

Open-environment Machine Learning

Zhi-Hua Zhou

*National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210023, China
zhouzh@nju.edu.cn*

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* (Open ML) 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, varied learning objectives, and discusses some theoretical issues.

1. 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 *instance*) describing appearance of the

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 $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, where $\mathbf{x}_i \in \mathcal{X}$ 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 success of machine learning are mostly on tasks assuming *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.

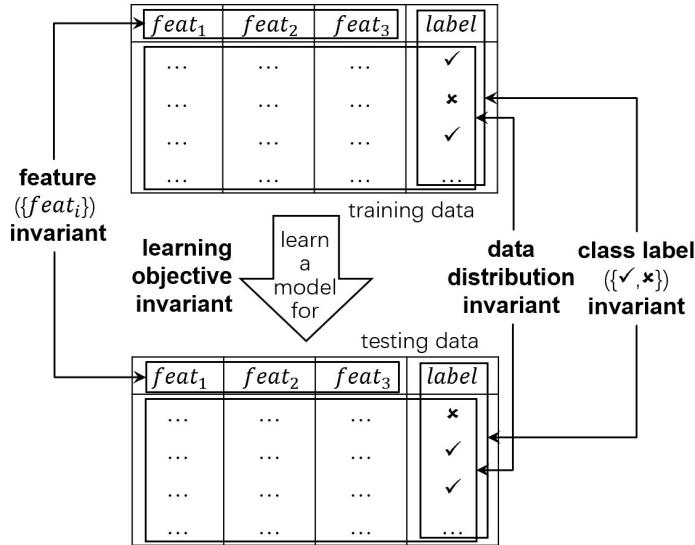


Figure 1: Typical invariant factors assumed in close-environment machine learning.

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*¹, or briefly, *Open Learning* or *Open ML*.

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, whereas 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* [59] 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 huge computational and storage costs. Though there are studies like *continual learning* [13] 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 are considerable research efforts on Open ML. This article will briefly introduce 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.

¹The name “Open-world Machine Learning” was used to refer machine learning with unseen class [57] or out-of-distribution (OOD) data [61]. In fact, it is not beyond closed-environment studies if the unseen class is *known* in advance, and related to Section 2 if unseen class is *unknown*. OOD is related to Section 4, though concerning only a different distribution is simpler than distribution changing with time.

2. Emerging New Classes

Close-environment machine learning studies generally assume that the class label of any unseen instance \hat{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 sending 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 [89] 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 [12]. Actually, the distribution of NewClass can be approximated by separating the distribution of known classes for that of the unlabeled data [76]. Such strategies, however, are not directly applicable when data are accumulated with time.

Consider the following setting of learning with emerging new class. A machine learning model is trained from some initial training data and then deployed to handle unseen instances coming like a stream. For incoming instances of known classes, the trained model should be able to make correct predictions. For incoming instances of unknown class, 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 human in the loop.

In the first glance, learning with emerging new class seems relevant to *zero-shot learning*, a hot topic in image classification, aiming to classifying visual classes that did not occur in training data set [63, 71, 8]. 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 [56]; in contrast, learning with emerging new class is a general machine learning setting which 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 training data, whereas learning with emerging new class are tackling the grand challenge that the unseen classes are unknown. Thus, approaches for learning with emerging new class can be more general, and can be converted and applied to zero-shot learning.

Classification with *reject* option [19, 3, 24] aims to avoiding unconfident predictions that are likely to be incorrect, assuming all classes known in advance. *Open set recognition/classification* [60, 25] extends reject option to consider the possibility that unknown class may occur in 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 unknown class, by assuming the availability of side information like aforementioned in zero-shot learning [25], whereas learning with emerging new class is a general machine learning setting which does not assume such external knowledge.

Learning with 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 [67, 66, 26, 31], mostly concerning on increment of training examples, i.e., E-IL (example-incremental learning)

defined in [90]. Besides E-IL, the other two types of incremental learning defined in [90] are A-IL (attribute-incremental learning) and C-IL (class-incremental learning). A-IL concerns about feature increment, related to what will be discussed in Section 3 of this article, though previous studies generally devoted to selecting adequate feature space given all data/features in advance [55, 86]. C-IL concerns about class increment, related to learning with emerging new class, though previous studies concerned little about the identification of NewClass and generally assumed that the incremental class is known [48].

Class discovery [27, 51] tries to discover rare classes, as a separate process from class prediction. As mentioned above, learning with 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 [52] to learning with 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, Figure 2(a) provides an illustration where 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 Figure 2(b). 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 *isolation forest* [45].

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 [52], which does not even need any storage for known class data. Alternatives include

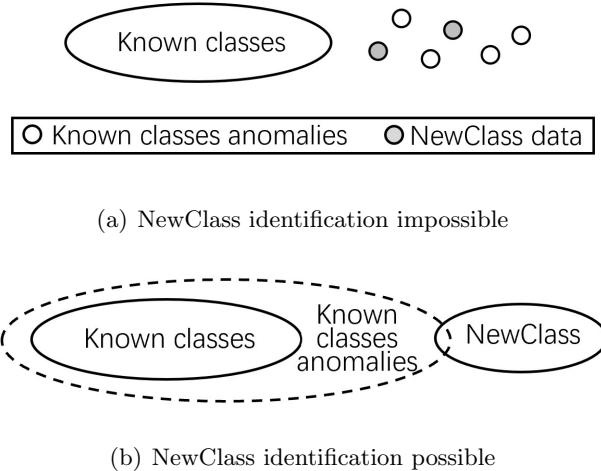


Figure 2: NewClass identification not always possible.

techniques that can localize the influence of difference classes such that changes according to NewClass won't significantly affect known classes, such as approaches based on global and local sketching [53].

If there are more than one new classes, the clustering structure of NewClass data can be exploited [92]. 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 enable the model update based on fewer NewClass data [93]. *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 [94]. A relevant topic is to examine what known classes are closely related to the NewClass, and an evaluation methodology concerning the mapping from NewClass to known classes has been developed [17].

There are situations where some NewClass instances appeared in training data but misperceived as known class instances, possibly due to insufficiency of feature information. This is even more challenging, with only one very preliminary study having been carried out [85].

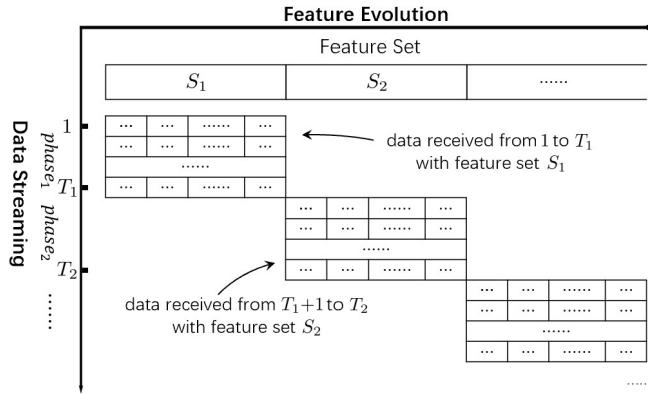
3. Decremental/Incremental Features

Close-environment machine learning studies generally assume that all possible instances, including unseen ones, reside in the same feature space, i.e., $\hat{\mathbf{x}} \in \mathcal{X}$. Unfortunately, this does not always hold. Taking for example the forest disease monitor mentioned in Section 2, some sensors could not continue sending signal due to battery exhausted, leading to decremental features, whereas some new sensors can be deployed, leading to incremental features. To be able to handle $\hat{\mathbf{x}} \in \hat{\mathcal{X}} \neq \mathcal{X}$ is also a requirement for Open ML. Note that in contrast to varied classes where only emerging new class requires special treatment whereas disappeared class can be simply ignored, both decremental and incremental features require attention since feature decrement can lead to seriously downgraded performance.

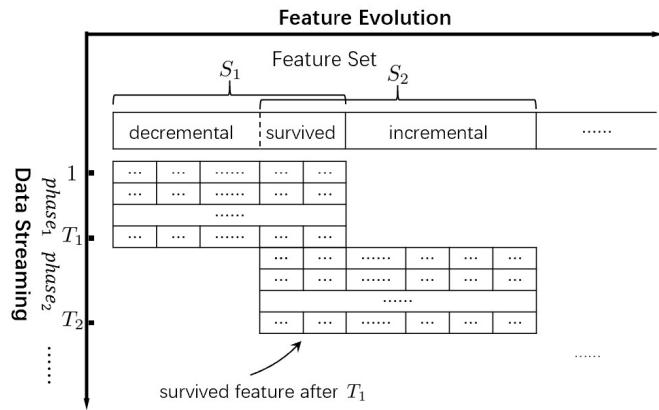
Consider the following setting of learning with decremental/incremental features. A machine learning model is trained from some initial training data and then deployed to handle unseen data coming like a stream, with decremental and/or incremental features. For incoming testing data, the model should be able to make correct predictions; for incoming additional training data, the model should be able to be refined accordingly. Ideally, it is desired 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 which is able to benefit from $\mathbf{x} \in \mathcal{X}$ for $\hat{\mathbf{x}} \in \hat{\mathcal{X}} \neq \mathcal{X}$, because machine learning is 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 Figure 3(a), if the feature spaces of phase₁ data (i.e., $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{T_1}, y_{T_1})\}$) and phase₂ data (i.e., $\{(\mathbf{x}_{T_1+1}, y_{T_1+1}), \dots, (\mathbf{x}_{T_2}, y_{T_2})\}$) are totally different, then the model trained in phase₁ are helpless for phase₂, and a new model has to be trained from scratch based on feature set S_2 for phase₂.

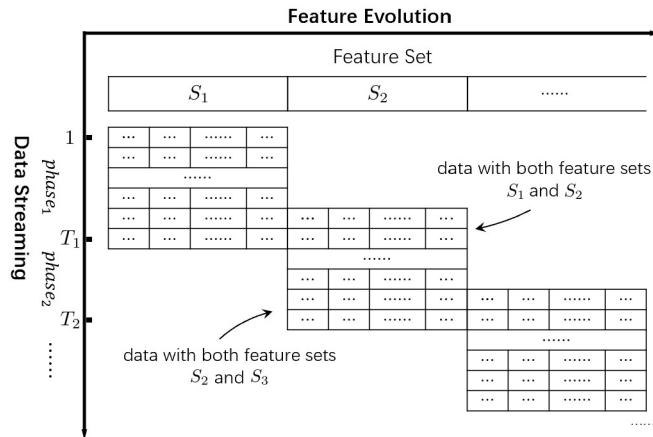
Fortunately, in many real tasks it is often the case that $\hat{\mathcal{X}} \cap \mathcal{X} \neq \emptyset$. In other words, there are features of phase₁ survive to be active in phase₂ though many other features vanish, as illustrated in Figure 3(b). For example, different sensors may have different battery lifetime,



(a) Phase₁ helpless to phase₂



(b) Phase₁ helpful to phase₂ through survived features



(c) Phase₁ helpful to phase₂ through shared instances

Figure 3: Helpless/helpful feature evolution.

and thus, some old sensors are still working after new sensors being deployed. Formally, in phase₁, $\mathcal{X} = \mathcal{X}^{de} \cup \mathcal{X}^s$ where \mathcal{X}^{de} and \mathcal{X}^s denote the decremental and survived feature sets, respectively; in phase₂, $\hat{\mathcal{X}} = \mathcal{X}^s \cup \mathcal{X}^{in}$ where \mathcal{X}^{in} denotes the incremental feature set. As \mathcal{X}^s is shared in both phases, in addition to training a model₁ from \mathcal{X} , a model₂ based on \mathcal{X}_s can be trained in phase₁, e.g., by [35]:

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

where $\langle \cdot, \cdot \rangle$ is inner product, \mathbf{w} and \mathbf{w}^s are parameters of model₁ and model₂, respectively, while $\alpha, \gamma > 0$ are the regularization coefficients. Such a process works like ‘compressing’ helpful predictive information from model₁ in \mathcal{X} to model₂ in \mathcal{X}^s . Then, in phase₂, in addition to the model₃ trained based on $\hat{\mathcal{X}}$, the model₂ trained in phase₁ can still be used. Thus, the prediction in phase₂ can be made by combining model₃ fed with $\hat{\mathbf{x}}$ and model₂ fed with the \mathcal{X}^s part of $\hat{\mathbf{x}}$. In this way, some experience learned from phase₁ can be exploited in phase₂ through the use of model₂.

Interestingly, even when $\hat{\mathcal{X}} \cap \mathcal{X} = \emptyset$, there are cases where it is possible to enable phase₁ learning to be helpful to phase₂, in particular, when feature increment occurs earlier than feature decrement [33], e.g., new sensors are deployed slightly before old sensors’ battery exhausted. As illustrated in Figure 3(c), in this situation there exists a small set of data with both sets of features that can help building a mapping $\psi : \hat{\mathcal{X}} \mapsto \mathcal{X}$. Thus, though $\hat{\mathbf{x}}$ received in phase₂ with features of $\hat{\mathcal{X}}$ only, model₁ learned in phase₁ can still be exploited by feeding it with $\psi(\hat{\mathbf{x}})$. Then, phase₂ prediction can be made by combining model₁ with model₂ 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 be even enhanced by exploiting ensemble learning [87].

Recently, there are studies about learning with feature decrement/increment at unpredictable phase [34], along with data distribution changes [78], etc., and applications such as sensor-based activity recognition [36].

4. 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 Section 2 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 exhibiting that ignoring the data distribution change may lead to seriously downgraded performance.

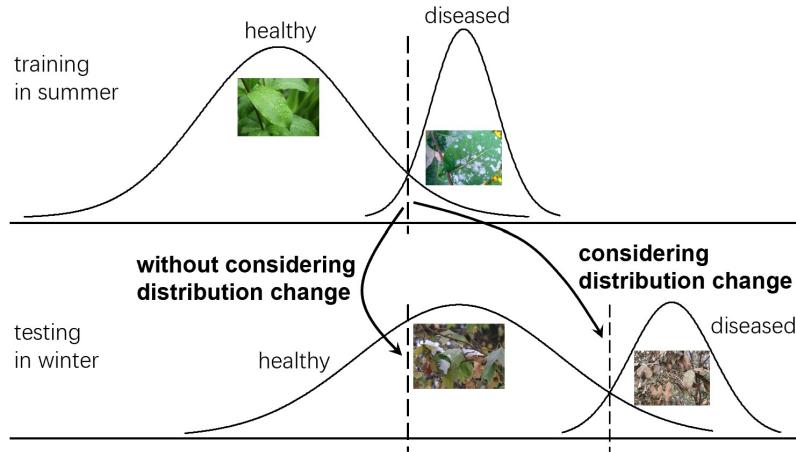


Figure 4: Data distribution change cannot be ignored.

There have been plentiful studies concerning about training/testing data distribution change. For example, *prior probability shift* and *covariate shift* [64] concern about $P_{\text{train}}(y|\mathbf{x}) =$

$P_{\text{test}}(y|\mathbf{x})$ but $P_{\text{train}}(y) \neq P_{\text{test}}(y)$ and $P_{\text{train}}(\mathbf{x}) \neq P_{\text{test}}(\mathbf{x})$, respectively, whereas *concept drift* [20] concerns about $P_{\text{train}}(y|\mathbf{x}) \neq P_{\text{test}}(y|\mathbf{x})$. Many relevant studies have been conducted under the umbrella of domain adaptation [37, 4, 40] or transfer learning [56]. Note that in steam situation, data distribution change can occur in any phase of the stream, not limited to the testing phase. To be able to handle various kind 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 data distribution can change arbitrarily in every moment without knowledge about how it could change. Fortunately, in many real tasks it is reasonable to assume that *current observation has close relation with recent observations*; in other words, the current instance and the most recent ones are usually from similar or even identical distribution, 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 [38, 42]. Forgetting based approaches assign a weight to each instance, and downweight old instances according to their age [41, 1]. Ensemble [87] based approaches add/remove component learners in the ensemble adaptively , and dynamically adjust the weights of learners for incoming instances [28].

Most sliding window or ensemble based approaches need to scan data for 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 which could not be known before stream ends. Recently, a simple yet effective approach based on forgetting mechanism was proposed to tackle this issue [81]. 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 [29], 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 [18], where the

key is to consider both the *bag*-level changes as well as instance-level changes [77].

5. 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, F1 measure, and Area under ROC Curve (AUC). Learning towards different objectives may lead to different models with different strengths. A model which 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 which 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 the system continue to work with energy consumption as low as possible. To be able to handle the 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 [10, 70]; indeed, a large variety of performance measures can be optimized by exploiting nonlinear auxiliary classifiers while keeping high computational efficiency [43]. This is also relevant to the strategy of *model reuse* [79, 69].

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 $\min(M_1, M_2, \dots, M_n)$ where M_i are the objectives; the smaller the better. There usually does not exist a single model which is optimal on all objectives; instead, the goal is to seek the *pareto front* consisting of solutions never inferior to other solution on all objectives simultaneously. Figure 5 provides an illustration, where the solu-

tions X and Y are not inferior to any other solution on both objectives simultaneously, so they reside in the pareto front.

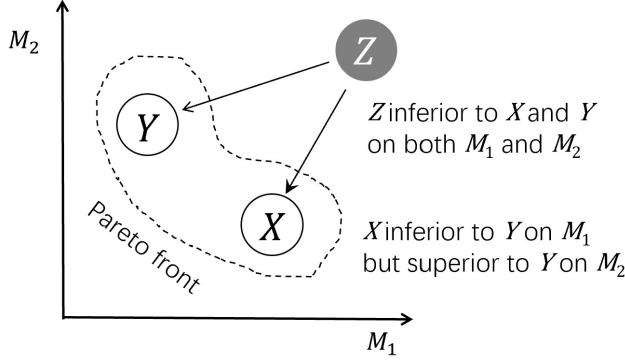


Figure 5: An illustration of pareto front.

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 recently, there are efforts trying to establish theoretical foundation of evolutionary learning [91], i.e., multi-objective machine learning 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 which is provable to achieve 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 could not express their objectives clearly, but can provide preference feedback like ‘model₁ is better for me than model₂’. It has been shown [16] that effective models can be obtained for such kind of implicit objectives by exploiting techniques like *bag of words* [75], assuming that each implicit objective is inherently a kind of combination of element objectives.

6. 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 Figure 1, current techniques for learning with emerging new classes are mostly based on heuristics [12, 52, 53, 94]. Note that when *all data* are at hand there are some theoretical results, e.g., when NewClasses exist in unlabeled data [76, 46]; however, these results are not directly applicable when data are accumulated with time, where NewClass *emerging* in stream. There are some theoretical analyses on the proposed algorithms for learning with decremental/incremental features [35, 33, 34], but a thorough theoretical study is lacking. Learning with multi-objectives using evolutional mechanisms has its theoretical foundation being established [91], but the varied learning objective issue as a whole is underexplored yet. Learning with changing distributions is with relatively more theoretical studies. For example, concept drift has a long thread of theoretical exploration [32, 11, 49], and some algorithms were proposed with theoretical analyses, from view of mistake and loss bounds [39], stability analysis [30], generalization and regret analysis [79], etc. There are also theoretical studies about relaxing the *i.i.d.* assumption [50, 58, 21].

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 [65, 47] where a learner interacts with the environment to explore the problem space. Once changes concerned 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 about adapting a reinforcement learner to a changed environment [72, 9], 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 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* [89]. However, in contrast to close-environment studies that

emphasize on the *majority* examples and thus generally assume *normal distribution*, in Open ML the *minority* 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 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 Figure 6.

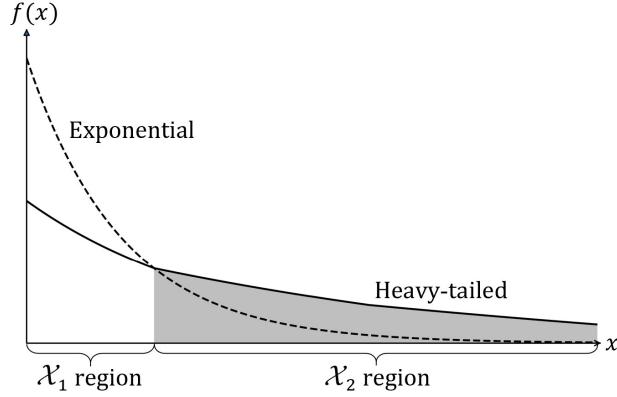


Figure 6: An illustration of heavy-tailed distribution.

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

$$P(E_i(h) \leq \epsilon_i) \geq 1 - \delta_i , \quad (2)$$

where $E_i(h) = P_{\mathbf{x} \in \mathcal{X}_i} (h_i(\mathbf{x}) \neq y)$, $i \in \{1, 2\}$, y is ground-truth output of \mathbf{x} , $0 < \epsilon_1 \leq \epsilon_2 \leq \epsilon$, $\delta_1, \delta_2 < 1$. Intuitively, this explains that the desired model should achieve excellent performance in \mathcal{X}_1 region in Figure 6 (i.e., the error should be smaller than ϵ_1 with a high probability), and satisfactory performance in \mathcal{X}_2 region (i.e., the error ϵ_2 must not be larger than ϵ though it can be larger than ϵ_1). The rigid threshold ϵ is to ensure that the worst performance can be bearable to user no matter what changes occur. This is relevant to *safe learning* [44] in 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

$$E = E_1(h) + \gamma E_2(h) , \quad (3)$$

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

Typical heavy-tailed distributions include 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 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 [6]. This poses problem for learning algorithms, even for simple L1-regression [74].

Considering data accumulated with time, performance measure requires attention. Here, the concern is that no matter what changes will occur, the learning process is running as online learning [7, 62]. In contrast to close-environment studies that assume stationary online setting, Open ML pays attention to non-stationary online setting. As a consequence, rather than static regret which measures the performance by the cumulative loss of the learner against that of the best constant point chosen in hindsight, general dynamic regret [95] which compares the cumulative loss of the learner against any sequence of comparators is more reasonable. Optimal results have been reported recently on online convex optimization with various mechanisms [73, 84, 82] and bandit convex optimization [80]. A nearly minimax optimal solution to non-stationary linear bandits under mild condition has been reported through a simple yet effective *restart* mechanism [83] with a scheduling scheme [68], 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., [2, 5, 54, 22, 23]. 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.

7. Conclusion

This article briefly introduces some research advances in open environment machine learning. It can hardly be a thorough review of all relevant work, and is mostly a brief summary of author and his colleagues's exploration along this direction, emphasizing on 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 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 usual case while keeping satisfactory performance no matter what unexpected unfortunate issues occur. This is crucial for achieving robust artificial intelligence [14, 15], and carries desired properties of learnware [88].

References

- [1] C. Anagnostopoulos, D. K. Tasoulis, N. M. Adams, N. G. Pavlidis, and D. J. Hand. Online linear and quadratic discriminant analysis with adaptive forgetting for streaming classification. *Statistical Analysis and Data Mining*, 5(2):139–166, 2012.
- [2] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.
- [3] P. L. Bartlett and M. H. Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(59):1823–1840, 2008.
- [4] S. Ben-David, T. Lu, T. Luu, and D. Pal. Impossibility theorems for domain adaptation. In *AISTATS*, pages 129–136, 2010.
- [5] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, 2003.

- [6] O. Catoni. Challenging the empirical mean and empirical variance: A deviation study. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 48(4):1148–1185, 2012.
- [7] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.
- [8] J. Chen, Y. Geng, Z. Chen, I. Horrocks, J. Z. Pan, and H. Chen. Knowledge-aware zero-shot learning: Survey and perspective. In *IJCAI*, pages 4366–4373, 2021.
- [9] S.-Y. Chen, Y. Yu, Q. Da, J. Tan, H.-K. Huang, and H.-H. Tang. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In *KDD*, pages 1187–1196, 2018.
- [10] C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In *NIPS*, pages 313–320, 2004.
- [11] K. Crammer, Y. Mansour, E. Even-Dar, and J. W. Vaughan. Regret minimization with concept drift. In *COLT*, pages 168–180, 2010.
- [12] Q. Da, Y. Yu, and Z.-H. Zhou. Learning with augmented class by exploiting unlabeled data. In *AAAI*, pages 1760–1766, 2014.
- [13] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Trans. Pattern Analysis and Machine Intelligence*, page 10.1109/TPAMI.2021.3057446, 2022.
- [14] T. G. Dietterich. Steps toward robust artificial intelligence. *AI Magazine*, 38(3):3–24, 2017.
- [15] T. G. Dietterich. Robust artificial intelligence and robust human organizations. *Frontiers of Computer Science*, 13(1):1–3, 2019.
- [16] Y.-X. Ding and Z.-H. Zhou. Preference based adaptation for learning objectives. In *NeurIPS*, pages 7839–7848, 2018.

- [17] E. R. Faria, I. R. Gonçalves, J. Gama, and A. C. P. L. F. Carvalho. Evaluation of multiclass novelty detection algorithms for data streams. *IEEE Trans. Knowledge and Data Engineering*, 27(11):2961–2973, 2015.
- [18] J. Foulds and E. Frank. A review of multi-instance learning assumptions. *Knowledge Engineering Review*, 25(1):1–25, 2010.
- [19] G. Fumera, F. Roli, and G. Giacinto. Reject option with multiple thresholds. *Pattern Recognition*, 33(12):2099–2101, 2000.
- [20] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):Article 44, 2014.
- [21] W. Gao, X.-Y. Niu, and Z.-H. Zhou. Learnability of non-i.i.d. In *ACML*, pages 158–173, 2016.
- [22] W. Gao, L. Wang, Y.-F. Li, and Z.-H. Zhou. Risk minimization in the presence of label noise. In *AAAI*, pages 1575–1581, 2016.
- [23] W. Gao, T. Zhang, B.-B. Yang, and Z.-H. Zhou. On the noise estimation statistics. *Artificial Intelligence*, 293:Article 103451, 2021.
- [24] Y. Geifman and R. El-Yaniv. SelectiveNet: A deep neural network with an integrated reject option. In *ICML*, pages 2151–2159, 2019.
- [25] C. Geng, S.-J. Huang, and S. Chen. Recent advances in open set recognition: A survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 43(10):3614–3631, 2020.
- [26] C. Giraud-Carrier. A note on the utility of incremental learning. *AI Communications*, 13(4):215–223, 2000.
- [27] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression. *Science*, 286(5439):531–537, 1999.

- [28] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet. A survey on ensemble learning for data stream classification. *ACM Computing Surveys*, 50(2):23:1–23:36, 2017.
- [29] L. Guo, L. Ljung, and P. Priouret. Performance analysis of the forgetting factor RLS algorithm. *International Journal of Adaptive Control Signal Processing*, 7(6):525–538, 1993.
- [30] M. Harel, S. Mannor, R. El-Yaniv, and K. Crammer. Concept drift detection through resampling. In *ICML*, pages 1009–1017, 2014.
- [31] H. He, S. Chen, K. Li, and X. Xu. Incremental learning from stream data. *IEEE Trans. Neural Networks*, 22(12):1901–1914, 2011.
- [32] D. P. Helmbold and P. M. Long. Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14(1):27–45, 1994.
- [33] B.-J. Hou, L. Zhang, and Z.-H. Zhou. Learning with feature evolvable streams. In *NIPS*, pages 1417–1427, 2017.
- [34] B.-J. Hou, L. Zhang, and Z.-H. Zhou. Prediction with unpredictable feature evolution. *IEEE Trans. Neural Networks and Learning Systems*, page DOI:10.1109/TNNLS.2021.3071311, 2021.
- [35] C. Hou and Z.-H. Zhou. One-pass learning with incremental and decremental features. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 40(11):2776–2792, 2018.
- [36] C. Hu, Y. Chen, X. Peng, H. Yu, C. Gao, and L. Hu. A novel feature incremental learning method for sensor-based activity recognition. *IEEE Trans. Knowledge and Data Engineering*, 31(6):1038–1050, 2019.
- [37] H. Daumé III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006.
- [38] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *ICML*, pages 487–494, 2000.

- [39] J. Z. Kolter and M. A. Maloof. Using additive expert ensembles to cope with concept drift. In *ICML*, pages 449–456, 2005.
- [40] W. M. Kouw and M. Loog. A review of domain adaptation without target labels. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 43(3):766–785, 2005.
- [41] I. Koychev. Gradual forgetting for adaptation to concept drift. In *ECAI-2000 Workshop on Current Issues in Spatio-Temporal Reasoning*, pages 101–106, 2000.
- [42] L. I. Kuncheva and I. Zliobaite. On the window size for classification in changing environments. *Intelligent Data Analysis*, 13(6):861–872, 2009.
- [43] N. Li, I. W. Tsang, and Z.-H. Zhou. Efficient optimization of performance measures by classifier adaptation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(6):1370–1382, 2013.
- [44] Y.-F. Li and Z.-H. Zhou. Towards making unlabeled data never hurt. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 37(1):175–188, 2015.
- [45] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *ICDM*, pages 413–422, 2008.
- [46] S. Liu, R. Garrepalli, D. Hendrycks, A. Fern, D. Mondal, and T. G. Dietterich. PAC guarantees and effective algorithms for detecting novel categories. *Journal of Machine Learning Research*, 23(44):1–47, 2022.
- [47] A. Y. Majid, S. Saaybi, T. van Rietbergen, V. Francois-Lavet, R. V. Prasad, and C. Verhoeven. Deep reinforcement learning versus evolution strategies: A comparative survey. arXiv:2110.01411, 2021.
- [48] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer. Class-incremental learning: Survey and performance evaluation on image classification. arXiv:2010.15277, 2021.
- [49] M. Mohri and A. M. Medina. New analysis and algorithm for learning with drifting distributions. In *ALT*, pages 124–138, 2012.

- [50] M. Mohri and A. Rostamizadeh. Rademacher complexity bounds for non-i.i.d. processes. In *NIPS*, pages 1097–1104, 2008.
- [51] S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52(1-2):91–118, 2003.
- [52] X. Mu, K. M. Ting, and Z.-H. Zhou. Classification under streaming emerging new classes: A solution using completely-random trees. *IEEE Trans. Knowledge and Data Engineering*, 29(8):1605–1618, 2017.
- [53] X. Mu, F. Zhu, J. Du, E.-P. Lim, and Z.-H. Zhou. Streaming classification with emerging new class by class matrix sketching. In *AAAI*, pages 2373–2379, 2017.
- [54] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari. Learning with noisy labels. In *NIPS*, pages 1196–1204, 2013.
- [55] S. Ozawa, S. L. Toh, S. Abe, S. Pang, and N. Kasabov. Incremental learning of feature space and classifier for face recognition. *Neural Networks*, 18(5-6):575–584, 2005.
- [56] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Trans. Knowledge and Data Engineering*, 22(10):1345–1359, 2009.
- [57] J. Parmar, S. S. Chouhan, V. Raychoudhury, and S. S. Rathore. Open-world machine learning: applications, challenges, and opportunities. arXiv:2105.13448, 2021.
- [58] A. Pentina and C. H. Lampert. Lifelong learning with non-i.i.d. tasks. In *NIPS*, pages 1540–1548, 2015.
- [59] B. Pfülb and A. Gepperth. A comprehensive, application-oriented study of catastrophic forgetting in DNNs. In *ICLR*, 2019.
- [60] W. J. Scheirer, A. R. Rocha, A. Sapkota, and T. E. Boult. Towards open set recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, 2013.
- [61] V. Sehwag, A. N. Bhagoji, L. Song, C. Sitawarin, D. Cullina, and M. Chiang. Analyzing the robustness of open-world machine learning. In *12th ACM Workshop on Artificial Intelligence and Security*, pages 105–116, 2019.

- [62] S. Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.
- [63] R. Socher, M. Ganjoo, C. D. Manning, and A. Y. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, pages 935–943, 2013.
- [64] M. Sugiyama and M. Kawanabe. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. MIT Press, Cambridge, MA, 2012.
- [65] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2012.
- [66] N. Syed, H. Liu, and K. Sung. Incremental learning with support vector machines. In *IJCAI-99 Workshop on Support Vector Machines*, 1999.
- [67] P. E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4(2):161–186, 1989.
- [68] C.-Y. Wei and H. Luo. Non-stationary reinforcement learning without prior knowledge: An optimal black-box approach. In *COLT*, pages 4300–4354, 2021.
- [69] X.-Z. Wu, S. Liu, and Z.-H. Zhou. Heterogeneous model reuse via optimizing multi-party multiclass margin. In *ICML*, pages 6840–6849, 2019.
- [70] X.-Z. Wu and Z.-H. Zhou. A unified view of multi-label performance measures. In *ICML*, pages 3780–3788, 2017.
- [71] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 41(9):2251–2265, 2019.
- [72] C. Zhang, Y. Yu, and Z.-H. Zhou. Learning environmental calibration actions for policy self-evolution. In *IJCAI*, pages 3061–3067, 2018.
- [73] L. Zhang, S. Lu, and Z.-H. Zhou. Adaptive online learning in dynamic environments. In *NeurIPS*, pages 1330–1340, 2018.

- [74] L. Zhang and Z.-H. Zhou. L1-regression with heavy-tailed distributions. In *NeurIPS*, pages 1084–1094, 2018.
- [75] Y. Zhang, R. Jin, and Z.-H. Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1):43–52, 2010.
- [76] Y.-J. Zhang, P. Zhao, L. Ma, and Z.-H. Zhou. An unbiased risk estimator for learning with augmented classes. In *NeurIPS*, pages 10247–10258, 2020.
- [77] Y.-L. Zhang and Z.-H. Zhou. Multi-instance learning with key instance shift. In *IJCAI*, pages 3441–3447, 2017.
- [78] Z.-Y. Zhang, P. Zhao, Y. Jiang, and Z.-H. Zhou. Learning with feature and distribution evolvable streams. In *ICML*, pages 11317–11327, 2020.
- [79] P. Zhao, L.-W. Cai, and Z.-H. Zhou. Handling concept drift via model reuse. *Machine Learning*, 109(3):533–568, 2020.
- [80] P. Zhao, G. Wang, L. Zhang, and Z.-H. Zhou. Bandit convex optimization in non-stationary environments. *Journal of Machine Learning Research*, 22(125):1–45, 2021.
- [81] P. Zhao, X. Wang, S. Xie, L. Guo, and Z.-H. Zhou. Distribution-free one-pass learning. *IEEE Trans. Knowledge and Data Engineering*, 33(3):951–963, 2021.
- [82] P. Zhao, Y.-X. Wang, and Z.-H. Zhou. Non-stationary online learning with memory and non-stochastic control. In *AISTATS*, pages 2101–2133, 2022.
- [83] P. Zhao, L. Zhang, Y. Jiang, and Z.-H. Zhou. A simple approach for non-stationary linear bandits. In *AISTATS*, pages 746–755, 2020.
- [84] P. Zhao, Y.-J. Zhang, L. Zhang, and Z.-H. Zhou. Dynamic regret of convex and smooth functions. In *NeurIPS*, pages 12510–12520, 2020.
- [85] P. Zhao, Y.-J. Zhang, and Z.-H. Zhou. Exploratory machine learning with unknown unknowns. In *AAAI*, pages 10999–11006, 2021.

- [86] G. Zhou, K. Sohn, and H. Lee. Online incremental feature learning with denoising autoencoders. In *AISTATS*, pages 1453–1461, 2012.
- [87] Z.-H. Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, Boca Raton, FL, 2012.
- [88] Z.-H. Zhou. Learnware: On the future of machine learning. *Frontiers of Computer Science*, 10(4):589–590, 2016.
- [89] Z.-H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2018.
- [90] Z.-H. Zhou and Z. Chen. Hybrid decision tree. *Knowledge-Based Systems*, 15(8):515–528, 2002.
- [91] Z.-H. Zhou, Y. Yu, and C. Qian. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer, Berlin, 2019.
- [92] Y. Zhu, K. M. Ting, and Z.-H. Zhou. Discover multiple novel labels in multi-instance multi-label learning. In *AAAI*, pages 2977–2983, 2017.
- [93] Y. Zhu, K. M. Ting, and Z.-H. Zhou. New class adaptation via instance generation in one-pass class incremental learning. In *ICDM*, pages 1207–1212, 2017.
- [94] Y. Zhu, K. M. Ting, and Z.-H. Zhou. Multi-label learning with emerging new labels. *IEEE Trans. Knowledge and Data Engineering*, 30(10):1901–1914, 2018.
- [95] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.