Algorithms Associated with Factorization Machines

Anonymous Author(s)
Affiliation
Address
email

1 Introduction

Factorization machines were proposed in Rendle (2010), which has been used heavily in recommendation system. FMs introduce higher-order term to model interaction between features which works reasonably well in recommendation system context, where input data is sparse but contains some low-dimensional structure, e.g. user tends to rate more movies in a specific genre. For $x \in \mathbb{R}^p$, FMs define $f : \mathbb{R}^p \to \mathbb{R}$ using the formalization shown in eq. (1).

$$\hat{y} = w_0 + w^T x + \sum_{i=1}^{p} \sum_{j=i+1}^{p} v_i^T v_j x_i x_j \tag{1}$$

$$\hat{y} = w_0 + w^T x + x^T W x \tag{2}$$

, where v_i is k-by-1 vector and intuitively every feature is mapped to a $\mathbb{R}^k$ space where the inner product between two vectors describes the strength of interaction between two features. Here, FMs have two potential drawbacks: i) k as hyperparameter of the model is needed to be chosen in practice; ii) FMs is not convex in V. From another perspective, if we let $V = (v_1, v_2, \dots, v_p)$, then $\sum_{i=1}^{p} \sum_{j=i+1}^{p} v_i^T v_j x_i x_j = x^T V V^T x - x^T \text{diag}(VV^T) x = x^T (VV^T - \text{diag}(VV^T)) x = x^T W x$. If we model x_i^2 term as well, W is equivalent to VV^T and $\text{rank}(W) = k$. Therefore, we can think of FMs as a linear model with second-order term where coefficients of second-order term are regularized by a low rank constraint (see eq. (2). With this idea, Yamada et al. (2015) proposed a convex formalization of FMs (cFMs), where they introduced trace norm to get rid of picking hyperparameter k and instead of using V they used W directly to formalize the problem, which leads the whole problem be convex. To solve cFMs problem, they proposed a coordinate descent method where they iteratively optimize w_0, w and W greedily. However, the introduction of W with trace norm regularizer makes the optimization expensive, because we need to deal with W directly, which is a p-by-p matrix. Additionally, Lin and Ye (2016) re-formed the FMs (referred as gFMs) by removing the implicit constraint that W should be positive semi-definite and W has zeros in diagonal entries. Namely, they replace $W = V^T V$ with $U^T V$. To solve gFMs, they proposed a mini-batch algorithm which guarantees to converge with $O(\epsilon)$ reconstruction error when the sampling complexity is $O(k^3 p \log(1/\epsilon))$.

2 Problem set up

The goal of the project is to explore the optimization method for FMs, cFMs, and gFMs in regression setting with squared error loss as criteria, see eq. (3),

$$L(w_0, w, V)/L(w_0, w, W) = \frac{1}{n} \sum_{i=1}^{n} (y^i - \hat{y}(x^i, w_0, w, V/W))^2 \tag{3}$$

with various smooth and non-smooth regularizations on w, V and W (depends on the formalization) in regression case. Here, we define FMs with the form in eq. (4). And we consider the following

Submitted to 30th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

regularizations on w: i) $\|w\|_2^2$; ii) $\|w\|_1$; iii) $\|\text{vec}(V)\|_2^2$.

$$\hat{y} = w_0 + w^T x + x^T V^T V x \tag{4}$$

$$\hat{y} = w_0 + w^T x + x^T U^T V x \tag{5}$$

$$\text{, where } U, V \in \mathbb{R}^{k \times p}$$

For cFMs formalization defined in eq. (2), we would like to first explore the case with trace norm penalty as stated in Yamada et al. (2015). Additionally, we would like to add sparsity constraint on interaction term, because the interaction between features should be sparity under the assumption that only features in the same genres have strong interaction and the interaction between different genres is relatively minor. Therefore, we plan to explore i) $\|W\|_{\text{tr}}$; ii) $\|\text{vec}(W)\|_1$, especially we want to explore the case where we need low rank and sparsity at the same time. For gFMs case, we plan to implement the algorithm proposed in Lin and Ye (2016) and compare its performance with others empirically.

3 Methods

3.1 Solving FMs

Rendle (2010) proposed a stochastic gradient descent method to optimize it. The gradient of every term is as follow:

$$\nabla_{w_0}\hat{y} = 1$$

$$\nabla_{w}\hat{y} = x$$

$$\nabla_{V}\hat{y} = 2Vxx^T$$

The gradient of smooth regularizer $\|w\|_2^2$ and $\|\text{vec}(V)\|_2^2$ is:

$$\nabla\|w\|_2^2 = 2w$$

$$\nabla\|\text{vec}(V)\|_2^2 = 2V$$

And the proximal operator of $\|w\|_1$ is the basic soft-thresholding function:

$$\{\text{prox}_t(x)\}_i = \begin{cases} x_i - t & \text{, if } x_i > t \\ 0 & \text{, if } x_i \in [-t, t] \\ x_i + t & \text{, if } x_i < -t \end{cases}$$

Since the only non-smooth term is $\|w\|_1$, then we can optimize the criteria $L(w_0, w, V)$ with proximal gradient descent and accelerated proximal method.

3.2 Solving cFMs

To solve cFMs in the form of:

$$\min_{w_0, w, W} L(w_0, w, W) + \lambda_1\|w\|_2^2 + \lambda_2\|W\|_{\text{tr}}$$

, we can re-form it in the following way:

$$\min_{w \in \mathbb{R}^p, W \in \mathbb{S}^{p \times p}} \sum_{i=1}^{n} 1/2(y_i - w^T x_i - x_i^T W x_i)^2 + \alpha/2\|w\|_2^2 + \beta\|W\|_{\text{tr}} \tag{6}$$

. To solve the problem, Yamada et al. (2015) proposes a two-block descent algorithm, which separates the original problem into two sub-problems:

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} 1/2(y_i - w^T x_i - \pi_i)^2 + \alpha/2\|w\|_2^2, \tag{7}$$

where $\pi_i = \langle W, x_i x_i^T \rangle$, and

$$\min_{W \in \mathbb{S}^{d*d}} \sum_{i=1}^{n} 1/2(y_i - w^T x_i - x_i^T W x_i)^2 + \beta\|W\|_{\text{tr}} \tag{8}$$

Alternatively perform standard method on eq. (7) and greedy coordinate descent on eq. (8) until convergence will get optimal solution w and W. Similarly, we also perform the two-block descent algorithm. But the difference from Yamada et al. (2015) is that, instead of greedy coordinate descent, here we solve eq. (8) using proximal gradient descent. As for solving eq. (7), we consider it as solving a standard ridge regression problem. By solving the gradient of objective function in eq. (7) equals 0, we get: $w' = (X^TX + \alpha/2 * I)^{-1}X^Ty$, where w' is $(w_0, w^T)^T$. Thus, we get our standard ridge solver with respect to w. Then we consider solve eq. (8) using proximal gradient descent. eq. (8) can be written as $g(W) + h(W)$, where $g(W) = \sum_{i=1}^{n} 1/2(y_i - w^Tx_i - x_i^TWx_i)^2$ and $h(W) = \beta\|W\|_{\text{tr}}$. Then we have

$$\nabla g(W) = -\sum_{i=1}^{n}(y_i - w^Tx_i - x_i^TWx_i)x_ix_i^T$$

.

From what we know in class, for matrix completion problem,

$$\text{prox}_t(W^+) = S_{\beta t}(W^+) = U\Sigma_{\beta t}V^T$$

where $W = U\Sigma V^T$ is a SVD, and $\Sigma_{\beta t}$ is diagonal matrix with

$$(\Sigma_{\beta t})_{ii} = \max\{\Sigma_{ii} - \beta t, 0\}$$

We then operate the backtracking line search on $g(W)$: Define

$$G_t(W) = \left(W - \text{prox}_t(W - t\nabla g(W))\right)/t$$

Then fix a parameter $0 < \beta < 1$. At each iteration, start with t = 1, and while

$$g(W - tG_t(W)) > g(W) - t\nabla g(W)^TG_t(W) + \frac{t}{2}\|G_t(W)\|_F^2$$

shrink $t = \beta t$, else performs prox gradient update

$$\text{prox}_t(W - t\nabla g(W))$$

Thus, we get our prox solver with respect to W.

Finally, perform the two-block descent algorithm: we iteratively perform the ridge solver and the prox solver until the optimal value meets our accuracy requirement.

Beyond trace norm penalty on W, the introduction of non-smooth sparsity constraint makes the whole algorithm described above fail. We plan to apply subgradient method as baseline method and test the performance of augmented Lagrange Multiplier method on this problem. Here, the optimization problem we consider is the following:

$$\min_{w_0,w,W} L(x, w_0, w, W) + \lambda_1\|w\|_2^2 + \lambda_2\|W\|_{\text{tr}} + \lambda_3\|\text{vec}(W)\|_1 \tag{9}$$

Since every term in the objective is convex, from the additive property of subgradient operator, a subgradient (let's use ∂f to denote a subgradient of f) of objective is given by the following quantities:

$$\partial_{W_{ij}}\|\text{vec}(W)\|_1 = \begin{cases} 1 & \text{, if } W_{ij} < 0 \\ -1 & \text{, if } W_{ij} > 0 \\ 0 & \text{, otherwise} \end{cases}$$

$$\partial\|W\|_{\text{tr}} = U^TV$$

, where U, V is given by $W = U^T\Sigma V$

Additionally, by combining subgradient and proximal method, we can use the following iterator to update W:

$$W^k \leftarrow \text{prox}_{\|\text{vec}(W)\|_1, t_k}(W^{k-1} - t_k(\nabla_W L^k + \partial\|W^k\|_{\text{tr}}))$$

, where t_k can be set as $\frac{1}{k}$.

Li et al. (2015) solved the problem with both trace norm and l_1 penaly by augmented Lagrange multipliers (ALM), which was used in Lin et al. (2010) to solve matrix completion problem and robust PCA. Inspired by their works, we can reformalize eq. (9) as follow:

$$\min_{w_0,w,W} L(x, w_0, w, W) + \lambda_1\|w\|_2^2 + \lambda_2\|W\|_{\text{tr}} + \lambda_3\|\text{vec}(P)\|_1 \tag{10}$$

$$\text{subject to} \quad W - P = 0 \tag{11}$$

The Lagrangian is:

$$\begin{aligned}\mathcal{L}(w_0, w, W, P, Y, \mu) &= L(x, w_0, w, W) + \lambda_1\|w\|_2^2 \\ &+ \lambda_2\|W\|_{\text{tr}} + \lambda_3\|\text{vec}(P)\|_1 + \langle Y, W - P\rangle + \frac{\mu}{2}\|\text{vec}(W - P)\|_2^2 \end{aligned} \tag{12}$$

, where Y is Lagrange multiplier. Then we can apply general ALM algorithm proposed in Lin et al. (2010) as stated in algorithm 2. Intuitively, μ^k can be seen as a parameter penalizing on the difference between W and P, which pushes the equality constraint to be satisfied as μ increases. In practice, we can increase μ^k geometrically (say with factor $\rho > 0$) and every time solving the subproblem in while loop, we use the previous solution as warm start.

Algorithm 1 ALM method solving eq. (10)

```
$\mu_0 > 0$
while not converge do
    solve $\arg\min \mathcal{L}(w_0, w, W^{k-1}, P, Y^k, \mu^k)$ for $w_0^k, w^k, P^k$
    solve $\arg\min \mathcal{L}(w_0^k, w^k, W, P^k, Y^k, \mu^k)$ for $W^k$
    $Y_{k+1} \leftarrow Y^k + \mu^k(W^k - P^k)$
    Update $\mu^k$ to $\mu^{k+1}$
end while
```

3.3 Solving gFMs

Recall that gFM proposed by Lin and Ye (2016) removes several redundant constraints compared to the original FM, while its learning ability is kept. Let's reform $\hat{y}$ in gFMs as follow:

$$\hat{y} = X^T w^* + \mathcal{A}(U^T V) + \xi \tag{13}$$

, where w^* combines w_0, w and we add an extra 1 in x for bias term. And note that $\mathcal{A}$ is the linear operator:

$$\mathcal{A} : \mathbb{R}^{p\times p} \to \mathbb{R}$$

$$\mathcal{A}(M) \triangleq [\langle A_1, M_1\rangle, \langle A_2, M\rangle, M, \cdots, \langle A_n, M\rangle]^T$$

where $A_i = x_i x_i^T$

Specifically, we plan to use the one-pass gFM algorithm proposed by the author in solving this problem.

As a mini-batch algorithm, it receives n training instances at each time then updates their parameters alternatively. To avoid the global convergence problem cast by the nonconvex learning problem with canonical gradient descent method, an estimation sequence is used instead.

4 Plan

We plan to implement the problems stated above, with different formalizations or regularization terms. After implementation, we plan to test their performance on synthetic data first. Specifically, we plan to explore the performance of alogrithm in two scenarios: i) $n > p$; ii) $n < p$. And for each case, we further explore the cases where: i) x is sparse for; ii) w is sparse; iii) V/W is sparse. For the choosing of p, we plan to use $p = 1000$, which is not computationally intensive, but is still within a practical scale of several real data sets. Based on synthetic data, we would like to do an empirical analysis of the performance of each alogrithm under differently structured data and give an empirical answer about whether advanced regularizer is necessary for factorization machine and under which scenario it is helpful or harmful.

Algorithm 2 One pass algorithm solving eq. (13)

```
Require: Mini-batch size $n$, number of total mini-batch update $T$, training instances $X = [x_1, x_2, \cdots, x_{nT}]^T, y = [y_1, y_2, \cdots, y_{nT}]^T$, desired rank $k \geq 1$
Ensure: $w^{(T)}$, $U^{(T)}$, $V^{(T)}$
Define $M^{(t)} \triangleq \left(\frac{U^{(t)}V^{(t)^T}+V^{(t)}U^{(t)^T}}{2}\right), H_1^{(t)} \triangleq \frac{1}{2n}\mathcal{A}'\left(y - \mathcal{A}(M^{(t)}) - X^{(t)^T}w^{(t)}\right), h_2^{(t)} \triangleq \frac{1}{n}\mathbf{1}^T\left(y - \mathcal{A}(M^{(t)}) - X^{(t)^T}w^{(t)}\right), h_3^{(t)} \triangleq \frac{1}{n}X^{(t)}\left(y - \mathcal{A}(M^{(t)}) - X^{(t)^T}w^{(t)}\right)$
Initialize: $w^{(0)} = 0$, $V^{(0)} = 0. U^{(0)} = \text{SVD}\left(H_1^0 - \frac{1}{2}h_2^{(0)}I, k\right)$
for $t = 1, 2, \cdots, T$ do
    Retrieve $x^{(T)} = [x_{(t-1)n+1}, \cdots, x_{(t-1)n+n}]$. Define $\mathcal{A}(M) \triangleq \left[X_i^{(t)^T}MX_i^{(t)}\right]$
    $\hat{U}^{(t)} = \left(H_1^{(t-1)} - \frac{1}{2}h_2^{(t-1)}I + M^{(t-1)^T}U^{(t-1)}\right)$
    Orthogonalize $\hat{U}^{(t)}$ via QR decomposition: $U^{(t)} = QR(\hat{U}^{(t)})$
    $w^{(t)} = h_3^{(t-1)} + w^{(t-1)}$
    $V^t = (H_1^{(t-1)} - \frac{1}{2}h_2^{(t-1)}I + M^{(t-1)})U^{(t)}$
end for
Output: $w^{(T)}$, $U^{(T)}$, $V^{(T)}$
```

Beyond testing our codes using synthetic data, we plan to apply them to real, complicated datasets like movielens with p equals to 100k, 1M, 2M etc. Furthermore, we would like to apply our algorithm in one of the ongoing Kaggle competition *Santander Product Recommendation*.

The partition of the tasks is listed below:

1. Implementation:
    - FMs: Yanyu Liang
    - cFMs: Xin Lu & Yanyu Liang
    - gFMs: Xupeng Tong
2. Data acquirment:
    - Synthetic data: Yanyu Liang & Xin Lu
    - Real data sets: Xupeng Tong
3. Empirical analysis: Xin Lu & Yanyu Liang
4. Real data testing: Xupeng Tong

References

S. Rendle, “Factorization machines,” in *2010 IEEE International Conference on Data Mining*, Dec 2010, pp. 995–1000.

M. Yamada, A. Goyal, and Y. Chang, “Convex factorization machine for regression,” *arXiv preprint arXiv:1507.01073*, 2015.

M. Lin and J. Ye, “A non-convex one-pass framework for generalized factorization machine and rank-one matrix sensing,” *arXiv preprint arXiv:1608.05995*, 2016.

J. Li, X. Chen, D. Zou, B. Gao, and W. Teng, “Conformal and low-rank sparse representation for image restoration,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 235–243.

Z. Lin, M. Chen, and Y. Ma, “The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices,” *arXiv preprint arXiv:1009.5055*, 2010.