Multitask Coactive Learning

Robby Goetschalckx **Alan Fern** **Prasad Tadepalli**
School of Computer Science
Oregon State University
Corvallis, OR 97330
goetschr, afern, tadepall@eecs.oregonstate.edu

Abstract

In this paper we investigate the use of coactive learning in a multitask setting. In coactive learning, an expert presents the learner with a problem and the learner returns a candidate solution. The expert then improves on the solution if necessary and presents the improved solution to the learner. The goal for the learner is to learn to produce solutions which cannot be further improved by the expert while minimizing the average expert effort. In this paper, we consider the setting where there are multiple experts (tasks), and in each iteration one expert presents a problem to the learner. While the experts are expected to have different solution preferences, they are also assumed to share similarities, which should enable generalization across experts. We analyze several algorithms for this setting and derive bounds on the average expert effort during learning. Our main contribution is the balanced Perceptron algorithm, which is the first coactive learning algorithm that is both able to generalize across experts when possible, while also guaranteeing convergence to optimal solutions for individual experts. Our experiments in three domains confirm that this algorithm is effective in the multitask setting, compared to natural baselines.

1 Introduction

Coactive learning [Raman *et al.*, 2013a; 2012; 2013b; Shivaswamy and Joachims, 2012] is a Machine Learning framework where the learner is presented with a sequence of problems and for each problem constructs a candidate solution. The expert then attempts to improve the solution if it is not of sufficient quality and shows the improved solution to the learner. The learner needs to discover a good utility function to reflect the expert's preferences over potential solutions, ultimately leading to a decrease in the amount of effort the expert needs to spend on improving candidate solutions. For example, consider a route planner which tries to adapt to the specific user's preferences over whether to take fast routes, easy routes or scenic routes. The system presents the user with a candidate solution and the user can then modify this trajectory according to their personal preferences. The learner then attempts to learn those preferences to improve future performance.

For a task such as route planning, the same system might be used by a number of different users who might have similar but non-identical utility functions. Also, one could consider different types of tasks: people might have different preferences for routes for different purposes (e.g. business vs. pleasure). Ideally, the system should learn each individual user's preferences, or learn to solve each individual task, while taking advantage of the similarities between the users and tasks to learn quickly.

The above motivates the study of coactive learning in a *multitask* setting. Specifically, in this paper, it is assumed that the sequence of tasks is generated by individuals from a user population. The learner is provided with the user identifier for each task, which can be used to customize task solutions to individuals. A trivial approach to this multitask learning problem would be to have an independent coactive learner for each user. However, when the preferences of different users have similarities, some amount of generalization can be expected to improve overall performance compared to independent learning. At the same time, if the users have little in common we would like the multitask learner to not perform much worse than independent learning.

The main contribution of this paper is to present and analyze a Perceptron algorithm for multitask coactive learning (the MCL algorithm). This is the first such algorithm that meets both of the above goals. Our bounds on the average expert cost during learning show that the algorithm is able to guarantee convergence to optimal solutions for individual experts, even when experts are not similar. At the same time, the algorithm can be expected to benefit from generalization across the expert population when experts are sufficiently similar. We also provide bounds for two extreme algorithms: one that treats all experts as if they are a single expert, and a second that learns independently for each expert. These bounds provide insight into when generalization can be beneficial compared to independent learning. The theoretical bounds are confirmed by our experimental evaluation in three domains. These experiments show that the balanced Perceptron algorithm is robust across experts of varying similarity and can significantly improve on the baseline algorithms.

2 Related Work

The most closely related prior work considered a multiuser coactive learning setting [Raman and Joachims, 2013]. However, the learner was not provided with a user identifier and simply treated all tasks as if coming from a single user. In general, such an approach will not be able to converge to a solution that works well for all users, especially when user groups have very different preferences. Rather, in our setting, by providing the learner with user identifiers, it is possible to converge to optimal solutions for individual users, while also exploiting commonalities when possible.

Multitask learning has been more widely studied outside of the coactive framework (e.g. classification). However, in most of the existing work it is either assumed that an entire dataset of examples is given (offline setting) [Evgeniou *et al.*, 2005; Evgeniou and Pontil, 2004] which allows for estimating the extent to which the tasks are similar, so that highly similar tasks can be clustered and learned together, or (as in [Cavallanti *et al.*, 2010]) the assumption is made that a matrix is given expressing the similarity between tasks. This matrix is used as a regularizer for an online linear classification algorithm. In this paper, since we do not assume any domain knowledge to be given regarding the similarity of individual users, we will focus on algorithms that attempt to generalize across users, while proving worst-case bounds that are not much worse than independent learning, even when users have no similarities.

In recent work, online approaches for multitask classification learning have been studied to learn and leverage task similarities in a way that is closely related to sparse coding [Ruvolo and Eaton, 2013; Maurer *et al.*, 2013; Ruvolo and Eaton, 2014]. Extensions of these approaches have been demonstrated in the context of temporal difference learning [Sreenivasan *et al.*, 2014] and policy gradient reinforcement learning [Bou-Ammar *et al.*, 2014]. However, none of these algorithms have been applied to coactive learning, where the optimization criterion is to reduce the user effort in improving the system's solutions. In this paper, we study an alternative approach based on Perceptron-style online learning.

3 Problem Statement and Notation

We consider a problem-solving setting, where X is a set of problem instances, and Y is a set of candidate solutions. A problem-solution pair will be described by a feature vector $\vec{\phi} : X \times Y \to \mathbb{R}^D$ where D is the dimension of the feature vectors. We assume that $\forall x, y : ||\vec{\phi}(x, y)|| \leq R$. We consider a multitask setting where there are M experts, or users, who will be providing problems to our learner. Each expert has a linear preference function, such that the preference of expert i for solution y to problem x is given by $\vec{u}_i^* \cdot \vec{\phi}(x, y)$, where we assume that $||\vec{u}_i^*|| = 1$. Note that this same framework can be used for a single expert, solving a sequence of tasks having M task types. We assume that there is a black-box problem-solver $\texttt{solve}(x, \vec{w}) = \text{argmax}_y \, \vec{\phi}(x, y) \cdot \vec{w}$ available, which takes a problem $x \in X$, an estimated weight vector $\vec{w}$ and returns a candidate solution $y \in Y$, which is optimal for the given estimate of the utility function[1].

At time-step t, the learning algorithm receives a problem instance $x_t \in X$ and an expert index $a(t)$. We assume the expert indices are sampled from an (unknown) fixed multinomial distribution $P(i)$. The algorithm uses its current estimate $\vec{w}^t_{a(t)}$ of expert $a(t)$'s weight vector to construct a solution, $y_t = \texttt{solve}(x_t, \vec{w}^t_{a(t)})$.

The candidate solution y_t is presented to expert $a(t)$, who either accepts it as good enough (at no cost) or spends an amount of effort improving the solution, obtaining a solution y'_t for which $\vec{u}^*_{a(t)} \cdot \vec{\phi}(x_t, y'_t) \geq \vec{u}^*_{a(t)} \cdot \vec{\phi}(x_t, y_t) + \kappa C_t$, where C_t is the *cost*, reflecting the amount of effort spent, and κ is a constant indicating the minimal return on investment, the minimal improvement in solution quality to vindicate the effort spent. We assume that $C_t \geq 0$. In this paper, the notation $\vec{\Delta}_t$ is used to indicate $\vec{\phi}(x_t, y'_t) - \vec{\phi}(x_t, y_t)$, so we get: $\vec{u}^*_{a(t)} \cdot \vec{\Delta}_t \geq \kappa C_t$. Since we have the bound of R on the feature vectors, we know that $||\vec{\Delta}_t|| \leq 2R$. The task is to minimize the average effort $\frac{1}{T}\sum_{t=1}^{T} C_t$.

We characterize the similarity of experts by a parameter δ that satisfies: $\forall i, j : \vec{u}_i^* \cdot \vec{u}_j^* \geq 1 - \delta$. Lower values of δ mean that the experts are more similar. We will denote the difference between expert preference weight vectors by $\vec{\xi}_{i,j} = \vec{u}_i^* - \vec{u}_j^*$. It will be useful to note that $\forall i, j : ||\vec{\xi}_{i,j}|| \leq \sqrt{2\delta}$, which follows from the law of cosines.

4 Balanced Perceptron-based Learner

In our framework, the learning problem consists of learning appropriate preference functions represented by weight vectors for all experts. In this section, we focus on learning algorithms that attempt to generalize across all experts. The performance of such algorithms will naturally depend on the similarity of experts as characterized by δ.

Each of the algorithms we consider are instances of a single algorithm, called Multitask Coactive Learner (MCL) (see algorithm 1) that is parameterized by parameters α and β, which control the amount of generalization across experts. The MCL algorithm maintains and updates a single global weight vector $\vec{w}_G$ and expert-specific weight vectors $\vec{w}_i$. Intuitively $\vec{w}_G$ is intended to capture the commonality among experts in order to support generalization, while $\vec{w}_i$ is intended to capture user-specific preference variations. After a problem instance for expert i is processed, resulting in feedback from expert i, the weight vector $\vec{w}_i$ is updated along with the global weight vector $\vec{w}_G$ (details below). The MCL algorithm starts with all expert-specific vectors and global vectors $\vec{w}_i^1 = \vec{w}_G^1 = \vec{0}$. Whenever we need to estimate the weight vector for expert i at time t, in order to produce a solution, the sum $\alpha \vec{w}_i + \beta \vec{w}_G$ is used.

[1]Using an analysis similar to that presented in [Goetschalckx *et al.*, 2014], this can be changed into a *locally* optimal solver, where the precise definition of "locally optimal" depends on both the black-box solver and the possible expert improvements. In the "path planning" experiments presented in this paper, only such local optimality is practical. The theoretical results in Section 5 still hold given the extra assumptions from [Goetschalckx *et al.*, 2014].

Algorithm 1 Multitask Coactive Learner (α, β)

```
$\vec{w}_G \leftarrow \vec{0}$
$\forall i : \vec{w}_i \leftarrow \vec{0}$
loop
    $(x, a(t)) \leftarrow$ new problem instance and userID
    $y \leftarrow \texttt{solve}(x, \alpha\vec{w}_{a(t)} + \beta\vec{w}_G)$
    $y' \leftarrow \texttt{improve}_{a(t)}(x, y)$
    $\vec{\Delta} \leftarrow \phi(x, y') - \phi(x, y)$
    if $\vec{\Delta} \cdot (\alpha\vec{w}_{a(t)} + \beta\vec{w}_G) \leq 0$ then
        $\vec{w}_{a(t)} \leftarrow \vec{w}_{a(t)} + \vec{\Delta}$
        $\vec{w}_G \leftarrow \vec{w}_G + \vec{\Delta}$
    end if
end loop
```

It remains to specify how the weights are updated by the algorithm. When a problem instance is processed with $\vec{\Delta}_t \neq \vec{0}$ (i.e., the expert was able to improve the learner's returned solution), both the active expert $a(t)$'s weight vector and the global weight vector are updated by a weighted Perceptron update $\vec{w}^{t+1}_{a(t)} = \vec{w}^{t}_{a(t)} + \vec{\Delta}_t$ and $\vec{w}^{t+1}_G = \vec{w}^t_G + \vec{\Delta}_t$. Thus, in addition to updating the total weight vector of user $a(t)$, the update impacts the total weight vectors of other users through the adjustment to $\vec{w}_G$. [2]

In general, the specific values used for α and β allow for a range of algorithms. We consider the following cases.

- **`global`** ($\alpha = 0, \beta = 1$). In this case, all experts are treated as if they were the same. This results in the perceptron-based approach presented as Algorithm 2 in [Raman and Joachims, 2013]. Clearly, unless all expert preferences are identical or the multinomial distribution $P(\cdot)$ is degenerate, meaning the same expert is always selected, this approach can never converge to the actual weight vectors.
- **`individual`** ($\alpha = 1, \beta = 0$). This means no generalization is performed, and we essentially have M independent perceptron-based coactive learners as presented in [Shivaswamy and Joachims, 2012]. This is guaranteed to learn the optimal weights eventually, but ignores any potential similarity between experts.
- **`balanced`** ($\alpha = \beta = 1$). This leads to a combined approach, both generalizing over different users yet at the same time specializing per user. This is equal to a coactive variant of the algorithm based on Section 3.2 of [Cavallanti *et al.*, 2010].

5 Average Cost Bounds

In this section, bounds on the average cost (expert effort) $\frac{1}{T}\sum_{t=1}^{T} C_t$ will be presented for the algorithms described in Section 4.

[2]In the case where `solve` is not guaranteed to give the local optimum (see Footnote 1), the update should only be performed if $\langle \alpha\vec{w}^t_{a(t)} + \beta\vec{w}^t_G, \vec{\Delta}_t\rangle \leq 0$); in the case where `solve` gives the global optimum, this condition is always satisfied.

5.1 `global`

We first consider bounding the average cost of **`global`**, which does not attempt to distinguish among different experts. In this case, it is impossible to converge to a solution with average effort equal to 0, unless all experts are exactly the same. The badness of the learned weight vector depends on the number of experts (M) and the difference between experts (δ) as quantified by the following result.

Theorem 1. *Using the* **`global`** *algorithm, the average effort for the first T examples is bound by* $\frac{1}{T}\sum_{t=1}^{T} C_t \leq \frac{2R}{\kappa}\left(\frac{1}{\sqrt{T}} + \sqrt{2\delta}\frac{M-1}{M}\right)$.

Proof. First, an upper bound on $||\vec{w}^{T+1}_G||^2 \leq 4R^2T$ can be shown similar to the work in [Shivaswamy and Joachims, 2012]. Let $\vec{u}^*_G = \sum_i \vec{u}^*_i$. We can prove a lower bound on $\vec{w}^{T+1}_G \cdot \vec{u}^*_G$:

$$\begin{aligned}
\vec{w}^{T+1}_G \cdot \vec{u}^*_G &= \vec{w}^T_G \cdot \vec{u}^*_G + \vec{\Delta}_t \cdot \vec{u}^*_G \\
&= \vec{w}^T_G \cdot \vec{u}^*_G + \vec{\Delta}_t \cdot \vec{u}^*_{a(t)} + \sum_{j \neq a(t)} \vec{\Delta}_t \cdot \vec{u}^*_j \\
&= \vec{w}^T_G \cdot \vec{u}^*_G + \vec{\Delta}_t \cdot \vec{u}^*_{a(t)} + \sum_{j \neq a(t)} \vec{\Delta}_t \cdot (\vec{u}^*_{a(t)} - \vec{\xi}_{a(t),j}) \\
&= \vec{w}^T_G \cdot \vec{u}^*_G + M\vec{\Delta}_t \cdot \vec{u}^*_{a(t)} - \sum_{j \neq a(t)} \vec{\Delta}_t \cdot \vec{\xi}_{a(t),j} \\
&\geq \vec{w}^T_G \cdot \vec{u}^*_G + M\kappa C_T - \sum_{j \neq a(t)} ||\vec{\Delta}^T||.||\vec{\xi}_{a(t),j}|| \\
&\geq \vec{w}^T_G \cdot \vec{u}^*_G + M\kappa C_T - (M-1)2R\sqrt{2\delta} \\
&\geq M\kappa \sum_{i=1}^{T} C_t - (M-1)2R\sqrt{2\delta}T
\end{aligned}$$

Composing with the upper bound and applying the Cauchy-Schwarz inequality gives us $M\kappa\sum_{t=1}^{T} C_t - T2R\sqrt{2\delta}(M-1) \leq 2R\sqrt{T}M$, which proves the claim. □

This bound shows that when $\delta > 0$ and $M > 1$, the average effort will not necessarily converge to 0. This makes sense, since differences between individual experts can never be learned.

5.2 `individual`

For the single expert setting, the results in [Goetschalckx *et al.*, 2014] give an average cost bound of $\frac{1}{T}\sum_{t=1}^{T} C_t \leq \frac{2R}{\kappa\sqrt{T}}$ for a Perceptron-based algorithm. Here we consider how this average cost depends on the number of experts M for $M > 1$, when we treat each expert as an independent learning problem (i.e. the **`individual`** algorithm). The following result shows that while **`individual`** will be guaranteed to converge to a perfect solution, ignoring existing similarities among experts can slow down learning.

First, it is useful to note that, if T_i gives the number of iterations where expert i was the active user, (so $T = \sum_i T_i$)

then we have that

$$\sum_{i=1}^{M} \sqrt{T_i} \leq \sqrt{MT} \tag{1}$$

and this bound is obtained with equality when all experts are selected equally often.

Theorem 2. *Using algorithm* **`individual`**, *the average effort spent during the first T iterations is bound by $\frac{1}{T}\sum_{t=1}^{T} C_t \leq \frac{2R}{\kappa\sqrt{T}}\sqrt{M}$.*

Proof. From a slight variation of the results in [Shivaswamy and Joachims, 2012], we observe that a Perceptron-based algorithm for a single user will have the bound $\frac{1}{T}\sum_{t=1}^{T} C_t \leq \frac{2R}{\kappa\sqrt{T}}$.

In the multi-user setting, each expert i will be responsible for T_i examples. This means that expert j will have a total cost bound by $\sum_{t,a(t)=j} C_t \leq \frac{2R\sqrt{T_j}}{\kappa}$. The total cost for all experts will be bound by

$$\sum_{t=1}^{T} C_t \leq \frac{2R}{\kappa} \sum_j \sqrt{T_j} \tag{2}$$

From inequality (1) we obtain the result: $\frac{1}{T}\sum_{t=1}^{T} C_t \leq \frac{2R}{\kappa\sqrt{T}}\sqrt{M}$. □

This result shows that there is a multiplicative penalty of $\sqrt{M}$ compared to the single expert case, which is due to the fact that each independent learner is learning from fewer examples. By considering the bounds for **`global`** and **`individual`** we can see that during the early stages of learning, when $T = O(\frac{M}{\delta})$, global will have a smaller worst case regret than **`individual`**, but for larger T **`individual`** will have a smaller worst case regret. This agrees with the intuition that generalization helps the most when there are larger numbers of users that are more similar, but eventually generalization will hurt performance in the limit.

5.3 General Case

We now show that the general MCL algorithm (with $\alpha > 0$) strikes a balance between **`global`** and **`individual`**. In particular, we would like an algorithm that can take advantage of generalization (unlike **`individual`**), but also converge to perfect solutions for each expert in the limit (unlike **`global`**).

For MCL with any $\beta \geq 0$ and $\alpha > 0$, we have the following bound:

Theorem 3. *Using the MCL algorithm with $\alpha > 0$ and $\beta \geq 0$, the average effort of the first T iterations is bound by: $\frac{1}{T}\sum_{t=1}^{T} C_t \leq \frac{2R}{\kappa}\frac{\sqrt{\alpha^2+\beta^2}}{\alpha}\frac{\sqrt{M}}{\sqrt{T}}$*

Proof. We work with $(M+1)D$-dimensional vectors, where all the user vectors are combined into the vector $\vec{\mathbf{u}} = [\vec{0}; \vec{u}_1^*; \vec{u}_2^*; \ldots; \vec{u}_M^*]$, and all the learned weights are combined into the vector $\vec{\mathbf{w}}^T = [\vec{w}_G^T; \vec{w}_1^T; \ldots; \vec{w}_M^T]$. If we define $\vec{\mathbf{\Delta}}^T = [\beta\vec{\Delta}^T; \vec{0}; \ldots; \alpha\vec{\Delta}^T; \ldots; \vec{0}]$ where the occurence of $\alpha\vec{\Delta}^T$ occurs at the $a(T)$'th position ($a(T)$ is the index of the expert responsible for the example at time T). Note that $\langle\vec{\mathbf{U}}, \vec{\mathbf{\Delta}}^T\rangle = \alpha\langle\vec{u}_{a(T)}^*, \vec{\Delta}^T\rangle \geq \alpha\kappa C_T$ and $\langle\vec{\mathbf{w}}^T, \vec{\mathbf{\Delta}}^T\rangle$ is the prediction that the MCL algorithm would make at time T, so we know that $\langle\vec{\mathbf{w}}^T, \vec{\mathbf{\Delta}}^T\rangle \leq 0$ if there was an update performed at time T. Using this notation, we can use the standard analysis for a perceptron algorithm for coactive learning. We get that $||\vec{\mathbf{w}}^{T+1}||^2 \leq (\alpha^2 + \beta^2)4R^2T$ and $\alpha\kappa\sum_{t=1}^{T} C_t \leq \langle\vec{\mathbf{U}}, \vec{\mathbf{w}}^{T+1}\rangle \leq 2R\sqrt{\alpha^2+\beta^2}\sqrt{T}\sqrt{M}$, which proves the bound. □

This proof shows that MCL using any value $\beta \geq 0$ and any $\alpha > 0$ is guaranteed to converge to an optimal solution for all users, as is the case for **`individual`** (in fact, Theorem 2 is a special case of Theorem 3, with $\alpha = 1, \beta = 0$). However, it is expected that allowing the algorithm to generalize over the different users will result in a large boost during the early stages, as is the case for **`global`**. In the next section, we will empirically verify this behavior for the specific setting of $\alpha = \beta = 1$, the **`balanced`** algorithm, resulting in a bound of $\frac{1}{T}\sum_{t=1}^{T} C_t \leq \frac{2R}{\kappa}\frac{\sqrt{2M}}{\sqrt{T}}$.

6 Experiments

The algorithms are evaluated on two synthetic and one real-world domain. All reported results are averages over 10 runs.

The synthetic domains have feature vectors of dimension 10. A base vector was generated with all coefficients generated from a uniform $[0, 1]$ distribution. All experiments used 50 experts. For each expert a user vector is generated by perturbing the base vector by a vector drawn from a normal distribution with mean $\vec{0}$ and diagonal covariance matrix $\sigma\mathbf{I_{10}}$ and then normalizing. Experiments were performed with $\sigma = 0.01, 0.05$ and 0.25, resulting in values of δ of about $0.001, 0.02$ and 0.38. At each iteration, an expert is selected according to a uniform multinomial.

6.1 Domains

Ranking.

Here, for each problem, the learner is presented with 30 vectors $\vec{v}_i$ and needs to sort them in order of estimated utility, where utility is given by $\vec{v}_i \cdot \vec{w}_i$. The expert tries to improve on the ranking according to their utility function by iteratively switching the positions of two subsequent items in the list if the true utility of the former is more than κ higher than the utility of the latter vector. In the experiment, the value $\kappa = 0.1$ was used. Each such switch added 1 to the total effort spent. The feature vector for a candidate solution is constructed as $\sum_{i=1}^{29}\sum_{j=i+1}^{30}(\vec{v}_j - \vec{v}_i)$.

Path Planning.

Here, for each problem, the environment consists of a 7-dimensional hypercube, where each edge is described by a 10-dimensional feature vector, and the solver needs to find the optimal path of length 7 from one corner to the diagonally opposite corner. The cost of a path consisting of edges with feature vectors $\vec{v}_1 \ldots \vec{v}_7$ is given by $\vec{w}_i \cdot \sum_j \vec{v}_j$. There are 7! such possible paths.

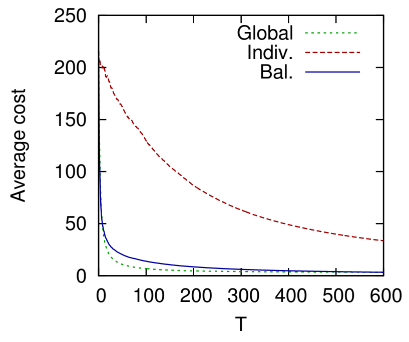

Figure 1: Ranking, $\sigma = 0.01$

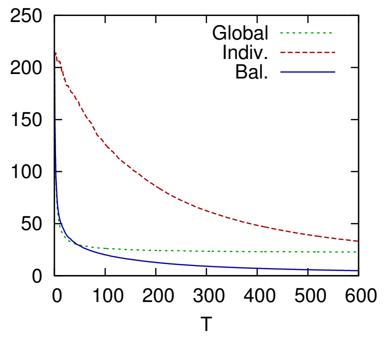

Figure 2: Ranking, $\sigma = 0.05$

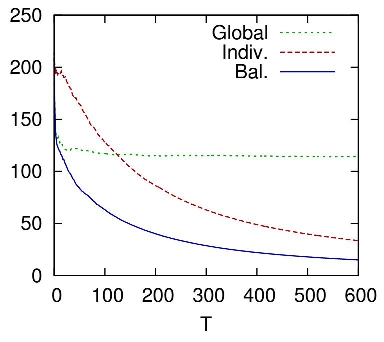

Figure 3: Ranking, $\sigma = 0.25$

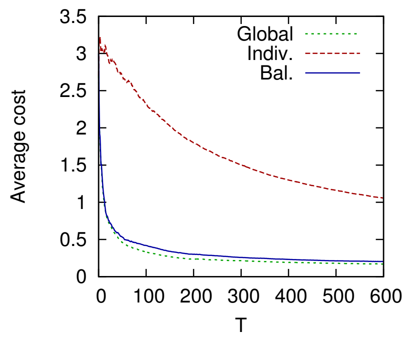

Figure 4: Path Planning, $\sigma = 0.01$

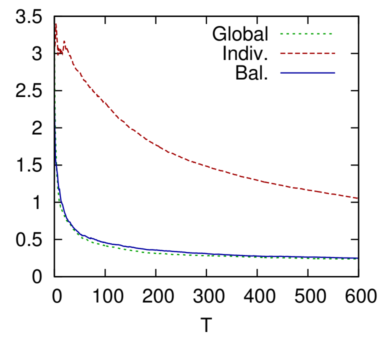

Figure 5: Path Planning, $\sigma = 0.05$

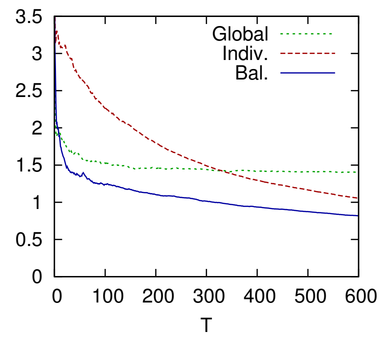

Figure 6: Path Planning, $\sigma = 0.25$

The solver uses a simple 2 step lookahead search. The expert can improve this trajectory by looking at three subsequent moves and reordering them, if any such reordering gives an improvement of at least $\kappa = 0.1$, until no such improvement is possible.

Spam Detection.

The third domain is a real-world domain, namely the spam detection dataset as presented in the 2006 ECML/PKDD Discovery Challenge [Bickel, 2008], task b. This task has 15 users, and for each user a set of 400 e-mail messages, represented by bag-of-word (150000-dimensional) feature vectors. Each e-mail has a classification of either being a spam e-mail or a clean e-mail.

In this paper we treat this dataset as a ranking task. At each step a user is randomly selected, and 10 e-mails are randomly selected from their set. The learner needs to present them to the user, ordered by whether they are thought to be spam or not. The user will then move non-spam e-mails to the top of the list until the list is properly sorted, the number of such moves is the measured cost. The feature vectors are constructed similar to the feature vectors of the ranking domain.

6.2 Results

The goal of the experiments is to observe the behavior of **`global`**, **`individual`** and **`balanced`** when applied to problems of varying expert similarity. We applied the algorithms to the artificial domains, with values of σ set to 0.01, 0.05 or 0.25.

Results are shown in Figures 1-6. Some things are very noticeable. Both **`global`** and **`balanced`** have a dramatic drop in average cost after just a few examples. When the experts are highly similar ($\sigma = 0.01$ and $\sigma = 0.05$) **`global`** outperforms **`individual`** by large margins, though the advantage diminishes for large T as expected.

The opposite is true for the case where experts are quite different ($\sigma = 0.25$), where generalizing from other experts hurts performance. The average cost of **`global`** drops fast for the first few examples, but then remains at the same level.

In both of these cases, **`balanced`** behaves similarly to the better of the two algorithms, showing that balanced is a robust way to allow for generalization while not sacrificing convergence when generalization is not possible. When facing a problem where it is not known beforehand what magnitude δ might have, **`balanced`** provides a safe option, guaranteeing that individual preferences will be learned yet also exploiting similarities between users.

Figure 7 shows results on the ECML/PKDD 2006 Discovery Challenge Spam detection dataset. We see that for this data set **`global`** outperforms **`individual`**, especially early in learning, indicating that there is a significant benefit from generalization across users in this domain. We also see here that **`balanced`** behaves similarly to **`global`**, but does outperform it by a small margin, especially later in the learning curve. This again shows that **`balanced`** is a robust approach for taking advantage of generalization opportunities.

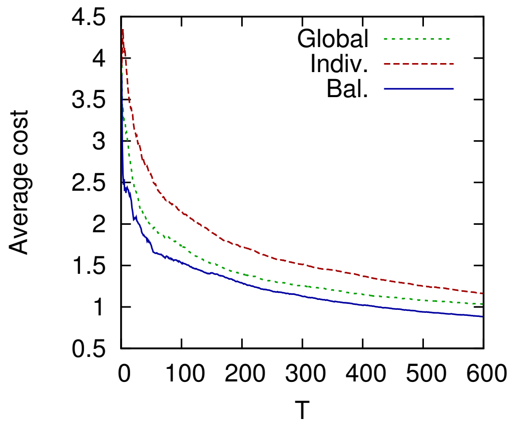

Figure 7: Spam Detection

7 Conclusions

In this paper, we presented and evaluated algorithms for multitask co-active learning. One of the main contributions is the MCL algorithm, which is the first coactive learning algorithm that can both take advantage of generalization opportunities across experts, and guarantee convergence to zero average cost. A theoretical bound was derived to show that it will eventually converge to predictions which are tailored to the specific experts or tasks.

Our empirical results show that MCL is able to strike a balance between generality and specificity. Both **`balanced`** and **`global`** (which ignores expert identifiers and treats them all as 1 expert) have a large drop in average cost over the first few examples, where **`individual`** (an algorithm which does not generalize over users) cannot share experience between users, and needs to have a larger amount of data per user to improve. On the other hand, **`global`** stops improving after just a handful of examples. It does not converge to an optimal solution, since it cannot learn any specific differences between different experts or tasks. The **`individual`** and **`balanced`** algorithms do not have this problem and eventually converge to optimal weight vectors for each expert.

When encountering a novel problem where it is not known to what extent different tasks or experts can be expected to share utility functions, choosing either the **`individual`** or **`global`** algorithms might be dangerous. Since **`balanced`** combines the strengths of both, and has performance close to the best of either in any given situation, it presents a safe choice.

Acknowledgements

The authors acknowledge support of the ONR ATL program N00014-11-1-0106.

References

[Bickel, 2008] Steffen Bickel. ECML-PKDD discovery challenge 2006 overview. In *ECML-PKDD Discovery Challenge Workshop*, pages 1–9, 2008.

[Bou-Ammar *et al.*, 2014] Haitham Bou-Ammar, Eric Eaton, Paul Ruvolo, and Matthew E. Taylor. Online Multi-Task Learning for Policy Gradient Methods. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014*, pages 1206–1214, 2014.

[Cavallanti *et al.*, 2010] Giovanni Cavallanti, Nicolo Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. *The Journal of Machine Learning Research*, 9999:2901–2934, 2010.

[Evgeniou and Pontil, 2004] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[Evgeniou *et al.*, 2005] Theodoros Evgeniou, Charles A Micchelli, Massimiliano Pontil, and John Shawe-Taylor. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6(4), 2005.

[Goetschalckx *et al.*, 2014] Robby Goetschalckx, Alan Fern, and Prasad Tadepalli. Coactive Learning for Locally Optimal Problem Solving. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1824–1830, 2014.

[Maurer *et al.*, 2013] Andreas Maurer, Massi Pontil, and Bernardino Romera-Paredes. Sparse coding for multitask and transfer learning. In *Proceedings of the 30th International Conference on Machine Learning*, pages 343–351, 2013.

[Raman and Joachims, 2013] Karthik Raman and Thorsten Joachims. Learning Socially Optimal Information Systems from Egoistic Users. In *ECML/PKDD (2)*, pages 128–144, 2013.

[Raman *et al.*, 2012] Karthik Raman, Pannaga Shivaswamy, and Thorsten Joachims. Online learning to diversify from implicit feedback. In *KDD*, pages 705–713, 2012.

[Raman *et al.*, 2013a] Karthik Raman, Thorsten Joachims, Pannaga Shivaswamy, and Tobias Schnabel. Stable Coactive Learning via Perturbation. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 837–845. JMLR Workshop and Conference Proceedings, May 2013.

[Raman *et al.*, 2013b] Karthik Raman, Thorsten Joachims, Pannaga Shivaswamy, and Tobias Schnabel. Stable coactive learning via perturbation. In *Proceedings of The 30th International Conference on Machine Learning*, pages 837–845, 2013.

[Ruvolo and Eaton, 2013] Paul Ruvolo and Eric Eaton. ELLA: An Efficient Lifelong Learning Algorithm. In

Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013, pages 507–515, 2013.

[Ruvolo and Eaton, 2014] Paul Ruvolo and Eric Eaton. Online Multi-Task Learning via Sparse Dictionary Optimization. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 2062–2068, 2014.

[Shivaswamy and Joachims, 2012] Pannaga Shivaswamy and Thorsten Joachims. Online Structured Prediction via Coactive Learning. *CoRR*, abs/1205.4213, 2012.

[Sreenivasan *et al.*, 2014] Vishnu Purushothaman Sreenivasan, Haitham Bou-Ammar, and Eric Eaton. Online Multi-Task Gradient Temporal-Difference Learning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 3136–3137, 2014.