Seminario Roma Tre

Learning over networks with non-convex cost functions

Presenter: Simone Scardapane

Dipartimento di Ingegneria dell'Informazione, Elettronica e Telecomunicazioni (DIET)

SAPIENZA
UNIVERSITÀ DI ROMA

Table of contents

A motivating scenario

Consider a ‘smart environment’ with hundreds of sensors; among them, embedded cameras taking high-res photos of possible security threats.

A motivating scenario

Consider a 'smart environment' with hundreds of sensors; among them, embedded cameras taking high-res photos of possible security threats.

- We can learn to recognize threats by training some predictive model (e.g., a neural network).

A motivating scenario

Consider a 'smart environment' with hundreds of sensors; among them, embedded cameras taking high-res photos of possible security threats.

- We can learn to recognize threats by training some predictive model (e.g., a neural network).
- Can we do this without sending the pictures to a centralized location?

A motivating scenario

Consider a 'smart environment' with hundreds of sensors; among them, embedded cameras taking high-res photos of possible security threats.

- We can learn to recognize threats by training some predictive model (e.g., a neural network).
- Can we do this without sending the pictures to a centralized location?
- Actually, can we do this without sending the pictures around *at all*?

Distributed supervised learning

Distributed learning (DL) refers to training models from data *distributed* over a set of agents (e.g., smart sensors).

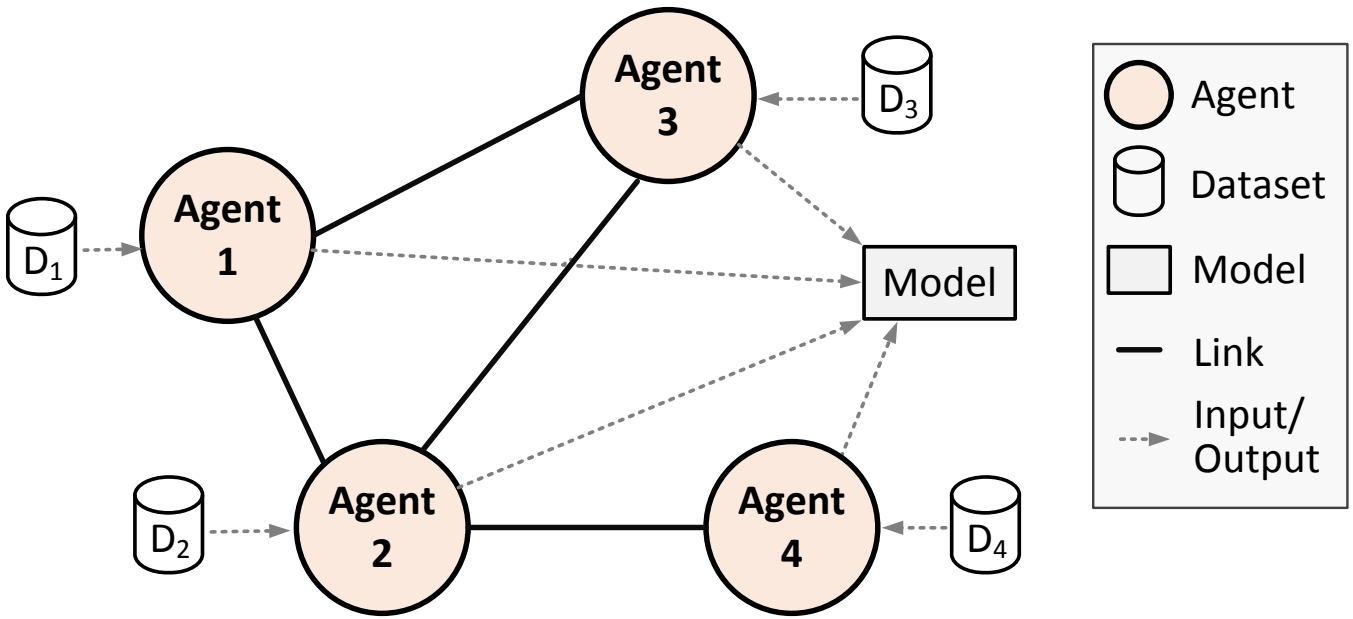


Figure 1 : Example of distributed learning with four agents agreeing on a common (e.g., neural network) model.

Constraints

We are interested in algorithms with the following constraints:

Communication Communication can be *sparse*, time-varying, with no centralization.

Privacy Exchange of training data is generally to be avoided.

Scalability Agents' networks can be very large, following many different topologies.

Some more motivation

Example of possible applications:

- Classification over peer-to-peer networks.
- Image recognition in *ad-hoc* sensor networks.
- Inference in distributed medical scenarios.
- Tracking and learning in a robotic swarm.

State-of-the-art

Distributed learning with *convex* objective functions is well established in the literature:

- **Kernel Ridge Regression**
  [Predd, Kulkarni and Poor, IEEE SPM, 2006]
- **Sparse Linear Regression**
  [Mateos, Bazerque and Giannakis, IEEE TSP, 2010]
- **Support Vector Machines**
  [Forero, Cano and Giannakis, JMLR, 2010]
- **Local convex solvers & communication**
  [Jaggi et al., NIPS, 2014]

This reflects the availability of general-purpose methods for distributed optimization of convex losses, e.g. the Alternating Direction Method of Multipliers.

Aims of the talk

1. We formalize the DL problem in a general way.
2. We describe a simple extension of gradient descent which includes communication steps to *diffuse* information.
3. We describe a more complex formulation based on a novel framework called **in-NEtwork nonconveX opTimization** (NEXT).
4. We show some experimental results and future lines of research.

Table of contents

Problem formulation

Distributed training of a model $f(\mathbf{w}; \mathbf{x}) \in \mathcal{H}$ can be cast as the minimization of a social cost function G plus a regularization term R:

$$\min_{\mathbf{w}} \; U(\mathbf{w}) = G(\mathbf{w}) + R(\mathbf{w}) = \sum_{i=1}^{I} G_i(\mathbf{w}) + R(\mathbf{w}) \,, \qquad (1)$$

where $G_i(\cdot)$ is the local cost function of agent i, defined as:

$$G_i(\mathbf{w}) = \sum_{m \in \mathcal{S}_i} L(d_{i,m}, f(\mathbf{w}; \mathbf{x}_{i,m})) \,, \qquad (2)$$

where $L(\cdot, \cdot)$ is a loss function, $(\mathbf{x}_{i,m}, d_{i,m})$ is a training example, and $\mathcal{S}_i$ is the ith local dataset.

Network model

- In this talk, we consider a fixed network topology where the ith agent is connected to a set of neighbors $\mathcal{N}_i$.
- We introduce weights c_{ij}, matching this topology, to fuse information:

$$\mathrm{c_{ij}} = \begin{cases} \theta_{\mathrm{ij}} \in [\vartheta, 1] & \text{if } \mathrm{j} \in \mathcal{N}_{\mathrm{i}}^{\mathrm{in}}; \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

  for some $\vartheta \in (0, 1)$, and define the matrix $\mathbf{C} \triangleq (c_{ij})_{i,j=1}^{I}$ which must respect:

$$\mathbf{C}\,\mathbf{1} = \mathbf{1} \quad \text{and} \quad \mathbf{1}^T\mathbf{C} = \mathbf{1}^T \quad . \tag{4}$$

- The weights define the communication topology.

Baseline: distributed gradient descent

A simple baseline to solve the above problems is a distributed gradient descent (DGD) procedure:

$$\boldsymbol{\psi}_i = \mathbf{w}_i[n] - \alpha[n]\nabla h_i(\mathbf{w}_i[n])\,, \tag{5}$$

$$\mathbf{w}_i[n+1] = \sum_{j\in\mathcal{N}_i} c_{ji}\boldsymbol{\psi}_j\,, \tag{6}$$

where $h_i(\mathbf{w}_i[n]) = G_i(\mathbf{w}_i[n]) + \frac{1}{I}R(\mathbf{w}_i[n])$.

---

[1] Bianchi, P., & Jakubowicz, J. (2013). **Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization**. *IEEE Trans. on Automatic Control*, 58(2), pp. 391-405.

[2] Sayed, A. H. (2014). **Adaptive networks**. *Proc. of the IEEE*, 102(4), pp. 460-497.

Example of diffusing information

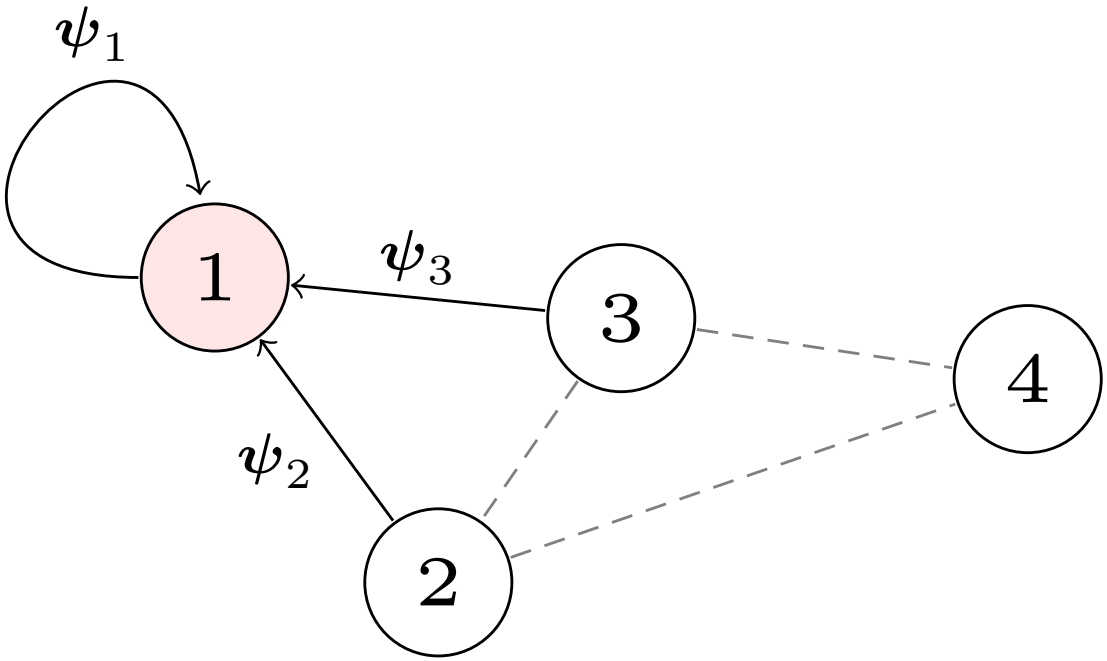


Figure 2 : An example of a diffusion step relative to agent 1 in a simple network with 4 agents. Gray links are inactive.

Table of contents

Step 1 - Local optimization

At every step, a local estimate $\mathbf{w}_i[n]$ is obtained by solving a strongly convex surrogate of the original cost function:

$$\widetilde{\mathbf{w}}_i[n] = \arg\min_{\mathbf{w}_i} \ \widetilde{G}_i(\mathbf{w}_i; \mathbf{w}_i[n]) + \boldsymbol{\pi}_i[n]^T(\mathbf{w}_i - \mathbf{w}_i[n]) + r(\mathbf{w}_i)\,,$$

where

$$\boldsymbol{\pi}_i[n] \triangleq \sum_{j \neq i} \nabla_{\mathbf{w}} \ G_j(\mathbf{w}_i[n]) \tag{7}$$

and $\widetilde{G}_i(\mathbf{w}_i; \mathbf{w}_i[n])$ is a convex approximation of G_i at the point $\mathbf{w}_i[n]$, preserving the first order properties of G_i.

$\boldsymbol{\pi}_i[n]$ is not available to the agents and must be approximated.

Step 2 - Computation of new estimate

The new estimate is obtained as the convex combination:

$$\mathbf{z}_i[n] = \mathbf{w}_i[n] + \alpha[n]\left(\widetilde{\mathbf{w}}_i[n] - \mathbf{w}_i[n]\right) , \tag{8}$$

where $\alpha[n]$ is a possibly time-varying step-size sequence.

Step 3 - Consensus phase

Each agent i updates $\mathbf{w}_i[n]$ with a consensus procedure:

$$\mathbf{w}_i[n+1] = \sum_{j \in \mathcal{N}_i^{in}} c_{ij}\, \mathbf{z}_i[n]. \tag{9}$$

Step 3 - Consensus phase

Each agent i updates $\mathbf{w}_i[n]$ with a consensus procedure:

$$\mathbf{w}_i[n+1] = \sum_{j \in \mathcal{N}_i^{in}} c_{ij}\, \mathbf{z}_i[n]. \tag{9}$$

$\boldsymbol{\pi}_i[n]$ is replaced with an estimate $\widetilde{\boldsymbol{\pi}}_i[n]$, asymptotically converging to $\boldsymbol{\pi}_i[n]$. We can update the local estimate $\widetilde{\boldsymbol{\pi}}_i[n]$ as:

$$\widetilde{\boldsymbol{\pi}}_i[n] \triangleq I \cdot \mathbf{y}_i[n] - \nabla G_i(\mathbf{w}_i[n]), \tag{10}$$

where $\mathbf{y}_i[n]$ is a local auxiliary variable to asymptotically track the average of the gradients, updated as:

$$\mathbf{y}_i[n+1] \triangleq \sum_{j=1}^{I} c_{ij}\mathbf{y}_j[n] + (\nabla G_i(\mathbf{w}_i[n+1]) - \nabla G_i(\mathbf{w}_i[n])) \ . \tag{11}$$

Distributed training of neural networks

Consider the distributed training of a neural network (NN) model. In this case, we can linearize only the NN mapping as:

$$\widetilde{f}(\mathbf{w}_i; \mathbf{w}_i[n], \mathbf{x}_{i,m}) = f(\mathbf{w}_i[n]; \mathbf{x}_{i,m}) + \mathbf{J}_{i,m}[n]^T(\mathbf{w}_i - \mathbf{w}_i[n]) \quad (12)$$

where

$$[\mathbf{J}_{i,m}[n]]_{kl} = \frac{\partial f_k(\mathbf{w}_i[n]; \mathbf{x}_{i,m})}{\partial \mathrm{w}_l}\,. \quad (13)$$

is the *weight Jacobian* relative to that example. We call this the *partial linearization* of the cost function.

References

[1] Di Lorenzo, P. and Scutari, G., 2016. **Next: In-network nonconvex optimization**. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2), pp. 120-136.

[2] Scardapane, S., Fierimonte, R., Di Lorenzo, P., Panella, M. and Uncini, A., 2016. **Distributed semi-supervised support vector machines**. *Neural Networks*, 80, pp. 43-52.

[3] Di Lorenzo, P. and Scardapane, S., 2016. **Parallel and distributed training of neural networks via successive convex approximation**. In *2016 IEEE 26th Int. Work. on Machine Learning for Sign. Process. (MLSP)* (pp. 1-6). IEEE.

[4] Scardapane, S. and Di Lorenzo, P., 2016. **A Framework for Parallel and Distributed Training of Neural Networks**. *arXiv preprint* arXiv:1610.07448 [cond. accepted on Neural Networks].

A practical example

We consider a squared loss with ℓ_2 norm regularization. Define:

$$\mathbf{A}_i[n] = \sum_{m=1}^{M} \mathbf{J}_{i,m}^T[n]\mathbf{J}_{i,m}[n] + \lambda \mathbf{I}\,, \tag{14}$$

$$\mathbf{b}_i[n] = \sum_{m=1}^{M} \mathbf{r}_{i,m}^T[n]\mathbf{J}_{i,m}[n]\,. \tag{15}$$

with

$$\mathbf{r}_{i,m}[n] = \mathbf{d}_{i,m} - f(\mathbf{w}_i[n]; \mathbf{x}_{i,m}) + \mathbf{J}_{i,m}[n]\mathbf{w}_i[n]\,. \tag{16}$$

The solution is given in closed form as:

$$\widetilde{\mathbf{w}}_i[n] = \mathbf{A}_i^{-1}[n](\mathbf{b}_i[n] - 0.5 \cdot \widetilde{\boldsymbol{\pi}}_i[n])\,. \tag{17}$$

Table of contents

Setup

We consider several benchmark datasets, keeping a test set, and splitting the training set over networks of 10 agents:

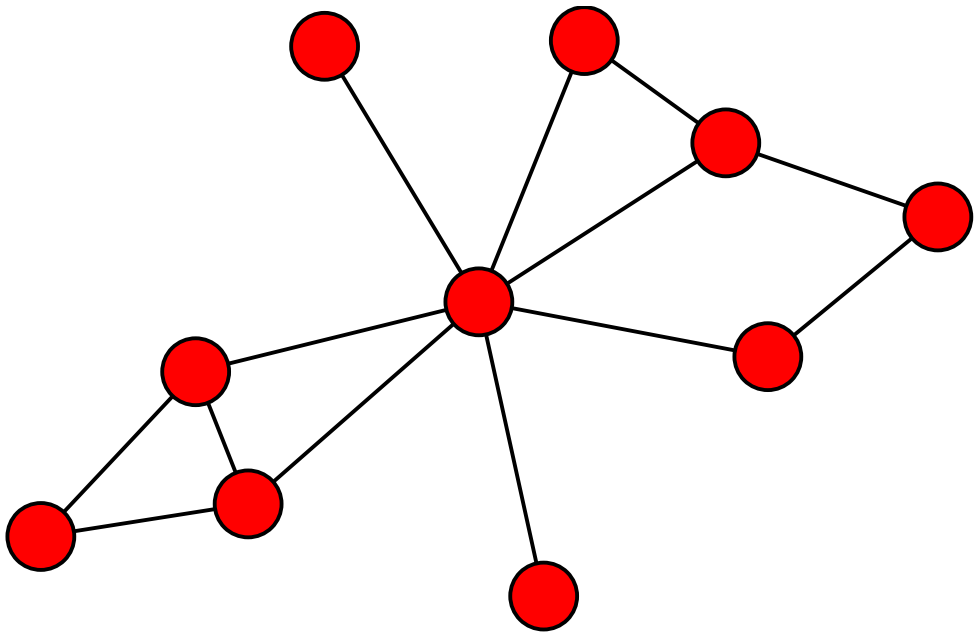
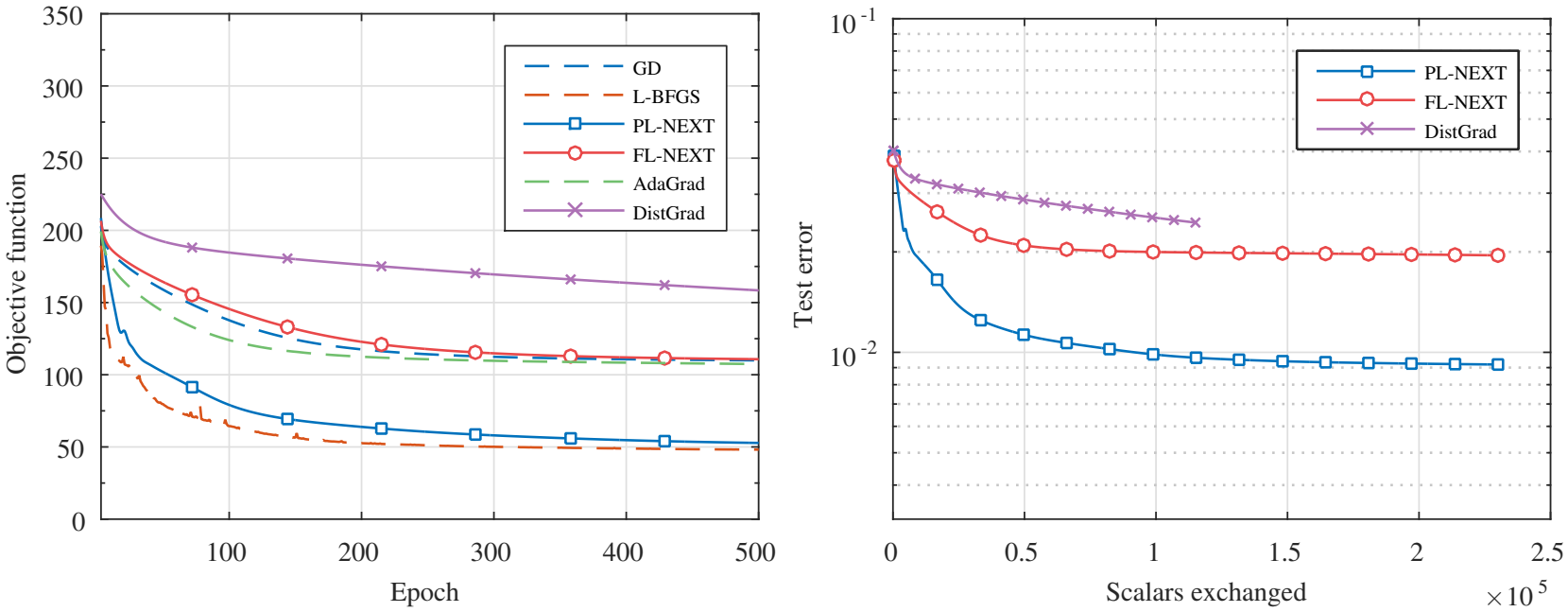


Figure 3 : Example of communication network with 10 agents (represented by red dots), possessing a sparse, time-invariant, symmetric connectivity.

Some results



(a) Objective function (b) Test error

Figure 4 : Objective function evolution and test error for a representative dataset.

Table of contents

Conclusions

1. Distributed learning with non-convex losses is an exciting field with a variety of possible applications.
2. The availability of general-purpose distributed optimization tools is very recent and almost unexplored.
3. Future work can include the possibility of stochastic optimization, asynchronous networks, and the extension to additional classes of problems.

Thanks for the attention,

questions?