Linear Algorithms for Online Multitask Classification

Giovanni Cavallanti[*] **Nicolò Cesa-Bianchi**[†] **Claudio Gentile**[‡]

Abstract

We design and analyze interacting online algorithms for multitask classification that perform better than independent learners whenever the tasks are related in a certain sense. We formalize task relatedness in different ways, and derive formal guarantees on the performance advantage provided by interaction. Our online analysis gives new stimulating insights into previously known co-regularization techniques, such as the multitask kernels and the margin correlation analysis for multiview learning. In the last part we apply our approach to spectral co-regularization: we introduce a natural matrix extension of the quasi-additive algorithm for classification and prove bounds depending on certain unitarily invariant norms of the matrix of task coefficients.

1 Introduction

A fundamental and fascinating problem in learning theory is the study of learning algorithms that influence each other. Although much is known about the behavior of individual strategies that learn a classification or regression task from examples, our understanding of interacting learning systems is still fairly limited. In this paper, we investigate this problem from the specific viewpoint of *multitask learning*, where each one of $K > 1$ learners has to solve a different task (typically, K classification or K regression tasks). In particular, we focus on multitask binary classification, where learners are *online* linear classifiers (such as the Perceptron algorithm). Our goal is to design online interacting algorithms that perform better than independent learners whenever the tasks are related in a certain sense. We formalize task relatedness in different ways, and derive formal guarantees on the performance advantage provided by interaction.

Our analysis builds on ideas that have been developed in the context of statistical learning. In the statistical analysis of multitask learning (e.g., [2, 3, 4, 11, 24, 26]) the starting point is a regularized empirical loss functional or Tikhonov functional —see, e.g., [10]. In the presence of several tasks, this functional is extended to allow for co-regularization among tasks. Roughly speaking, the co-regularization term forces the set of predictive functions for the K tasks to lie "close" to each other.

This co-regularization term is typically a squared norm in some Hilbert space of functions. We follow the approach pioneered by [11], where the K estimated solutions are linear functions parametrized by $\boldsymbol{u} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K) \in \mathbb{R}^{Kd}$ and the co-regularization is $\boldsymbol{u}^\top A\, \boldsymbol{u}$, where A is a positive definite matrix enforcing certain relations among tasks. The key observation in [11] is the following. Assume the instances of the multitask problem are of the form $(\boldsymbol{x}_t, i_t)$, where $\boldsymbol{x}_t \in \mathbb{R}^d$ is an attribute vector and $i_t \in \{1, \ldots, K\}$ indicates the task $\boldsymbol{x}_t$ refers to. Then one can reduce the K learning problems in $\mathbb{R}^d$ to a single problem in $\mathbb{R}^{Kd}$ by choosing a suitable embedding of the pairs $(\boldsymbol{x}_t, i_t)$ into a common RKHS space $\mathbb{R}^{Kd}$ with inner product $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \boldsymbol{u}^\top A\, \boldsymbol{v}$. This reduction allows us to solve a multitask learning problem by running any kernel-based single-task learning algorithm with the "multitask kernel" defined above. We build on this result by considering a natural online protocol for multitask linear classification. Within this protocol we analyze the performance of the Perceptron algorithm and some of its variants when run with a multitask kernel. Because such kernels are linear, we are not restricted to using kernel-based algorithms for efficiency reasons.

In Section 3 we consider the kernel Perceptron algorithm, and derive mistake bounds for the multitask kernels proposed in [11]. This reveals new insights into the role played by the regularizer matrix A. First, we see that the update in the kernel space defined by A factorizes in the "shared update" of K interacting Perceptrons each running in $\mathbb{R}^d$, thus providing a basic example of interactive online learning. Second, we exploit the simplicity of the mistake bound analysis to precisely quantify the performance advantage brought by the multitask approach over K independent online algorithms. In particular, in Subsections 3.2 and 3.3 we give examples where the mistake bound is used to guide the design of A. The first part of the paper is concluded with Sections 4 and 5, where we show multitask versions and mistake bound analyses for the second-order Perceptron algorithm of [7] and for the p-norm algorithm of [13, 14].

In the remaining sections of the paper, we depart from the approach of [11] to investigate the power of online learning when other forms of co-regularization are used. In Sec-

[*]DSI, Università di Milano, Italy.

[†]DSI, Università di Milano, Italy.

[‡]DICOM, Università dell'Insubria, Italy. This is the corresponding author. Email: `claudio.gentile@uninsubria.it`

tion 6 we consider the case when instances belong to a space that is different for each task, and the similarity among tasks is measured by comparing their margin sequences (see, e.g., [6, 27]). We introduce and analyze a new multitask variant of the second-order Perceptron algorithm. The mistake bound that we prove is a margin-based version of the bound shown in Subsection 3.2 for the multitask Perceptron. Finally, in Section 7 we consider spectral co-regularization [4] for online multiview learning. Here diversity is penalized using a norm function defined on the $d \times K$ matrix $U = \big[\boldsymbol{u}_1, \dots, \boldsymbol{u}_K\big]$ of view vectors. In the spirit of [19], we interpret this penalization function as a potential defined over arbitrary matrices. We then define a natural extension of the quasi-additive algorithm of [14, 20] to a certain class of matrix norms, and provide a mistake bound analysis depending on the singular values of U. The results we obtain are similar to those in [28, 29, 30], though we are able to overcome some of the difficulties encountered therein via a careful study of matrix differentials.

In the next section, we introduce the basic online multitask protocol and define the multitask Perceptron algorithm. In order to keep the presentation as simple as possible, and to elucidate the interactive character of the updates, we delay the introduction of kernels until the proof of the mistake bound.

In our initial online protocol, at each time step the multitask learner receives a pair $(\boldsymbol{x}_t, i_t)$, where i_t is the task index for time t and $\boldsymbol{x}_t$ is the instance for task i_t. Note that we view multitask learning as a sequential problem where at each time step the learner works on a single adversarially chosen task, rather than working simultaneously on all tasks (a similar protocol was investigated in [1] in the context of prediction with expert advice). One of the advantages of this approach is that, in most cases, the cost of running our multitask algorithms has a mild dependence on the number K of tasks.

We also remark that linear algorithms for online multitask learning have been studied in [9]. However, these results are sharply different from ours, as they do not depend on task relatedness.

2 Learning protocol and notation

There are K binary classification tasks indexed by $1, \dots, K$. At each time step $t = 1, 2, \dots$ the learner receives a task index $i_t \in \{1, \dots, K\}$ and the corresponding instance vector[1] $\boldsymbol{x}_t \in \mathbb{R}^d$ (which we henceforth assume to be normalized, $\|\boldsymbol{x}_t\| = 1$). Based on this information, the learner outputs a binary prediction $\widehat{y}_t \in \{-1, 1\}$ and then observes the correct label $y_t \in \{-1, 1\}$ for task i_t. As in the standard worst-case online learning model, no assumptions are made on the mechanism generating the sequence $\big(\boldsymbol{x}_t, y_t\big)_{t \ge 1}$. Moreover, similarly to [1], the sequence of tasks i_t is also generated in an adversarial manner.

We compare the learner's performance to that of a reference predictor that is allowed to use a different linear classifier for each of the K tasks. In particular, we compare the learner's mistake count to

$$\inf_{\boldsymbol{u}_1, \dots, \boldsymbol{u}_K \in R^d} \sum_t \ell_t(\boldsymbol{u}_{i_t}) \tag{1}$$

where $\ell_t(\boldsymbol{u}_{i_t}) = \big[1 - y_t\, \boldsymbol{u}_{i_t}^\top \boldsymbol{x}_t\big]_+$ is the hinge loss of the reference linear classifier (or *task vector*) $\boldsymbol{u}_{i_t}$ at time t. Our goal is to design algorithms that make fewer mistakes than K independent learners when the tasks are related, and do not perform much worse than that when the tasks are completely unrelated. In the first part of the paper we use Euclidean distance to measure task relatedness. We say that the K tasks are related if there exist reference task vectors $\boldsymbol{u}_1, \dots, \boldsymbol{u}_K \in \mathbb{R}^d$ having small pairwise distances $\|\boldsymbol{u}_i - \boldsymbol{u}_j\|$, and achieving a small cumulative hinge loss in the sense of (1). More general notions of relatedness are investigated in later sections.

3 The multitask Perceptron algorithm

We first introduce a simple multitask version of the Perceptron algorithm. This algorithm keeps a weight vector for each task and updates all weight vectors at each mistake using the Perceptron rule with different learning rates. More precisely, let $\boldsymbol{w}_{i,t-1}$ be the weight vector associated with task i at the beginning of time step t. If we are forced (by the adversary) to predict on task i_t, and our prediction happens to be wrong, we update $\boldsymbol{w}_{i_t,t-1}$ through the standard additive rule $\boldsymbol{w}_{i_t,t} = \boldsymbol{w}_{i_t,t-1} + \eta y_t\, \boldsymbol{x}_t$ (where $\eta > 0$ is a constant learning rate) but, at the same time, we perform a "half-update" on the remaining $K-1$ Perceptrons, i.e., we set $\boldsymbol{w}_{j,t} = \boldsymbol{w}_{j,t-1} + \frac{\eta}{2} y_t\, \boldsymbol{x}_t$ for each $j \neq i_t$. This rule is based on the simple observation that, in the presence of related tasks, any update step that is good for one Perceptron should also be good for the others. Clearly, this rule keeps the weight vectors $\boldsymbol{w}_{j,t}$, $j = 1, \dots, K$, always close to each other.

The above algorithm is a special case of the *multitask Perceptron algorithm* described below. This more general algorithm updates each weight vector $\boldsymbol{w}_{j,t}$ through a learning rate which is an arbitrary positive definite function of the pair (j, i_t). These learning rates are defined by a $K \times K$ *interaction matrix* A.

The pseudocode for the multitask Perceptron algorithm using a generic interaction matrix A is given in Figure 1. At the beginning of each time step, the counter s stores the mistakes made so far (plus one). The (column) vector $\boldsymbol{\phi}_t \in R^{Kd}$ denotes the *multitask instance* defined by

$$\boldsymbol{\phi}_t^\top = \big(\ \underbrace{0, \dots, 0}_{d(i_t - 1)\text{ times}}\ \ \boldsymbol{x}_t^\top\ \ \underbrace{0, \dots, 0}_{d(K - i_t)\text{ times}}\ \big) \tag{2}$$

where $\boldsymbol{x}_t \in \mathbb{R}^d$ is the instance vector for the current task i_t. (Note that $\|\boldsymbol{\phi}_t\| = 1$ since the instances $\boldsymbol{x}_t$ are normalized.) The weights of the K Perceptrons are maintained in a compound vector $\boldsymbol{w}_s^\top = \big(\boldsymbol{w}_{1,s}^\top, \dots, \boldsymbol{w}_{K,s}^\top\big)$, with $\boldsymbol{w}_{j,s} \in \mathbb{R}^d$ for all j. The algorithm predicts y_t through the sign $\widehat{y}_t$ of the i_t-th Perceptron's margin $\boldsymbol{w}_{s-1}^\top \boldsymbol{\phi}_t = \boldsymbol{w}_{i_t,s-1}^\top \boldsymbol{x}_t$. Then, if prediction and true label disagree, the update rule becomes $\boldsymbol{w}_s = \boldsymbol{w}_{s-1} + y_t \big(A \otimes I_d\big)^{-1} \boldsymbol{\phi}_t$, where $\otimes$ denotes the Kro-

[1]Throughout this paper all vectors are assumed to be column vectors.

```
Parameters: Positive definite $K \times K$ interaction matrix $A$.
Initialization: $\boldsymbol{w}_0 = \mathbf{0} \in \mathbb{R}^{Kd}$, $s = 1$.
At each time $t = 1, 2, \ldots$ do the following:
  1. Observe task number $i_t \in \{1, \ldots, K\}$ and the corresponding instance vector $\boldsymbol{x}_t \in \mathbb{R}^d$;
  2. Build the associated multitask instance $\boldsymbol{\phi}_t \in \mathbb{R}^{Kd}$;
  3. Predict $\widehat{y}_t = \mathrm{SGN}(\boldsymbol{w}_{s-1}^\top \boldsymbol{\phi}_t) \in \{-1, +1\}$;
  4. Get label $y_t \in \{-1, +1\}$;
  5. If $\widehat{y}_t \neq y_t$ then update:
       $\boldsymbol{w}_s = \boldsymbol{w}_{s-1} + y_t (A \otimes I_d)^{-1} \boldsymbol{\phi}_t$
       $s \leftarrow s + 1$ .
```

Figure 1: The multitask Perceptron algorithm.

necker product betweeen matrices[2] and I_d is the $d \times d$ identity matrix. Since $(A \otimes I_d)^{-1} = A^{-1} \otimes I_d$, the above update is equivalent to the K task updates

$$\boldsymbol{w}_{j,s} \leftarrow \boldsymbol{w}_{j,s-1} + y_t \, A^{-1}_{j,i_t} \boldsymbol{x}_t \qquad j = 1, \ldots, K \ .$$

The algorithm is mistake driven, hence $\boldsymbol{w}_{t-1}$ is updated (and is s increased) only when $\widehat{y}_t \neq y_t$.

3.1 Pairwise distance interaction matrix

We now analyze the choice of A that corresponds to the updates $\boldsymbol{w}_{i_t,s} \leftarrow \boldsymbol{w}_{i_t,s-1} + \eta \, y_t \, \boldsymbol{x}_t$ and $\boldsymbol{w}_{j,s} \leftarrow \boldsymbol{w}_{j,s-1} + \frac{\eta}{2} \, y_t \, \boldsymbol{x}_t$ for $j \neq i_t$ with $\eta = 2/(K+1)$. As it can be easily verified, this choice is given by

$$A = \begin{bmatrix} K & -1 & \ldots & -1 \\ -1 & K & \ldots & -1 \\ \ldots & \ldots & \ldots & \ldots \\ -1 & \ldots & \ldots & K \end{bmatrix} \qquad (3)$$

with

$$A^{-1} = \frac{1}{K+1} \begin{bmatrix} 2 & 1 & \ldots & 1 \\ 1 & 2 & \ldots & 1 \\ \ldots & \ldots & \ldots & \ldots \\ 1 & \ldots & \ldots & 2 \end{bmatrix} \ .$$

In order to keep in with the notation just introduced, we equivalently specify an online multitask problem by the sequence $(\boldsymbol{\phi}_1, y_1), (\boldsymbol{\phi}_2, y_2), \cdots \in \mathbb{R}^{dK} \times \{-1, 1\}$ of *multitask examples*, where $\boldsymbol{\phi}_t$ is the multitask instance defined in (2). Moreover, given a sequence of multitask examples and reference task vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K \in \mathbb{R}^d$, we introduce the "compound" reference task vector $\boldsymbol{u}^\top = (\boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_K^\top) \in \mathbb{R}^{Kd}$ and write

$$\ell_t(\boldsymbol{u}) \stackrel{\text{def}}{=} \left[1 - y_t \, \boldsymbol{u}^\top \boldsymbol{\phi}_t\right]_+ = \left[1 - y_t \, \boldsymbol{u}_{i_t}^\top \boldsymbol{x}_t\right]_+ = \ell_t(\boldsymbol{u}_{i_t}) \ .$$

Finally, we use $A_\otimes$ as a shorthand for $A \otimes I_d$, where d is understood from the context. We have the following result.

[2]The Kronecker or direct product between two matrices $A = [a_{i,j}]$ and B of dimension $m \times n$ and $q \times r$, respectively, is the block matrix of dimension $mq \times nr$ whose block on row i and column j is the $q \times r$ matrix $a_{i,j} B$.

Theorem 1 *The number of mistakes m made by the multitask Perceptron algorithm in Figure 1, run with interaction matrix (3) on any finite multitask sequence $(\boldsymbol{\phi}_1, y_1), (\boldsymbol{\phi}_2, y_2), \ldots \in \mathbb{R}^{Kd} \times \{-1, 1\}$, satisfies*

$$m \le \inf_{\boldsymbol{u} \in \mathbb{R}^{Kd}} \left(\sum_{t \in \mathcal{M}} \ell_t(\boldsymbol{u}) + \frac{2(\boldsymbol{u}^\top A_\otimes \boldsymbol{u})}{K+1} + \sqrt{\frac{2(\boldsymbol{u}^\top A_\otimes \boldsymbol{u})}{K+1} \sum_{t \in \mathcal{M}} \ell_t(\boldsymbol{u})} \right) ,$$

where $\mathcal{M}$ is the set of mistaken trial indices, and

$$\boldsymbol{u}^\top A_\otimes \boldsymbol{u} = \sum_{i=1}^K \|\boldsymbol{u}_i\|^2 + \sum_{1 \le i < j \le K} \|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2 \ .$$

Remark Note that when all tasks are equal, that is when $\boldsymbol{u}_1 = \cdots = \boldsymbol{u}_K$, the bound of Theorem 1 becomes the standard Perceptron mistake bound (see, e.g., [7]). In the general case of distinct $\boldsymbol{u}_i$ we have

$$\frac{\boldsymbol{u}^\top A_\otimes \boldsymbol{u}}{K+1} < \sum_{i=1}^K \|\boldsymbol{u}_i\|^2 - \frac{1}{K+1} \sum_{1 \le i,j \le K} \boldsymbol{u}_i^\top \boldsymbol{u}_j \ .$$

The sum of squares $\sum_i \|\boldsymbol{u}_i\|^2$ is the mistake bound one can prove when learning K independent Perceptrons (under linear separability assumptions). On the other hand, highly correlated reference task vectors (i.e., large inner products $\boldsymbol{u}_i^\top \boldsymbol{u}_j$) imply a large negative second term in the right-hand side of the above expression.

Theorem 1 is immediately proven by using the fact that the multitask Perceptron is a specific instance of the kernel Perceptron algorithm [12] using the linear kernel introduced in [11] (see also [15]). As mentioned in the introduction, this kernel is defined as follows: for any positive definite $K \times K$ interaction matrix A introduce the Kd-dimensional reproducing kernel Hilbert space $(\mathbb{R}^{Kd}, \ \langle \cdot, \cdot \rangle_{\mathcal{H}})$ with inner product $\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{\mathcal{H}} = \boldsymbol{u}^\top (A \otimes I_d) \boldsymbol{v}$. Then define the kernel feature map $\psi : \mathbb{R}^d \times \{1, \ldots, K\} \to \mathcal{H}$ such that

$$\psi(\boldsymbol{x}_t, i_t) = (A \otimes I_d)^{-1} \boldsymbol{\phi}_t \ . \qquad (4)$$

The kernel used by the multitask Perceptron is thus defined by

$$\begin{aligned} K\big((\boldsymbol{x}_s, i_s), (\boldsymbol{x}_t, i_t)\big) &= \langle \psi(\boldsymbol{x}_s, i_s), \psi(\boldsymbol{x}_t, i_t) \rangle_{\mathcal{H}} \\ &= \boldsymbol{\phi}_s^\top (A \otimes I_d)^{-1} \boldsymbol{\phi}_t \ . \qquad (5) \end{aligned}$$

Proof of Theorem 1: We use the following version of the kernel Perceptron bound (see, e.g., [7]),

$$\begin{aligned} m \ &\le \ \sum_t \ell_t(f) + \|h\|_{\mathcal{H}}^2 \left(\max_t \|\psi(\boldsymbol{x}_t, i_t)\|_{\mathcal{H}}^2 \right) \\ &+ \ \|h\|_{\mathcal{H}} \sqrt{\left(\max_t \|\psi(\boldsymbol{x}_t, i_t)\|_{\mathcal{H}}^2 \right) \sum_t \ell_t(h)} \end{aligned}$$

where h is any function in the RKHS $\mathcal{H}$ induced by the kernel. Let $A_\otimes = A \otimes I_d$. For the kernel (5) we have $\|\boldsymbol{u}\|_{\mathcal{H}}^2 = \boldsymbol{u}^\top A_\otimes \boldsymbol{u}$ and $\|\psi(\boldsymbol{x}_t, i_t)\|_{\mathcal{H}}^2 = \boldsymbol{\phi}_t^\top A_\otimes^{-1} A_\otimes A_\otimes^{-1} \boldsymbol{\phi}_t = \boldsymbol{\phi}_t^\top A_\otimes^{-1} \boldsymbol{\phi}_t = A^{-1}_{i_t,i_t}$. Observing that $A^{-1}_{i_s,i_s} = 2/(K+1)$ for the matrix A^{-1} defined in (3) concludes the proof. ■

3.2 A more general interaction matrix

In this section we slightly generalize the analysis of the previous section and consider an update rule of the form

$$\boldsymbol{w}_{j,s} = \boldsymbol{w}_{j,s-1} + \begin{cases} \frac{b+K}{(1+b)K}\, y_t\, \boldsymbol{x}_t & \text{if } j = i_t, \\ \frac{b}{(1+b)K}\, y_t\, \boldsymbol{x}_t & \text{otherwise,} \end{cases}$$

where b is a nonnegative parameter. The corresponding interaction matrix is given by

$$A = \frac{1}{K}\begin{bmatrix} a & -b & \dots & -b \\ -b & a & \dots & -b \\ \dots & \dots & \dots & \dots \\ -b & \dots & \dots & a \end{bmatrix}. \qquad (6)$$

with $a = K + b(K-1)$. It is immediate to see that the previous case (3) is recovered by choosing $b = K$. The inverse of (6) is

$$A^{-1} = \frac{1}{(1+b)K}\begin{bmatrix} b+K & b & \dots & b \\ b & b+K & \dots & b \\ \dots & \dots & \dots & \dots \\ b & \dots & \dots & b+K \end{bmatrix}.$$

When (6) is used in the multitask Perceptron algorithm, the proof of Theorem 1 can be adapted to prove the following result.

Corollary 2 *The number of mistakes m made by the multitask Perceptron algorithm in Figure 1, run with interaction matrix (6) on any finite multitask sequence $(\boldsymbol{\phi}_1, y_1), (\boldsymbol{\phi}_2, y_2), \dots \in \mathbb{R}^{Kd} \times \{-1, 1\}$, satisfies*

$$m \le \left(\sum_{t\in\mathcal{M}} \ell_t(\boldsymbol{u}) + \frac{(b+K)}{(1+b)K}\big(\boldsymbol{u}^\top A_\otimes \boldsymbol{u}\big)\right. \\ \left. + \sqrt{\frac{(b+K)}{(1+b)K}\big(\boldsymbol{u}^\top A_\otimes \boldsymbol{u}\big)\sum_{t\in\mathcal{M}} \ell_t(\boldsymbol{u})}\right)$$

for any $\boldsymbol{u} \in \mathbb{R}^{Kd}$, where

$$\boldsymbol{u}^\top A_\otimes \boldsymbol{u} = \sum_{i=1}^{K} \|\boldsymbol{u}_i\|^2 + bK\,\text{VAR}[\boldsymbol{u}], \qquad (7)$$

being $\text{VAR}[\boldsymbol{u}] = \frac{1}{K}\sum_{i=1}^{K}\|\boldsymbol{u}_i - \overline{\boldsymbol{u}}\|^2$ the "variance", of the task vectors, and $\overline{\boldsymbol{u}}$ the centroid $\big(\boldsymbol{u}_1 + \dots + \boldsymbol{u}_K\big)/K$.

It is interesting to investigate how the above bound depends on the trade-off parameter b. The optimal value of b (requiring prior knowledge about the distribution of $\boldsymbol{u}_1, \dots, \boldsymbol{u}_K$) is

$$b = \max\left\{0,\ \sqrt{\frac{K-1}{K}\,\frac{\|\overline{\boldsymbol{u}}\|^2}{\text{VAR}[\boldsymbol{u}]}} - 1\right\}.$$

Thus b grows large as the reference task vectors $\boldsymbol{u}_i$ get close to their centroid $\overline{\boldsymbol{u}}$ (i.e., as all $\boldsymbol{u}_i$ get close to each other). Substituting this choice of b gives

$$\frac{(b+K)}{(1+b)K}\big(\boldsymbol{u}^\top A_\otimes \boldsymbol{u}\big)$$

$$= \begin{cases} \|\boldsymbol{u}_1\|^2 + \dots + \|\boldsymbol{u}_K\|^2 & \text{if } b = 0, \\ \Big(\|\overline{\boldsymbol{u}}\| + \sqrt{K-1}\sqrt{\text{VAR}[\boldsymbol{u}]}\Big)^2 & \text{otherwise.} \end{cases}$$

When the variance $\text{VAR}[\boldsymbol{u}]$ is large (compared to the squared centroid norm $\|\overline{\boldsymbol{u}}\|^2$), then the optimal tuning of b is zero and the interaction matrix becomes the identity matrix, which amounts to running K independent Perceptron algorithms. On the other hand, when the optimal tuning of b is nonzero we learn K reference vectors, achieving a mistake bound equal to that of learning a *single* vector whose length is $\|\overline{\boldsymbol{u}}\|$ plus $\sqrt{K-1}$ times the standard deviation $\sqrt{\text{VAR}[\boldsymbol{u}]}$.

At the other extreme, if the variance $\text{VAR}[\boldsymbol{u}]$ is zero (namely, when all tasks coincide) then the optimal b grows unbounded, and the quadratic term $\frac{(b+K)}{(1+b)K}\big(\boldsymbol{u}^\top A_\otimes \boldsymbol{u}\big)$ tends to the average square norm $\frac{1}{K}\sum_{i=1}^{K}\|\boldsymbol{u}_i\|^2$. In this case the multitask algorithm becomes essentially equivalent to an algorithm that, before learning starts, chooses one task at random and keeps referring all instance vectors $\boldsymbol{x}_t$ to that task (somehow implementing the fact that now the information conveyed by i_t can be disregarded).

3.3 Encoding prior knowledge

We could also pick the interaction matrix A so to encode prior knowledge about tasks. For instance, suppose we know that only certain pairs of tasks are potentially related. We represent this knowledge in a standard way through an undirected graph $G = (V, E)$, where two vertices i and j are connected by an edge if and only if we believe task i and task j are related. A natural choice for A is then $A = I + L$, where $L = [L_{i,j}]_{i,j=1}^{K}$ is the Laplacian of G, defined as

$$L_{i,j} = \begin{cases} d_i & \text{if } i = j, \\ -1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

where d_i is the degree (number of incoming edges) of node i. If we now follow the proof of Theorem 1, which holds for any positive definite matrix A, we obtain the following result.

Corollary 3 *The number of mistakes m made by the multitask Perceptron algorithm in Figure 1, run with interaction matrix $I + L$ on any finite multitask sequence $(\boldsymbol{\phi}_1, y_1), (\boldsymbol{\phi}_2, y_2), \ldots \in \mathbb{R}^{Kd} \times \{-1, 1\}$, satisfies*

$$m \le \inf_{\boldsymbol{u}\in\mathbb{R}^{Kd}}\left(\sum_{t\in\mathcal{M}} \ell_t(\boldsymbol{u}) + c_G\,\boldsymbol{u}^\top\big(I+L\big)_\otimes\boldsymbol{u}\right. \\ \left. + \sqrt{c_G\,\boldsymbol{u}^\top\big(I+L\big)_\otimes\boldsymbol{u}\sum_{t\in\mathcal{M}} \ell_t(\boldsymbol{u})}\right)$$

where

$$\boldsymbol{u}^\top\big(I+L\big)_\otimes\boldsymbol{u} = \sum_{i=1}^{K}\|\boldsymbol{u}_i\|^2 + \sum_{(i,j)\in E}\|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2 \qquad (8)$$

and $c_G = \max_{i=1,\dots,K}\sum_{j=1}^{K}\frac{v_{j,i}^2}{1+\lambda_j}$. Here $0 = \lambda_1 < \lambda_2 \le \dots \le \lambda_K$ are the eigenvalues of the positive semidef-

inite matrix L, and $v_{j,i}$ denotes the i-th component[3] *of the eigenvector $\boldsymbol{v}_j$ of L associated with eigenvalue λ_j.*

Proof: Following the proof of Theorem 1, we just need to bound

$$\max_{i=1,\dots,K} A^{-1}_{i,i} = \max_{i=1,\dots,K} (I+L)^{-1}_{i,i} \ .$$

If $\boldsymbol{v}_1,\dots,\boldsymbol{v}_K$ are the eigenvectors of L, then

$$(I+L)^{-1} = \sum_{j=1}^{K} \frac{\boldsymbol{v}_j\,\boldsymbol{v}_j^\top}{1+\lambda_j}$$

which concludes the proof. ■

Ideally, we would like to have $c_G = \mathcal{O}(1/K)$. Clearly enough, if G is the clique on K vertices we expect to exactly recover the bound of Theorem 1. In fact, we can easily verify that the eigenvector $\boldsymbol{v}_1$ associated with the zero eigenvalue λ_1 is $\big(K^{-1/2},\dots,K^{-1/2}\big)$. Moreover, it is well known that all the remaining eigenvalues are equal to K (see, e.g., [16]). Therefore $c_G = \frac{1}{K} + \left(1-\frac{1}{K}\right)\frac{1}{K+1} = \frac{2}{K+1}$. In the case of more general graphs G, we can bound c_G in terms of the smallest nonzero eigenvalue λ_2,

$$c_G \le \frac{1}{K} + \left(1-\frac{1}{K}\right)\frac{1}{1+\lambda_2} \ .$$

The value of λ_2, known as the algebraic connectivity of G, is 0 only when the graph is disconnected. λ_2 is known for certain families of graphs. For instance, if G is a complete bipartite graph (i.e., if tasks can be divided in two disjoint subsets T_1 and T_2 such that every task in T_1 is related to every task in T_2 and for both $i = 1,2$ no two tasks in T_i are related), then it is known that $\lambda_2 = \min\big\{|T_1|,|T_2|\big\}$. We refer the reader to, e.g., [16] for further examples.

The advantage of using a graph G with significantly fewer edges than the clique is that the sum of pairwise distances in (8) will contain less than $K(K-1)$ terms. On the other hand, this reduction is balanced by a larger coefficient c_G in front of $\boldsymbol{u}^\top\big(I+L\big)_\otimes\boldsymbol{u}$. This coefficient, in general, is related to the total number of edges in the graph (observe that the trace of L is exactly twice this total number).

4 The second-order extension

In this section we consider the second-order kernel Perceptron algorithm of [7] with the multitask kernel (5). The algorithm, which is described in Figure 2, maintains in its internal state a matrix S (initialized to the empty matrix) and a multitask Perceptron weight vector $\boldsymbol{v}$ (initialized to the zero vector). Just like in Figure 1, we use the subscript s to denote the current number of mistakes plus one. Note that we have exploited the linearity of the kernel (5) to simplify the description of the algorithm. In particular, letting $A_\otimes = A\otimes I_d$, we have repeatedly used the fact that

$$\begin{aligned}\big\langle\psi(\boldsymbol{x}_s,i_s),\psi(\boldsymbol{x}_t,i_t)\big\rangle_{\mathcal{H}} &= \boldsymbol{\phi}_s^\top A_\otimes^{-1}\boldsymbol{\phi}_t\\ &= \Big(A_\otimes^{-1/2}\boldsymbol{\phi}_s\Big)^\top\Big(A_\otimes^{-1/2}\boldsymbol{\phi}_t\Big)\\ &= \widetilde{\boldsymbol{\phi}}_s^\top\,\widetilde{\boldsymbol{\phi}}_t \ ,\end{aligned}$$

[3]Note that the orthonormality of the eigenvectors imply $v_{1,i}^2 + \dots + v_{K,i}^2 = 1$ for all i.

Parameters: Positive definite $K\times K$ interaction matrix A.
Initialization: $S_0 = \emptyset,\ \boldsymbol{v}_0 = \boldsymbol{0}\in\mathbb{R}^{Kd},\ s=1$.
At each time $t = 1,2,\dots$ do the following:

1. Observe task number $i_t \in \{1,\dots,K\}$ and the corresponding instance vector $\boldsymbol{x}_t\in\mathbb{R}^d$;
2. Build the associated multitask instance $\boldsymbol{\phi}_t\in\mathbb{R}^{Kd}$ and compute $\widetilde{\boldsymbol{\phi}}_t = \big(A\otimes I_d\big)^{-1/2}\boldsymbol{\phi}_t$;
3. Predict $\widehat{y}_t = \textsc{sgn}\big(\boldsymbol{w}_{s-1}^\top\widetilde{\boldsymbol{\phi}}_t\big)\in\{-1,+1\}$,
   where $\boldsymbol{w}_{s-1} = \Big(I + S_{s-1}S_{s-1}^\top + \widetilde{\boldsymbol{\phi}}_t\widetilde{\boldsymbol{\phi}}_t^\top\Big)^{-1}\boldsymbol{v}_{s-1}$;
4. Get the label $y_t\in\{-1,1\}$;
5. If $\widehat{y}_t \ne y_t$ then update:
   $$\boldsymbol{v}_s = \boldsymbol{v}_{s-1} + y_t\widetilde{\boldsymbol{\phi}}_t\ ,\quad S_s = \Big[\,S_{s-1}\Big|\widetilde{\boldsymbol{\phi}}_t\,\Big]\ ,\quad s\leftarrow s+1\ .$$

Figure 2: The second-order multitask Perceptron algorithm.

where ψ is the kernel feature map (4). The algorithm computes a tentative (inverse) matrix

$$\Big(I + S_{s-1}S_{s-1}^\top + \widetilde{\boldsymbol{\phi}}_t\widetilde{\boldsymbol{\phi}}_t^\top\Big)^{-1} \ .$$

Such a matrix is combined with the current Perceptron vector $\boldsymbol{v}_{s-1}$ to predict the label y_t. If prediction $\widehat{y}_t$ and label y_t disagree both $\boldsymbol{v}$ and S get updated (no update takes place otherwise). In particular, the new matrix S_s is augmented by padding with the current vector $\widetilde{\boldsymbol{\phi}}_t$. Since supports are shared, the computational cost of an update is not significantly larger than that for learning a single-task (see Section 4.1).

Theorem 4 *The number of mistakes m made by the second-order multitask Perceptron of Figure 2, run with any positive definite interaction matrix A, on any finite multitask sequence $(\boldsymbol{\phi}_1,y_1),(\boldsymbol{\phi}_2,y_2),\ldots\in\mathbb{R}^{Kd}\times\{-1,1\}$, satisfies, for all $\boldsymbol{u}\in\mathbb{R}^{Kd}$,*

$$m \le \sum_{t\in\mathcal{M}}\ell_t(\boldsymbol{u}) + \sqrt{\left(\boldsymbol{u}^\top\big(A\otimes I_d\big)\boldsymbol{u} + \sum_{t\in\mathcal{M}}\big(\boldsymbol{u}_{i_t}^\top\boldsymbol{x}_t\big)^2\right)\sum_{j=1}^{m}\ln(1+\lambda_j)}$$

where $\mathcal{M}$ is the sequence of mistaken trial indices and $\lambda_1,\dots,\lambda_m$ are the eigenvalues of the $m\times m$ matrix of elements $\boldsymbol{x}_s^\top A^{-1}_{i_s,i_t}\boldsymbol{x}_t$, where $s,t\in\mathcal{M}$.

Proof: Recall the mistake bound for the second-order kernel Perceptron algorithm [7]:

$$m \le \sqrt{\left(\|h\|_{\mathcal{H}}^2 + \sum_{t\in\mathcal{M}} h(\boldsymbol{x}_t)^2\right)\sum_{j=1}^{m}\ln(1+\lambda_j)}$$

where $\lambda_1,\dots,\lambda_m$ are the eigenvalues of the $m\times m$ kernel Gram (sub)matrix including only time steps in $\mathcal{M}$. When the kernel is (5) with feature map f we have $\|\boldsymbol{u}\|_{\mathcal{H}}^2 = \boldsymbol{u}^\top A_\otimes \boldsymbol{u}^\top$ and $\langle \boldsymbol{u}^\top, \psi(\boldsymbol{x}_t, i_t)\rangle^2 = \left(\boldsymbol{u}^\top A_\otimes A_\otimes^{-1} \boldsymbol{\phi}_t\right)^2 = \left(\boldsymbol{u}_{i_t}^\top \boldsymbol{x}_t\right)^2$. Finally, the kernel Gram matrix is $K\big(\psi(\boldsymbol{x}_s, i_s), \psi(\boldsymbol{x}_t, i_t)\big) = \boldsymbol{\phi}_s^\top A_\otimes^{-1} \boldsymbol{\phi}_t = \boldsymbol{x}_s^\top A_{i_s,i_t}^{-1} \boldsymbol{x}_t$. This concludes the proof. ■

Again, this bound should be compared to the one obtained when learning K independent tasks. As in the first-order algorithm, we have the complexity term $\boldsymbol{u}^\top \left(A \otimes I_d\right)\boldsymbol{u}$. In this case, however, the interaction matrix A also plays a role in the scale of the eigenvalues of the resulting multitask Gram matrix. Roughly speaking, we gain a factor K from $\boldsymbol{u}^\top A_\otimes^{-1} \boldsymbol{u}$ (according to the arguments in Section 3). In addition, however, we gain a further factor K, since the trace of the multitask Gram matrix $\left[\boldsymbol{\phi}_s^\top A_\otimes^{-1} \boldsymbol{\phi}_t\right]_{s,t\in\mathcal{M}} = \left[\boldsymbol{x}_s^\top A_{i_s,i_t}^{-1} \boldsymbol{x}_t\right]_{s,t\in\mathcal{M}}$ is about $1/K$ times the trace of the original Gram matrix $\left[\boldsymbol{x}_s^\top \boldsymbol{x}_t\right]_{s,t\in\mathcal{M}}$. Since both factors are under the square root, the resulting gain over the K independent task bound is about K.

4.1 Implementation in dual variables

It is easy to see that the second-order multitask Perceptron can be run in dual variables by maintaining K classifiers that share the same set of support vectors. This allows an efficient implementation that does not impose any significant overhead with respect to the corresponding single-task version. Specifically, given some interaction matrix A, the margin at time t is computed as (see [7, Theorem 3.3])

$$\begin{aligned}\boldsymbol{w}_{s-1}^\top \widetilde{\boldsymbol{\phi}}_t &= \boldsymbol{v}_{s-1}^\top \Big(I + S_{s-1} S_{s-1}^\top + \widetilde{\boldsymbol{\phi}}_t \widetilde{\boldsymbol{\phi}}_t^\top\Big)^{-1} \widetilde{\boldsymbol{\phi}}_t \\ &= \boldsymbol{y}_s^\top \Big(I + S_s^\top S_s\Big)^{-1} S_s^\top \widetilde{\boldsymbol{\phi}}_t\ , \end{aligned} \tag{9}$$

where $\boldsymbol{y}_s$ is the s-dimensional vector whose first $s-1$ components are the labels y_i where the algorithm has made a mistake up to time $t-1$, and the last component is 0.

First, note that replacing $I + S_s^\top S_s$ with $I + S_{s-1}^\top S_{s-1}$ in (9) does not change the sign of the prediction. The margin at time t can then be computed by calculating the scalar product between $S_s^\top \widetilde{\boldsymbol{\phi}}_t$ and $\boldsymbol{y}_s^\top \left(I + S_{s-1}^\top S_{s-1}\right)^{-1}$. Now, each entry of the vector $S_s^\top \widetilde{\boldsymbol{\phi}}_t$ is of the form $A_{j,i_t}^{-1} \boldsymbol{x}_j^\top \boldsymbol{x}_t$, and thus computing $S_s^\top \widetilde{\boldsymbol{\phi}}_t$ requires $\mathcal{O}(s)$ inner products so that, overall, the prediction step requires $\mathcal{O}(s)$ scalar multiplications and $\mathcal{O}(s)$ inner products (independent of the number of tasks K).

On the other hand, the update step involves the computation of the vector $\boldsymbol{y}_s^\top \left(I + S_s^\top S_s\right)^{-1}$. For the matrix update we can write

$$I_s + S_s^\top S_s = \left[\begin{array}{cc} I_{s-1} + S_{s-1}^\top S_{s-1} & S_{s-1}^\top \widetilde{\boldsymbol{\phi}}_t \\ \widetilde{\boldsymbol{\phi}}_t^\top S_{s-1} & 1 + \widetilde{\boldsymbol{\phi}}_t^\top \widetilde{\boldsymbol{\phi}}_t \end{array}\right].$$

Using standard facts about the inverse of partitioned matrices (e.g., [17, Ch. 0]), one can see that the inverse of matrix $I_s + S_s^\top S_s$ can be computed from the inverse of $I_{s-1} + S_{s-1}^\top S_{s-1}$ with $\mathcal{O}(s)$ extra inner products (again, independent of K) and $\mathcal{O}(s^2)$ additional scalar multiplications.

5 The p-norm extension

We now extend our multitask results to the p-norm Perceptron algorithm of [14, 13]. As before, when the tasks are all equal we want to recover the bound of the single-task algorithm, and when the task vectors are different we want the mistake bound to increase according to a function that penalizes task diversity according to their p-norm distance.

We develop our p-norm multitask analysis for the specific choices of $p = 2\ln d$ (or $p = 2\ln K$ when $d \le K$) and for the pairwise distance matrix (3). It is well known that for $p = 2\ln d$ the mistake bound of the single-task p-norm Perceptron is essentially equivalent to the one of the zero-threshold Winnow algorithm of [22]. We now see that this property is preserved in the multitask extension.

We start with the following slightly more general algorithm based on arbitrary norms. Later, we specialize it to p-norms. The *quasi-additive multitask algorithm* of [20, 14] is defined for any norm $\|\cdot\|$ over $\mathbb{R}^{Kd}$. Initially, $\boldsymbol{w}_0 = \boldsymbol{0} \in \mathbb{R}^{Kd}$. If $s-1$ mistakes have been made in the first $t-1$ time steps, then the prediction at time t is $\mathrm{SGN}\big(\boldsymbol{w}_{s-1}^\top \boldsymbol{\phi}_t\big)$. If a mistake occurs at time t, then $\boldsymbol{w}_{s-1}$ is updated with the rule $\boldsymbol{w}_s = \nabla \frac{1}{2}\|\boldsymbol{v}_s\|^2$, where the *primal weight* $\boldsymbol{v}_s \in \mathbb{R}^{Kd}$ is updated using the multitask Perceptron rule, $\boldsymbol{v}_0 = \boldsymbol{0} \in \mathbb{R}^{Kd}$ and $\boldsymbol{v}_s = \boldsymbol{v}_{s-1} + y_t A_\otimes^{-1} \boldsymbol{\phi}_t$ for an arbitrary positive definite interaction matrix A.

This can be analyzed using the following technique (see [8] for details). Let $\boldsymbol{v}_m$ be the primal weight after any number m of mistakes. Then, by Taylor expanding $\frac{1}{2}\|\boldsymbol{v}_s\|^2$ around $\boldsymbol{v}_{s-1}$ for each $s = 1, \ldots, m$, and using the fact $y_t\, \boldsymbol{w}_{s-1}^\top \boldsymbol{\phi}_t \le 0$ whenever a mistake occurs at step t, we get

$$\frac{1}{2}\|\boldsymbol{v}_m\|^2 \le \sum_{s=1}^{m} D\left(\boldsymbol{v}_s \| \boldsymbol{v}_{s-1}\right) \tag{10}$$

$D\left(\boldsymbol{v}_s \| \boldsymbol{v}_{s-1}\right) = \frac{1}{2}\big(\|\boldsymbol{v}_s\|^2 - \|\boldsymbol{v}_{s-1}\|^2\big) - y_t\, \boldsymbol{w}_{s-1}^\top \boldsymbol{x}_t$ is a so-called Bregman divergence; i.e., the error term in the first-order Taylor expansion of $\frac{1}{2}\|\cdot\|^2$ around vector $\boldsymbol{v}_{s-1}$, at vector $\boldsymbol{v}_s$.

Fix any $\boldsymbol{u} \in \mathbb{R}^{Kd}$. Using the convex inequality for norms $\boldsymbol{u}^\top \boldsymbol{v} \le \|\boldsymbol{u}\|\,\|\boldsymbol{v}\|_*$ where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$ (see, e.g., [25, page 131]), and using the fact $\boldsymbol{u}^\top A_\otimes \boldsymbol{v}_s = \boldsymbol{u}^\top A_\otimes \boldsymbol{v}_{s-1} + y_t \boldsymbol{u}^\top \boldsymbol{\phi}_t \ge \boldsymbol{u}^\top A_\otimes \boldsymbol{v}_{s-1} + 1 - \ell_t(\boldsymbol{u})$, one then obtains

$$\|\boldsymbol{v}_m\| \ge \frac{\boldsymbol{u}^\top A_\otimes \boldsymbol{v}_m}{\|A_\otimes \boldsymbol{u}\|_*} \ge \frac{m - \sum_t \ell_t(\boldsymbol{u})}{\|A_\otimes \boldsymbol{u}\|_*}\ . \tag{11}$$

Combining (10) with (11) and solving for m gives

$$m \le \sum_{t\in\mathcal{M}} \ell_t(\boldsymbol{u}) + \|A_\otimes \boldsymbol{u}\|_* \sqrt{2\sum_{s=1}^{m} D\left(\boldsymbol{v}_s \| \boldsymbol{v}_{s-1}\right)}\ . \tag{12}$$

We obtain our multitask version of the p-norm Perceptron when $\|\boldsymbol{u}\| = \|\boldsymbol{u}\|_p = \left(|u_1|^p + |u_2|^p + \cdots\right)^{1/p}$. In particular, we focus our analysis on the choice $p = 2\ln\max\{K, d\}$, which gives mistake bounds in the dual norms $\|\boldsymbol{u}\|_1$ and $\|\boldsymbol{x}_t\|_\infty$, and on the pairwise distance matrix (3).

Using the analysis in [8] we obtain, for $t_s = t$,

$$D\left(\boldsymbol{v}_s \| \boldsymbol{v}_{s-1}\right) \le \frac{p-1}{2}\left\|A_\otimes^{-1}\phi_t\right\|_p^2 = \frac{p-1}{2}\|\boldsymbol{x}_t\|_p^2 \left\|A_{\downarrow i_t}^{-1}\right\|_p^2$$

where $A^{-1}_{\downarrow i_t}$ is the i_t-th column of A^{-1}. If we now use $p = 2\ln\max\{K,d\}$, then $\|\boldsymbol{x}_t\|_p^2 \le e\,\|\boldsymbol{x}_t\|_\infty^2$ and

$$\left\|A^{-1}_{\downarrow i_t}\right\|_p^2 \le e\,\left\|A^{-1}_{\downarrow i_t}\right\|_\infty^2 = e\left(A^{-1}_{i_t,i_t}\right)^2 = \frac{4\,e}{(K+1)^2}\,.$$

We now turn to the computation of the dual norm $\|A_\otimes \boldsymbol{u}\|_q$, where $q = p/(p-1)$ is the dual coefficient of p. We find that

$$\|A_\otimes \boldsymbol{u}\|_q^2 \le \|A_\otimes \boldsymbol{u}\|_1^2 = \left(\sum_{i=1}^K \left\|\boldsymbol{u}_i + \sum_{j\ne i}(\boldsymbol{u}_i - \boldsymbol{u}_j)\right\|_1\right)^2 .$$

Plugging back into (12) gives the following theorem.

Theorem 5 *The number of mistakes m made by the p-norm multitask Perceptron, run with the pairwise distance matrix (3) and $p = 2\ln\max\{K,d\}$, on any finite multitask sequence $(\boldsymbol{\phi}_1, y_1), (\boldsymbol{\phi}_2, y_2), \ldots \in \mathbb{R}^{Kd} \times \{-1,1\}$, satisfies, for all $\boldsymbol{u} \in \mathbb{R}^{Kd}$,*

$$m \le \sum_{t\in\mathcal{M}} \ell_t(\boldsymbol{u}) + H + \sqrt{2H\sum_{t\in\mathcal{M}} \ell_t(\boldsymbol{u})}$$

where H is equal to

$$\frac{4\,e^2\ln\max\{K,d\}}{(K+1)^2}\,X_\infty^2\left(\sum_{i=1}^K \left\|\boldsymbol{u}_i + \sum_{j\ne i}(\boldsymbol{u}_i - \boldsymbol{u}_j)\right\|_1\right)^2 .$$

and $X_\infty = \max_{t\in\mathcal{M}} \|\boldsymbol{x}_t\|_\infty$.

Remark When all tasks are equal, $\boldsymbol{u}_1 = \cdots = \boldsymbol{u}_K$, the coefficient H in the bound of Theorem 5 becomes

$$\left(4\,e^2\ln\max\{K,d\}\right)\left(\max_t \|\boldsymbol{x}_t\|_\infty\right)^2 \|\boldsymbol{u}_i\|_1^2\ .$$

If $K \le d$ this bound is equivalent (apart from constant factors) to the mistake bound for the single-task zero-threshold Winnow algorithm of [22].

6 Learning tasks in heterogeneous spaces

In this section we slightly deviate from the approach followed so far. We consider the case when the K task vectors $\boldsymbol{u}_i$ may live in different spaces: $\boldsymbol{u}_i \in \mathbb{R}^{d_i}$, $i = 1,\ldots,K$. This is a plausible assumption when attributes associated with different tasks have a completely different meaning. In such a case, the correlation among tasks is naturally measured through the task margins $\boldsymbol{u}_i^\top \boldsymbol{x}$ (or *views*) —see, e.g., the previous work of [6, 27] for a similar approach in the context of semi-supervised learning. In order to allow for the views to interact in a meaningful way, we slightly modify the learning protocol of Section 2. We now assume that, at each time t, we receive the adversarial choice of task i_t together with *all* instance vectors $\boldsymbol{x}_{i,t} \in \mathbb{R}^{d_i}$ for $i = 1,\ldots,K$. The co-regularization terms motivating our algorithm are proportional to the distance between the margin $\boldsymbol{u}_{i_t}^\top \boldsymbol{x}_{i_t,t}$ of task i_t and the average margin $\frac{1}{k}\sum_{j=1}^k \boldsymbol{u}_j^\top \boldsymbol{x}_{j,t}$ of all tasks:

$$b\,K\left(\boldsymbol{u}_{i_t}^\top \boldsymbol{x}_{i_t,t} - \frac{1}{K}\sum_{j=1}^K \boldsymbol{u}_j^\top \boldsymbol{x}_{j,t}\right)^2 , \tag{13}$$

where b is a positive constant. We add the above terms up to time t, resulting in a cumulative regularization term

$$b\,K\sum_{s\in\mathcal{M}_t}\left(\boldsymbol{u}_{i_s}^\top \boldsymbol{x}_{i_s,s} - \frac{1}{K}\sum_{j=1}^K \boldsymbol{u}_j^\top \boldsymbol{x}_{j,s}\right)^2 ,$$

where $\mathcal{M}_t$ is the set of mistaken trials up to time t. Thus, in trial t our prior knowledge about task relatedness is encoded as a (positive) correlation among the K margin sequences $(\boldsymbol{u}_j^\top \boldsymbol{x}_{j,1}, \boldsymbol{u}_j^\top \boldsymbol{x}_{j,2}, \ldots, \boldsymbol{u}_j^\top \boldsymbol{x}_{j,t})$, $j = 1,\ldots,K$.

The algorithm of Figure 3 is a natural multitask predictor operating with the above view-based regularization criterion. We call this algorithm the multiview-based multitask Perceptron algorithm, or MMPERC for brevity. MMPERC can be viewed as a variant of the second-order Perceptron using the cumulative covariance matrix of the past margin vectors in order to suitably transform instances.

MMPERC has a constant tradeoff parameter $b > 0$ (playing the same role as the one in Section 3.2), and maintains in its internal state a multitask matrix A (initialized to the identity matrix I) and a Perceptron multitask weight vector $\boldsymbol{v}$ (initialized to the zero vector). The subscript s plays the same role as in the previous algorithms. Unlike the algorithms in previous sections, MMPERC observes the task number i_t and the multitask instance $\boldsymbol{\Phi}_t^\top = \left(\boldsymbol{x}_{1,t}^\top, \boldsymbol{x}_{2,t}^\top, \ldots, \boldsymbol{x}_{K,t}^\top\right)$ made up of the instance vectors of all tasks. Then MMPERC computes a tentative matrix A'_t to be used for prediction. Matrix A'_t is obtained by adding the rank-one positive semidefinite matrix M_t to the previous matrix A_{s-1}. Here $\boldsymbol{\phi}_{j,t}$ is the $(d_1 + \cdots + d_K)$-dimensional vector

$$\boldsymbol{\phi}_{j,t}^\top = (\quad \underbrace{0,\ldots,0}_{d_1+\cdots+d_{j-1}\text{ times}} \quad \boldsymbol{x}_{j,t}^\top \quad \underbrace{0,\ldots,0}_{d_{j+1}+\cdots+d_K\text{ times}} \quad)$$

for $j = 1,\ldots,K$. Observe that M_t has been set so as to make the quadratic form $\boldsymbol{u}^\top M_t \boldsymbol{u}$ coincide with the regularization term (13). Similarly to the algorithms of previous sections, the tentative matrix A'_t and the current Perceptron vector $\boldsymbol{v}_{s-1}$ are used for predicting the true label y_t. If prediction $\widehat{y}_t$ and label y_t disagree both $\boldsymbol{v}$ and A get updated. In particular, A_s is set to the tentative matrix A'_t.

In this protocol we call example the triple $(i_t, \boldsymbol{\Phi}_t, y_t)$. Like the results contained in the previous sections, our analysis will provide a multitask bound on the number of prediction mistakes which is comparable to the one obtained by a single task plus a penalization term due to task relatedness. However, though this algorithm is a second-order prediction method, we only give a first-order analysis that disregards the eigenstructure of the data. This is due to the technical difficulty of handling a time-varying matrix A that in trial t includes all instance vectors $\boldsymbol{x}_{1,t}, \ldots \boldsymbol{x}_{K,t}$.

Theorem 6 *The number of mistakes m made by the algorithm in Figure 3, run on any multitask sequence $(i_1, \boldsymbol{\Phi}_1, y_1), (i_2, \boldsymbol{\Phi}_2, y_2), \ldots$ satisfies, for all $\boldsymbol{u}^\top =$*

Parameters: $b > 0$.
Initialization: $A_0 = I$, $\boldsymbol{v}_0 = \boldsymbol{0} \in \mathbb{R}^{d_1+\cdots+d_K}$, $s = 1$.
At each time $t = 1, 2, \ldots$ do the following:

1. Observe task number $i_t \in \{1, \ldots, K\}$;
2. Observe multitask instance vector
$$\boldsymbol{\Phi}_t^\top = \left(\boldsymbol{x}_{1,t}^\top, \ldots, \boldsymbol{x}_{K,t}^\top\right) \in \mathbb{R}^{d_1+\cdots+d_K};$$
3. Build the associated multitask instance $\boldsymbol{\phi}_{i_t,t}$;
4. Set $A'_t = A_{s-1} + M_t$ where
$$M_t = b\,K\left(\boldsymbol{\phi}_{i_t,t} - \frac{\boldsymbol{\Phi}_t}{K}\right)\left(\boldsymbol{\phi}_{i_t,t} - \frac{\boldsymbol{\Phi}_t}{K}\right)^\top;$$
5. Predict $\widehat{y}_t = \text{SGN}\big(\boldsymbol{v}_{s-1}^\top (A'_t)^{-1}\boldsymbol{\phi}_{i_t,t}\big) \in \{-1, +1\}$;
6. Get label $y_t \in \{-1, +1\}$;
7. If $\widehat{y}_t \neq y_t$ then update:
$$\boldsymbol{v}_s = \boldsymbol{v}_{s-1} + y_t\,\boldsymbol{\phi}_{i_t,t}\ , \quad A_s = A'_t\ , \quad s \leftarrow s+1\ .$$

Figure 3: The multiview-based multitask Perceptron algorithm (MMPERC).

$(\boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_K^\top)$,

$$m \le \sum_{t\in\mathcal{M}} \ell_t(\boldsymbol{u}) + \frac{K(b+1)-b}{bK(K-1)+K}\big(\boldsymbol{u}^\top A_m \boldsymbol{u}\big) + \sqrt{\frac{K(b+1)-b}{bK(K-1)+K}\big(\boldsymbol{u}^\top A_m \boldsymbol{u}\big)\sum_{t\in\mathcal{M}}\ell_t(\boldsymbol{u})}\ ,$$

where

$$\boldsymbol{u}^\top A_m \boldsymbol{u} = \sum_{i=1}^K ||\boldsymbol{u}_i||^2 + bK\sum_{t\in\mathcal{M}}\left(\boldsymbol{u}_{i_t}^\top \boldsymbol{x}_{i_t,t} - \frac{1}{K}\sum_{j=1}^K \boldsymbol{u}_j^\top \boldsymbol{x}_{j,t}\right)^2\ ,$$

being $\mathcal{M}$ the set of mistaken trials.

Remark It is the factor

$$\frac{K(b+1)-b}{bK(K-1)+K}\big(\boldsymbol{u}^\top A_m \boldsymbol{u}\big) \tag{14}$$

that quantifies the relatedness among tasks (the leading constant $b\,K$ in the second term of $\boldsymbol{u}^\top A_m, \boldsymbol{u}$ is needed for scaling purposes). Note that the notion of relatedness provided by $\boldsymbol{u}^\top A_m\,\boldsymbol{u}$ is analogous to the one used in Section 3.2. As suggested in Section 3.3, other measures of similarity are possible.

The parameter b allows for a limited trade-off between $\sum_{i=1}^K ||\boldsymbol{u}_i||^2$ and the cumulative "margin deviation"

$$\sum_{t\in\mathcal{M}}^m\left(\boldsymbol{u}_{i_t}^\top \boldsymbol{x}_{i_t,t} - \frac{1}{K}\sum_{j=1}^K \boldsymbol{u}_j^\top \boldsymbol{x}_{j,t}\right)^2 . \tag{15}$$

In particular, setting $b = 0$ corresponds to running K independent (first-order) Perceptron algorithms (no task relatedness), while letting b go to infinity is optimal only when each one of the margin deviation terms in (15) is zero (maximal task relatedness). Notice that setting $b = 1$ gives

$$(14) = \frac{2K-1}{K^2}\sum_{i=1}^K ||\boldsymbol{u}_i||^2 + \frac{2K-1}{K}\sum_{t\in\mathcal{M}}^m\left(\boldsymbol{u}_{i_t}^\top \boldsymbol{x}_{i_t,t} - \frac{1}{K}\sum_{j=1}^K \boldsymbol{u}_j^\top \boldsymbol{x}_{j,t}\right)^2$$

yielding a gain of K whenever the K tasks are significantly related, as measured by (15).

Proof of Theorem 6: Although the general structure of the analysis is based on the second-order Perceptron proof, we use the special properties of $I + M_t$ in order to compute the contribution of K and b to the final bound.

Let $t = t_s$ be the time step when the s-th mistake occurs. We write

$$\begin{aligned}
\boldsymbol{v}_s^\top A_s^{-1}\boldsymbol{v}_s &= (\boldsymbol{v}_{s-1} + y_t\boldsymbol{\phi}_{i_t,t})^\top A_s^{-1}(\boldsymbol{v}_{s-1} + y_t\boldsymbol{\phi}_{i_t,t}) \\
&\quad \text{(from the update rule in Figure 3)} \\
&= \boldsymbol{v}_{s-1}A_s^{-1}\boldsymbol{v}_{s-1} + 2y_t\boldsymbol{v}_{s-1}^\top A_s^{-1}\boldsymbol{\phi}_{i_t,t} \\
&\quad + \boldsymbol{\phi}_{i_t,t}^\top A_s^{-1}\boldsymbol{\phi}_{i_t,t} \\
&\le \boldsymbol{v}_{s-1}A_s^{-1}\boldsymbol{v}_{s-1} + \boldsymbol{\phi}_{i_t,t}^\top A_s^{-1}\boldsymbol{\phi}_{i_t,t} \\
&\le \boldsymbol{v}_{s-1}A_{s-1}^{-1}\boldsymbol{v}_{s-1} + \boldsymbol{\phi}_{i_t,t}^\top (I + M_t)^{-1}\boldsymbol{\phi}_{i_t,t}\ .
\end{aligned} \tag{16}$$

In order to prove the first inequality, note that on the s-th mistaken trial $A_s = A'_t$ and $y_t\boldsymbol{v}_{s-1}^\top (A'{}_t)^{-1}\boldsymbol{\phi}_{i_t,t} \le 0$. In order to prove the second inequality note that $A_s - A_{s-1}$ and $A_s - (I + M_t)$ are both positive semidefinite.

We now focus on computing the quadratic form $\boldsymbol{\phi}_{i_t,t}^\top (I + M_t)^{-1}\boldsymbol{\phi}_{i_t,t}$. Recall that $\boldsymbol{\Phi}_t = \sum_{j=1}^K \boldsymbol{\phi}_{j,t}$ is the sum of the orthonormal vectors $\boldsymbol{\phi}_{1,t}, \ldots, \boldsymbol{\phi}_{i_t,t}, \ldots, \boldsymbol{\phi}_{K,t}$. Thus, from the very definition of M_t in Figure 3 it is easy to verify that

$$(I + M_t)\,\boldsymbol{\phi}_{i_t,t} = (b(K-1)+1)\,\boldsymbol{\phi}_{i_t,t} - \frac{b(K-1)}{K}\boldsymbol{\Phi}_t\ . \tag{17}$$

Also, since $M_t\,\boldsymbol{\Phi}_t = 0$, we have $(I + M_t)\boldsymbol{\Phi}_t = \boldsymbol{\Phi}_t$, and thus $(I + M_t)^{-1}\boldsymbol{\Phi}_t = \boldsymbol{\Phi}_t$. Hence (17) allows us to write

$$\boldsymbol{\phi}_{i_t,t} = (b(K-1)+1)\,(I + M_t)^{-1}\,\boldsymbol{\phi}_{i_t,t} - \frac{b(K-1)}{K}\,\boldsymbol{\Phi}_t\ .$$

Taking the inner product of both sides with $\boldsymbol{\phi}_{i_t,t}$, and solving for $\boldsymbol{\phi}_{i_t,t}(I + M_t)^{-1}\boldsymbol{\phi}_{i_t,t}$ yields

$$\boldsymbol{\phi}_{i_t,t}(I + M_t)^{-1}\boldsymbol{\phi}_{i_t,t} = \frac{K(b+1)-b}{bK(K-1)+K}\ .$$

Substituting this into (16), recalling that $\boldsymbol{v}_0 = \boldsymbol{0}$, and summing over $s = 1, \ldots, m$ we obtain

$$\boldsymbol{v}_m^\top A_m^{-1}\boldsymbol{v}_m \le m\,\frac{K(b+1)-b}{bK(K-1)+K}\ .$$

A lower bound on the left-hand side can be obtained in the standard way (details omitted),

$$\sqrt{\boldsymbol{v}_m^\top A_m^{-1}\boldsymbol{v}_m} \ge \frac{m - \sum_{t\in\mathcal{M}}\ell_t(\boldsymbol{u})}{\left\|A_m^{1/2}\boldsymbol{u}\right\|}\ .$$

Thus we get $\boldsymbol{v}_m^\top A_m^{-1} \boldsymbol{v}_m \geq \frac{\left(m - \sum_{t\in\mathcal{M}} \ell_t(\boldsymbol{u})\right)^2}{\boldsymbol{u}^\top A_m \boldsymbol{u}}$. Finally, recall that by construction

$$\boldsymbol{u}^\top A_m \boldsymbol{u} = \sum_{i=1}^K ||\boldsymbol{u}_i||^2 + b\,K \sum_{t\in\mathcal{M}} \left(\boldsymbol{u}_{i_t}^\top \boldsymbol{x}_{i_t,t} - \frac{1}{K}\sum_{j=1}^K \boldsymbol{u}_j^\top \boldsymbol{x}_{j,t} \right)^2 .$$

Putting together and solving for m gives the desired bound. ■

6.1 Implementation in dual variables

Like the second-order multitask Perceptron algorithm, also MMPERC can be formulated in dual variables. Due to the need to handle K instance vectors at a time, the implementation we sketch below has an extra linear dependence on K, as compared to the one in Section 4.1.

Let $t = t_s$ be the trial when the s-th mistake occurs, $\boldsymbol{z}_r$ be the vector $\boldsymbol{z}_r = \sqrt{bK}(\boldsymbol{\phi}_{i_r,r} - \boldsymbol{\Phi}_r/K)$, S_s be the matrix whose columns are the vectors $\boldsymbol{\phi}_{i_r,r}$ corresponding to mistaken time steps r up to time t, and Z_s be the matrix whose columns are the vectors $\boldsymbol{z}_r$ corresponding to mistaken time steps r up to time t. It is easy to verify that $A_s = I + Z_s Z_s^\top$ and $\boldsymbol{v}_{s-1} = S_s\,\boldsymbol{y}_s$ where $\boldsymbol{y}_s$ is as in Section 4.1. From the inversion formula (e.g., [17, Ch. 0])

$$(I + Z_s Z_s^\top)^{-1} = I - Z_s(I + Z_s^\top Z_s)^{-1} Z_s^\top$$

we see that the margin $\boldsymbol{v}_{s-1}^\top (A_t')^{-1} \boldsymbol{\phi}_{i_t,t}$ in Figure 3 can be computed as

$$\boldsymbol{v}_{s-1}^\top (A_t')^{-1} \boldsymbol{\phi}_{i_t,t} = \boldsymbol{y}_s^\top S_s^\top \boldsymbol{\phi}_{i_t,t} - \boldsymbol{y}_s^\top S_s^\top Z_s (I + Z_s^\top Z_s)^{-1} Z_s^\top \boldsymbol{\phi}_{i_t,t} .$$

Calculating the vectors $S_s^\top \boldsymbol{\phi}_{i_t,t}$ and $Z_s^\top \boldsymbol{\phi}_{i_t,t}$ in the above expression takes $\mathcal{O}(s)$ inner products while other $\mathcal{O}(s)$ inner products are required to incrementally compute $S_s^\top Z_s$ from $S_{s-1}^\top Z_{s-1}$. Finally, when calculating the inverse $(I + Z_s^\top Z_s)^{-1}$ we exploit the same updating scheme of Section 4.1,

$$I_s + Z_s^\top Z_s = \left[\begin{array}{cc} I_{s-1} + Z_{s-1}^\top Z_{s-1} & Z_{s-1}^\top \boldsymbol{z}_s \\ \boldsymbol{z}_s^\top Z_{s-1} & 1 + \boldsymbol{z}_s^\top \boldsymbol{z}_s \end{array} \right] ,$$

where $Z_{s-1}^\top \boldsymbol{z}_s$ and $\boldsymbol{z}_s^\top \boldsymbol{z}_s$ require $\mathcal{O}(Ks)$ and $\mathcal{O}(K)$ inner products, respectively. Hence $(I_s + Z_s^\top Z_s)^{-1}$ can be computed from $(I_{s-1} + Z_{s-1}^\top Z_{s-1})^{-1}$ with $\mathcal{O}(Ks)$ extra inner products and $\mathcal{O}(s^2)$ additional scalar multiplications.

7 Spectral co-regularization

An extreme case of multitask learning is the *multiview* setting, where all tasks share the same label. In the multiview protocol, at each time step t the learner receives K instances $\boldsymbol{x}_{1,t}, \ldots, \boldsymbol{x}_{K,t} \in \mathbb{R}^d$, predicts with $\widehat{y}_t \in \{-1, 1\}$, and then receives the correct binary label y_t, which —unlike the general multitask case— is the same for all instances. What distinguishes multiview learning from a standard online binary classification task, defined on instances of the form $\boldsymbol{x}_t = (\boldsymbol{x}_{1,t}, \ldots, \boldsymbol{x}_{K,t})$, is that in multiview one postulates the existence of K vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$ such that *each* $\boldsymbol{u}_i$ is a good linear classifier for the corresponding sequence $(\boldsymbol{x}_{i,1}, y_1), (\boldsymbol{x}_{i,2}, y_2), \ldots$ of examples. In this respect a natural baseline for online multiview learning is the algorithm that chooses a random index $i \in \{1, \ldots, K\}$ and then runs a Perceptron algorithm on the sequence of examples $(\boldsymbol{x}_{i,t}, y_t)$ for $t \geq 1$. Equivalently, we may think of running K Perceptrons in parallel and then average their mistakes (see the remark after Theorem 10).

In this section we design multiview learning algorithms that in certain cases are able to perform significantly better than the above baseline. In order to do so, we arrange the K d-dimensional instances $\boldsymbol{x}_{1,t}, \ldots, \boldsymbol{x}_{K,t}$ in a $d \times K$ *instance matrix* X_t and penalize diversity among the reference vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$ using a matrix norm of the $d \times K$ matrix $U = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K]$.

We focus our attention on matrix norms that are unitarily invariant. Such norms are of the form $\|U\|_f = f(\boldsymbol{\sigma}_U)$, where $\boldsymbol{\sigma}_U = (\sigma_1, \ldots, \sigma_r)$ is the vector of the ordered singular values $\sigma_1 \geq \cdots \geq \sigma_r \geq 0$ of U and $f : \mathbb{R}^r \to \mathbb{R}$, with $r = \min\{K, d\}$, is an absolutely symmetric function —that is, f is invariant under coordinate permutations and sign-changes.

Matrix norms of this form control the distribution of the singular values of U, thus acting as spectral co-regularizers for the reference vectors (see, e.g., [4] for very recent developments on this subject). Known examples are the Schatten p-norms, $\|U\|_{s_p} \stackrel{\text{def}}{=} \|\boldsymbol{\sigma}_U\|_p$. For instance, the Schatten 2-norm is the Frobenius norm. For $p = 1$ the Schatten p-norm becomes the trace norm, a good proxy for the rank of U, since $\|U\|_{s_1} = \|\boldsymbol{\sigma}_U\|_1 \approx \|\boldsymbol{\sigma}_U\|_0 = \text{rank of } U$.

In order to obtain a multiview bound that depends on $\|\boldsymbol{\sigma}_U\|_p$, we extend the dual norm analysis of Section 5 to matrices. We start by defining the matrix version of the quasi-additive algorithm of [14, 20]. We remark that matrix versions of the EG algorithm and the Winnow algorithm (related to specific instances of the quasi-additive algorithm) have been proposed and analyzed in [28, 29, 30]. When dealing with the trace norm regularizer, their algorithms could be specialized to our multiview framework to obtain mistake bounds comparable to ours. See the brief discussion at the end of this section.

The quasi-additive matrix algorithm maintains a $d \times K$ matrix W. Initially, W_0 is the zero matrix. If $s - 1$ mistakes have been made in the first $t - 1$ time steps, then the prediction at time t is $\text{SGN}\big(\langle W_{s-1}, X_t \rangle\big)$, where X_t is the $d \times K$ matrix $[\boldsymbol{x}_{1,t}, \ldots, \boldsymbol{x}_{K,t}]$ in which $\boldsymbol{x}_{i,t}$ is the instance vector associated with the i-th view at time t, and $\langle W_{s-1}, X_t \rangle$ is the standard matrix inner product $\langle W_{s-1}, X_t \rangle = \text{TR}\big(W_{s-1}^\top X_t\big)$.

If a mistake occurs at time t, then W_{s-1} is updated with $W_s = \nabla \frac{1}{2} \|V_s\|_f^2$ where, in turn, the $d \times K$ matrix V_s is updated using a matrix Perceptron rule, $V_s = V_{s-1} + y_t\, X_t$.

A useful property of norms $\|U\|_f = f(\boldsymbol{\sigma}_U)$ is that their duals are easily computed.

Theorem 7 [21, Theorem 2.4] If f is absolutely symmetric and $\|U\|_f = f(\boldsymbol{\sigma}_U)$, then $\|U\|_{f^*} = f^*(\boldsymbol{\sigma}_U)$ where f^* is the convex dual of f.

In the case of Schatten p-norms, we have that the dual of vector norm $\|\cdot\|_p$ is vector norm $\|\cdot\|_q$, where $q = p/(p-1)$

is the dual coefficient of p.

An important feature of the quasi-additive algorithms for vectors is that the mapping $\mu : \boldsymbol{v} \mapsto \mu(\boldsymbol{v}) = \nabla \frac{1}{2} \|\boldsymbol{v}\|^2$ is invertible whenever the vector norm $\|\cdot\|$ satisfies certain regularity properties (see, e.g., [8, page 294]). We call such norms *Legendre*. Hence, we "do not lose information" when the primal weight vector $\boldsymbol{v}$ is mapped to the weight vector $\boldsymbol{w} = \mu(\boldsymbol{v})$ used for prediction. In particular, we always have that $\mu^{-1}(\boldsymbol{v}) = \nabla \frac{1}{2} \|\boldsymbol{v}\|_*^2$, where $\|\cdot\|_*$ is the dual norm of a Legendre norm $\|\cdot\|$ (see, e.g., [8, Lemma 11.5]). This property is preserved when the algorithm is applied to matrices. This is shown by the following result where, without loss of generality, we prove the property for $f(\boldsymbol{\sigma}_U)$ rather than $\frac{1}{2} f(\boldsymbol{\sigma}_U)^2$. (In fact, if f is Legendre, then $\frac{1}{2} f^2$ is also Legendre).

Theorem 8 *Let f be a Legendre function. If $\|U\|_f = f(\boldsymbol{\sigma}_U)$ then $\left(\nabla \|\cdot\|_f\right)^{-1} = \nabla \|\cdot\|_{f^*}$.*

The following result will be useful.

Theorem 9 [21, Theorem 3.1] Let $U \,\mathrm{DIAG}[\boldsymbol{\sigma}_A]\, V^\top$ be an SVD decomposition of a matrix A. If $\|\cdot\|_f$ is a matrix norm such that $\|A\|_f = f(\boldsymbol{\sigma}_A)$ for f Legendre, then $\nabla f(\boldsymbol{\sigma}_A) = \boldsymbol{\sigma}_{\nabla \|A\|_f}$. Moreover, $\nabla \|A\|_f = U \,\mathrm{DIAG}\left[\nabla f(\boldsymbol{\sigma}_A)\right] V^\top$.

Proof of Theorem 8: If $A = U \,\mathrm{DIAG}[\boldsymbol{\sigma}_A]\, V^\top$, then by Theorem 9

$$\nabla \|A\|_f = U \,\mathrm{DIAG}\left[\nabla f(\boldsymbol{\sigma}_A)\right] V^\top = U \,\mathrm{DIAG}\left[\boldsymbol{\sigma}_{\nabla \|A\|_f}\right] V^\top .$$

Therefore, using Theorem 9,

$$\begin{aligned}
\nabla \left\| \left(\nabla \|A\|_f\right) \right\|_{f^*} &= U \,\mathrm{DIAG}\Big[\nabla f^*\big(\nabla f(\boldsymbol{\sigma}_A)\big)\Big] V^\top \\
&= U \,\mathrm{DIAG}[\boldsymbol{\sigma}_A]\, V^\top \quad (f \text{ is Legendre}) \\
&= A
\end{aligned}$$

concluding the proof. ■

We now develop a general analysis of the quasi-additive matrix algorithms, and then specialize it (in Theorem 10 below) to a multiview algorithm operating with a Schatten p-norm regularizer.

We start by adapting the dual norm proof of Section 5 to an arbitrary matrix norm $\|A\|_f = f(\boldsymbol{\sigma}_A)$, where f is Legendre. Let V_m be the primal weight matrix after any number m of mistakes. By Taylor expanding $\frac{1}{2} \|V_s\|_f^2$ around V_{s-1} for each $s = 1, \ldots, m$, and using $y_t \langle W_{s-1}, X_t \rangle \le 0$, we get

$$\frac{1}{2} \|V_m\|_f^2 \le \sum_{s=1}^m D\left(V_s \| V_{s-1}\right)$$

where $D\left(V_s \| V_{s-1}\right)$ is the matrix Bregman divergence $\frac{1}{2}\left(\|V_s\|_f^2 - \|V_{s-1}\|_f^2\right) - y_t \langle W_{s-1}, X_t \rangle$.

Fix any $d \times K$ matrix U. First, we derive a matrix version of the convex inequality for vector norms. We use a classical result by von Neumann (see, e.g., [18, p. 182]) stating that $\langle V, U \rangle \le \boldsymbol{\sigma}_V^\top \boldsymbol{\sigma}_U$ for any two $d \times K$ matrices U and V. We have

$$\begin{aligned}
\|V_m\|_f \|U\|_{f^*} &= f\left(\boldsymbol{\sigma}_{V_m}\right) f^*\left(\boldsymbol{\sigma}_U\right) \quad \text{(by Theorem 7)} \\
&\ge \boldsymbol{\sigma}_{V_m}^\top \boldsymbol{\sigma}_U \quad \text{(by the convex ineq. for norms)} \\
&\ge \langle V_m, U \rangle \quad \text{(by von Neumann's ineq.).}
\end{aligned}$$

In addition, we have $\langle U, V_s \rangle = \langle U, V_{s-1} \rangle + y_t \langle U, X_t \rangle$. Thus we obtain

$$\|V_m\| \ge \frac{\langle U, V_m \rangle}{\|U\|_{f^*}} \ge \frac{Km - \sum_t \ell_t(U)}{\|U\|_{f^*}}$$

where $\ell_t(U) \overset{\text{def}}{=} \sum_{i=1}^K \left[1 - y_t \,\boldsymbol{u}_i^\top \boldsymbol{x}_{i,t}\right]_+ = \sum_{i=1}^K \ell_t(\boldsymbol{u}_i)$. Solving for m gives

$$m \le \frac{1}{K} \sum_{t \in \mathcal{M}} \ell_t(U) + \frac{\|U\|_{f^*}}{K} \sqrt{2 \sum_{s=1}^m D\left(V_s \| V_{s-1}\right)} \, . \quad (18)$$

Equation (18) is our general starting point for analyzing multiview algorithms working under spectral co-regularization. The analysis reduces to bounding from above the second-order term $D\left(\cdot \| \cdot\right)$ of the specific matrix norm $\|\cdot\|_f$ under consideration.

For the rest of this section we focus on the Schatten $2p$-norm $\|V\|_{s_{2p}} = \|\boldsymbol{\sigma}_V\|_{2p}$, where V is a generic $d \times K$ matrix, and p is a positive *integer* (thus $2p$ is an even number ≥ 2). Note that, in general, $\|V\|_{s_{2p}}^2 = \mathrm{TR}\left((V^\top V)^p\right)^{1/p}$.

In order to prove our main result, stated below, we use some facts from differential matrix calculus. A standard reference on this subject is [23], to which the reader is referred.

Theorem 10 *The number of mistakes m made by the $2p$-norm matrix Perceptron, run on any sequence $(X_1, y_1), (X_2, y_2), \ldots$ satisfies, for any $d \times K$ matrix U,*

$$\begin{aligned}
m \le \frac{1}{K} \sum_{t \in \mathcal{M}} \ell_t(U) &+ (2p - 1) \left(\frac{X_{s_{2p}} \|U\|_{s_{2q}}}{K}\right)^2 \\
&+ \frac{X_{s_{2p}} \|U\|_{s_{2q}}}{K} \sqrt{\frac{2p - 1}{K} \sum_{t \in \mathcal{M}} \ell_t(U)}
\end{aligned}$$

where $X_{s_{2p}} = \max_{t \in \mathcal{M}} \|X_t\|_{s_{2p}}$, $\|U\|_{s_{2q}}$ is the Schatten $2q$-norm of U, with $2q = \frac{2p}{2p-1}$, and $\mathcal{M}$ is the set of mistaken trial indices.

Remark Similarly to the vector case, when the parameter p is chosen to be logarithmic in $r = \min\{d, K\}$, the p-norm matrix Perceptron penalizes diversity using the trace norm of U. If the vectors $\boldsymbol{u}_i$ span a subspace of size $\ll K$, and instances tend to have K nonzero singular values of roughly the same magnitude, then $\|U\|_{s_{2q}} \approx \|U\|_{s_2}$ while $X_{s_{2p}}^2 \approx X_{s_\infty}^2 \approx X_{s_2}^2 / K$. Hence this choice of p leads (at least in the linearly separable case) to a factor K improvement over the bound achieved by the matrix algorithm based on the Frobenius norm ($p = 1$ in Theorem 10), which amounts to running K independent Perceptrons in parallel and then average their mistakes.

The following trace inequality is our main technical lemma.

Lemma 11 *Let A,B be positive semidefinite matrices, of size $d \times d$ and $K \times K$ respectively, with the same nonzero eigenvalues. Let X be an arbitrary real matrix of size $d \times K$. Then, for any pair on nonnegative exponents $l, g \ge 0$, we have $\mathrm{TR}(X^\top A^l X B^g) \le \left(\mathrm{TR}(X^\top X)^p\right)^{1/p} \left(\mathrm{TR}\; A^{(l+g)q}\right)^{1/q}$ where $1/p + 1/q = 1$, $p \ge 1$.*

Proof of Lemma 11: We first consider the case $l \le g$. By the Cauchy-Schwartz and Holder's inequalities applied to traces [23, Ch.11] we have

$$\begin{aligned}\mathrm{TR}(X^\top A^l X B^g) &= \mathrm{TR}\big(B^{(g-l)/2} X^\top A^l X B^{(g+l)/2}\big) \qquad (19)\\ &\le \mathrm{TR}\big(X^\top A^{2l} X B^{g-l}\big)^{1/2} \mathrm{TR}\big(X^\top X B^{g+l}\big)^{1/2}\\ &\le \mathrm{TR}\big(X^\top A^{2l} X B^{g-l}\big)^{1/2} T_p\big(X^\top X\big)^{1/2} T_q\big(B^{g+l}\big)^{1/2}\end{aligned}$$

where we used the shorthand $T_r(Z) = (\mathrm{TR} Z^r)^{1/r}$. In the case when $l > g$ we can simply swap the matrices $X^\top A^l$ and XB^g and reduce to the previous case.

We now recursively apply the above argument to the left-hand side of (19). Recalling that $T_q(A) = T_q(B)$ and $T_p(X^\top X) = T_p(XX^\top)$, after n steps we obtain

$$\begin{aligned}\mathrm{TR}\big(X^\top A^l X B^g\big) &\le \big(\mathrm{TR}(X^\top A^{l'} X B^{g'})\big)^{1/2^n} \times\\ &\times T_p\big(X^\top X\big)^{\sum_{i=1}^n (1/2)^i} T_q\big(B^{g+l}\big)^{\sum_{i=1}^n (1/2)^i}\end{aligned}$$

for some pair of exponents $l', g' \ge 0$ such that $l' + g' = l + g$. Since for any such pair l', g', we have $\mathrm{TR}(X^\top A^{l'} X A^{g'}) < \infty$, we can take the limit as $n \to \infty$. Recalling that $\sum_{i=1}^\infty (1/2)^i = 1$ completes the proof. ■

Proof of Theorem 10: We set for brevity $G : \mathbb{R}^{d\times K} \to R$,

$$G(V) = \frac{1}{2}\mathrm{TR}\big((V^\top V)^p\big)^{1/p} = \frac{1}{2}\,\|V\|_{s_{2p}}^2 .$$

Thus in our case

$$D\left(V_s \| V_{s-1}\right) = G(V_{s-1} + y_t X_t) - G(V_{s-1}) - y_t\,\langle W_{s-1}, X_t\rangle.$$

Since $G(V)$ is twice[4] continuously differentiable, by the mean-value theorem we can write

$$D\left(V_s \| V_{s-1}\right) = \frac{1}{2}\mathrm{VEC}(X_t)^\top H_G(\xi)\mathrm{VEC}(X_t), \qquad (20)$$

where $\mathrm{VEC}(X)$ is the standard columnwise vectorization of a matrix X, H_G denotes the Hessian matrix of (matrix) function G and ξ is some matrix on the line connecting V_{s-1} to V_s. Using the rules of matrix differentiation, the gradient ∇G of G is $\nabla G(V) = c(V)\mathrm{VEC}(D)^\top$ where we set for brevity $D = VB^{p-1}$, $c(V) = \mathrm{TR}(B^p)^{1/p-1}$, with $B = V^\top V$. Taking the second derivative $H_G = \nabla\nabla G$ gives $H_G(V) = \mathrm{VEC}\,(D)\,\nabla(c(V)) + c(V)\,\nabla\,(D)$. Now, recalling the definition of $c(V)$, it is not hard to show that $\mathrm{VEC}\,(D)\,\nabla(c(V))$ is the $Kd \times Kd$ matrix

$$2(1-p)\mathrm{TR}(B^p)^{1/p-2}\,\mathrm{VEC}(D)\,\mathrm{VEC}(D)^\top.$$

Since $p \ge 1$ this matrix is negative semidefinite, and we can disregard it when bounding from the above the quadratic form (20). Thus we continue by considering only the second term $c(V)\,\nabla\,(D)$ of the Hessian matrix. We have

$$\nabla\,(D) = \left(B^{p-1} \otimes I_d\right) + (I_k \otimes V)\nabla\left(B^{p-1}\right),$$

[4]In fact G is C^∞ everywhere but (possibly) in zero, since $\mathrm{TR}\Big((V^\top V)^p\Big)$ is just a polynomial function of the entries of V. Moreover $\mathrm{TR}\Big((V^\top V)^p\Big) = 0$ if and only if V is the zero matrix.

where

$$\nabla(B^{p-1}) = \left(\sum_{\ell=0}^{p-2} B^\ell \otimes B^{p-2-\ell}\right)\left(I_{K^2} + T_K\right)\left(I_k \otimes V^\top\right),$$

and T_K is the $K^2 \times K^2$ commutation matrix such that $T_K \mathrm{VEC}(M) = \mathrm{VEC}(M^\top)$ for any $K \times K$ matrix M. Putting together

$$\begin{aligned}(20) \quad \le \quad & \frac{c(V)}{2}\,\mathrm{VEC}(X_t)^\top (B^{p-1} \otimes I_d)\,\mathrm{VEC}(X_t)\\ & + \frac{c(V)}{2}\mathrm{VEC}(X_t)^\top (I_k \otimes V)\Sigma \times\\ & \times (I_{K^2} + T_K)\left(I_k \otimes V^\top\right)\mathrm{VEC}(X_t)\,, \quad (21)\end{aligned}$$

where we used the shorthand $\Sigma = \sum_{\ell=0}^{p-2} B^\ell \otimes B^{p-2-\ell}$. We now bound the two terms in the right-hand side of (21). By well-known relationships between Kronecker products and the VEC operator (see [23, Ch. 3]) we can write

$$\begin{aligned}&\frac{c(V)}{2}\,\mathrm{VEC}(X_t)^\top (B^{p-1} \otimes I_d)\,\mathrm{VEC}(X_t)\\ &\qquad = \frac{c(V)}{2}\,\mathrm{TR}(X_t^\top X_t B^{p-1}) \le \frac{1}{2}\,\left(\mathrm{TR}(X_t^\top X_t)^p\right)^{1/p}\,,\end{aligned}$$

independent of V. The majorization follows from Holder's inequality applied to the positive semidefinite matrices $X_t^\top X_t$ and B^{p-1}. Moreover, it is easy to see that the symmetric matrices Σ and T_K commute, thereby sharing the same eigenspace. Hence, $\Sigma\,(I_{K^2} + T_K) \preccurlyeq 2\Sigma$, and we can bound from above the second term in (21) by

$$c(V)\mathrm{VEC}(X_t)^\top \sum_{\ell=0}^{p-2} B^\ell \otimes A^{p-1-\ell}\mathrm{VEC}(X_t)\,,$$

where we set $A = VV^\top$. Again, [23, Ch. 3] allows us to rewrite this quadratic form as the sum of traces

$$c(V)\sum_{\ell=0}^{p-2} \mathrm{TR}(X_t^\top A^{p-1-\ell} X_t B^\ell)\,.$$

Since A and B have the same nonzero eigenvalues, we can apply Lemma 11 to each term and put together as in (21). After simplifying we get

$$(20) \le \frac{1}{2}\,(2p-1)\left(\mathrm{TR}(X_t^\top X_t)^p\right)^{1/p} = \frac{1}{2}\,(2p-1)||X_t||_{s_{2p}}^2.$$

The desired bound is then obtained by plugging back into (18), solving the resulting inequality for m, and overapproximating. ■

The result of Theorem 10 is similar to those obtained in [28, 29, 30]. However, unlike these previous results, our matrix algorithm has no learning rate to tune (a property inherited from the vector p-norm Perceptron of [13]) and works for arbitrary nonsquare matrices U. We also observe that the prediction $\widehat{y}_t = \mathrm{SGN}\left(\mathrm{TR}\big(W_{s-1}^\top X_t\big)\right)$ of the p-norm matrix Perceptron reduces to computing the sign of $\mathrm{TR}\big((V_{s-1}^\top V_{s-1})^{p-1} V_{s-1}^\top X_t\big)$ (recall the expression for ∇G calculated in the proof of Theorem 10). Since matrix V_s is updated additively, it is clear that both $V_{s-1}^\top V_{s-1}$ and $V_{s-1}^\top X_t$ do depend on instance vectors $\boldsymbol{x}_{i,t}$ only through inner products. This allows us to turn our p-norm matrix Perceptron into a kernel-based algorithm, and repeat the analysis given here using a standard RKHS formalism.

8 Conclusions and ongoing research

In this work we have studied the problem of learning multiple tasks online using various approaches to formalize the notion of task relatedness.

Our results can be extended in different directions. First, in Sections 3.2 and 6 it might be interesting to devise methods for dynamically adapting the b parameter as new data are revealed. Second, the mistakes of the second-order algorithm MMPERC have been bounded using a first-order analysis. A more refined analysis should reveal in the bound an explicit dependence on the spectral properties of the data. It is also worth noting that the significance of the mistake bound obtained for MMPERC relies on the fact that the algorithm assumes the tasks to be different, although somewhat related. In the case when the K observed instances share the same label at each time step (like in multiview learning), we could not devise an algorithm with a significant advantage over the following trivial baseline: run K Perceptrons in parallel and use the sum of margins to predict. Third, it would be interesting to study the problem of learning multiple tasks when K predictions have to be output in each step. In this case the main difficulty appears to be the control of the interaction among instances at each time step. Fourth, it would be also interesting to prove *lower* bounds on the number of mistakes, as a function of task relatedness. Finally, since multitask learning problems arise naturally in a variety of settings, spanning from biology to news processing, we plan to complement the theoretical analysis presented in this paper with experimental results, so as to evaluate the empirical performance of our algorithms in real-case scenarios.

Acknowledgments. Thanks to Sham Kakade, Massi Pontil, and Francis Bach for useful discussions. We also thank the COLT 2008 reviewers for their comments. This work was supported in part by the PASCAL2 Network of Excellence under EC grant no. 216886. This publication only reflects the authors' views.

References

[1] J. ABERNETHY, P.L. BARTLETT & A. RAKHLIN, Multitask learning with expert advice, Proc. 20th COLT, pp. 484–498, Springer, 2007.

[2] R.K. ANDO & T. ZHANG, A framework for learning predictive structures from multiple tasks and unlabeled data, *JMLR*, 6, pp. 1817–1853, MIT Press, 2005.

[3] A. ARGYRIOU, T. EVGENIOU & M. PONTIL Multi-Task feature learning, NIPS 19, pp. 41–48, MIT Press, 2007.

[4] A. ARGYRIOU, C.A. MICCHELLI, M. PONTIL & Y. YING, A spectral regularization framework for multi-task structure learning, NIPS 20, MIT Press, 2008.

[5] K. AZOURY AND M. WARMUTH, Relative loss bounds for on-line density estimation with the exponential family of distributions, *Machine Learning*, 43, pp. 211–246, 2001.

[6] U. BREFELD, T. GAERTNER, T. SCHEFFER, & S. WROBEL, Efficient co-regularised least squares regression, Proc. 23rd ICML, 2006.

[7] N. CESA-BIANCHI, A. CONCONI & C. GENTILE, A second-order Perceptron algorithm, *SIAM Journal on Computing*, 34/3, pp. 640–668, 2005.

[8] N. CESA-BIANCHI & G. LUGOSI, *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[9] O. DEKEL, P.M. LONG & Y. SINGER, Online learning of multiple tasks with a shared loss, *JMLR*, 8, pp. 2233–2264, 2007.

[10] T. EVGENIOU, M. PONTIL & T. POGGIO, Regularization networks and Support Vector Machines, *Advances in Computational Mathematics*, 13/1, pp. 1–50, Springer, 2000.

[11] T. EVGENIOU, C. MICCHELLI & M. PONTIL, Learning Multiple tasks with kernel methods, *JMLR*, 6, pp. 615–637, MIT Press, 2005.

[12] Y. FREUND & R.E. SCHAPIRE, Large margin classification using the Perceptron algorithm. *Machine Learning*, 37:3, pp. 277–296, 1999.

[13] C. GENTILE, The robustness of the p-norm algorithms, *Machine Learning*, 53, pp. 265–299, 2003.

[14] A. GROVE, N. LITTLESTONE & D. SCHUURMANS, General convergence results for linear discriminant updates, *Machine Learning*, 43, pp. 173–210, 2001.

[15] M. HERBSTER, M. PONTIL & L. WAINER, Online learning over graphs, Proc. 22nd ICML, pp. 305–312, ACM Press, 2005.

[16] L. HOGBEN, *Handbook of Linear Algebra*, Discrete Mathematics and Its Applications, 39, CRC Press, 2006.

[17] R.A. HORN & C.R. JOHNSON, *Matrix Analysis*. Cambridge University Press, 1985.

[18] R.A. HORN & C.R. JOHNSON, *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[19] A. JAGOTA & M.K. WARMUTH, Continuous and discrete time nonlinear gradient descent: relative loss bounds and convergence, Electr. Proc. 5th International Symposium on Artificial Intelligence and Mathematics, 1998. Electronic, http://rutcor.rutgers.edu/∼amai.

[20] J. KIVINEN & M. WARMUTH, Relative loss bounds for multidimensional regression problems, *Machine Learning*, 45, pp. 301–329, 2001.

[21] A.S. LEWIS, The convex analysis of unitarily invariant matrix functions, *Journal of Convex Analysis*, 2, pp. 173–183, 1995.

[22] N. LITTLESTONE, *Mistake bounds and logarithmic linear-threshold learning algorithms*. Ph.D. Thesis, University of California at Santa Cruz, 1989.

[23] J.R. MAGNUS, & H, NEUDECKER, *Matrix Differential Calculus with Applications in Statistics and Econometrics, revised edition*. John Wiley, 1999.

[24] A. MAURER, Bounds for linear multi-task learning, *JMLR*, 7, pp. 117–139, MIT Press, 2006.

[25] R.T. ROCKAFELLAR, *Convex Analysis*. Princeton University Press, 1970.

[26] D. ROSENBERG & P. BARTLETT, Rademacher complexity of co-regularized kernel classes, Proc. Artificial Intelligence and Statistics, 2007.

[27] V. SINDHWANI, P. NIYOGI & M. BELKIN, A co-regularized approach to semi-supervised learning. Proc. ICML Workshop on Learning with Multiple Views, 2005.

[28] K. TSUDA, G. RAETSCH & M.K. WARMUTH, Matrix exponentiated gradient updates for on-line learning and Bregman projection, *JMLR*, 6, pp. 995–1018, 2005.

[29] M.K. WARMUTH & D. KUZMIN, Online variance minimization, Proc. 19th COLT, Springer, 2006.

[30] M.K. WARMUTH, Winnowing subspaces. Proc. 24th ICML, pp. 999–1006, ACM Press, 2007.