MASTER MVA

Research internship
April 1, 2013 - September 18, 2013
Laboratoire IMAGINE, Ecole de Ponts ParisTech

Large scale online Collaborative Filtering

Author:
Marina VINYES

Supervisor:
Guillaume OBOZINSKI

September 15, 2013

Résumé

Dans les problèmes de filtrage collaboratif nous disposons d'un ensemble de d'évaluations attribuées par des utilisateurs à des objets. L'objectif est alors de prédire les evaluations manquantes. Il s'agit d'un problème de complétion de matrice. Une approche commune consiste à supposer que la matrice des évaluations est de rang faible et de formuler le problème comme un problème d'optimisation bi-convexe, où la matrice des évaluations est approximée par le produit de deux matrices avec un faible nombre de colonnes. En pratique nous pouvons avoir des millions d'utilisateurs, des dizaines de milliers d'objets et des millions d'évaluations visibles. C'est pourquoi, les algorithmes en ligne, qui traitent une seule donnée d'apprentissage à chaque itération, sont plus adaptés. Récemment, certains algorithmes en ligne et stochastiques, comme *Stochastic Dual Coordinate Ascent* et *Stochastic Average Gradient*, se sont révélés très efficaces pour résoudre des problèmes d'apprentissages en grande dimension. Nous appliquons les idées de ces méthodes dans le contexte plus général de l'apprentissage matriciel et nous proposons un algorithme stochastique en ligne pour résoudre le problème du filtrage collaboratif.

Abstract

In collaborative filtering we have a collection of ratings that users give to items. Our goal is to predict the rating for the user-item pairs that are not in the training dataset. We can see this problem as a matrix completion problem. One popular approach is to assume that the matrix is low rank and formulate the problem as a biconvex optimization problem using matrix factorization. In practice we have millions of users, tens of thousands of items, and millions of known ratings. Hence, online algorithms, that process one single training example at each step, are more appealing. Recently several stochastic online algorithms, like *Stochastic Dual Coordinate Ascent* and *Stochastic Average Gradient*, have shown good performance to solve large scale machine learning optimization problems. We apply the main ideas of these methods to the more general context of learning with matrices (instead of feature vectors) and we propose a stochastic online algorithm for the collaborative filtering problem.

Contents

List of Figures

Learning low rank matrix models

1 Learning with matrices

1.1 Presentation

In machine learning, and more specifically in supervised learning, the prediction function is in general parametrized by a vector, i.e. $\boldsymbol{w}$. There are, however, some problems where the prediction function is parametrized by a matrix. This is the case for multi-task learning [2] and in cases where we model the prediction function as bilinear [5, 4]. The bilinear model is used in relational learning [6] and in collaborative filtering [15, 16, 8]. Thereafter we will focus on learning with matrices, and more particularly on collaborative filtering. One of the applications of collaborative filtering are recommender systems. Recommender systems provide product recommendations for the customers according to their tastes. They have become extremely common in E-commerce. Recently several algorithms have been proposed in [17] and [11] to solve the corresponding optimization problems for such large matrices using stochastic and distributed algorithms.

1.2 Collaborative filtering

Description

In collaborative filtering we have a collection of ratings $(x_{ij})_{(i,j)\in\Omega}$, where x_{ij} is the rating that a user i gave to an item j. The prediction function takes a pair (i, j) that is not in the data set and predicts the rating. We can see this problem as a matrix completion problem where the users are the rows and the items are the columns. We want to complete the matrix $\boldsymbol{X}$ based on the observed ratings. A well-known example of this problem is the **Netflix Prize**.

Netflix Challenge

Netflix is an online movie rental service where customers can rate the movies that they watch according to a 1-to-5 rating scale. Figure 1.1 shows a simplified example of the database. Question marks represent missing ratings, meaning that the customer hasn't seen the movie. Obviously, the main goal is to predict the missing ratings in order to make personalized movie recommendations.
In October 2006, the company created an open competition, known as the **Netflix Prize** [3], for the best collaborative filtering algorithm to predict user ratings for films. They offered a US1$ million prize for the best algorithm achieving at least a 10% improvement over their current prediction algorithm. For this purpose, they provided a training data set of 100,480,507 ratings that 480,189 users gave to 17,770 movies.

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	?	?	?	5	3	?
User 2	?	1	5	?	?	?
User 3	?	2	?	?	1	3
User 4	2	?	4	?	?	?
User 5	4	?	5	5	?	?

Figure 1.1: Netflix database example

2 Problem's characteristics

We will consider the collaborative filtering problem. One of the challenges is to solve the problem for very large matrices, as in practice we can have millions of users, ten of thousands of items, and millions of known ratings.

2.1 Non convex formulation

In the case of collaborative filtering it is easy to see that we can complete the matrix in many different ways : it is an ill-defined problem. A popular approach is to impose a low rank constraint in order to obtain a well-defined problem. This constrains the complexity of the model that fits the data. In other words, we seek a simply explanation fitting the observed data. A straightforward formulation of the problem can be written as

$$\begin{aligned} \min_{\boldsymbol{Y} \in \mathcal{M}^{n \times p}} & \frac{1}{N} \sum_{(i,j) \in \Omega} \ell\left(x_{ij}, y_{ij}\right) \\ \text{s.t.} \quad & \operatorname{rank}\left(\boldsymbol{Y}\right) \leq r \end{aligned} \tag{1.1}$$

where $\boldsymbol{X} = x_{ij}$ is the incomplete rating matrix, Ω is the set of user-rating pairs for which the ratings are known, N the number of known ratings, the matrix $\boldsymbol{Y} = (y_{ij})$ is the variable of the problem and ℓ is the loss function. Due to the non-convexity of the rank constraint this optimization problem is hard to solve. A classical convex relaxation of the problem is obtained by replacing the rank constrain by a convex surrogate : the trace norm $\|\boldsymbol{Y}\|_*$.

2.2 Convex relaxation with the trace norm

Instead of considering the number of non-zero singular values, i.e the rank, we consider the sum of singular values, i.e. the trace norm

$$\|\boldsymbol{Y}\|_* = \sum_{k=1}^{n} \sigma_k(\boldsymbol{Y}), \tag{1.2}$$

where $\sigma_k(Y)$ is the kth largest singular value of $\mathbf{Y}$. Since the trace norm is convex, by adding a trace norm regularization, we obtain a relaxed problem that is convex

$$\min_{\boldsymbol{Y} \in \mathcal{M}^{n \times p}} \frac{1}{N} \sum_{(i,j) \in \Omega} \ell\left(x_{ij}, y_{ij}\right) + \lambda \|\boldsymbol{Y}\|_*, \tag{1.3}$$

where x_{ij}, y_{ij} are the entries of matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$, and λ is the regularization parameter. The formulation 1.3 is convex and can be optimized easily using the Singular Value Decomposition (SVD) of $\boldsymbol{Y}$ to calculate $\| \boldsymbol{Y} \|_*$. [7] uses a proximal gradient algorithm for regularized least squares problems. However SVD has to be computed at every iteration, which is costly, especially if we deal with large matrices. Indeed, in general, the complexity of an SVD computation is $O\left(\min\left(np^2, n^2p\right)\right)$ for an $n \times p$ matrix.

2.3 Biconvex formulation

A natural way to enforce that $\boldsymbol{Y}$ is a low rank matrix is to parametrize it as $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{V}^\top$, with $\boldsymbol{U}$ and $\boldsymbol{V}$ matrices with a small number of columns. It is easy to see that when this parametrization of $\boldsymbol{Y}$ is injected in 1.1, it yield an optimization problem which is bi-convex, that is convex in $\boldsymbol{U}$ when $\boldsymbol{V}$ is fixed, and convex in $\boldsymbol{V}$ when $\boldsymbol{U}$ is fixed. The problem is however not jointly convex in $\boldsymbol{U}$ and $\boldsymbol{V}$.
It is interesting to note that if one regularizes the squared Frobenius norm of $\boldsymbol{U}$ and $\boldsymbol{V}$ and allows the number of columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ to be large, the bi-convex problem becomes equivalent to the convex problem

$$\| \boldsymbol{Y} \|_* = \inf_{\substack{\boldsymbol{U},\boldsymbol{V}:\boldsymbol{Y}=\boldsymbol{U}\boldsymbol{V}^\top \\ \boldsymbol{U}\in\mathbb{R}^{n\times k},\boldsymbol{V}\in\mathbb{R}^{p\times k} \text{ for some } k\in\mathbb{N}}} \frac{1}{2}\left(\| \boldsymbol{U} \|_{Fro}^2 + \| \boldsymbol{V} \|_{Fro}^2\right), \tag{1.4}$$

The quantity $\frac{1}{2}\left(\| \boldsymbol{U} \|_{Fro}^2 + \| \boldsymbol{V} \|_{Fro}^2\right)$ is much easier to calculate than trace norm since it is half the squared sum of all $\boldsymbol{U}$'s and $\boldsymbol{V}$'s entries. Finally we can propose the following bi-convex formulation :

$$\min_{\boldsymbol{U}\in\mathcal{M}^{n\times k},\boldsymbol{V}\in\mathcal{M}^{p\times k}} \frac{1}{N}\sum_{(i,j)\in\Omega} \ell\left(x_{ij}, \boldsymbol{u}_i\boldsymbol{v}_j^\top\right) + \frac{\lambda}{2}\left(\| \boldsymbol{U} \|_{Fro}^2 + \| \boldsymbol{V} \|_{Fro}^2\right) \tag{1.5}$$

where $\boldsymbol{u}_i$ is the ith row of $\boldsymbol{U}$, $\boldsymbol{v}_j$ is the jth row of $\boldsymbol{V}$. The main point of this formulation is that we don't have to deal with the trace norm, which avoids us computing SVD. However we pay the price of convexity. The problem becomes biconvex so we can get stuck in a local minima. One popular approach to solve this optimization problem is *Block Coordinate Descent* (BCD) as described in [7]. The link between the convex formulation with trace norm regularization and the biconvex formulation is studied in [1]. They show that under some regularity conditions, when the number of columns k is sufficiently large the two problems are equivalent, in the sense that the biconvex problem has no local minimum and its global minimum corresponds to the global minimum of the convex formulation. The link between trace norm and Frobenius norm is used in [18] to propose a boosting approach for learning problems with trace norm regularization.

Online learning

1 Regularized Empirical Risk Minimization

In supervised learning we are given a training data set $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_n, y_n) \in \mathbb{R}^p \times \mathbb{R}$, where $\boldsymbol{x}_i$ are the inputs and y_i the outputs. Our goal is to predict the output y for a new input $\boldsymbol{x}$ not present in the training dataset. This comes down to finding a prediction function f that associates an output to every input. To do so we introduce a model for the prediction function, typically a linear model, $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$, where $\boldsymbol{w} \in \mathbb{R}^p$ is the parameter vector that we want to *learn* from the training data set. We introduce a loss function $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ that models the cost $\ell(y, y')$ of predicting y' instead of y. For example, we can choose the mean square loss, $\ell(y, y') = (y - y')^2$. The quality of a prediction function f is given by the risk $\mathcal{R}(f) = \mathbb{E}[\ell(Y, f(X))]$, where the expectation is with respect to the joint probability distribution of couples (X, Y). Ideally we want to find $\boldsymbol{w}$ that minimizes this risk. Unfortunately we don't know the joint probability distribution of pairs (X, Y). Since we have a set of realizations $(\boldsymbol{x}_i, y_i)$ we minimize instead the regularized empirical risk, to find a good estimate of $\mathbf{w}$.

$$\min_{\boldsymbol{w} \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \phi_i \left(\boldsymbol{w}^\top \boldsymbol{x}_i \right) + \frac{\lambda}{2} \| \boldsymbol{w} \|^2 \tag{2.1}$$

where we add an ℓ_2-regularization for the feature vector $\boldsymbol{w}$, with regularization parameter λ, and where $\phi_i(y) = \ell(y_i, y)$. in following chapters we will always consider convex and smooth function ϕ_i.

2 Motivation

In the general context described in section 1, where we want learn a parameter vector, *Stochastic Gradient* methods (Robbins and Monro, 1951), are classical algorithms for solving such large scale supervised machine learning optimization problems. When we have a large number of training examples and we are seeking a simple explanation, we expect some redundancy in these examples. *Stochastic Gradient* methods take advantage of this characteristic using online learning. In online learning, a random training example is drawn at each step and the parameter is updated based on that example only, so each iteration cost is low. This allows us to process a large amount of training examples. On the contrary, batch algorithms methods use the whole set of training examples at each iteration to optimize the cost function. This is the case for *Full Gradient* methods, where we have to process all data at each iteration. Such methods need smaller number of iterations to converge but the iteration cost scales with number of training examples in the set which can be prohibitive when this number is large.

Recently, the interest for stochastic gradient algorithms has been revived and several algorithms based on online learning, such as the *Stochastic Average Gradient Method* in [12] and the

Stochastic Dual Coordinate Ascent in [14], have shown good performance. We want to adapt these new methods to build online stochastic algorithms to solve large scale collaborative filtering problem efficiently. We first describe the main ideas behind this methods.

3 Stochastic Average Gradient Method (SAG)

The SAG method introduced in [12] combines the low iteration cost of the *Stochastic Gradient* with the linear convergence rate of *Full Gradient*. As an example we consider an ℓ_2-regularized empirical risk minimization problem

$$\min_{\boldsymbol{w}\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n} \phi_i\left(\boldsymbol{w}^\top \boldsymbol{x}_i\right) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \tag{2.2}$$

where $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^p \times \mathbb{R}$ are the training examples and ϕ_i is the cost function associated to the ith training example. The standard *Stochastic Gradient* iteration for this problem is

$$\boldsymbol{w}^{t+1} = (1-\lambda\tau_k)\,\boldsymbol{w}^t - \tau_k \phi'_{i_t}\left(\boldsymbol{w}^\top \boldsymbol{x}_{i_t}\right)\boldsymbol{x}_{i_t} \tag{2.3}$$

where τ_k is the step-size and i_t is the selected training example. The *Full gradient* iteration is given by

$$\boldsymbol{w}^{t+1} = (1-\lambda\tau_k)\,\boldsymbol{w}^t - \frac{\tau_k}{n}\sum_{i=1}^{n} \phi'_i\left(\boldsymbol{w}^\top \boldsymbol{x}_i\right)\boldsymbol{x}_i \tag{2.4}$$

SAG uses a gradient that depends on each training example but at each step this gradient is updated with respect to a single example.

$$\boldsymbol{w}^{t+1} = (1-\lambda\tau_k)\,\boldsymbol{w}^t - \frac{\tau_k}{n}\sum_{i=1}^{n} y_i^t \quad \text{where} \quad y_i^t = \begin{cases} \phi'_{i_t}\left(\boldsymbol{w}^\top \boldsymbol{x}_{i_t}\right)\boldsymbol{x}_{i_t} & \text{if } \; i = i_t \\ y_i^{t-1} & \text{otherwise} \end{cases} \tag{2.5}$$

4 Stochastic Dual Coordinate Ascent (SDCA)

4.1 Dual decomposition

When calculating the dual of a problem like 2.2 we note that a dual variable will be associated to each training example and we will have n separate dual problems, with n the number of training examples.

4.2 SDCA algorithm

The main idea in [14] is to take advantage of the dual decomposition, as in *Dual Coordinate Ascent* (DCA) and consider a stochastic version of it. In a dual problem we will have a dual variable associated to each training example. This encourages an online algorithm, where at each iteration we process a single training example by optimizing the dual objective with respect to its associated dual variable. More concretely, a dual problem of 2.2 is

$$\max_{\alpha\in\mathbb{R}^p} -\frac{1}{n}\sum_{i=1}^{n} \phi_i^*\left(-\alpha_i\right) - \frac{\lambda}{2}\left\|\frac{1}{n\lambda}\sum_{i=1}^{n}\alpha_i \boldsymbol{x}_i\right\|^2 \tag{2.6}$$

where ϕ_i^* is the Fenchel conjugate of ϕ_i defined as $\phi_i^*(u) = \max_z (zu - \phi_i(z))$. Since our problem is convex, Karush-Kuhn-Tucker conditions say that if α^* is an optimal solution of 2.6, then

$$\boldsymbol{w}^* = \frac{1}{\lambda n} \sum_{i=1}^{n} \alpha_i^* \boldsymbol{x}_i \tag{2.7}$$

is an optimal solution of 2.2. The main point of 2.7 is that each training example is associated with its single dual variable. At each iteration of SDCA, one training example i_t is selected at random, and the dual objective is optimized with respect to its corresponding dual variable α_{i_t}. Then the primal variable is updated. SDCA method uses iterations of the form

$$\alpha_{i_t}^{(t+1)} = \alpha_{i_t}^{(t)} + \Delta\alpha_{i_t} \tag{2.8}$$

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \frac{1}{n\lambda} \Delta\alpha_{i_t} \boldsymbol{x}_{i_t} \tag{2.9}$$

where i_t is selected uniformly among the set $\{1, ..., n\}$ and $\Delta\alpha_{i_t}$ maximizes $-\frac{1}{n}\phi_{i_t}^* (-\alpha_{i_t} - \Delta\alpha_{i_t}) - \frac{1}{\lambda} \left\| \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i + \Delta\alpha_{i_t} \boldsymbol{x}_{i_t} \right\|^2$. A duality gap can be calculated and used as a stopping criterion.

5 Alternating Direction Method of Multipliers (ADMM)

ADMM, presented in [13], combines the advantages of dual decomposition and the strong convergence guarantees of the Method of Multipliers (MM). MM add an additional term to the unconstrained objective

$$\begin{aligned} &\min_{\boldsymbol{w}, \boldsymbol{z}_i \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \phi_i \left(\boldsymbol{z}_i^\top \boldsymbol{x}_i \right) + \frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \frac{\rho}{2} \sum_{i=1}^{n} \|\boldsymbol{w} - \boldsymbol{z}_i\|^2 \\ &\text{s.t} \quad \boldsymbol{z}_i = \boldsymbol{w} \quad \forall i \in \{1, ..., n\} \end{aligned} \tag{2.10}$$

in order to obtain an augmented Lagrangian $\mathcal{L}_\rho$,

$$\mathcal{L}_\rho (\boldsymbol{w}, \boldsymbol{z}_i, \boldsymbol{y}) = \frac{1}{n} \sum_{i=1}^{n} \phi_i \left(\boldsymbol{z}_i^\top \boldsymbol{x}_i \right) + \frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \sum_{i=1}^{n} \boldsymbol{y}_i^\top (\boldsymbol{w} - \boldsymbol{z}_i) + \frac{\rho}{2} \sum_{i=1}^{n} \|\boldsymbol{w} - \boldsymbol{z}_i\|^2 \tag{2.11}$$

where $\rho > 0$ is called the *penalty parameter*. We note that $\mathcal{L}_0$ is the Lagrangian of the standard problem. The benefit of including the penalty term is that strict convexity or finiteness of the loss functions ϕ_i is no longer required to converge. ADMM takes advantage of this convergence properties and also exploit the dual decomposability by performing the iterations

$$\boldsymbol{z}_i^{(t+1)} = \arg\min \left(\frac{1}{n} \phi_i \left(\boldsymbol{z}_i^\top \boldsymbol{x}_i \right) + \boldsymbol{y}_i^{(t)\top} \left(\boldsymbol{w}^{(t)} - \boldsymbol{z}_i \right) + \frac{\rho}{2} \left\| \boldsymbol{w}^{(t)} - \boldsymbol{z}_i \right\|^2 \right) \tag{2.12}$$

$$\boldsymbol{w}^{(t+1)} = \arg\min \left(\frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \sum_{i=1}^{n} \boldsymbol{y}_i^{(t)\top} \boldsymbol{w} + \frac{\rho}{2} \left\| \boldsymbol{w} - \boldsymbol{z}_i^{(t+1)} \right\|^2 \right) \tag{2.13}$$

$$\boldsymbol{y}_i^{(t+1)} = \boldsymbol{y}_i^{(t+1)} + \rho \left(\boldsymbol{z}_i^{(t+1)} - \boldsymbol{w}^{(t+1)} \right) \tag{2.14}$$

We note that 2.12 can be computed in parallel for each i.

Online matrix learning for matrix completion

In this chapter we present our approach to adapt stochastic online learning algorithms to the collaborative problem. As discussed in first chapter, the low rank constraint or its convex relaxation are difficult or computationally expensive to accomodate. It appears that biconvex formulation leads to alternate optimization algorithms with good results in literature. Furthermore this formulation is better adapted to online approaches. From now on we will consider the biconvex formulation

$$\min_{\boldsymbol{U}\in\mathcal{M}^{n\times k},\boldsymbol{V}\in\mathcal{M}^{p\times k}} P\left(\boldsymbol{U},\boldsymbol{V}\right)=\frac{1}{N}\sum_{(i,j)\in\Omega}\phi_{ij}\left(\boldsymbol{u}_i\boldsymbol{v}_j^\top\right)+\frac{\lambda}{2}\left(\|\boldsymbol{U}\|_{Fro}^2+\|\boldsymbol{V}\|_{Fro}^2\right) \tag{3.1}$$

where $\boldsymbol{u}_i$ is the ith row of $\boldsymbol{U}$, $\boldsymbol{v}_j$ is the jth row of $\boldsymbol{V}$, $\phi_{ij}(u)=\ell\left(\boldsymbol{X}_{i,j},u\right)$ is the cost function associated to the training rating (i,j), and N is the number of visible ratings. We can use an alternating optimization approach, solving alternatively the convex problem 3.1 where the variable $\boldsymbol{V}$ (resp. $\boldsymbol{U}$) is fixed, noted $P_{\boldsymbol{V}}$ (resp. $P_{\boldsymbol{U}}$). As the problems $P_{\boldsymbol{V}}$ and $P_{\boldsymbol{U}}$ are symmetrical we can focus on $P_{\boldsymbol{V}}$. We also notice that, when we fix $\boldsymbol{V}$ the minimisation can be decomposed over $\boldsymbol{U}$'s rows. If we define $P_{\boldsymbol{V},i}$ the problem $P_{\boldsymbol{V}}$ where we fix all the rows of $\boldsymbol{U}$ but the ith

$$\min_{\boldsymbol{u}_i\in\mathbb{R}^k} P_{\boldsymbol{V},i}\left(\boldsymbol{u}_i\right)=\frac{1}{N}\sum_{j\in\Omega_i}\phi_{ij}\left(\boldsymbol{u}_i\boldsymbol{v}_j^\top\right)+\frac{\lambda}{2}\|\boldsymbol{u}_i\|^2, \tag{3.2}$$

where Ω_i is the set of movies j such that the $(user,movie)$-rating (i,j) is visible. To minimize $P_{\boldsymbol{V}}$, we can minimize $P_{\boldsymbol{V},i}$'s problems independently for different $\boldsymbol{U}$'s rows.

1 SDCA approach

1.1 Formulation

We apply SDCA to each $P_{\boldsymbol{V},i}$ problem. First of all, we need to derive a dual, and to do so we add a constraint to obtain the primal problem

$$\begin{aligned}\min_{\boldsymbol{u}_i\in\mathbb{R}^k,(w_{ij})_{j\in\Omega_i}\in\mathbb{R}} & \frac{1}{N}\sum_{j\in\Omega_i}\phi_{ij}\left(w_{ij}\right)+\frac{\lambda}{2}\|\boldsymbol{u}_i\|^2 \\ \text{s.t} \quad & w_{ij}=\boldsymbol{u}_i\boldsymbol{v}_j^\top \quad \forall j\in\Omega_i\end{aligned} \tag{3.3}$$

and we derive a dual of this problem,

$$\max_{(\alpha_{ij})_{j\in\Omega_i}\in\mathbb{R}} -\frac{1}{N}\sum_{j\in\Omega_i}\phi_{ij}^*\left(-\alpha_{ij}\right) - \frac{\lambda}{2}\left\|\frac{1}{N\lambda}\sum_{j\in\Omega_i}\alpha_{ij}\boldsymbol{v}_j\right\|^2, \tag{3.4}$$

We obtain the same formulation as in SDCA but this time we have a collection of problems to solve. When optimizing main the problem 3.1 with respect to $\boldsymbol{U}$, we have n independent problems to optimize. Due to the bi-convexity, an optimal solution of primal 3.3 always depends on matrix $\boldsymbol{V}$. Indeed the condition $\frac{\partial\mathcal{L}}{\partial\boldsymbol{u}_i}=0$, where $\mathcal{L}$ is the Lagrangian of the primal problem 3.3, say that if we define

$$\boldsymbol{u}_i(\alpha) = \frac{1}{N\lambda}\sum_{j\in\Omega_i}\alpha_{ij}\boldsymbol{v}_j \tag{3.5}$$

then if α^* is an optimal solution of the dual problem, $\boldsymbol{u}_i(\alpha^*)$ is an optimal solution of the primal problem.

1.2 Algorithm

An iteration of our SDCA algorithm consist of updating a certain α_{ij} where (i,j) is drawn randomly from the training set. Then we update row $\boldsymbol{u}_i$. If we define $(\beta_{ij})_{(i,j)\in\Omega}$ to be the dual variables for the corresponding dual problem where $\boldsymbol{U}$ is fixed the algorithm consists of updating sequentially several α_{ij} and their corresponding rows $\boldsymbol{u}_i$ then several β_{ij} and their corresponding rows $\boldsymbol{v}_j$. At each step α_{ij} and β_{ij} are updated by $\Delta\alpha_{ij}$ and $\Delta\beta_{ij}$ respectively, with

$$\Delta\alpha_{ij} = \frac{\left(x_{ij} - \alpha_{ij} - \frac{1}{N\lambda}\left\langle\sum_{j'\in\Omega_i}\alpha_{ij'}\boldsymbol{v}_{j'},\boldsymbol{v}_j\right\rangle\right)}{1+\frac{\|\boldsymbol{v}_j\|^2}{N\lambda}} \tag{3.6}$$

$$\Delta\beta_{ij} = \frac{\left(x_{ij} - \beta_{ij} - \frac{1}{N\lambda}\left\langle\sum_{i'\in\Omega_j}\alpha_{i'j}\boldsymbol{u}_{i'},\boldsymbol{u}_i\right\rangle\right)}{1+\frac{\|\boldsymbol{u}_j\|^2}{N\lambda}} \tag{3.7}$$

2 ADMM aproach

For ADMM, we consider the optimization problem :

$$\begin{aligned}\min_{\boldsymbol{u}_i\in\mathbb{R}^k,(w_{ij})_{j\in\Omega_i}\in\mathbb{R}} & \frac{1}{N}\sum_{j\in\Omega_i}\phi_{ij}\left(\left\langle\boldsymbol{z}_i^j,\boldsymbol{v}_j\right\rangle\right) + \frac{\lambda}{2}\|\boldsymbol{u}_i\|^2 + \rho_i\sum_{j\in\Omega_i}\left\|\boldsymbol{z}_i^j-\boldsymbol{u}_i\right\|^2 \\ \text{s.t} \quad & \boldsymbol{z}_i^j = \boldsymbol{u}_i \quad \forall j\in\Omega_i\end{aligned} \tag{3.8}$$

Then, for each sub-problem $P_{\boldsymbol{V},i}$ we have $|\Omega_i|$ copies of $\boldsymbol{u}_i$. This algorithm uses more memory than SDCA. The iterations are of the form

$$\boldsymbol{z}_i^{j(t+1)} = \arg\min\left(\frac{1}{N}\phi_{ij}\left(\boldsymbol{z}_i^{j\top}\boldsymbol{v}_j\right) + \boldsymbol{y}_i^{j(t)\top}\left(\boldsymbol{u}_i^{(t)}-\boldsymbol{z}_i^j\right) + \frac{\rho_i}{2}\left\|\boldsymbol{u}_i^{(t)}-\boldsymbol{z}_i\right\|^2\right) \tag{3.9}$$

$$\boldsymbol{u}_i^{(t+1)} = \arg\min\left(\frac{\lambda}{2}\|\boldsymbol{u}_i\|^2 + \sum_{j\in\Omega_i}\boldsymbol{y}_i^{j(t)\top}\boldsymbol{u}_i + \frac{\rho_i}{2}\sum_{j\in\Omega_i}\left\|\boldsymbol{u}_i-\boldsymbol{z}_i^{j(t+1)}\right\|^2\right) \tag{3.10}$$

$$\boldsymbol{y}_i^{j(t+1)} = \boldsymbol{y}_i^{j(t+1)} + \rho_i\left(\boldsymbol{z}_i^{j(t+1)} - \boldsymbol{u}_i^{(t+1)}\right) \tag{3.11}$$

Results

1 Tests with generated data

For the tests we consider the quadratic loss $\phi_{ij}(u) = \frac{1}{2}(x_{ij} - u)^2$. We generate $n \times n$ random matrices of rank k, and we reveal $s\%$ of the entries.

1.1 Behaviour of SDCA

Figure 4.1 and 4.3 show results for $n = 100$, $k = 3$, $s = 50$ and $\lambda = 0.015$. We achieve an ℓ_2 error $\|\boldsymbol{X}_0 - \boldsymbol{U}\,\boldsymbol{V}\|^2 / \|\boldsymbol{X}_0\|^2$ of 7×10^{-4}, were $\boldsymbol{X}_0$ is the generated matrix and $\boldsymbol{U}\,\boldsymbol{V}$ our approximation. Here we have 5000 visible entries. Since we perform 5×10^4 iterations, each training example is visited 10 times in average. Figure 4.1 show primal and dual objectives in function of the number of iterations.

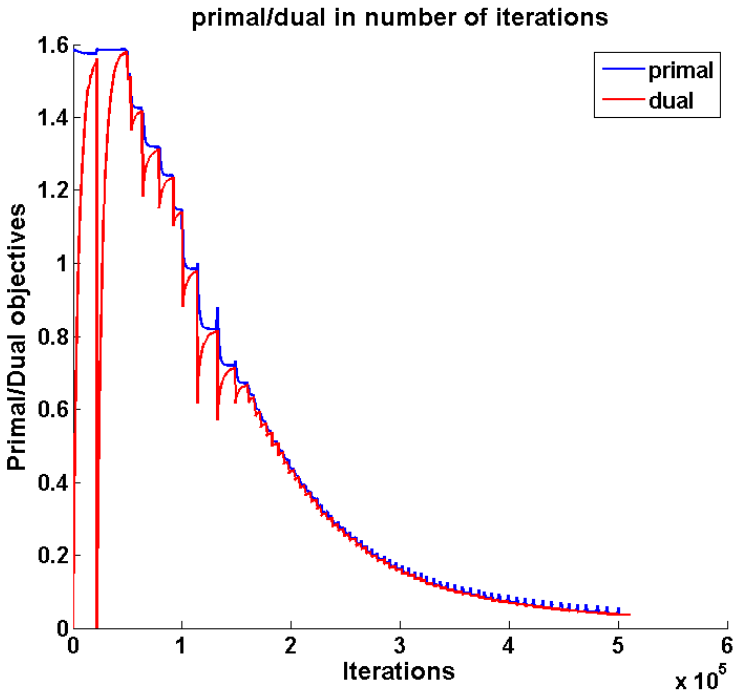

Figure 4.1: results for $n = 100$, $k = 3$, $s = 50$ and $\lambda = 0.015$: primal/dual objectives for SDCA

Figure 4.2 zooms on Figure 4.1 and we clearly see the alternate minimization. We iteratively alternate a cycle where we optimize on $\boldsymbol{U}$, then a cycle where we optimize on $\boldsymbol{V}$. In each cycle we consider a different dual. The evaluated primal objective stays the same during the

algorithm whereas a different dual is considered at each cycle. Since we perform a dual ascent, in a cycle, the dual objective increases at each iteration. This leads to a decrease, non necessarily monotonic, of the primal objective.

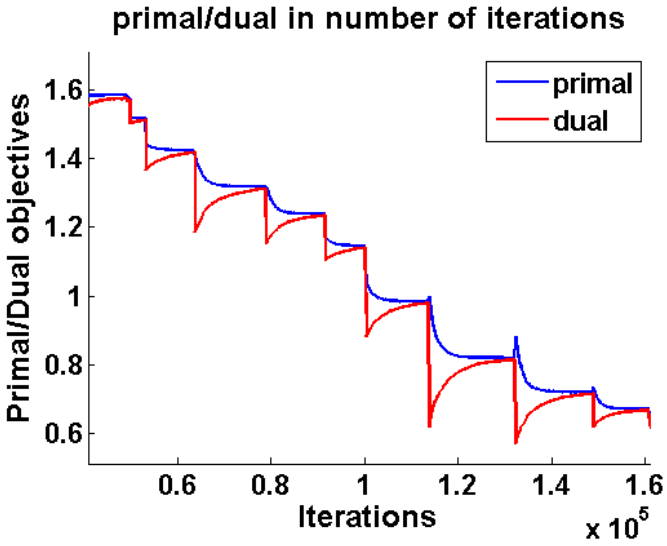

Figure 4.2: Zoom of Figure 4.1

The log-duality gap in function of the number of iterations is illustrated in Figure 4.3, showing that for each sub-problem we have a linear convergence rate.

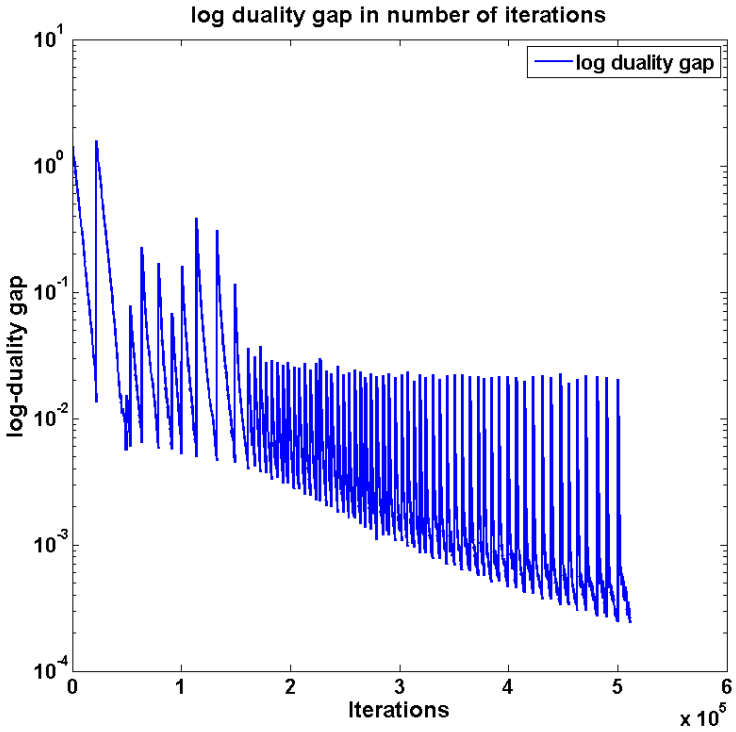

Figure 4.3: results for $n = 100$, $k = 3$, $s = 50$ and $\lambda = 0.015$:log-duality gaps for SDCA

Figure 4.4 shows primal and dual objectives for SDCA for a small value of λ, i.e. 0.0001. We notice the primal objective have some jumps when we alternate problems $P_{\boldsymbol{V}}$ and $P_{\boldsymbol{U}}$.

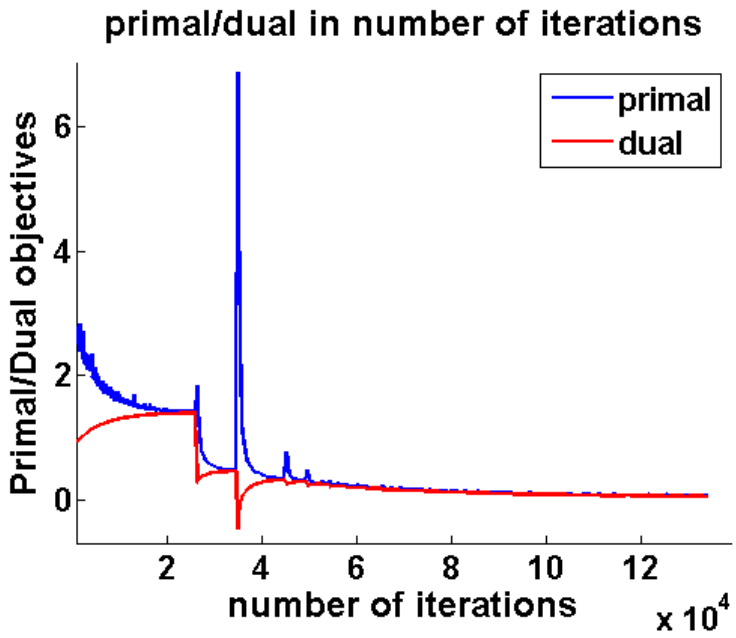

Figure 4.4: results for $n = 100$, $k = 3$, $s = 20$ and $\lambda = 0.0001$: Primal and dual objectives for SDCA

1.2 Comparison with other algorithms

We compare our SDCA method to a proximal gradient descent method called *Iterative Soft Thresholding* (IST) presented in [7], which is a batch method on generated data with $n = 100$, $k = 3$ and $s = 20$ and $\lambda = 0.015$. Figure 4.5 shows the ℓ_2 error in function of the number of visited entries. We remind that SDCA method processes one single training example at each step while the IST processes all the training dataset at each iteration. Hence one iteration of SDCA counts as one visited entry whereas one iteration of IST counts as N visited entries, where N is the number of known ratings. We see that SDCA needs to visit a smaller number of entries than IST to reach an acceptable solution. Here SDCA reaches an ℓ_2 error of 3×10^{-3} visiting 300,000 entries (it can visit an entry several times) wheras IST only reaches an ℓ_2 error of 0.3.

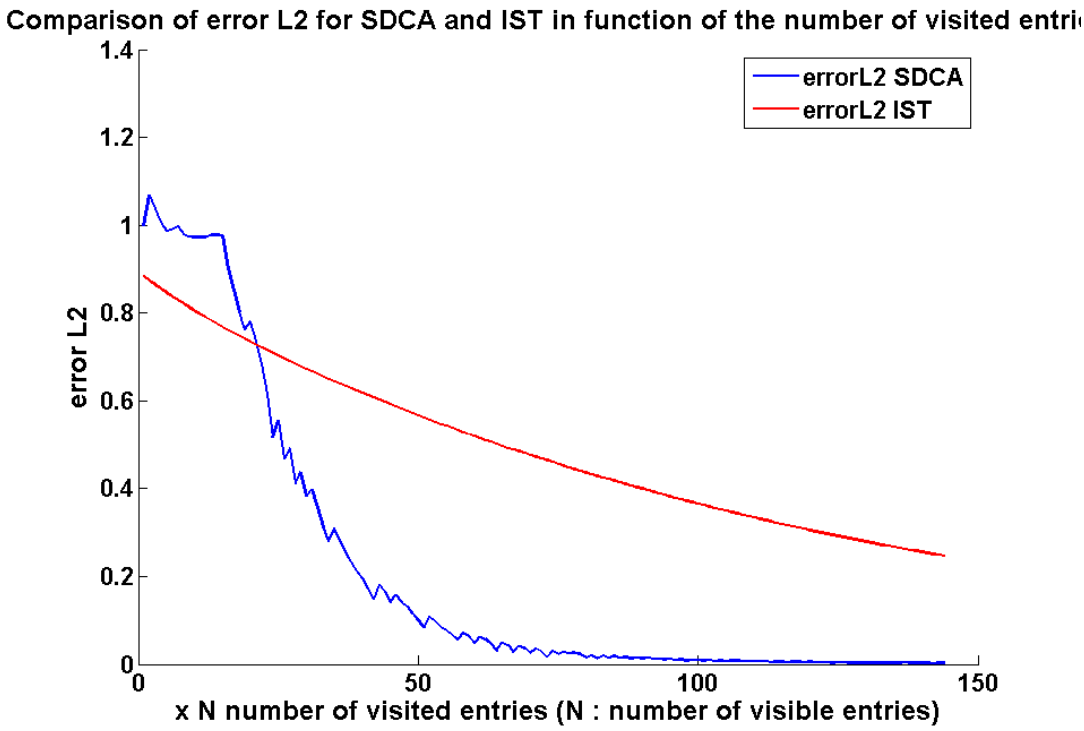

Figure 4.5: results for $n = 100$, $k = 3$, $s = 20$ and $\lambda = 0.015$: ℓ_2 error of SDCA and IST in function of the number of visited entries

Figure 4.6 compares SAG and SDCA on generated data with $n = 100$, $k = 3$ and $s = 20$ and $\lambda = 0.015$. Both are online algorithms but SAG operates on the primal whereas SDCA operates on a dual. SAG primal objective minimization curve is not necessarily monotone but it tends to be. Moreover, since the algorithm relies on primal iterations we do not have jumps in primal objective when alternating. The alternate minimization is visible on the ℓ_2 error curve of SDCA.

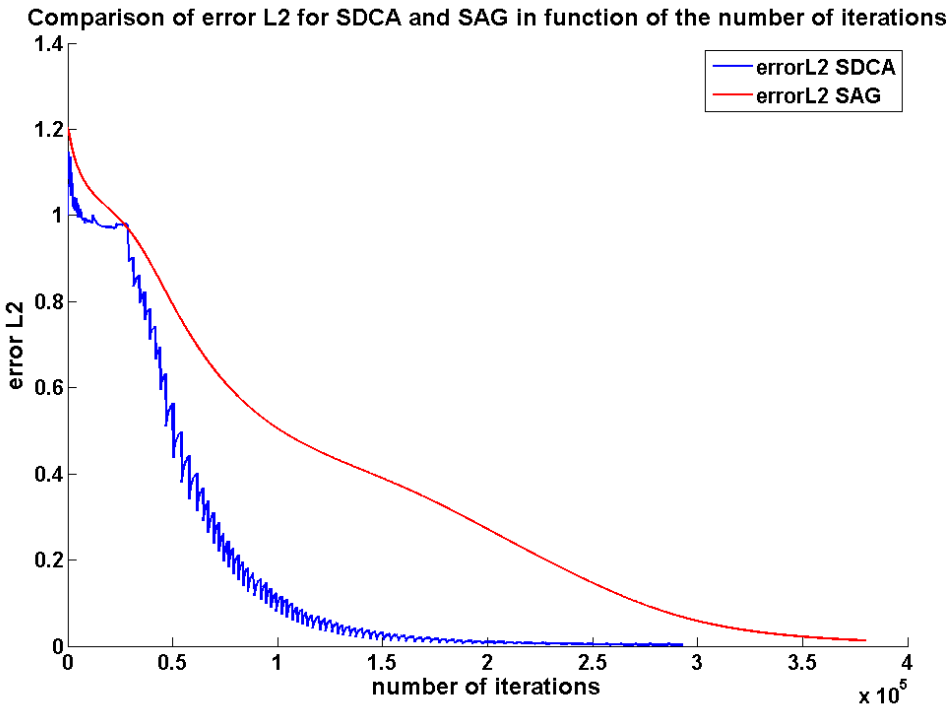

Figure 4.6: results for $n = 100$, $k = 3$, $s = 20$ and $\lambda = 0.015$: ℓ_2 error of SDCA and IST in function of the number of visited entries

2 Implementation details

Since we consider a regularized problem we have to fix the regularization parameter λ. If it is too large, the loss term in the primal objective will be too small compared to the regularization term and the solution will not fit well the data. On the contrary, when λ is small SDCA shows slower convergence. A way to adress this issue is to begin the optimization with a large λ and gradually decrease λ during iterations.

Another issue is how to set a stopping criterion. The duality gaps of the different sub-problems do not give us information on the general duality gap. In SDCA implementation we set a duality gap threshold that decreases during iterations. Each time the threshold is reached we alternate the minimization.

As shown is (3.5), the updates of $\boldsymbol{U}$ depend on $\boldsymbol{V}$ and the updates of $\boldsymbol{V}$ depend on $\boldsymbol{U}$. More particularly, if we define $\boldsymbol{A}$ the matrix of dual variables α_{ij} and $\boldsymbol{B}$ the matrix of dual variables β_{ij} we have that $\boldsymbol{U} = \boldsymbol{A}\,\boldsymbol{V}/N\lambda$ and $\boldsymbol{V} = \boldsymbol{B}\,\boldsymbol{U}/N\lambda$. Hence, after a cycle of updates of $\boldsymbol{U}$ leading to $\boldsymbol{U}^{(t)}, \boldsymbol{V}^{(t-1)}$ is no more up to date. We have to update the initial values of $\boldsymbol{V}$ to a $\boldsymbol{V}_{start} = \boldsymbol{B}\,\boldsymbol{U}^{(t)}/N\lambda$ before starting to optimize the problem on $\boldsymbol{V}$. Then we obtain $\boldsymbol{V}^{(t)}$ after optimizing on $\boldsymbol{V}$ from $\boldsymbol{V}_{start}$. This adds a step between updating cycles that has a cost of the number of non-zero α_{ij} values for processing $\boldsymbol{U}_{start}$ and the number of non-zero β_{ij} values for processing $\boldsymbol{V}_{start}$.

The dependency between $\boldsymbol{U}$ and $\boldsymbol{V}$ also prevent us to initialize both matrices as zero matrices. Indeed if we initialize at zero we will get stuck in this solution. In our experiments we

initialize $\boldsymbol{V}$ at random and $\boldsymbol{U} = \boldsymbol{A}\,\boldsymbol{V}/N\lambda$.

In our tests, we can choose the number of columns of matrices $\boldsymbol{U}$ and $\boldsymbol{V}$. We have observed that if we fix a small number of columns we tend to get stuck in a local minimum whereas when the number is sufficiently large we reach a better optimum. This is natural since when the number of columns is large we approach the convex problem with trace norm relaxation, as explained in section 2.3. For all the tests we fix the number of columns to 50.

Discussion

In SDCA, before each cycle, we have to update the starting point $\boldsymbol{U}_{start}$ or $\boldsymbol{V}_{start}$. For example, after optimizing over $\boldsymbol{U}$ we get a $\boldsymbol{U}^{(t)}$ and we update the starting point of the optimization over $\boldsymbol{V}$ to $\boldsymbol{V}_{start} = \boldsymbol{B}\,\boldsymbol{U}^{(t)}/N\lambda$. However this staring point is not necessarily better than $\boldsymbol{V}^{(t-1)}$. In certain cases the primal is degraded, and we observe jumps on primal objective. In this procedure each time we alternate we can loose optimization work that has already been done in the last cycle. In future work it would be interesting to find ways of addressing this issue.
The initialisation at random matrices is also a specificity of SDCA, since bi-convexity prevents us to initialize at zero, which is a stationary point for SDCA. Other methods, like IST start at zero matrix and take advantage of sparsity.
Future work should also compare SDCA with other algorithms as ADMM and *Block Coordinate Descent* (BCD). Some algorithms, as BCD, show good performance when considering the squared loss, notably because squared loss leads to close form updates. However this is not the case for other loss functions. It will be interesting to compare SDCA to these methods for other loss functions.

Conclusion

During this internship we have addressed the matrix completion problem, and in particular its bi-convex formulation. We have proposed an online algorithm based on SDCA that performs updates in a dual. Our preliminary results have shown that for certain. Further work should continue analysing SDCA performance. ON the other hand, SDCA has also shown some weaknesses. Since we do updates on the dual we do not have much control on the primal objective and some values of λ lead to jumps in primal objective.

We have decided to work on bi-convex formulation but it should be interesting to work on distributed and stochastic algorithms based on the convex formulation. In particular, [10] and [9] consider this ideas to solve individual sequences problems and PCA.

Annexe A

SDCA updates for collaborative filtering

Problème

Soient $\{x_{ij}\}$ les entrées accessibles de la matrice que l'on cherche à compléter. On cherche à approximer la matrice X par une matrice de rang faible qu'on écrit $UV^\top$ avec U et V peu de colonnes, i.e. k colonnes.
Le problème primal en u_i (la ligne i de la matrice U) s'écrit :

$$\min_{u_i \in \mathbb{R}^k} \sum_{j \in \Omega_i} \ell_{ij}(u_i v_j^\top) + \frac{\lambda}{2}\|u_i\|^2$$

On pose $w_{ij} = u_i v_j^\top$ et on rajoute la contrainte $w_{ij} = u_i v_j^\top$ pour tout $j \in \Omega_i$. Le lagrangien s'écrit :

$$\begin{aligned}
\mathcal{L}(u_i, w_{ij}, \alpha_{ij}) &= \sum_{j \in \Omega_i} \ell_{ij}(w_{ij}) + \frac{\lambda}{2}\|u_i\|^2 - \sum_{j \in \Omega_i} \alpha_{ij}(w_{ij} - u_i v_j^\top) \\
&= \sum_{j \in \Omega_i} (\ell_{ij}(w_{ij}) - \alpha_{ij} w_{ij}) + \frac{\lambda}{2}\left(\|u_i\|^2 + \frac{2}{\lambda}\sum_{j \in \Omega_i} \alpha_{ij} u_i v_j^\top\right)
\end{aligned}$$

$$\begin{aligned}
g(\alpha_i) &= \inf_{u_i, w_{ij}} \mathcal{L}(u_i, w_{ij}, \alpha_{ij}) \\
&= \sum_{j \in \Omega_i} \inf_{w_{ij}} (\ell_{ij}(w_{ij}) - \alpha_{ij} w_{ij}) + \frac{\lambda}{2} \inf_{u_i}\left(\|u_i\|^2 + \frac{2}{\lambda}\sum_{j \in \Omega_i} \alpha_{ij} u_i v_j^\top\right) \\
&= -\sum_{j \in \Omega_i} \ell_{ij}^*(\alpha_{ij}) - \frac{1}{2\lambda}\|\sum_{j \in \Omega_i} \alpha_{ij} v_j\|^2
\end{aligned}$$

Où ℓ_{ij}^* est la conjuguée de Fenchel de ℓ_{ij}. Pour la perte des moindres carrés $\ell_{ij}(y) = \frac{1}{2}(x_{ij} - y)^2$ nous avons :

$$\ell_{ij}^*(y) = \max_{t \in \mathbb{R}} yt - \frac{1}{2}(x_{ij} - t)^2$$

Le maximum est atteint pour t tel que $y + (x_{ij} - t) = 0$, i.e. $t = x_{ij} + y$. Donc

$$\ell_{ij}^*(y) = yx_{ij} + \frac{1}{2}y^2$$

Ainsi le dual s'écrit :

$$\max_{\alpha_i} -\frac{1}{2}\sum_{j\in\Omega_i}\alpha_{ij}^2 - \sum_{j\in\Omega_i}\alpha_{ij}x_{ij} - \frac{1}{2\lambda}\|\sum_{j\in\Omega_i}\alpha_{ij}v_j\|^2$$

Update des variables duales α_{ij}

Posons

$$h(\Delta\alpha_{ij}) = -\frac{1}{2}(\alpha_{ij}+\Delta\alpha_{ij})^2 - (\alpha_{ij}+\Delta\alpha_{ij})x_{ij} - \frac{1}{2\lambda}\|\sum_{j'\in\Omega_i}\alpha_{ij'}v_{j'} + \Delta\alpha_{ij}v_j\|^2$$

$$\begin{aligned} h'(\Delta\alpha_{ij}) &= -\alpha_{ij} - \Delta\alpha_{ij} - x_{ij} - \frac{1}{2\lambda}\left(2\sum_{j'\in\Omega_i}\alpha_{ij'}v_{j'}v_j^\top + 2\Delta\alpha_{ij}\|v_j\|^2\right) \\ &= -\left(1+\frac{\|v_j\|^2}{\lambda}\right)\Delta\alpha_{ij} - \alpha_{ij} - x_{ij} - \frac{1}{\lambda}\langle\sum_{j'\in\Omega_i}\alpha_{ij'}v_{j'}, v_j\rangle \end{aligned}$$

Ainsi on actualise α_{ij} de la façon suivante :

$$\begin{aligned} &\alpha_{ij} \leftarrow \alpha_{ij} + \Delta\alpha_{ij} \\ &\text{avec } \Delta\alpha_{ij} = \left(1+\frac{\|v_j\|^2}{\lambda}\right)^{-1}\left(-\alpha_{ij} - x_{ij} - \frac{1}{\lambda}\langle\sum_{j'\in\Omega_i}\alpha_{ij'}v_{j'}, v_j\rangle\right) \end{aligned}$$

Update des variables primales u_i, w_{ij}

Les conditions $\frac{\partial\mathcal{L}}{\partial u_i}(u_i, w_{ij}, \alpha_{ij}) = 0$ et $\frac{\partial\mathcal{L}}{\partial w_{ij}}(u_i, w_{ij}, \alpha_{ij}) = 0$ donnent :

$$u_i = -\frac{1}{\lambda}\sum_{j\in\Omega_i}\alpha_{ij}v_j$$

$$w_{ij} = x_{ij} + \alpha_{ij}$$

L'actualisation des variables primales s'écrit :

$$\begin{aligned} u_i &\leftarrow u_i - \frac{1}{\lambda}\Delta\alpha_{ij}v_j \\ w_{ij} &\leftarrow x_{ij} + \alpha_{ij} \end{aligned}$$

Duality gap

Loss duality gap

Objectif primal :

$$\sum_{j\in\Omega_i}(\ell_{ij}(w_{ij}) - \alpha_{ij}w_{ij})$$

Objectif dual :

$$-\sum_{j\in\Omega_i} \ell_{ij}^*(\alpha_{ij})$$

Duality gap :

$$\sum_{j\in\Omega_i} \left(\ell_{ij}(w_{ij}) - \alpha_{ij}w_{ij} + \ell_{ij}^*(\alpha_{ij})\right)$$

Pour la perte des moindres carrés :

$$\frac{1}{2}\sum_{j\in\Omega_i}(x_{ij}-w_{ij})^2 + \frac{1}{2}\sum_{j\in\Omega_i}\alpha_{ij}^2 + \sum_{j\in\Omega_i}\alpha_{ij}(x_{ij}-w_{ij})$$

Norm duality gap

Objectif primal :

$$\frac{\lambda}{2}\left(\|u_i\|^2 + \frac{2}{\lambda}\sum_{j\in\Omega_i}\alpha_{ij}u_i v_j^\top\right)$$

Objectif dual :

$$-\frac{1}{2\lambda}\|\sum_{j\in\Omega_i}\alpha_{ij}v_j\|^2$$

Duality gap :

$$\frac{\lambda}{2}\|u_i\|^2 + \sum_{j\in\Omega_i}\alpha_{ij}u_i v_j^\top + \frac{1}{2\lambda}\|\sum_{j\in\Omega_i}\alpha_{ij}v_j\|^2$$

Total duality gap

$$\frac{1}{2}\sum_{j\in\Omega_i}(x_{ij}-u_i v_j^\top)^2 + \frac{\lambda}{2}\|u_i\|^2 + \frac{1}{2}\sum_{j\in\Omega_i}\alpha_{ij}^2 + \sum_{j\in\Omega_i}\alpha_{ij}x_{ij} + \frac{1}{2\lambda}\|\sum_{j\in\Omega_i}\alpha_{ij}v_j\|^2$$

ADMM updates for collaborative filtering

Problème

Le problème initial s'écrit :

$$\min_{u_i}\frac{1}{2}\sum_{j\in\Omega_i}(x_{ij}-\langle u_i, v_j\rangle)^2 + \frac{\lambda}{2}\|u_i\|^2$$

Nous voulons appliquer l'algorithme ADMM (ref Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers - Boyd & Parikh) à ce problème. Cet algorithme combine l'avantage de la décomposition pour SDCA et les garanties de convergence de la méthode des multiplicateurs. Pour cela nous introduisons de nouvelles variables z_i^j.

$$\begin{aligned}&\min_{u_i,z_i}\frac{1}{2}\sum_{j\in\Omega_i}\left(x_{ij}-\langle z_i^j, v_j\rangle\right)^2 + \frac{\lambda}{2}\|u_i\|^2\\ &\text{s.t. } z_i^j = u_i\ \forall j\in\Omega_i\end{aligned}$$

Nous écrivons le lagrangien augmenté pour ce problème :

$$\mathcal{L}_{\rho_i}(u_i, z_i, \alpha_i) = \frac{1}{2}\sum_{j\in\Omega_i}\left(x_{ij} - \langle z_i^j, v_j\rangle\right)^2 + \frac{\lambda}{2}\|u_i\|^2 + \sum_{j\in\Omega_i}\alpha_i^{j\top}\left(z_i^j - u_i\right) + \frac{\rho_i}{2}\sum_{j\in\Omega_i}\|z_i^j - u_i\|^2$$

L'algorithme ADMM s'écrit comme suit. Il permet d'actualiser chaque variable z_i^j de façon indépendante.

$$z_i^{j(k+1)} := \arg\min_{z_i^j} \mathcal{L}_{\rho_i}(u_i^{(k)}, z_i, \alpha_i^{(k)}) \tag{1}$$

$$u_i^{(k+1)} := \arg\min_{u_i} \mathcal{L}_{\rho_i}(u_i, z_i^{(k+1)}, \alpha_i^{(k)}) \tag{2}$$

$$\alpha_i^{j(k+1)} := \alpha_i^{j(k)} + \rho_i\left(z_i^{j(k+1)} - u_i^{(k+1)}\right) \tag{3}$$

Calcul de (1):

$$\begin{aligned}\frac{\partial\mathcal{L}_{\rho_i}(u_i^{(k)}, z_i, \alpha_i^{(k)})}{\partial z_i^j} &= -\left(x_{ij} - \langle z_i^j, v_j\rangle\right)v_j + \alpha_i^{(k)} - \rho_i u_i^{(k)} + \rho_i z_i^j \\ &= z_i^j\left(\rho_i I_{rank} + v_j^\top v_j\right) - x_{ij}v_j + \alpha_i^{(k)} - \rho_i u_i^{(k)}\end{aligned}$$

Finalement nous avons la forme explicite de l'actualisation de z_i^j. Nous remarquons que chaque actualisation nécessite l'inversion d'une matrice de taille $rank \times rank$. $rank$ étant faible nous pouvons nous permettre cette inversion.

$$z_i^{j(k+1)} = \left(x_{ij}v_j - \alpha_i^{(k)} + \rho_i u_i^{(k)}\right)\left(\rho_i I_{rank} + v_j^\top v_j\right)^{-1} \tag{4}$$

Calcul de (2) :

$$\frac{\partial\mathcal{L}_{\rho_i}(u_i, z_i^{(k+1)}, \alpha_i^{(k)})}{\partial u_i} = \lambda u_i - |\Omega_i|\alpha_i^{(k)} + \rho_i\sum_{j\in\Omega_i}\left(u_i - z_i^{j(k+1)}\right)$$

Donc,

$$u_i^{(k+1)} = \frac{1}{\lambda + \rho_i}\sum_{j\in\Omega_i}\left(\alpha_i^{(k)} + \rho_i z_i^{j(k+1)}\right) \tag{5}$$

Les résidus primal $r_i^{(k)}$ et dual $s_i^{(k)}$ sont:

$$r_i^{(k)} = \left(z_i^{1(k)} - u_i^k, ..., z_i^{j_{|\Omega_i|}(k)} - u_i^k\right), \qquad s_i^{(k)} = \rho_i\left(u_i^{(k-1)} - u_i^k, ..., u_i^{(k-1)} - u_i^k\right)$$

Et leurs normes au carré sont :

$$\|r_i^{(k)}\|_2^2 = \sum_{j\in\Omega_i}\|z_i^{j(k)} - u_i^k\|_2^2, \qquad \|s_i^{(k)}\|_2^2 == |\Omega_i|\rho_i^2\|u_i^{(k-1)}\|_2^2$$

Bibliography

[1] J. Abernethy, F. Bach, T. Evgeniou, and J.P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *The Journal of Machine Learning Research*, 10:803–826, 2009.

[2] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 41–48. MIT Press, Cambridge, MA, 2007.

[3] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD Cup and Workshop*, 2007.

[4] Brice Hoffmann. *Développement d'approches de chémogénomique pour la prédiction des interactions protéine-ligand.* PhD thesis, École Nationale Supérieure des Mines de Paris, 2011.

[5] Laurent Jacob and Jean-Philippe Vert. Protein-ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics*, 24(19):2149–2156, 2008.

[6] Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems 25*, pages 3176–3184. 2012.

[7] Yunlong He Jingu Kim and Haesun Parki. Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework. *Journal of Global Optimization*, 2013.

[8] Joseph A Konstan and John Riedl. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22(1-2):101–123, 2012.

[9] Manfred K.Warmuth and Dima Kuzmin. Randomized online pca algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 2008.

[10] Andrew Cotter Raman Arora and Nathan Srebro. Stochastic optimization of pca with capped msg. *arXiv:1307.1674*, 2013.

[11] B. Recht and C. Ré. Parallel stochastic gradient algorithms for large-scale matrix completion. *Optimization Online*, 2011.

[12] Nicolas Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems 25*, pages 2672–2680. 2012.

[13] E. Chu B. Peleato S. Boyd, N. Parikh and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, pages 1–122, 2011.

[14] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14:567–599, 2013.

[15] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4, 2009.

[16] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research*, 10:623–656, 2009.

[17] C. Teflioudi, F. Makari, and R. Gemulla. Distributed matrix completion. *ICDM*, 2012.

[18] Dale Schuurmans Xinhua Zhang, Yao-Liang Yu. Accelerated training for matrix-norm regularization: A boosting approach. 2012.