Fifth Italian Workshop on Machine Learning and Data Mining (MLDM)

Learning over networks with non-convex cost functions

Presenter: Simone Scardapane

Dipartimento di Ingegneria dell’Informazione,
Elettronica e Telecomunicazioni (DIET)
# Table of contents

1. **Introduction**
   - Distributed supervised learning
   - State-of-the-art

2. **The NEXT Framework**
   - Problem Formulation
   - Derivation of the NEXT Algorithm

3. Application to distributed semi-supervised learning

4. Application to distributed training of neural networks

5. Conclusions
Distributed learning (DL) refers to the inference of a model from data distributed over a set of agents (e.g., smart sensors).

**Figura 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, without any centralization.

**Privacy**  Exchange of training data is generally to be avoided.

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

Example of possible applications:

- Classification over peer-to-peer networks.
- Image recognition in *ad-hoc* sensor networks.
- Inference in distributed medical scenarios.
Distributed learning with *convex* objective functions is well established:

- **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 ADMM.
Our contribution

We consider distributed inference problems with non-convex loss functions. Two examples are:

1. Distributed semi-supervised learning with a smooth version of the semi-supervised support vector machine.

2. Distributed training of neural network models.

In particular, we extend a novel framework called **in-NEtwork non-conveX opTimization** (NEXT), combining a convexification-decomposition technique and a dynamic consensus procedure [Di Lorenzo and Scutari, IEEE TSIPN, 2016].
Table of contents

1 Introduction
   Distributed supervised learning
   State-of-the-art

2 The NEXT Framework
   Problem Formulation
   Derivation of the NEXT Algorithm

3 Application to distributed semi-supervised learning

4 Application to distributed training of neural networks

5 Conclusions
Problem formulation

Distributed training of a model \( f(w; x) \in \mathcal{H} \) can be cast as the minimization of a social cost function \( G \) plus a regularization term \( r \): \[ \min_w U(w) = G(w) + r(w) = \sum_{i=1}^{I} g_i(w) + r(w) , \] (1)

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

\[ g_i(w) = \sum_{m \in S_i} l(d_{i,m}, f(w; x_{i,m})) , \] (2)

where \( l(\cdot, \cdot) \) is a loss function, and \((x_{i,m}, d_{i,m})\) is a training example.
Network model

• In this talk, we consider a fixed network topology where the $i$th agent is connected to a set of neighbors $\mathcal{N}_i$.

• We introduce weights $c_{ij}$, matching this topology, to fuse information:

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

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

$$\mathbf{C} \mathbf{1} = \mathbf{1} \quad \text{and} \quad \mathbf{1}^T \mathbf{C} = \mathbf{1}^T.$$  \hspace{1cm} (4)

• The weights define the communication topology.
A simple baseline to solve the above problems is a distributed gradient descent (DGD) procedure:

\[
\psi_i = w_i[n] - \alpha[n] \nabla h_i(w_i[n]), \tag{5}
\]

\[
w_i[n + 1] = \sum_{j \in \mathcal{N}_i} C_{ji} \psi_j, \tag{6}
\]

where \( h_i(w_i[n]) = g_i(w_i[n]) + \frac{1}{I} r(w_i[n]) \).


Step 1 - Local optimization

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

$$
\tilde{w}_i[n] = \arg \min_{w_i} \tilde{g}_i(w_i; w_i[n]) + \pi_i[n]^T(w_i - w_i[n]) + r(w_i),
$$

where

$$
\pi_i[n] \triangleq \sum_{j \neq i} \nabla_w g_j(w_i[n])
$$

and $\tilde{g}_i(w_i; w_i[n])$ is a convex approximation of $g_i$ at the point $w_i[n]$, preserving the first order properties of $g_i$.

$\pi_i[n]$ 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:

\[ z_i[n] = w_i[n] + \alpha[n] (\tilde{w}_i[n] - w_i[n]) \quad (9) \]

where \( \alpha[n] \) is a possibly time-varying step-size sequence.
Step 3 - Consensus phase

Each agent $i$ updates $w_i[n]$ with a consensus procedure:

$$w_i[n + 1] = \sum_{j \in N_i^{in}} c_{ij} z_i[n],$$

(10)

Finally, we replace $\pi_i[n]$ with a local estimate $\tilde{\pi}_i[n]$, asymptotically converging to $\pi_i[n]$. We can update the local estimate $\tilde{\pi}_i[n]$ as:

$$\tilde{\pi}_i[n] \triangleq I \cdot y_i[n] - \nabla g_i(w_i[n]),$$

(11)

where $y_i[n]$ is a local auxiliary variable to asymptotically track the average of the gradients, updated as:

$$y_i[n + 1] \triangleq \sum_{j=1}^{I} c_{ij} y_j[n] + (\nabla g_i(w_i[n + 1]) - \nabla g_i(w_i[n])).$$

(12)
Table of contents

1 Introduction
   Distributed supervised learning
   State-of-the-art

2 The NEXT Framework
   Problem Formulation
   Derivation of the NEXT Algorithm

3 Application to distributed semi-supervised learning

4 Application to distributed training of neural networks

5 Conclusions
Distributed semi-supervised learning

Assume a binary classification problem where some data at every agent is *unlabeled*. Following the standard semi-supervised support vector machine (S\(^3\)VM) approach we can use the following cost function [1]:

\[
l(d_{i,m}, f(w; x_{i,m})) = \max \left( 0, 1 - \hat{d}_{i,m} f(w; x_{i,m}) \right)^2
\] (13)

where \(\hat{d}_{i,m}\) is the true label for labeled data, or it is included in the optimization problem for unlabeled data. For simplicity, we consider a linear SVM with \(\ell_2\) regularization:

\[
f(w; x_{i,m}) = w^T x_{i,m} + b,
\] (14)

\[
r(w) = \lambda \| w \|_2^2.
\] (15)

---

Figura 2: A possible approximation of the hinge loss on unlabeled data.
A smooth approximation

A smooth approximation is obtained by substituting the hinge loss on unlabeled samples with an exponential approximation [1]:

\[
\hat{l}(d_{i,m}, f(w; x_{i,m})) = \exp \left\{ -5f(w; x_{i,m})^2 \right\} .
\]  

(16)

Being non-convex, we perform an additional linearization to apply the distributed framework:

\[
\tilde{l}(w_i; w_i[n]) \triangleq \hat{l}(d_{i,m}, f(w_i[n]; x_{i,m}))+
\[
\nabla \hat{l}(d_{i,m}, f(w_i[n]; x_{i,m})) (w_i - w_i[n]) .
\]  

(17)

At every step, each agent solves a problem which is roughly equivalent to a linear SVM optimization.


Experimental setup

We consider a binary classification problem on music songs taken from the GARAGEBAND dataset.

We run a 10-fold cross-validation repeated 15 times.

All optimization problems are solved with a gradient descent procedure with $T = 500$ maximum iterations.

Step sizes are chosen according to:

$$\alpha[n] = \frac{\alpha[0]}{(n + 1)^{\delta}}.$$
Figura 3: Evolution of objective function and gradient norm on a network of 25 agents.
(a) Classification error

(b) Training time

Figura 4: Classification error and training time when varying agents from 5 to 40.
Table of contents

1 Introduction
   Distributed supervised learning
   State-of-the-art

2 The NEXT Framework
   Problem Formulation
   Derivation of the NEXT Algorithm

3 Application to distributed semi-supervised learning

4 Application to distributed training of neural networks

5 Conclusions
Distributed training of neural networks

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

\[
\tilde{f}(w_i; w_i[n], x_{i,m}) = f(w_i[n], x_{i,m}) + J_{i,m}[n]^T(w_i - w_i[n])
\]  

(18)

where

\[
[J_{i,m}[n]]_{kl} = \frac{\partial f_k(w_i[n]; x_{i,m})}{\partial w_l}
\]

(19)

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


A practical example

We consider a squared loss and an $\ell_2$ norm regularization. Define:

$$A_i[n] = \sum_{m=1}^{M} J_{i,m}[n] J_{i,m}[n] + \lambda I,$$  \hspace{1cm} (20)

$$b_i[n] = \sum_{m=1}^{M} r_{i,m}[n] J_{i,m}[n].$$  \hspace{1cm} (21)

with

$$r_{i,m}[n] = d_{i,m} - f(w_i[n]; x_{i,m}) + J_{i,m}[n] w_i[n].$$  \hspace{1cm} (22)

The solution is given in closed form as:

$$\tilde{w}_i[n] = A_i^{-1}[n](b_i[n] - 0.5 \cdot \tilde{\pi}_i[n]).$$  \hspace{1cm} (23)
Some results

![Objective function and test error graphs](image)

**Figura 5**: Objective function evolution and test error for a representative dataset.
<table>
<thead>
<tr>
<th>1</th>
<th>Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distributed supervised learning</td>
</tr>
<tr>
<td></td>
<td>State-of-the-art</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>The NEXT Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Problem Formulation</td>
</tr>
<tr>
<td></td>
<td>Derivation of the NEXT Algorithm</td>
</tr>
</tbody>
</table>

| 3 | Application to distributed semi-supervised learning |

| 4 | Application to distributed training of neural networks |

| 5 | Conclusions |
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?