Neurocomputing 459 (2021) 249–289

Contents lists available at ScienceDirect

Neurocomputing

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

Online learning: A comprehensive survey

Steven C.H. Hoi [a,b], Doyen Sahoo [a], Jing Lu [c], Peilin Zhao [d]

[a] *Salesforce Research Asia, Singapore*
[b] *Singapore Management University, Singapore*
[c] *JD.com, China*
[d] *Tencent AI Lab, China*

ARTICLE INFO

Article history:
Received 14 December 2020
Revised 28 March 2021
Accepted 26 April 2021
Available online 30 April 2021
Communicated by Zidong Wang

Keywords:
Online learning
Online convex optimization
Sequential decision making

ABSTRACT

Online learning represents a family of machine learning methods, where a learner attempts to tackle some predictive (or any type of decision-making) task by learning from a sequence of data instances one by one at each time. The goal of online learning is to maximize the accuracy/correctness for the sequence of predictions/decisions made by the online learner given the knowledge of correct answers to previous prediction/learning tasks and possibly additional information. This is in contrast to traditional batch or offline machine learning methods that are often designed to learn a model from the entire training data set at once. Online learning has become a promising technique for learning from continuous streams of data in many real-world applications. This survey aims to provide a comprehensive survey of the online machine learning literature through a systematic review of basic ideas and key principles and a proper categorization of different algorithms and techniques. Generally speaking, according to the types of learning tasks and the forms of feedback information, the existing online learning works can be classified into three major categories: (i) online supervised learning where full feedback information is always available, (ii) online learning with limited feedback, and (iii) online unsupervised learning where no feedback is available. Due to space limitation, the survey will be mainly focused on the first category, but also briefly cover some basics of the other two categories. Finally, we also discuss some open issues and attempt to shed light on potential future research directions in this field.

1. Introduction

Machine learning plays a crucial role in modern data analytics and artificial intelligence (AI) applications. Traditional machine learning paradigms often work in a batch learning or offline learning fashion (especially for supervised learning), where a model is trained by some learning algorithm from an entire training data set at once, and then the model is deployed for inference without (or seldom) performing any update afterwards. Such learning methods suffer from expensive re-training cost when dealing with new training data, and thus are poorly scalable for real-world applications. In the era of big data, traditional batch learning paradigms become more and more restricted, especially when live data grows and evolves rapidly. Making machine learning scalable and practical especially for learning from continuous data streams has become an open grand challenge in machine learning and AI.

Unlike traditional machine learning, *online learning* is a subfield of machine learning and includes an important family of learning techniques which are devised to learn models incrementally from data in a sequential manner. Online learning overcomes the drawbacks of traditional batch learning in that the model can be updated instantly and efficiently by an online learner when new training data arrives. Besides, online learning algorithms are often easy to understand, simple to implement, and often founded on solid theory with rigorous regret bounds. Along with urgent need of making machine learning practical for real big data analytics, online learning has attracted increasing interest in recent years.

This survey aims to give a comprehensive survey of *online learning* literature. Online learning [1] has been extensively studied across different fields, ranging from machine learning, data mining, statistics, optimization and applied math, to artificial intelligence and data science. This survey aims to distill the core ideas of online learning methodologies and applications in literature. This survey is written mainly for machine learning audiences, and assumes readers with basic knowledge in machine learning. While trying our best to make

[1] The term of "online learning" in this survey is *not* related to "e-learning" in the online education field.

the survey as comprehensive as possible, it is very difficult to cover every detail since online learning research has been evolving rapidly in recent years. We apologize in advance for any missing papers or inaccuracies in description, and encourage readers to provide feedback, comments or suggestions. Finally, as a supplemental document to this survey, readers may check our updated version online at: http://libol.stevenhoi.org/survey.

1.1. What is online learning?

Traditional machine learning paradigm often runs in a batch learning fashion, e.g., a supervised learning task, where a collection of training data is given in advance to train a model by following some learning algorithm. Such a paradigm requires the entire training data set to be made available prior to the learning task, and the training process is often done in an offline environment due to the expensive training cost. Traditional batch learning methods suffer from some critical drawbacks: (i) low efficiency in both time and space costs; and (ii) poor scalability for large-scale applications because the model often has to be re-trained from scratch for new training data.

In contrast to batch learning algorithms, online learning is a method of machine learning for data arriving in a sequential order, where a learner aims to learn and update the best predictor for future data at every step. Online learning is able to overcome the drawbacks of batch learning in that the predictive model can be updated instantly for any new data instances. Thus, online learning algorithms are far more efficient and scalable for large-scale machine learning tasks in real-world data analytics applications where data are not only large in size, but also arriving at a high velocity.

1.2. Tasks and applications

Similar to traditional (batch) machine learning methods, online learning techniques can be applied to solve a variety of tasks in a wide range of real-world application domains. Examples of online learning tasks include the following:

Supervised learning tasks: Online learning algorithms can be derived for supervised learning tasks. One of the most common tasks is classification, aiming to predict the categories for a new data instance belongs to, on the basis of observing past training data instances whose category labels are given. For example, a commonly studied task in online learning is online binary classification (e.g., spam email filtering) which only involves two categories (“spam” vs “benign” emails); other types of supervised classification tasks include multi-class classification, multi-label classification, and multiple-instance classification, etc.

In addition to classification tasks, another common supervised learning task is regression analysis, which refers to the learning process for estimating the relationships among variables (typically between a dependent variable and one or more independent variables). Online learning techniques are naturally applied for regression analysis tasks, e.g., time series analysis in financial markets where data instances naturally arrive in a sequential way. Besides, another application for online learning with financial time series data is online portfolio section where an online learner aims to find a good (e.g., profitable and low-risk) strategy for making a sequence of decisions for portfolio selection.

Bandit learning tasks: Bandit online learning algorithms, also known as Multi-armed bandits (MAB), have been extensively used for many online recommender systems, such as online advertising for internet monetization, product recommendation in e-commerce, movie recommendation for entertainment, and other personalized recommendation, etc.

Unsupervised learning tasks: Online learning algorithms can be applied for unsupervised learning tasks. Examples include clustering or cluster analysis—a process of grouping objects such that objects in the same group (“cluster”) are more similar to each other than to objects in other clusters. Online clustering aims to perform incremental cluster analysis on a sequence of instances, which is common for mining data streams.

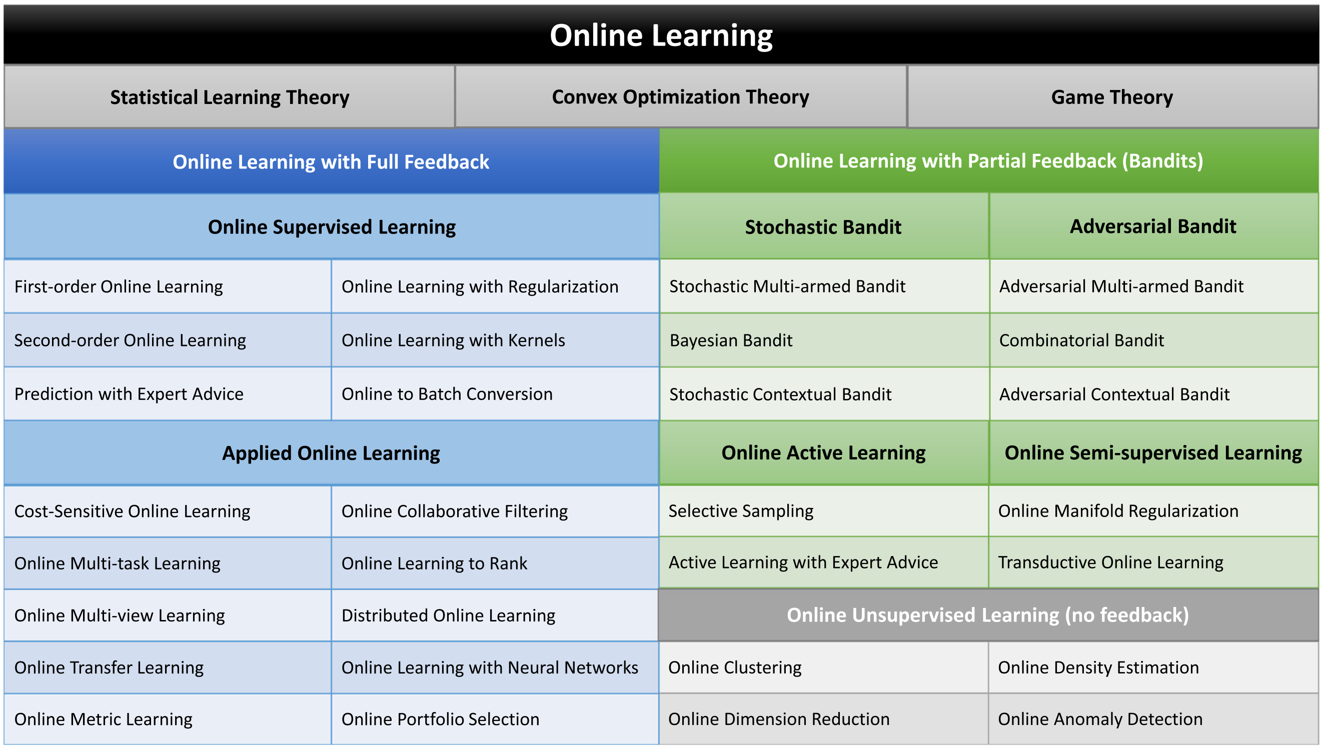
Other learning tasks: Online learning can also be used for other kinds of machine learning tasks, such as learning for recommender systems, learning to rank, or reinforcement learning. For example, collaborative filtering with online learning can be applied to enhance the performance of recommender systems by learning to improve collaborative filtering tasks sequentially from continuous streams of ratings/feedback information from users.

Last but not least, we note that online learning techniques are often used in two major scenarios. One is to improve efficiency and scalability of existing machine learning methodologies for batch machine learning tasks where a full collection of training data must be made available before the learning tasks. For example, Support Vector Machines (SVM) is a well-known supervised learning method for batch classification tasks, in which classical SVM algorithms (e.g., QP or SMO solvers [296]) suffer from poor scalability for very large-scale applications. In literature, various online learning algorithms have been explored for training SVM in an online (or stochastic) learning manner [297,336], making it more efficient and scalable than conventional batch SVMs. The other scenario is to apply online learning algorithms to directly tackle online streaming data analytics tasks where data instances naturally arrive in a sequential manner and the target concepts may be drifting or evolving over time. Examples include time series regression, such as stock price prediction, where data arrives periodically and the learner has to make decisions immediately before getting the next instance.

1.3. Taxonomy

To help readers better understand the online learning literature as a whole, we attempt to construct a taxonomy of online learning methods and techniques, as summarized in Fig. 1. In general, from a theoretical perspective, online learning methodologies are founded based on theory and principles from three major theory communities: learning theory, optimization theory, and game theory. From the perspective of specific algorithms, we can further group the existing online learning techniques into different categories according to their specific learning principles and problem settings. Specifically, according to the types of feedback information and the types of supervision in the learning tasks, online learning techniques can be classified into the following three major categories:

- **Online supervised learning:** This is concerned with supervised learning tasks where full feedback information is always revealed to a learner at the end of each online learning round. It can be further divided into two groups of studies: (i) “Online Supervised Learning” which forms the fundamental approaches and principles for Online Supervised Learning; and (ii) “Applied Online Learning” which constitute more non-traditional online supervised learning, where the fundamental approaches cannot be directly applied, and algorithms have been appropriately tailored to suit the non-traditional online learning setting.
- **Online learning with limited feedback:** This is concerned with tasks where an online learner receives partial feedback information from the environment during the online learning process. For example, consider an online multi-class classification task, at a particular round, the learner makes a prediction of class label for an incoming instance, and then receives the partial



Online Learning			
Statistical Learning Theory	Convex Optimization Theory		Game Theory
Online Learning with Full Feedback		Online Learning with Partial Feedback (Bandits)	
Online Supervised Learning		Stochastic Bandit	Adversarial Bandit
First-order Online Learning	Online Learning with Regularization	Stochastic Multi-armed Bandit	Adversarial Multi-armed Bandit
Second-order Online Learning	Online Learning with Kernels	Bayesian Bandit	Combinatorial Bandit
Prediction with Expert Advice	Online to Batch Conversion	Stochastic Contextual Bandit	Adversarial Contextual Bandit
Applied Online Learning		Online Active Learning	Online Semi-supervised Learning
Cost-Sensitive Online Learning	Online Collaborative Filtering	Selective Sampling	Online Manifold Regularization
Online Multi-task Learning	Online Learning to Rank	Active Learning with Expert Advice	Transductive Online Learning
Online Multi-view Learning	Distributed Online Learning	Online Unsupervised Learning (no feedback)	
Online Transfer Learning	Online Learning with Neural Networks	Online Clustering	Online Density Estimation
Online Metric Learning	Online Portfolio Selection	Online Dimension Reduction	Online Anomaly Detection

Fig. 1. Taxonomy of Online Learning Techniques.

feedback indicating whether the prediction is correct or not, instead of the particular true class label explicitly. For such tasks, the online learner often has to make the online updates or decisions by attempting to achieve some tradeoff between the exploitation of disclosed knowledge and the exploration of unknown information with the environment.

- **Online unsupervised learning:** This is concerned with online learning tasks where the online learner only receives the sequence of data instances without any additional feedback (e.g., true class label) during the online learning tasks. Unsupervised online learning can be considered as a natural extension of traditional unsupervised learning for dealing with data streams, which is typically studied in batch learning fashion. Examples of unsupervised online learning include online clustering, online dimension reduction, and online anomaly detection tasks, etc. Unsupervised online learning has less restricted assumptions about data without requiring explicit feedback or label information which could be difficult or expensive to acquire.

This article will conduct a systematic review of existing works for online learning, especially for online supervised learning and online learning with partial feedback. Finally, we note that it is always very challenging to make a precise categorization of all the existing online learning work, and it is likely that the above proposed taxonomy may not fully cover all the existing online learning work in literature, though we have tried our best to cover as much as possible.

1.4. Related work and further reading

This paper attempts to make a comprehensive survey of online learning research work. In literature, there are some related books, PHD theses, and articles published over the past years dedicated to online learning [73,333], in which many of them also include rich discussions on related work on online learning. For example, the book titled “Prediction, Learning, and Games” [73] gave a nice introduction about some niche subjects of online learning, particularly for online prediction with expert advice and online learning with partial feedback. Another work titled “Online Learning and Online Convex Optimization” [333] gave a nice tutorial about basics of online learning and foundations of online convex optimization. In addition, there are also quite a few PHD theses dedicated to addressing different subjects of online learning [205,332,427,224]. Readers are also encouraged to read some older related books, surveys and tutorial notes about online learning and online algorithms [125,49,304,45,14]. Finally, readers who are interested in applied online learning can explore some open-source toolboxes, including LIBOL [173,397] and Vowpal Wabbit [217].

2. Problem formulations and related theory

Without loss of generality, we first give a formal formulation of a classic online learning problem, i.e., online binary classification, and then introduce basics of statistical learning theory, online convex optimization and game theory as the theoretical foundations for online learning techniques.

2.1. Problem settings

Consider an online binary classification task, online learning takes place in a sequential way. On each round, a learner receives a data instance, and then makes a prediction of the instance, e.g., classifying it into some predefined categories. After making the prediction, the learner receives the true answer about the instance from the environment as a feedback. Based on the feedback, the learner can measure the loss suffered, depending on the difference between the prediction and the answer. Finally, the learner updates its prediction model by some strategy so as to improve predictive performance on future received instances.

Consider spam email detection as a running example of online binary classification, where the learner answers every question in binary: yes or no. The task is supervised binary classification from a machine learning perspective. More formally, we can formulate the problem as follows: consider a sequence of instances/objects represented in a vector space, $\mathbf{x}_t \in \mathbb{R}^d$, where t denotes the t-th round and d is the dimensionality, and we use $y_t \in \{+1, -1\}$ to denote the true class label of the instance. The online binary classification takes place sequentially. On the t-th round, an instance $\mathbf{x}_t$ is received by the learner, which then employs a binary classifier $\mathbf{w}_t$ to make a prediction on $\mathbf{x}_t$, e.g., $\hat{y}_t = \text{sign}(\mathbf{w}_t^\top \mathbf{x}_t)$ that outputs $\hat{y}_t = +1$ if $\mathbf{w}_t^\top \mathbf{x}_t \geqslant 0$ and $\hat{y}_t = -1$ otherwise. After making the prediction, the learner receives the true class label y_t and thus can measure the suffered loss, e.g., using the hinge-loss $\ell_t(\mathbf{w}_t) = \max(0, 1 - y_t \mathbf{w}_t^\top \mathbf{x}_t)$. Whenever the loss is nonzero, the learner updates the prediction model from $\mathbf{w}_t$ to $\mathbf{w}_{t+1}$ by applying some strategy on the training example $(\mathbf{x}_t, y_t)$. The procedure of Online Binary Classification is summarized in Algorithm 1.

Algorithm 1: Online Binary Classification process.

```
Initialize the prediction function as $\mathbf{w}_1$;
for $t = 1, 2, \ldots, T$ do
  Receive instance: $\mathbf{x}_t \in \mathbb{R}^d$;
  Predict $\hat{y}_t = \text{sign}(\mathbf{w}_t^\top \mathbf{x}_t)$ as the label of $\mathbf{x}_t$;
  Receive the true class label: $y_t \in \{-1, +1\}$;
  Suffer loss: $\ell_t(\mathbf{w}_t)$ which is a convex loss function on
  both $\mathbf{w}_t^\top \mathbf{x}_t$ and $y_t$;
  Update the prediction model $\mathbf{w}_t$ to $\mathbf{w}_{t+1}$;
end for
```

By running online learning over a sequence of T rounds, the number of mistakes made by the online learner can be measured as $M_T = \sum_{t=1}^T \mathbb{I}(\hat{y}_t \neq y_t)$. In general, the classic goal of an online learning task is to minimize the regret of the online learner's predictions against the best fixed model in hindsight, defined as

$$R_T = \sum_{t=1}^T \ell_t(\mathbf{w}_t) - \min_{\mathbf{w}} \sum_{t=1}^T \ell_t(\mathbf{w}) \quad (1)$$

where the second term is the loss suffered by the optimal model $\mathbf{w}^*$ that can only be known in hindsight after seeing all the instances and their class labels. From the theoretical perspective of regret minimization, if an online algorithm guarantees that its regret is sublinear as a function of T, i.e., $R_T = o(T)$, it implies that $\lim_{T\to\infty} R(T)/T = 0$ and thus on average the learner performs almost as well as the best fixed model in hindsight.

2.2. Statistical learning theory

Statistical learning theory, first introduced in the late 1960's, is one of key foundations for theoretical analysis of machine learning problems, especially for supervised learning. In literature, there are many comprehensive survey articles and books [363,362]. In the following, we introduce some basic concept and framework.

2.2.1. Empirical error minimization

Assume instance $\mathbf{x}_t$ is generated randomly from a fixed but unknown distribution $P(\mathbf{x})$ and its class label y is also generated with a fixed but unknown distribution $P(y|\mathbf{x})$. The joint distribution of labeled data is $P(\mathbf{x}, y) = P(\mathbf{x})P(y|\mathbf{x})$. The goal of a learning problem is to find a prediction function $f(\mathbf{x})$ that minimizes the expected value of the loss function:

$$R(f) = \int \ell(y, f(\mathbf{x})) dP(x, y)$$

which is also termed as the *True Risk* function. The solution $f^* = \arg\min R(f)$ is the optimal predictor. In general, the true risk function $R(f)$ cannot be computed directly because of the unknown distribution $P(x, y)$. In practice, we approximate the true risk by estimating the risk over a finite collection of instances $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ drawn i.i.d., which is called the "Empirical Risk" or "Empirical Error"

$$R_{emp}(f) = \frac{1}{N} \sum_{n=1}^N \ell(y_n, f(\mathbf{x}_n))$$

The problem of learning via the Empirical Error Minimization (ERM) is to find a hypothesis f over a hypothesis space $\mathcal{F}$ by minimizing the Empirical Error:

$$\hat{f}_n = \arg\min_{f\in\mathcal{F}} R_{emp}(f)$$

ERM is the theoretical base for many machine learning algorithms. For example, in the problem of binary classification, when assuming $\mathcal{F}$ is the set of linear classifiers and the hinge loss is used, the ERM principle indicates that the best linear model $\mathbf{w}$ can be trained by minimizing the following objective

$$R_{emp}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N \max(0, 1 - y_n \mathbf{w}^\top \mathbf{x}_n)$$

2.2.2. Error decomposition

The difference between the optimal predictor f^* and the empirical best predictor $\hat{f}_n$ can be measured by the Excess Risk, which can be decomposed as follows:

$$R(\hat{f}_n) - R(f^*) = \left(R(\hat{f}_n) - \inf_{f\in\mathcal{F}} R(f) \right) + \left(\inf_{f\in\mathcal{F}} R(f) - R(f^*) \right)$$

where the first term is called the Estimation Error due to the finite amount of training samples that may not be enough to represent the unknown distribution, and the second term is called the Approximation Error due to the restriction of model class $\mathcal{F}$ that may not be flexible enough to include the optimal predictor f^*. In general, the estimation error will be reduced when increasing the amount of training data, while the approximation error can be reduced by increasing the model complexity/capacity. However, the estimation error often increases when the model complexity grows, making it challenging for model selection.

2.3. Convex optimization theory

Many online learning problems can essentially be (re-)formulated as an Online Convex Optimization (OCO) task. In the following, we introduce some basics of OCO.

An online convex optimization task typically consists of two major elements: a convex set $\mathcal{S}$ and a convex cost function $\ell_t(\cdot)$. At each time step t, the online algorithm decides to choose a weight vector $\mathbf{w}_t \in \mathcal{S}$; after that, it suffers a loss $\ell_t(\mathbf{w}_t)$, which is computed based on a convex cost function $\ell_t(\cdot)$ defined over $\mathcal{S}$. The goal of the online algorithm is to choose a sequence of decisions $\mathbf{w}_1, \mathbf{w}_2, \ldots$ such that the regret in hindsight can be minimized.

More formally, an online algorithm aims to achieve a low regret R_T after T rounds, where the regret R_T is defined as:

$$R_T = \sum_{t=1}^T \ell_t(\mathbf{w}_t) - \inf_{\mathbf{w}^*\in\mathcal{S}} \sum_{t=1}^T \ell_t(\mathbf{w}^*), \quad (2)$$

where $\mathbf{w}^*$ is the solution that minimizes the convex objective function $\sum_{t=1}^{T} \ell_t(\mathbf{w})$ over $\mathcal{S}$.

For example, consider an online binary classification task for training online Support Vector Machines (SVM) from a sequence of labeled instances $(\mathbf{x}_t, y_t), t = 1, \ldots, T$, where $\mathbf{x}_t \in \mathcal{R}^d$ and $y_t \times \{+1, -1\}$. One can define the loss function $\ell(\cdot)$ as $\ell_t(\mathbf{w}_t) = \max(0, 1 - y_t \mathbf{w}^\top \mathbf{x})$ and the convex set S as $\{\forall \mathbf{w} \in \mathcal{R}^d | \|\mathbf{w}\| \leqslant C\}$ for some constant parameter C. There are a variety of algorithms to solve this problem.

For a comprehensive treatment of this subject, readers are referred to the books in [333,167]. Below we briefly review three major families of online convex optimization (OCO) methods, including first-order algorithms, second-order algorithms, and regularization based approaches.

2.3.1. First-order methods

First order methods aim to optimize the objective function using the first order (sub) gradient information. Online Gradient Descent (OGD) [445] can be viewed as an online version of Stochastic Gradient Descent (SGD) in convex optimization, and is one of the simplest and most popular methods for convex optimization.

At every iteration, based on the loss suffered on instance $\mathbf{x}_t$, the algorithm takes a step from the current model to update to a new model, in the direction of the gradient of the current loss function. This update gives us $\mathbf{u} = \mathbf{w}_t - \eta_t \nabla \ell_t(\mathbf{w}_t)$. The resulting update may push the model to lie outside the feasible domain. Thus, the algorithm projects the model onto the feasible domain, i.e., $\Pi_{\mathcal{S}}(\mathbf{u}) = \arg\min_{\mathbf{w} \in \mathcal{S}} \|\mathbf{w} - \mathbf{u}\|$ (where $\Pi_{\mathcal{S}}$ denotes the projection operation). OGD is simple and easy to implement, but the projection step sometimes may be computationally intensive which depends on specific tasks. In theory [445], OGD achieves sublinear regret $O\left(\sqrt{T}\right)$ for an arbitrary sequence of T convex cost functions (of bounded gradients), with respect to the best single decision in hindsight.

2.3.2. Second-order methods

Second-order methods aim to exploit second order information to speed up the convergence of the optimization. A popular approach is the Online Newton Step Algorithm. The Online Newton Step [163] can be viewed as an online analogue of the Newton-Raphson method in batch optimization. Like OGD, ONS also performs an update by subtracting a vector from the current model in each online iteration. While the vector subtracted by OGD is the gradient of the current loss function based on the current model, in ONS the subtracted vector is the inverse Hessian multiplied by the gradient, i.e., $A_t^{-1} \nabla \ell_t(\mathbf{w}_t)$ where A_t is related to the Hessian. A_t is also updated in each iteration as $A_t = A_{t-1} + \nabla \ell_t(\mathbf{w}_t) \nabla \ell_t(\mathbf{w}_t)^\top$. The updated model is projected back to the feasible domain as $\mathbf{w}_{t+1} = \Pi_{\mathcal{S}}^{A_t}\left(\mathbf{w}_t - \eta A_t^{-1} \nabla \ell_t(\mathbf{w}_t)\right)$, where $\Pi_{\mathcal{S}}^{A}(\mathbf{u}) = \arg\min_{\mathbf{w} \in \mathcal{S}} (\mathbf{w} - \mathbf{u})^\top A (\mathbf{w} - \mathbf{u})$. Different from OGD where the projection is made under the Euclidean norm, ONS projects under the norm induced by the matrix A_t. Although ONS's time complexity $O(n^2)$ is higher than OGD's $O(n)$, it guarantees a logarithmic regret $O(\log T)$ under relatively weaker assumptions of exp-concave cost functions.

2.3.3. Regularization

Unlike traditional convex optimization, the aim of OCO is to optimize the regret. Traditional approaches (termed as Follow the Leader (FTL)) can be unstable, leading to high regret (e.g. linear regret) in the worst case [167]. This motivates the need to stabilize the approaches through regularzation. Here we discuss the common regularization approaches.

Follow-the-Regularized-Leader (FTRL). The idea of Follow-the-Regularized-Leader (FTRL) [3,334] is to stablize the prediction of the Follow-the-Leader (FTL) [194,155] by adding a regularization term $R(\mathbf{w})$ which is strongly convex, smooth and twice differentiable. The idea is to solve the following optimization problem in each iteration:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathcal{S}} \left[\eta \sum_{s=1}^{t} \nabla \ell_s(\mathbf{w}_s)^\top \mathbf{w} + R(\mathbf{w}) \right]$$

where $\mathcal{S}$ is the feasible convex set and η is the learning rate. In theory, the FTRL algorithm in general achieves a sublinear regret bound $O\left(\sqrt{T}\right)$.

Online Mirror Descent (OMD). OMD is an online version of the Mirror Descent (MD) method [357,119] in batch convex optimization. The OMD algorithm behaves like OGD, in that it updates the model using a simple gradient rule. However, it generalizes OGD as it performs updates in the dual space. This duality is induced by the choice of the regularizer: the gradient of the regularization serves as a mapping from $\mathbb{R}^d$ to itself. Due to this transformation by the regularizer, OMD is able to obtain better bounds in terms of the geometry of the space.

In general, OMD has two variants of algorithms: lazy OMD and active OMD. The lazy version keeps track of a point in Euclidean space and projects it onto the convex feasible domain only when making prediction, while the active version keeps a feasible model all the time, which is a direct generalization of OGD. Unlike OGD, the projection step in OMD is based on the Bregman Divergence $\mathcal{B}_R$, i.e., $\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathcal{S}} \mathcal{B}_R(\mathbf{w} \| \mathbf{v}_{t+1})$, where $\mathbf{v}_{t+1}$ is the updated model after the gradient step. In general, the lazy OMD has the same regret bound as FTRL. The active OMD also has a similar regret bound. When $R(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$, OMD recovers OGD. If we use other functions as R, we can also recover some other interesting algorithms, such as the Exponential Gradient (EG) algorithm below.

Exponential Gradient (EG). Let $R(\mathbf{w}) = \mathbf{w} \ln \mathbf{w}$ be the negative entropy function and the feasible convex domain be the simplex $\mathcal{S} = \Delta_d = \left\{\mathbf{w} \in \mathbb{R}_+^d | \sum_i w_i = 1\right\}$, then OMD will recover the Exponential Gradient (EG) algorithm [203]. In this special case, the induced projection is the normalization by the $L1$ norm, which indicates

$$w_{t+1,i} = \frac{w_{t,i} \exp\left[-\eta (\nabla \ell_t(\mathbf{w}_t))_i\right]}{\sum_j w_{t,j} \exp\left[-\eta (\nabla \ell_t(\mathbf{w}_t))_j\right]}$$

As a special case of OMD, the regret of EG is bounded by $O\left(\sqrt{T}\right)$.

Adaptive (Sub)-Gradient Methods. In the previous algorithms, the regularization function R is always fixed and data independent, during the whole learning process. Adaptive (Sub)-Gradient (AdaGrad) algorithm [118] is an algorithm that can be considered as online mirror descent with adaptive regularization, i.e., the regularization function R can change over time. The regularizer R at the t-th step, is actually the function $R(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_{A_t^{1/2}}^2 = \frac{1}{2}\mathbf{w}^\top A_t^{1/2} \mathbf{w}$, which is constructed from the (sub)-gradients received before (and including) the t-th step. In each iteration the model is updated as:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathcal{S}} \left\| \mathbf{w} - \left[\mathbf{w}_t - \eta A_t^{-\frac{1}{2}} \nabla \ell_t(\mathbf{w}_t) \right] \right\|_{A_t^{\frac{1}{2}}}^2$$

where A_t is updated as:

$$A_t = A_{t-1} + \nabla \ell_t(\mathbf{w}_t) \nabla \ell_t(\mathbf{w}_t)^\top$$

We also note that there are also other emerging online convex optimization methods, such as Online Convex Optimization with long term constraints [183], which assumes that the constraints are only required to be satisfied in long term, and Online ADMM [371] which is an online version for the Alternating Direction Method of Multipliers (ADMM) [132,53] and is particularly suitable for distributed optimization applications. The RESCALEDEXP algorithm [103], proposed recently, does not use any prior knowledge about the loss functions and does not require the tuning of learning rate.

2.4. Game theory

Game theory is closely related to online learning. In general, an online prediction task can be formulated as a problem of learning to play a repeated game between a learner and an environment [129]. Consider online classification as an example, during each iteration, the algorithm chooses one class from a finite number of classes and the environment reveals the true class label. Assume the environment is stable (e.g., i.i.d), i.e., not played by an adversary. The algorithm aims to perform as well as the best fixed strategy. The classic online classification problem thus can be modeled by the game theory under the simplest assumption, full feedback and a stable environment. More generally, various settings in game theory can be related to many other types of online learning problems. For example, the feedback may be partly observed, or the environment is not i.i.d. or can be operated by an adversary who aims to maximize the loss of the predictor. In this section, we will introduce some basic concepts about game theory and some fundamental theory of learning in games. We will focus on regret-based minimization procedures and limit our attention to finite strategic or normal form games. A more comprehensive study on this subject can be found in [73,284].

2.4.1. Game playing and nash equilibrium

K-Player Normal-Form Games. Consider a game with K players $(1 < K < \infty)$, where each player $k \in \{1, \ldots, K\}$ can take N_k possible actions. The players' actions can be represented by a vector $\mathbf{i} = (\mathbf{i_1}, \ldots, \mathbf{i_K})$, where $i_k \in \{1, \ldots, N_k\}$ denotes the action of player k. The loss suffered by the player k is denoted by $\ell^{(k)}(\mathbf{i})$ since the loss is related to not only the action of player k but the action of all the other players. During each iteration of the game, each player tries to take actions in order to minimize its own loss.

Using a mixed strategy, player k takes actions based on a probability distribution $\mathbf{p^{(k)}} = \left(\mathbf{p_1^{(k)}}, \ldots, \mathbf{p_{N_k}^{(k)}}\right)$ over the set of $\{1, \ldots, N_k\}$ actions. In particular, the actions played by all the K players can be denoted as a random vector $\mathbf{I} = (\mathbf{I_1}, \ldots, \mathbf{I_K})$, where I_k is the action played by player k which is a random variable taking value over the set of $\{1, \ldots, N_k\}$ actions distributed according to $\mathbf{p^{(k)}}$. The expected loss of player k can be computed as

$$\mathbb{E}\ell^{(k)}(\mathbf{I}) = \sum_{\mathbf{i_1=1}}^{\mathbf{N_1}} \cdots \sum_{\mathbf{i_K=1}}^{\mathbf{N_K}} \mathbf{p_{i_1}^{(1)}} \times \cdots \times \mathbf{p_{i_K}^{(K)}} \ell^{(\mathbf{k})}(\mathbf{i_1}, \ldots, \mathbf{i_K})$$

Nash Equilibrium. This is an important notion in game theory. In particular, a collective strategy of all players $\mathbf{p^{(1)}} \times \cdots \times \mathbf{p^{(K)}}$ is called a *Nash equilibrium* if any mixed strategy among the K players $\mathbf{p^{(k)}}$ is replaced by any new mixed strategy $\mathbf{q^{(k)}}$ while all other $K-1$ players' mixed strategies make no change, we have

$$\mathbb{E}\ell^{(k)}(\mathbf{I}) \leqslant \mathbb{E}\ell^{(\mathbf{k})}(\mathbf{I}')$$

where $\mathbf{I}'$ denotes the actions played by the K players using the new strategies. This definition means that in a Nash Equilibrium, no player can achieve a lower loss by only changing its own strategy if other players do not change. In a Nash Equilibrium, each player gets its own optimal strategy and has no incentive of changing its strategy. One can prove that every finite game has at least one Nash equilibrium, but a game may have multiple Nash equilibria depending on the structure of the game and the loss functions.

2.4.2. Repeated two-player zero-sum games

A simple but important special class of K-Player Normal Form Games is the class of two-player zero-sum games where only one player plays against one opponent, i.e., $K = 2$. *Zero-sum* means that for any action, the sum of losses of all players is zero. This indicates that the game is purely competitive and a player's loss results in another player's gain. In such games, the first player is often called the row player, and the second player is called the column player whose goal is to maximize the loss of the first player. To simplify notation, we consider the row player has N possible actions and the column player has M possible actions. We denote by $L \in [0,1]^{N \times M}$ where $L(i,j)$ is the loss of the row player taking action i while the column player chooses action j, and the mixed strategies for the row and column players denoted by $\mathbf{p} = (\mathbf{p}_, \ldots, \mathbf{p_N})$ and $\mathbf{q} = (\mathbf{q}_, \ldots, \mathbf{q_N})$, respectively. For the two mixed strategies $\mathbf{p}$ and $\mathbf{q}$, the expected loss for the row player (which is equivalent to the expected gain of the column player) can be computed by

$$L(\mathbf{p}, \mathbf{q}) = \sum_{\mathbf{i=1}}^{\mathbf{N}} \sum_{\mathbf{j=1}}^{\mathbf{M}} \mathbf{p}(\mathbf{i})\mathbf{q}(\mathbf{j})\mathbf{L}(\mathbf{i}, \mathbf{j})$$

A pair of mixed strategies $(\mathbf{p}, \mathbf{q})$ is a Nash equilibrium if and only if

$$L(\mathbf{p}, \mathbf{q}') \leqslant \mathbf{L}(\mathbf{p}, \mathbf{q}) \leqslant \mathbf{L}(\mathbf{p}', \mathbf{q}), \quad \forall \mathbf{p}', \forall \mathbf{q}'$$

One natural solution to the two-player zero-sum games is to follow the minimax solution. In particular, for the row player using some strategy $\mathbf{p}$, the worst-case loss is at most $\max_{\mathbf{q}} L(\mathbf{p}, \mathbf{q})$ if the column player makes the decision after seeing $\mathbf{p}$. Therefore, the worst-case optimal strategy (also called the minimax optimal strategy) for the row player is $\mathbf{p}^* = \arg\min_{\mathbf{p}} \max_{\mathbf{q}} \mathbf{L}(\mathbf{p}, \mathbf{q})$. Similarly, the maximin optimal strategy for the column player is $\mathbf{q}^* = \arg\max_{\mathbf{q}} \min_{\mathbf{p}} \mathbf{L}(\mathbf{p}, \mathbf{q})$. The pair of $(\mathbf{p}^*, \mathbf{q}^*)$ is called a minimax solution of the game. Surprisingly there is no difference between $\min_{\mathbf{p}} \max_{\mathbf{q}} L(\mathbf{p}, \mathbf{q})$ and $\max_{\mathbf{q}} \min_{\mathbf{p}} L(\mathbf{p}, \mathbf{q})$, which is known as the von Neumann's minimax theorem, a fundamental result of game theory.

Theorem 1. (von Neumann's minimax theorem) *In a two-player zero-sum game, when two players follow the strategies of the minimax solution, they reach the same optimal value*

$$V^* = \min_{\mathbf{p}} \max_{\mathbf{q}} L(\mathbf{p}, \mathbf{q}) = \max_{\mathbf{q}} \min_{\mathbf{p}} \mathbf{L}(\mathbf{p}, \mathbf{q})$$

V^* is called the value of the game which is unique for a two-player zero-sum game. A pair of mixed strategies $(\mathbf{p}, \mathbf{q})$ is a Nash equilibrium if and only if it achieves the value of game.

We can now relate game theory to online learning as the problem of learning to play repeated two-player zero-sum games. In the context of online learning, the row player is also called the *learner* and the column player is called the *environment*. The repeated game playing between the row player and the column player is treated as a sequence of T rounds of interactions between the learner and the environment. On each round $t = 1, \ldots, T$,

- the leaner chooses a mixed strategy $\mathbf{p_t}$;
- the environment chooses a mixed strategy $\mathbf{q_t}$ (may be chosen by the knowledge $\mathbf{p_t}$);

- the learner observes the losses $L(i, \mathbf{q_t}) \quad \forall i \in [N]$

In general, the goal of the learner is to minimize the cumulative loss, i.e., $\sum_{t=1}^{T} L(\mathbf{p_t}, \mathbf{q_t})$.

3. Online supervised learning

3.1. Overview

In this section, we survey a family of "online supervised learning" algorithms which define the fundamental approaches and principles for online learning methodologies toward supervised learning tasks [333,305].

We first discuss linear online learning methods, where a target model is a linear function. More formally, consider an input domain $\mathcal{X}$ and an output domain $\mathcal{Y}$ for a learning task, we aim to learn a hypothesis $f : \mathcal{X} \mapsto \mathcal{Y}$, where the target model f is linear. For example, consider a typical linear binary classification task, our goal is to learn a linear classifier $f : \mathcal{X} \mapsto \{+1, -1\}$ as follows: $f(\mathbf{x}_t; \mathbf{w}) = sgn(\mathbf{w} \cdot \mathbf{x}_t)$, where $\mathcal{X}$ is typically a d-dimensional vector space $\mathbb{R}^d$, $\mathbf{w} \in \mathcal{X}$ is a weight vector specified for the classifier to be learned, and $sgn(z)$ is an indicator function that outputs +1 when $z > 0$ and -1 otherwise. We review two major types of linear online learning algorithms: first-order online learning and second-order online learning algorithms. Following this, we discuss Prediction with expert advice, and Online Learning with Regularization. This is followed by reviewing nonlinear online learning using kernel based methods. We discuss a variety of kernel-based online learning approaches, their computational challenges, and several approximation strategies for efficient learning. We end this section by discussing the theory for converting using online learning algorithms to learn a batch model that can generalize well.

3.2. First-order online learning

In the following, we survey a family of first-order linear online learning algorithms, which exploit the first order information of the model during learning process.

3.2.1. Perceptron

Perceptron [311,10,285] is the oldest algorithm for online learning. Algorithm 2 gives the Perceptron algorithm for online binary classification.

Algorithm 2: Perceptron

```
INIT: $\mathbf{w}_1 = 0$
for $t = 1, 2, \ldots, T$ do
    Given an incoming instance $\mathbf{x}_t$, predict
    $\hat{y}_t = f_t(\mathbf{x}_t) = sign(\mathbf{w}_t \cdot \mathbf{x}_t)$;
    Receive the true class label $y_t \in \{+1, -1\}$;
    if $\hat{y}_t \neq y_t$ then
        $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t \mathbf{x}_t$;
    end if
end for
```

In theory, by assuming the data is separable with some margin γ, the Perceptron algorithm makes at most $\left(\frac{R}{\gamma}\right)^2$ mistakes, where the margin γ is defined as $\gamma = \min_{t \in [T]} |\mathbf{x}_t \cdot \mathbf{w}^*|$ and R is a constant such that $\forall t \in [T], \|\mathbf{x}_t\| \leqslant R$. The larger the margin γ is, the tighter the mistake bound will be.

In literature, many variants of Perceptron algorithms have been proposed. One simple modification is the "normalized Perceptron" algorithm that differs only in the updating rule as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \frac{\mathbf{x}_t}{\|\mathbf{x}_t\|}$$

The mistake bound of the "normalized Perceptron" algorithm can be improved from $\left(\frac{R}{\gamma}\right)^2$ to $\left(\frac{1}{\gamma}\right)^2$ for the separable case due to the normalization effect.

3.2.2. Winnow

Unlike the Perceptron algorithm that uses additive updates, Winnow [252] employs multiplicative updates. The problem setting is slightly different from the Perceptron: $\mathcal{X} = \{0, 1\}^d$ and $y \in \{0, 1\}$. The goal is to learn a classifier $f(x_1, \ldots, x_n) = x_{i_1} \vee \ldots \vee x_{i_k}$ called monotone disjunction, where $i_k \in 1, \ldots, d$. The separating hyperplane for this classifier is given by $x_{i_1} + \ldots + x_{i_k}$. The Winnow algorithm is outlined in Algorithm 3.

Algorithm 3: Winnow

```
INIT: $\mathbf{w}_1 = \mathbf{1}^d$, constant $\alpha > 1$ (e.g.,$\alpha = 2$)
for $t = 1, 2, \ldots, T$ do
    Given an instance $\mathbf{x}_t$, predict $\hat{y}_t = \mathbb{I}_{\mathbf{w}_t \cdot \mathbf{x}_t \geqslant \theta}$ (outputs 1 if
    statement holds and 0 otherwise);
    Receive the true class label $y_t \in \{1, 0\}$;
    if $\hat{y}_t = 1, y_t = 0$
        set $w_i = 0$ for all $x_{t,i} = 1$ ("elimination" or "demotion"),
    end if
    if $\hat{y}_t = 0, y_t = 1$ then
        set $w_i = \alpha w_i$ for all $x_{t,i} = 1$ ("promotion").
    end if
end for
```

The Winnow algorithm has a mistake bound $\alpha k(\log_\alpha \theta + 1) + n/\theta$ where $\alpha > 1$ and $\theta \geqslant 1/\alpha$ and the target function is a k-literal monotone disjunction.

3.2.3. Passive-aggressive online learning (PA)

This is a popular family of first-order online learning algorithms which generally follows the principle of margin-based learning [95]. Specifically, given an instance $\mathbf{x}_t$ at round t, PA formulates the updating optimization as follows:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 \quad s.t. \ \ell_t(\mathbf{w}) = 0 \tag{3}$$

where $\ell_t(\mathbf{w}) = \max(0, 1 - y_t \mathbf{w} \cdot \mathbf{x}_t)$ is the hinge loss. The above resulting update is passive whenever the hinge loss is zero, i.e., $\mathbf{w}_{t+1} = \mathbf{w}_t$ whenever $\ell = 0$. In contrast, whenever the loss is nonzero, the approach will force $\mathbf{w}_{t+1}$ aggressively to satisfy the constraint regardless of any step-size; the algorithm is thus named as "Passive-Aggressive" (PA) [95]. Specifically, PA aims to keep the updated classifier $\mathbf{w}_{t+1}$ stay close to the previous classifier ("passiveness") and ensure every incoming instance to be classified correctly by the updated classifier ("aggressiveness"). The regular PA algorithm assumes training data is always separable, which may not be true for noisy training data in real-world applications. To overcome such limitation, two variants of PA relax the assumption as:

$$\begin{aligned} &\mathrm{PA-I}: \mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi \\ &\text{subject to} \quad \ell_t(\mathbf{w}) \leqslant \xi \text{ and } \xi \geqslant 0 \\ &\mathrm{PA-II}: \mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2 \\ &\text{subject to} \quad \ell_t(\mathbf{w}) \leqslant \xi \end{aligned} \tag{4}$$

where C is a positive parameter to balance the tradeoff between "passiveness" (first regularization term) and "aggressiveness" (second slack-variable term). By solving the three optimization tasks, we can derive the closed-form updating rules of three PA algorithms:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t, \quad \tau_t = \begin{cases} \ell_t/||\mathbf{x}_t||^2 & \text{(PA)} \\ \min\left\{C, \ell_t/||\mathbf{x}_t||^2\right\} & \text{(PA – I)} \\ \frac{\ell_t}{||\mathbf{x}_t||^2 + \frac{1}{2C}} & \text{(PA – II)} \end{cases}$$

It is important to note a major difference between PA and Perceptron algorithms. Perceptron makes an update only when there is a classification mistake. However, PA algorithms aggressively make an update whenever the loss is nonzero (even if the classification is correct). In theory [95], PA algorithms have comparable mistake bounds as the Perceptron algorithms, but empirically PA algorithms often outperform Perceptron significantly. The PA algorithms are outlined in Algorithm 4.

Algorithm 4: Passive Aggressive Algorithms

```
INIT: $\mathbf{w}_1$, Aggressiveness Parameter $C$;
for $t = 1, 2, \ldots, T$ do
    Receive $\mathbf{x}_t \in \mathbb{R}^d$, predict $\hat{y}_t$ using $\mathbf{w}_t$;
    Suffer loss $\ell_t(\mathbf{w}_t)$;
    Set $\tau = \begin{cases} \ell_t/||\mathbf{x}_t||^2 & \text{(PA)} \\ \min\left\{C, \ell_t/||\mathbf{x}_t||^2\right\} & \text{(PA-I)} \\ \frac{\ell_t}{||\mathbf{x}_t||^2 + \frac{1}{2C}} & \text{(PA-II)} \end{cases}$
    Update $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$;
end for
```

[182] extend the PA algorithm with the principle of total error rate minimization for binary classification.

3.2.4. Online Gradient Descent (OGD)

Many online learning problems can be formulated as an online convex optimization task, which can be solved by applying the OGD algorithm. Consider the online binary classification as an example, where we use the hinge loss function, i.e., $\ell_t(\mathbf{w}) = \max(0, 1 - y_t \mathbf{w} \cdot \mathbf{x}_t)$. By applying the OGD algorithm, we can derive the updating rule as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta_t y_t \mathbf{x}_t \tag{5}$$

where η_t is the learning rate (or step size) parameter. The OGD algorithm is outlined in Algorithm 5, where any generic convex loss function can be used. $\Pi_{\mathcal{S}}$ is the projection function to constrain the updated model to lie in the feasible domain.

Algorithm 5: Online Gradient Descent

```
INIT: $\mathbf{w}_1$, convex set $\mathcal{S}$, step size $\eta_t$;
for $t = 1, 2, \ldots, T$ do
    Receive $\mathbf{x}_t \in \mathbb{R}^d$, predict $\hat{y}_t$ using $\mathbf{w}_t$;
    Suffer loss $\ell_t(\mathbf{w}_t)$;
    Update $\mathbf{w}_{t+1} = \Pi_{\mathcal{S}}(\mathbf{w}_t - \eta_t \nabla \ell_t(\mathbf{w}_t))$
end for
```

OGD and PA share similar updating rules but differ in that OGD often employs some predefined learning rate scheme while PA chooses the optimal learning rate τ_t at each round (but subject to a predefined cost parameter C). In literature, different OGD variants have been proposed to improve either theoretical bounds or practical issues, such as adaptive OGD [165], and mini-batch OGD [107], amongst others.

3.2.5. Other first-order algorithms

In literature, there are also some other first-order online learning algorithms, such as Approximate Large Margin Algorithms (ALMA) [141] which is a large margin variant of the p-norm Perceptron algorithm, and the Relaxed Online Maximum Margin Algorithm (ROMMA) [244]. Many of these algorithms often follow the principle of large margin learning. The metaGrad algorithm [121] tries to adapt the learning rate automatically for faster convergence.

3.3. Second-order online learning

Unlike the first-order online learning algorithms that only exploit the first order derivative information of the gradient for the online optimization tasks, second-order online learning algorithms exploit both first-order and second-order information in order to accelerate the optimization convergence. Despite the better learning performance, second-order online learning algorithms often fall short in higher computational complexity. In the following we present a family of popular second-order online learning algorithms.

3.3.1. Second order perceptron (SOP)

SOP algorithm [69] is able to exploit certain geometrical properties of the data which are missed by the first-order algorithms.

For better understanding, we first introduce the whitened Perceptron algorithm, which strictly speaking, is not an online learning method. Assuming that the instances $\mathbf{x}_1, \ldots, \mathbf{x}_T$ are preliminarily available, we can get the correlation matrix $M = \sum_{t=1}^{T} \mathbf{x}_t \mathbf{x}_t^\top$. The whitened Perceptron algorithm is simply the standard Perceptron run on the transformed sequence $\left(M^{-1/2}\mathbf{x}_1, y_1\right), \ldots, \left(M^{-1/2}\mathbf{x}_T, y_T\right)$. By reducing the correlation matrix of the transformed instances, the whitened Perceptron algorithm can achieve significantly better mistake bound.

SOP can be viewed as an online variant of the whitened Perceptron algorithm. In online setting the correlation matrix M can be approximated by the previously seen instances. SOP is outlined in Algorithm 6

Algorithm 6: SOP

```
INIT: $\mathbf{w}_1 = 0$, $X_0$=[], $\mathbf{v}_0 = 0$, $k = 1$
for $t = 1, 2, \ldots, T$ do
    Given an incoming instance $\mathbf{x}_t$, set $S_t = [X_{k-1}\mathbf{x}_t]$,
    predict $\hat{y}_t = f_t(\mathbf{x}_t) = sign(\mathbf{w}_t \cdot \mathbf{x}_t)$, where
    $\mathbf{w}_t = \left(aI_n + S_t S_t^\top\right)^{-1} \mathbf{v}_{k-1}$
    Receive the true class label $y_t \in \{+1, -1\}$;
    if $\hat{y}_t \neq y_t$ then
        $\mathbf{v}_k = \mathbf{v}_{k-1} + y_t \mathbf{x}_t$, $X_k = S_t$, $k = k + 1$.
    end if
end for
```

Here $a \in \mathbb{R}^+$ is a parameter that guarantees the existence of the matrix inverse.

3.3.2. Confidence weighted learning (CW)

The CW algorithm [114] is motivated by the following observation: the frequency of occurrence of different features may differ a lot in an online learning task. (For example) The parameters of binary features are only updated when the features occur. Thus, the frequent features typically receive more updates and are estimated more accurately compared to rare features. However, no distinction is made between these feature types in most online algorithms. This indicates that the lack of second order information about the frequency or confidence of the features can hurt the learning.

In the CW setting, we model the linear classifier with a Gaussian distribution, i.e., $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the mean vector and $\Sigma \in \mathbb{R}^{d\times d}$ is the covariance matrix. When making a prediction, the prediction confidence $M = \mathbf{w}\cdot\mathbf{x}$ also follows a Gaussian distribution: $M \sim \mathcal{N}(\boldsymbol{\mu}_M, \Sigma_M)$, where $\mu_M = \boldsymbol{\mu}\cdot\mathbf{x}$ and $\Sigma_M = \mathbf{x}^\top\Sigma\mathbf{x}$.

Similar to the PA update strategy, the update rule in round t can be obtained by solving the following convex optimization problem:

$$\begin{aligned}&(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1}) = \arg\min_{\boldsymbol{\mu}\in\mathbb{R}^d} \mathrm{D}_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu},\Sigma)||\mathcal{N}(\boldsymbol{\mu}_t,\Sigma_t))\\ &s.t. \quad \Pr[y_t M_t \geqslant 0] \geqslant \eta\end{aligned} \tag{6}$$

The objective function means that the new distribution should stay close to the previous distribution so that the classifier does not forget the information learnt from previous instances, where the distance between the two distributions is measured by the KL divergence. The constraint means that the new classifier should classify the new instance $\mathbf{x}_t$ correctly with probability higher than a predefined threshold parameter $\eta \in (0,1)$.

Note that this is only the basic form of confidence weighted algorithms and has several drawbacks. 1) Similar to the hard margin PA algorithm, the constraint forces the new instance to be correctly classified, which makes this algorithm very sensitive to noise. 2) The constraint is in a probability form. It is easy to solve a problem with the constraint $g(\boldsymbol{\mu}_M, \Sigma_M) < 0$. However, a problem with a probability form constraint is only solvable when the distribution is known. Thus, this method faces difficulty in generalizing to other online learning tasks where the constraint does not follow a Gaussian distribution.

3.3.3. Adaptive regularization of weight vectors (AROW)

AROW [99] is a variant of CW that is designed for non-separable data. This algorithm adopts the same Gaussian distribution assumption on classifier vector $\mathbf{w}$ while the optimization problem is different. By recasting the CW constraint as regularizers, the optimization problem can be formulated as:

$$\mathcal{C}(\boldsymbol{\mu},\Sigma) = \mathrm{D}_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu},\Sigma)||\mathcal{N}(\boldsymbol{\mu}_t,\Sigma_t)) + \lambda_1\ell(y_t, \boldsymbol{\mu}\cdot\mathbf{x}_t) + \lambda_2\mathbf{x}_t^\top\Sigma\mathbf{x}_t \tag{7}$$

where $\ell(y_t, \boldsymbol{\mu}\cdot\mathbf{x}_t) = (\max(0, 1 - y_t\boldsymbol{\mu}\cdot\mathbf{x}_t))^2$ is the squared-hinge loss. During each iteration, the update rule is obtained by solving the optimization problem:

$$(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1}) = \arg\min_{\boldsymbol{\mu}\in\mathbb{R}^d}(\mathcal{C}(\boldsymbol{\mu},\Sigma))$$

which balances the three desires. First, the parameters should not change radically on each round, since the current parameters contain information about previous examples (first term). Second, the new mean parameters should predict the current example with low loss (second term). Finally, as we see more examples, our confidence in the parameters should generally grow (third term). λ_1 and λ_2 are two positive parameters that control the weight of the three desires.

Besides the robustness to noisy data, another important advantage of AROW is its ability to be easily generalized to other online learning tasks, such as Confidence Weighted Online Collaborative Filtering algorithm [261] and Second-Order Online Feature Selection [398].

3.3.4. Soft confidence weighted learning (SCW)

This is a variant of CW learning in order to deal with non-separable data [376,378]. Different from AROW which directly adds loss and confidence regularization, and thus loses the adaptive margin property, SCW exploits adaptive margin by assigning different margins for different instances via a probability formulation. Consequently, SCW tends to be more efficient and effective.

Specifically, the constraint of CW can be rewritten as $y_t(\boldsymbol{\mu}\cdot\mathbf{x}_t) \geqslant \phi\sqrt{\mathbf{x}_t^\top\Sigma\mathbf{x}_t}$. Thus, the loss function can be defined as:$\ell(\mathcal{N}(\boldsymbol{\mu},\Sigma);(\mathbf{x}_t,y_t)) = \max\left(0, \phi\sqrt{\mathbf{x}_t^\top\Sigma\mathbf{x}_t} - y_t(\boldsymbol{\mu}\cdot\mathbf{x}_t)\right)$. The original CW optimization can be rewritten as:

$$\begin{aligned}&(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1}) = \arg\min_{\boldsymbol{\mu}\in\mathbb{R}^d} \mathrm{D}_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu},\Sigma)||\mathcal{N}(\boldsymbol{\mu}_t,\Sigma_t))\\ &\text{subject to} \quad \ell(\mathcal{N}(\boldsymbol{\mu},\Sigma);(\mathbf{x}_t,y_t)) = 0\end{aligned}$$

Inspired by soft-margin PA variants, SCW generalized CW into two soft-margin formulations:

$$(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1}) = \arg\min_{\boldsymbol{\mu}\in\mathbb{R}^d} \mathrm{D}_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu},\Sigma)||\mathcal{N}(\boldsymbol{\mu}_t,\Sigma_t)) + C\ell(\mathcal{N}(\boldsymbol{\mu},\Sigma);(\mathbf{x}_t,y_t))$$

$$(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1}) = \arg\min_{\boldsymbol{\mu}\in\mathbb{R}^d} \mathrm{D}_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu},\Sigma)||\mathcal{N}(\boldsymbol{\mu}_t,\Sigma_t)) + C\ell^2(\mathcal{N}(\boldsymbol{\mu},\Sigma);(\mathbf{x}_t,y_t))$$

where $C \in \mathbb{R}^+$ is a parameter controls the aggressiveness of this algorithm, similar to the C in PA algorithm. The two algorithms are termed “SCW-I” and “SCW-II”.

3.3.5. Other second-order algorithms

The confidence weighted idea also works for other online learning tasks such as multi-class classification [96], active learning [113] and structured-prediction [273]. There are many other online learning algorithms that adopt second order information: IELLIP [409] assumes the objective classifier $\mathbf{w}$ lies in an ellipsoid and incrementally updates the ellipsoid based on the current received instance. Other approaches include New variant of Adaptive Regularization (NAROW) [289] and the Normal Herding method via Gaussian Herding (NHERD) [100]. Recently, Sketched Online Newton [267] made significant improvements to speed-up second order online learning.

3.4. Prediction with expert advice

This is an important online learning subject [313] with many applications. A general setting is as follows. A learner has N experts to choose from, denoted by integers $1,\ldots,N$. At each time step t, the learner decides on a distribution $\mathbf{p}_t$ over the experts, where $p_{t,i} \geqslant 0$ is the weight of each expert i, and $\sum_{i=1}^N p_{t,i} = 1$. Each expert i then suffers some loss $\ell_{t,i}$ according to the environment. The overall loss suffered by the learner is $\sum_{i=1}^N p_{t,i}\ell_{t,i} = \mathbf{p}_t^\top\ell_t$, i.e., the weighted average loss of the experts with respect to the distribution chosen by the learner.

Typically we assume that the loss suffered by any expert is bounded. Specifically, we assume $\ell_{t,i} \in [0,1]$ without loss of generality. Besides this condition, no assumption is made on the form of the loss, or about how they are generated. Suppose the cumulative losses experienced by each expert and the forecaster are calculated respectively as follows:

$$L_{t,i} = \sum_{s=1}^t \ell_{s,i}, \quad L_t = \sum_{s=1}^t \mathbf{p}_t^\top\ell_t.$$

The loss difference between the forecaster and the expert is known as the “regret”, i.e.,

$$R_{t,i} = L_t - L_{t,i}, \quad i = 1,\ldots,N.$$

The goal of learning the forecaster is to make the regret with respect to each expert as small as possible, which is equivalent to minimizing the overall regret, i.e.,

$$R_T = \max_{1\leqslant i\leqslant N} R_{T,i} = L_T - \min_{1\leqslant i\leqslant N} L_{T,i}$$

In general, online prediction with expert advice aims to find an ideal forecaster to achieve a vanishing per-round regret, a property known as the *Hannan-consistency* [155], i.e.,

$$R_T = o(T) \Longleftrightarrow \lim_{T\to\infty}\frac{1}{T}\left(L_T - \min_{1\leqslant i\leqslant N} L_{T,i}\right)$$

An online learner satisfying the above is called a Hannan-consistent forecaster [73]. Next we review some representative algorithms for prediction with expert advice.

3.4.1. Weighted majority algorithms

The weighted majority algorithm (WM) is a simple but widely studied algorithm that makes a binary prediction based on a series of expert advices [255,254]. The simplest version is shown in Algorithm 7, where $\beta \in (0,1)$ is a user specified discount rate parameter.

```
Algorithm 7: Weighted Majority
INIT: Initialize the weights $p_1, p_2, \ldots p_N$ of all experts to $1/N$.
for $t = 1, 2, \ldots, T$ do
    Get the prediction $x_1, \ldots, x_N$ from $N$ experts.
    Output 1 if $\sum_{i:x_i=1} p_i \geqslant \sum_{i:x_i=0} p_i$; otherwise output 0.
    Receive the true value; if the $i$-th expert made a mistake,
     then $p_i = p_i \ * \beta$
end for
```

3.4.2. Randomized multiplicative weights algorithms

This algorithm works under the same assumption that the expert advices are all binary [23]. While the prediction is random, the algorithm gives the prediction 1 with probability of $\gamma = \frac{\sum_{i:x_i=1} p_i}{\sum_i^N p_i}$ and 0 with probability of $1-\gamma$.

3.4.3. Hedge algorithm

The Hedge algorithm [128] is perhaps the most well-known approach for online prediction with expert advice, which can be viewed as a direct generalization of Littlestone and Warmuth's weighted majority algorithm [255,254]. The working of Hedge algorithm is shown in Algorithm 8.

```
Algorithm 8: Hedge Algorithm
INIT: $\beta \in [0,1]$, initial weight vector $\mathbf{w}_1 \in [0,1]^N$ with
   $\sum_{i=1}^N w_{1,i} = 1$
for $t = 1, 2, \ldots, T$ do
    set distribution $\mathbf{p}_t = \frac{\mathbf{w}_t}{\sum_{i=1}^N w_{t,i}}$;
    Receive loss $\ell_t \in [0,1]^N$ from environment;
    Suffer loss $\mathbf{p}_t^\top \ell_t$;
    Update the new weight vector to $w_{t+1,i} = w_{t,i}\beta^{\ell_{t,i}}$
end for
```

The algorithm maintains a weight vector whose value at time t is denoted $\mathbf{w}_t = (w_{t,1}, \ldots, w_{t,N})$. At all times, all weights are nonnegative. All of the weights of the initial weight vector $\mathbf{w}_1$ must be nonnegative and sum to one, which can be considered as a prior over the set of experts. If it is believed that one expert performs the best, it is better to assign it more weight. If no prior is known, it is better to set all the initial weights equally, i.e., $w_{1,i} = 1/N$ for all i. The algorithm uses the normalized distribution to make prediction, i.e., $\mathbf{p}_t = \mathbf{w}_t/\sum_{i=1}^N w_{t,i}$. After the loss ℓ_t is disclosed, the weight vector $\mathbf{w}_t$ is updated using a multiplicative rule $w_{t+1,i} = w_{t,i}\beta^{\ell_{t,i}}, \quad \beta \in [0,1]$, which implies that the weight of expert i will exponentially decrease with the loss $\ell_{t,i}$. In theory, the Hedge algorithm is proved to be Hannan consistent.

3.4.4. EWAF algorithms

Besides Hedge, there are some other algorithms for online prediction with expert advice under more challenging settings, including exponentially weighted average forecaster (EWAF) and Greedy Forecaster (GF) [73]. We will mainly discuss EWAF, which is shown in Algorithm 9

```
Algorithm 9: EWAF
INIT: a poll of experts $f_i, \quad i = 1, \ldots, N$ and
  $L_{0,i} = 0, \ i = 1, \ldots, N$, and learning rate $\eta$
for $t = 1, 2, \ldots, T$ do
    The environment chooses the next outcome $y_t$ and the
     expert advice $\{f_{t,i}\}$;
    The expert advice is revealed to the forecaster
    The forecaster chooses the prediction $\hat{p}_t = \frac{\sum_{i=1}^N \exp(-\eta L_{t-1,i}) f_{t,i}}{\sum_{i=1}^N \exp(-\eta L_{t-1,i})}$
    The environment reveals the outcome $y_t$;
    The forecaster incurs loss $\ell(\hat{p}_t, y_t)$ and;
    Each expert incurs loss $\ell(f_{t,i}, y_t)$
    The forecaster update the cumulative loss
     $L_{t,i} = L_{t-1,i} + \ell(f_{t,i}, y_t)$
end for
```

The difference between EWAF and Hedge is that the loss in Hedge is the inner product between the distribution and the loss suffered by each expert, while for EWAF, the loss is between the prediction and the true label, which can be much more complex.

3.4.5. Parameter-free online learning

A category of prediction with expert advice deals with learning without user specified learning rate. It is a difficult task to set a learning rate prior to the learning procedure. To address this issue, parameter-free online algorithms were proposed. Among the early efforts, [82] proposed a variant of Hedge Algorithm without the use of a learning rate. The proposed method achieved optimal regret matching the best bounds of all the previous algorithms (with optimally-tuned parameters). [89] improved this bound, but did not have a closed-form solution. There were further extensions which derived data-dependent bounds too [268,208,34].

3.5. Online learning with regularization

Traditional online learning methods learn a classifier $\mathbf{w} \in \mathbb{R}^d$ where the magnitude of each element $|\mathbf{w}^j|$ weights the importance of each feature, which are often non-zero. When dealing with high dimensional data, traditional online learning methods suffer from expensive computational time and space costs. This drawback is often addressed using regularization by performing Sparse online learning, which aims to exploit the sparsity property with real-world high-dimensional data. Specifically, a batch sparse learning problem can be formalized as:

$$P(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell_t(\mathbf{w}) + \phi_s(\mathbf{w})$$

where ϕ_s is a sparsity-inducing regularizer. For example, when choosing $\phi_s = \lambda\|\mathbf{w}\|_0$, it is equivalent to imposing a hard constraint on the number of nonzero elements in $\mathbf{w}$. Instead of choosing ℓ_0-norm which is hard to be optimized, a more commonly used regularizer is ℓ_1-norm, i.e., $\phi_s = \lambda||\mathbf{w}||_1$, which can induce sparsity of the weight vector but does not explicitly constrain the number of nonzero elements. The following reviews some popular sparse online learning methods.

3.5.1. Truncated gradient descent

A straightforward idea to sparse online learning is to modify Online Gradient Descent and round small coefficients of the weight vector to 0 after every K iterations:

$$\mathbf{w}_{t+1} = T_0(\mathbf{w}_t - \eta\nabla\ell_t(\mathbf{w}_t), \theta)$$

where the function $T_0(\mathbf{v}, \theta)$ performs an element-wise rounding on the input vector: if the j-th element v^j is smaller than the threshold θ, set $v^j = 0$. Despite its simplicity, this method struggles to provide satisfactory performance because the aggressive rounding strategy may ignore many useful weights which may be very small due to low frequency of appearance.

Motivated by addressing the above limitation, the Truncated Gradient Descent (TGD) method [218] explores a less aggressive version of the truncation function:

$$\mathbf{w}_{t+1} = T_1(\mathbf{w}_t - \eta\nabla\ell_t(\mathbf{w}_t), \eta g_i, \theta)$$

$$\text{where } T_1(v^j, \alpha, \theta) = \begin{cases} \max(0, v^j - \alpha) & \text{if } v^j \in [0, \theta] \\ \min(0, v^j + \alpha) & \text{if } v^j \in [-\theta, 0] \\ v^j & \text{otherwise} \end{cases}$$

where $g_i > 0$ is a parameter that controls the level of aggressiveness of the truncation. By exploiting sparsity, TGD achieves efficient time and space complexity that is linear with respect to the number of nonzero features and independent of the dimensionality d. In addition, it is proven to enjoy a regret bound of $O\left(\sqrt{T}\right)$ for convex loss functions when setting $\eta = O\left(1/\sqrt{T}\right)$.

3.5.2. Forward looking subgradients (FOBOS)

Consider the objective function in the t-th iteration of a sparse online learning task as $\ell_t(\mathbf{w}) + r(\mathbf{w})$, FOBOS [117] assumes f_t is a convex loss function (differentiable), and r is a sparsity-inducing regularizer (non-differentiable). FOBOS updates the classifier in the following two steps:

- (1) Perform Online Gradient Descent: $\mathbf{w}_{t+\frac{1}{2}} = \mathbf{w}_t - \eta_t\nabla\ell_t(\mathbf{w}_t)$
- (2) Project the solution in (i) such that the projection stays close to the interim vector $\mathbf{w}_{t+\frac{1}{2}}$ and (ii) has a low complexity due to r:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}}\left\{\frac{1}{2}||\mathbf{w} - \mathbf{w}_{\frac{1}{2}}||^2 + \eta_{t+\frac{1}{2}} r(\mathbf{w})\right\}$$

When choosing ℓ_1-norm as the regularizer, the above optimization can be solved with the closed-form solution for each coordinate:

$$w^j_{t+1} = \operatorname{sign}\left(w^j_{t+\frac{1}{2}}\right)\left[|w^j_{t+\frac{1}{2}}| - \eta_{t+\frac{1}{2}}\right]_+$$

The FOBOS algorithm with ℓ_1-norm regularizer can be viewed as a special case of TGD, where the truncation threshold $\theta = \infty$, and the truncation frequency $K = 1$. When $\eta_{t+\frac{1}{2}} = \eta_{t+1}$ and $\eta_t = O(1/\sqrt{t})$, this algorithm also achieves $O\left(\sqrt{T}\right)$ regret bound.

3.5.3. Regularized dual averaging (RDA)

Motivated by the theory of dual-averaging techniques [279], the RDA algorithm [402] updates the classifier by:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}}\left\{\bar{\mathbf{g}}_t^{\top}\mathbf{w} + \Psi(\mathbf{w}) + \frac{\beta_t}{t}h(\mathbf{w})\right\}$$

where $\Psi(\mathbf{w})$ is the original sparsity-inducing regularizer, i.e., $\Psi(\mathbf{w}) = \lambda||\mathbf{w}||_1$; $h(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2$ is an auxiliary strongly convex function and $\bar{\mathbf{g}}_t$ is the averaged gradients of all previous iterations, i.e., $\bar{\mathbf{g}} = \frac{1}{t}\sum_{\tau=1}^{t}\nabla\ell_\tau(\mathbf{w}_\tau)$. Setting the step size $\beta_t = \gamma\sqrt{t}$, one can derive the closed-form solution:

$$w^j_{t+1} = \begin{cases} 0 & \text{if } |\bar{g}_t^j| < \lambda \\ -\frac{\sqrt{t}}{\gamma}\left(\bar{g}_t^j - \lambda\operatorname{sign}(\bar{g}_t^j)\right) & \text{otherwise} \end{cases}$$

To further pinpoint the differences between RDA and FOBOS, we rewrite FOBOS in the same notation as RDA:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}}\left\{\mathbf{g}_t^{\top}\mathbf{w} + \Psi(\mathbf{w}) + \frac{1}{2\alpha_t}||\mathbf{w} - \mathbf{w}_t||_2^2\right\}$$

Specifically, RDA differs from FOBOS in several aspects. First, RDA uses the averaged gradient instead of the current gradient. Second, $h(\mathbf{w})$ is a global proximal function instead of its local Bregman divergence. Third, the coefficient for $h(\mathbf{w})$ is $\beta_t/t = \gamma/\sqrt{t}$ which is $1/\alpha_t = O(\sqrt{t})$ in FOBOS. Fourth, the truncation of RDA is a constant λ, while the truncation in FOBOS $\eta_{t+\frac{1}{2}}$ decrease with a factor $\sqrt{t}$. Clearly, RDA uses a more aggressive truncation threshold, thus usually generates significantly more sparse solutions. RDA also ensures the $O\left(\sqrt{T}\right)$ regret bound.

3.5.4. Adaptive regularization

One major issue with both FOBOS and RDA is that the auxiliary strongly convex function $h(\mathbf{w})$ may not fully exploit the geometry information of underlying data distribution.Instead of choosing $h(\mathbf{w})$ as an ℓ_2-norm $\frac{1}{2}||\mathbf{w}||^2$ in RDA or a Mahalanobis norm $||\cdot||_{H_t}$ in FOBOS, [116] proposed a data-driven adaptive regularization for $h(\mathbf{w})$, i.e.,

$$h_t(\mathbf{w}) = \frac{1}{2}\mathbf{w}^{\top}H_t\mathbf{w}$$

where $H_t = \left(\sum_{\tau=1}^{t}\mathbf{g}_\tau\mathbf{g}_\tau^{\top}\right)^{\frac{1}{2}}$ accumulates the second order info from the previous instances over time. Replacing the previous $h(\mathbf{w})$ in both RDA and FOBOS by the temporal adaptation function $h_t(\mathbf{w})$, [116] derived two generalized algorithms (Ada-RDA and Ada-FOBOS) with the solutions as follows respectively.

Ada-RDA:

$$w^j_{t+1} = \begin{cases} 0 & \text{if } |\bar{g}_t^j| < \lambda \\ -\frac{t}{\beta H_{t,jj}}\left(\bar{g}_t^j - \lambda\operatorname{sign}(\bar{g}_t^j)\right) & \text{otherwise} \end{cases}$$

Ada-FOBOS:

$$w^j_{t+1} = \operatorname{sign}\left(w^i_t - \frac{\alpha_t}{H_{t,jj}}g^j_t\right)\left[|w^i_t - \frac{\alpha_t}{H_{t,jj}}g^j_t| - \frac{\alpha_t\lambda}{H_{t,ii}}\right]_+ \quad (8)$$

In the above, H_t is approximated by a diagonal matrix since computing the root of a matrix is computationally impractical in high-dimensional data.

3.5.5. Online feature selection

Online feature selection [174,379,195,399,283,260] is closely related to sparse online learning in that they both aim to learn an efficient classifier for very high dimensional data. However, the sparse learning algorithms aim to minimize the ℓ-1 regularized loss, while the feature selection algorithms are motivated to explicitly address the feature selection issue and thus impose a hard constraint on the number of non-zero elements in classifier. Because of these similarities, they share some common strategies such as truncation and projection.

3.5.6. Others

Two stochastic methods were proposed in [337] for ℓ_1-regularized loss minimization. The Stochastic Coordinate Descent (SCD) algorithm randomly selects one coordinate from d dimensions and update this single coordinate with the gradient of the total loss of all instances. The Stochastic Mirror Descent Made Sparse (SMIDAS) algorithm combines the idea of truncating the gradient with mirror descent algorithm, i.e., truncation is performed on the vector in dual space. The disadvantage of the two algorithms is that their computational complexity depends on the dimensionality d. Besides, the two algorithms are designed in batch learning setting, i.e., they assume all instances are known prior to the learning task. Besides, there are also some recent sparse online learning algorithms proposed [370,369], which combine the ideas of sparse learning, second order online learning, and cost-sensitive classification together to make the online algorithms scalable for high-dimensional class-imbalanced learning tasks.

3.6. Online learning with kernels

We now survey a family of "Kernel-based Online Learning" algorithms for learning a nonlinear target function, where the nonlinearity is induced by kernels. We take the typical nonlinear binary classification task as an example. Our goal is to learn a nonlinear classifier $f:\mathbb{R}^d\rightarrow\mathbb{R}$ from a sequence of labeled instances $(\mathbf{x}_t,y_t), t=1,\ldots,T$, where $\mathbf{x}_t\in\mathbb{R}^d$ and $y_t\in\{+1,-1\}$. We build the classification rule as: $\hat{y}_t=\text{sign}(f(\mathbf{x}_t))$, where $\hat{y}_t$ is the predicted class label. We measure the classification confidence of certain instance $\mathbf{x}_t$ by $|f(\mathbf{x}_t)|$. Similar to the linear case, for an online classification task, one can define the hinge loss function $\ell(\cdot)$ for the t-th instance using the classifier at the t-th iteration:

$$\ell((\mathbf{x}_t,y_t);f_t)=\max(0,1-y_tf_t(\mathbf{x}_t))$$

Formally speaking, an online nonlinear learner aims to achieve the lowest regret $R(T)$ after time T, where the regret function $R(T)$ is defined as follows:

$$R(T)=\sum_{t=1}^{T}\ell_t(f_t)-\inf_{f}\sum_{t=1}^{T}\ell_t(f), \tag{9}$$

where $\ell_t(\cdot)$ is the loss for the classification of instance $(\mathbf{x}_t,y_t)$, which is short for $\ell((\mathbf{x}_t,y_t);\cdot)$. We denote by f^* the optimal solution of the second term, i.e., $f^*=\arg\min_f\sum_{t=1}^{T}\ell_t(f)$

In the following, we first introduce online kernel methods and then survey a family of scalable online kernel learning algorithms organized into two major categories: (i) budget online kernel learning using *budget maintenance* strategies and (ii) budget online kernel learning using *functional approximation* strategies. Then we briefly introduce some approaches for online learning with multiple kernels. Without loss of generality, we will adopt the above online binary classification setting for the discussions in this section.

3.6.1. Online kernel methods

We refer to the output f of the learning algorithm as a *hypothesis* and denote the set of all possible hypotheses by $\mathcal{H}=\{f|f:\mathbb{R}^d\rightarrow\mathbb{R}\}$. Here $\mathcal{H}$ a Reproducing Kernel Hilbert Space (**RKHS**) endowed with a kernel function $\kappa(\cdot,\cdot):\mathbb{R}^d\times\mathbb{R}^d\rightarrow\mathbb{R}$ [363] implementing the inner product$\langle\cdot,\cdot\rangle$ such that: 1) κ has the reproducing property $\langle f,\kappa(\mathbf{x},\cdot)\rangle=f(\mathbf{x})$ for $\mathbf{x}\in\mathbb{R}^d$; 2) $\mathcal{H}$ is the closure of the span of all $\kappa(\mathbf{x},\cdot)$ with $\mathbf{x}\in\mathbb{R}^d$, that is, $\kappa(\mathbf{x},\cdot)\in\mathcal{H}$ $\forall\mathbf{x}\in\mathcal{X}$. The inner product $\langle\cdot,\cdot\rangle$ induces a norm on $f\in\mathcal{H}$ in the usual way: $\|f\|_{\mathcal{H}}:=\left\langle f,f\right\rangle^{\frac{1}{2}}$. We denote by $\mathcal{H}_\kappa$ an RKHS with explicit dependence on kernel κ. Throughout the analysis, we assume $\kappa(\mathbf{x},\mathbf{x})\leqslant X^2, \forall\mathbf{x}\in\mathbb{R}^d, X\in\mathbb{R}^+$ is a constant.

The task of training the model of SVM $f(\mathbf{x})$ in batch is formulated as the optimization:

$$\min_{f\in\mathcal{H}_\kappa}\frac{\lambda}{2}\|f\|^2_{\mathcal{H}}+\frac{1}{T}\sum_{t=1}^{T}\ell(f(\mathbf{x}_t);y_t) \tag{10}$$

where $\lambda>0$ is a regularization parameter used to control model complexity. According to the Representer Theorem [326], the optimal solution of the above convex optimization problem lies in the span of T kernels, i.e., those centered on the training points. Consequently, the goal of a typical online kernel learning algorithm is to learn the kernel-based predictive model $f(\mathbf{x})$ for classifying a new instance $\mathbf{x}\in\mathbb{R}^d$ as follows: $f(\mathbf{x})=\sum_{t=1}^{T}\alpha_t\kappa(\mathbf{x}_t,\mathbf{x})$,where T is the number of processed instances, α_t denotes the coefficient of the t-th instance, and $\kappa(\cdot,\cdot)$ denotes the kernel function. We define support vector (SV) as the instance whose coefficient α is nonzero. Thus, we rewrite the previous classifier as $f(\mathbf{x})=\sum_{i\in\mathcal{SV}}\alpha_i\kappa(\mathbf{x}_i,\mathbf{x})$, where $\mathcal{SV}$ is the set of SV's and i is its index. We use the notation $|\mathcal{SV}|$ to denote the SV set size. In literature, different online kernel methods have been proposed. We begin by introducing the simplest one, that is, the kernelized Perceptron algorithm.

Kernelized Perceptron. This extends the Perceptron algorithm using the kernel trick. Algorithm 10 outlines the Kernelized Perceptron algorithm [130].

Algorithm 10: Kernelized Perceptron

```
INIT: $f_0=0$
for $t=1,2,\ldots,T$ do
  Given an incoming instance $\mathbf{x}_t$, predict $\hat{y}_t=\text{sign}(f_t(\mathbf{x}_t))$;
  Receive the true class label $y_t\in\{+1,-1\}$;
  if $\hat{y}_t\neq y_t$ then
    $\mathcal{SV}_{t+1}=\mathcal{SV}_t\cup(\mathbf{x}_t,y_t), f_{t+1}=f_t+y_t\kappa(\mathbf{x}_t,\cdot)$;
  end if
end for
```

The algorithm works similar to the standard Perceptron algorithm, except that the inner product, i.e., $f_t(\mathbf{x}_t)=\sum_i\alpha_i\mathbf{x}_i^\top\mathbf{x}_t$, is replaced by a kernel function in the kernel Percetron, i.e., $f_t(\mathbf{x}_t)=\sum_i\alpha_i\kappa(\mathbf{x}_i^\top,\mathbf{x}_t)$.

Kernelized OGD. This extends the OGD algorithm with kernels [202], as shown in Algorithm 11. Here, $\eta_t>0$ is the learning rate parameter, and ℓ_t' is used to denote the derivative of loss function with respect to the classification score $f_t(\mathbf{x}_t)$.

Algorithm 11: Kernelized OGD

```
INIT: $f_0=0$
for $t=1,2,\ldots,T$ do
  Given an incoming instance $\mathbf{x}_t$, predict $\hat{y}_t=\text{sign}(f_t(\mathbf{x}_t))$;
  Receive the true class label $y_t\in\{+1,-1\}$;
  if $\ell_t(f_t)>0$ then
    $\mathcal{SV}_{t+1}=\mathcal{SV}_t\cup(\mathbf{x}_t,y_t)$,
    $f_{t+1}=f_t-\eta_t\nabla\ell_t(f_t(\mathbf{x}_t))=f_t-\eta_t\ell_t'\kappa(\mathbf{x}_t,\cdot)$;
  end if
end for
```

Other Related Work. The kernel trick implies that the inner product between any two instances can be replaced by a kernel function, i.e., $\kappa(\mathbf{x}_i,\mathbf{x}_j)=\Phi(\mathbf{x}_i)^\top\Phi(\mathbf{x}_j), \forall i,j$, where $\Phi(\mathbf{x}_t)\in R^D$ denotes the feature mapping from the original space to a new D-dimensional space which can be infinite. Using the kernel trick, many existing linear online learning algorithms can be easily extended to their kernelized variants, such as the kernelized Perceptron and kernelized OGD as well as kernel PA variants [95]. However, some algorithms that use complex update rules are non-trivial to be converted into kernelized versions, such as Confidence-Weighted algorithms [114]. Moreover, some online kernel learning methods also attempt to make more effective updates at each iteration. For example, Double Updating Online Learning (DUOL) [431,432,429] improves the efficacy of traditional online kernel learning methods by not only updating the weight of the newly added SV, but also the weight for one existing SV. Finally, we note one major challenge of online kernel method is the computational efficiency and scalability due to the curse of kernelization [386]. In the following, we will discuss two types of techniques to scale up kernel-based online learning methods.

3.6.2. Scalable online kernel learning via budget maintenance

Despite enjoying the clear advantage of accuracy performance over linear models, online kernel learning falls short in some critical drawbacks, in which one critical issue is the growing unbounded number of support vectors with increasing computational and space complexity over time. To address this challenge, a family of algorithms, termed "budget online kernel learning", have been proposed to bound the number of SV's with a fixed budget $B = |\mathcal{SV}|$ using diverse budget maintenance strategies whenever the budget overflows. The general framework for budgeting strategies is shown in Algorithm 12. Most existing budget online kernel methods maintain the budget by three strategies: (i) SV Removal, (ii) SV Projection, and (iii) SV Merging. We briefly review each of them below.

Algorithm 12: Budget Online Kernel Learning

```
INIT: $f_0 = 0$
for $t = 1, 2, \dots, T$ do
    Given an incoming instance $\mathbf{x}_t$, predict
    $\hat{y}_t = \text{sign}(f_t(\mathbf{x}_t))$;
    Receive the true class label $y_t \in \{+1, -1\}$;
    if update is needed then
        update the classifier from $f_t$ to $f_{t+\frac{1}{2}}$ and
        $\mathcal{SV}_{t+\frac{1}{2}} = \mathcal{SV}_t \cup (\mathbf{x}_t, y_t)$
    end if
    if $|\mathcal{SV}_{t+\frac{1}{2}}| > B$
        Update Support Vector Set from $\mathcal{SV}_{t+\frac{1}{2}}$ to
        $\mathcal{SV}_{t+1}$ such that $|\mathcal{SV}_{t+1}| = B$
        Update the classifier from $f_{t+\frac{1}{2}}$ to $f_{t+1}$
    end if
end for
```

SV Removal. This strategy maintains the budget by a simple and efficient way: 1) update the classifier by adding a new SV whenever necessary (depending on the prediction mistake/loss); 2) if the SV size exceeds the budget, discard one of existing SV's and update the classifier accordingly.

To achieve this, we need to address the following concerns: (i) how to update the classifier and (ii) how to choose one of existing SV's for removal.

The first step of updating classifiers depends on which online learning method is used. For example, the Perceptron algorithm has been used in RBP [65], Forgetron [109], and Budget Perceptron [98]. The OGD algorithm has been adopted by BOGD [436] and BSGD+ removal [386], while PA has been used by BPA-S [388,394].

The second step of SV removal is to find one of existing SV's, denoted as $(\mathbf{x}_{del}, y_t)$, to be removed by minimizing the impact of the resulting classifier. One simple way is to randomly discard one of existing SV's uniformly with probability $\frac{1}{B}$, as adopted by RBP [65] and BOGD [436]. Instead of choosing randomly, another way as used in "Forgetron" [109] is to discard the oldest SV by assuming an older SV is less representative for the distribution of fresh training data streams. Despite enjoying the merits of simplicity and high efficiency, these methods are often too simple to achieve satisfactory learning results.

To optimize the performance, some approaches have tried to perform exhaustive search in deciding the best SV for removal. For instance, the Budget Perceptron algorithm [98] searches for one SV that is classified with high confidence by the classifier:

$$y_{del}\left(f_{t+\frac{1}{2}}(\mathbf{x}_{del}) - \alpha_{del}\kappa(\mathbf{x}_{del}, \mathbf{x}_{del})\right) > \beta$$

where $\beta > 0$ is a fixed tolerance parameter. BPA-S shares the similar idea of exhaustive search. For every $r \in [B]$, a candidate classifier $f^r = f_{t+\frac{1}{2}} - \alpha_r\kappa(\mathbf{x}_r, \cdot)$ is generated by discarding the r-th SV from $f_{t+\frac{1}{2}}$. By comparing the B candidate classifiers, the algorithm selects the one that minimizes the current objective function of PA:

$$f_{t+1} = argmin_{r\in[B]}\frac{1}{2}||f^r - f_t||^2_{\mathcal{H}} + C\ell_t(f^r)$$

where $C > 0$ is the regularization parameter of PA to balance aggressiveness and passiveness. Comparing the principles of different SV removal strategies, we observe that a simple rule may not always generate satisfactory accuracy, while an exhaustive search often incurs non-trivial computational overhead, which again may limit the application to large-scale problems. When deploying a solution in practice, one would need to balance the trade-off between effectiveness and efficiency.

SV Projection. SV Projection strategy first appeared in [290] where two new algorithms, Projectron and Projectron++, were proposed, which significantly outperformed the previous SV removal based algorithms such RBP and Forgetron. The SV projection method follows the setting of SV removal and identifies a support vector for removal during the update of the model. It then chooses a subset of $\mathcal{SV}$ as the projection base, which will be denoted by $\mathcal{P}$. Following this, a linear combination of kernels in $\mathcal{P}$ is used to approximate the removed SV. The procedure of finding the optimal linear combination can be formulated as a convex optimization of minimizing the projection error:

$$\boldsymbol{\beta} = \underset{\beta\in\mathbb{R}^{|\mathcal{P}|}}{argmin}E_{proj} = \underset{\beta\in\mathbb{R}^{|\mathcal{P}|}}{argmin}||\alpha_{del}\kappa(\mathbf{x}_{del}, \cdot) - \sum_{i\in\mathcal{P}}\beta_i\kappa(\mathbf{x}_i, \cdot)||^2_{\mathcal{H}}$$

Finally, the classifier is updated by combining this result with the original classifier:

$$f_{t+1} = f_{t+\frac{1}{2}} - \alpha_{del}\kappa(\mathbf{x}_{del}, \cdot) + \sum_{i\in\mathcal{P}}\beta_i\kappa(\mathbf{x}_i, \cdot)$$

There are several algorithms adopting the projection strategy, for example Projectron, Projectron++, BPA-P, BPA-NN [388] and BSGD+Project [386]. These methods differ in a few aspects. First, the update rules are based on different online learning algorithms. Generally speaking, PA based and OGD based tend to outperform Perceptron based algorithms because of their effective update. Second, the choice of discarded SV is different. Since projection itself is relative slow, exhaustive search based algorithms (BPA-NN, BPA-P) are extremely time consuming. Thus algorithms with simple selecting rules are prefered (Projectron, Projectron++, BSGD+Project). Third, the choice of projection base set $\mathcal{P}$ is different. In Projectron, Projectron++, BPA-P and BSGD+Project, the discarded SV is projected onto the whole SV set, i.e. $\mathcal{P} = \mathcal{SV}$. While in BPA-NN, $\mathcal{P}$ is only a small subset of $\mathcal{SV}$, made up of the nearest neighbors of the discarded SV $(\mathbf{x}_{del}, y_{del})$. In general, a larger projection base set implies a more complicated optimization problem and thus more time costs. The research direction of SV projection based budget learning is to find a proper way of selecting $\mathcal{P}$ so that the algorithm achieves the minimized projection error with a relative small projection base set.

SV Merging. [386] proposed a SV merging method called "BSGD+Merge" which replaces the sum of two SV's $\alpha_m\kappa(\mathbf{x}_m,\cdot)+\alpha_n\kappa(\mathbf{x}_n,\cdot)$ by a newly created SV $\alpha_z\kappa(\mathbf{z},\cdot)$, where α_m, α_n and α_z are the corresponding coefficients of $\mathbf{x}_m$, $\mathbf{x}_n$ and $\mathbf{z}$. Following the previous discussion, the goal of online budget learning through SV merging strategy is to find the optimal $\alpha_z\in\mathbb{R}$ and $\mathbf{z}\in\mathbb{R}^d$ that minimizes the gap between f_{t+1} and $f_{t+\frac{1}{2}}$.

As it is relatively complicated to optimize the two terms simultaneously, the optimization is divided into two steps. First, assuming the coefficient of $\mathbf{z}$ is $\alpha_m+\alpha_n$, this algorithm tries to create the optimal SV that minimizes the merging error as follows

$$\min_{\mathbf{z}}||(\alpha_m+\alpha_n)\kappa(\mathbf{z},\cdot)-(\alpha_m\kappa(\mathbf{x}_m,\cdot)+\alpha_n\kappa(\mathbf{x}_n,\cdot))||$$

The solution is $\mathbf{z}=h\mathbf{x}_m+(1-h)x_n$, where $0<h<1$ is a real number that can be found by a line search method. This solution indicates that the optimal created SV lies on the line connecting $\mathbf{x}_m$ and $\mathbf{x}_n$. After obtaining the optimal created SV $\mathbf{z}$, the next step is to find the optimal coefficient α_z, which can be formulated as

$$\min_{\alpha_z}||(\alpha_z\kappa(\mathbf{z},\cdot)-(\alpha_m\kappa(\mathbf{x}_m,\cdot)+\alpha_n\kappa(\mathbf{x}_n,\cdot))||.$$

The solution becomes $\alpha_z=\alpha_m\kappa(\mathbf{x}_m,\mathbf{z})+\alpha_n\kappa(\mathbf{x}_n,\mathbf{z})$. The remaining problem is which two SV's $\mathbf{x}_m$ and $\mathbf{x}_n$ should be merged. The ideal solution is to find the optimal pair with the minimal merging error through exhaustive search, which however requires $O\left(B^2\right)$ time complexity. How to find the optimal pair efficiently remains an open challenge.

Summary. Among various algorithms of budget online kernel learning using budget maintenance, the key differences are their updating rules and budget maintenance strategies. Table 1 gives a summary of different algorithms and their properties.

In addition to the previous budget kernel learning algorithms, there are also some other works in online kernel learning. For example, some studies [266,421] introduce the idea of sparse kernel learning to reduce the number of SV's in the online-to-batch-conversion problem, where an online algorithm can be used to train a kernel model efficiently for the batch setting (See Section 3.7).

3.6.3. Scalable online kernel learning via functional approximation

In contrast to the previous budget online kernel learning methods using budget maintenance strategies to guarantee efficiency and scalability, another emerging and promising strategy is to explore functional approximation techniques for achieving scalable online kernel learning [374,262,184].

The key idea is to construct a kernel-induced feature representation $\mathbf{z}(\mathbf{x})\in\mathbb{R}^D$ such that the inner product of instances in the new feature space can effectively approximate the kernel function:

$$\kappa(\mathbf{x}_i,\mathbf{x}_j)\approx\mathbf{z}(\mathbf{x}_i)^\top\mathbf{z}(\mathbf{x}_j)$$

Using the above approximation, the predictive model with kernels can be rewritten as follows:

$$f(\mathbf{x})=\sum_{i=1}^{B}\alpha_i\kappa(\mathbf{x}_i,\mathbf{x})\approx\sum_{i=1}^{B}\alpha_i\mathbf{z}(\mathbf{x}_i)^\top\mathbf{z}(\mathbf{x})=\mathbf{w}^\top\mathbf{z}(\mathbf{x})$$

where $\mathbf{w}=\sum_{i=1}^{B}\alpha_i\mathbf{z}(\mathbf{x}_i)$ denotes the weight vector to be learned in the new feature space.

As a consequence, solving a regular online kernel classification task can be turned into a linear online classification task on the new feature space derived from the kernel approximation. For example, the methods of online kernel learning with kernel approximation in [374,262] integrate some existing online learning algorithms (e.g., OGD) with kernel approximation techniques [249,342,78] to derive scalable online kernel learning algorithms, including Fourier Online Gradient Descent (FOGD) that explores random Fourier features for kernel functional approximation [303], and Nyström Online Gradient Descent (NOGD) that explores Nyström low-rank matrix approximation methods for approximating large-scale kernel matrix [392]. A recent work, Dual Space Gradient Descent [221,280] updates the model as the RBP algorithm, but also builds an FOGD model using the discarded SV's. The final prediction is the combination of the two models.

3.6.4. Online multiple kernel learning

Traditional online kernel methods usually assume a predefined good kernel is given prior to the online learning task. Such approaches could be restricted since it is often hard to choose a good kernel prior to the learning task. To overcome the drawback, Online Multiple Kernel Learning (OMKL) aims to combine multiple kernels automatically for online learning tasks without fixing any predefined kernel. In the following, we begin by introducing some basics of batch Multiple Kernel Learning (MKL) [343].

Table 1
Comparisons of different budget online kernel learning algorithms.

Algorithms	Update Strategy	Budget Strategy	Update Time	Space
Stoptron [290]	Perceptron	Stop	$O(1)$	$O(B)$
Tighter Perceptron [391]	Perceptron	Removal	$O\left(B^2\right)$	$O(B)$
Tightest Perceptron [387]	Perceptron	Removal	$O\left(B^2\right)$	$O(B)$
Budget Perceptron [98]	Perceptron	Removal	$O\left(B^2\right)$	$O(B)$
RBP [65]	Perceptron	Removal	$O(B)$	$O(B)$
Forgetron[109]	Perceptron	Removal	$O(B)$	$O(B)$
BOGD[436]	OGD	Removal	$O(B)$	$O(B)$
BPA-S [388]	PA	Removal	$O(B)$	$O(B)$
BSGD+removal [386]	OGD	Removal	$O(B)$	$O(B)$
Projectron [290]	Perceptron	Projection	$O\left(B^2\right)$	$O\left(B^2\right)$
Projectron++ [290]	Perceptron	Projection	$O\left(B^2\right)$	$O\left(B^2\right)$
BPA-P [388]	PA	Projection	$O\left(B^3\right)$	$O\left(B^2\right)$
BPA-NN [388]	PA	Projection	$O(B)$	$O(B)$
BSGD+projection [386]	OGD	Projection	$O\left(B^2\right)$	$O\left(B^2\right)$
BSGD+merging [386]	OGD	Merging	$O(B)$	$O(B)$

Given a training set $\mathcal{D} = \{(\mathbf{x}_t, y_t), t = 1, \ldots, T\}$ where $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \{-1, +1\}$, and a set of m kernel functions $\mathcal{K} = \{\kappa_i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, i = 1, \ldots, m\}$. MKL learns a kernel-based prediction function by identifying an optimal combination of the m kernels, denoted by $\theta = (\theta_1, \ldots, \theta_m)$, to minimize the margin-based classification error, which can be cast into the optimization below:

$$\min_{\theta\in\Delta} \min_{f\in\mathcal{H}_{K(\theta)}} \frac{1}{2}|f|^2_{\mathcal{H}_{K(\theta)}} + C\sum_{t=1}^{n} \ell(f(\mathbf{x}_t), y_t) \qquad (11)$$

where $\Delta = \{\theta \in \mathbb{R}^m_+ | \theta^\top \mathbf{1}_m = 1\}, K(\theta)(\cdot,\cdot) = \sum_{i=1}^m \theta_i \kappa_i(\cdot,\cdot),\ \ \ell(f(\mathbf{x}_t), y_t) = \max(0, 1 - y_t f(\mathbf{x}_t))$.In the above, we use notation $\mathbf{1}_T$ to represent a vector of T dimensions with all its elements being 1. It can also be cast into the following mini-max optimization problem:

$$\min_{\theta\in\Delta} \max_{\alpha\in\Xi} \left\{ \alpha^\top \mathbf{1}_T - \frac{1}{2}(\alpha \circ \mathbf{y})^\top \left(\sum_{i=1}^m \theta_i K^i \right) (\alpha \circ \mathbf{y}) \right\} \qquad (12)$$

where $K^i \in \mathbb{R}^{T\times T}$ with $K^i_{j,l} = \kappa_i(\mathbf{x}_j, \mathbf{x}_l)$, $\Xi = \left\{ \alpha | \alpha \in [0, C]^T \right\}$, and $\circ$ defines the element-wise product between two vectors. The above batch MKL optimization has been extensively studied [306,405], but an efficient solution remains an open challenge.

Some efforts of online MKL studies [186,270] have attempted to solve batch MKL optimization via online learning. Unlike these approaches that are mainly concerned in optimizing the optimal kernel combination as regular MKL, another framework of *Online Multiple Kernel Learning* (OMKL) [187,175,412] is focused on exploring effective online combination of multiple kernel classifiers via a significantly more efficient and scalable way. Specifically, the OMKL in [187,175] learns a kernel-based prediction function by selecting a subset of predefined kernel functions in an online learning fashion, which is in general more challenging than typical online learning because both the kernel classifiers and the subset of selected kernels are unknown, and more importantly the solutions to the kernel classifiers and their combination weights are correlated. [175] proposed novel algorithms based on the fusion of two types of online learning algorithms, i.e., the *Perceptron* algorithm that learns a classifier for a given kernel, and the *Hedge* algorithm [128] that combines classifiers by linear weights. Some stochastic selection strategies were also proposed by randomly selecting a subset of kernels for combination and model updating to further improve the efficiency. These methods were later extended for regression [320,321], learning from data with time-sensitive patterns [323] and imbalanced data streams[324]. In addition, there have been budgeting approaches to make OMKL scalable [263]. [143] perform online multi-kernel learning based on graph-structured feedback.

There also exist adaptive approaches to automatically tune the bandwidth parameters of the kernels during the online learning process [420,419].

3.7. Online to batch conversion

Let us denote by $\mathcal{A}$ an online learning algorithm for the purpose of training a binary classifier from a sequence of training examples. On each round, the algorithm receives an instance $\mathbf{x}_t \in \mathbb{R}^d$, the algorithm chooses a vector $\mathbf{w}_t \in \mathcal{S} \subseteq \mathbb{R}^d$ to predict the class label of the instance, i.e., $\hat{y}_t = \text{sign}(\mathbf{w}_t^\top \mathbf{x}_t)$. After that, the environment responds by disclosing the true label y_t and some convex loss function $\ell(\mathbf{w}; (\mathbf{x}_t, y_t))$, and the algorithm suffers a loss $\ell_t(\mathbf{w}_t)$ at the end of this round. For such a setting, consider a sequence of T rounds $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$, the online algorithm aims to minimize the following regret

$$\text{Reg}_\mathcal{A}(T) = \sum_{t=1}^T \ell(\mathbf{w}_t; (\mathbf{x}_t, y_t)) - \min_{\mathbf{w}\in\mathcal{S}} \sum_{t=1}^T \ell(\mathbf{w}; (\mathbf{x}_t, y_t))$$

However, for batch training setting, we are more interested in finding a model $\hat{\mathbf{w}}$ with good generalization ability, i.e., we want to achieve a small excess risk defined as

$$R(\hat{\mathbf{w}}) - \min_{\mathbf{w}\in\mathcal{S}} R(\mathbf{w})$$

where the generalization risk is $R(\mathbf{w}) = \mathbb{E}_{(\mathbf{x},y)}[\ell(w; (x, y))]$, and $(\mathbf{x}, y)$ satisfies a fixed unknown distribution.

Therefore, we would like to study the generalization performance of online algorithms through the Online to Batch Conversion [68,385,193,223], where the conversion relates the regret of the online algorithm to its generalization performance.

3.7.1. A general conversion theory

We now consider the generalization ability of online learning under the assumption that the loss $\ell(\mathbf{w}; (\mathbf{x}, y))$ is strongly convex, which is reasonable as many loss functions (e.g., square loss) are strongly convex, and even if some loss (e.g., hinge loss) is not strongly convex, we can impose some regularization term (e.g., $\frac{1}{2}\|\cdot\|$) to achieve strong convexity. We denote the dual norm of $\|\cdot\|$ as $\|\cdot\|_*$, where $\|\mathbf{v}\|_* = \sup_{\|\mathbf{w}\|\leqslant 1} \mathbf{v}^\top \mathbf{w}$. Let $Z = (\mathbf{x}, y)$ be a random variable taking values in some space $\mathcal{Z}$. Our goal is to minimize $R(\mathbf{w}) = \mathbb{E}_Z[\ell(\mathbf{w}; Z)]$ over $\mathbf{w} \in \mathcal{S}$. Specifically, we assume the loss $\ell : \mathcal{S} \times \mathcal{Z} \to [0, B]$ satisfies the following assumption:

Assumption LIST (LIpschitz and STrongly convex assumption) For all $z \in \mathcal{Z}$, the function $\ell_z(\mathbf{w}) = \ell(\mathbf{w}; z)$ is convex in $\mathbf{w}$ and satisfies:

1. ℓ_z has Lipschitz constant L w.r.t. the norm $\|\cdot\|$, i.e., $|\ell_z(\mathbf{w}) - \ell_z(\mathbf{w}')| \leqslant L\|\mathbf{w} - \mathbf{w}'\|$.
2. ℓ_z is λ-strongly convex w.r.t. $\|\cdot\|$, i.e., $\forall\theta \in [0, 1]$, $\forall \mathbf{w}, \mathbf{w}' \in \mathcal{S}$,

$$\ell_z(\theta\mathbf{w} + (1-\theta)\mathbf{w}') \leqslant \theta\ell_z(\mathbf{w}) + (1-\theta)\ell_z(\mathbf{w}') - \frac{\lambda}{2}\theta(1-\theta)\|\mathbf{w} - \mathbf{w}'\|^2.$$

For this kind of loss function, we consider an online learning setting where $Z_1, \ldots, Z_T$ are given sequentially in i.i.d. We then have

$$\mathbb{E}[\ell(\mathbf{w}; Z_t)] = \mathbb{E}[\ell(\mathbf{w}, (\mathbf{x}_t, y_t))] := R(\mathbf{w}), \ \forall t, \ \mathbf{w} \in \mathcal{S}.$$

Now consider an online learning algorithm $\mathcal{A}$, which is initialized as $\mathbf{w}_1$. Whenever Z_t is given, model $\mathbf{w}_t$ is updated to $\mathbf{w}_{t+1}$. Let $\mathbb{E}_t[\cdot]$ denote conditional expectation w.r.t. $Z_1, \ldots, Z_t$, we have $\mathbb{E}_t[\ell(\mathbf{w}_t; Z_t)] = R(\mathbf{w}_t)$. Using the above assumptions and the Freedman's inequality leads to the following theorem for the generalization ability of online learning

Theorem 1. ([68])*Under the assumption LIST, we have the following inequality, with probability at least* $1 - 4\delta \ln T$,

$$\frac{1}{T}\sum_{t=1}^T R(\mathbf{w}_t) - R(\mathbf{w}_*) \leqslant \frac{\text{Reg}_\mathcal{A}(T)}{T} + 4\sqrt{\frac{L^2 \ln\frac{1}{\delta}}{\lambda}} + \frac{\sqrt{\text{Reg}_\mathcal{A}(T)}}{T} + \max\left(\frac{16L^2}{\lambda}, 6B\right)\frac{\ln\frac{1}{\delta}}{T},$$

where $\mathbf{w}_* = \arg\min_{\mathbf{w}\in\mathcal{S}} R(\mathbf{w})$. *Further, using Jensen's inequality,* $\frac{1}{T}\sum_t R(\mathbf{w}_t)$ *can be replaced by* $R(\bar{\mathbf{w}}_T)$ *where* $\bar{\mathbf{w}}_T = \frac{1}{T}\sum_t \mathbf{w}_t$.

If the assumption LIST is satisfied by $\ell_z(\mathbf{w})$, then the Online Gradient Descent (OGD) algorithm that generates $\mathbf{w}_1, \ldots, \mathbf{w}_T$ has the following regret $\text{Reg}_\mathcal{A}(T) \leqslant \frac{L^2}{2\lambda}(1 + \ln T)$. Plugging this inequality back into the theorem and using $(1 + \ln T)/(2T) \leqslant \ln T/T$, $\forall T \geqslant 3$ gives the following Corollary.

Corollary 1. *Suppose assumption LIST holds for $\ell_z(\mathbf{w})$. Then the Online Gradient Descent (OGD) algorithm that generates $\mathbf{w}_1, \ldots, \mathbf{w}_T$ and finally outputs $\bar{\mathbf{w}}_T = \frac{1}{T}\sum_t \mathbf{w}_t$, satisfies the following inequality for its generalization ability, with probability at least $1 - 4\delta \ln T$,*

$$R(\bar{\mathbf{w}}_T) - R(\mathbf{w}_*) \leqslant \frac{L^2 \ln T}{\lambda T} + \sqrt{\ln\frac{1}{\delta}\frac{4L^2\sqrt{\ln T}}{\lambda T}} + \max\left(\frac{16L^2}{\lambda}, 6B\right)\frac{\ln\frac{1}{\delta}}{T},$$

for any $T \geqslant 3$, where $\mathbf{w}_* = \arg\min_{\mathbf{w}\in\mathcal{S}} R(\mathbf{w})$.

3.7.2. Other conversion theories

Online to batch conversion has been studied in literature [253,68,424,70,385,351]. For general convex loss functions, [68] proved the following generalization ability of online learning algorithm with probability at least $1 - \delta$

$$R(\bar{\mathbf{w}}_T) \leqslant \frac{1}{T}\sum_{t=1}^{T}\ell(\mathbf{w}_t; z_t) + \sqrt{\frac{2}{T}\ln\frac{1}{\delta}}$$
$$= \frac{\mathrm{Reg}_{\mathcal{A}}(T)}{T} + \min_{\mathbf{w}\in\mathcal{S}}\frac{1}{T}\sum_{t=1}^{T}\ell(\mathbf{w}; z_t) + \sqrt{\frac{2}{T}\ln\frac{1}{\delta}}, \qquad (13)$$

where the loss $\ell \leqslant 1$. [424] is another work that explicitly goes by the exponential moment method to drive sharper concentration results. In addition, [70] improved their initial generalization bounds using Bernstein's inequality by assuming $\ell(\cdot) \leqslant 1$, and proves the following inequality with probability at least $1 - \delta$

$$R(\hat{\mathbf{w}}) \leqslant \frac{1}{T}\sum_{t=1}^{T}\ell(\mathbf{w}_t; z_t) + O\left(\frac{\ln\left(T^2/\delta\right)}{T} + \sqrt{\frac{1}{T}\sum_{t=1}^{T}\ell(\mathbf{w}_t; z_t)\frac{\ln\left(T^2/\delta\right)}{T}}\right).$$

where $\hat{\mathbf{w}}$ is selected from $\mathbf{w}_1, \ldots, \mathbf{w}_T$, by minimizing a specifically designed penalized empirical risk. In particular, the generalization risk converges to $\frac{1}{T}\sum_{t=1}^{T}\ell(\mathbf{w}_t; z_t)$ at rate $O\left(\sqrt{\ln T^2/T}\right)$ and vanishes at rate $O\left(\ln T^2/T\right)$ whenever the loss $\sum_{t=1}^{T}\ell(\mathbf{w}_t; z_t)$ is $O(1)$.

4. Applied online learning for supervised learning

4.1. Overview

In this section, we survey the most representative algorithms for a group of non-traditional online learning tasks, wherein the supervised online algorithms cannot be used directly. These algorithms are motivated by new problem settings and applications which follow the traditional online setting, where the data arrives in a sequential manner. However, there was a need to develop new algorithms which were suited to these scenarios. Our review includes cost-sensitive online learning, online multi-task learning, online multi-view learning, online transfer learning, online metric learning, online collaborative filtering, online learning structured prediction, distributed online learning, online learning with neural networks, and online portfolio selection.

4.2. Cost-sensitive online learning

In a supervised classification task, traditional online learning methods are often designed to optimize mistake rate or equivalently classification accuracy. However, it is well-known that classification accuracy becomes a misleading metric when dealing with class-imbalanced data which is common for many real-world applications, such as anomaly detection [404], fraud detection, intrusion detection [238], etc. To address this issue, cost-sensitive online learning [88] represents a family of online learning algorithms that are designed to take care of different misclassification costs of different classes in a class-imbalanced classification task. Next, we briefly survey these algorithms.

Perceptron Algorithms with Uneven Margin (PAUM). PAUM [246] is a cost-sensitive extension of Perceptron [311] and the Perceptron with Margins (PAM) algorithms [211]. Perceptron makes an update only when there is a mistake, while PAM tends to make more aggressive updates by checking the margin instead of mistake. PAM makes an update whenever $y_t\mathbf{w}_t^\top\mathbf{x}_t \leqslant \tau$, where $\tau \in \mathbb{R}^+$ is a fixed parameter controlling the aggressiveness. To deal with class imbalance, PAUM extends PAM via an uneven margin setting, i.e., employing different margin parameters for the two classes: τ+ and τ-. Consequently, the update becomes $y_t\mathbf{w}_t^\top\mathbf{x}_t \leqslant \tau_{y_t}$. By properly adjusting the two parameters, PAUM achieves cost-sensitive updating effects for different classes. One of major limitations with PAUM is that it does not directly optimize a predefined cost-sensitive measure, thus, it does not fully resolve the cost-sensitive challenge.

Cost-sensitive Passive Aggressive (CPA). CPA [95] was proposed as a cost-sensitive variant of the PA algorithms. It was originally designed for multi-class classification by the following prediction rule:

$\hat{y}_t = \arg\max_y(\mathbf{w}_t\Phi(\mathbf{x}_t, y))$, where Φ is a feature mapping function that maps $\mathbf{x}_t$ to a new feature according to the class y. For simplicity, we restrict the discussion on the binary classification setting. Using $\Phi(\mathbf{x}, y) = \frac{1}{2}y\mathbf{x}$, we will map the formulas to our setting. The prediction rule is: $\hat{y}_t = \mathrm{sign}(\mathbf{w}_t^\top\mathbf{x}_t)$. We define the cost-sensitive loss as

$$\ell(\mathbf{w}, \mathbf{x}, y) = \mathbf{w}\cdot\Phi(\mathbf{x}, \hat{y}) - \mathbf{w}\cdot\Phi(\mathbf{x}, y) + \sqrt{\rho(y, \hat{y})},$$

where $\rho(y_1, y_2)$ is the function define to distinguish the different cost of different kind misclassifications and we have assumed $\rho(y, y) = 0$. When being converted to binary setting, the loss becomes

$$\ell(\mathbf{w}, \mathbf{x}, y) = \begin{cases} 0 & y_t = \hat{y} \\ |\mathbf{w}^\top\mathbf{x}| + \sqrt{\rho(y, \hat{y})} & y_t \neq \hat{y} \end{cases}$$

The mistake depends on the prediction confidence and the loss type. We omit the detailed update steps since it follows the similar optimization as PA learning as discussed before. Similar to PAUM, this algorithm also is limited in that it does not optimize a cost-sensitive measure directly.

Cost-Sensitive Online Gradient Descent (CSOGD). Unlike traditional OGD algorithms that often optimize accuracy, CSOGD [377,380,438,437] applies OGD to directly optimize two cost-sensitive measures:

- [(1)] maximizing the weighted sum of *sensitivity* and *specificity*, i.e, $sum = \eta_p \times sensitivity + \eta_n \times specificity$, where the two weights satisfy $0 \leqslant \eta_p, \eta_n \leqslant 1$ and $\eta_p + \eta_n = 1$.
- [(2)] minimizing the weighted *misclassificationcost*, i.e., $cost = c_p \times M_p + c_n \times M_n$, where M_p and M_n are the number of false negatives and false positives respectively, $0 \leqslant c_p, c_n \leqslant 1$ are the cost parameters for positive and negative classes, respectively, and we assume $c_p + c_n = 1$.

The objectives can be equivalently reformulated into the following objective:

$$\sum_{y_t=+1}\rho\mathbb{I}_{(y_t\mathbf{w}\cdot\mathbf{x}_t<0)} + \sum_{y_t=-1}\mathbb{I}_{(y_t\mathbf{w}\cdot\mathbf{x}_t<0)}$$

where we set $\rho = \frac{\eta_p T_n}{\eta_n T_p}$ when maximizing the weighted sum, T_p and T_n are the number of positive and negative instances respectively; when minimizing the weighted misclassification cost, we instead set $\rho = \frac{c_p}{c_n}$. The objective is however non-convex, making it hard to optimize directly.

Instead of directly optimizing the non-convex objective, we attempt to optimize a convex surrogate. Specifically, we replace the indicator function $\mathbf{I}_{(\cdot)}$ by a convex surrogate, and attempt to optimize either one of the following modified hinge-loss functions at each online iteration:

$$\ell^I(\mathbf{w};(\mathbf{x},y)) = \max(0, \rho * \mathbb{I}_{(y=1)} + \mathbb{I}_{(y=-1)} - y(\mathbf{w}\cdot\mathbf{x}))$$

$$\ell^{II}(\mathbf{w};(\mathbf{x},y)) = (\rho * \mathbb{I}_{(y=1)} + \mathbb{I}_{(y=-1)}) * \max(0, 1 - y(\mathbf{w}\cdot\mathbf{x}))$$

One can then derive cost-sensitive ODG (CSOGD) algorithms by applying OGD to optimize either one of the above loss functions. The detailed algorithms can be found in [377]. Two recent works extend the problem setting to cost-sensitive classification of multi-class problem [382,426]. Further there are efforts to do cost-sensitive online learning with kernels [430,178].

Online AUC Maximization. Instead of optimizing accuracy, some online learning studies have attempted to directly optimize the Area Under the ROC curve (AUC), i.e.,

$$\mathrm{AUC}(\mathbf{w}) = \frac{\sum_{i=1}^{T_+}\sum_{j=1}^{T_-}\mathbb{I}_{\mathbf{w}\cdot\mathbf{x}_i^+ > \mathbf{w}\cdot\mathbf{x}_j^-}}{T_+T_-} = 1 - \frac{\sum_{i=1}^{T_+}\sum_{j=1}^{T_-}\mathbb{I}_{\mathbf{w}\cdot\mathbf{x}_i^+ \leqslant \mathbf{w}\cdot\mathbf{x}_j^-}}{T_+T_-}$$

where $\mathbf{x}^+$ is a positive instance, $\mathbf{x}$- is a negative instance, T_+ is the total number of positive instances and T- is the total number of negative instances. AUC measures the probability for a randomly drawn positive instance to have a higher decision value than a randomly sampled negative instance, and it is widely used in many applications. Optimizing AUC online is however very challenging.

First of all, in the objective, the term $\sum_{i=1}^{T_+}\sum_{j=1}^{T_-}\mathbb{I}_{\mathbf{w}\cdot\mathbf{x}_i^+ \leqslant \mathbf{w}\cdot\mathbf{x}_j^-}$ is non-convex. A common way is to replace the indicator function by a convex surrogate, e.g., a hinge loss function

$$\ell\left(\mathbf{w}, \mathbf{x}_i^+ - \mathbf{x}_j^-\right) = \max\left\{0, 1 - \mathbf{w}\left(\mathbf{x}_i^+ - \mathbf{x}_j^-\right)\right\}$$

Consequently, the goal of online AUC maximization is equivalent to minimizing the accumulated loss $\mathcal{L}_t(\mathbf{w})$ over all previous iterations, where the loss at the t-th iteration is

$$\mathcal{L}_t(\mathbf{w}) = \mathbb{I}_{y_t=1}\sum_{\tau=1}^{t-1}\mathbb{I}_{y_\tau=-1}\ell(\mathbf{w}, \mathbf{x}_t - \mathbf{x}_\tau) + \mathbb{I}_{y_t=-1}\sum_{\tau=1}^{t-1}\mathbb{I}_{y_\tau=1}\ell(\mathbf{w}, \mathbf{x}_\tau - \mathbf{x}_t)$$

The above takes the sum of the pairwise hinge loss between the current instance $(\mathbf{x}_t, y_t)$ and all the received instances with the opposite class $-y_t$. Despite being convex, it is however impractical to directly optimize the above objective in online setting since one would need to store all the received instances and thus lead to growing computation and memory cost.

The Online AUC Maximization method in [435] proposed a novel idea of exploring *reservoir sampling* techniques to maintain two buffers, B_+ and B_- of size N_+ and N_-, which aim to store a sketch of historical instances. Specifically, when receiving instance $(\mathbf{x}_t, y_t)$, it will be added to buffer B_{y_t} whenever it is not full, i.e. $|B_{y_t}| < N_{y_t}$. Otherwise, $\mathbf{x}_t$ randomly replaces one instance in the buffer with probability $\frac{N_{y_t}}{N_{y_t}^{t+1}}$, where $N_{y_t}^{t+1}$ is the total number of instances with class y_t received so far. Reservoir sampling is able to guarantee the instances in the buffers maintain an unbiased sampling of the original full dataset. As a result, the loss $\mathcal{L}_t(\mathbf{w})$ can be approximated by only considering the instances in the buffers, and the classifier $\mathbf{w}$ can be updated by either OGD or PA algorithms.

Others. To improve the study in [435], a number of following studies have attempted to make improvements from different aspects. For example, the study in [385] generalized online AUC maximization as online learning with general pairwise loss functions, and offered new generalization bounds for online AUC maximization algorithms similar to [435]. The bounds were further improved by [196] which employs a generic decoupling technique to provide Rademacher complexity-based generalization bounds. In addition, the work in [135] overcomes the buffering storage cost by developing a regression-based algorithm which only needs to maintain the first and second-order statistics of training data in memory, making the resulting storage requirement independent from the training size. The recent work in [111] presented a new second-order AUC maximization method by improving the convergence using the adaptive gradient algorithm. The stochastic online AUC maximization (SOLAM) algorithm [414] formulates the online AUC maximization as a stochastic saddle point problem and greatly reduces the memory cost.

4.3. Online multi-task learning

Multi-task Learning [64] is an approach that learns a group of related machine learning tasks together. By considering the relationship between different tasks, multi-task learning algorithms are expected to achieve better performance than algorithms that learn each task individually. Batch multi-task learning problems are usually solved by transfer learning methods [292] which transfer the knowledge learnt from one task to another similar tasks. In Online Multi-task Learning (OML) [108,236,237,411,172], however, the tasks have to be solved in parallel with instances arriving sequentially, which makes the problem more challenging.

During time t, each of the task $i \in \{1, \dots K\}$ receives an instance $\mathbf{x}_{i,t} \in \mathbb{R}^{d_i}$, where d_i is the feature dimension of task i. The algorithm then makes a prediction for each task i based on the current model $\mathbf{w}_{i,t}$ as $\hat{y}_{i,t} = \mathrm{sign}\left(\mathbf{w}_{i,t}^\top \mathbf{x}_{i,t}\right)$. After making the prediction, the true labels $y_{i,t}$ are revealed and we get a loss function vector $\ell_{i,t} \in \mathbb{R}_+^K$. Finally, the models are updated by considering the loss vector and task relationship.

A straightforward baseline algorithm is to parallel update all the classifiers $\mathbf{w}_i, i \in \{1, \dots, K\}$. OML algorithm should utilize the relationships between tasks to achieve higher accuracy compared with the baseline. The multitask Perceptron algorithm [66] is a pioneering work of OML that considers the inter-task relationship. Assuming that a matrix $A \in \mathbb{R}^{K*K}$ is known and fixed, we can update the modePrasanthi Nairl i when an instance $\mathbf{x}_{j,t}$ for task j is received as follows:

$$\mathbf{w}_{i,t+1} = \mathbf{w}_{i,t} + y_{j,t}A_{j,i}^{-1}\mathbf{x}_{j,t}$$

Other approaches in [319,373] learned to optimize the relationship matrix, which also offers the flexibility of using a time-varying relationship.

Apart from learning the relationship explicitly, another widely used approach in OML field is to add some structure regularization terms to the original objective function [122,407,214]. For example, we may assume that each model is made up of two parts, a shared part across all tasks $\mathbf{w}_0$ and an individual part $\mathbf{v_i}$, i.e., $\mathbf{w}_i = \mathbf{w}_0 + \mathbf{v_i}$ where the common part helps to take advantage of the task similarity. Now the regularized loss becomes

$$\sum_{i=1}^K \left(\ell_{i,t} + ||\mathbf{v_i}||_\mathbf{2}^\mathbf{2}\right) + \lambda||\mathbf{w_0}||_\mathbf{2}^\mathbf{2}$$

It was also improved using more complex inter-task relationship [278].

4.4. Online multi-view learning

Multi-view learning deals with problems where data are collected from diverse domains or obtained from various feature extractors. By exploring features from different views, multi-view learning algorithms are usually more effective than single-view learning. In literature, there many surveys that offer comprehensive summary of state-of-the-art methods in multi-view learning in batch setting [347,403,245], while few works tried to address this problem in online setting.

Two-view PA. We first introduce a seminal work, the two-view online passive aggressive learning (Two-view PA) algorithm [281], which is motivated by the famous single-view PA algorithm [95] and the two-view SVM algorithm [123] in batch setting.

During each iteration t, the algorithm receives an instance $(\mathbf{x}_t^A, \mathbf{x}_t^B, y_t)$, where $\mathbf{x}_t^A \in \mathbb{R}^n$ is the feature vector in the first view, $\mathbf{x}_t^B \in \mathbb{R}^m$ is for the second view and $y_t \in \{1, -1\}$ is the label. The goal is to learn two classifiers $\mathbf{w}^A \in \mathbb{R}^n$ and $\mathbf{w}^B \in \mathbb{R}^m$, each for one view, and make accuracy prediction with their combination

$$\hat{y}_t = \text{sign}(\mathbf{w}^A \cdot \mathbf{x}^A + \mathbf{w}^B \cdot \mathbf{x}^B).$$

Thus the hinge loss at iteration t is redefined as

$$\ell_t(\mathbf{w}_t^A, \mathbf{w}_t^B) = \max\left(0, 1 - \frac{1}{2} y_t(\mathbf{w}^A \cdot \mathbf{x}_t^A + \mathbf{w}^B \cdot \mathbf{x}_t^B)\right)$$

In the single-view PA algorithm, the objective function in each iteration is a balance between two desires: minimizing the loss function at the current instance and minimizing the change made to the classifier. While to utilize the special information in the multi-view data, an additional term that measures the agreement between two terms is added. Thus, the optimization is as follows,

$$(\mathbf{w}_{t+1}^A, \mathbf{w}_{t+1}^B) = \arg\min_{\mathbf{w}^A, \mathbf{w}^B} \tfrac{1}{2}||\mathbf{w}^A - \mathbf{w}_t^A||_2^2 + \tfrac{1}{2}||\mathbf{w}^B - \mathbf{w}_t^B||_2^2 \\ + C\ell_t(\mathbf{w}_t^A, \mathbf{w}_t^B) + \gamma|y_t \mathbf{w}^A \cdot \mathbf{x}_t^A - y_t \mathbf{w}^B \cdot \mathbf{x}_t^B|$$

where γ and C are weight parameters. Fortunately, this optimization problem has a closed form solution.

Other related works: Other than solving classification tasks, online multi-view learning has been explored for solving similarity learning or distance metric learning, such as Online multimodal deep similarity learning [395] and online multi-modal distance metric learning [396].

4.5. Online transfer learning

Transfer learning aims to address the machine learning tasks of building models in a new target domain by taking advantage of information from another existing source domain through knowledge transfer. Transfer learning is important for many applications where training data in a new domain may be limited or too expensive to collect. There are two different problem settings, *homogeneous* setting where the target domain shares the same feature space as the old/source one, and *heterogeneous* setting where the feature space of the target domain is different from that of the source domain. Although several surveys on transfer learning are available [344,292], most of the referred algorithms are in batch setting.

Online Transfer Learning (OTL) algorithms aim to learn a classifier $f : \mathbb{R}^d \to \mathbb{R}$ from a well-trained classifier $h : \mathbb{R}^{d'} \to \mathbb{R}$ in the source domain and a group of sequentially arriving instances $\mathbf{x}_t \in R^d, t = 1, \ldots, T$ in the target domain. For conciseness, we will use the previous notations for the online classification task. We first introduce a pioneer work of OTL [433,428].

Homogeneous Setting. One key challenge of this task is to address the concept drifting issue that often occurs in this scenario. The algorithm in homogeneous setting ($d = d'$) is based on the ensemble learning approach. At time t, an instance $\mathbf{x}_t$ is received. The algorithm makes a prediction based on the weighted average of the classifier in the source domain $h(\mathbf{x}_t)$ and the current classifier in the target domain $f_t(\mathbf{x}_t)$,

$$\hat{y}_t = \text{sign}\left(w_{1,t}\Pi(h(\mathbf{x}_t)) + w_{2,t}\Pi(f_t(\mathbf{x}_t)) - \frac{1}{2}\right)$$

where $w_{1,t} > 0, w_{2,t} > 0$ are the weights for the two functions and need to be updated during each iteration. Π is a normalization function, i.e. $\Pi(a) = \max(0, \min(1, \frac{a+1}{2}))$.

In addition to updating the function f_t by using some online learning algorithms, the weights $w_{1,t}$ and $w_{2,t}$ should also be updated. One suggested scheme is

$$w_{1,t+1} = C_t w_{1,t} \exp(-\eta\ell^*(h)), \quad w_{2,t+1} = C_t w_{2,t} \exp(-\eta\ell^*(f_t))$$

where C_t is a normalization term to keep $w_{1,t+1} + w_{2,t+1} = 1$ and $\ell^*(g) = (\Pi(g(\mathbf{x}_t)) - \Pi(y_t))^2$.

[115] extend the homogeneous transfer learning by applying a distribution discrepancy minimization approach to improve the online transfer learning process. [349] propose to do transfer learning via an L1 regularization, which accounts for differences between the source and target parameters.

Heterogeneous Setting. Since heterogeneous OTL is generally very challenging, we consider one simpler case where the feature space of the source domain is a subset of that of the target domain. Without loss of generality, we assume the first d' dimensions of $\mathbf{x}_t$ represent the old features, denoted as $\mathbf{x}_t^{(1)} \in \mathbb{R}^{d'}$. The other dimensions form a feature vector $\mathbf{x}_t^{(2)} \in \mathbb{R}^{d-d'}$. The key idea is to adopt a co-regularization principle of online learning two classifiers $f_t^{(1)}$ and $f_t^{(2)}$ simultaneously from the two views, and predict an unseen example on the target domain by

$$\hat{y}_t = \text{sign}\left(\frac{1}{2} f_t^{(1)}\left(\mathbf{x}_t^{(1)}\right) + \frac{1}{2} f_t^{(2)}\left(\mathbf{x}_t^{(2)}\right)\right)$$

The function from source domain $h(\mathbf{x}^{(1)})$ is used to initialize $f_t^{(1)}$. The update strategy at time t is

$$\left(f_{t+1}^{(1)}, f_{t+1}^{(2)}\right) = \arg\min_{f^{(1)}, f^{(2)}} \frac{\gamma_1}{2}||f^{(1)} - f_t^{(1)}||_{\mathcal{H}}^2 + \frac{\gamma_1}{2}||f^{(2)} - f_t^{(2)}||_{\mathcal{H}}^2 + C\ell_t$$

where γ_1, γ_2 and C are positive parameters and ℓ_t is the hinge loss.

Other related work. Multi-source Online Transfer Learning (MSOTL) [140,355] solves a more challenging problem where k classifiers $h_1, \ldots h_k$ are provided by k sources. The goal is to learn the optimal combination of the k classifiers and the online updated classifier f_t. A naive solution is to construct a new $d + k$ dimensional feature representation $\mathbf{x}_t' = [\mathbf{x}_t, h_1(\mathbf{x}_t), \ldots, h_k(\mathbf{x}_t)]$ and the online classifier in this new feature space. An extension of MSOTL [139] aims to deal with transfer learning problem under two disadvantageous assumptions, negative transfer where instead of improving performance, transfer learning from highly irrelevant sources degrades the performance on the target domain, and imbalanced distributions where examples in one class dominate. The Co-transfer Learning algorithm [42,43] considers the transfer learning problem not only in multi-source setting but also in the scenario where a large group of instances are unlabeled.

4.6. Online metric learning

Distance metric learning (DML) [408] or similarity learning [160] is an important problem in machine learning, which enjoys many real-world applications, such as image retrieval, classification and clustering. The goal of classic DML is to seek a distance matrix $A \in \mathbb{R}^{d\times d}$ that defines the Mahalanobis distance between any two instances $\mathbf{x}_i \in R^d$ and $\mathbf{x}_j \in \mathbb{R}^d$

$$d_A(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top A(\mathbf{x}_i - \mathbf{x}_j) = ||W\mathbf{x}_i - W\mathbf{x}_j||_2^2$$

Typically, matrix $A \succeq 0$ is required to be symmetric positive semi-definite, i.e., there exist a matrix $W \in \mathbb{R}^{d\times d}$ such that $A = W^\top W$. It is often hard to collect training data with the exact true values of distances. Therefore, there are two types of problem settings for online DML: 1) Pairwise data, where at each round t the learner receives a pair of instances $(\mathbf{x}_t^1, \mathbf{x}_t^2)$ and a label y_t which is $+1$ if the pair is similar and -1 otherwise; 2) Triple data, where at each round t the learner receives a triple $(\mathbf{x}_t, \mathbf{x}_t^+, \mathbf{x}_t^-)$, with the feedback that $d_A(\mathbf{x}_t, \mathbf{x}_t^+) > d_A(\mathbf{x}_t, \mathbf{x}_t^-)$. The goal of online learning is to minimize the accumulated loss during the whole learning process $\sum_{t=1}^T \ell_t(A)$, where ℓ_t is the loss suffered from imperfect prediction at round t. When evaluating the output model for online-to-batch-conversion, we may use the metric in information retrieval to evaluate the actual performance, such as mean average precision (mAP) or precision-at-top-k.

Below, we briefly introduce a few representative work for DML in online setting.

Pseudo-metric Online Learning (POLA) The POLA algorithm [335] learns the distance matrix A from a stream of pairwise data. The loss at time t is an adaptation of the hinge loss

$$\ell_t(A, b) = \max\{0, 1 - y_t(b - d_A(\mathbf{x}_t^1, \mathbf{x}_t^2))\}$$

where b is the adaptive threshold value for similarity and will be updated incrementally along with matrix A. We denote $(A, b) \in \mathbb{R}^{d^2+1}$ as the new variable to learn. The update strategy mainly follows the PA approach

$$\left(A_{t+\frac{1}{2}}, b_{t+\frac{1}{2}}\right) = \arg\min_{(A,b)} ||(A, b) - (A_t, b_t)||_2^2 \quad s.t. \ell_t(A, b) = 0$$

The solution $\left(A_{t+\frac{1}{2}}, b_{t+\frac{1}{2}}\right)$ ensures correct prediction to current pair while makes the minimal change to the previous model. Then, the algorithm projects this solution to the feasible space $\{(A, b) : A \succeq 0, b \geqslant 1\}$ to obtain the updated model (A_{t+1}, b_{t+1}). Like PA, one can generalize POLA to soft-margin variants to be robust to noise.

Another similar work named Online Regularized Metric Learning [188] is simpler due to the adoption of fixed threshold, and adopts the following loss function

$$\ell_t(A) = \max(0, b - y_t(1 - d_A(\mathbf{x}_t^1, \mathbf{x}_t^2)))$$

whose gradient is

$$\nabla \ell_t(A) = y_t(\mathbf{x}_t^1 - \mathbf{x}_t^2)(\mathbf{x}_t^1 - \mathbf{x}_t^2)^\top$$

At time t, if the prediction is incorrect, the algorithm updates the matrix A by projecting the OGD updated matrix into the positive definite space.

Information Theoretic Metric Learning (ITML). In the above algorithms, distances between two matrices A_t and A are usually defined using the Frobenius norm, i.e. $||A_t - A||_F^2$. The ITML algorithm [105,181] adopts a different definition from an information theoretic perspective. Given a Mahalanobis distance parameterized by A, its corresponding multivariate Gaussian distribution is $p(\mathbf{x}, A) = \frac{1}{Z}\exp(-\frac{1}{2}d_A(\mathbf{x}, \boldsymbol{\mu}))$. The difference between matrices is defined as the KL divergence between two distributions. Assuming all distributions have the same mean, the KL divergence can be calculated as

$$\mathrm{KL}(p(\mathbf{x}; A), p(\mathbf{x}, A_t)) = \mathrm{tr}\left(AA_t^{-1}\right) - \log\det\left(AA_t^{-1}\right) - d$$

Similar to the PA update strategy, during time t the matrix is updated by optimizing

$$A_{t+1} = \arg\min_{A\succeq 0} \mathrm{KL}(p(\mathbf{x}; A), p(\mathbf{x}, A_t)) + \eta\ell_t(A)$$

where $\eta > 0$ is a regularization parameter. This optimization has a closed-form solution. Online Algorithm for Scalable Image Similarity Learning (OASIS) The OASIS algorithm learns a bilinear similarity matrix $W \in \mathbb{R}^d$ from a stream of triplet data, where the bilinear similarity measure between two instances is defined as

$$S_W(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top W\mathbf{x}_j$$

During time t, one triplet $(\mathbf{x}_t, \mathbf{x}_t^+, \mathbf{x}_t^-)$ is received. Ideally, we expect the $\mathbf{x}_t$ is more similar to $\mathbf{x}_t^+$ than to $\mathbf{x}_t^-$, i.e. $S_W(\mathbf{x}_t, \mathbf{x}_t^+) > S_W(\mathbf{x}_t, \mathbf{x}_t^-)$. Similar to the PA algorithm, for a large margin, the loss function is defined as the hinge loss

$$\ell_t(W) = \max\{0, 1 - S_W(\mathbf{x}_t, \mathbf{x}_t^+) + S_W(\mathbf{x}_t, \mathbf{x}_t^-)\}$$

The optimization problem to solve for updating W_t is

$$W_{t+1} = \arg\min_W \frac{1}{2}||W - W_t||_F^2 + C\xi s.t. \quad \ell_t(W) \leqslant \xi \text{ and } \xi \geqslant 0$$

where C is the parameter controlling the trade-off. The OASIS algorithm differs from the previous work in several aspects. First, it does not require the similarity matrix W to be positive semi-definite and thus saves computational cost for the projection step. The bilinear similarity matrix may be better than the Mahalanobis distance for some applications. Third, the triplet data may be easier to collect in some applications.

Most of the previous methods assume a linear proximity function, [400] overcomes the limitaiton using a kernelized approach for metric learning. Another approach in [136,137] performs sparse online metric learning for very high dimensional data. There is also some work that solves the online similarity learning in an active learning setting [159], which significantly reduces the cost of collecting labeled data. Some work in [84,85] also applied techniques of the online similarity learning for real-world applications of mobile application recommendation and tagging. The series of work in [401,396] proposed the online multi-modal distance metric learning algorithms which learn distance metrics in multiple modalities, enabling multimedia information retrieval applications.

4.7. Online collaborative filtering

Collaborative Filtering (CF) [168,20] is an important learning technique for recommender systems. Different from content-based filtering techniques, CF algorithms usually require minimal knowledge about the features of items or users apart from the previous preferences. The fundamental assumption of CF is that if two users rate many items similarly, they expect to share common preference on some other items. Several survey papers in [338,346] gave detailed reviews of regular CF techniques. However, most of them assume batch settings. Below we introduce basics of CF and then review several popular online algorithms for CF tasks.

An online CF algorithm works on a sequence of observed ratings given by n users to m items. At time $t \in \{1, 2, \ldots, T\}$, the algorithm receives the index of a user $u^{(t)} \in \{1, 2, \ldots n\}$ and the index of an item $i^{(t)} \in \{1, 2, \ldots m\}$ and makes a prediction of the rating $\hat{r}_{u,i}^{(t)} \in \mathbb{R}$ based on the knowledge of the previous ratings. Then the real rating $r_{u,i}^{(t)} \in \mathbb{R}$ is revealed and the algorithm updates the model based on the loss suffered from the imperfect prediction, denoted as $\ell\left(\hat{r}_{u,i}^{(t)}, r_{u,i}^{(t)}\right)$. The goal of online CF is to minimize the Root Mean

Square Error (RMSE) or Mean Absolute Error(MAE) along the whole learning process, defined as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(r_{u,i}^{(t)} - \hat{r}_{u,i}^{(t)}\right)^2}, \quad \mathrm{MAE} = \frac{1}{T}\sum_{t=1}^{T}|r_{u,i}^{(t)} - \hat{r}_{u,i}^{(t)}|$$

CF techniques are generally categorized into two types: memory-based methods and model-based methods. Below briefly introduces some popular algorithms in each category.

Memory-Based CF Methods. This follows the instance-based learning paradigm:

1. Calculate the similarity score between any pairs of items. For example, the cosine similarity between item i and item j is defined as,

$$S_{i,j} = \frac{\sum_{u\in\mathcal{U}_i\cap\mathcal{U}_j} r_{ui}\cdot r_{u,j}}{\sqrt{\sum_{u\in\mathcal{U}_i} r_{ui}^2 \sum_{u\in\mathcal{U}_j} r_{u,j}^2}}$$

where $\mathcal{U}_i$ denotes the set of users that have rated item i.
2. For each item i, find its k nearest neighbor set $\mathcal{N}_i$ based on the similarity score.
3. Predict the rating $r_{u,i}$ as the weighted average of ratings from user u to the neighbors of item j, where the weight is proportional to the similarity.

We name the above described algorithm as item-based CF, while similarly, the predictions may also be calculated as the weighted average of ratings from similar users, which is called user-based CF method. Memory-based CF methods were used for some early generation recommendation systems, but very few is online learning approach. One reason is because of data sparsity, as the similarity score $S_{i,j}$ is only available when there is at least one common user that rates the two items i and j, which might be unrealistic during the beginning stage. Another challenge is the large time consumption when updating the large number of similarity scores incrementally with the arrival of new ratings. The Online Evolutionary Collaborative Filtering [259] algorithm provides an efficient similarity score updating method to address this problem.

Model-Based CF Methods. Memory-based online CF methods suffer two limitations, i.e., sensitivity due to data sparsity and inefficiency for similarity score update. To address these issues, extensive work hasbeen focused on model-based CF algorithms. One of the most successful approaches is the matrix factorization methodology [209], which assumes the rating by a user to an item is determined by k potential features, $k \ll n, m$. Thus each user u can be represented by a vector $\mathbf{u}_u \in \mathbb{R}^k$, and each item i can be represented by a vector $\mathbf{v}_i \in \mathbb{R}^k$. The rating $r_{u,i}$ can then be approximated by the dot product of the corresponding user vector and item vector, i.e., $\hat{r}_{u,i} = \mathbf{u}_u^\top \mathbf{v}_i$. The CF problem can then be represented by the following optimization problem:

$$\arg\min_{U\in\mathbb{R}^{k\times n}, V\in\mathbb{R}^{k\times m}} \sum_{t=1}^{T} \ell\left(\mathbf{u}_u^{(t)}\cdot\mathbf{v}_i^{(t)}, r_{u,i}^{(t)}\right)$$

where the loss function is defined to optimize certain evaluation metric:

$$\ell_{rmse}(\hat{r}_{u,i}, r_{u,i}) = (r_{u,i} - \hat{r}_{u,i})^2 \quad \ell_{mae}(\hat{r}_{u,i}, r_{u,i}) = |r_{u,i} - \hat{r}_{u,i}|$$

The regularized loss at time t is

$$\mathcal{L}_t = \lambda||\mathbf{u}_u^{(t)}||_2^2 + \lambda||\mathbf{v}_i^{(t)}||_2^2 + \ell\left(\mathbf{u}_u^{(t)}\cdot\mathbf{v}_i^{(t)}, r_{u,i}^{(t)}\right)$$

where $\lambda > 0$ is the regularization parameter. A straightforward CF approach is to apply OGD on the regularized loss function [2],

$$\mathbf{u}_u^{(t+1)} = (1 - 2\eta\lambda)\mathbf{u}_u^{(t)} - \eta\frac{\partial\ell\left(\mathbf{u}_u^{(t)}\cdot\mathbf{v}_i^{(t)}, r_{u,i}^{(t)}\right)}{\partial\mathbf{u}_u^{(t)}},$$

$$\mathbf{v}_i^{(t+1)} = (1 - 2\eta\lambda)\mathbf{v}_i^{(t)} - \eta\frac{\partial\ell\left(\mathbf{u}_u^{(t)}\cdot\mathbf{v}_i^{(t)}, r_{u,i}^{(t)}\right)}{\partial\mathbf{v}_i^{(t)}}$$

where $\eta > 0$ is the learning rate. Later, several improved algorithms are proposed, such as Online Multi-Task Collaborative Filtering algorithm [373], Dual-Averaging Online Probabilistic Matrix Factorization algorithm [415], Adaptive Gradient Online Probabilistic Matrix Factorization algorithm [415] and Second-order Online Collaborative Filtering algorithm [261,257]. These algorithms adopt more advanced update strategies beyond OGD and thus can achieve faster adaptation for rapid user preference changes in real-world recommendation tasks.

Besides the algorithms introduced, there are many online CF methods that explore other challenging tasks. First, in many applications, both features of users and items are available and thus need to be considered for better prediction. This generalized CF problem can be solved by using tensor product kernel functions. For example, the Online Low-rank with Features algorithm [2] addresses this problem in online setting. However, it only adopts the linear kernel for efficiency. Perhaps, better performance might be achieved if online budget learning algorithms are adopted. Second, most CF algorithms are based on a regression model, which is mainly concerned with the accuracy of rating prediction, while there are some applications where ranking prediction might be much more important. Two algorithms based on OGD and Dual Averaging approaches are proposed to address this problem by replacing the regression-based loss with the ranking-based loss [251]. Third, for very large-scale applications, when the model has to be learnt using parallel computing, conventional OGD update is not suitable because of the possible conflict in updating the user/item vectors. The Streaming Distributed Stochastic Gradient Descent algorithm [15] provides an operable approach to addresses this problem. Finally, the CF methods for Google News recommendation [104] is a combination of memory-based and model-based algorithms. Also, to address the sparsity problem and imbalance of rating data, [258] incorporate content information via latent dirichlet allocation into online CF, while [241] use a dynamic regularization technique. A closely related modeling approach for recommendation systems is the factorization machine, which has also seen some online variants [250], including being based on deep convolutional networks [240]. There are also efforts in improving the optimization process [201].

4.8. Online learning to rank

Learning to rank is an important family of machine learning techniques for information retrieval [356,61,154,446,331,127]. Different from classification problems where instances are classified as either "relevant" or "not relevant", learning to rank aims to produce a permutation of a group of unseen instances which is similar to the knowledge acquired from the previously seen rankings. To evaluate the performance of ranking algorithms, metrics for information retrieval such as Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG) and Precision-At-Top-k are most popular.

Unlike traditional learning to rank methods which are often based on batch learning [154], we mainly focus on reviewing existing learning to rank methods in online settings [375,366], where instances are observed sequentially. Learning to rank techniques are generally categorized into two approaches: pointwise and pairwise. We will introduce some of the most representative algorithms in each category.

Pointwise Approach: We first introduce a simple Perceptron-based algorithm, the Prank [102,101], which provides a straightforward view of the commonly used problem setting for pointwise learning to rank approaches.

To define the online learning to rank problem setting formally, we have a finite set of ranks $\mathcal{Y} = \{1, \ldots, k\}$ from which a rank $y \in \mathcal{Y}$ is assigned to an instance $\mathbf{x} \in \mathbb{R}^d$. During time t, an instance $\mathbf{x}_t$ is received and the algorithm makes a prediction $\hat{y}_t$ based on the current model $H_t : \mathbb{R}^d \rightarrow \mathcal{Y}$. Then the true rank y_t is revealed and the model is updated based on the loss $\ell(\hat{y}_t, y_t)$. The loss, for instance, can be defined as $\ell(\hat{y}_t, y_t) = |\hat{y}_t, y_t|$. The goal of the online learning to rank task is to minimize the accumulated loss along the whole learning process $\sum_{t=1}^{T} \ell(\hat{y}_t, y_t)$.

The ranking rule of Prank algorithm consists of the combination of Perceptron weight $\mathbf{w} \in \mathbb{R}^d$ and a threshold vector $\mathbf{c} \in \{\mathbb{R}, \infty\}^d$, whose elements are in nondecreasing order i.e., $c^1 \leqslant c^2 \leqslant, \ldots, \leqslant c^k = \infty$. Like the Perceptron algorithm, the rank prediction is determined by the value of the inner product $\mathbf{w}_t^\top \mathbf{x}_t$,

$$\hat{y}_t = \min_{r \in \{1,\ldots,k\}} \{r : \mathbf{w}_t^\top \mathbf{x}_t < c_t^r\}$$

We can expand target rank y_t to vector $\mathbf{y}_t = \{+1, \ldots, +1, -1, \ldots, -1\} \in \mathbb{R}^k$. For $r = 1, \ldots, k$, $y_t^r = -1$ if $y_t < r$, and $y_t^r = 1$ otherwise. Thus, for a correct prediction, $y_t^r(\mathbf{w}_t^\top \mathbf{x}_t - c_t^r) > 0$ holds for all $r \in \mathcal{Y}$. When a mistake appears $\hat{y}_t \neq y_t$, there is subset $\mathcal{M}$ of $\mathcal{Y}$ where $y_t^r(\mathbf{w}_t^\top \mathbf{x}_t - c_t^r) > 0$ does not hold. The update rule is to move the corresponding thresholds for ranks in $\mathcal{M}$ and the weight vector toward each other:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \left(\sum_{r \in \mathcal{M}} y_t^r\right)\mathbf{x}_t, \ \textit{and} \ c_{t+1}^r = c_t^r - y_t^r, \forall r \in \mathcal{M}$$

In theory, the elements in threshold vector $\mathbf{c}$ are always in nondecreasing order and the total number of mistakes made during the learning process is bounded.

Online Aggregate Prank-Bayes Point Machine (OAP-BPM) [162] is an extension of the Prank algorithm by approximating the Bayes point. Specifically, the OAP-BPM algorithm generates N diverse solutions of $\mathbf{w}$ and $\mathbf{c}$ during each iteration and combines them for a better final solution. We denote $H_{j,t}$ as the j-th solution at time t. The algorithm samples N Bernoulli variables $b_{j,t} \in \{0, 1\}, j = \{1, \ldots, N\}$ independently. If $b_{i,t} = 1$, The j-th solution is updated using the Prank algorithm according to the current instance, $H_{j,t+1} = Prank(H_{j,t}, (\mathbf{x}_t, y_t))$. Otherwise, no update is conducted to the j-th solution. The solution $\mathbf{w}_{t+1}$ and $\mathbf{c}_{t+1}$ is the average over N solutions. This work shows better generalization performance than the basic Prank algorithm.

Pairwise Approach: One simple method is to address the ranking problem by transforming it to a classification problem [171]. In a more challenging problem setting, where no accurate rank y is available when collecting the data, only pairwise instances are provided. At time t, a pair of instances $(\mathbf{x}_t^1, \mathbf{x}_t^2)$ are received with the knowledge that $\mathbf{x}_t^1$ is ranked before $\mathbf{x}_t^2$ or the inverse case, and the aim is to find a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ that fits the instance pairs, i.e., $f(\mathbf{x}^1) > f(\mathbf{x}^2)$ when it is known $\mathbf{x}_t^1$ is ranked before $\mathbf{x}_t^2$ or otherwise $f(\mathbf{x}^1) < f(\mathbf{x}^2)$. When the function is linear, the problem can be rewritten as $\mathbf{w}^\top(\mathbf{x}^1 - \mathbf{x}^2) > 0$ when $\mathbf{x}^1$ is in front and otherwise $\mathbf{w}^\top(\mathbf{x}^1 - \mathbf{x}^2) < 0$, where $\mathbf{w} \in \mathbb{R}^d$ is the weight vector. This problem can easily be solved by using a variety of online classification algorithms (Online Gradient Descent [79] for example).

4.9. Distributed online learning

Distributed online learning [425,417,239] has become increasing popular due to the explosion in size and complexity of datasets. Similar to the mini-batch online learning, during each iteration, K instances are received and processed simultaneously. Usually, each node processes one of the instances and updates its local model. These nodes communicate with each other to make their local model consistent. When designing a distributed algorithm, besides computational time cost and accuracy, another important issue to consider is the communication load between nodes. This is because in real world systems with limited network capacity and large communication burden result in long latency.

Based on the network structure, distributed online learning algorithms can be classified into two groups: *centralized* and *decentralized* algorithms. A centralized network is made up of 1 master node and $K - 1$ worker nodes, where the workers can only communicate with the master node. By gathering and distributing information across the network, it is not difficult for distributed algorithms to reach a global consensus [53]. In decentralized networks, however, there is no master and each node can only communicate with its neighbors [6,276]. Although the algorithms are more complex, decentralized learning is more popular because of the robustness of network structure.

We can also group the distributed learning algorithms by *synchronized* and *asynchronized* working modes. Synchronized algorithms are easy to design and enjoy better theoretical bounds but the speed of the whole network is limited by the slowest node. Asynchronized learning algorithms, on the other hand, are complex and usually have worse theoretical bounds. The advantage is its faster processing speed [341].

4.10. Online learning with neural networks

In addition to kernel-based online learning approaches, another rich family of nonlinear online learning algorithms follows the general idea of neural network based learning approaches [393,295,63,222,248,298]. For example, the Perceptron algorithm could be viewed as the simplest form of online learning with neural networks (but it is not nonlinear due to its trivial architecture). Despite many extensive studies for online learning (or incremental learning) with neural networks, many of existing studies in this field fall short due to some critical drawbacks, including the lack of theoretical analysis for performance guarantee, heuristic algorithms without solid justification, and computational too expense to achieve efficient and scalable online learning. Due to the large body of related work, it is impossible to examine every piece of work in this area. In the following, we review several of the most popularly cited related papers and discuss their key ideas for online learning with neural networks.

A series of related work has explored online convex optimization methods for training classical neural network models [48], such as the Multi-layer Perceptron (MLP). For example, online/stochastic gradient descent has been extensively studied for training neural networks in sequential/online learning settings, such as the efficient back-propagation algorithm using SGD [222]. These works are mainly motivated to accelerate the training of batch learning tasks instead of solving online learning tasks directly and seldom give theoretical analysis.

In addition to the above, we also briefly review other highly cited works that address online/incremental learning with neural networks. For example, the study in [393] presented a novel learning algorithm for training fully recurrent neural networks for temporal supervised learning which can continually run over time. However, the work is limited in lacking theoretical analysis and performance guarantee, and the solution could be quite computationally expensive. The work in [295] presented a Resource-Allocating Network (RAN) that learns a two-layer network by a strategy for allocating new units whenever an unusual pattern occurs and a learning rule for refining the network using gradient descent. Although the algorithm was claimed to run in online learning settings, it may suffer poor scalability as the model complexity would grow over time. The study in [63] proposed a new neural network architecture called "ARTMAP" that autonomously learns to classify arbitrarily many, arbitrarily ordered vectors into recognition categories based on predictive success. This supervised learning system was built from a pair of ART modules that are capable of self organizing stable recognition categories in response to arbitrary sequences of input patterns. Although an online learning simulation has been done with ART (adaptive resonance theory), the solution is not optimized directly for online learning tasks and there is also no theoretical analysis. The work in [248] proposed an online sequential extreme learning machine (OS-ELM) which explores an online/sequential learning algorithm for training single hidden layer feedforward networks (SLFNs) with additive or radial basis function (RBF) hidden nodes in a unified framework. The limitation also falls short in some heuristic approaches and lacking theoretical analysis. Last but not least, there are also quite many studies in the field which claim that they design neural network solutions to work online, but essentially they are not truly online learning. They just adapt some batch learning algorithms to work efficiently for seqential learning environments, such as the series of Learning++ algorithms and their variants [298,120].

Recently, Hedge Backpropagation [322] was proposed to learn deep neural networks in the online setting with the aim to address slow convergence of deep networks through dynamic depth adaptation. There have been many other follow up efforts adopting the principle of dynamic architectures for online learning with neural networks [24,300,301,299,25]. Online learning with neural networks has also been extended to several applications including metric learning [138], distributed learning [177], click through rate prediction [213] and continual learning [413] among others. While challenging, there have been efforts in learning Convolutional Networks online as well, which is particularly desired in the presence of concept drifts [112].

Online learning of neural networks has also been suggested as a potential approach to also train neural networks in an energy efficient manner [384]. [7] extend the setting to do online learning with neural networks based on the hedge backpropagation principle with inconsistently available inputs.

4.11. Online Portfolio Selection

On-line Portfolio Selection (OLPS) is a natural application of online learning for sequential decisions of selecting a portfolio of stocks for optimizing certain metrics, e.g. cumulative wealth, risk adjusted returns, etc. [73,226,231,232]. Consider a financial market with m assets, in which we have to allocate our wealth. At every time period (or iteration), the price of the m stocks changes by a factor of $\mathbf{x}_t \in \mathbb{R}_+^m$. This vector is also called the price relative vector. $x_{t,i}$ denotes the ratio of the closing price of asset i at time t to the last closing price at time $t-1$. Thus, an investment in asset i changes by a factor of $x_{t,i}$ in period t. At the beginning of time period t the investment is specified by a portfolio vector $\mathbf{b}_t \in \delta_m$ where $\delta_m = \mathbf{b} : \mathbf{b} \succeq 0, \mathbf{b}^\top \mathbf{1} = 1$. The portfolio is updated in every time-period based on a specific strategy, and produces a sequence of mappings:

$$\mathbf{b}_1 = \frac{1}{m}, \quad \mathbf{b}_t : \mathbb{R}_+^{m(t-1)} \to \delta_m, \quad t = 2, 3, \ldots, T$$

where T is the maximum length of the investment horizon. To make a decision for constructing a portfolio at time t, the entire historical information from $\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}$ is available. The theoretical framework starts with a wealth of $S_0 = 1$, and at the end of every time period, the wealth changes as $S_t = S_{t-1} \times \left(\mathbf{b}_t^\top \mathbf{x}_t\right)$.

Most efforts in OLPS make a few (possibly unrealistic) assumptions, including no transaction costs, perfectly liquid market, and no impact cost (the portfolio selection strategy does not affect the market). Besides the traditional benchmarking approaches, the approaches for OLPS can be categorized into *Follow-the-winner, Follow-the-loser, Pattern Matching,* and *Meta-Learning* approaches [226].

The benchmark approaches, as the name suggests, are simple baseline methods whose performance can be used to benchmark the performance of proposed algorithms. Common baselines are Buy and Hold (BAH) strategy, Best Stock and Constant Rebalanced Portfolio (CRP). The idea of BAH is to start with a portfolio with equal investment in each asset, and never rebalance it. Best Stock is the performance of the asset with the highest returns at the end of the investment horizon. CRP [200] is a fixed portfolio allocation to which the portfolio is rebalanced to at the end of every period, and Best-CRP is the CRP which obtains the highest returns at the end of the investment horizon. It should be noted that Best Stock and Best-CRP strategies can only be executed in hindsight.

Follow-the-winner approaches adhere to the principle of increasing the relative portfolio allocation weight of the stocks that have performed well in the past. Many of the approaches are directly inspired by Convex Optimization theory, including Universal Portfolios [94], Exponential Gradient [170], Follow the Leader [133] and Follow the Regularized Leader[5]. In contrast to follow-the-winner, there is a set of approaches that aim to follow-the-loser, with the belief that asset prices have a tendency to revert back to a mean, i.e., if the asset price falls, it is likely to rise up in the next time-period. These are also called mean-reversion strategies. The early efforts in this category included Anti Correlation [47] which designed a strategy by betting making statistical bets on positive-lagged correlation and negative auto-correlation; and Passive-Aggressive Mean Reversion (PAMR) [234], which extended the Online Passive Aggressive Algorithms [95] to update the portfolio to an optimal "loser" portfolio - by selecting a portfolio that would have made an optimal loss in the last observed time-period. A similar idea was used to extend confidence-weighted online learning to develop Confidence-Weighted Mean Reversion [229,230]. The idea of PAMR was extended to consider multi-period asset returns, which led to the development of Online Moving Average Reversion (OLMAR) [225,228] and Robust Median Reversion (RMR) [179] strategies.

Another popular set of approaches is the Pattern-Matching approaches, which aim to find patterns (they may be able to exploit both follow-the-winner and follow-the-loser) for optimal sequential decision making. Most of these approaches are non-parametric. Exemplar approaches include [151,152,227]. Finally, meta-learning algorithms for portfolio selection aim to rebalance the portfolio on the basis of the expert advice. There are a set of experts that output a portfolio vector, and the meta-learner uses this information to obtain the optimal portfolio. In general, the meta-learner adheres to the follow-the-winner principle to identify the best performing experts. Popular approaches in this category include Aggregating Algorithms [365], Fast Universalization Algorithm [13] and Follow the Leading History [166]. Besides these approaches, there are also efforts in portfolio selection with aims to optimize the returns accounting for transaction costs. The idea is to incorporate the given transaction cost into the optimization objective [291,180,233].

A closely related area to Online Portfolio Selection is Online Learning for Time Series Prediction. Time series analysis and prediction [142,92,81,325] is a classical problem in machine learning, statistics, and data mining. The typical problem setting of time series prediction is as follows: a learner receives a temporal sequence of observations, $x_1, \ldots, x_t$, and the goal of the learner is to predict the future observations (e.g., x_{t+1} or onwards) as accurately as possible. In general, machine learning methods for time series prediction may also be divided into linear and non-linear, univariate and multivariate, and batch and online. Some time series prediction tasks may be resolved by adapting an existing batch learning algorithm using sliding window strategies. Recently there have been some emerging studies for exploring online learning algorithms for time series prediction [18,256].

5. Bandit online learning

5.1. Overview

Bandit online learning, a.k.a. the "Multi-armed Bandit (MAB) problem [310,197,364,55,144], is an important branch of online learning where a learner makes sequential decisions by receiving only partial feedback from the environment each time.

MAB problems are online learning tasks for sequential decisions with a trade-off between exploration and exploitation. Specifically, on each round, a player chooses one out of K actions, the environment then reveals the payoff of the player's action, and the goal of the learner is to maximize the total payoff obtained during online learning process. A fundamental challenge of MAB is to address the exploration-exploitation tradeoff [27], i.e., balancing between the *exploitation* of actions that gave highest payoffs in the past and the *exploration* of new actions that might give higher payoffs in the future.

MAB problems are collectively called "bandit" problems. Historically, the name "bandit" is referred to the scenario of playing slot machines in a casino, where a player faces a number of slot machines to insert coins (one slot machine is called one-armed bandit in American slang), the expected reward of each machine might be different, and the player's goal is to maximize the reward by repeatedly choosing where to insert the next coin.

MAB problems can be roughly divided into two major categories: *stochastic MAB* and *adversarial MAB*. The former assumes a stochastic environment where rewards (or "losses" equivalently) are i.i.d. and independent from each other, while the later removes the stochastic assumption where the rewards (or losses) can be arbitrary or more formally "adversarial" by adapting to the past decisions. Note that the notion of "reward" and "loss" are symmetric and can be translated from one to the other equivalently. To be consistent, if not mentioned specifically, we will use "loss" instead of "reward" for the rest discussion.

We now introduce the formal procedure of the MAB problem. Formally, a K-armed bandit problem takes place in a sequence of rounds with length $T \in \mathcal{N}$, where T is often unknown at the beginning (typically a learner is called an "anytime" algorithm if T is unknown in advance). At the t-th round, the player chooses one out of K actions $I_t \in [K] = \{1, \ldots, K\}$ using some strategy. After that, the environment reveals the loss $\ell_t(I_t)$ of the action to the forecaster. The goal is to minimize the total loss over the T rounds. In theory, we are interested in analyzing the behavior of the learner, typically by comparing the performance of its actions against with some optimal strategy. More formally, we can define the "regret" as the difference between the cumulative loss of the best fixed arm by an optimal strategy and that of the player after playing T rounds $I_1, \ldots, I_T$ as

$$R_T = \sum_{t=1}^{T} \ell_t(I_t) - \min_{i \in [K]} \sum_{t=1}^{T} \ell_t(i)$$

As both loss $\ell_t(i)$ and action I_t could be stochastic, we can define the expected regret as

$$\mathbb{E}[R_T] = \mathbb{E}\left[\sum_{t=1}^{T} \ell_t(I_t) - \min_{i \in [K]} \sum_{t=1}^{T} \ell_t(i)\right]$$

where the expectation is taken with respect to the random draw of both losses and learner's actions.

5.2. Stochastic bandits

For stochastic MAB problem, each arm $i \in [k]$ corresponds to an unknown distribution P_i on $[0, 1]$, and the losses $\ell_t(i)$ are independently drawn from the distribution P_i corresponding to the selected arm. Let us denote by μ_i the mean of the distribution P_i, and define

$$\mu^* = \min_{i \in [k]} \mu_i \quad \text{and} \quad i^* \in \arg\min_{i \in [k]} \mu_i$$

The expected regret can be rewritten as

$$\mathbb{E}[R_T] = \mathbb{E}\left[\sum_{t=1}^{T} \mu_{I_t}\right] - T \min_{i \in [K]} \mu_i = \mathbb{E}\left[\sum_{t=1}^{T} \mu_{I_t}\right] - T\mu^* = \mathbb{E}\left[\sum_{t=1}^{T} \left(\mu_{I_t} - \mu_*\right)\right]$$

Further let $N_i(s) = \sum_{t=1}^{s} \mathbb{I}(I_t = i)$ denote the number of times the player selected arm i on the first s rounds and $\Delta_i = \mu_i - \mu_*$ be the suboptimality parameter of arm i, we can simplify the expected regret as follows:

$$\mathbb{E}[R_T] = \mathbb{E}\left[\sum_{t=1}^{T} \Delta_{I_t}\right] = \sum_{i=1}^{K} \Delta_i \mathbb{E}[N_i(T)].$$

5.2.1. Stochastic multi-armed bandit

In this section we mainly introduce two well-known algorithms for stochastic MAB.

ϵ-Greedy. The first simplest algorithm for stochastic MAB is called the ϵ-Greedy rule [348]. The idea is to with probability $1 - \epsilon$ play the the current best arm of the highest average reward, and with probability ϵ play a random arm, where parameter $\epsilon > 0$ is a constant value in $(0, 1)$. Algorithm 13 gives a summary of this algorithm.

Algorithm 13: ϵ-Greedy

```
INPUT: parameter $\epsilon > 0$
INIT: empirical means $\mu_i = 0, \forall i \in [K]$
for $t = 1, 2, \ldots, T$ do
    with probability $1 - \epsilon$ play the current best arm
    $i_t = \arg\min_{i \in [K]} \mu_i$
    with probability $\epsilon$ play a random arm
    receive $\ell_t(i_t)$
    update the empirical means
    $\mu_{i_t} = \left(\mu_{i_t} + \ell_t(i_t)\right) / (N_{i_t}(t) + 1)$
end for
```

However, the constant exploration probability ϵ results in a linear growth in the regret. One way to fix it is to decrease the value of ϵ over time and let it go to zero at a certain rate. For example, an improved ϵ_t-greedy algorithm is to follow the epsilon-decreasing strategy by defining ϵ_t at round t as

$$\epsilon_t = \min\left\{1, \frac{cK}{d^2 t}\right\}$$

where $c > 0$ and $d \in (0,1)$. When $0 < d \leqslant \min_{i:\mu_i<\mu^*}\Delta_i < 1$ and $T > \frac{cK}{d}$, this improved ϵ_t-greedy algorithm can achieve the logarithm regret $O(\ln T)$.

UCB. Another well-known algorithm for stochastic MAB is the Upper Confidence Bound (UCB) algorithm [29], a strategy that simultaneously performs exploration and exploitation using a heuristic principle of *optimism in face of uncertainty*. The intuition is that, despite lacking knowledge about what actions are best, we will try to construct an optimistic guess as to how good the expected payoff/loss of each action is, and choose the action with the best guess. If our guess is correct, we will be able to exploit that action and incur little regret; but if our guess is wrong, then our optimistic guess will quickly decrease and we will then be compelled to switch to a different action, which therefore is able to balance the exploration-exploitation tradeoff.

Formally, the "optimism" comes in the form of Upper Confidence Bound (UCB). In particular, the idea is to calculate the confidence intervals of the averages, which is a region around our estimates such that the true value falls within with high probability, and repeatedly shrink the confidence bounds such that the average will become more reliable. Algorithm 14 gives a summary of the UCB algorithm.

Algorithm 14: UCB

```
INPUT: parameter $\epsilon > 0$
INIT: empirical means $\mu_i = 0, \forall i \in [K]$
for $t = 1, 2, \ldots, T$ do
    play the arm $i_t = \arg\min_i\left(\mu_i - \sqrt{\frac{2\ln t}{N_i(t)}}\right)$
    receive $\ell_t(i_t)$
    update the empirical means
    $\mu_{i_t} = \left(\mu_{i_t} + \ell_t(i_t)\right)/(N_{i_t}(t) + 1)$
end for
```

In theory, by running the UCB algorithm over T rounds, the expected regret is

$$\mathbb{E}[R_T] \leqslant 8\sum_{i:\mu_i<\mu^*}\left(\frac{\ln T}{\Delta_i}\right) + \left(1 + \frac{\pi^2}{3}\right)\left(\sum_{j=1}^{K}\Delta_j\right)$$

The above is a specific worst-case bound on the expected regret [29]. More concisely, one can show that the expected regret of UCB is at most $O\left(\sqrt{KT\ln T}\right)$. [29] also gave some variants of improved UCB algorithms. Improved algorithms and regret bound were also given in [1].

5.2.2. Bayesian bandits

Bayesian methods have been explored for studying bandit problems from the beginning of this field. One of the most well-known and classic algorithm in Bayesian Bandits is Thompson Sampling [354], which is considered one of the oldest algorithm to address the exploration-exploitation trade-off for bandit problems. Recent years have seen a lot of interests in analyzing both empirical performance [80] and theoretical properties of Thompson Sampling for bandit problems [11]. In the following, we introduce a Bayesian setting of Bernoulli bandit and then discusses the Thompson Sampling algorithm.

Consider a standard K-armed Bernoulli bandit, each action corresponds to the choice of an arm, and the reward of the i-th arm is either 0 or 1 which follows a Bernoulli distribution with mean μ_i, i.e., the probability of success for arm i (reward=1) is μ_i. The algorithm maintains Bayesian priors on the Bernoulli means μ_i's. At the beginning, the Thompson Sampling algorithm initializes each arm i to have prior $Beta(1,1)$ on μ_i, since $Beta(1,1)$ is the uniform distribution on $(0,1)$. On round t, after observing $S_i(t)$ successes (reward=1) and $F_i(t)$ failures (reward=0) on $k_i(t) = S_i(t) + F_i(t)$ times of playing arm i, the algorithm updates the distribution on θ_i as $Beta(S_i(t)+1, F_i(t)+1)$. The algorithm then samples from these posterior distributions of the θ_i's, and plays an arm according to the probability of its mean being the largest. Algorithm 15 gives a summary of Thompson Sampling algorithm for K-armed Bernoulli bandits problems.

Algorithm 15: Thompson Sampling

```
INIT: $S_i(1) = 0, F_i(1) = 0, \forall i = 1, \ldots, K$
for $t = 1, 2, \ldots, T$ do
    For each arm $i \in [K]$, sample $\theta_i(t)$ from the
    $Beta(S_i + 1, F_i + 1)$ distribution.
    Play arm $i(t) = \arg\max_i \theta_i(t)$, and observe reward $r_t$
    If $r_t = 1$, then $S_{i(t)} = S_{i(t)} + 1$; else $F_{i(t)} = F_{i(t)} + 1$.
end for
```

The above Thompson sampling algorithm can be easily extended to the general stochastic bandits setting, i.e., when the rewards for arm i are generated from an arbitrary unknown distribution with support $[0,1]$ and mean μ_i. It has also been extensively used for contextual bandit settings [80,12].

In theory, for the K-armed stochastic bandit problem, denoting $\Delta_i = \mu_1 - \mu_i$, the Thompson Sampling algorithm has the expected regret given in [11]

$$\mathbb{E}[R_T] \leqslant O\left(\left(\sum_{a=2}^{K}\frac{1}{\Delta_a^2}\right)^2 \ln T\right)$$

Other Related Work. In addition to the classic Thompson Sampling algorithm, another notable variant of Bayesian Bandit is called the Bayes-UCB algorithm [199], which is a UCB-like algorithm, where the upper confidence bounds are based on the quantiles of Beta posterior distributions, and is able to achieve the lower bound of [216] for Bernoulli rewards. More other extensive and recent studies of Bayesian Bandits can be found in [329,271,317].

5.3. Adversarial bandits

In the previous setting of stochastic bandits, we generally assume that the rewards are i.i.d., which are drawn independently from some unknown but fixed distribution. We now relax such stochastic assumption on the rewards. We assume the reward distribution can be affected by the previous actions taken by the player, which is termed as "Adversarial Bandits" problems [30]. In the following, we review several classes of adversarial bandits, including some fundamentals of adversarial MAB and other active topics such as linear bandits and combinatorial bandits.

5.3.1. Adversarial multi-armed bandit

We consider a K-armed adversarial bandit problem where $K > 1$ and the learner receives an arbitrary sequence of loss vectors $(\ell_1, \ldots, \ell_T)$ where $\ell_t \in [0,1]^K \forall t \in [T]$. On each round, the learner plays an action $I_t \in [K]$ and observes the loss $\ell_t(I_t)$. For adversarial bandits, a randomized policy is commonly used. In particular, given some policy π, the conditional distribution over the actions having observed $\Omega_{t-1} = \{(I_1,\ell_1), \ldots, (I_{t-1},\ell_{t-1})\}$ is $P_t = \pi(\cdot|\Omega_{t-1}) \in \mathcal{P}_{K-1}$. The performance of a policy π on the environment can be measured by the expected regret which is the expected loss of the policy relative to the best fixed action in hindsight:

$$\mathbb{E}[R_T] = \mathbb{E}\left[\sum_{t=1}^{T}\ell_t(I_t)\right] - \min_{i\in[K]}\sum_{t=1}^{T}\ell_t(i)$$

Exp3. The Exponential-weights for Exploration and Exploitation algorithm (Exp3) [31] is a popular algorithm for adversarial MAB. It follows the similar idea of prediction with expert advice and applies the Hedge (or Weighted-Majority) algorithm to the tradeoff of exploration and exploitation. Specifically, we first define a probability vector $\mathbf{p_t} \in \mathbb{R}^{\mathbf{K}}$ in which the i-th element $p_t(i)$ indicates the probability of drawing arm i at time t. This vector is initialized uniformly and updated at each round. On each round, the learner plays an action by drawing $I_t \sim \mathbf{p_t}$ where p_t is set as follows

$$p_t(i) = (1-\gamma)\frac{w_t(i)}{\sum_{j=1}^{K}w_t(j)} + \frac{\gamma}{K}, \forall i \in [K]$$

where $w_t(i)$ is the importance weight of each arm i learned by the Hedge algorithm, and γ is a parameter for weighting the exploration term.

Algorithm 16 gives a summary of the Exp3 algorithm. By tuning the optimal parameter of γ, the Exp3 algorithm is able to achieve the regret $O\left(\sqrt{TK\ln K}\right)$ in the adversarial setting.

Algorithm 16: Exp3

```
INPUT: parameter $\gamma \in (0,1]$
INIT: $w_1(i) = 1, \forall i \in [K]$
for $t = 1,2,\ldots,T$ do
    Set $p_t(i) = (1-\gamma)\frac{w_t(i)}{\sum_{j=1}^{K}w_t(j)} + \frac{\gamma}{K}, \forall i \in [K]$
    Play action by drawing $I_t \sim \mathbf{p_t}$
    Receive $\ell_t(I_t) \in [0,1]$,
    Update $w_t(i) = w_t(i)e^{-\gamma\frac{\ell_t(i)}{p_t(i)}}$, if $i = I_t$.
end for
```

Other Related Works. This is generally more challenging than the stochastic setting. A variety of algorithms have been explored in literature [55]. For example, the Exp3.P algorithm in [31] improves the loss estimation and probability update strategies to get a high probability bound. The Exp3.M algorithm in [360] explores the new problem setting of multiple plays.

5.3.2. Linear bandit and combinatorial bandit

We first introduce the problem setting of the *Linear Bandit* optimization problem [28,316,190]. During each iteration, the player makes its decision by choosing a vector from a finite set $\mathcal{S} \subseteq \mathbb{R}^{d}$ of elements $\mathbf{v(i)}$ for $i = 1,\ldots,k$. The chosen action at iteration t is indexed as I_t. The environment chooses a loss vector $\boldsymbol{\ell}_t \in \mathbb{R}^d$ and returns the linear loss as $c_t(I_t) = \boldsymbol{\ell}_t^\top \mathbf{v}(\mathbf{I_t})$. Note that the player has no access to the full knowledge of loss vector and the only information revealed to the player is the loss of its own decision $c_t(I_t)$. Obviously, when setting $d = k$ and $\mathbf{v(i)}$ is the standard basis vector, this problem is identical to that in the previous section.

Combinatorial bandit [74] is a special case of Linear Bandits, where $\mathcal{S}$ is a subset of binary hypercube $\{0,1\}^d$. The loss vector ℓ_t may be generated from an unknown but fixed distribution, which is termed as stochastic combinatorial bandit, or chosen from some adversarial environment, which is termed as adversarial combinatorial bandit. The goal is to minimize the expected regret

$$\overline{R}_T = \mathbb{E}\left[\sum_{t=1}^{T}c_t(I_t)\right] - \min_{i\in[K]} L_T(i)$$

where $L_T(i) = \mathbb{E}\sum_{t=1}^{T}c_t(i)$ is the expected sum of loss for choosing action i in all T iterations, not a random variable in stochastic setting. One example algorithm for Combinatorial Bandit is the COMBAND algorithm [74]. It first defines a sampling probability vector $\mathbf{p_t} \in \mathbb{R}^{\mathbf{k}}$ for sampling $\mathbf{v}(\mathbf{I_t})$ from $\mathcal{S}$

$$\mathbf{p_t} = (\mathbf{1}-\gamma)\mathbf{q_t} + \gamma\boldsymbol{\mu}$$

where $\mathbf{q_t} \in \mathbb{R}^{\mathbf{k}}$ is the exploitation probability vector that is updated during all iterations to follow the best action, $\boldsymbol{\mu} \in \mathbb{R}^d$ is a fixed exploration probability, and $\gamma \in [0,1]$ is the weight to control the exploitation and exploration trade-off. The algorithm draws the action I_t based on distribution $\mathbf{p_t}$ and gets the loss $c_t(I_t)$ from the environment. Second, an estimation of the loss vector ℓ_t is calculated with the new information,

$$\widetilde{\ell}_t = c_t(I_t)P_t^{+}\mathbf{v}(\mathbf{I_t})$$

where P_t^{+} is the pseudo-inverse of the expected correlation matrix $\mathbb{E}_{\mathbf{p_t}}[\mathbf{v}\mathbf{v}^\top]$. Finally, the exploitation weights are scaled based on the estimated loss vector,

$$\mathbf{q_{t+1}}(\mathbf{i}) \propto \mathbf{q_t}(\mathbf{i})\exp\left(-\eta\widetilde{\ell}_{\mathbf{t}}^{\top}\mathbf{v}(\mathbf{i})\right)$$

where $\eta > 0$ is a learning rate parameter and $\propto$ indicates that this scaling step is followed by a normalization step so that $\sum_{i=1}^{k}\mathbf{q(i)} = \mathbf{1}$.

The COMBAND algorithm achieves a regret bound better than $O\left(\sqrt{Td\ln|\mathcal{S}|}\right)$ for a variety of concrete choices of $\mathcal{S}$.

Other Related Works. Recently, many studies also address linear bandits and combinatorial bandits in different settings. The ESCB algorithm [93] efficiently exploits the structure of the problem and gets a better regret bound of $O(\ln(T))$. The CUCB algorithm [86] addresses the problem where the loss may be nonlinear. [93] provided a useful survey for closely related works [56,74] and gave a novel algorithm with promising bounds.

5.4. Contextual bandits

Contextual Bandit is a widely used extension of MAB by associating contextual information with each arm [416,243]. For example, in personalized recommendation problem, the task is to select products that are most likely to be purchased by a user. In this case, each product corresponds to an arm and the features of each product are easy to acquire [242].

In a contextual bandits problem, there is a set of policies $\mathcal{F}$, which may be finite or infinite. Each $f \in \mathcal{F}$ maps a context $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{d}$ to an arm $i \in [k]$. Different from the previous setting where the regret is defined by competing with the arm with the highest expected reward, the regret here is defined by comparing the decision I_t with the best policy $f^* = \arg\inf_{f\in\mathcal{F}}\ell_D(f)$, where D is the data distribution.

$$R_T(f) = \sum_{t=1}^{T}[\ell_{I_t,t} - \ell_t(f^*)]$$

In literature, there are comprehensive surveys on contextual bandit algorithms in both stochastic and adversarial settings [441,55]. Below we focus on two settings of contextual bandits: multiclass classification and expert advice.

5.4.1. The multiclass setting

In this setting, contextual bandit is regarded as a special case of online multi-class classification tasks in bandit setting. The goal is to learn a mapping from context space $\mathbb{R}^d$ to label space $\{1,\ldots,k\}$

from a sequence of instances $\mathbf{x}_t \in \mathbb{R}^d$. Different from classic online multi-class classification problems where a class label $y_t \in \{1, \ldots, k\}$ is revealed at the end of each iteration, in bandit setting, the learner only gets a partial feedback on whether $\hat{y}_t$ equals to y_t. In the following, we briefly review some representative works of contextual bandits for multi-class classification.

Bnditron is the first bandit algorithm for online multiclass prediction [192], which is a variant of the Perceptron. To efficiently make prediction and update the model, the Banditron algorithm keep a linear model W^t, which is initialized as $W^1 = 0 \in \mathbb{R}^{k\times d}$. At the t-th iteration, after receiving the instance $\mathbf{x}_t \in \mathbb{R}^d$, it will first set

$$\hat{y}_t = \arg\max_{r\in[k]} (W^t \mathbf{x}_t)_r$$

where $(\mathbf{z})_r$ denotes the r-th element of $\mathbf{z}$. Then the algorithm will define a distribution as

$$\Pr(r) = (1-\gamma)\mathbb{I}(r = \hat{y}_t) + \gamma/k, \forall r \in [k]$$

which roughly implies that the algorithm exploits with probability $1-\gamma$ and explores with the remaining probability by uniformly predicting a random label from $[k]$. The parameter γ controls the exploration-exploitation tradeoff. The algorithm then randomly sample $\tilde{y}_t$ according to the probability Pr and predicts it as the label of $\mathbf{x}_t$. After the prediction, the algorithm then receives the bandit feedback $\mathbb{I}(\tilde{y}_t = y_t)$. Then the algorithm uses this feedback to construct a matrix

$$\widetilde{U}^t_{r,j} = x_{t,j}\left(\mathbb{I}(\hat{y}_t = r) - \frac{\mathbb{I}(\tilde{y}_t = y_t)\mathbb{I}(\tilde{y}_t = r)}{\Pr(r)}\right)$$

since its expectation satisfies

$$\mathbb{E}\widetilde{U}^t_{r,j} = U^t_{r,j} = x_{t,j}(\mathbb{I}(\hat{y}_t = r) - \mathbb{I}(y_t = r))$$

where U^t is actually a (sub)-gradient of the following hinge loss

$$\ell(W; (\mathbf{x}_t, y_t)) = \max_{r\in[k]/\ \{y_t\}} \left[1 - (W\mathbf{x}_t)_{y_t} + (W\mathbf{x}_t)_r\right]_+$$

where $[z]_+ = \max(0, z)$. Then the algorithm will update the model by $W^{t+1} = W^t - \widetilde{U}^t$. The Banditron algorithm is summarized in Algorithm 17.

Algorithm 17: Banditron

```
**INIT**: $\mathbf{w}_{1,1} = 0, .., \mathbf{w}_{k,1} = 0$
**for** $t = 1, 2, \ldots, T$ **do**
  Receive an incoming instance $\mathbf{x}_t$
  $P(r) = (1-\gamma)1\left[r = \arg\max_i \mathbf{w}_{i,t}^\top \mathbf{x}_t\right] + \frac{\gamma}{k}$.
  Sample $\hat{y}_t$ according to $P(r), r \in \{1, \ldots, k\}$
  $\mathbf{u}_r = \mathbf{x}_t\left(\frac{1[y_t = \hat{y}_t = r]}{P(r)} - 1\left[r = \arg\max_i \mathbf{w}_{i,t}^\top \mathbf{x}_t\right]\right)$;
  $\mathbf{w}_{r,t+1} = \mathbf{w}_{r,t} + \mathbf{u}_r$
**end for**
```

This algorithm achieves $O\left(\sqrt{T}\right)$ in linear separable case and $O\left(T^{\frac{2}{3}}\right)$ in inseparable case.

Other Related Work. Following the Banditron, many algorithms have been proposed. For example, Bandit Passive Aggressive (Bandit PA) follows the PA learning principle and adopts the framework of one against all others to make prediction and update the model [83]. In general, some update principles are based on first-order gradient descent [381], while others adopt second order learning [97,164,422,40]. Most of these algorithms explore the k classes uniformly with probability γ, while [97] sample the classes based on the Upper Confidence Bound.

5.4.2. The expert setting

We now introduce a well-known algorithm called Exp4 [31] for contextual bandits in the expert settings. The Exp4 algorithm assumes that there are N experts who will give advice on the distribution over arms during all iterations. $\xi^n_{i,t}$ indicates the probability of picking arm $i \in [K]$ recommended by expert $n \in [N]$ during time $t \in [T]$. Obviously, $\sum_{i=1}^k \xi^n_{i,t} = 1$. During time t, the true reward vector is denoted by $\mathbf{r_t} \in [\mathbf{0}, \mathbf{1}]^{\mathbf{K}}$. Thus the expected reward of expert n is $\xi^n_t \cdot \mathbf{r_t}$. The regret is defined by comparing with the expert with the highest expected cumulative reward.

$$R_t = \max_{n\in[N]} \sum_{t=1}^{T} \xi^n_t \cdot \mathbf{r_t} - \mathbb{E}\sum_{\mathbf{t=1}}^{\mathbf{T}} \mathbf{r_{t,I_t}}$$

The Exp4 algorithm first defines a weight vector $\mathbf{w}_t \in \mathbb{R}^N$ that indicates the weights for the N experts. We set the weight as $\mathbf{w}_0 = 1$ and update it during each iteration. During iteration t, we calculate the probability of picking arm i as the weighted sum of advices from all N experts,

$$p_{i,t} = (1-\gamma)\frac{\sum_{n=1}^N w_{n,t}\xi^n_{i,t}}{\sum_{n=1}^N w_{n,t}} + \frac{\gamma}{K}$$

where $\gamma \in [0, 1]$ is a parameter to balance exploitation and exploration. We then draw an arm I_t according to probability $p_{i,t}$ and calculate an unbiased estimator of $\hat{r}_{i,t} = \frac{r_{i,t}}{p_{i,t}}\mathbb{I}_{i=I_t}$, which will be used to calculate the expected reward. Finally the weight $\mathbf{w}_t$ is updated according to the expected reward of each arm. The Exp4 algorithm is able to achieve the regret bound $O\left(\sqrt{TK\ln N}\right)$ as shown in [31] and tighter bounds were also given in [272].

Other Related Work. Another general contextual bandit algorithm is the epoch-greedy algorithm in [219] that is similar to ϵ-greedy with shrinking ϵ. This algorithm is computationally efficient given an oracle optimizer but has the weaker regret guarantee of $O\left(T^{2/3}\right)$. LinUCB [91] is an extension of UCB to contextual bandit problem, by assuming that there is a feature vector $\mathbf{x}_{t,i} \in \mathbb{R}^d$ at time t for each arm i. Similar to the UCB algorithm, a model is learnt to estimate the upper confidence bound of each arm $i \in [k]$ given the input of $\mathbf{x}_{t,i}$. The algorithm simply chooses the arm with the highest UCB. The LinREL algorithm [28] is similar to LinUCB in that it adopts the same problem setting and same maximizing UCB strategy. While, a different regularization term is used which leads to a different calculation of the UCB.

5.5. Other bandit variants

In literature, there are many other studies addressing on various types of bandit variants. We refer readers for more comprehensive studies on bandit topics in [55]. Below we briefly introduce a few other major variants.

Other than stochastic bandits and adversarial bandits, another fundamental topic of multi-armed bandits is called "*Markovian bandits*", which generally assumes the reward processes are neither i.i.d. (like in stochastic MAB) nor adversarial. Specifically, arms are associated with K Markov processes, each with its own state space. On each round, an arm is chosen in some state, a stochastic reward is drawn from some probability distribution, and the state of the reward process for the arm changes in a Markovian fashion, based on an underlying stochastic transition matrix. Both reward and new state are revealed to the player. The seminal work of [145] gives an optimal greedy policy that can be computed efficiently. A special case of Markovian bandits is Bayesian bandits [329,144,199], which are parametric stochastic bandits where the parameters of the reward distributions are assumed to be drawn from known priors, and the regret is computed by also averaging over the draw of parameters from the prior.

Another topic is to study *infinitely many-armed bandits* problems where the number of arms can be larger than the possible number of experiments or even infinite [39,383,57]. Among these studies, one niche sub-topic is *continuum-armed bandits* [204,90], where the arms lie in some Euclidean (or metric) space and their mean-reward is a deterministic and smooth (e.g., Lipschitz) function of the arms, a.k.a. *Lipschitz Bandit* [269]. Moreover, bandit online learning has several other applications including graph classification [410], and change detection [418].

6. Online active learning

6.1. Overview

In a standard online learning task (e.g., online binary classification), the learner receives and makes prediction for a sequence of instances generated from some unknown distribution. At the end of every round, it always assumes the learner will receive the true label (feedback) from the environment. For many real-world applications, obtaining the labels could be very expensive, and sometimes it is not always necessary/informative to query the true labels of every instance, e.g., if an instance is correctly classified with a high confidence. Motivated to address this challenge, online active learning is a special class of online learner that observes a sequence of unlabeled instances each time deciding whether to query the label of the incoming instance; if the label is queried, then the learner can use the labelled instance to update the prediction model; otherwise, the model will be kept unchanged.

In literature, there are two major kinds of settings for online active learning. One is called the "*selective sampling*" setting [26,131,67] by adapting classical online learning for active learning. The other is *online active learning with expert advice* by adapting the setting of prediction with expert advice for active learning [169,434]. Both operate in the similar problem settings where true label of an instance is only queried when some condition is satisfied, e.g., predictive confidence is below some threshold.

6.2. Selective sampling algorithms

In this section we review a family of popular Selective Sampling (SS) algorithms for online active learning tasks. In the following discussions, we use a typical online binary classification task as a running example. For notation, an example is a pair $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathbb{R}^d$ is an instance vector and $y \in \{-1, +1\}$ is the binary class label. Assume the learning proceeds in a sequence of T rounds, where T may not be known in advance. On each round t, a learner observes an instance $\mathbf{x}_t$, then outputs a prediction $\hat{y}_t \in \{-1, +1\}$ as the label for the instance, and then decides whether or not to query the label y_t. Whenever $\hat{y}_t \neq y_t$, the learner's prediction outcome is considered as a mistake, no matter if it has decided to query the label or not. For notation, we denote $M_t = \mathbb{I}(\hat{y}_t \neq y_t) \in \{0, 1\}$ as an indicator whether the learner makes a mistake at round t. For most cases, we also assume the leaner adopts a linear model to predict the class label using $\hat{y}_t = \text{sign}(\hat{p}_t)$, where $\hat{p}_t = \mathbf{w}_t^\top \mathbf{x}_t$.

6.2.1. First-order selective sampling algorithms

Selective-sampling Perceptron. This algorithm [72] decides whether or not to query the label y_t through a simple randomized rule: drawing a Bernoulli random variable $Z_t \in \{0, 1\}$ with probability

Algorithm 18: Selective-sampling Perceptron

```
INPUT: parameter $\delta > 0$
INIT: $\mathbf{w}_0 = (0, \dots, 0)^\top$
for $t = 1, 2, \dots, T$ do
  Observe an input instance $\mathbf{x}_t \in \mathbb{R}^d$
  Predict $\hat{y}_t = \text{sign}(\hat{p}_t)$, where $\hat{p}_t = \mathbf{w}_t^\top \mathbf{x}_t$
  Draw a Bernoulli random variable $Z_t \in \{0, 1\}$ of probability $\frac{\delta}{\delta + |\hat{p}_t|}$
  IF $Z_t = 1$ THEN
    Query label $y_t \in \{-1, +1\}$ and Update $\mathbf{w}_t$ by
  Perceptron: $\mathbf{w}_{t+1} = \mathbf{w}_t + M_t y_t \mathbf{x}_t$.
end for
```

$$\Pr(Z_t = 1) = \frac{\delta}{\delta + |\hat{p}_t|}$$

where $\delta > 0$ is a smooth parameter that can be used to control the number of labels queried during the online active learning process. If δ increases, the number of queried labels increases. If $Z_t = 1$, then the label y_t of $\mathbf{x}_t$ will be queried, and the model will be updated using the Perceptron rule. Algorithm 18 gives a summary of the Selective-sampling Perceptron algorithm.

In theory, assuming $\|\mathbf{x}_t\| \leqslant R$, for any $\mathbf{w} \in \mathbb{R}^d$, the expected number of mistakes of the Selective-sampling Perceptron algorithm can be bounded as:

$$\mathbb{E}\left[\sum_{t=1}^{T} M_t\right] \leqslant \left(1 + \frac{R^2}{2\delta}\right) \frac{\bar{L}_{\gamma,T}(\mathbf{w})}{\gamma} + \frac{\|\mathbf{w}\|^2 \left(2\delta + R^2\right)^2}{8\delta\gamma^2}.$$

where $\bar{L}_{\gamma,T}(\mathbf{w}) = \mathbb{E}\left[\sum_{t=1}^{T} Z_t M_t \ell_{\gamma,t}(\mathbf{w})\right]$, and $\ell_{\gamma,t}(\mathbf{w}) = \max(0, \gamma - y_t \mathbf{w}^\top \mathbf{x}_t)$. Furthermore, the expected number of labels queried by the algorithm equals $\sum_{t=1}^{T} \mathbb{E}\left[\frac{\delta}{\delta + |\hat{p}_t|}\right]$. This bound depends on the value of the parameter δ. By choosing the optimal value of δ as

$$\delta = \frac{R^2}{2} \sqrt{1 + \frac{4\gamma^2}{\|\mathbf{w}\|^2 R^2} \frac{\bar{L}_{\gamma,T}(\mathbf{w})}{\gamma}}$$

the expected number of mistakes can be bounded as

$$\mathbb{E}\left[\sum_{t=1}^{T} M_t\right] \leqslant \frac{\bar{L}_{\gamma,T}(\mathbf{w})}{\gamma} + \frac{\|\mathbf{w}\|^2 R^2}{2\gamma^2} + \frac{\|\mathbf{w}\| R}{\gamma} \sqrt{\frac{\bar{L}_{\gamma,T}(\mathbf{w})}{\gamma} + \frac{\|\mathbf{w}\|^2 R^2}{4\gamma^2}}$$

This is an expectation version of the mistake bound for the standard Perceptron Algorithm. Especially, in the special case when the data is linearly separable, the optimal value of δ is $R^2/2$ and this bound becomes the familiar Perceptron bound $(\|\mathbf{w}\| R)^2/\gamma^2$.

Instead of using a fixed constant parameter, [72] also proposed an adaptive parameter version of the selective sampling Perceptron algorithm as follows:

$$\Pr(Z_t = 1) = \frac{\delta_t}{\delta_t + |\hat{p}_t|}, \quad \text{s.t.} \quad \delta_t = \beta (R')^2 \sqrt{1 + \sum_{i=1}^{t-1} Z_i M_i},$$

where $\beta > 0$ is a predefined parameter, $R' = \max R_{t-1}, \|\mathbf{x}_t\|$, $R_{t-1} = \max\{\|\mathbf{x}_i\| | Z_i M_i = 1\}$.

Other first-order approaches. Instead of using Perceptron, the Passive-Aggressive Active learning algorithms in [264,265] are selective sampling algorithms by extending the framework of PA online learning algorithms. They also extended their algorithms for multi-class classification and cost-sensitive classification tasks. [430] proposed a cost-sensitive online active learning approach that directly optimizes cost-sensitive measures using PA-like algorithms for class-imbalanced classification tasks.

6.2.2. Second-order selective sampling algorithms

Selective-sampling Second-order Perceptron. Instead of using the standard Perceptron algorithm, [72] also proposed a selective-sampling algorithm based on the Second-order Perceptron.

Algorithm 19: Selective-sampling Second-order Perceptron

```
INPUT: parameter $\delta > 0$
INIT: $A_0 = I$, $\mathbf{w}_0 = (0,\dots,0)^\top$
for $t = 1,2,\dots,T$ do
  Observe an input instance $\mathbf{x}_t \in \mathbb{R}^d$
  Computer
  $\hat{p}_t = \left[(A_t + \mathbf{x}_t\mathbf{x}_t^\top)^{-\frac{1}{2}}\mathbf{u}_t\right]^\top \left[(A_t + \mathbf{x}_t\mathbf{x}_t^\top)^{-\frac{1}{2}}\mathbf{x}_t\right] = \mathbf{u}_t^\top (A_t + \mathbf{x}_t\mathbf{x}_t^\top)^{-1}\mathbf{x}_t$
  Predict $\hat{y}_t = \mathrm{sign}(\hat{p}_t)$
  Draw a Bernoulli random variable $Z_t \in \{0,1\}$ of probability $\frac{\delta}{\delta+|\hat{p}_t|}$
  IF $Z_t = 1$ then
    Query label $y_t \in \{-1,+1\}$ and Update $\mathbf{w}_t$ by Second-order Perceptron:
    $\mathbf{u}_{t+1} = \mathbf{u}_t + M_t y_t \mathbf{x}_t$, and $A_{t+1} = A_t + M_t\mathbf{x}_t\mathbf{x}_t^\top$
end for
```

Let $\mathbf{u}_t$ denote the weight vector computed by standard Perceptron, and $A_t = I + \sum_{i \leqslant t-1, Z_i M_i = 1} \mathbf{x}_i\mathbf{x}_i^\top$ denote the correlation matrix over the mistaken trials plus an identity matrix I, then the second-order Perceptron predicts the label of current instance $\mathbf{x}_t$ as

$$\hat{y}_t = \mathrm{sign}(\hat{p}_t), \quad \text{where } \hat{p}_t = \left[(A_t + \mathbf{x}_t\mathbf{x}_t^\top)^{-\frac{1}{2}}\mathbf{u}_t\right]^\top \left[(A_t + \mathbf{x}_t\mathbf{x}_t^\top)^{-\frac{1}{2}}\mathbf{x}_t\right] = \mathbf{u}_t^\top (A_t + \mathbf{x}_t\mathbf{x}_t^\top)^{-1}\mathbf{x}_t$$

The second-order algorithm differs from standard Perceptron in that, before each prediction, a linear transformation $(A_t + \mathbf{x}_t\mathbf{x}_t^\top)^{-1/2}$ is applied to both current Perceptron weight $\mathbf{u}_t$ and current instance $\mathbf{x}_t$. After prediction, the query strategy of this algorithm is the same with the previous selective sampling: draw a Bernoulli random variable $Z_t \in \{0,1\}$ with

$$\Pr(Z_t = 1) = \frac{\delta}{\delta + |\hat{p}_t|}.$$

Algorithm 19 gives a summary of the Selective-sampling Second-order Perceptron algorithm. In theory, if the algorithm runs on a sequence of T rounds, for any $\mathbf{w}$, the expected number of mistakes made by the algorithm is bounded:

$$\mathbb{E}\left[\sum_{t=1}^{T} M_t\right] \leqslant \frac{\overline{L}_{\gamma,T}(\mathbf{w})}{\gamma} + \frac{\delta}{2\gamma^2}\mathbf{w}^\top\mathbb{E}[A_T]\mathbf{w} + \frac{1}{2\delta}\sum_{i=1}^{d}\mathbb{E}\ln(1+\lambda_i)$$

where $\overline{L}_{\gamma,T}(\mathbf{w}) = \mathbb{E}\left[\sum_{t=1}^{T} Z_t M_t \ell_{\gamma,t}(\mathbf{w})\right]$ with $\ell_{\gamma,t}(\mathbf{w}) = \max(0, \gamma - y_t\mathbf{w}^\top\mathbf{x}_t)$, $\lambda_1,\dots,\lambda_d$ are the eigenvalues of the random correlation matrix $\sum_{t=1}^{T} Z_t M_t \mathbf{x}_t\mathbf{x}_t^\top$ and $A_T = I + \sum_{t=1}^{T} M_t Z_t \mathbf{x}_t\mathbf{x}_t^\top$. Moreover, the expected number of queries by the algorithm equals $\sum_{t=1}^{T}\mathbb{E}\left[\frac{\delta}{\delta+|\hat{p}_t|}\right]$. Furthermore, by setting $\delta = \gamma\sqrt{\frac{\sum_{i=1}^{d}\mathbb{E}\ln(1+\lambda_i)}{\mathbf{w}^\top\mathbb{E}[A_T]\mathbf{w}}}$, it leads to the optimal bound

$$\mathbb{E}\left[\sum_{t=1}^{T} M_t\right] \leqslant \frac{\overline{L}_{\gamma,T}(\mathbf{w})}{\gamma} + \frac{1}{\gamma}\sqrt{(\mathbf{w}^\top\mathbb{E}[A_T]\mathbf{w})\sum_{i=1}^{d}\mathbb{E}\ln(1+\lambda_i)}$$

[72] also proposed an improved selective-sampling algorithm based on second-order Perceptron, which modifies the sampling probability by incorporating the second-order information, i.e., with

$$\Pr(Z_t = 1) = \frac{\delta}{\delta + |\hat{p}_t| + \frac{1}{2}\hat{p}_t^2\left(1 + \mathbf{x}_t^\top A_t^{-1}\mathbf{x}_t\right)},$$

Other second-order approaches. [67] proposed a margin-based selective sampling algorithm which also exploits second-order information in the model:

$$\hat{y}_t = \mathrm{sign}(p_t), \text{ where } \begin{aligned} p_t &= \mathbf{w}_t^\top\mathbf{x}_t, \ \mathbf{w}_t = A_t^{-1}\mathbf{u}_t, \\ \mathbf{u}_t &= \sum_{i=1}^{t-1} Z_i y_i \mathbf{x}_i^\top \\ A_t &= \left(I + \sum_{i=1}^{t-1} Z_i\mathbf{x}_i\mathbf{x}_i^\top\right) \end{aligned}$$

But the query strategy is a margin-based sampling approach without explicitly exploiting the second-order information:

$$\Pr(Z_{t+1} = 1) = \mathbb{I}\left(p_t \leqslant \frac{4\ln t}{\sum_{i=1}^{t-1} Z_i}\right)$$

[161,158] proposed second-order online active learning algorithms by fully exploiting both the first-order and second-order information for online active learning tasks and also gave cost-sensitive extensions for class-imbalanced tasks.

6.2.3. Other selective sampling approaches

There are also a few other selective sampling approaches in which the base classifier is based on the Regularized Least Squares (RLS). In particular, on each round t, the linear classification model can be updated by the RLS estimate

$$\mathbf{w}_t = \left(I + S_{t-1}S_{t-1}^\top + \mathbf{x}_t\mathbf{x}^\top\right)^{-1} S_{t-1}Y_{t-1}$$

where matrix $S_{t-1} = \left[\mathbf{x}'_1,\dots,\mathbf{x}'_{N_{t-1}}\right]$ is the collection of N_{t-1} instances queried up to time $t-1$, and the vector $Y_{t-1} = \left(Y'_1,\dots,Y'_{N_{t-1}}\right)$ is the set of queried labels for the instances. The selective sampling algorithms that follow this paradigm include the Bound on Bias Query (BBQ) algorithms [71,288] and their improved variants [106,288]. A major drawback of these methods is that the RLS-based base learner is more like a fashion of batch learner instead of truly online learning, and thus the overall learning scheme might be inefficient or non-scalable if the number of queried labeled examples can be large.

6.3. Online active learning with expert advice

The idea of online active learning with expert advice dates back to classical Query by Committee (QBC) in [330,131], where the idea is to query the label of an instance based on the principle of *maximal disagreement* among a set of experts, i.e., the confidence criteria in this case is how much the expert hypotheses disagree on their evaluation of instance predictions. QBC bounds from below the average information gain provided by each requested label. [33] considers the setting of how to online combine an ensemble of active learners, which is executed based on a maximum entropy criterion. Another perhaps more dominating line of studies in [169,75,434,157] explore the exponentiated weighted average forecaster for online active learning tasks, where an instance is stochastically queried based on the available feedback on the importance of each expert in the pool.

Next we describe in detail one of the most recent approaches for online active learning with expert advice in [434]. Consider an unknown sequence of instances $\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^d$, a "forecaster" aims to predict the class labels of every incoming instance $\mathbf{x}_t$. The forecaster sequentially computes its predictions based on the predictions from a set of N "experts". Specifically, at the t-th round, after receiving an instance $\mathbf{x}_t$, the forecaster first accesses the predictions of the experts $\{f_{i,t} : \mathbb{R}^d \to [0,1] | i = 1, \ldots, N\}$, and then computes its own prediction $p_t \in [0,1]$ based on the predictions of the N experts. After p_t is computed, the true outcome $y_t \in \{0,1\}$ is disclosed. To solve this problem, the "Exponentially Weighted Average Forecaster" (EWAF) makes the following prediction:

$$p_t = \frac{\sum_{i=1}^N \exp(-\eta L_{i,t-1}) f_i(\mathbf{x}_t)}{\sum_{i=1}^N \exp(-\eta L_{i,t-1})}, \tag{14}$$

where η is a learning rate, $L_{i,t} = \sum_{j=1}^t \ell(f_i(\mathbf{x}_j), y_j), \quad L_t = \sum_{j=1}^t \ell(p_j, y_j)$ with $\ell(p_t, y_t) = |p_t - y_t|$.

Unlike the above regular learning, in an active learning with expert advice task, the outcome of an incoming instance is *only* revealed whenever the learner requests the label from the environment/oracle. To solve this problem, binary variables $z_s \in \{0,1\}, s = 1, \ldots, t$ are introduced to indicate if an active forecaster has requested the label of an instance at s-th trial. $\widehat{L}_{i,t}$ is used to denote the loss function experienced by the active learner w.r.t. the i^{th} expert, i.e., $\widehat{L}_{i,t} = \sum_{s=1}^t \ell(f_i(\mathbf{x}_s), y_s) z_s$. For this problem setting, [434] proposed a general framework of active forecasters, as shown in Algorithm 20.

Algorithm 20: Online Active Learning with Expert Advice

```
INPUT: a pool of experts $f_i,\ i = 1, \ldots, N$.
INIT: tolerance threshold $\delta$ and $\widehat{L}_{i,t} = 0,\ i \in [N]$.
for $t = 1, 2, \ldots, T$ do    Receive $\mathbf{x}_t$ and compute $f_i(\mathbf{x}_t),\ i \in [N]$;
    Compute $\hat{p}_t = \frac{\sum_{i=1}^N \exp\left(-\eta \widehat{L}_{i,t-1}\right) f_i(\mathbf{x}_t))}{\sum_{i=1}^N \exp\left(-\eta \widehat{L}_{i,t-1}\right)}$;
    If a confidence condition is not satisfied
        request label $y_t$ and update
    $\widehat{L}_{i,t} = \widehat{L}_{i,t-1} + \ell(f_i(\mathbf{x}_t), y_t),\ i \in [N]$;
end for
```

At each round, after receiving an instance $\mathbf{x}_t$, we compute the prediction of class label for the instance by aggregating the prediction of each expert in the pool, i.e., $f_i(\mathbf{x}_t)$. Then, we examine if a confidence condition is satisfied. If so, we will skip the label request; otherwise, the learner will request the class label for this instance from the environment. To decide when to request the class label or not, the key idea is to seek a confidence condition by estimating the difference between p_t and $\hat{p}_t$. Intuitively, the smaller the difference, the more confident we have for the prediction made by the forecaster. More specifically, the work in [434] proved that for a small constant $\delta > 0$, $\max_{1 \leqslant i,j \leqslant N} |f_i(\mathbf{x}_t) - f_j(\mathbf{x}_t)| \leqslant \delta$ implies $|p_t - \widehat{p}_t| \leqslant \delta$. This roughly means that, if any two experts do not disagree with each other too much on the instance, then we can skip requiring its label.

In addition to the above work, there are also a few other active learning strategies for online learning with expert advices, for example the active greedy forecaster [434]. Online active learning with expert advice can be applied in some real-world applications, e.g., crowdsourcing tasks [156] where the learner attempts to address both the diverse quality of annotators' performance with expert learning and efficient annotation in seeking informative data using active learning.

7. Online semi-supervised learning

7.1. Overview

Semi-Supervised Learning (SSL) has been an important class of machine learning tasks and techniques, which aims to make use of unlabeled data for learning tasks. It has been extensively studied mostly in the settings of batch learning and some comprehensive surveys can be found in [442,444]. When online learning meets semi-supervised learning, there are two major branches of research. One major branch of studies is to turn traditional batch semi-supervised learning methods into online algorithms such that they can work from data streams of both labeled and unlabeled data, which we call this setting as "Online Semi-supervised Learning" and we will review a popular framework of "online manifold regularization". The other branch of studies is to study classical online learning tasks in transductive learning settings (e.g., by assuming unlabeled data can be made available before online learning tasks), which we call this setting as "Transductive Online Learning". We note that online active learning as introduced previously can be generally viewed as a special type of online semi-supervised learning where an online learner has to deal with both labeled and unlabeled data, and infact several studies even combine the two into a unified framework [282].

7.2. Online manifold regularization

In the area of semi-supervised learning, one major framework for semi-supervised learning is based on manifold regularization [36], where the learner not only minimizes the loss on the labeled data, but also minimizes the difference of predictions on the unlabeled instances which are similar on the manifold. Specifically, consider instances $(\mathbf{x}_t, y_t),\ t \in \{1, \ldots T\}$, the idea is to minimize the following objective function

$$J(f) = \frac{1}{l} \sum_{t=1}^T \delta(y_t) \ell(f(\mathbf{x}_t), y_t) + \frac{\lambda_1}{2} ||f||^2 + \frac{\lambda_2}{2T} \sum_{s,t=1}^T (f(\mathbf{x}_s) - f(\mathbf{x}_t)) w_{st}$$

where the first term is the loss on labeled instances where $\delta(y_t) = 1$ if and only if y_t exists and l is the number of labeled data, the second term is a classic regularization term for supervised learning, and the last term is the manifold regularization on unlabeled data.

In literature, online manifold regularization has been explored [146], which attempts to turn batch manifold regularization algorithms into online/incremental algorithms. Specifically, the above objective can be returned online for each instance:

$$J_t(f) = \frac{T}{l} \delta(y_t) \ell(f(\mathbf{x}_t), y_t) + \frac{\lambda_1}{2} ||f||^2 + \lambda_2 \sum_{i=1}^{t-1} (f(\mathbf{x}_i) - f(\mathbf{x}_t)) w_{it}$$

It can be solved using Online Gradient Descent in $O\left(T^2\right)$ time. Unfortunately, such straightforward solution is expensive in both time and space, since the calculation of the last term requires to store all instances and measure the similarity w_{it} between the incoming instances and all existing ones.

To address this problem, the authors offer two sparse approximations of the objective function. The first solution is not to keep all instances but to keep only the newest τ ones, where τ is the buffer size. This strategy is simple but not very efficient since the discarded old instances may contain important information. The second solution adopts a random projection tree to find s cluster centers during online learning. Finally, instead of calculating the similarity between $\mathbf{x}_t$ and all existing instances, the algorithm only consider the s cluster centers as the most representative instances.

In addition to the above work, [361] proposed a fast approximate algorithm for online semi-supervised learning. which leverages the incremental k-center quantization method to group neighboring points so as to yield a set of reduced representative points, and as a result an approximate similarity graph can be constructed to find the harmonic solution in semi-supervised learning [443].

Finally, there were some related efforts on online active semi-supervised learning [147], which extends active learning in the online semi-supervised learning settings. For example, following such kind of setting, [147] developed the OASIS algorithm by using a general online Bayesian learning framework.

7.3. Transductive online learning

Transductive online learning [76,37] is a niche class of online learning tasks, where we want to learn from an arbitrary sequence of labeled examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ by making the assumption that the set of unlabeled instances $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ can be given in advance to the learner before an online learning task begins. In particular, the work [76] proposed an efficient algorithm based on the principle of prediction with expert advice by combining "random playout" and "randomized rounding" of loss subgradients. We note that this niche topic has received very few attention, possibly because of their assumption of obtaining unlabeled data in advance, which may be unrealistic in many applications. However, the studies in this niche family of studies may provide some theory insights about the linkage between online learning and batch learning as demonstrated in [76].

8. Online unsupervised learning

8.1. Overview

In this section we briefly review some key work in the literature of online unsupervised learning, where models are learned from unlabeled data streams where no explicit feedback is available. Broadly, we categorize the existing work into four major groups: Online Clustering, Online Dimension Reduction, Online Anomaly Detection, and Online Density Estimation. Due to the vast number of ways in which unsupervised learning in online settings have been explored in literature, and numerous applications for which algorithms are designed, it is almost impossible to make a comprehensive treatment on this topic in this survey. Instead, we try to focus on the key areas and give a general overview of the main ideas in each area which are closely related to online learning.

8.2. Online clustering

Clustering is an unsupervised learning process of grouping unlabeled data instances such that instances in the same group are similar, and instances between groups are dissimilar. It gives an effective mechanism to summarize the data, and does not require labels of the instances in order to perform the clustering. For batch learning settings, clustering is usually classified into following categories: partition based clustering, hierarchical clustering, density-based clustering, and grid-based clustering [38]. For online settings [8], partition-based and density-based clustering have been studied more extensively. In the following we briefly review some of online learning approaches for clustering on streaming data especially for these two categories.

Partitioning Based clustering methods split the instances into partitions where each partition represents a cluster. The partitions are designed on the basis of some distance measures (e.g. Euclidean distance). The number of clusters is usually pre-defined by the user. The most popular algorithms in this category are those based on *k-MEANS* and *k-MEDOIDS* algorithm. The k-MEANS algorithm is one of the oldest and most popular clustering methods, where the idea is to identify k centroids, where each centroid corresponds to one out of k clusters by minimizing the sum of square errors between each instance to their corresponding centroids. Sequential algorithms performing k-MEDOID or k-MEDIAN clustering usually try to break the stream of instances into chunks where the size of each chunk is set based on some pre-specified memory budget. Given a data stream D, it is broken into several chunks denoted by $D_1, D_2, \ldots, D_t, \ldots$where each chunk contains at most m instances, where m is the budget of the chunks. In such a case, k-MEDIANS can be directly applied to each chunk. This framework is called the STREAM framework. [149,287,148]. There are also sampling approaches designed for clustering when the data streams are extremely large [198]. Another method is the StreamKM++ [4]. In this approach, first, an adaptive non-uniform sampling approach is used to obtain small coresets from the data streams. The coreset construction is done by the utilization of coreset tree proposed in this paper which helps in significant speed up.

Density-based clustering. Most clustering techniques suffer from several drawbacks. First, many of them (e.g. k-MEANS) are designed for only spherical clusters and can not adapt to arbitrary cluster shapes. In addition, the value of k, or the number of clusters has to be known a priori. Lastly, these methods are susceptible to outliers. Density based clustering algorithms (most popularly DBSCAN and its variants) are able to address all these challenges. Density based approaches cluster dense regions which are separated by sparse regions. A cluster based on density can take on arbitrary shapes, does not require prior knowledge of the number of clusters, and is robust to outliers in the data. However, performing density based clustering on streaming data in an online manner is plagued with several challenges including dynamic evolution of the clusters, limited memory space, etc. Following [8,16], we categorize the online density-based clustering algorithms into *Micro-clustering Algorithms* and *Grid-based clustering Algorithms*. The micro-clustering algorithms aim to summarize a data in an online manner, and the clustering is performed using these summaries [59,352,315,247,309,286]. Grid-based methods, divide the entire instance space into grids, and each instance upon arrival is assigned a grid, and then the clustering is then done based on the density of the grids [134,87,185,358,367,308,17,41].

Other Clustering Methods. Hierarchical Clustering is a paradigm in which either a bottom-up approach or a top-down approach is used to gradually agglomerate the data points together. This results in a tree of clusters, which is also called a dendogram. Among the earliest approaches to incremental hierarchical clustering was CobWeb [126], which determines how to insert a new data point into the tree structure based on a category utility criteria. Recent hierarchical clustering algorithms include the ClusTree [210], which offers a compact self adapting index structure for storing stream summaries in addition to giving more importance to recent data, and Perch [207], which allows the clustering to scale to a large number of data points and clusters. [359] propose an incremental approach to do Hierarchical clustering of the data, in addition to accounting for variance and density of the data. Some techniques have been developed for online clustering for very high dimensional data where the data sparsity makes it very hard to perform clustering as many instances tend to be equidistant from one another. HPStream[9] introduces a concept of projected clustering to data streams. There are online clustering algorithms for other specific scenarios such as clustering of discrete and categorical streams, text streams, uncertain data streams, graph streams as well as distributed clustering [8].

8.3. Online dimension reduction

When the feature dimensions are very large, Dimension Reduction (DR) techniques can be used to improve learning efficiency, compress original data, visualize data better, and improve its applicability to real-world applications. Consider instance $\mathbf{x}_t \in \mathbb{R}^d$, the goal of dimension reduction is to learn a new instance $\hat{\mathbf{x}}_t \in \mathbb{R}^k$ where $k \leqslant d$ by following some principle of unsupervised learning. There have been several approaches to unsupervised dimensional reduction. We broadly categorize them into two major groups of studies: subspace learning and manifold learning More comprehensive surveys of classic dimension reduction techniques can be found in [58].

Subspace Learning. This class of DR methods aims to find an optimal linear mapping of input data in high-dimensional space to a lower-dimensional space. In general, there are two major types of approaches: linear methods and nonlinear methods. Popular linear subspace methods include Principal Component Analysis (PCA) and Independent Component Analysis (ICA), etc. Nonlinear methods often extend the linear subspace learning methods using kernel tricks. Examples include Kernel PCA, Kernel ICA, etc.

For online dimension reduction tasks, more popular efforts have been focused on addressing online PCA for unsupervised learning on streaming data settings in literature [390,22,21,275,124], while there are also a few studies for online ICA [235,368]. For nonlinear space learning methods, online Kernel-PCA has also received some research interests [215,176].

Manifold learning. This class of DR methods generally belongs to nonlinear DR techniques. Manifold learning assumes that input data lie on an embedded non-linear manifold within the high-dimensional space. DR by manifold learning aims to find a low-dimensional representation by preserving some properties of the manifold. For example, some methods preserving global properties include Multi-dimensional scaling (MDS), and IsoMap [353], while some preserving local properties including Locally Linear Embedding (LLE) [314] and Laplacian Eigenmaps.

For online manifold learning settings, the are some efforts for achieving the incremental approaches of manifold learning in literature. For example, [220] proposed an incremental learning algorithm for ISOMAP and [327] presented an online approach for LLE.

8.4. Online density estimation

Online density estimation refers to constructing an estimate of an underlying unobservable probability density function based on observed data streams [340]. In literature, there are many different approaches to perform density estimation, e.g., histograms, naive estimator, nearest neighbour methods, Parzen windows, etc. Among various approaches, Kernel Density Estimation (KDE) is probably one of the most extensively explored topics in density estimation, which is a non-parametric way to estimate the probability density function of a target random variable [328]. Here, we briefly review and categorize some of the commonly used approaches for kernel density estimation in online-learning settings. Given a sequence of instances $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, where $\mathbf{x}_t \in \mathbb{R}^d$, KDE estimates the density at a point $\mathbf{x}$ as

$$\mathbf{f}(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T}\kappa(\mathbf{x}, \mathbf{x}_t) = \frac{1}{Th}\sum_{t=1}^{T}\kappa\left(\frac{\mathbf{x} - \mathbf{x}_t}{h}\right)$$

where the kernel $\kappa(\mathbf{x}, \mathbf{x}_t)$ is a radially symmetric unimodal function that integrates to 1 and h is a smoothing parameter called the bandwidth. Like in the case of Online Learning with Kernels [202,262], this problem suffers from the curse of kernelization, which means to estimate the density at any point $\mathbf{x}$, it requires computing the kernel function with respect to all the data points observed in the data stream so far.

There have been several attempts to overcome this curse of kernelization, and can be grouped into *Merging* and *Sampling* approaches. Merging approaches require a pre-specified budget on how many instances or kernels can be stored in memory. A newly arriving sample will (typically) be stored in memory as a kernel, unless the budget is exceeded. If the budget is exceeded, two or more similar kernels get merged. The merging criteria depends on some objective function. Some efforts in this direction include [440,46,212], which usually differ in how they select the bandwidth values. Another approach in [60] performs clustering using self-organizing maps, and leverages this to perform kernel merging. Sampling approaches randomly select points to be kept in memory, but attempts to maintain a certain level of accuracy [439]. Recent approaches [302] try to perform online density estimation with efficient methods for bandwidth selection and also to capture changes in the data distribution. Finally, online KDE techniques can be applied and integrated with real-world applications, such as real-time visual tracking [153].

8.5. Online anomaly detection

Anomaly Detection (AD), also known as “outlier detection” or “novelty detection”, is the process of detecting abnormal behavior in the data. The definition of abnormal behaviors can be very subjective, and the notion of “anomaly” varies from domain to domain. Anomaly detection research is abundant in literature due to its wide applications. Example applications include but not limited to intrusion detection, fraud detection, medical anomaly detection, industrial damage detection, amongst others. Anomaly detection has been extensively studied by many communities in a wide range of diverse settings, ranging from supervised to unsupervised and semi-supervised learning, and batch learning to online learning settings. More comprehensive surveys of classic anomaly detection studies can be found in [77,150]. In this survey, we will focus online anomaly detection in unsupervised learning settings, which we believe it is one of the most popular and dominating scenarios in many real-world applications.

According to the literature surveys, unsupervised anomaly detection can be grouped into several major categories, including Distance based, Density based, Clustering based, Statistical methods, and others (such as subspace and one-class learning, etc). In the following, we briefly review some of popular work by focusing on online learning settings.

In literature, distance based online AD algorithms have been extensively studied in the context of unsupervised learning over data streams using distance-based methods [19,406,54,206]. Some typical strategies of distance-based online AD approaches is to apply the sliding window model where distance-based anomalies/outliers can be detected in the current window. In addition to distance-based methods, there are also some other studies that explore different methods for online AD in data streams, such as using one-class anomaly detector [350] or online clustering based approaches [345]. We note that, despite extensive and diverse studies in the field of online anomaly detection, from a machine learning perspective, many of theses approaches (e.g., sliding windows based) not purely learn in an online-learning fashion and many are not designed in machine learning based manners. We therefore keep the review of this part brief and concise.

9. Related areas and other terminologies

9.1. Overview

In this section, we discuss the relationship of online learning with other related areas and terminologies which sometimes may be confused. We note that some of the following remarks may be somewhat subjective, and their meanings may vary in diverse contexts whereas some terms and notions may be used interchangeably.

9.2. Incremental learning

Incremental learning, or decremental learning, represents a family of machine learning techniques [274,297,307], which are particularly suitable for learning from data streams. There are various definitions of incremental learning/decremental learning. The basic idea of incremental learning is to learn some models from a stream of training instances with limited space and computational costs, often attempting to approximate a traditional batch machine learning counterpart as much as possible. For example, incremental SVM [297] aims to train an SVM classifier the same as a batch SVM in an incremental manner where one training instance is added for updating the model each time (and similarly a training instance can be removed by updating the model decrementally).

Incremental learning can work either in online learning or batch learning manners [307]. For the incremental online learning [297], only one example is presented for updating the model at one time, while for the incremental batch learning [372], a batch of multiple training examples are used for updating the model each time. Incremental learning (or decremental learning) methods are often natural extensions of existing supervised learning or unsupervised learning techniques for addressing efficiency and scalability when dealing with real-world data particularly arriving in stream-based settings. Generally speaking, incremental learning can be viewed as a branch of online learning and extensions for adapting traditional offline learning counterparts in data-stream settings.

9.3. Sequential learning

Sequential learning is mainly concerned with learning from sequential training data [110], formulated as follows: a learner trains a model from a collection of N training data pairs $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}), i = 1, \ldots, N\}$ where $\mathbf{x}^{(i)} = \left(x_1^i, x_2^i, \ldots, x_{N_i}^i\right)$ is an N_i-dimensional instance vector and $\mathbf{y}^{(i)} = \left(y_1^i, y_2^i, \ldots, y_{N_i}^i\right)$ is an N_i-dimensional label vector. It can be viewed as a special type of supervised learning, known as structured prediction or structured (output) learning [32], where the goal is to predict structured objects (e.g., sequence or graphs), rather than simple scalar discrete ("classification") or real values ("regression"). Unlike traditional supervised learning that often assume data is independently and identically distributed, sequential learning attempts to exploit significant sequential correlation of sequential data when training the predictive models. Some classical methods of sequential learning include sliding window methods, recurrent sliding windows, hidden Markov models, conditional random fields, and graph transformer networks, etc. There are also many recent studies for structured prediction with application to sequential learning [32,312]. In general, sequential learning can be solved by either batch or online learning algorithms. Finally, it is worth mentioning another closely related learning, i.e., "sequence classification", whose goal is to predict a single class output for a whole input "sequence" instance. Sequence classification is a special case of sequential learning with the target class vector reduced to a single variable. It is generally simpler than regular sequential learning, and can be solved by either batch or online learning algorithms.

9.4. Stochastic learning

Stochastic learning refers to a family of machine learning algorithms by following the theory and principles of stochastic optimization [50,423,51], which have achieved great successes for solving large-scale machine learning tasks in practice [52]. Stochastic learning is closely related to online learning. Typically, stochastic learning algorithms are motivated to accelerate the training speed of some existing batch machine learning methods for large-scale machine learning tasks, which may be often solved by batch gradient descent algorithms. Stochastic learning algorithms, e.g., Stochastic Gradient Descent (SGD) or a.k.a Online Gradient Descent (OGD) in online learning terminology, often operate sequentially by processing one training instance (randomly chosen) each time in an online learning manner, which thus are computationally more efficient and scalable than the batch GD algorithms for large-scale applications. Rather than processing a single training instance each time, a more commonly used stochastic learning technique in practice is the mini-batch SGD algorithm [52,336], which processes a small batch of training instances each time. Thus, stochastic learning can be viewed as a special family of online learning algorithms and extensions, while online learning may explore more other topics and challenges beyond stochastic learning/optimizations.

9.5. Adaptive learning

This term is occasionally used in the machine learning and neural networks fields. There is no a very formal definition about what exactly is adaptive learning in literature. In literature, there are quite a lot of different studies more or less concerned with adaptive learning [63,62], which attempt to adapt a learning model/system (e.g., neural networks) for dynamically changing environments over time. In general, these existing works are similar to online learning in that the environment is often changing and evolving dynamically. But they are different in that they are not necessarily purely based on online learning theory and algorithms. Some of these works are based on heuristic adaptation/modification of existing batch learning algorithms for updating the models with respect to the environment changes. Last but not least, most of these existing works are motivated by different kinds of heuristics, generally lack solid theoretical analysis and thus can seldom give performance guarantee in theory.

9.6. Interactive learning

Traditional machine learning mostly works in a fully automated process where training data are collected and prepared typically with the aid of domain experts. By contrast, interactive (machine) learning aims to make the machine learning procedure interactive by engaging human (users or domain experts) in the loop [389,189]. The advantages of interactive learning include the natural integration of domain knowledge in the learning process, effective communication and continuous improvements for learning efficacy through the interaction between learning systems and users/experts. Online learning often plays an important role in an interactive learning system, in which active (online) learning can be used in finding the most informative instances to save labeling costs, incremental (online) learning algorithms could be applied for updating the models sequentially, and/or bandit online learning algorithms may be used for decision-making to trade off exploration and exploitation in some scenarios.

9.7. Reinforcement learning

Reinforcement Learning (RL) [191,348] is a branch of machine learning inspired by behaviorist psychology, which is often concerned with how software agents should take actions in an environment to maximize cumulative rewards. The goal of an agent in RL is to find a good policy and state-update function by attempting to maximize the the expected sum of discounted rewards. RL is different from supervised learning [35] in that the goal of supervised learning is to reconstruct an unknown function f that can assign the desired output values y to input data x; while the goal of RL is to find the input (policy/action) x that gives the maximum reward $R(x)$. In general, RL can work either in batch or online learning manners. In practice, RL methods are commonly applied to problems involving sequential dynamics and optimization of some objectives, typically with online exploration of the effects of actions. RL is closely related to bandit online learning with the similar goal of finding a good policy that has to balance the tradeoff between exploration (of uncertainty) and exploitation (of known knowledge). Many RL solutions follow the same ideas of multi-armed bandits, and some bandit algorithms were also inspired by the field of RL studies too. However, RL can be more general when learning to interact with more complex scenarios and environments.

9.8. Continual learning

Continual Learning, also called "Lifelong Learning" [318,339,293,44,277,294] is a field of machine learning inspired by human ability to learn new tasks throughout their lifespan. When new tasks arrive, humans are able to leverage existing knowledge, and more effectively learn the new tasks, and at the same time, they do not forget how to perform the old tasks. In formal settings, the tasks arrive sequentially, but instances for each task arrive as a batch, and thus each task is still learned in batch settings. While older methods used linear models for lifelong learning [318], recent efforts have been focused on continual learning with neural networks [293], in which one of key challenges is to address the *catastrophic forgetting*, a problem which traditional machine learning including neural networks is often susceptible to, but humans are immune to. When new tasks are learned, traditional machine learning tends to forget how to perform older tasks, and a major research direction in continual learning is to develop algorithms that can address this catastrophic forgetting. Although continual learning is closely related to online learning, most existing studies still follow the paradigm of batch training, which are not considered as online learning algorithms.

10. Conclusions

10.1. Concluding remarks

This paper gave a comprehensive survey of existing online learning works and reviewed ongoing trends of online learning research. In theory, online learning methodologies are founded primarily based on learning theory, optimization theory, and game theory. According to the type of feedback to the learner, the existing online learning methods can be roughly grouped into the following three major categories:

- **Supervised online learning** is concerned with the online learning tasks where full feedback information is always revealed to the learner, which can be further divided into three groups: (i) "“linear online learning” that aims to learn a linear predictive model, (ii) “nonlinear online learning” that aims to learn a nonlinear predictive model, and (iii) non-traditional online learning that addresses a variety of supervised online learning tasks which are different from traditional supervised prediction models for classification and regression.
- **Online learning with limited feedback** is concerned with the online learning tasks where the online learner receives partial feedback information from the environment during the learning process. The learner often has to make online predictions or decisions by achieving a tradeoff between the exploitation of disclosed knowledge and the exploration of unknown information.
- **Unsupervised online learning** is concerned with the online learning tasks where the online learner only receives the sequence of data instances without any additional feedback (e.g., true class label) during the online learning tasks. Examples of unsupervised online learning include online clustering, online representation learning, and online anomaly detection tasks, etc.

In this survey, we have focused more on the first category of work because it has received more research attention than the other two categories in literature. This is mainly because supervised online learning is a natural extension of traditional batch supervised learning, and thus an online supervised learning technique could be directly applied to a wide range of real-world applications especially for real-time machine learning from data streams where conventional batch supervised learning techniques may suffer critical limitations. However, we do note that in contrast to supervised online learning, the problems of online learning with limited feedback or unsupervised online learning are generally much more challenging, and thus should attract more research attentions and efforts in the future.

10.2. Future directions

Despite the extensive studies in literature, there are still many open issues and challenges, which have not been fully solved by the existing works and need to be further explored by the community efforts in the future work. In the following, we highlight a few important and emerging research directions for researchers who are interested in online learning.

First of all, although supervised online learning has been extensively studied, learning from non-stationary data streams remain an open challenge. In particular, one critical challenge with supervised online learning is to address "concept drifting" where the target concepts to be predicted may change over time in unforeseeable ways. Although many online learning studies have attempted to address concept drifting by a variety of approaches, they are fairly limited in that they often make some restricted assumptions for addressing certain types of concept drifting patterns. In general, there still lacks of formal theoretical frameworks or principled ways for resolving all types of concept drifting issues, particularly for non-stationary settings where target concepts may drift over time in arbitrary ways.

Second, an important growing trend of online learning research is to explore large-scale online learning for real-time big data analytics. Although online learning has huge advantages over batch learning in efficiency and scalability, it remains a non-trivial task when dealing with real-world big data analytics with extremely high volume and high velocity. Despite extensive research in large-scale batch machine learning, more future research efforts should address parallel online learning and distributed online learning by exploiting various computational resources, such as high-performance computing machines, cloud computing infrastructures, and perhaps low-cost IoT computing environments.

Third, another challenge of online learning is to address the "variety" in online data analytics tasks. Most existing online learning studies are often focused on handling single-source structured data typically by vector space representations. In many real-world data analytis applications, data may come from multiple diverse sources and could contain different types of data (such as structured, semi-structured, and unstructured data). Some existing studies, such as the series of online multiple kernel learning works, have attempted to address some of these issues, but certainly have not yet fully resolved all the challenges of variety. In the future, more research efforts should address the "variety" challenges, such as multi-source online learning, multi-modal online learning, etc.

Fourth, existing online learning works seldom address the data "veracity" issue, that is, the quality of data, which can considerably affect the efficacy of online learning. Conventional online learning studies often implicitly assume data and feedback are given in perfect quality, which is not always true for many real-world applications, particularly for real-time data analytics tasks where data arriving on-the-fly may be contaminated with noise or may have missing values or incomplete data without applying advanced pre-processing. More future research efforts should address the data veracity issue by improving the robustness of online learning algorithms particularly when dealing with real data of poor quality.

Fifth, due to the remarkable successes and impact of deep learning techniques for various applications in recent years, another emerging and increasingly important topic is "online deep learning" [322], i.e., learning deep neural networks from data streams on the fly in an online fashion. Despite some preliminary research, we note there are still many research challenges in this field, e.g., how to balance the tradeoff between learning accuracy, computational efficiency, learning scalability and model complexity.

Last but not least, we believe it can be valuable to explore "online continual learning" by extending traditional continual learning methods for pure online-learning settings, which is more natural for many real-world applications where data for either existing and novel tasks are often arriving in a streaming and continuous fashion. Some research progress in online deep learning might be applied here, but the key challenge of continual online learning is to resolve the catastrophic forgetting problem across tasks during the online learning process.

CRediT authorship contribution statement

Steven C.H. Hoi: Conceptualization, Investigation, Writing - original draft, Writing - review & editing, Supervision, Project administration. **Doyen Sahoo:** Investigation, Writing - original draft, Writing - review & editing. **Jing Lu:** Investigation, Writing - original draft, Writing - review & editing. **Peilin Zhao:** Investigation, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

aaaa

References

[1] Y. Abbasi-Yadkori, D. Pál, C. Szepesvári, Improved algorithms for linear stochastic bandits, in: Advances in Neural Information Processing Systems, 2011, pp. 2312–2320. .
[2] J. Abernethy, K. Canini, J. Langford, A. Simma, Online collaborative filtering (Tech. Rep.), University of California at Berkeley, 2007.
[3] J. Abernethy, E. Hazan, A. Rakhlin, Competing in the dark: An efficient algorithm for bandit linear optimization, in: COLT, 2008, pp. 263–274.
[4] M.R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, C. Sohler, Streamkm++: a clustering algorithm for data streams, J. Exp. Algorithmics 17 (2012) 2–4.
[5] A. Agarwal, E. Hazan, S. Kale, R.E. Schapire, Algorithms for portfolio management based on the newton method, ICML, ACM (2006) 9–16.
[6] A. Agarwal, M.J. Wainwright, J.C. Duchi, Distributed dual averaging in networks, Advances in Neural Information Processing Systems (2010) 550–558.
[7] R. Agarwal, A.A. Sekh, K. Agarwal, D.K. Prasad, Auxiliary network: scalable and agile online learning for dynamic system with inconsistently available inputs, 2020, arXiv preprint arXiv:2008.11828..
[8] C.C. Aggarwal, A survey of stream clustering algorithms, 2013..
[9] C.C. Aggarwal, J. Han, J. Wang, P.S. Yu, A framework for projected clustering of high dimensional data streams, in: VLDB, 2004..
[10] S. Agmon, The relaxation method for linear inequalities, Can. J. Math. 6 (1954) 382–392.
[11] S. Agrawal, N. Goyal, Analysis of thompson sampling for the multi-armed bandit problem, Conference on Learning Theory (2012) 31–39.
[12] S. Agrawal, N. Goyal, Thompson sampling for contextual bandits with linear payoffs, International Conference on Machine Learning (2013) 127–135.
[13] K. Akcoglu, P. Drineas, M.Y. Kao, Fast universalization of investment strategies, SIAM J. Comput. 34 (2004) 1–22.
[14] S. Albers, Online algorithms: a survey, Math. Program. (2003).
[15] M. Ali, C.C. Johnson, A.K. Tang, Parallel collaborative filtering for streaming data (Tech. Rep.), University of Texas Austin, 2011.
[16] A. Amini, T.Y. Wah, H. Saboohi, On density-based data streams clustering algorithms: a survey, J. Comput. Sci. Technol. 29 (2014) 116–141.
[17] A. Amini, W. Ying, Dengris-stream: a density-grid based clustering algorithm for evolving data streams over sliding window, in: Proc. International Conference on Data Mining and Computer Engineering, 2012, pp. 206–210.
[18] O. Anava, E. Hazan, S. Mannor, O. Shamir, Online learning for time series prediction, Conference on Learning Theory (2013) 172–184.
[19] F. Angiulli, F. Fassetti, Detecting distance-based outliers in streams of data, in: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, ACM, 2007, pp. 811–820.
[20] K. Ariu, N. Ryu, S.Y. Yun, A. Proutière, Regret in online recommendation systems, in: Advances in Neural Information Processing Systems 33, 2020.
[21] R. Arora, A. Cotter, K. Livescu, N. Srebro, Stochastic optimization for pca and pls, in: Allerton Conference, Citeseer, 2012a, pp. 861–868..
[22] R. Arora, A. Cotter, N. Srebro, Stochastic optimization of pca with capped msg, Advances in Neural Information Processing Systems (2013) 1815–1823.
[23] S. Arora, E. Hazan, S. Kale, The multiplicative weights update method: a meta-algorithm and applications, Theory Comput. 8 (2012) 121–164.
[24] A. Ashfahani, M. Pratama, Autonomous deep learning: continual learning approach for dynamic environments, in: Proceedings of the 2019 SIAM International Conference on Data Mining, 2019, pp. 666–674.
[25] A. Ashfahani, M. Pratama, E. Lughofer, Y.S. Ong, Devdan: deep evolving denoising autoencoder, Neurocomputing 390 (2020) 297–314.
[26] L.E. Atlas, D.A. Cohn, R.E. Ladner, Training connectionist networks with queries and selective sampling, in: D.S. Touretzky (Ed.), Advances in Neural Information Processing Systems 2, Morgan-Kaufmann, 1990, pp. 566–573.
[27] J.Y. Audibert, R. Munos, C. Szepesvári, Exploration–exploitation tradeoff using variance estimates in multi-armed bandits, Theoret. Comput. Sci. 410 (2009) 1876–1902.
[28] P. Auer, Using confidence bounds for exploitation-exploration trade-offs, J. Mach. Learn. Res. 3 (2002) 397–422.
[29] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Mach. Learn. 47 (2002) 235–256.
[30] P. Auer, N. Cesa-Bianchi, Y. Freund, R.E. Schapire, Gambling in a rigged casino: the adversarial multi-armed bandit problem, in: Focs, IEEE, 1995, p. 322..
[31] P. Auer, N. Cesa-Bianchi, Y. Freund, R.E. Schapire, The nonstochastic multiarmed bandit problem, SIAM J. Comput. 32 (2002) 48–77.
[32] G. BakIr, Predicting structured data, MIT press, 2007.
[33] Y. Baram, R.E. Yaniv, K. Luz, Online choice of active learning algorithms, J. Mach. Learn. Res. 5 (2004) 255–291.
[34] B. Barbaro, Tuning hyperparameters for online learning. Ph.D. thesis. Case Western Reserve University, 2018. .
[35] A.G. Barto, T.G. Dietterich, Reinforcement learning and its relationship to supervised learning, Handbook of learning and approximate dynamic programming 2 (2004) 47.
[36] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) 2399–2434.
[37] S. Ben-David, E. Kushilevitz, Y. Mansour, Online learning versus offline learning, Mach. Learn. 29 (1997) 45–63.
[38] P. Berkhin, A survey of clustering data mining techniques, Grouping multidimensional data. Springer (2006) 25–71.
[39] D.A. Berry, R.W. Chen, A. Zame, D.C. Heath, L.A. Shepp, Bandit problems with infinitely many arms, Ann. Stat. (1997) 2103–2116.
[40] A. Beygelzimer, F. Orabona, C. Zhang, Efficient online bandit multiclass learning with $\sqrt{T}$regret, in: International Conference on Machine Learning, 2017.
[41] V. Bhatnagar, S. Kaur, S. Chakravarthy, Clustering data streams using grid-based synopsis, Knowl. Inf. Syst. 41 (2014) 127–152.
[42] H. Bhatt, R. Singh, M. Vatsa, N. Ratha, Improving cross-resolution face matching using ensemble based co-transfer learning, 2014..
[43] H.S. Bhatt, R. Singh, M. Vatsa, N. Ratha, Matching cross-resolution face images using co-transfer learning, in: Image Processing (ICIP), 2012 19th IEEE International Conference on IEEE, 2012, pp. 1453–1456.
[44] M. Biesialska, K. Biesialska, M.R. Costa-jussà, Continual lifelong learning in natural language processing: A survey, in: Proceedings of the 28th International Conference on Computational Linguistics, 2020, pp. 6523–6541.
[45] A. Blum, On-line algorithms in machine learning, Springer, 1998.

[46] A.P. Boedihardjo, C.T. Lu, F. Chen, A framework for estimating complex probability density structures in data streams, in: Proceedings of the 17th ACM conference on Information and knowledge management ACM, 2008, pp. 619–628.
[47] A. Borodin, R. El-Yaniv, V. Gogan, Can we learn to beat the best stock, Advances in Neural Information Processing Systems (2004) 345–352.
[48] L. Bottou, Online algorithms and stochastic approximations, in: D. Saad (Ed.), Online Learning and Neural Networks, Cambridge University Press, Cambridge, UK. Revised, Oct 2012, 1998a..
[49] L. Bottou, Online learning and stochastic approximations, On-line learning in neural networks 17 (1998) 142.
[50] L. Bottou, Stochastic learning, Advanced lectures on machine learning. Springer (2004) 146–168.
[51] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186..
[52] O. Bousquet, L. Bottou, The tradeoffs of large scale learning, Advances in neural information processing systems (2008) 161–168.
[53] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (2011) 1–122.
[54] Y. Bu, L. Chen, A.W.C. Fu, D. Liu, Efficient anomaly monitoring over moving object trajectory streams, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 159–168.
[55] S. Bubeck, N. Cesa-Bianchi, Regret analysis of stochastic and nonstochastic multi-armed bandit problems, Found. Trends Mach. Learn. 5 (2012) 1–122.
[56] S. Bubeck, N. Cesa-Bianchi, S.M. Kakade, et al., Towards minimax policies for online linear optimization with bandit feedback, in: COLT, 2012..
[57] S. Bubeck, R. Munos, G. Stoltz, C. Szepesvári, X-armed bandits, J. Mach. Learn. Res. 12 (2011) 1655–1695.
[58] C.J. Burges et al., Dimension reduction: a guided tour, Mach. Learn. 2 (2010) 275–365.
[59] F. Cao, M. Ester, W. Qian, A. Zhou, Density-based clustering over an evolving data stream with noise, in: SDM, SIAM, 2006, pp. 328–339..
[60] Y. Cao, H. He, H. Man, Somke: Kernel density estimation over data streams by sequences of self-organizing maps, IEEE Trans. Neural Networks Learn. Syst. 23 (2012) 1254–1268.
[61] Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai, H. Li, Learning to rank: from pairwise approach to listwise approach, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 129–136.
[62] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, D.B. Rosen, Fuzzy artmap: a neural network architecture for incremental supervised learning of analog multidimensional maps, Neural Networks IEEE Trans. 3 (1992) 698–713.
[63] G.A. Carpenter, S. Grossberg, J.H. Reynolds, Artmap: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network, Neural Networks 4 (1991) 565–588.
[64] R. Caruana, Multitask learning, Learning to learn. Springer (1998) 95–133.
[65] G. Cavallanti, N. Cesa-Bianchi, C. Gentile, Tracking the best hyperplane with a simple budget perceptron, Mach. Learn. 69 (2007) 143–167.
[66] G. Cavallanti, N. Cesa-Bianchi, C. Gentile, Linear algorithms for online multitask classification, J. Mach. Learn. Res. 11 (2010) 2901–2934.
[67] N. Cesa-Bianchi, A. Conconi, C. Gentile, Learning probabilistic linear-threshold classifiers via selective sampling, in: Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, 2003, pp. 373–387..
[68] N. Cesa-Bianchi, A. Conconi, C. Gentile, On the generalization ability of on-line learning algorithms, IEEE Trans. Inf. Theory 50 (2004) 2050–2057.
[69] N. Cesa-Bianchi, A. Conconi, C. Gentile, A second-order perceptron algorithm, SIAM J. Comput. 34 (2005) 640–668.
[70] N. Cesa-Bianchi, C. Gentile, Improved risk tail bounds for on-line algorithms, IEEE Trans. Inf. Theory 54 (2008) 386–390.
[71] N. Cesa-Bianchi, C. Gentile, F. Orabona, Robust bounds for classification via selective sampling, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML2009, 2009, pp. 121–128.
[72] N. Cesa-Bianchi, C. Gentile, L. Zaniboni, Worst-case analysis of selective sampling for linear classification, J. Mach. Learn. Res. 7 (2006) 1205–1230.
[73] N. Cesa-Bianchi, G. Lugosi, Prediction, learning, and games, Cambridge University Press, New York, NY, USA, 2006.
[74] N. Cesa-Bianchi, G. Lugosi, Combinatorial bandits, J. Comput. Syst. Sci. 78 (2012) 1404–1422.
[75] N. Cesa-Bianchi, G. Lugosi, G. Stoltz, Minimizing regret with label efficient prediction, IEEE Trans. Inf. Theory 51 (2005) 2152–2162.
[76] N. Cesa-Bianchi, O. Shamir, Efficient transductive online learning via randomized rounding, Empirical Inference. Springer (2013) 177–194.
[77] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM computing surveys (CSUR) 41 (2009) 15.
[78] Y.W. Chang, C.J. Hsieh, K.W. Chang, M. Ringgaard, C.J. Lin, Training and testing low-degree polynomial data mappings via linear svm, J. Mach. Learn. Res. 11 (2010) 1471–1490.
[79] O. Chapelle, S.S. Keerthi, Efficient algorithms for ranking with svms, Inf. Retrieval 13 (2010) 201–215.
[80] O. Chapelle, L. Li, An empirical evaluation of thompson sampling, Advances in neural information processing systems (2011) 2249–2257.
[81] C. Chatfield, Time-series forecasting, CRC Press, 2000.
[82] K. Chaudhuri, Y. Freund, D.J. Hsu, A parameter-free hedging algorithm, Advances in neural information processing systems (2009) 297–305.
[83] G. Chen, G. Chen, J. Zhang, S. Chen, C. Zhang, Beyond banditron: A conservative and efficient reduction for online multiclass prediction with bandit setting model, in: 9th IEEE International Conference on Data Mining (ICDM2009), 2009, pp. 71–80..
[84] N. Chen, S.C. Hoi, S. Li, X. Xiao, Simapp: A framework for detecting similar mobile applications by online kernel learning, in: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, ACM, 2015, pp. 305–314.
[85] N. Chen, S.C. Hoi, S. Li, X. Xiao, Mobile app tagging, in: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, ACM, 2016, pp. 63–72.
[86] W. Chen, Y. Wang, Y. Yuan, Q. Wang, Combinatorial multi-armed bandit and its extension to probabilistically triggered arms, J. Mach. Learn. Res. 17 (2016) 1–33.
[87] Y. Chen, L. Tu, Density-based clustering for real-time stream data, in: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2007, pp. 133–142.
[88] Z. Chen, Z. Fang, W. Fan, A. Edwards, K. Zhang, Cstg: An effective framework for cost-sensitive sparse online learning, in: Proceedings of the 2017 SIAM International Conference on Data Mining SIAM, 2017, pp. 759–767.
[89] A. Chernov, V. Vovk, Prediction with advice of unknown number of experts, in: Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2010, pp. 117–125.
[90] S.R. Chowdhury, A. Gopalan, On kernelized multi-armed bandits, in: International Conference on Machine Learning, 2017.
[91] W. Chu, L. Li, L. Reyzin, R.E. Schapire, Contextual bandits with linear payoff functions, in: AISTATS, 2011, pp. 208–214..
[92] M. Clements, D. Hendry, Forecasting economic time series, Cambridge University Press, 1998.
[93] R. Combes, M.S.T.M. Shahi, A. Proutiere, et al., Combinatorial bandits revisited, Advances in Neural Information Processing Systems (2015) 2116–2124.
[94] T.M. Cover, Universal portfolios, in: The Kelly Capital Growth Investment Criterion: Theory and Practice. World Scientific, 2011, pp. 181–209..
[95] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, Y. Singer, Online passive-aggressive algorithms, J. Mach. Learn. Res. 7 (2006) 551–585.
[96] K. Crammer, M. Dredze, A. Kulesza, Multi-class confidence weighted algorithms, in: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, 2009, pp. 496–504.
[97] K. Crammer, C. Gentile, Multiclass classification with bandit feedback using adaptive regularization, in: Proceedings of 28th International Conference on Machine Learning (ICML2011), 2011, pp. 273–280..
[98] K. Crammer, J.S. Kandola, Y. Singer, Online classification on a budget, in: NIPS, 2003, p. 5..
[99] K. Crammer, A. Kulesza, M. Dredze, Adaptive regularization of weight vectors, Mach. Learn. (2009) 1–33.
[100] K. Crammer, D.D. Lee, Learning via gaussian herding, Advances in neural information processing systems (2010) 451–459.
[101] K. Crammer, Y. Singer, Online ranking by projecting, Neural Comput. 17 (2005) 145–175.
[102] K. Crammer, Y. Singer, et al., Pranking with ranking., in: Nips, 2001, pp. 641–647..
[103] A. Cutkosky, K.A. Boahen, Online convex optimization with unconstrained domains and losses, Advances In Neural Information Processing Systems (2016) 748–756.
[104] A.S. Das, M. Datar, A. Garg, S. Rajaram, Google news personalization: scalable online collaborative filtering, in: Proceedings of the 16th international conference on World Wide Web, ACM, 2007, pp. 271–280.
[105] J.V. Davis, B. Kulis, P. Jain, S. Sra, I.S. Dhillon, Information-theoretic metric learning, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 209–216.
[106] O. Dekel, C. Gentile, K. Sridharan. Robust selective sampling from single and multiple teachers, in: COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27–29, 2010, pp. 346–358..
[107] O. Dekel, R. Gilad-Bachrach, O. Shamir, L. Xiao, Optimal distributed online prediction using mini-batches, J. Mach. Learn. Res. 13 (2012) 165–202.
[108] O. Dekel, P.M. Long, Y. Singer, Online multitask learning, International Conference on Computational Learning Theory, Springer (2006) 453–467.
[109] O. Dekel, S. Shalev-Shwartz, Y. Singer, The forgetron: a kernel-based perceptron on a fixed budget, in: NIPS, 2005. .
[110] T.G. Dietterichx, Machine learning for sequential data: a review, in: Structural, syntactic, and statistical pattern recognition. Springer, 2002, pp. 15–30..
[111] Y. Ding, P. Zhao, S.C. Hoi, Y.S. Ong, An adaptive gradient method for online auc maximization, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
[112] S. Disabato, M. Roveri, Learning convolutional neural networks in presence of concept drift, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8.
[113] M. Dredze, K. Crammer. Active learning with confidence, in: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers, 2008, pp. 233–236..
[114] M. Dredze, K. Crammer, F. Pereira, Confidence-weighted linear classification, in: Proceedings of the 25th international conference on Machine learning ACM, 2008, pp. 264–271.

[115] Y. Du, Z. Tan, Q. Chen, Y. Zhang, C. Wang, Homogeneous online transfer learning with online distribution discrepancy minimization, 2019, arXiv preprint arXiv:1912.13226..
[116] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (2011) 2121–2159.
[117] J. Duchi, Y. Singer, Efficient online and batch learning using forward backward splitting, J. Mach. Learn. Res. 10 (2009) 2899–2934.
[118] J.C. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (2011) 2121–2159.
[119] J.C. Duchi, S. Shalev-Shwartz, Y. Singer, A. Tewari, Composite objective mirror descent, COLT (2010) 14–26.
[120] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, Neural Networks IEEE Trans. 22 (2011) 1517–1531.
[121] T. van Erven, W.M. Koolen, Metagrad: Multiple learning rates in online learning, Advances in Neural Information Processing Systems (2016) 3666–3674.
[122] T. Evgeniou, M. Pontil, Regularized multi–task learning, in: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2004, pp. 109–117.
[123] J. Farquhar, D. Hardoon, H. Meng, J.S. Shawe-taylor, S. Szedmak, Two view learning: Svm-2k, theory and practice, Advances in neural information processing systems (2006) 355–362.
[124] J. Feng, H. Xu, S. Mannor, S. Yan, Online pca for contaminated data, Advances in Neural Information Processing Systems (2013) 764–772.
[125] A. Fiat, G. Woeginger, Online algorithms: The state of the art, Springer, Heidelberg, 1998.
[126] D.H. Fisher, Knowledge acquisition via incremental conceptual clustering, Mach. Learn. 2 (1987) 139–172.
[127] D. Fotakis, T. Lianeas, G. Piliouras, S. Skoulakis, Efficient online learning of optimal rankings: Dimensionality reduction via gradient descent, in: Advances in Neural Information Processing Systems 33, 2020.
[128] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1997) 119–139.
[129] Y. Freund, R.E. Schapire, Adaptive game playing using multiplicative weights, Games Econ. Behav. 29 (1999) 79–103.
[130] Y. Freund, R.E. Schapire, Large margin classification using the perceptron algorithm, Mach. Learn. 37 (1999) 277–296.
[131] Y. Freund, H.S. Seung, E. Shamir, N. Tishby, Selective sampling using the query by committee algorithm, Mach. Learn. 28 (1997) 133–168.
[132] D. Gabay, B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite element approximation, Comput. Math. Appl. 2 (1976) 17–40.
[133] A.A. Gaivoronski, F. Stella, Stochastic nonstationary optimization for finding universal portfolios, Ann. Oper. Res. 100 (2000) 165–188.
[134] J. Gao, J. Li, Z. Zhang, P.N. Tan, An incremental data stream clustering algorithm based on dense units detection, in: Advances in Knowledge Discovery and Data Mining. Springer, 2005, pp. 420–425..
[135] W. Gao, R. Jin, S. Zhu, Z.H. Zhou, One-pass auc optimization, in: ICML, 2013..
[136] X. Gao, S.C. Hoi, Y. Zhang, J. Wan, J. Li, Soml: Sparse online metric learning with application to image retrieval, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.
[137] X. Gao, S.C. Hoi, Y. Zhang, J. Zhou, J. Wan, Z. Chen, J. Li, J. Zhu, Sparse online learning of image similarity, ACM Trans. Intell. Syst. Technol. 8 (2017) 64.
[138] Y. Gao, Y.F. Li, S. Chandra, L. Khan, B. Thuraisingham, Towards self-adaptive metric learning on the fly, The World Wide Web Conference (2019) 503–513.
[139] L. Ge, J. Gao, H. Ngo, K. Li, A. Zhang, On handling negative transfer and imbalanced distributions in multiple source transfer learning, Stat. Anal. Data Min.: ASA Data Sci. J. 7 (2014) 254–271.
[140] L. Ge, J. Gao, A. Zhang, Oms-tl: a framework of online multiple source transfer learning, in: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, ACM, 2013, pp. 2423–2428.
[141] C. Gentile, A new approximate maximal margin classification algorithm, J. Mach. Learn. Res. 2 (2001) 213–242.
[142] B. George, Time Series Analysis: Forecasting & Control, Pearson Education India, 1994, p. 3/e.
[143] P.M. Ghari, Y. Shen, Online multi-kernel learning with graph-structured feedback, International Conference on Machine Learning, PMLR (2020) 3474–3483.
[144] J. Gittins, K. Glazebrook, R. Weber, Multi-armed bandit allocation indices, John Wiley & Sons, 2011.
[145] J.C. Gittins, Bandit processes and dynamic allocation indices, J. R. Stat. Soc. Ser. B (1979) 148–177.
[146] A.B. Goldberg, M. Li, X. Zhu, Online manifold regularization: a new learning setting and empirical study, Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer (2008) 393–407.
[147] A.B. Goldberg, X. Zhu, A. Furger, J.M. Xu. Oasis: Online active semi-supervised learning, in: AAAI, 2011..
[148] S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams: Theory and practice, Knowl. Data Eng. IEEE Trans. 15 (2003) 515–528.
[149] S. Guha, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams, in: Foundations of Computer Science, 2000, Proceedings. 41st Annual Symposium on, IEEE, 2000, pp. 359–366..
[150] M. Gupta, J. Gao, C.C. Aggarwal, J. Han, Outlier detection for temporal data: A survey, IEEE Trans. Knowl. Data Eng. 26 (2014) 2250–2267.
[151] L. Gyorfi, D. Schafer, Nonparametric prediction, Nato Sci. Ser. Sub Ser. III 190 (2003) 341–356.
[152] L. Györfi, F. Udina, H. Walk, Nonparametric nearest neighbor based empirical portfolio selection strategies, Stat. Decis. Int. Math. J. Stochastic Methods Models 26 (2008) 145–157.
[153] B. Han, D. Comaniciu, Y. Zhu, L.S. Davis, Sequential kernel density approximation and its application to real-time visual tracking, IEEE Trans. Pattern Anal. Mach. Intell. 30 (2008) 1186–1197.
[154] L. Hang, A short introduction to learning to rank, IEICE Trans. Inf. Syst. 94 (2011) 1854–1862.
[155] J. Hannan, Approximation to bayes risk in repeated play, Contrib. Theory Games 3 (1957) 2.
[156] S. Hao, S.C. Hoi, C. Miao, P. Zhao, Active crowdsourcing for annotation, in: Web Intelligence and Intelligent Agent Technology (WI-IAT) 2015 IEEE/WIC/ACM International Conference on, 2015, pp. 1–8.
[157] S. Hao, P. Hu, P. Zhao, S.C. Hoi, C. Miao, Online active learning with expert advice, ACM Trans. Knowl. Discov. Data 12 (2018) 1–22.
[158] S. Hao, J. Lu, P. Zhao, C. Zhang, S.C. Hoi, C. Miao, Second-order online active learning and its applications, IEEE Trans. Knowl. Data Eng. (2017).
[159] S. Hao, P. Zhao, S.C. Hoi, C. Miao, Learning relative similarity from data streams: active online learning approaches, in: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, ACM, 2015, pp. 1181–1190.
[160] S. Hao, P. Zhao, Y. Liu, S.C. Hoi, C. Miao, Online multitask relative similarity learning, International Joint Conference on Artificial Intelligence (2017).
[161] S. Hao, P. Zhao, J. Lu, S.C. Hoi, C. Miao, C. Zhang, Soal: Second-order online active learning, in: Data Mining (ICDM), 2016 IEEE 16th International Conference on IEEE, 2016, pp. 931–936.
[162] E.F. Harrington, Online ranking/collaborative filtering using the perceptron algorithm, ICML (2003) 250–257.
[163] E. Hazan, A. Agarwal, S. Kale, Logarithmic regret algorithms for online convex optimization, Mach. Learn. 69 (2007) 169–192.
[164] E. Hazan, S. Kale, Newtron: an efficient bandit algorithm for online multiclass prediction, Advances in Neural Information Processing Systems (2011) 891–899.
[165] E. Hazan, A. Rakhlin, P.L. Bartlett, Adaptive online gradient descent, Advances in Neural Information Processing Systems (2007) 65–72.
[166] E. Hazan, C. Seshadhri, Efficient learning algorithms for changing environments, in: Proceedings of the 26th annual international conference on machine learning, ACM, 2009, pp. 393–400.
[167] E. Hazan et al., Introduction to online convex optimization, Found. Trends Optim. 2 (2016) 157–325.
[168] R. Heckel, K. Ramchandran, The sample complexity of online one-class collaborative filtering, in: International Conference on Machine Learning, 2017.
[169] D. Helmbold, S. Panizza, Some label efficient learning results, in: Proceedings of the Tenth Annual Conference on Computational Learning Theory, ACM, 1997, pp. 218–230..
[170] D.P. Helmbold, R.E. Schapire, Y. Singer, M.K. Warmuth, On-line portfolio selection using multiplicative updates, Math. Finance 8 (1998) 325–347.
[171] R. Herbrich, T. Graepel, K. Obermayer, Support vector learning for ordinal regression, 1999..
[172] M. Herbster, S. Pasteris, L. Tse, Online multitask learning with long-term memory, in: Advances in Neural Information Processing Systems 33, 2020.
[173] S.C. Hoi, J. Wang, P. Zhao, Libol: a library for online learning algorithms, J. Mach. Learn. Res. 15 (2014) 495–499.
[174] S.C. Hoi, J. Wang, P. Zhao, R. Jin, Online feature selection for mining big data, in: Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, ACM, 2012, pp. 93–100..
[175] S.C.H. Hoi, R. Jin, P. Zhao, T. Yang, Online multiple kernel classification, Mach. Learn. 90 (2013) 289–316.
[176] P. Honeine, Online kernel principal component analysis: a reduced-order model, IEEE Trans. Pattern Anal. Mach. Intell. (2012) 1814–1826.
[177] R. Hong, A. Chandra, Dlion: decentralized distributed deep learning in micro-clouds, in: 11th {*USENIX*} Workshop on Hot Topics in Cloud Computing (HotCloud 19), 2019. .
[178] J. Hu, H. Yang, I. King, M.R. Lyu, A.M.C. So, Kernelized online imbalanced learning with fixed budgets, in: AAAI, 2015, pp. 2666–2672..
[179] D. Huang, J. Zhou, B. Li, S.C. Hoi, S. Zhou, Robust median reversion strategy for on-line portfolio selection, in: Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, AAAI Press, 2013, pp. 2006–2012.
[180] D. Huang, Y. Zhu, B. Li, S. Zhou, S.C. Hoi, Semi-universal portfolios with transaction costs, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[181] P. Jain, B. Kulis, I.S. Dhillon, K. Grauman, Online metric learning and fast similarity search, Advances in neural information processing systems (2009) 761–768.
[182] S.I. Jang, Online passive-aggressive total-error-rate minimization, 2020, arXiv preprint arXiv:2002.01771..
[183] R. Jenatton, J. Huang, C. Archambeau, Adaptive algorithms for online convex optimization with long-term constraints, NIPS (2016).
[184] R. Jézéquel, P. Gaillard, A. Rudi, Efficient online learning with kernels for adversarial large scale problems, Advances in Neural Information Processing Systems (2019) 9432–9441.

[185] C. Jia, C. Tan, A. Yong, A grid and density-based clustering algorithm for processing data stream, in: Genetic and Evolutionary Computing, 2008. WGEC'08. Second International Conference on, IEEE, 2008, pp. 517–521..

[186] L. Jie, F. Orabona, M. Fornoni, B. Caputo, N. Cesa-bianchi, Om-2: An online multi-class multi-kernel learning algorithm, in: Proc. of the 4th IEEE Online Learning for Computer Vision Workshop, 2010.

[187] R. Jin, S.C.H. Hoi, T. Yang, Online multiple kernel learning: algorithms and mistake bounds, in: Algorithmic Learning Theory, 21st International Conference, ALT 2010, Canberra, Australia, October 6–8, 2010. Proceedings, 2010, pp. 390–404..

[188] R. Jin, S. Wang, Y. Zhou, Regularized distance metric learning: Theory and algorithm, Advances in neural information processing systems (2009) 862–870.

[189] D. Johnson, S. Levesque, T. Zhang, Interactive machine learning system for automated annotation of information in text. US Patent App. 10/630,854, 2003..

[190] K.S. Jun, A. Bhargava, R. Nowak, R. Willett, Scalable generalized linear bandits: Online computation and hashing, Advances in Neural Information Processing Systems, 2017.

[191] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: A survey, J. Artif. Intell. Res. (1996) 237–285.

[192] S.M. Kakade, S. Shalev-Shwartz, A. Tewari, Efficient bandit algorithms for online multiclass prediction, ICML (2008) 440–447.

[193] S.M. Kakade, A. Tewari, On the generalization ability of online strongly convex programming algorithms, Advances in Neural Information Processing Systems (2009) 801–808.

[194] A.T. Kalai, S. Vempala, Efficient algorithms for online decision problems, J. Comput. Syst. Sci. 71 (2005) 291–307.

[195] S. Kale, Z. Karnin, T. Liang, D. Pál, Adaptive feature selection: Computationally efficient online sparse linear regression under rip, in: International Conference on Machine Learning, 2017.

[196] P. Kar, B.K. Sriperumbudur, P. Jain, H.C. Karnick, On the generalization ability of online learning algorithms for pairwise loss functions, in: ICML, 2013..

[197] M.N. Katehakis, A.F. Veinott Jr, The multi-armed bandit problem: decomposition and computation, Math. Oper. Res. 12 (1987) 262–268.

[198] L. Kaufman, P.J. Rousseeuw, Clustering large applications (program clara), Finding groups in data: an introduction to cluster analysis (2008) 126–146.

[199] E. Kaufmann, O. Cappé, A. Garivier, On bayesian upper confidence bounds for bandit problems, Artificial Intelligence and Statistics (2012) 592–600.

[200] J.L. Kelly Jr, A new interpretation of information rate, in: The Kelly Capital Growth Investment Criterion: Theory and Practice, World Scientific, 2011, pp. 25–34.

[201] Z.A. Khan, S. Zubair, N.I. Chaudhary, M.A.Z. Raja, F.A. Khan, N. Dedovic, Design of normalized fractional sgd computing paradigm for recommender systems, Neural Comput. Appl. (2019) 1–18.

[202] J. Kivinen, A.J. Smola, R.C. Williamson, Online learning with kernels, Signal Processing, IEEE Transactions on 52 (2004) 2165–2176.

[203] J. Kivinen, M.K. Warmuth, Additive versus exponentiated gradient updates for linear prediction, in: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing (STOC'95), 1995, pp. 209–218..

[204] R.D. Kleinberg, Nearly tight bounds for the continuum-armed bandit problem, Advances in Neural Information Processing Systems (2005) 697–704.

[205] R.D. Kleinberg, Online decision problems with large strategy sets. Ph.D. thesis. Massachusetts Institute of Technology, 2005b. .

[206] M. Kloft, P. Laskov, Security analysis of online centroid anomaly detection, J. Mach. Learn. Res. 13 (2012) 3681–3724.

[207] A. Kobren, N. Monath, A. Krishnamurthy, A. McCallum, An online hierarchical algorithm for extreme clustering, 2017, arXiv preprint arXiv:1704.01858..

[208] W.M. Koolen, T. Van Erven, Second-order quantile methods for experts and combinatorial games, Conference on Learning Theory (2015) 1155–1175.

[209] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer (2009) 30–37.

[210] P. Kranen, I. Assent, C. Baldauf, T. Seidl, The clustree: indexing micro-clusters for anytime stream mining, Knowledge and information systems 29 (2011) 249–272.

[211] W. Krauth, M. Mézard, Learning algorithms with optimal stability in neural networks, J. Phys. A: Math. Gen. 20 (1987) L745.

[212] M. Kristan, A. Leonardis, D. Skocaj, Multivariate online kernel density estimation with gaussian kernels, Pattern Recogn. 44 (2011) 2630–2642.

[213] S.I. Ktena, A. Tejani, L. Theis, P.K. Myana, D. Dilipkumar, F. Huszár, S. Yoo, W. Shi, Addressing delayed feedback for continuous training with neural networks in ctr prediction, in: Proceedings of the 13th ACM Conference on Recommender Systems, 2019, pp. 187–195.

[214] A. Kumar, H. Daumé III, Learning task grouping and overlap in multi-task learning, in: Proceedings of the 29th International Conference on Machine Learning Omnipress, 2012, pp. 1723–1730.

[215] D. Kuzmin, M.K. Warmuth, Online kernel pca with entropic matrix updates, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 465–472.

[216] T.L. Lai, H. Robbins, Asymptotically efficient adaptive allocation rules, Adv. Appl. Math. 6 (1985) 4–22.

[217] J. Langford, L. Li, A. Strehl, Vowpal wabbit online learning project, 2007..

[218] J. Langford, L. Li, T. Zhang, Sparse online learning via truncated gradient, J. Mach. Learn. Res. 10 (2009) 777–801.

[219] J. Langford, T. Zhang, The epoch-greedy algorithm for multi-armed bandits with side information, NIPS (2008) 817–824.

[220] M.H. Law, A.K. Jain, Incremental nonlinear dimensionality reduction by manifold learning, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006) 377–391.

[221] T. Le, T. Nguyen, V. Nguyen, D. Phung, Dual space gradient descent for online learning, Advances In Neural Information Processing Systems (2016) 4583–4591.

[222] Y.A. LeCun, L. Bottou, G.B. Orr, K.R. Müller, Efficient backprop, in: Neural Networks: Tricks of the Trade, Springer, 1998, pp. 9–48..

[223] K.Y. Levy, Online to offline conversions and adaptive minibatch sizes, in: Advances in Neural Information Processing Systems, 2017..

[224] B. Li, Online portfolio selection, Ph.D. thesis, Nanyang Technological University, 2013. .

[225] B. Li, S.C. Hoi, On-line portfolio selection with moving average reversion, 2012, arXiv preprint arXiv:1206.4626..

[226] B. Li, S.C. Hoi, Online portfolio selection: A survey, ACM Comput. Surveys 46 (2014) 35.

[227] B. Li, S.C. Hoi, V. Gopalkrishnan, Corn: Correlation-driven nonparametric learning approach for portfolio selection, ACM Trans. Intell. Syst. Technol. 2 (2011) 21.

[228] B. Li, S.C. Hoi, D. Sahoo, Z.Y. Liu, Moving average reversion strategy for on-line portfolio selection, Artif. Intell. 222 (2015) 104–123.

[229] B. Li, S.C. Hoi, P. Zhao, V. Gopalkrishnan, Confidence weighted mean reversion strategy for on-line portfolio selection, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 434–442.

[230] B. Li, S.C. Hoi, P. Zhao, V. Gopalkrishnan, Confidence weighted mean reversion strategy for online portfolio selection, ACM Transactions on Knowledge Discovery from Data (TKDD) 7 (2013) 4.

[231] B. Li, S.C.H. Hoi, Online Portfolio Selection: Principles and Algorithms, Crc Press, 2015.

[232] B. Li, D. Sahoo, S.C. Hoi, Olps: a toolbox for on-line portfolio selection, J. Mach. Learn. Res. 17 (2016) 1–5.

[233] B. Li, J. Wang, D. Huang, S.C. Hoi, Transaction cost optimization for online portfolio selection, Quantitative Finance (2017) 1–14.

[234] B. Li, P. Zhao, S.C. Hoi, V. Gopalkrishnan, Pamr: Passive aggressive mean reversion strategy for portfolio selection, Mach. Learn. 87 (2012) 221–258.

[235] C.J. Li, Z. Wang, H. Liu, Online ica: Understanding global dynamics of nonconvex optimization via diffusion processes, Advances in Neural Information Processing Systems (2016) 4967–4975.

[236] G. Li, S.C. Hoi, K. Chang, R. Jain, Micro-blogging sentiment detection by collaborative online learning, IEEE Intl. Conference on Data Mining, IEEE (2010) 893–898.

[237] G. Li, S.C. Hoi, K. Chang, W. Liu, R. Jain, Collaborative online multitask learning, IEEE Trans. Knowl. Data Eng. 26 (2014) 1866–1876.

[238] G. Li, Y. Shen, P. Zhao, X. Lu, J. Liu, Y. Liu, S.C. Hoi, Detecting cyberattacks in industrial control systems using online learning algorithms, Neurocomputing 364 (2019) 338–348.

[239] G. Li, P. Zhao, X. Lu, J. Liu, Y. Shen, Data analytics for fog computing by distributed online learning with asynchronous update, in: ICC 2019-2019 IEEE International Conference on Communications (ICC), IEEE, 2019b. pp. 1–6. .

[240] K. Li, X. Zhou, F. Lin, W. Zeng, G. Alterovitz, Deep probabilistic matrix factorization framework for online collaborative filtering, IEEE Access 7 (2019) 56117–56128.

[241] K. Li, X. Zhou, F. Lin, W. Zeng, B. Wang, G. Alterovitz, Sparse online collaborative filtering with dynamic regularization, Inf. Sci. 505 (2019) 535–548.

[242] L. Li, W. Chu, J. Langford, R.E. Schapire, A contextual-bandit approach to personalized news article recommendation, in: Proceedings of the 19th international conference on World wide web ACM, 2010, pp. 661–670.

[243] L. Li, Y. Lu, D. Zhou, Provable optimal algorithms for generalized linear contextual bandits, in: International Conference on Machine Learning, 2017.

[244] Y. Li, P.M. Long, The relaxed online maximum margin algorithm, Mach. Learn. 46 (2002) 361–387.

[245] Y. Li, M. Yang, Z. Zhang, Multi-view representation learning: A survey from shallow methods to deep methods, 2016c, arXiv preprint arXiv:1610.01206..

[246] Y. Li, H. Zaragoza, R. Herbrich, J. Shawe-Taylor, J. Kandola, The perceptron algorithm with uneven margins, ICML (2002) 379–386.

[247] L. Li-xiong, K. Jing, G. Yun-fei, H. Hai, A three-step clustering algorithm over an evolving data stream, in: Intelligent Computing and Intelligent Systems, 2009, ICIS 2009, IEEE International Conference on, IEEE, 2009, pp. 160–164..

[248] N.Y. Liang, G.B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, Neural Netw. IEEE Trans. 17 (2006) 1411–1423.

[249] K.P. Lin, M.S. Chen, Efficient kernel approximation for large-scale support vector machine classification, in: Proceedings of the Eleventh SIAM International Conference on Data Mining SIAM, 2011, pp. 211–222.

[250] X. Lin, W. Zhang, M. Zhang, W. Zhu, J. Pei, P. Zhao, J. Huang, Online compact convexified factorization machine, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 1633–1642.

[251] G. Ling, H. Yang, I. King, M.R. Lyu, Online learning for collaborative filtering, in: Neural Networks (IJCNN), The 2012 International Joint Conference on IEEE, 2012, pp. 1–8.

[252] N. Littlestone, Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm, Mach. Learn. 2 (1988) 285–318.

[253] N. Littlestone. From on-line to batch learning, in: Proceedings of the Second Annual Workshop on Computational Learning Theory, COLT 1989, Santa Cruz, CA, USA, July 31 - August 2, 1989, pp. 269–284..
[254] N. Littlestone, M.K. Warmuth, The weighted majority algorithm, in: 30th Annual Symposium on Foundations of Computer Science, 1989, pp. 256–261.
[255] N. Littlestone, M.K. Warmuth, The weighted majority algorithm, Inf. Comput. 108 (1994) 212–261.
[256] C. Liu, S.C. Hoi, P. Zhao, J. Sun, Online arima algorithms for time series prediction, 2016a..
[257] C. Liu, S.C. Hoi, P. Zhao, J. Sun, E.P. Lim, Online adaptive passive-aggressive methods for non-negative matrix factorization and its applications, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, ACM, 2016, pp. 1161–1170.
[258] C. Liu, T. Jin, S.C. Hoi, P. Zhao, J. Sun, Collaborative topic regression for online recommender systems: an online and bayesian approach, Mach. Learn. 106 (2017) 651–670.
[259] N.N. Liu, M. Zhao, E. Xiang, Q. Yang, Online evolutionary collaborative filtering, in: Proceedings of 4th ACM conference on Recommender systems, 2010, pp. 95–102.
[260] Y.W. Liyanage, D.S. Zois, C. Chelmis, On-the-fly joint feature selection and classification, 2020, arXiv preprint arXiv:2004.10245..
[261] J. Lu, S. Hoi, J. Wang, Second order online collaborative filtering, Asian Conference on Machine Learning (2013) 325–340.
[262] J. Lu, S.C. Hoi, J. Wang, P. Zhao, Z.Y. Liu, Large scale online kernel learning, J. Mach. Learn. Res. (2015).
[263] J. Lu, D. Sahoo, P. Zhao, S.C. Hoi, Sparse passive-aggressive learning for bounded online kernel methods, ACM Trans. Intell. Syst. Technol. 9 (2018) 45.
[264] J. Lu, P. Zhao, S.C. Hoi, Online passive aggressive active learning and its applications, in: The 6th Asian Conference on Machine Learning (ACML2014), 2014.
[265] J. Lu, P. Zhao, S.C. Hoi, Online passive-aggressive active learning, Mach. Learn. 103 (2016) 141–183.
[266] J. Lu, P. Zhao, S.C. Hoi, Online sparse passive aggressive learning with kernels, in: Proceedings of the 2016 SIAM International Conference on Data Mining SIAM, 2016, pp. 675–683.
[267] H. Luo, A. Agarwal, N. Cesa-Bianchi, J. Langford, Efficient second order online learning by sketching, Advances in Neural Information Processing Systems (2016) 902–910.
[268] H. Luo, R.E. Schapire, Achieving all with no parameters: Adanormalhedge, Conference on Learning Theory (2015) 1286–1304.
[269] S. Magureanu, R. Combes, A. Proutière, Lipschitz bandits: regret lower bound and optimal algorithms, in: Proceedings of The 27th Conference on Learning Theory, COLT 2014, Barcelona, Spain, June 13–15, 2014, pp. 975–999..
[270] A.F.T. Martins, N.A. Smith, E.P. Xing, P.M.Q. Aguiar, M.A.T. Figueiredo, Online learning of structured predictors with multiple kernels, J. Mach. Learn. Res. 15 (2011) 507–515.
[271] B.C. May, N. Korda, A. Lee, D.S. Leslie, Optimistic bayesian sampling in contextual-bandit problems, J. Mach. Learn. Res. 13 (2012) 2069–2106.
[272] H.B. McMahan, M.J. Streeter, Tighter bounds for multi-armed bandits with expert advice, in: COLT, 2009..
[273] A. Mejer, K. Crammer, Confidence in structured-prediction using confidence-weighted models, in: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2010, pp. 971–981..
[274] R.S. Michalski, I. Mozetic, J. Hong, N. Lavrac, The multi-purpose incremental learning system aq15 and its testing application to three medical domains, Proc. AAAI 1986 (1986) 1–041.
[275] I. Mitliagkas, C. Caramanis, P. Jain, Memory limited, streaming pca, Advances in Neural Information Processing Systems (2013) 2886–2894.
[276] J.F. Mota, J.M. Xavier, P.M. Aguiar, M. Püschel, D-admm: A communication-efficient distributed algorithm for separable optimization, IEEE Trans. Signal Process. 61 (2013) 2718–2723.
[277] M. Mundt, Y.W. Hong, I. Pliushch, V. Ramesh, A wholistic view of continual learning with deep neural networks, in: Forgotten lessons and the bridge to active and open world learning, 2020, arXiv preprint arXiv:2009.01797.
[278] K. Murugesan, H. Liu, J. Carbonell, Y. Yang, Adaptive smoothed online multi-task learning, Advances in Neural Information Processing Systems (2016) 4296–4304.
[279] Y. Nesterov, Primal-dual subgradient methods for convex problems, Mathematical programming 120 (2009) 221–259.
[280] T.D. Nguyen, T. Le, H. Bui, D. Phung, Large-scale online kernel learning with random feature reparameterization, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17), 2017, pp. 2543–2549..
[281] T.T. Nguyen, K. Chang, S.C. Hui, Two-view online learning, Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer (2012) 74–85.
[282] X. Nie, M. Fan, X. Huang, W. Yang, B. Zhang, X. Ma, Online semisupervised active classification for multiview polsar data, IEEE Trans. Cybern. (2020).
[283] H. Ning, J. Zhang, T.T. Feng, E.K.w. Chu, T. Tian, Control-based algorithms for high dimensional online learning. Journal of the Franklin Institute 357 (2020) 1909–1942. .
[284] N. Nisan, T. Roughgarden, E. Tardos, V.V. Vazirani, Algorithmic game theory, vol. 1, Cambridge University Press Cambridge, 2007.
[285] A.B. Novikoff, On convergence proofs for perceptrons (Technical Report), STANFORD RESEARCH INST, Menlo Park CA, 1963.
[286] I. Ntoutsi, A. Zimek, T. Palpanas, P. Kröger, H.P. Kriegel, Density-based projected clustering over high dimensional data streams., in: SDM, SIAM, 2012, pp. 987–998..
[287] L. O'callaghan, A. Meyerson, R. Motwani, N. Mishra, S. Guha, Streaming-data algorithms for high-quality clustering, in: IEEE 29th International Conference on Data Engineering (ICDE), 2002, pp. 0685–0685. .
[288] F. Orabona, N. Cesa-Bianchi, Better algorithms for selective sampling, in: Proc. 28th International Conference on Machine Learning (ICML2011), 2011, pp. 433–440..
[289] F. Orabona, K. Crammer, New adaptive algorithms for online classification, Advances in neural information processing systems (2010) 1840–1848.
[290] F. Orabona, J. Keshet, B. Caputo, Bounded kernel-based online learning, J. Mach. Learn. Res. 10 (2009) 2643–2666.
[291] M. Ormos, A. Urbán, Performance analysis of log-optimal portfolio strategies with transaction costs, Quantitative Finance 13 (2013) 1587–1597.
[292] S.J. Pan, Q. Yang, A survey on transfer learning, Knowledge and Data Engineering, IEEE Transactions on 22 (2010) 1345–1359.
[293] G.I. Parisi, R. Kemker, J.L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: a review, 2018, arXiv preprint arXiv:1802.07569..
[294] Q. Pham, D. Sahoo, C. Liu, S.C. Hoi, Bilevel continual learning, 2020, arXiv preprint arXiv:2007.15553..
[295] J. Platt, A resource-allocating network for function interpolation, Neural computation 3 (1991) 213–225.
[296] J. Platt et al., Fast training of support vector machines using sequential minimal optimization. Advances in kernel methods support vector learning 3, 1999..
[297] G.C. Poggio, in: Incremental and decremental support vector machine learning, in: Advances in Neural Information Processing Systems 13 Proceedings of the 2000 Conference, MIT Press, 2001, p. 409.
[298] R. Polikar, L. Upda, S.S. Upda, V. Honavar, Learn++: An incremental learning algorithm for supervised neural networks, Syst. Man Cybern. Part C: Appl. Rev. IEEE Trans. 31 (2001) 497–508.
[299] M. Pratama, W. Pedrycz, G.I. Webb, An incremental construction of deep neuro fuzzy system for continual learning of non-stationary data streams, IEEE Trans. Fuzzy Syst. (2019).
[300] M. Pratama, D. Wang, Deep stacked stochastic configuration networks for lifelong learning of non-stationary data streams, Inf. Sci. 495 (2019) 150–174.
[301] M. Pratama, C. Za'in, A. Ashfahani, Y.S. Ong, W. Ding, Automatic construction of multi-layer perceptron network from streaming examples, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 1171–1180.
[302] A. Qahtan, S. Wang, X. Zhang, Kde-track: An efficient dynamic density estimator for data streams, IEEE Trans. Knowl. Data Eng. 29 (2017) 642–655.
[303] A. Rahimi, B. Recht, Random features for large-scale kernel machines, Advances in neural information processing systems (2007) 1177–1184.
[304] A. Rakhlin, Lecture notes on online learning. Notes appeared in the Statistical Learning Theory course at UC Berkeley, 2008..
[305] A. Rakhlin, K. Sridharan, A. Tewari, Online learning: Random averages, combinatorial parameters, and learnability, Advances in Neural Information Processing Systems (2010) 1984–1992.
[306] A. Rakotomamonjy, F.R. Bach, S. Canu, Y. Grandvalet, Simplemkl. J. Mach. Learn. Res. (JMLR) 11 (2008) 2491–2521.
[307] J. Read, A. Bifet, B. Pfahringer, G. Holmes, Batch-incremental versus instance-incremental learning in dynamic and evolving data, in: Advances in Intelligent Data Analysis XI. Springer, 2012, pp. 313–323..
[308] J. Ren, B. Cai, C. Hu, Clustering over data streams based on grid density and index tree, Journal of Convergence Information Technology 6 (2011).
[309] J. Ren, R. Ma, Density-based data streams clustering over sliding windows, in: Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on, IEEE, 2009, pp. 248–252..
[310] H. Robbins, Some aspects of the sequential design of experiments, Herbert Robbins Selected Papers. Springer (1985) 169–177.
[311] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, Psychological review 65 (1958) 386.
[312] D. Roth, K. Small, I. Titov, Sequential learning of classifiers for structured prediction problems, International Conference on Artificial Intelligence and Statistics (2009) 440–447.
[313] T. Roughgarden, O. Schrijvers, Online prediction with selfish experts, in: Advances In Neural Information Processing Systems, 2017..
[314] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.
[315] C. Ruiz, E. Menasalvas, M. Spiliopoulou, C-denstream: Using domain knowledge on a data stream, Discovery Science, Springer. (2009) 287–301.
[316] P. Rusmevichientong, J.N. Tsitsiklis, Linearly parameterized bandits, Mathematics of Operations Research 35 (2010) 395–411.
[317] D. Russo, B. Van Roy, An information-theoretic analysis of thompson sampling, J. Mach. Learn. Res. 17 (2016) 2442–2471.
[318] P. Ruvolo, E. Eaton, Ella: An efficient lifelong learning algorithm, International Conference on Machine Learning (2013) 507–515.
[319] A. Saha, P. Rai, H. DaumÃ, S. Venkatasubramanian, et al., Online learning of multiple tasks and their relationships, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 643–651.

[320] D. Sahoo, S.C. Hoi, B. Li, Online multiple kernel regression, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining ACM, 2014, pp. 293–302.
[321] D. Sahoo, S.C. Hoi, B. Li, Large scale online multiple kernel regression with application to time-series prediction, ACM Transactions on Knowledge Discovery from Data (TKDD) 13 (2019) 1–33.
[322] D. Sahoo, Q. Pham, J. Lu, S.C.H. Hoi, Online deep learning: Learning deep neural networks on the fly, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18, 2018, pp. 2660–2666..
[323] D. Sahoo, A. Sharma, S.C. Hoi, P. Zhao, Temporal kernel descriptors for learning with time-sensitive patterns, in: Proceedings of the 2016 SIAM International Conference on Data Mining SIAM, 2016, pp. 540–548.
[324] D. Sahoo, P. Zhao, S.C. Hoi, Cost-sensitive online multiple kernel classification, in: Proceedings of The 8th Asian Conference on Machine Learning, 2016, pp. 65–80.
[325] N.I. Sapankevych, R. Sankar, Time series prediction using support vector machines: a survey, Computational Intelligence Magazine, IEEE 4 (2009) 24–38.
[326] B. Schölkopf, R. Herbrich, A.J. Smola, A generalized representer theorem, COLT/EuroCOLT (2001) 416–426.
[327] S. Schuon, M. Durković, K. Diepold, J. Scheuerle, S. Markward, Truly incremental locally linear embedding, in: CoTeSys 1st International Workshop on Cognition for Technical Systems, 2008.
[328] D.W. Scott, Multivariate density estimation: theory, practice, and visualization, John Wiley & Sons, 2015.
[329] S.L. Scott, A modern bayesian look at the multi-armed bandit, Applied Stochastic Models in Business and Industry 26 (2010) 639–658.
[330] H.S. Seung, M. Opper, H. Sompolinsky, Query by committee, in: Proc, in: 5th annual workshop on Computational learning theory, 1992, pp. 287–294.
[331] P. Shah, A. Soni, T. Chevalier, Online ranking with constraints: A primal-dual algorithm and applications to web traffic-shaping, in: KDD, 2017..
[332] S. Shalev-Shwartz, Online learning: theory, algorithms, and applications. Ph. D. thesis. The Hebrew University of Jerusalem, 2007 .
[333] S. Shalev-Shwartz, Online learning and online convex optimization, Foundations and Trends in Machine Learning 4 (2011) 107–194.
[334] S. Shalev-Shwartz, Y. Singer, A primal-dual perspective of online learning algorithms, Mach. Learn. 69 (2007) 115–142.
[335] S. Shalev-Shwartz, Y. Singer, A.Y. Ng, Online and batch learning of pseudo-metrics, in: Proceedings of the twenty-first international conference on Machine learning ACM, 2004, p. 94.
[336] S. Shalev-Shwartz, Y. Singer, N. Srebro, A. Cotter, Pegasos: Primal estimated sub-gradient solver for svm, Mathematical programming 127 (2011) 3–30.
[337] S. Shalev-Shwartz, A. Tewari, Stochastic methods for l 1-regularized loss minimization, J. Mach. Learn. Res. (2011) 1865–1892.
[338] Y. Shi, M. Larson, A. Hanjalic, Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges, ACM Computing Surveys (CSUR) 47 (2014) 3.
[339] D.L. Silver, Q. Yang, L. Li, Lifelong machine learning systems: Beyond learning algorithms., in: AAAI Spring Symposium: Lifelong Machine Learning, 2013, p. 05.
[340] B.W. Silverman, Density estimation for statistics and data analysis, Routledge, 2018.
[341] P. Smyth, M. Welling, A.U. Asuncion, Asynchronous distributed learning of topic models, NIPS (2009) 81–88.
[342] S. Sonnenburg, V. Franc, Coffin: a computational framework for linear svms, in: Proceedings of the 27th International Conference on International Conference on Machine Learning Omnipress, 2010, pp. 999–1006.
[343] S. Sonnenburg, G. Rätsch, C. Schäfer, B. Schölkopf, Large scale multiple kernel learning, J. Mach. Learn. Res. (JMLR) 7 (2006) 1531–1565.
[344] R. Sousa, L.M. Silva, L.A. Alexandre, J. Santos, J.M. de Sá, Transfer learning: Current status, trends and challenges..
[345] E.J. Spinosa, F. de Leon, A. Ponce, J. Gama, Novelty detection with application to data streams, Intelligent Data Analysis 13 (2009) 405–422.
[346] X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, Advances in artificial intelligence 2009 (2009) 4.
[347] S. Sun, A survey of multi-view machine learning, Neural Comput. Appl. 23 (2013) 2031–2038.
[348] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT press, 1998.
[349] M. Takada, H. Fujisawa, Transfer learning via l1 regularization. Advances in Neural Information Processing Systems 33 (2020)..
[350] S.C. Tan, K.M. Ting, T.F. Liu, Fast anomaly detection for streaming data, in: IJCAI Proceedings-International Joint Conference on Artificial Intelligence, 2011, p. 1511..
[351] Y. Tao, S. Lu, From online to non-iid batch learning, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 328–337.
[352] D.K. Tasoulis, G. Ross, N.M. Adams, Visualising the cluster structure of data streams, in: Advances in Intelligent Data Analysis VII, Springer, 2007, pp. 81–92.
[353] J.B. Tenenbaum, V. De Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.
[354] W.R. Thompson, On the likelihood that one unknown probability exceeds another in view of the evidence of two samples, Biometrika 25 (1933) 285–294.
[355] T. Tommasi, F. Orabona, M. Kaboli, B. Caputo, C. Martigny, Leveraging over prior knowledge for online learning of visual categories, in: BMVC, 2012..
[356] A. Trotman, Learning to rank, Inf. Retrieval 8 (2005) 359–381.
[357] P. Tseng, On accelerated proximal gradient methods for Convex-Concave optimization, SIAM Journal on (2008), Optimization.
[358] L. Tu, Y. Chen, Stream data clustering based on grid density and attraction, ACM Transactions on Knowledge Discovery from Data (TKDD) 3 (2009) 12.
[359] Q. Tu, J. Lu, B. Yuan, J. Tang, J.Y. Yang, Density-based hierarchical clustering for streaming data, Pattern Recogn. Lett. 33 (2012) 641–645.
[360] T. Uchiya, A. Nakamura, M. Kudo, Algorithms for adversarial bandit problems with multiple plays, International Conference on Algorithmic Learning Theory, Springer. (2010) 375–389.
[361] M. Valko, B. Kveton, H. Ling, T. Daniel, Online semi-supervised learning on quantized graphs, Uncertainty in Artificial Intelligence, in, 2010.
[362] V.N. Vapnik, An overview of statistical learning theory, IEEE transactions on neural networks 10 (1999) 988–999.
[363] V.N. Vapnik, V. Vapnik, Statistical learning theory, volume 1, Wiley, New York, 1998.
[364] J. Vermorel, M. Mohri, Multi-armed bandit algorithms and empirical evaluation, in: Machine Learning: ECML 2005. Springer, 2005, pp. 437–448..
[365] V. Vovk, C. Watkins, Universal portfolio selection, in: Proceedings of the Eleventh Annual Conference on Computational Learning Theory, ACM, 1998, pp. 12–23..
[366] J. Wan, P. Wu, S.C. Hoi, P. Zhao, X. Gao, D. Wang, Y. Zhang, J. Li, Online learning to rank for content-based image retrieval, in: IJCAI, 2015, pp. 2284–2290..
[367] L. Wan, W.K. Ng, X.H. Dang, P.S. Yu, K. Zhang, Density-based clustering of data streams at multiple resolutions, ACM Transactions on Knowledge Discovery from Data (TKDD) 3 (2009) 14.
[368] C. Wang, Y. Lu, The scaling limit of high-dimensional online independent component analysis, Advances in Neural Information Processing Systems (2017) 6638–6647.
[369] D. Wang, P. Wu, P. Zhao, S.C. Hoi, A framework of sparse online learning and its applications, 2015a, arXiv preprint arXiv:1507.07146..
[370] D. Wang, P. Wu, P. Zhao, Y. Wu, C. Miao, S.C. Hoi, High-dimensional data stream classification via sparse online learning, in: Data Mining (ICDM), 2014 IEEE International Conference on IEEE, 2014, pp. 1007–1012.
[371] H. Wang, A. Banerjee, Online alternating direction method, in: Proceedings of the 29th International Conference on Machine Learning, ICML 2012 June 26 - July 1, 2012. Edinburgh, Scotland, UK, 2012.
[372] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining ACM, 2003, pp. 226–235.
[373] J. Wang, S.C. Hoi, P. Zhao, Z.Y. Liu, Online multi-task collaborative filtering for on-the-fly recommender systems, in: Proceedings of the 7th ACM conference on Recommender systems ACM, 2013, pp. 237–244.
[374] J. Wang, S.C. Hoi, P. Zhao, J. Zhuang, Z.y. Liu, Large scale online kernel classification, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, AAAI Press, 2013b, pp. 1750–1756..
[375] J. Wang, J. Wan, Y. Zhang, S.C. Hoi, Solar: Scalable Online Learning Algorithms for Ranking, ACL, 2015b..
[376] J. Wang, P. Zhao, S.C. Hoi, Exact soft confidence-weighted learning, in: Proceedings of the 29th International Coference on International Conference on Machine Learning Omnipress, 2012, pp. 107–114.
[377] J. Wang, P. Zhao, S.C. Hoi, Cost-sensitive online classification, Knowledge and Data Engineering, IEEE Transactions on 26 (2014) 2425–2438.
[378] J. Wang, P. Zhao, S.C. Hoi, Soft confidence-weighted learning, ACM Transactions on Intelligent Systems and Technology (TIST) 8 (2016) 15.
[379] J. Wang, P. Zhao, S.C. Hoi, R. Jin, Online feature selection and its applications, IEEE Trans. Knowl. Data Eng. 26 (2014) 698–710.
[380] J. Wang, P. Zhao, S.C.H. Hoi, Cost-sensitive online classification, in: 12th IEEE International Conference on Data Mining (ICDM2012), 2012b, pp. 1140–1145..
[381] S. Wang, R. Jin, H. Valizadegan, A potential-based framework for online multi-class learning with partial feedback, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 900–907.
[382] S. Wang, L.L. Minku, X. Yao, Dealing with multiple classes in online class imbalance learning, in: International Joint Conferences on Artificial Intelligence, 2016.
[383] Y. Wang, J.Y. Audibert, R. Munos, Algorithms for infinitely many-armed bandits, Advances in Neural Information Processing Systems (2009) 1729–1736.
[384] Y. Wang, Z. Jiang, X. Chen, P. Xu, Y. Zhao, Y. Lin, Z. Wang, E2-train: Training state-of-the-art cnns with over 80% energy savings, Advances in Neural Information Processing Systems (2019) 5138–5150.
[385] Y. Wang, R. Khardon, D. Pechyony, R. Jones, Generalization bounds for online learning algorithms with pairwise loss functions, in: COLT, 2012c, pp. 13–1..
[386] Z. Wang, K. Crammer, S. Vucetic, Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training, J. Mach. Learn. Res. 13 (2012) 3103–3131.
[387] Z. Wang, S. Vucetic, Tighter perceptron with improved dual use of cached data for model representation and validation, in: Neural Networks, 2009. IJCNN 2009. International Joint Conference on, IEEE, 2009, pp. 3297–3302..
[388] Z. Wang, S. Vucetic, Online passive-aggressive algorithms on a budget, Journal of Machine Learning Research - Proceedings Track 9 (2010) 908–915.

[389] M. Ware, E. Frank, G. Holmes, M. Hall, I.H. Witten, Interactive machine learning: letting users build classifiers, Int. J. Hum Comput Stud. 55 (2001) 281–292.
[390] M.K. Warmuth, D. Kuzmin, Randomized online pca algorithms with regret bounds that are logarithmic in the dimension, J. Mach. Learn. Res. 9 (2008).
[391] J. Weston, A. Bordes, L. Bottou, et al., Online (and offline) on an even tighter budget, in: Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics, 2005, pp. 413–420.
[392] C.K.I. Williams, M. Seeger, Using the nyström method to speed up kernel machines, in: T. Leen, T. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems 13, MIT Press, 2001, pp. 682–688.
[393] R.J. Williams, D. Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural computation 1 (1989) 270–280.
[394] C.H. Wu, H.H.S. Lu, H.M. Hang, Budgeted passive-aggressive learning for online multiclass classification. IEEE, Access. (2020).
[395] P. Wu, S.C. Hoi, H. Xia, P. Zhao, D. Wang, C. Miao, Online multimodal deep similarity learning with application to image retrieval, in: Proceedings of 21st ACM international conference on Multimedia, 2013, pp. 153–162.
[396] P. Wu, S.C. Hoi, P. Zhao, C. Miao, Z.Y. Liu, Online multi-modal distance metric learning with application to image retrieval, IEEE Trans. Knowl. Data Eng. 28 (2016) 454–467.
[397] Y. Wu, S.C. Hoi, C. Liu, J. Lu, D. Sahoo, N. Yu, Sol: A library for scalable online learning algorithms, Neurocomputing 260 (2017) 9–12.
[398] Y. Wu, S.C. Hoi, T. Mei, Massive-scale online feature selection for sparse ultra-high dimensional data, 2014, arXiv preprint arXiv:1409.7794..
[399] Y. Wu, S.C. Hoi, T. Mei, N. Yu, Large-scale online feature selection for ultra-high dimensional sparse data, ACM Transactions on Knowledge Discovery from Data (TKDD) 11 (2017) 48.
[400] H. Xia, S.C. Hoi, R. Jin, P. Zhao, Online multiple kernel similarity learning for visual search, IEEE Trans. Pattern Anal. Mach. Intell. 36 (2014) 536–549.
[401] H. Xia, P. Wu, S.C. Hoi, Online multi-modal distance learning for scalable multimedia retrieval, in: Proceedings of the sixth ACM international conference on Web search and data mining ACM, 2013, pp. 455–464.
[402] L. Xiao, Dual averaging method for regularized stochastic learning and online optimization, Advances in Neural Information Processing Systems (2009) 2116–2124.
[403] C. Xu, D. Tao, C. Xu, A survey on multi-view learning, 2013, arXiv preprint arXiv:1304.5634..
[404] K. Xu, Y. Li, R. Deng, K. Chen, J. Xu, Droidevolver: Self-evolving android malware detection system, in: 2019 IEEE European Symposium on Security and Privacy (EuroS&P), IEEE, 2019, pp. 47–62.
[405] Z. Xu, R. Jin, I. King, M.R. Lyu, An extended level method for efficient multiple kernel learning, in: NIPS, 2008..
[406] D. Yang, E.A. Rundensteiner, M.O. Ward, Neighbor-based pattern detection for windows over streaming data, in: Proceedings of the 12th International Conference on Extending Database Technology (EDBT), 2009, pp. 529–540.
[407] H. Yang, I. King, M.R. Lyu, Online learning for multi-task feature selection, in: Proceedings of the 19th ACM international conference on Information and knowledge management ACM, 2010, pp. 1693–1696.
[408] L. Yang, R. Jin, Distance metric learning: A comprehensive survey, Michigan State Universiy 2 (2006) 4.
[409] L. Yang, R. Jin, J. Ye, Online learning by ellipsoid method, in: Proceedings of 26th International Conference on Machine Learning, 2009, pp. 1153–1160.
[410] P. Yang, P. Zhao, X. Gao, Bandit online learning on graphs via adaptive optimization, International Joint Conferences on Artificial (2018), Intelligence.
[411] P. Yang, P. Zhao, J. Zhou, X. Gao, Confidence weighted multitask learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019, pp. 5636–5643.
[412] T. Yang, M. Mahdavi, R. Jin, J. Yi, S.C. Hoi, in: AAAI (Ed.), Online kernel selection: Algorithms and evaluations, 2012.
[413] Y. Yang, D.W. Zhou, D.C. Zhan, H. Xiong, Y. Jiang, Adaptive deep models for incremental learning: Considering capacity scalability and sustainability, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 74–82.
[414] Y. Ying, L. Wen, S. Lyu, Stochastic online auc maximization, Advances in Neural Information Processing Systems (2016) 451–459.
[415] L. Yuan-Xiang, L. Zhi-Jie, W. Feng, K. Li, Accelerated online learning for collaborative filtering and recommender systems, in: Data Mining Workshop (ICDMW), 2014 IEEE International Conference on IEEE, 2014, pp. 879–885.
[416] C. Zeng, Q. Wang, S. Mokhtari, T. Li, Online context-aware recommendation with time varying multi-arm bandit, KDD. (2016).
[417] C. Zhang, Online federated learning over decentralized networks. Ph.D. thesis, 2018..
[418] C. Zhang, S.C. Hoi, Partially observable multi-sensor sequential change detection: A combinatorial multi-armed bandit approach, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019, pp. 5733–5740.
[419] J. Zhang, H. Ning, Online kernel classification with adjustable bandwidth using control-based learning approach, Pattern Recogn. 108 (2020) 107566.
[420] J. Zhang, H. Ning, X. Jing, T. Tian, Online kernel learning with adaptive bandwidth by optimal control approach, in: IEEE Transactions on Neural Networks and Learning Systems, 2020.
[421] L. Zhang, R. Jin, C. Chen, J. Bu, X. He. Efficient online learning for large-scale sparse kernel logistic regression., in: AAAI, 2012..
[422] L. Zhang, T. Yang, R. Jin, Y. Xiao, Z.H. Zhou, Online stochastic linear optimization under one-bit feedback, International Conference on Machine Learning (2016) 392–401.
[423] T. Zhang, in: Solving large scale linear prediction problems using stochastic gradient descent algorithms, in: Proc 21th International Conference on Machine Learning (ICML'04), 2004.
[424] T. Zhang, Data dependent concentration bounds for sequential prediction algorithms, in: 18th Annual Conference on Learning Theory(COLT'05), 2005, pp. 173–187..
[425] W. Zhang, P. Zhao, W. Zhu, S.C. Hoi, T. Zhang, Projection-free distributed online learning in networks, International Conference on Machine Learning (2017) 4054–4062.
[426] X. Zhang, T. Yang, P. Srinivasan, Online asymmetric active learning with imbalanced data, KDD. (2016).
[427] P. Zhao, Kernel based online learning. Ph.D. thesis. Nanyang Technological University, 2013. .
[428] P. Zhao, S.C. Hoi, Otl: a framework of online transfer learning, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 1231–1238..
[429] P. Zhao, S.C. Hoi, Bduol: double updating online learning on a fixed budget, Machine Learning and Knowledge Discovery in Databases (2012) 810–826.
[430] P. Zhao, S.C. Hoi, Cost-sensitive online active learning with application to malicious url detection, in: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining ACM, 2013, pp. 919–927.
[431] P. Zhao, S.C. Hoi, R. Jin, Duol: A double updating approach for online learning, Advances in Neural Information Processing Systems (2009) 2259–2267.
[432] P. Zhao, S.C. Hoi, R. Jin, Double updating online learning, J. Mach. Learn. Res. 12 (2011) 1587–1615.
[433] P. Zhao, S.C. Hoi, J. Wang, B. Li, Online transfer learning, Artif. Intell. 216 (2014) 76–102.
[434] P. Zhao, S.C.H. Hoi, J. Zhuang, Active learning with expert advice, in: Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence, 2013.
[435] P. Zhao, R. Jin, T. Yang, S.C. Hoi, Online auc maximization, in: Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011b, pp. 233–240..
[436] P. Zhao, J. Wang, P. Wu, R. Jin, S.C.H. Hoi, in: ICML (Ed.), Fast bounded online gradient descent algorithms for scalable kernel-based online learning, 2012.
[437] P. Zhao, Y. Zhang, M. Wu, S.C. Hoi, M. Tan, J. Huang, Adaptive cost-sensitive online classification, IEEE Trans. Knowl. Data Eng. 31 (2018) 214–228.
[438] P. Zhao, F. Zhuang, M. Wu, X.L. Li, S.C. Hoi, Cost-sensitive online classification with adaptive regularization and its applications, in: Data Mining (ICDM), 2015 IEEE International Conference on IEEE, 2015, pp. 649–658.
[439] Y. Zheng, J. Jestes, J.M. Phillips, F. Li, in: Quality and efficiency for kernel density estimates in large data, in Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, 2013, pp. 433–444.
[440] A. Zhou, Z. Cai, L. Wei, W. Qian, M-kernel merging: Towards density estimation over data streams, DASFAA, IEEE. (2003) 285–292.
[441] L. Zhou, A survey on contextual multi-armed bandits, CoRR (2015), abs/1508.03326.
[442] X. Zhu, Semi-supervised learning literature survey, Computer Science, University of Wisconsin-Madison 2 (2006) 4.
[443] X. Zhu, Z. Ghahramani, J.D. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in: Proceedings of the 20th International conference on Machine learning (ICML-03), 2003, pp. 912–919..
[444] X. Zhu, A.B. Goldberg, Introduction to semi-supervised learning, Synthesis lectures on artificial intelligence and machine learning 3 (2009) 1–130.
[445] M. Zinkevich, Online convex programming and generalized infinitesimal gradient ascent, in: Proceedings of the Twentieth International Conference on Machine Learning(ICML 2003), 2003, pp. 928–936..
[446] M. Zoghi, T. Tunys, M. Ghavamzadeh, B. Kveton, C. Szepesvari, Z. Wen, Online learning to rank in stochastic click models, International Conference on Machine Learning (2017) 4199–4208.

Steven C.H. Hoi is currently the Managing Director of Salesforce Research Asia, and a Professor of Information Systems at Singapore Management University, Singapore. Prior to joining SMU, he was an Associate Professor with Nanyang Technological University, Singapore. He received his Bachelor degree from Tsinghua University, P.R. China, in 2002, and his Ph.D degree in computer science and engineering from The Chinese University of Hong Kong, in 2006. His research interests are machine learning and data mining and their applications to multimedia information retrieval, social media and web mining, and computational finance, etc. He has served as the Editor-in-Chief for Neurocomputing Journal, general co-chair for ACM SIGMM Workshops on Social Media, program co-chair for the fourth Asian Conference on Machine Learning, book editor for "Social Media Modeling and Computing", guest editor for ACM Transactions on Intelligent Systems and Technology. He is an IEEE Fellow and ACM Distinguished Member.

Doyen Sahoo is a Senior Research Scientist at Salesforce Research Asia. Prior to joining Salesforce, Doyen was a Research Fellow at the Living Analytics Research Center at Singapore Management University (SMU). He was also serving as Adjunct Faculty in SMU. Doyen earned his PhD in Information Systems from SMU in 2018 and B.Eng in Computer Science from Nanyang Technological University in 2012. His research interests include Online Learning, Deep Learning, Computer Vision, and he also works on applied research including AIOps, Computational Finance and Cyber Security applications. He has published over 40 articles in top tier conferences and journals including ICLR, CVPR, ACL, KDD, JMLR, etc.

Jing Lu is a senior engineer in JD.com, focusing on online advertising systems for e-commerce platforms. Prior to joining JD, she finished her Doctor's degree on Machine Learning, in School of Information System, Singapore Management University (2014-2018) and School of Computer Engineering in Nanyang Technological University (2012-2014). She is currently dedicated to the research area of personalized recommendation systems, CTR prediction in sponsored search, online learning and active learning and has published several research papers in top tier journals and conferences including, NIPS, KDD, ICCV, JMLR, TKDE, TIST etc.

Peilin Zhao is currently a Principal Researcher at Tencent AI Lab, China. Previously, he has worked at Rutgers University, Institute for Infocomm Research (I2R), Ant Group. His research interests include: Online Learning, Recommendation System, Automatic Machine Learning, Deep Graph Learning, and Reinforcement Learning etc. He has published over 100 papers in top venues, including JMLR, ICML, KDD, etc. He has been invited as a PC member, reviewer or editor for many conferences and journals, such as ICML, JMLR, etc. He received his bachelor's degree from Zhejiang University, and his Ph. D. degree from Nanyang Technological University.