OPTIMAL 0−1 LOSS CLASSIFICATION IN LINEAR, OVERLAPPING AND INTERPOLATION SETTINGS

by

REEM ABDULRHMAN ALANAZI

A thesis submitted to the University of Birmingham for the degree of

DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
September 2022

Dedication

To my sweetheart mum and my love dad.

Abstract

Classification problems represent a major subject in machine learning, and addressing them requires solving underlying optimisation problems. Optimising the 0–1 loss function is the natural pathway to address the classification problem. Because of the properties of 0–1 loss, which are that it is non-convex and non-differentiable, 0–1 loss is mathematically intractable and classified as non-deterministic polynomial-time hard (NP-hard). Consequently, the 0–1 loss function has been replaced by surrogate loss functions that can be optimised more efficiently, where their optimal solution is guaranteed with respect to these surrogate losses. At the same time, these functions may not provide the same optimal solution as the 0–1 loss function. Indeed, the loss function used during the empirical risk minimisation (ERM) is not the same loss function used in evaluation; the mismatch of the loss functions leads to an approximate solution that may not be an ideal solution for 0–1 loss. Thus, an additional source of error is produced because of this replacement.

In this thesis, optimising 0–1 loss directly is discussed from three perspectives. First, the consequences of replacing the 0–1 loss function with surrogate functions from the theoretical perspective for linearly separable data are outlined, showing that hinge loss is an exact minimiser for 0–1 loss under this setting. Second, in the case of overlapping, it is determined that the surrogate functions do not provide an exact solution. Therefore, we compared the actual time complexity of existing 0–1 loss optimisation algorithms based on the following second-order factors: dimensionality of data and amount of overlap between classes. In addition, a new heuristic 0–1 loss

optimisation algorithm was introduced that can optimise 0–1 risk exactly in practical way in $\mathcal{O}(\sqrt{D}\log(\frac{1}{\epsilon})N^3)$ polynomial complexity class. A new algorithm, called testing all pairs of points (T-PP), as well as its prioritised version, was introduced; the prioritised version has actual time complexity that is less than T-PP, and it is referred to as prioritising the search by support vectors (PS-SVs). T-PP and PS-SVs can terminate before the time limit, whereas the time taken by other direct 0–1 loss optimisation algorithms depends on second-order factors when the data size is fixed. Third, a new direction of ERM is discussed in a simple classification algorithm, which is the canonical K nearest neighbour (KNN) algorithm that still dominates by the conventional ERM. The interpolated local KNN is our new algorithm, which works on modern interpolation ERM view. The interpolated local KNN aims to fit the training points locally using a set of local K parameters that are determined by a pre-defined confidence score. The mechanism used in the interpolated local KNN helps to define the noise and overlap points and then isolate them from being used in predicting unseen points, whereas the isolated points have their own decision boundary that can be used to predict them correctly. Thus, we can have zero empirical risk and low generalisation risk compared with the canonical KNN.

These contributions show the possibility of optimising the 0–1 loss directly in polynomial time for linearly separable data using a surrogate loss function; for overlapping data, our new algorithms T-PP and PS-SVs can be used. Ultimately, we introduce a first interpolation version for the canonical KNN algorithm. This thesis focuses on the binary classification problem; our main focus in the future will be to extend this work to multiclass classification.

Acknowledgements

I would like to begin by expressing my great gratitude to Almighty God for granting me patience and determination to complete this thesis despite all the difficult circumstances I have encountered in my PhD journey.

First of all, I am eternally grateful to my parents, sisters and brothers for their unconditional love, prayers and constant support during all the years of the PhD programme.

I would also like to thank and acknowledge my supervisor, Dr Max Little, for his untiring, unlimited and constant support, advice and patience, which encouraged me to continue my research. It would have been impossible to complete this thesis without his continued motivation and endless support. I would also like to thank the wonderful PhD students in our research group for all the lovely discussions we have had.

My heartfelt appreciation and acknowledgement goes to King Faisal University for unconditional support during this scholarship and granting me this opportunity to complete this programme at the University of Birmingham. A special thanks to Dr Majed Al-Shammari, Dr Mohammed Alojail, and my internal supervisor, Dr Abdulaziz Albarrak, for their endless support, consideration and understanding of the circumstances I have encountered.

A very special thanks goes to my close friends, Manal Alharthi, Rana Algahtani and Murooj Alqhamdi, for their support, prayers and positive vibes.

Contents

1 Introduction **1**
- 1.1 Context of work . . . 1
- 1.2 Motivation . . . 3
- 1.3 Thesis outline . . . 5
- 1.4 Contributions and research questions . . . 6

2 Background review **9**
- 2.1 Foundations of mathematical optimisation . . . 9
- 2.2 Loss functions, empirical risk and empirical risk minimisation (ERM) . . . 14
- 2.3 The canonical 0–1 loss for classification and current optimisation algorithms . . . 16
  - 2.3.1 Combinatorial search method for 0–1 loss optimisation 18
    - 2.3.1.1 Brute-force search for 0–1 loss optimisation . 18
    - 2.3.1.2 Branch and bound for 0–1 loss optimisation . 18
    - 2.3.1.3 Prioritised combinatorial search . . . 20
    - 2.3.1.4 Combinatorial search approximation . . . 22
  - 2.3.2 Local derivative-free descent techniques . . . 22
- 2.4 Surrogate loss functions and associated optimisation algorithms 25
  - 2.4.1 Perceptron loss function . . . 26
    - 2.4.1.1 Perceptron algorithm . . . 27
  - 2.4.2 Hinge loss function . . . 29
    - 2.4.2.1 Primal optimisation algorithm . . . 32
  - 2.4.3 Logistic loss function . . . 33
  - 2.4.4 Smooth 0–1 loss . . . 35
    - 2.4.4.1 Hill climbing optimisation technique . . . 36
- 2.5 Competing views of ERM . . . 37
  - 2.5.1 ERM-based conventional wisdom view . . . 38
  - 2.5.2 ERM-based modern interpolation view . . . 40
- 2.6 Inadequacies of previous work . . . 44
- 2.7 Summary . . . 47

3 Exact canonical 0–1 loss on linearly separable data 48
3.1 Introduction . . . 49
3.2 The optimality of the canonical 0–1 loss in the surrogate losses 51
3.3 The hinge loss selects the 0–1 loss optimal linear model for linearly separable data . . . 53
3.4 The empirical risk $\hat{R}$ in practice . . . 55
3.5 Summary . . . 56

4 Direct 0–1 risk minimisation on overlapping data 57
4.1 Introduction . . . 58
4.2 Testing all pairs of points (T-PP) algorithm . . . 59
4.3 Prioritising the search for the optimal canonical 0–1 loss . . . 68
4.3.1 Initial solution based on hinge risk . . . 68
4.3.2 The heuristic strategy . . . 69
4.3.2.1 Prioritising the search by support vectors (PSSVs) . . . 70
4.3.3 Theoretical time complexity . . . 77
4.4 Experiments . . . 79
4.4.1 Experiments parameter setup . . . 79
4.4.2 Results and discussion . . . 81
4.5 Summary . . . 86

5 The interpolated local KNN algorithm 88
5.1 Introduction . . . 89
5.2 The canonical KNN in the classical U-shaped and interpolation regimes . . . 91
5.2.1 The effect of noise and overlap on the interpolation regime . . . 92
5.2.2 Challenges of interpolating the KNN algorithm . . . 96
5.2.3 The relation between the confidence level and interpolation . . . 98
5.3 The interpolated local KNN classifier . . . 99
5.4 Experiments . . . 104
5.4.1 Experiments setting . . . 104
5.4.2 Results and discussion . . . 105
5.5 The canonical KNN and the interpolated local KNN classifiers in image classification . . . 111
5.5.1 Benchmark image datasets and the experimental setting 113
5.5.2 Results and discussion . . . 115
5.6 Summary . . . 122

6 Conclusion and future work **123**
6.1 Summary of work 123
6.2 Future research directions 126

List of Figures

2.1.1 Difference between convex and strictly convex functions. Figure 2.1.1a the global optimal solution is a set of local minimum solutions which leads to making a solution not unique. Figure 2.1.1b there is only one local minimum solution; therefore, the global solution is unique. 11
2.1.2 Difference between convex and non-convex functions. Figure 2.1.2a Convex function with one critical point, and Figure 2.1.2b non-convex function, which has several critical points and the second derivative of these local points is positive in local minimum points and negative in local maximum points. 11
2.1.3 Convexity condition: the solid green line joining the two points $p_i = (x_i, f(x_i))$ and $p_j = (x_j, f(x_j))$ lies entirely inside the curve, and never comes out of any part of the curve. For any input point x_s between the two selected x_i and x_j. $f(x_s) = f(\alpha x_i + (1-\alpha) x_j)$ should be less than or equal to the sum of the separated outputs of $\alpha f(x_i)$ and $(1-\alpha) f(x_j)$. 12
2.1.4 Different types of functions, where some gradients of f do not exist, such as in 2.1.4a are undefined and have infinite slope like 2.1.4b or contain cusps or corners like 2.1.4c and 2.1.4d; the slope at the corner point (or cusp point) is not unique. . . 13
2.3.1 The 0–1 loss curve. The gradient of 0–1 loss is zero everywhere, and we have an infinite slope from 1 to 0. 17
2.3.2 Plot showing the main concept of PCS. Solid line is the decision boundary that optimally separates two classes, and dashed line is the decision boundary obtained from PCS, which goes through two points and gives the same prediction as the optimal one. 21
2.4.1 Loss value that will be allocated to any misclassified point for $l_{\text{perceptron}}$. Compared to l_{01}, $l_{\text{perceptron}}$ assigns a real loss > 0 for the misclassified point. 26

2.4.2 Difference between differentiable point in Figure 2.4.2a and non-differentiable point in Figure 2.4.2b. $y\hat{y} = 0$ has several tangent lines (shown in blue), and each tangent line has a different slope. 28
2.4.3 The loss function curves for l_{hinge} and l_{01}. l_{hinge} is an upper bound for l_{01}. 30
2.4.4 Logistic loss and 0–1 loss curves. The distant correctly classified points contribute to the total loss. 35
2.4.5 2.4.5a: degree of smoothness with respect to the k parameter, where a small value yields a smooth curve and a large k creates a sharp curve. 2.4.5b: Loss value that will be assigned to the point based on the prediction margin $y\hat{y}$. 37
2.5.1 Training and test errors with respect to the conventional view regime. The test error usually has a U-shaped curve and training error always has a decreasing curve, where the highest risk lies with underfitting and lowest risk lies with overfitting. Excessively increasing the model complexity leads to overfitting, whereas excessively decreasing the model complexity causes underfitting. Managing the model complexity is a very difficult task. The sweet spot shows the optimal balance between underfitting and overfitting. 40
2.5.2 Training and test errors with respect to the modern interpolation regime. The test error decreases in the modern regime after reaching an overfitting peak (interpolation threshold) and continues to decrease until the test error in the modern regime is less than that in the classical (conventional) regime. 44

3.4.1 shows the curves of surrogate loss ϕ and 0–1 loss. The horizontal dotted line shows the optimal $f^\star$ that minimise the $\hat{R}_l$ with respect to l_ϕ. The optimal $f_{01}^\star$ is not unique based on l_{01}, and l_{hinge} finds $f_{hinge}^\star$that exactly achieves the same the optimal solution. Optimal solution is the solid black line. . . 56

4.2.1 Main concept of the T-PP algorithm. 4.2.1a All these decision boundaries pass between the optimal pair of points $p^\star$ (coloured points with black circles), which not all have the same 0–1 risk. 4.2.1b The optimal function $f_{01}^\star$ can be found under the constraint that no points closer more than the $p^\star$ to the decision boundary of $f_{01}^\star$. The optimal and T-PP decision boundaries have the same 0–1 risk. 63

4.2.2 The steps of T-PP algorithm for finding the optimal function $f^\star$ between the optimal pair of points $p^\star$. 4.2.2a The initial decision boundary is fitted between only two points. 4.2.2b The initial decision boundary is modified frequently through an incremental procedure; a single point is added per time, and then the decision boundary is modified accordingly. 4.2.2c $f^\star$ is obtained by fitting the decision boundary between $p^\star$ and a subset of points. 4.2.2d The T-PP decision boundary and optimal decision boundary belong to the same equivalence class. 64

4.2.3 The second incremental procedure of T-PP algorithm aims to prioritise the search for finding the best possible function or the optimal one even when the pair of points are not close and have a large margin, as shown in 4.2.3a. 4.2.3b All points in the margin cannot satisfy the constraints of the first incremental process. Thus, we need to adjust the T-PP decision boundary to find the best possible decision boundary, not necessarily the optimal one. To do this, we keep the assignments of correctly classified points as in 4.2.3c, and only need to adjust the T-PP decision boundary incrementally for misclassified points starting from the nearest point to the T-PP decision boundary. We may obtain an optimal solution, as in 4.2.3d or a new decision boundary that is better than the previous T-PP decision boundary. 65

4.3.1 The pair of SVs (coloured points with black circles) and non-SVs (coloured points) are used to fit the decision boundary between the pair. Therefore, all points inside the label region (black coloured points) are assigned simultaneously without using incremental procedure, this helps to speed up the assignment process. Panels (a) and (b) show different selections of pairs of SVs points. 75

4.3.2 Time complexity of different D values. The time increases as the value of D increases. 78

4.3.3 Number of computations when the number of points is increased for BnB, PCS,CSA, and T-PP on $2D$ data. 79

4.4.1 Computational time for minimising the $\hat{R}_{01}$ with gradually increasing overlap rates for $2D$, $4D$ and $6D$ Gaussian distributed data. 83

4.4.2 Computational time for minimising the $\hat{R}_{01}$ with gradually increasing overlap rates for $8D$, $10D$, and $12D$ Gaussian distributed data. 84

4.4.3 Computational time for minimising the $\hat{R}_{01}$ with gradually increasing overlap rates for $14D$, and $16D$ Gaussian distributed data. 85

5.2.1 Snapshots of the synthetic datasets. The data have the unique label $M - i$ for the purpose of referencing. 94
5.2.2 Snapshots of the clean, noisy and overlapping data. 94
5.2.3 Average training and testing error reported on 10 random splits of the data. The data used are free from noise and overlap. The complexity of the model is measured as $\frac{1}{K}$; it decreases when the complexity parameter K increases. As clearly shown, both curves have the same trend, and the risk of misclassification increases as the K parameter increases. . 95
5.2.4 Average training and testing error reported on 10 random splits of the data; the training set was injected with 5% noise. The test error curve is U-shaped. It decreases when K increases, reaching a minimum test error, and then increases again. 95
5.2.5 Average training and test error reported on 10 random splits of the data; 10% of the training points overlap. The overlap was done on the boundary points. The classic U-shaped curve clearly appears. 95
5.2.6 Relationship between increasing the global K parameter and the interpolation, as in 5.2.6b, where the interpolation cannot be achieved if the level of confidence associated with each point is less than or equal to 0.50, as shown in 5.2.6c. To interpolate the data for $K > 1$, the level of confidence of each point should be >50, and the relationship is shown in 5.2.6d. A smaller K value results a larger interpolation score, therefore in order to achieve a large interpolation score the confidence level of all training points should be large. These relations have been computed for 5.2.6a, which has properties that represent most real-world data. 100
5.4.1 Double descent curve of the canonical KNN and interpolated local KNN on breast cancer, ILPD and CMC data. The 1NN algorithm is an intersection point. The interpolation regime starts when the training error reaches zero, and the classic U-shaped regime appears when the training error > 0. The complexity of KNN classifier is best measured as $\frac{1}{K}$, so that as K increases, the complexity of the boundary decreases. . . 106

5.4.2 Double descent curve of employee, spambase and pulsar data connects the classic U-shaped regime (showing the performance of the canonical KNN) and the interpolation regime (showing the performance of the interpolated local KNN) via the 1NN algorithm. 107
5.4.3 Double descent curve of phishing and magic data. The same set of K parameters was used in the canonical KNN and interpolated local KNN. The training and testing errors show the performance of the two algorithms, one based on the classic view of bias–variance trade-off and the other based on the interpolation view of fitting the training set exactly. 108
5.5.1 Small points for the image data used in the experiment. 115
5.5.2 Double descent curve of a set of image data. 117
5.5.3 Training and testing error curves of the interpolated local KNN across different confidence scores for MNIST data. . . . 118
5.5.4 Each mini-figure represents the performance of the interpolated local KNN at different confidence scores for CIFAR-10. 119
5.5.5 Training and testing results of the interpolated local KNN across different values of the K parameter and a fixed confidence score per mini-figure for Fashion-MNIST. 120
5.5.6 Generalisation risk according to changing the confidence score for the image datasets; the lower the degree of confidence, the lower the risk rate. 121

List of Tables

4.1 Information on datasets used in the experiment. 81
4.2 $\hat{R}_{01}$ and the actual time taken by the BnB, PCS, CSA, T-PP, and PS-SVs algorithms until termination or reaching the time limit. Bold text indicates that the algorithms obtain the same $\hat{R}_{01}$. The underlined text indicates that the algorithms reach the time limit. 86

5.1 General information for the real datasets used in the experiment. 105
5.2 Training and test errors of the canonical KNN and interpolated local KNN. There is a significant difference in performance if the P-value ≤ 0.05. Bold text indicates that the algorithm outperforms other algorithm, with a statistically significant difference in performance. 105

List of Notations

$\mathcal{M}$	Population space
$\mathcal{F}$	Function space
$\mathcal{X}$	Input space
$\mathcal{Y}$	Output space
$\mathbb{R}$	The real space
$\mathbb{R}_{\geq 0}$	The non-negative real space
x_n	Independent variables
y_n	Dependent variable
M	Training set
N	Number of training points
D	Number of dimensions
$f_l^\star$	Optimal learning function with respect to l
$\hat{R}_l$	Empirical risk with respect to l
α	Learning rate
f	Learning function
R_l	Generalization risk with respect to l
l	Loss function
ϵ	A small positive real value
L	Number of labels
$\hat{y}$	Prediction value
R	True risk
w^t	Classifier weights
b	Classifier bias

η	Conditional probability
$\nabla f(x)$	Gradient of the function f at x
v	Weight score
$p^\star$	Optimal pair of points
$R_l^\star$	Optimal risk with respect to l
ϕ	Surrogate function
p	A single poin
J	A subset of the training points
$y\hat{y}$	Prediction margin
ψ	Radius of the confidence area
C_p	Central point
$C(l_\phi, \eta)$	Classification-calibrated loss function
ζ	Slack variable
λ	Regularization parameter
$\hat{p}$	Predicted probability
t	A threshold value for prediction
k	Smoothing parameter
K	Number of nearest neighbor points
$\mathcal{O}$	Time complexity of an algorithm
q	A set of subgradients
P_D	A subset of D points
S	A hyperparameter of CSA algorithm
h	Displacement value
$\mathcal{P}$	True probability distribution

Chapter 1

Introduction

1.1 Context of work

Machine learning focuses on building algorithms to emulate human intelligence by learning from a given dataset. One of the main tasks of machine learning is classification, defined as grouping a set of objects under a particular category according to a given label: each object x_n is associated with a label y_n. Classification is used in real-life applications, such as customer behaviour prediction, document classification, spam filtering and credit card fraud detection. It contributes to solving a wide range of problems that humans cannot solve due to the large volumes of data that need to be analysed. Moreover, some business operations depend on making immediate decisions in response to specific or unusual behaviour. For instance, financial transactions need to be verified to avoid fraudulent activity based on a customer's transaction history. Thus, classification is of critical importance for numerous applications.

Improving classification accuracy requires solving an optimisation problem known as empirical risk minimisation (ERM). ERM aims to find the optimal function $f^\star$ that can classify given points, $M = \{(x_1, y_1), ..., (x_n, y_n)\}^N$ with the lowest empirical risk functional $\hat{R}_l(f^\star, M)$. The canonical 0–1 loss function is the original classification loss function, which counts the number of misclassified points.

The difficulty of minimising the empirical risk depends on the properties of the loss function [1, 2]. The properties of the 0–1 loss function, which are non-convex and non-differentiable, make the optimisation of the original loss function difficult to handle mathematically [3, 4, 5]. In binary classification, to find the optimal solution, we conduct a search in a solution space of size 2^N, for N points [5, 6, 7]. In fact, the intelligent heuristic search strategy can be used to optimise the current solution iteratively and cut off possible solutions that have worse objective values than the current one, which helps to reduce the search space significantly [8]. Despite, the search process can be accelerated, method-based heuristic searches are still not a practical solution in some cases [9].

A possible optimisation solution is to replace the 0–1 loss function with a convex loss function [10, 11, 12]. This is usually referred to as convex relaxation, turns a non-convex problem into a convex one and finds an approximate solution [13, 14]. Convex relaxation has the advantage of solving a problem in polynomial time [11, 13]. However, it is a modification of the original problem; therefore, the solution provided may not be exact solution.

Furthermore, convex relaxation introduces issues that do not exist in the original problem, such as sensitivity to outliers, and imbalance label

problem [15, 16, 17, 18, 19, 20], which are common issues in classification. In fact, convex loss functions provide a real loss value for any misclassified point depends on how far the point from the decision boundary. The loss value increases as the distance of the misclassified point increases. This makes the resulting decision boundary dominated by outliers. In addition, the classified points close to the decision boundary contribute to the total sum of the loss values, therefore if the density of points for any class varies greatly around the decision boundary, then the decision boundary obtained from minimizing the convex function is pushed toward the minority class in order to reduce the total loss values. In these issues, convex loss functions provide an approximate solution, not an exact solution. Moreover, although convex relaxation has received considerable attention in the machine learning field, few studies have attempted to optimise the original classification loss directly [21, 22, 23, 24].

1.2 Motivation

This thesis aims to contribute to optimising the original classification loss function. The difficulty of optimising the canonical 0–1 loss function means that few tools are available for minimising it directly. To understand the consequences of replacing the canonical 0–1 loss function with a surrogate loss function, we need to measure the difference between minimising the 0–1 loss directly and indirectly by comparing the empirical 0–1 risks of the optimal functions resulted from minimising the surrogate loss function and the 0–1 loss function. If both functions have the same 0–1 risk, the surrogate

loss function provides an exact solution for the 0–1 loss; if not, the surrogate loss function provides an approximate solution not exact. This thesis aims to provide a theoretical and empirical evidence for the use of the surrogate loss function in the ERM that can be easily verified in practice.

Convex loss functions cannot minimise the 0–1 risk in the presence of noise and overlapping data [22]. In such cases, it is necessary to optimise the canonical 0–1 loss directly. The existing algorithms for minimising the 0–1 risk are quite limited, and their theoretical and actual time complexities have not been compared. Such a comparison can determine which algorithm is characterised by less time complexity and is thus suitable for minimising the 0–1 loss directly. Moreover, a polynomial time algorithm is needed to avoid the errors introduced by replacing the original 0–1 loss with surrogate loss functions. As has been shown in the previous studies, surrogate loss functions provide an approximate solution in the presence of outliers, overlap and imbalance label [15, 16, 17, 18, 19, 20], thus an additional source of error is produced due to this replacement.

It has recently been suggested that minimising the empirical risk to exactly zero does not negatively affect generalisation performance. This view has long been marginalised due to the dominant conventional ERM paradigm in the machine learning community. The canonical K nearest neighbour (KNN) algorithm still works under the conventional view of ERM. Therefore, understanding the performance of this simple algorithm under the modern view of ERM could reveal the reasons behind the success of interpolation classifiers.

1.3 Thesis outline

This thesis consists of six chapters. Chapter 2 describes the basic properties of general loss functions and explains how they can be optimised mathematically. It then defines ERM, which drives the classification optimisation process. Subsequently, it presents the current exact 0–1 loss minimisation algorithms and the common classification algorithms that optimise surrogate loss functions. Finally, it discusses the conventional and modern ERM views. Chapter 3 investigates the optimality of the canonical 0–1 loss function indirectly in linearly separable data by optimising surrogate loss functions. Chapter 4 introduces the new heuristic direct 0–1 loss minimisation algorithms: T-PP and PS-SVs which work in polynomial time complexity. It then discusses the theoretical and actual time complexities of exact 0–1 loss minimisation algorithms and determines the best and worst time complexities based on second-order factors using a set of overlapping synthetic and real datasets. Chapter 5 investigates the performance of the canonical KNN algorithm under the conventional and modern ERM views. It then introduces a new interpolated local KNN algorithm. Subsequently, it compares the performance of the canonical KNN algorithm with that of the interpolated local KNN algorithm using a double descent curve. Finally, Chapter 6 summarizes this work in terms of main findings and contributions, and suggests future research directions.

1.4 Contributions and research questions

The main contributions of this thesis and the related research questions are as follows:

- Chapter 3

  Contributions

  - Studying the consequences of replacing the 0–1 loss function with surrogate loss functions in linearly separable data. The existing theoretical proof is not practically applicable. Therefore, we simplify the relationship between optimising the 0–1 loss function and surrogate loss functions by comparing the optimal function produced by each using the same measure.
  - Proving that minimising the hinge risk is equivalent to minimising the 0–1 risk in linearly separable data.

  Related research questions

  1. Does optimising a surrogate loss function instead of the canonical 0–1 loss function affect on choosing the optimal empirical function for linearly separable data?
  2. Can optimising the empirical risk for the hinge loss provide the same empirical risk for 0–1 loss for linearly separable data?

- Chapter 4

  Contributions

  - Introducing a new heuristic direct 0–1 loss minimisation algorithm based on linear function for overlapping case that works in polynomial complexity class $\mathcal{O}(\sqrt{D}\log(\frac{1}{\epsilon})N^3)$, called testing all pairs of points (T-PP) algorithm.
  - Reducing the actual time complexity of T-PP by introducing a prioritised version, called prioritising search by support vectors (PS-SVs). The search space of T-PP is reduced based on minimising the hinge risk, which is the upper bound of the 0–1 risk.
  - Comparing the best and worst actual time complexities of the existing direct 0–1 loss minimisation algorithms based on second-order factors.
  - Determining the time complexity class of the prioritised combinatorial search (PCS) algorithm.

  Related research questions

  1. What are the theoretical and actual time complexities of the existing direct 0–1 loss minimisation algorithms?
  2. Can the optimal function for the 0–1 risk be found by enumerating all pairs of points?

- Chapter 5

  Contributions

  - Studying the performance of the canonical KNN in interpolation and U-shaped regimes.
  - Introducing a first new KNN algorithm that works under the modern ERM view, called the interpolated local KNN algorithm.
  - Investigating the performance of the canonical and interpolated local KNN algorithms on image datasets.

  Related research questions

  1. Can the interpolation 1 nearest neighbour (1NN) classifier achieve a lower generalisation risk than KNN for $K > 1$?
  2. How can the canonical KNN be interpolated for $K > 1$?
  3. Does interpolating image data without feature extraction/reduction cause overfitting and then, bad generalization?

Chapter 2

Background review

This thesis focuses on optimising 0–1 loss in a classification problem. This chapter provides an overview of how 0–1 loss is optimised in two different ways, directly or indirectly. Direct optimisation uses the 0–1 loss function for minimising the classification error, whereas indirect optimisation uses surrogate loss functions as a proxy for minimising the classification error. Following that, an overview of the two main views of empirical risk minimisation, commonly named as a conventional wisdom view and modern interpolation view, is presented.

2.1 Foundations of mathematical optimisation

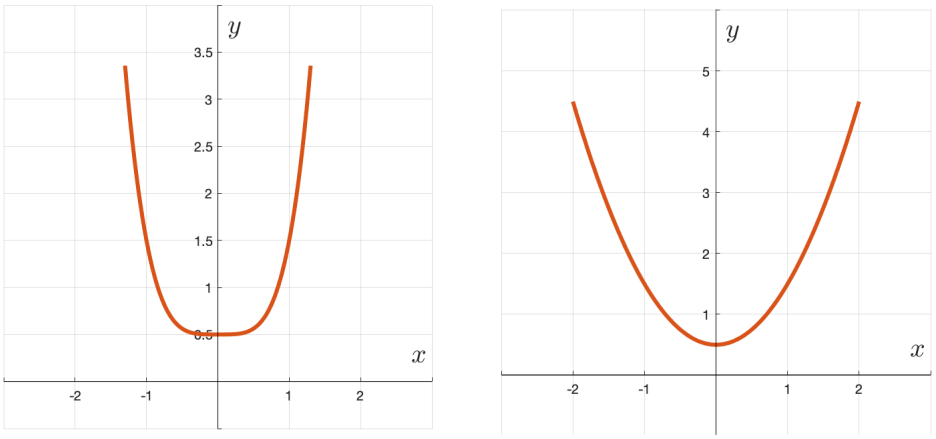
The loss function plays an important role in developing machine learning algorithms for classification. This section discusses the main properties that are important in optimising the loss function in a classification problem. Minimising the loss function requires solving an optimisation problem. The

optimisation algorithm is chosen according to the characteristics of the loss function, which are differentiability and convexity. Therefore, based on these properties, the common loss functions used in the literature have been classified into four groups: convex and smooth losses, convex and non-smooth losses, non-convex and smooth losses and non-convex and non-smooth losses [1, 11, 14, 25, 26]. This can be described as follows.

Convexity is an important requirement for solving an optimisation problem. The matter of convexity derives from the fact that finding a global solution is guaranteed and can be achieved by any local search [26]. The optimal minimum/maximum solution of $f(x)$ must have zero gradient $\nabla f(x) = 0$ in each dimension if the optimisation variable x is a vector [11]. Any point has $\nabla f(x) = 0$ is a critical point, where f can have several critical points. Therefore, the second derivative should be tested. If the result is positive for all critical points, then the function is convex, and if we only have one critical point, the function is strictly convex, see Figure 2.1.1. If the second derivative of the critical points is not all negative or positive, then f is not convex [26], see Figure 2.1.2.

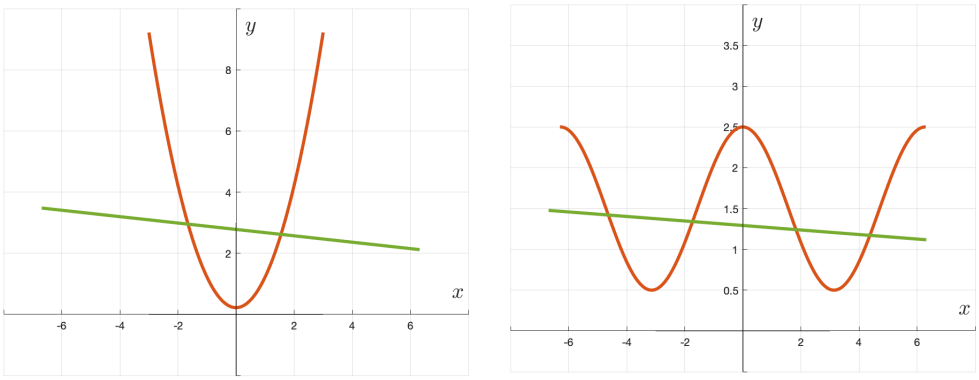
To determine whether the function is convex, a convexity condition should be tested. This condition states that for any pair of points x_i and x_j in the domain of f, and $\alpha \in [0, 1]$, the following inequality condition should be satisfied:

$$f(\alpha x_i + (1 - \alpha) x_j) \leq \alpha f(x_i) + (1 - \alpha) f(x_j) \tag{2.1.1}$$



(a) Convex function (b) Strictly convex function

Figure 2.1.1: Difference between convex and strictly convex functions. Figure 2.1.1a the global optimal solution is a set of local minimum solutions which leads to making a solution not unique. Figure 2.1.1b there is only one local minimum solution; therefore, the global solution is unique.



(a) Convex function (b) Non-convex function

Figure 2.1.2: Difference between convex and non-convex functions. Figure 2.1.2a Convex function with one critical point, and Figure 2.1.2b non-convex function, which has several critical points and the second derivative of these local points is positive in local minimum points and negative in local maximum points.

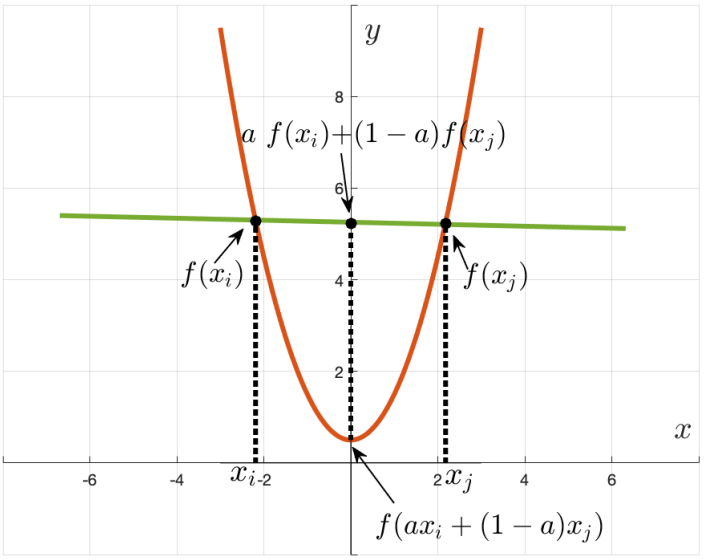
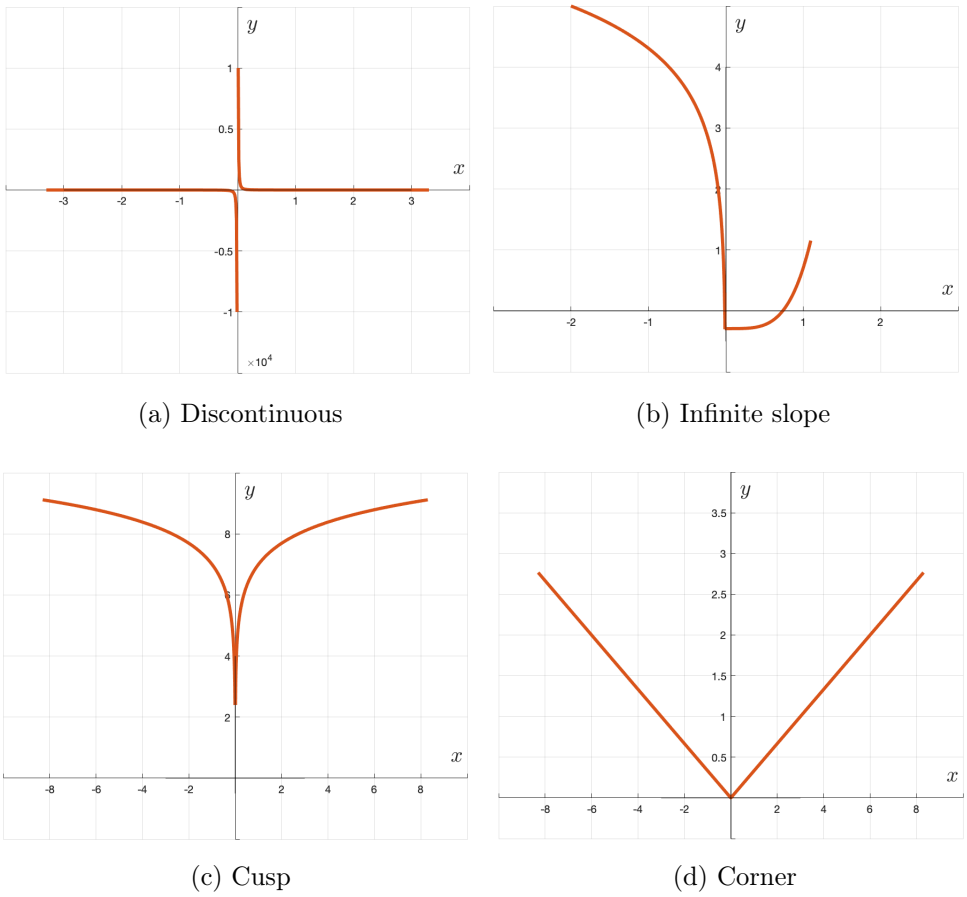


Figure 2.1.3: Convexity condition: the solid green line joining the two points $p_i = (x_i, f(x_i))$ and $p_j = (x_j, f(x_j))$ lies entirely inside the curve, and never comes out of any part of the curve. For any input point x_s between the two selected x_i and x_j. $f(x_s) = f(\alpha x_i + (1-\alpha)x_j)$ should be less than or equal to the sum of the separated outputs of $\alpha f(x_i)$ and $(1-\alpha)f(x_j)$.

in the case of a strictly convex function, the sign $\leq$ is replaced by $<$ when $x_i \neq x_j$ and $0 < \alpha < 1$ [13, 26]. For further explanation, please see Figure 2.1.3.

The second important property is differentiability. The differentiable function is a function whose gradient exists everywhere in the f domain. Each differentiable function is continuous and smooth [11, 12, 27], which means that f does not contain any breaks, corners, vertical lines or cusps, see Figure 2.1.4.

f is differentiable if it has a unique tangent line in the case of a single variable or a tangent hyperplane in the case of multiple variables [26]. If the gradient of any point in the f domain is the same from the left and right



(a) Discontinuous (b) Infinite slope

(c) Cusp (d) Corner

Figure 2.1.4: Different types of functions, where some gradients of f do not exist, such as in 2.1.4a are undefined and have infinite slope like 2.1.4b or contain cusps or corners like 2.1.4c and 2.1.4d; the slope at the corner point (or cusp point) is not unique.

points, then the function is differentiable. Formally, f is differentiable at x_n if the limit of the right and left sides exists and gives the same value [11], such as:

$$\lim_{h \to 0_-} \frac{f(x_n - h) - f(x_n)}{h} = \lim_{h \to 0_+} \frac{f(x_n + h) - f(x_n)}{h} = c \tag{2.1.2}$$

Therefore, the degree of difficulty in optimising any loss function can be identified from convexity and differentiability properties.

2.2 Loss functions, empirical risk and empirical risk minimisation (ERM)

Here, we define the data set space $\mathcal{M} = \mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is the feature space and $\mathcal{Y} = \{-1, 1\}$ is the space of binary class indicators. The goal of classification is to find an optimal classification function $f^\star \in \mathcal{F}$ which maps elements of the feature space to binary class indicators, $f : \mathcal{X} \to \mathcal{Y}$. The optimality of the classification function on a given data set M comprising a set of N points $x_n \in X$ and class indicators $y_n \in Y$ for $n = 1, 2 \ldots N$, is measured using the *risk functional* R_l parameterized by a *loss function* l. A *loss function*, $l : \mathbb{R} \times Y \to \mathbb{R}_{>0}$, quantifies the (non-negative) cost of misclassification for a given predicted class indicator $\hat{y}$ and known class indicator y by evaluating $l(\hat{y}, y)$. Multiple loss functions are in widespread use

in machine learning and will be discussed in this chapter.

The *empirical risk functional* $\hat{R}_l : \mathcal{F} \times M \to \mathbb{R}_{\geq 0}$ parameterized with respect to loss l, non-constant with respect to f, is defined as the following empirical average:

$$\hat{R}_l(f, M) = \frac{1}{N} \sum_{n=1}^{N} l(f(x_n), y_n) \tag{2.2.1}$$

The essential question is *how optimal prediction function $f^\star$ can be found based on the given data.* This question was addressed using a foundational framework presented in [28], named empirical risk minimisation (ERM). Because the underlying distribution is unknown, minimising the true risk over the population can be approximated by minimising the empirical risk over a given finite set of points. The ERM formula can be expressed as follows:

$$f_l^\star = \arg\min_{f \in \mathcal{F}} \hat{R}_l(f, M) \tag{2.2.2}$$

The basic idea of using empirical risk as a measure of the goodness of f originates from the law of large numbers theorem [29, 30]. It is a theorem that simply states that the mean of variables that have been drawn randomly and independently from a probability distribution $\mathcal{P}$ numerous times should be close to the true mean. As long as the number of repetitions increases, the average of the results becomes closer to the true value [29]. By applying the law of large numbers theorem to the empirical risk and true risk, we can see that the average loss of points (empirical risk) converges to that over all

points (the true risk) as the data size reaches infinity:

$$\hat{R}_l(f, M) = \frac{1}{N}\sum_{n=1}^{N} l\left(f\left(x_n\right), y_n\right) \rightarrow R = E\left[l\left(\hat{\mathcal{Y}}, \mathcal{Y}\right)\right] \qquad \text{for } N \rightarrow \infty \tag{2.2.3}$$

From this perspective, the use of empirical risk is justified and conditioned on the size of the training points. In practice, the optimal function is chosen such that it can minimise the empirical risk; therefore, the performance of machine learning algorithms varies depending on how the empirical risk is minimised, where the difference between them depends on the loss functions used in optimising the empirical risk. In fact, this is a very difficult problem and how close we can come to this ideal will depend upon many factors but crucially, it will depend upon how we measure classification risk.

2.3 The canonical 0–1 loss for classification and current optimisation algorithms

The natural approach to evaluate the performance of classifiers is to count the number of misclassified points. In statistics, the 0–1 loss function reflects the real performance of any classification algorithm, which penalises the misclassified points equally, and minimising this function during the learning process is equivalent to minimising the classification error [3, 13, 31]. However, this property makes the 0–1 loss to be NP-hard for global minimisation [6, 7]. Therefore, the 0–1 loss function is replaced by surrogate loss functions.

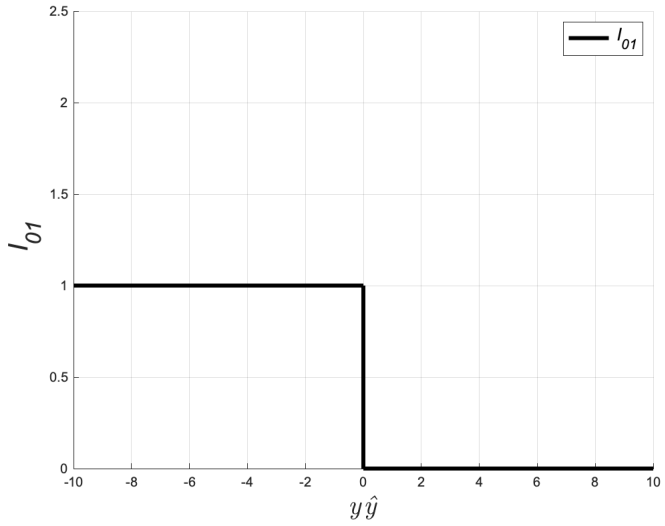


Figure 2.3.1: The 0–1 loss curve. The gradient of 0–1 loss is zero everywhere, and we have an infinite slope from 1 to 0.

Generally, surrogate loss functions produce a real continuous loss rather than discrete loss, which offers convexity and smoothness properties that are desirable in solving optimisation problems [2]. The 0–1 loss function for any given function $f \in \mathcal{F}$ can be expressed as:

$$l_{01}\left(\hat{y}_n, y_n\right) = \begin{cases} 1 & y_n \hat{y}_n \leq 0 \\ 0 & y_n \hat{y}_n > 0 \end{cases} \tag{2.3.1}$$

where $\hat{y}_n = w^t x_n + b$, in which w^t and b are classifier parameters of f. The misclassified point $y\hat{y} \leq 0$ takes the loss value of 1 and 0 when the point is correctly classified, see Figure 2.3.1.

2.3.1 Combinatorial search method for 0–1 loss optimisation

2.3.1.1 Brute-force search for 0–1 loss optimisation

The global optimal solution can be found by evaluating all possible solutions and return the optimal one. This process is known as a brute-force search or exhaustive search [32]. Brute-force search is a simple algorithm to implement; to find a global optimal classifier, we should pass through all possible combinations of labels (all assignments) that have size L^N, where N indicates the number of points and L indicates the number of labels. In a binary classification problem, we have 2^N possible labelling. In fact, brute-force search is an exponential search that is not efficient in solving large problems; it is typically used when the candidate solutions are limited [32, 33]. The complexity of the brute force increases exponentially according to the number of training points and the number of labels. Recently, some studies have proposed the use of heuristic techniques to reduce the search space and obtain the best possible solution in reasonable time [9, 22, 34].

2.3.1.2 Branch and bound for 0–1 loss optimisation

The branch and bound (BnB) technique is a global optimisation technique used for solving combinatory, discrete and general mathematical optimisation problems [35]. In general, for a given problem, BnB aims to explore the search space in intelligent way considering all solutions and return an optimal one. BnB is based on a divide and conquer strategy, which breaks down the problem into subproblems over a tree structure. Each node corresponds

to a subproblem, and the process of dividing a node into subproblems is called branching. The bounding process is used to avoid useless branching; therefore, at each node, the objective value is computed and the current node is bounded according to this value. If further branching does not achieve a better objective value than the previous one, the branch is pruned (or fathomed). In addition, branches can be pruned if we obtain an infeasible solution, which occurs when there is no solution satisfies the constraints of the problem [36, 37].

As mentioned above, BnB can be used to solve discrete problems; therefore, it is formulised in the 0–1 loss optimisation setting [21, 22]. The objective function aims to reduce the sum of individual losses and can be written as:

$$\min \sum_{n=1}^{N} 1\left[y_n \hat{y}_n \leq 0\right] \tag{2.3.2}$$

where $1\left[.\right]$ is an indicator function that gives 1 if the condition is ture and 0 otherwise.

Thus, we need to find the optimal assignments that achieve the minimum sum of losses. f can be obtained by solving linear programming (LP). LP is a mathematical optimisation that requires to find the optimisation variables that achieve the best value for the objective function (which can be maximisation or minimisation) and satisfy all given constraints [38]. For example, if we have three points, p_i, p_j and p_k, whose loss values $l_i, l_j$and l_k are $\{1, 0, 1\}$, then LP should find f that satisfies these constrains $\{y_i\hat{y}_i \leq 0, y_j\hat{y}_j > 0, y_k\hat{y}_k \leq 0\}$.

In fact, BnB requires a suitable heuristic strategy to avoid exponential

worst-case performance. Therefore, a good initial assignments can help to prune a large portion of assignments that have empirical risk larger than initial one, which helps to accelerate the search. For instance, points that are far away from the decision boundary produced by the initial solution should be selected first for branching, and they should be assigned the same initial losses in the tree; only points that are close to the decision boundary are more likely to be changed.

A part of the heuristic strategy proposed in [22], refers to checking the feasibility of solution. A forward checking is used; each point belonging to the convex hull of previously assigned points should have the same label, otherwise, the current node will produce an infeasible assignment.The actual time complexity of BnB depends on the heuristic strategy and the initial solution that is followed.

2.3.1.3 Prioritised combinatorial search

Prioritised combinatorial search (PCS) is another combinatorial algorithm which was designed to find a hyperplane that has a minimum 0–1 loss by checking all hyperplanes that pass through D selected points for a given data. This means that the size of the search space is $\binom{N}{D}$ and the optimal one will be one of the $\binom{N}{D}$ hyperplanes, see Figure 2.3.2.

PCS works as follows: select a subset of points $P_D \in M$ and find a hyperplane that passes through these selected points P_D; each point should satisfy the following equation:

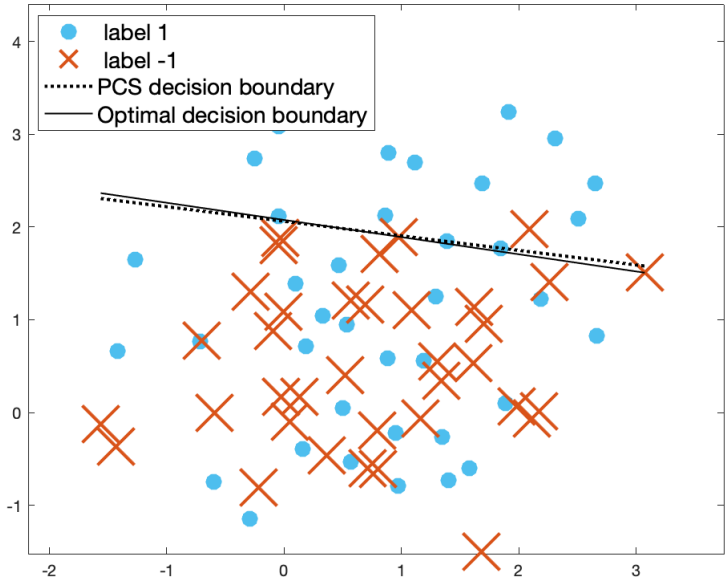


Figure 2.3.2: Plot showing the main concept of PCS. Solid line is the decision boundary that optimally separates two classes, and dashed line is the decision boundary obtained from PCS, which goes through two points and gives the same prediction as the optimal one.

$$f = w^t x_n + 1 = 0 \quad \forall x_n \in P_D \quad n = 1, 2 \ldots D \tag{2.3.3}$$

The above equation can be written as $w^t A = -1$, where $A = \{x_1, \ldots, x_D\}$ is a matrix and -1 is a unit vector. PCS sorts the points based on the initial solution in ascending order, in which points nearest to an initial solution are combined first. For each solution, 0–1 loss should be calculated and returned f that achieves a minimum 0–1 risk. Points lie on the decision boundary are classified into both labels and the label with the lowest 0–1 risk is selected.

2.3.1.4 Combinatorial search approximation

Combinatorial search approximation (CSA) is a prioritised version of PCS, with the main difference being that PCS goes through all combination of D points. CSA starts the search process from the points nearest to an initial solution and terminates when there is no more combination of D points left. The combination of D points depends on their distance to the initial solution; the nearest points will be selected first. In CSA, the search process is split into two stages. First, the search space is reduced to m points nearest to the initial hyperplane, where m is the value specified by the user. Second, the optimal combination obtained from the first stage is swapped between two points x_i and x_j, where x_i is in the current combination and x_j is any point that does not belong to the current combination. Then, the f that corresponds to the new combination is found and evaluated. If the new combination achieves a better 0–1 loss, then the current combination is updated [22]. PCS and CSA can only work when $N \gg D$. In addition, PCS is an intuitive direct 0–1 loss minimisation algorithm that has not been proven theoretically.

2.3.2 Local derivative-free descent techniques

Derivative-free local optimisation is an alternative approach of using local optimisation techniques when the derivatives of a function are unavailable. In fact, there is no systematic way to find the gradient amount and the direction of descent in which a small change in the classifier parameters can lead to a massive jump in the loss value [39]. Derivative-free local techniques, such as gradient descent and coordinate descent, have been used in [24, 40, 41, 42].

Perceptrons by coordinate descent (PCD) [40], is a derivative-free coordinate descent, which uses the perceptron algorithm to update a single coordinate at each step. Perceptron is a binary classification algorithm that uses the perceptron learning rule (PLR) [43], to update w_r when the prediction over some points is wrong. Let w be a classifier parameter, $w^{D+1} = \left(w^0, w^1, \ldots, w^D\right)$, w^0 be a bias term and w^i be a weight associated with each dimension of the features. Only misclassified points are used to adjust a single coordinate w^i which can be achieved as follows:

$$w^i_{1+r} = w^i_r + a\left(yx^i\right) \tag{2.3.4}$$

where α is a learning rate that determines the step size. PCD is applied repeatedly until all training points are correctly classified.

In [41, 42], stochastic gradient descent (SGD) was used instead of coordinate descent. SGD is a local technique that updates all classifier parameters at once, where the update depends on a small portion of randomly selected training points. If the selected points are classified correctly, then no change will be made to the current classifier parameters; otherwise, the classifier parameters will be updated according to only misclassified points. The pocket strategy is combined with the perceptron to solve the stability problem in [41], if the data are not quite linearly separable. The pocket strategy preserves the best-till-now classifier parameters that incur a small training error during iterations. This means that the classifier parameters will be updated as long as we have misclassified points; however, the best classifier parameters among all these updates will be saved and used to predict unseen points.

In [42], the voted-perceptron algorithm (V-PSGD) was proposed, which retains all classifier parameters $w_{r=1,2...R}$ that have been produced during the training process and assigns a weight to each w_r. The weight value v is calculated from the number of iterations over which the current classifier parameters have not changed. This means that a higher weight v is assigned to the classifier parameters that predict most of the training points correctly before change. For prediction, each unseen point should be classified by all weighted classifier parameters, and the final label is assigned based on the sum of votes such as follows:

$$\hat{y} = \text{sign}\left(\sum_{r=1}^{R} v_r \ \text{sign}\big(w_r^t x + w_r^0\big)\right) \tag{2.3.5}$$

Another method that uses perceptron is random coordinate descent (PRCD) [24]. PRCD is slightly different from the previously described algorithms, in which the descent direction and descent step are different. PRCD does not use misclassified points to update the classifier parameters. The descent direction yx in the (2.3.4) is replaced by the randomly generated d vector; thus, PLR can be described as follows:

$$w_{1+r} = w_r + ad \tag{2.3.6}$$

where, variable d is generated randomly by a gaussian distribution. Each component in vector d is obtained randomly in the range of the feature mean

and variance.

PRCD does not use a fixed descent step α, which should be determined at each iteration. For a given d, the optimal α is found by solving the following optimisation problem:

$$\min_{\alpha\in\mathbb{R}} \sum_{d^t x_i \neq 0} 1\left[y \operatorname{sign}\left(d^t x_i \left(\frac{w^t x_i}{d^t x_i} + \alpha\right)\right) \leq 0\right] \tag{2.3.7}$$

The inner product of $\operatorname{sign}(\langle w + ad, x\rangle)$ is equivalent to $\operatorname{sign}\left(d^t x_i \left(\frac{w^t x_i}{d^t x_i} + \alpha\right)\right)$; thus, α is treated as a bias term in (2.3.7), and d and $a \in \mathbb{R}$.

We can conclude that method-based local derivative-free descent techniques can minimise the 0–1 risk for linear separable data; however, for non-separable cases, the global optimal solution is not guaranteed to obtain by these techniques [44], even though the obtained solution in the linear separable data is not unique, as there are an infinite number of hyperplanes that yield the same training error. In addition, derivative-free local techniques are probably stuck in a local solution, and the right direction cannot be known. This makes the behaviour of local techniques wiggly and unstable.

2.4 Surrogate loss functions and associated optimisation algorithms

The current sophisticated machine learning classification algorithms comprise a surrogate loss function and an optimisation algorithm. In this section, the most commonly used loss functions as a relaxation of 0–1 loss, as well as their optimisation techniques, are discussed.

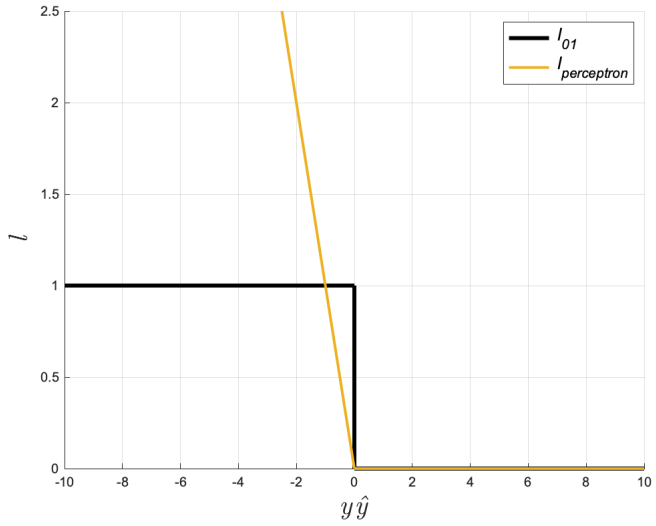


Figure 2.4.1: Loss value that will be allocated to any misclassified point for $l_{\text{perceptron}}$. Compared to l_{01}, $l_{\text{perceptron}}$ assigns a real loss > 0 for the misclassified point.

2.4.1 Perceptron loss function

Perceptron loss, proposed by Frank Rosenblatt [129], is a piecewise function that penalises the misclassified point according to its distance from the decision boundary. The formula of the perceptron loss function can be expressed as follows:

$$l_{\text{perceptron}}\left(\hat{y}_n, y_n\right) = \begin{cases} -y_n\hat{y}_n & y_n\hat{y}_n \leq 0 \\ 0 & y_n\hat{y}_n > 0 \end{cases} \tag{2.4.1}$$

For a given point x_n and any measurable continuous function f, a perceptron loss will be zero if sign $(\hat{y}_n) = y_n$ and real positive if sign $(\hat{y}_n) \neq y_n$. Figure 2.4.1 shows the loss value with respect to the predicted label.

The loss value of the misclassified point is not bounded, and the larger

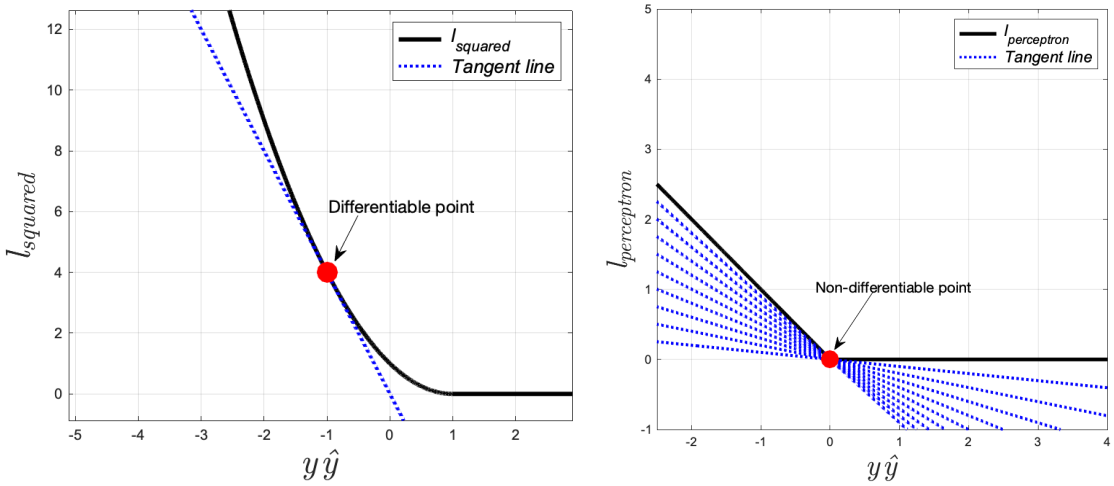
the distance, the greater is the loss value. The principle of perceptron loss is simple and easy to understand; moreover, it is continuous, convex and differentiable everywhere unless at the corner at point zero. These properties make perceptron loss more efficient and preferable in optimisation than the basic classification loss. The gradient of the non-differentiable point is a set of *sub-gradients*, usually referred to as *sub-differential*. Each point should have a single tangent line to be differentiable; if the point has several tangent lines, then this point has infinite sub-gradients, all of which satisfy the following inequality condition [46, 47]:

$$f(x) \geq f(x_n) + q(x - x_n) \tag{2.4.2}$$

The right hand side of (2.4.2) is a tangent line at point x_n; the line that passes x_n should be below the curve of function [26]. q is a gradient (also known as a slope). In the case of perceptron loss, the gradient at $y\hat{y} = 0$ is a set of sub-gradients in the range $[0, -yx]$. Figure 2.4.2 provides a visual explanation of the sub-gradients of a non-differentiable point (i.e. $y\hat{y} = 0$). In practice, it is sufficient to find any one sub-gradient $g \in q$, and $g = -yx$ is usually considered.

2.4.1.1 Perceptron algorithm

The perceptron algorithm is a simple machine learning algorithm that optimises the perceptron loss through the SGD optimisation algorithm. The starting point can be any random initial point. It requires the gradient of loss function to be known; thus, finding a global minimum point is straight-



(a) (b)

Figure 2.4.2: Difference between differentiable point in Figure 2.4.2a and non-differentiable point in Figure 2.4.2b. $y\hat{y} = 0$ has several tangent lines (shown in blue), and each tangent line has a different slope.

forward. The classifier parameter w and bias b are updated, as follows:

$$\nabla_w \left(y\hat{y}\right) = \begin{cases} -yx & y\hat{y} \leq 0 \\ 0 & y\hat{y} > 0 \end{cases} \tag{2.4.3}$$

$$\begin{aligned} \nabla_b \left(y\hat{y}\right) &= -y \\ w_{1+r} &= w_r + a\nabla \left(y\hat{y}\right) \\ b_{1+r} &= b_r \ + a\nabla_b \left(y\hat{y}\right) \end{aligned} \tag{2.4.4}$$

where w_{1+r} indicates the classifier parameters obtained after updating, and w_r, b_r only changes if $y\hat{y} \leq 0$.

The perceptron algorithm finds the decision boundary that can classify

datasets with minimum training error. It does not have any constraint on choosing the decision boundary. The model obtained by the perceptron algorithm sometimes exhibits poor generalisation performance on unseen points. This is due to the lack of a technique for finding the optimal unique solution such as optimising the separation distance of two classes. For more details on the perceptron algorithm, please refer to [44, 48].

2.4.2 Hinge loss function

Hinge loss is a common loss used in support vector machine (SVM) algorithm; it was introduced in [49]. Compared to perceptron loss, hinge loss is more robust and generalises well on unseen points. It helps to find the optimal decision hyperplane from an infinite number of hyperplanes. The optimal decision hyperplane can be defined as a hyperplane that separates the points of different labels and maximises the distance between their two closest points [49]. Not only does this provide a unique solution for the training set, but maximising the margin between two labels over the training set also leads to better classification performance on unseen points [9]. Hinge loss can be described as:

$$l_{\text{hinge}}\left(\hat{y}_n, y_n\right) = \max\left(0, 1 - y_n \hat{y}_n\right) \tag{2.4.5}$$

It clearly shows that hinge loss penalises points that have margin less than one even if correctly classified, whereas in perceptron loss, only misclassified points are penalised. Figures 2.4.1 and 2.4.3 show that perceptron loss and

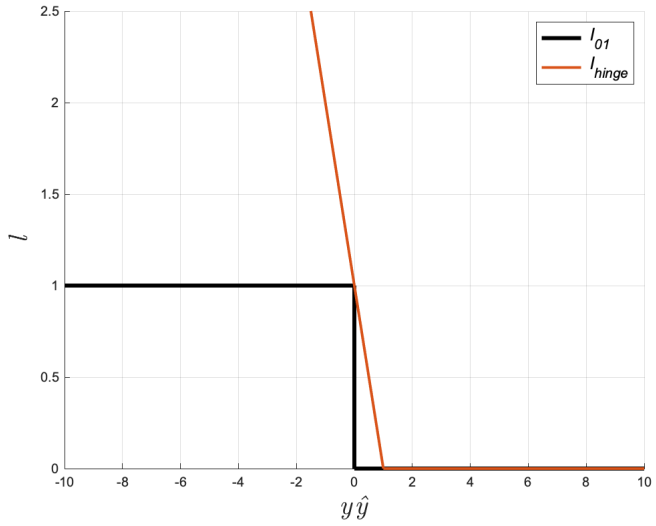


Figure 2.4.3: The loss function curves for l_{hinge} and l_{01}. l_{hinge} is an upper bound for l_{01}.

hinge loss have the same curve, and the only difference lies in the threshold value used for the punishment.

SVM is considered a powerful binary classification algorithm that provides an optimal and unique approximate solution for 0–1 loss if the data are free from noise [18, 19]. The optimisation problem of SVM is to find a linear separable hyperplane that can separate the points of different labels with a minimum sum of hinge loss, where the distance between the closest points from either label to the hyperplane is maximised. The optimisation problem can be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_n\hat{y_n} \geq 1, \quad n = 1, 2 \ldots N \end{aligned} \tag{2.4.6}$$

This formula (2.4.6) is hard-margin SVM, where the points satisfying $y_n\hat{y_n} = 1$ are referred to as support vectors (SVs), which are used to determine the hyperplane position, and deleting any of these SVs will change this position.

In real-world datasets, hard-margin SVM with a non-separable dataset is infeasible. The possible solution is to relax the constraint in (2.4.6) and let some points lie on the margin or even on the wrong side of the decision boundary. This can be achieved with a slight change in the objective function (2.4.6) by introducing the slack variable ζ. Thus, the optimisation problem can be reformulated as follows:

$$\begin{aligned} \text{minimize} \quad & \frac{\lambda}{2}\|w\|^2 + \sum_{n=1}^{N} \zeta_n \\ \text{subject to} \quad & y_n\hat{y_n} > 1 - \zeta_n, \quad n = 1, 2 \dots N \\ & \zeta_n \geq 0 \end{aligned} \tag{2.4.7}$$

The above formula is soft-margin SVM, where λ is the parameter that controls the trade-off between minimising the misclassification error (we should define a small value of λ) and maximising the margin (we should define a large value of λ). ζ_n takes 0 value for a correctly classified point and a positive real value if the point is far away from the right marginal hyperplane. A marginal hyperplane is one that passes through support vector points.

SVM can be formulated as a primal or a dual optimisation problem. The former problem should be solved over the feature space; the number of parameters that need to be optimised should be equal to the feature dimensions D plus the bias term b. In contrast, the latter problem should be solved over the input space; the number of parameters that need to be optimised should be equal to the number of points N, and the b term becomes a constraint that should be satisfied during the optimisation process. In some dual optimisation algorithms, the b term is included implicitly by increasing the feature space by one dimension filled with one value [50]. Here, we will describe only the primal optimisation problem.

2.4.2.1 Primal optimisation algorithm

The primal objective function can be solved as an unconstraint optimisation problem, where (2.4.7) can be simplified as:

$$\text{minimize} \quad \frac{\lambda}{2}\|w\|^2 + \sum_{n=1}^{N} \max\left(0, 1 - y_n \hat{y}_n\right) \tag{2.4.8}$$

The Pegasos algorithm (shorthand of Primal Estimated sub-GrAdient SOlver for SVM) was proposed to solve (2.4.8) [51]. It is very simple and quite similar to the perceptron algorithm. The gradient and updating step

of (2.4.8) can be descried as follows:

$$\begin{aligned} \nabla_{w_r} &= \lambda w - 1\left[y\hat{y} < 1\right] yx \\ \nabla_{b_r} &= -1\left[y\hat{y} < 1\right] y \\ w_{r+1} &= w_r + \alpha\nabla_{w_r} \\ b_{r+1} &= b_r \ + \alpha\nabla_{b_r} \end{aligned} \tag{2.4.9}$$

where α is defined as $\frac{1}{r\lambda}$ in the Pegasos algorithm, where r is the current iteration number.

The primal optimisation problem is preferred to be used when the number of points is very large and the feature dimensions are relatively small.

2.4.3 Logistic loss function

Logistic loss (or log loss function) is a loss function that was introduced with logistic regression (LR) [52], and has been extensively used in deep learning algorithms in several applications, such as image classification, object recognition/detection and multimedia applications [15, 53, 54, 55]. The LR minimises the negative logistic likelihood. The negative logistic likelihood function takes the predicted probability as input, where its value should be between zero and one, and outputs a loss value within $[0, \infty)$. Therefore, to minimise the logistic loss, we must maximise/minimise the predicted probability of a 1/0 label. The logistic loss function formula can be expressed as follows:

$$l_{\log}(\hat{y}_n, y_n) = -[y_n \log(\hat{p}_n) + (1 - y_n) \log(1 - \hat{p}_n)] \tag{2.4.10}$$

where $y_n \in \{0, 1\}$ is the true label (instead of $\{-1, 1\}$), $\hat{p}_n$ is obtained from a transformation function such as the sigmoid function $\hat{p}_n = \frac{1}{1+\exp(-\hat{y}_n)}$.

LR is a desirable algorithm for creating a conditional probability model, in which some applications, such biostatistical applications, need to quantify the probability of a particular disease to happen or the possibility of its occurrence in the future [56]. To find the optimal LR model, we must update the classifier parameters based on the predicted probability of points for the current classifier. The misclassified point is determined differently from SVM or the perceptron algorithm, in which if $\hat{p}_n < t$ and $y_n = 1$ or $\hat{p}_n \geq t$ and $y_n = 0$, where $t = \frac{1}{5}$, is a prediction threshold value, then the point is misclassified. In addition, correctly classified points contribute to the total loss as long as the loss value > 0 [54].

Logistic loss has suitable properties that facilitate its easy optimisation using any unconstrained optimisation technique. The gradient descent (or SGD) is used to identify the optimal classifier parameters, where the classifier parameters are iteratively updated based on the gradient of the logistic loss function, which can be computed as follows:

$$\begin{aligned} \nabla_{w_r} &= [\hat{p}_n - y_n]\, x_n \\ \nabla_{b_r} &= \hat{p}_n - y_n \end{aligned} \tag{2.4.11}$$

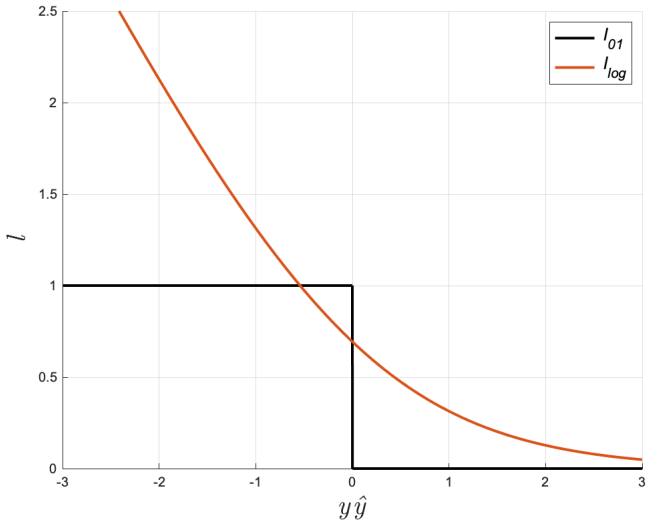


Figure 2.4.4: Logistic loss and 0–1 loss curves. The distant correctly classified points contribute to the total loss.

Although the logistic loss function is easy to optimise and maximise the probability of correct classification, the loss value is calculated more intensively than hinge loss, and it is very affected by the outliers, see Figure 2.4.4.

2.4.4 Smooth 0–1 loss

The 0–1 loss function can be smoothed using a modification of the sigmoid function, where the degree of smoothness is controlled by a smooth parameter k. Smooth 0–1 loss (SL) can be expressed as:

$$l_{\text{sl}}\left(\hat{y}_n, y_n\right) = \frac{1}{1 + \exp^{ky_n\hat{y}_n}} \tag{2.4.12}$$

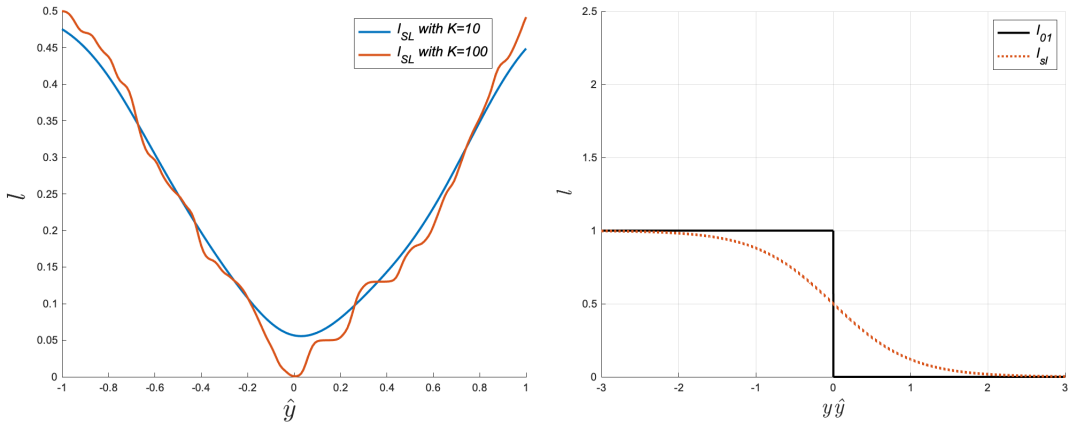
A small value of k yields a very smooth approximation of 0–1 loss with one local minima, whereas a large k produces a sharp approximation of 0–1

loss with several local minima (Figure 2.4.5). k is a hyperparameter that should be chosen properly for a better approximation. SL was proposed to limit the loss value that is assigned to the outliers. Smooth 0–1 loss approximation (SLA) is a classification algorithm that optimises SL by a combination of optimisation techniques: coordinate decent and hill climbing [22]. Both techniques belong to the family of local search, and due to the non-convexity of SL, an exhaustive search can be made to avoid getting stuck in the local solution. Coordinate decent was described in the previous section, and hill climbing will be described in the following subsection. SL can be derived as follows:

$$\nabla_w = \frac{-ky_n x_n \exp^{ky_n \hat{y}_n}}{(1+\exp^{ky_n \hat{y}_n})^2} \tag{2.4.13}$$
$$w_{r+1} = w_r - \alpha \nabla_w$$

2.4.4.1 Hill climbing optimisation technique

Hill climbing is an iterative algorithm that starts with an initial arbitrary solution and then uses a heuristic search to find the best solution by making an incremental change to the current solution. If the change achieves a better solution, then the current solution is replaced with the best one, and so on, until no further improvement can be identified [57]. Hill climbing can get stuck in the local minimum solution. The local minimum solution is the best among the nearest solutions but is not a global optimal solution. This means that a small change in the local minimum solution cannot produce



(a) (b)

Figure 2.4.5: 2.4.5a: degree of smoothness with respect to the k parameter, where a small value yields a smooth curve and a large k creates a sharp curve. 2.4.5b: Loss value that will be assigned to the point based on the prediction margin $y\hat{y}$.

any improvement. Therefore, if the current solution cannot improve after r specified increment changes, then another initial solution is created to find another best solution. Finally, all local minimum solutions are compared, and the best solution is selected. Hill climbing can perform parallel local search by generating several initial solutions simultaneously, and then performing local search with respect to each initial solution. Finally, the best local minimum solution among them is considered the best solution [58].

2.5 Competing views of ERM

In fact, ERM depends on two main factors: the loss function used and the capacity of the hypothesis space [59]. The capacity of the hypothesis space is managed by a trade-off between bias and variance, which has recently

become a controversial topic based on two main views: the conventional wisdom view and the modern interpolation view. These are described in the following subsection.

2.5.1 ERM-based conventional wisdom view

The conventional view is widely accepted in the machine learning community and is based on the bias–variance trade-off. The conventional classifiers rely on managing the model complexity to obtain a low generalization error. The Model complexity refers to how the training model captures the relation between the features set and the label [60]. It has been observed empirically, a training model that learns everything in the training set has a bad performance on the unseen points, this is known as overfitting. Overfitting occurs when the learning model is very complex and captures everything in the training data [60, 61]. Classifiers based on conventional view prevent overfitting by avoiding fully fitting the training data.

In fact, there are several factors cause overfitting, such as the presence of noise, the amount of training data is insufficient and the complexity of classifiers [60]. To mitigate the effects of overfitting, various strategies are suggested, such as early stopping, pruning, regularisation and ensemble technique. Early stopping is used to monitor the performance of the algorithm during training. This technique is widely used in neural networks, as by training the neural network and updating the classifier parameters, the validation error is calculated on a separate set called validation set. We keep tracking the validation error during the training process and terminate the

training process when the validation error stops decreasing or starts increasing. The classifier parameters with respect to the minimum validation error are used for prediction. This technique shows a higher generalisation level and guarantees that the fitted model will not overfit or underfit [62].

Pruning has the same objective and is generally used in the decision tree. Rather than creating a model that fits all training points, where each leaf node contains a single point or points with a single label, we limit the number of leaf nodes or prune the nodes according to the specified threshold parameter [63, 64]. Another way to avoid overfitting is regularisation, which is commonly used to reduce the model's complexity. It helps to shrink the classifier weights of unwanted features; this helps to have a model capture the essential features for classification and simplify the model [65, 66]. To achieve this, a regularisation parameter should be set properly to find a better balance between overfitting and underfitting. The ensemble technique is another good strategy to prevent overfitting through combining the predictions of multiple models trained on a subset of training set [67, 68]. Combining several classifiers does somewhat smoothen these decision boundaries and produce a new well-generalised decision boundary [68].

In contrast, underfitting occurs when the model is very simple and cannot capture the underlying relationship between the features and label, resulting in poor prediction on test set [69]. Sometimes, this can result in avoiding overfitting; to reduce the model complexity, we may produce a weak model [70]. In addition, underfitting can also be attributed to poor hyperparameter selection [71]. To avoid underfitting, we might need to increase the model complexity, which can be achieved by avoiding early stopping or increasing

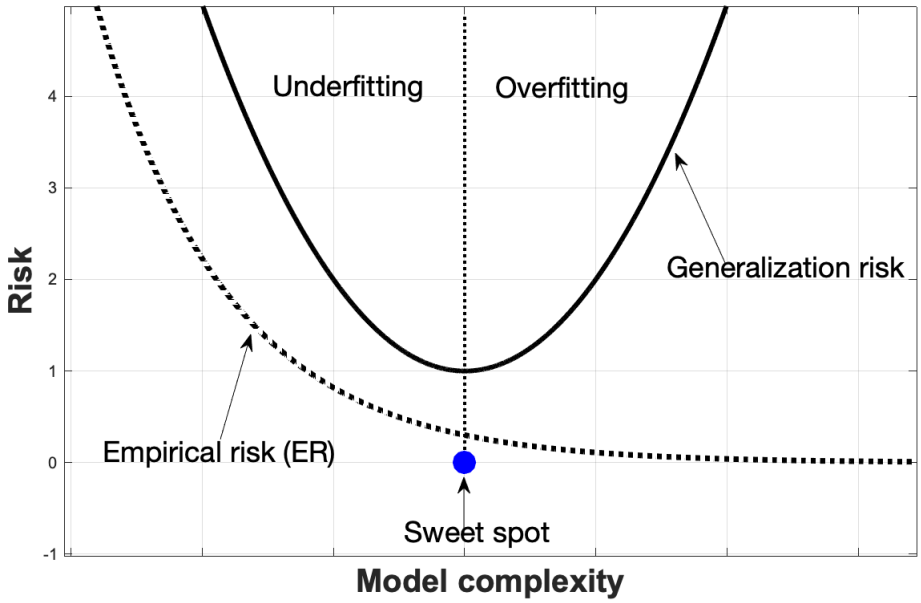


Figure 2.5.1: Training and test errors with respect to the conventional view regime. The test error usually has a U-shaped curve and training error always has a decreasing curve, where the highest risk lies with underfitting and lowest risk lies with overfitting. Excessively increasing the model complexity leads to overfitting, whereas excessively decreasing the model complexity causes underfitting. Managing the model complexity is a very difficult task. The sweet spot shows the optimal balance between underfitting and overfitting.

the number of predefined hyperparameters of the model [69, 70, 72]. Underfitting or equivalently high bias can be easily recognised when empirical risk is very large. The conventional view relies on finding a better balance between variance and bias; see Figure 2.5.1 for further explanation.

2.5.2 ERM-based modern interpolation view

The modern interpolation classifiers have shown that the training model can achieve zero empirical risk on the train set and high accuracy on the test set [73, 74, 75]. The interpolation view can be described as a regime in which

the train set is classified correctly, and when the parameter of the model complexity increases, the generalization risk decreases. The state-of-the-art machine learning algorithms based on modern interpolation view include a deep-structured neural network, random forest (RF) and adaptive boosting (AdaBoost). Minimising the empirical risk to have exactly zero has been avoided in the conventional view because of the training model practically having poor generalisation on the test set. The justification of this is the model with zero empirical risk is highly complex, capture the noise, memorise the training points and cannot estimate the true function that links the points to their true label [60, 61, 69]. However, classifiers based on modern interpolation view show different generalization performance that conflicts what has been observed with conventional classifiers. Despite the conventional classifiers showed that the training model that fits the noise points has a bad performance on the unseen points, the interpolation classifiers showed that the fit of the noise points does not degrade the ability of the classifier to predict the test points [75, 89]. In fact, increasing the model complexity after reaching overfitting peak and having zero training risk help to decrease the generalisation risk, this has been observed in the modern interpolating regime, see Figure 2.5.2.

Recently, several contributions have attempted to understand the success of interpolation algorithms. One of these attempts has been to understand the role of margin in the interpolation model [76, 77, 78, 79]. AdaBoost is a common algorithm whose success is likely attributed to its ability to increase the margin of points. This was observed in practice when AdaBoost was run for several iterations, and the generalisation error decreased even

when the empirical risk was zero [80]. This is justified as decreasing the generalisation error is attributed to increasing the prediction margin of the points. However, it is not clear how AdaBoost maximises the margin. In the optimisation theory, there is no explanation regarding why the training error is zero, and still the test error continues to decrease [81]. Based on this finding, several investigations were conducted. Breiman [76] took a step to investigate whether there is a relationship between the generalisation risk and margin. They designed an algorithm called Arc-gv, which explicitly maximises only the smallest margin of the training points. Arc-gv is different from AdaBoost, where the prediction coefficient changes according to the minimum margin of the training points [76].

They concluded that there is no relationship between increasing the margin and generalisation risk reduction. In fact, the generalisation risk of Arc-gv is worse than that of AdaBoost [76]. Since then, several contributions have been made to increase the margin of the points [77, 79, 82, 83]. All these contributions follow the same way, which updates the prediction coefficient with respect to the margin of the selected points; however, the margin was calculated differently in their work. Increasing the margin of the selected points results in a worse performance than AdaBoost. This finding was interpreted in [77] as the main reason why AdaBoost's success more likely relies on the margin of entire data is maximised, not just focusing on maximising the smallest margin of the training points (i.e. Arc-gv) [76], or a subset of the training points [77, 79, 82, 83]. In fact, the success of AdaBoost's performance is linked to the margin intuitively, and the role of the margin in reducing the generalisation risk remains an active area of research.

In addition, the success of deep-structured neural networks is attributed to over-parameterisation [73, 84, 85, 86]. As long as the structure of the neural network increases, the number of parameters increases, thereby increasing the model complexity. In [87] showed that increasing the structure of neural networks is related to the complexity of data, more layers are needed if the data is more complex. However, in [88] pointed out that over-parameterisation is necessary to have a smooth boundary that is robust to the slight noise added to the original features. Recently, it has been shown that deep neural networks are able to fit the training set exactly even if the data is corrupted by a high noise level [75, 89]. There is no clear understanding regarding why deep learning is successful; the complex structure of deep neural networks makes the predictive model less understandable. Even though several empirical studies have shown the role of over-parameterisation in reducing the generalisation risk and empirical risk, there are significant gaps in our understanding, and there is no rigorous explanation for the generalisation performance in the interpolation regime [73, 74, 75].

RF is an interpolation classifier that combines several decision trees, where each tree is trained on a subset of points with random feature selection. RF uses a decision tree as a base classifier, it fits the training data exactly by constructing deep decision tree without using pruning strategy; however, compared to a single decision tree built by using all features set (which is based on a conventional view), RF achieves superior performance and low generalisation risk, which makes it an off-the-shelf classification algorithm for many applications [90, 91]. The success of RF as a good interpolation classifier is attributed to the self-averaging mechanism, which helps

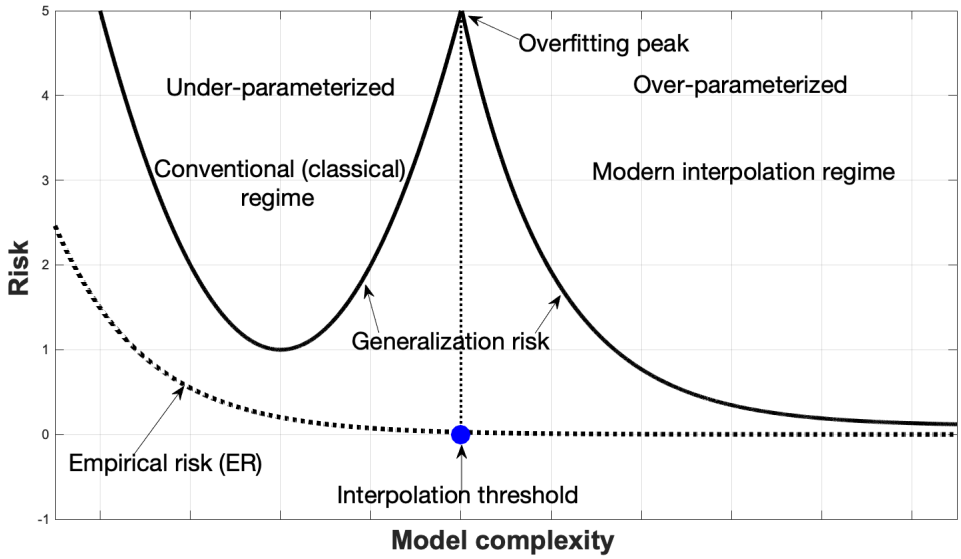


Figure 2.5.2: Training and test errors with respect to the modern interpolation regime. The test error decreases in the modern regime after reaching an overfitting peak (interpolation threshold) and continues to decrease until the test error in the modern regime is less than that in the classical (conventional) regime.

to smoothen the prediction results [90]. Adaboost, deep learning, and RF have been shown that they have a double descent curve. Any classifier has a double descent curve is considered as a modern interpolation classifier.

Despite the superiority of these interpolation classifiers, it is possible that the reason behind the success of these classifiers does not depend on a specific factor and may have several factors that can be combined to produce a powerful interpolation classifier.

2.6 Inadequacies of previous work

Two different ways to optimise the 0–1 risk were described. In addition, we discussed the ERM views that have conflicting opinions. In this section, we

will highlight the limitations of the existing works.

Minimising the convex surrogate loss functions is attractive due to the efficient method of finding the global optimal solution using numerous existing optimisation methods. However, the convexity property makes the loss value unbounded, thereby, increasing sensitivity to outliers and the amount of overlap between labels, as well as further issue produced for the imbalance label problem [15]. In real-world data, these issues are often present. In this context, the attention is focused on solving these issues by mitigating the outlier effects through the application of detection and removal strategies [18, 19] or limiting the loss value assigned to the unwanted point by introducing a robust non-convex surrogate loss function [3, 22]. To address the imbalance label problem, resampling and sampling weights are the common strategies [20, 92, 93]. In these cases, when the surrogate loss function is not a good approximation of the 0–1 risk function, directly optimising the 0–1 risk is required [22]. This is because the 0–1 loss provides a constant loss value (not continuous real value like surrogate loss functions) for any distant misclassified point, therefore it is robust in the presence of noise. In addition, correctly classified points do not contribute to the total sum of the loss values, 0–1 loss only penalizes the misclassified points, therefore 0–1 loss does not affect by the imbalance label problem.

The difficulty of directly solving the problem led to the limit of 0–1 risk minimisation algorithms. BnB solves the combinatorial optimisation problem relatively faster than the Brute-force algorithm [94]. However, its actual time complexity, in terms of the best and worst cases, varies from one dataset to another, with the reasons behind this variance remaining unknown. More-

over, the PCS algorithm is considered a direct 0–1 risk minimisation algorithm, while its complexity class is unknown [22]. Methods based on the local derivative-free descent techniques do not have limited solution space (such as BnB or PCS), therefore, the optimal global solution is not guaranteed as these methods do not enumerate all solutions and select the optimal one. There is no provably exact algorithm for 0-1 classification loss minimisation in polynomial time.

The second part of this thesis discusses the ERM views. The modern interpolation ERM view has recently appeared [73]. However, the main factors that participate in creating an interpolation classifier without degrading the generalisation performance remain to be elucidated [75, 77, 95]. Empirical studies have failed to unravel the reasons for AdaBoost's success [77, 79, 82, 83]. While the success of RF has only been practically demonstrated [90], the success of deep neural networks is tied to several explanations [73, 75, 87, 88]. Notwithstanding, the complex structure and the huge number of parameters of deep neural networks make it difficult to find a rigorous explanation [96].

Some algorithms still function according to the ERM-based conventional wisdom view and have not been examined under the ERM-based modern interpolation view, such as the KNN algorithm for $K > 1$ [74, 97, 98]. Understanding the success of the interpolation classifier depends on the complexity of the algorithm's structure. KNN is easy to implement and simple. Therefore, developing a KNN algorithm and studying its performance under the modern interpolation view is definitely a field of opportunity to cover this knowledge gap and contributes to understanding and analyzing the success

of interpolation classifiers.

2.7 Summary

In this chapter, the general characteristics of loss functions in terms of convexity and differentiability are presented. These properties are important in determining the optimisation algorithm that should be used, where the degree of optimisation difficulty can be determined from these properties. The loss function can be classified into convex and smooth, non-convex and smooth, convex and non-smooth and non-convex and non-smooth functions. All these functions have been discussed in this chapter, along with their possible optimisation algorithms. The common machine learning algorithms are based on ERM, where the loss function used in ERM is different. Recently, a new approach of optimising ERM has appeared, and fitting the data perfectly no longer has a negative effect on the generalisation risk. It can achieve zero emoirical risk as well as low generalisation risk. The reasons behind this are still under investigation.

Chapter 3

Exact canonical 0–1 loss on linearly separable data

Surrogate loss functions are used as a proxy for optimising the canonical 0–1 loss indirectly; however, the consequences of this replacement have been studied in the literature by introducing classification-calibrated loss functions. The guarantee of exact minimisation of the canonical 0–1 loss was proved under the conditional surrogate loss function. However, the true conditional probability is not given, which verifies that the classification-calibration theorem is not practically applicable. Therefore, we studied the canonical 0–1 loss and surrogate loss function based on ERM; if both losses have the same 0–1 empirical risk, then surrogate loss is the exact minimiser for 0–1 loss – otherwise, it is not. In addition, we proved that optimising hinge risk is the same as optimising 0–1 risk in the linearly separable case; therefore, it can be used as the exact minimiser for the 0–1 loss in linearly separable data.

3.1 Introduction

Although, the capacity of combinatorial search is exponential – the number of assignments for N points in the binary classification problem is 2^N– not all these assignments can be separated by linear function. Cover [99] provides an interesting measure that counts the number of learnable functions that can separate the data linearly into two classes. The number of assignments that can be partitioned by a linear function for N points in D dimensional space is given by a quantitative measure: $2\sum_{i=0}^{D}\binom{N}{i}$ for $N \gg D$. This measure helps to reduce the capacity of combinatorial search space to much less than 2^N. However, Cover did not provide an algorithm that could find all these linearly separable assignments or define the expected complexity of such an algorithm. So far, there is no provably exact, polynomial time algorithm for 0-1 classification loss minimization.

Alternatively, the convex loss function l_ϕ is commonly used. Due to this replacement, a fundamental question was raised in the machine learning community: *Does optimising the surrogate loss function l_ϕ on behalf of the canonical 0–1 loss l_{01} have statistical consequences for choosing an optimal function with respect to the original problem ?* [100, 101, 102, 103]. The difficulty lies in finding how the two different measures are related to each other in a way that ensures that the optimising l_ϕ leads to choosing the optimal function with respect to l_{01}. This has been addressed by introducing classification-calibration [100]. l_ϕ is said to be classification-calibrated if the optimal function $f_\phi^\star$ results of optimising the conditional surrogate risk provides the same sign as the Bayes function $f_{01}^\star = \text{sign}(\eta(X) - \frac{1}{2})$, where $\eta(x_n)$

is the conditional probability of a single point x_n, and $\eta(x_n) \in [0, 1]$. In other words, the condition $\text{sign}(f_\phi^\star(X)) = \text{sign}(\eta(X) - \frac{1}{2})$ should be fulfilled in order to say that l_ϕ is classification-calibrated. The conditional surrogate loss is defined as follows:

$$C(l_\phi, \eta) = \eta(x) l_\phi(f(x)) + (1 - \eta(x)) l_\phi(-f(x)) \tag{3.1.1}$$

The above equation simply states that $C(l_\phi, \eta)$ has a large penalty when the sign of $f(x)$ differs from the sign of Bayes classifier; therefore, minimising the surrogate loss function according to the conditional probability always leads to $f_\phi^\star(x)(\eta(x) - \frac{1}{2}) > 0$ for all $x \in X$, $\eta(x) \neq \frac{1}{2}$ and $f \in \mathcal{F}$, which is referred to as 'Fisher consistent' [9].

If l_ϕ is Fisher consistent and convex, the sequence of measurable functions $f_i : X \to \mathbb{R}$ implies the following:

$$R_\phi(f_i(X)) \to R_\phi^\star \Longrightarrow R_{01}(f_i(X)) \to R_{01}^\star. \tag{3.1.2}$$

Therefore, l_ϕ can be the exact minimiser for the 0–1 loss on linearly separable data if it has the following properties: Fisher consistency, differentiability and convexity. Pointwise Fisher consistency states a sufficient condition to have a Bayes consistency. However, the classification-calibration theorem cannot be verified in practice because the conditional probability is unknown. Moreover, the optimality of l_ϕ is linked to the Bayes rule, which is never at-

tained in practice due to the limited data size. In fact, we are interested in the $f_\phi^\star$ obtained from ERM. Therefore, the main contributions in this chapter are as follows:

- Simplifying the optimality of l_ϕ based on ERM (section 3.2).
- Proving that hinge risk provides the same 0–1 empirical risk (section 3.3).

3.2 The optimality of the canonical 0–1 loss in the surrogate losses

There are multiple alternative functions ϕ which are typically used as surrogates for the 0–1 loss. In particular, the hinge loss, a defining component of the SVM is a convex, piecewise linear, upper bound on the 0–1 loss. Similarly, the convex logistic loss, which is differentiable everywhere and also a upper bound on the canonical 0–1 loss. The optimal function $f_l^\star$ with respect to the choice l needs to be found by solving the optimisation problem. When the minimum is not unique and leads to a set of functions, we break ties by choosing one of them arbitrarily. Optimal classification functions are unavoidably specialized to that risk/data combination, as captured in the following theorem.

Theorem 1. *Let $f_{01}^\star, f_\phi^\star \in \mathcal{F}$ be the optimal classification functions under the empirical risk functionals $\hat{R}_{01}, \hat{R}_\phi$ respectively where ϕ is not the canonical*

0–1 loss. Then the following inequality holds:

$$\hat{R}_{01}\left(f_{01}^{\star}, M\right) \leq \hat{R}_{01}\left(f_{\phi}^{\star}, M\right). \tag{3.2.1}$$

Equality is only obtained when $f_{01}^{\star}(x) = f_{\phi}^{\star}(x)$ *for all* $x \in X$.

Proof. By the definition of $f_{01}^{\star}$, we have $\hat{R}_{01}\left(f_{01}^{\star}, M\right) \leq \hat{R}_{01}\left(f', M\right)$ for all $f' \in \mathcal{F}$, and since $f_{\phi}^{\star}$ is one of those functions f', then the inequality part of (3.2.1) holds. If the risk functional $\hat{R}_{\phi}$ has the same minima with respect to f as $\hat{R}_{01}$, then $f_{01}^{\star}(x) = f_{\phi}^{\star}(x)$ for all $x \in X$ so that the equality part of (3.2.1) holds. □

To paraphase, this simple theorem says that, for a given fixed data set M, the only prediction functions which are guaranteed to minimise the canonical 0–1 loss, are those which are optimal under that 0–1 loss. No other surrogate loss ϕ, however mathematically convenient, can achieve a better empirical risk. The inequality is not strict however, so there are indeed special cases where a function $f_{\phi}^{\star}$ optimal for a surrogate loss ϕ can achieve the same risk as the 0–1 loss; but this is not the case in general and is not something which is easy to arrange as it depends upon properties of the data set which may be unknown. We will discuss the important special case of this where the hinge loss and linear classification functions for linearly separable data later, for example.

What these results mean is that the common practice in machine learning of changing the risk function during evaluation of classification performance, or substituting a surrogate loss function for the 0–1 loss for mathematical convenience, loses the optimality guarantee provided by(2.2.2). One part of

this thesis is about trying to retain this guarantee to improve the quality of classifiers.

3.3 The hinge loss selects the 0–1 loss optimal linear model for linearly separable data

Hinge loss is an upper bound loss for 0–1 loss that can minimise $\hat{R}_{01}$ exactly under a certain circumstance. Optimisation algorithms that are used to reduce it are effectively able to minimise the 0–1 loss exactly and find the unique optimal $f^{\star}_{\text{hinge}}$, where $f^{\star}_{\text{hinge}} = f^{\star}_{01}$, only if $f^{\star}_{01}$ is unique, otherwise $f^{\star}_{\text{hinge}} \in F^{\star}_{01}$, where $F^{\star}_{01}$ is a set of functions that provide the same optimal 0–1 risk. Here, we will prove that optimising the hinge loss selects the optimal linear model for the 0-1 loss.

Theorem 2. *For linearly separable data and linear classification models:*

$$\hat{R}_{01}\left(f^{\star}_{01}, M\right) = \hat{R}_{01}\left(f^{\star}_{hinge}, M\right) = 0 \tag{3.3.1}$$

Proof. To begin, note that, for given, linearly separable data M, f can be obtained by solving a constraint satisfaction problem, formulated as:

$$\begin{aligned} \text{Find} \quad & f \\ \text{subject to} \quad & y_n \hat{y}_n \geq 1, \quad n = 1, 2 \ldots N \end{aligned} \tag{3.3.2}$$

Similarly, for given data M which is not linearly separable, f can be obtained

by solving a modification of the above, the following constrained optimisation problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{n=1}^{N} \zeta_n \\ \text{subject to} \quad & y_n \hat{y}_n \geq 1\text{-}\zeta_n, \quad n = 1, 2 \dots N \\ & \zeta_n \geq 0 \end{aligned} \tag{3.3.3}$$

Because the classification model is linear and the data is linearly separable, all N constraints of the constraint satisfaction problem (3.3.2) are satisfied. This implies that we must have $y_n \hat{y}_n \geq 1$ for all $n = 1, 2 \dots N$. Next, we note that the variable ζ can be expressed equivalently using the hinge loss, $\zeta = \max(0, 1 - y\hat{y})$. By substituting this into (3.3.3), we convert this into an unconstrained optimisation problem, then the minimization problem can be written as:

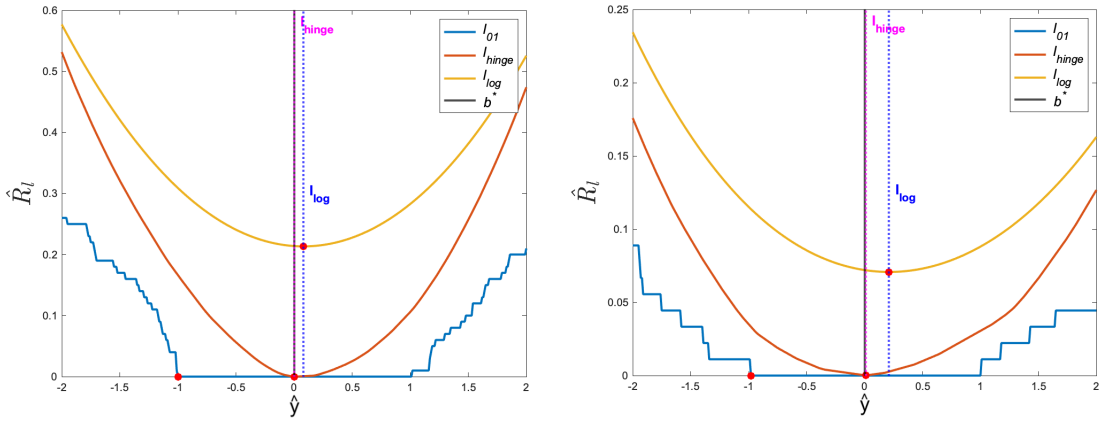
$$\text{minimize} \quad \sum_{n=1}^{N} \max(0, 1 - y_n \hat{y}_n) \tag{3.3.4}$$

Now, (i) since all the constraints in the satisfaction classification problem are satisfied, then $\hat{R}_{01}\left(f_{01}^{\star}, M\right) = 0$, therefore the optimal value for ζ in the constrained optimisation problem will be $\zeta_n = 0$ for all $n = 1, 2 \dots N$. Furthermore, (ii) as $\zeta = \max(0, 1 - y\hat{y})$, then we must have $\sum_{n=1}^{N} \max(0, 1 - y_n \hat{y}_n) = 0$. Finally, combining (i) and (ii), we have $\hat{R}_{01}\left(f_{01}^{\star}, M\right) = \hat{R}_{01}\left(f_{\text{hinge}}^{\star}, M\right) = 0$, thus proving theorem 2. □

To summarise, the hinge risk can be optimised to have exactly zero value for linearly separable data. This also means that the 0–1 loss can be replaced by the hinge loss in linearly separable case. It is possible to show that other surrogate loss functions (i.e log loss) have zero empirical risk under their measure on certain conditions such as the minimum prediction margin $\geq$ a specified threshold value, where the specified threshold guarantees to have zero surrogate empirical risk.

3.4 The empirical risk $\hat{R}$ in practice

Proving theorems 1 and 2 practically is just straightforward. A simple experiment can be done on a single dimensional data, X is generated randomly, where $X \in \mathbb{R}^1$ and $Y=\{-1, 1\}$, $y_i=\{1 : \forall x_i > 0 \text{ and } -1 : \forall x_i \leq 0\}$. We use a simple classification rule: $f(x; w; b) = w^t x + b = \hat{y}$, where w and $b \in \mathbb{R}$ are the parameters to be optimized for determining $f^\star$ over a given M and l. The optimal $w_l^\star$ and $b_l^\star$ depend crucially on a chosen loss function l. To figure out whether the optimal $f_\phi^\star$ produced by minimising $\hat{R}_\phi$ is equivalent to the optimal $f_{01}^\star$ produced by minimising $\hat{R}_{01}$ in linearly separable data and $\hat{R}_{01}\left(f_{\text{hinge}}^\star\right) = \hat{R}_{01}\left(f_{01}^\star\right) = 0$. Figure 3.4.1 shows the result of minimising $\hat{R}$ based on l_{01},l_{hinge} and $l_{\log}$.



(a) (b)

Figure 3.4.1: shows the curves of surrogate loss ϕ and 0–1 loss. The horizontal dotted line shows the optimal $f^\star$ that minimise the $\hat{R}_l$ with respect to l_ϕ. The optimal $f^\star_{01}$ is not unique based on l_{01}, and l_{hinge} finds $f^\star_{hinge}$that exactly achieves the same the optimal solution. Optimal solution is the solid black line.

3.5 Summary

In this chapter, we discussed the optimality of the canonical 0–1 loss function in the linearly separable case. In practice, we only have access to the training set. The optimal function with respect to the surrogate loss functions can be assessed in terms of the original empirical 0–1 risk. If the same empirical risk is achieved, then surrogate loss is the exact minimiser for the 0–1 loss. In addition, optimising the empirical hinge risk always leads to the same empirical risk for 0–1 loss in a linearly separable case.

Chapter 4

Direct 0–1 risk minimisation on overlapping data

Abstract

Algorithms based on optimising 0–1 loss directly are compared in terms of theoretical and actual time complexity in this chapter. T-PP and PS-SVs are the abbreviations for our new direct 0–1 risk minimisation algorithms, which aim to solve the combinatorial problem in polynomial time. There are second-order factors, which are dimensionality of data and overlap width that have an influence on increasing the search space, thereby increasing the execution time. According to our study, the theoretical and actual time complexity levels of PCS and CSA increase as the dimensionality of data increases, whereas in BnB, the time taken to minimise the 0–1 risk depends on the degree of overlap between classes. The best case is when there is a small overlap between classes, and the worst result is with a large overlap.

The actual time of T-PP is constant and is not affected by the dimensionality of data or the overlap width. PS-SVs reduces the search space of the T-PP algorithm, making its actual time complexity much better than that of T-PP. These findings are obtained from extensive experiments conducted on eight small synthetic and five real datasets.

4.1 Introduction

There are combinatorial problems that are thought to be much more difficult than polynomial complexity, under the hypothesis that $P \neq NP$ [104, 105]. For those problems that are shown or suspected to be NP-hard, prioritising the search process may be useful in practice. Prioritising the search for a non-convex problem highly depends on the initial solution and the heuristic strategy [4]. For instance, different starting points may result in different final solutions, and the gap between the initial solution and the global optimal solution can increase the solution time [4, 106]. In real-world data, it is difficult to compute the gap unless a global solution is given. This usually happens on artificial data, where the optimal solution is known, and evaluation will be carried out on a set of different initial solutions to find the best one with the minimum gap. For this kind of problem, a good initialisation strategy can help start the search in a more promising area [107].

The second key factor is the heuristic strategy that will be followed after the initial solution. The heuristic search refers to intelligent search methodologies that are designed to accelerate the search process for solving computationally hard combinatorial problems when the capacity of search space

is huge and the exact algorithm cannot find an optimum solution within a reasonable amount of time [108]. The heuristic strategy that obtains the best local minima among other heuristic search strategies can be translated into the best heuristic strategy to obtain the global minima solution [108]. Thus, finding a good initial solution and a good heuristic strategy contribute to speeding up the search process for combinatorial optimisation problems. BnB is a prioritised version of the Brute-force algorithm, and CSA is designed to reduce the search space of the PCS algorithm [22]. The main contributions in this chapter can be summarised as follows:

- Defining the complexity class of PCS (subsection 4.3.3).
- Proving that the hinge loss is an upper bound of 0–1 loss (subsection 4.3.1).
- Studying the actual time complexity based on second-order factors on small synthetic and real datasets (section 4.4).
- Introducing a new direct 0–1 loss minimisation algorithm, T-PP, and its prioritised version, PS-SV (section 4.2 and subsection 4.3.2.1).

4.2 Testing all pairs of points (T-PP) algorithm

T-PP is our new heuristic algorithm designed to minimise the 0-1 loss exactly in practical way in polynomial time complexity. However, there is no rigorous theoretical proof for its exactness in minimising the 0-1 loss at this moment, instead, we provide a heuristic strategy to find the optimal solution in a practical way. The basic idea behind T-PP is that a global optimum function

$f_{01}^{\star}$ can be identified by the closest pair of points, each pertaining to a different label. The pair of points closest to the decision boundary of $f_{01}^{\star}$ can be defined as the optimal pair $p^{\star}$. The T-PP algorithm simply functions by first searching for $p^{\star}$ and then $f_{01}^{\star}$.

To find $p^{\star}$, going through all the pairs of points is required. In the classification problem, each point is associated with a label. Therefore, it is sufficient to select only one pair of distinct points (one point from each label). This reduces the number of combinations of points needed to test to $(\frac{N}{2})^2$ pairs. In the worst case scenario, half of N points in each class, then, the total number of pairs equals half of N.

The main challenge in T-PP lies in finding $f_{01}^{\star}$, because there is an infinite number of decision boundaries that fit between a pair of points, where not all of them necessarily represent the same 0–1 risk. However, we are only interested in a set of functions that make a pair of points closest to the decision boundary, and have the same 0–1 risk (see Figure 4.2.1).

To find $f_{01}^{\star}$ for a given $p^{\star}$ or the best possible function for other not optimal pairs, other points should be included when fitting the decision boundary between a pair of points. This is achieved by performing two separately incremental procedures.

First incremental procedure

A single point per time is added. The points are in descending order depending on their absolute margin from the initial decision boundary, which is fitted between only one pair of points. After adding a single point, the current decision boundary is adjusted according to the newly added point.

If the updated decision boundary makes the 0–1 risk lower than the previous one, it is replaced by the new one. To ensure that the selected pair of points is still the closest to each adjusted decision boundary, constraints that prevent other points from being closer than the given pair must be set. The linear function f_{01} can be found by solving LP under the following constraints:

$$\begin{aligned} \text{Find} \quad & f \\ \text{subject to} \quad & y_{i,j}\hat{y}_{i,j} = 1, \quad i,j \in p \\ & y_n\hat{y}_n \geq 1, \quad n = 1, 2 \ldots J \end{aligned} \tag{4.2.1}$$

where the equality constraint is set for a pair of points to have a restricted margin equals to 1, whereas non-strict inequality constraints are used for the rest of the points to have a margin ≥ 1; J is the number of the previously added points that satisfy the constraint and the newly added point, and $J \leq N$. It is possible to have more than two points closer to the decision boundary. Therefore, non-strict inequality constraints are set for the other points.

Additionally, points between the pair of points and outliers that do not satisfy the constraints in (4.2.1) will not be included in the fitting of the decision boundary. LP should be solved after adding each point. This incremental procedure is repeated until no more points are left. Figure 4.2.2 presents the T-PP procedure. This procedure is described in Algorithm 4.1 from line 15 to 28.

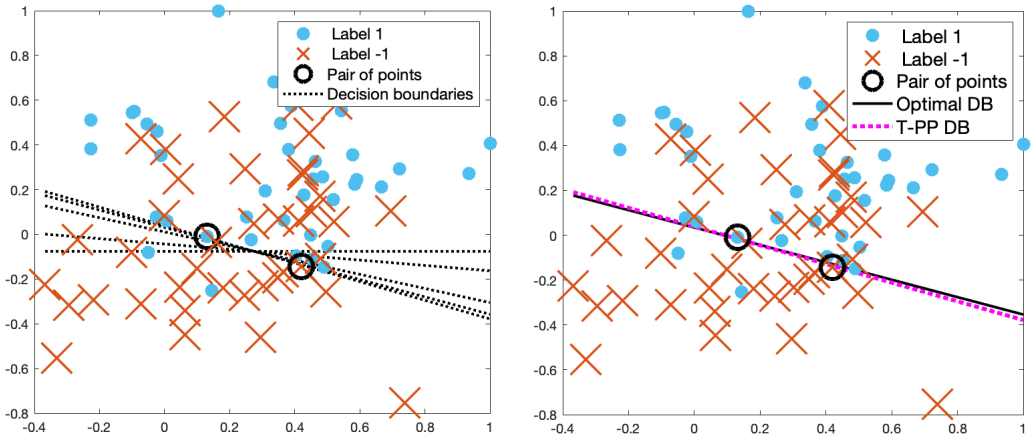
After concluding the first incremental procedure, we need to confirm whether all points belong to a particular label or not. Two cases should

be considered, one of which must be met. First, if there are no points with an absolute margin smaller than the absolute margin of the pair, then all points belong to a particular label, where the pair of points is the closest to the decision boundary of f_{01}. Second, if there are points located between a pair and have an absolute margin smaller than the absolute margin of the pair, then these points are not assigned to any label because they are not included in the fitting of f_{01}. Then, the decision boundary obtained from the first incremental procedure is adjusted using the second incremental procedure to label the points in the margin (see Algorithm 4.1 from line 29 to 34).

The second incremental procedure

In this step, the constraints are relaxed to assign the points within the margin. In this case, it is not necessary for a pair of points to have a margin $y\hat{y} = 1$ as stated in 4.2.1. The purpose of the second incremental procedure is to optimise the heuristic search to find the best possible function or the optimal one even when the selected pair of points is not the optimal. The decision boundary must be adjusted to correct the prediction of misclassified points within the margin by adding a single point per time, starting from the misclassified point closest to the current decision boundary. A new f_{01} can be obtained by solving LP under the following constraints:

$$\begin{aligned} \text{Find} \quad & f \\ \text{subject to} \quad & y_n \hat{y}_n > 0, \quad n = 1, 2 \ldots J \end{aligned} \tag{4.2.2}$$



(a) (b)

Figure 4.2.1: Main concept of the T-PP algorithm. 4.2.1a All these decision boundaries pass between the optimal pair of points $p^\star$ (coloured points with black circles), which not all have the same 0–1 risk. 4.2.1b The optimal function $f_{01}^\star$ can be found under the constraint that no points closer more than the $p^\star$ to the decision boundary of $f_{01}^\star$. The optimal and T-PP decision boundaries have the same 0–1 risk.

where, J is the number of points that are correctly predicted by the function obtained from the first incremental procedure including the newly added point.

If the new f_{01} reduces the 0–1 risk, the current function is replaced by the new f_{01}. Figure 4.2.3 shows the second incremental procedure. This procedure is described in Algorithm 4.2 from line 2 to 13.

The two incremental procedures take $\mathcal{O}(2N)$ in the worst case, and the complexity of solving LP is $\mathcal{O}(\sqrt{D}\log(\frac{1}{\epsilon}))$, where ϵ represents the maximum tolerable duality gap in an interior-point LP solver [26]. Then, the total complexity of T-PP is $\mathcal{O}(\sqrt{D}\log(\frac{1}{\epsilon})N^3)$, which belongs to the polynomial complexity class of order 3. T-PP is described in Algorithm 4.1.

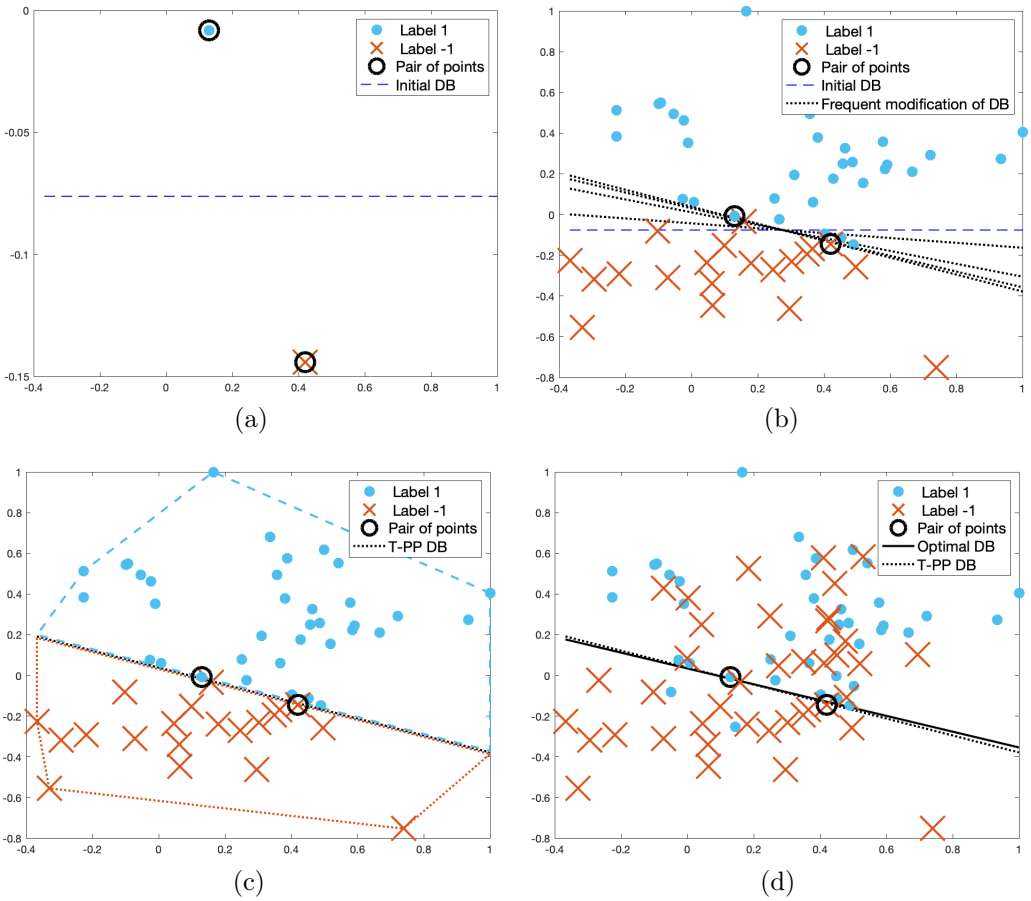


Figure 4.2.2: The steps of T-PP algorithm for finding the optimal function $f^\star$ between the optimal pair of points $p^\star$. 4.2.2a The initial decision boundary is fitted between only two points. 4.2.2b The initial decision boundary is modified frequently through an incremental procedure; a single point is added per time, and then the decision boundary is modified accordingly. 4.2.2c $f^\star$ is obtained by fitting the decision boundary between $p^\star$ and a subset of points. 4.2.2d The T-PP decision boundary and optimal decision boundary belong to the same equivalence class.

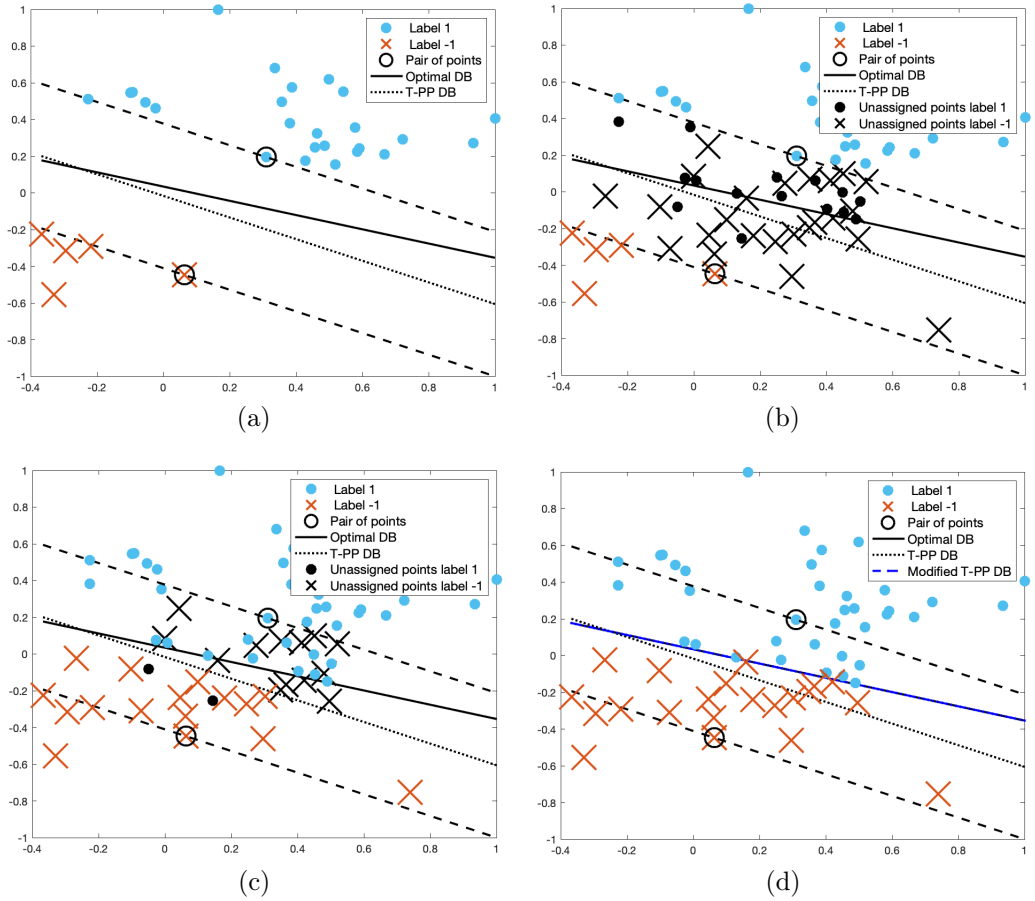


Figure 4.2.3: The second incremental procedure of T-PP algorithm aims to prioritise the search for finding the best possible function or the optimal one even when the pair of points are not close and have a large margin, as shown in 4.2.3a. 4.2.3b All points in the margin cannot satisfy the constraints of the first incremental process. Thus, we need to adjust the T-PP decision boundary to find the best possible decision boundary, not necessarily the optimal one. To do this, we keep the assignments of correctly classified points as in 4.2.3c, and only need to adjust the T-PP decision boundary incrementally for misclassified points starting from the nearest point to the T-PP decision boundary. We may obtain an optimal solution, as in 4.2.3d or a new decision boundary that is better than the previous T-PP decision boundary.

Algorithm 4.1 T-PP algorithm

Input: Training data(X,Y)
Output: $f_{01}^{\star}$ and $l_{0-1}^{\star}$

function T-PP(X,Y)
  $l_{0-1}^{\star} \leftarrow \infty$
  $P^{+} \leftarrow$ find points with positive label
  $P^{-} \leftarrow$ find points with negative label
  for i in P^{+} **do**
    $x_i \leftarrow X_{p_i^{+}}$
    for j in P^{-} **do**
      $x_j \leftarrow X_{p_j^{-}}$
      $S \leftarrow x_i, x_j$
      $C \leftarrow$ add equality constraint
      $\tilde{f_{01}} \leftarrow$ fit a decision boundary between x_j and x_i
      $\hat{Y} \leftarrow \tilde{f_{01}}(X; w; b)$
      $L \leftarrow$ sort N points in descending order $|\hat{Y}|$
      $\min_{0-1} \leftarrow \hat{Y} \neq Y$
      while L is not empty **do**
        $S \leftarrow x_{L_m}$ add new point
        $C \leftarrow$ add constraint for x_{L_m}
        $f \leftarrow$ solve (S,C) LP problem (4.2.1)
        $loss \leftarrow \hat{Y} \neq Y$
        if $loss \leq \min_{0-1}$ **then**
          $f_{01} \leftarrow f$
          $\min_{0-1} \leftarrow$ loss
        else
          Delete x_{L_m} from S
          Delete the constraint of x_{L_m} from C
        end if
        Delete x_{L_m} from L
      end while
      $\hat{Y} \leftarrow f_{01}(X; w; b)$
      if $\min(\hat{Y}) < 1$ **then**
        Q $\leftarrow$ Find point $Y\hat{Y} < 0$
        S $\leftarrow$ Find point $Y\hat{Y} > 0$
        f, $\min_{0-1} \leftarrow$ Incremental-assignment (S,Q, $\min_{0-1}$)
      end if
      if $\min_{0-1} < l_{0-1}^{\star}$ **then**
        $f_{01}^{\star} \leftarrow f_{01}$
        $l_{0-1}^{\star} \leftarrow \min_{0-1}$
      end if
    end for
  end for
end function

Algorithm 4.2 Incremental-assignment

Input: Training data(X,Y)

function INCREMENTAL-ASSIGNMENT(Assigned$_S$,Unassigned$_Q$,min$_{0-1}$)
  for i in Unassigned$_Q$ **do**
    $P \leftarrow X_{Assigned_S}$
    $x_i \leftarrow X_{Unassigned_{q_i}}$
    $f \leftarrow LP(P, x_i)$ find f by solving LP(4.2.2)
    $\hat{Y} \leftarrow f(X; w; b)$
    $loss \leftarrow \hat{Y} \neq Y$
    if loss $<$ min$_{0-1}$ **then**
      $f_{01} \leftarrow f$
      min$_{0-1} \leftarrow$ loss
      Add Unassigned$_{q_i}$ into Assigned$_S$ vector
    end if
  end for
  return f_{01}, min$_{l_{01}}$
end function

4.3 Prioritising the search for the optimal canonical 0–1 loss

Minimising the empirical 0–1 risk is discussed in the subsections that follow from the perspectives of the initial solution, heuristic search and theoretical time complexity.

4.3.1 Initial solution based on hinge risk

It is possible to limit the search space and eliminate a large portion of worst-case solutions. This can be done if the initial solution is obtained from minimising the surrogate upper bound risk. Hinge risk is an upper bound on the 0–1 risk; therefore, starting the combinatorial search from the initial solution obtained by minimising the hinge risk helps to narrow down the search and finds solutions with 0–1 risk values that are less than the initial one. Here, we show that hinge risk is an upper bound on the 0–1 risk.

To compare the 0–1 loss $l_{01}(\hat{y}, y) = 1\,[y\hat{y} < 0]$ and the hinge loss $l_{\text{hinge}}(\hat{y}, y) = \max[0, 1 - y\hat{y}]$, without loss of generality, we can restrict attention to the quantity $y\hat{y} \in \mathbb{R}$ for $y = 1$. There are three critical regions on the real line where the hinge loss intersects with the 0–1 loss—namely, $A_1 = (-\infty, 0]$, $A_2 = (0, 1)$ and $A_3 = [1, \infty)$. For each of these regions, based on A_i we have:

$$\begin{aligned} A_1 &: l_{01}(\hat{y}, 1) = 1 \leq 1 - \hat{y} = l_{\text{hinge}}(\hat{y}, 1) \\ A_2 &: l_{01}(\hat{y}, 1) = 0 \leq 1 - \hat{y} = l_{\text{hinge}}(\hat{y}, 1) \\ A_3 &: l_{01}(\hat{y}, 1) = 0 \leq 0 \qquad = l_{\text{hinge}}(\hat{y}, 1) \end{aligned} \tag{4.3.1}$$

therefore, we have $l_{01}(\hat{y}, 1) \leq l_{\text{hinge}}(\hat{y}, 1)$ for all $\hat{y} \in A_1 \cup A_2 \cup A_3 = \mathbb{R}$. The case $y = -1$ is the same but with the sign reversed. It follows that $l_{01}(\hat{y}, 1) \leq l_{\text{hinge}}(\hat{y}, 1)$ for all $\hat{y} \in \mathbb{R}$ and $y \in \{-1, 1\}$.

Now, inserting these loss functions into the empirical risk, we have:

$$\begin{aligned} \hat{R}_{01}(f, M) &= \frac{1}{N} \sum_{n=1}^{N} l_{01}(f(x_n), y_n) \\ &\leq \frac{1}{N} \sum_{n=1}^{N} l_{\text{hing}}(f(x_n), y_n) \\ &= \hat{R}_{\text{hinge}}(f, M). \end{aligned} \tag{4.3.2}$$

this incorporates the point that for each term in these summations, $l_{01}(f(x_n), y_n) \leq l_{\text{hinge}}(f(x_n), y_n)$, as well as the commutativity of addition.

4.3.2 The heuristic strategy

The heuristic strategies of PCS, CSA, and BnB are explained in more detail in 2.3.1.3, 2.3.1.4 and 2.3.1.2. However, the pseudo-codes for these algorithms are described in Algorithms 4.3, 4.4, and 4.5. A prioritised version of the T-PP algorithm is given below.

4.3.2.1 Prioritising the search by support vectors (PS-SVs)

PS-SVs is a fast version of T-PP that limits the number of pairs that need to be tested. To reduce the actual time complexity of T-PP and speed up the search process, pairs of points must be chosen wisely. For instance, points far away from the overlap area can be discarded because they will be far from the optimal decision boundary. In fact, the optimal solution is somewhere in the overlap area. Thus, we can confine the region in which the optimal pair of points exists. This region can be defined by minimising the hinge risk. Optimising hinge risk requires minimising the sum of the slack variables, which penalises the points in the margin much less than those out of the margin. In the case of small overlap, the SVM margin is definitely not large because increasing the SVM margin will increase the hinge risk. Thus, the selection of the pairs will be restricted to the support vector points obtained from minimising the hinge risk. PS-SVs have two main stages—namely, post-assignment and incremental assignment. Post-assignment can be achieved by testing all pairs of SVs, whereas incremental assignment depends on the post-assignment. Both are described in more detail as follows:

- **Post-assignment strategy**

  This strategy helps to reduce the number of steps needed to perform the first incremental procedure as described in 4.2. The pair of SVs and non-SVs points (correctly classified points based on the initial solution and have margins larger than the pair) are combined to find f_{01} by solving LP as in (4.2.1), where J is the number of non-SVs points. All points between the pair of SVs and non-SVs are assigned simultaneously

without the need to adjust the decision boundary incrementally as in the T-PP algorithm; thus, the search for the optimal function can be accelerated. The pair of SVs points is chosen in ascending order based on their absolute prediction margins resulting from the initial solution, in which each SVs point that has a positive label will be trained with all SVs points that have a negative label. Algorithm 4.6 (from line 2 to 5) shows how the SVs points are defined. Figure 4.3.1 demonstrates the post-assignment strategy, and Algorithm 4.7 describes how it works.

It is necessary to check whether the resulting f_{01} makes the pair of points closer to the decision boundary of f_{01} more than other points or not. This is performed by calculating the absolute prediction margins for all points; if they are larger than the margin of the pair, then there is no need to further update the function. Nevertheless, if some points' absolute prediction margins are smaller than the pair, then, f_{01} should be incrementally updated, see Algorithm 4.7 from line 32 to 37.

For the pair of points that are not optimal and not close to each other, the best possible function can be found through an incremental-assignment strategy.

- **Incremental-assignment strategy**

  This is exactly the same as the second incremental strategy used in the T-PP algorithm (Algorithm 4.2) and described in Figure 4.2.3.

PS-SVs is described in Algorithm 4.6.

Algorithm 4.3 PCS

Input: Training data(X,Y)
Output: $f_{01}^{\star}$ and $\min_{l_{01}}$

function OPTIMAL 0-1 LOSS PCS(X,Y)
  $\tilde{f} \leftarrow$ Get initial solution f for (X,Y)
  $\tilde{l} \leftarrow$ 0-1 loss vector produced by $\tilde{f}$
  $\min_{l_{01}} \leftarrow \sum_{n=1}^{N} \tilde{l}_n$
  $K \leftarrow$ get the points in ascending order based $\left|\tilde{f}(x_n)\right|$
  $p \leftarrow$ the initial combination of D points from K
  while $p \neq \phi$ **do**
    $(f, l_{01}) \leftarrow$ Get Solution(p)
    if $l_{01} < \min_{l_{01}}$ **then**
      ($f^{\star}$,$\min_{l_{01}}$) $\leftarrow$ (f,l_{01})
    end if
    $p \leftarrow$ next combination of points
  end while
  return $f^{\star}$ AND $\min_{l_{01}}$
end function
function GET SOLUTION(p)
  $A \leftarrow p^T$
  $w \leftarrow$ find w by solving $Aw = -1$
  if $l_{01}(w, 1) < l_{01}(-w, -1)$ **then**
    $f \leftarrow (w, 1)$
    $l_{01} \leftarrow l_{01}(w, 1)$
  else
    $f \leftarrow (-w, -1)$
    $l_{01} \leftarrow l_{01}(-w, -1)$
  end if
end function

Algorithm 4.4 CSA

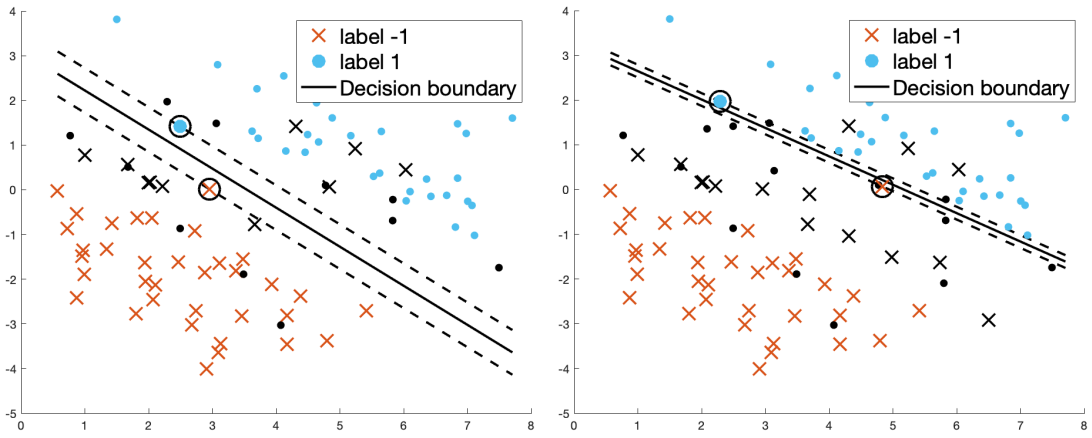
Input: Training data(X,Y), M
Output: $f_{01}^{\star}$ and $\min_{l_{01}}$

function OPTIMAL 0-1 LOSS CSA(X,Y)
  $\tilde{f} \leftarrow$ Get initial solution f for (X,Y)
  $\tilde{l} \leftarrow$ 0-1 loss vector produced by $\tilde{f}$
  $\min_{l_{01}} \leftarrow \sum_{n=1}^{N} \tilde{l}_n$
  $K \leftarrow$ get the points in ascending order based $\left|\tilde{f}(x_n)\right|$
  $V \leftarrow$ get first M points from K
  (f, p, $\min_{l_{01}}$) $\leftarrow$ Optimal combination for 0-1 Loss(V, $\min_{l_{01}}$)
  ($f^{\star}$, $p^{\star}$, $\min_{l_{01}}$) $\leftarrow$ Swapping process (p,N, $\min_{l_{01}}$)
  return $f^{\star}$ AND $\min_{l_{01}}$
end function
function OPTIMAL COMBINATION FOR 0-1 LOSS(V, $\min_{l_{01}}$)
  $p \leftarrow$ the initial combination of D points from V
  while $p \neq \phi$ **do**
    $(f, l_{01}) \leftarrow$ Get Solution(p)
    if $l_{01} < \min_{l_{01}}$ **then**
      ($f^{\star}$,$\min_{l_{01}}$) $\leftarrow$ (f,l_{01})
      $p^{\star} \leftarrow p$
    end if
    $p \leftarrow$ next combination of points
  end while
end function
function SWAPPING PROCESS($\tilde{p}$,N, $\min_{l_{01}}$)
  for i in N **do**
    for j in D **do**
      $p \leftarrow$ remove x_j from current combination $\tilde{p}$ and replace by x_i
      $(f, l_{01}) \leftarrow$ Get Solution(p)
      if $l_{01} < \min_{l_{01}}$ **then**
        ($f^{\star}$,$\min_{l_{01}}$) $\leftarrow$ (f,l_{01})
        $p^{\star} \leftarrow p$
      end if
    end for
  end for
end function
function GET SOLUTION(p)
  $A \leftarrow p^T$
  $w \leftarrow$ find w by solving $Aw = -1$
  if $l_{01}(w, 1) < l_{01}(-w, -1)$ **then**
    $f \leftarrow (w, 1)$
    $l_{01} \leftarrow l_{01}(w, 1)$
  else
    $f \leftarrow (-w, -1)$
    $l_{01} \leftarrow l_{01}(-w, -1)$
  end if
end function

Algorithm 4.5 BnB

Input: Training data(X,Y)
Output: $f_{01}^{\star}$ and $\min_{l_{01}}$

function OPTIMAL 0-1 LOSS MODEL(X,Y)
  $\tilde{f} \leftarrow$ Get initial classifier f for (X,Y)
  $\tilde{l} \leftarrow$ 0-1 loss vector produced by $\tilde{f}$
  $\min_{l_{01}} \leftarrow \sum_{n=1}^{N} \tilde{l}_n$
  $l_{\varnothing} \leftarrow$ initial unassigned $0-1$ loss vector
  BnB($l_{\varnothing}$, $\tilde{l}$, 1)
end function
function BNB($l_{\varnothing}$, $\tilde{l}$, M)
  if all components of $l_{\varnothing}$ are assigned **then**
    $f^{\star} \leftarrow$ get LP solution for $l_{\varnothing}$
    $\min_{l_{01}} \leftarrow \sum_{n=1}^{N}$ sign $(f^{\star}(X_n)) \neq Y_n$
    $\tilde{l} \leftarrow$ 0-1 loss vector produced by $f^{\star}$
    $l_{\varnothing} \leftarrow$ initial unassigned $0-1$ loss vector
    BnB($l_{\varnothing}$, $\tilde{l}$,1)
  else
    $n \leftarrow \arg\max \left| \tilde{f}(x_m) \right|$
    $l_{\varnothing_n} \leftarrow \tilde{l_n}$
    if $\sum_{n=1}^{N} l_{\varnothing_n} < \min_{l_{01}}$ **then**
      BnB($l_{\varnothing}$, $\tilde{l}$, m+1)
    else
      $l_{\varnothing} \leftarrow$ Feasibility($l_{\varnothing}$, n, $1 - \tilde{l}_n$)
      if $\sum_{n=1}^{N} l_{\varnothing_n} < \min_{l_{01}}$ **then**
        BnB($l_{\varnothing}$, $\tilde{l}$,m+1)
      else
        Infeasible solution, start a new branch BnB($l_{\varnothing}$, $\tilde{l}$,m-i)
      end if
    end if
  end if
end function
function FEASIBILITY($l_{\varnothing}$, n, $\tilde{l}_n$)
  $l_{\varnothing_n} \leftarrow \tilde{l_n}$
  $\hat{y} \leftarrow$ targets prediction vector implied by $l_{\varnothing}$
  Let Φ convex hull of $\{x_k \mid \hat{y}_k \neq \hat{y}_n\}$
  if $x_n \in \Phi$ AND $\hat{y}_k \neq \hat{y}_n$ **then**
    $l_{\varnothing_n} \leftarrow \infty$
  end if
end function



(a) (b)

Figure 4.3.1: The pair of SVs (coloured points with black circles) and non-SVs (coloured points) are used to fit the decision boundary between the pair. Therefore, all points inside the label region (black coloured points) are assigned simultaneously without using incremental procedure, this helps to speed up the assignment process. Panels (a) and (b) show different selections of pairs of SVs points.

Algorithm 4.6 PS-SVs

Input: Training data(X,Y)

Output: $f_{01}^{\star}$ and $\min_{l_{01}}$

1: **function** FIND SV POINTS(X,Y)
2: $\tilde{f} \leftarrow$ Get initial solution f for (X,Y)
3: Find $SVs_{+} \leftarrow y_i\hat{y}_i \leq 1$ and $y_i = 1$ sorted in ascending order
4: Find $SVs_{-} \leftarrow y_i\hat{y}_i \leq 1$ and $y_i = -1$ sorted in ascending order
5: **return** SVs_{+}, SVs_{-}
6: **end function**
7: $\min_{l_{01}}$, $f_{01}^{\star} \leftarrow$ Post-assignment function(SVs_{+},SVs_{-})

Algorithm 4.7 Post-assignment

Input: Training data(X,Y)

function POST-ASSIGNMENT(SVs_+,SVs_-)
  $l^\star_{0-1} \leftarrow \infty$
  for i in SVs_+ **do**
    $x_i \leftarrow X_{SVs_{+i}}$
    for j in SVs_- **do**
      $x_j \leftarrow X_{SVs_{-j}}$
      $S \leftarrow X_{non-SVs}$ find non-SV points for the pair of points
      $f \leftarrow LP(S, x_i, x_j)$ find f by solving LP(4.2.1)
      $\min_{0-1} \leftarrow \infty$
      $\hat{Y} \leftarrow f(X; w; b)$
      $loss \leftarrow \hat{Y} \neq Y$
      if loss $< \min_{0-1}$ **then**
        $f_{01} \leftarrow f$
        $\min_{0-1} \leftarrow$ loss
      end if
      $L \leftarrow |\hat{Y}| < 1$ sort the point in descending order
      $S \leftarrow |\hat{Y}| > 1$
      while L is not empty **do**
        $S \leftarrow x_{L_m}$ add new point
        $C \leftarrow$ add constraint for x_{L_m}
        $f \leftarrow$ solve (S,C) LP problem (4.2.1)
        $loss \leftarrow \hat{Y} \neq Y$
        if $loss \leq \min_{0-1}$ **then**
          $f_{01} \leftarrow f$
          $\min_{0-1} \leftarrow$ loss
        else
          Delete x_{L_m} from S
          Delete the constraint of x_{L_m} from C
        end if
        Delete x_{L_m} from L
      end while
      $\hat{Y} \leftarrow f_{01}(X; w; b)$
      if $\min(\hat{Y}) < 1$ **then**
        Q $\leftarrow$ Find point $Y\hat{Y} < 0$
        S $\leftarrow$ Find point $Y\hat{Y} > 0$
        f, $\min_{0-1} \leftarrow$ Incremental-assignment (S,Q, $\min_{0-1}$)
      end if
      if $\min_{0-1} < l^\star_{0-1}$ **then**
        $f^\star_{01} \leftarrow f_{01}$
        $l^\star_{0-1} \leftarrow \min_{0-1}$
      end if
    end for
  end for
  return $l^\star_{0-1}$, f_{01}
end function

4.3.3 Theoretical time complexity

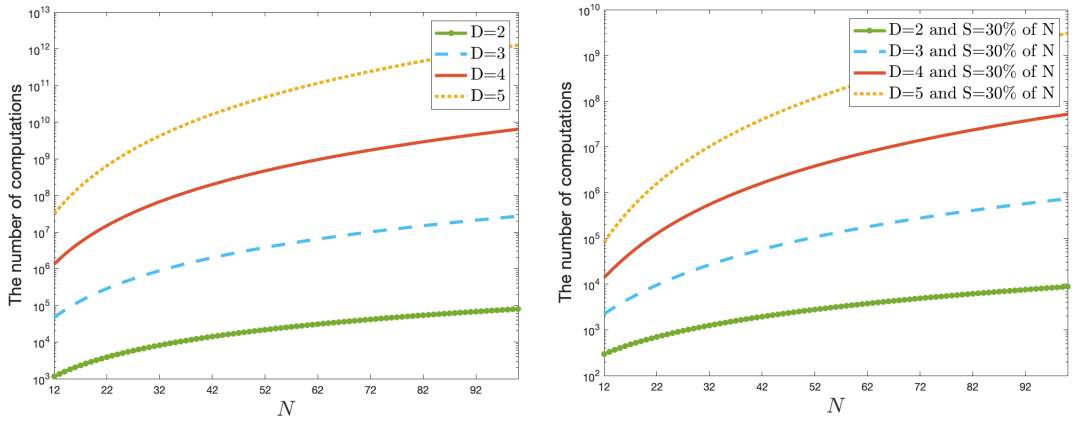
The existing combinatorial search algorithms differ in their complexity classes. BnB belongs to exponential complexity class; in the worst-case scenario. BnB can find the corresponding solution for N assignments by solving LP in polynomial time. On the other hand, the complexity class of PCS is not determined. In this section, we analyse the asymptotic computations of PCS and CSA algorithms. In [22], the PCS complexity formula is $\mathcal{O}(D^3\binom{N}{D})$, where D is the dimensionality of data and $\binom{N}{D}$ is the number of functions (capacity of the search space). For CSA, the time complexity is not given. To determine the complexity class $\mathcal{O}$, we need to simplify the factorial term $\binom{N}{D}$. $N, D \in \mathbb{N}$ and $1 \leq D \leq N$, then :

$$\binom{N}{D} = \frac{N!}{(N-D)!D!}. \tag{4.3.3}$$

since $D \geq 1$, then following inequality holds:

$$\begin{aligned} \frac{N!}{(N-D)!D!} &< \frac{N!}{(N-D)!} \\ &= N(N-1)\dots(N-D+1) \\ &< N \times N \times \dots \times N \\ &= N^D. \end{aligned} \tag{4.3.4}$$

Based on (4.3.4), PCS belongs to the polynomial complexity class of



(a) PCS (b) CSA

Figure 4.3.2: Time complexity of different D values. The time increases as the value of D increases.

order D, $\mathcal{O}\left(D^3N^D\right)$. The CSA algorithm is quite similar to PCS; however, the search process is divided into two stages. The first stage is the same as the PCS, where the difference is that the capacity of the search space is reduced by a pre-defined parameter S and the second stage is the swapping process. For the swapping process, each point $x_n \in N$ will be swapped with D points in the best combination of size D; this means that x_n will be swapped D times, and the swapping process takes $\mathcal{O}(DN)$. Thus, the computational complexity of CSA is $\mathcal{O}\left(D^3\binom{S}{D} + D^4N\right)$, where $S < N$ and $S > D$. The computational complexity of PCS and CSA depends on D, in which the greater the value D, the greater the computational complexity. However, in the classification problem, the dimensionality of data is often much lower than the number of points, $D \ll N$ (see Figure 4.3.2). Figure 4.3.3 shows the computational complexity of the algorithms.

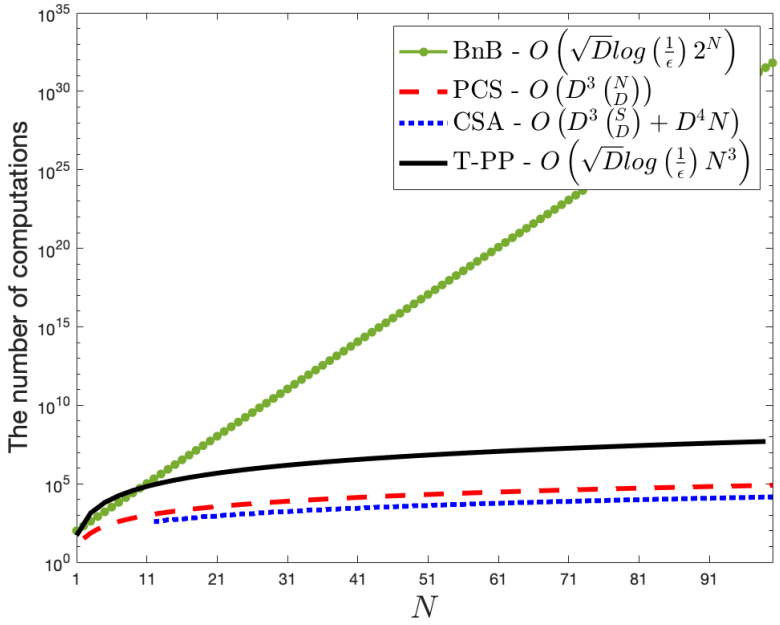


Figure 4.3.3: Number of computations when the number of points is increased for BnB, PCS,CSA, and T-PP on $2D$ data.

4.4 Experiments

4.4.1 Experiments parameter setup

A set of binary synthetic datasets was generated based on a different number of dimensions, $D = \{2, 4, 6, 8, 10, 12, 14, 16\}$ and a fixed number of points, N=80. The strategy we followed to generate these normally distributed data was as follows:

1. We defined two clusters, C_1 and C_{-1}, corresponding to label 1 and -1, respectively.

2. The centre point C_p of each cluster was generated randomly based on uniform distribution, and their features were distributed over different intervals depending on the number of dimensions, in which the cen-

tre of the cluster was generated in the interval [0,1] for $D = \{2, 4\}$, $[0, 0.75]$ for $D = \{6, 8\}$, $[0, 0.50]$ for $D = \{10, 12\}$ and [0,0.25] for $D = \{14, 16\}$. The sigma value associated with each centre point gradually increased from 0.1 to 1 when $D = \{2, 4, 6\}$ and from 0.3 to 1 when $D = \{8, 10, 12, 14, 16\}$. Increasing the sigma value gradually helps to increase the overlap.

3. The points were Gaussian distributed with a mean based on the feature values of centre point C_p and sigma value. Each label had the same number of points. The time limit for all comparative algorithms is 1.080×10^4 seconds.

The number of points used in the first stage of CSA was 40%, whereas in the second stage, all training points were included. The time limit of CSA was divided into 90% for the first stage and 10% for the second stage. The initial solution was obtained by minimising the hinge risk. In addition, this experiment was applied to small real datasets taken from the University of California, Irvine (UCI) machine learning repository [109]. The general information for the real datasets used is presented in Table 4.1, and the time limit was 1.440×10^3 minutes. All comparative algorithms were implemented using MATLAB 2020 running on a 2.3 GHz quad-core 10th-generation Intel core i7 processor and 32 GB RAM, and the original source code of algorithms was obtained from the author [110]. All datasets used in the experiment were normalised to have zero mean and unit variance for each feature, because the initial solution obtained by minimizing the hinge risk is affected if all input features do not have the same range.

	Number of points (N)	Dimension of dataset (D)
Haberman	306	3
Bupa	345	6
Heart	270	13
Cleveland	303	14
House votes84	435	16

Table 4.1: Information on datasets used in the experiment.

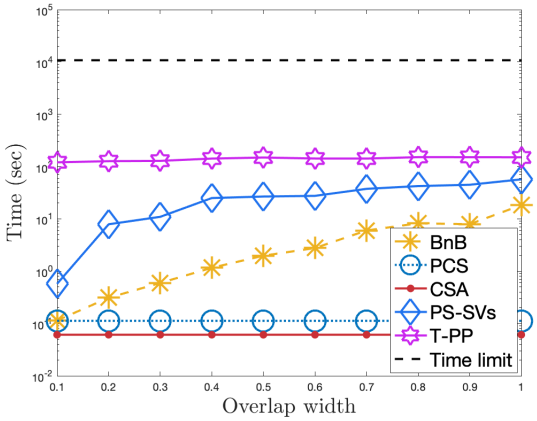
4.4.2 Results and discussion

Figures 4.4.1, 4.4.2, and 4.4.3 show the effect of increasing the overlap width on the time taken to reduce $\hat{R}_{01}$. BnB and PS-SVs were both affected by the amount of overlap between labels, whereas PCS, CSA and T-PP exhibited constant their time costs. The best and worst time complexities of BnB depend on the amount of overlap in the data, as the overlap width increases as the actual time complexity increases, reaching the time limit. PS-SVs depends on the number of SVs points; thus, it is clear that as the overlap width increases as the number of SVs points increases. PCS, CSA and T-PP were not affected by the amount of overlap because their search space was constant and did not change by changing the overlap width.

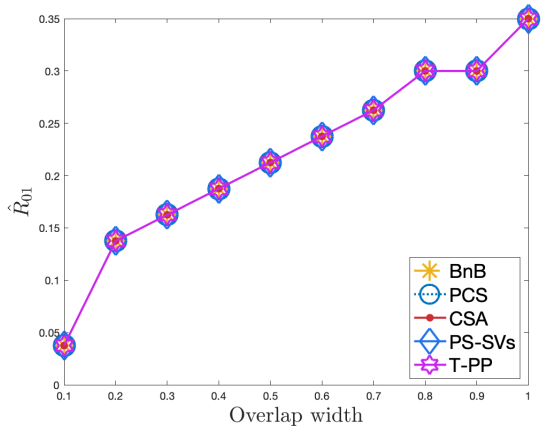
Obviously, The time complexity of PCS and CSA was only affected by the dimensionality of data, as shown in Figures 4.4.1, 4.4.2, and 4.4.3. The time complexity increases consistently as the dimensionality of data increases, and both reach the time limit when $D > 6$. The same finding observed with BnB could be due to the number of parameters that need to be optimised increasing as long as the features space is expanded. The complexity of T-PP and PS-SVs was not affected by the dimensionality of data as their solution spaces do not depend on the dimensionality.

In some cases, BnB and PCS could not terminate before the time limit; therefore, we cannot guarantee that the global optimal solution was obtained. In contrast, T-PP terminated fast and before the limit and achieved the same $\hat{R}_{01}$ when the other comparative algorithms terminated independently, as well as better $\hat{R}_{01}$ if the other algorithms terminated because of the time limit. This clearly shows that T-PP and its prioritised version PS-SVs can both find the global optimal solution in polynomial time.

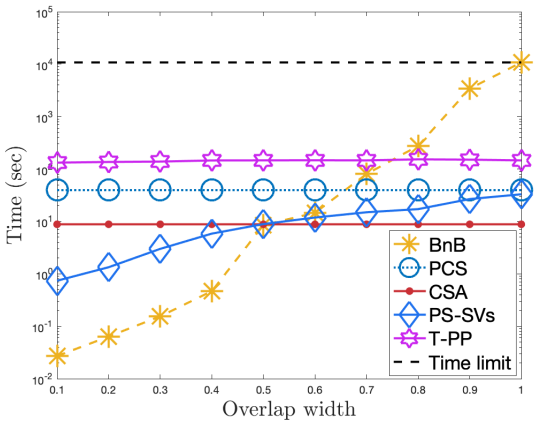
We obtained the same finding when we compared the time taken to reduce $\hat{R}_{01}$ on the real datasets, as presented in Table 4.2. PCS and CSA could only return the global solution when D was small enough (i.e. Haberman data). BnB terminated before the time limit when the $\hat{R}_{01}$ was very small (i.e. House votes84). In contrast, T-PP and PS-SVs achieved the same $\hat{R}_{01}$ as the other algorithms for the Haberman, House votes84 and Heart datasets, as well as a better $\hat{R}_{01}$ for the rest.



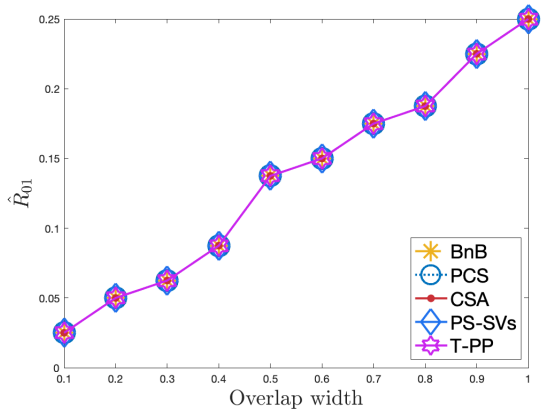
(a) The time taken by algorithms to minimise the $\hat{R}_{01}$ across different overlap rates, $2D$.



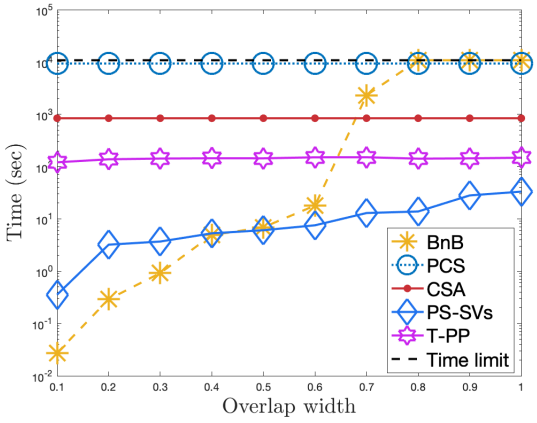
(b) The $\hat{R}_{01}$ achieved by algorithms across different overlap rates, $2D$.



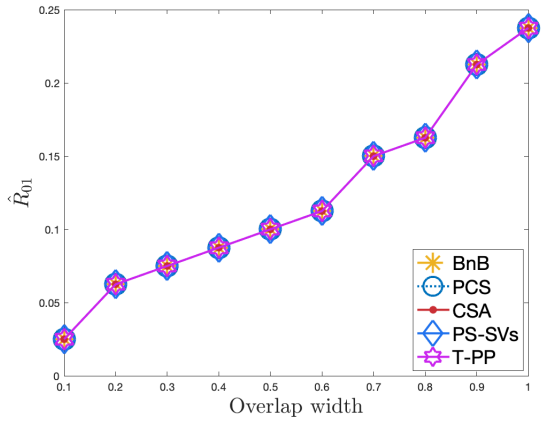
(c) The time taken by algorithms to minimise the $\hat{R}_{01}$ across different overlap rates, $4D$.



(d) The $\hat{R}_{01}$ achieved by algorithms across different overlap rates, $4D$.

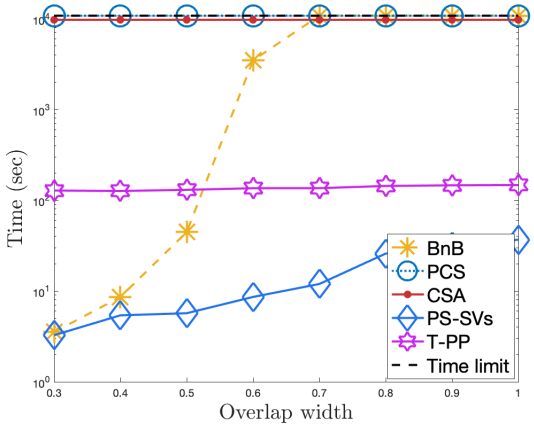


(e) The time taken by algorithms to minimise the $\hat{R}_{01}$ across different overlap rates, $6D$.

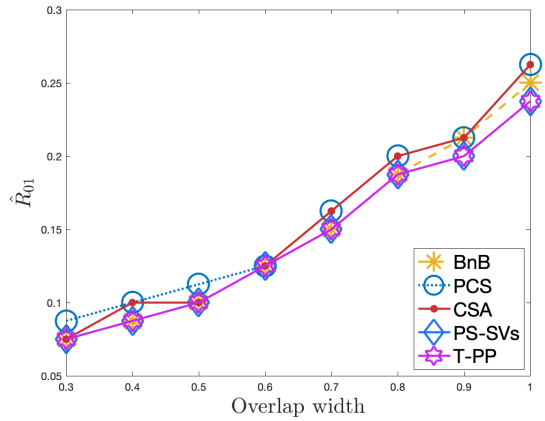


(f) The $\hat{R}_{01}$ achieved by algorithms across different overlap rates, $6D$.

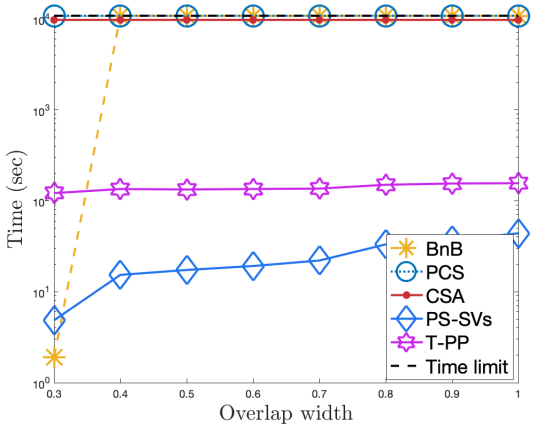
Figure 4.4.1: Computational time for minimising the $\hat{R}_{01}$ with gradually increasing overlap rates for $2D$, $4D$ and $6D$ Gaussian distributed data.



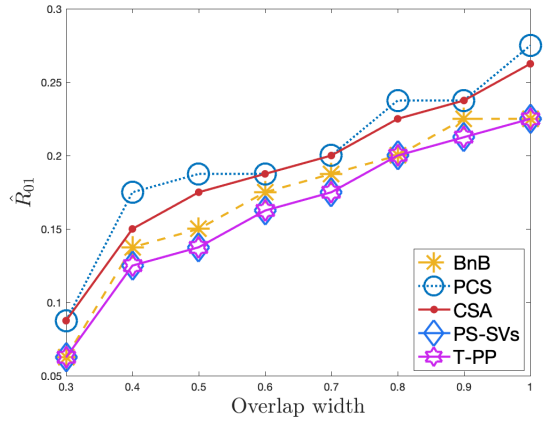
(a) The time taken by algorithms to minimise the $\hat{R}_{01}$ across different overlap rates, $8D$.



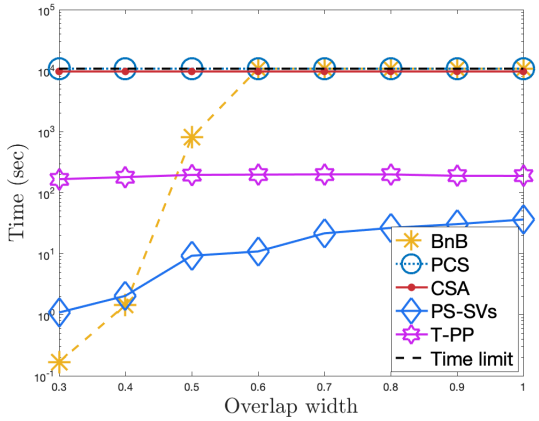
(b) The $\hat{R}_{01}$ achieved by algorithms across different overlap rates, $8D$.



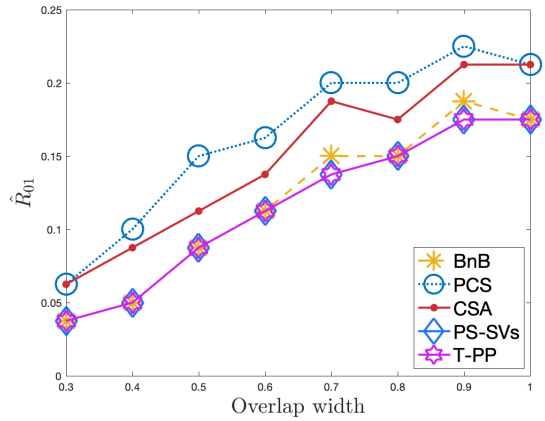
(c) The time taken by algorithms to minimise the $\hat{R}_{01}$ across different overlap rates, $10D$.



(d) The $\hat{R}_{01}$ achieved by algorithms across different overlap rates, $10D$.

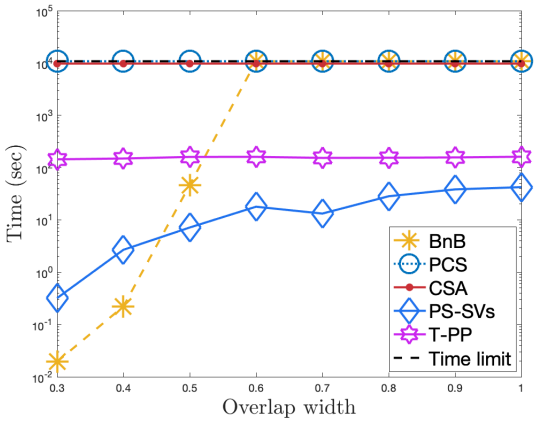


(e) The time taken by algorithms to minimise the $\hat{R}_{01}$ across different overlap rates, $12D$.

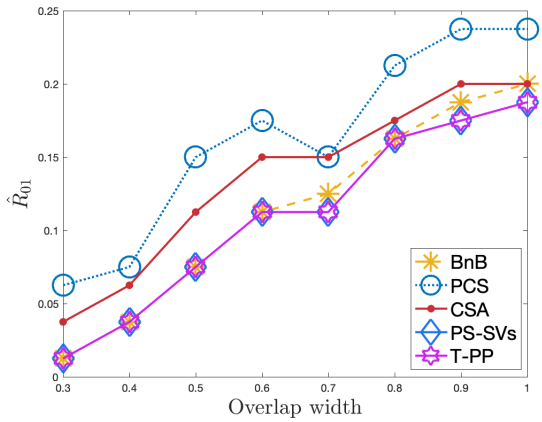


(f) The $\hat{R}_{01}$ achieved by algorithms across different overlap rates, $12D$.

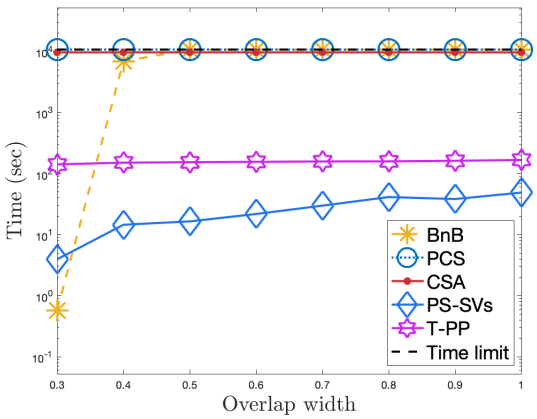
Figure 4.4.2: Computational time for minimising the $\hat{R}_{01}$ with gradually increasing overlap rates for $8D$, $10D$, and $12D$ Gaussian distributed data.



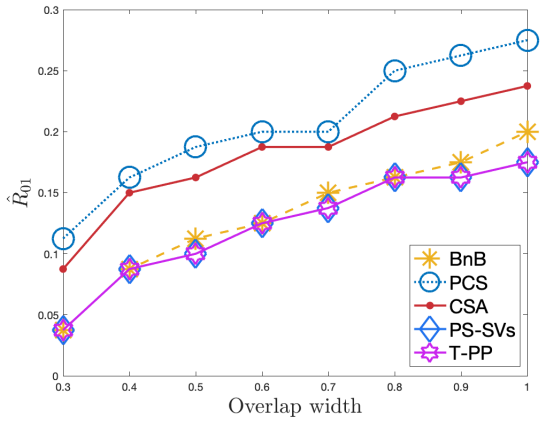
(a) The time taken by algorithms to minimise the $\hat{R}_{01}$ across different overlap rates, $14D$.



(b) The $\hat{R}_{01}$ achieved by algorithms across different overlap rates, $14D$.



(c) The time taken by algorithms to minimise the $\hat{R}_{01}$ across different overlap rates, $16D$.



(d) The $\hat{R}_{01}$ achieved by algorithms across different overlap rates, 16-D.

Figure 4.4.3: Computational time for minimising the $\hat{R}_{01}$ with gradually increasing overlap rates for $14D$, and $16D$ Gaussian distributed data.

	$\hat{R}_{01}$					Time (min)				
	BnB	PCS	CSA	T-PP	PS-SVs	BnB	PCS	CSA	T-PP	PS-SVs
Haberman	**0.216**	**0.216**	**0.216**	**0.216**	**0.216**	1.44 $\times 10^3$	17.8	2.6	1.74 $\times 10^2$	26.5
Bupa	0.246	0.252	0.246	**0.241**	**0.241**	1.44 $\times 10^3$	1.44 $\times 10^3$	1.44 $\times 10^3$	3.47 $\times 10^2$	64.7
Heart	**0.089**	0.111	0.107	**0.089**	**0.089**	1.44 $\times 10^3$	1.44 $\times 10^3$	1.44 $\times 10^3$	1.32 $\times 10^2$	3.6
Cleveland	0.145	0.172	0.172	**0.139**	**0.139**	1.44 $\times 10^3$	1.44 $\times 10^3$	1.44 $\times 10^3$	1.52 $\times 10^2$	4.4
House votes84	**0.012**	0.030	0.014	**0.012**	**0.012**	0.21	1.44 $\times 10^3$	1.44 $\times 10^3$	5.84 $\times 10^2$	0.20

Table 4.2: $\hat{R}_{01}$ and the actual time taken by the BnB, PCS, CSA, T-PP, and PS-SVs algorithms until termination or reaching the time limit. Bold text indicates that the algorithms obtain the same $\hat{R}_{01}$. The underlined text indicates that the algorithms reach the time limit.

4.5 Summary

In this experiment, direct 0–1 risk optimisation algorithms were compared in terms of time complexity. Their actual time complexities were examined under two factors: when the dimensionality of data and overlap width were increased. To summarise the main findings based on the empirical study, PCS, CSA and T-PP are not affected by the amount of overlap between two classes, because their solution space is constant and does not increase by expanding the overlap width. However, BnB and PS-SVs are affected by the amount of overlap, due to their solution space depends on the initial solution obtained by minimizing the hinge loss. In fact, the hinge risk increases with the increase of the overlap width. For PCS and CSA, it was clear that their time complexity depended highly on the dimensionality of data,

because their solution space increases with the increase of the dimensionality of data. T-PP and PS-SVs are our newly proposed algorithms, in which PS-SVs is a prioritising version of T-PP. Both could minimise 0–1 risk exactly in polynomial time and terminate before the time limit; in contrast, in most cases, BnB, PCS and CSA could not terminate independently.

Chapter 5

The interpolated local KNN algorithm

The current explanations for the success of interpolation classifiers are not fully understood. Understanding the success of interpolation classifiers requires understanding how they work. One of the simplest classification algorithms is the canonical K nearest neighbour (KNN); however, its performance is still dominated by the conventional ERM view. Therefore, the first new interpolation version of the KNN is introduced in this chapter; this algorithm achieves zero empirical risk and better generalisation performance compared with the classic KNN. The interpolated local KNN fits the training set exactly, even in the presence of noise and class overlap, where the noise and overlap points are defined during the training process and isolated from being used in predicting unseen points. Based on the experiments, the interpolated local KNN significantly outperforms the canonical KNN when the size of the dataset is large. In addition, the performance of the interpolated

local KNN is studied on three image datasets without any pre-processing or feature reduction/extraction process to ensure that the interpolation classifier performance does not overfit even with complex high-dimensional data. It is concluded that the interpolated local KNN outperforms the canonical KNN on the MNIST, CIFAR-10 and Fashion-MNIST datasets.

5.1 Introduction

The double descent curve is a phenomenon that was first noticed in [80] and overridden without a clear explanation or definition of why this curve appeared. With the recent development of the modern neural network, this phenomenon has attracted more attention in terms of the definition, depiction and study of the algorithm performance under two different generalisation regimes through which the double descent curve is formed; this interesting work was done by [111]. It has also been shown that the double curve is not specific for deep learning; it can be seen with other classification algorithms [111].

The double descent curve consists of two generalisation regimes—namely, the classical U-shaped regime and the interpolation regime. The classical U-shaped regime represents the classic view of the bias–variance trade-off. To avoid high variance (overfitting) and high bias (underfitting), the model complexity needs to be managed carefully to obtain an intermediate point that yields optimal performance. This can be done by tuning model parameters that act as regularisers to avoid fitting the data in favour of a smoother boundary [95]. This view has been studied in several models ranging from

a simple non-parametric model (i.e. KNN) to a complex parametric model (i.e. neural networks), concluding that a classical U-shaped curve appears in all models [112].

The interpolation regime represents a modern generalization view. Based on practical observation, the generalisation performance in the interpolation regime is better than the generalisation performance in the classical U-shaped regime [111, 113]. Some contributions have shown the connection between the classical U-shaped and interpolation through a double descent curve, so far the algorithms in which the double curve has been shown are RF [111], AdaBoost [90], and deep neural networks [113], however the rigorous explanation for the superiority of such interpolation classifiers is still under investigation.

The KNN algorithm is one of the simplest classification algorithms; however, its performance has only been discussed in references the classical U-shaped generalisation view, whereas its performance under the modern interpolation system has not been studied. In addition, the KNN is no longer interpolating for $K > 1$ due to the result of the majority vote for $K > 1$ is not guaranteed to achieve zero empirical risk in general [74, 97, 98]. The basic principle behind the KNN algorithm is simple: points are classified into a particular class based on the common class of their nearest points. The nearest points can be found by a distance metric T; here, the number of nearest points is managed by the K parameter, which controls the bias–variance trade-off in the classical U-shaped regime. The main contributions in this chapter can be summarised as follows:

- Analysing and comparing the performance of the canonical KNN al-

gorithm ($K > 1$) and 1 nearest neighbour (1NN) with/without the presence of noise and overlapping classes (section 5.2).

- Introducing the first interpolation KNN algorithm, the interpolated local KNN algorithm (section 5.3), and comparing the performances of the canonical KNN and the interpolation version of KNN on a set of real datasets, where the double descent curve is shown clearly (section 5.4).

- Studying the performance of the interpolated local KNN on three image datasets without any pre-processing or feature extraction to see whether the success of the current interpolation classifiers is a result of the feature extraction used or whether it can perform well using raw, complex, high-dimensional data (section 5.5).

5.2 The canonical KNN in the classical U-shaped and interpolation regimes

The 1NN rule is an example of an interpolating model that has been avoided in use because it has the worst generalisation performance of overfitting [114, 115]. Interpolating the training data to have zero training error can be done easily for any given training set, but interpolating the training data to have a small generalisation risk is a difficult task in general. This is because, the unseen points are not used in the training process, then improving their prediction is a challenge. Deep neural networks use a huge number of parameters, although the result of the over-parameterised model does not

cause overfitting; moreover, it was shown in [88] that over-parameterisation is necessary to have a smooth decision boundary. This could be a reason for the good performance of deep neural networks. In contrast, maximising the margin while interpolating the data yields a small generalisation risk, as suggested in [116]. In addition, the robustness of the local interpolation classifier in isolating and mitigating the influence of noise points helps to interpolate the data with a minimum generalisation risk, as shown in [90]. The 1NN rule does not consider any of these mentioned factors, whereas the KNN algorithm can create a smooth decision boundary by increasing the K parameter [117] and reducing the noise effect through the majority vote process [118]. Unfortunately, KNN is not an interpolated classifier. Therefore, the performance of KNN for $K \geq 1$ was studied in three cases—namely, noise-free and overlap-free, 5% noise and overlap-free and noise-free and 10% of overlap in classes.

5.2.1 The effect of noise and overlap on the interpolation regime

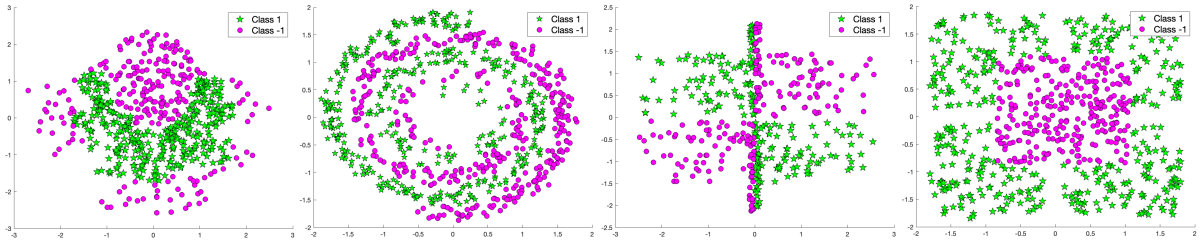
A simple experiment was conducted on non-linear binary synthetic datasets. The synthetic datasets were generated from the MATLAB script file given by [119], assuming the margin width and the density of points were variant at different local regions. This is a realistic assumption and reflects the most common non-linear real dataset. The generated data were free from noise and overlap in classes. The snapshot for the synthetic datasets used in this comparison is shown in Figure 5.2.1. To study the performance of

KNN in the presence of noise, 5% noise was added to the training set via random label-flipping of the training points. In the overlap case, the boundary points were determined by nearby points that had a different label; after that, label-flipping was performed on these points, and 10% of the training points overlapped. Figure 5.2.2 shows the dataset under the three cases. The performance of the interpolated classifier represented by 1NN and the non-interpolated classifier represented by KNN for $K > 1$ are discussed below.

We are interested in studying the generalisation curve in the absence of influencing factors (i.e. noise and overlap). We ask the following question: *Can the 1NN, as an example of the interpolation classifier, achieve a lower test error compared with KNN for* $K > 1$*?* Based on the results presented in Figure 5.2.3, the best test error was observed in the interpolation regime, which shows the performance of 1NN. All synthetic datasets follow the same trend; the test error increases consistently with increasing complexity parameter K. This indicates that the 1NN outperforms the KNN for $K > 1$ when the data are free from noise and overlap in classes.

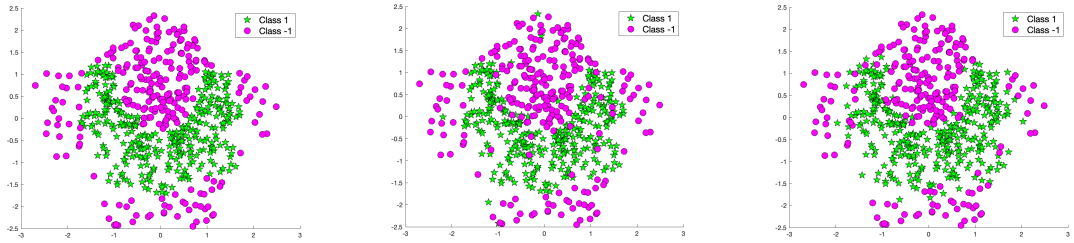
It is generally agreed that 1NN implies overfitting in the presence of noise and overlap [29, 117, 118]. To emphasise this common agreement on the 1NN and to show that the U-shaped curve can only appear in the presence of influencing factors, the performance of KNN for $K \geq 1$ is shown in Figures 5.2.4 and 5.2.5. The test error curve has the same trend in these cases; to avoid bad performance, increasing K reduces the effect of overfitting, resulting in better generalisation.

This gives us the insight that despite the presence of influencing factors, the superiority of the current interpolation classifiers could be related to their



(a) $M-1$ (b) $M-2$ (c) $M-3$ (d) $M-4$

Figure 5.2.1: Snapshots of the synthetic datasets. The data have the unique label $M-i$ for the purpose of referencing.



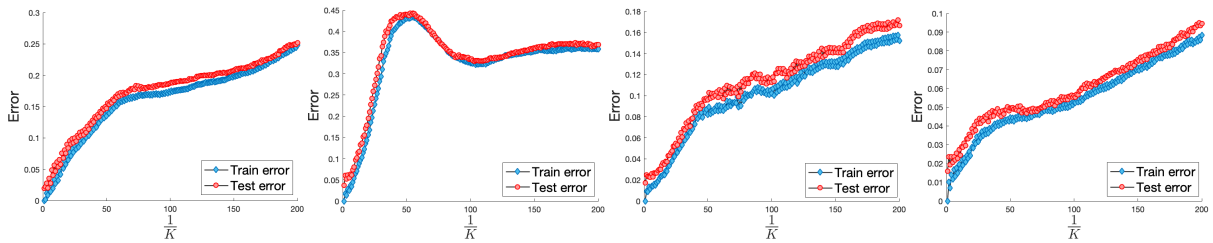
(a) Clean data, no noise or overlapping between classes exists.

(b) Noisy data

(c) Overlapping data

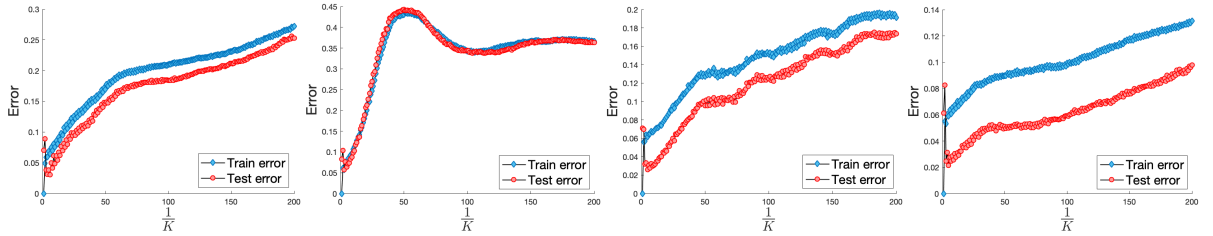
Figure 5.2.2: Snapshots of the clean, noisy and overlapping data.

ability to self-identify the noise and overlap area and then interpolate them locally in a way that prevents them from being used to classify unseen points. To verify this argument, a local interpolated KNN algorithm is proposed based on this point of view.



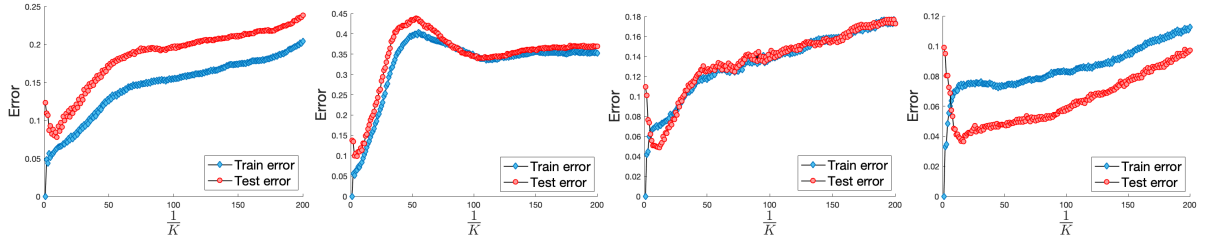
(a) $M-1$ (b) $M-2$ (c) $M-3$ (d) $M-4$

Figure 5.2.3: Average training and testing error reported on 10 random splits of the data. The data used are free from noise and overlap. The complexity of the model is measured as $\frac{1}{K}$; it decreases when the complexity parameter K increases. As clearly shown, both curves have the same trend, and the risk of misclassification increases as the K parameter increases.



(a) $M-1$ (b) $M-2$ (c) $M-3$ (d) $M-4$

Figure 5.2.4: Average training and testing error reported on 10 random splits of the data; the training set was injected with 5% noise. The test error curve is U-shaped. It decreases when K increases, reaching a minimum test error, and then increases again.



(a) $M-1$ (b) $M-2$ (c) $M-3$ (d) $M-4$

Figure 5.2.5: Average training and test error reported on 10 random splits of the data; 10% of the training points overlap. The overlap was done on the boundary points. The classic U-shaped curve clearly appears.

5.2.2 Challenges of interpolating the KNN algorithm

The canonical KNN uses a single K parameter selected globally by a cross-validation process. The decision boundary is determined by K nearest points without considering the spatial distribution or local density of the points. Since the local density of points is variant for a fixed value of K, the number of nearest points required to classify the selected training point correctly varies with the variety of the local density. It is expected that regions with low density may need a small value of K because of the low similarity between nearby points; they may not have the same label. To create an interpolated model by KNN, the K parameter must be defined locally for each training point considering the label of K nearest points. However, the main challenges are finding the optimal local K parameter for each training point and determining the value of local K for unseen points.

A few contributions have proposed improving the performance of the canonical KNN algorithm using a hybrid of K parameters; however, none of these studies have discussed the performance of local KNN in the interpolation setting [120, 121, 122, 123], and none of them have shown significant improvement on the canonical KNN except [123]. In [123], the local K parameter is determined for each training point via a complex cross-validation process, and the local K for an unseen point is the same as the local K of its first nearest point; in other words, it is not determined independently. In general, to find the optimal parameter among a set of pre-defined values, a validation process (i.e. cross-validation) is used; however, in defining a local parameter, the validation process usually does not provide a good es-

timate because of the limited number of points that will be trained locally and the computational complexity that will arise from training several local parameters for each point. In addition, training and optimising the local model separately may result in a solution that is not unique locally, which means that there is a set of local models that have the same optimal value, in which picking a random model may have a negative effect on global accuracy. Therefore, it is important to evaluate the global model after aggregating the local models, each of which uses different K parameters to see how these different local parameters interact with each other and maximise the overall accuracy. The optimal global model is the one that has the maximum accuracy; therefore, the optimal local K parameter for each point can be defined with respect to the performance of the global model.

Despite the challenge of defining the local K parameter for the training points under the guarantee that the combination of these local K parameters can produce an interpolation model, the biggest challenge lies in finding the local K parameter for unseen points under the guarantee of good generalisation performance. To achieve the dual objectives (fitting the training data perfectly and achieving a good generalisation of unseen data), we proposed a measure that computes the level of confidence in fitting the training point locally.

5.2.3 The relation between the confidence level and interpolation

The confidence of prediction counts the degree of certainty that the point will be in a particular class. The certainty of prediction can be calculated from a single point until its K nearest points, in which the level of confidence changes with the change of the K value, starting with a high level of confidence at $K = 1$ and decreasing with increasing K, $K \to N$. The level of confidence of a training point x_n for a given K nearest points can be measured as follows:

$$Cov(x_n, K) = \frac{1}{K} \sum_{i=1}^{K} I\left[y_n = y_i\right] \tag{5.2.1}$$

where, $I\left[.\right]$ is an indicator function that return 1 if the condition inside [.] is true, and 0 otherwise. $Cov(x_n, K)$ represents the confidence level of a particular point x_n for a given K, in which the confidence level is positive real value bounded as follows $0 < Cov(x_n, K) \leq 1$.

For a fixed K, the level of confidence can vary from one point to another depending on the local density of points and the number of nearest points that have the same label for a point of interest. The rate of change in the level of confidence also varies from one point to another. For instance, the level of confidence for the points close to the decision boundary may change faster than that for the points far away from the decision boundary. Therefore, to interpolate the data, the level of confidence should be considered when determining the local K for each point. To do this, the local K increment

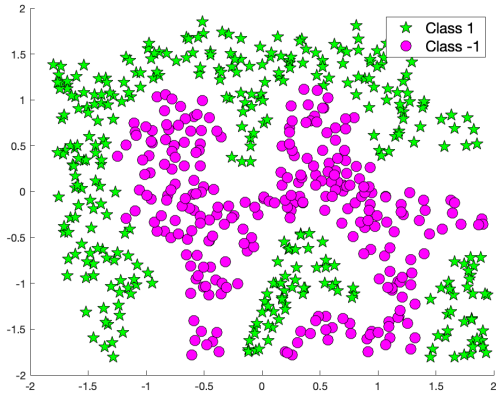
is coupled with a pre-defined confidence score; we prevent the local K from increasing if the result of this increment is a low confidence.

To ensure that the data are interpolated globally, we need to keep the overall confidence of points at a high level. An obvious example of this concept is 1NN, which is an interpolated algorithm where the level of confidence is 1 for all training points. For the canonical KNN, where the K is selected globally, the level of confidence of the training points is variant and low; therefore, there is no guarantee to interpolate. Figure 5.2.6 shows the relation between the level of confidence and the interpolation score (calculated as $1 - \hat{R}$) across different K values for the canonical KNN algorithm on 2 D synthetic data taken from [124].

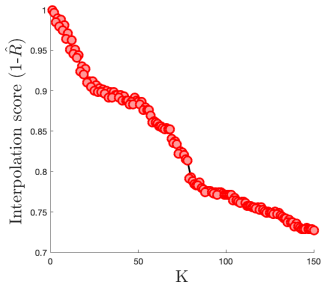
5.3 The interpolated local KNN classifier

This section explains the interpolated local KNN algorithm for training a set of local models, where these local models are combined to predict unseen points. In training, the local K parameter should be found for each training point x_n based on a given confidence score C. In the prediction, the K parameter is not used to define the number of nearest points; the K nearest points are defined locally and independently based on a new rule that uses ψ distance obtained during the training process, this process is described in Algorithm 5.3 from 6 to 9.

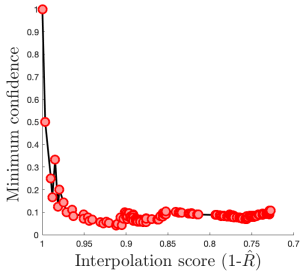
There are two main steps in fitting the training data. First, each training point should have a local K in which the average voting of the K nearest



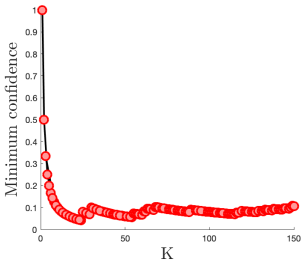
(a) $2D$ synthetic data are used to represent the relation between the level of confidence and the interpolation score. The local density of points and the margin width are varied in this data.



(b) Interpolation score and K



(c) Interpolation score and the minimum confidence



(d) The minimum confidence and K

Figure 5.2.6: Relationship between increasing the global K parameter and the interpolation, as in 5.2.6b, where the interpolation cannot be achieved if the level of confidence associated with each point is less than or equal to 0.50, as shown in 5.2.6c. To interpolate the data for $K > 1$, the level of confidence of each point should be >50, and the relationship is shown in 5.2.6d. A smaller K value results a larger interpolation score, therefore in order to achieve a large interpolation score the confidence level of all training points should be large. These relations have been computed for 5.2.6a, which has properties that represent most real-world data.

points does not exceed the pre-defined confidence score. The confidence level of each training point based on the K value can be computed in (5.2.1) and compared with a given confidence score C. For instance, the given confidence score is 0.90, and we have to compute the confidence level of the training point x_n when $K = 5$, the label of 5 nearest points is $y = \{1, 1, 1, -1, -1\}$; here, the confidence level of x_n is 0.60, which is < 0.90. Thus, we reject the local K value of 5 and make $K = 3$, because 3 makes the confidence level of $x_n \geq 0.90$.

We do not optimise the local K model for each training point separately. In fact, we optimise the global model. The global model combines all these local models, where each local model is for a single point. Therefore, the global K value is used to calculate the confidence level of all training points. If the current global K value makes the confidence level of some of the training points less than required, we set the previous value of K that does not exceed the required confidence score as a local K for a point. This means a global K value is used to set a local K value for each training point, where the local K value should make the confidence level of the training point $\geq$ the given confidence score. Then, all these local models are combined to form the global model that will be evaluated on the validation set to find the best global K value. Algorithm 5.1 (from line 5 to 12) describes how the local K parameter can be found for the training points.

Second, because determining the local K parameter for unseen points will be challenging since their label is not known, we cannot calculate the confidence level of classifying the query point into the true class. However, we can determine the local K parameter in another way by finding the number

of training points where their distance from a unseen/query point is $\leq$ the ψ distance associated with each training point. The ψ_n is a radius of the confidence area determined by the distance between the training point x_n and its kth point x_n^k. This means that during the training process, the ψ must be known to use it in the prediction process. The global model should be evaluated on the training set by ψ distance. This helps to define the training points that cannot be predicted correctly, even if they belong to the confidence area of other points. These points are more likely noise or in the overlap area. Thus, the use of these points may negatively affect the accuracy because they are useless when it comes to defining the decision boundary or predicting unseen points. Therefore, useless points are interpolated locally by setting $K = 1$ and the corresponding ψ distance to 0, Algorithm 5.2 (lines 6-13) describes how this process can be done. This means that all training points with a local $K = 1$ are only allowed to predict themselves and no point will be in their confidence area due to $\psi = 0$.

In prediction, the local K for the test point will be determined depending on the training points in which the test point is located inside their confidence area, defined by ψ. For instance, if a query point x_q belongs to five areas, then x_q will be predicted by the five nearest points, where the final label is assigned to x_q based on the majority vote. Algorithm 5.3 describes how the unseen set is predicted.

Algorithm 5.1 Local K parameter

Input: Training data(X,Y), a set of K, Confidence level C
Output: Confidence area defined by ψ

function FINDING THE CONFIDENCE REGION OF (X,Y)
    $D \leftarrow$ Get pairwise distance of X
    $\tilde{D} \leftarrow$ Sort D in ascending order
    $\tilde{Y} \leftarrow$ Get the label of the NN as sorted in ascending order
    for j in K **do**
        for i in N **do**
            Compute c_i as in (5.2.1) based on K_j and $\tilde{Y}_i$
            if $c_i \geq C$ **then**
                $\psi_i \leftarrow \tilde{D}_{K_j}$
            end if
        end for
    end for
    return ψ
end function

Algorithm 5.2 The evaluation process on the training set.

Input: Training data(X,Y), ψ
Output: Modified ψ, local K

function EVALUATION PROCESS (X,Y)
    $D \leftarrow$ Get pairwise distance of X
    $\tilde{D} \leftarrow$ Sort D in ascending order
    $\tilde{Y} \leftarrow$ Get the label of the NN as sorted in ascending order
    $\tilde{\psi} \leftarrow$ Get the ψ of the NN as sorted in ascending order
    for j in N **do**
        $l_j \leftarrow$ Get the labels of points from $\tilde{Y}_j$ if $\tilde{d}_j \leq \tilde{\psi}_j$
        $\hat{y}_j \leftarrow$ Do majority vote for l_j
        if $\hat{y}_j \neq y_j$ **then**
            $\psi_j \leftarrow 0$
            $K_j \leftarrow 1$
        end if
    end for
    return ψ, K
end function

Algorithm 5.3 The prediction process on unseen set.

Input: Training data(X,Y),Testing data($\tilde{X}$,$\tilde{Y}$), ψ
Output: Predicted label $\hat{Y}$

function CLASSIFICATION PROCESS FOR A GIVEN (X,Y)
    $D \leftarrow$ Get the distance between X and $\tilde{X}$
    $\tilde{D} \leftarrow$ Sort D in ascending order
    $\tilde{Y} \leftarrow$ Get the label of the NN as sorted in ascending order
    $\tilde{\psi} \leftarrow$ Get the ψ of the NN as sorted in ascending order
    for j in $\tilde{N}$ **do**
        $l_j \leftarrow$ Get the labels of points from $\tilde{Y}_j$ if $\tilde{D}_j \leq \tilde{\psi}_j$
        $\hat{y}_j \leftarrow$ Do majority vote for l_j
    end for
    return $\hat{Y}$
end function

5.4 Experiments

5.4.1 Experiments setting

A set of real datasets of different sizes was used in this experiment, and they can be found in UCI machine learning repository [109]; their general information is presented in Table 5.1. The dataset was split randomly 10 times into a 70% training set and a 30% testing set. The training model was chosen through a cross-validation process, where 10-fold cross-validation was used for $N \leq 5 \times 10^3$ and 5-fold cross-validation for $N > 5 \times 10^3$. The Euclidean distance was used as a metric distance, and the K parameter was chosen from a pre-defined set, $K \in \{1, 2, ..., 60\}$. The confidence level C was defined in advance as well, where $C \in \{0.95, 0.85, 0.75\}$. All data were normalised to have zero mean and unit variance. The experiments were conducted using a virtual machine reserved in Amazon EC2. The configuration of the instance is 256 GiB memory and 32 vCPUs; its name is r6i.8xlarge. To compare the

	Number of points (N)	Number of features (D)
Breast cancer	569	30
ILPD	583	10
CMC	1473	9
Employee	3305	4
Spambase	4597	57
Pulsar data	9273	8
Phishing	10922	30
Magic	19020	10

Table 5.1: General information for the real datasets used in the experiment.

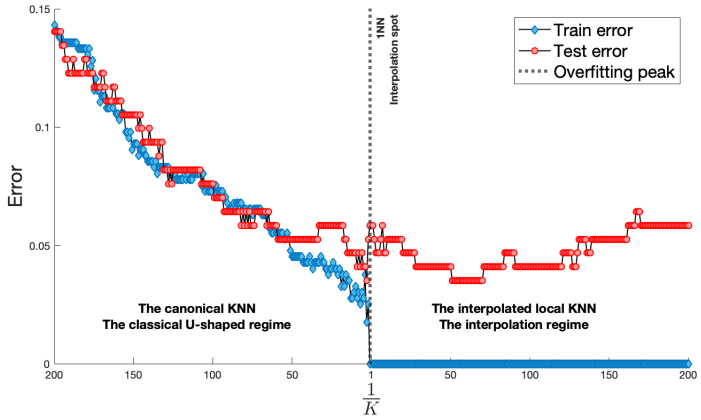
	Canonical KNN		Interpolated local KNN		
	Training error	Test error	Training error	Test error	P-value (test error)
Breast cancer	0.023 ± 0.010	0.042 ± 0.010	0.00 ± 0.00	0.039 ± 0.013	0.479
ILPD	0.276 ± 0.015	0.296 ± 0.035	0.00 ± 0.00	0.294 ± 0.035	0.583
CMC	0.291 ± 0.026	0.346 ± 0.014	0.00 ± 0.00	0.340 ± 0.016	0.249
Employee	0.00 ± 0.00	0.064 ± 0.010	0.00 ± 0.00	**0.060 ± 0.010**	0.022
Spambase	0.034 ± 0.037	0.096 ± 0.010	0.00 ± 0.00	**0.077 ± 0.010**	1.47×10^{-5}
Pulsar data	0.020 ± 0.001	0.024 ± 0.002	0.00 ± 0.00	**0.022 ± 0.002**	0.021
Phishing	0.00 ± 0.00	0.078 ± 0.005	0.00 ± 0.00	0.078 ± 0.005	0.343
Magic	0.136 ± 0.007	0.161 ± 0.005	0.00 ± 0.00	**0.151 ± 0.005**	1.28×10^{-6}

Table 5.2: Training and test errors of the canonical KNN and interpolated local KNN. There is a significant difference in performance if the P-value ≤ 0.05. Bold text indicates that the algorithm outperforms other algorithm, with a statistically significant difference in performance.

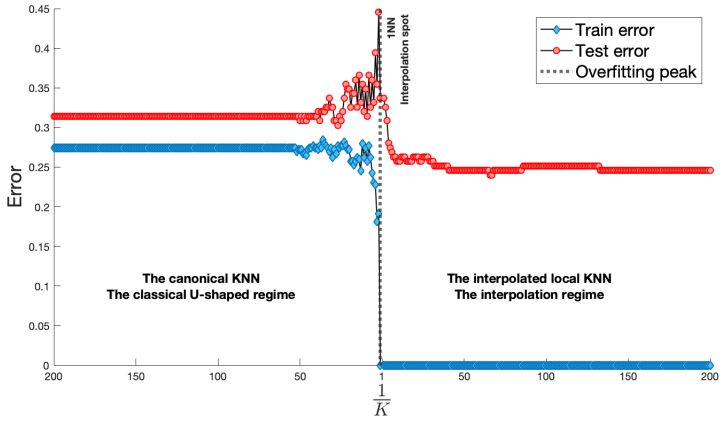
algorithm performance statistically, a paired t-test was used to compute the P-value.

5.4.2 Results and discussion

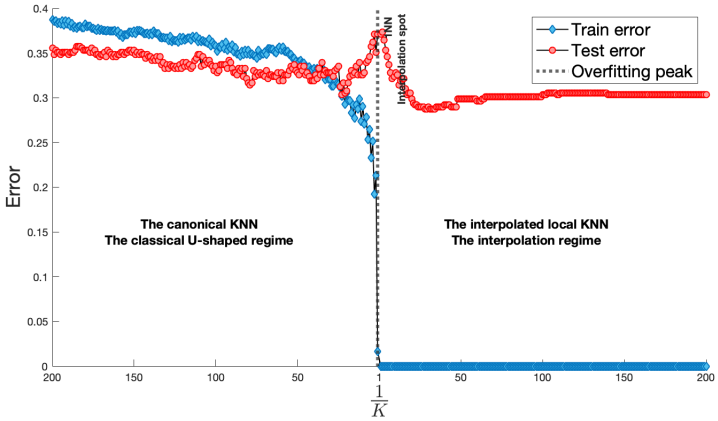
Table 5.2 represents the experimental results of the real data. There was no statistically significant difference in the performance on small data (i.e.



(a) Breast cancer

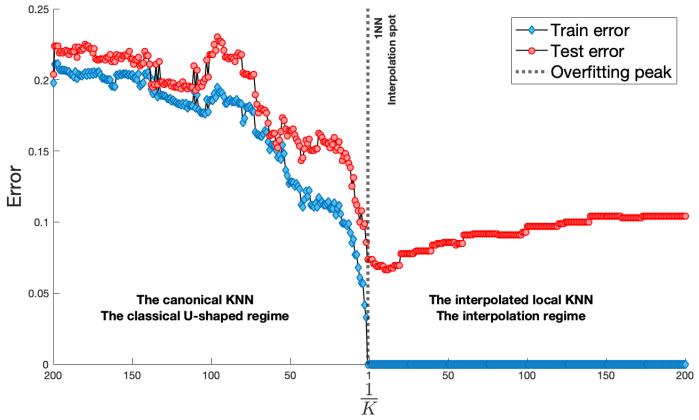


(b) ILPD

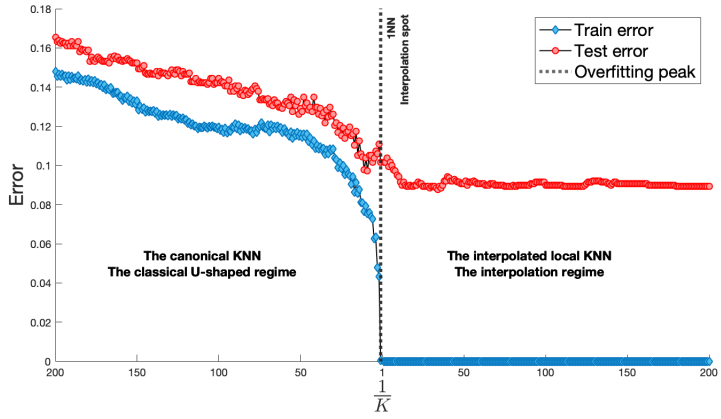


(c) CMC

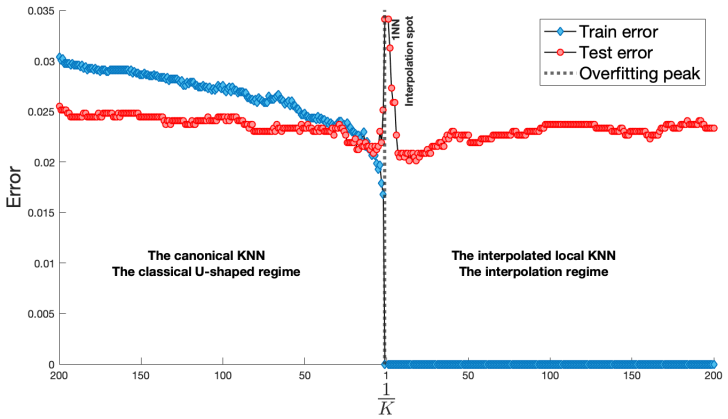
Figure 5.4.1: Double descent curve of the canonical KNN and interpolated local KNN on breast cancer, ILPD and CMC data. The 1NN algorithm is an intersection point. The interpolation regime starts when the training error reaches zero, and the classic U-shaped regime appears when the training error > 0. The complexity of KNN classifier is best measured as $\frac{1}{K}$, so that as K increases, the complexity of the boundary decreases.



(a) Employee data

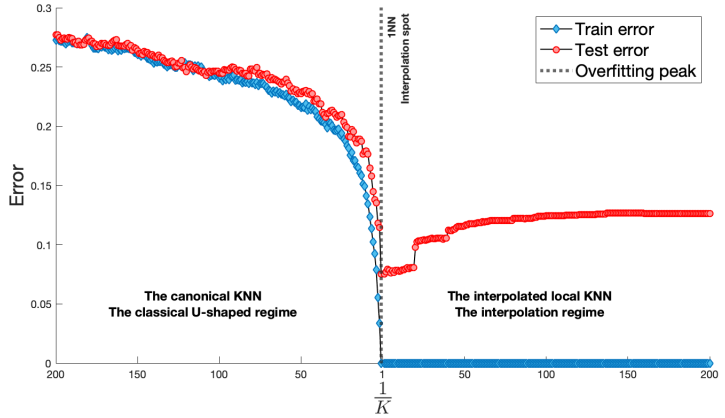


(b) Spambase

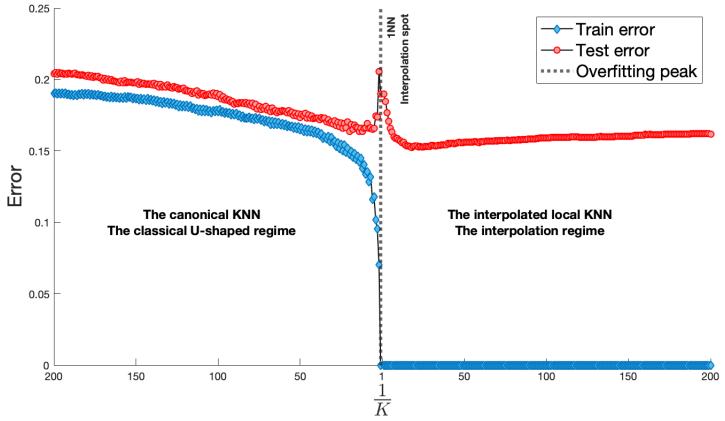


(c) Pulsar data

Figure 5.4.2: Double descent curve of employee, spambase and pulsar data connects the classic U-shaped regime (showing the performance of the canonical KNN) and the interpolation regime (showing the performance of the interpolated local KNN) via the 1NN algorithm.



(a) Phishing



(b) Magic

Figure 5.4.3: Double descent curve of phishing and magic data. The same set of K parameters was used in the canonical KNN and interpolated local KNN. The training and testing errors show the performance of the two algorithms, one based on the classic view of bias–variance trade-off and the other based on the interpolation view of fitting the training set exactly.

breast cancer, ILPD, CMC). Although the training set was fitted exactly, the interpolated model showed no overfitting. The superiority of the interpolated local KNN was observed when the size of the data, N, exceeded 3×10^3. This could be because increasing the number of points contributes to increasing the local density of points and thus contributes to finding a finer decision boundary. The best performance of employee and phishing data was achieved when $K = 1$, indicating that these data are free from noise and overlap, because the performance of the 1NN is destroyed in those cases. In contrast, the interpolated local KNN outperformed 1NN for employee data, and the same performance was achieved for phishing. Increasing K leads to reducing the complexity of the decision boundary and making it smoother, which positively reflects on the generalisation risk, this can be seen on employee data. A better interpolated classifier is the one with a smoother decision boundary, as pointed out in [90]. However, if the margin width is narrow, then a small value of K may yield a small test risk. This could be why there is no difference in the performance between 1NN and interpolated local KNN on phishing data.

Figures 5.4.1, 5.4.2, and 5.4.3 show the training and testing curves of the canonical KNN and the interpolated local KNN on the same set of K values. The U-shaped curve appeared in the classic regime for most of the datasets (except employee and phishing); the testing error increased dramatically after reaching the minimum value in most of the data. However, increasing the K in the interpolation regime led to different generalisation behaviours: reducing the test error (i.e. spambase), maintaining its value for a subset of K values (i.e. ILPD, CMC and breast cancer) and increasing the test error

slowly after passing the minimum (i.e. magic, phishing and employee data). However, the behaviour of the test error curve differs from the behaviour of the test error curve of the current interpolation classifies in such that increasing the complexity parameter contributes to decreasing the test error until reaching to a stable test error value.

The possible analysis of the behaviour of the test error curve in the interpolation regime could depend on the data itself in terms of the local density of points. If the local density of points is high for both classes, then increasing K leads to an increase in the confidence area of both classes. Thus, the confidence of prediction could increase, thereby decreasing the test error value. However, if the local density of points is low for any class, then increasing K leads to an increase in the confidence area of the high density of points, whereas the confidence area of the low density of points does not increase, due to the limited number of local points that have the same class. Therefore, increasing the K value makes the confidence area of the high density of points intersects with the confidence area of low density. The consequence of this is, that the test points belonging to the intersection area will be predicted to the class that has more local points, which could increase the test error after reaching the minimum value, and as we can see the amount of the increase in the test error is small, because the misclassified test points may come from the low density region.

It can be observed that the optimal K value in the interpolated local KNN was larger than the canonical KNN. According to our investigation in subsection 5.2.1, the canonical KNN chooses $K > 1$ if there is noise or overlap (or both). However, the strategy used in the interpolated local KNN prohibits

including these points in prediction, thereby obtaining a wide margin as a result of isolating overlap points. For a wide margin, the low test error in the interpolation regime can be reduced by making the decision boundary smoother, which could be done with a large K value.

5.5 The canonical KNN and the interpolated local KNN classifiers in image classification

Digital image classification is a core problem in computer vision. Recently, a great development has been witnessed in terms of improving algorithms to be able to classify images into the correct class with high precision. A digital image is a photograph taken by a camera; its content is represented by a matrix of pixels arranged in fixed rows and columns. Each cell in the matrix represents the pixel, which is the discrete integer value ranging between 0 and 255. The value of the pixel represents the colour depth, which can be a single value (grayscale) or a set of values (colour scale) [125]. For the purpose of classification, we convert the image into a single vector containing pixel values.

Despite the numerous classification algorithms that have shown superior performance over tabular real data, algorithms' performance in image classification relies on their ability to extract discriminant features, which have underlying non-linear complex relationships by nature [126, 127]. Therefore, the construction of a training model requires considering the spatial correlation of pixel values; thus, current algorithms should be adapted for accurate

classification [128].

Until now, the best and most accurate interpolation algorithm has been convolutional neural networks (CNNs). CNNs are state-of-the-art deep neural networks that are commonly used in image classification because of their superiority in achieving a low generalisation error and perfect fitting of the training set [88, 116]. The complex architecture of CNNs makes it difficult to understand how they work and why they succeed in the interpolation setting. In addition, the number of hyper-parameters required to build the CNNs is large and necessitates a validation process to adjust a set of hyper-parameters in favour of a good generalisation. Increasing the number of layers affects the generalisation error, in which the number of layers should be increased with the increase of the complexity of data [87].

In fact, a clear understanding has not been established of whether the success of the interpolation classifier depends on the feature extraction/reduction process or whether it can perform very well without it, where the role of feature extraction/reduction is to reduce the complex representation of the data, thereby reducing the test error significantly [129].

The success of canonical KNN in image classification is highly related to the complexity of the data [130]. This can be seen from the large contrast in KNN performance between classifying grayscale images and colour images. To obtain a good performance on the image data, it is necessary to find a good representation of the data before feeding it to the KNN algorithm [130, 131]. In addition, the canonical KNN is a more flexible algorithm that can work on linearly and non-linearly separable data without the need to map the data into higher dimensions or increase the number of layers to fit non-

linear data. No training process is required, and minimal hyper-parameters are needed for tuning [132].

Our goal is to evaluate the performance of the interpolated local KNN without any kind of pre-processing step (i.e. normalisation) or feature extraction/reduction technique (i.e. filter) to see whether the interpolation classifier outperforms the non-interpolation classifier on image data, which are by nature complex and high-dimensional data that can be easily overfitted. We are interested in answering the following question: *Does interpolating image data without feature extraction/reduction cause overfitting and then, bad generalization?* Answering this question can help to elucidate whether the success of CNNs as interpolation classifiers is somehow related to the algorithm or whether its success is related to the feature extraction/reduction used during training the data [73, 87]. If the interpolated local KNN does not overfit and achieve better generalisation performance than the canonical KNN does, then combining the feature extraction/reduction with the interpolated local KNN could produce a classifier that is competitive with CNNs without the need for tuning a huge number of hyper-parameters and training the model for a long time.

5.5.1 Benchmark image datasets and the experimental setting

Three image benchmark datasets are used in our experiment. A brief description of each dataset is given below, and Figure 5.5.1 gives snapshots of the image datasets.

1. The MNIST image dataset

   MNIST is a multiclass grayscale image dataset introduced by [133]; it consists of 7×10^4 handwritten digits with 6×10^4 for training and 10^4 for testing. The size of the digital images is 28×28 pixels, converted to a matrix with 784 features, where each vector x_n is labelled by a digit number $y_n \in \{0, 1, ..9\}$.

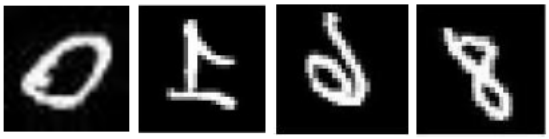
2. The CIFAR-10 image dataset

   CIFAR-10 is a multiclass colour-scale image that consists of 6×10^4 images of 32×32 RGB pixels; it was created by [134]. Each image is represented by a row in a matrix, where the number of features is 3.072×10^3 and assigned a single class from a set of 10 classes, $y \in \{0, 1, ..9\}$. In addition, 5×10^4 images are used for training and 10^4 for testing.

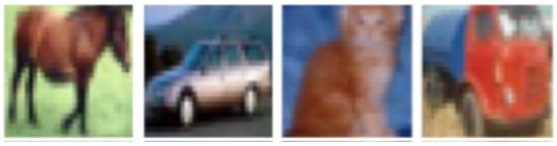
3. Fashion-MNIST image dataset

   Fashion-MNIST is a multiclass grayscale image dataset created by [135]. It consists of 7×10^4 fashion products categorised into 10 classes. The image size is 28×28 pixels, converted to 784 features. The training set has 6×10^4 data points, and the testing set has 10^4 data points.

The distance metric used in this experiment is the Minkowski metric, where $p = 3$ (also known as $L3$ distance), which has been found to have a low test error in practice [136]. A set of K parameters, $K \in \{1, 2, ..., 200\}$, and a set of confidence scores, $C \in \{1, 0.90, 0.80, ..., 0.0\}$ are included. The performances of the canonical KNN and the interpolated local KNN across



(a) MNIST



(b) CIFAR-10

(c) Fashion-MNIST

Figure 5.5.1: Small points for the image data used in the experiment.

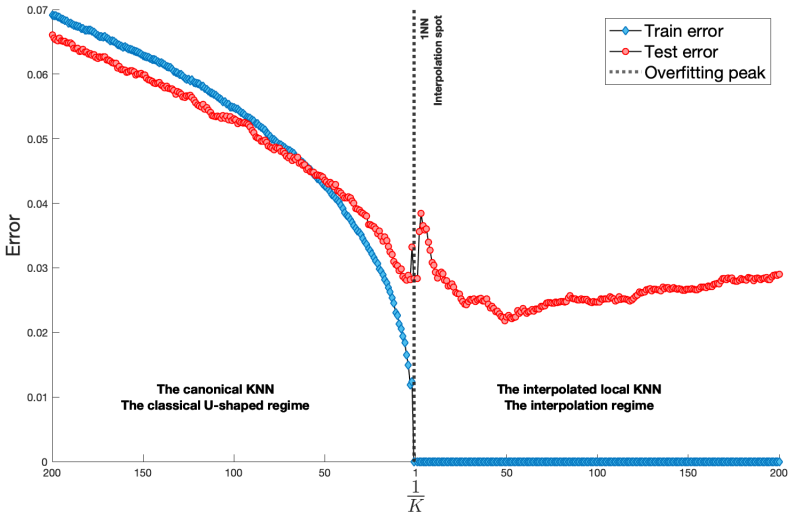
all these hyper-parameters are represented by a set of figures.

5.5.2 Results and discussion

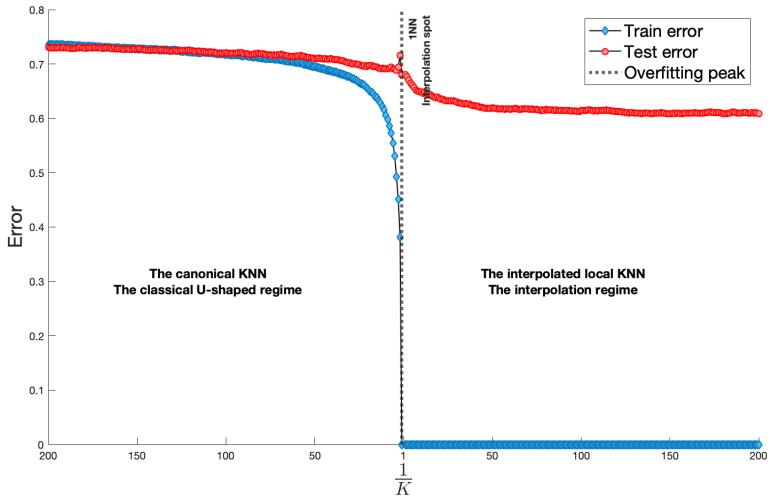
It is well known that the KNN algorithm is affected by the curse of dimensionality, then interpolating this sort of data without feature manipulation could have a negative effect on the generalization performance, however this is not what was observed when comparing the canonical KNN and interpolated local KNN. The interpolated local KNN outperforms the canonical KNN on all three images. This means that interpolating high-dimensional data to achieve zero empirical risk does not have a harmful effect on the generalization. Based on the double descent curves presented in Figure 5.5.2, the best test error was observed in the interpolation regime. MNIST and CIFAR-10 had the best performance for the canonical KNN when $K = 1$;

the test errors were 0.028 and 0.681, respectively, for these datasets, while it was 0.152 for Fashion-MNIST when $K = 5$. The performance of the interpolated KNN classifier did not overfit even without doing any sort of feature extraction/reduction; the best test errors for MNIST, CIFAR-10 and Fashion-MNIST were 0.021, 0.608 and 0.136, respectively.

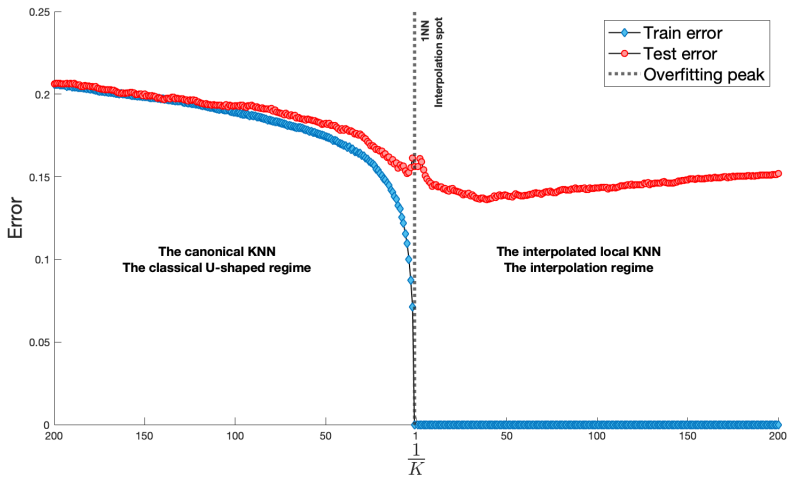
Figures 5.5.3, 5.5.4, and 5.5.5 show the descent test curve by changing the confidence score; here, decreasing the confidence score yields a better test error, as shown in Figure 5.5.6. This observation is different from what was observed with real data; the best test error was obtained for the real data when the confidence score was at least ≥ 0.75. This could be due to the high-dimensionality of image data, in which the points were spread out in the large feature space. If the confidence score is large, then it may be that the confidence areas for the training points do not intersect; thus, unseen points may be predicted by one or a few training points. To improve the classification accuracy, the confidence areas of the training points need to be intersected to obtain more confidence in prediction. To achieve this goal in the high-dimensional data, the confidence scores may be varied. For instance, points close to the margin of other classes may need a confidence score ≥ 0.65, whereas points far away from the margin may have a confidence score of 0.75. This can be done implicitly when the confidence score is low, as shown in Figure 5.5.6.



(a) MNIST

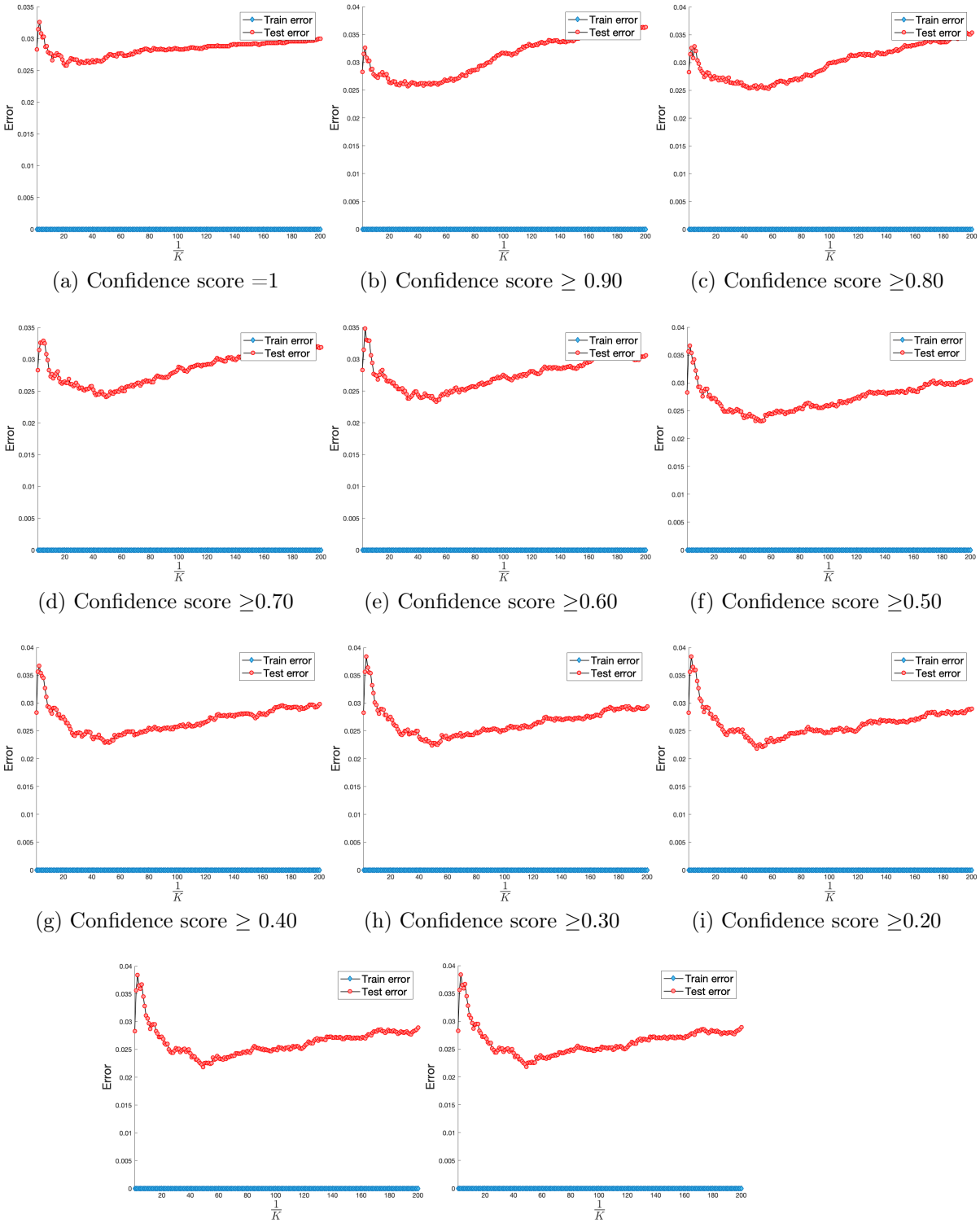


(b) CIFAR-10



(c) Fashion-MNIST

Figure 5.5.2: Double descent curve of a set of image data.



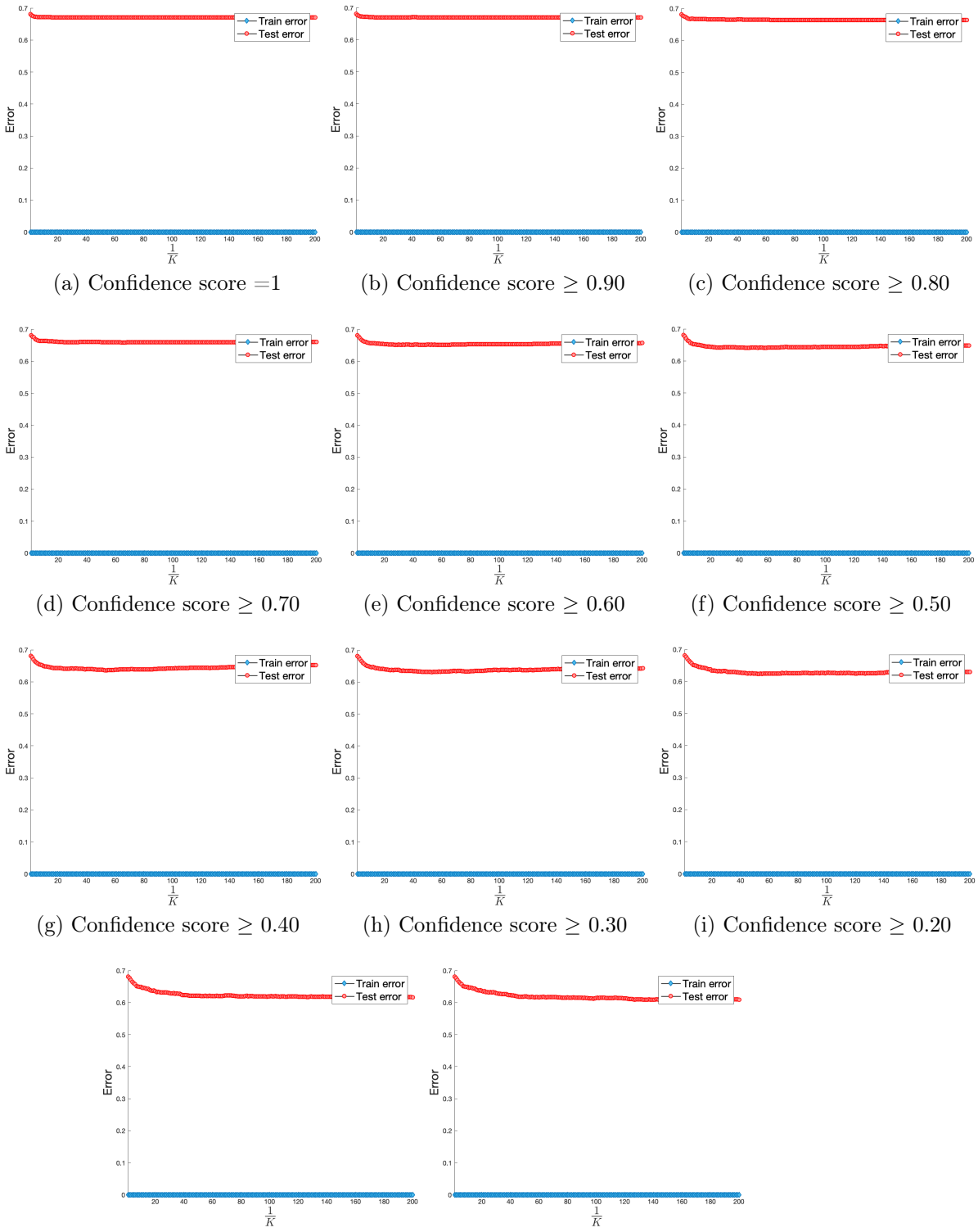
(a) Confidence score =1 (b) Confidence score $\geq$ 0.90 (c) Confidence score $\geq$0.80

(d) Confidence score $\geq$0.70 (e) Confidence score $\geq$0.60 (f) Confidence score $\geq$0.50

(g) Confidence score $\geq$ 0.40 (h) Confidence score $\geq$0.30 (i) Confidence score $\geq$0.20

(j) Confidence score $\geq$0.10 (k) Confidence score $\geq$0.0

Figure 5.5.3: Training and testing error curves of the interpolated local KNN across different confidence scores for MNIST data.



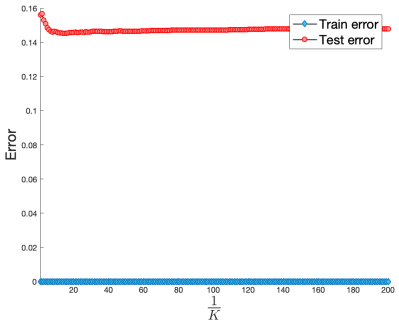
(a) Confidence score =1 (b) Confidence score ≥ 0.90 (c) Confidence score ≥ 0.80

(d) Confidence score ≥ 0.70 (e) Confidence score ≥ 0.60 (f) Confidence score ≥ 0.50

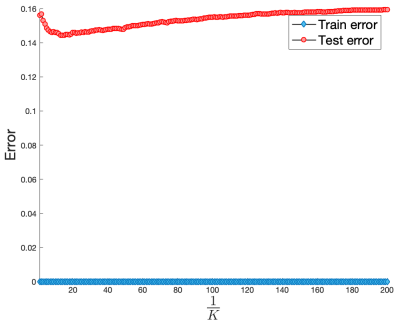
(g) Confidence score ≥ 0.40 (h) Confidence score ≥ 0.30 (i) Confidence score ≥ 0.20

(j) Confidence score ≥ 0.10 (k) Confidence score ≥ 0

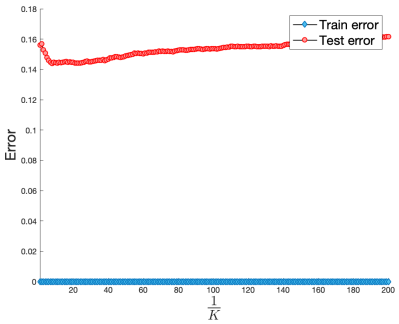
Figure 5.5.4: Each mini-figure represents the performance of the interpolated local KNN at different confidence scores for CIFAR-10.



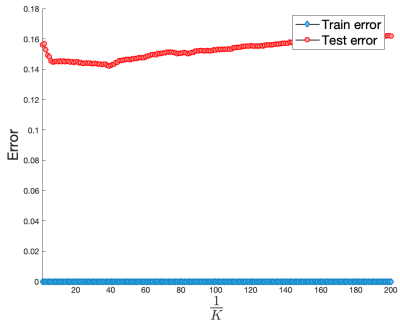
(a) Confidence score =1



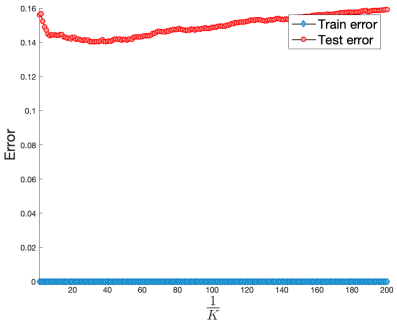
(b) Confidence score ≥ 0.90



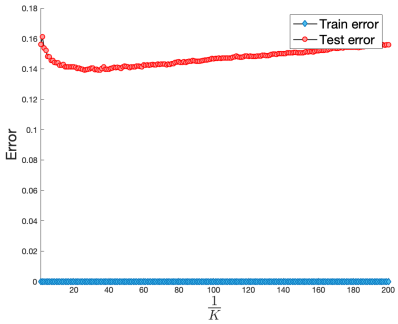
(c) Confidence score ≥ 0.80



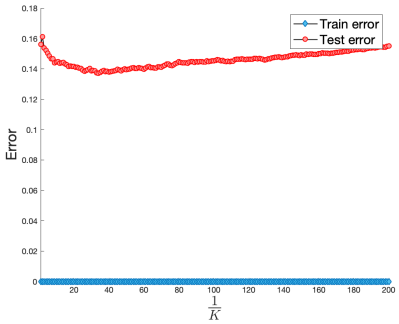
(d) Confidence score ≥ 0.70



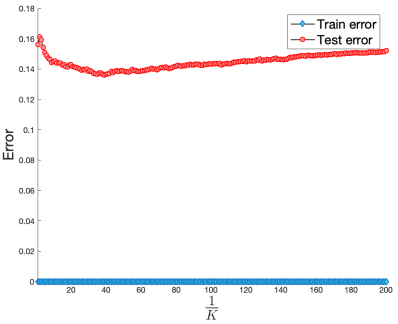
(e) Confidence score ≥ 0.60



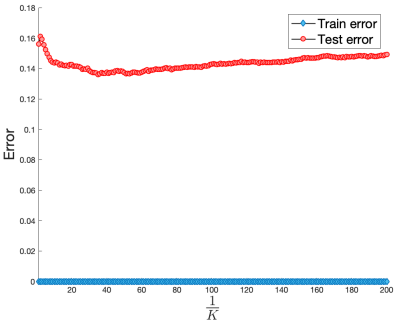
(f) Confidence score ≥ 0.50



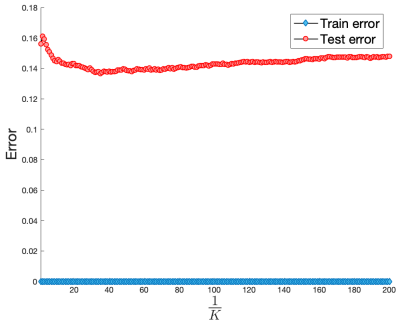
(g) Confidence score ≥ 0.40



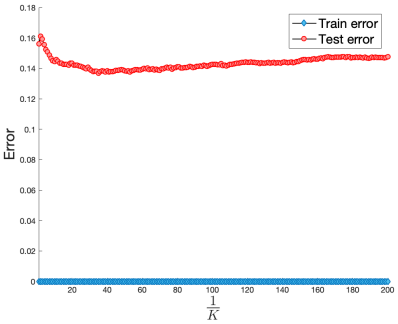
(h) Confidence level ≥ 0.30



(i) Confidence score ≥ 0.20

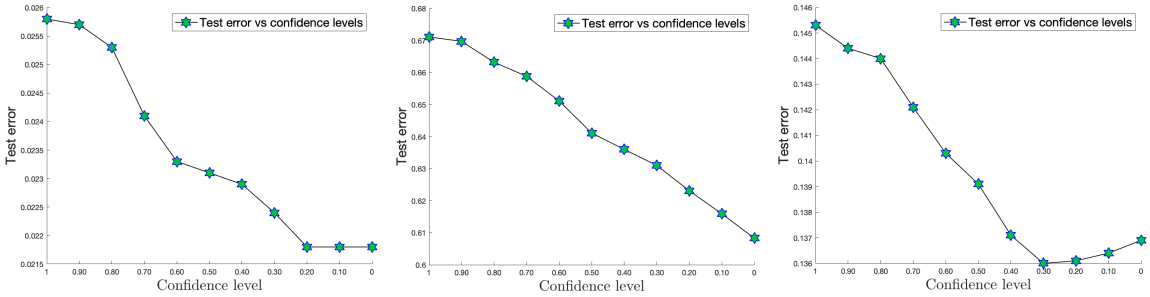


(j) Confidence score ≥ 0.10



(k) Confidence score ≥ 0

Figure 5.5.5: Training and testing results of the interpolated local KNN across different values of the K parameter and a fixed confidence score per mini-figure for Fashion-MNIST.



(a) MNIST (b) CIFAR-10 (c) Fashion-MNIST

Figure 5.5.6: Generalisation risk according to changing the confidence score for the image datasets; the lower the degree of confidence, the lower the risk rate.

5.6 Summary

In this chapter, a new interpolated local KNN was introduced that can achieve a low generalisation error compared with the canonical KNN algorithm on a set of real datasets and image data. The strategy used in the proposed interpolation algorithm was measuring the level of confidence for the training points while increasing the K parameter, where the confidence level for each training point should not exceed the pre-defined confidence score. A global interpolated model resulting from aggregating all local models is used to determine the noise and overlap points to isolate them from the use in the classification process. After evaluating the global interpolated model on the training set, it can be used on the unseen points. Therefore, overfitting does not occur because of the use of noise, and overlap points in classification are prevented. This could be the reason for the superiority of existing interpolation classifiers.

The interpolated model outperformed the non-interpolated model when the data were relatively large. Despite the nature of image data, which always exhibits high dimensionality and complexity, interpolation without pre-processing or feature extraction/reduction can produce a double descent curve, where the best generalization is achieved in the interpolation regime.

Chapter 6

Conclusion and future work

6.1 Summary of work

This thesis discussed the optimality of 0–1 loss for classification problems in three different situations—namely, the optimality of the 0–1 loss based on the use of surrogate loss functions on linearly separable data, minimising the 0–1 loss directly on overlapping data and studying the performance of the new interpolation local classifier based on the modern ERM.

The main focus in chapter 3 was on the use of the surrogate loss functions as an alternative loss for minimising the 0–1 loss. In previous studies, it was shown that surrogate loss functions provide an optimal solution for 0–1 loss under the guarantee of minimising the calibrated loss functions, which works as a link between 0–1 loss and the surrogate loss function. In practice, we cannot minimise the calibrated loss functions because the conditional probability of classifying the point into the true class is not given; therefore, it is difficult to verify the theorem in practice. We introduced a simple

theory in which minimising the surrogate loss functions yielded minimising the 0–1 loss function if the optimal functions had the same 0–1 empirical risk. In addition, we proved that minimising the hinge risk is equivalent to minimising the 0–1 risk if the data are linearly separable.

In chapter 4, we compared the current direct 0–1 risk minimisation algorithms in terms of their theoretical and practical time complexity. The practical time complexity of these algorithms was compared in terms of the dimensionality of data and amount of overlap between classes. PCS and CSA both belong to the polynomial complexity class of order D, whereas BnB has exponential time complexity in the worst-case scenario. Based on the experimental results, the actual time of PCS and CSA increased as the dimensionality of data increased, whereas the actual time of BnB depended on the amount of overlap between classes, in which the best runtime was achieved on a small overlap.

We introduced a new heuristic direct 0–1 risk minimisation algorithm that we considered to return the global optimal solution in $\mathcal{O}(\sqrt{D}\log(\frac{1}{\epsilon})N^3)$ polynomial time. T-PP aims to find the optimal solution by fitting a decision boundary between each pair of points, each in a different class. Based on the results, T-PP and its prioritised version PS-SVs find the same empirical risk as the other algorithms if the other algorithms terminated before the time limit, and they find a better empirical risk if the other algorithms reached the time limit. T-PP has a constant time when the dimensionality of data or overlap width increases, whereas PS-SVs is affected by increasing the overlap width, although it is still better than T-PP in terms of actual time.

Our understanding of minimising the empirical risk has changed after introducing a new modern generalisation view. Zero empirical risk does not cause overfitting, and the intuition behind the success of the classification algorithm under this view is still unclear. The main contributions in chapter 5 were understanding the interpolation model for the canonical KNN, introducing a new interpolation local classifier based on KNN and comparing the generalisation performance of the canonical KNN that works in the classic generalisation view and interpolated local KNN that works in the modern generalisation view. This was done on a set of real data and three common image datasets.

The conclusions that were obtained through the experiments showed that 1NN has better generalisation than KNN for $K > 1$ if the data are free from noise and overlap. The classical U-shaped generalisation curve does not appear in this case, whereas the presence of noise and overlap in the data negatively affects 1NN, and the classical U-shaped generalisation does appear in these cases. However, in the interpolation setting, the interpolated local KNN showed superior performance on large real data and achieved better performance on small data without significant improvement.

Moving to a more complex situation in which the raw image data is interpolated without any process of the features, the current interpolation algorithms conducted feature extraction/reduction on the image data, leaving a question of whether feature extraction is mandatory for interpolating the image data. Based on the experiment, image data interpolated by a simple algorithm that is highly sensitive to the variance in the feature values and the number of features. Interpolated classifier does not cause overfitting; the

test error can be minimised, while the empirical risk is exactly zero. The interpolated local KNN minimised the test error across all confidence scores.

6.2 Future research directions

There are still open directions that need to be investigated to optimise the 0–1 loss directly. Little theoretical development was provided in this thesis; the main focus of the study was to prove the feasibility of optimising the 0–1 loss directly in a practical way. In addition, optimising the 0–1 loss function was previously ignored because of the difficulty of solving this problem. Through the experiments done in this thesis, the ability to optimise 0–1 loss in polynomial time complexity was proven empirically, which is an important step in solving classification problems using its original loss function rather than surrogate loss functions. The proposed research directions that need more attention are as follows:

- The theoretical part related to the 0–1 loss function was dealt with in the literature by proving that 0–1 risk minimisation is NP-hard. Then, Cover's theory [99] provided a measure to count the number of assignments that can be separated by a linear function. However, Cover did not provide an algorithm that could find all these linearly separable assignments or define the complexity of such an algorithm. Based on our proof of the complexity of PCS, it can be seen that the time complexity of going through all linearly separable assignments given by Cover's measure is also a polynomial of order D. The intuition behind PCS allowed us to classify PCS as a direct 0–1 loss minimisation;

however, this has not been proven theoretically. This means that there is a gap between theory and practice in terms of finding an exact 0–1 risk minimisation algorithm based on Cover's theorem and proving the PCS algorithm is an exact 0–1 risk minimisation theoretically.

- We introduced T-PP as a heuristic algorithm to minimise the 0–1 risk directly; however, this has still not been proven theoretically.

- Despite the theoretical time complexity of PCS being polynomial, its heuristic strategy does not use the time effectively. Even in its prioritised version, CSA cannot terminate before the time limit in most cases; therefore, we suggest restricting the search space for CSA to SVs points obtained from minimising the hinge loss. This means that the search space of PS-SVs and CSA will be the same, with the difference relating to the approach finding the optimal function.

- In image classification, the success of interpolation classifiers is more likely related to their ability to reduce the dimensionality of the data, in which the abstract features describe the given training set 100% accurately without losing any information. To improve the performance of interpolated local KNN in image classification, a good data representation should be considered, in which better performance results of better data representation. Therefore, it would be interesting to study the role of extracting abstract features from image data on the performance of the interpolated local KNN.

Bibliography

[1] Zhao, L., Mammadov, M., & Yearwood, J. (2010, December). From convex to nonconvex: a loss function analysis for binary classification. In 2010 IEEE International Conference on Data Mining Workshops (pp. 1281-1288).

[2] Cohen, A. (2014). Surrogate Loss Minimization (Doctoral dissertation, Hebrew University of Jerusalem).

[3] Grabocka, J., Scholz, R., & Schmidt-Thieme, L. (2019). Learning surrogate losses. arXiv preprint arXiv:1905.10108.

[4] Xia, W. (2019). Solution Techniques For Non-convex Optimization Problems.

[5] Mokhtari, A., Ozdaglar, A., & Jadbabaie, A. (2019, April). Efficient nonconvex empirical risk minimization via adaptive sample size methods. In The 22nd International Conference on Artificial Intelligence and Statistics (pp. 2485-2494). PMLR.

[6] Daniely, A. (2016, June). Complexity theoretic limitations on learning halfspaces. In Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (pp. 105-117).

[7] Guruswami, V., & Raghavendra, P. (2009). Hardness of learning halfspaces with noise. SIAM Journal on Computing, 39(2), 742-765.

[8] Fuentetaja, R., Borrajo, D., & López, C. L. (2009, November). A look-ahead B&B search for cost-based planning. In Conference of the Spanish Association for Artificial Intelligence (pp. 201-211). Springer, Berlin, Heidelberg.

[9] Fellows, M. R., Fomin, F. V., Lokshtanov, D., Rosamond, F., Saurabh, S., & Villanger, Y. (2012). Local search: Is brute-force avoidable?. Journal of Computer and System Sciences, 78(3), 707-719.

[10] Lin, Y. (2004). A note on margin-based loss functions in classification. Statistics & probability letters, 68(1), 73-82.

[11] Bubeck, S. (2014). Convex optimization: Algorithms and complexity. arXiv preprint arXiv:1405.4980.

[12] Nie, F., Hu, Z., & Li, X. (2018). An investigation for loss functions widely used in machine learning. Communications in Information and Systems, 18(1), 37-52.

[13] Sra, S., Nowozin, S., & Wright, S. J. (Eds.). (2012). Optimization for machine learning. Mit Press.

[14] Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., & Kolter, Z. (2019). Differentiable convex optimization layers. arXiv preprint arXiv:1910.12430.

[15] Feng, L., Shu, S., Lin, Z., Lv, F., Li, L., & An, B. (2020, January). Can cross entropy loss be robust to label noise?. In IJCAI (pp. 2206-2212).

[16] Wu, Y., & Liu, Y. (2007). Robust truncated hinge loss support vector machines. Journal of the American Statistical Association, 102(479), 974-983.

[17] Singla, M., Ghosh, D., Shukla, K. K., & Pedrycz, W. (2020). Robust twin support vector regression based on rescaled hinge loss. Pattern Recognition, 105, 107395.

[18] Xu, L., Crammer, K., & Schuurmans, D. (2006, July). Robust support vector machine training via convex outlier ablation. In AAAI (Vol. 6, pp. 536-542).

[19] Kanamori, T., Fujiwara, S., & Takeda, A. (2017). Robustness of learning algorithms using hinge loss with outlier indicators. Neural Networks, 94, 173-191.

[20] Estabrooks, A., Jo, T., & Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. Computational intelligence, 20(1), 18-36.

[21] Tang, Y., Li, X., Xu, Y., Liu, S., & Ouyang, S. (2014, October). A mixed integer programming approach to maximum margin 0–1 loss classification. In 2014 International Radar Conference (pp. 1-6).

[22] Nguyen, T., & Sanner, S. (2013, May). Algorithms for direct 0–1 loss optimization in binary classification. In International Conference on Machine Learning (pp. 1085-1093). PMLR.

[23] Xie, M., Xue, Y., & Roshan, U. (2019, December). Stochastic coordinate descent for 01 loss and its sensitivity to adversarial attacks. In 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA) (pp. 299-304).

[24] Li, L., & Lin, H. T. (2007, August). Optimizing 0/1 loss for perceptrons by random coordinate descent. In 2007 International Joint Conference on Neural Networks (pp. 749-754).

[25] Chamon, L. F., Paternain, S., Calvo-Fullana, M., & Ribeiro, A. (2021). Constrained Learning with Non-Convex Losses. arXiv preprint arXiv:2103.05134.

[26] Little, M. A. (2019). Machine Learning for Signal Processing: Data Science, Algorithms, and Computational Statistics. Oxford University Press.

[27] Kochenderfer, M. J., & Wheeler, T. A. (2019). Algorithms for optimization. Mit

[28] Vapnik, V. (1992). Principles of risk minimization for learning theory. In Advances in neural information processing systems (pp. 831-838).

[29] Von Luxburg, U., & Schölkopf, B. (2011). Statistical learning theory: Models, concepts, and results. In Handbook of the History of Logic (Vol. 10, pp. 651-706). North-Holland.

[30] Shorack, G. R., & Wellner, J. A. (2009). Empirical processes with applications to statistics. Society for Industrial and Applied Mathematics.

[31] Feldman, V., Guruswami, V., Raghavendra, P., & Wu, Y. (2012). Agnostic learning of monomials by halfspaces is hard. SIAM Journal on Computing, 41(6), 1558-1590.

[32] Gerrish, S., & Scott, K. (2018). 14 Brute-force search your way to a good strategy.

[33] Fellows, M. R., Fomin, F. V., Lokshtanov, D., Rosamond, F., Saurabh, S., & Villanger, Y. (2012). Local search: Is brute-force avoidable?. Journal of Computer and System Sciences, 78(3), 707-719.

[34] Breyer, T., & Korf, R. (2008). Recent results in analyzing the performance of heuristic search. In Proceedings of the First International Workshop on Search in Artificial Intelligence and Robotics (held in conjunction with AAAI) (pp. 24-31).

[35] Land, A. H., & Doig, A. G. (2010). An automatic method for solving discrete programming problems. In 50 Years of Integer Programming 1958-2008 (pp. 105-132). Springer, Berlin, Heidelberg.

[36] Liu, S., Wang, M., Kong, N., & Hu, X. (2021). An enhanced branch-and-bound algorithm for bilevel integer linear programming. European Journal of Operational Research, 291(2), 661-679.

[37] Kianfar, K. (2010). Branch-and-Bound Algorithms. Wiley Encyclopedia of Operations Research and Management Science.

[38] Vielma, J. P. (2015). Mixed integer linear programming formulation techniques. Siam Review, 57(1), 3-57.

[39] Rios, L. M., & Sahinidis, N. V. (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. Journal of Global Optimization, 56(3), 1247-1293.

[40] Li, L. (2005). Perceptron learning with random coordinate descent.

[41] Gallant, S. I. (1990). Perceptron-based learning algorithms. IEEE Transactions on neural networks, 1(2), 179-191.

[42] Freund, Y., & Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. Machine learning, 37(3), 277-296.

[43] Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Cornell Aeronautical Lab Inc Buffalo NY.

[44] Kalyanakrishnan, S. (2017). The Perceptron Learning Algorithm and its Convergence.

[45] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.

[46] Gupta, S. D. (2018, October). On convergence of heuristics based on Douglas-Rachford splitting and ADMM to minimize convex functions over nonconvex sets. In 2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton) (pp. 56-63).

[47] Boyd, S., Xiao, L., & Mutapcic, A. (2003). Subgradient methods. lecture notes of EE392o, Stanford University, Autumn Quarter, 2004, 2004-2005.

[48] Minsky, M., & Papert, S. A. (2017). Perceptrons: An introduction to computational geometry. MIT press.

[49] Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory (pp. 144-152).

[50] Fan, R. E., Chang, K. W., Hsieh, C. J., Wang, X. R., & Lin, C. J. (2008). LIBLINEAR: A library for large linear classification. the Journal of machine Learning research, 9, 1871-1874.

[51] Shalev-Shwartz, S., Singer, Y., Srebro, N., & Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for svm. Mathematical programming, 127(1), 3-30.

[52] Cramer, J. S. (2002). The origins of logistic regression.

[53] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[54] Zhou, Y., Wang, X., Zhang, M., Zhu, J., Zheng, R., & Wu, Q. (2019). MPCE: a maximum probability based cross entropy loss function for neural network classification. Access, 7, 146331-146341.

[55] Valpola, H. (2015). From neural PCA to deep unsupervised learning. In Advances in independent component analysis and learning machines (pp. 143-171). Academic Press.

[56] Thiessen, E. D. (2017). What's statistical about learning? Insights from modelling statistical learning as a set of memory processes. Philosophical Transactions of the Royal Society B: Biological Sciences, 372(1711).

[57] Selman, B., & Gomes, C. P. (2006). Hill-climbing search. Encyclopedia of cognitive science, 81, 82.

[58] LaSalle, D., & Karypis, G. (2016, August). A parallel hill-climbing refinement algorithm for graph partitioning. In 2016 45th International Conference on Parallel Processing (ICPP) (pp. 236-241).

[59] McGoff, K., & Nobel, A. B. (2020). Empirical risk minimization and complexity of dynamical models. The Annals of Statistics, 48(4), 2031-2054.

[60] Ying, X. (2019, February). An overview of overfitting and its solutions. In Journal of Physics: Conference Series (Vol. 1168, No. 2, p. 022022).

[61] Van der Aalst, W. M., Rubin, V., Verbeek, H. M. W., van Dongen, B. F., Kindler, E., & Günther, C. W. (2010). Process mining: a two-step approach to balance between underfitting and overfitting. Software & Systems Modeling, 9(1), 87-111.

[62] Wu, X. X., & Liu, J. G. (2009, October). A new early stopping algorithm for improving neural network generalization. In 2009 Second International Conference on Intelligent Computation Technology and Automation (Vol. 1, pp. 15-18).

[63] Shamrat, F. J. M., Chakraborty, S., Billah, M. M., Das, P., Muna, J. N., & Ranjan, R. (2021, June). A comprehensive study on pre-pruning and post-pruning methods of decision tree classification algorithm. In 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 1339-1345).

[64] Peng, Y., Lu, Y. T., & Chen, Z. G. (2021, March). An Improved Error-Based Pruning Algorithm of Decision Trees on Large Data Sets. In 2021 IEEE 6th International Conference on Big Data Analytics (ICBDA) (pp. 33-37).

[65] Yoshida, H., Kawano, S., & Ninomiya, Y. (2021). Discriminant Analysis via Smoothly Varying Regularization. In Intelligent Decision Technologies (pp. 441-455). Springer, Singapore.

[66] Xu, X., & Zhu, D. (2021). New method for solving Ivanov regularization-based support vector machine learning. Computers & Operations Research, 105504.

[67] Xia, Y., Zhao, J., He, L., Li, Y., & Niu, M. (2020). A novel tree-based dynamic heterogeneous ensemble method for credit scoring. Expert Systems with Applications, 159, 113615.

[68] Ganaie, M. A., & Hu, M. (2021). Ensemble deep learning: A review. arXiv preprint arXiv:2104.02395.

[69] Li, Z., Liu, L., Dong, C., & Shang, J. (2020). Overfitting or Underfitting? Understand Robustness Drop in Adversarial Training. arXiv preprint arXiv:2010.08034.

[70] Sehra, S., Flores, D., & Montanez, G. D. (2021). Undecidability of Underfitting in Learning Algorithms. arXiv preprint arXiv:2102.02850.

[71] Keerthi, S. S., & Lin, C. J. (2003). Asymptotic behaviors of support vector machines with Gaussian kernel. Neural computation, 15(7), 1667-1689.

[72] Jabbar, H., & Khan, R. Z. (2015). Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). Computer Science, Communication and Instrumentation Devices, 163-172.

[73] Zhou, Z. H. (2021). Why over-parameterization of deep neural networks does not overfit?. Science China Information Sciences, 64(1), 1-3.

[74] Belkin, M., Hsu, D., & Mitra, P. (2018). Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. arXiv preprint arXiv:1806.05161.

[75] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. Communications of the ACM, 64(3), 107-115.

[76] Breiman, L. (1999). Prediction games and arcing algorithms. Neural computation, 11(7), 1493-1517.

[77] Gao, W., & Zhou, Z. H. (2013). On the doubt about margin explanation of boosting. Artificial Intelligence, 203, 1-18.

[78] Mathiasen, A., Larsen, K. G., & Grønlund, A. (2019, May). Optimal minimal margin maximization with boosting. In International Conference on Machine Learning (pp. 4392-4401). PMLR.

[79] Rätsch, G., & Warmuth, M. K. (2002, July). Maximizing the margin with boosting. In International Conference on Computational Learning Theory (pp. 334-350). Springer, Berlin, Heidelberg.

[80] Bartlett, P., Freund, Y., Lee, W. S., & Schapire, R. E. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. The annals of statistics, 26(5), 1651-1686.

[81] Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics, 28(2), 337-407.

[82] Wang, L., Sugiyama, M., Yang, C., Zhou, Z. H., & Feng, J. (2008, July). On the margin explanation of boosting algorithms. In COLT (pp. 479-490).

[83] Reyzin, L., & Schapire, R. E. (2006, June). How boosting the margin can also boost classifier complexity. In Proceedings of the 23rd international conference on Machine learning (pp. 753-760).

[84] Zou, D., & Gu, Q. (2019). An improved analysis of training over-parameterized deep neural networks. arXiv preprint arXiv:1906.04688.

[85] Ma, S., Bassily, R., & Belkin, M. (2018, July). The power of interpolation: Understanding the effectiveness of SGD in modern over-parametrized learning. In International Conference on Machine Learning (pp. 3325-3334). PMLR.

[86] Chen, Z., Cao, Y., Zou, D., & Gu, Q. (2019). How much over-parameterization is sufficient to learn deep relu networks?. arXiv preprint arXiv:1911.12360.

[87] Naitzat, G., Zhitnikov, A., & Lim, L. H. (2020). Topology of Deep Neural Networks. J. Mach. Learn. Res., 21(184), 1-40.

[88] Bubeck, S., & Sellke, M. (2021). A universal law of robustness via isoperimetry. Advances in Neural Information Processing Systems, 34, 28811-28822.

[89] Belkin, M., Ma, S., & Mandal, S. (2018, July). To understand deep learning we need to understand kernel learning. In International Conference on Machine Learning (pp. 541-549). PMLR.

[90] Wyner, A. J., Olson, M., Bleich, J., & Mease, D. (2017). Explaining the success of adaboost and random forests as interpolating classifiers. The Journal of Machine Learning Research, 18(1), 1558-1590.

[91] Olson, M. (2018). Essays on random forest ensembles (Doctoral dissertation, University of Pennsylvania).

[92] Shao, Y. H., Chen, W. J., Zhang, J. J., Wang, Z., & Deng, N. Y. (2014). An efficient weighted Lagrangian twin support vector machine for imbalanced data classification. Pattern Recognition, 47(9), 3158-3167.

[93] Sun, A., Lim, E. P., & Liu, Y. (2009). On strategies for imbalanced text classification using SVM: A comparative study. Decision Support Systems, 48(1), 191-201

[94] Violina, S. (2021). Analysis of Brute Force and Branch & Bound Algorithms to solve the Traveling Salesperson Problem (TSP). Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(8), 1226-1229.

[95] Belkin, M., Rakhlin, A., & Tsybakov, A. B. (2019, April). Does data interpolation contradict statistical optimality?. In The 22nd International Conference on Artificial Intelligence and Statistics (pp. 1611-1619). PMLR.

[96] Aggarwal, C. C. (2018). Neural networks and deep learning. Springer, 10, 978-3.

[97] Xing, Y., Song, Q., & Cheng, G. (2022). Benefit of interpolation in nearest neighbor algorithms. arXiv preprint arXiv:2202.11817.

[98] Mitra, P. P. (2021). Fitting elephants in modern machine learning by statistically consistent interpolation. Nature Machine Intelligence, 3(5), 378-386.

[99] Cover, T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. IEEE transactions on electronic computers, (3), 326-334.

[100] Bartlett, P. L., Jordan, M. I., & McAuliffe, J. D. (2006). Convexity, classification, and risk bounds. Journal of the American Statistical Association, 101(473), 138-156.

[101] Zhang, T. (2004). Statistical behavior and consistency of classification methods based on convex risk minimization. The Annals of Statistics, 32(1), 56-85.

[102] Agarwal, A., & Agarwal, S. (2015, June). On consistent surrogate risk minimization and property elicitation. In Conference on Learning Theory (pp. 4-22). PMLR.

[103] Zhang, J., Liu, T., & Tao, D. (2021). On the rates of convergence from surrogate risk minimizers to the Bayes optimal classifier. IEEE Transactions on Neural Networks and Learning Systems.

[104] Jain, P., & Kar, P. (2017). Non-convex optimization for machine learning. arXiv preprint arXiv:1712.07897.

[105] Danilova, M., Dvurechensky, P., Gasnikov, A., Gorbunov, E., Guminov, S., Kamzolov, D., & Shibaev, I. (2020). Recent theoretical advances in non-convex optimization. arXiv preprint arXiv:2012.06188.

[106] Allen-Zhu, Z. (2017). Natasha 2: Faster non-convex optimization than sgd. arXiv preprint arXiv:1708.08694.

[107] Diamond, S., Takapoui, R., & Boyd, S. (2016). A general system for heuristic solution of convex problems over nonconvex sets. arXiv preprint arXiv:1601.07277.

[108] Freund, R. M. (2004). Issues in non-convex optimization. Manuscript, Massachusetts Institute of Technology. With assistance from BW Anthony.

[109] Machine learning repository. https://archive.ics.uci.edu/ml/index.php. Accessed: 2019-10-15.

[110] Nguyen, T., & Sanner, S. (2013, May). Source code of algorithms for direct 0–1 loss optimization in binary classification. http://users.cecs.anu.edu.au/~ssanner/publications.html.

[111] Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2018). Reconciling modern machine learning practice and the bias-variance trade-off. arXiv preprint arXiv:1812.11118.

[112] Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. Neural computation, 4(1), 1-58.

[113] Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., & Sutskever, I. (2021). Deep double descent: Where bigger models and more data hurt. Journal of Statistical Mechanics: Theory and Experiment, 2021(12), 124003.

[114] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.

[115] Györfi, L., Kohler, M., Krzyzak, A., & Walk, H. (2002). A distribution-free theory of nonparametric regression (Vol. 1). New York: Springer.

[116] Liang, T., & Recht, B. (2021). Interpolating classifiers make few mistakes. arXiv preprint arXiv:2101.11815.

[117] Rastin, N., Jahromi, M. Z., & Taheri, M. (2021). A generalized weighted distance k-nearest neighbor for multi-label problems. Pattern Recognition, 114, 107526.

[118] Mitchell, H. B., & Mashkit, N. (1992). Noise smoothing by a fast k-nearest neighbour algorithm. Signal Processing: Image Communication, 4(3), 227-232.

[119] Kools, J. (2021). 6 functions for generating artificial datasets. MATLAB Central File Exchange.

[120] Gao, Y., Pan, J., Ji, G., & Yang, Z. (2012). A novel two-level nearest neighbor classification algorithm using an adaptive distance metric. Knowledge-Based Systems, 26, 103-110.

[121] Ferrer-Troyano, F. J., Aguilar-Ruiz, J. S., & Riquelme, J. C. (2001, December). Non-parametric nearest neighbor with local adaptation. In Portuguese Conference on Artificial Intelligence (pp. 22-29). Springer, Berlin, Heidelberg.

[122] Wettschereck, D., & Dietterich, T. (1993). Locally adaptive nearest neighbor algorithms. Advances in Neural Information Processing Systems, 6.

[123] García-Pedrajas, N., Del Castillo, J. A. R., & Cerruela-García, G. (2015). A proposal for local K values for K-nearest neighbor rule. IEEE transactions on neural networks and learning systems, 28(2), 470-475.

[124] MaSophie, Y. (2020, Augest). Support-Vector-Machines-and-Spam-Classifier. Github. https://github.com/YunfeiMaSophie/Support-Vector-Machines-and-Spam-Classifier

[125] Dasgupta, A., & Quan, C. Y. (2020). ME5405: Machine Vision.

[126] Bai, S. (2017). Growing random forest on deep convolutional neural networks for scene categorization. Expert systems with applications, 71, 279-287.

[127] Millard, K., & Richardson, M. (2015). On the importance of training data sample selection in random forest image classification: A case study in peatland ecosystem mapping. Remote sensing, 7(7), 8489-8515.

[128] Man, W., Ji, Y., & Zhang, Z. (2018, April). Image classification based on improved random forest algorithm. In 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA) (pp. 346-350).

[129] Jogin, M., Madhulika, M. S., Divya, G. D., Meghana, R. K., & Apoorva, S. (2018, May). Feature extraction using convolution neural networks (CNN) and deep learning. In 2018 3rd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT) (pp. 2319-2323).

[130] Sani, S., Wiratunga, N., & Massie, S. (2017, June). Learning deep features for kNN-based human activity recognition. CEUR Workshop Proceedings.

[131] Knutsen, K., Agnalt, E., Radiya, K., Oyvind, K., Kampffmeyer, M. C., & Jenssen, R. (2022). A clinically motivated self-supervised approach for content-based image retrieval of CT liver images. arXiv e-prints, arXiv-2207.

[132] Boiman, O., Shechtman, E., & Irani, M. (2008, June). In defense of nearest-neighbor based image classification. In 2008 IEEE conference on computer vision and pattern recognition (pp. 1-8).

[133] LeCun, Y. (1998). The MNIST database of handwritten digits. http://yann. lecun. com/exdb/mnist/.

[134] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[135] Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747.

[136] Xie, X. (2018). A k-nearest neighbor technique for brain tumor segmentation using minkowski distance. Journal of Medical Imaging and Health Informatics, 8(2), 180-185.