Data Mining and Knowledge Discovery
https://doi.org/10.1007/s10618-022-00845-0

Human-in-the-loop handling of knowledge drift

Andrea Bontempelli[1] · **Fausto Giunchiglia**[1,2] · **Andrea Passerini**[1] · **Stefano Teso**[1]

Received: 8 October 2021 / Accepted: 21 May 2022

Abstract
We introduce and study knowledge drift (KD), a special form of concept drift that occurs in hierarchical classification. Under KD the vocabulary of concepts, their individual distributions, and the *is-a* relations between them can all change over time. The main challenge is that, since the ground-truth concept hierarchy is unobserved, it is hard to tell apart different forms of KD. For instance, the introduction of a new *is-a* relation between two concepts might be confused with changes to those individual concepts, but it is far from equivalent. Failure to identify the right kind of KD compromises the concept hierarchy used by the classifier, leading to systematic prediction errors. Our key observation is that in human-in-the-loop applications like smart personal assistants the user knows what kind of drift occurred recently, if any. Motivated by this observation, we introduce TRCKD, a novel approach that combines two *automated* stages—drift detection and adaptation—with a new *interactive* disambiguation stage in which the user is asked to refine the machine's understanding of recently detected KD. In addition, TRCKD implements a simple but effective *knowledge-aware* adaptation strategy. Our simulations show that, when the structure of the concept hierarchy drifts, a handful of queries to the user are often enough to substantially improve prediction performance on both synthetic and realistic data.

Responsible editor: Albrecht Zimmermann and Peggy Cellier.

✉ Andrea Bontempelli
andrea.bontempelli@unitn.it

Fausto Giunchiglia
fausto.giunchiglia@unitn.it

Andrea Passerini
andrea.passerini@unitn.it

Stefano Teso
stefano.teso@unitn.it

[1] Department of Information Engineering and Computer Science, University of Trento, Trento, Italy

[2] Jilin University, Changchun, China

Published online: 30 July 2022

Keywords Concept drift · Hierarchical classification · Human-in-the-loop · Interactive machine learning

1 Introduction

We are concerned with human-in-the-loop applications of hierarchical classification. Our main interest lies in smart personal assistants (PAs) that must infer the location or social context of their user from sensor data (e.g., GPS, nearby Bluetooth devices) under the constraint that the hierarchy of relevant places and people changes over time (Giunchiglia et al. 2017). In these applications, the concept hierarchy embedded into the predictor can become obsolete and has to be continually re-aligned (Stojanovic et al. 2002).

We refer to this as *knowledge drift* (KD). KD is a complex phenomenon: concepts and *is-a* relations between them may appear, disappear, and change. The main challenge is distinguishing between different kinds of KD and especially between changes to the data distribution and to the concept hierarchy. Existing approaches to concept drift make no attempt at understanding whether shifts in the observed correlations between concepts are due to changes to the hierarchy (like adding an *is-a* relation) or not (Gama et al. 2014). These leave similar footprints on the data, but confusing one for the other—and confusing different kinds of KD—entails acquiring spurious *is-a* relations or neglecting changes to the hierarchy, leading to systematic mis-predictions on future instances.

Our key observation is that, in our human-in-the-loop setting, *an expert user can identify with little effort what kind of drift occurred*, if any. Several examples are given below. Motivated by this observation, we design TRCKD (TRaCking Knowledge Drift), an approach that tackles KD by combining *automated* drift detection and adaptation with a novel, *interactive* drift disambiguation step. In particular, TRCKD maintains two windows of examples for each concept—one holds old data points and the other the most recent ones—and it detects KD by checking whether the distribution of current and past examples have diverged in distribution. The two empirical distributions are compared using the maximum mean discrepancy (MMD), a flexible and efficient kernel-based divergence (Gretton et al. 2012). Whenever it detects KD, TRCKD guesses what kind of KD occurred using a simple heuristic and presents this initial description to a knowledgeable human supervisor. The latter is then tasked with either confirming the machine's description or improving it, if necessary, according to her own understanding. Finally, in order to adapt the model to the different kinds of KD, TRCKD implements a simple but effective knowledge-aware adaptation strategy that we ground on top of *k*NN-based multi-label classifiers (Spyromitros-Xioufis et al. 2011). Our experiments show that, when changes to the structure of the concept hierarchy occur, interactive drift disambiguation and knowledge-aware adaptation are key for good performance under KD, and that asking a handful of queries to the user is often enough to achieve substantial performance improvements.

Summarizing, we:

1. Identify *knowledge drift* as a special kind of concept drift.

2. Introduce the related issue of *drift disambiguation* and identify interaction with an expert user as a natural solution.
3. Design TRCKD, an approach for handling KD that combines automated detection and adaptation with interactive disambiguation, and instantiate it on top of kNN-based classifiers by implementing a knowledge-aware adaptation strategy.
4. Compare empirically TRCKD with state-of-the-art competitors on three representative data sets.

The remainder of this paper is structured as follows. In the next Section we introduce our problem setting and define different forms of KD. Next, in Sect. 3 we present TRCKD and detail its three stages: automatic detection, interactive disambiguation, and knowledge-aware adaptation. Then, we evaluate TRCKD on three challenging data sets in Sect. 4. In Sect. 5 we position our contribution with respect to the existing literature and conclude with some final remarks and a brief discussion of promising research directions.

2 Hierarchical classification and knowledge drift

We consider learning tasks in which each instance $\mathbf{x}$ belongs to one or more concepts (classes) organized in a ground-truth hierarchy $H = (C, I)$, a directed acyclic graph in which nodes $C = \{1, \ldots, c\}$ index concepts and edges $I \subseteq C \times C$ encode *is-a* relations. Instances are labeled by indicator vectors $\mathbf{y} \in \{0, 1\}^c$, whose i-th element y^i is 1 if $\mathbf{x}$ belongs to the i-th concept in H and 0 otherwise.

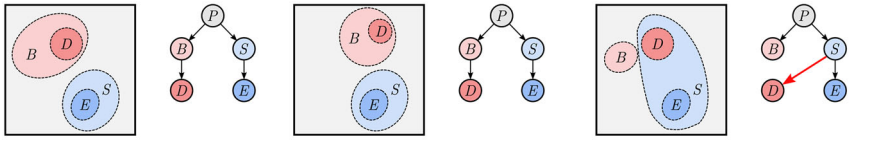
During operation, the machine receives a stream of examples $\mathbf{z}_t = (\mathbf{x}_t, \mathbf{y}_t)$ drawn from a ground-truth distribution $P_t(\mathbf{X}, \mathbf{Y})$ that is *consistent* with a corresponding ground-truth hierarchy H_t. In other words, if H_t asserts that concept j *is-a* specialization of concept i, then the probability $P_t(\mathbf{X}, \mathbf{y})$ of all labels $\mathbf{y}$ that violate this relation (i.e., $y^j = 1$ and $y^i = 0$), is zero. The goal is to learn a predictor $\hat{P}_t$ (as well as a hierarchy $\hat{H}_t$ consistent with it) that outputs high-quality predictions on future instances.[1] It goes without saying that, in order to avoid systematic prediction mistakes, the acquired hierarchy $\hat{H}_t$ must closely resemble the ground-truth H_t.

Example 1 Ann's PA receives observations $\mathbf{x}$ (like GPS coordinates, nearby Bluetooth devices) and uses them to predict that "Ann is studying at the library with Bob": "Studying", "Library", and "Bob" are concepts and the hierarchy states, among other things, that "Bob" *is-a* "Friend" of Ann's and a "Person".

What makes our setting challenging is that the (unobserved) ground-truth concept hierarchy H_t and data distribution P_t can both unexpectedly and frequently change over time $t = 1, 2, \ldots$.

Approaches for dealing with concept drift do not capture this scenario. Indeed, regular concept drift is restricted to *distribution shift*, in which the prior distribution $P_t(\mathbf{X})$ changes, and *individual* or *multi-label concept drift*, in which the conditional distribution $P_t(Y_i \mid \mathbf{X})$ of one or more concepts changes (Gama et al. 2014; Zheng

[1] We assume $\mathbf{y}$ to be given for ease of exposition. In practical applications where $\mathbf{y}$ is not given, it can be acquired using an active learning step.



(a) Pre-drift **(b)** Post concept drift **(c)** Post knowledge drift

Fig. 1 Left: Decision surface and hierarchy of a classifier for Ann's social context and five concepts: "*P*erson", "*B*oss", "*S*ubordinate", "*D*ave", and "*E*arl". Middle: Individual Concept Drift: Dave moves to a different office, so the decision surface changes but the hierarchy remains the same. Right: Knowledge Drift: Ann is promoted, "Dave" is now her subordinate. If the classifier knows that the hierarchy changed, it can transfer examples from "Dave" to "Subordinate", quickly improving its performance

et al. 2019), but the concepts themselves are not mutually constrained by a ground-truth hierarchy.

Example 2 During the semester, Ann spends most of her time studying at the library. Once the finals are over, Ann stops going to the library as often, and when she goes there she is less likely to be studying. This affects both the distribution of GPS coordinates and the conditional distribution of activities given GPS coordinates.

In stark contrast, changes to the hierarchy H_t give rise to *knowledge drift* (KD), a special form of drift in which changes to the distribution are specifically due to changes to the hierarchy, cf. Fig. 1. KD combines four types of atomic changes: *concept addition* and *removal*, which refer to the appearance of new concepts and the phasing out of obsolete concepts in C_t, respectively, and *relation addition* and *removal*, which refer to changes in the edges I_t themselves.

Example 3 While concepts like "Friend" are immutable, the specific friends that matter to Ann (which are also concepts) change over time, e.g., if Ann moves abroad. For instance, if Ann buys a vacation home, a new concept "Ann's vacation home" appears in the ground-truth hierarchy (concept addition). Conversely, if Ann's vacation home is sold, the corresponding concept is no longer meaningful and disappears (concept removal). If Ann receives a promotion and her old boss Dave becomes her subordinate, then "Dave" moves from being a child of "Boss" (relation removal) to being a child of "Subordinate" (relation addition).

Now, handling regular concept drift requires to detect changes in the data and adapt the model accordingly (Gama et al. 2014). KD, on the other hand, requires an additional step, namely to understand what concepts and relations in the hierarchy H_t were affected, if any. This *drift disambiguation* step is crucial for preventing the estimated hierarchy $\hat{H}_t$ from getting misaligned, which could in turn lead to systematic, cascading prediction errors. It is also very challenging.

To see this, consider relation addition (RA). Like all forms of KD, RA can only be identified by its effects on the data, and specifically from the correlation between the concepts that it entails. However, correlations can exist and vary independently of the hierarchy. For instance, if Ann is visiting a branch of her company in another city, she could be taken out for lunch and dinner by Mary, the branch manager. Data could

thus suggest a correlation between the concepts "Mary" and "Friend". Once Ann gets back home, she no longer hangs out with Mary, and the correlation drops dramatically. Since the machine has no sure way of telling apart RA from regular concept drift, it might wrongly add a spurious relation to its concept hierarchy—a mistake that takes plenty of examples to correct.

Similarly, concept removal (CR) implies not only that a concept i cannot occur (and should not be predicted) ever again, but also that its children are not longer attached to it. Treating CR as concept drift means that i) the conditional distribution $P_t(Y_i \mid \mathbf{X})$ is not constrained to zero, and that ii) the deleted concept might be predicted when one of its children is.[2] Naturally, multiple atomic changes can occur simultaneously, further complicating drift disambiguation.

3 Handling knowledge drift with TRCKD

Our key observation is that in many human-in-the-loop scenarios *the user can naturally disambiguate between different types of KD*. In our running example, for instance, Ann is perfectly aware that her vacation home has been recently sold, that Dave is now her subordinate and that Mary is a colleague and not a friend. It is therefore sensible to partially offload drift disambiguation to the user.

Motivated by this insight, we introduce TRCKD, a kNN-based approach for human-in-the-loop hierarchical classification under KD that combines *automated* detection and adaptation with a new, *interactive* drift disambiguation stage. Owing to their flexibility, kNN-based approaches are a popular choice for learning under drift (Gama et al. 2014) and achieve considerable performance in non-trivial learning tasks (Roseberry et al. 2019).[3] Our work builds on MW-kNN (Spyromitros-Xioufis et al. 2011), an kNN-based approach to multi-class classification under drift that adapts to change by passively forgetting old, potentially obsolete examples. TRCKD upgrades MW-kNN from multi-class to hierarchical classification and additionally integrates a sliding-window approach (Kifer et al. 2004) for drift detection. Moreover, TRCKD introduces a simple but effective *knowledge-aware* adaptation strategy specifically tailored for kNN-based classifiers (and that generalizes to other instance-based predictors).

The pseudo-code of TRCKD is listed in Algorithm 1. The algorithm takes a data set S_1 and a concept hierarchy H_1 consistent with it and uses them to train an initial classifier. Then, in each iteration $t = 1, 2, \ldots$ the machine receives a new example $\mathbf{z}_t = (\mathbf{x}_t, \mathbf{y}_t)$ and performs three steps: (1) It detects whether KD occurred, (2) It cooperates with the user to determine what concepts and relations were affected by KD, and (3) It adapts the classifier and the machine's hierarchy accordingly. We discuss these three steps in turn.

[2] The only "easy" case is concept addition, which is straightforward in our fully labeled setting and will not be considered further.

[3] While the ideas behind TRCKD do carry over to other models, a proper assessment using non-kNN architectures is outside of scope for this paper and left to future work.

Algorithm 1 The TRCKD algorithm. Inputs: initial data set S_1 and hierarchy H_1, $s := |S_1|$, window size w, threshold τ, empirical estimator $\widehat{\text{MMD}}$; $\mathbf{z}_t^i := (\mathbf{x}_t, y_t^i)$.

Fit initial classifier on S_1 and H_1, $\hat{H}_1 \leftarrow H_1$
for every concept i in $\hat{H}_1$ **do**
  $W_{\text{old}}^i \leftarrow \{\mathbf{z}_s^i, \ldots, \mathbf{z}_{s-w}^i\}$
end for
for $t = 1, 2, \ldots$ **do**
  Receive new example $\mathbf{z}_t$
  for every concept i in $\hat{H}_t$ **do**
    $W_{\text{cur}}^i \leftarrow \{\mathbf{z}_{s+t}^i, \mathbf{z}_{s+t-1}^i, \ldots, \mathbf{z}_{s+t-w}^i\}$
  end for
  if $\exists i : \widehat{\text{MMD}}(W_{\text{cur}}^i, W_{\text{old}}^i) \geq \tau$ **then**
    Illustrate detected KD to the user
    Adapt based on user's KD description
  end if
end for

3.1 Step 1: detection

For every concept i in $\hat{H}_t$, TRCKD maintains two windows of examples: W_{cur}^i holds the w most recent examples and is updated in each iteration, while W_{old}^i holds w reference (past) examples.[4] Predictions for concept i are obtained by applying kNN to W_{cur}^i.

TRCKD detects drift by looking for changes in distribution between the recent and past windows of each concept. To this end, it employs the *maximum mean discrepancy* (MMD), a discrepancy employed in hypothesis testing (Gretton et al. 2012) and domain adaptation (Zhang et al. 2013). Let P and Q be distributions over some space $\mathcal{X}$ and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a user-defined kernel. Then the MMD between P and Q relative to k is given by

$$\text{MMD}(P, Q)^2 = \mathbb{E}[k(\mathbf{a}, \mathbf{a}')] - 2\mathbb{E}[k(\mathbf{a}, \mathbf{b})] + \mathbb{E}[k(\mathbf{b}, \mathbf{b}')], \tag{1}$$

where $\mathbf{a}, \mathbf{a}'$ are drawn i.i.d. from P and $\mathbf{b}, \mathbf{b}'$ from Q. Estimating the MMD between W_{old}^i and W_{cur}^i requires to define a kernel between examples $\mathbf{z}$. TRCKD achieves this by defining two separate kernels over instances and labels k_X and k_Y and then taking their tensor product, i.e.,

$$k(\mathbf{z}, \mathbf{z}') = k((\mathbf{x}, y), (\mathbf{x}', y')) = k_X(\mathbf{x}, \mathbf{x}') \cdot k_Y(y, y'). \tag{2}$$

In our experiments, we employ a Gaussian kernel k_X for the instances and a delta kernel $k_Y(y, y') = \mathbb{1}\{y = y'\}$ for the labels.

The MMD is well-behaved, in the sense that if $P \equiv Q$ then $\text{MMD}(P, Q) = 0$ and if k is characteristic, the converse also holds (Szabó and Sriperumbudur 2017). However, unlike other well-behaved alternatives—for instance the total variation distance, the $\mathcal{A}$-distance (Kifer et al. 2004), and the mutual information (Pérez-Cruz 2009)—the MMD can be estimated efficiently (in linear time) even for high-dimensional data (Gretton

[4] TRCKD reserves 1/3 of each window to positive examples to account for class imbalance (Spyromitros-Xioufis et al. 2011).

et al. 2012). In addition, the MMD allows to select concrete examples (witness points) that illustrate the difference between the two distributions, simplifying the interaction with the user (Lloyd and Ghahramani 2015). It also performs well empirically: in our experiments, the MMD achieved a better false detection rate than ME (Jitkrittum et al. 2016), a state-of-the-art discrepancy with better discrimination power on paper.

3.2 Step 2: disambiguation

Upon detecting drift, TRCKD initiates interaction with the user by presenting a visualization of the detected KD and asking the user to verify and potentially improve a description of the detected KD.

If the KD detected by the machine identifies the right concepts—which should typically be if the drift detector is tuned well—then the user's jobs is to tell the machine whether the *relations* between the highlighted concepts have undergone drift and how. To this end, the user can modify the visualization by selecting or deselecting highlighted concepts and *is-a* edges. As long as the user is expert enough, she is likely to improve the machine's guess. A sufficiently motivated and knowledgeable user has also the option of editing any drifting concepts or relations that were not detected by the machine, providing even more guidance. Notice that even *partial* improvements to the detected KD are likely to improve future predictive performance and are in any case better than no adaptation and fully automated disambiguation.[5]

Extra context can be supplied to the user by presenting a handful of examples that summarize how the concepts affected by KD have changed. Such examples can be selected from the past and current windows of those concepts using MMD witness functions (Lloyd and Ghahramani 2015).

3.3 Step 3: adaptation

Once it receives the user's drift description, TRCKD adapts the machine's hierarchy and the windows accordingly. Here we present a simple knowledge-aware adaptation strategy for kNN-based approaches. In particular:

- For every instance of *individual concept drift* in the description, it transfers the contents of the current window of the affected concept to the past window and keeps only the u most recent examples in the former, i.e.:

$$W^i_{\text{old}} \leftarrow W^i_{\text{cur}}, \qquad W^i_{\text{cur}} \leftarrow \{\mathbf{z}^i_{s+t}, \mathbf{z}^i_{s+t-1}, \ldots, \mathbf{z}^i_{s+t-u}\} \tag{3}$$

  where u is a user-provided hyperparameter. All examples are not longer used for the classification task except for the most recent ones, which are likely drawn from the post-drift distribution and thus kept to facilitate recovery.

[5] The assumption is that the user is expert enough and therefore does not inject a lot of noise into the loop. This is a reasonable assumption to make applications like smart personal assistants.

- For *concept removal*, the past and current windows of the affected concept are deleted, that is:

$$W^i_{\text{old}} \leftarrow \emptyset, \qquad W^i_{\text{cur}} \leftarrow \emptyset \tag{4}$$

  Furthermore, all *is-a* relations between the removed concept are deleted and the children of the latter are attached to its parent. The concept will not occur and thus should not be predicted.[6] Notice that concept removal cannot be handled as concept drift, otherwise the deleted concepts would end up being predicted whenever one of its children is. The child concepts are no longer attached to it. The parent's window is reduced in size by w elements since it has to accommodate the examples of fewer child classes.
- For *relation addition*, the positive examples belonging to the child concept are copied to the ancestors' windows and the former are increased in size by w in order to take the examples of the new child. Given the child concept r:

$$W^i_{\text{old}} \leftarrow W^i_{\text{cur}}, \qquad W^i_{\text{cur}} \leftarrow W^i_{\text{cur}} \cup W^r_{\text{cur}} \tag{5}$$

  for each ancestor i. This adaptation ensures that the ancestors are predicted whenever the children is.
- For *relation removal*, the positive examples belonging to the child concept are removed from the parent's window and the latter is shrank accordingly. The child concept is also linked directly to its grand-parent. Given i and r the parent and the child respectively, then:

$$W^i_{\text{old}} \leftarrow W^i_{\text{cur}};\ W^i_{\text{cur}} = W^i_{\text{cur}} \setminus W^r_{\text{cur}} \tag{6}$$

Our experiments show that, despite its simplicity, our knowledge-aware adaptation strategy outperforms the knowledge-oblivious strategies of state-of-the-art kNN classifiers (Spyromitros-Xioufis et al. 2011; Roseberry et al. 2019). It is reasonable to expect the benefits of knowledge-aware adaptation to carry over to other classifiers, e.g., neural networks (Cao and Yang 2015).

4 Experiments

We empirically address the following research questions:

Q1 Is knowledge-aware adaptation useful for handling knowledge drift?
Q2 Does interaction with an expert user help adaptation?
Q3 Does TRCKD work in realistic, multi-drift settings?

The code of TRCKD as well as the complete experimental setup are available at: https://gitlab.com/abonte/handling-knowledge-drift.

[6] Notice that, by definition, a removed concept cannot recur unless it is added again to the hierarchy. This is handled separated via concept addition.

Competitors
We compared TRCKD against several alternatives:

- **PAW-kNN** punitive adaptive window kNN, a *state-of-the-art* multi-label approach that employs a single sliding window for all concepts to address gradual drift, and that adapts by discarding examples responsible for prediction mistakes in case of abrupt drift (Roseberry et al. 2019);
- **MW-kNN** the multi-window kNN approach of Spyromitros-Xioufis et al. (2011) designed specifically for multi-label problems that TRCKD builds on;
- **kNN 1-window** kNN with a single sliding window for all concepts that simply forgets old examples;
- **kNN** regular kNN with no adaptation.

Data sets
We ran experiments on three data sets from different domains:

- **H-STAGGER** a hierarchical version of STAGGER, a widely used synthetic data set of two-dimensional objects with three categorical attributes (shape, color, size) and labeled by drifting random formulas like "small and (green or red)" (Schlimmer and Granger 1986). H-STAGGER has 3 attributes with 4 values each and labels instances using 5 different drifting random formulas chosen to have a reasonable pos./neg. ratio. The hierarchy is created by selecting two concepts as part of a third one that acts as parent concept.
- **H-EMNIST** a data set of 28×28 handwritten digits and (uppercase, lowercase) letters (Cohen et al. 2017). The data set was converted to hierarchical classification by grouping different characters into higher level concepts, e.g., even numbers and vowels. The digits and letters are grouped in 5 concepts and the hierarchy is created as for H-STAGGER. Instances were embedded using a variational autoencoder (Kingma and Welling 2014).
- **H-20NG** a data set of newsgroup posts categorized in twenty topics.[7] The data set was converted to hierarchical classification by grouping different classes into super-topics (e.g., "religion' grouping alt.atheism, soc.religion.christian and talk.religion.misc). The documents were embedded using a pre-trained Sentence-BERT model (Reimers and Gurevych 2019) and compressed to 100 features using PCA.

All instances always belong to the *root* concept in the concept hierarchy. Each data sets is converted into a streams by sampling a sequence of examples at random. Table 1 reports, for each data sets, the number of attributes d, the number of concepts c, as well as the following three measures of annotation density taken from Zhang and Zhang (2010): the average number of positive labels per example LC, the empirical probability that a label is positive LD, and how many distinct combinations of positive categories (out of 2^c) are annotated in the data DL.

[7] From https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups.

Table 1 Data sets statistics (mean ± std. dev.) averaged over the runs of the sequential drift experiment

Name	$\|S\|$	d	c	LC	LD	DL
H-STAGGER	570	3	6	4.42 ± 0.16	0.63 ± 0.02	34.63 ± 11.89
H-EMNIST	570	10	9	3.50 ± 0.04	0.44 ± 0.00	55.38 ± 5.57
H-20NG	570	100	6	2.19 ± 0.00	0.31 ± 0.00	8.5 ± 0.5

$|S|$: number of instances, d: number of attributes, c: number of labels, LC: label cardinality, LD: label density, DL: distinct label set. The metrics are averaged on 8 runs and refer to the experiment for Q3

4.1 Experimental details

All experiments were run on a machine with eight 2.8 GHz CPUs and 32 GiB RAM. Each experiment was run ten times by randomizing the choice of examples in the stream: 2 runs were used for hyperparameter selection and 8 for evaluation. The plots report the average and the standard error over the 8 runs. All methods received exactly the same sequences of examples.

The results for concept drift are independent of our knowledge-handling strategy, so our evaluation focuses on concept deletion, relation addition, and relation removal. In these three cases, drift is injected into the stream by removing a random concept from the available ones, adding a random relation, and removing a random relation, respectively. KD starts after all competitors approximately reach their peak performance, namely after 100 iterations for H-STAGGER and H-EMNIST, and 170 for H-20NG.

Performance was measured in terms of micro F_1 score on a hold-out test set (of size 64 for H-STAGGER and 200 for H-EMNIST and H-20NG) randomly selected before each run and shared by all competitors. The micro F_1 score we use considers the sparsity of positive annotations that is characteristic of hierarchical classification; that is, most concepts are negative most of the time. Letting $\{(\mathbf{x}_i, \mathbf{y}_i) : i = 1, \ldots, n\}$ be the examples in the *test set*, $\hat{\mathbf{y}}_i$ their predictions and y_i^j the jth element of y_i (which is 1 if $\mathbf{x}_i$ belong to the jth concept and 0 otherwise), the micro F_1 for multi-label classification is defined as follows (Sorower 2010):

$$F_{1\text{-micro}} = \frac{2 \sum_{j=1}^{c} \sum_{i=1}^{n} y_i^j \hat{y}_i^j}{\sum_{j=1}^{c} \sum_{i=1}^{n} y_i^j + \sum_{j=1}^{c} \sum_{i=1}^{n} \hat{y}_i^j}$$

where j iterates over concepts and i over examples.

User replies were simulated by an oracle that always answers correctly to disambiguation queries. More specifically, the user confirms that a drift occurred if and only if the concept detected as drifting by MMD has actually undergone drift, i.e., the concept has been removed, has drifted or is the child or parent of an added/removed relation.

Table 2 Hyperparameter values

RQ	Drift	τ	TRCKD k	PAW-kNN k
H-STAGGER				
Q1–Q2	CD	0.04	3	3
	CR	0.04	11	3
	RA	0.04	11	3
	RR	0.04	11	3
Q3	All	0.04	11	3
H-EMNIST				
Q1–Q2	CD	0.04	5	3
	CR	0.05	3	3
	RA	0.05	3	3
	RR	0.04	3	3
Q3	All	0.05	3	3
H-20NG				
Q1–Q2	CD	0.05	3	3
	CR	0.05	3	3
	RA	0.04	3	3
	RR	0.04	3	3
Q3	All	0.05	3	3

CD is concept drift, CR concept removal, RA relation addition, and RR relation removal

4.2 Hyperparameters

The window size of all window-based methods was set to $w = 200$. This value enables these approaches to achieve the same performance (micro F_1) as kNN when *no* drift is present. To speed up detection, in TRCKD the MMD of each concept i is computed on the 70 most recent examples in W^i_{cur} only. This choice does not sacrifice reliability of detection.

The MMD threshold τ used by TRCKD, the number of neighbors k used by all competitors are selected in two independent runs by optimizing the micro F_1. τ is selected from $\{0.4, 0.5\}$ as these values were frequently observed to indicate drift in preliminary experiments. k was selected from $\{3, 5, 11\}$ and it was chosen independently for TRCKD plus its variants (including MW-kNN) and for PAW-kNN. Table 2 reports the values used in each experiment of these hyperparamters. All other hyperparameters were kept fixed across experiments. The penalty ratio of PAW-kNN was set to $p = 1$ as suggested in Roseberry et al. (2019), while the minimum and maximum window sizes were set to $m_{min} = 50$ for H-STAGGER and H-EMNIST and 80 for H-20NG data set, and $m_{max} = 200$ respectively. The number of examples retained in W_{cur} in case of concept drift adaptation was set to $u = 10$. The distribution ratio parameter of TRCKD and MW-kNN was set to $r = 2/3$ as in Spyromitros-Xioufis et al. (2011).

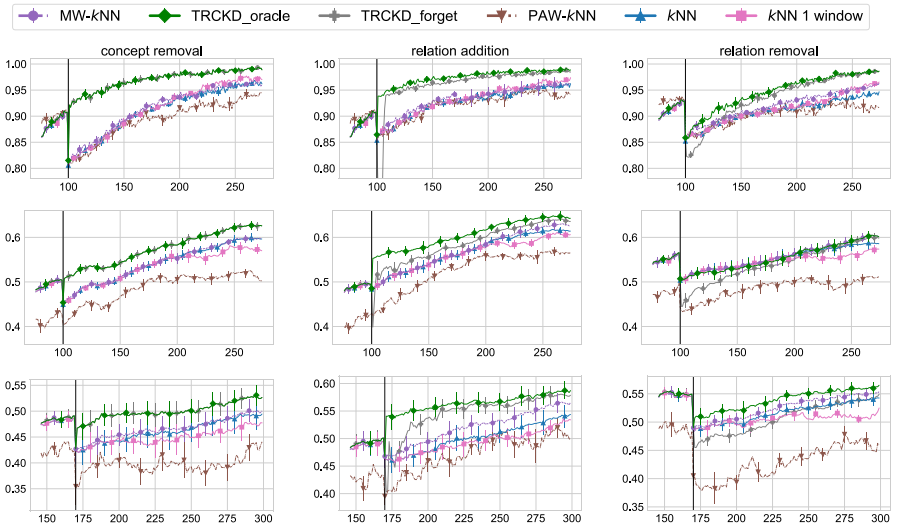


Fig. 2 Comparison in terms of micro F_1 between TRCKD and standard forgetting strategies for kNN-based methods. Top to bottom: results for H-STAGGER, H-EMNIST and H-20NG. Left to right: concept removal, relation addition, and relation removal. Error bars indicate std. error

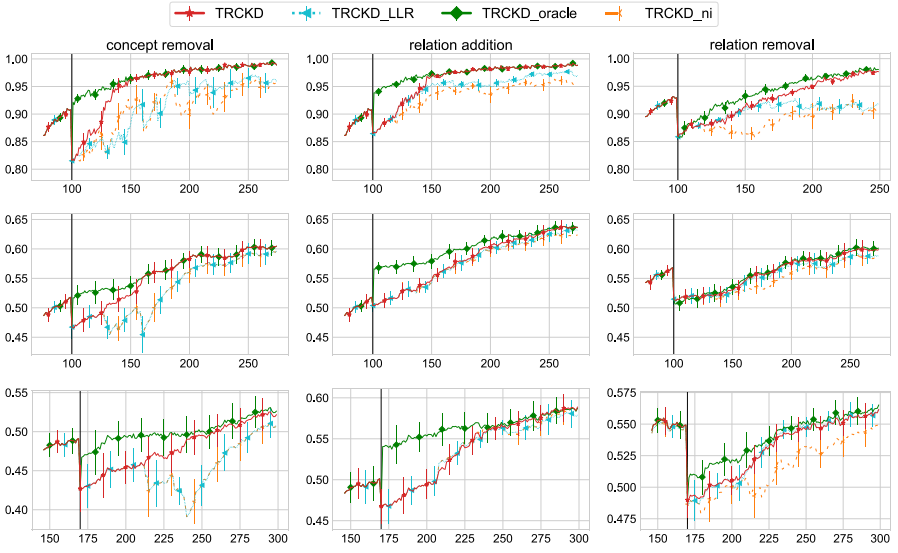


Fig. 3 Comparison in terms of micro F_1 between TRCKD and less interactive variants. Top to bottom: results for H-STAGGER, H-EMNIST and H-20NG. Left to right: concept drift, concept removal, relation addition, and relation removal. Error bars indicate std. error

4.3 Q1: knowledge-aware adaptation improves performance

To evaluate the impact of our adaptation strategy, we compare TRCKD against MW-kNN, PAW-kNN, kNN 1-window, regular kNN in a setting where all approaches are told exactly when KD occurs. In this setting, TRCKD is denoted TRCKD$_{oracle}$, as knowledge-aware disambiguation combined with exact detection implies a perfect knowledge of the kind of drift that occurred. We also compare to a simple knowledge-unaware variant of TRCKD, named TRCKD$_{forget}$, that adapts to all types of KD by forgetting old examples.

The results can be viewed in Fig. 2. The plots show that TRCKD$_{oracle}$ is the best performing method on all data sets and for all forms of KD. Note how the performance difference between TRCKD$_{oracle}$ and the alternatives is larger than standard errors in most cases, especially when considering the first iterations after the drift. Most often the runner up is TRCKD$_{forget}$: while it performs similarly to TRCKD$_{oracle}$ for concept removal (because our adaptation strategy boils down to forgetting in this simple setup), it does lag behind for relation addition and removal, showing a sizeable advantage for knowledge-aware adaptation. MW-kNN works reasonably well but suffers from relying on passive adaptation and does not always performs better than the two kNN baselines. PAW-kNN tends to underperform on H-EMNIST and H-20NG, especially when KD affects the relations. These results validate knowledge-aware adaptation on all data sets and allows us to answer **Q1** in the affirmative. For this reason, we will focus on knowledge aware adaptation in the following experiments.

4.4 Q2: interaction is beneficial

To measure the impact of interaction, we compare four variants of TRCKD that differ in what information they elicit from the supervisor, namely: TRCKD, TRCKD$_{oracle}$, TRCKD$_{LLR}$ and TRCKD$_{ni}$.

TRCKD$_{LLR}$ is a fully-automated version of TRCKD that follows up MMD detection by performing drift disambiguation with a likelihood ratio test. This test detects a relation y^j *is-a* y^i iff

$$P(y^j \mid y^i)/P(y^j \mid \neg y^i) \geq \beta \tag{7}$$

with $\beta = \infty$. (This is the best possible value for β as the ground-truth data is assumed to be always consistent with the ground-truth hierarchy.) If the test detects relation addition/removal, TRCKD$_{LLR}$ applies the corresponding knowledge-aware adaptation strategy, otherwise it defaults to emptying the current window of the detected concept(s).

TRCKD$_{ni}$ is like TRCKD except that instead of interacting with the user it assumes that all concepts detected as drifting by MMD have undergone individual concept drift and adapts by purging their current window.

The results in Fig. 3 are quite intuitive: TRCKD$_{oracle}$ substantially outperforms all alternatives in all cases except for relation removal in H-20NG. This shows that, if drift is detected correctly and timely, interactive disambiguation is extremely useful for

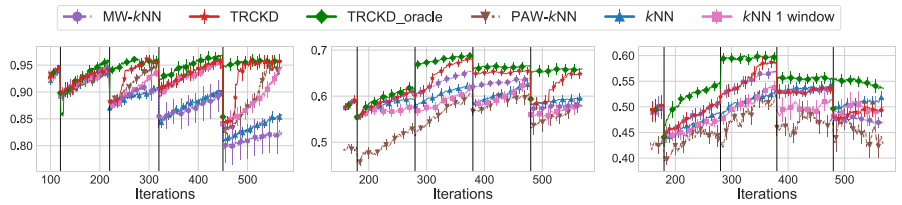


Fig. 4 TRCKD versus competitors on sequential KD in terms of micro F_1. Left: H-STAGGER. Middle: H-EMNIST. Right: H-20NG. Error bars indicate std. error

guiding knowledge-aware adaptation and quickly aligning the model to the ground-truth. TRCKD tends to perform substantially better than the no-interaction baselines TRCKD_{LLR} and TRCKD_{ni} and if MMD detection works well it quickly reaches the performance of the oracle. This allows us to answer **Q2** in the affirmative. If MMD underperforms (as in H-EMNIST), TRCKD does not get a chance to quickly interact with the user and shows no improvement. This could be fixed by better optimizing the choice of kernel and threshold used by MMD, perhaps by turning them into per-concept parameters. This is left to future work. The complexity of the task and the impact of the drift vary across the data sets. Thus, the difference between TRCKD and the competitor is smaller in some cases. Importantly, TRCKD interacts with the user 1.54 ± 0.78 times per run on average, showing that few interaction rounds are often enough to achieve a noticeable performance boost.

4.5 Q3: TRCKD works well in multi-drift settings

We consider a realistic scenario with four sequential KD events: concept drift, relation addition, relation removal, and concept removal. The results in Fig. 4 show that TRCKD tends to outperform all competitors except the oracle. The advantage is quite marked whenever the KD affects the concept hierarchy itself, up to $+10\%$ F_1 for H-STAGGER and $+5\%$ for H-EMNIST. The plots mirror the advantages shown by TRCKD in the previous experiments and highlight that the benefits knowledge-aware adaptation and interaction carry over to more realistic settings. This allows us to answer **Q3** in the affirmative. The lack of reactive adaptation penalizes MW-kNN and PAW-kNN, the latter especially on H-EMNIST.

4.6 Additional comparisons

TRCKD outperforms structure learning

Given the similarity between drift disambiguation and structure learning for probabilistic graphical models (Koller and Friedman 2009), it is natural to ask whether structure learning techniques could be used for handling KD in a fully automatical manner.

To answer this question, we evaluated a variant of TRCKD that uses graphical lasso to reconstruct the structure of the hierarchy from the most recent examples (Friedman et al. 2008). In particular, in each iteration a data set is built by combining the 70

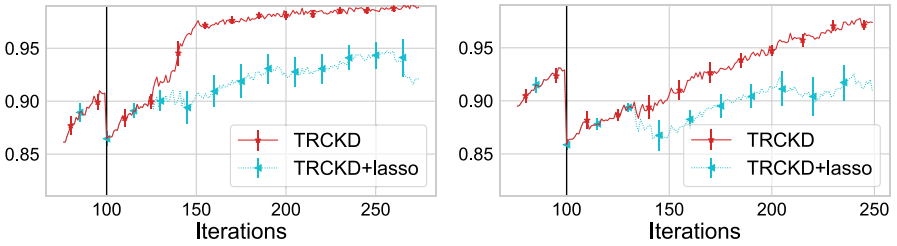


Fig. 5 Micro F_1 results on H-STAGGER, automatic drift type identification with Graphical Lasso and MMD for detection versus TRCKD. Left: relation addition. Right: relation removal

most recent examples (analogously to what is done for MMD) for all concepts in the machine's hierarchy. This data set set is fed to graphical lasso, which spits out an (sparse) undirected graph based on the empirical correlation between all the concepts. The directions of individual edges are set so to maximize the likelihood of the child implying the parent and edges are treated as *is-a* relations. The resulting directed graph replaces the machine's concept hierarchy. The difference between the previous and current concept hierarchy is used to perform knowledge-aware adaptation.

A comparison between TRCKD +lasso and TRCKD is reported in Fig. 5. It turns out that for relation addition and removal, graphical lasso often fails to estimate the ground-truth concept hierarchy, leading to *systematic prediction errors*. Furthermore, it is quite unstable and often detects spurious changes to the hierarchy. The main issue is that—like other fully automated approaches for structure learning—graphical lasso does require substantial amounts of data to perform reliably, and this is simply not the case in our non-stationary setting. This makes structured learning-based approaches unsuitable for this setup.

MMD outperforms mean embeddings

Another reasonable question is whether recent hypothesis tests outperform MMD for drift detection. Motivated by this, we replace MMD in TRCKD with Mean Embeddings (ME), a state-of-the-art kernel-based discrepancy between distributions (Jitkrittum et al. 2016). Importantly, the discrimination power of ME can be optimized for the task at hand by performing gradient ascent on an independent training set. In our experiment, we carried out this optimization on the pre-drift examples (i.e., on the first 100 examples in the stream for H-STAGGER and H-EMNIST and on the first 170 for H-20NG). We also used the very same tensor product kernel for both MMD and ME and for ME we tuned the width of the Gaussian kernel k_X over instances along with the discrimination power of ME.

Despite being more powerful than MMD on paper, ME did not perform as well in our tests. In particular, ME turned out to be overly sensitive and had severe false detection issues. In practice, ME tends to detect three to four times as many drifts as MMD. For instance, the average number of changepoints detected by TRCKD +ME for the case of H-EMNIST with sequential drifts is 15.25 ± 1.0 compared to 4.25 ± 0.5 of TRCKD +MMD. The ME final hyperparameters used for this experiment are number of witness points $J = 15$ and $\alpha = 0.01$. This overly sensitivity makes it inadequate for

interacting with the user: indeed, querying the user too frequently is likely to rapidly make her lose interest in the interaction in practice.

5 Related work

There is an enormous amount of work on concept drift, most of which focuses on single-label (Gama et al. 2014) and—to a lesser extend—multi-label (Zheng et al. 2019) classification. Surprisingly, drift in hierarchical classification, in which concepts are explicitly mutually constrained, has been so far neglected. In addition, we are not aware of any work on concept drift affecting the concept hierarchy nor on knowledge-aware adaptation strategies. As shown by our experiments, this setup is special enough that standard strategies struggle when applied or adapted to our setting.

The disambiguation step introduced with TRCKD is conceptually related to the problem of drift understanding (Lu et al. 2018), however works on this topic are unconcerned with hierarchical classification and, therefore, with drift in the background knowledge. To the best of our knowledge, this is the first work that tackles knowledge drift and drift disambiguation and to leverage interaction with a human supervisor to do so.

TRCKD makes ample use of well-known strategies. In particular, it combines ideas from MW-*k*NN (Spyromitros-Xioufis et al. 2011), a multi-label *k*NN approach that adapts passively by discarding old examples, with a proactive detection strategy based on sliding windows. Actively detecting and reacting to drift is key for enabling interaction with the user. Importantly, sliding windows offers distribution-free guarantees on detection accuracy under mild assumptions on the model class and on drifting frequency and speed (Kifer et al. 2004).

Maximum mean discrepancy

MMD has been applied extensively in domain adaptation (Zhang et al. 2013; Redko et al. 2020). The ME criterion (Jitkrittum et al. 2016), a more recent alternative, underperformed in our experiments but can act as a drop-in replacement for the MMD in applications where it performs better.

The idea of using concrete examples in the windows to explain drift was discussed in (Kifer et al. 2004), although not for MMD. It is true, however, that MMD lends itself to this task. For instance, Kim *et al.* propose a subset selection procedure for identifying both prototypes (examples that are representative of a particular learned concept) and criticism (examples that, conversely, are not representative) (Kim et al. 2016), built around a submodular objective defined using MMD. These ideas can be used directly for illustrate drifting concepts to the user and could be generalized to explain the effects of knowledge drift (Demšar and Bosnić 2018).

Drift over graph data

Drift detection has been studied in the context of graph classification. In this setting, the input data streams encodes a sequence of knowledge graphs (Paudel and Eberle 2020; Yao and Holder 2016; Zambon et al. 2018) or an ontology stream (Chen et al. 2017), and drift affects the distribution of said graphs. In contrast, in our target applications the machine receives a sequence of examples, consisting of a set

of subsymbolic observations and of concept annotations, while the knowledge graph controlling the relationship between concepts is completely unobserved. Since KD can only be inferred indirectly by monitoring the examples, this makes drift detection (and disambiguation) much harder.

Open world recognition

Our work is related to open world recognition, a streaming classification setting in which unanticipated classes appear over time (Boult et al. 2019). Open world recognition matches our setting in the case of concept addition, but it is unconcerned with other forms of KD. Furthermore, even though the overall goal is to achieve low error rate on the new, unknown classes (Scheirer et al. 2013), most algorithms for open word recognition achieve this by refusing to output any predictions for incoming instances that belong to the unknown classes (Scheirer et al. 2014; Rudd et al 2017; Boult et al. 2019). In stark contrast, and compatibly with other approaches to concept drift, TRaCking Knowledge Drift aims to adapt the model to new classes rather than reject challenging data points. Other recent work on interactive classification under concept addition (Bontempelli et al. 2020) focuses on handling noisy labels rather than on adapting to drift.

Novelty and anomaly detection

Our approach shares some similarities with recent work interactive anomaly detection for structured data (Ding et al. 2019). In this work, anomalies are first identified by a machine and then double-checked by a human supervisor. The key idea of leveraging interaction with a supervisor has similar motivations as in our task. More generally, concept drift is related to novelty, anomaly, and out-of-distribution detection, which tackle the problem of identifying unexpected or anomalous datapoints (Pimentel et al. 2014). Concept drift and novelty detection have also been combined (Spinosa et al. 2007; Masud et al. 2011). One key difference is that in these settings the set of concepts is typically fixed. Furthermore, and more importantly, KD—and more generally concept drift—involve updating the model to track changes in the world, wherease no adaptation is typically necessary in novelty detection.

Other topics

In continual learning a machine acquires new concepts or tasks over time, and the challenge is to prevent the learned model—typically a neural network—from forgetting previously acquired knowledge (Parisi et al. 2019; Flesch et al. 2018). In stark contrast, in our setting the goal is to intentionally forget obsolete information whenever necessary. Moreover, continual learning is unconcerned with forms of knowledge drift other than concept addition, whereas we tackle all forms of KD.

Another related but separate topic is active learning of graphical models, see for instance (Tong and Koller 2001), where the aim is to acquire a multivariate distribution (with a non-trivial factorization) by querying a human-in-the-loop. Specifically, the machine asks for the downstream value of certain variables upon intervention (i.e., manipulating a variable to a chosen value). Individually, these queries are not very informative and it may take tens of queries to acquire a model with a handful of variables. In contrast, TRaCking Knowledge Drift is only interested in obtaining a description of *change* to the concept *hierarchy*—not a whole model, not a distribution over concepts—and to this end it presents the user with an initial guess as guidance

and then elicits a one-shot description. These more expressive queries are tailored to the KD use case and lead to more efficient interaction.

Prototypical networks uses a prototypes representations for each class, which are used to classify a test point (Snell et al. 2017). Adding new classes cause catastrophic forgetting, which deteriorate the classifier performance . To avoid this effect, a set of training points that approximate the class mean are kept (Rebuffi et al. 2017). This solutions is then robust to data representations changes. These two works do not address the concept drift challenge. By estimating the drift in the previous task, it can be compensated in the new task without storing exemplar of previous tasks (Yu et al. 2020). The main limitations of these approaches is that they support one prototype for each class (e.g., different libraries are mapped to the same prototype). The solution does not provide a way to erase concepts that does not hold anymore.

6 Conclusion

We introduced the problem of knowledge drift in hierarchical classification and proposed to partially offload drift disambiguation to a user. We also proposed TRCKD, an approach for learning under KD that combines *automated* drift detection and adaptation, upgraded to hierarchical classifiers, with *interactive* drift disambiguation. Our empirical results indicate that TRCKD outperforms fully automated approaches by asking just a few questions to the user, even when detection performance is not ideal, showcasing the importance of interaction for handling knowledge drift.

Our work can be improved and extended in several directions. First of all, in practical applications the learner often receives no labels during its normal operation and must acquire any necessary supervision from the user during the interaction. Dealing with this setting requires to integrate a knowledge-aware active learning component into TRCKD. On the user interaction side, we plan to improve the interpretability of our interaction protocol (beyond using examples to illustrate the machine's behavior to the user) by adapting ideas from explainable AI (Demšar and Bosnić 2018) and explainable interactive learning (Schramowski et al. 2020). Another promising direction is to extend TRCKD to sequential learning methods beyond kNN, especially deep neural networks. The challenge here is to develop knowledge-aware adaptation strategies appropriate for this class of models. It is relatively straightforward to extend our approach to instance-based neural models, see for instance (Snell et al. 2017). Knowledge-aware adaptation for other kinds of neural networks could be approached by leveraging recent developments in machine unlearning—so to force the model to forget obsolete concepts and relations—cf. Cao and Yang (2015) and follow-ups. These extensions, however, are highly non-trivial and left to future work.

Acknowledgements We are grateful to the anonymous reviewers for helping us to improve the paper. The research of FG and AP has received funding from the European Union's Horizon 2020 FET Proactive project "WeNet—The Internet of us", grant Agreement No. 823783. The research of AB and ST has received funding from the "DELPhi—DiscovEring Life Patterns" project funded by the MIUR Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) 2017—DD n. 1062 del 31.05.2019. The research of ST and AP was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No. 952215.

Funding Open access funding provided by Universitá degli Studi di Trento within the CRUI-CARE Agreement.

References

Bontempelli A, Teso S, Giunchiglia F, et al (2020) Learning in the wild with incremental skeptical gaussian processes. In: IJCAI

Boult TE, et al (2019) Learning and the unknown: surveying steps toward open world recognition. In: AAAI

Cao Y, Yang J (2015) Towards making systems forget with machine unlearning. In: 2015 IEEE symposium on security and privacy

Chen J, Lécué F, Pan J, et al (2017) Learning from ontology streams with semantic concept drift. In: Twenty-sixth international joint conference on artificial intelligence, international joint conferences on artificial intelligence organization

Cohen G, Afshar S, Tapson J, et al (2017) Emnist: extending MNIST to handwritten letters. In: IJCNN

Demšar J, Bosnić Z (2018) Detecting concept drift in data streams using model explanation. Expert Syst Appl 92:546–559

Ding K, Li J, Liu H (2019) Interactive anomaly detection on attributed networks. In: Proceedings of the twelfth ACM international conference on web search and data mining, pp 357–365

Flesch T, Balaguer J, Dekker R et al (2018) Comparing continual task learning in minds and machines. Proc Natl Acad Sci 115:E10313–E10322

Friedman J, Hastie T, Tibshirani R (2008) Sparse inverse covariance estimation with the graphical lasso. Biostatistics 9:432–441

Gama J, Žliobaitė I, Bifet A et al (2014) A survey on concept drift adaptation. ACM Comput Surv 46:44–1

Giunchiglia F, Bignotti E, Zeni M (2017) Personal context modelling and annotation. In: PerCom

Gretton A, Borgwardt KM, Rasch MJ et al (2012) A kernel two-sample test. JMLR 13:723–773

Jitkrittum W, Szabó Z, Chwialkowski KP, et al (2016) Interpretable distribution features with maximum testing power. In: NeurIPS

Kifer D, Ben-David S, Gehrke J (2004) Detecting change in data streams. In: VLDB

Kim B, Khanna R, Koyejo OO (2016) Examples are not enough, learn to criticize! criticism for interpretability. In: Advances in neural information processing systems, vol 29

Kingma DP, Welling M (2014) Auto-encoding variational bayes. In: ICLR'14

Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques. MIT Press, Cambridge

Lloyd JR, Ghahramani Z (2015) Statistical model criticism using kernel two sample tests. In: Advances in neural information processing systems

Lu J, Liu A, Dong F et al (2018) Learning under concept drift: a review. IEEE Trans Knowl Data Eng 31:2346–2363

Masud M, Gao J, Khan L et al (2011) Classification and novel class detection in concept-drifting data streams under time constraints. IEEE Trans Knowl Data Eng 23:859–874

Parisi GI, Kemker R, Part JL et al (2019) Continual lifelong learning with neural networks: a review. Neural Netw 113:54–71

Paudel R, Eberle W (2020) An approach for concept drift detection in a graph stream using discriminative subgraphs. TKDD 14:1–25

Pérez-Cruz F (2009) Estimation of information theoretic measures for continuous random variables. In: NeurIPS

Pimentel MA, Clifton DA, Clifton L et al (2014) A review of novelty detection. Signal Process 99:215–249

Rebuffi SA, Kolesnikov A, Sperl G, et al (2017) ICARL: incremental classifier and representation learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition
Redko I, Morvant E, Habrard A, et al (2020) A survey on domain adaptation theory: learning bounds and theoretical guarantees. arXiv preprint arXiv:2004.11829
Reimers N, Gurevych I (2019) Sentence-bert: sentence embeddings using siamese bert-networks. In: EMNLP-IJCNLP
Roseberry M, Krawczyk B, Cano A (2019) Multi-label punitive kNN with self-adjusting memory for drifting data streams. TKDD 13:1–31
Rudd EM, Jain LP, Scheirer WJ et al (2017) The extreme value machine. IEEE Trans Pattern Anal Mach Intell 40:762–768
Scheirer WJ, de Rezende Rocha A, Sapkota A et al (2013) Toward open set recognition. IEEE Trans Pattern Anal Mach Intell 35:1757–1772
Scheirer WJ, Jain LP, Boult TE (2014) Probability models for open set recognition. IEEE Trans Pattern Anal Mach Intell 36:2317–2324
Schlimmer JC, Granger RH (1986) Incremental learning from noisy data. Mach Learn 1:317–354
Schramowski P et al (2020) Making deep neural networks right for the right scientific reasons by interacting with their explanations. Nat Mach Intell 2:476–486
Snell J, Swersky K, Zemel R (2017) Prototypical networks for few-shot learning. In: NeurIPS
Sorower MS (2010) A literature survey on algorithms for multi-label learning. Oregon State University, Corvallis
Spinosa EJ, de Leon F, de Carvalho AP, Gama J (2007) Olindda: a cluster-based approach for detecting novelty and concept drift in data streams. In: Proceedings of the 2007 ACM symposium on applied computing, pp 448–452
Spyromitros-Xioufis, et al (2011) Dealing with concept drift and class imbalance in multi-label stream classification. In: IJCAI
Stojanovic L, Maedche A, Motik B, et al (2002) User-driven ontology evolution management. In: ECAW
Szabó Z, Sriperumbudur BK (2017) Characteristic and universal tensor product kernels. JMLR 18:233–1
Tong S, Koller D (2001) Active learning for structure in Bayesian networks. In: International joint conference on artificial intelligence. Citeseer, pp 863–869
Yao Y, Holder LB (2016) Detecting concept drift in classification over streaming graphs. In: Proceedings of the KDD workshop on mining and learning with graphs
Yu L, Twardowski B, Liu X, et al (2020) Semantic drift compensation for class-incremental learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition
Zambon D, Alippi C, Livi L (2018) Concept drift and anomaly detection in graph streams. IEEE Trans Neural Netw Learn Syst 29:5592–5605
Zhang K, Schölkopf B, Muandet K, et al (2013) Domain adaptation under target and conditional shift. In: ICML
Zhang ML, Zhang K (2010) Multi-label learning by exploiting label dependency. In: ACM SIGKDD
Zheng X, Li P, Chu Z et al (2019) A survey on multi-label data stream classification. IEEE Access 8:1249–1275