Multi-task learning on the edge: cost-efficiency and theoretical optimality

Malik Tiomoko, Romain Couillet, Sami Fakhry

Abstract

** A retravailler lorsque le papier sera fini **

This article proposes a distributed multi-task learning (MTL) algorithm based on supervised principal component analysis (SPCA). Supporting experiments on synthetic and real data benchmark show how energy costs can be reduced while ensuring data privacy.

I. Introduction

The necessary revolution towards low carbon-footprint AI impacts our "consumption" of data storage, exchange as well as computational power. In this vision, transfer and multi-task learning are efficient solutions to reuse labelled datasets, generally distributed over the web, but their optimal designs (recently studied using large dimensional statistics [2]) in general demand to gather all data together. In parallel, edge computing aims to systematically perform heavy computations locally with minimal exchanges, yet in general at the expense of joint optimality.

In the present article, we introduce a cost-efficient "multi-task learning on the edge" paradigm which draws simultaneously the strength of minimalistic data exchanges from edge computing and the capability of multi-task learning to associate multiple (possibly statistically distinct) datasets together. Most importantly, by precisely exchanging the *sufficient statistics* (rather than the data themselves) and by optimizing the task-dependent data labels (rather than setting them all to ± 1 as conventionally, but sub-optimally done), the proposed scheme is shown to be information-theoretically close to optimal in our model setting.

Specifically, the work presented here provides a distributed and scalable extension of the supervised PCA-based multi-task learning algorithm (MTL-SPCA) of [2]. For large data sizes and numbers, the resulting algorithm is provably equivalent in performance to the original MTL-SPCA – itself shown in [2] to be competitive with alternative state-of-the-art algorithms (such as LS-SVM MTL [3], CDLS [4]) –, while simultaneously allowing for drastic cost reductions in data sharing. In effect, our distributed multitask algorithm only requires the *empirical average of the local data* attached to each task to be exchanged. In addition, the algorithm retrieves all advantages from MTL-SPCA, such as the theoretical absence of *negative transfers* when dealing with large datasets, which generally impede multitask (or transfer) learning algorithms.

As a result, the presently proposed algorithm benefits from the potential abundance of available distributed data sources, while maintaining a constant number of data exchanges on the edge, thereby guaranteeing a low computational cost footprint (as well as, in passing, enforcing data privacy).

The remainder of the article is structured as follows: Section II recalls the basics of MTL-SPCA in its centralized version, which is then extended in Section III to a fully distributed approach. Section IV provides supporting experiments, emphasizing on the algorithm performance in terms of computational cost, overall data transmission, and comparison to optimality.

II. Centralized Algorithm

A. Overview

The MTL-SPCA algorithm [2] is a fast supervised multi-task classification algorithm relying on

a preliminary PCA-like step (called SPCA as it exploits the "Supervised" knowledge of training data classes) which projects the data onto a *subspace of weighted data classes*. As for PCA, it is a spectral method which demands the evaluation of the dominant eigenvectors of the (label-weighted) data. But the main feature of MTL-SPCA lies in its providing finely tuned labels (so not necessarily ± 1 in a binary classification setting) for each data class in each task, which encompass the task relatedness information. In particular, being mathematically tractable, the labels are theoretically optimized, discarding altogether the very profound and classical problem of negative transfer in transfer and multi-task learning; a salient feature is that these *labels are not intrinsic to the data but differ for each target task*, and are optimized so to minimize the classification error for each task individually.

Of utmost importance for the present article, the inner functioning of MTL-SPCA relies on "condensed" data information arising from each task (the class-wise statistical means of the data), thereby easily allowing for a distributed extension.

B. Setting

We consider a scenario where k independent agents (call them clients) aim at solving a classification task on their own data based on training inputs $X_1, \ldots, X_k$, where $X_t \in \mathbb{R}^{p \times n_t}$ is the dataset of client (or task) t composed of n_t data vectors of size p.[1]

We then define $X = [X_1, \ldots, X_k] \in \mathbb{R}^{p \times n}$ (with $n = \sum_{t=1}^{k} n_t$) and, for task t we let $X_t = [X_{t1}, \ldots, X_{tm}]$ for $X_{tj} \in \mathbb{R}^{p \times n_{tj}}$ the subset of data of class j ($1 \leq j \leq m$) for client t (in particular $n_t = \sum_{j=1}^{m} n_{tj}$). Exploiting a one-versus-all approach, the m-class problem can be decomposed as a series of m 2-class problems so that, for notational ease, from now on, we restrict ourselves to a 2-class setting, i.e., we let $m = 2$.

We further denote, for class j in task t, $X_{tj} = [x_{t1}^{(j)}, \ldots, x_{tn_{tj}}^{(j)}] \in \mathbb{R}^{p \times n_{tj}}$. To each $x_{t\ell}^{(j)}$ is classically associated a label $y_{t\ell}^{(j)} \in \{\pm 1\}$. The authors in [2] prove this choice to be largely suboptimal and the main source of the negative transfer problem. Instead, [2] shows that, for large dimensions p, n_{ij}, if all data in X are independent Gaussian vectors with unit covariance, $y_{t\ell}^{(j)}$ should be taken constant for all ℓ and equal to $y_{t\ell}^{(j)} = [\tilde{y}]_{tj}$ where $\tilde{y} \in \mathbb{R}^{2k}$ is the vector:

$$\tilde{y}_t = \mathcal{D}_c^{-\frac{1}{2}} (\mathcal{M} + I_{2k}^{-1}) \mathcal{M} \mathcal{D}_c^{-\frac{1}{2}} (e_{t1} - e_{t2}) \in \mathbb{R}^{2k}$$

indexed as $11, 12, 21, \ldots, k2$; here $e_{tj} \in \mathbb{R}^{2k}$ the canonical vector equal to 1 in position (t, j),

$$\mathcal{M} = (1/c_0) \mathcal{D}_c^{\frac{1}{2}} M^{\mathsf{T}} M \mathcal{D}_c^{\frac{1}{2}} \in \mathbb{R}^{2k \times 2k}$$

is the (fundamental) inter-task similarity matrix,

$$c = [n_{11}/n, \ldots, n_{k2}/n]^T \in \mathbb{R}^{2k}$$

is the vector of data size ratios with $\mathcal{D}_c = \mathrm{diag}(c)$ and $c_0 = p/n$, and M is the matrix containing the statistical means of the class-wise task-wise data subsets

$$M = [\mu_{11}, \mu_{12}, \ldots, \mu_{k1}, \mu_{k2}], \quad \mu_{tj} = \mathbb{E}[x_{t1}^{(j)}].$$

Note importantly that, due to the canonical vectors e_{t1}, e_{t2}, the task-t optimal label vector $\tilde{y}_t$ *depends* on the target task.

Remark 1 (Estimating $\tilde{y}_t$)**.** *For large p, n_{ij}, the matrix $M^{\mathsf{T}} M$ can be effectively estimated empirically from*

$$[M^{\mathsf{T}} M]_{qq} = \frac{4}{n_{ij}^2} \mathbf{1}_{n_{ij}}^{\mathsf{T}} X_{ij;1}^{\mathsf{T}} X_{ij;2} \mathbf{1}_{n_{ij}} + o(1)$$

$$[M^{\mathsf{T}} M]_{qq'} = \frac{1}{n_{ij} n_{i'j'}} \mathbf{1}_{n_{ij}}^{\mathsf{T}} X_{ij}^{\mathsf{T}} X_{i'j'} \mathbf{1}_{n_{i'j'}} + o(1)$$

with $q = 2(i-1)+j$ and $q' = 2(i'-1)+j'$ different and $X_{ij} = [X_{ij;1}, X_{ij;2}]$ an even-sized division of X_{ij}. With this result, $\tilde{y}_t$ can be consistently estimated for all large p, n_{ij}.

Remark 1 is particularly important to what follows as it shows that the optimal hyperparameters $\tilde{y}_t$ is accessible from the knowledge of the empirical mean vectors $\frac{1}{n_{ij}} X_{ij} \mathbf{1}_{n_{ij}}$, and in particular does not require to access the individual column vectors of X_{ij}.

[1] For simplicity, we assume all data of size p irrespective of the task.

To classify a sample $\mathbf{x}$ arising at client t, the MTL-SPCA algorithm then consists in projecting $\mathbf{x}$ onto the vector

$$V = \frac{Xy}{\|Xy\|}$$

for $y = [\mathbf{1}_{n_{11}}\tilde{y}_{11}, \ldots, \mathbf{1}_{n_{k2}}\tilde{y}_{k2}]$. Precisely, the optimal (error-rate minimizing in a Gaussian setting) decision is given by the test

$$g(\mathbf{x}) \equiv V^{\mathsf{T}}\mathbf{x} \gtrless \zeta_t \equiv \frac{1}{2}(m_{t1} + m_{t2})$$

where

$$m_{tj} = ???$$

is the statistical mean of $g(\mathbf{x})$ for $\mathbf{x}$ in class j of task t. It is easily seen that m_{tj} is easily estimated empirically.

Here again, $Xy = \sum_{t,j} y_{tj} X_{tj} \mathbf{1}_{n_{tj}}$ is merely a linear combination of the empirical means $\frac{1}{n_{tj}} X_{tj}\mathbf{1}_{n_{tj}}$.[2]

III. Distributed Algorithm

A. Setting

The setting is essentially the same as described above, however the k data sets of each tasks are not locally gathered but spread across k clients representing the tasks. Concretely, the k clients send *sufficient statistics* to another client temporarily in charge of gathering the statistics and computing the prediction model. The *sufficient statistics* sent to this central client consist of empirical means computed from the local data of each client/task. When the model is computed, any client can ask for the model parameters to then infer on new data.

For each class $j \in \{1, \ldots, m\}$ the t^{th} client computes empirical mean by class $M_j = \frac{1}{n_{tj}} X_{tj}\mathbf{1}_{n_{tj}} \in \mathbb{R}^{p\times 1}$ and send $M = [M_1 \; M_2] \in \mathbb{R}^{p\times 2}$. The central client then gather k matrices of empirical means from all the clients in one big matrix $\hat{M} \in \mathbb{R}^{p\times 2k}$ and compute the inter-task similarity matrix $\mathcal{M} = (1/c_0)\mathcal{D}_c^{\frac{1}{2}} \hat{M}^T \hat{M} \mathcal{D}_c^{\frac{1}{2}}$. As in the non-distributed case, $\mathcal{M}$ plays a major role in the quality of the model. It allows to compute optimal labels for the target client, which allow to compute the largest eigenvector of $\frac{X\tilde{y}\tilde{y}^T X^T}{np}$, $V = \frac{Xy}{\|Xy\|}$. In this formula, X represents all the datasets gathered. Hence, it is not usable here as it would break the data privacy and render useless the distribution of the algorithm. Fortunately, V can be rewritten in a formula best suited for a distributed algorithm, involving the empirical means and optimal labels.

$$Xy = \sum_{t=1}^{k}\sum_{j=1}^{2} X_{tj}\mathbf{1}_{n_{tj}}\tilde{y}_{2t+j} = \sum_{t=1}^{k}\sum_{j=1}^{2} n_{tj} M_{tj}\tilde{y}_{2t+j} \in \mathbb{R}^{p\times 1}$$

which leads to $V = \frac{\sum_{t=1}^{k}\sum_{j=1}^{2} n_{tj} M_{tj}\tilde{y}_{2t+j}}{\|\sum_{t=1}^{k}\sum_{j=1}^{2} n_{tj} M_{tj}\tilde{y}_{2t+j}\|}$.

From there, the vector V can be sent back to the target client in need.

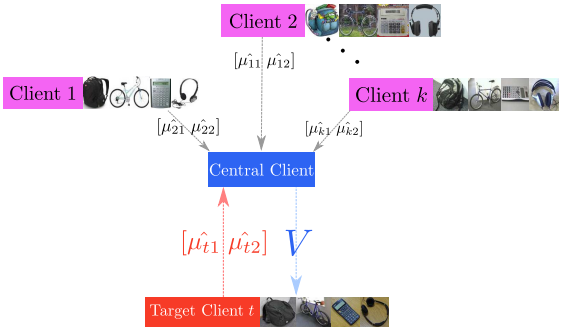

Fig. 1. Schematics of the distributed algorithm for the Amazon (Client 1), Caltech (Client 2), DSLR (Client k), Webcam (Target Client t) database. Client t draws the projection vector V from the empirical statistical means shared by all clients.

B. Algorithm

IV. Experiments

A. 2-class 2-task transfer learning

Figure 2 illustrates a 2-class 2-task classification setting trained and tested on synthetic Gaussian data $x_{tl}^{(j)} \sim \mathcal{N}((-1)^j \mu_t, I_p), \mu_2 = \beta\mu_1 + \sqrt{1-\beta^2}\mu_1^{\perp}$ for any $\mu_1^{\perp}$ orthogonal to μ_1. It depicts a simple transfer learning case with $n_{11} = n_{12} = 1000$

[2]The uppercase notation V, borrowed from [2], is reminiscent of the fact that, in a generic m-class setting, V generalizes to the SPCA projector onto the task-wise class-wise weighted data subspace. Here with $m = 2$, this projector reduces to the normalized vector Xy.

Algorithm 1: Distributed k-tasks algorithm

Data: $\forall t \in \{1;k\}, X_t$ is the training data of task t. Test data $\mathbf{x}$
Result: Estimated class $\hat{j} \in \{1;m\}$ of the test data $\mathbf{x}$ for target task t
initialization;
for $t \in \{1;k\}$ **do**
  $M_k = \left[\frac{1}{n_{tj}} X_{t1}\mathbf{1}_{n_{t1}} \ \frac{1}{n_{tj}} X_{t2}\mathbf{1}_{n_{t2}}\right]$;
end
Send the M_k matrices;
Gather the M_k matrices in $\hat{M}$;
Compute the inter-task similarity matrix $\mathcal{M}$;
Evaluate the optimal labels $\tilde{y}_t$ and V according to target task t;
Send back V to the target client t;
Compute the classification score $V^T\mathbf{x}$ and classify $\mathbf{x}$ in comparison with the threshold;

2-class Gaussian mixture transfer error rate for balanced/unbalanced classes

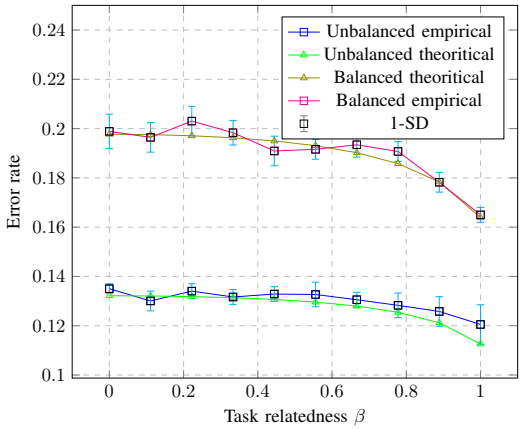

Fig. 2. Distributed. $n_{11} = n_{12} = 1000, n_{21} = n_{22} = 130$. Averaged over 20000 sample tests

and $n_{21} = n_{22} = 130$ for the balanced case and $n_{11} = n_{12} = 1000$ and $n_{21} = 50, n_{22} = 210$ for the unbalanced one.

1) Bayesian Optimality: In Figure 2 the proposed paradigm is shown to be information-theoretically close to optimal. It also depicts the good performance in theory of the model even with unbalanced classes (e.g. with $n_{21} = 50, n_{22} = 210$ for the target task). For an unbalanced case, the threshold is changed to $\frac{1}{2}(m_{t0} + m_{t1}) - \frac{1}{(m_{t0}-m_{t1})}\log(\frac{\rho_1}{\rho_2})$ - where $\rho_j = \frac{n_{tj}}{n}$ is the proportion of test data in class j and m_{tj} is the estimated mean of the data (see [2], Corrolary 1) - to better fit the imbalance. Knowing *a priori* the proportion of the test data, one could easily bias the threshold in favor of the most represented class (see the formula above). In practice, the test data proportion is unknown *a priori* as it is to be predicted, rendering this lower bound unusable.

B. Adding tasks

Let us now consider a 2-class k-tasks classification setting trained and tested on synthetic Gaussian data first, then on real data. For the synthetic data $x_{tl}^{(j)} \sim \mathcal{N}((-1)^j\mu_t, I_p), \mu_t = \beta_t\mu + \sqrt{1-\beta_t^2}\mu^\perp$ with β_t drawn uniformly at random in $[0;1]$ and with $\mu = e_1^{[p]}, \mu^\perp = e_p^{[p]}$. 2^i tasks are added simultaneously with n_{tj} predefined. It can be seen that the error rate reaches a saturation state when a certain amount of task is added. Rajouter des courbes avec beta constant + structure de la matrice M^TM

k-clients 2-class Gaussian mixture transfer error rate

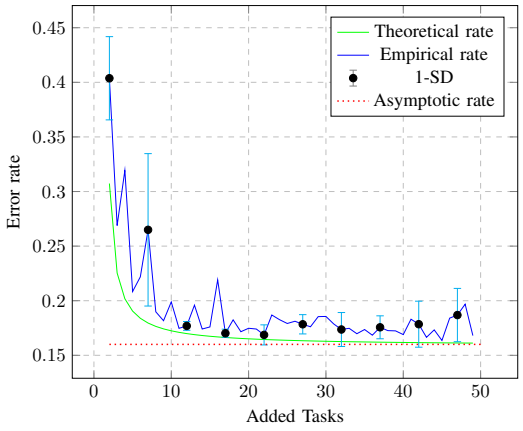

Fig. 3. Distributed. $n_{11} = n_{12} = 50, n_{21} = n_{22} = 6$. Averaged over 10000 sample tests

C. Data transmission

The biggest advantage of the "multi-task on the edge" paradigm proposed here is the drastic reduction of data sent from the clients to be gathered. Essentially, the k clients only send k vectors (1 per client) of size p containing the empirical means, whereas sending datasets requires $\sum_{c=1}^{k}(n_{c1}+n_{c2})=n$ vectors of size p and breaks the data privacy kept here.

Data Type	Art	Real World	Product	Clipart
Raw	$100 \times n_1$	$100 \times n_2$	$30 \times n_3$	$30 \times n_4$
Resnet Features	$65 \times n_1$	$65 \times n_2$	$65 \times n_3$	$65 \times n_4$
Distributed Resnet Features	65×1	65×1	65×1	65×1

TABLE I
DATA SENT COMPARISON BETWEEN RAW DATA, RESNET-50 FEATURES AND THE DISTRIBUTED ALGORITHM ON THE OFFICE-HOME DATASET [5] IN KBITS

In Table I each column represents a task. The numerical values for the raw data are the average size of the images for each task. A resnet-50 feature is a 2048-dimensional vector composed of float values of 4 bytes each, leading to a vector of roughly 65 kbits. In the distributed version one vector of size p is sent for each task. In the case of Resnet-50 features, it sums up to $k \times 65$ kbits, instead of $n \times 65$ kbits for the Resnet-50 features in the non-distributed one. As shown above, the error rate saturates rapidly when k grows. This way, k will always be negligible in front of n, and the distributed model will thus always demand less data to be stored and exchanged.

D. Marginal task update

In the proposed paradigm, adding a new task to an existing model can be done at a low computational cost.

$M = \begin{bmatrix}\hat{\mu}_{11} & \hat{\mu}_{12} & \dots & \hat{\mu}_{k1} & \hat{\mu}_{k2} & \hat{\mu}_{t'1} & \hat{\mu}_{t'2}\end{bmatrix}$ is the matrix of empirical means of the k tasks augmented by a new task t'. The calculation of the *symmetrical matrix* M^TM is then only based on the calculation of the $(2k+1)+(2k+2)=4k+3$ coefficients instead of $2(k+1)\times 2(k+1)$ if we were to compute the whole matrix M^TM. The new coefficients that needs to be computed are denoted in red.

$$M^TM = \begin{bmatrix} \hat{\mu}_{11}^T\hat{\mu}_{11} & \dots & \hat{\mu}_{11}^T\hat{\mu}_{k2} & \hat{\mu}_{11}^T\hat{\mu}_{t'1} & \hat{\mu}_{11}^T\hat{\mu}_{t'2} \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ \hat{\mu}_{t'1}^T\hat{\mu}_{11} & \dots & \dots & \hat{\mu}_{t'1}^T\hat{\mu}_{t'1} & \hat{\mu}_{t'1}^T\hat{\mu}_{t'2} \\ \hat{\mu}_{t'2}^T\hat{\mu}_{11} & \dots & \dots & \hat{\mu}_{t'2}^T\hat{\mu}_{t'1} & \hat{\mu}_{t'2}^T\hat{\mu}_{t'2} \end{bmatrix}$$

Thereby, in a distributed paradigm, the target task could benefit from other tasks at low computational cost.

E. Real data simulations

The real data consists of the Amazon Review dataset [1]. Here, new tasks (book reviews, dvd reviews and electronics reviews) are added to help classify reviews on kitchen items.

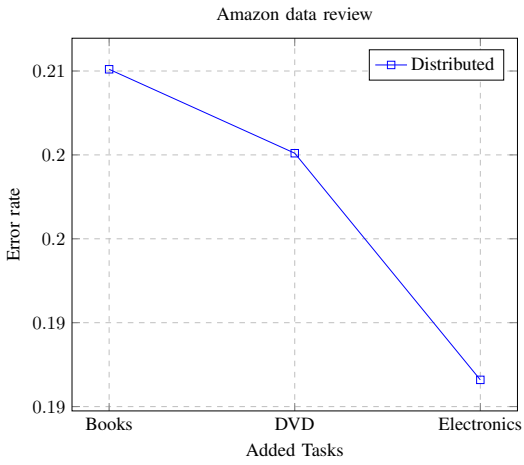

Fig. 4. Distributed. $n_{11}=n_{12}=200, n_{21}=50, n_{22}=90, p=400$.

V. CONCLUSION AND DISCUSSION

REFERENCES

[1] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[2] Malik Tiomoko, Romain Couillet, and Frédéric Pascal. Pca-based multi task learning: a random matrix approach. *Conference on Neural Information Processing Systems*, 2021.

[3] Malik Tiomoko, Romain Couillet, and Hafiz Tiomoko. Large dimensional analysis and improvement of multi task learning, 2020.

[4] Yao-Hung Hubert Tsai, Yi-Ren Yeh, and Yu-Chiang Frank Wang. Learning cross-domain landmarks for heterogeneous domain adaptation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5081–5090, 2016.

[5] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5018–5027, 2017.