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

Malik Tiomoko, Romain Couillet, Sami Fakhry

Abstract

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 [1]) 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 [1]. For large data size and numbers, the resulting algorithm is provably equivalent in performance to the original MTL-SPCA, itself proved in [1] to be competitive with alternative state-of-the-art algorithms (such as LS-SVM MTL [?], CDLS [?], ** d'autres ?? **), 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: ... TBD ...

II. Non-distributed Algorithm

A. Overview

The non-distributed MTL-SPCA algorithm relies on the calculation of the largest eigenvectors of the label-weighted data, instead of calculating the largest eigenvectors of the raw data as for a classical PCA. To improve the performance of the algorithm and avoid negative transfer, the labels are optimized according to the target task of the transfer. This optimization solely relies on the empirical average of the local data attached to each task. Thereby, the largest eigenvector can be reevaluated with optimal labels to better match the information contained in the target data. As it will be seen further, the algorithm can easily be distributed to benefit from various sources as it only depends on the empirical means of the data tied to each task.

B. *Setting*

More formally, the k-tasks data is condensed in a matrix of n vectors of size p, $X = [X_1, \ldots, X_k] \in \mathbb{R}^{p \times n}$, where each X_t is associated to task $t \in \{1; k\}$. For all task t, the data is separated in m classes $X_t = [X_{t1}, \ldots, X_{tm}] \in \mathbb{R}^{p \times n_t}$ with n_{tj} the number of data for class j, and $\sum_{t=1}^{k} n_t = n$. From an m-class setting can always be derived a 2-class setting utilizing a one-vs-all approach as in [1]. From now, to alleviate notations a 2-class setting will be prefer.

For each task t and each class j, we denote the data $X_{tj} = x^{(j)}_{tn_{tj}} \in \mathbb{R}^p$. To each $x^{(j)}_{tn_{tj}}$ is classically associated a label $y_j = \pm 1$. Here this notation is dismissed due to its proved sub-optimality, in favor of *optimal labels* given by the explicit formula for task t: $\tilde{y}_t = \mathcal{D}_c^{-\frac{1}{2}}(\mathcal{M} + I_{2k}^{-1})\mathcal{M}\mathcal{D}_c^{-\frac{1}{2}}(e_{t1} - e_{t2}) \in \mathbb{R}^{2k}$. $\tilde{y}_t$ is $2k$-dimensional vector representing the optimal labels attached to each class of each task. We denote $\mathcal{M} = (1/c_0)\mathcal{D}_c^{\frac{1}{2}} M^T M \mathcal{D}_c^{\frac{1}{2}} \in \mathbb{R}^{2k \times 2k}$ the inter-task similarity matrix weighted by the proportion of data in each class with $c = [n_{11}/n, \ldots, n_{2k}/n]^T \in \mathbb{R}^{2k}$, $\mathcal{D}_c = diag(c)$ and $c_0 = p/n > 0$. M is the matrix containing the empirical means of the data. $M^T M$ is therefore the matrix composed of the product between the empirical means which stand for the correlation between tasks. $M^T M$ is defined as below :

$$[M^T M]_{(im+j)(i'm+j')} = \frac{1}{n_{ij} n_{kl}} \mathbf{1}_{n_{ij}}^T X_{ij}^T X_{kl} \mathbf{1}_{n_{kl}}$$

$$[M^T M]_{(im+j)(im+j)} = \frac{4}{n_{ij}^2} \mathbf{1}_{n_{ij_1}}^T X_{ij_1}^T X_{ij_2} \mathbf{1}_{n_{ij_2}}$$

$\forall i, i' \in \{0; k-1\}$, $j, j' \in \{1; m\}$ and X_{ij_1}, X_{ij_2} the first and second half of the i^{th} task, j^{th} class data, respectively. The optimal labels are then used to compute the largest eigenvector of $\frac{X\tilde{y}\tilde{y}^T X^T}{np}$ denoted V. In a 2-class k-tasks classification setting $V = \frac{Xy}{\|Xy\|}$. From there, V is used to infer the class in which new data $\mathbf{x}$ belongs, by comparing $V^T\mathbf{x}$ with the threshold $\frac{1}{2}(m_{t0} + m_{t1}) - \frac{1}{(m_{t0} - m_{t1})} \log(\frac{\rho_1}{\rho_2})$ Changer le threshold et rajouter les courbes avec les différents threshold et courbes optimales where $\rho_j = \frac{n_{tj}}{n}$ is the proportion of data in class j and m_{tj} is the estimated mean of the data (see [1], Corrolary 1).

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 a third-party server in charge of gathering the statistics and computing the prediction model. The *sufficient statistics* sent to the server consist of empirical means computed from the local data of each client/task. When the model is computed, a client can ask the server 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 server 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 rewrite in a formula best suited for a distributed algorithm, involving the empirical means and optimal labels.

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

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 server sends back the vector V to the target client in need.

B. Algorithm

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;
```

IV. Results

A. 2-class 2-task transfer learning

Figure 1 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 $n_{11} = n_{12} = 1000$ and $n_{21} = n_{22} = 50$. Here, the empirical rate decreases at a $O(\frac{1}{\sqrt{np}})$ rate as the inter-task similarity $\beta \in [0; 1]$ parameter increases.

B. Adding tasks

Let us now consider a 2-class k-task 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 $[0.8; 1]$ 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 $M^T M$

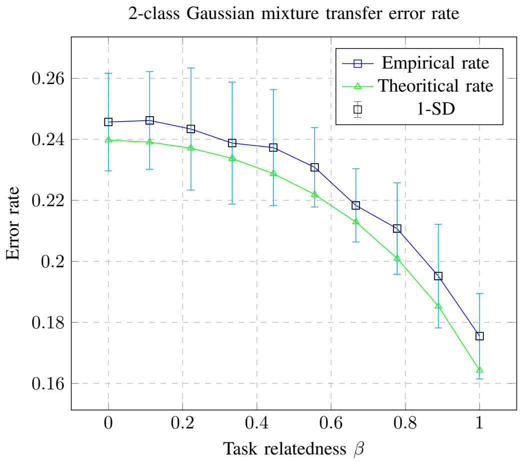

Fig. 1. Distributed. $n_{11} = n_{12} = 1000, n_{21} = n_{22} = 50$. Averaged over 2000 sample tests

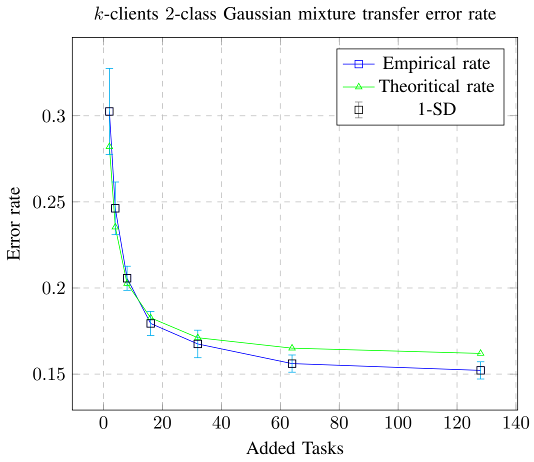

Fig. 2. Distributed. $n_{11} = n_{12} = 1000, n_{21} = n_{22} = 50$. Averaged over 2000 sample tests

1) 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 the server. 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	6	87837	787
Resnet Features	100	7	78	5415
Distributed Resnet Features	100	545	778	7507

TABLE I
DATA SENT COMPARISON BETWEEN RAW DATA, RESNET-50 FEATURES AND THE DISTRIBUTED ALGORITHM ON THE OFFICE-HOME DATASET [2]

2) Marginal task update: In the proposed paradigm, adding a new task to an existing model can be done at a low computational cost. Let say $M = \begin{bmatrix}\hat{\mu}_{11} & \hat{\mu}_{12} & \dots & \hat{\mu}_{k1} & \hat{\mu}_{k2} & \hat{\mu}_{t'1} & \hat{\mu}_{t'2}\end{bmatrix}$ 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.

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

C. Real data simulations

The real data consist of the Amazon Review dataset.

D. Bayesian optimality

In this section, the proposed scheme is shown to be information-theoretically close to optimal in our model setting.

V. CONCLUSION AND DISCUSSION

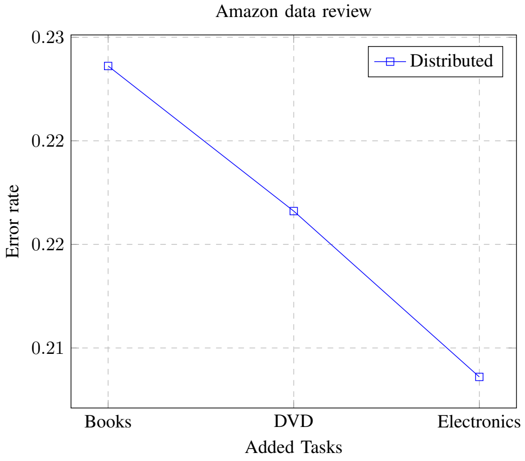

Fig. 3. Distributed. $n_{11}=n_{12}=1000, n_{21}=n_{22}=50$. Averaged over 2000 sample tests

REFERENCES

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

[2] 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.