Mathematics, Statistics, Optimization, and Machine Learning

Sunghee Yun

December 8, 2019

Contents

Part I

Mathematics

Chapter 1

Calculus

Chapter 2

Convex analysis

Chapter 3

Linear Algebra

Part II

Optimization

Chapter 4

Convex Optimization

4.1 Mathematical optimization problem

A mathematical optimization problem can be expressed as

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0 \text{ for } i = 1, \ldots, m \\ & h_i(x) = 0 \text{ for } i = 1, \ldots, p \end{array} \tag{4.1}$$

where $x \in \mathbf{R}^n$ is the optimization variable, $f_0 : \mathbf{R}^n \to \mathbf{R}$ is the objective function, $f_i : \mathbf{R}^n \to \mathbf{R}$ for $i = 1, \ldots, n$ are the inequality constraint functions, and $h_i : \mathbf{R}^n \to \mathbf{R}$ for $i = 1, \ldots, p$ are the equality constraint functions.

The conditions, $f_i(x) \leq 0$ for $i = 1, \ldots, m$, are called inequality constraints and the conditions, $h_i(x) = 0$ for $i = 1, \ldots, p$ are called equation constraints.

Note that this formulation covers pretty much every single-objective optimization problem. For example, consider the following optimization problem.

$$\begin{array}{ll} \text{maximize} & f(x_1, x_2) \\ \text{subject to} & x_1 \geq x_2 \\ & x_1 + x_2 = 2 \end{array} \tag{4.2}$$

This problem can be cast into an equivalent problem as follows.

$$\begin{array}{ll} \text{minimize} & -f(x_1, x_2) \\ \text{subject to} & -x_1 + x_2 \leq 0 \\ & x_1 + x_2 - 2 = 0 \end{array} \tag{4.3}$$

The feasible set for (4.1) is defined by the set of $x \in \mathbf{R}^n$ which satisfies all the contraints. Also, the optimal value for (4.1) is the infimum of $f_0(x)$ while x is in the feasible set. When the infimum is achievable, we define the optimal solution set as the set of all feasible x achieving the infimum value. These are defined in mathematically rigorous terms below.

- The feasible set for (4.1) is defined by

  $$\mathcal{F} = \{x \in \mathcal{D} \mid f_i(x) \leq 0 \text{ for } i = 0, \ldots, m, \; h_j(x) = 0 \text{ for } j = 1, \ldots, p\} \subseteq \mathbf{R}^n \tag{4.4}$$

  where

  $$\mathcal{D} = \left(\bigcap_{0 \leq i \leq m} \mathbf{dom}\, f_i \right) \cap \left(\bigcap_{1 \leq i \leq p} \mathbf{dom}\, h_i \right). \tag{4.5}$$

- The optimal value for (4.1) is defined by

  $$p^* = \inf_{x \in \mathcal{F}} f_0(x) \tag{4.6}$$

  We use the conventions that $p^* = -\infty$ if $f_0(x)$ is unbounded below for $x \in \mathcal{F}$ and that $p^* = \infty$ if $\mathcal{F} = \emptyset$.

- The optimal solution set for (4.1) is defined by

  $$\mathcal{X}^* = \{x \in \mathcal{F} \mid f_0(x) = p^*\}. \tag{4.7}$$

4.2 Convex optimization problem

A mathematical optimization problem is called a convex optimization problem if the objective function and all the inequality constraint functions are convex functions and all the equality constraint functions are affine functions.

Hence, a convex optimization problem can be expressed as

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0 \text{ for } i = 1, \ldots, m \\ & Ax = b \end{array} \tag{4.8}$$

where $x \in \mathbf{R}^n$ is the optimization variable, $f_i : \mathbf{R}^n \to \mathbf{R}$ for $i = 0, \ldots, n$ are convex functions, $h_i : \mathbf{R}^n \to \mathbf{R}$ for $i = 1, \ldots, p$ are the equality constraint functions, $A \in \mathbf{R}^{p \times n}$, and $b \in \mathbf{R}^p$.

A function, $f : \mathbf{R}^n \to \mathbf{R}$, is called a convex function if $\mathbf{dom}\, f \subseteq \mathbf{R}^n$ is a convex set and for all $x, y \in \mathbf{dom}\, f$ and all $0 \leq \lambda \leq 1$,

$$f(\lambda x + (1-\lambda) y) \leq \lambda f(x) + (1-\lambda) f(y) \tag{4.9}$$

where $\mathbf{dom}\, f \subseteq \mathbf{R}^n$ denotes the domain of f.

A convex optimization enjoys a number of nice theoretical and practical properties.

- A local minimum of a convex optimization problem is a global minimum, *i.e.*, if for some $R > 0$ and $x_0 \in \mathcal{F}$, $\|x - x_0\| < R$ and $x \in \mathcal{F}$ imply $f_0(x_0) \leq f_0(x)$, then $f_0(x_0) \leq f_0(x)$ for all $x \in \mathcal{F}$.

  Proof: Assume that $x_0 \in \mathcal{F}$ is a local minimum, *i.e.*, for some $R > 0$, $\|x - x_0\| < R$ and $x \in \mathcal{F}$ imply $f_0(x_0) \leq f_0(x)$.

  Now assume that x_0 is not a global minimum, *i.e.*, there exists $y \in \mathcal{F}$ such that $y \neq x_0$ and $f_0(y) < f_0(x_0)$. Then for $z = \lambda y + (1-\lambda) x_0$ with $\lambda = \min\{R/\|y - x_0\|, 1\}/2$, the convexity of f_0 implies

  $$f_0(z) \leq \lambda f_0(y) + (1-\lambda) f_0(x_0) \tag{4.10}$$

  since $0 < \lambda \leq 1/2 < 1$. Furthermore

  $$\|z - x_0\| = \lambda \|y - x_0\| \leq R/2, \tag{4.11}$$

  hence $f_0(z) \geq f_0(x_0)$, which together with (4.10) implies

  $$f_0(x_0) \leq f_0(z) \leq \lambda f_0(y) + (1-\lambda) f_0(x_0) < \lambda f_0(x_0) + (1-\lambda) f_0(x_0) = f_0(x_0), \tag{4.12}$$

  which is a contradiction. Therefore there is no $y \in \mathcal{F}$ such that $y \neq x_0$ and $f_0(y) < f_0(x_0)$. Therefore x_0 is a global minimum.

- For a unconstrained problem, *i.e.*, the problem (4.8) with $m = p = 0$, with differentiable objective function, $x \in \mathbf{dom}\, f_0$ is an optimal solution if and only if $\nabla f_0(x) = 0 \in \mathbf{R}^n$.

Proof: The Taylor theorem implies that for any $x, y \in \mathbf{dom}\, f_0$,

$$f_0(y) = f(x) + \nabla f_0(x)^T (y - x) + \frac{1}{2}(y-x)^T \nabla^2 f_0(z)(y-x) \tag{4.13}$$

for some z on the line segment having x and y as its end points, *i.e.*, $z = \alpha x + (1-\alpha)y$ for some $0 \leq \alpha \leq 1$. Since $\nabla^2 f(x) \succeq 0$ for any $z \in \mathbf{dom}\, f_0$, we have

$$f_0(y) \geq f_0(x) + \nabla f_0(x)^T (y - x) \tag{4.14}$$

Thus, if for some $x_0 \in \mathbf{R}^n$, $\nabla f_0(x_0) = 0$, for any $x \in \mathbf{dom}\, f_0$,

$$f_0(x) \geq f_0(x_0) + \nabla f_0(x_0)^T (x - x_0) = f_0(x_0), \tag{4.15}$$

hence x_0 is an optimal solution. Now assume that x_0 is an optimal solution, but $\nabla f_0(x_0) \neq 0$. Then for any $k > 0$, if we let $x = x_0$ and $y = x_0 - k\nabla f_0(x_0)$, (4.13) becomes

$$\begin{aligned} f_0(y) = f(x_0) + \nabla f_0(x_0)^T(-k\nabla f_0(x_0)) + \frac{k^2}{2}\nabla f_0(x_0)^T \nabla^2 f_0(z) \nabla f_0(x_0) \\ = \quad f(x_0) - k\|\nabla f_0(x_0)\|^2 + \frac{k^2}{2}\nabla f_0(x_0)^T \nabla^2 f_0(z) \nabla f_0(x_0) \end{aligned}$$

for all $y = x_0 - k\nabla f_0(x_0) \in \mathbf{dom}\, f_0$.

Since for $k < 2\|\nabla f_0(x_0)\|^2 / \nabla f_0(x_0)^T \nabla^2 f_0(z) \nabla f_0(x_0)$, $-k\|\nabla f_0(x_0)\|^2 + \frac{k^2}{2}\nabla f_0(x_0)^T \nabla^2 f_0(z) \nabla f_0(x_0) < 0$, thus $f_0(y) < f(x_0)$, hence the constradiction. Therefore, if x_0 is an optimal solution for the unconstrained problem, $\nabla f_0(x_0) = 0$.

4.3 Duality

4.3.1 The Lagrange dual problem

4.3.1.1 Examples

4.3.1.1.1 Standard form LP

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \succeq 0 \end{array} \tag{4.16}$$

The Lagrange dual problem is

$$\begin{array}{ll} \text{maximize} & -b^T \nu \\ \text{subject to} & A^T \nu + c \geq 0 \end{array} \tag{4.17}$$

4.3.1.1.2 Inequality form LP

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \preceq b \end{array} \tag{4.18}$$

The Lagrange dual problem is

$$\begin{array}{ll} \text{maximize} & -b^T \lambda \\ \text{subject to} & A^T \lambda + c = 0 \\ & \lambda \succeq 0 \end{array} \tag{4.19}$$

4.3.1.1.3 Least-squares solution of linear equations

$$\begin{array}{ll} \text{minimize} & (1/2)x^T x \\ \text{subject to} & Ax = b \end{array} \tag{4.20}$$

The Lagrange dual problem is

$$\text{maximize} \quad -(1/2)\nu^T AA^T \nu - b^T \nu \tag{4.21}$$

4.3.1.1.4 Entropy maximization

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n x_i \log x_i \\ \text{subject to} & Ax = b \\ & \mathbf{1}^T x = 1 \end{array} \tag{4.22}$$

with domain $\mathcal{D} = \mathbf{R}_+^n$

The Lagrange dual problem is

$$\begin{array}{ll} \text{maximize} & -b^T \lambda - \log \left(\sum_{i=1}^n \exp(-a_i^T \lambda) \right) \\ \text{subject to} & \lambda \succeq 0 \end{array} \tag{4.23}$$

4.3.2 Interpretations

4.3.2.1 Max-min characterization of weak and strong duality

We first note that for any $f : X \times Y \to \mathbf{R}$, we have

$$\sup_{y \in Y} \inf_{x \in X} f(x, y) \le \inf_{x \in X} \sup_{y \in Y} f(x, y). \tag{4.24}$$

This inequality is called *max-min inequality.*

We can prove this as follows. Let $g : Y \to \mathbf{R}$ be a function defined by $g(y) = \inf_{x \in X} f(x, y)$ and let $h : X \to \mathbf{R}$ be a function defined by $h(x) = \sup_{y \in Y} f(x, y)$. Then we have that for any $x \in X$ and $y \in Y$

$$g(y) = \inf_{x \in X} f(x, y) \le f(x, y), \tag{4.25}$$

which implies that for any $x \in X$

$$\sup_{y \in Y} g(y) \leq \sup_{y \in Y} f(x, y) = h(x). \tag{4.26}$$

This again implies that

$$\sup_{y \in Y} g(y) \leq \inf_{x \in X} h(x), \tag{4.27}$$

hence the proof.

4.3.2.2 Saddle-point interpretation

Suppose $f : X \times Y \to \mathbf{R}$. We refer a point $(\tilde{x}, \tilde{y}) \in X \times Y$ a *saddle-point* for f (and X and Y) if

$$f(\tilde{x}, y) \leq f(\tilde{x}, \tilde{y}) \leq f(x, \tilde{y}) \tag{4.28}$$

for all $x \in X$ and $y \in Y$.

Now if x^* and λ^* are primal and dual optimal points for a problem in which strong duality obtains, the form a saddle-point for the Lagrangian. Conversely, if (x, λ) is a saddle-point of the Lagragian, then x is primal optimal, λ is dual optimal, and the optimal duality gap is zero.

To prove these, assume that $x^* \in \mathcal{D}$ and $(\lambda^*, \nu^*) \in \mathbf{R}^m_+ \times \mathbf{R}^p$ are primal and dual optimal points for a problem in which strong duality obtains. Then for any $x \in \mathcal{D}$ and $(\lambda, \nu) \in \mathbf{R}^m_+ \times \mathbf{R}^p$, we have

$$L(x^*, \lambda, \nu) = f_0(x^*) + \sum_{i=1}^m \lambda_i f_i(x^*) + \sum_{i=1}^p \nu_i h_i(x^*) \leq f_0(x^*) = g(\lambda^*, \nu^*) \leq L(x, \lambda^*, \nu^*) \tag{4.29}$$

where the left inequality comes from the fact that $\lambda_i f_i(x^*) \leq 0$ for all $i = 1, \ldots, m$ and $h_i(x^*) = 0$ for all $i = 1, \ldots, p$ and the right inequality comes from the definition of (Lagrange) dual function. Now from the complementary slackness we know that $\lambda_i f_i(x^*) = 0$ for all $i = 1, \ldots, m$. Therefore

$$L(x^*, \lambda^*, \nu^*) = f_0(x^*), \tag{4.30}$$

thus we have

$$L(x^*, \lambda, \nu) \leq L(x^*, \lambda^*, \nu^*) \leq L(x, \lambda^*, \nu^*), \tag{4.31}$$

hence the proof.

Now suppose that $\tilde{x} \in \mathcal{D}$ and $(\tilde{\lambda}, \tilde{\nu}) \in \mathbf{R}^m_+ \times \mathbf{R}^p$ are the saddle-point of the Lagrangian, *i.e.*, for all $x \in \mathcal{D}$ and $(\lambda, \nu) \in \mathbf{R}^m_+ \times \mathbf{R}^p$,

$$L(\tilde{x}, \lambda, \nu) \leq L(\tilde{x}, \tilde{\lambda}, \tilde{\nu}) \leq L(x, \tilde{\lambda}, \tilde{\nu}). \tag{4.32}$$

First we show that $\tilde{x}$ is a feasible point. The left inequality says that for all $(\lambda, \nu) \in \mathbf{R}^m_+ \times \mathbf{R}^p$,

$$L(\tilde{x}, \lambda, \nu) = f_0(\tilde{x}) + \sum_{i=1}^m \lambda_i f_i(\tilde{x}) + \sum_{i=1}^p \nu_i h_i(\tilde{x}) \leq L(\tilde{x}, \tilde{\lambda}, \tilde{\nu}) \tag{4.33}$$

If $f_i(\tilde{x}) > 0$ for some $i \in \{1, \ldots, m\}$ or $h_i(\tilde{x}) \neq 0$ for some $i \in \{1, \ldots, p\}$, $L(\tilde{x}, \lambda, \nu)$ is unbounded above and the above inequality cannot hold. Therefore $f_i(\tilde{x}) \leq 0$ for all $i \in \{1, \ldots, m\}$ and $h_i(\tilde{x}) = 0$

for all $i \in \{1, \ldots, p\}$, *i.e.*, $\tilde{x}$ is primal feasible. Since the inequality must hold when $\lambda = 0$ and $\nu = 0$, we have

$$f(\tilde{x}) \leq L(\tilde{x}, \tilde{\lambda}, \tilde{\nu}). \tag{4.34}$$

The right inequality of (4.32) implies that

$$L(\tilde{x}, \tilde{\lambda}, \tilde{\nu}) \leq g(\tilde{\lambda}, \tilde{\nu}) = \inf_{x \in \mathcal{D}} L(x, \tilde{\lambda}, \tilde{\nu}), \tag{4.35}$$

which implies that $f_0(\tilde{x}) \leq g(\tilde{\lambda}, \tilde{\nu})$. Since $g(\lambda, \nu)$ is an underestimator of $f_0(x)$ for any feasible $x \in \mathcal{D}$ and $(\tilde{\lambda}, \tilde{\nu}) \in \mathbf{R}_+^m \times \mathbf{R}^p$, *i.e.*, $g(\tilde{\lambda}, \tilde{\nu}) \leq f_0(\tilde{x})$, thus $g(\tilde{\lambda}, \tilde{\nu}) = f_0(\tilde{x})$. Therefore $\tilde{x}$ is an optimal solution for the primal problem and $(\tilde{\lambda}, \tilde{\nu})$ is an optimal solution for the dual problem, hence the proof.

4.4 Unconstrained minimization

4.4.1 Gradient descent method

4.4.1.1 Examples

4.4.1.1.1 A quadratic problem in $\mathbf{R}^2$ We consider the quadratic objective function on $\mathbf{R}^2$

$$f(x) = \frac{1}{2}(x_1^2 + \gamma x_2^2) \tag{4.36}$$

where $\gamma > 0$.

We apply the gradient descent method with exact line search. The gradient of f is

$$\nabla f(x) = \begin{bmatrix} x_1 \\ \gamma x_2 \end{bmatrix} \tag{4.37}$$

Let $\tilde{f} : \mathbf{R}_+ \to \mathbf{R}$ defined by $\tilde{f}(t) = f(x - t\nabla f(x))$. Now

$$\tilde{f}(t) = f\left(\begin{bmatrix} (1-t)x_1 \\ (1-\gamma t)x_2 \end{bmatrix}\right) = \frac{1}{2}\left((1-t)^2 x_1^2 + \gamma(1-\gamma t)^2 x_2^2\right) \tag{4.38}$$

and

$$\frac{d}{dt}\tilde{f}(t) = -(1-t)x_1^2 - \gamma^2(1-\gamma t)x_2^2 = 0 \tag{4.39}$$

implies

$$t = \frac{x_1^2 + \gamma^2 x_2^2}{x_1^2 + \gamma^3 x_2^2} \tag{4.40}$$

minimizes $\tilde{f}(t)$. Since

$$1 - t = \frac{\gamma^2(\gamma - 1)x_2^2}{x_1^2 + \gamma^3 x_2^2} \tag{4.41}$$

and

$$1 - \gamma t = \frac{(1-\gamma)x_1^2}{x_1^2 + \gamma^3 x_2^2} \tag{4.42}$$

Thus the exact line search yields

$$x^+ = x - t\nabla f(x) = \left[\begin{array}{c} (1-t)x_1 \\ (1-\gamma t)x_2 \end{array}\right] = \frac{(1-\gamma)x_1x_2}{x_1^2+\gamma^3 x_2^2}\left[\begin{array}{c} -\gamma^2 x_2 \\ x_1 \end{array}\right]. \tag{4.43}$$

If $x = \alpha[\gamma\ 1]^T$, then

$$x^+ = \frac{\alpha^3(1-\gamma)\gamma}{\alpha^2\gamma^2(1+\gamma)}\left[\begin{array}{c} -\gamma^2 \\ \gamma \end{array}\right] = \alpha\frac{1-\gamma}{1+\gamma}\left[\begin{array}{c} -\gamma \\ 1 \end{array}\right]. \tag{4.44}$$

If $x = \alpha[-\gamma\ 1]^T$, then

$$x^+ = -\frac{\alpha^3(1-\gamma)\gamma}{\alpha^2\gamma^2(1+\gamma)}\left[\begin{array}{c} -\gamma^2 \\ -\gamma \end{array}\right] = \alpha\frac{1-\gamma}{1+\gamma}\left[\begin{array}{c} \gamma \\ 1 \end{array}\right]. \tag{4.45}$$

Therefore if $x^{(0)} = [\gamma\ 1]^T$, then

$$x^{(k)} = \left(\frac{1-\gamma}{1+\gamma}\right)^k \left[\begin{array}{c} (-1)^k\gamma \\ 1 \end{array}\right] = \left(\frac{\gamma-1}{\gamma+1}\right)^k \left[\begin{array}{c} \gamma \\ (-1)^k \end{array}\right]. \tag{4.46}$$

Chapter 5

Portfolio optimization

Part III

Statistics

Part IV

Machine Learning

Chapter 10

Optimization for Machine Learning

Chapter 11

Bayesian Network

Chapter 12

Collaborative Filtering

12.1 Item-based Collaborative Filtering

The purpose of the item-based collaborative filtering is to recommend items using the information obatined from similar items. Since the information from other items help provide better recommendation to customers, it is called an *item-based collaborative filtering.*

We formulate an item-based collaborative filtering as follows. We assume that there are n_C customers and n_I items. We further assume that we have partial information as to the taste of each customer for each item. In typical cases, we have customers can leave reviews with ratings for n items where $n \ll n_I$. When we have hundreds of thousands of items, this makes the data structure extremely sparse. This fact plays a critical role when we calculate similarities among items or customers, or learn the matrix factorization-based collaborative filtering models, e.g., using gradient descent (GD) method or alternating least squares (ALS) method.[1]

Now suppose that we have the rating matrix

$$R \in (\mathbf{R} \cup \{\mathrm{NaN}\})^{n_C \times n_I} \tag{12.1}$$

where $R_{ij} \in (\mathbf{R} \cup \{\mathrm{NaN}\})$ denotes *certain score* corresponding to ith customer and jth item and NaN denotes we do not know the values. Note that this certain score can refer to rating of movies, but can mean virtually anything related to customers' taste or activities. For example, it can mean the frequency of customers' clicks on certain menus where the items mean the menu items for this example.

This matrix is sparse, but not in a traditional way. Generally, we say a matrix is sparse if the number of nonzero entries is much less than $n_C \times n_I$. We say R is sparse because there are huge number of entries for which we *do not know* the true values for them. In a general recommendation problem, the purpose is to guess or estimate the true values, *e.g.*, what rating the customer would give if she did, or how frequently a customer would click on the menu item if she knew there exist such menu item.

Now we want to evaluate the similarity (or distance) among items. In many places in the recommendation systems and information retrieval literature, it is shown that applying some transformation or weighting on the values before calculating the similarity measure, *e.g.*, cosine similarity or correlation coefficient[2]. There weighting methods and similarity measures will be discussed in later sections.

12.1.1 Rating matrix modeling for menu personalization for mobile shopping app

Here we consider a problem of using a collaborative filtering for efficient menu personalization problem for an mobile shoppiong app. Suppose that we have some history data of the number of clicks or tabs on each menu item by each customer for certain period. Therefore we can define the matrix in (12.1) where n_C denotes the number of customers, n_I denotes the number of menu items, and R_{ij} denotes the number of clicks or tabs on jth menu item by ith customer with $1 \leq i \leq n_C$ and $1 \leq j \leq n_I$. recommendation system design cases where each component represents the rating

[1] These models will be discussed in later sections.

[2] The correlation coefficient is sometimes called the Pearson correlation coefficient after Karl Pearson, who was an English mathematician and biostatistician.

of an item, these numbers vary depending on the time interval for which we collect the data. So here we model this matrix as a function of time. Moreover, we can have more than one source for the customers' data. Thus we assume multi-source model.

12.1.1.1 Rating matrix modeling

Suppose that $R_s : \mathbf{R} \to \mathbf{R}^{n_C \times n_I}$ for $s \in \mathcal{S}$ is the rating matrix dependent on time, *i.e.*,

$$R_s(t) \in \mathbf{R}^{n_C \times n_I} \tag{12.2}$$

where $\mathcal{S}$ refers to the set of data sources and $R_s(t)_{ij}$ refers to the number of clicks or tabs for jth item by ith customer which was collected from data source, $s \in \mathcal{S}$. For example, $\mathcal{S} = \{\mathrm{mobile}, \mathrm{web}\}$.

For our purpose, we want to have one matrix which represents the customers' activity or behavior with their taste by properly aggregating the history data. Among many possible choices, we choose exponentially decaying weight method together with proper weight on data sources. We define the following aggregate rating matrix

$$\bar{R}(t) = \sum_{s \in \mathcal{S}} \sum_{\tau=0}^{\infty} \gamma^\tau w_s R_s(t - \tau) \tag{12.3}$$

where γ is a positive constant less than 1 and w_s are the weights on each data source such that $\sum_{s \in \mathcal{S}} w_s = 1$.

This aggregate matrix, however, has one problem. Suppose that the 1st customer and the 2nd customer have the very same activity pattern on menu items. However, the 1st customer happens to have actively used the shopping app or website for this week, but the 2nd customer has not used the app or website for the past few days or weeks. Because of the decay factor, the 2nd customer's activity would appear much less than those of the 1st customer even ideally we need to have the very same pattern for those two customers. Therefore we need to normalize the values so that these two customers' preferences are considered the same. For this we normalize each row, so that the sum of each row is always one.

$$\tilde{R}(t) = \mathbf{diag}(\bar{R}(t)\mathbf{1}_{n_I})^{-1}\bar{R}(t) \tag{12.4}$$

where $\mathbf{1}_{n_I} \in \mathbf{R}^{n_I}$ is the vector where every entry is 1 and $\mathbf{diag}(x)$ for $x \in \mathbf{R}^n$ refers to a diagonal matrix whose diagonal entries are the entries of x in the same order. We can readily see that

$$\tilde{R}(t)\mathbf{1}_{n_I} = \mathbf{diag}(\bar{R}(t)\mathbf{1}_{n_I})^{-1}\bar{R}(t)\mathbf{1}_{n_I} = \mathbf{1}_{n_C}, \tag{12.5}$$

i.e., the sum of each row is 1. From this point on, we will remove the subscript for $\mathbf{1}$ unless it can cause confusion as to the dimension of the vector.

One problem of this approach is that $\tilde{R}(t)$ cannot be defined if some row has all zero entries. Even if the sum of every row is nonzero, if the values is very small, *e.g.*, a customer has not shown any activity except for a few clicks or tabs on a handful of menu items, that doesn't mean that the corresponding row represents the customer's preference. Hence, we can augment the values of $\tilde{R}(t)$ by prior distribution of menu items as in the following section.

12.1.2 Value augmentation based on Bayesian MAP

To prevent the divide-by-zero errors from occurring while evaluating (12.4), we consider maximum a posteriori (MAP) estimation for all the rows of $\bar{R}(t)$ in (12.3).

We can think of a problem of filling out each entry in $\bar{R}(t)$ as performing N multinomial experiments on each item $j \in \{1, \ldots, n_I\}$ and count the occurrences for each item and fill in $\bar{R}(t)$ for each customer $i \in \{1, \ldots, n_C\}$ after normalization.

Now assume that we the prior as Dirichlet-multinomial model, *i.e.*,

$$\mathrm{Dir}(\theta|\alpha) = \frac{1}{B(\alpha)} \prod_{k=1}^{n_I} \theta_k^{\alpha_k - 1} I(x = k). \tag{12.6}$$

Then since the likelihood for the multinomial distribution has the form

$$p(\mathcal{D}|\theta) = \prod_{k=1}^{n_I} \theta_k^{N_k}, \tag{12.7}$$

the posterior is

$$p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta) \propto \prod_{k=1}^{n_I} \theta_k^{\alpha_k + N_k - 1} = \mathrm{Dir}(\theta|\alpha_1 + N_1, \ldots, \alpha_{n_I} + N_{n_I}). \tag{12.8}$$

It can be proved that the maximum a posteriori (MAP) estimate for $\theta_1, \ldots, \theta_{n_I}$ is

$$\hat{\theta}_k = \frac{N_k + \alpha_k - 1}{N + \alpha_0 - n_I} \tag{12.9}$$

where $\alpha_0 = \sum_{k=1}^{n_I} \alpha_k$ (K.P. Murphy). By choosing proper values for $\alpha_k > 1$, we can make every entry nonzero in $\bar{R}(t)$.

Adding this information to the rating matrix is equivalent to Bayesian inference since this uses a priori distribution. Therefore, it will play a regularization role in our inference.

12.1.3 Similarity measure among items

Suppose that we have the transformed matrix, $\tilde{R} \in \mathbf{R}^{n_C \times n_I}$. For the case of mobile shopping app menu personalization, $\tilde{R} = \tilde{R}(t)$ for some t. For general cases, $\tilde{R}$ equals to R with all NaNs replaced by 0.

Suppose that $c_1, \ldots, c_{n_C} \in \mathbf{R}^{n_I}$ are the row vectors of $\tilde{R}$ and $d_1, \ldots, d_{n_I} \in \mathbf{R}^{n_C}$ are the column vectors of $\tilde{R}$, *i.e.*,

$$\tilde{R} = \begin{bmatrix} c_1^T \\ \vdots \\ c_{n_C}^T \end{bmatrix} = \begin{bmatrix} d_1 & \cdots & d_{n_I} \end{bmatrix} \in \mathbf{R}^{n_C \times n_I}. \tag{12.10}$$

12.1.3.1 Cosine similarity

The cosine similarity measures the cosine of the angle between the two vectors, *i.e.*, it is defined by

$$s(d_i, d_j) = \frac{d_i^T d_j}{\|d_i\|_2 \|d_j\|_2} \tag{12.11}$$

where $x^T y$ denotes the inner product of two vectors x and y, and $\| \cdot \|_2$ denotes the 2-norm of a vector. The Jensen's inequality guarantees that $-1 \leq s(d_i, d_j) \leq 1$. It can be easily shown that we have $0 \leq s(d_i, d_j) \leq 1$ when every entry in $\tilde{R}$ is nonzero.

12.1.3.2 Cosine similarity when prior distribution is used

When a prior distribution is added to $\tilde{R}$ as in §12.1.2, the sparsity breaks and the matrix becomes a dense matrix. Therefore it seems that we lose huge advantage in computation efforts when calculating the similarities. However, because the added matrix is rank-one matrix, we can still exploit the sparsity and calculate the similarities at almost the same cost as before.

Assume that $\hat{\theta}_k$ in (12.9) has been calculated and is added to $\tilde{R}$. Thus we have a new rating matrix.

$$\tilde{R}_{\hat{\theta}} = \tilde{R} + \lambda \mathbf{1}_{n_C \times n_I} \, \mathbf{diag}(\hat{\theta}) = \tilde{R} + \lambda \begin{bmatrix} \hat{\theta}_1 \mathbf{1} & \cdots & \hat{\theta}_{n_I} \mathbf{1} \end{bmatrix} \in \mathbf{R}^{n_C \times n_I} \tag{12.12}$$

where $\hat{\theta} = \begin{bmatrix} \hat{\theta}_1 & \cdots & \hat{\theta}_{n_I} \end{bmatrix}^T \in \mathbf{R}^{n_I}$. Here λ plays a similar role as the coefficient for the regularization when we assume that

$$p(\hat{\theta}) \sim N(0, \lambda I_{n_I}). \tag{12.13}$$

Now let $\tilde{d}_1, \ldots, \tilde{d}_{n_I}$ are the column vectors of $\tilde{R}_{\hat{\theta}}$. Then the cosine similarity of ith item and jth item is

$$s(\tilde{d}_i, \tilde{d}_j) = \frac{\tilde{d}_i^T \tilde{d}_j}{\|\tilde{d}_i\|_2 \|\tilde{d}_j\|_2}. \tag{12.14}$$

Now note that

$$\tilde{d}_i^T \tilde{d}_j = (d_i + \lambda \hat{\theta}_i \mathbf{1})^T (d_j + \lambda \hat{\theta}_j \mathbf{1}) = d_i^T d_j + \lambda \hat{\theta}_j \mathbf{1}^T d_i^T + \lambda \hat{\theta}_i \mathbf{1}^T d_j + \lambda^2 \hat{\theta}_i \hat{\theta}_j n_C \tag{12.15}$$

hence

$$\|\tilde{d}_i\|_2^2 = \tilde{d}_i^T \tilde{d}_i = \|d_i\|_2^2 + 2\lambda \hat{\theta}_i \mathbf{1}^T d_i^T + \lambda^2 \hat{\theta}_i^2 n_C \tag{12.16}$$

We can pre-compute $\mathbf{1}^T d_i$ for $i = 1, \ldots, n_I$ which takes less than n_R where n_R refers to the number of nonzero entries in $\tilde{R}$. Hence, the additional computational cost is negligible.

We can also solve this problem at a different angle. In order to calculate the inner projects and the norms in (12.15) and (12.16), we can do the following matrix multiplication.

$$\begin{aligned}
\tilde{R}_{\hat{\theta}}^T \tilde{R}_{\hat{\theta}} &= (\tilde{R} + \lambda \mathbf{1}_{n_C \times n_I} \, \mathbf{diag}(\hat{\theta}))^T (\tilde{R} + \lambda \mathbf{1}_{n_C \times n_I} \, \mathbf{diag}(\hat{\theta})) && (12.17) \\
&= \tilde{R}^T \tilde{R} + \lambda \tilde{R}^T \mathbf{1}_{n_C \times n_I} \, \mathbf{diag}(\hat{\theta}) + \lambda \, \mathbf{diag}(\hat{\theta}) \mathbf{1}_{n_I \times n_C} \tilde{R} && (12.18) \\
&\quad + \lambda^2 \, \mathbf{diag}(\hat{\theta}) \mathbf{1}_{n_I \times n_C} \mathbf{1}_{n_C \times n_I} \, \mathbf{diag}(\hat{\theta}) && (12.19) \\
&= \tilde{R}^T \tilde{R} + \lambda \hat{\theta} \tilde{r}^T + \lambda \tilde{r} \hat{\theta}^T + \lambda^2 n_C \hat{\theta} \hat{\theta}^T && (12.20)
\end{aligned}$$

12.1.3.3 Correlation coefficient similarity

The correlation coefficient is defined by

$$s(d_i, d_j) = \frac{(d_i - \mathbf{1}^T d_i/n_C)^T (d_j - \mathbf{1}^T d_j/n_C)}{\|d_i - \mathbf{1}^T d_i/n_C\|_2 \|d_j - \mathbf{1}^T d_j/n_C\|_2}. \tag{12.21}$$

Again the Jensen's inequality guarantees that $-1 \leq s(d_i, d_j) \leq 1$, but the fact that every entry in $\tilde{R}$ is nonzero does not make this similarity nonnegative.

12.1.4 Data value transformation

When there are popular items across many customers, those values can dominate in the similarity measure evaluation. For example, the aggregate rating matrix, $\bar{R}$, looks like the following:

$$\begin{array}{c} \left[\begin{array}{ccccc} \cdots & 100 & \cdots & 130 & \cdots \\ \cdots & 2 & \cdots & 5 & \cdots \\ \cdots & 3 & \cdots & 3 & \cdots \\ \cdots & 10 & \cdots & 1 & \cdots \end{array}\right] \\ \begin{array}{ccccc} & \uparrow & & \uparrow & \\ & d_i & & d_j & \end{array} \end{array} \tag{12.22}$$

Note here that the scores in the first row are much larger than the other terms. This can be caused by the fact that some customers are way more active than other customers, *e.g.*, they can listen to some musics many times or uses an shopping app very frequently. Now the cosine similarity and the correlation coefficient similarity between d_i and d_j are

$$0.9956 \text{ and } 0.9951 \tag{12.23}$$

respectively. However, if we remove the first customer and recalculate both similarities, it yields

$$0.4611 \text{ and } -0.9176 \tag{12.24}$$

respectively.

Note that both similarity measures give very different measures. The correlation coefficient similarity, which also tells whether both have positive or negative correlations by its sign, gives opposite signs. This shows how *some* customers activity or behavior can change the item-to-item similarities drastically.

However, this doesn't seem to be right. The item-to-time similarities are supposed to represent the nature of the relation among items, hence should not be decided by a small number of extremely active customers.

For this reason, in many recommendation systems literature, it has been reported that the similarity measures mentioned above do not show good performance. To address this issue, various value transformation methods have been introduced. We will discuss some of these methods in the following sections.

12.1.4.1 TFIDF (or tf-idf)

The term frequency–inverse document frequency (TFIDF or tf-idf) is a numerical statistic which has been widely used in the field of information retrieval. It was orginally designed to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. Tf–idf is one of the most popular term-weighting schemes today; 83% of text-based recommender systems in digital libraries use tf–idf.

12.1.4.2 Okapi BM25 transformation

Instead, the Okapi BM25 score has shown much better performance when applied to the entries in $\tilde{R}$ before calculating similarity measures. It is defined by

$$\mathrm{bm}_{25}(i,j) = \log\left(\frac{n_C - \|d_j\|_0 + 0.5}{\|d_j\|_0 + 0.5}\right)\frac{\tilde{R}_{ij}(k_1+1)}{\tilde{R}_{ij} + k_1\left(1 - b + b\frac{\|c_i\|_0}{\bar{c}}\right)} \tag{12.25}$$

where $\|\cdot\|_0$ denotes the number of nonzero entries of a vector and k_1 and b are some parameters usually with $k_1 \in [1.2, 2.0]$ and $b \sim 0.75$.

12.1.5 Recommendation based on item similarities

Now finally, we can evaluate new (menu) item scores for each (menu) items for each customer. Here we suggest two methods.

- For customer i and item j, we calculate new recommendation as

  $$\hat{R}_{ij} = \frac{\sum_{i=1}^{n_I} s(i,j)\tilde{R}_{i,j}}{\sum_{i=1}^{n_I} s(i,j)} \tag{12.26}$$

  where $s(i,j)$ is the cosine similarity between d_i and d_j.

- The second candidate considers the deviation from the average, *i.e.*,

  $$\hat{R}_{ij} = \mathbf{1}^T c_i/n_I + \frac{\sum_{i=1}^{n_I} s(i,j)(\tilde{R}_{ij} - \mathbf{1}^T c_i/n_I)}{\sum_{i=1}^{n_I} s(i,j)} \tag{12.27}$$

  where $s(i,j)$ is the correlation coefficient similarity between d_i and d_j.

These new values are the ones obtained from the history data for that particular customer together with those for neighboring customers. Hence, this will give better recommendation.

12.2 Collaborative Filtering using Matrix Factorization

Here we discuss the collaborative filtering using matrix factorization, which uses latent factor models. There are two types of latent factors; one for customers and one for items.

The customer latent factors represent customers' taste or tendency. In psychological perspective, even customers themselves may not realize these, but they implicitly express these by their activities. They can be

- the degree of pursuing economical life
- the degree of interest in tableware
- the degree of caring children
- the degree of interest in books
- the degree of putting values on family

On the other hand, the item latent factors represent items' attributes. Again, customers may not realize these, but they tend to click on some menus or items according items attributes and their inclination. They can be

- the probability of leading to inexpensive items
- the probability of leading to tableware
- the probability of leading to items related to children
- the probability of leading to book purchase
- the probability of leading to items related to family

We will discuss how we can utilize these latent factors to design our collaborative filtering below. Note, however, that it is generally impossible to identify the actual latent factors a ML algorithm yields with any of the aforementioned factors. These latent factors are *not predefined or decided apriori*, but are learned or revealed through the values coming out of the algorithms. What's more important is that these methods work very well in practice.

We will also describe different types of problem formulations and different approaches to solve them, *e.g.*, stochastic gradient descent (SGD) method, alternating SGD, and alternating least squares (ALS).

12.2.1 Problem definition and formulations

As before, suppose that there are n_C customers and n_I items. We assume that we have a (sparse) rating matrix, $R \in (\mathbf{R} \cup \{\mathrm{NaN}\})^{n_C \times n_I}$, in (12.1). Sometimes we need to deal with the aggregate rating matrix, $\bar{R}(t) \in (\mathbf{R} \cup \{\mathrm{NaN}\})^{n_C \times n_I}$, in (12.3). To simplify notations, we will refer to both types of rating matrix as R.

Now we assume that there are n_L latent factors. This means that every customer n_L latent factors and every item has n_L latent factors.

Let $x_i \in \mathbf{R}^{n_L}$ be a column vector representing n_L latent factors for the ith customer and let $y_j \in \mathbf{R}^{n_L}$ be a column vector representing n_L latent factors for the jth item with

$$x_i = \left[\begin{array}{c} x_{i,1} \\ \vdots \\ x_{i,n_L} \end{array}\right] \in \mathbf{R}^{n_L} \tag{12.28}$$

and

$$y_j = \left[\begin{array}{c} y_{j,1} \\ \vdots \\ y_{j,n_L} \end{array}\right] \in \mathbf{R}^{n_L}. \tag{12.29}$$

Then $x_{i,1}, \ldots, x_{i,n_L}$ are the n_L tendency or taste factors of the ith customer and let $y_{j,1}, \ldots, y_{j,n_L}$ are the n_L attributes of the jth item.

Then we assume that the rating of the j item by the i customer can be inferenced (or estimated) by the sum of the corresponding factors, *i.e.*,

$$R_{i,j} \simeq \hat{R}_{i,j} = x_{i,1}y_{j,1} + \cdots + x_{i,n_L}y_{j,n_L}. \tag{12.30}$$

This is the most critical assumption in the matrix factorization.

The purpose of the collaborative filtering is to find these latent factors so as to make these inference as accurate as possible, *i.e.*, to solve the following optimization problem

$$\text{minimize} \quad \textstyle\sum_{1 \leq i \leq n_C, 1 \leq j \leq n_I : R_{i,j} \in \mathbf{R}} l(R_{i,j}, \hat{R}_{i,j}) \tag{12.31}$$

where the $n_L(n_C + n_I)$ optimization variables are $x_{i,k}$ and $y_{j,k}$ for $1 \leq i \leq n_C$, $1 \leq j \leq n_I$, and $1 \leq k \leq n_L$, and $l : \mathbf{R} \times \mathbf{R} \to \mathbf{R}_+$ is a loss function measuring the distance between the true value and the estimate, hoping that $\hat{R}_{i,j}$ can accurately predict the rating for $1 \leq i \leq n_C$ and $1 \leq j \leq n_I$ where $R_{i,j}$ is not given.

In most cases, we use squared loss for l, so we will also sue the squared loss (except some special cases).

$$l(y_1, y_2) = (y_1 - y_2)^2. \tag{12.32}$$

Now let us come up with more compact notation to describe our problem. Let $X \in \mathbf{R}^{n_C \times n_L}$ be the customer latent factor matrix whose ith row is x_i^T, *i.e.*,

$$X = \left[\begin{array}{c} x_1^T \\ \vdots \\ x_{n_C}^T \end{array}\right] = \left[\begin{array}{cccc} x_{1,1} & x_{1,2} & \cdots & x_{1,n_L} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n_L} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_C,1} & x_{n_C,2} & \cdots & x_{n_C,n_L} \end{array}\right] \in \mathbf{R}^{n_C \times n_L}. \tag{12.33}$$

Likewise, let $Y \in \mathbf{R}^{n_I \times n_L}$ be the item latent factor matrix whose jth row is y_j^T, *i.e.*,

$$Y = \left[\begin{array}{c} y_1^T \\ \vdots \\ y_{n_I}^T \end{array}\right] = \left[\begin{array}{cccc} y_{1,1} & y_{1,2} & \cdots & y_{1,n_L} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,n_L} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n_I,1} & y_{n_I,2} & \cdots & y_{n_I,n_L} \end{array}\right] \in \mathbf{R}^{n_I \times n_L}. \tag{12.34}$$

Note that these two matrices are extremely skinny meaning that we have much more rows then columns in general since n_C could be hundreds of millions and n_I could be hundreds of thousands, but n_L is 100 or so at most.

Now using this notation, we can say

$$\hat{R} = XY^T, \tag{12.35}$$

which is mathematically equivalent to

$$\hat{R}_{i,j} = x_i^T y_j = x_{i,1} y_{j,1} + \cdots + x_{i,n_L} y_{j,n_L}, \tag{12.36}$$

which again is equivalent to (12.30).

Then the optimization problem (12.31) can be rewritten as

$$\text{minimize} \quad \textstyle\sum_{1 \leq i \leq n_C, 1 \leq j \leq n_I : R_{i,j} \in \mathbf{R}} l(R_{i,j}, x_i^T y_j) \tag{12.37}$$

where the optimization variables are $X \in \mathbf{R}^{n_C \times n_L}$ and $Y \in \mathbf{R}^{n_I \times n_L}$.

If we use the squared loss for l, then we can write this equation more compactly as follows.

$$\text{minimize} \quad \|R - XY^T\|_{F,R}^2 \tag{12.38}$$

where $\|Z\|_{F,R}$ refers to Frobenius norm calculated for those entries $Z_{i,j}$ with $R_{i,j} \in \mathbf{R}$.

Note that the number of real variables we need to optimize is the same for all the formulations (12.31), (12.37), and (12.38) (because they are equivalent optimization problems).

12.2.2 Solution methods

12.2.2.1 Matrix factorization via singular value decomposition (SVD)

First we discuss obtaining the latent factors, *i.e.*, X and Y, using sparse singular value decomposition (SVD).

For any rank-k matrix $A \in \mathbf{R}^{m \times n}$, we always have a SVD

$$A = U \Sigma V^T \tag{12.39}$$

where $U \in \mathbf{R}^{m \times k}$, $V \in \mathbf{R}^{n \times k}$, and $\Sigma \in \mathbf{R}^{k \times k}$. (Refer to §12.3.1.1.) The basic idea of obtaining the latent factors using SVD is to apply SVD to the rating matrix.

However, the matrix $R \in (\mathbf{R} \cup \{\text{NaN}\})^{n_C \times n_I}$ can have many unknown values, hence we cannot apply SVD directly. Instead, we employ a certain type of missing data imputation to replace the unknowns with real values, then apply SVD to the matrix. There can be many options for the missing data imputation. Here we list some oft them.

- replacing every unknown with zeros
- replacing unknowns with global item average ratings
- replacing unknowns with global customer average ratings
- use some model based methods (*e.g.*, the item-based collaborative filtering itself can be considered as such a method)

Now suppose that $\tilde{R}$ is R with missing values replaced by proper real values. We compute the largest n_L singular values with corresponding singular vectors. Then we can obtain the approximation of $\tilde{R}$ by

$$\tilde{R} \simeq U_{n_L} \Sigma_{n_L} V_{n_L}^T \tag{12.40}$$

where

$$U_{n_L} = \left[\begin{array}{ccc} u_1 & \cdots & u_{n_L} \end{array} \right] \in \mathbf{R}^{n_C \times n_L} \tag{12.41}$$

$$V_{n_L} = \left[\begin{array}{ccc} v_1 & \cdots & v_{n_L} \end{array} \right] \in \mathbf{R}^{n_I \times n_L} \tag{12.42}$$

and

$$\Sigma_{n_L} = \mathbf{diag}(\sigma_1, \cdots, \sigma_{n_L}) \in \mathbf{R}^{n_L \times n_L}. \tag{12.43}$$

Refer to §12.3.1.1 for more details. This approximation is the best approximation to $\tilde{R}$ in Frobenius norm sense. (Refer to §12.3.1.2 for further details.)

Now since (12.40) can be rewritten as

$$\tilde{R} \simeq U_{n_L} \Sigma_{n_L} V_{n_L}^T = U_{n_L} \Sigma_{n_L}^{1/2} \Sigma_{n_L}^{1/2} V_{n_L}^T = (U_{n_L} \Sigma_{n_L}^{1/2})(V_{n_L} \Sigma_{n_L}^{1/2})^T \tag{12.44}$$

where

$$\Sigma_{n_L}^{1/2} = \mathbf{diag}(\sigma_1^{1/2}, \cdots, \sigma_{n_L}^{1/2}) \in \mathbf{R}^{n_L \times n_L} \tag{12.45}$$

we can obtain the customer and item latent factor matrices as follows.

$$X_{\mathrm{svd}} = U_{n_L} \Sigma_{n_L}^{1/2} = \left[\begin{array}{ccc} \sigma_1^{1/2} u_1 & \cdots & \sigma_{n_L}^{1/2} u_{n_L} \end{array} \right] \in \mathbf{R}^{n_C \times n_L} \tag{12.46}$$

$$Y_{\mathrm{svd}} = V_{n_L} \Sigma_{n_L}^{1/2} = \left[\begin{array}{ccc} \sigma_1^{1/2} v_1 & \cdots & \sigma_{n_L}^{1/2} v_{n_L} \end{array} \right] \in \mathbf{R}^{n_C \times n_L} \tag{12.47}$$

12.2.2.2 Matrix factorization via gradient descent (GD) method

One obvious way to obtain the latent factor matrices is to directly apply gradient descent method to the following optimization problem directly.

$$\text{minimize} \quad f(X, Y) = \textstyle\sum_{i,j: R_{i,j} \in \mathbf{R}} l(R_{i,j}, x_i^T y_j) \tag{12.48}$$

which is equivalent to (12.37). Here we denote the objective function of the optimization problem by $f : \mathbf{R}^{n_C \times n_L} \times \mathbf{R}^{n_I \times n_L} \to \mathbf{R}_+$.

In order to apply gradient descent, we need to evaluate the partial derivative of the objective function with respect to

- $x_{i,k}$ for all $1 \leq k \leq n_L$ and all $i \in \{1 \leq i \leq n_C \mid R_{i,j} \in \mathbf{R}$ for some $1 \leq j \leq n_I\}$
- $y_{j,k}$ for all $1 \leq k \leq n_L$ and all $j \in \{1 \leq j \leq n_I \mid R_{i,j} \in \mathbf{R}$ for some $1 \leq i \leq n_C\}$

For simplicity of the equation derivation, let us assume that

$$\{1 \leq i \leq n_C \mid R_{i,j} \in \mathbf{R} \text{ for some } 1 \leq j \leq n_I\} \quad = \quad \{1, \ldots, n_C\} \tag{12.49}$$

$$\{1 \leq j \leq n_I \mid R_{i,j} \in \mathbf{R} \text{ for some } 1 \leq i \leq n_C\} \quad = \quad \{1, \ldots, n_I\} \tag{12.50}$$

i.e., every item has at least one rating and every customer has at least one rating.

Now the gradient of $f(X,Y)$ with respect to each x_i is

$$\nabla_{x_i} f(X,Y) = \sum_{j:R_{i,j}\in\mathbf{R}} \frac{\partial}{\partial y_2} l(R_{i,j}, x_i^T y_j) y_j \in \mathbf{R}^{n_L} \text{ for } 1 \leq i \leq n_C \tag{12.51}$$

and that with respect to y_j is

$$\nabla_{y_j} f(X,Y) = \sum_{i:R_{i,j}\in\mathbf{R}} \frac{\partial}{\partial y_2} l(R_{i,j}, x_i^T y_j) x_i \in \mathbf{R}^{n_L} \text{ for } 1 \leq j \leq n_I \tag{12.52}$$

If we use the squared loss for l, the optimization problem becomes

$$\text{minimize} \quad f(X,Y) = \|R - XY^T\|_{F,R}^2 \tag{12.53}$$

which is equivalent to (12.38). Then (12.51) and (12.52) imply that the gradients of $f(X,Y)$ can be calculated by

$$\nabla_{x_i} f(X,Y) = -\sum_{j:R_{i,j}\in\mathbf{R}} (R_{i,j} - x_i^T y_j) y_j \in \mathbf{R}^{n_L} \text{ for } 1 \leq i \leq n_C \tag{12.54}$$

and that with respect to y_j is

$$\nabla_{y_j} f(X,Y) = -\sum_{i:R_{i,j}\in\mathbf{R}} (R_{i,j} - x_i^T y_j) x_i \in \mathbf{R}^{n_L} \text{ for } 1 \leq j \leq n_I \tag{12.55}$$

For notational convenience, we stack these vectors to form derivative matrices as below.

If $R \in \mathbf{R}^{n_C \times n_I}$, *i.e.*, there are no missing values, we can write the gradients more compact form, *i.e.*, as the derivatives of X and Y.

$$D_X f(X,Y) = -2(R - XY^T)Y \in \mathbf{R}^{n_C \times n_L} \tag{12.56}$$

and

$$D_Y f(X,Y) = -2(R^T - YX^T)X \in \mathbf{R}^{n_I \times n_L} \tag{12.57}$$

Finally, we describe the gradient descent method.

- choose learning rate strategy $\eta_k > 0$
- choose initial $\tilde{X}$ and $\tilde{Y}$
- let $X_0 := \tilde{X}$ and $Y_0 := \tilde{Y}$
- let $k := 0$
- for each iteration, update X and Y

$$\begin{aligned} X_{k+1} &= X_k - \eta_k D_X f(X_k, Y_k) && (12.58) \\ Y_{k+1} &= Y_k - \eta_k D_Y f(X_k, Y_k) && (12.59) \end{aligned}$$

- stops if certain stopping criterion is satisfied
- update $k := k+1$ and repeat iterations

12.2.2.3 Matrix factorization via alternating gradient descent (GD) method

Instead of updating X and Y simultaneously, we can update X and Y after the other is updated. The algorithm can be described as follows.

- choose learning rate strategy $\eta_k > 0$
- choose initial $\tilde{X}$ and $\tilde{Y}$
- let $X_0 := \tilde{X}$ and $Y_0 := \tilde{Y}$
- let $k := 0$
- for each iteration, update X and Y

$$\begin{aligned} X_{k+1} &= X_k - \eta_k D_X f(X_k, Y_k) && (12.60) \\ Y_{k+1} &= Y_k - \eta_k D_Y f(X_{k+1}, Y_k) && (12.61) \end{aligned}$$

- stops if certain stopping criterion is satisfied
- update $k := k + 1$ and repeat iterations

Note the difference in (12.59) and (12.61). This method updates Y with most recent X values.

12.2.2.4 Matrix factorization via stochastic gradient descent (SGD) method

Theoretically both GD and alternating GD converge (to local minima) with proper learning rate strategies. However, the purpose of the collaborative filtering is not minimize (the sum of) errors (or loss function values), but accurately predict missing ratings, *i.e.*, the ratings that a customer has never given, or sometimes accurately predict the ranking of the items. In this case, what we want to achieve is the solution to the following stochastic optimization problem.

$$\text{minimize} \quad \mathbf{E} \textstyle\sum_{i,j:R_{i,j}=\text{NaN}} l(\tilde{R}_{i,j}, x_i^T y_j) \tag{12.62}$$

where $\mathbf{E}(\cdot)$ denotes the expected value (or mean/average) of a random variable and $\tilde{R}_{i,j}$ denotes the rating that the ith customer would give to jth item in the future.

Unfortunately there is no direct way to solve this problem because this stochastic optimization problem is not (exactly) solvable. Most importantly, we do not have the future data (some of them would be never available). Therefore solving (12.48) is not what we want.

There is also another problem with solving (12.48); computational cost per iteration. The gradient evaluation takes $3n_L n_R$ multiplications and $3n_L n_R$ additions where n_R refers to the number of known ratings. Thus if the density of R is α, the number of multiplications and additions required to evaluate the gradient is $3\alpha n_L n_C n_I$. Therefore, when n_C or n_I or both are huge, the cost for the gradient calculation can be huge.

To resolve these two problems at the same time, we can use stochastic gradient descent (SGD) method with mini-batch methods. The mini-batch method is to use fixed size of training sets for each gradient descent iteration. This can save the computational cost considerably while approximately solving the stochastic optimization problem (12.62).

12.2.2.5 Matrix factorization via alternating least-squares (ALS)

When we use the squared loss, we solve the problem (12.53). The objective function of this problem is

$$f(X,Y) = \|R - XY^T\|_{F,R}^2 \tag{12.63}$$

This is a quadratic function of a bilinear function of X and Y. Unfortunately, it is not a convex function. If it were a convex function, probably we would not need to use (stochastic) gradient descent method because we have much more efficient methods to solve it (*e.g.*, Newton's method). And if it were a convex function, we would be able to obtain the global minimum (even with gradient descent method).

However, if we fix either X or Y, it becomes the convex function of the other. Indeed, the problem (12.53) becomes a least-squares problem when one of X and Y is fixed, which leads to the alternating least-squares algorithm.

- choose initial $\tilde{X}$ and $\tilde{Y}$
- let $X_0 := \tilde{X}$ and $Y_0 := \tilde{Y}$
- let $k := 0$
- for each iteration, update X and Y

$$X_{k+1} = \operatorname*{argmin}_X \|R - X{Y_k}^T\|_{F,R} \tag{12.64}$$

$$Y_{k+1} = \operatorname*{argmin}_Y \|R - X_{k+1}Y^T\|_{F,R} \tag{12.65}$$

- stops if certain stopping criterion is satisfied
- update $k := k + 1$ and repeat iterations

The ALS is named so since solving (12.64) or (12.65) is equivalent to solving a least-squares problem. Note that the functions in (12.64) and (12.65) can be separated by x_is or y_js, *e.g.*, the function in (12.64) is

$$f_Y(X) = \|R - XY^T\|_{F,R}^2 = \sum_{i=1}^{n_C} \|\tilde{r}_i^T - x_i^T Y^T\|_{F,\tilde{r}_i^T}^2 = \sum_{i=1}^{n_C} \|\tilde{r}_i - Yx_i\|_{F,\tilde{r}_i}^2 \tag{12.66}$$

where $\tilde{r}_i \in \mathbf{R}^{n_I}$ are the row vectors of R, thus solving (12.64) is equivalent to solving n_C uncorrelated problems separately. This is another great advantage of ALS since we can use parallelism to save the training time considerably. For example, if we have N processing units, the training time per iteration would be equivalent to that for solving each (small) least-squares n_C/N times. Note that we cannot separate the objective function in (12.63) when we consider X and Y simultaneously since each of n_R terms in (12.63) are intertwined through x_is and y_js.

When $R \in \mathbf{R}^{n_C \times n_I}$, it can be easily shown that

$$x_i^* = Y^\dagger \tilde{r}_i \tag{12.67}$$

where

$$Y^\dagger = (Y^T Y)^{-1} Y^T \tag{12.68}$$

is the pseudo-inverse of Y. Thus (12.64) becomes

$$X_{k+1} = R{Y_k}^{\dagger T} = RY_k\left({Y_k}^TY_k\right)^{-1}. \tag{12.69}$$

Likewise, (12.65) becomes

$$Y_{k+1} = R^TX_{k+1}{}^{\dagger T} = R^TX_{k+1}\left({X_{k+1}}^TX_{k+1}\right)^{-1}. \tag{12.70}$$

Note that in practice, we do not evaluate the pseudo-inverse as in (12.68) because it causes catastrophic numerical instability especially when the matrix is huge (as in most recommendation system cases). Instead, we use QR decomposition, *i.e.*, if $Y = QR$ is the QR decomposition of Y, $Y^\dagger = R^{-1}Q^T$.

We now discuss the interpretation of (12.69) and (12.70). Let X^* and Y^* be the optimal solutions for the problem (12.38). Then these should satisfy (12.64) and (12.65), *i.e.*,

$$\begin{aligned} X^* &= \operatorname*{argmin}_X \|R - X{Y^*}^T\|_{F,R} && (12.71)\\ Y^* &= \operatorname*{argmin}_Y \|R - X^*Y^T\|_{F,R} && (12.72)\end{aligned}$$

Then (12.69) and (12.70) imply that

$$\begin{aligned} X^*{Y^*}^T &= RY^*({Y^*}^TY^*)^{-1}{Y^*}^T = RY^{\mathrm{sim}} && (12.73)\\ &= X^*({X^*}^TX^*)^{-1}{X^*}^TR = X^{\mathrm{sim}}R && (12.74)\end{aligned}$$

where X^{sim} and Y^{sim}

$$\begin{aligned} X^{\mathrm{sim}} &= X^*({X^*}^TX^*)^{-1}{X^*}^T \in \mathbf{R}^{n_C\times n_C} && (12.75)\\ Y^{\mathrm{sim}} &= Y^*({Y^*}^TY^*)^{-1}{Y^*}^T \in \mathbf{R}^{n_I\times n_I} && (12.76)\end{aligned}$$

These yield very interesting interpretations for X^{sim} and Y^{sim}. These equations imply that the optimal prediction for the rating of the jth item by the ith customer is

$$\begin{aligned} {x_i^*}^Ty_j^* &= \sum_{k=1}^{n_I} Y^{\mathrm{sim}}_{k,j}R_{i,k} && (12.77)\\ &= \sum_{k=1}^{n_C} X^{\mathrm{sim}}_{i,k}R_{k,j} && (12.78)\end{aligned}$$

The equation (12.77) tells us that the optimal prediction is the weighted sum of the ith customer's ratings where the weights are determined by Y^{sim}. This is *exactly the same as the prediction by item-based collaborative filtering* where item-to-item similarity matrix is given by Y^{sim}. On the other hand, the equation (12.78) tells us that the optimal prediction is the weighted sum of the jth item ratings where the weights are determined by X^{sim}. This is *exactly the same as the prediction by user-based collaborative filtering* where user-to-user similarity matrix is given by X^{sim}(which we have not covered in this document).

12.2.2.6 Weighted matrix factorization via alternating least-squares (ALS)

Here we consider the weighted norm for the objective function.

$$f_W(X,Y) = \|W \bullet (R - XY^T)\|_{F,R}^2 \tag{12.79}$$

where $W \in \mathbf{R}_+{}^{n_C \times n_I}$ and $\bullet$ denotes the component-wise multiplication.

12.2.3 Collaborative filtering for implicit feedback dataset

- binarized variables

$$p_{i,j} = \left\{ \begin{array}{ll} 1 & R_{i,j} \in \mathbf{R} \\ 0 & R_{i,j} \not\in \mathbf{R} \end{array} \right. \tag{12.80}$$

- confidence variables

$$c_{i,j} = 1 + \alpha R_{i,j} \tag{12.81}$$

or

$$c_{i,j} = 1 + \alpha \log(1 + R_{i,j}/\epsilon) \tag{12.82}$$

- objective function

$$f(X,Y;\lambda) = \sum_{i=1}^{n_C} \sum_{j=1}^{n_I} c_{i,j}(p_{i,j} - x_i^T y_j)^2 + \lambda_X \|X\|_F^2 + \lambda_Y \|Y\|_F^2 \tag{12.83}$$

where $\lambda = (\lambda_X, \lambda_Y) \in \mathbf{R}_{++}{}^2$ is the coefficient for the regularization.

- gradients

$$\begin{array}{rcl} \nabla_{x_i} f(X,Y;\lambda) & = & -2 \sum_{j=1}^{n_I} c_{i,j}(p_{i,j} - x_i^T y_j) y_j + 2\lambda_X x_i \\ & = & 2 \left(\left(\sum_{j=1}^{n_I} c_{i,j} y_j y_j^T + \lambda_X I_{n_L} \right) x_i - \sum_{j=1}^{n_I} c_{i,j} p_{i,j} y_j \right) \\ & = & 2 \left(\left(Y^T \mathbf{diag}(\tilde{c}_i) Y + \lambda_X I_{n_L} \right) x_i - Y^T \mathbf{diag}(\tilde{c}_i) \tilde{p}_i \right) \end{array} \tag{12.84}$$

Likewise,

$$\nabla_{y_j} f(X,Y;\lambda) = 2 \left(\left(X^T \mathbf{diag}(c_j) X + \lambda_Y I_{n_L} \right) y_i - X^T \mathbf{diag}(c_j) p_j \right) \tag{12.85}$$

Here $\tilde{c}_i \in \mathbf{R}^{n_I}$ and $c_j \in \mathbf{R}^{n_C}$ are the ith row vector and the jth column vector of $C \in \mathbf{R}^{n_C \times n_I}$ respectively, and $\tilde{p}_i \in \mathbf{R}^{n_I}$ and $p_j \in \mathbf{R}^{n_C}$ are the ith row vector and the jth column vector of $P \in \mathbf{R}^{n_C \times n_I}$ respectively, *i.e.*,

$$C = \begin{bmatrix} \tilde{c}_1 \\ \tilde{c}_2 \\ \vdots \\ \tilde{c}_{n_C} \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & \cdots & c_{n_C} \end{bmatrix} \in \mathbf{R}^{n_C \times n_I} \tag{12.86}$$

$$P = \begin{bmatrix} \tilde{p}_1 \\ \tilde{p}_2 \\ \vdots \\ \tilde{p}_{n_C} \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & \cdots & p_{n_C} \end{bmatrix} \in \mathbf{R}^{n_C \times n_I} \tag{12.87}$$

12.2.3.1 Regularization coefficient conversion

$$f(X, Y; \lambda) = \sum_{i=1}^{n_C} \sum_{j=1}^{n_I} c_{i,j} (p_{i,j} - x_i^T y_j)^2 + \lambda_X \|X\|_F^2 + \lambda_Y \|Y\|_F^2 \tag{12.88}$$

where $\lambda = (\lambda_X, \lambda_Y) \in {\mathbf{R}_+}^2$.

Suppose that $(X_\lambda^*, Y_\lambda^*)$ is the optimal solution for the following problem:

$$\text{minimize} \quad f(X, Y; \lambda) = \textstyle\sum_{i=1}^{n_C} \sum_{j=1}^{n_I} c_{i,j} (p_{i,j} - x_i^T y_j)^2 + \lambda_X \|X\|_F^2 + \lambda_Y \|Y\|_F^2 \tag{12.89}$$

for some λ. Since for any X, Y, and $a \neq 0$,

$$f(aX, (1/a)Y; (1/a^2)\lambda_X, a^2\lambda_Y) = f(X, Y; \lambda_X, \lambda_Y), \tag{12.90}$$

for any X and Y,

$$\begin{aligned} f(aX_\lambda^*, (1/a)Y_\lambda^*; (1/a^2)\lambda_X, a^2\lambda_Y) &= f(X_\lambda^*, Y_\lambda^*; \lambda_X, \lambda_Y) \\ \leq \quad f((1/a)X, aY; \lambda_X, \lambda_Y) &= f(X, Y; (1/a^2)\lambda_X, a^2\lambda_Y). \end{aligned}$$

Therefore $(aX_\lambda^*, (1/a)Y_\lambda^*)$ is the optimal solution for the following problem:

$$\text{minimize} \quad f(X, Y; (1/a^2)\lambda_X, a^2\lambda_Y) \tag{12.91}$$

Therefore if we obtain an optimal solution for a certain $(\tilde{\lambda}_X, \tilde{\lambda}_Y)$, we have readily obtained optimal solutions for all (λ_X, λ_Y) pairs such that $\lambda_X \lambda_Y = \tilde{\lambda}_X \tilde{\lambda}_Y$.

Therefore solving the problem (12.89) is equivalent to solving the following optimization problem:

$$\text{minimize} \quad f(X, Y; \lambda) = \textstyle\sum_{i=1}^{n_C} \sum_{j=1}^{n_I} c_{i,j} (p_{i,j} - x_i^T y_j)^2 + \sqrt{\lambda_X \lambda_Y} (\|X\|_F^2 + \|Y\|_F^2) \tag{12.92}$$

Now if assume that we have an optimal regularization coefficient $\lambda \in \mathbf{R}_+$ for the following ML problem:

$$\text{minimize} \quad f(X, Y; \lambda) = \textstyle\sum_{i=1}^{n_C} \sum_{j=1}^{n_I} c_{i,j} (p_{i,j} - x_i^T y_j)^2 + \lambda (\|X\|_F^2 + \|Y\|_F^2) \tag{12.93}$$

If we consider the normalization of each of the three terms in the objective function, we can formulate the problem as follows.

$$\text{minimize} \quad f(X,Y;\lambda) = \textstyle\sum_{i=1}^{n_C} \sum_{j=1}^{n_I} c_{i,j}(p_{i,j} - x_i^T y_j)^2 / n_C n_I + \lambda(\|X\|_F^2 / n_C n_L + \|Y\|_F^2 / n_I n_L) \tag{12.94}$$

which is equivalent to

$$\text{minimize} \quad f(X,Y;\lambda) = \textstyle\sum_{i=1}^{n_C} \sum_{j=1}^{n_I} c_{i,j}(p_{i,j} - x_i^T y_j)^2 + \lambda((n_I/n_L)\|X\|_F^2 + (n_C/n_L)\|Y\|_F^2) \tag{12.95}$$

which again is equivalent to

$$\text{minimize} \quad f(X,Y;\lambda) = \textstyle\sum_{i=1}^{n_C} \sum_{j=1}^{n_I} c_{i,j}(p_{i,j} - x_i^T y_j)^2 + (\lambda\sqrt{n_C n_I}/n_L)(\|X\|_F^2 + \|Y\|_F^2) \tag{12.96}$$

Now assume that $\lambda^*(n_C, n_I, n_L)$ is the optimal values for λ in (12.93) (which, for example, can be approximately obtained by hyperparameter optimization with the formulation (12.93)). Then, for some other values $({n_C}', {n_I}', {n_L}')$, (12.96) implies that the approximate optimal λ can be found by

$$\begin{aligned} & \lambda^*(n_C, n_I, n_L)\sqrt{n_C n_I}/n_L = \lambda^*({n_C}', {n_I}', {n_L}')\sqrt{{n_C}'{n_I}'}/{n_L}' \\ \Leftrightarrow \quad & \lambda^*({n_C}', {n_I}', {n_L}') = \lambda^*(n_C, n_I, n_L){n_L}'\sqrt{n_C n_I}/n_L\sqrt{{n_C}'{n_I}'} \end{aligned}$$

So for example, if we have $n_I = {n_I}'$ and $n_L = {n_L}'$, then

$$\lambda^*({n_C}', {n_I}', {n_L}') = \sqrt{\frac{n_C}{{n_C}'}}\lambda^*(n_C, n_I, n_L). \tag{12.97}$$

12.3 Appendix

12.3.1 Linear algebra

12.3.1.1 Singular value decomposition (SVD)

For any rank-k matrix $A \in \mathbf{R}^{m\times n}$, there exist three matrix $U \in \mathbf{R}^{m\times k}$, $\Sigma \in \mathbf{R}^{k\times k}$, and $V \in \mathbf{R}^{n\times k}$, such that

$$A = U\Sigma V^T \tag{12.98}$$

where the columns of U are orthonormal, the columns of V are orthonormal, and Σ is a diagonal matrix with nonincreasing positive diagonal entries, *i.e.*,

$$U^T U = V^T V = I_k \in \mathbf{R}^{k\times k} \tag{12.99}$$

and

$$\Sigma = \mathbf{diag}(\sigma_1, \ldots, \sigma_k) = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \in \mathbf{R}^{k\times k} \tag{12.100}$$

with

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0 \tag{12.101}$$

where I_k referst to k-by-k identity matrix.

If we let $u_1, \ldots, u_k \in \mathbf{R}^n$ be the k column vectors of U, *i.e.*,

$$U = \begin{bmatrix} u_1 & \cdots & u_k \end{bmatrix} \in \mathbf{R}^{n\times k}, \tag{12.102}$$

$U^T U = I_k$ holds if and only if

$$u_i^T u_j = \delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{12.103}$$

where $\delta_{i,j}$ denotes the Kronecker delta, which means the length of each u_i is 1 and all of them are orthogonal to each other. Likewise, if we let $v_1, \ldots, v_k \in \mathbf{R}^m$ be the k column vectors of V, *i.e.*,

$$V = \begin{bmatrix} v_1 & \cdots & v_k \end{bmatrix} \in \mathbf{R}^{m\times k}, \tag{12.104}$$

$V^T V = I_k$ holds if and only if

$$v_i^T v_j = \delta_{i,j} \tag{12.105}$$

which means the length of each v_i is 1 and all of them are orthogonal to each other. Note that

- $\sigma_1, \ldots, \sigma_k$ are called *singular values* of A.
- $u_1, \ldots, u_k$ are called *left singular vectors* of A.
- $v_1, \ldots, v_k$ are called *right singular vectors* of A.

Note also that A can be expressed as

$$A = \sum_{i=1}^{k} \sigma_i u_i v_i^T, \tag{12.106}$$

i.e., A can be express as a linear combination of k rank-1 matrices.

12.3.1.2 Singular value decomposition as rank-k approximation

Given a matrix $A \in \mathbf{R}^{m\times n}$ where $\mathbf{rank}(A) = k$, consider the following optimization problem with $r \leq k$.

$$\begin{array}{ll} \text{minimize} & \|A - B\|_F \\ \text{subject to} & \mathbf{rank}(B) = r \end{array} \qquad (12.107)$$

where the optimization variable is $B \in \mathbf{R}^{m\times n}$ and $\|\cdot\|_F$ denotes the Frobenius norm.

It turns out that

$$B^* = U_r \Sigma_r V_r^T = \sum_{i=1}^r \sigma_i u_i v_i^T, \qquad (12.108)$$

is an optimal solution for the problem (12.107), *i.e.*,

$$\|A - U_r \Sigma_r V_r^T\|_F \leq \|A - B\|_F \qquad (12.109)$$

for every B with $\mathbf{rank}(B) = r$ where $U_r \in \mathbf{R}^{m\times r}$, $V_r \in \mathbf{R}^{n\times r}$, and $\Sigma_r \in \mathbf{R}^{r\times r}$ are defined as

$$\begin{aligned} U_r &= \begin{bmatrix} u_1 & \cdots & u_r \end{bmatrix} \in \mathbf{R}^{n\times r} && (12.110) \\ V_r &= \begin{bmatrix} v_1 & \cdots & v_r \end{bmatrix} \in \mathbf{R}^{m\times r} && (12.111) \\ \Sigma_r &= \mathbf{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbf{R}^{r\times r} && (12.112) \end{aligned}$$

This means the close rank-r matrix to A (in Frobenius norm sense) can be obtained with k largest singular values together with corresponding k left and right singular vectors.

Chapter 13

Time Series Anomaly Detection

Chapter 14

Reinforcement Learning