Mach Learn (2017) 106:1747–1770
DOI 10.1007/s10994-017-5661-5

Group online adaptive learning

Alon Zweig[1] · **Gal Chechik**[2,3]

Received: 30 March 2016 / Accepted: 6 July 2017 / Published online: 1 August 2017

Abstract Sharing information among multiple learning agents can accelerate learning. It could be particularly useful if learners operate in continuously changing environments, because a learner could benefit from previous experience of another learner to adapt to their new environment. Such group-adaptive learning has numerous applications, from predicting financial time-series, through content recommendation systems, to visual understanding for adaptive autonomous agents. Here we address the problem in the context of online adaptive learning. We formally define the learning settings of Group Online Adaptive Learning and derive an algorithm named Shared Online Adaptive Learning (SOAL) to address it. SOAL avoids explicitly modeling changes or their dynamics, and instead shares information continuously. The key idea is that learners share a common small pool of experts, which they can use in a weighted adaptive way. We define group adaptive regret and prove that SOAL maintains known bounds on the adaptive regret obtained for single adaptive learners. Furthermore, it quickly adapts when learning tasks are related to each other. We demonstrate the benefits of the approach for two domains: vision and text. First, in the visual domain, we study a visual navigation task where a robot learns to navigate based on outdoor video scenes. We show how navigation can improve when knowledge from other robots in related scenes is available. Second, in the text domain, we create a new dataset for the task of assigning submitted papers to relevant editors. This is, inherently, an adaptive learning task due to the dynamic nature of research fields evolving in time. We show how learning to assign editors improves when knowledge from other editors is available. Together, these results demonstrate the benefits for sharing information across learners in concurrently changing environments.

Editors: Thomas Gärtner, Mirco Nanni, Andrea Passerini and Celine Robardet.

✉ Alon Zweig
alon.zweig@gmail.com

Gal Chechik
gal.chechik@biu.ac.il

1 Qylur Intelligent Systems Inc., 1015 East Meadow Circle, Palo Alto, CA 94303, USA

2 Gonda Brain Research Center, Bar-Ilan University, 52900 Ramat Gan, Israel

3 Google Research, 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA

Keywords Multi-task learning · Knowledge transfer · Adaptive learning · Online learning · Domain adaptation

1 Introduction

Sharing information among learning agents is beneficial in many learning scenarios, including multi-task learning (Caruana 1997; Argyriou et al. 2008) and knowledge transfer (Pan and Yang 2010) and has been mostly studied in stationary environments. The current paper addresses information sharing in non-stationary environments. Here each individual learner adapts to its own drifting distribution while potentially benefiting from information shared by other learners.

As one illustrative example, consider a set of interactive domestic robots, each located in the kitchen of a different home. The distribution of objects encountered by each robot may change with time, and some objects may be found in more than one home. By sharing knowledge about their environments with other robots, each robot can learn to recognize objects faster and more accurately.

Various other real-life applications involve learning multiple models for non-stationary data streams. In financial time-series prediction, separate learners are trained for separate financial instruments and generalization across instruments can be useful. In recommending news items to users, the model of one user could be useful for recommending items for other users. All these cases involve multiple data streams, each sampled from a non-stationary and possibly different distribution, and learning could potentially improve by sharing information between learners.

The current paper presents a novel learning setting which has three key characteristics. First, environments change with time, hence we wish that learning agents *continuously adapt* to the changes. Second, history may repeat itself and agents may need to *record long-past events*. Third, environments share features, hence we wish that the learning agents *share information*. We name this learning setting Group Online Adaptive Learning (GOAL).

Unfortunately, in non-stationary environments, since the learning task changes over time, storing concepts and sharing them across learning agents introduces several challenges. First, constraints on memory complexity limit the number of "memories" that can be stored. Second, a learner can enjoy the advantage of sharing information only once he recognized the relevant shared information. This means a learner should converge quickly to using shared information if it is relevant and available in the stored set. Finally, sharing among learners can lead to *negative transfer*, where sharing interferes with learning (Pan and Yang 2010). This issue is particularly important when learning in changing environments since a learner that is helpful at one point in time may interfere at another time. A good sharing algorithm should achieve*safe sharing*, namely, when a learner L_1 no longer contributes to the learning process of another learner L_2, then L_2 should quickly adapt to avoid using the information shared by L_1 until it becomes useful again.

With these challenges in mind, we designed an algorithm we call *Shared Online Adaptive Learning* (SOAL), which learns efficiently in the GOAL setting. SOAL maintains a logarithmic sketch of the environment of each learning agent and shares this sketch with all learners. We show how this sketch enables quick adaption to the changing environment of each agent, without having to detect the changes explicitly. We also show, in the context of a regret bound, that this algorithm has safe sharing.

We illustrate the key properties of our suggested approach on a synthetic dataset and evaluate our approach on two learning tasks in the vision and text domains. In the visual

domain, we tested our algorithm on a hard task of analyzing dynamic visual scenes when multiple agents can share their learning models. In the text domain, we collected a new dataset and tested our approach on the editor assignment problem, learning jointly the relevance of papers for each of the editors. In all experiments, we show that our approach suffers smaller losses compared to existing approaches.

The paper is organized as follows: Sect. 2 discusses the group online adaptive learning setting, and Sect. 3 discusses the relation of this setting to existing research fields. In Sects. 4 and 5, we proposed a learning architecture and algorithm for the GOAL setting. We analyze the algorithm in Sect. 6, where we define the *group adaptive regret* and prove the 'safe-sharing' property. Experimental results on three datasets: synthetic, visual domain and text domain are presented in Sect. 7.

2 Group online adaptive learning

Group Online Adaptive Learning is a marriage between multi-task learning (multiple agents) and online adaptive learning (non-stationary environments). Figure 1 illustrates the relation between the approach taken in this paper and these two research directions. Now we define formally the learning setting and discuss in detail the relation of our work to these two research directions.

In Group Online Adaptive Learning, N learning agents $L_1, \ldots, L_N$ learn continuously with each agent facing its own non-stationary environment. At each time step $t = 1, \ldots, T$, an agent L_a, receives a sample $u_t^a \in U$ and predicts a label $x_t^a(u_t^a) \in Y$ using hypothesis $x_t^a \in K$. Here U denotes the input space, Y denotes the label space and K denotes the hypothesis space. Agent L_a suffers a loss $l_t^a(x_t^a(u_t^a), y_t^a)$, $(u_t^a, y_t^a) \sim D_t^a$ where $y_t^a \in Y$ and D_t^a is assumed to be non-stationary, namely for $s \neq t$ the following might hold: $\exists (u, y) \in U \times Y : p_{D_s^a}(u, y) \neq p_{D_t^a}(u, y)$.

In standard online learning, the performance of an online algorithm is measured in comparison to the best fixed hypothesis in hindsight, defined by the regret:

$$Regret(A) \equiv \sum_{t=1}^{T} l_t(x_t(u_t), y_t) - \min_{x^* \in K} \sum_{t=1}^{T} l_t(x^*(u_t), y_t).$$

Importantly, static regret is often inadequate when handling non-stationary data streams, since algorithms that minimize the regret fail to follow changes in the distribution. Another

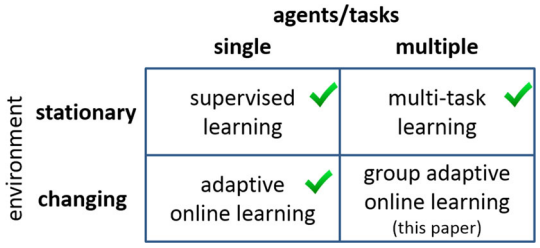

Fig. 1 Research areas in machine learning organized based on the multiplicity of agents or tasks and on the stationarity of the distribution. Multi-task approaches for stationary distributions also include *knowledge transfer* (Pan and Yang 2010) and *domain adaptation* (Sugiyama et al. 2007; Saenko et al. 2010). Single-agent approaches for non-stationary distributions include *online adaptive learning* methods (Hazan and Seshadhri 2009) and *concept drift* learning (Zliobaite 2009; Gama et al. 2014)

measure, *adaptive regret* (AR) (Hazan and Seshadhri 2009), aims to minimize the regret of an algorithm A at any interval $I = [r, \ldots, s] \subseteq [1, \ldots, T]$ during learning. The adaptive regret of an algorithm A on a sequence $(u_1, y_1), \ldots, (u_T, y_T) \in U \times Y$ is defined as:

$$Adaptive\text{-}Regret(A) \equiv \sup_{I=[r,s]} \left(\sum_{t=r}^{s} l_t(x_t(u_t), y_t) - \min_{x_I^* \in K} \sum_{t=r}^{s} l_t(x_I^*(u_t), y_t) \right). \quad (1)$$

Adaptive regret is localized in time in the sense that the loss of A is compared to the loss obtained at a single (worst) contiguous interval.

In the GOAL setting, we aim to optimize the group performance of all learners. We define:

Definition 1 On a set of sequences $\{(u_1^a, y_1^a), \ldots, (u_T^a, y_T^a) \in U \times Y\}_{a=1}^N$ the group adaptive regret of an online optimization algorithm A for multiple learners is defined as the sum of adaptive regrets that the individual learners suffer.

$$Group\text{-}Adaptive\text{-}Regret(A) \equiv \sum_{a=1,\ldots,N} \sup_{I=[r,s]} \left(\sum_{t=r}^{s} l_t^a(x_t^a(u_t^a), y_t^a) - \min_{x_I^{a*} \in K} \sum_{t=r}^{s} l_t^a(x_I^{a*}(u_t^a), y_t^a) \right). \quad (2)$$

When learning agents are trained to solve unrelated tasks, minimizing the *group adaptive regret* can be approached trivially by optimizing each adaptive learner separately. However, when the learning tasks are related, the regret can be decreased further by sharing information among learning agents.

As an example, imagine a set of cameras that need to track-and-recognize objects or people. Modern trackers learn appearances during tracking. Consider an object with a view that makes it very hard to recognize, perhaps because it blends with the background. Here, for illustration, all N agents (cameras) share the same worst sequence of images I, and suffer the same regret on this sequence. From the definition of adaptive regret as the supremum of regret over sequences we get that all agents share the same adaptive regret $AR(A)$. When each agent learns independently without sharing, the group adaptive regret equals $N \times AR(A)$. However, if one agent encounters this worse sequence before all other agents, it can share its experience (based on appearance of object viewpoints). The group regret could be reduced significantly to $AR(A) + \sum_{a=1,\ldots(N-1)} R_a$, where $\forall a$, $R_a \ll AR(A)$ denotes the regret that a learning agent suffers during the process of accessing the shared information. The gain that can be achieved by sharing depends on how effectively agents can use past experience of other agents. Section 6.2 shows that this regret R_a is indeed small for SOAL.

While information sharing could benefit learning, it may also harm learning if unrelated tasks are forced to share information. This effect is known as *negative transfer* (Pan and Yang 2010). For instance, in the multi-camera setting described above, *negative transfer* may occur if each camera tracks another object with a significantly different appearance, like having a different color in color-based tracking. In such a setting, trackers will learn incorrect object-appearance models by using false color information, for example by considering non-zero weights over color features learned by different unrelated trackers. Section 6.1 shows that SOAL avoids *negative transfer* in the sense that the adaptive regret of SOAL is asymptotically equivalent to that of single adaptive learner without sharing information.

3 Related work

The current paper is related to two lines of research: sharing among multiple learning tasks, and learning in a changing environment.

3.1 Sharing among multiple learning tasks

First, our work is closely related to previous learning methods considering sharing among multiple learning tasks. Information sharing has been studied mainly in the four settings: knowledge transfer (KT), domain adaptation (DA), life-long learning (LL) and multi-task learning (MTL). A formal definition is outside the scope of this paper, and since these terms are not always well delineated, we interpret them according to their most common usage. While these settings are closely related to GOAL, they differ by several key aspects.

The first key difference is that information transfer in GOAL occurs continuously. KL and DA address the problem of transferring learned knowledge to a new learning task or when the test data comes from a different domain than the train data (Pan and Yang 2010). Both KT and DA transfer information at a single transfer from the source tasks to the target task. For instance, knowledge-transfer methods have been applied in visual object recognition to transfer information one category (like car) to a related category (like motorbike), because it helps to recognize categories with few samples (Tommasi et al. 2010; Zweig and Weinshall 2007). As another example, in DA the conditional distribution of output values given input points is assumed to be shared among learning tasks (Sugiyama et al. 2007; Saenko et al. 2010). Unlike KT and DA, information transfer in GOAL occurs continuously as the data distribution drifts. This property of GOAL is similar to life-long learning (Thrun and Mitchell 1995; Ruvolo and Eaton 2014), a form of continuous transfer learning. In LL an agent receives a continuous stream of learning tasks and learns a growing set of models, one for each task.

A second key difference is that KT, DA and LL are task-sequential in nature, namely, learning first occurs in one task and applied later to another task. This can be viewed as an adaption process. A learner adapts learned knowledge to the new task. More formally, KT, DA and LL maintain two sets of learning tasks, L_s for which the learning has ended and L_t for which the learning occurs with the help of previously learnt L_s. Unlike these approaches, in GOAL every single learner operates in a non-stationary environment which may implicitly drift and switch the concept being learned at unknown points in time. As a result, all learners need to continuously adapt to the drifting concepts, yielding in a joined set of learners L acting as both sources and targets.

Multi-task learning (MTL) is another setting for sharing information across tasks. MTL is not task-sequential, having multiple agents learn simultaneously while sharing information (Caruana 1997; Argyriou et al. 2008). Among MTL approaches, the most related to GOAL are approaches dealing with an online setting (Lugosi et al. 2009; Dredze and Crammer 2008; Saha et al. 2011; Cavallanti et al. 2010; Zweig and Weinshall 2013; Adamskiy et al. 2012). These methods enable sharing of information while receiving a continues stream of data. Generally, in MTL it is often assumed that relations between tasks are known in advance, that relations between tasks remain the same, that all learners have access to all data streams, or that samples come from a stationary distribution.

The GOAL learning setting can also be viewed as online multi-task learning, with several key assumptions relaxed: Samples arrive from drifting distributions; relations among learners are not known in advance and may actually change in time; and finally, the information shared among learners is limited to be strongly sublinear in the length of the data stream.

3.2 Learning in a changing environment

The second research field which is closely related to our work is learning in changing environments, also known as "concept-drift". In this setting, the distribution of the data is non stationary and unknown to the learners. This topic has been extensively studied in the setting of a single learner (Zliobaite 2009; Gama et al. 2014). Changing environments have also been studied in the setting of distributed learning (Kamp et al. 2014; Gabel et al. 2015; Ang et al. 2013). In this setting, each node in a distributed network of learners receives its own set of samples but samples of all nodes are generated by the same distribution. Ang et al. (2013) relax this assumption on shared distribution and allow concept drift to occur in each node asynchronously. Still, they assume that a single sequence of drifted distributions generates the samples for all nodes. The GOAL setting described here is more general in the sense that no such assumptions are made on the relation among learners.

An important characteristic of methods for learning in changing environments is whether they explicitly detect changes in the environments. Gabel et al. (2015) proposed a method that explicitly detects changes in least-square regression models, in order to reduce costly model updates in a distributed environment. Methods that avoid explicit detection of changes typically optimize an online measure of performance designed to fit the non-stationary nature of the data. For instance, in Freund et al. (1997) and Herbster and Warmuth (1998) the optimal hypothesis used to compute the regret is assumed to change over time and come from a discrete set of experts. Alternatively, Hazan and Seshadhri (2009) suggest a continuous notion of regret for changing environments: adaptive-regret (AR). This paper adopts the notion of adaptive-regret and extends it to multi-task learning. In addition, we provide an algorithm to learn while sharing information, while preserving the adaptive guarantees of the algorithm presented in Hazan and Seshadhri (2009).

4 A single adaptive learner in a changing environment

This section describes our approach for learning a single adaptive agent. We extend the approach to group adaptive learning in Sect. 5.

To solve the group online adaptive learning problem, we use a known reduction from online learning to online convex optimization (Hazan 2016). The reduction allows us to build on the work of adaptive online convex optimization (Hazan and Seshadhri 2009).[1] Hazan and colleagues described an efficient approach to the problem of minimizing the adaptive regret (Eq. 1), for a single agent in a non-stationary data-stream (Hazan and Seshadhri 2009). The main idea is that the agent adaptively maintains a small set of experts, which themselves are online convex optimization algorithms.

Formally, at time t, the learner has access to a set of experts $S_t = \{E^1, \ldots, E^R\}$. Each of these experts E^j makes a prediction x_t^j at time t. The learner maintains a distribution p_t over expert advice, and uses it to compute its own prediction $x_t = \sum_j p_t^j x_t^j$. Equivalently, in the online learning setting, when the learner is presented with a sample u_t, it predicts

[1] Specifically, the loss in our learning problem is written as a function over the learned parameters, the sample and the label, whereas in standard online convex optimization notation the loss is a function of the parameters only. For a convex hypothesis class an online learning problem can be written as a convex optimization problem by defining the following loss: $f_t(x_t) \equiv l_t(x_t(u_t), y_t)$.

$x_t(u_t) = \sum_j p_t^j x_t^j(u_t)$,[2], where $x_t^j(u_t)$ is the prediction of expert E^j, given sample u_t at time t . After the learner made his prediction, it suffers a loss $f_t(x_t) \equiv l_t(x_t(u_t), y_t)$.

More specifically, the distribution over experts can be maintained using an efficient algorithm called *Follow the Leading History* (FLH) (Hazan and Seshadhri 2009). In FLH, three adaptive processes take place at each time step. First, the pool of experts S_t is revised: a new expert is added and some experts are removed. Second, the learning agent updates the weights p_t over the experts. Third, every expert E^j modifies its model.

- *Component 1: Maintaining the set of experts* S_t. At each time step t, FLH initiates a new online learner and adds it to the working set of experts S_t. Initiating a new online learner provides quicker adaptation by introducing an expert which has not seen past samples and is more effected by present samples.
  To limit the size of S_t, the set is then pruned in a way that guarantees that its size grows like $O(\log T)$. Specifically, once an expert is created, it is assigned a "lifetime", which predetermines the time it will retire from the working set. To compute the "lifetime", each time step is uniquely represented as $t = r2^k$ where $r \in \mathbb{N}$ is odd and $k \in \mathbb{N}$. The lifetime of an expert added at time t is set to $2^{k+d} + 1$, where d is a parameter.
- *Component 2: Updating weights over experts* p_t^j. FLH uses multiplicative updates of the weights, $p_{t+1}^j \propto p_t^j e^{-\alpha f_t(x_t^j)}$. The newly added expert receives a weight of $1/(t+1)$.
- *Component 3: Updating the expert models.* In FLH, the experts are themselves online convex optimization algorithms, and they all modify their models at each step given the loss they suffer.

When the loss f is α exp-concaveand $\forall x, |f(x)| \leq 1$, FLH obtains an adaptive regret upper bound of:

$$AR(A) \leq R(T)O(\log T) + O(\log^2 T). \tag{3}$$

This upper bound has three components. First, $R(T)$ is the standard regret of a single expert. Second, for an interval $[r, s]$, the regret between the loss of FLH and that of an expert that started at time r is bounded by $O(\log T)$ [Lemma 3.1 in Hazan and Seshadhri (2009)]. Third, when pruning the working set of experts, a further regret of size $\log(T)$ is suffered, since the pruned set may not contain the best expert (Lemma 3.2 in Hazan and Seshadhri (2007)). Together, the total upper bound on the adaptive regret is $R(T)\log(T)+O(\log^2(T))$. Similarly, for the case of a general convex loss function the adaptive-regret bound is $R(T)\log(T) + O(\sqrt{T\log^3(T)})$ (Zinkevich 2003).

5 Sharing in a changing environment

We turn to learning with multiple agents and describe a framework and an algorithm for sharing while learning in a non-stationary environment. The algorithm is designed to address three problems. First, to cope with a changing environment, the algorithm should allow each learning agent to quickly adapt to a new distribution. Second, to benefit from sharing across agents, the algorithm should allow each learner to use information available to other learners. Third, to avoid negative sharing, the algorithm should allow each learner to quickly disregard non useful information from other learners.

[2] As long as the loss function $f_t(x_t)$, as defined in the reduction of the online learning problem to the online convex optimization problem, is α-exp concave and $\forall x, |f(x)| \leq 1$ the regret bounds of Hazan and Seshadhri (2009) hold.

To achieve these three goals, the algorithm tracks efficiently the changing environment while exploiting information learned previously by other agents in the network. This approach is particularly useful when a learning agent experiences an event before other learners. In these scenarios, learners can benefit from the collective past experiences.

To illustrate why the problem is hard we first present two straw-man solutions. A first naive approach to achieve sharing would be to use an algorithm where each learning agent trains a unique set of experts on its own data stream, similarly to FLH. But, in addition to FLH, each learner also has access to experts trained by other agents and learns to weight them. Unfortunately, in this approach an agent may learn to trust an expert which continuously adapts to the changing environment of another learning agent. This can lead to negative transfer when the relation among the learning agents changes. This issue is manifested in the fact that this approach will not preserve the adaptive regret bound of Hazan and Seshadhri (2009). To prove the adaptive regret bound it is necessary that a learning agent use experts which adapt to its own distribution.

A second straw-man solution would be to use a classical MTL approach of selecting informative features jointly among different learning agents. This can be achieved by modifying the FLH algorithm to train the set of experts of each agent jointly with experts from other agents, using a group-lasso regularization term encouraging joint feature selection. Here again, the AR bounds of Hazan and Seshadhri (2009) do not hold and negative transfer may occur when agents do not have shared features.

The two straw-man solution described above suffer from potential negative transfer and lose guarantees on adaptive regret. They also cannot handle sharing information from long-past history. We devise an alternative approach that shares historical events from environments through *functions of the data*. The key idea is that learners share few functions of the data that can be weighted by each learner. To select these functions, we use the fact that each learner has already trained useful experts for its environment, and we retire these experts into a pool that is shared by all learners. This allows efficient sharing using a knowledge representation that is logarithmic in the size of the data that each learner observes and maintains the AR bounds.

5.1 Overview of the architecture and algorithm

We start with an overview of the architecture and the algorithm, and describe them in details in Sects. 5.2–5.4.

The architecture can be viewed as a network with two hidden layers as illustrated in Fig. 2. The network has the following components:

1. Data layer.
2. A Shared knowledge layer with a set of *shared* experts. Experts in this set stopped adapting, and serve as 'recorded snapshots' of the past, with the purpose of providing a sketch of historical information shared across all learners.
3. An adaptive layer with a set of *private* experts. Experts in this set continuously adapt, and are of two kinds. First, *historical* experts, which make predictions based on the experts in the *shared* set. Second, *contemporary* experts, which make predictions based directly on data samples.
4. Output layer.

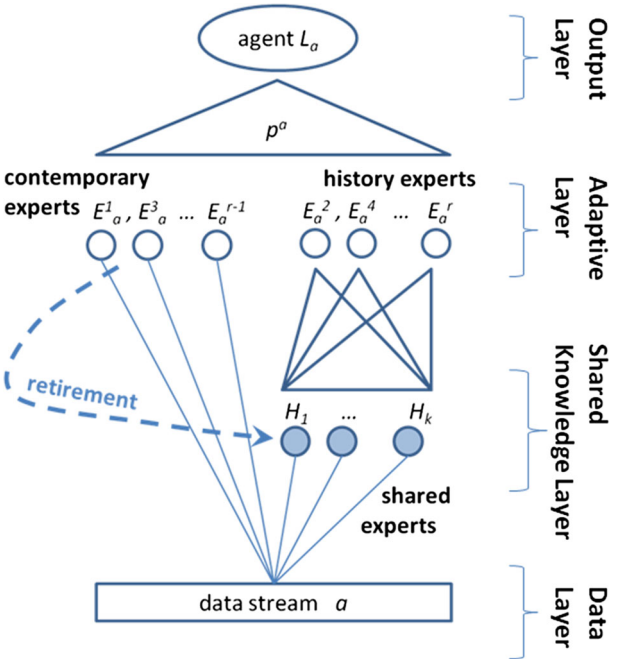

Fig. 2 The architecture. Each learning agent L_a has a pool of experts $E^1_a, \ldots, E^r_a$, half of which (contemporary) directly compute over current data, and the other half (historical) compute over the outputs of *shared* experts. *Circles* denote experts. *Filled circles* denote frozen experts that no longer adapt. The *dashed line* illustrates that contemporary experts retire to become "frozen" shared experts

Overview of the algorithm

We present an algorithm for training a set of learning agents we name *Shared Adaptive Online Learning* (SOAL), see Algorithm 1. Like FLH, this algorithm continuously adapts each of its learners to their environments, but unlike FLH, every learner also adaptively accesses a shared historical memory. The key idea of the approach is that each learner maintains a set of weights over both *historical* and *contemporary* adaptive experts (p^a in Fig. 2). This allows the learner to adapt quickly to changes in the environment , and also quickly recall experts from the collective memory of other learners.

The algorithm proceeds as follows. As in the case of a single adaptive learner describe in Sect. 4, three types of updates take place at each time step t:

- *Update the expert models.* All experts in the private set are themselves online optimizers. At *Step 1a* of Algorithm 1, all historical and contemporary experts output their predictions, suffer a loss and update their model. We denote by $E^j(f_{t-1})$ the output at time t of expert j using its modified model after suffering loss at time $t-1$.
- *Update weights over experts in the private set.* At each time step t, a distribution, p_t, over experts in the private set is updated (*Step 1*). This update involves several steps, first the distribution is updated given the losses of all experts in the private set. Resulting in a distribution $\hat{p}_{t+1}$ (*Step 1c*). At Steps *1d* and *1e*, two new experts are added to the private set, their weights are temporarily set to 0 so they are considered part of the distribution $\hat{p}_{t+1}$. Then, the new experts are given non-zero weights and all the

Algorithm 1 Shared-Online-Adaptive-Learning (SOAL)

Agent-specific initialization: For each learning agent L_a: Let $\mathbf{S}_t^a = \{\mathbf{E}_a^1, \ldots, \mathbf{E}_a^r\}$ be its set of private experts at time $t, t \in \{1, \ldots, T\}, t \leq r \leq 2t$, each $\mathbf{E}_a^j$ being an online optimization algorithm. Let p_t^a be a distribution over $\mathbf{S}_t^a$. Initialize $\mathbf{S}_1^a = \{\mathbf{E}_a^1\}. r = 1, p_1^{a^1} = 1$, for any a. Let f_t^a be the loss function applied to the predictions of L_a.
Shared initialization: Let $\mathbf{H}_t$ be a set of shared experts for all learning agents at time t. Initialize it to one of the private experts $\mathbf{H}_1 = \mathbf{E}_1^1$.
for $t = 1$ *to* T *do*

1. For each learning agent a: (for clarity, we drop the 'a' superscript below)
   a) $\forall j \in \mathbf{S}_t$, set the prediction of the j^{th} expert $x_t^{(j)} = \mathbf{E}^j(f_{t-1})$, compute loss $f_t(x_t^{(j)})$ and update $\mathbf{E}^j$
   b) play $x_t = \sum_{j \in \mathbf{S}_t} p_t^{(j)} x_t^{(j)}$
   c) Multiplicative update: Receive f_t (loss at time t) and perform update for $i \in \mathbf{S}_t$: $\hat{p}_{t+1}^{(i)} = p_t^{(i)} e^{-\alpha f_t(x_t^{(i)})} / \sum_{j \in \mathbf{S}_t} p_t^{(j)} e^{-\alpha f_t(x_t^{(j)})}$
   d) Assign a new contemporary expert $\mathbf{E}^{r+1}$, set $\hat{p}_{t+1}^{(r+1)}$=0
   e) Assign a new historical expert $\mathbf{E}^{r+2}$, set $\hat{p}_{t+1}^{(r+2)} = 0$
   f) Addition step: Set $\overline{p}_{t+1}^{(r+1)}$ and $\overline{p}_{t+1}^{(r+2)}$ to $1/(t+2)$ and for $i \neq r+1$ and $i \neq r+2$, : $\overline{p}_{t+1}^{(i)} = (1-(t+2)^{-1})\hat{p}_{t+1}^{(i)}$
   g) Pruning step: update $\mathbf{S}_t$ to the set $\mathbf{S}_{t+1}$. For all $i \in \mathbf{S}_{t+1}$: $p_{t+1}^{(i)} = \frac{\overline{p}_{t+1}^{(i)}}{\sum_{j \in \mathbf{S}_{t+1}} \overline{p}_{t+1}^{(j)}}$
   h) Add all contemporary experts removed from $\mathbf{S}_t$ to the set of shared experts $\mathbf{H}$.
2. Prune set $\mathbf{H}$ according to Section 5.2.

weights of the previously existing experts are scaled appropriately (*Step 1f*). This yields a new distribution $\overline{p}_{t+1}$. Finally, the distribution over experts in the private set, p_{t+1}, is obtained by removing all experts whose lifetime has expired from the private set and than normalizing $\overline{p}_{t+1}$ over all remaining experts in the set (*Step 1g*).

– *Maintaining all three sets of experts.* At each time step every learning agent adds two nodes to their adaptive layer by adding one historical expert (*Step 1e*) and one contemporary expert (*Step 1d*) to its pool of private experts and sets their lifetimes. The agent also removes nodes from the adaptive layer by removing experts from the private pool whose lifetime has expired (*Step 1g*), and moves them to the set of shared experts (adding nodes to the Shared Knowledge layer, *Step 1h*), keeping their lifetime. Then, shared experts that expired are removed (nodes deleted from the Shared Knowledge layer, *Step 2*).

The SOAL algorithm extends the FLH algorithm (Sect. 4) by adding two sets of experts: the shared set and the adaptive historical set. We now describe in details these two sets. In Sect. 6 we prove the added sets extend the FLH algorithm to make use of shared knowledge, while preserving its original guarantees.

5.2 Shared knowledge layer

Information is shared across learners using 'snapshots' of past experts, represented using a set of shared experts H. This set is dynamic, in the sense that some experts are added and others are removed, but the expert themselves are 'frozen', in the sense that once they are added to the pool they no longer adapt in response to loss.

The shared set is maintained as follows:

Addition Every time an expert retires from the pool of *contemporary* experts of one of the learning agents, it is added to the set of *shared* experts. The expert is assigned a lifetime equal to $c * l$, where l is the initial lifetime parameter the retired expert had in the private pool, and c is a constant (Step 1h in Algorithm 1).

Removal Every time step, t, all lifetimes of *shared* experts are reduced by 1. Experts whose lifetime expired are removed from the shared set (Step 2 in Algorithm 1).

The addition step is repeated for all N learners. Setting the lifetime of each expert added to the shared set to be the same as it was initially in the private set (up to a factor of c), guarantees that the contribution of each learner to the size of the shared set is proportional to the size of its private set which is $O(\log T)$ (see Sect. 4 component 1). Thus the size of the shared set is $O(N \log T)$.

We show below that this logarithmic dependency on T guarantees fast runtime (Sect. 5.4) and quick adaptation to shared events (Sect. 6.2). Section 7 demonstrates empirically that the shared set maintains a shared knowledge in the sense that using it reduces the loss significantly.

Other strategies for maintaining the set could be applied. For example, stronger pruning may not be harmful to frequently-repeating events, since their corresponding experts repeatedly get added to the pool. Prior knowledge on the statistics of event recurrence across learners could lead to tighter regret bounds.

5.3 Historical private experts

The shared set described above captures historical information dating back to the oldest expert in the set.

Given our assumption that the environment is dynamic, we assume that at any time point only some historical information is relevant to a present learner, and the relevance of different portions of the history may change abruptly. To allow learners to quickly find the relevant small number of shared experts that are most relevant at present, each learner maintains a set of historical private experts. Each expert in the historical private set is an online algorithm which learns multiplicative weights over the prediction of the experts [as in Freund et al. (1997)] in the shared set. In Sect. 6.2 we prove that the historical private experts enable quick adaptation in the presence of relevant shared information.

5.4 Runtime analysis

The total runtime of Algorithm 1 is $O(VN\log^2(T))$, where T is the number of time steps, N is the number of learning agents and V is the runtime of a single expert over the data. To see this, use the fact that the size of the private set is $O(\log(T))$ and that the number of historical private experts is half the size of the private set. Also note that at each time step, the runtime of a single historical private expert is $O(VN\log(T))$, where $O(N\log(T))$ is the size of the shared set. As a result, the total runtime of the historical private experts is $O(VN\log^2(T))$. From Hazan and Seshadhri (2007) the runtime of the contemporary experts is $O(V(\log(T))$. Thus the runtime of the historical private experts dominates and the total runtime becomes $O(VN\log^2(T))$.

6 Regret analysis

We now analyze the group adaptive regret (GAR, Definition 1) for Algorithm 1 and present two complementary results. First, we show that SOAL avoids negative-transfer of shared

information by proving that the adaptive regret of SOAL is asymptotically equivalent to that of FLH which does not share information. Second, we show that SOAL can quickly find relevant shared information.

6.1 Safe sharing

The SOAL algorithm is designed to allow *Safe Sharing*, namely, it allows a learning agent to ignore irrelevant information from other learners. We show that SOAL can share safely by proving that it has the same adaptive regret as FLH which does not share information. This means that although sharing introduces the additional complexity of learning over historical experts, it does not hurt the efficient adaptivity obtained with FLH.

Lemma 1 *Let $R_{FLH}(T)$ be the adaptive regret of FLH and $R_{SOAL}(T)$ be the adaptive regret of SOAL. When the loss f is α exp-concave and $\forall x, |f(x)| \leq 1$ the upper bound on R_{FLH} is asymptotically equivalent to that of R_{SOAL}.*

Proof By construction of the SOAL algorithm the size of the private experts set is twice the size of the expert set maintained by the FLH algorithm. As a result, that size is logarithmic in T, as in FLH. In addition, SOAL updates of the working set are the same as in FLH, yielding the same guarantees as in Lemma 3.2 in Hazan and Seshadhri (2009). The update of the weights p_t is closely similar to the FLH updates, with the difference being that the normalization factor in SOAL is $t+2$ instead of $t+1$ in FLH. It is trivial to show that this change does not effect the bound of Lemma 3.1 of Hazan and Seshadhri (2009). Theorem 3.1 in Hazan and Seshadhri (2009) is derived directly from their Lemmas 3.1 and 3.2 and the regret bound of the online convex algorithm $R(T)$. we conclude that the bounds R_{FLH} also apply to R_{SOAL}. □

Intuitively, the Proof of Lemma 1 is due to the fact that SOAL preserves the same set of contemporary experts as in FLH, while adding a compact set of historical experts. The lemma uses this compact size of the historical set (logarithmic in T) together with the assumptions that the α exp-concave loss functions are bounded, and the exponential updates of expert weights to guarantee that "bad" experts from the historical set do not dominate the learning. As a result, even if all experts in the historical set are "bad" SOAL can learn using the experts from the contemporary set, hence avoiding *negative transfer*.

Lemma 1 discusses the adaptive regret; we now show how it yields the 'safe sharing' property for group adaptive regret.

Corollary 1 *Let $R^G_{SOAL}(T)$ be the group adaptive regret (definition 1) and let $R^a_{FLH}(T)$ be the adaptive regret of FLH applied to a learner L_a. The upper bound on $R^G_{SOAL}(T)$ equals the upper bound on $\sum_a R^a_{FLH}(T)$.*

Corollary 1 is a direct consequence of the definition of the group adaptive regret as the sum over individual agent adaptive regret (Definition 1) and of Lemma 1.

We learn from the above that SOAL preserves the same regret bounds as FLH. At the same time, SOAL has the added advantages that it has access to shared information and tracks the private history of each agent. We show below that these additional features improve its learning performance.

6.2 Quick adaptation through sharing

The previous section showed that sharing information in SOAL does not harm the guarantees on adaptive regret of individual learners. We now discuss how SOAL can actually benefit

from using shared information captured in the shared set. Recall the effect of the standard regret $R(T)$ of a single expert on the adaptive regret, Eq. (3). Consider a case where learner L_a has already trained an expert E achieving low regret for the task of a second learner L_b. If this low-regret expert is kept in the shared set, the following lemma shows that the historical private experts have very low standard (non-adaptive) regret to the best expert in the shared set. This is important because the standard regret has a large contribution to the adaptive regret Eq. (3).

Lemma 2 *Consider a set of k experts added to the shared set whose lifetimes in the private set are* $\lambda_1, \ldots, \lambda_k$. *If these experts are pruned using lifetimes* $c\lambda_1, \ldots, c\lambda_k$, *then an online algorithm A which learns only over the shared set using multiplicative updated weights, as in* Freund et al. (1997), *attains a regret of at most* $O(\log(N \log(T)))$ *compared with the best expert in the shared set.*

Proof The pruning procedure described in Sect. 4 component 1 guarantees that the size of the private set is $O(\log(T))$ (Hazan and Seshadhri 2009). Applying the same pruning policy to the shared set and setting the lifetime of each expert added (from any of the N agents) to the set of shared experts to $c\lambda$ yields that the size of the shared set is $O(N \log(T))$. The performance of an online algorithm which learn multiplicative weights over the prediction of experts degrades logarithmically with the number of experts (Freund et al. 1997). Therefore, the regret of A to the best expert in the shared set is $O(\log(N \log(T)))$. □

To emphasize the advantage of learning over the outputs of experts in the shared set, consider the difference between the two types of online learners in the private set: the regular contemporary private learners and the historical private learners. The regular online learners in the contemporary set have a regret bound of $R(T) \leq O(\sqrt{T})$ (or $R(T) \leq O(\log(T))$ in the strong convex case) and this bound cannot be improved (Abernethy et al. 2008). The learners in the historical private set have a regret bound of $O(\log(N \log(T)))$ to the best expert in the shared set (Lemma 2). Thus, the bound on the regret of the historical private experts to the current optimal predictor is $R(T) \leq Z + O(\log(N \log(T)))$ where Z denotes the regret to the optimal predictor of the best expert in the shared set. When Z is large, namely no 'good' expert exists in the shared set, the adaptive learning of SOAL will assign low weights to all historical private learners (see *Safe Sharing* property proved in Lemma 1). When Z is small, namely at least one expert in the shared set captures information that is relevant to the learning task, the historical private learners would quickly converge ($O(\log(N \log(T)))$) to using the outputs of the 'good' experts in the shared set. As a result, SOAL would quickly assign higher weights to the historical private learners relative to the slower ($O(\sqrt{T})$) contemporary learners.

7 Experiments

We first use a synthetic data set to illustrate some of the properties of SOAL, and then evaluate its performance on two real-data tasks: navigating in a visual scene and assigning editors to journals.

7.1 Compared baselines and hyperparameter tuning

We compare the performance of seven learning models.

1. SOAL: *Shared Adaptive Online Learning*, as described in Algorithm 1 above.

2. History-only SOAL (HoSOAL): A variant of SOAL where experts are not shared between learners, hence each expert only learns from its own history.
3. Sharing-only SOAL (SoSOAL): A variant of SOAL where each learner is only aware of the shared experts of other learners and not of its own.
4. FLH: The adaptive algorithm the-Leading-History of Hazan and Seshadhri (2009).
5. Shared-Experts-FLH (SE-FLH): A variant of FLH where each learning agent updates its own set of experts. However, unlike FLH, it learns weights over all experts including experts trained by other agents.
6. MTL-FLH: A variant of FLH where the set of experts are trained jointly using a group-lasso regularization term encouraging joint feature selection, using the optimization approach of Yang et al. (2010).
7. Online: The online dual averaging algorithm of Xiao (2010) with a hinge-loss.

All seven approaches use an online algorithm as a building block for training experts. We chose to use the online dual averaging method of Xiao (2010) with a hinge loss. The value of the α hyper parameter of all SOAL and FLH models was tuned, and the results below are for the best α of each model. We also tuned and selected the best value of the parameter σ which controls the weight of regularization in case sparse feature selection is used, e.g. MTL-FLH model. For SOAL, the lifetime of an expert in the shared set was three times the lifetime of a private expert.

7.2 Synthetic experiments

To illustrate the adaptive and 'safe sharing' properties of SOAL, we created a synthetic binary classification task based on sparse feature selection.

We create five data sequences each with 100 positive samples and 100 negative samples. Each sample is represented using 225 binary features. The ith sequence of samples is drawn from the following conditional distribution: The ith feature is set to '1' for a positive sample and to '-1' otherwise. All remaining features are set uniformly at random to either '1' or '-1'.

We create two learning settings, each with five learners. To make comparisons across methods and across settings easier to interpret, we present performance relative to the Online baseline. Specifically, we show the ratio between the cumulative loss of an algorithm to the cumulative loss of the Online baseline. This ratio is averaged over the five learners in each setting.

Synthetic experiment 1: online setting

As a first experiment we illustrate how SOAL avoids negative transfer when sharing is not useful. Specifically we study the case of stationary online learning, with no benefit in sharing. This setting is simulated by presenting each learner with a single unique sequence of 200 samples created from a its own distribution. Five learners are considered. Figure 3a shows the cumulative loss ratio to the online baseline, for the conditions where information sharing cannot help. SOAL avoids negative transfer ('safe sharing') in the sense that its performance is not significantly inferior to that of FLH and regular online learning even though it enables information sharing in a setting where it cannot help. The performance of Shared-Experts-FLH and MTL-FLH is substantially worse,[3] even compared to the online baseline, with a loss ratio of 4.04 ± 0.74 and 2.11 ± 0.18, respectively. The poor performance of the alternative

[3] Results are omitted from the bar plot to avoid skewing of the data.

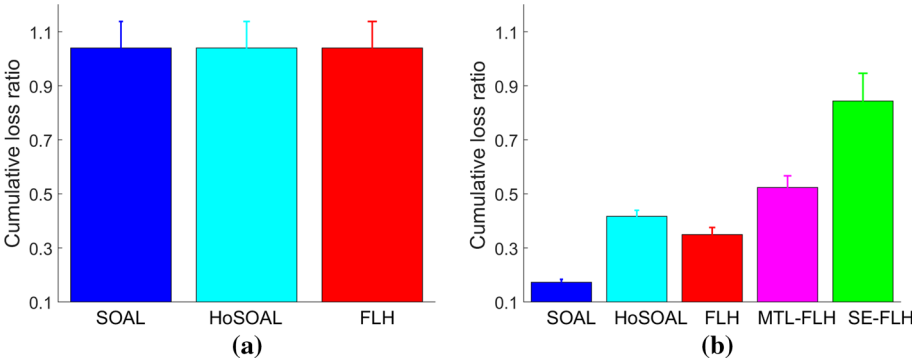

Fig. 3 Synthetic data experiment. The ordinate denotes the average ratio of cumulative loss of each methods compared with the online baseline. **a** Regular online setting data is drawn from stationary distributions and nothing can be gained by sharing. **b** GOAL setting data is drawn from non-stationary distributions hence sharing can help

sharing baselines demonstrates the potential danger in negative knowledge-transfer, which SOAL avoids.

Synthetic experiment 2: GOAL setting

We demonstrate here the superior performance of SOAL in the GOAL learning setting, where data is assumed to come from non stationary distributions and tasks can benefit from sharing. This setting is simulated by training five learning tasks, where each learner is presented with a long sequence of 1000 samples which is divided into five smaller sub-sequences. In total five small sub-sequences where used, each appears in the long sequence of each learner at a different position. For example, learner 1 learned sequences 1, 2, 3, 4, 5 while learner 2 learned the same sequences in a different order 2, 3, 4, 5, 1. Figure 3b shows a significant improvement in the performance of SOAL, which outperforms all other baselines. We also see an improvement in the performance of the two other sharing baselines Shared-Experts-FLH and MTL-FLH compared to their performance in the regular online setting. Indicating that they are actually able to share information where it is beneficial. Yet, their performance is significantly worth compared to SOAL and the other baselines because they are not able to properly learn in adaptive environments. FLH and History-only SOAL on the other hand, are able to quickly adapt but are outperformed by SOAL which is able to share relevant historical information in addition to its quick adaptation. This performance also illustrates that sharing based on the historical experts does not interfere with the adaptation properties, as is the case for Shared-Experts-FLH and MTL-FLH .

7.3 Visual navigation experiment

We used the dataset from Procopio et al. (2009). It contains multiple video scenes, all of which have been shown to contain time-varying (drifting) concepts. Each scene contains a sequence of 100 frames. Each pixel in each frame is labeled as “safe”, “non-safe” or “not-sure”. We only used safe and non-safe labels. The objective is to classify pixels as safe or non-safe. We analyzed the three scenes (1, 2, 3), each with 2 light conditions (A,B), yielding a total of 6 sequences (1A, 1B, 2A, …), which we viewed as six individual tasks. Figure 4 shows an example frame from each of the video sequences.

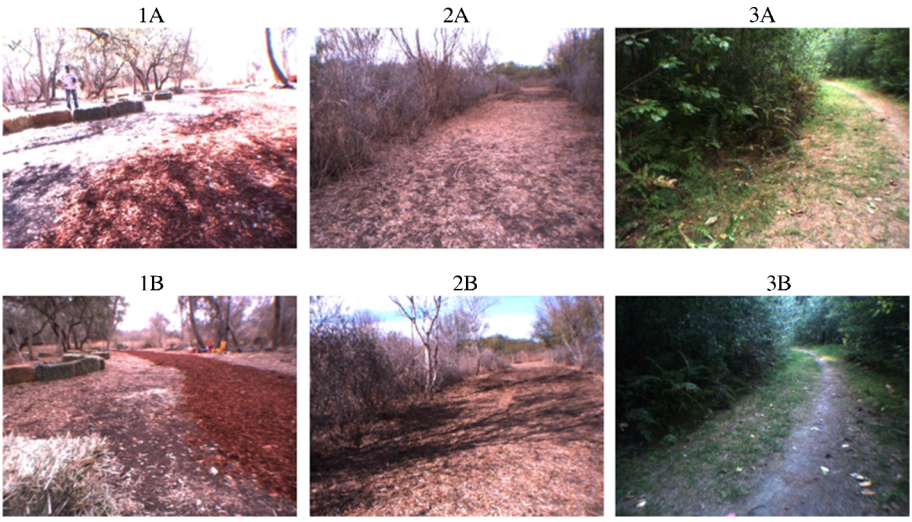

Fig. 4 Samples from the visual navigation database. Shown are representative frames from each of the six video sequences of Procopio et al. (2009). This database contains three different scenes: 1, 2, 3 and two different lighting conditions: **A**, **B**

To represent each pixel, we use the color-histogram features provided with the original dataset. To decrease redundancy we randomly selected up to 50 pixels from each frame, verifying an equal number of positive and negative labels from each frame. At each time step, the learner sees a single pixel. Frames are organized sequentially according to their original order. For computing regret we use the $0/1$ loss.

Formally: consider N learning tasks, $L_1, \ldots, L_N$. Each learner is presented with the task of learning to classify pixels in a video sequence as being "safe" or "non-safe". Each learner views its own private video sequence. Each time step, $t = 1 \ldots T$, all learners view a single pixel $u_t^a \in U^a$ from their private video sequence. Each learner a classifies the pixel as safe or not safe using its hypothesis x_t^a and suffers a loss based on the task specific labels $y_t^a \in \{0, 1\}$.

Vision experiment 1: each agent in a single scene

We started by a simple experiment training six learning agents, each exposed to a different scene. In this setting, sharing would help learning if scenes contain common informative features, and remembering history would help learning if informative features repeat along time. We tested the six adaptive algorithms and computed their regret compared to the naive online algorithm.

Figure 5a shows the ratio between the cumulative loss of each algorithm to that of the cumulative loss of the online baseline. SOAL and History-only SOAL achieve the lowest loss, strongly outperforming the other baselines. MTL-FLH encourages all learners to share weights at every timestep, which apparently is inadequate for this data, hurting the regret compared to standard FLH. In this setting, learning from historical experts is useful, helping History-only SOAL to slightly outperform SOAL. However, the regret obtained by Sharing-only SOAL, shows that even when learners are prevented from using their own history, and forced to use only shared experts their performance is significantly boosted compared

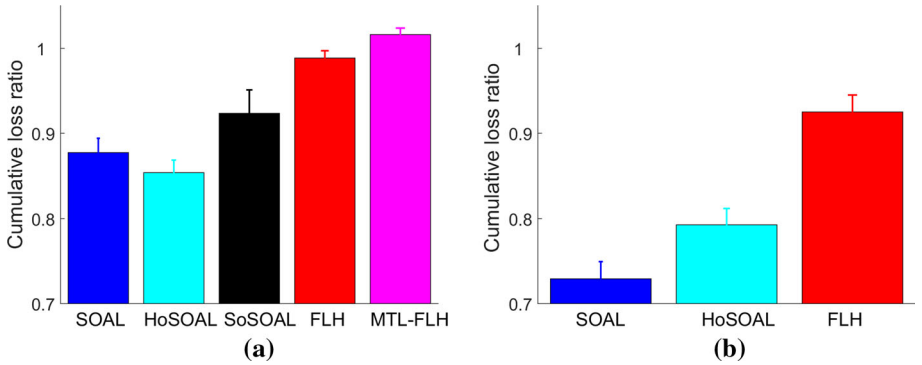

Fig. 5 Original scenes and random sequences of scenes. The ordinate denotes the average ratio of cumulative loss of each method compared with the online baseline. **a** Parallel learning in six agents, each agent exposed to a single scenes. *Error bars* denote the standard error of the mean (SEM) across 6 agents. **b** Learning in parallel random sequences. *Error bars* denote SEM over 10 agents, each exposed to its own random sequence of scenes

to no sharing. The performance of Shared-Experts-FLH is substantially worse,[4] 2.24 ± 0.32.

Vision experiment 2: random sequences of scenes

To further test the potential benefit of sharing among agents, we created a second learning setting, where each agent was presented with a random sequence of scenes. Sequences of five scenes were created by sampling uniformly with repetitions from the six available scenes. Similar results are obtained when sampling longer sequences, but are not shown here for clarity. A total of ten agents were trained in parallel. Each agent learns his own random sequence.

Figure 5b shows the relative cumulative loss compared to the online baseline, computed by summing over the entire sequence of a learner and averaging across the 10 learning agents. Again the SOAL and History-only SOAL achieve the lowest loss, strongly outperforming FLH.

Figure 6 shows four representative examples of random sequences. The left column shows the cumulative loss as a function of time for 4 learning methods. The right column traces the maximal weight of a shared expert during learning. This weight is computed by summing over all historical experts in the private set, while multiplying their own weight with the weight they give to the expert. These examples show how the difference between the cumulative loss grows as time progresses in favor of SOAL and History-only SOAL compared to the regular Online and FLH baselines. Typically SOAL also significantly outperforms History-only SOAL. This behavior is correlated with the high weight of a single expert in the shared set.

Vision experiment 3: switching losses, quick adaptation

We further studied the effect of sharing by analyzing the loss around the switch between scenes. The results above suggest that SOAL successfully uses history and shared information

[4] Results are omitted from the bar plot to avoid skewing of the data.

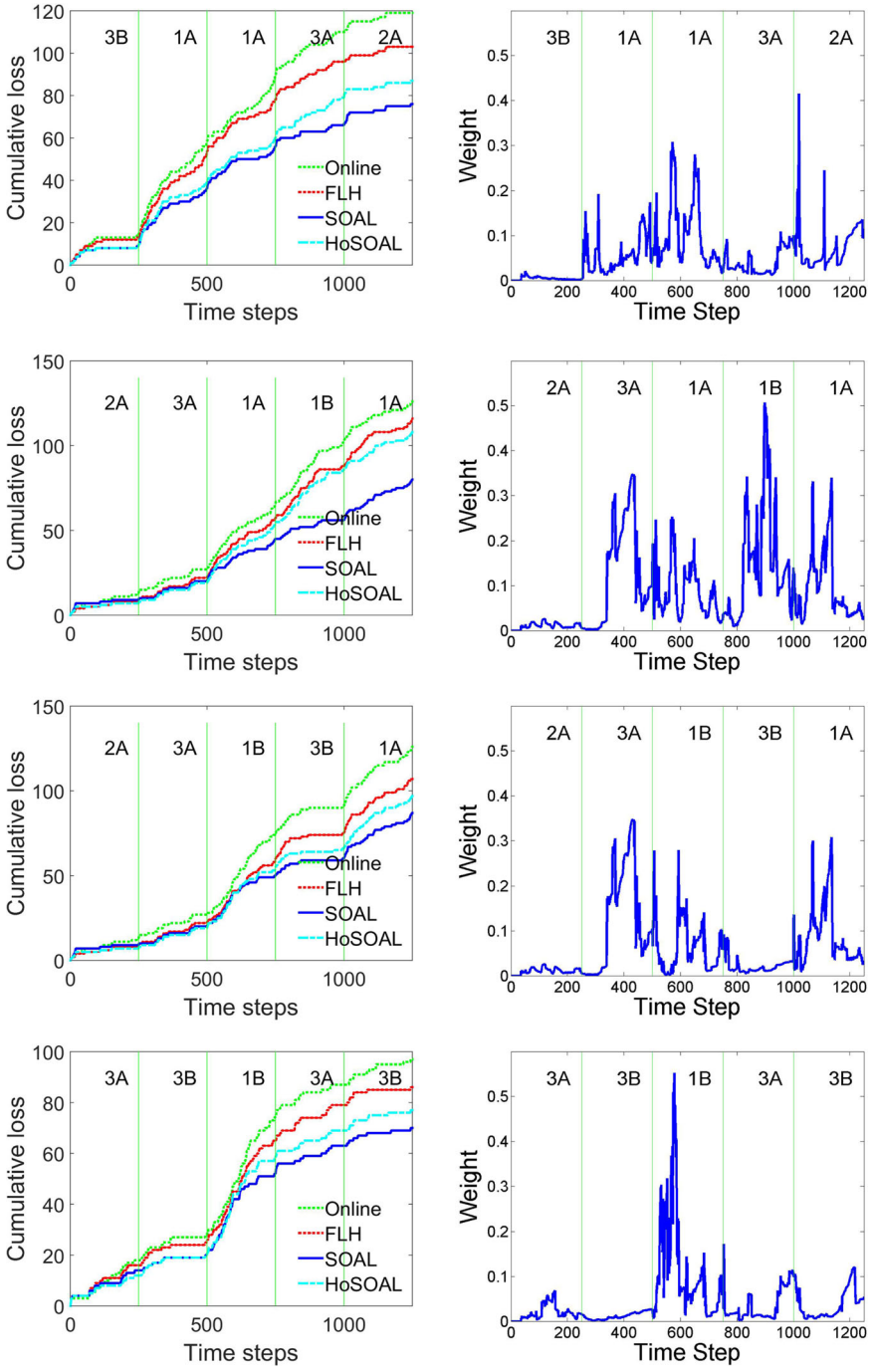

Fig. 6 Learning with a random sequence of scenes. *Left* A trace of the cumulative loss of four algorithms. Example of four representative random sequences. *Each row* corresponds to a different sequence. *Vertical lines* mark switches between scenes. *Right* The maximal weight of an expert in the shared set during the learning of SOAL

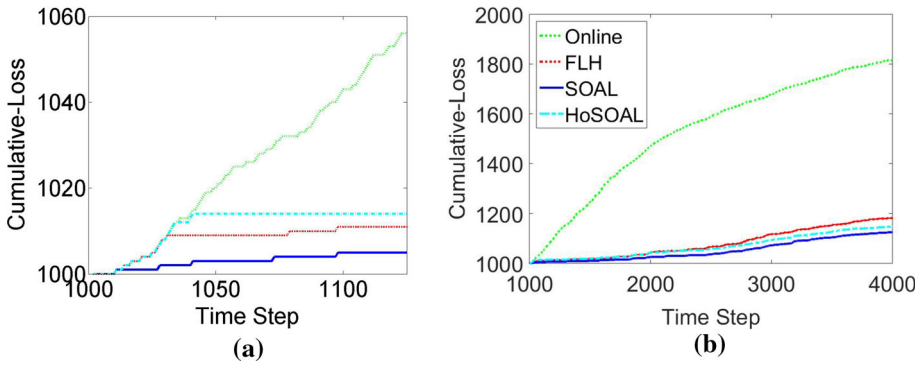

Fig. 7 Fast adaptive learning. **a** The cumulative loss of four algorithms shortly after switching to a new loss function. **b** The cumulative loss as in (**a**), but showing 4000 steps

to improve learning. To study sharing in details, we focused on a specific setting where two agents L_A, L_B are given two related sequences recorded in the same physical scene under different light conditions (Procopio et al. 2009).

For the first 1000 steps, learner L_A was trained to minimize an arbitrary loss function f_{rnd}, given the sequence $1A$. At the same time, learner L_B was trained to predict pixel safety on a sequence $1B$. At time 1000 we switched the loss of L_A to predict pixel safety for its sequence.

Figure 7 traces the cumulative loss of agent L_A trained with one of 4 algorithms, around the transition point and for a longer duration after the transition point. The naive online algorithm fails to capture the switch, while the other approaches do. Importantly, SOAL succeeds to switch much faster than the other limited versions, since it utilizes the set learned by L_B with the $1A$ scene. We also note that after longer training Historical-SOAL successfully uses past events from the 1A sequence, and outperforms FLH.

7.4 Editor-assignment experiment

We consider the task of assigning editors to review JMLR papers. When a new paper is submitted to a journal, the journal needs to assign an editor with relevant expertise to oversee the reviewing process. This is an online adaptive learning problem where the journal receives a continuous stream of papers, whose distribution changes in time.

We view this online assignment task as an instance of the GOAL setting because it has the two characteristics defining GOAL: the environment is non-stationary and shared information among editors. The environment is non stationary in two related aspects. First, many editors change their fields of interests over time. Second, the distribution of topics changes with the advance of the research.

Often different editors have overlapping research fields and some editors gain experience in those fields before others. As a result, the latter may benefit from exploiting information previously learned by other editors.

7.4.1 Data collection and pre-processing

We downloaded all papers in volumes 1–14 from the website of the journal of machine learning research http://www.jmlr.org. We removed papers which did not list an editor or a

submission date in standard format, leaving 910 papers which were accepted to the journal. After removing the name of editors from the papers, we represented each paper using a standard tf/idf representation based on a dictionary. To create the dictionary, we used a standard list of stop words, required a minimum length of 3 characters per word, at least 3 occurrences of a string in a document and we applied stemming (Porter 1980). We further reduced dimensionality using the unsupervised approach of Zeimpekis and Gallopoulos (2005), resulting with a sparse 5000 dimensionality representation.

7.4.2 The learning task

Editors may have overlapping areas of expertise and as a consequence, a paper could be assigned to more than one editor. Thus, we choose to model the assignment problem as a set of binary classification tasks, one task per editor. For each editor, we trained a single binary classifier, to determine if each paper is relevant to an editor or not. To create the stream of papers, papers were sorted according to submission date. At each time step, a single paper is presented to each editor–learner according to the sorted order.

Formally: consider N learning tasks, $L_1, \ldots, L_N$, each corresponding to a specific editor. Each time step, $t = 1 \ldots T$, a paper $u_t \in U$ is shown to all learners. Each learner a classifies the paper as relevant or not to its editor using its hypothesis x_t^a and suffers a loss based on the task specific labels $y_t^a \in \{-1, 1\}$.

As ground-truth labels, we used the actual JMLR editor assignments, collected and processed from the JMLR website. For the positive set, we used the JMLR papers edited by the editor and for the negative set we used all JMLR papers which were not edited by the editor. Occasionally a paper is edited by more than one editor, and is in the positive set of all its editors. We use the 0–1 loss, where an online learner suffers a loss of '1' when a paper is classified as relevant to an incorrect editor. Otherwise, the loss is '0'. We chose this loss because we only have partial data about which editor is relevant to a paper. Clearly, not all papers relevant to an editor were assigned to him by JMLR.

To ensure sufficient positive samples per editor, we chose editors who have edited 20 or more papers. In total, there are six such editors: *Gabor Lugosi* (GB), *John Shawe-Taylor* (JS), *Manfred Warmuth* (MW), *Michael Jordan* (MJ), *Peter Bartlett* (PB), *Tommi Jaakkola* (TJ), each with 25, 40, 22, 21, 31, 23 positive papers respectively.

Each learner learns using all 910 papers. This results in a substantially unbalanced learning setting with approximately 22–45 times more negatives than positives, i.e. an average label ratio of approximately 1:30. We also used a more balanced settings where 600 negative samples were selected randomly and removed from each learner's sample set, achieving an average positive-to-negative ratio of approximately 1:10.

7.4.3 Results

Performance comparison. We find that some methods do not perform well with highly unbalanced data, and tend to reject most positive samples. In unbalanced datasets with few positives, the trivial classifier which rejects all samples would achieve a low loss. We therefore set the following criteria for a method to be included in the loss comparison. We require that it achieves a minimum of chance level for positive classification. Achieving lower level of positive classification indicates that the method simply learns to reject due to the highly unbalanced data. We therefore avoid showing curves for MTL-FLH and Online baseline which reject all positive samples and SE-FLH which rejects 66% of the positive samples.

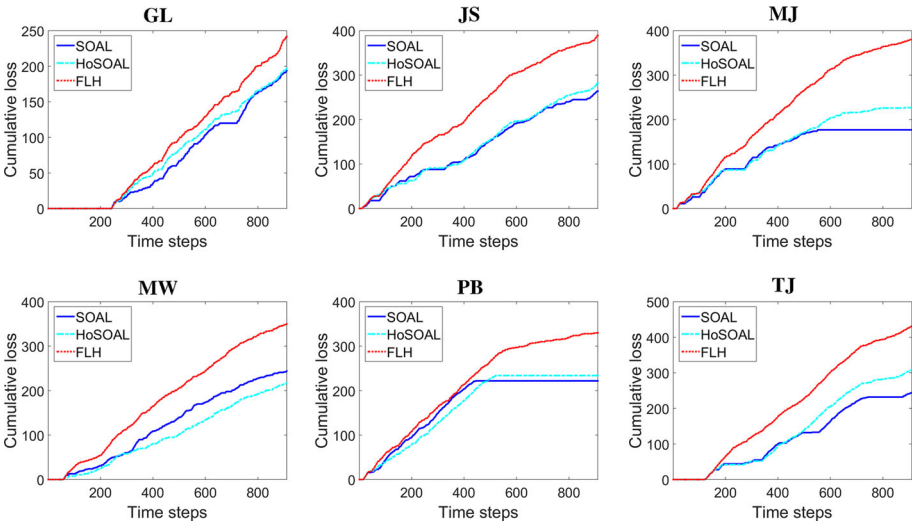

Fig. 8 Editor assignment cumulative loss. Cumulative loss comparison for all six editor learners computed over the entire JMLR dataset (910 papers)

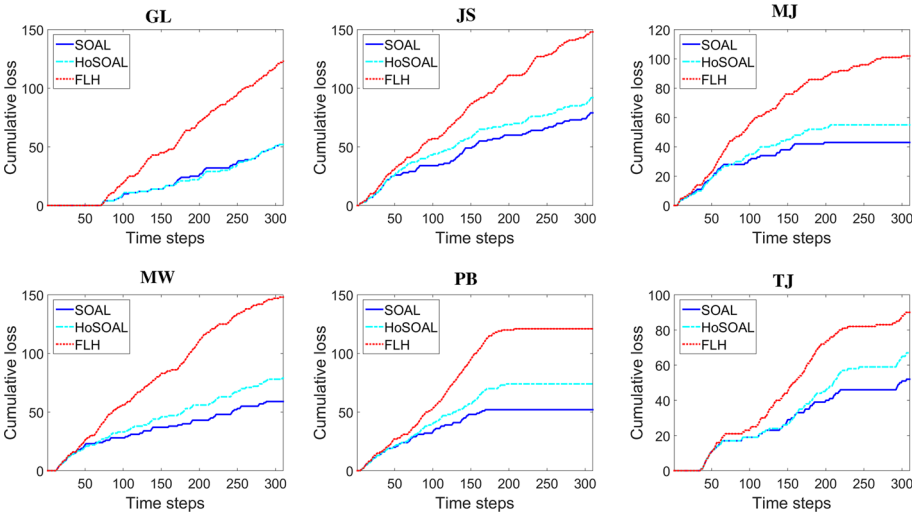

Fig. 9 Editor assignment cumulative loss for 1:10 condition. Cumulative loss comparison for all six editor learners computed over a dataset with an average positive to negative ratio of 1:10. SOAL outperforms all competing methods

Figure 8 compares the cumulative loss of three competing methods, we see that SOAL, History-only SOAL always outperform FLH. In Fig. 9 we show the cumulative loss for all editors in the 1:10 ratio experiment. In this experiment the unbalanced nature of the data was reduces by a factor of 3, from an average ratio of approximately 1:30 positive to negative samples, to an average ratio of approximately 1:10. We see that SOAL outperforms all other methods.

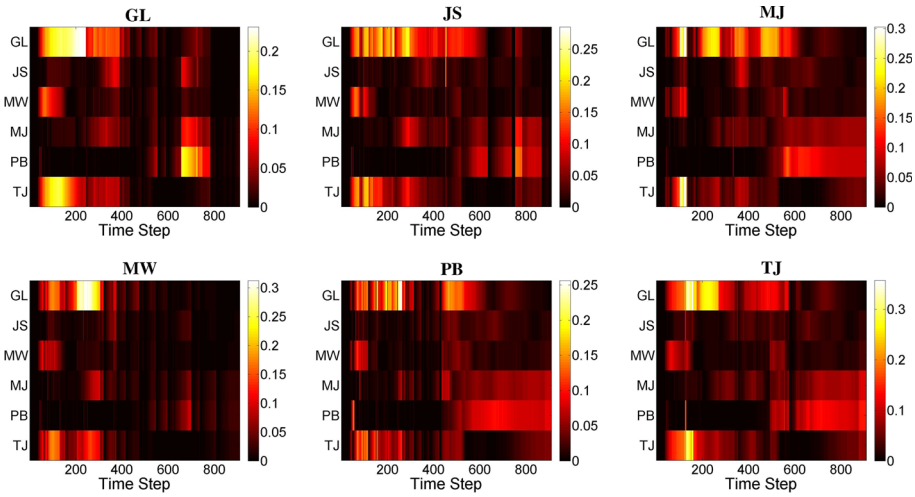

Fig. 10 Dynamics of weight assigned to shared experts. A heat map showing the strength of sharing among editors. Results are shown for the entire JMLR dataset and the six chosen editors. *Colors* correspond to the sum of weights that editors assign to each of the other editors. Summation is over the weights of all retired experts in the shared set corresponding to the editor which trained them. The y-axis corresponds to editors (Color figure online)

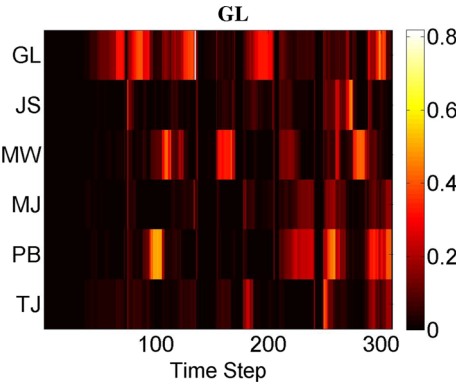

Fig. 11 Discovering related editors. A heat map showing the strength of sharing among editors in the 1:10 ratio condition. *Colors* correspond to the sum of weights that editor *GL* assigns to each of the editors. Summation is over the weights of all retired experts in the shared set corresponding to the editor which trained them. The y-axis corresponds to editors. *GL* assigns a high value to *PB* around time step 100. This interval in time corresponds to a paper *GL* edited and was co-authored by *PB* and to another paper which *GL* edited and was co-authored by an author for which *PB* edited a previous paper (Color figure online)

Dynamics of sharing in the JMLR full set experiment. Figure 10 presents the sum of weights that editors assign to each of the other editors. Summation is over the weights of all retired experts in the shared set corresponding to the editor which trained them. The learners take advantage of these experts, as can be seen by the weight values which are set away from zero during most of the time steps. All learners give experts retired from *GL*, high weight during the first 200 learning steps. *GL* sees only negative samples during these learning steps. This indicates that all learners share *GL's* knowledge on negative classification.

Discovering related editors. Figure 11 shows the sum of weights that editor *GL* assigns to each of the editors. Summation is over the weights of all retired experts in the shared set corresponding to the editor which trained them. The high value on the *PB* row around step 100 is because *GL* learns that editor *PB* becomes very relevant when *GL* is presented with sample number 98, which is a positive sample of a paper edited by *GL* and written by *PB* (Bartlett and Tewari 2007). *GL* keeps *PB's* high weight also when presented with another positive sample, sample number 100 (Micchelli et al. 2006) which shares the same co-author as a paper previously reviewed by *PB* (Micchelli and Pontil 2005).

8 Summary

We presented an algorithm addressing a novel learning setting where learning agents learn jointly in separate changing environments. This setting applies to many real-world scenarios where information in data-streams repeats in time and across learners. Our approach addresses two challenging tasks in a single framework: online learning of changing environments and joint learning across tasks. Traditional joint learning approaches, which assume the relation among tasks to be fixed, find it hard to model relations that change while the environment changes as well.

We define *group adaptive regret* and prove that when learners adapt jointly our algorithm maintains the upper bound over the adaptive regret of each individual learner. In this sense, we achieve *safe sharing* where the adaptive approach guards the learner from *negative transfer*. In addition, the algorithm only saves a compact (logarithmic) sketch of the shared information which enables quick adaptation in cases where shared information becomes beneficial. It remains an interesting future work to analyze the statistical conditions under which minimizing the *group adaptive regret* leads to minimization of the adaptive regret of individual learners.

References

Abernethy, J., Bartlett, P. L., Rakhlin, A., & Tewari, A. (2008). Optimal strategies and minimax lower bounds for online convex games. In: *Proceedings of the nineteenth annual conference on computational learning theory*.

Adamskiy, D., Warmuth, M. K., & Koolen, W. M. (2012). Putting Bayes to sleep. In: *Advances in neural information processing systems* (pp. 135–143).

Ang, H. H., Gopalkrishnan, V., Zliobaite, I., Pechenizkiy, M., & Hoi, S. C. (2013). Predictive handling of asynchronous concept drifts in distributed environments. *IEEE Transactions on Knowledge and Data Engineering*, *25*(10), 2343–2355.

Argyriou, A., Evgeniou, T., & Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, *73*(3), 243–272.

Bartlett, P. L., & Tewari, A. (2007). Sparseness vs estimating conditional probabilities: Some asymptotic results. *The Journal of Machine Learning Research*, *8*, 775–790.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*(1), 41–75.

Cavallanti, G., Cesa-Bianchi, N., & Gentile, C. (2010). Linear algorithms for online multitask classification. *The Journal of Machine Learning Research*, *11*, 2901–2934.

Dredze, M., & Crammer, K. (2008). Online methods for multi-domain learning and adaptation. In: *Proceedings of the conference on empirical methods in natural language processing, association for computational linguistics (ACL)* (pp. 689–697).

Freund, Y., Schapire, R. E., Singer, Y., & Warmuth, M. K. (1997). Using and combining predictors that specialize. In: *Proceedings of the twenty-ninth annual ACM symposium on theory of computing* (pp. 334–343). ACM.

Gabel, M., Keren, D., & Schuster, A. (2015). Monitoring least squares models of distributed streams. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 319–328). ACM.

Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, *46*(4), 44.

Hazan, E., & Seshadhri, C. (2007). Adaptive algorithms for online decision problems. In: *Electronic colloquium on computational complexity (ECCC)* (Vol. 14, Issue 088).

Hazan, E., & Seshadhri, C. (2009). Efficient learning algorithms for changing environments. In: *Proceedings of the 26th annual international conference on machine learning* (pp. 393–400). ACM.

Hazan, E., et al. (2016). Introduction to online convex optimization. *Foundations and Trends ® in Optimization*, *2*(3–4), 157–325.

Herbster, M., & Warmuth, M. K. (1998). Tracking the best expert. *Machine Learning*, *32*(2), 151–178.

Kamp, M., Boley, M., Keren, D., Schuster, A., & Sharfman, I. (2014). Communication-efficient distributed online prediction by dynamic model synchronization. In: *Machine learning and knowledge discovery in databases: European conference, ECML PKDD* (pp. 623–639). Springer.

Lugosi, G., Papaspiliopoulos, O., & Stoltz, G. (2009). Online multi-task learning with hard constraints. arXiv preprint arXiv:0902.3526.

Micchelli, C. A., & Pontil, M. (2005). Learning the kernel function via regularization. *Journal of Machine Learning Research*, *6*, 1099–1125.

Micchelli, C. A., Xu, Y., & Zhang, H. (2006). Universal kernels. *The Journal of Machine Learning Research*, *7*, 2651–2667.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.

Procopio, M. J., Mulligan, J., & Grudic, G. (2009). Learning terrain segmentation with classifier ensembles for autonomous robot navigation in unstructured environments. *Journal of Field Robotics*, *26*(2), 145–175.

Ruvolo, P., & Eaton, E. (2014). Online multi-task learning via sparse dictionary optimization. In: *AAAI conference on artificial intelligence (AAAI-14)*.

Saenko, K., Kulis, B., Fritz, M., & Darrell, T. (2010). Adapting visual category models to new domains. In: *Computer vision–ECCV 2010* (pp. 213–226). Springer.

Saha, A., Rai, P., Venkatasubramanian, S., & Daume, H. (2011). Online learning of multiple tasks and their relationships. In: *International conference on artificial intelligence and statistics* (pp. 643–651).

Sugiyama, M., Krauledat, M., & Müller, K. R. (2007). Covariate shift adaptation by importance weighted cross validation. *The Journal of Machine Learning Research*, *8*, 985–1005.

Thrun, S., & Mitchell, T. M. (1995). *Lifelong robot learning*. Berlin: Springer.

Tommasi, T., Orabona, F., & Caputo, B. (2010). Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In: *2010 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3081–3088). IEEE.

Xiao, L., et al. (2010). Dual averaging methods for regularized stochastic learning and online optimization. *The Journal of Machine Learning Research*, *11*, 2543–2596.

Yang, H., Xu, Z., King, I., & Lyu, M. R. (2010). Online learning for group lasso. In: *Proceedings of the 27th international conference on machine learning (ICML)* (pp. 1191–1198).

Zeimpekis, D., & Gallopoulos, E. (2005). CLSI: A flexible approximation scheme from clustered term-document matrices. In: *Proceedings on SIAM data mining conference* (pp. 631–635). SIAM.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In: *Proceedings of the 20th annual international conference on machine learning (ICML)*. School of Computer Science, Carnegie Mellon University.

Zliobaite, I. (2009). *Learning under concept drift: An overview*. Technical report. Vilnius University.

Zweig, A., & Weinshall, D. (2007). Exploiting object hierarchy: Combining models from different category levels. In: *IEEE 11th international conference on computer vision (ICCV)* (pp. 1–8). IEEE.

Zweig, A., & Weinshall, D. (2013). Hierarchical regularization cascade for joint learning. In: *Proceedings of the 30th international conference on machine learning (ICML)* (pp. 37–45).