Algorithms Associated with Factorization Machines

Yanyu Liang
Xin Lu
Xupeng Tong

Carnegie Mellon University

{yanyul,xlu2,xtong}@andrew.cmu.edu

December 13, 2016

Overview

Factorization Machines

- Factorization machines [?] models $y \in \mathbb{R}$ given $x \in \mathbb{R}^p$ using the following expression:

$$\begin{aligned}\hat{y} &= w_0 + w^T x + \sum_{i,j} (V_i x_i)^T (V_j x_j) \\ &= w_0 + w^T x + x^T V^T V x\end{aligned}$$

  , where V_i is $k \times 1$ vector.

- FMs is widely used in recommendation system, since it implicitly regularizes the model complexity by setting a k which is much smaller than p.

Other Formulation

- $\sum_{i,j}(V_i x_i)^T(V_j x_j)$ makes FMs nonconvex, and researcher has proposed convex FMs [**?**] by replacing low rank constraint by trace norm (replace $V^T V$ with W).
- Also, by replacing one V in $V^T V$ with U, [**?**] proposed generalized FMs with an online learning algorithm

Our Goal

- Propose ADMM method to sovle convexFMs with element-wise l_1 constraint
- Overview optimization algorithm to solve classic FMs problems

ConvexFMs with l_1 constraint

- l_1 penalty is widely used and it is potential helpful in many applications beyond recommendation systems.
- Consider the regression problem in convexFMs with l_1 penalty:

$$\min_{w_0, w, W} \sum_{i=1}^{n} \underbrace{(y_i - w_0 - w^T x_i - x_i^T W x_i)^2}_{f(w_0, w, W)} + \lambda_1 \|W\|_{\text{tr}} + \lambda_2 \|W\|_1 + \lambda_3 \|w\|_2^2$$

ADMM Formulation

- By introduing axilluary variable U, it can be fit into ADMM framework and the augmented Lagrangian is:

$$\mathcal{L}(w_0, w, W) = f(w_0, w, W) + \lambda_1 \|U\|_{\mathrm{tr}} + \lambda_2 \|W\|_1 + \lambda_3 \|w\|_2^2 + \langle W - U, u \rangle + m\|W - U\|_2^2$$

- Then the ADMM loop is:
  1. Update w_0, w, W:

$$w_0^k, w^k, W^k = \arg\min_{w0,w,W} f(w_0, w, W) + \lambda_2 \|W\|_1 + \frac{\rho}{2}\|W - U^{k-1} + u^{k-1}\|_2^2$$

  2. Update U:

$$U^k = \arg\min_U \lambda_1 \|U\|_{\mathrm{tr}} + \frac{\rho}{2}\|W^k - U + u^{k-1}\|_2^2$$

  3. Update u:

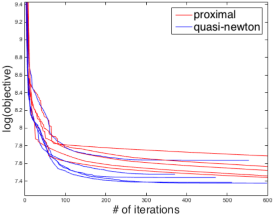
$$u^k = u^{k-1} + W^k - U^k$$

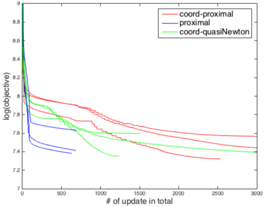
ADMM Subproblem

- The second step can be solve exactly with proximal operator of trace norm.
- We explored two approaches to solve the first subprobelm:
  - proximal graident descent on w_0, w, W (proximal)
  - blockwise coordinate descent on w_0, w and W respectively, where we applied coordinate descent on the first block and tried proximal gradient (coor-proximal) and Quasi-Newton (coor-newton) method on the second block

ADMM Results

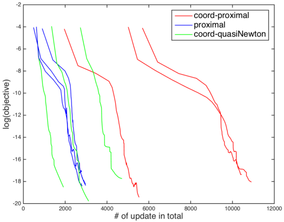
- Quasi-Newton method performs better than proximal graident descent in solving $\arg\min_W g(W)$ subproblem (as we expected)
- proximal gradient descent performs better and stable than blockwise coordinate descent

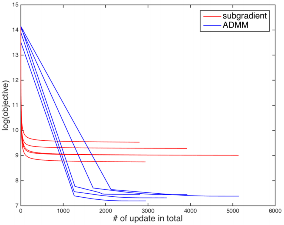




ADMM Results (con'd)

- ADMM converges and it achieves feasiblity along the path
- with the same number of updates (consider inner loop), ADMM outperforms sub-gradient method





generalized Factorization Machine formulation

gFM proposed by [**?**] removes several redundant constraints compared to the original FM, while its learning ability is kept. Reforming $\hat{y}$ in gFMs as follow:

$$\hat{y} = X^T w^* + \mathcal{A}(U^T V) + \xi$$

generalized Factorization Machine formulation con'd

- When $w = 0$, gFM is equal to the symmetric rank-one matrix sensing problem
- When $w \neq 0$, the gFM has an extra perturbation term $X^T w$.

One pass algorithm solving generalized Factorization Machine

Algorithm 1 One pass algorithm solving

Ensure: $w^{(T)}, U^{(T)}, V^{(T)}$

1: Initialize: $w^{(0)} = 0, V^{(0)} = 0. U^{(0)} = \mathrm{SVD}\left(H_1^0 - \frac{1}{2}h_2^{(0)}I, k\right)$

2: **for** $t = 1, 2, \cdots, T$ **do**

3: Retrieve $x^{(T)} = [x_{(t-1)n+1}, \cdots, x_{(t-1)n+n}]$. Define $\mathcal{A}(M) \triangleq \left[{X_i^{(t)}}^T M X_i^{(t)} \right]$

4: $\hat{U}^{(t)} = \left(H_1^{(t-1)} - \frac{1}{2}h_2^{(t-1)} I + {M^{(t-1)}}^T U^{(t-1)} \right)$

5: Orthogonalize $\hat{U}^{(t)}$ via QR decomposition: $U^{(t)} = QR(\hat{U}^{(t)})$

6: $w^{(t)} = h_3^{(t-1)} + w^{(t-1)}$

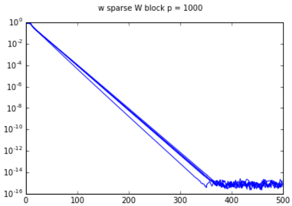
7: $V^t = (H_1^{(t-1)} - \frac{1}{2}h_2^{(t-1)} I + M^{(t-1)}) U^{(t)}$

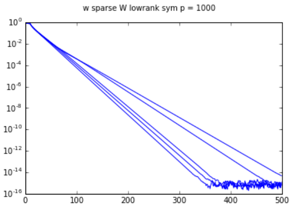
8: **end for**

9: **Output:** $w^{(T)}, U^{(T)}, V^{(T)}$

Result

- Fine tune the rank k in order to have a better convergence rate
- Higher learning rate might cause the algorithm unable to learn or even increased error rate





References I

The End