MICAS Masters – Introduction to Optimization: Final Project

Solutions to be handed in by Friday 15 Nov.

Michèle Wigger, michele.wigger@telecom-paris.fr

Exercise 1: Matrix Factorization for Recommendation System

- Data matrix $\boldsymbol{X}$ of size $M \times N$. M = number of users, N = number of items
- $[\boldsymbol{X}]_{u,i}$ is rating (score) that user u gave to item i
- Recall: $[\boldsymbol{X}]_{u,i} = \boldsymbol{p}_u^T \boldsymbol{q}_i$, where $\boldsymbol{p}_u \in \mathbb{R}^d$ vector of **user preferences**, $\boldsymbol{q}_i \in \mathbb{R}^d$ vector of **item characteristics**
- Use entries w/ known rating as training set, $(u, i) \in \mathcal{K}$
- entries of $\mathcal{K}$ fulfill low-rank MF model: $[\boldsymbol{X}]_{u,i} = \boldsymbol{p}_u^T \boldsymbol{q}_i$, $\forall (u, i) \in \mathcal{K}$

We will build on that example by considering a realistic implementation using popular movie recommendation datasets, e.g., the MovieLens dataset.[1].

We denote by $\mathcal{K}$ the set of **rated entries** in $\boldsymbol{X}$, i.e., entries of the rating matrix which have a score/rating by at least one user. Then the low-rank MF model is learned from the set of rated entries, $\mathcal{K}$, i.e.,

$$\begin{cases} \text{argmin} & \sum_{(u,i)\in\mathcal{K}} \left(([\mathbf{X}]_{u,i} - \boldsymbol{p}_u^T \boldsymbol{q}_i)^2 + \rho_1^{(u)} \|\boldsymbol{p}_u\|_2^2 + \rho_2^{(i)} \|\boldsymbol{q}_i\|_2^2 \right) = f(\{\boldsymbol{p}_u \ , \boldsymbol{q}_i\}_{(u,i)\in\mathcal{K}}) \\ \text{s.t.} & \mathbf{p}_u \in \mathbb{R}^d \ , \ \mathbf{q}_i \in \mathbb{R}^d \ , \qquad \forall \ (u, i) \in \mathcal{K} \end{cases} \tag{1}$$

In general, $\{\rho_1^{(u)} > 0, \rho_2^{(i)} > 0\}_{(u,i)\in\mathcal{K}}$ are regularization parameters to prevent overfitting, tuned via cross-validation. However, here we will focus on the training part (rather than the testing) and assume that all these regularization parameters are **strictly positive** and **given** to us (choose them at random between $]0, 10]$, such that they are all strictly positive). Due to the presence of coupling in the cost function, Block-Coordinate Descent (BCD) methods are prevalent for solving (1). We thus propose to use the following standard method to solve (1): Block-Coordinate Descent (BCD) with closed-form solution for each subproblem.

[1] You download that dataset along with its description here https://grouplens.org/datasets/movielens/100k/

0a) Starting from (1), apply the BCD method to derive the subproblem for each block of coordinates, In other words, derive the subproblem corresponding to each of the optimization variables in (1), $\{\boldsymbol{p}_i \ , \ \boldsymbol{q}_u\}_{(u,i) \ \in \mathcal{K}}$

0b) show that each subproblem is strongly convex.

1) **BCD with closed-form solution:** also called alternating least-squares
a) derive the update for each of the subproblems (part 0a)) in **closed-form solution**
b) show that algorithm which results from these updates **converges monotonically to a stationary point** of (1)
c) derive an estimate of the **computational complexity** per BCD iteration (in $\mathcal{O}()$ notation)

Exercise 2: Traveling Salesman Problem

The *traveling salesman problem* addresses the following problem: "Given a list of cities and the distances between each pair of cities, find the shortest possible route that visits each city exactly once and returns to the origin city."

In this exercise we wish to compare how the heuristic algorithms we have seen in class solve the traveling salesman problem at hand of some data sets in:

`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/`

- Solve the traveling salesman problem using the simulated annealing method. Thereby test three different ways for the step where you change to a nearby solution.
- Solve the traveling salesman problem using a genetic algorithm. Test different algorithms for the crossing phases and mutation phases.
- Solve the traveling salesman problem using the ant colony optimization algorithm. Test different parameters in your implementations. Compare also to at least one variant of the ant colony optimization algorithm.
- Compare your three algorithms that you have implemented in 1.-3 at hand of smaller and larger data sets. Which of your algorithms do you prefer in terms of solution accuracy and run time?