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

Sami Fakhry[1], Romain Couillet[1,2,⋆], Malik Tiomoko[1]
[1]GIPSA-lab, [2]LIG-lab, Grenoble-Alps University, France.

Abstract

This article proposes a distributed multi-task learning (MTL) algorithm based on supervised principal component analysis (SPCA) [1], [2], which is: (i) theoretically optimal for Gaussian mixtures, (ii) computationally cheap and scalable. Supporting experiments on synthetic and real benchmark data demonstrate that significant energy gains can be obtained with no performance loss.

I. Introduction

The mandatory low carbon-footprint revolution in technologies impacts our "consumption" of data storage, exchange, and computational power. In this view, transfer and multi-task learning [4], [13], [10], [12] are efficient solutions to exploit remotely located datasets, but their optimal designs in general demand to gather all data together. In parallel, edge computing [6], [8], [7], [11] proposes to maintain computations locally with minimal exchanges but at the expense of performance.

We introduce here a cost-efficient "multi-task learning on the edge" paradigm which draws the strengths of minimalistic data exchanges from edge computing and of multi-task learning to exploit multiple (possibly statistically distinct) datasets together. By exchanging the *sufficient statistics* (rather than the data) and by optimizing the task-dependent data labels (rather than setting them all to ± 1 as conventionally done), the proposed scheme is proved information-theoretically optimal.

⋆Couillet's work is supported by the MIAI LargeDATA chair at Univ.Grenoble-Alps.

Specifically, the article provides a distributed and scalable extension of the supervised PCA-based multi-task learning algorithm (MTL-SPCA) [9]. For large and numerous data, the resulting algorithm is provably equivalent in performance to the original (and Gaussian-optimal) MTL-SPCA, while simultaneously allowing for drastic cost reductions in data sharing. In particular, the algorithm recovers all advantages from MTL-SPCA, such as the absence of the deleterious problem of *negative transfers*.

Reproducibility. Code of all figures and algorithms provided in the article are available at https://github.com/Sami-fak/DistributedMTLSPCA.

II. Centralized Algorithm

MTL-SPCA [9] is a fast supervised multi-task classification algorithm relying on a preliminary PCA-like step which projects the data onto a *subspace of weighted data classes*. Similar to PCA, MTL-SPCA is a mathematically tractable spectral method. This tractability allows for setting *theoretically optimal data labels* (rather than ± 1), these *labels not being intrinsic to the data but differing for each target task* so to minimize the classification error for each individual task. Of utmost importance here, 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.

Consider k independent agents (i.e., *clients*) solving a classification task on their own data based on training inputs $X_1, \ldots, X_k$, where $X_i \in \mathbb{R}^{p \times n_i}$ is the dataset of client (or *task*) i composed of n_i data of size p. We let $X = [X_1, \ldots, X_k] \in \mathbb{R}^{p \times n}$ ($n = \sum_{t=1}^{k} n_t$) and, for task i, $X_i = [X_{i1}, \ldots, X_{im}]$ for

$X_{ij} \in \mathbb{R}^{p\times n_{ij}}$ the class-j subset ($1 \le j \le m$) for client i (so $n_i = \sum_{j=1}^m n_{ij}$). With a one-versus-all approach, the m-class problem can be decomposed as a series of m 2-class problems: for readability, we thus restrict ourselves to $m = 2$ classes.

We further denote, for class j in task i, $X_{ij} = [x_{i1}^{(j)}, \ldots, x_{in_{ij}}^{(j)}] \in \mathbb{R}^{p\times n_{ij}}$. To each $x_{i\ell}^{(j)}$ is classically associated a label $y_{i\ell}^{(j)} \in \{\pm 1\}$. In [9], this choice is proved largely suboptimal and the main source of negative transfers. Instead, for large p, n_{ij}, if all data in X are independent Gaussian vectors with covariance I_p, the label $y_{i\ell}^{(j)}$ minimizing the classification error *for target task t* should be taken constant for all ℓ and equal to $y_{i\ell}^{(j)} = [\tilde{y}^{[t]}]_{ij}$ where $\tilde{y}^{[t]} \in \mathbb{R}^{2k}$ is the vector:

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

here $e_{tj} = e_{2(t-1)+j} \in \mathbb{R}^{2k}$ is the canonical vector with 1 at the $2(t-1)+j$-th coordinate,

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

is the (fundamental) inter-task similarity matrix,

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

the vector of size ratios with $\mathcal{D}_c = \mathrm{diag}(c)$, $c_0 = p/n$, and M the matrix of statistical means

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

Note importantly that the optimal label vector $\tilde{y}^{[t]}$ for task t *depends on the target task*.

Remark 1 (Estimating $\tilde{y}^{[t]}$)**.** *For large Gaussian data, $M^\mathsf{T}M$ can be effectively estimated as [9]*

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

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

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, $\hat{\tilde{y}}^{[t]}$, defined as $\tilde{y}^{[t]}$ with $M^\mathsf{T}M$ replaced by the above estimates, is a consistent estimate for $\tilde{y}^{[t]}$ as p, n_{ij} grow large.

Remark 1 shows that the optimal $\tilde{y}^{[t]}$ is empirically accessible from the vectors $\frac{1}{n_{ij}}X_{ij}\mathbf{1}_{n_{ij}}$: it does not require to know the individual data.

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

$$V_t = XJ\tilde{y}^{[t]}/\|XJ\tilde{y}^{[t]}\| = Xy/\|Xy\|$$

with $J = \left[j_{11}, \ldots, j_{2k}\right]$, where $j_{tj} = \left[0, \ldots, 0, \mathbf{1}_{n_{tj}}, 0, \ldots, 0\right]^T$, and $y = J\tilde{y}^{[t]}$. Precisely, the error-rate minimizing decision (in a Gaussian setting) results from the test

$$g_t(\mathbf{x}) \equiv V_t^\mathsf{T}\mathbf{x} - \zeta_t \gtrless 0, \ \zeta_t \equiv \frac{1}{2}(\hat{m}_{t1} + \hat{m}_{t2}) \quad (1)$$

Here again, $Xy = \sum_{ij}[\tilde{y}^{[t]}]_{ij}X_{ij}\mathbf{1}_{n_{ij}}$ is a linear combination of the empirical means $\frac{1}{n_{ij}}X_{ij}\mathbf{1}_{n_{ij}}$.

III. Distributed Algorithm

All operations leading to $g_t(\cdot)$ can be written as a function of the empirical means

$$\hat{\mu}_{ij} \equiv \frac{1}{n_{ij}}X_{ij}\mathbf{1}_{n_{ij}}, \quad i \in \{1, \ldots, k\}, \ j \in \{1, 2\}.$$

The complete dataset X, thus, *needs not to be accessible to client t*: only the $\hat{\mu}_{ij}$'s are necessary and MTL-SPCA can be fully distributed by only sharing the local (estimated) statistical means across clients. Or, more efficiently, by updating a set of statistical means within a *central client*, to and from which every client may upload the $\hat{\mu}_{ij}$'s and download V_t. To get rid of the dataset dependence tied to the centralized version, $V_t = Xy/\|Xy\|$ is here rewritten using:

$$Xy = \sum_{i=1}^{k}\sum_{j=1}^{2} n_{ij}\hat{\mu}_{ij}[\tilde{y}^{[t]}]_{ij}. \quad (2)$$

The distributed approach is depicted in Figure 1 for the popular multitask learning Amazon-Caltech-DSLR-Webcam database. Every client is a data center. Target client "webcam" aims to classify images of bags and bikes, exploiting images of the same objects from other data centers slightly differing in their features (number of data, size, shape, resolution, background, diversity, etc). Distributed MTL-SPCA only requires here for the "webcam" (target) client to access the empirical means from the other two data centers.

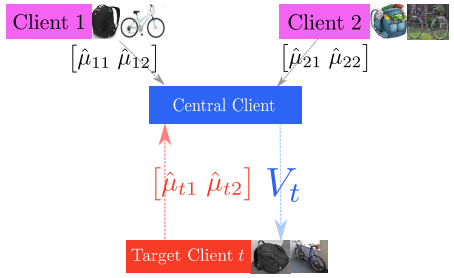

Fig. 1. Schematics of the distributed algorithm for the Amazon (Client 1), Caltech (Client 2), and Webcam (Target Client t) database. Client t draws the projection vector V_t from the empirical statistical means shared by all clients.

Explicitly, using Remark 1 applied to all constituents of $g_t(\cdot)$ – that is, writing Xy and the $\hat{m}_{ij}$'s as functions of the $\hat{\mu}_{ij}$'s –, MTL-SPCA is distributed as per Algorithm 1 (by the central client) and Algorithm 2 (by target task t).

Algorithm 1: Central Client

Data: $\hat{M}_1, \dots \hat{M}_k$ ($\hat{M}_i = [\hat{\mu}_{i1}, \hat{\mu}_{i2}]$), for $\hat{\mu}_{ij} \equiv \frac{1}{n_{ij}} X_{ij} \mathbf{1}_{n_{ij}}$, gathered as $\hat{M} = [\hat{M}_1, \dots, \hat{M}_k] \in \mathbb{R}^{p \times 2k}$;
Estimate similarity $\mathcal{M}$, optimal label $\tilde{y}^{[t]}$, and then projection V_t and decision threshold ζ_t (Remark 1);
Send estimates of V_t, ζ_t to client t.

Algorithm 2: Target task t

Data: Training X_t for task t. Test data $\mathbf{x}$.
Result: Class $\hat{j} \in \{1, 2\}$ of $\mathbf{x}$ for task t.
Compute class empirical means $\hat{\mu}_{t1}, \hat{\mu}_{t2}$;
Send $M_t = [\hat{\mu}_{t1}, \hat{\mu}_{t2}]$ to *Central Client*;
Retrieve V_t, ζ_t from *Central Client*;
Compute score $g_t(\mathbf{x})$ and classify $\mathbf{x}$ in class $\hat{j}$ from decision $g_t(\mathbf{x}) \underset{\hat{j}=2}{\overset{\hat{j}=1}{\gtrless}} \zeta_t$.

Remark 2 (Transmission costs)**.** *A decisive advantage of MTL-SPCA is to allow for a "multi-task on the edge" approach, drastically reducing computational costs compared to a centralized implementation. In operation, the k clients only send $(m-1)k$ size-p vectors ($m-1$ per client) to a* Central Client*, thereby preventing the transfer of the complete datasets (which would require $\sum_{t,c} n_{tc} = n$ size-p vectors): so a $O(n)$-fold gain. This "compacity" of the sufficient statistics also ensures data privacy by emitting an averaged version of individual data.*

IV. Experiments

A. 2-class 2-task transfer learning

In this first setting, class data would classically be labelled as ± 1: of course, MTL-SPCA will overwrite those by optimal labels. Figure 2 illustrates

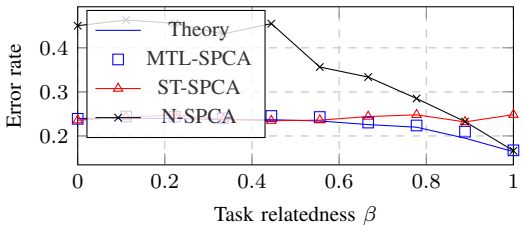

Fig. 2. Classification error for various task similarities (β) in a distributed scenario for classical label ± 1 (N-SPCA), proposed method (MTL-SPCA), and single-task SPCA (ST-SPCA). The theoretical performance for the proposed method is shown in a solid line. Averaged over 10 000 test samples. **MTL-SPCA strongly benefits from increases in task relatedness.**

the empirical versus theoretical (from [9]) classification error performance of MTL-SPCA under this scenario for $x_{tl}^{(j)} \sim \mathcal{N}((-1)^j \mu_t, I_p)$, where $\mu_2 = \beta \mu_1 + \sqrt{1-\beta^2}\mu_1^\perp$ for any $\mu_1^\perp$ orthogonal to μ_1, both of norm $\|\mu_1\| = \|\mu_1^\perp\| = 1$ with $p = 100$, $n_{11} = n_{12} = 1000$ and $n_{21} = n_{22} = 50$. Here $\beta \in [0, 1]$ enforces task relatedness. Unlike ST-SPCA (Single Task), MTL-SPCA benefits from increases in task relatedness and avoids negative transfer (N-SPCA) where labels are set to ± 1.

B. Adding tasks

We next study the increase of the number $k-1$ of source (2-class) tasks, first on synthetic Gaussian and then on real data. For synthetic data, in Figure 3, $x_{tl}^{(j)} \sim \mathcal{N}((-1)^j \mu_t, I_p)$, $\mu_t = \beta \mu + \sqrt{1-\beta^2}\mu^{\perp(t)}$ with $\beta \in [0, 1]$ fixed, $\mu = e_1^{[p]}$ and $\mu^{\perp(t)}$ random of unit norm with $[\mu^{\perp(t)}]_1 = 0$, for $t \in \{1, \dots, k\}$. Tasks are successively added to help

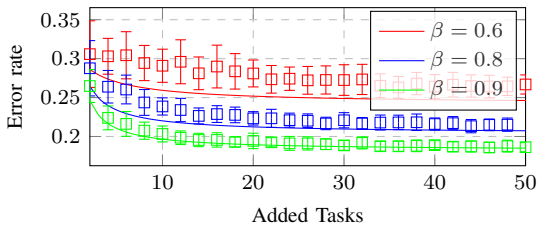

Fig. 3. Theoretical (solid lines) versus empirical (markers) error rates when adding source tasks with relatedness β to target task. **Marked improvement on first added tasks before asymptotic saturation.**

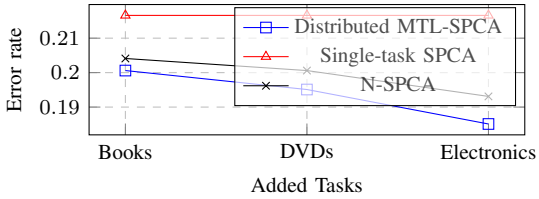

Fig. 4. Classification error for the Amazon Review dataset with "Kitchen" as target task under successive addition of sources. MTL-SPCA (thus with optimized labels $\tilde{y}$) versus single-task SPCA (no source data) and the "naive" (N-PCA) algorithm with labels $y_i \in \{\pm 1\}$. **Transfer learning significantly helps with a clear improvement induced by optimal labels.**

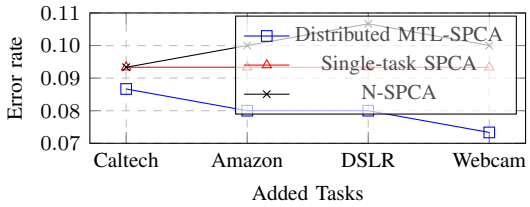

Fig. 5. Classification error for the Office+Caltech dataset, targetting Caltech images of classes "Backpack" and "Touring-Bike"; same algorithms as in Figure 4. **Clear advantage of MTL-SPCA and a strong manifestation of the "negative transfer" effect for N-SPCA.**

classify the target task (task 2) with $n_{t1} = n_{t2} = 50$ for $t \neq 2$ and $n_{21} = n_{22} = 20$, and $p = 100$. Not surprisingly, larger values of β induce higher performance levels with an ultimate saturation as $k \to \infty$. The slight mismatch between theory (from [9]) and practice decreases as $n, p \to \infty$, revealing a rather unexpected phenomenon: that additional degrees of freedom further help the algorithm to "discover" and exploit the affinity between tasks.

Real data arise from Amazon Review (textual user reviews, positive or negative, on books, DVDs, electronics, and kitchen items) and Office31+Caltech (two out of ten classes of images from different modalities: Amazon, Caltech-256, high-resolution DSLR, or low-resolution webcam images) [3], [5]. Figure 4 depicts the classification performance focusing on the "kitchen" task (task 1) by successively adding the three other datasets as sources in the order given in the figure, for $n_{11} = n_{12} = n_{31} = n_{32} = 200$, $n_{21} = n_{41} = 50$, $n_{22} = n_{42} = 90$, $p = 400$. SURF features ($p = 800$) are considered as input data for Office+Caltech images. In Figure 5, features from Amazon (task 2), DSLR (task 3), then webcam images (task 4) are added to classify the "Backpack" and "Touring-Bike" classes of Caltech (task 1), for $n_{11} = 50$, $n_{12} = 60$, $n_{21} = 90$, $n_{22} = 80$, $n_{31} = n_{32} = 12$, $n_{41} = n_{42} = 20$.

Both figures demonstrate the overall superiority of MTL-SPCA. Most importantly, while N-SPCA may severely suffer from negative transfer (performing worse than with no extra task), MTL-SPCA both ignores the negative transfer problem but also improves where N-SPCA decays.

V. Conclusion and Discussion

MTL-SPCA is one example of a simple answer to the difficult multi-task learning problem, yet a solution which is (i) optimal in the (rather large) Gaussian isotropic case, (ii) easy to implement in a distributed manner, and (iii) cost-efficient and environmentally friendly when compared to modern machine learning mechanisms. What it only takes here to make MTL-SPCA so powerful is a statistical analysis and improvement by random matrix theory [9], a comparison to the information-theoretic optima, and a thorough inspection of the incompressible sufficient statistics at play in view of a distributed implementation.

We believe that a more systematic "sober" approach to ML problems, based on modern mathematical techniques, is prone to significantly reduce the environmental footprint of ML algorithms, which is becoming an absolute necessity.

REFERENCES

[1] Eric Bair, Trevor Hastie, Debashis Paul, and Robert Tibshirani. Prediction by supervised principal components. *Journal of the American Statistical Association*, 101(473):119–137, 2006.

[2] Elnaz Barshan, Ali Ghodsi, Zohreh Azimifar, and Mansoor Zolghadri Jahromi. Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds. *Pattern Recognition*, 44(7):1357–1371, 2011.

[3] 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.

[4] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[5] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073, 2012.

[6] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235, 2019.

[7] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[8] Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

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

[10] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[11] Blesson Varghese, Nan Wang, Sakil Barbhuiya, Peter Kilpatrick, and Dimitrios S Nikolopoulos. Challenges and opportunities in edge computing. In *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 20–26. IEEE, 2016.

[12] Qiang Yang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan. *Transfer learning*. Cambridge University Press, 2020.

[13] Yu Zhang and Qiang Yang. An overview of multi-task learning. *National Science Review*, 5(1):30–43, 2018.