Multi-task Learning with a Shared Annotator

Haim Cohen

Multi-task Learning with a Shared Annotator

Research Thesis

In Partial Fulfillment of The
Requirements for the Degree of
Master of Science in Electrical Engineering

Haim Cohen

Submitted to the Senate of
the Technion — Israel Institute of Technology
Kislev, 5777 Haifa December 2016

The Research Thesis Was Done Under The Supervision of Prof. Koby Crammer in the Faculty of Electrical Engineering.

I would like to thank Koby for his remarkable guidance and support.
Special thanks to Zohar, my beloved wife for her love, patience and support. Thanks to my kids, Shefer and Saar as well, for bringing cuteness and happiness during busy days.

Publications

Parts of this work were published in
"Learning Multiple Tasks in Parallel with a Shared Annotator".
Haim Cohen and Koby Crammer. NIPS 2014.

Contents

List of Figures

List of Tables

Abstract

We introduce a new multi-task framework, in which K online learners are sharing a single annotator with limited bandwidth. This limitation, does not allow the algorithm to query the true label from more than one task at a time. On each round, each one of the K learners receives an input example, and makes a prediction about the label of that input. Then, a shared (stochastic) mechanism decides which one of the K inputs will be annotated. This decision is based on the estimation of the nature of each task relative to the other and exploited this diversity to focus on harder tasks over the easier. The learner that receives the feedback (label) may update its prediction rule, and then we proceed to the next round.

We develop online algorithms set for multi-task binary classification that learns in this setting, and bound its performances in the worst-case setting. The algorithms applies an exploration-exploitation approach in order to allocate the limited feedback in a way that focuses on the feedback of the harder tasks and as a result, reduces the total number of prediction mistakes.

Additionally, we show that our algorithm can be used to solve two bandits problems: contextual bandits, and dueling bandits with context, both allowed to decouple exploration and exploitation. Empirical study with OCR data , vowel prediction (VJ project) and NLP - sentiment analysis and document domain classification data shows that our algorithms outperforms algorithms that use uniform allocation, and essentially produces better performance for the same labour of the annotator.

Abbreviations and Notations

$SHAMPO$	—	SHared Annotator for Multiple PrOblems
$ECOC$	—	Error COrrecting Code
OCR	—	Optical Character Recognition
NLP	—	Natural language processing
$MFCC$	—	Mel-Frequency Cepstral Coefficients
$(x)_+$	—	$\max\{x, 0\}$
$\mathbb{E}_{t-1}[x]$	—	$\mathbb{E}_t[x \vert Z_1, Z_2, \cdots, Z_{t-1}]$
$\mathbb{I}[X]$	—	The indicator of event X
$\Vert\mathbf{u}\Vert$	—	ℓ_2-norm of the vector $\mathbf{u}$

Chapter 1

Introduction

Machine learning is a field of computer science that concerned with data processing and the ability of the computer to learn from this data. One main objective of this field is the development of algorithms capable of inference based on observable data, such as text documents, pictures, audio, video etc...

In the *Supervised Learning* setting, the input of the learning algorithm are input-label pairs. The goal of the algorithm is to learn the underlying connection between the inputs and their labels, thus being able to *predict* a label for a previously unseen input. When the possible label values are from a discrete finite set, this learning problem is called *classification*. The basic classification problem is the *binary classification*, i.e. classifying each data instance into one of only two possible classes, while in the *multiclass classification* setting, there are three or more possible classes . In contradiction to the multiclass classification task, the binary case is simpler because, eliminating one class, gives you the correct one straightforward. However, this is more difficult in the *multiclass classification*, when even when we eliminate one possible class, we yet have some more classes the we need to decide which of these classes is the correct one.

1.1 Online Learning

One of the main features of the classification problem is the way that the data is collected. In some applications, the labeled data is collected first, so that we have an access to the entire training dataset at once. Then, we use this whole examples collection as an input to a *Batch learning* algorithm and learn a classification model

about this problem. However, in a lot of real life application this is not the case. In some problems like spam filtering, there is a flow of data that is transmitted in sequence, and it takes time to collect a large amount of data to learn from, so we don't want to wait too long before we can have a decent prediction about the continuing incoming examples. For those applications we use *Online Learning* based algorithms. In this setting, at any time we keep the learned model in memory, and update it when a new labeled example is coming in. Unlike the *Batch learning*, in the *Online Learning*, the learner perception about the classification task become stronger when time is passing by.

The *Online Learning* is performed in rounds, where in each round t, the algorithm gets an input instance $\mathbf{x}_t$ in some domain $\mathcal{X}$, and predicts a correspond measure, $\hat{p}_t$ based on the algorithm decision rule. This measure can be in the label domain, $\mathcal{Y}$, or can be mapped into $\hat{y}_t$ which is the predicted label in $\mathcal{Y}$. After predicting the label, the true label (y_t in the labels domain, $\mathcal{Y}$) is revealed and the learner suffers a non negative loss of $l\left(\hat{p}_t, y_t\right)$ that measures how much the prediction is compatible with the true label. The desired property of such function is to generate low values when the prediction is close to the actual label in some sense, and high values when the opposite is true. Then, the algorithm updates its decision rule, based on the past known data and the revealed label.

1.2 Selective Sampling

Usually, in an online binary learning task setting, we improve the prediction over time, and that means that the algorithm have less and lees prediction mistakes when it updates its model. Sometimes, annotating the data consume expensive resources, like time, money or manpower, and we would like to avoid using those resources when we can. In other words, we would like to avoid querying labels for the input examples when it is possible. For example, if we update the model only when there is a prediction mistake (as in Perceptron), we actually don't really use the information about the correct label when there is no need to update. In such cases, it will be helpful to assess every time how much we are sure about our prediction, and no update should be done, so no query should be issued, or if we not sure about the prediction, hence we should issue a query and update the model using the update rule and the correct label. This approach, that queries labels only for selected examples is called *Selective sampling.*

1.3 Multi-Task with Shared Annotator

In supervised learning setting, the main bottleneck is the need to annotate data. A common protocol is problem centric: first collect data or inputs automatically (with low cost), and then pass it on to a user or an expert to be annotated. Annotation can be outsourced to the crowed by a service like Mechanical Turk (like google's recaptcha project), or performed by experts as in the Linguistic data Consortium. Then, this data may be used to build models, either for a single task or many tasks. This approach is not making optimal use of the main resource - the annotator - as some tasks are harder than others, yet we need to give the annotator all the data to be annotated for each task a-priori.

Another aspect of this problem is the need to adapt systems to individual users, to this end, such systems may query the user for the label of some input. Yet, if a few systems will do so independently, the user will be flooded with queries, and will avoid interaction with those systems. For example, sometimes there is a need to annotate news items from few agencies. One person cannot handle all of them, and only some items can be annotated, which ones? Our setting is designed to handle exactly this problem, and specifically, how to make best usage of annotation time. This settings can also handle the case when we want to limit the updates number, for example if we have a lot of clients that generate data, but only one server with a limited computation power is allocated to process the received data and we want to limit the amount of updates for all tasks.

We propose a new framework of online multi-task learning with a shared annotator. Here, algorithms are learning few tasks simultaneously, yet they receive feedback using a central mechanism that trades off the amount of feedback (or labels) each task receives. We derive a specific set of algorithms based on the good-old Perceptron algorithm, called SHAMPO (SHared Annotator for Multiple PrOblems) for binary classification, and analyze it in the mistake bound model, showing that our algorithm may perform well compared with methods that observe all annotated data. we show how to reduce few contextual bandit problems into our framework, and provide specific bounds for such settings. We evaluate our algorithm with four different datasets for OCR, vowel prediction (VJ) and document classification, and show that it can improve performance either on average over all tasks, or even if their output is combined towards a single shared task, such as multi-class prediction. We conclude with discussion of related work, and few of the many routes to extend this work.

Chapter 2

The Shared Annotator Setting

In our setting, there are K binary classification tasks to be learned simultaneously. In opposed to the common multi-task classification settings, here, no dependency between the tasks is assumed during the analysis, but the tasks can be dependent as well. The model learning is performed in rounds as an online learning algorithm, as following: On each round t, there are K input-label pairs $(\mathbf{x}_{i,t}, y_{i,t})$, one for each classification task, where $i = 1, 2, \ldots, K$ is the task index and t is the step index. The inputs $\mathbf{x}_{i,t} \in \mathbb{R}^{d_i}$ are vectors, and the labels $y_{i,t} \in \{-1, +1\}$ are binary. In the general case, the input-spaces for each problem may be different, and inputs may have different number of elements. Yet, in order to simplify the notation and without loss of generality, from now on, in our analysis we assume that for all of the tasks, $\mathbf{x}_{i,t} \in \mathbb{R}^d$. i.e. all the tasks are in the same dimension and $d_i = d$ holds for all tasks. In practice, since the proposed algorithms use the margin that is affected by the vector norm, there is a need to scale all the vectors into a ball.

On round t, the learning algorithm receives K input vectors $\mathbf{x}_{i,t}$, for $i = 1, \ldots, K$ tasks and produce K binary-labels output $\hat{y}_{i,t}$, where $\hat{y}_{i,t} \in \{-1, +1\}$ is the label predicted for the input $\mathbf{x}_{i,t}$ corresponds to the task i. The algorithm then chooses a task $J_t \in \{1, \ldots, K\}$ and asks from an annotator its true-label. Unlike the usual online multi-task setting, and due to the limitation in our case, it does not observe any other label. Then, the algorithm updates its model, using the received feedback, and proceeds to the next round and input examples. For the ease of the calculations below, we denote by K indicators $Z_t = (Z_{1,t}, \ldots, Z_{K,t})$, the identity of the task which was queried by the algorithm on round t, and set $Z_{J_t,t} = 1$ and $Z_{i,t} = 0$ for $i \neq J_t$. Clearly from the definition, the condition, $\sum_i Z_{i,t} = 1, \forall i, t$, always holds. In order to use it in the analysis below, we define

the notation $\mathrm{E}_{t-1}[x]$ as well, to be the conditional expectation $\mathrm{E}[x|Z_1, \ldots Z_{t-1}]$ given all previous choices of the tasks to be queried.

Step	1	2	3	4	5
Task 1	Q	NQ	Q	NQ	NQ
Task 2	NQ	NQ	Q	NQ	NQ
Task 3	Q	NQ	Q	NQ	NQ
Task 4	Q	Q	NQ	NQ	Q

Table 2.1: In selective sampling we focus on when to issue a query for a single task (a row).

Step	1	2	3	4	5
Task 1	Q	NQ	NQ	NQ	NQ
Task 2	NQ	NQ	Q	Q	NQ
Task 3	NQ	NQ	NQ	NQ	Q
Task 4	NQ	Q	NQ	NQ	NQ

Table 2.2: In the SHAMPO setting we focus on when to issue a query for a single step (a column).

Schematic illustration of a single iteration of multi-task algorithms is shown in Fig. 2.1. The top panel shows the standard setting of online multi-task algorithms with a shared annotator, that labels all inputs, which are fed to the corresponding algorithms to update corresponding models. The bottom panel shows the SHAMPO algorithm, which couples labeling annotation and learning process, and synchronizes a single annotation per round. At most one task performs an update per round, the one with the annotated input.

As we have mentioned before, our setting is similar to the Selective Sampling case, in a sense that at every round we ask a single binary question. However, while the Selective Sampling algorithm asks "when" to issue a query, at each time step, our SAHMPO algorithm asks on "which one" of the tasks to issue a query. An example for the difference between the Selective Sampling multi-task problem and the SHAMPO setting is Table 2.1 and Table 2. Where the cells are assigned with 'Q' (Queried) for tasks and rounds that the algorithm issue a query, and with

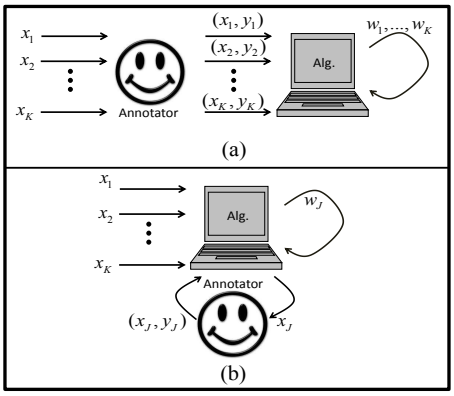


Figure 2.1: Illustration of a single iteration of multi-task algorithms: (a) standard setting and (b) SHAMPO setting

'NQ' (Not Queried) on tasks and rounds that the algorithm decided not to issue a query. As a result of the setting difference, when we run multi-task problem as an independent Selective sampling tasks, we allow at each round, to query more than one query (e.g. Table 2.1 step 2), or even less than one query, i.e. not asking for a feedback at all (e.g. Table 2.1 step 4). The SHAMPO setting on the other hand, is more limited, since for every round, the algorithm issue one, and only one query.

Chapter 3

SHMPO (SHared Annotator for Multiple Problems) Algorithms

I this chapter, we describe algorithms for multi-task learning with a shared annotator setting, that uses linear models. We start with a basic first order SHAMPO algorithm which is based on the perceptron algorithm and continue to number of algorithms that extends the basic one. Two steps are yet to be specified: how to pick a task to be labeled at each round and how to perform an update, once the true label for that task is given.

3.1 First Order SHAMPO

We focus here on linear-functions of the form $f(\mathbf{x}) = \mathrm{sign}(p)$ for the quantity $p = \mathbf{w}^\top \mathbf{x}$, $\mathbf{w} \in \mathbb{R}^d$, which often called the *margin*. Specifically, the algorithm maintains a set of K weight vectors, $\mathbf{w}_{i,t},\ i \in \{1, \ldots, K\}$. On round t, the algorithm predicts a label for each one of the tasks, $\hat{y}_{i,t} = \mathrm{sign}(\hat{p}_{i,t})$ where $\hat{p}_{i,t} = \mathbf{w}_{i,t-1}^\top \mathbf{x}_{i,t}$. On rounds for which the label of some task J_t is queried, the algorithm updates the model of the queried task only, such that for the rest of the tasks, $i \neq J_t$, we have $\mathbf{w}_{i,t} = \mathbf{w}_{i,t-1}$ and no update is made for those unqueried tasks.

We say that the algorithm has a prediction mistake on task i in round t if $y_{i,t} \neq \hat{y}_{i,t}$, and denote this event by the indicator $M_{i,t} = 1$, otherwise, if there is no mistake, we set $M_{i,t} = 0$. The goal of the algorithm is to minimize the cumulative mistakes $\sum_t \sum_i M_{i,t}$. Models are also evaluated using the *Hinge-*

loss. Specifically, let $\mathbf{u}_i \in \mathbb{R}^d$ be some vector associated with problem i. We denote the Hinge-loss of it, with respect to some input-label pair $(\mathbf{x}_{i,t}, y_{i,t})$, by $\ell_{\gamma,i,t}(\mathbf{u}_i) = \left(\gamma - y_{i,t}\mathbf{u}_i^\top \mathbf{x}_{i,t}\right)_+$, where, $(x)_+ = \max\{x, 0\}$, and $\gamma > 0$ is some parameter. The cumulative loss over all tasks and a sequence of n input steps, is $L_{\gamma,n} = L_\gamma(\{\mathbf{u}_i\}) = \sum_{t=1}^n \sum_{i=1}^K \ell_{\gamma,i,t}(\mathbf{u}_i)$. We also use the following expected hinge-loss over the random queried choices of the algorithm,

$$\bar{L}_{\gamma,n} = \bar{L}_{\{\mathbf{u}_i\}} = \mathrm{E}\left[\sum_{t=1}^n \sum_{i=1}^K M_{i,t} Z_{i,t} \ell_{\gamma,i,t}(\mathbf{u}_i)\right].$$

Now, we proceed by describing our algorithm and specify how to choose a task to query its label, and how to perform an update.

In order to select a task to query on, the algorithm uses the absolute value of the margin $\hat{p}_{i,t}$. We can see $|\hat{p}_{i,t}|$ as the certainty measure of the label prediction. intuitively if $|\hat{p}_{i,t}|$ is small, then there is uncertainty about the labeling of $\mathbf{x}_{i,t}$, and vice-versa for large values of $|\hat{p}_{i,t}|$. Similar argument was used by Tong and Koller [2000] for picking an example to be labeled in batch active learning. Example for the using of the margin as a certainty measure is shown in Fig. 3.1 which illustrated a state of a linear binary classification algorithm of two dimensional problem. The circles and crosses represent two different classes, and the separator hyperplane is shown. One of the circles examples have lower margin than the other, so its label can easily been replaced with the other one. Prima facie, under this claim, at each point we should query the true label for the task that corresponds to the smallest margin. Yet, if the model $\mathbf{w}_{i,t-1}$ is not accurate enough, due to small number of observed examples, this estimation may be rough, and may lead to a wrong conclusion. We thus perform an exploration-exploitation strategy, and query tasks randomly, with a bias towards tasks with low margin $|\hat{p}_{i,t}|$.

To the best of our knowledge, exploration-exploitation usage in this context of choosing an examples to be labeled (e.g. in settings such as semi-supervised learning or selective sampling) is novel and new. In order to get this property, we induced a distribution over the tasks in the time step t, such that the probability to

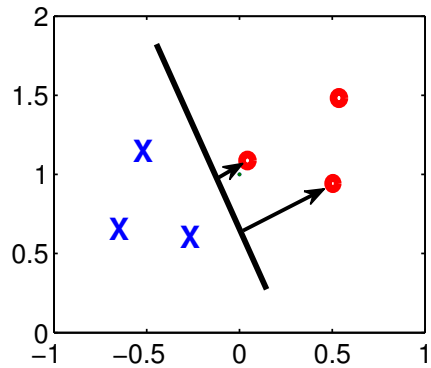


Figure 3.1: Example for the using of the margin as a certainty measure.

issue a query on the task j ($j = 1, \ldots, K$) is:

$$\Pr\left[J_t = j\right] = \frac{\left(b + |\hat{p}_{j,t}| - \min_{m=1}^{K} |\hat{p}_{m,t}|\right)^{-1}}{D_t}$$
$$\text{for } \; D_t = \sum_{i=1}^{K} \left(b + |\hat{p}_{i,t}| - \min_{m} |\hat{p}_{m,t}|\right)^{-1}. \tag{3.1}$$

where $m = 1, \ldots, K$ and $b \geq 0, b \in \mathbb{R}$ is a tradeoff parameter, between exploration and exploitation. Clearly, it is a distribution, since $\Pr\left[J_t = j\right] \geq 0$ and $\sum_j \Pr\left[J_t = j\right] = 1$. When we examine the extreme cases of b we see that for $b \to 0$ we have $\Pr\left[J_t = j\right] \to 1$ for the task with minimal margin, $J_t = \arg\min_{i=1}^{K} |\hat{p}_{i,t}|$, and $\Pr\left[J_t = j\right] \to 0$ for all the rest. In this case , pure exploitation is being made. The pure exploration happens when $b \to \infty$, then the distribution is uniform, $\Pr\left[J_t = j\right] = 1/K, \;\; \forall j$. Fig. 3.2 shows an example of this distribution for the case of two tasks ($K = 2$) at an arbitrary step t^*. For visualization purpose, we fix $|\hat{p}_{2,t^*}| = 0.5$ and see how the probability of task 1 to be chosen, is affected by varying b and $|\hat{p}_{1,t^*}|$ values. Three different vertical areas can be easily seen in the graph. The upper (green) area, where $b >> \max\left\{|\hat{p}_{1,t^*}|, |\hat{p}_{2,t^*}|\right\}$, shows uniform distribution ($\Pr\left[J_{t^*} = 2\right] = \Pr\left[J_{t^*} = 1\right] = 1/K = 0.5$) which represents the exploration over the tasks. In the lower area, the probability is compatible with the exploitation method and is changing between probability 1 in the left, and probability 0 to the right with a sharp threshold at $|\hat{p}_{1,t^*}| = 0.5$, which is very close to the delta distribution $\Pr\left[J_{t^*} = 1\right] = \mathbb{I}\left[\hat{p}_{1,t^*} < \hat{p}_{2,t^*}\right]$. Whereas, the

intermediate area, is the exploration-exploitation area that is given by a distribution that is biased toward the lowest margin task.

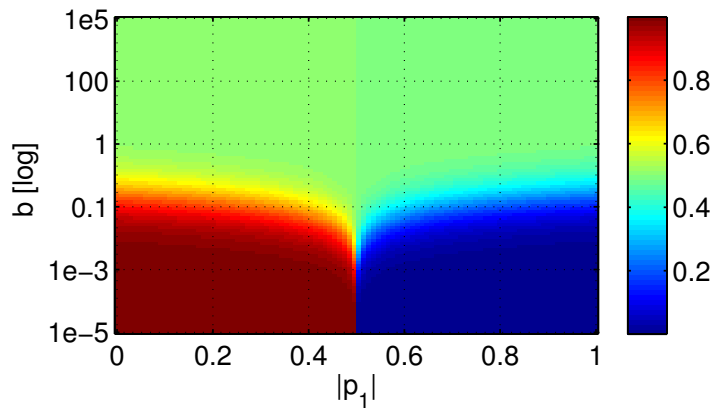


Figure 3.2: Example of the distribution over 2 tasks.

As noted above we denote by $Z_{i,t} = 1$ iff $i = J_t$. The update of the algorithm is performed with the perceptron rule, that is $\mathbf{w}_{J_t,t} = \mathbf{w}_{J_t,t-1} + M_{J_t,t}\, y_{J_t,t}\, \mathbf{x}_{J_t,t}$ and $\mathbf{w}_{i,t} = \mathbf{w}_{i,t-1}$ for $i \neq J_t$. For simplicity of presentation we write the update for all of the tasks in one term as, $\mathbf{w}_{i,t} = \mathbf{w}_{i,t-1} + Z_{i,t}\, M_{i,t}\, y_{i,t}\, \mathbf{x}_{i,t}$. One can notice that although this notation uses labels for all tasks, only the label of the task J_t is used in practice, as for other tasks $Z_{i,t} = 0$. We call this algorithm *SHAMPO* for SHared Annotator for Multiple PrOblems. The pseudo-code of this algorithm appears in Alg. 3.1.

The following theorem states that the expected cumulative number of mistakes that the algorithm makes, may not be higher than for an algorithm that observes the labels of all inputs.

Theorem 1 *If SHAMPO algorithm runs on K tasks with K parallel example pair sequences $(\mathbf{x}_{i,1}, y_{i,1}), \ldots, (\mathbf{x}_{i,n}, y_{i,n}) \in \mathbb{R}^d \times \{-1, 1\}$, $i = 1, ..., K$ with input parameter $b > 0$, then for all $\gamma > 0$, all $\mathbf{u}_i \in \mathbb{R}^d$ and all $n \geq 1$, there exists*

$0 < \delta \leq K$, *such that,*

$$\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right] \leq \frac{\delta}{\gamma}\left[\left(1+\frac{X^2}{2b}\right)\bar{L}_{\gamma,n} + \frac{\left\{2b+X^2\right\}^2 \tilde{U}^2}{8\gamma b}\right], \tag{3.2}$$

where $X = \max_{i,t}\|\mathbf{x}_{i,t}\|$, $\tilde{U}^2 = \sum_{i=1}^{K}\|\mathbf{u}_i\|^2$ *and the expectation is over the random choices of the algorithm.*

Algorithm 3.1 SHAMPO: SHared Annotator for Multiple PrOblems.

Parameters: $b \in \mathbb{R} > 0$.
Initialize: $\mathbf{w}_{i,0} = \mathbf{0}$ for $i = 1, \ldots, K$

for $t = 1, 2, \ldots, n$ **do**

1. Observe K instance vectors, $\mathbf{x}_{i,t}$, $(i = 1, \ldots, K)$.
2. Compute margins $\hat{p}_{i,t} = \mathbf{w}_{i,t-1}^{\top}\mathbf{x}_{i,t}$.
3. Predict K labels, $\hat{y}_{i,t} = \mathrm{sign}(\hat{p}_{i,t})$.
4. Draw task J_t with the distribution:

$$\Pr\left[J_t = j\right] = \frac{\left(b + |\hat{p}_{j,t}| - \min_{m=1}^{K}|\hat{p}_{m,t}|\right)^{-1}}{D_t},$$
$$D_t = \sum_i \left(b + |\hat{p}_{i,t}| - \min_{m=1}^{K}|\hat{p}_{m,t}|\right)^{-1}.$$

5. Query the true label ,$y_{J_t,t} \in \{-1, 1\}$.
6. Set the indicator $M_{J_t,t} = 1$ iff $y_{J_t,t} \neq \hat{y}_{J_t,t}$.
7. Update with the perceptron rule:

$$\begin{aligned}\mathbf{w}_{J_t,t} &= \mathbf{w}_{J_t,t-1} + M_{J_t,t}\, y_{J_t,t}\, \mathbf{x}_{J_t,t} \\ \mathbf{w}_{i,t} &= \mathbf{w}_{i,t-1} \text{ for } i \neq J_t\end{aligned} \tag{3.3}$$

end for
Output: $\mathbf{w}_{i,n}$ for $i = 1, \ldots, K$.

Proof: In this proof we extend the proof of the bound on selective sampling perceptron algorithm of Cesa-Bianchi et al. [2006a], which was based on the standard proof of the Percetron mistake bound, with a few modifications. First, we subtruct a non-negative quantity from the left hand side, which make the bound a little bit

more loose. Then, we sum the inequality on all the tasks and iterations and exploit the dependency of the query selection between tasks, based on the derived probability $\Pr[J_t = j]$. In addition, we add a bound on the scaling quantity, D_t, that appeares in the bound.

Fix n examples sequences, $(\mathbf{x}_{i,1}, y_{i,1}), \ldots, (\mathbf{x}_{i,n}, y_{i,n})$ for each one of the K tasks. Let t be certain trial and i to be an update task on this trial, such that $M_{i,t} = 1$. We can write,

$$\begin{aligned}
\gamma - \ell_{\gamma,i,t}(\mathbf{u}_i) &= \gamma - \left(\gamma - y_{i,t}\mathbf{u}_i^T\mathbf{x}_{i,t}\right)_+ \\
&\leq \; y_{i,t}\mathbf{u}_i^T\mathbf{x}_{i,t} \\
&= y_{i,t}\left(\mathbf{u}_i + \mathbf{w}_{i,t-1} - \mathbf{w}_{i,t-1}\right)^T\mathbf{x}_{i,t} \\
&= y_{i,t}\mathbf{w}_{i,t-1}^T\mathbf{x}_{i,t} + \left(\mathbf{u}_i - \mathbf{w}_{i,t-1}\right)^T y_{i,t}\mathbf{x}_{i,t} \\
&= y_{i,t}\mathbf{w}_{i,t-1}^T\mathbf{x}_{i,t} + \left(\mathbf{u}_i - \mathbf{w}_{i,t-1}\right)^T\left(\mathbf{w}_{i,t} - \mathbf{w}_{i,t-1}\right) \\
&= y_{i,t}\mathbf{w}_{i,t-1}^T\mathbf{x}_{i,t} + \frac{1}{2}\|\mathbf{u}_i - \mathbf{w}_{i,t-1}\|^2 - \frac{1}{2}\|\mathbf{u}_i - \mathbf{w}_{i,t}\|^2 + \frac{1}{2}\|\mathbf{w}_{i,t-1} - \mathbf{w}_{i,t}\|^2 \\
&= y_{i,t}\hat{p}_{i,t} + \frac{1}{2}\|\mathbf{u}_i - \mathbf{w}_{i,t-1}\|^2 - \frac{1}{2}\|\mathbf{u}_i - \mathbf{w}_{i,t}\|^2 + \frac{1}{2}\|\mathbf{w}_{i,t-1} - \mathbf{w}_{i,t}\|^2 \; .
\end{aligned}$$

The last inequality holds for all $\gamma > 0$ and for all $\mathbf{u}_i \in \mathbb{R}^d$, so we can replace γ and $\mathbf{u}_i$ by their scaling $\alpha\gamma$ and $\alpha\mathbf{u}_i$ respectively, where $\alpha > 0$ is a scaling parameter that will be determined shortly. Since the inequality is computed on an update task, $y_{i,t}\hat{p}_{i,t} \leq 0$ holds , i.e. $y_{i,t}\hat{p}_{i,t} = -\left|\hat{p}_{i,t}\right|$, and we get

$$\begin{aligned}
&\alpha\gamma + \left|\hat{p}_{i,t}\right| \leq \\
&\qquad \alpha\ell_{\gamma,i,t}(\mathbf{u}_i) + \frac{1}{2}\|\alpha\mathbf{u}_i - \mathbf{w}_{i,t-1}\|^2 - \frac{1}{2}\|\alpha\mathbf{u}_i - \mathbf{w}_{i,t}\|^2 + \frac{1}{2}\|\mathbf{w}_{i,t-1} - \mathbf{w}_{i,t}\|^2.
\end{aligned} \tag{3.4}$$

Now, in order to generalize the inequality, we multiply it by the indicator $M_{i,t}Z_{i,t}$, which means that the inequality holds for trials and tasks where there is an update,

as stated above.

$$M_{i,t}Z_{i,t}(\alpha\gamma + |\hat{p}_{i,t}|) \leq M_{i,t}Z_{i,t}\alpha\ell_{\gamma,i,t}(\mathbf{u}_i) + \frac{M_{i,t}Z_{i,t}}{2}\left\|\alpha\mathbf{u}_i - \mathbf{w}_{i,t-1}\right\|^2 - \frac{M_{i,t}Z_{i,t}}{2}\left\|\alpha\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 + \frac{M_{i,t}Z_{i,t}}{2}\left\|\mathbf{w}_{i,t-1} - \mathbf{w}_{i,t}\right\|^2. \tag{3.5}$$

Yet, in the other cases, when no update is performed and $M_{i,t}Z_{i,t} = 0$, the equality $\mathbf{w}_{i,t} = \mathbf{w}_{i,t-1}$ holds as well, so we can rid off the indicator for any one of the last three terms and we have

$$M_{i,t}Z_{i,t}(\alpha\gamma + |\hat{p}_{i,t}|) \leq M_{i,t}Z_{i,t}\alpha\ell_{\gamma,i,t}(\mathbf{u}_i) + \frac{1}{2}\left\|\alpha\mathbf{u}_i - \mathbf{w}_{i,t-1}\right\|^2 - \frac{1}{2}\left\|\alpha\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 + \frac{M_{i,t}Z_{i,t}}{2}\left\|\mathbf{w}_{i,t-1} - \mathbf{w}_{i,t}\right\|^2.$$

Next, we sum the inequality above, over t, recall the facts that $\mathbf{w}_{i,0} = 0$ and $\left\|\mathbf{w}_{i,t-1} - \mathbf{w}_{i,t}\right\|^2 \leq X^2$ for $X = \max_{i,t}\left\|\mathbf{x}_{i,t}\right\|$ to get,

$$\sum_{t=1}^{n} M_{i,t}Z_{i,t}\left(\alpha\gamma + |\hat{p}_{i,t}| - \frac{X^2}{2}\right) \leq \alpha\sum_{t=1}^{n} M_{i,t}Z_{i,t}\ell_{\gamma,i,t}(\mathbf{u}_i) + \frac{\alpha^2}{2}\left\|\mathbf{u}_i\right\|^2. \tag{3.6}$$

Substituting $\alpha = (2b + X^2)/2\gamma$ (where $b \in \mathbb{R},\ \ b > 0$) in Eq. (3.6), we get

$$\sum_{t=1}^{n} M_{i,t}Z_{i,t}(b + |\hat{p}_{i,t}|) \leq \frac{2b + X^2}{2\gamma}\sum_{t=1}^{n} M_{i,t}Z_{i,t}\ell_{\gamma,i,t}(\mathbf{u}_i) + \frac{\left(2b + X^2\right)^2}{8\gamma^2}\left\|\mathbf{u}_i\right\|^2.$$

Now, we subtract a non negative quantity $\sum_{t=1}^{n} M_{i,t}Z_{i,t}\min_j |\hat{p}_{j,t}|$ from the l.h.s. and get,

$$\sum_{t=1}^{n} M_{i,t}Z_{i,t}\left(b + |\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}|\right) \leq \frac{2b + X^2}{2\gamma}\sum_{t=1}^{n} M_{i,t}Z_{i,t}\ell_{\gamma,i,t}(\mathbf{u}_i) + \frac{\left(2b + X^2\right)^2}{8\gamma^2}\left\|\mathbf{u}_i\right\|^2. \tag{3.7}$$

At this point we take the expectation of all the terms. Recall that the conditional expectation of $Z_{i,t}$ is $(b + |\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}|)^{-1}/D_t$ and that $M_{i,t}$ and $\hat{p}_{i,t}$ are

measurable with respect to the σ-algebra that generated by $Z_1, \ldots, Z_{t-1}$. We start with the left term,

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{n} M_{i,t} Z_{i,t}\left(b + |\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}|\right)\right] = \\
&\mathbb{E}\left[\mathbb{E}_{t-1}\left[\sum_{t=1}^{n} M_{i,t} Z_{i,t}\left(b + |\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}|\right)\right]\right] = \\
&\mathbb{E}\left[\sum_{t=1}^{n} M_{i,t}\left(b + |\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}|\right)\mathbb{E}_{t-1}\left[Z_{i,t}\right]\right] = \\
&\mathbb{E}\left[\sum_{t=1}^{n} \frac{M_{i,t}}{D_t}\right] .
\end{aligned}$$

Substituting the last term in the expectation of Eq. (3.7),

$$\mathbb{E}\left[\sum_{t=1}^{n} \frac{M_{i,t}}{D_t}\right] \leq \frac{2b + X^2}{2\gamma} \bar{L}_{\gamma,i,n}(u_i) + \frac{\left(2b + X^2\right)^2}{8\gamma^2} \|\mathbf{u}_i\|^2 . \tag{3.8}$$

Since that the normalization factor is a sum of positive components, hence D_t is maximal when each of the components gets its maximal value. This happens when $|\hat{p}_{m,t}| = |\hat{p}_{j,t}| \quad \forall m, j \in \{1, \ldots K\}$. This allows us to bound the normalization factor as follows,

$$D_t = \sum_{m=1}^{K} \left(b + |\hat{p}_{m,t}| - \min_j |\hat{p}_{j,t}|\right)^{-1} \leq \sum_{m=1}^{K} b^{-1} = \frac{K}{b} .$$

Since $M_{i,t} > 0 \; \forall i, t$, thus there exists $\delta_i \in \mathbb{R} > 0$ such that

$$\mathbb{E}\left[\sum_{t=1}^{n} \frac{M_{i,t}}{D_t}\right] = \frac{b}{\delta_i} \mathbb{E}\left[\sum_{t=1}^{n} M_{i,t}\right] . \tag{3.9}$$

and

$$\frac{b}{\delta_i} \geq \min \frac{1}{D_t} = \frac{1}{\max D_t} \geq \frac{b}{K} . \tag{3.10}$$

Which implies that $0 < \delta_i \leq K$.

Plugging Eq. (3.9) in Eq. (3.8) we get,

$$\frac{b}{\delta_i}\mathbb{E}\left[\sum_{t=1}^{n} M_{i,t}\right] \leq \frac{2b+X^2}{2\gamma}\bar{L}_{\gamma,i,n}(\mathbf{u}_i) + \frac{\left(2b+X^2\right)^2}{8\gamma^2}\|\mathbf{u}_i\|^2.$$

Summing up the last inequality over all K tasks and setting $\delta = \max_i \delta_i$ yields,

$$\frac{1}{\delta}\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right] \leq \frac{1}{\gamma}\left(1+\frac{X^2}{2b}\right)\bar{L}_{\gamma,n} + \frac{\left(2b+X^2\right)^2}{8\gamma^2 b}\sum_{i=1}^{K}\|\mathbf{u}_i\|^2\,, \qquad (3.11)$$

which concludes the proof. ■

One can use the bound to tune the algorithm for an optimal value of b. However, this may not be possible as unfortunately $\bar{L}_{\gamma,n}$ depends implicitly on the value of b (Similar issue appears also after the discussion of Theorem 1 in a different context by Cesa-Bianchi et al. [2006c]).
Alternatively, we can take a loose estimate of $\bar{L}_{\gamma,n}$, and replace it with $L_{\gamma,n}$ (which is $\sim K$ times larger). The optimal value of b can be now calculated,

$$b = \frac{X^2}{2}\sqrt{1+\frac{4\gamma L_{\gamma,n}}{U^2X^2}}.$$

Substituting this value in the bound of Eq. (3.2) with $L_{\gamma,n}$ leads to the following bound,

$$\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right] \leq \frac{\delta}{\gamma}\left[L_{\gamma,n} + \frac{U^2X^2}{2\gamma} + \frac{U^2}{2\gamma}\sqrt{1+\frac{4\gamma L_{\gamma,n}}{U^2X^2}}\right]$$

which has the same dependency in the number of inputs n as algorithm that observes all of them.

We continue now with an analysis of an extension of our algorithm that allows more than a single query per round. In this setting, the ability of the annotator to annotate data instances is less limited. Here, we allow the algorithm to query κ labels at a time ($\kappa < K$), instead of one. On each iteration t, the modified algorithm samples without repetitions, κ labels to be annotated, and perform the same update as of Eq. (3.3). Formally, on each round we have $\sum_i Z_{i,t} = \kappa$ for $Z_{i,t} \in \{0,1\}$ where the first task-index to be queried is drawn according to Eq. (3.1). The second task is drawn from the same distribution, eliminating the first choice and adjusting

the normalization factor, and so on. Once κ problems are drawn, the algorithm receives κ labels for the κ corresponding inputs, and updates the κ models according to Eq. (3.3).

Corollary 2 *If SHAMPO algorithm gets feedback for κ tasks on each round, instead of only a single problem, the expected cumulative weighted mistakes is bounded as follows*

$$\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right] \leq \frac{1}{C(\kappa,K)}\left(\frac{1}{\gamma}\left(1+\frac{X^2}{2b}\right)\bar{L}^{\kappa}_{\gamma,n} + \kappa\frac{\left(2b+X^2\right)^2}{8\gamma^2 b}\tilde{U}^2\right) ,$$

where $C(\kappa,K) = \sum_{j=K-\kappa+1}^{K}\frac{1}{j}$, and $\bar{L}^{\kappa}_{\gamma,n}$ is the expected loss of K models $\{\mathbf{u}_i\}$ over the κ instances that are annotated per round t.

Proof: We follow the proof of Theorem 3.1 until Eq. (3.11), recall $\delta_i < K$. We repeat the process κ times, and get the equivalent inequality for sampling κ problems without repetitions,

$$\left(\sum_{j=K-\kappa+1}^{K}\frac{1}{j}\right)\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right] \leq \frac{1}{\gamma}\left(1+\frac{X^2}{2b}\right)\bar{L}^{\kappa}_{\gamma,n} + \kappa\frac{\left(2b+X^2\right)^2}{8\gamma^2 b}U^2 , \tag{3.12}$$

where all expectations are now with respect to the sampling repetitions, and specifically $\bar{L}^{\kappa}_{\gamma,n}$ is the expected loss of a set of linear models $\{\mathbf{u}_i\}$ where κ problems are sampled rather than a single one. ■

It can be shown that this bound is a generalized form of Theorem 3.1. For $\kappa = 1$ we get the bound of Theorem 3.1 (with $\delta = K$), while for $\kappa = K$ we have $C(\kappa,K) > \ln(K+1)$, thus recovering the bound of K Perceptron trained in parallel, with a sightly $(\ln(K+1))$ worse dependency in the number of tasks.

3.2 Kernel-Based SHAMPO

SHAMPO algorithm can also be used with Kernels in order to get a non linear classifiers in case of non separated data set, by transforming the feature vector $\mathbf{x}$

to $\Phi(\mathbf{x})$. Define the binary parameter $\alpha_{i,t} = Z_{i,t}M_{i,t}$, we can rewrite the weight vector $\mathbf{w}_{i,t}$ in its dual form as a linear combination of the input vectors

$$\mathbf{w}_{i,t} = \mathbf{w}_{i,t-1} + Z_{i,t}M_{i,t}y_{i,t}\mathbf{x}_{i,t} = \sum_{s=1}^{t} Z_{i,s}M_{i,s}y_{i,s}\mathbf{x}_{i,s} = \sum_{s=1}^{t} \alpha_{i,s}y_{i,s}\mathbf{x}_{i,s}.$$

Now wee can rid of the weight algorithm and store $\alpha_{i,t}$ values and the relevant input-label pairs. Doing that, we get a new representation of the margin

$$\begin{aligned}\hat{p}_{i,t} = \mathbf{w}_{i,t-1}^{\top}\mathbf{x}_{i,t} = \left(\sum_{s=1}^{t} \alpha_{i,s}y_{i,s}\mathbf{x}_{i,s}\right)^{\top}\mathbf{x}_{i,t} &= \sum_{s=1}^{t} \alpha_{i,s}y_{i,s}\left({\mathbf{x}_{i,s}}^{\top}\mathbf{x}_{i,t}\right) \\ &= \sum_{s=1}^{t} \alpha_{i,s}y_{i,s}K(\mathbf{x}_{i,s}, \mathbf{x}_{i,t})\end{aligned}$$

where $K(\mathbf{x}, \boldsymbol{v}) = \Phi(\mathbf{x})^{\top}\Phi(\boldsymbol{v})$ is the kernel function that represents the inner product of a high dimension space.

3.3 Aggressive SHAMPO

So far, we used the margin to build the distribution over the tasks, but in the update stage, we used the same regular perceptron update rule, that makes an update only when there is a prediction mistake. However, we already said that the margin can tells us about the uncertainty in the prediction, so it may be helpful to proceed with this approach in the update stage as well and make the strict update threshold a little bit softer.

In this version of SHAMPO algorithm, the update is been preformed not only when there is a mistake, but also in the case of correct prediction with low margin, i.e. low certainty. We define this margin to be $\lambda \in \mathbb{R} > 0$. In addition to the previous mistake indicator, $M_{i,t}$, we introduce one more indicator, $G_{i,t}$. We keep the notation of $M_{i,t} = 1$ when there is wrong prediction on the task i in round t and $M_{i,t} = 0$ otherwise, and set $G_{i,t} = 1$ when there is no prediction mistake on the task i in round t, but the margin is lower than our threshold, i.e. $0 < |\hat{p}_{i,t}| < \lambda$, therefore an aggressive update takes place, and $G_{i,t} = 0$ otherwise. An update is performed if either there is a mistake ($M_{J_i,t} = 1$) or the margin is low ($G_{J_i,t} = 1$). Note that these events are mutually exclusive thus, for simplicity, we define the

update indicator which is $U_{i,t} = M_{i,t} + G_{i,t}$. This indicator is $U_{i,t} = 1$ on update trials and $U_{i,t} = 0$ when no update has been made.

Algorithm 3.2 SHAMPO aggressive perceptron.

Parameters: $b, 0 \leq \lambda \in \mathbb{R} \leq b/2$.
Initialize: $\mathbf{w}_{i,0} = \mathbf{0}$ for $i = 1, \ldots, K$

for $t = 1, 2, \ldots, n$ **do**

1. Observe K instance vectors, $\mathbf{x}_{i,t}$, $(i = 1, \ldots, K)$.
2. Compute margins $\hat{p}_{i,t} = \mathbf{w}_{i,t-1}^{\top} \mathbf{x}_{i,t}$.
3. Predict K labels, $\hat{y}_{i,t} = \text{sign}(\hat{p}_{i,t})$.
4. Draw problem J_t with the distribution:

$$\Pr\left[J_t = j\right] = \frac{\left(b + |\hat{p}_{j,t}| - \min_{m=1}^{K} |\hat{p}_{m,t}|\right)^{-1}}{D_t},$$
$$D_t = \sum_i \left(b + |\hat{p}_{i,t}| - \min_{m=1}^{K} |\hat{p}_{m,t}|\right)^{-1}.$$

5. Query the true label ,$y_{J_t,t} \in \{-1, 1\}$.
6. Set the indicator $M_{J_t,t} = 1$ iff $y_{J_t,t} \neq \hat{y}_{J_t,t}$.
7. Iff $M_{J_t,t} = 0$ and $|p_{J_t,t}| < \lambda$, set the indicator $G_{J_t,t} = 1$
8. For $U_{i,t} = M_{i,t} + G_{i,t}$ update with the perceptron rule:

$$\mathbf{w}_{J_t,t} = \mathbf{w}_{J_t,t-1} + U_{J_t,t}\, y_{J_t,t}\, \mathbf{x}_{J_t,t}$$
$$\mathbf{w}_{i,t} \;\; = \mathbf{w}_{i,t-1} \text{ for } i \neq J_t$$

end for
Output: $\mathbf{w}_{i,n}$ for $i = 1, \ldots, K$.

Theorem 3 *If Aggressive SHAMPO algorithm runs on K problems with K parallel example pair sequences $(\mathbf{x}_{i,1}, y_{i,1}), \ldots, (\mathbf{x}_{i,n}, y_{i,n}) \in \mathbb{R}^d \times \{-1, 1\}$, $i = 1, ..., K$ with input parameters $b > 0$, and $0 \leq \lambda \leq b/2$ then for all $\gamma > 0$, all $\mathbf{u}_i \in \mathbb{R}^d$ and all $n \geq 1$, there exists $0 < \delta \leq K$, such that,*

$$\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right] \leq \frac{\delta}{\gamma}\left[\left(1+\frac{X^2}{2b}\right)\bar{L}_{\gamma,n} + \frac{\left(2b+X^2\right)^2\tilde{U}^2}{8\gamma b}\right] + \left(2\frac{\lambda}{b}-1\right)\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} G_{i,t}\right] ,$$

where $X = \max_{i,t}\|\mathbf{x}_{i,t}\|$, $\tilde{U}^2 = \sum_{i=1}^{K}\|\mathbf{u}_i\|^2$ *and the expectation is over the random choices of the algorithm.*

The theorem above, states that when the update is aggressive, the bound on the expected number of errors can be tighter than the non aggressive algorithm because we have added a new term to the bound which can be negative.

In order to achieve tighter bound on the expected cumulative mistakes than we had in the regular SHAMPO perceptron algorithm (Theorem 3.1), we need to restrict the aggressive margin λ to be $\lambda \leq b/2$ so the last term will be negative. Indeed, one can say that this bound shows that the best choice of λ is $\lambda = 0$, i.e. it seems that aggressive update is not a good idea. However, in such case, $G_{i,t} = 0$, $\forall i, t$ which means that this would not be the best choice of λ. This discrimination actually shows that there is a sweet point where $0 < \lambda < b/2$ such that the bound will be the tightest as this bound allows .

Proof: This proof is derived from the proof of the non-aggressive algorithm, when now, we exploit the fact that expectation is additive and split the expectation on the left hand side to the mistake update part and aggressive update part, which we restrict to be non-nagative such that it may lead to a bound that is tighter than the one of the non-aggressive version.

We follow the proof of Theorem 3.1 on an update trial, reminding that now the update indicator is $U_{i,t} = 1$, until we get,

$$\gamma - \ell_{\gamma,i,t}(\mathbf{u}_i) \leq y_{i,t}\hat{p}_{i,t} + \frac{1}{2}\|\mathbf{u}_i - \mathbf{w}_{i,t-1}\|^2 - \frac{1}{2}\|\mathbf{u}_i - \mathbf{w}_{i,t}\|^2 + \frac{1}{2}\|\mathbf{w}_{i,t-1} - \mathbf{w}_{i,t}\|^2 .$$

Whereas in the non aggressive version proof we claimed that $y_{i,t}\hat{p}_{i,t} = -|\hat{p}_{i,t}|$ in an update trial, here, it is not true when an aggressive update takes place since we update when the prediction is correct as well. For that reason, we remain with the

term $y_{i,t}\hat{p}_{i,t}$ as is for now and we will deal with it later. In addition, in this proof, we change the update indicator from $M_{i,t}$ to $U_{i,t}$. Considering these two changes, we proceed the proof until $Eq.$ (3.7). Now we have,

$$\begin{aligned}\sum_{t=1}^{n} U_{i,t} Z_{i,t}\left(b - y_{i,t}\hat{p}_{i,t} - \min_j |\hat{p}_{j,t}|\right) \leq \\ \frac{2b + X^2}{2\gamma}\sum_{t=1}^{n} U_{i,t} Z_{i,t}\ell_{\gamma,i,t}(\mathbf{u}_i) + \frac{\left(2b + X^2\right)^2}{8\gamma^2}\left\|\mathbf{u}_i\right\|^2 .\end{aligned} \tag{3.13}$$

Since the inequality is computed on an update task, there is a need to discriminate between the two possible options: either $y_{i,t}\hat{p}_{i,t} \leq 0$ holds when there is prediction mistake ($M_{i,t} = 1$), i.e. $y_{i,t}\hat{p}_{i,t} = -|\hat{p}_{i,t}|$, or $0 \leq y_{i,t}\hat{p}_{i,t} \leq \lambda$ holds when the prediction is correct, but an aggressive update occurs ($G_{i,t} = 1$) and in that case $y_{i,t}\hat{p}_{i,t} = |\hat{p}_{i,t}|$.

At this point we take the expectation of all the terms as before taking under consideration the two cases. Recall that the conditional expectation of $Z_{i,t}$ is $(b + |\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}|)^{-1}/D_t$ and that $M_{i,t}$ and $G_{i,t}$ (therefor also $U_{i,t}$) and $\hat{p}_{i,t}$ are measurable with respect to the σ-algebra that generated by $Z_1, \ldots, Z_{t-1}$. We start with the left term,

$$\begin{aligned}&\mathbb{E}\left[\sum_{t=1}^{n} U_{i,t} Z_{i,t}\left(b - y_{i,t}\hat{p}_{i,t} - \min_j |\hat{p}_{j,t}|\right)\right] \\ &= \mathbb{E}\left[\mathbb{E}_{t-1}\left[\sum_{t=1}^{n} U_{i,t} Z_{i,t}\left(b - y_{i,t}\hat{p}_{i,t} - \min_j |\hat{p}_{j,t}|\right)\right]\right] \\ &= \mathbb{E}\left[\sum_{t=1}^{n} U_{i,t}\left(b - y_{i,t}\hat{p}_{i,t} - \min_j |\hat{p}_{j,t}|\right)\mathbb{E}_{t-1}\left[Z_{i,t}\right]\right] \\ &= \mathbb{E}\left[\sum_{t=1}^{n} \frac{(M_{i,t} + G_{i,t})}{D_t}\frac{(b - y_{i,t}\hat{p}_{i,t} - \min_j |\hat{p}_{j,t}|)}{b + |\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}|}\right] \\ &= \mathbb{E}\left[\sum_{t=1}^{n} \frac{1}{D_t}\left(M_{i,t} + \frac{b - |\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}|}{b + |\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}|} G_{i,t}\right)\right] .\end{aligned} \tag{3.14}$$

Substituting the last term in the expectation of Eq. (3.13) we get,

$$\mathbb{E}\left[\sum_{t=1}^{n}\frac{1}{D_t}\left(M_{i,t}+\frac{b-|\hat{p}_{i,t}|-\min_j|\hat{p}_{j,t}|}{b+|\hat{p}_{i,t}|-\min_j|\hat{p}_{j,t}|}G_{i,t}\right)\right]\leq \frac{2b+X^2}{2\gamma}\bar{L}_{\gamma,i,n}(u_i)+\frac{\left(2b+X^2\right)^2}{8\gamma^2}\|\mathbf{u}_i\|^2 . \tag{3.15}$$

Now we bound the factor that is multiplied by $G_{i,t}$ using the fact that $G_{i,t}=1$ only when $|\hat{p}_{i,t}|<\lambda$,

$$\left(1-2\frac{\lambda}{b}\right)=\frac{b-2\lambda}{b}\leq\frac{b-|\hat{p}_{i,t}|-\min_j|\hat{p}_{j,t}|}{b+|\hat{p}_{i,t}|-\min_j|\hat{p}_{j,t}|}\leq 1. \tag{3.16}$$

We can use now Eq. (3.16) to substitute the left hand side of Eq. (3.15) with the lower term

$$\mathbb{E}\left[\sum_{t=1}^{n}\frac{1}{D_t}\left(M_{i,t}+\left(1-2\frac{\lambda}{b}\right)G_{i,t}\right)\right].$$

Since $b/2\geq\lambda$ we have that $M_{i,t}+\left(1-2\frac{\lambda}{b}\right)G_{i,t}\geq 0$, and as stated before in Eq. (3.10), there exists $\delta_i\in\mathbb{R},\ 0<\delta_i\leq K$ such that,

$$\mathbb{E}\left[\sum_{t=1}^{n}\frac{1}{D_t}\left(M_{i,t}+\left(1-2\frac{\lambda}{b}\right)G_{i,t}\right)\right]=\frac{b}{\delta_i}\mathbb{E}\left[\sum_{t=1}^{n}\left(M_{i,t}+\left(1-2\frac{\lambda}{b}\right)G_{i,t}\right)\right].$$

Plugging this in Eq. (3.15) we get,

$$\frac{b}{\delta_i}\mathbb{E}\left[\sum_{t=1}^{n}M_{i,t}\right]\leq\frac{2b+X^2}{2\gamma}\bar{L}_{\gamma,i,n}(\mathbf{u}_i)+\frac{\left(2b+X^2\right)^2}{8\gamma^2}\|\mathbf{u}_i\|^2+\frac{b}{\delta_i}\left(2\frac{\lambda}{b}-1\right)\mathbb{E}\left[\sum_{t=1}^{n}G_{i,t}\right].$$

Summing up the last inequality over all K tasks and setting $\delta = \max_i \delta_i$ yields,

$$\frac{1}{\delta}\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right] \leq \frac{1}{\gamma}\left(1+\frac{X^2}{2b}\right)\bar{L}_{\gamma,n} + \frac{\left(2b+X^2\right)^2}{8\gamma^2 b}\sum_{i=1}^{K}\|\mathbf{u}_i\|^2 + \frac{1}{\delta}\left(2\frac{\lambda}{b}-1\right)\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} G_{i,t}\right]$$

Which concludes the proof ■

3.4 SHAMPO with Prior Knowledge

We saw before, that b is a tradeoff parameter between exploitation and uniform distribution over the tasks (exploration). It is possible that we have prior knowledge about the tasks (which tasks are harder than the others). In this case, we would like to add this prior knowledge to the distribution such that the exploration will be done with a prior distribution. This is especially important in the first steps of the algorithms when instead starting in pure uniform exploration (since we initialize our model vectors, $\mathbf{w}_{i,0} = \mathbf{0}$), we can start with a reasonable prior distribution. In order to do that we change the probability to be

$$\Pr\left[J_t = j\right] = \frac{1}{D_t}\frac{a_j}{\left(b + |\hat{p}_{j,t}| - \min_{m=1}^{K}|\hat{p}_{m,t}|\right)}$$

where $a_j > 1 \;\; \forall j \in \{1,\ldots,K\}$ and D_t is the normalization factor. I fact, the bound on a_i is for the analysis only and even lower nonnegative values are acceptable since we normalize it. We can use this distribution with the SHAMPO perceptron algorithm, but for generalization, we would like to analyze the bound of the SHAMPO aggressive perceptron algorithm.

Theorem 4 *If Aggressive SHAMPO algorithm with prior runs on K problems with K parallel example pair sequences $(\mathbf{x}_{i,1}, y_{i,1}), \ldots, (\mathbf{x}_{i,n}, y_{i,n}) \in \mathbb{R}^d \times \{-1, 1\}$, $i = 1, ..., K$ with input parameters $b > 0$, and $\lambda > 0$ and $a_i > 1 \; \forall i \in \{1, \cdots, K\}$ then for all $\gamma > 0$, all $\mathbf{u}_i \in \mathbb{R}^d$ and all $n \geq 1$, there exists $0 < \delta \leq \sum_{j=1}^{K} a_j$, such that,*

$$\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right] \leq \frac{\delta}{\gamma}\left[\left(1+\frac{X^2}{2b}\right)\bar{L}_{\gamma,n} + \frac{\left(2b+X^2\right)^2 \tilde{U}^2}{8\gamma b}\right] + \left(2\frac{\lambda}{b}-1\right)\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} a_i G_{i,t}\right],$$

where $X = \max_{i,t}\|\mathbf{x}_{i,t}\|$, $\tilde{U}^2 = \sum_{i=1}^{K}\|\mathbf{u}_i\|^2$ *and the expectation is over the random choices of the algorithm.*

This bound brings an advantage and disadvantage. We can reduce the last term in the bound choosing the right weights for a_i, but then, we suffer a factor of $\sum_{j=1}^{K} a_j$ in the rest part of the bound. In addition, now the expected loss is changed a bit to be

$$\bar{L}_{\gamma,n} = \mathrm{E}\left[\sum_{t=1}^{n}\sum_{i=1}^{K} U_{i,t} Z_{i,t} \ell_{\gamma,i,t}(\mathbf{u}_i)\right],$$

since we update sometimes on a correct prediction.

Proof: Here we extend the proof of Theorem 3, while now, the expectation of the random variable is affected by the prior variables, which may lead to a bound which is even tighter than the one of the aggressive algorithm due to the prior coeffitients that increases the effect of the aggressiveness term.

We follow the proof of Theorem 3 until Eq. (3.13), than we take the expectation of this term. First, we take the expectation of the left hand side where now the conditional expectation of $Z_{i,t}$ changed to

$$\mathbb{E}\left[Z_{i,t}\right] = \frac{1}{D_t}\frac{a_i}{\left(b+|\hat{p}_{i,t}| - \min_{m=1}^{K}|\hat{p}_{m,t}|\right)}.$$

Doing that, the expectation becomes

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{n} U_{i,t} Z_{i,t}\left(b - y_{i,t}\hat{p}_{i,t} - \min_j |\hat{p}_{j,t}|\right)\right] \\
&= \mathbb{E}\left[\mathbb{E}_{t-1}\left[\sum_{t=1}^{n} U_{i,t} Z_{i,t}\left(b - y_{i,t}\hat{p}_{i,t} - \min_j |\hat{p}_{j,t}|\right)\right]\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{n} \frac{a_i}{D_t}\left(M_{i,t} + \frac{b - |\hat{p}_{j,t}| - \min_j |\hat{p}_{j,t}|}{b + |\hat{p}_{j,t}| - \min_j |\hat{p}_{j,t}|} G_{i,t}\right)\right] .
\end{aligned}$$

Recall that $a_i \geq 1 \forall i$, we can bound $M_{i,t} \leq M_{i,t} a_i$ and get

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{n} \frac{1}{D_t}\left(M_{i,t} + \frac{b - |\hat{p}_{j,t}| - \min_j |\hat{p}_{j,t}|}{b + |\hat{p}_{j,t}| - \min_j |\hat{p}_{j,t}|} G_{i,t} a_i\right)\right] \leq \\
&\qquad \frac{2b + X^2}{2\gamma} \bar{L}_{\gamma,i,n}(u_i) + \frac{\left(2b + X^2\right)^2}{8\gamma^2} \|\mathbf{u}_i\|^2 .
\end{aligned} \tag{3.17}$$

We use now the same bound as in Eq. (3.16) and plug it into the left side of the inequality,

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{n} \frac{1}{D_t}\left(M_{i,t} + \left(1 - 2\frac{\lambda}{b}\right) G_{i,t} a_i\right)\right] \leq \\
&\qquad \mathbb{E}\left[\sum_{t=1}^{n} \frac{1}{D_t}\left(M_{i,t} + \frac{b - |\hat{p}_{j,t}| - \min_j |\hat{p}_{j,t}|}{b + |\hat{p}_{j,t}| - \min_j |\hat{p}_{j,t}|} G_{i,t} a_i\right)\right] .
\end{aligned}$$

Since $b/2 \geq \lambda$ we have that $M_{i,t} + \left(1 - 2\frac{\lambda}{b}\right) G_{i,t} a_i \geq 0$, thus there exists δ_i such that,

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{n} \frac{1}{D_t}\left(M_{i,t} + \left(1 - 2\frac{\lambda}{b}\right) G_{i,t} a_i\right)\right] = \\
&\qquad \frac{b}{\delta_i} \mathbb{E}\left[\sum_{t=1}^{n}\left(M_{i,t} + \left(1 - 2\frac{\lambda}{b}\right) G_{i,t} a_i\right)\right] ,
\end{aligned} \tag{3.18}$$

and

$$\frac{b}{\delta_i} \geq \min \frac{1}{D_t} = \frac{1}{\max D_t} \geq \frac{b}{\sum_{j=1}^{K} a_j} \ ,$$

where the last inequality follows from,

$$D_t = \sum_{j=1}^{K} \frac{a_j}{\left(b + |\hat{p}_{j,t}| - \min_{m=1}^{K} |\hat{p}_{m,t}|\right)} \leq \frac{\sum_{j=1}^{K} a_j}{b}.$$

The last bound implies that $0 < \delta_i \leq \sum_{i=1}^{K} a_i$.
Combining Eq. (3.17) and Eq. (3.18) leads to,

$$\frac{b}{\delta_i}\mathbb{E}\left[\sum_{t=1}^{n} M_{i,t}\right] \leq \frac{2b + X^2}{2\gamma}\bar{L}_{\gamma,i,n}(\mathbf{u}_i) + \frac{\left(2b + X^2\right)^2}{8\gamma^2}\|\mathbf{u}_i\|^2 + \frac{b}{\delta_i}\left(2\frac{\lambda}{b} - 1\right) a_i \mathbb{E}\left[\sum_{t=1}^{n} G_{i,t}\right].$$

Summing up the last inequality over all K tasks and setting $\delta = \max_i \delta_i$ yields,

$$\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right] \leq \frac{\delta}{\gamma}\left[\left(1 + \frac{X^2}{2b}\right)\bar{L}_{\gamma,n} + \frac{\left(2b + X^2\right)^2}{8\gamma b}\sum_{i=1}^{K}\|\mathbf{u}_i\|^2\right] + \left(2\frac{\lambda}{b} - 1\right)\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} a_i G_{i,t}\right] \ ,$$

which concludes the proof. ■

We see that this bound is a global bound for the last two algorithms. If we set $a_i = 1, \ \forall i \in \{1, \cdots, K\}$ we get the same bound as SHAMPO aggressive algorithm and if we update only when a prediction mistake occurs, it means that $G_{i,t} = 0, \ \forall i \in \{a, \cdots, K\}, t > 0$ and we get the bound of the ordinary SHAMPO perceptron algorithm.

3.5 Adaptive Parameter SHAMPO

As it has been proved in the analysis and will be shown in experiments, the algorithm parameter b can tune the distribution over the tasks between exploration

and exploitation, yet, there is still a need to supply the algorithm with this input parameter. So far, all of the tasks sheared the same parameter b which was determined at the beginning of the algorithm and remained constant during the whole process. For this reason, once the initial parameters were supplied, the margin was the only factor that had a varying influence on the distribution. Now, we would like to suggest another algorithm that allow us to dominate the distribution using the number of updates for each task. We assume as in Cesa-Bianchi et al. [2006c] that the algorithm makes more mistakes on harder task than easier one, so a natural choice at any time, can be one that set higher probability to the tasks that made more prediction mistakes up to the same time.

The adaptive algorithm is similar to the one in Alg. 3.1 with three variations in order to comply with the intuition. Here we introduce two more variables per task. First, the scalar $N_{i,n} = \sum_{t=1}^{n} Z_{i,t} M_{i,t}$ that holds the number of updates that has been done for the ith task up to the time step n. For simplicity of analysis we will write it in it's recursive way $N_{i,t} = N_{i,t-1} + Z_{i,t} M_{i,t}$. The second variable is $X_{i,t}$ which is holds at the end of each cycle, the maximal norm of the input vectors that were involved in the updates. The algorithm still have a parameter $\beta > 0 \in \mathbb{R}$ to be determined, however, it has less influence on the distribution than the parameter b in Alg. 3.1 . Using those three new parameters, we introduce the adaptive version of b

$$b_{i,t-1} = \beta X_{i,t} \sqrt{1 + N_{i,t-1}}$$

The adaptive version of the SHAMPO perceptron algorithm is described in Alg. 3.3.

Theorem 5 *If adaptive SHAMPO algorithm runs on K tasks with K parallel example pair sequences $(\mathbf{x}_{i,1}, y_{i,1}), ...(\mathbf{x}_{i,n}, y_{i,n}) \in \mathbb{R}^d \times \{-1, 1\}$, $i = 1, ..., K$ with input parameter $\beta > 0$, then for all $\gamma > 0$, all $\mathbf{u}_i \in \mathbb{R}^d$ and all $n \geq 1$, there exists $0 < \delta \leq K$, such that,*

$$\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right] \leq \delta\left[\frac{\delta B^2}{2} + \frac{1}{\gamma}\bar{L}_{\gamma,n} + \frac{KR}{2\beta} + B\sqrt{\frac{\delta B^2}{4} + \frac{1}{\gamma}\bar{L}_{\gamma,n} + \frac{KR}{2\beta}}\right]$$

where $R = \max_i \left(\|u_i\| X_i\right)/\gamma$, $B = \left(R + \frac{2+3R}{2\beta}\right)$ and the expectation is over the random choices of the algorithm.

Algorithm 3.3 Adaptive SHAMPO algorithm

Parameters: $\beta \in \mathbb{R} > 0$.
Initialize: $\mathbf{w}_{i,0} = \mathbf{0}, N_{i,0} = 0, \tilde{X}_{i,0} = 0$ for $i = 1, \ldots, K$

for $t = 1, 2, ..., n$ **do**

1. Observe K instance vectors, $\mathbf{x}_{i,t}$, $(i = 1, \ldots, K)$.
2. Compute margins $\hat{p}_{i,t} = \mathbf{w}_{i,t-1}^{\top} \mathbf{x}_{i,t}$.
3. Predict K labels, $\hat{y}_{i,t} = \text{sign}(\hat{p}_{i,t})$.
4. Set $\tilde{X}_{i,t} = \max\left(X_{i,t-1}, \|\mathbf{x}_{i,t}\|\right)$
5. Draw task J_t with the distribution:

$$\Pr\left[J_t = j\right] = \frac{1}{D_t} \frac{b_{i,t}}{b_{i,t} + |\hat{p}_{j,t}| - \min_{m=1}^{K} |\hat{p}_{m,t}|},$$
$$D_t = \sum_i \frac{b_{i,t}}{b_{i,t} + |\hat{p}_{j,t}| - \min_{m=1}^{K} |\hat{p}_{m,t}|}.$$

   Where $b_{i,t} = \beta \tilde{X}_{i,t}^2 \sqrt{1 + N_{i,t}}$.
6. Query the true label ,$y_{J_t,t} \in \{-1, 1\}$.
7. Set the indicator $M_{J_t,t} = 1$ iff $y_{J_t,t} \neq \hat{y}_{J_t,t}$.
8. Update model's parameters:

$$\begin{aligned}
&\mathbf{w}_{J_t,t} = \mathbf{w}_{J_t,t-1} + M_{J_t,t}\, y_{J_t,t}\, \mathbf{x}_{J_t,t} \\
&N_{J_t,t} = N_{J_t,t-1} + M_{J_t,t} \\
&X_{J_t,t} = \tilde{X}_{J_t,t} \\
&\mathbf{w}_{i,t} = \mathbf{w}_{i,t-1}, \;\; N_{i,t} = N_{i,t-1}, \;\; X_{i,t} = X_{i,t-1} \quad \text{for } i \neq J_t
\end{aligned}$$

end for
Output: $\mathbf{w}_{i,n}$ for $i = 1, \ldots, K$.

Proof: The proof here is an extended version of the proof in Theorem 3.1. As in Cesa-Bianchi et al. [2004], we replace the scaling variable with a time varying variable while we start with a bound on a single task, which is similar to the selective sampling proof, and we continue by summing both sides of the inequality over all tasks. The expectation here is with respect to the new probabilty that is built of K different b values, on for each task.

We start here from Eq. (3.4) and replace the constant scaling parameter, for-

merly α, with a time varying factor $c_{i,t-1}/\gamma$, where $c_{i,t} \geq 0$ is defined as following

$$c_{i,t-1} = \frac{1}{2}\left(\max\left\{X_{i,t-1}, ||\mathbf{x}_{i,t}||\right\}\right) + b_{i,t-1}.$$

Plugging the new scaling factor into the given inequality we get

$$\begin{aligned}
&M_{i,t}Z_{i,t}(c_{i,t-1} + |\hat{p}_{i,t}|) \leq M_{i,t}Z_{i,t}\frac{c_{i,t-1}}{\gamma}\ell_{\gamma,i,t}(\mathbf{u}_i) + \\
&\quad \frac{1}{2}\left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t-1}\right\|^2 - \frac{1}{2}\left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 + \frac{M_{i,t}Z_{i,t}}{2}\left\|\mathbf{w}_{i,t-1} - \mathbf{w}_{i,t}\right\|^2.
\end{aligned}$$

From the update rule of the algorithm we can bound $\|\mathbf{w}_{i,t-1} - \mathbf{w}_{i,t}\|^2 \leq \|\mathbf{x}_{i,t}\|^2$. This inequality holds for any time step, in particular for an update iteration. Using this bound and dividing the inequality by $b_{i,t-1}$ yields

$$\begin{aligned}
M_{i,t}Z_{i,t}\left(\frac{c_{i,t-1} + |\hat{p}_{i,t}| - \|\mathbf{x}_{i,t}\|^2/2}{b_{i,t-1}}\right) &\leq M_{i,t}Z_{i,t}\frac{c_{i,t-1}}{b_{i,t-1}}\frac{\ell_{\gamma,i,t}(\mathbf{u}_i)}{\gamma} \\
&+ \frac{1}{2b_{i,t-1}}\left(\left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t-1}\right\|^2 - \left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2\right).
\end{aligned} \tag{3.19}$$

Now, we decompose the last term of the inequality to two differences,

$$\begin{aligned}
&\frac{1}{2b_{i,t-1}}\left(\left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t-1}\right\|^2 - \left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2\right) = \\
&\quad \frac{1}{2b_{i,t-1}}\left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t-1}\right\|^2 - \frac{1}{2b_{i,t}}\left\|\frac{c_{i,t}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 + \\
&\quad \frac{1}{2b_{i,t}}\left\|\frac{c_{i,t}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 - \frac{1}{2b_{i,t-1}}\left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2.
\end{aligned}$$

At this point expand the last two terms such that

$$\begin{aligned}
&\frac{1}{2b_{i,t}}\left\|\frac{c_{i,t}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 - \frac{1}{2b_{i,t-1}}\left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 \\
&\qquad = \frac{\|\mathbf{u}_i\|^2}{2\gamma^2}\left(\frac{c_{i,t}^2}{b_{i,t}} - \frac{c_{i,t-1}^2}{b_{i,t-1}}\right) + \frac{\mathbf{u}_i^T\mathbf{w}_{i,t}}{\gamma}\left(\frac{c_{i,t-1}}{b_{i,t-1}} - \frac{c_{i,t}}{b_{i,t}}\right) \\
&\qquad + \frac{\|\mathbf{w}_{i,t}\|^2}{2}\left(\frac{1}{b_{i,t}} - \frac{1}{b_{i,t-1}}\right).
\end{aligned}$$

We know that by definition, $b_{i,t}$ is non decreasing over time such that the last term is always negative and can be waved. In addition, the definition of $c_{i,t}$ leads to

$$\frac{c_{i,t}}{b_{i,t}} = \frac{1}{2\beta\sqrt{1+N_{i,t}}} + 1$$

which combining with the fact that $N_{i,t}$ is nondecreasing over time as well, we get the inequality

$$\frac{c_{i,t}}{b_{i,t}} \leq \frac{c_{i,t-1}}{b_{i,t-1}}.$$

Combining these facts and using $Cauchy - Schwarz$ the inequality becomes,

$$\begin{aligned}\frac{1}{2b_{i,t}}\left\|\frac{c_{i,t}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 - \frac{1}{2b_{i,t-1}}\left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 \\ \leq \frac{\|\mathbf{u}_i\|^2}{2\gamma^2}\left(\frac{c_{i,t}^2}{b_{i,t}} - \frac{c_{i,t-1}^2}{b_{i,t-1}}\right) + \frac{\|\mathbf{u}_i\|\,\|\mathbf{w}_{i,t}\|}{\gamma}\left(\frac{c_{i,t-1}}{b_{i,t-1}} - \frac{c_{i,t}}{b_{i,t}}\right).\end{aligned} \tag{3.20}$$

Here, recall that on an update trial $y_{i,t}\mathbf{w}_{i,t-1}^T\mathbf{x}_{i,t} \leq 0$, we define $X_i = \max_t \|\mathbf{x}_{i,t}\|$ and use again the update rule and bound the weight vector norm with the number of updates ,

$$\begin{aligned}\|\mathbf{w}_{i,t}\|^2 &= \|\mathbf{w}_{i,t-1} + M_{i,t}Z_{i,t}y_{i,t}\mathbf{x}_{i,t}\|^2 \\ &= \|\mathbf{w}_{i,t-1}\|^2 + 2M_{i,t}Z_{i,t}y_{i,t}\mathbf{w}_{i,t-1}^T\mathbf{x}_{i,t} + M_{i,t}Z_{i,t}\|\mathbf{x}_{i,t}\|^2 \\ &\leq \|\mathbf{w}_{i,t-1}\|^2 + M_{i,t}Z_{i,t}\|\mathbf{x}_{i,t}\|^2 \\ &\leq \|\mathbf{w}_{i,t-1}\|^2 + M_{i,t}Z_{i,t}X_i^2\end{aligned}$$

Applying this inequality recursively, combined with the initial value $\mathbf{w}_{i,0} = 0$ leads us to

$$\|\mathbf{w}_{i,t}\| \leq X_i\sqrt{N_{i,t}} \tag{3.21}$$

which holds for any i and t. Combining the last inequality into Eq. (3.20) we get

$$\begin{aligned}\frac{1}{2b_{i,t}}\left\|\frac{c_{i,t}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 - \frac{1}{2b_{i,t-1}}\left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 \leq \\ \frac{\|\mathbf{u}_i\|^2}{2\gamma^2}\left(\frac{c_{i,t}^2}{b_{i,t}} - \frac{c_{i,t-1}^2}{b_{i,t-1}}\right) + \frac{\|\mathbf{u}_i\|\,X_i\sqrt{N_{i,t}}}{\gamma}\left(\frac{c_{i,t-1}}{b_{i,t-1}} - \frac{c_{i,t}}{b_{i,t}}\right).\end{aligned} \tag{3.22}$$

We will write now, a simpler bound on the last term. First we note that on t that $M_{i,t}Z_{i,t} = 1$, i.e. in the case of update, $N_{i,t} = N_{i,t-1} + 1$. Using the bound $\sqrt{1+x} - \sqrt{x} \leq \frac{1}{2\sqrt{x}}$, yields

$$\begin{aligned}
\sqrt{N_{i,t}}\left(\frac{c_{t-1}}{b_{t-1}} - \frac{c_t}{b_t}\right) &= \frac{\sqrt{N_{i,t}}}{2\beta}\left(\frac{1}{\sqrt{1+N_{i,t-1}}} - \frac{1}{\sqrt{1+N_{i,t}}}\right) \\
&= \frac{\sqrt{N_{i,t}}}{2\beta}\left(\frac{1}{\sqrt{N_{i,t}}} - \frac{1}{\sqrt{1+N_{i,t}}}\right) \\
&= \frac{1}{2\beta}\frac{\sqrt{1+N_{i,t}} - \sqrt{N_{i,t}}}{\sqrt{1+N_{i,t}}} \\
&\leq \frac{1}{4\beta}\frac{1}{\sqrt{N_{i,t}}\sqrt{1+N_{i,t}}} \\
&\leq \frac{1}{4\beta}\frac{1}{N_{i,t}}.
\end{aligned}$$

If $M_{i,t}Z_{i,t} = 0$, the left hand side becomes 0 because in such case, $c_{i,t} = c_{i,t-1}$ and $b_{i,t} = b_{i,t-1}$, which means that this bound holds for all i and t and we can write it as

$$\sqrt{N_{i,t}}\left(\frac{c_{t-1}}{b_{t-1}} - \frac{c_t}{b_t}\right) \leq \frac{M_{i,t}Z_{i,t}}{4\beta}\frac{1}{N_{i,t}}.$$

Plugging the last bound and Eq. (3.22) into Eq. (3.19) gives us the next inequality that holds for any $i, t, \gamma > 0$, and $u_i \in \mathbb{R}^d$,

$$\begin{aligned}
M_{i,t}Z_{i,t}\left(\frac{c_{i,t-1} + |\hat{p}_{i,t}| - \|\mathbf{x}_{i,t}\|^2/2}{b_{i,t-1}}\right) &\leq M_{i,t}Z_{i,t}\frac{c_{i,t-1}}{b_{i,t-1}}\frac{\ell_{\gamma,i,t}(\mathbf{u}_i)}{\gamma} \\
&+ \frac{1}{2b_{i,t-1}}\left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t-1}\right\|^2 - \frac{1}{2b_{i,t}}\left\|\frac{c_{i,t}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 \\
&+ \frac{\|\mathbf{u}_i\|^2}{2\gamma^2}\left(\frac{c_{i,t}^2}{b_{i,t}} - \frac{c_{i,t-1}^2}{b_{i,t-1}}\right) + \frac{\|\mathbf{u}_i\| X_i}{\gamma}\frac{M_{i,t}Z_{i,t}}{4\beta}\frac{1}{N_{i,t}}.
\end{aligned}$$

Recall the definition of $c_{i,t}$, we can write $c_{i,t-1} - \|\mathbf{x}_{i,t}\|^2/2 \geq b_{i,t-1}$ and the

ineqality gets the form

$$
\begin{aligned}
M_{i,t}Z_{i,t}\left(\frac{b_{i,t-1}+|\hat{p}_{i,t}|}{b_{i,t-1}}\right) &\leq M_{i,t}Z_{i,t}\frac{c_{i,t-1}}{b_{i,t-1}}\frac{\ell_{\gamma,i,t}(\mathbf{u}_i)}{\gamma} \\
&+ \frac{1}{2b_{i,t-1}}\left\|\frac{c_{i,t-1}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t-1}\right\|^2 - \frac{1}{2b_{i,t}}\left\|\frac{c_{i,t}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,t}\right\|^2 \\
&+ \frac{\|\mathbf{u}_i\|^2}{2\gamma^2}\left(\frac{c_{i,t}^2}{b_{i,t}} - \frac{c_{i,t-1}^2}{b_{i,t-1}}\right) + \frac{\|\mathbf{u}_i\|\,X_i}{\gamma}\frac{M_{i,t}Z_{i,t}}{4\beta}\frac{1}{N_{i,t}}.
\end{aligned}
$$

Summing both sides over $t = 1, \cdots, n$, and taking into consideration the initialization, $w_0 = 0$ we obtain,

$$
\begin{aligned}
\sum_{t=1}^{n} M_{i,t}Z_{i,t}\left(\frac{b_{i,t-1}+|\hat{p}_{i,t}|}{b_{i,t-1}}\right) &\leq \frac{1}{\gamma}\sum_{t=1}^{n} M_{i,t}Z_{i,t}\frac{c_{i,t-1}}{b_{i,t-1}}\ell_{\gamma,i,t}(\mathbf{u}_i) \\
&+ \frac{\|\mathbf{u}_i\|^2}{2\gamma^2}\frac{c_{i,n}^2}{b_{i,n}} - \frac{1}{2b_{i,n}}\left\|\frac{c_{i,n}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,n}\right\|^2 + \frac{\|\mathbf{u}_i\|\,X_i}{4\beta\gamma}\sum_{t=1}^{n}\frac{M_{i,t}Z_{i,t}}{N_{i,t}}.
\end{aligned} \tag{3.23}
$$

Now, we will bound the terms in the right hand side. The first term can be bounded as following,

$$
\begin{aligned}
\frac{1}{\gamma}\frac{c_{i,t-1}}{b_{i,t-1}}\ell_{\gamma,i,t}(\mathbf{u}_i) &= \frac{1}{\gamma}\left(\frac{1}{2\beta\sqrt{1+N_{i,t-1}}} + 1\right)\ell_{\gamma,i,t}(\mathbf{u}_i) \\
&\leq \frac{1}{2\gamma\beta\sqrt{1+N_{i,t-1}}}\left(\gamma + \|u_i\|X_i\right) + \frac{\ell_{\gamma,i,t}(\mathbf{u}_i)}{\gamma} \\
&= \frac{1}{2\beta\sqrt{1+N_{i,t-1}}}\left(1 + \frac{\|u_i\|X_i}{\gamma}\right) + \frac{\ell_{\gamma,i,t}(\mathbf{u}_i)}{\gamma}
\end{aligned}
$$

Here, we used the fact that $\ell_{\gamma,i,t}(\mathbf{u}_i) \leq \gamma + \|u_i\|X_i$. Now we proceed by bounding

the two middle terms of Eq. (3.23) such that

$$\begin{aligned}
\frac{\|\mathbf{u}_i\|^2}{2\gamma^2}\frac{c_{i,n}^2}{b_{i,n}} - \frac{1}{2b_{i,n}}\left\|\frac{c_{i,n}}{\gamma}\mathbf{u}_i - \mathbf{w}_{i,n}\right\|^2 &= \frac{c_{i,n}u_i^T\mathbf{w}_{i,n}}{b_{i,n}\gamma} - \frac{\|\mathbf{w}_{i,n}\|^2}{2b_{i,n}} \\
&\leq \frac{c_{i,n}u_i^T\mathbf{w}_{i,n}}{b_{i,n}\gamma} \\
&\leq \frac{c_{i,n}}{b_{i,n}}\frac{\|u_i\|\ \|\mathbf{w}_{i,n}\|}{\gamma} \\
&\leq \left(\frac{1}{2\beta\sqrt{1+N_{i,n}}}+1\right)\frac{\|u_i\|X_i\sqrt{N_{i,n}}}{\gamma},
\end{aligned}$$

where we used Eq. (3.21) in the last step.
Putting all together in Eq. (3.23) gives us

$$\begin{aligned}
&\sum_{t=1}^{n} M_{i,t}Z_{i,t}\left(\frac{b_{i,t-1}+|\hat{p}_{i,t}|}{b_{i,t-1}}\right) \\
&\qquad \leq \frac{1}{2\beta}\left(1+\frac{\|u_i\|X_i}{\gamma}\right)\sum_{t=1}^{n}\frac{M_{i,t}Z_{i,t}}{\sqrt{1+N_{i,t-1}}} + \frac{1}{\gamma}\sum_{t=1}^{n}M_{i,t}Z_{i,t}\ell_{\gamma,i,t}(\mathbf{u}_i) \\
&\qquad + \left(\frac{1}{2\beta\sqrt{1+N_{i,n}}}+1\right)\frac{\|u_i\|X_i\sqrt{N_{i,n}}}{\gamma} + \frac{\|\mathbf{u}_i\|\ X_i}{4\beta\gamma}\sum_{t=1}^{n}\frac{M_{i,t}Z_{i,t}}{N_{i,t}}.
\end{aligned}$$

The first and the last sums in the right hand side can get more elegant form. Recall $M_{i,t}Z_{i,t} = 1$ implies $N_{i,t} = N_{i,t-1} + 1$, we can write

$$\sum_{t=1}^{n}\frac{M_{i,t}Z_{i,t}}{\sqrt{1+N_{i,t-1}}} = \sum_{t=1}^{n}\frac{M_{i,t}Z_{i,t}}{\sqrt{N_{i,t}}} = \sum_{r=1}^{N_{i,n}}\frac{1}{\sqrt{r}} \leq 2\sqrt{N_{i,n}}.$$

Now, we apply the same bound on the second sum, but with more loose bound,

$$\sum_{t=1}^{n}\frac{M_{i,t}Z_{i,t}}{N_{i,t}} \leq \sum_{t=1}^{n}\frac{M_{i,t}Z_{i,t}}{\sqrt{N_{i,t}}} \leq 2\sqrt{N_{i,n}}.$$

Defining $R_i = (\|u_i\| X_i)/\gamma$, and combining the last three bounds together we get

$$\begin{aligned}
&\sum_{t=1}^{n} M_{i,t} Z_{i,t} \left(\frac{b_{i,t-1} + |\hat{p}_{i,t}|}{b_{i,t-1}} \right) \\
&\qquad \leq \frac{1}{\gamma} \sum_{t=1}^{n} M_{i,t} Z_{i,t} \ell_{\gamma,i,t}(\mathbf{u}_i) + \frac{1}{\beta} \left(1 + \frac{3R_i}{2} \right) \sqrt{N_{i,n}} \\
&\qquad + \frac{R_i}{2\beta} + R_i \sqrt{N_{i,n}}.
\end{aligned}$$

Now, we subtract non negative term from the left side, sum the inequality over all tasks and take the expectation.

$$\begin{aligned}
&\mathbb{E} \left[\sum_{i=1}^{K} \sum_{t=1}^{n} M_{i,t} Z_{i,t} \left(\frac{b_{i,t-1} + |\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}|}{b_{i,t-1}} \right) \right] \\
&\leq \frac{1}{\gamma} \mathbb{E} \left[\sum_{i=1}^{K} \sum_{t=1}^{n} M_{i,t} Z_{i,t} \ell_{\gamma,i,t}(\mathbf{u}_i) \right] + \mathbb{E} \left[\sum_{i=1}^{K} \left(R_i + \frac{2 + 3R_i}{2\beta} \right) \sqrt{N_{i,n}} \right] \\
&\quad + \mathbb{E} \left[\sum_{i=1}^{K} \frac{R_i}{2\beta} \right].
\end{aligned}$$

First we look at the left hand side of the inequality.

$$\sum_{i=1}^{K} \mathbb{E} \left[\sum_{t=1}^{n} M_{i,t} Z_{i,t} \left(\frac{b_{i,t-1} + |\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}|}{b_{i,t-1}} \right) \right] = \sum_{i=1}^{K} \mathbb{E} \left[\sum_{t=1}^{n} \frac{M_{i,t}}{D_t} \right]$$

The bound of D_t now is

$$D_t = \sum_{i=1}^{K} \frac{b_{i,t-1}}{b_{i,t-1} + |\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}|} \leq K$$

Since $M_{i,t} > 0 \forall i, t$, thus there exists $\delta_i \in \mathbb{R},\ 0 < \delta_i \leq K$ such that

$$\sum_{i=1}^{K} \mathbb{E} \left[\sum_{t=1}^{n} \frac{M_{i,t}}{D_t} \right] = \sum_{i=1}^{K} \frac{1}{\delta_i} \mathbb{E} \left[\sum_{t=1}^{n} M_{i,t} \right] \leq \frac{1}{\delta} \mathbb{E} \left[\sum_{i=1}^{K} \sum_{t=1}^{n} M_{i,t} \right]$$

where $\delta = max_i \delta_i$. Next we handle the middle term of the rhs.

$$\mathbb{E}\left[\sum_{i=1}^{K}\left(R_i + \frac{2+3R_i}{2\beta}\right)\sqrt{N_{i,n}}\right] \leq \mathbb{E}\left[\sum_{i=1}^{K}\left(R + \frac{2+3R}{2\beta}\right)\sqrt{N_{i,n}}\right]$$
$$= \left(R + \frac{2+3R}{2\beta}\right)\mathbb{E}\left[\sum_{i=1}^{K}\sqrt{N_{i,n}}\right],$$

Where $R = \max_i \left(\|u_i\| X_i\right)/\gamma$. Now, considering that $N_{i,n}$ is non-negative and exploiting the concavity of the square root function we can continue as following

$$\mathbb{E}\left[\sum_{i=1}^{K}\sqrt{N_{i,n}}\right] \leq \mathbb{E}\left[\sqrt{K\sum_{i=1}^{K}N_{i,n}}\right] = \sqrt{K}\mathbb{E}\left[\sqrt{\sum_{i=1}^{K}N_{i,n}}\right]$$
$$\leq \sqrt{K}\sqrt{\mathbb{E}\left[\sum_{i=1}^{K}N_{i,n}\right]}.$$

Here we used the bound for sum of squares, $\sum_{i=1}^{K}\sqrt{a_i} \leq \sqrt{K\sum_{i=1}^{K}a_i}$ for $a_i \in \mathbb{R}, a_i \geq 0$ combining with Jensen inequality. Moreover, we can continue, recall that by definition, $N_{i,n} = \sum_{t=1}^{n} Z_{i,t}M_{i,t}$ and $Z_{i,t}M_{i,t} \leq M_{i,t}$ such that

$$\sqrt{K}\sqrt{\mathbb{E}\left[\sum_{i=1}^{K}N_{i,n}\right]} \leq \sqrt{K}\sqrt{\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n}M_{i,t}\right]}.$$

Plugging the result in the inequality, we get

$$\frac{1}{\delta}\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n}M_{i,t}\right]$$
$$\leq \frac{1}{\gamma}\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n}M_{i,t}Z_{i,t}\ell_{\gamma,i,t}(\mathbf{u}_i)\right] + \mathbb{E}\left[\sum_{i=1}^{K}\left(R_i + \frac{2+3R_i}{2\beta}\right)\sqrt{N_{i,n}}\right] + \mathbb{E}\left[\sum_{i=1}^{K}\frac{R_i}{2\beta}\right]$$
$$\leq \frac{1}{\gamma}\bar{L}_{\gamma,n} + \left(R + \frac{2+3R}{2\beta}\right)\sqrt{K}\sqrt{\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n}M_{i,t}\right]} + \frac{KR}{2\beta}$$

which is quadratic inequality in $\sqrt{\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n}M_{i,t}\right]}$.

Solving this equation and gives us the bound for the cumulative mistakes

$$\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right] \leq \delta\left[\frac{\delta B^2}{2} + \frac{1}{\gamma}\bar{L}_{\gamma,n} + \frac{KR}{2\beta} + B\sqrt{\frac{\delta B^2}{4} + \frac{1}{\gamma}\bar{L}_{\gamma,n} + \frac{KR}{2\beta}}\right]$$

where $B = \left(R + \frac{2+3R}{2\beta}\right)$. ■

3.6 Second Order SHAMPO

The second order version of the perceptron algorithm, was proposed and analyzed by Cesa-Bianchi et al. [2005]. This algorithm was adopted to the online binary classification-settings from the ridge-regression (squared loss and Euclidean regularization) framework as described by Hoerl and Kennard [1970] and Vovk [1997], and further analyzed in Azoury and Warmuth [2001] and Forster and Warmuth [2002]. It was also shown that this variation has the effect of reducing the number of mistakes compare to the first order version. Cesa-Bianchi et al. [2006c] also adapted the second order perceptron algorithm to the selective sampling case, which was extended later by Crammer [2014].

In this algorithm, we store and update two model quantities per task instead of one, the matrix $A_{i,t} = I + \sum_t M_{i,t} Z_{i,t} \mathbf{x}_{i,t} \mathbf{x}_{i,t}^T$ and the vector $\mathbf{w}_{i,t} = \sum_t M_{i,t} Z_{i,t} \mathbf{x}_{i,t} y_{i,t}$. Whereas, the probability to issue a query on task i in time t is similar to the first order SHAMPO algorithm (Eq. (3.1)), the update that is done here in the case of a queried mistake, is the second order perceptron update. The pseudo code of the second order SHAMPO algorithm is shown in Alg. 3.4

Theorem 6 *If second order SHAMPO algorithm runs on K tasks with K parallel example pair sequences $(\mathbf{x}_{i,1}, y_{i,1}), \ldots (\mathbf{x}_{i,n}, y_{i,n}) \in \mathbb{R}^d \times \{-1, 1\}$, $i = 1, ..., K$ with input parameter $b > 0$, then for all $\gamma > 0$, all $\mathbf{u}_i \in \mathbb{R}^d$ and all $n \geq 1$, there exists $0 < \delta \leq K$, such that,*

$$\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right]$$
$$\leq \frac{\delta}{\gamma}\bar{L}_{\gamma,n}(\mathbf{u}_i) + \frac{\delta b}{2\gamma^2}\sum_{i=1}^{K}\mathbf{u}_i^T\mathbb{E}\left[A_{i,n}\right]\mathbf{u}_i + \frac{\delta}{2b}\sum_{i=1}^{K}\sum_{k=1}^{d}\mathbb{E}\left[\ln\left(1+\lambda_{i,k}\right)\right],$$

where $A_{i,n} = I + \sum_{t=1}^{n} M_{i,t} Z_{i,t} \mathbf{x}_{i,t} \mathbf{x}_{i,t}^T$ *,* $\lambda_{i,k}$ *is the* $k^t h$ *eigenvalue of the matrix* $A_{i,n}$ *and the expectation is over the random choices of the algorithm.*

Algorithm 3.4 Second order SHAMPO

Parameters: $b \in \mathbb{R} > 0$.
Initialize: $\mathbf{w}_{i,0} = \mathbf{0}$, $A_{i,0} = I$

for $t = 1, 2, ..., n$ **do**

1. Observe K instance vectors, $\mathbf{x}_{i,t}$, $(i = 1, \ldots, K)$.
2. Compute $\hat{p}_{i,t} = \mathbf{x}_{i,t}^T \left(A_{i,t-1} + \mathbf{x}_{i,t} \mathbf{x}_{i,t}^T \right)^{-1} \mathbf{w}_{i,t-1}$.
3. Predict K labels, $\hat{y}_{i,t} = \mathrm{sign}(\hat{p}_{i,t})$.
4. Draw problem J_t with the distribution:

$$\Pr\left[J_t = j\right] = \frac{\left(b + |\hat{p}_{j,t}| - \min_{m=1}^{K} |\hat{p}_{m,t}|\right)^{-1}}{D_t},$$
$$D_t = \sum_i \left(b + |\hat{p}_{i,t}| - \min_{m=1}^{K} |\hat{p}_{m,t}| \right)^{-1}.$$

   Where $m, i \in \{1, ..., K\}$ and D_t is the normalization factor.
5. Query the true label ,$y_{J_t,t} \in \{-1, 1\}$.
6. Set the indicator $M_{J_t,t} = 1$ iff $y_{J_t,t} \neq \hat{y}_{J_t,t}$.
7. Update with the second order perceptron rule:

$$\mathbf{w}_{J_t,t} = \mathbf{w}_{J_t,t-1} + M_{J_t,t}\, \mathbf{x}_{J_t,t} y_{J_t,t}$$
$$A_{J_t,t} = A_{J_t,t-1} + M_{J_t,t} \mathbf{x}_{J_t,t} \mathbf{x}_{J_t,t}^T$$
$$\mathbf{w}_{i,t} = \mathbf{w}_{i,t-1}, \quad A_{i,t} = A_{i,t-1} \;\; \forall i \neq J_t$$

end for
Output: $\mathbf{w}_{i,n}$ for $i = 1, \ldots, K$.

Proof: Define the regularized cumulative square loss of the updated rounds on the task i up to the round t by

$$\Phi_{i,t}(\mathbf{u}_i) = \frac{1}{2} \|\mathbf{u}_i\|^2 + \frac{1}{2} \sum_{s=1}^{t} Z_{i,t} M_{i,t} \left(y_{i,t} - \mathbf{u}_i^T \mathbf{x}_{i,t} \right)^2.$$

We now show that the algorithm incurs on each mistaken trial a square loss $(y_{i,t} - \hat{p}_{i,t})^2$

bounded by the difference $\inf_{\mathbf{u}_i} \Phi_{i,t+1}(\mathbf{u}_i) - \inf_{\mathbf{u}_i} \Phi_{i,t}(\mathbf{u}_i)$ plus a quadratic term involving $A_{i,t}^{-1}$. Then, we sum the inequality over mistaken rounds and bound the difference telescopes and the sum of the quadratic terms using known results. Then, the margin we use in the probabilistic analysis is obtained as cross-term when the square loss is expanded. Finally, we sum both sides over all tasks and apply the expectation using the correspondance probability.
It was proved by Forster [1999] in the linear regression case and adopted by Cesa-Bianchi et al. [2006c] and Crammer [2014] for the second order classification that of the single task that,

$$\begin{aligned}\frac{1}{2} Z_{i,t} M_{i,t} \left(y_{i,t} - \hat{p}_{i,t}\right)^2 = &\inf_{\mathbf{u}_i} \Phi_{i,t+1}(\mathbf{u}_i) - \inf_{\mathbf{u}_i} \Phi_{i,t}(\mathbf{u}_i) + \frac{Z_{i,t} M_{i,t}}{2} \mathbf{x}_{i,t}^T A_{i,t}^{-1} \mathbf{x}_{i,t} \\ &- \frac{Z_{i,t} M_{i,t}}{2} \mathbf{x}_{i,t}^T A_{i,t-1}^{-1} \mathbf{x}_{i,t} \hat{p}_{i,t}^2.\end{aligned}$$

We can drop now the last term which is nonnegative because $A_{i,t-1}$ is positive definite matrix and so that $A_{i,t-1}^{-1}$. Now, we sum up the equation over t and get

$$\begin{aligned}\sum_{t=1}^{n} \frac{Z_{i,t} M_{i,t}}{2} \left(y_{i,t} - \hat{p}_{i,t}\right)^2 &\leq \inf_{\mathbf{u}_i} \Phi_{i,n+1}(\mathbf{u}_i) - \inf_{\mathbf{u}_i} \Phi_{i,1}(\mathbf{u}_i) + \sum_{t=1}^{n} \frac{Z_{i,t} M_{i,t}}{2} \mathbf{x}_{i,t}^T A_{i,t}^{-1} \mathbf{x}_{i,t} \\ &\leq \Phi_{i,n+1}(\mathbf{u}_i) + \sum_{t=1}^{n} \frac{Z_{i,t} M_{i,t}}{2} \mathbf{x}_{i,t}^T A_{i,t}^{-1} \mathbf{x}_{i,t} \\ &\leq \frac{1}{2} \left\|\mathbf{u}_i\right\|^2 + \frac{1}{2} \sum_{t=1}^{n} Z_{i,t} M_{i,t} \left(y_{i,t} - \mathbf{u}_i^T \mathbf{x}_{i,t}\right)^2 + \sum_{t=1}^{n} \frac{Z_{i,t} M_{i,t}}{2} \mathbf{x}_{i,t}^T A_{i,t}^{-1} \mathbf{x}_{i,t}\end{aligned}$$

since $\inf_{\mathbf{u}_i} \Phi_{i,1}(\mathbf{u}_i) = 0$. We now expand the squares and the inequality becomes

$$\begin{aligned}\sum_{t=1}^{n} \frac{Z_{i,t} M_{i,t}}{2} \left(\hat{p}_{i,t}^2 - 2\hat{p}_{i,t} y_{i,t}\right) &\leq \frac{1}{2} \left\|\mathbf{u}_i\right\|^2 \\ + \frac{1}{2} \sum_{t=1}^{n} Z_{i,t} M_{i,t} \left(\mathbf{u}_i^T \mathbf{x}_{i,t}\right)^2 &+ \sum_{t=1}^{n} Z_{i,t} M_{i,t} \mathbf{u}_i^T \mathbf{x}_{i,t} y_{i,t} + \sum_{t=1}^{n} \frac{Z_{i,t} M_{i,t}}{2} \mathbf{x}_{i,t}^T A_{i,t}^{-1} \mathbf{x}_{i,t}.\end{aligned} \tag{3.24}$$

Now we handle the right hand side of the inequality one by one. First we start by

writing the two first terms in the form

$$\begin{aligned}\frac{1}{2}\left\|\mathbf{u}_i\right\|^2+\frac{1}{2}\sum_{t=1}^{n}Z_{i,t}M_{i,t}\left(\mathbf{u}_i^T\mathbf{x}_{i,t}\right)^2&=\frac{1}{2}\mathbf{u}_i^T\left(I+\frac{1}{2}\sum_{t=1}^{n}Z_{i,t}M_{i,t}\mathbf{x}_{i,t}\mathbf{x}_{i,t}^T\right)\mathbf{u}_i\\&=\frac{1}{2}\mathbf{u}_i^TA_{i,n}\mathbf{u}_i\end{aligned}\tag{3.25}$$

Denote the kth eigenvalue of the matrix $A_{i,n}$ by $1+\lambda_{i,k}$, we can bound the last term by

$$\begin{aligned}\frac{1}{2}\sum_{t=1}^{n}Z_{i,t}M_{i,t}\mathbf{x}_{i,t}^TA_{i,t}^{-1}\mathbf{x}_{i,t}&\leq\frac{1}{2}\sum_{t=1}^{n}\ln\left(\frac{\det A_{i,t}}{\det A_{i,t-1}}\right)\\&=\frac{1}{2}\ln\left(\frac{\det A_{i,n}}{\det A_{i,0}}\right)=\frac{1}{2}\ln\left(\det A_{i,n}\right)=\frac{1}{2}\sum_{k=1}^{d}\ln\left(1+\lambda_{i,k}\right),\end{aligned}\tag{3.26}$$

as proved by Forster [1999].
Plugging Eq. (3.25) and Eq. (3.26) back into Eq. (3.24) and dropping the positive term from the left hand side, recall that on round when there is a mistake ($M_{i,t}=1$), $\hat{p}_{i,t}y_{i,t}\leq 0$, we obtain

$$\sum_{t=1}^{n}Z_{i,t}M_{i,t}\left(\left|\hat{p}_{i,t}\right|+\mathbf{u}_i^T\mathbf{x}_{i,t}y_{i,t}\right)\leq\frac{1}{2}\mathbf{u}_i^TA_{i,n}\mathbf{u}_i+\frac{1}{2}\sum_{k=1}^{d}\ln\left(1+\lambda_{i,k}\right).$$

By the definition of hinge loss, $\gamma-\ell_{\gamma,i,t}(\mathbf{u}_i)\leq y_{i,t}\mathbf{u}_i^T\mathbf{x}_{i,t}$. Replacing $\mathbf{u}_i$ vectors with their scaling $\frac{b}{\gamma}\mathbf{u}_i$, yields

$$\begin{aligned}&\sum_{t=1}^{n}Z_{i,t}M_{i,t}\left(\left|\hat{p}_{i,t}\right|+b\right)\\&\qquad\leq\frac{b}{\gamma}\sum_{t=1}^{n}Z_{i,t}M_{i,t}\ell_{\gamma,i,t}(\mathbf{u}_i)+\frac{b^2}{2\gamma^2}\mathbf{u}_i^TA_{i,n}\mathbf{u}_i+\frac{1}{2}\sum_{k=1}^{d}\ln\left(1+\lambda_{i,k}\right).\end{aligned}$$

Now, we subtract a non negative quantity $\sum_{t=1}^{n}M_{i,t}Z_{i,t}\min_j\left|\hat{p}_{j,t}\right|$ from the left

hand side and get,

$$\sum_{t=1}^{n} Z_{i,t} M_{i,t} \left(|\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}| + b \right)$$
$$\leq \frac{b}{\gamma} \sum_{t=1}^{n} Z_{i,t} M_{i,t} \ell_{\gamma,i,t}(\mathbf{u}_i) + \frac{b^2}{2\gamma^2} \mathbf{u}_i^T A_{i,n} \mathbf{u}_i + \frac{1}{2} \sum_{k=1}^{d} \ln\left(1 + \lambda_{i,k}\right).$$

At this point, we take the expectation on both inequality sides. First we start from the left side. As in Eq. (3.14) of the first order proof,

$$\mathbb{E}\left[\sum_{t=1}^{n} Z_{i,t} M_{i,t} \left(|\hat{p}_{i,t}| - \min_j |\hat{p}_{j,t}| + b \right) \right] = \mathbb{E}\left[\sum_{t=1}^{n} \frac{M_{i,t}}{D_t} \right].$$

Taking the expectation from the right hand side the equation becomes

$$\mathbb{E}\left[\sum_{t=1}^{n} \frac{M_{i,t}}{D_t} \right] \leq \frac{b}{\gamma} \bar{L}_{\gamma,i,n}(\mathbf{u}_i) + \frac{b^2}{2\gamma^2} \mathbf{u}_i^T \mathbb{E}\left[A_{i,n}\right] \mathbf{u}_i + \frac{1}{2} \sum_{k=1}^{d} \mathbb{E}\left[\ln\left(1 + \lambda_{i,k}\right)\right].$$

We use now Eq. (3.9) and get

$$\frac{b}{\delta_i} \mathbb{E}\left[\sum_{t=1}^{n} M_{i,t} \right] \leq \frac{b}{\gamma} \bar{L}_{\gamma,i,n}(\mathbf{u}_i) + \frac{b^2}{2\gamma^2} \mathbf{u}_i^T \mathbb{E}\left[A_{i,n}\right] \mathbf{u}_i + \frac{1}{2} \sum_{k=1}^{d} \mathbb{E}\left[\ln\left(1 + \lambda_{i,k}\right)\right]$$

We conclude the proof by summing up the last inequality over all K tasks and setting $\delta = \max \delta_i$, such that

$$\frac{1}{\delta} \mathbb{E}\left[\sum_{i=1}^{K} \sum_{t=1}^{n} M_{i,t} \right]$$
$$\leq \frac{1}{\gamma} \bar{L}_{\gamma,n}(\mathbf{u}_i) + \frac{b}{2\gamma^2} \sum_{i=1}^{K} \mathbf{u}_i^T \mathbb{E}\left[A_{i,n}\right] \mathbf{u}_i + \frac{1}{2b} \sum_{i=1}^{K} \sum_{k=1}^{d} \mathbb{E}\left[\ln\left(1 + \lambda_{i,k}\right)\right].$$

■

3.7 Second Order Aggressive SHAMPO

Algorithm 3.5 Second order aggressive SHAMPO.

Parameters: $b \in \mathbb{R} > 0$.
Initialize: $\mathbf{w}_{i,0} = \mathbf{0}$, $A_0 = I$

for $t = 1, 2, \ldots, n$ **do**

1. Observe K instance vectors, $\mathbf{x}_{i,t}$, $(i = 1, \ldots, K)$.
2. Compute $\hat{p}_{i,t} = \mathbf{x}_{i,t}^T \left(A_{i,t-1} + \mathbf{x}_{i,t}\mathbf{x}_{i,t}^T \right)^{-1} \mathbf{w}_{i,t-1}$.
3. Predict K labels, $\hat{y}_{i,t} = \text{sign}(\hat{p}_{i,t})$.
4. Compute $r_{i,t} = \mathbf{x}_{i,t}^T A_{i,t-1}^{-1} \mathbf{x}_{i,t}$.
5. Draw problem J_t with the distribution:

$$\Pr\left[J_t = j\right] = \frac{1}{D_t} \frac{1}{b + \left(\Theta\left(\left|\hat{p}_{j,t}\right|, r_{j,t}\right)\right)_+},$$
$$D_t = \sum_i \left(b + \left(\Theta\left(\left|\hat{p}_{i,t}\right|, r_{i,t}\right)\right)_+\right)^{-1}.$$

   Where $m, i \in \{1, \ldots, K\}$ and D_t is the normalization factor.

6. Set $Z_{J,t} = 1,\ Z_{i,t} = 0,\ \forall i \neq J$.
7. **if** $\Theta\left(\left|\hat{p}_{J_t,t}\right|, r_{i,t}\right) \geq 0$ **then**
8. **if** $y_{J_t,t} \neq \hat{y}_{J_t,t}$ **then**
9. $U_{J_t,t} = 1$
10. **else**
11. $U_{J_t,t} = 0$
12. **end if**
13. **else**
14. set $U_{J_t,t} = 1$
15. **end if**
16. Update:

$$\begin{aligned} \mathbf{w}_{J_t,t} &= \mathbf{w}_{J_t,t-1} + U_{J_t,t}\, \mathbf{x}_{J_t,t} y_{J_t,t} \\ A_{J_t,t} &= A_{J_t,t-1} + U_{J_t,t} \mathbf{x}_{J_t,t} \mathbf{x}_{J_t,t}^T \end{aligned} \tag{3.27}$$

end for
Output: $\mathbf{w}_{i,n}$ for $i = 1, \ldots, K$.

All the algorithms that were shown so far, estimated the uncertainty of the algorithm prediction using the low margin rule. Another approach of estimating the certainty of prediction could be comparing the input example with all the previous labeled examples. If the algorithm already received similar examples before, and requested their labels, probably the prediction that is based on those similar examples, can be considered with high confidence. Crammer [2014] used the same approach in the selective sampling setting, by introducing a distribution that depends on both margin and this uncertainty measure. An aggressive update was applied as well, based on the same measure.

We adapt this approach to our setting of multitask learning sharing a single annotator. First, we introduce one more quantity that is the projection of the vector $\mathbf{x}_{i,t}$ on the inverse of the covariance-like matrix $A_{i,t-1}$

$$r_{i,t} = \mathbf{x}_{i,t}^T A_{i,t-1}^{-1} \mathbf{x}_{i,t}$$

We can think about this quantity as an uncertainty measure. Where large values of $r_{i,t}$ represent high uncertainty, whereas small value of the same quantity are related to low uncertainty, since the algorithm already received labeled examples that are close to $\mathbf{x}_{i,t}$. In the edge case, for $\|\mathbf{x}_{i,t}\| \leq 1$, when the example $\mathbf{x}_{i,t}$ is orthogonal to all previous queried examples that came from the same task, $r_{i,t} = 1$.
We define here the function

$$\Theta\left(|\hat{p}_{i,t}|, r_{i,t}\right) = \left(1 + r_{i,t}\right) \hat{p}_{i,t}^2 + 2\left|\hat{p}_{i,t}\right| - \frac{r_{i,t}}{1 + r_{i,t}}. \tag{3.28}$$

which depends on both certainty measurements: $|\hat{p}_{i,t}|$ and $r_{i,t}$, and introduce the new probability over the tasks

$$\Pr\left[J_t = j\right] = \frac{1}{D_t} \frac{1}{b + \left(\Theta\left(|\hat{p}_{j,t}|, r_{j,t}\right)\right)_+} \tag{3.29}$$

where D_t is the normalization factor

$$D_t = \sum_i \left(b + \left(\Theta\left(|\hat{p}_{i,t}|, r_{i,t}\right)\right)_+\right)^{-1}.$$

One can see that even though $\Theta\left(|\hat{p}_{i,t}|, r_{i,t}\right)$ function can be negative, we consider only its positive values, so we always get a positive values of $\Pr\left[J_t = j\right]$ which make it a distribution. At each round we draw a task J_t using this distribution.

After we draw a task to query on, we decide if a model update should be done based on two quantities: the function $\Theta\left(|\hat{p}_{J_t,t}|, r_{i,t}\right)$ and the mistake indicator $M_{J_t,t}$. If $\Theta\left(|\hat{p}_{i,t}|, r_{J_t,t}\right) < 0$, than an aggressive update is been made, independent on the prediction value ,else, we update only when there is a prediction mistake (i.e. $M_{J_t,t} = 1$).

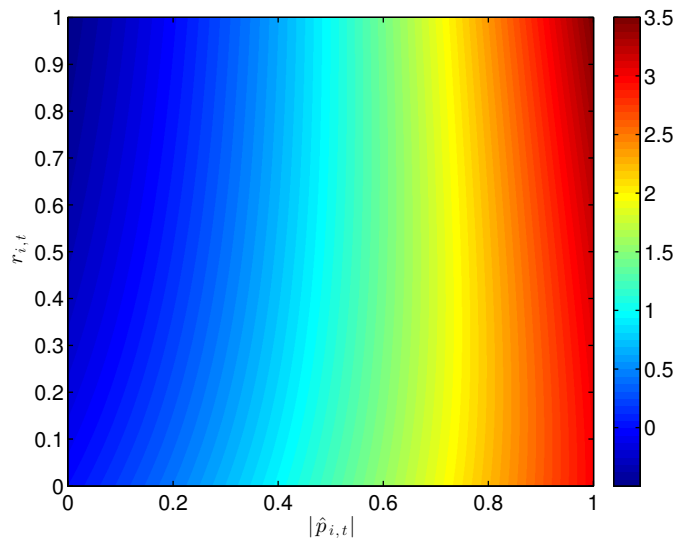


Figure 3.3: $\Theta\left(|\hat{p}_{i,t}|, r_{i,t}\right)$ vs. $|\hat{p}_{i,t}|$ and $r_{i,t}$

To understand this update rule, we examine the behavior of $\Theta\left(|\hat{p}_{i,t}|, r_{i,t}\right)$ function. This function, is quadratic in $|\hat{p}_{J_t,t}|$, solving the inequality $\Theta\left(|\hat{p}_{i,t}|, r_{J_t,t}\right) < 0$ for $|\hat{p}_{J_t,t}|$ leads to the following bound on the margin,

$$|\hat{p}_{i,t}| \leq \frac{-1 + \sqrt{1 + r_{i,t}}}{1 + r_{i,t}}.$$

This dynamic aggressive update threshold is actually upper bounded by a constant, since when $r_{i,t}$ is maximal, ($r_{i,t} = 1$), this aggressive threshold gets its highest value, $\frac{\sqrt{2}-1}{2} \approx 0.2$. Fig. 3.3 shows the value of $\Theta\left(|\hat{p}_{i,t}|, r_{J_t,t}\right)$ vs. $|\hat{p}_{i,t}|$ and $r_{i,t}$. The upper left area (deep blue) is the aggressive update domain which happens when the margin is small. The aggressiveness threshold depends on $r_{i,t}$ while high values gives us higher threshold since it is pointing on a high uncertainty.

We don't have a full analysis on the second order aggressive algorithm in the

sense of the mistake bounds here, yet, it was well analyzed in the selective sampling setting and shown remarkable empirical improvement over the rest of the algorithms in our experiments.

Chapter 4

From Multi-task to Contextual Bandits

Although our algorithm is designed for many binary-classification tasks that can be independent, it can also be applied in two settings of contextual bandits, when decoupling exploration and exploitation is allowed as in Yu and Mannor [2009], Avner et al. [2012]. The problem of this setting is predicting a label $\hat{Y}_t \in \{1, \ldots, C\}$ given an input $\mathbf{x}_t$. As before, the algorithm works in rounds. On round t the algorithm receives an input $\mathbf{x}_t$ and outputs multicalss label $\hat{Y}_t \in \{1, \ldots, C\}$. Then, it queries for some information about the label via a single binary "yes-no" question, and uses the feedback to update its model. We consider here two forms of binary questions. For generality, we consider here the bounds for the first order aggressive SHAMPO with prior algorithm. Clearly, bounds for the other algorithms can be derived as well in the same way.

4.1 One-vs-Rest

The first setting is termed *one-vs-rest*. The algorithm asks if the true label is some label $\bar{Y}_t \in \{1, \ldots, C\}$, possibly not the predicted label, i.e. it may be the case that $\bar{Y}_t \neq \hat{Y}_t$. Given the response whether $\bar{Y}_t$ is the true label Y_t, the algorithm updates its models. The reduction we perform is by introducing K tasks, one per class. The problem of the learning algorithm for task i is to decide whether the true label is class i or not. Given the output of all the binary classifiers, the algorithm generates a single multi-class prediction to be the single label for which the output

of the corresponding binary classifier is positive. If such class does not exist, or there are more than one, a random prediction is used, i.e., given an input $\mathbf{x}_t$ we define $\hat{Y}_t = \arg\max_i \hat{y}_{i,t}$, where ties are broken arbitrarily. The label to be queried is $\bar{Y}_t = J_t$, i.e. the problem index that SHAMPO is querying. We analyze the performance of this reduction as a multiclass prediction algorithm.

Corollary 7 *Assume the first order aggressive SHAMPO with prior algorithm is executed with $K = C$ one-vs-rest problems, on a sequence $(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n) \in \mathbb{R}^d \times \{1, ..., C\}$, and input parameter $b > 0$, $0 \leq \lambda \leq b/2$ and prior $1 \leq a_i \ \forall i$. Then for all $\gamma > 0$ and all $\mathbf{u}_i \in \mathbb{R}^d$, there exists $0 < \delta \leq \sum_{i=1}^{C} a_i$ such that the expected number of multi-class errors is bounded as follows*

$$\mathbb{E}\left[\sum_t [\![Y_t \neq \hat{Y}_t]\!]\right]$$
$$\leq \frac{\delta}{\gamma}\left[\left(1 + \frac{X^2}{2b}\right)\bar{L}_{\gamma,n} + \frac{\left(2b + X^2\right)^2 U^2}{8\gamma b}\right] + \left(2\frac{\lambda}{b} - 1\right)\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} a_i G_{i,t}\right] ,$$

where $[\![I]\!] = 1$ if the predicate I is true, and zero otherwise.

Proof: The corollary follows directly from Theorem 4 by noting that,

$$[\![Y_t \neq \hat{Y}_t]\!] \leq \sum_{i=1}^{K} M_{i,t}.$$

Meaning, there is a multiclass mistake if there is at least one prediction mistake of one of the one-vs-rest tasks, since when all the binary prediction are correct, the multiclass prediction is correct as well. ■

The closest setting is contextual bandits, yet we allow decoupling of exploration and exploitation. Ignoring this decoupling, the Banditron algorithm Kakade et al. [2008] is the closest to ours, with a regret of $O(T^{2/3})$. Hazan et al Hazan and Kale [2011] proposed an algorithm with $O(\sqrt{T})$ regret but designed for the $\log$ loss, with coefficient that may be very large, and another Crammer and Gentile [2013] algorithm has $O(\sqrt{T})$ regret with respect to prediction mistakes, yet they assumed stochastic labeling, rather than adversarial.

4.2 One-vs-One

In the second setting, termed by *one-vs-one*, the algorithm picks two labels $\bar{Y}_t^+, \bar{Y}_t^- \in \{1 \dots C\}$, possibly both not the predicted label. The feedback for the learner is three-fold: it is $y_{J_t,t} = +1$ if the first alternative is the correct label, $\bar{Y}_t^+ = Y_t$, $y_{J_t,t} = -1$ if the second alternative is the correct label, $\bar{Y}_t^- = Y_t$, and it is $y_{J_t,t} = 0$ otherwise (in this case there is no error and we set $M_{J_t,t} = 0$). The reduction we perform is by introducing $K = \binom{C}{2}$ tasks, one per pair of classes. The goal of the learning algorithm for a task indexed with two labels (y_1, y_2) is to decide which one is the correct label, given it is one of the two. Given the output of all (binary) classifiers the algorithm generates a single multi-class prediction using a tournament in a round-robin approach as in Fürnkranz [2002]. If there is no clear winner, a random prediction is used. We now analyze the performance of this reduction as a multiclass prediction algorithm.

Corollary 8 *Assume the first order aggressive SHAMPO with prior algorithm is executed, with $K = \binom{C}{2}$ one-vs-one problems, on a sequence $(\mathbf{x}_1, Y_1), \dots, (\mathbf{x}_n, Y_n) \in \mathbb{R}^d \times \{1, \dots, C\}$, and input parameter $b > 0$, $0 \le \lambda \le b/2$ and prior $1 \le a_i \ \forall i$. Then for all $\gamma > 0$ and all $\mathbf{u}_i \in \mathbb{R}^d$, there exists $0 < \delta \le \sum_{i=1}^{\binom{C}{2}} a_i$ such that the expected number of multi-class errors can be bounded as follows*

$$\mathbb{E}\left[\sum_t [\![Y_t \neq \hat{Y}_t]\!]\right] \le \frac{2}{(\binom{C}{2} - 1)/2 + 1} \times$$
$$\left\{\frac{\delta}{\gamma}\left[\left(1 + \frac{X^2}{2b}\right)\bar{L}_{\gamma,n} + \frac{\left(2b + X^2\right)^2 U^2}{8\gamma b}\right] + \left(2\frac{\lambda}{b} - 1\right)\mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} a_i G_{i,t}\right]\right\}$$

Proof: The corollary follows directly from Theorem 4 by noting that the multi-class error is bounded by the multitask errors with a constant factor as following,

$$[\![Y_t \neq \hat{Y}_t]\!] \le \frac{2}{(\binom{C}{2} - 1)/2 + 1} \sum_{i=1}^{\binom{C}{2}} M_{i,t}.$$

The last fact is derived from Allwein et al. [2001] analysis of the training error with the corresponding hamming distance of the ECOC matrix for the One-vs-One setting $\rho = (\binom{C}{2} - 1)/2 + 1$ combined with the fact that the ECOC can't correct $\rho/2$ errors as in Dietterich and Bakiri [1995]. As a result, we get

$[\![Y_t \neq \hat{Y}_t]\!] \leq \frac{2}{\rho} \sum_{i=1}^{K} M_{i,t}$. Plugging ρ and K conclude the proof. ■

Note, that the bound is essentially independent of C as the coefficient in the bound is upper bounded by 4.

We conclude this section with two algorithmic modifications, we employed in this setting. Currently, when the feedback is zero, there is no update of the weights, because there are no errors. This causes the algorithm to effectively ignore such examples, as in these cases the algorithm is not modifying any model, furthermore, if such example is repeated, a problem with possibly "0" feedback may be queried again. We fix this issue with one of two modifications: In the first one, if the feedback is zero, we modify the model to reduce the chance that the chosen problem, J_t, would be chosen again for the same input (i.e. not to make the same wrong-choice of choosing irrelevant problem again). To this end, we modify the weights a bit, to increase the confidence (absolute margin) of the model for the same input, and replace the update rule in Eq. (3.3) with,

$$\mathbf{w}_{J_t,t} = \mathbf{w}_{J_t,t-1} + [\![y_{J_t,t} \neq 0]\!]\, y_{J_t,t}\, \mathbf{x}_{J_t,t} + [\![y_{J_t,t} = 0]\!] \eta \hat{y}_{J_t,t} \mathbf{x}_{J_t,t} \ ,$$

for some small $\eta > 0$. In other words, if there is a possible error (i.e. $y_{J_t,t} \neq 0$) the update follows the Perceptron's rule. Otherwise, the weights are updated such that the absolute margin will increase, as

$$\begin{aligned} |\mathbf{w}_{J_t,t}^\top \mathbf{x}_{J_t,t}| &= \Big(\mathbf{w}_{J_t,t-1} + \eta \hat{y}_{J_t,t} \mathbf{x}_{J_t,t})^\top \mathbf{x}_{J_t,t}\Big) \\ &= |\mathbf{w}_{J_t,t-1}^\top \mathbf{x}_{J_t,t} + \eta \mathrm{sign}(\mathbf{w}_{J_t,t-1}^\top \mathbf{x}_{J_t,t}) \|\mathbf{x}_{J_t,t}\|^2| \\ &= |\mathbf{w}_{J_t,t-1}^\top \mathbf{x}_{J_t,t}| + \eta \|\mathbf{x}_{J_t,t}\|^2 > |\mathbf{w}_{J_t,t-1}^\top \mathbf{x}_{J_t,t}|. \end{aligned}$$

We call this method *one-vs-one-weak*, as it performs weak updates for zero feedback. The second alternative is not to allow 0 value feedback, and if this is the case, to set the label to be either $+1$ or -1, randomly. We call this method *one-vs-one-random*.

Chapter 5

Experiments

We evaluated the SHAMPO algorithm using four different datasets: USPS, MNIST (both OCR), Vocal Joystick (VJ, vowel recognition) and NLP dataset contains document classification and sentiment classification. The USPS dataset, contains $7,291$ training examples and $2,007$ test examples, each one of them is a 16×16 pixels gray-scale images converted to a 256 dimensional vector. The MNIST dataset with 28×28 gray-scale images, contains $60,000$ examples in the training set and $10,000$ in the testing set. In both cases there are 10 possible labels, the 0 to 9 digits. The VJ project was built to enable individuals with motor impairments to use vocal parameters to control objects on a computer screen (buttons, sliders, etc.) and ultimately electro-mechanical instruments (e.g., robotic arms, wireless home automation devices). The VJ tasks is to predict a vowel from eight possible vowels. Each example is a frame of spoken value described with 13 MFCC coefficients transformed into 26 features. There are $572,911$ training examples and $236,680$ test examples. In order to allow comparing the the margin between examples, we scaled the example vectors by normalize the vectors. We tried to scale the vectors in to a unit ball, but normalization gives the best performance. We also added a constant entry for each one of the vectors to represent the bias element. We created binary tasks from these multi-class datasets using two reductions: One-vs-Rest setting and One-vs-One setting. For example, in both USPS and MNIST there are 10 binary one-vs-rest tasks and 45 binary one-vs-one tasks. The NLP document classification is one-vs-one binary classification include of spam filtering, news items and news-group classification, sentiment classification, and product domain categorization. A total of 36 binary prediction tasks over all, with a total of $252,609$ examples, and input dimension varying between $8,768$ and $1,447,866$. Details of

the individual binary tasks can be found elsewhere Crammer et al. [2012]. Since the NLP dataset doesn't have a specific test set, we evaluated the algorithm performance using 10 folds cross validations. This yielded seven collections (USPS, MNIST and VJ; each as one-vs-rest or one-vs-one) and document classification.

5.1 Hard and Easy Tasks Tradeoff

First, we wanted to show that indeed, the SHMAPO algorithms are optimal for different collections of hard and easy tasks, and also for tasks from different domains and dimensions. For that reason, we created an eighth collection, named MIXED, which consists of 40 tasks: 10 random tasks from each one of the four basic datasets (one-vs-one versions). Then, from each one of the eight dataset collections we generated between 6 to 10 combinations (or problems), each problem was created by sampling between 2 and 8 tasks which yielded a total of 64 multitask problems. We tried to diversify problems difficulty by including both hard and easy binary classification problems. For example, from the VJ one-vs-one dataset we generated 10 multitask problems with up to 5 tasks each, and different number of hard and easy tasks : 1 (easy task) + 1 (hard task),1+3, 1+5, 1+7, 3+1, 3+3, 3+5, 5+1, 5+3, 7+1. The hardness of a binary problem is evaluated by the number of mistakes the Perceptron algorithm performs on each task, where for each multitask problem, we sample uniformly from each group of tasks (hard and easy). The exact details on the sub data sets collection are collected in Table 5.1. The max tasks columns indicates the size of the largest sub multitask collection for a certain dataset. The number of easy and hard group number is the size of the group of tasks from which we sample the hard or easy task. The last column is the number of multitask problems collections that were generated for the dataset (which sums up to 64) . For example, from the data set VJ one-vs-Rest we generated 6 multitask problems, with at most 5 tasks for a single multitask problem, when the easy tasks were sampled from the first 4 easier tasks and the harder tasks were sampled from the next 4 tasks, which are harder. It is important to point out that since we sample the data sets (for some datasets, we don't count the hardest tasks), we should not compare the results of this experiment with the results of experiments on the whole dataset tasks.

We evaluated two baselines in addition to our algorithm. Algorithm *uniform* picks a random task to be queried and updated (corresponding to $b \to \infty$), *exploit* which picks the tasks with the lowest absolute margin (i.e. the "hardest instance"),

Table 5.1: The data subset collections details

Dataset	Max Tasks	Easy group #	Hard group #	Collections
VJ 1 vs 1	8	10	10	10
VJ 1 vs Rest	5	4	4	6
USPS 1 vs 1	8	20	20	10
USPS 1 vs Rest	6	5	5	6
MNIST 1 vs 1	8	10	10	10
MNIST 1 vs Rest	6	5	5	6
NLP documents	6	8	8	6
MIXED	8	10	10	10

this combination corresponds to $b \approx 0$ of SHAMPO. We tried for SHAMPO 13 values for b, equally spaced on a logarithmic scale between 10^{-7} and 10^{5}. All algorithms made a single pass over the training data. We ran two versions of our First Order algorithm: plain version, without aggressiveness (updates on mistakes only, $\lambda = 0$) and an Aggressive version $\lambda = b/2$ (we tried lower values of λ as in the bound, but we found that $\lambda = b/2$ gives better results), both with uniform prior ($a_i = 1$). We used separate training set and a test set, to build a model and evaluate it. We repeated this procedure 50 times and computed mean test error of all runs on all tasks for each dataset.

The results are evaluated using 2 quantities. First, the average test error (over all the dataset combinations) and the average score. For each combination we assigned a score of 1 to the algorithm with the lowest test error, and a score of 2, to the second best, and all the way up to a score of 6 to the algorithm with the highest test error.

Table 5.2: Test errors percentage . Scores are shown in parenthesis.

	Aggressive $\lambda = b/2$			**Plain**		
Dataset	*exploit*	*SHAMPO*	*uniform*	*exploit*	*SHAMPO*	*uniform*
VJ 1 vs 1	5.22 (2.9)	**4.57 (1.1)**	5.67 (3.9)	5.21 (2.7)	6.93 (4.6)	6.26 (5.8)
VJ 1 vs Rest	13.26 (3.5)	**11.73 (1.2)**	12.43 (2.5)	13.11 (3.0)	14.17 (5.0)	14.71 (5.8)
USPS 1 vs 1	3.31 (2.5)	**2.73 (1.0)**	19.29 (6.0)	3.37 (2.5)	4.83 (4.0)	5.33 (5,0)
USPS 1 vs Rest	5.45 (2.8)	**4.93 (1.2)**	10.12 (6.0)	5.31 (2.0)	6.51 (4.0)	7.06 (5.0)
MNIST 1 vs 1	1.08 (2.3)	**0.75 (1.0)**	5.9 (6.0)	1.2 (2.7)	1.69 (4.1)	1.94 (4.9)
MNIST 1 vs Rest	4.74 (2.8)	**3.88 (1.0)**	10.01 (6.0)	4.44 (2.8)	5.4 (3.8)	6.1 (5.0)
NLP documents	19.43 (2.3)	**16.5 (1.0)**	23.21 (5.0)	19.46 (2.7)	21.54 (4.7)	21.74 (5.3)
MIXED	2.75 (2.4)	**2.06 (1.0)**	13.59 (6.0)	2.78 (2.6)	4.2 (4.3)	4.45 (4.7)
Mean score	(2.7)	**(1.1)**	(5.2)	(2.6)	(4.3)	(5.2)

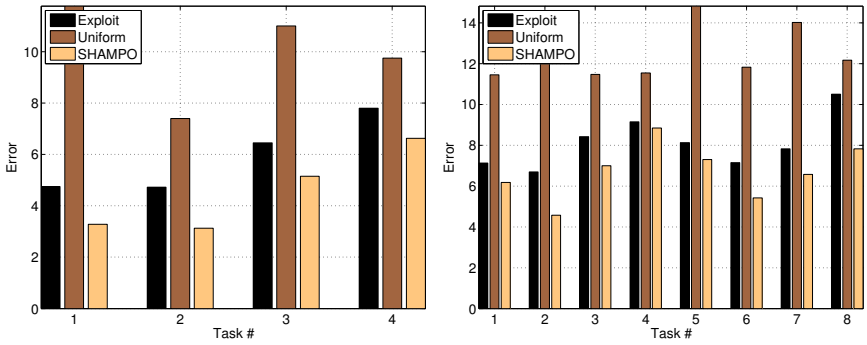
Results are summarized in Table 5.2. In general, *exploit* is better than *uniform* (for majority of the datasets) and aggressive algorithm is better than non-aggressive one. Aggressive SHAMPO yields the best results both evaluated as average (over tasks per combination and over combinations). Remarkably, even in the mixed dataset (where tasks are of different nature: images, audio and documents), the aggressive SHAPO improves over uniform (4.45% error) and the aggressive-exploit baseline (2.75%), and achieves a mean test error of 2.06%.

Indeed, Table 5.2 shows that aggressive SHAMPO outperforms other alternatives. Yet, we claim that a good prior may improve results. We compute prior over the 45 USPS one-vs-one tasks and 10 USPS one-vs-rest, by running the perceptron algorithm on $1,000$ examples and computing the number of mistakes. We set the prior to be proportional to this number. We then reran aggressive SHAMPO with prior, comparing it to aggressive SHAMPO with no prior (i.e. $a_i = 1$). The results are summarized in Table 5.3. The prior improves performance in the USPS 1 vs 1 collection and USPS 1 vs Rest, when evaluated using score-rank, on averaged it is slightly worse than aggressive SHAMPO with not prior. Aggressive SHAMO with prior achieves average error of 1.47 (vs. 2.73 with no prior) on 1-vs-1 USPS and 4.97 (vs 4.93) on one-vs-rest USPS, with score rank of 1.0 (vs 2.9) and 1.7 (vs 2.0) respectively. However, Fig. 5.3(c) and Fig. 5.3(d) shows the test error of the non aggressive SHAMPO algorithm with prior for all values of b we evaluated. The prior version results does not show a remarkable improvement over the plain SHAMPO algorithm. Again, we chose here a specific prior generator system, however, another method or another prior knowledge may lead to better results.

Table 5.3: Test errors percentage . Scores are shown in parenthesis.

	Aggressive $\lambda = b/2$			**Aggressive $\lambda = b/2$ with prior**		
Dataset	*exploit*	*SHAMPO*	*uniform*	*exploit*	*SHAMPO*	*uniform*
USPS 1 vs 1	3.31 (3.9)	2.73 (2.9)	19.29 (5.6)	1.92 (2.2)	**1.47 (1.0)**	17.66 (5.4)
USPS 1 vs Rest	5.45 (3.5)	**4.93** (2.0)	10.12 (5.7)	5.23 (2.8)	4.97 **(1.7)**	9.64 (5.3)

Fig. 5.1(a) and Fig. 5.1(b) show the test error of the three algorithms on two of document classification combinations, with four and eight tasks. Three algorithms are evaluated: uniform, exploit, and aggressive SHAMPO with $\lambda = b/2$. Clearly, not only SHAMPO performs better, but it does so on each task individually. (Our analysis above bounds the total number of mistakes over all tasks.)



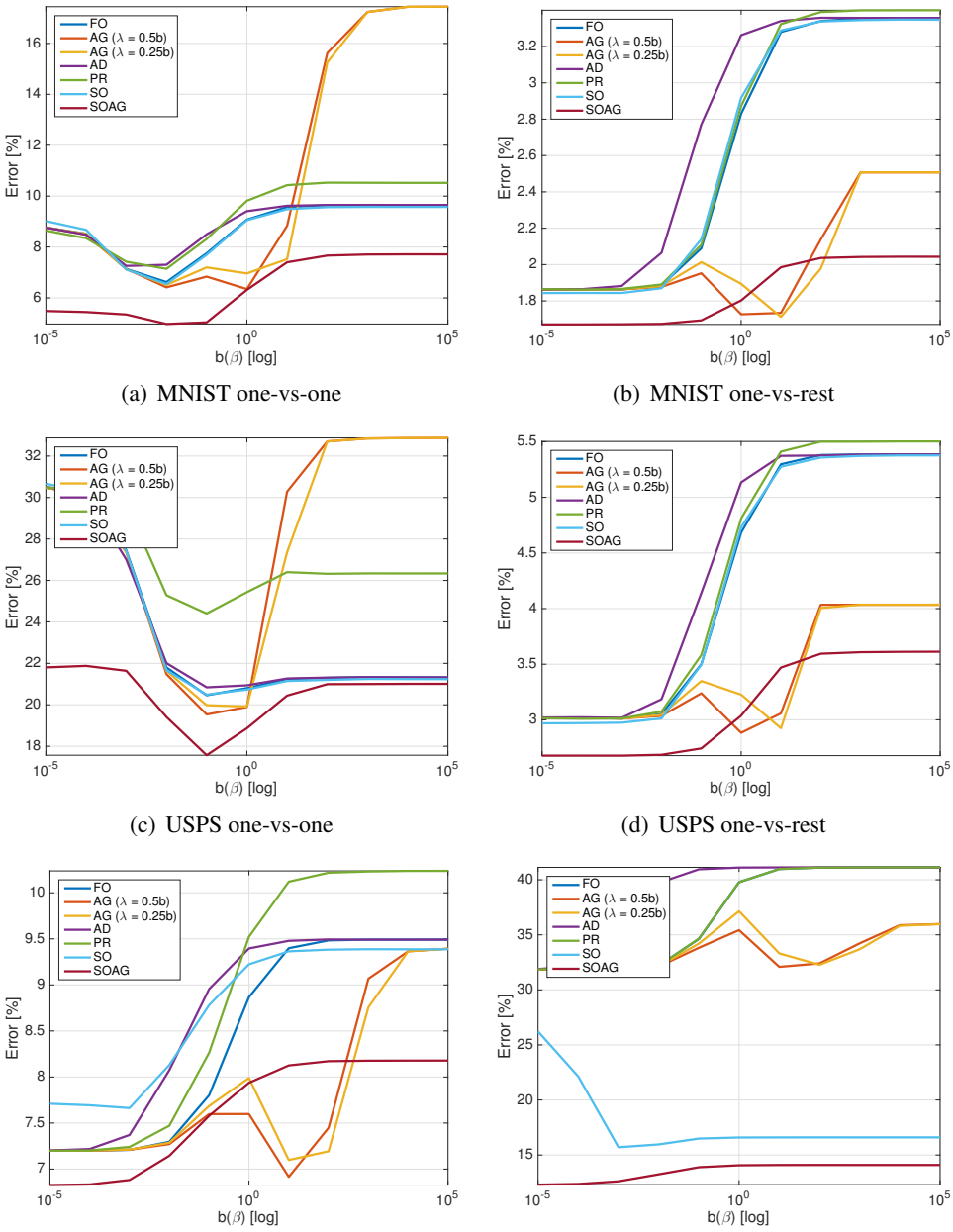
(a) 4 text classification tasks (b) 8 text classification tasks

Figure 5.1: Test error of aggressive SHAMPO on (a) four and (b) eight binary text classification tasks.

5.2 Multi-task Binary Classification

First, we evaluated all the mentioned variations of SHAMPO algorithms. We ran a single pass over all of the training examples for all datasets and 11 equally spaced values of b between 10^{-5} and 10^{5}. The algorithms performance was evaluated using mean of accumulated training error of all tasks and the mean of test errors over all of the tasks for each datasets. Since the NLP dataset doesn't have a test set, we evaluated those quantities using 10 cross validation folds. We repeated the experiments 100 times and took the mean of all algorithm runs. To simplify, we denote each one of those algorithms the abbreviations: First Order (FO), First Order AGgressive (AG) with two different values of λ, $\lambda = b/2$ and $\lambda = b/4$, First Order with PRior (PR), First Order ADaptive (AD), Second Order (SO) and Second Order AGgressive (SOAG). Recall, in the adaptive algorithm the constant b parameter is replaced with the constant β. We added confidence intervals only in the test error figures Fig. 5.3, since in the rest experiments the confidence intervals are very small.

Fig. 5.2 shows the mean cumulative training errors over all tasks. The behavior is similar for all of the datasets. We can see that as expected, in the exploration interval ($b \gg 1$, since the vector are normalized) all the algorithms shows high error rate, where in the vast majority, this is also the highest error. In the exploitation interval ($b \to 0$), we see lower cumulative error than the exploration case for the



(a) MNIST one-vs-one

(b) MNIST one-vs-rest

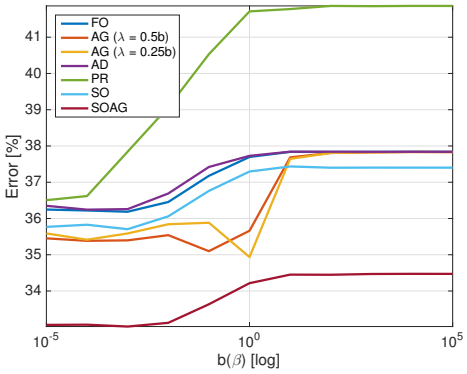
(c) USPS one-vs-one

(d) USPS one-vs-rest

(e) VJ one-vs-one

(f) VJ one-vs-rest

Figure 5.2: Training error - all datasets and algorithms

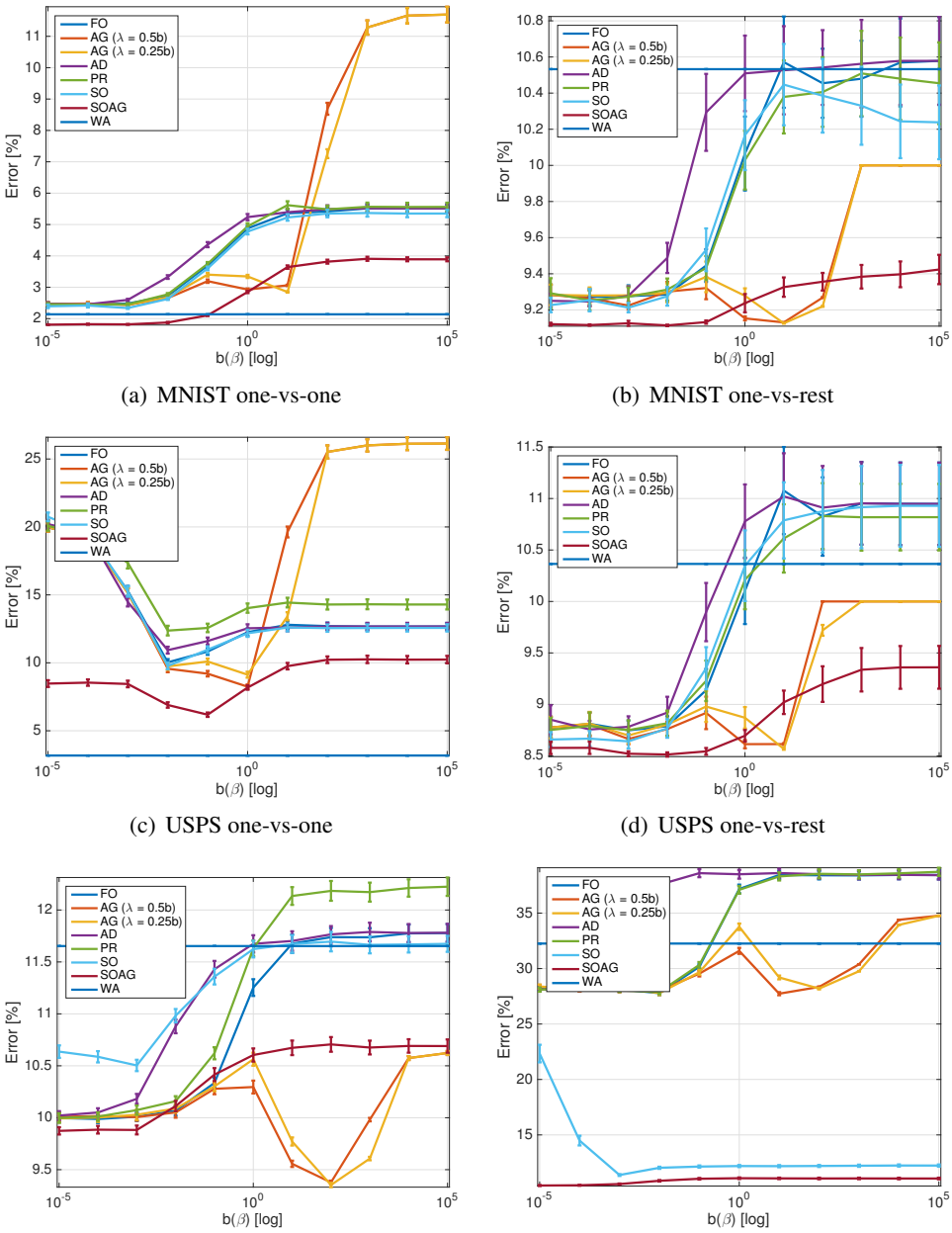


(g) NLP one-vs-one

Figure 5.2: Training error - all datasets and algorithms (cont.)

most of the datasets. This makes sense, since we query (and update) more on the harder tasks. Between those two extreme cases, there is a point where cumulative error is lower. This point is where b gets its optimal value. This value can change between datasets and algorithms and next we see a convenient experimental way to find it. There are algorithms and datasets (e.g. VJ one-vs-one in Fig. 5.2(e)), that this point converges into the exploitation area, it may depends on the examples noise and the number of examples in the dataset. As we claimed before, a pure exploitation approach may not be the best choice, at least on the first online steps, especially for problem with small number of examples. We can see the same phenomena in Fig. 5.3 which shows the mean test error for all tasks and all round for different b values and different tasks. This figure shows the actual prediction error on a new unseen data which can help us to estimate our performace with respect to other algorithms on the same dataset. Here, we added one more algorithm, the Watch All algorithm (WA), which runs K parallel online - perceptron algorithms, one per task, while each one of them query and watches all of the tasks labels, without any limitation on the feedback. We show this algorithm here as a benchmark, to measure how much SHAMPO algorithms looses or gain performance with respect to the algorithm that can watch all the labels.

When we compare all of the algorithms, there is no such absolute winner, however, it is obvious from results that the aggressive algorithms shows better performance than the rest of them. For most of the algorithms, the aggressive version of the second order (SOAG) seems to preform better then the rest. It is probably due



(a) MNIST one-vs-one

(b) MNIST one-vs-rest

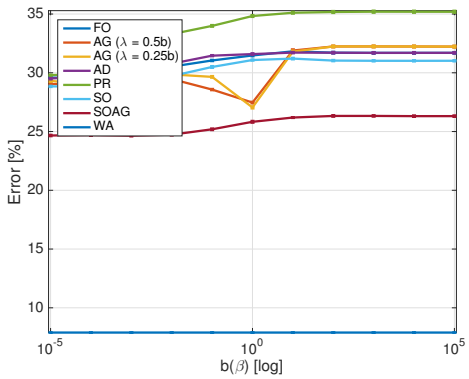
(c) USPS one-vs-one

(d) USPS one-vs-rest

(e) VJ one-vs-one

(f) VJ one-vs-rest

Figure 5.3: Testing error - all datasets and algorithms

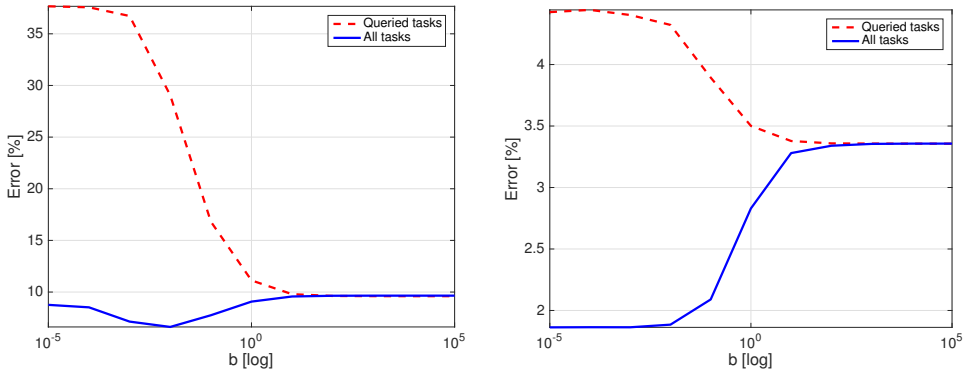


(g) NLP ove-vs-one

Figure 5.3: Testing error - all datasets and algorithms (cont.)

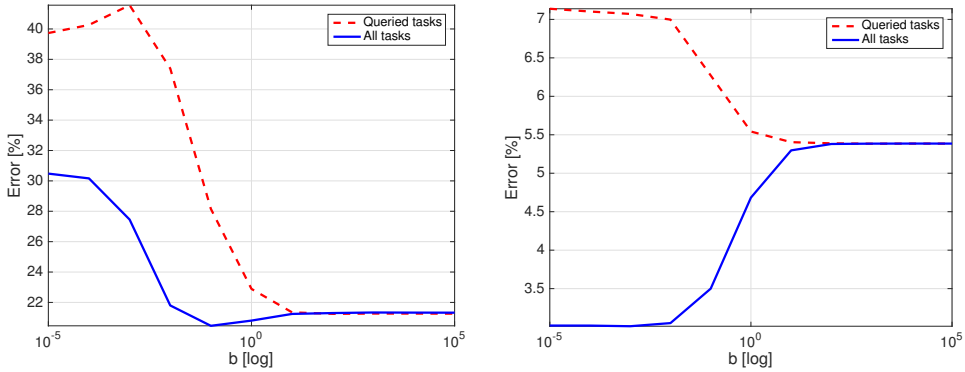
to the fact that it draws tasks to query on, based on two measurements ($r_{i,t}$ and $\hat{p}_{i,t}$) and also make an aggressive update. The second best algorithm is the first order aggressive algorithm (AG) which performs the best on a limited b interval because the aggressiveness depends strongly on b. The first order algorithm with prior (PR) doesn't show an improvement over the ordinary first order (FO) algorithm. In fact, we already mentioned before that better prior may lead to an improvements in the results. The adaptive b algorithm (AD), does not show an improvement over FO as well.

Now, we focus on the tasks that the algorithm chooses to annotate on each iteration for various values of b. Fig. 5.4 shows the total number of mistakes that first order SHAMPO algorithm made during training time on all datasets. We show here two quantities: fraction of mistakes over all training examples (denoted by "All tasks" - blue) and fraction of mistakes over only queried examples (denoted by "Queried tasks" - dashed red). In pure exploration (large values of b) both quantities are the same, as the choice of task to be labeled is independent of the task and example, and essentially the fraction of mistakes in queried examples is a good estimate of the fraction of mistakes over all examples. The other extreme is when performing pure exploitation (low values of b), here, the fraction of mistakes made on queried examples went up, while the overall fraction of mistakes went down. This indicates that the algorithm indeed focuses its queries on the harder inputs, which in turn, improves overall training mistake. There is a sweet point of b (that is changed between datasets), for which SHAMPO is still focusing on the



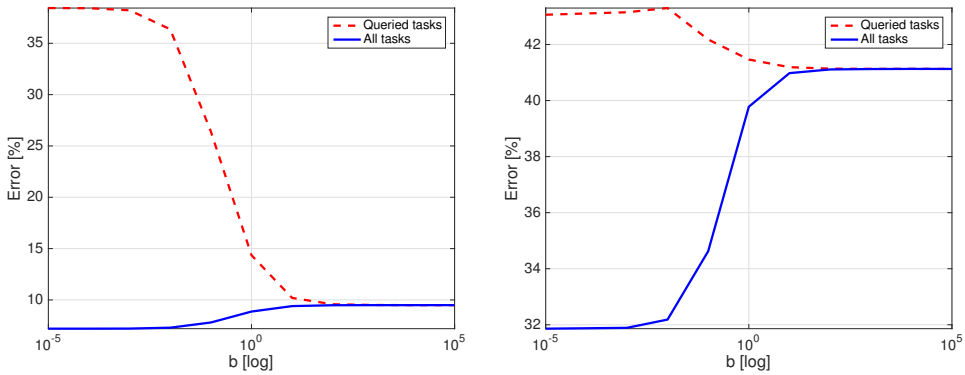
(a) MNIST one-vs-one

(b) MNIST one-vs-rest



(c) USPS one-vs-one

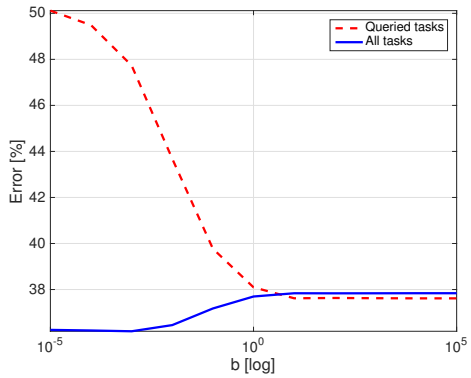
(d) USPS one-vs-rest



(e) VJ one-vs-one

(f) VJ one-vs-rest

Figure 5.4: Training error: queried vs. all - FO algorithm, all datasets

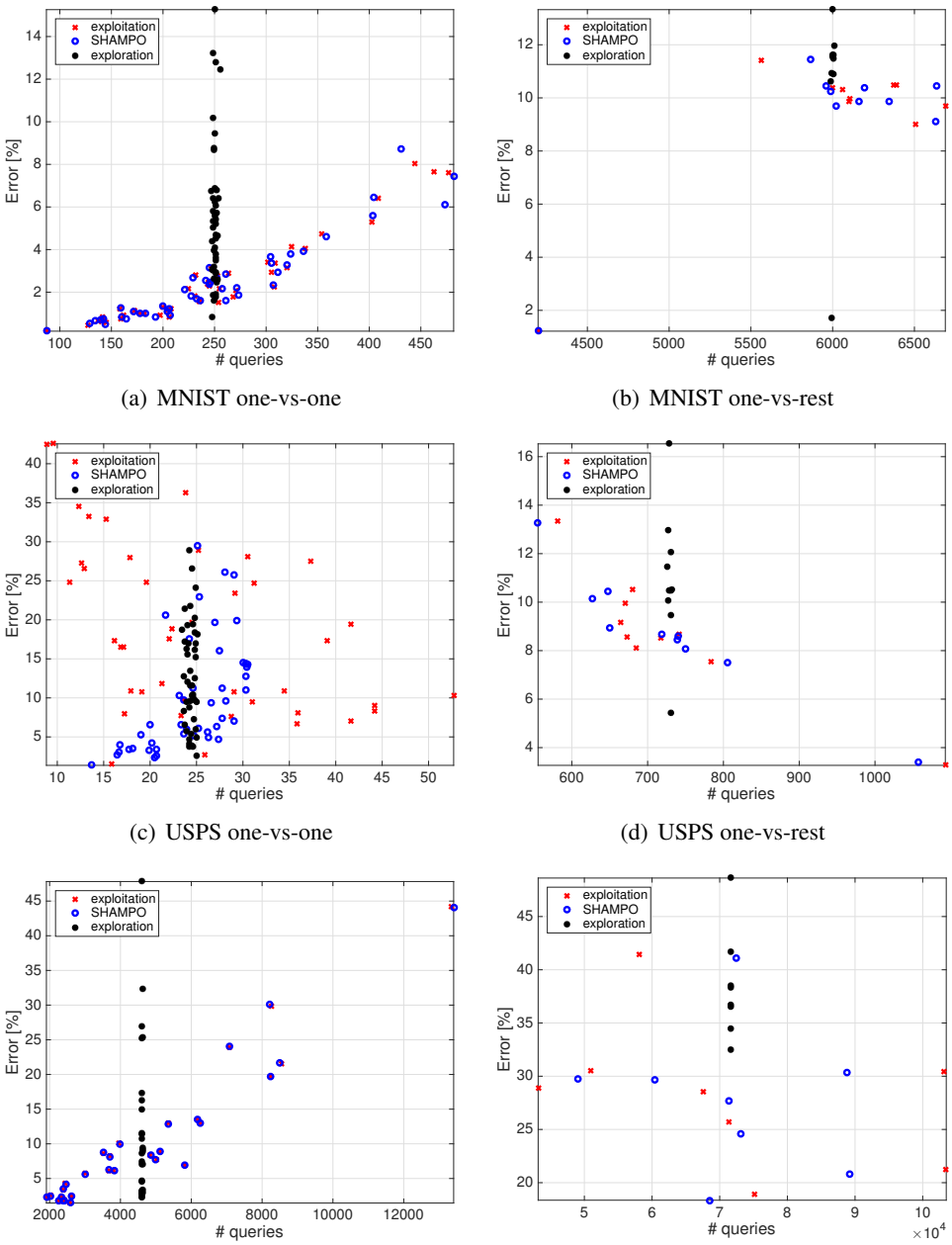


(g) NLP one-vs-one

Figure 5.4: Training error: queried vs. all - FO algorithm, all datasets (cont.)

harder examples, yet reduces the total fraction of training mistakes even more. The existence of such tradeoff is predicted by Theorem 3.1. So far we saw algorithms and bounds that depends on the value of b and proof that an optimal value that get a minimal accumulative train error exists. The inevitable question is: how to find such optimal value? We suggested an adaptive algorithm, yet, this algorithm have an initial value to set. Is there a practical way to find this value? Recall, the only prediction mistakes that the algorithm knows about, are the mistakes of the queried tasks. When we look at the accumulative mistakes that the algorithm made in Fig. 5.4, it is possible to find an answer for this question. The value of b which minimizes the mean test error, is about the point for which there is a change in the error of queried examples (the only quantity SHAMPO observes), which provides a rough rule-of-thumb to pick b automatically on-the-fly.

Another perspective of the phenomena is that for values of $b \ll 1$ SHAMPO focuses on the harder examples, is illustrated in Fig. 5.5 where test error vs number of queries is plotted for each task of all the datasets. We show three cases: uniform, exploit and a mid-value, the optimal one, which tradeoffs exploration and exploitation. We take the MNIST one-vs-one data set as an example and refer a few points. First performing uniform querying, all tasks have about the same number of queries (266), close to the number of examples per problem $(12,000)$, divided by the number of problems (45). Second, when having a tradeoff between exploration and exploitation ($b \approx 0.01$) , harder problems (as indicated by test error) get more queries than easier problems. For example, the four problems with test



(a) MNIST one-vs-one

(b) MNIST one-vs-rest

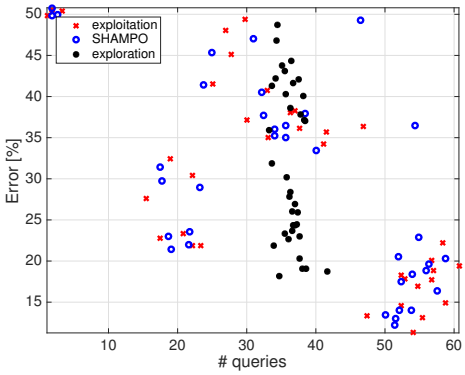
(c) USPS one-vs-one

(d) USPS one-vs-rest

(e) VJ one-vs-one

(f) VJ one-vs-rest

Figure 5.5: Queries vs. Testing error - all datasets and algorithms

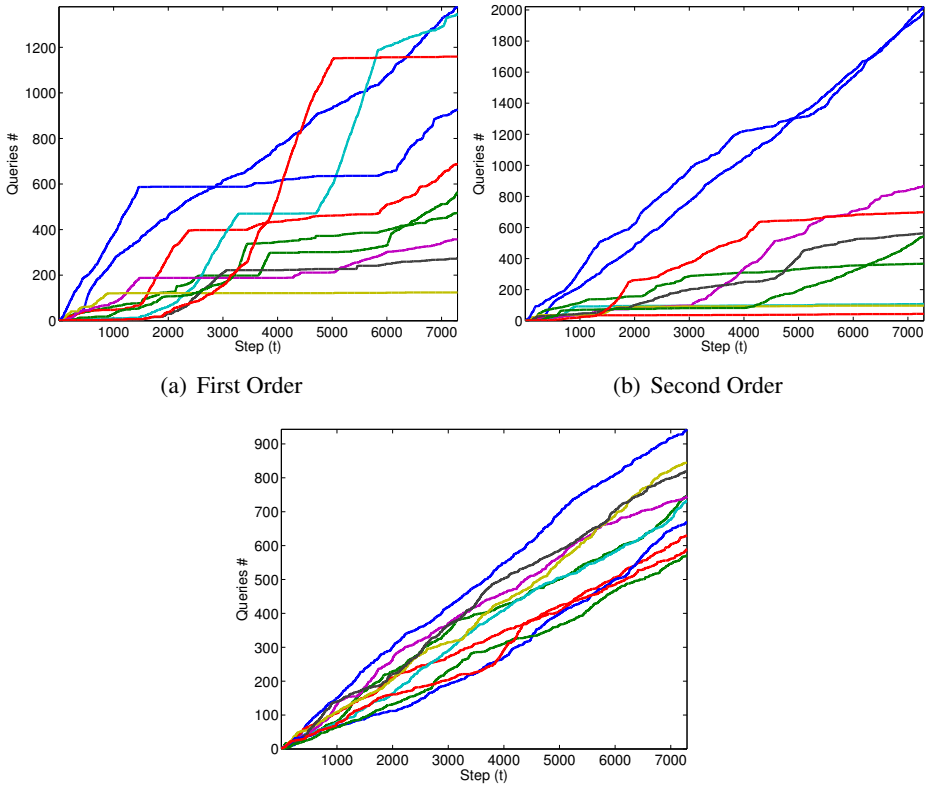


(g) NLP one-vs-one

Figure 5.5: Queries vs. Testing error - all datasets and algorithms (cont.)

error greater than 6% get at least 400 queries, which is about twice the number of queries received by each of the 10 problems with test error less than 1%. Third, as a consequence, SHAMPO performs equalization, giving the harder problems more labeled data, and as a consequence, reduces the error of these problems, however, is not increasing the error of the easier problems which gets less queries (in fact it reduces the test error of almost all 45 problems compare to the exploration method!). The tradeoff mechanism of SHAMPO, reduces the test error of each problem by more than 40% compared to full exploration. Fourth, exploits performs similar equalization, yet in some hard tasks it performs worse than SHAMPO. This could be because it overfits the training data, by focusing on hard-examples too much, as SHAMPO has a randomness mechanism.

In the past experiments, we concentrated on the final state of the algorithms: the training or testing mistakes at the end of the learning process. We used this final state to explain how the behavior of SHAMPO algorithms can exploit the use of the annotator and manage the queries between tasks in a way that reduces the cumulative mistake ratio. Indeed, the most desired property of such algorithm is reducing the cumulative mistake, yet, it is important to look into the learning process and see how the algorithm controls the queries over the tasks and validate our hypothesis and analysis. For this purpose, we chose the USPS one-vs-rest dataset, since it has a small number of tasks and examples. We ran a single run of the algorithm and logged the queries that each one of the tasks issued, and the prediction mistake of the algorithm for each one of the tasks at every time step. We



(a) First Order

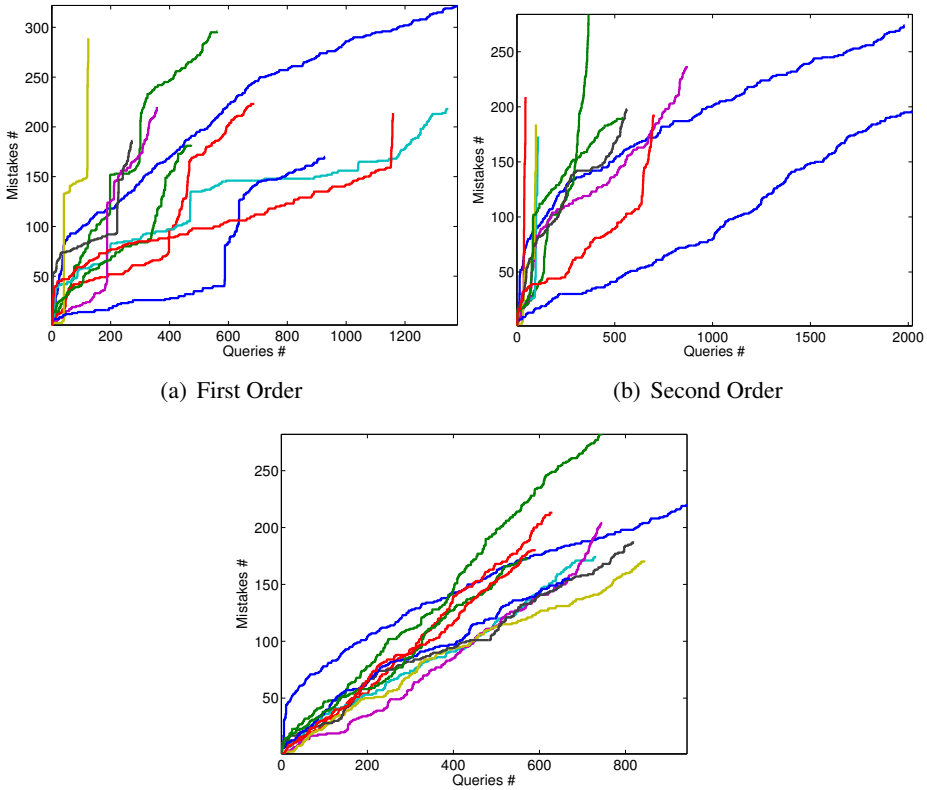
(b) Second Order

(c) Second Order Aggressive

Figure 5.6: Number of queries vs. time step, USPS one-vs-rest, all tasks

chose to show here the results of three algorithms: First Order, Second Order and Second Order Aggressive, in order to show different behavior of those algorithms. The algorithms parameter was initialized to $b = 10^{-3}$ which is close to the optimal value for all of these algorithms, as shown in Eq. (5.3(d))

The cumulative number of queries that each task issued up to a certain time step is shown in Eq. (5.6). The FO algorithm run (Fig. 5.6(a)) ends with $100 - 1,400$ queries for each task and alternate the preferred task to query (or not to query) on, sharply. There are tasks that issued a lot of queries in a short time interval, and then stopped querying for another time intervals repeatedly. This shows that this algorithm is less stable than the others. The SO algorithm run (Fig. 5.6(b)) ends with $50 - 2,000$ queries for each task, yet it focuses mainly on two tasks while the



(a) First Order

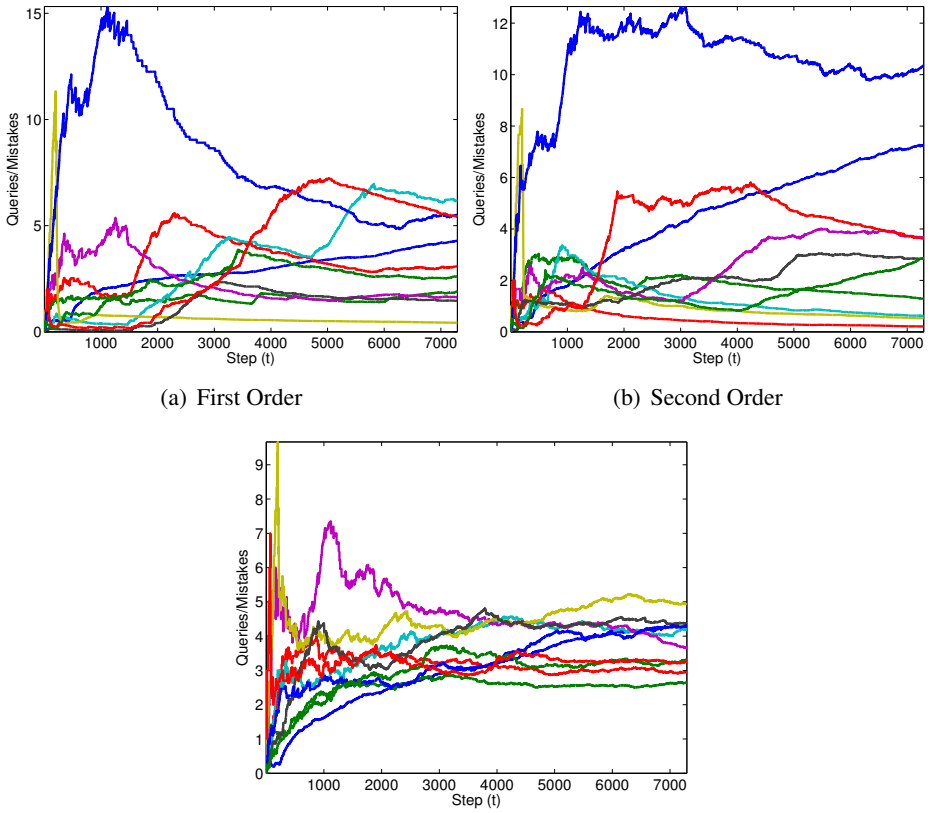
(b) Second Order

(c) Second Order Aggressive

Figure 5.7: Number of queries vs. number of mistakes, USPS one-vs-rest, all tasks

rest didn't issue more than 900 queries. The SOAG algorithm (Fig. 5.6(c)) uses the exploration-exploitation remarkably. It focuses more on (probably) harder tasks (expitation), since we see that it ends with $550 - 950$ queries for each task. Yet, it does't leave tasks too much time without querying (exploration), so we see smooth curves.

Eq. (5.7) shows the number of mistakes vs. number of queries. An expected behavior of the algorithm is that when the number of mistakes goes up (harder task), the number of queries will rise as well. In fact the relation between margin and high number of mistakes may not strictly holds for any task and dataset. We can see that the FO (Eq. (5.7(a))) shows this trend, but it's broken in the middle



(a) First Order

(b) Second Order

(c) Second Order Aggressive

Figure 5.8: $Queries/Mistakes$, USPS one-vs-rest, all tasks

(there are intervals when the mistakes goes up, but no queries are issued). This probably shows the intervals when the algorithm focuses on tasks that had higher margin at the same time. The same phenomena (even stronger) is seen in the SO algorithm (Eq. (5.7(b))) since it focuses mainly on two tasks over the others. The SOAG (Eq. (5.7(c))) on the other hand, shows the desired property of SHAMPO algorithm during all of the run time.

We also evaluated the error-query ratio over time for these algorithms as in Eq. (5.8). This measurement shows for each task, how many queries it issues per error. When this value is too low for one task compare to the other tasks, it means that the algorithm issue a lot of queries for the task, even though not many

prediction mistakes were done on the same task, which is a bad behavior. Saying that, this is not always bad when this measure is high. Recall in our setting we have to issue a query at every step, even when the model is converged and we don't make many mistakes, one of the tasks have to issue a query, which increases the number of queries for the tasks, which in turn, increase this ratio. In this case, we expect an exploration, which cause all the tasks to share the same, or close error-query ratio. We can see this exact behavior for the SOAG algorithm (Eq. (5.8(c))), when all the tasks show stable values in the range $(2.5 - 5)$ queries per mistake. However, low value of this measure is not desired as well, since it means that the algorithm makes a lot of mistakes on a certain task, but does not issue enough queries on that task, in order to improve the model and decrease the number of mistakes. We see in the FO algorithm (Eq. (5.8(a))) and even stronger in the SO (Eq. (5.8(b))) algorithm, that there is a difference between the task with the highest value and the task with the lowest value (1-7 for FO and 0.5-10 for SO).

In this experiment we saw that even though we want the algorithm to concentrate on the difficult tasks, the algorithm, should avoid the attraction to concentrate too much on a specific tasks and ignore the rest of them, i.e. an exploitation without an exploration may not be a good idea. We also noticed that the Second Order Aggressive algorithm produces high performance, not only in the sense of prediction mistakes, but also in the exploration-exploitation manner.

5.2.1 Choosing the Tradeoff Parameter

In all SHAMPO versions, there is a tradeoff parameter that need to be initialized. We analyzed the performance of all of SHAMPO algorithms on different datasets and shown how the cummulative classification error changes with this parameter. Yet, there is one inevitable question: how to choose the optimal value? An optimal parameter b^* is the one that minimize the cummulative error, i.e.

$$b^* = \underset{b}{\operatorname{argmin}}\, \mathbb{E}\left[\sum_{i=1}^{K}\sum_{t=1}^{n} M_{i,t}\right].$$

We have shown in Sec. 3.5 a version of an adaptive b parameter that changes with time, however, in this case there is different parameter for each task, and yet, we need to initialize β first.

We discussed already in Theorem 3.1 that the optimal value can't be calculated from the the bound, but we can write an optimal value as a function of $L_{\gamma,n}$. In fact,

the cummulative error is unknown during runtime so we have to find an empircal way to compute b. Recall the intuition behind SHAMPO algorithm, we would like to query on a tasks that related to wrong predictions. In Fig. 5.4 we see that for all datasets, the lowest error rate of the queried tasks occurs on high values of b, which indicates a high error rate on the overall cummulative mistakes rate. When we decrease b gradually, we see that the queried error increases continiously, and then, stay on a constant value. One can notice that the optimal b and it's close vicinity lies in the "knee" area of the queried tasks error. Now, the question is how to find this point? We can initialize our algorithm with high value of $b = b_0$ (pure exploration), a logarithmic step size - s and minimal number of examples - m. Then, we wait m iteration, compute the mean errors and set $b = b_0/s$, run the algorithm for m more iterations and compute the derivative betewen the last two computed mean errors o a logarithmic scale. We continue with this method until we find the area where the derivative hit -1 for the second time, then we keep this value unchanged. The final value of b is possibly close to the optimal.

5.3 Reduction of Multi-task to Contextual Bandits

We also evaluated SHAMPO as a contextual bandit algorithm, by breaking a multi-class problem into few binary tasks, and integrating their output into a single multi-class problem. We focus on the VJ data, as there are many examples, and linear models perform relatively well Lin et al. [2009]. We implemented all three reductions mentioned in Chapter 4, namely, *one-vs-rest*, *one-vs-one-random* which picks a random label if the feedback is zero, *one-vs-one-weak* (which performs updates to increase confidence when the feedback is zero), where we set $\eta = 0.2$, and the Banditron algorithm Kakade et al. [2008]. All algorithms show the existence of a tradeoff between exploration and exploitation, where *one-vs-one-random* is most sensitive to the choice of parameters, yet it also achieves the best results, for a large range of values for b. The *one-vs-rest* reduction and the Banditron have a test error of about 43.5%, and the *one-vs-one-random* of about 42.5%. Finally, *one-vs-one-weak* achieves an error of 39.4%. This is slightly worse than PLM Lin et al. [2009] with test error of 38.4% (and higher than MLP with 32.8%), yet all of these algorithms observe only one bit of feedback per example, while both MLP and PLM observe 3 bits (as class identity can be coded with 3 bits for 8 classes). We claim that our setting can be easily used to adapt a system to individual user, as we only need to assume the ability to recognize three words, such as three letters.

Given an utterance of the user, the system may ask: “Did you say (a) ’a’ like ’bad’ (b) ’o’ like in ’book’) (c) none”. The user can communicate the correct answer with no need for a another person to key in the answer.

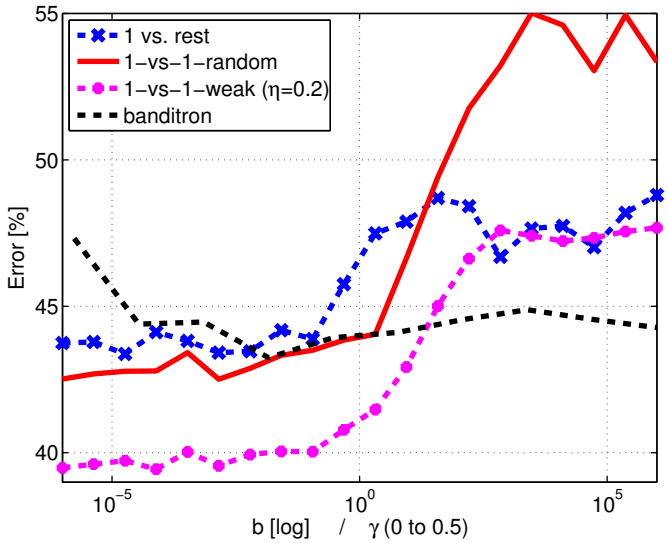


Figure 5.9: Multi-class error on VJ data for three bandit reductions and the banditron.

5.4 Edge Cases

We discussed about the advantages of the SHAMPO algorithm with a tradoff value that control exploration-expoitation and provided analysis of the cummulative mistake bounds. We also supplied experiments that support the intuition. However, the SHAMPO algorithms not always works better than the other options (pure exploration or pure exploitation).

One of the assumptions of the SHAMPO algorithms is that we can compare margins of different tasks. In order to comply with this assumption a scaling of the input vector is required. Yet, when in one or more tasks where the norm of one vector is very big with respect to the majority of the vectors. When we scale the vectors in this setting into a ball, we can’t longer compare the margin, because the margins of the tasks with the she high norm are much lower than the margin of the

rest tasks and the SHAMPO algorithms may not work properly. This problem can be solved by just a scaling, but normalizing all the input vectors onto a unit ball.

Another reason that using the SHAMPO algorithm can lead to a failure, can be when an adversary is involved. In the first order SHAMPO for example, when we set the b parameter to a low value ($b \rightarrow 0$), the SHAMPO algorithm is not stochastic any more and an adversary can expliot this case and choose every iterations, examples that don't add information to the algorithm. In order to select an uninformative examples, the adversary can choose examples that are similar to the examples that the algorithm got feedback before. As we shown before, using the second order agressive SHAMPO with the confidence measure $r_{i,t}$ could help to solve this problem. In fact, the adversary can select the uninformative example which cause a low margin and in turn will lead to an update, while for the rest of the task, the adversary can choose any example, since no update will be done on those tasks due to a delta distribution. Another option that the adversary can act, is choosing an example that the algorithm will predict a correct prediction on it, but will result with a small margin. For the rest of the tasks, the adversary can choose examples that lead to wrong prediction (with slightly higher margins), because no update will be made on those tasks any way. The adversary can also choose examples with very high margin for all of the tasks, which may decrease the learning rate.

Chapter 6

Related Work

In the past few years there is a large volume of work on multi-task learning, which clearly we can not cover here. The reader is referred to a recent survey on the topic by Pan and Yang [2010]. Most of this work is focused on exploring relations between tasks, that is, find similarities and dissimilarities between tasks, and use it to share data directly ,e.g. Crammer and Mansour [2012], or model parameters as in Evgeniou and Pontil [2004], Daumé et al. [2010], Argyriou et al. [2008]. In the online settings, there are only a handful of work on multi-task learning. Dekel et al. [2006] consider the setting where all algorithms are evaluated using a global loss function, and all work towards the shared goal of minimizing it. Lugosi et al. [2009], assume that there are constraints on the predictions of all learners, and focus in the expert setting. Agarwal et al. [2008], formalize the problem in the framework of stochastic convex programming with few matrix regularization, each captures some assumption about the relation between the models. Cavallanti et al. [2010] and Cesa-Bianchi et al. [2006b], assume a known relation between tasks which is exploited during learning. Unlike these approaches, we assume the ability to share an annotator rather than data or parameters, thus our methods can be applied to problems with no common input space. One can expand our algorithm in the case of a related task, and update all the tasks of the model on each round, based on the queried task and not only the queried task.

On the other hand, there are also works that have been done on binary classification with partial feedback . Our analysis is derived from Cesa-Bianchi et al. [2006c], yet they focus in selective sampling (see also Cesa-Bianchi et al. [2009], Dekel et al. [2010], Crammer [2014]), that is, making individual binary decisions of whether to query, while our algorithm always query, and needs to decide for

which task to query on each round. Selective sampling algorithms such as first and second order selective sampling perceptron and BBQ of Cesa-Bianchi et al. [2006c, 2009] deals with binary selective sampling algorithms. Dekel et al. [2010] have also a multiple teachers selective sampling algorithm, with setting that is similar to SHAMPO with κ updates per round , but is force all of the inputs to be in the same space.

Dietterich and Bakiri [1995] introduced the use of ECOC matrix to reduce a multi-class classification problem into a multi-task problem we use this method, combines with SHAMPO algorithms to get multi-class classification. There have been recent work in contextual bandits each with slightly different assumptions. Algorithms like contextual bandits such the Banditron of Kakade et al. [2008] and Newtron of Hazan and Kale [2011] used the exploration exploitation method in order to reduce the regret, but in their setting, for each round the choice is also the prediction, so it suitable to our setting only for the case of One-vs-Rest. To the best of our knowledge, we are the first to consider decoupled exploration and exploitation in this context.

Finally, there is recent work in learning with relative or preference feedback in various settings as in Yue et al. [2009, 2012], Yue and Joachims [2011], Shivaswamy and Joachims [2011]. Unlike this work, our work allows again decoupled exploitation and exploration, and also non-relevant feedback.

Chapter 7

Summary and Conclusions

We proposed a new framework for online multi-task learning, where learners share a single annotator. We presented number of (SHAMPO) algorithms that work in this settings : The First Order , First Order Aggressive, First Order Adaptive , First Order with Prior, Second Order and Aggressive Second Order. Mistake bound analysis was done for most of the proposed algorithms. Then, we showed how learning in such model can be used to learn in contextual-bandits setting with few types of feedback. Empirical results show that our algorithms, with wise choosing of parameters, does better than only random uniform choice of feedback, for the same price. It focuses the annotator on the harder instances, and is improving performance in various tasks and settings.

Further work can be done to improve those algorithms. We have shown that there exist an optimal value of b that may vary for different multitask problems. We fed the algorithm with the parameter b, yet we did not talk about how to find such optimal value. Now, after we shown that such parameter exists, we can try to derive an algorithm that finds this optimal value for every dataset. We can also investigate ways to generate good prior distribution for the SHAMPO with prior algorithm. Another way can be deriving another distribution that select the task to be queried at every step, not only by the margin and the number of queried mistakes (as in the adaptive version) up to the same step, but also consider the relation between number of queries, and number of updates for every task, since for a task that query all the time and doesn't improve the results, we may want to reduce the queries rate. Another improvement can be done if we assume a relation between tasks and at each step we update not only the queried tasks, but also similar tasks or all the tasks, exploiting the relation between the tasks .

Bibliography

Alekh Agarwal, Alexander Rakhlin, and Peter Bartlett. Matrix regularization techniques for online multitask learning. Technical Report UCB/EECS-2008-138, EECS Department, University of California, Berkeley, Oct 2008. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-138.html`.

E.L. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research*, 1:113–141, 2001.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multitask feature learning. *Machine Learning*, 73(3):243–272, 2008.

Orly Avner, Shie Mannor, and Ohad Shamir. Decoupling exploration and exploitation in multi-armed bandits. In *ICML*, 2012.

Katy S. Azoury and Manfred K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43 (3):211–246, 2001.

Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 11: 2901–2934, 2010.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Worst-case analysis of selective sampling for linear-threshold algorithms. In *NIPS*, 2004.

Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. A second-order perceptron algorithm. *Siam Journal of Commutation*, 34(3):640–668, 2005.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Worst-case analysis of selective sampling for linear classification. *Journal of Machine Learning Research*, 7:1205–1230, 2006a.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Incremental algorithms for hierarchical classification. *The Journal of Machine Learning Research*, 7: 31–54, 2006b.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Worst-case analysis of selective sampling for linear classification. *The Journal of Machine Learning Research*, 7:1205–1230, 2006c.

Nicolo Cesa-Bianchi, Claudio Gentile, and Francesco Orabona. Robust bounds for classification via selective sampling. In *ICML 26*, 2009.

Koby Crammer. Doubly aggressive selective sampling algorithms for classification. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 140–148. Citeseer, 2014.

Koby Crammer and Claudio Gentile. Multiclass classification with bandit feedback using adaptive regularization. *Machine Learning*, 90(3):347–383, 2013.

Koby Crammer and Yishay Mansour. Learning multiple tasks using shared hypotheses. In *Advances in Neural Information Processing Systems 25*. 2012. URL `http://books.nips.cc/papers/files/nips25/NIPS2012_0706.pdf`.

Koby Crammer, Mark Dredze, and Fernando Pereira. Confidence-weighted linear classification for text categorization. *J. Mach. Learn. Res.*, 98888:1891–1926, June 2012. ISSN 1532-4435.

Hal Daumé, III, Abhishek Kumar, and Avishek Saha. Frustratingly easy semi-supervised domain adaptation. In *DANLP 2010*, 2010.

Ofer Dekel, Philip M. Long, and Yoram Singer. Online multitask learning. In *COLT*, pages 453–467, 2006.

Ofer Dekel, Claudio Gentile, and Karthik Sridharan. Robust selective sampling from single and multiple teachers. In *COLT*, pages 346–358, 2010.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *arXiv preprint cs/9501101*, 1995.

Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 109–117, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: 10.1145/1014052.1014067. URL `http://doi.acm.org/10.1145/1014052.1014067`.

Jurgen Forster. On relative loss bounds in generalized linear regression. In *FCT*, 1999. ISBN 3-540-66412-2.

Jürgen Forster and Manfred K Warmuth. Relative expected instantaneous loss bounds. *Journal of Computer and System Sciences*, 64(1):76–102, 2002.

Johannes Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

Elad Hazan and Satyen Kale. Newtron: an efficient bandit algorithm for online multiclass prediction. *Advances in Neural Information Processing Systems (NIPS)*, 2011.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of the 25th international conference on Machine learning*, pages 440–447. ACM, 2008.

Hui Lin, Jeff Bilmes, and Koby Crammer. How to lose confidence: Probabilistic linear machines for multiclass classification. In *Tenth Annual Conference of the International Speech Communication Association*, 2009.

Gábor Lugosi, Omiros Papaspiliopoulos, and Gilles Stoltz. Online multi-task learning with hard constraints. In *COLT*, 2009.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. ISSN 1041-4347. doi: http://doi.ieeecomputersociety.org/10.1109/TKDE.2009.191.

Pannagadatta K. Shivaswamy and Thorsten Joachims. Online learning with preference feedback. *CoRR*, abs/1111.0712, 2011.

Simon Tong and Daphne Koller. Support vector machine active learning with application sto text classification. In *ICML*, pages 999–1006, 2000.

V. Vovk. Competitive on-line linear regression. Technical Report Technical Report CSD-TR-97-13, Department of Computer Science, Royal Holloway, University of London, 1997.

Jia Yuan Yu and Shie Mannor. Piecewise-stationary bandit problems with side observations. In *ICML*, 2009.

Yisong Yue and Thorsten Joachims. Beat the mean bandit. In *ICML*, pages 241–248, 2011.

Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. In *COLT*, 2009.

Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *J. Comput. Syst. Sci.*, 78(5):1538–1556, 2012.

ההסתברות נוטה לכיוון אחת מן המשימות – מיצוי (exploitation). בין שני ערכי הקיצון אלו קיים ערך של הפרמטר המביא לפילוג ביניים שמאזן בין מקרי הקיצון, ערך זה הינו הערך האופטימלי. האלגוריתם שאנו מציעים, מבוסס על אלגוריתם הפרספטרון המוכר עבור סיווג בינארי, בשילוב הפילוג אותו בנינו. את אלגוריתם זה אנו מכנים בשם SHAMPO (קיצור של Shared Annotator for Multiple PrOblems).

בנוסף, אנו מציעים מספר ווריאציות נוספות של אלגוריתם SHAMPO. וריאציה של אלגוריתם אגרסיבי, בו עדכון המודל מתבצע לא רק בעת שגיאה, אלא גם כאשר אין שגיאת סיווג, אך השוליים קטנים מתחת לסף מסויים. וריאציה נוספת, בה נעשה שימוש במידע פריורי על מידת הקושי של משימות הסיווג, בבניית הפילוג על פני המשימות. ישנה וריאציה נוספת של האלגוריתם בה הפרמטר b הינו אדפטיבי, ומשתנה בהתאם למספר העדכונים שבוצעו עבור כל אחת ממשימות הסיווג. לסיום, מוצגת ווריאציה המשלבת את אלגוריתם הפרספטרון מסדר שני. עבור כל אחת מוווריאציות אלו של האלגוריתם אנו מבצעים אנליזה של עבור חסם הסכום המצטבר של השגיאות בכל צעד.

לאחר מכן אנו מציגים כיצד ניתן להשתמש בגישה זו של פתרון מספר משימות בינאריות במקביל עם פרשן משותף, על מנת לפתור בעיות סיווג מרובה מחלקות באמצעות מספר בעיות המוכרות בשם Contextual Bandits בשילוב אלגוריתם SHAMPO ובצירוף חסמים ספציפיים עבור בעיות אלו. לסיום, אנו מודדים את האלגוריתמים שהצענו על מספר מאגרי מידע שונים: שני מאגרי מידע של זיהוי ספרות בתמונות (OCR) בשם MNIST ו- USPS. מאגר מידע נוסף של זיהוי תנועות קוליות (Vowels Prediction) בשם VJ (Vocal Joystick). ומאגר מידע נוסף המשולב מכמה תתי מאגרים של סיווג מסמכים וניתוח הלך רוח (Sentiment Analysis). אנו מראים כי שימוש באלגוריתם SHAMPO ובחירה מתאימה של הפרמטר b, משפר את הביצועים על ממוצע השגיאות על פני כל המשימות ואף עבור שילוב של משימות הסיווג הבינאריות למשימת סיווג אחת מרובת מחלקות. אנו מראים גם כי אכן למרות שהמשימות הקלות יותר דורשות פחות משוב, השגיאה עליהן אינה פוחתת (לפחות לא באופן משמעותי).

אנו מסיימים את חיבור זה בדיון על עבודות וחיבורים המתקשרים לחיבור זה ובמספר דרכים נוספות להרחבת עבודה זו.

שקיבל, אל הפרשן. הפרשן מוגבל ואינו מסוגל לסווג מספר דוגמאות במקביל, אלא דוגמה יחידה בכל צעד. כלומר, המשימות חולקות פרשן משותף וישנו צורך בכל אחד מן הצעדים, להחליט מהי המשימה שאת הדוגמה שהתקבלה עבורה, יתייג הפרשן. לאחר שמתקבל הסיווג הנכון מן הפרשן, האלגוריתם מעדכן את המודל של במשימה המתאימה, על ידי שימוש במשוב שהתקבל מן הפרשן.

משימת האלגוריתם היא להחליט בכל שלב, מהי המשימה (או מהי הדוגמה) עבורה יבקש משוב מן הפרשן. באופן אינטואיטיבי, נעדיף בכל שלב לבקש מן הפרשן לסווג את הדוגמה עבורה הסיווג שנתן האלגוריתם, בעל הוודאות הנמוכה ביותר. באופן זה נוכל להיות בטוחים שלא יבוזבז זמן פרשן על דוגמאות ומשימות, עבורן האלגוריתם די בטוח בסיווג שלו והצורך שלו במשוב עבור משימות ודוגמאות אלו הינו פחות. לשם כך, אנו מציעים להשתמש במדד המקובל של הערך המוחלט של השוליים המוגדר כמרחק הדוגמה מן המסווג הלינארי באותו השלב. ניתן לראות בגודל זה כמדד כלשהו עבור מידת הביטחון בסיווג האלגוריתם את הדוגמה. כאשר השוליים גדולים, הביטחון בסיווג של האלגוריתם גדול יחסית, מכיוון שהדוגמה מרוחקת מהמסווג, ואילו כאשר השוליים קטנים יחסית, הביטחון בסיווג קטן יותר. בטענה דומה השתמשו במאמרים נוספים העוסקים בלמידה אקטיבית עבור קבוצת דוגמאות ובמאמרים בנושא דגימה סלקטיבית, על מנת להחליט אלו מן הדוגמאות יש לתייג.

אם כן, על פי מדד השוליים, הבחירה האינטואיטיבית תהיה מן הסתם בכל צעד, הדוגמה והמשימה עבורה השוליים הם הקטנים ביותר מהשוליים של שאר הדוגמאות המשוייכות למשימות השונות באותו הצעד. למעשה, כאשר המודל אינו מדוייק מספיק, הסיווג יהיה בקירוב גס מאוד, מה שעלול להוביל לשגיאות בסיווג ובהשלכת גודל השוליים על הביטחון בסיווג. תופעה זו קורה בעיקר בצעדים הראשוניים של האלגוריתם, כאשר המודל עדיין לא הושפע ממספיק דוגמאות ואינו מכוייל מספיק. לשם כך, בנינו פילוג על פני כל משימות הסיווג, כאשר בכל צעד מוגרלת המשימה עבורה יש לבקש מהפרשן סיווג. פילוג זה, אכן נותן משקל רב יותר למשימות ככל שהדוגמאות שלהן בעלות שוליים קטנים יותר באותו הצעד. הפילוג המוצע הינו הכללה של מקרי הקצה (exploration-exploitation) באמצעות שימוש בפרמטר חיובי קבוע – b. כאשר פרמטר זה גדול מאוד (שואף לאינסוף), הפילוג הופך לאחיד, וכך מתבצעת דגימה אחידה בכל צעד על פני כל המשימות - גישוש (exploration) . כאשר פרמטר זה קטן במיוחד (שואף לאפס), במקרה זה, הפילוג הופך לפילוג דלתא, כאשר כל משקל

תקציר

בלמידה מונחית, צוואר הבקבוק העיקרי הוא הצורך בקבלת הסיווג הנכון עבור המידע. שיטה נפוצה בקטגוריה זו עובדת באופן הבא: ראשית יש צורך לאסוף מידע רב באופן אוטומטי (שלב זה לכשעצמו הוא קל יחסית ובעל מחיר זול), ולאחר מכן להעביר את המידע למשתמש או למומחה על מנת שיתן את הסיווג הנכון עבור כל אחת מן הדוגמאות שאספנו. שלב זה של פירוש וסיווג המידע המתקבל יכול להיעשות באמצעות מיקור חוץ לקהל רחב (כדוגמת פרוייקט ה recaptcha של גוגל) או על ידי מומחה (כמו ב- Linguistic data Consortium). לאחר מכן, ניתן להשתמש במידע שסווג על מנת לבנות מודלים עבור משימת סיווג יחידה, או מספר משימות. החיסרון בגישה זו הוא שאינה מנצלת באופן אופטימלי את המשאב העיקרי (והיקר בדרך כלל), הלא הוא הפרשן. פעמים רבות, משימות סיווג שונות אינן בעלות דרגת קושי זהה, ישנן משימות קלות יותר וישנן קשות יותר. לכן, יתכן ומתן כל כמות המידע עבור כל המשימות יחד, לסיווג ע"י פרשן אינה תמיד רעיון טוב.

היבט נוסף של בעיה זו הינו הצורך להעביר את המידע למשתמש לצורך סיווג, כך שהמערכת תבקש מהמשתמש את הסיווג הנכון עבור כל דוגמה שמתקבלת. עם זאת, כאשר מספר מערכות מעבירות למשתמש מידע לסיווג במקביל ובאופן בלתי תלוי אחת בשניה, המשתמש יהיה מוצף במידע ולא יהיה מסוגל לתפקד באופן הרצוי.

לדוגמה, לעיתים ישנו הצורך לסווג מידע חדשותי ממספר סוכנויות. אדם יחיד אינו מסוגל להתמודד עם כמות גדולה של מידע במקביל, אלא הוא מוגבל לסיווג מספר יחידות מידע חדשותי בזמן מוגבל. אם כך, כיצד יוחלט איזו מהן יסווג?

הגדרת הבעיה בה ענו עוסקים בחיבור זה, תוכננה להתמודד בדיוק עם בעיות מסוג זה, ובפרט, כיצד למצות באופן האופטימלי את זמן הפרשן המסווג המוגבל.

בחיבור זה, אנו מציעים גישה חדשה של למידת מספר משימות סיווג בינאריות במקביל באופן מקוון, החולקות יחדיו פרשן יחיד. לימוד המשימות מתבצע בצעדים, כאשר בכל צעד, מתקבלות לסיווג מספר דוגמאות, דוגמה אחת מכל משימת סיווג. האלגוריתם נותן עבור כל אחת מהדוגמאות המתקבלות, את הסיווג המתאים ביותר על פי המודל שלמד עד צעד זה. לאחר מכן, נדרש האלגוריתם לעדכן את המודל, על מנת לספק סיווג מדוייק יותר בעתיד. לשם כך המודל מעביר את הדוגמאות

פרסומים

חלקים מעבודה זו פורסמו במאמר

Learning Multiple Tasks in Parallel with a Shared Annotator
Haim Cohen and Koby Crammer. NIPS 2014.

המחקר נעשה בהנחיית פרופ' קובי קרמר בפקולטה להנדסת חשמל.

תודה לקובי על ההדרכה, ההכוונה והליווי במהלך כל הדרך.
תודה מיוחדת לאשתי זוהר ולשני בני שפר וסער, על התמיכה, הסבלנות והעידוד במהלך התקופות העמוסות.

לימוד משימות מרובות עם פרשן משותף

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים בהנדסת חשמל

חיים כהן

הוגש לסנט הטכניון – מכון טכנולוגי לישראל
כסליו תשע"ז חיפה דצמבר 2016

לימוד משימות מרובות עם פרשן משותף

חיים כהן