Ambos, Henrik; Tran, Nguyen; Jung, Alexander

Classifying Big Data over Networks Via the Logistic Network Lasso

Published in:
2018 CONFERENCE RECORD OF 52ND ASILOMAR CONFERENCE ON SIGNALS, SYSTEMS, AND COMPUTERS

DOI:
10.1109/ACSSC.2018.8645260

Published: 19/02/2019

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Ambos, H., Tran, N., & Jung, A. (2019). Classifying Big Data over Networks Via the Logistic Network Lasso. In M. B. Matthews (Ed.), *2018 CONFERENCE RECORD OF 52ND ASILOMAR CONFERENCE ON SIGNALS, SYSTEMS, AND COMPUTERS* (Vol. 2018-October, pp. 855-858). Article 8645260 IEEE. https://doi.org/10.1109/ACSSC.2018.8645260

CLASSIFYING BIG DATA OVER NETWORKS VIA THE LOGISTIC NETWORK LASSO

Henrik Ambos, Nguyen Tran and Alexander Jung

Department of Computer Science, Aalto University, Finland; firstname.lastname(at)aalto.fi

ABSTRACT

We apply network Lasso to solve binary classification and clustering problems on network structured data. In particular we generalize ordinary logistic regression to non-Euclidean data defined over a complex network structure. The resulting logistic network Lasso classifier amounts to solving a convex optimization problem. A scalable classification algorithm is obtained by applying the alternating direction methods of multipliers.

Index Terms— compressed sensing, big data over networks, semi-supervised learning, classification, clustering, complex networks, convex optimization

I. INTRODUCTION

We consider the problem of classifying or clustering a large set of data points which conform to an underlying network structure. Such network structured datasets arise in a wide range of important application domains, e.g., image- and video processing, and social networks [1].

A number of recent works studied fundamental limits and efficient methods for signal processing and machine learning for massive network-structured datasets [2]–[4]. In particular, the recently introduced extension of the least absolution selection and shrinkage operator (Lasso) to network-structured data, coined the network Lasso (nLasso), allows to tackle the problem of processing massive network-structured datasets using modern convex optimization methods [4].

Most of the existing work on nLasso based methods have focused on the squared error loss which is suitable for predicting numeric target variables. In contrast, we apply nLasso to binary classification problems using the logistic loss function to measure the empirical (or training) error incurred by a classifier. This results in a particular instance of regularized empirical risk minimization with the total variation of the classifier as regularization term. This is a convex optimization problem which we solve using the alternating direction method of multipliers (ADMM) [5].

The paper is organized as follows: We formalize the problem of classifying network structured data as a convex optimization problem in Section II. As detailed in Section III, this optimization problem, which we coin logistic nLasso (lnLasso), can be solved efficiently by applying ADMM which results in a message passing algorithm. The results of illustrative numerical experiments are discussed in Section IV.

II. PROBLEM FORMULATION

We consider network structured datasets that can be represented by an undirected weighted graph (the "empirical graph") $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$. The nodes $i \in \mathcal{V}$ represent individual data points, some of which are connected by an undirected edges $\{i,j\} \in \mathcal{E}$ if data point i and j are considered similar. Each edge $\{i,j\}$ is associated with a positive weight W_{ij} whose strength quantifies a similarity between data points i and j.

On top of the network structure (which is encoded in the weight matrix $\mathbf{W}$), datasets often convey additional information such as labels y_i of the data points $i \in \mathcal{V}$. E.g., in a social network, the label y_i might encode membership to a particular social group. In what follows, we focus on binary classification problems, i.e., those involving binary labels $y_i \in \{-1, 1\}$.

The acquisition of reliable label information is often costly so that we assume to have access only to the labels y_i for the nodes $i \in \mathcal{M}$ belonging to a small sampling set $\mathcal{M} \subseteq \mathcal{V}$ of size $M = |\mathcal{M}| \ll |\mathcal{V}|$. Thus, since the true (correct) labels y_i are typically not known for most data points $i \in \mathcal{V}$, we model them as random variables whose distribution is characterized by the log odds ratio

$$x[i] = \log \frac{\text{Prob}\{y_i = 1\}}{\text{Prob}\{y_i = -1\}}. \tag{1}$$

We interpret the ratio $x[i]$ as the signal value at node i of a graph signal $x[\cdot] : \mathcal{V} \rightarrow \mathbb{R}$.

A graph signal $x[\cdot]$ can be used as a classifier (or predictor), by classifying a data point $i \in \mathcal{V}$ as $\hat{y}_i = 1$ if $x[i] > 0$ and $\hat{y}_i = -1$ otherwise (cf. (1)). The classifier $x[\cdot]$ should agree with the available label information $\{y_i\}_{i \in \mathcal{M}}$. Using the probabilistic model (1), a principled approach to choosing $x[\cdot]$ is by maximizing the likelihood of observing $\{y_i\}_{i \in \mathcal{M}}$ or, equivalently, to minimize the empirical error

$$\widehat{E}(x[\cdot]) := \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \ell(y_i x[i]) \tag{2}$$

with the logistic loss function

$$\ell(z) := \log(1 + \exp(-z)). \tag{3}$$

However, using only the criterion (2) for learning a classifier $x[\cdot]$ is not sufficient as (2) is invariant to all signal values $x[i]$ at non-sampled nodes $i \in \mathcal{V} \setminus \mathcal{M}$.

In order to recover the entire classifier $x[\cdot] : \mathcal{V} \rightarrow \mathbb{R}$ from the incomplete information $\{y_i\}_{i \in \mathcal{M}}$, we need more structure. This additional structure is provided by the empirical graph $\mathcal{G}$ which relates the individual data points which we aim to classify. In particular, we assume that any reasonable classifier $x[\cdot]$ for a data set with underlying empirical graph $\mathcal{G}$ needs to conform with the underlying graph structure. The amount to which the graph signal $x[\cdot]$ agrees with the network structure can be measured using the total variation (TV)

$$\|x[\cdot]\|_{\mathrm{TV}} := \sum_{\{i,j\} \in \mathcal{E}} W_{ij} \, |x[j] - x[i]| \,. \tag{4}$$

Having a small TV means that the signal values $x[i]$ are approximately constant over well connected subset of nodes ("clusters"). This "clustering hypothesis" is underlying many methods for (semi-) supervised learning [6].

In view of the above discussion, we are lead quite naturally to learning a classifier $x[i]$ based on few initial labels $\{y_i\}_{i \in \mathcal{M}}$ by balancing a small empirical error $\widehat{E}(x[\cdot])$ (cf. (2)) with a small TV $\|x[\cdot]\|_{\mathrm{TV}}$ (cf. (4)). Thus, we arrive at the following optimization problem

$$\hat{x}[\cdot] \in \underset{x[\cdot] \in \mathbb{R}^{\mathcal{V}}}{\arg\min} \, \widehat{E}(x[\cdot]) + \lambda \|x[\cdot]\|_{\mathrm{TV}}. \tag{5}$$

The regularization parameter λ in (5) allows to trade-off a small TV of the predictor $x[\cdot]$ against a small empirical error, i.e., a small average logistic loss incurred for the labelled data points $i \in \mathcal{M}$. Choosing a small value of λ will result in a classifier $\hat{x}[\cdot]$ with a small empirical error $\widehat{E}(x[\cdot])$ (cf. (2)), while choosing a large value of λ favours a classifier $\hat{x}[\cdot]$ with a small TV $\|\hat{x}[\cdot]\|_{\mathrm{TV}}$.

We refer to the learning problem (5) as lnLasso. It amounts to a non-smooth convex optimization problem. In what follows we show how to obtain an efficient implementation of lognLasso in the form of message passing over the underlying empirical graph $\mathcal{G}$.

III. LOGISTIC NETWORK LASSO VIA ADMM

A standard trick to obtain distributed methods for solving convex optimization problems of the form (5) is to introduce, for each each pair of connected nodes $i, j \in \mathcal{V}$, an auxiliary variables z_{ij}. These auxiliary variables act as local copies of the global optimization variables $x[i]$, i.e., we require

$$z_{ij} = x[j] \text{ for all } i \in \mathcal{V} \text{ and } j \in \mathcal{N}(i). \tag{6}$$

We can then reformulate lnLasso (5) as

$$\begin{aligned} \hat{x}[\cdot] \in & \underset{x[\cdot] \in \mathbb{R}^{\mathcal{V}}}{\arg\min} \, \widehat{E}(x[\cdot]) + \lambda \sum_{\{i,j\} \in \mathcal{E}} W_{i,j} |z_{ij} - z_{ji}| \\ & \text{s.t.} \quad x[i] = z_{ij} \quad i \in \mathcal{V}, \quad j \in \mathcal{N}(i). \end{aligned} \tag{7}$$

The reformulation (7) of the lnLasso is appealing since the objective function consists of two independent terms, the empirical risk $\widehat{E}(x[\cdot])$ and the scaled TV $\lambda \sum_{\{i,j\} \in \mathcal{E}} W_{i,j} |z_{ij} - z_{ji}|$. However, both terms are coupled via the consistency constrains (6).

In order to solve the non-smooth convex optimization problem (7), we apply ADMM. To this end, we define the augmented Lagrangian of (7) as [5]

$$\begin{aligned} \mathcal{L}(x, z, u) := & \widehat{E}(x) + \lambda \sum_{\{i,j\} \in \mathcal{E}} W_{ij} \, |z_{ij} - z_{ji}| \qquad (8) \\ & + \frac{\rho}{2} \sum_{\{i,j\} \in \mathcal{E}} \big[(x[i] - z_{ij} + u_{ij})^2 - u_{ij}^2 + (x[j] - z_{ji} + u_{ji})^2 - u_{ji}^2 \big]. \end{aligned}$$

ADMM amounts to coordinate-wise optimization of the augment Lagrangian by iterating the following updates:

$$x^{(k+1)}[\cdot] := \underset{x[\cdot] \in \mathbb{R}^{\mathcal{V}}}{\arg\min} \, \mathcal{L}(x, z^{(k)}, u^{(k)}) \tag{9}$$

$$z^{(k+1)}[\cdot] := \underset{z[\cdot] \in \mathbb{R}^{\mathcal{V} \times \mathcal{V}}}{\arg\min} \, \mathcal{L}(x^{(k+1)}, z, u^{(k)}) \tag{10}$$

$$u_{ij}^{(k+1)} := u_{ij}^{(k)} + x^{(k+1)}[i] - z_{ij}^{(k+1)}. \tag{11}$$

At a sampled node $i \in \mathcal{M}$, the update (9) becomes

$$\begin{aligned} & x^{(k+1)}[i] = \qquad (12) \\ & \underbrace{\underset{x \in \mathbb{R}}{\arg\min} \, \ell(y_i x) + \frac{M\rho}{2} \sum_{j \in \mathcal{N}(i)} (x - z_{ij}^{(k)} + u_{ij}^{(k)})^2}_{:= f(x)} . \end{aligned}$$

The presence of the logistic loss function (cf. (3)) does not allow for a simple closed-form solution of (12). However, since (12) is an unconstrained convex optimization problem over a real-valued variable $x \in \mathbb{R}$ and with a smooth objective function, we can apply Newton method. To this end, note that solving (12) amounts to finding roots of

$$f'(x) = -\frac{y_i}{M(1 + e^{y_i x})} + \rho \sum_{j \in \mathcal{N}(i)} (x - z_{ij}^{(k)} + u_{ij}^{(k)}).$$

Newton's method iterates the updates

$$x^{(n+1)} = x^{(n)} - \frac{f'(x^{(n)})}{f''(x^{(n)})} \tag{13}$$

with the second derivative (cf. (12))

$$f''(x) = \frac{y_i^2 e^{y_i x}}{M(1 + e^{y_i x})^2} + \rho \, |\mathcal{N}(i)| \,.$$

Starting from any initial guess $x^{(0)}$, the iterates (13) converge rapidly to the solution of (12). At non-sampled nodes $i \in \mathcal{V} \setminus \mathcal{M}$, the update (9) becomes

$$x = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \big(z_{ij} - u_{ij} \big). \tag{14}$$

The update (10) can be worked out as (see, e.g., [4])

$$z_{ij}^{(k+1)} = \theta(x_i^{(k+1)} + u_{ij}^{(k)}) + (1-\theta)(x_j^{(k+1)} + u_{ji}^{(k)})$$
$$z_{ji}^{(k+1)} = (1-\theta)(x_i^{k+1} + u_{ij}^{(k)}) + \theta(x_j^{(k+1)} + u_{ji}^{(k)}).$$

, with

$$\theta = \max\left(1/2, 1 - \frac{\lambda W_{ij}}{\rho|x_i^{(k+1)} + u_{ij}^{(k)} - x_j^{(k+1)} - u_{ji}^{(k)}|}\right). \quad (15)$$

We are now ready to write down the ADMM implementation of lnLasso as Algorithm 1.

Algorithm 1 Logistic Network Lasso via ADMM

```
Input: empirical graph $\mathcal{G}$, sampling set $\mathcal{M}$, initial labels $\{y_i\}_{i\in\mathcal{M}}$.
Initialize: $k := 0$ and $x^{(0)}[i] = 0$, $z_{ij}^{(0)} = 0$ and $u_{ij}^{(0)} = 0$
 1: repeat
 2:     for $i \in \mathcal{V}$ do
 3:         if $i \in \mathcal{M}$ then
 4:             update $x^{(k+1)}[i]$ according to (12)
 5:         else
 6:             update $x^{(k+1)}[i]$ according to (14)
 7:         end if
 8:     end for
 9:     for $\{i,j\} \in \mathcal{E}$ do
10: $\theta := \max\left\{1/2, 1 - \frac{(\lambda/\rho)W_{ij}}{|x^{(k+1)}[i]+u_{ij}^{(k)} - x^{(k+1)}[j] - u_{ji}^{(k)}|}\right\}$
11: $z_{ij}^{(k+1)} := \theta(x^{(k+1)}[i] + u_{ij}^{(k)}) + (1-\theta)(x^{(k+1)}[j] + u_{ji}^{(k)})$
12: $z_{ji}^{(k+1)} := (1-\theta)(x^{(k+1)}[i] + u_{ij}^{(k)}) + \theta(x^{(k+1)}[j] + u_{ji}^{(k)})$
13: $u_{ij}^{(k+1)} := u_{ij}^{(k)} + (x^{(k+1)}[i] - z_{ij}^{(k+1)})$
14: $u_{ji}^{(k+1)} := u_{ji}^{(k)} + (x^{(k+1)}[j] - z_{ji}^{(k+1)})$
15: $k := k+1$
16:     end for
17: until convergence
Output: classifier $x^{(k)}[i]$ for all $i \in \mathcal{V}$
```

IV. NUMERICAL EXPERIMENTS

We assessed the performance of lnLasso Algorithm 1 by applying it to a synthetic dataset whose empirical graph $\mathcal{G}_{\rm syn} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ is partitioned into two clusters $\mathcal{F} = \{\mathcal{C}_1, \mathcal{C}_2\}$, i.e., $\mathcal{V} = \mathcal{C}_1 \cup \mathcal{C}_2$ and $\mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset$ (cf. Figure (1)). The empirical graph of the dataset was generated by using the Barabási-Albert model, which is a scale free random graph model based on polynomial degree distribution (see [7] for more details). We tuned the model such that both clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ both contain 100 nodes. The two clusters are connected via $|\partial\mathcal{F}| = 30$ boundary (or inter-cluster) edges. The weights of edges $\{i,j\}$ connecting nodes $i,j \in \mathcal{C}_1$ within cluster $\mathcal{C}_1$ are chosen as $W_{i,j} \sim [\mathcal{N}(10,1)]_+$ and the weights for edges $\{i,j\}$ within cluster $\mathcal{C}_2$ are chosen as $W_{i,j} \sim [\mathcal{N}(12,1)]_+$. The weights of boundary edges $\{i,j\} \in \partial\mathcal{F}$ are set as $W_{i,j} \sim [\mathcal{N}(3,1)]_+$.

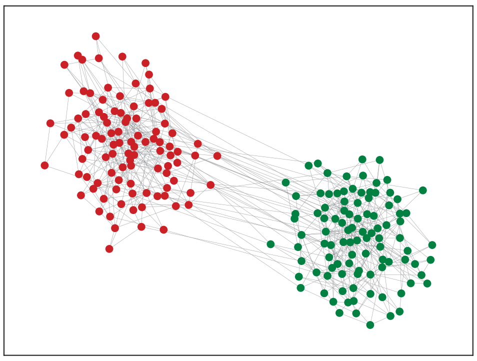

Fig. 1. The empirical graph $\mathcal{G}_{\rm syn}$ of the synthetic dataset whose data points are labelled as $y_i = 1$ (shown in red) for $i \in \mathcal{C}_1$ and $y_i = -1$ (shown in green) for $i \in \mathcal{C}_2$.

The true underlying labels $y_i = 1$ for $i \in \mathcal{C}_1$ and $y_i = -1$ for $i \in \mathcal{C}_2$ are observed only for the nodes in the sampling set $\mathcal{M}$. We constructed the sampling set $\mathcal{M}$ by selecting uniformly at random 10 different nodes. In order to learn a classifier $x[\cdot]$ from the observed labels $\{y_i\}_{i\in\mathcal{M}}$, we apply lnLasso Algorithm 1 with manually tuned parameters $\lambda = 5\cdot10^{-4}$ and $\rho = 5\cdot10^{-4}$ and a fixed number of 100 iterations. The classifier $\hat{x}[i]$ delivered by Algorithm 1 is then used to label the data points as $\hat{y}_i = 1$ if $\hat{x}[i] > 0$ and $\hat{y}_i = -1$ otherwise. The resulting labelling $\hat{y}_i$ is depicted in Figure (3). The convergence of Algorithm 1 in terms of the objective function value achieved by the sequence of iterates $x^{(k)}[i]$ is shown in Figure (4).

As can be seen by comparing the true underlying labels (cf. Figure (1)) with the recovered labels (cf. Figure (3)), lnLasso is able to perfectly recover the true underlying clustering. Moreover, as can be observed in Figure (4), perfect recovery was attained only after some 20 iterations.

V. CONCLUSIONS

We proposed a novel method for classifying network-structured datasets. This method, which we coin lnLasso, is obtained by applying the generic nLasso optimisation framework to classification problems with underling network structure. In particular, lnLasso amounts to regularized empirical risk minimization using the logistic loss to measure empirical error and TV for requiring the classifier to conform

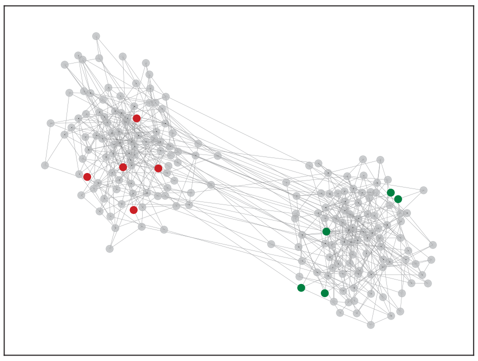

Fig. 2. The set of sampled nodes, $\mathcal{M}$.

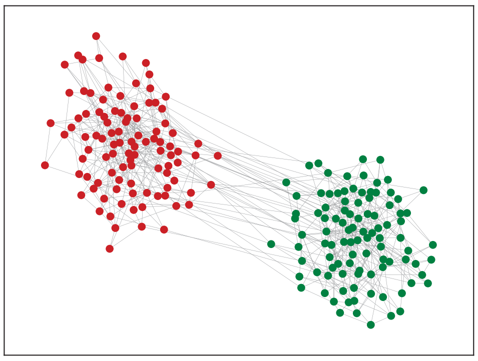

Fig. 3. The labels recovered by LNL after 100 iterations.

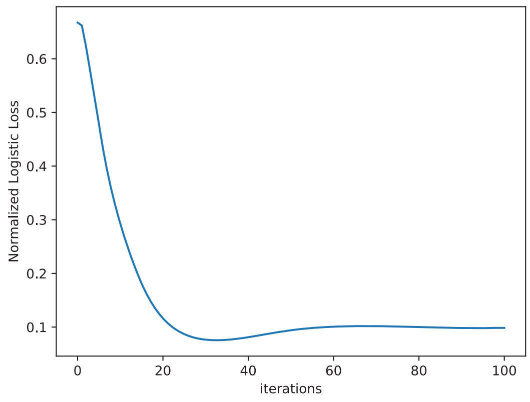

Fig. 4. Trend of logistic loss.

with the underlying network structure. A scalable implementation of lnLasso in the form of message passing over the underlying network structured is obtained via ADMM. Some illustrative numerical experiments verify the efficiency of lnLasso.

VI. REFERENCES

[1] S. Muthukrishnan S. Bhagat, C. Graham, *Social network data analytics*, Springer, 2011.

[2] A. Jung, N.T. Quang, and A. Mara, "When is network lasso accurate?," *Frontiers in Appl. Math. and Stat.*, vol. 3, pp. 28, 2018.

[3] A. Jung and M. Hulsebos, "The network nullspace property for compressed sensing of big data over networks," *Front. Appl. Math. Stat.*, Apr. 2018.

[4] D. Hallac, J. Leskovec, and S. Boyd, "Network lasso: Clustering and optimization in large graphs," in *Proc. SIGKDD*, 2015, pp. 387–396.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, vol. 3 of *Foundations and Trends in Machine Learning*, Now Publishers, Hanover, MA, 2010.

[6] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*, The MIT Press, Cambridge, Massachusetts, 2006.

[7] R. Albert and A. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74(1), no. 47, 2002.