

UNIVERSITY OF TARTU  
Institute of Computer Science  
Computer Science Curriculum

Mohamed Maher Abdelrahman

# Instance-based Label Smoothing for Better Classifier Calibration

Master's Thesis (30 ECTS)

Supervisor: Meelis Kull, PhD

Tartu 2020

# Instance-based Label Smoothing for Better Classifier Calibration

**Abstract:** *Binary classification* is one of the fundamental tasks in machine learning, which involves assigning one of two classes to an instance defined by a set of features. Although accurate predictions are essential in most of the tasks, knowing the model confidence is indispensable in many of them. Many probabilistic classifiers' predictions are not well-calibrated and tend to be *overconfident*, requiring further calibration as a post-processing step to the model training.

*Logistic calibration* is one of the most popular calibration methods, that fits a logistic regression model to map the outputs of a classification model into calibrated class probabilities. Various regularization methods could be applied to logistic regression fitting to reduce its overfitting on the training set. *Platt scaling* is one of these methods, which applies label smoothing to the class labels and transforms them into target probabilities before fitting the model to reduce its overconfidence. Also, label smoothing is widely used in classification neural networks. In previous works, it was shown that label smoothing has a positive calibration and generalization effect on the network predictions. However, it erases information about the similarity structure of the classes by treating all incorrect classes as equally probable, which impairs the distillation performance of the network model.

In this thesis, we aim to find better ways of reducing overconfidence in logistic regression. Here we derive the formula of a Bayesian approach for the optimal predicted probabilities in case of knowing the generative model distribution of the dataset. Later, this formula is approximated by a sampling approach to be applied practically. Additionally, we propose a new *instance-based* label smoothing method for logistic regression fitting. This method motivated us to present a novel label smoothing approach that enhanced the distillation and calibration performance of neural networks compared with standard label smoothing.

The evaluation experiments confirmed that the approximated formula for the derived optimal predictions is significantly outperforming all other regularization methods on synthetic datasets of known generative model distribution. However, in more realistic scenarios when this distribution is unknown, our proposed instance-based label smoothing had a better performance than Platt scaling in most of the synthetic and real-world datasets in terms of log loss and calibration error. Besides, neural networks trained with instance-based label smoothing, outperformed the standard label smoothing regarding log loss, calibration error, and network distillation.

**Keywords:** Machine Learning, Logistic Regression, Platt Scaling, Label Smoothing,

Probabilistic Classifiers, Bayesian Reasoning, Neural Networks

**CERCS:** P176 Artificial intelligence

## **Märgendite andmepunktipõhine silumine klassifikaatorite paremaks kalibreerimiseks**

**Lühikokkuvõte:** Binaarne klassifikatsioon on üks masinõppe peamistest ülesannetest. See hõlmab endas tunnuste komplekti poolt määratud andmepunktide määramist ühte kahest klassist. Kuigi täpsed ennustused on enamikes ülesannetes olulised, on väga paljudel juhtudel oluline ka mudeli enesekindlus. Paljude tõenäosuslike klassifikatsioonide ennustused pole hästi kalibreeritud ja kipuvad olema liigse enesekindlusega, seetõttu vajades täiendavat kalibreerimist mudeli treenimise järeltöötluses.

Logistiline kalibreerimine on üks populaarsematest kalibreerimismeetoditest. See sobitab logistilise regressiooni mudeli klassifikatsiooni mudeli väljunditele, et saada kalibreeritud tõenäosusjaotus üle kõigi klasside. Treeningandmete ülesobitamise vähendamiseks saab logistilise regressioonimudeli sobitamisel rakendada mitmeid regulariseerimismeetodeid. Platti skaleerimine on üks neist meetoditest ning kasutab märgendite silumist klasside märgenditel ja muudab nad sihttõenäosusteks enne logistilise regressioonimudeli rakendamist, seekaudu langetades mudeli liigset enesekindlust. Lisaks sellele kasutatakse märgendite silumist ka klassifikatsiooni teostavates närvivõrkudes. Eelnev teadustöö on näidanud, et märgendite silumisel on positiivsete efektidega mudelite kalibreeritusele ja üldistusvõimele. Samas kustutab see informatsiooni klasside sarnasusstruktuuridest, koheldes kõiki valesid klasse võrdtõenäolisena, seekaudu kahjustades treenitud võrgu destilleerimisvõimekust.

Antud magistritöös on eesmärk leida paremaid meetodeid logistilise regressiooni liigse enesekindluse vähendamiseks. Selle jaoks on tuletatud uus Bayesi lähenemisel baseeruv valem optimaalsete tõenäosusprognoside leidmiseks, eeldusel et generatiivse mudeli jaotus on teada. Selle valemi rakendamiseks praktikas on leitud ka lähendusmeetod. Lisaks sellele on välja pakutud uus märgendite andmepunktipõhine silumise meetod logistilise regressiooni sobitamiseks. Sellest meetodist motiveerituna on esitatud ka uudne lähenemine närvivõrkude märgendite silumisele, mis täiendas destilleerimis- ja kalibreerimisvõimekust võrreldes tavalise märgendite silumisega.

Meetodite hindamiseks teostatud eksperimendid kinnitasid, et töös tuletatud optimaalsete ennustuste lähendusvalem edestab tulemustelt kõiki teisi regulariseerimismeetodeid autorite poolt sünteesitud andmestikes, kus on generatiivse mudeli jaotus teada. Seevastu realistlikumatel juhtudel, kus see jaotus pole teada, näitab uus märgendite andmepunktipõhine silumine paremat jõudlust võrreldes Platti skaleerimisega enamikes nii tegelikes kui sünteesitud andmestikes nii logaritmilise kaofunktsiooni kui ka kalibreerimisvea poolest. Lisaks, märgendite andmepunktipõhine silumine edestas närvivõrkude treenimisel traditsioonilist märgendite silumist nii logaritmilise kahju, kalibreerimisvea kui ka võrgu destilleerimise poolest.

**Võtmesõnad:** masinõppe, logistiline regressioon, Platti skaleerimine, märgendite silumine, tõenäosuslikud klassifikaatorid, Bayesi statistika, närvivõrgud

**CERCS:** P176 Tehisintellekt

## *Acknowledgments*

I would like to thank my thesis supervisor Dr. Meelis Kull for the guidance, encouragement, and advice he has provided to me throughout a full academic year. It would have not been possible for me to defend if it was not with his help and support. He always had the patience to listen to my ideas, provide fruitful feedback and, steer me in the right direction whenever I needed it the most. I am also very grateful for the many opportunities he gave me like joining the Machine Learning group at Institute of computer science, University of Tartu, where I gained various research skills, learned about many interesting topics in the field of Machine Learning and enjoyed the interesting discussions among Meelis, and my colleagues in the group.

I would like also to express my profound gratitude to my parents, and my sisters for their love, endless support, and encouragement throughout all years of my studies. Finally, I dedicate my work to the memory of Prof. Dr. Sherif Sakr who always believed in my ability to be successful in the academic arena.

Mohamed Maher  
May, 2020

# Contents

<b>1</b>	<b>Introduction</b>	<b>8</b>
1.1	Why should we care about calibration? . . . . .	8
1.2	Logistic regression and sigmoid calibration . . . . .	9
1.3	Reducing overconfidence by label smoothing . . . . .	9
1.4	Thesis structure . . . . .	11
<b>2</b>	<b>Background</b>	<b>12</b>
2.1	Types of classifiers . . . . .	12
2.1.1	Labeling classifiers . . . . .	12
2.1.2	Scoring classifiers . . . . .	12
2.2	Classifier calibration . . . . .	13
2.2.1	Calibration of linear classifiers scores . . . . .	13
2.3	Evaluation metrics . . . . .	16
2.3.1	Log Loss . . . . .	16
2.3.2	Confidence reliability plot . . . . .	16
2.3.3	Expected calibration error . . . . .	17
2.4	Logistic regression . . . . .	18
2.5	Bayesian reasoning in machine learning . . . . .	20
2.6	Deep neural networks . . . . .	21
2.6.1	Temperature scaling . . . . .	22
<b>3</b>	<b>Related work</b>	<b>23</b>
3.1	Using logistic regression in classifiers calibration . . . . .	23
3.2	Solutions to overfitting in logistic regression . . . . .	24
3.2.1	Regularization . . . . .	25
3.2.2	Bayesian logistic regression . . . . .	26
3.2.3	Label smoothing, and Platt scaling . . . . .	26
<b>4</b>	<b>Optimal predictions for datasets with known generative models distribution</b>	<b>28</b>
4.1	Bayes-optimal model for a binary classification task with normally distributed likelihoods . . . . .	28
4.2	Derivation of the optimal predictions given a known dataset generative model and a training set . . . . .	30
4.2.1	Practicality Concerns . . . . .	34
<b>5</b>	<b>Instance-based label smoothing using kernel density estimation</b>	<b>37</b>
5.1	Disadvantages of the constant factor label smoothing . . . . .	37
5.2	The intuition behind the instance-based label smoothing . . . . .	37
5.3	Kernel density estimation . . . . .	40
5.4	Methodology . . . . .	41

<b>6</b>	<b>Label smoothing in classification with neural networks</b>	<b>44</b>
6.1	Standard label smoothing: Fors and Againsts . . . . .	44
6.2	Instance-based label smoothing in neural networks . . . . .	45
<b>7</b>	<b>Experimental results</b>	<b>48</b>
7.1	Synthetic datasets experiments . . . . .	49
7.1.1	Setup . . . . .	49
7.1.2	Results . . . . .	51
7.2	Real datasets experiments for logistic regression . . . . .	57
7.2.1	Setup . . . . .	57
7.2.2	Results . . . . .	57
7.3	Experiments for label smoothing in neural networks . . . . .	59
7.3.1	Setup . . . . .	60
7.3.2	Results . . . . .	60
<b>8</b>	<b>Conclusion</b>	<b>63</b>
	<b>References</b>	<b>67</b>
	<b>Appendix</b>	<b>68</b>
	I. Source Code . . . . .	68
	II. Licence . . . . .	68

Abbreviation	Definition
<b>pdf</b>	Probability density function
<b>KDE</b>	Kernel density estimation
<b>MLE</b>	Maximum likelihood estimation
<b>MAP</b>	Maximum a posteriori
<b>CD</b>	Critical Difference diagram
<b>SVM</b>	Support vector machines model
<b>LR</b>	Logistic regression model
<b>DNN</b>	Deep neural networks
<b>ECE</b>	Expected calibration error
<b>DNNs</b>	Deep Neural Networks
<b>CNNs</b>	Convolutional Neural Networks

Table 1. List of abbreviations

Symbol	Description
$D_{tr}, D_{te}$	Training, and testing datasets
$D^+, D^-$	Positive, and negative class instances of dataset $D$
$\mathcal{X}, \mathcal{Y}$	Features space, and labels space of a particular dataset.
$\hat{p}(x)$	predicted probability of instance $x$
$\hat{f}(z)$	Estimated probability density of an instance $z$
$K_h(z)$	Non-negative kernel function of bandwidth $h$
$T$	Temperature scaling parameter in neural networks
$\epsilon^+, \epsilon^-$	Smoothing factor using for positive, and negative classes respectively.
$\epsilon_{Platt}^+, \epsilon_{Platt}^-$	Platt scaling label smoothing factors for positive, and negative classes respectively.
$\mu_0, \mu_1, \sigma$	Mean of negative, positive classes, and standard deviation of their Gaussian distributions respectively
$\mathcal{D}_x$	The probability distribution of a variable $x$
$p^*$	Optimal output predicted probability
$\delta(t - c)$	Dirac delta function at $c$
$N_{MC}$	Number of Monte Carlo samples
$\mathcal{U}(a, b), \mathcal{B}(a, b)$	Uniform, and Beta distributions of parameters $a$ , and $b$ .
$\mathcal{N}(\mu, \sigma)$	Gaussian distribution of mean $\mu$ , and standard deviation $\sigma$ .
$clr$	Class balance ratio = $\frac{ D_{tr}^+ }{ D_{tr}^+  +  D_{tr}^- }$ .

Table 2. Notation used in mathematical formulas described throughout the thesis.

# 1 Introduction

During the past decade, *machine learning* has been used in a wide range of applications. Thanks to the accelerated research progress in this field, machine learning *models* have been widely utilized in real-world use cases like healthcare [PSA18], transportation [ADLB19], and agriculture [JDPS19].

Although these models outperform humans in many tasks and even with significant degrees, deploying them in more critical tasks that require making decisions that directly impact people's lives could be quite risky. Naturally, critical decisions are not going to be taken except when the model has a high degree of *confidence* in its predictions. The confidence of a model usually refers to its estimated output probability or score of an observation belonging to a specific class. However, highly accurate models do not necessarily reflect correct confidence levels. Therefore, taking measures regarding the calibration of model predictions to avoid scenarios of overconfidence is an important part of building machine learning models. For instance, a model that predicts with a 90% confidence level should have around 90 correct out of every 100 predictions. Despite the continuous advancements in improving the accuracy of models, relying on these models' predictions requires parallel progress in calibrating their outputs.

## 1.1 Why should we care about calibration?

Consider a highly accurate machine learning model that was trained and employed in a *self-driving car*. Suppose that this model predicts an empty road with confidence of almost 100% while a pedestrian is crossing the street. Given such a perfect level of confidence, the model might take the decision to continue accelerating, which could lead to disastrous consequences.

Second, after the novel Coronavirus (COVID19) has hit the world, the machine learning research community started to direct their efforts towards offering solutions to this crisis. One type of these efforts is using machine learning for a fast diagnosis of COVID19. For example, 13 published studies were proposing different machine learning models to detect this viral infection based on tomography scans [WVCB<sup>+</sup>20]. Although most of these models claim very accurate results, one needs to know when to rely on the model predictions, or when to ask for a medical expert confirmation using other diagnostic tests.

Suppose that two of these diagnostic models were tested over a group of 100 infected patients. Model **A** has accuracy and confidence of 90% on each of these patients, and Model **B** has 95% accuracy but 99.9% confidence. Although the second model has higher accuracy than the first, its erroneous predictions with overconfidence could be problematic to the patient's physical health and the whole population. However, Model A is *calibrated*, which makes it a more reliable model.

The above examples and many others show how confidence in machine learning is a matter of interest. No doubt that based on the consequences of the made decisions, different tasks need different levels of confidence to act upon. However, these levels should be highly accurate, such that a non-calibrated model is not going to be useful even if it has high confidence in its predictions. Thus, using highly calibrated machine learning models, and reducing overconfident predictions could be as crucial as developing highly accurate models in some real-life scenarios.



## 1.2 Logistic regression and sigmoid calibration

*Regression* models are a broad class of predictive models in machine learning. The aim of using these models is to find the relationship between a group of independent variables (*features*), and a numeric dependent variable (*target*). On the other hand, if the target variable belongs to a set of discrete values, the task is referred to as *classification*. A particular case of classification tasks is when the set of possible target values includes only two different states. In that case, the problem is referred to as a *binary classification* task. There is no doubt that the set of binary classification problems widely exists in our real-world tasks, where machine learning solutions tend to sort items into one out of two classes. The predominance of this particular set of tasks could be imagined from many examples in various fields like the prediction of churn or not, spam or not, illness or not, etc. Moreover, having multiple classes in other tasks could be interpreted as a prediction of one class versus the remaining ones, which could be achieved by aggregating the outputs from multiple binary classification models.

*Logistic Regression* (LR) model is a prevalent probabilistic machine learning model that is employed in binary classification tasks. Firstly, it can be described by simple equations that ease its deployment and provide a quantified representation for the relative importance of each feature to the target variable. Secondly, it squashes the output scores into a range between 0 and 1. Hence, it outputs the probability of belonging to a particular class as a continuous variable. LR is used for binary classification by employing a decision rule on its estimated output probabilities. An observation belongs to the positive class if it satisfies that decision rule such that its estimated probability exceeds a certain threshold. Otherwise, it belongs to the negative class [HJ15].

Many highly accurate binary classification models like decision trees, naive Bayes [ZE01], and support vector machines (SVM) [Dri01] could lack calibrated confidence levels. However, they usually output scores that are linearly related to the target variable. An important use-case for LR is calibrating classifiers' output predictions such that a univariate LR model is fitted to convert the output scores of the uncalibrated classifiers into better-calibrated probabilities. This process is well known by *logistic* or *sigmoid calibration*, and it is usually beneficial if the distortion in the predicted scores has a sigmoid shape. Consequently, better certainty estimates for model predictions could be obtained. Figure 1 (left) shows an example for the reliability diagram of an uncalibrated SVM model with average distorted predicted scores similar to a sigmoid function. In the reliability diagram, the predicted probabilities are divided into ten bins along the x-axis, and the average score in each bin is plotted against the average model accuracy. After performing logistic calibration, the model output predictions become very close to that of the perfect calibrated model. A perfectly calibrated model would be on the diagonal of the diagram.

## 1.3 Reducing overconfidence by label smoothing

When LR is trained like any other supervised classifier, the training set is usually not enough to learn the actual distribution of the dataset. Hence, it is a critical problem that fitting a standard LR model on the training set would suffer from overconfidence. For example, Figure 1 (right) shows a standard LR fitted using a training sample of 20 univariate instances drawn from Gaussian distributions with the same standard deviation, and the means  $-2$ ,  $2$  for negative and positive classes, respectively. The model is overfitted to the training set with highly overconfident

predictions compared to the optimal LR model given the actual distribution of the dataset.

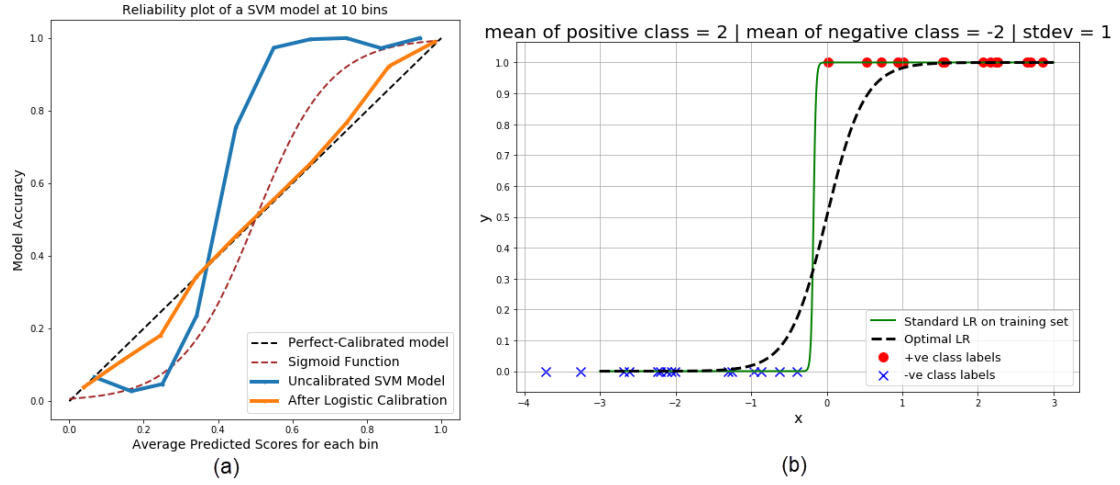


Figure 1. (a) Effect of logistic calibration on an uncalibrated SVM model with distorted scores with a Sigmoid shape. (b) Overconfident standard LR model fitted with a training sample compared with the optimal model on the testing set distribution.

Few approaches have been proposed to reduce the overconfidence issues in LR fitting, and particularly logistic calibration. The most common approach applies an engineering trick for smoothing the dataset target labels before model fitting, which is named as *Platt scaling* [P<sup>+</sup>99]. Applying label smoothing teaches the model not to be very confident of the labels of all instances during training. Although this approach works quite well in some cases, it was shown that the selected smoothing factor by that approach is not the best one in many other cases. Based on these findings, more questions are raised about how to choose better smoothing factors, and whether label smoothing is the best way to reduce overfitting or there exist better alternatives [Pä19].

Label smoothing is not only adopted in LR fitting but also widely populated with *deep neural networks*. It helps the networks to take into account the presence of noisy labeled instances in the training set. This helps in reducing the overconfidence of the network predictions and improves its accuracy.

In this thesis, we try to investigate and discuss previous approaches used in reducing overfitting of LR and propose new ones based on our findings. To this end, the contribution of this thesis work is as follows:

- Study the different approaches used in reducing the overconfidence of the LR model.
- Analytical derivation for the optimal output probability predictions if the exact distribution of the generative model of the dataset is known.
- Demonstration of a new instance-based label smoothing approach using kernel density estimation to reduce overconfidence in LR.

- A new label smoothing parametric method for neural networks enhances the network performance in terms of generalization, calibration, and network distillation.
- Empirical evaluation and discussion of the proposed methods on synthetic and real datasets.

## 1.4 Thesis structure

In the following Section 2, an overview of the types of classifiers, LR model, and classifiers calibration are introduced for the background. In Section 3, we discuss the LR overconfidence issue, and how it was addressed in the literature. In Section 4, we derive how to find the optimal output probability predictions with prior knowledge of the exact generative distribution of the considered dataset. A new approach for label smoothing using kernel density estimation for LR fitting is proposed in Section 5. Next, an instance-based label smoothing approach in neural networks motivated by our work with logistic calibration is developed and presented in Section 6. Section 7 includes the detailed setup, results, and discussions for the experimental evaluation of the proposed methods. Finally, the main conclusions of this study, in addition to the possible future research directions, are addressed in Section 8. Figure 2 shows a summary of the thesis content with an overview of its flow and organization.

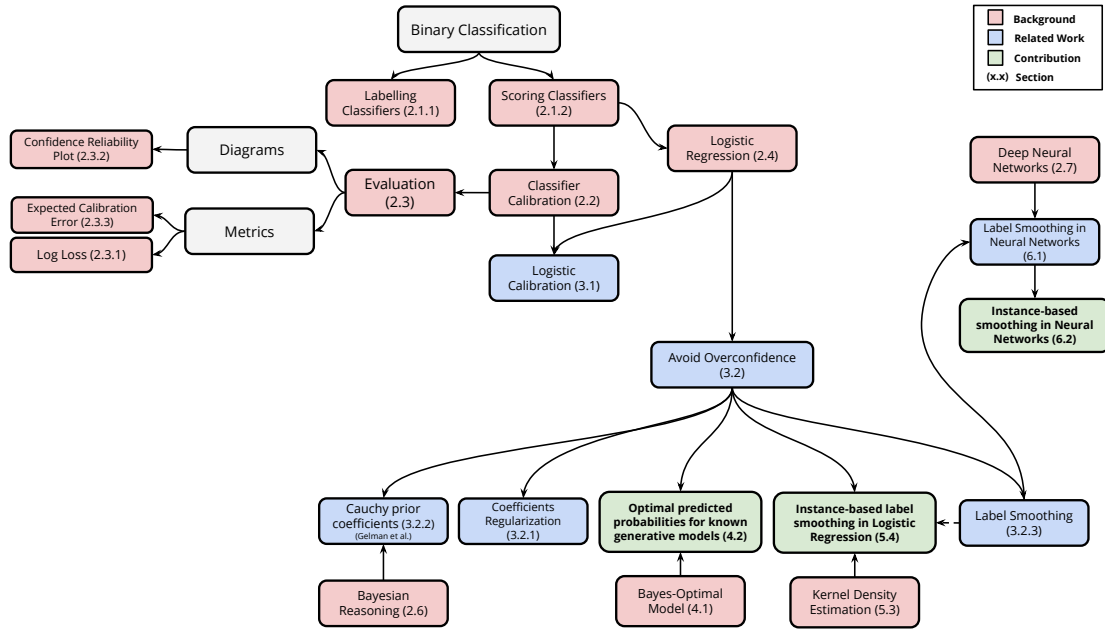


Figure 2. A high-level overview of the thesis flow and organization in the following sections.

## 2 Background

In the following sections, we start by introducing the notation being used in our study, and an overview of the necessary background information, including different classifiers types and the calibration process. Then, we introduce the logistic regression (LR) model and discuss how it is being trained and used in calibrating linear classifiers. After that, Bayesian inference in machine learning is shortly introduced. Finally, we present a brief overview of deep neural networks and examples of their types.

### 2.1 Types of classifiers

A *classifier* is a category of learning algorithms that implements a function to solve a classification task such that it takes an input of several features, and it outputs the predicted class label out of a set of finite discrete classes. Labeling classifiers are the ones that output the final predicted class labels. However, to obtain a confidence level associated with the class prediction, scoring classifiers are needed. Next, we introduce these different types of classifiers, along with their notation.

#### 2.1.1 Labeling classifiers

A labeling classifier is a predictive model function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  that computes the predicted class denoted by  $y_i$  of an instance  $i$ , given its features vector values denoted by  $\mathbf{x}_i$ . The set of all possible values of features is referred to as feature space  $\mathcal{X}$ , and the set of different classes values is denoted by  $\mathcal{Y}$  such that  $\mathbf{x}_i \in \mathcal{X}$ , and  $y_i \in \mathcal{Y}$ . In this study, we focus on the **binary classification** tasks where  $\mathcal{Y} = \{0, 1\}$  such that 0 represents a negative class, and 1 represents a positive class. A dataset  $D = D^+ \cup D^-$  is a collection of positive class instances  $D^+ = \{(x_i, y_i) \mid i \in \{1, 2, \dots, n\}, y_i = 1, x_i \in \mathcal{X}\}$  and negative class instances  $D^- = \{(x_i, y_i) \mid i \in \{1, 2, \dots, n\}, y_i = 0, x_i \in \mathcal{X}\}$  where  $n = |D|$  is the total number of instances in the dataset  $D$  [Fla12].

#### 2.1.2 Scoring classifiers

A scoring classifier  $\hat{\mathbf{S}} : \mathcal{X} \rightarrow \mathbb{R}^K$  is a function which performs a mapping from the instance space  $\mathcal{X}$  to a  $K$ -dimensional vector of real numbers  $\hat{\mathbf{S}}(\mathbf{x}) = (\hat{s}_0(\mathbf{x}), \dots, \hat{s}_{K-1}(\mathbf{x}))$  representing how likely that the instance  $\mathbf{x}$  belongs to the different possible classes, where  $K$  is the total number of classes in  $\mathcal{Y}$ . Most of the learning classifier algorithms can be used to output scores instead of labels only. For example, the *nearest neighbors* algorithm returns the ratio of positive neighbors to the total number of neighbors, and *support vector machines* returns the signed distance of an instance away from the decision boundary between classes [Fla12].

A *probabilistic* classifier  $\hat{\mathbf{P}} : \mathcal{X} \rightarrow [0, 1]^K$  is a special case of the scoring classifiers such that the output scores represent probabilities that the instance  $\mathbf{x}$  belongs to the different classes. The predicted probabilities vector  $\hat{\mathbf{P}}(\mathbf{x})$  should add up to 1,  $\sum_{k=0}^{K-1} \hat{P}_k(\mathbf{x}) = 1$ . In binary classification tasks ( $K = 2$ ), the probability vector  $\hat{\mathbf{P}}(\mathbf{x}) = (\hat{P}_0(\mathbf{x}), \hat{P}_1(\mathbf{x}))$  shows how likely that the instance

$\mathbf{x}$  belongs to the negative, and positive classes respectively. The probability that  $\mathbf{x}$  belongs to the positive class is shortly denoted as  $\hat{P}$  where  $\hat{P} = \hat{P}_1 = 1 - \hat{P}_0$ .

Following a *decision rule*, scoring classifiers can predict the final predicted class label  $\hat{y}$  using a threshold  $t$ . For example, an instance  $\mathbf{x}$  is labelled as positive when  $\hat{P} > t$ , and negative otherwise [HOFR11]. Scoring classifiers can be easily converted into probabilistic classifiers by ensuring that their scores are positive, then normalizing each class's score by the total summation of output scores. However, these derived class probabilities are usually termed as *uncalibrated* scores.

## 2.2 Classifier calibration

A probabilistic classifier must have an output of the estimated classes' probabilities. The more the predicted class probability goes into an extreme (i.e., 0 or 1), the more confident the model is, and vice versa. These probabilities are not usually calibrated, which means that they do not reflect the model's true uncertainty levels. Hence, there is a need for confidence calibration of classifiers to make their predictions more reliable [CG04].

The calibration process maps the model's output of estimated probabilities into new values in a way that makes the confidence of these new calibrated predictions match the actual class distribution as much as possible. For instance, 90 out of 100 patients were accurately diagnosed as infected with a disease by a perfect calibrated model with a probability of  $\hat{P}_1 = 0.9$  for every patient. The confidence levels of the perfectly calibrated model match exactly with the accuracy of its predictions during inference. So, it is almost impractical to achieve perfect calibration in the test phase. The decision rule threshold  $t$  used to predict the final class label should be set to  $\frac{1}{2}$  for a well-calibrated probabilistic model of a binary classification task, assuming equal class distribution and misclassification costs. Thus, an instance  $\mathbf{x}$  is classified as positive when  $\hat{P}_1 > \hat{P}_0$  (i.e. positive class probability > negative class probability). This can be simply derived as follows:

$$\begin{aligned}\hat{P}_1 &> \hat{P}_0 \\ \hat{P}_1 &> 1 - \hat{P}_1 \\ 2\hat{P}_1 &> 1 \\ \hat{P}_1 &> \frac{1}{2}.\end{aligned}$$

### 2.2.1 Calibration of linear classifiers scores

Calibration of linear classifiers scores, is the process of converting model output scores represented by the signed distance from the decision boundary into calibrated probabilities. The geometric interpretation behind these scores can be shown in Figure 3. The output of any linear model  $s$  can be formulated as  $\hat{s}(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x}_i - t$ , where  $t$  is the perpendicular distance from the origin to the decision boundary. The dot product of an instance  $\mathbf{x}_i$ , and coefficient vector  $\mathbf{w}$  of the decision boundary represents the projection of  $\mathbf{x}_i$  on  $\mathbf{w}$ . Using a threshold  $t = 0$ , the final class label is determined based on the sign of  $\hat{s}(\mathbf{x}_i)$ .

Having sufficiently many instances as identically distributed independent random variables in both classes, the central limit theorem [LC86] can be used to show that it is reasonable, and

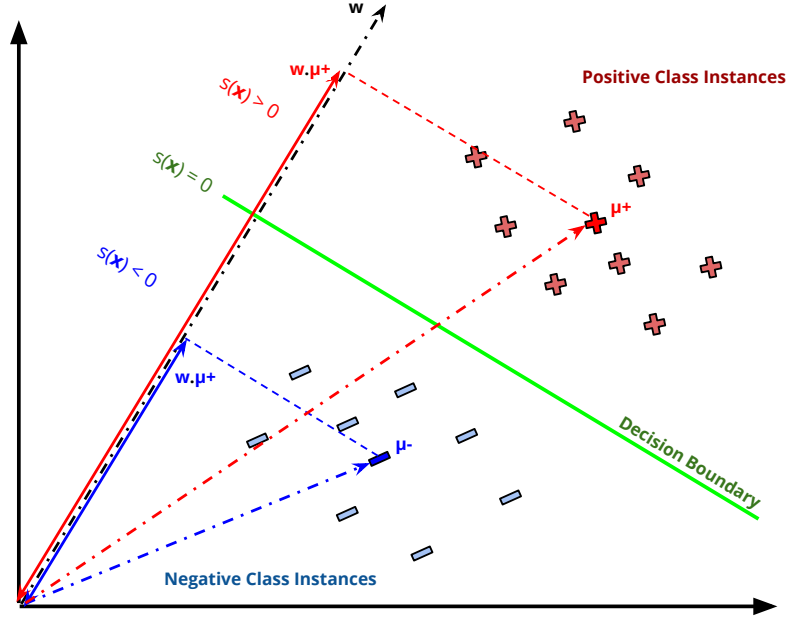


Figure 3. Geometric interpretation of linear classifiers predicted scores.

often not far from the truth to assume that the distances of each class instances are approximately normally distributed with a mean of  $\bar{s}^+ = \mathbf{w} \cdot \boldsymbol{\mu}^+ - t$ , or  $\bar{s}^- = \mathbf{w} \cdot \boldsymbol{\mu}^- - t$  for the positive and negative classes respectively, where  $\boldsymbol{\mu}^+ = \frac{1}{|D^+|} \sum_{i, y_i=1}^{D^+} \mathbf{x}_i$ , and  $\boldsymbol{\mu}^- = \frac{1}{|D^-|} \sum_{i, y_i=0}^{D^-} \mathbf{x}_i$  [Fla12]. Hence, the density function of the distance scores can be written as follows, assuming that the normal distributions for both classes have the same standard deviation  $\sigma$ :

$$\begin{aligned} p(S = s(\mathbf{x})|y = 1) &= \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-(s - \bar{s}^+)^2}{2\sigma^2}, \\ p(S = s(\mathbf{x})|y = 0) &= \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-(s - \bar{s}^-)^2}{2\sigma^2}. \end{aligned} \quad (1)$$

Using the previously concluded density functions, we can predict probabilities instead of scores using Bayes' rule as follows:

$$\begin{aligned} P(y = 1|S = s(\mathbf{x})) &= \frac{p(S=s(\mathbf{x})|y=1)P(y=1)}{p(S=s(\mathbf{x})|y=1)P(y=1) + p(S=s(\mathbf{x})|y=0)P(y=0)} \\ &= \frac{1}{1 + \frac{P(y=0)}{P(y=1)} \cdot \frac{p(S=s(\mathbf{x})|y=0)}{p(S=s(\mathbf{x})|y=1)}}. \end{aligned} \quad (2)$$

Assuming that the probabilities of both classes are equal,  $P(y = 1) = P(y = 0)$ , and

substituting (1) in (2):

$$\begin{aligned}
\frac{p(S=s(\mathbf{x})|y=0)}{p(S=s(\mathbf{x})|y=1)} &= \exp \left( \frac{(s(\mathbf{x})-\bar{s}^+)^2 - (s(\mathbf{x})-\bar{s}^-)^2}{2\sigma^2} \right) \\
&= \exp \left( \frac{(\bar{s}^- - \bar{s}^+)}{\sigma^2} s(\mathbf{x}) + \frac{(\bar{s}^{+2} - \bar{s}^{-2})}{2\sigma^2} \right) = \exp(-a s(\mathbf{x}) + b) \\
\Rightarrow P(y=1|s(\mathbf{x})) &= \frac{1}{1 + \exp(-(a s(\mathbf{x}) + b))}, \text{ where } a = \frac{(\bar{s}^+ - \bar{s}^-)}{\sigma^2}, b = \frac{(\bar{s}^{-2} - \bar{s}^{+2})}{2\sigma^2}.
\end{aligned} \tag{3}$$

Finally, from Equation 3, it is shown that converting distance scores into probabilities can be done by mapping all the scores using a sigmoid function (a.k.a logistic function) with a slope of  $a$ , and shift of  $b$ . Examples of sigmoid functions with different slopes can be viewed in Figure 4.

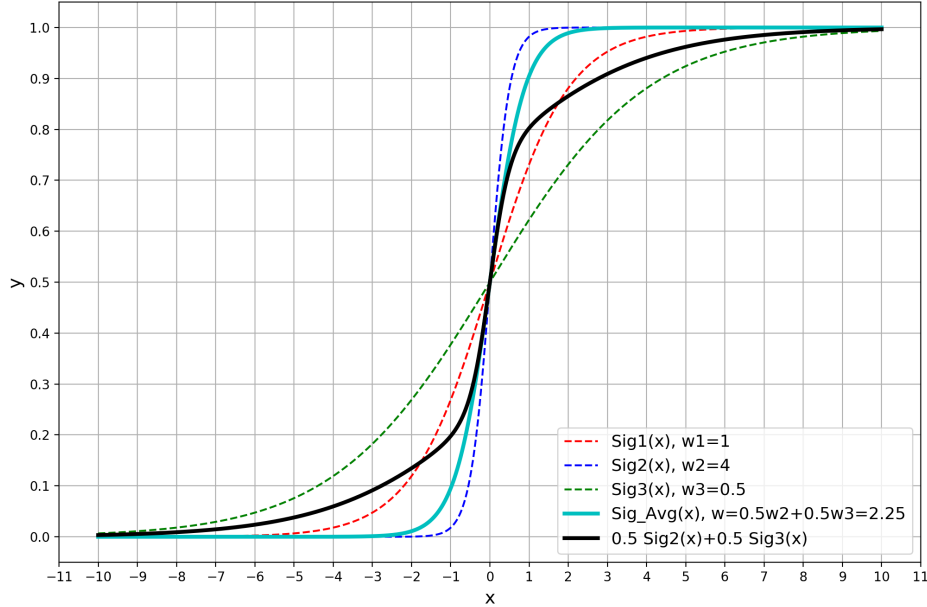


Figure 4. Univariate sigmoid functions with variable slope, and zero shift (red, green, blue); sigmoid with weighted average **slope coefficients** of other two sigmoids (cyan); the weighted average of the outputs of two sigmoids functions **output values** (black).

## 2.3 Evaluation metrics

Next, we discuss different visualization and quantitative evaluation methods of the model over-confidence and calibration error.

### 2.3.1 Log Loss

The log loss is a prevalent loss function for measuring the quality of probabilistic models (a.k.a cross-entropy, or negative log-likelihood loss) [HTF09]. It is considered as a special form of Kullback-Leibler divergence (KLD) [KL51]. KLD computes the relative entropy between two probability distributions as a measurement of how they diverge from each other as shown in the following equation where  $\mathbf{y}$  and  $\hat{\mathbf{P}}$  are the target and predicted probability distributions respectively, and  $K$  is the number of classes:

$$L_{KLD}(\mathbf{y}, \hat{\mathbf{P}}) = \sum_{k=1}^K y_k \log \frac{y_k}{\hat{P}_k}.$$

Log loss in multi-classification tasks is a special case of KLD when the target class distribution is 1 for the true class label, and 0 for all other classes. Thus, log loss can be written as follows:

$$L_{log}(\mathbf{y}, \hat{\mathbf{P}}) = - \sum_{k=1}^K y_k \log(\hat{P}_k).$$

In binary classification tasks, this formula can be reduced as follows, where  $y_1, \hat{P}_1$  are the target, and predicted probabilities of the positive class, respectively:

$$L_{log}(\mathbf{y}, \hat{\mathbf{P}}) = -y_1 \log(\hat{P}_1) - y_0 \log(\hat{P}_0) = -y_1 \log(\hat{P}_1) - (1 - y_1) \log(1 - \hat{P}_1).$$

To get the intuition behind how the log loss penalizes prediction errors, Figure 5 illustrates the loss value against different predicted probabilities for the correct class label. This loss value reaches its minimum at a predicted probability of 1.0 (matching target probability), representing correct and confident prediction. However, it starts to increase as we go far from that value gradually until suddenly, the loss *overgrows* for a predicted probability ranging from 0.5 to 0.0, where the loss tends to  $\infty$ . Hence, log loss measures the uncertainty of the predicted probabilities compared with the correct labels. The loss function gives more penalty for *wrong and confident* predictions than less confident ones.

### 2.3.2 Confidence reliability plot

The confidence reliability plot is a visualization method for the model calibration error in binary classifiers [NMC05]. It shows the relationship between the average output estimated probability (model confidence) on the x-axis, and the corresponding model accuracy on the y-axis. Figure 6 shows an example of a confidence reliability diagram for an uncalibrated model with error bars in red between the model calibration line, and a perfect-calibrated model (diagonal line).



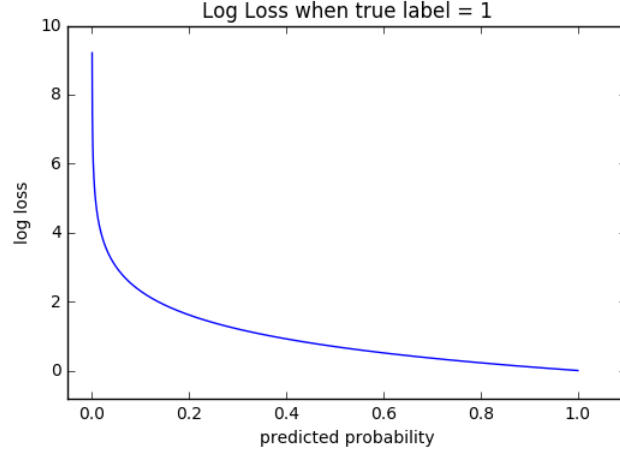


Figure 5. Cross-entropy value at different predicted probabilities of true class.

To draw the confidence reliability plot, the possible output probabilities  $[0, 1]$  are binned into equally spaced  $N$  bins such that a bin  $b_n$  has a prediction probability lies into an interval  $I_n = (\frac{n-1}{N}, \frac{n}{N}]$ ,  $n \in 1, 2, \dots, N$ . Then, the average of the model output estimated probabilities located inside each bin  $confidence(b_n)$  is plotted on the x-axis against the proportion of accurately classified instances  $accuracy(b_n)$  inside the same bin on the y-axis. Thus, each bin  $b_n$  is represented by a bar with a height equal to  $accuracy(b_n)$ . The model calibration curve is represented by the connection between these plotted points together:

$$confidence(b_n) = \frac{1}{|b_n|} \sum_{i \in b_n} \hat{P}_i$$

$$accuracy(b_n) = \frac{1}{|b_n|} \sum_{i \in b_n} 1(y_i = \hat{y}_i)$$

where  $1(a = b)$  equals 1 when  $a = b$ , and 0 otherwise.

A perfect calibrated model is characterized by having equal confidence and accuracy values for all bins,  $confidence(b_n) = accuracy(b_n) | n \in N$ . A model is usually said to be calibrated when its output predictions on the testing set are calibrated. Hence, it is almost impractical to have a perfect-calibrated model as it is difficult to learn the exact testing set distribution from a sample of instances of the training set.

### 2.3.3 Expected calibration error

Since the reliability plot shows only a visualization for confidence behavior, it still lacks a numerical representation to ease the evaluation of different calibration approaches. Additionally, there is no information related to the number of instances inside each bin in the reliability plot. For instance, the model may have most of its probability estimations lying inside one or two bins that are not calibrated like the other bins. Consequently, this can mistakenly lead to some wrong conclusions about the overall calibration behavior of the model.

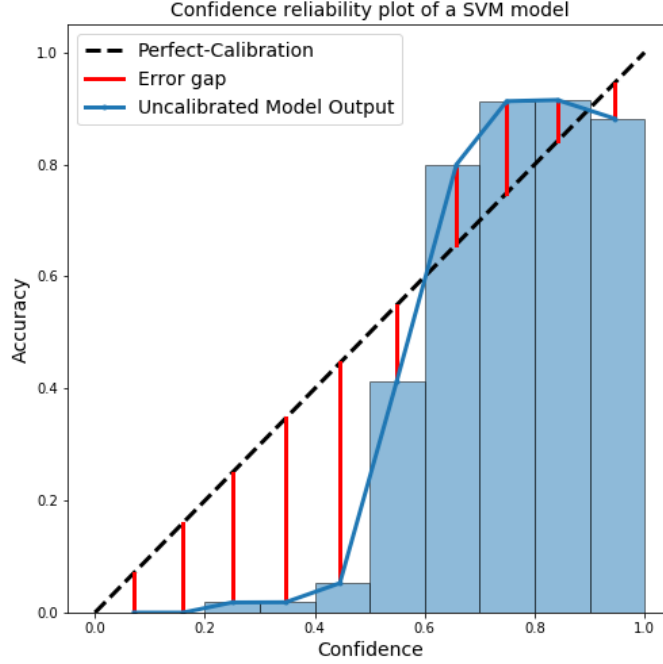


Figure 6. An example of the confidence reliability diagram for an uncalibrated SVM model trained on a randomly generated 2-class dataset using Python Scikit-learn `make_classification` function<sup>1</sup>.

The Expected Calibration Error (ECE) is a measure for the overall expected error between the model accuracy and confidence across all the bins. In the following equation, the ECE is calculated as the weighted average of the calibration error which takes into account the number of instances inside each bin too [NCH15]:

$$ECE = \sum_{n=1}^N \left( \frac{|b_n|}{\sum_{j=1}^N |b_j|} \times |accuracy(b_n) - confidence(b_n)| \right).$$

## 2.4 Logistic regression

Logistic calibration is one of the widely used calibration methods for binary scoring classifiers. In logistic calibration, a univariate logistic regression (LR) model is fitted to convert the output scores of the uncalibrated classifiers into better-calibrated probabilities. Here we introduce the LR model, how it is being fitted to a dataset, and discuss its use cases.

<sup>1</sup>[https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make\\_classification.html](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html)

LR is one of the commonly used linear classification models that finds the best fitting *logistic* curve (Figure 4) on binary class datasets. In contrast with the other linear classifiers which require further post-processing step for calibrating their predictions, LR has the advantage that, on average, its estimated probabilities are calibrated on the training set as an integral part of the model fitting [KDG<sup>+</sup>02]. The LR model is given by the following equation, where  $\mathbf{w}$  represents the coefficients vector for every feature dimension in the features vector  $\mathbf{x}$ :

$$\hat{P}(\mathbf{x}) = \frac{e^{\mathbf{w} \cdot \mathbf{x} - t}}{e^{\mathbf{w} \cdot \mathbf{x} - t} + 1} = \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} - t)}}.$$

In case of univariate tasks (i.e.  $\mathbf{x} = [x]$ ,  $\mathbf{w} = [w]$ ),  $w$  is termed as the slope of the sigmoid function.

Given that the class labels are 0, and 1 for the negative, and positive classes respectively, the conditional likelihood probability of the class label given a particular instance  $i$  can be defined as a Bernoulli distribution as follows:

$$P(Y = y_i | \mathbf{X} = \mathbf{x}_i) = \hat{P}(\mathbf{x}_i)^{y_i} (1 - \hat{P}(\mathbf{x}_i))^{1-y_i}.$$

Then, the model is fitted by maximizing the conditional log-likelihood function over all the training dataset instances as follows:

$$\begin{aligned} \mathbf{w}, t &= \underset{\mathbf{w}, t}{\operatorname{argmax}} \prod_i P(y_i | \mathbf{x}_i) = \underset{\mathbf{w}, t}{\operatorname{argmax}} \prod_i \left( \hat{P}(\mathbf{x}_i)^{y_i} (1 - \hat{P}(\mathbf{x}_i))^{1-y_i} \right) \\ &= \underset{\mathbf{w}, t}{\operatorname{argmax}} \log \left( \prod_i \left( \hat{P}(\mathbf{x}_i)^{y_i} (1 - \hat{P}(\mathbf{x}_i))^{1-y_i} \right) \right) \\ &= \underset{\mathbf{w}, t}{\operatorname{argmax}} \sum_i \left( y_i \log \hat{P}(\mathbf{x}_i) + (1 - y_i) \log(1 - \hat{P}(\mathbf{x}_i)) \right) \\ &= \underset{\mathbf{w}, t}{\operatorname{argmax}} \sum_{i, y_i=1} \log \hat{P}(\mathbf{x}_i) + \sum_{i, y_i=0} \log(1 - \hat{P}(\mathbf{x}_i)) \\ &= \underset{\mathbf{w}, t}{\operatorname{argmin}} \sum_{i, y_i=1} -\log \hat{P}(\mathbf{x}_i) + \sum_{i, y_i=0} -\log(1 - \hat{P}(\mathbf{x}_i)). \end{aligned}$$

Maximizing the log-conditional likelihood for the model parameters  $\mathbf{w}$ , and threshold  $t$  can be shown to be a convex optimization problem, which means that their partial derivatives should be set to zero as shown in the following equations:

$$\begin{aligned} \frac{\partial}{\partial t} \left( \sum_{i, y_i=1} \log \hat{P}(\mathbf{x}_i) + \sum_{i, y_i=0} \log(1 - \hat{P}(\mathbf{x}_i)) \right) &= 0 \\ \nabla_{\mathbf{w}} \left( \sum_{i, y_i=1} \log \hat{P}(\mathbf{x}_i) + \sum_{i, y_i=0} \log(1 - \hat{P}(\mathbf{x}_i)) \right) &= 0. \end{aligned}$$

However, these equations can not be solved analytically, and require numerical optimization algorithms to find the optimal point. Hence, solving the previous equation for  $t$ :

$$\begin{array}{l|l}
\text{If the target class is positive } y_i = 1: & \text{If the target class is negative } y_i = 0: \\
\frac{\partial}{\partial t} \log \hat{P}(\mathbf{x}) = -1 - \frac{\partial}{\partial t} \log(e^{\mathbf{w} \cdot \mathbf{x} - t} + 1) & \frac{\partial}{\partial t} \log(1 - \hat{P}(\mathbf{x})) = \frac{\partial}{\partial t} - \log(e^{\mathbf{w} \cdot \mathbf{x} - t} + 1) \\
= -1 - \frac{1}{e^{\mathbf{w} \cdot \mathbf{x} - t} + 1} e^{\mathbf{w} \cdot \mathbf{x} - t} (-1) & = \frac{-1}{e^{\mathbf{w} \cdot \mathbf{x} - t} + 1} e^{\mathbf{w} \cdot \mathbf{x} - t} (-1) \\
= \hat{P}(\mathbf{x}) - 1. & = \hat{P}(\mathbf{x}). \\
\Rightarrow \frac{\partial}{\partial t} \left( \sum_{i, y_i=1} \log \hat{P}(\mathbf{x}_i) + \sum_{i, y_i=0} \log(1 - \hat{P}(\mathbf{x}_i)) \right) = \sum_{i, y_i=1} (\hat{P}(\mathbf{x}) - 1) + \sum_{i, y_i=0} \hat{P}(\mathbf{x}) & (4) \\
= \sum_i (\hat{P}(\mathbf{x}) - y_i) = 0. &
\end{array}$$

Finally, Equation 4 shows that the solution for this partial derivative satisfies that the average of the predicted estimated probabilities  $\sum_i \hat{P}(\mathbf{x}_i)$  is equal to the ratio of positive class instances  $\frac{\sum_i y_i}{n}$ , where  $n$  is the size of the dataset. This condition is a necessary but not sufficient one for the classifier models to be calibrated on the training set, as discussed in Section 2.2.

## 2.5 Bayesian reasoning in machine learning

In this study, a Bayesian LR model is discussed as an approach to reduce the overconfidence of the LR predictions [GJP<sup>+</sup>08]. Additionally, we use a Bayesian approach to derive the optimal probability predictions of binary classification datasets given the exact distribution of their generative models in Section 4. Thus, next, we explain the concept of Bayesian reasoning, how it is being used in machine learning, and compare it with the frequentist approach.

The Bayesian reasoning approach uses statistics in a way that refers to the future and is considered as a measure for a prior belief. The main critique of Bayesian reasoning is that there is no single method for choosing the prior. So, different people will come up with different priors subjectively. Consequently, they may arrive at different posteriors and different conclusions. Hence, in the following Bayes rule (Equation 5), the Bayesian reasoning should start with a prior  $p(A)$ . Then, the collected data is used to represent the likelihood  $p(B|A)$  of such events to happen, and it is used in updating the prior belief to produce the posterior information  $p(A|B)$ . This process is repeated whenever new data is obtained.

$$p(A|B) = \frac{p(B|A) \cdot p(A)}{p(B)}. \quad (5)$$

In machine learning, Bayesian reasoning is used to infer the model coefficients ( $A = \mathbf{w}$ ) from the data ( $B = \text{Data}$ ). Substituting back into Bayes rule (Equation 5), the model coefficients are interpreted as the prior information that, in many cases, we have beliefs about what values those coefficients could be. The likelihood probability  $P(\text{Data}|\mathbf{w})$  represents how likely that the current data could be obtained given the model parameters. The posterior probability  $P(\mathbf{w}|\text{Data})$  is the probability distribution over different model coefficients that we end up with, taking into account both the prior beliefs and data. Later, the coefficients of the maximum estimated posterior probability could be used for the model. This approach is called *maximum a posteriori estimation*.

(MAP). The strengths of Bayesian reasoning in machine learning can be viewed in embedding the domain knowledge into the learning process in the form of the prior. However, a huge amount of data is needed to learn the true model if a wrong prior was mistakenly chosen. Consequently, as a rule of thumb, priors are usually set to include a wide range of values to avoid such cases [Bar12]. Using the Bayesian reasoning in fitting the LR by defining a proper prior distribution for the model coefficients could act as a *regularization* mechanism that reduces overconfidence problem [GJP<sup>+</sup>08].

On the other hand, the *frequentist* approach follows a more objective paradigm by referring to the data we have from past events without any dependence on prior beliefs [BB04]. In classical machine learning, the *maximum likelihood estimation (MLE)* is usually followed, where the likelihood probability is used only to evaluate different models. Therefore, the model with a higher likelihood of the data is considered the best one without taking any prior information into account. When a uniform prior distribution is used over possible values of model coefficients, the MLE is equivalent to the MAP.

## 2.6 Deep neural networks

Although this thesis is focusing on using label smoothing to reduce the model overconfidence in binary classification tasks, label smoothing is also widely used with deep neural networks (DNNs). Besides, DNNs have achieved outstanding performance in many fields like image classification, image segmentation, and neural machine translation with a vast amount of research in the last few years [Den15]. Hence, it would be interesting to apply our findings to enhance the standard label smoothing approach in DNNs and overcome its shortcomings. Next, we introduce DNNs, discuss their most important applications, and how their predictions are calibrated.

DNNs are an important branch of machine learning, which targets the problem of automatic learning of good representation for the input data. The smallest building unit of a neural network is called a neuron, which takes different inputs, adds them up with different weights, and passes the output to other nodes. These neurons are grouped into layers based on the network architecture, and their weights are learned during the training process. Simple representations can be obtained from the first layers of the network, while the complexity of these representations increases as it goes through the deeper layers. A wide range of network types and thousands of architectures have been proposed during the last few years like convolutional neural networks that are suitable for computer vision tasks, and recurrent neural networks for time-series analysis [GBC16].

Convolutional neural networks (CNNs) are one type of network architectures that use a grid-like topology of sequential filters for image processing. Each filter is considered as a 2-dimensional grid of pixels that slide over the input image spatially and compute dot-products with pixel values, and added together to construct the feature maps [L<sup>+</sup>89]. During the past decade, several advancements have occurred in CNN architectures which lead to a substantial improvement in image classification accuracy that even outperforms humans like CNNs with residual connections (ResNet) [HZRS16], CNNs with dense blocks that take identity mapping as input from all the previous layers (DenseNet) [HLVDMW17], and CNNs with inception modules that approximates a sparse CNN with a dense construction, and uses residual connections in addition to batch normalization layers (InceptionV4) [SIVA17]

Although DNNs are necessary to achieve small error rates over a wide range of datasets, it has been shown that deeper networks suffer from significant calibration error [KL15]. Therefore, calibrating neural network outputs is one of the hot research topics in the machine learning field with a long recent list of contributions [GPSW17].

### 2.6.1 Temperature scaling

One of the most popular calibration techniques in neural networks is temperature scaling, which was found to help reduce the overconfident probability predictions of the neural networks and make them more reliable. A learned scalar parameter by a neural network  $T$  is used to divide the logits (inputs to the softmax function of a classification network which is used to predict output probabilities of different classes), as shown in Figure 7. This scaling parameter is learned after training the network by minimizing the cross-entropy. During inference, this learned parameter is used directly as a post-processing step to scale the logit values [GPSW17].

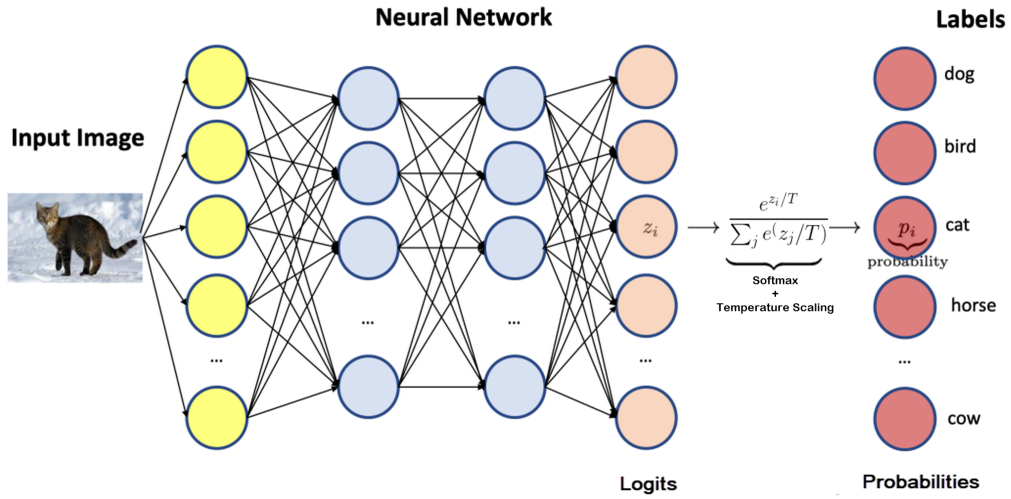


Figure 7. Training a classification neural network with temperature scaling where a parameter  $T$  is learned by the network and used to divide the logits values before mapping them into probabilities using a softmax layer.

### 3 Related work

In this section, we introduce the previous literature work that used a logistic regression model in the calibration of binary classifiers. Also, the overfitting problem in logistic regression has been discussed. Finally, we present other related work and different methodologies for reducing this problem.

#### 3.1 Using logistic regression in classifiers calibration

Platt [P<sup>+</sup>99] has proposed using the logistic regression model as a post-processing step on another linear support vector machine (SVM) model to convert its predictions into calibrated class probabilities. This process is known by logistic or sigmoid calibration. The logistic calibration has been widely used and implemented in the most popular machine learning tools, and frameworks like *Scikit-Learn* in Python<sup>2</sup>. Figure 8 illustrates the logistic calibration method. After fitting an SVM model on a training set, the SVM model is used to predict the scores on a separate validation set. These scores are considered as the input features to the logistic regression model, and the target variable is the true class labels corresponding to the input features. Finally, the final model to be used during the inference stage consists of a sequential pipeline of the SVM model to get the scores, and the logistic regression model to convert these scores into calibrated probabilities.

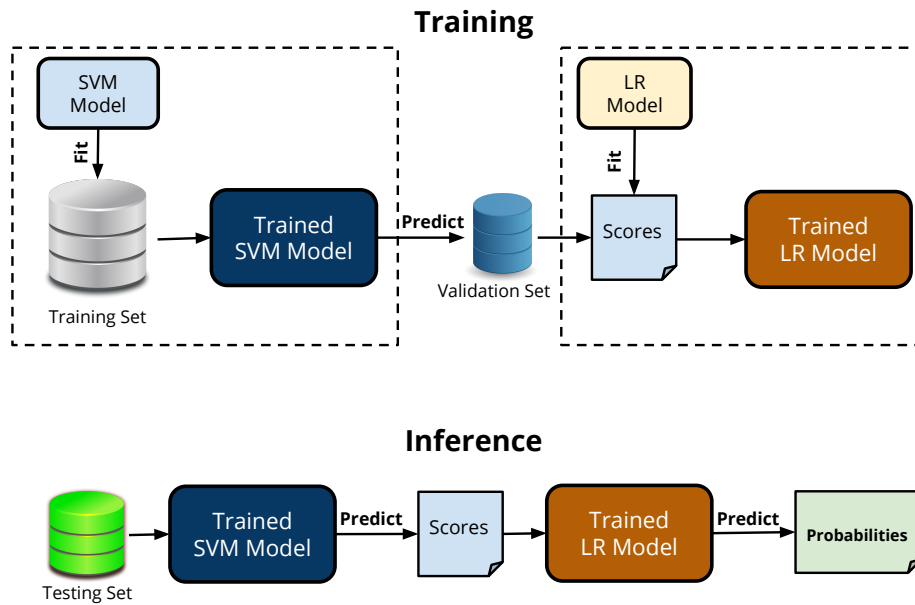


Figure 8. Logistic calibration process during training, and inference.

---

<sup>2</sup><https://scikit-learn.org/stable/>

Logistic calibration can be applied over many classifiers as well. For example, naive Bayesian classifier often predicts uncalibrated and overconfident probabilities due to the presence of dependent features in the dataset, which contradicts with the assumption made by naive-Bayes that all features should be conditionally independent of each other given the class [Fla12]. In that case, logistic calibration can help in calibrating the output estimated probabilities of the naive-Bayes model. Moreover, tree-based ensemble classifiers like boosted trees and bagging trees are susceptible to have uncalibrated estimated probabilities that do not reflect proper confidence levels. The base-level trees constructing these ensembles usually have different predictions due to the random instance or feature subsetting made while fitting them. Consequently, the variance in the trees' predictions will push the ensemble model estimated probabilities far away from the full confidence levels (0 and 1) [NMC05].

There exist other calibration methods like isotonic calibration, which is a non-parametric calibration method [ZE02, AK], and Beta Calibration that is characterized by more flexibility in the calibration curve shape than logistic sigmoid [KSFF17]. However, the former is susceptible to overfitting, especially with large numbers of instances or under-fitting with low numbers of instances. The latter has more parameters to tune than logistic calibration. Thus, logistic calibration is still a widely used solution for classifier calibration.

## 3.2 Solutions to overfitting in logistic regression

Using machine learning to learn a mapping function like what is done in case of score calibration is not an easy task, and often susceptible to overfitting problems such that the learner produces excellent results on the training data, but worse results on the test data [Fla12]. The overfitting problem has always been a hot research topic as it is considered as the thorn in a throat for machine learning. This problem can happen for different reasons like:

1. Using too complex models or a large number of coefficients that makes the model to learn a very strict decision boundary tailored to separate between training instances. However, it loses its generality on new data [Haw04]. It has always been a rule-of-thumb that the fewer coefficients to be used in the learner algorithm to do the same task, the better the generalization ability achieved by the model.
2. The minimum value of loss function achieved during a model fitting on the training data does not imply that the best model is obtained on the test data [Bab04]. For example, the logistic regression is trained to minimize the cross-entropy loss on the training set, but having not sufficient representative training instances, or some noisy labels could ruin the model performance on new the test data.

An example of the consequences of the overfitting problem in LR is the *separation* problem that occurs if the target variable separates one of the feature variables or a combination of them [AA84]. For example, suppose that for all the observed instances in a dataset, the class label is always positive when a particular continuous feature  $x_j > 0$ , and it is always negative when  $x_j \leq 0$ . During the MLE fitting of the LR, the slope of the fitted sigmoid curve will tend to  $\infty$ , as shown in Figure 9. Having such cases implies overconfidence of the model output probability



predictions, whatever the values of the input features. Any wrong predictions will be very costly in terms of the log loss as the model becomes overconfident of its predictions, and no longer calibrated [ZZ19].

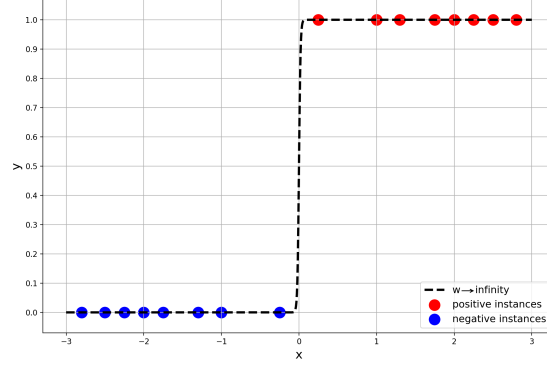


Figure 9. LR fitted curve in case of full separation.

### 3.2.1 Regularization

Regularization is one of the most common approaches to reduce overfitting. It is usually done by adding a penalizing term to the magnitude of the model coefficients inside the loss function to be minimized during the training process. Hence, minimizing the values of these coefficients will reduce the likelihood that the model learns a too complex function. In Figure 9, we show how an infinite slope for a logistic sigmoid can lead to an overconfident model in case of separation. Adding the norm of the slope to the loss function to be minimized will avoid having large slopes. Consequently, the model tends to reduce its overfitting on the training data, and start to have a good generalization behavior. The two most common regularization methods applied in logistic regression are the  $L1$  and  $L2$  regularization where a regularization constant  $\lambda$  is multiplied by the norm of the coefficients vector, or the norm squared, respectively. Then, this term is added to the loss function [Fla12] as shown in the following equations:

$$L1 \text{ Regularization: } \arg \min_{\mathbf{w}, t} \sum_{i, y_i=1} -\log \hat{P}(\mathbf{x}_i) + \sum_{i, y_i=0} -\log(1 - \hat{P}(\mathbf{x}_i)) + \lambda \sum_j^{|\mathbf{w}|} |w_j|$$

$$L2 \text{ Regularization: } \arg \min_{\mathbf{w}, t} \sum_{i, y_i=1} -\log \hat{P}(\mathbf{x}_i) + \sum_{i, y_i=0} -\log(1 - \hat{P}(\mathbf{x}_i)) + \lambda \sum_j^{|\mathbf{w}|} w_j^2.$$

Adding the  $L1$  regularization term makes the optimization problem of LR fitting more computationally expensive to solve as it is no longer differentiable [LLAN06]. However,  $L2$  regularization could be more precise to converge to a global minimum point as the loss objective function is still differentiable. The main benefit of  $L1$  is its ability to push features' coefficients to 0, acting as a method for feature selection [Ng04]. Selecting the regularization parameter,  $\lambda$  could be tricky. On

the one hand, a large regularization parameter will make the learned coefficients too small. This is due to the higher penalty made on the regularized term than the main objective function term. This could lead to an under-fitted model. On the other hand, a small regularization parameter might ignore the regularization term, and have a negligible effect on the learned coefficients. So, it is always required to tune for a good regularization parameter while fitting the models [Fla12].

### 3.2.2 Bayesian logistic regression

Following the Bayesian reasoning in machine learning (Section 2.5), setting a prior to the values of model coefficients can act as a regularization method. Several research studies have tried fitting a Bayesian version of the logistic regression with maximum a posteriori instead of maximum likelihood estimation [GJP<sup>+</sup>08]. The question that has always been investigated in all these studies was what is the proper prior that covers most of the space of the true models' coefficients, and acts as a good regularization to overfitting problems.

Some research effort has been made on choosing the default prior distribution of the coefficients. These work either include methods of fully informative prior distributions using information obtained from domain-specific knowledge, or non-informative prior distributions that can generally work in a wide range of applications. An example for the former is [HBJ<sup>+</sup>14], where the authors proposed using g-prior distributions when the domain expert is confident about his overall information of the whole population distribution. An example of the later is the double exponential distribution (a.k.a Laplace distribution) of the model coefficients. Since the posterior mode estimates of Laplace distribution can reach zero, this prior selection was appropriate for text categorization tasks studied in their work, where there are many collected but less relevant features [GLM07].

Another example of the non-informative Bayesian models is the proposed work by Gelman *et al.* [GJP<sup>+</sup>08]. In this work, a student-t prior distribution with a 0 mean was chosen to force the model coefficients to be located in a reasonable range. It was found empirically that a Cauchy distribution with center 0 and scale 2.5, which is a longer-tailed version of the student-t distribution outperforms other prior distributions like Laplace and Gaussian.

Although informed prior Bayesian reasoning achieved many successes in various tasks, it is always difficult for the domain expert to make sure of reasonable prior estimations for the model coefficients. It is more reasonable that the domain knowledge can be reflected through the prior dataset distribution instead of learning algorithm coefficients. Also, even if an excellent prior was chosen, it is still questionable whether there could be a better selection.

### 3.2.3 Label smoothing, and Platt scaling

If some class labels in the dataset are noisy, this will eventually influence the model learning heavily. Thus, some uncertainty could be added to these noisy labels to smoothen their values. So, instead of setting the target class  $y$  to 0, 1 for the negative and positive classes, the new target label values are set to  $0 + \epsilon^-$ , and  $1 - \epsilon^+$ . The variables  $\epsilon^-$ , and  $\epsilon^+$  are called smoothing factors [GBC16]. Label smoothing has been applied with various deep learning [MKH19] and machine learning models [AW18], and it achieves good performance in many tasks.

The smoothing factors can be interpreted as teaching the model that there is a small error chance with value  $\epsilon$  in the label of a particular instance. There is a possibility that this label is noisy and it should be the opposite class. Consequently, the overconfidence in the model predictions is reduced as the confidence value that the model is trying to learn during its training is a little bit less than the full confidence of 0 or 1 for negative and positive classes, respectively, as shown in Figure 10.

Platt used specific smoothing factors in his previous work [P<sup>+</sup>99], defined as follows:

$$\begin{aligned}\epsilon_{platt}^+ &= \frac{1}{|D^+|+2} \\ \epsilon_{platt}^- &= \frac{1}{|D^-|+2}.\end{aligned}\tag{6}$$

The smoothing factor of each class is dependent on the number of instances in this class. Having more training instances implies less label smoothing to be applied and vice versa. For example, consider a dataset containing 20 positive instances and 10 negative instances. Then, the new smoothed labels are  $1 - \frac{1}{20+2} = 0.9545$ , and  $\frac{1}{10+2} = 0.0833$  for the positive and negative classes, respectively. As the size of the training dataset instances approaches infinity, the smoothing factors tend to zero, which means we go back to the original case of unsmoothed labels.

Although Platt scaling approach works quite well with many datasets, Liina Partel [Pä19] showed that there is a high chance of finding better smoothing factors that are also dependent on the dataset size, class ratio, and dataset distribution. These claims suggest that there should be other formulas for choosing better smoothing factors. Otherwise, the straightforward approach is to try out many smoothing factors, which is computationally expensive, to find the factor that minimizes a target loss function.

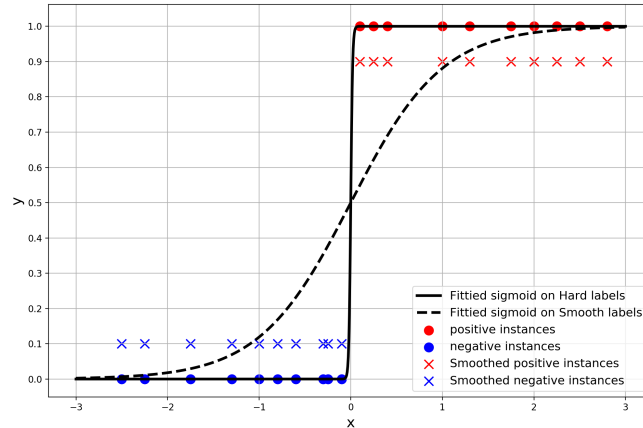


Figure 10. Effect of label smoothing on logistic regression fitting.

## 4 Optimal predictions for datasets with known generative models distribution

Although there are some solutions to reduce overfitting in logistic regression, it is still an open research question to find better ways that minimize the expected target loss function between the model predictions and the target labels of the population dataset, given a training sample. Log loss is a proper scoring rule that is usually minimized during the logistic regression fitting. There is no single method for finding the optimal predictions with all datasets due to the no-free-lunch theorem. In this section, we aim to derive the optimal probability predictions if we have information about the exact distribution of the generative model of the dataset.

In Section 4.1, we explain the Bayes-optimal model in case of exactly knowing the data set distribution, which is an idealized scenario. Next, in Section 4.2, we derive a Bayesian approach to find the optimal output probabilities in case of knowing the generative model distribution of the dataset. Finally, in Section 4.2.1, we discuss how to apply the derived formula practically using a sampling approach to obtain near-optimal predictions. Table 2 includes the notation to be used in the following derivations.

### 4.1 Bayes-optimal model for a binary classification task with normally distributed likelihoods

While looking for the optimal predictions of a probabilistic classifier, it is important to define here the Bayes-optimal model, which is the best possible classifier to be learned. The model is mainly used to derive theoretical best models as we are going to use in our later derivations.

A Bayes-optimal model predicts a probability distribution over classes. It is defined for  $K$  classes dataset as  $f^*(\mathbf{x}) = (P_1^*(\mathbf{x}), P_2^*(\mathbf{x}), \dots, P_K^*(\mathbf{x}))$ , where  $f^*$  denotes the Bayes-optimal model function, and  $P_i^*(\mathbf{x}) = P(Y_i = 1 | \mathbf{X} = \mathbf{x})$ . These predictions have the minimum expected proper loss (like log loss) with the target label distribution over the whole dataset space. The model predicts the class with the maximum a posteriori probability of occurrence (MAP) given by  $\arg \max_{i \in \{1, \dots, K\}} P_i^*(\mathbf{x})$ . Thus, for each instance, the model predicts the true probabilities of getting every label, given the features of this instance. If these features determine the label uniquely, the model predicts an output probability of 1 for the true class and 0 for the other classes. Otherwise, the model should be less confident [Fla12]. In binary classification tasks, only one probability  $P^*$  is predicted by the model such that  $P^* = P_1^*(\mathbf{x})$  means the probability of an observation  $\mathbf{x}$  belongs to the positive class, and  $1 - P^* = P_0^*(\mathbf{x})$  means the probability that it belongs to the negative class. This model is theoretical and cannot be learned as it requires an infinite training dataset covering the whole population, which is certainly an impractical scenario.

Next, we will derive the Bayes-optimal model for a binary classification dataset. The derivation will be extended later to find the exact model formula for the optimal predictions when the probability density functions (*pdfs*) of the classes' likelihoods are normally distributed.

**Lemma 1.** *The Bayes-optimal model for a binary classification dataset with balanced class distribution, is  $f^*(x) = \frac{p(X=x|Y=1)}{p(X=x|Y=0)+p(X=x|Y=1)}$ .*

*Proof.* The posterior probability predictions of the Bayes-optimal model function for instance  $\mathbf{x}$  is defined as:

$$f^*(x) = P(Y = 1|X = x)$$

Using Bayes' rule:

$$f^*(x) = \frac{p(X = x|Y = 1)P(Y = 1)}{p(X = x)}$$

From the law of total probability:

$$\begin{aligned} p(X = x) &= p(X = x|Y = 0)P(Y = 0) + p(X = x|Y = 1)P(Y = 1) \\ \Rightarrow f^*(x) &= \frac{p(X = x|Y = 1)P(Y = 1)}{p(X = x|Y = 0)P(Y = 0) + p(X = x|Y = 1)P(Y = 1)} \end{aligned}$$

If the class distribution is balanced,  $P(Y = 0) = P(Y = 1)$ :

$$\Rightarrow f^*(x) = \frac{p(X = x|Y = 1)}{p(X = x|Y = 0) + p(X = x|Y = 1)}$$

□

Suppose we have sufficiently many instances as identically distributed independent random variables in both classes. It would not be unreasonable to assume that the likelihood probabilities  $p(X = x|y)$  belong to normal distributions using the central limit theorem [Fla12]. Following these assumptions, consider that the normal distributions of the classes' likelihoods have the same standard deviation  $\sigma$  and means of  $\mu_0, \mu_1$  for the negative and positive classes, respectively as shown in Figure 11. In this case, the Bayes-optimal model is a logistic sigmoid, as shown in Theorem 1 below.

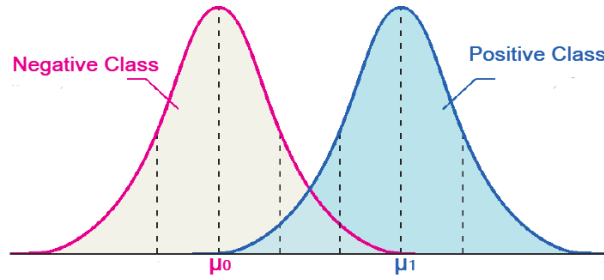


Figure 11.  $pdfs$  of the likelihoods of the negative and positive classes are normally distributed with means of  $\mu_0, \mu_1$ , respectively.

**Theorem 1.** In a balanced binary classification dataset with normally distributed likelihood probabilities of each class, the Bayes-optimal model is a logistic sigmoid function with slope  $a = \frac{\mu_1 - \mu_0}{\sigma^2}$ , and intercept  $b = \frac{\mu_0^2 - \mu_1^2}{2\sigma^2}$ .

*Proof.* The likelihood probability of an instance  $x$  given the positive class is:

$$p(X = x|Y = 1) = \mathcal{N}(x|\mu_1, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}$$

Similarly, the likelihood probability of  $x$  given the negative class is:

$$p(X = x|Y = 0) = \mathcal{N}(x|\mu_0, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu_0)^2}{2\sigma^2}}$$

From Lemma 1, the optimal predicted probabilities of a Bayes-optimal model are equal to:

$$f^*(x) = \frac{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}}{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}} + e^{-\frac{(x-\mu_0)^2}{2\sigma^2}}}$$

Dividing both numerator and denominator by  $e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}$  and simplifying the results:

$$\begin{aligned} \Rightarrow f^*(x) &= \frac{1}{1 + \frac{e^{-\frac{(x-\mu_0)^2}{2\sigma^2}}}{e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}}} = \frac{1}{1 + e^{-\frac{(x-\mu_0)^2}{2\sigma^2} + \frac{(x-\mu_1)^2}{2\sigma^2}}} = \frac{1}{1 + e^{-\frac{((x-\mu_0)^2 - (x-\mu_1)^2)}{2\sigma^2}}} \\ &= \frac{1}{1 + e^{-\left(\frac{\mu_1 - \mu_0}{\sigma^2} x + \frac{\mu_0^2 - \mu_1^2}{2\sigma^2}\right)}} = \frac{1}{1 + e^{-(ax+b)}} \end{aligned}$$

where  $a = \frac{\mu_1 - \mu_0}{\sigma^2}$ , and  $b = \frac{\mu_0^2 - \mu_1^2}{2\sigma^2}$ . □

Bayes-optimality is concerned with the properties of the model on the testing set distribution. Thus, a sample of the training set instances will not be enough to find the actual  $\mu_0, \mu_1$  of the population distribution. Besides, the optimal  $a$  and  $b$  values of the Bayes-optimal model can not be learned. So, finding the optimal predictions of the logistic regression model for a single dataset is impossible without knowing the actual parameters of the population distribution of this dataset.

## 4.2 Derivation of the optimal predictions given a known dataset generative model and a training set

Learning the actual parameters  $\mu_0, \mu_1, \sigma$  of the population distribution from the training set is an unrealistic scenario since we always have uncertainty about them. Although we can estimate these parameters from the training set, but these are inevitably slightly different from the population parameters. Instead, suppose that this uncertainty is perfect and exactly known such that the actual values of  $\mu_0, \mu_1, \sigma$  are drawn from the distributions denoted as  $\mathcal{D}_{\mu_0}, \mathcal{D}_{\mu_1}, \mathcal{D}_{\sigma}$  representing

our uncertainty. A generative model is defined by these uncertainty distributions, and used to generate datasets with distribution parameters of  $\mu_0, \mu_1, \sigma$  as follows:

$$\begin{aligned}\sigma &\sim \mathcal{D}_\sigma, \\ \mu_0 &\sim \mathcal{D}_{\mu_0} \\ \mu_1 &\sim \mathcal{D}_{\mu_1}.\end{aligned}$$

In this case, it turns out that we can derive the optimal probability predictions under the uncertainty of  $(\mu_0, \mu_1, \sigma)$  defined by the prior information about the generative model. Our task is to find the model which minimizes the expected log loss given the generative model distribution and a training set.

Figure 12 shows an example for a uniformly distributed generative model (solid lines) for  $\mu_0$ , and  $\mu_1$ . These distributions are defined as  $\mathcal{D}_{\mu_0} = \mathcal{U}[-2.5, 0]$ , and  $\mathcal{D}_{\mu_1} = \mathcal{U}[0, 2.5]$ , where  $\mu_0 \sim \mathcal{D}_{\mu_0}$ ,  $\mu_1 \sim \mathcal{D}_{\mu_1}$ , and  $\mathcal{U}[a, b]$  is a uniform distribution starting at  $a$  and ending at  $b$ . Then, randomly sampled dataset parameters from the generative model are drawn, and the dataset distribution is plotted (dotted lines). The sampled means of the normally distributed likelihoods of the negative and positive classes of this example dataset are  $\mu_0 = -1.5$  and  $\mu_1 = 1.2$ , respectively. The standard deviation of both classes is 1 ( $\mathcal{D}_\sigma = \delta(t - 1)$ ), where  $\delta(t - c)$  is a Dirac delta function at  $c$ .

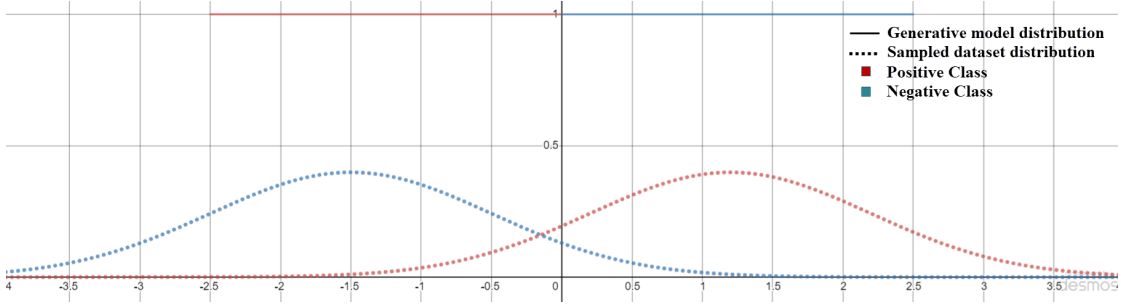


Figure 12. An example of uniformly distributed generative models for the parameters of normally distributed likelihoods of the generated dataset.

To generalize, we will denote the parameters of the distributions of the classes' likelihoods as  $\theta$ . The generative model distribution of  $\theta$  representing its exact uncertainty, is denoted by  $\mathcal{D}_\theta$ . In case that the likelihoods of the classes are normally distributed for both the positive and negative classes as presented in the previous example, then,  $\theta = (\mu_0, \mu_1, \sigma)$ , and  $\mathcal{D}_\theta = \mathcal{D}_{\mu_0} \times \mathcal{D}_{\mu_1} \times \mathcal{D}_\sigma$ .

If the uncertainty distribution is discrete with exactly a single value at  $\theta$ , the task is reduced to finding the Bayes-optimal model given the dataset distribution parameters  $\theta$  (Theorem 1). Let the Bayes-optimal model function in this case is denoted as follows:

$$f^*(\mathbf{x}; \theta) = p(Y = 1 | \mathbf{X} = \mathbf{x}, \Theta = \theta). \quad (7)$$

Now, given that the uncertainty of the parameters of the dataset is not just a single value but a continuous distribution, the Bayes-optimal model can be derived from Theorem 2 as shown

below. It is worthwhile to mention that Theorem 2 is not used later in the derivation of the optimal predictions given a training dataset  $D_{tr}$ , and the generative model distribution  $\mathcal{D}_\theta$ . However, it has been proved to ease the understanding of the next Theorem 3 that is conceptually similar but with an extra conditioning on a training set  $D_{tr}$ .

**Theorem 2.** *Given a generative model with distribution  $\mathcal{D}_\theta$ , the optimal probability predictions  $P(Y = 1|\mathbf{X} = \mathbf{x})$  for an instance  $\mathbf{x}$ , are equal to  $\int_{\theta \in \mathcal{D}_\theta} f^*(\mathbf{x}; \theta) p(\Theta = \theta) d\theta$ , where  $f^*$  is the Bayes-optimal model for a fixed  $\theta$ .*

*Proof.* Let  $g^*(\mathbf{x}) = P(Y = 1|\mathbf{X} = \mathbf{x})$  denote the optimal model function given a generative model of distribution  $\mathcal{D}_\theta$ . Then, using the law of total probability:

$$g^*(\mathbf{x}) = p(Y = 1|\mathbf{X} = \mathbf{x}) = \int_{\theta \in \mathcal{D}_\theta} p(Y = 1, \Theta = \theta|\mathbf{X} = \mathbf{x}) d\theta$$

Using the chain rule of probability:

$$\Rightarrow g^*(\mathbf{x}) = \int_{\theta \in \mathcal{D}_\theta} p(Y = 1|\Theta = \theta, \mathbf{X} = \mathbf{x}) p(\Theta = \theta|\mathbf{X} = \mathbf{x}) d\theta$$

From Equation 7, substituting with the Bayes-optimal model  $f^*$  for a fixed  $\theta$ :

$$\Rightarrow g^*(\mathbf{x}) = \int_{\theta \in \mathcal{D}_\theta} f^*(\mathbf{x}; \theta) p(\Theta = \theta|\mathbf{X} = \mathbf{x}) d\theta$$

Since  $\theta$  is unconditionally independent of the instance  $\mathbf{x}$ :

$$\Rightarrow g^*(\mathbf{x}) = \int_{\theta \in \mathcal{D}_\theta} f^*(\mathbf{x}; \theta) p(\Theta = \theta) d\theta$$

□

Finally, in a typical scenario, another evidence always exists, which is the training dataset. Next, we derive the optimal predicted probabilities given a training set  $D_{tr}$ , and the generative model distribution  $\mathcal{D}_\theta$ .

**Theorem 3.** *Given a training set  $D_{tr}$ , and a generative model of distribution  $\mathcal{D}_\theta$ , the optimal probability predictions  $P(Y = 1|\mathbf{X} = \mathbf{x}, D = D_{tr})$  for an instance  $\mathbf{x}$ , are equal to  $\int_{\theta \in \mathcal{D}_\theta} f^*(\mathbf{x}; \theta) p(D = D_{tr}|\Theta = \theta) p(\Theta = \theta) d\theta$ , where  $f^*$  is the Bayes-optimal model for a fixed  $\theta$ .*

*Proof.* Let the posterior probability of the population distribution parameters  $\theta$  given the training set  $D_{tr}$  is  $p(\Theta = \theta|D = D_{tr})$ .

Following Bayes' rule (Section 2.5), the posterior probability can be written as:

$$p(\Theta = \theta|D = D_{tr}) = \frac{p(D = D_{tr}|\Theta = \theta)p(\Theta = \theta)}{p(D = D_{tr})}$$



The denominator  $p(D = D_{tr})$  is a positive constant. Also, we are interested in comparing the probability of the event  $\Theta = \theta$ . So, for simplicity,  $p(D = D_{tr})$  can be omitted:

$$p(\Theta = \theta | D = D_{tr}) \propto p(D = D_{tr} | \Theta = \theta) p(\Theta = \theta). \quad (8)$$

Let  $h^*(\mathbf{x}, D_{tr}) = P(Y = 1 | \mathbf{X} = \mathbf{x}, D = D_{tr})$  denote the optimal model function given a generative model of distribution  $\mathcal{D}_\theta$ , and a training set  $D_{tr}$ . Then, using the law of total probability:

$$h^*(\mathbf{x}; D_{tr}) = p(Y = 1 | \mathbf{X} = \mathbf{x}, D = D_{tr}) = \int_{\theta \in \mathcal{D}_\theta} p(Y = 1, \Theta = \theta | \mathbf{X} = \mathbf{x}, D = D_{tr}) d\theta$$

Using the chain rule of probability:

$$\Rightarrow h^*(\mathbf{x}; D_{tr}) = \int_{\theta \in \mathcal{D}_\theta} p(Y = 1 | \Theta = \theta, \mathbf{X} = \mathbf{x}, D = D_{tr}) p(\Theta = \theta | \mathbf{X} = \mathbf{x}, D = D_{tr}) d\theta$$

Since  $\theta$ , and the class label of the instance  $Y$  are unconditionally independent of the instance  $\mathbf{x}$ , and the training set  $D_{tr}$ , respectively:

$$\Rightarrow h^*(\mathbf{x}; D_{tr}) = \int_{\theta \in \mathcal{D}_\theta} p(Y = 1 | \Theta = \theta, \mathbf{X} = \mathbf{x}) p(\Theta = \theta | D = D_{tr}) d\theta$$

From Equation 7, substituting using the Bayes-optimal model  $f^*$  for a fixed  $\theta$ :

$$\Rightarrow h^*(\mathbf{x}; D_{tr}) = \int_{\theta \in \mathcal{D}_\theta} f^*(\mathbf{x}; \theta) p(\Theta = \theta | D = D_{tr}) d\theta. \quad (9)$$

Substituting Equation 8 back into Equation 9:

$$\Rightarrow h^*(\mathbf{x}; D_{tr}) = \int_{\theta \in \mathcal{D}_\theta} f^*(\mathbf{x}; \theta) p(D = D_{tr} | \Theta = \theta) p(\Theta = \theta) d\theta$$

□

If the *pdfs* of the classes' likelihoods are normally distributed such that  $\theta = (\mu_0, \mu_1, \sigma)$ , and the uncertainty of dataset distribution parameters  $\mathcal{D}_\theta$  is  $\mathcal{D}_{\mu_0} \times \mathcal{D}_{\mu_1} \times \mathcal{D}_\sigma$ , where  $\mu_0 \sim \mathcal{D}_{\mu_0}$ ,  $\mu_1 \sim \mathcal{D}_{\mu_1}$ ,  $\sigma \sim \mathcal{D}_\sigma$ . Then, from Theorem 3, the optimal predicted probabilities can be written as follows:

$$h^*(\mathbf{x}; D_{tr}) = \int_{\mu_0} \int_{\mu_1} \int_{\sigma} f^*(\mathbf{x}; \mu_0; \mu_1; \sigma) p(D_{tr} | \mu_0, \mu_1, \sigma) p(\mu_0, \mu_1, \sigma) d\sigma d\mu_1 d\mu_0. \quad (10)$$

From Theorem 1, the optimal model  $f^*$  given an exact dataset distribution parameters  $\theta = (\mu_0, \mu_1, \sigma)$ , is a logistic sigmoid function with a slope  $a = \frac{\mu_1 - \mu_0}{\sigma^2}$  and an intercept  $b = \frac{\mu_0^2 - \mu_1^2}{2\sigma^2}$ . Hence, the optimal probability predictions (10) is simplified to the posterior mean of predictions of the Bayes-optimal models  $f^*$  for each  $\mu_0 \in \mathcal{D}_{\mu_0}$ ,  $\mu_1 \in \mathcal{D}_{\mu_1}$ ,  $\sigma \in \mathcal{D}_\sigma$  as follows:

$$h^*(\mathbf{x}; D_{tr}) = \int_{\mu_0} \int_{\mu_1} \int_{\sigma} \frac{1}{1 + e^{-(ax+b)}} p(D_{tr} | \mu_0, \mu_1, \sigma) p(\mu_0) p(\mu_1) p(\sigma) d\sigma d\mu_1 d\mu_0. \quad (11)$$

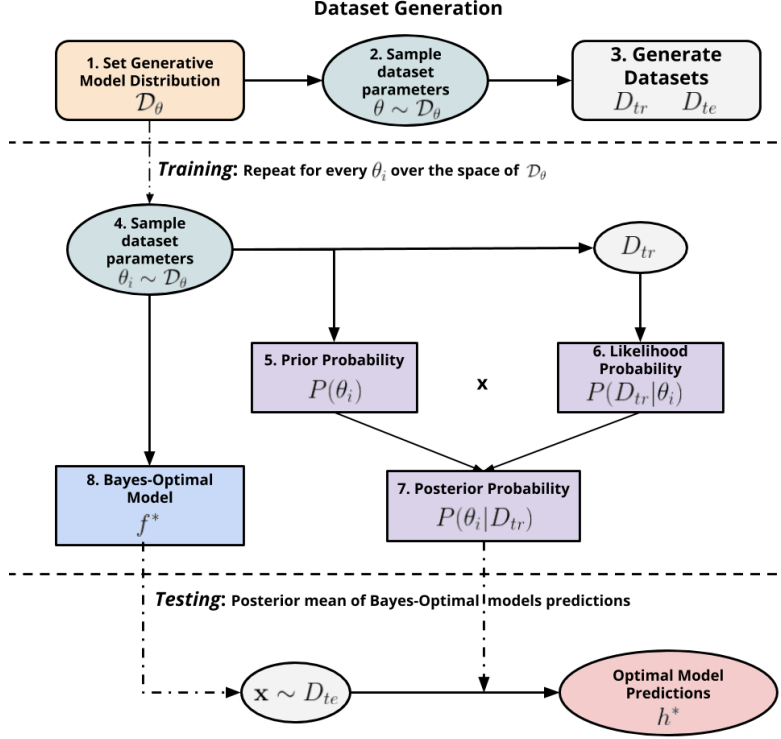


Figure 13. Steps of training and testing phases for obtaining the optimal predicted output probabilities in case of knowing the generative model distribution of a dataset.

#### 4.2.1 Practicality Concerns

If the uncertainty distribution  $\mathcal{D}_\theta$  of the dataset distribution parameters  $\theta$  is a continuous distribution, evaluating the optimal model  $h^*$  for every  $\theta \in \mathcal{D}_\theta$  is practically impossible as  $\mathcal{D}_\theta$  has an infinite number of possible parameter configurations. Therefore, we introduce a set of approximations that can be applied to Equation 11 to obtain near-optimal predictions.

1. We use a Monte Carlo sampling approach for a large number of parameter configurations for the dataset distribution  $\theta$  out of the uncertainty distribution  $\mathcal{D}_\theta$ . As the number of samples  $N_{MC}$  grows, the loss converges to a similar value to the predictions of the optimal model  $h^*$ . Thus, Equation 11 can be approximated as follows:

$$h_{approx}^*(\mathbf{x}; D_{tr}) = \sum_{i=1}^{N_{MC}} (f^*(\mathbf{x}; \theta_i) p(D = D_{tr}|\Theta = \theta_i) p(\Theta = \theta_i)), \text{ where } \theta_i \in \mathcal{D}_\theta.$$

The implementation steps of this approximated formula for the optimal predictions is summarized in Figure 13.

2. During inference, the evaluation of the posterior mean of predicted probabilities out of  $N_{MC}$  models for every single instance is computationally expensive. Alternatively, we present two further different simplifications to reduce the complexity of the calculations.

- (a) We use a single model with coefficients equal to the posterior mean of the coefficients of the sampled  $N_{MC}$  Bayes-optimal models. Hence, instead of computing the posterior mean of  $N_{MC}$  models' predictions for each instance as in Equation 11, one can predict approximated probabilities to the near-optimal ones using only a single logistic function defined as  $\frac{1}{1+e^{-(\mathbf{a}_{approx}^* \cdot \mathbf{x} + b_{approx}^*)}}$ , and derived from all the sampled models as shown below. Later, we refer to this method as *bayes\_coeffs*:

$$\begin{aligned} \mathbf{a}_{approx}^* &= \sum_{i=1}^{N_{MC}} (\mathbf{a}^*(\mathbf{x}; \theta_i) p(D = D_{tr} | \Theta = \theta_i) p(\Theta = \theta_i)) \\ b_{approx}^* &= \sum_{i=1}^{N_{MC}} (b^*(\mathbf{x}; \theta_i) p(D = D_{tr} | \Theta = \theta_i) p(\Theta = \theta_i)), \\ \text{where } \theta_i &\in \mathcal{D}_\theta, \\ \mathbf{a}^* &\text{are the coefficients of the Bayes-optimal model } f^* \\ b^* &\text{is the intercept term of the Bayes-optimal model } f^* \end{aligned}$$

$$\Rightarrow h_{approx}^*(\mathbf{x}; D_{tr}) = \frac{1}{1+e^{-(\mathbf{a}_{approx}^* \cdot \mathbf{x} + b_{approx}^*)}}.$$

- (b) We use only the Bayes-optimal model with the MAP probability among all the sampled  $N_{MC}$  models to be used in making predictions directly as shown below. This method is referred to as *bayes\_map*:

$$\begin{aligned} \theta_{MAP} &= \underset{\theta_i}{\operatorname{argmax}} (p(D = D_{tr} | \Theta = \theta_i) p(\Theta = \theta_i)), \\ \text{where } \theta_i &\in \mathcal{D}_\theta, i \in \{1, 2, \dots, N_{MC}\} \\ \Rightarrow h_{approx}^*(\mathbf{x}; D_{tr}) &= f^*(\mathbf{x}; \theta_{MAP}). \end{aligned}$$

As the number of Monte Carlo samples  $N_{MC}$  drawn from the uncertainty distribution  $\mathcal{D}_\theta$  increases, the predicted probabilities are going to be better such that we reach the optimal predictions when  $N_{MC} \rightarrow \infty$ . However, a large finite number of samples could be enough to achieve near-optimal predictions.

To select a good value for  $N_{MC}$  that can be used further in computing the probability predictions, we carried out a simple experiment to test the validity of our approximations and the convergence of the loss value as the  $N_{MC}$  increases. In this experiment, we generate random datasets with normally distributed likelihoods with means  $\mu_0, \mu_1$  for negative and positive classes, respectively. The standard deviation of both classes is set to  $\sigma = 1$ .

The uncertainty distributions of the mean values were defined once as  $\mathcal{D}_{\mu_0} = \mathcal{U}[-2, 0], \mathcal{D}_{\mu_1} = \mathcal{U}[0, 2]$ , and another time as  $\mathcal{D}_{\mu_0} = \mathcal{U}[-3, -1], \mathcal{D}_{\mu_1} = \mathcal{U}[1, 3]$ . These selections were made to ensure that the mean of the positive class distribution is always larger than that of the negative class ( $\mu_1 \geq \mu_0$ ) with different separation distances between these means. When the means are close to each other, the likelihood distributions of the classes are highly overlapped, making the loss convergence of the learned model more difficult and vice versa. In both cases, we generate a small training set of 50 instances to make it difficult for the model to estimate the actual population distribution parameters from the training set and a testing set of  $10^6$  instances. We increase the value of  $N_{MC}$  with small steps from 1 to  $10^6$ , and the test cross-entropy loss values between the predicted probabilities of our proposed sampling approximations and the target labels are reported.

Figure 14 shows the results of the two examples for the convergence of the cross-entropy loss on the y-axis, with the growth of the number of drawn samples out of the uncertainty distribution  $N_{MC}$  on the x-axis. The loss value has almost converged as  $N_{MC} > 10^4$ . Additionally, both *bayes\_coeffs* and *bayes\_map* provide a good approximation to the near-optimal predicted probabilities of the posterior mean of models' predictions (*bayes\_preds*). Thus, we fixed the value of  $N_{MC}$  to  $10^5$  in our evaluation experiments in Section 7.

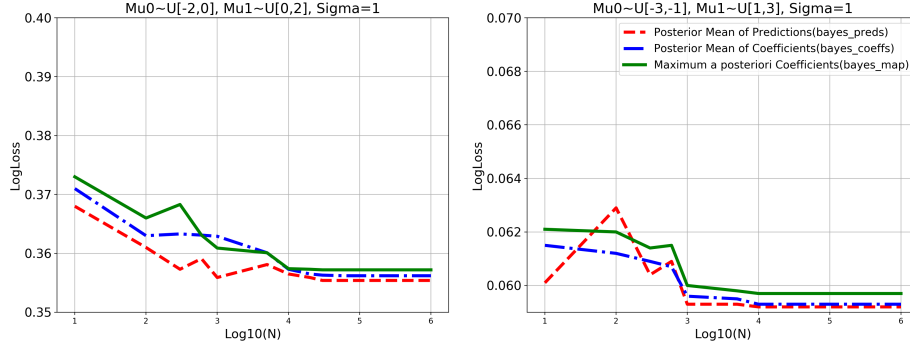


Figure 14. Convergence of cross-entropy loss value as the number of samples drawn from uniform generative model distribution increases. Left image has  $\mu_0 \in \mathcal{U}[-2, 0]$ ,  $\mu_1 \in \mathcal{U}[0, 2]$ ,  $\sigma = 1$ , Right image has  $\mu_0 \in \mathcal{U}[-3, -1]$ ,  $\mu_1 \in \mathcal{U}[1, 3]$ ,  $\sigma = 1$ .

## 5 Instance-based label smoothing using kernel density estimation

In this section, we explain our proposed method for the instance-based label smoothing for logistic regression fitting. First, we start by motivating the shortcomings of Platt scaling and constant factor label smoothing. Next, we illustrate the intuition behind the proposed method and how it works. We then introduce the essential background for the kernel density estimation and formulate the steps of the proposed method.

### 5.1 Disadvantages of the constant factor label smoothing

Platt scaling is the logistic calibration process that applies label smoothing to the hard target class labels of the training set and transforms them into soft target probabilities before fitting the model. This process helps in reducing the overconfidence in the output model predictions. The smoothing factors of Platt scaling are defined as  $\epsilon_{platt}^+ = \frac{1}{|D^+|+2}$ ,  $\epsilon_{platt}^- = \frac{1}{|D^-|+2}$  for the positive and negative classes, respectively. Thus, these factors are dependent only on the number of instances in each class ( $|D^+|$ ,  $|D^-|$ ).

Liina Pärlt [Pä19] has shown that, in many cases, there exist better smoothing factors than the ones proposed by Platt scaling. These smoothing factors are dependent on the training set size, class ratio, and the separation distance between the distributions of the classes' likelihoods. However, in Platt scaling, the smoothing factor is dependent on the training set size only. Figure 15 shows the best smoothing factors corresponding to datasets with different sizes and separation distances between the normally distributed likelihoods of the classes. When the size of the training set is fixed (same x-axis value), Platt scaling always has the same smoothing factors, whatever the separation distance of the classes' likelihoods. However, different smoothing factors were found to be better than Platt scaling according to the separation distance between the classes.

The previous results suggest that there is usually a better smoothing factor than the ones used by Platt scaling. However, there is no straightforward way to combine the training set size, class ratio, and the separation distance between the distributions of the classes' likelihoods to get the value of a better smoothing factor. Hence, we need to perform a long and tedious search in a wide range of smoothing factor values to find a suitable one.

### 5.2 The intuition behind the instance-based label smoothing

As long as we introduce some noise to the labels of the overconfident training instances in the form of target softening, this will help in reducing the model overconfidence. On the other hand, if it is expected that the fitted model will not be overconfident of its predictions for some instances. Then, it is not useless to smoothen the labels of these instances. We propose an instance-based label smoothing approach that gives more flexibility to select a better smoothing factor if we can estimate the confidence of the model predictions.

Figure 16 illustrates an example of the logistic calibration of the output scores of a certain scoring classifier. The scores of the positive and negative classes are fully separated, such that the class label is always positive when the score value is  $> 0$ . Otherwise, it is always negative. This

**Best smoothing factors corresponding to classes' likelihoods at different separation levels**

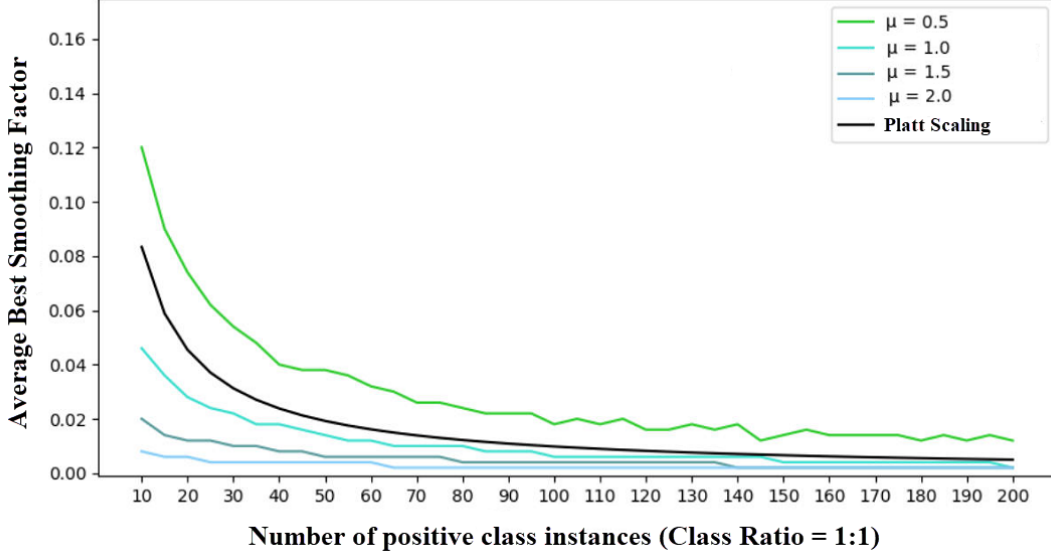


Figure 15. Best smoothing factors corresponding to datasets with different sizes and separation distances between the normally distributed likelihoods of the classes.  $p(X = x|Y = 1) \sim \mathcal{N}(\mu, 1)$ ,  $p(X = x|Y = 0) \sim \mathcal{N}(-\mu, 1)$  [Pä19].

separation leads to an overfitted logistic sigmoid function (solid black line) and, consequently, overconfident probability predictions. If the labels were smoothed with a constant factor (marked with crosses), the fitted sigmoid function would be less confident of its output predictions (dashed black line).

During the process of label smoothing, suppose that a positive class instance  $x_1$  corresponding to a classifier prediction score of 0.1 (yellow vertical line), has a target label value of  $1 - \epsilon_{platt}^+ = 0.9$ . Similarly, another positive class instance  $x_2$  corresponding to a classifier output score of 2.9 (blue vertical line) has the same smoothed target label value 0.9. Thus, both instances are smoothed using the same factor, although the fitted logistic sigmoid curve's confidence will mostly be different for each of them. Consequently, this suggests that if we can estimate that the logistic sigmoid curve will be overconfident of its predictions, a large smoothing factor can be used. Otherwise, there is no need to introduce label smoothing.

The logistic sigmoid function is a continuous monotonic function that always preserves the same input order given to it. This property ensures that instances that are more likely to exist in an overlapped region between the distributions of the classes' likelihoods will have less confidence than instances that exist mostly within one class distribution only. For example, the output confidence level of the sigmoid function for the instance  $x_1 = 0.1$  is undoubtedly less than or equal to the output confidence of instance  $x_2 = 2.9$ . Hence, one can estimate the output confidence of the fitted sigmoid function on each instance based on this score value  $x$  (input to

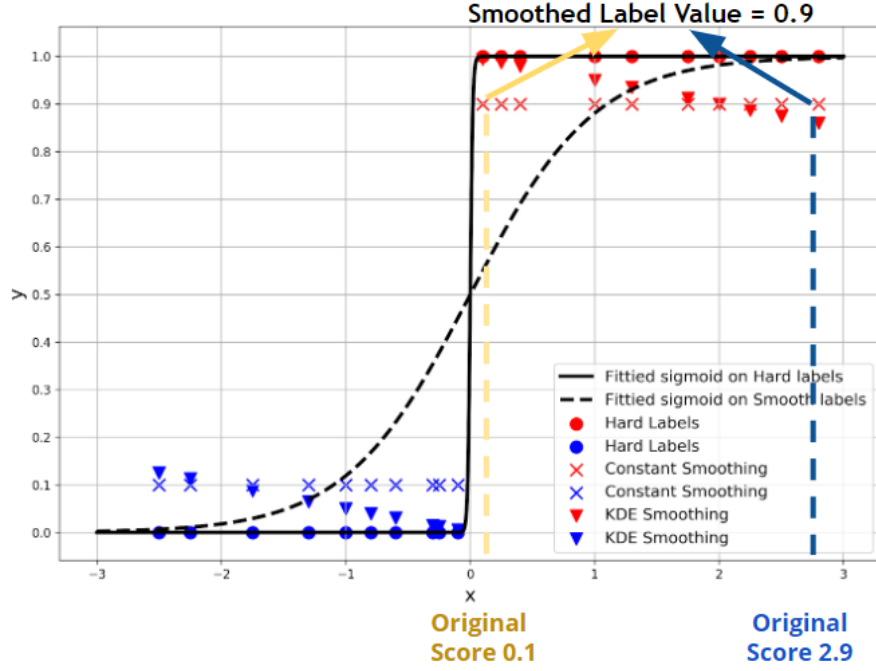


Figure 16. Difference between hard labels (dots), smoothed labels with a constant smoothing factor (crosses), and instance-based label smoothing (triangles).

the logistic sigmoid) and the *pdfs* of the classes' likelihoods of the positive and negative classes. Instance with high estimated confidence will be smoothed with large factors, and instances with low estimated confidence will be smoothed with small smoothing factors.

Even in cases of full separation like Figure 16, the monotonic property of the sigmoid function will ensure that the output predicted confidence levels of the instances with small smoothing factors like  $x_1 = 0.1$  are less than that of the instances with large smoothing factors like  $x_2 = 2.9$ . Therefore, we claim that using an instance-based label smoothing with factors (marked with triangles) that are proportional to the estimated confidence levels of these instances, reduces the overconfidence in the predicted probabilities better than constant smoothing factor methods like Platt scaling.

To get more intuition behind how the proposed method works, Figure 17 shows the difference among the fitted logistic sigmoid curves with Platt scaling (platt), without label smoothing (normal), and with instance-based label smoothing (ibls) for a randomly sampled univariate dataset of normally distributed likelihoods for the positive, and negative classes with standard deviation  $\sigma = 1$ ,  $\mu_0 = -\mu_1$ , and  $\mu_1 = \{0.5, 1, 1.5, 2\}$  (The separation between the distributions of the classes' likelihoods increases from subplot (a) to subplot (d), respectively).

The smoothing factor of the instance-based label smoothing method is directly proportional to the instance value ( $x$ ), and consequently, to its estimated confidence level too (higher estimated

confidence = larger smoothing factor). The maximum smoothing factor that can be reached is set to that obtained from Platt scaling  $\epsilon_{platt}$ .

In subplot (a), the instances of both classes are highly overlapped. Thus, the instance-based label smoothing assigns less smoothing factors than Platt scaling even for the instances with extremely low or high feature values on the far most right/left. Hence, the slope of the fitted instance-based sigmoid curve (0.94) is higher than that of Platt scaling (0.89), and it is closer to the optimal sigmoid curve (1). As the separation between the classes increases until the separation occurs in subplot (d), the instance-based label smoothing assigns large smoothing factors approaching that of Platt scaling. Thus, the sigmoid curve fitted using the instance-based label smoothing matches that of Platt scaling with a slope of 1.08. On the other hand, the normal sigmoid curve is too much overconfident of its predictions with a slope of 14.52.

The figure aims to show that the instance-based label smoothing approach is more reasonable and flexible than Platt scaling. Large smoothing factors will be used if the model is estimated to be overconfident of its predictions. On the other hand, smaller smoothing factors are assigned when the model is estimated to be not confident about its predictions. Consequently, not only the training set size is taken into account like Platt scaling, but also the separation between the distributions of the classes' likelihoods. More experimentation to test our proposed method (*ibls*) is carried out in Section 7.

Later on, we will refer to the logistic regression model fitted with instance-based label smoothing as *ibls*.

### 5.3 Kernel density estimation

In our instance-based label smoothing method, we start by looking for an approach that can estimate the *pdfs* of the classes' likelihoods using the training set. The *pdfs* will be used later to estimate each instance's confidence level based on its location within these *pdfs*. For that reason, kernel density estimation (*KDE*) is employed to estimate the distribution of the classes' likelihoods from the available training set instances.

*KDE* is considered as one of the most efficient non-parametric density estimation methods, that involves fitting a model to an arbitrary distribution of a data sample to make inference about the whole population [Par62]. *KDE* is useful in cases when it is needed to predict the continuous distribution shape of some random variables instead of having discrete binned histograms. Additionally, it can be used to generate data points that seem to belong to the same dataset distribution of the original sample [DLP11].

Suppose that an independent and identically distributed sample  $(z_1, z_2, \dots, z_m)$  was collected out of a univariate random variable  $Z$ . The estimated probability density function  $\hat{f}$  of the random variable  $Z$  is shown as follows, where  $K$  is the non-negative kernel function, and  $h$  is the kernel scale parameter (a.k.a bandwidth):

$$\hat{f}_h(z) = \frac{1}{m} \sum_{i=1}^m K_h(z - z_i) = \frac{1}{m \cdot h} \sum_{i=1}^m K\left(\frac{z - z_i}{h}\right).$$

Though the kernel  $K$  could be many different shapes, it is prevalent that the kernel type is chosen to be a Gaussian distribution. However, the bandwidth parameter still has a strong



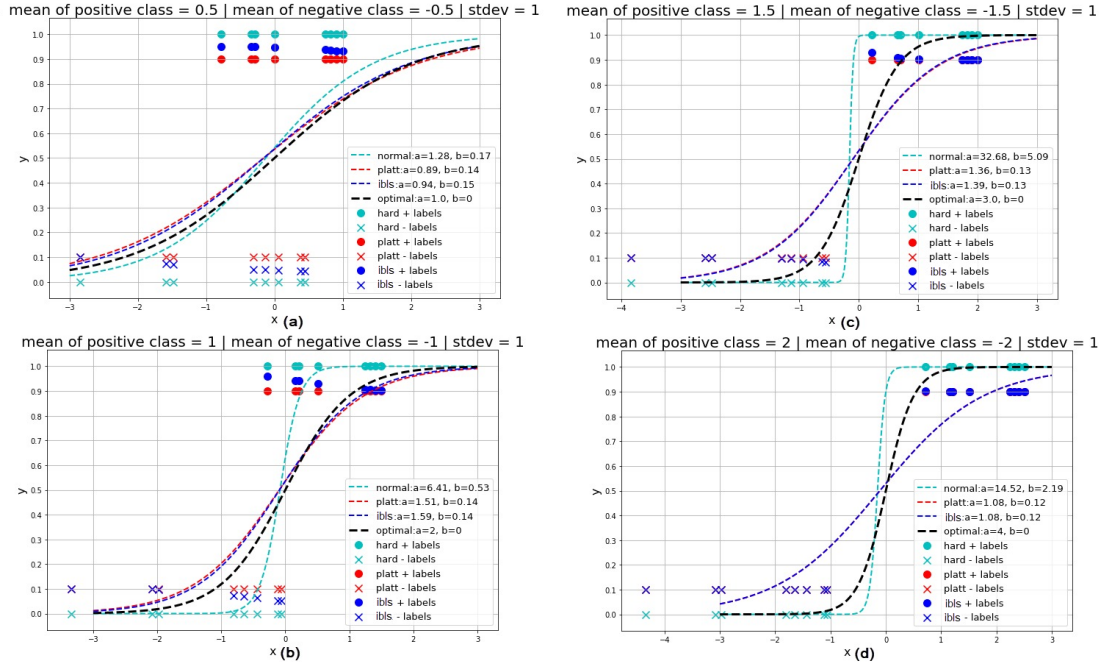


Figure 17. The difference among fitted logistic sigmoid functions with no label smoothing (normal-cyan), Platt scaling (Platt-red), instance-based smoothing (ibls-blue), and optimal curve given the parameters of the normally distributed classes' likelihoods used to generate the dataset (optimal-black). The separation distance between the means of the classes' likelihoods increases as we go from (a) to (d) until separation problem occurs in (d).

influence on the final estimated density function. A too narrow bandwidth could lead to an under-smoothed density function with many data artifacts. On the other hand, a too wide bandwidth could lead to an over-smoothed estimated density function for the random variable. However, as a rule of thumb, a general formula was proposed to compute a robust value for the kernel bandwidth  $h$  that could achieve the minimum mean integrated squared error calculated as  $\mathbb{E} \left( \int \left( \hat{f}_m(z) - f(z) \right)^2 dz \right)$ , where  $\hat{f}_m$  is the estimated density function,  $f$  is the unknown density, and the expectation is performed on the drawn sample of size  $m$  for the random variable  $Z$  [Sil86].

## 5.4 Methodology

Next, we aim to explain the steps of our proposed method *ibls* for computing the smoothing factors of the instance-based label smoothing approach.

Suppose that a scoring classifier is trained on a binary classification dataset. The prediction

scores of this classifier for the positive or negative class instances can be assumed as random variables. First, we aim to have an estimated distribution (*pdf*) of the classifier predictions per each class to estimate the probability density of any new instance that it belongs to each of these classes. For example, if the scoring classifier has an output score of 2.9 for an instance  $x$ , what is the probability density that  $x$  is drawn from the distribution of the positive class likelihoods ( $p(x = 2.9|y = 1)$ )?

Although the actual probability density function of the likelihoods of the positive/negative class needs the whole population instances, an approximate probability distribution could be estimated by fitting a kernel density estimator using the available training set samples. Thus, based on the scoring classifier predictions on the training set instances, two different *pdf*s will be estimated for the output classifier scores. One distribution will be estimated for the positive class and another one for the negative class.

Next, the following function illustrates how the smoothing factor for an arbitrary instance  $x_i$  with a target label  $y_i$ , is computed:

$$\epsilon_{ibls}(x_i) = \begin{cases} \epsilon_{platt}^+ \times \frac{f^+(x_i)}{f^+(x_i) + f^-(x_i)} & \text{if } y_i = 1 \\ \epsilon_{platt}^- \times \frac{f^-(x_i)}{f^+(x_i) + f^-(x_i)} & \text{if } y_i = 0. \end{cases}$$

where  $f^+(x_i)$ , and  $f^-(x_i)$  are the estimated density scores for the instance  $x_i$  given that it belongs to the positive class and the negative class, respectively.

**Example:**

Suppose the same two positive class instances mentioned earlier. The predictions of a scoring classifier for these two instances are  $x_1 = 0.1$ , and  $x_2 = 2.9$ . Given that  $\epsilon_{platt}^+ = 0.1$ , and the estimated *pdf*s of the classes' likelihood is shown in Figure 18, such that  $f^+(x_1) = 0.7$ ,  $f^-(x_1) = 0.4$ , and  $f^+(x_2) = 0.9$ ,  $f^-(x_2) = 0.05$ .

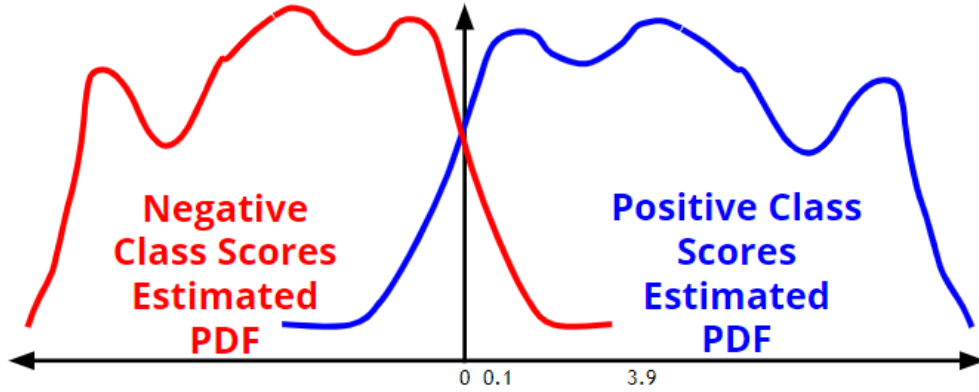


Figure 18. Example of the estimated *pdf*s of negative (red), and positive (blue) classes' likelihoods.

The target smoothed labels for both instances are evaluated as follows:

$$\epsilon_{ibls}(x_1) = 1 - \epsilon_{platt}^+ \times \frac{f^+(x_1)}{f^+(x_1) + f^-(x_1)} = 1 - 0.1 \times \frac{0.7}{0.7+0.4} \approx 0.936$$

$$\epsilon_{ibls}(x_2) = 1 - \epsilon_{platt}^+ \times \frac{f^+(x_2)}{f^+(x_2) + f^-(x_2)} = 1 - 0.1 \times \frac{0.9}{0.9+0.05} \approx 0.905$$

Thus, the higher the estimated confidence of an instance, the larger the smoothing factor to be used for this instance, and vice versa. Algorithm 1 summarizes the steps of the proposed method *ibls*.

---

**Algorithm 1:** Instance-based label smoothing (*ibls*) with KDE

---

**Input:** Training set  $D_{tr} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$  such that  $\mathbf{x}_i \in \mathcal{X}$ ,  $y_i \in \mathcal{Y}$ , and  $d = |\mathbf{x}_i|$

**Result:** model  $\hat{s} : \mathcal{X} \rightarrow [0, 1]$  representing the positive class probability for any instance  $\mathbf{x} \in \mathcal{X}$

```

1  $D_{tr}^+ \leftarrow \{(\mathbf{x}, y) \in D_{tr} : y = 1\}$ ,  $D_{tr}^- \leftarrow \{(\mathbf{x}, y) \in D_{tr} : y = 0\}$ ;
2  $bw^+ \leftarrow \left(\frac{|D_{tr}^+| \times (d+2)}{4}\right)^{\frac{-1}{d+4}}$ ,  $bw^- \leftarrow \left(\frac{|D_{tr}^-| \times (d+2)}{4}\right)^{\frac{-1}{d+4}}$  [Sil86];
3  $kde^+ \leftarrow \text{null}$ ,  $kde^- \leftarrow \text{null}$ ;
4 foreach  $(\mathbf{x}_i, y_i) \in D_{tr}^+$  do
5   |  $kde^+ \leftarrow kde^+ + \text{Kernel}_{\text{Gaussian}}(\mu = \mathbf{x}_i, \sigma = bw^+)$ ;
6 end
7 foreach  $(\mathbf{x}_i, y_i) \in D_{tr}^-$  do
8   |  $kde^- \leftarrow kde^- + \text{Kernel}_{\text{Gaussian}}(\mu = \mathbf{x}_i, \sigma = bw^-)$ ;
9 end
10  $kde^+ \leftarrow \frac{kde^+}{|D_{tr}^+|}$ ,  $kde^- \leftarrow \frac{kde^-}{|D_{tr}^-|}$ ;
11  $\epsilon_{platt}^+ \leftarrow \frac{1}{|D_{tr}^+|}$ ,  $\epsilon_{platt}^- \leftarrow \frac{1}{|D_{tr}^-|}$ ;
12 foreach  $(\mathbf{x}_i, y_i) \in D_{tr}$  do
13   | if  $y_i = 0$  then
14     | |  $y_i \leftarrow \epsilon_{platt}^- \times \frac{kde^-.pdf(\mathbf{x})}{kde^+.pdf(\mathbf{x}) + kde^-.pdf(\mathbf{x})}$ ;
15     | else
16     | |  $y_i \leftarrow 1 - \epsilon_{platt}^+ \times \frac{kde^+.pdf(\mathbf{x})}{kde^+.pdf(\mathbf{x}) + kde^-.pdf(\mathbf{x})}$ ;
17     | end
18 end
19  $\hat{s} \leftarrow \text{logistic\_regression.fit}(D_{tr})$ ;
20 return  $\hat{s}$ ;

```

---

## 6 Label smoothing in classification with neural networks

The idea of using instance-based label smoothing in logistic regression seems to be more efficient than constant smoothing factors like Platt scaling. A similar approach inspired by this idea could also be applied in neural networks over multiple classes. In this section, we introduce how neural networks are trained with label smoothing. Then, we summarize the main advantages and drawbacks of traditional label smoothing on network performance. After that, we propose a new parametric approach for instance-based label smoothing, that aims to enhance the performance of standard label smoothing.

### 6.1 Standard label smoothing: Fors and Againsts

While label smoothing is used in logistic regression fitting to reduce overconfidence problems, a similar approach is widely adopted in neural network models to improve accuracy across a wide range of tasks like image classification and neural machine translation [MKH19]. Similar to logistic regression, during training a neural network, the cross-entropy loss is minimized with softened target labels. The original hard target label of an instance is denoted by  $\mathbf{y}$ , where  $y_k = 1$  if  $k$  is the index of the true class out of  $K$  classes, and 0 otherwise. However, in a network trained with label smoothing, the softened target probability value is computed as  $y_k^{LS} = y_k(1 - \epsilon) + \frac{\epsilon}{K}$  where  $\epsilon$  is the smoothing factor [SVI<sup>+</sup>16]. As a rule of thumb, the value of used  $\epsilon$  is usually set to 0.01, 0.05, or 0.1 [MKH19]. Figure 19 shows an example of the label smoothing of an instance in a 3-class dataset. Thus, label smoothing encourages the softmax output probability vector to be close to the probability of the correct class (dog=0.933) but at the same time, *equally distant* from the remaining classes (equal probabilities for cat & ship).

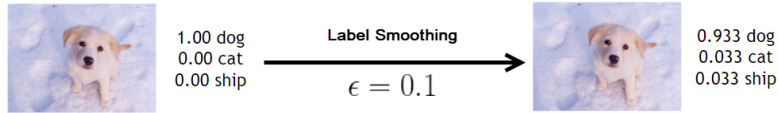


Figure 19. Example of label smoothing for an instance with a true label dog in a 3-class dataset.

Label smoothing has shown a positive effect on model generalization, and calibration of the predicted probabilities [GBC16]. However, it pushes the network towards treating all the incorrect classes as equally probable while training with hard target labels does not similarly force this structure. For instance, a cat to a dog is not equally distant as a cat to a ship. Hence, label smoothing leads to a loss in the information about the dataset class similarity structure.

Muller *et al.* [MKH19] quantified this loss of information using an approximation to the mutual information function between the input instances of two arbitrary classes, and the continuous output representation for the difference between the two logits (out of  $K$  classes) of these two classes. Logits are the score values produced from the penultimate layer units of the neural network. A decrease in mutual information value was found when training the network with

label smoothing. Consequently, label smoothing was found to impair the network performance in transfer learning and network distillation. [MKH19, KSL19].

Network distillation is the process of compressing a large trained neural network (teacher network) into a smaller network (student). The student network is trained with targets equal to the teacher’s predictions, so it tries to learn how to behave like the teacher network. This process is widely adopted to train more efficient networks that can be deployed on mobile and edge devices to make predictions quickly [HVD15]. A student AlexNet convolutional neural network, distilled from a teacher ResNet-56 network trained with label smoothing on CIFAR-10 dataset, was shown to have less accuracy than a teacher network trained with temperature scaling and without label smoothing. The impairment in the accuracy becomes more significant as the smoothing factor increases [MKH19]. Kornblith *et al.* [KSL19] trained an Inception network on ImageNet dataset with different training settings. The penultimate layer features of the trained network on different pairs of classes were transferred to be used in fitting a logistic regression model. The fitted models’ average accuracy was the worst when the Inception network was trained with label smoothing and high dropout probabilities.

## 6.2 Instance-based label smoothing in neural networks

We showed that the main disadvantage of the standard label smoothing in neural networks is the constant smoothing factor for all the incorrect classes. This encourages the instances of the same class to lie in a tight cluster, which is equally separated from all other classes. Thus, some information related to the class similarity structure is erased, which is mainly reflected in impairing the performance of transfer learning or network distillation.

Instead, we propose a new instance-based label smoothing approach, where the target probability distribution is not uniformly distributed among incorrect classes. Figure 20 shows the 3-class dataset of dogs, cats, and ships. Intuitively, the similarity distance from a dog to a cat is much smaller than that from a dog to a ship. So, if the input instance is a dog, the required target probability vector has to favor the cats’ class more than the ships’ class according to the similarity between this dog instance to the instances of both classes. Following this approach, we claim that in addition to keeping the advantages of label smoothing, the information carried by the similarity distances among different classes is still preserved by choosing the corresponding target probability distribution for each instance that keeps the class similarity structure of the dataset.

Next, we explain how the target probability distribution vector of the smoothed labels ( $y_k^{ILS}$ ) is computed. Let  $x$  denote an input instance,  $\epsilon$  is an initial constant smoothing factor similar to what has been used in standard label smoothing,  $f_k(x)$  is the  $k^{th}$  logit output of the same architecture network trained *without* label smoothing that corresponds to the  $k^{th}$  class score value of the instance  $x$ , and  $t$  is the index of the correct class out of  $K$  classes. We aim to replace the uniformly distributed target probabilities of the incorrect classes. So, each incorrect class will be assigned a target probability proportional to the output score of this class relative to all the remaining classes using a network trained with cross-entropy loss on the hard target labels (without label smoothing).

First, a factor  $q$  is subtracted from the output logits of the network trained without label

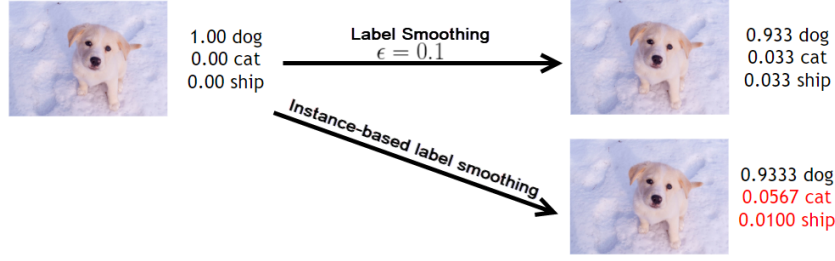


Figure 20. An example showing the difference between the standard label smoothing with a constant smoothing factor, and the instance-based label smoothing.

smoothing. this transformation is done to ensure that all the logits values are  $\geq 0$ , where  $q = \min(f_1(x), f_2(x), \dots, f_K(x))$ . This step is crucial for the next steps of the instance-based smoothing factors calculation, as it will be assumed that all the logits score values are not negative. Also, the translated logits are scaled with a scaling factor  $u_{scale}$ . Different translation and scaling parameters  $(q, u_{scale})$  will result at the end in different class similarity structures, as shown next. It is required to tune the parameters  $(q, u_{scale}, \epsilon)$  until we get the network model with the best validation performance:

$$f'_i(x) = u_{scale}(f_i(x)) - q \quad \forall i \in \{1, 2, \dots, K\}.$$

Second, the correct class of index  $t$  is assigned a smoothed target label with a smoothing factor of  $\epsilon^{ILS}$ . This smoothing factor is computed from the initial standard smoothing factor  $\epsilon$  after multiplying it by the ratio between the scaled logit value of the correct class (the score that  $x$  belongs to class  $t$ ) to the sum of the values of logits of all the classes. This particular choice is similar to what we did in section 5 with logistic regression. The target probability of the correct class depends on the estimated confidence of the network trained without label smoothing. The more confident the network is, the higher the smoothing factor to be used to reduce its overconfidence and vice versa, as shown below:

$$\epsilon^{ILS} = \frac{f'_t(x)}{\sum_l^K f'_l(x)} \cdot \epsilon$$

$$y_k^{ILS} = (1 - \epsilon^{ILS}), \text{ where } k = t.$$

Third, the target probability distribution of the incorrect classes is calculated as the ratio between the score of each class ( $f'_k(x)$ ) to the sum of the scores of all the incorrect classes  $(\sum_l^K f'_l(x)) - f'_t(x)$  as follows:

$$y_k^{ILS} = \epsilon^{ILS} \frac{f'_k(x)}{(\sum_l^K f'_l(x)) - f'_t(x)}, \text{ where } k \neq t.$$

Following this method ensures that the target smoothed probabilities of the incorrect classes will amount to  $\epsilon^{ILS}$ , and the target smoothed label of the correct class is  $1 - \epsilon^{ILS}$ . Thus, the sum of the target smoothed probability vector is equal to 1. Additionally, the proposed instance-based

label smoothing procedure will not keep the probability distribution of the incorrect classes uniformly distributed. However, the computed soft targets can preserve a similar class similarity structure to the original dataset with the help of the outputs of the network trained without label smoothing.

Figure 21 illustrates the process of training the network with smoothed target probabilities that are computed in proportional with the output probability distribution from the same network trained without label smoothing (Instance-based label smoothing). Our proposed method is a straightforward way out of many possible other methods that can smooth target labels in different ways. However, our proposed method aims to use label smoothing to avoid the trained network overconfidence while taking into account the class similarity structure, which is erased in case of standard constant label smoothing.

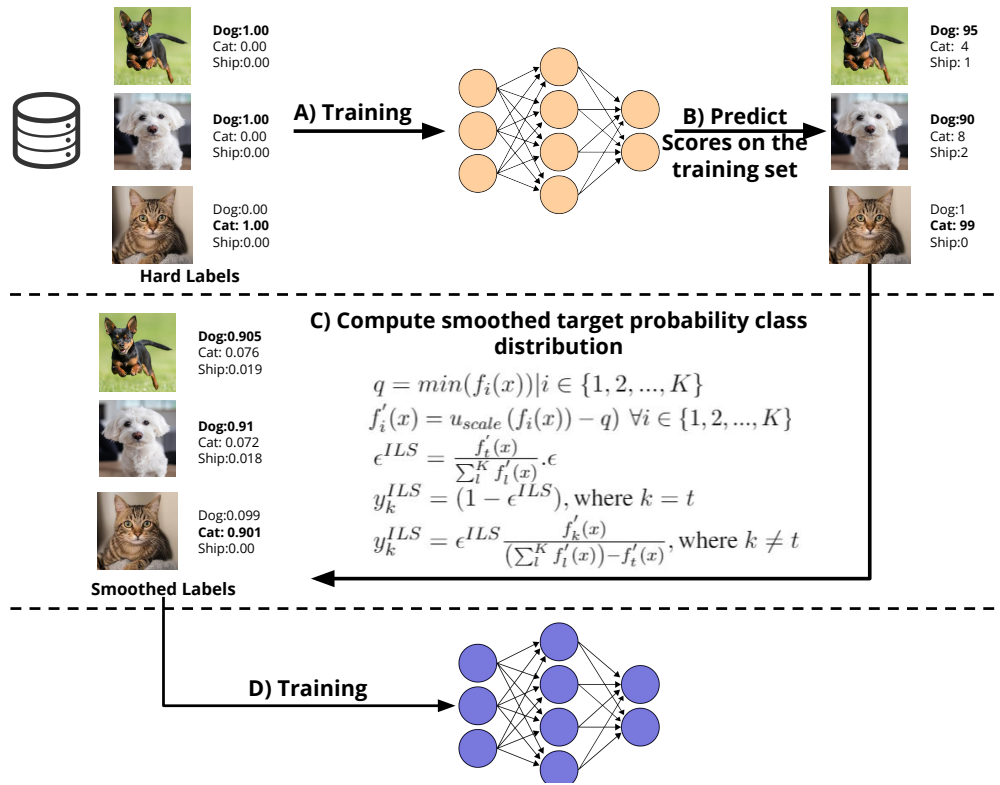


Figure 21. Example of the instance-based label smoothing process starting with A) fitting a network without label smoothing B) using the learned probability distributions of the instances to help in C) computing the target probability distribution with label smoothing for each instance, and finally, D) training the network again with the smoothed labels.

## 7 Experimental results

In Section 4, the optimal probability predictions were derived if the distribution of the generative model of the dataset is known. Later, this derived formula was approximated using Monte Carlo samples from the generative model distribution to be used practically (*bayes\_preds*). This method was further simplified into *bayes\_coeff* and *bayes\_map* to make faster predictions during inference. In this section, our approximated formula is evaluated against different regularization techniques for LR fitting including our proposed instance-based label smoothing using kernel density estimation (*ibls*) (Section 5).

Table 3 lists the different methods evaluated in our experiments along with their descriptions. It is worth mentioning that the last three highlighted methods (*bayes\_preds*, *bayes\_coeff*, *bayes\_map*) have the advantage of being *informed* about the exact distribution of the generative model of the dataset over the other listed methods (*normal*, *smoothed*, *L1*, *L2*, *ibls*, *cauchy\_map*), which are *non-informed* about this distribution.

Since our derivation only holds when the generative model distribution is known, in practice, having such knowledge is the exception, not the rule. Thus, in Section 7.2, we evaluate the different regularization methods listed in Table 3 on 40 real datasets except for the informed methods as we do not know the generative model distribution of these datasets. We compared the performance of the evaluated methods after calibrating the scores of a linear SVM model fitted on these datasets once. Moreover, we fit multivariate LR models directly over the whole list of features of the datasets at another time.

Method	Description
<b>normal</b>	Vanilla logistic sigmoid curve (Section 2.4)
<b>platt</b>	Label smoothing using Platt scaling (Section 3.2.3)
<b>smoothed</b>	Label Smoothing with a fixed smoothing factor selected from a set of 40 smoothing factors that have a range from $1e^{-5}$ to $1e^{-1}$ , and selected based on 10-fold stratified cross-validation (Section 3.2.3)
<b>L1</b>	L1 regularization for logistic sigmoid coefficients with a penalization parameter that belongs to $\{10^{-5}, 10^{-4}, 10^{-3}, \dots, 10^3\}$ selected based on 10-fold stratified cross-validation (Section 3.2.1)
<b>L2</b>	L2 regularization for logistic sigmoid coefficients with a penalization parameter that belongs to $\{10^{-5}, 10^{-4}, 10^{-3}, \dots, 10^3\}$ selected based on 10-fold stratified cross-validation (Section 3.2.1)
<b>ibls</b>	Instance-based label smoothing using KDE (Section 5)
<b>cauchy_map</b>	Bayesian logistic regression with the maximum a posteriori logistic sigmoid curve using Cauchy prior for model coefficients (Section 3.2.2) [GJP <sup>+</sup> 08]
<b>bayes_preds</b>	The posterior mean of output probability predictions of optimal logistic sigmoids (Section 4)
<b>bayes_coeff</b>	The posterior mean of coefficients of optimal logistic sigmoids (Section 4)
<b>bayes_map</b>	Maximum a posteriori optimal logistic sigmoid (Section 4)

Table 3. Description of the different evaluated methods that reduce the overconfidence problem in logistic regression fitting. The shaded methods represent informed methods that have information about the exact generative model distribution of the dataset.

Finally, in Section 7.3, we test our proposed method for the instance-based label smoothing with convolutional neural networks over multiple image classification datasets. Multiple experiments are carried out where the accuracy, log loss, calibration error, and distillation performance are reported and discussed for our method compared with networks trained with/without standard label smoothing, and with/without temperature scaling.

To ensure the reproducibility of the results, the source codes implemented and used in our evaluation, in addition to all the log files containing the detailed numerical results, have been



made available<sup>3</sup>. The experiments in this section were carried out in part in the high-performance computing center of the University of Tartu<sup>4</sup>.

## 7.1 Synthetic datasets experiments

In these experiments, we aim to investigate and answer the following questions:

- Q1: How the performance of the evaluated methods changes by increasing the separation between the classes' likelihoods?
- Q2: How the performance of the evaluated methods changes by increasing the size of the training set  $|D_{tr}|$ ?
- Q3: How the performance of the evaluated methods changes by altering the class balance ratio  $clr_{tr} (\frac{|D_{tr}^+|}{|D_{tr}^+|+|D_{tr}^-|})$ ?

### 7.1.1 Setup

Parameter	Value	Parameter	Value
$\mathcal{D}_{\mu_0}$	$\sim \mathcal{U}(u_0, 0)$ or $\mathcal{B}(a_0, b_0)$	$\mathcal{D}_{\mu_1}$	$\sim \mathcal{U}(0, u_1)$ or $\mathcal{B}(a_1, b_1) + shift$
$\mathcal{D}_{\sigma}$	$\{1\}$	Number of generated datasets	$10^3$
$D_{tr}^+, D_{tr}^-$	$\sim \mathcal{N}(\mu_0, 1)$	$D_{tr}^+, D_{te}^+$	$\sim \mathcal{N}(\mu_1, 1)$
$ D_{tr} $	50, 100, 500	$ D_{te} $	$10^4$
$clr_{tr}$	0.5, 0.7, 0.9	$clr_{te}$	0.5
Evaluation Metrics	Log loss, ECE	$N_{MonteCarlo}$	$10^5$

Table 4. Experimental setup for the synthetic datasets.

The setup of these experiments starts by defining the distribution of the generative models of the datasets  $\mathcal{D}_{\theta} = \{\mathcal{D}_{\mu_0}, \mathcal{D}_{\mu_1}, \sigma\}$ . Beta ( $\mathcal{B}$ ) and uniform ( $\mathcal{U}$ ) distributions were selected for  $\mathcal{D}_{\mu_0}, \mathcal{D}_{\mu_1}$  while  $\mathcal{D}_{\sigma}$  is set to  $\{\delta(t - 1)\}$  for simplicity. The size of training set ( $|D_{tr}|$ ) was varied to be 50, 100, 500, and the class balance ratio ( $clr_{tr} = \frac{|D_{tr}^+|}{|D_{tr}^+|+|D_{tr}^-|}$ ) was also varied to be 0.5, 0.7, 0.9. The testing set  $|D_{te}|$  was fixed to be balanced with  $10^4$  instances in all the experiments. Each experiment was repeated for 1,000 different generated datasets. The number of Monte Carlo samples  $N_{MC}$  of dataset parameter configurations drawn out of the generative model prior distribution was set to  $10^5$  for *bayes\_preds*, *bayes\_coeffs* and *bayes\_map*. Table 4 summarizes the used configurations through all the experimental setup in the synthetic datasets experiments.

The critical difference (CD) diagrams of different methods ranks were reported for all experiments according to the log loss and the expected calibration error at ten bins.

CD diagram is a visualization tool for the results of the *Wilcoxon-Holm* posthoc analysis. This statistical analysis is designed to identify the pairwise statistical significance among a set of

<sup>3</sup><https://github.com/mmaher22/Instance-based-smoothing>

<sup>4</sup><https://hpc.ut.ee/>

methods on a set of predictive tasks. Hence, a CD diagram shows a comparison between these methods' performance on the specified set of tasks. The null hypothesis to be tested is that the average ranks of each pair of methods do not differ with statistical significance. The analysis starts by computing the Friedman nonparametric statistical test to detect the differences among the methods across multiple tasks and test the null hypothesis over the entire set of methods [Mil39]. Then, the Wilcoxon signed-rank test is applied between each pair of methods to assess whether the ranks of their population mean differ [Wil92], Holm correction is finally applied to reject the null hypothesis [Hol79].

An example of the resulting plots is shown in Figure 22. The plot has separate lines for each of the evaluated methods. Each line starts with the label of a specific method and ends on the x-axis representing the *average rank* position of the respective method performance across all the tasks. Horizontal bold lines connecting some methods represent the set of connected methods such that the test has failed to exclude the null hypothesis. On the other hand, any pair of unconnected methods have a different average rank with statistical significance.

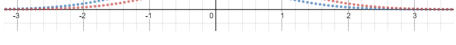
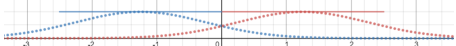
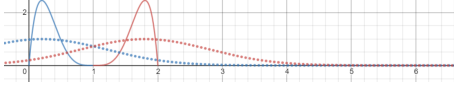
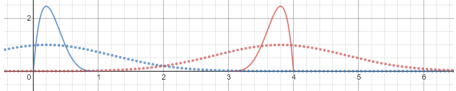
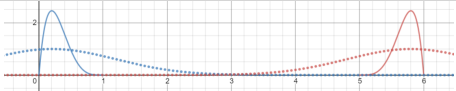
Subplot	$\mathcal{D}_{\mu_0}$	$\mathcal{D}_{\mu_1}$	Generative Model	$ D_{tr} $	$clr_{tr}$
a	$\sim \mathcal{U}(-0.5, 0)$	$\sim \mathcal{U}(0, 0.5)$		50	0.5
b	$\sim \mathcal{U}(-2.5, 0)$	$\sim \mathcal{U}(0, 2.5)$		50	0.5
c	$\sim \mathcal{B}(2, 5)$	$\sim \mathcal{B}(5, 2) + 1$		50	0.5
d	$\sim \mathcal{B}(2, 5)$	$\sim \mathcal{B}(5, 2) + 3$		50	0.5
e	$\sim \mathcal{B}(2, 5)$	$\sim \mathcal{B}(5, 2) + 5$		50	0.5

Table 5. Setup of the experiments used to answer how the performance of the evaluated methods changes by increasing the separation between the classes' likelihoods. The training set size and the class ratio are kept constant, while the separation between classes is changed with different generative model distributions. Solid lines are the generative model distributions. Dashed lines are examples of a randomly picked sample from the generative model. Blue and red colors represent negative and positive classes respectively. Uniform and beta distributions are denoted as  $\mathcal{U}$ ,  $\mathcal{B}$  respectively. The  $+1$ ,  $+3$ ,  $+5$  in the beta distributions of the  $\mathcal{D}_{\mu_1}$  column, represent the amount of shift for these distributions in the positive x-axis direction.

## 7.1.2 Results

### Q1: How the performance of the evaluated methods changes by increasing the separation between the classes' likelihoods?

To discover how the separation between the distributions of the positive and negative classes affects the performance of the evaluated methods, five experiments summarized in Table 5 were carried out using different generative model distributions  $\mathcal{D}_\theta$ . Figure 22 summarizes the results of these experiments where subplots (a) and (b) correspond to uniform generative model distribution but with ranges where the separation problem is more probable in case (b). Similarly, subplots (c, d, e) show the results when using asymmetric beta distributions for the generative model. The separation between the classes' distributions increases as we go from (c) to (e).

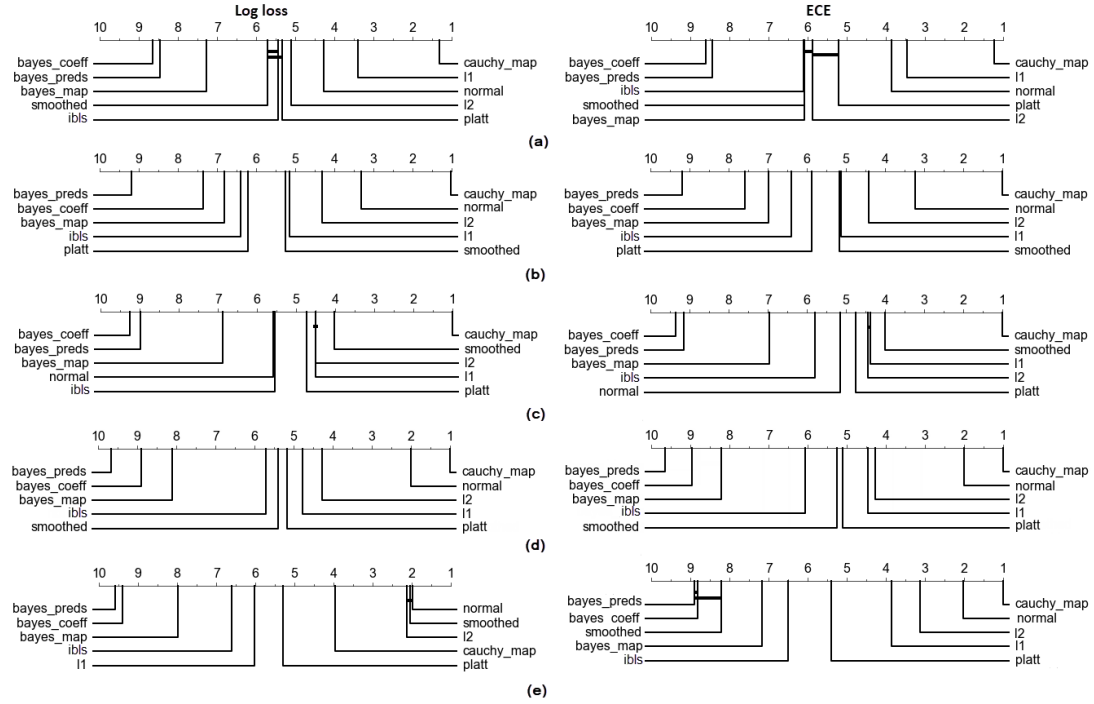


Figure 22. CD diagrams for log loss (left) and ECE (right) - (higher rank is better). Different separation distances between classes' likelihoods for each subplot (a)-(e) (refer to Table 5). Bayesian derived methods with the same prior as generative models are significantly the best ones. If the separation between classes distributions is small, *normal*, *ibls*, *smoothed* are the best options in non-informed methods but if it is large, *ibls*, and *platt* are the best ones.

The approximated formula for the derived optimal predictions (*bayes\_preds*) has significantly better average ranking over all other methods in terms of both log loss and ECE in all the experiments. Both the other informed methods (*bayes\_map*, *bayes\_map*) come second, and third respectively.

It is worthwhile to mention that *bayes\_preds*, *bayes\_coeff* and *bayes\_map* are informed about the generative model distribution of the datasets, while the remaining methods do not have this extra information. Thus, when comparing the other methods, the *ibls* is consistently better than *platt* in all the experiments. Using a uniform generative model prior (subplots a, b), at a small separation distance between positive and negative classes' distributions (a), the *smoothed* version of LR has the best performance in all the non-informed methods. This agrees with the work of Liina Partel [Pä19] that in many datasets, there exists a better smoothing factor than Platt scaling, where this factor is dependent on the separation between classes.

On the other hand, when the distance between the classes' likelihoods increases, the separation problem becomes more probable (subplot b). So, all the positive class instances are fully separated from the negative class ones. In this case, *smoothed* average rank decreases relative to *platt*. This can be interpreted as the best smoothing factor chosen by *smoothed* is always the smallest possible factor in the defined range since the classes of the validation set are also indeed fully separated. Consequently, the learned model does not benefit from label smoothing on the separated validation set, and it tends to select the smallest possible factor.

Experiment #	a	a	b	b	c	c	d	d	e	e
Method	Log loss	ECE	Log loss	ECE	Log loss	ECE	Log loss	ECE	Log loss	ECE
<i>bayes_coeff</i>	0.6605±0.024	<b>0.0308±0.0175</b>	0.2958±0.1831	0.0318±0.0258	<b>0.4928±0.0489</b>	<b>0.0258±0.0126</b>	0.1159±0.0252	0.0102±0.0068	<b>0.0109±0.004</b>	<b>0.0021±0.001</b>
<i>bayes_map</i>	0.6647±0.0254	0.0482±0.0191	0.297±0.1837	0.035±0.027	0.4992±0.0512	0.047±0.0159	0.1189±0.0273	0.0154±0.0113	0.0126±0.0052	0.0036±0.0019
<i>bayes_preds</i>	<b>0.6604±0.0237</b>	0.0311±0.0179	<b>0.1846±0.1616</b>	<b>0.0096±0.0198</b>	0.4929±0.0486	0.0264±0.0122	<b>0.0342±0.0538</b>	<b>0.0031±0.0059</b>	<b>0.0109±0.004</b>	<b>0.0021±0.0011</b>
<i>cauchy_map</i>	0.7238±0.015	0.1408±0.0425	0.7722±0.0296	0.4012±0.1049	0.7581±0.0227	0.2948±0.038	0.7846±0.0268	0.4917±0.0219	0.7679±0.0005	0.5272±0.0026
<i>ibls</i>	<b>0.6700±0.0285</b>	<b>0.0509±0.0152</b>	<b>0.2972±0.1805</b>	0.0398±0.02	<b>0.5005±0.0481</b>	<b>0.0534±0.0126</b>	<b>0.1263±0.0259</b>	0.0278±0.0121	<b>0.0777±0.0043</b>	0.0656±0.0019
<i>L1</i>	0.6885±0.036	0.0754±0.0343	0.3021±0.1838	0.0454±0.0208	0.5028±0.0487	0.0586±0.0083	0.1285±0.0266	0.0307±0.0136	0.0789±0.0047	0.0626±0.0029
<i>L2</i>	0.6771±0.0261	0.0518±0.0317	0.306±0.1779	0.0499±0.0201	0.5029±0.0479	0.0588±0.0078	0.136±0.0284	0.0395±0.0187	∞	<b>0.0025±0.0009</b>
<i>normal</i>	0.6714±0.0288	0.059±0.0142	0.3124±0.1734	<b>0.0391±0.0207</b>	0.5007±0.0482	0.054±0.0125	<b>0.1266±0.025</b>	<b>0.0268±0.0113</b>	∞	0.0027±0.0011
<i>platt</i>	0.6705±0.0282	0.0537±0.0154	0.3132±0.1676	0.0598±0.0125	0.5023±0.0471	0.057±0.0078	0.1552±0.0247	0.0621±0.0142	0.0807±0.0045	0.0677±0.0018
<i>smoothed</i>	0.6701±0.0274	0.0519±0.0266	0.3014±0.1806	0.0463±0.0194	0.5065±0.048	0.0639±0.0109	0.127±0.0262	0.0293±0.0109	∞	0.0026\$pm\$0.0009

Table 6. Mean  $\pm$  standard deviation log loss and ECE of the evaluated methods on synthetic datasets with different separation distance between the classes' likelihoods. Refer to Table \ref{tab:exp-setup1} for the experiments setup. Methods informed with the generative model distribution are highlighted in a different color.

Regarding the experiments of generative models with beta distributions, the distributions were selected to be asymmetric in contrast with the uniform distributions in the previous case. In subplot (c), *normal* LR has the best performance among the non-informed methods followed by *ibls*. However, as the separation distance between the classes becomes larger (d, e), the separation problem becomes more probable in many of the generated datasets. The performance of *normal* is the worst as the model becomes overconfident with probability predictions of almost 0, 1 for negative, and positive classes, respectively.

Table 6 average log loss and calibration error values for all the evaluated methods. Regarding the informed methods, *bayes\_coeff* has a very close performance to the approximated formula for the optimal predictions (*bayes\_preds*), and both methods are significantly outperforming all the non-informed methods. Infinite log loss occurred multiple times for wrong and confident predictions made by *normal*, *smoothed*, and *L2* methods when the separation between the classes was the maximum. Additionally, *L2* has less regularization effect than *L1* where the model coefficients can reach zero with *L1* regularization but the opposite isn't possible. Our instance-based label smoothing achieved the best average loss in all the experiments and the best average calibration error in two of them. However, it is consistently outperforming Platt scaling (*platt*).

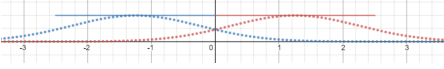
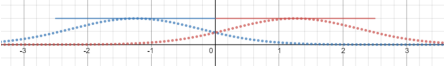
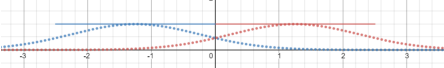
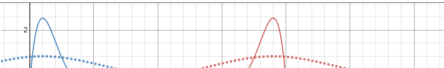

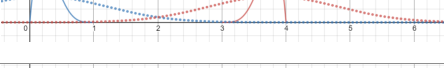
Subplot	$\mathcal{D}_{\mu_0}$	$\mathcal{D}_{\mu_1}$	Generative Model	$ D_{tr} $	$clr_{tr}$
a	$\sim \mathcal{U}(-2.5, 0)$	$\sim \mathcal{U}(0, 2.5)$		50	0.5
b	$\sim \mathcal{U}(-2.5, 0)$	$\sim \mathcal{U}(0, 2.5)$		100	0.5
c	$\sim \mathcal{U}(-2.5, 0)$	$\sim \mathcal{U}(0, 2.5) + 1$		500	0.5
d	$\sim \mathcal{B}(2, 5)$	$\sim \mathcal{B}(5, 2) + 3$		50	0.5
e	$\sim \mathcal{B}(2, 5)$	$\sim \mathcal{B}(5, 2) + 3$		100	0.5
f	$\sim \mathcal{B}(2, 5)$	$\sim \mathcal{B}(5, 2) + 3$		500	0.5

Table 7. Setup of experiments used to answer how the performance of evaluated methods changes by increasing the number of training set instances. The class ratio is kept constant with two different generative model distributions. Solid lines are the generative model distributions. Dashed lines are examples of a randomly picked sample from the generative model. Blue and red represent negative and positive classes respectively. Uniform and beta distributions are denoted as  $\mathcal{U}$ ,  $\mathcal{B}$  respectively. The  $+3$  in the beta distributions of the  $\mathcal{D}_{\mu_1}$  column, represent the amount of shift for these distributions in the positive x-axis direction.

**Q2: How the performance of the evaluated methods changes by increasing the size of the training set  $|D_{tr}|$ ?**

As the training set size increases, it becomes more representative of the whole population distribution. In that case, it is more reasonable to have less regularization. To evaluate the effect of the training set size on the evaluated methods, we used the same two generative models of beta ( $\mathcal{B}$ ) and uniform ( $\mathcal{U}$ ) distributions as the previous experiment. The separation distance between the generative models of positive and negative classes was fixed during this experiment. Then, we generated training sets with  $clr_{tr}$  equals to 0.5 (balanced), 0.7, and 0.9 (90% positive instances). Table 7 summarizes the setup configuration of this experiment.

Figure 23 includes the CD diagrams of the methods average ranking. As the training set size increases, the *normal* LR average rank achieves a leap in both log loss and ECE. Since separation problems are more probable to occur at small training set sizes, this can affect the performance of *normal* profoundly. However, as the number of instances increases to 500 (subplots (c, f)), the training set becomes more representative of the whole distribution of the test set, which means

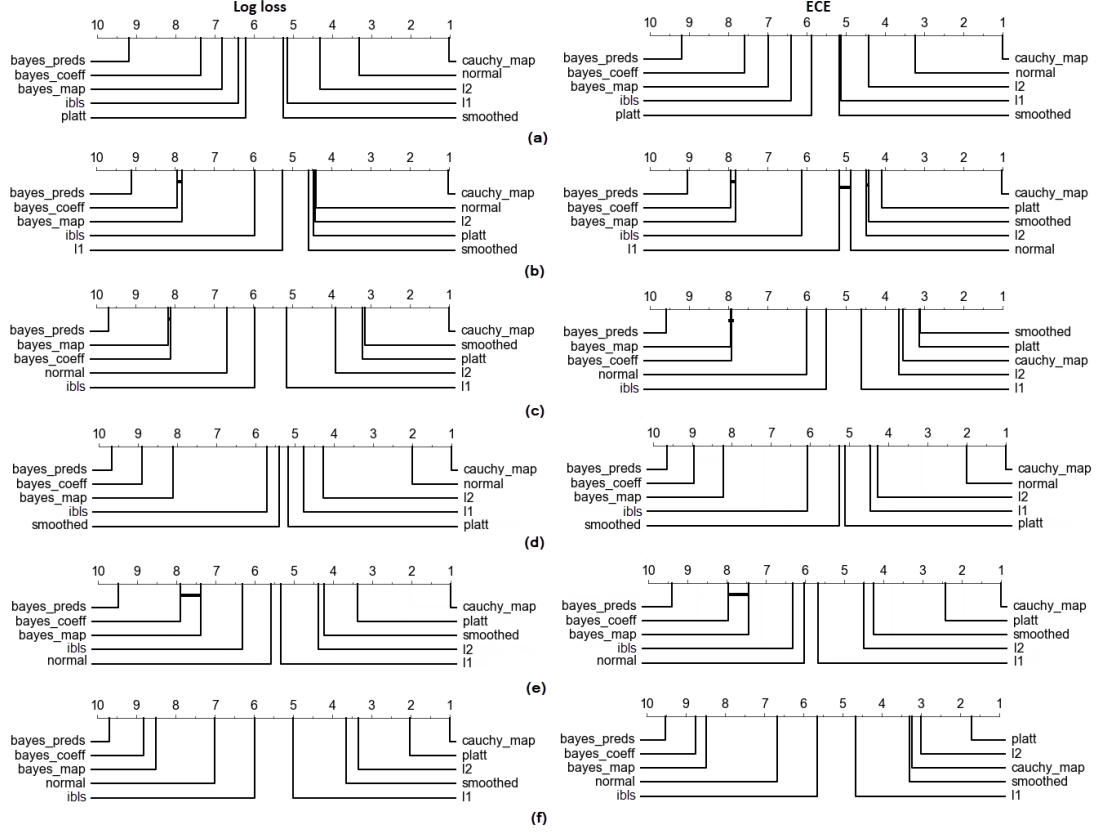


Figure 23. CD diagrams for log loss (left) and ECE (right) - (higher rank is better). Different training set sizes for subplots (a)-(f) (refer to Table 7). Bayesian derived methods with the same prior as generative models are significantly, and consistently the best ones. As the training set size becomes larger, *normal* becomes the best method as training set distribution becomes more representative of the testing set. Otherwise, *ibls* is the regularization method to use.

that no regularization is needed. As it is always difficult to tell how many instances are enough, we still need to use one of the regularization methods. *ibls* is considered the best to be used among the non-informed methods with an average rank that is consistently higher than other methods in all experiments except for the largest training set when it comes second after *normal*. All numerical results of average loss and calibration errors can be found in our repository (Appendix: Section 8).

**Q3: How the performance of the evaluated methods changes by altering the class balance ratio  $clr_{tr} (\frac{|D_{tr}^+|}{|D_{tr}^+| + |D_{tr}^-|})$ ?**

As *Platt* scaling and *ibls* instance-based smoothing take into account the class balance ratio *clr*, we aim in this experiment to evaluate how these methods perform compared with other

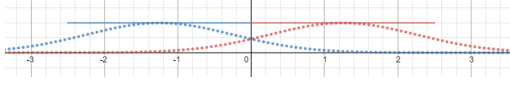
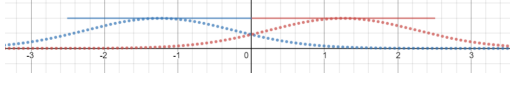
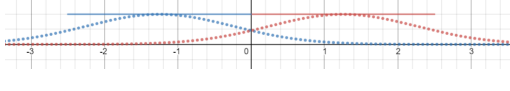
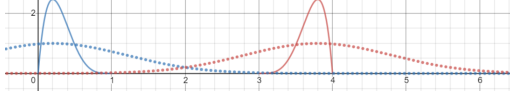
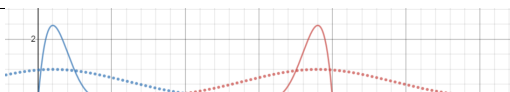

Subplot	$\mathcal{D}_{\mu_0}$	$\mathcal{D}_{\mu_1}$	Generative Model	$ D_{tr} $	$clr_{tr}$
<b>a</b>	$\sim \mathcal{U}(-2.5, 0)$	$\sim \mathcal{U}(0, 2.5)$		100	0.5
<b>b</b>	$\sim \mathcal{U}(-2.5, 0)$	$\sim \mathcal{U}(0, 2.5)$		100	0.7
<b>c</b>	$\sim \mathcal{U}(-2.5, 0)$	$\sim \mathcal{U}(0, 2.5) + 1$		100	0.9
<b>d</b>	$\sim \mathcal{B}(2, 5)$	$\sim \mathcal{B}(5, 2) + 3$		100	0.5
<b>e</b>	$\sim \mathcal{B}(2, 5)$	$\sim \mathcal{B}(5, 2) + 3$		100	0.7
<b>f</b>	$\sim \mathcal{B}(2, 5)$	$\sim \mathcal{B}(5, 2) + 3$		100	0.9

Table 8. Setup of experiments used to answer if changing the training set class balance ratio  $clr$  affects the performance of evaluated methods. The training set size is kept constant with two different generative model distributions. Solid lines are the generative model distributions. Dashed lines are examples of a randomly picked sample from the generative model. Blue and red represent negative and positive classes respectively. Uniform and beta distributions are denoted as  $\mathcal{U}$ ,  $\mathcal{B}$  respectively. The  $+3$  in the beta distributions of the  $\mathcal{D}_{\mu_1}$  column, represents the amount of shift for these distributions in the positive x-axis direction.

regularization methods at different  $clr$ . As reported in Table 8, we used the same generative model distributions to generate training sets of 100 instances. The  $clr$  of these datasets was varied from 0.5 (balanced) to 0.7 and 0.9 (90% positive). The CD diagrams of the average ranking in terms of log loss and ECE are shown in Figure 24. All numerical results of average loss and calibration errors can be found in our repository (Appendix: Section 8).

As shown in Figure 24, as the training set becomes more imbalanced (subplots c,f), *Platt*, and *ibls* average ranks become significantly better than other regularization methods like *L1*, *L2*, and the Bayesian logistic regression with a Cauchy prior for the model coefficients *cauchy\_map* especially in terms of log loss. The best-performing methods are still *bayes\_preds*, *bayes\_coeff*, and *bayes\_map* as the test set is already known to be balanced. This information is already taken into account by these methods (the Bayes-optimal model for each random sample drawn from the

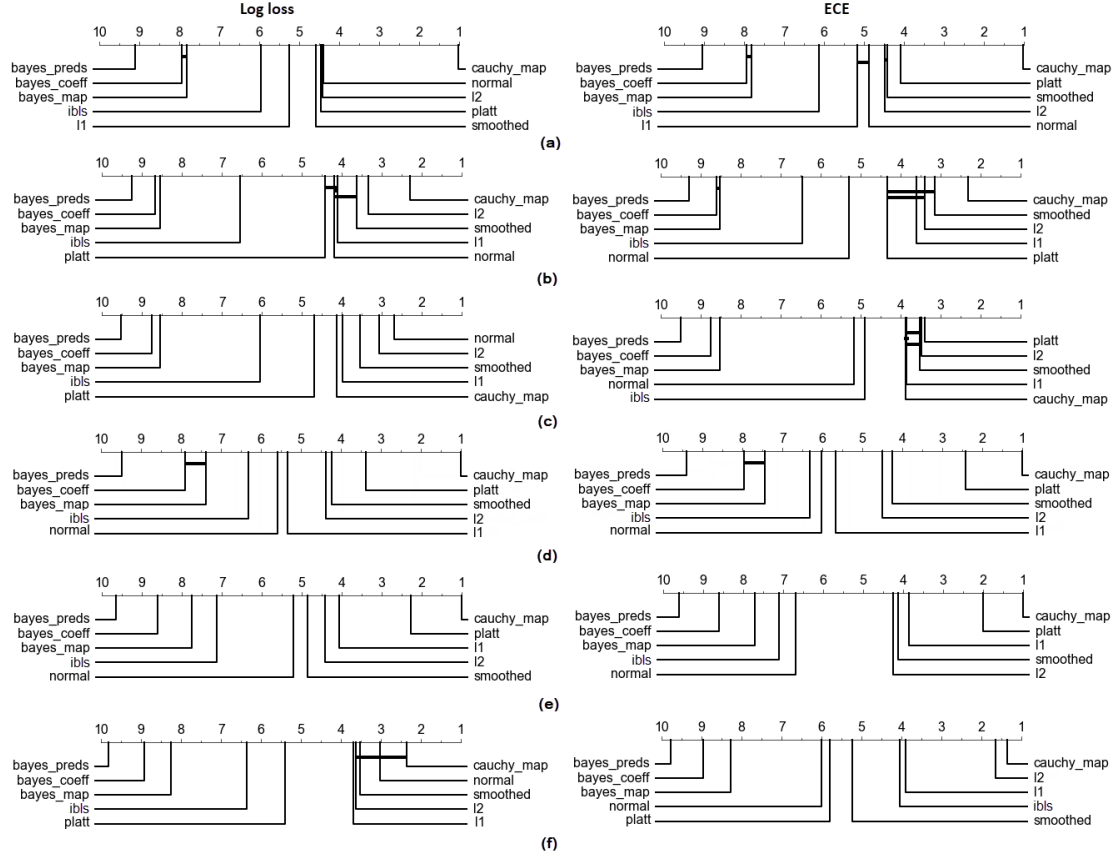


Figure 24. CD diagrams for log loss (left) and ECE (right) - (higher rank is better). Different training set class ratios for subplots (a-d) (refer to Table 8). Bayesian derived methods with the same prior as generative models are significantly the best ones. Instance-based smoothing (*ibls*) and Platt scaling (*platt*) are the best non-informed methods with different class ratios of the training set.

generative model is derived from a balanced dataset).

Overall, in all the experiments, the informed methods that approximate the formula for the derived optimal probabilities (*bayes\_preds*, *bayes\_coeff*, *bayes\_map*) are significantly better than all other non-informed methods at training set sizes, class balance ratio, and the separation distance between the distributions of the classes' likelihoods. It is also clear that the posterior mean of the coefficients of the sampled Bayes-optimal models (*bayes\_coeff*) offers an excellent approximation to the posterior mean of the Bayes-optimal models' predictions (*bayes\_preds*). At the same time, *bayes\_coeff* is much faster during inference as it requires the evaluation of predictions using only a single model instead of  $N_{MC}$  models using *bayes\_preds*.

Unfortunately, the distribution of the generative models of the datasets is rarely known. Thus, other non-informed regularization methods are still essential in real-life scenarios. Our proposed



instance-based smoothing method (*ibls*) outperforms other regularization methods like Platt scaling (*platt*) significantly in most of the synthetic datasets.

## 7.2 Real datasets experiments for logistic regression

In these experiments, we aim to investigate and answer the following questions:

- Q1: What is the relative performance of different LR overconfidence reduction methods for calibrating SVM model scores on real datasets?
- Q2: How do the performance of these methods change for univariate data like logistic calibration, and multi-dimensional data like fitting LR directly on the whole dataset?

### 7.2.1 Setup

In the following two experiments, a set of 40 benchmark classification datasets, was collected from OpenML<sup>5</sup> [VvRBT13]. We set restrictions on all the selected datasets to have exactly 2 classes, numerical features only, no missing data, and a maximum of 50000 instances and 50 features for computational resources constraints. A histogram for the number of instances, number of features, and class balance ratio can be seen in Figure 25 where the selected datasets cover the restricted space mentioned. Detailed information about these datasets can be found in our open source repository (Appendix: Section 8). All datasets were standardized, and divided into 70 – 30% training, testing stratified splits. All experiments were repeated 5 times for different splits, and the average of log loss and ECE are reported. The methods reported in Table 3 were compared except for the informed methods (*bayes\_preds*, *bayes\_coeff*, *bayes\_map*), as we do not have information about the distribution of the generative models of these datasets.

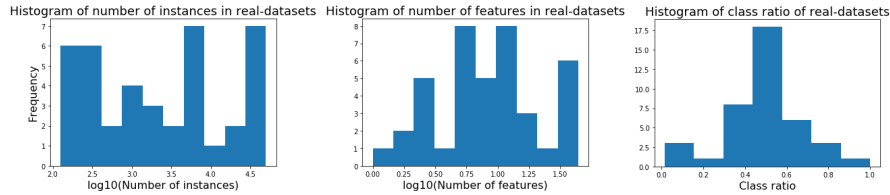


Figure 25. Histogram for properties of the selected real-world datasets.

### 7.2.2 Results

In the first experiment, the training set was again divided randomly into 70 – 30% stratified splits. The larger split is used in fitting linear soft-margin SVM model from scikit-learn Python library<sup>6</sup> with a regularization parameter set to 1. The smaller splits were used for logistic calibration of

<sup>5</sup><http://openml.org/>

<sup>6</sup><https://scikit-learn.org/>

the SVM model prediction scores. Finally, the CD diagrams for log loss and ECE are plotted, as shown in Figure 26. Also, the average log loss and ECE value are reported in Table 9

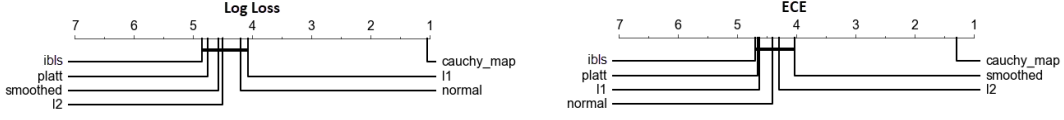


Figure 26. CD diagrams for log loss (left) and ECE (right) - (higher rank is better). Univariate LR models fitted to calibrate linear SVM model scores on real-datasets.

Even though there is no significant difference between the average ranks of different regularization methods, *ibls* label smoothing still has the advantage over the other methods with a slightly better average rank in terms of both log loss and ECE. Interestingly, *platt* scaling outperforms the smoothed LR. One reason to interpret this result is that a better smoothing factor does not exist in the defined search space of smoothing factors, or the way we choose this factor is not effective, especially with a small number of instances. However, this is the best that could be done in a reasonable computational complexity. Besides, although *platt* and *ibls* are non-parametric methods, they seem to have better calibration abilities than other methods, which usually require extensive search looking for good parameter configurations.

The results agree with what is reported in Table 9. The average log loss values for all the evaluated methods are too close to each other except for *cauchy\_map*. All the methods have a relatively large standard deviation compared to both loss and calibration errors. This agrees to what is reported in the CD diagram (Figure 26), where there is no significant difference among the average rank of the methods. However, *ibls* and *platt* have the lowest loss value with slightly smaller standard deviation for *ibls* but *smoothed* had the lowest calibration error.

Method	cauchy_map	ibls	L1	L2	normal	platt	smoothed
Log loss	0.5055 $\pm$ 0.3325	<b>0.1323 <math>\pm</math> 0.0723</b>	0.1324 $\pm$ 0.0727	0.1330 $\pm$ 0.0722	0.1480 $\pm$ 0.1035	<b>0.1323 <math>\pm</math> 0.0726</b>	0.1325 $\pm$ 0.0729
ECE	0.1337 $\pm$ 0.0837	0.0174 $\pm$ 0.0154	0.0169 $\pm$ 0.0166	0.0175 $\pm$ 0.0172	0.0180 $\pm$ 0.0165	0.0180 $\pm$ 0.0164	<b>0.0164 <math>\pm</math> 0.0164</b>

Table 9. Mean  $\pm$  standard deviation log loss and ECE for SVM model calibration on 40 real datasets.

The second experiment was carried out by fitting multivariate LR models using the different evaluated methods over the full list of features of the datasets. The CD diagrams for log loss and ECE are plotted in Figure 27 and the numerical values are reported in Table (10).

It can be noticed that there is no significant difference between the average ranks of the evaluated methods. *cauchy\_map* has achieved a large leap for multivariate datasets in contrast with *platt*, which had the worst performance. It is worth mentioning that during *cauchy\_map* fitting, the datasets were standardized to have a mean of 0, and a standard deviation of 0.5 as suggested by *Gelman et al.* [GJP<sup>+</sup>08] to make sure that fitted LR coefficients lie within the selected prior of Cauchy distribution. Even though *cauchy\_map* performs worst in most of the univariate experiments, it still performs much better with multivariate datasets. On the other hand,

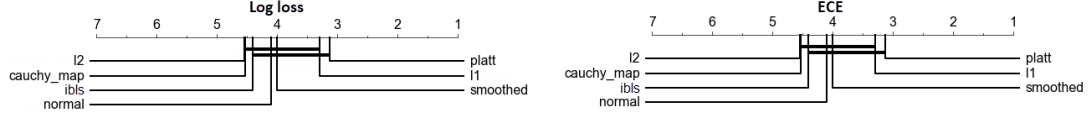


Figure 27. CD diagrams for log loss (left) and ECE (right) - (higher rank is better) for multivariate LR models on real datasets.

a slight impairment in *ibls* performance is noticed. However, as *ibls* is a non-parametric method, it still has an advantage over *L1*, *L2*, and *smoothed* which have some parameters to tune. While investigating why there exists a gap between *L1* and *L2*, it was found out that many coefficients in LR models fitted with *L1* were equal to zero which cannot happen in case of *L2* regularization. This effect could be avoided by setting the regularization penalty parameter of *L1* to include only small values. Table 10 shows the numerical results of this experiment. *L2* has the best average log loss and calibration error among all methods. There are no large differences among the reported values except for *normal*, which has a relatively large average log loss compared to the other methods. However, at the same time, it has a big standard deviation too.

Method	cauchy_map	ibls	L1	L2	normal	platt	smoothed
<b>Log loss</b>	$0.6470 \pm 0.3746$	$0.6738 \pm 0.3889$	$0.6658 \pm 0.3368$	<b><math>0.6405 \pm 0.3510</math></b>	$0.7419 \pm 0.5961$	$0.6665 \pm 0.3895$	$0.6433 \pm 0.3609$
<b>ECE</b>	$0.1000 \pm 0.0961$	$0.0895 \pm 0.0653$	$0.0990 \pm 0.0652$	<b><math>0.0791 \pm 0.0594</math></b>	$0.1002 \pm 0.1001$	$0.1050 \pm 0.0968$	$0.0902 \pm 0.0667$

Table 10. Mean  $\pm$  standard deviation log loss and ECE for multivariate logistic regression fitting on 40 real-datasets.

Overall, in real-datasets experiments, *ibls* got the best average rank in terms of both the log loss and calibration error when calibrating the scores of linear SVM models. There was no significant difference in the average ranking with many other methods like *platt*, *L1*, and *normal*. On the other hand, (*L2*) regularization and the Bayesian logistic regression with Cauchy prior for the model coefficients (*cauchy\_map*) were the best methods when fitting multivariate logistic regression models directly on the real-datasets in terms of both log loss and calibration error but also with no significant difference in the average ranking. These results show that methods based on label smoothing (*platt*, *ibls*, *smoothed*) do not perform on multi-dimensional datasets as good as on univariate datasets (logistic calibration). However, as *ibls* is a non-parametric approach with a good average rank among all methods, it shows good potential in reducing LR overfitting with both univariate and multivariate datasets.

### 7.3 Experiments for label smoothing in neural networks

In this set of experiments, we aim to evaluate our proposed instance-based label smoothing method for neural networks compared with cross-entropy with/without standard label smoothing, and calibrated using temperature scaling. The evaluation process involves how the network performs in terms of accuracy, log loss, and ECE with ten bins. Additionally, we used the trained networks as teacher networks utilized in network distillation to train a 5-layer shallow

convolutional student network [HVD15]. The accuracy and cross-entropy loss for the student network were evaluated and reported.

Dataset	Classes	Channels	Number of Instances Train set / Test set	Image Dimensions	Trained Network Architecture
CIFAR-10	10	3	50,000 / 10,000	32x32	ResNet18 [HZRS16] / InceptionV4 [SIVA17]
Fashion-MNIST	10	1	60,000 / 10,000	28x28	DenseNet [HLVDMW17]
CIFAR-100	100	3	50,000 / 10,000	32x32	ResNet50 [HZRS16]

Table 11. Properties of the datasets used in neural networks experiments.

### 7.3.1 Setup

The experiments began by downloading three multi-class benchmark datasets for image classification, which are CIFAR-10, CIFAR-100, and Fashion-MNIST. Table 11 summarizes properties of these datasets. All datasets are already divided into training and testing splits. However, the training sets were further divided into 80-20% training, validation splits. We train various state-of-the-art convolutional neural network architectures like *DenseNet*, *ResNet* and *Inception* networks over these datasets. Following the work of Pereyra *et al.* [PTC<sup>+</sup>17], the training process lasts for 100 epochs. An early stopping approach was followed to avoid the overfitting of the models, such that the training process is stopped if the validation set cross-entropy loss has not improved for ten consecutive epochs. All networks have been trained using a stochastic gradient descent optimizer with a momentum of 0.9. The learning rate was initialized at 0.1, with a weight decay of  $1e^{-4}$ . After that, the epoch with the best validation log loss is used to report the performance on the testing set. Each experiment was repeated three times, and average performance is finally reported.

The smoothing factor  $\epsilon$  parameter of the standard label smoothing method was tuned to be one of the following values  $\{0.01, 0.05, 0.1\}$ . Similarly, in instance-based label smoothing, the maximum smoothing factor  $\epsilon$  and the scaling factor  $u_{scale}$  were selected from  $\{0.01, 0.05, 0.1\}$  and  $\{1, 10, 100, 1000\}$  respectively. The final networks used in the comparisons are the ones that achieved the best validation log loss.

### 7.3.2 Results

The performance results for all the evaluated methods are summarized in table 12. In general, instance-based label smoothing achieved better performance than all other methods in terms of log loss of both teacher and distilled student networks in all datasets. Regarding the classification accuracy, instance-based label smoothing achieved the highest accuracy in CIFAR-10 and Fashion-MNIST datasets. However, as the number of classes increased significantly in CIFAR-100, label smoothing has slightly better accuracy. Besides, a cross-entropy loss without label smoothing and calibrated with temperature scaling has the lowest calibration error in CIFAR-10, but instance-based label smoothing was better in CIFAR-100 and Fashion-MNIST. Our proposed method consistently outperforms constant label smoothing in terms of all evaluation metrics except for Fashion-MNIST, where a tie exists in ECE. Temperature scaling has a noticeable improvement in terms of all evaluation metrics for the instance-based label smoothing in CIFAR-100 only, but no improvement was observed with other datasets.

Dataset	Network	Training Type	Temperature Scaling	Test Accuracy	Test LogLoss	Test ECE	Distilled Student Accuracy	Distilled Student LogLoss
CIFAR10	ResNet18	Without label smoothing	No	88.81 $\pm$ 0.75	0.362 $\pm$ 0.0165	0.042 $\pm$ 0.0027	84.11 $\pm$ 0.02	0.972 $\pm$ 0.0240
			Yes	88.81 $\pm$ 0.75	0.327 $\pm$ 0.0179	<b>0.006 <math>\pm</math> 0.0030</b>	84.04 $\pm$ 0.02	0.972 $\pm$ 0.0422
		With label smoothing	No	89.82 $\pm$ 1.03	0.317 $\pm$ 0.0271	0.025 $\pm$ 0.0021	84.76 $\pm$ 0.02	0.951 $\pm$ 0.0168
			Yes	89.82 $\pm$ 1.03	0.316 $\pm$ 0.0289	0.023 $\pm$ 0.0045	84.72 $\pm$ 0.01	0.958 $\pm$ 0.0253
		Instance-based label smoothing	No	<b>90.65 <math>\pm</math> 0.36</b>	<b>0.303 <math>\pm</math> 0.0101</b>	0.014 $\pm$ 0.0018	<b>85.61 <math>\pm</math> 0.01</b>	<b>0.942 <math>\pm</math> 0.0217</b>
			Yes	<b>90.65 <math>\pm</math> 0.36</b>	0.316 $\pm$ 0.0097	0.033 $\pm$ 0.0004	85.06 $\pm$ 0.01	0.952 $\pm$ 0.0209
		Without label smoothing	No	84.01 $\pm$ 1.22	0.501 $\pm$ 0.0273	0.050 $\pm$ 0.0071	82.19 $\pm$ 0.0134	0.999 $\pm$ 0.027
			Yes	84.01 $\pm$ 1.22	0.466 $\pm$ 0.0333	<b>0.012 <math>\pm</math> 0.0058</b>	<b>85.81 <math>\pm</math> 0.0107</b>	0.929 $\pm$ 0.0161
CIFAR10	InceptionV4	Without label smoothing	No	85.58 $\pm$ 0.36	0.438 $\pm$ 0.0098	0.0282 $\pm$ 0.0073	83.92 $\pm$ 0.0017	0.964 $\pm$ 0.0014
			Yes	85.58 $\pm$ 0.36	0.449 $\pm$ 0.0149	0.044 $\pm$ 0.0088	84.61 $\pm$ 0.0155	0.953 $\pm$ 0.0216
		With label smoothing	No	<b>85.90 <math>\pm</math> 0.95</b>	<b>0.431 <math>\pm</math> 0.0195</b>	0.027 $\pm$ 0.0037	85.25 $\pm$ 0.0043	<b>0.921 <math>\pm</math> 0.0067</b>
			Yes	<b>85.90 <math>\pm</math> 0.95</b>	0.448 $\pm$ 0.0333	0.042 $\pm$ 0.0019	85.56 $\pm$ 0.0168	0.944 $\pm$ 0.0243
		Instance-based label smoothing	No	62.12 $\pm$ 2.64	1.664 $\pm$ 0.0153	0.159 $\pm$ 0.0209	60.58 $\pm$ 0.43	1.486 $\pm$ 0.0064
			Yes	62.12 $\pm$ 2.64	1.397 $\pm$ 0.0766	0.033 $\pm$ 0.0202	60.64 $\pm$ 0.15	<b>1.478 <math>\pm</math> 0.0082</b>
		Without label smoothing	No	<b>68.16 <math>\pm</math> 1.93</b>	1.325 $\pm$ 0.0892	0.041 $\pm$ 0.0045	60.42 $\pm$ 0.60	1.483 $\pm$ 0.0148
			Yes	<b>68.16 <math>\pm</math> 1.93</b>	1.325 $\pm$ 0.0836	0.083 $\pm$ 0.0129	60.38 $\pm$ 0.34	1.487 $\pm$ 0.0109
CIFAR100	ResNet50	Without label smoothing	No	67.36 $\pm$ 1.52	1.350 $\pm$ 0.0570	0.049 $\pm$ 0.0098	60.52 $\pm$ 0.45	1.480 $\pm$ 0.0048
			Yes	67.36 $\pm$ 1.52	<b>1.201 <math>\pm</math> 0.0582</b>	<b>0.029 <math>\pm</math> 0.0006</b>	<b>60.84 <math>\pm</math> 0.05</b>	<b>1.478 <math>\pm</math> 0.0033</b>
		With label smoothing	No	92.68 $\pm$ 0.88	0.220 $\pm$ 0.0056	0.026 $\pm$ 0.0040	93.5 $\pm$ 0.0030	0.725 $\pm$ 0.0033
			Yes	92.68 $\pm$ 0.88	0.247 $\pm$ 0.0804	0.025 $\pm$ 0.0246	93.5 $\pm$ 0.0021	0.727 $\pm$ 0.0115
		Without label smoothing	No	92.52 $\pm$ 0.19	0.223 $\pm$ 0.0035	<b>0.010 <math>\pm</math> 0.0037</b>	93.5 $\pm$ 0.0041	0.722 $\pm$ 0.0096
			Yes	92.52 $\pm$ 0.19	0.326 $\pm$ 0.1558	0.040 $\pm$ 0.0157	93.3 $\pm$ 0.0101	0.719 $\pm$ 0.0204
		With label smoothing	No	<b>93.96 <math>\pm</math> 0.16</b>	<b>0.203 <math>\pm</math> 0.0010</b>	<b>0.010 <math>\pm</math> 0.0009</b>	93.5 $\pm$ 0.0008	<b>0.703 <math>\pm</math> 0.0120</b>
			Yes	<b>93.96 <math>\pm</math> 0.16</b>	0.215 $\pm$ 0.0167	0.029 $\pm$ 0.0047	<b>93.6 <math>\pm</math> 0.0021</b>	0.714 $\pm$ 0.0034
FashionMNIST	DenseNet	Without label smoothing	No	92.68 $\pm$ 0.88	0.220 $\pm$ 0.0056	0.026 $\pm$ 0.0040	93.5 $\pm$ 0.0030	0.725 $\pm$ 0.0033
			Yes	92.68 $\pm$ 0.88	0.247 $\pm$ 0.0804	0.025 $\pm$ 0.0246	93.5 $\pm$ 0.0021	0.727 $\pm$ 0.0115
		With label smoothing	No	92.52 $\pm$ 0.19	0.223 $\pm$ 0.0035	<b>0.010 <math>\pm</math> 0.0037</b>	93.5 $\pm$ 0.0041	0.722 $\pm$ 0.0096
			Yes	92.52 $\pm$ 0.19	0.326 $\pm$ 0.1558	0.040 $\pm$ 0.0157	93.3 $\pm$ 0.0101	0.719 $\pm$ 0.0204
		Without label smoothing	No	<b>93.96 <math>\pm</math> 0.16</b>	<b>0.203 <math>\pm</math> 0.0010</b>	<b>0.010 <math>\pm</math> 0.0009</b>	93.5 $\pm$ 0.0008	<b>0.703 <math>\pm</math> 0.0120</b>
			Yes	<b>93.96 <math>\pm</math> 0.16</b>	0.215 $\pm$ 0.0167	0.029 $\pm$ 0.0047	<b>93.6 <math>\pm</math> 0.0021</b>	0.714 $\pm$ 0.0034
		With label smoothing	No	92.68 $\pm$ 0.88	0.220 $\pm$ 0.0056	0.026 $\pm$ 0.0040	93.5 $\pm$ 0.0030	0.725 $\pm$ 0.0033
			Yes	92.68 $\pm$ 0.88	0.247 $\pm$ 0.0804	0.025 $\pm$ 0.0246	93.5 $\pm$ 0.0021	0.727 $\pm$ 0.0115

Table 12. Each experiment was repeated thrice and Mean  $\pm$  standard deviation of the evaluation metrics for instance-based label smoothing compared with constant label smoothing, and vanilla cross-entropy were reported. All evaluation results were reported both before and after temperature scaling.

In general, the instance-based label smoothing outperforms all other methods in terms of log loss, and distilled student network loss in all experiments. Besides, it has the lowest calibration error in two out of the four datasets and the highest accuracy in three datasets. Our results suggest that there is a significant improvement in the performance of the models trained with the instance-based label smoothing approach, which reduced overconfident predictions by smoothing the labels while keeping the class similarity structure of the dataset.

To understand how our proposed method differs from the constant factor label smoothing, we visualized the output logits of all the testing set instances using t-Distributed Stochastic Neighbor Embedding (t-SNE). t-SNE is a dimensionality reduction method that is well suited for the visualization of high-dimensional datasets [MH08]. Figure 28 shows the visualization of only two reduced components of the output logits from each network. It can be easily noticed how an overlap exists among a subset of the classes when training without label smoothing. On the other hand, standard label smoothing with constant smoothing factors among all the instances tends to cluster the instances of each class tightly together, which may erase some information about the class similarity structure in the dataset. However, our proposed instance-based label smoothing

has the advantage of separating the clusters of the classes a little bit from each other in addition to keeping a similar structure to the one learned without label smoothing.

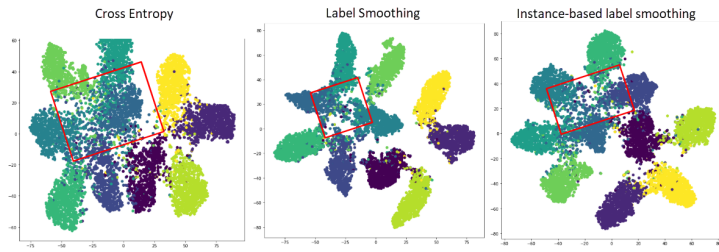


Figure 28. Visualization for first T-SNE two components of output logits of all classes on the testing set for CIFAR-10 with ResNet18.

Similarly, in Figure 29, only three different classes (dogs, cats, ships) were visualized to emphasize the effect of label smoothing and make it clearer. It can be easily observed that the distance between dog and cat classes is much smaller than that between dog/ship or cat/ship, which is more reasonable. Label smoothing tends to push all classes into tight, widely separated clusters. The distances between these clusters become larger with standard label smoothing. On the other hand, the instance-based label smoothing separated between the overlapped dog and cat clusters to some extent, which shows the better discrimination ability learned by the network. Simultaneously, the T-SNE visualization of the classes' logits shows a parallel class similarity structure to what the network learned without label smoothing. These visualizations support the benefits of our instance-based label smoothing approach that enhanced the discrimination and calibration performance of the network. Besides, it does not erase information about the class structure of the dataset like standard label smoothing. Thus, it exhibits better transfer learning and distillation performance.

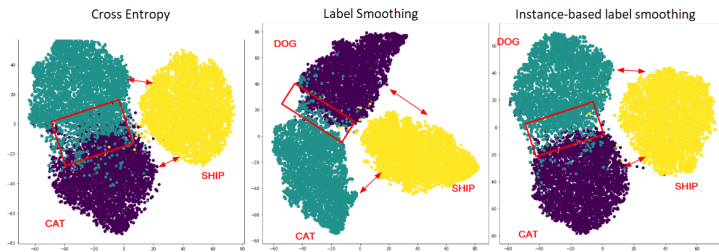


Figure 29. Visualization for first T-SNE two components of output logits of dog, cat, and ship classes on the testing set for CIFAR-10 with ResNet18.

## 8 Conclusion

Calibration of classifiers is a challenging and pervasive problem in the real world. One of the most important binary classifier calibration methods is logistic calibration, where a logistic regression model is fitted to convert scores of a binary classifier into calibrated probabilities. Several regularization methods can be applied to logistic regression to avoid producing over-confident probability predictions like Platt scaling [P<sup>+</sup>99].

In this thesis, using a Bayesian approach, we derived the optimal probability predictions that minimize a target loss function in case the exact uncertainty of the dataset distribution parameters is known. Second, a new instance-based label smoothing approach is proposed using kernel density estimation for logistic regression fitting. Finally, we extend our method to develop a parametric instance-based label smoothing approach in neural networks. This approach can overcome the problem of information erasure about the dataset class similarity structure caused by the standard label smoothing, which impairs the network performance in transfer learning, and distillation [MKH19].

The experiments showed that our proposed label smoothing approach outperforms Platt-scaling, and other regularization methods mainly when separation problems occur in training sets, and at small training sets sizes. Overall, our proposed instance-based label smoothing outperforms other methods in terms of log loss and calibration error in most of the experimented synthetic, and 40 real-world datasets. The experiments also confirmed that the sampling-based estimates provide good approximations to the derived formula of the optimal predicted probabilities on synthetic datasets created from different generative models with known distributions. Additionally, In multi-class classification neural networks, our proposed instance-based label smoothing achieved lower calibration error and loss values over three image classification benchmark datasets. Simultaneously, more information about class similarity is kept than constant label smoothing, which is useful for better distillation and transfer learning.

The possible future work could involve using our derived near-optimal probabilities to be analytically compared with other parametric regularization methods like label smoothing,  $L1$ , and  $L2$  regularization. This could help in finding the set of best regularization parameters for each respective method like the smoothing factor in label smoothing or regularization penalty in  $L1/L2$ ). Later, these findings could drive some insights into better regularization methods. Besides, our proposed parametric instance-based label smoothing approach could be evaluated extensively on different types of neural network architectures, and datasets other than image classification tasks. These experiments could help in drawing more insights about the generalization ability of the proposed method, and what range of parameters were frequently producing the best network performance.

## References

- [AA84] Adelin Albert and John A Anderson. On the existence of maximum likelihood estimates in logistic regression models. *Biometrika*, 71(1):1–10, 1984.
- [ADLB19] Rusul Abduljabbar, Hussein Dia, Sohani Liyanage, and Saeed Asadi Bagloee. Applications of artificial intelligence in transport: An overview. *Sustainability*, 11(1):189, 2019.
- [AK] Mari-Liis Allikivi and Meelis Kull. Non-parametric bayesian isotonic calibration: Fighting overconfidence in binary classification. *Machine Learning and Knowledge Discovery in Databases (ECML-PKDD’19)*, pages 68–85.
- [AW18] Arkar Min Aung and Jacob Whitehill. Harnessing label uncertainty to improve modeling: An application to student engagement recognition. In *FG*, pages 166–170, 2018.
- [Bab04] Michael A Babyak. What you see may not be what you get: a brief, nontechnical introduction to overfitting in regression-type models. *Psychosomatic medicine*, 66(3):411–421, 2004.
- [Bar12] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.
- [BB04] M Jésus Bayarri and James O Berger. The interplay of bayesian and frequentist analysis. *Statistical Science*, pages 58–80, 2004.
- [CG04] Ira Cohen and Moises Goldszmidt. Properties and benefits of calibrated classifiers. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 125–136. Springer, 2004.
- [Den15] Li Deng. Achievements and challenges of deep learning. *Dosegljivo: http://research.microsoft.com/pubs/246516/APSIPA\_May2015.pdf [Dostopano: 31. 3. 2016]*, 2015.
- [DLP11] Richard A Davis, Keh-Shin Lii, and Dimitris N Politis. Remarks on some nonparametric estimates of a density function. In *Selected Works of Murray Rosenblatt*, pages 95–100. Springer, 2011.
- [Dri01] Joseph Drish. Obtaining calibrated probability estimates from support vector machines. *Final project for CSE*, 254, 2001.
- [Fla12] Peter Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.
- [GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.



- [GJP<sup>+</sup>08] Andrew Gelman, Aleks Jakulin, Maria Grazia Pittau, Yu-Sung Su, et al. A weakly informative default prior distribution for logistic and other regression models. *The annals of applied statistics*, 2(4):1360–1383, 2008.
- [GLM07] Alexander Genkin, David D Lewis, and David Madigan. Large-scale bayesian logistic regression for text categorization. *technometrics*, 49(3):291–304, 2007.
- [GPSW17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.
- [Haw04] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.
- [HBJ<sup>+</sup>14] Timothy E Hanson, Adam J Branscum, Wesley O Johnson, et al. Informative  $g$ -priors for logistic regression. *Bayesian Analysis*, 9(3):597–612, 2014.
- [HJ15] Frank E Harrell Jr. *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2015.
- [HLVDMW17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.
- [HOFR11] José Hernández-Orallo, Peter A Flach, and Cèsar Ferri Ramirez. Brier curves: a new cost-based visualisation of classifier performance. In *ICML*, pages 585–592, 2011.
- [Hol79] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.
- [HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.
- [HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- [HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [JDPS19] Kirtan Jha, Aalap Doshi, Poojan Patel, and Manan Shah. A comprehensive review on automation in agriculture using artificial intelligence. *Artificial Intelligence in Agriculture*, 2019.
- [KDG<sup>+</sup>02] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

- [KL51] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.
- [KL15] Volodymyr Kuleshov and Percy S Liang. Calibrated structured prediction. In *Advances in Neural Information Processing Systems*, pages 3474–3482, 2015.
- [KSFF17] Meelis Kull, Telmo Silva Filho, and Peter Flach. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Artificial Intelligence and Statistics*, pages 623–631, 2017.
- [KSL19] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2661–2671, 2019.
- [L<sup>+</sup>89] Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, 19:143–155, 1989.
- [LC86] Lucien Le Cam. The central limit theorem around 1935. *Statistical science*, pages 78–91, 1986.
- [LLAN06] Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y Ng. Efficient  $\ell^1$  regularized logistic regression. In *AAAI*, volume 6, pages 401–408, 2006.
- [MH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
- [Mil39] Friedman Milton. A correction: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association. American Statistical Association*, 34(205):109, 1939.
- [MKH19] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pages 4696–4705, 2019.
- [NCH15] Mahdi Pakdaman Naeni, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- [Ng04] Andrew Y Ng. Feature selection,  $\ell^1$  vs.  $\ell^2$  regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78, 2004.
- [NMC05] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005.
- [P<sup>+</sup>99] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

- [Par62] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.
- [PSA18] Trishan Panch, Peter Szolovits, and Rifat Atun. Artificial intelligence, machine learning and health systems. *Journal of global health*, 8(2), 2018.
- [PTC<sup>+</sup>17] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.
- [Pä19] Liina Anette Pärtel. Label smoothing in logistic calibration of classifiers, 2019.
- [Sil86] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.
- [SIVA17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.
- [SVI<sup>+</sup>16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.
- [VvRBT13] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.
- [Wil92] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.
- [WVCB<sup>+</sup>20] Laure Wynants, Ben Van Calster, Marc MJ Bonten, Gary S Collins, Thomas PA Debray, Maarten De Vos, Maria C Haller, Georg Heinze, Karel GM Moons, Richard D Riley, et al. Prediction models for diagnosis and prognosis of covid-19 infection: systematic review and critical appraisal. *bmj*, 369, 2020.
- [ZE01] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Icml*, volume 1, pages 609–616. Citeseer, 2001.
- [ZE02] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699, 2002.
- [ZZ19] Guoping Zeng and Emily Zeng. On the relationship between multicollinearity and separation in logistic regression. *Communications in Statistics-Simulation and Computation*, pages 1–9, 2019.

# Appendix

## I. Source Code

Source code for implementation of different methods proposed in this thesis is located in the following GitHub repository:

<https://github.com/mmaher22/Instance-based-smoothing>

## II. Licence

### Non-exclusive licence to reproduce thesis and make thesis public

I, **Mohamed Maher Abdelrahman**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,  
**Instance-based Label Smoothing for Better Classifier Calibration**,  
supervised by Meelis Kull.
2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Mohamed Maher Abdelrahman

**15/05/2020**