Journal of Machine Learning Research 11 (2010) 2901-2934 Submitted 10/09; Revised 9/10; Published 10/10

Linear Algorithms for Online Multitask Classification

Giovanni Cavallanti CAVALLANTI@DSI.UNIMI.IT
Nicolò Cesa-Bianchi CESA-BIANCHI@DSI.UNIMI.IT
DSI, Università degli Studi di Milano
via Comelico, 39
20135 Milano, Italy

Claudio Gentile CLAUDIO.GENTILE@UNINSUBRIA.IT
DICOM, Università dell'Insubria
via Mazzini, 5
21100 Varese, Italy

Editor: Manfred Warmuth

Abstract

We introduce new Perceptron-based algorithms for the online multitask binary classification problem. Under suitable regularity conditions, our algorithms are shown to improve on their baselines by a factor proportional to the number of tasks. We achieve these improvements using various types of regularization that bias our algorithms towards specific notions of task relatedness. More specifically, similarity among tasks is either measured in terms of the geometric closeness of the task reference vectors or as a function of the dimension of their spanned subspace. In addition to adapting to the online setting a mix of known techniques, such as the multitask kernels of Evgeniou *et al.*, our analysis also introduces a matrix-based multitask extension of the p-norm Perceptron, which is used to implement spectral co-regularization. Experiments on real-world data sets complement and support our theoretical findings.

Keywords: mistake bounds, perceptron algorithm, multitask learning, spectral regularization

1. Introduction

In this work we study online supervised learning algorithms that process multiple data streams at the same time. More specifically, we consider the *multitask* classification learning problem where observed data describe different learning tasks.

Incremental multitask learning systems, which simultaneously process data from multiple streams, are widespread. For instance, in financial applications a trading platform chooses investments and allocates assets using information coming from multiple market newsfeeds. When the learning tasks are unrelated, running different instances of the same underlying algorithm, one for each task, is a sound and reasonable policy. However, in many circumstances data sources share similar traits and are therefore related in some way. Unsurprisingly, this latter situation is quite common in real-world applications. In these cases the learning algorithm should be able to capitalize on data relatedness.

In multitask classification an online linear classifier (such as the Perceptron algorithm) learns from examples associated with $K > 1$ different binary classification tasks. Our goal is to design online interacting algorithms that perform better than independent learners whenever the tasks are related. We formalize task relatedness in different ways, and derive precise formalizations of the

©2010 Giovanni Cavallanti, Nicolò Cesa-Bianchi and Claudio Gentile.

advantage resulting from such interaction. Our investigation considers two variants of the online multitask protocol: (1) at each time step the learner acts on a single adversarially chosen task; (2) all tasks are simultaneously processed at each time step. Each setup allows for different approaches to the multitask problem and caters for different real-world scenarios. For instance, one of the advantages of the first approach is that, in most cases, the cost of running multitask algorithms has a mild dependence on the number K of tasks. The multitask classifiers we study here manage to improve, under certain assumptions, the cumulative regret achieved by a natural baseline algorithm through the information acquired and shared across different tasks.

Our analysis builds on ideas that have been developed in the context of statistical learning where the starting point is a regularized empirical loss functional or Tikhonov functional. In that framework the objective includes a co-regularization term in the form of a squared norm in some Hilbert space of functions that favors those solutions (i.e., predictive functions for the K tasks) that lie "close" to each other. In this respect, we study two main strategies. The first approach followed here is to learn K linear functions parameterized by $u^\top = (u_1^\top, \dots, u_K^\top) \in \mathbb{R}^{Kd}$ through the minimization of an objective functional involving the sum of a loss term plus the regularization term $u^\top A u$, where A is a positive definite matrix enforcing certain relations among tasks. Following Evgeniou et al. (2005), the K different learning problems are reduced to a single problem by choosing a suitable embedding of the input instances into a common Reproducing Kernel Hilbert Space (RKHS). This reduction allows us to solve a multitask learning problem by running any kernel-based single-task learning algorithm with a "multitask kernel" that accounts for the co-regularization term in the corresponding objective functional. We build on this reduction to analyze the performance of the Perceptron algorithm and some of its variants when run with a multitask kernel.

As described above, we also consider a different learning setup that prescribes the whole set of K learning tasks to be worked on at the same time. Once again we adopt a regularization approach, this time by adding a bias towards those solutions that lie on the same low dimensional subspace. To devise an algorithm for this model, we leverage on the well-established theory of potential-based online learners. We first define a natural extension of the p-norm Perceptron algorithm to a certain class of matrix norms, and then provide a mistake bound analysis for the multitask learning problem depending on spectral relations among different tasks. Our analysis shows a factor K improvement over the algorithm that runs K independent Perceptrons and predicts using their combined margin (see Section 1.1). The above is possible as long as the the example vectors observed at each time step are unrelated, while the sequences of multitask data are well predicted by a set of highly related linear classifiers.

1.1 Main Contributions

The contribution of this paper to the current literature is twofold. First, we provide theoretical guarantees in the form of mistake bounds for various algorithms operating within the online multitask protocol. Second, we present various experiments showing that these algorithms perform well on real problems.

Our theoretical results span across the two previously mentioned settings. In the adversarially chosen task setting, we extend the ideas introduced by Evgeniou et al. (2005) to the online learning setup, and present upper bounds which depend on task relatedness. On one hand, we show that whenever the reference vectors associated with different tasks are related, we achieve an improvement of a factor up to K over the baseline approach where K online classifiers are run in parallel and tasks are processed in isolation. On the other hand, when tasks are unrelated our bounds become not much worse than the one achieved by separately running K classifiers. In this context, the notion of relatedness among tasks follows from the specific choice of a matrix parameter that essentially defines the so-called multitask kernel. We also provide some new insight on the role played by this kernel when used as a plug-in black-box by the Perceptron or other Perceptron-like algorithms.

In the simultaneous task setting, we introduce and analyze a matrix-based multitask extension of the p-norm Perceptron algorithm (Grove et al., 2001; Gentile, 2003) which allows us to obtain a factor K improvement in a different learning setting, where the baseline, which is still the algorithm that maintains K independent classifiers, is supposed to output K predictions per trial.

On the experimental side, we give evidence that our multitask algorithms provide a significant advantage over common baselines. In particular, we show that on a text categorization problem, where each task requires detecting the topic of a newsitem, a large multitask performance increase is attainable whenever the target topics are related. Additional experiments on a spam data set confirm the potential advantage of the p-norm Perceptron algorithm in a real-world setting.

This work is organized as follows. In Section 2 we introduce notation and formally define the *adversarially chosen task* protocol. The multitask Perceptron algorithm is presented in Section 3 where we also discuss the role of the multitask feature map and show (Section 4) that it can be used to turn online classifiers into multitask classifiers. We detail the matrix-based approach to the simultaneous multitask learning framework in Section 5. Section 6 is devoted to the theoretical analysis of a general potential-based algorithm for this setup. We conclude the paper with a number of experiments establishing the empirical effectiveness of our algorithms (Section 7).

1.2 Related Work

The problem of learning from multiple tasks has been the subject of a number of recently published papers. In Evgeniou et al. (2005) a batch multitask learning problem is defined as a regularized optimization problem and the notion of multitask kernel is introduced. More specifically, they consider a regularized functional that encodes multitask relations over tasks, thus biasing the solution of the problem towards functions that lie close to each other. Argyriou et al. (2007, 2008) build on this formalization to simultaneously learn a multitask classifier and the underlying spectral dependencies among tasks. A similar model but under cluster-based assumptions is investigated in Jacob et al. (2009). A different approach is discussed in Ando and Zhang (2005) where a structural risk minimization method is presented and multitask relations are established by enforcing predictive functions for the different tasks to belong to the same hypothesis set. Complexity results for multitask learning under statistical assumptions are also given in Maurer (2006).

In the context of online learning, multitask problems have been studied in Abernethy et al. (2007) within the learning with expert advice model. In this model the forecaster has access to a fixed set of experts and is expected to make predictions for K different tasks. Regret bounds are given under the assumption that the set of best experts for the K tasks is small, as a way to formalize task similarity. Whereas these studies consider a multitask protocol in which a single task is acted upon at each time step (what we call in this paper the adversarially chosen task protocol), the work of Lugosi et al. (2009) investigates the problem where an action for each task must be chosen at every step. The relatedness among tasks is captured by imposing restrictions on the joint action chosen at each step.

Online linear multitask algorithms for the simultaneous task setting have been studied in Dekel et al. (2007), where the separate learning tasks are collectively dealt with through a common multitask loss. Their approach, however, is fundamentally different from the one considered here. In fact, using a common loss function has more the effect of prioritizing certain tasks over the others, whereas our regularized approach hopes to benefit from the information provided by each task to speed up the learning process for the other ones. Nonetheless, it is not difficult to extend our analysis to consider a more sophisticated notion of multitask loss (see Remark 12 in Section 6.2), thus effectively obtaining a shared loss regularized multitask algorithm.

Online matrix approaches to the multitask and the related multiview learning problems were considered in various works. Matrix versions of the EG algorithm and the Winnow algorithm (related to specific instances of the quasi-additive algorithms) have been proposed and analyzed in Tsuda et al. (2005), Warmuth (2007), and Warmuth and Kuzmin (2006). When dealing with the trace norm regularizer, their algorithms could be generalized to our simultaneous multitask framework to obtain mistake bounds comparable to ours. However, unlike those papers, we do not have learning rate tuning issues and, in addition, we directly handle general nonsquare task matrices.

Finally, Agarwal et al. (2008) consider multitask problems in the restricted expert setting, where task relatedness is enforced by a group norm regularization. Their results are essentially incomparable to ours.

2. The Adversarially Chosen Task Protocol: Preliminaries

The adversarially chosen task protocol works as follows. Let K be the number of binary classification tasks indexed by $1,\dots,K$. Learning takes place in a sequential fashion: At each time step $t=1,2,\dots$ the learner receives a task index $i_t \in \{1,\dots,K\}$ and observes an instance vector $x_t \in \mathbb{R}^d$ which plays the role of side information for the task index i_t. Based on the pair $\big(x_t, i_t\big)$ it outputs a binary prediction $\widehat{y}_t \in \{-1,1\}$ and then receives the correct label $y_t \in \{-1,1\}$ for task index i_t. So, within this scheme, the learner works at each step on a single chosen task among the K tasks and operates under the assumption that instances from different tasks are vectors of the same dimension. No assumptions are made on the mechanism generating the sequences $\big(x_1,y_1\big),\big(x_2,y_2\big),\dots$ of task examples. Moreover, similarly to Abernethy et al. (2007), the sequence of task indices $i_1,i_2,\dots$ is also generated in an adversarial manner. To simplify notation we introduce a "compound" description for the pair $\big(x_t,i_t\big)$ and denote by $\phi_t \in \mathbb{R}^{dK}$ the vector

$$\phi_t^\top \stackrel{\text{def}}{=} \Big(\ \underbrace{0,\dots,0}_{(i_t-1)d \text{ times}}\ \ x_t^\top\ \ \underbrace{0,\dots,0}_{(K-i_t)d \text{ times}}\ \Big)\,. \tag{1}$$

Within this protocol (studied in Sections 3 and 4) we use ϕ_t or $\big(x_t,i_t\big)$ interchangeably when referring to a multitask instance. In the following we assume instance vectors are of (Euclidean) unit norm, that is, $\|x_t\|=1$, so that $\|\phi_t\|=1$.

We measure the learner's performance with respect to that of a (compound) reference predictor that is allowed to use a different linear classifier, chosen in hindsight, for each one of the K tasks. To remain consistent with the notation used for multitask instances, we introduce the "compound" reference task vector $u^\top = \big(u_1^\top,\dots,u_K^\top\big)$ and define the hinge loss for the compound vector u as

$$\ell_t(u) \stackrel{\text{def}}{=} \max\big\{0, 1-y_t\, u^\top \phi_t\big\} = \max\big\{0, 1-y_t\, u_{i_t}^\top x_t\big\}.$$

It is understood that the compound vectors are of dimension Kd. Our goal is then to compare the learner's mistakes count to the cumulative hinge loss

$$\sum_t \ell_t(u) \tag{2}$$

suffered by the compound reference task vector u. This of course amounts to summing over time steps t the losses incurred by reference task vectors u_{i_t} with respect to instance vectors x_{i_t}.

In this respect, we aim at designing algorithms that make fewer mistakes than K independent learners when the tasks are related, and do not perform much worse than those when the tasks are completely unrelated. For instance, if we use Euclidean distance to measure task relatedness, we say that the K tasks are related if there exist reference task vectors $u_1, \dots, u_K \in \mathbb{R}^d$ having small pairwise distances $\|u_i - u_j\|$, and achieving a small cumulative hinge loss in the sense of (2). More general notions of relatedness are investigated in the next sections.

Finally, we find it convenient at this point to introduce some matrix notation. We use I_d to refer to the $d \times d$ identity matrix but drop the subscript whenever it is clear from context. Given a matrix $M \in \mathbb{R}^{m \times n}$ we denote by $M_{i,j}$ the entry that lies in the i-th row, j-th column. Moreover, given two matrices $M \in \mathbb{R}^{m \times n}$ and $N \in \mathbb{R}^{m \times r}$ we denote by $[M, N]$ the $m \times (n+r)$ matrix obtained by the horizontal concatenation of M and N. The Kronecker or direct product between two matrices $M \in \mathbb{R}^{m \times n}$ and $N \in \mathbb{R}^{q \times r}$ is the block matrix $M \otimes N$ of dimension $mq \times nr$ whose block on row i and column j is the $q \times r$ matrix $M_{i,j}N$.

3. The Multitask Perceptron Algorithm

We first introduce a simple multitask version of the Perceptron algorithm for the protocol described in the previous section. 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 $w_{i,t}$ be the weight vector associated with task i at time t. If we are forced (by the adversary) to predict on task i_t, and our prediction happens to be wrong, we update $w_{i_t,t-1}$ through the standard additive rule $w_{i_t,t} = w_{i_t,t-1} + \eta y_t x_t$ (where $\eta > 0$ is a constant learning rate) but, at the same time, we perform a "half-update" on the remaining $K-1$ Perceptrons, that is, we set $w_{j,t} = w_{j,t-1} + \frac{\eta}{2} y_t x_t$ 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 $w_{j,t}$, $j = 1, \dots, K$, 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 $w_{j,t}$ through learning rates defined by a $K \times K$ *interaction matrix* A. It is A that encodes our beliefs about the learning tasks: different choices of the interaction matrix result in different geometrical assumptions on the tasks.

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 1. The weights of the K Perceptrons are maintained in a compound vector $w_s^\top = \big(w_{1,s}^\top, \dots, w_{K,s}^\top\big)$, with $w_{j,s} \in \mathbb{R}^d$ for all j. The algorithm predicts y_t through the sign $\widehat{y}_t$ of the i_t-th Perceptron's margin $w_{s-1}^\top \phi_t = w_{i_t,s-1}^\top x_t$. Then, if the prediction and the true label disagree, the compound vector update rule is $w_s = w_{s-1} + \big(A \otimes I_d\big)^{-1} \phi_t$. Since $\big(A \otimes I_d\big)^{-1} = A^{-1} \otimes I_d$, the above update is equivalent to the K task updates

$$w_{j,s} = w_{j,s-1} + y_t A^{-1}_{j,i_t} x_t \qquad j = 1, \dots, K\,.$$

```
Parameters: Positive definite $K \times K$ interaction matrix $A$.
Initialization: $w_0 = 0 \in \mathbb{R}^{Kd}$, $s = 1$.
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 $x_t \in \mathbb{R}^d \, : \, ||x_t|| = 1$;
    2. Build the associated multitask instance $\phi_t \in \mathbb{R}^{Kd}$;
    3. Predict label $y_t \in \{-1, +1\}$ with $\widehat{y}_t = \mathrm{SGN}\big(w_{s-1}^\top \phi_t\big)$;
    4. Get label $y_t \in \{-1, +1\}$;
    5. If $\widehat{y}_t \neq y_t$ then update:
           $w_s = w_{s-1} + y_t \big(A \otimes I_d\big)^{-1} \phi_t, \quad s \leftarrow s + 1 .$
```

Figure 1: The multitask Perceptron algorithm.

The algorithm is mistake-driven, hence w_{s-1} is updated (and s is increased) only when $\widehat{y}_t \neq y_t$. In the following we use $A_\otimes$ as a shorthand for $A \otimes I_d$.

We now show that the algorithm in Figure 1 has the potential to make fewer mistakes than K independent learners when the tasks are related, and does not perform much worse than that when the tasks are completely unrelated. The bound dependence on the task relatedness is encoded as a quadratic form involving the compound reference task vector u and the interaction matrix A.

We specify the online multitask problem by the sequence $(\phi_1, y_1), (\phi_2, y_2), \ldots \in \mathbb{R}^{dK} \times \{-1, 1\}$ of multitask examples.

Theorem 1 *The number of mistakes m made by the multitask Perceptron algorithm in Figure 1, run with an interaction matrix A on any finite multitask sequence of examples $(\phi_1, y_1), (\phi_2, y_2), \ldots \in \mathbb{R}^{Kd} \times \{-1, 1\}$ satisfies, for all $u \in \mathbb{R}^{Kd}$,*

$$m \le \sum_{t \in \mathcal{M}} \ell_t(u) + \max_{i=1,\ldots,K} (A^{-1})_{i,i} \big(u^\top A_\otimes u\big) + \sqrt{\max_{i=1,\ldots,K} (A^{-1})_{i,i} \big(u^\top A_\otimes u\big) \sum_{t \in \mathcal{M}} \ell_t(u)}$$

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

Theorem 1 is readily proven by using the fact that the multitask Perceptron is a specific instance of the kernel Perceptron algorithm, for example, Freund and Schapire (1999), using the so-called linear *multitask* kernel introduced in Evgeniou et al. (2005) (see also Herbster et al., 2005). This kernel is defined as follows: for any positive definite $K \times K$ interaction matrix A introduce the Kd-dimensional RKHS $\mathcal{H} = \mathbb{R}^{Kd}$ with the inner product $\langle u, v \rangle_{\mathcal{H}} = u^\top A_\otimes v$. Then define the kernel feature map $\psi : \mathbb{R}^d \times \{1, \ldots, K\} \to \mathcal{H}$ such that $\psi(x_t, i_t) = A_\otimes^{-1} \phi_t$. The kernel used by the multitask Perceptron is thus defined by

$$\mathcal{K}\big((x_s, i_s), (x_t, i_t)\big) = \big\langle \psi(x_s, i_s), \psi(x_t, i_t) \big\rangle_{\mathcal{H}} = \phi_s^\top A_\otimes^{-1} \phi_t \ . \tag{3}$$

Remark 2 *Although the multitask kernel is appealing because it makes the definition of the multitask Perceptron simple and intuitive, one easily sees that the RKHS formalism is not necessary here since the kernel is actually linear. In fact, by re-defining the feature mapping as $\psi : \mathbb{R}^d \times \{1,\dots,K\} \to \mathbb{R}^{Kd}$, where $\mathbb{R}^{Kd}$ is now endowed with the usual Euclidean product, and by letting $\psi(x_t, i_t) = A_\otimes^{-1/2}\phi_t$, one gets an equivalent formulation of the multitask Perceptron based on $A_\otimes^{-1/2}$ rather than $A_\otimes^{-1}$. In the rest of the paper we occasionally adopt this alternative linear kernel formulation, in particular whenever it makes the definition of the algorithm and its analysis simpler.*

Proof [Theorem 1] We use the following version of the kernel Perceptron bound (see, e.g., Cesa-Bianchi et al., 2005),

$$m \le \sum_t \ell_t(h) + \|h\|_{\mathcal{H}}^2 \left(\max_t \|\psi(x_t, i_t)\|_{\mathcal{H}}^2\right) + \|h\|_{\mathcal{H}} \sqrt{\left(\max_t \|\psi(x_t, i_t)\|_{\mathcal{H}}^2\right) \sum_t \ell_t(h)}$$

where h is any function in the RKHS $\mathcal{H}$ induced by the kernel. The proof is readily concluded by observing that, for the kernel (3) we have

$$\|u\|_{\mathcal{H}}^2 = u^\top A_\otimes u \qquad \text{and} \qquad \|\psi(x_t, i_t)\|_{\mathcal{H}}^2 = \phi_t^\top A_\otimes^{-1} \phi_t = (A^{-1})_{i_t, i_t}$$

since ϕ_t singles out the i_t's block of matrix $A_\otimes^{-1}$. ■

In the next three subsections we investigate the role of the quadratic form $u^\top A_\otimes u$ and specialize Theorem 1 to different interaction matrices.

3.1 Pairwise Distance Interaction Matrix

The first choice of A we consider is the following simple update step (corresponding to the multitask Perceptron example we made at the beginning of this section).

$$w_{j,s} = w_{j,s-1} + \begin{cases} \frac{2}{K+1} y_t x_t & \text{if } j = i_t, \\ \frac{1}{K+1} y_t x_t & \text{otherwise.} \end{cases}$$

As it can be easily verified, this choice is given by

$$A = \begin{bmatrix} K & -1 & \dots & -1 \\ -1 & K & \dots & -1 \\ \dots & \dots & \dots & \dots \\ -1 & \dots & \dots & K \end{bmatrix} \tag{4}$$

with

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

We have the following result.

Corollary 3 *The number of mistakes m made by the multitask Perceptron algorithm in Figure 1, run with the interaction matrix (4) on any finite multitask sequence of examples* $(\phi_1, y_1), (\phi_2, y_2), \ldots \in \mathbb{R}^{Kd} \times \{-1, 1\}$ *satisfies, for all* $u \in \mathbb{R}^{Kd}$,

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

where

$$u^\top A_\otimes u = \sum_{i=1}^K \|u_i\|^2 + \sum_{1 \le i < j \le K} \left\|u_i - u_j\right\|^2 .$$

In other words, running the Perceptron algorithm of Figure 1 with the interaction matrix (4) amounts to using the Euclidean distance to measure task relatedness. Alternatively, we can say that the regularization term of the regularized target functional favors task vectors $u_1, \ldots, u_K \in \mathbb{R}^d$ having small pairwise distances $\left\|u_i - u_j\right\|$.

Note that when all tasks are equal, that is when $u_1 = \cdots = u_K$, the bound of Corollary 3 becomes the standard Perceptron mistake bound (see, e.g., Cesa-Bianchi et al., 2005). In the general case of distinct u_i we have

$$\frac{2\left(u^\top A_\otimes u\right)}{K+1} = \frac{2K}{K+1} \sum_{i=1}^K \|u_i\|^2 - \frac{4}{K+1} \sum_{1 \le i < j \le K} u_i^\top u_j .$$

The sum of squares $\sum_{i=1}^K \|u_i\|^2$ 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 $u_i^\top u_j$) imply a large negative second term in the right-hand side of the above expression.

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

$$w_{j,s} = w_{j,s-1} + \begin{cases} \frac{b+K}{(1+b)K} y_t x_t & \text{if } j = i_t, \\ \frac{b}{(1+b)K} y_t x_t & \text{otherwise} \end{cases}$$

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

$$A = \frac{1}{K} \begin{bmatrix} a & -b & \ldots & -b \\ -b & a & \ldots & -b \\ \ldots & \ldots & \ldots & \ldots \\ -b & \ldots & \ldots & a \end{bmatrix} \tag{5}$$

with $a = K + b(K-1)$. It is immediate to see that the previous case (4) is recovered by choosing $b = K$. The inverse of (5) is

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

When (5) is used in the multitask Perceptron algorithm, Theorem 1 can be specialized to the following result.

Corollary 4 *The number of mistakes m made by the multitask Perceptron algorithm in Figure 1, run with the interaction matrix (5) on any finite multitask sequence of examples* $(\phi_1, y_1), (\phi_2, y_2), \ldots \in \mathbb{R}^{Kd} \times \{-1, 1\}$ *satisfies, for all* $u \in \mathbb{R}^{Kd}$,

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

where

$$u^\top A_\otimes u = \sum_{i=1}^{K} \|u_i\|^2 + bK\,\mathrm{VAR}[u]$$

being $\mathrm{VAR}[u] = \frac{1}{K}\sum_{i=1}^{K} \|u_i - \bar{u}\|^2$ *the "variance", of the task vectors, and* $\bar{u}$ *the centroid* $\frac{1}{K}\big(u_1 + \cdots + u_K\big)$.

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 $u_1, \ldots, u_K$) is

$$b = \max\left\{0, \sqrt{(K-1)\frac{\|\bar{u}\|^2}{\mathrm{VAR}[u]}} - 1\right\} .$$

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

$$\frac{(b+K)}{(1+b)K}\big(u^\top A_\otimes u\big) = \begin{cases} \|u_1\|^2 + \cdots + \|u_K\|^2 & \text{if } b = 0, \\ \Big(\|\bar{u}\| + \sqrt{K-1}\sqrt{\mathrm{VAR}[u]}\Big)^2 & \text{otherwise.} \end{cases}$$

When the variance $\mathrm{VAR}[u]$ is large (compared to the squared centroid norm $\|\bar{u}\|^2$), 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 $\|\bar{u}\|$ plus $\sqrt{K-1}$ times the standard deviation $\sqrt{\mathrm{VAR}[u]}$.

At the other extreme, if the variance $\mathrm{VAR}[u]$ is zero (namely, when all tasks coincide) then the optimal b grows unbounded, and the quadratic term $\frac{(b+K)}{(1+b)K}\big(u^\top A_\otimes u\big)$ tends to the average square norm $\frac{1}{K}\sum_{i=1}^{K}\|u_i\|^2$. 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 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 as 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 the $K \times K$ matrix L is the Laplacian of G, defined as

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

Here we denoted by d_i 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 5 *The number of mistakes m made by the multitask Perceptron algorithm in Figure 1, run with the interaction matrix $I + L$ on any finite multitask sequence of examples $(\phi_1, y_1), (\phi_2, y_2), \ldots \in \mathbb{R}^{Kd} \times \{-1, 1\}$ satisfies, for all $u \in \mathbb{R}^{Kd}$,*

$$m \le \sum_{t \in \mathcal{M}} \ell_t(u) + c_G\, u^\top (I+L)_\otimes u + \sqrt{c_G\, u^\top (I+L)_\otimes u \sum_{t \in \mathcal{M}} \ell_t(u)}$$

where

$$u^\top (I+L)_\otimes u = \sum_{i=1}^{K} \|u_i\|^2 + \sum_{(i,j) \in E} \left\|u_i - u_j\right\|^2 \tag{6}$$

and $c_G = \max_{i=1,\ldots,K} \sum_{j=1}^{K} \frac{v_{j,i}^2}{1+\lambda_j}$. Here $0 = \lambda_1 < \lambda_2 \le \cdots \le \lambda_K$ are the eigenvalues of the positive semidefinite matrix L, and $v_{j,i}$ denotes the i-th component[1] of the eigenvector v_j of L associated with eigenvalue λ_j.

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

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

If $v_1, \ldots, v_K$ are the eigenvectors of L, then

$$(I+L)^{-1} = \sum_{j=1}^{K} \frac{v_j v_j^\top}{1+\lambda_j}$$

which concludes the proof. ■

Ideally, we would like to have $c_G = O\left(\frac{1}{K}\right)$. 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 v_1 associated with the zero eigenvalue λ_1 is $\left(K^{-1/2}, \ldots, K^{-1/2}\right)$. Moreover, it is well known that all the remaining eigenvalues are equal to K—see, for example, Hogben (2006). 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

1. Note that the orthonormality of the eigenvectors implies $v_{1,i}^2 + \cdots + v_{K,i}^2 = 1$ for all i.

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 $\lambda_2 = \min\{|T_1|, |T_2|\}$.

The advantage of using a graph G with significantly fewer edges than the clique is that the sum of pairwise distances in (6) will contain less than $\binom{K}{2}$ terms. On the other hand, this reduction has to be contrasted to a larger coefficient c_G in front of $u^\top (I+L)_\otimes u$. 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). The role of prior knowledge is thus to avoid the insertion in A of edges connecting tasks that are hardly related, thus preventing the presence of large terms in the sum $u^\top (I+L)_\otimes u$.

4. Turning Perceptron-like Algorithms Into Multitask Classifiers

We now show how to obtain multitask versions of well-known classifiers by using the multitask kernel mapping detailed in Section 3.

4.1 The Multitask p-norm Perceptron Algorithm

We first consider the p-norm Perceptron algorithm of Grove et al. (2001) and Gentile (2003). 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.

The algorithm resembles the Perceptron algorithm and maintains its state in the compound *primal* weight vector $v_s \in \mathbb{R}^{Kd}$ where s stores the mistakes made so for (plus one). What sets the *multitask p-norm Perceptron* aside from the algorithm of Section 3 is that the prediction at time t is computed, for an arbitrary positive definite interaction matrix A, as $\mathrm{SGN}\big(w_{s-1}^\top A_\otimes^{-1} \phi_t\big)$ where the *dual* weight vector w_{s-1} is a (one-to-one) transformation of the weight vector v_{s-1}, specifically $w_{s-1} = \nabla \frac{1}{2}\|v_{s-1}\|_p^2$, with $p \geq 2$. If a mistake occurs at time t, $v_{s-1} \in \mathbb{R}^{Kd}$ is updated using the multitask Perceptron rule, $v_s = v_{s-1} + y_t A_\otimes^{-1}\phi_t$. We are now ready to state the mistake bound for the the multitask p-norm Perceptron algorithm. In this respect we focus on a specific choices of p and A.

Theorem 6 *The number of mistakes m made by the p-norm multitask Perceptron, run with the pairwise distance matrix (4) and $p = 2\ln\max\{K,d\}$, on any finite multitask sequence of examples $(\phi_1,y_1),(\phi_2,y_2),\ldots \in \mathbb{R}^{Kd} \times \{-1,1\}$ satisfies, for all $u \in \mathbb{R}^{Kd}$,*

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

where

$$H = \frac{8\,e^2 \ln\max\{K,d\}}{(K+1)^2} X_\infty^2 \left(\sum_{i=1}^{K} \Big\| u_i + \sum_{j\neq i}\big(u_i - u_j\big) \Big\|_1 \right)^2$$

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

Proof Let v_m be the primal weight vector after any number m of mistakes. By Taylor-expanding $\frac{1}{2}\|v_s\|_p^2$ around v_{s-1} for each $s = 1,\ldots,m$, and using the fact $y_t\, w_{s-1}^\top A_\otimes^{-1}\phi_t \leq 0$ whenever a mistake occurs at step t, we get

$$\frac{1}{2}\|v_m\|_p^2 \leq \sum_{s=1}^{m} D\left(v_s \| v_{s-1}\right) \tag{7}$$

where $D(v_s \| v_{s-1}) = \frac{1}{2}\left(\|v_s\|_p^2 - \|v_{s-1}\|_p^2\right) - y_t\, w_{s-1}^\top A_\otimes^{-1}\phi_t$ is the so-called Bregman divergence, that is, the error term in the first-order Taylor expansion of $\frac{1}{2}\|\cdot\|_p^2$ around vector v_{s-1}, at vector v_s.

Fix any $u \in \mathbb{R}^{Kd}$. Using the convex inequality for norms $u^\top v \le \|u\|_q \|v\|_p$ where $q = p/(p-1)$ is the dual coefficient of p (so that $\|\cdot\|_q$ is the dual norm of $\|\cdot\|_p$), and the fact that

$$u^\top A_\otimes v_s = u^\top A_\otimes v_{s-1} + y_t u^\top \phi_t \ge u^\top A_\otimes v_{s-1} + 1 - \ell_t(u),$$

one then obtains

$$\|v_m\|_p \ge \frac{u^\top A_\otimes v_m}{\|A_\otimes u\|_q} \ge \frac{m - \sum_{t\in\mathcal{M}} \ell_t(u)}{\|A_\otimes u\|_q} \,. \tag{8}$$

Combining (7) with (8) and solving for m gives

$$m \le \sum_{t\in\mathcal{M}} \ell_t(u) + \|A_\otimes u\|_q \sqrt{2\sum_{s=1}^{m} D(v_s \| v_{s-1})} \,. \tag{9}$$

Following the analysis contained in, for example, Cesa-Bianchi and Lugosi (2006), one can show that the Bregman term can be bounded as follows, for $t_s = t$,

$$D(v_s \| v_{s-1}) \le \frac{p-1}{2}\left\|A_\otimes^{-1}\phi_t\right\|_p^2 = \frac{p-1}{2}\|x_t\|_p^2 \left\|A_{\downarrow i_t}^{-1}\right\|_p^2$$

where $A_{\downarrow i_t}^{-1}$ is the i_t-th column of A^{-1}.

We now focus our analysis on the choice $p = 2\ln\max\{K,d\}$ which gives mistake bounds in the dual norms $\|u\|_1$ and $\|x_t\|_\infty$, and on the pairwise distance matrix (4). 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 Littlestone (1989). We now see that this property is preserved in the multitask extension. We have $\|x_t\|_p^2 \le e\,\|x_t\|_\infty^2$ and

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

As for the dual norm $\|A_\otimes u\|_q$, we get

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

Substituting into (9) gives the desired result. ■

The rightmost factor in the expression for H in the statement of Theorem 6 reveals the way similarity among tasks is quantified in this case. To gain some intuition, assume the task vectors u_i are all sparse (few nonzero coefficients). Then H is small when the task vectors u_i have a common pattern of sparsity; that is, when the nonzero coordinates tend to be the same for each task vector. In the extreme case when all task vectors are equal (and not necessarily sparse), H becomes

$$\left(\frac{K}{K+1}\right)^2 \left(8e^2\ln\max\{K,d\}\right)\left(\max_{t=1,\dots,n}\|x_t\|_\infty\right)^2 \|u_1\|_1^2 \,. \tag{10}$$

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 Littlestone (1989).

```
Parameters: Positive definite $K \times K$ interaction matrix $A$.
Initialization: $S_0 = \emptyset,\ v_0 = 0 \in \mathbb{R}^{Kd},\ s = 1$.
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 $x_t \in \mathbb{R}^d \ :\ ||x_t|| = 1$;
    2. Build the associated multitask instance $\phi_t \in \mathbb{R}^{Kd}$
       and compute $\widetilde{\phi}_t = \big(A \otimes I_d\big)^{-1/2}\phi_t$;
    3. Predict label $y_t \in \{-1, +1\}$ with $\widehat{y}_t = \mathrm{SGN}\big(w_{s-1}^\top \widetilde{\phi}_t\big)$,
       where $w_{s-1} = \Big(I + S_{s-1}S_{s-1}^\top + \widetilde{\phi}_t\widetilde{\phi}_t^\top\Big)^{-1} v_{s-1}$;
    4. Get label $y_t \in \{-1, 1\}$;
    5. If $\widehat{y}_t \neq y_t$ then update:
           $v_s = v_{s-1} + y_t\widetilde{\phi}_t\ , \quad S_s = \Big[S_{s-1}, \widetilde{\phi}_t\Big]\ , \quad s \leftarrow s + 1\ .$
```

Figure 2: The second-order multitask Perceptron algorithm.

Remark 7 *Note that for $p = 2$ our p-norm variant of the multitask Perceptron algorithm does not reduce to the multitask Perceptron of Figure 1. In order to obtain the latter as a special case of the former, we could use the fact that the multitask Perceptron algorithm is equivalent to the standard* 2*-norm Perceptron run on "multitask instances" $A_\otimes^{-1/2}\phi_t$—see Remark 2. One then obtains a proper p-norm generalization of the multitask Perceptron algorithm by running the standard p-norm Perceptron on such multitask instances. Unfortunately, this alternative route apparently prevents us from obtaining a bound as good as the one proven in Theorem 6. For example, when p is chosen as in Theorem 6 and all task vectors are equal, then multitask instances of the form $A_\otimes^{-1/2}\phi_t$ yield a bound K times worse than (10), which is obtained with instances of the form $A_\otimes^{-1}\phi_t$.*

Finally, we should mention that an alternative definition of the p-norm Perceptron for a related problem of predicting a labelled graph has been recently proposed in Herbster and Lever (2009).

4.2 The Multitask Second-order Perceptron Algorithm

We now turn to the second-order kernel Perceptron algorithm of Cesa-Bianchi et al. (2005). The algorithm, described in Figure 2, maintains in its internal state a matrix S (initialized to the empty matrix $\emptyset$) and a multitask Perceptron weight vector 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. The algorithm computes a tentative (inverse) matrix

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

Such a matrix is combined with the current Perceptron vector v_{s-1} to predict the label y_t. If the prediction $\widehat{y}_t$ and the label y_t disagree both v_{s-1} and S_{s-1} get updated (no update takes place otherwise). In particular, the new matrix S_s is augmented by padding with the current vector $\widetilde{\phi}_t$. Since supports are shared, the computational cost of an update is not significantly larger than that for learning a single-task (see Subsection 4.2.1).

Theorem 8 *The number of mistakes m made by the multitask Second Order Perceptron algorithm in Figure 2, run with an interaction matrix A on any finite multitask sequence of examples $(\phi_1,y_1),(\phi_2,y_2),\ldots \in \mathbb{R}^{Kd} \times \{-1,1\}$ satisfies, for all $u \in \mathbb{R}^{Kd}$,*

$$m \le \sum_{t\in\mathcal{M}} \ell_t(u) + \sqrt{\left(u^\top A_\otimes u + \sum_{t\in\mathcal{M}} \left(u_{i_t}^\top x_t\right)^2\right) \sum_{j=1}^m \ln(1+\lambda_j)}$$

where $\mathcal{M}$ is the sequence of mistaken trial indices and $\lambda_1,\ldots,\lambda_m$ are the eigenvalues of the matrix whose (s,t) entry is $x_s^\top A_{i_s,i_t}^{-1} x_t$, with $s,t \in \mathcal{M}$.

Proof From the mistake bound for the kernel second-order Perceptron algorithm (Cesa-Bianchi et al., 2005) we have, for all h in $\mathcal{H}$,

$$m \le \sum_{t\in\mathcal{M}} \ell_t(h) + \sqrt{\left(\|h\|_{\mathcal{H}}^2 + \sum_{t\in\mathcal{M}} h(\phi_t)^2\right) \sum_{i=1}^m \ln(1+\lambda_i)}$$

where $\lambda_1,\ldots,\lambda_m$ are the eigenvalues of the kernel Gram matrix including only time steps in $\mathcal{M}$. Making the role of $A_\otimes$ explicit in the previous expression yields

$$\|u\|_{\mathcal{H}}^2 = u^\top A_\otimes u$$

and

$$\langle u, \psi(x_t,i_t)\rangle_{\mathcal{H}}^2 = \left(u^\top A_\otimes A_\otimes^{-1} \phi_t\right)^2 = \left(u_{i_t}^\top x_t\right)^2.$$

Finally, the kernel Gram matrix has elements $\mathcal{K}\big(\psi(x_s,i_s),\psi(x_t,i_t)\big) = \phi_s^\top A_\otimes^{-1} \phi_t = x_s^\top A_{i_s,i_t}^{-1} x_t$, where $s,t \in \mathcal{M}$. This concludes the proof. ■

Again, this bound should be compared to the one obtained when learning K independent tasks. As in the Perceptron algorithm, we have the complexity term $u^\top A_\otimes u$. 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, when the tasks are close and A is the pairwise distance matrix, we essentially gain a factor $\sqrt{K}$ from the fact that $u^\top A_\otimes u$ is close to K times the complexity of the single task (according to the arguments in Section 3). On the other hand, the trace of the multitask Gram matrix $\left[\phi_s^\top A_\otimes^{-1} \phi_t\right]_{s,t\in\mathcal{M}} = \left[x_s^\top A_{i_s,i_t}^{-1} x_t\right]_{s,t\in\mathcal{M}}$ is about the same as the trace of the single task matrix, since the K times larger dimension of the multitask matrix is offset by the factor $1/K$ delivered by $A_\otimes^{-1}$ in $\left[\phi_s^\top A_\otimes^{-1} \phi_t\right]_{s,t\in\mathcal{M}}$ when compared to the single task Gram matrix $\left[x_s^\top x_t\right]_{s,t\in\mathcal{M}}$. So, in a sense, the spectral quantity $\sum_{j=1}^m \ln(1+\lambda_j)$ is similar to the corresponding quantity for the single task case. Putting together, unlike the first-order Perceptron, the gain factor achieved by a multitask second-order perceptron over the K independent tasks bound is about $\sqrt{K}$.

4.2.1 IMPLEMENTING THE MULTITASK SECOND-ORDER PERCEPTRON IN DUAL FORM

It is easy to see that the second-order multitask Perceptron can be run in dual form 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 Cesa-Bianchi et al., 2005, Theorem 3.3)

$$\begin{aligned} w_{s-1}^\top \widetilde{\phi}_t &= v_{s-1}^\top \left(I + S_{s-1} S_{s-1}^\top + \widetilde{\phi}_t \widetilde{\phi}_t^\top\right)^{-1} \widetilde{\phi}_t \\ &= y_s^\top \left(I + S_s^\top S_s\right)^{-1} S_s^\top \widetilde{\phi}_t \end{aligned} \tag{11}$$

where 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.

Note that replacing $I + S_s^\top S_s$ with $I + S_{s-1}^\top S_{s-1}$ in (11) does not change the sign of the prediction. The margin at time t can then be computed by calculating the scalar product between $S_s^\top \widetilde{\phi}_t$ and $y_s^\top \left(I + S_{s-1}^\top S_{s-1}\right)^{-1}$. Now, each entry of the vector $S_s^\top \widetilde{\phi}_t$ is of the form $A_{j,i_t}^{-1} x_j^\top x_t$, and thus computing $S_s^\top \widetilde{\phi}_t$ requires $O(s)$ inner products so that, overall, the prediction step requires $O(s)$ scalar multiplications and $O(s)$ inner products (independent of the number of tasks K).

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

$$I + S_s^\top S_s = \begin{bmatrix} I + S_{s-1}^\top S_{s-1} & S_{s-1}^\top \widetilde{\phi}_t \\ \widetilde{\phi}_t^\top S_{s-1} & 1 + \widetilde{\phi}_t^\top \widetilde{\phi}_t \end{bmatrix}.$$

Using standard facts about the inverse of partitioned matrices (see, e.g., Horn and Johnson, 1985, Ch. 0), one can see that the inverse of matrix $I + S_s^\top S_s$ can be computed from the inverse of $I + S_{s-1}^\top S_{s-1}$ with $O(s)$ extra inner products (again, independent of K) and $O(s^2)$ additional scalar multiplications.

5. The Simultaneous Multitask Protocol: Preliminaries

The multitask kernel-based regularization approach adopted in the previous sections is not the only way to design algorithms for the multiple tasks scenario. As a different strategy, we now aim at measuring tasks relatedness as a function of the dimension of the space spanned by the task reference vectors. In matrix terms, this may be rephrased by saying that we hope to speed up the learning process, or reduce the number of mistakes, whenever the matrix of reference vectors is spectrally sparse. For reasons that will be clear in a moment, and in order to make the above a valid and reasonable goal for a multitask algorithm, we now investigate the problem of *simultaneously* producing multiple predictions after observing the corresponding (multiple) instance vectors. We therefore extend the traditional online classification protocol to a fully simultaneous multitask environment where at each time step t the learner observes exactly K instance vectors $x_{i,t} \in \mathbb{R}^d$, $i = 1, \ldots, K$. The learner then outputs K predictions $\widehat{y}_{i,t} \in \{-1, +1\}$ and obtains the associated labels $y_{i,t} \in \{-1, +1\}$, $i = 1, \ldots, K$. We still assume that the K example sequences are adversarially generated and that $\|x_{i,t}\| = 1$. We call this setting the *simultaneous multitask setting*.

Once again the underlying rationale here is that one should be able to improve the performance over the baseline by leveraging the additional information conveyed through multiple instance vectors made available all at once, provided that the tasks to learn share common characteristics. Theoretically, this amounts to postulating the existence of K vectors $u_1, \dots, u_K$ such that *each* u_i is a good linear classifier for the corresponding sequence $(x_{i,1}, y_{i,1}), (x_{i,2}, y_{i,2}), \dots$ of examples. As before, the natural baseline is the algorithm that simultaneously run K independent Perceptron algorithms, each one observing its own sequence of examples and acting in a way that is oblivious to the instances given as input to and the labels observed by its peers. Of course, we now assume that this baseline outputs K predictions per trial. The expected performance of this algorithm is simply K times the one of a single Perceptron algorithm. An additional difference that sets the protocol and the algorithms discussed here apart from the ones considered in the previous sections is that the cumulative count of mistakes is not only over time but also over tasks; that is, at each time step more the one mistake might occur since $K > 1$ predictions are output.

In the next section we show that simultaneous multitask learning algorithms can be designed in such a way that the cumulative number of mistakes is, in certain relevant cases, provably better than the K independent Perceptron algorithm. The cases where our algorithms outperform the latter are exactly those when the overall information provided by the different example sequences are related, that is, when the reference vectors associated with different tasks are "similar", while the instance vectors received during each time step are unrelated (and thus overall more informative). These notions of similarity and unrelatedness among reference and instance vectors will be formally defined later on.

5.1 Notation and Definitions

We denote by $\langle M, N \rangle = \text{TR}\big(M^\top N\big)$, for $M, N \in \mathbb{R}^{d \times K}$ the Frobenious matrix inner product. Let $r = \min\{d, K\}$ and define the function $\sigma : \mathbb{R}^{d\times K} \to \mathbb{R}^r$ such that $\sigma(M) = \big(\sigma_1(M), \dots, \sigma_r(M)\big)$, where $\sigma_1(M) \ge \cdots \ge \sigma_r(M) \ge 0$ are the singular values of a matrix $M \in \mathbb{R}^{d\times K}$. In the following, we simply write σ_i instead of $\sigma_i(M)$ whenever the matrix argument is clear from the context.

Following Horn and Johnson (1991) we say that a function $f : \mathbb{R}^r \to \mathbb{R}$ is a *symmetric gauge function* if it is an absolute norm on $\mathbb{R}^r$ and is invariant under permutation of the components of its argument. We consider matrix norms of the form $\|\cdot\| : \mathbb{R}^{d\times K} \to \mathbb{R}$ such that $\|\cdot\| = f \circ \sigma$ where f is symmetric gauge function. A matrix norm is said unitarily (orthogonally, indeed, since we only consider matrices with real entries) invariant if $\|UAV\| = \|A\|$ for any matrix A and for any unitary (orthogonal) matrices U and V for which UAV is defined. It is well known that a matrix norm is unitarily invariant if and only if it is a symmetric gauge function of the singular values of its argument.

One important class of unitarily invariant norms is given by the Schatten p-norms, $\|U\|_{s_p} \overset{\text{def}}{=} \|\sigma(U)\|_p$, where the right-hand expression involves a vector norm. Note that the Schatten 2-norm is the Frobenius norm, while for $p = 1$ the Schatten p-norm becomes the trace norm $\|U\|_{s_1} = \|\sigma(U)\|_1$, which is a good proxy for the rank of U, $\|\sigma(U)\|_0$.

Let M be a matrix of size $d \times K$. We denote by $\text{VEC}(M)$ the vector of size Kd obtained by stacking the columns of M one underneath the other. Important relationships can be established among the Kronecker product, the VEC operator and the trace operator. In particular, we have

$$\text{VEC}(MNO) \quad = \quad (O^\top \otimes M)\text{VEC}(N) \tag{12}$$

for any M,N,O for which MNO is defined, and

$$\mathrm{VEC}(M)^\top \mathrm{VEC}(N) \quad = \quad \mathrm{TR}(M^\top N) \tag{13}$$

for any M,N of the same order. We denote by T_{K^2} the $K^2 \times K^2$ commutation matrix such that $T_{K^2}\mathrm{VEC}(M) = \mathrm{VEC}(M^\top)$. We recall that T_{K^2} also satisfies $T_{K^2}(M \otimes N) = (M \otimes N)T_{K^2}$ for any $M,N \in \mathbb{R}^{d\times K}$.

We rely on the notation introduced by Magnus and Neudecker (1999) to derive calculus rules for functions defined over matrices. Given a differentiable function $F : \mathbb{R}^{m\times p} \to \mathbb{R}^{n\times q}$, we define the Jacobian of F at M as the matrix $\nabla F(M) \in \mathbb{R}^{nq\times mp}$

$$\nabla F(X) = \frac{\partial \mathrm{VEC}\big(F(M)\big)}{\partial \mathrm{VEC}(M)^\top} \ . \tag{14}$$

It is easy to see that (14) generalizes the well-known definition of Jacobian for vector valued functions of vector variables. The following rules, which hold for any matrix $M \in R^{K\times K}$, can be seen as extensions of standard vector derivation formulas

$$\nabla \mathrm{TR}(M^p) \quad = \quad p\,\mathrm{VEC}(M^{p-1})^\top \qquad\qquad p = 1,2,\ldots \tag{15}$$

$$\nabla M^\top M \quad = \quad (I_{K^2} + T_{K^2})(I_K \otimes M^\top) \ . \tag{16}$$

6. The Potential-based Simultaneous Multitask Classifier

As discussed in Section 5, a reasonable way to quantify the similarity among reference vectors, as well as the unrelatedness among example vectors, is to arrange such vectors into matrices, and then deal with special properties of these matrices. In order to focus on this concept, we lay out vectors as columns of $d \times K$ matrices and extend the dual norm analysis of Subsection 4.1 to matrices. The idea is to design a classifier which is able to perform much better than the K independent Perceptron baseline discussed in Section 5 whenever the set of reference vectors $u_i \in \mathbb{R}^d$ (arranged into a $d \times K$ reference matrix U), have some matrix-specific, for example, *spectral*, properties.

Our potential-based matrix algorithm for classification shown in Figure 3 generalizes the classical potential-based algorithms operating on vectors to simultaneous multitask problems with matrix examples. This family of potential-based algorithms has been introduced in the learning literature by Kivinen and Warmuth (2001) and Grove et al. (2001), and by Nemirovski and Yudin (1978) and Beck and Teboulle (2003) in the context of nonsmooth optimization. The algorithm maintains a $d \times K$ matrix W. Initially, W_0 is the zero matrix. If $s-1$ updates have been made in the first $t-1$ time steps, then the K predictions at time t are $\mathrm{SGN}\big(w_{i,s-1}^\top x_{i,t}\big)$, $i = 1,\ldots,K$, where the vector $w_{i,s-1} \in \mathbb{R}^d$ is the i-th column of the the $d \times K$ matrix W_s and $x_{i,t} \in \mathbb{R}^d$ is the instance vector associated with the i-th task at time t. An update is performed if at least one mistake occurs. When the s-th update occurs at time t then W_s is computed as

$$W_s = \nabla \frac{1}{2}\,\|V_s\|^2$$

where, in turn, the columns of the $d \times K$ matrix V_s are updated using the Perceptron rule,[2] $v_{i,s} = v_{i,s-1} + y_{i,t}\, x_{i,t}\, \mathbb{1}_{\{\widehat{y}_{i,t} \neq y_{i,t}\}}$ which, as in the basic Perceptron algorithm, is mistake driven. In other

2. Here and throughout this section, $\mathbb{1}_{\{\widehat{y}_{i,t} \neq y_{i,t}\}}$ denotes the indicator function which is 1 if the label associated with the i-th task is wrongly predicted at time t, and 0 otherwise.

```
Parameters: Unitarily invariant norm $\|\cdot\|$.
Initialization: $V_0 = [v_{0,0}, \dots, v_{K,0}] = 0$, $W_0 = [w_{0,0}, \dots, w_{K,0}] = \nabla \frac{1}{2} \|V_0\|^2$, $s = 1$.
At each time $t = 1, 2, \dots$ do the following:
    1. Get multitask instance vectors $x_{1,t}, \dots, x_{K,t} \in \mathbb{R}^d$;
    2. Predict labels $y_{i,t} \in \{-1,+1\}$ with $\widehat{y}_{i,t} = \mathrm{SGN}(w_{i,s-1}^\top x_{i,t})$,    $i = 1, \dots, K$;
    3. Get labels $y_{i,t} \in \{-1,1\}$,    $i = 1, \dots, K$;
    4. If $\widehat{y}_{i,t} \neq y_{i,t}$ for some $i$ then update:
           $v_{i,s} = v_{i,s-1} + y_{i,t} x_{i,t} \mathbb{1}_{\{\widehat{y}_{i,t} \neq y_{i,t}\}}$    $i = 1, \dots, K$
           $W_s = \nabla \frac{1}{2} \|V_s\|^2$
           $s \leftarrow s + 1$ .
```

Figure 3: The potential-based matrix algorithm for the simultaneous multitask setting.

words, the i-th column in V_{s-1} is updated if and only if the label associated with the i-th task was wrongly predicted. We say that W_s is the dual matrix weight associated with the primal matrix weight V_s. So far we left the mapping from V_s to W_s partially unspecified since we did not say anything other than it is the gradient of some unitarily invariant (squared) norm.

6.1 Analysis of Potential-based Matrix Classifiers

We now develop a general analysis of potential-based matrix algorithms for (simultaneous) multitask classification. Then, in Section 6.2 we specialize it to Schatten p-norms. The analysis proceeds along the lines of the standard proof for potential-based algorithms. Before turning to the details, we introduce a few shorthands. Let $\mathbb{1}_{i,t} = \mathbb{1}_{\{\widehat{y}_{i,t} \neq y_{i,t}\}}$, and $\mathbb{1}_t$ be the K-dimensional vector whose i-th component is $\mathbb{1}_{i,t}$. Also, e_i denotes the i-th vector of the standard basis for $\mathbb{R}^K$. Finally, we define the matrix $M_t = \sum_{i=1}^K \mathbb{1}_{i,t} y_{i,t} x_{i,t} e_i^\top$ whose i-th column is the example vector $y_{i,t} x_{i,t}$ if the label $y_{i,t}$ was wrongly predicted at time t, or the null vector otherwise. It is easy to see that, by using this notation, the update of the primal weight matrix V can be written as $V_s = V_{s-1} + M_t$.

Let $\mathcal{M}$ be the set of trial indices where at least one mistake occurred over the K tasks, and set $m = |\mathcal{M}|$. We start by Taylor-expanding $\frac{1}{2} \|V_s\|^2$ around V_{s-1} for each $s = 1, \dots, m$ and obtain

$$\frac{1}{2} \|V_m\|^2 \le \sum_{s=1}^{m} D(V_s \| V_{s-1}) \tag{17}$$

where $D(V_s \| V_{s-1}) = \frac{1}{2}\big(\|V_s\|^2 - \|V_{s-1}\|^2\big) - \langle W_{s-1}, M_t \rangle$ is the matrix Bregman divergence associated with $\nabla \frac{1}{2}\|\cdot\|^2$. The upper bound in (17) follows from

$$\langle W_{s-1}, M_t \rangle = \operatorname{TR}(W_{s-1}^\top M_t) = \sum_{i=1}^{K} \mathbb{1}_{i,t} y_{i,t} w_{s-1}^\top x_{i,t} \le 0$$

the last inequality holding because $\mathbb{1}_{i,t}$ is 1 if only if a mistake in the prediction for the i-th task occurs at time t.

Fix any $d \times K$ comparison matrix U and denote by $\|\cdot\|_*$ the matrix dual norm. By the convex inequality for matrix norms we have $\|V_m\| \, \|U\|_* \ge \langle V_m, U \rangle$, where $\|U\|_* = f^*\big(\sigma(U)\big)$ and f^* is the Legendre dual of function f—see Lewis (1995, Theorem 2.4). From $\langle U, V_s \rangle = \langle U, V_{s-1} \rangle + \langle U, M_t \rangle$. we obtain

$$\|V_m\| \ge \frac{\langle U, V_m \rangle}{\|U\|_*} \ge \frac{\sum_{t \in \mathcal{M}} \|\mathbb{1}_t\|_1 - \sum_{t \in \mathcal{M}} \ell_t^{\mathbb{1}}(U)}{\|U\|_*}$$

where

$$\ell_t^{\mathbb{1}}(U) \overset{\text{def}}{=} \sum_{i=1}^{K} \mathbb{1}_{i,t} \big[1 - y_{i,t} u_i^\top x_{i,t}\big]_+ = \sum_{i=1}^{K} \mathbb{1}_{i,t} \ell_t(u_i)$$

and $\|\mathbb{1}_t\|_1$ counts the number of mistaken tasks at time t. Solving for $\mu = \sum_{t \in \mathcal{M}} \|\mathbb{1}_t\|_1$ gives

$$\mu \le \sum_{t \in \mathcal{M}} \ell_t^{\mathbb{1}}(U) + \|U\|_* \sqrt{2 \sum_{s=1}^{m} D(V_s \| V_{s-1})} \, . \tag{18}$$

Equation (18) is our general starting point for analyzing potential-based matrix multitask algorithms. In particular, the analysis reduces to bounding from above the Bregman term for the specific matrix norm under consideration.

6.2 Specialization to Schatten p-norms

In this section we focus on Schatten p-norms, therefore measuring similarity (or dissimilarity) in terms of spectral properties. This amounts to saying that a set of reference vectors are similar if they span a low dimensional subspace. Along the same lines, we say that a set of K example vectors are dissimilar if their spanned subspace has dimension close to K. The rank of a matrix whose columns are either the reference vectors or the example vectors exactly provides this information. Here we use certain functions of the singular values of a matrix as proxies for its rank. It is easy to see that this leads to a kind of regularization that is precisely enforced through the use of unitarily-invariant norms. In fact, unitarily-invariant matrix norms control the distribution of the singular values of U, thus acting as spectral co-regularizers for the reference vectors—see, for example, Argyriou et al. (2008) for recent developments on this subject. In different terms, by relying only on the singular values (or on the magnitudes of principal components), unitarily invariant norms are a natural way to determine and measure the most informative directions for a given set of vectors.

For these reasons we now specialize the potential-based matrix classifier of Figure 3 to the Schatten $2p$-norm and set $\|V\| = \|V\|_{s_{2p}} = \|\sigma(V)\|_{2p}$, 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,

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

We are now ready to state our main result of this section. The proof, along with surrounding comments, can be found in the appendix.

Theorem 9 *The overall number of mistakes μ made by the $2p$-norm matrix multitask Perceptron (with p positive integer) run on finite sequences of examples $(x_{i,1}, y_{1,t}), (x_{i,2}, , y_{i,2}), \dots \in \mathbb{R}^{d\times K} \times \{-1,+1\}$, for $i = 1, \dots, K$, satisfies, for all $U \in \mathbb{R}^{d\times K}$,*

$$\mu \le \sum_{t\in\mathcal{M}} \ell_t^{\mathbb{1}}(U) + (2p-1)\left(\mathbf{M}_{s_{2p}} \|U\|_{s_{2q}}\right)^2 + \mathbf{M}_{s_{2p}} \|U\|_{s_{2q}} \sqrt{(2p-1)\sum_{t\in\mathcal{M}} \ell_t^{\mathbb{1}}(U)}$$

where

$$\mathbf{M}_{s_{2p}} = \max_{t\in\mathcal{M}} \frac{\|M_t\|_{s_{2p}}}{\sqrt{\|\mathbb{1}_t\|_1}}$$

and $\|U\|_{s_{2q}}$ is the Schatten $2q$-norm of U, with $2q = \frac{2p}{2p-1}$.

Remark 10 *In order to verify that in certain cases the bound of Theorem 9 provides a significant improvement over the K independent Perceptron baseline, we focus on the linearly separable case; that is, when the sequences $(x_{i,1}, y_{i,1}), (x_{i,2}, y_{i,2}), \dots$ are such that there exists a matrix $U \in \mathbb{R}^{d\times K}$ whose columns u_i achieve a margin of at least 1 on each example: $y_{i,t} u_i^\top x_{i,t} \ge 1$ for all $t = 1, 2, \dots$ and for all $i = 1, \dots, K$. In this case the bound of Theorem 9 reduces to*

$$\mu \le (2p-1)\left(\mathbf{M}_{s_{2p}} \|U\|_{s_{2q}}\right)^2 . \tag{19}$$

It is easy to see that for $p = q = 1$ the $2p$-norm matrix multitask Perceptron decomposes into K independent Perceptrons, which is our baseline. On the other hand, similarly to the vector case, a trace norm/spectral norm bound can be established when the parameter p is properly chosen. Note first that for basic properties of norms $\|U\|_{s_{2q}} \le \|U\|_{s_1}$ and $\|M\|_{s_{2p}} \le r^{1/(2p)} \|M\|_{s_\infty}$, with $r = \min\{d, K\}$. It now suffices to set $p = \lceil \ln r \rceil$ in order to rewrite (19) as

$$\mu \le (2\ln r + 1) e \left(\mathbf{M}_{s_\infty} \|U\|_{s_1}\right)^2$$

where U is now penalized with the trace norm and M_t is measured with the spectral norm $\|\cdot\|_{s_\infty}$. If the columns of U span a subspace of dimension $\ll K$, and the matrices of mistaken examples M_t tend to have K nonzero singular values of roughly the same magnitude, then $\|U\|_{s_1} \approx \|U\|_{s_2}$ while $\mathbf{M}^2_{s_\infty} \approx \mathbf{M}^2_{s_2}/K$. Hence this choice of p may lead to a factor K improvement over the bound achieved by the independent Perceptron baseline. See also Remark 11 below. Note that in Theorem 9 (and in the above argument) what matters the most is the quantification in terms of the spectral properties of U via $\|U\|_{s_{2q}}$. The fact that p has to be a positive integer is not a big limitation here, since $2q = \frac{2p}{2p-1}$ can be made arbitrarily close to 1 anyway.

Remark 11 *The bound of Theorem 9 is not in closed form, since the terms $\|\mathbb{1}_t\|_1$ occur in both the left-hand side (via μ) and in the right-hand side (via $\mathbf{M}_{s_{2p}}$). These terms play an essential role to assess the potential advantage of the $2p$-norm matrix multitask Perceptron. In order to illustrate the influence of $\|\mathbb{1}_t\|_1$ on the bound, let us consider the two extreme cases $\|\mathbb{1}_t\|_1 = 1$ for all $t \in \mathcal{M}$, and $\|\mathbb{1}_t\|_1 = K$ for all $t \in \mathcal{M}$. In the former case, the right-hand side of (19) becomes $(2p-1)\|U\|^2_{s_{2q}}$*

(since $\mathbf{M}_{s_{2p}} = 1$), which is always *worse than the baseline case $p = q = 1$. In the latter case, the bound becomes*

$$(2p-1)\,\frac{\left(\max_{t\in\mathcal{M}}\|M_t\|_{s_{2p}}^2\right)\|U\|_{s_{2q}}^2}{K}$$

which, according to the discussion in the previous remark, opens up the possibility for the factor K improvement. This is precisely the reason why the spectral regularization is not advantageous in the adversarially chosen task framework described in Section 2. Since the K instance-label pairs are the information obtained for each task at any given time step, it appears reasonable that a multitask approach has a chance to improve when such information is abundant, as is the case when $\|\mathbb{1}_t\|_1 = K$, and, at the same time, the tasks to be learned are sufficiently similar. For example, in the extreme case when the K tasks do actually coincide, it is as if we had to learn a single task, but received K independent pieces of information per time step, rather than just one.

Remark 12 *The $2p$-norm matrix multitask Perceptron algorithm updates the primal vector associated with a given task whenever an example for that task is wrongly predicted. Specifically, at time t the mistake-driven update rule for the i-th task vector is defined as $v_{i,s} = v_{i,s-1} + y_{i,t}\,x_{i,t}\,\mathbb{1}_{\{\widehat{y}_{i,t}\neq y_{i,t}\}}$. It is now straightforward to generalize the above update mechanism to the shared loss framework of Dekel et al. (2007), where the sharing is performed via a norm applied to the vector of task losses. Let $\ell_t(W)$ be the vector whose entries are the hinge losses incurred by the K columns of W and pick any vector norm $\|\cdot\|$. The goal is to bound the cumulative shared loss $\sum_t \|\ell_t(W)\|$. To do so, introduce an additional parameter $C > 0$ and write the update as $v_{i,s} = v_{i,s-1} + y_{i,t}\,x_{i,t}\,\tau_{i,t}$, where the vector $\tau_t = [\tau_{1,t},\dots,\tau_{K,t}]^\top$ is such that $\tau_t = \arg\max_{\tau:\|\tau\|_*\le C}\tau^\top\ell_t(W_{s-1})$ and $\|\cdot\|_*$ is the dual of $\|\cdot\|$. Since each entry of τ_t is dependent on all the K hinge losses suffered by the $2p$-norm matrix multitask Perceptron algorithm at time t, the update now acts so as to favor certain tasks over the others according to the shared loss induced by $\|\cdot\|$. By adapting our proof to the analysis given in Dekel et al. (2007), it is not hard to show that*

$$\sum_{t=1}^{T}\|\ell_t(W_{t-1})\| \le \sum_{t=1}^{T}\|\ell_t(U)\| + \frac{\|U\|_{S_{2q}}^2}{2C} + \frac{T\mathbf{M}_{s_{2p}}^2}{2}$$

where, in analogy with our previous definitions,

$$\mathbf{M}_{s_{2p}} = \max_{t=1,\dots,T}\frac{\|M_t\|_{s_{2p}}}{\sqrt{\|\tau_t\|_*}} \qquad \textit{and} \qquad M_t = \sum_{i=1}^{K}\tau_{i,t}\,y_{i,t}\,x_{i,t}\,e_i^\top\ .$$

Observe that $\mathbf{M}_{s_{2p}}$ depends on C through τ_t, thus preventing an easy optimization over C. Moreover, since the upper bound depends on dual Schatten $2p$-norms, the discussions in Remark 10 and Remark 11 still apply, with the caveat that in order to have $\mathbf{M}_{s_\infty}^2 \approx \mathbf{M}_{s_2}^2/K$ it must be $\tau_{1,t}\approx\cdots\approx\tau_{K,t}$.

6.2.1 IMPLEMENTATION IN DUAL FORM

As for the algorithms in previous sections, the $2p$-norm matrix multitask Perceptron algorithm can also be implemented in dual variables. Setting $X_t = [x_{1,t},\dots,x_{K,t}]$, it suffices to observe that the predictions $\widehat{y}_{i,t} = \mathrm{SGN}\big(w_{i,s-1}^\top x_{i,t}\big)$ of the $2p$-norm matrix Perceptron reduces to computing the sign of the diagonal entries of the matrix $(V_{s-1}^\top V_{s-1})^{p-1}V_{s-1}^\top X_t$—recall the expression for ∇G calculated in the proof of Theorem 9. 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 $x_{i,t}$ only through inner products. This allows us to turn our $2p$-norm matrix multitask Perceptron into a kernel-based algorithm, and repeat the analysis given here using a standard RKHS formalism—see Warmuth (2009) for a more general treatment of kernelizable matrix learning algorithms.

7. Experiments

We evaluated our multitask algorithms on several real-world data sets. Since we are more interested in the multitask kernel for sequential learning problems rather than the nature of the underlying classifiers, we restricted the experiments for the adversarially chosen task model to the multitask Perceptron algorithm of Section 3. In particular, we compare the performance of the multitask Perceptron algorithm with parameter $b > 0$ to that of the same algorithm run with $b = 0$, which is our multitask baseline. Recall from Subsection 3.2 that $b = 0$ amounts to running an independent standard Perceptron on each task. We also evaluated the $2p$-norm matrix multitask Perceptron algorithm under a similar experimental setting, reporting the achieved performance for different values of the parameter p. Finally, we provide experimental evidence of the effectiveness of the $2p$-norm matrix multitask Perceptron algorithm when applied to a learning problem which requires the simultaneous processing of a significant number of tasks.

In our initial experiments, we empirically evaluated the multitask kernel using a collection of data sets derived from the first 160,000 newswire stories in the Reuters Corpus Volume 1 (RCV1, for details see NIST, 2004). Since RCV1 is a hierarchical multiclass and multilabel data set, we could not use it right away. In fact, in order to evaluate the performance of our multitask algorithms in the presence of increasing levels of correlation among target tasks, we derived from RCV1 a collection of data sets where each example is associated with one task among a set of predefined tasks. We generated eight multitask data sets (D1 through D8) in such a way that tasks in different data sets have different levels of correlation, from almost uncorrelated (D1) to completely overlapped (D8). The number of tasks in each of the eight data sets was set to four.

Roughly speaking, we hand-crafted tasks by clustering categories from the original data set. We started from non intersecting sets of categories, which represent nonrelated tasks, and from there we progressively enlarged the intersection areas, thus obtaining tasks which get closer and closer. The whole process involved several steps. We first defined tasks as sets of RCV1 categories (RCV1 is a multilabel data set where labels are sets of hierarchically organized categories). In order to obtain the four tasks in D1, we first chose four subsets of categories from the initial set of all categories in the RCV1 taxonomy in such a way that each subset is both made up of hierarchically related categories and contains at least 15% of positive examples. More precisely, each of the four tasks in D1 is made up of second-level and third-level categories from one of the four main RCV1 sub-trees (CORPORATE/INDUSTRIAL, ECONOMICS, GOVERNMENT/SOCIAL, MARKETS). Since categories in different tasks belong to *different* sub-trees in the RCV1 taxonomy, and each task is composed by categories from the *same* sub-tree, the resulting four tasks in D1 describe very different but consistent topics. Tasks in D2-D8 are generated as follows. First, task one is kept the same in all the eight data sets. As for the other three tasks, we progressively added categories from the first task and dropped some from their own set of categories. We repeated this process seven times. During the first three times (corresponding to data sets D2, D3, and D4) we augmented task two to four with topics from task one; during the last four times (corresponding to data sets D5-D8) we progressively dropped their own initial categories. The whole process is illustrated in Figure 4. As a

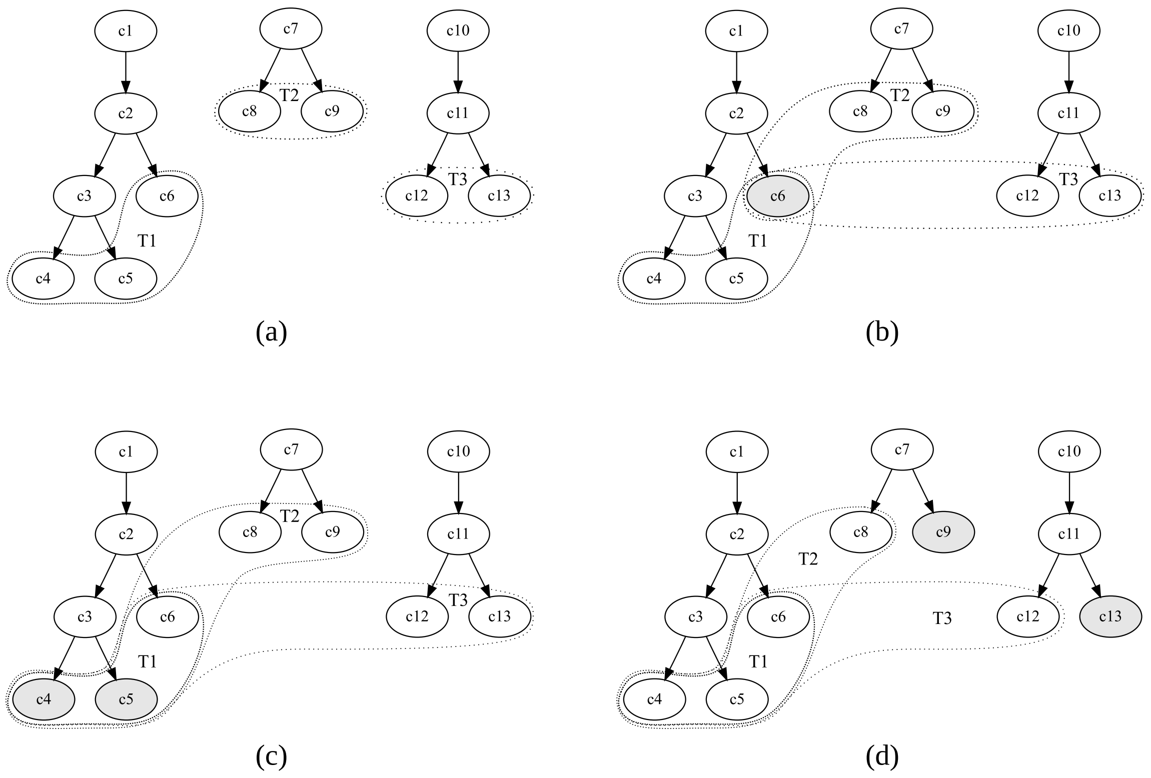

Figure 4: A sample of the task generation process given a taxonomy of 13 categories with three main sub-hierarchies. Tasks are marked as T1, T2 and T3. (a) Uncorrelated tasks are initially defined as sets of categories from different sub-hierarchies. (b and c) Partially overlapped tasks are obtained by first augmenting T2 and T3 with the addition of categories from T1 (category c6 first, then categories c4 and c5), then (d) by shrinking both T2 and T3 with the removal of their initial nodes (these are categories c9 and c13). The shrinking step is repeated until T1, T2 and T3 do coincide.

result of the above construction, as we go from D1 to D8, tasks two, three, and four get both closer to each other and to task one. The last set of four tasks (corresponding to data set D8) is made up of four occurrences of the first task, that is, tasks are completely overlapped in D8.

Once the eight sets of four tasks have been chosen, we generated labels for the corresponding multitask examples as follows. We went through the whole RCV1 data set (whose news example are sorted in chronological order), and gathered examples four by four, where the first example is associated with the first task, the second with the second task, and so on. A multitask example, defined as a set of four (instance, binary label) pairs, was then derived by replacing, for each of the four RCV1 examples, the original RCV1 categories with -1 if the intersection between the associated task and the categories was empty (i.e., if the example did not belong to any of the RCV1 categories which are part of that task), $+1$ otherwise. Since we used 160,000 multilabel and multiclass examples, this process ended up with eight multitask data sets of 40,000 (multitask) examples each.

Subsequences	D1	D2	D3	D4	D5	D6	D7	D8
1-10,000	0.159	0.198	0.215	0.212	0.208	0.203	0.195	0.175
10,001-20,000	0.116	0.158	0.167	0.170	0.167	0.164	0.152	0.134
20,001-30,000	0.104	0.141	0.158	0.155	0.150	0.147	0.138	0.122
30,001-40,000	0.085	0.118	0.125	0.125	0.119	0.113	0.105	0.091

Subsequences	D1	D2	D3	D4	D5	D6	D7	D8
1-10,000	0.395	0.482	0.508	0.499	0.489	0.492	0.461	0.410
10,001-20,000	0.297	0.394	0.428	0.427	0.410	0.394	0.371	0.322
20,001-30,000	0.274	0.374	0.399	0.389	0.375	0.368	0.332	0.291
30,001-40,000	0.231	0.323	0.339	0.337	0.323	0.315	0.287	0.249

Table 1: Online training error rates made by the baseline $b = 0$ on consecutive sequences of multitask examples after a single pass on the data sets D1-D8. The task i_t is chosen either randomly (top) or adversarially (bottom).

We note that the tasks considered here are not linearly separable and, as result of the above construction, different tasks in each data set may have different degrees of nonseparability. This explains why the baseline error rates for data sets D1-D8 are different—see Tables 1 and 2.

Figure 5 shows the fraction of wrongly predicted examples during the online training of the multitask Perceptron algorithm with interaction matrix (5), when the task i_t is chosen either randomly[3] (left) or in an adversarial manner (right). The latter means that i_t is selected so as the resulting signed margin is smallest over the four tasks. This implies that in the first case each task is invoked on average 10,000 times. Nothing can be said in hindsight about the task choices for the adversarial criterion, since this choice is heavily dependent on the online behavior of the classifiers and the noise in the tasks at hand. In both cases we show to what extent the incurred cumulative training error for different values of parameter b exceeds the one achieved by the multitask baseline $b = 0$, depicted here as a straight horizontal line (a negative value means that the chosen value of b achieves an error lower than $b = 0$). Recall that $b = 0$ amounts to running four independent Perceptron algorithms, while $b = 4$ corresponds to running the multitask Perceptron algorithm with the interaction matrix (4). The actual fractions of training error mistakes achieved by the baseline $b = 0$ are reported in Table 1. In order to illustrate how the generalization capabilities of our algorithms progress over time, four pairs of plots are reported, each one showing, from top to bottom, the fraction of mistakes occurred in the example subsequences 1-10,000, 10,001-20,000, 20,001-30,000, and 30,001-40,000, respectively.

Figure 5 confirms that multitask algorithms get more and more competitive as tasks get closer (since tasks are uncorrelated in D1 and totally overlapped in D8). Unsurprisingly, this advantage is higher as we increase the value of b. In fact, Figure 5 clearly shows that the delta errors from $b = 0$ decrease faster, when going from D1 to D8, as we increase b. This amounts to saying that the more we bias our algorithm towards a possible correlation the more we benefit from an actual correlation among tasks. Moreover, it is worth observing that the (rather conservative) choice $b = 1$ obtains an

3. In this case the results are averaged over 3 runs. The observed results were within a 0.002 interval of the plotted values in all 3 runs.

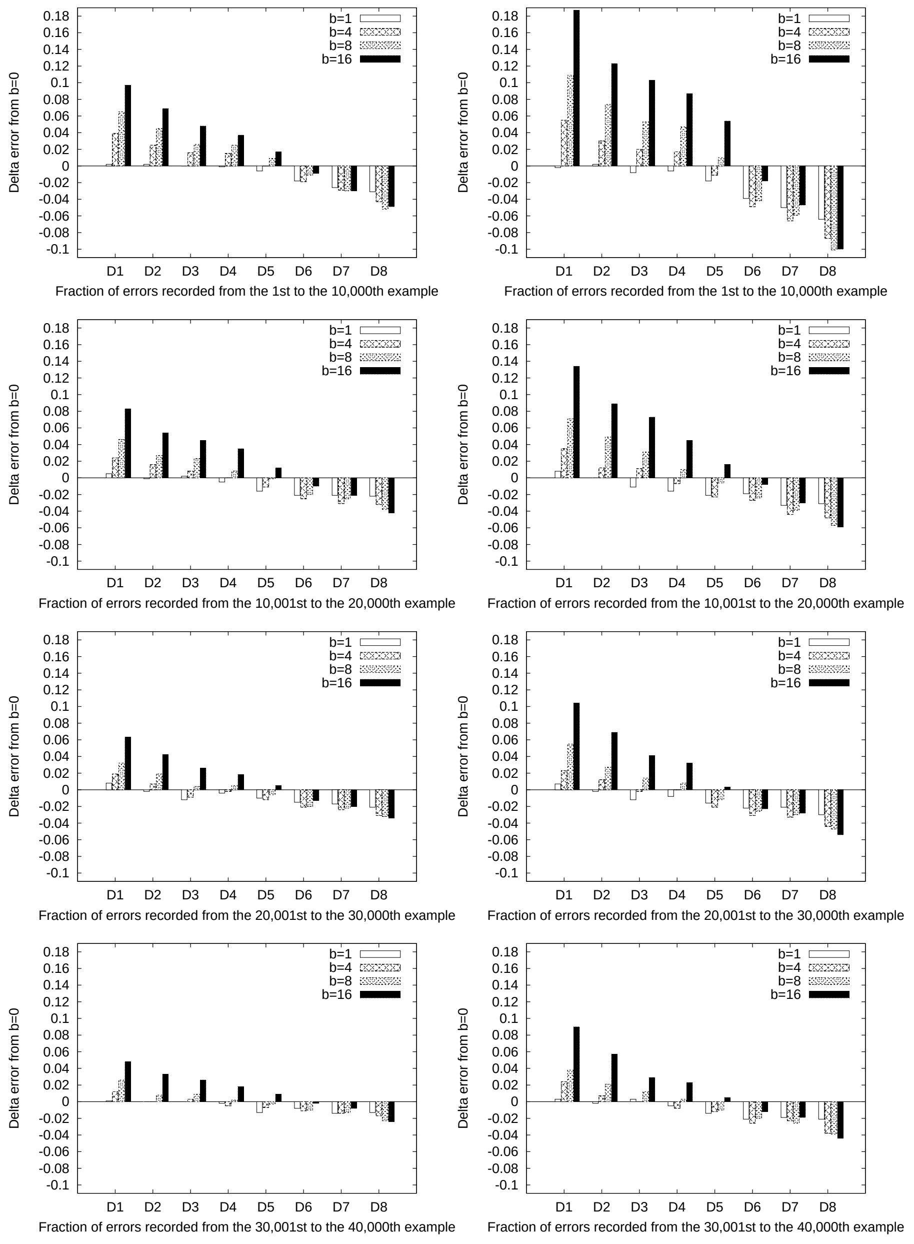

Figure 5: Online behavior of the multitask Perceptron algorithm. We report the extent to which during a single pass over the entire data set the fraction of training mistakes made by the multitask Perceptron algorithm (with $b = 1,4,8,16$) exceeds the one achieved by the baseline $b = 0$, represented here as an horizontal line. On the x-axis are the multitask data sets whose tasks have different levels of correlations, from low (D1) to high (D8). Task indices are randomly chosen in the left plots and adversarially selected in the right ones. The pair of plots on top reports the online training behavior on the first $10,000$ (multitask) examples, the second from top refers to the second $10,000$ examples, and so on.

Subsequences	D1	D2	D3	D4	D5	D6	D7	D8
1-10,000	0.154	0.198	0.207	0.206	0.202	0.196	0.185	0.166
10,001-20,000	0.104	0.141	0.159	0.157	0.152	0.149	0.135	0.120
20,001-30,000	0.095	0.126	0.139	0.138	0.131	0.127	0.115	0.099
30,001-40,000	0.085	0.125	0.132	0.135	0.132	0.124	0.117	0.101

Table 2: Fractions of training errors made by the baseline $p = 1$ on consecutive sequences of multitask examples after a single pass on the data sets D1-D8.

Parameters	Training Error	Test Error	StdDev
P=1	19.0%	13.4%	1.7%
P=2	16.7%	10.9%	1.3%
P=3	16.4%	10.6%	1.6%
P=4	16.6%	9.7%	1.5%
P=5	17.1%	10.2%	1.9%

Table 3: Training errors recorded after a single pass over the spam data set, along with the corresponding test errors. The values are averaged over 40 repetitions of a 10-fold cross-validation scheme. The standard deviation for the test error is reported in parentheses. The standard deviation for the training error is negligible and is therefore omitted.

overall better performance than the multitask baseline $b = 0$, with a higher cumulative error only on the first data set.

We then evaluated the $2p$-norm matrix multitask Perceptron algorithm on the same data set. In this case, we dropped the task choice mechanism since the algorithm is designed to receive all the four instance vectors and to output four predicted labels at each time step. We limited our experiments to the first 10,000 multitask examples. This allows us to make the results achieved by the $2p$-norm matrix multitask Perceptron algorithm somehow comparable (i.t.o. total number of binary labels received) with the ones achieved by the multitask Perceptron algorithm under the random task selection model (the four plots on the left in Figure 5). In Figure 6 we report the (differential) fractions of online training mistakes made by the algorithm on the subsequences 1-2,500, 2,501-5,000, 5,001-7,500 and 7,501-10,000, of multitask examples. The actual fractions of online training error mistakes achieved by the baseline $p = 1$ are showed in Table 2. As expected, the more the tasks get closer to each other the less is the number of wrongly predicted labels output by the $2p$-norm matrix multitask Perceptron algorithm and the larger is the gap from the baseline. In particular, it can be observed that even in D1 the performance of the $2p$-norm matrix multitask Perceptron algorithm is no worse than the one achieved by the baseline. In fact, while our construction tends to guarantee that tasks get closer as we move from D1 to D8 (recall how the tasks are defined in terms of subsets of RCV1 categories), we do not really know in advance how dissimilar the tasks in D1 are, and the performance of the $2p$-norm matrix multitask Perceptron algorithm reveals that they are indeed not so dissimilar.

As a further assessment, we evaluated the empirical performance of the p-norm matrix multitask algorithm on the ECML/PKDD 2006 Discovery Challenge spam data set (for details, see

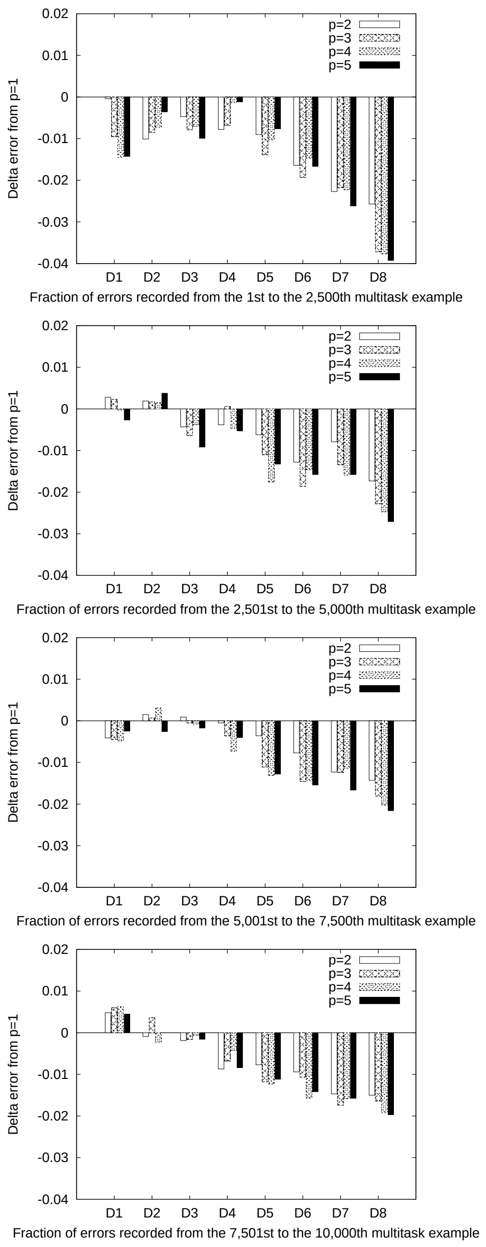

Figure 6: Online behavior of the $2p$-norm matrix multitask Perceptron algorithm. Again, we report the extent to which during a single pass over the entire data set the fraction training mistakes made by the $2p$-norm matrix multitask Perceptron algorithm (with $p = 2, 3, 4, 5$) exceeds the one achieved by the baseline $p = 1$, represented here as an horizontal line. On the x-axis are the multitask data sets whose tasks have different levels of correlations, from low (D1) to high (D8). The top plot reports the online training behavior on the first $2,500$ (multitask) examples, the second from top refers to the second $2,500$ examples, and so on.

ECML/PKDD, 2006). The data set includes 15 sub-data sets, each one containing 400 spam/ham emails for 15 different users. Email messages are encoded by a standard bag-of-words vector representation. Naturally enough, we associated each user with a different task. The experiments were run using a 10-fold cross-validation scheme. Each run consists of a single training epoch followed by a test phase. Since these data are not ordered chronologically as those in RCV1, we repeated the 10-fold cross-validation process 40 times, preceding each repetition with independent shuffles of the 15 data sets. Table 3 shows that the $2p$-norm matrix multitask algorithm exploits underlying latent relations and manages to achieve a significant decrease in both the training and test errors. In particular, the best performance is achieved when p is set to 4, which results in a test error of 9.7%, an improvement of more than 25% relative to the baseline $p = 1$, or nearly a 4.0% decrease in absolute terms. Whereas these are average values, the advantage of the $2p$-norm matrix multitask algorithm is still significant even when the standard deviation is factored in. Moreover, in most runs, the deviation from the average tended to be the on the same side for both $p = 1$ and $p > 1$. In other words, if on a given fold the p-norm matrix multitask Perceptron algorithm with $p > 1$ made a larger number of mistakes than its average, the same held true for the baseline. We stress that the theoretical improvement of a factor K (in this case $K = 15$), is within reach only if the tasks are linearly separable and overlapped. In practice we should and could not expect these conditions to be generally met to their full extent. In particular, while we cannot state how exactly the different spam classification tasks are spectrally related (since the task are not synthetically generated), it is apparent that such relations do actually hold to a certain extent. In fact, by considering the specifics of the learning problem, it is intuitively reasonable that the target spam/ham discriminant functions, though changing from user to user, still share a significant number of common traits.

8. Conclusions and Open Problems

We have studied the problem of sequential multitask learning using two different approaches to formalize the notion of task relatedness: via the Euclidean distance between task vectors, or via a unitarily invariant norm applied to the matrix of task vectors. These two approaches naturally correspond to two different online multitask protocols: one where a single task is selected at each time step, and one where the learner operates simultaneously on all tasks at each time step. We believe that both these protocols have their own merits, each one having its array of possible practical applications. Moreover, while the Schatten-p norm regularization assumption does not make sense in the adversarially chosen task protocol, it is not difficult to adapt the multitask kernel-based algorithm of Section 3 to the fully simultaneous protocol and derive a mistake bound analysis in the lines of the one given in Section 6.

In our worst-case sequential prediction scenario, the best we can hope for i.t.o. prediction performance is a factor K improvement over the baseline running K independent classification algorithms, this is what we essentially achieved in our analysis. We have complemented our theoretical findings with experiments on real-world data sets showing that our algorithms are efficient and can effectively take advantage of latent task relatedness.

We conclude by mentioning a few directions along which our results could be extended.

1. In Section 3.2 it might be interesting to devise methods for dynamically adapting the b parameter as new data are revealed.

2. In Section 3.3 we have shown a simple adaptation to the case when the graph of tasks (i.e., interaction matrix A) is known ahead of time. Is it possible to achieve meaningful bounds when the graph is hidden, and can only be progressively inferred through the choices of i_t?

3. Our multitask regularization techniques rely on the fact that different tasks need to be embedded, either naturally or through some sensible preprocessing, in the same d-dimensional space. It would be interesting to devise a multitask algorithm that does not impose such a constraint.

4. Finally, we believe it would also be interesting to prove *lower* bounds on the number of mistakes as a function of task relatedness.

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.

Appendix A.

The following trace inequality, which can be seen as a kind of Holder's inequality applied to non-square matrices, is our main technical tool in the proof of Theorem 9.

Lemma 13 *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 $l, g \geq 0$, we have*

$$\text{TR}(X^\top A^l X B^g) \leq \big(\text{TR}(X^\top X)^p\big)^{1/p} \big(\text{TR}\, A^{(l+g)q}\big)^{1/q}$$

where $1/p + 1/q = 1$, $p \geq 1$.

Proof We first consider the case $l \leq g$. By the Cauchy-Schwartz and Holder's inequalities applied to traces (Magnus and Neudecker, 1999, Chapter 11) we have

$$\begin{aligned}\text{TR}(X^\top A^l X B^g) &= \text{TR}\big(B^{(g-l)/2} X^\top A^l X B^{(g+l)/2}\big) \\ &\leq \text{TR}\big(X^\top A^{2l} X B^{g-l}\big)^{1/2} \text{TR}\big(X^\top X B^{g+l}\big)^{1/2} \\ &\leq \text{TR}\big(X^\top A^{2l} X B^{g-l}\big)^{1/2} T_p\big(X^\top X\big)^{1/2} T_q\big(B^{g+l}\big)^{1/2}\end{aligned} \tag{20}$$

where we used the shorthand $T_r(Z) = (\text{TR}Z^r)^{1/r}$. 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 (20). Recalling that $T_q(A) = T_q(B)$ and $T_p(X^\top X) = T_p(XX^\top)$, after n steps we obtain

$$\text{TR}\big(X^\top A^l X B^g\big) \leq \big(\text{TR}(X^\top A^{l'} X B^{g'})\big)^{1/2^n} T_p\big(X^\top X\big)^{\sum_{i=1}^n (1/2)^i} T_q\big(B^{g+l}\big)^{\sum_{i=1}^n (1/2)^i}$$

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. ■

We are now ready to prove Theorem 9.

Proof [Theorem 9] We set

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

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

$$D(V_s \| V_{s-1}) = \frac{1}{2} (\mathrm{VEC} M_t)^\top H_G(\xi) \mathrm{VEC}(M_t) \tag{21}$$

where 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. We start by computing the first derivative,

$$\nabla G(V) = \frac{1}{2} \nabla \mathrm{TR}\big((V^\top V)^p\big)^{1/p} = \frac{1}{2p} \mathrm{TR}\big((V^\top V)^p\big)^{1/p - 1} \nabla \mathrm{TR}\big((V^\top V)^p\big) \ . \tag{22}$$

Then, by applying the chain rule to $\nabla\big(\mathrm{TR}(V^\top V)^p\big)$ and using (15) and (16) we obtain

$$\begin{aligned}
\nabla \mathrm{TR}\big((V^\top V)^p\big) &= p \mathrm{VEC}\big((V^\top V)^{p-1}\big)^\top (I_{K^2} + T_{K^2})(I_K \otimes V^\top) \\
&= p \mathrm{VEC}\big((V^\top V)^{p-1}\big)^\top (I_K \otimes V^\top) \\
&\quad + p \Big(T_{K^2} \mathrm{VEC}\big((V^\top V)^{p-1}\big)\Big)^\top (I_K \otimes V^\top) \\
&\quad \text{(since } V^\top V \text{ and } T_{K^2} \text{ are symmetric)} \\
&= 2p \big((I_K \otimes V) \mathrm{VEC}(V^\top V)^{p-1}\big)^\top \\
&= 2p \mathrm{VEC}\big(V (V^\top V)^{p-1}\big)^\top
\end{aligned} \tag{23}$$

the last equation following from (12). We now substitute (23) back into (22) and obtain

$$\nabla G(V) = c(V) \mathrm{VEC}(D)^\top$$

where we set for brevity $D = V B^{p-1}$ and $c(V) = \mathrm{TR}(B^p)^{1/p-1}$ with $B = V^\top V$. Taking the second derivative $H_G = \nabla^2 G$ gives

$$H_G(V) = \mathrm{VEC}\,(D)\, \nabla c(V) + c(V)\, \nabla D \,.$$

Recalling the definition of $c(V)$ and using (15) it is easy to see that $\mathrm{VEC}\,(D)\, \nabla c(V)$ is the $Kd \times Kd$ matrix

$$(1-p) \mathrm{TR}\big(B^p\big)^{1/p-2} \mathrm{VEC}(D)\, \mathrm{VEC}(D)^\top. \tag{24}$$

Since $p \ge 1$ this matrix is negative semidefinite, and we can disregard it when bounding from the above the quadratic form (21). Thus we continue by considering only the second term $c(V)\, \nabla\,(D)$ of the Hessian matrix. We have

$$\nabla D = \big(B^{p-1} \otimes I_d\big) + (I_K \otimes V) \nabla B^{p-1}$$

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

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

Putting together

$$\begin{aligned}(21) \quad \leq \quad & \frac{c(V)}{2}\,\mathrm{VEC}(M_t)^{\top}(B^{p-1} \otimes I_d)\,\mathrm{VEC}(M_t) \\ & + \frac{c(V)}{2}\,\mathrm{VEC}(M_t)^{\top}(I_K \otimes V)\Sigma \times \\ & \times (I_{K^2} + T_{K^2})\left(I_K \otimes V^{\top}\right)\mathrm{VEC}(M_t) \end{aligned} \tag{25}$$

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 (25). Using again (12) combined with (13) we can write

$$\frac{c(V)}{2}\,\mathrm{VEC}(M_t)^{\top}(B^{p-1} \otimes I_d)\,\mathrm{VEC}(M_t) = \frac{c(V)}{2}\,\mathrm{TR}(M_t^{\top} X_t B^{p-1}) \leq \frac{1}{2}\,\mathrm{TR}\big((M_t^{\top} M_t)^p\big)^{1/p}$$

independent of V. The majorization follows from Holder's inequality applied to the positive semidefinite matrices $M_t^{\top} M_t$ and B^{p-1} (Magnus and Neudecker, 1999, Chapter 11). Moreover, it is easy to see that the symmetric matrices Σ and T_{K^2} commute, thereby sharing the same eigenspace, in fact

$$T_{K^2}\Sigma = \sum_{\ell=0}^{p-2} T_{K^2}\big(B^{\ell} \otimes B^{p-2-\ell}\big) = \sum_{\ell=0}^{p-2}\big(B^{p-2-\ell} \otimes B^{\ell}\big)T_{K^2} = \Sigma\, T_{K^2}\ .$$

Hence, $\Sigma\,(I_{K^2} + T_{K^2}) \preccurlyeq 2\Sigma$, and we can bound from above the second term in (25) by

$$c(V)\mathrm{VEC}(M_t)^{\top}\sum_{\ell=0}^{p-2} B^{\ell} \otimes A^{p-1-\ell}\mathrm{VEC}(M_t) \tag{26}$$

where we set $A = VV^{\top}$. Again, (12) and (13) allow us to rewrite this quadratic form as the sum of traces

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

Since A and B have the same nonzero eigenvalues, we can apply Lemma 13 to each term and put together as in (25). After simplifying we get

$$(21) \leq \frac{1}{2}\,(2p-1)\mathrm{TR}\big((M_t^{\top} M_t)^p\big)^{1/p} = \frac{1}{2}\,(2p-1)||M_t||_{s_{2p}}^2.$$

Substituting back into (18), and recalling that $\|U\|_* = \|U\|_{s_{2q}}$ in the case of Schatten norms, yields

$$\begin{aligned}\mu \quad & \leq \quad \sum_{t \in \mathcal{M}} \ell_t^{\mathbb{1}}(U) + \|U\|_{s_{2q}}\sqrt{(2p-1)\sum_{t \in \mathcal{M}}\|M_t\|_{s_{2p}}^2} \\ & \leq \quad \sum_{t \in \mathcal{M}} \ell_t^{\mathbb{1}}(U) + \|U\|_{s_{2q}}\sqrt{(2p-1)\max_{t \in \mathcal{M}}\frac{\|M_t\|_{s_{2p}}^2}{\|\mathbb{1}_t\|_1}\,\mu}\ .\end{aligned}$$

Solving the inequality for μ, and overapproximating via $\sqrt{a+b} \le \sqrt{a} + \sqrt{b}$ we now obtain

$$\mu \le \sum_{t \in \mathcal{M}} \ell_t^{\mathbb{1}}(U) + (2p-1)\left(\mathbf{M}_{s_{2p}} \|U\|_{s_{2q}}\right)^2 + \mathbf{M}_{s_{2p}} \|U\|_{s_{2q}} \sqrt{(2p-1) \sum_{t \in \mathcal{M}} \ell_t^{\mathbb{1}}(U)}$$

thereby concluding the proof. ■

The core of the above analysis is an upper bound on the second-order term of the Taylor expansion of the Schatten p-norm around V_{s-1}. Our proof of this bound is based on a direct matrix analysis. A more general bound has independently been derived in Juditsky and Nemirovski (2009, Proposition 3.1) and used in Kakade et al. (2009), where the unitarily invariant norms of our analysis are replaced by general convex functions.

References

J. Abernethy, P.L. Bartlett, and A. Rakhlin. Multitask learning with expert advice. In *Proceedings of the 20th Annual Conference on Learning Theory*, pages 484–498. Springer, 2007.

A. Agarwal, A. Rakhlin, and P. Bartlett. Matrix regularization techniques for online multitask learning. Technical Report UCB/EECS-2008-138, EECS Department, University of California, Berkeley, 2008.

R.K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems 19*, pages 41–48. MIT Press, 2007.

A. Argyriou, C.A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In *Advances in Neural Information Processing Systems 20*, pages 25–32. Curran Associates, 2008.

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operation Research Letters*, 31:167–175, 2003.

U. Brefeld, T. Gaertner, T. Scheffer, and S. Wrobel. Efficient co-regularised least squares regression. In *Proceedings of the 23rd International Conference on Machine Learning*. Omnipress, 2006.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order Perceptron algorithm. *SIAM Journal on Computing*, 43(3):640–668, 2005.

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

ECML/PKDD Discovery Challenge, 2006. URL: www.ecmlpkdd2006.org/challenge.html.

T. Evgeniou, C. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:614–637, 2005.

Y. Freund and R.E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

C. Gentile. The robustness of the *p*-norm algorithms. *Machine Learning*, 53(3): 265–299, 2003.

A.J. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. *Machine Learning*, 43(3):173–210, 2001.

M. Herbster and G. Lever. Predicting the labelling of a graph via minimum *p*-seminorm interpolation. In *Proceedings of the 22nd Annual Conference on Learning Theory*. Omnipress, 2009.

M. Herbster, M. Pontil, and L. Wainer. Online learning over graphs. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 305–312. Omnipress, 2005.

L. Hogben. *Handbook of Linear Algebra*. CRC Press, 2006.

R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

R.A. Horn and C.R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

L. Jacob, F. Bach, and J.P. Vert. Clustered multi-task learning. In *Advances in Neural Information Processing Systems 21*, pages 745–752. Curran Associates, 2009.

A. Juditsky and A. Nemirovski. Large deviations of vector-valued martingales in 2-smooth normed spaces. Manuscript, 2009.

S. Kakade and D. Foster. Multi-view regression via canonical correlation analysis. In *Proceedings of the 20th Annual Conference on Learning Theory*, pages 82–96. Springer, 2007.

S. Kakade, S. Shalev-Shwartz, and A. Tewari. On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization. Manuscript, 2009.

J. Kivinen and M. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45:301–329, 2001.

A.S. Lewis. The convex analysis of unitarily invariant matrix functions. *Journal of Convex Analsys*, 2(1):173–183, 1995.

N. Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms*. PhD thesis, University of California at Santa Cruz, 1989.

G. Lugosi, O. Papaspiliopoulos, and G. Stoltz. Online multi-task learning with hard constraints. In *Proceedings of the 22nd Annual Conference on Learning Theory*. Omnipress, 2009.

J.R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley, 1999.

A. Maurer. Bounds for linear multi-task learning. *Journal of Machine Learning Research*, 7:117–139, 2006.

A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Nauka Publishers, 1978.

NIST, 2004. URL: trec.nist.gov/data/reuters/reuters.html.

V. Sindhwani D.R. Rosenberg. An RKHS for multi-view learning and manifold co-regularization. In *Proceedings of the 26th International Conference on Machine Learning*, pages 976–983. Omnipress, 2008.

V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularized approach to semi-supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning Workshop on Learning with Multiple Views*, 2005.

K. Tsuda, G. Raetsch, and M.K. Warmuth. Matrix exponentiated gradient updates for on-line learning and Bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.

M.K. Warmuth. Winnowing subspaces. In *Proceedings of the 24th International Conference on Machine Learning*, pages 999–1006. Omnipress, 2007.

M.K. Warmuth. Kernelization of matrix updates. Manuscript, 2009.

M.K. Warmuth and D. Kuzmin. Online variance minimization. In *Proceedings of the 19th Annual Conference on Learning Theory*, pages 514–528. Springer, 2006.