Humble Active Learning from Peers for Personalization

Che Shen
cs3627
cs3627@columbia.edu

Yuemei Zhang
yz3193
yz3193@columbia.edu

Jingxi Xu
jx2324
jx2324@columbia.edu

Abstract

In this paper, we develop efficient active and multitask learning algorithms for large-scale personalization system with limited number of queries. Our contributions are as follows. 1) We reimplement the active learning from peers algorithm proposed by [Murugesan and Carbonell, 2017], and our implementation achieves similar results on every dataset used by that paper. 2) We propose a new algorithm - humble active learning from peers that learns more efficiently from the labels and peer predictions. 3) We evaluate the performance of two algorithms on spam detection task and music recommendation task. Our algorithm significantly outperforms the baseline algorithm on both tasks.

Our source code is available at GitHub repo: https://github.com/JingxiXu/humble-peer-amtl.

1. Introduction

Personalization has been a topic of interest in the machine learning community for years. Not only because the problem settings of personalization problems can be a direct application of many machine learning methods, but also as a result of the increasing demand of highly personalized online systems. One of the biggest challenge in developing personalization related applications such as personalized spam filters, personalized music recommenders is the sometimes prohibitively large amount of data that are required for training personalized applications.

Take the personalized spam filters as a simple example. If each spam filter is just a simple binary classifier which outputs zero or one by reading the email features, it is a fairly simple problem to learn a good binary classifier with many handy machine learning methods including support vector machine, logistic regression, etc. Several thousands of labeled emails are usually sufficient for this task. However, in real world email systems, the number of registered users is potentially to be in millions, and to learn a highly personalized spam filter for each of them by getting several thousands of labeled data for each user is not feasible.

Intuitively, however, we know that we don't really need that much labelled data to train millions of personalized classifiers if we can have some information about the relationship or similarity among users. Some spams are universal to everyone like financial spams and some spams are regarded as spams only for affinity groups. The emails about discount on tickets to basketball games are not spams for people who like basketball, but are spams to people who don't. Therefore, when the classifier of user A is not confident whether the predicted label is correct or not, instead of asking the oracle for the true label, if we know that the classifier of user B is quite confident about this sample and user A is similar to user B, the learner of user A can just use the label returned by user B as the ground truth without asking the oracle.

Thus, in this research project, we aim to develop efficient machine learning algorithms for large-scale personalization systems with labelled data as few as possible.

2. Related work

One popular method in the machine learning community which is widely used to efficiently reduce the number of labels needed to train a model is active learning. Active learning is a special case of semi-supervised machine learning in which a learning algorithm is able to interactively query the user/oracle to obtain the desired outputs at new data points. [Settles, 2012] writes a comprehensive survey on existing active learning methods.

Generally speaking, there are three scenarios for active learning: 1) membership query synthesis where the learner synthesizes the query to be asked so that the question might not make sense at all; 2) stream-based selective sampling where the unlabelled training data is coming sequentially and after each sample is received, the learner decides whether to ask the true label or simply abandon this sample; 3) pool-based sampling where at each round the learner picks a sample from the pool of unlabelled data based on some metrics, query the true label and then add it to the labelled pool. In our research project, we are particularly interested in the stream-based selective sampling scenarios because in online personalization and recommendation systems, new data always appears sequentially (e.g., a certain user just provides a new rating on a song or just labels an email as spam) and the system has to deal with the new sample (either abandons it or learns from it) immediately. Most approaches in active learning for sequential data adopts a measure of uncertainty or confidence to decide whether to query the label or not [Cesa-Bianchi et al., 2006, Dekel et al., 2012, Orabona and Cesa-Bianchi, 2011, Cavallanti et al., 2009, Agarwal, 2013].

Another approach that is helpful to achieve low number of queries is multitask learning, which leverages the relationship between the tasks to transfer relevant knowledge from information-rich tasks to information-poor ones. In a broad sense, there are two settings for multitask learning: 1) batch learning, where the entire learning set is available to the learner; 2) online learning, where the learner sees the sample sequentially. The online multitask setting has received increasing attention in the machine learning community in recent years [Cavallanti et al., 2010, Abernethy et al., 2007, Dekel et al., 2007, Lugosi et al., 2009, Saha et al., 2011, Murugesan et al., 2016, Dekel et al., 2006]. In online multitask setting, the learner receives a sample along with a task identifier at each round, it predicts the output and update itself based the sample's true label. It is worth noting that online multitask learning differs from active learning in that the learner does not have the right to choose whether to query the true label or not and it always predicts and then updates weights after seeing each data.

The most related work which we reimplement and analyze in this paper combines active learning with online multitask learning [Murugesan and Carbonell, 2017]. It enables the multitask learner to make the decision of whether to ask for the true label or not. This method is called online multitask learning with selective sampling, or simply active learning from peers.

3. Problem Setup

The problem formulation and setup are mostly the same as shown in [Murugesan and Carbonell, 2017]. Suppose we are given K tasks and the k-th task is associated with N_k training samples. We consider each task to be a binary classification problem for brevity. The data for task k is $\{x_k^{(i)}, y_k^{(i)}\}_{i=1}^{N_k}$, where $x_k^{(i)} \in \mathbb{R}$ is the i-th instance from the k-th task and $y_k^{(i)}$ is the corresponding label. When the notation is clear from the context, we drop task index k and simply write $((x^{(i)}, k), y^{(i)})$.

Let $\{w_k^{(t)}\}_{k\in[K]}$ be the set of weights learned for the K binary classifiers at round t. Then the loss of task k on the sample $((x^{(t)}, k), y^{(t)})$ at round t is $\ell_{kk}^{(t)} = \left(1 - y^{(t)}\langle x^{(t)}, w_k^{(t)}\rangle\right)_+$; and the loss of task $m (m \neq k)$ is $\ell_{km}^{(t)} = \left(1 - y^{(t)}\langle x^{(t)}, w_m^{(t)}\rangle\right)_+$.

4. Algorithm

In this section we introduce our new algorithm, which is called "humble active learning from peers". We first briefly review the "active learning from peers" algorithm from [Murugesan and Carbonell, 2017]. Then we analyze this algorithm to figure out the potential shortcomings. Finally, we introduce our new algorithm which overcomes these shortcomings and achieves better performance on several data sets.

4.1. Active Learning from Peers

The algorithm "active learning from peers" is introduced in [Murugesan and Carbonell, 2017] as a multitask active learning algorithm. The algorithm is shown in Algorithm 1.

Algorithm 1 Active Learning from Peers

```
Input $b_2 \geq b_1 > 0, \lambda > 0$, number of rounds $T$
 1: initialize $w_m^{(0)} = 0, \forall m \in [K], \boldsymbol{\tau}^{(0)}$
 2: for $t = 1, 2, .., T$ do
 3:     Receive $(x^{(t)}, k)$
 4:     Compute $p_{kk}^{(t)} = \langle x^{(t)}, w_k^{(t-1)}\rangle$
 5:     Predict $\hat{y}^{(t)} = sign(\hat{p}_{kk}^{(}t))$
 6:     Draw a Bernoulli random variable $P^{(t)}$ with probability $\frac{b_1}{b_1 + |\hat{p}_{kk}^{(t)}|}$
 7:     if $P^{(t)} = 1$ then
 8:         Compute $p_{km}^{(t)} = \langle x^{(t)}, w_m^{(t-1)}\rangle \forall m \neq k, m \in [K]$
 9:         Compute $\tilde{p}^{(t)} = \sum_{m\neq k, m\in[K]} \tau_{km}^{(t-1)} \hat{p}_{km}^{(t)}$
10:         Set $\tilde{y}^{(t)} = sign(\tilde{p}^{(t)})$
11:         Draw a Bernoulli random variable $Q^{(t)}$ with probability $\frac{b_2}{b_2 + |\tilde{p}^{(t)}|}$
12:     end if
13:     Set $Z^{(t)} = P^{(t)}Q^{(t)}, \tilde{Z}^{(t)} = P^{(t)}(1 - Q^{(t)})$
14:     Query true label $y^{(t)}$ if $Z^{(t)} = 1$
15:     set $M^{(t)} = 1$ if $\hat{y}^{(t)} \neq y^{(t)}$
16:     Update $w_k^{(t)} = w_k^{(t-1)} + (M^{(t)}Z^{(t)}y^{(t)} + \tilde{Z}^{(t)}\tilde{y}^{(t)})x^{(t)}$
17:     Update $\tau$:
            $$\tau_{km}^{(t)} = \frac{\tau_{km}^{(t-1)} e^{-\frac{Z^{(t)}}{\lambda} l_{km}^{(t)}}}{\sum_{m'\in[K], m'\neq k} \tau_{km'}^{(t-1)} e^{-\frac{Z^{(t)}}{\lambda} l_{km'}^{(t)}}}, m \in [K], m \neq k$$
18: end for
```

This algorithm provides an efficient way for multitask learning – each task uses not only its own data, but also data from other tasks. More specifically, in each round, $p_{kk}^{(t)}$ represents the confidence of the classifier of task k, just as the common confidence measure for SVM or perceptron. The algorithm makes use of this confidence value by sampling a Bernoulli random variable, $P^{(t)}$, to decide whether to query the true label of this data point. The larger $p_{kk}^{(t)}$ is, the more likely for $P^{(t)}$ to be 0. Apart from looking at its own confidence, it also looks at the confidence value from its peer tasks. The peer confidence value is computed through a weighted sum of each task's confidence value. The algorithm makes use of this value by sampling another Bernoulli random variable $Q^{(t)}$. It queries the true label if and only if $P^{(t)}, Q^{(t)}$ are both 1. Information from peers and from the task itself is leveraged to make a decision in this process.

At the end of each round the algorithm updates its parameters $w_k^{(t)}$ and $\boldsymbol{\tau}$. The update of $w_k^{(t)}$ follows the common scheme used in perceptron – add the new feature vector to the weight vector if it is misclassified. One thing needs noticing here is that, if the task itself is not confident enough but its peers are confident, it will use its peers' prediction as the true label to update its weight vector. (This case is indicated by $\tilde{Z}^{(t)}$ in the algorithm.) Rules for updating $\boldsymbol{\tau}$ can be viewed as solving an optimization problem. The details of the derivation can be found in [Murugesan et al., 2016].

The above algorithm efficiently uses information from peer tasks in the training period. But it has the defect that during the test period it cannot use the information from peer tasks any more. In [Murugesan and Carbonell, 2017] it simply uses each task's own weight vector in testing period to predict the label. So all the information from peer tasks are ignored. Indeed it makes more sense to use a weighted sum of the prediction given by each task to make the final prediction, but this simple change does not improve the performance of the algorithm.

4.2. Humble Active Learning from Peers

To make better use of peer tasks in testing period, we proposed a new algorithm, called "Humble Active Learning from Peers", which is shown in Algorithm 2

In our algorithm, we do not treat the task itself and peer tasks separately. Instead, we use a weighted sum of their output as the confidence value. This is why it is called "humble" – it always asks its peer tasks when making a decision. At the same time, similar to "active learning from peers" algorithm, we maintain a $k \times k$ relationship matrix $\boldsymbol{\tau}$. The value of entry τ_{ij} represents the "closeness" of task j to i, or, in other words, how much task i can trust task j in making the decision. We keep updating this matrix so that it can correctly reflect the relationship between tasks.

In each round, when a new data $(x^{(t)}, k)$ comes, we first compute the output of each task $\langle x^{(t)}, w_m^{(t-1)} \rangle$ for $m \in [K]$. (i.e. the inner product of the new feature vector with the weight vector of each task.) Then, we compute the weighted sum $p = \sum_{m \in [K]} p_{km}^{(t)} \tau_{km}^{(t-1)}$ of these output. We will discuss

Algorithm 2 Humble Active Learning from Peers

```
Input  $b > 0, C > 0$, number of rounds $T$
1:  initialize $w_m^{(0)} = 0, \forall m \in [K], \boldsymbol{\tau}^{(0)}$
2:  for $t = 1, 2, .., T$ do
3:      Receive $(x^{(t)}, k)$
4:      Compute $p_{km}^{(t)} = \langle x^{(t)}, w_m^{(t-1)} \rangle$ for $m \in [K]$
5:      $p = \sum_{m \in [K]} p_{km}^{(t)} \tau_{km}^{(t-1)}$
6:      Predict $y(\hat{t}) = sign(p)$
7:      Draw a Bernoulli random variable $P^{(t)}$ with probability $\frac{b}{b+|c|}$
8:      if $P^{(t)} = 1$ then
9:          Query true label $y^{(t)}$
10:         if $y^{(t)} \neq \hat{y}^{(t)}$ then
11:             Update $w_k^{(t)} = w_k^{(t-1)} + y^{(t)} x^{(t)}$
12:         end if
13:         Update $\tau$:
                $$\tau_{km}^{(t)} = \frac{\tau_{km}^{(t-1)} e^{-C \cdot l_{km}^{(t)}}}{\sum_{m' \in [K]} \tau_{km'}^{(t-1)} e^{-C \cdot l_{km'}^{(t)}}}, m \in [K]$$
14:     end if
15: end for
16: Output $\boldsymbol{\tau}^{(t)} \cdot w^{(t)}$ as the final weight
```

about it later. We make prediction based on $sign(p)$, and $|p|$ can be viewed as the confidence value of the prediction. We decide whether to query the true label by sampling a Bernoulli random variable with expectation $\frac{b}{b+|p|}$. As $|p|$ goes higher, we are less likely to query the true label.

If we indeed query the true label, we update the weight vector and the relationship matrix $\boldsymbol{\tau}$. Our updating rules are similar to those in "active learning from peers", but they have two key differences. First, in the "peers" algorithm, if peer tasks output a high confidence value while the task itself does not, the algorithm will update the weight vector according to the prediction made by peer tasks. In our algorithm, we only update weights when we have queried the true label. Secondly, in updating $\boldsymbol{\tau}$, "peers" algorithm keeps $\sum_{m' \in [K], m' \neq k} \tau_{km'}^{(t)} = 1$ for all $k \in [K]$ (i.e. the weight sum of all tasks *except* the current task is equal to 1), while $\tau_{kk}, k \in [K]$ is always set to 1. In our algorithm, we keep $\sum_{m' \in [K]} \tau_{km'}^{(t)} = 1$ (i.e. the weight sum of all tasks is equal to 1). The intuition behind this is that, we may have several similar tasks, and when predicting one of them, all these similar tasks should have similar weights. On the other hand, if there is no task similar to the current task, we should make predictions solely depending on the current task's prediction.

At the end of the algorithm, we output $\boldsymbol{\tau}^{(t)} \cdot w^{(t)}$, a weighted sum of weight vectors by $\boldsymbol{\tau}$, instead of the raw weight vectors $w^{(t)}$. This enables each task to make use of peer tasks in the testing period in a more natural way.

5. Experiments

In this section, we evaluate Algorithm 1 (PEER) proposed by [Murugesan and Carbonell, 2017] and Algorithm 2 (HUMBLE) in the online setting on real-world datasets.

5.1. Experimental Setup

Metrics: For the multitask active learning algorithms, there are three metrics for evaluation: the number of queries, the training mistake rate, and the accuracy on the test set. In the real-world scenario, true labels are usually expensive to obtain, so we expect the algorithms to use only a limited number of queries. In the training phase, the algorithm keeps asking for true labels and updating the model, and we compute the average mistake rate. In the test phase, we evaluate the classification accuracy of the final model.

Parameters: There are two parameters to tune for algorithm PEER, b_1 and b_2. We follow the setting in [Murugesan and Carbonell, 2017], set the value of $b_1 = 1$ and tune b_2 from 20 different values. Similarly, for algorithm HUMBLE, we set $b = 1$ and tune parameter C.

Dataset: We test our algorithm on dataset obtained from ECML PAKDD 2006 Discovery challenge[1] for the spam detection task, and Kaggle Million Song Dataset Challenge[2] for music recommendation task. Each user is considered as a single binary classification task. The email dataset consists of labeled training data of fifteen users. We build a personalized spam classifier for each user. The length of the feature vector is approximately 150K, with each feature indicating the number of appearance of a word. We randomly sample 100 emails per user for training, and use the rest of the dataset as test set. The music dataset contains the listening history of the users. The task is to predict whether a user has listened a song more than once. We use song lyrics in bag-of-words format as the features from the musiXmatch dataset[3] [Bertin-Mahieux et al., 2011], with feature length 5000. We pick the top 100 users with most songs in their listening history. Each user has 20 to 40 history records, and there are 3074 samples in total. We randomly sample 10% of the data as test set, and use the rest for training. Compared with the email dataset, the number of users in the music dataset is relatively larger, while there are fewer training samples corresponding to each user.

In order to evaluate how efficiently the algorithms make use of the labels, we give the algorithms the same query budget. During the training phase, the algorithms cannot make more queries than the budget limit. Note that when the peers are confident about their classification result, one task can still learn from the peers and update its weights, so when the algorithms have used all the query budget, they continue updating the model using unlabeled samples and prediction of peers.

In the next experiment, we fix the query budget to be constant and change the size of unlabeled samples to evaluate how efficiently the algorithms can learn from unlabeled data.

Spam Email Detection			
Model	Accuracy	#Queries	Mistake rate
PEER	0.8497 (0.007)	1108.8 (32.10)	0.2255 (0.005)
HUMBLE	0.8867 (0.031)	1046.6 (14.74)	0.1735 (0.006)

Table 1. means and standard errors

Music Recommendation			
Model	Accuracy	#Queries	Mistake rate
PEER	0.6325 (0.019)	2209.6 (20.73)	0.3477 (0.006)
HUMBLE	0.6808 (0.014)	1306.2 (46.00)	0.3322 (0.007)

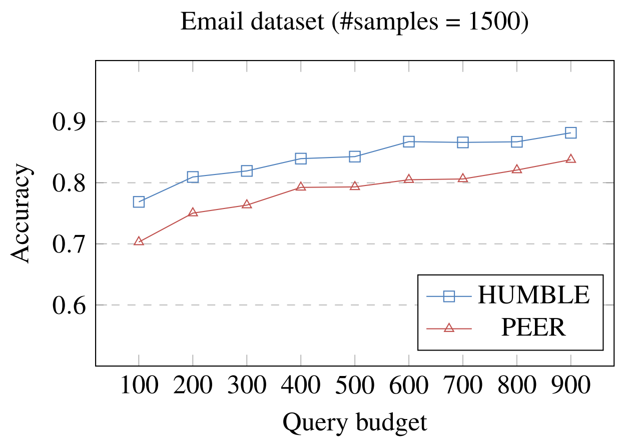
Table 2. means and standard errors

5.2. Results

We test the algorithms on the two datasets over 10 random runs. The first experiment is in the setting without any query budget, and the algorithms query the true label whenever they are not confident. Table 1 and Table 2 show the means and standard errors of the test accuracy, number of queries, and average training mistake. HUMBLE algorithm achieves higher accuracy and lower mistake rate, using fewer queries on both datasets.

Figure 1 shows the test accuracy with different query budgets. The humble algorithm achieves higher accuracy even with very limited query budget on the email dataset. On the music dataset, the accuracy of HUMBLE increases as we give it more query budget, while the accuracy of PEER remains the same when query budget is more than 400. Then we set the query budget to be equal to 300 and 800, respectively, and test the algorithms with different number of samples. Figure 2 shows the training mistake rate. On the email dataset, HUMBLE has better performance over PEER with approximately 10% lower mistake rate. Similarly, on the music dataset, the mistake rate

[1]http://ecmlpkdd2006.org/challenge.html
[2]https://www.kaggle.com/c/msdchallenge/data
[3]musiXmatch dataset, the official lyrics collection for the Million Song Dataset, available at: http://labrosa.ee.columbia.edu/millionsong/musixmatch



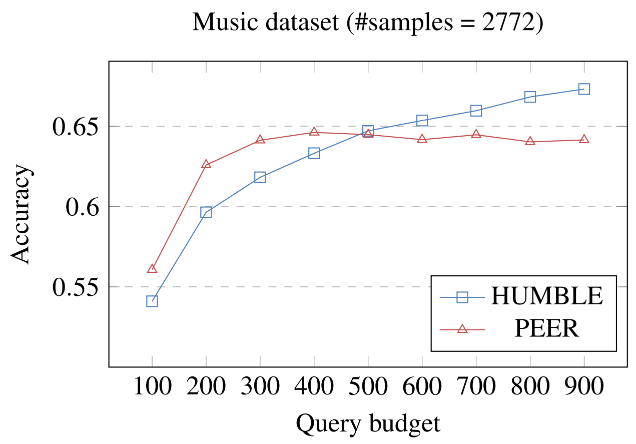
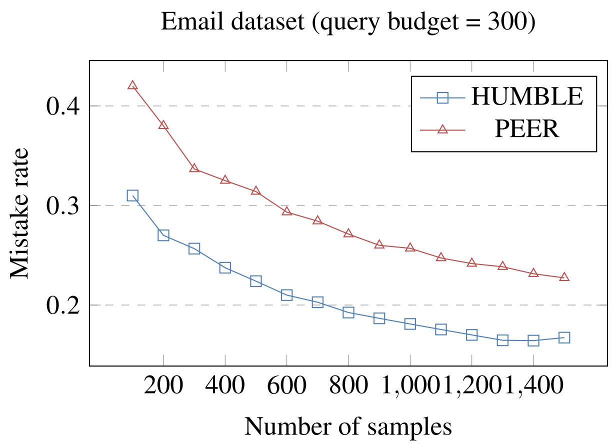


Figure 1. Accuracy



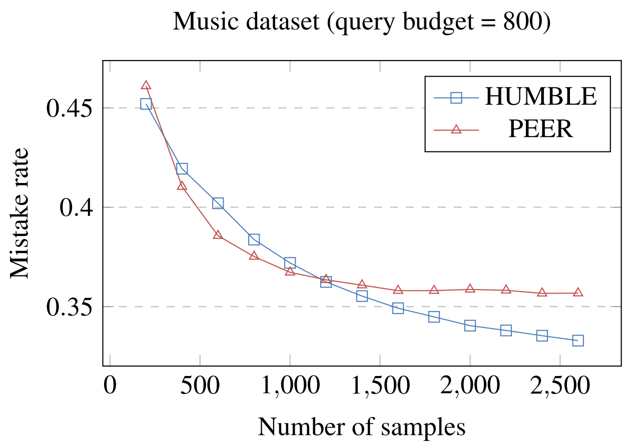


Figure 2. Mistake rate

of HUMBLE continues decreasing as we give it more samples. The results show that HUMBLE has stronger learning ability over PEER, and can more efficiently learn from the true labels and peer predictions.

Figure 3 shows a plot of the value of relationship matrix τ after running on the musiXmatch dataset. Entry τ_{ij} represents the weight of task j in making predictions for task i. It can be seen there are several similar lines and columns in the figure. This means that several users have similar taste in music and when making predictions for one user, we can make use of other user's information.

6. Discussion

Multitask active learning is a new area and many problems in this area still needs to be explored. Some of these problems are common scenarios in real world applications. For example:

Question 1: What will happen if the number of data is unbalanced among tasks? How to improve the algorithm to transfer knowledge from information-rich tasks to information-poor ones?

Question 2: If the number of tasks is very large but each task only have very few data, do we have better algorithms to leverage these data?

Question 3: How to use multitask learning to solve cold-

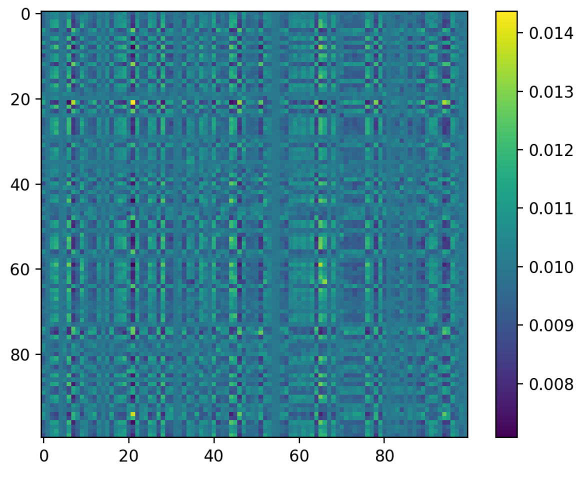


Figure 3. plot of relationship matrix τ

start problem? In terms of personalization, when a new user signs up, how to use the data of other old users to predict this user's preference?

7. Conclusion

We propose a new online multitask learning algorithm that not only learns to perform each task but also learns the inter-task relationship. The primary intuition we leverage in this paper is that some tasks might be relative to other tasks and they can probably use information from other tasks in making predictions. A key difference between our work and previous works in multitask active learning is that, when making predictions in one task, our algorithm considers all the tasks together, while the algorithms in previous work consider the current task first and then consider other tasks only when the current task cannot give a confident prediction. We empirically evaluate our algorithm on many datasets and our algorithm performs significantly better than other algorithms.

References

[Abernethy et al., 2007] Abernethy, J., Bartlett, P., and Rakhlin, A. (2007). Multitask learning with expert advice. In *International Conference on Computational Learning Theory*, pages 484–498. Springer.

[Agarwal, 2013] Agarwal, A. (2013). Selective sampling algorithms for cost-sensitive multiclass prediction. In *International Conference on Machine Learning*, pages 1220–1228.

[Bertin-Mahieux et al., 2011] Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. (2011). The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

[Cavallanti et al., 2009] Cavallanti, G., Cesa-Bianchi, N., and Gentile, C. (2009). Linear classification and selective sampling under low noise conditions. In *Advances in Neural Information Processing Systems*, pages 249–256.

[Cavallanti et al., 2010] Cavallanti, G., Cesa-Bianchi, N., and Gentile, C. (2010). Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 11(Oct):2901–2934.

[Cesa-Bianchi et al., 2006] Cesa-Bianchi, N., Gentile, C., and Zaniboni, L. (2006). Worst-case analysis of selective sampling for linear classification. *Journal of Machine Learning Research*, 7(Jul):1205–1230.

[Dekel et al., 2012] Dekel, O., Gentile, C., and Sridharan, K. (2012). Selective sampling and active learning from single and multiple teachers. *Journal of Machine Learning Research*, 13(Sep):2655–2697.

[Dekel et al., 2006] Dekel, O., Long, P. M., and Singer, Y. (2006). Online multitask learning. In *International Conference on Computational Learning Theory*, pages 453–467. Springer.

[Dekel et al., 2007] Dekel, O., Long, P. M., and Singer, Y. (2007). Online learning of multiple tasks with a shared loss. *Journal of Machine Learning Research*, 8(Oct):2233–2264.

[Lugosi et al., 2009] Lugosi, G., Papaspiliopoulos, O., and Stoltz, G. (2009). Online multi-task learning with hard constraints. *arXiv preprint arXiv:0902.3526*.

[Murugesan and Carbonell, 2017] Murugesan, K. and Carbonell, J. (2017). Active learning from peers. In *Advances in Neural Information Processing Systems*, pages 7011–7020.

[Murugesan et al., 2016] Murugesan, K., Liu, H., Carbonell, J., and Yang, Y. (2016). Adaptive smoothed online multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4296–4304.

[Orabona and Cesa-Bianchi, 2011] Orabona, F. and Cesa-Bianchi, N. (2011). Better algorithms for selective sampling. In *International conference on machine learning*, pages 433–440. Omnipress.

[Saha et al., 2011] Saha, A., Rai, P., DaumÃ, H., Venkatasubramanian, S., et al. (2011). Online learning of multiple tasks and their relationships. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 643–651.

[Settles, 2012] Settles, B. (2012). Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114.