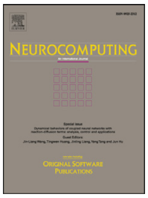
Neurocomputing 275 (2018) 1261–1274

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Incremental on-line learning: A review and comparison of state of the art algorithms

Viktor Losing[a,b,*], Barbara Hammer[a], Heiko Wersing[b]

[a] Bielefeld University, Universitaetsstr. 25, Bielefeld 33615, Germany
[b] HONDA Research Institute Europe, Carl-Legien-Str. 30, Offenbach 63065 Germany

ARTICLE INFO

Article history:
Received 8 July 2016
Revised 10 June 2017
Accepted 24 June 2017
Available online 28 September 2017

Communicated by Prof. Zidong Wang

Keywords:
Incremental learning
On-line learning
Data streams
Hyperparameter optimization
Model selection

ABSTRACT

Recently, incremental and on-line learning gained more attention especially in the context of big data and learning from data streams, conflicting with the traditional assumption of complete data availability. Even though a variety of different methods are available, it often remains unclear which of them is suitable for a specific task and how they perform in comparison to each other. We analyze the key properties of eight popular incremental methods representing different algorithm classes. Thereby, we evaluate them with regards to their on-line classification error as well as to their behavior in the limit. Further, we discuss the often neglected issue of hyperparameter optimization specifically for each method and test how robustly it can be done based on a small set of examples. Our extensive evaluation on data sets with different characteristics gives an overview of the performance with respect to accuracy, convergence speed as well as model complexity, facilitating the choice of the best method for a given application.

1. Introduction

Nowadays, large parts of all conceivable information are collected and stored in digital form accumulating to enormous daily increasing amounts. Every day Google receives 3.5 billion search queries; nearly 2 billion active users of Facebook share 4.5 billion pieces of content; Amazon sells about 13 million items world wide. All kinds of customer information, raw transactional data as well as individual clicking behavior, is collected to provide services such as personalized recommendations. Estimated 35% of Amazons 107 billion dollar net sales are attributed to its recommendation engine. These pioneering companies demonstrated that information can be the central pillar of a multi-billion dollar business. Even small firms adopted this approach and now digitize every transaction they are involved in to boost their turnovers.

Data collection is also done by mobile devices such as mobiles, smart watches and fitness bands continuously tracking various user information as call logs, GPS positions, heart rates and activities. It is omnipresent in science as well: astronomical observatories, earth sensing satellites and climate observation networks generate terabytes of data on a daily basis. Meanwhile, the rate at which data arises rapidly increases further – 90% of all the data in the world has been generated over the last two years.

Machine learning methods are employed to mine the collected data for relevant information and/or to predict future developments by generated models. However, classical batch machine learning approaches in which all data is simultaneously accessed do not meet the requirements to handle the sheer volume in the given time, leading to more and more accumulated data unprocessed. Furthermore, they do not continuously integrate new information into already constructed models but instead regularly reconstruct new models from the scratch. This is not only very time consuming but also leads to potentially outdated models.

Overcoming this state of affair requires a paradigm shift to sequential data processing in streaming scheme. This does not only allow to use information as soon as it is available leading to all-time up to date models, but also reduces the costs for data storage and maintenance.

Incremental and on-line algorithms fit naturally to this scheme, since they continuously incorporate information into their model, and traditionally aim for minimal processing time and space. Due to their ability of continuous large-scale and real-time processing, they recently gained more attention particularly in the context of Big Data [1].

Incremental algorithms are also very suitable for learning beyond the production phase which enables devices to adapt to individual customer habits and environments. This is particularly interesting for smart home products [2,3]. Here, the main challenge

[*] Corresponding author at: Bielefeld University, Universitaetsstr. 25, 33615 Bielefeld, Germany.
E-mail address: vlosing@techfak.uni-bielefeld.de (V. Losing).

https://doi.org/10.1016/j.neucom.2017.06.084
0925-2312/© 2017 Elsevier B.V. All rights reserved.

is not large-scale processing but rather continuous and efficient learning from few data. Even though incremental learning could be replaced in this case by repetitive batch learning in the cloud, this solution has crucial drawbacks. A permanent connection to the cloud is required to provide anytime models, which may not always be feasible. Furthermore, the customers may not be willing to provide data of their daily life due to privacy reasons. Hence, learning directly on the device in an efficient way is still very desirable. A lot of ambiguity is involved regarding the definition of incremental and on-line learning in the literature. Some authors use them interchangeably, while others distinguish them in different ways. Additional terms such as lifelong- or evolutionary learning are also used synonymously. We define an incremental learning algorithm as one that generates on a given stream of training data $s_1, s_2, \dots, s_t$ a sequence of models $h_1, h_2, \dots, h_t$. In our case s_i is labeled training data $s_i = (\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \{1, \dots, C\}$ and $h_i : \mathbb{R}^n \{1, \dots, C\}$ is a model function solely depending on h_{i-1} and the recent p examples s_i, ..., s_{i-p}, with p being strictly limited. We specify on-line learning algorithms as incremental learning algorithms which are additionally bounded in model complexity and run-time, capable of endless/lifelong learning on a device with restricted resources.

Incremental learning algorithms face the following challenges:

- The model has to adapt gradually i.e. h_{i+1} is constructed based on h_i without a complete retraining.
- Preservation of previously acquired knowledge and without the effect of catastrophic forgetting [4].
- Only a limited number of p training examples are allowed to be maintained.

We explicitly assume the data to be labeled and do not focus on the, nonetheless, crucial scenario of learning from un- or partially labeled data streams. The setting of supervised incremental learning can be applied in most prediction scenarios. In these, after a system has made a prediction the true label can often be inferred with some delay. For example, consider the course of action a car driver will take at a crossing. As soon as the car has passed the crossing the recorded data can be analyzed and labeled in an automatic way. The supervised setting also includes tasks in which labels are explicitly provided. For instance, an individual user marks emails as spam for spam classification, but also in human robot interactions the labels may be explicitly demanded.

An algorithm has to be chosen according to the preconditions of a given task since there cannot exist a method which optimally performs in every scenario [5]. Different interesting incremental learning algorithms have been published so far with various strengths and weaknesses. However, there are only a few sources providing information about them, since basically no comparative in-depth study, experimentally comparing the most popular methods according to the most relevant criteria, is available. An extensive research in the literature leads usually to the original publications of considered algorithms which help only to some extent due to the following reasons:

Authors are naturally focused to demonstrate the merits of their method and, therefore, apply them in specific settings (particularly settings the algorithm has been designed for). Proposed algorithms are usually compared against one or two other methods on a few datasets, providing only a limited overall picture of the algorithms qualities. Even if one accepts the effort to reproduce the results, it often turns out to be impossible, because of proprietary datasets or unknown hyperparameters settings. In the end, one has either to pick a method based on the own experience, which usually comprises only a fraction of available algorithms, or simply invest a lot of resources to try out several approaches.

In this paper, we contribute to fill this gap by analyzing the core attributes of eight popular methods. Our study aims for a fundamental comparison of the algorithmic overall performance unrestricted to certain scenarios such as platforms with very limited resources. However, the performance for specific settings can be inferred from the general results provided in this article. We guide the choice for an algorithm based on essential information (e.g. number of dimensions/samples) that is usually available in advance.[1] Our evaluation in off- and on-line setting enables an extensive comparison in terms of accuracy, convergence speed and model complexity. Experiments on diverse datasets assess strengths and weaknesses of the respective methods and provide guidance on their applicability for specific tasks. Furthermore, we analyze the process of hyperparameter optimization (HPO) and investigate how robustly they can be estimated based on a small set of examples.

Our focus lies in the classification under supervised learning for incremental/on-line algorithms. We primarily perform an evaluation on stationary datasets (i.e. we assume the stream $s_1, s_2, \dots$ is i.i.d.). However, we briefly evaluate and discuss the methods in the context of concept drift. A recent overview of methods especially designed to deal with non-stationary environments is given in [6].

This article is organized as follows. In Section 2, we discuss related contributions, in particular those targeting the field of incremental learning in a general way. Section 3 provides a brief description of the considered algorithms. The evaluation framework consisting of an analysis in off-line and on-line scheme is introduced in Section 4. The main part of our work with practical focus can be found in Section 5, which goes into detail about the performed experiments. Here, we analyze the algorithms in different settings and discuss properties such as time efficiency, suitability for lifelong learning, HPO and so forth. Finally, Section 6 briefly summarizes our results and depicts them compressed in tabular form.

2. Related work

Numerous incremental and on-line algorithms have been published, often adapting existing batch methods to the incremental setting [7,8]. Massive theoretical work has been done to evaluate their generalization ability and convergence speed in the stationary setting [9,10], often accompanied by assumptions such as linearly separable data [11].

Although the field of incremental and on-line learning is well established and particularly employed in the context of Big Data or the Internet of Things technology [12], there are only a few publications targeting the field in a general way. Most of these are surveys describing available methods and some domains of applications [13,14].

Giraud-Carrier [15] gives some motivation for incremental learning and define the notion of incrementality for learning tasks. They argue in favor of applying incremental learning methods for incremental tasks but also point to arising issues such as ordering effects or the question of trustworthiness.

One survey was recently published by Gepperth and Hammer [16]. They formalize incremental learning in general and discuss theoretical as well as practical challenges which arise in the setting. Furthermore, an overview of commonly employed algorithms with corresponding real world applications is given.

Incremental learning is more frequently treated in the setting of streaming scenarios [17,18], although most of the work particularly targets concept drift [6,19,20]. Domingos and Hulten define key properties for incremental algorithms which are required to keep

[1] The number of dimensions as well as the amount of incoming data examples can be usually at least estimated. Furthermore, it can be inferred how crucial a quick reaction of the system is. For some tasks, it is even possible to guess whether a linear classifier is sufficient (e.g. text classification).

up with the rapidly increasing rate of data output [21]. They stress the necessity of combining models, strictly limited in terms of processing time and space, with theoretical performance guarantees.

Publications with a practical focus are very rare in the field of incremental learning. One of them was done by Read et al. [22] in the setting of concept drift. Batch-incremental methods with instance-incremental approaches were compared and analyzed in their pros and cons. The reached conclusion is that instance-incremental algorithms are equally accurate but use fewer resources and that lazy methods with a sliding window perform exceptionally well.

A massive study comprising the evaluation of 179 batch classifier on 121 datasets was done by Fernández-Delgado et al. [23]. This quantitative study considered also different implementations in varying languages and toolboxes. The best result was achieved by the Random Forest [24] algorithm closely followed by the Support Vector Machine (SVM) [25] with Gaussian kernel.

However, such work is still sorely missed for incremental algorithms. In this paper, we pursue a more qualitative approach and instead of a massive comparison, provide an in depth evaluation of the major approaches within stationary environments. Next to the accuracy, we also inspect the model complexity which allows an inference of required resources in terms of time and space. The consideration of rather neglected aspects such as convergence speed and HPO rounds off our analysis.

3. Algorithms

Our comparison of methods covers a broad range of algorithm families. Bayesian, linear, and instance-based models as well as tree-ensembles and neural networks are represented. Model-dependent methods such as the Incremental Support Vector Machine are denoted by an acronym (SVM), whereas model-independent methods as Stochastic Gradient Descent are denoted by an acronym with an additional index (SGD_{Lin}), specifying the applied model. In the following the methods are briefly described.

Incremental Support Vector Machine (ISVM) is the most popular exact incremental version of the SVM and was introduced in [7]. Additionally to the set of support vectors a limited number of examples, so called “candidate vectors”, is maintained. These are examples which could be promoted to support vectors depending on the future examples. The smaller the set of candidate vectors is, the higher is the probability of missing potential support vectors. The ISVM is a *lossless* algorithm – it generates the same model as the corresponding batch algorithm – if the set of candidate vectors contains all previously seen data. Recent applications can be found in [26,27].

LASVM is an online approximate SVM solver and was proposed in [28]. In an alternative manner, it checks whether the currently processed example is a support vector and removes then obsolete support vectors. For both steps, it heavily utilizes sequential direction searches as it is also done in the Sequential Minimal Optimization (SMO) algorithm [29]. In contrast to the ISVM, it does not maintain a set of candidate vectors but only considers the current example as possible support vector. This leads to an approximate solution but significantly reduces the training time. It was recently applied in [30,31].

On-line Random Forest (ORF) [32] is an incremental version of the Random Forest algorithm. A predefined number of trees grows continuously by adding splits whenever enough samples are gathered within one leaf. Instead of computing locally optimal splits, a predefined number of random values are tested according to the scheme of Extreme Random Trees [33]. The split value optimizing the Gini index the most is selected. Tree ensembles are very popular, due to their high accuracy, simplicity and parallelization capability. Furthermore, they are insensitive to feature scaling and can be easily applied in practice. This method has been lately applied in [34,35].

Incremental Learning Vector Quantization (ILVQ) is an adaptation of the static Generalized Learning Vector Quantization (GLVQ) [36] to a dynamically growing model, which inserts new prototypes when necessary. The insertion rate is guided by the number of misclassified samples. We use the version in [37] which introduced a prototype placement strategy minimizing the loss on a sliding window of recent samples. Metric learning, as described in [38,39], can also be applied to extend the classification abilities further.

Learn++ (LPP_{CART}) [40] processes incoming samples in chunks with a predefined size. For each chunk an ensemble of base classifiers is trained and combined through weighted majority voting to an “ensemble of ensembles”. Similar to the AdaBoost [41] algorithm, each classifier is trained with a subset of chunk examples drawn according to a distribution, ensuring a higher sample probability for misclassified inputs. LPP is a model independent algorithm and several different base classifiers such as SVM, Classification and Regression Trees [42] (CART) and Multilayer Perceptron [43] have been successfully applied by the authors. As the original author we employ the popular CART as base classifiers. Chunk-wise trained models inherently incorporate an adaption delay depending on the chunk size. This algorithm was recently utilized in [44,45].

Incremental Extreme Learning Machine (IELM) reformulates the batch ELM least-squares solution into a sequential scheme [8]. As the batch version it drastically reduces the training complexity by randomizing the input weights. The network is static and the number of hidden neurons has to be predefined. This method is able to process the data one-by-one or in chunks, which significantly reduces the overall processing time. However, a valid initialization of the output weights requires at least as many examples as the number of used hidden neurons. Recent applications are given in [46,47].

Naive Bayes (NB_{Gauss}) fits one axis-parallel Gaussian distribution per class and uses them as likelihood estimation in the Naive Bayes algorithm [48]. The sparse model allows a very efficient learning in terms of processing time and memory requirements. This algorithm learns efficiently from few training examples [49] and has been successfully applied in real world situations such as Spam filtering and document classification[2] [50,51]. The major drawbacks of this lossless algorithm are the independence assumption of the features as well as its inability to handle multimodal distributions. This method was recently used in [52,53].

Stochastic Gradient Descent (SGD_{Lin}) is an efficient optimization method for learning a discriminative model by minimizing a loss function such as the Hinge – or Logistic loss. We use SGD to learn a linear model by minimizing the Hinge loss function. Revived recently in the context of large-scale learning [54–56], SGD coupled with linear models performs especially well for sparse, high-dimensional data as often encountered in the domain of text classification or natural language processing. However, linear models are a misfit whenever non-linear class boundaries are required, which is par-

[2] In the context of features based on text, the Naive Bayes algorithm is usually applied with the multinomial or Bernoulli event model.

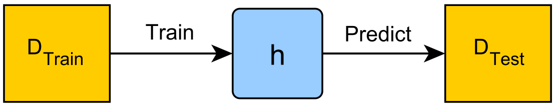


Fig. 1. Classical scheme of evaluating a batch algorithm in off-line mode.

ticularly often the case for low dimensional data. Recent applications can be found in [57,58].

Even though new versions of the algorithms are continuously proposed, we argue that the chosen methods reflect the general properties of the respective family. Therefore, the conclusions in this paper are commonly applicable for current and upcoming variations of the corresponding algorithm. This is particularly highlighted by both SVMs which perform very similar with the difference that LASVM is able to process slightly larger datasets due to its approximate nature. However, both share the same drawbacks regarding large or noisy datasets. These drawbacks are also shared by a recent LASVM version proposed in [59], albeit in a slightly weaker degree since a mechanism to reduce the number of support vectors is introduced. Various extensions for the LPP [60,61] and the IELM [62,63] algorithm have been proposed. Most of them are tackling non-stationary environments by introducing forgetting mechanisms. However, the major focus of this article is incremental learning in stationary environments where forgetting is rather harmful and deteriorates the performance. Furthermore, the basic principle of the algorithms and the corresponding ad- and disadvantages remain. In the case of LPP, it is the flexibility of arbitrary base classifiers on the one hand, and the limited knowledge integration across chunks on the other. Methods for speeding up the convergence of SGD were presented in [55,64]. However, the results obtained by the SGD algorithm in our experiments are not due to a slow convergence of the SGD algorithm, but rather highlight the general benefits and limitations of linear models, such as a low model complexity and linear class boundaries.

4. Evaluation framework

The learning objective in supervised classification is to predict a target variable $y \in \{1, \dots, c\}$ given a set of features $\mathbf{x} \in \mathbb{R}^n$. We consider two different evaluation settings which allow the inference of different aspects regarding the algorithmic performance and provide together even a deeper insight.

4.1. Off-line setting

In the *off-line* setting a batch algorithm generates a model h based on a training set $D_{\text{train}} = \{(\mathbf{x}_i, y_i) | i \in \{1, \dots, j\}\}$. In the subsequent test phase the model is applied on another set $D_{\text{test}} = \{(\mathbf{x}_i, y_i) | i \in \{1, \dots, k\}\}$, whose labels are kept hidden. Fig. 1 depicts the process. The model predicts a label $\hat{y}_i = h(\mathbf{x}_i)$ for every point $x_i \in D_{\text{test}}$ and the 0–1 loss $\mathcal{L}(\hat{y}_i, y_i) = \mathbb{1}(\hat{y}_i \neq y_i)$ is calculated. The average accuracy on the test set enables an analysis in terms of the generalization ability to unseen examples. The evaluation of an incremental algorithm in this setting is different as it is shown by Fig. 2. Instead of accessing all training data at once, it is sequentially processed in predefined order. The algorithm generates to the sequence of tuples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots (\mathbf{x}_j, y_j)$ a corresponding sequence of models $h_1, h_2, \dots, h_j$. Thereby, a model h_i is solely based on the previously constructed model and a limited amount of p recent tuples

$$h_i = \text{train}(h_{i-1}, (\mathbf{x}_i, y_i), \dots, (\mathbf{x}_{i-p+1}, y_{i-p+1})).$$

Only the last model h_j is applied on the test set to determine the *off-line* accuracy ξ

$$\xi(D_{\text{test}}) = \frac{1}{k}\sum_{i=1}^{k} 1 - \mathcal{L}(\hat{y}_i, y_i) = \frac{1}{k}\sum_{i=1}^{k} 1 - \mathcal{L}(h_j(\mathbf{x}_i), y_i).$$

Hence, this setting allows only an inference about the generalization ability of the last model and neglects all preceding models. Such an evaluation is useful in Big Data scenarios, for example, where a lot of training data is available to continuously construct a model as accurate as possible.

4.2. On-line setting

Data stream classification is usually evaluated in the *on-line* setting, which is depicted in Fig. 3. A potentially infinite sequence $S = (s_1, s_2, \dots, s_t, \dots)$ of tuples $s_i = (\mathbf{x}_i, y_i)$ arrives one after another. As t represents the current time stamp, the learning objective is to predict the corresponding label y_t for a given input x_t, which is supposed to be unknown. The prediction $\hat{y}_t = h_{t-1}(\mathbf{x}_t)$ is done according to the previously learned model h_{t-1}. Afterwards, the true label is revealed and the loss $\mathcal{L}(\hat{y}_t, y_t)$ determined. The *on-line* accuracy for a sequence up to the current time t is given by

$$E(S) = \frac{1}{t}\sum_{i=1}^{t} 1 - \mathcal{L}(h_{i-1}(x_i), y_i). \quad (1)$$

The main difference to the previous setting is that all intermediate models are considered for the performance evaluation, but each of them predicts only the following example. Additionally, the data for training and testing is not strictly disjunct, but instead each instance is initially used for model testing and then for the adaption.

Regarding non-stationary data, a high *on-line* accuracy does not necessarily imply a high generalization ability of the models. For instance in case of strong auto-correlation of the labels, an algorithm simply predicting the previous label achieves accurate results without learning any structure in the data. However, for i.i.d. data the *on-line* accuracy of an incremental algorithm is in

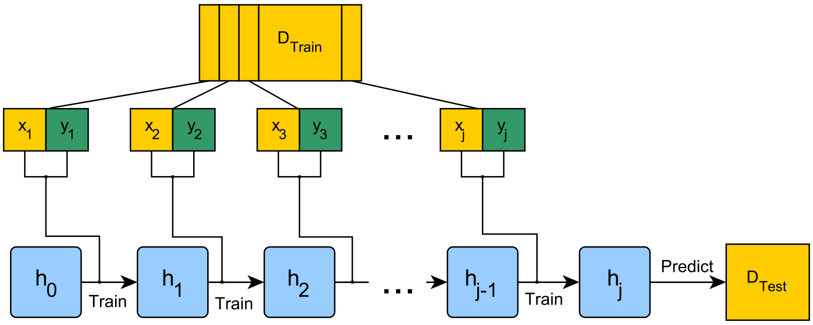


Fig. 2. The process of testing an incremental algorithm in the off-line setting. Noticeably, only the last constructed model is used for prediction. All data used during training $(\mathbf{x}_i, y_i)$ is obtained from the training set D_{train}.

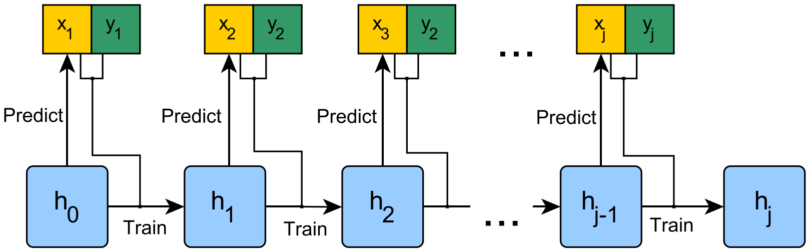


Fig. 3. The online-learning scheme. Data is not split into training- and testing set. Instead, each model predicts subsequently one example, which is afterwards used for the construction of the next model.

Table 1
The evaluated datasets and their characteristics.

Dataset	#Train	#Test	#Feat.	#Class
Border	4000	1000	2	3
Overlap	3960	990	2	4
Letter	16,000	4000	16	26
SUSY	4,500,000	500,000	18	2
Outdoor	2600	1400	21	40
COIL	1800	5400	21	100
DNA	1400	1186	180	3
USPS	7291	2007	256	10
Isolet	6238	1559	617	26
MNist	60,000	10,000	784	10
Gisette	6000	1000	5000	2

general correlated with the average generalization ability of all constructed models.

The *on-line* accuracy is a reasonable evaluation measure for tasks requiring an immediate prediction even after a few training examples.

The combination of both accuracies, off- and on-line, enables conclusions about the learning curve: in case of two different models *A*, *B* having the same *off-line* accuracy, but *A* having a higher *on-line* accuracy implies that *A* converges on average faster than *B* and vice versa.

5. Experiments

In this chapter, we precisely describe how the experiments were conducted. This includes the evaluated datasets, the process of HPO and the different settings in which the algorithms were compared. Furthermore, we discuss whether methods are capable of lifelong learning and give a brief analysis of their training and run-time complexities.

5.1. Datasets and implementations

We used the implementations of the Scikit-learn package [65] for SGD_{Lin} and NB_{Gauss}. All the others are derived from the code of the respective authors. Only publicly available datasets (see [66,67]), predefining a fixed train-test-split, were used to enable reproducibility and comparability of our results. Table 1 gives the main attributes of the selected datasets.

Artificial and real world problems are included, differing widely in the number of classes, instances and dimensions. Even though the largest data set has about 4.5 million instances, our evaluation does not specifically target learning from big data. Instead, our focus is the practical evaluation of incremental learning algorithms in terms of different key properties. Sources for all implementations and datasets are available at https://github.com/vlosing/Online-learning.

Table 2
All relevant hyperparameters of the considered algorithms. The most critical are those which cannot be generally chosen and, therefore, require a task specific setting.

	Hyperparameter	Task independent
SVMs	Kernel function	√
	RBF σ	✗
	Regularization	✗
	# stored candidate vectors (only ISVM)	√
ORF	Growing speed	✗
	# evaluated random splits	√
	# trees	√
ILVQ	Learning rate	✗
	Growing speed	✗
	Window size	√
	(Metric learning rate)	(✗)
LPP_{CART}	Chunk size	✗
	# base classifier per chunk	√
	(Parameter of base classifier)	(✗)
IELM	Activation function	√
	# hidden nodes	✗
NB_{Gauss}	None	
SGD_{Lin}	Loss function	√
	Learning rate	✗

5.2. Hyperparameter optimization

The model selection is varying in complexity depending on the parameter amount and type. Table 2 gives an overview of all relevant hyperparameters.

The most crucial parameters are those adjusting the scale such as learning rates or σ of the RBF kernel. These do not only affect the achieved accuracy, but also strongly influence the overall model complexity. For instance, an inappropriately chosen σ can increase the number of support vectors quite drastically. A wrongly set learning rate of the ILVQ leads to more errors during training and, therefore, more inserted prototypes.

Some models allow to control the speed of model expansion directly such as the ILVQ and ORF. This does not only affect the model complexity but also influences the convergence rate as well as the achieved accuracy and may lead to overfitting when a too aggressive growth is set. Rather uncritical are parameters increasing the leeway of an algorithm. Larger values are in this case always beneficial for the performance and only limited by the amount of available resources. The number of trees of the ORF or the window size of the ILVQ are such parameters. Generally speaking, tree based models are easy to tune and perform usually well out of the box, whereas scale sensitive models such as ISVM, LASVM or ILVQ require an accurate, dataset dependent configuration of multiple parameters to deliver good results.

Both SVM algorithms are solely paired with the RBF kernel. We use the metric learning of ILVQ only for datasets with up to 250 dimensions (the distance calculations using the metric is quadratic in the number of dimensions and hence not feasible for very high

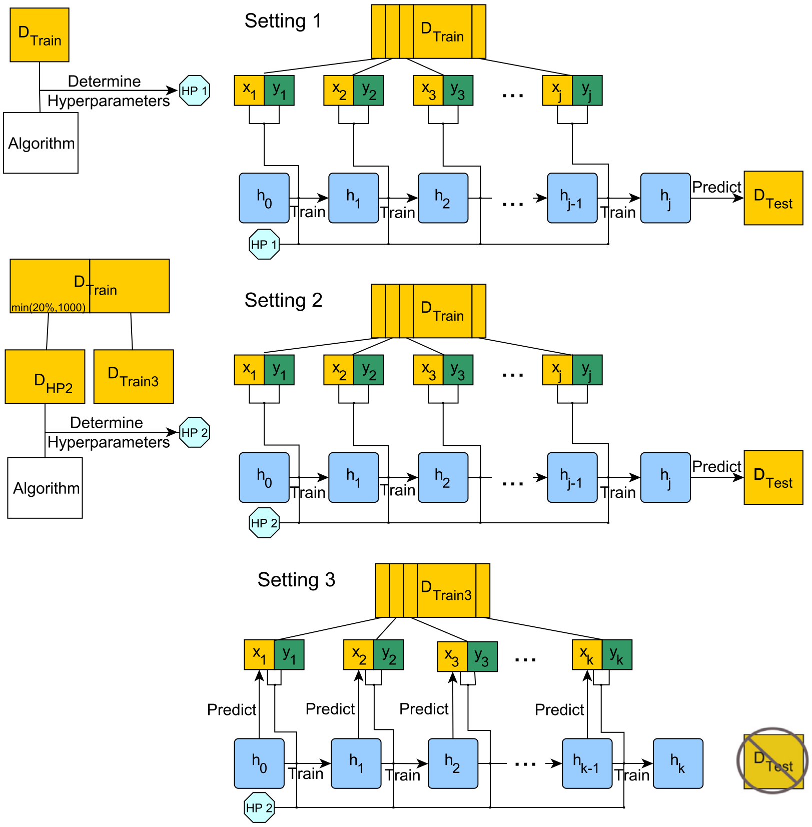


Fig. 4. The first setting determines the hyperparameters by using the whole training set for optimization. It evaluates then the off-line accuracy on the test set. The second setting uses a small subset of the training set to determine the hyperparameters. In the third setting, the same hyperparameters are used as in the second one, but here, we evaluate the on-line accuracy on the training set (samples that are used in the HPO are excluded). The test set is not used in the third setting.

dimensional data). The NB_{Gauss} algorithm is parameterless, hence no tuning is required at all. We minimize the hinge loss function with SGD_{Lin} and adjust only the learning rate. LPP_{CART} requires the number of base classifier per chunk as well as the parameters of the base classifier itself (non-parametric Classification and Regression Trees in our case).

All parameter are set by Hyperopt [68] using the Tree-of-Parzen-Estimators [69] search algorithm. Each parameter is individually adjusted by performing 250 iterations of a 3-fold cross validation using only the training data.

5.3. Measure of model complexity

We measure the model complexity by the number of parameters required for the representation, enabling a comparison with respect to memory consumption. However, the models are fundamentally different so that this measure, even though there is some correlation, should not generally be equated with training- or run-time. We rather use this measure as an indicator to decide whether an algorithm struggles (unreasonable high amount of parameters) or is especially suited (sparse representation paired with high accuracy) for a given task.

5.4. Evaluation settings

We evaluate the algorithms in three different scenarios as it is illustrated by Fig. 4. During the first we compare them in classical off-line scheme and use the complete training set for the HPO. This allows a conclusion about the generalization ability of the final model. However, the usage of the whole training set for the HPO is usually not possible in practical applications and contradicts the paradigm of incremental learning. Therefore, we optimize the parameters only with a proportion of the training examples. This is not only closer to practice but also, in combination with the results of the first setting, enables to infer whether the hyperparameters of a corresponding method can be reliably estimated on a subset of the data. Since the number of training examples vary considerably across the datasets, we decided to use a relative proportion bounded by a maximum number of examples. Precisely, we use 20% of the training data for HPO but never more than 1000 examples. The last evaluation uses the hyperparameters of the second scenario, but examines the methods in the on-line setting. Here, we draw conclusions about the learning curves of the respective algorithms. To keep the number of training instances similar among all evaluations, we use only the training set (samples that are used in the HPO are excluded) as data stream in the on-line setting.

5.5. Results

The evaluation of LASVM, NB_{Gauss}, ORF and SGD_{Lin} is straightforward, since these access consecutively only the current training example. But methods such as ISVM and ILVQ store additionally a window of recent samples or require chunk-wise training, as LPP_{CART} and IELM[3] do. In both cases, results depend on the window-/chunk size. Therefore, we treated the window-/chunk size as another hyperparameter and used once again Hyperopt to

[3] IELM requires for the initialization at least as many samples as it has hidden neurons but afterwards it can be updated after each sample.

Table 3
Off-line accuracy (left) and model complexity (right) after training, measured by the overall number of parameters, averaged over 10 repetitions. In the first setting, the Hyperparameters were optimized using the whole training data, whereas only a small subset of it was used in the second setting. The processing was canceled whenever it took longer than 24 h and we mark the corresponding experiments as DNF. We calculated two different rankings. The first is the average rank based on all datasets the algorithms were able to deliver a result. The second rank ($\widehat{\text{rank}}$), however, punishes algorithms with DNF entries. In this case, they are ranked as the last in the respective dataset.

Setting 1	Off-line accuracy								Complexity							
	ISVM	LASVM	ORF	ILVQ	LPP_{CART}	IELM	SGD_{Lin}	NB_{Gauss}	ISVM	LASVM	ORF	ILVQ	LPP_{CART}	IELM	SGD_{Lin}	NB_{Gauss}
Border	**99.4**	98.4	97.6	96.8	96.5	96.0	35.5	96.4	797	251	3.7k	193	1.9k	750	**9**	12
Overlap	82.2	81.0	**83.7**	83.0	81.7	83.4	67.6	66.4	10k	7.2k	2.3k	235	1.9k	900	**12**	16
Letter	97.0	**97.1**	93.2	93.9	87.0	70.0	56.4	63.4	131k	140k	168k	14k	51k	8.4k	**442**	832
SUSY	DNF	DNF	**79.5**	79.0	DNF	DNF	78.7	73.5	DNF	DNF	86.0	51.2	DNF	DNF	**19**	72
Outdoor	70.9	**71.4**	71.0	66.9	68.5	70.9	23.2	60.5	43k	42k	8.8k	11k	6.2k	12k	**880**	1.7k
COIL	**96.5**	93.2	92.9	94.3	89.2	91.5	12.4	90.0	58k	93k	61k	18k	9.2k	36k	**2.2k**	4.2k
DNA	**94.9**	**94.9**	89.6	92.1	90.5	88.8	93.0	88.4	237k	190k	5.0k	33k	1.6k	55k	**543**	1.1k
USPS	95.4	**95.6**	92.5	91.4	90.3	92.1	89.0	75.4	710k	520k	33k	15k	9.6k	106k	**2.6k**	5.1k
Isolet	96.2	**96.7**	92.5	92.0	90.0	91.9	91.5	80.1	2.8M	3.4M	31k	21k	**12.0k**	322k	16k	32k
MNist	DNF	**98.5**	94.3	94.8	92.4	89.1	86.0	55.6	DNF	14M	111k	315k	73k	397k	**7.9k**	16k
Gisette	**98.0**	97.9	94.6	93.0	94.2	91.4	93.1	75.4	7.0M	6.2M	4.7k	263k	**2.5k**	2.5M	5.0k	20k
∅	92.3	**92.4**	89.2	88.9	88.0	86.5	66.0	75.0	1.2M	2.5M	54k	64k	17k	344k	**3.2k**	7.3k
∅ rank	**1.9**	2.0	3.0	3.7	5.3	5.3	6.3	6.8	7.1	7.0	5.1	4.0	3.4	5.3	**1.3**	2.5
∅ $\widehat{\text{rank}}$	2.7	**2.3**	3.0	3.7	5.3	5.3	6.3	6.8	7.0	6.8	5.1	4.0	3.5	5.3	**1.3**	2.5
Setting 2																
Border	**99.4**	98.8	96.3	97.1	96.4	96.2	33.7	96.4	1.1k	1.5k	8.0k	286	2.6k	875	**9**	12
Overlap	81.8	80.7	81.2	82.1	81.0	**82.2**	68.4	66.4	10.9	9.2 k	31k	496	1.9k	546	**12**	16
Letter	96.4	**97.5**	92.2	95.1	84.9	67.4	53.9	63.4	163	173k	473k	11k	70 k	6.5k	**442**	832
SUSY	DNF	DNF	**79.3**	73.4	DNF	DNF	77.0	73.5	DNF	DNF	1.5M	453k	DNF	DNF	**19**	72
Outdoor	**71.9**	69.0	57.6	63.2	69.4	71.6	25.7	60.5	40k	53k	925	11 k	7.3k	8.3k	**880**	1.7k
COIL	**95.6**	92.5	89.9	93.6	88.1	83.6	12.1	90.0	180k	113k	14k	17 k	6.1k	15k	**2.2k**	4.2k
DNA	**95.2**	95.1	88.2	90.8	92.0	86.7	93.3	88.4	333k	327k	500	42k	1.8k	76k	**543**	1.1k
USPS	**95.6**	94.7	91.4	91.4	88.9	89.7	89.8	75.4	744k	978k	61k	91k	12k	50k	**2.6k**	5.1k
Isolet	**96.5**	96.4	92.0	89.4	90.7	88.5	89.6	80.1	6.2M	4.2M	123k	67k	**14k**	159k	16k	32k
MNist	DNF	**98.1**	95.0	94.3	91.3	84.2	87.7	55.6	DNF	16M	253k	1.2M	162k	169k	**7.9k**	16k
Gisette	97.8	**97.9**	92.7	94.1	94.0	83.8	95.4	75.4	7.0M	6.7M	16k	470k	**3.0k**	1.0M	5.0k	20k
∅	**92.2**	92.1	86.9	87.7	87.7	83.4	66.1	75.0	1.6M	2.9M	226k	216k	28k	151k	**3.2k**	7.3k
∅ rank	**1.4**	2.4	4.4	3.6	4.7	5.6	5.7	6.4	7.1	7.0	4.8	4.6	3.6	4.7	**1.3**	2.4
∅ $\widehat{\text{rank}}$	**2.4**	2.6	4.4	3.6	4.7	5.5	6.7	6.4	7.0	6.8	4.8	4.6	3.7	4.7	**1.3**	2.5

find the best value. We allowed a maximum size of 500 samples. All methods were trained single-pass, in the same order after initial shuffling.

5.5.1. *Off-line setting – HPO with all training samples*

Table 3 shows on the top the accuracies and corresponding model complexities at the end of training. Both SVMs achieve on average the highest accuracy, often with a large margin, but at the expense of having by far the most complex models. The large amount of parameters is partly due to the fact that the model is designed to discriminate two classes, resulting in multiple SVMs to perform schemes such as one vs. all in case of more classes. Another reason is the linear growth of support vectors with the amount of samples. The model gets exceedingly complex for noisy or overlapping datasets such as *Isolet* or *Overlap*. The SVMs deliver very similar results and mainly are different in terms of their training run-time. The high training-complexity of ISVM, resulting from the computation and incremental update of the inverse kernel matrix, prevents an application for datasets consisting of substantially more than 10000 samples such as *MNist*. The approximate nature of LASVM allows it to process the *MNist* dataset but it also reaches its limit for significantly larger datasets as *SUSY*. The instance based ILVQ constructs a far sparser model and achieves high accuracies throughout all datasets. It handles efficiently noisy datasets by sustaining its sparse model.

As expected, tree based models require a comparably large amount of parameters for low dimensional data, but are eminently efficient in high dimensional spaces, due to their compressing representation. The opposite is true for instance based models.[4]

The ORF has the third highest accuracies and constantly beats LPP_{CART}. One explanation, already noticed in [70], is that LPP_{CART} trains each base classifier with samples of only one chunk. Therefore, the knowledge integration across chunks is limited since it is exclusively established by the weighting process. Furthermore, the ORF benefits more from the sub-linear tree complexity because it generates a few deep trees instead of the numerous, shallow ones as done by LPP_{CART}. In contrast to the SVMS, which were not able to process the large dataset SUSY due to algorithmic aspects, were IELM and LPP_{Cart} only limited by their specific implementation.

The linear model of SGD_{Lin} uses the fewest parameters and performs especially well for high dimensional data. However, it struggles by design with non-linear separability as it is the case for the *Border* dataset, or whenever a small amount of examples is available per class (*COIL, Outdoor*). The last rank of NB_{Gauss} obscures the fact that it performs reasonably well without severe hiccups, incorporating a simple and sparse model. Nonetheless, the restriction to unimodal distributions is reflected in the results of the *MNist* and *Isolet* datasets.

The typical effects of different window-/chunk sizes are shown in Fig. 5 exemplary for the *Overlap* dataset. Usually the algorithms do benefit from an increased window-/chunk size. For instance, a larger window enables the ILVQ to find better positions for new

[4] The number of parameters for instance based models can often be clearly reduced with a sparse representation for sparse high-dimensional data as *MNist*. However, our results rely on a dense vector representation.

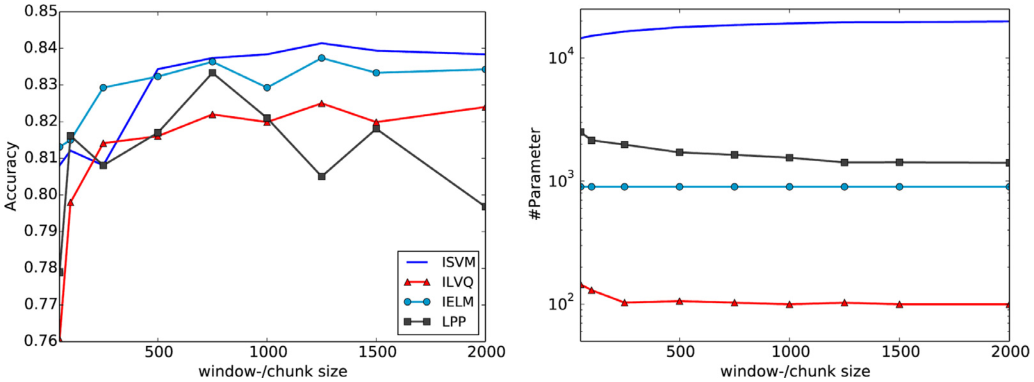


Fig. 5. Influence of the window-/chunk size on the off-line accuracy (left) and model complexity (right) for dataset *Overlap*.

prototypes and the ISVM to miss less support vectors. Simultaneously, the model complexity of ILVQ is reduced since the insertion rate is coupled with the training-error. The IELM benefits from large chunks due to a more stable initialization of the output weight matrix. In case of LPP_{CART}, however, larger chunks reduce the overall number of base classifier, but at the same time each of them is trained on more training data, requiring a balancing of these criteria.

5.5.2. Off-line setting – HPO with a small set of training samples

An overview of the achieved performance in terms of accuracy and model complexity is given at the bottom of Table 3. The results of NB_{Gauss} are only reported for the sake of consistency, since it incorporates no meta parameters and consequently achieves similar results.

Regarding the accuracy, most methods perform slightly worse than in the first setting, leading to the conclusion that hyperparameters can be robustly chosen based on few samples. However, the method losing the most performance is the IELM. This can be explained by the drastically sparser constructed model which is sufficient for the classification of a few examples but not complex enough for the whole dataset. Hence, the number of hidden neurons is underestimated in the optimization.

By contrast, all dynamically growing models tend to use significantly more parameters for various reasons: the kernel width σ of the SVMs is estimated less accurate with few examples leading to an increased number of support vectors. In case of the ILVQ and ORF, the growth is explicitly controlled by a meta parameter. Here, the required rate is overestimated because the model is obliged to converge faster when few instances are available. This leads to a more complex model than necessary and can even end up in overfitting. One solution could be to adjust the growth rate during learning guided by a supervised signal, e.g. the current accuracy.

SGD_{Lin} is the only algorithm which incorporates hyperparameters and achieves, nonetheless, similar results as in the first evaluation. Its model complexity is exclusively determined by the number of dimensions and the amount of different classes in the dataset. The only considered parameter, the learning rate, is reliably estimated on a subset of the data.

5.5.3. On-line setting – same hyperparameters as in setting 2 (Section 5.5.2)

The resulting on-line accuracies are given by Table 4.

In general, the on-line accuracies are slightly lower accounting for the relatively high number of false classifications done at the beginning of learning. The SVMs maintain also in this setting the upper hand, albeit with less dominance. Tree based methods in particular lose the most performance, indicating that the construction of an accurate tree model requires distinctly more examples than an instance based one. This is due to the fact that split nodes are only added when they are necessary for the classification of the data seen so far. A few training examples can already be differentiated along one or two dimensions. Sophisticated tree models consisting of multiple splits are only required for larger amounts of training data.

In contrast, instance based methods immediately classify examples along every dimension. Fig. 6 highlights the different adaption rates between both model types by depicting exemplary learning curves.

The on-line accuracy is expected to be slightly below the off-line accuracy for i.i.d. data because more mistakes are made at the beginning. However, in case of the *Outdoor* dataset the algorithms have partly a 20% higher on-line accuracy. Fig. 7 depicts the learning curves in both settings. The only explanation for this discrepancy is that data in the training set is quite different from those in the test set, implying a non identical and independent distribution. As noted in [37] this visual dataset consists of objects recorded outdoors. The lighting conditions significantly vary within the dataset regarding the respective object and affect the underlying color based representation.

5.6. Restriction of the overall classifier complexity

Methods as SGD_{Lin}, NB_{Gauss} and IELM are on-line algorithms and viable in endless learning applications, since they are constant in their complexity. ILVQ and LPP_{CART} can be easily restricted to a certain limit by strategies such as removing the “worst” prototype/classifier [71,72]. In case of the SVMs, however, it is less trivial. Even though approaches as [73] do reduce the number of support vectors, there is to the best of our knowledge no practical method to bound them strictly. This applies to a lesser degree also for the ORF. It learns by growing its trees continuously. Therefore, a depth reduction or pruning mechanism would be necessarily at some point.

5.7. Training- and run-time

The algorithm implementations vary in the written programming languages as well as their efficiency. For instance, the fastest method NB_{Gauss}, written in C, required four seconds for the *Isolet* dataset while the slowest method the ISVM, implemented in Matlab, took ten minutes. Simply measuring the run time results not in a fair comparison, since the impact of the specific implementation is unclear. Therefore, we do not explicitly compare

Table 4
On-line accuracy averaged over ten repetitions. The on-line accuracy uses each example of a given input stream first for testing and afterwards for model construction (see Eq. (1)). We used the hyperparameters of the second off-line setting, which are optimized on a small set of training examples (see Section 5.5.2). The model complexity is neglected because it is similar to those of the second off-line setting, due to the same hyperparameters. Only the training set of the original data was utilized as input stream (samples that were used in the HPO are excluded). We calculated two different rankings. The first is the average rank based on all datasets the algorithms were able to deliver a result for. The second ranking ($\widehat{\text{rank}}$), punishes algorithms with DNF entries. In this case, they are ranked as the last in the respective dataset.

Setting 3	On-line accuracy							
	ISVM	LASVM	ORF	ILVQ	LPP_{CART}	IELM	SGD_{Lin}	NB_{Gauss}
Border	**98.5**	97.6	94.0	94.7	88.4	88.0	37.5	94.4
Overlap	**81.7**	78.8	78.2	81.1	72.7	74.8	67.9	67.5
Letter	91.3	**92.7**	75.4	88.4	79.3	35.4	41.0	64.2
SUSY	DNF	DNF	**79.3**	78.5	DNF	DNF	78.7	73.5
Outdoor	**86.4**	82.3	34.2	82.6	68.5	73.3	18.0	65.0
COIL	75.4	66.3	66.6	**79.1**	58.7	63.1	9.6	70.2
DNA	**89.5**	**89.5**	73.1	84.6	67.9	49.1	84.7	86.1
USPS	**96.7**	96.6	84.5	92.7	86.6	88.8	88.5	76.0
Isolet	**93.6**	92.9	69.2	84.7	76.3	80.7	74.3	75.2
MNist	DNF	**97.5**	87.1	90.8	89.0	86.5	83.7	56.5
Gisette	96.3	**96.4**	90.3	91.1	86.7	80.5	92.1	74.0
$\varnothing$	**89.9**	89.1	75.6	86.3	77.4	72.0	61.4	73.0
$\varnothing$ rank	**1.3**	2.1	5.1	2.7	5.5	5.8	6.0	5.7
$\varnothing$ $\widehat{\text{rank}}$	**2.7**	2.8	4.7	3.2	5.0	5.5	5.8	5.8

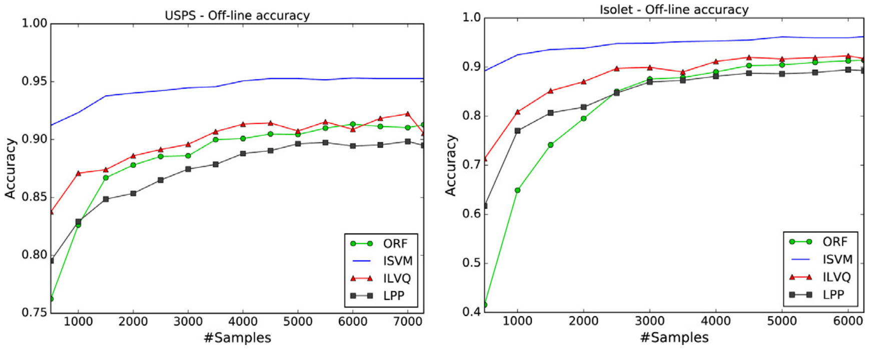


Fig. 6. Learning curves of tree – and instance based models in comparison. Instance based methods are particularly at the beginning more accurate and converge faster.

Table 5
The evaluated datasets and their characteristics.

Dataset	# Instances	#Feat.	#Class	Drift type
Inter. RBF	200,000	2	15	Abrupt real
Electricity	45,312	5	2	Unknown
Moving RBF	200,000	10	5	Incremental real
Cover type	581,012	52	7	Unknown

training- and run-time but instead give a broad categorization based on complexity analysis and practical experience.

The training of both SVMS take by far the most time since a quadratic programming problem is solved. However, LASVM is due to its approximate manner significantly faster than ISVM but has the same worst case complexity. Clearly faster is LPP_{CART} and since we use it in combination with CART its complexity is $\mathcal{O}(n\log(n))$, with n being the number of training examples. By performing the training chunk-wise, n is kept small and the training time is significantly reduced. The ORF has the same complexity class but the random splits drastically reduce the time in practice. The ILVQ and IELM have a similar training complexity $\mathcal{O}(np)$, where p is the number of prototypes/hidden neurons and usually $p \ll n$. However, the insertion of new prototypes in ILVQ requires additional calculations slowing it noticeably down. SGD_{Lin} and NB_{Gauss} are clearly the quickest with linear complexity $\mathcal{O}(n)$. In general, the train- and run time of growing models (LASVM, ISVM, ILVQ, LPP, ORF) naturally increase with model size, affecting the processing time, particularly, for large datasets. The run time of tree based methods is sub-linear in regard to the model size $\mathcal{O}(\log l)$, l being the number of leaves, which makes them extremely efficient. All remaining models have a linear relation between model complexity and run time. Nonetheless, the sparse models of SGD_{Lin} and NB_{Gauss} are usually the fastest in the field.

5.8. Concept drift

Learning from data streams in non-stationary environments is a crucial part of incremental learning. Various algorithms have been published explicitly tackling this challenge [74–76]. It is typically distinguished between *real* drift, referring to a changing class posterior distribution, and *virtual* drift, implying only a varying input distribution. These types of changes can occur in an *abrupt* or *incremental* way. A more extensive categorization is given in [20]. We exemplary investigate the robustness of the methods to different types of real concept drift, as a practically important scenario in practice. Fig. 8 illustrates the setting of the experiments. We optimized the meta parameters using the first 1000 instances and performed an evaluation in the on-line setting on the remaining instances.

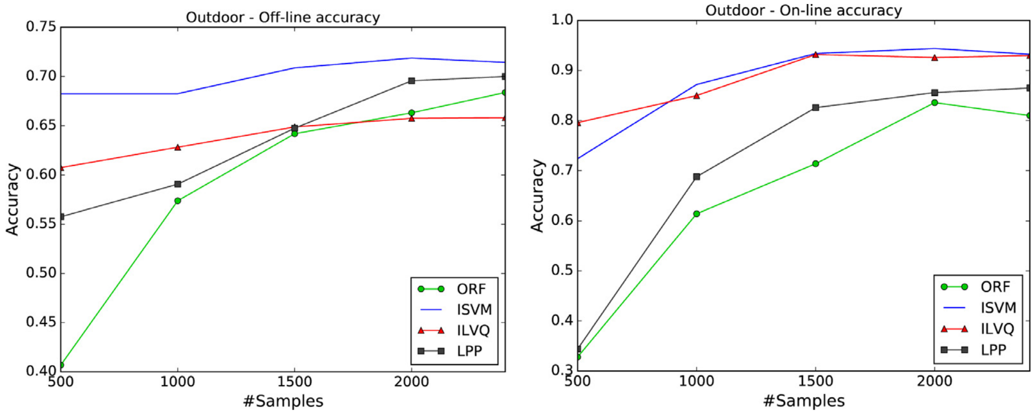


Fig. 7. Learning curves for the outdoor dataset in the off- and on-line setting. The dramatic discrepancy is subject to distinctly different training examples compared to those of the test set, implying the overall data of being not i.i.d.

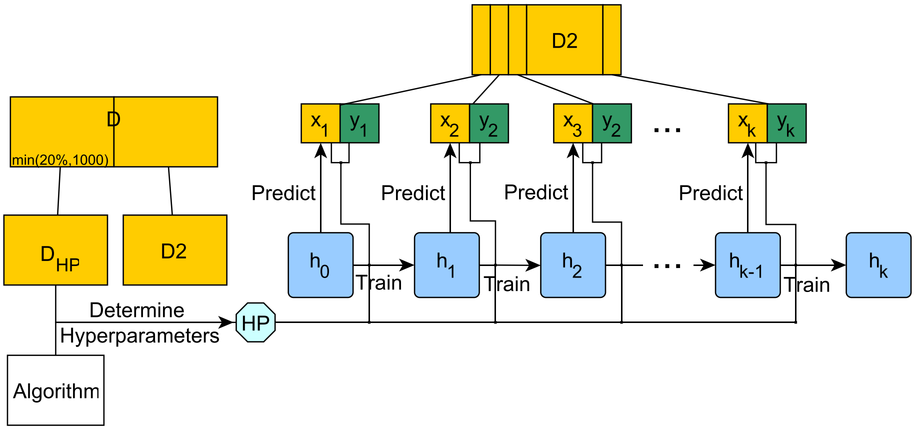


Fig. 8. The concept drift experiments are using streaming datasets, which have a predefined order and there is no splitting into train- and test examples. The evaluation is performed in the on-line setting and the first 1000 samples are used for HPO.

Table 6
Achieved on-line accuracy (left) and model complexity (right). Hyperparameters were optimized on the first 1000 samples.

Drift setting	On-line accuracy						Complexity					
	ORF	ILVQ	LPP_{CART}	IELM	SGD_{Lin}	NB_{Gauss}	ORF	ILVQ	LPP_{CART}	IELM	SGD_{Lin}	NB_{Gauss}
Inter. RBF	45.9	**76.8**	29.4	29.5	44.3	29.9	762k	46k	166k	900	**45**	60
Electricity	69.9	72.5	67.5	54.8	**84.6**	63.2	140k	1.4k	30k	560	**6**	20
Moving RBF	45.6	**76.6**	18.0	15.9	40.6	17.2	721k	2.6k	32k	1.0k	**45**	60
Cover type	89.6	88.3	39.7	51.3	**94.6**	54.6	1.3M	292k	76k	6.0k	385	756
∅	62.7	**78.5**	38.7	37.9	66.0	41.2	729k	85k	76k	2.1k	123	234
∅ rank	2.3	**1.8**	5.0	5.5	2.0	4.5	6.0	4.3	4.8	3.0	**1.0**	2.0

5.8.1. Datasets

Mainly artificial datasets with known drift characteristics were utilized. It is usually unclear whether concept drift is present at all within a given real world dataset. However, we included two commonly used real world benchmark. The characteristics of the datasets are given in Table 5 and we precisely describe them in the following.[5]

Interchanging RBF. Fifteen Gaussians with random covariance matrices are replacing each other every 3000 samples. Thereby, the number of Gaussians switching their position increases each time by one until all are simultaneously changing their location. This allows to evaluate an algorithm in the context of abrupt drift with increasing strength. Altogether 66 abrupt drifts are occurring within this dataset.

Electricity market dataset. This problem is often used as a benchmark for concept drift classification. Initially described in [77], it was used thereafter for several performance comparisons [74,78,79]. A critical note to its suitability as a benchmark can be found in [80]. The dataset holds information of the Australian New South Wales Electricity Market, whose

[5] All datasets are available at https://github.com/vlosing/Online-learning.

Table 7
Discretized assessment of the core algorithmic properties. Especially, the major categories accuracy and model complexity are highly affected by the evaluated datasets and represent the average results on the diverse tasks considered in our experiments.

	SVMs	ORF	ILVQ	LPP_{CART}	IELM	SGD_{Lin}	NB_{Gauss}
Endless learning	✗	✗	(√)	(√)	√	√	√
Accuracy	★★★	★★	★★	★★	★★	★	★
Convergence speed	★★★	★★	★★★	★★	★★	★★	★★★
Model complexity	★	★★	★★	★★	★★	★★★	★★★
Training time	★	★★★	★★	★★	★★★	★★★	★★★
Run time	★★	★★★	★★	★★★	★★	★★★	★★★
Complexity of HPO	★	★★★	★	★★★	★★★	★★	–
Robustness of subset based HPO	★★	★★	★★	★★	★★	★★★	–
Viable for concept drift	✗	✗	(√)	✗	✗	(√)	✗

prices are affected by supply and demand. Each sample, characterized by attributes such as day of week, time stamp, market demand etc., refers to a period of 30 min and the class label identifies the relative change (higher or lower) compared to the last 24 h.

Moving RBF. Gaussian distributions with random initial positions, weights and standard deviations are moved with constant speed v in d-dimensional space. The weight controls the partitioning of the examples among the Gaussians. We used the Random RBF generator in MOA [81] with the same parameterization as in [82] (10 dimensions, 50 Gaussians, 5 classes, $v = 0.001$).

Forest Cover Type. Assigns cartographic variables such as elevation, slope, soil type, … of 30×30 meter cells to different forest cover types. Only forests with minimal human-caused disturbances were used, so that resulting forest cover types are more a result of ecological processes. It is often used as a benchmark for drift algorithms [82–84].

5.8.2. Results

The resulting on-line accuracies as well as model complexities are given by Table 6. We excluded the SVMs from the ranking, since the highly overlapping distributions led to an extensive growth of support vectors and to DNF results in all datasets but Electricity. Methods simply learning an average model for all seen data instances such as NB_{Gauss} and LPP_{CART} are inappropriate for non-stationary environments as can be seen by the poor results. In general, a mechanism to forget obsolete knowledge is crucial to be able to deal with concept drift. This is given to some extent for the ILVQ and the SGD_{Lin}. Both incorporate a learning rate, which, if set flexible enough for the rate of drift, allows the model to adapt to new concepts. A common technique to deal with concept drift is the sliding window [82,85]. The ILVQ utilizes one to insert new prototypes such that the classification on recent examples is optimized. Hence, it weights new information stronger by design and has, therefore, the highest capacity of the methods to deal with concept drift. Note that the methods considered in this article are not especially designed to handle concept drift. Nonetheless, our brief evaluation shows that some of the methods yield surprisingly accurate results for the considered datasets, while others simply fail. It might be advisable to use dedicated techniques developed for non-stationary environments in applications where strong drift is expected, such as those incorporating explicit drift detection [74,86] or recent approaches incorporating dedicated memory models [87].

6. Conclusion

We analyzed the most common algorithms of incremental learning on diverse, stationary and non-stationary datasets. The outcomes of our experiments are summarized in Table 7. It provides a fast overview about the core attributes of the diverse set of considered methods, guiding the choice of an appropriate algorithm for a given task. Regarding the results, the SVMs deliver usually the highest accuracy at the expense of the most complex model. The approximate nature of the LASVM reduces the training time and allows it to perform for larger data sets in comparison to the ISVM. The ORF performs slightly worse but has a very fast training- and run-time. However, its model as well as those of both SVMs grows linearly with the number of samples and cannot be limited in a straightforward way. Therefore, these algorithms are not suited for learning in endless streams in contrast to all remaining methods, having either a constant or easily boundable complexity. The ILVQ offers an accurate and sparse alternative to the SVMs. LPP_{CART} is quite flexible since the base classifier can be arbitrary selected, however, it may struggle by its limited knowledge integration across chunks. Tree based models are especially suitable for high-dimensional data because of their compressed representation as well as their sub-linear run-time, which does not depend on the number of dimensions. However, the compressed representation infringes the learning speed, such that instance based models are converging faster and are more appropriate for learning tasks comprising only a few examples. The sparse models of SGD_{Lin} and NB_{Gauss} make them to particularly viable choices for large-scale learning in high dimensional space on the one hand, but it turns out to be not complex enough for low dimensional tasks on the other. NB_{Gauss} and tree based methods are the easiest to apply in practice requiring no or just a little HPO. Whereas the SVMs and the ILVQ require the most delicate setting.

In the future, we want to extend our work and provide an analysis especially targeting data streams comprising concept drift. In contrast to the work here, we want to exclusively focus on algorithms which were designed to handle concept drift. It is particularly interesting to see how algorithms are able to deal with specific types of drifts at various strengths.

Appendix A. Results grouped by dataset

In the following, we provide the results of the experiments grouped by each dataset to facilitate the comparison between the three different settings.

Border	Accuracy								Complexity							
	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$
Setting 1	**99.4**	98.4	97.6	96.8	96.5	96.0	35.5	96.4	797	251	3.7k	193	1.9k	750	**9**	12
Setting 2	**99.4**	98.8	96.3	97.1	96.4	96.2	33.7	96.4	1.1k	1.5k	8.0k	286	2.6k	875	**9**	12
Setting 3	**98.5**	97.6	94.0	94.7	88.4	88.0	37.5	94.4	–	–	–	–	–	–	–	–
Overlap	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$
Setting 1	82.2	81.0	**83.7**	83.0	81.7	83.4	67.6	66.4	10k	7.2k	2.3k	235	1.9k	900	**12**	16
Setting 2	81.8	80.7	81.2	82.1	81.0	**82.2**	68.4	66.4	10.9	9.2 k	31k	496	1.9k	546	**12**	16
Setting 3	**81.7**	78.8	78.2	81.1	72.7	74.8	67.9	67.5	–	–	–	–	–	–	–	–
Letter	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$
Setting 1	97.0	**97.1**	93.2	93.9	87.0	70.0	56.4	63.4	131k	140k	168k	14k	51k	8.4k	**442**	832
Setting 2	96.4	**97.5**	92.2	95.1	84.9	67.4	53.9	63.4	163	173k	473k	11k	70 k	6.5k	**442**	832
Setting 3	91.3	**92.7**	75.4	88.4	79.3	35.4	41.0	64.2	–	–	–	–	–	–	–	–
SUSY	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$
Setting 1	DNF	DNF	**79.5**	79.0	DNF	DNF	78.7	73.5	DNF	DNF	86.0	51.2	DNF	DNF	**19**	72
Setting 2	DNF	DNF	**79.3**	73.4	DNF	DNF	77.0	73.5	DNF	DNF	1.5M	453k	DNF	DNF	**19**	72
Setting 3	DNF	DNF	**79.3**	78.5	DNF	DNF	78.7	73.5	–	–	–	–	–	–	–	–
Outdoor	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$
Setting 1	70.9	**71.4**	71.0	66.9	68.5	70.9	23.2	60.5	43k	42k	8.8k	11k	6.2k	12k	**880**	1.7k
Setting 2	**71.9**	69.0	57.6	63.2	69.4	71.6	25.7	60.5	40k	53k	925	11 k	7.3k	8.3k	**880**	1.7k
Setting 3	**86.4**	82.3	34.2	82.6	68.5	73.3	18.0	65.0	–	–	–	–	–	–	–	–
COIL	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$
Setting 1	**96.5**	93.2	92.9	94.3	89.2	91.5	12.4	90.0	58k	93k	61k	18k	9.2k	36k	**2.2k**	4.2k
Setting 2	**95.6**	92.5	89.9	93.6	88.1	83.6	12.1	90.0	180k	113k	14k	17 k	6.1k	15k	**2.2k**	4.2k
Setting 3	75.4	66.3	66.6	**79.1**	58.7	63.1	9.6	70.2	–	–	–	–	–	–	–	–

DNA	Off-line accuracy								Complexity							
	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$
Setting 1	**94.9**	**94.9**	89.6	92.1	90.5	88.8	93.0	88.4	237k	190k	5.0k	33k	1.6k	55k	**543**	1.1k
Setting 2	**95.2**	95.1	88.2	90.8	92.0	86.7	93.3	88.4	333k	327k	500	42k	1.8k	76k	**543**	1.1k
Setting 3	**89.5**	**89.5**	73.1	84.6	67.9	49.1	84.7	86.1	–	–	–	–	–	–	–	–

USPS	Accuracy								Complexity							
	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$
Setting 1	95.4	**95.6**	92.5	91.4	90.3	92.1	89.0	75.4	710k	520k	33k	15k	9.6k	106k	**2.6k**	5.1k
Setting 2	**95.6**	94.7	91.4	91.4	88.9	89.7	89.8	75.4	744k	978k	61k	91k	12k	50k	**2.6k**	5.1k
Setting 3	**96.7**	96.6	84.5	92.7	86.6	88.8	88.5	76.0	–	–	–	–	–	–	–	–
Isolet	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$
Setting 1	96.2	**96.7**	92.5	92.0	90.0	91.9	91.5	80.1	2.8M	3.4M	31k	21k	**12.0k**	322k	16k	32k
Setting 2	**96.5**	96.4	92.0	89.4	90.7	88.5	89.6	80.1	6.2M	4.2M	123k	67k	**14k**	159k	16k	32k
Setting 3	**93.6**	92.9	69.2	84.7	76.3	80.7	74.3	75.2	–	–	–	–	–	–	–	–
MNIST	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$
Setting 1	DNF	**98.5**	94.3	94.8	92.4	89.1	86.0	55.6	DNF	14M	111k	315k	73k	397k	**7.9k**	16k
Setting 2	DNF	**98.1**	95.0	94.3	91.3	84.2	87.7	55.6	DNF	16M	253k	1.2M	162k	169k	**7.9k**	16k
Setting 3	DNF	**97.5**	87.1	90.8	89.0	86.5	83.7	56.5	–	–	–	–	–	–	–	–
Gisette	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$	ISVM	LASVM	ORF	ILVQ	LPP$_{CART}$	IELM	SGD$_{Lin}$	NB$_{Gauss}$
Setting 1	**98.0**	97.9	94.6	93.0	94.2	91.4	93.1	75.4	7.0M	6.2M	4.7k	263k	**2.5k**	2.5M	5.0k	20k
Setting 2	97.8	**97.9**	92.7	94.1	94.0	83.8	95.4	75.4	7.0M	6.7M	16k	470k	**3.0k**	1.0M	5.0k	20k
Setting 3	96.3	**96.4**	90.3	91.1	86.7	80.5	92.1	74.0	–	–	–	–	–	–	–	–

References

[1] M. Chen, S. Mao, Y. Liu, Big data: a survey, Mob. Netw. Appl. 19 (2) (2014), doi:10.1007/s11036-013-0489-0.

[2] R. Yang, M.W. Newman, Learning from a learning thermostat: lessons for intelligent systems for the home, in: Proceedings of the UbiComp'13, ACM, 2013, pp. 93–102, doi:10.1145/2493432.2493489.

[3] B.D. Carolis, S. Ferilli, D. Redavid, Incremental learning of daily routines as workflows in a smart home environment, ACM Trans. Interact. Intell. Syst. 4 (4) (2015) 20:1–20:23, doi:10.1145/2675063.

[4] R.M. French, Catastrophic forgetting in connectionist networks, Trends Cognit. Sci. 3 (4) (1999) 128–135.

[5] D.H. Wolpert, The supervised learning no-free-lunch theorems, in: Soft Computing and Industry, Springer, 2002, pp. 25–42.

[6] G. Ditzler, M. Roveri, C. Alippi, R. Polikar, Learning in nonstationary environments: a survey, Comput. Intell. Mag. 10 (4) (2015) 12–25.

[7] G. Cauwenberghs, T. Poggio, Incremental and decremental support vector machine learning, in: Proceedings of the 2001 Neural Information Processing Systems (NIPS), 2001.

[8] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Netw. 17 (6) (2006) 1411–1423, doi:10.1109/TNN.2006.880583.

[9] N. Cesa-Bianchi, G. Lugosi, Prediction, Learning, and Games, Cambridge University Press, 2006.

[10] T.L. Watkin, A. Rau, M. Biehl, The statistical mechanics of learning a rule, Rev. Mod. Phys. 65 (2) (1993) 499.

[11] A. Novikoff, On convergence proofs of perceptrons, Proceedings of the 1962 Symposium on Mathematical Theory of Automata 12 (2) (1962) 615–622.

[12] L. Atzori, A. Iera, G. Morabito, The internet of things: a survey, Comput. Netw. 54 (15) (2010) 2787–2805.

[13] R. Ade, P. Deshmukh, Methods for incremental learning: a survey, Int. J. Data Mining Knowl. Manag. Process 3 (4) (2013) 119.

[14] P. Joshi, P. Kulkarni, Incremental learning: areas and methods – a survey, Int. J. Data Mining Knowl. Manag. Process 2 (5) (2012) 43.

[15] C. Giraud-Carrier, A note on the utility of incremental learning, AI Commun. 13 (4) (2000) 215–223.

[16] A. Gepperth, B. Hammer, Incremental learning algorithms and applications, in: Proceedings of the 2016 European Symposium on Artificial Neural Networks (ESANN), 2016.

[17] M.M. Gaber, A. Zaslavsky, S. Krishnaswamy, Mining data streams: a review, ACM Sigm. Record 34 (2) (2005) 18–26.

[18] C.C. Aggarwal, Data classification: algorithms and applications, 1st, Chapman & Hall/CRC, 2014.

[19] I. Zliobaite, Learning under concept drift: an overview, CoRR abs/1010.4784 (2010) arXiv:1010.4784.

[20] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, Proc. ACM Comput. Surv. (CSUR) 46 (4) (2014) 44.

[21] P. Domingos, G. Hulten, A general framework for mining massive data streams, J. Comput. Graph. Stat. 12 (4) (2003) 945–949.

[22] J. Read, A. Bifet, B. Pfahringer, G. Holmes, Batch-incremental versus instance-incremental learning in dynamic and evolving data, in: Proceedings of the 2012 International Symposium on Intelligent Data Analysis, Springer, 2012, pp. 313–323.

[23] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems, J. Mach. Learn. Res. 15 (1) (2014) 3133–3181.

[24] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.

[25] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.

[26] B. Biggio, I. Corona, B. Nelson, B.I. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto, F. Roli, Security evaluation of support vector machines in adversarial environments, in: Support Vector Machines Applications, Springer, 2014, pp. 105–153.

[27] Y. Lu, K. Boukharouba, J. Boonært, A. Fleury, S. Lecoeuche, Application of an incremental SVM algorithm for on-line human recognition from video surveillance using texture and color features, Neurocomputing 126 (2014) 132–140.

[28] A. Bordes, S. Ertekin, J. Weston, L. Bottou, Fast kernel classifiers with online and active learning, J. Mach. Learn. Res. 6 (2005) 1579–1619.

[29] J. Platt, Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, Technical report, 1998, p. 21.

[30] C.-J. Hsieh, S. Si, I.S. Dhillon, A divide-and-conquer solver for kernel support vector machines., in: Proceedings of the 2014 International Conference on Machine Learning (ICML), 2014, pp. 566–574.

[31] Z. Cai, L. Wen, Z. Lei, N. Vasconcelos, S.Z. Li, Robust deformable and occluded object tracking with dynamic graph, IEEE Trans. Image Process. 23 (12) (2014) 5497–5509.

[32] A. Saffari, C. Leistner, J. Santner, M. Godec, H. Bischof, On-line random forests, in: Proceedings of the 2009 IEEE Twelfth International Conference on Computer Vision Workshops, 2009.

[33] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (1) (2006), doi:10.1007/s10994-006-6226-1.

[34] B. Lakshminarayanan, D.M. Roy, Y.W. Teh, Mondrian forests: efficient online random forests, in: Proceedings of the 2014 Advances in Neural Information Processing Systems, 2014, pp. 3140–3148.

[35] F. Pernici, A. Del Bimbo, Object tracking by oversampling local features, IEEE Trans. Pattern Anal. Mach. Intell. 36 (12) (2014) 2538–2551.

[36] A. Sato, K. Yamada, Generalized learning vector quantization, in: Proceedings of the 1995 Neural Information Processing Systems (NIPS), MIT Press, 1995.

[37] V. Losing, B. Hammer, H. Wersing, Interactive online learning for obstacle classification on a mobile robot, in: Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN) 2015, 2015, pp. 1–8, doi:10.1109/IJCNN.2015.7280610.

[38] P. Schneider, M. Biehl, B. Hammer, Adaptive relevance matrices in learning vector quantization, Neural Comput. 21 (12) (2009) 3532–3561, doi:10.1162/neco.2009.11-08-908.

[39] K. Bunte, P. Schneider, B. Hammer, F.-M. Schleif, T. Villmann, M. Biehl, Limited rank matrix learning, discriminative dimension reduction and visualization, Neural Netw. 26 (2012) 159–173.

[40] R. Polikar, L. Upda, S. Upda, V. Honavar, Learn++: an incremental learning algorithm for supervised neural networks, J. IEEE Trans. Syst. Man Cybern. 31 (4) (2001) 497–508, doi:10.1109/5326.983933.

[41] Y. Freund, R.E. Schapire, A short introduction to boosting, in: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1999, pp. 1401–1406.

[42] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wadsworth and Brooks, Monterey, CA, 1984. New edition.

[43] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning Internal Representations by Error Propagation, Technical report, DTIC Document, 1985.

[44] M. De-la Torre, E. Granger, P.V. Radtke, R. Sabourin, D.O. Gorodnichy, Partially-supervised learning from facial trajectories for face recognition in video surveillance, Inf. Fusion 24 (2015) 31–53.

[45] J.F.G. Molina, L. Zheng, M. Sertdemir, D.J. Dinter, S. Schönberg, M. Rädle, Incremental learning with SVM for multimodal classification of prostatic adenocarcinoma, PloS One 9 (4) (2014) e93600.

[46] J. Tang, C. Deng, G.-B. Huang, Extreme learning machine for multilayer perceptron, IEEE Trans. Neural Netw. Learn. Syst. 27 (4) (2016) 809–821.

[47] J. Tang, C. Deng, G.-B. Huang, B. Zhao, Compressed-domain ship detection on spaceborne optical image using deep neural network and extreme learning machine, IEEE Trans. Geosci. Remote Sens. 53 (3) (2015) 1174–1185.

[48] H. Zhang, The Optimality of Naive Bayes, in: B. Valerie, M. Zdravko (Eds.), FLAIRS Conference, AAAI Press, 2004.

[49] C. Salperwyck, V. Lemaire, Learning with few examples: an empirical study on leading classifiers, in: Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN), IEEE, 2011, pp. 1010–1019.

[50] V. Metsis, I. Androutsopoulos, G. Paliouras, Spam filtering with naive Bayes-which Naive Bayes? in: Proceedings of the 2006 Conference on Email and Anti-Spam, CEAS, 2006, pp. 27–28.

[51] S. Ting, W. Ip, A.H. Tsang, Is naive Bayes a good classifier for document classification? Int. J. Softw. Eng. Appl. 5 (3) (2011) 37–46.

[52] W. Lou, X. Wang, F. Chen, Y. Chen, B. Jiang, H. Zhang, Sequence based prediction of DNA-binding proteins based on hybrid feature selection using random forest and gaussian naive Bayes, PLoS One 9 (1) (2014) e86703.

[53] J.C. Griffis, J.B. Allendorfer, J.P. Szaflarski, Voxel-based gaussian Naïve Bayes classification of ischemic stroke lesions in individual T1-weighted MRI scans, J. Neurosci. Methods 257 (2016) 97–108.

[54] T. Zhang, Solving large scale linear prediction problems using stochastic gradient descent algorithms, in: Proceedings of the Twenty-First International Conference on Machine Learning, ACM, 2004, p. 116.

[55] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of the COMPSTAT'2010, Springer, 2010, pp. 177–186.

[56] P. Richtárik, M. Takáč, Parallel coordinate descent methods for big data optimization, Math. Program. 156 (1–2) (2016) 433–484.

[57] Z. Akata, F. Perronnin, Z. Harchaoui, C. Schmid, Good practice in large-scale learning for image classification, IEEE Trans. Pattern Anal. Mach. Intell. 36 (3) (2014) 507–520.

[58] M. Sapienza, F. Cuzzolin, P.H. Torr, Learning discriminative space–time action parts from weakly labelled videos, Int. J. Comput. Vis. 110 (1) (2014) 30–47.

[59] S. Ertekin, L. Bottou, C.L. Giles, Nonconvex online support vector machines, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2) (2011) 368–381.

[60] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011) 1517–1531.

[61] G. Ditzler, R. Polikar, Incremental learning of concept drift from streaming imbalanced data, IEEE Trans. Knowl. Data Eng. 25 (10) (2013) 2283–2301.

[62] J. Zhao, Z. Wang, D.S. Park, Online sequential extreme learning machine with forgetting mechanism, Neurocomputing 87 (2012) 79–89.

[63] Y. Ye, S. Squartini, F. Piazza, Online sequential extreme learning machine in nonstationary environments, Neurocomputing 116 (2013) 94–101.

[64] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, in: Proceedings of the 2013 Advances in Neural Information Processing Systems, 2013, pp. 315–323.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[66] M. Lichman, UCI machine learning repository, 2013.

[67] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (2011) 27:1–27:27.

[68] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, D.D. Cox, Hyperopt: a python library for model selection and hyperparameter optimization, Comput. Sci. Discovery 8 (1) (2015) 014008.

[69] J.S. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter

optimization, in: Proceedings of the 2011 Advances in Neural Information Processing Systems, 2011, pp. 2546–2554.

[70] H. He, S. Chen, K. Li, X. Xu, Incremental learning from stream data, IEEE Trans. Neural Netw. 22 (12) (2011) 1901–1914.

[71] M. Grbovic, S. Vucetic, Learning vector quantization with adaptive prototype addition and removal, in: Proceedings of the 2009 International Joint Conference on Neural Networks (IJCNN) 2009, IEEE, 2009, pp. 994–1001.

[72] R. Elwell, R. Polikar, Incremental learning in nonstationary environments with controlled forgetting, in: Proceedings of the 2009 International Joint Conference on Neural Networks (IJCNN), IEEE, 2009, pp. 771–778.

[73] T. Downs, K. Gates, A. Masters, Exact simplification of support vector solutions, J. Mach. Learn. Res. 2 (2002) 293–297.

[74] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Proceedings of the Advances in artificial intelligence – SBIA 2004, Springer, 2004, pp. 286–295.

[75] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: an ensemble method for drifting concepts, J. Mach. Learn. Res. 8 (2007) 2755–2790.

[76] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011) 1517–1531, doi:10.1109/TNN.2011.2160459.

[77] M. Harries, U.N. cse tr, N.S. Wales, SPLICE-2 Comparative Evaluation: Electricity Pricing, Technical report, 1999.

[78] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: Proceedings of the Fourth International Workshop on Knowledge Discovery from Data Streams, 6, 2006, pp. 77–86.

[79] L.I. Kuncheva, C.O. Plumpton, Adaptive learning rate for online linear discriminant classifiers, in: Structural, Syntactic, and Statistical Pattern Recognition, Springer, 2008, pp. 510–519.

[80] I. Zliobaite, How good is the Electricity benchmark for evaluating concept drift adaptation, CoRR (2013) arXiv:1301.3524.

[81] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, Moa: Massive online analysis, J. Mach. Learn. Res. 11 (2010) 1601–1604.

[82] A. Bifet, B. Pfahringer, J. Read, G. Holmes, Efficient data stream classification via probabilistic adaptive windows, in: Proceedings of the Twenty-eighth Annual ACM Symposium on Applied Computing, SAC '13, ACM, New York, NY, USA, 2013, pp. 801–806, doi:10.1145/2480362.2480516.

[83] J. Gama, R. Rocha, P. Medas, Accurate decision trees for mining high-speed data streams, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2003, pp. 523–528.

[84] N.C. Oza, S. Russell, Experimental comparisons of online and batch versions of bagging and boosting, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 359–364.

[85] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Mach. Learn. 23 (1) (1996) 69–101.

[86] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing., in: Proceedings of the 2007 SIAM International Conference on Data Mining (SDM), 7, SIAM, 2007, p. 2007.

[87] V. Losing, B. Hammer, H. Wersing, Knn classifier with self adjusting memory for heterogeneous concept drift, in: Proceedings of the 2016 IEEE Sixteenth International Conference on Data Mining (ICDM), 2016, pp. 291–300, doi:10.1109/ICDM.2016.0040.

Viktor Losing received his M.Sc. in Intelligent Systems at the University of Bielefeld in 2014. Since 2015 he is a Ph.D. student at the CoR-Lab of the University of Bielefeld in cooperation with the HONDA Research Institute Europe. His research interests comprise incremental and on-line learning, learning under concept drift as well as their applications in real world problems.

Barbara Hammer received her Ph.D. in Computer Science in 1995 and her venia legendi in Computer Science in 2003, both from the University of Osnabrück, Germany. From 2000–2004, she was leader of the junior research group Learning with Neural Methods on Structured Data at University of Osnabrück before accepting an offer as professor for Theoretical Computer Science at Clausthal University of Technology, Germany, in 2004. Since 2010, she is holding a professorship for Theoretical Computer Science for Cognitive Systems at the CITEC cluster of excellence at Bielefeld University, Germany. Her areas of expertise include hybrid systems, self-organizing maps, clustering, and recurrent networks as well as applications in bioinformatics, industrial process monitoring, or cognitive science. She is currently chairing the IEEE CIS Technical Committee on Data Mining, and the Fachgruppe Neural Networks of the GI.

Heiko Wersing received the diploma in physics in 1996 from Bielefeld University, Germany. In 2000, he received his Ph.D. in science from the Faculty of Technology, Bielefeld University. In 2000, he became a member of the Future Technology Research Group of Honda R&D Europe, GmbH, Offenbach, Germany. Currently, he holds a position as a chief scientist in the Honda Research Institute Europe, at Offenbach. His current research interests include recurrent neural networks, models of perceptual grouping and feature binding, principles of sparse coding, biologically motivated object recognition and online learning.