Runtime Evaluation of Cognitive Systems for Non-Deterministic Multiple Output Classification Problems

Aravind Kota Gopalakrishna[a,*], Tanir Ozcelebi[a], Johan J. Lukkien[a], Antonio Liotta[b]

[a]*System Architecture and Networking Group, Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands*
[b]*Data Science Research Centre, University of Derby, UK*

Abstract

Cognitive applications that involve complex decision making such as smart lighting have non-deterministic input-output relationships, i.e., more than one output may be acceptable for a given input. We refer them as *non-deterministic multiple output classification (nDMOC)* problems, which are particularly difficult for machine learning (ML) algorithms to predict outcomes accurately. Evaluating ML algorithms based on commonly used metrics such as Classification Accuracy (CA) is not appropriate. In a batch setting, Relevance Score (RS) was proposed as a better alternative, which determines how relevant a predicted output is to a given context. We introduce two variants of RS to evaluate ML algorithms in an online setting. Furthermore, we evaluate the algorithms using different metrics for two datasets that have non-deterministic input-output relationships. We show that instance-based learning provides superior RS performance and the RS performance keeps improving with an increase in the number of observed samples, even after the CA performance has converged to its maximum. This is a crucial result as it illustrates that RS is able to capture the performance of ML algorithms in the context of nDMOC problems while

*Corresponding Author
Email addresses: `aravindkg7@gmail.com` (Aravind Kota Gopalakrishna), `t.ozcelebi@tue.nl` (Tanir Ozcelebi), `j.j.lukkien@tue.nl` (Johan J. Lukkien), `a.liotta@derby.ac.uk` (Antonio Liotta)

CA cannot.

Keywords: Cognitive Systems, Machine learning, Classification problems, Performance metric, Relevance score, Non-deterministic multiple output classification, Human factors, Smart lighting
2010 MSC: 00-01, 99-00

1. Introduction

Consider a cognitive system such as *smart lighting* [1] that employs machine learning (ML) to automatically select light presets that are desirable in different observed contexts. Here, the observed context can be expressed using features such as the user count, user preferences learned from history, time of the day and more. Assume that a user, Tom enters his living room and upon detecting his context the smart lighting system predicts a lighting preset L_A and actuates it. L_A is known to have satisfied Tom at a previous time for the same context. Therefore, it is rather a safe prediction. However, Tom at this time prefers a different lighting preset L_B and as a result goes to the control interface to switch to this preset. This kind of inconsistent behavior is rather common in applications that involve humans, whereby cognitive factors such as perception play an important role. As these are very difficult to measure, the observed context in practice does not include non-deterministic factors such as psychophysics (e.g. the human audio-visual system has limited accuracy) [2]; psychology (e.g. biases and mood affect perception of reality) [3, 4]; cost and lifestyle (e.g. expectations grow substantially with economic implications) [5]; and all sorts of other cognitive elements [6, 7].

In the literature, classification problems are broadly categorized into *multi-class classification* and *multi-label classification* problems. Multi-class classification problems [8] are those categories of problems where, for a given input instance, there is exactly one correct output class. Examples of multi-class classification include image recognition and patient diagnosis [9]. Multi-label classification problems [10] constitute the category of problems where more than one output classes are selected for each input instance. Examples of multi-label classification include text [11] and music categorization [12]. There exists another class of applications, as illustrated by the smart lighting example above [13], where the aim is to predict the most suitable lighting preset (among several suitable presets) based on an *observed context*. These applications pose what we call a *non-deterministic multiple output classification (nDMOC)* problem.

An interesting property of such an application is that; i) the relationship between input and output is not deterministic and varies over time (due to human inconsistency, bias drift, or other high-level cognitive factors), which means that there may be more than one acceptable output that is satisfactory for a given context, ii) the most desirable output may change unpredictably over time [14]. Precisely, statistical regularity is difficult to observe in such applications. By this we mean that the underlying probability distribution from which the outputs are drawn is not fixed over a period of time. These types of problems arise when, i) it is not possible to identify all the features that are relevant to determine the output or ii) it is very challenging to model some features such as user mood and perception. In such cases, while the ML algorithms that are used in multi-class classification problems suit the need, their performances vary.

We must select an appropriate metric to assess the performance of the ML algorithms, as some performance metrics would produce results that are meaningless in nDMOC applications [15, 16]. Note that different metrics are indicators for different performance aspects of ML algorithms [17]. For instance, in the smart lighting case, the commonly used metrics such as classification accuracy (CA) can only measure whether the predicted output matches Tom's current preference. However, these metrics such as CA, precision and recall fail to capture whether the output is consistent with Tom's past behavior and whether or not it is relevant for the current context. CA gives a score of zero for an inaccurate prediction, which is not suitable for nDMOC problems. In our earlier work [18], we proposed a new evaluation metric, dubbed *Relevance Score (RS)* that suitably penalizes the performance of ML algorithms, when the predicted outputs show inconsistencies compared with the earlier choices of the user.

RS is the degree of relevance of a predicted output for a given context with respect to the actual output [18, 19]. When there is a mismatch between predicted and actual outputs, RS will account for the level of inconsistency of non-deterministic multiple-output cases. Thus, for instance, if a user is consistently inconsistent with his choices, any predicted output will be highly relevant (high RS), although CA will be low. The RS metric will determine the relevance of a prediction based on the number of times these outputs have been selected before in a similar context. Therefore, the RS metric does not deal with the learning aspects of existing prediction algorithms. Instead, RS allows evaluating predictions from a different perspective where the CA metric fails.

In our earlier work [18, 19], we have studied the performance of ML algorithms in a batch learning setting, where the ML algorithms are trained on a fixed set of samples. Subsequently, the concept learned was used to predict outputs on newly observed contexts. We found that ML algorithms in the batch setting are not appropriate in the context of nDMOC problems, as the underlying input-output relationships change over time. This motivated us to explore online ML frameworks for nDMOC problems. In an online setting, the ML algorithm learns from one sample at a time. Here, the input arrives as a sequence of individual samples, in contrast to the batch learning [20]. Upon receiving an input, the algorithm predicts the output. The actual output is then shown to the algorithm, which is used as a feedback to improve future predictions.

In our previous work [21], we studied the performance of ML algorithms in an online setting, particularly in the context of smart lighting. We created a smart lighting dataset, based on a user-centric environment, whereby our subjects' interaction with the system were traced, recording also contextual information such as user identity, type of activity, influence of external light, time of the day and more [22]. The dataset contains 236 samples where six contextual features are mapped to one of the eight preset lighting conditions [1]. What came out of the study is that ML algorithms failed to make accurate predictions but were more effective in predicting relevant outputs.

The main contribution of this paper is that we introduce two variants of the RS metric. This allows us to investigate the behavior of ML algorithms in different scenarios, where the data are derived either from a single distribution or multiple distributions. We use a much larger dataset than our previous work to further investigate the behavior of ML algorithms, particularly when the input-output relationship has various degrees of randomness. We consider the well-known *Wine* dataset [23] for experiments. Even though there are other candidate nDMOC datasets such as Abalone and Ecoli [24], we selected the Wine dataset for the following reasons: i) it contains more samples i.e. around 5000; ii) it contains more output classes; and iii) human factors are involved in selecting outputs, where human experts make decision on the quality of wine based on their senses. The experiments are performed using several ML algorithms in an online setting with the CA and RS metrics. The results show that instance-based learning is the most suitable approach in the context of nDMOC problems. The RS metric captures the non-deterministic relationship that exists between input and output more effectively than the CA metric, and accounts for the randomness that is intrinsic in human-perception. Also, we find that the learning rate of the ML algorithms vary significantly as measured using RS, based on the degree of randomness present in the data.

The paper is organized as follows. Section 2 gives an introduction to the existing ML frameworks and discusses the drawbacks of conventional ML performance metrics in the context of nDMOC problems. Section 3 proposes how to adapt the RS metric for evaluating ML algorithms in an *online* setting at runtime. Section 4 describes the experimental setting and presents results that demonstrate the significance of the online RS metric. Section 5 discusses our findings and Section 6 draws conclusions.

2. Existing Machine Learning Frameworks

Generally, classification problems that rely on labeled data (data with output labels) are addressed using ML algorithms, either in batch learning mode or online learning mode.

2.1. Batch Learning

In batch learning, ML algorithms are trained using a set of labeled samples. These algorithms then predict for the unseen contexts based on the knowledge gained from the training samples. These constitute a *parametric* class of algorithms, meaning that the data is assumed to be derived from a known distribution [25], which can be summarized using a finite number of parameters. This means that the observed pattern in the dataset remains the same over a long period of time. In nDMOC problems, dynamic factors such as changing user preferences, mood and perception may lead to inconsistent output selections. A fixed data distribution cannot be assumed because the observed pattern in the training samples changes over time.

In batch learning, ML algorithms do not adapt when the input-output relationships change. This is because they are trained on a fixed set of labeled samples. Subsequently, the concept learned is used to predict outputs on newly observed context. When we studied the performance of ML algorithms in a batch learning setting [1] we found that they are not appropriate in the context of nDMOC problems. This has motivated us to study the ML algorithms in an online setting.

2.2. Online Learning

These algorithms fall under the category of non-parametric models, which do not assume any fixed distribution [26] and hence can adapt continuously to unknown or changing distributions. This makes online learning algorithms preferable when the distribution from which the samples are drawn is unknown.

In contrast to batch learning, in online setting, the input arrives as a sequence of samples [20]. As a new input arrives, the ML algorithm should predict an output. Immediately after the prediction is made, the actual output is made available to the ML algorithm. This information is used as a feedback to update the prediction hypothesis used by the algorithm. A key advantage of online learning over batch learning is that the algorithm adapts itself with every new observed context.

In formal terms, at each step i, the algorithm is given a context $x_i \in X$ where $X = X_1 \times X_2 \times X_3 \times \ldots \times X_n$ is an n-dimensional input space and $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$. The algorithm predicts an output class label $y_P \in Y = \{y^{(1)}, y^{(2)}, \ldots, y^{(L)}\}$ as a function of the context x_i and weights w_i. Each weight w_i is an n-dimensional real-valued vector $(w_{i1}, w_{i2}, \ldots, w_{in})$ that determines the contribution of x_i. These weights depend on the observed contexts and their corresponding predicted outputs. After y_P is predicted, the algorithm receives the true output class label $y_A \in Y$. The quality of the y_P is then assessed by a loss function $l(y_P, y_A)$. Based on the result of $l(y_P, y_A)$, the algorithm modifies w_i based on an update function $\Delta = f(w_i, l(y_P, y_A))$ to predict better outputs for future contexts.

In this paper, we have considered two online learning algorithms for our experiments that are implemented in *LIBOL* (A Library for Online Algorithms) [27] i.e. *Adaptive Regularization of Weights* (AROW) [28], and *Soft-Confidence Weighted* (SCW) algorithm [29]. We use the default parameters implemented in LIBOL. AROW and SCW are second-order algorithms. In multi-class classification problems, online learning algorithms maintain classification functions for each output class label. First-order learning algorithms keep updating these functions using only the information from the input features. However, second-order online algorithms make use of underlying patterns in the data in addition to relationships among features.

Adaptive Regularization Of Weights. Crammer et al. introduced AROW to address the need for fast convergence and resistance to over fitting due to noisy samples (samples with false output labels) [28]. The algorithm has several combined properties: large margin training, confidence weighting and the capacity to handle non-separable data. AROW performs adaptive regularization of the prediction function, when processing each new context in each learning step. This makes it more robust to sudden changes in the prediction function due to noisy labels in the learning tasks. Noisy labels refer to a setting where the output class labels are corrupted [30]. The main idea is to retain the formalization of parameter confidence introduced by the Confidence-Weighted (CW) learning method [31], which is aggressive in learning. AROW removes the aggressiveness by softening the margin requirement thus achieving robustness to training noise without affecting the convergence speed.

Soft Confidence Weighted. Wang et al., proposed SCW as an improvement over CW and AROW [29]. Even though the aggressive updating strategy of CW learning results in faster learning, it could wrongly change the parameters of the distribution while dealing with a noisy context, leading to erroneous learning. This can make the CW algorithm perform poorly in many real-world applications with relatively large noise. The SCW learning algorithm was proposed to overcome these limitations by relaxing the update strategy. Like the AROW algorithm, SCW has the following three properties: large margin training, confidence weighting, and capability to handle non-separable data. Additionally, SCW also employs adaptive margins.

2.3. Instance-based Learning

In this form of learning, the input-output relationship is not deduced when the training samples are provided. It is deduced when a new input whose output needs to be predicted arrives. In other words, instance based learning algorithms do not generate any useful representations from the observed samples [32]. To assign a class label for a new instance, its relationship to the samples that were already encountered is determined. The main advantage of this kind of learning is that the target function is estimated for each new instance to be predicted rather than estimating once [33]. This makes instance-based learning useful when the input-output relationship changes over time.

k-Nearest Neighbor (kNN) is the most popular instance-based algorithm as it is simple and efficient. It can be used either as a batch learner or as an online learner. In online learning mode, it works as follows. The samples are in the form of input-output pairs. When a new context or input arrives, for which the output class label needs to be determined, the new context is compared to every context in the training set (set of already observed samples). The most similar contexts (nearest neighbors) with their output class labels are taken. The integer value k, determines the number of nearest neighbors to consider. Finally, from the k nearest neighbors, the output class label that occurs the most (majority voting) is assigned as the output class label for the new context.

Formalizing the kNN algorithm. Let $X = X_1 \times X_2 \times X_3 \times \ldots \times X_n$ denote an n-dimensional input feature space where the i^{th} sample $x_i \in X$ is given by $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$. Let y_i denote the output class label where $y_i \in Y = \{y^{(1)}, y^{(2)}, \ldots, y^{(L)}\}$ and L is the number of possible outcomes. Let m denote the number of samples in the training set i.e., $i = 1, 2, \ldots, m$. The goal is to determine the output class label for a new context x_q. The set of k nearest neighbors from the training set to x_q is determined by computing the distances from each training sample (d_i where $i = 1, 2, \ldots, m$). One of the following distance functions are commonly used to compute the distance between two contexts [34]; i) Manhattan ($d_1(x_i, x_q) = \sum_{j=1}^{n} |x_{ij} - x_{qj}|$), ii) Euclidean ($d_2(x_i, x_q) = \sqrt{\sum_{j=1}^{n} (x_{ij} - x_{qj})^2}$), or iii) Minkowski ($d_z(x_i, x_q) = \sqrt[z]{\sum_{j=1}^{n} (x_{ij} - x_{qj})^z}$) where $z > 2$. The output class label y_q is assigned as follows; $y_q = \underset{y_i}{\operatorname{argmax}}(\#(y_i))$ for $i = 1, 2, \ldots, k$.

For our experiments, we use Manhattan distance to compute the distance between two contexts, as we find that it provides better CA performance than the Euclidean distance on the smart lighting dataset [1]. It is straightforward to compute the Manhattan distance for contexts with numerical features. However, for categorical features, we assign a value of 1, if the compared features are not equal; otherwise it is 0. If we have different scales for the features of numerical data type, then the distance computed between two contexts will be a function of features with larger values, thereby ignoring the role of features with smaller values. Therefore, we normalize the distance computed between two numerical values of a feature as follows. The *normalized* distance for a feature is the ratio of the *absolute* distance to the range for that feature. The absolute distance is the actual difference between the numerical values of a feature for two different contexts. In this paper, we use the following kNN methods.

Conventional k-Nearest Neighbor (in the remainder referred to as k_{Conv}). This algorithm is same as kNN. In an online setting, we do not have any fixed training samples and the algorithm should predict from the first context observed. Therefore, for practical purposes we do not have a fixed k value and the value of k is updated as the number of observed samples increases. We compute the value of k as the square root of the number of contexts observed [35], i.e. at step i, if an output needs to be predicted for context x_i, then $k = \sqrt{i}$.

Distance weighted kNN (in the remainder referred to as Dwk). In $\mathrm{k_{Conv}}$, equal weights are given to the distance computed from a new context x_q to k nearest neighbors. However, the output class labels to be assigned to the test context x_q may be influenced by the distances of its nearest neighbors, i.e. the closest neighbors may contribute more than the farther ones. This has motivated the need for applying weights to the computed distances between the contexts. For our experiments, the weights on the distances are assigned as follows [36]. If the distance of k nearest neighbors to a new context are ordered from closest to farthest as d_j where $j = 1, 2, \ldots, k$, the weight w_j attributed to the j-th nearest neighbor is defined by

$$w_j = \begin{cases} \dfrac{d_k - d_j}{d_k - d_1}, & d_k \neq d_1, \\ 1, & d_k = d_1. \end{cases}$$

Once the weights w_j are computed, the algorithm assigns the new context an output class label, for which the weights among the k-nearest neighbors sum to the largest value. As in the case of k_{Conv}, the Dwk algorithm is used with $k = \sqrt{i}$.

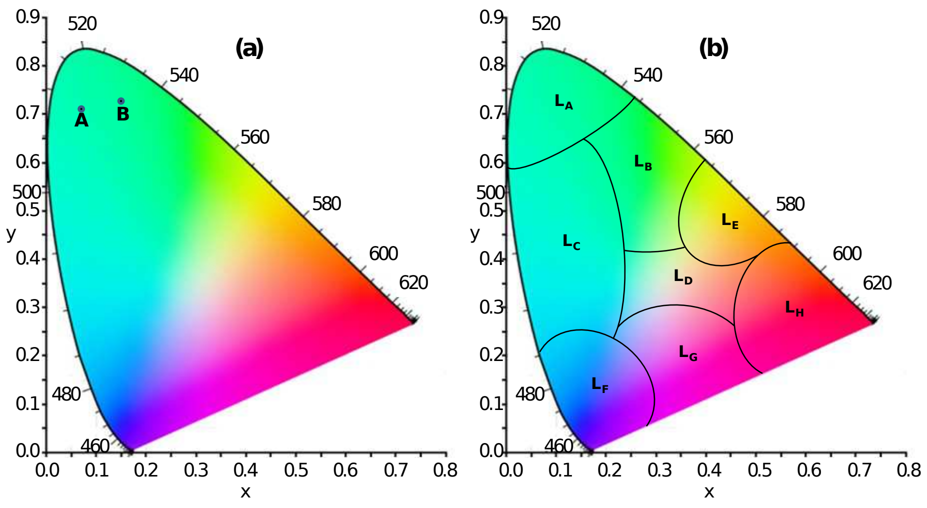

Figure 1: CIE (International Commission on Illumination) 1931 color space

a. Spatially distinct points A and B on the color space that is perceived to be same

b. An example of categories of light conditions where two points inside a category is assumed to be same

2.4. Drawbacks of Conventional Performance Metrics

The commonly used performance evaluation metrics for multi-class classification problems are CA, *precision* and *recall*. For a set of samples, CA is the proportion of predicted cases that are accurately predicted. Precision is the proportion of predicted positive cases that are true positives. Conversely, recall is the proportion of true positive cases that are correctly predicted as positive [37]. The commonly used performance evaluation metrics for multi-label classification problems are *hamming-loss*, *precision* and *recall* [38]. Hamming-loss is the fraction of the wrong labels to the total number of labels. Generally, the performance of a learning algorithm in an online setting is evaluated along its run on a sequence of question-answer pairs [39]. For the classification problems, CA is the most commonly used performance metric. However, CA is not an appropriate metric to evaluate the ML algorithms in the context of nDMOC problems.

To better understand the drawbacks of CA, consider a smart lighting system as exemplified in the introduction, where the goal is to predict and provide a desired lighting condition to users for every observed context. Figure 1 shows the CIE 1931 color space from which the smart lighting system can create an ambient lighting condition. Current generation of lighting systems can provide millions of saturated colors [40]. This means that if an ML model is used to predict a suitable lighting condition, there will be a choice of millions of output class labels. However, this is not practical because it is very unlikely that a human eye can differentiate between similar lighting conditions. For example, Figure 1a shows two spatially separated points, A and B, on the color space that appear to be same. A solution can be to group similar colors into separate categories as shown in Figure 1b, where a million class labels are reduced to eight categories.

Let us assume that an ML algorithm h encounters ten data samples from a user Tom. For a given fixed context x, Tom selects a lighting preset L_A six times, lighting preset L_B three times, lighting preset L_C one time and does not select the lighting preset L_D. This means that for x the output probabilities for L_A, L_B, L_C and L_D based on frequency of occurrence are 0.6, 0.3, 0.1 and 0.0, respectively, i.e. a sample x can be classified into multiple output classes. Assume that, for a new context x_q, h selects L_B, whereas the Tom actually desires L_C. Here, h is not right in selecting the desired output, but it is not entirely wrong either, as Tom has not been consistent in selecting the desired lighting presets for the observed context. If CA were used as a metric to evaluate the performance of h, the selection made would be assessed as completely wrong. This is neither correct nor desirable. CA would not make a separation between selecting L_A or L_D. CA fails to capture the non-deterministic nature of this problem, as it is more desirable to measure how relevant the output is, for an observed context.

Let us now consider precision and recall metrics in the context of nDMOC problems. These metrics are generally used in binary classification problems where there are only two output classes (positive and negative). However, Sokolova and Lapalme [41] present different mechanisms to compute precision

Table 1: Representational comparison of two ML performance metrics for a smart lighting system considering a series of five predictions

Actual Output	**Predicted Output**	**CA Metric**	**Alternative Metric**
L_A	L_B	0	Very Relevant (80)
L_C	L_C	100	Absolutely Relevant (100)
L_B	L_C	0	Absolutely irrelevant (0)
L_B	L_B	100	Absolutely relevant (100)
L_A	L_C	0	Moderately relevant (70)
	Average	$200/5 = 40\%$	$350/5 = 70\%$

and recall for the case of multi-class classification problems.

In general, for a given classification problem, the goal is to select an ML algorithm that have high precision and recall (as close to 1.0) on all the considered output classes. For the above example and from Table 1, the precision for L_A is the ratio of the count of accurately predicted L_A and the count of predicted L_A, i.e. 0. Similarly, precision for L_B and L_C are $1/2 = 0.5$ and $1/3 = 0.33$ respectively. The recall for L_A is the ratio of the count of accurately predicted L_A and the count of actual L_A, i.e. $0/2 = 0$, whereas the recall for L_B and L_C are $1/2 = 0.5$ and $1/1 = 1$ respectively. In this example, we observe that the precision and recall are too low for most output classes. Due to non-deterministic input-output relationships in the data, ML algorithms are unable to precisely select or recall an output for a given context. Therefore, precision and recall are also not appropriate for such applications.

From the application point-of-view, it is more desirable to measure how relevant the selection is for a given input. The example in Table 1 shows that when the predicted and actual outputs do not match, CA fails to capture the relevance of the predicted output. The hypothetical *Alternative* metric exemplified in Table 1 computes the relevance of the predicted output for a given context and is more appropriate than CA.

In this direction, we use RS as the performance metric, which is more suitable for the category of nDMOC applications. Per sample RS is a score between 0 and 100, whereas for CA, it is either 0 or 100. In particular, it provides a performance function that maps an input-output pair (x, y) to a real number $RS : (x, y) \in (X, Y) \to [0, 100]$ rather than $\{0, 100\}$ as in CA.

3. Proposed Modifications to Relevance Score

In this section, we present the RS metric tailored to evaluate ML algorithms *in an online setting* for nDMOC problems. We describe the steps to compute two variants of RS, dubbed *Relevance Score - Case-by-Case* (RS_{CC}) and *Relevance Score - General* (RS_{Gen}). When we use the term RS, it refers to both $\mathrm{RS}_{\mathrm{CC}}$ and $\mathrm{RS}_{\mathrm{Gen}}$. A key step in computing RS is to compute an error score *(ErrScore)*, which is a real number that indicates the degree of mismatch between an actual output y_A and a predicted output y_P for an input x.

3.1. Relevance Score - Case-by-Case

We compute $\mathrm{RS}_{\mathrm{CC}}$ for a given predicted output, y_P by an ML algorithm in a number of steps. First, we define the concept of distance between outputs. For a given context, in a batch learning setting, $\mathrm{RS}_{\mathrm{CC}}$ is computed as described in Gopalakrishna et. al [18] [19]. In online learning, similar to the case of batch learning, $\mathrm{RS}_{\mathrm{CC}}$ is computed in two phases; i) computing posterior probabilities, and ii) evaluating the predicted output.

3.1.1. Computing Posterior Probabilities

Assume that for a context $x_q \in X$, an ML algorithm predicts an output y_P, while the actual output is y_A where $y_P, y_A \in Y$. The posterior probabilities for the output class labels are then computed from the set of observed samples. Let $\{x_1, x_2, \ldots, x_m\}$ denote the set of observed samples where $x_m = x_q$, Compute $P(y^{(l)}|x_q)$ where $l = 1, 2, \ldots, L$ using the observed samples. These probabilities are used to compute $ErrScore$. For the context x_q, the posterior probability of the predicted output is denoted by $P(y_P)$, the probability of the actual output is denoted by $P(y_A)$ and the probability of the most frequently selected output is denoted by $P(y_H)$.

A major difference in computing RS under batch and online learning settings is that, in batch learning, RS is computed based on the knowledge acquired from the entire dataset. This means that the underlying distribution is known to the RS function. However, in online learning, the samples arrive in sequence and there is no prior knowledge about the underlying distribution in the data. Therefore, the posterior probabilities need to be recomputed every time when an output is predicted for a new input.

3.1.2. Evaluating the Prediction

In this phase, the predicted outputs on the test data are evaluated. For a given context x, let d denote the probability distance between two outputs in general. We define d as the difference between the probabilities of two different outputs. The d value for different outputs is given by the following equations 1, 2 and 3.

$$d_{HP} = |P(y_H) - P(y_P)| \tag{1}$$

$$d_{PA} = |P(y_P) - P(y_A)| \tag{2}$$

$$d_{HA} = |P(y_H) - P(y_A)| \tag{3}$$

For a fixed context x, there are several possibilities based on the predicted and the actual outcomes and their computed probabilities. The cases may be evaluated qualitatively in terms of relevance of the predicted outputs. The decreasing order of relevance we consider for different cases is summarized in Table 2. The ordering is based on the consequence of individual cases as explained next. *ErrScore* is thus a score obtained by quantifying these cases.

Case 1: $y_P = y_A$*.* In this case, the predicted output and the actual output are equal. Thus, there is no error in the predicted output i.e., $ErrScore = 0$.

Table 2: Possibilities based on the predicted output, actual output, relationship between their probabilities and the corresponding decreasing order of relevance of the predicted output in qualitative terms

Case	**Outcome**	**Probability Condition**	**Qualitative Relevance**
1	$y_P = y_A$	-	Absolutely Relevant
2	$y_P \neq y_A$	$P(y_H) = P(y_P) = P(y_A)$	Very Relevant
3	$y_P \neq y_A$	$P(y_H) = P(y_P); P(y_P) > P(y_A)$	Relevant
4	$y_P \neq y_A$	$P(y_H) > P(y_P) > P(y_A)$	Moderately Relevant
5	$y_P \neq y_A$	$P(y_H) > P(y_A) > P(y_P)$	Slightly Irrelevant
6	$y_P \neq y_A$	$P(y_H) = P(y_A); P(y_A) > P(y_P)$	Absolutely Irrelevant

Case 2: $y_P \neq y_A; P(y_H) = P(y_P) = P(y_A)$. In this case, the predicted output and the actual output are not equal, whereas their probabilities are equal. This means that the y_H, y_P and y_A have occurred equally frequently, i.e., any of these three outcomes is equally good (statistically) for that context. Therefore, the $ErrScore$ in this case is also zero.

Case 3: $y_P \neq y_A; P(y_H) = P(y_P) > P(y_A)$. In this case, as shown in Figure 2a, the probabilities of the predicted output and the most frequently selected output are equal, i.e., we have $d_{HP} = 0$. This means that the prediction algorithm has selected the most frequently selected output but it has not been able to capture the change in output. The error is small and is equal to $\beta \cdot d_{PA}$ where β is a positive real constant, whose value depends on the application.

Case 4: $y_P \neq y_A; P(y_H) > P(y_P) > P(y_A)$. In this case, as shown in Figure 2b, the probabilities of predicted, actual and most frequently selected outcomes are not equal. The probability of the predicted output lies in between that of the most frequently selected output and the actual output. This means that the prediction algorithm has predicted an output that is not most frequently selected, but it is more likely than the actual output. Hence, the error is equal

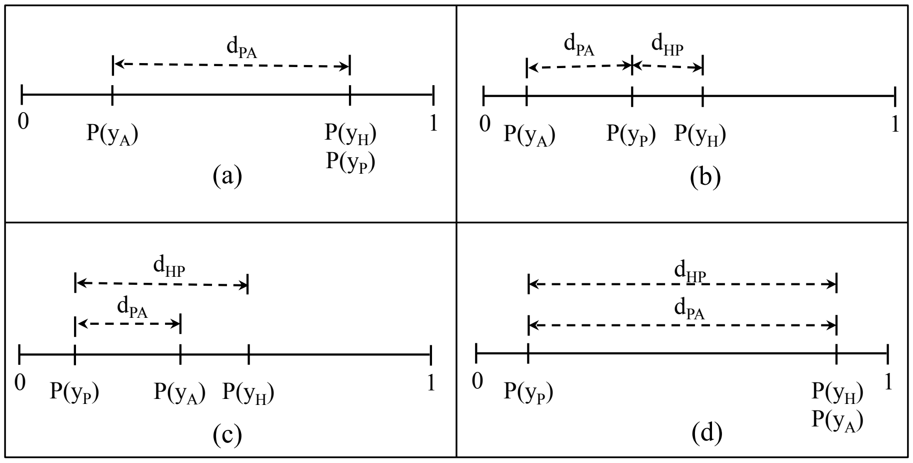

Figure 2: Possibilities that arise when predicted and actual outcomes are not same:
(**a**) $y_P \neq y_A$ and $P(y_H) = P(y_P); P(y_P) > P(y_A)$
(**b**) $y_P \neq y_A$ and $P(y_H) > P(y_P) > P(y_A)$
(**c**) $y_P \neq y_A$ and $P(y_H) > P(y_A) > P(y_P)$
(**d**) $y_P \neq y_A$ and $P(y_H) = P(y_A); P(y_A) > P(y_P)$

to $(\alpha \cdot d_{HP} + \beta \cdot d_{PA})$ where α denotes the positive real constant.

Case 5: $y_P \neq y_A; P(y_H) > P(y_A) > P(y_P)$. In this case, as shown in Figure 2c, the probability of the predicted output is less than that of the most frequently selected output and of the actual output. This means that the prediction algorithm has predicted an output that is not most frequently selected, and is less probable than the actual output. Hence, d_{HP} is higher than that in Case 4 and the error is equal to $(\alpha \cdot d_{HP} + \beta \cdot d_{PA})$ as in Case 4.

Case 6: $y_P \neq y_A; P(y_H) = P(y_A) > P(y_P)$. In this case, as shown in Figure 2d, the probability of the actual output and that of the most frequently selected output are equal, but the ML algorithm predicts a different output. The performance of the ML algorithm was not sufficiently good to select the same output, the error is much higher than that of the previous cases. Since $d_{HP} = d_{PA}$, the error is equal to $(\alpha + \beta) \cdot d_{HP}$.

Combining above equations and normalizing over $(\alpha + \beta)$, we have the following $ErrScore$ computation as given by 4,

$$ErrScore = \frac{\alpha(d_{HP}) + \beta(d_{PA})}{\alpha + \beta}. \tag{4}$$

The RS value for a context x is thus calculated as a function of $ErrScore$ as given by 5 and 6,

$$RS_{CC} = (1 - ErrScore) \times 100, \tag{5}$$

$$RS_{CC} = (1 - \frac{\alpha(d_{HP}) + \beta(d_{PA})}{\alpha + \beta}) \times 100. \tag{6}$$

3.2. Relevance Score - General

As mentioned in Section 1, one of the main reasons for nDMOC problems to arise is that been difficult to model certain features such as user mood, behavior and other cognitive factors. In nDMOC applications, if there are features that capture or identify only certain aspect of an entity, we dub them as *abstract* features. For example, in case of smart lighting example, the feature *User ID* only identifies who the user is, but there are no other features that reflect their characteristics such as mood, behavior and other cognitive factors. Hence, UID can be considered *abstract* feature. It is important to note that every nDMOC application need not necessarily have abstract features.

$\mathrm{RS_{Gen}}$ is a variant of RS, computed by discarding the values of *abstract* features. Leaving the abstract feature out during the RS computation makes the dataset as if it is derived from a single distribution. Note that the ML algorithms are not rerun on the new dataset. By ignoring the values of the abstract feature (c), the input dimension is reduced by one (X_{-c}). The posterior probabilities $P(y^{(l)}|x)$ for $l = 1, 2, \ldots, L$ are then computed. The remaining process to compute $\mathrm{RS_{Gen}}$ is same as that of the $\mathrm{RS_{CC}}$. The motivation to compute $\mathrm{RS_{Gen}}$ is to study the performance of ML algorithms when the data seems to be from a single distribution, although the dataset still has non-deterministic input-output relationships.

4. Experiments and Results

We perform the experiments to study the behavior of ML algorithms in an online setting using CA and RS as metrics, in the context of nDMOC problems. The performance of the ML algorithms in an online setting is analyzed through learning curves. A learning curve gives a measure of predictive performance (for example, CA) on a given task as a function of some measure of varying amounts of learning effort such as time and number of samples [42]. Generally, in ML and statistics, a learning curve represents the generalization performance of the algorithm as a function of the number of observed samples (training set) [43], i.e. error measured on the test samples versus the number of observed samples. If an ML algorithm is deployed in production, the learning curves help in analyzing its runtime performance. The crossing of learning curves indicates that an ML algorithm may perform better than the other for a larger number of observed samples [44]. In our experiments, a learning curve represents the prediction performance of an ML algorithm as a function of the number of observed samples.

We explain the process of producing learning curves using the CA metric. The process is repeated for the RS metric in the same way. The selection of α and β parameters depends on the application. In the RS computation, the parameters α and β are the weights on the probabilistic distances d_{HP} and d_{PA} respectively. Therefore, having α higher than the β penalizes RS when the predicted output is far from the frequently selected output; and β higher than α penalizes RS when the predicted output is away from the actual output. Since human factors are involved in both the applications that we have considered, in the long term, it is less likely that most humans tend to switch their preferences very often [1]. Hence, the RS metric is used with $\alpha = 2$ and $\beta = 1$ [18] [19]. For each sample i in the considered dataset, we compute the CA of the predicted output (CA_i). Subsequently, for each sample i in the dataset, we compute the average CA until the sample i written as $CA(i)$, as in equation 7. ($CA(i)$) is then plotted as a function of i. In order to obtain a smooth learning curve, we

Table 3: Description of Datasets

Dataset	**Breakout**	**Wine (White)**
Number of samples	236	4898
Number of input features	6	8
Type of input features	5-Categorical	11-Numeric
	1-Numeric	
Output Cardinality	8	10
Type of output	Categorical	Categorical
		Numerical

repeat the process on 10 randomly shuffled datasets for each learning algorithm and then average the resulting curves.

$$CA(i) = \frac{\sum_{k=1}^{i} CA_k}{i} \tag{7}$$

We perform the experiments using the Breakout dataset [1] and the Wine dataset available from the UCI machine learning repository. The datasets are summarized in Table 3. While the Breakout dataset contains 236 samples, the Wine dataset contains 4898 samples. The number of input features in the Breakout dataset is 6 and that of the Wine dataset is 8. The number of output class labels or output categories in the Breakout dataset is 8, while that of the Wine dataset is 11. As discussed in Section 2.2, we have considered two online ML algorithms; *AROW* and *SCW*, and two instance-based algorithms (in an online setting); k_{Conv} and *Dwk*.

4.1. Experiments on the Breakout *dataset*

The Breakout dataset is used for the purpose of developing smart lighting by identifying the relationship between the contexts and their corresponding output lighting conditions. The dataset contains 236 samples with six input features and eight output class labels. The inputs include data gathered implicitly from sensors (Passive Infra Red (PIR) sensors for monitoring movements, sound pressure sensors for monitoring sound volume intensity and light sensors for measuring external light influence) and explicitly from users that constitute a context. The output class labels are of categorical data type that consists of eight preset lighting conditions from which the users can select a lighting condition for a given context. The output lighting conditions are a combination of the following three lighting properties: Warm/Cool, Bright/Dim and Dynamic/Static [1]. For example, Warm-Dim-Static is one combination of a lighting condition. Therefore, the problem can be viewed as a classification task. Of all the features, the feature *Time of the Day* (ToD) is of numerical data type and the rest are of categorical data type. Since the categorical data types are needed for computing RS values, the feature ToD is converted into categorical data type by assigning a category to each of its feature value. In our experiments, we have assigned the following categories: Category 1 [8.00, 12.00], Category 2 (12.00, 16.00], Category 3 (16.00, 20.00] and Category 4 (20.00, 8.00). We also reduce the number of categories to two for the feature *Intensity of Activity* (IoA) where Category 1 [0-2] and Category 2 [3-10].

Figure 3 shows the learning curves for the online and instance-based ML algorithms with CA as the metric. The CA performance of the online ML algorithms is much lower compared to that of instance-based algorithms and both perform poorly according to CA assessment. The CA performance of AROW varies significantly as a function of the number of samples. However, the learning curve of the SCW algorithm is smoother. It can be observed that after around 30 samples, while the CA performance of AROW almost converges, the CA performance of SCW still improves. Even though the initial learning rate is low for SCW, it consistently improves to provide better performance than AROW. In case of instance-based algorithms, the performance of Dwk converges after around 30 samples, similar to AROW. The k_{Conv} algorithm provides better CA performance than Dwk.

Figure 4 shows the learning curves for the online and instance-based ML algorithms with RS_{CC} as the metric. The learning curve using the RS_{CC} metric indicates how relevantly the ML algorithms predict as a function of the observed

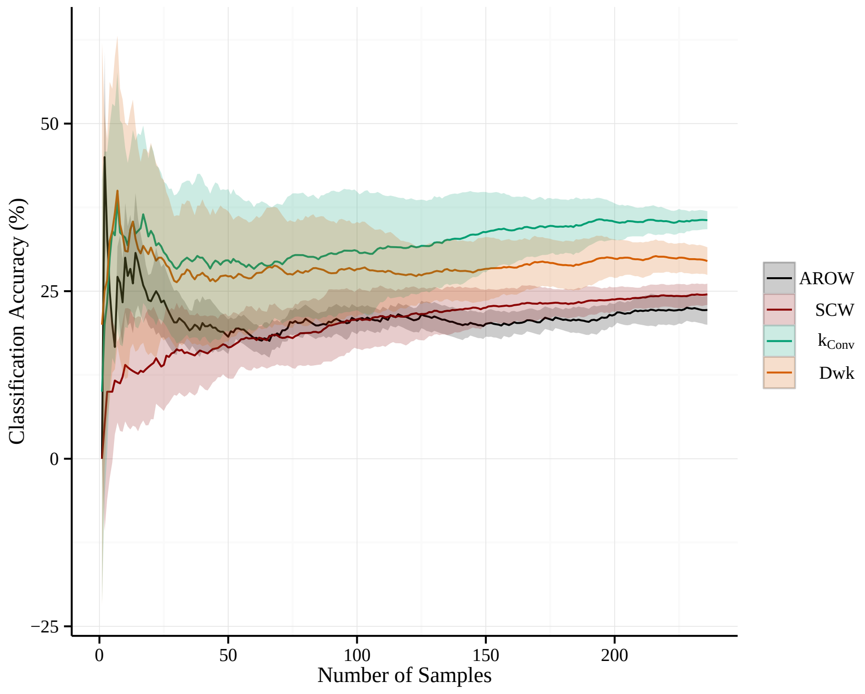

Figure 3: Learning curves of the Online and Instance-based ML algorithms for the Breakout dataset with *Classification Accuracy* as the metric

samples. Similar to the case of the CA metric, the RS_{CC} performance of online ML algorithms is much lower compared to that of the instance-based algorithms. As expected, the RS_{CC} performance of the ML algorithms is higher than that of the CA performance. This indicates that the predicted outputs may not be accurate but are still relevant for the given contexts. k_{Conv} provides the maximum RS_{CC} performance among all the considered ML algorithms. We can see that the sample size is not enough to observe the convergence of the performance of the considered ML algorithms.

Figure 5 shows the learning curves for the online and instance-based ML algorithms with RS_{Gen} as the metric. RS_{Gen} is computed by ignoring the *abstract* feature i.e. user-identity UID from the Breakout dataset. The RS_{CC} metric evaluates how well the ML algorithms predict for each context described

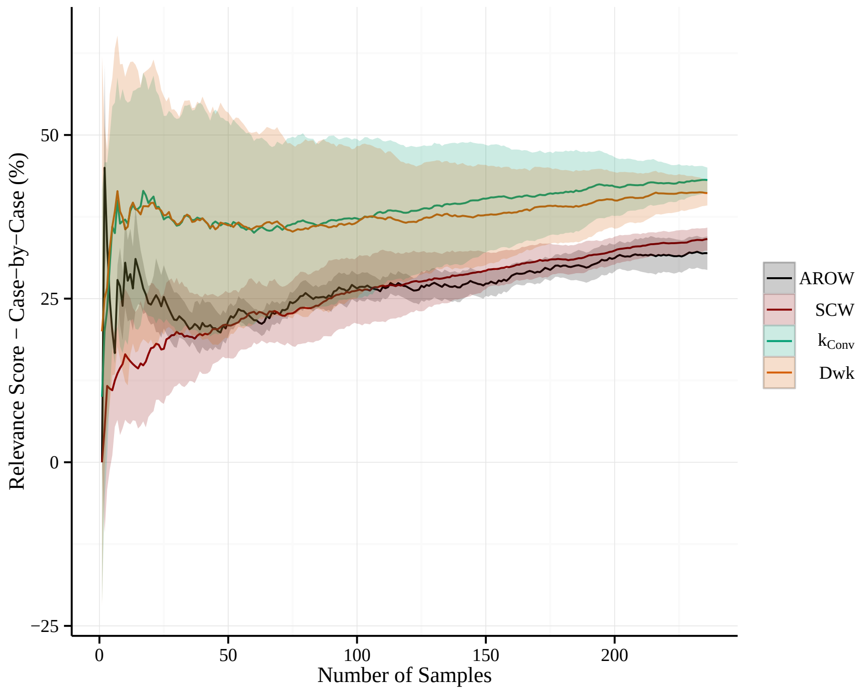

Figure 4: Learning curves of the Online and Instance-based ML algorithms for the Breakout dataset with *Relevance Score - Case-by-Case* as the metric

by five objective features and the user. By ignoring *UID*, in case of RS_{Gen}, the dataset looks as if it is collected from a single user. Therefore, the RS_{Gen} metric evaluates how well the ML algorithms predict for each context without being specific to users.

Therefore, the RS_{Gen} learning curves indicate that when there is large number of samples for each user, the RS performance will become smoother and keep improving. The RS_{Gen} performance of the online ML algorithms is lower compared to that of the instance-based algorithms. The learning curves of the ML algorithms are smoother compared to that of the CA and RS_{CC}. Again, k_{Conv} provides the maximum RS_{Gen} performance among all the considered ML algorithms. In case of the instance-based algorithms, the learning curve of k_{Conv} crosses that of Dwk after around 170 samples, indicating better RS_{Gen} perfor-

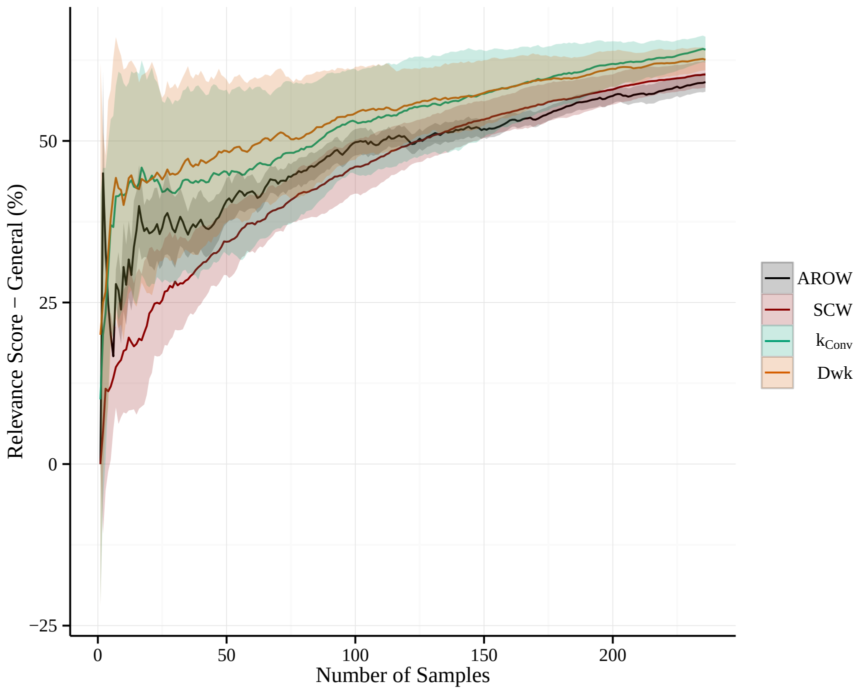

Figure 5: Learning curves of the Online and Instance-based ML algorithms for the Breakout dataset with *Relevance Score - General* as the metric

mance can be achieved using $\mathrm{k_{Conv}}$ in the long run. As in the case of $\mathrm{RS_{CC}}$, the sample size is not enough to observe the convergence of the performance of the considered ML algorithms.

From this study, we observe that even though the considered ML algorithms are not designed for the nDMOC problems, the ML algorithms still continue to learn as the number of observed samples increases. The learning curves using $\mathrm{RS_{Gen}}$ indicate that as more samples are collected from each user, the prediction gets better and consistent. However, these algorithms cannot predict more accurately with increasing samples as seen in the learning curves using the CA metric. Also, we observe that the standard deviation of the prediction performance, which is very high initially as indicated in the Table 4, reduces as more samples are observed, meaning that the performance becomes more consistent

Table 4: The performance of the k-Nearest Neighbor (k_{Conv}) algorithm as a function of the number of samples observed.

Number of Samples	**Classification Accuracy (%)**	**Std. Dev (CA)**	**Relevance Score - CC (%)**	**Std. Dev ($\mathbf{RS_{CC}}$)**	**Relevance Score - Gen (%)**	**Std. Dev ($\mathbf{RS_{Gen}}$)**
50	29.6	10.6	36.4	15.7	45.3	12.5
100	31.0	8.8	37.2	12.1	53.0	8.1
150	33.8	6.0	40.3	8.2	57.3	6.7
200	35.4	2.8	42.1	4.5	61.9	3.5
236	35.6	1.4	43.1	1.9	64.1	2.0

i.e. the variability in the performance reduces. Therefore, the experimental results using the RS metrics indicate that the ML algorithms still learn and also provide better relevant outputs. Also, we see that the performance of the ML algorithms measured using all the three metrics considered is quite poor i.e. we do not see where the performance converges. This is due to the size of the dataset.

4.2. Experiments on the Wine *dataset*

In order to further understand the behavior of the ML algorithms such as the learning curve and convergence on a larger dataset, we performed the experiments on Wine [23] dataset available from the UCI ML repository. Like the Breakout dataset, the Wine dataset also has one-to-many input-output relationships. However, it has more samples compared to the Breakout dataset.

The *Wine* dataset is used to determine the quality of a wine sample based on the measurements from objective tests such as pH values, density and sulphates. The Wine dataset is of two types: red with 1599 samples and white with 4898 samples. The inputs include measurements from objective tests that are of numerical data type and the output is based on sensory data (median of at least 3 evaluations made by wine experts). The output class labels are numeric and ordered with the wine quality ranging between 0 (very bad) and 10 (excellent). Therefore, the problem addressed using the Wine dataset can be considered

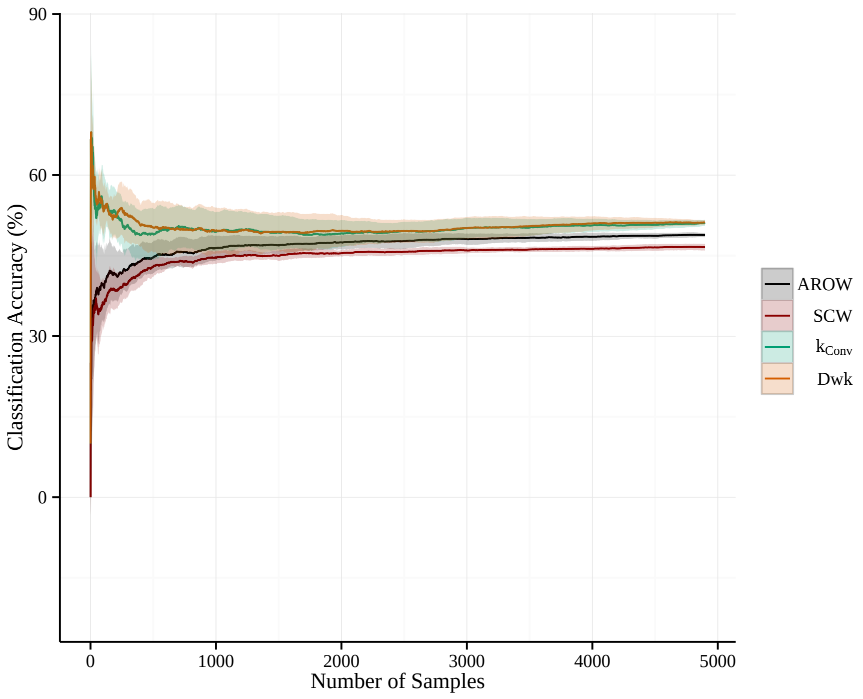

Figure 6: Learning curves of the Online and Instance-based ML algorithms using Classification Accuracy as the metric for the Wine (white) dataset (W_5) with each feature having 5 categories

either as a classification or a regression task. For our experiments, we consider the Wine (white) dataset as a classification task.

In order to apply RS, we need the input features to be of categorical data type whereas the input features in the Wine dataset are of numerical data type. Thus, for our experiments, we divide each input feature into 5 equal categories. We denote the corresponding dataset by W_5. The output class labels are unaltered. We use the same ML algorithms: $\mathrm{k_{Conv}}$, Dwk, AROW and SCW that were used for the Breakout dataset. We do not consider the $\mathrm{RS_{Gen}}$ metric, as all the input features in the Wine dataset are measurements using objective tests and there are no abstract features.

Figure 6 shows the learning curves of the considered ML algorithms for W_5 with the CA metric. The learning curves start to converge after 1000 samples.

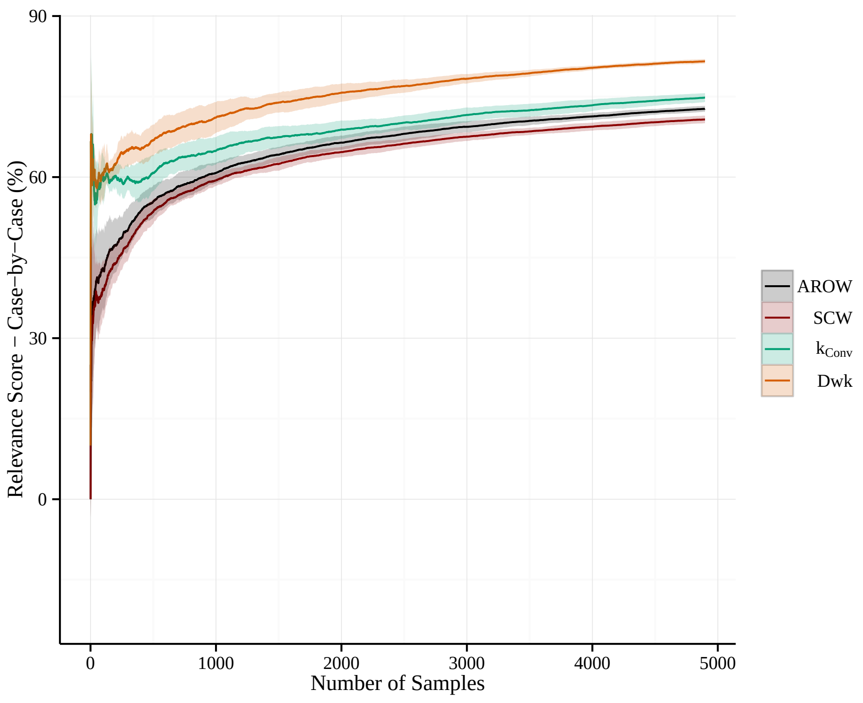

Figure 7: Learning curves of the Online and Instance-based ML algorithms using Relevance Score - Case-by-Case as the metric for the Wine (white) dataset (W_5) with each feature having 5 categories

This means that after 1000 samples, the ML algorithms do not continue to learn. Hence, the performance of these algorithms remains the same. The instance-based algorithms perform slightly better than the online ML algorithms i.e. around 2%. Unlike in the case of Breakout dataset, AROW provides better CA performance (about 2%) than SCW. An interesting behavior of the instance-based algorithms is observed with the CA metric. Initially, when there are few samples, their CA performance increases drastically to 60%. Then, the performance gradually decreases to 50%, when more samples are observed. However, in the case of online algorithms, the CA performance keeps steadily increasing. The reason is that in case of online ML algorithms, the weights are modified based on the observed context and the actual output, when new samples are

Table 5: The performance of the Distance weighted k-Nearest Neighbor algorithm on the Wine (White) dataset as a function of the number of samples observed.

Number of Samples	**Classification Accuracy (%)**	**Std. Dev (CA)**	**Relevance Score - CC (%)**	**Std. Dev ($\mathbf{RS_{CC}}$)**
1000	49.6	4.3	71.1	2.9
2000	49.6	2.9	75.7	1.7
3000	50.1	2.1	78.3	0.9
4000	51.0	1.3	80.4	0.4
4898	51.2	0.4	81.6	0.4

encountered. This makes the learning of online ML algorithms, slow and steady.

Figure 7 shows the learning curves of the considered ML algorithms for W_5 with the $\mathrm{RS_{CC}}$ metric. While the CA performance of the ML algorithms converge after 1000 samples, the $\mathrm{RS_{CC}}$ performance still keeps improving even after 4500 samples. This shows that for nDMOC problems, the ML algorithms can provide more relevant output with more number of observed samples, even though their accuracy does not improve after a certain number of samples. Here, the Dwk algorithm provides the best performance among the considered algorithms. This indicates that the distance of the neighboring contexts has significant influence on the output selection. This means that the closest neighbors represent the observed context more than the farthest neighbors. Also, we observe that the standard deviation of the prediction performance, which is very high initially as indicated in the Table 5, reduces as the more number of samples are observed.

From the experiment using Wine dataset with 4898 samples, we confirm that in the context of nDMOC problems, the ML algorithms continue to learn as the number of samples increases. Even though the ML algorithms cannot predict more accurately after few samples, the relevance of the predicted outputs keeps improving.

4.3. Significance of the Relevance Score Metric

The objective of this experiment is to demonstrate the significance of RS metric in the context of nDMOC problems. As mentioned earlier, the output class labels in the Wine dataset are numeric and ordered with the wine quality ranging between 0 (very bad) and 10 (excellent). Therefore, the Absolute Difference (AD) can also be considered as a metric to evaluate the performance of prediction algorithms. AD is computed as the difference between the predicted and actual output. The key difference between CA and AD is that CA measures whether the predicted output is right or wrong, whereas AD depends on the actual error. Therefore, for the Wine dataset, the performance measured using AD represents the ground truth against which the performance using CA and $\mathrm{RS}_{\mathrm{CC}}$ can be compared. The performance values obtained using the AD metric have been normalized to 100 i.e., [0, 100] using the expression $(10 - Error) \times 10$ whereby $Error = |y_A - y_P|$, so that the results can be directly compared with those of the CA and $\mathrm{RS}_{\mathrm{CC}}$ metrics.

We use the Dwk algorithm as it provides the maximum prediction performance among the other considered algorithms. Since the RS metric can be computed when the input features are categorical, we consider two cases. In each case we divide each input feature of the Wine dataset into 5 and 3 equal categories. We denote the corresponding datasets by W_5 and W_3, respectively. The output class labels are unaltered. When we divide each input feature into 5 equal categories, we group the feature values that are close to each other into a category. This means that we are introducing uncertainty between the input and output by creating the overlap in the input space. Consequently, there is more possibility of observing the same input being mapped to different output class labels. It is more difficult for an ML algorithm to predict the exact output for a given input, when there is more than one acceptable output.

Figure 8 shows the learning curves for the Dwk algorithm for the W_5 with all the three metrics considered. In case of W_5, there are not many samples that have one-to-many input-output relationships and the input space is less overlapped. As a result, the CA performance of W_5 remains low. Also, the

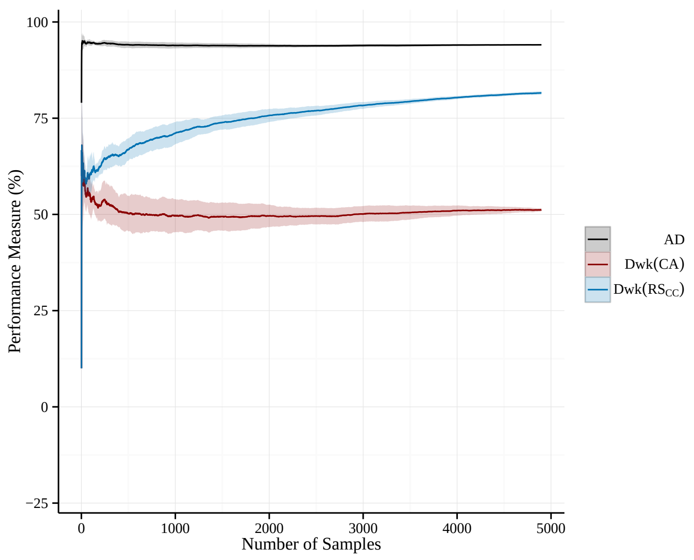

Figure 8: Learning curves of the Distance weighted k-Nearest Neigbor (Dwk) algorithm for the Wine (white) dataset (W_5) with each feature having 5 categories

considered input features are not sufficient to determine the output more accurately. However, the performance measured using the AD metric is very high i.e. around 95%, which means that (when $y_P \neq y_A$) for the given contexts, the ML algorithm is predicting outputs that are not totally wrong. For example, if $y_P = 3$ and $y_A = 4$ the CA metric scores a zero whereas the AD metric scores 90%. The performance measured using the RS$_{\text{CC}}$ metric is more close to AD than the CA.

Figure 9 shows the learning curves for the Dwk algorithm for the W_3 with all the three metrics considered.In case of W_3, we increase the overlapping of the input space, i.e. we introduce more uncertainty between the input and output by reducing the number of categories for each input feature to 3. It can be seen that for W_3 the performance goes below 50% with the CA metric. This shows that when the input-output relationship in the dataset becomes more uncertain, i.e. when the randomness increases, it becomes more difficult for Dwk to predict accurate outputs. However, the performance with the AD

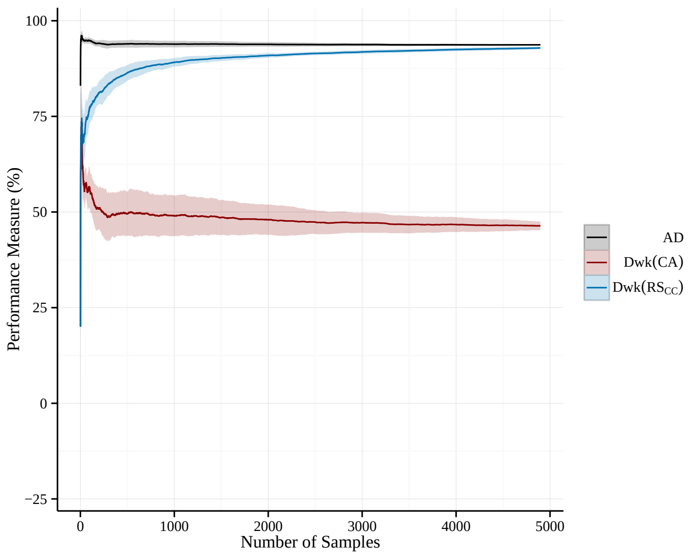

Figure 9: Learning curves of the Distance weighted k-Nearest Neigbor (Dwk) algorithm for the Wine (white) dataset (W_3) with each feature having 3 categories

metric is almost the same as in case of W_3. This reflects that introducing little uncertainty does not really affect the performance of the Dwk algorithm as seen through the AD metric. When $y_P \neq y_A$, Dwk predicts a relevant output despite the increased randomness. The performance measured using the RS$_{\text{CC}}$ metric increases significantly and tends to get closer to the performance measured using the AD metric. This shows that when there are one-to-many input-output relationships in a dataset, RS$_{\text{CC}}$ gets closer to the ground truth.

An observation is that the ML algorithms achieve high RS$_{\text{CC}}$ performance more quickly in the case of W_3 than in W_5. Moreover, the CA performance of the Dwk algorithm starts to decrease after 1000 samples. However, since the predicted outputs are still relevant for the contexts, high RS$_{\text{CC}}$ performance is achieved, even with few samples. When more samples are observed, the uncertainty between the input-output still remain high and hence the RS$_{\text{CC}}$ performance remains high, but with reduced learning rate. From this study, we find that when there is less uncertainty in the dataset, the initial RS$_{\text{CC}}$

performance of ML algorithms will be low, after that the learning proceeds in a faster rate. When there is more uncertainty in the dataset, the initial RS_{CC} performance of ML algorithms will be high, after that the learning proceeds in a slower rate. For the nDMOC problems, we observe that using CA as a metric the learning converges faster with lower performance, which is not really the case as observed using the RS metric.

5. Discussion

In general, the performance of online ML algorithms is lower than that of the instance-based algorithms for all the considered metrics. This may be because the online ML algorithms do not store the observed samples, and hence significant information is lost. In online algorithms, the learning happens smoothly, while for the instance-based algorithms, learning becomes smooth only after an initial set of samples has been observed. This behavior is even more evident with the Wine dataset, meaning that sufficient information (underlying distribution of the samples) is obtained with more samples.

The experiments using the CA and RS metrics shows that RS is more appropriate to evaluate the performance of the ML algorithms for nDMOC problems. ML algorithms are not very accurate within the first few observed samples. However, they keep learning and provide more relevant predictions as the number of observed samples increases, which is particularly true for the RS metric. This means that the ML algorithms predict relevant outputs more consistently than the accurate outputs.

The experiments using the two different natured Wine dataset show that when there is more uncertainty in the input-output relationship, the ML algorithms achieve high RS_{CC} performance more quickly and then the learning rate decreases significantly as the number of samples increases. In contrast, when there is less uncertainty in the input-output relationship, the ML algorithms have low RS_{CC} initially but are able to keep learning at a faster rate over time.

6. Conclusions

In cognitive applications involving ML, nDMOC problems are characterized by a one-to-many relationship between input and output. By this we mean that there is no single output that may be entirely right for a given context. In such cases, conventional ML algorithms fail to make accurate predictions. Even more, when common metrics such as CA are used, the overall performance is poor, often leading to inconsistent or misleading results. We considered consistency of users' preferences and relevance of predicted outputs, which allowed us to tackle the difficult case of cognitive systems.

In this paper, we experimented with the instance-based and online ML algorithms, discussing the crucial impact that various metrics have in assessing the performance of these algorithms. We focused on the *Relevance Score* metric and its effectiveness in an online setting. The RS metric evaluates the performance of an ML algorithm by the relevance of the predicted output rather than considering its accuracy. We further introduced two variants of RS: Relevance Score - Case-by-Case ($\mathrm{RS}_{\mathrm{CC}}$) and Relevance Score - General ($\mathrm{RS}_{\mathrm{Gen}}$) that can be used to study various aspects of an ML algorithm. The RS for a predicted output and a given context is computed based on the posterior probabilities computed from all the observed samples in a dataset.

The behavior of the ML algorithms is studied through experiments, first on the Breakout dataset and then on the Wine dataset, using the CA and RS metrics. The performance of the ML algorithms is analyzed and compared to each other using learning curves. We found that the instance-based algorithms provide better prediction performance than online ML algorithms in the context of nDMOC problems. The learning curves also show that using the RS metric, the performance of the ML algorithms keeps improving as the number of observed samples grows, even after the CA performance converges. The experiments on two variants of the Wine dataset show that the RS metric becomes more appropriate than the CA metric as the uncertainty between the input and output increases. Furthermore, the learning rate of the ML algorithms depends on the degree of uncertainty in the dataset. Finally, we conclude that instance-based learning is the most suitable approach and RS is an appropriate performance metric for the class of applications where dynamic factors such as variable human perceptions lead to nDMOC problems.

When it comes to predicting human preferences, the focus should not be to aim for 100% accuracy but, rather, in determining the degree of relevance of a prediction, considering a given context. In cases such as smart lighting, being able to achieve an absolute accuracy of 50% would not be a great result if the users were 100% consistent in their choices. However, a similar accuracy in the case of very inconsistent users, if the RS score is high, we conclude that the algorithm makes very relevant predictions even when it fails to make accurate predictions. ML algorithms cannot make accurate predictions for users who are consistently inconsistent, which is the case where human predictions would fail too. However, we have shown that ML can indeed still be used for nDMOC problems and that they keep learning, while the convergence may take a significantly large number of instances.

Acknowledgements

This work was supported by Smart Context-aware Services (SmaCS) project through the Point One grant no. 10012172. The authors would like to thank TU/e Intelligent Lighting Institute for providing the Breakout Dataset.

References

[1] A. K. Gopalakrishna, T. Ozcelebi, A. Liotta, J. J. Lukkien, Statistical Infererence for Intelligent Lighting: A Pilot Study, in: Intelligent Distributed Computing VIII, Vol. 570 of Studies in Computational Intelligence, Springer International Publishing, 2015, pp. 9–18. doi:10.1007/978-3-319-10422-5_3.

[2] A. B. Watson, L. Kreslakeb, Measurement of visual impairment scales for

digital video, in: Proceedings of the SPIE, 2001, pp. 79–89. doi:10.1117/12.429526.

[3] R. N. Shepard, Psychological relations and psychophysical scales: On the status of "direct" psychophysical measurement, Journal of Mathematical Psychology 24 (1) (1981) 21–57. doi:10.1016/0022-2496(81)90034-1.

[4] M. Caciotta, S. Giarnetti, F. Leccese, B. Orioni, M. Oreggia, C. Pucci, S. Rametta, Flavors mapping by Kohonen network classification of Panel Tests of Extra Virgin Olive Oil, Measurement: Journal of the International Measurement Confederation 78 (2016) 366–372. doi:10.1016/j.measurement.2015.09.051.

[5] U. Gneezy, S. Meier, P. Rey-Biel, When and Why Incentives (Don't) Work to Modify Behavior, Journal of Economic Perspectives 25 (4) (2011) 191–210. doi:10.1257/jep.25.4.191.

[6] K. Aquino, D. Freeman, A. Reed, W. Felps, V. K. Lim, Testing a Social-Cognitive Model of Moral Behavior: The Interactive Influence of Situations and Moral Identity Centrality, Journal of Personality and Social Psychology 97 (1) (2009) 123–141. doi:10.1037/a0015406.

[7] I. Rojek, Models for Better Environmental Intelligent Management within Water Supply Systems, Water Resources Management 28 (2014) 3875–3890. doi:ModelsforBetterEnvironmentalIntelligentManagementwithinWaterSupplySystems.

[8] M. Aly, Survey of Multiclass Classification Methods, Tech. rep., California Institute of Technology (2005).

[9] I. H. Witten, E. Frank, M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd Edition, Morgan Kaufmann, 2011, Ch. 1, pp. 21–26.

[10] G. Tsoumakas, I. Katakis, Multi-Label Classification: An Overview, International Journal of Data Warehousing and Mining 3(3) (2007) 1–13. doi:10.1.1.104.9401.

[11] A. K. McCallum, Multi-label text classification with a mixture model trained by EM, in: AAAI'99 Workshop on Text Learning, 1999. doi:10.1.1.35.888.

[12] K. Trohidis, G. Tsoumakas, G. Kalliris, I. Vlahavas, Multi-Label Classification of Music into Emotions, EURASIP Journal on Audio, Speech, and Music Processing (1) (2011) 4. doi:10.1186/1687-4722-2011-426793.

[13] A. K. Gopalakrishna, T. Ozcelebi, A. Liotta, J. J. Lukkien, Exploiting Machine Learning for Intelligent Room Lighting Applications, in: The 6th IEEE International Conference on Intelligent Systems, Vol. 1, 2012, pp. 406–411. doi:10.1109/IS.2012.6335169.

[14] V. Menkovski, A. Liotta, Adaptive psychometric scaling for video quality assessment, Signal Processing: Image Communication 27 (8) (2012) 788–799. doi:10.1016/j.image.2012.01.004.

[15] N. Japkowicz, Why question machine learning evaluation methods?, in: AAAI'06 workshop on evaluation methods for machine learning, 2006, pp. 6–11.

[16] R. K. Jain, The Art of Computer System Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling, John Wiley and Sons, Inc., 1991.

[17] M. Sokolova, N. Japkowicz, S. Szpakowicz, Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation, in: AI 2006: Advances in Artificial Intelligence, Vol. 4304, Springer Berlin Heidelberg, 2006, pp. 1015–1021. doi:10.1007/11941439_114.

[18] A. K. Gopalakrishna, T. Ozcelebi, A. Liotta, J. J. Lukkien, Relevance as a Metric for Evaluating Machine Learning Algorithms, in: P. Perner (Ed.),

Machine Learning and Data Mining in Pattern Recognition, Vol. 7988 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 195–208. doi:978-3-642-39712-7_15.

[19] A. K. Gopalakrishna, T. Ozcelebi, J. J. Lukkien, A. Liotta, Relevance in Cyber-Physical Systems with Humans in the Loop, Concurrency and Computation: Practice and Experience 29. doi:10.1002/cpe.3827.

[20] S. Albers, Interactive Computation, Vol. 97, Springer-Verlag Berlin Heidelberg, 2006, Ch. Online Algorithms, pp. 143–164. doi:10.1007/3-540-34874-3.

[21] A. K. Gopalakrishna, T. Ozcelebi, J. J. Lukkien, A. Liotta, Evaluating Machine Learning Algorithms for Applications with Humans in the Loop, in: 14th IEEE International Conference on Networking, Sensing and Control, IEEE, 2017. doi:10.1109/ICNSC.2017.8000136.

[22] S. Offermans, A. K. Gopalakrishna, H. A. van Essen, T. Ozcelebi, Breakout 404: A Smart Space Implementation for Lighting Services in the Office Domain, in: Proceedings of the 9th International Conference on Networked Sensing Systems, 2012, pp. 1–4. doi:10.1109/INSS.2012.6240532.

[23] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, Decision Support Systems 47 (4) (2009) 547–553. doi:10.1016/j.dss.2009.05.016.

[24] M. Lichman, UCI Machine Learning Repository (2013).

[25] A. C. Davison, Statistical Models, Vol. 11 of Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press, 2003. doi:10.1017/CBO9780511815850.

[26] Y. Kumar, G. Sahoo, Analysis of Parametric and Non Parametric Classifiers for Classification Technique using WEKA, International Journal on Information Technology and Computer Science 4 (7) (2012) 43–49. doi:10.5815/ijitcs.2012.07.06.

[27] S. C. H. Hoi, J. Wang, P. Zhao, LIBOL: A Library for Online Learning Algorithms, Journal of Machine Learning Research 15 (2014) 495–499. URL http://jmlr.org/papers/v15/hoi14a.html

[28] K. Crammer, A. Kulesza, M. Dredze, Adaptive regularization of weight vectors, Machine Learning 91 (2) (2013) 155–187. doi:10.1007/s10994-013-5327-x.

[29] J. Wang, P. Zhao, S. C. H. Hoi, Exact soft confidence-weighted learning, in: International Conference on Machine Learning, 2012. doi:10.1.1.421.4367.

[30] N. Natarajan, A. Tewari, I. Dhillon, P. D. Ravikumar, Learning with Noisy Labels, in: Neural Information Processing Systems (NIPS), 2013, pp. 1196–1204.

[31] K. Crammer, M. Dredze, A. Kulesza, Multi-class Confidence Weighted Algorithms, in: Conference on Empirical Methods in Natural Language Processing, 2009, pp. 496 – 504.

[32] D. W. Aha, D. Kibler, M. K. Albert, Instance-Based Learning Algorithms, Machine Learning 6 (1991) 37–66. doi:10.1007/BF00153759.

[33] T. Mitchell, Machine Learning, Mcgraw Hill, 1996, Ch. 8, pp. 239–258.

[34] M. Khan, Q. Ding, W. Perrizo, k-nearest Neighbor Classification on Spatial Data Streams Using P-trees, in: Advances in Knowledge Discovery and Data Mining, Vol. 2336 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2002, pp. 517 – 528. doi:10.1007/3-540-47887-6_51.

[35] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, John Wiley and Sons, Inc., 2001.

[36] S. A. Dudani, The Distance-Weighted k-Nearest-Neighbor Rule, IEEE Transactions on Systems, Man and Cybernetics SMC-6 (4) (1976) 325–327. doi:10.1109/TSMC.1976.5408784.

[37] D. M. W. Powers, Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness and Correlation, Tech. rep., Flinders University (2007). doi:10.1.1.214.9232.

[38] S. Godbole, S. Sarawagi, Discriminative Methods for Multi-labeled Classification, in: Advances in Knowledge Discovery and Data Mining, Vol. 3056 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 22–30. doi:10.1007/978-3-540-24775-3_5.

[39] S. Shalev-Shwartz, Online Learning: Theory, Algorithms and Applications, Ph.D. thesis, Hebrew University (2007).

[40] P. C. Kinetics, Controlling LED Lighting, Tech. rep., Philips Solid-State Lighting Solutions Incorporation (2011).

[41] M. Sokolova, G. Lapalme, A systematic analysis of performancemeasures for classification tasks, Information Processing and Management 45 (4) (2009) 427–437. doi:10.1016/j.ipm.2009.03.002.

[42] C. Perlich, Learning Curves in Machine Learning, Springer US, Boston, MA, 2010, pp. 577–580. doi:10.1007/978-0-387-30164-8_452.

[43] B. W. Flury, M. J. Schmid, A. Narayanan, Error Rates in Quadratic Discrimination with Constraints on the Covariance Matrices, Journal of Classification 11 (1994) 101–120. doi:10.1007/BF01201025.

[44] C. Perlich, F. Provost, J. S. Simonoff, Tree Induction vs. Logistic Regression: A Learning-Curve Analysis, Journal of Machine Learning Research 4 (2003) 211 – 255. doi:10.1162/153244304322972694.