REVIEW

*National Science Review*
9: nwac123, 2022
https://doi.org/10.1093/nsr/nwac123
Advance access publication 1 July 2022

INFORMATION SCIENCE

# Open-environment machine learning

Zhi-Hua Zhou

## ABSTRACT

Conventional machine learning studies generally assume *close-environment* scenarios where important factors of the learning process hold invariant. With the great success of machine learning, nowadays, more and more practical tasks, particularly those involving *open-environment* scenarios where important factors are subject to change, called *open-environment machine learning* in this article, are present to the community. Evidently, it is a grand challenge for machine learning turning from close environment to open environment. It becomes even more challenging since, in various big data tasks, data are usually accumulated with time, like *streams*, while it is hard to train the machine learning model after collecting all data as in conventional studies. This article briefly introduces some advances in this line of research, focusing on techniques concerning emerging new classes, decremental/incremental features, changing data distributions and varied learning objectives, and discusses some theoretical issues.

**Keywords:** machine learning, artificial intelligence, open-environment machine learning, open ML

## INTRODUCTION

Machine learning has achieved great success in various applications, particularly in *supervised learning* tasks such as classification and regression. In machine learning, typically, a predictive model optimizing a specific objective is learned from a training data set composed of training examples, each corresponding to an event/object. A training example consists of two parts: a feature vector (or called an *instance*) describing the appearance of an event/object, and a *label* indicating the corresponding ground-truth output. In classification, the label indicates the class to which the training instance belongs; in regression, the label is a real-value response corresponding to the instance. This article mainly focuses on classification, though most discussions are also applicable to regression and other machine learning tasks. Formally, consider the task of learning $f : \mathcal{X} \mapsto \mathcal{Y}$ from a training data set $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$, where $\boldsymbol{x}_i \in \mathcal{X}$ is a feature vector in the feature space $\mathcal{X}$, and $y_i \in \mathcal{Y}$ is a ground-truth label in the given label set $\mathcal{Y}$.

It is noticeable that current successes in machine learning mostly involve tasks that assume *close-environment* scenarios, where important factors of the learning process hold invariant. For example, all the class labels to be predicted are known in advance, the features describing training/testing data never change, all data are from an identical distribution and the learning process is optimized towards an unchangeable unique objective. Figure 1 illustrates those typical invariant factors assumed in close-environment machine learning studies.

The close-environment assumptions offer a simplified abstraction that enables complicated tasks to be handled in an easier way, leading to the prosperous development of machine learning techniques. With the great achievements attained by these techniques, nowadays, more and more challenging tasks beyond the close-environment setting are present to the community, requesting new generation of machine learning techniques that are able to handle *changes* in important factors of the learning process. We call this *open-environment machine learning*, or *open learning* or *open ML* for short. Note that the name 'open-world machine learning' was used to refer to machine learning with unseen class [1] or out-of-distribution (OOD) data [2]. In fact, it is not beyond close-environment studies if the unseen class is *known* in advance, and related to the section 'Emerging New Classes' if the unseen class is *unknown*. OOD is related to the section 'Changing Data Distributions', though concerning only a different distribution is simpler than distribution changing with time.

National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

E-mail: zhouzh@nju.edu.cn

**Received** 28 February 2022; **Revised** 8 May 2022; **Accepted** 14 June 2022

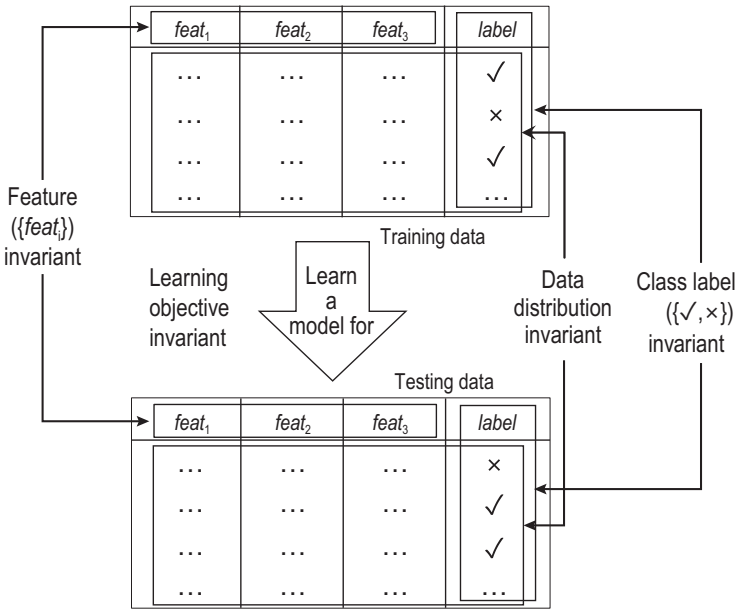


**Figure 1.** Typical invariant factors assumed in close-environment machine learning studies.

There seems a straightforward solution: to artificially generate many training examples by mimicking the possible changes in advance, and then feed these data to a powerful machine learning model such as a deep neural network. Such a solution, however, is only applicable when users have knowledge about, or at least can estimate, what changes and how the changes will occur. Unfortunately, it is not the case in most practical tasks. It becomes even more challenging when we consider the fact that data in real big data tasks are usually accumulated with time, e.g. instances are being received one by one, like a *stream*. It is impossible to train a machine learning model after we get *all data* at hand as in conventional studies; a more reasonable way is to enable the trained model to be refined/updated according to the newly received data. Unfortunately, it is well known that *catastrophic forgetting* [3] can occur if a trained deep neural network is to be refined with new data only, whereas a frequent re-training based on storing all received data may lead to unbearably large computational and storage costs. Though there are studies like *continual learning* [4] trying to help deep neural networks resist forgetting, many passes scanning over large batches of training data and offline training are generally required, with serious computational and storage concerns on big stream data.

Despite the grand challenges, recently, there have been considerable research efforts on open ML. This article briefly introduces some advances in this line of research, focuses on techniques concerning emerging new classes, decremental/incremental features, changing data distributions and varied learning objectives. Some theoretical issues will also be discussed.

## EMERGING NEW CLASSES

Close-environment machine learning studies generally assume that the class label of any unseen instance $\hat{\boldsymbol{x}}$ must be a member of the given label set known in advance, i.e. $\hat{y} \in \mathcal{Y}$. Unfortunately, this does not always hold. For example, consider a forest disease monitor aided by a machine learning model trained with signals sent from sensors deployed in the forest. It is evident that one can hardly enumerate all possible class labels in advance, because some forest diseases can be totally novel, such as those caused by invasive insect pests never encountered in this region before. To be able to handle $\hat{y} \notin \mathcal{Y}$ is a basic requirement for open ML.

It might be thought that we can artificially generate some virtual training examples for the new classes, just like popular training tricks employed in adversarial deep neural networks. Here, the difficulty lies in the fact that we can hardly imagine what unknown class (called *NewClass* in the following) might occur, whereas training a model accommodating *all possible classes* is impossible or unbearably expensive.

Technically, if *all data* are at hand, especially including the unlabeled instances to be predicted, then handling NewClass can be treated as a special semi-supervised learning [5] task, e.g. by establishing the semi-supervised large margin separator corresponding to the tightest contour for each known class, and then regarding unlabeled instances falling outside all contours as NewClass instances [6]. Actually, the distribution of NewClass can be approximated by separating the distribution of known classes for that of the unlabeled data [7]. Such strategies, however, are not directly applicable when data are accumulated with time.

Consider the following setting of learning with an emerging new class. A machine learning model is trained from some initial training data and then deployed to handle unseen instances being received like a stream. For incoming instances of known classes, the trained model should be able to make correct predictions. For incoming instances of unknown classes, the model should be able to report that a NewClass instance is encountered; the user can then create a new label for the NewClass. After encountering a few instances of this NewClass, the trained model should be able to be refined/updated such that the NewClass becomes a known class whose incoming instances can be accurately predicted. Ideally, it is desired that the whole process

does not require retraining based on storage of *all data* received, since this would be terribly expensive or even infeasible in real big data tasks. Evidently, the above describes an unsupervised/supervised mixing task with a human in the loop.

At first glance, learning with an emerging new class seems relevant to *zero-shot learning*, a hot topic in image classification, aiming to classify visual classes that did not occur the in training data set [8–10]. Note that zero-shot learning is assumed to work with *side information*, i.e. external knowledge, such as class definitions/descriptions/properties, that can help associate the seen and unseen classes, and, thus, it can be treated as a kind of transfer learning [11]; in contrast, learning with an emerging new class is a general machine learning setting that does not assume such external knowledge. In other words, zero-shot learning assumes that the unseen classes are known, though they did not occur in the training data, whereas learning with an emerging new class tackles the grand challenge that the unseen classes are unknown. Thus, approaches for learning with an emerging new class can be more general, and can be converted and applied to zero-shot learning.

Classification with a *reject* option [12–14] aims to avoid unconfident predictions that are likely to be incorrect, assuming that all classes are known in advance. *Open set recognition/classification* [15,16] extends the reject option to consider the possibility that the unknown class may occur in the testing phase, with the goal to recognize known classes and reject NewClass. Neither of them attempts to enable the trained model to accommodate NewClass. Some generalized open set recognition studies try to recognize the unknown class, by assuming the availability of side information like aforementioned in zero-shot learning [16], whereas learning with an emerging new class is a general machine learning setting that does not assume such external knowledge.

Learning with an emerging new class is actually a kind of *incremental learning*, which emphasizes that a trained model only requires slight modification to accommodate new information. There was a long history of studies on incremental learning [17–20], mostly concerning the increment of training examples, i.e. E-IL (example-incremental learning) defined in [21]. Besides E-IL, the other two types of incremental learning defined in [21] are A-IL (attribute-incremental learning) and C-IL (class-incremental learning). A-IL concerns feature increments, related to what will be discussed in the section entitled 'Decremental/Incremental Features' below, though previous studies have generally been devoted to selecting an adequate feature space given all data/features in advance [22,23]. C-IL concerns class increments, related to learning with an emerging new class, though previous studies concerned little for the identification of NewClass and generally assumed that the incremental class is known [24].

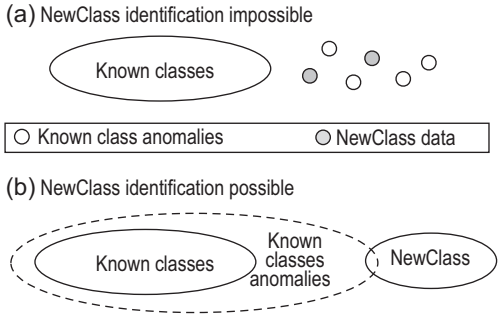


**Figure 2.** NewClass identification is not always possible.

*Class discovery* [25,26] tries to discover rare classes, as a separate process from class prediction. As mentioned above, learning with an emerging new class is an unsupervised/supervised mixing task, while those studies are somewhat relevant to its first phase, mostly unsupervised part.

In a general solution [27] to learning with an emerging new class, the first phase, NewClass identification, is realized by anomaly detection. Here, the challenge is to distinguish the NewClass data from anomalies of known classes. In general, this is not always possible; for example, Fig. 2a provides an illustration in which the NewClass and anomalies of known classes can hardly be distinguished. Fortunately, in many real tasks it is reasonable to assume that *NewClass instances are more 'abnormal' than anomalies of known classes*, as illustrated in Fig. 2b. If this does not hold in the original feature space, we can try to identify an adequate feature space by kernel mapping or representation learning. After that, the identification of NewClass instances reduces to anomaly detection from streams, which can be tackled by approaches such as the *isolation forest* [28].

A major challenge of the second phase is to refine/update a trained model to accommodate NewClass without sacrificing performance on known classes. For deep neural networks, a retraining based on all data (or at least on smartly selected subsamples) would be required to avoid catastrophic forgetting, with huge computational and storage costs. It would be ideal to do local refinement only for accommodating NewClass rather than doing global changes that may seriously affect known classes. One solution is to exploit the advantage of tree/forest models through refining only tree leaves involving NewClass in an incremental way [27], which does not even need any storage for known class data. Alternatives include techniques that can localize the influence of difference classes such that changes according to NewClass will not significantly affect

known classes, such as approaches based on global and local sketching [29].

If there is more than one new class, the clustering structure of NewClass data can be exploited [30]. Note that there is usually a large gap between the moments when NewClass is detected for the first time and when the model has been refined. To reduce this gap, some efforts have been devoted to enabling the model to update based on fewer NewClass data [31]. *Multi-label learning* with emerging new classes is more challenging because in this scenario the NewClass instances may also hold known class labels, and may even appear in dense regions of known classes, where the key is to detect significant changes in feature combinations and/or label combinations [32]. A relevant topic is to examine what known classes are closely related to NewClass, and an evaluation methodology concerning the mapping from NewClass to known classes has been developed [33].

There are situations where some NewClass instances appeared in the training data but were misperceived as known class instances, possibly due to an insufficiency in the feature information. This is even more challenging, with only one very preliminary study having been carried out [34].

## DECREMENTAL/INCREMENTAL FEATURES

Close-environment machine learning studies generally assume that all possible instances, including unseen ones, reside in the same feature space, i.e. $\hat{\boldsymbol{x}} \in \mathcal{X}$. Unfortunately, this does not always hold. Taking, for example, the forest disease monitor mentioned in the section entitled 'Emerging New Classes,' some sensors could not continue sending signals due to an exhausted battery, leading to decremental features, whereas some new sensors can be deployed, leading to incremental features. To be able to handle $\hat{\boldsymbol{x}} \in \hat{\mathcal{X}} \neq \mathcal{X}$ is also a requirement for open ML. Note that in contrast to varied classes where only an emerging new class requires special treatment whereas a disappeared class can be simply ignored, both decremental and incremental features require attention since feature decrement can lead to seriously downgraded performance.

Consider the following setting of learning with decremental/incremental features. A machine learning model is trained from some initial training data and then deployed to handle unseen data being received like a stream, with decremental and/or incremental features. For incoming testing data, the model should be able to make correct predictions; for incoming additional training data, the model should be able to be refined accordingly. Ideally, it is desirable that the whole process does not require retraining based on storage of *all data* received.

In general, it is not always possible to build a machine learning model that is able to benefit from $\boldsymbol{x} \in \mathcal{X}$ for $\hat{\boldsymbol{x}} \in \hat{\mathcal{X}} \neq \mathcal{X}$, because machine learning has to learn from experience to improve performance, whereas in most cases there might be little helpful experience from the learning in $\mathcal{X}$ to the learning in $\hat{\mathcal{X}}$ when $\hat{\mathcal{X}} \cap \mathcal{X} = \emptyset$. For example, as illustrated in Fig. 3a, if the feature spaces of phase$_1$ data (i.e. $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{T_1}, y_{T_1})\}$) and phase$_2$ data (i.e. $\{(\boldsymbol{x}_{T_1+1}, y_{T_1+1}), \ldots, (\boldsymbol{x}_{T_2}, y_{T_2})\}$) are totally different then the model trained in phase$_1$ is helpless for phase$_2$, and a new model has to be trained from scratch based on feature set $S_2$ for phase$_2$.

Fortunately, in many real tasks it is often the case that $\hat{\mathcal{X}} \cap \mathcal{X} \neq \emptyset$. In other words, there are features of phase$_1$ that survive to be active in phase$_2$ though many other features vanish, as illustrated in Fig. 3b. For example, different sensors may have different battery lifetimes, and thus some old sensors still work after new sensors are deployed. Formally, in phase$_1$, $\mathcal{X} = \mathcal{X}^{de} \cup \mathcal{X}^{s}$, where $\mathcal{X}^{de}$ and $\mathcal{X}^{s}$ denote the decremental and survived feature sets, respectively; in phase$_2$, $\hat{\mathcal{X}} = \mathcal{X}^{s} \cup \mathcal{X}^{in}$, where $\mathcal{X}^{in}$ denotes the incremental feature set. As $\mathcal{X}^{s}$ is shared in both phases, in addition to training model$_1$ from $\mathcal{X}$, model$_2$ based on $\mathcal{X}_s$ can be trained in phase$_1$, e.g. by [35]

$$\begin{aligned}\min_{\boldsymbol{w},\boldsymbol{w}^s} &\sum_{i=1}^{T_1}(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle - y_i)^2 + \sum_{i=1}^{T_1}(\langle \boldsymbol{w}^s, \boldsymbol{x}_i^s\rangle - y_i)^2 \\ &+ \alpha \sum_{i=1}^{T_1}(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle - \langle \boldsymbol{w}^s, \boldsymbol{x}_i^s\rangle)^2 \\ &+ \gamma(\|\boldsymbol{w}\|^2 + \|\boldsymbol{w}^s\|^2), \end{aligned} \tag{1}$$

where $\langle \cdot, \cdot \rangle$ is the inner product, $\boldsymbol{w}$ and $\boldsymbol{w}^s$ are parameters of model$_1$ and model$_2$, respectively, while $\alpha, \gamma > 0$ are the regularization coefficients. Such a process works like 'compressing' helpful predictive information from model$_1$ in $\mathcal{X}$ to model$_2$ in $\mathcal{X}^s$. Then, in phase$_2$, in addition to model$_3$ trained based on $\hat{\mathcal{X}}$, model$_2$ trained in phase$_1$ can still be used. Thus, the prediction in phase$_2$ can be made by combining model$_3$ fed with $\hat{\boldsymbol{x}}$ and model$_2$ fed with the $\mathcal{X}^s$ part of $\hat{\boldsymbol{x}}$. In this way, some experience learned from phase$_1$ can be exploited in phase$_2$ through the use of model$_2$.

Interestingly, even when $\hat{\mathcal{X}} \cap \mathcal{X} = \emptyset$, there are cases where it is possible to enable phase$_1$ learning to be helpful to phase$_2$, in particular, when feature increment occurs earlier than feature decrement [36], e.g. new sensors are deployed slightly before

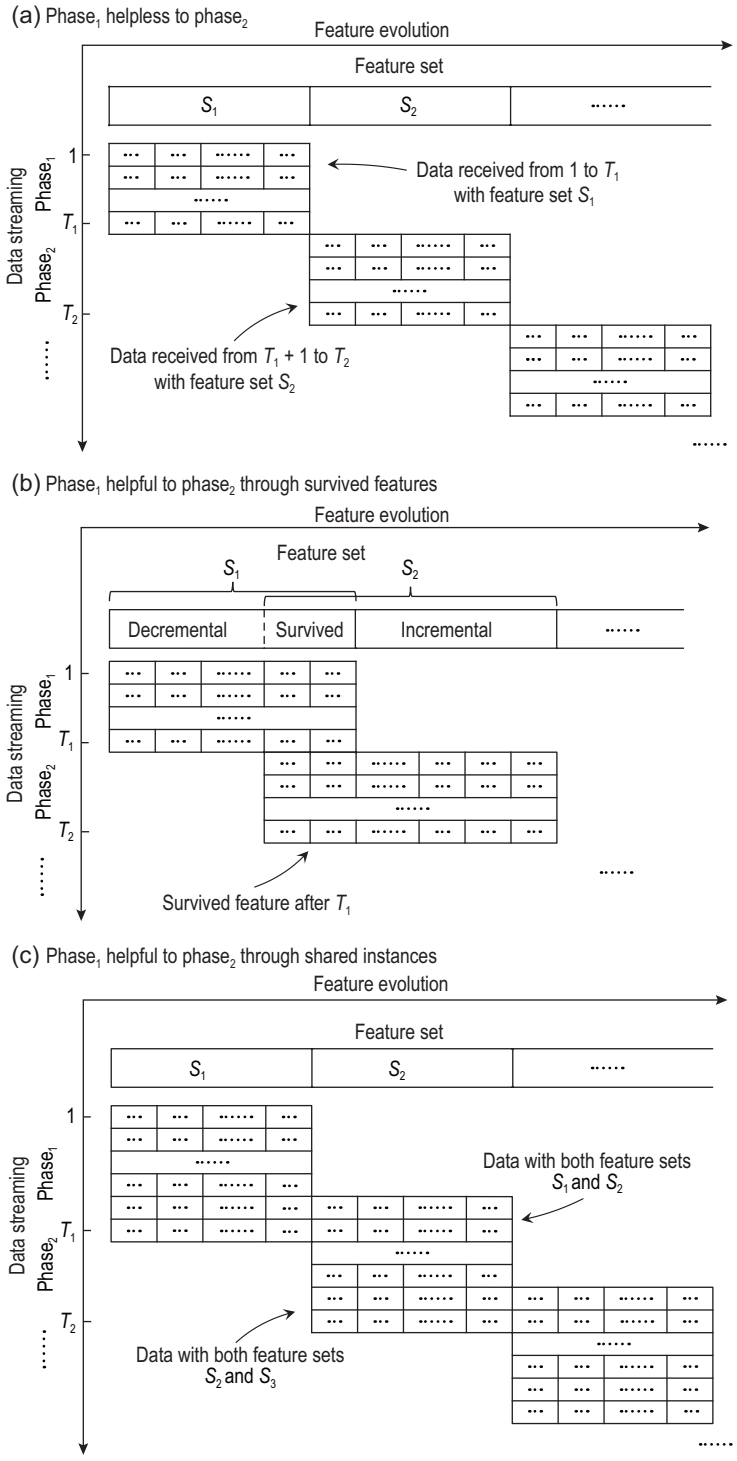


**Figure 3.** Helpless/helpful feature evolution.

old sensors' batteries are exhausted. As illustrated in Fig. 3c, in this situation there exists a small set of data with both sets of features that can help build the mapping $\psi : \hat{\mathcal{X}} \mapsto \mathcal{X}$. Thus, though $\hat{\boldsymbol{x}}$ received in phase$_2$ with features of $\hat{\mathcal{X}}$ only, model$_1$ learned in phase$_1$ can still be exploited by feeding it with $\psi(\hat{\boldsymbol{x}})$. Then, the phase$_2$ prediction can be made by combining model$_1$ with model$_2$ trained from $\hat{\mathcal{X}}$, either through weighted selection or weighted combination. It has been proved that the cumulative loss of the weighted combination is comparable to the minimum loss between the two models, and the cumulative loss of the weighted selection is comparable to the loss of optimal selection.

The training of these models can be accomplished by online learning techniques such as online gradient descent, and thus the above strategies can be naturally applied to stream data. It is noticeable that the above strategies can be naturally extended to more phases, and predictions can be made by the combination of multiple models from different feature spaces. Thus, the performance of later phases can even be enhanced by exploiting ensemble learning [37].

Recently, there have been studies of learning with feature decrement/increment at the unpredictable phase [38], along with data distribution changes [39], etc., and applications such as sensor-based activity recognition [40].

## CHANGING DATA DISTRIBUTIONS

Close-environment machine learning studies generally assume that all data, including both training and testing data, are independent samples from an identical distribution (i.e. *i.i.d.* samples). Unfortunately, this does not always hold. Taking, for example, the forest disease monitor mentioned in the section entitle 'Emerging New Classes' again, the model may be built in summer based on sensor signals received in that season, but it is hoped to work well in all seasons. Figure 4 provides an illustration that shows that ignoring the data distribution change may lead to seriously downgraded performance.

There have been plenty of studies concerning the training/testing data distribution change. For example, for the *prior probability shift* and *covariate shift* [41], we have $P_{\text{train}}(y|\boldsymbol{x}) = P_{\text{test}}(y|\boldsymbol{x})$ with $P_{\text{train}}(y) \neq P_{\text{test}}(y)$ and $P_{\text{train}}(\boldsymbol{x}) \neq P_{\text{test}}(\boldsymbol{x})$, respectively, whereas for the *concept drift* [42], we have $P_{\text{train}}(y|\boldsymbol{x}) \neq P_{\text{test}}(y|\boldsymbol{x})$. Many relevant studies have been conducted under the umbrella of domain adaptation [43–45] or transfer learning [11]. Note that in the stream situation, data distribution change can occur in any phase of the stream; it is not limited to the testing phase. To be able to handle various kinds of data distribution change is an important requirement for open ML.

In general, learning with changing data distributions is not always possible, e.g. if the data distribution can change arbitrarily in every moment without knowledge about how it could change.

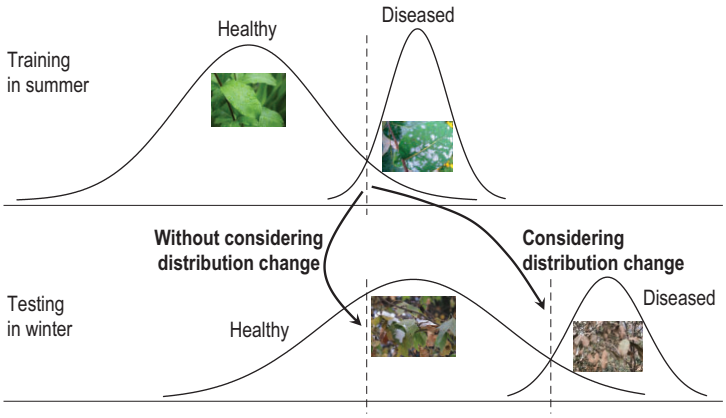


**Figure 4.** The data distribution change cannot be ignored.

Fortunately, in many real tasks it is reasonable to assume that the *current observation has a close relation to recent observations*; in other words, the current instance and the most recent ones are usually from similar or even identical distributions, and *the far the dissimilar*. Thus, we can try to exploit some recent data in the stream to help.

General approaches are often based on *sliding window*, *forgetting* or *ensemble* mechanisms. Sliding-window-based approaches hold recent instances and discard old ones falling outside the window, with a fixed or adaptive window size [46,47]. Forgetting-based approaches assign a weight to each instance, and downweight old instances according to their age [48,49]. Ensemble-based [37] approaches add/remove component learners in the ensemble adaptively, and dynamically adjust the weights of learners for incoming instances [50].

Most sliding-window- or ensemble-based approaches need to scan data multiple times. In real big data tasks, it is often hoped that the stream data can be scanned only once and the storage size required by the learning process is independent from the data volume that could not be known before the stream ends. Recently, a simple yet effective approach based on the forgetting mechanism was proposed to tackle this issue [51]. The approach does not require prior knowledge about the change, and each instance can be discarded once scanned. Furthermore, inspired by an analysis in control theory [52], a high-probability estimate error analysis based on vector concentration demonstrates that the estimate error decreases until convergence.

Data distribution change can occur in more complicated situations, such as on data with rich structures. There are studies on this issue in multi-instance learning [53], where the key is to consider both the *bag*-level changes as well as instance-level changes [54].

## VARIED LEARNING OBJECTIVES

The performance of learning $f : \mathcal{X} \mapsto \mathcal{Y}$ can be measured by a performance measure $M_f$, such as accuracy, the F1 measure and Area under ROC Curve (AUC). Learning towards different objectives may lead to different models with different strengths. A model that is optimal on one measure does not mean that it can also be optimal on other measures. Close-environment machine learning studies generally assume that the $M_f$ that will be used to measure the learning performance should be invariant and known in advance. Unfortunately, this does not always hold. Taking, for example, the sensor dispatch task, initially many sensors are to be dispatched to pursue a high accuracy of monitoring, whereas after a relatively high accuracy has been achieved, other sensors are to be dispatched to ensure that the system continues to work with energy consumption as low as possible. To be able to handle varied objectives is desired for open ML.

Learning with varied learning objectives has rarely been studied. Here, the great challenge is to enable a trained machine learning model to switch smoothly from one objective to another, without requiring recollecting data to train a totally new model. There are studies on adapting a trained model to a new objective, based on the observation that many performance measures are relevant [55,56]; indeed, a large variety of performance measures can be optimized by exploiting non-linear auxiliary classifiers while keeping high computational efficiency [57]. This is also relevant to the strategy of *model reuse* [58,59].

In addition to switching from one objective to another, learning with varied learning objectives can also be accomplished by pursuing multiple objectives simultaneously, if these objectives are explicitly known in advance. This resorts to *Pareto optimization*. Formally, the goal is to optimize min $(M_1, M_2, \ldots, M_n)$, where the $M_i$ are the objectives; the smaller, the better. There usually does not exist a single model that is optimal on all objectives; instead, the goal is to seek the *Pareto front* consisting of solutions never inferior to other solutions on all objectives simultaneously. Figure 5 provides an illustration, where solutions $X$ and $Y$ are not inferior to any other solution on both objectives simultaneously, so they reside in the Pareto front.

Evolutional algorithms, such as genetic algorithms, have been commonly used for Pareto optimization in practice, though they are often criticized as they appear to be pure heuristics. It is worth mentioning that efforts have recently been made to try to establish a theoretical foundation for evolutionary learning [60], i.e. multi-objective machine learning

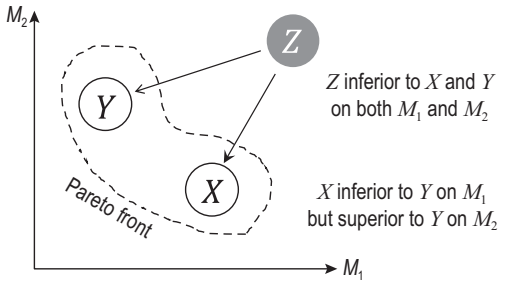


**Figure 5.** An illustration of the Pareto front.

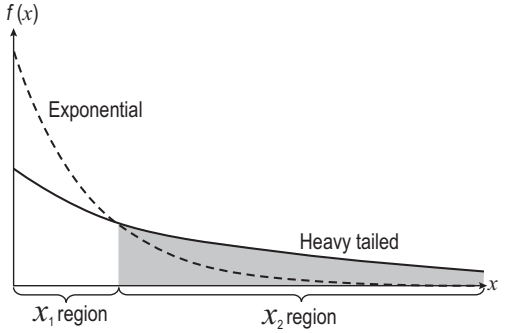


**Figure 6.** An illustration of a heavy-tailed distribution.

by exploiting evolutionary mechanisms, and it has been shown that the theoretical advances can help guide the design of powerful new algorithms, such as an evolutionary algorithm that provably achieves better approximation guarantees than conventional algorithms for the first time.

Besides explicit multiple objectives, implicit multiple objectives also require attention for open ML. For example, there are situations where users cannot express their objectives clearly, but can provide preference feedbacks like '$\text{model}_1$ is better for me than $\text{model}_2$.' It has been shown [61] that effective models can be obtained for such kinds of implicit objective by exploiting techniques such as *bag of words* [62], assuming that each implicit objective is inherently a kind of combination of element objectives.

## THEORETICAL ISSUES

Open ML is a new research direction and, therefore, too many theoretical issues are to be explored.

Among the four threads shown in Fig. 1, current techniques for learning with emerging new classes are mostly based on heuristics [6,27,29,32]. Note that, when *all data* are at hand, there are some theoretical results, e.g. when NewClasses exist in unlabeled data [7,63]; however, these results are not directly applicable when data are accumulated with time, where NewClass *emerges* in stream. There are some theoretical analyses on the proposed algorithms for learning with decremental/incremental features [35,36,38], but a thorough theoretical study is lacking. Learning with multi-objectives using evolutional mechanisms has its theoretical foundation established [60], but the varied learning objective issue as a whole is currently underexplored. Learning with changing distributions has been the subject of relatively more theoretical studies. For example, concept drift has a long thread of theoretical exploration [64–66], and some algorithms were proposed with theoretical analyses, from the view of mistake and loss bounds [67], stability analysis [68], generalization and regret analysis [58], etc. There have also been theoretical studies about relaxing the *i.i.d.* assumption [69–71].

Open ML is challenging mostly because we can hardly know what changes and how the changes will occur in advance. This is quite different from typical scenarios handled by reinforcement learning [72,73] where a learner interacts with the environment to explore the problem space. Once changes in open ML occur, previous exploration efforts of the reinforcement learner may become invalid since the problem space is altered by the changes. There are studies in which a reinforcement learner adapts to a changed environment [74,75], but the changes should not occur frequently or continuously.

Technically, in open ML one does not have data reflecting unknown changes in the initial training set, while an adequate model update must be conducted after receiving a few instances upon changes occur as soon as possible. From this aspect, open ML is somewhat relevant to *weakly supervised learning* [5]. However, in contrast to close-environment studies that emphasize on the *majority* examples and thus generally assume a *normal distribution*, in open ML the *minority* of examples or even those that have never been observed are much more important, though at the meantime a good performance on the majority is still demanded. Thus, instead of a normal distribution, it would be more favorable to consider *heavy-tailed distributions* (especially *fat-tailed distributions* where very rare events may cause extremely large losses rather than commonly studied *long-tailed distributions*) where the tails are not exponentially bounded, as illustrated in Fig. 6.

Evidently, one hopes that the learned model $h(\boldsymbol{x})$ satisfies

$$P\left(E_i(h) \leq \epsilon_i\right) \geq 1 - \delta_i, \tag{2}$$

where $E_i(h) = P_{\boldsymbol{x} \in \mathcal{X}_i}(h_i(\boldsymbol{x}) \neq y)$, $i \in \{1, 2\}$, $y$ is the ground-truth output of $\boldsymbol{x}$, $0 < \epsilon_1 \leq \epsilon_2 \leq \epsilon$, $\delta_1$, $\delta_2 < 1$. Intuitively, this reveals that the desired model should achieve excellent performance in the $\mathcal{X}_1$ region in Fig. 6 (i.e. the error should be smaller than

$\epsilon_1$ with a high probability), and satisfactory performance in the $\mathcal{X}_2$ region (i.e. the error $\epsilon_2$ must not be larger than $\epsilon$, though it can be larger than $\epsilon_1$). The rigid threshold $\epsilon$ ensures that the worst performance is bearable to the user no matter what changes occur. This is relevant to *safe learning* [76] in the weakly supervised scenario, and the principle *optimizing the worst-case performance after achieving a good average performance* can be helpful. Consequently, the total error is

$$E = E_1(h) + \gamma E_2(h), \tag{3}$$

where $\gamma$ is the coefficient to trade off the $\mathcal{X}_1$ and $\mathcal{X}_2$ regions, and can be set by the user according to the relative importance of these regions; $E$ is bounded by $(1+\gamma)\epsilon$ according to (2). The above understanding offers a perspective to regard the $\mathcal{X}_2$ region as a regularization force to learning in the $\mathcal{X}_1$ region.

Typical heavy-tailed distributions include the Pareto distribution, Cauchy distribution, etc. When they are assumed instead of the commonly used normal distribution, new challenges arise. For example, the central limit theorem does not hold, and frequent sample statistics, such as the popularly used sample mean and variance, would be misleading (i.e. they can be very different from the population mean and variance). These issues must be considered in open ML. For example, if the input and output spaces are heavy tailed, empirical risk minimization becomes invalid, since empirical risk is no longer a good approximation of risk [77]. This poses problems for learning algorithms, even for simple L1 regression [78].

Considering data accumulated with time, the performance measure requires attention. Here, the concern is that no matter what changes occur, the learning process is running as online learning [79,80]. In contrast to close-environment studies that assume a stationary online setting, open ML pays attention to the non-stationary online setting. As a consequence, rather than static regret that measures the performance by the cumulative loss of the learner against that of the best constant point chosen in hindsight, general dynamic regret [81] that compares the cumulative loss of the learner against any sequence of comparators is more reasonable. Optimal results have recently been reported on online convex optimization with various mechanisms [82–84] and bandit convex optimization [85]. A nearly minimax optimal solution to non-stationary linear bandits under a mild condition has been reported through a simple yet effective *restart* mechanism [86] with a scheduling scheme [87], which is more friendly to resource-constrained learning tasks than sliding window or forgetting mechanisms.

Open ML is also related to learning with noisy data, for which there are many theoretical studies, e.g. [88–92]. Note that, in contrast to close-environment studies where noises can be simply depressed by techniques such as smoothing, important signals in open ML might be hidden in signals that are regarded as noise, and rare important events might be depressed by oversimplified smoothing.

## CONCLUSION

This article briefly introduces some research advances in open-environment machine learning. It can hardly be a thorough review of all the relevant work, and is mostly a brief summary of the author's and his colleagues's exploration along this direction, emphasizing general principles and strategies rather than specific learning algorithms. Many strategies and ideas mentioned in this article can be realized with various learning techniques, possibly with different strengths to be explored in the future. Note that the varied issues are discussed separately in this article, while in real practice they often occur simultaneously. It is fundamentally important to enable machine learning models to achieve excellent performance in the usual case, while keeping satisfactory performance no matter what unexpected unfortunate issues occur. This is crucial for achieving robust artificial intelligence [93,94], and carries the desired properties of learnware [95].

## FUNDING

This work was supported by the National Natural Science Foundation of China (61921006) and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

***Conflict of interest statement.*** None declared.

## REFERENCES

1. Parmar J, Chouhan SS and Raychoudhury V *et al.* Open-world machine learning: applications, challenges, and opportunities. arXiv: 1409.1556v2.
2. Sehwag V, Bhagoji AN and Song L *et al.* Analyzing the robustness of open-world machine learning. In: *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. New York: Association for Computing Machinery, 2019, 105–16.
3. Pfülb B and Gepperth A. A comprehensive, application-oriented study of catastrophic forgetting in DNNs. In: *7th International Conference on Learning Representations (ICLR)*. New Orleans, LA: OpenReview.net, 2019.
4. Delange M, Aljundi R and Masana M *et al.* A continual learning survey: defying forgetting in classification tasks. *IEEE Trans Pattern Anal Mach Intell* 2022; doi: 10.1109/TPAMI.2021.3057446.

5. Zhou ZH. A brief introduction to weakly supervised learning. *Natl Sci Rev* 2018; **5**: 44–53.
6. Da Q, Yu Y and Zhou ZH. Learning with augmented class by exploiting unlabeled data. In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*. Palo Alto, CA: AAAI Press, 2014, 1760–6.
7. Zhang YJ, Zhao P and Ma L *et al*. An unbiased risk estimator for learning with augmented classes. In: *Advances in Neural Information Processing Systems 33 (NeurIPS)*. Cambridge, MA: MIT Press, 2020, 10247–58.
8. Socher R, Ganjoo M and Manning CD *et al*. Zero-shot learning through cross-modal transfer. In: *Advances in Neural Information Processing Systems 26 (NIPS)*. Cambridge, MA: MIT Press, 2013, 935–43.
9. Xian Y, Lampert CH and Schiele B *et al*. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Trans Pattern Anal Mach Intell* 2019; **41**: 2251–65.
10. Chen J, Geng Y and Chen Z *et al*. Knowledge-aware zero-shot learning: survey and perspective. In: *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*. San Francisco, CA: Margan Kaufmann, 2021, 4366–73.
11. Pan SJ and Yang Q. A survey on transfer learning. *IEEE Trans Knowl Data Eng* 2009; **22**: 1345–59.
12. Fumera G, Roli F and Giacinto G. Reject option with multiple thresholds. *Pattern Recognit* 2000; **33**: 2099–101.
13. Bartlett PL and Wegkamp MH. Classification with a reject option using a hinge loss. *J Mach Learn Res* 2008; **9**: 1823–40.
14. Geifman Y and El-Yaniv R. SelectiveNet: a deep neural network with an integrated reject option. In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. New York, NY: ACM, 2019, 2151–9.
15. Scheirer WJ, Rocha AR and Sapkota A *et al*. Towards open set recognition. *IEEE Trans Pattern Anal Mach Intell* 2013; **35**: 1757–72.
16. Geng C, Huang SJ and Chen S. Recent advances in open set recognition: a survey. *IEEE Trans Pattern Anal Mach Intell* 2020; **43**: 3614–31.
17. Utgoff PE. Incremental induction of decision trees. *Mach Learn* 1989; **4**: 161–86.
18. Syed N, Liu H and Sung K. Incremental learning with support vector machines. In: *Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Articial Intelligence (IJCAI-99)*. San Francisco, CA: Margan Kaufmann, 1999.
19. Giraud-Carrier C. A note on the utility of incremental learning. *AI Commun* 2000; **13**: 215–23.
20. He H, Chen S and Li K *et al*. Incremental learning from stream data. *IEEE Trans Neural Netw* 2011; **22**: 1901–14.
21. Zhou ZH and Chen Z. Hybrid decision tree. *Knowl Based Syst* 2002; **15**: 515–28.
22. Ozawa S, Toh SL and Abe S *et al*. Incremental learning of feature space and classifier for face recognition. *Neural Netw* 2005; **18**: 575–84.
23. Zhou G, Sohn K and Lee H. Online incremental feature learning with denoising autoencoders. In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cambridge, MA: MIT Press, 2012, 1453–61.
24. Masana M, Liu X and Twardowski B *et al*. Class-incremental learning: survey and performance evaluation on image classification. arXiv: 2010.15277v2.
25. Golub T, Slonim D and Tamayo P *et al*. Molecular classification of cancer: class discovery and class prediction by gene expression. *Science* 1999; **286**: 531–7.
26. Monti S, Tamayo P and Mesirov J *et al*. Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Mach Learn* 2003; **52**: 91–118.
27. Mu X, Ting KM and Zhou ZH. Classification under streaming emerging new classes: a solution using completely-random trees. *IEEE Trans Knowl Data Eng* 2017; **29**: 1605–18.
28. Liu FT, Ting KM and Zhou ZH. Isolation forest. In: *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*. Piscataway, NJ: IEEE Computer Society, 2008, 413–22.
29. Mu X, Zhu F and Du J *et al*. Streaming classification with emerging new class by class matrix sketching. In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*. Palo Alto, CA: AAAI Press, 2017, 2373–9.
30. Zhu Y, Ting KM and Zhou ZH. Discover multiple novel labels in multi-instance multi-label learning. In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*. Palo Alto, CA: AAAI Press, 2017, 2977–84.
31. Zhu Y, Ting KM and Zhou ZH. New class adaptation via instance generation in one-pass class incremental learning. In: *International Conference on Data Mining (ICDM)*. Piscataway, NJ: IEEE Computer Society, 2017, 1207–12.
32. Zhu Y, Ting KM and Zhou ZH. Multi-label learning with emerging new labels. *IEEE Trans Knowl Data Eng* 2018; **30**: 1901–14.
33. Faria ER, Gonçalves IR and Gama J *et al*. Evaluation of multiclass novelty detection algorithms for data streams. *IEEE Trans Knowl Data Eng* 2015; **27**: 2961–73.
34. Zhao P, Zhang YJ and Zhou ZH. Exploratory machine learning with unknown unknowns. In: *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*. Palo Alto, CA: AAAI Press, 2021, 10999–1006.
35. Hou C and Zhou ZH. One-pass learning with incremental and decremental features. *IEEE Trans Pattern Anal Mach Intell* 2018; **40**: 2776–92.
36. Hou BJ, Zhang L and Zhou ZH. Learning with feature evolvable streams. In: *Advances in Neural Information Processing Systems 30 (NIPS)*. Cambridge, MA: MIT Press, 2017, 1417–27.
37. Zhou ZH. *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL: Chapman & Hall/CRC, 2012.
38. Hou BJ, Zhang L and Zhou ZH. Prediction with unpredictable feature evolution. *IEEE Trans Neural Netw Learn Sys* 2021; doi: 10.1109/TNNLS.2021.3071311.
39. Zhang ZY, Zhao P and Jiang Y *et al*. Learning with feature and distribution evolvable streams. In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. New York, NY: ACM, 2020, 11317–27.
40. Hu C, Chen Y and Peng X *et al*. A novel feature incremental learning method for sensor-based activity recognition. *IEEE Trans Knowl Data Eng* 2019; **31**: 1038–50.
41. Sugiyama M and Kawanabe M. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. Cambridge, MA: MIT Press, 2012.
42. Gama J, Zliobaite I and Bifet A *et al*. A survey on concept drift adaptation. *ACM Comput Surv* 2014; **46**: 44.
43. III HD and Marcu D. Domain adaptation for statistical classifiers. *J Artif Intell Res* 2006; **26**: 101–26.
44. Ben-David S, Lu T and Luu T *et al*. Impossibility theorems for domain adaptation. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cambridge, MA: MIT Press, 2010, 129–36.
45. Kouw WM and Loog M. A review of domain adaptation without target labels. *IEEE Trans Pattern Anal Mach Intell* 2005; **43**: 766–85.
46. Klinkenberg R and Joachims T. Detecting concept drift with support vector machines. In: *Proceedings of the 17th International Conference on Machine Learning (ICML)*. New York, NY: ACM, 2000, 487–94.

47. Kuncheva LI and Zliobaite I. On the window size for classification in changing environments. *Intell Data Anal* 2009; **13**: 861–72.
48. Koychev I. Gradual forgetting for adaptation to concept drift. In: *Proceedings of ECAI 2000 Workshop Current Issues in Spatio-Temporal Reasoning*. Amsterdam: IOS Press, 2000, 101–6.
49. Anagnostopoulos C, Tasoulis DK and Adams NM *et al*. Online linear and quadratic discriminant analysis with adaptive forgetting for streaming classification. *Stat Anal Data Min* 2012; **5**: 139–66.
50. Gomes HM, Barddal JP and Enembreck F *et al*. A survey on ensemble learning for data stream classification. *ACM Comput Surv* 2017; **50**: 23.
51. Zhao P, Wang X and Xie S *et al*. Distribution-free one-pass learning. *IEEE Trans Knowl Data Eng* 2021; **33**: 951–63.
52. Guo L, Ljung L and Priouret P. Performance analysis of the forgetting factor RLS algorithm. *Int J Adapt Control Signal Process* 1993; **7**: 525–38.
53. Foulds J and Frank E. A review of multi-instance learning assumptions. *Knowl Eng Rev* 2010; **25**: 1–25.
54. Zhang YL and Zhou ZH. Multi-instance learning with key instance shift. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. San Francisco, CA: Margan Kaufmann, 2017, 3441–7.
55. Cortes C and Mohri M. AUC optimization vs. error rate minimization. In: *Advances in Neural Information Processing Systems 16 (NIPS)*. Cambridge, MA: MIT Press, 2004, 313–20.
56. Wu XZ and Zhou ZH. A unified view of multi-label performance measures. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. New York, NY: ACM, 2017, **70**: 3780–8.
57. Li N, Tsang IW and Zhou ZH. Efficient optimization of performance measures by classifier adaptation. *IEEE Trans Pattern Anal Mach Intell* 2013; **35**: 1370–82.
58. Zhao P, Cai LW and Zhou ZH. Handling concept drift via model reuse. *Mach Learn* 2020; **109**: 533–68.
59. Wu XZ, Liu S and Zhou ZH. Heterogeneous model reuse via optimizing multiparty multiclass margin. In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. New York, NY: ACM, 2019, 6840–9.
60. Zhou ZH, Yu Y and Qian C. *Evolutionary Learning: Advances in Theories and Algorithms*. Berlin: Springer, 2019.
61. Ding YX and Zhou ZH. Preference based adaptation for learning objectives. *Advances in Neural Information Processing Systems 31 (NeurIPS)*. Cambridge, MA: MIT Press, 2018, 7839–48.
62. Zhang Y, Jin R and Zhou ZH. Understanding bag-of-words model: a statistical framework. *Int J Mach Learn Cybern* 2010; **1**: 43–52.
63. Liu S, Garrepalli R and Hendrycks D *et al*. PAC guarantees and effective algorithms for detecting novel categories. *J Mach Learn Res* 2022; **23**: 1–47.
64. Helmbold DP and Long PM. Tracking drifting concepts by minimizing disagreements. *Mach Learn* 1994; **14**: 27–45.
65. Crammer K, Mansour Y and Even-Dar E *et al*. Regret minimization with concept drift. In: *Proceedings of the 23rd Conference on Learning Theory (COLT)*. New York, NY: ACM, 2010, 168–80.
66. Mohri M and Medina AM. New analysis and algorithm for learning with drifting distributions. In: *Proceedings of the 23rd International Conference on Algorithmic Learning Theory (ALT)*. Berlin: Springer, 2012, 124–38.
67. Kolter JZ and Maloof MA. Using additive expert ensembles to cope with concept drift. In: *Proceedings of the 22nd International Conference on Machine Learning (ICML)*. New York, NY: ACM, 2005, 449–56.
68. Harel M, Mannor S and El-Yaniv R *et al*. Concept drift detection through resampling. In: *Proceedings of the 31st International Conference on Machine Learning (ICML)*. New York, NY: ACM, 2014, 1009–17.
69. Mohri M and Rostamizadeh A. Rademacher complexity bounds for non-i.i.d. processes. In: *Advances in Neural Information Processing Systems 21 (NIPS)*. Cambridge, MA: MIT Press, 2008, 1097–104.
70. Pentina A and Lampert CH. Lifelong learning with non-i.i.d. tasks. In: *Advances in Neural Information Processing Systems 28 (NIPS)*. Cambridge, MA: MIT Press, 2015, 1540–8.
71. Gao W, Niu XY and Zhou ZH. Learnability of non-i.i.d. In: *Proceedings of the 8th Asian Conference on Machine Learning (ACML)*. New York, NY: ACM, 2016, 158–73.
72. Sutton RS and Barto AG. *Reinforcement Learning: An Introduction*, 2nd edn. Cambridge, MA: MIT Press, 2012.
73. Majid AY, Saaybi S and van Rietbergen T *et al*. Deep reinforcement learning versus evolution strategies: a comparative survey. arXiv: 2110.01411v1.
74. Zhang C, Yu Y and Zhou ZH. Learning environmental calibration actions for policy self-evolution. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*. San Francisco, CA: Margan Kaufmann, 2018, 3061–7.
75. Chen SY, Yu Y and Da Q *et al*. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In: *Proceedings of the 24th {ACM} {SIGKDD} International Conference on Knowledge Discovery & Data Mining (KDD)*. New York, NY: ACM, 2018, 1187–96.
76. Li YF and Zhou ZH. Towards making unlabeled data never hurt. *IEEE Trans Pattern Anal Mach Intell* 2015; **37**: 175–88.
77. Catoni O. Challenging the empirical mean and empirical variance: a deviation study. *Ann Inst Henri Poincare* 2012; **48**: 1148–85.
78. Zhang L and Zhou ZH. L1-regression with heavy-tailed distributions. In: *Advances in Neural Information Processing Systems 31 (NeurIPS)*. Cambridge, MA: MIT Press, 2018, 1084–94.
79. Cesa-Bianchi N and Lugosi G. *Prediction, Learning, and Games*. Cambridge: Cambridge University Press, 2006.
80. Shalev-Shwartz S. Online learning and online convex optimization. *Found Trends Mach Learn* 2011; **4**: 107–94.
81. Zinkevich M. Online convex programming and generalized infinitesimal gradient ascent. In: *Proceedings of the 20th International Conference on Machine Learning (ICML)*. New York, NY: ACM, 2003, 928–36.
82. Zhang L, Lu S and Zhou ZH. Adaptive online learning in dynamic environments. *Advances in Neural Information Processing Systems 31 (NeurIPS)*. La Jolla, CA: NeurIPS, 2018, 1330–40.
83. Zhao P, Zhang YJ and Zhang L *et al*. Dynamic regret of convex and smooth functions. *Advances in Neural Information Processing Systems 33 (NeurIPS)*. Cambridge, MA: MIT Press, 2020, 12510–20.
84. Zhao P, Wang YX and Zhou ZH. Non-stationary online learning with memory and non-stochastic control. In: *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cambridge, MA: MIT Press, 2022, 2101–33.
85. Zhao P, Wang G and Zhang L *et al*. Bandit convex optimization in non-stationary environments. *J Mach Learn Res* 2021; **22**: 1–45.
86. Zhao P, Zhang L and Jiang Y *et al*. A simple approach for non-stationary linear bandits. In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cambridge, MA: MIT Press, 2020, 746–55.
87. Wei CY and Luo H. Non-stationary reinforcement learning without prior knowledge: an optimal black-box approach. In: *Proceedings of 34th Conference on Learning Theory (COLT)*. New York, NY: ACM, 2021, 4300–54.
88. Angluin D and Laird P. Learning from noisy examples. *Mach Learn* 1988; **2**: 343–70.

89. Blum A, Kalai A and Wasserman H. Noise-tolerant learning, the parity problem, and the statistical query model. *J ACM* 2003; **50**: 506–19.
90. Natarajan N, Dhillon IS and Ravikumar PK *et al.* Learning with noisy labels. In: *Advances in Neural Information Processing Systems 26 (NIPS).* Cambridge, MA: MIT Press, 2013, 1196–204.
91. Gao W, Wang L and Li YF *et al.* Risk minimization in the presence of label noise. In: *Proceedings of the 13th AAAI Conference on Artificial Intelligence.* Palo Alto, CA: AAAI Press, 2016, 1575–81.
92. Gao W, Zhang T and Yang BB *et al.* On the noise estimation statistics. *Artif Intell* 2021; **293**: 103451.
93. Dietterich TG. Steps toward robust artificial intelligence. *AI Mag* 2017; **38**: 3–24.
94. Dietterich TG. Robust artificial intelligence and robust human organizations. *Front Comput Sci* 2019; **13**: 1–3.
95. Zhou ZH. Learnware: on the future of machine learning. *Front Comput Sci* 2016; **10**: 589–90.