**Indian Institute of Technology Kanpur**
**CS777: Statistical and Algorithmic Learning Theory**

*Assignment Number:* 3
*Student Name:* Gurpreet Singh
*Roll Number:* 150259
*Date:* April 21, 2018

# QUESTION 1

## Part 1

We have

$$f(\mathbf{x}) \quad = \quad \mathbf{c} + \langle \mathbf{b},\ \mathbf{x} \rangle + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}$$

where $\mathbf{A}$ is symmetric.

The objective $\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}$ would be convex if $\mathbf{A} \succeq \mathbf{0}$. This suggests us that we could add a diagonal matrix with large enough diagonal elements to $\mathbf{A}$ to make the objective convex. In fact, adding the matrix $|\,\lambda_{\min}\,| \cdot \mathbf{I}$ to $\mathbf{A}$ would make the matrix surely semi-positive definitive, since the eigenvalues of this matrix will be greater than $\lambda_{\min} + |\,\lambda_{\min}\,|$. Therefore, we can rewrite the objective as

$$\begin{aligned} f(\mathbf{x}) \quad &= \quad \mathbf{c} + \langle \mathbf{b},\ \mathbf{x} \rangle + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\left(\mathbf{A} + \lambda \cdot \mathbf{I} - \lambda \cdot \mathbf{I}\right)\mathbf{x} \\ &= \quad \mathbf{c} + \langle \mathbf{b},\ \mathbf{x} \rangle + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\left(\mathbf{A} + \lambda \cdot \mathbf{I}\right)\mathbf{x} - \frac{\lambda}{2}\mathbf{x}^{\mathrm{T}}\mathbf{x} \end{aligned}$$

where $\lambda \geq |\,\lambda_{\min}(\mathbf{A})\,|$.

Since adding linear terms does not hurt the convexity of any function, we can write this as

$$f(\mathbf{x}) \quad = \quad g(\mathbf{x}) - h(\mathbf{x})$$

where

$$\begin{aligned} g(\mathbf{x}) \quad &= \quad \mathbf{c} + \langle \mathbf{b},\ \mathbf{x} \rangle + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\left(\mathbf{A} + \lambda \cdot \mathbf{I}\right)\mathbf{x} \\ h(\mathbf{x}) \quad &= \quad \frac{\lambda}{2}\mathbf{x}^{\mathrm{T}}\mathbf{x} \end{aligned}$$

Since both the objectives $g$ and $h$ are convex, we can say that the objective $f$ is a DC function.

## Part 2

The update step of DCA can be written as

$$\mathbf{x}^{t+1} \quad = \quad \arg\min_{\mathbf{x}}\ g(\mathbf{x}) - \left\langle \nabla h(\mathbf{x}^t),\ \mathbf{x} \right\rangle$$

Due to the presence of $\arg\min$ , we are free to add any constants to the above optimization problem. Hence, the above objective is the same as

$$\begin{aligned} \mathbf{x}^{t+1} \quad &= \quad \arg\min_{\mathbf{x}}\ g(\mathbf{x}) - \left\langle \nabla h(\mathbf{x}^t),\ \mathbf{x} \right\rangle + \left\langle \nabla h(\mathbf{x}^t),\ \mathbf{x}^t \right\rangle - h(\mathbf{x}^t) \\ &= \quad \arg\min_{\mathbf{x}}\ g(\mathbf{x}) - \left( h(\mathbf{x}^t) + \left\langle \nabla h(\mathbf{x}^t),\ \mathbf{x} - \mathbf{x}^t \right\rangle \right) \end{aligned}$$

Due to the convexity of the function $h$, we have

$$\forall\, \mathbf{x} \in \mathbb{R}^d, \quad g(\mathbf{x}) - \Big(h(\mathbf{x}^t) + \big\langle \nabla h(\mathbf{x}^t),\ \mathbf{x} - \mathbf{x}^t \big\rangle\Big) \quad \geq \quad g(\mathbf{x}) - h(\mathbf{x})$$

Hence, we could say that the LHS in the above statement is a Majorization of the term on the RHS (the objective function). Hence, we are essentially minimizing the majorization of the objective function in each iteration, which is the idea of the MajMin Algorithm.

## Part 3, 4

Consider the update step of DCA. Since adding a linear term to a convex function does not hurt the convexity of the function, we know that $g(\mathbf{x}) - \big\langle \mathbf{z}^{t+1},\ \mathbf{x} \big\rangle$ is convex, since $g(\mathbf{x})$ is convex.

Also, since $g(\mathbf{x})$ is differentiable at all points, its derivative will exist for all $\mathbf{x} \in \mathbb{R}$ and the differential will be continuous. This suggests that we can differentiate this function and set the derivative to zero in order to compute the optimal point.

Hence, the solution to $\arg\min_{\mathbf{x}}\ g(\mathbf{x}) - \big\langle \mathbf{z}^{t+1},\ \mathbf{x} \big\rangle$ is given by

$$\begin{aligned} \nabla \left\{ g(\mathbf{x}) - \Big\langle \mathbf{z}^{t+1},\ \mathbf{x} \Big\rangle \right\} &= 0 \\ \implies \nabla g(\mathbf{x}) &= \mathbf{z}^{t+1} \end{aligned}$$

This will be true iff there exists a solution for the above expression. Assume there in fact does exist a solution. Then, we can say that the update step of DCA exactly mimics the update step for CCCP.

Therefore, if we initialize both the algorithms at the same starting point, say $\mathbf{x}_1^0$ (for DCA) and $\mathbf{x}_2^0$ (for CCCP) where $\mathbf{x}_1^0 = \mathbf{x}_2^0$, and if at the current time step $t$, the updates / points are equal, *i.e.* $\mathbf{x}_1^t = \mathbf{x}_2^t$, then the subsequent update $\mathbf{x}^{t+1}$ would also be same as long as there exists a solution for the equation $\nabla g(\mathbf{x}) = \nabla h(\mathbf{x}^t)$.

Suppose at some time step, might even be the first, the solution to the above expression does not exist. At such a point, there are only two cases under which this can happen. These cases are discussed below.

**Note.** $\nabla h_i(\mathbf{x})$ represents the $i^{\text{th}}$ coordinate of the derivative of $h$ at point $\mathbf{x}$.

**Case 1** $\exists\, i \in [\, d\,]$ such that $\nabla h_i(\mathbf{x}^t) < \min_{\mathbf{x}} \nabla g_i(\mathbf{x})$

If this is the case, then there will be no solution to $\nabla g(\mathbf{x}) = \nabla h(\mathbf{x}^t)$. However, in this case, the difference of the function diverges as we decrease the value of the $i^{\text{th}}$ coordinate of $\mathbf{x}$. That is, as we decrease the value of $\mathbf{x}_i$, the value of the function $f$ decreases (because of positive gradient), and therefore the minimum is achieved at negative infinity, or we can say no solution exists for the objective function.

**Case 2** $\exists\, i \in [\, d\,]$ such that $\nabla h_i(\mathbf{x}^t) > \max_{\mathbf{x}} \nabla g_i(\mathbf{x})$

This case is similar to the previous case, but instead of the minimum achieved at negative infinity, the minimum would be achieved at positive infinity. However, even in this case, we can conclude that no finite solution exists to the minimization object.

Therefore, the answer for Part 3 would be to conclude that no finite solution exists. This case is also similar to DCA since for DCA, we would keep on iterating until we (theoretically) reach positive or negative infinity, since the gradient's $i^{\text{th}}$ coordinate will remain positive and negative respectively,

no matter what the updates (because of convexity of $h$). Hence, even in that case, our ultimate conclusion would be that no solution exists, or rather no finite solution exists. This also proves that the two algorithms are in fact essentially the same.

**Indian Institute of Technology Kanpur**
**CS777: Statistical and Algorithmic Learning Theory**

*Assignment Number:* 3
*Student Name:* Gurpreet Singh
*Roll Number:* 150259
*Date:* April 21, 2018

# QUESTION 2

A brief sketch of the algorithm is given in Algorithm 1

**Algorithm 1**: A single Pixel Camera

1. **Design Matrix**

   Sample each entry of Design Matrix $\mathbf{A}$ from a Normal Gaussian.

   $$\mathbf{A} = \left[ \mathbf{a}_1{}^{\mathrm{T}}, \mathbf{a}_2{}^{\mathrm{T}} \dots \mathbf{a}_k{}^{\mathrm{T}} \right]^{\mathrm{T}} \sim \mathcal{N}^{k \times d}(0, 1)$$

2. **Sparse Recovery**

   (a) Recover the sparse vector $\hat{\mathbf{x}}$ using the Iterative Hard-Thresholding (IHT) Algorithm.

   (b) Compute the original vector $\mathbf{x}$ as $\mathbf{x} = \mathbf{F}^{\mathrm{T}}\hat{\mathbf{x}}$

For the given algorithm, I have assumed that the matrix $\mathbf{F}$ for Fourier transforming is known or given to us. The details of the algorithm and its convergence analysis is given in the following sections.

1. **Estimating the Weights / Design Matrix**

   The design matrix $\mathbf{A}$ is given as the matrix with the weights $\{\mathbf{a}_1, \mathbf{a}_2 \dots \mathbf{a}_k\}$ as the rows of the matrix. This design matrix is used to compute the encoding of each pixel as required by the system. More specifically, the encoding $\mathbf{y}$ is given as $\mathbf{y} = \mathbf{A}\mathbf{x}$. From the concepts of sparse recovery, we know that a suitable Design Matrix must satisfy the Restricted Isometry Property (RIP), however deterministically generating such a matrix is NP-Hard, therefore, we sample this matrix instead.

   I have used Theorem 7.1 and the settings described in **?**, with the sampling of the design matrix as an independent sampling of each of the entry from a Normal Gaussian, *i.e.* $\forall i \in [k]\, j \in [d]$, $\mathbf{A}_{i,j} \sim \mathcal{N}(0, 1)$.

   From Theorem 7.1 (**?**) and the following details, we have that the matrix sampled, *i.e.* $\mathbf{A}$ satisfies the Restricted Isometry Property (RIP) at order $c$ with constant $\delta$ with high probability, *i.e.* at least $1 - 2\exp(-\Omega(d))$, given $c < k/2$ and $d \geq \Omega\left(\frac{c}{\delta^2}\log\frac{k}{c}\right)$.

2. **Sparse Recovery of the original vectors**

   Assuming we have sampled a design matrix which satisfies the RIP property at order $c$, we can now use the Iterative Hard-Thresholding (Algorithm 2) (**??**) to recover the sparse vectors.

   Using the IHT Algorithm, we can generate the sparse vector $\hat{\mathbf{x}}$ which can be then converted to the original vector using Inverse Fourier Transform (assuming the matrix $\mathbf{F}$) is known.

**Algorithm 2**: Iterative Hard-Thresholding

**Input:** Design Matrix $\mathbf{A}$, encoding $\mathbf{y}$, step length $\eta$ and projection sparsity level $l$

**Output:** Sparse vector $\hat{\mathbf{x}}$

Initialize $\mathbf{x}^0 \leftarrow \mathbf{0}$
**for** $t = 0, 1 \dots T$ **do**
  $\mathbf{z}^{t+1} \leftarrow \mathbf{x}^t - \eta \cdot \mathbf{A}^{\mathrm{T}} \left( \mathbf{A}\mathbf{x}^t - \mathbf{y} \right)$
  $\mathbf{x}^{t+1} \leftarrow \Pi_{\mathcal{B}_0(l)} \left( \mathbf{z}^{t+1} \right)$
**end**
return $\hat{\mathbf{x}} = \mathbf{x}^T$

We can use the standard convergence bound for the IHT Algorithm given in **?** as well as **?**. The bound is stated below.

For a design matrix $\mathbf{A}$ which satisfies RIP at order $c$ and constant $\delta < 1/2$, the IHT Algorithm, if run with $\eta = 1$ and projection sparsity level $l = c/3$ ensures $\left\| \mathbf{x}^t - \mathbf{x} \right\|_2 \leq (2\delta)^t \left\| \mathbf{x}^0 - \mathbf{x} \right\|_2^2$. Therefore, we have guaranteed convergence (for large number of iterations).

From the above analysis, we can say that with probability at least $1 - \exp\left(-\Omega(d)\right)$, and parameters $c$ and $\delta$ which satisfy $c < k/2$ and $d \geq \Omega\left(\frac{c}{\delta^2} \log \frac{k}{c}\right)$, our algorithm, after $T$ steps of the IHT algorithm will output a sparse vector $\hat{\mathbf{x}}$ such that $\| \hat{\mathbf{x}} - \mathbf{F}\mathbf{x} \| \leq (2\delta)^T \left\| \hat{\mathbf{x}}^0 - \mathbf{F}\mathbf{x} \right\|$. We can then approximate the original vector $\mathbf{x} \approx \mathbf{F}^{\mathrm{T}}\hat{\mathbf{x}}$.

**Indian Institute of Technology Kanpur**
**CS777: Statistical and Algorithmic Learning Theory**

*Assignment Number:* 3
*Student Name:* Gurpreet Singh
*Roll Number:* 150259
*Date:* April 21, 2018

# QUESTION 3

---

We want to minimize $h(x) : \mathbb{R}^d \to \mathbb{R}$ where

$$h(\mathbf{x}) \quad = \quad \max\{f(\mathbf{x}), g(\mathbf{x})\}$$

where $f, g : \mathbb{R}^d \to \mathbb{R}$ are convex functions.

**Claim 3.1.** The minimization objective, $h(\mathbf{x})$ is convex.

**Proof.** Consider two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}$, then from the convexity of the functions $f$ and $g$, we have

$$\begin{aligned} \forall\, \alpha \in [\,0, 1\,], \quad f(\alpha \cdot \mathbf{x} + (1-\alpha) \cdot \mathbf{y}) \quad &\leq \quad \alpha \cdot f(\mathbf{x}) + (1-\alpha) \cdot f(\mathbf{y}) \\ g(\alpha \cdot \mathbf{x} + (1-\alpha) \cdot \mathbf{y}) \quad &\leq \quad \alpha \cdot g(\mathbf{x}) + (1-\alpha) \cdot g(\mathbf{y}) \end{aligned}$$

Since $\alpha$ and $1-\alpha$ are positive, we have

$$\begin{aligned} f(\alpha \cdot \mathbf{x} + (1-\alpha) \cdot \mathbf{y}) \quad &\leq \quad \alpha \cdot f(\mathbf{x}) + (1-\alpha) \cdot f(\mathbf{y}) \\ &\leq \quad \alpha \cdot h(\mathbf{x}) + (1-\alpha) \cdot h(\mathbf{y}) \end{aligned}$$

and

$$\begin{aligned} g(\alpha \cdot \mathbf{x} + (1-\alpha) \cdot \mathbf{y}) \quad &\leq \quad \alpha \cdot g(\mathbf{x}) + (1-\alpha) \cdot g(\mathbf{y}) \\ &\leq \quad \alpha \cdot h(\mathbf{x}) + (1-\alpha) \cdot h(\mathbf{y}) \end{aligned}$$

Therefore, we have

$$h(\alpha \cdot \mathbf{x} + (1-\alpha) \cdot \mathbf{y}) \quad \leq \quad \alpha \cdot h(\mathbf{x}) + (1-\alpha) \cdot h(\mathbf{y})$$

□

Since the objective is convex, we can use convex optimization techniques in order to minimize the given objective. I have chosen the method to be subgradient descent, with update step given as

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^t - \eta_t \cdot \mathbf{g}^t$$

where $\mathbf{g}^t \ \in \ \partial h(\mathbf{x}^t)$

Suppose $\partial h(\mathbf{x})$ represents the set of subgradients at point $\mathbf{x} \in \mathbb{R}$, then using the definition of subgradients, $\forall\, \mathbf{g}^t \in \partial h(\mathbf{x}^t)$ we can write

$$h(\mathbf{x}^*) \quad \geq \quad h(\mathbf{x}^t) + \left\langle \mathbf{g}^t,\ \mathbf{x}^* - \mathbf{x}^t \right\rangle$$

We can now use the following properties discussed in class to obtain a bound on $\Phi_t$

**Property 3.1.** For any two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$,

$$\| \mathbf{a} + \mathbf{b} \|_2^2 - \| \mathbf{a} \|_2^2 - \| b \|_2^2 \quad \overset{(a)}{=} \quad 2 \langle a,\ b \rangle \quad \overset{(b)}{=} \quad \| \mathbf{a} \|_2^2 + \| b \|_2^2 - \| \mathbf{a} - \mathbf{b} \|_2^2 \tag{1}$$

Using property 1a, we can write the above inequality as

$$\begin{aligned} h(\mathbf{x}^*) - h(\mathbf{x}^t) &\geq \frac{1}{2\eta_t} \left( \left\| \mathbf{x}^* - \mathbf{x}^t + \eta_t \cdot \mathbf{g}^t \right\|_2^2 - \left\| \mathbf{x}^* - \mathbf{x}^t \right\|_2^2 - \left\| \eta_t \cdot \mathbf{g}^t \right\|_2^2 \right) \\ \implies h(\mathbf{x}^t) - h(\mathbf{x}^*) &\leq \frac{1}{2\eta_t} \left( \left\| \mathbf{x}^* - \mathbf{x}^t \right\|_2^2 + \left\| \eta_t \cdot \mathbf{g}^t \right\|_2^2 - \left\| \mathbf{x}^* - \mathbf{x}^t + \eta_t \cdot \mathbf{g}^t \right\|_2^2 \right) \\ &= \frac{1}{2\eta_t} \left( \left\| \mathbf{x}^* - \mathbf{x}^t \right\|_2^2 + \left\| \eta_t \cdot \mathbf{g}^t \right\|_2^2 - \left\| \mathbf{x}^* - \mathbf{x}^{t+1} \right\|_2^2 \right) \end{aligned}$$

Adding for all $t = 0 \dots T$, we get

$$\begin{aligned} \sum_{t=0}^{T} 2\eta_t \left( h(\mathbf{x}^t) - h(\mathbf{x}^*) \right) &\leq \sum_{t=0}^{T} \left\| \mathbf{x}^* - \mathbf{x}^t \right\|_2^2 + \left\| \mathbf{g}^t \right\|_2^2 - \left\| \mathbf{x}^* - \mathbf{x}^{t+1} \right\|_2^2 \\ &\leq \left\| \mathbf{x}^* - \mathbf{x}^0 \right\|_2^2 + T \left\| \mathbf{g}^t \right\|_2^2 \\ \implies \sum_{t=0}^{T} \Phi_t &\leq \frac{1}{2\eta_t} \left\| \mathbf{x}^* - \mathbf{x}^0 \right\|_2^2 + \frac{\eta_t}{2} \left\| \mathbf{g}^t \right\|_2^2 \end{aligned}$$

Suppose if the subgradients are bounded, say by G, then for constant step length $\eta$, we can say

$$\implies \frac{1}{T} \sum_{t=0}^{T} \Phi_t \quad \leq \quad \frac{1}{2\eta T} \left\| \mathbf{x}^* - \mathbf{x}^0 \right\|_2^2 + \frac{\eta}{2} \cdot \mathrm{G}^2$$

Minimizing RHS with respect to $\eta$, we have

$$\frac{1}{T} \sum_{t=1}^{T} \Phi_t \quad \leq \quad \frac{1}{\sqrt{T}} \left\| \mathbf{x}^* - \mathbf{x}^0 \right\|_2^2 \mathrm{G}^2$$

Note that the subgradients of the function $h$ at point $\mathbf{x} \in \mathbb{R}$ can be given by the subgradients of the functions $f$ or $g$ whichever is maximum at point $\mathbf{x}$. Therefore, at every time step, in order to compute the subgradient, all we need is the subgradient of the maximum of the two functions at the time step, given that the gradients/subgradients of the functions $f$ and $g$ are bounded. This is given more formally below.

Define $\mathcal{S}_f = \left\{ \mathbf{x} \in \mathbb{R} \,\middle|\, h(\mathbf{x}) \quad = \quad f(\mathbf{x}) \right\}$, and similarly define $\mathcal{S}_g$.

**Claim 3.2.** $\forall\, \mathbf{x} \in \mathcal{S}_g,\ \forall\, \mathbf{g} \in \partial g(\mathbf{x}) \implies \mathbf{g} \in \partial h(\mathbf{x})$ and $\forall\, \mathbf{x} \in \mathcal{S}_f,\ \forall\, \mathbf{g} \in \partial f(\mathbf{x}) \implies \mathbf{g} \in \partial h(\mathbf{x})$

**Proof.** At point $\mathbf{x} \in \mathcal{S}_g$, for any subgradient of $g(\mathbf{x})$, $\mathbf{g} \in \partial g(\mathbf{x})$, from the definition of subgradients, we have $\forall\, \mathbf{y} \in \mathbb{R}$,

$$g(\mathbf{y}) \quad \geq \quad g(\mathbf{x}) + \langle \mathbf{g},\ \mathbf{y} - \mathbf{x} \rangle$$

Since $\mathbf{x} \in \mathcal{S}_g$, we know $h(\mathbf{x}) = g(\mathbf{x})$ and from the definition of $h$, we have $\forall\, \mathbf{y} \in \mathbb{R}$, $h(\mathbf{y}) > g(\mathbf{y})$. Therefore, we can say $\mathbf{g}$ is indeed a subgradient.

This can be similarly proved for $f$ and $\mathcal{S}_f$ □

Since $\mathcal{S}_g \bigcup \mathcal{S}_f = \mathbb{R}$, we have the subgradients at all points using the subgradients of $g$ and $f$.

Hence, we have found a convergence bound for the given objective using subgradient descent. More formally, for two convex functions $f, g : \mathbb{R}^d \to \mathbb{R}$ with gradients and subgradients bounded by G, the regret bound using subgradient-descent for the objective $h(\mathbf{x}) = \max\{f(\mathbf{x}), g(\mathbf{x})\}$, using constant step length can be given as

$$\frac{1}{T}\sum_{t=1}^{T}\Phi_t \quad \leq \quad \frac{1}{\sqrt{T}}\left\|\mathbf{x}^* - \mathbf{x}^0\right\|_2^2 \mathrm{G}^2 \tag{2}$$

**Indian Institute of Technology Kanpur**
**CS777: Statistical and Algorithmic Learning Theory**

*Assignment Number:* 3
*Student Name:* Gurpreet Singh
*Roll Number:* 150259
*Date:* April 21, 2018

# QUESTION 4

Suppose if we randomly sample some entries of the matrix $\mathbf{C}$, with the coordinates given by the sample set $\mathcal{S}$. Then we can wish to find a margin such that the sum of the productivity of land on both sides is as equal as possible.

Since the entries of the matrix are sparse, finding the margin is hard directly. What I propose is to complete the matrix (Matrix Completion) using Low-Rank Matrix Factorization, assuming that the matrix is actually low-rank, and then use this "filled" matrix for finding the best "margin". The sketch of the algorithm is given in Algorithm 3.

**Algorithm 3**: Tale of Two Tracts of Land

1. Sample $n$ coordinates uniformly i.i.d. from the complete set of coordinates $[\,L\,] \times [\,L\,]$
2. Find a factorization of the matrix $\mathbf{C} = \mathbf{U}\mathbf{V}^{\mathrm{T}}$ using Alternating Minimization, minimizing the objective

$$\underset{\mathbf{U}\in\mathbb{R}^{L\times k},\mathbf{V}\in\mathbb{R}^{L\times k}}{\arg\min} \sum_{(i,j)\in\mathcal{S}} \left\| \mathbf{U}_i{}^{\mathrm{T}}\mathbf{V}_j - \mathbf{C}_{i,j} \right\|$$

3. Using $\mathbf{U}$ and $\mathbf{V}$, find the best margin which divides the matrix such that there is equal weight on each side of the margin. This is a deterministic step, where we fill the matrix completely and then find the best margin.

The matrix completion objective is solved using Alternating Minimization, as discussed in class, for which the aglgorithm is given in 4.

**Algorithm 4**: AltMin for Matrix Completion

**Input:** The sample data $\mathcal{S}$ and C

**Output:** The estimates $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$, which are the factors of the matrix

Initialize $\mathbf{U}^0$
**for** $t = 0, 1 \dots T$ **do**
  $\mathbf{V}^{t+1} \leftarrow \arg\min_{\mathbf{V} \in \mathbb{R}^{L \times k}} \sum_{(i,j) \in \mathcal{S}} \left\| \mathbf{V}_j{}^{\mathrm{T}} \mathbf{U}_i^t - \mathbf{C}_{i,j} \right\|$
  $\mathbf{U}^{t+1} \leftarrow \arg\min_{\mathbf{V} \in \mathbb{R}^{L \times k}} \sum_{(i,j) \in \mathcal{S}} \left\| \mathbf{U}_i{}^{\mathrm{T}} \mathbf{V}_j^{t+1} - \mathbf{C}_{i,j} \right\|$
**end**
return $\mathbf{U}^{t+1}$, $\mathbf{V}^{t+1}$