Boundary Uncertainty-based Classifier Evaluation

David Ha

ID 4G163101

May 2019

Graduate School of Science and Engineering
Department of Information and Computer Science
Doshisha University
Kyoto, JAPAN

Thesis Committee:
Shigeru Katagiri, Chair
Tsuneo Kato
Xugang Lu

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Acknowledgments

Thanks to Professor Katagiri who gave me this research opportunity, his supervision, and the necessary environment.

Thanks to all the professors who lent me their advice so far, just to name a few: Professor Watanabe, Professor Ohsaki, Professor Wu, Professor Kato, Professor Lu.

Thanks to Doshisha University who hosted me in its inspiring environment, and to Ecole Centrale de Lille who gave me the chance to live this experience. Thanks to the laboratory mates who supported me, in particular to the ones who actively worked with me in this research. A Emilie: le Prudent et le Temeraire. A Charles et Juliette: merci aux tout premiers. A Nicolas: j'ai apprecie les discussions.

Thanks to every colorful encounter who accepted me on their track, and helped me run through Springs and Falls, rifts and meadows. Thanks to the precious guide of the sunny hill.

Thanks to my family, who has always guided me and welcomed me.

May 31, 2019

Abstract

We propose a general method that makes the accurate evaluation of any classifier model a realistic task, both in a theoretical sense despite the finiteness of the available data, and in a practical sense in terms of computation and memory costs.

Classification, namely predicting a class label on a data, is a ubiquitous task in our daily lives. Common applications include classification of pictures, prediction of whether a patient is positive, and assignment of a topic label to a document. The tedious task of manually assigning a label to each data can be performed by computers if they can learn the data patterns that distinguish one class from another (training step), and then use the learned patterns to classify unseen data (classification step).

Given a classification task, the ultimate goal of pattern classification is to classify unseen (infinite) data with the lowest possible error probability. The classifier evaluation challenge arises from the bias of the classification error estimate that is only based on finite data. The difficulty of accurately estimating the error probability may be understood as the need to integrate the classification error all over the data space, only based on the binary information of correctness at each data point.

Several approaches such as Structural Risk Minimization, Cross Validation, and Bootstrap aim at overcoming this fundamental difficulty. However, all this approaches come at a quite high cost, such as a quite loose estimation, or the difficulty to apply the approach to a wide range of classifiers, or the sacrifice of some data to separately estimate the error probability, or repetitions of the classifier training that can be prohibitively expensive in real-life tasks, or the impossibility to directly

optimize the estimate of the error probability in a training procedure.

We propose to bypass the existing difficulties by defining a new classifier evaluation measure called "boundary uncertainty" whose finite estimate can be considered a reliable representative of its expectation. This characteristic of the boundary uncertainty estimate makes classifier evaluation a realistic task even using finite data, furthermore at a low cost in contrast to existing approaches. Similarly to the error probability, lower values of the (sign reversed) boundary uncertainty correspond to more optimal classifier statuses.

The possibility to easily and accurately estimate the boundary uncertainty comes from a sharp focus of the boundary uncertainty on the classification decision boundary, which contains the essential information of the classification process. As a result, the difficulty of the estimation task is dramatically reduced. The optimal decision boundary is uniquely identified by equality of the adjacent class posterior probabilities along the boundary. Based on this necessary and sufficient condition for optimality, our boundary uncertainty measures the similarity between the decision boundary with the optimal decision boundary in order to evaluate classifier optimality.

In this thesis, the possibility to evaluate the classifier optimality based on the estimation of our boundary uncertainty is demonstrated on three widely used classifier models, and on more than thirteen real-life datasets. We illustrate these results with a theoretical analysis, and propose guidelines to further improve our classifier evaluation method.

Contents

1 Introduction **1**

1.1 Traditional error probability-based classifier evaluation 1

1.2 Boundary uncertainty-based classifier evaluation 4

2 A First Boundary Uncertainty-Based Classifier Evaluation Method **7**

2.1 Introduction 7

2.2 Notations and goal 8

2.2.1 The classification problem 8

2.2.2 Goal 9

2.3 Proposed method for boundary uncertainty-based classifier evaluation 10

2.3.1 Outline 10

2.3.2 Step 1: selection of the near-boundary set $\mathcal{N}_B(\Lambda)$ 12

2.3.3 Step 2: computation of uncertainty measure $U(\Lambda)$ 13

2.3.4 Adaptation of Step 1 to multi-class datasets 17

2.3.5 Adaptation of Step 2 to multi-class datasets 18

2.4 Experimental evaluation 20

2.4.1 Datasets 20

2.4.2 Classifier 21

2.4.3 Hyperparameters 21

2.4.4 Evaluation estimation 22

2.4.5 Results of Step 1 24

2.4.6 Results on the classifier selection 25

2.4.7 Influence of data imbalance on classifier status selection 29

2.5 Summary 30

3 Optimality analysis of uncertainty measure **33**

3.1 Overview and preparations 33

3.2 Convergence to true value for ratio-based probability density estimator 34

3.3 Convergence to true value for ratio-based joint probability density estimator . . . 35

3.4 Probabilistic convergence to true value for ratio-based posterior probability estimator 37

3.5 Practical advantages supported by optimality in ratio-based posterior probability estimation 38

3.6 Experiments 39

3.7 Summary 42

4 An Improved Boundary Uncertainty-Based Classifier Evaluation Method **45**

4.1 Background for our improved boundary uncertainty estimation 45

4.1.1 Reminder: Goal 46

4.1.2 Towards Proposal 1 47

4.1.3 Reminder: Outline of Proposal 1 48

4.1.4 From Proposal 1 to Proposal 2 50

4.2 Outline of Proposal 2 52

4.3 Continuous measure of near-boundary-ness . . . 53
4.4 Improved Step 1 . . . 54
4.4.1 Class-by-class determination of the number of anchors to generate . . . 54
4.4.2 Class-by-class generation of anchors . . . 56
4.5 Improved Step 2 . . . 57
4.6 Benefits in terms of memory and time . . . 60
4.6.1 Experiments . . . 61
4.6.2 Classifiers . . . 61
4.6.3 Datasets . . . 62
4.6.4 Hyperparameters . . . 63
4.6.5 Treatment of the class imbalance in datasets . . . 64
4.6.6 Effect of the dimensionality on Proposal 1 . . . 65
4.6.7 Classifier evaluation results . . . 66
4.6.8 About the equivalence between maximum boundary uncertainty and classifier optimality . . . 70
4.7 Summary . . . 72
4.7.1 Sample-dependent Parzen estimator . . . 73
4.7.2 Cross Validation Maximum Likelihood (CVML) estimation of h . . . 74

5 Conclusion **79**

5.1 Summary of Dissertation . . . 79
5.2 Future Works . . . 80

A Notation List **85**

A.1 Notations 85

B Publication List **87**

List of Figures

2.1 Graphical explanation of our two-step method on two-dimensional data. (a): Decision boundary $B(\Lambda)$ represented in blue. (b): Step 1 arbitrarily generates anchors (red dots) on $B(\Lambda)$. (c): Step 1 selects the nearest neighbor for each anchor. This results in a set of near-boundary samples denoted $\mathcal{N}_B(\Lambda)$. (d): From now, the method only considers $\mathcal{N}_B(\Lambda)$ to focus on the decision boundary. (e): Step 2 uses a partitioning method to break down $\mathcal{N}_B(\Lambda)$ into clusters, represented in green dashed ovals. (f): After estimating the class posterior probabilities in each cluster by application of the kNN class posterior probability estimation rule (shortened to "kNN estimation" for convenience), Step 2 computes the boundary uncertainty $U(\Lambda)$. 11

2.2 Anchor generation based on a random pair $[a, b]$ belonging to a pair of different estimated classes $\widehat{C}_1$ and $\widehat{C}_2$, where we illustrate the need for a case-by-case treatment. In the two-class data case, the segment $[a, b]$ strides only two class regions, and only the boundary $B_{12}(\Lambda)$ can pass through the segment (case (A)). In the multi-class data case, the segment $[a, b]$ can stride multiple class regions, and multiple boundaries can pass through the segment (case (B)); when the segment strides multiple class regions, we divide the segment in the class-by-class manner and generate an anchor for every pair of two adjacent classes. 17

2.3 SVM evaluation results on the GMM dataset. In the upper graph, the horizontal axis corresponds to the kernel width γ, the left vertical axis corresponds to L_{te} (red) and L_{val} (green), and the right vertical axis corresponds to $-U$ (blue). For each of the four classifier statuses A, B, C, D, we represent the following three plots in the corresponding row. From left to right: data with true class labels; data with labels estimated by the classifier, and anchors in black dots; zoom on the set of one nearest neighbors of the anchors, represented with their true class label. These nearest neighbors are assumed representative of the decision boundary. Local balance between purple and red labels all along the decision boundary corresponds to more optimal classifier statuses. 24

2.4 Classifier evaluation results for synthetic GMM dataset and 11 real-life datasets. From top to bottom: GMM, Abalone, Breast Cancer, Cardiotocography, Ionosphere, and Landsat Satellite. Horizontal axis indicates the value of γ. Each panel shows the estimated error probability curve based on the CV's training folds (L_{tr}, *yellow*), the estimated error probability curve based on the CV's validation folds (L_{val}, *green*), and our uncertainty measure curve ($-U$, blue). In the top GMM panel, we also display the error probability estimate based on the extra 20,000 validation samples (L_{val2}, red). 28

2.5 Parameter status selection results for Cardiotocography data. In the top panel, the blue curve represents uncertainty measure $-U$ with the prior probability correction; the red dashed curve for $-U$ without the correction. In the bottom panel, the black curve represents the number of near-boundary samples; the gray and red curves represent the numbers of near-boundary samples belonging to C_0 and to C_1, respectively. 30

3.1 Comparison of three neighbor selection schemes in Step 2 (near-boundary sample selection) for Ionosphere dataset: fixed kNN considering neighbors selected from $\mathcal{T}$ (top panel), fixed kNN considering neighbors selected from $\mathcal{N}_B(\Lambda)$ (middle panel), and adaptive partitioning in $\mathcal{N}_B(\Lambda)$ (bottom panel). 40

3.2 Effect of using different k in applying kNN to $\mathcal{T}$ for the Ionosphere dataset. L_{val} is shown by green dashed line. Curves for $k = 5, 7, 10, 15$ are shown in red, green, orange, and blue, respectively. 41

3.3 Effect of using different k in applying kNN to $\mathcal{N}_B(\Lambda)$ for the Ionosphere dataset. L_{val} is shown by green dashed line. Curves for $k = 5, 7, 10, 15$ are shown in red, green, orange, and blue, respectively. 41

3.4 Geometric distance distribution for the samples used in the posterior probability estimation for Breast Cancer data (left) and Ionosphere data (right) ($\gamma = 2^{-5}$). The green histogram shows distance distribution for the near-bounday samples; the red histogram for the neighbors selected from the entire sample set. 42

4.1 kNN estimation at the decision boundary performed in different ways, from (i) to (iv). (i): ideal estimation: at $\boldsymbol{a} \in B(\Lambda)$ (red dot), possibility to draw class labels an infinite number of times. (ii): basic kNN estimation at $\boldsymbol{a}$ in presence of finite data: use of the class labels surrounding $\boldsymbol{a}$ (green circle), regardless of their closeness to $B(\Lambda)$. (iii.1) to (iii.3): kNN estimation focused on $B(\Lambda)$ as done in Proposal 1. First, selection of the samples nearest to $B(\Lambda)$ (red crosses selected by on-boundary red dots). Second, clustering on these selected samples to obtain target volumes focused on $B(\Lambda)$ (green circles in (iii.2) and (iii.3)). (iv): kNN estimation centered on $B(\Lambda)$ as done in Proposal 2, detailed in the later sections. . 47

4.2 Graphical explanation of our original two-step classifier evaluation method on two-dimensional data. (a): Decision boundary $B(\Lambda)$ represented in blue. (b): Step 1 arbitrarily generates anchors (red dots) on $B(\Lambda)$. (c): Step 1 selects the nearest neighbor for each anchor. This results in a set of near-boundary samples denoted $\mathcal{N}_B(\Lambda)$. (d): From now, the method only considers $\mathcal{N}_B(\Lambda)$ to focus on the decision boundary. (e): Step 2 uses a partitioning method to break down $\mathcal{N}_B(\Lambda)$ into clusters, represented in green dashed ovals. (f): After estimating the class posterior probabilities in each cluster by application of the kNearest Neighbor class posterior probability estimation rule, Step 2 computes the boundary uncertainty $U(\Lambda)$. Step 2 is repeated R times, and the final estimate of $U(\Lambda)$ is the average of the results over the R runs. 49

4.3 Determination of N_a^0 on synthetic data for some trained SVM classifier. The originally multi-dimension vector samples of C_0 are solely represented by their measure of near-boundary-ness (horizontal axis). To estimate N_a^0, we read the value at zero of the histogram on this one-dimensional data. In this example, the histogram indicates that $N_a^0 = 175$ anchors should be generated for C_0 to respect $P(C_0)$. 55

4.4 Illustration of anchor generation in Proposal 2 for two-class data. (a): Estimated labels for class 1 ($\widehat{C_1}$) and class 2 ($\widehat{C_2}$) are shown in yellow squares and green triangles, respectively. (b): for each training sample, $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ is represented in a black arrow, and corresponds to the locally most efficient direction to search for $B(\Lambda)$. (c): For each training sample $\boldsymbol{x}$, an on-boundary anchor (red dot) is tentatively generated along the direction provided by $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ within a distance d from $\boldsymbol{x}$. 56

4.5 The horizontal axis corresponds to the degree of near-boundary-ness for samples of class C_i, and the opposite of the degree of near-boundary-ness for samples of class C_j. On this axis, the anchor $\boldsymbol{a}$ (value zero) and its $M = 39$ nearest neighbors are represented. Among $NN(M, \boldsymbol{a})$, 24 neighbors belong to C_i (orange), and 15 neighbors belong to C_j (blue). We perform a smooth histogram for each class in red and black, respectively. We use these two histograms to obtain a smooth count k_i and k_j of C_i and C_j on $B(\Lambda)$. Here $k_i = 1.5$ and $k_j = 1.9$. 58

4.6 Different possible function candidates to the uncertainty measure. The binary Shannon entropy and the Gini impurity are represented in brown and pink, respectively. 64

4.7 Histograms of the absolute value of the near-boundary-ness of the training samples $\mathcal{T}$ (on-boundary-ness corresponds to the value 0). Blue: histogram for all $\mathcal{T}$. Orange: histogram for the selected near-boundary samples $\mathcal{N}_B(\Lambda)$. 65

4.8 From left to right and top to bottom, SVM classifier status selection results for Abalone_01, Breast Cancer, Cardiotocography, German, GMM, Ionosphere, Landsat Satellite_47, Spambase. Left vertical axis, green: L_{val} and red: L_{te}. Right vertical axis, blue: $-U(\Lambda)$. Horizontal axis: γ. 66

4.9 From left to right and top to bottom, PBC classifier status selection results for Abalone, Avila, Breast Cancer, Cardiotocography, German, GMM_5c, GMM, Letter Recognition, MNIST (test), Landsat Satellite_47, Landsat Satellite, Spambase, Thyroid, Wine Quality Red, Wine Quality White. Left vertical axis, green: L_{val} and red: L_{te}. Right vertical axis, blue: $-U(\Lambda)$. Horizontal axis: k. 67

4.10 SVM evaluation results on the GMM_inclusion dataset. In the upper graph, the horizontal axis corresponds to the kernel width γ, the left vertical axis corresponds to L_{te} (red) and L_{val} (green), and the right vertical axis corresponds to $-U$ (blue). For each of the four classifier statuses A, B, C, D, we represent the following three plots in the corresponding row. From left to right: data with true class labels; data with labels estimated by the classifier, and anchors in black dots; zoom on the set of one nearest neighbors of the anchors, represented with their true class label. 70

List of Tables

2.1 Datasets 20

4.1 Datasets 62

1 Introduction

1.1 Traditional error probability-based classifier evaluation

Classifier evaluation lies at the heart of the optimal classifier design. We first remind some basics of pattern classification, and then remind the existing challenges in classifier evaluation. Pattern classification is the task of correctly assigning a class label to any unseen data point. To do so, a classifier is given training data pairs (samples) that consist of a data point and of a corresponding class label assumed correct. The classifier uses these training pairs to learn a mapping between data points and class labels, and then applies the learned mapping to classify unseen data points.

However, statistical pattern classification fundamentally assumes that all classes exist at each data point with a given probability. It can be shown [1] that the optimal classification decision is the decision that assigns each data point to the class with the highest posterior probability at this data point, and that this classification decision leads to the lowest achievable classification error probability, called Bayes error or minimum error probability. This intuitive and natural result can equivalently be viewed in terms of decision regions: the optimal classifier forms decision regions inside each of which a given class has the highest posterior probability. Decision regions are delimited by the decision boundary that is locally determined by the equality between the two highest class posterior probabilities. For convenience, we call the optimal decision boundary "Bayes boundary". On the Bayes boundary, the optimal classifier cannot decide for a single class, so we call the samples on the Bayes boundary "uncertain samples".

Under this assumption, the goal of pattern classification is not to "correctly" classify a data point, but rather to "optimally" assign it to the class label that has the highest probability. However, in presence of finite data, class posterior probabilities at the training data points are essentially unknown, therefore the empirical error rate cannot be representative of the error probability. This issue is called overfitting, and usually corresponds to a serious bias downward of the empirical error rate [2].

An analytic estimation of this bias is provided by Structural Risk Minimization (SRM) in the form of an upper bound on the expected error probability. In practice the bound tends to be loose, however the trend rather than the value itself of the upper bound may be sufficient to select the classifier status that is closest to optimality [3]. Although SRM applies to any classifier, deriving the upper bound for each considered classifier model is not always easy [2, 4]. Therefore, even assuming that the trend is reliable enough, directly using the analysis provided by SRM to practically evaluate or select classifier statuses may not always be possible.

Several approaches aim at directly estimating the error probability in a data-driven way without explicitly and analytically estimating this bias. A common approach called Hold Out (HO) evaluation can be simply applied to any classifier as follows: split the available finite data into training data and validation data, then train the classifier on the training data, and finally use the empirical error rate on the validation data as an estimate of the error probability. The role of the validation data is reduce the bias due to the training data, by simulating infinite data with new data points and new data labels. However, even in this case, there is still a bias of the HO estimate due to the finiteness of the validation data, hence performing the estimation on several validation sets is preferable.

Based on this idea, in Cross Validation (CV) [5] several training-validation splits are prepared, and then the error probability estimate is taken as the average over the empirical error rate from each validation fold, which also reduces the variance of the error probability estimation. The extreme case of CV is when each sample in turn is used as a validation set. This case called Leave-One-Out leads to an unbiased estimation of the Bayes error. Perhaps similar to CV, Bootstrap produces several sample sets by applying sampling with replacement to the given set, and

then increases the estimation reliability by repeating the estimation on each resampled set, while averaging the estimates over those resampled sets.

While Bootstrap and CV reduce both bias and variance of the error probability estimate [6, 7], they require costly training repetitions that can be prohibitive in real-life tasks [8]. Furthermore, evaluation of the error probability solely based on the training data would be preferable to the sacrifice of some precious data for evaluation.

One alternative to reduce the variance of the error probability estimation is to consider a smooth error count instead of a binary error count [9]. One issue is how to appropriately smooth the error count to bridge the gap with the error probability, without the need to empirically adjust smoothing parameters with validation data. Minimum Classification Error training [10] proposes a rationale to such smoothing that can be easily applied to any classifier model. It was shown that the smoothing in [10] is equivalent to consider virtual samples that provide a direct relationship between the resulting smoothed empirical error rate and the error probability [11], however it is yet not necessarily sufficient to accurately estimate the error probability [12].

As can be seen above, a wide range of directions were explored to accurately estimate the error probability. However due to the difficulty of estimating the error probability that requires to integrate errors over the entire space only based on the rough information of "correctness", this estimation comes at a cost. So far, an accurate estimation either requires several costly training repetitions, or the accurate error probability estimate is not directly embedded in the training objective, or the method is not easily applicable to a wide range of classifiers.

Incidentally, even assuming that an accurate estimate of the error probability can be obtained, its value itself cannot directly inform whether the optimal classification decision is reached or how close it is to the current classification decision, because the Bayes error that should serve as a reference for optimality is essentially unknown [13] and its value is specific to each dataset.

Another approach is to directly estimate probability distributions for each class to obtain the class posterior probabilities. Several methods aim at such estimation, such as information criteria [14] and Bayesian model selection [15]. However, for classification, the quality of the class posterior

probability estimation only needs, and should focus on the decision boundary. Probability distribution estimation methods tend to be mainly influenced by the center of the class probability distributions where more samples are gathered, which leads in a poorer estimation quality near the classification decision where usually less samples can be found. Such methods may therefore not be the most accurate or the most efficient for classification [16, 17, 18].

1.2 Boundary uncertainty-based classifier evaluation

We remark that achieving or detecting the minimum error probability status does not necessarily require to use the classification error as a classifier evaluation measure, and focusing on accurately estimating the information close to the decision boundary is not only easier, but also a more direct estimation goal in regard to classification.

The optimal decision is achieved if and only if the decision boundary is the Bayes boundary, and we remind that the Bayes boundary is defined by the samples where the two highest class posterior probabilities are equal. We use this constraint on the decision boundary to define a new classifier evaluation measure called "boundary uncertainty". Our boundary uncertainty quantifies the similarity between the decision boundary and the Bayes boundary by measuring on the decision boundary how close to equality are the two posterior probabilities of the two adjacent classes, and then averages the local uncertainty scores into a single boundary score. In our definition, closer values of class posterior probabilities at each location on the decision boundary correspond to a higher value of the boundary uncertainty. Furthermore, under mild assumptions that we further discuss in Chapter 4, *a maximum value of our boundary uncertainty gives a necessary and sufficient condition for classifier optimality*. Besides, this maximum value is the same regardless of the dataset, which clearly informs whether the optimal classification decision was reached in contrast to an error probability-based classifier evaluation.

A class posterior probability estimation that focuses on the decision boundary is a much simpler task than traditional class posterior probability estimation in the entire data space and error

probability estimation. Indeed, only requiring to know the class posterior probabilities on the decision boundary can be intuitively understood as only having to smooth training samples onto the decision boundary. This smoothing is locally a one-dimensional task, where the only relevant dimension is the degree of closeness of data points to the decision boundary. Finding an appropriate smoothness in this single dimension is a quite simple task that can even be done analytically without the need for validation data.

Furthermore, our boundary uncertainty estimation is robust to estimation errors, which makes it quite easy to accurately estimate. To fix ideas, let's assume that in a given neighborhood of the decision boundary, the class posterior probabilities are 0.3 for one class and 0.7 for the other class. The class posterior probability ratio at a given data point could indifferently consist of 0.3 for the first class class and 0.7 for the second class, or 0.7 for the first class and 0.3 for the second class. This would still result in the same overall closeness of the two class posteriors around the equality value 0.5.

In contrast to error probability estimation or probability distribution estimation, the key novelty of our boundary uncertainty-based evaluation approach is to reduce the classifier evaluation problem to an estimation target that is easy to estimate, robust to estimation errors, and that efficiently focuses on the goal of classification. Admittedly, sharply focusing on the decision boundary has a price. Focusing on the decision boundary means that we have basically fewer samples available for the estimation; we may compensate for this with a more robust design of our boundary uncertainty estimation method. Furthermore, a focus only on the few essential samples implies that a change on the decision boundary directly impacts the value of the boundary uncertainty, which may help to discriminate between classifier statuses.

Among our several attempts [19, 20, 21] at boundary uncertainty estimation, [21] was our first proposal that showed consistently encouraging results on several datasets and classifier models. We introduce this first proposal in Chapter 2. After analyzing the strong points and weaknesses of this first proposal in Chapter 3, we propose in Chapter 4 an improved estimation of the boundary uncertainty that is significantly more reliable and more scalable.

We tested our two boundary uncertainty-based classifier evaluation methods on well-known and widely used classifiers such as Support Vector Machines (SVM), Prototype-Based Classifiers (PBC), and Multilayer Perceptrons (MLP). Experimental results and analysis on more then thirteen real-life datasets and two synthetic datasets demonstrate the possibility to perform a reliable classifier evaluation and classifier selection using our boundary uncertainty.

2 A First Boundary Uncertainty-Based Classifier Evaluation Method

2.1 Introduction

In this chapter, we propose a first basic method to estimate the boundary uncertainty that can be transparently applied to any classifier model. In this chapter we show the experimental results of our method to evaluate and select the classifier status of SVMs, but more experimental results can be found about the application to Prototype-Based Classifiers and Multilayer Perceptrons in [22].

In our comparative experiments, we compare classifier evaluation results based on our decision boundary uncertainty with classifier evaluation results based on a reliable estimate of the error probability. The simplest candidate to perform such a reliable estimate would be a Hold Out (HO) evaluation that uses a large validation set. However, a large amount of data for both classifier training and validation is not always available. We only performed such gorgeous evaluation of the error probability on synthetic datasets, because arbitrarily large amounts of data can be generated for synthetic datasets. For real-life datasets, we instead performed the reliable evaluation of the error probability using Cross Validation (CV). Not only is the reliability of CV a well-documented topic [6], but the possibility of transparently applying CV to any classifier model makes it both an appropriate and a strong competitor.

Although our proposed method is not restricted to SVMs, error probability estimation methods that are SVM-specific are still relevant candidates for comparison. Existing SVM-specific meth-

ods rely on the approximation of LOO-CV [23], or on Minimum Description Length [24], or on Information Criteria [25]. While [23] produces accurate results for small-sized datasets, it may be excessively time-consuming and unpractical for large-sized datasets. Moreover, compared to [23], the other less time-consuming methods [24, 25] basically provide less reliable classifier evaluation. Basically, these studies seemed to consider LOO as the most reliable method to estimate the error probability. Therefore, we simply focused our comparison with CV-based error probability estimation, and ensured that the number of CV folds is sufficiently high to approximate LOO.

2.2 Notations and goal

2.2.1 The classification problem

Given a pattern sample $\boldsymbol{x} \in \mathcal{X}$, where $\mathcal{X}$ is a D-dimensional pattern space, we consider a task of classifying $\boldsymbol{x}$ into one of J classes $(C_1, \ldots, C_J)$. Our proposal assumes a general form of classification decision rule:

$$\widehat{C}(\boldsymbol{x}) = C_k \text{ iff } k = \arg\max_{j \in [1,J]} g_j(\boldsymbol{x}; \Lambda), \tag{2.1}$$

where $\widehat{C}(\cdot)$ is a classification operator, Λ is a set of classifier parameters, and $g_j(\boldsymbol{x}; \Lambda)$ is a discriminant function for C_j. $g_j(\boldsymbol{x}; \Lambda)$ represents the degree of confidence of the classifier for class C_j, and is assumed continuous in regard to $\boldsymbol{x}$ and Λ. The notation $\widehat{C}(\boldsymbol{x})$ rather than $C(\boldsymbol{x})$ is meant to clearly distinguish between the original class label $C(\boldsymbol{x})$, and the class label estimated by the classifier $\widehat{C}(\boldsymbol{x})$.

At a given sample location $\boldsymbol{x}$, the highest (dominant) discriminant function score $g_j(\cdot; \Lambda)$ enables the classifier to assign $\boldsymbol{x}$ to class C_j. When the two highest scores are equal, the classifier cannot decide between the two corresponding classes, and we say that $\boldsymbol{x}$ lies on the decision boundary between the two classes.

For convenience, we denote $D_j(\boldsymbol{x})$ as the index of the j-th highest discriminant function at $\boldsymbol{x}$ ranked in descending order (for example, the highest score at $\boldsymbol{x}$ corresponds to $g_{D_1(\boldsymbol{x})}(\boldsymbol{x}; \Lambda)$). Dependency of $D_j(\boldsymbol{x})$ in Λ is dropped to simplify the notations. Furthermore, we denote $B_{iy}(\Lambda)$ as the decision boundary that separates C_y and C_i. Naturally, $B_{iy}(\Lambda)$ and $B_{yi}(\Lambda)$ are interchangeable.

The concept of boundary directly extends to three or more classes; however, equality among the three or more highest scores at a sample location is less likely. Therefore, the rest of this thesis assumes boundaries between two classes. Under this assumption, at a given sample location $\boldsymbol{x}$, we only consider the scores of class indexes $D_1(\boldsymbol{x})$ and $D_2(\boldsymbol{x})$, and multi-class classification tasks are locally treated as two-class tasks. Although this assumption is reasonable, we can quite easily adapt our procedures to consider more than two $D_j(\cdot)$, $j \in [1, J]$ if necessary.

2.2.2 Goal

Our goal is to evaluate classifier statuses based on the estimation of the boundary uncertainty, whose higher values correspond to more optimal status. Given a classifier model whose parameter status Λ leads to the decision boundary $B(\Lambda)$, given a data point $\boldsymbol{x}$ on $B(\Lambda)$, we denote $U(\boldsymbol{x}; \Lambda)$ as the uncertainty at $\boldsymbol{x}$; $U(\boldsymbol{x}; \Lambda)$ takes higher values when class posterior probabilities are closer to equality. To summarize the overall amount of uncertainty on $B(\Lambda)$, we define the boundary uncertainty $E[U(\Lambda)]$ is the expectation of uncertainties on $B(\Lambda)$:

$$E[U(\Lambda)] = \int_{\boldsymbol{x} \in \left(B(\Lambda) + \delta V\right)} U(\boldsymbol{x}; \Lambda) p(\boldsymbol{x}) d\boldsymbol{x}, \tag{2.2}$$

where $p(\boldsymbol{x})$ corresponds to the density at sample $\boldsymbol{x}$. The δV in Eq. (2.2) means that we consider a volume of infinitesimal width centered on $B(\Lambda)$ instead of just $B(\Lambda)$ itself, to avoid a probability measure trivially equal to zero. Although slightly paradoxical, this aspect of our current formalization does not impact on the conceptual potential of using boundary uncertainty to perform classifier evaluation. To lighten notations, in what follows we shorten integrals or sums over $B(\Lambda) + \delta V$ to integrals or sums over $B(\Lambda)$, and "on $B(\Lambda)$" actually strictly refers to "quasi on $B(\Lambda)$".

The integral in Eq. (2.2) assumes that we can sample an infinite amount of data from $p(\boldsymbol{x})$ on $B(\Lambda)$, which is not the case with finite training data. With a finite training set $\mathcal{T}$, we instead consider:

$$U(\Lambda) = \sum_{\boldsymbol{x} \in B(\Lambda)} U(\boldsymbol{x}; \Lambda) p(\boldsymbol{x}). \tag{2.3}$$

2.3 Proposed method for boundary uncertainty-based classifier evaluation

2.3.1 Outline

Our method takes as input a set Λ_{TR} of candidate classifier statuses to evaluate. For each input classifier status Λ, our method performs the estimation of Eq. (4.1) through a two-step treatment, and then chooses the classifier status Λ that corresponds to the highest (estimated) boundary uncertainty. The two steps correspond to:

- Step 1: selection the training samples on $B(\Lambda)$, or at least closest to $B(\Lambda)$.
- Step 2: estimation of $E[U(\Lambda)]$ using Eq. (4.1).

We summarize our boundary uncertainty-based method in Algorithm 1, and illustrate it in Figure 4.2.

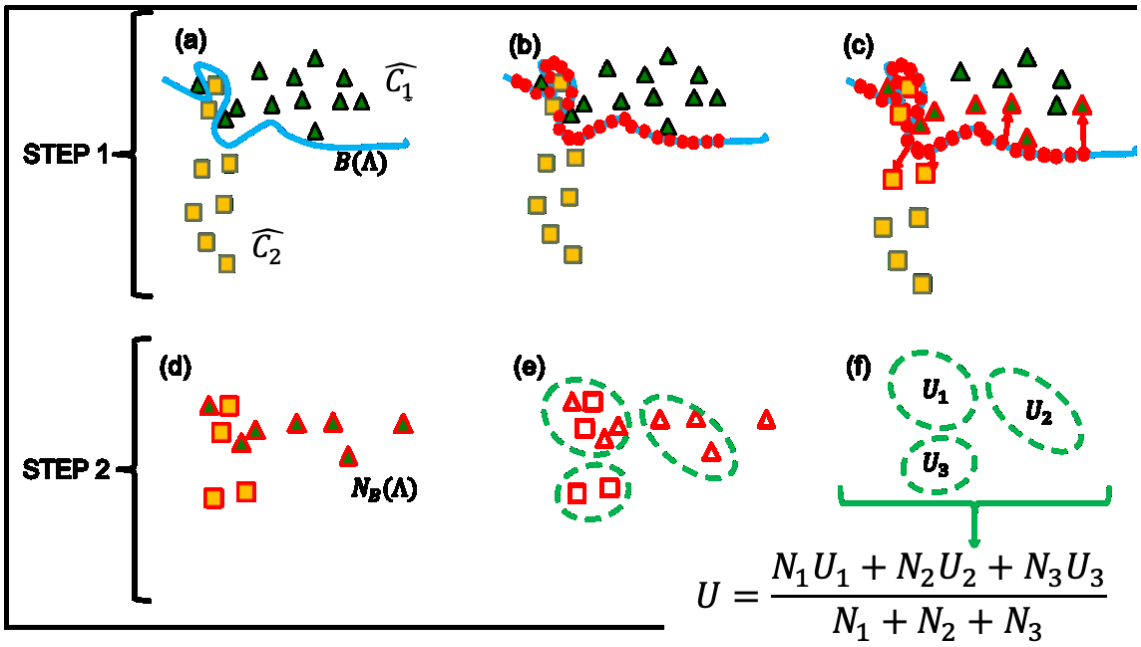

Figure 2.1: Graphical explanation of our two-step method on two-dimensional data. (a): Decision boundary $B(\Lambda)$ represented in blue. (b): Step 1 arbitrarily generates anchors (red dots) on $B(\Lambda)$. (c): Step 1 selects the nearest neighbor for each anchor. This results in a set of near-boundary samples denoted $N_B(\Lambda)$. (d): From now, the method only considers $N_B(\Lambda)$ to focus on the decision boundary. (e): Step 2 uses a partitioning method to break down $N_B(\Lambda)$ into clusters, represented in green dashed ovals. (f): After estimating the class posterior probabilities in each cluster by application of the kNN class posterior probability estimation rule (shortened to "kNN estimation" for convenience), Step 2 computes the boundary uncertainty $U(\Lambda)$.

Algorithm 1: Boundary uncertainty-based classifier selection

Input: Set Λ_{TR} of trained classifier parameters

Output: $\arg\max_{\Lambda\in\Lambda_{TR}} U(\Lambda)$

for $\Lambda \in \Lambda_{TR}$ **do**

`/* Step 1: Selection of near-boundary samples */`

Randomly generate anchors on $B(\Lambda)$ (Algorithm 2, or Algorithms 4 and 5);

Select the set $\mathcal{N}_B(\Lambda)$ that consists of the 1-nearest neighbor of each anchor;

`/* Step 2: Computation of the boundary uncertainty */`

Break $\mathcal{N}_B(\Lambda)$ into clusters (Algorithm 3);

Apply the kNN estimation to the clusters, deduce $U(\Lambda)$ (Eq. (2.7));

end

return $\arg\max_{\Lambda\in\Lambda_{TR}} U(\Lambda)$

2.3.2 Step 1: selection of the near-boundary set $\mathcal{N}_B(\Lambda)$

We first cover the two-class data case, C_1 and C_2. In this case, we simply denote $B(\Lambda)$ as the boundary $B_{12}(\Lambda)$ between estimated classes $\widehat{C}_1$ and $\widehat{C}_2$.

Although the sum Eq. (4.1) should ideally be performed on $B(\Lambda)$, in practice there are no training samples on $B(\Lambda)$, but rather at best training samples close to $B(\Lambda)$. To detect such training samples, we first generate data points exactly on $B(\Lambda)$ called "anchors", whose set we denote $A(\Lambda)$. Then, we select the nearest training sample for each anchor. We call the selected training samples "near-boundary set", and denote it $\mathcal{N}_B(\Lambda)$.

To generate anchors, we remark that anchors are defined by $f(\cdot;\Lambda) = (g_1 - g_2)(\cdot;\Lambda) = 0$, where $g_1(\cdot;\Lambda)$ and $g_2(\cdot;\Lambda)$ are fully known. Training samples assigned to $\widehat{C}_1$ by the classifier satisfy $f(\boldsymbol{x};\Lambda) > 0$, while the training samples assigned to $\widehat{C}_2$ satisfy $f(\boldsymbol{x};\Lambda) < 0$. Therefore, given $\{\boldsymbol{x}, \boldsymbol{x}'\} \in \widehat{C}_1 \times \widehat{C}_2$, the theorem of intermediate values applied to $f(\cdot;\Lambda)$ guarantees that we can find at least one anchor on $[\boldsymbol{x};\boldsymbol{x}']$, as illustrated in Case (A) of Figure 2.2). The search of each

anchor between such pair of training samples can be done efficiently by dichotomy, because it simply corresponds to the search of (necessarily existing) zeros of $f(\cdot;\Lambda)$ along segments $[\boldsymbol{x};\boldsymbol{x}']$.

One issue is how many anchors to generate, and which couples of training samples to use for the anchor generation. We remark that the training set is finite, therefore the number of training samples that are closest to $B(\Lambda)$ is finite, and a finite number of anchors should be enough to select all near-boundary samples. Based on this remark, we proceed as follows. We generate anchors in batches from randomly selected pairs from $\widehat{C}_1 \times \widehat{C}_2$, and after the generation of each anchor batch, we add to $\mathcal{N}_B(\Lambda)$ their nearest neighbors. If almost no samples are added to $\mathcal{N}_B(\Lambda)$, or if a preset maximum number of iterations i_M is reached, then we stop the process. We summarize the generation of anchors and the selection of $\mathcal{N}_B(\Lambda)$ in Algorithm 2.

Algorithm 2: selection of the near-boundary set $\mathcal{N}_B(\Lambda)$ for two-class data

Input: Classifier model parameters Λ that were trained on $\mathcal{T}$

Output: $\mathcal{N}_B(\Lambda)$

if $\forall \boldsymbol{x}, \boldsymbol{x} \in \widehat{C}_1$ *or* $\forall \boldsymbol{x}, \boldsymbol{x} \in \widehat{C}_2$ **then** $\mathcal{N}_B(\Lambda) \leftarrow \emptyset$;
else
  while *not stop* **do**
    Randomly pick $\{\boldsymbol{x}, \boldsymbol{x}'\} \in \widehat{C}_1 \times \widehat{C}_2$;
    Find $\alpha \in [0,1] : g_1(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{x}';\Lambda) = g_2(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{x}';\Lambda)$;
  end
end
return $\mathcal{N}_B(\Lambda)$

2.3.3 Step 2: computation of uncertainty measure $U(\Lambda)$

The goal of this section is to define the uncertainty measure $U(\cdot;\Lambda)$. Given a target sample $\boldsymbol{a}$, $U(\boldsymbol{a};\Lambda)$ should reach its maximum when $P(C_1|\boldsymbol{a}) = P(C_2|\boldsymbol{a})$, and take a lower value when the two class posterior probabilities are less equal. As one possible choice (e.g. Gini impurity [26]), we adopt the Shannon entropy defined in Eq (2.4). We discuss the influence of this choice in

later chapters.

$$U(\boldsymbol{a};\Lambda) = -\sum_{j=1}^{2} P(C_j|\boldsymbol{a}) \log(P(C_j|\boldsymbol{a})). \tag{2.4}$$

To use Eq. (2.4), we must estimate class posterior probabilities and choose the target sample $\boldsymbol{a}$ in Eq (2.4). The kNN class posterior probability estimation rule (simply called kNN estimation in what follows) provides a simple local method to estimating class posterior probabilities that does not need to assume a probability model on the data, although it requires to properly adapt the concept of "local" to the estimation task. The kNN class posterior estimation rule estimates $P(C_1|\boldsymbol{a})$ and $P(C_2|\boldsymbol{a})$ as follows:

I. Define a local volume $V(\boldsymbol{a})$ around $\boldsymbol{a}$. The size and shape of $V(\boldsymbol{a})$ can be chosen, however the size should not be too big. A bigger size provides more samples for the class posterior probability estimation, however the resulting estimation focuses less accurately on $\boldsymbol{a}$.

II. Denote k the number of training samples contained by $V(\boldsymbol{a})$. Count the number k_1 of training samples contained by $V(\boldsymbol{a})$ that are labeled C_1, and the number k_2 of training samples contained by $V(\boldsymbol{a})$ that are labeled C_2.

III. Application of the Bayes theorem gives $P(C_1|\boldsymbol{a}) = k_1/k$ and $P(C_2|\boldsymbol{a}) = k_2/k$.

Assuming that we could choose $\boldsymbol{a}$, we must set its surrounding volume $V(\boldsymbol{a})$ in a way that the class posterior probability estimation focuses on $B(\Lambda)$. The simplest choice would be to select the k nearest neighbors to $\boldsymbol{a}$ (denoted $NN(\boldsymbol{a}, k)$) using the Euclidean distance. This would correspond to the definition of a spherical volume $V(\boldsymbol{a})$. However, such volume would mostly contain non-boundary information, and the resulting class posterior probability estimation would not be useful to evaluate boundary uncertainty.

To define volumes that only contain on-boundary information, we propose to only use samples from $\mathcal{N}_B(\Lambda)$ as follows: first, break down $\mathcal{N}_B(\Lambda)$ into smaller volumes (clusters); second, apply the kNN estimation to each obtained cluster. With this method, in each volume, the centroid of the volume can be implicitly seen as a target sample $\boldsymbol{a}$ for the estimation, and each volume

only contains near-boundary information that is relevant for the estimation of the boundary uncertainty. Ideally, such volumes should locally adapt to the density as specified in Eq. (4.1) by the coefficient $p(\boldsymbol{x})$.

Clustering methods provide a natural way to form clusters whose size adapt to the density. We therefore use the hierarchical clustering method summarized in Algorithm 3, that consists of R iterations over which we average estimation results to increase the reliability. Each r-th iteration of the outer loop ($r \in [1, R]$) of Algorithm 3 forms smaller clusters out of $\mathcal{N}_B(\Lambda)$ by hierarchically (recursively) applying the 2-means clustering to $\mathcal{N}_B(\Lambda)$ until each cluster p in $\mathcal{P}^{(r)}$ contains less than N_M samples. We also define a minimum number of cluster members N_m under which we do not use the cluster for class posterior probability estimation. Here, each iteration r is controlled by a specific random initialization of the 2-means clusterings, and $\mathcal{P}^{(r)}$ is the set of clusters produced by the r-th iteration.

Given a cluster centroid $\boldsymbol{a}$ in $\mathcal{P}^{(r)}$, we denote $k(\boldsymbol{a})$ as the number of samples in the cluster that surrounds $\boldsymbol{a}$, and $k_i(\boldsymbol{a})$ as the number of samples in this cluster that belong to class C_i. For convenience, we denote $C^{(r)}$ as the set of the cluster centroids associated with $\mathcal{P}^{(r)}$. With these notations, the application of Eq. (4.1) and Eq (2.4) to the clusters obtained at iteration r gives:

$$U(\Lambda)^{(r)} = -\sum_{\boldsymbol{a}\in C^{(r)}} \frac{k(\boldsymbol{a})}{k(\mathcal{P}^{(r)})}\left[\frac{k_1(\boldsymbol{a})}{k(\boldsymbol{a})}\log\frac{k_1(\boldsymbol{a})}{k(\boldsymbol{a})} + \frac{k_2(\boldsymbol{a})}{k(\boldsymbol{a})}\log\frac{k_2(\boldsymbol{a})}{k(\boldsymbol{a})}\right], \tag{2.5}$$

where $U(\boldsymbol{a};\Lambda) = \frac{k_1(\boldsymbol{a})}{k(\boldsymbol{a})}\log\frac{k_1(\boldsymbol{a})}{k(\boldsymbol{a})} + \frac{k_2(\boldsymbol{a})}{k(\boldsymbol{a})}\log\frac{k_2(\boldsymbol{a})}{k(\boldsymbol{a})}$ contributes to $U(\Lambda)^{(r)}$ with a weight $k(\boldsymbol{a})/k(\mathcal{P}^{(r)})$. This simplifies to:

$$U(\Lambda)^{(r)} = -\frac{1}{k(\mathcal{P}^{(r)})}\sum_{\boldsymbol{a}\in C^{(r)}}\left[k_1(\boldsymbol{a})\log\frac{k_1(\boldsymbol{a})}{k(\boldsymbol{a})} + k_2(\boldsymbol{a})\log\frac{k_2(\boldsymbol{a})}{k(\boldsymbol{a})}\right], \tag{2.6}$$

We use as final estimate $U(\Lambda)$ of the boundary uncertainty the average over each estimate $U^{(r)}(\Lambda)$. If we denote the concatenation of $C^{(1)}, ..., C^{(r)}$ as C, and the concatenation of $\mathcal{P}^{(1)}, ..., \mathcal{P}^{(R)}$ as $\mathcal{P}$,

then $U(\Lambda)$ is written as:

$$U(\Lambda) = -\frac{1}{k(\mathcal{P})} \sum_{\boldsymbol{a} \in C} \left[k_1(\boldsymbol{a}) \log \frac{k_1(\boldsymbol{a})}{k(\boldsymbol{a})} + k_2(\boldsymbol{a}) \log \frac{k_2(\boldsymbol{a})}{k(\boldsymbol{a})} \right], \tag{2.7}$$

Algorithm 3: Hierarchical divisive clustering

Input: $\mathcal{N}_B(\Lambda), N_m, N_M, R$

Output: $\mathcal{P}$

$\mathcal{P} \leftarrow \emptyset$
for $r \in [1, R]$ **do**
  $\mathcal{P}^{(r)} \leftarrow \mathcal{N}_B(\Lambda)$
  while $\exists p \in \mathcal{P}^{(r)} : k(p) > N_M$ **do**
    for $p \in \mathcal{P}^{(r)}$ **do**
      if $k(p) > N_M$ **then** Perform 2-means clustering(p) ;
      if $k(p) < N_m$ **then** Remove p from $\mathcal{P}^{(r)}$;
    end
  end
  $\mathcal{P} \leftarrow \mathcal{P}, \mathcal{P}^{(r)}$
end
return $\mathcal{P}$

Incidentally, if all the training samples have the same estimated label, then it is neither possible to generate anchors, nor to select $\mathcal{N}_B(\Lambda)$. This case corresponds to a strong bias of $B(\Lambda)$, and we accordingly assign a default worst value for $U(\Lambda)$.

2.3.4 Adaptation of Step 1 to multi-class datasets

Algorithm 4: Anchor generation based on one random sample pair for multi-class data

Input: Random sample pair $\{a, b\}$, that satisfies $C(a) \neq C(b)$

1 $\boldsymbol{x}_a \leftarrow a, \boldsymbol{x}_b \leftarrow b$

2 **while** $[\boldsymbol{x}_a, \boldsymbol{x}_b]$ *does not satisfy* $(\mathcal{R})$ **do**

3 $\quad \boldsymbol{x}_b \leftarrow (\boldsymbol{x}_a + \boldsymbol{x}_b)/2;$

4 **end**

5 Generate an anchor from $[\boldsymbol{x}_a, \boldsymbol{x}_b]$ and then select the nearest near-boundary sample s as in Algorithm 2

6 **return** s

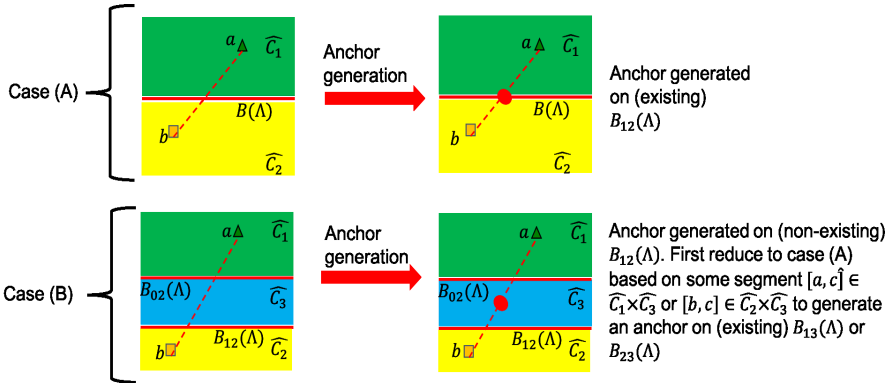

Figure 2.2: Anchor generation based on a random pair $[a, b]$ belonging to a pair of different estimated classes $\widehat{C}_1$ and $\widehat{C}_2$, where we illustrate the need for a case-by-case treatment. In the two-class data case, the segment $[a, b]$ strides only two class regions, and only the boundary $B_{12}(\Lambda)$ can pass through the segment (case (A)). In the multi-class data case, the segment $[a, b]$ can stride multiple class regions, and multiple boundaries can pass through the segment (case (B)); when the segment strides multiple class regions, we divide the segment in the class-by-class manner and generate an anchor for every pair of two adjacent classes.

For the case of two classes C_1 and C_2, Algorithm 2 used the magnitude of $g_1(\cdot;\Lambda) - g_2(\cdot;\Lambda)$ along random segments $[\boldsymbol{x}, \boldsymbol{x}']$ to generate anchors on the decision boundary between $\widehat{C}_1$ and $\widehat{C}_2$. However, in the multi-class case, if a third estimated class $\widehat{C}_3$ lies between $\widehat{C}_1$ and $\widehat{C}_2$, then considering the difference between $g_1(\cdot;\Lambda)$ and $g_2(\cdot;\Lambda)$ no longer makes any sense (case (B) in Figure 2.2). In such a case, before applying Algorithm 2, we must come back to a segment

$[\boldsymbol{x}_a, \boldsymbol{x}_b]$, along which we consider only two adjacent estimated classes, and then instead consider either $g_1(\cdot;\Lambda) - g_3(\cdot;\Lambda)$ to generate anchors on $B_{13}(\Lambda)$ or $g_2(\cdot;\Lambda) - g_3(\cdot;\Lambda)$ to generate anchors on $B_{23}(\Lambda)$.

A segment $[\boldsymbol{x}_a, \boldsymbol{x}_b]$ is included in a region of adjacency between only two estimated classes, if $\forall \boldsymbol{x} \in [\boldsymbol{x}_a, \boldsymbol{x}_b]$ and the two highest discriminant function scores are $g_{C(\boldsymbol{x}_a)}(\boldsymbol{x}, \Lambda)$ and $g_{C(\boldsymbol{x}_b)}(\boldsymbol{x}, \Lambda)$ (this situation corresponds to case (A) in Figure 2.2.). In practice, we approximate this property using the two highest discriminant function scores at only $(\boldsymbol{x}_a + \boldsymbol{x}_b)/2$ (Algorithm 4). For convenience, we call this approximated property $(\mathcal{R})$. So long as $(\mathcal{R})$ is not satisfied, we halve the considered segment (case (B) in Figure 2.2). Incidentally, after an anchor is produced on $B_{ij}(\Lambda)$, it is reasonable to search for only its one nearest neighbor among the near-boundary samples $\boldsymbol{x} \in \widehat{C}_i$ or $\boldsymbol{x} \in \widehat{C}_j$.

Algorithm 2 enables the creation of anchors on specific estimated boundaries; however, its randomness leaves little control over which boundary $B_{ij}(\Lambda)$ should create an anchor. To address this potential issue, we preliminarily sorted all samples in a matrix A, whose element A_{ij} contains the list of training samples for which the highest discriminant function score is $g_i(\cdot;\Lambda)$ and the second highest score is $g_j(\cdot;\Lambda)$ (Algorithm 5). When selecting candidates for near-boundary sample set $\mathcal{N}_{ij}^B(\Lambda)$ for C_i and C_j, we used A to preferentially form anchors based on pairs of samples $\boldsymbol{x}_a, \boldsymbol{x}_b \in A_{ij} \times A_{ji}$.

The number of anchors to be generated can be fixed similarly to that in the two-class case. The final near-boundary set $\mathcal{N}_B(\Lambda)$ is the concatenation of various sets $\{\mathcal{N}_{ij}^B(\Lambda)\}$.

2.3.5 Adaptation of Step 2 to multi-class datasets

For readability in this section, given a sample $\boldsymbol{a}$, we simply denote $i = D_1(\boldsymbol{a})$ and $j = D_2(\boldsymbol{a})$ as the indexes of the two locally highest discriminant functions, and $\{I, II\}$ as the class indexes of the two locally highest class posterior probabilities. In the multi-class case, it might happen that $\{i, j\} \neq \{I, II\}$. This corresponds to a portion of boundary $B_{ij}(\Lambda)$ that is not even on a region near

Algorithm 5: Selection of $\mathcal{N}_B(\Lambda)$ for multi-class data

Input: Classifier parameters Λ trained on $\mathcal{T}$

$\forall i, j \in [1, J], \mathcal{N}_{ij}^B(\Lambda) \leftarrow \emptyset$
Construct A
for $i \in [1, J]$ **do**
  for $j \in [1, J]$ **do**
    while *stop criterion not met* **do**
      if $A_{ij} = \emptyset$ **then** break;
      else
        if $A_{ji} \neq \emptyset$ **then**
          Select random pair $\{a, b\} \in A_{ij} \times A_{ji}$;
        end
        else
          Select random pairs $\{a, b\} \in A_{ij} \times A_{ki}, k \neq j$;
        end
        Denote s: output of Algorithm 4 provided with input (a, b);
        If $s \notin \mathcal{N}_{ij}^B(\Lambda)$ then add to $\mathcal{N}_{ij}^B(\Lambda)$;
      end
    end
  end
end
return $\mathcal{N}_B(\Lambda) = \{\mathcal{N}_{ij}^B(\Lambda)\}_{i,j\in[1,J]^2}$

true classes C_i and C_j, in other words $B_{ij}(\Lambda)$ is strongly biased around $\boldsymbol{a}$. In this case, by default we set the uncertainty measure to its worst value. Similarly to the concept of entropy branching, this results in the following conditional branching definition of $U_{ij}(\boldsymbol{a}; \Lambda)$:

I. If $\{I, II\} \neq \{i, j\}$, then set $U_{ij}(\boldsymbol{a}; \Lambda)$ to its worst value.

II. Else normalize $P(C_I|\boldsymbol{a})$ and $P(C_{II}|\boldsymbol{a})$ so that their sum equals 1. Apply the uncertainty measure defined in the two-class case to the normalized posterior probabilities, by replacing class indexes $\{1, 2\}$ with $\{I, II\}$. Here, normalization is necessary because formally reducing the problem to two classes assumes $P(C_I|\boldsymbol{a}) + P(C_{II}|\boldsymbol{a}) = 1$, which does not necessarily hold in the presence of more than two classes.

2.4 Experimental evaluation

2.4.1 Datasets

We conducted evaluations on fixed-dimensional vector pattern datasets from the UCI Machine Learning Repository[1]. Especially for the Abalone, Wine Quality Red, and Wine Quality White datasets, we used our custom versions, where the original categories were grouped into three categories due to the presence of very few represented classes. For the Wine Quality White dataset, we used a randomly sampled subset of the mother dataset. For analysis purposes, we also prepared a two-dimensional two-class synthetic dataset called GMM, which modeled each class with two Gaussian mixtures and 1100 samples. We summarize these datasets in Table 4.1, where N refers to the number of samples available, D to the dimensionality, and J to the number of classes.

For all datasets, we normalized the datasets, by removing the mean and then scaling to a unit variance dimension-wise.

Table 2.1: Datasets

Dataset	N	D	J	Remarks
GMM	2,200	2	2	synthetic data
Abalone	4,177	7	3	custom version
Breast Cancer	683	9	2	2:1 imbalance
Cardiotocography	1,831	30	2	10:1 imbalance
Ionosphere	351	34	2	2:1 imbalance
Letter Recognition	20,000	16	26	
MNIST (test)	10,000	784	10	
Landsat Satellite	6,435	36	7	
Sonar (all data)	208	60	2	
Spambase	4,601	57	2	
Thyroid	7,200	21	3	18:1:1 imbalance
Wine Quality Red	1599	11	3	custom version
Wine Quality White	1470	11	3	custom version

[1] http://archive.ics.uci.edu/ml/index.php

2.4.2 Classifier

As a classifier for evaluation, we chose SVM [3] using a Gaussian kernel whose implementation is available online[2]. In this case, Λ consists of a set of kernel weights optimized during the training, regularization parameter C, and Gaussian kernel width γ. To simplify the analysis, we fixed C beforehand using another CV-based preliminary experiment and then focused on the optimal setting (status selection) of the single parameter γ .

For the multi-class datasets, we used a one-versus-all multi-class SVM. Actually, the one-versus-all formalization of the multi-class problem is different from the multi-class formalization described in Section 2.2.1. However, in our understanding, the SVM implementation that we used draws boundaries such that a region near the decision boundary $B_{ij}(\Lambda)$ is characterized by $g_i(\cdot;\Lambda)$ and $g_j(\cdot;\Lambda)$, being the two highest discriminant function scores. This is adequate for Step 1 (Section 2.3.4) to be applicable.

2.4.3 Hyperparameters

Although our proposed method contains several hyperparameters, we indifferently set them to the same values for all the datasets as follows. In Step 1 (Algorithm 2), we set the maximum number of dichotomy loops to 30 and the maximum number of repetitions l_M for generating anchors to a high value such as 10,000. In Step 2 (Algorithm 3), N_m and N_M control the granularity of the clusters, each of which should ideally have as small a volume as possible and yet contain enough samples to perform a reliable estimation of the ratio between class posterior probabilities. We imposed clusters to contain 10 or fewer samples by setting $N_m = 8$ and $N_M = 12$. Moreover, the number of repetitions R should be higher than 1 to increase the reliability of the estimation, so we set it to 10.

[2]http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

2.4.4 Evaluation estimation

We want to check whether classifier statuses that were assigned a high boundary uncertainty by our evaluation method really correspond to more optimal classifier statuses, and in particular whether the highest boundary uncertainty really corresponds to the most optimal classifier status. However, so far evaluating the optimality of a classifier status requires to know (the true value of) its error probability, which is unknown.

So far, CV is one of the most reliable methods to estimate the error probability. In our experiments, the CV estimate of the error probability therefore plays both the role of a competitor, and of a reference that is assumed close to the true value of the error probability. To further ensure a reliable estimation of the error probability, we preliminary tested several values of the number of folds in CV, and increased it until no significant change was observed in the resulting error probability estimation. Ten folds seemed sufficient to estimate the error probabilities of SVMs trained with different values of γ. Only for the Breast Cancer, Ionosphere, and Sonar datasets, we applied Leave One Out evaluation owing to the few samples available. We denote L_{val} as the CV estimate of the error probability, and remind that L_{val} is the average over the empirical error rates corresponding to each validation fold. To later visualize the gap between the empirical error rate on the training data and the true error probability (here approximated by L_{val}), we also computed the average L_{tr} of the empirical error rates corresponding to the training folds.

For the GMM dataset, in order to obtain a nearly perfect estimate of the error probability, we simulated infinite data by generating another large validation set made of $20,000$ independent samples using the same Gaussian mixture model as that used for the original 1100 samples. We denote L_{val2} as the error probability estimate based on this large validation set, and we assume that L_{val2} is the true value of the error probability.

To compute our boundary uncertainty, we used all of the samples for each dataset. For convenience of discussion, we use a sign-reversed measure $-U(\Lambda)$ and analyze the similarity between $-U(\Lambda)$ and L_{val} in later sections. In our graphs (Figures 2.3, 2.4, 2.4, L_{tr}, L_{val}, L_{val2}, and $-U$ are represented by a yellow, green, red, and blue curve respectively.

It is known that the quality of posterior probability estimation can degrade when the samples are imbalanced in classes, and traditional solutions to the presence of an under-represented minority class modify the classifier objective or resample the minority-class samples to force class balance [27]. In our method, rather than changing the samples or the classification process, we simply evaluate classification uncertainty in a non-intrusive way. Our solution is to superficially take the class imbalance into account during the posterior probability estimation. More precisely, we replace the estimated posterior probabilities $\widehat{P}(C_i|\boldsymbol{x})$ with $\widehat{P}(C_i|\boldsymbol{x})/\widehat{P}(C_i)$ in Eq. (2.4). By doing so, we will raise the estimated posterior probability for a low prior probability, while we will bring down the estimated posterior probability for a high prior probability. The next sections assume this simple correction.

2.4.5 Results of Step 1

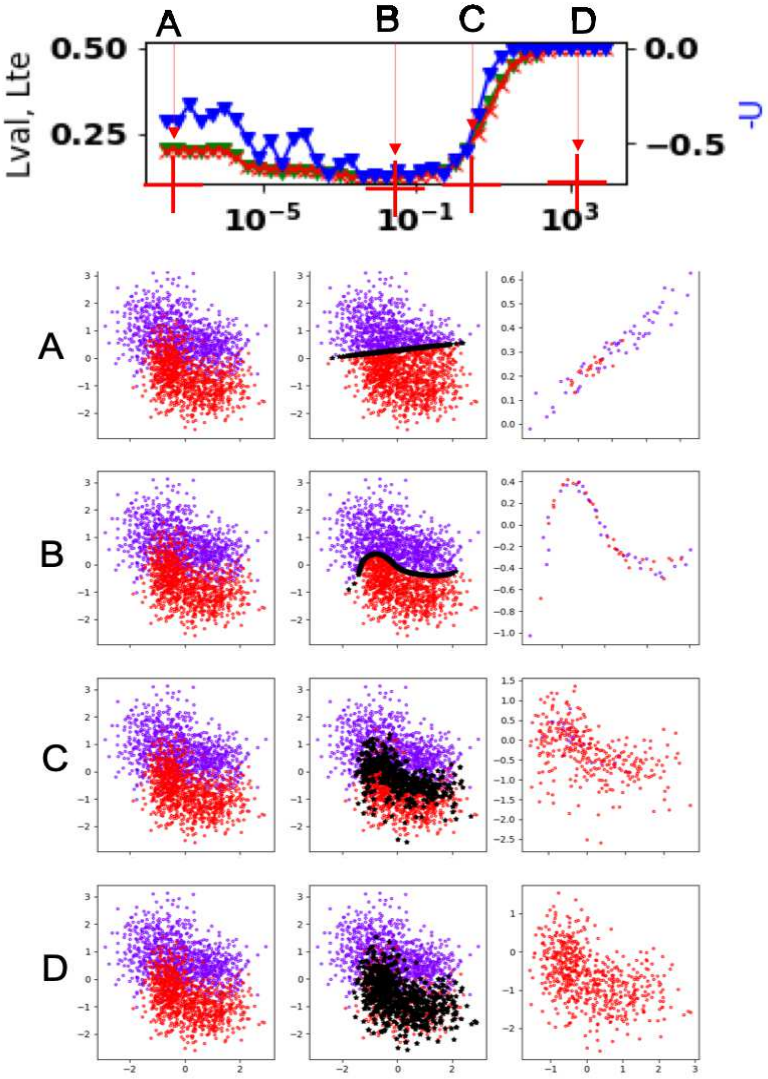

Figure 2.3: SVM evaluation results on the GMM dataset. In the upper graph, the horizontal axis corresponds to the kernel width γ, the left vertical axis corresponds to L_{te} (red) and L_{val} (green), and the right vertical axis corresponds to $-U$ (blue). For each of the four classifier statuses A, B, C, D, we represent the following three plots in the corresponding row. From left to right: data with true class labels; data with labels estimated by the classifier, and anchors in black dots; zoom on the set of one nearest neighbors of the anchors, represented with their true class label. These nearest neighbors are assumed representative of the decision boundary. Local balance between purple and red labels all along the decision boundary corresponds to more optimal classifier statuses.

For the GMM dataset, we observe the near-boundary sets obtained by Algorithm 2 with different settings of γ (Figure 2.3). A higher γ should correspond to a more complex boundary (a lower one for a simpler boundary). The results show that the selected near-boundary samples were accurately selected along the decision boundary. For excessively low values of γ, the estimated boundaries were too simple. For excessively high values of γ, the estimated boundaries were very complex, and formed discontinuous regions in the data space. For $\gamma = 2$, the decision boundary seems identical to the Bayes boundary.

2.4.6 Results on the classifier selection

On figure 2.3, L_{val} (green) and L_{val2} (red) closely fit each other. This confirms that the CV-based estimation of the error probability can reliably serve as a competitor and reference in our comparison.

For all datasets in Figures 2.3, 2.4, 2.4, for lower boundary complexity (lower γ), L_{tr} is close to L_{val}, but a gap between these two estimates can be seen growing wider and wider as the classifier draws more complex boundaries (higher γ). This confirms the impossibility of evaluating a classifier using the classification error on the training data, as predicted by [2, 28]. Next, to assess the utility of our boundary uncertainty estimate, we check whether its values and its trend behave as expected.

Trend of the boundary uncertainty. For almost all the datasets, our uncertainty measure $-U$ (blue) almost always shows the same trend as L_{val}; in particular, the minimum of $-U$ is close to the minimum of L_{val}, namely to the optimal classifier status. As found in the values of L_{val}, classification for the Abalone, Landsat Satellite, Wine Quality Red, and Wine Quality White datasets was rather difficult; for these datasets, the Bayes error values estimated by CV were higher than 0.3. Even for these difficult datasets, $-U$ closely followed L_{val}, which shows the potential of our uncertainty measure-based method to select the optimal classifier status.

Value of the boundary uncertainty. As explained in Chapter 1, based on mild assumptions that we can check more thoroughly in a later stage of our research, a classifier status is optimal if and only if the boundary uncertainty reaches its maximum value, and the latter is the same regardless of the dataset. Using the binary Shannon entropy as the uncertainty measure gives a maximum boundary uncertainty value of $\ln 2$. We can use this particular value in several ways based on either of these three sets of assumptions:

- Assuming that the set of candidate classifier statuses contains the optimal classifier status, and assuming that CV is reliable, then the minimum of CV corresponds to the optimal classifier status, and $-U$ (blue curve) should be $-\ln 2 \approx -0.7$. Under these assumptions, we can assess the reliability of our boundary uncertainty estimation by checking whether the minimum value of $-U$ is around -0.7.
- Assuming that CV is reliable, but assuming that the set of candidate classifier statuses does *not* contain the optimal classifier status, then we cannot assess the reliability of our boundary uncertainty estimation solely based on its values, and observing its trend instead is more useful. This case may happen if the classifier is intrinsically not powerful enough, or if the dataset is very difficult, or if we did not appropriately chose our candidate classifier statuses.
- Assuming that our boundary uncertainty is reliable, then we can check whether the set of candidate classifier statuses contains the optimal classifier status, by checking if the minimum value observed for $-U$ is around -0.7.

At this early stage of our research, the first and the second set of assumptions are safer than the third one. If we assume the first set of assumptions, then we can see that on the Breast Cancer, GMM, Wine Quality Red, and Wine Quality White datasets, $-U$ reaches a minimum value of around -0.7, which may indicate a reliable boundary uncertainty estimation; on the other datasets, the minimum of $-U$ is quite different from -0.7, therefore on these more difficult tasks the boundary uncertainty estimation may lack reliability and require refinements. However, on the Letter Recognition, MNIST (test), Spambase and Sonar datasets, the trend of $-U$ looks quite reasonable. Therefore, the second and third set of assumptions might also be reasonable,

and it might mean that owing to the difficulty of these two datasets, the classifier could not achieve the optimal classifier status. However, based on the notorious difficulty of the Wine Quality Red and Wine Quality White datasets, drawing any conclusions is difficult, and either our method or the classifier, or CV, or a combination of the three may suffer from an unreliable estimation.

This discussion above illustrates the difficulty of drawing clearcut conclusions about the classifier optimality with the current approaches. If we improve our boundary uncertainty estimation enough so that its results can always be considered reliable, our boundary uncertainty-based approach may be very useful to draw clear conclusions about the classifier optimality.

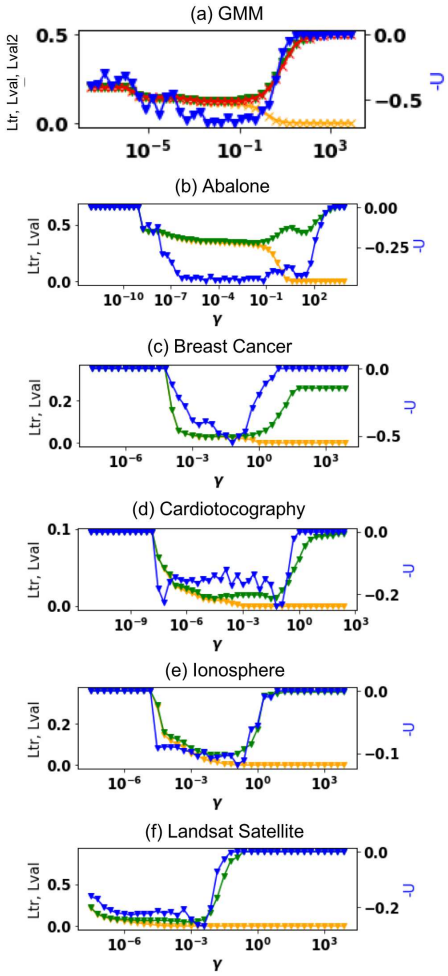

Figure 2.4: Classifier evaluation results for synthetic GMM dataset and 11 real-life datasets. From top to bottom: GMM, Abalone, Breast Cancer, Cardiotocography, Ionosphere, and Landsat Satellite. Horizontal axis indicates the value of γ. Each panel shows the estimated error probability curve based on the CV's training folds (L_{tr}, *yellow*), the estimated error probability curve based on the CV's validation folds (L_{val}, *green*), and our uncertainty measure curve ($-U$, blue). In the top GMM panel, we also display the error probability estimate based on the extra 20,000 validation samples (L_{val2}, red).

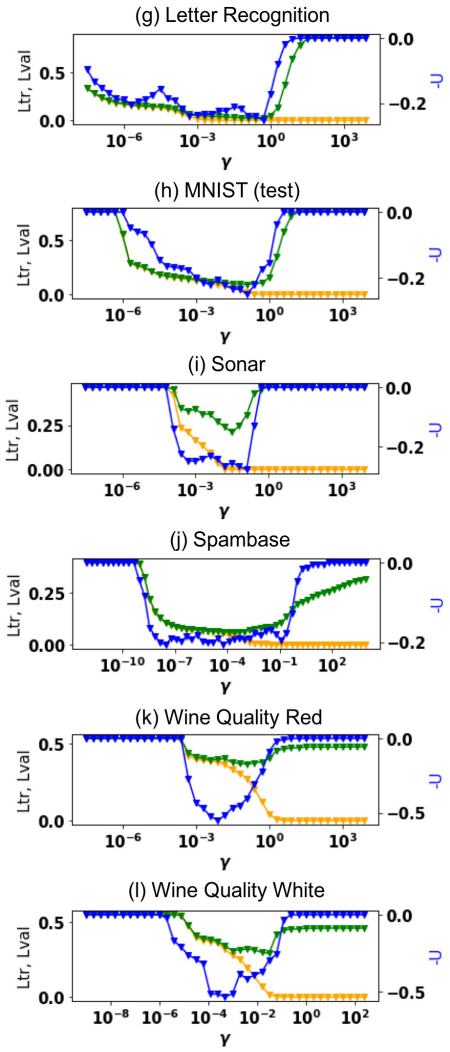

Figure 2.4 (cont.): From top to bottom: Letter Recognition, MNIST (test), Sonar, Spambase, Wine Quality Red, and Wine Quality White.

2.4.7 Influence of data imbalance on classifier status selection

To better understand the influence of class imbalance, we focus on the Cardiotocography dataset (176 samples for C_0 and 1,655 samples for C_1) in Figure 2.5. We break down the number of near-boundary samples (black curve) into the numbers of near-boundary samples belonging to C_0 (gray curve) and to C_1 (red curve). As γ increases, the number of near-boundary samples belonging to C_0 rapidly increases, stays high, and drastically decreases in $N_B(\Lambda)$, while the

number of samples belonging to C_1 almost always stays around 176; for most γ values, the total number of near-boundary samples is dominated by one of the two classes. In this case, the computation of the uncertainty measure is obviously biased and there is no way that uncertainty around B^* can be achieved. The posterior probability computation with the superficial prior probability correction described in Section 2.4.4 indeed gives better results (blue curve) than without the prior correction (red dashed curve); however, more efficient posterior estimation methods must be applied for such an imbalanced case.

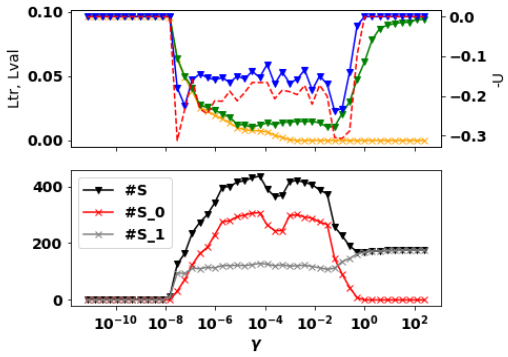

Figure 2.5: Parameter status selection results for Cardiotocography data. In the top panel, the blue curve represents uncertainty measure $-U$ with the prior probability correction; the red dashed curve for $-U$ without the correction. In the bottom panel, the black curve represents the number of near-boundary samples; the gray and red curves represent the numbers of near-boundary samples belonging to C_0 and to C_1, respectively.

2.5 Summary

In order to overcome the fundamental limitations of the standard error probability-based methods, we defined a new robust classifier evaluation criterion that can potentially be estimated without the need for validation data for any classifier model, and takes the optimal boundary as a reference. The experimental results and a comparison with the benchmark CV method indicate the possibility of selecting the optimal model on several real-life classification tasks. Despite the encouraging results, there is room for improvement in terms of accuracy, furthermore the proposed method relies on random repetitions and some heuristic settings that can be

time-consuming. A better use of the information provided by the discriminant functions can probably simplify the current complicated Step 1. Furthermore, a theoretical analysis about the quality of the proposed boundary uncertainty estimation is necessary.

3 Optimality analysis of uncertainty measure

The reliability of our proposal in Chapter 2 relies on the quality of the estimation of the uncertainty measure, which relies on the estimation of class posterior probabilities along the estimated boundary. In this chapter, we mathematically analyze the class posterior probability estimation procedure introduced in Section 2.3.3, by analyzing its convergence properties. We show how the reliability of this estimator ensures that our boundary uncertainty-based classifier evaluation can reliably evaluate classifier statuses. Then, we experimentally validate our analysis by comparing the results from Chapter 2 with boundary uncertainty-based classifier evaluation performed with more naive class posterior probability estimation methods.

3.1 Overview and preparations

We consider point $\boldsymbol{x}$ in a finite-dimensional Euclidian space; furthermore let $\boldsymbol{X}_1, ..., \boldsymbol{X}_N$ be N samples independently sampled from a probability distribution function (pdf) $p(\cdot)$[1]. Then, every point in the space is assumed to belong to one or multiple classes among J classes $C_1, ..., C_J$[2]. Moreover, letting $\mathcal{R}^{(N)}(\boldsymbol{x})$ and $V^{(N)}(\boldsymbol{x})$ be a small region containing $\boldsymbol{x}$ and the volume of this region, respectively, we assume the following:

I. The distance between any two points in $\mathcal{R}^{(N)}(\boldsymbol{x})$ goes to 0 as $N \to \infty$.

II. pdf $p(\cdot)$ and class likelihoods $p(\cdot|C_j)$ $(j = 1, ..., J)$ are continuous functions.

III. For all N and all $\boldsymbol{x}$, $\mathcal{R}^{(N)}(\boldsymbol{x})$ is a finite and closed set.

[1]The term "sample" means random variable in this subsection.
[2]Class label (index) is basically a random variable in this subsection.

In the above, no assumption is made about the shape of $\mathcal{R}^{(N)}(\boldsymbol{x})$.

3.2 Convergence to true value for ratio-based probability density estimator

First, the probability $P^{(N)}(\boldsymbol{x})$ that sample $\boldsymbol{X}$ is contained in $\mathcal{R}^{(N)}(\boldsymbol{x})$ is equal to the expected value of $1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X})$:

$$P^{(N)}(\boldsymbol{x}) = \mathrm{E}\Big[1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X})\Big] = \int_{\mathcal{R}^{(N)}(\boldsymbol{x})} p(\boldsymbol{u})d\boldsymbol{u}, \tag{3.1}$$

where $1_A(\cdot)$ refers to the indicator function that outputs 1 if the predicate A holds true, and 0 otherwise. Then, there exists point $\boldsymbol{x}^{(N)} \in \mathcal{R}^{(N)}(\boldsymbol{x})$ such that $P^{(N)}(\boldsymbol{x}) = p(\boldsymbol{x}^{(N)}) \cdot V^{(N)}(\boldsymbol{x})$ (see Appendix 1). Furthermore, the variance of $1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X})$ is

$$\begin{aligned}
\mathrm{Var}\Big[1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X})\Big] &= \mathrm{E}\Big[\{1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X})\}^2\Big] - \{P^{(N)}(\boldsymbol{x})\}^2 \\
&= \mathrm{E}\Big[1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X})\Big] - \{P^{(N)}(\boldsymbol{x})\}^2 \\
&= P^{(N)}(\boldsymbol{x})\{1 - P^{(N)}(\boldsymbol{x})\}.
\end{aligned} \tag{3.2}$$

For the N i.i.d. samples, we next denote the number of samples contained in $\mathcal{R}^{(N)}(\boldsymbol{x})$ by $K^{(N)}(\boldsymbol{x})$. Then, $K^{(N)}(\boldsymbol{x})$ is represented as

$$K^{(N)}(\boldsymbol{x}) = \sum_{n=1}^{N} 1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X}_n). \tag{3.3}$$

Because $\boldsymbol{X}_1, ..., \boldsymbol{X}_N$ are i.i.d. samples, the expectation and variance of $K^{(N)}(\boldsymbol{x})$ are

$$\begin{aligned}
\mathrm{E}\Big[K^{(N)}(\boldsymbol{x})\Big] &= N\mathrm{E}\Big[1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X})\Big] \\
&= NP^{(N)}(\boldsymbol{x}), \qquad (3.4) \\
\mathrm{Var}\Big[K^{(N)}(\boldsymbol{x})\Big] &= N\mathrm{Var}\Big[1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X})\Big] \\
&= NP^{(N)}(\boldsymbol{x})\{1 - P^{(N)}(\boldsymbol{x})\}, \qquad (3.5)
\end{aligned}$$

and accordingly the following holds:

$$\begin{aligned} \mathrm{E}\Big[\frac{K^{(N)}(\boldsymbol{x})}{NV^{(N)}(\boldsymbol{x})}\Big] &= \frac{1}{NV^{(N)}(\boldsymbol{x})}\mathrm{E}\Big[K^{(N)}(\boldsymbol{x})\Big] \\ &= p(\boldsymbol{x}^{(N)}), \qquad (3.6) \\ \mathrm{Var}\Big[\frac{K^{(N)}(\boldsymbol{x})}{NV^{(N)}(\boldsymbol{x})}\Big] &= \frac{NP^{(N)}(\boldsymbol{x})\{1-P^{(N)}(\boldsymbol{x})\}}{\{NV^{(N)}(\boldsymbol{x})\}^2} \\ &= \frac{p(\boldsymbol{x}^{(N)})\{1-P^{(N)}(\boldsymbol{x})\}}{NV^{(N)}(\boldsymbol{x})}. \qquad (3.7) \end{aligned}$$

If $V^{(N)}(\boldsymbol{x})$ is chosen so that it satisfies $V^{(N)}(\boldsymbol{x}) \to 0$ and $NV^{(N)}(\boldsymbol{x}) \to \infty$[3], then from the continuity of $p(\cdot)$ it follows that

$$p(\boldsymbol{x}^{(N)}) \to p(\boldsymbol{x}) \qquad (N \to \infty); \tag{3.8}$$

furthermore, from Eqs. (3.6) and (3.7), we reach

$$\mathrm{E}\Big[\frac{K^{(N)}(\boldsymbol{x})}{NV^{(N)}(\boldsymbol{x})}\Big] \to p(\boldsymbol{x}), \mathrm{Var}\Big[\frac{K^{(N)}(\boldsymbol{x})}{NV^{(N)}(\boldsymbol{x})}\Big] \to 0 (N \to \infty). \tag{3.9}$$

Therefore, $K^{(N)}(\boldsymbol{x})/\{NV^{(N)}(\boldsymbol{x})\}$ converges to $p(\boldsymbol{x})$ in the mean-square (L_2) sense, and in particular to $p(\boldsymbol{x})$ in probability:

$$\frac{K^{(N)}(\boldsymbol{x})}{NV^{(N)}(\boldsymbol{x})} \xrightarrow{\mathrm{P}} p(\boldsymbol{x}) \qquad (N \to \infty), \tag{3.10}$$

where $\xrightarrow{\mathrm{P}}$ denotes convergence in probability.

3.3 Convergence to true value for ratio-based joint probability density estimator

The reasoning in this subsection is similar to that in the previous subsection.

The data available for training is a set of N realizations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \ldots, \boldsymbol{x}_N\}$, where $\boldsymbol{x}_n$ is a realization of $\boldsymbol{X}_n$ that is deterministically associated with class label (index) y_n. On the other

[3]For example, given $V > 0$, we can set $V^{(N)}(\boldsymbol{x})$ as $V/(N^{\alpha})$ $(0 < \alpha < 1)$ or $V/(\ln N)^{\beta}$ $(N > 1, \beta > 0)$, etc.

hand, each random variable $\boldsymbol{X}_n$ ($n = 1, \ldots, N$) is probabilistically associated with multiple class labels, since different class regions can overlap in the Euclidean sample space. Therefore, we regard a class label for $\boldsymbol{X}_n$ ($n = 1, \ldots, N$) as a random variable Y_n ($\in \{1, \ldots, J\}$) and discuss sample pairs $\{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n), \ldots, (\boldsymbol{X}_N, Y_N)\}$ as follows, assuming that these sample pairs are independent.

For one sample pair $(\boldsymbol{X}, Y)$, we denote the expectation of $1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X}) \cdot 1_{\{j\}}(Y)$ by $P_j^{(N)}(\boldsymbol{x})$. Then, this expectation is the probability that $\boldsymbol{X}$ is included in $V^{(N)}(\boldsymbol{x})$ and labeled by C_j (see Appendix 2):

$$\begin{aligned} P_j^{(N)}(\boldsymbol{x}) &= \mathrm{E}\Big[1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X}) \cdot 1_{\{j\}}(Y)\Big] \\ &= \Pr(C_j) \int_{\mathcal{R}^{(N)}(\boldsymbol{x})} p(\boldsymbol{u}|C_j) d\boldsymbol{u}. \end{aligned} \tag{3.11}$$

Moreover, because $p(\cdot|C_j)$ is continuous, there exists point $\boldsymbol{x}^{(N,j)} \in \mathcal{R}^{(N)}(\boldsymbol{x})$, and $P_j^{(N)}(\boldsymbol{x})$ is equal to $\Pr(C_j) p(\boldsymbol{x}^{(N,j)}|C_j) \cdot V^{(N)}(\boldsymbol{x})$ (see Appendix 1); the variance of $1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X}) \cdot 1_{\{j\}}(Y)$ can be written as

$$\begin{aligned} &\mathrm{Var}\Big[\quad 1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X}) \cdot 1_{\{j\}}(Y)\Big] \\ &= \quad \mathrm{E}\Big[\{1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X}) \cdot 1_{\{j\}}(Y)\}^2\Big] - \{P_j^{(N)}(\boldsymbol{x})\}^2 \\ &= \quad \mathrm{E}\Big[1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X}) \cdot 1_{\{j\}}(Y)\Big] - \{P_j^{(N)}(\boldsymbol{x})\}^2 \\ &= \quad P_j^{(N)}(\boldsymbol{x})\{1 - P_j^{(N)}(\boldsymbol{x})\}. \end{aligned} \tag{3.12}$$

For the N i.i.d. samples, we denote the number of samples included in $\mathcal{R}^{(N)}(\boldsymbol{x})$ and labeled by C_j as $K_j^{(N)}(\boldsymbol{x})$. Then, $K_j^{(N)}(\boldsymbol{x})$ is represented as

$$K_j^{(N)}(\boldsymbol{x}) = \sum_{n=1}^{N} 1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X}_n) \cdot 1_{\{j\}}(Y_n). \tag{3.13}$$

Moreover, because $(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_N, Y_N)$ are i.i.d., the expectation and variance of $K_j^{(N)}(\boldsymbol{x})$ are

$$\mathrm{E}\Big[K_j^{(N)}(\boldsymbol{x})\Big] = NP_j^{(N)}(\boldsymbol{x}), \tag{3.14}$$

$$\mathrm{Var}\Big[K_j^{(N)}(\boldsymbol{x})\Big] = NP_j^{(N)}(\boldsymbol{x})\{1 - P_j^{(N)}(\boldsymbol{x})\}, \tag{3.15}$$

and it follows that

$$\mathrm{E}\Big[\frac{K_j^{(N)}(\boldsymbol{x})}{NV^{(N)}(\boldsymbol{x})}\Big] = \frac{NP_j^{(N)}(\boldsymbol{x})}{NV^{(N)}(\boldsymbol{x})} = \Pr(C_j)p(\boldsymbol{x}^{(N,j)}|C_j), \tag{3.16}$$

$$\begin{aligned}\mathrm{Var}\Big[\frac{K_j^{(N)}(\boldsymbol{x})}{NV^{(N)}(\boldsymbol{x})}\Big] &= \frac{NP_j^{(N)}(\boldsymbol{x})\{1 - P_j^{(N)}(\boldsymbol{x})\}}{\{NV^{(N)}(\boldsymbol{x})\}^2} \\ &= \frac{\Pr(C_j)p(\boldsymbol{x}^{(N,j)}|C_j)\{1 - P_j^{(N)}(\boldsymbol{x})\}}{NV^{(N)}(\boldsymbol{x})}.\end{aligned} \tag{3.17}$$

Here, if we set $V^{(N)}(\boldsymbol{x})$ similarly to that used to derive Eq. (3.8), and then we reach the following, since $p(\cdot|C_j)$ is a continuous function:

$$\mathrm{E}\Big[\frac{K_j^{(N)}(\boldsymbol{x})}{NV^{(N)}(\boldsymbol{x})}\Big] \to \Pr(C_j)p(\boldsymbol{x}|C_j), \mathrm{Var}\Big[\frac{K_j^{(N)}(\boldsymbol{x})}{NV^{(N)}(\boldsymbol{x})}\Big] \to 0 (N \to \infty). \tag{3.18}$$

Accordingly, $K_j^{(N)}(\boldsymbol{x})/\{NV^{(N)}(\boldsymbol{x})\}$ converges to $\Pr(C_j)p(\boldsymbol{x}|C_j)$ in the mean-square sense, and in particular to $\Pr(C_j)p(\boldsymbol{x}|C_j)$ in probability:

$$\frac{K_j^{(N)}(\boldsymbol{x})}{NV^{(N)}(\boldsymbol{x})} \xrightarrow{\mathrm{P}} \Pr(C_j)p(\boldsymbol{x}|C_j) \qquad (N \to \infty). \tag{3.19}$$

3.4 Probabilistic convergence to true value for ratio-based posterior probability estimator

In previous subsections, we found that the simple ratio-based estimators for the probability density and the joint probability density converge to their true values in probability, respectively (see Eqs. (3.10) and (3.19)). From these results and the nature of four arithmetic operations for the

random variable sequences that converge in probability, we finally obtain

$$\begin{aligned}\frac{K_j^{(N)}(\boldsymbol{x})}{K^{(N)}(\boldsymbol{x})} &= \frac{K_j^{(N)}(\boldsymbol{x})/\{NV^{(N)}(\boldsymbol{x})\}}{K^{(N)}(\boldsymbol{x})/\{NV^{(N)}(\boldsymbol{x})\}} \\ &\xrightarrow{\mathrm{P}} \frac{\Pr(C_j)p(\boldsymbol{x}|C_j)}{p(\boldsymbol{x})} = \Pr(C_j|\boldsymbol{x}) \quad (N \to \infty). \end{aligned} \tag{3.20}$$

Because the above convergences in Eqs. (3.8) through (3.20) assume that $V^{(N)}(\boldsymbol{x}) \to 0$ $(N \to \infty)$, the following should be satisfied for $K^{(N)}(\boldsymbol{x})$, $K_j^{(N)}(\boldsymbol{x})$ and N:

I. If $p(\boldsymbol{x}), p(\boldsymbol{x}|C_j) > 0$, then $K^{(N)}(\boldsymbol{x}), K_j^{(N)}(\boldsymbol{x}) \to \infty$ $(N \to \infty)$.

II. $K^{(N)}(\boldsymbol{x})/N, K_j^{(N)}(\boldsymbol{x})/N \to 0$ $(N \to \infty)$.

The first condition is necessary because if $K^{(N)}(\boldsymbol{x})$ and $K_j^{(N)}(\boldsymbol{x})$ do not go to infinity when $N \to \infty$, then they would tend to 0 for sufficiently small values of $V^{(N)}(\boldsymbol{x})$. The second condition ensures that $K^{(N)}(\boldsymbol{x})/\{NV^{(N)}(\boldsymbol{x})\}$ and $K_j^{(N)}(\boldsymbol{x})/\{NV^{(N)}(\boldsymbol{x})\}$ do not diverge even when the first condition is satisfied.

3.5 Practical advantages supported by optimality in ratio-based posterior probability estimation

In addition to the property of convergence to the true posterior probability in Eq. (3.20), the formalization clearly expresses the practical advantages, which will be useful in a real-life finite sample regime, of our own *k*NN-based posterior probability estimation using only near-boundary samples. Assuming that point $\boldsymbol{x}$ is sufficiently close to estimated boundary $B(\Lambda)$, we summarize them in the following:

I. We should decrease region $\mathcal{R}^{(N)}(\boldsymbol{x})$ to reduce the bias in the expectation of $K_j^{(N)}(\boldsymbol{x})/K^{(N)}(\boldsymbol{x})$ so that $p(\boldsymbol{x}^{(N)})$ and $p(\boldsymbol{x}^{(N,j)}|C_j)$ become closer to $p(\boldsymbol{x})$ in Eq. (3.6) and $p(\boldsymbol{x}|C_j)$ in Eq. (3.16), respectively.

II. By increasing $\mathcal{R}^{(N)}(\boldsymbol{x})$, we can basically reduce the variance of kNN-based posterior probability estimate $K_j^{(N)}(\boldsymbol{x})/K^{(N)}(\boldsymbol{x})$. This result can also be proved in a more accurate manner using the perturbation technique [29].

III. When estimated boundary $B(\Lambda)$ is close to the Bayes boundary, we can reduce both the bias and the variance, based on Eqs. (3.6), (3.7), (3.16), and (3.17), in the posterior probability estimate by applying region $\mathcal{R}^{(N)}(\boldsymbol{x})$ to only the near-boundary samples in $\mathcal{N}_B(\Lambda)$ and increasing its size (in other words, increasing the number of the near-boundary samples used for kNN-based posterior probability estimation). Here, all individual estimates $P(C_j|\boldsymbol{x})$ will be close to 0.5, and, moreover, the numerators in Eqs. (3.7) and (3.17) are bounded by $N/4$; therefore, the corresponding variances are bounded by $1/\{4N(V^{(N)}(\boldsymbol{x}))^2\}$ and tend to 0 as $V^{(N)}(\boldsymbol{x})$ grows larger. This valuable property enables our method to use large regions (clusters) for the posterior probability estimation in Algorithm 3 and leads to accurate and reliable performances even if it adopts a simple kNN-based estimation.

IV. When accepting the incursion of samples outside $\mathcal{N}_B(\Lambda)$ to region $\mathcal{R}^{(N)}(\boldsymbol{x})$ and increasing its size, we clearly face a dilemma between the bias and the variance: Increasing $\mathcal{R}^{(N)}(\boldsymbol{x})$ decreases the variance but increases the bias, while decreasing $\mathcal{R}^{(N)}(\boldsymbol{x})$ increases the variance but decreases the bias. This phenomenon, which is generally observed in a regular kNN-based posterior probability estimation, proves again the validity of using only near-boundary samples for posterior probability estimation.

V. All the estimators' properties such as the convergence to true value hold regardless of the shape of region $\mathcal{R}^{(N)}(\boldsymbol{x})$.

3.6 Experiments

To validate the analysis conducted in this chapter, we performed a step-by-step analysis of Step 2, which executes the posterior probability estimation using the kNN. However, instead of the traditional estimation based on a fixed-size neighborhood selected from the entire $\mathcal{T}$, we chose to consider only the near-boundary samples in $\mathcal{N}_B(\Lambda)$ as neighbors and also determined the number

of neighbors adaptively by the hierarchical clustering procedure.

In this section, we analyze the influence of this choice on the quality of the estimation of posterior probabilities along the estimated boundary. To this end, we compared the overall parameter status selection results obtained on the Ionosphere dataset by three neighbor-selection schemes (Figure 3.1): fixed kNN considering neighbors selected from the entire $\mathcal{T}$ (top panel), fixed kNN considering neighbors selected from $\mathcal{N}_B(\Lambda)$ (middle panel), and adaptive partitioning in $\mathcal{N}_B(\Lambda)$ (bottom panel). For the top and middle panels, we first fixed k to 5 (results shown using different values of k). The setting of N_m, N_M, R, for the adaptive partitioning was the same as in Chapter **??**.

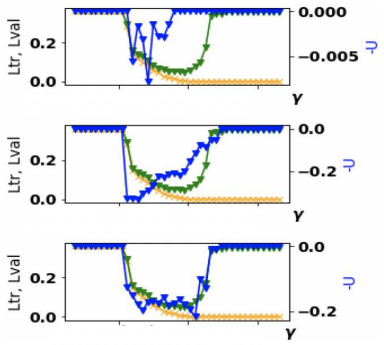

Figure 3.1: Comparison of three neighbor selection schemes in Step 2 (near-boundary sample selection) for Ionosphere dataset: fixed kNN considering neighbors selected from $\mathcal{T}$ (top panel), fixed kNN considering neighbors selected from $\mathcal{N}_B(\Lambda)$ (middle panel), and adaptive partitioning in $\mathcal{N}_B(\Lambda)$ (bottom panel).

First, the top panel basically shows noise: there is no trend and the range of values for $-U(\Lambda)$ is close to 0 (see the right vertical axis), which means the method does not measure any reliable uncertainty measure score around $B(\Lambda)$, even when $B(\Lambda)$ is close to B^*. By contrast, the middle panel seems to clearly detect a range of suitable candidate parameter values similar to the range provided by the CV-based method. The neat improvements from the top panel to the middle panel show the necessity of filtering out samples outside of $\mathcal{N}_B(\Lambda)$. Here, note that the range of $-U(\Lambda)$ in the top panel is significantly smaller than that in the middle and bottom panels. Accordingly, these results can be understood as follows. To evaluate the classifier status, our measure strictly focuses on estimation of the posterior probability imbalance *on the estimated*

boundary. In practice, with finite samples, $\mathcal{N}_B(\Lambda)$ contains only samples close to $B(\Lambda)$. Step 1 can be seen as a filter that effectively cuts noise, i.e. samples away from the estimated boundary.

Second, the clear improvement from the middle panel to the bottom panel shows that the adaptive partition of the near-boundary samples further enhances the quality of the neighbors used in the kNN-based estimation. Intuitively, adapting the number of neighbors to the local density and ignoring excessively small neighbors improves the quality of the posterior estimation.

For exhaustiveness, using the Ionosphere dataset, we also tried $k = 5, 7, 10, 15$ when using the kNN posterior estimation in the top and middle panels of Figure 3.1, and we summarized the results in Figure 3.2 (for top panel of Figure 3.1) and Figure 3.3 (for middle panel of Figure 3.1). Compared to the choice of k for $\mathcal{T}$, the choice of k for the near-boundary sample set $\mathcal{N}_B(\Lambda)$ clearly has a minor impact on the quality of the uncertainty measure computation.

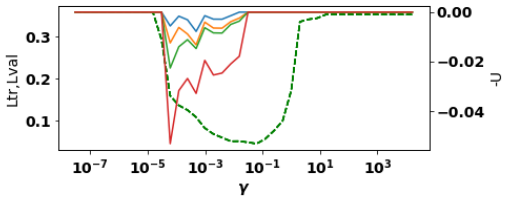

Figure 3.2: Effect of using different k in applying kNN to $\mathcal{T}$ for the Ionosphere dataset. L_{val} is shown by green dashed line. Curves for $k = 5, 7, 10, 15$ are shown in red, green, orange, and blue, respectively.

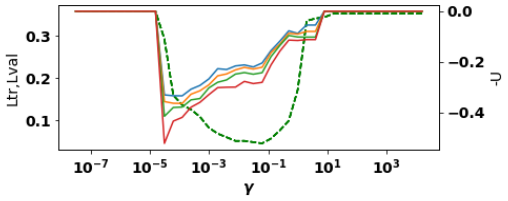

Figure 3.3: Effect of using different k in applying kNN to $\mathcal{N}_B(\Lambda)$ for the Ionosphere dataset. L_{val} is shown by green dashed line. Curves for $k = 5, 7, 10, 15$ are shown in red, green, orange, and blue, respectively.

To further analyze the influence of neighbor selection, focusing on one typical case of $k = 7$ and

on the two datasets of Breast Cancer and Ionosphere, we observed how far the near-boundary samples in $\mathcal{N}_B(\Lambda)$ and their neighbors selected from $\mathcal{T}$ are from $B(\Lambda)$, by plotting the histogram of their geometric distance to $B(\Lambda)$; the near-boundary samples in green and the neighbors in red (Figure 3.4). An important point here is that differently from the neighbor selection for our proposed method, which selects the neighbors (of the cluster centroids) only from $\mathcal{N}_B(\Lambda)$, we selected the neighbors (of the near-boundary samples) from the entire set of $\mathcal{T}$. In the figure, the value of 0 on the horizontal axis represents on-boundary samples on $B(\Lambda)$. As the figure shows, the near-boundary samples in $\mathcal{N}_B(\Lambda)$ are not quite on $B(\Lambda)$, but the neighbors that can include the samples outside $\mathcal{N}_B(\Lambda)$ are spreading further. This helps understand that the use of other samples than the near-boundary samples as the neighbors in the kNN can degrade the quality of the computation of posterior probability and uncertainty measure, proving the importance of restricting the neighbors to the near-boundary samples.

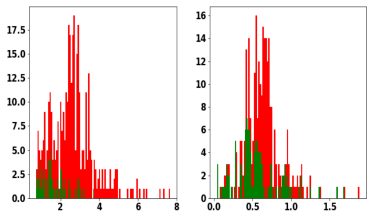

Figure 3.4: Geometric distance distribution for the samples used in the posterior probability estimation for Breast Cancer data (left) and Ionosphere data (right) ($\gamma = 2^{-5}$). The green histogram shows distance distribution for the near-bounday samples; the red histogram for the neighbors selected from the entire sample set.

3.7 Summary

In this chapter, we mathematically analyzed and proved the validity of our posterior probability estimation procedure, which plays a central role in the proposed method for finding the optimal classifier parameter status. Furthermore, this analysis of the posterior probability estimation along the estimated boundary may clarify guidelines on the selection and on the use of the boundary neighborhood to improve the accuracy of our classifier evaluation procedure.

Appendix 1

$p(\cdot)$ is continuous, therefore it has minimum value m and maximum value M within the bounded and closed set $\mathcal{R}^{(N)}(\boldsymbol{x})$:

$$m \leq p(\boldsymbol{u}) \leq M \qquad (\boldsymbol{u} \in \mathcal{R}^{(N)}(\boldsymbol{x})).$$

Then, integration over $\mathcal{R}^{(N)}(\boldsymbol{x})$ gives

$$m \cdot V^{(N)}(\boldsymbol{x}) \leq \int_{\mathcal{R}^{(N)}(\boldsymbol{x})} p(\boldsymbol{u}) d\boldsymbol{u} \leq M \cdot V^{(N)}(\boldsymbol{x}),$$

and thus

$$m \leq \frac{P^{(N)}(\boldsymbol{x})}{V^{(N)}(\boldsymbol{x})} \leq M.$$

Because m and M correspond to values taken by $p(\cdot)$ on $\mathcal{R}^{(N)}(\boldsymbol{x})$, the intermediate value theorem guarantees the existence of $\boldsymbol{x}^{(N)} \in \mathcal{R}^{(N)}(\boldsymbol{x})$, which satisfies

$$\frac{P^{(N)}(\boldsymbol{x})}{V^{(N)}(\boldsymbol{x})} = p(\boldsymbol{x}^{(N)}).$$

Appendix 2

We denote the probability space that governs the random variables by $(\Omega, \mathcal{F}, \Pr)$, where Ω is the space of all events, $\mathcal{F}$ is a completely additive class over Ω, and Pr is a probability measure over

$\mathcal{F}$. Then, the following holds:

$$
\begin{aligned}
P_j^{(N)}(\boldsymbol{x}) &= \mathrm{E}\Big[1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X}) \cdot 1_{\{j\}}(Y)\Big] \\
&= \int_{\Omega} 1_{\mathcal{R}^{(N)}(\boldsymbol{x})}(\boldsymbol{X}(\omega)) \cdot 1_{\{j\}}(Y(\omega)) \Pr(d\omega) \\
&= \int_{\Omega} 1_{\boldsymbol{X}^{-1}(\mathcal{R}^{(N)}(\boldsymbol{x})) \cap Y^{-1}(\{j\})}(\omega) \Pr(d\omega) \\
&= \Pr\Big(\boldsymbol{X}^{-1}(\mathcal{R}^{(N)}(\boldsymbol{x})) \cap Y^{-1}(\{j\})\Big) \\
&= \Pr\{\boldsymbol{X} \in \mathcal{R}^{(N)}(\boldsymbol{x}) \cap Y = j\} \\
&= \Pr\{Y = j\} \cdot \Pr\Big(\{\boldsymbol{X} \in \mathcal{R}^{(N)}(\boldsymbol{x})\} \Big| \{Y = j\}\Big) \\
&= \Pr(C_j) \int_{\mathcal{R}^{(N)}(\boldsymbol{x})} p(\boldsymbol{u}|C_j) d\boldsymbol{u},
\end{aligned}
$$

where for mapping $f : U \to V$ and set B ($\subset V$), $f^{-1}(B)$ represents the inverse image of B: $f^{-1}(B) = \{x \in U \mid f(x) \in B\}$, and also $\Pr(C_j) = \Pr\{Y = j\} = \Pr(Y^{-1}(\{j\}))$.

4 An Improved Boundary Uncertainty-Based Classifier Evaluation Method

We propose an improved method to perform classifier evaluation. In Chapter 2, we originally proposed a classifier evaluation method that introduced two novelties: the concept of classifier evaluation in terms of "boundary uncertainty" instead of the traditional classification error probability, and the possibility to accurately estimate the boundary uncertainty based on the same data that was used to train the classifier, in a single run. Our original method led to the successful evaluation and selection of the optimal classifier status for SVMs on several real-life datasets. However, our boundary uncertainty estimation also lacked accuracy on some datasets. Furthermore, it relied on several hyperparameters, heuristics, and random repetitions that were not computationally efficient, and that affected the accuracy of the estimation. These drawbacks clearly called for refinements that we introduce in this improved method. Results of classifier evaluation for SVM classifiers and Prototype-Based Classifiers (PBC) on more then thirteen real-life datasets and two synthetic datasets show the increased accuracy of our classifier selection.

4.1 Background for our improved boundary uncertainty estimation

For convenience, we call "Proposal 1" our original proposal described in Chapter 2, and "Proposal 2" the improved proposal.

4.1.1 Reminder: Goal

We assume the same notations as in Chapter 2, and remind that given a classifier parameter status Λ, our estimation goal is the boundary uncertainty $U(\Lambda)$:

$$U(\Lambda) = \sum_{\boldsymbol{x} \in B(\Lambda)} U(\boldsymbol{x}; \Lambda) p(\boldsymbol{x}), \tag{4.1}$$

where $U(\boldsymbol{x}; \Lambda)$ called uncertainty measure takes as input the two highest values of the class posterior probabilities $P(C_I|\boldsymbol{x}), P(C_{II}|\boldsymbol{x})$ at $\boldsymbol{x}$, and measures the degree of equality between $P(C_I|\boldsymbol{x})$ and $P(C_{II}|\boldsymbol{x})$.

We can see from Eq. (4.1) that the accurate estimation of $U(\Lambda)$ requires a sharp focus of the class posterior probability estimation on $B(\Lambda)$. More precisely this estimation should satisfy three requirements: (A) access to target samples located on the decision boundary; (B) sampling of these on-boundary samples from the probability density function $p(\boldsymbol{x})$; (C) at these on-boundary samples, focus of their class posterior probability estimation on the decision boundary, because Eq. (4.1) only considers on-boundary samples.

In both Proposal 1 and Proposal 2, we use the kNN class posterior probability estimation rule to estimate the class posterior probabilities (for convenience from now we shorten "k Nearest Neighbor class posterior probability estimation rule" to "kNN estimation"), because this method does not require to assume a probability model, but instead bases its estimation on the local information of class labels. The way we apply the kNN estimation to appropriately focus on the local information of the decision boundary plays a key role in the accuracy of the boundary uncertainty estimation, therefore we first detail the background of this focus before reminding the broader outline of Proposal 1, and how Proposal 1 handles the above requirements (A), (B), (C).

4.1.2 Towards Proposal 1

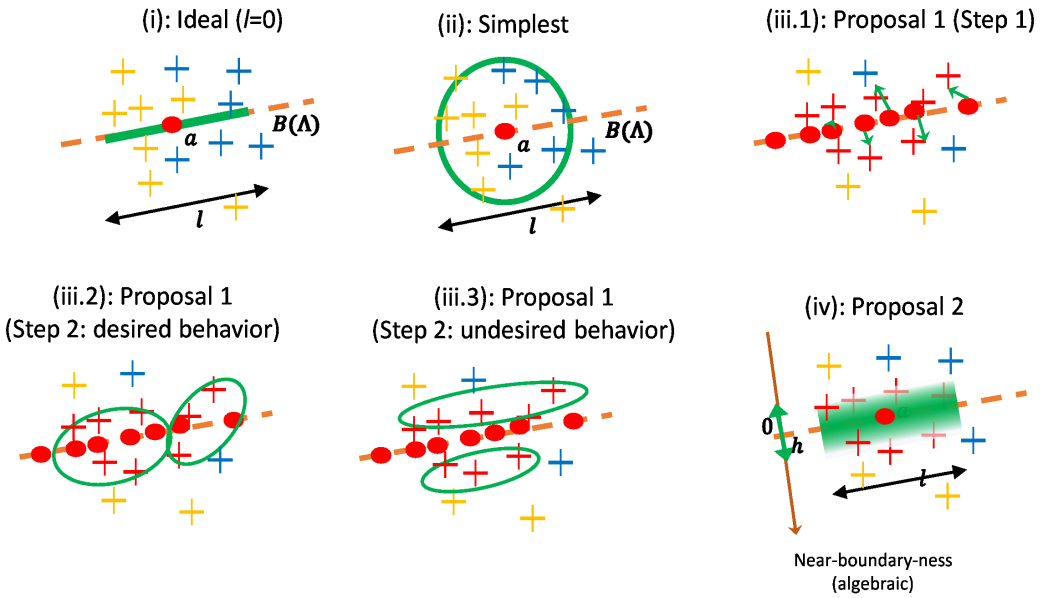

Figure 4.1: kNN estimation at the decision boundary performed in different ways, from (i) to (iv). (i): ideal estimation: at $\boldsymbol{a} \in B(\Lambda)$ (red dot), possibility to draw class labels an infinite number of times. (ii): basic kNN estimation at $\boldsymbol{a}$ in presence of finite data: use of the class labels surrounding $\boldsymbol{a}$ (green circle), regardless of their closeness to $B(\Lambda)$. (iii.1) to (iii.3): kNN estimation focused on $B(\Lambda)$ as done in Proposal 1. First, selection of the samples nearest to $B(\Lambda)$ (red crosses selected by on-boundary red dots). Second, clustering on these selected samples to obtain target volumes focused on $B(\Lambda)$ (green circles in (iii.2) and (iii.3)). (iv): kNN estimation centered on $B(\Lambda)$ as done in Proposal 2, detailed in the later sections.

In Figure 4.1, we illustrate several possible ways (i) to (iv) of applying the kNN estimation. To fix ideas, we consider an anchor $\boldsymbol{a} \in B(\Lambda)$ (red dot in Figure 4.1), and the goal is to estimate the (degree of equality between) class posterior probabilities at $\boldsymbol{a}$.

(i): The most ideal situation corresponds to the possibility to draw class labels from $\boldsymbol{a}$ an infinite number of times. Class posterior probabilities correspond to the expected frequencies of class labels. Actually, our estimation target is actually more $B(\Lambda)$ itself than only the point $\boldsymbol{a}$ itself. Therefore, drawing class labels from a volume included in $B(\Lambda)$ around $\boldsymbol{a}$ (instead of only from $\boldsymbol{a}$) would not incur a serious bias in terms of our estimation target (the boundary uncertainty) as was explained in Chapter 3. This quite ideal target volume is illustrated in (i) with a green

segment included in $B(\Lambda)$ of length l. When many samples are available, sampling from a green segment characterized by a smaller l to tend to the ideal case becomes possible.

(ii): The quite ideal situations described in (i) are not easy satisfy when only a finite amount of data is available. The most basic way of applying the kNN estimation is to use the class labels of the M nearest neighbors to $\boldsymbol{a}$ to approximate a repeated class label drawing at $\boldsymbol{a}$ ($M = 5$ in the illustration). This results in considering class labels in a spherical volume around $\boldsymbol{a}$ (green circle in (ii)). However such spherical volume does not specifically focus on the class labels along the decision boundary $B(\Lambda)$.

(iii): To focus on $B(\Lambda)$, in Proposal 1 proposed to restrict the class labels used in the kNN estimation to the samples that are "closest" to $B(\Lambda)$, and the set of near-boundary samples were denoted $\mathcal{N}_B(\Lambda)$. $\mathcal{N}_B(\Lambda)$ was obtained by generating on-boundary anchors (red dots in (iii.1)), and then selecting the set of their one-nearest neighbors as $\mathcal{N}_B(\Lambda)$ (selection represented by green arrow in (iii,1)). Then, Proposal 1 used a clustering procedure to locally break down $\mathcal{N}_B(\Lambda)$ into smaller target volumes that were then used for the kNN estimation (green ovals in (iii.2) and (iii.3)).

If the clustering went as desired, the obtained target volumes would center on the decision boundary as in (iii.2), which brought us closer to the quite ideal green segment-like shape in (i). However the undesirable case (iii.3) was also possible. In (iii.3), the target volumes obtained by the clustering are completely included in one side or the other of the decision boundary, which leads to a biased class posterior probability estimation, "biased" in the sense that it is off-centered from the decision boundary that is central to our purpose.

(iv): This corresponds to our current Proposal 2, and we will describe it later.

4.1.3 Reminder: Outline of Proposal 1

With the above principles for our decision boundary-focused kNN estimation in mind, we now remind the entire outline of Proposal 1 in Figure 4.2.

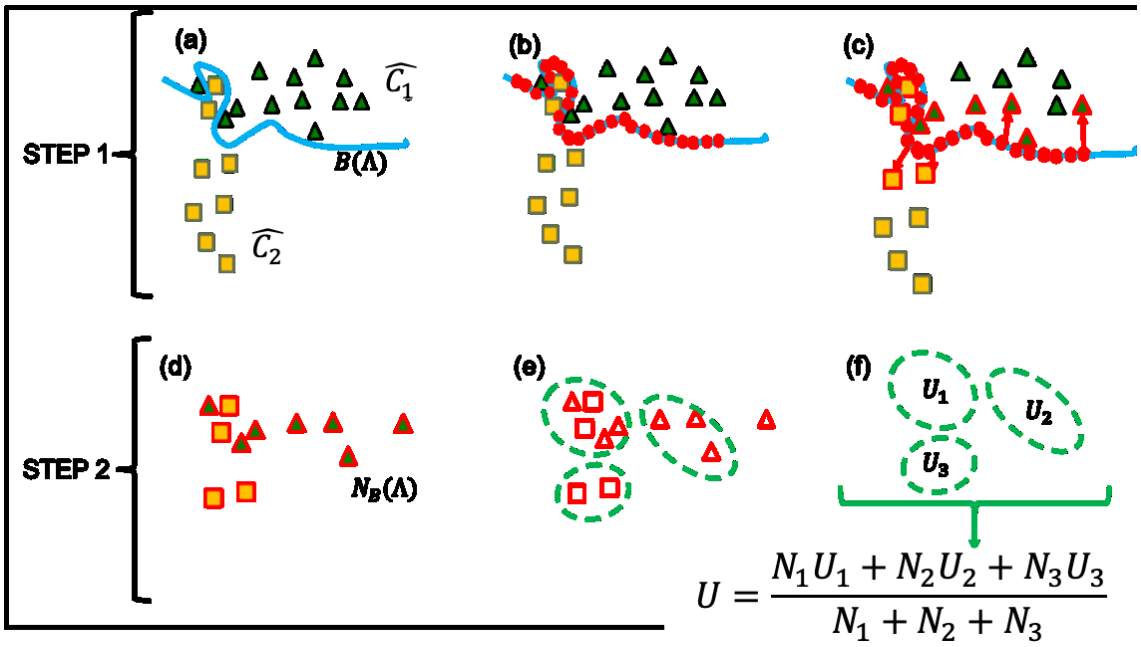

Figure 4.2: Graphical explanation of our original two-step classifier evaluation method on two-dimensional data. (a): Decision boundary $B(\Lambda)$ represented in blue. (b): Step 1 arbitrarily generates anchors (red dots) on $B(\Lambda)$. (c): Step 1 selects the nearest neighbor for each anchor. This results in a set of near-boundary samples denoted $\mathcal{N}_B(\Lambda)$. (d): From now, the method only considers $\mathcal{N}_B(\Lambda)$ to focus on the decision boundary. (e): Step 2 uses a partitioning method to break down $\mathcal{N}_B(\Lambda)$ into clusters, represented in green dashed ovals. (f): After estimating the class posterior probabilities in each cluster by application of the kNearest Neighbor class posterior probability estimation rule, Step 2 computes the boundary uncertainty $U(\Lambda)$. Step 2 is repeated R times, and the final estimate of $U(\Lambda)$ is the average of the results over the R runs.

Proposal 1 handled each of the three accuracy requirements (A), (B), (C) described in Section 4.1.1 as follows (the handling of the requirements does not necessarily follow the flow of Proposal 1). (A): specification of near-boundary target volumes for the kNN estimation (green dashed ovals in Figure 4.2). The volume centroids may be seen de facto as near-boundary target samples for the kNN estimation; (B): implicit adaptation of these target volumes to the density $p(\boldsymbol{x})$, because a clustering procedure is used to form the target volumes, and clustering procedures tend to adapt to the density ((e) in Figure 4.2); (C): restriction of these volumes to near-boundary samples ((d) in Figure 4.2), so that the kNN estimation focuses on the decision boundary.

4.1.4 From Proposal 1 to Proposal 2

However, the handling of (A), (B), (C) by Proposal 1 suffered several drawbacks. Although these drawbacks overlap, and sometimes result from the same item in Proposal 1, for clarity we separate them into three categories: lack of accuracy of the boundary uncertainty estimation, unclearness and use of heuristic settings, and memory and time costs. We enlisted all the drawbacks in each category for exhaustivity, however the understanding of some drawbacks in the two last categories may require to refer to Chapter 2. Because these drawbacks tend to be less essential to estimation accuracy, they may be skipped without hindering the understanding of Proposal 2.

Lack of accuracy of the boundary uncertainty estimation. The following characteristics of the estimation in Proposal 1 fundamentally hindered its accuracy:

- Discrete definition of "near-boundary-ness". In Proposal 1, a sample is considered near or not to the decision boundary. Not only is a binary "threshold" for near-boundary-ness difficult to define, but slight changes in the threshold definition result in accepting as near-boundary samples either too many samples that are far from the decision boundary, or to too few samples to reliably estimate the boundary uncertainty.
- Too rough *k*NN estimation that assigns the same weight to all the selected near-boundary samples regardless of their relative closeness to the decision boundary.
- Possible bias in the class posterior probability estimation as illustrated in (iii.3) of Figure 4.1.

Unclearness and use of heuristic settings. Several treatments in Proposal 1 heavily relied on randomness, repetitions, and heuristic settings that increased the computations. Not only did this pose an obstacle to clearly assess the time costs of our method, but this also seriously hindered the scalability of our method:

- No rationale to control the number of anchors to generate, hence the necessity to generate

some high number of anchors by default.

- No rationale to choose the pairs of training samples that anchors are generated from, hence the necessity to randomly pick the pairs by default.
- No direct control on the formation of target volumes. This formation depends on repeated random initializations in the clustering procedure and heuristic settings to indirectly control the size of the target volumes.
- No clear relationship guaranteed between $p(\boldsymbol{x})$ and the target volumes (requirement (B)).

Memory and time costs. The following treatments were time-consuming, and were an obstacle to scalability:

- Expensive class-by-class storage and treatment of random pairs as described in Section 2.3.4.
- Tedious case-by-case checks when generating anchors as described in Algorithm 4.
- Nearest neighbor search *for every newly generated anchor*. This cost is further aggravated by the absence of a rationale to set the necessary number of anchors to generate.
- Necessity to perform repeat the tree clustering R times in Step 2 because of the random initializations in the clustering procedure.

While preserving the overall two-step structure of Proposal 1, Proposal 2 improves the two steps to better handle requirements (A), (B), (C) as follows (the handling of the requirements does not necessarily follow the flow of Proposal 2). Requirement (B): the improved Step 1 also generates anchors to select near-boundary samples, however in contrast to the arbitrary generation of anchors in Proposal 1, the number and the location of the generated anchors is carefully prepared to emulate sampling from $p(\boldsymbol{x})$ as indicated in Eq. (4.1). These carefully sampled anchors are then directly used as target samples for the kNN estimation. Requirement (C): the kNN estimation in the improved Step 2 implicitly weighs samples based on their closeness to the decision boundary, and samples closed to the decision boundary are given a higher weight. This avoids the difficult dilemma in Proposal 1 between a too strict and a too loose definition of near-boundary samples.

Besides the simplicity, this weighting decreases the variance by potentially using more samples for the estimation, while not decreasing the bias by weighting down samples further away from $B(\Lambda)$. Requirement (A): although we detail it in Section 4.5, the kNN estimation in Proposal 2 is less biased in the sense that it is centered *exactly* on $B(\Lambda)$.

Besides these improvements in terms of accuracy, Proposal 2 is deterministic, more scalable, and free from repetitions and unclearness as we will describe in the next sections.

4.2 Outline of Proposal 2

Proposal 2 takes the same inputs and output as Proposal 1: its purpose is to choose the most optimal classifier status among a set Λ_{TR} of trained classifier parameters. We provide the outline of Proposal 2 in Algorithm 6, where the differences with Proposal 1 are emphasized in italics. Details of this implementation are described in Sections 4.4.1, 4.4.2, and 4.5.

Algorithm 6: Outline of the improved procedure for classifier selection

Input: Set Λ_{TR} of trained classifier parameters, $M \leftarrow 40$
Output: $\arg\max_{\Lambda \in \Lambda_{TR}} U(\Lambda)$
for $\Lambda \in \Lambda_{TR}$ **do**
    `/* Step 1: Selection of near-boundary samples */`
    Appropriately generate anchors on $B(\Lambda)$ (Section 4.4);
    `/* Step 2: Computation of the boundary uncertainty */`
    For each anchor, apply a *weighted* kNN estimation to the cluster formed by the anchor itself and its M-nearest neighbors (Section 4.5);
    Compute $U(\Lambda)$ (Eq. (4.1));
end
return $\arg\max_{\Lambda \in \Lambda_{TR}} U(\Lambda)$

4.3 Continuous measure of near-boundary-ness

Before entering the description of the improved Step 1 and Step 2, we describe a new backbone component of Proposal 2, that is, the definition of a smooth measure of near-boundary-ness (here, "smooth", or "continuous", is opposed to "binary"). This measure is used both in the improved Step 1 to appropriately generate anchors, and in the improved Step 2 to weigh samples and focus exactly on $B(\Lambda)$ during the kNN estimation.

The Euclidean distance between a sample and the decision boundary $B(\Lambda)$, or some other traditional distance (e.g., Manhattan distance) may be an intuitive measure of near-boundary-ness. However, such distance may not be the most appropriate measure of near-boundary-ness, as we describe in the following qualitative analysis.

The decision boundary is defined by equality between the two locally highest discriminant functions, that we denote $g_{D_1(\cdot)}$ and $g_{D_2(\cdot)}$. If the values of $g_{D_1(\cdot)}$ and $g_{D_2(\cdot)}$ change a lot around $B(\Lambda)$, then samples should be chosen very "close to" $B(\Lambda)$ for $g_{D_1(\cdot)}$ and $g_{D_2(\cdot)}$ to be close to equality. Conversely, if $g_{D_1(\cdot)}$ and $g_{D_2(\cdot)}$ are almost uniform around $B(\Lambda)$, then samples can be chosen quite "far from" $B(\Lambda)$, while still almost satisfying the equality between $g_{D_1(\cdot)}$ and $g_{D_2(\cdot)}$. Here, "close to" and "far from" were meant in terms of some traditional distance such as the Euclidian distance. Based on this qualitative analysis, near-boundary-ness of a sample should be directly defined based on some measure of difference between $g_{D_1(\cdot)}$ and $g_{D_2(\cdot)}$, rather than based on some traditional distance that ignores the definition of $B(\Lambda)$. Incidentally, the computation of such difference is easy for any classifier model.

To preserve the information of which side (correct or incorrect classification) of the boundary the samples are found, by convention we impose the value of the near-boundary-ness to be negative in the case of a correctly classified sample $\boldsymbol{x}$ ($\epsilon(\boldsymbol{x}) = -1$), and negative otherwise ($\epsilon(\boldsymbol{x}) = 1$). We therefore define the measure of near-boundary-ness as $nb(\boldsymbol{x}; \Lambda) = \epsilon(\boldsymbol{x})(g_{D_1(\boldsymbol{x})}(\boldsymbol{x}; \Lambda) - g_{D_2(\boldsymbol{x})}(\boldsymbol{x}; \Lambda))$, that we shorten to $nb(\boldsymbol{x})$ for simplicity.

4.4 Improved Step 1

Ideally, the training set $\mathcal{T}$ would contain on-boundary samples sampled from $p(\cdot)$ so that we can apply Eq. (4.1), however there are usually no such samples. This improved Step 1 aims at generating such samples. To fix ideas, we consider $\boldsymbol{x} \in B(\Lambda)$. Then $p(\boldsymbol{x})$ can be decomposed as:

$$p(\boldsymbol{x}) = \sum_{j=1}^{J} P(C_j)p(\boldsymbol{x}|C_j), \tag{4.2}$$

which we interpret as follows: whenever we draw a sample $\boldsymbol{x}$ on $B(\Lambda)$ according to $p(\cdot)$, it has the probability $p(\boldsymbol{x}, C_j) = P(C_j)p(\boldsymbol{x}|C_j)$ to be labeled (exist) with C_j. The value of $p(\boldsymbol{x}, C_j)$ is determined by two components: the class prior probability $P(C_j)$ and the class likelihood $p(\boldsymbol{x}|C_j)$. $P(C_j)$ globally constraints the number of samples of each class in the dataset. Then, for each class, $p(\cdot|C_j)$ locally constraints how to distribute the samples of this class accross the data space.

The particularity is that we want to apply these sampling constraints *just on the decision boundary* $B(\Lambda)$. For each class C_j, the improved Step 1 emulates these two constraints to generate anchors as follows:

I. Emulation of the class prior probability constraint on $B(\Lambda)$: determine the number N_a^j of anchors that should be generated on $B(\Lambda)$ (Section 4.4.1).

II. Emulation of the class likelihood constraint on $B(\Lambda)$: generate N_a^j on-boundary samples that stay as close as possible to the class likelihood distributions observed in the training set $\mathcal{T}$ (Section 4.4.2).

4.4.1 Class-by-class determination of the number of anchors to generate

Given a class C_j, the total number of samples in $\mathcal{T}$ of C_j carries the constraint defined by $P(C_j)$. Therefore, to reflect the constraint defined by $P(C_j)$ on $B(\Lambda)$, we propose to determine the number of samples in $\mathcal{T}$ of C_j that fall (within a small region centered) on $B(\Lambda)$. This number

provides an estimate of N_a^j.

Concretely, we proceed as follows. We first represent the samples of C_j solely by their measure of near-boundary-ness. In this new one-dimensional representation, $B(\Lambda)$ is identified by the value zero. By applying a histogram on the distribution of these newly represented samples, the desired N_a^j corresponds to the value of the histogram at zero. If we denote H the automatically determined bin width, then this value at zero corresponds in the original data space to the total number of samples of C_j that fall in a region centered on $B(\Lambda)$ with a width H in the direction orthogonal to $B(\Lambda)$. Before further explanations, we first illustrate this estimation in Figure 4.3.

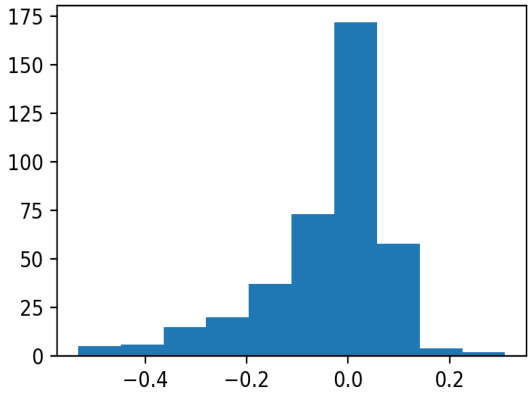

Figure 4.3: Determination of N_a^0 on synthetic data for some trained SVM classifier. The originally multi-dimension vector samples of C_0 are solely represented by their measure of near-boundary-ness (horizontal axis). To estimate N_a^0, we read the value at zero of the histogram on this one-dimensional data. In this example, the histogram indicates that $N_a^0 = 175$ anchors should be generated for C_0 to respect $P(C_0)$.

4.4.2 Class-by-class generation of anchors

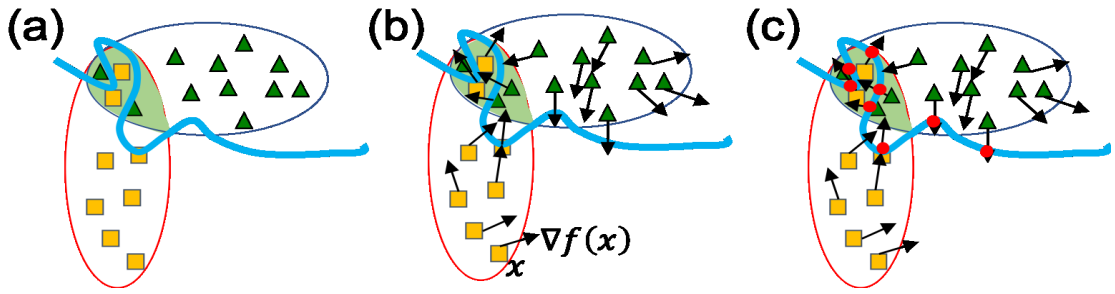

Figure 4.4: Illustration of anchor generation in Proposal 2 for two-class data. (a): Estimated labels for class 1 ($\widehat{C}_1$) and class 2 ($\widehat{C}_2$) are shown in yellow squares and green triangles, respectively. (b): for each training sample, $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ is represented in a black arrow, and corresponds to the locally most efficient direction to search for $B(\Lambda)$. (c): For each training sample $\boldsymbol{x}$, an on-boundary anchor (red dot) is tentatively generated along the direction provided by $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ within a distance d from $\boldsymbol{x}$.

Given a class C_j, the goal of this section is to generate N_a^j on-boundary samples (anchors) while staying as close as possible to the class likelihood distribution $p(\boldsymbol{x}|C_j)$. The training samples of C_j naturally follow $p(\boldsymbol{x}|C_j)$, therefore we propose to generate anchors that are as close as possible to the distribution of training samples.

Illustrations of our proposed anchor generation are provided in Figure 4.4. We proceed by performing parsimonious searches simultaneously around all the training samples: for each training sample $\boldsymbol{x}$, search for the decision boundary in a small neighborhood around $\boldsymbol{x}$, and if the decision boundary can be found in this neighborhood, then generate an anchor on it. The size of the neighborhood is set to be the same for all the training samples. A big size implies that we can produce an anchor from almost every training sample. A small size implies that we can only produce an anchor from training samples that are very close to $B(\Lambda)$. By adjusting the size of the neighborhoods, we can generate exactly N_a^j anchors as desired, and the obtained anchors are as close as possible to $p(\boldsymbol{x}|C_j)$.

We now describe how to search in the neighborhood around each training sample, and how to adjust the size of the neighborhoods. To fix ideas, we consider a training sample $\boldsymbol{x}$, and

we remind that $B(\Lambda)$ is defined by: $B(\Lambda) = \{\boldsymbol{x} \in \mathcal{X}, g_{D_2}(\boldsymbol{x}; \Lambda) - g_{D_1}(\boldsymbol{x}; \Lambda) = 0\}$. We denote $f(\boldsymbol{x}; \Lambda) = g_{D_2}(\boldsymbol{x}; \Lambda) - g_{D_1}(\boldsymbol{x}; \Lambda)$. To simplify notations, we shorten to $f(\boldsymbol{x})$.

To generate an anchor close to $\boldsymbol{x}$, we want to find in a small neighborhood of $\boldsymbol{x}$ a sample $\boldsymbol{x}'$ that satisfies $\widehat{C}(\boldsymbol{x}) \neq \widehat{C}(\boldsymbol{x}')$. If we can find such $\boldsymbol{x}'$, then the existence of an anchor (zero of $f(\cdot)$) on the segment $[\boldsymbol{x}; \boldsymbol{x}']$ is guaranteed by the theorem of intermediate values applied to $f(\cdot)$.

By construction, $\nabla_x f(\boldsymbol{x})$ locally points to the direction where the difference $g_{D_2}(\cdot; \Lambda) - g_{D_1}(\cdot; \Lambda)$ decreases in absolute value. Therefore, in order to search for $B(\Lambda)$ in the neighborhood of $\boldsymbol{x}$, the most efficient direction to explore is given by $\nabla_x f(\boldsymbol{x})$. We can use this property to search for anchors by progressively shifting away from $\boldsymbol{x}$ along $\nabla_x f(\boldsymbol{x})$.

Denote $u(\boldsymbol{x}) = \frac{\nabla_x f(\boldsymbol{x})}{\|\nabla_x f(\boldsymbol{x})\|}$ the unit vector carried by $\nabla_x f(\boldsymbol{x})$. Given a small distance d to determine, define:

$$\boldsymbol{x}_b(d; \boldsymbol{x}) = \boldsymbol{x} + du(\boldsymbol{x}). \tag{4.3}$$

Here, d corresponds to the size of the search neighborhoods that we described above. In practice, we can search for the desired value of d that produces N_a^y anchors by dichotomy: first define a wide range $[d_m; d_M]$, and then consider the middle d_0 of the segment $[d_m; d_M]$. If d_0 generates a number of anchors higher than N_a^y, then reduce the search to $[d_m; d_0]$, else reduce the search $[d_0; d_M]$. Repeat this segment reduction until we obtain N_a^j anchors, or until we obtain a number of anchors close enough to N_a^j, or until a maximum number i_M of dichotomy iterations was reached.

4.5 Improved Step 2

Given an anchor $\boldsymbol{a} \in B(\Lambda)$ appropriately sampled from $p(\cdot)$ produced by Step 1, consider the cluster formed by the M nearest neighbors $NN(M, \boldsymbol{a})$ of $\boldsymbol{a}$, and resume the notation $i = D_1(\boldsymbol{a})$, $j = D_2(\boldsymbol{a})$ for the indexes of the two most represented classes in $NN(M, \boldsymbol{a})$. The main idea in this

improved Step 1 is to perform a *smooth* count of the class labels of $NN(M, \boldsymbol{a})$ *exactly* on $B(\Lambda)$ as was illustrated in (iv) of Figure 4.1. This smoother and appropriately centered estimation relies on two components:

I. Representation of the M neighbors of $\boldsymbol{a}$ by their degree of near-boundary-ness. Again, the decision boundary $B(\Lambda)$ in this new one-dimensional representation is identified by the value zero.

II. Smooth count of these M labels for each class C_i and C_j on the zero of this one-dimensional represented data, that we denote k_i and k_j, respectively. The kNN estimation gives $P(C_i|\boldsymbol{a}) = \frac{k_i}{k_i+kj}$ and $P(C_j|\boldsymbol{a}) = \frac{k_j}{k_i+k_j}$, respectively.

We illustrate this estimation in Figure 4.5.

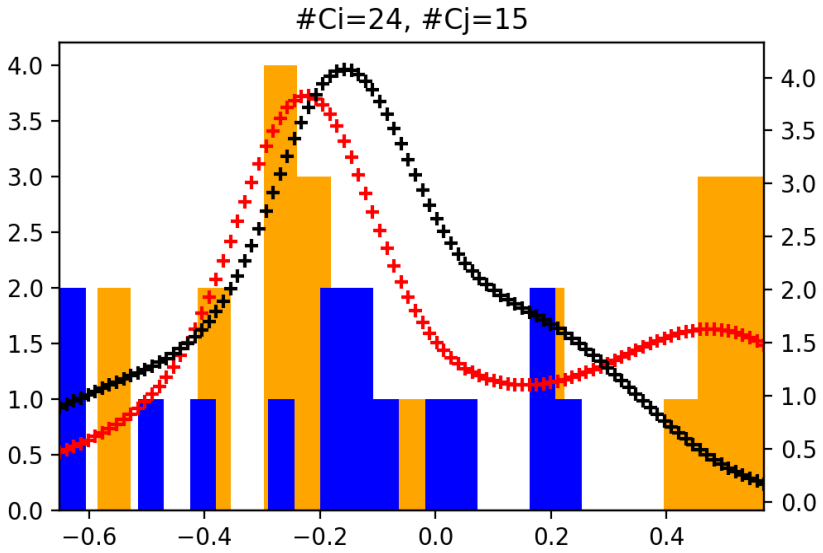

Figure 4.5: The horizontal axis corresponds to the degree of near-boundary-ness for samples of class C_i, and the opposite of the degree of near-boundary-ness for samples of class C_j. On this axis, the anchor $\boldsymbol{a}$ (value zero) and its $M = 39$ nearest neighbors are represented. Among $NN(M, \boldsymbol{a})$, 24 neighbors belong to C_i (orange), and 15 neighbors belong to C_j (blue). We perform a smooth histogram for each class in red and black, respectively. We use these two histograms to obtain a smooth count k_i and k_j of C_i and C_j on $B(\Lambda)$. Here $k_i = 1.5$ and $k_j = 1.9$.

To perform the smooth histogram described in Figure 4.5 with a Parzen estimation [30]. As a

reminder, given $h > 0$, a kernel function $\phi(\cdot)$, N_V training samples $\boldsymbol{x}_1, ..., \boldsymbol{x}_{N_v}$, and a target sample $\boldsymbol{x}$, a Parzen estimator counts the number of samples $count(\boldsymbol{a}; h)$ that falls within a region centered on $\boldsymbol{x}$ whose volume is controlled by h, and whose shape is controlled by $\phi(\cdot)$:

$$count(\boldsymbol{x}; h) = \sum_{m=1}^{k} \phi\Big(\frac{\boldsymbol{x} - \boldsymbol{x}_m}{h}\Big). \tag{4.4}$$

For example, a common choice for the kernel function is the Gaussian kernel. In our case, given a target anchor $\boldsymbol{a} \in B(\Lambda)$ and $\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_M$ its M nearest neighbors, the target sample for the Parzen estimation is the value $nb(\boldsymbol{a}) = 0$ of the near-boundary-ness of a target anchor $\boldsymbol{a}$, and the training samples for the Parzen estimation are the values $nb(\boldsymbol{a}_1), nb(\boldsymbol{a}_2), ..., nb(\boldsymbol{a}_{N_v})$ of the near-boundary-ness of the M nearest neighbors. Implicitly, when counting on the zero, data points are weighted based on their distance to the zero, i.e. based on their near-boundary-ness, and the weight for each data points is $\phi\Big(\frac{nb(\boldsymbol{a}) - nb(\boldsymbol{a}_m)}{h}\Big)$. When using a Gaussian kernel, we can see that this weight decreases exponentially fast as data points get further away from $B(\Lambda)$.

The crucial issue that determines the quality of the count (and by extension, of the class posterior probability estimation and therefore of our boundary uncertainty estimation) is the value of the kernel width h. A too small value of h leads to an overfit for the boundary uncertainty estimation, while a too large value smoothes the data too much and also leads to an underfit for the boundary uncertainty estimation.

[31] provides a setting of h based on the analytic estimation of the expected Mean Squared Error, however it seems that the method does not always perform accurately depending on the data distribution. Therefore, as another possibility, we use a setting of h that estimates the expected likelihood based on a more CV-like data-driven way, and for this simple one-dimensional task the CV-like mechanism can be directly incorporated in the objective function. Although details about this expected likelihood-based setting are provided in Section 4.7, through this example we can see that by sharply focusing our class posterior probability estimation on the decision boundary, we reduced the estimation problem to a simple task (nonparametric local count on one-dimensional data) that can be carried out accurately and efficiently.

Finally, by denoting $A(\Lambda)$ the set of anchors generated by the improved Step 1, and S the number of generated anchors, we estimate the boundary uncertainty with:

$$U(\Lambda) = \frac{1}{S} \sum_{\boldsymbol{a} \in A(\Lambda)} U(\boldsymbol{x}; \Lambda), \tag{4.5}$$

Interpretation in the data space. This one-dimensional count is equivalent in the data space to a count of class labels in a region centered on $B(\Lambda)$, and of width h in the direction orthogonal to $B(\Lambda)$.

4.6 Benefits in terms of memory and time

One of the main motivations underlying the proposal in this chapter was to improve the scalability of our boundary uncertainty-based classifier evaluation. In this section, we enlist the time and memory savings compared to Proposal 1. This Section may require to refer to Chapter 2, and may be skipped if necessary because it is less essential to the accuracy of the boundary uncertainty estimation.

No expensive storage of sample pairs. To search for anchors, Section 2.3.4 covered the necessity to store lists of random pairs of training samples in a matrix A of size J^2. This can quickly become expensive for larger tasks. In Proposal 2, the anchor generation in Section 4.4.2 starts the search for anchors from training samples considered individually, which removes the need for A.

No case-by-case treatments during the anchor generation. The case-by-case treatment in Algorithm 4 was necessary because training samples in each random pair were potentially far from each other. In Proposal 2, the anchor generation in Section 4.4.2 does not consider such far pairs, but instead considers pairs of training samples $\boldsymbol{x}$ and generated samples $\boldsymbol{x}'$ located at

a distance d, where d is so small as it could be. This parsimonious distance between pairs skips the need for the above case-by-case treatment.

No repetitive nearest-neighbor searches. In Algorithm 6, we can avoid searching the k nearest neighbors for each newly generated anchor. Indeed, Section 4.4.2 generates anchors that are (very) close to the training samples that they were generated from. Assuming an anchor $\boldsymbol{a}$ was generated from the neighborhood of $\boldsymbol{x} \in \mathcal{T}$, we can reasonably approximate $NN(\boldsymbol{a}, M)$ with $NN(\boldsymbol{x}, M)$. We can therefore preliminary compute and store $NN(\mathcal{T}, k)$, and then apply the above approximation. As a reminder, the time complexity of searching for the nearest neighbor in a dataset of size N using a KD tree varies from $O(\log(N))$ to $O(N)$ depending on the dimensionality D of the task.

Easily parallelized procedures. In Step 1, the generation of anchors in batches can easily be vectorized. In Step 2, the estimation of $\boldsymbol{h}$ and the smooth count in the Parzen estimation (Eq. (4.4)) can be efficiently implemented on parallel hardware.

4.6.1 Experiments

The experimental setting in this section is the basically the same as in Section 2.4, and we re-use notations, datasets, and layout of the graphs from Section 2.4. In this section, we therefore focus our description on the differences with Section 2.4.

4.6.2 Classifiers

We test our improved boundary uncertainty-based classifier evaluation on a Gaussian kernel SVM classifier and on a Prototype-Based Classifier (PBC). For the SVM, our goal is to select the optimal value of the Gaussian kernel width γ as was done in Chapter 2, although this time we focus our SVM experiments on two-class datasets. For the PBC, prototypes are applying

the $k$means clustering to each class, and by using the resulting class centroids as prototypes $\{\boldsymbol{p}_{jm}\}_{m\in[1,k]}$ that represent each class j. We constraint k to be the same for all classes, and our goal is to select is the optimal number of prototypes per class k. More details about PBCs can be found in [32].

4.6.3 Datasets

On top of the datasets used in Chapter 2, we also consider the Avila and the Thyroid datasets from the UCI Machine Learning Repository. For the evaluation of two-class SVMs, we prepared two more two-class datasets: Abalone_01 corresponds to the classes 0 and 1 from the Abalone dataset, Landsat Satellite_47 corresponds to the classes 4 and 7 from the Landsat Satellite dataset.

Table 4.1: Datasets

Dataset	N	D	J	Remarks
GMM	2,000	2	2	synthetic data
GMM_5c	1000	2	5	synthetic data
Abalone	4,177	7	3	custom version, high class overlap
Abalone_01	4,177	7	3	custom version
Avila	20,772	10	10	access to test set (20,772 samples)
Breast Cancer	683	9	2	2:1 imbalance
Cardiotocography	1,831	30	2	10:1 imbalance
Ionosphere	351	34	2	2:1 imbalance, few data and quite high dimension
Letter Recognition	20,000	16	26	easy
MNIST (test)	10,000	784	10	easy but many (correlated) dimension
Landsat Satellite_47	2,134	36	2	custom version
Landsat Satellite	6,435	36	7	
Spambase	4,601	57	2	many zeros in the feature values
Thyroid	7,200	21	3	18:1:1 imbalance
Wine Quality Red	1599	11	3	custom version, difficult
Wine Quality White	1470	11	3	custom version, difficult

4.6.4 Hyperparameters

The main hyperparameter in this procedure is the size M of the cluster around each anchor $\boldsymbol{a}$ in Step 2. M should be small to focus on the local information around $\boldsymbol{a}$, however it should be high enough to perform a meaningful Parzen estimation (Algorithm 7). A rule of thumb, 30 or 40 samples seem to be a minimum estimate a Gaussian in one dimension, so we set $M = 40$ for all the datasets in our experiments.

In Section 4.4.2, setting d_m and d_M was necessary. Although a magnitude of d_m and d_M might be deduced from the bin width H described in Section 4.4.1, this bin width cannot be used directly, because contrary to d_m and d_M, H is not homogeneous to a Euclidian distance. Therefore for now we simply set safely extreme values for $d_m = 2^{-10}$ and $d_M = 2^5$, a maximum number of iterations for the dichotomy $i_M = 20$, and a tolerance $N_a^y \pm 10$ anchors.

These other hyperparameters correspond to the maximum number of iterations in some iterative procedure, and they should simply be set to a high value. From our experience with these procedures, for all datasets, we set the maximum number of weights re-estimation $k_M = 3$ in Algorithm 7, and the maximum number of iterations $l_M = 10$ in the CVML estimation of h in Algorithm 8.

The histogram counts in the improved Step 1 determine the desirable number of anchors, which may not require so much accuracy as the counts used in Step 2 for the kNN rule. Therefore for the improved Step 1, we used traditional histograms instead of smoothed histograms. Derivation of the optimal width was done using [1], which itself seems to rely on [33, 34].

Choice of the uncertainty measure Chapter 2 used the binary Shannon entropy as an uncertainty measure, however the bell-like shape of this function does not almost penalize non-uncertainty in the range $[0.4, 0.6]$, and then strongly penalizes non-uncertainty outside this range. To achieve a more stable measurement of uncertainty, in this experiment we use the Gini impurity, because its simple triangle shape does not exhibit the extreme behaviors that result from the

[1] https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.histogram_bin_edges.html

curvature of the binary Shannon entropy.

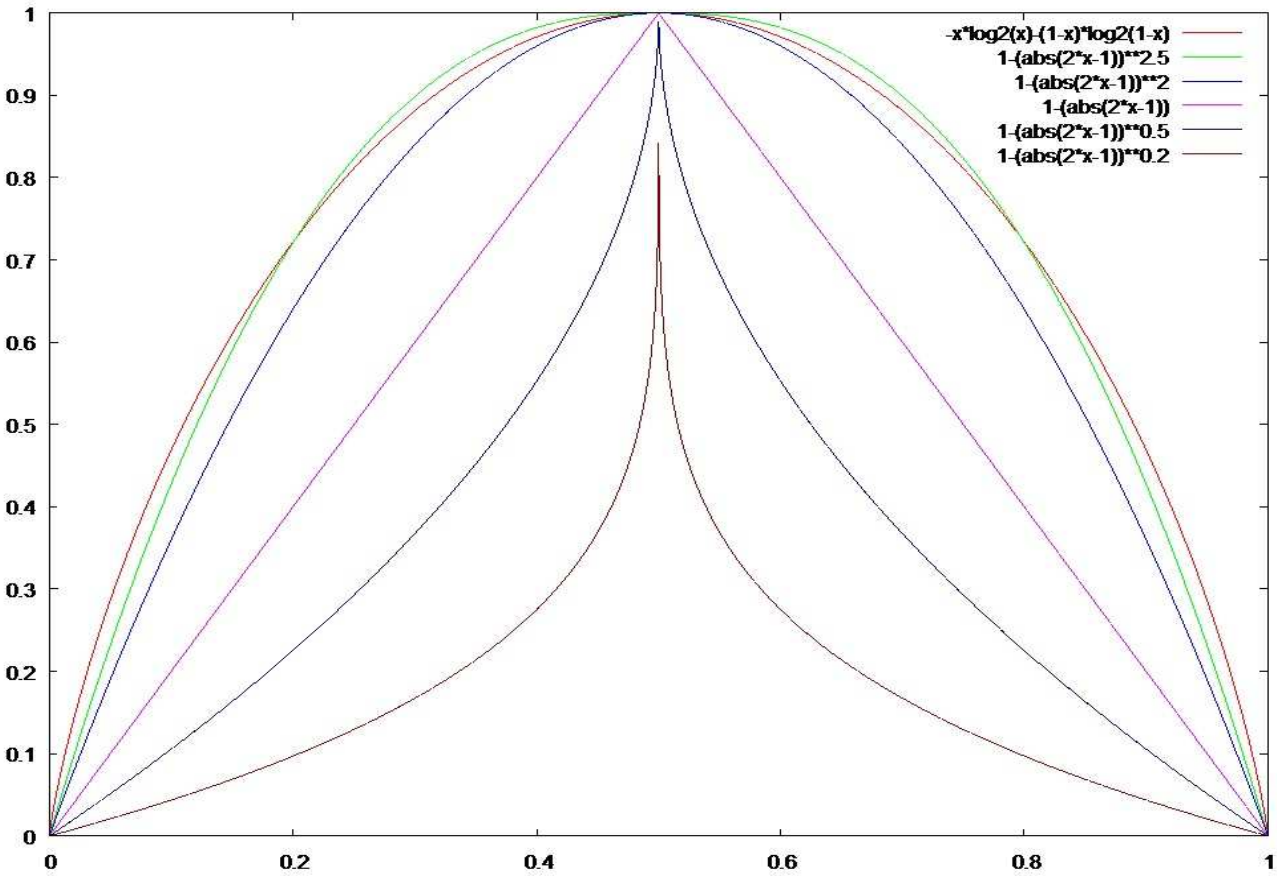

Figure 4.6: Different possible function candidates to the uncertainty measure. The binary Shannon entropy and the Gini impurity are represented in brown and pink, respectively.

4.6.5 Treatment of the class imbalance in datasets

In chapter 2, to address the class imbalance in datasets, we modified the class posterior probabilities output by the *k*NN estimation (we divided them by their corresponding class prior probabilities). Although this modification seemed to have a positive effect shown in Section 2.4.7, unequal prior probabilities are fundamentally handled in the *k*NN class posterior probability estimation through its application of the Bayes theorem. Therefore, our ad hoc modification was not the correct solution to address the potential degradation of our boundary uncertainty estimation

in the case of imbalanced datasets. For this reason, we do no longer apply our modidication of estimated class posterior probabilities in our new experiments.

4.6.6 Effect of the dimensionality on Proposal 1

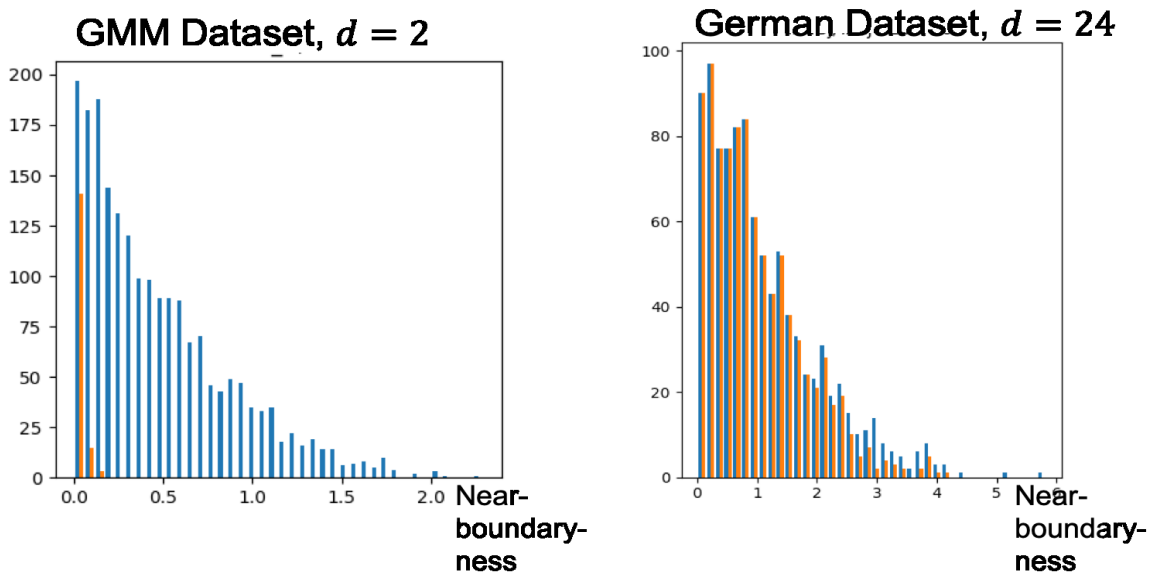

Figure 4.7: Histograms of the absolute value of the near-boundary-ness of the training samples $\mathcal{T}$ (on-boundary-ness corresponds to the value 0). Blue: histogram for all $\mathcal{T}$. Orange: histogram for the selected near-boundary samples $\mathcal{N}_B(\Lambda)$.

The results in Figure 4.7 clearly show that in higher dimensions, all training samples tend to be close to $B(\Lambda)$ when adopting the definition of near-boundary-ness of Proposal 1. It seems that in higher dimensions, for almost any training sample, it is possible to find a close on-boundary anchor. This may be a consequence of the curse of dimensionality. Figure 4.7 shows that our binary definition of near-boundary-ness was too rough, and that a more continuous definition was necessary.

4.6.7 Classifier evaluation results

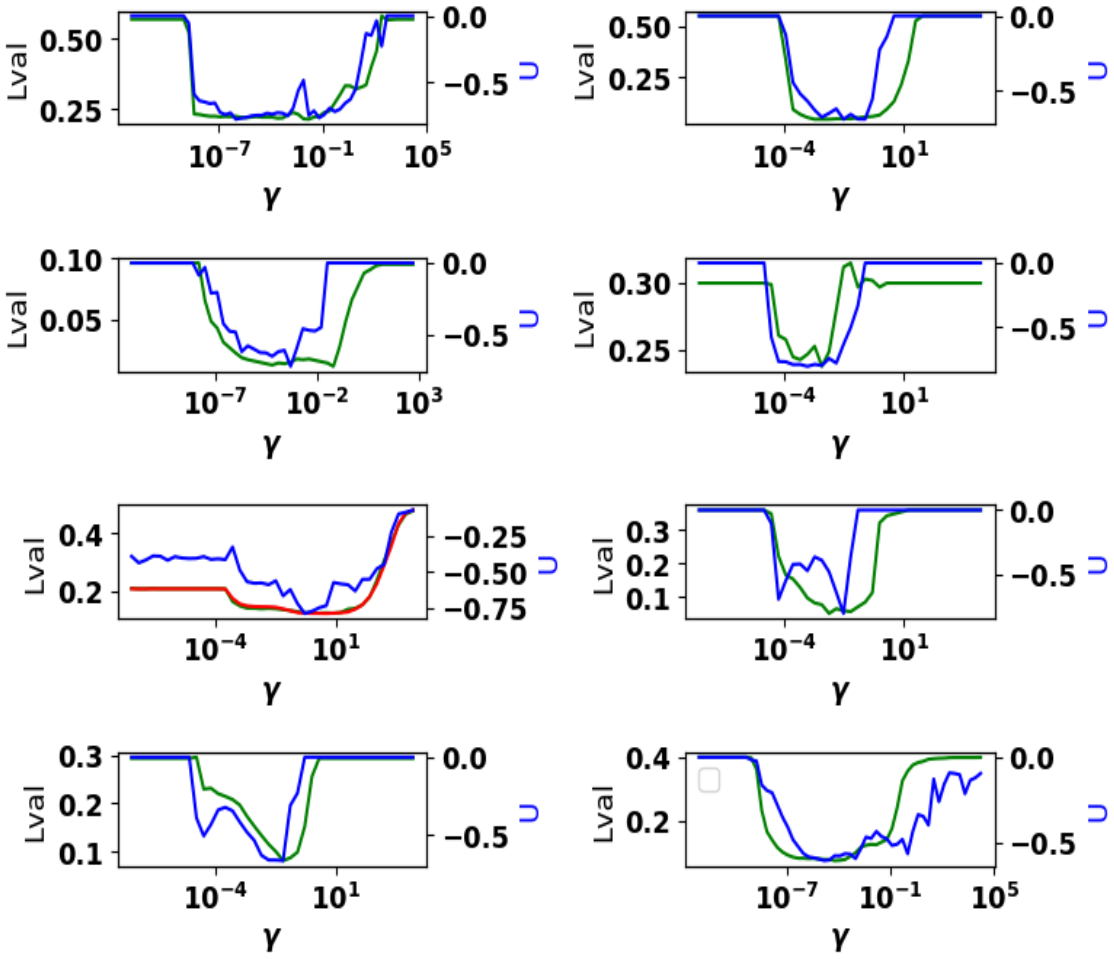

Figure 4.8: From left to right and top to bottom, SVM classifier status selection results for Abalone_01, Breast Cancer, Cardiotocography, German, GMM, Ionosphere, Landsat Satellite_47, Spambase. Left vertical axis, green: L_{val} and red: L_{te}. Right vertical axis, blue: $-U(\Lambda)$. Horizontal axis: γ.

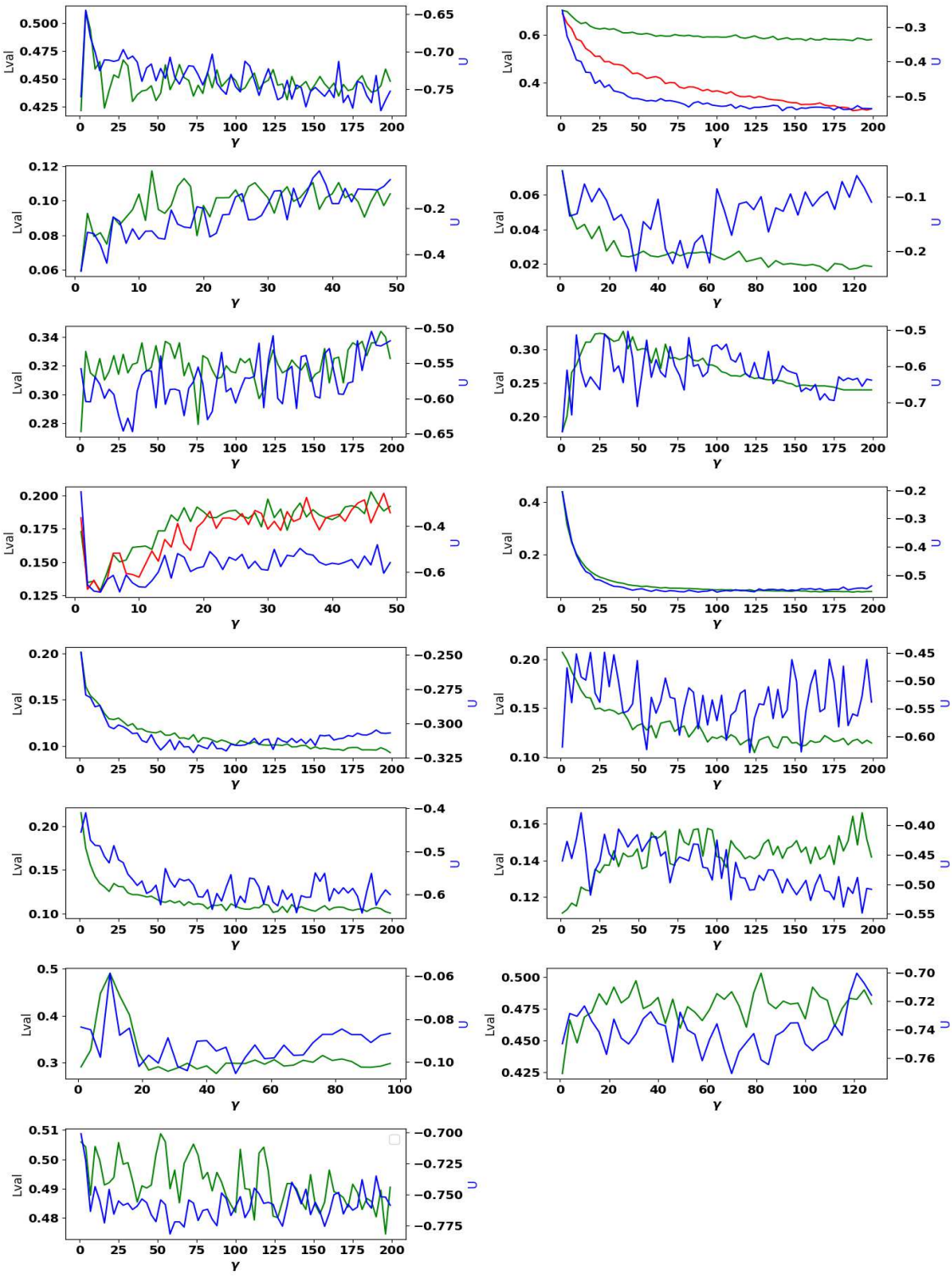

Figure 4.9: From left to right and top to bottom, PBC classifier status selection results for Abalone, Avila, Breast Cancer, Cardiotocography, German, GMM_5c, GMM, Letter Recognition, MNIST (test), Landsat Satellite_47, Landsat Satellite, Spambase, Thyroid, Wine Quality Red, Wine Quality White. Left vertical axis, green: L_{val} and red: L_{te}. Right vertical axis, blue: $-U(\Lambda)$. Horizontal axis: k.

Similarly to Chapter 2, we now assess the experimental results of our boundary uncertainty by observing its trend and its values. These results can be compared to the results in Section 2.4.6.

Trend of the boundary uncertainty. Figure 4.8 for SVMs show that the improved procedure consistently performs at least as well as our previous procedure in Section 2.4, or better on the more challenging datasets, such as the Cardiotocography, the German, and the Spambase datasets.

Figure 4.9 for PBCs also shows consistently favorable trends, except on the Spambase dataset and perhaps on the Cardiotocography dataset. Analyzing this issue will be a next research item.

Value of the boundary uncertainty. For all the datasets, $-U$ shows a minimum value of around -0.75 for the SVM classifier. The minimum value of the boundary uncertainty is -1, which shows that our boundary uncertainty estimation is quite reliable, but still has room for improvement.

Influence of imbalanced datasets on our boundary uncertainty estimation Despite not using our ad hoc modification of class posterior probabilities (Section 4.6.5), basically none of the classifier evaluation results seems disappointing on imbalanced datasets including on the Cardiotocography and the Thyroid datasets in Figures 4.8 and 4.9.

The discrete selection of $\mathcal{N}_B(\Lambda)$ (a training sample was close to $B(\Lambda)$, or not) was more likely the cause behind the weakness of our previous procedure to imbalanced datasets. Indeed, the lower panel in Figure 2.5 showed that $\mathcal{N}_B(\Lambda)$ reached a saturation for the minority class for almost any Λ, while more samples from the majority class could be freely added to $\mathcal{N}_B(\Lambda)$ as the complexity of $B(\Lambda)$ increased.

This saturation phenomenon pointed to the lack of sensitivity of our previous class posterior estimation based on the discrete selection of $\mathcal{N}_B(\Lambda)$: by ignoring informations such as the relative distance of samples to $B(\Lambda)$, our previous method heavily relied on pure counts of class

labels, and therefore suffered in more extreme settings (dataset imbalance aggravated by higher dimensionality), where the local number of class labels carries little information.

4.6.8 About the equivalence between maximum boundary uncertainty and classifier optimality

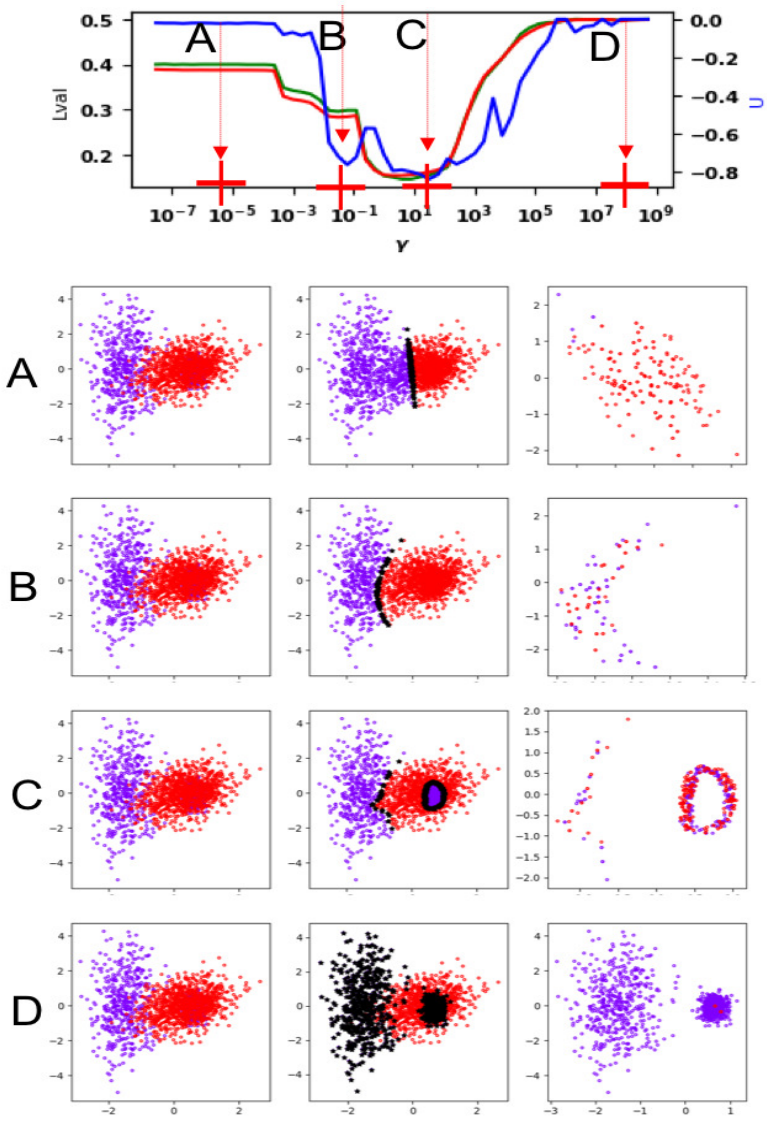

Figure 4.10: SVM evaluation results on the GMM_inclusion dataset. In the upper graph, the horizontal axis corresponds to the kernel width γ, the left vertical axis corresponds to L_{te} (red) and L_{val} (green), and the right vertical axis corresponds to $-U$ (blue). For each of the four classifier statuses A, B, C, D, we represent the following three plots in the corresponding row. From left to right: data with true class labels; data with labels estimated by the classifier, and anchors in black dots; zoom on the set of one nearest neighbors of the anchors, represented with their true class label.

So far, we assumed that the maximum value of our boundary uncertainty provides a necessary and sufficient condition for classifier optimality. In this section, we refine on this statement, and illustrate with a synthetic data called GMM_inclusion that we specifically prepared for this purpose.

Our definition of boundary uncertainty actually only measures the degree of *inclusion* of the decision boundary in the Bayes boundary. Therefore, the maximum value of our boundary uncertainty only guarantees a strict inclusion of the decision boundary in the Bayes boundary, but not necessarily *equality* of the two boundaries. In a theoretical sense, a maximum value of the boundary uncertainty only gives a necessary condition for classifier optimality, but not a sufficient condition, and the strict inclusion of the Bayes boundary in the decision boundary should also be ensured. However, in a practical sense, strict inclusion of the decision boundary in the Bayes boundary without satisfying the opposite strict inclusion would correspond to a decision boundary that somehow managed to perfectly fit some parts of the Bayes boundary, while completely "missing" some other parts of the Bayes boundary.

We illustrate with Figure 4.10 a situation where this issue may almost happen. For the GMM_inclusion dataset, for which we tried several values of the classifier status γ (horizontal axis), and estimated the error probability with CV (L_{val}: green curve), our boundary uncertainty ($-U$: blue curve), and the error probability with a large validation set (L_{val2}: red curve). The coincidence of the minimums of $L_{val}, L_{val2}, -U$ seem to clearly point at the classifier status (C) as the optimal classifier status, and we represent the corresponding decision boundary (presumably the Bayes boundary) through on-boundary anchors on the third row of graphs.

We can see that this Bayes boundary is made of several disconnected boundary parts. If the decision boundary does not have a sufficient representation capability to fit all these parts, then it may simply result in a very biased decision boundary such as in the classifier status (A). However, for such kind of Bayes boundary, under quite specific classifier settings of insufficient representation capability, we may also get quite close to strict inclusion of the decision boundary in the Bayes boundary such as for the classifier status (B).

Although probably not so common, we should handle such phenomenon in a later stage of our research. This phenomenon should be quite easy to detect. Indeed, a decision boundary that "misses" entire parts of the Bayes boundary would basically result in a trivially high amount of incorrect classifications even far from the decision boundary. Every time the maximum boundary uncertainty value is reached, we will therefore perform a rough check to detect such increasing amount of incorrect classification further away from the decision boundary. Adding this check to our classifier evaluation method will ensure that we provide a necessary and sufficient condition for classifier optimality.

4.7 Summary

In this chapter we improved the reliability and scalability of our boundary uncertainty-based classifier evaluation. In the improved Step 1, we emulate the more ideal situation of on-boundary anchors sampled from the mother set. In the improved Step 2, we apply at these ideally sampled targets a robust kNN estimation that is unbiased (centered on the boundary) and reduces variance through an appropriate smoothing. Proposal 1 is basically free from hyperparameters, and scalable contrary to Proposal 1. However, there is still room for improvement in terms of reliability as is shown on some datasets. Furthermore, there are still items to clarify: two items to clarify are the case of strict inclusion of the decision boundary in the Bayes boundary, and the use of discriminant functions to represent the degree of near-boundary-ness.

Appendix

The Parzen procedures detailed in this section can be found in [35].

4.7.1 Sample-dependent Parzen estimator

For convenience, we denote $\boldsymbol{h} = \{h_1, ..., h_k\}$. Given a one-dimensional training set $\{z_1, ..., z_{N_z}\}$, a sample-dependent Parzen estimator can be written as:

$$p(z; \boldsymbol{h}) = \frac{1}{N_z} \sum_{n=1}^{N_z} \frac{1}{z_n} \phi\Big(\frac{z - z_n}{h_n}\Big) \tag{4.6}$$

In contrast to a basic Parzen estimator whose kernel width is the same for all the training samples, the idea behind a sample-dependent Parzen estimator is that the window width value for each training point is adapted to the value of the density estimate at this point. Training samples where the density is lower are assigned a larger width to improve the quality of the smoothing.

To produce sample-dependent Parzen windows, we perform the following decomposition into a common factor h and a sample-dependent contribution w_n:

$$\forall n \in [1, N_z], h_n = h w_n \tag{4.7}$$

For convenience, we denote $\boldsymbol{w} = \{w_1, ..., w_{N_z}\}$

Algorithm 7 introduces a practical procedure for weights estimation, where the common factor h is adequately estimated at each step. For the purpose of completeness, Section 4.7.2 describes the procedure used to estimate h.

Algorithm 7: parzen: Estimation of $\boldsymbol{h}$ for a sample dependent kernel density estimator

Input: Training samples $\boldsymbol{z}_t, k_M$
Output: $h, \boldsymbol{w}$

$\boldsymbol{w} \leftarrow \mathbf{1}$
$h \leftarrow h^{(0)}$ using Eq. (4.20)
$k \leftarrow 0$
while $k < k_M$ **do**
  $g \leftarrow \left(\prod_{n=1}^{N_z} p(x_n; h\boldsymbol{w}) \right)^{\frac{1}{N_z}}$ using Eq. (4.6)
  $\boldsymbol{w} \leftarrow \left(\frac{p(\boldsymbol{z}_t; \boldsymbol{h})}{g} \right)^{-\eta}$
  $h \leftarrow$ parzen($h\boldsymbol{w}$) (Algorithm 8)
  $k \leftarrow k + 1$
end

4.7.2 Cross Validation Maximum Likelihood (CVML) estimation of h

Algorithm 7 assumed a proper estimation of h at every step. This section describes the estimation of h from the N_z training samples $\boldsymbol{z}_t = \{z_n\}_{n \in [1, N_z]}$. The purpose of the Parzen estimation is to find h that optimally fits the underlying density distribution. Therefore, to estimate the optimal h, the likelihood appears as a natural objective function to optimize.

However, naively using the likelihood defined over the entire $\boldsymbol{z}_t$ would return $h = 0$, in other words we would simply overfit to the training distribution. To circumvent this issue, the Parzen density is constructed by removing the n-th data sample, z_n from $\boldsymbol{z}_t$:

$$\forall n \in [1, N_z], p_{-n}(\boldsymbol{z}_t; \boldsymbol{h}) = \frac{1}{N_z - 1} \sum_{m \neq n}^{N_z} \frac{1}{h w_m} \phi\Big(\frac{z - z_m}{h w_m}\Big). \tag{4.8}$$

The associated likelihood function is the product over the training samples $\boldsymbol{z}_t$ of the Parzen density functions:

$$f(h; \boldsymbol{w}) = \prod_{n=1}^{N_z} p_{-n}(\boldsymbol{z}_t; \boldsymbol{h}) \tag{4.9}$$

Although the gradient descent is a possible way of maximizing $f(h; \boldsymbol{w})$, it would require a careful

manual initialization of the learning coefficient. To avoid such manual tuning, we re-formalize the CVML estimation procedure by using the concept of auxiliary function. We first summarize the optimization procedure is summarized in Algorithm 8.

Algorithm 8: Cross Validation Maximum Likelihood (CVML) estimation of the window width h

Input: Training samples $\boldsymbol{z}_t$, $\epsilon \leftarrow 0.001$
Output: h

$l \leftarrow 0$ $stop \leftarrow False$
$h \leftarrow$ parzen_init($\boldsymbol{z}_t$)
$F_{prev} \leftarrow F(h; \boldsymbol{w})$ using Eq. (4.10)
while *not stop* **do**
    E Step: compute $\boldsymbol{q}$ using Eq. (4.11)
    M step: compute h using Eq. (4.18)
    $F_{curr} \leftarrow F(h; \boldsymbol{w})$ using Eq. (4.10)
    $stop \leftarrow l = l_M$ or $|\frac{F_{curr} - F_{prev}}{F_{curr}}| < \epsilon$
    $l \leftarrow l + 1$
    $F_{prev} \leftarrow F_{curr}$
end

What follows are the details of the derivation of Algorithm 8. We define $F(h; \boldsymbol{w})$ as the logarithm of $(N_z - 1)^{N_z} f(h; \boldsymbol{w})$.

$$F(h; \boldsymbol{w}) = \sum_{n=1}^{N_z} \ln\left(\sum_{m \neq n}^{N_z} \frac{1}{hw_m} \phi\left(\frac{z_n - z_m}{hw_m} \right) \right). \tag{4.10}$$

While being a monotone increasing function of $f(h; \boldsymbol{w})$, $F(h; \boldsymbol{w})$ is more optimization-friendly. Next, define

$$q_{n,m}(h; \boldsymbol{w}) = \frac{\frac{1}{hw_m} \phi\left(\frac{z_n - z_m}{hw_m} \right)}{\sum_{k \neq n}^{N_z} \frac{1}{hw_k} \phi\left(\frac{z_n - z_k}{hw_k} \right)}, \tag{4.11}$$

where $\{q_{n,m}\}_{m \neq n}^{N_z}$ satisfies $q_{n,m} > 0$ and $\sum_{m \neq n}^{N_z} q_{n,m} = 1$. For convenience, we denote $\boldsymbol{q}$ as $\{q_{n,m}, n \in [1, N_z], m \in [1, N_z]\}$.Assuming that $F(h; \boldsymbol{w})$ is optimized by an iterative procedure and that estimate $\widehat{h}$ has already been calculated before the last preceding iteration step, we define the follow-

ing auxiliary function for the successive iteration step:

$$W(h;\boldsymbol{w}) = \sum_{n=1}^{N_z}\sum_{m\neq n}^{N_z} q_{n,m}(\widehat{h};\boldsymbol{w})\ln\left(\frac{\phi\left(\frac{z_n - z_k}{hw_k}\right)}{q_{n,m}(\widehat{h};\boldsymbol{w})}\right). \tag{4.12}$$

We also define the following difference function:

$$K(h;\boldsymbol{w}) = F(h;\boldsymbol{w}) - W(h;\boldsymbol{w}). \tag{4.13}$$

Substituting (4.10) and (4.12) into (4.13), and considering (4.11), we reach the following expression:

$$\begin{aligned} K(h;\boldsymbol{w}) &= \sum_{n=1}^{N_z}\sum_{m\neq n}^{N_z} q_{n,m}(\widehat{h};\boldsymbol{w})\ln\left(\frac{q_{n,m}(h;\boldsymbol{w})}{q_{n,m}(\widehat{h};\boldsymbol{w})}\right) \\ &\geq \sum_{n=1}^{N_z}\sum_{m\neq n}^{N_z} q_{n,m}(\widehat{h};\boldsymbol{w})\left(1 - \frac{q_{n,m}(h;\boldsymbol{w})}{q_{n,m}(\widehat{h};\boldsymbol{w})}\right) = 0. \end{aligned} \tag{4.14}$$

We used the logarithm-based inequality: $\forall x, -\ln(x) >= 1 - x$, that only achieves equality for $x = 1$. Therefore, the inequality in (4.14) becomes an equality if and only if $\forall n, m \in [1, N_z], m \neq n, q_{n,m}(h) = q_{n,m}(\widehat{h})$.

In other words, $K(h;\boldsymbol{w})$ is minimized at $h = \widehat{h}$, and the minimum value is zero. Considering (4.13), this leads to:

$$F(\widehat{h};\boldsymbol{w}) = W(\widehat{h};\boldsymbol{w}) \tag{4.15}$$

Also,

$$\nabla F(\widehat{h};\boldsymbol{w}) = \nabla W(\widehat{h};\boldsymbol{w}). \tag{4.16}$$

From (4.14) and (4.15), we get:

$$F(h;\boldsymbol{w}) \geq W(h;\boldsymbol{w}) > W(\widehat{h};\boldsymbol{w}) = F(\widehat{h};\boldsymbol{w}). \tag{4.17}$$

Furthermore, based on (4.16), unless $\widehat{h}$ is a stationery point of $W(h; \boldsymbol{w})$, $\nabla W(\widehat{h}; w_1, ..., w_n)$ is nonzero. We can find h such that $W(h; \boldsymbol{w}) > W(\widehat{h}; \boldsymbol{w})$. Consequently, applying the following steps leads to a monotonic increase of $F(h; \boldsymbol{w})$ until h reaches its local maximum point:

I. Initialize $\widehat{h}$.

II. Find h such that $W(h; \boldsymbol{w}) > W(\widehat{h}; \boldsymbol{w})$.

III. Replace $\widehat{h}$ with h, go back to item II.

The point h which maximizes the auxiliary function $W(h; \boldsymbol{w})$ is given as the following closed-form formula:

$$h = \sqrt{\frac{1}{N_z} \sum_{n=1}^{N_z} \sum_{m \neq n}^{N_z} q_{n,m}(\widehat{h}; \boldsymbol{w}) \, (z_n - z_m)^2}. \tag{4.18}$$

Computing h from $q_{n,m}(\widehat{h}; \boldsymbol{w})$ in Eq. (4.18) (item II), and then replacing $\widehat{h}$ with h (item III) leads to the iterative procedure described in Algorithm 8.

Initialization The iterative procedure described in Algorithm 8 requires a proper initialization for h. $q_{n,m}$ corresponds to the degree to which each data point z_m is assigned to another point $z_n (m \neq n)$. Based on this interpretation, a possible initialization method is to assign each data point z_n to its nearest-neighbour $z_{k(n)}$.

is obtained by redefining $q_{n,m}(\boldsymbol{h})$ as

$$q_{n,m}(\boldsymbol{h}) = \begin{cases} 1, & \text{if } m = k(n) \\ 0, & \text{otherwise.} \end{cases} \tag{4.19}$$

Substituting this into Eq. (4.18) provides the initial value for h:

$$h^{(0)} = \sqrt{\frac{1}{N_z} \sum_{n=1}^{N_z} \sum_{m \neq n}^{N_z} \left(z_n - z_{k(n)}\right)^2}. \tag{4.20}$$

5 Conclusion

5.1 Summary of Dissertation

We introduced a new method to evaluate a general form of classifier. The purpose was to overcome the fundamental limitations of the standard methods (i.e. training repetition, data splitting in validation, difficulty in applying them). Accordingly, we defined a new classifier evaluation measure called boundary uncertainty that is easy to accurately estimate despite the finiteness of the data, due to a sharp focus of the estimation task on specific and well-defined regions. More precisely, our boundary uncertainty sharply focuses on the equality of the class posterior probabilities on the decision boundary. Incidentally, this estimation target is robust to estimation errors. We proposed a procedure to accurately estimate our boundary uncertainty.

The experimental results and a comparison with the benchmark CV method first indicated the possibility of selecting the optimal model on several real-life classification tasks. To better understand our encouraging results, we mathematically analyzed and proved the validity of our posterior probability estimation procedure, which plays a central role in the proposed method for finding the optimal classifier parameter status.

Based on this analysis, we improved the accuracy of our procedure to improve its accuracy, and at the same time, its scalability and robustness. The improved procedure clearly showed improvements compared to the baseline, and can readily be applied on any types of classifier.

5.2 Future Works

- Further improvements in terms of accuracy and scalability, in particularly Step 1, as it still constitute the most costly and complicated part of our method.
- Theoretical analysis of the improved procedures to assess the accuracy of our boundary uncertainty estimate.
- Refine the method to even more safely ensure the equivalence between classifier optimality and maximum value of boundary uncertainty (detection of the strict inclusion of the decision boundary in the Bayes boundary).
- Application to larger tasks, such as speech classification.
- Boundary uncertainty-based classifier training: a characteristic of our boundary uncertainty is its potential to be used as a training objective function. To this end, our improved procedure successfully formalized the boundary uncertainty as a function of the classifier parameters, which opens the way to a new form of classifier training.

Bibliography

[1] C. Bishop, "Pattern recognition and machine learning," chapter 1.5.1 Minimizing the misclassification rate, pp. 39–40. Bishop, 2006.

[2] T. Hastie, R. Tobshirani, and J. Friedman, "The elements of statistical learning," chapter 7.4. Optimism of the Error Rate, pp. 228–230. Springer Science, 2009.

[3] C. Burges, "A tutorial for support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 1485–1510, 2009.

[4] I. Guyon, V. Vapnik, B. Boser, L. Bottou, and S.A. Solla, "Structural risk minimization for character minimization," in *Proceedings of Advances in Neural Information Processing Systems 4 (NIPS 1991)*, 1991, pp. 471–479.

[5] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society*, vol. 36, no. 2, pp. 111–147, 1974.

[6] L. Devroye, L. Gyorfi, and G. Lugosi, "A probabilistic theory of pattern recognition," chapter 24 Deleted Estimates of the Error Probability, pp. 407–419. Springer, 1997.

[7] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*, Chapman and Hall, 1993.

[8] S. Demyanov, J. Bailey, K. Ramamohanarao, and C. Leckie, "Aic and bic based approaches for svm parameter value estimation with rbf kernels," *Journal of Machine Learning Research*, vol. 25, pp. 97–112, 2012.

[9] L. Devroye, L. Gyorfi, and G. Lugosi, "A probabilistic theory of pattern recognition," chapter 31.1 Smoothing the Error Count, pp. 550–554. Springer, 1997.

[10] B.-H. Juang and S. Katagiri, "Discriminative learning for minimum error classification," *IEEE Transactions on Signal Processing*, vol. 30, pp. 3043–3054, 1992.

[11] E. MacDermott and S. Katagiri, "A derivation of minimum classification error from the

theorretical classification risk using parzen estimation," in *Computer Speech and Language*, 2004, vol. 18, pp. 107–122.

[12] T. Ohashi, H. Watanabe, J. Tokuno, S. Katagiri, M. Ohsaki, S. Matsuda, and H. Kashioka, "Increasing virtual samples through loss smoothness determination in large geometric margin minimum classification error training," in *Proceedings of 2012 International Conference on Acoustics, Speech and Signal Processing (ICASSP 2012)*, 2012, pp. 2081–2084.

[13] L. Devroye, L. Gyorfi, and G. Lugosi, "A probabilistic theory of pattern recognition," chapter 8.5. Estimating the Bayes error, p. 128. Springer, 1997.

[14] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, pp. 716–722, 1974.

[15] C. Bishop, "Pattern recognition and machine learning," chapter 3.4. Bayesian model selection, pp. 42–44. Bishop, 2006.

[16] G. Bouchard and G. Celleux, "Selection of generative models in classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 544–554, 2006.

[17] A. Biem, "A model selection criterion for classification: Application to hmm topology optimization," in *Proceedings of IEEE Seventh International Conference on Document Analysis and Recognition*, 2003.

[18] A. Biem, "Discriminative model selection for belief net structures," in *Proceedings of AAAI Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 2005.

[19] D. Ha, J. Maes, Y. Tomotoshi, C. Melle, H. Watanabe, S. Katagiri, and M. Ohsaki, "A class boundary selection criterion for classification," in *Proceedings of IPSJ Kansai-Branch Convention 2017*, 2017.

[20] D. Ha, J. Maes, Y. Tomotoshi, H. Watanabe, S. Katagiri, and M. Ohsaki, "A classification-uncertainty-based criterion for classification boundary," in *IEICE Technical Report*, 2018, pp. 121–126.

[21] D. Ha, E. Delattre, Y. Tomotoshi, M. Senda, H. Watanabe, S. Katagiri, and M. Ohsaki, "Op-

timal classifier model status selection using bayes boundary uncertainty," in *Proceedings of 2018 IEEE International Workshop in Machine Learning for Signal Processing*, 2018.

[22] Y. Tomotoshi, D. Ha, E. Delattre, H. Watanabe, S. Katagiri, and M. Ohsaki, "Optimal classifier status selection using class boundary uncertainty measure for prototype-based and neural network classifier," in *International Symposium on Integrated Uncertainty in Knowledge Modeling and Decision Making*, 2019.

[23] O. Chapelle and V. Vapnik, "Model selection for support vector machines," *Advances in Neural Information Processing Systems*, vol. 12, 2000.

[24] U. Luxburg, Olivier Bousquet, and Bernhard Scholkopf, "Compression approach to support vector model selection," *Journal of Machine Learning Research*, vol. 5, pp. 293–323, 2004.

[25] S. Demyanov, J. Bailey, K. Ramamohanarao, and C. Leckie, "Aic and bic based approaches for svm parameter value estimation with rbf kernels," 2012, vol. 25, pp. 97–112.

[26] C. Bishop, "Pattern recognition and machine learning," chapter Combining models, p. 666. Springer, 2006.

[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[28] V. Vapnik, "Statistical learning theory," chapter The Structural Risk Minimization Principle. Wiley, 1998.

[29] K. Fukunaga, "Introduction to statistical pattern recognition (2nd edition)," chapter 2.1., pp. 313–321. Academic Press, 1990.

[30] C. Bishop, "Pattern recognition and machine learning," chapter Kernel density estimators, pp. 122–124. Bishop, 2006.

[31] B. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, 1986.

[32] H. Watanabe, T. Ohashi, S. Katagiri, M. Ohsaki, S. Matsuda, and H. Kashioka, "Robust and efficient pattern classification using large geometric margin minimum classification

error training," *Journal of Signal Processing Systems*, vol. 74, pp. 297–310, 2014.

[33] D. Doane, "Aesthetic frequency classifications," *The American Statistician*, vol. 30, pp. 181–183, 1976.

[34] D. Freedman and P.Diaconis, "On the histogram as a density estimator: L2 theory," *Zeitschrift fur Wahrscheinlichkeitstheorie und Verwandte Gebiete*, vol. 57, pp. 453–476, 1981.

[35] H. Watanabe, J. Tokuno, T. Ohashi, S. Katagiri, and M. Ohsaki, "Minimum classification error training with automatic setting of loss smoothness," in *2011 IEEE international Workshop on machine Learning for Signal Processing*, 2011.

A Notation List

A.1 Notations

Formalization of the Classification Problem	
$\mathcal{X}$	pattern space
D	dimensionality of the pattern space
$\boldsymbol{x}$	data sample in the pattern space
J	number of classes
C_j	class j
Λ	classifier parameter status
$\widehat{C}(\cdot)$	classifier decision function (classification operator)
$\widehat{C}_j$	estimated class j
$B(\Lambda)$	estimated boundary
$B_{ij}(\Lambda)$	boundary between $\widehat{C}_i$ and $\widehat{C}_j$
B^*	Bayes boundary
Λ^*	classifier parameter status that corresponds to the Bayes boundary

Proposed Procedure	
$\mathcal{T}$	training set
$NN(\boldsymbol{x}, m)$	set of m nearest neighbors of sample $\boldsymbol{x}$ in $\mathcal{T}$
$k(\mathcal{S})$	given finite set $\mathcal{S}$, number of elements in $\mathcal{S}$
$\mathcal{N}_B(\Lambda)$	set of selected near-boundary samples
$A(\Lambda)$	set of generated anchors
$E[U(\Lambda)]$	expected value of the boundary uncertainty
$U(\Lambda)$	estimated boundary uncertainty

B Publication List

Journal Papers

I. **David Ha**, Yuya Tomotoshi, Masahiro Senda, Hideyuki Watanabe, Shigeru Katagiri, Miho Ohsaki, "A Practical Method Based on Bayes Boundary-ness for Optimal Classifier Parameter Status Selection," *Journal of Signal Processing Systems*, 2019.

Peer-Reviewed International Conference Proceedings

I. **David Ha**, Hideyuki Watanabe, Yuya Tomotoshi, Emilie Delattre, Shigeru Katagiri, "Optimality Analysis of Boundary-Uncertainty-Based Classifier Selection Method," *ACM International Conference on Signal Processing and Machine Learning*, 2018.

II. **David Ha**, Emilie Delattre, Yuya Tomotoshi, Masahiro Senda, Hideyuki Watanabe, Shigeru Katagiri, Miho Ohsaki, "Optimal Classifier Model Status Selection Using Bayes Boundary Uncertainty," *IEEE International Workshop on Machine Learning for Signal Processing*, 2018.

III. Yuya Tomotoshi, **David Ha**, Emilie Delattre, Hideyuki Watanabe, Shigeru Katagiri, Miho Ohsaki, "Optimal Classifier Status Selection Using Class Boundary Uncertainty Measure for Prototype-Based and Neural Network Classifier," *International Symposium on Integrated Uncertainty in Knowledge Modeling and Decision Making*, 2019.

IV. **David Ha**, Emilie Delattre, Yuya Tomotoshi, Masahiro Senda, Hideyuki Watanabe, Shigeru Katagiri, Miho Ohsaki, "Improvement for Boundary-Uncertainty-Based Classifier Parameter Status Selection Method," *IEEE International Conference on Computational Electro-*

magnetics, 2019.

s

Non-Reviewed Domestic Workshop Proceedings

I. **David Ha**, Juliette Maes, Yuya Tomotoshi, Charles Melle, Hideyuki Watanabe, Shigeru Katagiri, Miho Ohsaki, "A Class Boundary Selection Criterion for Classification," *Proceedings of IPSJ Kansai-Branch Convention 2017*, 2017.

II. **David Ha**, Juliette Maes, Yuya Tomotoshi, Hideyuki Watanabe, Shigeru Katagiri, Miho Ohsaki, "A Classification-Uncertainty-Based Criterion for Classification Boundary," *IEICE Technical Report*, pp. 121-126, 2018.