Noname manuscript No.
(will be inserted by the editor)

On classifier behavior in the presence of mislabeling noise

Katsiaryna Mirylenka ·
George Giannakopoulos ·
Le Minh Do ·
Themis Palpanas

Received: date / Accepted: date

Abstract Machine learning algorithms perform differently in settings with varying levels of training set mislabeling noise. Therefore, the choice of the right algorithm for a particular learning problem is crucial. The contribution of this paper is towards two, dual problems: first, comparing algorithm behavior; and second, choosing learning algorithms for noisy settings.

We present the "Sigmoid Rule" Framework (SRF), which can be used to choose the most appropriate learning algorithm depending on the properties of noise in a classification problem. The framework uses an existing model of the expected performance of learning algorithms as a sigmoid function of the signal-to-noise ratio in the training instances. We study the characteristics of the sigmoid function using five representative non-sequential classifiers, namely, Naïve Bayes, kNN, SVM, a decision tree classifier, and a rule-based classifier, and three widely used sequential classifiers based on Hidden Markov models, Conditional Random Fields and Recursive Neural Networks. Based on the sigmoid parameters we define a set of intuitive criteria that are useful for comparing the behavior of learning algorithms in the presence of noise. Furthermore, we show that there is a connection between these parameters and

K. Mirylenka
IBM Research–Zurich: Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
E-mail: kmi@zurich.ibm.com,
work mainly done while at the University of Trento, Italy

G. Giannakopoulos
Institute of Informatics and Telecommunications of NCSR Demokritos, Greece
15310, Ag. Paraskevi, Athens, Greece
E-mail: ggianna@iit.demokritos.gr

L. Do
University of Trento, via Sommarive 5, Trento, Italy, 39123
E-mail: leminh.do@studenti.unitn.it

T. Palpanas
Department of Computer Science, Paris Descartes University, France
E-mail: themis@mi.parisdescartes.fr

the characteristics of the underlying dataset, showing that we can estimate an expected performance over a dataset regardless of the underlying algorithm. The framework is applicable to concept drift scenarios, including modeling user behavior over time, and mining of noisy time series of evolving nature.

Keywords Classification · Sequential classifiers · Classifier evaluation · Handling noise · Concept drift

1 Introduction

Transforming vast amounts of collected – possibly noisy – data into useful information through the processes of clustering and classification has important applications in many domains. One example is preference detection of customers for marketing. Another example is detection of negligent "providers" for Mechanical turk and reduction of their behavior on task results. The machine learning and data mining communities have extensively studied the behavior of classifiers in different settings (e.g., [23, 52]), however, the effect of noise on the classification task is still an important and open problem.

The importance of studying noisy data settings is augmented by the fact that noise is very common in a variety of large scale data sources, such as sensor networks and the Web, while Machine learning algorithms (both classic and sequential) perform differently in settings with varying levels of labeling noise. Thus, there rises a need for a unified framework aimed at studying the behavior of learning algorithms in the presence of noise, depending on the specifics of each particular dataset.

Labeling noise is common in cases of *concept drift*, where a target concept shifts over time, rendering previous training instances obsolete. Essentially, in the case of concept drift, feature noise causes the labels of previous training instances to be out of date and, thus, equivalent to label noise. Drifting concepts appear in a variety of settings in the real world, such as the state of a free market, or the traits of the most viewed movie. Thus, we expect this work to offer intuition and tools for meta-learning in similar, noise-prone, real cases of learning.

Earlier work has shown that the performance[1] of a classifier in the presence of noise can be effectively approximated by a *sigmoid* function, which relates the signal-to-noise ratio in training data to the expected performance of the classifier [19]. We call this fact the "Sigmoid Rule". The sigmoid rule states that the expected performance of each learning algorithm can be seen as a sigmoid function. This function indicates the impact of signal-to-noise ratio in the training set to the performance of the algorithm, under an assumption of monotonicity of the change of performance (i.e., when cleaner data are provided the algorithm is expected to perform better). The added value of this view is that it allows a single model, with a limited number of parameters, to be used as an illustration of the behavior of all learning algorithms in a

[1] When we write *performance* of an algorithm, we mean the classification accuracy.

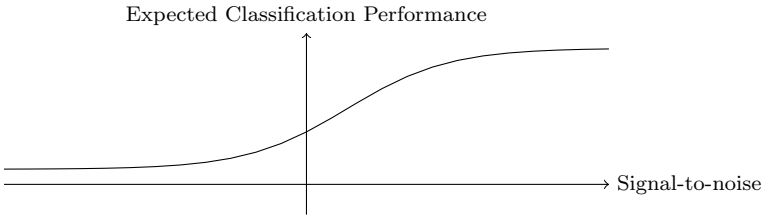


Fig. 1: An example of a sigmoid function relating signal-to-noise (ratio of mislabeled instances) in the *training* set (horizontal axis) to expected classification performance (vertical axis).

noisy setting. We illustrate an example of a sigmoid function in Figure 1 and elaborate on the concept in Section 3.1.

In this work, we study the effect of training set *label noise*[1] on a classification task. We further build on this study to achieve two, dual goals:

- We show how to support the comparison between algorithms and their behavior in a noisy setting. The workflow in this case is as follows: we select an algorithm; we sample the input for training data; we estimate a set of parameters (based on the proposed sigmoid rule framework we discuss below) that characterize the behavior of the algorithm in the presence of noise; we compare algorithms based on those parameters. Thus, in Section 4, we examine how much added benefit we can derive from the sigmoid rule model, by elaborating on the parameters of the sigmoid and illustrating the connection of each such parameter to the learner behavior. Based on the most prominent parameters we define several dimensions characterizing the behavior of learning algorithms. We, finally, illustrate how these dimensions can be used to select learning algorithms in different settings, based on user requirements.
- We show how to allow for the prediction of algorithm performance, given dataset attributes related to a specific classification problem. The workflow in this case is as follows: we calculate a set of dataset attributes (i.e., the number of classes, features and instances and the fractal dimensionality [48]); we use estimation (regression) models to predict the expected performance regardless of the choice of the underlying algorithm. We study the correlation between dataset characteristics and modeled performance in Section 7.

Moreover, we extend our study to sequential classification cases, to illustrate the broad applicability and usefulness of the "Sigmoid Rule" Framework (SRF).

The SRF uses a model of the expected performance of learning algorithms based on a sigmoid function of the signal-to-noise ratio in the training in-

[1]Throughout the rest of this work we will use the term *noise* to refer to the label noise, unless otherwise indicated.

stances. SRF is specialized for noisy settings, providing more insights regarding the behavior of the algorithms. Essentially, it describes a meta-model (the SRF sigmoid) on various algorithms, thus making it a special case of meta-modeling.

Contributions. In summary, we make the following contributions.

- We define a set of intuitive criteria based on the SRF that can be used to compare the behavior of learning algorithms in the presence of noise. This set of criteria provides both quantitative and qualitative support for learner selection in different settings.
- We demonstrate that there is a connection between the SRF dimensions and the characteristics of the underlying dataset, using both a correlation study and regression modeling based on linear and logistic regression. In both cases we discovered statistically significant relations between SRF dimensions and dataset characteristics. Our analysis shows that these relations are stronger for the sequential classifiers than for the non-sequential, traditional classifiers, where the instances are considered to be independent.
- Our results are based on an extensive experimental evaluation, using 10 synthetic and 31 real datasets from diverse domains. The heterogeneity of the dataset collection validates the general applicability of the SRF for both non-sequential and sequential learners.

The rest of this paper[1] is organized as follows. We review the related work (Section 2) and define the problem studied (Section 3). We then focus on the sigmoid function and define the SRF dimensions (Section 4), describing their qualitative value. We test our framework on multiple datasets (Section 6), statistically analyzing the connection between dataset properties and SRF dimensions (Section 7) for both sequential and non-sequential classification tasks. Section 8 provides a summary of the proposed approach and the experimental results. Finally, we conclude in Section 9.

2 Related Work

Since 1984, when Valiant published his theory of learnable [53], setting the basis of Probably Approximately Correct (PAC) learning, numerous different machine learning algorithms and classification problems have been devised and studied in the machine learning community. Given the variety of existing learning algorithms, researchers are often interested in obtaining the best algorithm for their particular tasks. The algorithm selection is considered to be a part of the meta-learning domain [21].

According to the No-Free-Lunch theorems (NFL) described in the work [58] and proven in the works [59,57], there is no overall best classification algorithm, given the fact that all algorithms have equal performance on average over all datasets. Nevertheless, we are usually interested in applying machine

[1] A preliminary version of this work has appeared in [39].

learning algorithms to a specific dataset with certain characteristics, in which case we are not limited by the NFL theorems. The results of NFL theorems hint at comparing different classification algorithms on the basis of dataset characteristics [3], which is the aim of our work in this research direction.

Concerning the measures of performance that help distinguish among learners, in [3] the authors compared algorithms on a large number of datasets, using measures of performance that take into account the distribution of the classes, which is a characteristic of a dataset. The Area Under the receiver operating Curve (AUC) is another measure used to assess machine learning algorithms and to divide them into groups of classifiers that have statistically significant difference in performance [6]. In all the above studies, the analysis of performance has been applied for the datasets without extra noise, while we study the behavior of classification algorithms in noisy settings. In our study, we use the fact shown by Giannakopoulos and Palpanas [19,20] about concept drift, which illustrates that a sigmoid function can accurately describe performance in the presence of varying levels of label noise in the training set. In an early influential work [30], the problem of *concept attainment* in the presence of noise was also indicated and studied in the STAGGER system. In this paper, we analytically study the sigmoid function to determine the set of parameters that can be used to support the selection of the learner in different noisy classification settings.

The behavior of machine learning classifiers in the presence of noise is also considered in the work [27]. The artificial datasets used for classification were created on the basis of predefined linear and nonlinear regression models, and noise was injected in the features, instead of the class labels as in our case. Noisy models of non-markovian processes using reinforcement learning algorithms and the methods of temporal difference are analyzed in the paper [44]. In the work [10], the authors examine multiple-instance induction of rules for different noise models. There are also theoretical studies on regression algorithms for noisy data [51] and works on denoising, like [34], where a wavelet-based noise removal technique was shown to increase the efficiency of four learners. Both noise-related studies [51,34] dealt with noise in the attributes. However, we study label-related noise and do not consider a specific noise model, which is a different problem. The label-related noise corresponds mostly to the concept drift scenario, as was also discussed in the Section 1. Moreover, we estimate performance as a function of the signal-to-noise ratio, contrary to the works described above.

In the work of Nettleton et al. [42], the authors study the effect of different types of noise on the precision of supervised learning techniques. They address this issue by systematically comparing how different degrees of noise affect supervised learners that belong to different paradigms. They consider 4 classifiers, namely the Naïve Bayes classifier, the decision tree classifier, the IBk instance-based learner and the SMO support vector machine. They test them on a collection of data sets that are perturbed with noise in the input attributes and noise in the output class, and observe that Naïve Bayes is more robust to noisy data than the other three techniques. The performance was

evaluated on 13 datasets. However, no general framework was proposed for modeling the effect of noise on the performance of the classifiers. In contrast, we propose the Sigmoid Rule Framework for comparing the behavior of classifiers. We also introduce different dimensions that can be used to formulate the criteria for comparing the algorithms depending on user needs. Moreover, we also apply this framework to sequential classifiers, which were not considered in above studies.

Previous studies on meta-learning and concept drift [29,56] describe approaches that adjust to changing user interests, taking into account the *context* [56]. Meta-learning can occur at various levels, e.g., training instance selection, learning algorithm and parameters selection, window size selection, based on online measurements. These measurements may include error rates, distribution of class instances in a window of recent instances, and other related data or classification traits. However, these works do not explicitly address label noise, and cannot provide an a priori estimation of classifier performance in such situations, as in our work.

Recent works on ensemble-based meta-learning (e.g., [13]) deal with local characteristics of the data and optimize Ensembles of Classifiers (EoCs), but do less to provide dataset-related and algorithm-independent results, as we do in this work (Section 7). In [25], the challenges of *evolutionary model building* are discussed, including a number of challenges and related approaches in the literature aimed at (evolving) streaming data in classification tasks. In this overview, the author disuses (through a multitude of works) a variety of required characteristics of learning methods, and how evolutionary computation can contribute towards these characteristics. We believe that our work is complementary, in that it can provide a basis for future meta-learning and evolutionary algorithms, where fine traits of learning (e.g., stability of performance vs. maximum performance) can prioritize selection in a learning setting.

The SRF view of meta-learning provides another important tool to the currently available approaches. This view may be even more effective taking into account references to recurring contexts [43], where the variance of noise levels may be predictable – after a period of observation – and, thus, the knowledge of an algorithm's behavior within these levels can constitute important knowledge. Furthermore, in this work we propose a novel direction of meta-learning, moving the focus from "accuracy to transparency and trust" [43]. We achieve this by explaining classification expected behavior under novel, interesting axes of analysis, which take into account not only the absolute values of the performance of a classifier, but also characterize the behavior of the performance change from various perspectives.

Other meta-learning literature has focused on the following topics: impact of the dataset and algorithm features on the learning performance [7], [8], [33], [46], [18], [41]; identification and handling of different types of noise [7], [46] and concept drift [46], [18] in the learning setting; meta-learning frameworks for fine-tuning of the learning systems [7], [8], [47], [1].

Concerning the study of dataset and algorithm features, the works of [7] and [8] describe a number of dataset features and discuss their usefulness on

the performance of individual classifiers. However, these works do not include a systematic analysis of all dataset-classifier pairs. Paper [33] introduces a dataset characteristic called *hardness* and demonstrates that it contributes significantly to the performance of any classifier, which supports our conclusion that the dataset characteristics can be used to assess classifier performance a priori. The paper [41] addresses the problem of choosing the parameters of a learning algorithm that would maximize its performance on a new dataset based on its characteristics. The proposed solution uses a meta-classifier with 81 features, which include the number of attributes and classes in the dataset, statistical moments, information theory measures, and others. The work does not choose the most efficient dataset features. Authors of works [46] and [18] apply meta-learning – to regression problems and to noise filters respectively. Similarly to [41], these works also use various dataset statistics as features, without a systematic analysis of feature importance.

Concerning noise, in our work we study the correlation between labeling noise (as opposed to feature noise) and the expected performance. Works [46] and [18] also consider noisy settings. Similarly to our work, paper [46] is focused on concept drift scenarios. However, it does not provide generally applicable criteria for selecting the learning algorithm and does not deal with explicit label noise. Paper [18] discusses the performance of noise filters, but does not provide a generic model of algorithm behavior in the presence of noise. In comparison to these works, we additionally address the performance of sequential models for classification in the presence of noise. We provide the added value of an estimated analytic connection between noise levels and performance (described by the sigmoid curve). The sigmoid rule allows an adaptive meta-learning system to have prior knowledge regarding algorithms and their expected performance in different levels of noise and under different requirements (stability, average performance, etc.). Meta-learning frameworks that can estimate noise levels in a dataset will also benefit from our analysis.

General meta-learning frameworks are described in detail in works [7] and [8]. Efficiency modifications for fast discovery of optimal algorithms are proposed in work [1]. In our work we focus on a systematic analysis of the performance of classifiers depending on the dataset and classifier features, and the level of noise. A future study can adopt the architecture of meta-learning framework for classifier selection in noisy settings. Work [47] proposes collaborative filtering approach for meta-learning, arguing that the relevant algorithm and dataset features can be chosen while applying collaborative filtering. It is not clear how this approach will perform in the case of concept drift scenario and in the case of multi-criteria performance objectives. In this work we analyze the performance of a classifier in the environment with varying label noise and study the effect of dataset characteristics on performance behavior. We leave the comparison of meta-learning platforms for future work.

Sequential data and especially time series data collected from sensors are often affected by noise. That is, the quality of the data may be decreased by errors due to limitations in the tolerance of the measurement equipment. Sequential classification based on Markov chains with noisy training data is

considered in the work of Dupont [14]. The paper shows that smoothed Markov chains are very robust to classification noise. The same paper discusses the relation between the classification accuracy and the test set perplexity that is often used to measure prediction quality. However, unlike our work, no specific formalization is proposed for analyzing the effect of different noise levels on the performance of the classifier. Development of such formalization is becoming increasingly important as sequential and time series data classification are applied in diverse domains, such as financial analysis, bioinformatics and information retrieval. For example, a recent survey [60] considers various methods of sequential classification in terms of methodologies and application domains. The methods for sequential classification we focus on are introduced in the works [45], [49] and [15], which describe Hidden Markov models, Conditional Random Fields, and Recurrent Neural Networks correspondingly. The techniques described in studies [40], [38], such as Conditional Heavy Hitters, allow estimating sequential models in real time. The efficiency of such techniques enables the applicability of sequential classification to sensor data analysis and moving trajectories.

3 Problem Statement and Preliminaries

We consider settings where a classification task is performed under varying levels of label noise, similarly to a concept drift scenario, where the labels are gradually changing and the noise level is not known in advance. Such settings are common and have been detailed in previous works (e.g., see definition of the "problem of the demanding lord" [20], where a user changes her preferences gradually over time and an automatic system is called to adapt to this change).

Our goal is to recommend a classifier that is beneficial for a given dataset, based on custom algorithm optimality criteria (e.g., stability under noise vs. maximum possible performance). The only required knowledge related to the task is on characteristics of the dataset, such as the number of classes, number of attributes, and intrinsic dimensionality.

More formally, we define a training set as $\mathbb{T}$. Part of the instances in this dataset have true labels. We define this set of instances $\mathbb{S}$ and term them "signal" instances. The rest of the instances have noisy (i.e., erroneous) labels. We define this set as $\mathbb{N}$ and term them "noisy" instances. Hence, $\mathbb{S} \subseteq \mathbb{T}$, $\mathbb{N} \subseteq \mathbb{T}$ and $\mathbb{S} \cup \mathbb{N} = \mathbb{T}$ while $\mathbb{S} \cap \mathbb{N} = \emptyset$. Then $S = |\mathbb{S}|$ and $N = |\mathbb{N}|$ represent the signal magnitude and the noise magnitude of a training set $\mathbb{T}$, respectively. We also allow $\mathbb{S} = \emptyset \Rightarrow S = 0$, $\mathbb{N} = \mathbb{T} \Rightarrow N = |\mathbb{T}|$, when all the training set is not valid any longer because a shift has just occurred. Correspondingly, $\mathbb{N} = \emptyset \Rightarrow N = 0$, $\mathbb{S} = \mathbb{T} \Rightarrow S = |\mathbb{T}|$, when no noise is induced in the training set.

Given the above definitions, we define the signal-to-noise ratio Z of a given moment in time:

$$Z = \log(1 + S) - \log(1 + N) = \log \frac{1 + S}{1 + N}, \tag{1}$$

Table 1: The definitions of the main concepts.

Concept	Definition
Signal-to-Noise Ratio	the ratio of correctly labeled training instances, to the incorrectly labeled training instances
Performance	a measure of the accuracy of an algorithm in a noisy training setting. In a classification problem, we use accuracy
Characteristic Transfer Function (CTF)	is a function connecting the signal-to-noise in the training dataset to the expected performance of a learning algorithm
Sigmoid Rule Framework (SRF)	is an approach that models the expected performance of a learning algorithm, using a sigmoid function as the CTF for any algorithm

where $\log(\cdot)$ represents the natural logarithm. We add one to signal and noise magnitudes so that the logarithm is defined for their zero values.

3.1 Characteristic Transfer Function

In order to describe the performance of a classifier we use the "sigmoid rule", which was introduced and discussed in previous studies [19,20]. The sigmoid rule uses a function that relates signal-to-noise ratio of the training set to the expected performance of a learner. This function is called the *characteristic transfer function* (CTF) of the learning algorithm. In this work, we also refer to it as the *sigmoid function* of the algorithm, and use the terms CTF and sigmoid function interchangeably.

Given that an algorithm has a minimum performance of m and a maximum performance of M for a given domain of signal-to-noise values, then the function that describes the average performance f as a function of the signal-to-noise ratio Z (Equation 1), is of the form [19,20]:

$$f(Z) = m + (M - m)\frac{1}{1 + b \cdot \exp(-c(Z - d))}, \tag{2}$$

where $m \leq M$, $m, M \in [0, 1]$, $b, c \in \mathbb{R}^+$, $d \in R$ are the parameters of the sigmoid function. The number of parameters can be reduced, though their presence allows for a more accurate estimation of the performance of a classifier, as has been empirically shown [20]. An example of a sigmoid function can be seen in Figure 2. The definitions of the main concepts used in the paper are given in Tabel 1.

The basis of the CTF lies on Probably Approximately Correct learning [24, 53] and, more specifically, on the expectation that – in PAC learnable settings – enough (randomly drawn) samples can help a system select a hypothesis identifying concepts. Thus, in PAC learning, more correctly labeled instances decrease the probability of error, reaching a limit ϵ. In the CTF modeling of learning we have tried to model what happens in the presence of mislabeled instances, while keeping the limit cases aligned to PAC learning.

Let us consider the case where a set of concepts $\mathbb{C}$ is (PAC) learnable. This means that, provided with enough learning instances, the system will end up learning (a hypothesis H identifying) the concepts. In this case, the selected, correctly labeled instances that were input to the learning system as training are all signal, with no noise. The error of this system tends to be below ϵ.

Now, let us consider the case where there exists a "false labeling" probability $1 \geq P_f \geq 0$, when the input instances are provided to the learner. These instances constitute a population of training instances that do not reflect the original, true instance distributions for $\mathbb{C}$. In this case, the system will (try to) learn a different set of concepts $\mathbb{C}'$, which is a function of the distributions of $\mathbb{C}$ and the "false concepts" set distributions $\mathbb{F} \neq \mathbb{C}$. It is easy to deduce that, if the system learns $\mathbb{C}'$, i.e., it forms a hypothesis $H' \neq H$ that can map instances to $\mathbb{C}'$ with an error probability below ϵ after enough instances, then a portion of these mappings/predictions will stand in $\mathbb{C}'$ but not in $\mathbb{C}$. The more mislabeled instances are provided as input to the learner, the higher the probability that the predictions will be wrong, with respect to $\mathbb{C}$. In the extreme case that all instances of $\mathbb{C}$ are provided as inputs and the learner learns all the space, the probability of error can be exactly P_f, since the learner will have mistakenly and perfectly learned all the falsely-labeled instances. If all the input instances are provided mislabeled, then the learner will have learned the full set of false concepts $\mathbb{C}' = \mathbb{F}$ and $\mathbb{C}$ will have no effect on the learning. In this case the performance of the system with respect to $\mathbb{C}$ will be minimum. Different values of P_f are expected to vary the error rates between the minimum and the maximum value. If we are interested in the average error rate (and dually the performance) of the learner over several experiments in the presence of noise, we can only assume that, after enough tries, increasing P_f will lead to higher error $\epsilon' = \epsilon + P_f > \epsilon, P_f > 0$. At this point we provide an alternative view of P_f: it is directly connected the percentage of noise $N = P_f \cdot |\mathbb{T}|$ and signal $S = (1 - P_f)|\mathbb{T}|$ in a signal-to-noise ratio Z from Equation 1, where $\mathbb{T}$ is a training set as is introduced above.

To summarize, the intuition behind the sigmoid in Equation 2 is based on the fact that a (non-trivial) learning algorithm starts to perform well after a certain ratio of good to bad examples has been observed. From that moment on, the performance of the algorithm constantly improves (on average) as the ratio is improved (monotonicity assumption), until the point where the best performance is reached. Then, no matter how much the ratio of good to bad examples increases, there is little change, because the algorithm cannot do much better, since it has learned the input concepts (and also possibly due to its generalization property, where such a property is applicable). We consider the CTF to be characteristic of an algorithm for a given dataset (i.e., set of concepts, instance and hypothesis space). It has been demonstrated that the sigmoid can be estimated from training sets of varying Z and, then, it can be used as a known function for a given algorithm [20]. In this work, we demonstrate that the characteristics of a dataset (e.g., Vapnik-Chervonenkis or VC dimension [54]) have a strong influence over the sigmoid parameters (refer to Section 6). This property of the sigmoid allows reusing sigmoids

across datasets with similar characteristics and suggesting the most promising learner for a new dataset, based on custom optimality criteria.

The behavior of different machine learning algorithms in the presence of noise can be compared along several *axes of comparison*, based on the parameters of the sigmoid function. The parameters related to *performance* include:

(a) the minimal performance m,
(b) the maximal performance M,
(c) the span of the performance range $r_{alg} = M - m$.

With regard to the *sensitivity* of performance to the change in the signal-to-noise ratio, we consider:

(a) within which range of noise levels, change in the noise leads to significant change in performance;
(b) how we can tell apart algorithms that improve their performance even when the signal-to-noise levels are low, from those that only improve in high ranges of the signal-to-noise ratio;
(c) how we can measure the stability of performance of an algorithm against varying noise;
(d) at what noise level an algorithm reaches its average performance;
(e) whether reducing the noise in a dataset is expected to have a significant impact on the performance.

To answer these questions we perform an analytic study of the sigmoid function of a particular algorithm by the means of traditional function analysis. This analysis helps to devise measurable dimensions that can answer our questions about the parameters related to performance of classifiers.

In the next section we introduce the Sigmoid Rule Framework (SRF) which proposed several axes for algorithm comparison, based on the CTF. The set of parameters of the CTF and the proposed SRF is summarized in Table 2 for quick reference.

4 The Sigmoid Rule Framework

We investigate the properties of the sigmoid function (Equation 2) in order to determine how each of the parameters m, M, b, c, and d affects the shape of the sigmoid, and how this translates to the expected performance of the learning algorithm. We start by a direct analysis of the sigmoid function and its parameters. In Figure 2 we see an illustration of a sigmoid, where we have indicated the points related to the curve parameters. The reader is referred to Table 2 for a summary of the symbols.

The domain of the sigmoid is, in the general case, $Z \in (-\infty, +\infty)$. The range of values is (m, M), as also can be seen on Figure 2. Then, we consider the derivatives of the sigmoid function in order to study its characteristics.

Since the first order derivative is always positive, $f'(Z) > 0$ for $\forall Z \in (-\infty, +\infty)$, the function $f(Z)$ is monotonically increasing, which corresponds

Table 2: Notation of sigmoid-related variables and SRF dimensions.

Parameter	Description
	Sigmoid-related variables
S	signal magnitude in training set
N	noise magnitude in training set
Z	signal-to-noise ratio
m, M, b, c, d	parameters of sigmoid
$f(Z)$	sigmoid function
Z_{inf}	point of inflection of sigmoid (zero of the 2nd derivative)
$Z^{(3)}_{1,2}$	zeroes of the 3d derivative of sigmoid
d_s	distance between Z_{inf} and $Z^{(3)}_1$, characterizes the slope of sigmoid
$[Z_*, Z^*]$	active noise range of an algorithm, where performance changes significantly
$f^{-1}(\cdot)$	inverse sigmoid function
	SRF dimensions
m	minimal performance of a classification algorithm
M	maximal performance of a classification algorithm
r_{alg}	span of the performance range
c	slope indicator of sigmoid function of an algorithm
d_{alg}	width of the active area of algorithm performance
r_{alg}/d_{alg}	performance improvement of an algorithm over signal-to-noise ratio change

to the monotonicity assumption made in [20]. For the second order derivative, $f''(Z)$, it holds that $f''(Z) = 0$ if $Z = d + \frac{1}{c}\log b$, and the point $Z_{inf} = d + \frac{1}{c}\log b$ is *the point of inflection*, which shows when the curvature of the function changes its sign.

In the case of the sigmoid, the point of inflection, Z_{inf} (the middle point in Figure 2), is the point of symmetry, corresponding to the average performance of the algorithm. Furthermore, Z_{inf} indicates the shift of the sigmoid with respect to the origin, which is equal to $d + \frac{1}{c}\log b$. So, the shift of the sigmoid and the average performance for the signal-to-noise range depends on three parameters: b, c, and d.

Learning algorithms can be compared based on the slope of their sigmoids. The slope of the sigmoid reflects the improvement of an expected performance of an algorithm per change in the signal-to-noise ratio. To estimate the slope, we use the distance d_s between the point of inflection Z_{inf} (zero of the second derivative) and the zeros of the third derivative (both zeros of the third derivative are at the same distance from the point of inflection). Figure 2 shows the sigmoid curve along with its points of interest.

The zeros of the third order derivative are $Z^{(3)}_{1,2} = d - \frac{1}{c}\log\frac{2\pm\sqrt{3}}{b}$. Thus, we have for d_s: $d_s = Z^{(3)}_1 - Z_{inf} = Z_{inf} - Z^{(3)}_2 = \frac{1}{c}\log\frac{1}{2-\sqrt{3}}$. The larger this distance is, the more expanded the sigmoid curve is. Since $\log\frac{1}{2-\sqrt{3}} \approx 1.32$, d_s is in inverse proportion to parameter c: $d_s = \frac{a}{c}$, where $a \approx 1.32$. We find that the parameter c directly influences the slope of the sigmoid curve. We term c as the *slope indicator* of the CTF. Figure 3 depicts sigmoids with different c values, illustrating gradual and sharp slopes of the sigmoid function.

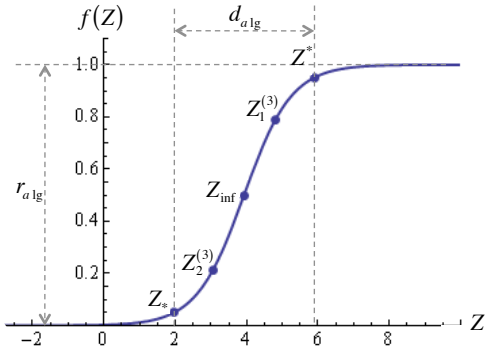


Fig. 2: Sigmoid function and points of interest (i.e. an inflection point, essentially the point where the curve changes significantly).

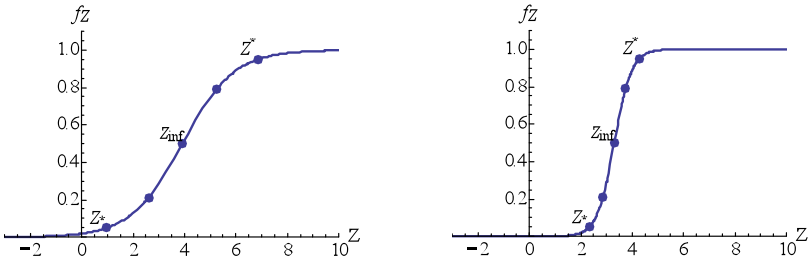


Fig. 3: Varying c parameter. $c = 1$ (left), $c = 3$ (right). Other parameters are the same: $m = 0$, $M = 1$, $b = 2.5$, $d = 3$.

In the following section, we formulate and discuss dimensions that characterize the behavior of algorithms, based on the axes of comparison discussed above.

4.1 SRF Dimensions

In this section we determine several SRF dimensions based on the sigmoid properties, in addition to m, M, r_{alg}, and c, discussed in Section 4. We define *active noise range* as the range $[Z_*, Z^*]$ in which the change in noise induces a measurable change in the performance. In order to calculate $[Z_*, Z^*]$ we assume that there is a good-enough performance on a given task, approaching M for a given algorithm. We know that $f(Z) \in (m, M)$, and we say that the performance is "good enough" if $f(Z) = M - (M - m) * p$, $p = 0.05$[1]. We define

[1]Instead of 0.05, one can use any value close to 0, describing a normalized measure of distance from the optimal performance. In the case of $p = 0.05$, the distance from the optimal performance is 5%.

the *learning improvement* of the algorithm as the size of the signal-to-noise interval in which $f(Z) \in [m + (M - m) * p, M - (M - m) * p]$. Then, using the inverse

$$f^{-1}(y) = d - \frac{1}{c} \log \left(\frac{1}{b} \left(\frac{M - m}{y - m} - 1 \right) \right),$$

where $y = f(Z)$, we calculate the points Z^* and Z_*, which respectively are the bottom and the top points in Figure 2 for a given p. We term the distance $d_{alg} = Z^* - Z_*$ as the *width of the active area* of the machine learning classifier (see Figure 2). Then we define a ratio between the width of the active area and the span of the performance range $\frac{r_{alg}}{d_{alg}}$, which describes (and is termed as) the *learning performance improvement over signal-to-noise ratio change.*

In the following paragraphs we describe how this analysis allows us to compare the performance of learning algorithms in the presence of noise.

5 Using SRF to Compare Algorithms

Given the performance dimensions described above, we can compare algorithms as follows. For *performance*-related comparison we can use minimal performance m, maximal performance M, and the span of performance range r_{alg}. Algorithms not affected by the presence or absence of noise will have a minimal r_{alg} value. In a setting with a randomly changing level of noise, this parameter is related to the possible variance in performance. Related to the *sensitivity* of performance to the change of the signal-to-noise ratio, we can use the following dimensions.

1. The *active noise range* $[Z_*, Z^*]$, which shows how early the algorithm starts operating (measured by Z_*) and how early it reaches good-enough performance (measured by Z^*). We say that the algorithm *operates* when the level of noise in the data is within the active noise range of the algorithm.
2. The width of the active area $d_{alg} = Z^* - Z_*$ of the algorithm, which is related to the speed of changing performance for a given r_{alg} in the domain of noise. A high d_{alg} value indicates that the algorithm varies its performance in a broad range of signal-to-noise ratios, implying less performance stability in an environment with heavily varying degrees of noise.
3. The parameter c of the sigmoid function (*slope indicator*), which is related to the distance $d_s = 1.32/c$. The distance has similar properties as the inversion of r_{alg}/d_{alg}, but it is independent of the predefined parameter p, which shows the percentage of performance considered as good-enough for a given task and is not directly connected to the active noise range. d_s reflects the change in the slope of the function. If c is large, then d_s is small and indicates higher stability of the algorithm in the presence of noise, and vice versa.
4. The point of inflection Z_{inf}, which shows the signal-to-noise ratio for which an algorithm gives the average performance. The parameter can be used

to choose the algorithm that reaches its average performance earlier in a noisy environment, or the algorithm whose speed of improvement changes earlier.

A parameter related to both *performance* and *sensibility* is the learning performance improvement over signal-to-noise ratio change, r_{alg}/d_{alg}. It can be used to determine whether reducing the noise in a dataset is expected to have a significant impact on the performance. An algorithm with a high value of r_{alg}/d_{alg} would imply that it makes sense to put more effort into reducing noise. Furthermore, using this dimension a decision maker can choose more stable algorithm, when the variance of noise is known. In this case, the algorithm with the lowest value of r_{alg}/d_{alg} should be chosen in order to limit the corresponding variance in performance.

Based on the above discussion, we favor the classifiers with the following properties:
- higher maximal performance M,
- larger span of performance range r_{alg},
- higher learning performance improvement over signal-to-noise ratio change r_{alg}/d_{alg},
- shorter width of the active area of the algorithm d_{alg}, and
- larger slope indicator c.

We expect to get high performance from an algorithm if the level of noise in the dataset is very low, and low performance if the level of noise in the dataset is very high. Decision makers can easily formulate different criteria, based on the proposed dimensions (summarized in Table 2).

Based on the above, the workflow to select an algorithm given some user preference is as follows: we select an algorithm; we sample the input data for training; we estimate the SRF parameters; we repeat the process for different algorithms; finally, we analyze the SRF parameters in order to select the algorithm with the most appropriate behavior, i.e., the one that best matches the user preferences.

6 Experimental Evaluation

In the following paragraphs we describe the experimental setup, the datasets used, and the results of our experiments. We first consider non-sequential and then sequential classifiers.

6.1 Experimental Setup for Non-Sequential Classifiers

In our study we applied the following machine learning algorithms, implemented in Weka 3.6.3 [22]:
(a) IBk – k-nearest neighbor classifier;
(b) Naïve Bayes classifier;
(c) SMO – support vector classifier (cf. [28]);

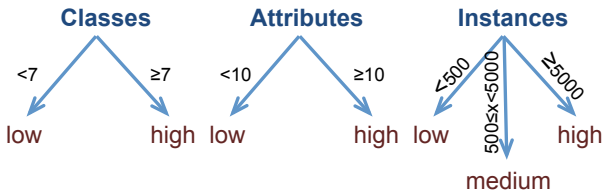


Fig. 4: Dataset grouping labels.

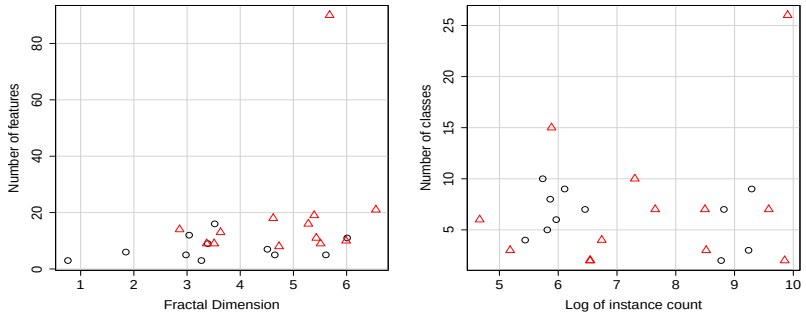


Fig. 5: Distribution of the dataset characteristics. Real data: triangles, Artificial: circles.

(d) NbTree – a decision tree with Naïve Bayes classifiers at the leaves;
(e) JRip – a RIPPER [11] rule learner implementation.

We have chosen representative algorithms from different families of classification approaches, covering popular classification schemes. The experiments were performed on a 2.4GHz machine with 4GB RAM.

We used a total of 24 datasets for our experiments. Most of the real datasets come from the UCI machine learning repository [17], and one from the study [19]. Fourteen of the dataset are real, while ten are synthetic. All the datasets are divided into groups according to the number of classes, attributes (features) and instances in the dataset, as shown in Figure 4.

There are 12 possible groups that include all combinations of the parameters. Two datasets from each group were employed for the experiments. We created artificial datasets in the cases were real datasets with a certain set of characteristics were not available. The distribution of the dataset characteristics is illustrated in Figure 5. The traits of the datasets illustrated are the number of classes, the number of attributes (features), the number of instances, and the estimated intrinsic (fractal) dimension.

Table 3 contains the description of the datasets and their sources.

Table 3: The description of the datasets used. x_1 is a number of classes, x_2 is the number of attributes, x_3 is the number of instances, and x_4 is the fractal dimensionality of a dataset.

Dataset	**Parameters**			
	# of classes	# of attr	# of inst	fract dim
	x_1	x_2	x_3	x_4
1. Wine[1]	3	13	178	3.63
2. Australian Credit Approval[2]	2	14	690	2.86
3. real from [19]	4	6	230	1.85
4. Yeast[3]	10	8	1484	4.73
5. artificial	5	12	335	3.04
6. artificial	9	16	450	3.52
7. artificial	10	5	310	2.98
8. artificial	7	9	637	3.39
9. artificial	8	3	352	0.76
10. Breast Cancer Wisconsin[4]	2	9	699	3.51
11. Breast Tissue[5]	6	9	106	3.37
12. Letter Recognition[6]	26	16	20000	5.28
13. artificial	6	5	390	4.65
14. artificial	3	7	10287	4.51
15. artificial	2	3	6452	3.27
16. Vehicle Silhouettes[7]	4	18	846	4.62
17. MAGIC Gamma Telescope[8]	2	10	19020	5.99
18. Waveform[9]	3	21	5000	6.55
19. Shuttle[10]	7	9	14500	5.51
20. artificial	9	5	10845	5.61
21. Libras Movement[11]	15	90	360	5.68
22. Image[12]	7	19	2100	5.39
23. Wine Quality[13]	7	11	4898	5.43
24. artificial	7	11	6762	6.01

We build ten artificial datasets using the following procedure. Having randomly sampled the number of classes, features and instances, we sample the parameters of each feature distribution. We assume that the features follow the Gaussian distribution with mean value (μ) in the interval $[-100, 100]$ and standard deviation(σ) in the interval $[0.1, 30]$. The μ and σ intervals allow

[1]See http://archive.ics.uci.edu/ml/datasets/Wine

[2]See http://archive.ics.uci.edu/ml/datasets/Statlog+%28Australian+Credit+Approval%29

[3]See http://archive.ics.uci.edu/ml/datasets/Yeast

[4]See http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Prognostic%29

[5]See http://archive.ics.uci.edu/ml/datasets/Breast+Tissue

[6]See http://archive.ics.uci.edu/ml/datasets/Letter+Recognition

[7]See http://archive.ics.uci.edu/ml/datasets/Statlog+%28Vehicle+Silhouettes%29

[8]See http://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope

[9]See http://archive.ics.uci.edu/ml/machine-learning-databases/waveform/

[10]See http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/shuttle/

[11]See http://archive.ics.uci.edu/ml/datasets/Libras+Movement

[12]See http://archive.ics.uci.edu/ml/machine-learning-databases/image/

[13]See http://archive.ics.uci.edu/ml/datasets/Wine+Quality

overlapping features across classes. The description of the parameters of the generated datasets is shown in Table 3.

The noise was induced as follows. We created the stratified training sets, equal in size to the stratified test sets. To induce noise, we created noisy versions of the training sets by mislabeling the instances. Using different levels l_n of noise, $l_n = 0, 0.05, ..., 0.95$[1], a training set with l_n noise is a set with fraction l_n of mislabeled instances. Hence we obtained 20 versions of the datasets with varying noise levels.

The equal percentages of instances from different classes (for stratification) were put into test and training sets (as in the original dataset with 0% of noise), using the following procedure:

1. Calculate the number N of observations in the original dataset; the number of observations in the test or training set should be the integer part of $N/2$;
2. Calculate the percentages of each class in the original dataset;
3. Calculate the number of observations from each class that should be in the training and test sets based on the percentages from the previous step (rounding the numbers if necessary);
4. Divide the observations according to the classes to obtain "class datasets", i.e., sets of instances with only one class per set;
5. From each "class dataset", sample observations (based on the results of step 3) for the training set, and place the remaining observations of each "class dataset" into the test set;
6. Make "noisy" training datasets from the original training dataset with different levels l_n of noise.
7. Get the performance of the model trained on each "noisy" dataset $l_n = 0, 0.05, ..., 0.95$ and tested on a noise-free stratified test set.

Given the performance of a classifier for a particular dataset, we estimate the parameters of the sigmoid function for each algorithm-dataset pair using a genetic algorithm (described in detail in Appendix A.1). The idea is that the algorithm tries to find parameters that minimize the distance between the distribution of values of the real data to that of the estimated sigmoid. To this end, we use as a fitness measure a function of the D statistic of the Kolmogorov-Smirnov test [36], which measures the maximum distance between the Cumulative distribution functions of the two sample sets. Having the sets of estimated sigmoids describing the performance for particular algorithms and datasets with certain characteristics, we can reason about the properties of the algorithms along the SRF dimensions.

6.2 Experimental Setup for Sequential Classifiers

We now consider sequential datasets, where the order of instances is important. Using these datasets, we study whether sequential classifiers adhere to the

[1] We note that high levels of noise such as 95% are often observed in the presence of *concept drift*, e.g., when learning computer-user browsing habits in a network environment with a single IP, and several different users sharing it.

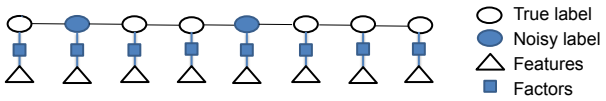


Fig. 6: Linear chain CRF architecture.

sigmoid rule framework. The main difference to the non-sequential case is that for the classification task we also use the information about the previous instances in addition to the features of the current instance. For sequential data, we apply three classification algorithms based on Hidden Markov Models (HMM) [45], Conditional Random Fields (CRF) [32], and Recurrent Neural Networks (RNN) [15].

For the experiments, we consider the whole sequence of observations as an instance. For each algorithm, we define the order of dependency, and consider sequences of the length corresponding to that order. For the implementation of these sequential classifiers, we use the jahmm [26], Mallet [37], and Weka [22] libraries for HMM, CRF, and RNN, respectively. The experiments and the libraries are implemented in Java, and the classification algorithms are called as library functions. The experiments were performed on a 2.67GHz machine with 4GB RAM.

Hidden Markov Models. For HMM-based classification, we perform the experiments in two settings by using HMMs of the 3rd and the 4th order. In Figures 8a and 8b, we depict the sigmoids of the HMM algorithm for the "Robot walk 4" dataset. The green solid line corresponds to the true measurements of the performance of the HMM-based classifier for different values of signal-to-noise ratio Z, while the dashed red line corresponds to the estimated sigmoid, the parameters being assessed with a generic algorithm (see Appendix A.1).The Pearson's correlation test between the obtained performance and the estimated performance based on the sigmoid function shows statistically significant correlation values, with $coef > 0.99$ for significance level $\alpha = 0.01$. This holds for both the first and the second setting (the third and the fourth order models, correspondingly). Similar results were obtained for the rest of the datasets and sequential learners. These results confirm that the sequential learners adhere to the "Sigmoid Rule" framework.

Conditional Random Fields. To better cover sequential classifiers, we employ and study linear and quadratic CRFs. These go beyond class labels, which HMMs use, and also take into account data attributes to determine a class. We experiment with CRF in two different settings, according to the order of the connection between the instances. To label an unseen instance, the most likely Viterbi labeling $y^* = \arg\max_y p(y^{(i)}|x^{(i)})$ is computed. In the first setting we consider a linear chain CRF, in which an instance depends only on the previous one, as depicted in Figure 6. Since we consider only label noise, we induce noise only in the labels of the instances. In the second setting, we consider the model of the second order, where an instance depends on the two previous instances. As an example, the sigmoids of the CRF algorithm for the

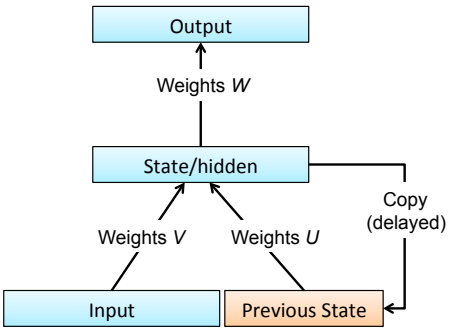


Fig. 7: The architecture of RNN.

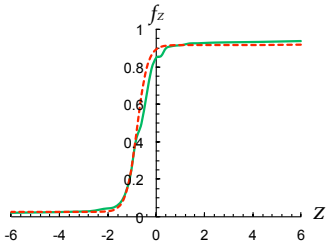
"Robot walk 4" dataset are shown in Figures 8c and 8d. Like in the case with HMMs, the sigmoids fit the performance curves of the CRF classifier well for all the datasets.

Recurrent Neural Networks. Whereas HMM takes into account only the data labels of the sequence, and CRF uses both class labels and the attributes of the previous instances in the sequence in order to classify the current instance, RNN uses the attributes of the current instance, and only the label of the previous instances. For RNN-based classification, we provide the previous label as input to calculate the output of the current instances (see Figure 7), as in the work [4], and we use two settings with 2nd and 3rd order dependencies: in the first setting, the two previous labels are used, while in the second setting the three previous labels are used. The sigmoids of the RNN algorithms for the "Robot walk 4" dataset are shown in Figures 8e and 8f. According to Pearson's correlation test for setting 1 ($coef = 0.9947$, $p - value = 0.0053$) and setting 2 ($coef = 0.9962$, $p - value = 0.0038$) between the obtained performance and estimated sigmoid function, we can say that the classifier based on RNN also follows the "Sigmoid Rule". As in the previous cases of HMMs and CRFs, the sigmoids fit well the performance curves for all the datasets for the RNN classifier. Thus, the "Sigmoid Rule" fits RNN-based classifiers as well.

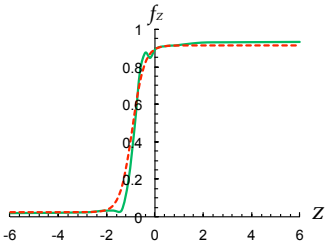
6.2.1 Datasets

We used 18 real datasets for our sequential experiments. All the datasets contain time series where the instances appear in the chronological order. The label of the last observation is assigned to the sequence to imply causality.

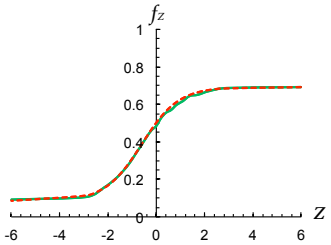
The distribution of the characteristics of these datasets is illustrated in Figure 9. These characteristics are: the number of classes, the number of attributes, the number of instances, and the largest lag value that corresponds to significant autocorrelation of the labels. Autocorrelation is a linear dependence of a variable with itself at two points in time [5]. We use the autocorrelation



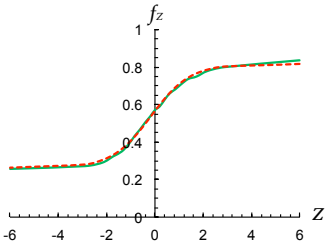
(a) Sigmoid CTF of HMM 1st setting.



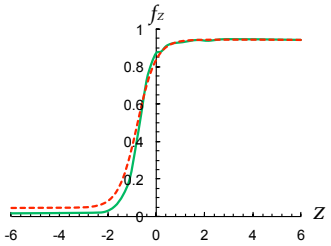
(b) Sigmoid CTF of HMM 2nd setting.



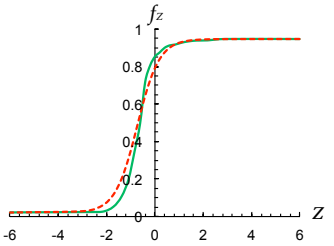
(c) Sigmoid CTF of CRF 1st setting.



(d) Sigmoid CTF of CRF 2nd setting.



(e) Sigmoid CTF of RNN, 1st setting.



(f) Sigmoid CTF of RNN, 2nd setting.

Fig. 8: Sigmoid CTF of the two settings of the HMM, CRF and RNN-based classification algorithm for the "Robot walk 4" dataset, indicating different algorithm behaviors. Green solid line: true measurements, dashed red line: estimated sigmoid.

function (ACF) implemented in R[1] to calculate autocorrelation with maximum lag equal to 50 for all datasets. We choose the first lag that gives the smallest autocorrelation falling out of the significance band, as shown, for example, in Figure 10 for the Pioneer gripper dataset, where we chose lag equal to 37.

Description of the datasets are shown in Table 4.

[1] `http://stat.ethz.ch/R-manual/R-patched/library/nlme/html/ACF.html`

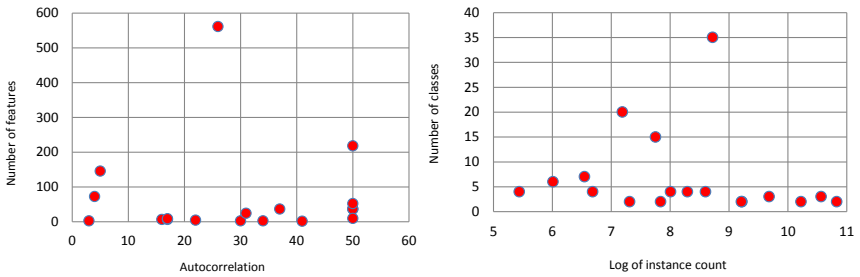


Fig. 9: Distributions of the characteristics of sequential datasets.

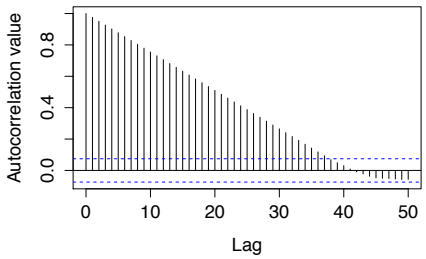


Fig. 10: Autocorrelation plot of the Pioneer gripper dataset.

We induce noise into sequential data in a way similar to the independent data. We randomly split all data into equally sized training and test parts. Mislabeled instances are in the training data only. We induce varying levels of noise from 0% to 100% as follows. The noisy version of a class C_i of a training set is created by selecting the level of noise l_n, and replacing some instance with an instance of another class with probability l_n. Then sequential classifiers are trained using the training data with various levels of noise, and are used to predict the test data. Given the classification of all the test instances, we

[1] Dataset from [19].

[2] See http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

[3] See http://archive.ics.uci.edu/ml/datasets/Pioneer-1+Mobile+Robot+Data

[4] See http://archive.ics.uci.edu/ml/datasets/Wall-Following+Robot+Navigation+Data

[5] See http://archive.ics.uci.edu/ml/datasets/Diabetes

[6] See http://archive.ics.uci.edu/ml/datasets/Ozone+Level+Detection

[7] See http://archive.ics.uci.edu/ml/datasets/OPPORTUNITY+Activity+Recognition

[8] See http://archive.ics.uci.edu/ml/datasets/CalIt2+Building+People+Counts

[9] See http://archive.ics.uci.edu/ml/datasets/PAMAP2+Physical+Activity+Monitoring

[10] See http://archive.ics.uci.edu/ml/datasets/Daphnet+Freezing+of+Gait

[11] See http://archive.ics.uci.edu/ml/datasets/Dodgers+Loop+Sensor

Table 4: Dataset descriptions. x_1 is the number of classes, x_2 is the number of instances, x_3 is the autocorrelation lag, and x_4 is the number of attributes (features).

Dataset	**Parameters**			
	# of classes	# of inst	autocorr lag	# of attr
	x_1	x_2	x_3	x_4
1. Climate[1]	4	230	16	6
2. Human Activity[2]	6	408	26	561
3. Pioneer gripper[3]	7	697	37	36
4. Robot walk 4[4]	4	800	22	4
5. Diabetes[5]	20	1327	3	2
6. Ozone Level 1 hour[6]	2	1500	4	72
7. Pioneer turn[3]	15	2324	50	36
8. Ozone Level 8 hour[6]	2	2534	5	145
9. Opportunity[7]	4	3000	50	218
10. Robot walk 2[4]	4	4000	30	2
11. Robot walk 24[4]	4	5434	31	24
12. Pioneer move[3]	35	6129	50	36
13. Callt2[8]	2	10082	41	1
14. Electric 2[1]	2	10000	17	6
15. PAMAP[9]	3	16000	50	52
16. Electric 3[1]	2	27552	17	8
17. Daphnet Freezing[10]	3	38774	50	9
18. Dodger[11]	2	50400	34	2

compute the overall performance of the algorithm using classification accuracy:

$$P = \frac{\#\text{CorrectClassifications}}{\#\text{TotalClassifications}}$$

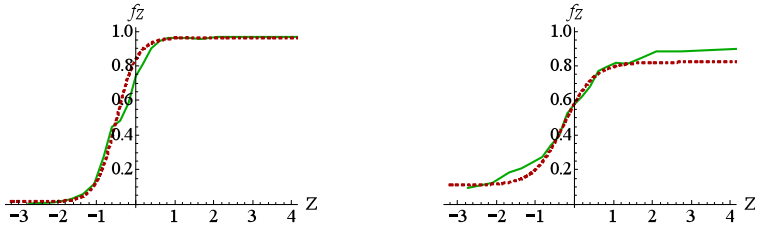
We perform repeated random sub-sampling validation with 20 splits by randomly choosing half of the instances as training data and the remaining instances as test data, and calculate the average performance of the classifier. For the sequential datasets the correct order of the instances is kept for both training and test phases. The parameters of the corresponding sigmoid are assessed using the values of signal-to-noise ratio and corresponding average performance levels.

In the following sections we consider sequential classifiers in more detail.

6.3 Experiments on Classifier Behavior Comparison

We perform experiments of "noisy" classification using repeated random sub-sampling validation[12] with 10 splits per algorithm per dataset per noise level, and calculate the average performance for each setting. Repeated random sub-sampling is a common technique of cross-validation, in which a dataset is randomly split into the training and testing parts and the accuracy results are

[12]Repeated random sub-sampling validation is also known as Monte Carlo cross-validation [31].



(a) Sigmoid CTF of the SMO algorithm. (b) Sigmoid CTF of the IBk algorithm.

Fig. 11: Sigmoid CTF of SMO and IBk algrorithms for "Wine" dataset. Green solid line: True Measurements, dashed red line: estimated sigmoid.

averaged over the splits. In the case of sequential learners, we use 20 splits. Given the 20 levels of the signal-to-noise ratio and the corresponding algorithm performance, (i.e., classification accuracy) we estimated the parameters of the sigmoid. The search in the space of sigmoid parameters is performed by a genetic algorithm[1], as was proposed in [19].

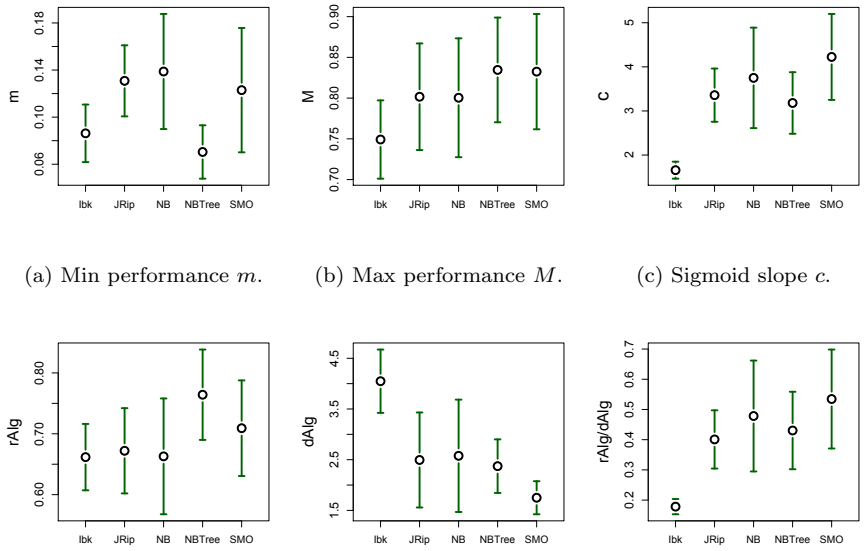
A sample of the true and sigmoid-estimated performance graphs for varying levels of noise for a non-sequential classifier can be seen in Figure 11. The corresponding graphs for sequential classifiers are shown in Figure 8. Although in our experiments the parameters of the sigmoid were estimated offline, the Sigmoid Rule Framework can be applied in an online scenario, following a training period.

Figure 12 illustrates the values of the mean and the standard deviation of the SRF parameters per algorithm, over all 24 datasets for the non-sequential classifiers. As an example of an interpretation of the figure using SRF, the plots indicate that (for the range of datasets studied) SMO is expected to improve its performance faster on average than all other algorithms, when the signal-to-noise ratio increases. This conclusion is based on the $\frac{r_{alg}}{d_{alg}}$ dimension and the values of the slope indicator c (refer to Figures 12f, 12c). Figure 11 offers a visual comparison of the performances of the SMO and IBk algorithms for the "Wine" dataset, where we can see that SMO improves its performance twice as fast as IBk.

At the same time, the confidence interval for $\frac{r_{alg}}{d_{alg}}$ and c of the Naïve Bayes algorithm overlaps with that of the SMO, so these algorithms can also be recommended if the same criteria are optimized. Nevertheless, SMO and Naïve Bayes algorithms can be well differentiated by the d_{alg} criterion (see Figure 12e).

IBk has a smaller potential for improvement of performance (but also smaller potential for loss) than SMO, when noise levels change, given that the span of the performance range r_{alg} is higher for SMO (Figure 12d). This difference can also be seen in Figure 11, where the distance between the minimum and the maximum performance values is larger for SMO.

[1]The settings of the genetic algorithm can be found in the appendix.



(a) Min performance m. (b) Max performance M. (c) Sigmoid slope c.

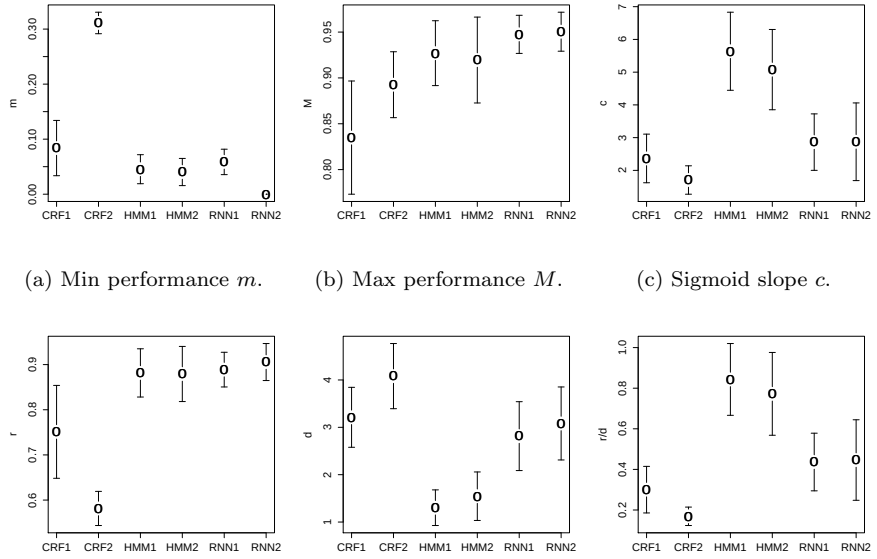
(d) Performance span r_{alg}. (e) Active area width d_{alg}. (f) Perf. improvement $\frac{r_{alg}}{d_{alg}}$.

Fig. 12: Estimated SRF parameters per algorithm. X axis labels (left-to-right): IBk, JRip, NB, NBTree, SMO.

Figure 13 illustrates the values of the mean and the standard deviation for the SRF parameters per algorithm, over all 18 datasets for the sequential case. The plots indicate that HMM is expected to improve its performance faster than the other algorithms, when the signal-to-noise ratio increases (Figures 13c and 13f). Though the two settings of RNN perform similarly along these parameters, they are clearly distinct from the rest of the algorithms. On the other hand, CRF has smaller potential for improvement of its performance, but also smaller risk for low performance than HMM, when noise levels change (Figures 13f and 13d). An example of the fast improvement of HMM when compared to other families of algorithms can be seen on Figure 8.

As can be seen from Figure 13, all the families of the sequential classifiers have their specific behavior along the devised SRF dimensions, with certain variations depending on the order of the model.

We stress that the parameter estimation does not require previous knowledge of the noise levels, but is dataset dependent. In the special case of a classifier selection process, having an estimate of the noise level in the dataset helps to reach a decision through the use of SRF. In the following, we demonstrate the existence of this connection between the parameters of the sigmoid curve and the characteristics of the datasets. Therefore, SRF can help in choosing the better learner, provided that we know the characteristic of the dataset.



(a) Min performance m. (b) Max performance M. (c) Sigmoid slope c.

(d) Performance span r_{alg}. (e) Active area width d_{alg}. (f) Perf. improvement $\frac{r_{alg}}{d_{alg}}$.

Fig. 13: SRF parameters per algorithm. X axis labels (left-to-right): CRF 1st setting and 2nd setting, HMM 1st setting and 2nd setting, RNN 1st setting and 2nd setting.

7 Modeling Expected Performance in New Datasets

We now study the connection between the dataset characteristics and the sigmoid parameters, irrespective of the choice of the algorithm. In essence, this study shows if and how the framework can be used to estimate performance of algorithms in a noisy setting, in the presence of a new dataset.

We consider the results obtained from all the algorithms as different samples of the SRF parameters for a particular dataset. We use regression analysis to observe the cumulative effect of the dataset characteristics on a single parameter, and we use correlation analysis to detect any connection between each pair of the dataset characteristic and the sigmoid parameter. We examine the connections between the dataset characteristics and the sigmoid parameters both individually, and all together, in order to draw the complete picture.

7.1 Linear regression

We use the "lm" method (Fitting Linear Models) implemented in the package "stats" in R 2.11.1 to estimate the values of the parameters of the linear

regression model for each chosen SRF parameter y:

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 + \varepsilon,$$

where x_i, $i = 1, 2, ...$ correspond to the features of a dataset, such as the number of classes, the number of instances, and other dataset characteristics specific to non-sequential and sequential cases, which are going to be discussed below; ε is the error term, $E(\varepsilon) = 0$. For choosing the best model, we use a modification of the *bestlm* function from *nsRFA* package in R. This function also exploits the function *leaps* of the R package *leaps*. The criterion for the best model is the adjusted R^2 statistic (the closer the adjusted R^2 is to one, the better the estimation is). R^2 defines the proportion of variation in the dependent variable (y) that can be explained by the predictors (X variables) in the regression model.

As some variation of y can be predicted by chance, the adjusted value of R_a^2 is used. Adjusted R^2 takes into account the number of observations (N) and the number of predictors (k), and is computed using the formula:

$$R_a^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - k - 1}.$$

After applying the linear model, we exclude the non-significant parameters, and search for the model that maximizes the adjusted R^2 statistic (i.e., the best linear model).

In addition to R_a^2, we applied a statistical test with significance level $\alpha = 0.05$ for each chosen regression model in order to check if the model fits the data well. If the model describes the distribution of the values statistically significantly, then, after the hypothesis testing, the p-value should be within the significance level.

We applied a leave-one-out validation process, where one dataset is left out from training and used for testing on every run. We performed this analysis separately with each of the variables m, M, r_{alg}, d_{alg}, $\frac{r_{alg}}{d_{alg}}$, and c as a dependent variable y.

Non-sequential case.

For the non-sequential classifiers, the following dataset parameters were chosen: a number of classes (x_1), a number of features (x_2), a number of instances (x_3), and the intrinsic dimensionality (x_4) of a dataset[1](i.e., the fractal correlation dimension [9,48], calculated using a box counting process).

The results of model fitting and prediction of the SRF dimensions are reported in Table 5, where average errors between the observed and predicted SRF dimensions are shown. For each SRF dimension chosen, we have observed 5 values (since 5 machine learning algorithms were used), and having estimated them for 24 datasets, we ended up with 120 predictions for a single SRF dimension. We calculated four types of errors: (1) MSE – root mean square error, $MSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$; (2) MAE – mean absolute

[1]We would like to thank Christos Faloutsos for kindly providing the code for the fractal dimensionality estimation.

Table 5: Prediction error of linear regression models for non-sequential classifiers.

Parameters	**Error measures**					**average(R_a^2)**
	MSE	MAE	RMSE	RMAE	NRMSE	
Min performance m	0.11	0.09	148.92	29.17	0.24	0.54
Max performance M	0.35	0.30	0.51	0.41	0.71	0.88
Performance span r_{alg}	0.32	0.27	0.71	0.46	0.37	0.85
Active area width d_{alg}	1.98	1.41	0.97	0.68	0.15	0.67
Perf. improvement $\frac{r_{alg}}{d_{alg}}$	0.37	0.27	4.83	1.46	0.23	0.55
Sigmoid slope c	2.40	1.81	1.41	0.78	0.26	0.65

error, $MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$; (3) RMSE – relative root mean square error, $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{\hat{y}_i - y_i}{y_i})^2}$; (4) RMAE[1] – relative mean absolute error, $RMAE = \frac{1}{n}\sum_{i=1}^{n}|\frac{\hat{y}_i - y_i}{y_i}|$; and (5) NRMSE – normalized root mean square error, $NRMSE = \frac{MSE}{y_{max} - y_{min}}$. In the above formulas, $\hat{y}_i$ stands for the predicted value of the parameter, while y_i is the actual value, and n is the number of predictions.

The last column of Table 5 shows the average of the adjusted R^2 statistic for models that where estimated for all the SRF dimensions (averaged on the 24 datasets).

Figure 14 illustrates how our models fit the test data, showing that in most cases the true values of the sigmoid parameters for each dataset (illustrated by circles that correspond to 5 algorithms for each test dataset i, $i = 1, 2, ..., 24$) are within the 95% confidence level zone around the estimated values. This finding further supports the connection between the dataset parameters and the SRF dimensions.

According to the results, the chosen parameters of the datasets can be used to reason about the parameters of the sigmoid. This allows us to recommend algorithms for a new dataset with known characteristics, if we have estimated the SRF dimensions for a set of algorithms on the datasets with similar characteristics.

The relation between noise level and average classification performance over the 'Climate' dataset is illustrated in Figure 15. There, we depict the curve of the estimated performance, based on the regression models, as well as the curves of the actual performances of several algorithms. The plot shows that the estimation is quite good for the IBk, NBTree and JRip algorithms. The estimation for the other algorithms is not as good, which may be attributed to the fact that in our experiments, we trained the regression models using all available datasets that have very different characteristics. We expect that training models with a large number of datasets with similar characteristics can further improve the accuracy.

Sequential case.

[1]RMAE is also known as mean absolute percentage error.

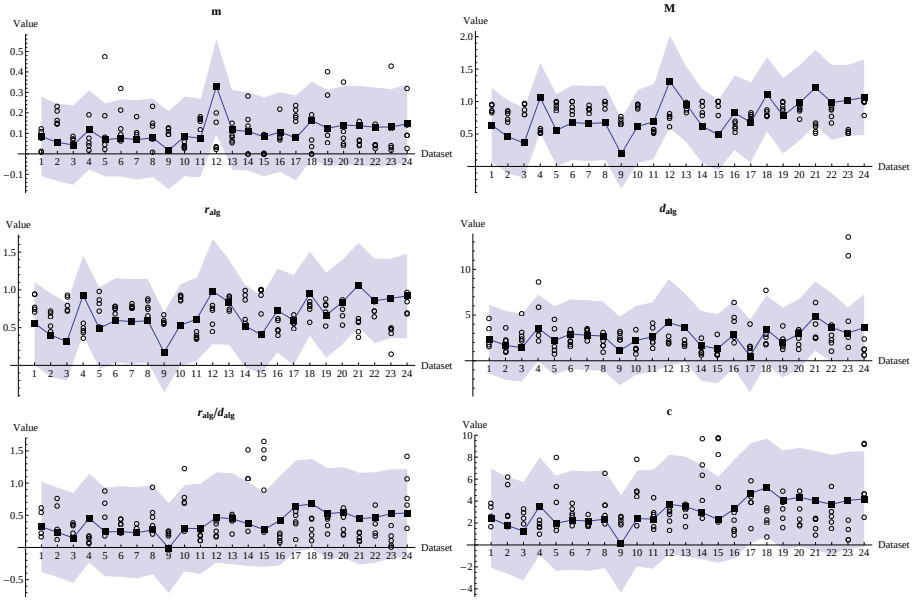


Fig. 14: Real and estimated values of the sigmoid parameters for non-sequential classifiers. Real values: Black rectangles, Estimated values: circles, Gray zone: 95% prediction confidence interval.

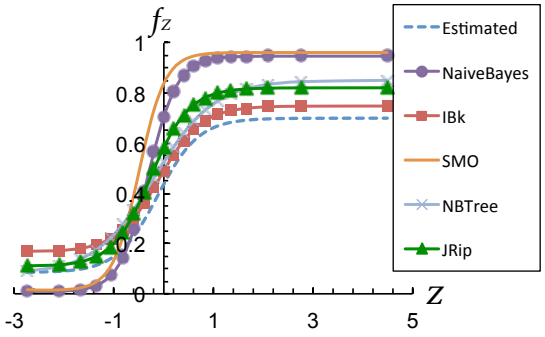


Fig. 15: Sigmoid with regression-estimated parameters with actual sigmoids of the corresponding classification algorithms for Climate dataset.

In this set of experiments, not all the sequential classification algorithms use the features of the data instances. Therefore, we examine the influence of the dataset on the CTF parameters by studying the number of classes (x_1), the number of instances (x_2), and the maximal autocorrelation lag of a dataset (x_3), which is described in more detail in Section 6.2.

We also applied a leave-one-out method for six dependent variables, as in the non-sequential dataset case. The results of model fitting and predictions of SRF dimensions are reported in Table 6.

For each SRF dimension chosen, we have observed 6 values, since 3 sequential learning algorithms were used, and each algorithm is applied in 2 different settings of dependency order. These 6 SRF dimensions are estimated for 18 datasets, thus 108 predictions were done for each SRF dimension. We also calculated four types of errors, as with the non-sequential datasets (Table 6).

Table 6: Prediction error of linear regression models for sequential data.

Parameters	**Error measures**					**average(R_a^2)**
	MSE	MAE	RMSE	RMAE	NRMSE	
Min performance m	0.13	0.10	46.80	8.67	0.31	0.25
Max performance M	0.41	0.33	0.45	0.37	0.71	0.81
Performance span r_{alg}	0.37	0.30	0.48	0.39	0.33	0.81
Active area width d_{alg}	2.01	1.57	0.98	0.69	0.44	0.58
Perf. improvement $\frac{r_{alg}}{d_{alg}}$	0.36	0.29	1.24	0.88	0.18	0.67
Sigmoid slope c	2.38	1.88	0.85	0.68	0.20	0.67

Just like the non-sequential data, for sequential data classification, we can see the relationship between the dataset parameters and the SRF dimensions. Figure 16 illustrates how our models fit the test data.

The results show that most of the true parameter values are within the 95% prediction confidence levels. Thus, we can use the dataset parameters in order to reason about the parameters of the sigmoid of the algorithms. This enables us to recommend the algorithm of choice in advance for a new dataset with known characteristics. We can additionally recommend the order of the sequential classifier, if we have computed the SRF dimensions for a set of algorithms for datasets with similar characteristics.

7.2 Logistic regression

We also employed logistic regression, by applying the same leave-one-out process, for the purpose of predicting the SRF dimensions. While applying logistics regression, we use the Akaike Information Criterion (AIC) instead of the adjusted R^2_{adj} in order to choose the best model.

Being a measure of the relative quality of a statistical model for given data, AIC provides the means for model selection. In the general case, the AIC is calculated as follows

$$AIC = 2k - 2\ln(L),$$

where k is a number of parameters in the statistical model, and L is the maximized value of the likelihood function for the estimated model [2]. We choose the best prediction model by minimizing the average AIC.

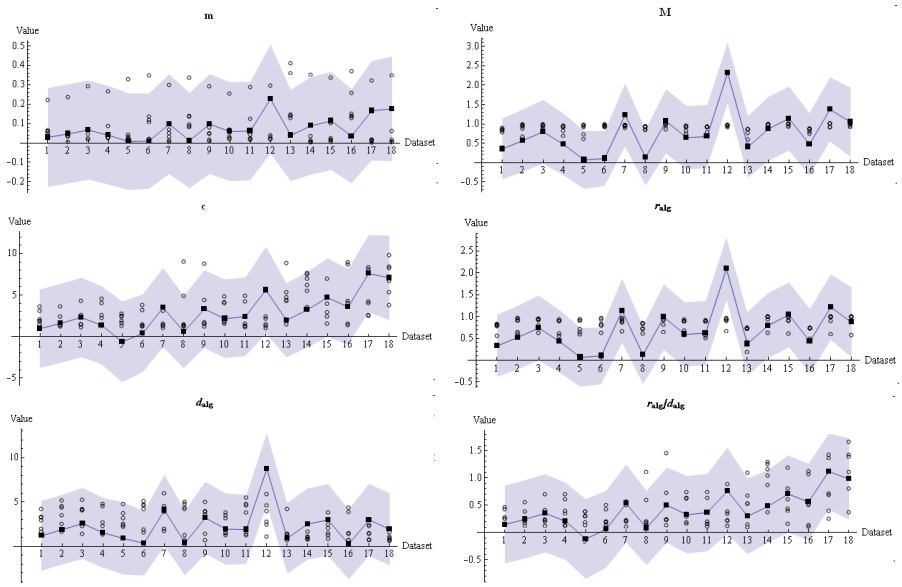


Fig. 16: Real and estimated values of the sigmoid parameters for sequential classifiers. Real values: black rectangles, Estimated values: circles, Gray zone: 95% prediction confidence interval.

We also calculate four types of errors, as in linear regression. The results of model fitting and prediction of SRF dimension for non-sequential data sets are shown in Table 7, while the results for sequential data sets are shown in Table 8.

Table 7: Prediction error of logistic regression models for non-sequential data.

Parameters	**Error measures**					**AIC**
	MSE	MAE	RMSE	RMAE	NRMSE	
Min performance m	0.14	0.10	300.26	51.73	0.29	37.26
Max performance M	0.19	0.16	0.27	0.22	0.38	54.30
Performance span r_{alg}	0.20	0.17	0.51	0.30	0.24	120.91
Active area width d_{alg}	2.07	1.27	0.70	0.51	0.16	-220.05
Perf. improvement $\frac{r_{alg}}{d_{alg}}$	0.38	0.29	3.90	1.47	0.23	-289.60
Sigmoid slope c	2.15	1.42	0.76	0.47	0.23	-246.88

According to the experimental evaluation, the logistic regression model gives better prediction for some SRF dimensions, producing relatively smaller errors compared to the linear regression results. For both the non-sequential (Tables 5 and 7) and sequential (Tables 6 and 8) cases, the parameters M and r_{alg} are better predicted with the logistic regression model. Additionally, for the non-sequential case, c is better predicted using the logistic regression

Table 8: Prediction error of logistic regression models for sequential data.

Parameters	Error measures					AIC
	MSE	MAE	RMSE	RMAE	NRMSE	
Min performance m	0.14	0.11	15.70	5.29	0.35	38.11
Max performance M	0.13	0.09	0.15	0.10	0.32	36.19
Performance span r_{alg}	0.17	0.14	0.35	0.20	0.20	68.19
Active area width d_{alg}	2.56	2.04	0.70	0.65	0.48	131.51
Perf. improvement $\frac{r_{alg}}{d_{alg}}$	0.38	0.33	1.96	1.38	0.25	-255.32
Sigmoid slope c	3.38	2.46	0.64	0.60	0.38	-223.99

model. The accurate prediction of the SRF dimensions supports the connection between the dataset characteristics and the performance parameters. Thus, given the characteristics of a dataset, we can reason about the parameters of the sigmoid of the algorithms, which enables us to recommend the algorithms of choice in advance for the dataset at hand.

7.3 Correlation analysis

We used three different correlation coefficients – the Pearson's correlation coefficient for linear correlation, the Spearman's rho, and the Kendall's tau coefficients for monotonic correlation – to analyze the connection between the parameters of the datasets and the SRF dimensions. Each coefficient varies from -1 (total negative correlation) to 1 (total positive correlation). A value of 0 implies that there is no correlation between the variables.

We qualitatively interpret the strength of the absolute values of correlation as follows: $[0.0; 0.1)$ →No Correlation, $[0.1; 0.3)$ →Low Correlation, $[0.3; 0.5)$ →Medium Correlation, $[0.5; 1]$ →Strong Correlation. This interpretation is widely accepted [12], [16], [55], though the borders may vary slightly depending on the domain (for example, in medicine, higher thresholds for strong correlation are usually required [50]).

Non-sequantial case.

In the non-sequential case, the parameters of a dataset are as before: x_1 – the number of classes, x_2 – the number of features, x_3 – the number of instances, and x_4 – the intrinsic dimensionality. For the analysis we use the same data as for the non-sequential regression analysis in Section 7.1. Therefore, for each SRF dimension we have 5 values corresponding to each classifier, and, after training the classifiers on 24 datasets, we obtained 120 values for a single SRF dimension. Then, we correlate the SRF dimension values with the corresponding parameters of a dataset. All the correlation coefficients showed similar correlation trends, thus in Table 9 we show only the Spearman's rho correlation coefficient. Other correlation coefficients, along with full list of p-values can be found in Table 11, in Appendix A.2.

In Table 9 Green (dark) cells mark the pairs that have medium statistically significant correlation. If the $p-value$ of the coefficient is lower than 0.05 we

emphasize the correlation value with **underlined bold**, if $p - value < 0.1$ - with ***italics-bold***.

Table 9: The values of the Spearman's rank correlation coefficients between parameters of the dataset and the parameters of the sigmoid. Non-sequential case.

Parameter	m	M	r_{alg}	d_{alg}	$\frac{r_{alg}}{d_{alg}}$	c
# of classes x_1	0.02	**-0.26**	**-0.25**	**0.31**	**-0.34**	**-0.34**
# of attr x_2	0.03	**-0.26**	**-0.24**	0.14	**-0.20**	-0.15
# of inst x_3	-0.05	0.03	0.02	**-0.21**	***0.18***	***0.18***
fract dim x_4	-0.03	**-0.20**	***-0.18***	0.06	-0.11	-0.07

Summarizing the results from all the correlation coefficients (refer to Table 9 and 11), some interesting conclusions can be drawn. First, the number of classes (x_1) is inversely correlated to $\frac{r_{alg}}{d_{alg}}$, c, r_{alg} and M. Thus, the higher the number of classes is, the lower the sensitivity to noise variation (check on $\frac{r_{alg}}{d_{alg}}$); the lower the number of classes, the higher the impact of reducing noise on performance (check r_{alg} and M). These conclusions are also supported by the direct correlation between the number of classes and the width of the active area of the algorithm d_{alg}. Thus, the number of classes (x_1) significantly influences the behavior of an algorithm, regardless of the family of the algorithm. We also note the complete lack of significant correlation between the minimum performance m and all of the SRF dimensions: given enough noise an algorithm always performs badly.

Second, the number of features (x_2) provides a minor reduction of sensitivity to noise variation (resulting from low correlation to d_{alg} and c). This conclusion is also supported by the negative influence on $\frac{r_{alg}}{d_{alg}}$, r_{alg}. We also note that the number of features affects the maximal performance M, which shows (rather contrary to the intuition) that more features may negatively affect performance in a noise-free scenario. This is most probably related to features that are not essentially related to the labeling process, thus inducing feature noise. On the other hand, if the number of instances is too low, then it is hard for the classifier to generalize over many features.

Third, there is a correlation between the number of instances (x_3), $\frac{r_{alg}}{d_{alg}}$, and the slope indicator c. This shows that larger datasets (providing more instances) increase the sensitivity to noise variation. Furthermore, in such highly populated datasets, reducing the label noise is expected to have a significant impact on performance.

Last, the fractal dimensionality (x_4) of a dataset has low, but statistically significant, negative influence on M and on r_{alg}. Fractal dimensionality is indicative of the "complexity" of the dataset. Thus, if a dataset is complex (high x_4), machine learning is difficult even at low noise levels. We note that

low r_{alg} may be preferable in cases where the algorithm should be stable even for low signal-to-noise ratios.

Sequential case.

For the sequential case, we study the connection between SRF dimensions and the following dataset characteristics: the number of classes (x_1), the number of instances (x_2), and the maximal autocorrelation lag of a dataset (x_3). For the analysis we use the same data as for the sequential regression analysis in Section 7.1. Therefore, for each SRF dimension we have 6 values corresponding to each classifier, and, after training the classifiers on 18 datasets, we obtained 108 values for a single SRF dimension. Then, we correlate the SRF dimension values with the corresponding parameters of a dataset.

The results are shown in Table 10, where Yellow (light) cells mark the pairs that have strong statistically significant correlation, green (dark) cells mark the pairs that have medium statistically significant correlation. If $p-value < 0.05$ the cell is emphasized with **underlined bold**, if $p-value < 0.1$ – with ***italics-bold***. Similarly to the non-sequential case, we report only the Spearman's rho correlation coefficient here, while all the other coefficients with the corresponding $p-values$ can be found in Appendix A.2 in Table 12.

Table 10: The Spearman's rank correlation coefficients between parameters of the dataset and parameters of the sigmoid. Sequential case.

Parameter	m	M	r_{alg}	d_{alg}	$\frac{r_{alg}}{d_{alg}}$	c
# of classes x_1	***-0.174***	-0.004	0.082	**0.393**	**-0.290**	**-0.393**
# of inst x_2	-0.099	**-0.264**	***0.165***	**-0.483**	**0.409**	**0.483**
autocorr lag x_3	**-0.335**	**0.511**	**0.434**	-0.127	**0.221**	0.127

The results demonstrate that the number of classes x_1 is inversely correlated with the slope indicator c, and positively correlated with the active performance range d_{alg}, as in the non-sequential case. This leads to the following conclusion: the higher the number of classes is, the lower the sensitivity to noise variation is, regardless of the learning algorithm.

The number of instances x_2 is positively correlated with the slope indicators r_{alg}/d_{alg} and c, and is negatively correlated with the active area of the algorithm. Thus, the number of instances has the opposite effect of the number of classes, and in datasets with a large number of instances, reducing the label noise is expected to have a significant impact on performance. The same effect is observed for non-sequential datasets.

The autocorrelation lag x_3 has an influence on the performance range: if the dependency history is deep (high x_3), then the maximal performance grows (M), and the minimal performance drops (m). Thus, the expected accuracy of sequential classification task is higher if the dataset has deeper history, which is an intuitive result. Finally, compared to the correlation analysis of non-

sequential data, sequential data overall shows stronger correlations indicating robust use potential across classifier types.

7.4 Regression and Correlation Analysis: Summary

For both cases of non-sequential and sequential classification there is a strong cumulative and individual effect of dataset characteristics on the SRF dimensions. This allows us in most of the cases to reason about the expected performance of the algorithms given certain dataset characteristics. If the SRF dimensions are estimated for a set of algorithms for datasets with certain characteristic, this allows for recommending an algorithm for a new dataset with similar characteristics. In the case where a specific SRF dimension is of great importance, the dataset characteristics provide an insight on how the dimension will behave, and if it is possible to increase or decrease its value. For example, in the case of a large number of instances in the dataset, the sensitivity to noise variation is increasing, thus, the performance improvement ratio r_{alg}/d_{alg} will also be larger. Therefore, reducing the label noise will have significant impact on performance. This effect is even stronger for sequential classifiers.

8 Discussion

Machine learning algorithms are often used in noisy environments. Therefore, it is important to know a priori the properties of an algorithm depending on the characteristics of the dataset and the noise in a setting. In this work, we investigated whether some simple dataset properties (namely, the number of classes, the number of features, the number of instances, the fractal dimensionality for the non-sequential case, and the maximal autocorrelation lag for the sequential case) can help in the above direction.

We proposed the Sigmoid Rule Framework, describing a set of dimensions that may be used by a decision maker to choose a good classifier for a specific case. Our approach is applicable to user modeling tasks, when the user changes behavior over time, and to any concept drift for data series mining or label noise problems. For example, in the case of two robots that communicate trying to determine the best line of action given a set of sensor readings, the proposed method could be applicable. Let us consider the case where the communication is faulty, allowing labeling noise. In that case, one may want to use the most robust algorithm, i.e. the algorithm that is most unlikely to lead to very poor performance if the noise is high. In this example, and taking into account Figure 8 of the 'Robot Walk 4' data, the designer may select the classifier based on the Conditional Random Fields with second-order dependencies (corresponding to curve (d)), because it has the highest minimum expected performance and is quite robust in the change of noise levels. If the preference in the performance is changed and one would like to

have the algorithm with the fastest performance improvement, the HMM with second order dependencies can be chosen (Figure 8b).

Given such problems, we proposed – in Section 5 – a workflow and method to select a classifier based on user preferences. We also showed that the parameters related to the behavior of learners correlate with the dataset characteristics, and the range of their variation may be predicted using linear or logistic regression models.

Therefore, SRF can be a useful meta-learning framework, specialized in settings that include labeling noise. Similar to several meta-learning methods, SRF uses characteristics that are efficient to compute (number of classes, number of attributes, fractal dimension). However, we expect the overall complexity of applying SRF to be higher than corresponding meta-learning approaches, since SRF requires an initialization phase, where it estimates performance over varying degrees of noise. We should note though, that in concept drift settings a similar burden is put in detecting drift (e.g., [29]) during classification. Using SRF, this burden may be applied once, at the beginning of the training, for the selection of an overall best algorithm over all possible noise settings.

As we discussed in the previous section, the SRF model shows promising results regarding the expected performance of algorithms given a specific dataset, regardless of the underlying algorithm. Nevertheless, these results do not provide enough precision to predict the performance of a specific algorithm. This is an interesting research question, which we are currently pursuing.

Overall, we claim that our approach presents a paradigm shift to how algorithms can be evaluated, and the results presented in this study provide a proof-of-concept for this new paradigm. The focus of our experimental evaluation was to illustrate the value of the proposed method in the specific context of noisy settings. It would be very interesting to further extend the proposed approach, and compare it to well-established alternatives, by studying its applicability to a variety of settings other than the originally intended labeling noise, such as, dataset imbalance, adaptation to new domains, and complexity and effectiveness in non-noisy settings.

As future work, we would like to examine more of the dataset features used in the literature, in order to evaluate how they affect the sigmoid of an algorithm for a given problem, both theoretically and experimentally. From the application point of view we will evaluate whether meta-learning frameworks after using the SRF dimensions to enrich their view of the algorithms, improve their performance and increase the number of cases where they are efficient. We expect that the features generated in our framework, namely SRF dimensions, can provide added value to existing meta-learning frameworks by enriching the representation of algorithms and by embedding information of expected performance in different noisy settings. Our study may provide insight on the expected impact of noise removal (e.g. instance filtering) on the average performance of a given learning algorithm.

9 Conclusions

In this work we studied two, dual problems: comparing algorithm behavior, and choosing learning algorithms for noisy settings.

We presented the "Sigmoid Rule" Framework (SRF), which uses a model of the expected performance of learning algorithms, that is a sigmoid function of the signal-to-noise ratio in the training instances. We studied the parameters of the sigmoid function using five representative non-sequential classifiers, and three widely used sequential classifiers. Based on the sigmoid parameters we defined a set of intuitive criteria that are useful for comparing the behavior of learning algorithms in the presence of noise. The framework is applicable to concept drift scenarios, including modeling user behavior over time, and mining of noisy time series of evolving nature.

The experimental evaluation showed that there is a connection between the SRF parameters and the characteristics of the underlying dataset, indicating that we can estimate the expected performance over a dataset regardless of the underlying algorithm.

As a summary, the main contributions of our work are the following:

1. A unified view of the performance of learning algorithms in a training label noise setting.
2. A framework for estimating an analytic function that connects the expected performance to varying noise levels.
3. The extension of the above framework for sequential classifiers.
4. The study of the framework parameters as decision factors for algorithm selection in specific settings.
5. The illustration of a robust connection between specific dataset characteristics and the expected classifier performance.

As future work, we are going to verify the SRF framework for large groups of datasets and classifiers, validate the prediction of SRF dimensions given the characteristics of the datasets, build a recommender of the optimal classifier for a given set of user requirements about the classifier behavior in noisy settings and validate it on real tasks.

Acknowledgements We thank the editor and the reviewers of this paper for their valuable comments and advice. K. Mirylenka also thanks Daniil Mirylenka for proofreading the paper.

References

1. Abdulrahman, S.M., Brazdil, P., van Rijn, J.N., Vanschoren, J.(2015) Algorithm selection via meta-learning and sample-based active testing. Proceedings of the 2015 International Workshop on Meta-Learning and Algorithm Selection (MetaSel) co-located with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD), 55–66
2. Akaike, H. (1974) A new look at the statistical model identification. IEEE Transactions on Automatic Control, 19(6), 716–723. DOI 10.1109/TAC.1974.1100705

3. Ali, S., Smith, K. (2006) On learning algorithm selection for classification. Applied Soft Computing 6(2), 119–138
4. Bodén, M. (2002) A guide to recurrent neural networks and backpropagation. The Dallas project, SICS technical report (2), 1–10. URL http://130.102.79.1/~mikael/papers/rn_dallas.pdf
5. Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015) Time series analysis: forecasting and control. John Wiley & Sons
6. Bradley, A. (1997) The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7), 1145–1159
7. Brazdil, P. B., Soares, C., Pinto Da Costa, J. (2003) Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. Machine Learning 50(3), 251–277.
8. Brazdil, P., Giraud Carrier, C., Soares, C., Vilalta, R. (2009) Development of metalearning systems for algorithm recommendation. Metalearning: Applications to Data Mining, 31–59
9. Camastra, F., Vinciarelli, A. (2002) Estimating the intrinsic dimension of data with a fractal-based method. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(10), 1404–1407
10. Chevaleyre, Y., Zucker, J.D. (2000) Noise-tolerant rule induction from multi-instance data. In Proceedings of the workshop on Attribute-Value and Relational Learning: Crossing the Boundaries, co-located with International Conference on Machine Learning (ICML), 47–52
11. Cohen, W.W. (1995) Fast effective rule induction. In Proceedings of the Twelfth International Conference on Machine Learning, 115–123
12. Corder, G.W., Foreman, D.I. (2009) Nonparametric statistics for non-statisticians: a step-by-step approach. John Wiley & Sons
13. Cruz, R.M., Sabourin, R., Cavalcanti, G.D., Ren, T.I.: Meta-des (2015) A dynamic ensemble selection framework using meta-learning. Pattern Recognition 48(5), 1925–1935
14. Dupont, P. (2006) Noisy sequence classification with smoothed Markov chains. In: Proceedings of the 8th French Conference on Machine Learning (CAP 2006), 187–201
15. Elman, J. (1990) Finding structure in time. Cognitive science **211**, 179–211.
16. Eom, S.B., Ketcherside, M.A., Lee, H.H., Rodgers, M.L., Starrett, D. (2004) The determinants of web-based instructional systems' outcome and satisfaction: An empirical investigation. Instructional technologies: Cognitive aspects of online programs, 96–139
17. Lichman, M. (2013) UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml. University of California, Irvine, School of Information and Computer Sciences
18. Garcia, L. PF, de Carvalho A. CPLF, Lorena A. C. (2016) Noise detection in the meta-learning level. Neurocomputing 176, 14–25
19. Giannakopoulos, G., Palpanas, T. (2010) The effect of history on modeling systems' performance: The problem of the demanding lord. IEEE 10th International Conference on Data Mining (ICDM), doi: 10.1109/ICDM.2010.90
20. Giannakopoulos, G., Palpanas, T. (2013) Revisiting the effect of history on learning performance: the problem of the demanding lord. Knowledge and Information Systems, 36(3), 653–691. doi: 10.1007/s10115-012-0568-8
21. Giraud-Carrier, C., Vilalta, R., Brazdil, P. (2004) Introduction to the special issue on meta-learning. Machine Learning 54(3), 187–193
22. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. (2009) The weka data mining software: an update. ACM SIGKDD Explorations Newsletter 11(1), 10–18
23. Han, J., Kamber, M. (2006) Data mining: concepts and techniques. Morgan Kaufmann
24. Haussler, D. (1990) Probably approximately correct learning. University of California, Santa Cruz, Computer Research Laboratory
25. Heywood, M.I. (2015) Evolutionary model building under streaming data for classification tasks: opportunities and challenges. Genetic Programming and Evolvable Machines 16(3), 283–326
26. Jahmm: a java implementation of hidden Markov model. URL https://code.google.com/p/jahmm/

27. Kalapanidas, E., Avouris, N., Craciun, M., Neagu, D. (2003) Machine learning algorithms: a study on noise sensitivity. In Proc. 1st Balcan Conference in Informatics, 356–365
28. Keerthi, S., Shevade, S., Bhattacharyya, C., Murthy, K. (2001) Improvements to platt's SMO algorithm for SVM classifier design. Neural Computation 13(3), 637–649
29. Klinkenberg, R. (2005) Meta-learning, model selection, and example selection in machine learning domains with concept drift. In: Lernen, Wissensentdeckung und Adaptivität (LWA) 2005, GI Workshops, Saarbrücken, October 10th-12th, 2005, 164–171
30. Kuh, A., Petsche, T., Rivest, R.L. (1990) Learning time-varying concepts. Conference on Neural Information Processing Systems (NIPS), 183–189
31. Kuhn, M., Johnson, K. (2013) Applied Predictive Modeling. Springer-Verlag New York doi:10.1007/978-1-4614-6849-3
32. Lafferty, J., McCallum, A., Pereira, F. (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. Proceedings of the Eighteenth International Conference on Machine Learning (ICML), 282–289
33. won Lee, J., Giraud-Carrier, C. (2008) New insights into learning algorithms and datasets. IEEE Seventh International Conference on Machine Learning and Applications (ICMLA'08), 135–140
34. Li, Q., Li, T., Zhu, S., Kambhamettu, C. (2002) Improving medical/biological data classification performance by wavelet preprocessing. Proceedings of the IEEE International Conference on Data Mining (ICDM), 657–660
35. Marsaglia, G., Tsang, W.W., Wang, J. (2003) Evaluating Kolmogorovś distribution. Journal of Statistical Software 8(1), 1–4. DOI 10.18637/jss.v008.i18
36. Massey Jr, F.J. (1951) The Kolmogorov-Smirnov test for goodness of fit. Journal of the American statistical Association 46(253), 68–78
37. McCallum, A.K. (2002) Mallet: A machine learning for language toolkit. Http://mallet.cs.umass.edu
38. Mirylenka, K., Cormode, G., Palpanas, T., Srivastava, D. (2015) Conditional heavy hitters: Detecting interesting correlations in data streams. The International Journal on Very Large Data Bases (VLDB) 24(3), 395–414
39. Mirylenka, K., Giannakopoulos, G., Palpanas, T. (2012) SRF: A framework for the study of classifier behavior under training set mislabeling noise. In: Advances in Knowledge Discovery and Data Mining, *Lecture Notes in Computer Science*, vol. 7301, 109–121
40. Mirylenka, K., Palpanas, T., Cormode, G., Srivastava, D. (2013) Finding interesting correlations with conditional heavy hitters. IEEE 29th International Conference on Data Engineering (ICDE), 1069–1080
41. Mantovani, R.G., Rossi, A. L. D., Vanschoren, J., Carvalho, A.C.P.L.F. (2015) Meta-learning Recommendation of Default Hyper-parameter Values for SVMs in Classification Tasks. Proceedings of the 2015 International Workshop on Meta-Learning and Algorithm Selection (MetaSel), European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD), 80–92
42. Nettleton, D.F., Orriols-Puig, A., Fornells, A. (2010) A study of the effect of different types of noise on the precision of supervised learning techniques. Artificial Intelligence Review 33(4), 275–306 DOI 10.1007/s10462-010-9156-z.
43. Pechenizkiy, M. (2015) Predictive analytics on evolving data streams anticipating and adapting to changes in known and unknown contexts. In IEEE International Conference on High Performance Computing & Simulation (HPCS), 658–659
44. Pendrith, M., Sammut, C. (1994) On reinforcement learning of control actions in noisy and non-markovian domains. Tech. rep., School of Computer Science and Engineering, the University of New South Wales, Sydney, Australia
45. Rabiner, L., Juang, B. (1986) An introduction to hidden Markov models. IEEE ASSP Magazine 3(1), 4–16
46. Rossi, A. L. D., Ponce de Leon Ferreira de Carvalho, A. C., Soares, C., Feres de Souza, B. (2014) MetaStream: a meta-learning based method for periodic algorithm selection in time-changing data. Neurocomputing 127: 52–64.
47. Smith, M. R., Mitchell, L., Giraud-Carrier, C., & Martinez, T. (2014) Recommending learning algorithms and their associated hyperparameters. arXiv preprint arXiv:1407.1890.

48. de Sousa, E., Traina, A., Traina Jr, C., Faloutsos, C. (2006) Evaluating the intrinsic dimension of evolving data streams. In: Proceedings of the 2006 ACM symposium on Applied computing, 643–648
49. Sutton, C., McCallum, A. (2010) An introduction to conditional random fields. arXiv preprint arXiv:1011.4088
50. Taylor, R. (1990) Interpretation of the correlation coefficient: a basic review. Journal of diagnostic medical sonography 6(1), 35–39
51. Teytaud, O. (2001) Learning with noise. Extension to regression. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'01) , vol. 3, 1787–1792
52. Theodoridis, S., Koutroumbas, K. (2003) Pattern Recognition. Academic Press
53. Valiant, L. (1984) A theory of the learnable. Communications of the ACM 27(11), 1134–1142
54. Vapnik, V.N., Vapnik, V. (1998) Statistical learning theory, vol. 1. Wiley New York
55. Waluyan, L., Sasipan, S., Noguera, S., Asai, T. (2009) Analysis of potential problems in people management concerning information security in cross-cultural environment -in the case of malaysia-. In Proceedings of the Third International Symposium on Human Aspects of Information Security & Assurance (HAISA), 13–24
56. Widmer, G. (1997) Tracking context changes through meta-learning. Machine Learning 27(3), 259–286
57. Wolpert, D. (1996) The existence of a priori distinctions between learning algorithms. Neural Computation 8, 1391–1421
58. Wolpert, D. (2001) The supervised learning no-free-lunch theorems. In: Proc. 6th Online World Conference on Soft Computing in Industrial Applications. 25–42. Springer London
59. Wolpert, D.H. (1996) The lack of a priori distinctions between learning algorithms. Neural Computation 8, 1341–1390
60. Xing, Z., Pei, J., Keogh, E. (2010) A brief survey on sequence classification. ACM SIGKDD Explorations Newsletter 12(1), 40. doi: 10.1145/1882471.1882478

A Appendices

A.1 The Genetic Algorithm Settings

We used the JGAP[1] genetic algorithms package for the search in the sigmoid parameter space and Java Statistical Classes library for the statistical measurement the Kolmogorov-Smirnov (KS) test[2]. Default operators for double numbers were used. The alleles were five parameters, one per parameter:

- m with allowed values in $[0.0, 0.5]$.
- M with allowed values in $[0.5, 1.0]$.
- b with allowed values in $[0.0, 50.0]$.
- c with allowed values in $[0.0, 50.0]$.
- d with allowed values in $[-5.0, 5.0]$.

Essentially, employing the genetic algorithm, we try to maximize the following quantity:

$$fitness(i) = 100 * \left(1 + \frac{1}{D+1}\right),$$

where i is a candidate individual in the genetic algorithm, corresponding to a given set of parameter values and $fitness(i)$ the value of the fitness function for that individual. The parameter D is the D statistic of the KS test [35], which is higher when the fit is worse. The population per iteration is 10000 individuals. Our search ends when there is no significant (that is $> 10^{-5}$) improvement after 20 consecutive iterations of the genetic algorithms, or when 1000 iterations have been completed.

[1] See http://jgap.sourceforge.net/.

[2] See http://www.jsc.nildram.co.uk/.

A.2 Correlation Results

Non-sequential case. All the correlation coefficients with the corresponding $p-values$ are shown in Table 11. Green (dark) cells mark the pairs that have medium correlation. $P-values$ of importance are emphasized as follows: ($p-value < 0.05$ **underlined bold** cells, $p-value < 0.1$ ***italics-bold*** cells).

Table 11: The values of three correlation coefficients between parameters of the dataset and the parameters of the sigmoid. Non-sequential case.

Pearson's correlation coefficient							
Parameter		m	M	r_{alg}	d_{alg}	$\frac{r_{alg}}{d_{alg}}$	c
# of classes x_1	$coef$	-0.03	-0.26	-0.21	0.13	-0.29	-0.28
	p_{value}	0.736	**0.005**	**0.025**	0.179	**0.001**	**0.002**
# of attr x_2	$coef$	-0.07	-0.31	- 0.23	0.13	-0.21	-0.18
	p_{value}	0.463	**0.001**	**0.012**	0.173	**0.025**	**0.049**
# of inst x_3	$coef$	0.14	0.08	-0.01	-0.12	0.12	0.16
	p_{value}	0.130	0.402	0.950	0.209	0.193	***0.093***
fract dim x_4	$coef$	0.04	-0.16	-0.16	0.13	-0.09	-0.06
	p_{value}	0.690	***0.086***	***0.092***	0.168	0.364	0.549
Spearman's rank correlation coefficient							
Parameter		m	M	r_{alg}	d_{alg}	$\frac{r_{alg}}{d_{alg}}$	c
# of classes x_1	$coef$	0.02	-0.26	-0.25	0.31	-0.34	-0.34
	p_{value}	0.810	**0.005**	**0.008**	**0.001**	**0.000**	**0.000**
# of attr x_2	$coef$	0.03	-0.26	-0.24	0.14	-0.20	-0.15
	p_{value}	0.718	**0.006**	**0.011**	0.124	**0.030**	0.110
# of inst x_3	$coef$	-0.05	0.03	0.02	-0.21	0.18	0.18
	p_{value}	0.579	0.752	0.863	**0.024**	***0.053***	***0.054***
fract dim x_4	$coef$	-0.03	-0.20	-0.18	0.06	-0.11	-0.07
	p_{value}	0.712	**0.036**	***0.058***	0.559	0.241	0.426
Kendall's τ rank correlation coefficient							
Parameter		m	M	r_{alg}	d_{alg}	$\frac{r_{alg}}{d_{alg}}$	c
# of classes x_1	$coef$	0.01	-0.17	-0.18	0.22	-0.24	-0.24
	p_{value}	0.909	**0.009**	**0.007**	**0.001**	**0.000**	**0.000**
# of attr x_2	$coef$	0.02	-0.18	-0.16	0.10	-0.14	-0.11
	p_{value}	0.715	**0.007**	**0.013**	0.111	**0.032**	***0.094***
# of inst x_3	$coef$	-0.05	0.01	0.01	-0.14	0.12	0.12
	p_{value}	0.400	0.818	0.896	**0.032**	***0.063***	***0.071***
fract dim x_4	$coef$	-0.03	- 0.12	-0.11	0.04	-0.07	-0.06
	p_{value}	0.681	***0.062***	***0.081***	0.507	0.267	0.378

Sequential case. All the correlation coefficients with the corresponding $p-values$ are shown in Table 12, where Yellow (light) cells mark the pairs that have strong statistically significant correlation, green (dark) cells mark the pairs that have medium statistically significant correlation. If $p-value < 0.05$ the cell is emphasized with **underlined bold**, if $p-value < 0.1$ – with ***italics-bold***.

Table 12: Three correlation coefficients between parameters of the dataset and parameters of the sigmoid. Sequential case.

Pearson's correlation coefficient							
Parameter		m	M	r_{alg}	d_{alg}	$\frac{r_{alg}}{d_{alg}}$	c
# of classes x_1	$coef$	-0.117	0.063	0.113	0.238	-0.220	-0.265
	p_{value}	0.228	0.520	0.246	**0.013**	**0.022**	**0.006**
# of inst x_2	$coef$	0.019	0.191	0.088	-0.446	0.523	-0.526
	p_{value}	0.842	**0.048**	0.362	**0.000**	**0.000**	**0.000**
autocorr lag x_3	$coef$	-0.199	0.421	0.359	-0.117	0.226	0.125
	p_{value}	**0.039**	**0.000**	**0.000**	0.227	**0.018**	0.196
Spearman's rank correlation coefficient							
Parameter		m	M	r_{alg}	d_{alg}	$\frac{r_{alg}}{d_{alg}}$	c
# of classes x_1	$coef$	-0.174	-0.004	0.082	0.393	-0.290	-0.393
	p_{value}	***0.072***	0.964	0.396	**0.000**	**0.002**	**0.000**
# of inst x_2	$coef$	-0.099	-0.264	0.165	-0.483	0.409	0.483
	p_{value}	0.309	**0.006**	***0.088***	**0.000**	**0.000**	**0.000**
autocorr lag x_3	$coef$	-0.335	0.511	0.434	-0.127	0.221	0.127
	p_{value}	**0.000**	**0.000**	**0.000**	0.190	**0.021**	0.190
Kendall's τ rank correlation coefficient							
Parameter		m	M	r_{alg}	d_{alg}	$\frac{r_{alg}}{d_{alg}}$	c
# of classes x_1	$coef$	-0.124	0.001	0.067	0.302	-0.219	-0.302
	p_{value}	***0.080***	0.991	0.347	**0.000**	**0.002**	**0.000**
# of inst x_2	$coef$	-0.068	0.176	-0.16	-0.328	0.280	0.328
	p_{value}	0.306	**0.008**	***0.081***	**0.000**	**0.000**	**0.000**
autocorr lag x_3	$coef$	-0.234	-0.354	0.295	-0.089	0.160	0.089
	p_{value}	**0.000**	**0.000**	**0.000**	0.191	**0.019**	0.191