Proximal Algorithms: Study And Parallel Implementations

Harit Vishwakarma
ME Student, CSA, IISc

December 7, 2015

Outline

1. Objective
2. Brief Overview
3. Implementations

Objective

Study and Parallel Implementations of:

- ADMM
- ISTA/FISTA

Objective

Study and Parallel Implementations of:

- ADMM
- ISTA/FISTA

Using Them To Solve ML Problems

- Lasso (Regression)
- SVM (Classification)

Objective

Study and Parallel Implementations of:

- ADMM
- ISTA/FISTA

Using Them To Solve ML Problems

- Lasso (Regression)
- SVM (Classification)

Work Done

- Studied and Implemented the methods in Apache Spark.
- Sequential and Parallel Versions.

Outline

1. Objective
2. Brief Overview
   - ADMM
   - Proximal Gradient Method
3. Implementations

Outline

ADMM: Alternating Direction Method of Multipliers

Problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c, \end{array}$$

where $f, g : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\}$ are closed proper convex and f is differentiable.

Both f, g can be non-smooth.

ADMM: Alternating Direction Method of Multipliers

Problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c, \end{aligned}$$

where $f, g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ are closed proper convex and f is differentiable.
Both f, g can be non-smooth.

Update Step

$$x^{k+1} := \underset{x}{\text{argmin}}(f(x) + \frac{\rho}{2}||Ax + Bz^k - c + u^k||_2^2)$$

$$z^{k+1} := \underset{z}{\text{argmin}}(g(z) + \frac{\rho}{2}||Ax^{k+1} + Bz - c + u^k||_2^2)$$

$$u^{k+1} := u^k + x^{k+1} - z^{k+1}$$

Outline

1. Objective
2. Brief Overview
   - ADMM
   - Proximal Gradient Method
3. Implementations

Proximal Gradient Method

Problem

$$\min_x f(x) + g(x)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\}$ are closed proper convex and f is differentiable.
g can be used to encode the constraints.

Proximal Gradient Method

Problem

$$\min_x f(x) + g(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ are closed proper convex and f is differentiable.

g can be used to encode the constraints.

Update Step

$$x^{k+1} := prox_{\lambda^k g}(x^k - \lambda^k \nabla f(x^k))$$

$\lambda^k > 0$ is the step size.

Proximal Gradient Method

Problem

$$\min_{x} f(x) + g(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ are closed proper convex and f is differentiable.
g can be used to encode the constraints.

Update Step

$$x^{k+1} := prox_{\lambda^k g}(x^k - \lambda^k \nabla f(x^k))$$

$\lambda^k > 0$ is the step size.

Convergence

When ∇f is Lipschitz continuous with constant L, it converges with rate $O(1/k)$ when a fixed step size $\lambda^k = \lambda \in (0, 1/L]$ is used.

Outline

1. Objective
2. Brief Overview
3. Implementations
   - ADMM
   - ISTA/FISTA

Outline

Parallelizing ADMM: Splitting on Examples

Problem Setup

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \Sigma_{i=1}^{N} f_i(x_i) + g(z) \\ \text{subject to} & x_i - z = 0, \; i = 1 \ldots N \end{array}$$

Updates

$$x_i^{k+1} := \underset{x_i}{\text{argmin}}(f_i(x_i) + \frac{\rho}{2}||x_i - z^k + u^k||^2)$$

$$z^{k+1} := \underset{z}{\text{argmin}}(g(z) + \frac{N\rho}{2}||z - \bar{x}^{k+1} - \bar{u}^k||^2)$$

$$u_i^{k+1} := u_i^k + x_i^{k+1} - z^{k+1}$$

ADMM Splitting over Examples: Lasso

Original Lasso Problem

$$\underset{x}{\text{minimize}} \quad \frac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_1$$

Rewriting Lasso Problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2} \sum_{i=1}^{N} \|A_i x_i - b_i\|_2^2 + \lambda \|z\|_1 \\ \text{subject to} & x_i - z = 0, \; i = 1 \ldots N \end{array}$$

ADMM Splitting over Examples: Lasso

Updates

$$x_i^{k+1} := \underset{x_i}{\operatorname{argmin}}(\|A_i x_i - b_i\|_2^2 + \frac{\rho}{2}||x_i - z^k + u^k||^2)$$

$$z^{k+1} := S_{\lambda/N\rho}(\bar{x}^{k+1} + \bar{u}^k)$$

$$u_i^{k+1} := u_i^k + x_i^{k+1} - z^{k+1}$$

Procedure `par_admm_lasso()`

    while *not converged OR* $i \leq$ *max_iters* **do**

        t = mapPartitions(part_train)

        z = AllReduce t

        z = soft_threshold(λ/ρ, z)

        broadcast z

        $i = i + 1$

    end

    return z

Procedure `part_train()`

    read local x_i, u_i

    receive z from master

    $u_i = u_i + x_i - z$

    $x_i = (A_k^T A_k + \rho I)^{-1}(A_k^T b_k + \rho(z - u_i))$

    persist x_i, u_i locally

    $z_i = x_i + u_i$

    return z_i

Algorithm 1: ADMM Algorithm for Lasso on Map Reduce

Cost Analysis

Over All Complexity

- $t_i^{cm} = t_i^{cm,mw} + t_i^{cm,wm}$
- $t_i = t_i^{cp} + t_i^{cm}$
- $t_{all} = (t_i) * Iters$

Individual Costs

- Local Compute: $t_i^{cp,local} = O(n_k * d^2 + d^3)$
- Reduce Complexity: $t_i^{red} = O(d * log_2(N))$
- $t_i^{cp} = O(t_i^{cp,local} + t_i^{red})$
- Master to Workers Communication: $t_i^{cm,mw} = O(N * d)$
- Workers to Master Communication: $t_i^{cm,wm} = O(d * log_2(N))$

Measure Progress

$$r^k = x^k - z^k$$

$$s^k = \rho(z^{k+1} - z^k)$$

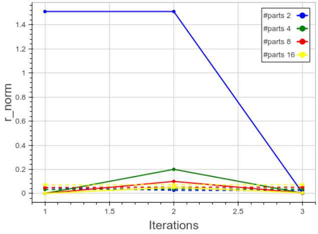


Figure : r_norm with iterations

ADMM Lasso : Convergence

Measure Progress

$$r^k = x^k - z^k$$

$$s^k = \rho(z^{k+1} - z^k)$$

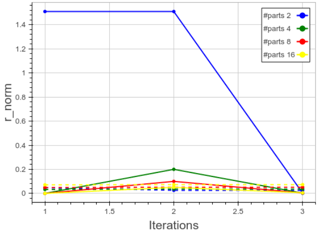


Figure : r_norm with iterations

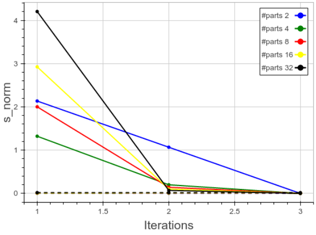


Figure : s_norm with iterations

ADMM Lasso: Scalability

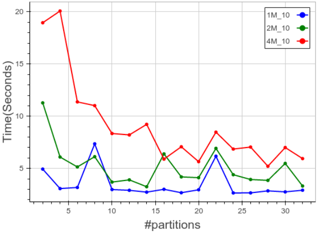


Figure : Time vs #Parts

ADMM Lasso: Scalability

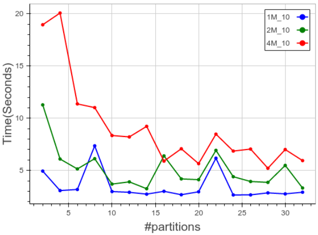


Figure : Time vs #Parts

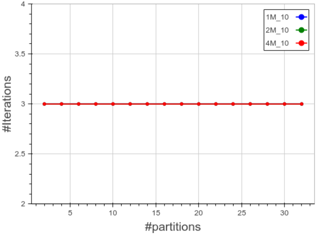


Figure : Iterations vs #Parts

ADMM Lasso: ρ variation

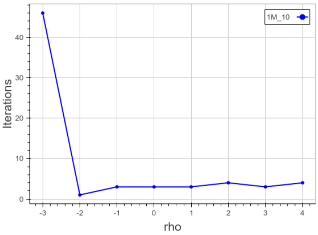


Figure : ρ vs #Iteration

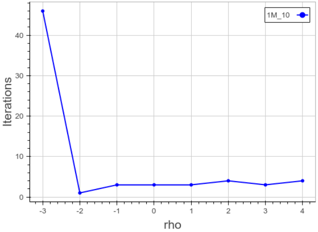


Figure : ρ vs #Iteration

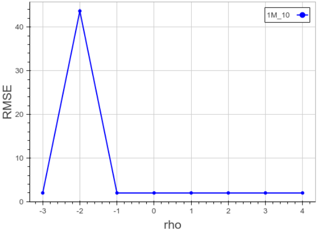


Figure : ρ vs RMSE

ADMM Splitting over Examples: SVM

Original SVM Problem

$$\underset{x}{\text{minimize}} \quad \frac{1}{n}\mathbf{1}^T(\mathbf{1} - y * (Ax))_+ + \frac{\lambda}{2}\|x\|_2^2$$

Rewriting SVM Problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{n}\sum_{i=1}^{N}\mathbf{1}_i^T(\mathbf{1}_i - y_i * (A_i x_i))_+ + \frac{\lambda}{2}\|z\|_2^2 \\ \text{subject to} \quad & x_i - z = 0, \; i = 1 \ldots N \end{aligned}$$

ADMM Splitting over Examples: SVM

Updates

$$x_i^{k+1} := \underset{x_i}{\text{argmin}}(\mathbf{1}_i^T(\mathbf{1}_i - y_i * (A_i x_i))_+ + \frac{\rho}{2}||x_i - z^k + u^k||^2)$$

$$z^{k+1} := \tfrac{N\rho}{\lambda+N\rho}(\bar{x}^{k+1} + \bar{u}^k)$$

$$u_i^{k+1} := u_i^k + x_i^{k+1} - z^{k+1}$$

Implementation Details

- Algorithm is same as Lasso.
- x_i^{k+1} is obtained using scs solver of cvxpy.

Cost Analysis

Over All Complexity

- $t_i^{cm} = t_i^{cm,mw} + t_i^{cm,wm}$
- $t_i = t_i^{cp} + t_i^{cm}$
- $t_{all} = (t_i) * Iters$

Individual Costs

- Local Compute: $t_i^{cp,local} = O(scs\_time)$
- Reduce Complexity: $t_i^{red} = O(d * log_2(N))$
- $t_i^{cp} = O(t_i^{cp,local} + t_i^{red})$
- Master to Workers Communication: $t_i^{cm,mw} = O(N * d)$
- Workers to Master Communication: $t_i^{cm,wm} = O(d * log_2(N))$

ADMM SVM: Scalability

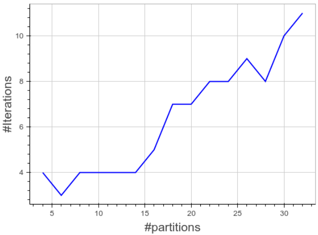


Figure : Iterations vs #Parts

ADMM SVM: Scalability

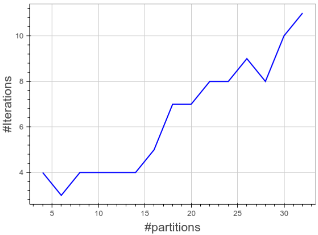


Figure : Iterations vs #Parts

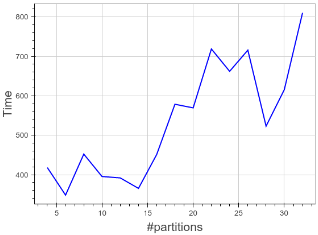


Figure : Time taken vs #parts

ADMM SVM: Scalability

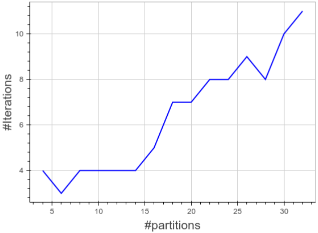


Figure : Iterations vs #Parts

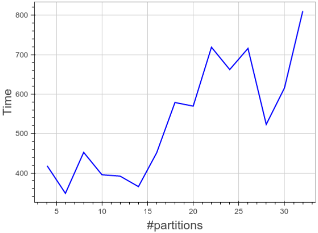


Figure : Time taken vs #parts

Accuracy

Accuracy remains same at 97.49 .

ADMM Lasso: ρ variation

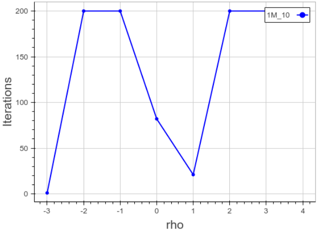


Figure : ρ vs #Iteration

ADMM Lasso: ρ variation

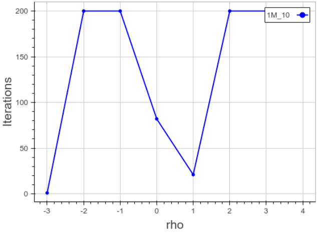


Figure : ρ vs #Iteration

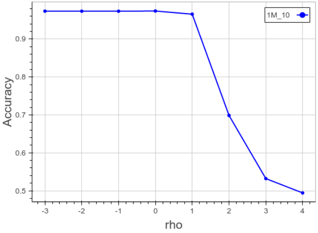


Figure : ρ vs Acc.

ADMM Splitting over Features

Problem Setup

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(\Sigma_{i=1}^N z_i - b) + \Sigma_{i=1}^N r_i(x_i) \\ \text{subject to} & A_i x_i - z_i = 0, \ i = 1 \dots N \end{array}$$

Updates

$$x_i^{k+1} := \underset{x_i}{\text{argmin}}(r_i(x_i) + \frac{\rho}{2}||A_i x_i - A_i x_i^k - \bar{z}^k + \overline{Ax}^k + u^k||^2)$$

$$\bar{z}^{k+1} := \underset{\bar{z}}{\text{argmin}}(f(N\bar{z} - b) + \frac{N\rho}{2}||\bar{z} - \overline{Ax}^{k+1} - u^k||^2)$$

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

ADMM Splitting over Features: Lasso

Rewriting Problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}\left\|(\Sigma_{i=1}^{N} z_i - b)\right\|_2^2 + \Sigma_{i=1}^{N}\lambda\left\|x_i\right\|_1 \\ \text{subject to} & A_i x_i - z_i = 0,\ i = 1\ldots N \end{array}$$

Updates

$$x_i^{k+1} := \underset{x_i}{\operatorname{argmin}}(\lambda\left\|x_i\right\|_1 + \frac{\rho}{2}||A_i x_i - A_i x_i^k - \bar{z}^k + \overline{Ax}^k + u^k||^2)$$

$$\bar{z}^{k+1} := \frac{1}{N+\rho}(b + \rho\overline{Ax}^{k+1} + \rho u^k)$$

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

Procedure `par_admm_lasso_fs()`

while *not converged OR* $i \leq$ *max_iters* **do**

- t = mapPartitions(part_train_fs)
- $\overline{Ax}^k$ = AllReduce t
- $\bar{z}^k = \frac{1}{N+\rho}(b + \rho\overline{Ax}^k + \rho u^k)$
- $u^k = u^k + \overline{Ax}^k - \bar{z}^k$
- broadcast $\overline{Ax}^k, \bar{z}^k, u^k$
- $i = i + 1$

end

return z

Procedure `part_train_fs()`

- read local x_i^k
- receive $\overline{Ax}^k, \bar{z}^k, u^k$ from master
- $x_i^k = \text{argmin}(\lambda \|x_i\|_1 + \frac{\rho}{2} \left\|A_i x_i - A_i x_i^k - \bar{z}^k + \overline{Ax}^k + u^k\right\|_2^2)$
- persist x_i^k locally
- **return** $A_i x_i$

Algorithm 2: ADMM Algorithm for Lasso on Map Reduce

Cost Analysis

Over All Complexity

- $t_i^{cm} = t_i^{cm,mw} + t_i^{cm,wm}$
- $t_i = t_i^{cp} + t_i^{cm}$
- $t_{all} = (t_i) * Iters$

Individual Costs

- Local Compute: $t_i^{cp,local} = O(cvx\_time)$
- Reduce Complexity: $t_i^{red} = O(n * log_2(N))$
- $t_i^{cp} = O(t_i^{cp,local} + t_i^{red})$
- Master to Workers Communication: $t_i^{cm,mw} = O(3 * n * N)$
- Workers to Master Communication: $t_i^{cm,wm} = O(n * log_2(N))$

Outline

1. Objective
2. Brief Overview
3. Implementations
   - ADMM
   - ISTA/FISTA

Using Proximal Gradient Method For Lasso (ISTA)

Original Lasso Problem

$$f(x) = \frac{1}{2}\|Ax - b\|_2^2 ; g(x) = \lambda \|x\|_1$$

$$\nabla f(x) = A^T(Ax - b)$$

ISTA Update

$$x^{k+1} := soft\_threshold_{\lambda^k}(x^k - \lambda^k \nabla f(x^k))$$

FISTA Update

$$x^{k+1} := soft\_threshold_{\lambda^k}(y^k - \lambda^k \nabla f(y^k))$$

$$h^{k+1} := \frac{1 + \sqrt{1 + 4(h^k)^2}}{2}$$

$$y^{k+1} := x^{k+1} + \frac{h^k - 1}{h^{k+1}}(x^{k+1} - x^k)$$

Parallelizing ISTA/FISTA

Gradient Computation in Parallel

$$\nabla f(x^k) = A^T A x^k - A^T b$$

$$A^T A = \sum_{i=1}^{N} A_i^T A_i; \qquad A^T b = \sum_{i=1}^{N} A_i^T b_i$$

Compute $A^T A$ and $A^T b$ once and keep in memory.

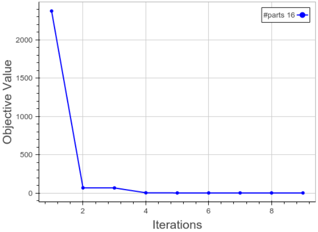
Cost Analysis

$$ReduceCost = O(d^2 * log_2 N)$$

$$Comm.cost = O(d^2 * log_2 N)$$

$$LocalCost = O((n_k * d^2) * N)$$

Where $A \in \mathbb{R}^{nxd}$,
n_k is number of rows of A in k^{th} partition
N is the total Number of partitions



#parts 16
Objective Value
2000
1500
1000
500
0
2
4
6
8
Iterations

Figure : ISTA

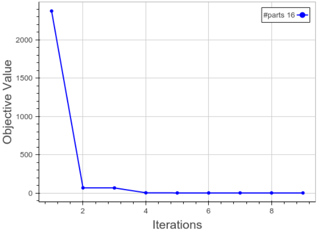


Figure : ISTA

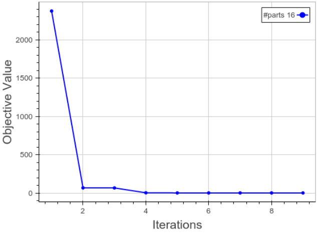


Figure : FISTA

Results: ISTA Running Time

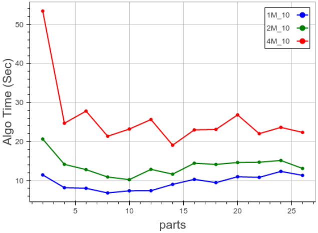


Figure : Algo. Time vs #Parts

Results: ISTA Running Time

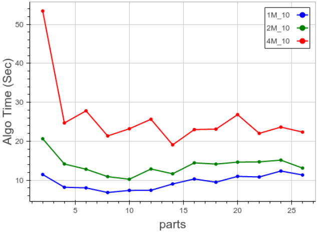


Figure : Algo. Time vs #Parts

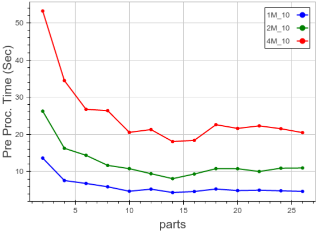


Figure : Pre. Proc. vs #parts

References

- http://web.stanford.edu/ boyd/papers/admm_distr_stats.html
- http://stanford.edu/ boyd/papers/prox_algs.html