previous value of c_k. This guarantees monotonic decrease of objective function in (2.4).

The approximation of solution for a centroid with α gradient steps will take at most $\mathcal{O}(\alpha ND)$. Therefore, solving all K centroids will take $\mathcal{O}(\alpha NDK)$.

To avoid extra calculation of distances between samples and centroids, it is possible to store all the distances between samples and centroids in a sorted manner and just updating it after each time a centroid is updated. This will take at most $\mathcal{O}(NK)$. Therefore, centroid step will take at most $\mathcal{O}(\alpha NDK)$.

Initialization: For initialization of EM-SRNN, we used several methods and final accuracy of all methods were found to be similar. We used random initialization from trainset, K-means and K-means over each class. However, K-means over each class had far better initial error than the rest.

Theorem 1 (Convergence of EM-SRNN)**.** *For any dataset consisting of X as set of inputs, Y as corresponding set of target labels and K as number of centroids, the EM-SRNN decreases the loss function monotonically and converges in a finite number of iterations.*

Proof. By iterating over the assignment and the centroid steps, the loss function decreases or at least remains the same. The total number of different assignments of samples to the centroids is finite. The objective function in (2.1) is bounded from below by 0. Together, the loss function will not decrease over more than a finite number of iterations in both steps. Thus, at some point none of EM-SRNN steps can decrease the error and the algorithm terminates. This proof is similar to proof of convergence for K-means. □

The EM-SRNN guarantees monotonic decrease of the error in every iteration

and converges in a finite number of iterations (in the experiments in next section, it took around 20 iterations over both steps).

Based on the computational complexity analysis provided in the previous sections, we can state the complexity as the following theorem.

Theorem 2 (Linear complexity of EM-SRNN)**.** *The computational complexity of EM-SRNN is* $\mathcal{O}(\alpha NDK)$*.*

2.3 Experiments

In this section, experimental results are presented to demonstrate the merits of the proposed algorithm in terms of accuracy and interpretability by human vision. EM-SRNN is compared with several state of the art models that are similar to EM-SRNN. Intuitively, SRNN partitions the space into K disjoint parts. Where K is the number of centroids/prototypes. For centroid i, the partition that belongs to i^{th} centroid consists of a partition where all points in that partition have smaller distance to i^{th} centroid than their distance to any other centroid. Thus, there are K different partitions. These partitions are disjoint since all the points inside each partition is only assigned to its closest centroid. Therefore, one of the closest models to compare with it is decision trees since they partition the space into their leafs and each leaf presents one partition. The decision tree partitions space through its split nodes. Each split node cuts the space into two disjoint partitions. The space of each child node of a split node consist of space inside the cuts created by its ancestors. As a result each leaf node consists of one partition. Another similar model to SRNN is using Radial Basis Functions (RBFs) as transformation to a SVM classifier (RBF-SVM). First K radial basis functions are learned using K-means and then SVM is learned on these basis functions [124]. The sense of similarity between RBF-SVM and SRNN is that they both operate over K centroids/prototypes. We also showed the performance of K-means over each dataset. For this model, a K-means is learned and then label of each centroid is assigned using the assignment step explained earlier in the proposed method. We also have shown the performance of running K-means only over samples of

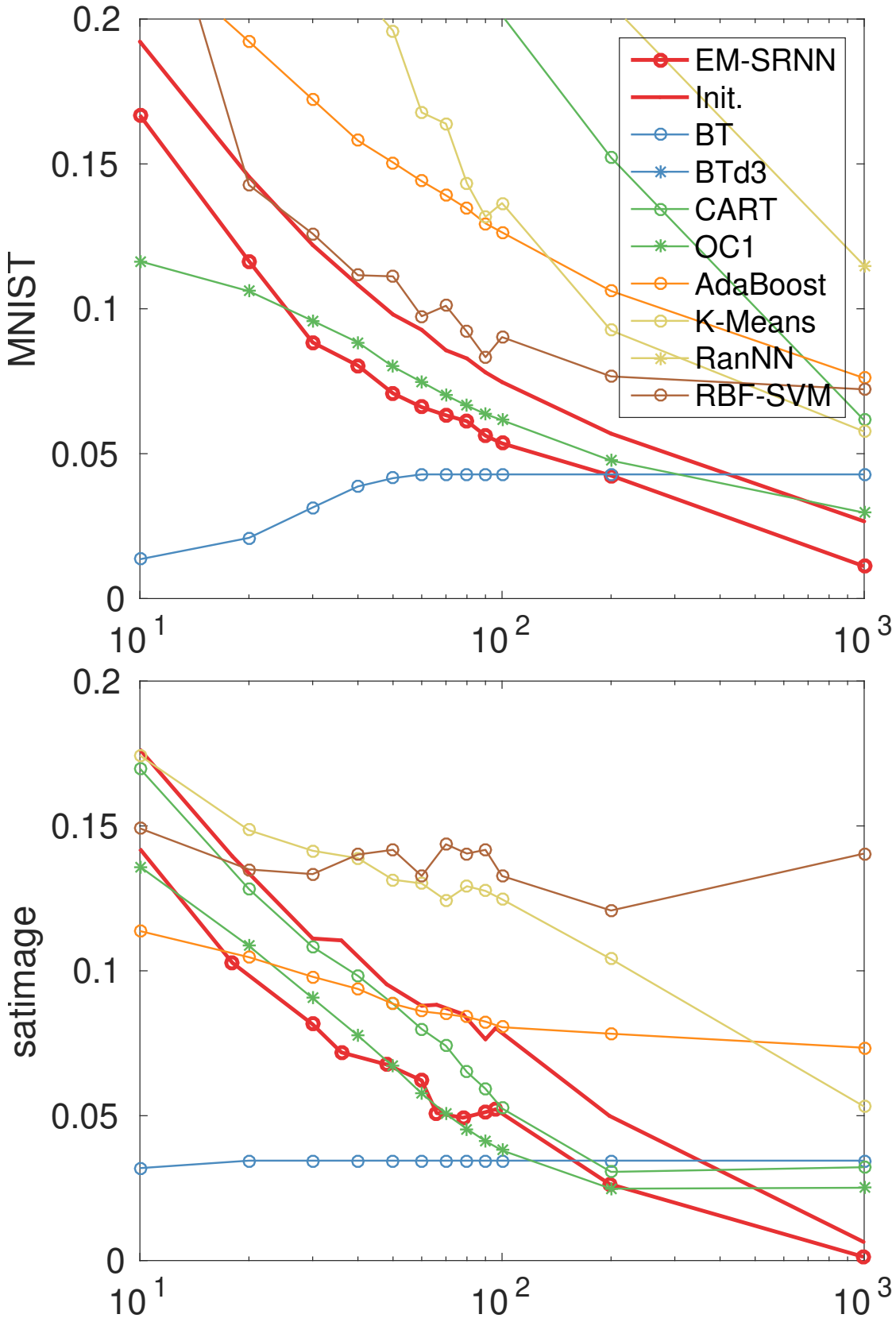


Figure 2.1: Comparison of various models over several datasets of MNIST and satimage. Vertical axis presents the train errors. Horizontal axis presents the number of components in each model.

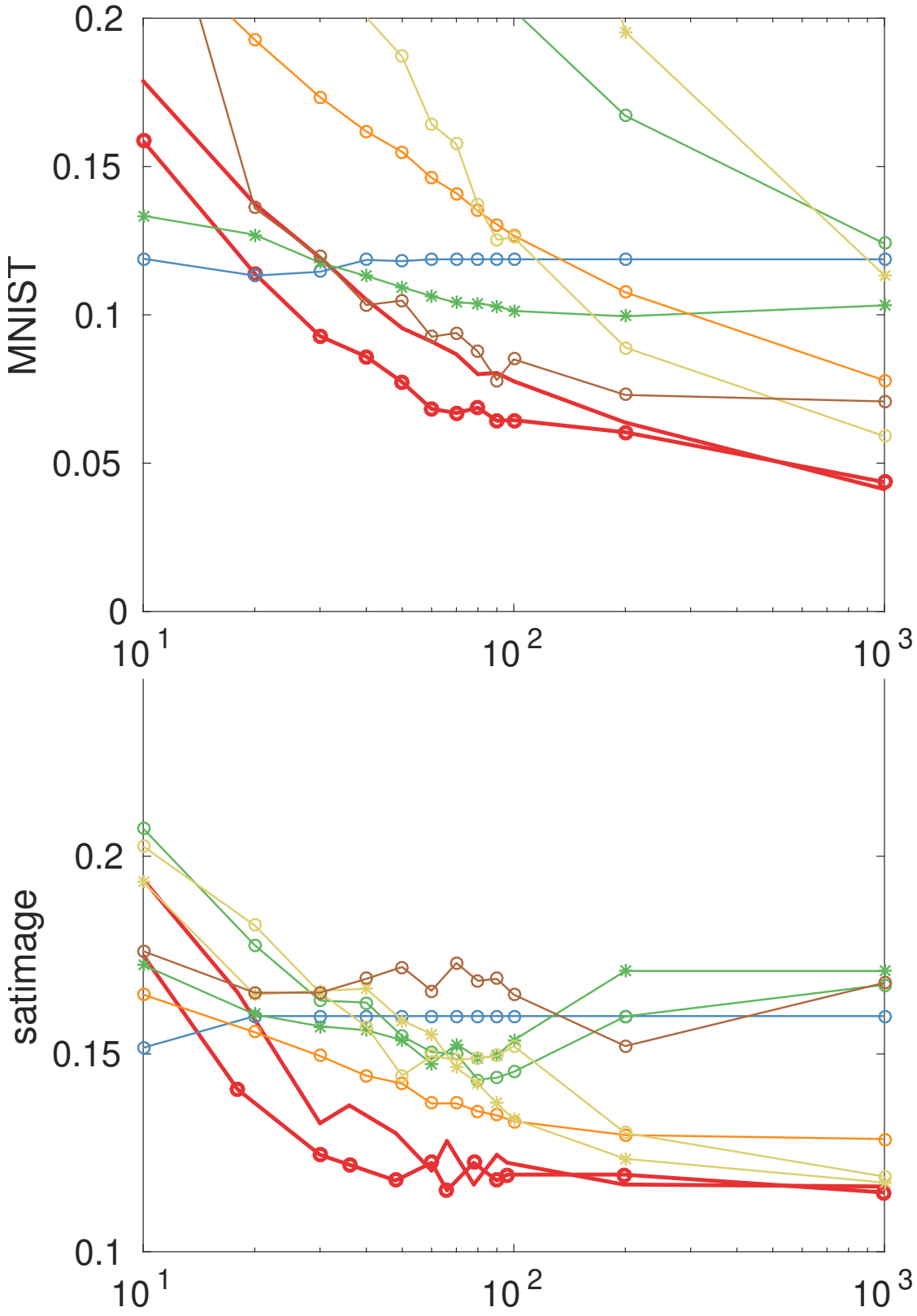


Figure 2.2: Comparison of various models over datasets of MNIST and satimage. Vertical axis presents the test errors. Horizontal axis presents the number of components in each model.

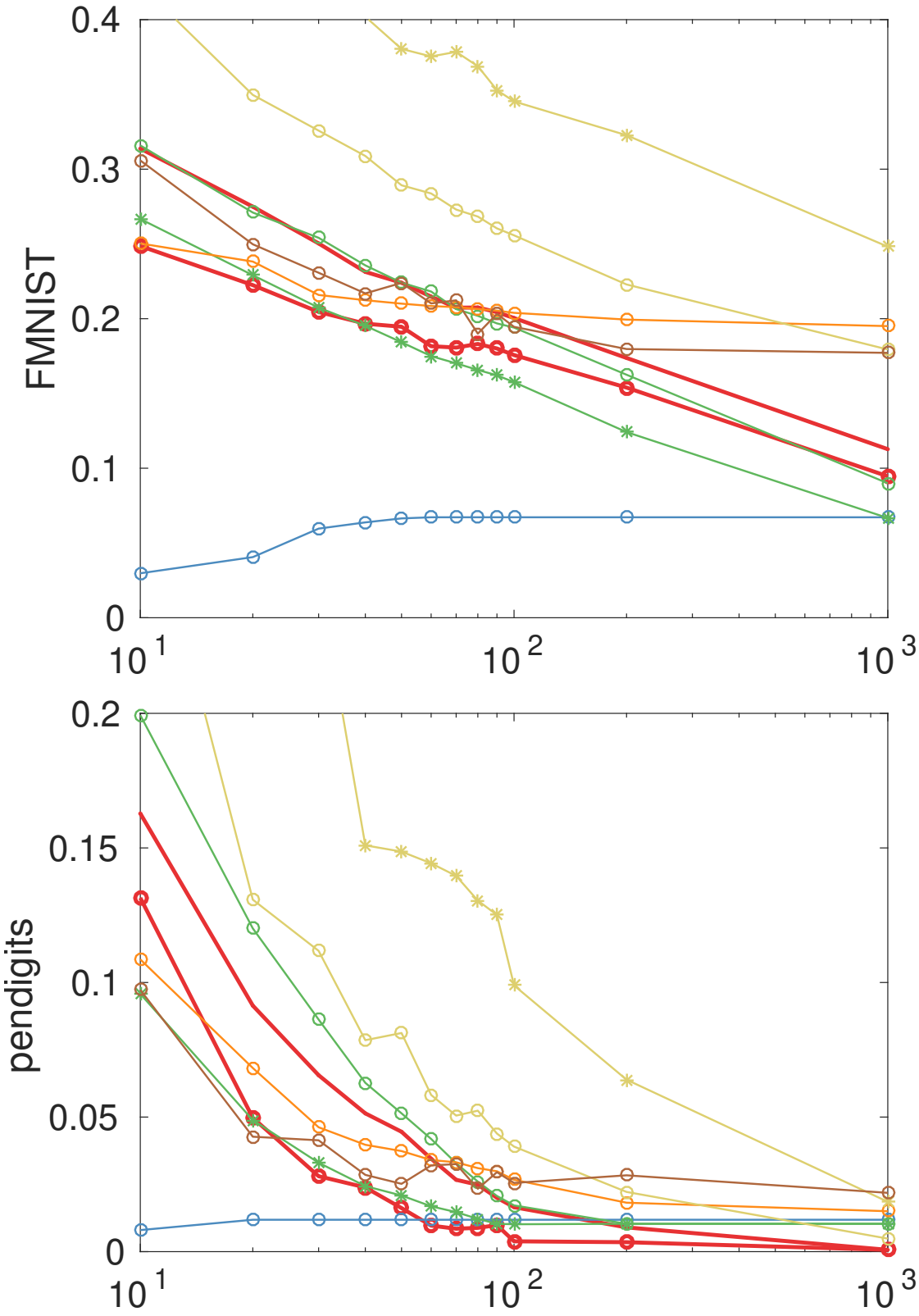


Figure 2.3: Comparison of various models over datasets of FMNIST and pendigits. Vertical axis presents the train errors. Horizontal axis presents the number of components in each model.

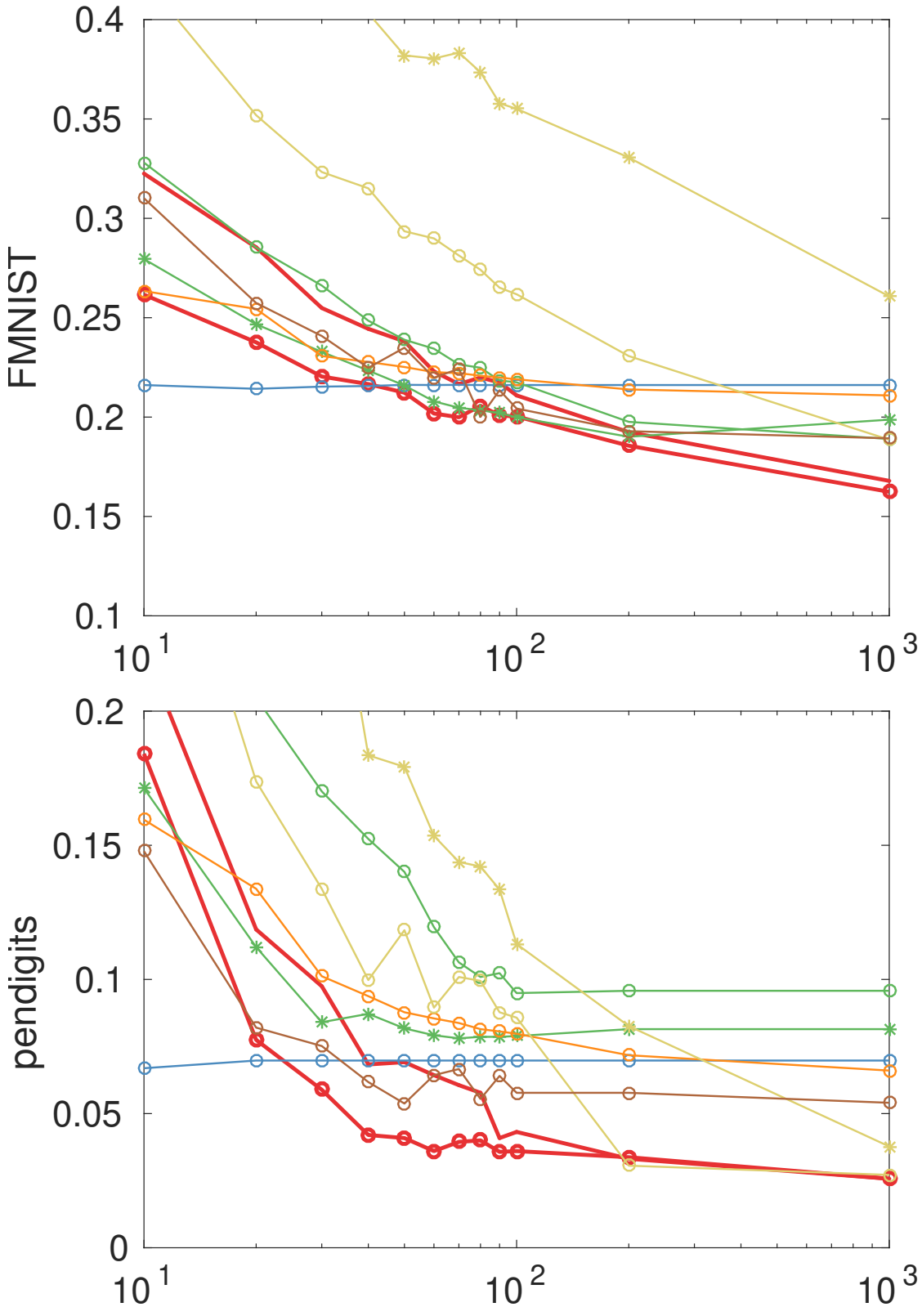


Figure 2.4: Comparison of various models over datasets of FMNIST and pendigits. Vertical axis presents the test errors. Horizontal axis presents the number of components in each model.

each class which is one of the initializations. The setup of this model is that $\frac{K}{M}$ centroids are learned for each class using K-means algorithm and each centroids label is the same as its training sample's label. AdaBoost.M2 [44] is also included and the sense of similarity is that each term in AdaBoost can have a similar effect as a centroid in SRNN. The trees in this model are constrained to have same number of leaf nodes as K. In the Random Nearest Neighbor (RanNN), K random samples are selected to be used as a Nearest Neighbor model. The model is also compared with Boundary Tree (BT) that has similar number of children at the decision nodes of the tree.

In figures 2.1, 2.2, 2.3 and 2.4, EM-SRNN is compared with other models for train and test errors. The horizontal axis presents the number of centroids, trees, leaf nodes, RBFs and number of children for EM-SRNN, GBT, tree, RBF-SVM and BT, respectively. The vertical axis shows the classification error over train sets (curves (a),(c)) and test set (curves (b),(d)). Overall, we observed that EM-SRNN outperforms other models in test with a large margin in most datasets. All the datasets are downloaded from [25]. From the train error curves in figure 2.1, it is clear that EM-SRNN is effective in decreasing the error. On average, EM-SRNN decreased train error by 25%. As number of components increase all the train error decreases in all models. However, overfitting happens in some of the models and test error increases as number of components increase. An interesting observation for EM-SRNN is that it seems the test error does not increase by adding new components to the model. In other words, by adding new components to the EM-SRNN the test error only decreases or remains the same. This observation has been consistent in all the datasets used in the experiments. However, note that by increasing the number of components the model's accuracy becomes similar to nearest neighbor model. The reason BT has very low train error is because the algorithm tends to save almost the whole trainset in its structure. However, BT had low accuracy and for higher accuracy, it is better to use them in an ensemble fashion [83].

The computational complexity of EM-SRNN is $\mathcal{O}(\alpha NDK)$. This computational complexity is faster than some of other models such as decision trees

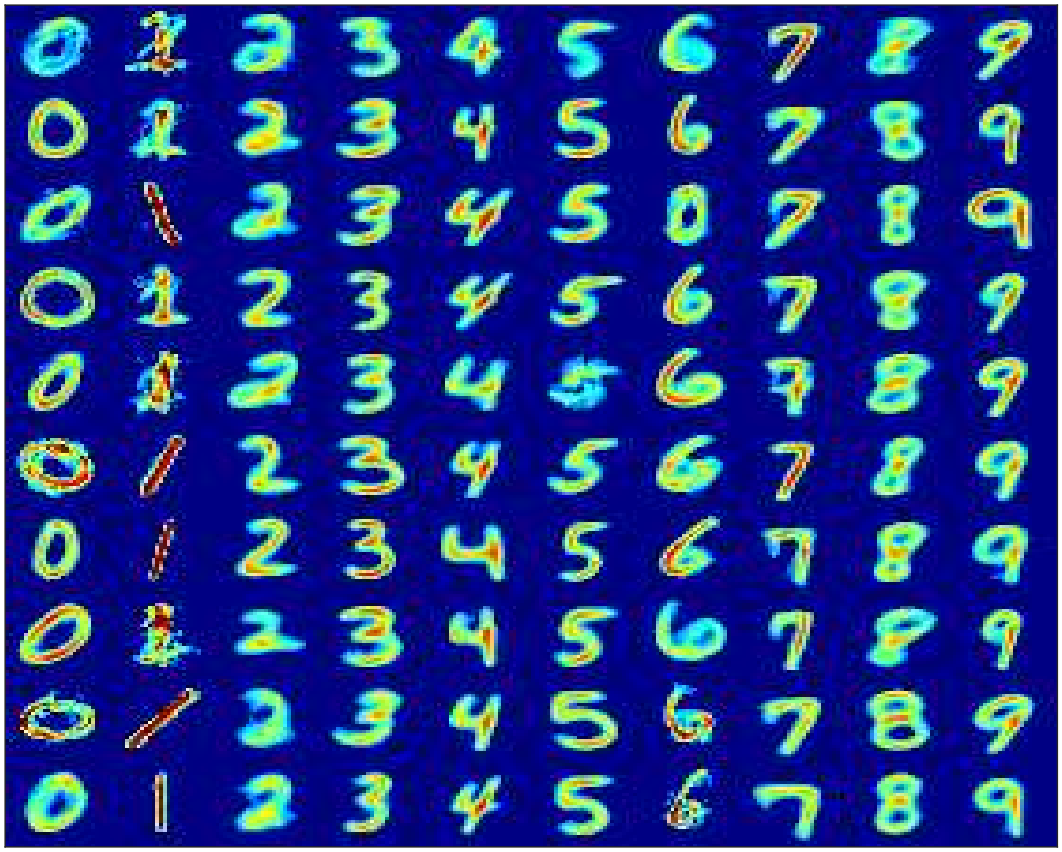


Figure 2.5: Centroids of EM-SRNN shown for MNIST. Each column presents centroids of the same class.

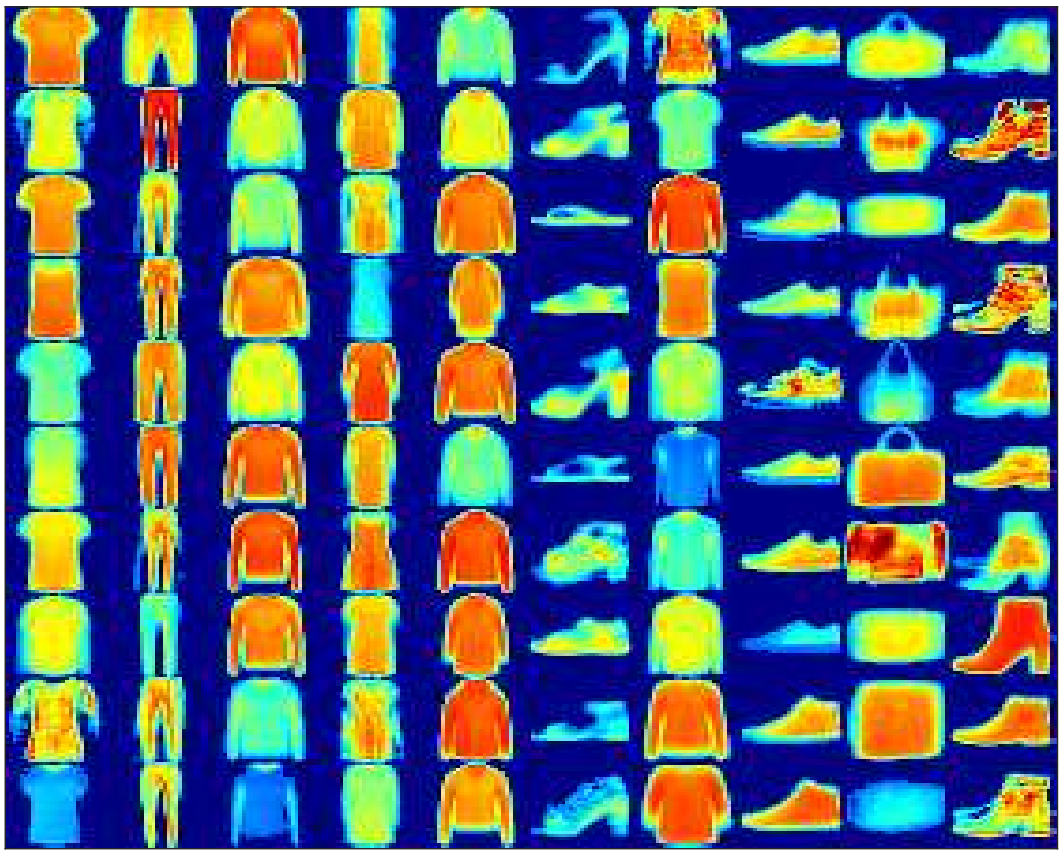


Figure 2.6: Centroids of EM-SRNN shown for FMNIST. Each column presents centroids of the same class.

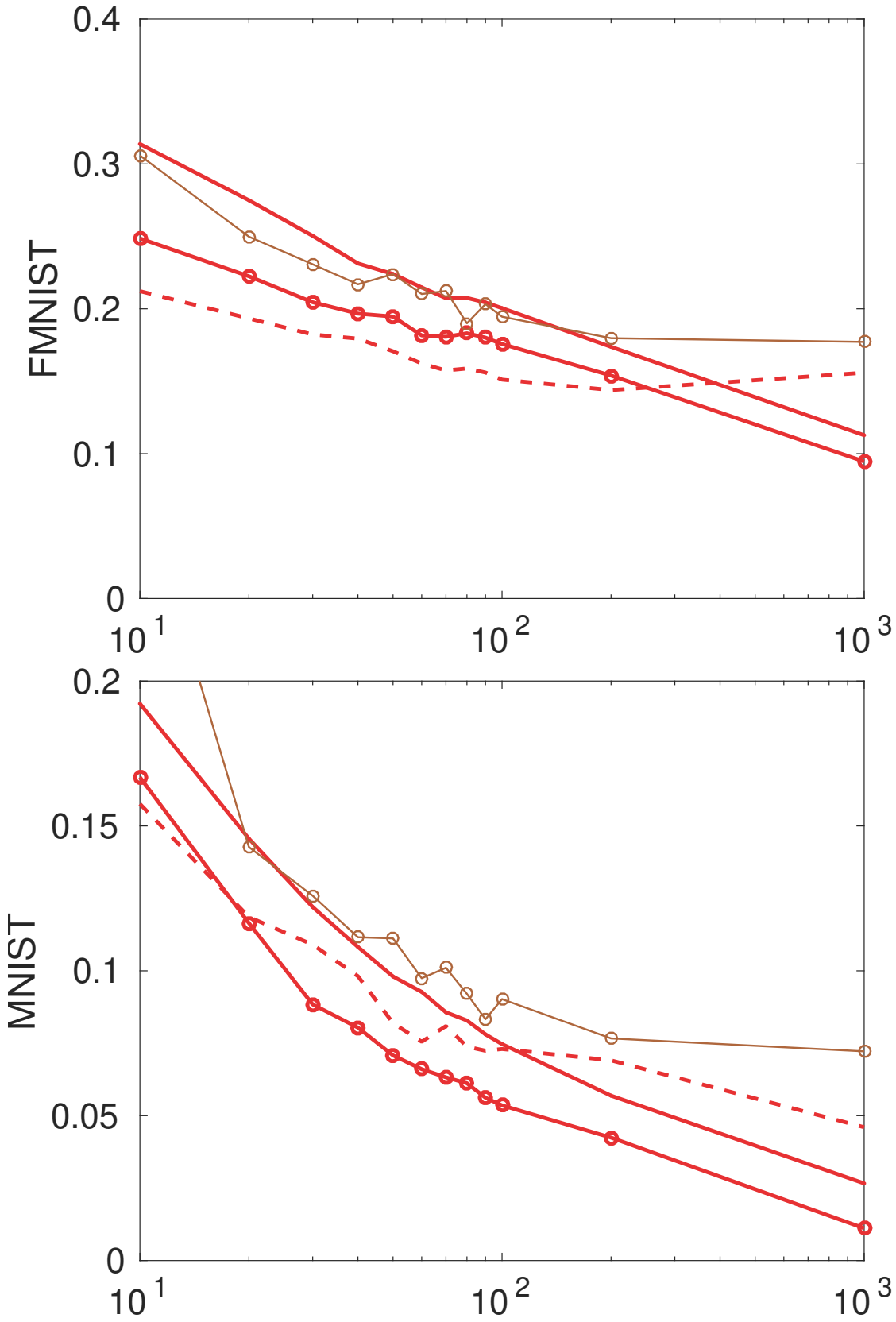


Figure 2.7: Comparison of EM-SRNN, EM-SRNN-SVM and RBF-SVM. RBF was trained over the centroids of EM-SRNN to observe how much improvement can occur. The vertical axis shows the train errors.

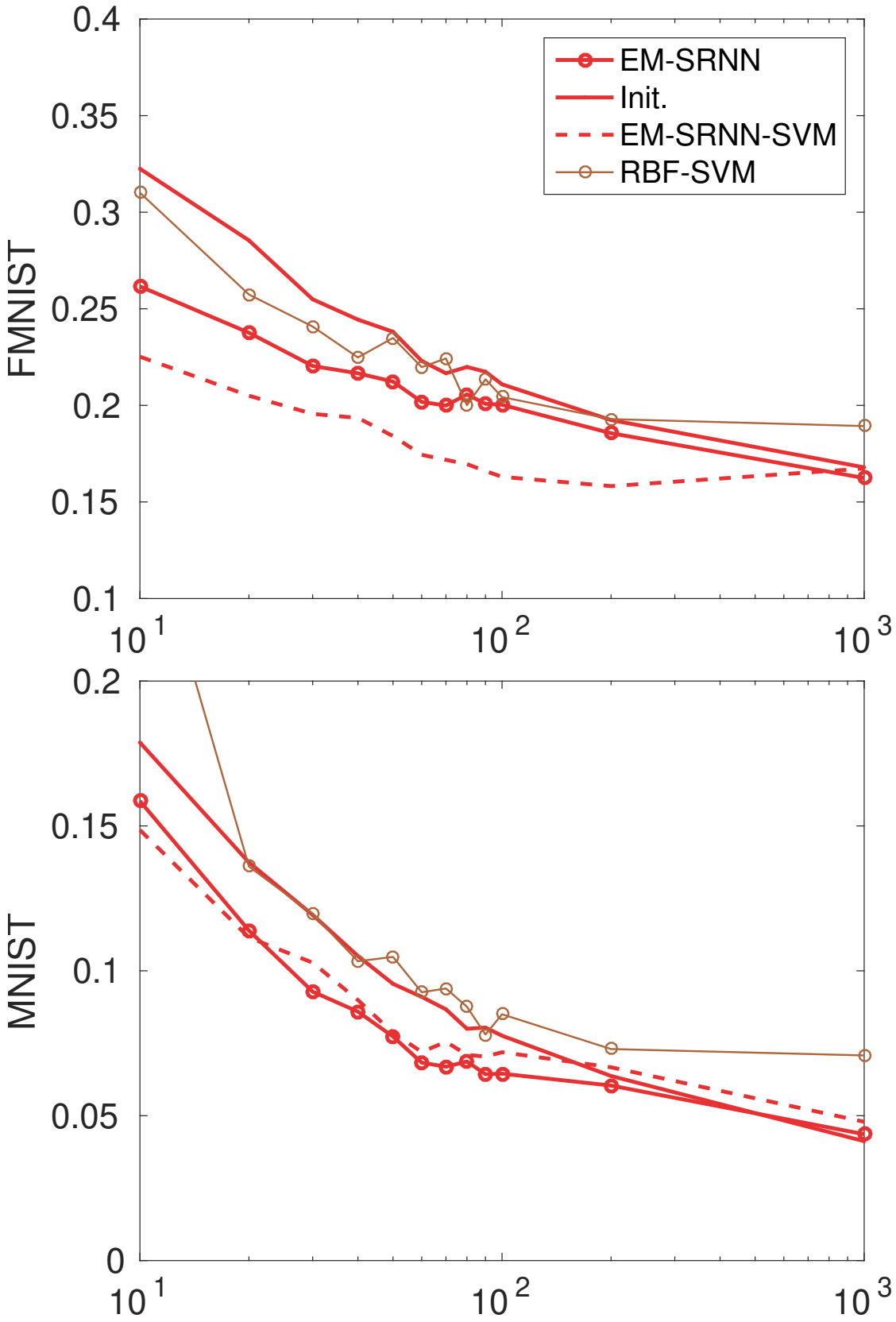


Figure 2.8: Comparison of EM-SRNN, EM-SRNN-SVM and RBF-SVM. RBF was trained over the centroids of EM-SRNN to observe how much improvement can occur. The vertical axis test errors.

$\mathcal{O}(NlogNDlogK)$, AdaBoost $\mathcal{O}(KNlogND\Delta)$ (Δ is tree's depth) is faster. However, the inference of decision tree models can be faster since the computational complexity of the inference is broken down in a tree structure. Other models such as K-means and RBF-SVM seem to be faster than EM-SRNN. However, K-means perform poorly at the test time and RBF-SVM requires many restarts to find proper parameters for the hyper-parameters of the RBFs.

Interpretability: EM-SRNN model classifies the input data based on a measure of similarity between the input sample and its centroids. This potentially means that EM-SRNN should have learned important features of each class and of sub-clusters of the classes so that it can find the most similar centroid to an input. In other words, such features are recognizable by human eye. Such features even have values/intensity similar to its assigned samples in the trainset, essentially making them a representative of those samples. This type of interpretability is different from type of interpretability in decision trees since decision trees focus mostly on only dissimilarities of trainset samples. Therefore, for a decision tree, this might not be easy by a human eye to understand the contributing features of trainset to a specific class or sub-cluster of a class and further computation is needed. On the other hand, EM-SRNN can perfectly present features and their intensities that contribute to a sub-cluster; hence, making it possible to understand by human vision how a cluster of that sub-class will look like.

In figures 2.5 and 2.6, the centroids of EM-SRNN are visualized for MNIST and FMNIST datasets for $K = 100$ (for each class, 10 centroids are used). The used color map is heatmap; where, blue pixels represents 0 and red pixels represents 1. In each column of images in figures 2.5 and 2.6, centroids of one class are shown. Based on mentioned type of interpretability for EM-SRNN, what we should see in images of figures 2.5 and 2.6 is various shapes for a same class. For examples, in figure 2.6, at 6^{th} column from left in FMNIST dataset, we can see various Sandals with different shapes that are clearly recognizable by human eye. Some of them contain high heels while some of them are flat at the bottom. It is even interesting to see that pixels in some of the centroids have low values which is also true about the dataset itself. The first column present various types of T-shirts. As can

be observed some of the T-shirts have wider bottom while others have tighter bottoms. Some of the T-shirts have no sleeves or shorter sleeves than the others. Second column shows various types of pants. An important observation here is the tightness of pants legs. The third column seems to be jackets. Another example is column nine that contains various types of bags that may or may not have handles. There can be seen other examples in FMNIST, we see variations of the samples are learned through EM-SRNN.

Also in MNIST, we can observe various ways of writing different numbers that are learned by the EM-SRNN. For example, various shapes of writing 1 can be observed. Some write it obliquely while others write it vertically and some other would add a tiny line at the bottom and an oblique line to the top of character of 1. Zero is also written in various shapes and orientations. For example some of the zeros are written almost horizontally, some are written vertically and some are identical to a circle. Similar variations can be observed for other numbers such as 8, 9 or 6. Number 3 is almost written with the same pattern for everybody.

In general, EM-SRNN has made it possible to observe various sub-clusters and modes of the data for each class separately while at the same time it can classify the samples with a proper accuracy.

Using EM-SRNN as basis for RBF: One of the well-known models for classification is kernel SVMs. One popular method is to learn some radial basis functions (RBF) and then train SVM on top of these functions. The hyperparameter of RBF must be tuned properly and then the best setup is selected by cross-validation. The best hyperparameter is found by a grid search over various values for the hyperparameter. Conventionally, the centers for RBFs are found through K-means algorithm. However, the whole procedure is a filter approach and there is no guarantee for finding the best centers (given that K-means is an unsupervised algorithm). Here, we are interested to how the model would perform if centers are gathered from EM-SRNN. In figures 2.7, 2.8, 2.9 and 2.10 such experiment is done and the accuracies are presented. The models presented in figures 2.7, 2.8, 2.9 and 2.10 consist of EM-SRNN, initialization of EM-SRNN (Init.), RBF-SVM and applying SVM to EM-SRNN (EM-SRNN-SVM). By observing the

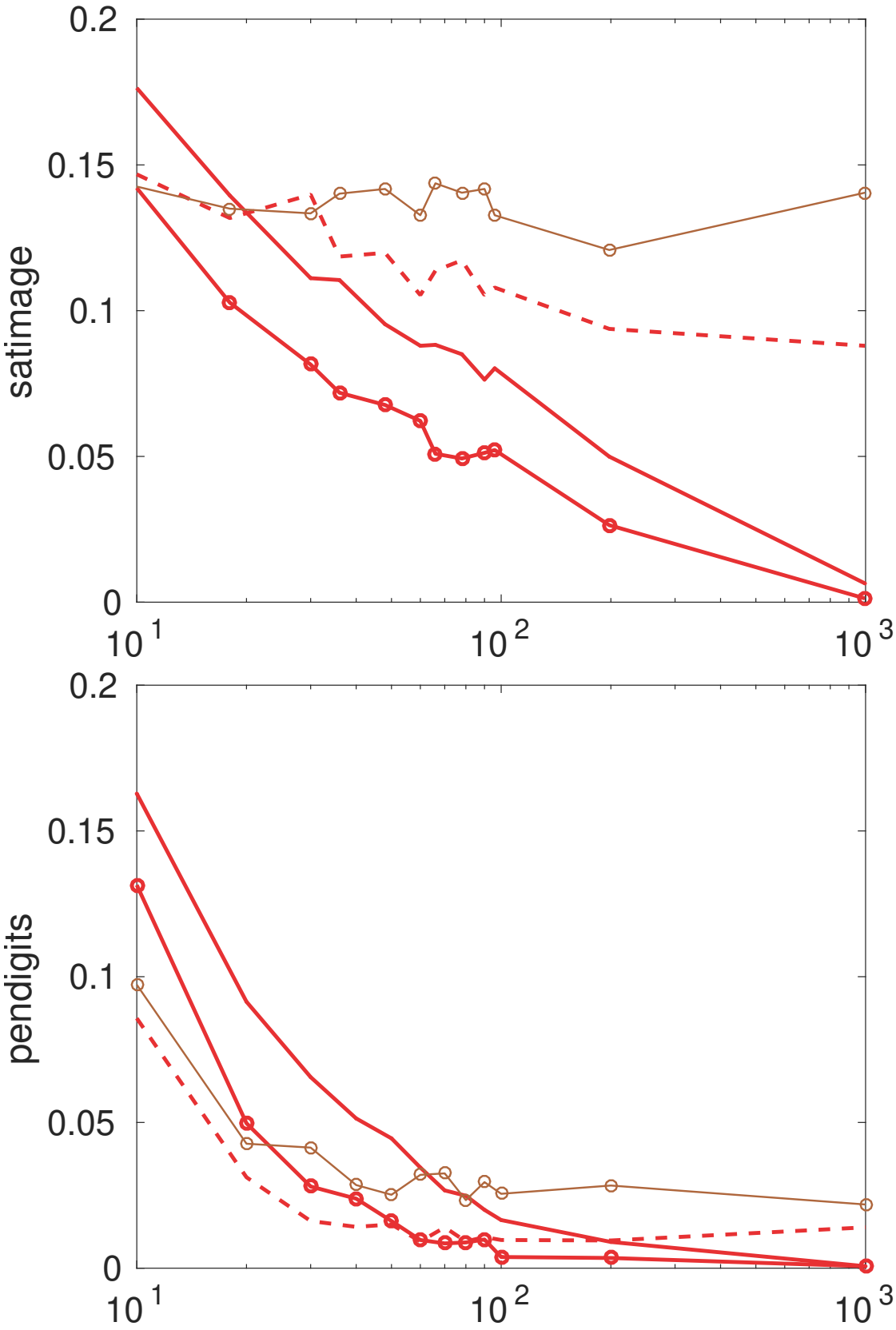


Figure 2.9: Comparison of EM-SRNN, EM-SRNN-SVM and RBF-SVM. RBF was trained over the centroids of EM-SRNN to observe how much improvement can occur. The vertical axis shows the train errors.

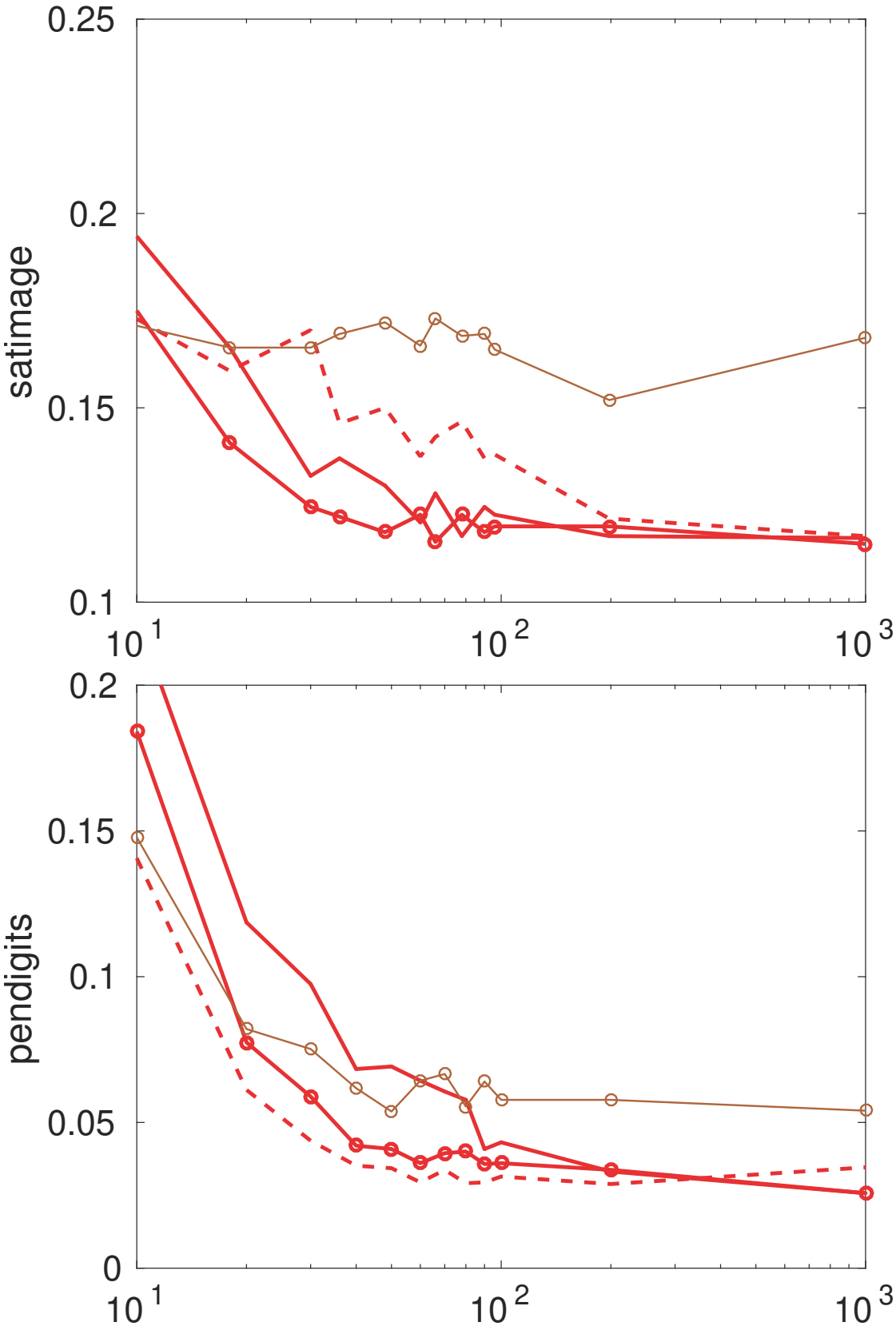


Figure 2.10: Comparison of EM-SRNN, EM-SRNN-SVM and RBF-SVM. RBF was trained over the centroids of EM-SRNN to observe how much improvement can occur. The vertical axis shows test errors.

performance of EM-SRNN-SVM, we have noticed that in most cases, applying SVM to EM-SRNN would decrease train error marginally and it might even increase the test accuracy. Note that EM-SRNN itself has high accuracy compared to other similar models. This actually indicates that EM-SRNN has learned such good centroids for classification that training SVM on top of can not improve it. A proper approach to improve further in this direction would be to learn both centroids and SVM jointly (a wrapper approach).

2.4 Discussion

The objective function in (2.7) is a surrogate objective function of (2.5). Mathematically speaking, various options can exist for this surrogate function. In our study we tried several options including using other distance metrics, a probabilistic approach, log of distances or using $relu(.)$ over the first term and restricting it to the penalize objective function for outside of r_{ik}. However, all used methods were either hard to handle numerically or were not able to follow the desired path of μ. Additionally, we also tried using the coefficient of μ in various ways such as multiplying it to other terms. However, out of all tried variants, only the one in (2.7) was both numerically easy to handle and was achieving lower errors for (2.5). Note that the problem (2.5) is very similar to the objective function of SVM but in SVM the objective function is quadratic and constraints are linear while in (2.5) by reformulating the problem it is trivial to notice that (2.5) is in fact an optimization problem with piecewise constant objective function (a 0-1 loss) and quadratic constraints (constraint over distance of centroid from samples). It must be mentioned that the constraints are concave; hence, even replacing the objective function with a quadratic objective function (as a surrogate) does not help solving (2.5) since it is an NP-hard problem [94].

2.5 Summary

This chapter presented the first algorithm for direct optimization of 0-1 loss for the SRNN model that has provable convergence guarantee and monotonic decrease of objective function. The algorithm consists of two steps. First step is the assignment step in which the assignment of samples are calculated and based on that the label of each centroid is determined. Second step consists centroid step and optimization each centroid individually. This consists of changing position of one centroid at a time while guaranteeing the decrease in 0-1 loss function. Since the centroid problem is NP-hard, we approximated it with a novel surrogate objective function. This step consists of finding local optimum of surrogate objective function along a path as a function of μ. Among all solutions, the one with lowest error is selected. It was shown that EM-SRNN is a linear-time algorithm, thus, making it possible to be applied to large high-dimensional datasets. EM-SRNN is specifically designed to be interpretable by humans. Therefore, for the first time we investigated interpretability of the model and it was shown how each centroid represents a modality of each class that is recognizable by humans. It was shown that EM-SRNN has superior or similar performance when compared to other state-of-the-art methods.

Chapter 3

Adversarial Label-Poisoning Attacks and Defense for General Multi-Class Models Based On Synthetic Reduced Nearest Neighbor

State-of-the-art machine learning models are vulnerable to data poisoning attacks whose purpose is to undermine the integrity of the model. However, the current literature on data poisoning attacks is mainly focused on ad hoc techniques that are only applicable to specific machine learning models. Additionally, the existing data poisoning attacks in the literature are limited to either binary classifiers or to gradient-based algorithms. To address these limitations, this chapter first proposes a novel model-free label-flipping attack based on the multi-modality of the data, in which the adversary targets the clusters of classes while constrained by a label-flipping budget. The complexity of our proposed attack algorithm is linear in time over the size of the dataset. Also, the proposed attack can increase the error up to two times for the same attack budget. Second, a novel defense technique based on the Synthetic Reduced Nearest Neighbor (SRNN) model is

proposed. The defense technique can detect and exclude flipped samples on the fly during the training procedure. Through extensive experimental analysis, we demonstrate that (i) the proposed attack technique can deteriorate the accuracy of several models drastically, and (ii) under the proposed attack, the proposed defense technique significantly outperforms other conventional machine learning models in recovering the accuracy of the targeted model.

3.1 Introduction

Machine learning models are known to be vulnerable to data poisoning attacks [93] at training-time. In such attacks, the adversary intentionally manipulates the training data by perturbing, adding, or removing training samples with the goal of deteriorating the integrity of the model, thus, resulting in under-performing models with low accuracy [5].

Alternatively, this scenario can be seen as learning under noisy data [82]. Such attacks have the intention of altering the decision boundaries of targeted model, thus, threatening the integrity of the model [66]. The modification to data is done by manipulating samples' information such as features and labels, or by inserting and removing samples. Generally, it is assumed that attacks are constrained by an attack budget to account for realistic consitions of such attacks [93, 88]. Altering the training samples can be seen as modifying the modalities of the data that generated the training samples, thus, deteriorating the consistency of the trainset and the trained model. Much of the literature in this field is focused on the poisoning attacks against specific models and their robustness against ad hoc attacks [93, 108]. Authors of [66] theoretically investigated the accuracy of classifiers under an adversarial attack that can modify a specific portion of the trainset.

Label manipulation is a common attack surface for adversaries[93]. The adversarial label manipulation attack tends to perturb the minimum number of labels (constrained by an attack budget), such that the resulting error is maximized. Finding the optimum of such attack is shown to be at least NP-hard [14, 86]. A baseline strategy is to perturb the labels at random. The study in [14] established

that random perturbation of labels can degrade the accuracy of SVM classifiers by flipping about 40% of the labels. However, this conclusion is limited to only binary classification problems in SVM models. Authors of [14] further showed that heuristics can improve the success of adversary in degrading the performance of the SVM. [86] proposed a similar approach, in which the attack consists of training a new ML model after poisoning each new sample to measure that sample's effect on the trained model's accuracy. However, this approach is computationally expensive. This is mainly due to the lack of knowledge about the relationship between the training set and test set [93]. In case this relationship is known, then it is feasible to find near optimal samples for label manipulation [129].

While the body of work on label noise is vast, to the extent of our knowledge, no study has yet focused on the label noise that exists in an underlying manifold of the data such as clusters. It also has never been explored how such noise or intentional adversarial attack can be dealt with. This is a very significant type of attack since the adversary can focus on attacking minority groups/clusters hidden inside the dataset, thus, deteriorating the integrity and diversity of the model, while remaining undetected.

Much of the current research is focused on ad hoc data poisoning attacks that are designed for specific machine learning models. Furthermore, the majority of existing attacks in the literature are limited only to binary class problems and gradient-based algorithms.

Therefore, to tackle the aforementioned issues, this chapter proposes the first novel adversarial label-flipping algorithm that is not restricted to a specific machine learning model. The proposed attack technique is the first that is based on the multi-modality of the data.

Afterwards, we propose a regularized SRNN model that can effectively find up to 80% of malicious samples through novel regularization and pruning techniques.

The optimization algorithm used here is based on the EM algorithm of [115] and is composed of three steps. The first step is the assignment step and consists of optimizing the regularization term and assigning the optimal label to each centroid. The update step consists of optimizing the centroids. We will show that in the

update step by using a proper surrogate objective function, the malicious samples are automatically excluded. The contributions of this chapter are as follows:

1. We propose a novel label poisoning attack mechanism based on SRNN which is not limited to specific machine learning models and classification tasks.

2. We propose a defense technique comprised of a novel regularization method and pruning strategy for eliminating maliciously perturbed training samples.

3. We experimentally demonstrate that the performance of the proposed attack mechanism is superior to similar attack/noisy techniques against a variety of well-known machine learning models and classification tasks.

4. We experimentally establish the feasibility of our proposed defense technique, and demonstrate that it can detect up to 80% of the malicious samples in a variety of known resilient machine learning models and classification tasks.

3.2 Related Work

Many papers have focused on manipulations in the feature space as the mode of the attack. In such settings, the adversary can corrupt both the labels and input features of the trainset. The literature in this area is generally focused on online learning setups and clustering as the model, where the adversarial strategy is to slowly displace the center of the cluster to induce misclassifications. Authors in [68] used such an approach in a trainset used for anomaly detection and demonstrated how the approach can gradually shift the decision boundaries of a centroid model. Another similar approach is introduced in [16].

Various other studies in the literature focus on gradient-based methods for selecting samples to poison in the context of SVM models [15, 84, 129, 130]. Manipulations of features in samples selected in this manner have been shown to incurr a devastating effect on reinforcement learning agents [8].

A data poisoning attack that uses labels as surface of attack can be modeled as noise in labels. Authors in [42] present a comprehensive survey of the label noise. In [43], inspired by [106], authors distinguish three types of label noises. First

type is label Noise Completely at Random (NCAR). This type of noise happens at random regardless of the features or the true class. The second type is Noise at Random (NAR). This type of noise happen when some classes are more likely to be noisy. However, this type of noise is easy to detect by a human since a specific class is being constantly targeted. The third type of noise is Noise Not at Random (NNAR). In such settings, mislabeling of the samples is related to the features of the samples and the mislabeling can happen at the decision boundaries.

In general, there are three approaches to tackle the noisy label issue. First approach is to use label noise-robust machine learning models. It has been shown that most models are intrinsically robust to noise [82]. However, some of the models are more robust than the others. For example, ensemble models are more robust than other machine learning models [39, 100, 67, 101, 99]. Also, in decision trees, the split criterion can improve robustness of the tree [1]. The second approach is to remove noisy samples using methods such as outlier detection [6, 58] or anomaly detection [24]. Some studies in the literature use heuristics to remove noisy samples. Examples of such heuristics can be found in reduced nearest neighbor [47]. The work in[47] proposes to remove samples whose removal does not affect misclassification of other samples. AdaBoost-based methods can also be used such as [123, 65]. The third approach consists of learning a model that is noise-tolerant such as [46, 64, 112]. However, these methods are based on assumptions made for the probability distribution of the noise. Other practical examples of the third approach consist of applying clustering algorithms for detecting mislabled samples through a nearest neighbor approach [132, 18] or confidence of the model prediction [37, 36]. Further, cross-validation itself can improve the robustness of the model against label noise [55].

As explored in this section, the state of the art in poisoning attacks is mostly comprised of attacks that focus either on specific machine learning models or are at most gradient based approaches. Therefore, they might not be applicable to general machine learning models, such as decision trees and forest models.

3.3 Proposed Method

3.3.1 Preliminaries

In this subsection we change the notation for SRNN for better compatibility with contents of this chapter.

Synthetic Reduced Nearest Neighbor(SRNN): A synthetic reduced nearest neighbor (also known as prototype nearest neighbor(PNN)) is a set of synthetic samples (centroids) that infer an input similar to a nearest neighbor model [115]. Assume a dataset of samples consisting of N tuples of observation, $\{(x_i, y_i)\}_{i=1}^{N}$. $x_i \in \mathbb{R}^D$ is the i^{th} sample's features and its target response is $y_i \in \{1, 2, 3, ..., M\}$. The SRNN model consists of a set of centroids/prototypes $C = \{(c_j, \hat{y_j})\}_{j=1}^{K}$. At the test time, prediction of an input is the label of closest centroid to that input. The problem of learning a SRNN model is similar to that of a k-means problem except that each centroid is associated with a label that represents the prediction of the centroid. Mathematically speaking, the optimization of the SRNN model using 0-1 loss is:

$$\begin{aligned} \min_{\{(c_j,\hat{y_j})\}_1^K} \quad & \sum_{i=1}^{N} L(y_i, NN(x_i)) \\ \text{s.t.} \quad & NN(x_i) = \hat{y_{j_i^*}} \\ & j_i^* = \underset{\{j\}_1^K}{\operatorname{argmin}} \quad d(x_i - c_j) \end{aligned} \tag{3.1}$$

where, $NN(.)$ represents the nearest neighbor function. $d(.)$ is a distance metric (chosen to be the Euclidean distance in this study). For simplicity, in the rest of this chapter, we use r_{ij} for $d(x_i - c_j)$. j_i^* represents the index of closest centroid to i^{th} sample. L is a 0-1 loss function that outputs 0 if both of its input arguments are equal, otherwise, it outputs 1. The problem of (3.1) is in fact the problem of finding a set of K synthetic samples that achieve minimum error as a nearest neighbor model with K samples. Intuitively, each centroid represents a modality of the data that the samples are generated from.

3.3.2 Modality-based adversarial label flipping

One common objective of adversarial data poisoning is to undermine the integrity of the trained model. This can be cast as weakening the performance of the trained model at the test time [93]. In other words, the goal of the attacker is to increase the error of the trained model. In this chapter, based on SRNN model, we propose the modality-based (or cluster-based) perturbation of the training labels. The problem of selecting optimal samples for proposed attack technique is as follows:

$$\begin{aligned} \max_{\{I_i, y_i^p\}_1^N} \quad & \sum_{(x_i,y_i)\in S^{train}} L(y_i, NN^*(x_i)) \\ s.t \quad & NN^* = \operatorname*{argmin}_{\{(c_j,\hat{y_j})\}_1^K} \sum_{x_i \in S^p} L(y_i, NN(x_i)) \\ & S^p = \{(x_i, (y_i(I_i - 1) + y_i^p(I_i)))\} \\ & \sum_{i=1}^{N} I_i \leq Cost \end{aligned} \tag{3.2}$$

Where, $NN^*(.)$ represents the optimal model trained over the poisoned dataset S^p. The goal is to increase the error over the trainset with true labels S^{train}. y_i^p represents the perturbed label and I_i is an indicator variable that is either 0 or 1. I_i is used for representing selected samples. The second constraint represents the poisoned dataset with perturbed labels. The third constraint shows the maximum allowed number of perturbations, given an attack budget ($Cost$).

The problem in (3.2) is a selection problem. The problem consists of selecting samples and changing their labels to another class such that the error of NN^* is maximized over S^{train}. Intuitively, the trained model should perform poorly on the trainset with true labels in the hope that it performs poorly at the test time. Finding optimal solution of (3.2) is NP-hard [14, 86]. As a consequence, finding global optimum of (3.2) is computationally intractable and not practical. Therefore, we propose an efficient greedy algorithm that approximates the solution for problem (3.2) and also satisfies the constraints. Initially, there are no samples selected for poisoning. Therefore, the adversary has to train NN^* over the trainset. Next step consists of selecting samples. This is done by fixing centroids of NN^*

while continuing optimization only over $\{I_i, y_i^p, \hat{y_j}\}$. At this step, the assignment of the train set samples are fixed and cannot be changed since the centroids are fixed. Therefore, the problem consists of changing labels of samples in S^p such that the prediction labels of some of the centroids in NN^* are changed, thus, increasing the error over S^{train}. This problem can be stated as follows:

$$\begin{aligned} \max_{\{I_i, y_i^p\}_1^N, \{y_j\}_{j=1}^K} \quad & \sum_{j=1}^{K} \sum_{(x_i, y_i) \in S_j^{train}} L(y_i, \hat{y_j}) \\ s.t \quad & \hat{y_j} = mode(\{y_i\}_{S_j^p}) \forall j = 1...K \\ & S^p = \{(x_i, (y_i(I_i - 1) + y_i^p(I_i)))\} \\ & \sum_{i=1}^{N} I_i \leq Cost \end{aligned} \tag{3.3}$$

In (3.3), S_j^p and S_j^{train} represent the trainset samples assigned to j^{th} centroid with perturbed and true labels, respectively. Note that the difference between (3.3) and (3.2) is that the centroids of clusters are fixed, thus, assignments are fixed. $\hat{y_j}$ represents the j^{th} centroid's optimal label. Therefore, in order to increase the objective function, $\hat{y_j}$ has to be changed through selecting samples for poisoning. $\hat{y_j}$ can only get changed if the majority label in S_j^p gets changed. Many heuristics can be applied here such as changing some of the labels to the second most frequent label in the S_j^p. One intuitive approach is to randomly change the labels of half plus one of the samples in S_j^{train} to the minority (i.e, least frequent) labels. This causes the information of the cluster (modality of the data) to become obscure and misleading. In other words, the attacker has to spend a specific cost to turn the $\hat{y_j}$ into a false label (minority label of the cluster). However, the attack has a limited budget and has to select clusters based on its budget. From a practical point of view, it can be seen as recognizing vulnerable groups in a data. All that remains is selecting clusters to attack. Assuming the cost of changing j^{th} label is $Cost_j$,

then the problem can be simplified as follows:

$$\begin{aligned} \max_{\{I_j\}_{j=1}^{K}} \quad & \sum_{j=1}^{K} Cost_j I_j \\ s.t & \sum_{j=1}^{K} Cost_j I_j < Cost \\ & I_j = \{0,1\} \quad \forall j = 1...K \end{aligned} \tag{3.4}$$

Problem (3.4) can be solved greedily by selecting from clusters with lower cost until the first constraint is violated. Other algorithms such as dynamic programming or other greedy approaches can also solve the problem in (3.4).

The proposed attack technique can have the capacity to increase the error over the trainset by two times the attack budget.

Theorem 3 (Fixed SRNN upper bound)**.** *Assuming that the centroids are fixed, and known to both user and adversary, the modality-based attack can increase the error of NN^* by at most $\mathcal{O}(2 \times Cost)$ over the trainset with unperturbed labels.*

Proof. The centroids are fixed. Therefore, the loss incurred by each centroid is independent of the other centroids, thus, it is possible to break down the total loss by each centroids loss.

$$\sum_{i=1}^{N} L(y_i, NN^*(x_i)) = \sum_{j=1}^{K} L_j \tag{3.5}$$

Where, $L_j = \sum_{x_i \in S_j^{train}} L(y_i, NN^*(x_i))$. Changing $\hat{y}_j$ to label of minority class will increase the error in the j^{th} cluster by at most $|S_j|$. The cost of such action is $Cost_j = \frac{|S_j|}{2}$. Therefore, the change in loss of j^{th} centroid is $\Delta L_j \leq 2 \times Cost_j$. The second constraint of problem (3.4) is satisfied by the adversary. Therefore, using (3.5) and the second constraint of (3.4), we have

$$\Delta \sum_{j=1}^{K} L_j = \sum_{j=1}^{K} \Delta L_j I_j \leq \sum_{j=1}^{K} 2 \times Cost_j I_j < 2 \times Cost. \tag{3.6}$$

□

Computational complexity: The proposed attack technique first trains a SRNN that takes $\mathcal{O}(NDK)$ [115]. Then the attack technique perturbs labels. The perturbation step takes $\mathcal{O}(N)$ which is embarrassingly fast. Compared to other label flipping attacks [84], this attack is computationally very cheap and feasible.

Intuitively, the proposed attack targets the minority groups (smaller clusters). In our experiments, the proposed attack technique achieved significantly higher test error for various machine learning models compared to other label-flipping attacks and no attack. In other words, for an adversary, it would be more efficient in terms of budget to target vulnerable smaller clusters of a dataset and undermine the integrity of any machine learning model significantly.

3.3.3 Defense via Regularized Synthetic Reduced Nearest Neighbor (RSRNN)

We introduce a new parameter named *confidence range* $r_{j_{j=1}}^{K}$, as well as two new regularization terms for SRNN as the defense technique. Any sample beyond the confidence range, $r_{ij^*} > r_{j_i^*}$, is considered to be malicious. Note that j_i^* represents the index of closest centroid to sample i. First term consists of regularizing the confidence range for each centroid. Second regularization term consists of adding cost complexity function over the SRNN structure. The cost function facilitates the pruning of centroids and further recognizes the attacked modalities of the data. The optimization problem of training RSRNN is given in (3.7):

$$\begin{aligned} &\min_{\{(c_j,\hat{y_j},r_j)\}_1^K} \quad \sum_{i=1}^{N} L(y_i, NN(x_i)) + \lambda \sum_{j=1}^{K} r_j + \alpha \sum_{j=1}^{K} cost(S_j) \\ &\text{s.t.} \quad NN(x_i) = \begin{cases} \hat{y_{j_i^*}} & r_{ij^*} < r_{j^*} \\ Malicious & \text{otherwise} \end{cases} \end{aligned} \tag{3.7}$$

where, λ is the penalty coefficient of r_j, α is the cost complexity coefficient, and $cost(.)$ represents the cost function of j^{th} centroid. S_j consists of the samples whose closest centroid from C is j^{th} centroid ($S_j = \{x_i | j = j_i^* \quad \forall i = 1, 2, ..., N\}$). To solve the optimization problem in (3.7), we follow the same EM algorithm as in [115] which was inspired by K-means algorithm [81]. The optimization approach

consists of three steps: the assignment step, the update/centroid step, and the pruning step based on a validation set.

Assignment Step:

This step has two parts. First part consists of assigning the train samples to their closest centroid. This is essentially calculating S_j for $j = 1...K$. Second part is finding optimal values to $\{\hat{y}_j\}_{j=1}^{K}$. The problem for j^{th} centroid can be written as

$$\min_{\hat{y}_j} \quad \sum_{x_i \in S_j} L(y_i, \hat{y}_j) U(r_j - r_{ij}) \tag{3.8}$$

Where $U(.)$ is a step function and is used to impose the constraint in (3.7) for *Malicious* samples. The problem in (3.8) is that of finding the best constant predictor over the set of S_j. Its optimum is the most frequent label of samples in S_j ($\hat{y}_j^* = \text{mode}(\{y_i | \forall x_i \in S_j\})$).

Update step:

This step consists of optimizing each centroid while the centroid labels are kept constant. In this step, first, $\{r_j\}_{j=1}^{K}$ are fixed and $\{c_j\}_{j=1}^{K}$ are optimized, and then $\{c_j\}_{j=1}^{K}$ are fixed and $r_j{}_{j=1}^{K}$ are optimized. It is shown that the problem for optimizing each centroid is a binary classification task. Further, the centroid problem is NP-hard, hence, it will be approximated using a novel surrogate loss function. The optimization problem of this step for j^{th} centroid over c_j is as follows:

$$\begin{aligned} \min_{c_j} \quad & \sum_{x_i \in S_j} (L(y_i, \hat{y}_j) U(r_j - r_{ij}) + U(r_{ij} - r_j)) + \\ & \sum_{x_i \in S_j^c} L(y_i, NN_{C'}(x_i)) \end{aligned} \tag{3.9}$$

where S_j^c represents the complement set of S_j (rest of samples that are not assigned to S_j). $NN_{C'}$ represents nearest neighbor function over the set of centroids without j^{th} centroid ($C' = C - (c_j, \hat{y}_j)$). Here, the assignment of samples are not fixed and a sample might eventually get assigned to S_j or S_j^c depending on the position of c_j in the feature space. This is in fact a binary classification problem because each

sample has to get assigned to S_j or S_j^c. However, prior to the optimization, the optimal assignment of each sample that contributes to decreasing the objective function in (3.9) is not known. From the perspective of EM algorithm, this is a latent variable because it is not clear whether the sample is generated by the distribution of j^{th} centroid or the rest of centroids. The optimal assignment of each sample can be extracted by evaluating the correctness of prediction if the sample assigned to S_j or S_j^c. For example a sample might get classified correctly only if the sample is assigned to S_j because the label of the sample matches the label of c_j. On the other hand, the same sample might be classified incorrectly, or categorized as malicious if assigned to S_j^c. Therefore, this sample has to be assigned to S_j during the optimization to contribute to decreasing the objective function of (3.9) (potentially, c_j has to be closer to the sample than the rest of centroids). From the EM algorithm point of view, the calculation of outcome of assigning a sample to S_j or S_j^c is in fact calculating the posterior probability of the sample being generated by the distribution of $P(y_i|x_i \in S_j)$ or $P(y_i|x_i \in S_j^c)$. In general 8 scenarios can happen for each sample and the optimal assignment of a sample can be found based on these scenarios.

All 8 scenarios are shown in table 3.1. The scenarios are based on three factors. The factors are correctness of classification if the sample is assigned to S_j, correctness of classification if the sample is assigned to S_j^c, and sample categorized as malicious if assigned to S_j^c. Therefore, in total there are 8 scenarios. Note that the case of a sample being recognized as malicious by S_j is not considered here as a factor. This is because during the optimization, the c_j can move and a sample might fall in the r_j ball or may remain outside of it. However, this factor is incorporated inside the surrogate objective function in the following part of this section. In table 3.1, $j_i'^*$ represents the index of closest centroid to sample i from the set C'. From the table 3.1, the samples that belong to scenarios 1-3 cannot affect the objective function of (3.9), since the samples will be classified incorrectly or are considered as malicious regardless of any set they are assigned to. The samples that belong to scenario 4 are the samples that have to be assigned to S_j^c because if they are assigned to S_j then, they are classified incorrectly, hence, they increase

Table 3.1: Update step scenarios. All scenarios for assigning a sample to S_j or S_j^c.

#	$L(y_i, \hat{y}_j)$	$L(y_i, \hat{y_{j_i'^*}})$	$NN_{C'} == Mal.$	ASSIGN
1	1	1	1	x
2	1	1	0	x
3	1	0	1	x
4	1	0	0	S_j^{c*}
5	0	1	1	S_j^*
6	0	1	0	S_j^*
7	0	0	1	S_j^*
8	0	0	0	x

the objective function of (3.9). We call such set as S_j^{c*}. Samples of scenarios 5-7 have to be assigned to S_j to decrease the objective function of (3.9). We denote such set of samples as S_j^*. Samples of scenario 8 are always classified correctly; thus, they are ineffective in the objective function of (3.9). Intuitively, c_j has to be replaced in the space such that c_j is closest centroid to samples of S_j^*. Also, preferably, c_j should be at a distance of r_j from all samples of S_j^*. At the same time c_j should remain further away from samples of S_j^{c*} with at least a distance of $r_{ij_i'^*}$. Using sets of S_j^* and S_j^{c*} and the given intuition in previous paragraph, the problem in (3.9) can be rewritten as follows

$$\begin{aligned} \min_{c_j} \quad & \sum_{x_i \in S_j^*} (U(r_{ij} - r_{ij_i'^*}) \vee U(r_{ij} - r_j)) + \\ & \sum_{x_i \in S_j^{c*}} U(r_{ij_i'^*} - r_{ij}) \end{aligned} \tag{3.10}$$

where $\vee$ is logical OR operator. Finding the global optimum of (3.10) is NP-hard and cannot be solved directly [14, 86]. Therefore, a surrogate objective function will be used to approximate the solution to (3.10). Intuitively, the interest of problem (3.10) is to replace the c_j close to the samples of S_j^*. At the same, time we are interested in nullifying the effect of possible $Malicious$ samples of S_j^* that are outside of the r_j automatically. Further, c_j has to stay outside the ball of $r_{ij_i'^*}$ for the samples in S_j^{c*}. Based on the given intuition, the solution to problem (3.10)

is approximated by solving

$$c_j^*(\mu) = \underset{c_j}{\text{argmin}} \quad \sum_{x_i \in S_j^*} min(r_{ij}, r_j) + \sum_{x_i \in S_j^{c*}} relu(\mu r_{ij_i'^*} - r_{ij}) \tag{3.11}$$

where $min(.)$ returns the minimum of its input arguments. $relu(.)$ is a rectified linear unit. μ is a coefficient and acts like a slack variable [115] that is increased from 0 to 1. For every value of μ along the path, the problem of (3.11) can be solved using stochastic gradient descent. Along this path, the c_j^* that returns the smallest loss for (3.10) is selected. This is similar to solving an SVM for a linear binary classifier [115] while changing the slack variable.

Finally, $\{r_j\}_{j=1}^K$ have to be optimized while fixing the rest of parameters. The problem of optimizing r_j for a centroid is

$$\min_{r_j} \quad \sum_{x_i \in S_j} L(y_i, \hat{y}_j) U(r_j - r_{ij}) + U(r_{ij} - r_j) + \lambda r_j \tag{3.12}$$

From problem (3.12), it can be observed that objective function is piecewise-constant over r_j and objective function has a jump at every r_{ij}. This problem can be solved efficiently in $\mathcal{O}(|S_j| log(|S_j|))$. This is done by sorting r_{ij} for every $x_i \in S_j$ and evaluating the objective function of (3.12) for every $r_j = r_{ij}$ through an incremental algorithm. In total, finding optimum r_j for all centroids is $\mathcal{O}(Nlog(N))$.

Excluding malicious samples on the fly: Note that the objective function in (3.11) has two terms that contain c_j. The first term is $min(r_{ij}, r_j)$ that encourages samples of set S_j^* to be close to c_j but any sample that falls outside of the ball of r_j will become ineffective in optimization of (3.11) because the gradient with respect to that sample becomes 0. This is interesting since the optimization algorithm is in fact excluding samples that are suspicious to be malicious on-the-fly. Further, other malicious samples such as scenarios 1 and 3 are automatically not considered in the optimization.

In total, computational complexity of this step is $\mathcal{O}(NDK + NlogN)$ [115].

Finally, by iterating over update step and assignment step, the first two terms

of the objective function in (3.7) decrease over the trainset until no further improvement can happen over the parameters.

It is noteworthy that the surrogate objective function for update step also decreases the $\{cost(S_j)\}_{j=1}^{K}$ since it tends to create pure sets for each S_j.

Pruning step:

After optimizing the first two terms of (3.7), the third term needs further attention. Similar to pruning for decision trees, the cost complexity function encourages to remove (prune) the impure centroids based on their assignment set S_j and the coefficient of α. In this chapter, Gini-index is used as the cost function. However, the dataset's integrity can be under question, meaning that some of the samples/modes are malicious. Here, the intuition and aim of this step is not only to prune the centroids but also remove the malicious modalities of the trainset. Also note that any malicious sample is considered as a loss, thus, only centroids and modes are considered as malicious if removal of them decreases the error over the validation set. Therefore, for this step, all other parameters including assignment of samples are kept fixed. Further, a clean validation set is used to prune centroids and samples of malicious modes. The clean validation set is a set whose integrity is assured and in practice a tiny set for validation set such as $5-10$ percent of the whole trainset would be sufficient.

How to prune? After optimizing (3.7) using the assignment step, pruning is performed. The pruning step consists of removing malicious centroids and samples. To do so, other parameters are kept constant, a validation set is needed and optimization over α must be performed since α is a hyperparameter. A direct selection of α is not practical since α can accept any values from 0 to ∞. Thus, we will show that Values of α that cause change to the structure of RSRNN are quantized and correspond to a specific cut-off threshold over $cost(S_j)$. $cost(S_j)$ is a real number in $[0, 1]$. Therefore, a range of numbers between $[0, 1]$ are used to remove centroids based on the trainset and evaluate their performance over the validationset. Any cut-off that had smallest validation error was selected.

For each centroid, the cut-off threshold is the threshold that by removing the centroid, the objective function of (3.7) decreases. Such threshold can be calculated as follows:

$$cost(S_j) = \frac{L(y_i, NN_{C-c_j}(x_i)) + \lambda r_j}{\alpha} \tag{3.13}$$

Where, $NN_{C-c_j}(.)$ is the nearest neighbor function over all centroids except j^{th} centroid.

In practice, we used cut-off thresholds between $[0.2 - 0.9]$ with steps of 0.05. For each cut-off, the centroid and its sampleset (S_j) was removed. Further, for each cut-off a retraining consisting of only initialization with K centroids over the cleaned dataset was applied and its validation error was evaluated. Please note that the purpose of this pruning approach is detecting malicious samples along with removing malicious centroids. Finally, the model with smallest validation error was selected for further optimization over the cleaned dataset.

Finally, after removing the malicious samples and centroids, it is possible to either restart training using original SRNN over the cleaned data, or continue training with the remaining centroids and select the final model based on the error over the validation set.

3.4 Computational Complexity and Convergence

The attack technique consist of first training SRNN model that takes $\mathcal{O}(NDK)$ [115] and then perturbing labels of samples. The second step consists of calculating cost of each cluster (takes $\mathcal{O}(|S_j|)$) and sorting the costs of clusters (takes $\mathcal{O}(KlogK)$). Therefore, in total, the attack technique takes $\mathcal{O}(NDK + N + KlogK)$.

Theorem 4 (Convergence of RSRNN)**.** *Iterating over the first two steps of optimizing RSRNN converge to a local minimum of the first two terms of objective function in* (3.7)*. This takes a finite number of iterations.*

Proof. The objective function has a lower bound of zero. Both the assignment and the centroid steps decrease or do not change the first two terms of (3.7). Different

combinations of assignments of the samples to the centroids are finite. As a result, the loss function does not decrease after a finite number of iterations over both steps. This proof is similar to proof of convergence for K-means [81]. □

3.5 Experimental Results

In this section, experimental results of the proposed attack and defense techniques are presented and compared with similar techniques. The attack techniques which are applicable to various machine learning models are used for comparison in this study. In terms of defense technique, models that are known to be resilient against the attack techniques are used for comparison. Further, the merits of the proposed attack and defense techniques are studied and presented. In the next subsection, the effect of proposed attack is explored and in the second subsection, the performance of the RSRNN is compared with several other resilient models.

3.5.1 Attack Experiments

3.5.2 Attack Experiments

In this subsection, the performance of the modality-based adversarial label filliping (SRNN-att) is presented. The SRNN-att is compared with several other label filliping attacks that are generic and can affect all machine learning models. These label filliping techniques are gathered from the literature of label noise because they are found to be the most similar techniques that exist on the topic of this chapter, as they are not designed for a specific machine learning model and can potentially deteriorate the performance of any machine learning model.

Accordingly, in our experiments, SRNN-att is compared with NCAR, NNAR and no attack. NCAR works by changing the labels at random. NNAR is a practical approach that aims at changing labels at the margins of decision boundaries. Here, to apply this type of attack, a state of the art model is trained and then labels of the samples that are predicted with low confidence by the model are targeted. Labels of such samples are changed to the second most probable label,

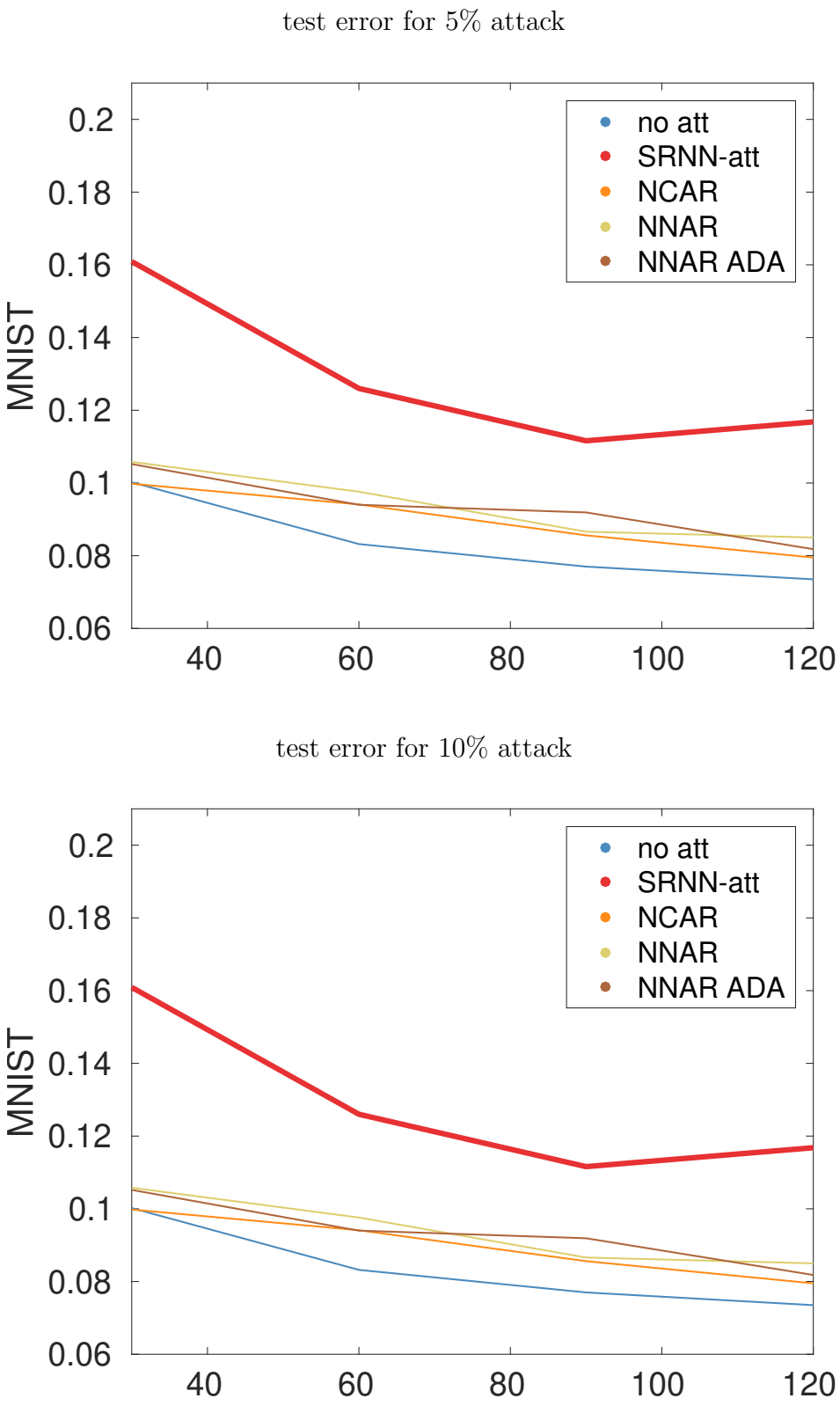


Figure 3.1: Different attack techniques are presented for MMNIST dataset. Vertical axis presents error ratio over the testset and horizontal axis presents number of base models.

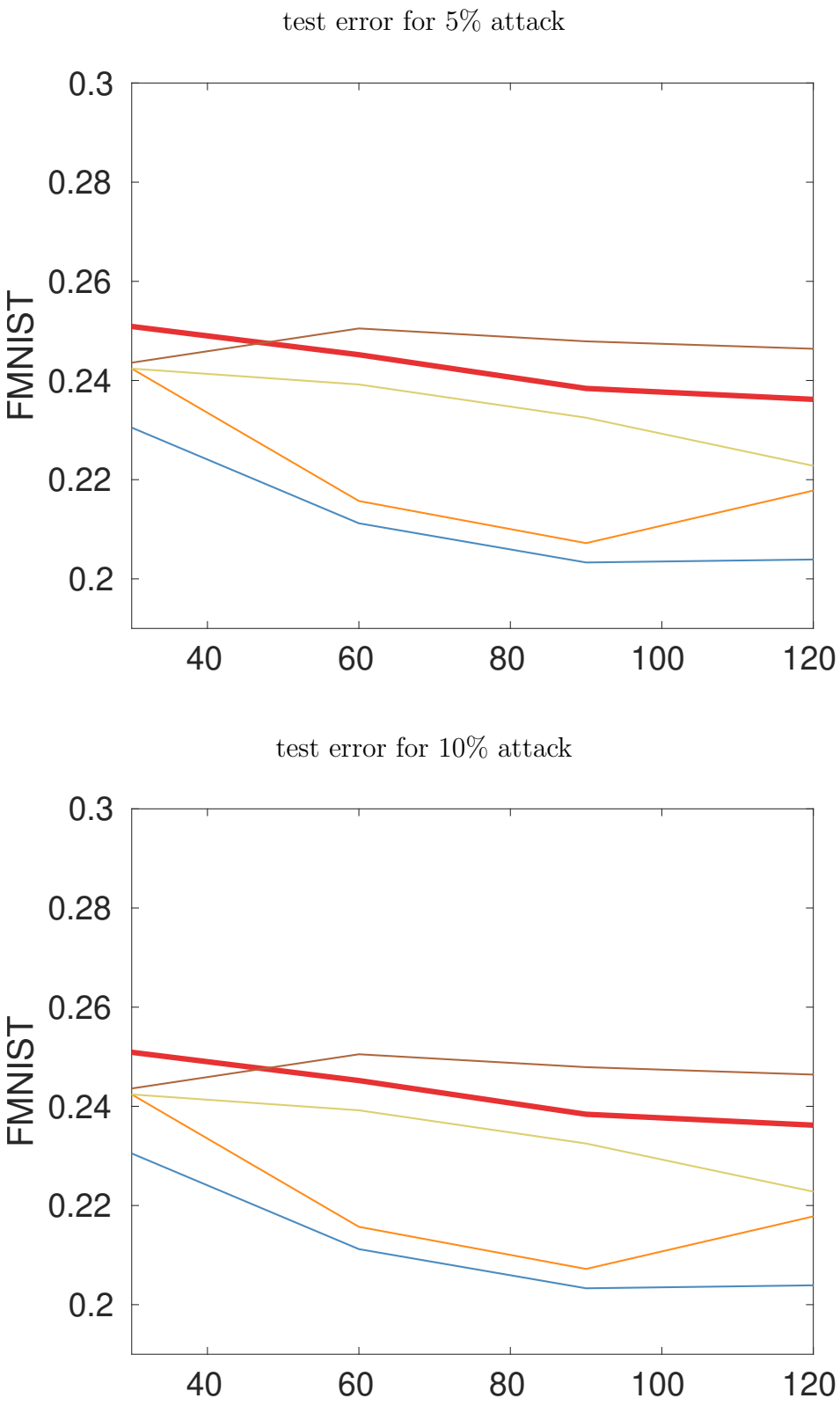


Figure 3.2: Different attack techniques are presented for FMNIST dataset. Vertical axis presents error ratio over the testset and horizontal axis presents number of base models.

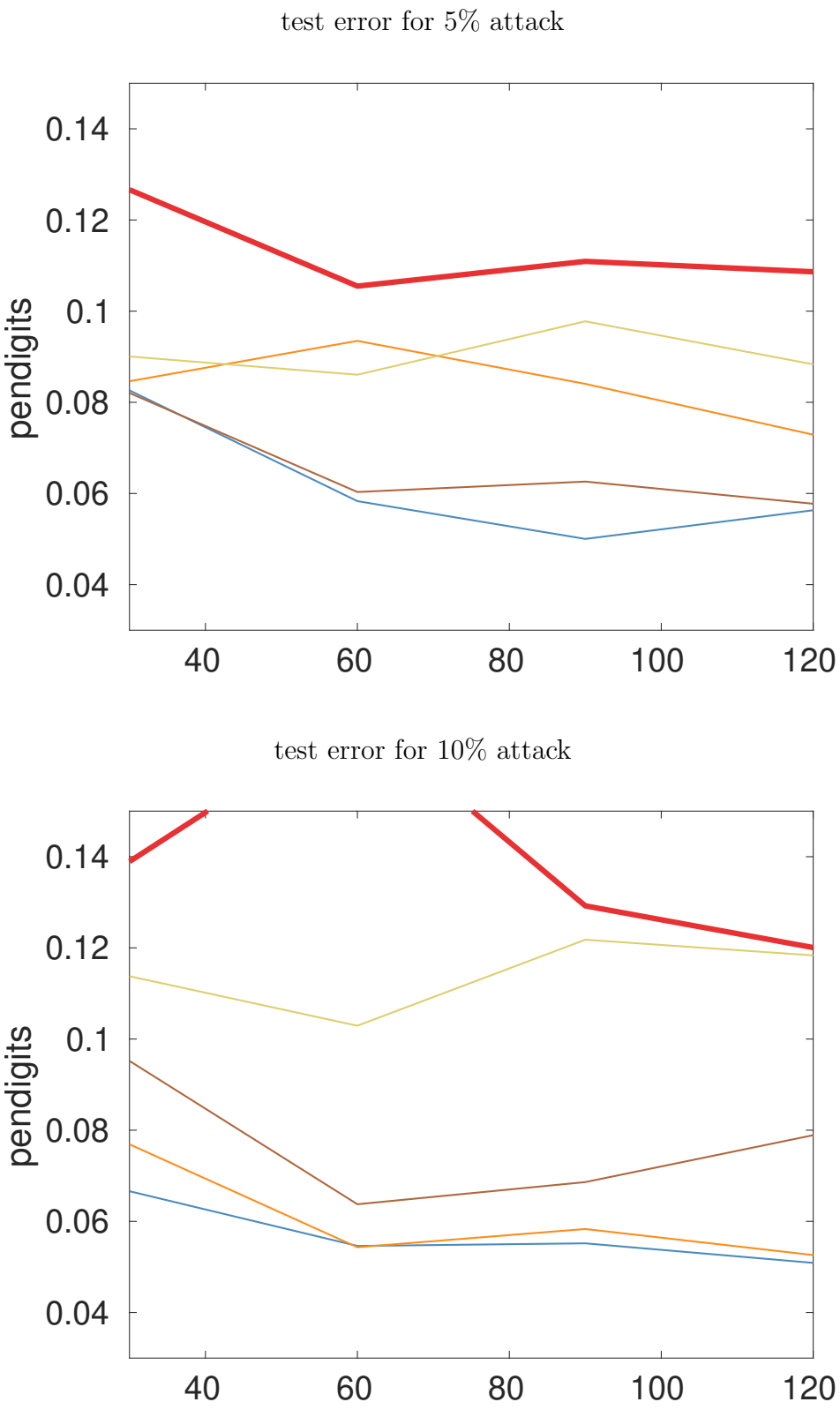


Figure 3.3: Different attack techniques are presented for pendigits dataset. Vertical axis presents error ratio over the testset and horizontal axis presents number of base models.

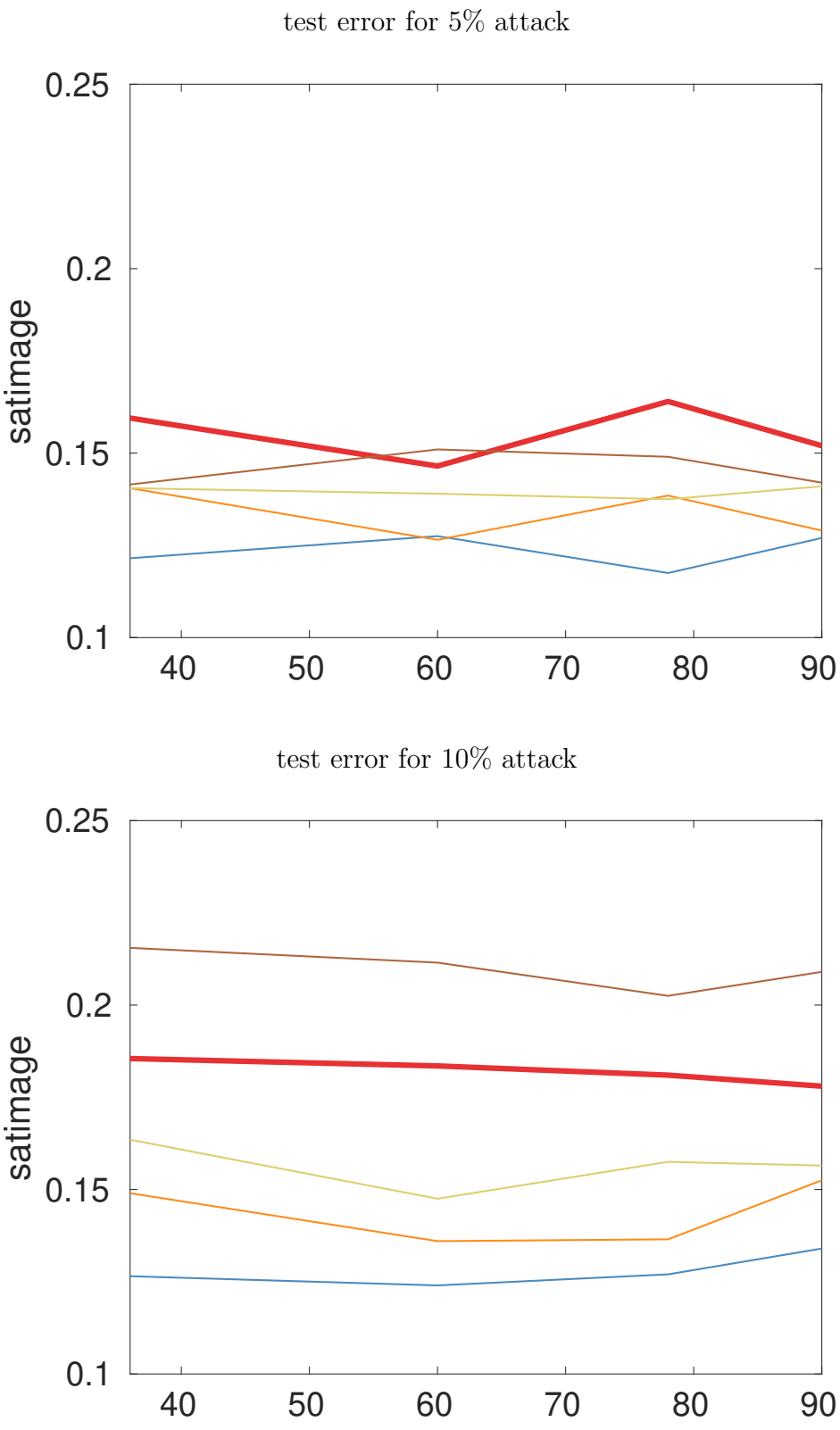


Figure 3.4: Different attack techniques are presented for satimage dataset. Vertical axis presents error ratio over the testset and horizontal axis presents number of base models.

thus, intuitively, changing the decision boundaries. In the experiments, NNAR and NNAR-ADA represent the NNAR attack using a forest model and AdaBoost model, respectively. The size of these models is selected such that they have the same size as that of the SRNN model used for SRNN-att.

In order to evaluate performance of the attack techniques, several machine learning models using the poisoned dataset are trained. In this chapter, SRNN, RBF-SVM, OC1, CART, K-means, AdaBoost, Nearest Neighbor, Random Nearest Neighbor are used. The details of these models are explained in the further experiments section.

In order to obtain a fair comparison, for each dataset, the lowest error rate of each attack model among all the trained machine learning models is presented in figures 3.1,3.2, 3.3 and 3.4. In figures 3.1,3.2, 3.3 and 3.4, vertical axis shows error ratio over test set and horizontal axis represents number of centroids, leaf nodes, RBFs and trees for SRNN, tree, RBF-SVM and forest models, respectively. In fact, all the models are presented as a kind of ensemble of smaller models, thus, it is possible to compare them based on the number of terms they contain. Figures 3.1,3.2, 3.3 and 3.4 show that in all attack techniques, the SRNN-att consistently achieved the highest test error with a significant margin, or performed at least as good as the next best attack when compared with other attack techniques. Additionally, it is noteworthy that in a few instances one of the other techniques was able to increase the test error significantly but none of the other techniques was able to consistently achieve a high rise in the test error.

In figure 3.5, results for best performing models are shown for the MNIST dataset. As can be observed, SRNN-att was able to make the models underperform more than other attacks. Complete set of experiments are presented in the section 3.6.

3.5.3 Defense Experiments

For the evaluation of the proposed defense against the proposed attack technique, the proposed RSRNN approach is compared with the other state of the art models that are known to be resilient against label flipping issues. Other models

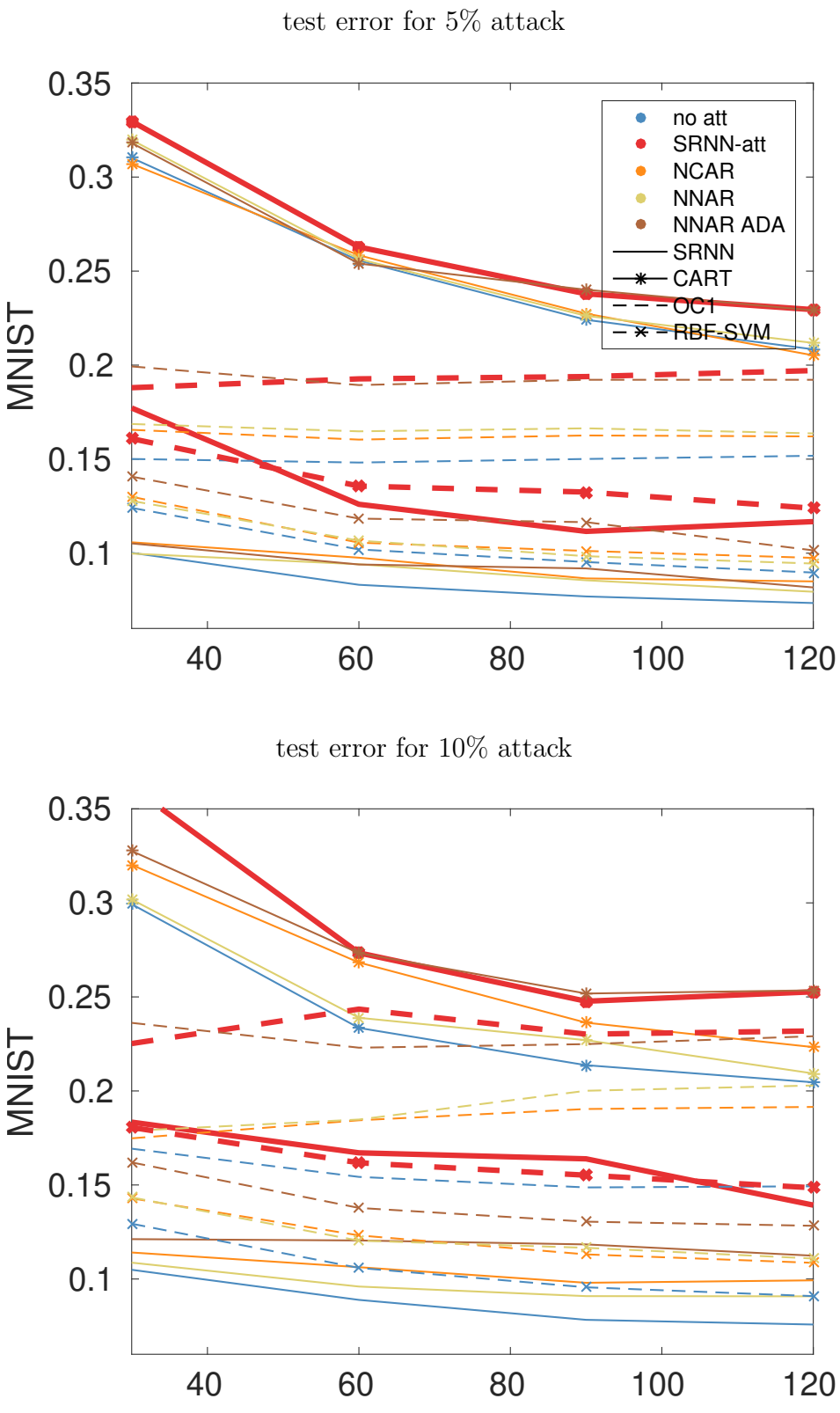


Figure 3.5: Different attack techniques and models are presented for MNIST dataset. Colors and curve shapes present the attack techniques and models, respectively. model Vertical axis presents error ratio over the testset and horizontal axis presents number of base models.

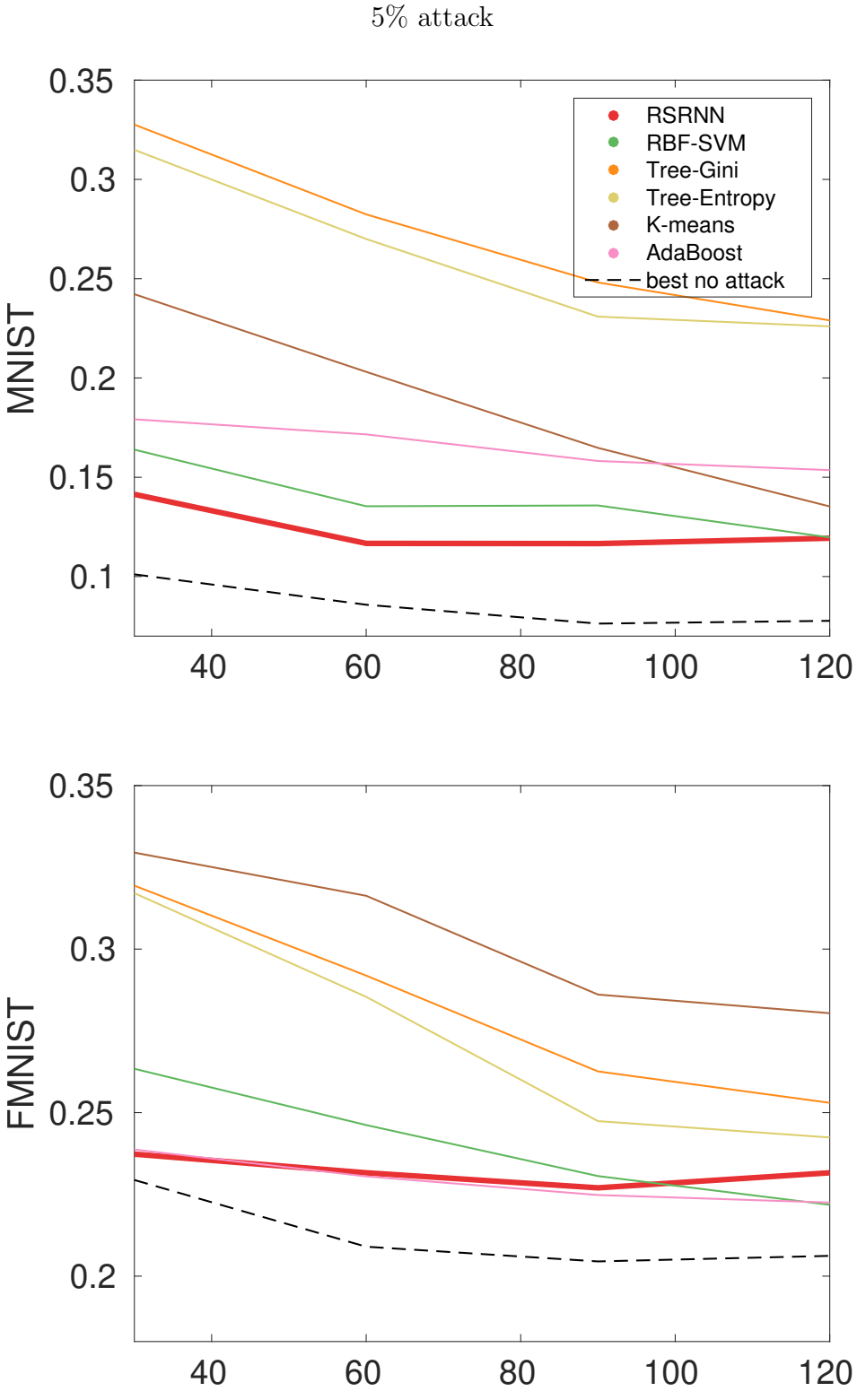


Figure 3.6: Different defenses against SRNN-att. SRNN-att model was trained with 30 centroids. Vertical axis shows test error ratio and horizontal axis represents number of basis models.

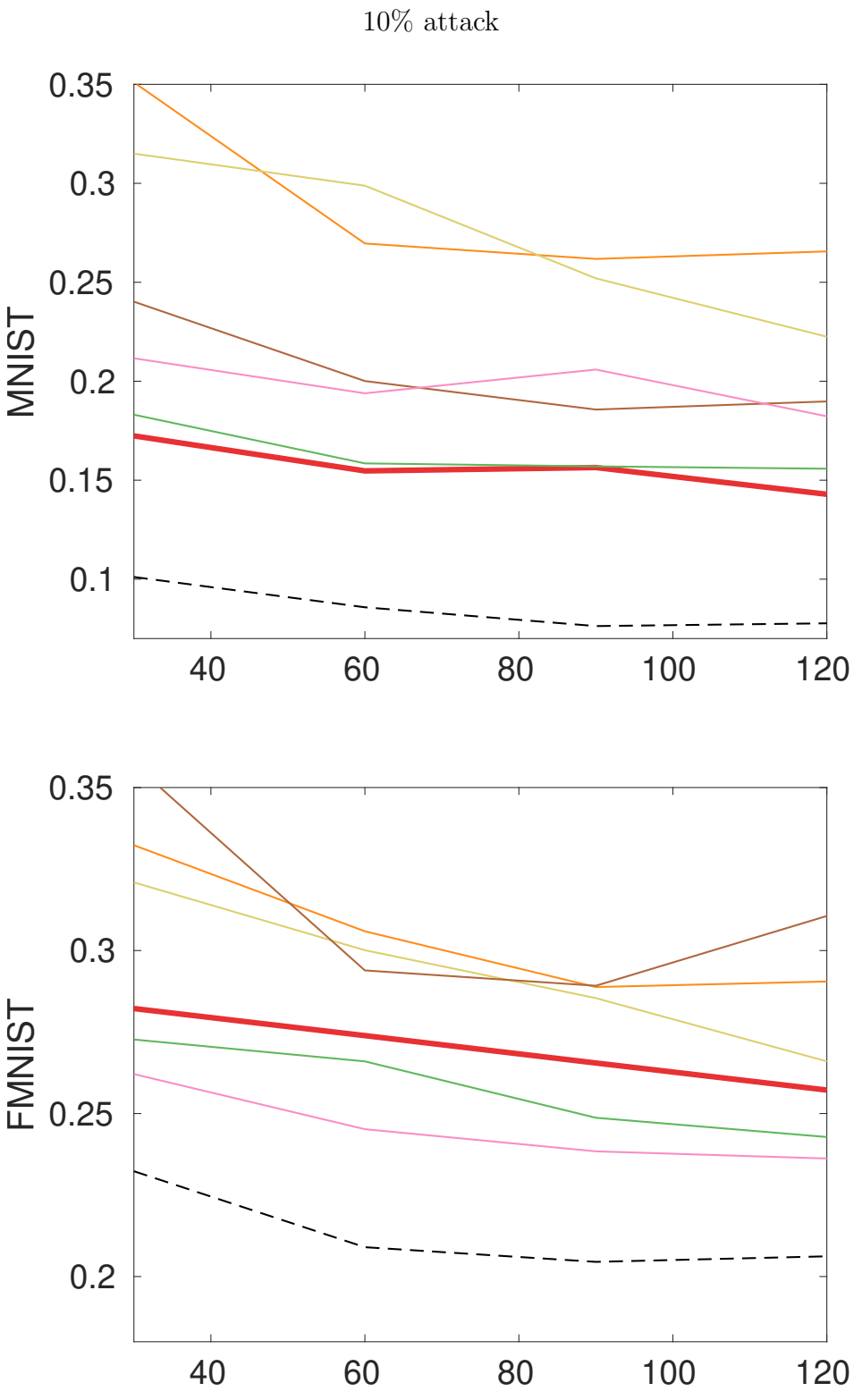


Figure 3.7: Different defenses against SRNN-att. SRNN-att model was trained with 30 centroids. Vertical axis shows test error ratio and horizontal axis represents number of basis models.

consist of trees, RBF-SVM, AdaBoost, K-means. For trees, tree with two split criteria of Gini index and Cross-entropy were used because they are known to be resilient against noisy or adversarial samples [1]. Another approach against adversarial/noisy label flipping attack techniques is using validationset for model selection [93, 42]. This validation set is used for selecting the best model that has smallest error over the validation set[55]. The validation set is used for selecting parameters of K-means and RBF-SVM in the experiments of this subsection, thus, making the models more resilient against the label noise. Finally, it is also known that ensemble models such as boosting algorithms are more robust against noise in labels [42]. Therefore, AdaBoost is added to the experiments of this subsection. The details of training each model is presented in the section 3.6.

Figures 3.6, 3.7, 3.8 and 3.9 present the results of experiments in this subsection. As can be observed from figures 3.6, 3.7, 3.8 and 3.9, RSRNN was able to constantly outperform other models with a large margin. At the same time, RSRNN was able to improve the results of SRNN by up to $2-3\%$. Additionally, RSRNN was able to detect a large portion of malicious samples up to 70% with a true positive of $50-60\%$. Experiments regarding performance of RSRNN in detecting malicious samples is presented in the section 3.6.

Finally, the size of validation set used in the experiments of this section is only 8% of the trainset. Further details of the experiment setup and more experimental results on the efficiency of the model under various configurations of the attack are presented in the 3.6.

3.6 Further Experimental Results of RSRNN

Dataset setups: We performed the experiments over two setups for each dataset. In one setup, 20% of the dataset for selected as trainset and a validation set with size of 8% of the trainset was selected. The validationset contains the original(unperturbed) labels for the samples. The trainset was poisoned using SRNN-att. The SRNN-att was trained over 80% of the dataset for all setups. The reason for such setup is that in the proposed method the attacker was supposed to have

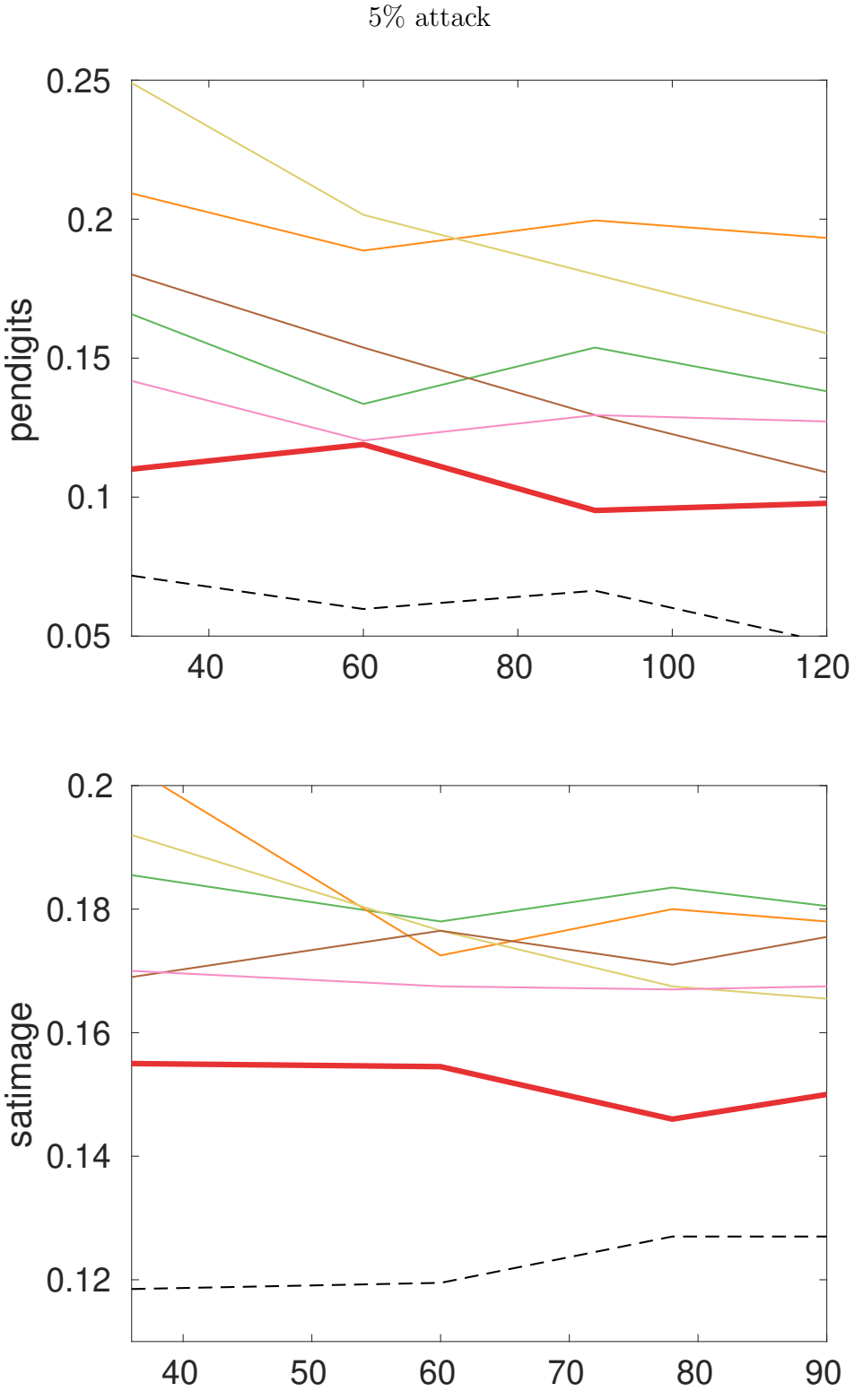


Figure 3.8: Different defenses against SRNN-att. SRNN-att model was trained with 30 centroids. Vertical axis shows test error ratio and horizontal axis represents number of basis models.

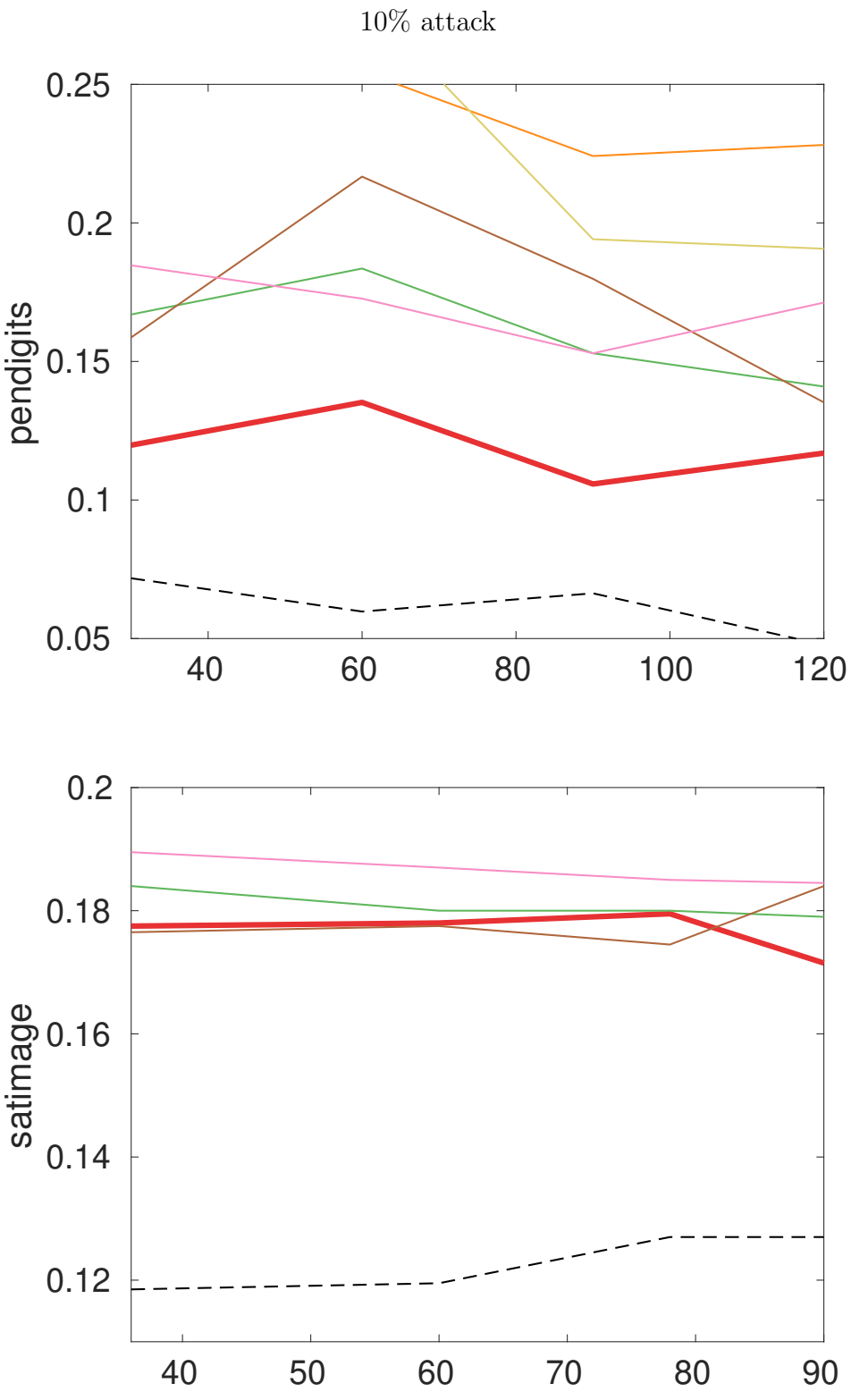
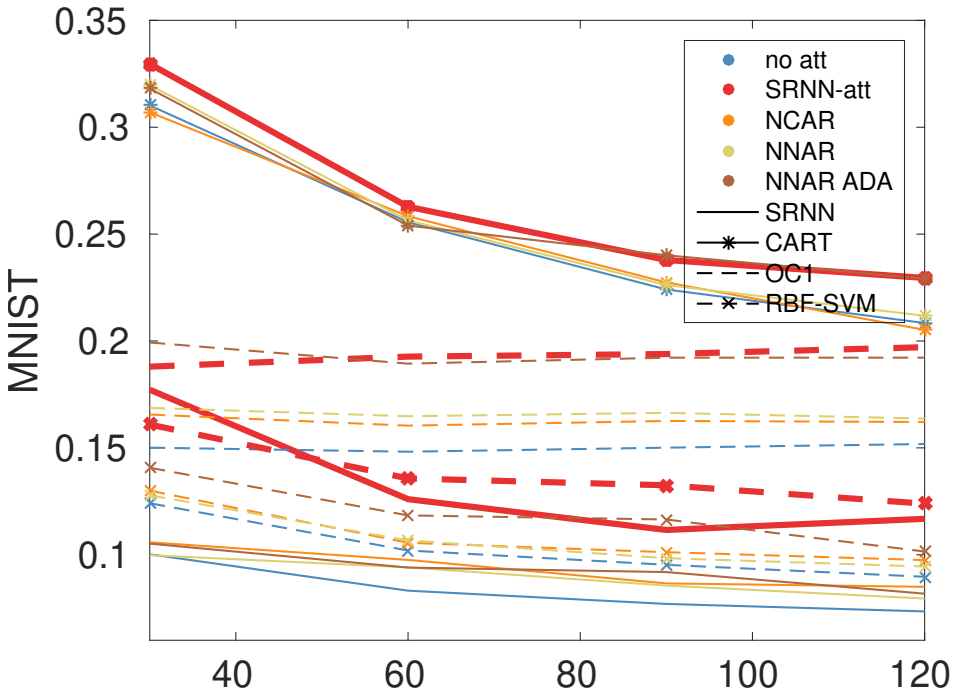


Figure 3.9: Different defenses against SRNN-att. SRNN-att model was trained with 30 centroids. Vertical axis shows test error ratio and horizontal axis represents number of basis models.

the knowledge of the optimal centroids. Therefore, to mimic such assumption, the attacker was trained over a larger set of samples. The second setup, 80% of the dataset was used as trainset and a validationset with the size of 8% of the trainset was selected. The trainset was poisoned using same SRNN-att as the first setup. The second setup is used for all datasets. In the experiments of the previous section, the first setup was used for datasets of MNIST, FMNIST and pendigits. For satimage the second setup was used since using the first setup was causing overfitting in all trained models. The experiments over MNIST, FMNIST and pendigits using the second setup are presented in this section. Finally, we noticed that SRNN-att can increase the error significantly even if it is trained over the same portion of samples as the other models.

Models setups: In the experiments of the previous section, various models were used. Each model can be represented as a kind of ensemble model that consist of several base models. Here, the setups for each model is presented. For decision tree models of OC1 [87] and CART [21] the number of leaf nodes represents the number of base models. For RBF-SVM, first several K centroids using K-means over the trainset was found and used as centers for RBFs. Each centroid represents one base model. The hyperparameters of RBFs were found using the trainset for the attack experiments. In the defense experiments, the RBF hyperparameters were selected using the clean validationset. For the K-means model, a simple unsupervised K-means was trained over the features of the trainset and then the labels were selected using the assignment step. The labels for attack and defense experiments were selected using the trainset and validationset, respectively. The number of centroids represents the number of base models. For AdaBoost, each base model was a tree of depth 4 and the model was trained such that it contained similar number of parameters as RBF-SVM. SRNN was trained using same number of centroids as the other models. In random Nearest Neighbors, K random samples were selected as a the model. In the experiments, we have select models that are similar in the sense that each model would partition the input space or uses centroids as basis of the model. Neural Networks do not fall into any of the categories mentioned here.

5% attack



10% attack

Figure 3.10: Different attack techniques and models are presented for MNIST dataset. Colors and curve shapes present the attack techniques and models, respectively. model Vertical axis presents error ratio over the testset and horizontal axis presents number of base models. For this figure, first setup was used.

3.6.1 Attack Experiments

In this subsection, the performance of all models over each dataset is presented in figures 3.10, 3.11, and 3.12 for the first setup.

As can be observed from figures 3.10, 3.11, and 3.12, SRNN-att was able to increase the error of all models higher than other attack techniques.

Figures 3.13, 3.14, and 3.15, present the best of all models under each attack for second setup.

Figures 3.16, 3.17,and 3.18 present all models for second setup.

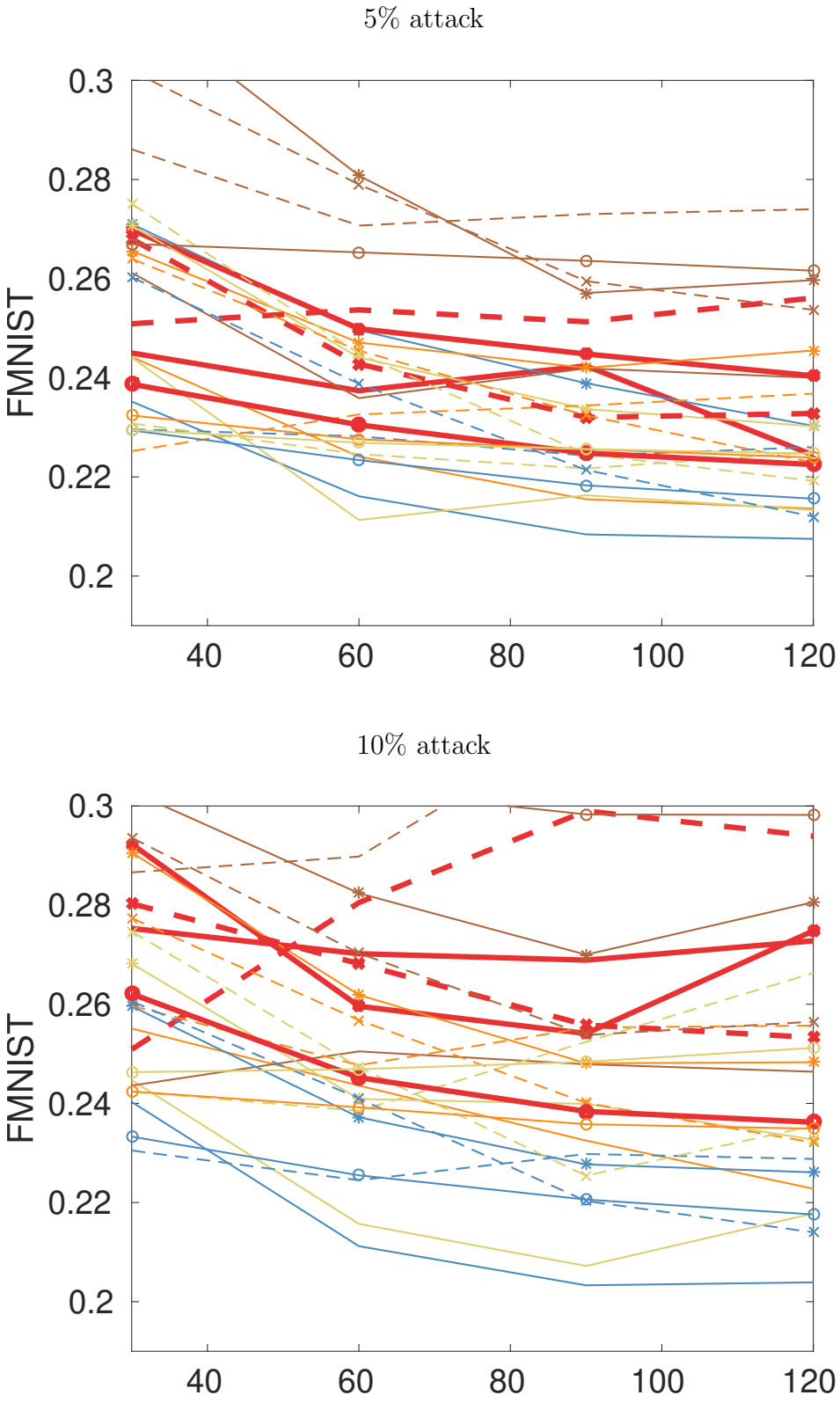


Figure 3.11: Different attack techniques and models are presented for FMNIST dataset. Colors and curve shapes present the attack techniques and models, respectively. model Vertical axis presents error ratio over the testset and horizontal axis presents number of base models. For this figure, first setup was used.

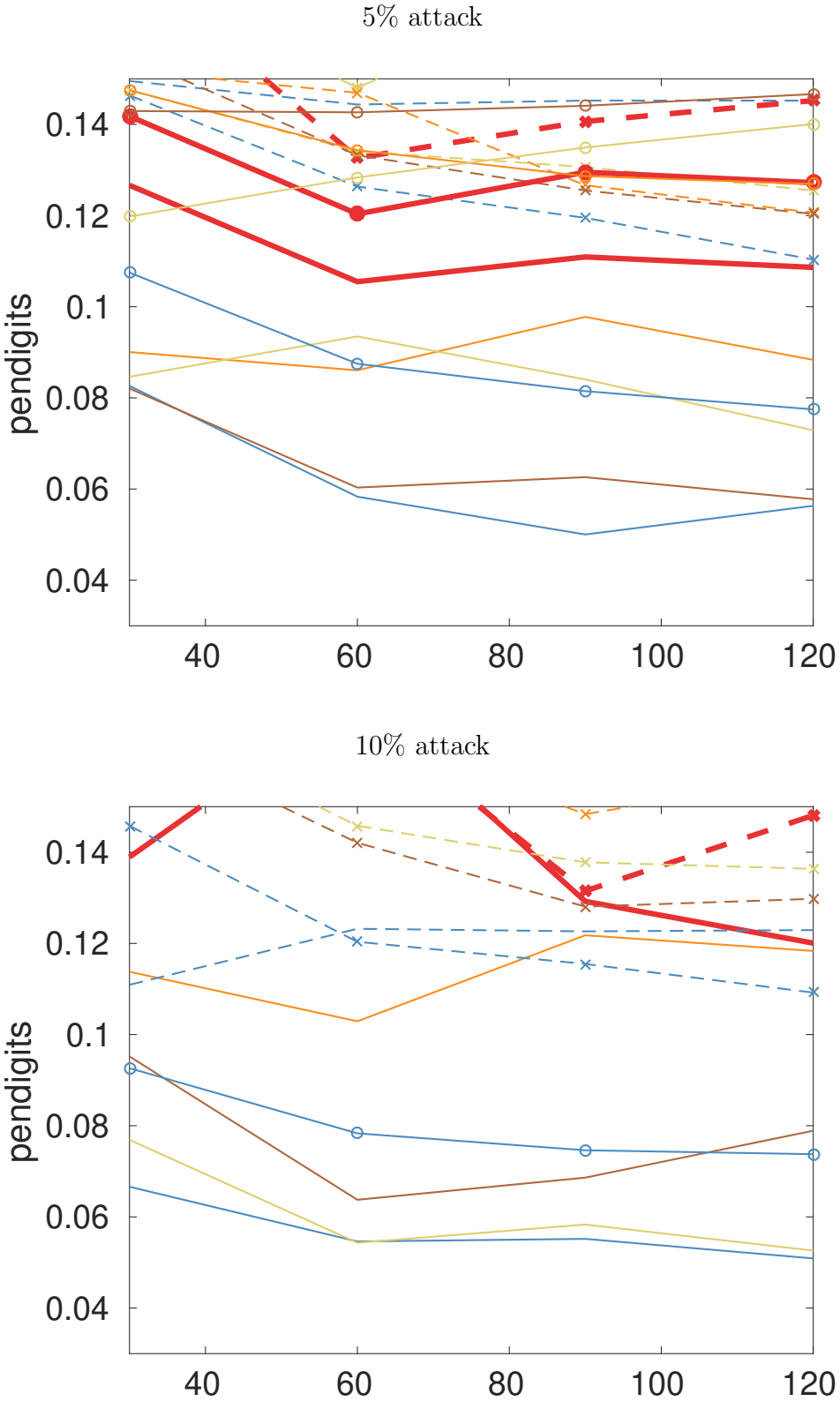


Figure 3.12: Different attack techniques and models are presented for pendigits dataset. Colors and curve shapes present the attack techniques and models, respectively. model Vertical axis presents error ratio over the testset and horizontal axis presents number of base models. For this figure, first setup was used.

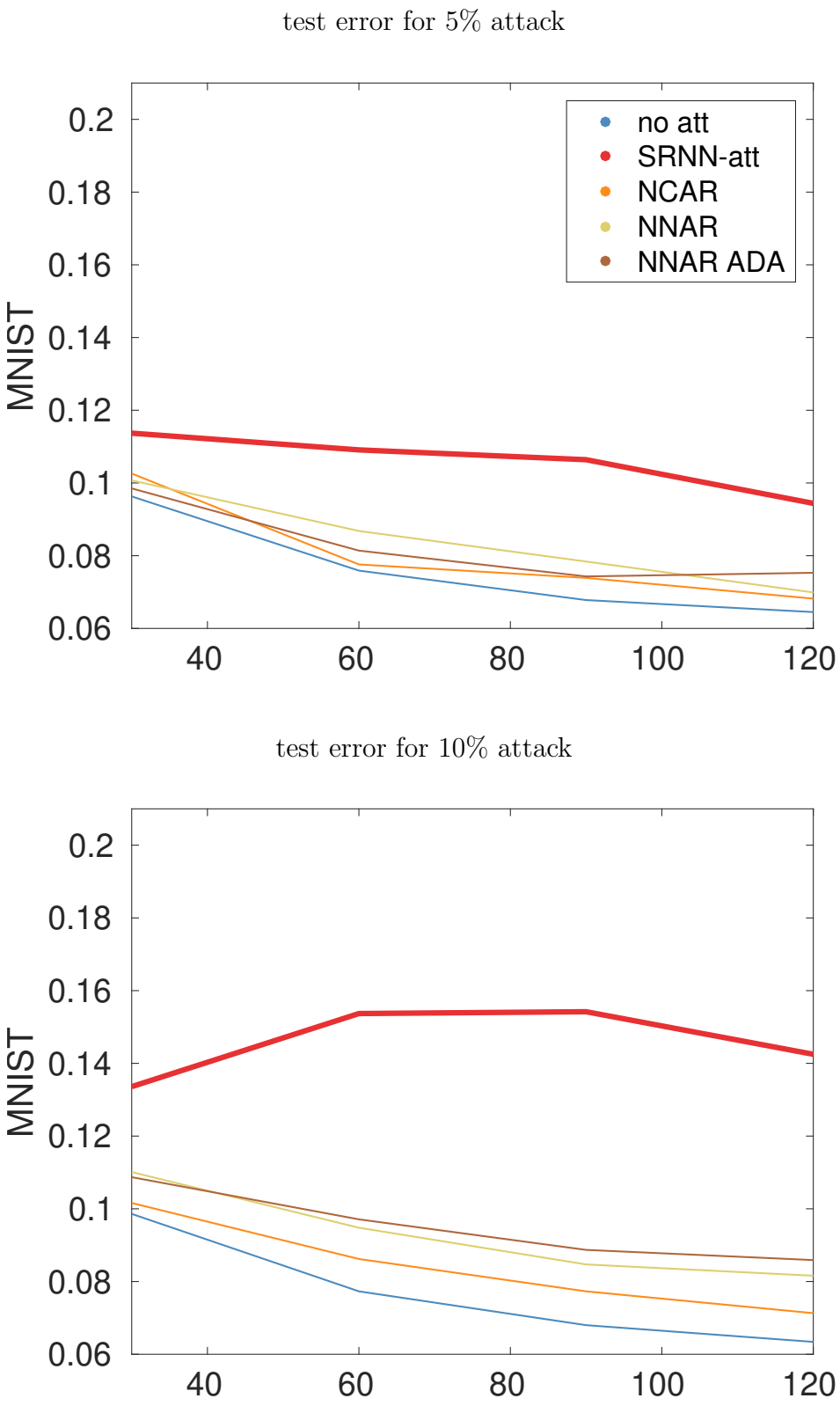


Figure 3.13: Different attack techniques are presented for MNIST dataset. Vertical axis presents error ratio over the testset and horizontal axis presents number of base models. The second setup was used for experiments.

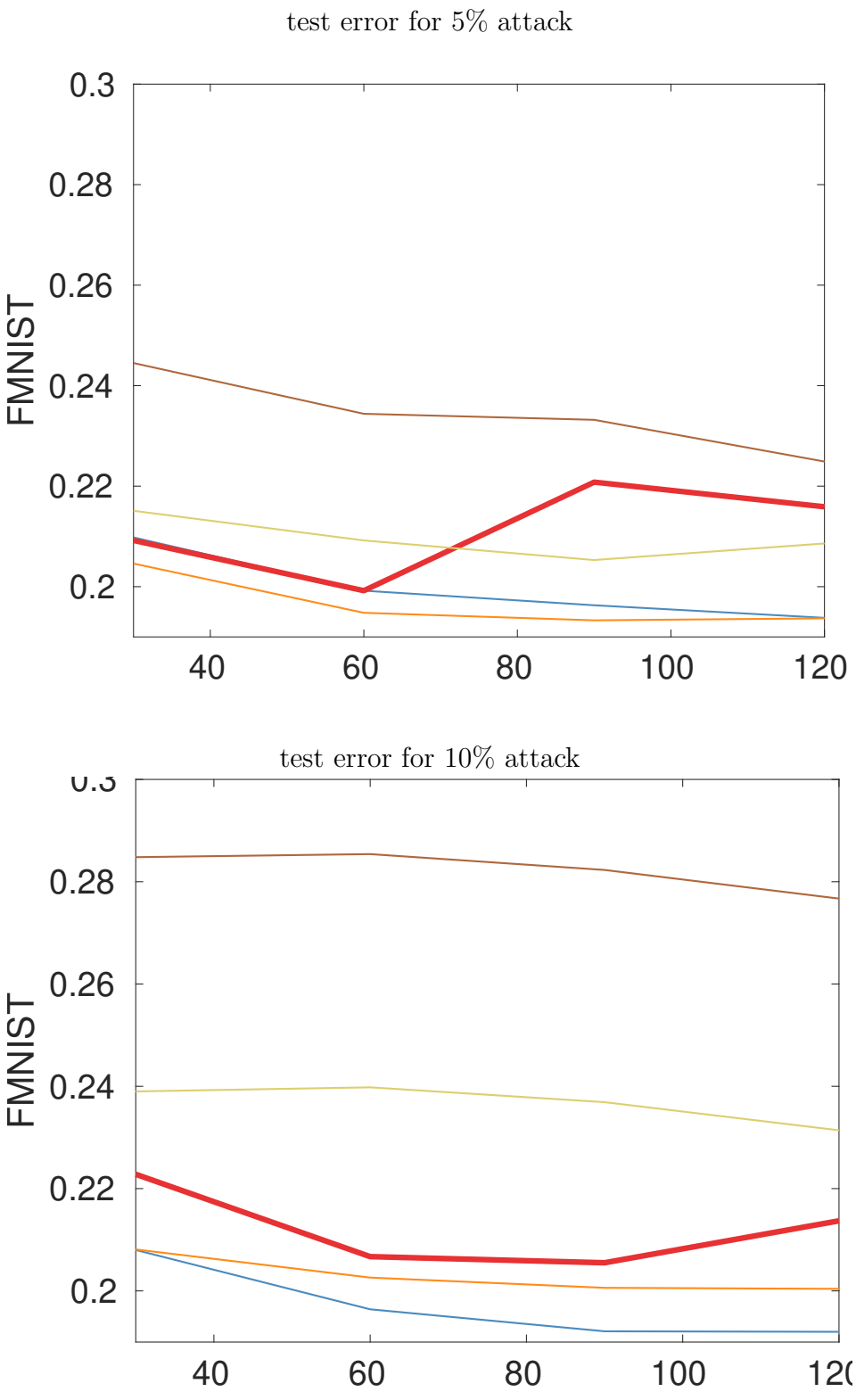


Figure 3.14: Different attack techniques are presented for FMNIST dataset. Vertical axis presents error ratio over the testset and horizontal axis presents number of base models. The second setup was used for experiments.

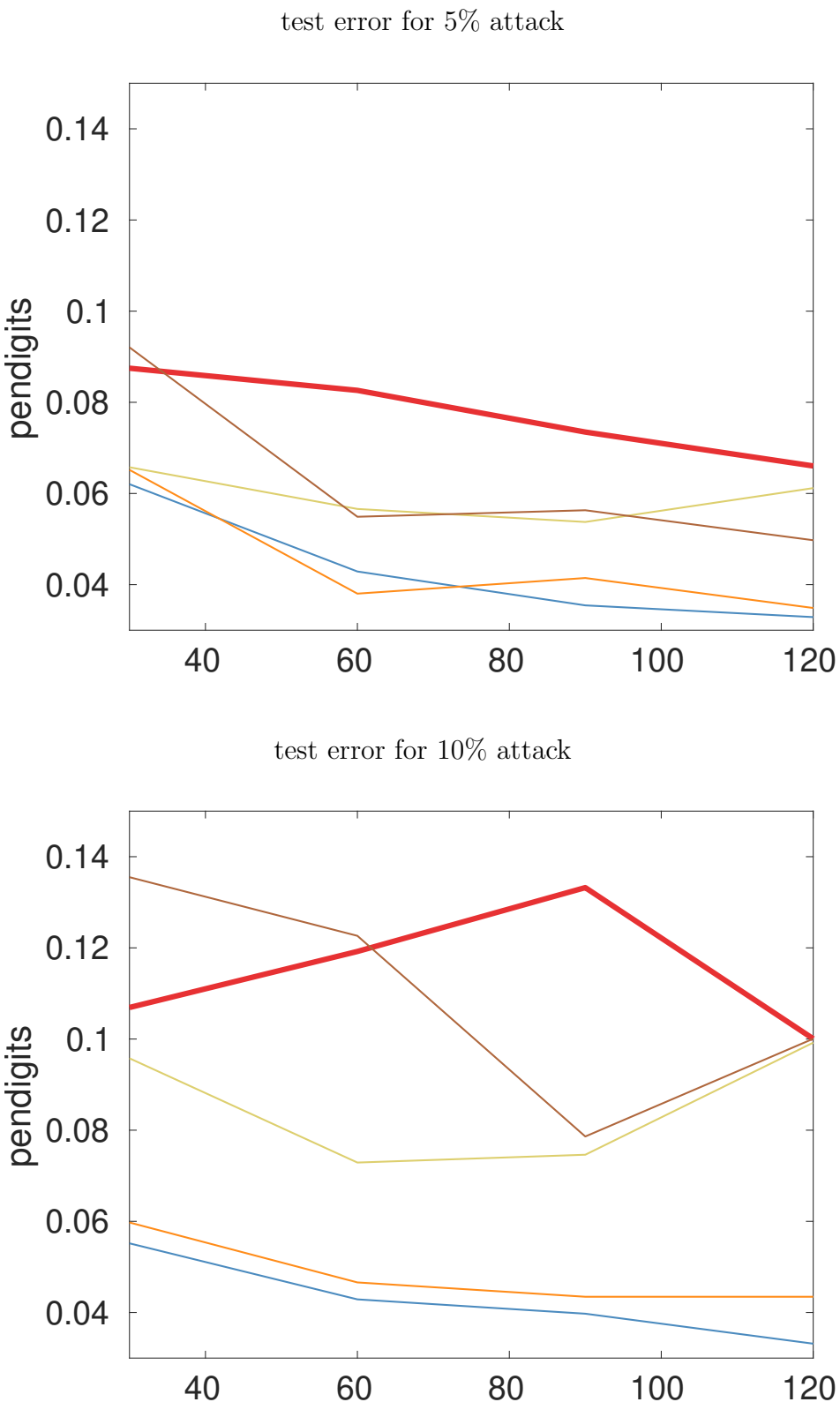


Figure 3.15: Different attack techniques are presented for pendigits dataset. Vertical axis presents error ratio over the testset and horizontal axis presents number of base models. The second setup was used for experiments.

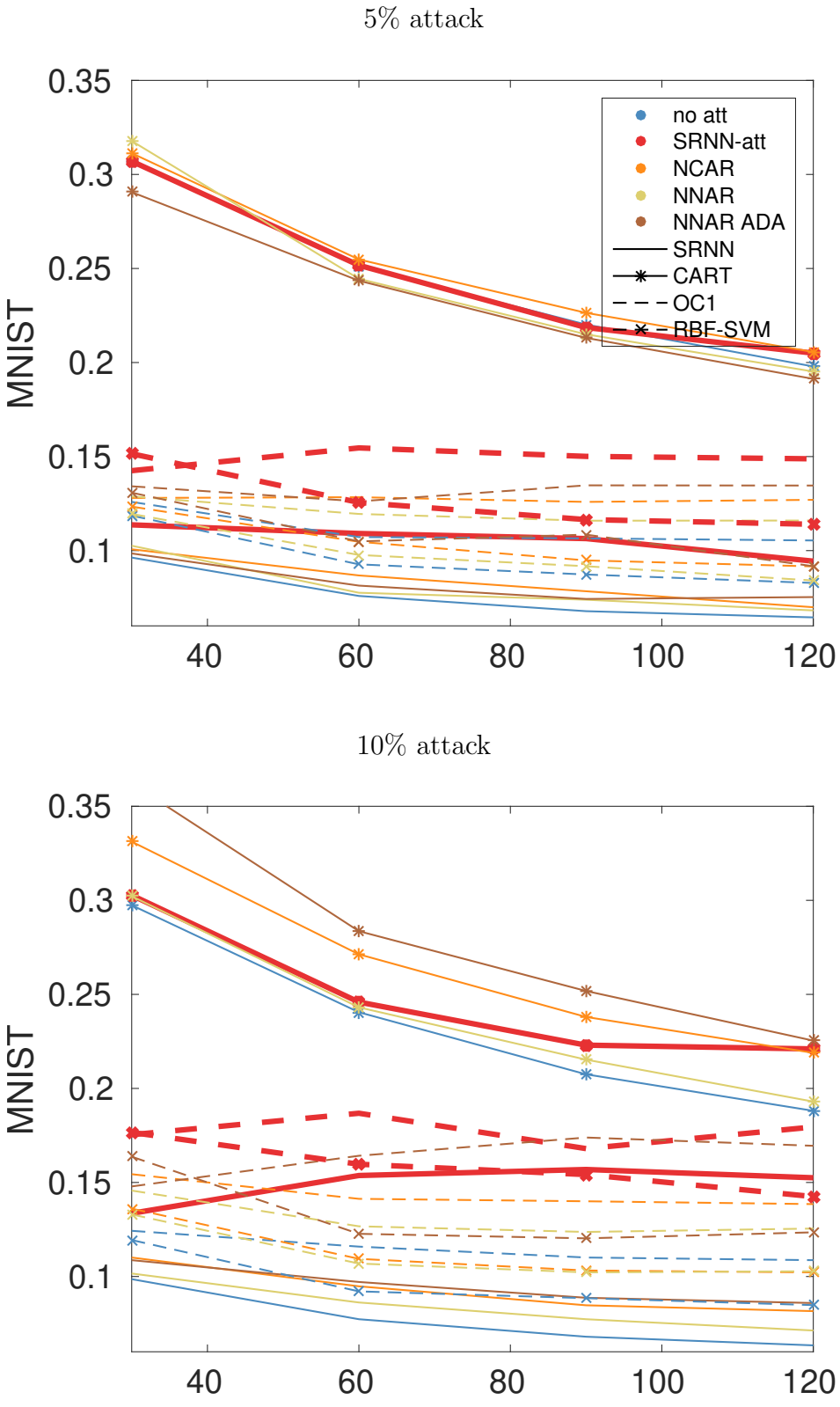


Figure 3.16: Different attack techniques and models are presented for MNIST dataset. Colors and curve shapes present the attack techniques and models, respectively. model Vertical axis presents error ratio over the testset and horizontal axis presents number of base models. For this figure, second setup was used.

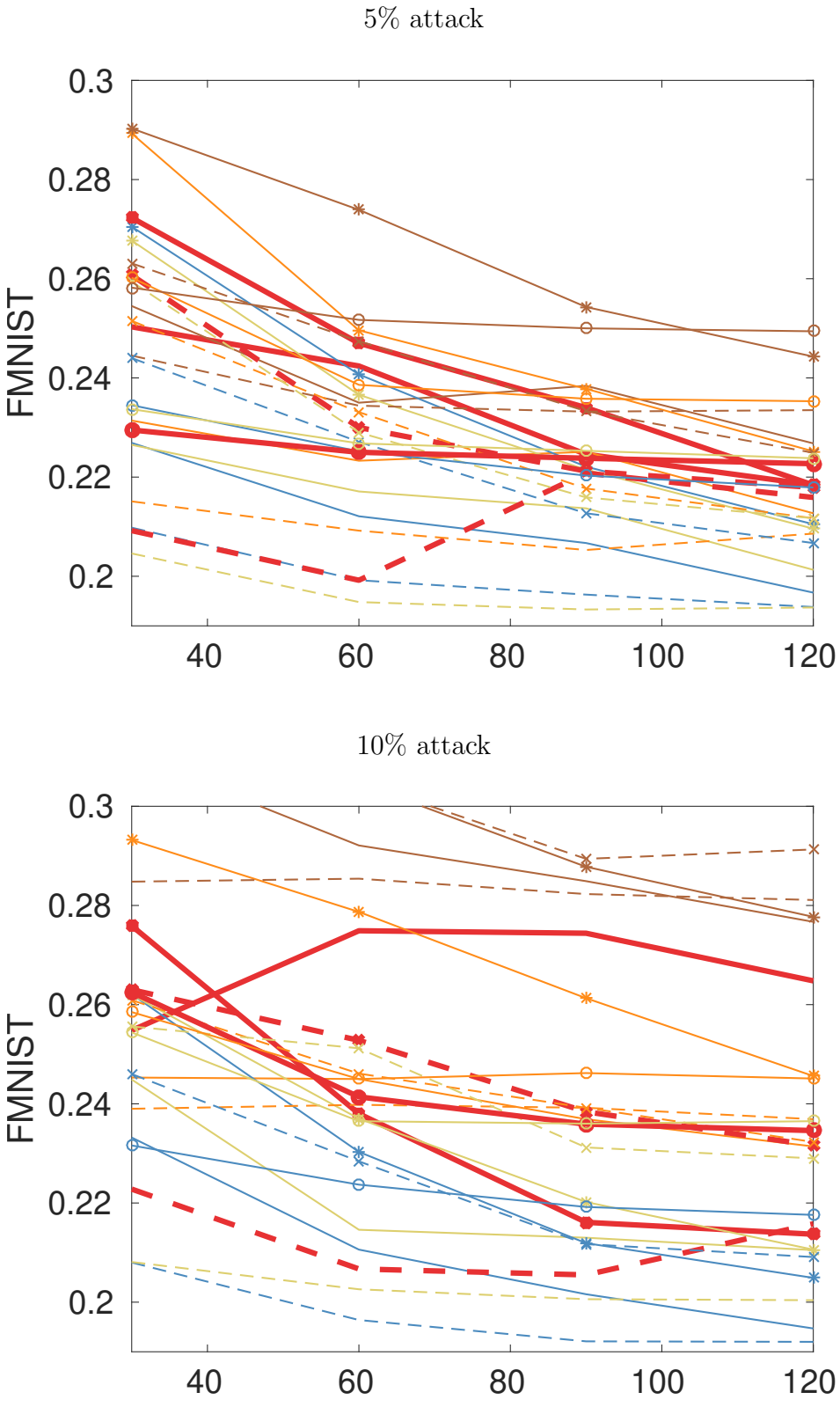


Figure 3.17: Different attack techniques and models are presented for FMNIST dataset. Colors and curve shapes present the attack techniques and models, respectively. model Vertical axis presents error ratio over the testset and horizontal axis presents number of base models. For this figure, second setup was used.

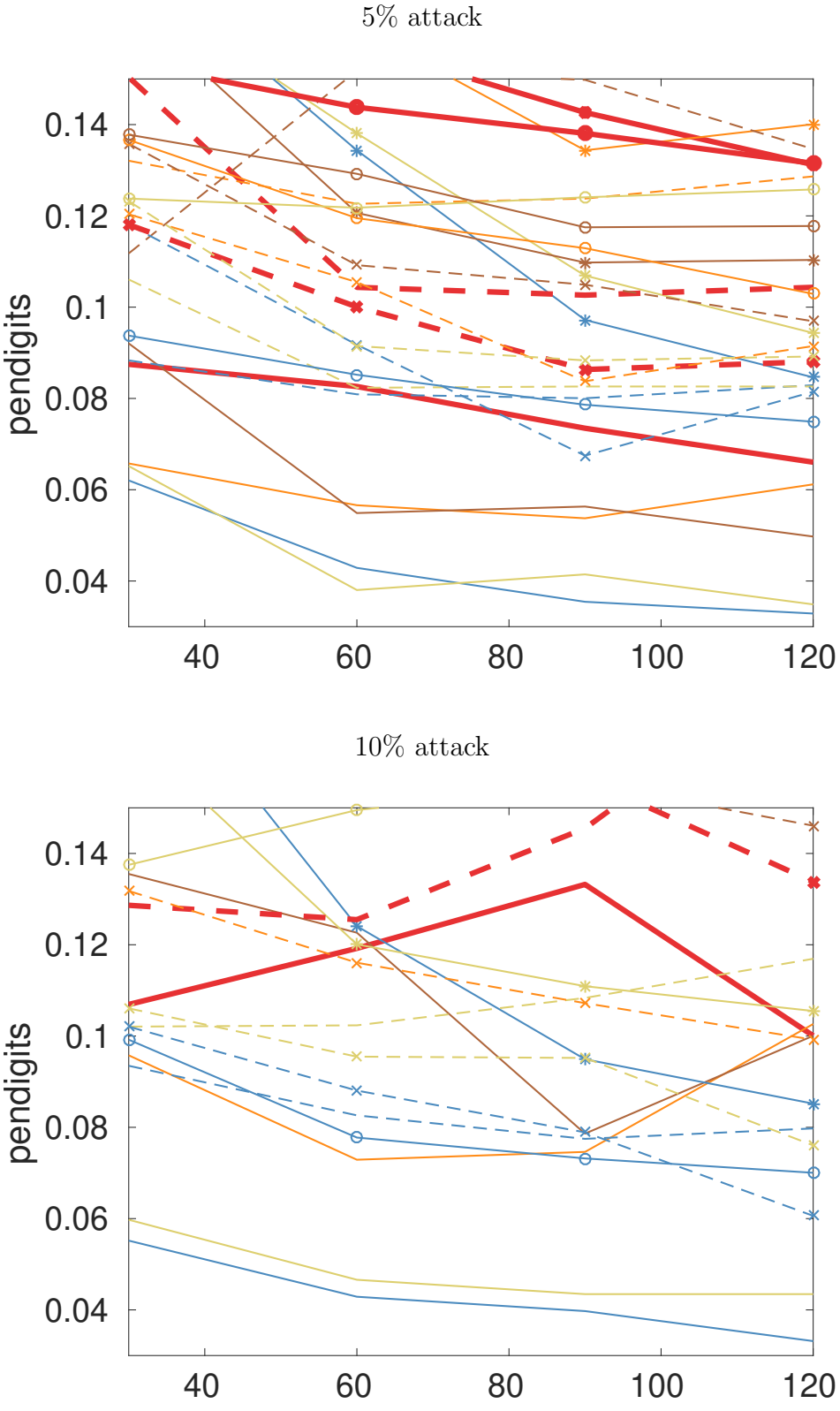


Figure 3.18: Different attack techniques and models are presented for pendigits dataset. Colors and curve shapes present the attack techniques and models, respectively. model Vertical axis presents error ratio over the testset and horizontal axis presents number of base models. For this figure, second setup was used.

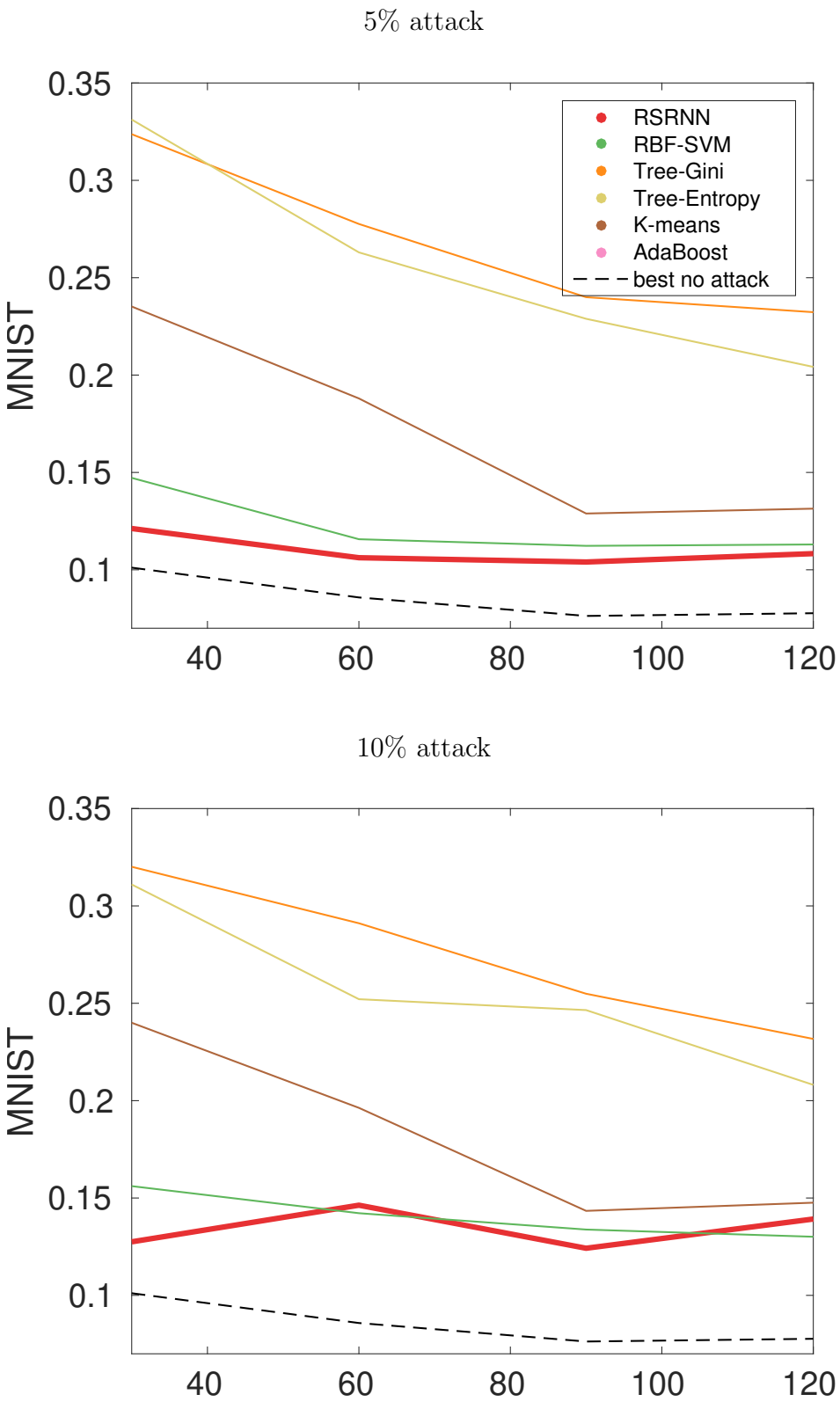


Figure 3.19: Different defenses against SRNN-att. SRNN-att model was trained with 60 centroids. Vertical axis shows test error ratio and horizontal axis represents number of basis models. First setup for datasets was used.

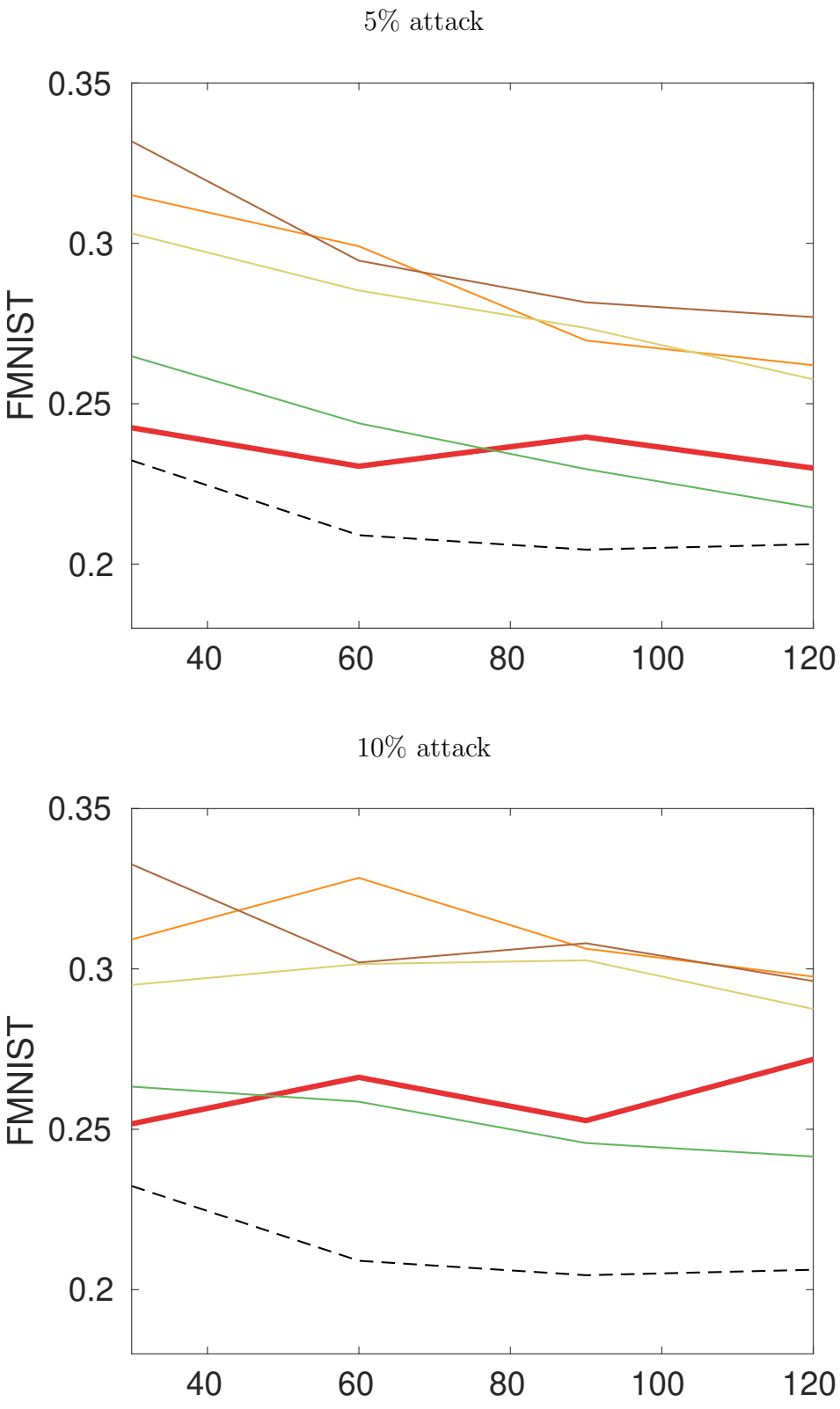


Figure 3.20: Different defenses against SRNN-att. SRNN-att model was trained with 60 centroids. Vertical axis shows test error ratio and horizontal axis represents number of basis models. First setup for datasets was used.

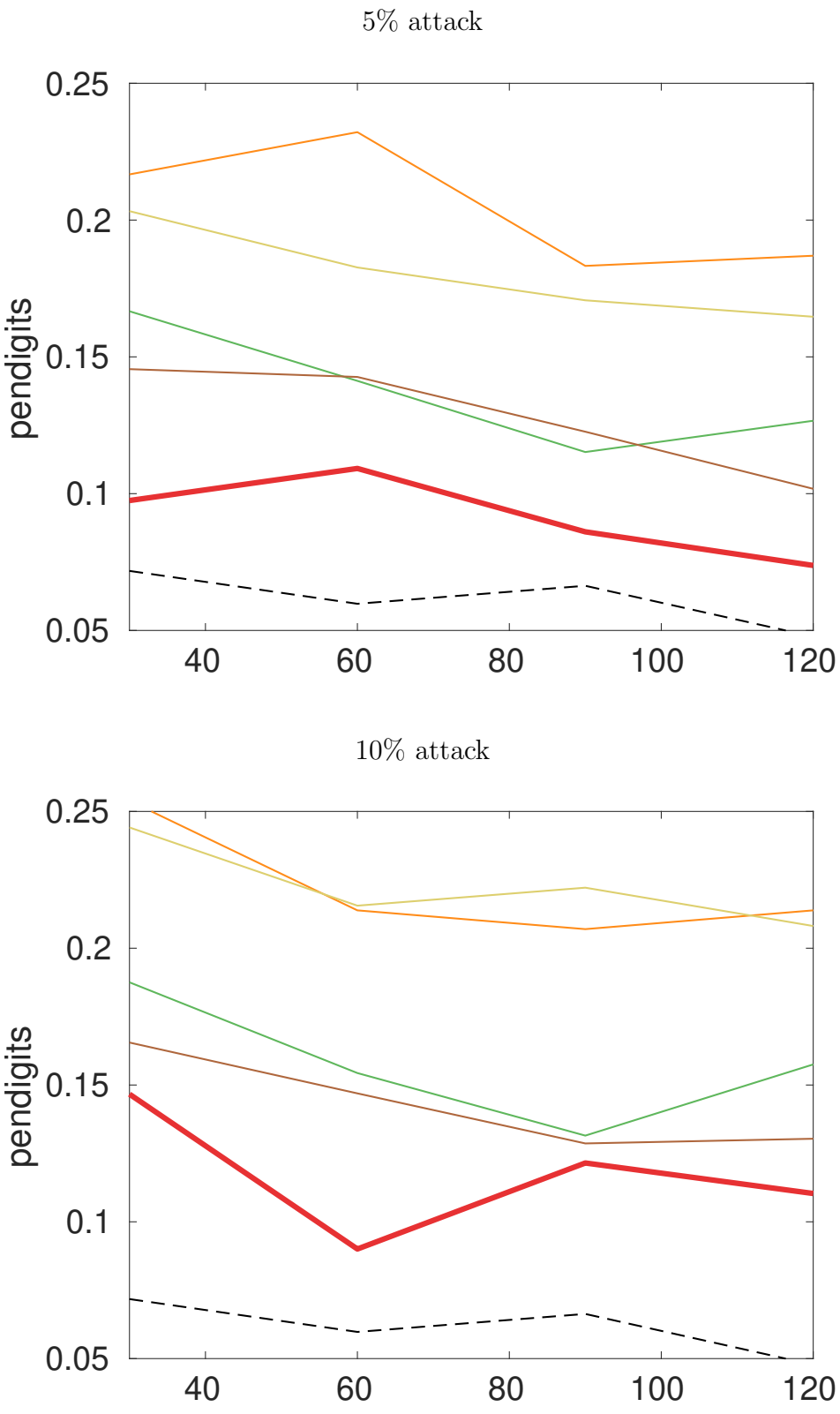


Figure 3.21: Different defenses against SRNN-att. SRNN-att model was trained with 60 centroids. Vertical axis shows test error ratio and horizontal axis represents number of basis models. First setup for datasets was used.

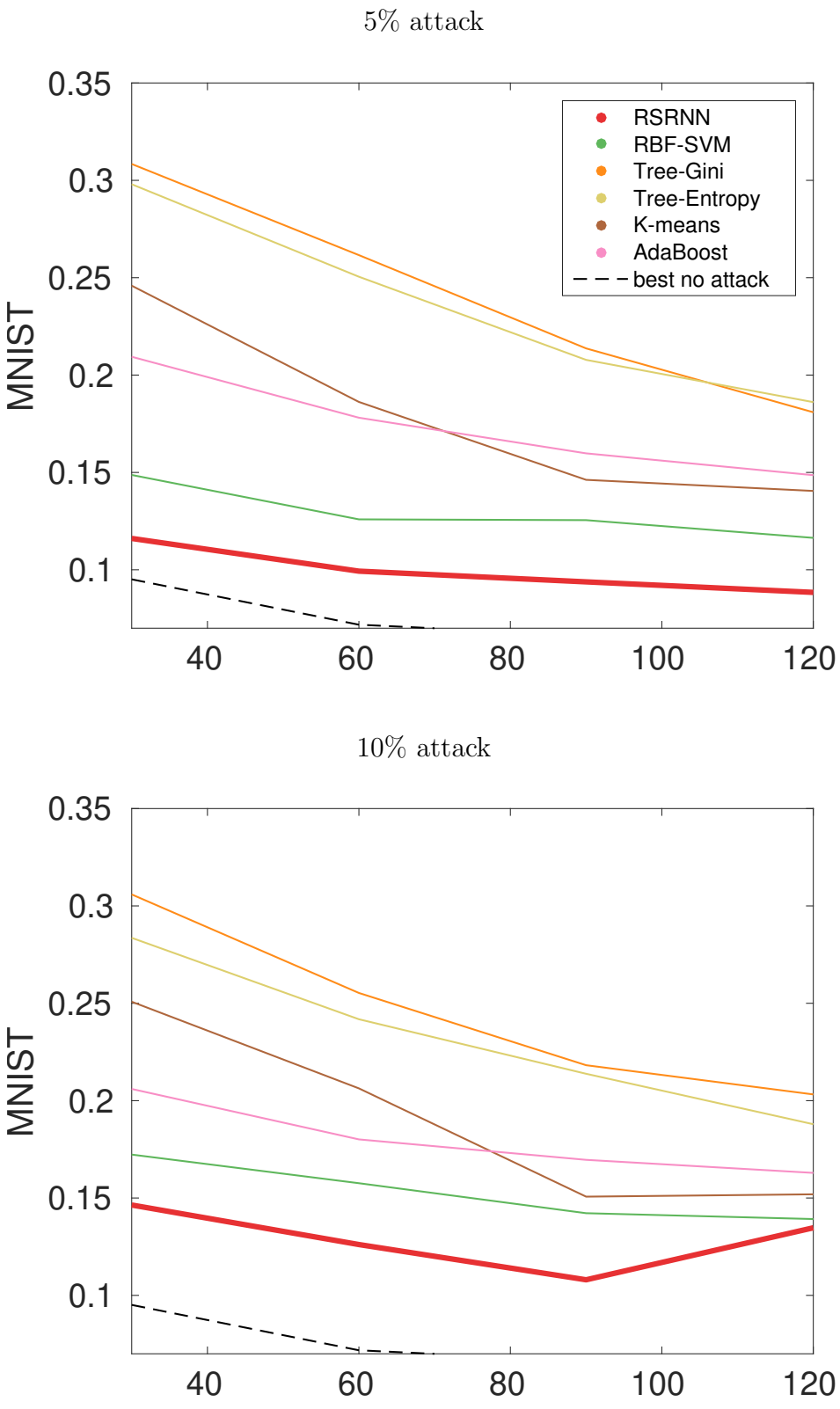


Figure 3.22: Different defenses against SRNN-att. SRNN-att model was trained with 30 centroids. Vertical axis shows test error ratio and horizontal axis represents number of basis models. Second setup for datasets was used.

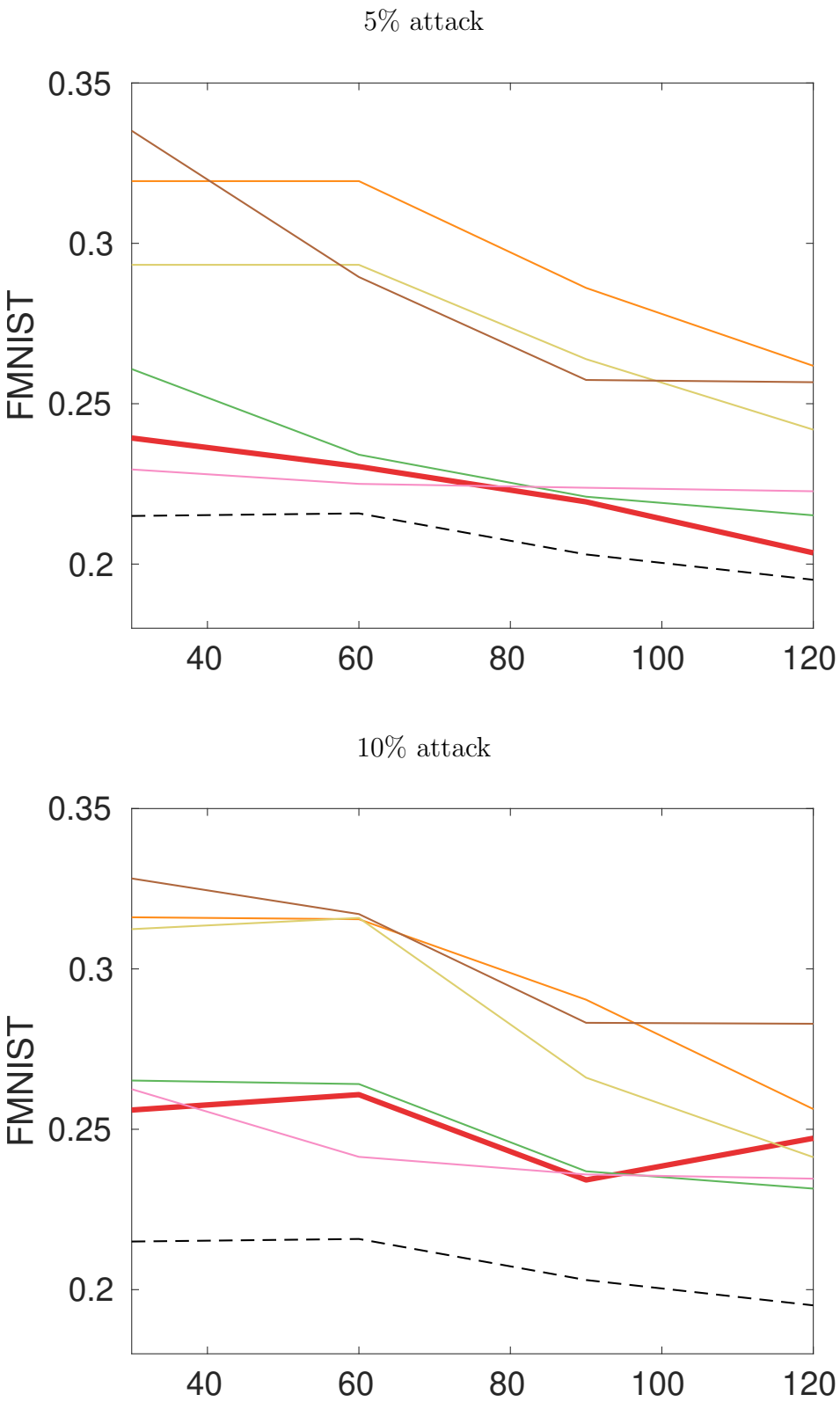


Figure 3.23: Different defenses against SRNN-att. SRNN-att model was trained with 30 centroids. Vertical axis shows test error ratio and horizontal axis represents number of basis models. Second setup for datasets was used.

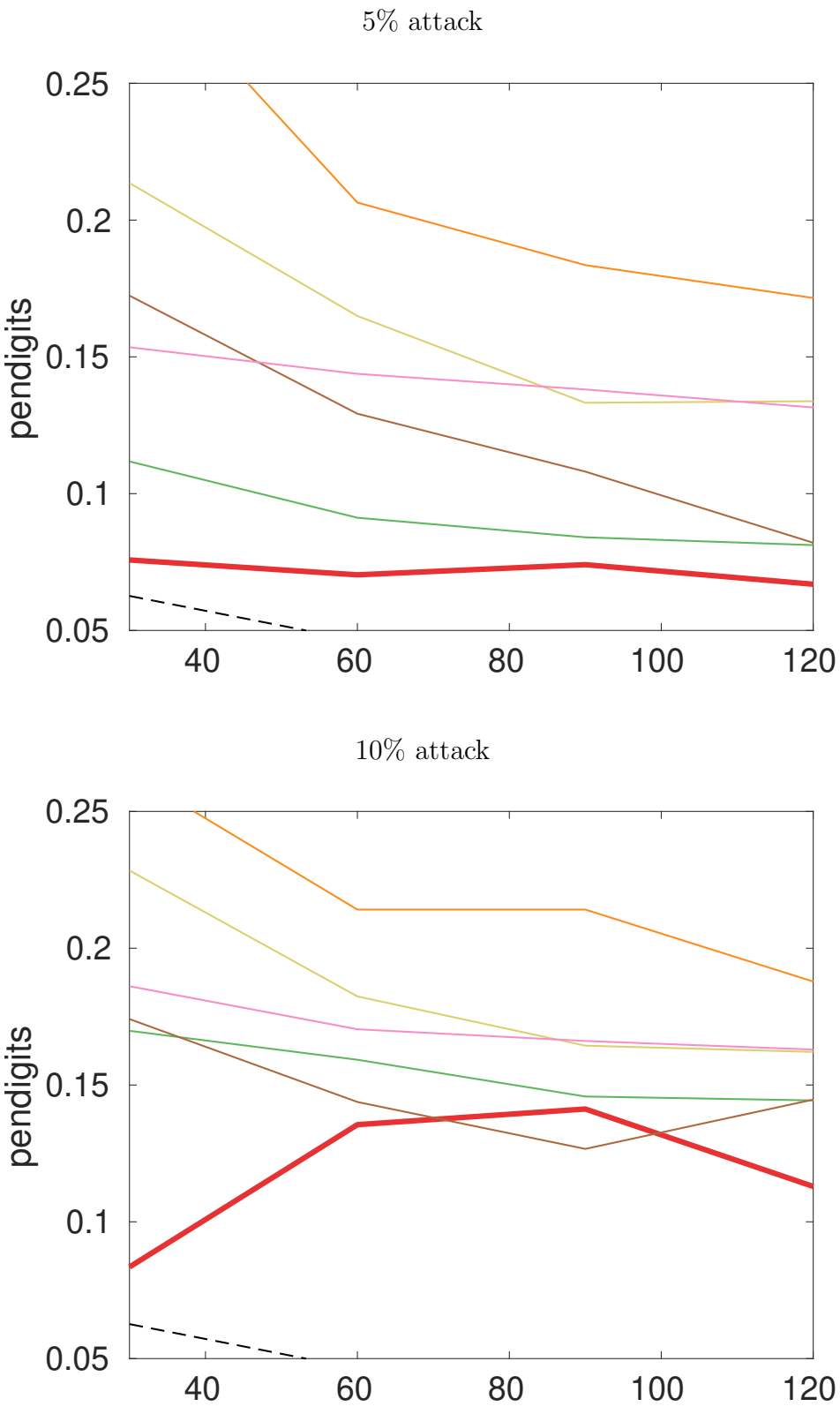


Figure 3.24: Different defenses against SRNN-att. SRNN-att model was trained with 30 centroids. Vertical axis shows test error ratio and horizontal axis represents number of basis models. Second setup for datasets was used.

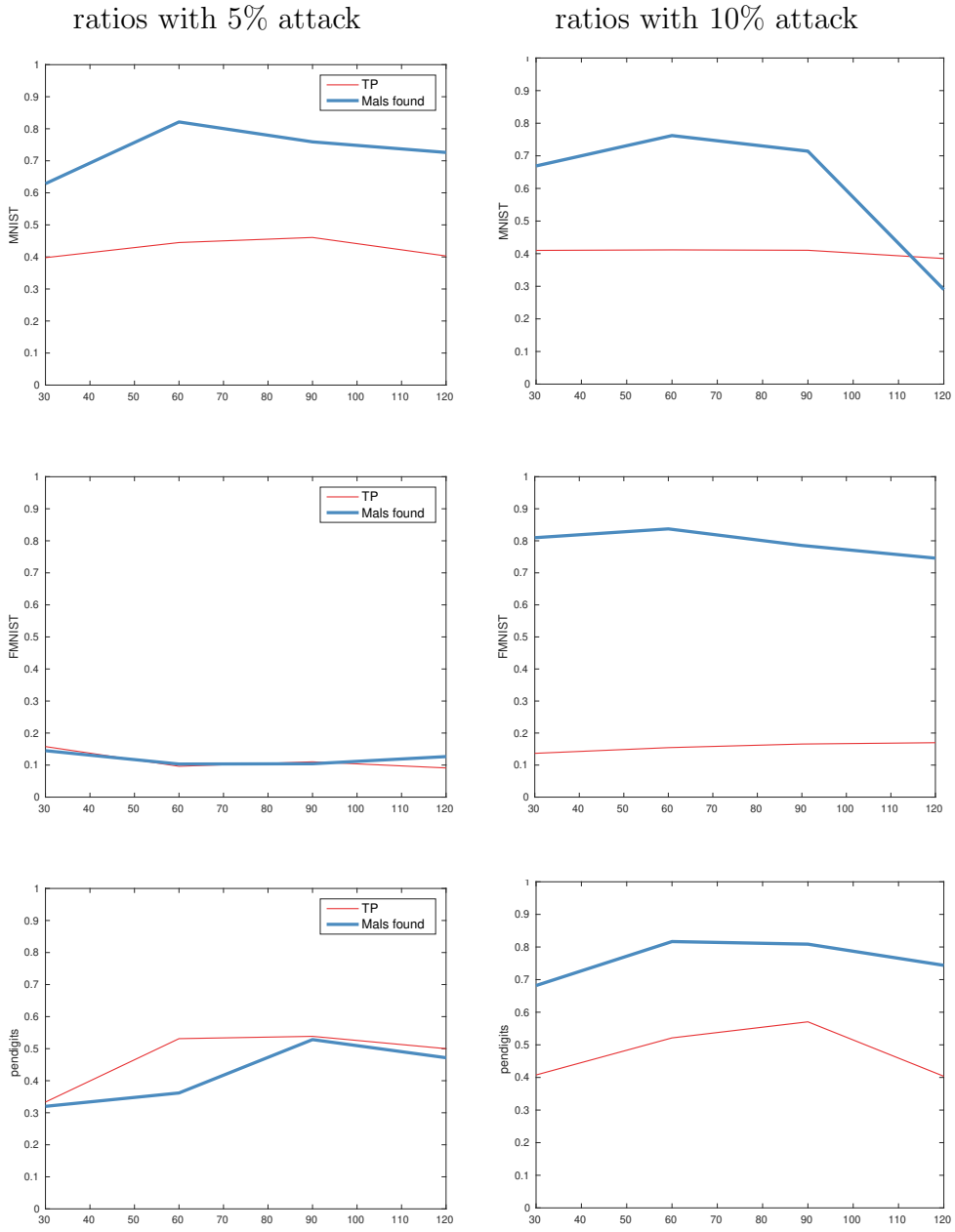


Figure 3.25: The vertical axis represents the ratio. The horizontal axis presents the number of base models. The red line shows the TP and blue line shows the ratio of found malicious samples. The curves correspond to the same experiments as in figures 3.6, 3.8, 3.7, and 3.9.

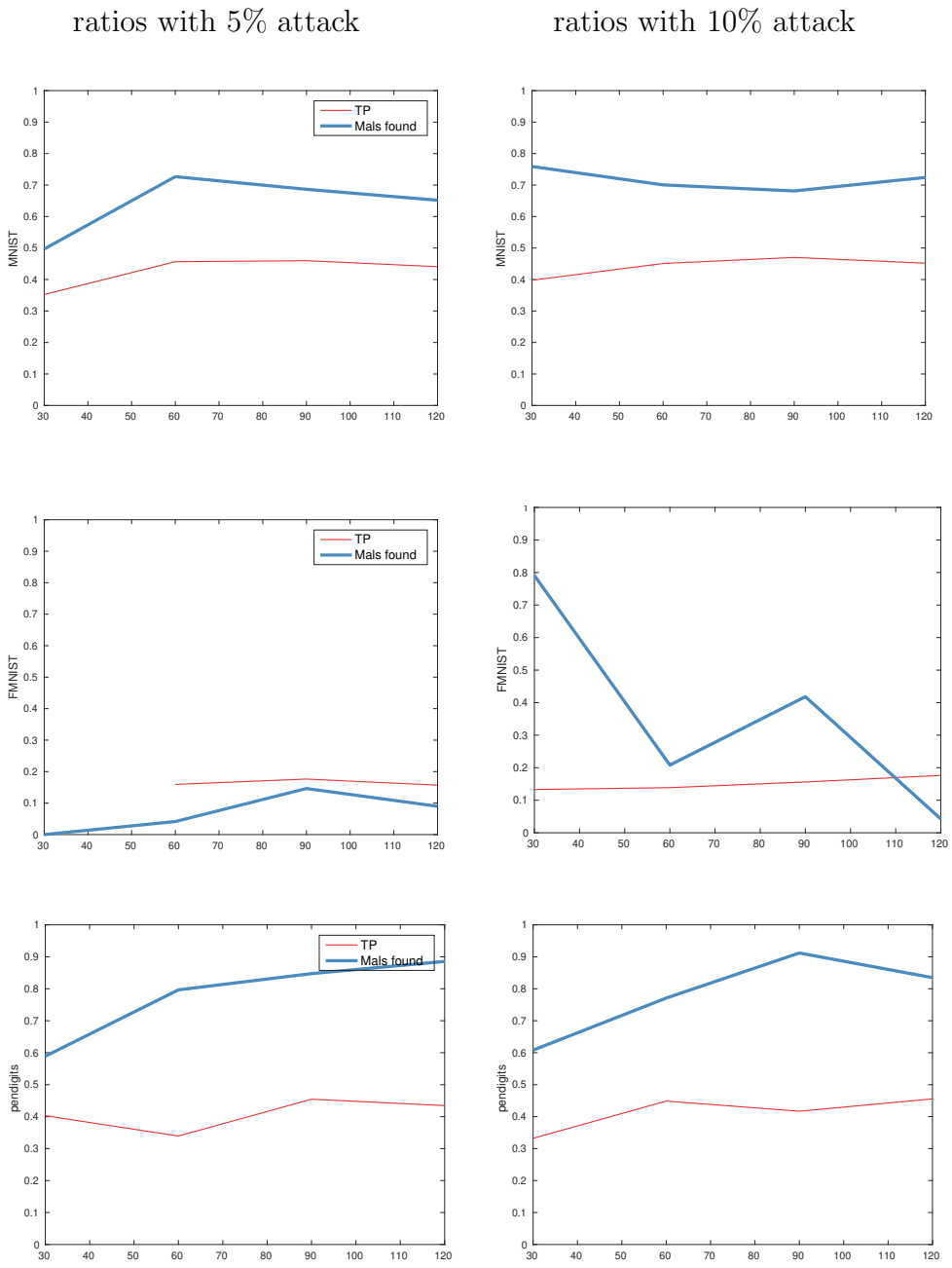


Figure 3.26: The vertical axis represents the ratio. The horizontal axis presents the number of base models. The red line shows the TP and blue line shows the ratio of found malicious samples. The curves correspond to the same experiments as in figures 3.22, 3.23, and 3.24.

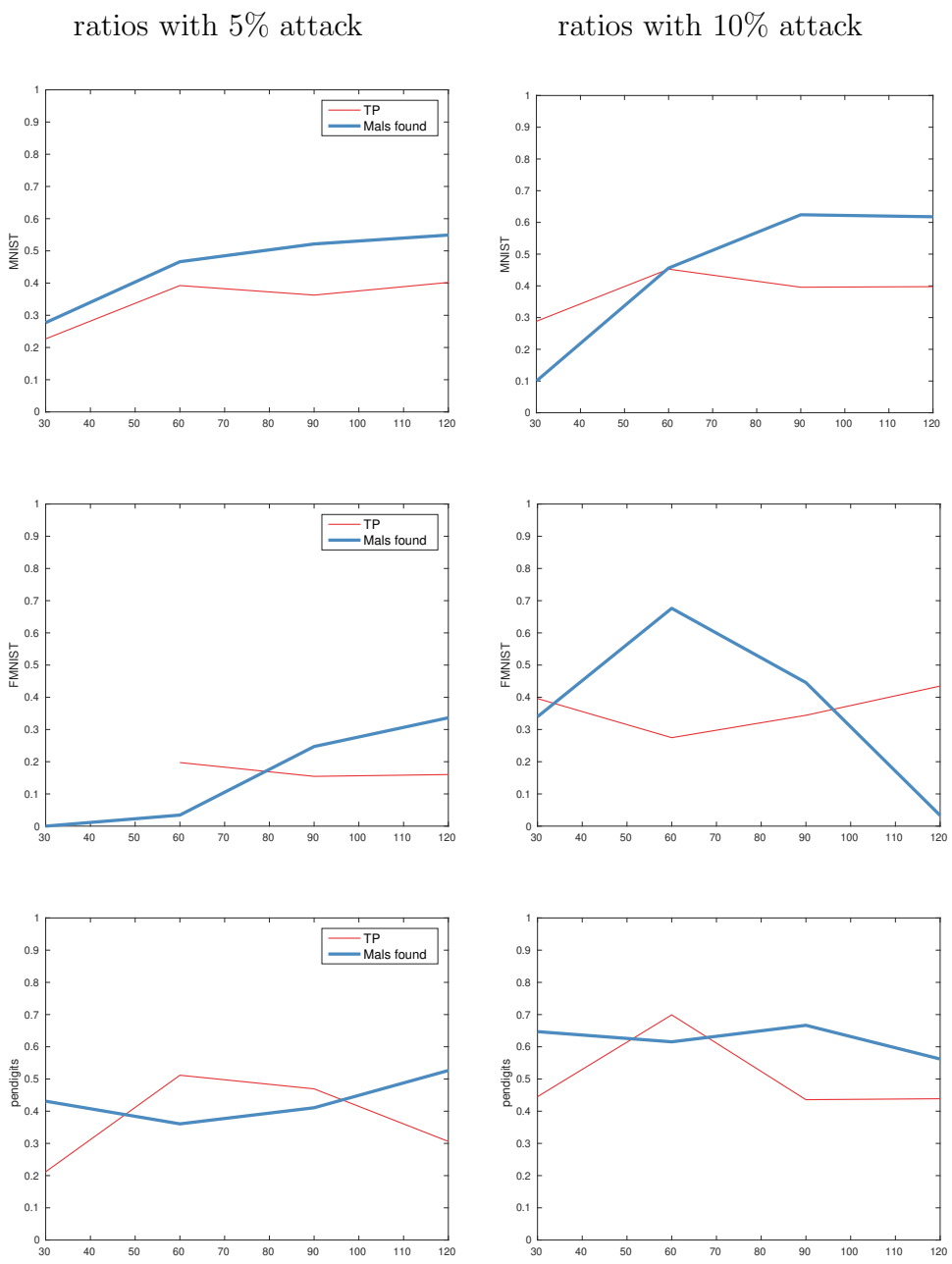


Figure 3.27: The vertical axis represents the ratio. The horizontal axis presents the number of base models. The red line shows the TP and blue line shows the ratio of found malicious samples. The curves correspond to the same experiments as in figure 3.19,3.20, and 3.21.

3.6.2 Defense Experiments

In the experiments of this section, we further investigated the resilience of the proposed defense technique against SRNN-att with 60 centroids. Figures 3.19, 3.20, and 3.21 present results of this experiments. 20% of each dataset was used as the trainset.

Figures 3.19, 3.20, and 3.21 present that the defense technique was able to achieve lower or as good as best of other models.

Figures 3.22, 3.23, and 3.24 present experiments with 80% of datasets used as trainset and 30 centroids for SRNN-att.

In experiments of figures 3.22, 3.23, and 3.24, RSRNN was able to outperform other defense techniques by having smaller test error.

3.6.3 Malicious Sample Detection

As explained in the pruning section, the algorithm is capable of detecting the malicious samples. In this part, detection ratio and true positive (TP) ratio of RSRNN are shown. Figure 3.25 presents the ratio of malicious found samples and TP for figures 3.6, 3.8, 3.7, and 3.9. Figures 3.26 and 3.27 present the ratio of found malicious samples and TP for figures 3.22, 3.23, 3.24, 3.19, 3.20, and 3.21. It is observable that the RSRNN was able to find up to 0.7 of malicious samples with a true positive ratio of 0.45.

3.7 Summary

In this chapter, a novel data poisoning attack was proposed that is able to deteriorate the performance and undermine the integrity of state of the art machine learning models. This is the first data poisoning technique that is not limited to only binary classification, a specific model, or gradient-based approaches. At the same time, the attack technique is very fast since it only takes a linear time over the dataset to select samples for label flipping. The proposed attack shows the fact that for an adversary, it would be more efficient to target minority groups in a dataset with the goal of undermining the integrity of the learned model and

achieving this goal. The properties of this attack does not require the knowledge of the user model since its attack can affect any model.

In addition, a novel defense technique based on SRNN model was proposed that is resistant against the proposed attack technique. The defense technique intrinsically detects and excludes the adversarial samples on the fly during its training procedure. Additionally, our experimental results showed that the RSRNN model is capable of detecting a large portion the malicious samples, thus, making it more robust to data poisoning. Finally, in experiments, RSRNN showed ability to achieve significantly lower test error compared to other known resilient models against the label flipping data poisoning.

Chapter 4

A Regression Model Synthetic Reduced Nearest Neighbor: an Expectation Maximization Approach

A well-known and accurate approach to any task in machine learning is nearest neighbor models. In this chapter, we propose the synthetic reduced nearest neighbor for regression. To the best of our knowledge, no algorithm for optimizing regression synthetic reduced nearest neighbor or prototype nearest neighbor has been proposed in the literature. The existing prototype nearest neighbor models rely on a specific initialization where a k-means model is trained over each class. However, such initialization is only applicable to the classification tasks. Therefore, in this chapter, a novel initialization and expectation maximization approach are proposed for constructing the regression synthetic reduced nearest neighbor. The initialization is based on running k-means algorithm over the output responses of the samples to create various clusters of outputs and then learning several centroids in the input space for each clusters found from the outputs. In other words, the initialization consists of finding output clusters and running k-means in the input space for each found output cluster. The optimization is done by apply-

ing an expectation maximization algorithm similar to the k-means algorithm that optimizes the centroids in the input space. The optimization proposed algorithm consists of two steps. 1- The assignment step, where assignments of the samples to each centroid is found and the optimum of output response for each centroid is calculated. 2- The update/centroid step, where each centroid is updated such that the loss function of the whole model is minimized. We will show that the centroid step operates over all the samples and consists of solving a specific weighted binary classification. However, finding the optimum of the centroid problem is NP-hard and no surrogate objective function exists for approximating the solution. Therefore, a new surrogate is proposed and used to approximate the solution for the centroid step. In this chapter, the consistency of the model is studied and it will be shown that the model is consistent under mild-assumptions. Finally, the model is compared with several other state-of-the-art regression models.

4.1 Introduction

One of the main topics of research in Machine Learning is the relation between the features and output responses [55, 104, 118, 29]. Synthetic reduced nearest neighbor is able to find the relation between features of the inputs and the subclusters of each class [115]. However, to the best of our knowledge, there are no algorithms toward optimizing or even constructing regression reduced nearest neighbor. regression reduced nearest neighbor can potentially have extensive applications toward epidemiological studies [114, 27], medical studies [29, 52, 51] and generally the regression tasks [118]. Therefore, an important open research problem in the literature is optimization of a Regression Synthetic Reduced Nearest Neighbor (SRNN-Reg).

The SRNN-Reg is capable of finding various modalities of the input data and relate them to the modalities of the output responses. The SRNN-Reg is designed to handle both single-response and multi-response regression. The multi-response regression consists of learning the relation between input samples and several real output responses. The SRNN-Reg partitions the input space into various piecewise

constant regions. Each region is represented by a centroid and its output response. From this perspective, the SRNN-Reg is similar to other piecewise constant models, such as [77, 7, 12, 117, 116]. SRNN-Reg is capable of learning an accurate relation between each cluster of the data and its corresponding output responses. Therefore, SRNN-Reg can potentially be interpretable to human since the centroids essentially represent a cluster of the data.

At the test time, the SRNN-Reg operates the same as a nearest neighbor model except that the centroids represent the dataset [55].

Regression is one of the fundamental tasks in the literature of machine learning. A regression task consists of learning the relation between samples of the input space and output space. Specifically, the function that maps inputs (independent variable X) to the output (dependent variable Y) $f : X \rightarrow Y$. Author in [75] applied least squares in regression for the first time. Afterwards Gauss [48] developed and published the theory for least squares in 1821. Regression is a supervised learning task where Y is a continuous vector ($Y \in R^d$). This is known as multi-response regression. Regression has been the workhorse of various fields [113] and various regression models have been improved and expanded fundamentally over the past few decades [55]. This expansion has been so enormous such that listing all such models and their relationships is a difficult task and is out of the scope of this chapter. However, a brief review of the recent models is presented in this chapter. A key objective function for regression is least squares.

$$||\hat{Y} - Y||^2 \tag{4.1}$$

Where, $\hat{Y}$ is the prediction. A least square error can be seen an unbiased linear model of the minimum variance of underlying data under some assumptions. There are six assumptions that has to be satisfied based on Gauss-Markov Theorem [48]. Ordinary least squares might fail to properly predict outcomes if it is applied to certain areas or the assumptions are not true. Therefore, it is important to understand the assumptions and apply the proper changes to the objective function of (4.1) and modify the model [113]. Such changes can be imposing regularizing terms or constraints over the objective function. In the literature, the researchers have extensively dealt with some of the well-known concerns that might violate

the assumptions, such as Ridge [60, 59], Lasso [118], Elastic Net [134], trees [97], forest [20], boosting [22] and so on.

Various works have considered applying the decision trees to the multi-response regression [21, 33]. In [21, 96], authors considered training of a decision tree for each individual output response. However, such approach constructs a large model specially if the number of output responses are high. Another approach [33] consists of constructing a single decision tree for a all the output responses. In other words, this model predicts all the output values simultaneously through a single decision tree. However, a model for all the outputs might not be sufficient [69].

Authors in [69] have explored and compared two approaches for dealing with multi-response regression problem by comparing learning a model for each output separately (i.e., multiple regression trees) and learning one model for all outputs simultaneously (i.e., a single multi-target regression tree). In order to improve predictive performance, [70] has also considered two ensemble learning techniques, namely, bagging [19, 78] and random forests [20] for regression trees and multi-target regression trees.

In the context of regression trees, various approaches of inducing a tree are presented in the literature. Most decision tree induction methods are concentrated on the splitting criterion used at the growing phase of the tree [61, 76].

Several famous regression models are presented bellow.

The most well-known regression model is a linear regression. Linear regression is a linear approach that defines the relationship between a multivariate input and a scalar output response to be linear. The simplest version of linear regression consists of one scalar variable and it is called simple linear regression. For more than one input variable, the model is known as multiple linear regression [41]. Another version of linear regression is called multivariate linear regression, where multiple output responses are predicted, rather than a single scalar variable [102]. In linear regression, the function that defines the relation between the inputs and the outputs is a linear predictor functions and its parameters are estimated from the data. Such functions are called linear models [107].

Another related regression model to the linear regression is Ridge regression.

The Ridge regression is essentially the same as linear regression where the independent variables are assumed to be highly correlated [57]. It has uses in the fields including econometrics, chemistry, and engineering [53, 125]. The theory was first introduced in [131, 118].

Other well-known regression models consist of bagging, boosting, random forest [29], oblique trees [87, 91, 56], and regression SVM [40].

A related topic to the problem of this chapter is nearest neighbor regression. Nearest neighbor regression and local estimators are well-established methods in the literature of ordinary univariate location estimators ([10, 111, 120]. However, as per our extensive search, no prototype nearest neighbor regression was found and this chapter is the first to propose such model.

Although that there is extensive research on the regression, nearest neighbor, and prototype nearest neighbor (for classification [73, 126, 133]) but there has been no research on optimization and construction of regression synthetic reduced nearest neighbor.

This chapter proposes the first regression synthetic reduced nearest neighbor. In this chapter, a novel initialization (that by itself is competitive to other existing regression models) is proposed. For optimization of the model, an expectation maximization algorithm is proposed. The proposed algorithm directly minimizes the least squares error of the model. The proposed optimization has a provable convergence and monotonically decreases the loss function. Therefore, the algorithm has a convergence guarantee on minimizing the loss function and achieving a local optimum in the sense that no more iterations can decrease the error. It is also worth mentioning that the algorithm does not cycle. The proposed optimization algorithm consists of two steps and is inspired by K-means algorithm [81]. First step is the assignment step. The assignment step is composed of finding samples assignments and optimum output response of the centroid. Second step is the update step where the centroid is optimized such that the loss function is minimized. The centroid step is affected by all the samples and we will show that this update step is a kind of NP-hard weighted binary classification problem. The update step is solved through a surrogate objective function that is similar to ob-

jective function of SVM. It is shown that the algorithm is efficient because of its linear computational complexity. Finally, the model is tested on various datasets with various sizes and dimensionality. It is shown that SRNN-Reg is capable of competing and performing better than other similar regression models.

4.2 Proposed Method

4.2.1 Preliminaries

Assume a dataset consisting of tuples (x_i, y_i) where x, y and i represent input features, output responses and index number, respectively. Each tuple represent a data x_i and its corresponding output response y_i. Here, $x_i \in \mathbb{R}^D$ and $y_i \in \mathbb{R}^d$. The Regression Synthetic Reduced Nearest Neighbor (SRNN-Reg) consists of K tuples of synthetically produced centroids/prototypes $(c_j, \hat{y}_j)$ where c, $\hat{y}_j$ and j represent the centroid's point in the input space, output prediction and index. At the inference time, the SRNN-Reg operates like a nearest neighbor model where the centroids are used as the samples. The problem of training SRNN-Reg is as follows:

$$\begin{aligned} \min_{\{(c_j,\hat{y_j})\}_1^K} \quad & \sum_{i=1}^{N} ||y_i - \hat{y}_{j_i^*}||^2 \\ \text{s.t} \quad & j_i^* = \underset{\{j\}_1^K}{\operatorname{argmin}} \quad d(x_i - c_j) \end{aligned} \tag{4.2}$$

where $d(.)$ is a distance metric. Through this chapter we use the l-2 norm as the distance metric:

$$d(x_i - c_j) = \sqrt{||x_i - c_j||^2} \tag{4.3}$$

The prediction of the model consists of the output prediction of closest centroid to the input sample. We define SRNN-Reg as follows:

$$NN(x) = \sum_{j=1}^{K} y_j I(x \in R_j) \tag{4.4}$$

where, $NN(.)$ represents a nearest neighbor function of the K centroids. $I(.)$ is an indicator function that produces 1 if the input x is in the region of R_j. Similar

notation is used in [55]. R_j represents the region where the closest centroid to the points in that region is c_j.

4.2.2 Initialization

Every numerical optimization algorithm needs an initialization [90]. Here, we propose a novel initialization for the regression synthetic reduced nearest neighbor. Previously, in the literature, the initialization of SRNN models consisted of learning a K-means model for each class of the data. For example, in case of M classes and K centroids, $\frac{K}{M}$ centroids are learned for each class as initialization of the SRNN [73, 126, 133, 115]. However, such approach is not applicable to the regression. This initialization also has close ties with naive Bayes and density estimation [109]. Here, we expand this initialization to the case of SRNN-Reg.

Intuitively, the output responses can consist of several modalities. In other words, it is possible that the output responses are generated from several distributions. The clusters of such distributions can be approximated by running a K-means over the output space (M centroids). Assume that S_m represents the set of samples assigned to each output cluster. Next step consists of learning $\frac{K|S_m|}{N}$ centroids over the input features of the S_m for all M clusters. In other words, we learn centroids over the input features of each output cluster relative to the population of that cluster. The found centroids at the second step are used as initialization for the SRNN-Reg. At this step, $\hat{y}_j$ is found using the following formula:

$$\hat{y}_j = mean(y_i \in S_j) \tag{4.5}$$

where S_j represents the set of samples that are assigned to j^{th} centroid. S_j essentially consists of samples where j^{th} centroid is the closest centroid to them. $mean(.)$ represents the average of its input set. Note that $S_j \in R_j$.

4.2.3 Consistency

Here we discuss the consistency of the SRNN-Reg. We show that SRNN-Reg is nonparametric and consistent under mild assumptions for continuous features.

Assume an independent identically distributed (iid) dataset where it is being generated from $f(x)$. $f(x)$ represents the true function for relation between the inputs and outputs. Also assume that $f(x)$ is a piecewise constant function. Over a set of N observations, consistency of a nonparametric estimator $\hat{f}_N(x)$ (such as SRNN-Reg) is shown using the following formula [95]:

$$Pr(\lim_{N\to\infty} \int_x (\hat{f}_N(x) - f(x))^2 dx = 0) = 1 \tag{4.6}$$

The proof of consistency of SRNN-Reg is similar to proof of consistency for regression trees [21] and density estimators in [98].

Theorem 5 (Consistency of SRNN-Reg)**.** *The estimator defined in* (4.4) *satisfies equation* (4.6)*.*

Proof. Assume B and d_1 denote the collection of all sets $t \subset X$ and a fixed positive integer, respectively. Assume that B describes the sets that are the solution to a system of d_1 inequalities $b^T x \leq c$ where $b \in \mathbb{R}^d$ and $c \in \mathbb{R}$. Every region in the SRNN-Reg in formula (4.4) can be seen as a solution of a system of d_1 inequalities of the form $b^T x \leq c$ where b is a hot-one-vector (only one element of b is 1 and the rest are 0). Therefore, $SRNN - Reg \subset B$.

Assume a random point X_n from function f on X, $(n \geq 1)$. $\hat{F}_N$ represents the empirical function learned by SRNN-Reg over X_n For $N \geq 1$, $1 \leq n \leq N$, and defined on a set $t \subset X$ by

$$\hat{F}_N(t) = \frac{1}{N} \sum_{n=1}^{N} y_n I(X_n \in t) = mean(y_n \in R_t) = \int_t \hat{f}_N(x) dx \tag{4.7}$$

where $y_n = f(x_n)$ and $\hat{f}_(x)$ is the estimator presented in (4.4). Using a general version of Glivenko-Cantelli theorem [122]

$$Pr(\lim_{N\to\infty} sup_{t\in B} |\hat{F}_N(t) - \int_t f(x)dx| = 0) = 1 \tag{4.8}$$

By replacing equation (4.7) in (4.8), we have

$$Pr(\lim_{N\to\infty} sup_{t\in B} |\int_t \hat{f}_N(x)dx - \int_t f(x)dx| = 0) = 1 \tag{4.9}$$

then:

$$Pr(\lim_{N\to\infty} \sup_{t\in B} \int_t |\hat{f}_N(x) - f(x)|dx \geq 0) = 1 \tag{4.10}$$

Further, by assuming that the region of t leans toward 0 as $N \to \infty$ ($Pr(\lim_{N\to\infty} \int_t dx = 0) = 1$). Rest of the steps will follow similar to theorem 1 in [98] and we get

$$Pr(\lim_{N\to\infty} \int_x (\hat{f}_N(x) - f(x))^2 dx = 0) = 1 \tag{4.11}$$

□

Please note that the steps are similarly done in [98] except the step of (4.7) which is different.

The consistency shows that as the number of samples go to infinity and as $\frac{K}{|S_j|} \to 0$, then the SRNN-Reg is consistent. This is provable thanks to the assumption that the true function that relates the inputs to the outputs is a piecewise constant function. This essentially means that the SRNN itself is consistent also for classification tasks.

4.2.4 The Expectation Maximization of SRNN-Reg

The problem (4.2) represents the training problem of SRNN-Reg. SRNN generally is known as prototype nearest neighbor in other papers of literature and centroids are also called prototypes. Problem (4.2) resembles K-means [81] problem except that the loss function is $||x_i - c_j||$. The expectation maximization in this chapter follows same approach as to K-means algorithm and the approach in [115]. The optimization consists of two steps, assignment step and the update step.

Assignment Step

The assignment step consists of calculating the assignment of the samples to each centroid and finding the optimum output prediction of each centroid. The

problem of this step is as follows:

$$\begin{aligned} \min_{\{\hat{y_j}\}_1^K} \quad & \sum_{i=1}^{N} ||y_i - \hat{y_j}^*||^2. \\ \text{s.t} \quad & j_i^* = \underset{\{j\}_1^K}{\operatorname{argmin}} \quad d(x_i - c_j). \end{aligned} \tag{4.12}$$

Note that problem (4.12) only tends to optimize over $\hat{y_j}$ for $j = 1...K$. Using sets S_j, problem (4.12) can be simplified to:

$$\min_{\{\hat{y_j}\}_1^K} \quad \sum_{j=1}^{K} \sum_{\{i|(x_i,y_i)\in S_j\}}^{N} ||y_i - \hat{y_j}||^2. \tag{4.13}$$

The problem (4.13) can be separated over each centroid and its corresponding set. This can be done since the regions are distinct and samples can not be shared among the regions. The prediction for each region is the label of j^{th} centroid $\hat{y_j}$ (the centroid that represents the region). As a result, the problem of optimizing label for each region is

$$\min_{\hat{y_j}} \quad \sum_{\{i|(x_i,y_i)\in S_j\}}^{N} ||y_i - \hat{y_j}||^2. \tag{4.14}$$

whose minimum is presented in formula (4.5). In other words, the optimum of $\hat{y_j}$ is the mean of response of samples in the R_j.

As a result, in the assignment step, the S_j and $\hat{y_j}$ have to be calculated for all the samples.

Update Step

The update step consists of updating the position of centroids such that the objective function in (4.2) is minimized [81]. At this step the output prediction of the centroids are kept constant. The update step is affected by all the samples in the dataset because by changing the position of centroid, the assignments of the samples can get changed and as a result the prediction for each train sample gets changed. Further it will be shown that finding optimum of the problem in this step is NP-hard. Therefore, the centroid problem is approximated through a novel surrogate objective function.

Each centroid is optimized individually [115]. The optimization problem for k^{th} centroid consists of moving the k^{th} centroid such that the samples' assignment are changed in favor of decreasing the objective function (4.2). The centroid problem is

$$\begin{aligned} \min_{c_k} \quad & \sum_{i=1}^{N} ||y_i - \hat{y_j}^*||^2. \\ \text{s.t} \quad & j_i^* = \underset{\{j\}_1^K}{\operatorname{argmin}} \quad d(x_i - c_j). \end{aligned} \tag{4.15}$$

Note that the optimization is only over c_k. We rewrite the problem (4.15) over the assignment of a sample to k^{th} centroid or to the rest of centroids. For simplicity we introduce the notation $r_{ij} = d(x_i - c_j)$. The problem is

$$\begin{aligned} \min_{c_j} \quad & \sum_{i=1}^{N} ||y_i - \hat{y}_k||^2 U(r_{ij_i^*} - r_{ik}) + ||y_i - \hat{y}_{j_i^*}||^2 U(r_{ik} - r_{ij_i^*}). \\ \text{s.t} \quad & j_i^* = \underset{\{j\}_1^K, j \neq k}{\operatorname{argmin}} \quad d(x_i - c_j). \end{aligned} \tag{4.16}$$

In problem (4.16), $U(.)$ is a step function where it outputs 1 if the input is larger than 0 and otherwise it will produce 0. Note that the input arguments of the step functions are negative of each other. This means a sample has to either get assigned to the k^{th} centroid or the rest of centroids. The sample can not get assigned to both terms of (4.16). The assignment of i^{th} sample to each term will produce a continuous error. This essentially means that the problem (4.16) is a weighted binary classification problem. The problem (4.16) encourages the sample to get assigned to the side that produces lower error. For simplicity we introduce the sets S_c and $S_{c'_k}$. The S_{c_k} represents the set of samples that produce lower error if assigned to the c_k centroid and $S_{c'_k}$ represents the set of other samples. let $t_i = abs(||y_i - \hat{y}_k||^2 - ||y_i - \hat{y}_{j_i^*}||^2)$ where $abs(.)$ returns the absolute value of its input. The problem of (4.16) is equivalent to

$$\min_{c_j} \quad \sum_{(x_i, y_i) \in S_c} t_i U(r_{ij_i^*} - r_{ik}) + \sum_{(x_i, y_i) \in S_{c'_k}} t_i U(r_{ik} - r_{ij_i^*}) \tag{4.17}$$

The problem (4.17) is a NP-hard problem [89]. Therefore, inspired by the SVM, we approximate the solution to problem (4.17) using the following surrogate objective

function

$$c_k^*(\mu) = \underset{c_k}{\text{argmin}} \quad \sum_{x_i \in S_c} t_i r_{ik} + \sum_{x_i \in S_{c'_k}} t_i relu(\mu r_{ij^*} - r_{ik}) \tag{4.18}$$

where μ is a penalty coefficient. Intuitively, the surrogate objective function encourages the k^{th} centroid to stay close to samples of S_c while staying away from samples of $S_{c'_k}$. μ is increased from 0 to 1 and along this path, the $c_k^*(\mu)$ that produces the smallest error for (4.17) is selected. This surrogate is a modified version of surrogate objective function in [115]. The μ acts similar to slack variables in a SVM problem. The objective function of problem (4.18) is a continuous function; thus, a local optimum of the problem can be found using gradient-based algorithms.

Finally, the algorithm iterates over both assignment step and update step until no further improvement over the original objective function can be made. At this point the algorithm stops and it does not cycle.

4.2.5 Relation to EM algorithm

The proposed algorithm is originally inspired by the EM algorithm used for K-means. The assignment step consists of finding the samples assigned to each centroid and finding the optimal output prediction of each centroid. The assignment of samples is the same as calculating the prior probabilities in an EM algorithm. Finding optimum output prediction of each centroid can also be considered as a part of maximization step.

In the update step, the centroids have to be updated. At this step, the outcome of assigning each sample to each centroid is evaluated ($||y_i - y_j||^2_{j=1}^K \forall i = 1, 2, ..., N$). This evaluation is equivalent to calculating the posterior probabilities. Then the centroid problem is approximated using these outcomes which is the maximization step.

4.2.6 Relation of The Centroid Problem to SVM

The problem of finding the best centroid that is closer to samples of S_c than any other centroid can be cast as a feasibility problem.

$$\begin{aligned} \text{find} \quad & c_k \\ \text{s.t} \quad & r_{ik} < r_{ij_i^*} \quad \forall (x_i, y_i) \in S_c \\ & r_{ik} > r_{ij_i^*} \quad \forall (x_i, y_i) \in S_{c'_k} \end{aligned} \tag{4.19}$$

However, this feasibility problem is NP-hard since the the second set of constraints are concave [103]. These constraints make the problem different from similar SVM problems where the constraints are convex and global solution can be found in efficient time. Therefore, we approximated the solution through a novel surrogate objective function.

4.2.7 Properties of the Algorithm

computational complexity Optimizing the centroid problem takes $\mathcal{O}(ND)$ since it uses a gradient based algorithm for solving the surrogate objective. All the K centroids have to optimized at each iteration. Therefore, the computational complexity of the algorithm is $\mathcal{O}(\alpha NDK)$ where α is the number of iterations.

Convergence The convergence to a local optimum is similar to that of the K-means algorithm [81, 115]. At each iteration, the error decreases and the objective function is bounded by 0 from bellow. Further, the different combination of assignment of the samples to the centroids is finite. Therefore, the algorithm stops after a finite number of iterations.

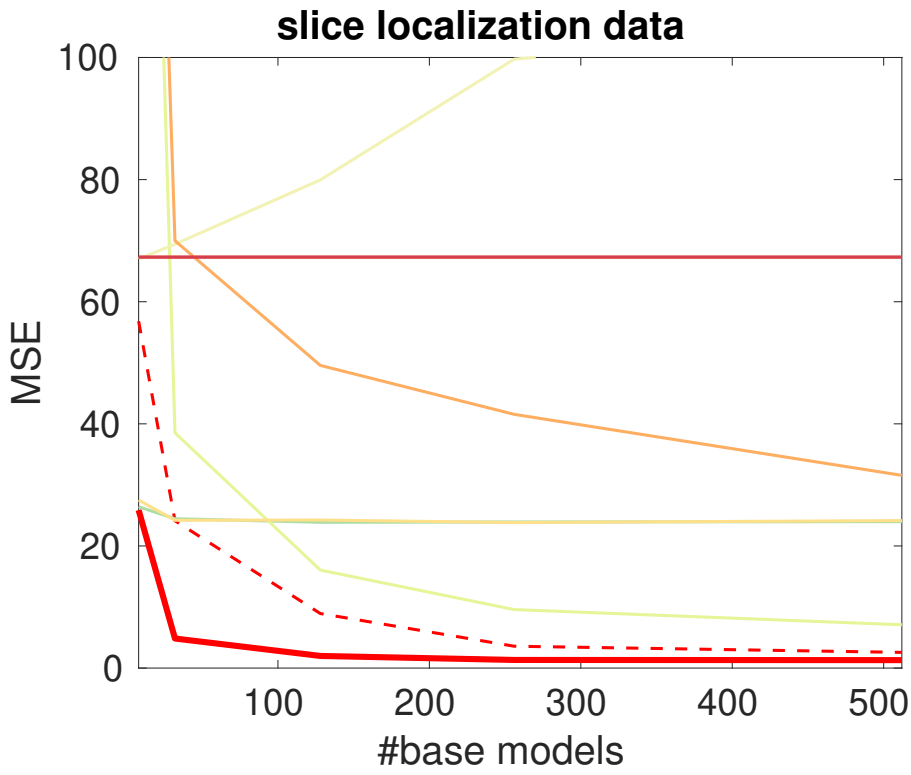
4.3 Experimental Results

In this section, the experimental results are presented to demonstrate the merits of proposed proposed algorithm. Various datasets are downloaded and used for evaluation from UCI repository [38]. We partitioned the datasets using the following setups: 1- if the dataset contained a separate test set, the dataset was

used as provided in the repository. In case cross-validation was needed (for specific models) and validation set was not provided then the we partitioned the trainset to 80% train and 20% validation sets. 2- if the test set was not provided we partitioned the dataset to 80% train and 20% test set. For models that needed validationset, we divided the trainset to 80% train and 20% validation set. 3- if all sets were provided by the repository, then the sets were used as provided. The setup for SRNN-Reg is as follows: at the initialization phase, we used $K = 4$ for the output clusters for slice localization data and for the rest of datasets $K = 2$ was used for the output clusters. The number of input centroids used for each output cluster were proportional to the population of each cluster. The centroid problem was optimized using batch stochastic gradient descent. We used various setups for each dataset to achieve the smallest possible train error.

Various models are used for comparison with SRNN-Reg. The basis of comparison is based on the number of base models used in each model. All the models comparable to the SRNN-Reg can be seen as a kind of ensemble model that consists of several base models. In case of forest and boosting, each tree is a base model. For prototype models such as SRNN-Reg and K-means, each centroid is a base model. For Radial Basis Function (RBF) models, each RBF is considered as a base model. We also compared our model with linear regression and Ridge regression. The two models are presented with straight lines. Figures 4.1, 4.2, 4.3, 4.4, and 4.5 present performance of each model over various datasets based on mean squared error versus number of base models. SRNN-Reg (SRNN) and its initialization (SRNN0) are compared with Random Forest (RF), K-means, Regression Boosting (regboost), bagging, Radial Basis Function+ linear regression (RBF+lin-reg), linear regression (lin-reg), and ridge regression (ridgeCV). For the RF, each tree was trained using 70% of the trainset and 0.7 of features were randomly selected and used at the split nodes. For the K-means, a K-means model with desired number of centroids were trained over trainset and the assignment step of SRNN-Reg was applied to the model. For the regboost, we used trees of depth 3 (MAX_DEPTH=3). For bagging, similar setup to RF was used except that no feature randomization is applied. RBF+lin-reg consists of first training RBFs

train error



test error

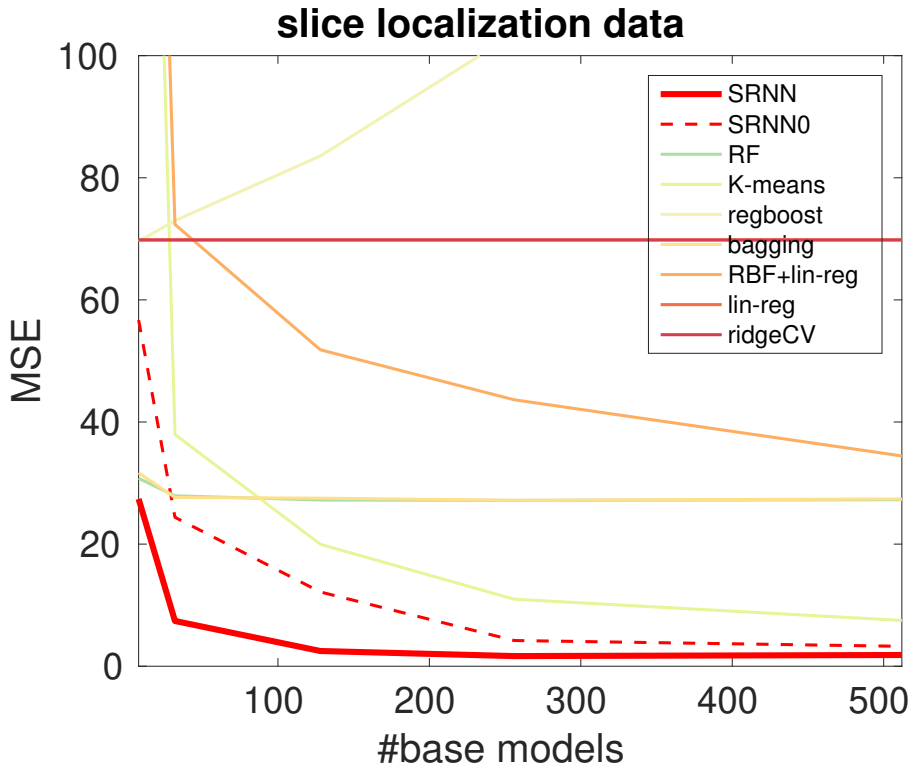
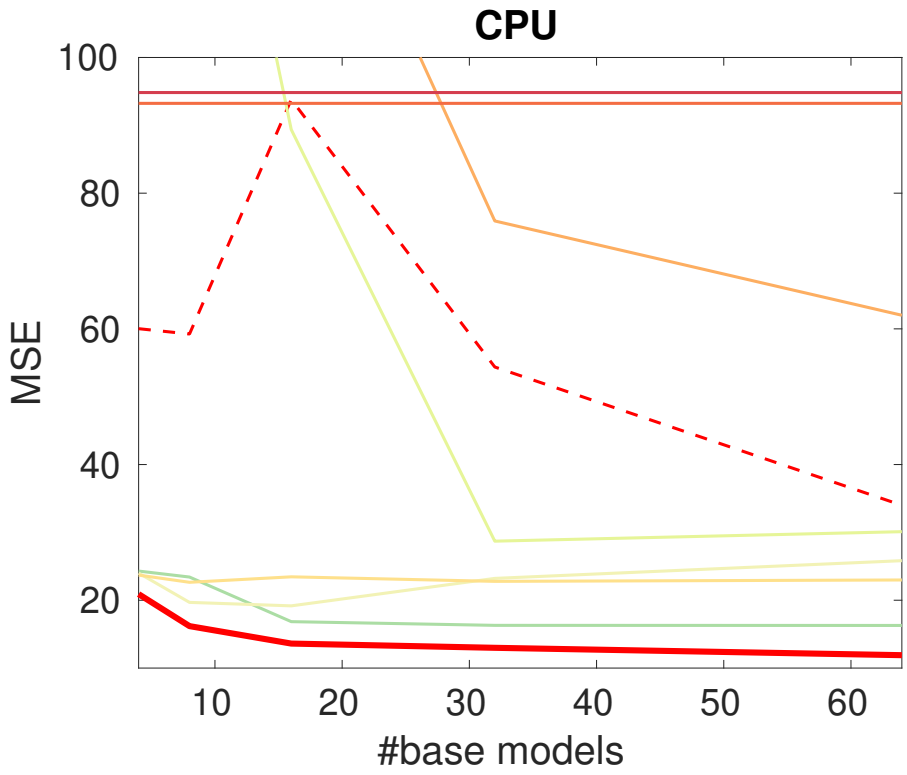


Figure 4.1: Horizontal axis represents the number of base models(e.g., centroids, RBFs, trees and etc.). The vertical axis represents the mean squared error. SRNN-Reg is compared with various similar models on the dataset of slice localization data.

train error



test error

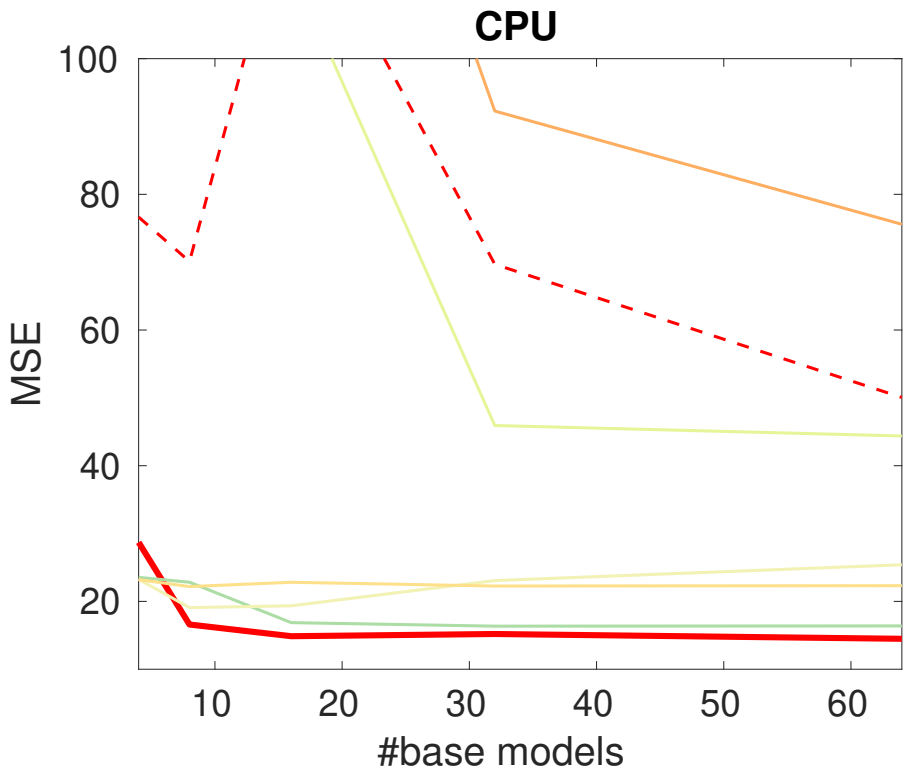
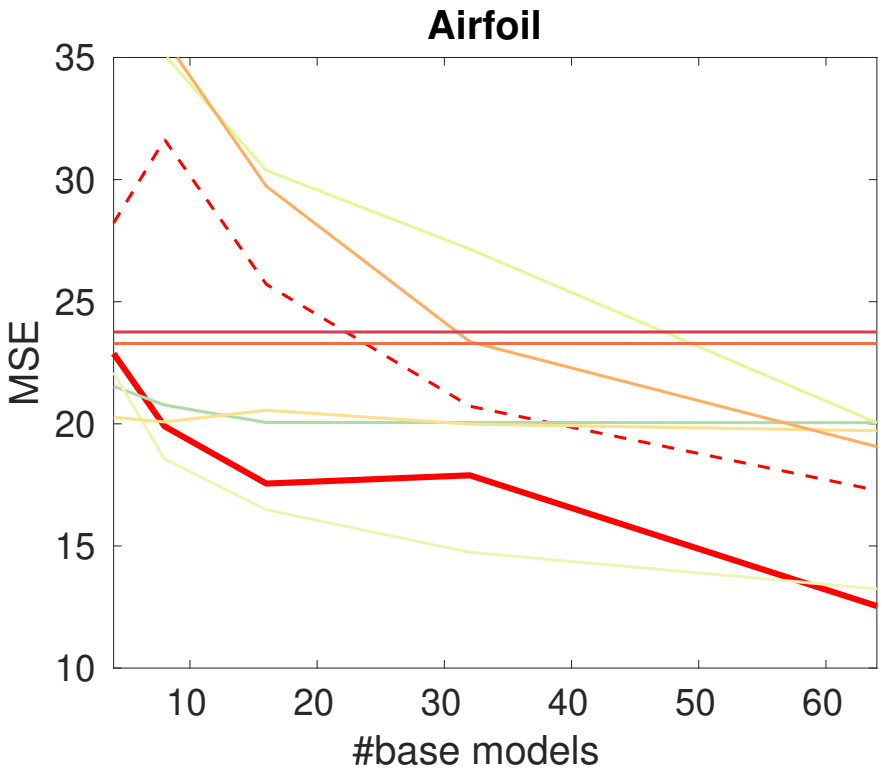


Figure 4.2: Horizontal axis represents the number of base models(e.g., centroids, RBFs, trees and etc.). The vertical axis represents the mean squared error. SRNN-Reg is compared with various similar models on the dataset of CPU.

train error



test error

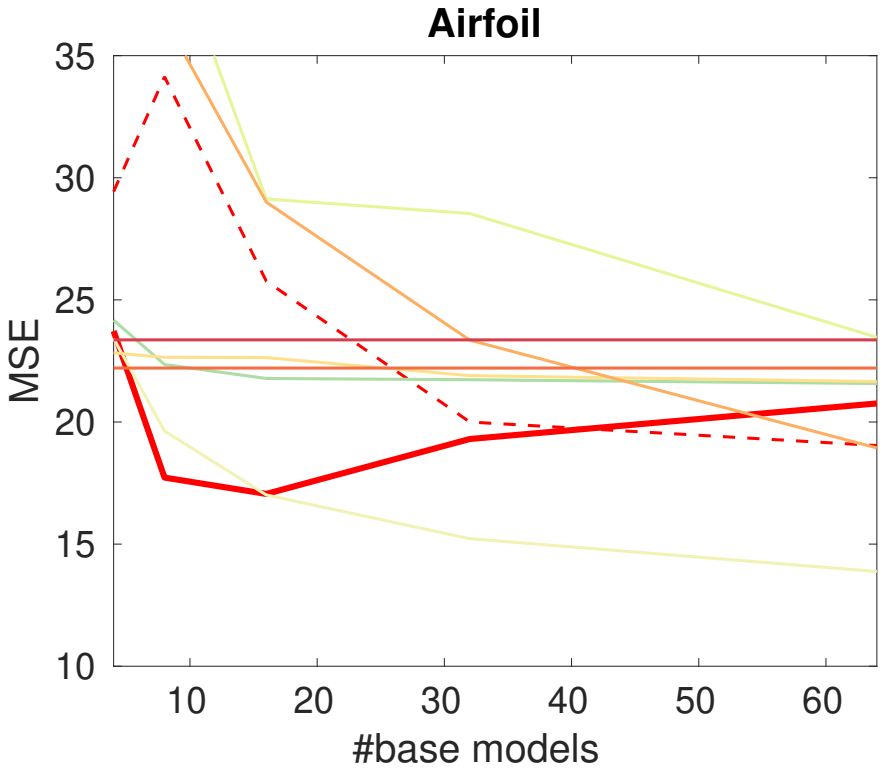
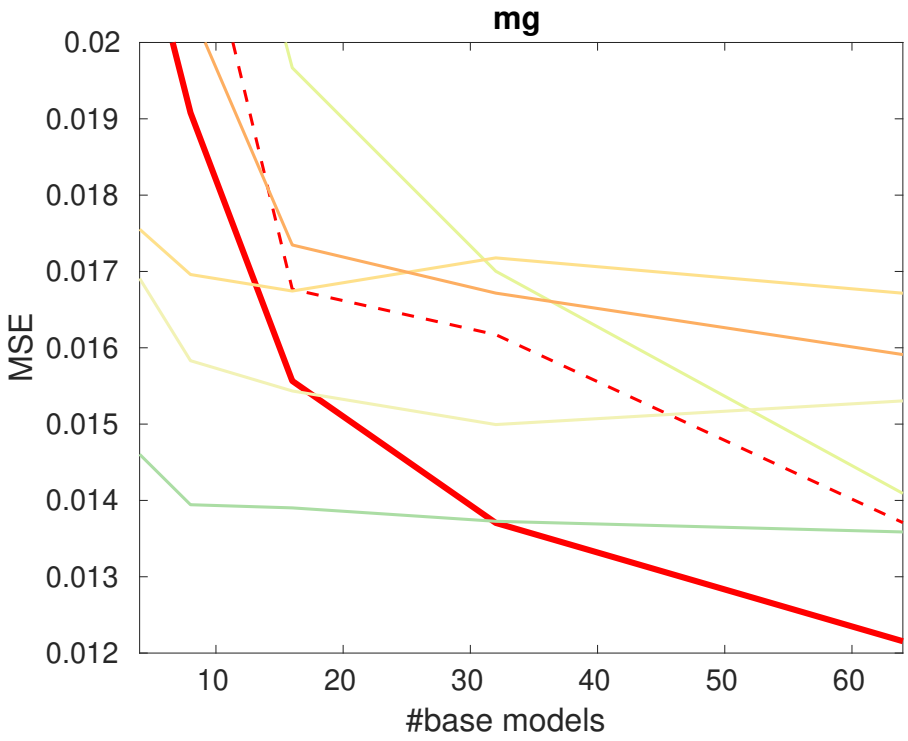


Figure 4.3: Horizontal axis represents the number of base models(e.g., centroids, RBFs, trees and etc.). The vertical axis represents the mean squared error. SRNN-Reg is compared with various similar models on the dataset of Airfoil.

train error



test error

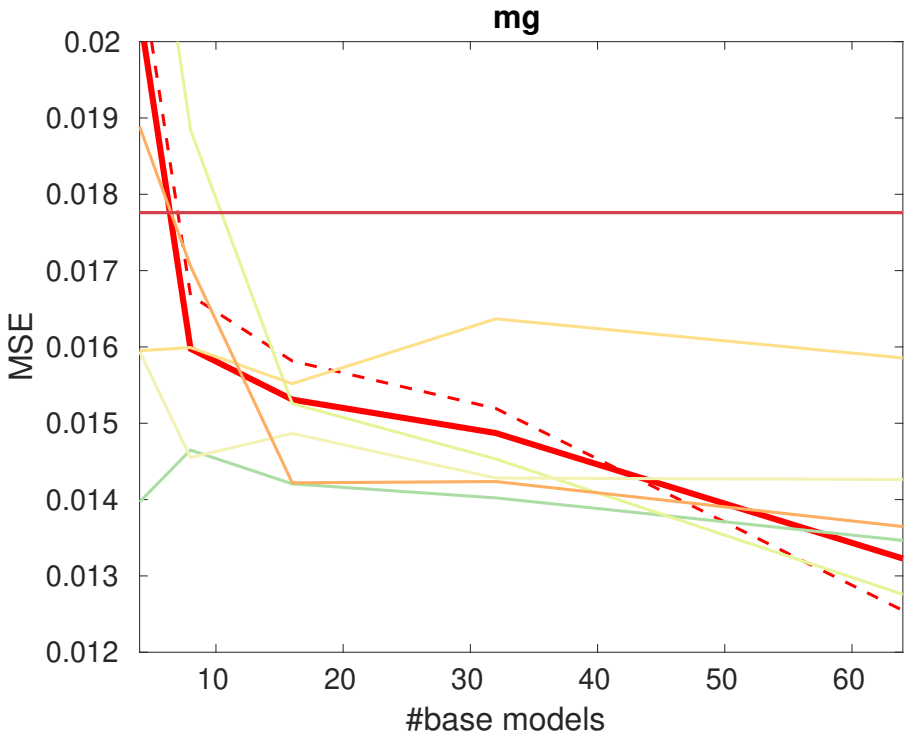
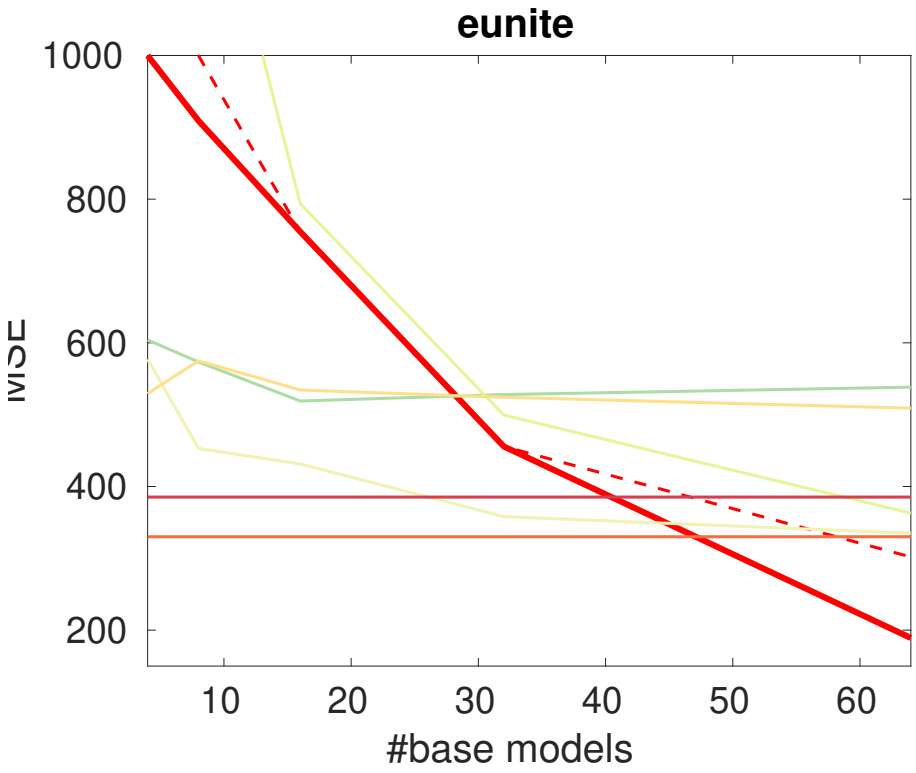


Figure 4.4: Horizontal axis represents the number of base models(e.g., centroids, RBFs, trees and etc.). The vertical axis represents the mean squared error. SRNN-Reg is compared with various similar models on the dataset of mg.

train error



test error

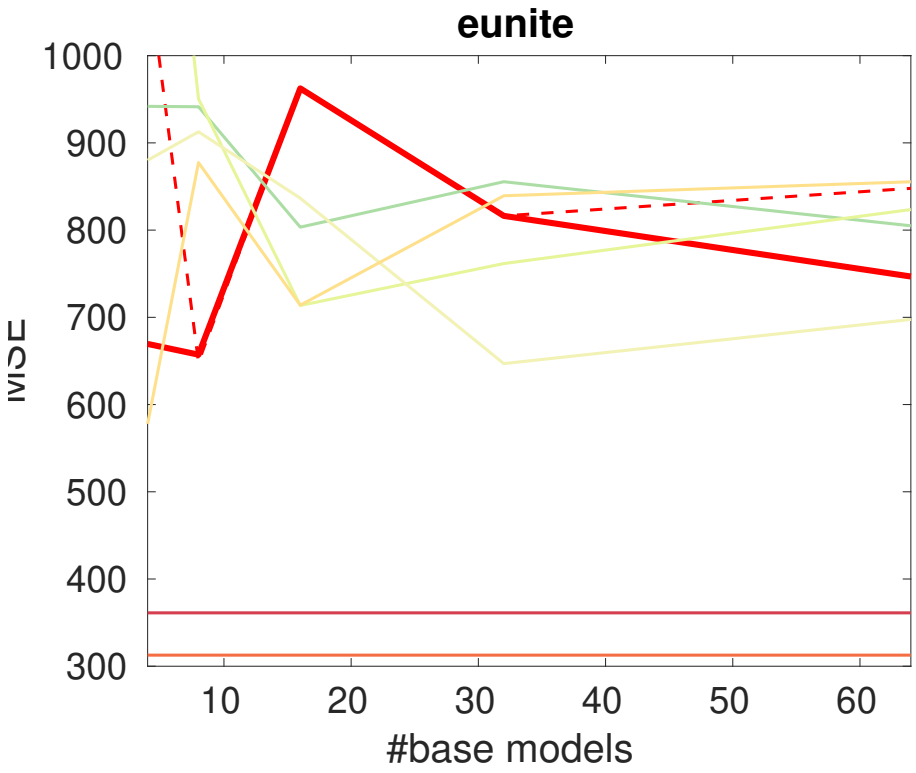


Figure 4.5: Horizontal axis represents the number of base models(e.g., centroids, RBFs, trees and etc.). The vertical axis represents the mean squared error. SRNN-Reg is compared with various similar models on the dataset of eunite.

(same number of RBFs as the centroids in other models) and the training linear regression over the outputs of RBFs. The width of RBFs were selected by cross-validation. The penalty coefficient of ridgeCV was selected by cross-validation.

As can be observed from figures 4.1, 4.2, 4.3, 4.4, and 4.5, the SRNN-Reg was able to achieve better or comparable train and test errors to other models.

4.4 Summary

In this chapter, as per our search, we have proposed the first regression synthetic reduced nearest neighbor. We also proposed a novel EM-based algorithm for optimizing the model. A new initialization technique was introduced in this chapter. The consistency of the algorithm was proved and its properties were explored. We showed that the algorithm is computationally efficient and can converge to a local optimum in the sense than no move can improve the train error of the model any further. The approach is inspired by the same type of EM algorithm used for K-means. The update step was an NP-hard weighted binary classification problem. The optimum of such problems are typically approximated using a surrogate objective function, such as hinge loss in SVM for 0-1 binary classification problem. Therefore, we approximated the solution to the update step through a novel surrogate objective function. Further, we analyzed the relation of the update step with SVM. Experimentally, we showed that the SRNN-Reg performs better or competitive to the other similar models in the literature, such as ensembles and centroid based models.

Chapter 5

Conclusion

In this dissertation, we proposed a novel expectation maximization algorithm that is capable of optimizing piecewise-constant models and their applications. We showed that the algorithm is computationally efficient in all cases studied in this dissertation. Additionally, the proposed algorithm achieves a local optimum of the original objective function. Compared to other algorithms that are applied to the models studied here, the algorithm is interpretable, easy to understand, and does not act like a black box optimizer (like optimization of neural nets through gradient-based optimizers). The algorithm mostly acts like the EM algorithm of the K-means that iterates over two steps: 1- the assignment where samples are assigned to each partition and label of each partition is determined by the set of samples in the partition. 2- the expectation step where the boundaries of the partitions are modified through some surrogate objective function. Experimentally, all the studied models have shown promising accuracies and performances when compared with the other similar state-of-the-art algorithms. The models presented in this dissertation can have direct applications in various studies, such as epidemiological and medical studies. Finally, the studied algorithms filled several gaps in the literature that were not studied before this dissertation, such as the regression SRNN which never existed, adversarial attack based on SRNN which is the first multi-class adversarial label flipping attack, and Robust SRNN which is the first defense technique against multi-class attack.

In the second chapter, the first EM-based optimization was proposed for the

SRNN. Typically, other existing algorithms for optimizing SRNN rely on smoothening the inference of the SRNN model and apply a gradient based optimization. However, such approaches are sub-optimal, black-box and do not guarantee minimization of the original algorithm. The proposed algorithm in second chapter could directly minimize the 0-1 loss of the objective function and had convergence guarantee.

In the third chapter, the first multi-class adversarial label-poisoning attack based on SRNN was proposed. As per our knowledge, this is the first multi-class adversarial label-poisoning. Other existing attacks are only applicable to binary classification tasks. Further, a novel robust SRNN model based on EM-algorithm was proposed that was capable of finding malicious samples with a high precision.

In the fourth chapter, the first synthetic reduced nearest neighbor for task of regression was proposed. Its properties were studied and it was shown that the model was consistent. The relation between the update step and SVM-like problem was explored. Finally, it was shown that the SRNN-Reg was able to achieve similar or better than the other similar state-of-the-art regression models.

Bibliography

[1] Joaquín Abellán and Serafín Moral. Building classification trees using the total uncertainty criterion. *International Journal of Intelligent Systems*, 18(12):1215–1225, 2003.

[2] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*, pages 459–468. IEEE, 2006.

[3] Fabrizio Angiulli. Fast condensed nearest neighbor rule. In *Proceedings of the 22nd international conference on Machine learning*, pages 25–32, 2005.

[4] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[5] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25, 2006.

[6] Richard J Beckman and R Dennis Cook. Outlier. s. *Technometrics*, 25(2):119–149, 1983.

[7] Jean-Michel Begon, Arnaud Joly, and Pierre Geurts. Globally induced forest: A prepruning compression scheme. In *International Conference on Machine Learning*, pages 420–428, 2017.

[8] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275. Springer, 2017.

[9] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):509–522, 2002.

[10] Jacqueline K Benedetti. On the nonparametric estimation of regression functions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(2):248–253, 1977.

[11] Sergio Bermejo and Joan Cabestany. Adaptive soft k-nearest-neighbor classifiers. *Pattern recognition*, 32(12):2077–2079, 1999.

[12] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[13] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104, 2006.

[14] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *Asian conference on machine learning*, pages 97–112, 2011.

[15] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[16] Battista Biggio, Konrad Rieck, Davide Ariu, Christian Wressnegger, Igino Corona, Giorgio Giacinto, and Fabio Roli. Poisoning behavioral malware clustering. In *Proceedings of the 2014 workshop on artificial intelligent and security workshop*, pages 27–36, 2014.

[17] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[18] Charles Bouveyron and Stéphane Girard. Robust supervised classification with mixture models: Learning from data with uncertain labels. *Pattern Recognition*, 42(11):2649–2658, 2009.

[19] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[20] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[21] Leo J. Breiman, Jerome H. Friedman, R. A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, Calif., 1984.

[22] Peter Bühlmann and Bin Yu. Boosting with the l 2 loss: regression and classification. *Journal of the American Statistical Association*, 98(462):324–339, 2003.

[23] Miguel Á. Carreira-Perpiñán and Pooya Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. In *Advances in Neural Information Processing Systems*, pages 1219–1229, 2018.

[24] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[25] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[26] Chin-Liang Chang. Finding prototypes for nearest neighbor classifiers. *IEEE Transactions on Computers*, 100(11):1179–1184, 1974.

[27] Ricardo Cisneros, Hamed Gharibi, Marcela R Entwistle, Pooya Tavallali, Mukesh Singhal, and Donald Schweizer. Nitrogen dioxide and asthma emergency department visits in california, usa during cold season (november to february) of 2005 to 2015: A time-stratified case-crossover analysis. *Science of The Total Environment*, 754:142089, 2021.

[28] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[29] Antonio Criminisi and Jamie Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Advances in Computer Vision and Pattern Recognition. Springer-Verlag, 2013.

[30] Sanjoy Dasgupta and Kaushik Sinha. Randomized partition trees for exact nearest neighbor search. In *Conference on Learning Theory*, pages 317–337, 2013.

[31] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216, 2007.

[32] Mark De Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. Orthogonal range searching: Querying a database. *Computational Geometry: Algorithms and Applications*, pages 95–120, 2008.

[33] Glenn De'Ath. Multivariate regression trees: a new technique for modeling species–environment relationships. *Ecology*, 83(4):1105–1117, 2002.

[34] Christine Decaestecker. Finding prototypes for nearest neighbour classification by means of gradient descent and deterministic annealing. *Pattern Recognition*, 30(2):281–288, 1997.

[35] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[36] Thierry Denoeux. A neural network classifier based on dempster-shafer theory. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 30(2):131–150, 2000.

[37] Thierry Denoeux. A k-nearest neighbor classification rule based on dempster-shafer theory. In *Classic works of the Dempster-Shafer theory of belief functions*, pages 737–760. Springer, 2008.

[38] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[39] Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000.

[40] Harris Drucker, Chris JC Burges, Linda Kaufman, Alex Smola, Vladimir Vapnik, et al. Support vector regression machines. *Advances in neural information processing systems*, 9:155–161, 1997.

[41] David A Freedman. *Statistical models: theory and practice*. cambridge university press, 2009.

[42] Benoît Frénay, Ata Kabán, et al. A comprehensive introduction to label noise. In *ESANN*, 2014.

[43] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

[44] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[45] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. Analyzing and improving representations with the soft nearest neighbor loss. In *International Conference on Machine Learning*, pages 2012–2020, 2019.

[46] Anil Gaba and Robert L Winkler. Implications of errors in survey data: a bayesian model. *Management Science*, 38(7):913–925, 1992.

[47] Geoffrey Gates. The reduced nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 18(3):431–433, 1972.

[48] Carl Friedrich Gauss. *Theoria combinationis observationum erroribus minimis obnoxiae*, volume 2. H. Dieterich, 1823.

[49] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[50] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520, 2005.

[51] Franz Graf, Hans-Peter Kriegel, Sebastian Pölsterl, Matthias Schubert, and Alexander Cavallaro. Position prediction in ct volume scans. In *Proceedings of the 28th International Conference on Machine Learning (ICML) Workshop on Learning for Global Challenges, Bellevue, Washington, WA*, 2011.

[52] Franz Graf, Hans-Peter Kriegel, Matthias Schubert, Sebastian Pölsterl, and Alexander Cavallaro. 2d image registration in ct images using radial image descriptors. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 607–614. Springer, 2011.

[53] Marvin Gruber. *Improving efficiency by shrinkage: The James–stein and ridge regression estimators*. Routledge, 2017.

[54] Chirag Gupta, Arun Sai Suggala, Ankit Goyal, Harsha Vardhan Simhadri, Bhargavi Paranjape, Ashish Kumar, Saurabh Goyal, Raghavendra Udupa, Manik Varma, and Prateek Jain. Protonn: Compressed and accurate knn for resource-scarce devices. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1331–1340. JMLR. org, 2017.

[55] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. New York, NY: Springer, second edition, 2009.

[56] David Heath, Simon Kasif, and Steven Salzberg. Induction of oblique decision trees. In *IJCAI*, volume 1993, pages 1002–1007, 1993.

[57] Donald E Hilt and Donald W Seegrist. *Ridge, a computer program for calculating ridge regression estimates*, volume 236. Department of Agriculture, Forest Service, Northeastern Forest Experiment , 1977.

[58] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.

[59] Arthur E Hoerl and Robert W Kennard. Ridge regression: applications to nonorthogonal problems. *Technometrics*, 12(1):69–82, 1970.

[60] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[61] Elena Ikonomovska, João Gama, and Sašo Džeroski. Incremental multi-target model trees for data streams. In *Proceedings of the 2011 ACM symposium on applied computing*, pages 988–993. ACM, 2011.

[62] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, March 1994.

[63] Michael I Jordan and Lei Xu. Convergence results for the em approach to mixtures of experts architectures. *Neural networks*, 8(9):1409–1431, 1995.

[64] Lawrence Joseph, Theresa W Gyorkos, and Louis Coupal. Bayesian estimation of disease prevalence and the parameters of diagnostic tests in the absence of a gold standard. *American journal of epidemiology*, 141(3):263–272, 1995.

[65] Amitava Karmaker and Stephen Kwek. A boosting approach to remove class label noise 1. *International Journal of Hybrid Intelligent Systems*, 3(3):169–177, 2006.

[66] Michael Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, 1993.

[67] Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Supervised neural network modeling: an empirical investigation into learning from imbalanced data with labeling errors. *IEEE Transactions on Neural Networks*, 21(5):813–830, 2010.

[68] Marius Kloft and Pavel Laskov. Online anomaly detection under adversarial impact. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 405–412, 2010.

[69] Dragi Kocev, Sašo Džeroski, Matt D White, Graeme R Newell, and Peter Griffioen. Using single-and multi-target regression trees and ensembles to model a compound index of vegetation condition. *Ecological Modelling*, 220(8):1159–1168, 2009.

[70] Dragi Kocev, Celine Vens, Jan Struyf, and Sašo Džeroski. Tree ensembles for predicting structured outputs. *Pattern Recognition*, 46(3):817–833, 2013.

[71] Teuvo Kohonen. Improved versions of learning vector quantization. In *1990 ijcnn international joint conference on Neural networks*, pages 545–550. IEEE, 1990.

[72] Matej Kristan, Danijel Skocaj, and Aleš Leonardis. Incremental learning with gaussian mixture models. In *Computer Vision Winter Workshop*, pages 25–32, 2008.

[73] Matt Kusner, Stephen Tyree, Kilian Weinberger, and Kunal Agrawal. Stochastic neighbor compression. In *International Conference on Machine Learning*, pages 622–630, 2014.

[74] Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.

[75] Adrien Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes: avec un supplément contenant divers perfectionnemens de ces méthodes et leur application aux deux comètes de 1805*. Courcier, 1806.

[76] Jurica Levatić, Michelangelo Ceci, Dragi Kocev, and Sašo Džeroski. Semi-supervised learning for multi-target regression. In *International Workshop on New Frontiers in Mining Complex Patterns*, pages 3–18. Springer, 2014.

[77] Alexander Hanbo Li and Andrew Martin. Forest-type regression with general losses and robust forest. In *International Conference on Machine Learning*, pages 2091–2100, 2017.

[78] Guohua Liang, Xingquan Zhu, and Chengqi Zhang. An empirical study of bagging predictors for different learning algorithms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 2011.

[79] Cheng-Lin Liu and Masaki Nakagawa. Prototype learning algorithms for nearest neighbor classifier with application to handwritten character recognition. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318)*, pages 378–381. IEEE, 1999.

[80] Ting Liu, Andrew W Moore, Ke Yang, and Alexander G Gray. An investigation of practical approximate nearest neighbor algorithms. In *Advances in neural information processing systems*, pages 825–832, 2005.

[81] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[82] Naresh Manwani and PS Sastry. Noise tolerance under risk minimization. *IEEE transactions on cybernetics*, 43(3):1146–1151, 2013.

[83] Charles Mathy, Nate Derbinsky, José Bento, Jonathan Rosenthal, and Jonathan Yedidia. The boundary forest algorithm for online supervised and unsupervised learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[84] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, pages 2871–2877, 2015.

[85] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[86] Mehran Mozaffari-Kermani, Susmita Sur-Kolay, Anand Raghunathan, and Niraj K Jha. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE journal of biomedical and health informatics*, 19(6):1893–1905, 2014.

[87] Sreerama K Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of artificial intelligence research*, 2:1–32, 1994.

[88] Blaine Nelson and Anthony D Joseph. Bounding an attacks complexity for a simple learning model. In *Proc. of the First Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML), Saint-Malo, France*, page 111, 2006.

[89] Tan Nguyen and Scott Sanner. Algorithms for direct 0–1 loss optimization in binary classification. In *International Conference on Machine Learning*, pages 1085–1093, 2013.

[90] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[91] Mohammad Norouzi, Maxwell D Collins, David J Fleet, and Pushmeet Kohli. Co2 forest: Improved random forest by continuous optimization of oblique splits. *arXiv preprint arXiv:1506.06155*, 2015.

[92] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[93] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414. IEEE, 2018.

[94] Jaehyun Park and Stephen Boyd. General heuristics for nonconvex quadratically constrained quadratic programming. *arXiv preprint arXiv:1703.07870*, 2017.

[95] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[96] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[97] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[98] Parikshit Ram and Alexander G Gray. Density estimation trees. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 627–635. ACM, 2011.

[99] Gunnar Rätsch, Takashi Onoda, and K-R Müller. Soft margins for adaboost. *Machine learning*, 42(3):287–320, 2001.

[100] Gunnar Rätsch, Takashi Onoda, and Klaus R Müller. Regularizing adaboost. In *Advances in neural information processing systems*, pages 564–570, 1999.

[101] Gunnar Rätsch, Bernhard Schölkopf, Alexander Johannes Smola, Sebastian Mika, Takashi Onoda, and Klaus-Robert Müller. Robust ensemble learning for data mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 341–344. Springer, 2000.

[102] Alvin C Rencher. *Methods of multivariate analysis*, volume 492. John Wiley & Sons, 2003.

[103] Sartaj Sahni. Computationally related problems. *SIAM Journal on computing*, 3(4):262–279, 1974.

[104] Fadil Santosa and William W Symes. Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1307–1330, 1986.

[105] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[106] Joseph L Schafer and John W Graham. Missing data: our view of the state of the art. *Psychological methods*, 7(2):147, 2002.

[107] HILARY L. SEAL. Studies in the History of Probability and Statistics. XV The historical development of the Gauss linear model. *Biometrika*, 54(1-2):1–24, 06 1967.

[108] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[109] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[110] Patrice Simard, Yann LeCun, and John S Denker. Efficient pattern recognition using a new transformation distance. In *Advances in neural information processing systems*, pages 50–58, 1993.

[111] Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

[112] Tim B Swartz, Yoel Haitovsky, Albert Vexler, and Tae Y Yang. Bayesian identifiability and misclassification in multinomial data. *Canadian Journal of Statistics*, 32(3):285–302, 2004.

[113] Yunpeng Tai. A survey of regression algorithms and connections with deep learning. *arXiv preprint arXiv:2104.12647*, 2021.

[114] Pooya Tavallali, Hamed Gharibi, Mukesh Singhal, Donald Schweizer, and Ricardo Cisneros. A multi-pollutant model: a method suitable for studying complex relationships in environmental epidemiology. *Air Quality, Atmosphere & Health*, 13:645–657, 2020.

[115] Pooya Tavallali, Peyman Tavallali, Mohammad Reza Khosravi, and Mukesh Singhal. Interpretable synthetic reduced nearest neighbor: An expectation maximization approach. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1921–1925. IEEE, 2020.

[116] Pooya Tavallali, Mehran Yazdi, and Mohammad R Khosravi. A systematic training procedure for viola-jones face detector in heterogeneous computing architecture. *Journal of Grid Computing*, pages 1–16, 2020.

[117] Pooya Tavallali, Mehran Yazdi, and Mohammad Reza Khosravi. Robust cascaded skin detector based on adaboost. *Multimedia Tools and Applications*, 78(2):2599–2620, 2019.

[118] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[119] Du Tran and Alexander Sorokin. Human activity recognition with metric learning. In *European conference on computer vision*, pages 548–561. Springer, 2008.

[120] John W Tukey et al. *Exploratory data analysis*, volume 2. Reading, Mass., 1977.

[121] J Uhlmann. Implementing metric trees to satisfy general proximity/similarity queries. In *Proc. Command and Control Symposium, Washington, DC*, 1991.

[122] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015.

[123] Sofie Verbaeten and Anneleen Van Assche. Ensemble methods for noise elimination in classification problems. In *International Workshop on Multiple classifier systems*, pages 317–325. Springer, 2003.

[124] B Walczak and DL Massart. Local modelling with radial basis function networks. *Chemometrics and Intelligent Laboratory Systems*, 50(2):179–198, 2000.

[125] Hansheng Wang, Guodong Li, and Chih-Ling Tsai. Regression coefficient and autoregressive order shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(1):63–78, 2007.

[126] Wenlin Wang, Changyou Chen, Wenlin Chen, Piyush Rai, and Lawrence Carin. Deep metric learning with data summarization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 777–794. Springer, 2016.

[127] Kilian Q Weinberger and Lawrence K Saul. Fast solvers and efficient implementations for distance metric learning. In *Proceedings of the 25th international conference on Machine learning*, pages 1160–1167, 2008.

[128] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[129] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *ECAI*, pages 870–875, 2012.

[130] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning*, pages 1689–1698, 2015.

[131] Xin Yan and Xiaogang Su. *Linear regression analysis: theory and computing*. World Scientific, 2009.

[132] Zoulficar Younes, Thierry Denœux, et al. Evidential multi-label classification approach to learning from data with imprecise labels. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 119–128. Springer, 2010.

[133] Kai Zhong, Ruiqi Guo, Sanjiv Kumar, Bowei Yan, David Simcha, and Inderjit Dhillon. Fast classification with binary prototypes. In *Artificial Intelligence and Statistics*, pages 1255–1263, 2017.

[134] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

[135] AV Zukhba. Np-completeness of the problem of prototype selection in the nearest neighbor method. *Pattern Recognition and Image Analysis*, 20(4):484–494, 2010.