Algorithms Associated with Factorization Machines

Yanyu Liang Xupeng Tong Xin Lu
Computational Biology Department
Carnegie Mellon University
Pittsburgh, PA 15213
`{yanyul,xtong,xlu2}@andrew.cmu.edu`

Abstract

Factorization machines (FMs) is widely used in recommendation system and it models the interaction between features by projecting $x \in \mathbb{R}^p$ onto the column space of $V \in \mathbb{R}^{k\times p}$. With the fact that $k < p$, it overcomes overfitting by limiting the second-order interaction term to be a low rank matrix. Based on FMs, researchers have proposed two extensions: i) convex FMs; ii) generalized FMs. In this report, we proposed and implemented two-block coordinate descent method with proximal gradient as subroutine for solving the original form of convex FMs proposed by [BFU15]. Beyond that, we extended convex FMs by introducing an element-wise sparsity constraint on interaction matrix, and we proposed and implemented an ADMM approach to solve this problem. Finally, we did empirical analysis on the performance of the algorithm proposed by [LY16] for solving generalized FMs.

1 Background

Factorization machines were proposed in [Ren10], 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 has a potential drawback, which is that FMs is nonconvex in V and may be stuck at local optimum. From another perspective, if we let $V = (v_1, v_2, ..., v_p)$, 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, [BFU15] proposed a convex formulation of FMs (cFMs), where instead of using V they used W directly to formalize the problem, which leads the whole problem to be convex. To solve cFMs problem, they proposed a coordinate descent method where they iteratively optimize w_0, w and W, and in updating W, they greedily added rank-1 matrix. Besides convex formulation, [LY16] reformulated the FMs by removing the implicit constraint that W should be symmetric, positive semi-definite, and W has zeros in diagonal entries, and they called the new model as generalized FMs (gFMS). 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 Our work outline

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

$$f(w_0, w, W) = \sum_{i=1}^{n} (y^i - \hat{y}^i)^2 \tag{3}$$

$$\begin{aligned} \text{, where} \quad & \hat{y} = w_0 + w^T x + x^T W x \quad \text{(cFMs)} \\ & \hat{y} = w_0 + w^T x + x^T U^T V x \quad \text{(gFMs)} \end{aligned}$$

In cFMs problem, we implemented a blockwise coordinate descent approach for solving the same objective (eq. (4)) as [BFU15] proposed. Beyond that, we introduced an additional regularizer $\|\text{vec}(W)\|_1$ to the problem (eq. (5)) and solved it using ADMM approach.

$$L(w_0, w, W) = f(w_0, w, W) + \lambda_1 \|W\|_{\text{tr}} + \lambda_3 \|w\|_2^2 \tag{4}$$

$$L(w_0, w, W) = f(w_0, w, W) + \lambda_1 \|W\|_{\text{tr}} + \lambda_2 \|\text{vec}(W)\|_1 + \lambda_3 \|w\|_2^2 \tag{5}$$

For gFMs, we applied the algorithm proposed in [LY16] and analyzed its performance empirically.

3 Methods

In this section, we describe the algorithms we used in details. Section 3.1 introduces the proximal gradient descent for solving cFMs with trace norm constraint on W. Section 3.2 introduces the ADMM approach for solving cFMs with both trace norm and element-wise l_1 norm constraints.

3.1 Solving cFMs by proximal gradient descent

To solve cFMs in the form of eq. (4), 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, [BFU15] 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 [BFU15] 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. We choose just perform gradient descent: let eq. (7) be $f_1(w)$, then $\nabla f_1(w) = -X'(y - diag(XZX') - Xw) + \alpha w$. So the gradient update is $w^+ = w - t\nabla f_1(w)$. 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 $g(W) = \sum_{i=1}^{n} 1/2(y_i - w^T x_i - x_i^T W x_i)^2$ and $h(W) = \beta \|W\|_{\text{tr}}$. Then we have

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

.

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

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

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

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

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.

3.2 Solving sparse cFMs by ADMM

Recall that the objective eq. (5) has two nonsmooth terms in W, therefore the proximal gradient descent is not trivially applicable. However, by introducing an auxiliary Z to replace one of the W in the nonsmooth terms and adding another equality constraint $W - Z = 0$, the problem can be solved using ADMM. To minimize the number of calls for trace norm proximal opterator (it requires SVD which is computationally expensive), we chose to replace $\|W\|_{\text{tr}}$ as $\|Z\|_{\text{tr}}$, and then the augmented Lagrangian multiplier is:

$$\begin{aligned}\mathcal{L}(w_0, w, W, Z, u) = f(w_0, w, W) + \lambda_1\|Z\|_{\text{tr}} + \lambda_2\|\text{vec}(W)\|_1 + \lambda_3\|w\|_2^2 \\ + \langle W - Z, u\rangle + m\|W - Z\|_F^2\end{aligned}$$

, where $u \in \mathbb{R}^{p\times p}$ is dual variable and m is parameter of the augmented term.

Then the ADMM iterations can be summarized as follow:

1. Update w_0, w, W:
$$\begin{aligned}w_0^k, w^k, W^k = \arg\min_{w0,w,W} f(w_0, w, W) + \lambda_2\|\text{vec}(W)\|_1 + \lambda_3\|w\|_2^2 \\ + \frac{\rho}{2}\|W - Z^{k-1} + u^{k-1}\|_F^2\end{aligned}$$
2. Update Z:
$$Z^k = \arg\min_Z \lambda_1\|Z\|_{\text{tr}} + \frac{\rho}{2}\|W^k - Z + u^{k-1}\|_F^2$$
3. Update u:
$$u^k = u^{k-1} + W^k - Z^k$$

The second step can be solved exactly using matrix soft-thresholding operator as described before. But there is no closed form solution for the first subproblem. To solve the first problem numerically, we applied two strategies: i) proximal gradient descent on w_0, w, W; ii) blockwise coordinate descent on w_0, w and W respectively, namely:

$$w_0, w = \arg\min_{w0,w} f(w_0, w, W) + \lambda_3\|w\|_2^2 \tag{9}$$

$$W = \arg\min_W f(w_0, w, W) + \lambda_2\|\text{vec}(W)\|_1 + \frac{\rho}{2}\|W - Z + u\|_F^2 \tag{10}$$

Furthermore, in the coordinate descent, we update w_0, w using another coordinate descent which converges quickly. But for the update of W, as the problem size is much bigger, it converges more slowly. Therefore, we tried two approches here to solve the subproblem on updating W: a) proximal gradient descent; b) quasi-Newton method (OWL-QN[AG07]). Then the overall procedure is summarized in Algorithm 1.

Algorithm 1 ADMM for solving eq. (5)

```
Input: $w_0^0, w^0, W^0, Z^0, u^0, \rho$, maxstep, tol
Output: $w_0, w^k, W$
1: for $k = 1$ : maxstep do
2:     while forward error $>$ tol do
3:         update $w_0^k, w^k, W^k$ with proximal gradient descent / blockwise coordinate descent
4:     end while
5:     $Z^k = \text{prox}_{\text{tr}, \frac{\lambda_1}{\rho}}(W^k + u^{k-1})$
6:     $u^k = W^k - Z^k$
7: end for
```

4 Results

In this section, we show the results of our algorithms running on simulated data, where p varies in $10, 30, 100, 500, 1000$. We used four types of W: i) $W = V^T V$ (SYM); ii) $W = U^T V$ (ASYM); iii) sparse W (SPARSE); iv) block diagonal W (BLOCK) shown in Figure 1.

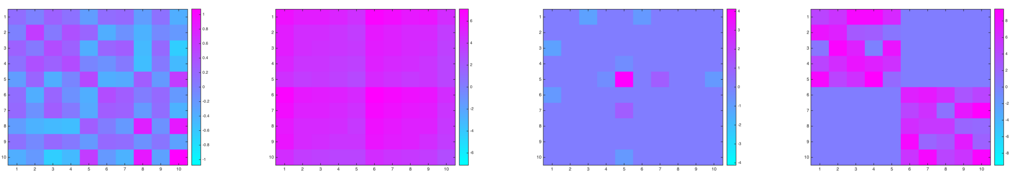

Figure 1: W matrix for four types of data (left to right: SYM, ASYM, SPARSE, BLOCK)

Section 4.1 presents the results in solving eq. (4) with method proposed in Section 3.1. Section 4.2 presents the results in implementing the methods for eq. (5) proposed in Section 3.2. Finally, we present the results of gFMs in Section 4.3.

4.1 cFMs

After several trials, based on our computing capacity, we found the proper feature dimension we are able to use for testing convexFMs is 10.

Figure 2 shows the convergence rate of four data types. The order of the convergence rate is as follows: SYM>SPARSE>ASYM≈BLOCK. Particularly, symmetric and sparse converge much more faster than asymmetric and block. Another interesting observation we want to point out is that, symmetric and sparse converge in a form we called "one stage convergence": objective value dramatically drop down at the first several iteration, then it almost don't quite drop much. As for, asymmetric and block, they perform like "two stage convergence", the first step is similar to one stage version, but there is another objective value drop down stage, but the speed is slower than the first one.

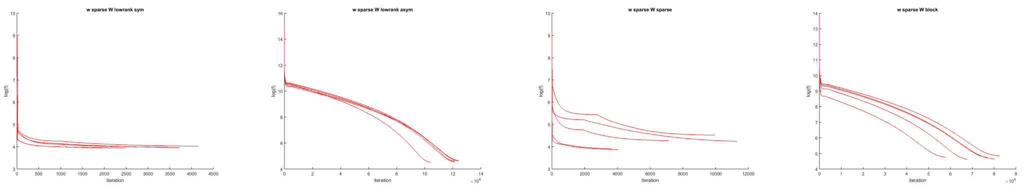

Figure 2: cFMs: The convergences of two algorithms on four types of simulated data (left to right: SYM, ASYM, SPARSE, BLOCK)

4.2 sparse cFMs

We first compare the performance of proximal gradient and quasi-Newton method on solving eq. (10). The comparison of these two approaches in terms of convergence is shown in Figure 3. From the

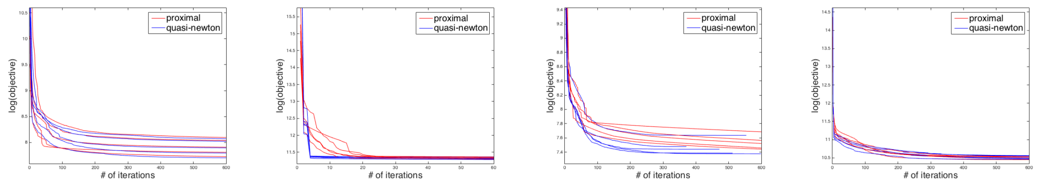

Figure 3: sparse cFMs (solve eq. (10)): The convergences of two algorithms on four types of simulated data (left to right: SYM, ASYM, SPARSE, BLOCK)

results we can see, Quasi-Newton is almost always outperforms first-order method in the sense that it can achieve the same results with fewer iterations, but the performance also depends on the property of the data. Furthermore, from the convergence curve we can see, if we stop the algorithm early, in SYM and BLOCK data, Quasi-Newton gives roughly the same result as proximal. Such results may result from the fact that in these data, the initial point is fairly far from the true optimizer.

Then we compared the performance of the following three algorithms in solving the first step in ADMM: i) proximal gradient descend on w_0, w, W as a whole (referred as proximal); ii) block-wise coordinate descent with proximal gradient descent for W and coordinate descent for w_0, w as inner loops (referred as coord-quasiNewton); iii) block-wise coordinate descent with OWL-QN update for W and coordinate descent for w_0, w as inner loops (referred as coord-proximal). Their performances were tested on the same data sets, and the convergence curves are shown in Figure 4. The percentages of W updates in the coordinate decent methods are shown in Figure 5. From the results we can see,

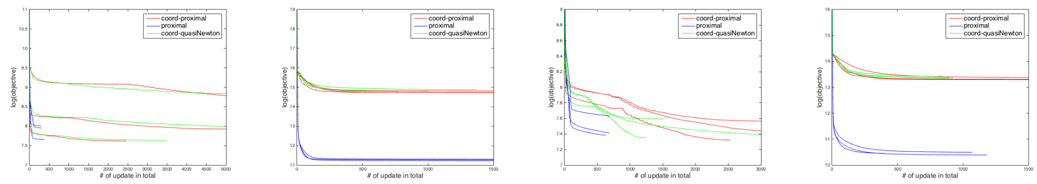

Figure 4: sparse cFMs (solve ADMM step1): The convergences of two algorithms on four types of simulated data (left to right: SYM, ASYM, SPARSE, BLOCK)

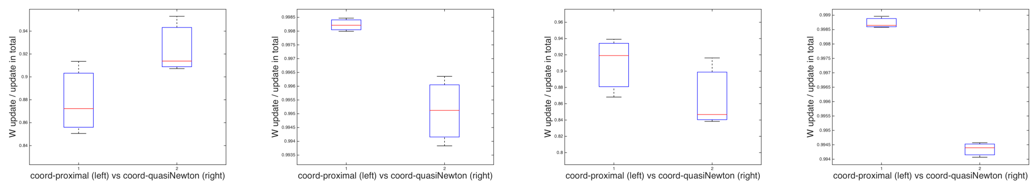

Figure 5: sparse cFMs (solve ADMM step1): The percentage of W updates in total updates for updating $w0, w, W$ (left to right: SYM, ASYM, SPARSE, BLOCK)

solving eq. (10) used most of the updates, and it is reasonable since eq. (10) is a much bigger problem comparing to solving eq. (9). Furthermore, proximal seems to be the most efficient algorithm in this problem. It converges quickly and always stops somewhere close to the optimum, whereas the blockwise coordinate descent may not converge to the right answer. The reason why coordinate descent algorithm get stuck at some point may result from the fact that inner problem is not solved inexactly. Moreover, coord-quasiNewton works than coord-proximal, which is consistent with the previous results.

By applying the above three approaches as a subroutine in ADMM, the performance of the ADMM is shown in Figure 6. The feasibility (element-wise average of $\|W - Z\|_F^2$) is shown in Figure 7.

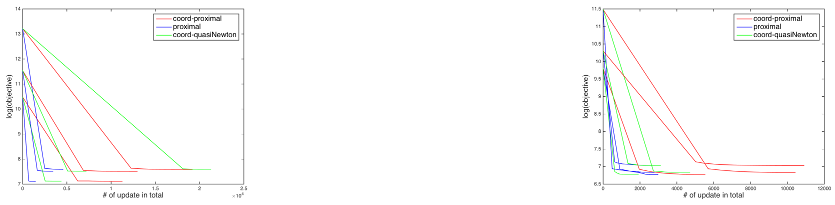

Figure 6: sparse cFMs (ADMM): The convergences of two algorithms on four types of simulated data (left to right: SYM, SPARSE)

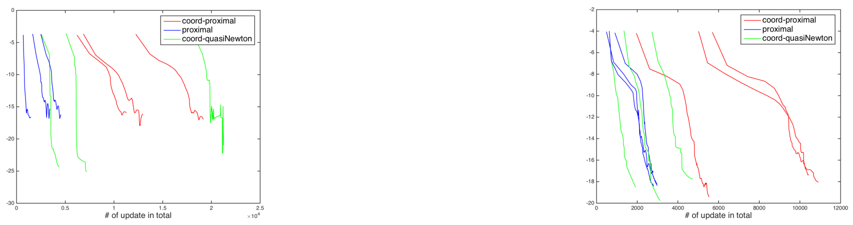

Figure 7: sparse cFMs (ADMM): The decrease of $\|Z - W\|_F^2$ along ADMM iterations (left to right: SYM, SPARSE, y-axis is $\log(\|Z - W\|_F^2/p^2)$)

From the figures we can see, ADMM (proximal) works best almost always. The feasibility curve shows that, as ADMM iteration goes, Z and W gradually match to each other.

Finally, the performance of ADMM (proximal) is compared to the subgradient method. The performance of ADMM and subgradient method on the same data set are shown in Figure 8. The

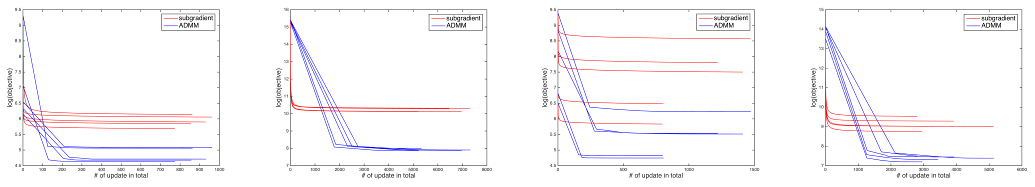

Figure 8: ADMM vs subgradient: The convergences of ADMM (proximal) and subgradient on four types of simulated data (left to right: SYM, ASYM, SPARSE, BLOCK)

comparison is made with the same amount of updades. At the very first step, ADMM needs many steps to converge, but it converges much more quickly than the sub-gradient method.

4.3 gFMs

By running the methods proposed by [LY16] (see Figure 9), we observed that when gFM is dealing with W that has a sparse structure, the error rate on the training data explodes, probably because its updating through the sequence of estimation does not necessarily require the error change in a descent manner.

Similarly, choosing the rank k of the training data as close to the original rank k in is important. As it determines the initialization of U in the algorithm described, estimating k by a large number would almost certainly cause the error rate explodes. The updating rule used in this algorithm is highly sensitive to apriori knowledge about the structure though it can't be easily estimated without certain domain knowledges.

The learning rate, as used to describe the algorithm of gFM, differs from the learning rate we know from algorithms we familiar with from the first order or second order algorithms. Instead, they use learning rate here to denote the degree a single estimation can learn. We tried setting up different learning rate and found that within a certain small window of learning rate, the error rate can decrease

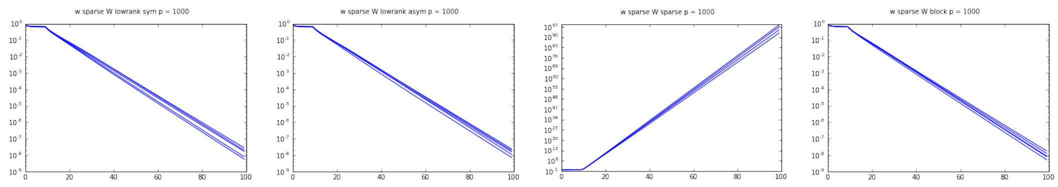

Figure 9: The convergences of two algorithms on four types of simulated data (top left to bottom right: SYM, ASYM, SPARSE, BLOCK)

linearly with a considerable rate, while with a slightly larger learning rate, linearly increased error rate can be observed.

5 Discussion

By replacing greedy coordinate descent algorithms in [BFU15] with proximal gradient descent, we must incorporate SVD, which require much more computing capacity when dealing with large scale matrix. In addition, "one and two stage convergence" and the different convergence rates of each data type still remain mysterious, which leaves a potential interesting theoretical research topic to dive into. We may discover some potential performence improvement space of proximal gradient descent for matrix completion problems.

The motivation to introduce sparsity constraint on W is that in real world application, the interaction between features can be very sparse. For example, in gene expression data, only the genes in the same pathway (regulatory module) have strong coexpression pattern and other coincident correlations should not be considered (they may lead to overfitting). Therefore, in some problem setting, the introduction of element-wise sparsity constraint is potentially useful. Furthermore, within the proposed ADMM, the element-wise sparsity constraint can be easlily extended to other more sophisticated ones, such as group lasso, since the subrountine is equivalent to a regular lasso problem.

Regarding the implementation of ADMM approach, the most time consuming step is to update w_0, w, W since there is no analytical solution. It is the bottleneck for applying the ADMM approach to large-scale problem and it is necessary to develop more sophisticated framework to solve it efficiently, such as second-order method, to use screening techniques to reduce the size of the problem before solving, to construct solution path with warm starts.

For gFMs, generally speaking, its formuation and its associated algorithm bridges the theoretical gap between the matrix sensing problem and the factorization machine. The problem it deals with is naturally non-convex which means the conventional gradient descent framework can hardly achieve a global convergence. It is proved that the algorithm they proposed can achieve a linear convergence rate and it is also shown in our experiment. However, we found it to be highly sensitive to hyperparameters such as learning rate and the rank k before applying this algorithm, which makes it somewhat impractical in a real world setting.

References

[AG07] Galen Andrew and Jianfeng Gao. Scalable training of l 1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, pages 33–40. ACM, 2007.

[BFU15] Mathieu Blondel, Akinori Fujino, and Naonori Ueda. Convex factorization machines. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 19–35. Springer, 2015.

[LY16] Ming Lin and Jieping Ye. A non-convex one-pass framework for generalized factorization machine and rank-one matrix sensing. In *Advances In Neural Information Processing Systems*, pages 1633–1641, 2016.

[Ren10] S. Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000, Dec 2010.