lions@epfl

EE-556: MATHEMATICS OF DATA: FROM THEORY TO COMPUTATION
LABORATORY FOR INFORMATION AND INFERENCE SYSTEMS
FALL 2019

EPFL

INSTRUCTOR:
PROF. VOLKAN CEVHER

HEAD TA
THOMAS SANCHEZ

HOMEWORK-4 (FOR WEEKS 13–14)

This homework consists of three parts, and it covers Lectures 9–11 in the course. In the first part, we will motivate the projection free convex optimization by comparing the computational cost of the projection and the linear minimization oracle (lmo). In the second part, we will implement the Frank-Wolfe method for solving the blind deconvolution problem. Finally in the third part, we will use a variant of Frank-Wolfe method to solve the k-means clustering problem.

1 Projection free convex low-rank matrix optimization

Consider the matrix optimization problem:

$$\boldsymbol{X}^\star \in \arg\min_{\boldsymbol{X}} \left\{ f(\boldsymbol{X}) : \operatorname{rank}(\boldsymbol{X}) \leq r,\ \boldsymbol{X} \in \mathbb{R}^{p\times m} \right\},$$

for some convex objective function f, where $r < \min\{p, m\}$ forces the solution to be low-rank. Such an optimization problem appears in many real-world applications—you will see two examples in the second and third parts of this homework. However, the matrix rank function is discrete-valued and non-convex, so solving this optimization problem is hard (NP-hard in general).

Convex relaxation methods are extensively studied for solving low-rank matrix optimization problems. As pointed out in [5], the nuclear norm is an effective convex surrogate to obtain (approximately) low-rank solutions:

$$\boldsymbol{X}^\star \in \arg\min_{\boldsymbol{X}} \left\{ f(\boldsymbol{X}) : \|\boldsymbol{X}\|_* \leq \kappa,\ \boldsymbol{X} \in \mathbb{R}^{p\times m} \right\}.$$

When projection onto the nuclear norm ball is easy to perform, we can solve this problem by projected gradient methods. However, as the problem dimensions increase, the projection becomes a computational bottleneck.

PROBLEM 1.1: PROJECTION ONTO THE NUCLEAR NORM BALL (15 Points)

(a) (10 Points) Let $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ be the singular value decomposition of $\boldsymbol{Z} \in \mathbb{R}^{p\times m}$. Denote the diagonal of $\boldsymbol{\Sigma} \in \mathbb{R}^{s\times s}$ by a vector $\sigma \in \mathbb{R}^s$, where $s = \min\{p, m\}$. Let σ^{ℓ_1} be the projection of σ onto the ℓ_1-norm ball $\{\mathbf{x} : \mathbf{x} \in \mathbb{R}^s, \|\mathbf{x}\|_1 \leq \kappa\}$. Show that the projection of this matrix onto the nuclear norm ball $\mathcal{X} = \{\boldsymbol{X} : \boldsymbol{X} \in \mathbb{R}^{p\times m}, \|\boldsymbol{X}\|_* \leq \kappa\}$ can be computed by projecting σ onto the ℓ_1 norm ball, i.e.,

$$\operatorname{proj}_{\mathcal{X}}(\boldsymbol{Z}) = \boldsymbol{U}\boldsymbol{\Sigma}^{\ell_1}\boldsymbol{V}^\top$$

where $\Sigma^{\ell_1} \in \mathbb{R}^{s\times s}$ denotes the diagonal matrix with diagonal σ^{ℓ_1}.

(Hint: Use Mirsky's inequality: $\|\boldsymbol{X} - \boldsymbol{Z}\|_F \geq \|\boldsymbol{\Sigma}_{\boldsymbol{X}} - \boldsymbol{\Sigma}_{\boldsymbol{Z}}\|_F$, where $\boldsymbol{\Sigma}_{\boldsymbol{X}}, \boldsymbol{\Sigma}_{\boldsymbol{Z}} \in \mathbb{R}^{s\times s}$ are the diagonal matrices of the singular values of $\boldsymbol{X}, \boldsymbol{Z}$ respectively.)

(b) (5 Points) Implement the projection operator as a function called `projNuc`. You can use the provided file `projL1`.

Set $\kappa = 5000$ and measure the computation time of the projection operator with the 100k and 1M MovieLens datasets. You can do this by running the script `Pr11b`, which loads the datasets, constructs the data matrix, and times the evaluation of the projection operator. Write the values you get in your report. Run your script at least for 5 times and report the average timings.

These datasets consist of the ratings given by MovieLens users to movies in a given list. Of course, a user may not rate all of the movies. Modeling the ratings as entries of a low-rank matrix, where one dimension corresponds to different users, and the other corresponds to different movies, the goal of low-rank matrix completion is to predict the users' ratings on unrated movies. This is essential to building a movie recommendation system.

The 100k dataset consists of 100,000 ratings from 1000 users on 1700 movies. The 1M dataset consists of 1 million ratings from 6000 users on 4000 movies. Note that both of these datasets are relatively small; for instance, the famous Netflix competition data consists of 100480507 ratings that 480189 users gave to 17770 movies. Projecting a matrix of this size would require a huge amount of time.

Problem 1.1 shows that projection onto the nuclear norm ball requires computing the singular value decomposition. The computational complexity of the singular value decomposition is $O(\min(m^2p, mp^2))$, and this can easily become a computational bottleneck if m and p are large. This increased the popularity of algorithms that leverage the linear minimization oracle (lmo) instead (e.g., [2, 7]):

$$\text{lmo}_{\mathcal{X}}(\boldsymbol{Z}) = \arg\min_{\boldsymbol{X}\in\mathcal{X}} \langle \boldsymbol{X}, \boldsymbol{Z}\rangle \qquad \text{where} \qquad \langle \boldsymbol{X}, \boldsymbol{Z}\rangle = \text{Tr}(\boldsymbol{Z}^\top \boldsymbol{X}).$$

Note that $\text{lmo}_{\mathcal{X}}(\boldsymbol{Z})$ is not single valued in general. With abuse of terminology, when we say that we compute the lmo, we actually mean that we compute an instance $\boldsymbol{X}$ such that $\boldsymbol{X} \in \text{lmo}_{\mathcal{X}}(\boldsymbol{Z})$.

PROBLEM 1.2: LMO OF NUCLEAR NORM (15 Points)

(a) (10 Points) Show that the $\text{lmo}_{\mathcal{X}}$ when $\mathcal{X}$ is the nuclear norm ball: $\mathcal{X} = \{\boldsymbol{X} : \boldsymbol{X} \in \mathbb{R}^{p\times m}, \|\boldsymbol{X}\|_* \le \kappa\}$ gives the following output:

$$-\kappa \mathbf{u}\mathbf{v}^T \in \text{lmo}_{\mathcal{X}}(\boldsymbol{Z}),$$

where $\mathbf{u}$ and $\mathbf{v}$ are the left and right singular vectors that correspond to the largest singular value of $\boldsymbol{Z}$.

(Hint: By definition $\kappa \mathbf{u}\mathbf{v}^T \in \mathcal{X}$. You just need to show $\langle \boldsymbol{X}, \boldsymbol{Z}\rangle \ge \langle -\kappa \mathbf{u}\mathbf{v}^T, \boldsymbol{Z}\rangle$ for all $\boldsymbol{X} \in \mathcal{X}$.)

(b) (5 Points) Implement the lmo with $\mathcal{X}$ as a function called `lmoNuc`. Set $\kappa = 5000$ and measure the computation time for the 100k and 1M Movielens datasets. You can do so by running the script `Pr12b`, which loads the datasets, constructs the data matrix, and times the evaluation of the lmo.

Write the values you get in your report. Run your script at least for 5 times and report the average timings. Compare these values with the computation time of the projection operator from Problem 1.1.b.

2 Hands-on experiment 1: Crime Scene Investigation with Blind Deconvolution

You are working with the police. You are asked to identify the license plate of a car involved in a crime scene investigation. Unfortunately, the CCTV image of the car is blurry. In this part, we simulate this scene by trying to deblur a plate image found from the internet[1].

This is an instance of the blind deconvolution problem. The set-up of the problem is as follows: Given two unknown vectors $\mathbf{x}, \boldsymbol{w} \in \mathbb{R}^L$, we observe their circular convolution $\mathbf{y} = \boldsymbol{w} * \mathbf{x}$, i.e.,

$$y_\ell = \sum_{\ell'=1}^{L} w_{\ell'} x_{\ell-\ell'+1}$$

where the index $\ell - \ell' + 1$ in the sum above is understood to be modulo L. The goal of blind deconvolution is to separate $\boldsymbol{w}$ and $\mathbf{x}$.

The keyword blind comes from the fact that we do not know much priori information about the signals. We simply assume $\boldsymbol{w}$ and $\mathbf{x}$ belong to *known* subspaces of $\mathbb{R}^L$ of dimension K and N, i.e., we can write

$$\boldsymbol{w} = \boldsymbol{B}\boldsymbol{h}, \quad \boldsymbol{h} \in \mathbb{R}^K$$
$$\mathbf{x} = \boldsymbol{C}\boldsymbol{m}, \quad \boldsymbol{m} \in \mathbb{R}^N$$

for some $L \times K$ matrix $\boldsymbol{B}$ and $L \times N$ matrix $\boldsymbol{C}$. The columns of these matrices provide bases for the subspaces in which $\boldsymbol{w}$ and $\mathbf{x}$ live.

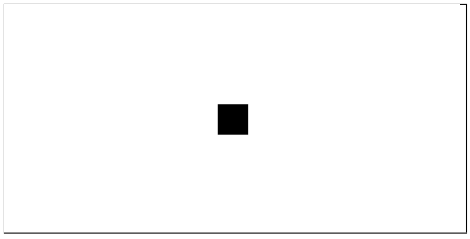
In the image deblurring example, $\mathbf{x}$ corresponds to the image we want to recover, $\boldsymbol{w}$ to a 2D blur kernel. We assume that we know (or have an estimate of) the support (i.e., the location of nonzeros) of the blur kernel. In real applications, support can be estimated by an expert using the physical information such as the distance of object to the focus and the camera, the speed of the camera and/or the object, camera shutter speed, etc.

In this experiment, we use a very rough estimate for the support and simply use a box at the center of the domain. We roughly tuned the size of the box. Interestingly, it is possible to make the plate readable even with this very rough estimate.

The 2D convolution $\mathbf{y} = \boldsymbol{w} * \mathbf{x}$ produces a blurred image. As we have seen in the last homework, natural images have sparse wavelet expansions. Hence, the image $\mathbf{x}$ can be expressed as $\mathbf{x} = \boldsymbol{C}\boldsymbol{m}$ with $\boldsymbol{C}$ is the matrix formed by a subset of the columns of the wavelet transform matrix. In addition, the blur kernel $\boldsymbol{w}$ can be written as $\boldsymbol{w} = \boldsymbol{B}\boldsymbol{h}$ with $\boldsymbol{B}$ is the matrix formed by a subset of the columns of the identity matrix.

[1] https://plus.maths.org/issue37/features/budd/blurredplate.jpg

Blurred image of licence plate $\mathbf{y}$



Estimate of the support of blur kernel $\boldsymbol{w}$

PROBLEM 2.1: FRANK-WOLFE FOR BLIND DECONVOLUTION (25 POINTS)

Let $\boldsymbol{b}$ be the L-point normalized discrete Fourier transform (DFT) of the observation $\mathbf{y}$, i.e, $\boldsymbol{b} = \mathbf{F}\mathbf{y}$, where F is the DFT matrix. Then, $\boldsymbol{b}$ can be written as $\boldsymbol{b} = \boldsymbol{A}(\mathbf{X})$ where $\mathbf{X} = \boldsymbol{h}\boldsymbol{m}^\top$ and $\boldsymbol{A}$ is a linear operator. Explicit expression of this linear operator $\boldsymbol{A}$ is out of the scope of this homework, c.f., [1] for further details. This reformulation allows us to express $\mathbf{y}$, which is a nonlinear combination of the coefficients of $\boldsymbol{h}$ and $\boldsymbol{m}$, as a linear combination of the entries of their outer product $\mathbf{X} = \boldsymbol{h}\boldsymbol{m}^\top$. Note that given $\boldsymbol{B}$ and $\boldsymbol{C}$, recovering $\boldsymbol{m}$ and $\boldsymbol{h}$ from $\boldsymbol{b}$ is the same as recovering $\mathbf{x}$ and $\boldsymbol{w}$ from $\mathbf{y}$.

Since $\mathbf{X}$ is a rank one matrix, we can use the nuclear norm to enforce approximately low-rank solutions. Then, we can formulate the blind deconvolution problem as follows:

$$\boldsymbol{X}^\star \in \arg\min_{\boldsymbol{X}} \left\{ \frac{1}{2}\|\mathbf{A}(\boldsymbol{X}) - \boldsymbol{b}\|_2^2 : \|\boldsymbol{X}\|_* \leq \kappa,\ \boldsymbol{X} \in \mathbb{R}^{p\times m} \right\}. \tag{1}$$

Here $\kappa > 0$ is a tuning parameter.

We will apply the Frank-Wolfe algorithm to solve the optimization problem given in (1). The Frank-Wolfe algorithm is one of the earliest algorithms that avoids the proximal operator. Instead, it leverages the lmo [2]:

$$\text{lmo}(\nabla f(\boldsymbol{Z})) = \arg\min_{\boldsymbol{X}\in\mathcal{X}} \langle \nabla f(\boldsymbol{Z}), \boldsymbol{X} \rangle,$$

where $\mathcal{X} = \{\boldsymbol{X} : \|\boldsymbol{X}\|_* \leq \kappa,\ \boldsymbol{X} \in \mathbb{R}^{p\times m}\}$ as in Part 1. It applies to the generic constrained minimization template with a smooth objective function, $\min_{\boldsymbol{X}}\{f(\boldsymbol{X}) : \boldsymbol{X} \in \mathcal{X}\}$ as follows:

Frank-Wolfe's algorithm
1. Choose $\boldsymbol{X}^0 \in \mathcal{X}$.
2. For $k = 0, 1, \ldots$ perform:
$\begin{cases} \hat{\mathbf{X}}^k & := \text{lmo}(\nabla f(\boldsymbol{X}^k)), \\ \mathbf{X}^{k+1} & := (1-\gamma_k)\mathbf{X}^k + \gamma_k \hat{\mathbf{X}}^k, \end{cases}$
where $\gamma_k := 2/(k+2)$.

(a) (5 Points) Recall that the Frank-Wolfe algorithm applies only for smooth objectives. Show that the objective function is smooth, in the sense its gradient is Lipschitz continuous.

(b) (20 Points) Complete the missing lines in the `FrankWolfe` function to implement the algorithm. We provide you the linear operators that you need to compute the lmo in the code. Note that we do not need to store and use the linear operator $\boldsymbol{A}$ in the ambient dimensions. In fact, for storage and arithmetic efficiency, we should avoid forming $\boldsymbol{A}$. You can find more details about this aspect as comments in the code.

Test your implementation using the script `Test_deblur`. Tune the parameter κ until the license plate number becomes readable. What is the license plate number? You can also tune the support of the blur kernel and try to get better approximations.

3 Hands-on experiment 2: k-means Clustering by Semidefinite Programming

Clustering is an unsupervised machine learning problem in which we try to partition a given data set into k subsets based on distance between data points or similarity among them. The goal is to find k centers and to assign each data point to one of the centers such that the sum of the square distances between them are minimal [4]. This problem is known to be NP-hard.

Problem Definition: Given a set of n points in a $d-$dimensional Euclidean space, denoted by

$$S = \{\mathbf{s}_i = (s_{i1}, \cdots, s_{id})^\top \in \mathbb{R}^d \ \ i = 1, \cdots, n\}$$

find an assignment of the n points into k disjoint clusters $\mathcal{S} = (S_1, \cdots, S_k)$ whose centers are $\mathbf{c}_j (j = 1, \cdots, k)$ based on the total sum of squared Euclidean distances from each point $\mathbf{s}_i$ to its assigned cluster centroid $\mathbf{c}_i$, i.e.,

$$f(S, \mathcal{S}) = \sum_{j=1}^{k} \sum_{i=1}^{|S_j|} \|\mathbf{s}_i^j - \mathbf{c}_j\|^2,$$

where $|S_j|$ is the number of points in S_j, and $\mathbf{s}_i^j$ is the i^{th} point in S_j.

[4] proposes an SDP-relaxation to approximately solve the abovementioned model-free $k-$means clustering problem . The resulting optimization problem[2] takes the standard semidefinite programming form

$$\boldsymbol{X}^\star \in \arg\min_{\boldsymbol{X}} \Big\{ \langle \boldsymbol{C}, \boldsymbol{X} \rangle : \underbrace{\boldsymbol{X}\mathbf{1} = \mathbf{1}}_{A_1(\boldsymbol{X})=b_1}, \underbrace{\boldsymbol{X}^\top \mathbf{1} = \mathbf{1}}_{A_2(\boldsymbol{X})=b_2}, \underbrace{\boldsymbol{X} \geq 0}_{B(\boldsymbol{X})\in\mathcal{K}}, \underbrace{\mathrm{Tr}(\boldsymbol{X}) \leq \kappa,\ \boldsymbol{X} \in \mathbb{R}^{p\times p},\ \mathbf{X} \geq 0}_{\mathcal{X}} \Big\}, \tag{2}$$

where $C \in \mathbb{R}^{p\times p}$ is the Euclidean distance matrix between the data points. $\mathrm{Tr}(\boldsymbol{X}) \leq \kappa$ enforces approximately low-rank solutions, the linear inclusion constraint $\boldsymbol{X} \geq 0$ is element-wise nonnegativity of $\boldsymbol{X}$, the linear equality constraints $\boldsymbol{X}\mathbf{1} = \mathbf{1}$ and $\boldsymbol{X}^\top \mathbf{1} = \mathbf{1}$ require row and column sums of $\boldsymbol{X}$ to be equal to 1's, and $\mathbf{X} \succeq 0$ means that $\mathbf{X}$ is positive semi-definite. Recall that $\mathrm{Tr}(\boldsymbol{X}) = \|\boldsymbol{X}\|_*$ for any positive semi-definite matrix $\boldsymbol{X}$.

Algorithm: Program in (2) can be reformulated as

$$\begin{aligned} \min_{x\in\mathcal{X}} \quad & f(x) + g_1(A_1(x)) + g_2(A_2(x)), \\ \text{subject to} \quad & B(x) \in \mathcal{K} \end{aligned} \tag{3}$$

where $f(x) = \langle \boldsymbol{C}, x \rangle$ is a smooth convex function, g_1 is the indicator function of singleton $\{b_1\}$, g_2 is the indicator function of singleton $\{b_2\}$ and $\mathcal{K}$ is the positive orthant for which computing the projection is easy.

Note that the classical Frank-Wolfe method does not apply to this problem due to nonsmooth terms g_1, g_2. In the sequel, we will attempt to solve this problem with the HomotopyCGM method proposed in [7] to handle the non-smooth problems with a conditional gradient based method. The algorithm to solve the problem in (3) is given below.

Conditional Gradient Method(CGM) for composite problems

1. Choose $x^0 \in \mathcal{X}$ and $\beta_0 > 0$

2. For $k = 1, 2, \ldots$ perform:

$$\begin{cases} \gamma_k & = 2/(k+1), \ \text{ and } \ \beta_k = \beta_0/\sqrt{k+1} \\ v_k & = \beta_k \nabla f(x_k) + A_1^\top (A_1 x_k - b_1) + A_2^\top (A_2 x_k - b_2) + (x_k - \mathrm{proj}_{\mathcal{K}}(x_k)) \\ \hat{x}^k & := \mathrm{argmin}_{x\in\mathcal{X}} \langle v_k, x \rangle \\ x^{k+1} & := (1-\gamma_k) x^k + \gamma_k \hat{x}^k \end{cases}$$

3.Output: x^k

Dataset: We use a similar setup described by [3]. We use the fashion-MNIST data in [6] which is released as a possible replacement for the MNIST handwritten digits . Each data point is a 28x28 grayscale image, associated with a label from 10 classes. Figure 3 depicts 3 row samples from each class. Classes are labeled from 0 to 9. First, we extract the meaningful features from this dataset using a simple 2 layers neural network with a sigmoid activation. Then, we apply neural network to 1000 test samples from the same dataset, which gives us a vector $\mu \in \mathbb{R}^{10}$ where each entry represents the probability being in that class. Then, we form the pairwise distance matrix $\boldsymbol{C}$ by using this probability vectors.[3]

PROBLEM 3.1: CONDITIONAL GRADIENT METHOD FOR CLUSTERING FASHION-MNIST DATA (45 POINTS)

(a) (10 Points) Show that the domain $\mathcal{X} = \{\boldsymbol{X} : \mathrm{Tr}(\boldsymbol{X}) \leq \kappa,\ \boldsymbol{X} \in \mathbb{C}^{p\times p},\ \mathbf{X} \succeq 0\}$ is a convex set.

[2] See section (2) of [4] for details of this relaxation.

[3] In the code, you do not need to worry about any of the processing details mentioned here. You are directly given the matrix $\mathbf{C}$.

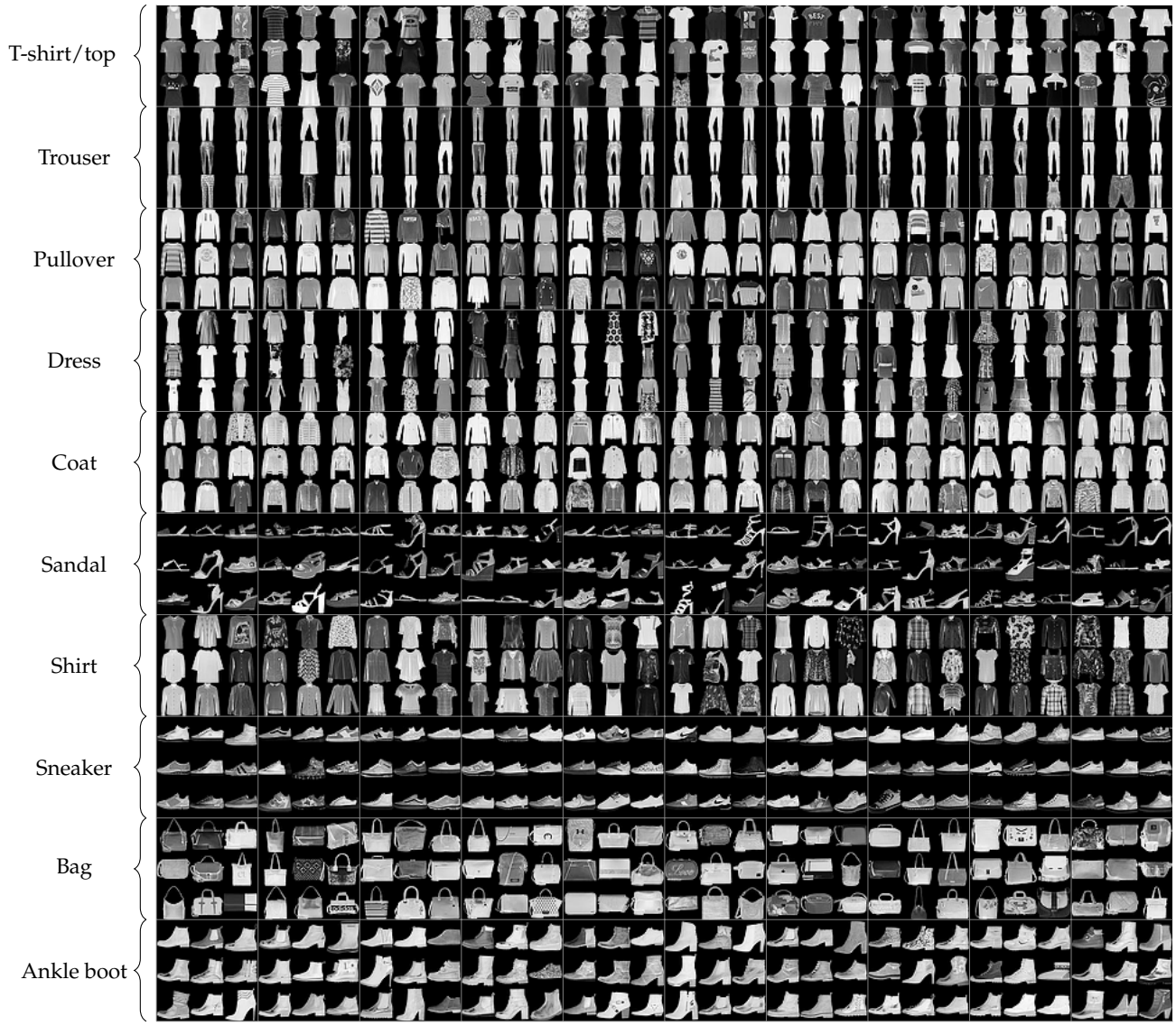


Figure 1: Fashion MNIST data

(b) (10 Points) Given a linear inclusion constraint $Tx \in \mathcal{Y}$, the corresponding quadratic penalty function is given by

$$\mathrm{QP}_{\mathcal{Y}}(x) = \mathrm{dist}^2(Tx, \mathcal{Y}) = \min_{y \in \mathcal{Y}} \|y - Tx\|^2.$$

Write down the constraints in (3) in the quadratic penalty form and **show that** the penalized objective takes the form

$$f(x) + \frac{1}{2\beta}\|A_1(x) - b_1\|^2 + \frac{1}{2\beta}\|A_2(x) - b_2\|^2 + \frac{1}{2\beta}\mathrm{dist}^2(x, \mathcal{K}),$$

and **show that** the gradient of the penalized objective is equal to v_k/β in the algorithm.
(Hint: You can write $\mathrm{dist}^2(Tx, \mathcal{Y}) = \|y^* - Tx\|^2$, where $y^* = \arg\min_{y \in \mathcal{Y}} \|y - Tx\|^2$. and take the derivative with respect to $\mathbf{X}$ without worrying about y^* depending on $\mathbf{X}$, thanks to Danskin's theorem [4].)

(c) (5 Points) Write down v_k explicitly by deriving the gradient and projection specific to Problem (3).

(d) (20 points) Complete the missing lines in the function definition of `HomotopyCGM`, and run the `Test_clustering` to solve the k-means clustering problem. Using the function `value_kmeans`, compute and report the k-means value before and after

[4] https://en.wikipedia.org/wiki/Danskin's_theorem

running the algorithm. Plot the results and SDP solution $\boldsymbol{X}$ after 5000 iterations. What is the final objective value? Is it below the optimal value provided to you? If yes, explain the reason.
(Hint: Note that when X is as given in (2), $\kappa uu^\top \in \mathrm{lmo}_{\mathcal{X}}(X)$, where u is the eigenvector corresponding to the smallest eigenvalue of X.)

4 Guidelines for the preparation and the submission of the homework

Work on your own. Do not copy or distribute your codes to other students in the class. Do not reuse any other code related to this homework. Here are few warnings and suggestions for you to prepare and submit your homework.

- This homework is due at 09:00AM, 6 January, 2020
- Submit your work before the due date. Late submissions are not allowed and you will get 0 point from this homework if you submit it after the deadline.
- Questions of 0 point are for self study. You do not need to answer them in the report.
- Your final report should include detailed answers and it needs to be submitted in PDF format.
- The PDF file can be a scan or a photo. Make sure that it is eligible.
- The results of your simulations should be presented in the final report with clear explanation and comparison evaluation.
- We provide some Python scripts that you can use to implement the algorithms, but you can implement them from scratch using any other convenient programming tool (in this case, you should also write the codes to time your algorithm and to evaluate their efficiency by plotting necessary graphs).
- Even if you use the Python scripts that we provide, you are responsible for the entire code you submit. Apart from completing the missing parts in the scripts, you might need to change some written parts and parameters as well, depending on your implementations.
- The codes should be well-documented and should work properly. Make sure that your code runs without error. If the code you submit does not run, you will not be able to get any credits from the related exercises.
- Compress your codes and your final report into a single ZIP file, name it as `ee556_2019_hw4_NameSurname.zip`, and submit it through the moodle page of the course.

References

[1] AHMED, A., RECHT, B., AND ROMBERG, J. Blind deconvolution using convex programming. *IEEE Transactions on Information Theory 60*, 3 (2014), 1711–1732.

[2] JAGGI, M. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proc. 30th Int. Conf. Machine Learning* (2013).

[3] MIXON, D. G., VILLAR, S., AND WARD, R. Clustering subgaussian mixtures by semidefinite programming. *arXiv preprint arXiv:1602.06612* (2016).

[4] PENG, J., AND WEI, Y. Approximating k-means-type clustering via semidefinite programming. *SIAM journal on optimization 18*, 1 (2007), 186–205.

[5] RECHT, B., FAZEL, M., AND PARRILO, P. A. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev. 52*, 3 (2010), 471–501.

[6] XIAO, H., RASUL, K., AND VOLLGRAF, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[7] YURTSEVER, A., FERCOQ, O., LOCATELLO, F., AND CEVHER, V. A conditional gradient framework for composite convex minimization with applications to semidefinite programming. *arXiv preprint arXiv:1804.08544* (2018).