Signal Processing 2019-2020
Date: February 2020
Project report

Signal Processing@Data Science A.Y. 2019-2020

SVM - Parallel Consensus and Sharing Optimization

Joanna Broniarek

1. Introduction

There is a social loan platform that provides loans and investment opportunities for individual and corporate clients. In recent years, the share of loans outstanding on the platform has increased significantly, which is why there is a need to develop a model supporting the underwriting process. **The idea is to increase the detection of unpaid loans before issuing / offering a loan on the platform**, so that the model result can be used as an additional contribution to the application assessment process.

The goal of a project is to solve the problem in a distributed way, with each processor handling a subset of the training data. This is useful either when there are so many training examples that it is inconvenient or impossible to process them on a single machine or when the data is naturally collected or stored in a distributed fashion.

In order to classify default/non-default clients I used the SVM classifier and implemented it in a parallel fashion. I compared both Parallel Consensus Optimization and Sharing Optimization to the Centralised version.

2. Data

Data was taken from Kaggle Competition (https://www.kaggle.com/wendykan/lending-club-loan-data).

I performed some data preprocessing and cleaning, like: removing rows with None values, transformation of categorical columns into numerical, etc. For the project I used a subset of data, which consists of .. The number of

Then, I splitted the dataset into train and test set with percentages 70% and 30% ,respectively.

3. SVM

Training data: $(q_i, p_i)\ \forall i = 1, ..., n$ Where q_i is a feature vector of the i-th example and $q_i \in R^m$. The p_i is a binary outcome class $p_i \in \{-1, 1\}$.

The optimization problem is defined by the formula:

$$\min_{w,b} \sum_{i=1}^{n} max(0, 1 - p_i(w^T q_i - b)) + \lambda ||w||^2 + \lambda b^2 \tag{1}$$

which can be rewritten as

$$\min_{x} \sum_{i=1}^{n} max(0, 1 + A\mathbf{x}) + \lambda ||\mathbf{x}||^2 \tag{2}$$

where $\mathbf{x} = [\mathbf{w}\ b]^T$, $A = [a_1^T\ a_2^T\ ...a_n^T]^T$, $a_i = [-p_i x\ -p_i]^T$.

4. Parallel Consensus Optimization with Regularization

Considering the problem of minimizing an additive function, i.e. a sum of terms that all share a common variable:

$$\text{minimize} \quad \sum_{i=1}^{n} f_i(x) + g(x)$$

with variable $x \in R^n$, the problem is to minimize each of the 'local' objective functions f_i and regularization term g, each of which depends on the same 'global' variable x. The aim is to solve this problem in a way that allows each f_i to be handled in parallel by a separate processor. Hence, a transformation of the problem into consensus form is:

$$\begin{aligned} \text{minimize} \quad & f_i(x_i) + g(z) \\ \text{subject to} \quad & x_i = z \end{aligned}$$

with variables $x \in R^n$, $i = 1, \ldots, N$. The idea is to create N copies of the original 'global' variable x so that the objective is separable, but also adding a consensus constraint that requires all these 'local' variables x_i to agree. Then, the ADMM [1] method can be applied to solve a problem.

In the case of SVM the minimization formula is:

$$\begin{aligned} \text{minimize} & \sum_{i=1}^{N} max(0, 1 + A_i x_i) + \lambda \|z\|^2 \\ \text{subject to} \quad & x_i = z, \quad i = 1, \ldots, N \end{aligned}$$

The augmented Lagrangian (in a scaled form) is :

$$L_a(\{x_i\}_{i=1}^N, z, \{u_i\}_{i=1}^N) = \sum_{i=1}^{M} l_i(A_i x_i) + \rho/2 \sum_{i=1}^{M} \|x_i - z + u_i\|_2^2 + \lambda \|z\|_2^2, \tag{3}$$

where $l_i(A_i x_i) = max(0, 1 + A_i x_i)$.

Hence, the algorithm I obtain is:

$$x_i^{k+1} = \underset{x_i}{\operatorname{argmin}}(l_i(A_i x_i) + \frac{\rho}{2}\|x_i - z^k + u_i^k\|_2^2), \quad \forall i = 1, ..., N \tag{4}$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}}(\lambda \|z\|_2^2 + \frac{N\rho}{2}\|(\bar{x}^{k+1} + \bar{u}^k) - z\|_2^2) = \frac{N\rho}{2\lambda + N\rho}(\bar{x}^{k+1} + \bar{u}^k) \tag{5}$$

$$u_i^{k+1} = u_i^k + x_i^{k+1} - z^{k+1}, \quad \forall i = 1, ..., N \tag{6}$$

Figure 1. Disagreement vs Number of Iterations

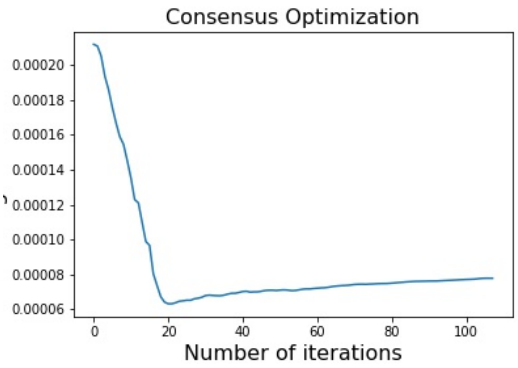
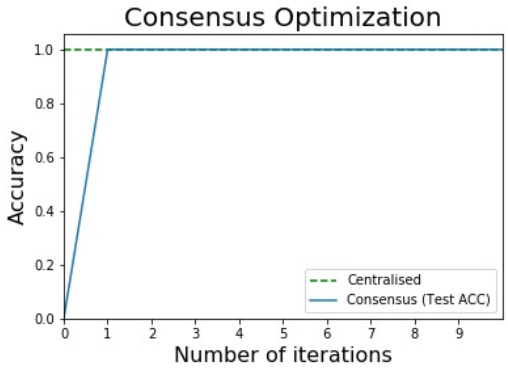


Figure 2. Accuracy vs Number of Iterations



5. Parallel Sharing Optimization

In the case when there is a very large number of features, the goal is to solve such problems in a distributed fashion with each processor handling a subset of the features.

The formulation of a sharing problem is

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(x_i) + g(\sum_{i=1}^{N} z_i) \\ \text{subject to} \quad & x_i - z_i = 0, \quad \forall i = 1, \ldots, N \end{aligned}$$

where $f_i(x_i)$ is a local cost function handled by a i-th processor and $g(\sum_{i=1}^{N} z_i)$ is a shared objective function. In the case of SVM, the minimization formula is:

$$\begin{aligned} \text{minimize} \quad & \mathbf{1}\, l(\sum_{i=1}^{N} z_i) + \lambda \|x\|^2 \\ \text{subject to} \quad & A_i x_i - z_i = 0, \quad i = 1, \ldots, N \end{aligned}$$

where $\lambda\|x\|^2 = \sum_{i=1}^{N} \lambda x_i^2$

ADMM method

The augmented Lagrangian (in a scaled form) is :

$$L_a(\{x_i\}_{i=1}^N, \{z_i\}_{i=1}^N, \{u_i\}_{i=1}^N) = \mathbf{1}\, l(\sum_{i=1}^N z_i) + \lambda\|x\|^2 + \rho/2 \sum_{i=1}^N \|A_i x_i - z_i + u_i\|_2^2. \tag{7}$$

Then, ADMM is:

$$x_i^{k+1} = \underset{x_i}{\operatorname{argmin}}(\lambda x_i^2 + \rho/2\|A_i x_i - z_i^k + u_i^k\|_2^2), \quad \forall i = 1, ..., N \tag{8}$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}}(\mathbf{1}\, l(\sum_{i=1}^N z_i) + \rho/2 \sum_{i=1}^N \|A_i x_i^{k+1} - z_i + u_i^k\|_2^2) \tag{9}$$

$$u_i^{k+1} = u_i^k + A_i x_i^{k+1} - z_i^{k+1}, \quad \forall i = 1, ..., N \tag{10}$$

The problem can be simplified by at first calculating the average $\bar{z}$.

Minimizing z-update over $z_1, \ldots, z_N$ with $\bar{z}$ fixed, has the solution:

$$z_i = A_i x_i - u_i + \bar{z} - \overline{Ax} - \bar{u}$$

Hence, z-update can be obtained by:

$$z^{k+1} = \underset{z}{\operatorname{argmin}}(\mathbf{1}\, l(N\bar{z}) + \frac{N\rho}{2}\|\overline{Ax}^{k+1} - \bar{z} + \bar{u}^k\|_2^2) \tag{11}$$

Substituting z_i^{k+1} in the u-update gives:

$$u_i^{k+1} = \bar{u}^k + \overline{Ax}^{k+1} - \bar{z}^{k+1}, \quad \forall i = 1, ..., N \tag{12}$$

which shows that the dual variables u_i^k are all equal.

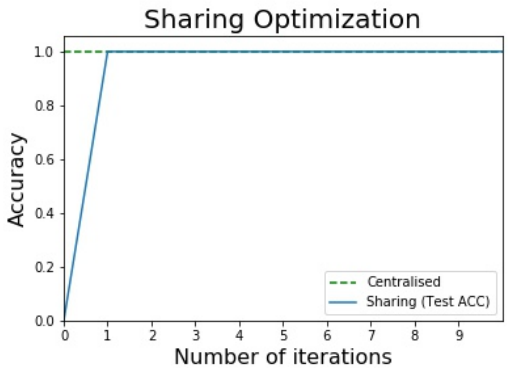
Substituting in the expression for z_i^k in the x-update, the final algorithm becomes:

$$x_i^{k+1} = \underset{x_i}{\operatorname{argmin}}(\lambda x_i^2 + \rho/2\|A_i x_i - A_i x_i^k + \overline{Ax}^k - \bar{z}^k + u^k\|_2^2) = (\frac{2\lambda}{\rho} + A_i^T A_i)^{-1}(A_i^T A_i x_i^k - A_i^T \overline{Ax}^k + A_i^T \bar{z}^k - A_i^T u^k), \quad \forall i = 1, ..., N \tag{13}$$

$$\bar{z}^{k+1} = \underset{\bar{z}}{\operatorname{argmin}}(\mathbf{1}\, l(N\bar{z}) + \frac{N\rho}{2}\|\bar{z} - (\overline{Ax}^{k+1} + u^k)\|_2^2) = \begin{cases} \bar{z}_i = v_i - 1/\rho & \text{if} \quad v_i > 1/\rho - 1/N \\ \bar{z}_i = v_i & \text{if} \quad v_i < -1/N \\ \bar{z}_i = -1/N & \text{otherwise} \end{cases} \tag{14}$$

$$u^{k+1} = u^k + \overline{Ax}^{k+1} - \bar{z}^{k+1} \tag{15}$$

Figure 3. Accuracy vs Number of Iterations



References

1Proximal Algorithms, Neal Parikh & Stephen Boyd, Stanford University