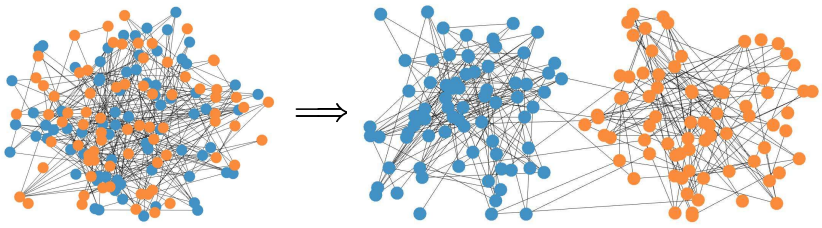


**Figure 5.2** A graph with two densely connected clusters, which can be used to model two tight communities in a social network. Image reprinted with permission from Professor Yuxin Chen of Princeton University.

## 5.2 Robust PCA via Principal Component Pursuit

In each of the above problems, the dataset $\boldsymbol{Y}$ can be modeled as a superposition of a low-rank matrix and a sparse matrix:

$$\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{S}_o. \tag{5.2.1}$$

We like to simultaneously find the low-rank $\boldsymbol{L}_o$ and the sparse $\boldsymbol{S}_o$ from the given $\boldsymbol{Y}$. For the majority of this chapter, we will simplify notation by assuming $\boldsymbol{Y} \in \mathbb{R}^{n \times n}$ is a square matrix. Extensions of both the theory and algorithms to the non-square case $\boldsymbol{Y} \in \mathbb{R}^{n_1 \times n_2}$ are for the most part straightforward, some of which will be discussed in Section 5.3 or left to the reader as exercises.

### 5.2.1 Convex Relaxation for Sparse Low-Rank Separation

Like sparse vector recovery in Chapter 3 and low-rank matrix recovery in Chapter 4, we *might* expect to find efficient algorithms that solve for such well-structured instances. Based on our knowledge from previous chapters, we should have a very clear idea of how to approach this! A natural idea is to solve a problem with two matrix valued variables of optimization, $\boldsymbol{L}$ and $\boldsymbol{S}$, in which we try to make the nuclear norm of $\boldsymbol{L}$ small and the $\ell^1$ norm of $\boldsymbol{S}$ small:

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \\ \text{subject to} & \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}. \end{array} \tag{5.2.2}$$

Here, $\lambda > 0$ is a positive weight parameter. The linear equality constraint $\boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}$ is convex; moreover, since a sum of two convex functions is convex, the objective is also convex. This is a convex program, which we refer to as *Principal Component Pursuit* (PCP).

The relative ease with which we derived this convex relaxation highlights a conceptual advantage of "convex modeling": because convex sets and functions can be combined in nontrivial ways to form new convex sets and functions, it is often straightforward to extend the models to handle new situations of

practical interest. Indeed, although it should be straightforward to write down the optimization problem (5.2.2), this opens the door to many new applications, including those listed in the previous section.

Nevertheless, two crucial questions remain. First, since most of these applications involve large data sets, we will need both efficient and scalable algorithms for solving the problem (5.2.2). Second, to deploy the algorithms with confidence, we will need to understand if and when they correctly recover the target low-rank and sparse components $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$. We will address these questions in Sections 5.2.2 and 5.3, respectively. We will then close the chapter by addressing several additional extensions to problem with both corruptions *and* missing data, which further highlight the flexibility of the framework and allow us to model additional nuisance factors in practical applications.

### 5.2.2 Solving PCP via Alternating Directions Method

The PCP problem can be solved to very high accuracy in polynomial time using semidefinite programming. Classical polynomial time algorithms for SDP are based on interior point methods [GB14], which converge to highly accurate solutions in very few steps, but have a high per-step cost ($O(n^6)$ for a problem involving $n \times n$ matrices). This complexity limits such methods to be practical only for small problems, say with $n < 100$. However, for most aforementioned applications of PCP/RPCA, $n$ can be very large. In such situations, a more appropriate goal is to achieve moderate accuracy with algorithms that are both scalable and efficient. In this section, we sketch one way of achieving this, using the technology of Lagrange duality – in particular, the *alternating directions method of multipliers* (ADMM), which will be studied in more details for general cases in Chapter 8.

The main challenge in efficiently solving the PCP problem is coping with the constraint $\boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}$. As in the previous chapter on matrix completion, we use the machinery of Lagrange duality. Here, the *Lagrangian* is

$$\mathcal{L}(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}) \doteq \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 + \langle \boldsymbol{\Lambda}, \boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y} \rangle . \tag{5.2.3}$$

which is used for characterizing optimality conditions of the constrained program. To derive a practical algorithm, as we will introduce formally in Chapter 8 Section 8.4, it is better to work with the *augmented* Lagrangian:[2]

$$\mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}) \doteq \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 + \langle \boldsymbol{\Lambda}, \boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y} \rangle + \frac{\mu}{2} \|\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y}\|_F^2. \tag{5.2.4}$$

A generic Lagrange multiplier algorithm, like the one we derived for matrix completion, would solve PCP by repeatedly setting

$$(\boldsymbol{L}_{k+1}, \boldsymbol{S}_{k+1}) = \arg\min_{\boldsymbol{L}, \boldsymbol{S}} \mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}_k), \tag{5.2.5}$$

[2] One may also refer to the classic book [Ber82] for a systematic exposition of the augmented Lagrangian method.

**Algorithm 5.1 (Principal Component Pursuit by ADMM)**

1: **Initialize:** $\boldsymbol{S}_0 = \boldsymbol{\Lambda}_0 = 0, \mu > 0$.
2: **while** not converged **do**
3: Compute $\boldsymbol{L}_{k+1} = \mathcal{D}_{1/\mu}(\boldsymbol{Y} - \boldsymbol{S}_k - \mu^{-1}\boldsymbol{\Lambda}_k)$;
4: Compute $\boldsymbol{S}_{k+1} = \mathcal{S}_{\lambda/\mu}(\boldsymbol{Y} - \boldsymbol{L}_{k+1} - \mu^{-1}\boldsymbol{\Lambda}_k)$;
5: Compute $\boldsymbol{\Lambda}_{k+1} = \boldsymbol{\Lambda}_k + \mu(\boldsymbol{L}_{k+1} + \boldsymbol{S}_{k+1} - \boldsymbol{Y})$;
6: **end while**
7: **Output:** $\boldsymbol{L}_\star \leftarrow \boldsymbol{L}_k; \boldsymbol{S}_\star \leftarrow \boldsymbol{S}_k$.

and then updating the Lagrange multipliers (here as a matrix)

$$\boldsymbol{\Lambda}_{k+1} = \boldsymbol{\Lambda}_k + \mu(\boldsymbol{L}_{k+1} + \boldsymbol{S}_{k+1} - \boldsymbol{Y}). \tag{5.2.6}$$

Notice that at each iteration, we need to solve a convex program (5.2.5) with both $\boldsymbol{L}$ and $\boldsymbol{S}$ as unknown. Although this is a convex program, it can be very inefficient to solve with generic algorithms such as subgradient descent. We can avoid doing that by recognizing that the two subproblems: $\min_{\boldsymbol{L}} \mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda})$ and $\min_{\boldsymbol{S}} \mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda})$ both have very simple and efficient solutions.

Let $\mathcal{S}_\tau : \mathbb{R} \to \mathbb{R}$ denote the shrinkage operator

$$\mathcal{S}_\tau[x] = \operatorname{sgn}(x) \max(|x| - \tau, 0),$$

and extend it to matrices by applying it to each element. It is easy to show that

$$\arg\min_{\boldsymbol{S}} \mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}) = \mathcal{S}_{\lambda/\mu}(\boldsymbol{Y} - \boldsymbol{L} - \mu^{-1}\boldsymbol{\Lambda}). \tag{5.2.7}$$

Similarly, for matrices $\boldsymbol{M}$, let $\mathcal{D}_\tau(\boldsymbol{M})$ denote the singular value thresholding operator given by $\mathcal{D}_\tau(\boldsymbol{M}) = \boldsymbol{U}\mathcal{S}_\tau(\boldsymbol{\Sigma})\boldsymbol{V}^*$, where $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ is any singular value decomposition. It is not difficult to show that

$$\arg\min_{\boldsymbol{L}} \mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}) = \mathcal{D}_{1/\mu}(\boldsymbol{Y} - \boldsymbol{S} - \mu^{-1}\boldsymbol{\Lambda}). \tag{5.2.8}$$

Thus, a more practical strategy is to first minimize $\mathcal{L}_\mu$ with respect to $\boldsymbol{L}$ (fixing $\boldsymbol{S}$), then minimize $\mathcal{L}_\mu$ with respect to $\boldsymbol{S}$ (fixing $\boldsymbol{L}$), and then finally update the Lagrange multiplier matrix $\boldsymbol{\Lambda}$ based on the residual $\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y}$ according to (5.2.6). We summarize this strategy as Algorithm 5.1.

As it turns out, the above alternating strategy is a special case of a more general class of augmented Lagrange multiplier methods known as *alternating directions* methods of multipliers (ADMM). We will formally introduce ADMM in Section 8.5 of Chapter 8 and study its convergence and other matters. Algorithm 5.1 performs excellently on a wide range of instances: as we will see below, relatively small numbers of iterations suffice to achieve good relative accuracy. The dominant cost of each iteration is computing $\boldsymbol{L}_{k+1}$ via singular value thresholding. This requires us to compute those singular vectors of $\boldsymbol{Y} - \boldsymbol{S}_k + \mu^{-1}\boldsymbol{\Lambda}_k$ whose corresponding singular values exceed the threshold $\mu$. Empirically, we have observed that the number of such large singular values is often bounded

by $\text{rank}(\boldsymbol{L}_o)$, allowing the next iterate to be computed efficiently via a partial SVD.[3]

Very similar ideas can be used to develop simple and effective augmented Lagrange multiplier algorithms for the robust matrix completion problem (5.6.1) to be introduced in Section 5.6, with similarly good performance.

### 5.2.3 Numerical Simulations and Experiments of PCP

In this section, we perform numerical simulations and experiments of Algorithm 5.1 for PCP and illustrate several of its many applications in image and video analysis. We first investigate its ability to correctly recover matrices of various rank from errors of various density. We then sketch applications in background modeling from video and removing shadows and specularities from face images.

One important implementation detail in PCP is the choice of $\lambda$. As we will see in the next section, theoretical analysis to justify the effectiveness of PCP suggests one natural choice,

$$\lambda = 1/\sqrt{\max(n_1, n_2)},$$

which will be used throughout this section. For practical problems, however, it is often possible to improve performance by choosing $\lambda$ according to prior knowledge about the solution. For example, if we know that $\boldsymbol{S}$ is very sparse, increasing $\lambda$ will allow us to recover matrices $\boldsymbol{L}$ of larger rank. For practical problems, we recommend $\lambda = 1/\sqrt{\max(n_1, n_2)}$ as a good rule of thumb, which can then be adjusted slightly to obtain possibly better results.

*I. Simulation: exact recovery from varying fractions of error.*
We first verify how the algorithm does on recovering randomly generated instances, under favorable conditions (i.e., rank of $\boldsymbol{L}$ is very low and $\boldsymbol{S}$ is rather sparse). We consider square matrices of varying dimension $n = 500, \ldots, 3000$. We generate a rank-$r$ matrix $\boldsymbol{L}_o$ as a product $\boldsymbol{L}_o = \boldsymbol{U}\boldsymbol{V}^*$ where $\boldsymbol{U}$ and $\boldsymbol{V}$ are $n \times r$ matrices with entries independently sampled from a $\mathcal{N}(0, 1/n)$ distribution. $\boldsymbol{S}_o$ is generated by choosing a support set $\mathfrak{S}$ of size $k$ uniformly at random, and setting $\boldsymbol{S}_o = \mathcal{P}_{\mathfrak{S}}[\boldsymbol{E}]$, where $\boldsymbol{E}$ is a matrix with independent Bernoulli $\pm 1$ entries.

Table 5.1 reports the results for a challenging scenario: $\text{rank}(\boldsymbol{L}_o) = 0.05 \times n$ and $k = 0.10 \cdot n^2$. In all cases, we set $\lambda = 1/\sqrt{n}$. Notice that in all cases, solving the convex PCP gives a result $(\hat{\boldsymbol{L}}, \hat{\boldsymbol{S}})$ with the correct rank and sparsity. Moreover, the relative error $\|\hat{\boldsymbol{L}} - \boldsymbol{L}_o\|_F / \|\boldsymbol{L}_o\|_F$ is small, less than $10^{-5}$ in all examples considered.[4]

The last two columns of Table 5.1 give the number of partial singular value

[3] Further performance gains might be possible by replacing this partial SVD with an approximate SVD, as suggested in [GM09] for nuclear norm minimization.

[4] We measure relative error in terms of $\boldsymbol{L}$ only, since we usually view the sparse and low-rank decomposition as recovering a low-rank matrix $\boldsymbol{L}_o$ from gross errors. $\boldsymbol{S}_o$ is of course also well-recovered: in this example, the relative error in $\boldsymbol{S}$ is actually smaller than that in $\boldsymbol{L}$.

| Dim. $n$ | rank($\boldsymbol{L}_o$) | $\|\boldsymbol{S}_o\|_0$ | rank($\hat{\boldsymbol{L}}$) | $\|\hat{\boldsymbol{S}}\|_0$ | $\frac{\|\hat{\boldsymbol{L}}-\boldsymbol{L}_o\|_F}{\|\boldsymbol{L}_o\|_F}$ | # SVD | time(s) |
|---|---|---|---|---|---|---|---|
| 500 | 25 | 25,000 | 25 | 25,000 | $1.2 \times 10^{-6}$ | 17 | 4.0 |
| 1,000 | 50 | 100,000 | 50 | 100,000 | $2.4 \times 10^{-6}$ | 16 | 13.7 |
| 2,000 | 100 | 400,000 | 100 | 400,000 | $2.4 \times 10^{-6}$ | 16 | 64.5 |
| 3,000 | 150 | 900,000 | 150 | 900,000 | $2.5 \times 10^{-6}$ | 16 | 191.0 |

**Table 5.1 Correct Recovery for Random Problems of Varying Sizes.** Here, $\boldsymbol{L}_o = \boldsymbol{U}\boldsymbol{V}^* \in \mathbb{R}^{n\times n}$ with $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{n\times r}$; $\boldsymbol{U}, \boldsymbol{V}$ have entries sampled from i.i.d. $\mathcal{N}(0, 1/n)$. $\boldsymbol{S}_o \in \{-1, 0, 1\}^{n\times n}$ has support chosen uniformly at random and independent random signs; $\|\boldsymbol{S}_o\|_0$ is the number of nonzero entries in $\boldsymbol{S}_o$. In all cases, the rank of $\boldsymbol{L}_o$ and $\ell^0$-norm of $\boldsymbol{S}_o$ are correctly estimated. Moreover, the number of partial singular value decompositions (# SVD) required to solve PCP is almost constant.

decompositions computed in the course of the optimization (# SVD) as well as the total computation time.[5] As we see from Algorithm 5.1, the dominant cost in solving the convex program comes from computing one partial SVD per iteration. Strikingly, in Table 5.1, the number of SVD computations is nearly constant regardless of dimension, and in all cases less than 17, suggesting that the ADMM algorithm gives a reasonably practical solver for PCP.

*II. Experiment: background modeling from surveillance video.*
Video is a natural candidate for low-rank modeling, due to the correlation between frames. One of the most basic algorithmic tasks in video surveillance is to estimate a good model for the background variations in a scene. This task is complicated by the presence of foreground objects: in busy scenes, every frame may contain some anomaly. Moreover, the background model needs to be flexible enough to accommodate changes in the scene, for example due to varying illumination. In such situations, it is natural to model the background variations as approximately low rank. Foreground objects, such as cars or pedestrians, generally occupy only a fraction of the image pixels and hence can be treated as sparse errors.

We investigate whether the convex PCP program can separate these sparse errors from the low-rank background. Here, it is important to note that the error support may not be well-modeled as Bernoulli: errors tend to be spatially coherent, and more complicated models such as Markov random fields may be more appropriate [CSD+09, ZWMM09]. Hence, our theorems do not necessarily guarantee the algorithm will succeed with high probability. Nevertheless, as we will see, PCP still gives visually appealing solutions to this practical low-rank and sparse separation problem, without using any additional information about the spatial structure of the error.

[5] This experiment was performed in Matlab on a Mac Pro with dual quad-core 2.66 GHz Intel Xenon processors and 16 GB RAM.

(a) Original frames (b) Low-rank $\hat{\boldsymbol{L}}$ (c) Sparse $\hat{\boldsymbol{S}}$

**Figure 5.3** Background modeling from video. Three frames from a 200 frame video sequence taken in an airport [LHGT04]. (a) Frames of original video $\boldsymbol{Y}$. (b)-(c) Low-rank $\hat{\boldsymbol{L}}$ and sparse components $\hat{\boldsymbol{S}}$ obtained by PCP.

We consider two example videos introduced in [LHGT04]. The first is a sequence of 200 grayscale frames taken in an airport. This video has a relatively static background, but significant foreground variations. The frames have resolution $176 \times 144$; we stack each frame as a column of our matrix $\boldsymbol{Y} \in \mathbb{R}^{25,344 \times 200}$. We decompose $\boldsymbol{Y}$ into a low-rank term and a sparse term by solving the convex PCP problem (5.2.2) with $\lambda = 1/\sqrt{n_1}$. Figure 5.3(a) shows three frames from the video; (b) and (c) show the corresponding columns of the low rank matrix $\hat{\boldsymbol{L}}$ and sparse matrix $\hat{\boldsymbol{S}}$ (its absolute value is shown here). Notice that $\hat{\boldsymbol{L}}$ correctly recovers the background, while $\hat{\boldsymbol{S}}$ correctly identifies the moving pedestrians. One person appearing in the images in $\hat{\boldsymbol{L}}$ does not move throughout the video, hence it was (correctly) modeled as part of the static background.

We have noticed that the number of iterations for the real data is typically higher than that of the simulations with random matrices given in Table 5.1. The reason for this discrepancy might be that the structures of real data could slightly deviate from the idealistic low-rank and sparse model. Nevertheless, it is important to realize that practical applications such as video surveillance often provide additional information about the signals of interest, e.g. the support of the sparse foreground is spatially piecewise contiguous and temporarily continuous among frames. Or they even impose additional requirements, e.g. the

recovered background needs to be non-negative etc. We note that the simplicity of our objective and solution suggests that one can easily incorporate additional constraints and more accurate models of the signals so as to obtain much more efficient and accurate solutions.

*III. Experiment: removing imperfections from face images.*
Face recognition is another problem domain in computer vision where low-dimensional linear models have received a great deal of attention. This is mostly due to the work of Basri and Jacobs, who showed that for convex, Lambertian objects, images taken under distant illumination lie approximately in a nine-dimensional linear subspace known as the *harmonic plane* [BJ03]. However, since faces are neither perfectly convex nor Lambertian, real face images often violate this low-rank model, in part due to cast shadows and specularities. These errors may be large in magnitude but sparse in the spatial domain. It is reasonable to believe that if we have enough images of the same face, PCP will be able to remove these errors. As with the previous example, some caveats apply: the theoretical result suggests the performance should be good, but does not guarantee it, since again the error support may not follow a Bernoulli model. Nevertheless, as we will see, the results are visually striking.

Figure 5.4 shows face images of one subject taken from the Extended Yale Face Database B [GBK01]. Here, each image has resolution $192 \times 168$; and there are a total of 58 illuminations per subject, which we stack as columns of our matrix $\boldsymbol{Y} \in \mathbb{R}^{32,256 \times 58}$. We again solve PCP with $\lambda = 1/\sqrt{n_1}$.

Figure 5.4 plots the low-rank term $\hat{\boldsymbol{L}}$ and the magnitude of the sparse term $\hat{\boldsymbol{S}}$ obtained as the solution to the convex program. The sparse term $\hat{\boldsymbol{S}}$ compensates for cast shadows and specular regions. In one example (bottom row of Figure 5.4 left), this term also compensates for errors in image acquisition. These results may be useful for conditioning the training data for face recognition, as well as face alignment and tracking under illumination variations.

*IV. Simulation: phase transition in rank and sparsity.*
The above simulations and experiments suggest that for well-structured problem instances (datasets that indeed admit a low-rank and sparse decomposition $\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{S}_o$), PCP accurately recovers both $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$. With this as motivation, we next systematically investigate the ability of the algorithm to recover matrices of varying rank from errors of varying sparsity. We consider square matrices of dimension $n_1 = n_2 = 400$. We generate low-rank matrices $\boldsymbol{L}_o = \boldsymbol{U}\boldsymbol{V}^*$ with $\boldsymbol{U}$ and $\boldsymbol{V}$ independently chosen $n \times r$ matrices with i.i.d. Gaussian entries of mean zero and variance $1/n$. For our first experiment, we assume a Bernoulli model for the support of the sparse term $\boldsymbol{S}_o$, with random signs: each entry of $\boldsymbol{S}_o$ takes on value 0 with probability $1 - \rho_s$, and values $\pm 1$ each with probability $\rho_s/2$. For each $(r, \rho_s)$ pair, we generate 10 random problem instances, each of which is

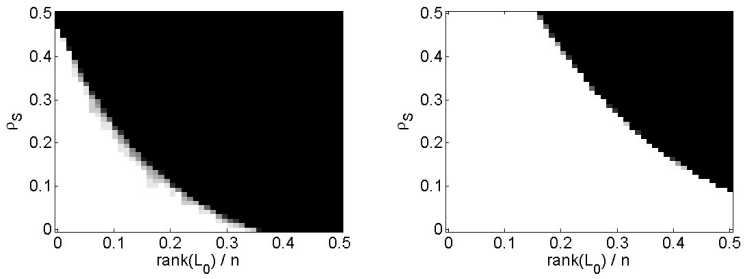
(a) $\boldsymbol{Y}$ (b) $\hat{\boldsymbol{L}}$ (c) $\hat{\boldsymbol{S}}$

**Figure 5.4** Removing shadows, specularities, and saturations from face images. (a) Cropped and aligned images of a person's face under different illuminations from the Extended Yale Face Database B [GBK01]. The size of each image is $192 \times 168$ pixels, a total of 58 different illuminations per person. (b) Low-rank approximation $\hat{\boldsymbol{L}}$ recovered by convex programming. (c) Sparse error $\hat{\boldsymbol{S}}$ corresponding to specularities in the eyes, shadows around the nose region, or brightness saturations on the face. Notice in the bottom left that the sparse term also compensates for errors in image acquisition.

solved via the ADMM Algorithm 5.1. We declare a trial to be successful if the recovered $\hat{\boldsymbol{L}}$ satisfies $\|\hat{\boldsymbol{L}} - \boldsymbol{L}_o\|_F / \|\boldsymbol{L}_o\|_F \leq 10^{-3}$.

Figure 5.5 (left) plots the fraction of correct recoveries in grey scale for each pair $(r, \rho_s)$. Notice that there is a large white region in which the recovery is exact. This inspires us to characterize the working conditions of the algorithm in more precise terms in the next section. The simulation already highlights an interesting aspect of PCP: The recovery is correct even though in some cases $\|\boldsymbol{S}_o\|_F \gg \|\boldsymbol{L}_o\|_F$ (e.g., for $r/n = \rho_s$, $\|\boldsymbol{S}_o\|_F$ is $\sqrt{n} = 20$ times larger!). As we shall see in the next section, this is to be expected from the analysis (see Lemma 5.4): The optimal solution to PCP is unique and correct only depending on the signs and support of $\boldsymbol{S}_o$ and the orientation of the singular spaces of $\boldsymbol{L}_o$.

Finally, inspired by the connection between matrix completion and Robust PCA, we compare the breakdown point of PCP for the low-rank and sparse separation problem to the breakdown behavior of the nuclear-norm heuristic for matrix completion (studied in the previous chapter). By comparing the two heuristics, we can begin to answer the question *how much is gained by knowing*




(a) Principal Component Pursuit (b) Matrix Completion

**Figure 5.5 Correct Recovery for Varying Rank and Sparse Corruptions (left) or Missing Entries (right).** Fraction of correct recoveries across 10 trials, as a function of $\text{rank}(\boldsymbol{L}_o)$ (x-axis) and sparsity of $\boldsymbol{S}_o$ (y-axis). Here, $n_1 = n_2 = 400$. In all cases, $\boldsymbol{L}_o$ is a product of independent $n \times r$ i.i.d. $\mathcal{N}(0, 1/n)$ matrices. Trials are considered successful if $\|\hat{\boldsymbol{L}} - \boldsymbol{L}_o\|_F / \|\boldsymbol{L}_o\|_F < 10^{-3}$. Left: low-rank and sparse decomposition, in which the signs of the sparse matrix $\boldsymbol{\Sigma}_o = \text{sign}(\boldsymbol{S}_o)$ are random. Right: matrix completion. For matrix completion, $\rho_s$ is the probability that an entry is omitted from the observation.

*the location* $\mathfrak{S}$ *of the corrupted entries*? Here, we again generate $\boldsymbol{L}_o$ as a product of Gaussian matrices. However, we now provide the algorithm with only an incomplete subset $\boldsymbol{M} = \mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{L}_o]$ of its entries. Each $(i, j)$ may be included in $\mathfrak{S}$ independently with probability $1 - \rho_s$, so rather than a probability of error, here, $\rho_s$ stands for the probability that an entry is omitted.

We solve the nuclear norm minimization problem

$$\text{minimize} \quad \|\boldsymbol{L}\|_* \quad \text{subject to} \quad \mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{L}] = \mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{M}]$$

using an augmented Lagrange multiplier algorithm very similar to the one discussed in the above section. We again declare $\boldsymbol{L}_o$ to be successfully recovered if $\|\hat{\boldsymbol{L}} - \boldsymbol{L}_o\|_F / \|\boldsymbol{L}_o\|_F < 10^{-3}$. Figure 5.5 (right) plots the fraction of correct recoveries for varying $r, \rho_s$. Notice that nuclear norm minimization successfully recovers $\boldsymbol{L}_o$ over a wider range of $(r, \rho_s)$. The difference between breakdown points can be viewed as the price of not knowing ahead of time which entries are unreliable.

## 5.3 Identifiability and Exact Recovery

Simulations and real examples of the previous section reveals a similar phenomenon of RPCA to that of Matrix Completion: when the solution is sufficiently structured (i.e. sufficient low-rank and sparse), the convex relaxation (and the associated algorithm) succeeds. Our next goal will be to understand this phenomenon at a more mathematical level and to provide a theory that delineates when the convex optimization PCP solves the RPCA problem correctly.

### 5.3.1 Identifiability Conditions

At first sight, the RPCA problem (5.1.1) of separating a matrix into a low-rank one and a sparse one may seem impossible to solve. In general, there is not enough information to perfectly disentangle the low-rank and the spare components since the number of unknowns to infer $\boldsymbol{L}_o \in \mathbb{R}^{n\times n}$ and $\boldsymbol{S}_o \in \mathbb{R}^{n\times n}$ is twice as many as the observations given in $\boldsymbol{Y} \in \mathbb{R}^{n\times n}$. Clearly, we will need both $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$ to be well structured, in the sense that $\boldsymbol{L}_o$ is sufficiently low-rank, and $\boldsymbol{S}_o$ is sufficiently sparse.

However, identifiability issues arise even for very structured examples. For instance, suppose the matrix $\boldsymbol{Y}$ is equal to $\boldsymbol{e}_1\boldsymbol{e}_1^*$ (this matrix has a one in the top left corner and zeros everywhere else). Then since $\boldsymbol{Y}$ is both sparse and low-rank, how can we decide whether it is low-rank or sparse? To make the problem meaningful, we need to impose that the low-rank component $\boldsymbol{L}_o$ is *not* sparse so it can be differentiated from $\boldsymbol{S}_o$.[6]

*Incoherence Conditions on $\boldsymbol{L}_o$.*
In the matrix completion problem of the previous chapter (Section 4.4), we have introduced the notion of $\nu$-incoherence to ensure that a low-rank matrix is not too sparse. Let us write the singular value decomposition of $\boldsymbol{L}_o \in \mathbb{R}^{n\times n}$ as

$$\boldsymbol{L}_o = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^*,$$

where $r$ is the rank of the matrix, $\sigma_1, \dots, \sigma_r$ are the positive singular values, and $\boldsymbol{U} = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_r]$, $\boldsymbol{V} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_r]$ are the matrices of left- and right-singular vectors. Then according to (4.4.13) and (4.4.14), $\boldsymbol{L}_o$ is $\nu$-incoherent

$$\max_i \|\boldsymbol{e}_i^*\boldsymbol{U}\|_2^2 \le \frac{\nu r}{n}, \quad \max_j \|\boldsymbol{e}_j^*\boldsymbol{V}\|_2^2 \le \frac{\nu r}{n}. \tag{5.3.1}$$

For technical reasons that we will see later in our derivation, in low-rank and sparse separation we need a stronger notion of incoherence than the one that sufficed for matrix completion. In addition to the above two incoherence conditions, we further require:

$$\|\boldsymbol{U}\boldsymbol{V}^*\|_\infty \le \frac{\sqrt{\nu r}}{n}. \tag{5.3.2}$$

Here and below, $\|\boldsymbol{M}\|_\infty = \max_{i,j} |\boldsymbol{M}_{ij}|$, i.e. is the $\ell^\infty$ norm of $\boldsymbol{M}$ viewed as a long vector. This incoherence condition asserts that for small values of $\nu$, the singular vectors are reasonably spread out. As it turns out, the above condition is not just needed for technical reasons, its necessity can also be justified from the complexity conjecture regarding the planted clique problem, as one can see through Exercises 5.4 to 5.5.

[6] In Chapter 15, we will study the case when a matrix is simultaneously low-rank and sparse, when the goal is to recover it as a whole instead of separating low-rank and sparse components.

Regardless, one can show that the above incoherence conditions are not atypical, in that they hold with high probability for low-rank matrices that are generated with random orthogonal factors $\boldsymbol{U}$ and $\boldsymbol{V}$.

*Randomness of $\boldsymbol{S}_o$.*
Another identifiability issue arises if the sparse matrix has low rank. This will occur if, say, all the nonzero entries of $\boldsymbol{S}_o$ occur in a column or in a few columns. Suppose for instance, that the first column of $\boldsymbol{S}_o$ is the opposite of that of $\boldsymbol{L}_o$, and that all the other columns of $\boldsymbol{S}_o$ vanish. Then it is clear that we would not be able to recover $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$ by any method whatsoever since $\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{S}_o$ would have a column space equal to, or included in that of $\boldsymbol{L}_o$. To avoid such meaningless situations, we may assume that the sparsity pattern of the sparse component $\boldsymbol{S}_o$ is selected independently and identically according to a Bernoulli distribution

$$\mathfrak{S} \sim \mathrm{Ber}(\rho_s).$$

Under this model, the expected number of nonzero entries in $\boldsymbol{S}_o$ is $\mathbb{E}\left[|\mathfrak{S}|\right] = \rho_s \cdot n^2$.

*Uniqueness.*
The incoherence conditions are sufficient to ensure that we will not confuse the low-rank matrix $\boldsymbol{L}_o$ with the sparse matrix $\boldsymbol{S}_o$. However, they do not yet give a tractable algorithm for recovering them from the sum $\boldsymbol{L}_o + \boldsymbol{S}_o$. One natural approach is to seek a pair $(\boldsymbol{L}^*, \boldsymbol{S}^*)$ that is in some sense the *simplest*. In our context, we would desire $\boldsymbol{L}^*$ to have the lowest possible rank and $\boldsymbol{S}^*$ the sparsest. Or more precisely, we wish to minimize certain measure of "simplicity" or "compactness" that encourages a decomposition such that $\boldsymbol{L}$ is low-rank and $\boldsymbol{S}$ is sparse. Thus, if the ground truth is such that the rank of $\boldsymbol{L}_o$ is low enough and $\boldsymbol{S}_o$ is sparse enough, then they will be the only optimal solution that minimizes such a measure. In this section, we will try to show that for a properly chosen $\lambda \in \mathbb{R}_+$

$$\|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1$$

is precisely such a measure of model simplicity.

Similar to the case with recovering a sparse vector (Chapter 3) or with recovering a low-rank matrix (Chapter 4), could we expect that under reasonable conditions, the above convex program PCP can actually recover the correct low-rank $\boldsymbol{L}_o$ and sparse $\boldsymbol{S}_o$?

In fact, under the minimal conditions discussed in the identifiability section above, the solution to the convex PCP program exactly recovers the low-rank and sparse components, provided that the rank of $\boldsymbol{L}_o$ is not too large and $\boldsymbol{S}_o$ is reasonably sparse. To be more precise, the following statement is true:

THEOREM 5.3 (Principal Component Pursuit). *Suppose $\boldsymbol{L}_o$ is $n \times n$ and obeys (5.3.1)–(5.3.2). Suppose that the support $\mathfrak{S}$ of $\boldsymbol{S}_o$ follows the Bernoulli model with parameter $\rho < \rho_s$, and that the signs of the nonzero entries of $\boldsymbol{S}_o$ are chosen*

*independently from the uniform distribution on* $\{\pm 1\}$. *Then, there is a numerical constant* $C$ *such that with probability at least* $1 - Cn^{-10}$ *(over the choice of signs and support of* $\boldsymbol{S}_o$*), PCP (5.2.2) with* $\lambda = 1/\sqrt{n}$ *is* exact, *i.e.* $\hat{\boldsymbol{L}} = \boldsymbol{L}_o$ *and* $\hat{\boldsymbol{S}} = \boldsymbol{S}_o$, *provided that*

$$\operatorname{rank}(\boldsymbol{L}_o) \leq C_r \frac{n}{\nu \log^2 n}. \tag{5.3.3}$$

*Above,* $C_r$ *and* $\rho_s$ *are positive numerical constants.*

### 5.3.2 Correctness of Principal Component Pursuit

In this section, we prove Theorem 5.3. This section can be skipped for first time readers who are not theory oriented or are not strongly interested in the techniques needed for a rigorous proof of the theorem.

*Dual Certificates for Optimality.*
As for each optimization problem we have encountered thus far, we begin by writing down an optimality condition. To prove that the target pair $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ is the unique optimal solution to the convex program, we then must prove that under our assumptions, this condition is satisfied with high probability.

The key tool for obtaining optimality conditions is the KKT conditions of convex optimization; these conditions are naturally phrased in terms of the subdifferential of the objective function. Recall the subdifferential of the $\ell^1$ norm

$$\partial \|\cdot\|_1 (\boldsymbol{S}_o) = \{\boldsymbol{\Sigma}_o + \boldsymbol{F} \mid \mathcal{P}_{\mathfrak{S}}[\boldsymbol{F}] = \boldsymbol{0},\ \|\boldsymbol{F}\|_\infty \leq 1\}, \tag{5.3.4}$$

and the nuclear norm

$$\partial \|\cdot\|_* (\boldsymbol{L}_o) = \{\boldsymbol{U}\boldsymbol{V}^* + \boldsymbol{W} \mid \mathcal{P}_{\mathsf{T}}[\boldsymbol{W}] = \boldsymbol{0},\ \|\boldsymbol{W}\| \leq 1\}. \tag{5.3.5}$$

Here, $\boldsymbol{U}$ and $\boldsymbol{V}$ are matrices of left and right singular vectors of $\boldsymbol{L}_o$, corresponding to nonzero singular values, and

$$\mathsf{T} \doteq \left\{\boldsymbol{U}\boldsymbol{R}^* + \boldsymbol{Q}\boldsymbol{V}^* \mid \boldsymbol{R}, \boldsymbol{Q} \in \mathbb{R}^{n \times r}\right\}$$

is the tangent space to the variety of rank $r$ matrices at $\boldsymbol{L}_o$.

To the optimization problem

$$\min_{\boldsymbol{L}, \boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}, \tag{5.3.6}$$

associate a matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{n \times n}$ of Lagrange multipliers, and the Lagrangian

$$\mathcal{L}(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}) = \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 + \langle \boldsymbol{\Lambda}, \boldsymbol{Y} - \boldsymbol{L} - \boldsymbol{S} \rangle. \tag{5.3.7}$$

The KKT conditions imply that $(\boldsymbol{L}_\star, \boldsymbol{S}_\star)$ are optimal if there exists $\boldsymbol{\Lambda}$ such that $\boldsymbol{0} = \partial_{\boldsymbol{L}} \mathcal{L}(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda})$ *and* $\boldsymbol{0} \in \partial_{\boldsymbol{S}} \mathcal{L}(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda})$. Thus,

$$\boldsymbol{\Lambda} \in \partial \|\cdot\|_* (\boldsymbol{L}_\star) \quad \text{and} \quad \boldsymbol{\Lambda} \in \lambda \partial \|\cdot\|_1 (\boldsymbol{S}_\star). \tag{5.3.8}$$

To show optimality, *it is enough to find a matrix* $\mathbf{\Lambda}$ *that is in both the subdifferential of the nuclear norm, and the subdifferential of the* $\ell^1$ *norm, at the same time.*

*From the KKT Conditions to Usable Optimality Conditions.*
Although the KKT conditions are a useful guide, the form that we have derived is neither strong enough nor robust enough for our purposes. We need to strengthen them to guarantee *unique* optimality, so that we can eventually ensure that the true pair $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ is the only solution to the PCP problem. Moreover, as in matrix completion, it will be easier to demonstrate that a modified condition is satisfied, in which we merely guarantee that there exists $\mathbf{\Lambda}$ which is *close to* the two subdifferentials, rather than lying exactly within them.

We introduce a simple condition for the pair $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ to be the unique optimal solution to PCP. These conditions, given in the following lemma, are stated in terms of a dual vector, the existence of which certifies optimality.

LEMMA 5.4 (Unique Optimality). *Assume that* $\|\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\| < 1$ *or equivalently* $\mathfrak{S} \cap \mathsf{T} = \{\mathbf{0}\}$. *Then* $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ *is the unique optimal solution to the PCP problem if there exists* $\mathbf{\Lambda}$ *such that*

$$\textit{[Subdifferential of } \|\cdot\|_*\textit{]:} \quad \mathcal{P}_{\mathsf{T}}[\mathbf{\Lambda}] = \boldsymbol{U}\boldsymbol{V}^*, \quad \|\mathcal{P}_{\mathsf{T}^\perp}[\mathbf{\Lambda}]\| < 1, \tag{5.3.9}$$

*and*

$$\textit{[Subdifferential of } \lambda\|\cdot\|_1\textit{]:} \quad \mathcal{P}_{\mathfrak{S}}[\mathbf{\Lambda}] = \lambda\mathbf{\Sigma}_o, \quad \|\mathcal{P}_{\mathfrak{S}^c}[\mathbf{\Lambda}]\|_\infty < \lambda. \tag{5.3.10}$$

There are two aspects of this lemma which deserve comment. First, compared to the KKT condition, it has the extra requirement that $\|\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\| < 1$. This condition means that the subspace of matrices supported on $\mathfrak{S}$ does not intersect the tangent space $\mathsf{T}$ to the low-rank matrices at $\boldsymbol{L}_o$. Second, compared to the KKT condition, which just requires that $\mathbf{\Lambda}$ lie in the subdifferentials of $\|\cdot\|_*$ and $\lambda\|\cdot\|_1$, this condition requires that $\mathbf{\Lambda}$ lie within the *relative interiors* of these two sets, by requiring $\|\mathcal{P}_{\mathsf{T}^\perp}[\mathbf{\Lambda}]\|$ to be strictly less than one and $\|\mathcal{P}_{\mathfrak{S}^c}[\mathbf{\Lambda}]\|_\infty$ to be strictly less than $\lambda$. Under these stronger conditions, we can guarantee that $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ is the *unique* optimal solution to the PCP problem.

*Proof* We consider a feasible perturbation $(\boldsymbol{L}_o + \boldsymbol{H}, \boldsymbol{S}_o - \boldsymbol{H})$ and show that the objective increases whenever $\boldsymbol{H} \neq \mathbf{0}$, hence proving that $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ is the unique optimal solution. To do this, let $\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}] = \bar{\boldsymbol{U}}\bar{\mathbf{\Sigma}}\bar{\boldsymbol{V}}^*$ denote the reduced singular value decomposition of $\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]$. Set $\boldsymbol{W} \doteq \bar{\boldsymbol{U}}\bar{\boldsymbol{V}}^* \in T^\perp$, and notice that

$$\langle \boldsymbol{W}, \boldsymbol{H}\rangle = \langle \boldsymbol{W}, \mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\rangle = \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_*. \tag{5.3.11}$$

Note further that $\boldsymbol{U}\boldsymbol{V}^* + \boldsymbol{W} \in \partial\|\cdot\|_*(\boldsymbol{L}_o)$.

Similarly, set $\boldsymbol{F} \doteq -\operatorname{sign}(\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}])$, notice that $\lambda(\mathbf{\Sigma}_o + \boldsymbol{F}) \in \partial\lambda\|\cdot\|_1(\boldsymbol{S}_o)$, and that $-\lambda\langle \boldsymbol{F}, \boldsymbol{H}\rangle = \lambda\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_1$.

Using the subgradient inequality for both $\|\cdot\|_*$ and $\lambda\|\cdot\|_1$, we obtain that

$$\begin{aligned}
&\|\boldsymbol{L}_o + \boldsymbol{H}\|_* + \lambda\|\boldsymbol{S}_o - \boldsymbol{H}\|_1 \\
&\geq \|\boldsymbol{L}_o\|_* + \lambda\|\boldsymbol{S}_o\|_1 + \langle \boldsymbol{U}\boldsymbol{V}^* + \boldsymbol{W}, \boldsymbol{H}\rangle - \lambda\langle \boldsymbol{\Sigma}_o + \boldsymbol{F}, \boldsymbol{H}\rangle \\
&= \|\boldsymbol{L}_o\|_* + \lambda\|\boldsymbol{S}_o\|_1 + \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_* + \lambda\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_1 + \langle \boldsymbol{U}\boldsymbol{V}^* - \lambda\boldsymbol{\Sigma}_o, \boldsymbol{H}\rangle, \\
&= \|\boldsymbol{L}_o\|_* + \lambda\|\boldsymbol{S}_o\|_1 + \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_* + \lambda\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_1 + \langle \mathcal{P}_{\mathsf{T}}[\boldsymbol{\Lambda}] - \lambda\mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}], \boldsymbol{H}\rangle \\
&= \|\boldsymbol{L}_o\|_* + \lambda\|\boldsymbol{S}_o\|_1 + \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_* + \lambda\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_1 + \langle \mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{\Lambda}] - \lambda\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{\Lambda}], \boldsymbol{H}\rangle \\
&\geq \|\boldsymbol{L}_o\|_* + \lambda\|\boldsymbol{S}_o\|_1 + \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_* + \lambda\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_1 \\
&\quad - \quad \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{\Lambda}]\|\,\|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_* - \lambda\,\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{\Lambda}]\|_\infty\,\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_1 \\
&\geq \|\boldsymbol{L}_o\|_* + \lambda\,\|\boldsymbol{S}_o\|_1 + (1-\beta)\left\{\|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_* + \lambda\,\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_1\right\},
\end{aligned}$$

where $\beta = \max\left\{\|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{\Lambda}]\|, \lambda^{-1}\,\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{\Lambda}]\|_\infty\right\} < 1$. Since by assumption, $\mathfrak{S}\cap\mathsf{T} = \{\boldsymbol{0}\}$, we have $\|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_* + \lambda\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_1 > 0$ unless $\boldsymbol{H} = \boldsymbol{0}$. □

This lemma gives a sufficient condition for $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ to be the *unique* optimal solution. It is still challenging to work with, because it demands that $\boldsymbol{\Lambda}$ is an element of both $\partial\,\|\cdot\|_*(\boldsymbol{L}_o)$ and $\partial\lambda\,\|\cdot\|_1(\boldsymbol{S}_o)$. This forces $\boldsymbol{\Lambda}$ to exactly satisfy the equalities $\mathcal{P}_{\mathsf{T}}[\boldsymbol{\Lambda}] = \boldsymbol{U}\boldsymbol{V}^*$ and $\mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}] = \lambda\boldsymbol{\Sigma}_o$. As we did for matrix completion, it will be helpful to state a modified optimality condition, which accepts $\boldsymbol{\Lambda}$ that satisfied these equalities *approximately*. We state this new condition as follows:

LEMMA 5.5. *Assume* $\|\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\| \leq 1/2$ *and* $\lambda < 1$. *Then with the same notation,* $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ *is the unique solution if there exists* $\boldsymbol{\Lambda}$ *such that*

$$\textit{[Approx. subgradient of } \|\cdot\|_*\textit{]:} \quad \|\mathcal{P}_{\mathsf{T}}[\boldsymbol{\Lambda}] - \boldsymbol{U}\boldsymbol{V}^*\|_F \leq \tfrac{\lambda}{8}, \quad \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{\Lambda}]\| < \tfrac{1}{2}, \tag{5.3.12}$$

*and*

$$\textit{[Approx. subgradient of } \lambda\|\cdot\|_1\textit{]:} \quad \|\mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}] - \lambda\boldsymbol{\Sigma}_o\|_F \leq \tfrac{\lambda}{8}, \quad \|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{\Lambda}]\|_\infty < \tfrac{\lambda}{2}. \tag{5.3.13}$$

*Proof* Consider any nonzero $\boldsymbol{H} \in \mathbb{R}^{n\times n}$. We demonstrate that in a particular sense, $\boldsymbol{H}$ cannot be simultaneously concentrated on $\mathsf{T}$ and $\mathfrak{S}$. Observe that

$$\begin{aligned}
\|\mathcal{P}_{\mathfrak{S}}[\boldsymbol{H}]\|_F &\leq \|\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}[\boldsymbol{H}]\|_F + \|\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_F \\
&\leq \tfrac{1}{2}\|\boldsymbol{H}\|_F + \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_F \\
&\leq \tfrac{1}{2}\|\mathcal{P}_{\mathfrak{S}}[\boldsymbol{H}]\|_F + \tfrac{1}{2}\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_F + \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_F,
\end{aligned}$$

and, therefore,

$$\|\mathcal{P}_{\mathfrak{S}}[\boldsymbol{H}]\|_F \leq \|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_F + 2\|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_F.$$

Symmetric reasoning establishes that

$$\|\mathcal{P}_{\mathsf{T}}[\boldsymbol{H}]\|_F \leq \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_F + 2\,\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_F\,. \tag{5.3.14}$$

With these observations in hand, we proceed in a similar spirit to the proof of

Lemma 5.4. Notice that

$$\begin{aligned} \boldsymbol{U}\boldsymbol{V}^* &= \mathcal{P}_{\mathsf{T}}[\boldsymbol{\Lambda}] + (\boldsymbol{U}\boldsymbol{V}^* - \mathcal{P}_{\mathsf{T}}[\boldsymbol{\Lambda}]) \\ &= \boldsymbol{\Lambda} - \mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{\Lambda}] + (\boldsymbol{U}\boldsymbol{V}^* - \mathcal{P}_{\mathsf{T}}[\boldsymbol{\Lambda}]), \qquad (5.3.15) \\ \lambda\boldsymbol{\Sigma}_o &= \mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}] + (\lambda\boldsymbol{\Sigma}_o - \mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}]) \\ &= \boldsymbol{\Lambda} - \mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{\Lambda}] + (\lambda\boldsymbol{\Sigma}_o - \mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}]), \qquad (5.3.16) \end{aligned}$$

and so

$$\boldsymbol{U}\boldsymbol{V}^* - \lambda\boldsymbol{\Sigma}_o = -\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{\Lambda}] + \mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{\Lambda}] \ + \ (\boldsymbol{U}\boldsymbol{V}^* - \mathcal{P}_{\mathsf{T}}[\boldsymbol{\Lambda}]) \ + \ (\lambda\boldsymbol{\Sigma}_o - \mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}]).$$

Following the proof of Lemma 5.4, we have

$$\begin{aligned} &\|\boldsymbol{L}_o + \boldsymbol{H}\|_* + \lambda\|\boldsymbol{S}_o - \boldsymbol{H}\|_1 \\ &\geq \|\boldsymbol{L}_o\|_* + \lambda\|\boldsymbol{S}_o\|_1 + \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_* + \lambda\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_1 + \langle \boldsymbol{U}\boldsymbol{V}^* - \lambda\boldsymbol{\Sigma}_o, \boldsymbol{H}\rangle \\ &\geq \|\boldsymbol{L}_o\|_* + \lambda\|\boldsymbol{S}_o\|_1 + \tfrac{1}{2}\big(\|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_* + \lambda\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_1\big) - \tfrac{\lambda}{8}\|\mathcal{P}_{\mathsf{T}}[\boldsymbol{H}]\|_F - \tfrac{\lambda}{8}\|\mathcal{P}_{\mathfrak{S}}[\boldsymbol{H}]\|_F \\ &\geq \|\boldsymbol{L}_o\|_* + \lambda\|\boldsymbol{S}_o\|_1 + \underbrace{\big(\tfrac{1}{2} - \tfrac{\lambda}{8} - \tfrac{\lambda}{4}\big)}_{\geq 1/8}\|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}]\|_* + \underbrace{\big(\tfrac{\lambda}{2} - \tfrac{\lambda}{4} - \tfrac{\lambda}{8}\big)}_{\geq \lambda/8}\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}]\|_1 \\ &> \|\boldsymbol{L}_o\|_* + \lambda\|\boldsymbol{S}_o\|_1, \qquad (5.3.17) \end{aligned}$$

where the final (strict) inequality holds because $\boldsymbol{H} \neq \boldsymbol{0}$ and $\mathfrak{S} \cap \mathsf{T} = \{\boldsymbol{0}\}$. □

*Showing that the Optimality Conditions Can be Satisfied.*
We next show that under our conditions, with high probability the conditions of Lemma 5.5 can be satisfied. To do this, we have to show two things:

1 $\|\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\| < 1/2$;
2 existence of a near dual certificate $\boldsymbol{\Lambda}$ as in Lemma 5.5.

Let $\Omega = \mathfrak{S}^c$. These are the *clean* entries. Notice that if $\mathfrak{S} \sim \mathrm{Ber}(\rho_s)$, $\Omega \sim \mathrm{Ber}(1-\rho_s)$. We are going to show 1 and 2 by building on machinery developed in Chapter 4 for *matrix completion*. In particular, in that chapter we showed that if

$$\rho_{\mathrm{clean}} = 1 - \rho_s > C_0 \frac{\nu r \log n}{n}, \qquad (5.3.18)$$

with high probability

$$\big\|\mathcal{P}_{\mathsf{T}} - \rho_{\mathrm{clean}}^{-1}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}^c}\mathcal{P}_{\mathsf{T}}\big\| \ < \ \tfrac{1}{8}. \qquad (5.3.19)$$

Under this condition,

$$\begin{aligned} \|\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\| &= \|\mathcal{P}_{\mathsf{T}} - \mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}^c}\mathcal{P}_{\mathsf{T}}\| \\ &\leq \|\rho_{\mathrm{clean}}\mathcal{P}_{\mathsf{T}} - \mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}^c}\mathcal{P}_{\mathsf{T}}\| + \|(1-\rho_{\mathrm{clean}})\mathcal{P}_{\mathsf{T}}\| \\ &= \rho_{\mathrm{clean}}\big\|\mathcal{P}_{\mathsf{T}} - \rho_{\mathrm{clean}}^{-1}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}^c}\mathcal{P}_{\mathsf{T}}\big\| + 1 - \rho_{\mathrm{clean}} \\ &\leq \tfrac{\rho_{\mathrm{clean}}}{8} + 1 - \rho_{\mathrm{clean}} \\ &< \tfrac{1}{4}, \qquad (5.3.20) \end{aligned}$$

provided $\rho_{\text{clean}} > \frac{6}{7}$. This implies that

$$\|\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\| \;=\; \|\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\|^{1/2} \;\leq\; \tfrac{1}{2}. \tag{5.3.21}$$

This establishes statement 1. By exactly the same reasoning, for any constant $\sigma > 0$, there exists a constant $\rho_{\text{clean},\star}(\sigma) < 1$ such that if $\rho_{\text{clean}} > \rho_{\text{clean},\star}$, with high probability $\|\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\| \;<\; \sigma$.

*Constructing the Certificate* $\boldsymbol{\Lambda}$.
To show that $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ is the unique optimal solution, we further need to establish statement 2. That is, there exists a matrix $\boldsymbol{\Lambda}$ that is simultaneously close to the subdifferential $\partial\|\cdot\|_*(\boldsymbol{L}_o)$ and the subdifferential $\partial\lambda\|\cdot\|_1(\boldsymbol{S}_o)$ as in Lemma 5.5.

In the previous paragraph, we saw that the clean entries $\Omega = \mathfrak{S}^c$ are distributed as a Bernoulli subset, with parameter

$$\rho_{\text{clean}} \doteq 1 - \rho_s. \tag{5.3.22}$$

This is exactly the same model of randomness as in our analysis of matrix completion! We use this fact as a starting point for our construction. Proposition 4.31 implies that as long as the rank of $\boldsymbol{L}_o$ is not too large, i.e.,

$$r < \frac{\rho_{\text{clean}} n}{C_0 \nu \log^2 n}, \tag{5.3.23}$$

with high probability there exists a matrix $\boldsymbol{\Lambda}_{\boldsymbol{L}}$ supported only on the clean set $\Omega$ satisfying

*1* $\|\mathcal{P}_{\mathsf{T}}[\boldsymbol{\Lambda}_{\boldsymbol{L}}] - \boldsymbol{U}\boldsymbol{V}^*\|_F \leq \frac{1}{4n}$,
*2* $\|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{\Lambda}_{\boldsymbol{L}}]\| \leq \frac{1}{4}$,
*3* $\|\boldsymbol{\Lambda}_{\boldsymbol{L}}\|_\infty < \frac{C\log n}{\rho_{\text{clean}}}\|\boldsymbol{U}\boldsymbol{V}^*\|_\infty$.

This certificate $\boldsymbol{\Lambda}_{\boldsymbol{L}}$ lies close enough to the subdifferential of the nuclear norm. Furthermore, let us further verify that it satisfies the condition $\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{\Lambda}_{\boldsymbol{L}}]\|_\infty < \frac{\lambda}{2}$ in Lemma 5.5. This is because $\boldsymbol{U}\boldsymbol{V}^*$ is $\nu$-incoherent: from (5.3.2), $\|\boldsymbol{U}\boldsymbol{V}^*\|_\infty \leq \frac{\sqrt{\nu r}}{n}$ and with the assumption on the rank $r$ of the matrix $\boldsymbol{L}_o$, we have

$$\|\boldsymbol{\Lambda}_{\boldsymbol{L}}\|_\infty < \frac{C\log n}{\rho_{\text{clean}}}\frac{\sqrt{\nu r}}{n} \leq \frac{C}{\sqrt{C_0\rho_{\text{clean}}\nu}}\frac{1}{\sqrt{n}} = \frac{C}{\sqrt{\rho_{\text{clean}}C_0\nu}}\lambda. \tag{5.3.24}$$

By properly choosing the constant $C_0$ and $C$ we can ensure that the coefficient $\frac{C}{\sqrt{\rho_{\text{clean}}C_0\nu}} < 1/4$.

But $\boldsymbol{\Lambda}_{\boldsymbol{L}}$ is not yet close to the the subdifferential of the $\ell^1$ norm – in particular, elements of the subdifferential of the $\ell^1$ norm should satisfy $\mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}] = \lambda\boldsymbol{\Sigma}_o$, but $\mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}_{\boldsymbol{L}}] = \boldsymbol{0}$. To correct this, we choose

$$\boldsymbol{\Lambda} = \boldsymbol{\Lambda}_{\boldsymbol{L}} + \boldsymbol{\Lambda}_{\boldsymbol{S}},$$

where the second element $\boldsymbol{\Lambda}_{\boldsymbol{S}}$ satisfies $\mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}_{\boldsymbol{S}}] = \lambda\boldsymbol{\Sigma}_o$. We need to show that we can choose $\boldsymbol{\Lambda}_{\boldsymbol{S}}$ such that this combined certificate $\boldsymbol{\Lambda}$ remains close to the subdifferential of the nuclear norm at $\boldsymbol{L}_o$, and is also close to the subdifferential of $\lambda\|\cdot\|_1$ at $\boldsymbol{S}_o$. The following lemma shows that this is possible:

LEMMA 5.6. *Under the conditions of the Theorem 5.3, with high probability, there exists* $\mathbf{\Lambda}_{\boldsymbol{S}}$ *such that*

*1* $\mathcal{P}_{\mathfrak{S}}[\mathbf{\Lambda}_{\boldsymbol{S}}] = \lambda \mathbf{\Sigma}_o$,
*2* $\|\mathcal{P}_{\mathfrak{S}^c}[\mathbf{\Lambda}_{\boldsymbol{S}}]\|_\infty < \frac{\lambda}{4}$,
*3* $\mathcal{P}_{\mathsf{T}}[\mathbf{\Lambda}_{\boldsymbol{S}}] = \mathbf{0}$,
*4* $\|\mathcal{P}_{\mathsf{T}^\perp}[\mathbf{\Lambda}_{\boldsymbol{S}}]\| < \frac{1}{4}$.

Under the assumptions of Theorem 5.3, we have in total for $\mathbf{\Lambda} = \mathbf{\Lambda}_{\boldsymbol{L}} + \mathbf{\Lambda}_{\boldsymbol{S}}$:

$$\|\mathcal{P}_{\mathsf{T}}[\mathbf{\Lambda}] - \boldsymbol{U}\boldsymbol{V}^*\|_F = \|\mathcal{P}_{\mathsf{T}}[\mathbf{\Lambda}_{\boldsymbol{L}}] - \boldsymbol{U}\boldsymbol{V}^*\|_F \leq \frac{1}{4n} \tag{5.3.25}$$

$$\|\mathcal{P}_{\mathsf{T}^\perp}[\mathbf{\Lambda}]\| \leq \|\mathcal{P}_{\mathsf{T}^\perp}[\mathbf{\Lambda}_{\boldsymbol{L}}\|_] + \|\mathcal{P}_{\mathsf{T}^\perp}[\mathbf{\Lambda}_{\boldsymbol{S}}]\| \leq \frac{1}{2} \tag{5.3.26}$$

$$\mathcal{P}_{\mathfrak{S}}[\mathbf{\Lambda}] = \mathcal{P}_{\mathfrak{S}}[\mathbf{\Lambda}_{\boldsymbol{S}}] = \lambda \mathbf{\Sigma}_o \tag{5.3.27}$$

$$\begin{aligned} \|\mathcal{P}_{\mathfrak{S}^c}[\mathbf{\Lambda}]\|_\infty &\leq \|\mathcal{P}_{\mathfrak{S}^c}[\mathbf{\Lambda}_{\boldsymbol{L}}]\|_\infty + \|\mathcal{P}_{\mathfrak{S}^c}[\mathbf{\Lambda}_{\boldsymbol{S}}]\|_\infty \\ &\leq \|\mathbf{\Lambda}_{\boldsymbol{L}}\|_\infty + \frac{\lambda}{4} \\ &\leq \frac{\lambda}{4} + \frac{\lambda}{4} = \frac{\lambda}{2}, \end{aligned} \tag{5.3.28}$$

where in the final inequality we have used the inequality (5.3.24). So with the so constructed $\mathbf{\Lambda}_{\boldsymbol{S}}$ and $\mathbf{\Lambda}_{\boldsymbol{L}}$, the combined

$$\mathbf{\Lambda} = \mathbf{\Lambda}_{\boldsymbol{L}} + \mathbf{\Lambda}_{\boldsymbol{S}}$$

satisfies all conditions of Lemma 5.5 under the assumptions of Theorem 5.3. So if we could prove Lemma 5.6, Theorem 5.3 would follow.

*Constructing the Dual Certificate* $\mathbf{\Lambda}_{\boldsymbol{S}}$ *Using Least Squares.*
To finish our proof, we need to verify Lemma 5.6, by showing that we can indeed construct $\mathbf{\Lambda}_{\boldsymbol{S}}$ that satisfies the requisite properties. To do this, we resort to a strategy that has proved useful at several points over the past few chapters: the method of least squares (minimum energy). Namely, we choose $\mathbf{\Lambda}_{\boldsymbol{S}}$ to satisfy the constraints $\mathcal{P}_{\mathfrak{S}}[\mathbf{\Lambda}_{\boldsymbol{S}}] = \lambda \mathbf{\Sigma}_o$ and $\mathcal{P}_{\mathsf{T}}[\mathbf{\Lambda}_{\boldsymbol{S}}] = \mathbf{0}$, but have the smallest possible energy: formally,

$$\mathbf{\Lambda}_{\boldsymbol{S}} = \arg\min_{\tilde{\mathbf{\Lambda}}} \left\|\tilde{\mathbf{\Lambda}}\right\|_F^2 \quad \text{subject to} \quad \mathcal{P}_{\mathfrak{S}}[\tilde{\mathbf{\Lambda}}] = \lambda \mathbf{\Sigma}_o,\ \mathcal{P}_{\mathsf{T}}[\tilde{\mathbf{\Lambda}}] = \mathbf{0}. \tag{5.3.29}$$

This optimization problem is feasible, provided $\mathfrak{S} \cap \mathsf{T} = \{\mathbf{0}\}$. The constraints ensure that $\mathbf{\Lambda}_{\boldsymbol{S}}$ satisfies criteria $(i)$ and $(iii)$ of Lemma 5.6 automatically.

To check that criteria $(ii)$ and $(iv)$ are satisfied, i.e., that $\mathcal{P}_{\mathfrak{S}^c}[\mathbf{\Lambda}_{\boldsymbol{S}}]$ has small $\ell^\infty$ norm and $\mathcal{P}_{\mathsf{T}^\perp}[\mathbf{\Lambda}_{\boldsymbol{S}}]$ has small operator norm, we utilize the scalar and operator Bernstein's inequalities, respectively. These calculations are facilitated by the existence of a closed-form solution to (5.3.29):

$$\mathbf{\Lambda}_{\boldsymbol{S}} = \lambda \mathcal{P}_{\mathsf{T}^\perp} \sum_{k=0}^{\infty} \left(\mathcal{P}_{\mathfrak{S}} \mathcal{P}_{\mathsf{T}} \mathcal{P}_{\mathfrak{S}}\right)^k [\mathbf{\Sigma}_o]. \tag{5.3.30}$$

Exercise 5.13 asks you to check that this construction indeed satisfies the constraints, and that it is indeed the solution to the energy minimization problem (5.3.29).

*Proof of Lemma 5.6* Let $\mathcal{E}$ be the event that $\|\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}}\| \leq \sigma$. This holds with high probability in the support set $\mathfrak{S}$. Notice that on the event $\mathcal{E}$,

$$\sum_{k=0}^{\infty} \left\| \left(\mathcal{P}_{\mathsf{S}}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathsf{S}}\right)^k \right\| \leq \sum_{k=0}^{\infty} \sigma^{2k} = \frac{1}{1-\sigma^{2k}} < \infty. \tag{5.3.31}$$

So, on $\mathcal{E}$, the summation in (5.3.30) converges, and

$$\boldsymbol{\Lambda}_{\boldsymbol{S}} = \lambda \mathcal{P}_{\mathsf{T}^\perp} \sum_{k=0}^{\infty} \left(\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}}\right)^k [\boldsymbol{\Sigma}_o] \tag{5.3.32}$$

is well-defined. Property $(iii)$, which states that $\mathcal{P}_{\mathsf{T}}[\boldsymbol{\Lambda}_{\boldsymbol{S}}] = \mathbf{0}$ follows immediately, since $\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathsf{T}^\perp} = 0$. Property $(i)$, which states that $\mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}_{\boldsymbol{S}}] = \lambda\boldsymbol{\Sigma}_o$, is a consequence of the construction of $\boldsymbol{\Lambda}_{\boldsymbol{S}}$ as the solution to a least squares problem (5.3.30). To verify this property, we can note that

$$\begin{aligned} \mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}_{\boldsymbol{S}}] &= \lambda \sum_{k=0}^{\infty} \left(\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}}\right)^k [\boldsymbol{\Sigma}_o] - \lambda \sum_{k=1}^{\infty} \left(\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}}\right)^k [\boldsymbol{\Sigma}_o] \\ &= \lambda\boldsymbol{\Sigma}_o, \end{aligned} \tag{5.3.33}$$

as desired. Properties $(iv)$ and $(ii)$ state that $\boldsymbol{\Lambda}_{\boldsymbol{S}}$ is small, in two appropriate senses. These require a bit more work.

**Verifying** $(iv)$**.** Write

$$\boldsymbol{\Lambda}_{\boldsymbol{S}} = \underbrace{\lambda\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{\Sigma}_o]}_{\boldsymbol{\Lambda}_{\boldsymbol{S}}^{(1)}} + \underbrace{\lambda\mathcal{P}_{\mathsf{T}^\perp} \sum_{k=1}^{\infty} \left(\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}}\right)^k [\boldsymbol{\Sigma}_o]}_{\boldsymbol{\Lambda}_{\boldsymbol{S}}^{(2)}}. \tag{5.3.34}$$

For the second term, we introduce the more concise notation

$$\mathcal{R} = \mathcal{P}_{\mathsf{T}^\perp} \sum_{k=1}^{\infty} \left(\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}}\right)^k, \tag{5.3.35}$$

so that

$$\boldsymbol{\Lambda}_{\boldsymbol{S}}^{(2)} = \lambda\mathcal{R}[\boldsymbol{\Sigma}_o]. \tag{5.3.36}$$

Notice that

$$\|\mathcal{R}\| \leq \frac{\sigma^2}{1-\sigma^2}. \tag{5.3.37}$$

The norm of $\boldsymbol{\Lambda}_{\boldsymbol{S}}^{(1)}$ can be controlled by noting that

$$\left\|\boldsymbol{\Lambda}_{\boldsymbol{S}}^{(1)}\right\| = \lambda \left\|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{\Sigma}_o]\right\| \leq \lambda \left\|\boldsymbol{\Sigma}_o\right\|. \tag{5.3.38}$$

With high probability,

$$\|\mathbf{\Sigma}_o\| \leq C\sqrt{\rho m}, \tag{5.3.39}$$

whence for $\rho < \rho_\star$ a small constant, $\left\|\mathbf{\Lambda}_{\boldsymbol{S}}^{(1)}\right\| \leq \frac{1}{16}$. To control the norm of $\mathbf{\Lambda}_{\boldsymbol{S}}^{(2)}$, let $\mathsf{N}$ be a $\frac{1}{2}$ net for $\mathbb{S}^{n-1}$. By Lemma 3.25, such a net exists, with size $|\mathsf{N}| \leq 6^n$. Moreover,

$$\begin{aligned}
\left\|\mathbf{\Lambda}_{\boldsymbol{S}}^{(2)}\right\| &= \sup_{\boldsymbol{u},\boldsymbol{v}\in\mathbb{S}^{n-1}} \boldsymbol{u}^* \mathbf{\Lambda}_{\boldsymbol{S}}^{(2)} \boldsymbol{v} \\
&\leq 4 \max_{\boldsymbol{u},\boldsymbol{v}\in\mathsf{N}} \boldsymbol{u}^* \mathbf{\Lambda}_{\boldsymbol{S}}^{(2)} \boldsymbol{v} \\
&= 4 \max_{\boldsymbol{u},\boldsymbol{v}\in\mathsf{N}} \langle \boldsymbol{u}\boldsymbol{v}^*, \lambda\mathcal{R}\left[\mathbf{\Sigma}_o\right]\rangle \\
&= 4 \max_{\boldsymbol{u},\boldsymbol{v}\in\mathsf{N}} \langle \lambda\mathcal{R}[\boldsymbol{u}\boldsymbol{v}^*], \mathbf{\Sigma}_o\rangle \\
&= 4 \max_{\boldsymbol{u},\boldsymbol{v}\in\mathsf{N}} \langle \boldsymbol{X}_{\boldsymbol{u},\boldsymbol{v}}, \mathbf{\Sigma}_o\rangle\,.
\end{aligned} \tag{5.3.40}$$

Conditioned on the support $\mathfrak{S}$ of the sparse error term, we can observe that the random variable $\langle \boldsymbol{X}_{\boldsymbol{u},\boldsymbol{v}}, \mathbf{\Sigma}_o\rangle$ is a linear combination of Rademacher ($\pm 1$) random variables. Hoeffding's inequality gives

$$\mathbb{P}\Big[\langle \boldsymbol{X}_{\boldsymbol{u},\boldsymbol{v}}, \mathbf{\Sigma}_o\rangle > t \;\mid\; \mathfrak{S}\Big] \leq \exp\left(-\frac{t^2}{2\left\|\boldsymbol{X}_{\boldsymbol{u},\boldsymbol{v}}\right\|_F^2}\right). \tag{5.3.41}$$

On $\mathcal{E}$, using the bound (5.3.37), we can control the norm of $\boldsymbol{X}_{\boldsymbol{u},\boldsymbol{v}}$, via

$$\left\|\boldsymbol{X}_{\boldsymbol{u},\boldsymbol{v}}\right\|_F \leq \frac{\lambda\sigma^2}{1-\sigma^2}. \tag{5.3.42}$$

So, for each $\boldsymbol{u}, \boldsymbol{v}$,

$$\mathbb{P}\Big[\langle \boldsymbol{X}_{\boldsymbol{u},\boldsymbol{v}}, \mathbf{\Sigma}_o\rangle > t \;\mid\; \mathcal{E}\Big] \leq \exp\left(-\frac{t^2}{2\left\|\boldsymbol{X}_{\boldsymbol{u},\boldsymbol{v}}\right\|_F^2}\right). \tag{5.3.43}$$

Hence,

$$\begin{aligned}
&\mathbb{P}\Big[\,\left\|\mathbf{\Lambda}_{\boldsymbol{S}}^{(2)}\right\| \;>\; t\,\Big] \\
&\leq \mathbb{P}\Big[\,\max_{\boldsymbol{u},\boldsymbol{v}\in\mathsf{N}} \langle \boldsymbol{X}_{\boldsymbol{u},\boldsymbol{v}}, \mathbf{\Sigma}_o\rangle \;>\; \tfrac{t}{4}\,\Big] \\
&\leq \mathbb{P}\Big[\,\max_{\boldsymbol{u},\boldsymbol{v}\in\mathsf{N}} \langle \boldsymbol{X}_{\boldsymbol{u},\boldsymbol{v}}, \mathbf{\Sigma}_o\rangle \;>\; \tfrac{t}{4} \mid \mathcal{E}\,\Big] \quad + \quad \mathbb{P}\Big[\mathcal{E}^c\Big] \\
&\leq |\mathsf{N}|^2 \times \max_{\boldsymbol{u},\boldsymbol{v}\in\mathsf{N}} \mathbb{P}\Big[\,\langle \boldsymbol{X}_{\boldsymbol{u},\boldsymbol{v}}, \mathbf{\Sigma}_o\rangle \;>\; \tfrac{t}{4} \mid \mathcal{E}\,\Big] \quad + \quad \mathbb{P}\Big[\mathcal{E}^c\Big] \\
&\leq 6^{2n} \times \exp\left(-\frac{t^2(1-\sigma^2)^2}{2\lambda^2\sigma^4}\right) \quad + \quad \mathbb{P}\left[\mathcal{E}^c\right].
\end{aligned} \tag{5.3.44}$$

Setting $t = \frac{1}{8}$, and ensuring that $\sigma$ is appropriately small, we obtain that with high probability $\left\|\mathbf{\Lambda}_{\boldsymbol{S}}^{(2)}\right\| \leq \frac{1}{8}$; combining with our bound on $\left\|\mathbf{\Lambda}_{\boldsymbol{S}}^{(1)}\right\|$, we obtain that $\|\mathbf{\Lambda}_{\boldsymbol{S}}\| < \frac{1}{4}$ with high probability, as desired.

**Verifying** $(ii)$**.** We finish by verifying that with high probability, $\|\mathcal{P}_{\mathfrak{S}^c}[\mathbf{\Lambda}_{\boldsymbol{S}}]\|_\infty < \frac{\lambda}{4}$. For this, notice that

$$\begin{aligned}\mathcal{P}_{\mathfrak{S}^c}[\mathbf{\Lambda}_{\boldsymbol{S}}] &= \lambda \mathcal{P}_{\mathfrak{S}^c}\mathcal{P}_{\mathsf{T}^\perp}\sum_{k=0}^{\infty}\left(\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}}\right)^k[\mathbf{\Sigma}_o] \\ &= \lambda \mathcal{P}_{\mathfrak{S}^c}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}}\sum_{k=0}^{\infty}\left(\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}}\right)^k[\mathbf{\Sigma}_o] \\ &\doteq \lambda \mathcal{H}[\mathbf{\Sigma}_o]. \end{aligned} \tag{5.3.45}$$

On $\mathcal{E}$, for any $(i,j) \in \mathfrak{S}^c$, we have

$$\begin{aligned}\left\|\mathcal{H}^*[\boldsymbol{e}_i\boldsymbol{e}_j^*]\right\|_F &= \left\|\left[\sum_{k=0}^{\infty}(\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}})^k\right]\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}[\boldsymbol{e}_i\boldsymbol{e}_j^*]\right\|_F \\ &\le \left\|\left[\sum_{k=0}^{\infty}(\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}})^k\right]\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\right\|\left\|\mathcal{P}_{\mathsf{T}}[\boldsymbol{e}_i\boldsymbol{e}_j^*]\right\|_F \\ &\le \frac{\sigma}{1-\sigma^2}\times\sqrt{\frac{2\nu r}{n}} \\ &\le C\sqrt{\log n}. \end{aligned} \tag{5.3.46}$$

Notice that

$$\|\mathcal{P}_{\mathfrak{S}^c}[\mathbf{\Lambda}_{\boldsymbol{S}}]\|_\infty = \lambda \max_{i,j}|\boldsymbol{e}_i^*\mathcal{H}[\mathbf{\Sigma}_o]\boldsymbol{e}_j| = \lambda\max_{i,j}\left|\left\langle\mathcal{H}[\boldsymbol{e}_i\boldsymbol{e}_j^*],\mathbf{\Sigma}_o\right\rangle\right|. \tag{5.3.47}$$

Write

$$Y_{ij} = \left\langle\mathcal{H}[\boldsymbol{e}_i\boldsymbol{e}_j^*],\mathbf{\Sigma}_o\right\rangle \quad \in \mathbb{R}. \tag{5.3.48}$$

Using Hoeffding's inequality again, we have

$$\mathbb{P}\Big[|Y_{ij}| > t \;\Big|\; \mathfrak{S}\Big] \;\le\; 2\exp\left(-\frac{t^2}{2\left\|\mathcal{H}[\boldsymbol{e}_i\boldsymbol{e}_j^*]\right\|_F^2}\right). \tag{5.3.49}$$

Hence,

$$\mathbb{P}\Big[|Y_{ij}| > t \;\Big|\; \mathcal{E}\Big] \;\le\; 2n^{-12} \tag{5.3.50}$$

We have

$$\begin{aligned}&\mathbb{P}\Big[\|\mathcal{P}_{\mathfrak{S}^c}[\mathbf{\Lambda}_{\boldsymbol{S}}]\|_\infty \ge \tfrac{\lambda}{4}\Big] \quad\le\quad \mathbb{P}\Big[\max_{i,j}|Y_{ij}| > \tfrac{1}{4}\Big] \\ &\le \mathbb{P}\Big[\max_{i,j}|Y_{ij}| > \tfrac{1}{4} \;\Big|\; \mathcal{E}\Big] \quad+\quad \mathbb{P}\Big[\,\mathcal{E}^c\,\Big] \\ &\le \sum_{i,j}\mathbb{P}\Big[|Y_{ij}| > \tfrac{1}{4} \;\Big|\; \mathcal{E}\Big] \quad+\quad \mathbb{P}\Big[\,\mathcal{E}^c\,\Big] \\ &\le n^2\times 2n^{-12} \quad+\quad \mathbb{P}\left[\mathcal{E}^c\right] \\ &\le 2n^{-10} \quad+\quad \mathbb{P}\left[\mathcal{E}^c\right]. \end{aligned} \tag{5.3.51}$$

This completes the proof. □

### 5.3.3 Some Extensions to the Main Result

Several refinements or extensions to Theorem 5.3 are possible. We describe these here, and leave their proofs as exercises.

*Nonsquare Matrices.*
In the general rectangular case where $\boldsymbol{L}_o$ is $n_1 \times n_2$, let $n_{(1)} \doteq \max\{n_1, n_2\}$. Then PCP with $\lambda = 1/\sqrt{n_{(1)}}$ succeeds with probability at least $1 - cn_{(1)}^{-10}$, provided that $\operatorname{rank}(\boldsymbol{L}_o) \leq \rho_r n_{(2)} \nu^{-1} (\log n_{(1)})^{-2}$ and $m \leq \rho_s n_1 n_2$. A rather remarkable fact is that there is no tuning parameter in solving the PCP program. Under the assumption of the theorem, solving

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \frac{1}{\sqrt{n_{(1)}}} \|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \boldsymbol{Y} = \boldsymbol{L} + \boldsymbol{S},$$

always returns the correct answer. This is surprising because one might have expected that one would have to choose the right scalar $\lambda$ to balance the two terms in $\|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1$ appropriately (perhaps depending on their relative size). This is, however, clearly not the case. In this sense, the choice $\lambda = 1/\sqrt{n_{(1)}}$ is universal. Further, it is not so clear *a priori* why $\lambda = 1/\sqrt{n_{(1)}}$ is a correct choice no matter what $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$ are. It is the mathematical analysis which reveals the correctness of this value. In fact, the proof of the theorem gives a range of correct values, and we have selected arguably the simplest one in that range.

*Dense Error Correction.*
In the above Theorem 5.3, one may wonder how large the fraction of non-zero entries in $\boldsymbol{S}_o$, namely $\rho_s$, can be in practice. The result will not be very useful if $\rho_s$ has to be extremely small. As it turns out, in most cases, $\rho_s$ can be rather significant, and in some extreme cases, $\boldsymbol{S}_o$ does not even have to be sparse at all!

To be more precise, under the same assumptions of Theorem 5.3, one can rigorously prove: For any $\rho_s < 1$, as $n$ becomes large[7], Principal Component Pursuit (5.2.2) exactly recovers $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ with high probability,[8] provided

$$\lambda = C_1 \left(4\sqrt{1-\rho_s} + \frac{9}{4}\right)^{-1} \sqrt{\frac{1-\rho_s}{\rho_s n}}, \quad r < \frac{C_2 n}{\nu \log^2 n}, \tag{5.3.52}$$

where $0 < C_1 \leq 4/5$ and $C_2 > 0$ are certain constants. In other words, provided the rank of a matrix is of the order of $n/\nu \log^2 n$, if we can choose a $\lambda$ that is dependent on $\rho_s$, PCP recovers the low-rank matrix exactly even when an arbitrarily large fraction of its entries are corrupted by errors of arbitrary magnitudes and the locations of the uncorrupted entries are unknown!

Furthermore, by slightly modifying the proof for the above statement, one can

[7] For $\rho_s$ closer to one, the dimension $n$ must be larger; formally, $n > n_0(\rho_s)$.
[8] By "high probability", we mean with probability at least $1 - cn^{-\beta}$ for some fixed $\beta > 0$.

show that the exact recovery will be guaranteed with high probability if the rank $r$ and the parameter $\lambda$ are chosen as follows instead:

$$\lambda = \frac{1}{\sqrt{n \log n}}, \quad r < \frac{C_2 n}{\nu \log^3 n}. \tag{5.3.53}$$

That is, if there is reason to believe the rank of the matrix $\boldsymbol{L}_o$ is more restricted (say, in practice, fixed), we only have to set $\lambda = \frac{1}{\sqrt{n \log n}}$ which does not depend on any knowledge in the fraction of errors $\rho_s$. With such settings, PCP would succeed in finding the correct solution. As we will see in Section 5.6, a similar choice of $\lambda$ works for recovering a low-rank matrix with both missing and corrupted entries.

In this book, we chose not to provide detailed proof to these extensions as their proof strategy and techniques are very similar to the proof of Theorem 5.3. Nevertheless, interested readers might resort to the work [GWL+10, CJSC13] for more details.

*Derandomization of Error Signs.*
In the above Theorem 5.3, both the support and signs of the error term $\boldsymbol{S}_o$ are assumed to be random. In practice, such random models might be considered as less practical as many practical sparse signals might not be entirely random. As it turns out, the randomness assumption on the signs of $\boldsymbol{S}_o$ is not crucial for the conclusion of the theorem.

More precisely, one can prove that: Suppose $\boldsymbol{L}_o$ obeys the conditions of Theorem 5.3 and that the locations of the nonzero entries of $\boldsymbol{S}_o$ follow the Bernoulli model with parameter $\rho_s$, and the signs of $\boldsymbol{S}_o$ are i.i.d. $\pm 1$ (independent from the locations). Then if the PCP solution is exact with high probability, then it is also exact with at least the same probability for the model in which the signs (and values) of $\boldsymbol{S}_o$ are fixed and the locations are sampled from the Bernoulli model with parameter $\frac{1}{2}\rho_s$.

That is, we may consider $\boldsymbol{S}_o$ comes from an *arbitrary prefixed* matrix $\boldsymbol{S}$ as

$$\boldsymbol{S}_o = \mathcal{P}_{\mathfrak{S}}[\boldsymbol{S}],$$

where $\mathfrak{S}$ is sampled from a Bernoulli model with parameter $\frac{1}{2}\rho_s$. Then the PCP recovers the correct $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$ with high probability too. In other words, to remove the randomness in the signs, we lose half of the density in the error term $\boldsymbol{S}_o$. Note that the values and signs of the fixed matrix $\boldsymbol{S}$ can even be chosen to be the most "adversarial": $\boldsymbol{S} = \boldsymbol{L}_o$!

What about the randomness in the locations $\mathfrak{S}$? Can we remove it too without significantly reducing the strength of the conclusion? As we have discussed earlier in this section, the randomness of the support of $\boldsymbol{S}_o$ is to ensure identifiability. If the support of $\boldsymbol{S}_o$ is not sufficiently random in both columns and rows, say it concentrates on certain row or column, then it might easily become impossible to recover the corresponding row or column of $\boldsymbol{L}_o$. One may refer to [CSPW09] for

conditions under which PCP succeeds with deterministic models for the support of $\boldsymbol{S}_o$.

*Sparse Outlier Pursuit.*
In many applications, the corruptions might concentrate on a small number of columns of the low-rank matrix $\boldsymbol{L}_o$, instead of individual entries. In other words, the data matrix is of the form:

$$\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{O}_o,$$

where $\boldsymbol{O}_o$ is a matrix with a sparse number of nonzero columns. The corresponding columns can be viewed as "outliers" which have little to do with the low-rank matrix $\boldsymbol{L}_o$. This problem is also known as Robust PCA with sparse (column-wise) outliers [XCS12]. In this case, one may consider a norm that promotes column-wise sparsity: the sum of $\ell^2$ norms of all columns, also known as the $(2,1)$-norm:

$$\|\boldsymbol{O}\|_{2,1} = \sum_{i}^{n_2} \|\boldsymbol{O}_i\|_2, \tag{5.3.54}$$

where $\boldsymbol{O}_i \in \mathbb{R}^{n_1}$ are the columns of the matrix $\boldsymbol{O} \in \mathbb{R}^{n_1 \times n_2}$. So to decompose the matrix $\boldsymbol{Y}$, one may consider the following convex program known as "outlier pursuit":

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{O}\|_{2,1} \quad \text{subject to} \quad \boldsymbol{L} + \boldsymbol{O} = \boldsymbol{Y}. \tag{5.3.55}$$

Like the PCP for sparse corruptions, the above program can recover the correct low-rank and the column-sparse components under fairly broad conditions,[9] as detailed in the work of [XCS12].

## 5.4 Stable Principal Component Pursuit with Noise

The PCP model and result (Theorem 5.3) is limited to situations in which the low-rank component is exactly low-rank and the sparse component is exactly sparse. However, in real world applications the observations are often perturbed by noise, which may be stochastic or deterministic, affecting every entry of the data matrix. For example, in face recognition that we mentioned earlier, the human face is not a strictly convex and Lambertian surface hence the low-rank model (due to photometric properties) is only approximately low-rank. In ranking and collaborative filtering, user's ratings could be noisy because of the lack of control in the data collection process. Therefore, for the PCP method to be applicable to a wider range of real world problems, we need to examine if it can handle small entry-wise (dense) noise.

[9] The columns of the low-rank matrix $\boldsymbol{L}_o$ satisfy certain incoherent condition, and the fraction of outliers is bounded accordingly.

In the presence of noise, the new measurement model becomes

$$\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{S}_o + \boldsymbol{Z}_o, \tag{5.4.1}$$

where $\boldsymbol{Z}_o$ is a small error term that could affect the value of each entry of the matrix. However, all we assume about $\boldsymbol{Z}_o$ here is that $\|\boldsymbol{Z}_o\|_F \leq \varepsilon$ for some $\varepsilon > 0$.

To recover the unknown matrices $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$, one may consider solving the following optimization problem, as a relaxed version to PCP (5.2.2):

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \|\boldsymbol{Y} - \boldsymbol{L} - \boldsymbol{S}\|_F \leq \varepsilon. \tag{5.4.2}$$

where we choose $\lambda = 1/\sqrt{n}$. Note that with this choice, we typically have $\lambda < 1/2$ for large $n$. Our main result is that under the same conditions as PCP, the above convex program gives a stable estimate of $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$:

THEOREM 5.7 (Stability of PCP to Bounded Noise). *Under the same assumptions of Theorem 5.3, that is, $\boldsymbol{L}_o$ obeys the incoherence conditions and the support of $\boldsymbol{S}_o$ is uniformly distributed of size $m$. Then if $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$ satisfy*

$$\operatorname{rank}(\boldsymbol{L}_o) \leq \frac{\rho_r n}{\nu \log^2 n} \quad \text{and} \quad m \leq \rho_s n^2, \tag{5.4.3}$$

*with $\rho_r, \rho_s > 0$ being sufficiently small numerical constants, with high probability in the support of $\boldsymbol{S}_o$, for any $\boldsymbol{Z}_o$ with $\|\boldsymbol{Z}_o\|_F \leq \varepsilon$, the solution $(\hat{\boldsymbol{L}}, \hat{\boldsymbol{S}})$ to the convex program (5.4.2) satisfies*

$$\|\hat{\boldsymbol{L}} - \boldsymbol{L}_o\|_F^2 + \|\hat{\boldsymbol{S}} - \boldsymbol{S}_o\|_F^2 \leq C\varepsilon^2, \tag{5.4.4}$$

*where the constant $C = \left(16\sqrt{5}n + \sqrt{2}\right)^2$.*

Here, we would like to point out two ways to view the significance of this result. To some extent, the model (5.4.2) unifies the classical PCA and the robust PCA by considering both gross sparse errors and small entry-wise noise in the measurements. So on the one hand, the above theorem says that the low-rank and sparse decomposition via PCP is stable in the presence of small entry-wise noise, hence making PCP more widely applicable to practical problems where the low-rank structure is not exact. On the other hand, the theorem convincingly justifies that the classical PCA can now be made robust to sparse gross corruptions via certain convex program. Since this convex program can be solved very efficiently via algorithms similar to Algorithm 5.1, at a cost not so much higher than the classical PCA, this model and result can be applied to many practical problems where both small noise and gross corruption are present simultaneously.

Before we set out to prove the above result, let us first introduce some new notation. For any matrix pair $\boldsymbol{X} = (\boldsymbol{L}, \boldsymbol{S})$ let

$$\|\boldsymbol{X}\|_F \doteq \left(\|\boldsymbol{L}\|_F^2 + \|\boldsymbol{S}\|_F^2\right)^{1/2}, \quad \|\boldsymbol{X}\|_\diamond = \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1.$$

Define a projection operator

$$\mathcal{P}_\mathsf{T} \times \mathcal{P}_\mathfrak{S} : (\boldsymbol{L}, \boldsymbol{S}) \mapsto (\mathcal{P}_\mathsf{T}[\boldsymbol{L}], \mathcal{P}_\mathfrak{S}[\boldsymbol{S}]).$$

Also we define the subspaces $\Gamma \doteq \{(\boldsymbol{Q}, \boldsymbol{Q}) \mid \boldsymbol{Q} \in \mathbb{R}^{n\times n}\}$ and $\Gamma^\perp \doteq \{(\boldsymbol{Q}, -\boldsymbol{Q}) \mid \boldsymbol{Q} \in \mathbb{R}^{n\times n}\}$, and let $\mathcal{P}_\Gamma$ and $\mathcal{P}_{\Gamma^\perp}$ denote their respective projection operators.

LEMMA 5.8. *Suppose that* $\|\mathcal{P}_\mathsf{T}\mathcal{P}_\mathfrak{S}\| \le 1/2$. *Then for any pair* $\boldsymbol{X} = (\boldsymbol{L}, \boldsymbol{S})$, $\|\mathcal{P}_\Gamma(\mathcal{P}_\mathsf{T} \times \mathcal{P}_\mathfrak{S})[\boldsymbol{X}]\|_F^2 \ge \frac{1}{4}\|(\mathcal{P}_\mathsf{T} \times \mathcal{P}_\mathfrak{S})[\boldsymbol{X}]\|_F^2$.

*Proof* For any matrix pair $\boldsymbol{X}' = (\boldsymbol{L}', \boldsymbol{S}')$, $\mathcal{P}_\Gamma[\boldsymbol{X}'] = \left(\frac{\boldsymbol{L}'+\boldsymbol{S}'}{2}, \frac{\boldsymbol{L}'+\boldsymbol{S}'}{2}\right)$ and so $\|\mathcal{P}_\Gamma[\boldsymbol{X}']\|_F^2 = \frac{1}{2}\|\boldsymbol{L}' + \boldsymbol{S}'\|_F^2$. So,

$$\begin{aligned}\|\mathcal{P}_\Gamma(\mathcal{P}_\mathsf{T} \times \mathcal{P}_\mathfrak{S})[\boldsymbol{X}]\|_F^2 &= \tfrac{1}{2}\|\mathcal{P}_\mathsf{T}[\boldsymbol{L}] + \mathcal{P}_\mathfrak{S}[\boldsymbol{S}]\|_F^2 \\ &= \tfrac{1}{2}\left(\|\mathcal{P}_\mathsf{T}[\boldsymbol{L}]\|_F^2 + \|\mathcal{P}_\mathfrak{S}[\boldsymbol{S}]\|_F^2 + 2\langle \mathcal{P}_\mathsf{T}[\boldsymbol{L}], \mathcal{P}_\mathfrak{S}[\boldsymbol{S}]\rangle\right).\end{aligned}$$

Now,

$$\begin{aligned}\langle \mathcal{P}_\mathsf{T}[\boldsymbol{L}], \mathcal{P}_\mathfrak{S}[\boldsymbol{S}]\rangle &= \langle \mathcal{P}_\mathsf{T}[\boldsymbol{L}], (\mathcal{P}_\mathsf{T}\mathcal{P}_\mathfrak{S})\mathcal{P}_\mathfrak{S}[\boldsymbol{S}]\rangle \\ &\ge -\|\mathcal{P}_\mathsf{T}\mathcal{P}_\mathfrak{S}\|\|\mathcal{P}_\mathsf{T}[\boldsymbol{L}]\|_F\|\mathcal{P}_\mathfrak{S}[\boldsymbol{S}]\|_F.\end{aligned}$$

Since $\|\mathcal{P}_\mathsf{T}\mathcal{P}_\mathfrak{S}\| \le 1/2$,

$$\begin{aligned}&\|\mathcal{P}_\Gamma(\mathcal{P}_\mathsf{T} \times \mathcal{P}_\mathfrak{S})[\boldsymbol{X}]\|_F^2 \\ &\ge \tfrac{1}{2}\left(\|\mathcal{P}_\mathsf{T}[\boldsymbol{L}]\|_F^2 + \|\mathcal{P}_\mathfrak{S}[\boldsymbol{S}]\|_F^2 - \|\mathcal{P}_\mathsf{T}[\boldsymbol{L}]\|_F\|\mathcal{P}_\mathfrak{S}[\boldsymbol{S}]\|_F\right) \\ &\ge \tfrac{1}{4}\left(\|\mathcal{P}_\mathsf{T}[\boldsymbol{L}]\|_F^2 + \|\mathcal{P}_\mathfrak{S}[\boldsymbol{S}]\|_F^2\right) = \tfrac{1}{4}\|(\mathcal{P}_\mathsf{T} \times \mathcal{P}_\mathfrak{S})[\boldsymbol{X}]\|_F^2,\end{aligned}$$

where we have used that for any $a, b$, $a^2 + b^2 - ab \ge (a^2 + b^2)/2$. □

*Proof of Theorem 5.7.* The proof for the noisy case largely relies on the method and results we have developed before for proving the noiseless case of PCP. From the proof of Theorem 5.3, we know that, with high probability, there exists a dual certificate $\boldsymbol{\Lambda}$ satisfying the conditions in Lemma (5.5):

$$\begin{cases} \|\mathcal{P}_\mathsf{T}[\boldsymbol{\Lambda}] - \boldsymbol{U}\boldsymbol{V}^*\|_F \le \frac{\lambda}{8}, & \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{\Lambda}]\| < \frac{1}{2}, \\ \|\mathcal{P}_\mathfrak{S}[\boldsymbol{\Lambda}] - \lambda\boldsymbol{\Sigma}_o\|_F \le \frac{\lambda}{8}, & \|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{\Lambda}]\|_\infty < \frac{\lambda}{2}. \end{cases} \tag{5.4.5}$$

Our proof uses two crucial properties of $\hat{\boldsymbol{X}} = (\hat{\boldsymbol{L}}, \hat{\boldsymbol{S}})$. First, since $\boldsymbol{X}_o$ is also a feasible solution to (5.4.2), we have $\|\hat{\boldsymbol{X}}\|_\diamondsuit \le \|\boldsymbol{X}_o\|_\diamondsuit$. Second, we use triangle inequality to get

$$\begin{aligned}\|\hat{\boldsymbol{L}} + \hat{\boldsymbol{S}} - \boldsymbol{L}_o - \boldsymbol{S}_o\|_F &\le \|\hat{\boldsymbol{L}} + \hat{\boldsymbol{S}} - \boldsymbol{Y}\|_F + \|\boldsymbol{L}_o + \boldsymbol{S}_o - \boldsymbol{Y}\|_F \\ &\le 2\varepsilon.\end{aligned} \tag{5.4.6}$$

Furthermore, set $\hat{\boldsymbol{X}} = \boldsymbol{X}_o + \boldsymbol{H}$ where $\boldsymbol{H} = (\boldsymbol{H}_L, \boldsymbol{H}_S)$. We want to bound the norm of the perturbation $\|\boldsymbol{H}\|_F^2 = \|\boldsymbol{H}_L\|_F^2 + \|\boldsymbol{H}_S\|_F^2$, Notice that unlike the noise-free case, here $\boldsymbol{H}_L + \boldsymbol{H}_S$ is not necessarily equal to zero. So in order to leverage results from the noise-free case, we decompose the perturbation into the two orthogonal components in $\Gamma$ and $\Gamma^\perp$ respectively: $\boldsymbol{H}^\Gamma = \mathcal{P}_\Gamma[\boldsymbol{H}]$ and $\boldsymbol{H}^{\Gamma^\perp} = \mathcal{P}_{\Gamma^\perp}[\boldsymbol{H}]$. Then $\|\boldsymbol{H}\|_F^2$ can be expanded as

$$\begin{aligned}\|\boldsymbol{H}\|_F^2 &= \|\boldsymbol{H}^\Gamma\|_F^2 + \|\boldsymbol{H}^{\Gamma^\perp}\|_F^2 \\ &= \|\boldsymbol{H}^\Gamma\|_F^2 + \|(\mathcal{P}_\mathsf{T} \times \mathcal{P}_\Omega)[\boldsymbol{H}^{\Gamma^\perp}]\|_F^2 + \|(\mathcal{P}_{\mathsf{T}^\perp} \times \mathcal{P}_{\mathfrak{S}^c})[\boldsymbol{H}^{\Gamma^\perp}]\|_F^2.\end{aligned} \tag{5.4.7}$$

Since (5.4.6) gives us

$$\|\boldsymbol{H}^{\Gamma}\|_F = \big(\|(\boldsymbol{H}_L+\boldsymbol{H}_S)/2\|_F^2 + \|(\boldsymbol{H}_L+\boldsymbol{H}_S)/2\|_F^2\big)^{1/2} \le \sqrt{2}/2\times 2\varepsilon = \sqrt{2}\varepsilon, \tag{5.4.8}$$

it suffices to bound the second and third terms on the right-hand-side of (5.4.7).

**a. Bound the third term of (5.4.7).** Let $\boldsymbol{\Lambda}$ be a dual certificate satisfying (5.4.5). Then we have

$$\|\boldsymbol{X}_o+\boldsymbol{H}\|_\diamond \ge \|\boldsymbol{X}_o+\boldsymbol{H}^{\Gamma^\perp}\|_\diamond - \|\boldsymbol{H}^{\Gamma}\|_\diamond. \tag{5.4.9}$$

Since $\boldsymbol{H}_L^{\Gamma^\perp}+\boldsymbol{H}_S^{\Gamma^\perp}=0$, following the proof of Lemma 5.5, we have

$$\begin{aligned}
&\|\boldsymbol{X}_o+\boldsymbol{H}^{\Gamma^\perp}\|_\diamond \\
&\ge \|\boldsymbol{X}_o\|_\diamond + 1/8\|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}_L^{\Gamma^\perp}]+\lambda/8\|_*\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}_S^{\Gamma^\perp}]\|_1 \\
&\ge \|\boldsymbol{X}_o\|_\diamond + \frac{1}{8}\Big(\|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}_L^{\Gamma^\perp}]\|_* + \lambda\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}_S^{\Gamma^\perp}]\|_1\Big),
\end{aligned}$$

which implies that

$$\|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}_L^{\Gamma^\perp}]\|_* + \lambda\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}_S^{\Gamma^\perp}]\|_1 \le 8\|\boldsymbol{H}^{\Gamma}\|_\diamond. \tag{5.4.10}$$

For any matrix $\boldsymbol{Y}\in\mathbb{R}^{n\times n}$, we have the following inequalities:

$$\|\boldsymbol{Y}\|_F \le \|\boldsymbol{Y}\|_* \le \sqrt{n}\|\boldsymbol{Y}\|_F, \quad \frac{1}{\sqrt{n}}\|\boldsymbol{Y}\|_F \le \lambda\|\boldsymbol{Y}\|_1 \le \sqrt{n}\|\boldsymbol{Y}\|_F,$$

where we assume $\lambda=\frac{1}{\sqrt{n}}$. Therefore

$$\begin{aligned}
&\|(\mathcal{P}_{\mathsf{T}^\perp}\times\mathcal{P}_{\mathfrak{S}^c})[\boldsymbol{H}^{\Gamma^\perp}]\|_F \\
&\le \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}_L^{\Gamma^\perp}]\|_F + \|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}_S^{\Gamma^\perp}]\|_F \\
&\le \|\mathcal{P}_{\mathsf{T}^\perp}[\boldsymbol{H}_L^{\Gamma^\perp}]\|_* + \lambda\sqrt{n}\|\mathcal{P}_{\mathfrak{S}^c}[\boldsymbol{H}_S^{\Gamma^\perp}]\|_1 \\
&\le 8\sqrt{n}\|\boldsymbol{H}^{\Gamma}\|_\diamond = 8\sqrt{n}\big(\|\boldsymbol{H}_L^{\Gamma}\|_* + \lambda\|\boldsymbol{H}_S^{\Gamma}\|_1\big) \\
&\le 8n(\|\boldsymbol{H}_L^{\Gamma}\|_F + \|\boldsymbol{H}_S^{\Gamma}\|_F) \le 8\sqrt{2}n\|\boldsymbol{H}^{\Gamma}\|_F \le 16n\varepsilon,
\end{aligned} \tag{5.4.11}$$

where the last equation uses the fact that $\boldsymbol{H}_L^{\Gamma}=\boldsymbol{H}_S^{\Gamma}$.

**b. Bound the second term of (5.4.7).** By Lemma 5.8,

$$\|\mathcal{P}_{\Gamma}(\mathcal{P}_{\mathsf{T}}\times\mathcal{P}_{\mathfrak{S}})[\boldsymbol{H}^{\Gamma^\perp}]\|_F^2 \ge \frac{1}{4}\|(\mathcal{P}_{\mathsf{T}}\times\mathcal{P}_{\mathfrak{S}})[\boldsymbol{H}^{\Gamma^\perp}]\|_F^2.$$

But since $\mathcal{P}_{\Gamma}[\boldsymbol{H}^{\Gamma^\perp}]=\boldsymbol{0}=\mathcal{P}_{\Gamma}[\mathcal{P}_{\mathsf{T}}\times\mathcal{P}_{\mathfrak{S}})[\boldsymbol{H}^{\Gamma^\perp}]+\mathcal{P}_{\Gamma}[\mathcal{P}_{\mathsf{T}^\perp}\times\mathcal{P}_{\mathfrak{S}^c})[\boldsymbol{H}^{\Gamma^\perp}]$, we have

$$\begin{aligned}
\|\mathcal{P}_{\Gamma}(\mathcal{P}_{\mathsf{T}}\times\mathcal{P}_{\mathfrak{S}})[\boldsymbol{H}^{\Gamma^\perp}]\|_F &= \|\mathcal{P}_{\Gamma}(\mathcal{P}_{\mathsf{T}^\perp}\times\mathcal{P}_{\mathfrak{S}^c})[\boldsymbol{H}^{\Gamma^\perp}]\|_F \\
&\le \|(\mathcal{P}_{\mathsf{T}^\perp}\times\mathcal{P}_{\mathfrak{S}^c})[\boldsymbol{H}^{\Gamma^\perp}]\|_F.
\end{aligned}$$

Combining the previous two inequalities, we have

$$\|(\mathcal{P}_{\mathsf{T}}\times\mathcal{P}_{\mathfrak{S}})[\boldsymbol{H}^{\Gamma^\perp}]\|_F^2 \le 4\|(\mathcal{P}_{\mathsf{T}^\perp}\times\mathcal{P}_{\mathfrak{S}^c})[\boldsymbol{H}^{\Gamma^\perp}]\|_F^2,$$

which, together with (5.4.11), gives us the desired result,

$$\|\boldsymbol{H}^{\Gamma^\perp}\|_F^2 \le 5\|(\mathcal{P}_{\mathsf{T}^\perp} \times \mathcal{P}_{\mathfrak{S}^c})[\boldsymbol{H}^{\Gamma^\perp}]\|_F^2 \le 5 \times 16^2 n^2 \varepsilon^2. \tag{5.4.12}$$

Combining this bound with (5.4.8), we obtain the conclusion for Theorem 5.7. □

Notice that in the statement of the Theorem 5.7, the constant $C$ still depends on the dimension $n$, which arguably could still be removed or reduced. Indeed, with slightly stronger condition, say the magnitude of the low-rank component $\boldsymbol{L}_o$ is bounded, one could obtain better estimates by solving a Lasso-type program:

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 + \tfrac{\mu}{2} \|\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y}\|_F^2 \quad \text{subject to} \quad \|\boldsymbol{L}\|_\infty < \alpha. \tag{5.4.13}$$

With properly chosen weights $\lambda$ and $\mu$, the bound on the estimation error incurred by the noise can be significantly improved, compared to that in Theorem 5.7. The analysis and result are similar to those for stable sparse recovery (Theorem 3.31) and stable low-rank recovery (Theorem 4.20) where the noise is assumed to be random (Gaussian). For detailed analysis of the error bound for this program, we refer the reader to the work of [ANW12]. The same analysis also applies to the stable version of the outlier pursuit program (5.3.55):

$$\min_{\boldsymbol{L},\boldsymbol{O}} \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{O}\|_{2,1} + \tfrac{\mu}{2} \|\boldsymbol{L} + \boldsymbol{O} - \boldsymbol{Y}\|_F^2 \quad \text{subject to} \quad \|\boldsymbol{L}\|_\infty < \alpha. \tag{5.4.14}$$

## 5.5 Compressive Principal Component Pursuit

From the above sections, we saw that under fairly broad conditions, via convex optimization, a low-rank matrix $\boldsymbol{L}_o$ and the sparse matrix $\boldsymbol{S}_o$ can be recovered correctly if we observe fully their superposition $\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{S}_o$. This is possible because the pair $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ have far fewer degrees of freedom than the number of observations $n^2$. Since this target is so low-dimensional, it is natural to wonder whether it would be possible to recover it from an even smaller set of general linear measurements $\boldsymbol{Y}$. That is, are we are able to perform "compressive sensing" of a low-rank structure and a sparse model superimposed together. Mathematically, we assume the observations have the form:

$$\boldsymbol{Y} \;\doteq\; \mathcal{P}_{\mathsf{Q}}[\boldsymbol{L}_o + \boldsymbol{S}_o], \tag{5.5.1}$$

where $\mathsf{Q} \subseteq \mathbb{R}^{n_1 \times n_2}$ is a linear subspace, and $\mathcal{P}_{\mathsf{Q}}$ denotes the projection operator onto that subspace. In fact, this problem may arise whenever we observe a "deformed" version of certain 2D array $\boldsymbol{M}$, say $\boldsymbol{M} \circ \tau = \boldsymbol{L}_o + \boldsymbol{S}_o$ where $\tau$ is certain domain deformation. One natural approach to recover the deformation $\tau$ and the low-rank and sparse components is to linearize the above equation respect to $\tau$ and obtain the differential of the above equation at a given $\tau_o$:

$$\boldsymbol{M} \circ \tau_o + \boldsymbol{J} \circ d\tau \approx \boldsymbol{L}_o + \boldsymbol{S}_o,$$

where $\boldsymbol{J}$ is the Jacobian matrix and $d\tau$ is the infinitesimal deformation. To eliminate the unknown $d\tau$, let $\mathsf{Q}$ be the left kernel of the Jacobian $\boldsymbol{J}$, i.e., $\mathsf{Q}$ is a subspace spanned by all matrices $\mathsf{Q} \doteq \{\boldsymbol{Q} \mid \langle \boldsymbol{Q}, \boldsymbol{J} \rangle = 0\}$. So we have

$$\boldsymbol{Y} \doteq \mathcal{P}_{\mathsf{Q}}[\boldsymbol{M} \circ \tau_o] = \mathcal{P}_{\mathsf{Q}}[\boldsymbol{L}_o + \boldsymbol{S}_o].$$

Can we simultaneously recover the low-rank and sparse components correctly from highly compressive measurements via the natural convex program

$$\min \quad \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \mathcal{P}_{\mathsf{Q}}[\boldsymbol{L} + \boldsymbol{S}] = \boldsymbol{Y} \ ? \tag{5.5.2}$$

We call this convex program *Compressive Principal Component Pursuit*, or shortly *CPCP*. In this section, we study when this program can correctly recover $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$. As before, throughout this section, we assume the low-rank matrix $\boldsymbol{L}_o$ is $\nu$-incoherent and the support of the sparse component $\boldsymbol{S}_o$, say $\mathfrak{S}$, is (Bernoulli) random.

To recover both $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$ correctly, we must require measurements $\mathsf{Q}$ to be incoherent with *both* the low-rank and the sparse component. To ensure the incoherence property, we may assume that $\mathsf{Q}$ is a randomly chosen subspace in the matrix space $\mathbb{R}^{n_1 \times n_2}$.

More precisely, suppose the dimension of the subspace $\mathsf{Q}$ is $q$, and we assume $\mathsf{Q}$ is distributed according to the Haar measure on the Grassmannian $\mathbb{G}(\mathbb{R}^{m \times n}, q)$. On a more intuitive level, this means that $\mathsf{Q}$ is equal in distribution to the linear span of a collection of $q$ independent iid $\mathcal{N}(0,1)$ matrices. In notation more familiar from compressive sensing, we may let $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_q$ denote such a set of matrices, and define an operator $\mathcal{Q} : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^q$ via

$$\mathcal{Q}[\boldsymbol{M}] \ = \ \left(\langle \boldsymbol{Q}_1, \boldsymbol{M} \rangle, \ldots, \langle \boldsymbol{Q}_q, \boldsymbol{M} \rangle\right)^* \ \in \ \mathbb{R}^q. \tag{5.5.3}$$

Our analysis also pertains to the equivalent convex program:

$$\min \quad \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \mathcal{Q}[\boldsymbol{L} + \boldsymbol{S}] = \mathcal{Q}[\boldsymbol{L}_o + \boldsymbol{S}_o]. \tag{5.5.4}$$

Since $\mathcal{Q}$ has full rank $q$ almost surely, (5.5.4) and (5.5.2) are completely equivalent.

With these assumptions, the following theorem gives a tight bound on the number of (random) measurements required to correctly recover the pair $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ from $\mathcal{P}_{\mathsf{Q}}[\boldsymbol{L}_o + \boldsymbol{S}_o]$ via CPCP:

THEOREM 5.9 (Compressive PCP). *Let $\boldsymbol{L}_o, \boldsymbol{S}_o \in \mathbb{R}^{n_1 \times n_2}$, with $n_1 \geq n_2$, and suppose that $\boldsymbol{L}_o \neq \boldsymbol{0}$ is a rank-$r$, $\nu$-incoherent matrix with*

$$r \ \leq \ \frac{c_r n_2}{\nu \log^2 n_1}, \tag{5.5.5}$$

*and $\operatorname{sign}(\boldsymbol{S}_o)$ is iid Bernoulli-Rademacher with nonzero probability $\rho < c_\rho$. Let $\mathsf{Q} \subset \mathbb{R}^{n_1 \times n_2}$ be a random subspace of dimension*

$$\dim(\mathsf{Q}) \ \geq \ C_{\mathsf{Q}} \cdot (\rho n_1 n_2 + n_1 r) \cdot \log^2 n_1 \tag{5.5.6}$$

*distributed according to the Haar measure, probabilistically independent of* $\operatorname{sign}(\boldsymbol{S}_o)$. *Then with probability at least* $1 - Cn_1^{-9}$ *in* $(\operatorname{sign}(\boldsymbol{S}_o), \mathsf{Q})$, *the solution to*

$$\min \quad \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \quad \textit{subject to} \quad \mathcal{P}_{\mathsf{Q}}[\boldsymbol{L} + \boldsymbol{S}] = \mathcal{P}_{\mathsf{Q}}[\boldsymbol{L}_o + \boldsymbol{S}_o] \tag{5.5.7}$$

*with* $\lambda = 1/\sqrt{n_1}$ *is unique, and equal to* $(\boldsymbol{L}_o, \boldsymbol{S}_o)$. *Above,* $c_r, c_\rho, C_{\mathsf{Q}}, C$ *are positive numerical constants.*

Here, the magnitudes of the nonzeros in $\boldsymbol{S}_o$ are arbitrary, and no randomness is assumed in $\boldsymbol{L}_o$. The randomness in this result occurs in the sign and support pattern of $\boldsymbol{S}_o$ and in the measurements $\mathsf{Q}$. The bounds on $r$ and $\rho$ essentially match those of PCP for the fully observed case, possibly with different constants. So, again, $r$ and $\|\boldsymbol{S}_o\|_0$ can be rather large. On the other hand, when these quantities are small, the bound on $\dim(\mathsf{Q})$ ensures that the number of measurements needed for accurate recovery is also commensurately small. In fact, this result can be obtained via general arguments that can also be applied to other compressive sensing and decomposition problems of a family of low-dimensional structures in high-dimensional space (as we will introduce in the next Chapter). Since the approach and techniques of the proof are rather similar to that for the PCP, we here do not elaborate and instead point interested readers to [WGMM13] for a complete proof.

## 5.6 Matrix Completion with Corrupted Entries

We have seen that the main result on PCP (Theorem 5.3) asserts that it is possible to recover a low-rank matrix even though a significant fraction of its entries are corrupted. Furthermore, the above section reveals that both the low-rank and sparse components can be recovered even if only a small number of general linear measurements of the corrupted matrix $\boldsymbol{Y}$ are given.

In many applications, however, the (linear) measurements of the corrupted matrix available to us are not general and have very peculiar structures. For instance, we only get to see a small fraction of the entires of $\boldsymbol{Y}$, and the remaining of the entries may be missing. For instance, in the case of taking face images under different illuminations, we can use random corruptions to model pixels associated with surfaces that violate the Lambertian property (such as specular surfaces); and we may assume the intensities of pixels which are blocked from light sources (in the shadow areas) are missing. Hence the data (matrix) have both corrupted and missing entries. Can we still expect to recover the low-rank matrix? As the observations are no longer general (e.g., they are not incoherent with the sparse term $\boldsymbol{S}_o$), results from the above section do not directly apply to the situations here. This section addresses this problem.

To be precise, as before, we assume $\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{S}_o$ is a low-rank matrix $\boldsymbol{L}_o$ corrupted by a sparse matrix $\boldsymbol{S}_o$ whose support $\mathfrak{S}$ is distributed as $\mathfrak{S} \sim \operatorname{Ber}(\rho_s)$ for some small constant $\rho_s < 1$.

We further assume we only observe a small fraction $\rho_o$ of the entries of $\boldsymbol{Y}$. Let $\mathsf{O}$ be a support distributed as $\mathsf{O} \sim Ber(\rho_o)$, where $\mathsf{O}$ stands for "observed" entries. We may assume $\mathfrak{S}$ and $\mathsf{O}$ are independent Bernoulli variables.

Let $\mathcal{P}_\mathsf{O}$ be the orthogonal projection onto the linear space of matrices supported on $\mathsf{O} \subset [n_1] \times [n_2]$,

$$\mathcal{P}_\mathsf{O}[\boldsymbol{X}] = \begin{cases} X_{ij}, & (i,j) \in \mathsf{O}, \\ 0, & (i,j) \notin \mathsf{O}. \end{cases}$$

Then imagine we only have available those entries of $\boldsymbol{L}_o + \boldsymbol{S}_o$ such that $(i,j) \in \mathsf{O} \subset [n_1] \times [n_2]$, which we conveniently write as

$$\mathcal{P}_\mathsf{O}([\boldsymbol{Y}]) = \mathcal{P}_\mathsf{O}[\boldsymbol{L}_o + \boldsymbol{S}_o] = \mathcal{P}_\mathsf{O}[\boldsymbol{L}_o] + \boldsymbol{S}'_o.$$

This models the following problem: we wish to recover $\boldsymbol{L}_o$ but only see a few entries about $\boldsymbol{L}_o$, and among those a fraction happens to be corrupted, and we of course do not know which one. As is easily seen, this is an extension to the Matrix Completion problem of the previous chapter, which seeks to recover $\boldsymbol{L}_o$ from under-sampled but otherwise perfect data $\mathcal{P}_\mathsf{O}[\boldsymbol{L}_o]$; and this is also an extension to the RPCA problem as there we only see a small fraction of the corrupted matrix $\boldsymbol{Y}$.

We propose recovering $\boldsymbol{L}_o$ (and $\boldsymbol{S}'_o$) by solving the following problem:

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \\ \text{subject to} & \mathcal{P}_\mathsf{O}[\boldsymbol{L} + \boldsymbol{S}] = \mathcal{P}_\mathsf{O}[\boldsymbol{Y}]. \end{array} \tag{5.6.1}$$

In words, among all decompositions matching the available data, Principal Component Pursuit finds the one that minimizes the weighted combination of the nuclear norm, and of the $\ell^1$ norm. Our observation is that under some conditions, this simple approach recovers the low-rank component exactly. In fact, the techniques developed here establish this result:

THEOREM 5.10 (Matrix Completion with Corruptions). *Suppose $\boldsymbol{L}_o$ is $n \times n$, obeys the conditions (5.3.1)–(5.3.2). Suppose $\rho_0 > C_0 \frac{\nu r \log^2 n}{n}$ and $\rho_s \leq C_s$, and let $\lambda = \frac{1}{\sqrt{\rho_0 n \log n}}$. Then the optimal solution to the convex program (5.6.1) is exactly $\boldsymbol{L}_o$ and $\boldsymbol{S}'_o$ with probability at least $1 - Cn^{-3}$ for some constant $C$, provided the constants $C_0$ is large enough and $C_s$ is small enough.*

In short, perfect recovery from incomplete and corrupted entries is possible by convex optimization. The approach and techniques of the proof are similar to that of the PCP, we refer interested readers for a complete and rigorous proof to the work of [Li13].

On the one hand, this result extends the RPCA result in the following way: If all the entries are available, i.e. $\rho_0 = 1$, the above theorem guarantees perfect recovery as long as $1 > C_0 \frac{\nu r \log^2 n}{n}$ or $r < C_0^{-1} n \nu^{-1} (\log n)^{-2}$ for small enough $C_0^{-1}$, which is exactly Theorem 5.3. The choice of $\lambda$ here reduces to the case $\lambda = \frac{1}{\sqrt{n \log n}}$ for dense error correction discussed in Section 5.3.3. On the other

hand, this result extends the Matrix Completion results developed in the previous chapter too. Indeed, if $\rho_s = 0$, we have a pure matrix completion problem from about $\rho_0$ fraction of entries, and the above theorem guarantees perfect recovery as long as $\rho_0 > C_0 \frac{\nu r \log^2 n}{n}$ for large enough $C_0$, which is exactly Theorem 4.26.

We remark that the recovery is exact, however, via a different algorithm. To be sure, in matrix completion one typically minimizes the nuclear norm $\|\boldsymbol{L}\|_*$ subject to the constraint $\mathcal{P}_{\mathsf{O}}[\boldsymbol{L}] = \mathcal{P}_{\mathsf{O}}[\boldsymbol{L}_o]$. Here, our program would solve

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \\ \text{subject to} & \mathcal{P}_{\mathsf{O}}[\boldsymbol{L} + \boldsymbol{S}] = \mathcal{P}_{\mathsf{O}}[\boldsymbol{L}_o], \end{array} \tag{5.6.2}$$

and return $\hat{\boldsymbol{L}} = \boldsymbol{L}_o$, $\hat{\boldsymbol{S}} = \boldsymbol{0}$! In this context, Theorem 5.10 implies that matrix completion is robust vis a vis gross errors.

## 5.7 Summary

In this chapter we have studied the problem of simultaneously recovering a low-rank matrix $\boldsymbol{L}_o$ and a sparse matrix $\boldsymbol{S}_o$ from their superposition:

$$\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{S}_o \quad \in \mathbb{R}^{n \times n}.$$

This problem can be viewed as a *robust principal component analysis* (RPCA) problem – how to robustly estimate a low-dimensional subspace while being robust to gross (random) corruptions in the data. We have learned that under certain benign *incoherence* conditions between $\boldsymbol{L}_o$ and $\boldsymbol{S}_o$, the two matrices can be correctly recovered with high probability from minimizing a weighted sum of nuclear norm of $\boldsymbol{L}_o$ and $\ell^1$ norm of $\boldsymbol{S}_o$, also known as *principal component pursuit* (PCP):

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{L}\|_* + \frac{1}{\sqrt{n}} \|\boldsymbol{S}\|_1 \\ \text{subject to} & \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}, \end{array}$$

as long as the following conditions are satisfied:

$$|\boldsymbol{S}_o| \le \rho_s n^2, \quad \text{rank}\,(\boldsymbol{L}_o) = O(n \log^{-2} n),$$

where $\rho_s > 0$ is some constant factor.

We have also sturdied how the basic PCP program naturally extends to several important variants of the RPCA problem, including when there are additive (Gaussian) noises $\boldsymbol{Z}_o$, with randomly projected measurements on a subspace $\mathsf{Q}$, and when only entries in a subset $\mathsf{O}$ are observed:

$$\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{S}_o + \boldsymbol{Z}_o, \quad \mathcal{P}_{\mathsf{Q}}[\boldsymbol{Y}] = \mathcal{P}_{\mathsf{Q}}[\boldsymbol{L}_o + \boldsymbol{S}_o], \quad \mathcal{P}_{\mathsf{O}}[\boldsymbol{Y}] = \mathcal{P}_{\mathsf{O}}[\boldsymbol{L}_o + \boldsymbol{S}_o],$$

respectively.

As we have seen from Chapter 2 to this Chapter 4, we have developed in parallel the basic theory and algorithms for recovering sparse signals or low-rank matrices, via convex optimization. In this chapter, we also see how these two

| Sparse v.s. Low-rank | Sparse Vector | Low-rank Matrix |
|---|---|---|
| Low-dimensionality of | individual signal $\boldsymbol{x}$ | a set of signals $\boldsymbol{X}$ |
| Low-dim measure | $\ell^0$ norm $\|\boldsymbol{x}\|_0$ | rank $(\boldsymbol{X})$ |
| Convex surrogate | $\ell^1$ norm $\|\boldsymbol{x}\|_1$ | nuclear norm $\|\boldsymbol{X}\|_*$ |
| Compressive sensing | $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ | $\boldsymbol{Y} = \mathcal{A}(\boldsymbol{X})$ |
| Stable recovery | $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{z}$ | $\boldsymbol{Y} = \mathcal{A}(\boldsymbol{X}) + \boldsymbol{Z}$ |
| Error correction | $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}$ | $\boldsymbol{Y} = \mathcal{A}(\boldsymbol{X}) + \boldsymbol{E}$ |
| Recovery of mixed structures | $\mathcal{P}_{\mathsf{Q}}[\boldsymbol{Y}] = \mathcal{P}_{\mathsf{Q}}[\boldsymbol{L}_o + \boldsymbol{S}_o] + \boldsymbol{Z}$ | |

**Table 5.2** Comparison between sparse vectors and low-rank matrices.

low-dimensional models can be combined together to model more sophisticated structures in the data. Table 5.2 summarizes the similarity of these two most basic low-dimensional models studied so far. In the next chapter, we will see how the same ideas generalize to broader family of low-dimensional models.

## 5.8 Notes

*Second Order Convex Methods.*
For small problem sizes, the above principal component pursuit program can be solved using off-the-shelf tools such as interior point methods [GB14]. This method was initially suggested for rank minimization in [FHB04, RFP10] and for low-rank and sparse decomposition [CSPW09]. However, despite their superior convergence rates, interior point methods are typically limited to small-size problems, say $n < 100$, due to the $O(n^6)$ complexity of computing a step direction.

*First Order Convex Methods.*
The limited scalability of interior point methods has inspired a recent flurry of work on first-order methods. Exploiting an analogy with iterative thresholding algorithms for $\ell^1$-minimization [HYZ08, YOGD08], the work of [CCS08] has developed an algorithm that performs nuclear-norm minimization by repeatedly shrinking the *singular values* of an appropriate matrix, essentially reducing the complexity of each iteration to the cost of an SVD. However, for our low-rank and sparse decomposition problem, this form of iterative thresholding converges slowly, requiring up to $10^4$ iterations. [GM09] suggests to improve convergence using continuation techniques, and has demonstrated how Bregman iterations [YOGD08] can be applied to nuclear norm minimization.

*Accelerated Methods.*
The convergence of iterative thresholding can be significantly improved using ideas from Nesterov's accelerated first-order algorithm for smooth minimization [Nes83], which was extended to non-smooth functions in [Nes05, Nes07], and later successfully applied to $\ell^1$-minimization by [BT09, BBC09]. Based on [BT09], [TY09] has developed a proximal gradient algorithm for matrix completion which they has named as *Accelerated Proximal Gradient (APG)*. Around the same time, a very similar APG algorithm was suggested for low-rank and sparse decomposition in [LGW+09]. In theory, these algorithms inherit the optimal $O(1/k^2)$ convergence rate of the accelerated methods. Empirical evidences suggest that these algorithms can solve the convex PCP problem at least 50 times faster than straightforward iterative thresholding (see [LGW+09]).

*Augmented Lagrange Multipler Methods.*
However, despite their good convergence guarantees, the practical performance of APG depends strongly on the design of a good continuation scheme. Generic continuation does not guarantee good accuracy and convergence across a wide range of problem settings.[10] In this chapter, we have instead chosen to solve the convex PCP problem (5.2.2) using an augmented Lagrange multiplier (ALM) algorithm introduced in [LCWM09, YY09]. In our experience, ALM achieves much higher accuracy than APG, in fewer iterations. It works stably across a wide range of problem settings with no tuning of parameters. Moreover we observe an appealing (empirical) property: the rank of the iterates often remains bounded by $\operatorname{rank}(\boldsymbol{L}_o)$ throughout the optimization, allowing them to be computed efficiently. APG, on the other hand, does not have this property.

A systematic development of all these convex optimization methods for recovering both sparse and low-rank models will be given in Chapter 8. We will also study natural nonconvex formulations of the low-rank and sparse recovery problems in Chapter 7 and develop efficient algorithms for the nonconvex programs in Chapter 9.

## 5.9 Exercises

5.1 (RPCA as an Underdetermined Linear Inverse Problem). *Consider the space* $\mathsf{V}$ *of pairs* $(\boldsymbol{L}, \boldsymbol{S}) \in \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$. *This is a vector space over* $\mathbb{R}$. *Consider the function*

$$\|\cdot\|_{\diamond} : \mathsf{V} \to \mathbb{R} \tag{5.9.1}$$

*via*

$$\|(\boldsymbol{L}, \boldsymbol{S})\|_{\diamond} = \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 . \tag{5.9.2}$$

[10] In our experience, the optimal choice may depend on the relative magnitudes of the $\boldsymbol{L}$ and $\boldsymbol{S}$ terms and the sparsity of $\boldsymbol{S}$.

*Show that $\|\cdot\|_\diamond$ is a norm on $\mathsf{V}$, by showing that it satisfies the axioms of a norm. For $\boldsymbol{x} = (\boldsymbol{L}, \boldsymbol{S})$ in $\mathsf{V}$, let $\mathcal{A}[\boldsymbol{x}] = \boldsymbol{L} + \boldsymbol{S}$. Interpret the PCP problem as*

$$\min \|\boldsymbol{x}\|_\diamond \quad \textit{subject to} \quad \mathcal{A}[\boldsymbol{x}] = \boldsymbol{Y}. \tag{5.9.3}$$

5.2 (Matrix Rigidity). *Give an example of "rigid" matrices in terms of Definition 5.1 and give some examples of "soft" matrices. Can you identify the main difference between rigid and soft matrices? Propose an algorithm that can compute the rigidity of any given matrix. Discuss the worst computational complexity of the proposed algorithm.*

5.3 (Find Maximum Low-rank Matrix). *Given an $n \times n$ matrix $\boldsymbol{M}$, design an algorithm that finds the largest submatrix $\boldsymbol{S}$ such that*

$$\operatorname{rank}(\boldsymbol{S}) \leq r$$

*for some given small rank $r$. Discuss the complexity of your algorithm.*

5.4 (Planted Clique via RPCA). *In the* planted clique *problem (see Definition 5.2), we are given a large graph $\mathcal{G}$ of $n$ nodes. Now suppose it has a largest clique of size $n_o$. Consider the adjacent matrix $\boldsymbol{A}$ of the graph $\mathcal{G}$. Then we have:*

$$\boldsymbol{A} = \boldsymbol{L}_o + \boldsymbol{S}_o,$$

*where $\boldsymbol{L}_o$ is an $n_o \times n_o$ rank-1 matrix whose entries are all ones, and $\boldsymbol{S}_o$ is a matrix with at least half entries are zeros. Determine (i) $\operatorname{rank}(\boldsymbol{L}_o)$, (ii) $\nu(\boldsymbol{L}_o)$ according (5.3.1), (iii) $\nu_\infty(\boldsymbol{L}_o)$ according to (5.3.2). How big does the clique $\mathsf{C}$ need to be for PCP to succeed?*

5.5 (Lower Bounds from Planted Clique). *Show the necessity of the condition (5.3.2) based on the hardness conjecture of finding the largest clique in a graph.*

5.6 (Finding Planted Cliques). *Develop an experiment to test how the PCP algorithm works on the planted clique problem. Is the working range consistent with the hardness conjecture?*

5.7 (Low-Rank Representation*). *Low-Rank Representation (LRR) [LLY+13] is an extension of RPCA. It aims to solve the problem of clustering a set of $n$ data points in $\mathbb{R}^m$: $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$ that are drawn from a union of multiple low-dimensional subspaces, with potential noise and corruption. The key idea is to find a self-expressive representation for $\boldsymbol{X} = \boldsymbol{X}\boldsymbol{Z}$. But to avoid using each point $\boldsymbol{x}_i$ to represent itself, we enforce points from the same subspace to form a "cluster". In other words, the coefficients $\boldsymbol{Z}$ is preferably a low-rank matrix, as we have discussed in the community discovery problem. To account for possible sparse corruptions or outliers (points sampled outside of these subspaces), we solve $\boldsymbol{X} = \boldsymbol{X}\boldsymbol{Z} + \boldsymbol{E}$ with $\boldsymbol{E}$ being sparse or column sparse. This leads to the following program:*

$$\min_{\boldsymbol{Z},\boldsymbol{E}} \|\boldsymbol{Z}\|_* + \lambda \|\boldsymbol{E}\|_{2,1} \quad \textit{subject to} \quad \boldsymbol{X} = \boldsymbol{X}\boldsymbol{Z} + \boldsymbol{E}.$$

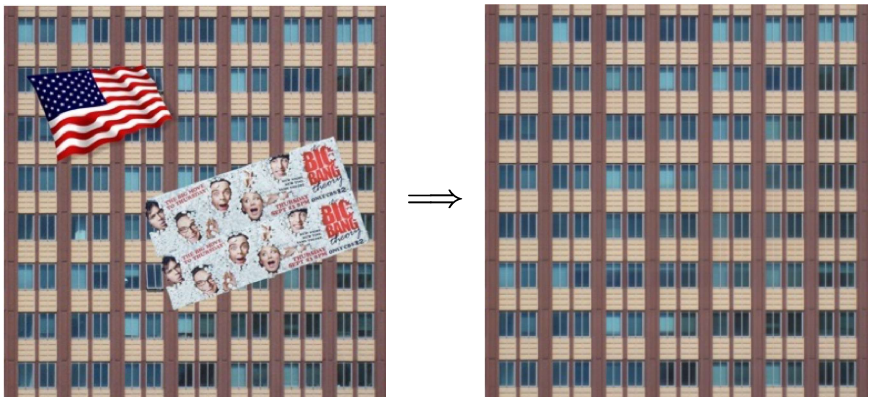


**Figure 5.6** Write a program to take an input as the image on the left with occlusion and output a clean image on the right. Note that the image on the right is indeed a recovered image from the left by a program like PCP.

*Program a MATLAB function for LRR and use it for clustering a set of frontal face images from several people, say using the Yale face dataset [Ext].*

5.8 (Background Subtraction*). *Code a MATLAB program that utilizes Robust PCA to separate the foreground images and background images in video sequences captured by stationary cameras.*

5.9 (Robust Texture Inpainting*). *Code a MATLAB program that utilizes Robust PCA to perform texture inpainting to compensate corrupted texture images without knowing the location of the corrupted pixels:*

```
(I_hat, E) = robust_inpainting(I),
```

*where* `I` *is the input texture image,* `I_hat` *is the recovered texture image, and* `E` *is the detected corruption in the same image space. See Figure fig:inpainting as an example. To test how good your algorithm is, try different types and sizes of occlusion on the input images. Try any ideas that may further improve the performance of your algorithm, say by taking into account additional structures of the possible occlusion, in addition to being sparse.*

5.10 (A Monotonicity Property of PCP). *Call* $\boldsymbol{S}'$ *a* trimmed version *of* $\boldsymbol{S}$ *if* $\text{supp}(\boldsymbol{S}') \subset \text{supp}(\boldsymbol{S})$ *and* $\boldsymbol{S}'_{ij} = \boldsymbol{S}_{ij}$ *whenever* $\boldsymbol{S}'_{ij} \neq 0$. *Prove that whenever* $(\boldsymbol{L}_o, \boldsymbol{S}_o)$ *is the unique optimal solution to the PCP problem with data* $\boldsymbol{Y}_o = \boldsymbol{L}_o + \boldsymbol{S}_o$, $(\boldsymbol{L}_o, \boldsymbol{S}')$ *is the unique optimal solution to the PCP problem with data* $\boldsymbol{Y}' = \boldsymbol{L}_o + \boldsymbol{S}'$.

5.11 (Derandomizing the Signs). *In this exercise, we "derandomize" the signs in the RPCA problem, using the elimination property from Exercise 5.10. Suppose*

*that for a given $\boldsymbol{L}_o$, RPCA succeeds with high probability when* $\operatorname{sign}(\boldsymbol{S}_o)$ *is a* $\text{Bernoulli}(\rho_s)$*-Rademacher matrix. Prove that with at least the same probability, RPCA succeeds when* $\mathfrak{S} \sim_{\text{iid}} \text{Ber}(\rho_s/2)$*, and* $\operatorname{sign}(\boldsymbol{S}_o) = \mathcal{P}_{\mathfrak{S}}[\bar{\boldsymbol{\Sigma}}]$ *for some fixed matrix of signs* $\bar{\boldsymbol{\Sigma}} \in \{\pm 1\}^{n \times n}$.

5.12. *Show that for two projection operators* $\mathcal{P}_{\mathfrak{S}}, \mathcal{P}_{\mathsf{T}}$*, we have:*

$$\|\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\| \;=\; \|\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\|^{1/2}. \tag{5.9.4}$$

5.13 (Least Squares for Dual Certificates). *To prove Theorem 5.3, in Lemma 5.6 we used the method of least squares (minimum energy) to construct a dual certificate* $\boldsymbol{\Lambda}_{\boldsymbol{S}}$ *satisfying* $\mathcal{P}_{\mathfrak{S}}[\boldsymbol{\Lambda}_{\boldsymbol{S}}] = \lambda\boldsymbol{\Sigma}_o$ *and* $\mathcal{P}_{\mathsf{T}}[\boldsymbol{\Lambda}_{\boldsymbol{S}}] = \mathbf{0}$*. We asserted that whenever* $\|\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\| < 1$*, the solution to the problem (5.3.29):*

$$\min \left\|\tilde{\boldsymbol{\Lambda}}\right\|_F^2 \quad \textit{subject to} \quad \mathcal{P}_{\mathfrak{S}}[\tilde{\boldsymbol{\Lambda}}] = \lambda\boldsymbol{\Sigma}_o, \quad \mathcal{P}_{\mathsf{T}}[\tilde{\boldsymbol{\Lambda}}] = \mathbf{0} \tag{5.9.5}$$

*is given in closed form by the Neumann series (5.3.30):*

$$\boldsymbol{\Lambda}_{\boldsymbol{S}} = \lambda\mathcal{P}_{\mathsf{T}^\perp}\sum_{k=0}^{\infty}(\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\mathcal{P}_{\mathfrak{S}})^k[\boldsymbol{\Sigma}_o]. \tag{5.9.6}$$

*Show that (i) when* $\|\mathcal{P}_{\mathfrak{S}}\mathcal{P}_{\mathsf{T}}\| < 1$*, the infinite summation in (5.9.6) converges, and (ii) that* $\boldsymbol{\Lambda}_{\boldsymbol{S}}$ *solves (5.9.5).*

5.14 (Dense Errors with Random Signs). *Prove that with an appropriate choice of* $\lambda$*, PCP can handle any constant fraction* $\rho_s < 1$ *of errors.*

# 6 Recovering General Low-Dimensional Models

"*An idea which can be used once is a trick. If one can use it more than once it becomes a method.*"
– George Pólya and Gábor Szegö, *Problems and Theorems in Analysis I*

In the first five chapters of this book, we introduced two main families of low-dimensional models for high-dimensional data: sparse models and low-rank models. In Chapter 5, we saw how we could combine these basic models to accommodate data matrices that are superpositions of sparse and low-rank matrices. This generalization allowed us to model richer classes of data, including data containing erroneous observations. In this chapter, we further generalize these basic models to a situation in which the object of interest consists of a superposition of a few elements from some set of "atoms" (Section 6.1). This construction is general enough to include all of the models discussed so far, as well as several other models of practical importance. With this general idea in mind, we then discuss unified approaches to studying the power of low-dimensional signal models for estimation, measured in terms of the number of measurements needed for exact recovery or recovery with sparse errors (Section 6.2). These analyses generalize and unify the ideas developed over the earlier chapters, and offer definitive results on the power of convex relaxation. Finally, in Section 6.3, we discuss limitations of convex relaxation, which in some situations will force us to consider nonconvex alternatives, to be studied in later chapters.

## 6.1 Concise Signal Models

We have considered two models of low-dimensional signal structure. A *sparse vector* $\boldsymbol{x} \in \mathbb{R}^n$ is a superposition of a few coordinate basis vectors:

$$\boldsymbol{x} = \sum_{i \in \mathsf{I} \subset [n]} x_i \boldsymbol{e}_i. \tag{6.1.1}$$

A *low-rank matrix* $\boldsymbol{X} \in \mathbb{R}^{n \times n}$ is a superposition of just a few rank-one matrices $\boldsymbol{u}_i \boldsymbol{v}_i^*$:

$$\boldsymbol{X} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^*, \quad r < n. \tag{6.1.2}$$

The standard basis vectors form the elementary components for constructing sparse vectors. The rank-one matrices form the elementary components for constructing low-rank matrices.

### 6.1.1 Atomic Sets and Examples

These are two specific examples of a more general situation, in which the signal $\boldsymbol{x}$ of interest can be expressed as a superposition of a few elementary components, selected from some set $\mathcal{D}$:

$$\boldsymbol{x} = \sum_i \alpha_i \boldsymbol{d}_i, \quad \boldsymbol{d}_i \in \mathcal{D}. \tag{6.1.3}$$

For sparse vectors, we can take

$$\mathcal{D} = \mathcal{D}_{\text{sparse}} \equiv \{\pm \boldsymbol{e}_i \mid i = 1, \ldots, n\}. \tag{6.1.4}$$

For low-rank matrices, we can take

$$\mathcal{D} = \mathcal{D}_{\text{low-rank}} \equiv \{\boldsymbol{u}\boldsymbol{v}^* \mid \|\boldsymbol{u}\|_2 = \|\boldsymbol{v}\|_2 = 1\}. \tag{6.1.5}$$

The set $\mathcal{D}$ is sometimes referred to as an *atomic set* – it consists of a collection of elementary components ("atoms") from which the structured signal of interest can be constructed. In the literature, such an atomic set is often called a "dictionary," hence the capital letter $\mathcal{D}$. Here, for simplicity, we assume the dictionary $\mathcal{D}$ is already known or given in advance. In Chapter 7, we will study how to learn the dictionary from data when it is not known ahead of time.

There are at least two reasons to consider the general notion of an atomic set. The first is that it gives a unified way of thinking about the models that we have already studied. The second is that it allows us to model other structures of practical interest. We describe a few examples below; Exercises 6.1-6.2 develop several others.

*Column Sparse Matrices.*
In Chapter 5, we described how to estimate an underlying low-rank matrix $\boldsymbol{L}$ even in the presence of grossly corrupted *observations* (or entries), which we modeled using a sparse matrix $\boldsymbol{S}$. In statistical applications, a different type of corruption can occur: some *data samples* (or vectors) may be outliers. Hence, some columns of the data matrix $\boldsymbol{Y}$ may be entirely corrupted. We can model this situation as

$$\boldsymbol{Y} = \boldsymbol{L} + \boldsymbol{C}, \tag{6.1.6}$$

where $\boldsymbol{C} = [\boldsymbol{c}_1 \mid \boldsymbol{c}_2 \mid \cdots \mid \boldsymbol{c}_{n_2}]$ is a matrix whose column $\boldsymbol{c}_i$ is nonzero if and only if the $i$-th sample $\boldsymbol{y}_i$ is an outlier (e.g., see the work [XCS12]).

In this situation, we can write

$$\mathcal{D} = \mathcal{D}_{\text{column sparse}} \equiv \{\boldsymbol{u}\boldsymbol{e}_i^* \mid \boldsymbol{u} \in \mathbb{R}^n, \|\boldsymbol{u}\|_2 = 1, i \in \{1, \ldots, n_2\}\}. \tag{6.1.7}$$

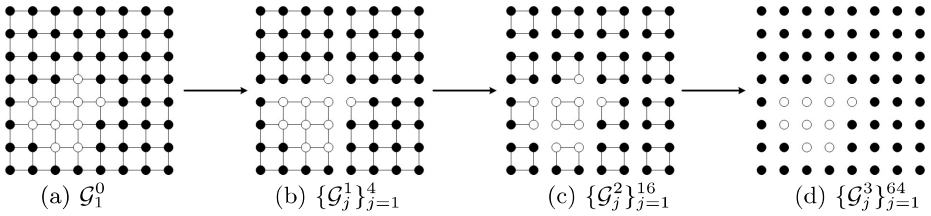



**Figure 6.1** Illustration of a four-level hierarchical tree group structure defined on a 2D grid of pixels of an image. Each circle represents a pixel, and connected circles represent a node/group in the tree. An $8 \times 8$ group in (a) is divided into 4 sub-group in (b) according to spatial continuity, and each sub-group can be viewed as a child node of (a). The similar relation goes from (b) to (c), and from (c) to (d). Black circles represents a pixel with zero value and white circles are nonzero.

If $\mathsf{I} \subseteq [n]$ is the set of indices of outliers, we can write

$$\boldsymbol{C} = \sum_{i \in \mathsf{I}} \alpha_i \boldsymbol{D}_i, \tag{6.1.8}$$

with $\boldsymbol{D}_i = \frac{\boldsymbol{c}_i}{\|\boldsymbol{c}_i\|_2} \boldsymbol{e}_i^* \in \mathcal{D}$ and $\alpha_i = \|\boldsymbol{c}_i\|_2$.

*Spatially Continuous Sparse Patterns*

Besides column-wise sparsity, for matrices $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$, we may consider atoms of the more general form $\boldsymbol{X}_\mathsf{I}$ with support $\mathsf{I} \subset [n_1] \times [n_2]$ and $\|\boldsymbol{X}_\mathsf{I}\|_p = 1$ for some norm $\|\cdot\|_p$. Popular choices of the norm include $p = 2$ or $p = \infty$. In theory, we may choose any set of supports

$$\mathcal{G} \doteq \{\mathsf{I}_i, i = 1, \ldots, N\}$$

for the atomic set. For instance, if $\mathcal{G}$ consists of supports representing columns of the matrix and $p = 2$, it recovers the above column-sparse atomic set. But if we view a matrix as a 2D grid of pixels for an image, we may select an atomic set that promotes spatial continuity of the image:

$$\mathcal{D}_{\text{spatial continuous}} \equiv \left\{ \boldsymbol{X}_\mathsf{I} \mid \boldsymbol{X}_\mathsf{I} \in \mathbb{R}^{n_1 \times n_2}, \|\boldsymbol{X}_\mathsf{I}\|_p = 1, \mathsf{I} \in \mathcal{G} \right\} \tag{6.1.9}$$

with $\mathcal{G}$ containing supports that are spatially adjacent. One such choice consists of all $8 \times 8$, $4 \times 4$, $2 \times 2$, and $1 \times 1$ subgrids. As shown in Figure 6.1, these support sets form a natural tree structure with $8 \times 8$ patches as root nodes, denoted as $\mathcal{G}^0$, and then branches into groups of smaller patches, with $\mathcal{G}^i$ indicate patches after $i$ partitions. This choice of atomic set promotes sparse patterns that are spatially continuous in terms of the grid topology. For instance, in applications to robust face recognition [JCM12], such a choice of atomic set was used to model spatially continuous occlusions, say due to wearing a sunglasses or a mask.

*Simultaneously Sparse and Low-rank Matrices.*
Another important low-dimensional model for matrices is the simultaneously sparse and low-rank matrices. These matrices arise naturally in applications in which we wish to find a low-rank approximation to a data matrix which uses only a few features (sparse PCA), or when we want to find small but densely connected communities in a large graph (community detection). They also arise naturally in modeling imagery data such as regular textures (see Chapter 15), videos, and hyper-spectral images, which exhibit both low-rank *and* sparsity are present. For example, videos may be low-rank along the time axis; since each frame of the video is also a natural image, individual frames should be sparse in an appropriate basis.

We can idealize this situation a bit by considering matrices $\boldsymbol{X} \in \mathbb{R}^{n \times n}$ whose nonzero entries are populated on a single block of size $k \times k$ (so $\|\boldsymbol{X}\|_0 \leq k^2 \ll n^2$) *and* whose rank is substantially smaller than $k$. Such matrices can be constructed using the atomic set

$$\mathcal{D} = \mathcal{D}_{\text{sparse and low-rank}} \equiv \{\boldsymbol{u}\boldsymbol{v}^* \mid \|\boldsymbol{u}\|_2 = \|\boldsymbol{v}\|_2 = 1,\ \|\boldsymbol{u}\|_0 \leq k, \|\boldsymbol{v}\|_0 \leq k\}\,. \tag{6.1.10}$$

This class of models is of fundamental importance for both theory and applications. Under the hardness of planted clique (see Section 5.1.2), this class of models is hard. To the best of our knowledge, there is no computationally tractable tight convex relaxation to this class of models, as we will discuss further in Section 6.3.

*Low-rank Tensors.*
Another important example of a low-dimensional model which does not admit efficient algorithms is high-order tensors. The rank($\boldsymbol{\mathcal{X}}$) of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_K}$ is the smallest number $r$ of components in an expression

$$\boldsymbol{\mathcal{X}} = \sum_{i=1}^{r} \boldsymbol{u}_i \otimes \boldsymbol{v}_i \otimes \cdots \otimes \boldsymbol{w}_i. \tag{6.1.11}$$

This is known as the Candecomp-Parafac (CP) rank. There are several different notions of tensor rank, which may be appropriate in different situations [KB09].

A *low-rank* tensor $\boldsymbol{\mathcal{X}}$ can be expressed as a superposition of just a few elements from the atomic set

$$\mathcal{D} = \mathcal{D}_{\text{low-rank tensor}} \equiv \{\boldsymbol{u} \otimes \boldsymbol{v} \otimes \cdots \otimes \boldsymbol{w} \mid \|\boldsymbol{u}\|_2 = \|\boldsymbol{v}\|_2 = \|\boldsymbol{w}\|_2 = 1\}\,. \tag{6.1.12}$$

Notice that, when the order of the tensor is $K = 2$, this generalizes the atomic set for low-rank matrices which we discussed above.

This class of models is very important for applications. However, there is an important distinction from the matrix case: for tensors of order $K \geq 3$, problems such as computing the rank, or finding a decomposition of the form (6.1.11) are NP-hard [HL13]. The low-rank tensors are our first example of a low-dimensional signal model which *does not* admit tight efficient algorithms! We will discuss this matter further in Section 6.3

*Sinusoids with Continuous Frequency.*
In applications such as RF communications and line spectrum estimation in scientific imaging, we encounter signals which have relatively narrow support in the Fourier domain. The *multi-tone* signals are a useful idealization of this situation: a multitone signal is a superposition of a few complex exponentials:

$$\boldsymbol{x} = \sum_i \alpha_i \xi(\omega_i, \phi_i) \in \mathbb{C}^N, \tag{6.1.13}$$

where

$$\xi(\omega, \phi)[n] = \exp\left(2\pi \mathrm{i}\left(\omega n + \phi\right)\right). \tag{6.1.14}$$

For such multi-tone signals, we can take

$$\mathcal{D} = \{\xi(\omega, \phi) \mid \omega \in [0,1],\ \phi \in [0,1]\}. \tag{6.1.15}$$

The model (6.1.14) is a sparse model, but the atomic set (dictionary) is continuous! Surprisingly (and unlike our previous two examples) in many situations, it *is* possible to compute efficiently with such a continuous dictionary. The advantage of this formulation is that it avoids artifacts associated with discretizing the set of frequencies.

### 6.1.2 Atomic Norm Minimization for Structured Signals

In the previous section, we saw how to use the notion of an atomic set to capture various types of low-dimensional signal structure. The value of these low-dimensional signal models is that they can render ill-posed inverse problems well-posed: instead of requiring a number of observations which is proportional to the ambient dimension $n$, we may hope to recover the signal $\boldsymbol{x}$ from a number of measurements which is instead determined by the number of intrinsic degrees of freedom. For example, suppose that $\boldsymbol{x} = \sum_{i=1}^{k} \alpha_i \boldsymbol{d}_i$ is a superposition of $k < n$ elements from $\mathcal{D}$, and that we observe $\boldsymbol{y} = \mathcal{A}[\boldsymbol{x}]$, where $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^m$ is a linear map. How can we use the knowledge that $\boldsymbol{x}$ is simple to recover it?

Recall that to recover a sparse vector, we minimize the $\ell^1$ norm of the coefficients $\alpha_i$ in an expression $\boldsymbol{x} = \sum_i \alpha_i \boldsymbol{d}_i$ of $\boldsymbol{x}$ with respect to $\mathcal{D}_{\text{sparse}}$. To recover a low-rank matrix, we minimize the nuclear norm of $\boldsymbol{X}$ – also the sum of the coefficients $\alpha_i$ in an expression $\boldsymbol{X} = \sum_i \alpha_i \boldsymbol{d}_i$ with respect to $\mathcal{D}_{\text{low-rank}}$. In both cases, *to recover a signal which consists of a superposition of a few elements from an atomic set, we minimize the sum of the coefficients in an expression of $\boldsymbol{x}$ as a superposition of elements from that set.* This principle immediately generalizes to other atomic sets. To this end, we define a function $\|\cdot\|_{\mathcal{D}}$ called the *atomic gauge*, which measures the minimum of the sum of the coefficients $\alpha_i$, over all ways of expressing $\boldsymbol{x}$ as a superposition of elements from $\mathcal{D}$:

Definition 6.1 (Atomic Gauge). *The* atomic gauge *associated with the set $\mathcal{D}$*

*is the function*

$$\|\boldsymbol{x}\|_{\mathcal{D}} \doteq \inf\left\{\sum_{i=1}^{k}\alpha_i \;\middle|\; \alpha_1,\ldots,\alpha_k \geq 0 \text{ and } \exists\, \boldsymbol{d}_1,\ldots,\boldsymbol{d}_k \in \mathcal{D} \text{ s.t. } \sum_i \alpha_i \boldsymbol{d}_i = \boldsymbol{x}\right\}. \tag{6.1.16}$$

The notion of atomic gauge is general enough to include all of the convex relaxations that we have studied so far:

Example 6.2 (Examples of Atomic Gauges). *The following are examples of atomic gauges:*

- **Sparse vectors:** $\|\boldsymbol{x}\|_{\mathcal{D}_{\text{sparse}}} = \|\boldsymbol{x}\|_1$.
- **Low-rank matrices:** $\|\boldsymbol{X}\|_{\mathcal{D}_{\text{low rank}}} = \|\boldsymbol{X}\|_*$.
- **Column sparse matrices:** $\|\boldsymbol{X}\|_{\mathcal{D}_{\text{column sparse}}} = \sum_i \|\boldsymbol{x}_i\|_2$.

From these examples, we can see that the atomic gauge is often actually a norm. In fact, this is true whenever the atomic set $\mathcal{D}$ is symmetric:

Lemma 6.3 (Atomic Gauges and Norms). *For any set $\mathcal{D}$, $\|\cdot\|_{\mathcal{D}}$ is a convex function. Moreover, if $\mathcal{D}$ is a symmetric set whose convex hull contains an open ball about $\mathbf{0}$, i.e., $\boldsymbol{d} \in \mathcal{D} \implies -\boldsymbol{d} \in \mathcal{D}$, and $\mathbf{0} \in \operatorname{int}(\operatorname{conv}[\mathcal{D}])$[1] then $\|\cdot\|_{\mathcal{D}}$ is a norm.*

*Proof* Convexity follows from the definition: Consider any $\boldsymbol{x}$, $\boldsymbol{x}'$, and any $\lambda \in [0,1]$. For any $\varepsilon > 0$, let

$$\boldsymbol{x} = \sum_{i=1}^{r}\alpha_i \boldsymbol{d}_i, \qquad \boldsymbol{x}' = \sum_{i=1}^{r'}\alpha_i' \boldsymbol{d}_i' \tag{6.1.17}$$

be such that

$$\sum_{i=1}^{r}\alpha_i \leq \|\boldsymbol{x}\|_{\mathcal{D}} + \varepsilon, \quad \text{and} \quad \sum_{i=1}^{r'}\alpha_i' \leq \|\boldsymbol{x}'\|_{\mathcal{D}} + \varepsilon. \tag{6.1.18}$$

Then noting that

$$\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{x}' = \sum_{i=1}^{r}\lambda\alpha_i \boldsymbol{d}_i + \sum_{i=1}^{r'}(1-\lambda)\alpha_i' \boldsymbol{d}_i', \tag{6.1.19}$$

we have that

$$\|\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{x}'\|_{\mathcal{D}} \leq \sum_{i=1}^{r}\lambda\alpha_i + \sum_{j=1}^{r'}(1-\lambda)\alpha_i' \tag{6.1.20}$$

$$\leq \lambda\|\boldsymbol{x}\|_{\mathcal{D}} + (1-\lambda)\|\boldsymbol{x}'\|_{\mathcal{D}} + \varepsilon. \tag{6.1.21}$$

Since $\varepsilon > 0$ can be made arbitrarily small,

$$\|\lambda\boldsymbol{x} + (1-\lambda)\boldsymbol{x}'\|_{\mathcal{D}} \leq \lambda\|\boldsymbol{x}\|_{\mathcal{D}} + (1-\lambda)\|\boldsymbol{x}'\|_{\mathcal{D}}. \tag{6.1.22}$$

[1] Here $\operatorname{conv}[\mathcal{D}]$ is the convex hull spanned by $\mathcal{D}$, and $\operatorname{int}(\cdot)$ is the (open) interior of a set.

It is similarly immediate from the definition that $\|\boldsymbol{x}\|_{\mathcal{D}}$ is positively homogeneous: for $\alpha > 0$

$$\|\alpha \boldsymbol{x}\|_{\mathcal{D}} = \alpha \|\boldsymbol{x}\|_{\mathcal{D}} . \tag{6.1.23}$$

Symmetry of $\mathcal{D}$ implies that $\|-\boldsymbol{x}\|_{\mathcal{D}} = \|\boldsymbol{x}\|_{\mathcal{D}}$; combining with positive homogenity, we obtain that for every $\alpha \in \mathbb{R}$

$$\|\alpha \boldsymbol{x}\|_{\mathcal{D}} = |\alpha| \|\boldsymbol{x}\|_{\mathcal{D}} . \tag{6.1.24}$$

Finally, if $\operatorname{conv}(\mathcal{D})$ contains an open ball about $\mathbf{0}$, i.e., $\exists \varepsilon > 0$ such that $\mathsf{B}(\mathbf{0}, \varepsilon) \subseteq \operatorname{conv}(\mathcal{D})$, then $\|\boldsymbol{x}\|_{\mathcal{D}}$ is finite-valued for every $\boldsymbol{x}$, i.e., $\|\boldsymbol{x}\|_{\mathcal{D}} \leq \|\boldsymbol{x}\|_{\ell^2} / \varepsilon$. This, together with the previous considerations implies that $\|\cdot\|_{\mathcal{D}}$ is a norm. □

The atomic gauge allows us to define a general class of convex problems for recovering a structured signal $\boldsymbol{x}_o$ from underdetermined and/or noisy observations. For example, for recovering $\boldsymbol{x}_o$ from noise-free measurements $\boldsymbol{y} = \mathcal{A}[\boldsymbol{x}_o]$, we can try to minimize the atomic norm $\|\boldsymbol{x}\|_{\mathcal{D}}$ of $\boldsymbol{x}$ subject to the measurement constraint:

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_{\mathcal{D}} \quad \text{subject to} \quad \mathcal{A}[\boldsymbol{x}] = \boldsymbol{y}. \tag{6.1.25}$$

In the presence of noise, we can instead solve an optimization problem which balances between fidelity to the observed data and model simplicity, measured by the atomic gauge:

$$\min_{\boldsymbol{x}} \tfrac{1}{2} \|\mathcal{A}[\boldsymbol{x}] - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{x}\|_{\mathcal{D}} . \tag{6.1.26}$$

This is a convex optimization problem, which generalizes the Lasso problem studied in Chapter 3, as well as nuclear norm minimization problems studied in section Chapter 4 for low-rank recovery.

For some choices of $\mathcal{D}$, these problems admit very efficient algorithms – important examples include $\mathcal{D}_{\text{sparse}}$, $\mathcal{D}_{\text{low-rank}}$, $\mathcal{D}_{\text{column sparse}}$, and $\mathcal{D}_{\text{sinusoids}}$. For other choices of $\mathcal{D}$, they may be intractable – examples include $\mathcal{D}_{\text{low rank tensor}}$ and $\mathcal{D}_{\text{sparse and low rank}}$. The key property that distinguishes the examples for which the convex problems (6.1.25)-(6.1.26) are tractable is whether the simpler problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_{\mathcal{D}} + \tfrac{1}{2} \|\boldsymbol{x} - \boldsymbol{z}\|_2^2 \tag{6.1.27}$$

admits an efficient solution. This simpler problem, called the *proximal problem* associated with the gauge $\|\cdot\|_{\mathcal{D}}$, will form the basis for efficient and scalable algorithms, which we will study in more depth in Chapter 8.

*Other Approaches to Structured Sparsity.*
The atomic norm is based on a *synthesis* model, in which the target signal $\boldsymbol{x}$ is constructed as a sparse superposition of atoms. A dual approach to deriving optimization problems for recovering structured sparse signals is based on *analysis* models, which ask certain projections of the signal $\boldsymbol{x}$ to be zero. We illustrate

this approach through the notion of *group sparsity* for vectors in $\mathbb{R}^n$. Given a collection of supports $\mathcal{G} \subseteq 2^{[n]}$ of the indices $\{1, \ldots, n\}$, we can write

$$\|\boldsymbol{x}\|_{\mathcal{G}} = \sum_{\mathsf{I} \in \mathcal{G}} \|\boldsymbol{x}_{\mathsf{I}}\|_2. \tag{6.1.28}$$

As long as $\cup_{I \in \mathcal{G}} I = \{1, \ldots, n\}$, this is a norm. Minimizing (6.1.28) encourages as many of the $\boldsymbol{x}_{\mathsf{I}}$ to be zero as possible.

How does this construction relate to the atomic norm model described above? *When the groups* $\mathsf{I} \in \mathcal{G}$ *do not overlap*, they are equivalent. Writing

$$\mathcal{D}_{\text{group}} \equiv \{\boldsymbol{x}_{\mathsf{I}} \mid \mathsf{I} \in \mathcal{G}, \|\boldsymbol{x}_{\mathsf{I}}\|_2 = 1\}, \tag{6.1.29}$$

we have

$$\|\boldsymbol{x}\|_{\mathcal{D}_{\text{group}}} = \|\boldsymbol{x}\|_{\mathcal{G}}. \tag{6.1.30}$$

However, when the groups $\mathsf{I} \in \mathcal{G}$ do overlap, the atomic norm and the group sparsity norm (6.1.28) differ, and optimizing them produces subtly different effects. For concreteness, consider $\boldsymbol{x} \in \mathbb{R}^3$, let us consider two different groups of supports:

$$\mathcal{G}_1 = \{\{1,2\},\{3\}\}, \quad \mathcal{G}_2 = \{\{1,2,3\},\{1,2\},\{1\},\{2\},\{3\}\}. \tag{6.1.31}$$

Notice that supports in $\mathcal{G}_1$ do not overlap but those in $\mathcal{G}_2$ do. These groups give two corresponding group sparsity norms:

$$\|\boldsymbol{x}\|_{\mathcal{G}_1} = \|\boldsymbol{x}_{\{1,2\}}\|_2 + |\boldsymbol{x}_3|, \quad \|\boldsymbol{x}\|_{\mathcal{G}_2} = \|\boldsymbol{x}_{\{1,2,3\}}\|_2 + \|\boldsymbol{x}_{\{1,2\}}\|_2 + |\boldsymbol{x}_1| + |\boldsymbol{x}_2| + |\boldsymbol{x}_3|.$$

Figure 6.2 shows the norm ball defined by the norms associated with these groups. In the latter situation, the group sparse norm differs from the atomic norm: minimizing the atomic norm encourages the signal to be expressible as just a few atoms, whereas minimizing the group sparse norm encourages many of the $\boldsymbol{x}_{\mathsf{I}}$ to be zero. There is a vast literature that studies group sparsity inducing norms for structured signals. The manuscript of Bach et. al. [BJMO12] gives a systematic introduction to these norms and their associated optimization algorithms.

## 6.2 Geometry, Measure Concentration, and Phase Transition

In Chapter 3 and Chapter 4, we have characterized conditions for $\mathcal{D}_{\text{sparse}}$ and $\mathcal{D}_{\text{low-rank}}$ under which the program (6.1.25) can recover the correct solution $\boldsymbol{x}_o$. We would like to know for a more general atomic set $\mathcal{D}$ whether the program (6.1.25) also succeeds under broad conditions. Furthermore, as we have alluded earlier in these chapters, there seems to be a sharp *phase transition* between the success and failure of the program (6.1.25). This section provides a rigorous explication of the phase transition phenomenon in a general setting, using tools from high-dimensional statistics and geometry of convex cones.

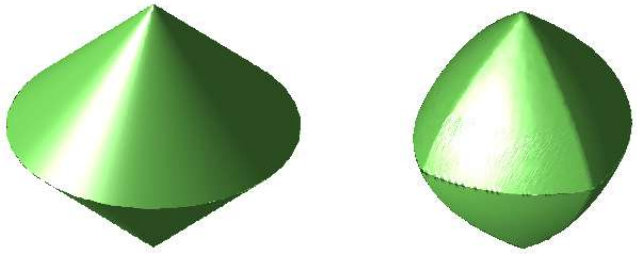


**Figure 6.2** **Left** is for a non-overlapping group sparsity norm-1 ball in 3-dimensional space: $\|\boldsymbol{x}\|_{\mathcal{G}_1}$. **Right** is for a structured sparsity norm-1 ball with overlapping subsets in 3-dimensional space: $\|\boldsymbol{x}\|_{\mathcal{G}_2}$. Singular points appearing on these balls characterize the sparsity-inducing behavior of the associated norms.

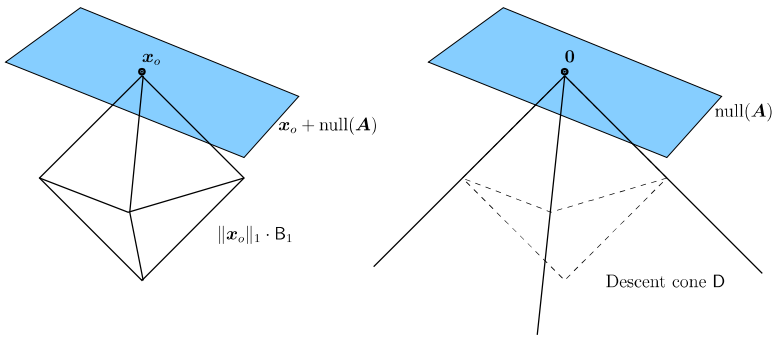



**Figure 6.3** **Cones and the Coefficient Space Geometry.** $\ell^1$ minimization uniquely recovers $\boldsymbol{x}_o$ if and only if the intersection of the descent cone D with null($\boldsymbol{A}$) is $\{\boldsymbol{0}\}$.

### 6.2.1 Success Condition as Two Non-Intersecting Cones

*Geometry of $\ell^1$ Norm Minimization.*
Let us first draw inspiration from the familiar case of $\ell^1$ norm to make general conclusions about atomic minimization. Suppose that $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o$ for a $k$-sparse vector $\boldsymbol{x}_o$. Recall the geometric picture of the $\ell^1$ ball in Figure 6.3 (left), which we introduced in Section 3.1 and in Section 3.6.2. There, we argued that $\boldsymbol{x}_o$ is the unique optimal solution to the $\ell^1$ minimization problem:

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}. \tag{6.2.1}$$

if and only if the affine subspace $\boldsymbol{x}_o + \text{null}(\boldsymbol{A})$ of feasible solutions $\boldsymbol{x}$ intersects the scaled $\ell^1$ ball

$$\mathsf{B}_1 = \{\boldsymbol{x} \mid \|\boldsymbol{x}\|_1 \leq \|\boldsymbol{x}_o\|_1\} \tag{6.2.2}$$

only at $\boldsymbol{x}_o$, as illustrated in Figure 6.3 (left).

We can express the same geometry more cleanly in terms of the *descent cone*:

$$\mathsf{D} \doteq \{\boldsymbol{v} \mid \|\boldsymbol{x}_o + t\boldsymbol{v}\|_1 \leq \|\boldsymbol{x}_o\|_1 \text{ for some } t > 0\}. \tag{6.2.3}$$

This is the set of directions $\boldsymbol{v}$ for which a small (but nonzero) perturbation of $\boldsymbol{x}_o$ in the $\boldsymbol{v}$ direction does not increase the objective function $\|\cdot\|_1$. The descent cone $\mathsf{D}$ is visualized in Figure 6.3 (right).

Notice that the perturbation $\boldsymbol{x}_o + t\boldsymbol{v}$ is a feasible solution for $t \neq 0$ if and only if $\boldsymbol{v} \in \text{null}(\boldsymbol{A})$. The feasible perturbations which do not increase the objective function reside in the intersection $\mathsf{D} \cap \text{null}(\boldsymbol{A})$. Because $\mathsf{D}$ is a convex cone and $\text{null}(\boldsymbol{A})$ is a subspace (a special convex cone), $\mathsf{D}$ and $\text{null}(\boldsymbol{A})$ always intersect at $\boldsymbol{0}$. It is not difficult to see that $\boldsymbol{x}_o$ is the unique optimal solution to the $\ell^1$ problem if and only if $\boldsymbol{0}$ is the only point of intersection between $\text{null}(\boldsymbol{A})$ and $\mathsf{D}$. This was proved in Lemma 3.34.

Hence, to study whether $\ell^1$ minimization succeeds, we may equivalently check that the subspace $\text{null}(\boldsymbol{A})$ does not have nontrivial intersection with the cone $\mathsf{D}$. Because $\boldsymbol{A}$ is a random matrix, $\text{null}(\boldsymbol{A})$ is a random subspace, of dimension $n - m$. If $\boldsymbol{A}$ is Gaussian, then $\text{null}(\boldsymbol{A})$ follows the uniform distribution on the set of subspaces $\mathsf{S} \subset \mathbb{R}^n$ of dimension $n - m$.[2] Clearly, the probability that the random subspace $\text{null}(\boldsymbol{A})$ intersects the descent cone $\mathsf{D}$ depends on properties of $\mathsf{D}$. Intuitively, we would expect intersections to be more likely if $\mathsf{D}$ is "big" in some sense.

REMARK 6.4. *Notice that the probability of success mentioned above is for a given* fixed $\boldsymbol{x}_o$ *with respect to a randomly chosen* $\boldsymbol{A}$*. As we have discussed in Section 3.6.1, this is a weaker notion of success guarantee than the case with incoherence and RIP that we studied in Chapter 3 which states that for a* fixed *matrix* $\boldsymbol{A}$*, the* $\ell^1$ *minimization* (6.2.1) *succeeds for* all *sufficiently sparse* $\boldsymbol{x}_o$ *with high probability.*

### *The General Case with the Atomic Norm*

For a general atomic norm $\|\cdot\|_{\mathcal{D}}$, the condition for the program (6.1.25) to succeed is very similar to the program (6.2.1) for the $\ell^1$ norm. We only need to replace the descent cone of $\ell^1$ norm with the descent cone associated with the atomic norm:

$$\mathsf{C} \doteq \{\boldsymbol{v} \mid \|\boldsymbol{x}_o + t\boldsymbol{v}\|_{\mathcal{D}} \leq \|\boldsymbol{x}_o\|_{\mathcal{D}} \text{ for some } t > 0\}, \tag{6.2.4}$$

and replace the null space of $\boldsymbol{A}$ with the null space of $\mathcal{A}$:

$$\mathsf{S} \doteq \text{null}(\mathcal{A}).$$

Then similar to Lemma (3.34), we have:

PROPOSITION 6.5. *Suppose that* $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}_o)$*. Then* $\boldsymbol{x}_o$ *is the unique optimal solution to the atomic norm minimization problem if and only if* $\mathsf{C} \cap \mathsf{S} = \{\boldsymbol{0}\}$*.*

[2] To be more precise, $\text{null}(\boldsymbol{A})$ is distributed according to the Haar measure on the Grassmannian $\mathsf{G}_{n,m-n}$.

For a given atomic norm, the descent cone $\mathsf{C}$ is fixed. The measurement operator $\mathcal{A}$ is typically a random operator. Its null space $\mathsf{S} = \mathrm{null}(\mathcal{A})$ is a random subspace. Hence, to characterize the probability of success of the program (6.1.25), the problem reduces to charactering the probability of a random linear subspace $\mathsf{S}$ intersecting a given convex cone $\mathsf{C}$.

### 6.2.2 Intrinsic Volumes and Kinematic Formula

How can we calculate the probability of one random linear subspace $\mathsf{S}$ intersecting a convex cone $\mathsf{C}$? More over, what does the probability depend on? To get intuition for what to expect in the general case, let us start with the simplest case when the convex cone $\mathsf{C}$ itself is a linear subspace $\mathsf{S}'$.

*Example: Two Intersecting Subspaces*
When does a randomly chosen subspace $\mathsf{S}$ intersect another subspace $\mathsf{S}'$? From elementary geometry, we know that if the sum of the dimensions $\dim(\mathsf{S})+\dim(\mathsf{S}')$ is greater than the ambient dimension $n$, then $\mathsf{S}$ and $\mathsf{S}'$ necessarily have a nontrivial intersection. Conversely, if $\dim(\mathsf{S}) + \dim(\mathsf{S}') \leq n$, the probability that $\mathsf{S}$ intersects $\mathsf{S}'$ nontrivially is zero:

Proposition 6.6 (Intersection of Two Linear Subspace). *Let $\mathsf{S}'$ be any linear subspace of $\mathbb{R}^n$, and let $\mathsf{S}$ be a uniform random subspace. Then*

$$\mathbb{P}\left[\mathsf{S} \cap \mathsf{S}' = \{\mathbf{0}\}\right] = 0, \quad \dim(\mathsf{S}) + \dim(\mathsf{S}') > n; \tag{6.2.5}$$

$$\mathbb{P}\left[\mathsf{S} \cap \mathsf{S}' = \{\mathbf{0}\}\right] = 1, \quad \dim(\mathsf{S}) + \dim(\mathsf{S}') \leq n. \tag{6.2.6}$$

Figure 6.4 illustrates two examples on how two subspaces in $\mathbb{R}^3$ intersect in general. From the example of two intersecting subspaces, we see that the probability of whether or not they intersect only at the origin $\mathbf{0}$ depends only on the sum of their dimensions.

*Intrinsic Volumes.*
In our case, however, we are dealing with the intersection of a linear subspace and a convex cone. Or in more general cases that we will see later, we need to study the intersection of two convex cones.[3] Hence, it is natural to ask whether the notion of "dimension" for subspaces can be generalized to convex cones? If so, we may expect to characterize the probability for two convex cones to intersect in a similar way as Proposition 6.6 for linear subspaces. We next develop a more generalized way to measure the "dimension" or "size" of a given convex cone.

[3] For instance, for the problem of decomposing sparse and low-rank matrices, we need to study the intersection of the descent cone of $\ell^1$ norm and that of the nuclear norm.

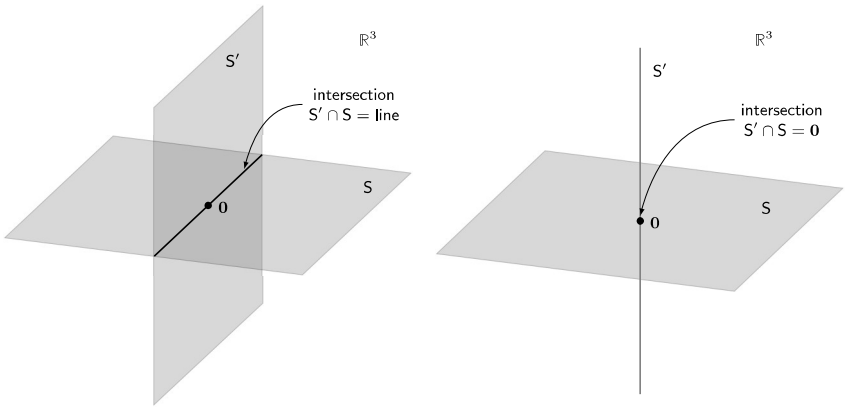



**Figure 6.4** **Left:** intersection of two generic 2D planes in $\mathbb{R}^3$ contains a line; **Right:** intersection of a 2D plane and a 1D line, in general position, is only the origin $\mathbf{0}$.

In mathematics, such topics are studied in the field of conic integral geometry [SW08, Ame11].[4]

EXAMPLE 6.7 (Equivalent Definitions of Dimension for Subspaces). *Again, let us first draw some ideas from the special case of a linear subspace. Notice that the dimension, say $d$, of a linear subspace $\mathsf{S}$ can also be equivalently computed as the average (squared) length of a random (Gaussian) vector, say $\boldsymbol{g} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, projected onto the subspace:*

$$d = \dim(\mathsf{S}) \;=\; \mathbb{E}_{\boldsymbol{g}} \left[ \|\mathcal{P}_{\mathsf{S}}[\boldsymbol{g}]\|_2^2 \right], \tag{6.2.7}$$

*where $\mathcal{P}_{\mathsf{S}}[\boldsymbol{g}]$ is the unique nearest vector to $\boldsymbol{g}$ in $\mathsf{S}$:*

$$\mathcal{P}_{\mathsf{S}}[\boldsymbol{g}] \;\doteq\; \arg\min_{\boldsymbol{x} \in \mathsf{S}} \|\boldsymbol{x} - \boldsymbol{g}\|_2^2 . \tag{6.2.8}$$

*We may also take the random vector $\boldsymbol{g}$ as uniformly distributed on the unit sphere $\mathbb{S}^{n-1}$. In this case we have:*

$$d = \dim(\mathsf{S}) \;=\; n \cdot \mathbb{E}_{\boldsymbol{g}} \left[ \|\mathcal{P}_{\mathsf{S}}[\boldsymbol{g}]\|_2^2 \right], \quad \boldsymbol{g} \sim \text{uniform}(\mathbb{S}^{n-1}). \tag{6.2.9}$$

*We leave this as an exercise to the reader.*

As it turns out, projecting a (random) vector is precisely the right way to measure the "size" of a convex cone. Like a subspace, for a closed convex cone $\mathsf{C} \subseteq \mathbb{R}^n$ and a vector $\boldsymbol{z}$, there is a unique nearest vector to $\boldsymbol{z}$ in $\mathsf{C}$, denoted $\mathcal{P}_{\mathsf{C}}[\boldsymbol{z}]$:

$$\mathcal{P}_{\mathsf{C}}[\boldsymbol{z}] \;\doteq\; \arg\min_{\boldsymbol{x} \in \mathsf{C}} \|\boldsymbol{x} - \boldsymbol{z}\|_2^2 . \tag{6.2.10}$$

Figure 6.5 shows the projections $\mathcal{P}_{\mathsf{C}_i}[\boldsymbol{z}]$ of a vector $\boldsymbol{z}$ onto two convex cones $\mathsf{C}_1$

[4] For a more thorough survey of the history of spherical or conic integral geometry, one may refer to [ALMT13].

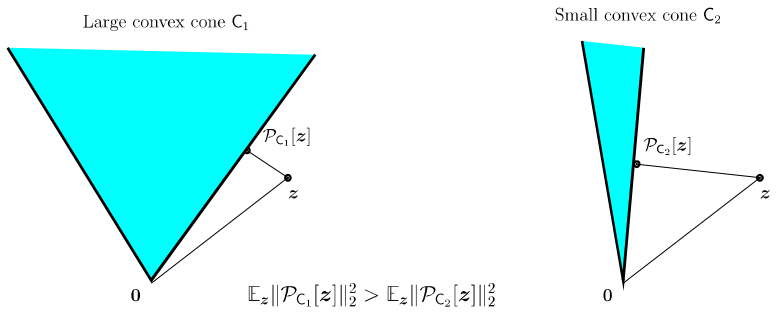



**Figure 6.5 Projections onto a Closed Convex Cone.** For a closed convex cone $\mathsf{C}$, $\mathcal{P}_{\mathsf{C}}[\boldsymbol{z}]$ is the nearest point to $\boldsymbol{z}$ in $\mathsf{C}$. Notice that in this case, the projection of $\boldsymbol{z}$ onto the larger cone $\mathsf{C}_1$ has greater norm than the projection of $\boldsymbol{z}$ onto the smaller cone $\mathsf{C}_2$: $\|\mathcal{P}_{\mathsf{C}_1}[\boldsymbol{z}]\|_2^2 > \|\mathcal{P}_{\mathsf{C}_2}[\boldsymbol{z}]\|_2^2$. We can measure the "size" of a convex cone $\mathsf{C}$ by averaging $\|\mathcal{P}_{\mathsf{C}}[\boldsymbol{z}]\|_2^2$ over all directions $\boldsymbol{z}$; this average is known as the *statistical dimension* of the cone, denoted $\delta(\mathsf{C})$.

and $\mathsf{C}_2$. Notice that it is always true that

$$\|\mathcal{P}_{\mathsf{C}}[\boldsymbol{z}]\|_2 \leq \|\boldsymbol{z}\|_2 . \tag{6.2.11}$$

Moreover, in Figure 6.5, the norm of the projection is larger for the wider $\mathsf{C}_i$. Thus, we could take $\|\mathcal{P}_{\mathsf{C}}[\boldsymbol{z}]\|_2^2$ as an indication of the "size" of $\mathsf{C}$.

However, unlike a linear subspace, a convex cone, like the descent cone of the $\ell^1$ norm, may consist of many faces of different dimensions. In particular, the descent cone of the $\ell^1$ norm is a special case of an important family of convex cones known as polyhedral cones. Each polyhedral cone is the intersection of a finite number of half spaces. Given a polyhedral cone in $\mathbb{R}^n$, in theory, it could have faces in dimension $k = 0, 1, \ldots, n$. We may consider the projection of a standard normal random vector $\boldsymbol{g}$ onto faces of a particular dimension $k$.

Definition 6.8 (Intrinsic Volume). *If $\mathsf{C}$ is a polyhedral cone in $\mathbb{R}^n$, then the $k$th intrinsic volume $v_k(\mathsf{C})$ is defined to be:*

$$v_k(\mathsf{C}) \doteq \mathbb{P}\left[\mathcal{P}_{\mathsf{C}}[\boldsymbol{g}] \in \text{a } k\text{-dim face of } \mathsf{C}\right], \quad k = 0, 1, \ldots, n, \tag{6.2.12}$$

*where $\boldsymbol{g} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.*

According to the definition, the intrinsic volumes are actually a probability distribution on $\{0, 1, \ldots, n\}$. Hence we have $v_k(\mathsf{C}) \geq 0$ for all $k = 0, 1, \ldots, n$ and

$$\sum_{k=0}^{n} v_k(\mathsf{C}) = 1. \tag{6.2.13}$$

The intrinsic volumes have many interesting properties that have been systematically developed in conic integral geometry.

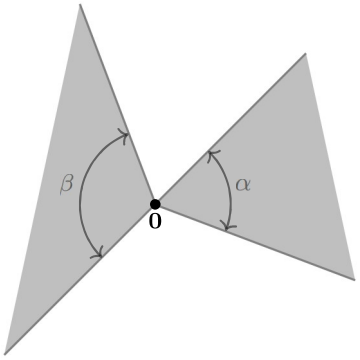



**Figure 6.6** The probability of two planar cones intersecting is the sum of their angles (as fraction of $2\pi$).

EXAMPLE 6.9 (Intrinsic Volumes of a Linear Subspace). *If* $\mathsf{C}$ *is* $d$ *dimensional linear subspace* $\mathsf{S}$*, then we have*

$$v_k(\mathsf{S}) = \begin{cases} 1 & d = k, \\ 0 & \textit{else.} \end{cases}$$

*We leave this as an exercise to the reader.*

EXAMPLE 6.10 (Intrinsic Volumes of a Cone in $\mathbb{R}^2$). *Consider a convex cone* $\mathsf{C}$ *in* $\mathbb{R}^2$ *similar to the ones illustrated in Figure 6.5. Denote the angle of the cone as* $\alpha$*. Then it is easy to show that*

$$v_2(\mathsf{C}) = \alpha/2\pi, \quad v_1(\mathsf{C}) = 1/2, \quad \textit{and} \quad v_0(\mathsf{C}) = (\pi - \alpha)/2\pi.$$

*We leave the proof as an exercise to the reader.*

*Conic Kinematic Formula.*
As usual, let us start with a simple example.

EXAMPLE 6.11 (Two Cones in $\mathbb{R}^2$). *Notice that if we have two convex cones* $\mathsf{C}_1$ *and* $\mathsf{C}_2$ *in* $\mathbb{R}^2$*, with angle* $\alpha$ *and* $\beta$ *respectively. Let* $\mathsf{C}_1$ *be fixed and we rotate* $\mathsf{C}_2$ *by a rotation* $\boldsymbol{R}$ *uniformly chosen from* $\mathbb{S}^1$*. Then the two cones* $\mathsf{C}_1$ *and* $\boldsymbol{R}(\mathsf{C}_2)$ *will always have non-trivial overlap (besides at the origin* $\mathbf{0}$*) if and only if* $\alpha+\beta > 2\pi$*. If* $\alpha + \beta \leq 2\pi$*, the probability that they have non-trivial intersection is precisely* $(\alpha + \beta)/2\pi = v_2(\mathsf{C}_1) + v_2(\mathsf{C}_2)$*, as shown in Figure 6.6. Or equivalently, we have*

$$\mathbb{P}[\mathsf{C}_1 \cap \boldsymbol{R}(\mathsf{C}_2) \neq \{\mathbf{0}\}] = \min\left\{1, v_2(\mathsf{C}_1) + v_2(\mathsf{C}_2)\right\}. \tag{6.2.14}$$

*We leave the verification as an exercise to the reader.*

The above example suggests that the probability that two convex cones intersect non-trivially depends on their intrinsic volumes. However, for convex cones in a high-dimensional space, the situation can be much more complicated than in

the 2D space. Surprisingly, as one of the main result in conic integral geometry, the probability of two convex cones intersecting can be precisely characterized in terms of their intrinsic volumes. This is known as the *kinematic formula*.

PROPOSITION 6.12 (The Kinematic Formula for Two Convex Cones). *Consider two convex (polyhedral) cones* $\mathsf{C}_1$ *and* $\mathsf{C}_2$ *in* $\mathbb{R}^n$. *Let* $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ *be a random matrix uniformly distributed in the orthogonal group* $\mathsf{O}(n,\mathbb{R})$. *Then we have*

$$\mathbb{P}[\mathsf{C}_1 \cap \boldsymbol{A}(\mathsf{C}_2) \neq \{\mathbf{0}\}] = \sum_{i=0}^{n}(1+(-1)^{i+1})\sum_{j=i}^{n} v_i(\mathsf{C}_1)v_{d+i-j}(\mathsf{C}_2), \tag{6.2.15}$$

*where* $\boldsymbol{A}(\mathsf{C}_2)$ *is a cone obtained by applying the random orthogonal matrix* $\boldsymbol{A}$ *to all of the elements of* $\mathsf{C}_2$.

One may check that equation (6.2.14) for two convex cones in $\mathbb{R}^2$ is a special case of this formula. Interested readers may refer to [SW08] for a proof of this formula.

Despite its rigor and elegance, the kinematic formula is challenging to directly use, since the intrinsic volumes $v_k(\mathsf{C})$ are typically not computable except for very simple cones. For the descent cones of most atomic norms, explicit expressions for their intrinsic volumes are not known (and also difficult to compute numerically). Without such expressions, how can we assess the probability $\mathbb{P}[\mathsf{C}_1 \cap \boldsymbol{A}(\mathsf{C}_2) \neq \{\mathbf{0}\}]$? This is where measure concentration in high-dimensional spaces comes to help: one can use the fact that $\mathcal{P}_{\mathsf{C}}[\boldsymbol{g}]$ is a function of many independent random variables to argue that the intrinsic volumes concentrate, giving simple, but accurate bounds on the probability of intersection (6.2.15).

### 6.2.3 Statistical Dimension and Phase Transition

As we have seen earlier in the case of a subspace (6.2.7), averaging the projection of a random vector $\boldsymbol{g}$ onto the subspace gives an equivalent way of measuring the dimension of the subspace. This concept has led to the notion of the intrinsic volumes for a convex cone, which give the probability $v_k$ that the random vector is projected onto the interior of a $k$-dimensional face. It is then natural to wonder if the average of the projection over the entire cone or all faces gives an equivalent measure of "dimension" of the cone. This leads to the notion of *statistical dimension* of a convex cone.

*Statistical Dimension and Approximate Kinematic Formula*

DEFINITION 6.13 (Statistical Dimension). *Given* $\mathsf{C}$ *is a closed convex cone in* $\mathbb{R}^n$, *then its statistical dimension, denoted as* $\delta(\mathsf{C})$, *is given by:*

$$\delta(\mathsf{C}) \doteq \mathbb{E}_{\boldsymbol{g}}\left[\|\mathcal{P}_{\mathsf{C}}[\boldsymbol{g}]\|_2^2\right], \tag{6.2.16}$$

*where* $\boldsymbol{g} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.

To see the connection with the intrinsic volumes defined above, we may consider computing the overall expectation through conditional expectation of $\boldsymbol{g}$ projected on $k$-dimensional faces, denoted as $\mathsf{S}_k$. We know from (6.2.7) that this expectation is exactly the dimension of the subspace $k$. Therefore, conceptually, we have

$$\mathbb{E}_{\boldsymbol{g}}\left[\|\mathcal{P}_{\mathsf{C}}[\boldsymbol{g}]\|_2^2\right] = \sum_{k=0}^{n} k \cdot v_k(\mathsf{C}). \tag{6.2.17}$$

The right hand side is often taken as an alternative definition of the statistical dimension of a convex cone.[5]

To great extent, the statistical dimension is the natural generalization of the notion of "dimension" of subspaces to convex cones. It is easy to show that it has the following nice properties:

1 For a linear subspace $\mathsf{S}$, we have

$$\delta(\mathsf{S}) = \dim(\mathsf{S}).$$

2 It is invariant under orthogonal transformation:

$$\delta(\mathsf{C}) = \delta(\boldsymbol{A}(\mathsf{C}))$$

for all orthogonal matrix $\boldsymbol{A}$ in the orthogonal group $\mathsf{O}(n, \mathbb{R})$.

3 The sum of the statistical dimension of a cone $\mathsf{C}$ and that of its orthogonal complement, also known as the polar cone $\mathsf{C}^o$,[6] equals the dimension of the ambient space:

$$\delta(\mathsf{C}) + \delta(\mathsf{C}^o) = n.$$

4 For the direct product of two closed convex cones $\mathsf{C}_1$ and $\mathsf{C}_2$, we have:

$$\delta(\mathsf{C}_1 \times \mathsf{C}_2) = \delta(\mathsf{C}_1) + \delta(\mathsf{C}_2).$$

We leave the proof of these properties to the reader as exercises, as well as a few other useful properties and facts.

*Phase Transition of Atomic Norm Minimization.*
As we have seen in Proposition 6.6 for linear subspaces, the sum of the statistical dimensions precisely controls whether two subspaces $\mathsf{S}$ and $\mathsf{S}'$ have nontrivial intersection: once $\delta(\mathsf{S}) + \delta(\mathsf{S}') > n$, the probability of nontrivial intersection goes from zero to one. For general convex cones, there is a similar phenomenon: if $\mathsf{S}$ is a random subspace of $\mathbb{R}^n$, and $\mathsf{C}$ a closed convex cone, then we have:

$$\begin{aligned} \delta(\mathsf{S}) + \delta(\mathsf{C}) \gg n \implies \mathsf{S} \cap \mathsf{C} \neq \{\mathbf{0}\} \text{ with high probability;} \\ \delta(\mathsf{S}) + \delta(\mathsf{C}) \ll n \implies \mathsf{S} \cap \mathsf{C} = \{\mathbf{0}\} \text{ with high probability.} \end{aligned}$$

The following theorem makes this precise:

[5] A formal proof can be obtained using the *spherical Steiner formula* [SW08]. Interested reader may refer to [ALMT14] for a detailed derivation.

[6] The polar cone $\mathsf{C}^o$ is defined to be: $\mathsf{C}^o = \{\boldsymbol{y} \in \mathbb{R}^n : \langle \boldsymbol{y}, \boldsymbol{x} \rangle \leq 0, \forall \boldsymbol{x} \in \mathsf{C}\}$.

THEOREM 6.14. *Let* $\mathsf{C}$ *denote any closed convex cone in* $\mathbb{R}^n$*, and let* $\mathsf{S}$ *be a uniformly distributed random subspace of dimension* $\delta(\mathsf{S})$*. Then*

$$\mathbb{P}\left[\mathsf{S} \cap \mathsf{C} = \{\mathbf{0}\}\right] \leq C \exp\Big( - c\frac{\big(n - \delta(\mathsf{S}) - \delta(\mathsf{C})\big)^2}{n}\Big), \;\; \delta(\mathsf{S}) + \delta(\mathsf{C}) \geq n;$$
$$\mathbb{P}\left[\mathsf{S} \cap \mathsf{C} = \{\mathbf{0}\}\right] \geq 1 - C \exp\Big( - c\frac{\big(\delta(\mathsf{S}) + \delta(\mathsf{C}) - n\big)^2}{n}\Big), \;\; \delta(\mathsf{S}) + \delta(\mathsf{C}) \leq n$$

*for some constant* $C, c > 0$.

The above equations are also known as the *approximate kinematic formula* which captures the essential behavior of the kinematic formula (6.2.15) in a high-dimensional space due to measure concentration. This theorem is a special case of a somewhat more general result controlling the probability that two randomly oriented convex cones intersect (that we will elaborate later). The proof relies on technical results in spherical integral geometry. We refer the interested reader to Theorem 1 of [ALMT14], its proof, and references therein.

Theorem 6.14 then implies our main claim about the phase transition in atomic norm minimization (6.1.25). In our situation, the cone $\mathsf{C}$ of interest is the descent cone $\mathsf{D}$ of the atomic norm $\|\cdot\|_{\mathcal{D}}$ at $\boldsymbol{x}_o$. We wish to know whether $\mathsf{S} = \mathrm{null}(\mathcal{A})$ has nontrivial intersection with $\mathsf{C}$. The dimension of $\mathsf{S}$ is $n - m$, and so the above heuristics become

$$\textbf{FAILURE: } \delta(\mathsf{D}) \gg m \implies \mathrm{null}(\mathcal{A}) \cap \mathsf{D} \neq \{\mathbf{0}\} \text{ with high probability;}$$
$$\textbf{SUCCESS: } \delta(\mathsf{D}) \ll m \implies \mathrm{null}(\mathcal{A}) \cap \mathsf{D} = \{\mathbf{0}\} \text{ with high probability.}$$

In the first case, the atomic norm minimization (6.1.25) fails to recover $\boldsymbol{x}_o$; in the second case it succeeds. Using Theorem 6.14 to make this precise, we obtain:

COROLLARY 6.15 (Phase Transition for Atomic Norm Minimization). *Let* $\mathcal{A} \in \mathbb{R}^{m \times n}$ *be (the matrix representation of) a random linear operator, and suppose that* $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}_o)$*. Let* $\mathsf{D}$ *denote the descent cone of the atomic norm* $\|\cdot\|_{\mathcal{D}}$ *at* $\boldsymbol{x}_o$*. Then*

$$\mathbb{P}\left[\,(6.1.25) \text{ uniquely recovers } \boldsymbol{x}_o\,\right] \;\leq\; C \exp\left(-c\frac{(\delta(\mathsf{D}) - m)^2}{n}\right), \;\; m \leq \delta(\mathsf{D});$$
$$\mathbb{P}\left[\,(6.1.25) \text{ uniquely recovers } \boldsymbol{x}_o\,\right] \;\geq\; 1 - C \exp\left(-c\frac{(m - \delta(\mathsf{D}))^2}{n}\right), \;\; m \geq \delta(\mathsf{D}).$$

Thus, when the number of (random) measurements $m$ is substantially smaller than $\delta(\mathsf{D})$, recovery fails with high probability; when $m$ is substantially larger than $\delta(\mathsf{D})$ recovery succeeds with high probability. To great extent, the above theorem explains the phase transition phenomena, around $\delta(\mathsf{D})$, that we have observed in Chapter 3 for sparse vector recovery and in Chapter 4 for low-rank matrix recovery.

### 6.2.4 Statistical Dimension of Descent Cone of the $\ell^1$ Norm

According to the above corollary, the success of the atomic norm minimization (6.1.25) depends on whether the number of independent measurements exceeds the statistical dimension $\delta(\mathsf{D})$ of the descent cone of the atomic norm. Hence, it is extremely important to be able to accurately estimate $\delta(\mathsf{D})$. In this section, we give a detailed derivation of the statistical dimension of the descent cones of the $\ell^1$ norm. One may derive in a similar way an expression for the descent cone of the nuclear norm, which we state (without proof) in Theorem 4.23 in Chapter 4. Interested readers may find details for the nuclear norm in [ALMT14].

In Chapter 3, we have given an expression for the phase transition of $\ell^1$ norm minimization in Theorem 3.35. We here give a detailed calculation and show that the statistical dimension $\delta(\mathsf{D})$ of the descent cone $\mathsf{D}$ is very close to $n\psi(k/n)$, where the function $\psi(\cdot)$ is defined in (3.6.6). We state this result as a lemma below:

LEMMA 6.16. *Let* $\mathsf{D}$ *be the descent cone of the* $\ell^1$ *norm at any* $\boldsymbol{x}_o \in \mathbb{R}^n$ *satisfying* $\|\boldsymbol{x}_o\|_0 = k$. *Then*

$$n\psi\left(\frac{k}{n}\right) - 4\sqrt{n/k} \le \delta(\mathsf{D}) \le n\psi\left(\frac{k}{n}\right). \tag{6.2.18}$$

*Proof* For this, we will need two basic facts about projections onto convex cones. The first is the generalized pythagorean formula, which implies that for a closed convex cone $\mathsf{D}$ with polar cone

$$\mathsf{D}^\circ = \{\boldsymbol{v} \mid \langle \boldsymbol{v}, \boldsymbol{x}\rangle \le 0 \ \forall\, \boldsymbol{x} \in \mathsf{D}\}, \tag{6.2.19}$$

for any $\boldsymbol{z} \in \mathbb{R}^n$,

$$\|\mathcal{P}_{\mathsf{D}}\boldsymbol{z}\|_2^2 = \|\boldsymbol{z} - \mathcal{P}_{\mathsf{D}^\circ}\boldsymbol{z}\|_2^2 = \mathrm{dist}^2(\boldsymbol{z}, \mathsf{D}^\circ). \tag{6.2.20}$$

This allows us to replace the norm of the projection of $\boldsymbol{z}$ onto $\mathsf{D}$ with the distance of $\boldsymbol{z}$ to the polar cone $\mathsf{D}^\circ$. The second fact is that the polar of the descent cone is the conic hull of the subdifferential

$$\mathsf{S} \doteq \partial \|\cdot\|_1 (\boldsymbol{x}_o) = \{\boldsymbol{v} \mid \boldsymbol{v}_{\mathsf{I}} = \mathrm{sign}(\boldsymbol{x}_{o\mathsf{I}}),\ \|\boldsymbol{v}_{\mathsf{I}^c}\|_\infty \le 1\}. \tag{6.2.21}$$

Namely,

$$\begin{aligned} \mathsf{D}^\circ = \mathrm{cone}(\mathsf{S}) \ &= \bigcup_{t\ge 0} t\,\mathsf{S} \\ &= \{t\boldsymbol{v} \mid t \ge 0,\ \boldsymbol{v}_{\mathsf{I}} = \boldsymbol{\sigma}_{\mathsf{I}},\ \|\boldsymbol{v}_{\mathsf{I}^c}\|_\infty \le 1\}, \end{aligned} \tag{6.2.22}$$

where $\boldsymbol{\sigma}_{\mathsf{I}}$ is a shorthand for $\mathrm{sign}(\boldsymbol{x}_{o\mathsf{I}})$. For any vector $\boldsymbol{z}$, the nearest vector $\hat{\boldsymbol{z}} \in t\mathsf{S}$ satisfies

$$\hat{\boldsymbol{z}}_i = \begin{cases} t\,\mathrm{sign}(\boldsymbol{z}_i) & i \in \mathsf{I}, \\ \boldsymbol{z}_i & i \in \mathsf{I}^c,\ |\boldsymbol{z}_i| \le t, \\ t\,\mathrm{sign}(\boldsymbol{z}_i) & i \in \mathsf{I}^c,\ |\boldsymbol{z}_i| > t, \end{cases} \tag{6.2.23}$$

and the distance is given by

$$\begin{aligned}\operatorname{dist}^2(\boldsymbol{z}, t\mathsf{S}) &= \|\boldsymbol{z} - \hat{\boldsymbol{z}}\|_2^2 \\ &= \|\boldsymbol{z}_{\mathsf{I}} - t\boldsymbol{\sigma}_{\mathsf{I}}\|_2^2 + \sum_{j \in \mathsf{I}^c} \max\{|z_j| - t, 0\}^2 . \end{aligned} \tag{6.2.24}$$

Hence, for any vector $\boldsymbol{z}$,

$$\begin{aligned}\operatorname{dist}^2(\boldsymbol{z}, \mathsf{D}^\circ) &= \min_{t \geq 0} \operatorname{dist}^2(\boldsymbol{z}, t\mathsf{S}) \\ &= \min_{t \geq 0} \left\{ \|\boldsymbol{z}_{\mathsf{I}} - t\boldsymbol{\sigma}_{\mathsf{I}}\|_2^2 + \sum_{j \in \mathsf{I}^c} \max\{|z_j| - t, 0\}^2 \right\}. \end{aligned} \tag{6.2.25}$$

Using these facts, we calculate

$$\begin{aligned}\delta(\mathsf{D}) &= \mathbb{E}_{\boldsymbol{g} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[ \|\mathcal{P}_{\mathsf{D}} \boldsymbol{g}\|_2^2 \right] \\ &= \mathbb{E}_{\boldsymbol{g} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[ \operatorname{dist}^2(\boldsymbol{g}, \mathsf{D}^\circ) \right] \\ &= \mathbb{E}_{\boldsymbol{g}} \left[ \min_{t \geq 0} \operatorname{dist}^2 (\boldsymbol{g}, t\,\mathsf{S}) \right] \\ &\leq \min_{t \geq 0} \ \mathbb{E}_{\boldsymbol{g}} \left[ \operatorname{dist}^2 (\boldsymbol{g}, t\,\mathsf{S}) \right] \\ &= \min_{t \geq 0} \ \mathbb{E}_{\boldsymbol{g}} \left[ \|\boldsymbol{g}_{\mathsf{I}} - t\boldsymbol{\sigma}_{\mathsf{I}}\|_2^2 + \sum_{j \in \mathsf{I}^c} \max\{|g_j| - t, 0\}^2 \right] \\ &= \min_{t \geq 0} \left\{ |\mathsf{I}|(1 + t^2) + 2|\mathsf{I}^c| \int_{s=t}^{\infty} (s - t)^2 \varphi(s) ds \right\} \\ &= n\,\psi(k/n), \end{aligned} \tag{6.2.26}$$

where $\varphi(s) = \frac{1}{\sqrt{2\pi}} e^{-s^2/2}$ is the Gaussian density and $\psi(\cdot)$ is defined in (3.6.6). Thus, we have established $n\psi(k/n)$ as an upper bound on the statistical dimension; and hence $m^\star = n\psi(k/n)$ as a lower bound on the phase transition.

To finish, we show that this upper bound on $\delta(\mathsf{D})$ is tight, by establishing a (nearly) matching lower bound. Let $\hat{t}$ minimize $\mathbb{E}_{\boldsymbol{g}} \left[ \operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S}) \right]$. Then

$$0 = \frac{d}{dt} \mathbb{E}_{\boldsymbol{g}} \left[ \operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S}) \right] \Big|_{t = \hat{t}} \ = \ \mathbb{E}_{\boldsymbol{g}} \left[ \frac{d}{dt} \operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S}) \Big|_{t = \hat{t}} \right]. \tag{6.2.27}$$

Let $t_{\boldsymbol{g}}$ minimize $\operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S})$ with respect to $t$. By convexity of this function in $t$,

$$\operatorname{dist}^2(\boldsymbol{g}, t_{\boldsymbol{g}}\mathsf{S}) \geq \operatorname{dist}^2(\boldsymbol{g}, \hat{t}\mathsf{S}) + \left( t_{\boldsymbol{g}} - \hat{t} \right) \frac{d}{dt} \operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S}) \big|_{t = \hat{t}}. \tag{6.2.28}$$

Notice that by (6.2.27),

$$0 = \hat{t}\, \mathbb{E}_{\boldsymbol{g}} \left[ \frac{d}{dt} \operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S}) \Big|_{t = \hat{t}} \right] = \mathbb{E}\left[ t_{\boldsymbol{g}} \right] \mathbb{E}_{\boldsymbol{g}} \left[ \frac{d}{dt} \operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S}) \Big|_{t = \hat{t}} \right], \tag{6.2.29}$$

and so

$$\begin{aligned}\mathbb{E}_{\boldsymbol{g}}\left[\min_t \operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S})\right] &= \mathbb{E}_{\boldsymbol{g}}\left[\operatorname{dist}^2(\boldsymbol{g}, t_{\boldsymbol{g}}\mathsf{S})\right] \\ &\geq \mathbb{E}_{\boldsymbol{g}}\left[\operatorname{dist}^2(\boldsymbol{g}, \hat{t}\mathsf{S})\right] + \mathbb{E}_{\boldsymbol{g}}\left[(t_{\boldsymbol{g}} - \mathbb{E}_{\boldsymbol{g}}[t_{\boldsymbol{g}}])\,\frac{d}{dt}\operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S})\Big|_{t=\hat{t}}\right], \\ &\geq \mathbb{E}_{\boldsymbol{g}}\left[\operatorname{dist}^2(\boldsymbol{g}, \hat{t}\mathsf{S})\right] - \operatorname{var}(t_{\boldsymbol{g}})^{1/2}\operatorname{var}\left(\frac{d}{dt}\operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S})\Big|_{t=\hat{t}}\right)^{1/2}. \end{aligned} \tag{6.2.30}$$

In the last line we have used the Cauchy-Schwarz inequality for random variables.

To conclude, we bound the variance of the two terms. For $t_{\boldsymbol{g}}$, let $\boldsymbol{v}_{\boldsymbol{g}} \in \mathsf{S}$ be such that $t_{\boldsymbol{g}}\boldsymbol{v}_{\boldsymbol{g}}$ is the nearest element to $\boldsymbol{g}$ in $\mathsf{D}^\circ$. Notice that

$$\|\boldsymbol{g} - \boldsymbol{g}'\|_2 \geq \|t_{\boldsymbol{g}}\boldsymbol{v}_{\boldsymbol{g}} - t_{\boldsymbol{g}'}\boldsymbol{v}_{\boldsymbol{g}'}\|_2 \geq \|t_{\boldsymbol{g}}\boldsymbol{\sigma}_{\mathsf{I}} - t_{\boldsymbol{g}'}\boldsymbol{\sigma}_{\mathsf{I}}\|_2 = |t_{\boldsymbol{g}} - t_{\boldsymbol{g}'}|\sqrt{k}, \tag{6.2.31}$$

whence $t_{\boldsymbol{g}}$ is a $1/\sqrt{k}$-Lipschitz function of $\boldsymbol{g}$. By the Gaussian Poincare inequality,[7] its variance is bounded as $\operatorname{var}(t_{\boldsymbol{g}}) \leq 1/k$.

Meanwhile, by Danskin's theorem,

$$\frac{d}{dt}\operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S}) = \frac{d}{dt}\|\boldsymbol{g} - t\boldsymbol{v}_g\|_2^2 \;=\; 2\boldsymbol{v}_{\boldsymbol{g}}^*\left(t\boldsymbol{v}_{\boldsymbol{g}} - \boldsymbol{g}\right). \tag{6.2.32}$$

Note that because $t\boldsymbol{v}_{\boldsymbol{g}}$ is the projection of $\boldsymbol{g}$ onto the convex set $\mathsf{D}^\circ$, for any other $\boldsymbol{v} \in \mathsf{S}$,

$$(t\boldsymbol{v}_{\boldsymbol{g}} - t\boldsymbol{v})^*(t\boldsymbol{v}_{\boldsymbol{g}} - \boldsymbol{g}) \leq 0, \tag{6.2.33}$$

whence

$$\boldsymbol{v}_{\boldsymbol{g}}^*(t\boldsymbol{v}_{\boldsymbol{g}} - \boldsymbol{g}) \leq \boldsymbol{v}_{\boldsymbol{g}'}^*(t\boldsymbol{v}_{\boldsymbol{g}} - \boldsymbol{g}), \tag{6.2.34}$$

and

$$\begin{aligned}&\frac{d}{dt}\operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S}) - \frac{d}{dt}\operatorname{dist}^2(\boldsymbol{g}', t\mathsf{S}) \;=\; 2\boldsymbol{v}_{\boldsymbol{g}}^*(t\boldsymbol{v}_{\boldsymbol{g}} - \boldsymbol{g}) - 2\boldsymbol{v}_{\boldsymbol{g}'}^*(t\boldsymbol{v}_{\boldsymbol{g}'} - \boldsymbol{g}') \\ &\leq 2\boldsymbol{v}_{\boldsymbol{g}'}^*(t\boldsymbol{v}_{\boldsymbol{g}} - \boldsymbol{g}) - 2\boldsymbol{v}_{\boldsymbol{g}'}^*(t\boldsymbol{v}_{\boldsymbol{g}'} - \boldsymbol{g}') \\ &\leq 2\|\boldsymbol{v}_{\boldsymbol{g}'}\|_2\left(\|t\boldsymbol{v}_{\boldsymbol{g}} - t\boldsymbol{v}_{\boldsymbol{g}'}\|_2 + \|\boldsymbol{g} - \boldsymbol{g}'\|_2\right) \\ &\leq 4\|\boldsymbol{v}_{\boldsymbol{g}'}\|_2\|\boldsymbol{g} - \boldsymbol{g}'\|_2 \\ &\leq 4\sqrt{n}\,\|\boldsymbol{g} - \boldsymbol{g}'\|_2. \end{aligned} \tag{6.2.35}$$

By the same reasoning,

$$\begin{aligned}\frac{d}{dt}\operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S}) - \frac{d}{dt}\operatorname{dist}^2(\boldsymbol{g}', t\mathsf{S}) &\geq 2\boldsymbol{v}_{\boldsymbol{g}}^*\left(t\boldsymbol{v}_g - \boldsymbol{g} - t\boldsymbol{v}_{\boldsymbol{g}'} + \boldsymbol{g}'\right) \\ &\geq -4\sqrt{n}\,\|\boldsymbol{g} - \boldsymbol{g}'\|_2, \end{aligned} \tag{6.2.36}$$

whence

$$\left|\frac{d}{dt}\operatorname{dist}^2(\boldsymbol{g}, t\mathsf{S}) - \frac{d}{dt}\operatorname{dist}^2(\boldsymbol{g}', t\mathsf{S})\right| \;\leq\; 4\sqrt{n}\,\|\boldsymbol{g} - \boldsymbol{g}'\|_2, \tag{6.2.37}$$

[7] which states that if $f$ is an $L$-Lipschitz function and $\boldsymbol{g}$ a Gaussian vector, then $\operatorname{var}(f(\boldsymbol{g})) \leq L^2$.

and $\frac{d}{dt}\text{dist}^2(\boldsymbol{g}, t\mathsf{S})$ is $4\sqrt{n}$-Lipschitz. By the Gaussian Poincare inequality,

$$\text{var}\left(\frac{d}{dt}\text{dist}^2(\boldsymbol{g}, t\mathsf{S})\Big|_{t=\hat{t}}\right) \;\leq\; 4\sqrt{n}, \tag{6.2.38}$$

and so

$$\mathbb{E}_{\boldsymbol{g}}\left[\min_t \text{dist}^2(\boldsymbol{g}, t\mathsf{S})\right] \;\geq\; \min_t \mathbb{E}_{\boldsymbol{g}}\left[\text{dist}^2(\boldsymbol{g}, t\mathsf{S})\right] - 4\sqrt{n/k}. \tag{6.2.39}$$

Thus,

$$n\psi(k/n) - 4\sqrt{n/k} \;\leq\; \delta(\mathsf{D}) \;\leq\; n\psi(k/n). \tag{6.2.40}$$

Combining this bound with the above results proves that the phase transition occurs within $O(\sqrt{n})$ of $m^\star = n\psi(k/n)$. □

### 6.2.5 Phase Transition in Decomposing Structured Signals

*Examples of Decomposing Structured Signals.*
In the robust face recognition problem that we have seen in Chapter 2 and later studied in Chapter 13, we want to solve a problem of recovering a sparse $\boldsymbol{x}_o$ and a sparse error $\boldsymbol{e}_o$ from the mixed measurements:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o + \boldsymbol{e}_o, \tag{6.2.41}$$

where $\boldsymbol{A}$ is a known matrix, drawn from certain random distribution. This problem can be viewed as a special case of the so-called *morphological component analysis* [SDC03, SED05, ESQD05].

In the robust principal component analysis (RPCA) problem that we have studied in Chapter 5, we want to recover a low-rank matrix $\boldsymbol{L}_o$ and a sparse matrix $\boldsymbol{S}_o$ from their sum:

$$\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{S}_o. \tag{6.2.42}$$

Or in the compressive principal component pursuit, we want to recover the low-rank and sparse matrices from a random projection of their sum:

$$\boldsymbol{Y} \;\doteq\; \mathcal{P}_{\mathsf{Q}}[\boldsymbol{L}_o + \boldsymbol{S}_o], \tag{6.2.43}$$

where $\mathsf{Q} \subseteq \mathbb{R}^{n_1 \times n_2}$ is a random linear subspace and $\mathcal{P}_{\mathsf{Q}}$ denotes the projection operator onto that subspace.

*Incoherence through Randomness.*
As we have seen in developing solutions to the above problems, we often require the two mixed structured signals to be "incoherent" to each other. Otherwise the decomposition itself is not well-defined and solutions will not be unique. Hence to understand the underlying geometric reason when such decompositions are possible and the solution is unique, a simple but illuminating model is to assume

when we mix two structured signals, say $\boldsymbol{x}_o$ and $\boldsymbol{s}_o$, together, one signal is in a random position with respect to the other:

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}_o) + \boldsymbol{z}_o, \tag{6.2.44}$$

where $\mathcal{A}$ is a random orthogonal transformation in the space of $\boldsymbol{x}_o$. The random operator $\mathcal{A}$ ensures that $\boldsymbol{x}_o$ is in general position to $\boldsymbol{z}_o$ hence the two components $\mathcal{A}(\boldsymbol{x}_o)$ and $\boldsymbol{z}_o$ are incoherent to each other.

*Decomposition through Atomic Norm Minimization*
Now assume $\boldsymbol{x}_o$ is a low-dimensional structured signal associated with an atomic set $\mathcal{D}_1$, and $\boldsymbol{z}_o$ with $\mathcal{D}_2$. As we have seen in the face recognition and the robust PCA cases, a natural convex program to recover $\boldsymbol{x}_o$ and $\boldsymbol{z}_o$ is

$$\min_{\boldsymbol{x},\boldsymbol{z}} \|\boldsymbol{x}\|_{\mathcal{D}_1} \quad \text{subject to} \quad \|\boldsymbol{z}\|_{\mathcal{D}_2} \le \|\boldsymbol{z}_o\|_{\mathcal{D}_2},\ \boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{z}, \tag{6.2.45}$$

where $\|\cdot\|_{\mathcal{D}_1}$ and $\|\cdot\|_{\mathcal{D}_2}$ are the atomic norms associated with $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively.[8]

Now let $\mathsf{C}_1(\boldsymbol{x}_o)$ be the descent cone of the atomic norm $\|\cdot\|_{\mathcal{D}_1}$ at $\boldsymbol{x}_o$ and $\mathsf{C}_2(\boldsymbol{z}_o)$ the cone for $\|\cdot\|_{\mathcal{D}_2}$ at $\boldsymbol{z}_o$. Suppose $(\boldsymbol{x}_o, \boldsymbol{z}_o)$ is not the (unique) optimal solution to the above program and

$$(\boldsymbol{x}_o + \Delta\boldsymbol{x}, \boldsymbol{z}_o + \Delta\boldsymbol{z})$$

is an optimal solution. Then we must have $\Delta\boldsymbol{x}$ is in the descent cone $\mathsf{C}_1(\boldsymbol{x}_o)$ and $\Delta\boldsymbol{z}$ is in the descent cone $\mathsf{C}_2(\boldsymbol{z}_o)$. Furthermore, from the constraint $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{z}$ we have

$$-\mathcal{A}(\Delta\boldsymbol{x}) = \Delta\boldsymbol{z}.$$

In other words, $\Delta\boldsymbol{z}$ must be in the intersection of the cone $\mathsf{C}_2(\boldsymbol{z}_o)$ and $-\mathcal{A}(\mathsf{C}_1(\boldsymbol{x}_o))$:

$$\mathbf{0} \neq \Delta\boldsymbol{z} \in \mathsf{C}_2(\boldsymbol{z}_o) \cap -\mathcal{A}(\mathsf{C}_1(\boldsymbol{x}_o)),$$

as illustrated in Figure 6.7 on the right. For $(\boldsymbol{x}_o, \boldsymbol{z}_o)$ to be the only optimal solution to the program (6.2.45), we must have the intersection of the two cones to be trivial – only contains the origin $\mathbf{0}$, as illustrated in Figure 6.7 on the left.

*Phase Transition for Decomposition.*
As we have alluded to earlier, in a high-dimensional space $\mathbb{R}^n$, we anticipate the probability of two cones intersecting transitions sharply around

$$\delta(\mathsf{C}_1(\boldsymbol{x}_o)) + \delta(\mathsf{C}_2(\boldsymbol{z}_o)) = n.$$

[8] The optimization problem (6.2.45) is equivalent to the problem

$$\min_{\boldsymbol{x},\boldsymbol{z}} \|\boldsymbol{x}\|_{\mathcal{D}_1} + \lambda\|\boldsymbol{z}\|_{\mathcal{D}_2} \quad \text{subject to} \quad \boldsymbol{y} = \mathcal{A}(\boldsymbol{x}) + \boldsymbol{z},$$

under an appropriate (instance specific) choice of $\lambda > 0$. This form may be more familiar from our discussion of face recognition and robust PCA. In this section, we study the constrained form (6.2.45), which is slightly more convenient for geometric analysis.

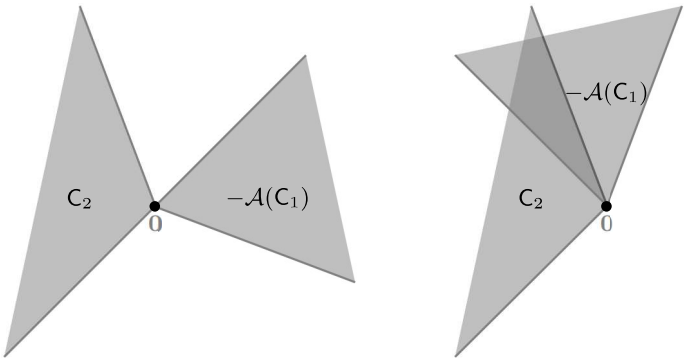



**Figure 6.7** The success of the program (6.2.45) depends on if the random cone $-\mathcal{A}(\mathsf{C}_1)$ intersects with the fixed cone $\mathsf{C}_2$. **Left:** if the intersection is trivial, then the decomposition problem succeeds; **Right:** if the intersection is not trivial, then the decomposition problem fails.

In other words,

$$\delta(\mathsf{C}_1) + \delta(\mathsf{C}_2) \gg n \implies \mathsf{C}_1 \cap \mathsf{C}_2 \neq \{\mathbf{0}\} \text{ with high probability;}$$
$$\delta(\mathsf{C}_1) + \delta(\mathsf{C}_2) \ll n \implies \mathsf{C}_1 \cap \mathsf{C}_2 = \{\mathbf{0}\} \text{ with high probability.}$$

The following theorem makes this precise:

THEOREM 6.17. *Let $\mathsf{C}_1$ and $\mathsf{C}_2$ be two closed convex cones in $\mathbb{R}^n$, and let $\mathcal{A}$ be a random orthogonal matrix uniformly distributed in the orthogonal group. Then*

$$\mathbb{P}\left[-\mathcal{A}(\mathsf{C}_1) \cap \mathsf{C}_2 = \{\mathbf{0}\}\right] \leq C \exp\left(-c\frac{\left(n - \delta(\mathsf{C}_1) - \delta(\mathsf{C}_2)\right)^2}{n}\right), \quad \delta(\mathsf{C}_1) + \delta(\mathsf{C}_2) \geq n;$$
$$\mathbb{P}\left[-\mathcal{A}(\mathsf{C}_1) \cap \mathsf{C}_2 = \{\mathbf{0}\}\right] \geq 1 - C \exp\left(-c\frac{\left(\delta(\mathsf{C}_1) + \delta(\mathsf{C}_2) - n\right)^2}{n}\right), \quad \delta(\mathsf{C}_1) + \delta(\mathsf{C}_2) \leq n,$$

*for some constant $C, c > 0$.*

The above bounds can be considered an *approximate kinematic formula* which captures the essential behavior of the kinematic formula (6.2.15) in a high-dimensional space due to measure concentration [ALMT14]. This is a more general statement than Theorem 6.14 where one of the two cones is a subspace.

## 6.3 Limitations of Convex Relaxation

Our story up to this point has been one of success. The development up to this point has demonstrated general ways of constructing regularizers that encode various structural assumptions about the signals we are interested in computing with. For sparse vectors, low-rank matrices, and several other structures discussed in Section 6.1, these regularizers have turned out to be computationally tractable, and to yield statistical performance which is nearly the best possible

under their assumptions. In a sense, it is surprising that we do not have to pay a stronger statistical price for effective and efficient algorithms. Nevertheless, one should not expect convex relaxation to work equally effectively for *all* challenging problems. Below we discuss a few scenarios in which convex relaxation becomes limited or even may fail to work.

### 6.3.1 Suboptimality of Convex Relaxation for Multiple Structures

In Section 6.1.1 we have discussed that in some problems such as sparse PCA, we would like to recover a matrix $\boldsymbol{X}$ that is simultaneously sparse and low-rank. In fact, such problems arise naturally in practical applications such as structured texture inpainting or repairing that we will study in great detail in Chapter 15, see Section 15.3. The images of regular patterns shown in Figure 15.4, if viewed as matrices, are both low-rank and sparse in the Fourier or wavelet domain.

A sparse and low-rank matrix is a special case of a signal that has multiple structures. It seems that one natural convex relaxation to promote multiple structures is to use a weighted sum of their corresponding atomic norms. For instance, we may minimize

$$\lambda_1 \|\boldsymbol{X}\|_1 + \lambda_2 \|\boldsymbol{X}\|_* \tag{6.3.1}$$

to promote the recovered matrix to be *both* sparse and low-rank.[9] This is exactly what we will be practicing in Chapter 15 for regular texture repairing, and indeed, empirically, the combined regularization does work better than using only one.

However, the above combined convex regularization is not optimal in terms of statistical efficiency. To see this, let us consider the simple problem of estimating a sparse and low-rank matrix $\boldsymbol{X}_o \in \mathbb{R}^{n\times n}$ from noisy measurements:

$$\boldsymbol{Y} = \boldsymbol{X}_o + \boldsymbol{Z} \ \in \mathbb{R}^{n\times n}, \tag{6.3.2}$$

where $\boldsymbol{Z} \in \mathbb{R}^{n\times n}$ is matrix whose entries are i.i.d Gaussian noise. Hence using the above combined regularization, one may use the following convex program to estimate $\boldsymbol{X}_o$:

$$\hat{\boldsymbol{X}}(\boldsymbol{Y}) = \arg\min_{\boldsymbol{X}} \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{X}\|_F^2 + \lambda_1 \|\boldsymbol{X}\|_1 + \lambda_2 \|\boldsymbol{X}\|_*. \tag{6.3.3}$$

To evaluate the goodness of the estimate $\hat{\boldsymbol{X}}(\boldsymbol{Y})$, we measure the mean square error (MSE) with respect to the ground truth:

$$\text{MSE} \doteq \mathbb{E}[\|\hat{\boldsymbol{X}}(\boldsymbol{Y}) - \boldsymbol{X}_o\|_F^2]. \tag{6.3.4}$$

Suppose the ground truth $\boldsymbol{X}_o$ is a $k \times k$ sparse matrix and of rank less than $r$.

[9] Asking $\boldsymbol{X}$ to be *simultaneously* low-rank and sparse is quite different from asking it to be *decomposable* as a sum of low-rank and sparse, $\boldsymbol{X} = \boldsymbol{L} + \boldsymbol{S}$. The latter problem, studied in Chapter 5 does admit convex relaxations, which succeed when $\boldsymbol{L}$ and $\boldsymbol{S}$ are sufficiently structured and incoherent.

It has been shown in [OJFH13] that the above convex program (6.3.3) leads to a mean square error bounded from below as:

$$\text{MSE} \geq c \cdot \min\{k^2, n\}$$

for some $c > 0$. Nevertheless, as shown in [OJFH13], it is actually relatively easy to solve a *non-convex* program to obtain an estimate with much lower MSE:

$$\text{MSE} \leq C \cdot k$$

for some $C > 0$. Unlike the case with a single low-dimensional structure, the convex relaxation gives an estimate that is suboptimal in terms of statistical accuracy. This suboptimality can also be felt in the number of (noiseless) random measurements required to reconstruct $\boldsymbol{X}_o$: minimizing any combination of the $\ell^1$ norm and nuclear norm requires at least $c\min\{k^2, n\text{rank}(\boldsymbol{X}_o)\}$ measurements, even $\boldsymbol{X}_o$ has only $O(k\text{rank}(\boldsymbol{X}_o))$ degrees of freedom [OJF+15]. This can be explained in terms of the statistical dimension of the descent cone associated to a combined convex regularization such as (6.3.1), as we will describe in the next section.

### 6.3.2 Intractable Convex Relaxation for High-Order Tensors

Section 6.1 also gave the first hint that a tight correspondence between the statistical and computational limits might not obtain for certain types of low-dimensional structures. For example, for recovering a high-order low-rank tensor $\boldsymbol{\mathcal{X}}_o$ of the form (6.1.11), the atomic norm associated with the set (6.1.12) (as a natural generalization of the nuclear norm) has excellent statistical performance, but its computationally intractable.

In practice, people often seek computationally tractable alternative to approximately promote low-rank for high-order tensors. One popular choice is to convert a high-order tensor to matrix forms and consider the so-called Tucker rank [Tuc66,KB09]. Given a $K$-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times \cdots \times n_K}$, for each of its mode $i = 1, \ldots, K$, we construct the matrix $\boldsymbol{\mathcal{X}}_{(i)} \in \mathbb{R}^{n_i \times \prod_{j \neq i} n_j}$ by concatenating all the mode-$i$ fibers of $\boldsymbol{\mathcal{X}}$ as columns of $\boldsymbol{\mathcal{X}}_{(i)}$. Then the so-called Tucker rank is defined as:

$$\text{rank}_{tc}(\boldsymbol{\mathcal{X}}) \doteq \left(\text{rank}\left(\boldsymbol{\mathcal{X}}_{(1)}\right), \text{rank}\left(\boldsymbol{\mathcal{X}}_{(2)}\right), \ldots, \text{rank}\left(\boldsymbol{\mathcal{X}}_{(K)}\right)\right). \tag{6.3.5}$$

Hence, to recover a tensor $\boldsymbol{\mathcal{X}}_o$ of low (Tucker) rank, say from random measurements $\boldsymbol{\mathcal{Y}} = \mathcal{A}(\boldsymbol{\mathcal{X}}_o)$, we may impose that the ranks of all $K$ unfolded matrices $\boldsymbol{\mathcal{X}}_{(i)}$ to be low. A natural convex regularization is to minimize a weighted sum of nuclear norms of all the $K$ matrices:

$$\min_{\boldsymbol{\mathcal{X}}} \sum_{i=1}^{K} \lambda_i \|\boldsymbol{\mathcal{X}}_{(i)}\|_* \quad \text{subject to} \quad \boldsymbol{\mathcal{Y}} = \mathcal{A}(\boldsymbol{\mathcal{X}}), \tag{6.3.6}$$

where $\lambda_i \geq 0$ are chosen weights. Notice that this convex regularization is of the same nature as that (6.3.1) for a sparse and low-rank matrix. Each term

$\lambda_i \|\boldsymbol{\mathcal{X}}_{(i)}\|_*$ imposes some additional structure on the same high-order tensor $\boldsymbol{\mathcal{X}}$. In practice, however, one may choose to use any subset of the $K$ matrices, as we will see an example with camera calibration from multiple images in Chapter 15.

We may understand the role of composing multiple norms from the perspective of statistical dimension. That is, we want to know by superposing multiple norms, how the statistical dimension of the descent cone of the composite norm changes. To this end, let us consider the general problem of recovering a high-dimensional signal $\boldsymbol{x}_o \in \mathbb{R}^n$ that has $K$ low-dimensional structures simultaneously. Let $\|\cdot\|_{(i)}$ be the (atomic) norm associated with the $i$-th structure, $i = 1, \ldots, K$. Then, given random measurements $\boldsymbol{y} = \mathcal{A}(\boldsymbol{x}_o)$, we may try to recover $\boldsymbol{x}_o$ by minimizing the composite norm:

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_{com} \doteq \sum_{i=1}^{K} \lambda_i \|\boldsymbol{x}\|_{(i)} \quad \text{subject to} \quad \boldsymbol{y} = \mathcal{A}(\boldsymbol{x}). \tag{6.3.7}$$

Analysis of [MHWG13] has shown that the statistical dimension of the descent cone of the composite norm $\|\cdot\|_{com}$ is actually dominated by the largest among all cones for the norms $\lambda_i\|\cdot\|_{(i)}$. So adding more penalty terms gives diminishing return in terms of improving statistical efficiency. In particular, [MHWG13] has shown that using the composite nuclear norm in (6.3.6) to solve for a (Tucker) rank-$r$ tensor uniquely, the number of measurements needed is essentially $O(rn^{K-1})$; with better arrangement of the unfolded matrices, one can reduce the number of measurements to $O(r^{K/2}n^{K/2})$, whereas a certain nonconvex (potentially intractable) formulation needs only $O(r^K + nrK)$ measurements. There are good reasons to believe, in order to bridge the gap, we may have to deal with the nonconvex nature of high-order tensor estimation directly.

### 6.3.3 Lack of Convex Relaxation for Bilinear Problems

So far, we have mainly considered the problem of recovering a low-dimensional signal $\boldsymbol{x}_o$ from a set of (random or incoherent) measurements $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o$ where the measurement operator/matrix $\boldsymbol{A}$ is known. However, in may practical applications, we do not know the matrix $\boldsymbol{A}$.

For instance, consider $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is some (invertible) transformation on some sparse signals, and we have observed many samples of such signals

$$\boldsymbol{y}_i = \boldsymbol{A}\boldsymbol{x}_i \ \in \mathbb{R}^n, \quad i = 1, 2, \ldots, m.$$

If we do not know the transformation $\boldsymbol{A}$ in advance, we want to recover the transformation so that $\boldsymbol{x}_i = \boldsymbol{A}^{-1}\boldsymbol{y}_i$ will be maximally sparse. In other words, if we stack $\boldsymbol{y}_i$ as columns of a matrix $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m] \in \mathbb{R}^{n\times m}$ and similarly $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m] \in \mathbb{R}^{n\times m}$, we want to decompose $\boldsymbol{Y}$ into

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} \ \in \mathbb{R}^{n\times m},$$

such that $\boldsymbol{X}$ is the sparsest. This is a special matrix factorization problem, also

known as the *dictionary learning* problem, with $\boldsymbol{A}$ being the (complete) sparsifying dictionary to be identified. In applications such as scientific imaging, the matrix $\boldsymbol{A}$ may even have additional structures such as being a convolution. Just like many other structured matrix factorization problems, there is no non-trivial convex relaxation to these nonlinear problems. For these problems, we are often forced to deal with their nonlinear and nonconvex nature directly. Nevertheless, we will see in Section 7.3.2 of Chapter 7, such nonconvex problem has extremely nice structures and properties. Such nice properties make the seemingly challenging nonconvex problem amenable to extremely efficient optimization algorithms (as we will see Section 9.6.2 of Chapter 9).

### 6.3.4 Nonlinear Low-Dimensional Structures

All the low-dimensional models (sparse, low-rank) that we have studied so far assume the low-dimensional structures of the data are piecewise or locally linear – hence they can be represented as linear superposition of a few atoms. As we will see in many of the application chapters, for most real-world data, nonlinearality can easily come from the measurement process or certain nonlinear deformations of the otherwise structured data (say image rectification in Chapter 15). As result, the intrinsic structures of such data are still very low-dimensional, but they are *not necessarily linear*. The support of their distribution may become *nonlinear submanifolds*, instead of linear subspaces! For instance, in speech recognition or object recognition in images, the information we care about is *invariant* to certain group of transformations: shifting, translation, scaling or rotation of the signals (say image rectification in Chapter 15 or classification in Chapter 16). Mathematically speaking, we care about the (low-dimensional) structures of equivalent classes of the signals under such transformations. Such structures are known to be highly nonlinear and complicated [WDCB05].

Hence, to make the fundamental models, concepts and methods developed in this book truly applicable and useful for real-world data and problems, we often need to learn such a nonlinear transform of the data:

$$f(\boldsymbol{x}) : \boldsymbol{x} \mapsto \boldsymbol{z}, \quad f \in \mathcal{F} \tag{6.3.8}$$

in some family of functions $\mathcal{F}$.[10] After the transformation, we expect the intrinsic structures of $\boldsymbol{z} = f(\boldsymbol{x})$ become low-dimensional linear subspaces (as in sparse and low-rank models), which are easier to interpret and use. As we will see in the application chapters, principles and computational tools developed in this book can be readily extended to undo such nonlinear mappings and reveal the low-dimensional structures of real-world data in terms of the canonical (linear) models that we are familiar with.

[10] Typically, we assume $f$ is a smooth or at least continuous mapping, which can be parameterized as polynomials (see Chapter 15) or as deep networks (see Chapter 16).

### 6.3.5 Return of Nonconvex Formulation and Optimization

The above difficulties with convex relaxation have compelled people to reexamine these more challenging problems in their natural nonconvex setting. Somewhat surprisingly, even in the nonconvex setting, low-dimensional structures of the signals have played a crucial role in making such nonconvex problems amenable to efficient and effective solutions. These nonconvex programs are very different from generic nonconvex problems that are known to suffer from local minima and slow convergence. Instead, they have surprisingly good geometric and statistical properties which, if properly leveraged, give rise to simple, efficient algorithms. We will reveal properties of these nonconvex problems in Chapter 7 and develop scalable algorithms to solve them with (optimal) convergence and complexity guarantees in Chapter 9.

To apply fundamental theory and models of this book to real-world problems, in the last Chapter 16, we will touch upon the very important and challenging issue with real-world data: the intrinsic low-dimensional structures of the data can be highly nonlinear and multi-modal. Modern practice of machine learning, especially deep learning, is precisely aiming to learn a nonlinear mapping that leads to certain optimal (linear) representation of the data. We will see how the concepts and principles developed in this book for low-dimensional models play a fundamental role in rigorously interpreting and potentially improving the design of deep networks.

## 6.4 Notes

As mentioned in Chapter 3, the phase transition phenomenon associated with the $\ell^1$ norm minimization was studied in the observation space by Tanner & Donoho from the perspective of random projection of high-dimensional polytopes [Don05, DT09, DT10]. Later studies focused on analyzing in the coefficient space as this approach applies to more general low-dimensional structures [Sto09, OH10, CRPW12, ALMT14]. The upper bound on the statistical dimension of the descent cone of the $\ell^1$ norm is due to Stojnic [Sto09], which derives empirically sharp guarantees for recovery by $\ell^1$ minimization. The proof of the corresponding lower bound follows Amelunxen et. al. [ALMT14], as does our use of the term "statistical dimension" and much of the exposition in this chapter.

The study of low-dimensional structures through convex relaxation has been generalized through the introduction of atomic norm [BTR12] and linear inverse problems via convex optimization [CRPW12]. These earlier works have led to the unified framework based on statistical dimension of the descent cones [ALMT14], presented this chapter. Statistical analysis of the recovery and decomposition problems under noisy measurements has also been systematically developed in a series work from Wainwright and colleagues [Wai09a, ANW12].

Limitations of convex relaxation for certain low-dimensional structures have been revealed through the work of [OJFH13, OJF+15] for sparse low-rank matrices and later [MHWG13] for high-order tensors. In subsequent years, nonconvex formulations have received tremendous attention, as surveyed in the recent papers [SQW15, JK+17, CLC19, Sun19a]. In the next Chapter 7, we will give a more detailed account for the key rationale behind the nonconvex approach, and try to elucidate why and when a nonconvex program is expected to work well. In Chapter 9, we systematically introduce effective and efficient optimization algorithms for solving this class of nonconvex problems in high-dimensional spaces.

## 6.5 Exercises

6.1 (Nonnegative sparse vectors and low-rank matrices). *Consider the* nonnegative *sparse vectors. Identify an atomic set* $\mathcal{D}_{\text{nonnegative sparse}}$ *such that a vector* $\boldsymbol{x}$ *is nonnegative and* $k$*-sparse if and only if it is nonnegative combination of* $k$ *elements of* $\mathcal{D}_{\text{nonnegative sparse}}$.

*Now consider low-rank matrices with nonnegative factors, i.e., matrices that can be expressed as*

$$\boldsymbol{X} = \sum_{i=1}^{r} \boldsymbol{a}_i \boldsymbol{b}_i^*, \tag{6.5.1}$$

*with* $\boldsymbol{a}_i$ *and* $\boldsymbol{b}_i$ *element-wise nonnegative. Identify an atomic set* $\mathcal{D}_{\text{nonnegative low-rank}}$ *such that a matrix* $\boldsymbol{X}$ *is of the form* (6.5.1) *if and only if it can be expressed as a nonnegative linear combination of* $r$ *elements of* $\mathcal{D}_{\text{nonnegative low-rank}}$. *Can you guess which of the atomic norms* $\|\cdot\|_{\mathcal{D}_{\text{nonnegative sparse}}}$ *and* $\|\cdot\|_{\mathcal{D}_{\text{nonnegative low-rank}}}$ *leads to tractable optimization problems?*

6.2 (The $k$-support norm). *Consider the atomic set defined as*

$$\mathcal{D}_k = \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x}\|_0 \le k, \|\boldsymbol{x}\|_2 = 1\}. \tag{6.5.2}$$

*Show that the atomic norm given by the gauge function of this set is the so-called* $k$*-support norm:*

$$\|\boldsymbol{x}\|_k^{sp} = \min\left\{\sum_{\mathsf{I}\in\mathcal{G}_k} \|\boldsymbol{v}_\mathsf{I}\|_2 \;\; s.t. \;\; \sum_{\mathsf{I}\in\mathcal{G}_k} \boldsymbol{v}_\mathsf{I} = \boldsymbol{x}\right\}. \tag{6.5.3}$$

*This gives an alternative convex regularizer for recovering sparse vectors.*

6.3. *Consider an atomic set for* $\mathbb{R}^2$*:*

$$\mathcal{D} = \{\boldsymbol{x}_1 \in \mathbb{S}^1, \boldsymbol{x}_2 = [\pm 1, 0]^*\}. \tag{6.5.4}$$

*What is the associated atomic (gauge) norm* $\|\boldsymbol{x}\|_\mathcal{D}$ *for a* $\boldsymbol{x} \in \mathbb{R}^2$*? From this example, what can you say about a group atomic set* (6.1.29) *that has two supports* $\mathsf{I}' \subset \mathsf{I}$*?*

6.4. *Prove that the definitions of the dimension of a linear subspace are equivalent in Example 6.7.*

6.5. *Compute the intrinsic volumes of a $d$-dimensional subspace in $\mathbb{R}^n$ according to the Definition 6.8 for convex cones.*

6.6. *Compute the intrinsic volumes of a cone in $\mathbb{R}^2$ described in Example 6.10.*

6.7. *Derive the kinematic formula for two cones in $\mathbb{R}^2$ described in Example 6.11.*

6.8. *Prove the following properties of the statistical dimension of close convex clones:*

*1 The sum of the statistical dimension of a cone $\mathsf{C} \subset \mathbb{R}^n$ and that of its polar cone $\mathsf{C}^o \subset \mathbb{R}^n$ satisfies*

$$\delta(\mathsf{C}) + \delta(\mathsf{C}^o) = n.$$

*2 For the direct product of two closed convex cones $\mathsf{C}_1$ and $\mathsf{C}_2$, we have:*

$$\delta(\mathsf{C}_1 \times \mathsf{C}_2) = \delta(\mathsf{C}_1) + \delta(\mathsf{C}_2).$$

6.9. *In the derivation of (6.2.26), first, show that for $\boldsymbol{g} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, we have*

$$\mathbb{E}_{\boldsymbol{g}} \left[ \|\boldsymbol{g}_{\mathsf{I}} - t\boldsymbol{\sigma}_{\mathsf{I}}\|_2^2 \right] = |\mathsf{I}|(1 + t^2).$$

*Second, discuss how you can solve the following minimization problem:*

$$\psi(\eta) = \min_{t \geq 0} \left\{ \eta(1 + t^2) + 2(1 - \eta) \int_{s=t}^{\infty} (s - t)^2 \varphi(s) ds \right\}?$$

# 7 Nonconvex Methods for Low-Dimensional Models

"*The mathematical sciences particularly exhibit order, symmetry, and limitations; and these are the greatest forms of the beautiful.*"

– Aristotle, *Metaphysica*

## 7.1 Introduction

As engineering and the sciences become increasingly data and computation driven, the role of optimization has expanded to touch almost every stage of the data analysis pipeline, from the signal and data acquisition to modeling, analysis, and prediction. While the challenges in computing with physical data are many and varied, basic recurring issues arise from *nonlinearities* at different stages of this pipeline:

- ***Nonlinear Measurements*** *are ubiquitous in imaging, optics, and astronomy. A canonical example are magnitude measurements, which arise when, due to physical limitations, it is easy to measure the (Fourier) modulus of a complex signal, but hard to measure the phase. For example, we might measure the Fourier magnitude of a complex signal* $\boldsymbol{x} \in \mathbb{C}^n$ *[Pat34, Pat44, SEC+15, JEH17]:*[1]

$$\underset{\text{observation}}{\boldsymbol{y}} = \left| \mathcal{F}\left( \underset{\text{unknown signal}}{\boldsymbol{x}} \right) \right| \in \mathbb{R}^m. \tag{7.1.1}$$

  *Here,* $\boldsymbol{x}$ *represents a signal or image of interest, and the goal is to reconstruct* $\boldsymbol{x}$ *from the* nonlinear *measurements* $\boldsymbol{y}$*. This is sometimes called a* Fourier phase retrieval *problem.*
- ***Nonlinear Models*** *are often well-suited to express the variability of real datasets. For example, observations in microscopy, neuroscience, and astronomy can often be approximated as sparse superpositions of basic motifs.*[2] *We can cast the problem of finding these motifs as one of seeking a representation of the form*

$$\underset{\text{data}}{\boldsymbol{Y}} = \underset{\text{motifs}}{\boldsymbol{A}} \; \underset{\text{sparse coefficients}}{\boldsymbol{X}}. \tag{7.1.2}$$

[1] In contrast, in the MRI example of Section 2.1 of Chapter 2, we studied a much simplified linear model in which we assume to have the full complex measurements of Fourier transform of a brain image. In reality that is not the case.

[2] Mathematically, one may view such motifs as the atoms of a dictionary that we have studied in the previous chapter.

*Here, the columns of $\boldsymbol{Y} \in \mathbb{R}^{m\times p}$ are observed data vectors, the columns of $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ are basic motifs, and $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ is a sparse matrix of coefficients that expresses each observed data point as a superposition of motifs. This is sometimes called a* sparse dictionary model. *A typical goal is to infer both $\boldsymbol{A}$ and $\boldsymbol{X}$ from observed data. Because both $\boldsymbol{A}$ and $\boldsymbol{X}$ are unknown, this model should be considered nonlinear (strictly, bilinear). Natural images may have even more variability, which is better modeled by hierarchical models (convolutional neural networks) with more complicated nonlinearities [LB95a, GPAM+14, GBC16].*

### 7.1.1 Nonlinearity, Symmetry, and Nonconvexity

In the two examples described above, nonlinearities are not just a nuisance: they are part of the structure of the problems we face. They have strong implications on the sense in which we can hope to solve these problems, and, as we will see in this chapter, on our ability to efficiently compute solutions.

Notice that both models exhibit certain *symmetries*. The model $\boldsymbol{y} = |\mathcal{F}(\boldsymbol{x})|$ in (7.1.1) exhibits a *phase symmetry*: both $\boldsymbol{x}$ and $\boldsymbol{x}e^{i\phi}$ (for any $\phi \in [0, 2\pi)$) produce the same observation $\boldsymbol{y}$. The sparse dictionary model $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$ in (7.1.2) exhibits a *permutation symmetry*: for any signed permutation $\boldsymbol{\Pi}$, $(\boldsymbol{A}, \boldsymbol{X})$ and $(\boldsymbol{A}\boldsymbol{\Pi}, \boldsymbol{\Pi}^*\boldsymbol{X})$ produce the same observation $\boldsymbol{Y}$.[3] In either case, we can only hope to recover the physical ground truth up to these basic symmetries.

*Nonconvex Programs from Symmetry.*
A typical computational approach to find the correct solution is to formulate an optimization problem

$$\min_{\boldsymbol{z}} \varphi(\boldsymbol{z}), \tag{7.1.3}$$

and attempt to solve it with iterative methods such as gradient descent [Cau47].[4] Here, $\boldsymbol{z}$ represents the signal or model to be recovered – for example, in phase retrieval, $\boldsymbol{z} = \boldsymbol{x}$, while in dictionary learning the optimization variable $\boldsymbol{z}$ is the pair $(\boldsymbol{A}, \boldsymbol{X})$. Typically, $\varphi(\cdot)$ measures quality of fit to observed data and the extent to which the solution satisfies assumptions such as sparsity. As we shall see, most natural choices of $\varphi$ inherit the symmetries of the data generation model: e.g., for phase recovery, we have

$$\varphi(e^{i\theta}\boldsymbol{x}) = \varphi(\boldsymbol{x}), \quad \forall \theta \in [0, 2\pi) = \mathbb{S}^1,$$

while for dictionary learning,

$$\varphi((\boldsymbol{A}, \boldsymbol{X})) = \varphi((\boldsymbol{A}\boldsymbol{\Pi}, \boldsymbol{\Pi}^*\boldsymbol{X})), \quad \forall \boldsymbol{\Pi} \in \mathsf{SP}(n),$$

[3] Here, and below, the notation $\boldsymbol{M}^*$ denotes the complex conjugate transpose of a matrix $\boldsymbol{M}$. If $\boldsymbol{M}$ is real-valued, this is simply the matrix transpose.

[4] We will give a full exposition of optimization methods in the next part of the book, in particular Chapter 9 for nonconvex programs. In this chapter, we focus on characterizing geometric properties of the optimization problems and their algorithmic implications.

where $\mathsf{SP}(n)$ indicates the group of signed permutations. As we see, *symmetries of the observation models become symmetries in the objective function of the associated optimization problems.*

If we are judicious in our choice of $\varphi(\cdot)$, we can hope that the true $\boldsymbol{x}$ is a (near) global minimizer; our task becomes one of solving the optimization problem (7.1.3) to global optimality. In contrast to certain applications of optimization (e.g., in finance, logistics, etc.), we care not just about decreasing the objective function, but about obtaining the physical ground truth. As such, we are forced to care not just about ensuring that our algorithms converge, but that they converge to global minimizers.

In applied optimization, a time-honored approach to guaranteeing global optimality is to seek formulations that are *convex*. The global minimizers of a convex function form a convex set. Moreover, every local minimizer (indeed, every critical point) of a convex function is global. As a result, many convex problems can be efficiently solved to global optimality by local methods. This makes the area of convex analysis and optimization a model for how geometric understanding can support practical computation, as we have practiced extensively for sparse and low-rank models in the previous chapters.

Unfortunately, as alluded to above, symmetric programs we encounter in statistics, signal processing and related areas are typically nonconvex [SQW15, JK+17, CLC19, Sun19a], and they do not admit any obvious or meaningful convex relaxation. So we need to look for other geometric principles that will enable us to guarantee high-quality (preferably globally optimal) solutions. Indeed, these problems exhibit multiple global minimizers, which may be disjoint (due to permutation symmetry) or may reside on a continuous nonconvex set (due to rotation or phase symmetry). Any optimization formulation that inherits these symmetries will be most likely nonconvex.[5]

*Worst Case Obstructions to Nonconvex Optimization.*
This observation might suggest a certain pessimism: *nonconvex optimization is impossible in general.* There are simple classes of nonconvex problems (e.g., in polynomial optimization) that are already *NP-hard*. At a more intuitive level, there are two geometric obstructions to solving nonconvex problems globally. First, nonconvex problems can exhibit *spurious local minimizers*, i.e. local minimizers that are not global. Local descent methods can get trapped; finding the global optimum is hard in general. Perhaps surprisingly, even finding a *local* minimizer can be NP-hard in general [MK87, Nes00]. Figure 7.1 (right) illustrates one of the challenges: it is possible to construct objective functions that are so flat that it is impossible to efficiently determine a direction of descent.

Of course, it is possible to find global optima under minimal assumptions by

[5] **Disclaimer**: Not *every* symmetric problem is nonconvex. Indeed, the objective function $\varphi(\boldsymbol{z}) = \frac{1}{2}\|\boldsymbol{z}\|_2^2$ is rotationally symmetric $\varphi(\boldsymbol{R}\boldsymbol{z}) = \varphi(\boldsymbol{z})$ for all $\boldsymbol{R} \in \mathsf{O}(n)$, $\boldsymbol{z} \in \mathbb{R}^n$ and convex. It is easy to construct additional examples of this type. However, the symmetric

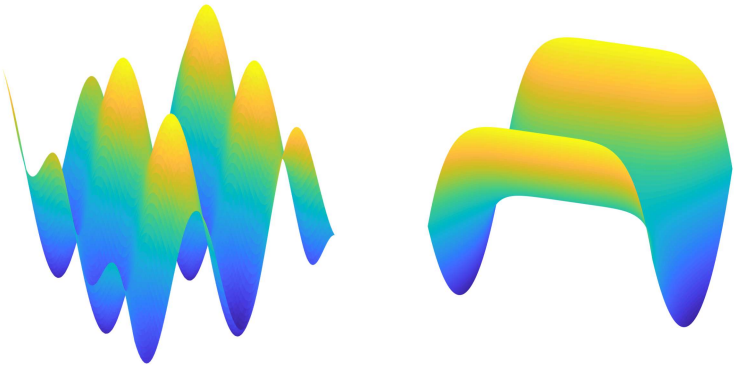



**Figure 7.1 Two Geometric Obstructions to Nonconvex Optimization.** Descent methods can become trapped near local minimizers (left) or stagnate near flat saddle points (right).

exhaustively exploring the space of optimization, e.g., by discretization of the space [EMS18] or by random search [Haj90, BLO05]. The worst-case obstructions described above still rear their heads, in the form of search times that are exponential in dimension. Such a brute force approach is only applicable to problems in which the dimension of the search space is not so-high.

*Calculus and the Local Geometry of Optimization.*
Because of these worst-case obstructions, the classical literature on efficient nonconvex optimization[6] has focused on guaranteeing

1 convergence to some critical point ($\bar{\boldsymbol{z}}$ such that $\nabla\varphi(\bar{\boldsymbol{z}}) = \boldsymbol{0}$),
2 or convergence to some local minimizer, for functions $\varphi$ which are not too flat.

The curvature of a smooth function $\varphi(\cdot)$ around a critical point $\bar{\boldsymbol{z}}$ can be studied through the Hessian $\nabla^2\varphi(\bar{\boldsymbol{z}})$. If $\nabla^2\varphi(\bar{\boldsymbol{z}})$ is nonsingular, the signs of its eigenvalues completely determine whether $\bar{\boldsymbol{z}}$ is a minimizer, maximizer or saddle point – see Figure 7.2 (right). In particular, if $\bar{\boldsymbol{z}}$ is a saddle point or a maximizer, there is a direction of negative curvature – a direction along which the second derivative is negative. This information can be used to escape saddles and converge to a local minimizer, either explicitly (using the Hessian) or implicitly (using gradient information to approximate the negative curvature direction).

In Chapter 9, we will introduce a variety of iterative methods that trade-off in various ways between the amount of computation used to determine a good

problems encountered in statistics, signal processing, and related areas are typically nonconvex; moreover their nonconvexity can be directly attributed to symmetry.

[6] We will systematically study representative algorithms for nonconvex optimization in Chapter 9 and characterize what kind of guarantees they can provide and the associated computational complexity.

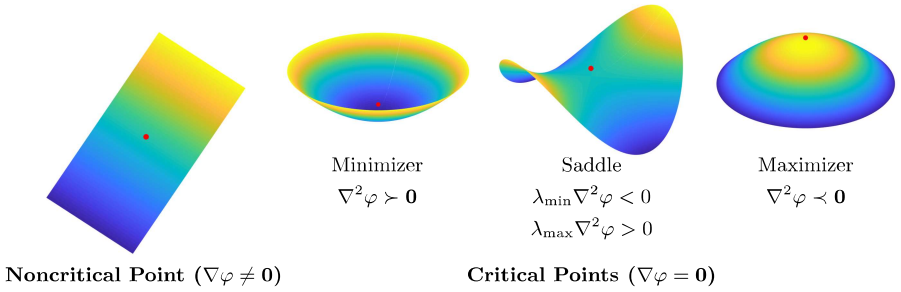



**Figure 7.2 Calculus and the *Local* Geometry of Optimization.** The *gradient* $\nabla\varphi$ captures the slope of the function $\varphi$. At *critical points* $\bar{\boldsymbol{z}}$, $\nabla\varphi(\bar{\boldsymbol{z}}) = \mathbf{0}$. The type of critical point (minimizer, maximizer, saddle) can often be determined from the curvature of $\varphi$ at $\bar{\boldsymbol{z}}$, which is captured by the *Hessian* $\nabla^2\varphi(\bar{\boldsymbol{z}})$.

direction of negative curvature at a given iteration and the number of iterations required to converge [Gol80, CGT00, NP06, LSJR16, JNJ18, LPP+19]. However, the high-level message of these methods is consistent: if all critical points are nondegenerate[7], we can escape them and efficiently converge to a local minimizer. In fact, slightly less is required: it is enough that every non-minimizing critical point have a direction of strict negative curvature[8] [JGN+17, JNJ18, LPP+17, LSJR16].

Results of this nature control the worst-case behavior of methods over very broad classes of problems. In such a general setting, it is not possible to provide strong guarantees on *what* local minimizer methods converge to, and whether that minimizer is global. Nevertheless, it is difficult to overstate the impact of this kind of thinking for stimulating the development of useful methods and elucidating their properties. Moreover, methods developed to guarantee good worst-case performance often outperform their worst-case guarantees on practical problem instances – witness longstanding "folk theorems" on the ease optimizing neural networks [CHM+14, Kaw16, SJL18, AZLS19, DLL+19, Sun19b], solving problems in quantum mechanics [KKP+18, STDV18, HLWY19] or clustering separated data [QZC19, KQC+19, QZC20, WYD20]. Delineating problem classes that capture the difficulty (or ease!) of naturally occurring optimization problems is a pressing challenge for the mathematics of data science [SQW15, JK+17, CLC19, Sun19a].

[7] In the language of differential topology, if the function $\varphi$ is Morse [Mil63, Bot82].

[8] In the recent literature, this is called a "strict saddle" property [GHJY15, SQW15]. Concrete rates of convergence are typically stated in terms of quantitative versions of this property, which explicitly control the size of the gradient and the smallest eigenvalue of the Hessian uniformly over the domain of optimization.

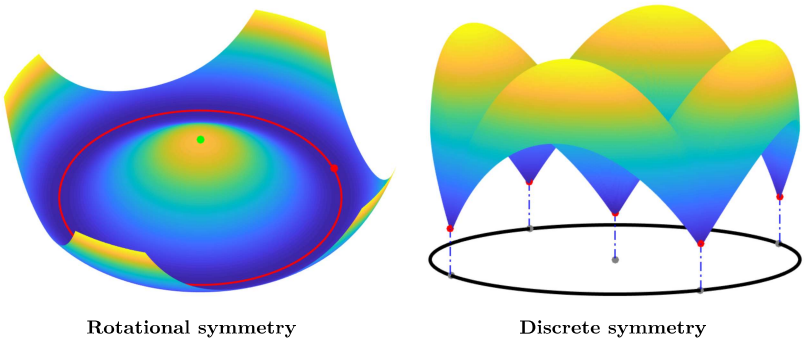



**Figure 7.3 Symmetry and the *Global* Geometry of Optimization.** Model problems with continuous (left) and discrete (right) symmetry. For these particular problems, and others we will study, every local minimizer is global.

### 7.1.2 Symmetry and the Global Geometry of Optimization

The goal of this chapter is to illustrate a particular family of nonconvex problems associated with low-dimensional models which, under surprisingly mild conditions, can be solved globally with efficient methods. This family includes a number of contemporary problems in signal processing, data analysis and related fields [SQW15, JK+17, CLC19, Sun19a]. The most important high-level property of these problems is that they are all *symmetric* – in slightly more formal language:

DEFINITION 7.1 (Symmetric Function). *Let $\mathbb{G}$ be a group acting on $\mathbb{R}^n$. A function $\varphi : \mathbb{R}^n \to \mathbb{R}^{n'}$ is $\mathbb{G}$-symmetric if for all $\boldsymbol{z} \in \mathbb{R}^n$, $\mathfrak{g} \in \mathbb{G}$, $\varphi(\mathfrak{g} \circ \boldsymbol{z}) = \varphi(\boldsymbol{z})$.*

As argued above, symmetry forces us to grapple with properties of nonconvex functions. On the other hand, the particular symmetric nonconvex functions encountered in practice are often quite benign. Figure 7.3 shows two examples – one with rotational symmetry ($\mathbb{G}$ an orthogonal group) and one with discrete symmetry ($\mathbb{G}$ a discrete group, such as the signed permutations). We will develop these examples in more mathematical detail below. For now, we simply observe that these two instances do not exhibit spurious local minimizers or flat saddles. The absence of these worst-case obstructions can be attributed to symmetry. In slogan form, we shall see that:

**Slogan 1:** *the (only!) local minimizers are symmetric versions of the ground truth.*
**Slogan 2:** *a local critical point has negative curvature in directions that break symmetry.*

When these two slogans are in force, efficient (local) methods produce global minimizers. Moreover, symmetry constrains the global layout of the critical points, leading to additional structure that facilitates efficient optimization. We will show examples where the saddle points of symmetric problems "cascade",

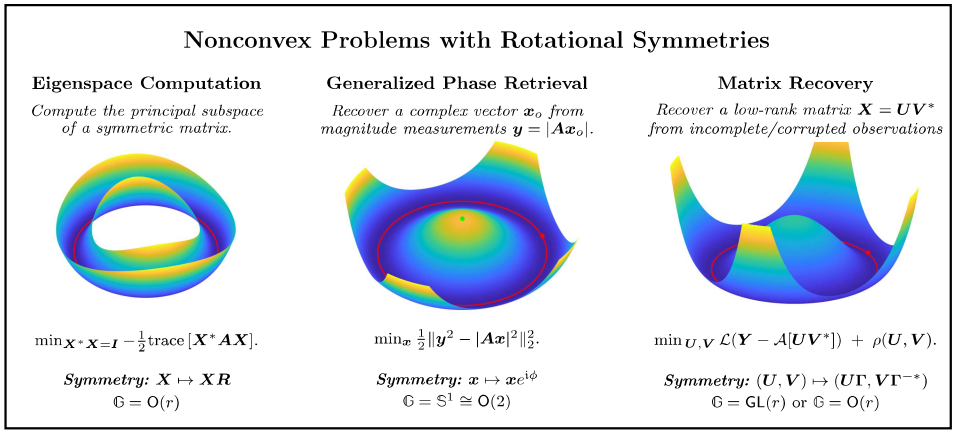



**Figure 7.4** Three examples of nonconvex optimization problems with rotational symmetries (Section 7.2). Each of these three tasks can be reduced to optimization problems in various ways; for each, we give a representative formulation and discuss its symmetries.

with negative curvature directions feeding into negative curvature directions, a property which appears to prevent first order methods from stagnating [GBW19].

Before we embark, a few disclaimers are in order. First, slogans 1 and 2 are only slogans. As we will see, they have been established rigorously for specific problems under specific (restrictive) technical hypotheses. We hope to convey a sense of the beauty and robustness of certain observed phenomena in optimization, while also making clear that the existing mathematics supporting these claims is, in places, lacking uniformity and simplicity. There is a need for more unified analysis and better technical tools. We highlight some potential avenues for this in Section 7.4. The second, more fundamental, disclaimer is that not all symmetric problems have benign global geometry. It is easy to construct counterexamples. Nevertheless, as we will see, symmetry provides a lens through which one can understand the geometric properties that enable efficient optimization for our particular family of problems. Moreover, when we study these problems through their symmetries, common structures and common intuitions emerge: problems with similar symmetries exhibit similar geometric properties and behaviors.

### 7.1.3 A Taxonomy of Symmetric Nonconvex Problems

In this chapter, we identify two families of symmetric nonconvex problems, which exhibit similar geometric characteristics.

- The first family of problems exhibit continuous *rotational symmetries*: the

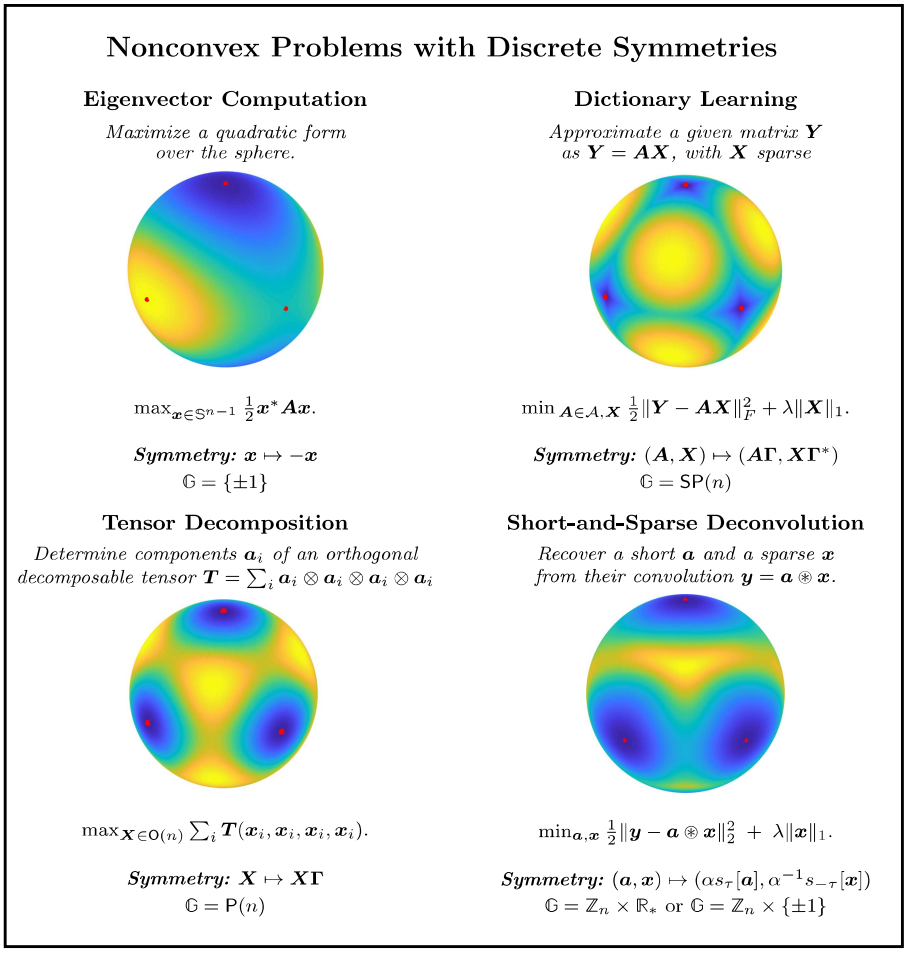



**Figure 7.5** Four examples of problems with discrete symmetries. We discuss this family of problems in more detail in Section 7.3.

group $\mathbb{G}$ is $\mathsf{O}(n)$ or $\mathsf{SO}(n)$. The phase retrieval problem described above is a canonical example; Figure 7.4 illustrates this family.

- The second family of problems exhibit *discrete symmetries*: signed permutations $\mathsf{SP}(n)$, signed shifts $\mathbb{Z}_n \times \{\pm 1\}$, or products of these. The dictionary learning problem discussed above is a canonical example; Figure 7.5 shows several others.

In the remainder of this chapter, we explore the geometry of these two families of problems in more depth. Section 7.2 studies problems with rotational symmetries, beginning with a very simple model problem in which the goal to recover a single complex scalar from magnitude measurements. The analysis helps extract conclusions that carry over to more complicated measurement models for

phase recovery [CESV13, CLS15b, SEC+15, SQW18, FS20] and related problems in low-rank matrix factorization and recovery [GLM16, GJZ17, CLC19].

Section 7.3 studies problems with discrete symmetries, starting again from another simple model problem and extracting conclusions that carry over to problems such as dictionary learning [SQW17a, SQW17b, GBW19, QZL+19], blind deconvolution [LS17, ZKW18, KZLW19, LQK+19, LB18, QLZ19] and tensor decomposition [GHJY15, GM17].

As mentioned above, this area is rich with open problems; we highlight a few of these in Section 7.4. These open problems span both geometry and algorithms. Nevertheless, our main focus throughout this survey is geometric: we will concentrate on the connection between symmetry and geometry. As described above, these geometric analyses have strong implications: in many cases, they guarantee that problems can be solved globally in polynomial time. In order to keep the development focused on geometric intuitions, we will only treat computational issues at a high level. We recommend the survey paper [CLC19] for a more detailed exposition of issues at the interface of statistics and computation, for problems with rotational symmetry. Section 7.4 also briefly discusses similar considerations for problems exhibiting discrete symmetries, where we refer readers to the paper [QZL+20b] for more computational and application aspects on these problems.

## 7.2 Nonconvex Problems with Rotational Symmetries

In this section, we study the first main class of problems in our taxonomy of symmetric nonconvex problems: problems with continuous *rotational symmetry*. This class includes important model problems in phase recovery [SEC+15, FS20] and low-rank estimation [CLC19]. We begin by developing a few basic intuitions through a toy phase retrieval problem; we then show how these intuitions help to explain the geometry of a range of problems from imaging to machine learning.

### 7.2.1 Minimal Example: Phase Retrieval with One Unknown

We first consider a model problem, in which our goal is to recover a single complex scalar $x_o \in \mathbb{C}$ from $m$ magnitude measurements

$$y_1 = |a_1 x_o| \,, \ldots, \, y_m = |a_m x_o| \,, \tag{7.2.1}$$

where $a_1, \ldots, a_m \in \mathbb{C}$ are known complex scalars. Collecting our observations $y_i$ into a single vector $\boldsymbol{y} \in \mathbb{R}^m$ and collecting the $a_i$ into a single vector $\boldsymbol{a} \in \mathbb{C}^m$, we can express this measurement model more compactly as

$$\boldsymbol{y} \;=\; |\boldsymbol{a} x_o| \,. \tag{7.2.2}$$

Our goal is to determine $x_o$, up to a phase. This is a heavily simplified (indeed, trivialized!) version of the *generalized phase retrieval* problem [CSV13, CLS15b,

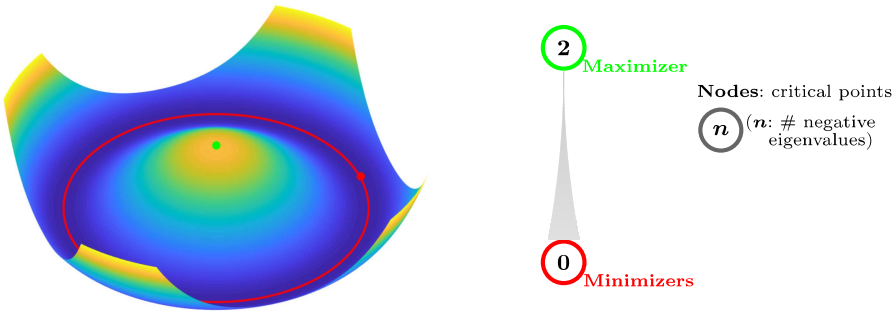



**Figure 7.6 Phase Retrieval with a Single Unknown. Left:** we plot the objective function $\varphi(x)$ for phase retrieval with a single complex unknown. All local minimizers (red) are symmetric copies $x_o e^{\mathrm{i}\phi}$ of the ground truth $x_o \in \mathbb{C}$. There is also a local maximizer (green) at $x = 0$; at this point, $\varphi$ exhibits negative curvature in directions that break symmetry. **Right:** critical points arranged according to objective function $\varphi$, labelled according to their index (number of negative eigenvalues).

SQW18], which we will describe in more detail in Section 7.2.2. Here our goal is simply to understand the consequences of the phase symmetry of the measurement model (7.2.2) for optimization. To this end, we study a model optimization problem,

$$\min \ \varphi(x) \ \doteq \ \tfrac{1}{4} \left\| \boldsymbol{y}^2 - |\boldsymbol{a}x|^2 \right\|_2^2, \tag{7.2.3}$$

which minimizes the sum of squared differences between the squared magnitudes of $\boldsymbol{a}x$ and those of $\boldsymbol{a}x_o$. Note that

$$\varphi(x) = \tfrac{1}{4} \|\boldsymbol{a}\|_4^4 \left( |x|^2 - |x_o|^2 \right)^2 . \tag{7.2.4}$$

This is a function of a complex scalar $x = x_r + \mathrm{i}x_i$. We can study its geometry by identifying $x$ with a two-dimensional real vector $\bar{\boldsymbol{x}} = (x_r, x_i)$. The slope and curvature of the function $\varphi(\bar{\boldsymbol{x}})$ are captured by the gradient and Hessian,

$$\nabla \varphi(\bar{\boldsymbol{x}}) \ = \ \|\boldsymbol{a}\|_4^4 \Big( |x|^2 - |x_o|^2 \Big) \begin{bmatrix} x_r \\ x_i \end{bmatrix}, \tag{7.2.5}$$

$$\nabla^2 \varphi(\bar{\boldsymbol{x}}) \ = \ \|\boldsymbol{a}\|_4^4 \Big( \left( |x|^2 - |x_o|^2 \right) \boldsymbol{I} + 2 \bar{\boldsymbol{x}} \bar{\boldsymbol{x}}^* \Big). \tag{7.2.6}$$

Figure 7.6 visualizes the objective $\varphi(\cdot)$ and its critical points. By setting $\nabla \varphi = 0$, and inspecting the Hessian, we obtain that there exist two families of critical points: global minimizers at $x = x_o e^{\mathrm{i}\phi}$, and a global maximizer at $x = 0$. We notice that:

- ***Symmetric copies of the ground truth are minimizers***. The points $x_o e^{\mathrm{i}\phi}$ are the only local minimizers. In problems with phase ambiguities, we expect a circle $\mathsf{O}(2) \cong \mathbb{S}^1$ of minimizers. In addition, the Hessian is positive

semidefinite, but rank deficient at the global minimizers: the zero curvature direction (along which the objective $\varphi$ is flat) is precisely the direction that is tangent to the set of equivalent solutions $g \circ \boldsymbol{x}_\star$ at $\boldsymbol{x}_\star$ with $g \in \mathbb{S}^1$. Normal to this set, the objective function exhibits positive curvature – a form of restricted strong convexity.

- ***Negative curvature in symmetry breaking directions***. There is a local maximizer at $x = 0$, which is equidistant from the target solutions $\{x_o e^{\mathrm{i}\phi}\}$. At this point $\nabla^2 \varphi \prec \mathbf{0}$; there is negative curvature in every direction, and movement in any direction breaks symmetry.

### 7.2.2 Generalized Phase Retrieval

The univariate phase retrieval problem is an extreme idealization of a basic problem in imaging: recovering a signal from phaseless measurements [CESV13, SEC+15]. This problem arises in many application areas, including electron microscopy [MIJ+02], diffraction and array imaging [BDP+07, CMP10], acoustics [BCE06, Bal10], quantum mechanics [Cor06, Rei65] and quantum information [HMW13], where the goal is to image complex molecular structures. Illuminating a sample with coherent light produces a diffraction pattern, which is approximately the Fourier transform of the sample's density. If we could measure this diffraction pattern, we could recover an image of the sample with atomic resolution, simply by inverting the Fourier transform. However, there is a wrinkle: typically, the magnitude of the Fourier transform is much easier to measure than the phase – the magnitude can be measured by aggregating energy over time, whereas measuring the phase of a high frequency signal requires the detector to be sensitive to very rapid changes. The Fourier phase retrieval problem asks us to reconstruct a complex signal from magnitude measurements only:

$$\text{find} \quad \boldsymbol{x} \quad \text{such that} \quad |\mathcal{F}[\boldsymbol{x}]| \;=\; \boldsymbol{y}.$$

This problem is widespread in scientific imaging [Mil90, Rob93, Wal63, DF87]. It is also challenging: it is ill-posed in one dimension, and in higher dimensions even the most effective numerical methods remain sensitive to initialization and tuning [Fie13]. We refer readers to recent survey papers [SEC+15, JEH15, FS20] for more details. We like to emphasize here that one main reason for this difficulty resides in the symmetries of the measurement operator $|\mathcal{F}[\cdot]|$: in addition to phase symmetry, the mapping $\boldsymbol{x} \mapsto |\mathcal{F}[\boldsymbol{x}]|$ is invariant under shifts and conjugate reversal of the signal $\boldsymbol{x}$. We will discuss more challenges and open problems around Fourier measurements in later sections.

In recent years, the applied mathematics community has investigated variants of the above problem in which the Fourier transform $\mathcal{F}$ is replaced by a more general linear operator $\mathcal{A}(\cdot)$ [CSV13, CESV13, CLS15a]. A "generic" map $\boldsymbol{x} \mapsto |\mathcal{A}[\boldsymbol{x}]|$ has simpler symmetries – typically only a phase symmetry, $|\mathcal{A}[\boldsymbol{x}e^{j\phi}]| = |\mathcal{A}[\boldsymbol{x}]|$. This makes generic phase recovery problems easier to study and easier to solve. While the Fourier model is more widely applicable to physical imaging, the

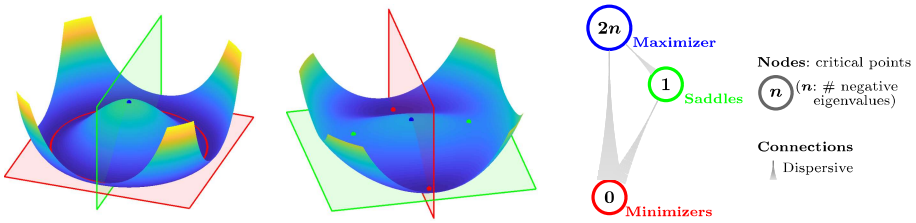



**Figure 7.7 Generalized Phase Retrieval.** We plot two slices of the landscape of the generalized phase retrieval problem with Gaussian measurements. Left: slice containing symmetric copies of the ground truth $\boldsymbol{x}_o e^{\mathrm{i}\phi}$. Middle slice containing minimizers $\boldsymbol{x}_o$, $-\boldsymbol{x}_o$ and one orthogonal direction. Notice that at both the maximizer and saddle points, there is negative curvature in the direction that breaks symmetry between $\boldsymbol{x}_o$ and $-\boldsymbol{x}_o$. Right: critical points arranged according to objective $\mathbb{E}[\varphi]$, labeled with their indices (number of negative eigenvalues). Connections between critical points are "dispersive": downstream negative curvature directions are the image of upstream negative curvature directions under gradient flow.

generic phase retrieval model does capture aspects of certain less conventional imaging setups, including ptychography [YDZ+15, JEH16, Pfe18] (i.e., $\mathcal{A}(\cdot)$ is the Short Time Fourier Transform), coded illuminations [TW15, KBRW19], and coded diffraction patterns [CLS15b]. A model $m$-dimensional version of the generalized phase retrieval problem can be formulated as follows:

$$\text{find} \quad \boldsymbol{x} \in \mathbb{C}^n \quad \text{such that} \quad |\boldsymbol{A}\boldsymbol{x}| = \boldsymbol{y}, \tag{7.2.7}$$

where $\boldsymbol{A} \in \mathbb{C}^{m\times n}$ is a matrix which represents the measurement process.

As in univariate phase retrieval, we can attempt to recover $\boldsymbol{x}_o$ by minimizing the misfit to the observed data, e.g., by solving

$$\min_{\boldsymbol{x}\in\mathbb{C}^n} \varphi(\boldsymbol{x}) \equiv \frac{1}{4m}\sum_{k=1}^{m}\left(y_k^2 - |\boldsymbol{a}_k^*\boldsymbol{x}|^2\right)^2, \tag{7.2.8}$$

where $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m \in \mathbb{C}^n$ are the rows of $\boldsymbol{A}$. We saw above that the univariate version of this function has a very simple landscape, which is dictated almost entirely by phase symmetry, and that it has no spurious local minimizers. *Should we expect similar behavior in this higher dimensional setting?*

## Geometry of Generalized Phase Retrieval

One way of generating intuition is to assume that the sampling vectors $\boldsymbol{a}_i$ are chosen at random, and analyze $\varphi(\boldsymbol{x})$ using tools from statistics. Figure 7.7 visualizes $\varphi(\boldsymbol{x})$ when the $\boldsymbol{a}_k$ are Gaussian vectors[9] and $m$ is large. As $m \to \infty$, $\varphi(\boldsymbol{x})$ converges to its expectation $\mathbb{E}[\varphi]$, which can be calculated in closed form. In Figure 7.7 (left), we can see the characteristic phase symmetry, identical to

[9] Formally, $\boldsymbol{a}_k$ are independent random vectors, with $\boldsymbol{a}_k = \boldsymbol{a}_k^r + \mathrm{i}\boldsymbol{a}_k^i$ with $\boldsymbol{a}_k^r$ and $\boldsymbol{a}_k^i$ independent iid $\mathcal{N}(0, \frac{1}{2})$.

our univariate example above. However, this problem is higher dimensional. Figure 7.7 (center) plots the objective over a two-dimensional slice containing the ground truth and an orthogonal direction. We observe:

- ***Symmetric copies of the ground truth are minimizers.*** All the local minimizers are on the circle of points $\boldsymbol{x}_o e^{\mathrm{i}\phi}$, which corresponds to the ground truth up to the (rotational) phase symmetry. Problems with higher dimensional symmetries will have larger sets of minimizers – e.g., $\mathsf{O}(r)$ symmetry leads to a manifold of minimizers that is isometric to $\mathsf{O}(r)$.
- ***Negative curvature in symmetry-breaking directions.*** In higher dimensional examples, we encounter a variety of local maximizers, saddle points, etc. Nevertheless, these critical points occur near balanced superpositions of equivalent solutions, and exhibit negative curvature in directions $\pm\boldsymbol{x}_o$, which breaks symmetry.
- ***Cascade of saddle points.*** As shown schematically in Figure 7.7, the critical points can be graded based on the number of negative eigenvalues of the Hessian[10]: critical points with higher objective have more negative eigenvalues. Moreover, the objective has a "dispersive" property: upstream negative curvature discourages stagnation near the stable manifold of downstream critical points.

### Practical Variations and Extensions

The exposition in the previous section is still quite idealized: the measurements are Gaussian, and we have infinitely many of them. Moreover, we have assumed a particular objective $\varphi(\boldsymbol{x})$, which is not widely used in practice. Fortunately, the qualitative conclusions of the previous subsection carry over to more structured and challenging settings for generalized phase retrieval.[11] We briefly describe these extensions, while noting technical caveats and open problems.

*Practical Sample Complexity.*
Phase retrieval is a sensing problem; measurements cost resources. It is important to minimize the number of measurements $m$ required to accurately reconstruct $\boldsymbol{x}$. Under the Gaussian model, the particular loss function $\varphi(\cdot)$ in (7.2.8) is a sum of independent heavy-tailed random variables. Relatively straightforward considerations show that when $m \gtrsim n^2$, gradients and Hessians concentrate uniformly about their expectations, and the objective has no spurious local minimizers. This number of samples is clearly suboptimal – $n^2$ measurements to recover about $n$ complex numbers. The challenge is that the objective function (7.2.8) contains fourth moments of Gaussian variables, and is therefore somewhat heavy-tailed. Using arguments that are tailored to this situation, the required number of samples can be improved to $m \gtrsim n \log^3 n$ [SQW18]. Moreover, modifying the

[10] In differential geometry, or more specifically in the Morse theory [Mil63, Bot82], the number of negative eigenvalues is also known as *the index* of the critical point.

[11] But *not* to the Fourier model, which has different symmetries. We discuss challenges and open problems around Fourier measurements in Section 7.3 and Section 7.4.

objective (7.2.8) to remove large terms (a la robust statistics) can improve this to essentially optimal ($m \gtrsim n$) [CC17].[12]

*Different Objective Functions.*
The "squares of the squares" formulation in (7.2.8) is smooth and hence simple to analyze, but is typically not preferred in practice, especially when observations are noisy. Alternatives include $\varphi(\boldsymbol{x}) = \sum_i \left| y_i^2 - |\boldsymbol{a}_i^* \boldsymbol{x}|^2 \right|$ [WGE17], $\varphi(\boldsymbol{x}) = \sum_i |y_i - |\boldsymbol{a}_i^* \boldsymbol{x}||^2$ [DDP17], and maximum likelihood formulations that model (Poisson) noise in the observations $y_i$ [CC17]. Although these formulations differ in details, the major features of the objective landscape are independent of the choice of $\varphi$. For Gaussian $\boldsymbol{a}_i$, the expectation $\mathbb{E}[\varphi]$ has no spurious minimizers; moreover, all objectives have a minimizer at zero and a family of saddle points orthogonal to $\boldsymbol{x}_o$. On the other hand, proving (or disproving) that these objectives have benign global geometry for small $m$ is an open problem. Existing small sample analyses [CC17, WGE17, DDP17] control the behavior of the objective in a neighborhood of $\boldsymbol{x}_o e^{\mathrm{i}\phi}$, and initialize in this neighborhood using statistical properties of the measurement model.

*Structured Measurements.*
Geometric intuitions for Gaussian $\boldsymbol{A}$ carry over to several models that are more closely connected with imaging practice. Examples include convolutional models, in which we observe the modulus of the convolution $\boldsymbol{y} = |\boldsymbol{a} \circledast \boldsymbol{x}|$ of the unknown signal $\boldsymbol{x}$ with a known sequence $\boldsymbol{a}$ [QZEW17] and coded diffraction patterns, in which we make multiple observations $\boldsymbol{y}_l = |\mathcal{F}[\boldsymbol{d}_l \odot \boldsymbol{x}]|$, where $\odot$ denotes an element-wise product [CLS15a]. If the filter $\boldsymbol{a}$ or the masks $\boldsymbol{d}_l$ are chosen at random from appropriate distributions, these structured measurements yield the same asymptotic objective function $\mathbb{E}[\varphi]$. In particular, in the large sample limit ($\boldsymbol{a}$ being long in the convolutional model, or many diffraction patterns in the coded diffraction model), these measurements still lead to optimization problems with no spurious local minimizers. Similar to the situation with nonsmooth objective functions, the best known theoretical sample complexities are obtained by initializing near the ground truth, using statistical properties of $\boldsymbol{A}$. Globally analyzing structured measurements in the small sample regime is a challenging open problem.

The above discussion only scratches the surface of the growing literature on generalized phase retrieval, we refer readers to [SEC+15, JEH15, FS20] for a more comprehensive survey on recent developments. The main purpose of this chapter is to reveal that the unifying thread through all of these models, objectives and problems is the simple model geometry in Figure 7.7. In the next section, we

[12] Other approaches to producing analyses with small sample complexity include restricting the analysis to a small neighborhood of the ground truth, and initializing in this neighborhood using spectral methods that leverage the statistics of the measurement model [CLS15b, WdM15], or forgoing uniform geometric analysis and directly reasoning about trajectories of randomly initialized gradient descent [MWCC18].

will see a similar phenomenon with low-rank matrices: a model geometry from matrix factorization recurs across a sequence of increasingly challenging matrix recovery problems.

### 7.2.3 Low Rank Matrix Recovery

As we have discussed and studied in great detail in Chapter 4, the problem of recovering a low-rank matrix from incomplete and unreliable observations finds broad applications in robust statistics, recommender systems, data compression, computer vision, and so on [DR16]. In matrix recovery problems, the goal is to estimate a matrix $\boldsymbol{X}_o \in \mathbb{R}^{n_1 \times n_2}$ from incomplete or noisy observations. Typically, this problem is ill-posed without some assumptions on the matrix $\boldsymbol{X}_o$. In many applications, $\boldsymbol{X}_o$ can be assumed to be *low rank*, or approximately so:

$$r = \operatorname{rank}(\boldsymbol{X}_o) \ll \min\{n_1, n_2\}. \tag{7.2.9}$$

Any rank-$r$ matrix can be expressed as a product of a tall $n_1 \times r$ matrix and a wide $r \times n_2$ matrix:

$$\boldsymbol{X}_o = \boldsymbol{U}\boldsymbol{V}^*, \quad \boldsymbol{U} \in \mathbb{R}^{n_1 \times r}, \boldsymbol{V} \in \mathbb{R}^{n_2 \times r}. \tag{7.2.10}$$

A very popular strategy for recovering $\boldsymbol{X}_o$ is to start with some objective function $\psi(\boldsymbol{X})$ that enforces consistency with observed data, and then parameterize $\boldsymbol{X}$ in terms of the factors $\boldsymbol{U}$, $\boldsymbol{V}$ [BM03], yielding the optimization problem

$$\min_{\boldsymbol{U},\boldsymbol{V}} \varphi(\boldsymbol{U}, \boldsymbol{V}) \equiv \psi(\boldsymbol{U}\boldsymbol{V}^*). \tag{7.2.11}$$

#### Symmetries of Low Rank Models

Formulations like (7.2.11) are almost always nonconvex, due to symmetries of the factorization (7.2.10). Indeed, for any invertible $r \times r$ matrix $\boldsymbol{\Gamma}$,

$$\boldsymbol{U}\boldsymbol{V}^* = \boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{\Gamma}^{-1}\boldsymbol{V}^* = (\boldsymbol{U}\boldsymbol{\Gamma})\left(\boldsymbol{V}\boldsymbol{\Gamma}^{-*}\right)^*. \tag{7.2.12}$$

Because of this ambiguity, the problem (7.2.11) always possess a *general linear* (invertible matrix) symmetry:

$$(\boldsymbol{U}, \boldsymbol{V}) \equiv (\boldsymbol{U}\boldsymbol{\Gamma}, \boldsymbol{V}\boldsymbol{\Gamma}^{-*}), \quad \forall \boldsymbol{\Gamma} \in \mathsf{GL}(r). \tag{7.2.13}$$

Because a general linear matrix $\boldsymbol{\Gamma}$ can have a determinant arbitrarily close to zero, and hence be arbitrarily ill-conditioned, the equivalence class of solutions $(\boldsymbol{U}, \boldsymbol{V})$ has somewhat complicated geometry, as a subset of $\mathbb{R}^{n_1 \times r} \times \mathbb{R}^{n_2 \times r}$.[13] Fortunately, it is not difficult to reduce this general linear symmetry to a simpler and better conditioned orthogonal symmetry $\mathsf{O}(r)$, either by using information about the target $\boldsymbol{X}_o$, or by adding additional penalty terms to (7.2.11).

[13] For example, it is neither closed nor bounded.

*Rotational Symmetries for Symmetric $\boldsymbol{X}_o$.*
If the target solution $\boldsymbol{X}_o$ is *symmetric and positive semidefinite*, then it admits factorization of the form $\boldsymbol{X}_o = \boldsymbol{U}_o\boldsymbol{U}_o^*$, and so we can take $\boldsymbol{U} = \boldsymbol{V}$. This gives a slightly simpler problem

$$\min_{\boldsymbol{U}} \ \varphi(\boldsymbol{U}) \equiv \psi(\boldsymbol{U}\boldsymbol{U}^*), \tag{7.2.14}$$

with a smaller symmetry group. For any $\boldsymbol{\Gamma} \in \mathsf{O}(r)$, $\boldsymbol{U}\boldsymbol{U}^* = \boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{\Gamma}^*\boldsymbol{U}^* = (\boldsymbol{U}\boldsymbol{\Gamma})(\boldsymbol{U}\boldsymbol{\Gamma})^*$, and so the problem (7.2.14) exhibits an orthogonal symmetry $\varphi(\boldsymbol{U}) \equiv \varphi(\boldsymbol{U}\boldsymbol{\Gamma})$, for all $\boldsymbol{\Gamma} \in \mathsf{O}(r)$.

*Rotational Symmetries for General $\boldsymbol{X}_o$ via Penalization.*
For general (non-symmetric) matrices $\boldsymbol{X}$, it is possible to additional penalties to (7.2.11) in such a way that the general linear symmetry reduces to an orthogonal symmetry. At a high level, the idea is to add a penalty $\rho(\boldsymbol{U}, \boldsymbol{V})$ that enforces $\boldsymbol{U}^*\boldsymbol{U} \approx \boldsymbol{V}^*\boldsymbol{V}$; this prevents $\boldsymbol{U}$ and $\boldsymbol{V}$ from having vastly different scales.[14] The penalty $\rho$ can be chosen such to be $\mathsf{O}(r)$-symmetric, such that the combined problem

$$\min_{\boldsymbol{U},\boldsymbol{V}} \ \varphi(\boldsymbol{U}, \boldsymbol{V}) \equiv \phi(\boldsymbol{U}, \boldsymbol{V}) + \rho(\boldsymbol{U}, \boldsymbol{V}), \tag{7.2.15}$$

possesses an $\mathsf{O}(r)$ symmetry: $\varphi(\boldsymbol{U}, \boldsymbol{V}) \equiv \varphi(\boldsymbol{U}\boldsymbol{\Gamma}, \boldsymbol{V}\boldsymbol{\Gamma})$, for all $\boldsymbol{\Gamma} \in \mathsf{O}(r)$.

*Model Problems and the Matrix Recovery Zoo.*
There are many variants of matrix recovery, which are motivated by different applications and impose different assumptions on the observations and the noise [DR16, GJZ17, CLC19]. Although these problems have their own technical challenges, they have certain qualitative features in common. At a slogan level, "matrix *recovery* problems act like matrix *factorization* problems" [GJZ17]. In the next section, we will begin by describing in detail the geometry of matrix factorization, and then describe how these intuitions carry over to matrix recovery from incomplete or unreliable observations.

### Geometry of Matrix Factorization

Our first model problem starts with a complete, noise-free observation $\boldsymbol{Y} = \boldsymbol{X}_o$ of a symmetric, positive semidefinite matrix $\boldsymbol{X}_o \in \mathbb{R}^{n\times n}$ of rank $r < n$, and attempts to factor it as $\boldsymbol{X}_o = \boldsymbol{U}\boldsymbol{U}^*$ by minimizing the misfit to the observed data [LLA+19]:

$$\min_{\boldsymbol{U}\in\mathbb{R}^{n\times r}} \ \varphi(\boldsymbol{U}) \ \doteq \ \tfrac{1}{4}\|\boldsymbol{Y} - \boldsymbol{U}\boldsymbol{U}^*\|_F^2 . \tag{7.2.16}$$

This is a nonconvex optimization problem, with orthogonal symmetry $\varphi(\boldsymbol{U}) \equiv \varphi(\boldsymbol{U}\boldsymbol{\Gamma})$. Figure 7.8 visualizes the the objective landscape for this problem. It

[14] For example, $\rho(\boldsymbol{U}, \boldsymbol{V}) = \frac{1}{2}\|\boldsymbol{U}^*\boldsymbol{U} - \boldsymbol{V}^*\boldsymbol{V}\|_F$ accomplishes this.

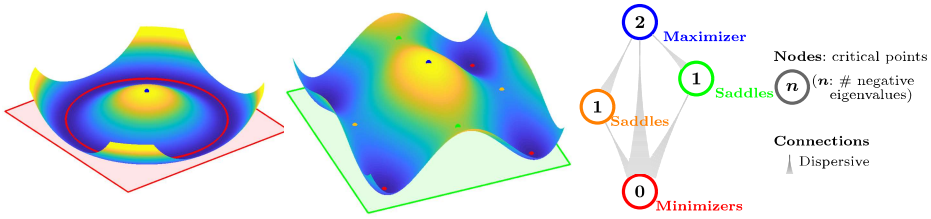



**Figure 7.8 Geometry of Matrix Factorization.** Geometry of a model problem in which the target $\boldsymbol{X}_o$ is a symmetric matrix of rank two, with eigenvalues $\frac{3}{4}$ and $\frac{1}{2}$. **Left:** plot of the objective $\varphi$ over a slice of the domain containing all optimal solutions. **Center:** two families of saddle points, corresponding to rank-one approximations. **Right:** objective value $\varphi$ versus index for the four families of critical points in this problem. Again the critical points are *graded*, in the sense that $\varphi$ decreases with decreasing index, and the paths between them are dispersive, in the sense that downstream negative curvature directions are the image of upstream negative curvature directions under gradient flow.

turns out that the critical points of $\varphi$ are dictated by the eigenvalue decomposition of the symmetric matrix $\boldsymbol{X}_o$ – *every critical point* $\boldsymbol{U}$ *is generated by selecting and appropriately scaling a subset of the eigenvectors of* $\boldsymbol{X}_o$*, and then applying a right rotation* $\boldsymbol{U} \mapsto \boldsymbol{U}\boldsymbol{R}$. At a slogan level, critical points correspond to "under-factorizations" of the ground truth. Inspecting the Hessian, we find that:

- ***Symmetric copies of the ground truth are minimizers***. Local minimizers are the critical points which select all of the top $r$ eigenvectors, which correspond to the ground truth up to rotation symmetry;
- ***Negative curvature in symmetry-breaking directions***. At a saddle point, there is strict negative curvature in any direction which increases the number of top eigenvectors that participate.
- ***Cascade of saddle points***. Saddle points are critical points selecting subsets of the top $r$ eigenvectors. These saddle points can be graded based on number of selected eigenvectors. [15]

Figure 7.8 (center) visualizes these effects.

This model geometry carries over to non-symmetric matrices. For example, considering a penalized low-rank estimation problem

$$\min_{\boldsymbol{U} \in \mathbb{R}^{n_1 \times r}, \boldsymbol{V} \in \mathbb{R}^{n_2 \times r}} \varphi(\boldsymbol{U}, \boldsymbol{V}) \doteq \tfrac{1}{4} \|\boldsymbol{Y} - \boldsymbol{U}\boldsymbol{V}^*\|_F^2 + \rho(\boldsymbol{U}, \boldsymbol{V}), \tag{7.2.17}$$

we obtain a problem with $\mathsf{O}(r)$ symmetry. Critical points are generated by appropriately scaling subsets of the *singular* vectors of $\boldsymbol{Y}$. We leave the details to the reader as an exercise.

[15] A natural descent algorithm only visit at most $r$ saddle points whose trajectory depends on the containment of the active eigenvectors at those saddle points.

### From Factorization to Matrix Recovery and Completion

We next describe how precise geometric analyses of matrix factorization extend to the more realistic problem of recovering a low-rank matrix from incomplete and unreliable observations, which we have studied in Chapter 4 via convex optimization. As we shall see, with their natural nonconvex formulations, the matrix recovery problems often retain important qualitative features of matrix factorization. We will illustrate this phenomenon through several instances of a model recovery problem, in which we observe $m$ linear functions of an unknown matrix $\boldsymbol{X}_o \in \mathbb{R}^{n_1 \times n_2}$:

$$y_i \;=\; \langle \boldsymbol{A}_i, \boldsymbol{X}_o \rangle\,, \quad 1 \le i \le m, \tag{7.2.18}$$

and the goal is to recover $\boldsymbol{X}_o$. This model is flexible enough to represent matrix completion from missing entries [CR09], as well as more exotic sensing problems [RFP10,DR16]. We can write this observation model more compactly by defining a linear operator $\mathcal{A} : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^m$ with $\mathcal{A}(\boldsymbol{X}) := [\langle \boldsymbol{A}_i, \boldsymbol{X} \rangle]_{1 \le i \le m}$. In this notation,

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{X}). \tag{7.2.19}$$

If $m < n_1 n_2$, the number of observations is smaller than the number of unknowns, and the recovery problem is ill-posed. Fortunately, matrices encountered in applications have low-complexity structures; for instance, they are usually low-rank or approximately so. As above, a rank-$r$ $\boldsymbol{X}_o$ admits a factorization $\boldsymbol{X}_o = \boldsymbol{U}_o \boldsymbol{V}_o^*$, so that that we can enforce this low-rank structure by directly recovering the factors $\boldsymbol{U} \in \mathbb{R}^{n_1 \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n_2 \times r}$, up to symmetry.[16] A natural approach is to minimize the misfit to the observed data:

$$\begin{aligned} \min_{\boldsymbol{U},\boldsymbol{V}} \varphi(\boldsymbol{U},\boldsymbol{V}) &\doteq \frac{1}{4m} \sum_{i=1}^{m} \left(y_i - \langle \boldsymbol{A}_i, \boldsymbol{U}\boldsymbol{V}^* \rangle\right)^2 + \rho(\boldsymbol{U},\boldsymbol{V}) \\ &= \frac{1}{4m} \left\| \boldsymbol{y} - \mathcal{A}(\boldsymbol{U}\boldsymbol{V}^*) \right\|_F^2 \;+\; \rho(\boldsymbol{U},\boldsymbol{V}), \end{aligned} \tag{7.2.20}$$

where as above $\rho$ is a regularizer that encourages the factors to be balanced.

*Matrix Sensing.*

If $\mathcal{A} = \mathcal{I}$ is the identity operator, (7.2.20) is simply the factorization problem. In this special situation, the measurement operator $\mathcal{A}$ *exactly* preserves the geometry of *all* $n_1 \times n_2$ matrices, in the sense that $\|\mathcal{A}[\boldsymbol{X}]\|_F = \|\boldsymbol{X}\|_F$ for all $\boldsymbol{X}$. When the number of measurements is small ($m < n_1 n_2$), this is impossible. Fortunately, (7.2.20) still "behaves like factorization", and hence can be used to recover $\boldsymbol{X}_o$, as long as $\mathcal{A}$ *approximately* preserves the geometry of the *low-rank* matrices

[16] For simplicity, we here and below assume the rank $r$ is known. As it turns out this is not so crucial: When $r$ is not known, one may simply over-parameterize the matrix with larger factors $\boldsymbol{U} \in \mathbb{R}^{n_1 \times n}$ and $\boldsymbol{V} \in \mathbb{R}^{n_2 \times n}$ where $n$ can be much larger than the true $r$. Then one can show that, gradient descent algorithms in general converge the correct low-rank solution. We leave the details as exercises to the reader.

– a much lower dimensional set [PKCS16, BNS16, ZLTW18, LZT18, LLA+19].[17] When this approximation is sufficiently accurate, there is a bijection between the critical points of the sensing problem (7.2.20) and those of factorization, which preserves the index (number of negative eigenvalues). Under this condition, every local minimum of the sensing problem is global [BNS16].

*Matrix Completion.*

The most practical and important instance of the general sensing model (7.2.20) is the *matrix completion* problem [CR09], in which the goal is to recover a low-rank matrix from a subset of $m < n_1 n_2$ entries, supported on say $\Omega$. This model problem arises e.g., in collaborative filtering [RS05, Kor09], where the goal is to predict users' preferences for various products based a few observed preferences. Variants of this problem also appear in sensor networks (determining positions of sensors from a few distance measurements) [BLWY06,SY07], imaging (recovering shape from illumination[18]) [WGS+10, ZYZY14], and the geosciences [YMO13, KDSA+15], just to name a few.

We have studied the matrix completion problem in great detail in Chapter 4 via the convex approach. Here, for its natural nonconvex formulation:

$$\min \frac{1}{4m} \|\boldsymbol{y} - \mathcal{P}_\Omega(\boldsymbol{U}\boldsymbol{V}^*)\|_F^2 \ + \ \rho(\boldsymbol{U}, \boldsymbol{V}), \tag{7.2.21}$$

matrix completion also inherits the geometry of matrix factorization, with several technical caveats, which are consequences of the fact that it is challenging to recover $\boldsymbol{X}_o$ that are concentrated on a small number of entries: if we fail to sample these important entries, we will fail to recover $\boldsymbol{X}_o$. This basic issue affects both for the well-posedness of the matrix completion problem and for our ability to solve it globally using nonconvex optimization. Local optimization methods could potentially become trapped in the region of the space in which $\boldsymbol{U}\boldsymbol{V}^*$ is nearly sparse, since the measurements do not effectively sense such matrices. One simple fix is to add an additional regularizer on the rows $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ of the factors, which encourages them to have small norm. This forces the energy of $\boldsymbol{U}\boldsymbol{V}^*$ to be spread across many entries.[19] Ge et al. [GLM16] proved that the resulting problem has benign global geometry whenever we observe a sufficiently large random subset $\Omega$ and the target matrix $\boldsymbol{X}_o$ is not too concentrated on a few entries, in a precise technical sense.[20]

[17] This intuition can be formalized through the *rank restricted isometry property* (rank RIP) [RFP10, DR16], which we have also studied in Chapter 4.

[18] We will feature this particular application thoroughly in Chapter 14.

[19] In detail, one can add a penalty $\rho_{\mathrm{mc}}(\boldsymbol{U}, \boldsymbol{V}) \ = \ \lambda_1 \sum_{i=1}^{n_1} \left(\|\boldsymbol{e}_i^* \boldsymbol{U}\| - \alpha_1\right)_+^4 + \lambda_2 \sum_{j=1}^{n_2} \left(\left\|\boldsymbol{e}_j^* \boldsymbol{V}\right\| - \alpha_2\right)_+^4$ to (7.2.20).

[20] Formally, $\boldsymbol{X}_o$ is $\mu$-incoherent, in the sense that for its compact SVD $\boldsymbol{X}_o = \boldsymbol{U}_o \boldsymbol{\Sigma}_o \boldsymbol{V}_o^*$, we have $\|\boldsymbol{e}_i^* \boldsymbol{U}_o\|_2 \le \sqrt{\mu r / n_1}$ and $\|\boldsymbol{e}_j^* \boldsymbol{V}_o\|_2 \le \sqrt{\mu r / n_2}$.

*Robust Matrix Recovery.*
Many data analysis problems confront the analyst with data sets that are not only incomplete, but also corrupted. Robust matrix recovery is the problem of estimating low-rank matrix $\boldsymbol{X}_o$ from such an unreliable observation (as we have seen in Chapter 5). Different models of corruption may be applicable in different application scenarios. For example, in imaging and vision, individual features (entries of the matrix) may be corrupted, e.g., due to occlusion [CLMW11,PGW+12]. This can be modeled as a sparse error: $\boldsymbol{Y} = \boldsymbol{X}_o + \boldsymbol{S}_o$, with both $\boldsymbol{X}_o = \boldsymbol{U}_o\boldsymbol{V}_o^*$ *and* $\boldsymbol{S}_o$ unknown. We may start from the natural formulation:

$$\min_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{S}} \tfrac{1}{2}\|\boldsymbol{U}\boldsymbol{V}^* + \boldsymbol{S} - \boldsymbol{Y}\|_F^2 + g_s(\boldsymbol{S}) + \rho_r(\boldsymbol{U},\boldsymbol{V}), \tag{7.2.22}$$

where $g_s(\boldsymbol{S})$ is a regularizer that encourages $\boldsymbol{S}$ to be sparse. Partially minimizing with respect to $\boldsymbol{S}$, we obtain

$$\min_{\boldsymbol{U},\boldsymbol{V}} \psi(\boldsymbol{U}\boldsymbol{V}^* - \boldsymbol{Y}) + \rho_r(\boldsymbol{U},\boldsymbol{V}), \tag{7.2.23}$$

where $\psi(\cdot)$ is a new function that measures data fidelity. For example, if $g_s$ is a weighted $\ell^1$ penalty $\lambda\|\cdot\|_1$, then $\psi$ entry-wise is of the form:

$$h_\lambda(u) \doteq \min_x \tfrac{1}{2}(u-x)^2 + \lambda|x|.$$

One can show that so-defined $h_\lambda$ is given by the so-called *Huber function* [Hub92]:

$$h_\lambda(u) \;=\; \begin{cases} \lambda|u| - \lambda^2/2 & |u| > \lambda, \\ u^2/2 & |u| \le \lambda. \end{cases} \tag{7.2.24}$$

We leave the verification as an exercise to the reader.

The problem (7.2.23) is again a matrix factorization problem, but with a different loss $\psi(\boldsymbol{U}\boldsymbol{V}^* - \boldsymbol{Y})$. While there are a number of open issues around the global (and even local! [LZMCSV20, CCD+19]) geometry of this problem, known results again suggest that for certain choices of $g_s$ and $\rho_r$ it indeed inherits the geometry of factorization [CLC19]. Similar to matrix completion, technical issues arise due to the possibility of encountering low-rank matrices $\boldsymbol{U}\boldsymbol{V}^*$ that are themselves sparse. If the regularizer $\rho_r$ is chosen to discourage such solutions, it is possible to prove that the resulting objective function has no spurious local minimizers, and negative curvature at every non-minimizing critical point.

Equation (7.2.22) is just one model for matrix recovery from unreliable observations. Versions in which entire columns of $\boldsymbol{Y}$ are corrupted are also of interest for robust statistical estimation (see e.g., [XCS10]), where they model outlying data vectors. Certain variants of this problem also inherit the geometry of factorization – local minimizers are global, saddle points are generated by partial factorizations of the ground truth, and exhibit negative curvature in directions that introduce additional ground truth factors [LM18]. It is also possible to formulate this version of the robust matrix recovery problem as one of finding a hyperplane that contains the majority of the data points. This dual viewpoint

leads to nonconvex problems with a sign symmetry, which again have benign geometry under certain conditions on the input data [TV18, ZWR+18].

### 7.2.4 Other Nonconvex Problems with Rotational Symmetry

*Other Low-Rank Recovery Problems.*
There are a number of nonlinear inverse problems that can be converted to rank-one recovery problems, and hence inherit the good geometry of low-rank recovery. Examples include subspace deconvolution [ARR14, LLB16, LS17], phase synchronization [Bou16, LXB18, MMMO17, ZB18], community detection [BBV16], amongst others.

*Deep and Linear Neural Networks.*
Most neural network learning problems are nonconvex. Neural network problems arising in practical deep learning typically exhibit complicated symmetries, which include compositions of permutations. For example, for a fully connected network, if we arbitrarily permute the order of the nodes in each intermediate layer, the network can represent the same function. *Linear* neural networks, whose predictions

$$\boldsymbol{y} \approx f(\boldsymbol{x}) = \boldsymbol{W}^L \boldsymbol{W}^{L-1} \cdots \boldsymbol{W}^0 \boldsymbol{x}$$

are a *linear* function of the input $\boldsymbol{x}$, have attracted attention as a more approachable object of theoretical investigation. This model exhibits rotational symmetries at each layer. Using similar considerations to those described above, [Kaw16] and related work prove that every local minimum is global. As with matrix factorization, critical points of natural optimization models correspond to "under-factorizations." However, in contrast to matrix factorization, this problem does possess "flat" saddle points at which the Hessian has no negative eigenvalues – this is the result of the compound effect of symmetries at multiple layers. We will study more general and practical deep networks in Chapter 16. In particular, we will see how certain (symmetric) structural regularization, such as orthogonality for each layer $\boldsymbol{W}$, would be crucial for ensuring good performance of deep networks in practice.

## 7.3 Nonconvex Problems with Discrete Symmetries

In this section, we study nonconvex problems with discrete symmetry groups $\mathbb{G}$. Canonical examples include sparse dictionary learning (signed permutation symmetry) [SQW17a, SQW17b, QZL+19, ZYL+20], sparse blind deconvolution (signed shift symmetry) [ZLK+17, ZKW18, KZLW19, LQK+19, QLZ19, LB18], tensor decomposition [GHJY15, GM17] and clustering (permutation symmetry). Problems of this type are not easily amenable to convexification; understanding nonconvex optimization landscapes becomes critical. Design choices, such as the

choice of objective function and constraints, also seem to play a critical role: many of the examples we review below are formulated as constrained optimization problems over compact manifolds such as spheres or orthogonal groups.[21] We again begin by studying a very simple model problem: *dictionary learning with one-sparse data*. We extract several key intuitions for problems with discrete symmetries, and then examine how these intuitions carry over to less idealized (and more useful!) problem settings.

### 7.3.1 Minimal Example: Dictionary Learning with One Sparsity

We introduce some basic intuitions through a model problem, which is a highly idealized version of *dictionary learning*. In this model problem, we observe a matrix $\boldsymbol{Y}$ which is the product of an orthogonal matrix $\boldsymbol{A}_o \in \mathsf{O}(m)$ (called a dictionary) and a matrix $\boldsymbol{X}_o \in \mathbb{R}^{m\times n}$ whose columns are one-sparse, i.e., each column of $\boldsymbol{X}_o$ has one nonzero entry:

$$\underset{\text{data}}{\boldsymbol{Y}} = \underset{\text{orthogonal dictionary}}{\boldsymbol{A}_o} \quad \underset{\text{1-sparse coefficients}}{\boldsymbol{X}_o}. \tag{7.3.1}$$

This observation model exhibits a **signed permutation symmetry** ($\mathbb{G} = \mathsf{SP}(n)$): for a given pair $(\boldsymbol{A}_o, \boldsymbol{X}_o)$, and any $\boldsymbol{\Gamma} \in \mathsf{SP}(n)$, the pair $(\boldsymbol{A}_o\boldsymbol{\Gamma}, \boldsymbol{\Gamma}^*\boldsymbol{X}_o)$ also reproduces $\boldsymbol{Y}$. The goal is to recover $\boldsymbol{A}_o$ and $\boldsymbol{X}_o$, up to this symmetry.

A natural approach for recovering $\boldsymbol{A}_o$ is to search for an orthogonal matrix $\boldsymbol{A}$ such that $\boldsymbol{A}^*\boldsymbol{Y}$ is *as sparse as possible*:

$$\min\ h(\boldsymbol{A}^*\boldsymbol{Y}) \quad \text{such that} \quad \boldsymbol{A} \in \mathsf{O}(m), \tag{7.3.2}$$

where $h(\boldsymbol{X}) = \sum_{ij} h(\boldsymbol{X}_{ij})$ is a function that promotes sparsity. There are many possible choices for $h$ [ZYL+20, LCD+19, SXZ+20] (and we will explore some in the exercises); for concreteness, here we take $h$ to be the Huber function

$$h_\lambda(u) = \begin{cases} \lambda|u| - \lambda^2/2 & |u| > \lambda, \\ u^2/2 & |u| \le \lambda. \end{cases} \tag{7.3.3}$$

This can be viewed as a differentiable surrogate for the (sparsity promoting) $\ell^1$ norm.

In (7.3.2), we solve for the entire dictionary $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m]$ at once. An even simpler model problem can be formulated by instead solving for the columns $\boldsymbol{a}_i$ one at a time:

$$\min\ h_\lambda(\boldsymbol{a}^*\boldsymbol{Y}) \quad \text{such that} \quad \boldsymbol{a} \in \mathbb{S}^{m-1}. \tag{7.3.4}$$

Here, the goal is to recover a signed column $\pm\boldsymbol{a}_i$ of the dictionary $\boldsymbol{A}$.[22] This problem asks us to minimize an $\ell^1$-like function over the sphere.[23]

[21] Optimization algorithms that exploit structures of such manifolds will be studied in Section 9.6 of Chapter 9.

[22] The entire dictionary can be recovered by solving a sequence of problems of this type; see [SWW12, SQW17a, SQW17b].

[23] The problem (7.3.4) can also be interpreted geometrically as searching for a sparse vector in the linear subspace $\mathrm{row}(\boldsymbol{Y})$; see also [QSW14, QZL+20b].

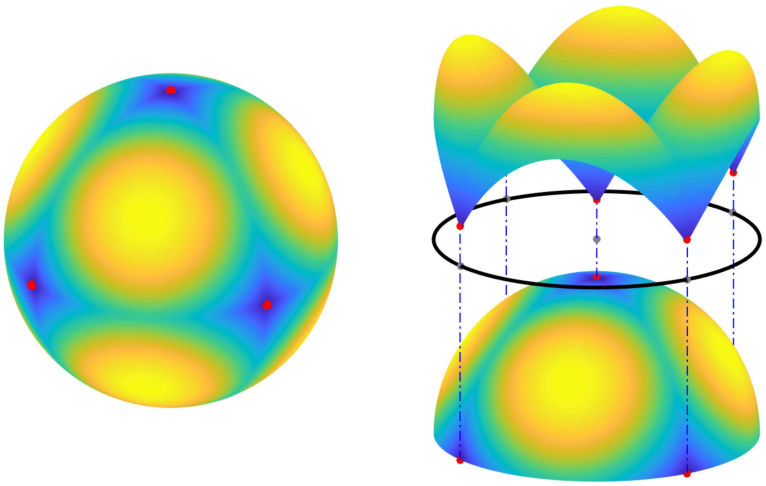


**Figure 7.9 A Model Problem with Discrete Symmetry.** The huber function $h_\lambda(\boldsymbol{u})$ is a differentiable approximation to the $\ell^1$ norm. Minimizing $h_\lambda$ encourages sparsity. **Left**: $h_\lambda(\boldsymbol{u})$ as a function on the sphere $\mathbb{S}^2$. Local minimizers (red) are signed standard basis vectors $\pm\boldsymbol{e}_i$. These are the maximally sparse vectors on $\mathbb{S}^2$. **Right**: graph of $h_\lambda$; notice the strong negative curvature at points that are not sparse.

To further simplify matters, we assume that the true dictionary $\boldsymbol{A}_o$ is the identity matrix. This does not change our geometric conclusions – changing to another $\boldsymbol{A}_o$ simply rotates the objective function. Similarly, since in this model problem each column of $\boldsymbol{X}_o$ has one nonzero entry, we lose little generality in taking $\boldsymbol{X}_o = \boldsymbol{I}$. With these idealizations, the problem simply becomes one of minimizing a sparsity surrogate over the sphere

$$\min \ \varphi(\boldsymbol{a}) \equiv h_\lambda(\boldsymbol{a}) \quad \text{such that} \quad \boldsymbol{a} \in \mathbb{S}^{m-1}. \tag{7.3.5}$$

Here, recovering a signed column of the true dictionary $\boldsymbol{A}_o = \boldsymbol{I}$ corresponds to recovering one of the signed standard basis vectors $\pm\boldsymbol{e}_1, \ldots, \pm\boldsymbol{e}_m$ in this model problem.

*Geometry of the Model Problem.*
The 1-sparse dictionary learning model problem also exhibits a signed permutation symmetry: for any $\boldsymbol{\Gamma} \in \mathsf{SP}(m)$, $\varphi(\boldsymbol{\Gamma}\boldsymbol{a}) = \varphi(\boldsymbol{a})$. The set of target solutions $\pm\boldsymbol{e}_1, \ldots, \pm\boldsymbol{e}_m$ is also symmetric. Figure 7.9 plots the objective function, and these target solutions, in a three dimensional example. Clearly, in this example, these target solutions are the only local minimizers.

To study this phenomenon more formally, we need to understand the slope (gradient) and curvature (Hessian) of $\varphi$ as a function over the sphere $\mathbb{S}^{m-1}$. Recall that as we have encountered an optimization problem over the sphere

in Section 4.2.1 of Chapter 4 when we characterize the computation of singular value decomposition (SVD). The sphere is a smooth manifold; its tangent space at a point $\boldsymbol{a}$ can be identified with $\boldsymbol{a}^\perp$:

$$T_{\boldsymbol{a}}\mathbb{S}^{m-1} = \{\, \boldsymbol{\delta} \mid \boldsymbol{a}^*\boldsymbol{\delta} = 0 \,\}.$$

The orthogonal projector onto the tangent space is simply given by $\boldsymbol{P}_{\boldsymbol{a}^\perp} = \boldsymbol{I} - \boldsymbol{a}\boldsymbol{a}^*$. The slope of $\varphi$ over the sphere (formally, the Riemannian gradient) is simply the component of the standard gradient that is tangent to the sphere:

$$\operatorname{grad}[\varphi](\boldsymbol{a}) = \boldsymbol{P}_{\boldsymbol{a}^\perp} \nabla\varphi(\boldsymbol{a}). \tag{7.3.6}$$

The curvature of $\varphi$ over the sphere is slightly more complicated. For a direction $\boldsymbol{\delta} \in T_{\boldsymbol{a}}\mathbb{S}^{m-1}$, the second derivative of $\varphi$ along the geodesic curve (great circle)[24]

$$\gamma(t) = \exp_{\boldsymbol{a}}(t\boldsymbol{\delta}) = \boldsymbol{a}\cos(t) + \boldsymbol{\delta}\sin(t)$$

is given by $\boldsymbol{\delta}^*\operatorname{Hess}[\varphi](\boldsymbol{a})\boldsymbol{\delta}$, where $\operatorname{Hess}[\varphi]$ is the *Riemannian Hessian*[25]

$$\operatorname{Hess}[\varphi](\boldsymbol{a}) \;=\; \boldsymbol{P}_{\boldsymbol{a}^\perp}\Big( \underbrace{\nabla^2\varphi(\boldsymbol{a})}_{\text{curvature of } \varphi} \;-\; \underbrace{\langle \nabla\varphi(\boldsymbol{a}), \boldsymbol{a}\rangle\, \boldsymbol{I}}_{\text{curvature of the sphere}} \Big)\, \boldsymbol{P}_{\boldsymbol{a}^\perp}. \tag{7.3.7}$$

This expression contains two terms. The first is the standard (Euclidean) Hessian $\nabla^2\varphi$, which accounts for the curvature of the objective function $\varphi$. The second term accounts for the curvature of the sphere itself. Analogous to the case in Euclidean space, critical points are characterized by $\operatorname{grad}[\varphi](\boldsymbol{a}) = \boldsymbol{0}$; curvature can be studied through $\operatorname{Hess}[\varphi](\boldsymbol{a})$.[26]

To study the critical points, we begin by calculating the Euclidean gradient of $\varphi$ given in (7.3.5):

$$\nabla\varphi(\boldsymbol{a}) = \lambda \operatorname{sign}(\boldsymbol{a}) \odot \mathbb{1}_{|\boldsymbol{a}|>\lambda} \;+\; \boldsymbol{a} \odot \mathbb{1}_{|\boldsymbol{a}|\le\lambda}, \tag{7.3.8}$$

where $\odot$ denotes element-wise multiplication. Using this expression, we can show that the Riemannian gradient vanishes ($\operatorname{grad}[\varphi](\boldsymbol{a}) = \boldsymbol{0}$) if and only if $\nabla\varphi(\boldsymbol{a}) \propto \boldsymbol{a}$ (here, $\propto$ denotes proportionality, i.e., $\exists s$ such that $\nabla\varphi(\boldsymbol{a}) = s\boldsymbol{a}$). This occurs whenever

$$\boldsymbol{a} \;\propto\; \operatorname{sign}(\boldsymbol{a}). \tag{7.3.9}$$

We can therefore index critical points by the support $\mathsf{I}$ and sign pattern $\boldsymbol{\sigma}$ of $\boldsymbol{a}$, writing $\boldsymbol{a}_{\mathsf{I},\boldsymbol{\sigma}}$. To understand which critical points are minimizers or saddles, we can study the Hessian $\operatorname{Hess}[\varphi](\boldsymbol{a})$. The Euclidean Hessian is $\nabla^2\varphi(\boldsymbol{a}) = \mathbb{1}_{|\boldsymbol{a}|\le\lambda}$; its Riemannian counterpart is

$$\operatorname{Hess}[\varphi](\boldsymbol{a}_{\mathsf{I},\boldsymbol{\sigma}}) \;=\; \boldsymbol{P}_{\boldsymbol{a}_{\mathsf{I},\boldsymbol{\sigma}}^\perp} \left(\boldsymbol{P}_{|\boldsymbol{a}_{\mathsf{I},\sigma}|\le\lambda} - \lambda|\mathsf{I}|\boldsymbol{I}\right) \boldsymbol{P}_{\boldsymbol{a}_{\mathsf{I},\boldsymbol{\sigma}}^\perp}. \tag{7.3.10}$$

[24] Here $\exp(\cdot)$ represents the exponential map from a tangent vector, here $\boldsymbol{\delta}$, to a geodesic curve on a manifold, here the great circle on the sphere.

[25] This expression can be derived in a simple way by letting $\|\boldsymbol{\delta}\| = 1$, and calculating $\frac{d^2}{dt^2}\big|_{t=0} \varphi\big(\boldsymbol{a}\cos t + \boldsymbol{\delta}\sin t\big)$. We leave this as an exercise to the reader.

[26] For a more general reference to extending the notion of gradient and Hessian to optimization on manifolds, we refer readers to [AMS09].

At critical points $\boldsymbol{a}_{\mathsf{I},\sigma}$ the Hessian exhibits $(|\mathsf{I}| - 1)$ negative eigenvalues, and $n - |\mathsf{I}|$ positive eigenvalues. Based on these calculations, we obtain the following conclusions about the geometry of $\varphi$:

- ***Symmetric copies of the ground truth are minimizers.*** Local minimizers are the signed standard basis vectors $\boldsymbol{a} = \pm\boldsymbol{e}_i$ with the positive Riemannian Hessian; the objective function is strongly convex in the vicinity of local minimizers.
- ***Negative curvature in symmetry breaking directions.*** Saddle points are balanced superpositions of target solutions: $\boldsymbol{a}_{\mathsf{I},\sigma} = \frac{1}{\sqrt{|\mathsf{I}|}} \sum_{i\in\mathsf{I}} \sigma_i \boldsymbol{e}_i$ for $\mathsf{I} \subseteq \{1,\dots,m\}$ and signs $\sigma_i \in \{\pm 1\}$. There is negative curvature in directions $\boldsymbol{\delta} \in \operatorname{span}(\{\boldsymbol{e}_i \mid i \in \mathsf{I}\})$ that break the balance between target solutions.
- ***Cascade of saddle points.*** Saddle points are graded: points $\boldsymbol{a}_{\mathsf{I},\boldsymbol{\sigma}}$ with larger objective value have more directions of negative curvature. Moreover, similar to the examples discussed in the last section, the objective function exhibits a "dispersive" structure: downstream negative curvature directions are the image of upstream negative curvature directions under gradient flow. This means that negative curvature upstream helps to prevent local gradient descent methods from stagnating near downstream saddle points.

The above phenomena are exactly opposite to the worst-case scenarios in which gradient descent may take exponential time to escape saddle points. For instance, the work [DJL+17] has constructed the so-called "octopus" function whose upstream unstable manifold is channeled into stable manifolds of downstream saddle points. As we see here natural nonconvex programs associated with low-dimensional structures are far from such worst case scenarios. In the following subsections, we will see how these basic phenomena recur in more practical nonconvex problems with discrete symmetries, including general dictionary learning (Section 7.3.2), blind deconvolution (Section 7.3.3), and others.

### 7.3.2 Dictionary Learning

The one-sparse dictionary learning problem is an extreme simplification of basic modern data processing problem: seeking a concise representation of data. The goal of dictionary learning is to produce a sparse model for an observed dataset $\boldsymbol{Y} = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_p] \in \mathbb{R}^{m\times p}$. Namely, we seek matrices $\boldsymbol{A}_o \in \mathbb{R}^{m\times n}$ and $\boldsymbol{X}_o \in \mathbb{R}^{n\times p}$ such that

$$\boldsymbol{Y} \approx \underbrace{\boldsymbol{A}_o}_{\text{dictionary}} \quad \underbrace{\boldsymbol{X}_o}_{\text{sparse coefficients}} \tag{7.3.11}$$

with $\boldsymbol{X}_o$ as sparse as possible. Sparsity is desirable for data compression, and to facilitate tasks such as sensing, denoising, super-resolution, etc. [WMM+10, Ela10]

In the representation (7.3.11), the data points $\boldsymbol{y}_j$ are approximated as superpositions $\boldsymbol{y}_j \approx \boldsymbol{A}_o \boldsymbol{x}_{oj}$ of a few columns of the matrix $\boldsymbol{A}_o = [\boldsymbol{a}_{01}, \dots, \boldsymbol{a}_{0n}]$.

This matrix is sometimes called a *dictionary*. Clearly, the size of the dictionary, $n$, has an impact on the accuracy, sparsity, and utility of this data representation. The appropriate dictionary size depends on application: for learning from a single image, a complete ($n = m$) dictionary may suffice, whereas for learning from larger collections of images, an overcomplete ($n > m$) dictionary may be more appropriate [MKD06, EA06, YWHM10]. Below, we discuss how our basic intuitions from the orthogonal, one-sparse case, carry over to each of these more realistic model problems.

*Complete Dictionary Learning.*
Let us first consider the complete case $n = m$, in which $\boldsymbol{A}_o \in \mathbb{R}^{n\times n}$ is a square invertible matrix. There are two basic issues in moving from one-sparse dictionary learning problem to more general complete dictionary learning problems. First, the target dictionary $\boldsymbol{A}_o$ may not be orthogonal. Second, the columns of the coefficient matrix $\boldsymbol{X}_o$ are generally not one-sparse. For theoretical purposes, both of these issues can be addressed using probabilistic properties of $\boldsymbol{X}_o$. First, using the statistics of $\boldsymbol{Y} = \boldsymbol{A}_o\boldsymbol{X}_o$ it is possible to reduce the problem of learning a general invertible $\boldsymbol{A}_o \in \mathsf{GL}(n)$ to one of learning an orthogonal matrix $\bar{\boldsymbol{A}} = (\boldsymbol{A}_o\boldsymbol{A}_o^*)^{-1/2}\boldsymbol{A}_o$. Concretely, if $\boldsymbol{X}_o$ is a sparse random matrix with independent symmetric entries,

$$\bar{\boldsymbol{Y}} = (\boldsymbol{Y}\boldsymbol{Y}^*)^{-1/2}\boldsymbol{Y} \;\propto\; \bar{\boldsymbol{A}}\boldsymbol{X}_o$$

satisfies a sparse model with orthogonal dictionary $\bar{\boldsymbol{A}} \in \mathsf{O}(n)$.

Similar to our discussion above, one can recover the columns of $\boldsymbol{A}$ by solving the optimization problem for a sparsity-promoting function $h$:

$$\min\ \varphi(\boldsymbol{a}) \equiv h\left(\boldsymbol{a}^*\bar{\boldsymbol{Y}}\right) \quad \text{such that} \quad \boldsymbol{a} \in \mathbb{S}^{n-1}. \tag{7.3.12}$$

This is essentially to find a sparse vector $\boldsymbol{a}^*\bar{\boldsymbol{Y}}$ in the row space of $\boldsymbol{X}_o$. If we repeat this process $m$ times, we in principle can recover all the $n$ sparse rows of $\boldsymbol{X}_o$. Although the columns of $\boldsymbol{X}_o$ are not one-sparse, when the number samples is large, this objective function retains all of the qualitative properties observed in the one-sparse problem, including local minimizers near symmetric solutions and saddle points near balanced superpositions of symmetric solutions, with negative curvature in symmetry breaking directions. The proofs of these properties rely heavily on probabilistic reasoning: one argues that the "population" objective function $\mathbb{E}[\varphi]$ has benign structure, and then argues that when the number $p$ of samples is large, gradients and Hessians of $\varphi$ are uniformly close to those of $\mathbb{E}[\varphi]$, and hence $\varphi$ has the same benign properties [SQW17a, SQW17b].

In early chapters, we have studied $\ell^1$ norm extensively as it is the (unique) convex envelope of the sparse $\ell^0$ norm. Nevertheless, once we consider nonconvex surrogates, there are many more choices of sparse promoting functions. Some can be extremely effective when the optimization domain is confined to a structured space such as the sphere. For example, it is easy to show that the maximizers of

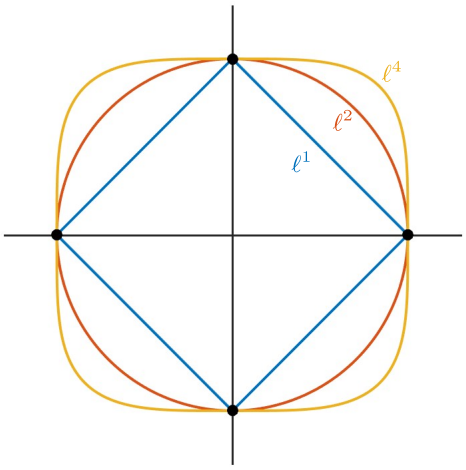



**Figure 7.10** Illustration of the $\ell^1$-ball, $\ell^2$-ball, and $\ell^4$-ball in $\mathbb{R}^2$.

$\ell^4$ norm of a vector $\boldsymbol{x} \in \mathbb{R}^n$ over the sphere $\mathbb{S}^{n-1}$ are equivalent to minimizers of $\ell^0$ norm over the sphere:

$$\operatorname*{argmax}_{\boldsymbol{x}\in\mathbb{S}^{n-1}} \|\boldsymbol{x}\|_4 \quad = \quad \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{S}^{n-1}} \|\boldsymbol{x}\|_0 \,. \tag{7.3.13}$$

Figure 7.10 illustrates the relationships of $\ell^1, \ell^2$ and $\ell^4$ balls. Notice that for points on the sphere (the $\ell^2$ ball), points that minimize $\ell^1$ norm coincide with those maximize $\ell^4$ norm.

Hence given $\bar{\boldsymbol{Y}}$ in order to find the orthogonal dictionary $\bar{\boldsymbol{A}}$, we may consider solving the following (nonconvex) $\ell^4$ norm maximization problem over the orthogonal group $\mathsf{O}(n)$:

$$\max \ \|\boldsymbol{A}^*\bar{\boldsymbol{Y}}\|_4^4 \quad \text{subject to} \quad \boldsymbol{A}^* \in \mathsf{O}(n). \tag{7.3.14}$$

It has been shown that with sufficient samples, say $p$ in the order of $O(n^2 \log n)$, the global maximizers of the above program are the correct dictionary [ZYL+20]. The overall landscape of is rather benign and leads to a very efficient power-iteration like algorithm [ZYL+20, ZMZM20].[27] We will leave the algorithmic details as an exercise for the readers and study it more in Chapter 9.

[27] The algorithm has been shown to converge superlinearly locally and overwhelming empirical evidences show it always converges to the globally optimal solution. However, a rigorous proof of its global optimality remains an open problem [ZYL+20]. The authors of the book is offering a thousand dollars for anyone who could provide such a proof.

*Overcomplete Dictionary Learning.*
In practice, *overcomplete* dictionaries, in which the number of dictionary atoms $n$ is larger than the signal dimension $m$, are often favored compared to complete dictionaries. Overcomplete dictionaries have greater expressive power, yielding sparser coefficient matrices $\boldsymbol{X}$. Our current theoretical understanding of the objective landscape associated with overcomplete dictionary learning is still developing. One suggestive result shows that when the dictionary is moderately overcomplete ($n \leq 3m$), under appropriate technical hypotheses, a formulation based on maximizing the $\ell^4$ norm exhibits benign global geometry [QZL+19]: again, every local minimizer is global and saddle points exhibit strict negative curvature.[28] These results suggest that overcomplete dictionary learning problems can exhibit benign global geometry; there are a number of open questions around

1 the degree of overcompleteness $n/m$ that this structure can tolerate and
2 the extent to which similar properties hold in more conventional *synthesis* dictionary learning formulations, in which one optimizes over both $\boldsymbol{A}$ and $\boldsymbol{X}$ simultaneously.

### 7.3.3 Sparse Blind Deconvolution

Convolutional models arise in a wide range of problems in imaging and data analysis. The most basic convolutional data model expresses an observation $\boldsymbol{y}$ as the convolution of two signals $\boldsymbol{a}_o$ and $\boldsymbol{x}_o$. *Blind deconvolution* aims to recover $\boldsymbol{a}_o$ and $\boldsymbol{x}_o$ from the observation $\boldsymbol{y} = \boldsymbol{a}_o \circledast \boldsymbol{x}_o$, up to certain intrinsic symmetries that we describe below. This problem is ill-posed in general – there are infinitely many $(\boldsymbol{a}_o, \boldsymbol{x}_o)$ that convolve to produce $\boldsymbol{y}$. To make progress, some low dimensional priors about $\boldsymbol{a}_o$ and $\boldsymbol{x}_o$ are essential. Different priors yield different nonconvex optimization problems; in this section, we will focus on several variants of blind deconvolution with sparsity priors on $\boldsymbol{x}_o$, and then briefly mention other popular variants of blind deconvolution.

#### Short and Sparse (SaS) Blind Deconvolution

Analyzing signals comprised of repeated motifs is a common task in areas such as neuroscience, materials science, astronomy, and natural and scientific imaging [SPM02, PSG+16, CSL+20, LQK+19]. Such signals can be modeled as the *convolution* of a short motif $\boldsymbol{a}_o$ and a sparse coefficient signal $\boldsymbol{x}_o$, which encodes where the motif occurs in time/space. Mathematically, the observation $\boldsymbol{y} \in \mathbb{R}^m$

[28] When the dictionary is overcomplete, dictionary atoms $\boldsymbol{a}_i$ are correlated and $\boldsymbol{a}^* \bar{\boldsymbol{Y}}$ is no longer sparse, even if $\boldsymbol{a}$ is chosen as one of the atoms $\boldsymbol{a}_i$. Rather, at $\boldsymbol{a} = \boldsymbol{a}_i$, $\boldsymbol{a}^* \bar{\boldsymbol{Y}}$ is *spiky*, with a few large entries amongst many small ones. $\ell^4$ maximization is well-suited to encouraging this kind of spikiness. The most widely used practical dictionary learning algorithms are based on synthesis sparsity. Understanding the global geometry of this kind of formulation remains an important open problem

is the windowed[29] convolution of the short $\boldsymbol{a}_o$, which is supported on $k$ ($k \ll m$) consecutive entries and the sparse $\boldsymbol{x}_o$:

$$\boldsymbol{y} = \mathcal{P}_m \left[ \boldsymbol{a}_o \circledast \boldsymbol{x}_o \right]. \tag{7.3.15}$$

Here, $\circledast$ denotes linear convolution and $\mathcal{P}_m(\cdot)$ retains the entries supported on indices $0, \ldots, m-1$.

The inverse problem of recovering $\boldsymbol{a}_o$ and $\boldsymbol{x}_o$ from $\boldsymbol{y}$ is called *short and sparse* blind deconvolution (SaS-BD) [ZLK+17, ZKW18, KZLW19]. The linear convolution $\circledast$ exhibits a *signed shift symmetry*:

$$\boldsymbol{a}_o \circledast \boldsymbol{x}_o = \alpha s_\tau[\boldsymbol{a}_o] \circledast \alpha^{-1} s_{-\tau}[\boldsymbol{x}_o]. \tag{7.3.16}$$

Here $\alpha$ is some nonzero scalar and $s_\tau[\boldsymbol{v}]$ denotes a shift of vector $\boldsymbol{v}$ by $\tau$ entries, i.e. $s_\tau[\boldsymbol{v}](i) = \boldsymbol{v}(i-\tau)$. As with the other nonconvex problems we have studied up to this point, we should expect this symmetry to play an critical role in shaping the landscape of optimization – in particular, we would expect the *global* minimizers to be symmetric copies of the ground truth.[30]

*Symmetry Breaking?*

However, there is a wrinkle: in order to obtain a finite dimensional optimization problem, one typically constrains the length-$k$ signal $\boldsymbol{a}_o$ to be supported on $\{0, \ldots, k-1\}$. This constraint appears to remove the shift symmetry: now only a scaled version $(\alpha \boldsymbol{a}_o, \alpha^{-1} \boldsymbol{x}_o)$ of the truth exactly reproduces the observation. Perhaps surprisingly, even with this constraint, symmetry *still* shapes the landscape of optimization. However, instead of dictating the global minimizers, in constrained formulations, symmetry dictates the *local* minimizers. The reason is simple: a shift of $\boldsymbol{a}_o$ by $\tau$ samples is not supported on $\{0, \ldots, k-1\}$, and hence is not feasible. However, its truncation to $\{0, \ldots, k-1\}$ *is* feasible, and still approximates $\boldsymbol{y}$:

$$\boldsymbol{y} \approx \mathcal{P}_k \left[ s_\tau[\boldsymbol{a}_o] \right] \circledast s_{-\tau}[\boldsymbol{x}_o]. \tag{7.3.17}$$

Because this approximation is not perfect, truncated shifts are not global minimizers. However, they are very close to *local* minimizers [ZLK+17, ZKW18]. These points have suboptimal objective value and do not exactly reproduce $(\boldsymbol{a}_o, \boldsymbol{x}_o)$. Despite this, the optimization landscape is still sufficiently benign[31] that it is possible to exactly recover $(\boldsymbol{a}_o, \boldsymbol{x}_o)$ with efficient methods – one can, e.g., first find a local minimizer that is close to a truncated shift of $\boldsymbol{a}_o$, and then refine it to exactly recover $\boldsymbol{a}_o$.

[29] Rather than having complete access to the convolved signal (which could be infinitely long), we observe $m$ consecutive entries of it.

[30] Notice that the scale and shift symmetries are intrinsic to the convolution operator in (7.3.15). Although we focus on *sparse* deconvolution, these symmetries will persist in deconvolution with any shift-invariant structural model for $\boldsymbol{a}_o$ and $\boldsymbol{x}_o$. Moreover, as we will see below, they persist even in the presence of artificial symmetry-breaking mechanisms, in the sense that they still dictate the local minimizers.

[31] In particular, there is negative curvature in symmetry breaking directions.

This problem illustrates how hard it is to avoid symmetry in studying deconvolution problems: even with an explicit symmetry breaking constraint, symmetry still shapes the landscape of optimization! The main motivation for studying this more complicated deconvolution model is its applicability (as we will see in Chapter 12 for an application in scientific imaging). Giving formulations that better respect the symmetry structure, and hence have no spurious local minimizers, remains an important open problem.

### Multi-Channel Sparse (MCS) Blind Deconvolution

The problem of *multi-channel sparse blind deconvolution* assumes access to multiple observations $\boldsymbol{y}_i = \boldsymbol{a}_o \circledast \boldsymbol{x}_i \in \mathbb{R}^k$ generated from circular convolution (also denoted by $\circledast$) of $\boldsymbol{a}_o \in \mathbb{R}^k$ and distinct sparse signals $\boldsymbol{x}_i$ [LB18, QLZ19, QZL+19, SC19]. Here, shift symmetry becomes a *cyclic* shift symmetry: there exist $k$ equivalent solutions corresponding to $k$ different cyclic shifts. The resulting optimization landscape exhibits similar characteristics to that of complete dictionary learning, described in Section 7.3.1 and Figure 7.9. In particular, any local minimizer is a scaled cyclic shift of the ground truth [LB18, QLZ19, SC19].

### Geometry of Sparse Blind Deconvolution

Despite the technical difference of the convolution operator in MCS and SaS blind deconvolution problems, their optimization landscapes share the following key phenomena:

- ***Symmetric copies of the ground truth are minimizers.*** In above two variants of sparse blind deconvolution problems, the local minimizers are either a cyclic shifted or shifted truncation of the ground truth under conditions. Both can be viewed as a result of the inherent shift symmetry.
- ***Negative curvature in symmetry breaking directions.*** Near saddle points, there is negative curvature in the direction of any particular (truncated) shifted copy of the ground truth, and the objective value decreases by moving towards this symmetry breaking direction.
- ***Cascade of saddle points.*** The saddle points are approximately balanced superpositions of several shifts of the ground truth. The more shifts participate, the larger the objective value and the more negative eigenvalues the Hessian exhibits.

### Other Blind Deconvolution Variants

*Subspace blind deconvolution* is another widely studied variant of blind deconvolution that leverages a low dimensional model for the pair $(\boldsymbol{a}_o, \boldsymbol{x}_o)$. In this variant, $\boldsymbol{a}_o$ and $\boldsymbol{x}_o$ are assumed to lie on known low-dimensional subspaces [ARR14]. This problem can be cast as a rank-one matrix recovery problem, which exhibits a similar geometry to the problems studied in Section 7.2.

*Convolutional dictionary learning* extends the basic convolution model by allowing for multiple basic motifs $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N$ [GCW18]. More precisely, we observe

one or more signals of the form $\boldsymbol{y} = \sum_{i=1}^{N} \boldsymbol{a}_i \circledast \boldsymbol{x}_i$, and the goal is to recover all the $\boldsymbol{a}_i$ and $\boldsymbol{x}_i$. In addition to the symmetries inherited from the convolution operator, this problem processes an additional *permutation symmetry*: permuting the index $i$ does not change the approximation to $\boldsymbol{y}$. Despite this additional complexity, empirically local minimizers remain symmetric copies of the ground truth [ZLK+17,LQK+19]; under certain technical hypotheses, one can prove that natural first order algorithms always recover one such symmetric copy [QZL+19].

In fact, one may model natural images to be (sparsely) generated by such a convolutional dictionary. In some applications, it may not be necessary to recover the dictionary $\{\boldsymbol{a}_i\}$ and sparse codes $\{\boldsymbol{x}_i\}$ precisely. For example, we only want to classify similar images into the same category. But the assumption of such a model is crucial for obtaining an (approximately) correct solution, say via a deep network. We will discuss the connection of this type of models to deep (convolutional) networks in Chapter 16.

### 7.3.4 Other Nonconvex Problems with Discrete Symmetry

*Symmetric Tensor Decomposition.*
Tensors can be regarded as high dimensional generalization of matrices. Tensor decomposition problems find many applications in statistics, data science, and machine learning [KB09,AGH+14,SDLF+17,JGKA19]. Although we can usually generalize algebraic notions from matrices to tensors, their counterpart in tensors are often not as well-behaved or easy to compute [KB09]. In fact, many natural tensor problems are NP-hard in the worst case [HL13].

Nonetheless, recent results suggest that certain appealing special cases of tensor decomposition are tractable [AGH+14,GHJY15,JGKA19]. This is especially true for orthogonal tensor decomposition, where the task is to decompose a $p$-th order symmetric tensor into this orthogonal components. More specifically, an orthogonal tensor $\mathcal{T}$ can be presented in the following form

$$\mathcal{T} = \sum_{k=1}^{r} \boldsymbol{a}_k^{\otimes p}, \quad r \le n, \tag{7.3.18}$$

with $\{\boldsymbol{a}_k\}_{k=1}^r$ are a collection of orthogonal vectors, and $\boldsymbol{a}^{\otimes p}$ denotes the $p$-way outer product of a vector $\boldsymbol{a}$. The orthogonal tensor decomposition shares many similarities with the other nonconvex problems with discrete symmetry discussed above:

- the problem exhibits a *signed permutation symmetry* which is similar to dictionary learning: given $\mathcal{T}$ we can only hope to recover the orthogonal components $\{\boldsymbol{a}_k\}_{k=1}^r$ up to order permutation;
- when $p$ is even order, as shown in Figure 7.5, a natural nonconvex formulation

$$\min_{\boldsymbol{x} \in \mathbb{S}^{n-1}} -\mathcal{T}(\boldsymbol{x}, \ldots, \boldsymbol{x}) = -\|\boldsymbol{A}^* \boldsymbol{x}\|_p^p \quad \text{with} \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_1 & \cdots & \boldsymbol{a}_r \end{bmatrix} \tag{7.3.19}$$

manifests a similar optimization landscape, for which every local minimizer

is close to one of the signed orthogonal components and other critical points exhibit strict negative curvature.

These results have inspired further endeavors beyond orthogonal tensors [QZL+19, SBRL19, GM17]. One particular case of interest is decomposing a symmetric tensor $\mathcal{T}$ in (7.3.18) with $r > n$ and nonorthogonal $\{\boldsymbol{a}_k\}_{k=1}^r$, which is often referred as *overcomplete* tensor decomposition. In particular, when $p = 4$, $r \in O(n^{1.5})$ and $\{\boldsymbol{a}_k\}_{k=1}^r$ are i.i.d. Gaussian, [GM17] shows that (7.3.19) has no bad local minimizer over a level set whose measure geometrically shrinks w.r.t. the problem dimension; for $p = 4$, $r < 3n$, and incoherent $\{\boldsymbol{a}_k\}_{k=1}^r$, [QZL+19] presented a global analysis for overcomplete tensor decomposition, disclosing its connection to overcomplete dictionary learning. Nonetheless, these results are still far from providing a complete understanding of overcomplete tensor decomposition. One interesting question remains largely open is when bad local minimizers exist for large rank $r \gg n$ in the nonorthogonal case.

*Clustering.*
Clustering is arguably the most fundamental problem in unsupervised learning. This problem possesses a *permutation symmetry*: one can generate equivalent clusters by permuting the indices for cluster centers. Popular nonconvex algorithms include the Lloyd algorithm and variants of Expectation Maximization. Despite the broad applications and empirical success of these methods, few theoretical guarantees have been obtained until recently. The problem of demixing two balanced, identical data clusters manifests global convergence to (a symmetric copy of) the ground truth [BWY17, XHM16, DTZ16, QZC19, KQC+19]. We see similar geometric properties hold here: *symmetric copies of the ground truth are minimizers* and *saddle points exhibit directions of strict negative curvature*. Moreover, the saddle points are also located at balanced superpositions of local minimizers. Sometimes, these saddle points may contain redundant cluster estimates. In this case, the redundant cluster estimates can be interpreted as a under-parametrized solution (with a smaller $k$ specified).

However, in general clustering problems with more than two clusters, local minimizers provably exist [DS07, JZB+16]. When the clusters are sufficiently separated, these local minimizers possess characteristic structures [QZC20]: they correspond imbalanced segmentations of the data, in which a subset of the true clusters are optimally under-segmented and another subset is optimally over-segmented.

*Deep Neural Networks.*
Deep neural networks have more complicated symmetry groups than the problems described above. For example, natural objective functions associated with fitting a fully connected neural network are invariant under simultaneous permutations of the features at *each* layer. We currently lack tools for reasoning about the global geometry of such problems. However, progress has been made on certain special cases: for example, certain problems associated with fitting shallow

networks share similar geometry to tensor decomposition [JSA15, MM18]. With varying technical assumptions, all local solutions have been shown to be global in a 1-layer neural network [HV17, FJZT17, GLM17, GMOV18, SJL18]. However, general deep nonlinear neural networks can exhibit flat saddles and spurious local minimizers [SS17, VBGS17]. We refer interested readers to [Sun19b] for more comprehensive development on optimization theory and algorithm of deep learning.

In Chapter 16, we will study deep learning from the perspective of learning discriminative low-dimensional representations. We will see how data clustering and representation learning can be naturally unified in a nonconvex objective function that inherits the rich symmetric structures of both deep networks and data clustering.

*Fourier Phase Retrieval.*
The problem of *Fourier* phase retrieval is crucial to scientific imaging. In this problem, the goal is to recover $\boldsymbol{x}_o$ from observation $\boldsymbol{y} = |\mathcal{F}(\boldsymbol{x}_o)|$. Apart from the rotational (phase) symmetry, the problem of Fourier phase retrieval manifests two additional symmetries [32]: *(cyclic)-shift symmetry* $|\mathcal{F}(\boldsymbol{x})| = |\mathcal{F}(s_\tau[\boldsymbol{x}])|$ and *conjugate inversion symmetry* $|\mathcal{F}(\boldsymbol{x})| = |\mathcal{F}(\bar{\boldsymbol{x}})|$, where $\bar{\boldsymbol{x}}(n) = \bar{\boldsymbol{x}}(-n)$ [BBE17]. This complicated symmetry structure is reflected in a complicated optimization landscape, which is challenging to study analytically. Many basic problems in the algorithmic theory of Fourier phase retrieval remain open.

## 7.4 Notes and Open Problems

In this chapter, we have reviewed recent advances in provable nonconvex methods for signal processing and machine learning, through the lens of symmetry. It is an exciting time to work on both the theory and practice of nonconvex optimization. For complementary perspectives on the area, we refer interested readers to other recent review papers [JK+17, Sun19a, CLC19, QZL+20b]. In the following, we close by discussing several methodological points and general directions for future work.

*Convexification.*
In the past decades, convex relaxation has been demonstrated a powerful tool for solving nonconvex problems such as sparse recovery (Chapters 2–3), low-rank matrix completion (Chapters 4–5), and even more general atomic structures (Chapter 6). For these problems, convex relaxation achieves near-optimal sample complexity. Which nonconvex problems are amenable to convex relaxation? There are general results that suggest that *unimodal* functions (i.e., functions

[32] When $\boldsymbol{x}$ is one dimensional, the problem becomes even more pessimistic — there exist multiple one dimensional signals with the same Fourier magnitude, but not related by an obvious symmetry.

with one local minimizer) on convex sets can be convexified, by endowing the space with an appropriate geometry [RC93].[33] The symmetric problems encountered in this survey are not unimodal. The degree to which they are amenable to convex relaxation varies substantially:

- *Problems with rotational symmetry.* Many problems with rotational symmetry *can* be convexified by lifting to a higher dimensional space [CR09, CLMW11, CSV13], e.g., by replacing the factor $\boldsymbol{U}$ with a matrix valued variable $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{U}^*$. This collapses the $\mathsf{O}(r)$ symmetry; the resulting problems can often be converted to semidefinite programs and solved globally. Typically, nonconvex formulations are still preferred in practice, due to their scalability to large datasets. Section 7.2 and the references therein describe alternative geometric principles that help to explain the success of these methods.
- *Problems with discrete symmetry.* Most of the discrete symmetric problems described in Section 7.3 do not admit simple convex relaxations. For example, complete dictionary learning can be reduced to a sequence of linear programs [SWW12], but only in the highly sparse case, in which the target sparse representation has $O(\sqrt{n})$ nonzero entries per length-$n$ data vector. This limitations are attributable in part to the more complicated discrete symmetry structure. Natural ideas, such as taking a quotient by the symmetry group, encounter obstacles at both the conceptual and implementation levels. One general methodology which *does* meet with success in this setting is sum-of-squares relaxation, which for variants of dictionary learning and tensor decomposition leads to quasipolynomial or even polynomial time algorithms [BKS15].

*Efficient First-Order Algorithms.*
In this chapter, we have described families of symmetric nonconvex optimization problems with benign global geometry: local minimizers are global and saddle points exhibit strict negative curvature. Although we have not emphasized algorithmic aspects of these problems, this geometric structure *does* have strong implications for computation – a variety of methods the key is leveraging negative curvature to efficiently obtain minimizers. We will provide a systematic introduction to nonconvex optimization algorithms and their convergence and complexity properties in Chapter 9.

One class of methods explicitly models negative curvature, e.g., using a second order approximation to the objective function. Methods in this class include trust region methods [CGT00], cubic regularization [NP06], and curvilinear search [Gol80]. These methods can be challenging to scale to very large problems, since they typically require computation and storage of the Hessian. It is also possible to leverage negative curvature using more scalable first-order

[33] These are existence results; their direct implications for efficient computation are limited, since they apply to NP-hard problems. It is also worth noting that many of our discrete symmetric problems in Section 7.3 are formulated over compact manifolds such as $\mathbb{S}^{n-1}$; the only continuous geodesically convex function on a compact Riemannian manifold is a constant [BO69, Yau74].

methods such as gradient descent. In the vicinity of a saddle point, the gradient method essentially performs a power iteration which moves in directions of negative curvature. Although this scheme *can* stagnate at or near saddle points, it is possible to guarantee efficient escape by perturbing the iterates with an appropriate amount of random noise [GHJY15, JGN+17, JNJ18, CB19, SFF19].

The methods described above are efficient across the broad class of *strict saddle functions* [GHJY15, SQW15], i.e., functions whose saddle points all have directions of strict negative curvature. This is a worst case performance guarantee. Perhaps surprisingly, the most widely used first order method, gradient descent, is not efficient for worst case strict saddle functions: although randomly initialized gradient descent *does* obtain a minimizer with probability one [LSJR16, LPP+19], for certain functions it can take time exponential in dimension [DJL+17]. These challenging functions have a large numbers of saddle points, which are conspicuously arranged such that upstream negative curvature directions align with *positive* curvature directions for downstream saddle points.

This worst case behavior is in some sense the opposite of what is observed in the type of highly symmetric functions studied here: functions encountered in generalized phase retrieval [CCFM18], dictionary learning [GBW19], deconvolution [QLZ19, SC19], etc., exhibit a global negative curvature structure, in which upstream negative curvature directions align with *negative* curvature directions of downstream saddle points. In this situation, *randomly initialized gradient descent is efficient*. This points to another gap between naturally occurring nonconvex optimization problems and their worst case counterparts. There is substantial room for future work in this direction.

*Disciplined Formulations and Analysis.*
Our understanding of nonconvex optimization is still far from satisfactory – analyses are delicate, case-by-case, and pertain to problems with elementary symmetry (e.g., rotation or permutation) and simple constraints (e.g., the sphere or simple homogeneous spaces).

- *A Unified Theory.* Analogous to the study of convex functions [BV04], there is a pressing need for simpler analytic tools, to identify and generalize benign properties for new nonconvex problems, despite some recent endeavors [QZL+19, LCD+19] of identifying general conditions and operations preserving benign geometric structures. Unlike the convex case in which convex surrogates are typically unique, one can have multiple nonconvex surrogates for the same problem. For instance, to promote low-rankness of a matrix, one could choose to use the log det function [FHB03], random dropout in training deep neural networks [SHK+14], or over-parameterization with matrix products (see the exercises). Nevertheless, as we will see, those surrogates are fundamentally related to the convex surrogates (such as the nuclear norm) and yet offer other benefits such as simpler implementation or broader range of working conditions.

- *Complicated Symmetries and Constraints.* Practical nonconvex problems often involve *multiple symmetries* (e.g., Fourier phase retrieval and deep neural networks) and/or *complicated manifolds* (e.g., Stefiel manifolds [HLWY19]). We need better technical tools to understand the impact of compound symmetries (especially compound discrete symmetries) on the optimization landscape, despite some steps in this directions [LCD+19, HLWY19, ZYL+20]. More interesting and challenging phenomena arise when the symmetry of the problem and manifold/group structure of the domain are intertwined. For instance, in dictionary learning via $\ell^4$ maximization, we have both the signed permutation symmetry $\mathsf{SP}(n)$ and the orthogonal group $\mathsf{O}(n)$. In Section 9.6 of Chapter 9, we will see power-iteration or fixed-point type algorithms are very natural and effective in exploiting such manifold structures. However, unified analyses and understandings for broader problem classes are still lacking.
- *Nonsmoothness.* In many scenarios we encounter nonconvex problems with *nonsmooth* functions [DDMP18, DD18, LZMCSV20, LCD+19, BJS19, ZWR+18, CDDD19, CCD+19], for better promoting solution sparsity or robustness. As we will see in Chapter 8, in the convex setting, nonsmoothness usually can be dealt very effectively. However, in the nonconvex setting, most of our current analysis is local [CDDD19, LCD+19], and (subgradient) optimization [LCD+19, BJS19, ZWR+18] could be slow to converge. Attempts to obtain global analyses and fast optimization methods might benefit from more sophisticated tools from variational analysis [RW09] and development of efficient second-order or higher-order methods [DR19].

## 7.5 Exercises

7.1. *In this section, we study how to derive the Huber function given in* (7.2.24). *First, find a closed-form solution to the problem:*

$$x_\star(u) = \arg\min_x \tfrac{1}{2}(u-x)^2 + \lambda|x|.$$

*Then, show that the function defined below:*

$$h_\lambda(u) \doteq \min_x \tfrac{1}{2}(u-x)^2 + \lambda|x| = \tfrac{1}{2}(u-x_\star)^2 + \lambda|x_\star|$$

*has the same form as the Huber function* (7.2.24).

7.2 (Complete Dictionary Learning via $\ell^4$ Norm Maximization). *In this exercise, we derive and practice an algorithm to solve $\ell^4$ norm maximization problem* (7.3.14) *for complete dictionary learning.*

*1 Derive the gradient $\varphi(\boldsymbol{A}^*) = \|\boldsymbol{A}^*\bar{\boldsymbol{Y}}\|_4^4$ with respective to $\boldsymbol{A}^*$.*
*2 Derive a projected gradient ascent algorithm for maximizing $\varphi(\boldsymbol{A}^*)$:*

$$\boldsymbol{A}^*_{k+1} = \mathcal{P}_{\mathsf{O}(n)}[\boldsymbol{A}^*_k + \gamma \cdot \nabla\varphi(\boldsymbol{A}^*_k)].$$

*3 Conduct simulation of the algorithm and play with different step size $\gamma$ of the gradient ascent. What happens if you make the step size to be infinite? That is,*

$$\boldsymbol{A}_{k+1}^* = \mathcal{P}_{\mathsf{O}(n)}[\nabla\varphi(\boldsymbol{A}_k^*)].$$

7.3 (Sparsity Regularization via Over-parameterization and Gradient Descent). *Given a vector $\boldsymbol{y} \in \mathbb{R}^m$ and a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, consider the optimization problem*

$$\min_{\{\boldsymbol{u},\boldsymbol{v}\}\subseteq\mathbb{R}^m} f(\boldsymbol{u},\boldsymbol{v}) \doteq \frac{1}{4}\|\boldsymbol{y} - \boldsymbol{A}(\boldsymbol{u}\odot\boldsymbol{u} - \boldsymbol{v}\odot\boldsymbol{v})\|_2^2, \tag{7.5.1}$$

*where $\odot$ denotes the Hadamard (i.e., entry-wise) product between two vectors. Let $(\boldsymbol{u}_t(\gamma), \boldsymbol{v}_t(\gamma))$ be given by the gradient flow dynamics (i.e., gradient descent with infinitesimally small step size) of (7.5.1):*

$$\begin{cases} \dot{\boldsymbol{u}}_t(\gamma) &= -\nabla f\left(\boldsymbol{u}_t(\gamma), \boldsymbol{v}_t(\gamma)\right) = -\left(\boldsymbol{A}^*\boldsymbol{r}_t(\gamma)\right)\odot\boldsymbol{u}_t(\gamma), \\ \dot{\boldsymbol{v}}_t(\gamma) &= -\nabla f\left(\boldsymbol{u}_t(\gamma), \boldsymbol{v}_t(\gamma)\right) = \left(\boldsymbol{A}^*\boldsymbol{r}_t(\gamma)\right)\odot\boldsymbol{v}_t(\gamma), \end{cases} \tag{7.5.2}$$

*with the initial condition $\boldsymbol{u}_o(\gamma) = \boldsymbol{v}_o(\gamma) = \gamma\cdot\boldsymbol{1}$ (i.e., a vector with all entries being $\gamma$), and $\boldsymbol{r}_t(\gamma) \doteq \boldsymbol{A}\left(\boldsymbol{u}_t(\gamma)\odot\boldsymbol{u}_t(\gamma) - \boldsymbol{v}_t(\gamma)\odot\boldsymbol{v}_t(\gamma)\right) - \boldsymbol{y}$. Let*

$$\boldsymbol{x}_t(\gamma) = \boldsymbol{u}_t(\gamma)\odot\boldsymbol{u}_t(\gamma) - \boldsymbol{v}_t(\gamma)\odot\boldsymbol{v}_t(\gamma), \tag{7.5.3}$$

*and assume that the following conditions hold:*

- *the limit $\boldsymbol{x}_\infty(\gamma) := \lim_{t\to\infty}\boldsymbol{x}_t(\gamma)$ exists and satisfies $\boldsymbol{A}\boldsymbol{x}_\infty(\gamma) = \boldsymbol{y}$ for all $\gamma$;*
- *the limit $\boldsymbol{x}_\infty := \lim_{\gamma\to 0}\boldsymbol{x}_\infty(\gamma)$ exists.*

*Then, show that $\boldsymbol{x}_\infty$ is a global solution to the following optimization problem*

$$\min_{\boldsymbol{x}}\|\boldsymbol{x}\|_1 \quad \textit{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}. \tag{7.5.4}$$

*(Hint: Note that from Chapter 3, the conclusion holds if and only if there exists a $\boldsymbol{\lambda}\in\mathbb{R}^m$, a dual certificate, such that the condition $\boldsymbol{A}^\top\boldsymbol{\lambda}\in\partial\|\boldsymbol{x}_\infty\|_1$ holds. Then, show that $\boldsymbol{\lambda} = \lim_{\gamma\to 0}\frac{-\lim_{t\to\infty}\int_0^t \boldsymbol{r}_\tau(\gamma)d\tau}{\log(1/\gamma)}$ provides such a dual certificate.)*

*Conceptually, this phenomenon is the same as the one we have seen in Exercise 2.10 of Chapter 2: The gradient descent with proper initialization introduces implicit bias on which solution (among all infinitely-many optimal solutions) it eventually converges to.*

7.4 (Low-rank Regularization via the $\log\det(\cdot)$ Function). *When a matrix $\boldsymbol{X}\in\mathbb{R}^{n\times n}$ is symmetric and positive semi-definite, the nuclear norm $\|\boldsymbol{X}\|_*$ is the same as its trace of the matrix. In this exercise, we try to study the connection of the convex nuclear norm (or the trace norm) with another popular smooth but*

*nonconvex surrogate for minimizing* $\operatorname{rank}(\boldsymbol{X})$ *is to minimize the quantity*[34]

$$\min_{\boldsymbol{X}\in\mathsf{C}} f(\boldsymbol{X}) \doteq \log\det(\boldsymbol{X}+\delta\boldsymbol{I}), \tag{7.5.5}$$

*where* $\delta > 0$ *is a small regularization constant and* $\boldsymbol{X}$ *belongs to some constraint set* $\mathsf{C}$. *To see how this objective is related to the trace norm:*

*1 First, show that* $\nabla_{\boldsymbol{X}} f(\boldsymbol{X}) = (\boldsymbol{X}+\delta\boldsymbol{I})^{-1}$.
*2 Second, the first-order expansion of* $f(\boldsymbol{X})$ *around a point* $\boldsymbol{X}_k$ *is given by:*

$$f(\boldsymbol{X}) \approx f(\boldsymbol{X}_k) + \operatorname{trace}\left((\boldsymbol{X}_k+\delta\boldsymbol{I})^{-1}(\boldsymbol{X}-\boldsymbol{X}_k)\right) + o(\|\boldsymbol{X}-\boldsymbol{X}_k\|).$$

*Then to minimize* $f(\boldsymbol{X})$, *we can use a greedy descent algorithm with the iteration*

$$\boldsymbol{X}_{k+1} = \arg\min_{\boldsymbol{X}\in\mathsf{C}} \operatorname{trace}\left((\boldsymbol{X}_k+\delta\boldsymbol{I})^{-1}\boldsymbol{X}\right). \tag{7.5.6}$$

*Notice that when* $\boldsymbol{X}_k$ *is initialized around* $\boldsymbol{X}_o = \boldsymbol{I}$, *then the above iteration becomes minimizing the trace norm* $\boldsymbol{X}_{k+1} = \arg\min_{\boldsymbol{X}\in\mathsf{C}} \operatorname{trace}(\boldsymbol{X})$.

7.5 (Low-rank Regularization through Matrix Product). *Given a matrix* $\boldsymbol{Y} \in \mathbb{R}^{m\times n}$, *we may consider to compute a low-rank approximation to it through the proximal operator of the nuclear norm:*

$$\min_{\boldsymbol{X}} \|\boldsymbol{Y}-\boldsymbol{X}\|_2^2 + \lambda\|\boldsymbol{X}\|_*.$$

*Use Proposition 4.6 to show that if we parametrize* $\boldsymbol{X}$ *as matrix product:* $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^* \doteq \sum_k \boldsymbol{u}_k\boldsymbol{v}_k^*$, *then the above convex program is equivalent to the following non-convex program:*

$$\min_{\boldsymbol{U},\boldsymbol{V}} \|\boldsymbol{Y}-\boldsymbol{U}\boldsymbol{V}^*\|_2^2 + \lambda\sum_k \|\boldsymbol{u}_k\|_2\|\boldsymbol{v}_k\|_2. \tag{7.5.7}$$

7.6 (Stochastic Matrix Factorization). *Consider approximate a given matrix* $\boldsymbol{Y} \in \mathbb{R}^{m\times n}$ *by a random superposition of a set of rank-1 factors:*

$$\boldsymbol{Y} \approx \frac{1}{\theta}\sum_{k=1}^{d} r_k\boldsymbol{u}_k\boldsymbol{v}_k^*,$$

*where* $r_k \sim Ber(\theta)$ *are i.i.d Bernoulli variables, and* $\boldsymbol{u}_k$ *are columns from a matrix* $\boldsymbol{U} \in \mathbb{R}^{m\times d}$, *similarly for* $\boldsymbol{v}_k$. *The goal is to minimize the expected error:*

$$\mathbb{E}\left\|\boldsymbol{Y} - \frac{1}{\theta}\boldsymbol{U}\,diag(\boldsymbol{r})\boldsymbol{V}^*\right\|_F^2,$$

*with respect to* $\boldsymbol{r}$, *the vector of all the* $d$ *Bernoulli variables. Show that*

$$\mathbb{E}\left\|\boldsymbol{Y} - \frac{1}{\theta}\boldsymbol{U}\,diag(\boldsymbol{r})\boldsymbol{V}^*\right\|_F^2 = \left\|\boldsymbol{Y}-\boldsymbol{U}\boldsymbol{V}^*\right\|_F^2 + \frac{1-\theta}{\theta}\sum_{k=1}^{d}\|\boldsymbol{u}_k\|_2^2\|\boldsymbol{v}_k\|_2^2. \tag{7.5.8}$$

[34] For example, the $\log\det(\cdot)$ function arises in the context of lossy data compression [MDHW07] as a good measure of the binary coding length for encode data that span a low-dimensional subspace. As we will see in Chapter 16, this nonconvex measure plays a crucial role in a principled approach to derive and interpret modern deep neural networks. In that context, the convex nuclear norm becomes inadequate.

*Notice that the second term is very similar to that in the previous exercise, except for the square. The stochastic factorization can be used to model the so-called "dropout" techniques used in training deep neural networks, introduced by [SHK+14].*

7.7. *Consider the factorization of a matrix* $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^* \doteq \sum_{k=1}^{d} \boldsymbol{u}_k \boldsymbol{v}_k^*$ *and the associated quantity:*

$$\rho(\boldsymbol{U}, \boldsymbol{V}) \doteq \sum_{k=1}^{d} \|\boldsymbol{u}_k\|_2^2 \|\boldsymbol{v}_k\|_2^2.$$

*Show that if we may allow the factor to be of arbitrarily large size, that is,* $d$ *can be arbitrarily large, then we have*

$$\inf_{d, \boldsymbol{X}=\boldsymbol{U}\boldsymbol{V}^*} \rho(\boldsymbol{U}, \boldsymbol{V}) = 0.$$

*This property shows that the second term in the previous exercise prefers redundant factorization if we allow* $d$ *to be free.*

7.8 (Dropout as Low-rank Regularization). *Now in the stochastic matrix factorization exercise above, consider the sampling probability* $\theta$ *of the Bernoulli random variables is a function of the number* $d$ *of columns in* $\boldsymbol{U}$ *and* $\boldsymbol{V}$*: for a given* $p$*,* $0 < p < 1$*,*

$$\theta(d) = \frac{p}{d - (d-1)p}. \tag{7.5.9}$$

*Then show that*

$$\inf_{d, \boldsymbol{X}=\boldsymbol{U}\boldsymbol{V}^*} \frac{1-\theta(d)}{\theta(d)} \sum_{k=1}^{d} \|\boldsymbol{u}_k\|_2^2 \|\boldsymbol{v}_k\|_2^2 = \frac{1-p}{p} \|\boldsymbol{X}\|_*^2. \tag{7.5.10}$$

*Conclude that with the above choice of sampling rate, the above dropout technique is equivalent to:*

$$\min_{d, \boldsymbol{U}, \boldsymbol{V}} \mathbb{E} \left\| \boldsymbol{Y} - \frac{1}{\theta(d)} \boldsymbol{U}\, diag(\boldsymbol{r}) \boldsymbol{V}^* \right\|_F^2 = \min_{\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{X}\|_2^2 + \frac{1-p}{p} \|\boldsymbol{X}\|_*^2. \tag{7.5.11}$$

7.9 (Nuclear Norm Squared as a Regularizer). *The above exercise shows that the dropout technique used in deep learning is essentially equivalent to regularize parameters of two adjacent layers through a nuclear norm square penalty. Given a matrix* $\boldsymbol{Y}$ *with singular value decomposition* $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$*, show that the optimal solution to the following program:*

$$\min_{\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{X}\|_2^2 + \lambda \|\boldsymbol{X}\|_*^2$$

*is of the form* $\boldsymbol{X}_\star = \boldsymbol{U}\mathcal{S}_\mu(\boldsymbol{\Sigma})V^*$ *where* $\mathcal{S}_\mu$ *is a shrinkage operator with certain threshold depending on both* $\lambda$ *and* $\boldsymbol{\Sigma}$*. This concludes that stochastic matrix factorization (a.k.a. dropout in deep learning) is essentially imposing low-rank regularization on the resulting matrix.*

7.10 (Low-rank Regularization through Over-parameterization and Implicit Bias). *In this exercise, we reconsider the* affine rank minimization *problem studied in Chapter 4:*

$$\min_{\boldsymbol{X}} \operatorname{rank}(\boldsymbol{X}) \quad \textit{subject to} \quad \mathcal{A}[\boldsymbol{X}] = \boldsymbol{y}. \tag{7.5.12}$$

*Here $\boldsymbol{y} = \mathcal{A}[\boldsymbol{X}_o] \in \mathbb{R}^m$ is the observation, and we consider the special case that $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ is a symmetric matrix and $\mathcal{A}$ is a linear map: $\mathbb{R}^{n\times n} \to \mathbb{R}^m$. Hence each measurement is of the form $y_i = \langle \boldsymbol{A}_i, \boldsymbol{X}\rangle$ for some matrix $\boldsymbol{A}_i \in \mathbb{R}^{n\times n}$, see also (4.3.2). For simplicity, we here further assume the measurement matrices $\boldsymbol{A}_i$ are commutable, i.e., $\boldsymbol{A}_i\boldsymbol{A}_j = \boldsymbol{A}_j\boldsymbol{A}_i$ for all $i, j$.*

*To recover the low-rank solution $\boldsymbol{X}_o$, we over-parameterize $\boldsymbol{X}$ as $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{U}^*$ with $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ and consider solving the following nonconvex program:*

$$\min_{\boldsymbol{U}} f(\boldsymbol{U}) \doteq \|\mathcal{A}[\boldsymbol{U}\boldsymbol{U}^*] - \boldsymbol{y}\|_2^2. \tag{7.5.13}$$

*Obviously the above program does not have a unique solution as $\boldsymbol{X}$ is over-parameterized by $\boldsymbol{U}$. We are interested in how we can still recover the correct solution $\boldsymbol{X}_o$ by taking a special optimization strategy. Let us construct $\boldsymbol{U}(t)$ as the solution to the gradient flow of $f(\boldsymbol{U})$:*

$$\dot{\boldsymbol{U}}(t) = -\nabla f(\boldsymbol{U}(t)) = -\mathcal{A}^*[\mathcal{A}[\boldsymbol{U}(t)\boldsymbol{U}^*(t)] - \boldsymbol{y}]\boldsymbol{U}(t), \tag{7.5.14}$$

*where $\mathcal{A}^*$ is the adjoint of the linear map $\mathcal{A}$. Let $\boldsymbol{e}(t) \doteq \mathcal{A}[\boldsymbol{U}(t)\boldsymbol{U}^*(t)] - \boldsymbol{y} \in \mathbb{R}^m$.*

1 *Show that, under the above flow of $\boldsymbol{U}(t)$, $\boldsymbol{X}(t) = \boldsymbol{U}(t)\boldsymbol{U}^*(t)$ satisfies the following differential equation:*

$$\dot{\boldsymbol{X}}(t) = -\mathcal{A}^*[\boldsymbol{e}(t)]\boldsymbol{X}(t) - \boldsymbol{X}(t)\mathcal{A}^*[\boldsymbol{e}(t)]. \tag{7.5.15}$$

2 *Starting from $\boldsymbol{X}(0) = \boldsymbol{X}_o$, derive the solution to $\boldsymbol{X}(t)$ for the special case of $m = 1$.*

3 *Assume the following limits exist* [35]

$$\boldsymbol{X}_\infty(\boldsymbol{X}_o) = \lim_{t\to\infty} \boldsymbol{X}(t), \quad \textit{and} \quad \hat{\boldsymbol{X}} = \lim_{\varepsilon\to 0} \boldsymbol{X}_\infty(\varepsilon\boldsymbol{X}_o).$$

*Show that $\hat{\boldsymbol{X}}$ is the optimal solution to the following (familiar) program:*

$$\min_{\boldsymbol{X}} \|\boldsymbol{X}\|_* \quad \textit{subject to} \quad \mathcal{A}[\boldsymbol{X}] = \boldsymbol{y}, \tag{7.5.16}$$

*where here $\mathcal{A}[\boldsymbol{X}] = \langle \boldsymbol{A}_1, \boldsymbol{X}\rangle$ since $m = 1$.*

4 *Now generalize this to the case of $m$ measurements, show that $\hat{\boldsymbol{X}}$ is the optimal solution to the above convex program as long as $\boldsymbol{A}_i, i = 1, \ldots, m$ are commutable.*

*One may view this as an extension of the over-parameterization for sparsity in Exercise 7.3 to the case for low-rank matrices.*

[35] We leave the conditions under which such limits exist for students as extra bonus questions.

# Part II

## Computation for Large-Scale Problems

# 8 Convex Optimization for Structured Signal Recovery

*"In our opinion, convex optimization is a natural next topic after advanced linear algebra (topics like least-squares, singular values), and linear programming."*
– Stephen Boyd and Lieven Vandenberghe, *Convex Optimization*

In the previous theoretical Part I of the book, we showed that under fairly broad conditions on the number of measurements needed, many important classes of structured signals can be recovered via computationally tractable optimization problems, such as $\ell^1$ minimization for recovering sparse signals and nuclear norm minimization for recovering low-rank matrices. As we will see in Part III of this book, many of these structures are essential for modeling high-dimensional data that arise in a wide variety of applications. Hence, from a practical perspective, it is important that we develop efficient and scalable algorithms for these classes of optimization problems. We take on this task in the coming two chapters.

In this chapter, we mainly focus on the *convex approach* for structured signal recovery (and leave nonconvex optimization to the next Chapter 9). There are two compelling reasons to study the convex approach first. First, the previous chapters have established very precise conditions under which convex programs give correct solutions to the recovery problems. Second, as we will see through this chapter, the class of convex programs we are dealing with have unique properties that lend themselves to faster and more scalable solutions than generic convex programs. Although we will primarily use $\ell^1$ norm or nuclear norm minimization as working examples, the techniques that we introduce are fairly general, extending to a much broader class of convex programs with similar structure.

This chapter (or book) is not intended to give a comprehensive introduction to convex analysis and optimization, for which there are already excellent references such as [BV04,BNO03,N+18]. Instead, this chapter will focus mainly on showing how one can exploit special structures of the problems so as to develop more efficient and scalable algorithms than generic convex optimization methods. To make the book self-contained, we briefly survey related concepts and properties of convex functions as well as generic optimization methods in Appendices B-D.

## 8.1 Challenges and Opportunities

In this chapter, we will describe a few basic ideas which go a long way towards achieving this, by leveraging special properties of the particular convex optimization problems that arise in structured signal recovery. Our discussion will center around four model problems: basis pursuit (i.e., equality constrained $\ell^1$ minimization), its regularized version (basis pursuit denoising), principal component pursuit, and its regularized version. We recap these four optimization problems below:

Recall the problem of recovering a sparse vector $\boldsymbol{x}_o \in \mathbb{R}^n$ from observations $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o \in \mathbb{R}^m$ via convex program, also known as *basis pursuit* (BP):

$$\begin{array}{ll} \min_{\boldsymbol{x}} & \|\boldsymbol{x}\|_1 \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}. \end{array} \tag{8.1.1}$$

A variant of the problem considers the noisy case, in which the observations $\boldsymbol{y}$ are contaminated by moderate Gaussian noise $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o + \boldsymbol{z}$, also known as the *Lasso*:

$$\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1, \tag{8.1.2}$$

where $\lambda$ is a scalar weight parameter.

In robust PCA, the goal is to recover a low-rank matrix $\boldsymbol{L}_o$ from sparsely corrupted observations $\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{S}_o \in \mathbb{R}^{m\times n}$. A natural approach suggested in earlier chapters is to solve the so called *principal component pursuit* (PCP) program:

$$\begin{array}{ll} \min_{\boldsymbol{L},\boldsymbol{S}} & \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 \\ \text{subject to} & \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}, \end{array} \tag{8.1.3}$$

where $\lambda > 0$ is a scalar weight. Again, if the data are noisy we could also consider to solve a stable version of the PCP program:

$$\min_{\boldsymbol{L},\boldsymbol{S}} \ \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 + \frac{\mu}{2}\|\boldsymbol{Y} - \boldsymbol{L} - \boldsymbol{S}\|_F^2, \tag{8.1.4}$$

to produce stable estimates $\hat{\boldsymbol{L}}$ and $\hat{\boldsymbol{S}}$, where $\lambda, \mu > 0$ are two scalar weights.

*Challenge of Scale.*
When the dimension of the problem is not so high, one could simply apply classical second-order convex optimization algorithms, such as the interior-point methods (see e.g. [BV04]), to solve the above convex programs. These powerful methods, under favorable conditions such as for smooth strongly convex functions, need only very few iterations to converge to a highly accurate solution: $O(\log(1/\varepsilon))$, where $\varepsilon$ is the target accuracy. However, for problems in $n$ variables, each iteration requires the solution to a system of linear equations of size $n \times n$, incurring a typical per-iteration cost of $O(n^3)$. For applications in modern signal processing, the number of variables $n$ can be quite high – in the case of image processing, it is typically of the same magnitude as the number of pixels,

easily in the range of millions. For such problems, the cost of a single iteration is prohibitively large. As a result, we will need to consider simple alternatives with cheaper iterations.

EXAMPLE 8.1 (Solving large-scale BP problems via interior-point methods). *Just to motivate yourself from a practical perspective, construct a simulation to plot the average run time of BP on CVX from 100 to 1,000 dimensions. See how a generic convex optimization solver (that is mainly based on second-order, interior point method) scales for this class of optimization problems.*

*Difficulty with Nonsmoothness.*
The large scale of the problems forces us to consider simple, scalable algorithms that use only first-order information about the objective function. The prototypical first-order method is *gradient descent*. However, one technical difficulty arises though as the objective functions may contain nonsmooth terms that are not differentiable. For instance, the $\ell^1$ norm $\|\boldsymbol{x}\|_1$ in the BPDN problem does not have a gradient in the normal sense. In such cases, the simplest solution is to employ a generic subgradient method, as we did in Chapter 2. Although the subgradient method has very simple and efficient iterations, its rate of convergence is very poor, typically $O(1/\sqrt{k})$.[1] That means it usually takes many (thousands of) iterations for the algorithm to converge to the optimal solution. In this chapter, we will show how to exploit some important properties of structured signal recovery. Such properties allow us to develop gradient descent algorithms as if the objective is smooth, the so called Proximal Gradient (PG) method (Section 8.2). The same properties also allow us to utilize acceleration techniques that were designed for smooth functions and lead to much more scalable and fast-converging algorithms, with convergence rates much better than the generic situation (Section 8.3).

*Enforcing Equality Constraints.*
To solve the basis pursuit problem (8.1.1), we need to ensure that the final solution $\boldsymbol{x}$ satisfies the equality constraint $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ exactly. A naive way to enforce the equality constraint is to incorporate it as a penalty term and minimize: $\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 + \frac{\mu}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$. This is a similar optimization problem as the BPDN except that we need to solve a series of problems of this type for an increasing sequence of $\mu_i \to \infty$ so as to enforce the equality constraint in the end. However, as the weight $\mu_i$ increases, the corresponding BPDN problem becomes increasingly ill-conditioned and hence algorithms converge slower. In Section 8.4, we will see how to employ the *augmented Lagrange multiplier* (ALM) technique to alleviate this difficulty.

*Exploiting Separable Structures.*
Often the structured signal that we are recovering is a superposition of multiple structured terms. This is the case for the principal component pursuit (PCP)

[1] See [Nem95, Nem07] for a characterization of typical subgradient methods.

program that we have mentioned earlier:

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \boldsymbol{Y} = \boldsymbol{L} + \boldsymbol{S}. \tag{8.1.5}$$

As we will show in Section 8.5, such separable structures of the objective function can be naturally exploited through methods such as *alternating direction of multipliers method* (ADMM). We end up with more simple and efficient algorithms for solving this class of convex programs as ADMM converts the global optimization to several subproblems of much smaller dimension.

Finally, in Section 8.6, we will study how to exploit additional structures either in the objective function or in the constraint set to further improve the scalability of the optimization algorithms against the problem dimension or the sample size.

## 8.2 Proximal Gradient Methods

As one can see, the optimization problems we are dealing with can be reduced to solve the structured convex minimization problems with objective functions of the form:

$$F(\boldsymbol{x}) \doteq f(\boldsymbol{x}) + g(\boldsymbol{x}), \tag{8.2.1}$$

where $f(\boldsymbol{x})$ is a smooth convex term and $g(\boldsymbol{x})$ is a convex but nonsmooth term. For instance, in the Lasso problem (8.1.2), we could set $f(\boldsymbol{x}) \doteq \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$ and $g(\boldsymbol{x}) \doteq \lambda\|\boldsymbol{x}\|_1$ with $\lambda > 0$. We want to develop both scalable and efficient algorithms for this type of problems.

Since the composite objective function $F(\boldsymbol{x})$ is not differentiable, generic gradient algorithms do not apply. The first recourse in this situation is to replace the gradient with a *subgradient*, yielding the simple subgradient method with the iteration:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_k \boldsymbol{g}_k, \quad \boldsymbol{g}_k \in \partial F(\boldsymbol{x}_k). \tag{8.2.2}$$

The main disadvantage to this approach is its relatively poor convergence rate.[2] Let $\boldsymbol{x}_\star$ be the (global) minimizer of $F(\boldsymbol{x})$. In general, the convergence rate of subgradient method for nonsmooth objective functions, in terms of function value $F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star)$, is (see [Nes03]):

$$O(1/\sqrt{k}). \tag{8.2.3}$$

The constants in the big-$O$ notation depend on various properties of the problem. The important point is that for even a moderate target accuracy

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star) \leq \varepsilon,$$

we will have to set $k = O(\varepsilon^{-2})$ very large.

[2] Also, the step size $\gamma_k$ can be challenging to set.

### 8.2.1 Convergence of Gradient Descent

We can compare the behavior of the subgradient method with the behavior of the simple *gradient descent* method for minimizing a smooth function. Consider, briefly, the simpler problem

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}), \tag{8.2.4}$$

with $f$ convex and differentiable. The gradient descent iteration for this problem is

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_k \nabla f(\boldsymbol{x}_k). \tag{8.2.5}$$

This iteration comes from a first-order approximation to $f$ at $\boldsymbol{x} = \boldsymbol{x}_k$:

$$f(\boldsymbol{x}') \;\geq\; f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle \,. \tag{8.2.6}$$

Because $f$ is convex, this first-order approximation provides a global lower bound on $f$. Nevertheless, we expect this lower bound to be more accurate in a neighborhood of $\boldsymbol{x}$. The size of this neighborhood depends substantially on the properties of $f$. For example, if $f$ is relatively smooth, and its gradient does not vary much from point to point, we might imagine that the first-order approximation at $\boldsymbol{x}$ would be accurate over a relatively large region. To make this more formal, we say that a differentiable function $f(\boldsymbol{x})$ has *L-Lipschitz continuous gradients* if

$$\|\nabla f(\boldsymbol{x}') - \nabla f(\boldsymbol{x})\|_2 \leq L \|\boldsymbol{x}' - \boldsymbol{x}\|_2, \quad \forall \boldsymbol{x}', \boldsymbol{x} \in \mathbb{R}^n \tag{8.2.7}$$

for some $L > 0$. The quantity $L$ is known as the *Lipschitz constant* of $\nabla f$.

When the Lipschitz condition holds, a bit of calculus shows that we can complement the linear lower bound (8.2.6) with a corresponding quadratic upper bound:

LEMMA 8.2. *Suppose that $f$ is differentiable, and $\nabla f$ is $L$-Lipschitz. Then for every $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^n$,*

$$f(\boldsymbol{x}') \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle + \frac{L}{2} \|\boldsymbol{x}' - \boldsymbol{x}\|_2^2 \,. \tag{8.2.8}$$

*Proof* We calculate:

$$\begin{aligned}
f(\boldsymbol{x}') &= f(\boldsymbol{x} + t(\boldsymbol{x}' - \boldsymbol{x}))|_{t=1} && (8.2.9)\\
&= f(\boldsymbol{x}) + \int_{t=0}^{1} \frac{d}{dt} f(\boldsymbol{x} + t(\boldsymbol{x}' - \boldsymbol{x}))\, dt && (8.2.10)\\
&= f(\boldsymbol{x}) + \int_{t=0}^{1} \langle \nabla f(\boldsymbol{x} + t(\boldsymbol{x}' - \boldsymbol{x})), \boldsymbol{x}' - \boldsymbol{x} \rangle\, dt && (8.2.11)\\
&= f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle \\
&\quad + \int_{t=0}^{1} \langle \nabla f(\boldsymbol{x} + t(\boldsymbol{x}' - \boldsymbol{x})) - \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle\, dt && (8.2.12)\\
&\le f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle \\
&\quad + \int_{t=0}^{1} \|\nabla f(\boldsymbol{x} + t(\boldsymbol{x}' - \boldsymbol{x})) - \nabla f(\boldsymbol{x})\|_2 \|\boldsymbol{x}' - \boldsymbol{x}\|_2\, dt \\
&\le f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle + \int_{t=0}^{1} tL \|\boldsymbol{x}' - \boldsymbol{x}\|_2^2\, dt && (8.2.13)\\
&= f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle + \frac{L}{2} \|\boldsymbol{x}' - \boldsymbol{x}\|_2^2, && (8.2.14)
\end{aligned}$$

giving the claim. □

Thus, when $\nabla f$ is Lipschitz, we have a matching quadratic upper bound

$$\begin{aligned}
f(\boldsymbol{x}') \;\le\; \hat{f}(\boldsymbol{x}', \boldsymbol{x}) \doteq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle + \frac{L}{2} \|\boldsymbol{x}' - \boldsymbol{x}\|_2^2 && (8.2.15)\\
= \frac{L}{2} \left\|\boldsymbol{x}' - (\boldsymbol{x} - \tfrac{1}{L} \nabla f(\boldsymbol{x}))\right\|_2^2 + h(\boldsymbol{x}), && (8.2.16)
\end{aligned}$$

for some function $h(\boldsymbol{x})$ that does not depend on $\boldsymbol{x}'$. This upper bound agrees with $f$ at the point $\boldsymbol{x}$ at which it is formed: $f(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}, \boldsymbol{x})$. Suppose that we minimize this upper bound, with respect to $\boldsymbol{x}'$. By inspecting the second identity above, the minimizer has a very familiar form:

$$\arg\min_{\boldsymbol{x}'} \hat{f}(\boldsymbol{x}', \boldsymbol{x}) \;=\; \boldsymbol{x} - \frac{1}{L} \nabla f(\boldsymbol{x}). \tag{8.2.17}$$

This is simply a gradient descent step, taken from $\boldsymbol{x}$, with a special choice of step size $\gamma = 1/L$. Moreover, because $\hat{f}(\boldsymbol{x}, \boldsymbol{x}) = f(\boldsymbol{x})$, this minimization does not increase the objective function: if $\boldsymbol{x}'_\star \in \arg\min_{\boldsymbol{x}'} \hat{f}(\boldsymbol{x}', \boldsymbol{x})$, then

$$f(\boldsymbol{x}'_\star) \;\le\; \hat{f}(\boldsymbol{x}'_\star, \boldsymbol{x}) \;\le\; \hat{f}(\boldsymbol{x}, \boldsymbol{x}) \;=\; f(\boldsymbol{x}). \tag{8.2.18}$$

Thus, if we apply the gradient descent method with step size $1/L$, we are guaranteed to produce a monotone sequence of function values $f(\boldsymbol{x}_k)$. Furthermore, we can show convergence[3] to the optimal function value at a rate of $O(1/k)$:

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_\star) \;\le\; \frac{L \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2}{2k} \;=\; O(1/k). \tag{8.2.19}$$

[3] We will not prove (8.2.19) here, since we will obtain a more general result below which implies it.

This is still not a particularly fast rate of convergence, but it is much better than the $O(1/\sqrt{k})$ rate of convergence experienced by the subgradient algorithm on nonsmooth functions.

### 8.2.2 From Gradient to Proximal Gradient

Can we draw inspiration from the gradient method to produce a more efficient algorithm for minimizing the composite function $F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$, with $f$ differentiable? Again, the gradient method does not directly apply, since $F$ is nondifferentiable. Nevertheless, if the gradient $\nabla f$ of the smooth term is Lipschitz, we can still make a simpler upper bound to $F$, by upper bounding $f$, say around the current iterate $\boldsymbol{x}_k$, by a quadratic and leaving the nonsmooth term $g$ intact:

$$\hat{F}(\boldsymbol{x}, \boldsymbol{x}_k) \;=\; f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x} - \boldsymbol{x}_k \rangle + \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2 + g(\boldsymbol{x}). \tag{8.2.20}$$

Since above repeatedly minimizing $\hat{f}$ of (8.2.15) produced the gradient method, resulting in a better convergence rate, let us try minimizing the upper bound $\hat{F}$ around $\boldsymbol{x}_k$:

$$\boldsymbol{x}_{k+1} \;=\; \arg\min_{\boldsymbol{x}} \; \hat{F}(\boldsymbol{x}, \boldsymbol{x}_k). \tag{8.2.21}$$

For commonly encountered $g$, this minimization often takes on a very simple form. Completing the square in (8.2.20), we obtain that

$$\hat{F}(\boldsymbol{x}, \boldsymbol{x}_k) \;=\; \frac{L}{2} \left\| \boldsymbol{x} - (\boldsymbol{x}_k - \tfrac{1}{L} \nabla f(\boldsymbol{x}_k)) \right\|_2^2 + g(\boldsymbol{x}) \;+\; h(\boldsymbol{x}_k), \tag{8.2.22}$$

where $h(\boldsymbol{x}_k)$ is a term that depends only on $\boldsymbol{x}_k$.

Hence, the iteration (8.2.21) becomes

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \; \frac{L}{2} \left\| \boldsymbol{x} - (\boldsymbol{x}_k - \tfrac{1}{L} \nabla f(\boldsymbol{x}_k)) \right\|_2^2 + g(\boldsymbol{x}) \tag{8.2.23}$$

$$= \arg\min_{\boldsymbol{x}} g(\boldsymbol{x}) + \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{w}_k\|_2^2, \tag{8.2.24}$$

where for convenience we define $\boldsymbol{w}_k \doteq \boldsymbol{x}_k - (1/L)\nabla f(\boldsymbol{x}_k)$. Thus, at each step of the iteration (8.2.21), we have to minimize the function $g$ plus a separable quadratic $\frac{L}{2} \|\boldsymbol{x} - \boldsymbol{w}_k\|_2^2$. In a sense, this problem asks us to make $g$ as small as possible, while not straying too far from the point $\boldsymbol{w}_k$. Because $\|\cdot\|_2^2$ is strongly convex, this problem always has a unique solution. So, the sequence $\boldsymbol{x}_k$ defined recursively by (8.2.21) is well-defined.

In fact, the operation of minimizing a convex function $g$ plus a separable quadratic $\|\boldsymbol{x} - \boldsymbol{w}_k\|_2^2$ recurs so frequently in convex analysis and optimization that it has its own name. This is known as the *proximal operator* for the convex function $g(\boldsymbol{x})$:

DEFINITION 8.3 (Proximal Operator). *The proximal operator of a convex function $g$ is*

$$\text{prox}_g[\boldsymbol{w}] \doteq \arg\min_{\boldsymbol{x}} \left\{ g(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{w}\|_2^2 \right\}. \tag{8.2.25}$$

In this language, iteration (8.2.21) can be written as

$$\boldsymbol{x}_{k+1} = \text{prox}_{g/L}[\boldsymbol{w}_k]. \tag{8.2.26}$$

Fortunately, many of the convex functions (or norms) that we encounter in structured signal recovery either have closed-form proximal operators or proximal operators that can be computed very efficiently via numerical means. We give a few examples below:

PROPOSITION 8.4. *Proximal operators for the indicator function, $\ell^1$ norm, and nuclear norm are given by:*

1 *Let $g(\boldsymbol{x}) = I_{\mathcal{D}}$ be the indicator function for a closed convex set $\mathcal{D}$, namely, $I_{\mathcal{D}}(\boldsymbol{x}) = 0, \boldsymbol{x} \in \mathcal{D}$ otherwise $I_{\mathcal{D}}(\boldsymbol{x}) = \infty$. Then $\text{prox}_g[\boldsymbol{w}]$ is the projection operator:*

$$\text{prox}_g[\boldsymbol{w}] \;=\; \arg\min_{\boldsymbol{x}\in\mathcal{D}} \|\boldsymbol{x}-\boldsymbol{w}\|_2^2 \;=\; \mathcal{P}_{\mathcal{D}}[\boldsymbol{w}].$$

2 *Let $g(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1$ be the $\ell^1$ norm. Then $\text{prox}_g[\boldsymbol{w}]$ is the* soft-thresholding *function applied element-wise:*

$$(\text{prox}_g[\boldsymbol{w}])_i = \text{soft}(w_i, \lambda) \doteq \text{sign}\,(w_i)\max(|w_i| - \lambda, 0).$$

3 *Let $g(\boldsymbol{X}) = \lambda\|\boldsymbol{X}\|_*$ be the matrix nuclear norm. Then $\text{prox}_g[\boldsymbol{W}]$ is the* singular-value soft thresholding *function:*

$$\text{prox}_g[\boldsymbol{W}] = \boldsymbol{U}\text{soft}(\boldsymbol{\Sigma}, \lambda)\boldsymbol{V}^*,$$

*where $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$ are the singular value decomposition (SVD) of $\boldsymbol{W}$. In other words, $\text{prox}_g[\boldsymbol{W}]$ applies component-wise soft thresholding on the singular values of $\boldsymbol{W}$.*

*Proof* We prove the second assertion and leave the rest to the reader as exercises. The objective function reaches minimum when the subdifferential of $\lambda\|\boldsymbol{x}\|_1 + \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{w}\|_2^2$ contains zero,

$$0 \in (\boldsymbol{x}-\boldsymbol{w}) + \lambda\partial\|\boldsymbol{x}\|_1 = \begin{cases} x_i - w_i + \lambda, & x_i > 0 \\ -w_i + \lambda[-1,1], & x_i = 0 \\ x_i - w_i - \lambda, & x_i < 0 \end{cases}, \quad i = 1, \ldots, n.$$

Therefore, the solution to this optimality condition is the soft-thresholding function applied element-wise:

$$x_{i\star} = \text{soft}(w_i, \lambda) \doteq \text{sign}\,(w_i)\max(|w_i| - \lambda, 0), \quad i = 1, \ldots, n.$$

See Figure 8.1 left for an illustration of the soft-thresholding function. We leave

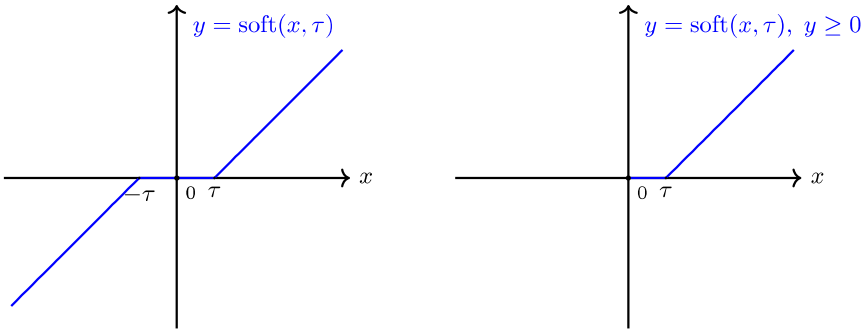



**Figure 8.1** Illustrations of soft-thresholding (or shrinkage) operators associated with proximal operators of the $\ell^1$ norm (left) and the nuclear norm (right), respectively. Note that singular values are always nonnegative. Typically the threshold $\tau \geq 0$ is a small value.

the first and third assertions as exercises to the reader. [*Hint:* for the first, use the definition; for the third, use the subdifferential of $\|\cdot\|_*$.] □

Example 8.5 (Proximal Operators for Powers of Nuclear Norm). *In problems such as high-order low-rank tensor completion [ZZWM14] or stochastic matrix factorization [CMH+17] (also known as "dropout" in deep learning, see Exercise 7.8 of Chapter 7), we may need to find the proximal operator for a given matrix* $\boldsymbol{W}$*:*

$$\text{prox}_g[\boldsymbol{W}] \doteq \arg\min_{\boldsymbol{X}} \left\{ g(\boldsymbol{X}) + \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\|_F^2 \right\}, \tag{8.2.27}$$

*for* $g(\boldsymbol{X})$ *as certain powers of nuclear norm or its exponential,*[4] *say*

$$g(\boldsymbol{X}) = \lambda\|\boldsymbol{X}\|_*^2 \quad \textit{or} \quad g(\boldsymbol{X}) = \lambda e^{\|\boldsymbol{X}\|_*}. \tag{8.2.28}$$

*For each of these two cases, one can show that the proximal operator takes the form:*

$$\text{prox}_g[\boldsymbol{W}] = \boldsymbol{U}\,\text{soft}(\boldsymbol{\Sigma}, \tau)\boldsymbol{V}^*,$$

*where* $\tau$ *is certain threshold that depends on* $\lambda$ *and the singular values of* $\boldsymbol{W}$*. See Figure 8.1 right for an illustration of the soft-thresholding function on the singular values. In fact, this is true if* $g(\boldsymbol{X}) = f(\|\boldsymbol{X}\|_*)$ *for any monotonic convex function* $f$*. The only question is whether the associated threshold* $\tau$ *can be solved in closed-form or efficiently computed numerically. We explore some of these extensions in the exercises (see Exercise 8.4). The reader may further explore whether the same property holds for any unitary invariant matrix norm (introduced in Appendix A.9).*

Thus, for the problems of our interest, we can compute the proximal operator

[4] The reader may refer to [ZM20] for the more general case.

**Proximal Gradient (PG)**

**Problem Class:**

$$\min_{\boldsymbol{x}} F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$$

$f, g : \mathbb{R}^n \to \mathbb{R}$ are convex, $\nabla f$ $L$-Lipschitz and $g$ (maybe) nonsmooth.

**Basic Iteration:** set $\boldsymbol{x}_0 \in \mathbb{R}^n$.
Repeat:

$$\boldsymbol{w}_k \leftarrow \boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k),$$
$$\boldsymbol{x}_{k+1} \leftarrow \mathrm{prox}_{g/L}[\boldsymbol{w}_k].$$

**Convergence Guarantee:**

$F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star)$ converges at a rate of $O(1/k)$.

**Figure 8.2** An overview of the Proximal Gradient Method.

efficiently. In this setting, it provides an alternative replacement for the gradient step. Unlike the subgradient method, this *proximal gradient* algorithm enjoys a convergence rate of $O(1/k)$ – exactly the same as if the nonsmooth term was not present! More formally:

THEOREM 8.6 (Convergence of Proximal Gradient). *Let $F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$, where $f$ is a convex, differentiable function with $L$-Lipschitz continuous gradients, and $g$ a convex function. Consider the following iterative update scheme:*

$$\boldsymbol{w}_k \leftarrow \boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k), \quad \boldsymbol{x}_{k+1} \leftarrow \mathrm{prox}_{g/L}[\boldsymbol{w}_k].$$

*Assume $F(\boldsymbol{x})$ has a minimum at $\boldsymbol{x}_\star$. Then for any $k \geq 1$,*

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star) \leq \frac{L\|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2}{2k}.$$

We will give a detailed proof to this theorem in Section 8.2.4. Thus, for a composite nonsmooth convex function, under certain conditions, we can still obtain an efficient "gradient descent" like algorithm that has the same convergence rate $O(1/k)$ as that for a smooth function. As long as the nonsmooth part has an easy-to-solve proximal operator, the proximal gradient algorithm has very cheap iterations. Hence it is typically much more scalable than second-order methods. We summarize properties of the iterative process that we have derived so far for minimizing the convex composite problem in Figure 8.2, which is also known as the *proximal gradient* method.

### 8.2.3 Proximal Gradient for the Lasso and Stable PCP

For the rest of the section, we will see how to apply the proximal gradient algorithm to several important cases of structured signal recovery problems that we have encountered.

*Proximal Gradient for the Lasso.*
As the first instance, the Lasso problem (8.1.2) obviously falls into the class of problems that can be addressed by the proximal gradient method. We can view $g$ to be the $\ell^1$ norm function $\lambda\|\boldsymbol{x}\|_1$ whose proximal operator is given in Proposition 8.4; $f$ is simply the quadratic data term $\frac{1}{2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2$ whose gradient is clearly Lipschitz: the Lipschitz constant $L$ can be the largest eigenvalue of the matrix $\boldsymbol{A}^*\boldsymbol{A}$, which can be computed in advance.

The resulting proximal gradient descent algorithm for Lasso is sometimes referred to as the *iterative soft-thresholding algorithm* (ISTA), which is summarized in Algorithm 8.1. In terms of computational complexity, the main cost is calcu-

**Algorithm 8.1** Proximal Gradient (PG) for Lasso

1: **Problem:** $\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2+\lambda\|\boldsymbol{x}\|_1$, given $\boldsymbol{y}\in\mathbb{R}^m$, $\boldsymbol{A}\in\mathbb{R}^{m\times n}$.
2: **Input:** $\boldsymbol{x}_0\in\mathbb{R}^n$ and $L\geq\lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})$.
3: **for** $(k=0,1,2,\ldots,K-1)$ **do**
4: $\quad\boldsymbol{w}_k \leftarrow \boldsymbol{x}_k-\frac{1}{L}\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_k-\boldsymbol{y})$.
5: $\quad\boldsymbol{x}_{k+1}\leftarrow \mathrm{soft}(\boldsymbol{w}_k,\lambda/L)$.
6: **end for**
7: **Output:** $\boldsymbol{x}_\star\leftarrow\boldsymbol{x}_K$.

lating the gradient $\nabla f(\boldsymbol{x})=\boldsymbol{A}^*\boldsymbol{A}\boldsymbol{x}-\boldsymbol{A}^*\boldsymbol{y}$ in the inner loop, which in general takes time $O(mn)$.

EXAMPLE 8.7. *We randomly generate a sparse signal $\boldsymbol{x}\in\mathbb{R}^{1,000}$ and then add a small Gaussian noise $\boldsymbol{n}$ to all of its coefficients, as shown in Figure 8.3 Top. With the added Gaussian noise, the signal $\boldsymbol{w}=\boldsymbol{x}+\boldsymbol{n}$ is not sparse anymore. Then we may try to recover $\boldsymbol{x}$ from $\boldsymbol{w}$ by solving the following problem $\min_{\boldsymbol{x}}\lambda\|\boldsymbol{x}\|_1+\frac{1}{2}\|\boldsymbol{w}-\boldsymbol{x}\|_2^2$, where $\lambda$ is proportional to the noise level. We know the solution to this problem is simply the soft thresholding $\hat{\boldsymbol{x}}=\mathrm{soft}(\boldsymbol{w},\lambda)$. The result is shown in Figure 8.3 Bottom. We see that the operator successfully removes most of the noise in $\boldsymbol{w}$ and returns a sparse estimate for $\boldsymbol{x}$.*

*Proximal Gradient for Stable PCP.*
According to Proposition 8.4, the nuclear norm $\|\boldsymbol{X}\|_*$ also has a simple proximal operator. Hence we could apply proximal gradient algorithm to solve low-rank matrix recovery problems. For instance, the stable principal component pursuit (PCP) program is also for the form that is amenable to proximal gradient

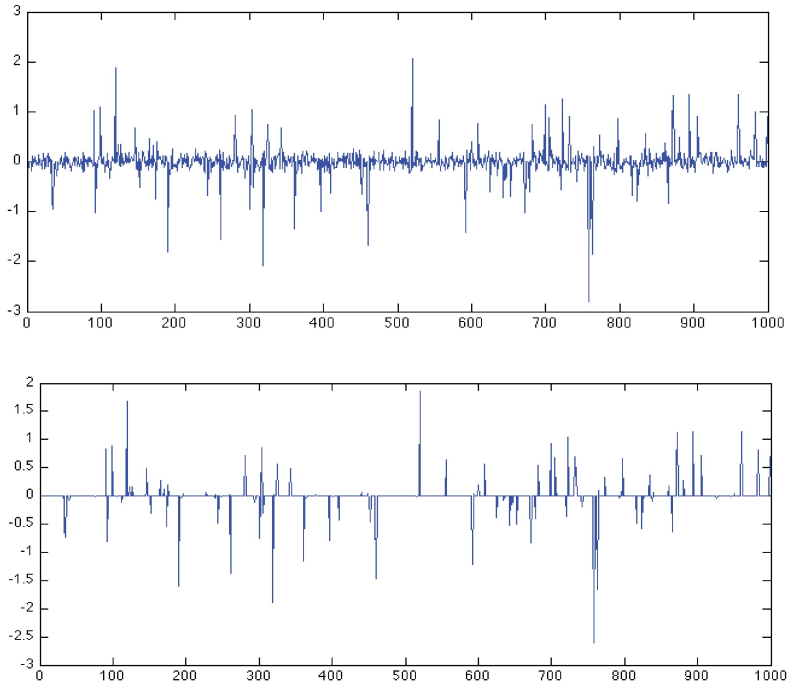



**Figure 8.3** **Top:** A sparse signal $\boldsymbol{x}$ perturbed by small Gaussian noise $\boldsymbol{n}$. **Bottom:** The output from a properly chosen soft-thresholding function $\text{soft}(\boldsymbol{w}, \lambda)$.

method:

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 + \frac{\mu}{2}\|\boldsymbol{Y} - \boldsymbol{L} - \boldsymbol{S}\|_F^2. \tag{8.2.29}$$

Notice that however, for this problem the nonsmooth term $g(\boldsymbol{L}, \boldsymbol{S}) = \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1$ now contains two nonsmooth functions $\|\boldsymbol{L}\|_*$ and $\lambda\|\boldsymbol{S}\|_1$, each having a simple proximal operator.

We leave as an exercise for the reader to prove the following simple fact about the proximal operator of a separable convex function, which comes handy for this problem: Let $\boldsymbol{x} = [\boldsymbol{x}_1; \boldsymbol{x}_2]$ and $g(\boldsymbol{x}) = g_1(\boldsymbol{x}_1) + g_2(\boldsymbol{x}_2)$ be a separable function. Then

$$\text{prox}_g[\boldsymbol{w}] = \left(\text{prox}_{g_1}[\boldsymbol{w}_1], \text{prox}_{g_2}[\boldsymbol{w}_2]\right),$$

where $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ in $\boldsymbol{w} = [\boldsymbol{w}_1; \boldsymbol{w}_2]$ correspond to the variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in $\boldsymbol{x}$, respectively.

Hence, the proximal operator for $g(\boldsymbol{L}, \boldsymbol{S})$ can be computed separately from the proximal operators for the $\ell^1$ norm for $\boldsymbol{S}$ and nuclear norm for $\boldsymbol{L}$, respectively. The rest of the proximal gradient algorithm for the stable principal component pursuit program then is easy to derive (and we leave this as an exercise to the

reader. See Exercise 8.6.). We summarize the overall algorithm below for clarity.

**Algorithm 8.2** Proximal Gradient for Stable Principal Component Pursuit

**Problem:** $\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 + \frac{\mu}{2}\|\boldsymbol{Y} - \boldsymbol{L} - \boldsymbol{S}\|_F^2$, given $\boldsymbol{Y}$, $\lambda, \mu > 0$.
**Input:** $\boldsymbol{L}_0 \in \mathbb{R}^{m\times n}$, $\boldsymbol{S}_0 \in \mathbb{R}^{m\times n}$.
**for** $(k = 0, 1, 2, \ldots, K-1)$ **do**
  $\boldsymbol{W}_k \leftarrow \boldsymbol{Y} - \boldsymbol{S}_k$ and compute $\boldsymbol{W}_k = \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^*$.
  $\boldsymbol{L}_{k+1} \leftarrow \boldsymbol{U}_k \mathrm{soft}(\boldsymbol{\Sigma}_k, 1/\mu)\boldsymbol{V}_k^*$.
  $\boldsymbol{S}_{k+1} \leftarrow \mathrm{soft}((\boldsymbol{Y} - \boldsymbol{L}_k), \lambda/\mu)$.
**end for**
**Output:** $\boldsymbol{L}_\star \leftarrow \boldsymbol{L}_K$; $\boldsymbol{S}_\star \leftarrow \boldsymbol{S}_K$.

### 8.2.4 Convergence of Proximal Gradient

In this subsection, we prove Theorem 8.6. We find it convenient to do this in two steps. In the first step, we provide an analysis of a simpler algorithm, known as the *proximal point algorithm*, which only consists of repeated application of the proximal operator. This algorithm is of independent interest, and we will reuse its analysis when we encounter Augmented Lagrangian techniques.

PROPOSITION 8.8 (Convergence of Proximal Point Algorithm). *Let $g : \mathbb{R}^n \to \mathbb{R}$ be a convex function, and $\boldsymbol{x}_\star$ a minimizer of $g$. Let $\boldsymbol{x}_0 \in \mathbb{R}^n$ be arbitrary, and consider the iteration*

$$\boldsymbol{x}_{k+1} = \mathrm{prox}_{\gamma_k g}[\boldsymbol{x}_k], \tag{8.2.30}$$

*with $\gamma_k \in \mathbb{R}_+$. Then*

$$g(\boldsymbol{x}_{k+1}) - g(\boldsymbol{x}_\star) \;\leq\; \frac{\|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2}{2\sum_{i=0}^k \gamma_i}. \tag{8.2.31}$$

*Moreover, if $\sum_{i=0}^\infty \gamma_i = +\infty$, then $\boldsymbol{x}_k \to \boldsymbol{x}_\star$, a minimizer of $g$.*

*Proof* By construction,

$$\boldsymbol{0} \in \gamma_k \partial g(\boldsymbol{x}_{k+1}) + \boldsymbol{x}_{k+1} - \boldsymbol{x}_k. \tag{8.2.32}$$

Equivalently, $\boldsymbol{x}_k - \boldsymbol{x}_{k+1} \in \gamma_k \partial g(\boldsymbol{x}_{k+1})$. Using the convexity of $g(\boldsymbol{x})$, we have that

$$\langle \boldsymbol{x}_k - \boldsymbol{x}_{k+1}, \boldsymbol{x}_k - \boldsymbol{x}_{k+1}\rangle \leq \gamma_k \left(g(\boldsymbol{x}_k) - g(\boldsymbol{x}_{k+1})\right). \tag{8.2.33}$$

Since left hand side is nonnegative, $g(\boldsymbol{x}_k) \geq g(\boldsymbol{x}_{k+1})$. So, the objective function value is nonincreasing.

Using the subgradient inequality again, we have that

$$\langle \boldsymbol{x}_\star - \boldsymbol{x}_{k+1}, \boldsymbol{x}_k - \boldsymbol{x}_{k+1}\rangle \leq \gamma_k \left(g(\boldsymbol{x}_\star) - g(\boldsymbol{x}_{k+1})\right). \tag{8.2.34}$$

Let us use this fact to bound the distance of $\boldsymbol{x}_{k+1}$ to the optimum. Notice that

$$\begin{aligned}
&\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_\star\|_2^2 = \|\boldsymbol{x}_k - \boldsymbol{x}_\star + \boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \\
&= \|\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2^2 + 2\langle \boldsymbol{x}_k - \boldsymbol{x}_\star, \boldsymbol{x}_{k+1} - \boldsymbol{x}_k\rangle + \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \\
&= \|\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2^2 - \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 + 2\langle \boldsymbol{x}_{k+1} - \boldsymbol{x}_\star, \boldsymbol{x}_{k+1} - \boldsymbol{x}_k\rangle \\
&\le \|\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2^2 + 2\langle \boldsymbol{x}_{k+1} - \boldsymbol{x}_\star, \boldsymbol{x}_{k+1} - \boldsymbol{x}_k\rangle \\
&\le \|\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2^2 + 2\gamma_k \left(g(\boldsymbol{x}_\star) - g(\boldsymbol{x}_{k+1})\right). \qquad (8.2.35)
\end{aligned}$$

Since $g(\boldsymbol{x}_{k+1}) \ge g(\boldsymbol{x}_\star)$, the distance of $\boldsymbol{x}_k$ to $\boldsymbol{x}_\star$ also does not increase. In fact, we can say slightly more. Summing the relationship (8.2.35), we obtain

$$\sum_{i=0}^{k} 2\gamma_i \left(g(\boldsymbol{x}_{i+1}) - g(\boldsymbol{x}_\star)\right) \;\le\; \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2. \qquad (8.2.36)$$

Since $g(\boldsymbol{x}_i)$ is nonincreasing, this implies that

$$2\left(\sum_{i=0}^{k} \gamma_i\right)\left(g(\boldsymbol{x}_{k+1}) - g(\boldsymbol{x}_\star)\right) \;\le\; \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2. \qquad (8.2.37)$$

This gives convergence in function values, as in (8.2.31).

Since $\|\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2$ is nonincreasing, the sequence $\boldsymbol{x}_k$ is bounded, and hence has a cluster point $\bar{\boldsymbol{x}}$. Since $g(\boldsymbol{x}_k) \searrow g(\boldsymbol{x}_\star)$, $g(\bar{\boldsymbol{x}}) = g(\boldsymbol{x}_\star)$, and hence $\bar{\boldsymbol{x}}$ is optimal. Applying inequality (8.2.35) with $\boldsymbol{x}_\star$ replaced by $\bar{\boldsymbol{x}}$, we obtain that $\|\boldsymbol{x}_k - \bar{\boldsymbol{x}}\|_2$ is also nonincreasing, whence the cluster point $\bar{\boldsymbol{x}}$ is a limit of the sequence $\{\boldsymbol{x}_k\}$. □

The key idea in the above proof is to use the optimality condition (8.2.32) for the proximal operator, together with the subgradient inequality to relate the suboptimality $g(\boldsymbol{x}_{k+1}) - g(\boldsymbol{x}_\star)$ in objective function to the distance to the feasible set. To prove Theorem 8.6, we follow a very similar program.

*Proof of Theorem 8.6* Notice that by construction, there exists $\boldsymbol{\gamma} \in \partial g(\boldsymbol{x}_{k+1})$ such that

$$\nabla f(\boldsymbol{x}_k) + L(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + \boldsymbol{\gamma} = \boldsymbol{0}. \qquad (8.2.38)$$

The subgradient and gradient inequalities for the convex functions $f$ and $g$ give that for any $\boldsymbol{x}$,

$$f(\boldsymbol{x}) \ge f(\boldsymbol{x}_k) + \langle \boldsymbol{x} - \boldsymbol{x}_k, \nabla f(\boldsymbol{x}_k)\rangle, \qquad (8.2.39)$$
$$g(\boldsymbol{x}) \ge g(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{x} - \boldsymbol{x}_{k+1}, \boldsymbol{\gamma}\rangle, \qquad (8.2.40)$$

whence

$$F(\boldsymbol{x}) \;\ge\; f(\boldsymbol{x}_k) + g(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{x} - \boldsymbol{x}_k, \nabla f(\boldsymbol{x}_k)\rangle + \langle \boldsymbol{x} - \boldsymbol{x}_{k+1}, \boldsymbol{\gamma}\rangle. \qquad (8.2.41)$$

Recall the definition of an upper bound $\hat{F}$ of $F$ defined (8.2.20). So, we have

$$\begin{aligned} F(\boldsymbol{x}) - F(\boldsymbol{x}_{k+1}) &\geq F(\boldsymbol{x}) - \hat{F}(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k) \\ &\geq f(\boldsymbol{x}_k) + g(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{x} - \boldsymbol{x}_k, \nabla f(\boldsymbol{x}_k) \rangle \\ &\quad + \langle \boldsymbol{x} - \boldsymbol{x}_{k+1}, \boldsymbol{\gamma} \rangle - \hat{F}(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k) \\ &= -\frac{L}{2} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 + \langle \boldsymbol{x} - \boldsymbol{x}_{k+1}, \nabla f(\boldsymbol{x}_k) + \boldsymbol{\gamma} \rangle \\ &= -\frac{L}{2} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 + L \langle \boldsymbol{x} - \boldsymbol{x}_{k+1}, \boldsymbol{x}_k - \boldsymbol{x}_{k+1} \rangle \\ &= \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{x}_{k+1}\|_2^2 - \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2 . \end{aligned} \tag{8.2.42}$$

Evaluating this expression at $\boldsymbol{x} = \boldsymbol{x}_\star$, we see that $\|\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2$ is nonincreasing. Moreover, rearranging the relationship and summing from 0 to $k-1$, we obtain that

$$\sum_{i=0}^{k-1} \{F(\boldsymbol{x}_{i+1}) - F(\boldsymbol{x}_\star)\} \;\leq\; \frac{L}{2} \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2 . \tag{8.2.43}$$

Evaluating (8.2.42) at $\boldsymbol{x} = \boldsymbol{x}_k$, we obtain

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{k+1}) \;\geq\; \frac{L}{2} \|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\|_2^2 . \tag{8.2.44}$$

Hence, (8.2.43) implies that

$$k \{F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star)\} \;\leq\; \frac{L}{2} \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2 . \tag{8.2.45}$$

Rearranging, we get the desired conclusion. □

## 8.3 Accelerated Proximal Gradient Methods

In the previous section, we have seen that by exploiting the fact that if the nonsmooth part of a convex function has an easily computable proximal operator, we are able to extend the gradient descent algorithm for smooth functions to the special class of composite objective functions that we encounter in structured signal recovery. The resulting algorithm enjoys the same $O(1/k)$ convergence rate as in the smooth case. Recognizing special structure in our problem of interest yields a significantly more accurate and efficient algorithm.

### 8.3.1 Acceleration via Nesterov's Method

With the taste of victory still on our lips, we might naturally ask whether further improvements are still possible – is our proximal gradient algorithm optimal for this class of functions? For the question to be meaningful, we need to restrict our attention to methods with efficient iterations, such as gradient-like methods. For example, we could restrict our attention to *first-order methods*, which base

their future actions on the past iterates $\boldsymbol{x}_0, \dots, \boldsymbol{x}_k$, objective values at these iterates, and the gradients $\nabla f(\boldsymbol{x}_0), \dots, \nabla f(\boldsymbol{x}_k)$. The corresponding question for *smooth functions* was studied in great depth in the late 1970's and early 1980's by Russian optimization theorists, including Polyak, Nesterov, Nemirovski, and Yudin.[5]

They asked the very natural question: *for minimizing a smooth function $f$, is the gradient method optimal amongst first-order methods?* Again, to study this problem one needs a model for computation. They considered a *black box* model, in which the algorithm produces a sequence of iterates $\boldsymbol{x}_0, \dots, \boldsymbol{x}_k$. At each iteration, the algorithm is provided with the value $f(\boldsymbol{x}_i)$ and the gradient $\nabla f(\boldsymbol{x}_i)$. It produces the next iterate as a function of the history of iterates, gradients, and function values up to this time:

$$\boldsymbol{x}_{k+1} = \varphi_k\big(\boldsymbol{x}_0, \dots, \boldsymbol{x}_k, f(\boldsymbol{x}_0), \dots, f(\boldsymbol{x}_k), \nabla f(\boldsymbol{x}_0), \dots, \nabla f(\boldsymbol{x}_k)\big). \tag{8.3.1}$$

With this model in mind, one can begin to study algorithms from a worst-case perspective. To do so, we fix a class of functions $\mathcal{F}$, and ask how well the algorithm does on the "worst function" from this class:

$$\sup_{f \in \mathcal{F},\, \boldsymbol{x}_0} \Big\{ f(\boldsymbol{x}_k) - \inf_{\boldsymbol{x}} f(\boldsymbol{x}) \Big\}. \tag{8.3.2}$$

One can study various classes of functions $\mathcal{F}$. However, for our purposes, one interesting class is the convex differentiable functions $f : \mathsf{B}(0, r) \to \mathbb{R}$, defined on a ball of radius $r$, with $L$-Lipschitz gradients:

$$\mathcal{F}_{L,r} \doteq \{ f : \mathsf{B}(0,r) \to \mathbb{R} \mid \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{x}')\|_2 \leq L \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \quad \forall\, \boldsymbol{x}, \boldsymbol{x}' \in \mathsf{B}(0,r) \}. \tag{8.3.3}$$

The gradient method achieves a rate of $O(1/k)$ over this class:

$$\sup_{f \in \mathcal{F}_{L,r},\, \boldsymbol{x}_0} \Big\{ f(\boldsymbol{x}_k) - \inf_{\boldsymbol{x}} f(\boldsymbol{x}) \Big\} \;\leq\; \frac{CLr^2}{k}, \tag{8.3.4}$$

where $C > 0$ is a constant. However, the best lower bound that anyone could prove was of much lower order:

$$\sup_{f \in \mathcal{F}_{L,r},\, \boldsymbol{x}_0} \Big\{ f(\boldsymbol{x}_k) - \inf_{\boldsymbol{x}} f(\boldsymbol{x}) \Big\} \;\geq\; \frac{cLr^2}{k^2}, \tag{8.3.5}$$

where $c > 0$ is some constant. Was this merely a gap in the theory? Or might there actually exist a "faster" gradient method than gradient descent itself?

In 1983, Yurii Nesterov closed this gap, to remarkable effect [Nes83]. He demonstrated a relatively simple first-order method, which achieved the optimal rate of convergence, $O(1/k^2)$. The analysis of this algorithm is straightforward to read – the 1983 paper is only five pages! However, it is not straightforward to build intuition into *how* the method achieves this rate. To gain some loose appreciation for what is going on, we start from a simpler idea.

[5] For a more comprehensive introduction to this circle of ideas, see [Nes03] or [Nem95].

Let us first consider the gradient descent method for a smooth function with Lipschitz gradients. At each iteration, the update simply follows the direction of the gradient:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k),$$

where a good choice of $\alpha$ for an $L$-Lipschitz function is $1/L$. The negative gradient $-\nabla f(\boldsymbol{x}_k)$ indicates the direction in which the function drops its value the fastest. Instead of updating along this most greedy direction at the current estimate $\boldsymbol{x}_k$, a slightly more conservative strategy is to update by keeping some momentum from the previous update direction: $\boldsymbol{x}_k - \boldsymbol{x}_{k-1}$. That leads an update rule that is known as the *heavy ball method* [Pol64]:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k) + \beta\big(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\big). \tag{8.3.6}$$

For properly chosen parameters, the heavy ball method can reduce oscillations in the trajectories and leads to faster convergence.

Like the heavy ball method, Nesterov's acceleration method uses a momentum step. It introduces an auxiliary point $\boldsymbol{p}_{k+1}$ of the form similar to that in the heavy ball method:

$$\boldsymbol{p}_{k+1} \doteq \boldsymbol{x}_k + \beta_{k+1}\big(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\big).$$

At each iteration, we move to this new point, and then descend from it:

$$\boldsymbol{x}_{k+1} = \boldsymbol{p}_{k+1} - \alpha \nabla f(\boldsymbol{p}_{k+1}). \tag{8.3.7}$$

The weights $\alpha = 1/L$ and $\{\beta_{k+1}\}$ are carefully chosen to achieve the optimal convergence rate:

$$t_1 = 1, \quad t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}, \quad \beta_{k+1} = \frac{t_k - 1}{t_{k+1}}. \tag{8.3.8}$$

These particular values come from the convergence analysis. One can rigorously show that with this update scheme, the resulting algorithm achieves the theoretically optimal convergence rate $O(1/k^2)$ for the class of smooth functions with Lipschitz gradient.

*Accelerating the* Proximal *Gradient Method.*
As we have seen in the previous section, convex programs that arise in the context of structured signal recovery are often of the composite form $F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$, where $f$ is a smooth term whose gradient is $L$-Lipschitz and $g(\boldsymbol{x})$ is convex but not necessarily smooth. In the proximal gradient method introduced in the previous section, we have seen that at the $k$-th iteration, the value of the objective function $F(\boldsymbol{x})$ can be upper-bounded by

$$\begin{aligned} \hat{F}(\boldsymbol{x}, \boldsymbol{x}_k) &\doteq f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x} - \boldsymbol{x}_k \rangle + \tfrac{L}{2}\|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2 + g(\boldsymbol{x}) \\ &\doteq \tfrac{L}{2}\|\boldsymbol{x} - \boldsymbol{w}_k\|_2^2 + g(\boldsymbol{x}) \ + \ \text{terms that do not depend on } \boldsymbol{x}, \end{aligned}$$

**Accelerated Proximal Gradient (APG)**

**Problem Class:**

$$\min_{\boldsymbol{x}} F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$$

$f, g : \mathbb{R}^n \to \mathbb{R}$ are convex, $\nabla f$ $L$-Lipschitz and $g$ nonsmooth.

**Basic Iteration:** set $\boldsymbol{x}_0 \in \mathbb{R}^n$, $\boldsymbol{p}_1 = \boldsymbol{x}_1 \leftarrow \boldsymbol{x}_0$, and $t_1 \leftarrow 1$.
Repeat for $k = 1, 2, \ldots, K$:

$$t_{k+1} \leftarrow \frac{1 + \sqrt{1 + 4t_k{}^2}}{2}, \quad \beta_{k+1} \leftarrow \frac{t_k - 1}{t_{k+1}}.$$
$$\boldsymbol{p}_{k+1} \leftarrow \boldsymbol{x}_k + \beta_{k+1}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}).$$
$$\boldsymbol{x}_{k+1} \leftarrow \mathrm{prox}_{g/L}\Big[\boldsymbol{p}_{k+1} - \frac{1}{L}\nabla f(\boldsymbol{p}_{k+1})\Big].$$

**Convergence Guarantee:**

$F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star)$ converges at a rate of $O(1/k^2)$.

**Figure 8.4** An overview of the Accelerated Proximal Gradient Method.

where $\boldsymbol{w}_k = \boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)$. As we have seen in the previous section, the gradient descent algorithm for the smooth part $f(\boldsymbol{x})$ corresponds to directly minimizing its quadratic approximation of $f(\boldsymbol{x})$ at $\boldsymbol{x}_k$.

In this language, if $g \equiv 0$, Nesterov's method corresponds to: *extrapolating* to find the point $\boldsymbol{p}_{k+1}$ based on the past two iterates, and then *minimizing* a quadratic upper bound $\hat{f}$ to $f$, taken at the new point $\boldsymbol{p}_{k+1}$, by taking a gradient step. Let us attempt to do the same thing for our composite function $\hat{F}$. Set

$$\boldsymbol{p}_{k+1} = \boldsymbol{x}_k + \beta_{k+1}\big(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\big), \tag{8.3.9}$$

instead of the current estimate $\boldsymbol{x}_k$. Then minimize $\hat{F}(\boldsymbol{x}, \boldsymbol{p}_{k+1})$ to obtain the next iterate $\boldsymbol{x}_{k+1}$:

$$\boldsymbol{x}_{k+1} = \mathrm{prox}_{g/L}\Big[\boldsymbol{p}_{k+1} - \frac{1}{L}\nabla f(\boldsymbol{p}_{k+1})\Big]. \tag{8.3.10}$$

We summarize this scheme in Figure 8.4. Theorem 8.9 establishes that with this simple modification to the proximal gradient method, the resulting new algorithm achieves the theoretically optimal convergence rate $O(1/k^2)$ for this class of methods, despite the presence of a nonsmooth term in the objective function.

THEOREM 8.9 (Convergence of Accelerated Proximal Gradient). *Let the sequence $\{\boldsymbol{x}_k\}$ be generated by the above accelerated proximal gradient scheme for the convex composite function $F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$, where the gradient of $f$ is $L$-*

*Lipschitz. Let $\boldsymbol{x}_\star$ be a minimizer of $F(\boldsymbol{x})$. Then for any $k \geq 1$,*

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star) \leq \frac{2L\|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2}{(k+1)^2}.$$

### 8.3.2 APG for Basis Pursuit Denoising

Applying the APG algorithm in Figure 8.4 to the basis pursuit denoising problem 8.1.2, we obtain Algorithm 8.3. This algorithm is also known as Fast Iterative Shrinkage-Thresholding Algorithm (FISTA), coined by Beck and Teboulle [BT09].

**Algorithm 8.3** Accelerated Proximal Gradient (APG) for BPDN

**Problem:** $\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$, given $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$.
**Input:** $\boldsymbol{x}_0 \in \mathbb{R}^n$, $\boldsymbol{p}_1 = \boldsymbol{x}_1 \leftarrow \boldsymbol{x}_0$, and $t_1 \leftarrow 1$, and $L \geq \lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})$.
**for** $(k = 1, 2, \ldots, K-1)$ **do**
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$; $\beta_{k+1} \leftarrow \frac{t_k - 1}{t_{k+1}}$.
  $\boldsymbol{p}_{k+1} \leftarrow \boldsymbol{x}_k + \beta_{k+1}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$.
  $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{p}_{k+1} - \frac{1}{L}\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{p}_{k+1} - \boldsymbol{y})$.
  $\boldsymbol{x}_{k+1} \leftarrow \mathrm{soft}(\boldsymbol{w}_{k+1}, \lambda/L)$.
**end for**
**Output:** $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

### 8.3.3 APG for Stable Principal Component Pursuit

Similarly we could apply the APG algorithm in Figure 8.4 to solving the stable principal component pursuit (PCP) problem 8.2.29. Again, notice that the APG scheme respects the natural separable structure of the objective function.

**Algorithm 8.4** Accelerated Proximal Gradient (APG) for Stable PCP

**Problem:** $\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 + \frac{\mu}{2}\|\boldsymbol{Y} - \boldsymbol{L} - \boldsymbol{S}\|_F^2$, given $\boldsymbol{Y}$, $\lambda, \mu > 0$.
**Input:** $\boldsymbol{L}_0 \in \mathbb{R}^{m\times n}$, $\boldsymbol{S}_0 \in \mathbb{R}^{m\times n}$, $\boldsymbol{P}_1^S = \boldsymbol{S}_1 \leftarrow \boldsymbol{S}_0$, $\boldsymbol{P}_1^L = \boldsymbol{L}_1 \leftarrow \boldsymbol{L}_0$, $t_1 \leftarrow 1$.
**for** $(k = 1, 2, \ldots, K-1)$ **do**
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$, $\beta_{k+1} \leftarrow \frac{t_k - 1}{t_{k+1}}$.
  $\boldsymbol{P}_{k+1}^L \leftarrow \boldsymbol{L}_k + \beta_{k+1}\big(\boldsymbol{L}_k - \boldsymbol{L}_{k-1}\big)$.
  $\boldsymbol{P}_{k+1}^S \leftarrow \boldsymbol{S}_k + \beta_{k+1}\big(\boldsymbol{S}_k - \boldsymbol{S}_{k-1}\big)$.
  $\boldsymbol{W}_{k+1} \leftarrow \boldsymbol{Y} - \boldsymbol{P}_{k+1}^S$ and compute the SVD: $\boldsymbol{W}_{k+1} = \boldsymbol{U}_{k+1}\boldsymbol{\Sigma}_{k+1}\boldsymbol{V}_{k+1}^*$.
  $\boldsymbol{L}_{k+1} \leftarrow \boldsymbol{U}_{k+1}\mathrm{soft}(\boldsymbol{\Sigma}_{k+1}, 1/\mu)\boldsymbol{V}_{k+1}^*$.
  $\boldsymbol{S}_{k+1} \leftarrow \mathrm{soft}((\boldsymbol{Y} - \boldsymbol{P}_{k+1}^L), \lambda/\mu)$.
**end for**
**Output:** $\boldsymbol{L}_\star \leftarrow \boldsymbol{L}_K; \boldsymbol{S}_\star \leftarrow \boldsymbol{S}_K$.

### 8.3.4 Convergence of APG

Our convergence analysis follows, almost verbatim, Beck and Teboulle [BT09]. Let $\varphi(\boldsymbol{y})$ denote the operator that takes a step in the direction of the gradient of $f$ at $\boldsymbol{y}$, and then applies the proximal operator of $g/L$:

$$\varphi(\boldsymbol{y}) = \operatorname{prox}_{g/L}\left[\boldsymbol{y} - \frac{1}{L}\nabla f(\boldsymbol{y})\right]. \tag{8.3.11}$$

In this language, the accelerated proximal gradient iteration is

$$\boldsymbol{x}_{k+1} = \varphi(\boldsymbol{p}_{k+1}). \tag{8.3.12}$$

The following lemma allows us to compare the value of $F(\boldsymbol{x})$ at any point $\boldsymbol{x}$ to the value at $\varphi(\boldsymbol{p})$ for an arbitrary point $\boldsymbol{p}$:

LEMMA 8.10. *For every $\boldsymbol{x}, \boldsymbol{p} \in \mathbb{R}^n$,*

$$F(\boldsymbol{x}) - F(\varphi(\boldsymbol{p})) \;\geq\; \frac{L}{2}\left\|\varphi(\boldsymbol{p}) - \boldsymbol{p}\right\|_2^2 + L\left\langle \boldsymbol{p} - \boldsymbol{x}, \varphi(\boldsymbol{p}) - \boldsymbol{p}\right\rangle. \tag{8.3.13}$$

*Proof* For it, we note that from the optimality condition for the proximal problem, $\boldsymbol{z} = \varphi(\boldsymbol{p})$ if and only if there exists $\boldsymbol{\gamma} \in \partial g(\boldsymbol{z})$ such that

$$\boldsymbol{\gamma} + L(\boldsymbol{z} - \boldsymbol{p}) + \nabla f(\boldsymbol{p}) = \boldsymbol{0}. \tag{8.3.14}$$

Using the subgradient inequalities for $f$ and $g$, we obtain that

$$f(\boldsymbol{x}) \geq f(\boldsymbol{p}) + \langle \boldsymbol{x} - \boldsymbol{p}, \nabla f(\boldsymbol{p})\rangle \tag{8.3.15}$$

$$g(\boldsymbol{x}) \geq g(\varphi(\boldsymbol{p})) + \left\langle \boldsymbol{x} - \varphi(\boldsymbol{p}), \boldsymbol{\gamma}\right\rangle. \tag{8.3.16}$$

Hence,

$$\begin{aligned}
F(\boldsymbol{x}) - F(\varphi(\boldsymbol{p})) &\geq f(\boldsymbol{p}) + g(\varphi(\boldsymbol{p})) + \langle \boldsymbol{x} - \boldsymbol{p}, \nabla f(\boldsymbol{p})\rangle \\
&\quad + \left\langle \boldsymbol{x} - \varphi(\boldsymbol{p}), \boldsymbol{\gamma}\right\rangle - F(\varphi(\boldsymbol{p})) \\
&\geq f(\boldsymbol{p}) + g(\varphi(\boldsymbol{p})) + \langle \boldsymbol{x} - \boldsymbol{p}, \nabla f(\boldsymbol{p})\rangle \\
&\quad + \left\langle \boldsymbol{x} - \varphi(\boldsymbol{p}), \boldsymbol{\gamma}\right\rangle - \hat{F}(\boldsymbol{\varphi}(\boldsymbol{p}), \boldsymbol{p}) \\
&= -\frac{L}{2}\left\|\varphi(\boldsymbol{p}) - \boldsymbol{p}\right\|_2^2 + \left\langle \boldsymbol{x} - \varphi(\boldsymbol{p}), \nabla f(\boldsymbol{p}) + \boldsymbol{\gamma}\right\rangle \\
&= -\frac{L}{2}\left\|\varphi(\boldsymbol{p}) - \boldsymbol{p}\right\|_2^2 + L\left\langle \boldsymbol{x} - \varphi(\boldsymbol{p}), \boldsymbol{p} - \varphi(\boldsymbol{p})\right\rangle \\
&= \frac{L}{2}\left\|\varphi(\boldsymbol{p}) - \boldsymbol{p}\right\|_2^2 + L\left\langle \boldsymbol{p} - \boldsymbol{x}, \varphi(\boldsymbol{p}) - \boldsymbol{p}\right\rangle,
\end{aligned} \tag{8.3.17}$$

as desired. □

Using Lemma 8.10, we obtain a relationship between the suboptimality in function values and the distance of an interpolated point to the optimum:

LEMMA 8.11. *Let $\{(\boldsymbol{x}_k, \boldsymbol{p}_k)\}$ be the sequence generated by the proximal gradient method. Set*

$$v_k = F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star) \tag{8.3.18}$$

*and*

$$\boldsymbol{u}_k = t_k \boldsymbol{x}_k - (t_k - 1)\boldsymbol{x}_{k-1} - \boldsymbol{x}_\star. \tag{8.3.19}$$

*Then*

$$\frac{2}{L} t_k^2 v_k - \frac{2}{L} t_{k+1}^2 v_{k+1} \;\geq\; \|\boldsymbol{u}_{k+1}\|_2^2 - \|\boldsymbol{u}_k\|_2^2 . \tag{8.3.20}$$

*Proof* Let us apply the previous lemma with $\boldsymbol{x} = \boldsymbol{x}_k$ and $\boldsymbol{p} = \boldsymbol{p}_{k+1}$. This gives

$$\frac{2}{L}\left(v_k - v_{k+1}\right) \;\geq\; \left\|\boldsymbol{x}_{k+1} - \boldsymbol{p}_{k+1}\right\|_2^2 + 2\left\langle \boldsymbol{p}_{k+1} - \boldsymbol{x}_k, \boldsymbol{x}_{k+1} - \boldsymbol{p}_{k+1}\right\rangle . \tag{8.3.21}$$

Applying the lemma with $\boldsymbol{x} = \boldsymbol{x}_\star$ and $\boldsymbol{p} = \boldsymbol{p}_{k+1}$, we get

$$-\frac{2}{L} v_{k+1} \;\geq\; \left\|\boldsymbol{x}_{k+1} - \boldsymbol{p}_{k+1}\right\|_2^2 + 2\left\langle \boldsymbol{p}_{k+1} - \boldsymbol{x}_\star, \boldsymbol{x}_{k+1} - \boldsymbol{p}_{k+1}\right\rangle . \tag{8.3.22}$$

Multiplying the first inequality by $t_{k+1} - 1$ and add that to the second inequality, we get

$$\begin{aligned}
&\frac{2}{L}\left((t_{k+1} - 1)v_k - t_{k+1} v_{k+1}\right) \\
\geq\, & t_{k+1}\left\|\boldsymbol{x}_{k+1} - \boldsymbol{p}_{k+1}\right\|_2^2 + 2\left\langle \boldsymbol{x}_{k+1} - \boldsymbol{p}_{k+1}, t_{k+1}\boldsymbol{p}_{k+1} - (t_{k+1} - 1)\boldsymbol{x}_k - \boldsymbol{x}_\star\right\rangle .
\end{aligned}$$

Multiplying both sides by $t_{k+1}$, and using that $t_k^2 = t_{k+1}(t_{k+1} - 1)$, we get

$$\begin{aligned}
&\frac{2}{L}\left(t_k^2 v_k - t_{k+1}^2 v_{k+1}\right) \\
\geq\, & \left\|t_{k+1}\left(\boldsymbol{x}_{k+1} - \boldsymbol{p}_{k+1}\right)\right\|_2^2 + 2t_{k+1}\left\langle \boldsymbol{x}_{k+1} - \boldsymbol{p}_{k+1}, t_{k+1}\boldsymbol{p}_{k+1} - (t_{k+1} - 1)\boldsymbol{x}_k - \boldsymbol{x}_\star\right\rangle \\
=\, & \|t_{k+1}\boldsymbol{x}_{k+1} - (t_{k+1} - 1)\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2^2 - \left\|t_{k+1}\boldsymbol{p}_{k+1} - (t_{k+1} - 1)\boldsymbol{x}_k - \boldsymbol{x}_\star\right\|_2^2 \\
=\, & \|\boldsymbol{u}_{k+1}\|_2^2 - \|\boldsymbol{u}_k\|_2^2 ,
\end{aligned}$$

where the last equality follows from plugging in $\boldsymbol{p}_{k+1} = \boldsymbol{x}_k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$, as per the APG algorithm. □

To prove the desired result, we note two simple facts, and then perform a calculation. First,

Lemma 8.12. *Let $\{(a_k, b_k)\}$ be sequences of positive real numbers satisfying*

$$a_k - a_{k+1} \;\geq\; b_{k+1} - b_k \quad \forall\, k, \qquad a_1 + b_1 \leq c. \tag{8.3.23}$$

*Then $a_k \leq c$ for every $k$.*

Second,

Lemma 8.13. *The sequence $\{t_k\}$ generated by the accelerated proximal gradient method satisfies*

$$t_k \;\geq\; \frac{k+1}{2} \quad \forall\, k \geq 1. \tag{8.3.24}$$

With these facts in mind, we prove Theorem 8.9.

*Proof of Theorem 8.9* Define

$$a_k \doteq \frac{2}{L} t_k^2 v_k, \quad b_k \doteq \|\boldsymbol{u}_k\|_2^2, \quad c \doteq \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2. \tag{8.3.25}$$

By Lemma 8.11, for every $k$,

$$a_k - a_{k+1} \geq b_{k+1} - b_k \tag{8.3.26}$$

so, provided $a_1 + b_1 \leq c$, we obtain that $a_k \leq c$ for every $k$, whence

$$\frac{2}{L} t_k^2 v_k \;\leq\; \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2. \tag{8.3.27}$$

Since $t_k \geq (k+1)/2$, this gives

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star) \;\leq\; \frac{2L\,\|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2}{(k+1)^2}. \tag{8.3.28}$$

So, it just remains to check that $a_1 + b_1 \leq c$. Since $t_1 = 1$, $a_1 = \frac{2}{L} v_1$, while $b_1 = \|\boldsymbol{x}_1 - \boldsymbol{x}_\star\|_2^2$. In Lemma 8.10, set $\boldsymbol{x} = \boldsymbol{x}_\star$, $\boldsymbol{p} = \boldsymbol{p}_1$, to obtain

$$F(\boldsymbol{x}_\star) - F(\boldsymbol{x}_1) \geq \frac{L}{2}\,\|\boldsymbol{x}_1 - \boldsymbol{p}_1\|_2^2 + L\,\langle \boldsymbol{p}_1 - \boldsymbol{x}_\star, \boldsymbol{x}_1 - \boldsymbol{p}_1 \rangle \tag{8.3.29}$$

$$= \frac{L}{2}\left(\|\boldsymbol{x}_1 - \boldsymbol{x}_\star\|_2^2 - \|\boldsymbol{p}_1 - \boldsymbol{x}_\star\|_2^2\right). \tag{8.3.30}$$

Since $\|\boldsymbol{p}_1 - \boldsymbol{x}_\star\|_2^2 = \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^2$, this gives the result. □

### 8.3.5 Further Developments on Acceleration

Acceleration is a surprising phenomenon for gradient-based (hence first-order) optimization methods. The previous subsection merely introduced the very basic concept and technique of acceleration, probably just enough for practitioners to apply such methods to the low-dimensional model estimation problems. As noted before, our derivation relies on techniques of Beck and Teboulle [BT09], also known as *momentum analysis*. This is actually different from the original construction by Nesterov based on *estimation sequences*. For a more detailed description of the origin of acceleration techniques, one may refer to the classic book "*Introductory Lectures on Convex Programming: A Basic Course*" by Nesterov [Nes03].

Nesterov's original construction by estimation sequence is often viewed as an algebra trick and thus is difficult to comprehend. Considering the great significance and impact of acceleration methods in modern large-scale optimization problems (that arise in compressive sensing and machine learning), it is of interest to explain the acceleration phenomenon in a more intuitive way so that one could potentially design acceleration methods in a more principled way or for broader classes of problems. To this end, there has been an increasing interest to explain acceleration from the perspective of *continuous dynamics*.

It is widely known that (non-accelerated) gradient descent can be viewed as

discretization of a first-order ordinary differential equation (ODE) associated with the gradient flow. Recently, the work of [SBC14] (and many subsequent work [KBB16, KBB15, WWJ16], etc.) have shown Nesterov's accelerated gradient descent can be explained as the discretization of a second-order ODE. Similar idea of speeding up iterative methods via discretizing second-order ODE can be traced back to the work of Polyak [Pol64] in the 1960's. Because of the simplicity of continuous-time dynamics, such a point of view provides a good explanation of the inner mechanism of acceleration. However, to obtain actual iterative algorithms, such formulation requires proper discretization which is often not a trivial problem.

Meanwhile, the recent work of "*Approximate Duality Gap Technique*" (ADGT) [DO19] provides a new framework that revisits Nesterov's original estimation sequence construction from a continuous-time perspective. Within this framework, in continuous time, the estimation sequence can be viewed as a way to constructing more precise lower and upper bounds for the difference between the optimal value and the current value by exploiting the convexity of the objective function; while in discrete time, the upper bound constructed by ADGT will incur a discretization error, which then can be canceled out by exploiting the smoothness property of objective function (e.g., by gradient descent). As the duality gap becomes more precise, the optimal accelerated convergence rate can be guaranteed.

These recent developments have enriched our understanding of Nesterov's acceleration method. Nevertheless, as we have noted before, as far as first-order methods are concerned, the Nesterov' construction has reached the optimal iteration complexity $O(1/k^2)$ for this class of methods. To achieve better iteration complexity, one must resort to high-order information. Somewhat surprisingly, the ADGT framework does allow us to further generalize acceleration techniques to *high-order* settings and lead to accelerated algorithms that can achieve the optimal iteration complexity [SJM19].

## 8.4 Augmented Lagrange Multipliers

So far, we have described how to solve certain classes of *unconstrained* convex optimization problems arising in structured signal recovery. However, in some scenarios – e.g., if the noise level is low, or if the target solution $\boldsymbol{x}$ is known ahead of time to possess additional application-specific structure – it may be desirable to exactly enforce equality constraints such as in the exact BP program (8.1.1) or the PCP (8.1.5).

In this section, we describe a framework for solving *equality constrained* problems of the form

$$\begin{array}{ll} \min_{\boldsymbol{x}} & g(\boldsymbol{x}), \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \end{array} \tag{8.4.1}$$

where $g : \mathbb{R}^n \to \mathbb{R}$ is a convex function, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is a matrix and $\boldsymbol{y} \in \text{range}(\boldsymbol{A})$

(so that the problem is feasible). One very intuitive approach to producing an approximate solution to (8.4.1) is to simply replace the equality constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ with a penalty function $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$, and solve the unconstrained problem

$$\min_{\boldsymbol{x}} \; g(\boldsymbol{x}) + \tfrac{\mu}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \tag{8.4.2}$$

for a very large value of $\mu$. As $\mu$ increases to $+\infty$, the solution set of this problem approaches the solution set of equality constrained problem (8.4.1). This is known as the *penalty method* in the optimization literature, and has a long history, with many variants. Its main advantage is that it leaves us with a simpler unconstrained problem, to which we can directly apply scalable first-order methods such as the proximal gradient methods of the previous two sections.

However, there is a serious drawback to this approach. For first-order methods such as PG and APG, the rate of convergence is dictated by how quickly the gradient $\nabla(\mu f) = \mu\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})$ can change from point-to-point, which is measured through the Lipschitz constant

$$L_{\nabla \mu f} = \mu\|\boldsymbol{A}\|_2^2 .$$

This increases linearly with $\mu$: *The larger $\mu$ is, the harder the unconstrained problem* (8.4.2) *is to solve!* One practical approach to mitigating this effect is to solve a sequence of unconstrained problems, with increasing $\mu$, and use the solution to each as an initial guess for the next. This *continuation* technique is often valuable in practice. Nevertheless, it suffers from the same drawback: as $\mu$ increases, accurate solutions become increasingly difficult to obtain.

Lagrange duality gives a more principled mechanism for studying and solving the constrained problem (8.4.1) via solving unconstrained problems. In particular, it will give us a mechanism for exactly solving the constrained problem (8.4.1) by solving a sequence of unconstrained problems *whose difficulty does not increase*. The central object in Lagrange duality is the Lagrangian

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \doteq g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\rangle , \tag{8.4.3}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is a vector of *Lagrange multipliers* corresponding to the equality constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$. In particular, we can characterize optimal solutions $(\boldsymbol{x}, \boldsymbol{\lambda})$ as saddle points of the Lagrangian

$$\sup_{\boldsymbol{\lambda}} \inf_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \sup_{\boldsymbol{\lambda}} \inf_{\boldsymbol{x}} \; g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\rangle . \tag{8.4.4}$$

If we define the dual function

$$d(\boldsymbol{\lambda}) \doteq \inf_{\boldsymbol{x}} \; g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\rangle , \tag{8.4.5}$$

then the saddle point characterization of optimal solutions suggests a natural computational approach to finding $(\boldsymbol{x}, \boldsymbol{\lambda})$:

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \; \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_k), \tag{8.4.6}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + t_{k+1}(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y}). \tag{8.4.7}$$

It is not difficult to show that $\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y}$ is a subgradient[6] of the dual function $d(\boldsymbol{\lambda})$. This iteration corresponds to a subgradient ascent algorithm for maximizing the dual function, and hence is called *dual ascent*. In (8.4.7), $t_{k+1}$ is the step size. For certain problem classes, dual ascent yields efficient, convergent algorithms, which produce an optimal primal-dual pair $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$. However, for problems arising in structured signal recovery, the straightforward iteration (8.4.6)-(8.4.7) may fail:

EXAMPLE 8.14. *Show that*

$$\inf_{\boldsymbol{x}} \ \|\boldsymbol{x}\|_1 + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle \ = \ \begin{cases} -\infty & \|\boldsymbol{A}^*\boldsymbol{\lambda}\|_\infty > 1, \\ -\langle \boldsymbol{\lambda}, \boldsymbol{y} \rangle & \|\boldsymbol{A}^*\boldsymbol{\lambda}\|_\infty \leq 1. \end{cases} \tag{8.4.8}$$

*So, for basis pursuit, if the dual ascent step* (8.4.7) *happens to produce a* $\boldsymbol{\lambda}$ *such that* $\|\boldsymbol{A}^*\boldsymbol{\lambda}\|_\infty > 1$*, the algorithm will break down. Notice, in particular, that when* $\|\boldsymbol{A}^*\boldsymbol{\lambda}\|_\infty > 1$*, we can produce arbitrarily large (in magnitude) negative values of* $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ *by choosing* $\boldsymbol{x}$ *far away from the feasible set* $\{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}\}$.

This bad behavior occurs more generally. Thus, for structured signal recovery, the classical Lagrangian is sufficient for characterizing optimality conditions, but it does not penalize the equality $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ strongly enough for (8.4.6)-(8.4.7) to lead to a useful algorithm. A natural remedy is to *augment* the Lagrangian with an extra penalty term, by introducing the function

$$\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{\lambda}) \doteq g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle + \frac{\mu}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 . \tag{8.4.9}$$

This function is known as the *augmented Lagrangian* [Hes69, Roc73, Pow69]. As before, $\mu > 0$ is a penalty parameter. The augmented Lagrangian can be regarded as the Lagrangian for the constrained problem

$$\begin{array}{ll} \min & g(\boldsymbol{x}) + \frac{\mu}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}. \end{array} \tag{8.4.10}$$

Since the penalty term $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$ is zero for all feasible $\boldsymbol{x}$, the optimal solutions of this problem coincide with the optimal solutions of the original problem (8.4.1).

Despite this formal equivalence, augmentation has dramatic consequences for numerical optimization. In particular, it can render the dual ascent iteration provably convergent, under very weak assumptions on the objective function $g$. To achieve this, we apply dual ascent to the regularized problem (8.4.10), and make a very particular choice of step size, $t_{k+1} = \mu$, yielding the iteration

$$\boldsymbol{x}_{k+1} \in \arg\min_{\boldsymbol{x}} \ \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{\lambda}_k), \tag{8.4.11}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \mu \, (\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y}). \tag{8.4.12}$$

This iteration, with the particular choice $t_{k+1} = \mu$ is known as the *Method of*

[6] Strictly, a *supergradient* since the dual $d(\boldsymbol{\lambda})$ is concave.

*Multipliers.* The update step (8.4.11) for $\boldsymbol{x}$ is a convex optimization problem itself and can typically be solved via the proximal gradient methods introduced in the previous sections.

REMARK 8.15. *The choice $t_{k+1} = \mu$ is important, because it allows us to avoid the breakdown described in Example 8.14: To see this, since $\boldsymbol{x}_{k+1}$ minimizes the convex function $\mathcal{L}_\mu$,*

$$\begin{aligned}\mathbf{0} &\in \partial \mathcal{L}_\mu(\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_k),\\ &= \partial g(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^*\boldsymbol{\lambda}_k + \mu \boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y}),\\ &= \partial g(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^*\boldsymbol{\lambda}_{k+1},\\ &= \partial \mathcal{L}(\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}).\end{aligned}$$

*Thus, $\boldsymbol{x}_{k+1}$ minimizes the unaugmented Lagrangian $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_{k+1})$ with $\boldsymbol{\lambda} = \boldsymbol{\lambda}_{k+1}$ fixed. This means that $d(\boldsymbol{\lambda}_{k+1}) > -\infty$, and $\boldsymbol{\lambda}_{k+1}$ is* dual feasible *for the original problem. In particular, $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_{k+1})$ is bounded below. Because $\boldsymbol{\lambda}_{k+1}$ is always dual feasible, the bad behavior in Example 8.14 cannot occur.*

Under appropriate assumptions on $g$, the iterates $\boldsymbol{x}_k$ produced by this modified algorithm converge to an optimal solution $\boldsymbol{x}_\star$ to the constrained problem (8.4.1). We state a slightly more general result that allows the penalty parameters $\mu$ to vary from iteration to iteration, as long as they remain bounded away from zero:

THEOREM 8.16 (Convergence of Augmented Lagrangian). *Let $g : \mathbb{R}^n \to \mathbb{R}$ be a convex, coercive function,*[7] *$\boldsymbol{A} \in \mathbb{R}^{m\times n}$ an arbitrary matrix, and $\boldsymbol{y} \in \operatorname{range}(\boldsymbol{A})$. Then the problem*

$$\begin{aligned}&\min_{\boldsymbol{x}} && g(\boldsymbol{x})\\ &\text{subject to} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y},\end{aligned} \tag{8.4.13}$$

*has at least one optimal solution. Moreover, the ALM iteration*

$$\boldsymbol{x}_{k+1} \in \arg\min_{\boldsymbol{x}} \ \mathcal{L}_{\mu_k}(\boldsymbol{x}, \boldsymbol{\lambda}_k), \tag{8.4.14}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \mu_k\,(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y}). \tag{8.4.15}$$

*with sequence $\{\mu_k\}$ bounded away from zero produces a sequence $\{\boldsymbol{\lambda}_k\}$ that converges to a dual optimal solution of the rate $O(1/k)$. Moreover, every limit point of the sequence $\{\boldsymbol{x}_k\}$ is optimal for (8.4.13).*

Figure 8.5 summarizes our general observations on the ALM method up to this point.

REMARK 8.17 (More general convergence theorems). *The statement of Theorem 8.16 represents a deliberate tradeoff between simplicity and generality. With somewhat more technical analysis, it is possible to show convergence of ALM for much more general classes of $g$. The most practically important extension allows $g$ to be an extended real-valued function (a function from $\mathbb{R}^n$ to $\mathbb{R} \cup \{+\infty\}$). For*

[7] A function $g(\boldsymbol{x})$ is said to be coercive if $\lim_{\|\boldsymbol{x}\|\to\infty} g(\boldsymbol{x}) = +\infty$.

**Augmented Lagrange Multipler (ALM)**

**Problem Class:**

$$\begin{array}{ll} \min_{\boldsymbol{x}} & g(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}. \end{array}$$

$g : \mathbb{R}^n \to \mathbb{R}$ convex, $\boldsymbol{y} \in \text{range}(\boldsymbol{A})$.

**Basic Iteration:** set

$$\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{\lambda}) = g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle + \tfrac{\mu}{2} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \right\|_2^2 .$$

Repeat:

$$\begin{aligned} \boldsymbol{x}_{k+1} &\in \arg\min_{\boldsymbol{x}} \ \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{\lambda}_k), \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \mu \left( \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y} \right). \end{aligned}$$

**Convergence Guarantee:**

If $g$ is coercive, every limit point of $\{\boldsymbol{x}_k\}$ is optimal.

**Figure 8.5** An overview of the Augmented Lagrangian Method of Multipliers.

*example, if we wish to optimize a real-valued convex function $g_0$ over the set of $\boldsymbol{x}$ that satisfy the equality constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$,* and *reside in some additional (nonempty closed, convex) constraint set $\mathsf{C}$:*

$$\begin{array}{ll} \min_{\boldsymbol{x}} & g_0(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \ \boldsymbol{x} \in \mathsf{C}, \end{array} \tag{8.4.16}$$

*we can apply ALM to the problem*

$$\begin{array}{ll} \min_{\boldsymbol{x}} & g(\boldsymbol{x}) \doteq g_0(\boldsymbol{x}) + I_{\boldsymbol{x} \in \mathsf{C}} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \end{array} \tag{8.4.17}$$

*where $I_{\boldsymbol{x} \in \mathsf{C}}$ is the indicator function for $\mathsf{C}$:*

$$I_{\boldsymbol{x} \in \mathsf{C}} = \begin{cases} 0 & \boldsymbol{x} \in \mathsf{C}, \\ +\infty & \boldsymbol{x} \notin \mathsf{C}. \end{cases} \tag{8.4.18}$$

*The survey of Eckstein [Eck12] and the monograph of Bertsekas [Ber82] are good introductory points for the more general theory, which enables such modifications.*

*Implementation Considerations.*
The most important practical consideration is how to choose the sequence of penalty parameters $\{\mu_k\}$. As discussed above, this choice induces a tradeoff between the cost of solving subproblems and the overall number of outer iterations

– larger $\mu$ leaves us with fewer outer iterations, but harder subproblems. A typical strategy is to increase $\mu$ geometrically, up to some pre-fixed ceiling:

$$\mu_k = \min\left\{\beta\mu_k, \mu_{\max}\right\},$$

where $\beta \approx 1.25$ is typical. The ceiling $\mu_{\max}$ is strongly problem dependent; choosing it "optimally" is something of a black art.

Our description and analysis of ALM assume that each of the subproblems is solved exactly. However, practically speaking, it may not be necessary to obtain high-accuracy solutions to the subproblems, especially in the early iterations. This can be justified theoretically. The choice of iterative method for solving the unconstrained subproblems is largely problem dependent. However, because the penalty term is quadratic, for many problems of interest in this book, the subproblems have composite form, and the APG algorithm applies.

In using APG (or any other iterative solver) to solve the unconstrained subproblems, it is highly advisable to use the previous iterate $\boldsymbol{x}_k$ as an initialization to solve for the subsequent iterate $\boldsymbol{x}_{k+1}$. While the subproblems are convex, and the global optimality of iterative algorithms does not depend on initialization, choosing an appropriate initializer can drastically reduce the overall number of iterations.

### 8.4.1 ALM for Basis Pursuit

We may apply ALM to the exact BP problem (8.1.1), which we summarize as Algorithm 8.5. This algorithm was introduced by [YOGD08], where it was interpreted as a *Bregman iteration*.

**Algorithm 8.5** Augmented Lagrange Multipler (ALM) for BP

**Problem:** $\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1$ subject to $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, given $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{A} \in \mathbb{R}^{m\times n}$.
**Input:** $\boldsymbol{x}_0 \in \mathbb{R}^n$, $\boldsymbol{\lambda}_0 \in \mathbb{R}^m$, and $\beta > 1$.
**for** $(k = 0, 1, 2, \ldots, K-1)$ **do**
  $\boldsymbol{x}_{k+1} \leftarrow \arg\min \mathcal{L}_{\mu_k}(\boldsymbol{x}, \boldsymbol{\lambda}_k)$ using APG.
  $\boldsymbol{\lambda}_{k+1} \leftarrow \boldsymbol{\lambda}_k + \mu_k(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y})$.
  $\mu_{k+1} \leftarrow \min\{\beta\mu_k, \mu_{\max}\}$.
**end for**
**Output:** $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

### 8.4.2 ALM for Principal Component Pursuit

In Chapter 4 Section 4.4, we have presented an important application of ALM algorithm, that is to solve the low-rank matrix completion (MC) problem (Algorithm 4.1).

We here (and the section below) discuss how to extend it to the more challenging low-rank and sparse matrix decomposition problem studied in Chapter 5. We recall principal component pursuit (PCP) (5.2.2) proposed in Chapter 5 solves the following program:

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \boldsymbol{L}+\boldsymbol{S}=\boldsymbol{Y}. \tag{8.4.19}$$

First, we rewrite the above program as a standard ALM objective function:

$$\mathcal{L}_\mu(\boldsymbol{L},\boldsymbol{S},\boldsymbol{\Lambda}) \;\doteq\; \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 + \langle \boldsymbol{\Lambda}, \boldsymbol{L}+\boldsymbol{S}-\boldsymbol{Y}\rangle + \tfrac{\mu}{2}\|\boldsymbol{L}+\boldsymbol{S}-\boldsymbol{Y}\|_F^2,$$

where $\mathcal{L}_\mu(\cdot)$ consists of a Lagrangian term with a Lagrange multiplier matrix $\boldsymbol{\Lambda}$ of the same size as $\boldsymbol{Y}$ and an augmented quadratic term that encourages the equality condition $\boldsymbol{L}+\boldsymbol{S}=\boldsymbol{Y}$. The ALM algorithm for this problem is summarized in Algorithm 8.6. However, in the step 4 of the algorithm, one is required to solve $\min_{\boldsymbol{L},\boldsymbol{S}} \mathcal{L}_{\mu_k}(\boldsymbol{L},\boldsymbol{S},\boldsymbol{\Lambda}_k)$. Unfortunately, there is no closed-form solution for the proximal operator for the nuclear norm and $\ell^1$ norm combined. We will address this difficulty in the next section with an alternating direction method.

**Algorithm 8.6** Augmented Lagrange Multipler (ALM) for PCP

1: **Problem:** $\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1$ subj to $\boldsymbol{L}+\boldsymbol{S}=\boldsymbol{Y}$, given $\boldsymbol{Y}$ and $\lambda>0$.
2: **Input:** $\boldsymbol{L}_0, \boldsymbol{S}_0, \boldsymbol{\Lambda}_0 \in \mathbb{R}^{m\times n}$ and $\beta > 1$.
3: **for** $(k = 0, 1, 2, \ldots, K-1)$ **do**
4: $\quad \{\boldsymbol{L}_{k+1}, \boldsymbol{S}_{k+1}\} \leftarrow \arg\min \mathcal{L}_{\mu_k}(\boldsymbol{L},\boldsymbol{S},\boldsymbol{\Lambda}_k)$ using APG.
5: $\quad \boldsymbol{\Lambda}_{k+1} \leftarrow \boldsymbol{\Lambda}_k + \mu_k(\boldsymbol{L}_{k+1}+\boldsymbol{S}_{k+1}-\boldsymbol{Y})$.
6: $\quad \mu_{k+1} \leftarrow \min\{\beta\mu_k, \mu_{\max}\}$.
7: **end for**
8: **Output:** $\boldsymbol{L}_\star \leftarrow \boldsymbol{L}_K, \boldsymbol{S}_\star \leftarrow \boldsymbol{S}_K$.

### 8.4.3 Convergence of ALM

In this subsection, we prove Theorem 8.16. The proof will actually reveal another interpretation of the method of Augmented Lagrangian, as an application of the proximal point algorithm to the dual problem.

*Proof* Let $d(\boldsymbol{\lambda})$ denote the dual function

$$d(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x}} g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\rangle\,. \tag{8.4.20}$$

The dual function is concave, and so its negative

$$q(\boldsymbol{\lambda}) = -d(\boldsymbol{\lambda}) \tag{8.4.21}$$

is convex.

Note that for any $\boldsymbol{\lambda}$,

$$\begin{aligned} d(\boldsymbol{\lambda}) &\leq g(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y} \rangle \\ &= g(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{\lambda}_{k+1}, \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y} \rangle + \langle \boldsymbol{\lambda} - \boldsymbol{\lambda}_{k+1}, \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y} \rangle . \end{aligned} \tag{8.4.22}$$

Now recall from Remark 8.15 that the augmented Lagrangian method ensures that $\boldsymbol{x}_{k+1}$ minimizes the unaugmented Lagrangian $g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle$ with $\boldsymbol{\lambda} = \boldsymbol{\lambda}_{k+1}$ fixed. Hence, by definition of the function $d(\boldsymbol{\lambda})$, we have $d(\boldsymbol{\lambda}_{k+1}) = g(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{\lambda}_{k+1}, \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y} \rangle$. Applying this to the above inequality, we obtain

$$d(\boldsymbol{\lambda}) \leq d(\boldsymbol{\lambda}_{k+1}) + \langle \boldsymbol{\lambda} - \boldsymbol{\lambda}_{k+1}, \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y} \rangle . \tag{8.4.23}$$

As $q(\boldsymbol{\lambda}) = -d(\boldsymbol{\lambda})$, we have

$$q(\boldsymbol{\lambda}) \geq q(\boldsymbol{\lambda}_{k+1}) + \langle \boldsymbol{\lambda} - \boldsymbol{\lambda}_{k+1}, \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{k+1} \rangle . \tag{8.4.24}$$

Hence $\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{k+1}$ is in the subgradient of $q(\cdot)$ at $\boldsymbol{\lambda}_{k+1}$, and

$$\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k+1} = \mu_k(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{k+1}) \in \mu_k \partial q(\boldsymbol{\lambda}_{k+1}), \tag{8.4.25}$$

and so

$$\boldsymbol{\lambda}_{k+1} = \operatorname{prox}_{\mu_k q}[\boldsymbol{\lambda}_k]. \tag{8.4.26}$$

Thus, dual ascent corresponds to the proximal point iteration applied to $q(\cdot)$. Under our assumptions, the dual optimal value $\sup_{\boldsymbol{\lambda}} d(\boldsymbol{\lambda}) > -\infty$ is finite, hence a dual optimal solution $\bar{\boldsymbol{\lambda}}$ exists. Proposition 8.8 then implies that $\boldsymbol{\lambda}_k \to \boldsymbol{\lambda}_\star$, where $\boldsymbol{\lambda}_\star$ is some dual optimal point. This and the fact that $\mu_k$ is bounded away from zero give that

$$\|\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}\|_2 = \frac{\|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1}\|_2}{\mu_k} \to 0, \tag{8.4.27}$$

and so the sequence $\{\boldsymbol{x}_k\}$ approaches the feasible set. The sequence $\{\boldsymbol{\lambda}_k\}$ inherits the same convergence rate as the proximal gradient method. Hence according to Proposition 8.8, the rate of convergence is at least $O(1/k)$ say $\mu_k > \mu_o$ for some $\mu_o > 0$.

From coercivity of $g$, there exists at least one primal optimal solution $\boldsymbol{x}_\star$. By optimality of $\boldsymbol{x}_{k+1}$, we have

$$g(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{\lambda}_k, \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y} \rangle + \frac{\mu}{2} \|\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y}\|_2^2 \leq g(\boldsymbol{x}_\star). \tag{8.4.28}$$

For any cluster point $\bar{\boldsymbol{x}}$, continuity of $g$ and $\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y} \to \boldsymbol{0}$ imply that $g(\bar{\boldsymbol{x}}) \leq g(\boldsymbol{x}_\star)$, whence $g(\bar{\boldsymbol{x}}) = g(\boldsymbol{x}_\star)$. Hence, every cluster point is optimal. □

## 8.5 Alternating Direction Method of Multipliers

The previous section showed how the augmented Lagrangian method (ALM) could be used to solve equality constrained convex optimization problems, by reducing them to a sequence of unconstrained subproblems. These subproblems

may still be challenging optimization problems if we need to minimize against all the variables simultaneously, as in step 4 of the Algorithm 8.6. In many situations, though, it is possible to exploit special separable structures of the objective function and to alleviate the difficulty by reducing the overall optimization to multiple subproblems of smaller sizes, as the following example shows.

EXAMPLE 8.18 (Principal Component Pursuit). *We solve*

$$\begin{array}{ll} \min_{\boldsymbol{L},\boldsymbol{S}} & \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \\ \text{subject to} & \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}. \end{array} \tag{8.5.1}$$

*The objective function is* separable *into two terms,* $\|\cdot\|_*$ *and* $\|\cdot\|_1$*, each of which has an efficient proximal operator.*

In this section, we study a family of augmented Lagrangian algorithms that can exploit this special, separable structure. We begin by treating a generic problem of the form

$$\begin{array}{ll} \min_{\boldsymbol{x},\boldsymbol{z}} & g(\boldsymbol{x}) + h(\boldsymbol{z}) \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{y}, \end{array} \tag{8.5.2}$$

where $g$ and $h$ are convex functions, $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrices, and $\boldsymbol{y} \in \text{range}([\boldsymbol{A} \mid \boldsymbol{B}])$. The Lagrangian $\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda})$ associated with this problem simply is:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = g(\boldsymbol{x}) + h(\boldsymbol{z}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y} \rangle . \tag{8.5.3}$$

As in the previous section, we form the augmented Lagrangian $\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda})$ associated with this problem:

$$\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = g(\boldsymbol{x}) + h(\boldsymbol{z}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y} \rangle + \frac{\mu}{2} \|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y}\|_2^2 . \tag{8.5.4}$$

In many applications, including the examples listed above, it is easy to minimize $\mathcal{L}_\mu$ with respect to $\boldsymbol{x}$, when $\boldsymbol{\lambda}$ and $\boldsymbol{z}$ are fixed, and also easy to minimize it with respect to $\boldsymbol{z}$ when $\boldsymbol{\lambda}$ and $\boldsymbol{x}$ are fixed. This suggests a simple, alternating iteration

$$\boldsymbol{z}_{k+1} \in \arg\min_{\boldsymbol{z}} \mathcal{L}_\mu(\boldsymbol{x}_k, \boldsymbol{z}, \boldsymbol{\lambda}_k), \tag{8.5.5}$$

$$\boldsymbol{x}_{k+1} \in \arg\min_{\boldsymbol{x}} \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}_{k+1}, \boldsymbol{\lambda}_k), \tag{8.5.6}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \mu \left(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_{k+1} - \boldsymbol{y}\right). \tag{8.5.7}$$

This is known as the *alternating directions method of multipliers* (ADMM). In the numerical analysis literature, this style of updating is referred to as a *Gauss-Seidel iteration*. We recommend [BPC+11] for a friendly introduction to these methods, as well as useful recommendations on stopping criteria, parameter setting, etc.

### 8.5.1 ADMM for Principal Component Pursuit

When applied to the principal component pursuit program (8.4.19), the ADMM iteration takes on a particularly simple form. Here, the two groups of variables are

the unknown low-rank matrix $\boldsymbol{L}$ and the unknown sparse error $\boldsymbol{S}$. The augmented Lagrangian is

$$\mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}) = \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 + \langle \boldsymbol{\Lambda}, \boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y} \rangle + \frac{\mu}{2} \|\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y}\|_F^2 . \quad (8.5.8)$$

The ADMM iteration sequentially updates $\boldsymbol{L}$, then $\boldsymbol{S}$, then $\boldsymbol{\Lambda}$. Each of these updates has a very simple familiar form. For example,

$$\begin{aligned}
\boldsymbol{L}_{k+1} &= \arg\min_{\boldsymbol{L}} \mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}_k, \boldsymbol{\Lambda}_k) \\
&= \arg\min_{\boldsymbol{L}} \|\boldsymbol{L}\|_* + \langle \boldsymbol{\Lambda}_k, \boldsymbol{L} + \boldsymbol{S}_k - \boldsymbol{Y} \rangle + \frac{\mu}{2} \|\boldsymbol{L} + \boldsymbol{S}_k - \boldsymbol{Y}\|_F^2 \\
&= \arg\min_{\boldsymbol{L}} \|\boldsymbol{L}\|_* + \frac{\mu}{2} \left\|\boldsymbol{L} + \boldsymbol{S}_k - \boldsymbol{Y} + \mu^{-1}\boldsymbol{\Lambda}_k\right\|_F^2 + \varphi(\boldsymbol{S}_k, \boldsymbol{\Lambda}_k) \\
&= \mathrm{prox}_{\mu^{-1}\|\cdot\|_*} \left[\boldsymbol{Y} - \boldsymbol{S}_k - \mu^{-1}\boldsymbol{\Lambda}_k\right] . \qquad (8.5.9)
\end{aligned}$$

Thus, the update step for the low-rank term can be evaluated simply by computing the proximal operator for the nuclear norm.

A similar simple rule can be derived for the sparse term:

$$\begin{aligned}
\boldsymbol{S}_{k+1} &= \arg\min_{\boldsymbol{S}} \mathcal{L}_\mu(\boldsymbol{L}_{k+1}, \boldsymbol{S}, \boldsymbol{\Lambda}_k) \\
&= \arg\min_{\boldsymbol{S}} \lambda \|\boldsymbol{S}\|_1 + \langle \boldsymbol{\Lambda}_k, \boldsymbol{L}_{k+1} + \boldsymbol{S} - \boldsymbol{Y} \rangle + \frac{\mu}{2} \|\boldsymbol{L}_{k+1} + \boldsymbol{S} - \boldsymbol{Y}\|_F^2 \\
&= \arg\min_{\boldsymbol{S}} \lambda \|\boldsymbol{S}\|_1 + \frac{\mu}{2} \left\|\boldsymbol{S} + \boldsymbol{L}_{k+1} - \boldsymbol{Y} + \mu^{-1}\boldsymbol{\Lambda}_k\right\|_F^2 + \varphi(\boldsymbol{L}_{k+1}, \boldsymbol{\Lambda}_k) \\
&= \mathrm{prox}_{\lambda\mu^{-1}\|\cdot\|_1} \left[\boldsymbol{Y} - \boldsymbol{L}_{k+1} - \mu^{-1}\boldsymbol{\Lambda}_k\right] . \qquad (8.5.10)
\end{aligned}$$

Combining these two observations, we obtain a simple, lightweight algorithm for solving the Principal Component Pursuit program.

**Algorithm 8.7** ADMM for Principal Component Pursuit

1: **Problem:** $\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 + \langle \boldsymbol{\Lambda}, \boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y} \rangle + \frac{\mu}{2}\|\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y}\|_F^2$, given $\boldsymbol{Y}$, $\lambda, \mu > 0$.
2: **Input:** $\boldsymbol{L}_0, \boldsymbol{S}_0, \boldsymbol{\Lambda}_0 \in \mathbb{R}^{m\times n}$.
3: **for** $(k = 0, 1, 2, \ldots, K-1)$ **do**
4: $\quad \boldsymbol{L}_{k+1} \leftarrow \mathrm{prox}_{\mu^{-1}\|\cdot\|_*} \left[\boldsymbol{Y} - \boldsymbol{S}_k - \mu^{-1}\boldsymbol{\Lambda}_k\right]$.
5: $\quad \boldsymbol{S}_{k+1} \leftarrow \mathrm{prox}_{\lambda\mu^{-1}\|\cdot\|_1} \left[\boldsymbol{Y} - \boldsymbol{L}_{k+1} - \mu^{-1}\boldsymbol{\Lambda}_k\right]$.
6: $\quad \boldsymbol{\Lambda}_{k+1} \leftarrow \boldsymbol{\Lambda}_k + \mu(\boldsymbol{L}_{k+1} + \boldsymbol{S}_{k+1} - \boldsymbol{Y})$.
7: **end for**
8: **Output:** $\boldsymbol{L}_\star \leftarrow \boldsymbol{L}_K; \boldsymbol{S}_\star \leftarrow \boldsymbol{S}_K$.

### 8.5.2 Monotone Operators

There has been a rich history and literature on charactering the convergence and convergence rates of the ADMM algorithm under various conditions [DY16].

The ADMM can be naturally viewed as an approximation to the classical ALM method studied in the previous section: In the case the objective function is separable, one uses a single pass of "Gauss-Seidel" block minimization to substitute for full minimization of the augmented Lagrangian in each iteration (8.4.11). However, as pointed out in [Eck12], this interpretation does not seem to lead to any known convergence proof for the ADMM.

In the remainder of this section, we give a rigorous proof for the convergence of the ADMM algorithm from the perspective of *monotone operators*, following the work of [HY12, GHY14, Xu17]. As we will see that this approach leads to an alternative proof for the convergence (and convergence rate) of the ALM that is different from the one given in the previous Section 8.4.3. To large extent, this new approach gives a truly unified convergence analysis for both ALM and ADMM. Many of the concepts and techniques to be introduced are very useful in their own right. But for readers who are not immediately concerned with convergence guarantees, they may skip the rest of the section without loss of continuity.

*Monotonicity.*

A *relation* $\mathcal{R}$ on $\mathbb{R}^n$ is defined to be a subset of $\mathbb{R}^n \times \mathbb{R}^n$. Typically, we may view $\mathcal{R}$ as a set-valued mapping. If $\forall \boldsymbol{x} \in \mathbb{R}^n$, $\mathcal{R}(\boldsymbol{x})$ is a singleton or empty, $\mathcal{R}$ is then a function in the conventional sense. Operations such as inverse, composition, scalar multiplication, and addition can be defined as natural extensions to those for functions.

Definition 8.19 (Monotone Relation). *A relation $\mathcal{F}$ on $\mathbb{R}^n$ is monotone if*

$$(\boldsymbol{u} - \boldsymbol{v})^*(\boldsymbol{x} - \boldsymbol{y}) \geq 0 \quad \forall\, (\boldsymbol{x}, \boldsymbol{u}), (\boldsymbol{y}, \boldsymbol{v}) \in \mathcal{F}. \tag{8.5.11}$$

*Moreover, $\mathcal{F}$ is maximal monotone if there is no other monotone relation that properly contains it.*

From this definition, we leave as Exercise 8.14 for the reader to show that given two monotone relations: $\mathcal{F}_1, \mathcal{F}_2$, their sum $\mathcal{F}_1 + \mathcal{F}_2$ is also monotone.

Lemma 8.20 (Monotonicity of Subgradient). *Given a convex function $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$, we have $\mathcal{F}(\boldsymbol{x}) = \partial f(\boldsymbol{x})$ is monotone. That is, for any $\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{v}, \boldsymbol{v}' \in \mathbb{R}^n$ such that $\boldsymbol{v} \in \partial f(\boldsymbol{x})$ and $\boldsymbol{v}' \in \partial f(\boldsymbol{x}')$, we have*

$$\langle \boldsymbol{x} - \boldsymbol{x}', \boldsymbol{v} - \boldsymbol{v}' \rangle \geq 0. \tag{8.5.12}$$

*Proof* From the definition of subgradient, we have

$$f(\boldsymbol{x}') \geq f(\boldsymbol{x}) + \langle \boldsymbol{v}, \boldsymbol{x}' - \boldsymbol{x} \rangle, \quad f(\boldsymbol{x}) \geq f(\boldsymbol{x}') + \langle \boldsymbol{v}', \boldsymbol{x} - \boldsymbol{x}' \rangle. \tag{8.5.13}$$

Adding these two inequalities together we obtain:

$$f(\boldsymbol{x}) + f(\boldsymbol{x}') \geq f(\boldsymbol{x}) + f(\boldsymbol{x}') + \langle \boldsymbol{v} - \boldsymbol{v}', \boldsymbol{x}' - \boldsymbol{x} \rangle. \tag{8.5.14}$$

Canceling $f(\boldsymbol{x}) + f(\boldsymbol{x}')$ from both sides obtains the desired result. □

Now consider the linear equality constrained convex problems of the form

$$\begin{array}{ll} \min & g(\boldsymbol{x}), \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \end{array} \tag{8.5.15}$$

where $g : \mathbb{R}^n \to \mathbb{R}$ is a convex function, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a matrix and $\boldsymbol{y} \in \text{range}(\boldsymbol{A})$. The associated Lagrangian is

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \doteq g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle , \tag{8.5.16}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$. Now consider the relation defined on $\mathbb{R}^n \times \mathbb{R}^m$ by the KKT operator:

$$\mathcal{F}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \partial_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \\ -\partial_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \end{bmatrix} = \begin{bmatrix} \partial g(\boldsymbol{x}) + \boldsymbol{A}^* \boldsymbol{\lambda} \\ \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} \end{bmatrix} . \tag{8.5.17}$$

LEMMA 8.21 (Monotonicity of the KKT Operator). *The KKT operator associated with the linear equality constrained convex optimization problem* (8.5.15) *gives a monotone relation.*

*Proof* We leave the proof to the reader as part of Exercise 8.14. □

*Mixed Variational Inequality (MVI).*
To simplify notation, let us define $\boldsymbol{w} = \left(\begin{smallmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{smallmatrix}\right) \in \mathbb{R}^n \times \mathbb{R}^m$. Then we have:

LEMMA 8.22. *The linear equality constrained optimization problem* (8.5.15) *is equivalent to the problem of solving the mixed variational inequality (MVI): finding* $\boldsymbol{w}_\star \in \mathbb{R}^n \times \mathbb{R}^m$ *such that* $\forall \boldsymbol{w}$

$$g(\boldsymbol{x}) - g(\boldsymbol{x}_\star) + (\boldsymbol{w} - \boldsymbol{w}_\star)^* \mathcal{F}(\boldsymbol{w}_\star) \geq 0, \tag{8.5.18}$$

*where* $\mathcal{F}$ *is a monotone operator:*

$$\mathcal{F}(\boldsymbol{w}) = \mathcal{F}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \boldsymbol{A}^* \boldsymbol{\lambda} \\ \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} \end{bmatrix} = \begin{bmatrix} 0 & \boldsymbol{A}^* \\ -\boldsymbol{A} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{y} \end{bmatrix} . \tag{8.5.19}$$

*Proof* The Lagrangian of (8.5.15) is

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \doteq g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle , \tag{8.5.20}$$

It is equivalent to finding a pair $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$ such that

$$(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star) = \operatorname*{argmin}_{\boldsymbol{x} \in \mathbb{R}^n} \operatorname*{argmax}_{\boldsymbol{\lambda} \in \mathbb{R}^m} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}), \tag{8.5.21}$$

which is a saddle point of $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ and thus satisfies: $\forall \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{\lambda} \in \mathbb{R}^m$,

$$\mathcal{L}(\boldsymbol{x}_\star, \boldsymbol{\lambda}) \leq \mathcal{L}(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star) \leq \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_\star), \tag{8.5.22}$$

which is equivalent to: $\forall \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{\lambda} \in \mathbb{R}^m$,

$$\langle \boldsymbol{\lambda} - \boldsymbol{\lambda}_\star, \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_\star \rangle \geq 0, \tag{8.5.23}$$

$$g(\boldsymbol{x}) - g(\boldsymbol{x}_\star) + \langle \boldsymbol{\lambda}_\star, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{x}_\star \rangle \geq 0. \tag{8.5.24}$$

By the definition of $\mathcal{F}(\boldsymbol{w}_\star)$, on one hand, summing (8.5.23) and (8.5.24), we obtain (8.5.18). On the other hand, in (8.5.18), by setting $\boldsymbol{x} = \boldsymbol{x}^*$, we obtain (8.5.23); by setting $\boldsymbol{\lambda} = \boldsymbol{\lambda}_\star$, we obtain (8.5.24). Therefore (8.5.23) and (8.5.24) are equivalent to (8.5.18).

□

The above Lemma establishes a fundamental connection between constrained convex optimization (8.5.15) and mixed variational inequality (MVI) of the type (8.5.18). As it turns out, it is much easier to characterize convergence of such algorithms, including ALM and ADMM, using MVI. As we will soon see, their iterations can all be interpreted as solving the associated mixed variational inequality approximately. MVIs also arise in a variety of other settings hence it is of independent value to understand their properties and how to solve them.

To this end, let us consider the general mixed variational inequality problem.

Problem 8.23 (Mixed Variational Inequality Problem). *Find* $\boldsymbol{w}_\star = (\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$ *such that in certain closed convex set* $\Omega \subseteq \mathbb{R}^{n\times m}$*, we have*

$$\forall \boldsymbol{w} \in \Omega, \quad \theta(\boldsymbol{u}) - \theta(\boldsymbol{u}_\star) + (\boldsymbol{w} - \boldsymbol{w}_\star)^* \mathcal{F}(\boldsymbol{w}_\star) \geq 0, \tag{8.5.25}$$

*where* $\mathcal{F}$ *is monotone,* $\boldsymbol{u}$ *is a sub-vector of* $\boldsymbol{w}$*, and* $\theta(\boldsymbol{u})$ *is a general convex function in* $\boldsymbol{u}$*.*

It is easy to show that (8.5.25) is equivalent to the following condition:

$$\forall \boldsymbol{w} \in \Omega, \quad \theta(\boldsymbol{u}) - \theta(\boldsymbol{u}_\star) + (\boldsymbol{w} - \boldsymbol{w}_\star)^* \mathcal{F}(\boldsymbol{w}) \geq 0. \tag{8.5.26}$$

We leave the proof as an exercise to the reader and others may find one in [HY12, Theorem 2.1].

To find a solution to (8.5.26), a natural approach is to find approximate solution $\tilde{\boldsymbol{w}}$ that is an $\varepsilon$-accurate solution. Or more precisely, $\forall \boldsymbol{w} \in \Omega$,

$$\theta(\boldsymbol{u}) - \theta(\tilde{\boldsymbol{u}}) + (\boldsymbol{w} - \tilde{\boldsymbol{w}})^* \mathcal{F}(\boldsymbol{w}) \geq -\varepsilon, \tag{8.5.27}$$

or equivalently

$$\theta(\tilde{\boldsymbol{u}}) - \theta(\boldsymbol{u}) + (\tilde{\boldsymbol{w}} - \boldsymbol{w})^* \mathcal{F}(\boldsymbol{w}) \leq \varepsilon. \tag{8.5.28}$$

To find an $\varepsilon$-accurate solution $\tilde{\boldsymbol{w}}$ for (8.5.28), a popular method is the following *proximal point algorithm* (PPA): in the $k$-th iteration ($k \geq 1$), generating the new iterate $\boldsymbol{w}_{k+1} \in \Omega$ such that

$$\theta(\boldsymbol{u}) - \theta(\boldsymbol{u}_{k+1}) + (\boldsymbol{w} - \boldsymbol{w}_{k+1})^* (\mathcal{F}(\boldsymbol{w}_{k+1}) + \boldsymbol{Q}(\boldsymbol{w}_{k+1} - \boldsymbol{w}_k)) \geq 0, \tag{8.5.29}$$

where $\boldsymbol{Q}$ is symmetric and positive semidefinite. This objective is intended to emulate the proximal method that we have introduced earlier: while in each iteration we try to achieve the objective, say (8.5.27), but we do not want to deviate from the previous $\boldsymbol{w}_k$ too much. If we are able to find such iterate $\boldsymbol{w}_{k+1}$, then we have the following nice convergence result for the PPA:

THEOREM 8.24 (Convergence of the Proximal Point Algorithm). *For all integers $k \geq 1$, define $\tilde{\boldsymbol{w}}_k \doteq \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{w}_i$, where $\boldsymbol{w}_i$ is generated by (8.5.29), then we have $\tilde{\boldsymbol{w}}_k \in \Omega$ and $\forall \boldsymbol{w} \in \Omega$,*

$$\sum_{i=1}^{k} (\theta(\boldsymbol{u}_i) - \theta(\boldsymbol{u}) + (\boldsymbol{w}_i - \boldsymbol{w})^* \mathcal{F}(\boldsymbol{w}_i)) \leq \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2 \tag{8.5.30}$$

*and*

$$\theta(\tilde{\boldsymbol{u}}_k) - \theta(\boldsymbol{u}) + (\tilde{\boldsymbol{w}}_k - \boldsymbol{w})^* \mathcal{F}(\boldsymbol{w}) \leq \frac{1}{2k}\|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2, \tag{8.5.31}$$

*where $\tilde{\boldsymbol{u}}_k$ (resp. $\boldsymbol{u}$) is the corresponding subvector of $\tilde{\boldsymbol{w}}_k$ (resp. $\boldsymbol{w}$).*

*Proof* By (8.5.29), we have

$$\theta(\boldsymbol{u}) - \theta(\boldsymbol{u}_{k+1}) + (\boldsymbol{w} - \boldsymbol{w}_{k+1})^* \mathcal{F}(\boldsymbol{w}_{k+1}) \geq (\boldsymbol{w} - \boldsymbol{w}_{k+1})^* \boldsymbol{Q}(\boldsymbol{w}_k - \boldsymbol{w}_{k+1}). \tag{8.5.32}$$

Meanwhile, we have the following relation

$$\begin{aligned} & (\boldsymbol{w} - \boldsymbol{w}_{k+1})^* \boldsymbol{Q}(\boldsymbol{w}_k - \boldsymbol{w}_{k+1}) \\ = & \frac{1}{2}\big(\|\boldsymbol{w} - \boldsymbol{w}_{k+1}\|_{\boldsymbol{Q}}^2 - \|\boldsymbol{w} - \boldsymbol{w}_k\|_{\boldsymbol{Q}}^2\big) + \frac{1}{2}\|\boldsymbol{w}_k - \boldsymbol{w}_{k+1}\|_{\boldsymbol{Q}}^2 \\ \geq & \frac{1}{2}(\|\boldsymbol{w} - \boldsymbol{w}_{k+1}\|_{\boldsymbol{Q}}^2 - \|\boldsymbol{w} - \boldsymbol{w}_k\|_{\boldsymbol{Q}}^2). \end{aligned} \tag{8.5.33}$$

By combining (8.5.32) and (8.5.33), we have

$$\theta(\boldsymbol{u}) - \theta(\boldsymbol{u}_{k+1}) + (\boldsymbol{w} - \boldsymbol{w}_{k+1})^* \mathcal{F}(\boldsymbol{w}) \geq \frac{1}{2}\big(\|\boldsymbol{w} - \boldsymbol{w}_{k+1}\|_{\boldsymbol{Q}}^2 - \|\boldsymbol{w} - \boldsymbol{w}_k\|_{\boldsymbol{Q}}^2\big). \tag{8.5.34}$$

Summing (8.5.34) over $i = 1, 2, \ldots, k$, we have

$$\begin{aligned} & k\left(\Big(\theta(\boldsymbol{u}) - \sum_{i=1}^{k}\frac{1}{k}\theta(\boldsymbol{u}_i) + \big(\boldsymbol{w} - \sum_{i=1}^{k}\frac{1}{k}\boldsymbol{w}_i\big)^* \mathcal{F}(\boldsymbol{w})\Big)\right) \\ \geq & \frac{1}{2}(\|\boldsymbol{w} - \boldsymbol{w}_k\|_{\boldsymbol{Q}}^2 - \|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2) \;\geq\; -\frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2. \end{aligned} \tag{8.5.35}$$

By the convexity of $\theta(\boldsymbol{u})$, we have

$$\theta\left(\sum_{i=1}^{k}\frac{1}{k}\boldsymbol{u}_i\right) \leq \sum_{i=1}^{k}\frac{1}{k}\theta(\boldsymbol{u}_i). \tag{8.5.36}$$

Combining (8.5.35) and (8.5.36) leads to the statement of the theorem. □

Notice the theorem implies that the convergence rate of PPA is at least $O(1/k)$.

### 8.5.3 Convergence of ALM and ADMM

*Reducing ALM and ADMM to PPA.*
Now let us use the above result to show the convergence (and convergence rate) of the ALM algorithm that we have previously studied in Section 8.4.3.

THEOREM 8.25 (Reducing ALM to PPA). *The update rule of ALM in* (8.4.11) *and* (8.4.12) *reduces to the following PPA problem: in the k-th iteration, finding a* $\boldsymbol{w}_{k+1} \doteq (\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1})$ *such that* $\forall \boldsymbol{w} \in \mathbb{R}^n \times \mathbb{R}^m$ ,

$$g(\boldsymbol{x}) - g(\boldsymbol{x}_{k+1}) + (\boldsymbol{w} - \boldsymbol{w}_{k+1})^* \big(\mathcal{F}(\boldsymbol{w}_{k+1}) + \boldsymbol{Q}(\boldsymbol{w}_{k+1} - \boldsymbol{w}_k)\big) \geq 0, \quad (8.5.37)$$

*where*

$$\mathcal{F}(\boldsymbol{w}) \doteq \begin{bmatrix} \boldsymbol{A}^* \boldsymbol{\lambda} \\ \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} \end{bmatrix} \quad \textit{and} \quad \boldsymbol{Q} \doteq \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \frac{1}{\mu}\boldsymbol{I}_m \end{bmatrix}. \quad (8.5.38)$$

*Proof* By the optimality condition (8.4.11), we have $\forall \boldsymbol{x} \in \mathbb{R}^n$,

$$g(\boldsymbol{x}) - g(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{x} - \boldsymbol{x}_{k+1}, \boldsymbol{A}^* \boldsymbol{\lambda}_k + \mu \boldsymbol{A}^* (\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y}) \rangle \geq 0. \quad (8.5.39)$$

By (8.4.12), (8.5.39) is equivalent to $\forall \boldsymbol{x} \in \mathbb{R}^n$,

$$g(\boldsymbol{x}) - g(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{x} - \boldsymbol{x}_{k+1}, \boldsymbol{A}^* \boldsymbol{\lambda}_{k+1} \rangle \geq 0. \quad (8.5.40)$$

The update rule for $\lambda$ (8.4.12) itself is also equivalent to $\forall \boldsymbol{\lambda} \in \mathbb{R}^m$

$$\langle \boldsymbol{\lambda} - \boldsymbol{\lambda}_{k+1}, (\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{k+1}) + \frac{1}{\mu}(\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k) \rangle = 0. \quad (8.5.41)$$

Then by the definition of $\mathcal{F}(\boldsymbol{w}_{k+1})$ and $\boldsymbol{Q}$ in (8.5.38), combining (8.5.40) and (8.5.41), gives (8.5.37). □

This theorem gives another proof for the convergence of the ALM based on PPA, which is different from the proximal-gradient based proof given in Section 8.4.3. According to Theorem 8.24, the convergence rate of ALM is at least $O(1/k)$, the same as the previous proof. The reason for going through this new approach is that this leads to a unified proof for the convergence for the ADMM algorithm, at least its symmetric version below.

Now let us consider the ADMM method for the problem (8.5.2). Recall that the associate augmented Lagrangian is

$$\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) \doteq g(\boldsymbol{x}) + h(\boldsymbol{z}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y} \rangle + \frac{\mu}{2} \|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y}\|_2^2.$$

Then in the $k$-th iteration, consider the following ADMM update rules:[8]

$$\boldsymbol{x}_{k+1} = \operatorname*{argmin}_{\boldsymbol{x}} \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}_k, \boldsymbol{\lambda}_k), \quad (8.5.42)$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \mu(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_k - \boldsymbol{y}), \quad (8.5.43)$$

$$\boldsymbol{z}_{k+1} = \operatorname*{argmin}_{\boldsymbol{z}} \mathcal{L}_\mu(\boldsymbol{x}_{k+1}, \boldsymbol{z}, \boldsymbol{\lambda}_{k+1}). \quad (8.5.44)$$

[8] Notice that these update rules are in slightly different order than those in (8.5.5) - (8.5.7). The rules here are also known as a *symmetric* version of ADMM. The proof of convergence for the symmetric version is relatively simpler. The proof for the conventional ADMM rules can follow a similar strategy but the analysis is a little more involved.

THEOREM 8.26 (Reducing ADMM to PPA). *The update rules of ADMM in (8.5.42) to (8.5.44) can be reduced to the following PPA problem: in the k-th iteration, finding a* $\boldsymbol{w}_{k+1} \doteq (\boldsymbol{x}_{k+1}, \boldsymbol{z}_{k+1}, \boldsymbol{\lambda}_{k+1})$ *such that* $\forall \boldsymbol{w}$,

$$\begin{aligned}&(g(\boldsymbol{x}) + h(\boldsymbol{z})) - (g(\boldsymbol{x}_{k+1}) + h(\boldsymbol{z}_{k+1})) + \\ &(\boldsymbol{w} - \boldsymbol{w}_{k+1})^*\big(\mathcal{F}(\boldsymbol{w}_{k+1}) + \boldsymbol{Q}(\boldsymbol{w}_{k+1} - \boldsymbol{w}_k)\big) \geq 0,\end{aligned} \tag{8.5.45}$$

*where*

$$\mathcal{F}(\boldsymbol{w}) \doteq \begin{bmatrix} \boldsymbol{A}^*\boldsymbol{\lambda} \\ \boldsymbol{B}^*\boldsymbol{\lambda} \\ \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{B}\boldsymbol{z} \end{bmatrix} \quad \textit{and} \quad \boldsymbol{Q} \doteq \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \mu\boldsymbol{B}^*\boldsymbol{B} & -\boldsymbol{B}^* \\ \boldsymbol{0} & -\boldsymbol{B} & \frac{1}{\mu}\boldsymbol{I}_m \end{bmatrix} \succeq 0. \tag{8.5.46}$$

*Proof* By the optimality condition (8.5.42), we have $\forall \boldsymbol{x}$,

$$g(\boldsymbol{x}) - g(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{x} - \boldsymbol{x}_{k+1}, \boldsymbol{A}^*\boldsymbol{\lambda}_k + \mu\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_k - \boldsymbol{y})\rangle \geq 0.$$

By (8.5.43), (8.5.47) is equivalent to $\forall \boldsymbol{x}$

$$g(\boldsymbol{x}) - g(\boldsymbol{x}_{k+1}) + \langle \boldsymbol{x} - \boldsymbol{x}_{k+1}, \boldsymbol{A}^*\boldsymbol{\lambda}_{k+1}\rangle \geq 0. \tag{8.5.47}$$

The update rule (8.5.43) is also equivalent to $\forall \boldsymbol{\lambda}$

$$\langle \boldsymbol{\lambda} - \boldsymbol{\lambda}_{k+1}, (\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{B}\boldsymbol{z}_{k+1}) + \boldsymbol{B}(\boldsymbol{z}_{k+1} - \boldsymbol{z}_k) + \frac{1}{\mu}(\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k)\rangle = 0. \tag{8.5.48}$$

By the optimality condition of (8.5.44), we have $\forall \boldsymbol{z}$,

$$h(\boldsymbol{z}) - h(\boldsymbol{z}_{k+1}) + \langle \boldsymbol{z} - \boldsymbol{z}_{k+1}, \boldsymbol{B}^*\boldsymbol{\lambda}_{k+1} + \mu\boldsymbol{B}^*(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_{k+1} - \boldsymbol{y})\rangle \geq 0,$$

which is equivalent to $\forall \boldsymbol{z}$

$$\begin{aligned}&h(\boldsymbol{z}) - h(\boldsymbol{z}_{k+1}) \\ &+ \langle \boldsymbol{z} - \boldsymbol{z}_{k+1}, \boldsymbol{B}^*\boldsymbol{\lambda}_{k+1} + \boldsymbol{B}^*(\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k) + \mu\boldsymbol{B}^*\boldsymbol{B}(\boldsymbol{z}_{k+1} - \boldsymbol{z}_k)\rangle \\ &\geq 0,\end{aligned} \tag{8.5.49}$$

Then with the definition of $\mathcal{F}(\boldsymbol{w}_{k+1})$ and $\boldsymbol{Q}$ in (8.5.46), by combining (8.5.47), (8.5.48) and (8.5.49), we obtain (8.5.45). □

This theorem implies that ADMM can be reduced to PPA hence it inherits the $O(1/k)$ convergence rate established earlier for PPA.

*Convergence of ALM and ADMM.*
Notice that the convergence in terms of PPA only guarantees the sum of objective function value and the constraint, i.e., left hand side of (8.5.31), converges.[9] As it turns out, in our context, the constraints are mostly linear equalities. By exploiting nice properties of such constraints, it is possible to ensure that the objective function value and the constraint accuracy converge separately [Xu17]. This only requires minor modification to the above proofs.

[9] So rigorously speaking, there is no guarantee that each of the term would necessarily converge separately.

THEOREM 8.27 (Convergence of ALM). *Assume* $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$ *is the optimal solution of* (8.4.9). *Then the update rules of ALM in* (8.4.11) *and* (8.4.12) *have the following guarantee that letting* $\tilde{\boldsymbol{x}}_k \doteq \frac{1}{k}\sum_{i=1}^k \boldsymbol{x}_i$ *and given* $\rho > \|\boldsymbol{\lambda}_\star\|_2$, *we have*

$$\|\boldsymbol{A}\tilde{\boldsymbol{x}}_k - \boldsymbol{y}\|_2 \leq \frac{1}{2(\rho - \|\boldsymbol{\lambda}_*\|_2)k}\|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2, \tag{8.5.50}$$

*and*

$$-\frac{\|\boldsymbol{\lambda}_\star\|_2}{2(\rho - \|\boldsymbol{\lambda}_\star\|_2)k}\|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2 \leq g(\tilde{\boldsymbol{x}}_k) - g(\boldsymbol{x}_\star) \leq \frac{1}{2k}\|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2, \tag{8.5.51}$$

*with* $\boldsymbol{w} \doteq \begin{bmatrix} \boldsymbol{x}_\star \\ \frac{\rho(\boldsymbol{A}\tilde{\boldsymbol{x}}_k - \boldsymbol{y})}{\|\boldsymbol{A}\tilde{\boldsymbol{x}}_k - \boldsymbol{y}\|_2} \end{bmatrix}$, $\boldsymbol{w}_0 \doteq \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{\lambda}_0 \end{bmatrix}$.

*Proof* For ALM, let $\boldsymbol{w} \doteq \begin{bmatrix} \boldsymbol{x}_\star \\ \boldsymbol{\lambda} \end{bmatrix}$, where $\boldsymbol{x}_\star$ is the global minimum with $\boldsymbol{A}\boldsymbol{x}_\star = \boldsymbol{y}$ and $\boldsymbol{\lambda} \in \mathbb{R}^m$ is to be determined. Then for the $\mathcal{F}(\boldsymbol{w})$ defined in (8.5.38), we have

$$\begin{aligned} & (\boldsymbol{w} - \boldsymbol{w}_{k+1})^* \mathcal{F}(\boldsymbol{w}_{k+1}) \\ =\ & \langle \boldsymbol{x}_\star - \boldsymbol{x}_{k+1}, \boldsymbol{A}^*\boldsymbol{\lambda}_{k+1}\rangle + \langle \boldsymbol{\lambda} - \boldsymbol{\lambda}_{k+1}, \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{k+1}\rangle \\ =\ & \langle \boldsymbol{\lambda}_{k+1}, \boldsymbol{A}\boldsymbol{x}_\star - \boldsymbol{y}\rangle + \langle \boldsymbol{\lambda}, \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{k+1}\rangle \\ =\ & \langle \boldsymbol{\lambda}, \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{k+1}\rangle, \end{aligned} \tag{8.5.52}$$

which is a linear function with respect to $\boldsymbol{x}_{k+1}$.

Then combining the (8.5.30) of Theorem 8.24, and Theorem 8.25, we have

$$\sum_{i=1}^k (g(\boldsymbol{x}_i) - g(\boldsymbol{x}) + (\boldsymbol{w}_i - \boldsymbol{w})^* \mathcal{F}(\boldsymbol{w}_i)) \leq \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2. \tag{8.5.53}$$

So by out setting of $\boldsymbol{w}$, applying the convexity of $g(\boldsymbol{x})$, and combining (8.5.52) and (8.5.53), it follows that

$$\begin{aligned} & k(g(\tilde{\boldsymbol{x}}_k) - g(\boldsymbol{x}_\star) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\tilde{\boldsymbol{x}}_k - \boldsymbol{y}\rangle) \\ \leq\ & \sum_{i=1}^k (g(\boldsymbol{x}_i) - g(\boldsymbol{x}) + (\boldsymbol{w}_i - \boldsymbol{w})^* \mathcal{F}(\boldsymbol{w}_i)) \\ \leq\ & \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2, \end{aligned} \tag{8.5.54}$$

where $\tilde{\boldsymbol{x}}_k \doteq \frac{1}{k}\sum_{i=1}^k \boldsymbol{x}_i$. By setting $\boldsymbol{\lambda} \doteq \frac{\rho(\boldsymbol{A}\tilde{\boldsymbol{x}}_k - \boldsymbol{y})}{\|\boldsymbol{A}\tilde{\boldsymbol{x}}_k - \boldsymbol{y}\|_2}$ with $\rho > 0$ to be determined, we have

$$g(\tilde{\boldsymbol{x}}_k) - g(\boldsymbol{x}_\star) + \rho\|\boldsymbol{A}\tilde{\boldsymbol{x}}_k - \boldsymbol{y}\|_2 \leq \frac{1}{2k}\|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2. \tag{8.5.55}$$

Assume that $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$ is the optimal solution of (8.4.9), then by the KKT condition we have: $\forall \boldsymbol{x}$

$$g(\boldsymbol{x}) - g(\boldsymbol{x}_\star) - \langle \boldsymbol{\lambda}_\star, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\rangle \geq 0. \tag{8.5.56}$$

So we have

$$g(\tilde{\boldsymbol{x}}_k) - g(\boldsymbol{x}_\star) \geq -\|\boldsymbol{\lambda}_\star\|_2 \|\boldsymbol{A}\tilde{\boldsymbol{x}}_k - \boldsymbol{y}\|_2. \tag{8.5.57}$$

By combining (8.5.55) and (8.5.57), with the setting $\rho > \|\boldsymbol{\lambda}_\star\|_2$, we have

$$\|\boldsymbol{A}\tilde{\boldsymbol{x}}_k - \boldsymbol{y}\|_2 \leq \frac{1}{2(\rho - \|\boldsymbol{\lambda}_*\|_2)k} \|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2, \tag{8.5.58}$$

and

$$-\frac{\|\boldsymbol{\lambda}_\star\|_2}{2(\rho - \|\boldsymbol{\lambda}_\star\|_2)k} \|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2 \leq g(\tilde{\boldsymbol{x}}_k) - g(\boldsymbol{x}_\star) \leq \frac{1}{2k} \|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2. \tag{8.5.59}$$

□

THEOREM 8.28 (Convergence of ADMM). *Assume* $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$ *is the optimal solution of* (8.5.4). *Then the update rules of ADMM* (8.5.42) *to* (8.5.44) *have the following guarantee that letting* $\tilde{\boldsymbol{x}}_k = \frac{1}{k}\sum_{i=1}^k \boldsymbol{x}_i$ *and given* $\rho > \|\boldsymbol{\lambda}_\star\|_2$, *we have*

$$\|\boldsymbol{A}\tilde{\boldsymbol{x}}_k - \boldsymbol{y}\|_2 \leq \frac{1}{2(\rho - \|\boldsymbol{\lambda}_\star\|_2)k} \|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2, \tag{8.5.60}$$

*and*

$$-\frac{\|\boldsymbol{\lambda}_\star\|_2}{2(\rho - \|\boldsymbol{\lambda}_\star\|_2)k} \|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2 \leq g(\tilde{\boldsymbol{x}}_k) + h(\tilde{\boldsymbol{z}}_k) - (g(\boldsymbol{x}_\star) + h(\boldsymbol{z}_\star)) \leq \frac{1}{2k} \|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2,$$

*with* $\boldsymbol{w} = \begin{bmatrix} \boldsymbol{x}_\star \\ \boldsymbol{z}_\star \\ \frac{\rho(\boldsymbol{A}\tilde{\boldsymbol{x}}_k + \boldsymbol{B}\tilde{\boldsymbol{z}}_k - \boldsymbol{y})}{\|\boldsymbol{A}\tilde{\boldsymbol{x}}_k + \boldsymbol{B}\tilde{\boldsymbol{z}}_k - \boldsymbol{y}\|_2} \end{bmatrix}$, $\boldsymbol{w}_0 = \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{z}_0 \\ \boldsymbol{\lambda}_0 \end{bmatrix}$.

*Proof* For ADMM, by setting $\boldsymbol{w} \doteq \begin{bmatrix} \boldsymbol{x}_\star \\ \boldsymbol{z}_\star \\ \boldsymbol{\lambda} \end{bmatrix}$, where $(\boldsymbol{x}_\star, \boldsymbol{z}_\star)$ is the global minimum of the equality constrained convex problem that satisfies $\boldsymbol{A}\boldsymbol{x}_\star + \boldsymbol{B}\boldsymbol{z}_\star - \boldsymbol{y} = \boldsymbol{0}$ and $\boldsymbol{\lambda} \in \mathbb{R}^m$ is to be determined. Then we have

$$\begin{aligned}
&(\boldsymbol{w} - \boldsymbol{w}_{k+1})^* \mathcal{F}(\boldsymbol{w}_{k+1}) \\
&= \langle \boldsymbol{x}_\star - \boldsymbol{x}_{k+1}, \boldsymbol{A}^*\boldsymbol{\lambda}_{k+1} \rangle + \langle \boldsymbol{z}_\star - \boldsymbol{z}_{k+1}, \boldsymbol{B}^*\boldsymbol{\lambda}_{k+1} \rangle + \langle \boldsymbol{\lambda} - \boldsymbol{\lambda}_{k+1}, \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{B}\boldsymbol{z}_{k+1} \rangle \\
&= \langle \boldsymbol{\lambda}_{k+1}, \boldsymbol{A}\boldsymbol{x}_\star + \boldsymbol{B}\boldsymbol{z}_\star - \boldsymbol{y} \rangle + \langle \boldsymbol{\lambda}, \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{B}\boldsymbol{z}_{k+1} \rangle \\
&= \langle \boldsymbol{\lambda}, \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{B}\boldsymbol{z}_{k+1} \rangle,
\end{aligned} \tag{8.5.61}$$

which is a linear function with respect to $\boldsymbol{x}_{k+1}$ and $\boldsymbol{z}_{k+1}$.

Then combining the (8.5.30) of Theorem 8.24, and Theorem 8.26, we have

$$\sum_{i=1}^k \left(g(\boldsymbol{x}_i) + h(\boldsymbol{z}_i) - (g(\boldsymbol{x}_\star) + h(\boldsymbol{x}_\star)) + (\boldsymbol{w}_i - \boldsymbol{w})^* \mathcal{F}(\boldsymbol{w}_i)\right) \leq \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2. \tag{8.5.62}$$

So applying the convexity of $g(\boldsymbol{x})$ and $h(\boldsymbol{z})$, and combining (8.5.61) and (8.5.62), it follows that

$$\begin{aligned} & k(g(\tilde{\boldsymbol{x}}_k) + h(\tilde{\boldsymbol{z}}_k) - (g(\boldsymbol{x}_\star) + h(\boldsymbol{z}_\star)) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\tilde{\boldsymbol{x}}_k + \boldsymbol{B}\tilde{\boldsymbol{z}}_k - \boldsymbol{y} \rangle) \\ \leq & \sum_{i=1}^{k} (g(\boldsymbol{x}_i) - g(\boldsymbol{x}_\star) + h(\boldsymbol{x}_i) - h(\boldsymbol{x}_\star) + (\boldsymbol{w}_i - \boldsymbol{w})^* \mathcal{F}(\boldsymbol{w}_i)) \\ \leq & \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2, \end{aligned} \tag{8.5.63}$$

where $\tilde{\boldsymbol{x}}_k \doteq \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{x}_i$. By setting $\boldsymbol{\lambda} \doteq \frac{\rho(\boldsymbol{A}\tilde{\boldsymbol{x}}_k + \boldsymbol{B}\tilde{\boldsymbol{z}}_k - \boldsymbol{y})}{\|\boldsymbol{A}\tilde{\boldsymbol{x}}_k + \boldsymbol{B}\tilde{\boldsymbol{z}}_k - \boldsymbol{y}\|_2}$ with $\rho > 0$ to be determined, we have

$$(g(\tilde{\boldsymbol{x}}_k)+h(\tilde{\boldsymbol{z}}_k))-(g(\boldsymbol{x}_\star)+h(\boldsymbol{z}_\star))+\rho\|\boldsymbol{y}-\boldsymbol{A}\tilde{\boldsymbol{x}}_k-\boldsymbol{B}\tilde{\boldsymbol{z}}_k\|_2 \leq \frac{1}{2k}\|\boldsymbol{w}-\boldsymbol{w}_0\|_{\boldsymbol{Q}}^2. \tag{8.5.64}$$

Assume that $(\boldsymbol{x}_\star, \boldsymbol{z}_\star, \boldsymbol{\lambda}_\star)$ is the optimal solution of (8.5.4), then by the KKT condition, we have: $\forall \boldsymbol{x}, \boldsymbol{z}$,

$$g(\boldsymbol{x}) + h(\boldsymbol{z}) - (g(\boldsymbol{x}_\star) + h(\boldsymbol{z}_\star)) - \langle \boldsymbol{\lambda}_\star, \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y} \rangle \geq 0. \tag{8.5.65}$$

So we have

$$g(\tilde{\boldsymbol{x}}_k) + h(\tilde{\boldsymbol{z}}_k) - (g(\boldsymbol{x}_\star) + h(\boldsymbol{z}_\star)) \geq -\|\boldsymbol{\lambda}_\star\|_2 \|\boldsymbol{A}\tilde{\boldsymbol{x}}_k + \boldsymbol{B}\tilde{\boldsymbol{z}}_k - \boldsymbol{y}\|_2. \tag{8.5.66}$$

By combining (8.5.64) and (8.5.66), with the setting $\rho > \|\boldsymbol{\lambda}_\star\|_2$, we have

$$\|\boldsymbol{A}\tilde{\boldsymbol{x}}_k + \boldsymbol{B}\tilde{\boldsymbol{z}}_k - \boldsymbol{y}\|_2 \leq \frac{1}{2(\rho - \|\boldsymbol{\lambda}_\star\|_2)k}\|\boldsymbol{w} - \boldsymbol{w}_0\|_{\boldsymbol{Q}}^2,$$

and

$$-\frac{\|\boldsymbol{\lambda}_\star\|_2}{2(\rho - \|\boldsymbol{\lambda}_\star\|_2)k}\|\boldsymbol{w}-\boldsymbol{w}_0\|_{\boldsymbol{Q}}^2 \leq g(\tilde{\boldsymbol{x}}_k)+h(\tilde{\boldsymbol{z}}_k)-(g(\boldsymbol{x}_\star)+h(\boldsymbol{z}_\star)) \leq \frac{1}{2k}\|\boldsymbol{w}-\boldsymbol{w}_0\|_{\boldsymbol{Q}}^2.$$

□

The above convergence rate $O(1/k)$ is actually optimal for first-order methods, according to [OX18]. However, when the linear constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ satisfies certain special properties, one may achieve convergence rate faster than $O(1/k)$, as we will discuss more in the Notes section.

*Alternating among Multiple Separable Terms.*
Finally, we want to point out that, more generally, separable structures also arise in many large scale learning problems, where the goal is to fit a parametric model to a collection of observation vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p$. Typically, we are provided with a *loss* $L(\boldsymbol{y}, \boldsymbol{x})$, which could be, e.g., the log likelihood of observation $\boldsymbol{y}$ given parameters $\boldsymbol{x}$ or the logistic loss in training a classifier with a deep network. Our goal is to minimize $\sum_i L(\boldsymbol{y}_i, \boldsymbol{x})$ over $\boldsymbol{x}$.

In very large scale applications, it may be prohibitively expensive to store the $\boldsymbol{y}_i$ centrally, or to transmit them during the operation of an iterative algorithm. Rather, we can assume that they are stored in a distributed fashion,

in $N$ locations: the $j$-th location stores $\{\boldsymbol{y}_i, i \in \mathsf{I}_j\}$. The loss on this subset is $f_j(\boldsymbol{x}) = \sum_{i \in \mathsf{I}_j} L(\boldsymbol{y}_i, \boldsymbol{x})$. Our overall goal is then to solve

$$\min_{\boldsymbol{x}} \sum_{j=1}^{N} f_j(\boldsymbol{x}). \tag{8.5.67}$$

Again, this objective function appears to separate into independent terms. To exploit this structure, we can introduce $N$ additional parameter vectors $\boldsymbol{x}_j$, which are constrained to coincide with $\boldsymbol{x}$:

$$\begin{array}{ll} \min_{\{\boldsymbol{x}_j\}} & \sum_{j=1}^{N} f_j(\boldsymbol{x}_j) \\ \text{subject to} & \boldsymbol{x}_j = \boldsymbol{x}, \quad j = 1, \ldots, N. \end{array} \tag{8.5.68}$$

It is common practice that people apply similar alternating schemes to optimize this class of problems. But the convergence and complexity analysis for ADMM with multiple terms are much more difficult, as we will discuss more in Notes.

## 8.6 Leveraging Problem Structures for Better Scalability

In the previous sections, we showed how the special structure of optimization problems arising in sparse and low-dimensional data analysis can be leveraged to obtain efficient and scalable algorithms. One key piece of structure was the existence of an easy-to-compute proximal operator. For example, for nuclear norm minimization we showed that at a point $\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$,

$$\mathrm{prox}_{\lambda\|\cdot\|_*}[\boldsymbol{Z}] = \boldsymbol{U}\mathrm{soft}(\boldsymbol{\Sigma}, \lambda)\boldsymbol{V}^*, \tag{8.6.1}$$

where $\mathrm{soft}(\cdot, \lambda)$ is the soft thresholding operator on the singular values. Using the proximal operator, we obtain proximal gradient methods that enjoy the same convergence rate as if the objective was smooth, even though it is nonsmooth. Each iteration consists of simple linear operations, followed by the application of $\mathrm{prox}_{\lambda\|\cdot\|_*}[\cdot]$. Each iteration can be computed in time polynomial in the size of the target matrix: the proximal operator can be computed in time $O(n_1 n_2 \max\{n_1, n_2\})$ in the worst case. This is sufficient for moderate-sized datasets where $n_1$ and $n_2$ are each in the thousands.

Nevertheless, many problems in data science, scientific imaging, and machine learning require even more scalable solutions. The Frank-Wolfe method and Stochastic Gradient Descent (SGD) are two such methods. The two methods exploit two complementary types of structures that are common in high-dimensional optimization problems on large-scale datasets. The Frank-Wolfe exploits structures in the constraints or in the data (e.g. the atomic structures) so as to *reduce the dependency of an algorithm's complexity on the dimension* $n$, typically from linear to sublinear. Roughly speaking, SGD exploits finite-sum structure in the objective function, say a sum of errors or losses for a large number of samples. By leveraging gradients computed from small random batches of

samples instead of the full set, SGD can *reduce the dependency of an algorithm's complexity on the sample size* $m$, again from linear to sublinear. In this section, we illustrate basic ideas behind both methods and illustrate their connections to our problems.

### 8.6.1 Frank-Wolfe for Structured Constraint Set

In this section, we introduce a classical method from optimization, known as the *Frank-Wolfe* or *conditional gradient* algorithm, which is scalable enough to solve extremely large sparse and low-rank recovery problems. The key property of this method is that in each iteration, it solves a subproblem which is simpler and easier to compute than the proximal operator.

In its classical form, the Frank-Wolfe algorithm, originally proposed in [FW56], applies to the problem of optimizing a smooth, convex function over a *compact* convex set:

$$\begin{array}{ll} \min_{\boldsymbol{x}} & f(\boldsymbol{x}), \\ \text{subject to} & \boldsymbol{x} \in \mathsf{C}. \end{array} \tag{8.6.2}$$

Here, the objective function $f$ is assumed to be a convex,[10] differentiable function whose gradient $\nabla f(\boldsymbol{x})$ is $L$-Lipschitz. The constraint set $\mathsf{C}$ is assumed to be a compact (hence closed and bounded) convex set with a diameter

$$\operatorname{diam}(\mathsf{C}) \doteq \max\{\|\boldsymbol{x} - \boldsymbol{x}'\|_2 \mid \boldsymbol{x}, \boldsymbol{x}' \in \mathsf{C}\}. \tag{8.6.3}$$

*Constrained Formulations of Sparse and Low-rank Recovery.*
Many of the sparse and low-rank recovery problems that we have considered thus far can be reformulated in terms of (8.6.2). For example, for sparse recovery, we can choose $\mathsf{C} = \{\boldsymbol{x} \mid \|\boldsymbol{x}\|_1 \le \tau\}$ to be an $\ell^1$ ball, and solve

$$\begin{array}{ll} \min_{\boldsymbol{x}} & \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2, \\ \text{subject to} & \|\boldsymbol{x}\|_1 \le \tau. \end{array} \tag{8.6.4}$$

Similarly, for low-rank matrix completion, we can choose $\mathsf{C}$ to be a nuclear norm ball and solve

$$\begin{array}{ll} \min_{\boldsymbol{X}} & \frac{1}{2}\|\mathcal{P}_\Omega[\boldsymbol{X}] - \boldsymbol{Y}\|_F^2, \\ \text{subject to} & \|\boldsymbol{X}\|_* \le \tau. \end{array} \tag{8.6.5}$$

Exercises 8.10 – 8.11 explore further reformulations of unconstrained sparse and low-rank optimization in the form (8.6.2).

Similar to the other methods we have discussed thus far, Frank-Wolfe is an iterative method, which generates a sequence of iterates $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, \ldots$ as follows. At each iteration, we generate a new point $\boldsymbol{v}_k$ by solving a constrained optimization problem

$$\boldsymbol{v}_k \in \arg\min_{\boldsymbol{v} \in \mathsf{C}} \langle \boldsymbol{v}, \nabla f(\boldsymbol{x}_k) \rangle. \tag{8.6.6}$$

[10] The Frank-Wolfe algorithm can also work when $f(\boldsymbol{x})$ is nonconvex. One can show that it also converges but has a convergence rate $O(1/\sqrt{k})$ [LJ16].

**Frank-Wolfe Method (FW)**

**Problem Class:**

$$\begin{array}{ll} \min_{\boldsymbol{x}} & f(\boldsymbol{x}), \\ \text{subject to} & \boldsymbol{x} \in \mathsf{C}. \end{array}$$

$f : \mathbb{R}^n \to \mathbb{R}$ convex, differentiable, $\nabla f(\boldsymbol{x})$ $L$-Lipschitz.
$\mathsf{C}$ a compact convex set.

**Basic Iteration:** Repeat

$$\begin{aligned} \boldsymbol{v}_k &\in \arg\min_{\boldsymbol{v} \in \mathsf{C}} \langle \boldsymbol{v}, \nabla f(\boldsymbol{x}_k) \rangle, \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \gamma_k (\boldsymbol{v}_k - \boldsymbol{x}_k), \end{aligned}$$

with $\gamma_k = \frac{2}{k+2}$.

**Convergence Guarantee:**

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_\star) \leq \frac{2L \operatorname{diam}^2(\mathsf{C})}{k+2}.$$

**Figure 8.6** An overview of the Frank-Wolfe Method.

We then set

$$\boldsymbol{x}_{k+1} = (1 - \gamma_k)\boldsymbol{x}_k + \gamma_k \boldsymbol{v}_k \in \mathsf{C}, \tag{8.6.7}$$

where $\gamma_k \in (0, 1)$ is a specially chosen step size. Figure 8.6 summarizes the properties of this method.

*Interpretation as Minimizing a First-Order Approximation.*
The Frank-Wolfe method can be interpreted as follows. At a given point $\boldsymbol{x}_k$, we form a first-order approximation to the objective function $f$:

$$f(\boldsymbol{v}) \approx \hat{f}(\boldsymbol{v}, \boldsymbol{x}_k) \doteq f(\boldsymbol{x}_k) + \langle \boldsymbol{v} - \boldsymbol{x}_k, \nabla f(\boldsymbol{x}_k) \rangle. \tag{8.6.8}$$

We minimize the approximation $\hat{f}(\boldsymbol{v}, \boldsymbol{x}_k)$ over $\boldsymbol{v} \in \mathsf{C}$ to produce $\boldsymbol{v}_k$. We then take a step in the direction $\boldsymbol{w}_k = \boldsymbol{v}_k - \boldsymbol{x}_k$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \gamma_k \boldsymbol{w}_k. \tag{8.6.9}$$

*Computing the Step Direction.*
The crucial subproblem in the Frank-Wolfe method involves minimizing a linear function over a compact convex set $\mathsf{C}$:

$$\min_{\boldsymbol{v} \in \mathsf{C}} \langle \boldsymbol{v}, \nabla f(\boldsymbol{x}) \rangle. \tag{8.6.10}$$

Depending on the constraint set $\mathsf{C}$, this could itself be a challenging (or even intractable!) optimization problem. Fortunately, for the problems of interest in

this book, this subproblem can be solved in an efficient and scalable manner. We give two examples below:

EXAMPLE 8.29 (Frank-Wolfe Subproblem over an $\ell^1$ Ball). *Given a vector $\boldsymbol{g}$, consider the problem*

$$\min_{\boldsymbol{v}} \langle \boldsymbol{v}, \boldsymbol{g} \rangle \quad \text{subject to} \quad \|\boldsymbol{v}\|_1 \le \tau. \tag{8.6.11}$$

*Let $i$ be any index for which $\boldsymbol{g}_i = \|\boldsymbol{g}\|_\infty$, and $\sigma_i = \operatorname{sign}(\boldsymbol{g}_i)$. Then (8.6.11) has a solution*

$$\boldsymbol{v}_\star = -\tau \sigma_i \boldsymbol{e}_i, \tag{8.6.12}$$

*where $\boldsymbol{e}_i$ is the $i$-th standard basis vector. The solution $\boldsymbol{v}_\star$ can be computed in linear time, simply by finding the largest magnitude entry of $\boldsymbol{g}$.*

EXAMPLE 8.30 (Frank-Wolfe Subproblem of a Nuclear Norm Ball). *Given a matrix $\boldsymbol{G}$, consider the problem*

$$\min_{\boldsymbol{V}} \langle \boldsymbol{V}, \boldsymbol{G} \rangle \quad \text{subject to} \quad \|\boldsymbol{V}\|_* \le \tau. \tag{8.6.13}$$

*Let $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* = \sum_{i=1}^{n_1} \boldsymbol{u}_i \sigma_i \boldsymbol{v}_i$ denote the singular value decomposition of $\boldsymbol{G}$. Then (8.6.13) has an optimal solution*

$$\boldsymbol{V}_\star = -\tau \boldsymbol{u}_1 \boldsymbol{v}_1^*. \tag{8.6.14}$$

*This optimal solution can be computed in time $O(n_1 n_2)$ by computing (only) the leading singular vector pair $(\boldsymbol{u}_1, \boldsymbol{v}_1)$ of $\boldsymbol{G}$, see Section 4.2.1 of Chapter 4 for details.*

The latter example illustrates the special virtue of the Frank-Wolfe method in nuclear norm minimization: the key subproblem only requires us to compute *one* singular value/vectors triple. For a problem involving $n_1 \times n_2$ matrices, this can be done in time $O(n_1 n_2)$ – a dramatic improvement over proximal gradient methods, which require a full singular value decomposition in each iteration.

This scalability comes at a price, though. Compared to accelerated proximal gradient methods, which converge at a rate of $O(1/k^2)$ in function values, the Frank-Wolfe method only achieves a rate of $O(1/k)$.[11] The following theorem gives a precise bound on the worst-case rate of convergence for Frank-Wolfe over the class of convex functions with Lipschitz gradient.

THEOREM 8.31 (Convergence of Frank-Wolfe). *Let $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$ denote the sequence of iterates generated by the Frank-Wolfe method, with step size $\gamma_k = \frac{2}{k+2}$. Then*

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_\star) \le \frac{2L\operatorname{diam}^2(\mathsf{C})}{k+2}. \tag{8.6.15}$$

[11] When the function is nonconvex, the worst convergence rate reduces to $O(1/\sqrt{k})$ [LJ16].

*Proof* For ease of notation, write $d = \operatorname{diam}(\mathsf{C})^2$, $\boldsymbol{x} = \boldsymbol{x}_k$, $\boldsymbol{x}^+ = \boldsymbol{x}_{k+1}$, $\gamma = \gamma_k$, and $\boldsymbol{v} = \boldsymbol{v}_k$. Note that

$$\boldsymbol{x}^+ - \boldsymbol{x} = \gamma\,(\boldsymbol{v} - \boldsymbol{x})\,. \tag{8.6.16}$$

Because $\nabla f(\boldsymbol{x})$ is $L$-Lipschitz, we can use the upper bound (8.2.8) to obtain

$$\begin{aligned} f(\boldsymbol{x}^+) &\le f(\boldsymbol{x}) + \big\langle \nabla f(\boldsymbol{x}), \boldsymbol{x}^+ - \boldsymbol{x} \big\rangle + \tfrac{L}{2} \big\| \boldsymbol{x}^+ - \boldsymbol{x} \big\|_2^2 \\ &\le f(\boldsymbol{x}) + \gamma \,\langle \nabla f(\boldsymbol{x}), \boldsymbol{v} - \boldsymbol{x} \rangle + \tfrac{\gamma^2 L}{2} \|\boldsymbol{v} - \boldsymbol{x}\|_2^2 \\ &\le f(\boldsymbol{x}) + \gamma \,\langle \nabla f(\boldsymbol{x}), \boldsymbol{v} - \boldsymbol{x} \rangle + \tfrac{\gamma^2 L}{2} d^2. \end{aligned} \tag{8.6.17}$$

Meanwhile, by convexity,

$$\begin{aligned} f(\boldsymbol{x}_\star) &\ge f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}_\star - \boldsymbol{x} \rangle \\ &\ge f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{v} - \boldsymbol{x} \rangle\,, \end{aligned} \tag{8.6.18}$$

where the final line follows because $\boldsymbol{v}$ is chosen to minimize $\langle \nabla f(\boldsymbol{x}), \boldsymbol{v} \rangle$. Combining these two inequalities, we find that

$$\langle \nabla f(\boldsymbol{x}), \boldsymbol{v} - \boldsymbol{x} \rangle \le -\Big( f(\boldsymbol{x}) - f(\boldsymbol{x}_\star) \Big), \tag{8.6.19}$$

whence, plugging into (8.6.17) and subtracting $f(\boldsymbol{x}_\star)$ from both sides, we obtain

$$f(\boldsymbol{x}^+) - f(\boldsymbol{x}_\star) \le (1 - \gamma) \Big( f(\boldsymbol{x}) - f(\boldsymbol{x}_\star) \Big) + \tfrac{\gamma^2}{2} L d^2. \tag{8.6.20}$$

We use this basic relationship together with an inductive argument to bound the rate of convergence of the Frank-Wolfe method. Let $\varepsilon_k$ denote the suboptimality (in function values) at iteration $k$:

$$\varepsilon_k = f(\boldsymbol{x}_k) - f(\boldsymbol{x}_\star). \tag{8.6.21}$$

Set $\gamma_k = \frac{2}{k+2}$, so $\gamma_0 = 1$. Applying (8.6.20), we find that

$$\varepsilon_1 \le \tfrac{1}{2} L d^2. \tag{8.6.22}$$

Suppose now that for $\ell = 1, \dots, k$, $\varepsilon_\ell \le \frac{2}{\ell+2} L d^2$. Applying (8.6.20) again, we find that

$$\begin{aligned} \varepsilon_{k+1} &\le \frac{k}{k+2} \varepsilon_k + \frac{2}{(k+2)^2} L d^2 \\ &\le \frac{k+1}{(k+2)^2} \times 2 L d^2 \\ &\le \frac{2 L d^2}{(k+1)+2}. \end{aligned} \tag{8.6.23}$$

Hence, the relationship $\varepsilon_\ell \le \frac{2}{\ell+2} L d^2$ holds for all iterations $\ell$, as claimed. □

### 8.6.2 Frank-Wolfe for Stable Matrix Completion

In the context of nuclear norm minimization, the above result can be viewed as follows: Frank-Wolfe allows us to derive methods that produce moderate-quality solutions to extremely large problems, for which methods with better worst cases rates simply take too long to compute even a single iteration. To be more specific, we in this section illustrate the general Frank-Wolfe method for the particular problem of recovering a low-rank matrix from incomplete and noisy observations

$$\boldsymbol{Y} = \mathcal{P}_\Omega[\boldsymbol{X}_o + \boldsymbol{Z}], \tag{8.6.24}$$

where $\boldsymbol{X}_o \in \mathbb{R}^{n_1 \times n_2}$ has low rank, $\boldsymbol{Z} \in \mathbb{R}^{n_1 \times n_2}$ is a matrix of small, dense noise, and $\Omega \subseteq [n_1] \times [n_2]$ is the set of observed entries. One approach to approximately recovering $\boldsymbol{X}_o$ is to minimize the reconstruction error over the set of all matrices of small nuclear norm:

$$\begin{array}{ll} \min & f(\boldsymbol{X}) \equiv \frac{1}{2} \left\| \mathcal{P}_\Omega[\boldsymbol{X}] - \boldsymbol{Y} \right\|_F^2, \\ \text{subject to} & \left\| \boldsymbol{X} \right\|_* \leq \tau. \end{array} \tag{8.6.25}$$

Here, the constraint $\left\| \boldsymbol{X} \right\|_* \leq \tau$ encourages $\boldsymbol{X}$ to have low rank. The constraint set $\mathsf{C} = \{\boldsymbol{X} \mid \left\| \boldsymbol{X} \right\|_* \leq \tau\}$ is closed and bounded. Moreover, the gradient

$$\nabla f(\boldsymbol{X}) = \mathcal{P}_\Omega \left[ \boldsymbol{X} - \boldsymbol{Y} \right] \tag{8.6.26}$$

is 1-Lipschitz, and so the Frank-Wolfe method indeed applies to this problem.

The key step in the Frank-Wolfe method is to minimize a linear function $\langle \boldsymbol{V}, \nabla f(\boldsymbol{X}) \rangle$ over the constraint set $\mathsf{C}$. As described above, this problem can be solved in closed form: if

$$\nabla f(\boldsymbol{X}) = \sum_{i=1}^{n_1} \boldsymbol{u}_i \sigma_i \boldsymbol{v}_i^* \tag{8.6.27}$$

is the singular value decomposition of $\nabla f$, then

$$-\tau \boldsymbol{u}_1 \boldsymbol{v}_1^* \in \arg\min_{\boldsymbol{V} \in \mathsf{C}} \langle \boldsymbol{V}, \nabla f(\boldsymbol{X}) \rangle \,. \tag{8.6.28}$$

The leading singular value/vectors can be extracted from the matrix $\nabla f(\boldsymbol{X})$ efficiently, without computing the entire SVD (8.6.27). Typically, this is done using the power method, which was described in some detail in Chapter 4 and Exercise 4.6.[12] To cleanly describe the method, we simply let

$$(\boldsymbol{u}_1, \sigma_1, \boldsymbol{v}_1) \doteq \text{LeadSV}(\boldsymbol{G}) \tag{8.6.29}$$

denote the operation which extracts a leading singular value/vectors triple from a matrix $\boldsymbol{G}$. Using this notation, the complete Frank-Wolfe algorithm for stable matrix completion is described in Algorithm 8.8.

The Frank-Wolfe method requires only a single singular value/vectors triple

[12] Or by the more efficient Lanczos method to be introduced in Section 9.3.2 of the next Chapter.

**Algorithm 8.8** Frank-Wolfe for Stable Matrix Completion

**Problem:** given $\boldsymbol{Y} \in \mathbb{R}^{n_1 \times n_2}$ and $\Omega \subseteq [n_1] \times [n_2]$,

$$\min_{\boldsymbol{X}} \tfrac{1}{2} \left\| \mathcal{P}_\Omega[\boldsymbol{X}] - \boldsymbol{Y} \right\|_F^2 \quad \text{subject to} \quad \|\boldsymbol{X}\|_* \leq \tau.$$

**Input:** $\boldsymbol{X}_0 \in \mathbb{R}^{n_1 \times n_2}$ satisfying $\|\boldsymbol{X}_0\|_* \leq \tau$.
**for** $(k = 0, 1, 2, \ldots, K-1)$ **do**
  $(\boldsymbol{u}_1, \sigma_1, \boldsymbol{v}_1) \leftarrow \text{LeadSV}\left(\mathcal{P}_\Omega\left[\boldsymbol{X}_k - \boldsymbol{Y}\right]\right)$.
  $\boldsymbol{V}_k \leftarrow -\tau \boldsymbol{u}_1 \boldsymbol{v}_1^*$.
  $\boldsymbol{X}_{k+1} \leftarrow \frac{k}{k+2} \boldsymbol{X}_k + \frac{2}{k+2} \boldsymbol{V}_k$.
**end for**
**Output:** $\boldsymbol{X}_\star \leftarrow \boldsymbol{X}_K$.

at each iteration. Moreover, since $\boldsymbol{V}_k = -\tau \boldsymbol{u}_1 \boldsymbol{v}_1^*$ has rank one, the rank of $\boldsymbol{X}_k$ increases by at most one at each iteration. In this sense, Frank-Wolfe can be viewed as a *greedy method*. It constructs a low-rank matrix $\boldsymbol{X}_\star$ by adding on one (optimally chosen) rank-one factor at a time.

### 8.6.3 Connection to Greedy Methods for Sparsity

In sparse and low-rank approximation, *greedy methods* are sometimes favored for their simplicity and scalability. For sparse approximation, the Frank-Wolfe method gives one such greedy algorithm. Consider the problem

$$\begin{array}{ll} \min_{\boldsymbol{x}} & f(\boldsymbol{x}) \equiv \frac{1}{2} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \right\|_2^2, \\ \text{subject to} & \|\boldsymbol{x}\|_1 \leq \tau. \end{array} \tag{8.6.30}$$

Notice that

$$\nabla f(\boldsymbol{x}) = \boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}). \tag{8.6.31}$$

The Frank-Wolfe subproblem

$$\min_{\boldsymbol{v}} \langle \boldsymbol{v}, \nabla f(\boldsymbol{x}) \rangle \quad \text{subject to} \quad \|\boldsymbol{v}\|_1 \leq \tau. \tag{8.6.32}$$

has an especially simple solution: letting $i$ be the index of the largest magnitude entry of $\nabla f$, and $\sigma$ its sign,

$$\boldsymbol{v}_\star = -\tau \sigma \boldsymbol{e}_i. \tag{8.6.33}$$

Algorithm 8.9 describes in detail the Frank-Wolfe method for problem (8.6.30). At each iteration, it increases the number of nonzero entries in the vector $\boldsymbol{x}$ by at most one, by adding on a multiple of $\boldsymbol{e}_{i_k}$. Let

$$\mathsf{I}_k = \{i_1, \ldots, i_{k-1}\} = \text{supp}(\boldsymbol{x}_k) \tag{8.6.34}$$

denote the collection of indices that have been chosen up to time $k$. We generate

**Algorithm 8.9** Frank-Wolfe for Noisy Sparse Recovery

**Problem:** given $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$,

$$\min_{\boldsymbol{x}} \tfrac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \quad \text{subject to} \quad \|\boldsymbol{x}\|_1 \leq \tau.$$

**Input:** $\boldsymbol{x}_0 \in \mathbb{R}^n$ satisfying $\|\boldsymbol{x}_0\|_1 \leq \tau$.
**for** $(k = 0, 1, 2, \ldots, K-1)$ **do**
  $\boldsymbol{r}_k \leftarrow \boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}$.
  $i_k \leftarrow \arg\max_i |\boldsymbol{a}_i^* \boldsymbol{r}_k|$.
  $\sigma \leftarrow \operatorname{sign}\left(\boldsymbol{a}_{i_k}^* \boldsymbol{r}_k\right)$.
  $\boldsymbol{v}_k \leftarrow -\tau\sigma\boldsymbol{e}_{i_k}$.
  $\boldsymbol{x}_{k+1} \leftarrow \frac{k}{k+2}\boldsymbol{x}_k + \frac{2}{k+2}\boldsymbol{v}_k$.
**end for**
**Output:** $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

$\mathsf{I}_{k+1}$ from $\mathsf{I}_k$ by introducing a (potentially) new index

$$\mathsf{I}_{k+1} = \mathsf{I}_k \cup \{i_k\}\,. \tag{8.6.35}$$

This new index is chosen according to the largest-magnitude entry in the gradient $\nabla f$. Write

$$\boldsymbol{A} = [\boldsymbol{a}_1 \mid \cdots \mid \boldsymbol{a}_n] \tag{8.6.36}$$

for the columns of $\boldsymbol{A}$, and let

$$\boldsymbol{r}_k = \boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y} \tag{8.6.37}$$

denote the measurement residual at point $\boldsymbol{x}_k$. Since $\nabla f(\boldsymbol{x}_k) = \boldsymbol{A}^* \boldsymbol{r}_k$, the Frank-Wolfe method chooses the index $i_k$ corresponding to the column $\boldsymbol{a}_{i_k}$ that is most correlated with the residual $\boldsymbol{r}_k$.

*Matching Pursuit.*
A number of classical *greedy methods* for sparse approximation have this basic structure. A canonical example is the *Matching Pursuit* algorithm [MZ93]. This algorithm generates a sequence of iterates $\boldsymbol{x}_0 = \boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$, by repeatedly choosing a column of $\boldsymbol{a}_{i_k}$ of $\boldsymbol{A}$ that is most correlated with the residual $\boldsymbol{r}_k$. Similar to Frank-Wolfe, Matching Pursuit[13] sets

$$i_k = \arg\max_i \left| [\nabla f(\boldsymbol{x}_k)]_i \right| = \arg\max_i \left| \boldsymbol{a}_i^* \boldsymbol{r}_k \right|. \tag{8.6.38}$$

However, rather than stepping a predetermined length in the $\boldsymbol{e}_{i_k}$ direction, Matching Pursuit chooses the step size $t_k$ by solving a one-dimensional mini-

[13] Despite the strong parallels to the Frank-Wolfe method, Matching Pursuit was motivated independently from a rather different perspective for solving a more specific class of signal processing tasks [MZ93].

**Algorithm 8.10** Matching Pursuit for Sparse Approximation

**Problem:** find a sparse $\boldsymbol{x}$ such that $f(\boldsymbol{x}) \equiv \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$ is small.
$\boldsymbol{x}_0 \leftarrow \boldsymbol{0}$.
**for** $(k = 0, 1, 2, \ldots, K-1)$ **do**
  $\boldsymbol{r}_k \leftarrow \boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}$.
  $i_k \leftarrow \arg\max_i |\boldsymbol{a}_i^* \boldsymbol{r}_k|$.
  $t_k \leftarrow -\frac{\langle \boldsymbol{a}_{i_k}, \boldsymbol{r}_k \rangle}{\|\boldsymbol{a}_{i_k}\|_2^2}$.
  $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + t_k \boldsymbol{e}_{i_k}$.
**end for**
**Output:** $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

mization problem:

$$t_k = \arg\min_t f\left(\boldsymbol{x}_k + t\boldsymbol{e}_{i_k}\right) \;=\; -\frac{\langle \boldsymbol{a}_{i_k}, \boldsymbol{r}_k \rangle}{\|\boldsymbol{a}_{i_k}\|_2^2}. \tag{8.6.39}$$

This can be viewed as a form of *exact line search* and typically leads to more rapid convergence in practice. The overall Matching Pursuit algorithm is specified as Algorithm 8.10.

*Orthogonal Matching Pursuit.*
Matching pursuit achieves better convergence by choosing the step size $t_k$ in an optimal manner. Since

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_k \boldsymbol{e}_{i_k}, \tag{8.6.40}$$

this is equivalent to making an optimal choice of the $i_k$-th entry in $\boldsymbol{x}_{k+1}$, while leaving all of the other entries fixed. It is possible to further improve the rate of convergence of this approach by choosing *all* of the nonzero entries of $\boldsymbol{x}_{k+1}$ optimally (rather than just the $i_k$-th entry). In notation, let $\mathsf{I}_k = \{i_1, i_2, \ldots, i_{k-1}\}$ denote the collection of indices that have been chosen up to step $k$. The *Orthogonal Matching Pursuit* method [PRK93, TG07] selects an index $i_k$ that maximizes the correlation $|\boldsymbol{a}_i^* \boldsymbol{r}_k|$ of a column of $\boldsymbol{A}$ with the residual $\boldsymbol{r}_k = \boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}$. It sets $\mathsf{I}_{k+1} = \mathsf{I}_k \cup \{i_k\}$, and then updates all of the nonzero entries in $\boldsymbol{x}$ by setting

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \; \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \quad \text{subject to} \quad \operatorname{supp}(\boldsymbol{x}) \subseteq \mathsf{I}_{k+1}. \tag{8.6.41}$$

This problem can be solved in closed form:

$$[\boldsymbol{x}_{k+1}]_{\mathsf{I}_{k+1}} = \left(\boldsymbol{A}_{\mathsf{I}_{k+1}}^* \boldsymbol{A}_{\mathsf{I}_{k+1}}\right)^{-1} \boldsymbol{A}_{\mathsf{I}_{k+1}}^* \boldsymbol{y}, \tag{8.6.42}$$

$$[\boldsymbol{x}_{k+1}]_{\mathsf{I}_{k+1}^c} = \boldsymbol{0}. \tag{8.6.43}$$

The name *Orthogonal* Matching Pursuit comes from the observation that the residual

$$\boldsymbol{r}_{k+1} = \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y} = \left(\boldsymbol{A}_{\mathsf{I}_{k+1}} \left(\boldsymbol{A}_{\mathsf{I}_{k+1}}^* \boldsymbol{A}_{\mathsf{I}_{k+1}}\right)^{-1} \boldsymbol{A}_{\mathsf{I}_{k+1}}^* - \boldsymbol{I}\right) \boldsymbol{y} \tag{8.6.44}$$

**Algorithm 8.11** Orthogonal Matching Pursuit for Sparse Approximation

**Problem:** find a sparse $\boldsymbol{x}$ such that $f(\boldsymbol{x}) \equiv \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$ is small.
$\boldsymbol{x}_0 \leftarrow \boldsymbol{0}$, $\mathsf{I}_0 \leftarrow \varnothing$.
**for** $(k = 0, 1, 2, \ldots, K-1)$ **do**
  $\boldsymbol{r}_k \leftarrow \boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}$.
  $i_k \leftarrow \arg\max_i |\boldsymbol{a}_i^* \boldsymbol{r}_k|$.
  $\mathsf{I}_{k+1} \leftarrow \mathsf{I}_k \cup \{i_k\}$.
  $[\boldsymbol{x}_{k+1}]_{\mathsf{I}_{k+1}} \leftarrow \left(\boldsymbol{A}_{\mathsf{I}_{k+1}}^* \boldsymbol{A}_{\mathsf{I}_{k+1}}\right)^{-1} \boldsymbol{A}_{\mathsf{I}_{k+1}}^* \boldsymbol{y}$.
  $[\boldsymbol{x}_{k+1}]_{\mathsf{I}_{k+1}^c} \leftarrow \boldsymbol{0}$.
**end for**
**Output:** $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

is orthogonal to the range range$\left(\boldsymbol{A}_{\mathsf{I}_{k+1}}\right)$ of the dictionary columns selected through the first $k+1$ iterations.

The overall Orthogonal Matching Pursuit algorithm is given as Algorithm 8.11. This method is sometimes favored by practitioners due to its simplicity, and the fact that it maintains an explicit *active set* $\mathsf{I}_k$. The latter property is useful for problems in which the support of the sparse solution $\boldsymbol{x}_\star$ is the object of interest.[14]

Although OMP has many variants and extensions, it was originally derived for the specific problem of finding sparse near-solutions to a linear system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$. Like $\ell^1$ minimization, OMP is guaranteed to succeed whenever $\boldsymbol{y}$ is generated from some sufficiently sparse $\boldsymbol{x}_o$ and the columns of $\boldsymbol{A}$ are sufficiently spread in the high-dimensional space $\mathbb{R}^m$. In particular:

THEOREM 8.32 (Convergence of Orthogonal Matching Pursuit). *Suppose that* $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o$, *with*

$$k = \|\boldsymbol{x}_o\|_0 \leq \frac{1}{2\mu(\boldsymbol{A})}. \tag{8.6.45}$$

*Then after* $k$ *iterations, the OMP algorithm terminates with* $\boldsymbol{x}_k = \boldsymbol{x}_o$ *and* $\mathsf{I}_k = \operatorname{supp}(\boldsymbol{x}_o)$.

Exercise 8.12 guides the reader through a proof of Theorem 8.32. The key message of the proof is that under the conditions of the theorem, at each iteration $\ell$ the algorithm selects an index $i_\ell$ that belongs to the true support set $\operatorname{supp}(\boldsymbol{x}_o)$.

The form of Theorem 8.32 can be directly compared to that of Theorem 3.3 of Chapter 3. These results imply that *both* OMP and $\ell^1$ minimization recover $\boldsymbol{x}_o$ whenever $\|\boldsymbol{x}_o\|_0 \leq 1/2\mu(\boldsymbol{A})$. Hence, at an intuitive level, both methods succeed whenever the target solution is sparse and the matrix $\boldsymbol{A}$ is "nice".

[14] For example, in the RF spectrum sensing discussed in Chapter 11, the goal is to determine which bands of the RF spectrum are occupied, in order to avoid interference. The specific energy levels within these bands are of secondary importance.

However, as shown in Chapter 3, the incoherence condition requires $\boldsymbol{x}_o$ to be extremely sparse. $\ell^1$ minimization also recovers denser $\boldsymbol{x}_o$ under the stronger condition that $\boldsymbol{A}$ satisfies the Restricted Isometry Property $\delta(\boldsymbol{A}) < c$. While various improved analyses of OMP are available, the RIP is not sufficient for OMP to succeed. In this sense, convex relaxation achieves a better uniform guarantee. However, OMP can be modified to also guarantee sparse recovery under the RIP. The key ideas are to allow the algorithm to *remove* elements from the active set $\mathsf{I}_k$ at each iteration, and to add multiple elements. The resulting method, called Compressed Sampling Matching Pursuit (COSAMP) [NT09] is described in more detail in Exercise 8.13. The large and varied literature on greedy algorithms also includes greedy methods for more general problems such as low-rank recovery, as we will give more references in the final Notes section.

### 8.6.4 Stochastic Gradient Descent for Finite Sum

The type of optimization we often encounter is of the type:

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \tag{8.6.46}$$

where $f(\boldsymbol{x})$ is typically the measurement error term, also known as the "data" term, say $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$, and $g(\boldsymbol{x})$ is typically a regularization for promoting certain low-dimensional structure on the solution $\boldsymbol{x}$, also known as the "model" term, say the $\ell^1$ norm $\|\boldsymbol{x}\|_1$ for a vector or the nuclear norm for a matrix. As we have seen in the previous section, to strive for better scalability, the Frank-Wolfe method exploits the (compositional) structure in the term $g(\boldsymbol{x})$, by restricting the search for good descent direction to a small set of coordinates or directions. See Section D.4 of the Appendix B for a more explicit scheme, known as *block coordinate descent*, to exploit such structures. Such schemes typically allow us to reduce the dependency of algorithmic complexity on the dimension $n$, from $O(n)$ to sublinear in $n$, say[15]

$$O(n) \to O(n^{1/2}).$$

A remaining question is whether there are good structures in the data term $f(\boldsymbol{x})$ that can be exploited too for better scalability. Indeed, in many problems that arise in compressive sensing or machine learning, the data term is typically a *finite sum* of (statistically independent) terms, say measurement errors. That is, $f(\boldsymbol{x})$ is typically of the form:

$$f(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^{m} h_i(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \tag{8.6.47}$$

where each $h_i(\boldsymbol{x})$ is an independent sample of the function $f(\boldsymbol{x})$ hence $\mathbb{E}[h_i(\boldsymbol{x})] =$

[15] For example, in the case of recovering an $n_1 \times n_2$ low-rank matrix, the Frank-Wolfe method reduces the dependency from $O(n_1 \times n_2)$ to $O(n_1 + n_2)$.

$f(\boldsymbol{x})$. For example, for $m$ measurements of a sparse vector $\boldsymbol{x}$: $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \in \mathbb{R}^m$, we can write the data fitting term as:

$$\frac{1}{m}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = \frac{1}{m}\sum_{i=1}^{m}(y_i - \boldsymbol{a}_i^*\boldsymbol{x})^2, \tag{8.6.48}$$

where $\boldsymbol{a}_i^*$ is the $i$-th row of $\boldsymbol{A}$. This is also the case in many machine learning problems, where the total loss to minimize is a sum of logistic or $\ell^p$ losses for a large set of training samples.

Notice when the number of samples $m$ is very large, even gradient descent algorithm can be very expensive: to evaluate the gradient of $f(\boldsymbol{x})$, the complexity is typically linear in the number of samples, i.e., of the order $O(m)$. To further reduce the complexity's dependency on $m$, one key idea is to use the so-called *stochastic gradient descent (SGD)* method [RM51, Bot10]. That is, at east iteration $k$, instead of using all the $m$ samples to compute the full gradient $\nabla f(\boldsymbol{x})$, we compute the gradient approximately with a random batch of samples $B_k \subset [m]$ of a fixed size $b \ll m$:

$$f_k(\boldsymbol{x}) \doteq \frac{1}{b}\sum_{i\in B_k} h_i(\boldsymbol{x}), \quad \nabla f_k(\boldsymbol{x}) \doteq \frac{1}{b}\sum_{i\in B_k} \nabla h_i(\boldsymbol{x}). \tag{8.6.49}$$

We then use this approximate gradient to replace the full gradient in the descent scheme, leading to the stochastic gradient descent (SGD) scheme:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_k \nabla f_k(\boldsymbol{x}_k). \tag{8.6.50}$$

This reduces the computational cost of each iteration to be $O(n)$. Following our proofs for the gradient descent, using the fact $\mathbb{E}[\nabla f_k(\boldsymbol{x})] = \nabla f(\boldsymbol{x})$, it is easy to show that the expected value of the objective function $\mathbb{E}[f(\boldsymbol{x}_k)]$ converges using the stochastic gradient descent scheme.

However, despite high scalability, SGD has poor convergence rate due to a constant variance of the stochastic gradient $\mathbb{E}[\|\nabla f_k(\boldsymbol{x}) - \nabla f(\boldsymbol{x})\|] > 0$. To improve the convergence behavior of SGD, several methods of *variance reduced* SGD have been developed in the past decade or so [JZ13, DBLJ14, LMH15, AZ17]. In such *variance reduction* methods, instead of directly using $\nabla f_k(\boldsymbol{x})$, one computes a full gradient $\nabla f(\tilde{\boldsymbol{x}})$ at an anchor point $\tilde{\boldsymbol{x}}$ beforehand. Then one uses the following variance reduced gradient

$$\tilde{\nabla} f_k(\boldsymbol{x}) \doteq \nabla f_k(\boldsymbol{x}) - \nabla f_k(\tilde{\boldsymbol{x}}) + \nabla f(\tilde{\boldsymbol{x}}) \tag{8.6.51}$$

as a proxy for the full gradient $\nabla f(\boldsymbol{x})$ during each iteration:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_k \tilde{\nabla} f_k(\boldsymbol{x}_k). \tag{8.6.52}$$

As a result, the amortized per-iteration cost is still the same as SGD. However, the variance reduced gradient (8.6.51) is unbiased and can reduce the variance from $\mathbb{E}[\|\nabla f_k(\boldsymbol{x}) - \nabla f(\boldsymbol{x})\|]$ to $\mathbb{E}[\|\nabla f_k(\boldsymbol{x}) - \nabla f_k(\tilde{\boldsymbol{x}})\|]$. In theory, the variance $\mathbb{E}[\|\nabla f_k(\boldsymbol{x}) - \nabla f_k(\tilde{\boldsymbol{x}})\|]$ can vanish asymptotically, thus the convergence rate of SGD can be substantially improved.

Roughly speaking, these methods can reduce the variance of stochastic gradient by exploiting the structure of finite sum. With variance reduction, they have the same per-iteration cost with SGD in the amortized sense whereas they can achieve better total complexity than the gradient descent method in terms of dependence on the number of samples, typically from $O(m)$ to sublinear in $m$, say

$$O(m) \to O(m^{1/2}).$$

All of these methods can work with the Nesterov's acceleration schemes introduced in earlier sections and can be extended to the nonsmooth setting for structured signal recovery [DBLJ14, XZ14, AZ17]. More specifically, to achieve a prescribed accuracy in the objective function, say $|f(\boldsymbol{x}_k) - f(\boldsymbol{x}_\star)| \leq \varepsilon$, instead of the generic rate $O(\varepsilon^{-2})$ of stochastic gradient descent for general convex functions, one can achieve the accelerated rate of

$$O(\varepsilon^{-2}) \to O(\varepsilon^{-1/2}).$$

The recent work [SJM20b], which combines the variance reduction and acceleration methods, has achieved an overall computational complexity that practically meets the theoretical lower bound for this class of finite-sum problems. In addition, the variance reduced SGD can also be used in conjunction with the Frank-Wolfe method to simultaneously exploit the finite-sum structure and low-dimensional structure for even better scalability, for example see the work [HL16].

## 8.7 Notes

*Greedy Algorithms.*
The name *basis pursuit* was first suggested by Chen and Donoho in their early work on recovering sparse representation [Che95, CDS01]. Many greedy algorithms such as Matching Pursuit (MP) [MZ93] were first used to solve the associated optimization problems, for an incoherent measurement matrix. The original idea of Orthogonal Matching Pursuit (OMP) can be traced to the work [PRK93] on wavelets in early 1990's and was later reintroduced by [TG07] to the problem of compressed sensing with random measurements. The OMP algorithm was later improved by [NT09] as the (Compressed Sampling Matching Pursuit) COSAMP algorithm which works for measurement matrices with the RIP property. As we have seen in this chapter, these greedy algorithms bear great resemblance to the Franke-Wolfe algorithm [FW56] developed in 1950's.

*Convex Optimization Approach.*
An almost parallel line of study strives to develop efficient algorithms based on convex optimization. The study of proximal operator for various convex functions can be traced to the work of Moreau in 1960's [Mor62]. The Iterative Shrinkage Threshold Algorithm (ISTA), with its early roots tracing back

to [Tib96], has been studied under different names, such as forward-backward splitting [CW05], thresholded Landweber [DDM04], and separable approximation (SpaRSA) [WNF08]. The Fast Iterative Shrinkage Threshold Algorithm (FISTA), based on Nesterov's acceleration technique [Nes83], was introduced later by [BT09].

*Large-Scale Algorithmic Implementations.*
The methods featured in this Chapter aim to elucidate main ideas and techniques for solving the BP (8.1.1) or Lasso (8.1.2) type programs. The algorithms given in here can already solve problems of moderate size efficiently. Nevertheless, for very large-scale problems, say with $\boldsymbol{x}$ being hundreds of millions in dimension. One could resort to more scalable approaches. For example, one can screen the variables in $\boldsymbol{x}$ so that we do not have to work with all variables at once. For instance, for the Lasso-type (or any $\ell^1$ regularized convex) problems, more careful studies of the primal dual variables lead to efficient screening rules [TBF+12, GVR12]. Based on different screening strategies, one can subsequently develop more scalable greedy algorithms, including sequential screening methods [WZL+14] or dynamical screening methods [NFGS15]. Another related strategy is to maintain and update a relatively small working or active set according to some violation rules [JG15]. This has also led to some of the more recent scalable algorithms such as the BLITZ [JG15] and CELER [MSG18].

*Convergence of ALM and ADMM.*
The convergence of ALM and ADMM type algorithms has been studied long in history (see e.g. [Hes69, Roc73, Pow69] for ALM and [LM79, KM89] for ADMM). Like the ALM method, the most natural approach is to recognize ADMM as some known algorithm applied to the dual. In fact, ADMM turns out to be equivalent to Douglas-Rachford splitting applied to the dual. For more details on this, see [EB92, CW05]. We recommend [Eck12] for a tutorial introduction to a more formal convergence analysis of ADMM. For more recent analysis of generalized version of ADMM including their convergence rates, we recommend the work of [DY16]. ADMM has also been widely applied to problems when the number of terms are more than three [BPC+11]. The convergence analysis of ADMM with more than three terms is much more difficult and it has been shown to diverge in many cases.

The proof given in this book follows the framework of [HY12] which was applied to the linear equality case by [Xu17]. We have seen that *monotone operators* play a powerful role in providing unified convergence analysis for both ALM and ADMM. In fact, monotone operators not only help with the convergence analysis. They may also lead to rather unified ways of algorithm design for convex optimization, by interpreting the optimal solution as the fixed point to certain contracting mappings associated with the monotone operators of the Lagrangian. The reader may refer to the recent manuscript [RB16] for a more systematic survey on this method.

*Exploit Structures in Data Measurements.*
In this chapter, the algorithms developed typically treat the data-fitting term for the measurement $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ as a general smooth convex function. The convergence rates of all the algorithms are characterized under this (somewhat unnecessarily) general assumption. For instance, the rate $O(1/k)$ for ALM and ADMM, proven in Theorem 8.27 and 8.28, is actually optimal for this class of problems for first-order methods, according to [Nes03, Nem04, OX18]. However, in the compressed sensing setting, the data matrix is often a random measurement matrix and thus is full rank and well-conditioned. This property induces implicit *strong convexity* in the data fitting term. Recent work [SJM20a] has shown that, somewhat surprisingly, for this class of problems, the bound $O(1/k)$ for ALM-like algorithms can be broken and one can obtain accelerated algorithms that achieve an improved convergence rate of $O(1/k^2 \log k)$.

*Exploit Structures in Sparsity-Promoting Norms.*
In this chapter, we have mainly used the model problems of recovering sparse signals and/or low-rank matrices to introduce key algorithmic ideas that lead to provably efficient and effective algorithms for convex optimization. We only customize all the general algorithms to the $\ell^1$ norm and the nuclear norm. As we have alluded to in Chapter 6, there are many other norms that promote a broader family low-dimensional structures. In particular, the so-called *group sparsity* norms can be used to promote various sparse patterns in signals and images. One may develop efficient optimization algorithms that are specially tailored to such norms. Interested readers may refer to the manuscript [BJMO12] on this topic.

## 8.8 Exercises

8.1 (Proximal Operators). *Prove the first and the third assertions of Proposition 8.4.*

8.2 (Average Proximal Operator). *Given multiple matrices* $\{\boldsymbol{W}_i \in \mathbb{R}^{m\times n}\}_{i=1}^k$, *show that we have:*

$$\mathit{soft}\Big(\frac{1}{k}\sum_{i=1}^k \boldsymbol{W}_i, \frac{\lambda}{k}\Big) = \arg\min_{\boldsymbol{X}} \|\boldsymbol{X}\|_* + \frac{1}{2}\sum_{i=1}^k \|\boldsymbol{X} - \boldsymbol{W}_i\|_F^2. \tag{8.8.1}$$

*This can be viewed as the proximal to find a low-rank matrix such that the averaged squared Frobenius norms to multiple matrices are small.*

8.3 (Hybrid Singular Value Thresholding). *Consider a matrix* $\boldsymbol{W}$ *of rank* $r$ *and with singular values* $\{\sigma_i\}_{i=1}^r$ *in descending order. Let* $h : \mathbb{R} \to \mathbb{R}_+$ *be an increasing function and* $h(0) \le 1$.

1 *Show that, given any* $\lambda \in (0, \sigma_1)$*, there exists a unique integer* $j \in [1, r]$ *such that the solution* $t_j$ *to the following equation:*

$$h\Big(\sum_{i=1}^{j} \sigma_i - j t_j\Big) = \frac{t_j}{\lambda}$$

*satisfies the condition:*

$$\sigma_{j+1} \leq t_j < \sigma_j.$$

2 *Design an algorithm that can compute this unique* $j$ *and* $t_j$ *efficiently. Notice that the worst you can do is to do a sequential search for all* $j$*'s.*

*Denote this unique solution as* $t_j^*(\lambda)$*, and this gives a so-called* hybrid thresholding operator *on the matrix* $\boldsymbol{W}$ *with singular value decomposition* $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$*:*

$$\mathcal{H}(\boldsymbol{W}, \lambda) = \boldsymbol{U} soft\big(\boldsymbol{\Sigma}, t_j^*(\lambda)\big)\boldsymbol{V}^*. \tag{8.8.2}$$

8.4 (Proximal Operator for Function of Nuclear Norm). *Let* $f : \mathbb{R} \to \mathbb{R}$ *be any convex and differentiable function with an increasing derivative* $f'(x)$ *and* $f'(0) \leq 1$*. Then given any matrix* $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ *and a* $\lambda > 0$*, we have*

$$\mathcal{H}(\boldsymbol{W}, \lambda) = \arg\min_{\boldsymbol{X}} \lambda f(\|\boldsymbol{X}\|_*) + \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\|_F^2, \tag{8.8.3}$$

*where* $\mathcal{H}(\boldsymbol{W}, \lambda)$ *is the hybrid thresholding operator defined in the previous exercise. Notice that the squared nuclear norm* $\|\boldsymbol{X}\|_*^2$ *or exponential* $e^{\|\boldsymbol{X}\|_*}$ *discussed in Example 8.5 are all special cases to the above result.*

8.5. *Given multiple matrices* $\{\boldsymbol{W}_i \in \mathbb{R}^{m \times n}\}_{i=1}^k$*, consider a function* $f$ *of the same property as in the previous exercise. Then we have:*

$$\mathcal{H}\Big(\frac{1}{k}\sum_{i=1}^{k} \boldsymbol{W}_i, \frac{\lambda}{k}\Big) = \arg\min_{\boldsymbol{X}} \lambda f(\|\boldsymbol{X}\|_*) + \frac{1}{2}\sum_{i=1}^{k} \|\boldsymbol{X} - \boldsymbol{W}_i\|_F^2. \tag{8.8.4}$$

8.6 (Iterative Soft-Thresholding Algorithm for PCP). *Regarding solving the stable principal component pursuit program using proximal gradient descent:*

1 *Apply the proximal gradient method to the PCP program. Based on separability of the two nonsmooth terms in the objective function, write down the corresponding proximal operators and the associated* $\boldsymbol{w}_1$ *and* $\boldsymbol{w}_2$*. Justify the updates for* $\boldsymbol{L}_{k+1}$ *and* $\boldsymbol{S}_{k+1}$ *in Algorithm 8.2.*
2 *Code a MATLAB function that implements the Iterative Soft-Thresholding Algorithm 8.2 for PCP, and demonstrated it on synthetic problem instances in which the data are superpositions of low-rank and sparse matrices.*

8.7 (Lasso and Elastic Net). *Use the PG and APG methods to solve the following two problems:*

1 *Suppose the observation* $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o + \boldsymbol{n}$, *where* $\boldsymbol{x}_o$ *is sparse and* $n$ *is some noise. Given* $\boldsymbol{y}$ *and* $\boldsymbol{A}$, *we would like to solve the* Lasso *problem of the following form*

$$\min_{\boldsymbol{x}} \underbrace{\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2}_{f(\boldsymbol{x})} + \underbrace{\lambda \|\boldsymbol{x}\|_1}_{g(\boldsymbol{x})},$$

*to approximately recover the sparse vector* $\boldsymbol{x}_o$. *For the PG and APG with a constant step size, compute the step size based on the Lipschitz constant of* $\nabla f$. *Report your chosen step size for the proximal gradient in analytic form (i.e with respect to derived Lipschitz constant). For each method implemented, report the number of iterations needed for convergence, norm difference with respect to the ground truth, a plot of objective value convergence in log scale and the run time.*

2 *Furthermore, consider the following* elastic net *problem:*

$$\min_{\boldsymbol{x}} \underbrace{\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \mu\|\boldsymbol{x}\|_2^2}_{f(\boldsymbol{x})} + \underbrace{\lambda \|\boldsymbol{x}\|_1}_{g(\boldsymbol{x})},$$

*where* $f$ *is* $\mu$*-strongly convex for* $\mu > 0$. *Use PG and APG to solve the problem and conduct similar reports as those for the previous problem.*

8.8 (Implementation: Augmented Lagrange Multiplier Algorithm for PCP). *Derive an algorithm for the (equality constrained) principal component pursuit problem,*

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 \quad \textit{subject to} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}. \tag{8.8.5}$$

*This algorithm will solve a sequence of unconstrained problems; sketch how these problems can be solved using (accelerated) proximal gradient. Which solver do you expect to be more efficient, the one you've derived in this exercise, or a solver based on the alternating directions method of multipliers, which alternates between* $\boldsymbol{L}$ *and* $\boldsymbol{S}$*?*

8.9 (Data Self-Expressive Representation). *In many data processing problems such as subspace clustering [VMS16], the inter-relationships among all the data are best revealed through using data to represent (or regress) themselves. More precisely, given a set of data points* $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m] \in \mathbb{R}^{n\times m}$, *we try to represent every data point as the (sparse) linear combination of other data points as:*

$$\boldsymbol{X} = \boldsymbol{X}\boldsymbol{C} \tag{8.8.6}$$

*where* $\boldsymbol{C} \in \mathbb{R}^{m\times m}$ *is a matrix of coefficients. Since we do not want the pathological solution where every point is represented by itself, we can enforce the diagonal entries of* $\boldsymbol{C}$ *to be zero:* $\mathrm{diag}(\boldsymbol{C}) = \boldsymbol{0}$. *In addition, we like to represent each point with the fewest points hence we prefer the sparse solution for* $\boldsymbol{C}$. *This is particularly the case when the data lie on low-dimensional structures such as a*

*union subspaces, or approximately so for submanifolds. This entails us to solve the following program:*

$$\min \|\boldsymbol{C}\|_1 \quad \textit{subject to} \quad \boldsymbol{X} = \boldsymbol{X}\boldsymbol{C},\ \textit{diag}(\boldsymbol{C}) = \boldsymbol{0}. \tag{8.8.7}$$

*Use the techniques provided in this chapter and write an algorithm to solve this problem.*

*One may also interpret the data points as nodes of a graph and the coefficient matrix $\boldsymbol{C}$ as a matrix of the "state transition" probability. In this case, if the data form certain "clusters" or "communities", we may expect the matrix $\boldsymbol{C}$ to be low rank [LLY+13]. Replace the $\ell^1$ norm in the above problem and write an algorithm to solve the following program:*

$$\min \|\boldsymbol{C}\|_* \quad \textit{subject to} \quad \boldsymbol{X} = \boldsymbol{X}\boldsymbol{C},\ \textit{diag}(\boldsymbol{C}) = \boldsymbol{0}. \tag{8.8.8}$$

8.10 (Unconstrained Problems via Frank-Wolfe). *Consider an unconstrained optimization problem of the form*

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) + g(\boldsymbol{x}), \tag{8.8.9}$$

*with $f$ differentiable with Lipschitz continuous gradient. Derive a Frank-Wolfe like method for this problem by instead solving*

$$\min_{\boldsymbol{x},t} f(\boldsymbol{x}) + t \quad \textit{subject to} \quad g(\boldsymbol{x}) \le t,\ t \le t_0, \tag{8.8.10}$$

*where $t_0$ is an upper bound on $g(\boldsymbol{x}_\star)$ at any optimal solution $\boldsymbol{x}_\star$ (you may assume that $t_0$ is provided by the user).*

8.11 (Sparse and Low-Rank via Frank-Wolfe). *Consider the constrained problem*

$$\min f(\boldsymbol{L},\boldsymbol{S}) \equiv \tfrac{1}{2}\|\boldsymbol{Y} - \boldsymbol{L} - \boldsymbol{S}\|_F^2 \quad \textit{subject to} \quad \|\boldsymbol{L}\|_* \le \tau_L,\ \|\boldsymbol{S}\|_1 \le \tau_S. \tag{8.8.11}$$

*Derive a Frank-Wolfe algorithm for solving this problem. By how much can the rank of $\boldsymbol{L}$ increase at each iteration? By how much can the number of nonzeros in $\boldsymbol{S}$ increase at each iteration?*

*Suppose we modify the algorithm by, after each Frank-Wolfe iteration, taking a projected gradient step*

$$\boldsymbol{S}^+ = \mathcal{P}_{\|\boldsymbol{S}\|_1 \le \tau_S}\left[\boldsymbol{S} - \tfrac{1}{L}\nabla_{\boldsymbol{S}} f(\boldsymbol{L},\boldsymbol{S})\right] \tag{8.8.12}$$

*where $L$ is a Lipschitz constant of the gradient of $f$. What are some potential advantages of this hybrid method, in terms of the number of iterations required to converge?*

8.12 (Sparse Recovery by Orthogonal Matching Pursuit). *The goal of this exercise is to prove Theorem 8.32, which shows that OMP correctly recovers any target sparse solution $\boldsymbol{x}_o$ with $k = \|\boldsymbol{x}_o\|_0 \le \frac{1}{2\mu(\boldsymbol{A})}$. Let $\mathsf{I} = \operatorname{supp}(\boldsymbol{x}_o)$.*

- OMP selects a true support index in the first iteration. *Let* $i_{\max}$ *index a maximum-magnitude entry of* $\boldsymbol{x}_o$*, i.e.,* $\boldsymbol{x}_o(i_{\max}) = \|\boldsymbol{x}_o\|_\infty$*. Using the incoherence of* $\boldsymbol{A}$*, argue that*

$$\left|\boldsymbol{a}_{i_{\max}}^* \boldsymbol{r}_0\right| \geq \left|\boldsymbol{a}_j^* \boldsymbol{r}_0\right| \quad \forall\, j \in \mathsf{I}^c. \tag{8.8.13}$$

- *Argue by induction that OMP selects some* $i_\ell \in \mathsf{I}$ *for every iteration* $\ell = 0, \ldots, k-1$*.*
- *Using the fact that* $\boldsymbol{r}_\ell \perp \operatorname{span}(\boldsymbol{A}_{\mathsf{I}_\ell})$*, argue that OMP selects a* new *index* $i_\ell \in \mathsf{I}$ *at each iteration* $\ell = 0, \ldots, k-1$*. Conclude that OMP terminates with* $\boldsymbol{x}_k = \boldsymbol{x}_o$ *and* $\mathsf{I}_k = \mathsf{I}$*, as claimed.*

8.13 (Greedy Methods that Succeed Under RIP). *The Compressive Sampling Matching Pursuit (COSAMP) algorithm modifies OMP by adding and subtracting multiple indices from the active set* $\mathsf{I}_\ell$ *at each iteration* $\ell$*. This algorithm takes as input a target number of nonzero entries,* $s$*, and modifies OMP as follows:*

- *At iteration* $\ell$*, let* $\mathsf{I}_{1/2}$ *be the support of the largest* $2s$ *entries of* $\boldsymbol{u}_\ell = \boldsymbol{A}^* \boldsymbol{r}_\ell$*.*
- *Let* $\mathsf{I}_{\ell+1/2} = \mathsf{I}_\ell \cup \mathsf{I}_{1/2}$
- *Solve for* $\boldsymbol{x}_{\ell+1/2}$ *by least squares on the support* $\mathsf{I}_{\ell+1/2}$*.*
- *Then let* $\boldsymbol{x}_{\ell+1}$ *be* $\boldsymbol{x}_{\ell+1/2}$ *pruned to its largest* $s$ *entries.*

*Implement the COSAMP algorithm, and compare its breakdown in terms of sparsity level to OMP.*

8.14 (Monotone Relation). *Show the following properties for monotone relations:*

1 *Given two monotone relations:* $\mathcal{F}_1, \mathcal{F}_2$*, their sum* $\mathcal{F}_1 + \mathcal{F}_2$ *is also monotone.*
2 *For an affine function* $\mathcal{F}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ *to be monotone if and only if* $\boldsymbol{A} + \boldsymbol{A}^* \succeq 0$*.*
3 *Prove the monotonicity of the KKT operator of equality constrained convex optimization, that is Lemma 8.21.*

8.15 (ADMM for Basis Pursuit). *One way of solving the basis pursuit problem*

$$\min \|\boldsymbol{x}\|_1 \quad \textit{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \tag{8.8.14}$$

*is to introduce an auxiliary variable* $\boldsymbol{z}$*, and solve the problem*

$$\min \|\boldsymbol{x}\|_1 \quad \textit{subject to} \quad \boldsymbol{A}\boldsymbol{z} = \boldsymbol{y},\ \boldsymbol{x} = \boldsymbol{z}. \tag{8.8.15}$$

*Derive an algorithm for this problem, by applying the* alternating directions method of multipliers *(ADMM). Implement your algorithm in a language of your choice, and investigate both its convergence speed and ability to reconstruct a target signal* $\boldsymbol{x}_o$ *using synthetic problem instances.*

8.16 (Dual of Principal Component Pursuit). *Show that the dual program to the PCP program is*

$$\max_{\boldsymbol{\Lambda}} \operatorname{trace}(\boldsymbol{Y}^* \boldsymbol{\Lambda}) \quad \textit{subject to} \quad J(\boldsymbol{\Lambda}) \leq 1,$$

*where* $\mathbf{\Lambda}$ *is the matrix of Lagrange multipliers for the equality constraint* $\boldsymbol{Y} = \boldsymbol{L} + \boldsymbol{S}$*, and* $J(\mathbf{\Lambda}) = \max(\|\mathbf{\Lambda}\|_2, \lambda^{-1}\|\mathbf{\Lambda}\|_\infty)$.

# 9 Nonconvex Optimization for High-Dimensional Problems

"*Premature optimization is the root of all evil.*"
– Donald Ervin Knuth, *The Art of Computer Programming*

*The previous chapter and this chapter are due in no small part to contributions from Dr. Chaobing Song.*

In Chapter 8, we introduced optimization techniques that efficiently solve many convex optimization problems that arise in recovering structured signals from incomplete or corrupted measurements, using *known* low-dimensional models. In contrast, as we saw in Chapter 7, problems associated with *learning* low-dimensional models from sample data are often nonconvex: either they do not have tractable convex relaxations or the nonconvex formulation is preferred due to physical or computational constraints (such as limited memory). In this chapter, we introduce optimization algorithms for nonconvex programs.

This chapter is not intended to give a complete exposition of nonconvex optimization, which has a long history and a vast literature. We will rather provide an overview of the most fundamental ideas and representative methods, with any eye towards (i) how problems leverage negative curvature to guarantee local (and sometimes global) optimality and (ii) how to characterize more precisely the computational complexity of different algorithms in order to achieve the optimal efficiency. Unlike previous chapters, some methods will be presented without detailed proofs, but with pointers to relevant references where appropriate.[1]

As mentioned in the previous chapter, one major difference between nonconvex and convex problems is that nonconvex objective functions may exhibit spurious critical points[2] other than the (desired, global) minimizers. These can include spurious local minimizers, local maximizers, and various types of saddle points, etc. Generally speaking, for nonconvex optimization we need to give up on the ambition of guaranteeing global optimality across broad classes of problems, and content ourselves to develop methods which guarantee to produce local optima in general, and global optima for specially structured problems such as those

[1] Indeed, in some situations the best known guarantees of worst case performance have lengthy and technical proofs. For a more comprehensive exposition of basic techniques for *nonlinear optimization*, one may refer to classic textbooks such as [Ber03].

[2] points $\boldsymbol{x}_\star$ whose gradient vanishes: $\nabla f(\boldsymbol{x}_\star) = \mathbf{0}$.

described in Chapter 7. The key to both of these goals is leveraging *negative curvature* of the objective function. In Chapter 7, problem symmetries induced negative curvature in symmetry-breaking directions, in some situation leading to nonconvex functions with benign global geometry. Identifying directions of negative curvature allowed us to prove that some such functions have no spurious local minimizers. Here, we will use negative curvature in a different, algorithmic way: to build methods that escape saddle points and converge to minimizers.

We will explore a variety of means to accomplish this, which require different types of local information about the objective function, from both the gradient and the full Hessian matrix (Section 9.2) or the gradient alone (Sections 9.3–9.5). At the technical level, we will reveal clearly that useful negative curvature information in the Hessian can be efficiently computed or approximated from a sequence of gradient evaluations, either explicitly (Sections 9.3 and 9.4) or implicitly with noise (Section 9.5).

Recall that in Section 8.3.1 of Chapter 8, we have discussed whether a long history of states and gradients may help improve the convergence of first-order methods. Nesterov's method has shown that in the convex case two previous states and one gradient per iteration are sufficient to achieve the optimal rate of convergence. We will see in this chapter that in the nonconvex case a longer sequence of gradient evaluations is needed to achieve another objective: escaping unstable saddle points. Roughly speaking, how efficiently and accurately one can use gradients to estimate the direction of negative curvature is the key to achieve different, and eventually optimal, tradeoffs between per-iteration cost and rate of convergence. This is reflected through the improved sophistication in algorithm design and analysis from Section 9.3, to 9.4, and to 9.5.

Last but not the least, in our context, many nonconvex problems arise due to the fact that the optimization is constrained over a nonlinear submanifold. The submanifold typically has very good geometric structures. We will discuss in Section 9.6 how to exploit such structures to develop more efficient algorithms.

## 9.1 Challenges and Opportunities

In this chapter, we focus on the problem of minimizing a function $f(\boldsymbol{x})$

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n, \tag{9.1.1}$$

which we assume to be twice continuously differentiable.[3] We know that at any local minimizer $\boldsymbol{x}_\star$, the gradient vanishes:

$$\nabla f(\boldsymbol{x}_\star) = \boldsymbol{0},$$

[3] For simplicity, we focus on smooth, unconstrained optimization problems. Generally speaking, like the convex case in Chapter 8, constrained problems can be dealt with using the Lagrange multiplier method and in our context most non-smooth objectives admit efficient proximal operators. We defer discussions of nonsmoothness and constraints to the Notes section, as well as the exercises at the end of this chapter.

i.e., $\boldsymbol{x}_\star$ is a critical point. In Section 9.1.1, we review what is arguably the simplest and most widely used optimization method: *gradient descent*. We will see that in general, gradient methods guarantee convergence to a critical point. For *convex* $f$, this suffices to solve the problem in a very strong sense: for convex $f$, every critical point is a global minimizer. In contrast, *nonconvex* $f$ can exhibit other types of critical points, including local minimizers, local maximizers, and saddle points. Hence, convergence to a critical point is not sufficient even to guarantee local optimality: to achieve this, we must somehow use information about the curvature of the objective function.[4] We therefore next review a classical approach to leveraging curvature information to rapidly minimize *convex* functions $f$, *Newton's method*, and, with these two methods as motivating background, embark on a tour of approaches to using (negative) curvature information to locally minimize *nonconvex* $f$.

### 9.1.1 Finding Critical Points via Gradient Descent

Perhaps the simplest and most widely used optimization method is *gradient descent*,[5] which generates a sequence of iterates

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_k \nabla f(\boldsymbol{x}_k) \tag{9.1.2}$$

by repeatedly stepping in the direction of the negative gradient of the function $f$. Because this method only requires one to compute the gradient of the objective function $f$ at each iteration, it is often quite scalable. The choice of $-\nabla f$ as a descent direction makes intuitive sense, since this is the direction of steepest descent of the object function $f$; indeed, $\nabla f$ is the slope of a first-order approximation to $f$ at the given point $\boldsymbol{x}_k$:

$$f(\boldsymbol{y}) \approx f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{y} - \boldsymbol{x}_k \rangle \,. \tag{9.1.3}$$

In (9.1.2), $\gamma_k > 0$ is a step size, which can be chosen adaptively from iteration to iteration, or can be set ahead of time based on knowledge of the objective function $f$. In particular, suppose that gradient $\nabla f$ is Lipschitz continuous:

$$\forall \boldsymbol{x}, \boldsymbol{y} \quad \|\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\|_2 \le L_1 \|\boldsymbol{y} - \boldsymbol{x}\|_2 \tag{9.1.4}$$

for some $L_1 > 0$.[6] In this setting, one can augment the *local* approximation

[4] Strictly speaking, many of the methods we describe guarantee convergence not to a local minimizer, but to a second-order stationary point, i.e., a point satisfying $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$ and $\nabla^2 f(\boldsymbol{x}) \succeq \boldsymbol{0}$. For "generic" (i.e., Morse) $f$, every such point is a local minimizer; this is also the case for the functions studied in Chapter 7. However, it is possible to construct objectives $f$ with second-order stationary points that are not minimizers; take, e.g., $f(x) = -x^4$.

[5] Like most natural ideas, gradient methods have a rich history, having been (re)discovered many times. The first formal exposition is believed to have been given by Augustin Cauchy in 1847, in the context of finding numerical solutions to equations [Cau47].

[6] Or equivalently, as $f$ is twice differentiable, the absolute values of eigenvalues of the Hessian $\nabla^2 f(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$ are uniformly bounded by $L_1$.

(9.1.3) to produce a *global upper bound*

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{y} - \boldsymbol{x}_k \rangle + \frac{L_1}{2} \|\boldsymbol{y} - \boldsymbol{x}_k\|_2^2. \tag{9.1.5}$$

As we showed in Chapter 8, the upper bound on the right hand side is minimized at $\boldsymbol{y}_\star = \boldsymbol{x}_k - \frac{1}{L_1} \nabla f(\boldsymbol{x}_k)$, i.e., taking on gradient step is equivalent to minimizing a quadratic upper bound on the objective function $f$.

This observation suggests choosing $\gamma_k = 1/L_1$. This step size guarantees that (i) the gradient method is a *descent* method, i.e., the objective function does not increase from iteration to iteration, and (ii) that the iterates $\boldsymbol{x}_k$ converge to a critical point $\boldsymbol{x}_\star$, satisfying $\nabla f(\boldsymbol{x}_\star) = \boldsymbol{0}$. Intuitively, we might expect to converge to a critical point $\boldsymbol{x}_\star$ that is a local minimizer. Although this is often the case in practice, in general all one can guarantee is convergence to *some* critical point, which could be a maximizer or a saddle point. Indeed, if $\boldsymbol{x}_k$ happens to be a saddle point and so $\nabla f(\boldsymbol{x}_k) = \boldsymbol{0}$, the iteration (9.1.2) will never leave $\boldsymbol{x}_k$.

In Chapter 8, we obtained useful intuition (and good methods!) by not only proving that methods converge, but assessing *how rapidly* they converge, and seeking methods whose convergence rate is the best possible. In the nonconvex setting, it does not make sense to measure the progress of gradient descent in terms of function values, because it may not converge to a global minimizer. Instead, one typically measures how close $\boldsymbol{x}_k$ is to being a critical point through the norm of the gradient $\|\nabla f(\boldsymbol{x}_k)\|_2$. In this setting, one can show:

Proposition 9.1 (Convergence Rate of Gradient Descent for Nonconvex Functions). *Suppose that $f(\boldsymbol{x})$ is a (possibly nonconvex) differentiable function with $\nabla f$ Lipschitz continuous with constant $L_1$. The gradient descent scheme (9.1.2) with the step size $\gamma_k = 1/L_1$ converges to a critical point $\boldsymbol{x}_\star$. Furthermore, the gradient norm at the best iterate $\min_{0 \leq i \leq k-1} \|\nabla f(\boldsymbol{x}_i)\|$ goes to zero at the rate $O(1/\sqrt{k})$.*

*Proof* $\forall k \geq 1$, the gradient descent iteration $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} - \frac{1}{L_1} \nabla f(\boldsymbol{x}_{k-1})$ is equivalent to:

$$\boldsymbol{x}_k := \operatorname*{argmin}_{\boldsymbol{x}} \left\{ f(\boldsymbol{x}_{k-1}) + \langle \nabla f(\boldsymbol{x}_{k-1}), \boldsymbol{x} - \boldsymbol{x}_{k-1} \rangle + \frac{L_1}{2} \|\boldsymbol{x} - \boldsymbol{x}_{k-1}\|_2^2 \right\}.$$

Also note that, according to Lemma 8.2, Lipschitz continuity (9.1.4) is equivalent to: $\forall \boldsymbol{x}, \boldsymbol{y}$

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{L_1}{2} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2. \tag{9.1.6}$$

It follows that

$$\begin{aligned} f(\boldsymbol{x}_k) &\leq f(\boldsymbol{x}_{k-1}) + \langle \nabla f(\boldsymbol{x}_{k-1}), \boldsymbol{x}_k - \boldsymbol{x}_{k-1} \rangle + \frac{L_1}{2} \|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2^2 \\ &\leq f(\boldsymbol{x}_{k-1}) - \frac{1}{2L_1} \|\nabla f(\boldsymbol{x}_{k-1})\|_2^2. \end{aligned} \tag{9.1.7}$$

Hence the value of the objective function decreases with the iteration. Telescoping (9.1.7), we obtain

$$f(\boldsymbol{x}_k) \leq f(\boldsymbol{x}_0) - \frac{1}{2L_1} \sum_{i=0}^{k-1} \|\nabla f(\boldsymbol{x}_i)\|_2^2.$$

This gives

$$\frac{k}{2L_1} \min_{i \in \{0,1,\ldots,k-1\}} \|\nabla f(\boldsymbol{x}_i)\|_2^2 \leq \sum_{i=0}^{k-1} \frac{1}{2L_1} \|\nabla f(\boldsymbol{x}_i)\|_2^2 \leq f(\boldsymbol{x}_0) - f(\boldsymbol{x}_k).$$

With respect to the critical point $\boldsymbol{x}_\star$ to which the sequence converges, we have $f(\boldsymbol{x}_0) - f(\boldsymbol{x}_k) \leq f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star)$. So we have

$$\min_{i \in \{0,1,\ldots,k-1\}} \|\nabla f(\boldsymbol{x}_i)\|_2 \leq \sqrt{\frac{2L_1(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star))}{k}}. \tag{9.1.8}$$

□

We note several key differences from the analyses of gradient and proximal gradient methods in Chapter 8. First, and most importantly, in the nonconvex setting we only guarantee convergence to a (first-order) critical point – which may not be a minimizer. Second, in contrast to the convex setting, here, the gradient method is essentially optimal amongst first-order methods. In Chapter 8, we were able to improve the behavior of (proximal) gradient descent, by comparing its rate of convergence to the best achievable rate of convergence for first-order methods, i.e., methods assuming access to only the *first-order oracle*:

$$\text{the gradient } \nabla f(\boldsymbol{x}) \text{ of the function } f(\boldsymbol{x}), \tag{9.1.9}$$

at each iteration. The convergence rate in Proposition 9.1 can be interpreted as saying that to achieve $\|\nabla f\| \leq \varepsilon_g$, we require $O(\varepsilon_g^{-2})$ iterations. This turns out to be the best (worst case) rate that first-order methods can achieve for the class of functions $f$ with Lipschitz gradients: in contrast to the convex case, introducing momentum or other forms of acceleration does not improve the worst case performance.

However, if we assume a little more about the objective function, namely that the Hessian is also Lipschitz, the picture changes dramatically. In this setting, it is possible to obtain information about the curvature of the objective function by comparing gradients at nearby points: the second derivative $\nabla^2 f(\boldsymbol{x})\boldsymbol{\delta}$ in the $\boldsymbol{\delta}$ direction satisfies $\nabla^2 f(\boldsymbol{x})\boldsymbol{\delta} \approx \nabla f(\boldsymbol{x} + \boldsymbol{\delta}) - \nabla f(\boldsymbol{x})$. We will see that by using this information, it is possible to fundamentally improve the convergence rate of gradient descent *and* to enable it to escape (nondegenerate) saddle points and maximizers. This highlights the importance of curvature information in nonconvex optimization. In the coming sections, we will first review efforts to explicitly leverage curvature information through the Hessian $\nabla^2 f(\boldsymbol{x})$, before describing lighter-weight methods that leverage curvature using gradients only.

### 9.1.2 Finding Critical Points via Newton's Method

The simplest and most natural approach to incorporating curvature information into iterative methods is to replace the first-order approximation (9.1.3) with a second-order approximation. Suppose that the Hessian $\nabla^2 f(\boldsymbol{x})$ is Lipschitz:

$$\forall \boldsymbol{x}, \boldsymbol{y} \quad \|\nabla^2 f(\boldsymbol{y}) - \nabla^2 f(\boldsymbol{x})\| \leq L_2 \|\boldsymbol{y} - \boldsymbol{x}\|_2, \tag{9.1.10}$$

where $\|\cdot\|$ denotes the spectral norm of a matrix. Then in the vicinity of a point $\boldsymbol{x}$, we can accurately approximate $f(\boldsymbol{y})$ with the Taylor expansion

$$f(\boldsymbol{y}) \approx \hat{f}(\boldsymbol{y}, \boldsymbol{x}) \doteq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \tfrac{1}{2} (\boldsymbol{y} - \boldsymbol{x})^* \nabla^2 f(\boldsymbol{x}) (\boldsymbol{y} - \boldsymbol{x}). \tag{9.1.11}$$

This approximation $\hat{f}$ has the same slope and curvature as $f$ at $\boldsymbol{y} = \boldsymbol{x}$. When $f$ is a strongly convex function, the eigenvalues of $\nabla^2 f$ are all positive and the approximation is also strongly convex. In this setting, the approximation $\hat{f}$ has a unique minimizer

$$\boldsymbol{y}_\star = \arg\min_{\boldsymbol{y}} \hat{f}(\boldsymbol{y}, \boldsymbol{x}) = \boldsymbol{x} - \left[\nabla^2 f(\boldsymbol{x})\right]^{-1} \nabla f(\boldsymbol{x}). \tag{9.1.12}$$

The expression on the right hand side can be obtained simply by setting the derivative $\nabla_{\boldsymbol{y}} \hat{f}(\boldsymbol{y}, \boldsymbol{x}) = \boldsymbol{0}$ and solving for $\boldsymbol{y}$. This suggests the following iterative approach to minimizing $f$: starting from an initial point $\boldsymbol{x}_0$, we generate a sequence of iterates $\boldsymbol{x}_k$ by setting

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\nabla^2 f(\boldsymbol{x}_k)\right]^{-1} \nabla f(\boldsymbol{x}_k). \tag{9.1.13}$$

This is known as the *Newton iteration*, and is closely related to the Newton-Raphson method for finding roots of polynomials. Indeed, searching for a critical point of a smooth function $f$ is equivalent to looking for a solution (root) of the equation $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$.[7] Newton's method clearly belongs to the class of methods which assume access to the *the second-order oracle*:

$$\text{the gradient } \nabla f(\boldsymbol{x}) \text{ and the Hessian } \nabla^2 f(\boldsymbol{x}), \tag{9.1.14}$$

Typically, this makes the iterations of Newton's method much more expensive than those of gradient descent: generally, one needs to compute and store the full $n \times n$ Hessian matrix $\nabla^2 f(\boldsymbol{x}_k)$ and its inverse. The benefit of this per-iteration complexity is a drastic reduction in the number of iterations required to converge to an accurate solution. Consider, for example, a strongly convex objective function $f$, (i.e., an $f$ which satisfies $\nabla^2 f(\boldsymbol{x}) \succeq \lambda \boldsymbol{I}$ for all $\boldsymbol{x}$), with Lipschitz Hessian. Because $f$ is strongly convex, it has a unique minimizer $\boldsymbol{x}_\star$.

[7] Like gradient descent, Newton's method (or the Newton-Raphson method) has a long history. It can be interpreted as a method for solving the critical point equation $\nabla f(\boldsymbol{x}_\star) = \boldsymbol{0}$, i.e., finding "roots" of $\nabla f(\boldsymbol{x})$. The Newton-Raphson method was originally introduced in the late 1680's by Isaac Newton and Joseph Raphson for finding roots of polynomials, and generalized to find critical points of smooth functions by Thomas Simpson in 1750 [Sim50].

We will show that the iterates produced by Newton's method satisfy

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_\star\|_2 \leq \frac{L_2}{2\lambda}\|\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2^2. \tag{9.1.15}$$

This means that as long as $\boldsymbol{x}_0$ is close to $\boldsymbol{x}_\star$ (say, $\|\boldsymbol{x}_0 - \boldsymbol{x}_\star\| < \frac{2\lambda}{L_2}$), the iterates $\boldsymbol{x}_k$ converge to $\boldsymbol{x}_\star$ extraordinarily rapidly:

$$\|\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2 \leq \left(\frac{L_2}{2\lambda}\right)^{2^{k+1}} \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2^{2^k}. \tag{9.1.16}$$

In optimization, this is referred to as *superlinear* convergence: $\log \|\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2$ diminishes faster than any linear function of $k$. Slightly more formally:

PROPOSITION 9.2 (Convergence Rate of Newton's Method). *Let $f(\boldsymbol{x})$ be strongly convex, with $\lambda_{\min}(\nabla^2 f(\boldsymbol{x})) \geq \lambda > 0$ for all $\boldsymbol{x}$, and assume that $\nabla^2 f$ is Lipschitz continuous with constant $L_2$, and let $\boldsymbol{x}_\star$ be the (unique) minimizer of $f$ over $\mathbb{R}^n$. Assuming $\|\boldsymbol{x}_0 - \boldsymbol{x}_\star\| < \frac{2\lambda}{L_2}$, the iterates $\boldsymbol{x}_k$ converge to $\boldsymbol{x}_\star$, with quadratic rate (9.1.16).*

*Proof* Using the Taylor expansion of the gradient $\nabla f(\boldsymbol{x})$ around the critical point $\boldsymbol{x}_\star$ and the mean value theorem, we have

$$\|\nabla f(\boldsymbol{x}_\star) - [\nabla f(\boldsymbol{x}_k) + \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x}_\star - \boldsymbol{x}_k)]\|_2 \leq \frac{L_2}{2}\|\boldsymbol{x}_\star - \boldsymbol{x}_k\|_2^2.$$

With $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\nabla^2 f(\boldsymbol{x}_k)\right]^{-1} \nabla f(\boldsymbol{x}_k)$, this gives

$$\|\nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x}_\star - \boldsymbol{x}_{k+1})\|_2 \leq \frac{L_2}{2}\|\boldsymbol{x}_\star - \boldsymbol{x}_k\|_2^2.$$

The operator norm of the Hessian inverse $\left[\nabla^2 f(\boldsymbol{x})\right]^{-1}$ is bounded uniformly from above, by $\lambda^{-1} < \infty$. Combining this with the above inequality, we obtain:

$$\|\boldsymbol{x}_\star - \boldsymbol{x}_{k+1}\|_2 \leq \frac{L_2}{2\lambda}\|\boldsymbol{x}_\star - \boldsymbol{x}_k\|_2^2.$$

□

Despite its fast rate of convergence for strongly convex problems, there are several limitations that render Newton's method inapplicable in our setting of high-dimensional, nonconvex optimization. First, Newton's method requires us to compute $\left[\nabla^2 f(\boldsymbol{x})\right]^{-1} \nabla f(\boldsymbol{x})$. Simply storing the $n \times n$ Hessian matrix is infeasible when $n$ is large. Typical approaches to solving the Newton system require $O(n^3)$ arithmetic operations, making even a single step of Newton's method computationally prohibitive when $n$ is large. These limitations are why, in Chapter 8, we focused on convex optimization methods with cheaper iterations.

Second, and more fundamentally, in the nonconvex setting, we have no control over what kind of critical point the iterates $\boldsymbol{x}_k$ converge to! Close inspection shows that the argument of Proposition 9.2 works just as well to prove convergence to a maximizer, with essentially the same quadratic rate. There is a simple

reason why Newton's method cannot distinguish between minimizers, maximizers and saddles. In the nonconvex setting, solving $\nabla_{\boldsymbol{y}} \hat{f}(\boldsymbol{y}, \boldsymbol{x}) = \mathbf{0}$ does not necessarily yield a minimizer of the quadratic approximation $\hat{f}$ – rather, it asks for a *critical point* of this approximation, which, depending on the signs of the eigenvalues of $\nabla^2 f(\boldsymbol{x})$, could be a minimizer, maximizer or saddle point. Hence, in the nonconvex setting, the classical Newton's method can be interpreted not as repeatedly *minimizing* approximations to the objective, but as repeatedly finding critical points of approximations to the objective. Exercise 9.1 guides the reader through examples showing that Newton's method may converge to minimizers, maximizers, and saddle points.

Clearly, if our goal is to leverage negative curvature to *minimize* nonconvex functions, some modifications to Newton's method are required. In the subsequent sections, we will show how to modify Newton's method to escape saddle points and obtain minimizers (strictly speaking, to obtain second-order critical points satisfying $\nabla f(\boldsymbol{x}_\star) = \mathbf{0}$ and $\nabla^2 f(\boldsymbol{x}_\star) \succeq \mathbf{0}$). We will then show how to reduce the per-iteration complexity by leveraging negative curvature without the full Hessian, or even using only gradient information, yielding methods that are applicable to high-dimensional problems. Finally, similar to our development of proximal and accelerated proximal gradient methods in Chapter 8, we will show how to carefully combine gradient and curvature information to obtain first-order methods that achieve the best known rate of convergence.

## 9.2 Cubic Regularization of Newton's Method

As we saw in the previous section, when applied to strongly convex problems, Newton's method converges extremely rapidly. In comparison to the gradient method, it better leverages positive curvature of the objective function. However, when applied to *nonconvex* problems, it does not distinguish between minimizers, maximizers and saddle points. In particular, it is incapable of leveraging *negative curvature* to escape saddle points. To develop second-order methods that make better use of negative curvature, a natural idea is to build a local model of the objective function which contains both first and second-order information, i.e., to write

$$f(\boldsymbol{y}) \approx f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \tfrac{1}{2} (\boldsymbol{y} - \boldsymbol{x})^* \nabla^2 f(\boldsymbol{x}) (\boldsymbol{y} - \boldsymbol{x}). \tag{9.2.1}$$

and to determine a step direction by *minimizing* this model. This is in contrast to Newton's method, which only seeks a critical point of this approximation.

Here, a comparison to our development of gradient methods in Chapter 8 is instructive. There, we motivated gradient and proximal gradient methods from a first-order approximation to the objective function,

$$f(\boldsymbol{y}) \approx f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle . \tag{9.2.2}$$

In contrast, the second-order approximation in (9.2.1) retains information about

the curvature of $f$, through the Hessian $\nabla^2 f$. In particular, $f$ has directions of negative curvature at $\boldsymbol{x}$ if and only if the smallest eigenvalue $\lambda_{\min}(\nabla^2 f)$ is negative. In particular, any eigenvector corresponding to this smallest eigenvalue gives a direction of negative curvature.

### 9.2.1 Convergence to Second-order Stationary Points

How can we use the model (9.2.1) to reduce the objective function $f$? In our study of gradient and proximal gradient methods, we found it useful to augment the *local* approximation in (9.2.2) to produce a *global* upper bound on $f(\boldsymbol{y})$. Minimizing this global upper bound produced a new point $\boldsymbol{x}^+$ with $f(\boldsymbol{x}^+) \leq f(\boldsymbol{x})$, i.e., it guarantees descent in the objective value $f$. Here, we proceed in the same spirit. Suppose that the Hessian $\nabla^2 f$ is Lipschitz, i.e.,

$$\forall \boldsymbol{x}, \boldsymbol{y} \quad \|\nabla^2 f(\boldsymbol{y}) - \nabla^2 f(\boldsymbol{x})\| \leq L_2 \|\boldsymbol{y} - \boldsymbol{x}\|_2, \tag{9.2.3}$$

for some $L_2 > 0$. Here $\|\cdot\|$ denotes the spectral norm of matrix. In this setting, we have that for all $\boldsymbol{x}, \boldsymbol{y}$,

$$\begin{aligned} f(\boldsymbol{y}) &\leq \hat{f}(\boldsymbol{y}, \boldsymbol{x}) \\ &\doteq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \tfrac{1}{2}(\boldsymbol{y} - \boldsymbol{x})^* \nabla^2 f(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x}) + \tfrac{L_2}{6} \|\boldsymbol{y} - \boldsymbol{x}\|_2^3 . \end{aligned} \tag{9.2.4}$$

The right hand side is a global upper bound on $f(\boldsymbol{y})$, which has the same value, slope and curvature at $\boldsymbol{x}$. Similar to our discussion of gradient descent in Chapter 8 and Section 9.1.1, given an iterate $\boldsymbol{x}_k$, we can produce the next iterate $\boldsymbol{x}_{k+1}$ by minimizing this upper bound:

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{y}} \hat{f}(\boldsymbol{y}, \boldsymbol{x}_k). \tag{9.2.5}$$

The resulting method is known as the *Cubic Regularized Newton's Method*, and is described in Figure 9.1.

In contrast to gradient descent, the problem of minimizing the approximation $\hat{f}$ in (9.2.5) is itself in general a nonconvex problem – we intentionally choose an approximation that retains negative curvature information! Perhaps surprisingly, this particular nonconvex problem *can* be solved efficiently: it can be reduced to solving a one-dimensional convex optimization problem [NP06]. We guide the reader through a derivation of this subproblem in Exercise 9.3, and describe more scalable alternatives in the next section, after discussing convergence issues.

Remark 9.3 (The Trust Region Method). *The cubic regularized Newton's method is not the only way of using the second-order approximation* (9.2.1) *to solve nonconvex optimization problems. One important (and historically earlier) alternative is the* trust region method. *Rather than building a global upper bound to* $f(\boldsymbol{y})$, *the trust region method chooses a step direction by minimizing the approximation* (9.2.1) *over a small neighborhood of the point* $\boldsymbol{x}$, *where the approximation*

*is known to be accurate:*

$$\boldsymbol{x}_{k+1} = \arg\min_{\|\boldsymbol{y}-\boldsymbol{x}_k\|_2 \le \delta_k} f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{y} - \boldsymbol{x}_k \rangle + \tfrac{1}{2}(\boldsymbol{y} - \boldsymbol{x}_k)^* \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{y} - \boldsymbol{x}_k). \tag{9.2.6}$$

*Like the cubic Newton subproblem, this subproblem can be solved efficiently; like cubic Newton, the resulting method is able to leverage negative curvature, as captured by the Hessian $\nabla^2 f$. The main difference is simply the use of the constraint $\|\boldsymbol{y} - \boldsymbol{x}_k\|_2 \le \delta_k$ instead of the cubic penalty $\|\boldsymbol{y} - \boldsymbol{x}_k\|_2^3$. We guide the interested reader through the development of the trust region method and the solution of the trust region subproblem in Exercise 9.2.*

We next show that the iterates produced by the Cubic Regularized Newton's Method converge to point $\boldsymbol{x}_\star$ that satisfies

$$\nabla f(\boldsymbol{x}_\star) = \boldsymbol{0}; \quad \nabla^2 f(\boldsymbol{x}_\star) \succeq \boldsymbol{0}, \tag{9.2.7}$$

i.e., a second-order stationary solution. We will measure our progress in terms of the following quantity

$$\mu(\boldsymbol{x}) \doteq \max\left\{ \sqrt{\frac{1}{L_2}\|\nabla f(\boldsymbol{x})\|_2},\ -\frac{2}{3L_2}\lambda_{\min}\big(\nabla^2 f(\boldsymbol{x})\big) \right\}, \tag{9.2.8}$$

where $\lambda_{\min}$ is the smallest eigenvalue of the Hessian $\nabla^2 f(\boldsymbol{x})$, which we desire to be nonnegative. If $\mu(\boldsymbol{x}) \to 0$, $\boldsymbol{x}_k$ converges to a solution $\boldsymbol{x}_\star$ that satisfies (9.2.7). The following theorem shows that this indeed occurs, and controls the rate at which $\mu(\boldsymbol{x}_k)$ approaches zero:

THEOREM 9.4 (Convergence Rate of Cubic Newton's Method). *Suppose $f(\boldsymbol{x})$ is bounded from below. Then the sequence $\{\boldsymbol{x}_k\}$ generated by the cubic regularized Newton step (9.2.5) converges to a non-empty set of limit points $\boldsymbol{X}_\star$. Let $\boldsymbol{x}_\star \in \boldsymbol{X}_\star$. Then we further have $\lim_{k\to\infty} \mu(\boldsymbol{x}_k) = 0$ and for any $k \ge 1$, we have*

$$\min_{1 \le i \le k} \mu(\boldsymbol{x}_i) \le C \left( \frac{f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star)}{k \cdot L_2} \right)^{1/3} \tag{9.2.9}$$

*for some constant $C > 0$.*

*Sketch of Proof.* Since $\boldsymbol{x}_k$ is the minimizer of $\hat{f}(\boldsymbol{y}, \boldsymbol{x}_{k-1})$ as defined in (9.2.4), it satisfies the first-order optimality condition:

$$\nabla f(\boldsymbol{x}_{k-1}) + \nabla^2 f(\boldsymbol{x}_{k-1})(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) + \frac{L_2}{2}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) = \boldsymbol{0}. \tag{9.2.10}$$

In addition, from the derivation of the global minimizer for (9.2.5), one can also show that $\boldsymbol{x}_k$ satisfies the condition (see Proposition 1 of [NP06]):

$$\nabla^2 f(\boldsymbol{x}_{k-1}) + \frac{L_2}{2}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2 \boldsymbol{I} \succeq \boldsymbol{0}. \tag{9.2.11}$$

**Cubic Regularized Newton's Method**

**Problem Class:**

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n,$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is nonconvex, and is twice continuously differentiable, with both gradient and Hessian Lipschitz continuous. We have access to the second-order oracle: $\nabla f(\boldsymbol{x}) \in \mathbb{R}^n$ and $\nabla^2 f(\boldsymbol{x}) \in \mathbb{R}^{n\times n}$.

**Setup:** Let $\hat{f}(\boldsymbol{y}, \boldsymbol{x})$ be defined similarly as in (9.2.4):

$$\hat{f}(\boldsymbol{y}, \boldsymbol{x}) \doteq \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle + \frac{1}{2}(\boldsymbol{y} - \boldsymbol{x})^* \nabla^2 f(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x}) + \frac{L_2}{6}\|\boldsymbol{y} - \boldsymbol{x}\|_2^3.$$

**Initialization:** Set $\boldsymbol{x}_0 \in \mathbb{R}^n$,

**Iteration:** For $k = 0, 1, 2, \ldots$

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{y}} \hat{f}(\boldsymbol{y}, \boldsymbol{x}_k).$$

**Convergence Guarantee:** $\boldsymbol{x}_k$ converges with $\lim_{k\to\infty} \mu(\boldsymbol{x}_k) = 0$.

**Figure 9.1** An overview of the Cubic Regularization of Newton's Method.

Since $\hat{f}(\boldsymbol{y}, \boldsymbol{x}_k)$ defined in (9.2.4) is an upper bound of $f(\boldsymbol{y})$, at iterate $\boldsymbol{x}_k$ we have:

$$\begin{aligned}
f(\boldsymbol{x}_k) &\le f(\boldsymbol{x}_{k-1}) + \langle \nabla f(\boldsymbol{x}_{k-1}), \boldsymbol{x}_k - \boldsymbol{x}_{k-1}\rangle \\
&\quad + \frac{1}{2}\langle \nabla^2 f(\boldsymbol{x}_{k-1})(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}), \boldsymbol{x}_k - \boldsymbol{x}_{k-1}\rangle \\
&\quad + \frac{L_2}{6}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2^3 \\
&= f(\boldsymbol{x}_{k-1}) \\
&\quad + \langle \nabla f(\boldsymbol{x}_{k-1}), \boldsymbol{x}_k - \boldsymbol{x}_{k-1}\rangle + \langle \nabla^2 f(\boldsymbol{x}_{k-1})(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}), \boldsymbol{x}_k - \boldsymbol{x}_{k-1}\rangle \\
&\quad - \frac{1}{2}\langle \nabla^2 f(\boldsymbol{x}_{k-1})(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}), \boldsymbol{x}_k - \boldsymbol{x}_{k-1}\rangle + \frac{L_2}{6}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2^3 \\
&= f(\boldsymbol{x}_{k-1}) - \frac{L_2}{2}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2^3 \\
&\quad - \frac{1}{2}\langle \nabla^2 f(\boldsymbol{x}_{k-1})(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}), \boldsymbol{x}_k - \boldsymbol{x}_{k-1}\rangle + \frac{L_2}{6}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2^3 \\
&\le f(\boldsymbol{x}_{k-1}) - \frac{L_2}{12}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2^3, \qquad (9.2.12)
\end{aligned}$$

where from the second equality to the third is by the first-order optimality condition (9.2.10), the last inequality is by applying the second optimality condition (9.2.11). The last inequality (9.2.12) indicates that the cubic regularized Newton's method is indeed a descent method.

Telescoping (9.2.12), we have

$$f(\boldsymbol{x}_k) \leq f(\boldsymbol{x}_0) - \frac{L_2}{12}\sum_{i=1}^{k}\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|_2^3.$$

So we have

$$\frac{L_2 k}{12}\min_{i\in[k]}\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|_2^3 \leq \frac{L_2}{12}\sum_{i=1}^{k}\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|_2^3 \leq f(\boldsymbol{x}_0) - f(\boldsymbol{x}_k) \leq f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star),$$

where $\boldsymbol{x}_\star$ is a minimizer to which the sequence converges. So we have

$$\min_{i\in[k]}\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|_2 \leq \left(\frac{12(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star))}{L_2 k}\right)^{\frac{1}{3}}.$$

Now to ensure the convergence rate, we need to determine the relationship between $\nabla f(\boldsymbol{x}_k)$, $\nabla^2 f(\boldsymbol{x}_k)$, and $\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2$.

First, note that through Taylor expansion and the mean value theorem, the Lipschitz Hessian condition implies that for all $\boldsymbol{x}, \boldsymbol{y}$,

$$\|\nabla f(\boldsymbol{y}) - (\nabla f(\boldsymbol{x}) + \nabla^2 f(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x}))\|_2 \leq \frac{L_2}{2}\|\boldsymbol{y} - \boldsymbol{x}\|_2^2. \tag{9.2.13}$$

Combining with (9.2.10), we have

$$\begin{aligned}
\|\nabla f(\boldsymbol{x}_k)\|_2 &= \|\nabla f(\boldsymbol{x}_k) - (\nabla f(\boldsymbol{x}_{k-1}) + \nabla^2 f(\boldsymbol{x}_{k-1})(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})) \\
&\quad + (\nabla f(\boldsymbol{x}_{k-1}) + \nabla^2 f(\boldsymbol{x}_{k-1})(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}))\|_2 \\
&\leq \|\nabla f(\boldsymbol{x}_k) - (\nabla f(\boldsymbol{x}_{k-1}) + \nabla^2 f(\boldsymbol{x}_{k-1})(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}))\|_2 \\
&\quad + \|\nabla f(\boldsymbol{x}_{k-1}) + \nabla^2 f(\boldsymbol{x}_{k-1})(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})\|_2 \\
&\leq L_2\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2^2.
\end{aligned} \tag{9.2.14}$$

Second, by (9.1.10) and (9.2.11), we have

$$\nabla^2 f(\boldsymbol{x}_k) \succeq \nabla^2 f(\boldsymbol{x}_{k-1}) - L_2\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2 \boldsymbol{I} \succeq -\frac{3L_2}{2}\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\|_2 \boldsymbol{I}.$$

Therefore we have[8]

$$\|\nabla f(\boldsymbol{x}_k)\|_2 \leq L_2\left(\frac{12(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star))}{L_2 k}\right)^{\frac{2}{3}}, \tag{9.2.15}$$

$$-\lambda_{\min}(\nabla^2 f(\boldsymbol{x}_k)) \leq \frac{3L_2}{2}\left(\frac{12(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star))}{L_2 k}\right)^{\frac{1}{3}}. \tag{9.2.16}$$

By definition of $\mu(\boldsymbol{x})$, we have $\mu(\boldsymbol{x}_k) \leq \left(\frac{12(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star))}{L_2 k}\right)^{\frac{1}{3}}$ converges as $k \to \infty$. □

[8] Strictly speaking, we here should consider the iterate that achieves $\min_{i\in[k]}\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\|_2$. We here use the last iterate $\boldsymbol{x}_k$ for simplicity.

The fact that $\mu(\boldsymbol{x}_k) \to 0$ implies that the cubic regularized Newton iteration (9.2.5) indeed converges asymptotically to stationary limit points with $\nabla f(\boldsymbol{x}_\star) = \mathbf{0}$ and $\nabla^2 f(\boldsymbol{x}_\star) \succeq \mathbf{0}$. Furthermore, the bound (9.2.9) on $\mu$ implies that with a finite number of $k$ iterations

$$\min_{1 \le i \le k} \|\nabla f(\boldsymbol{x}_i)\|_2 \le O(k^{-2/3}),$$

which, as expected, is improved over the bound $O(k^{-1/2})$ for first-order (accelerated) gradient descent (see Proposition 9.1), and is tight for methods with access to the second-order oracle (9.1.14).

### 9.2.2 More Scalable Solution to the Subproblem

The subproblem (9.2.5) in the cubic regularized Newton's method essentially aims to minimize the following function:

$$\min_{\boldsymbol{w}} \psi(\boldsymbol{w}) \doteq \langle \nabla f(\boldsymbol{x}), \boldsymbol{w} \rangle + \frac{1}{2} \boldsymbol{w}^* \nabla^2 f(\boldsymbol{x}) \boldsymbol{w} + \frac{L_2}{6} \|\boldsymbol{w}\|_2^3 . \tag{9.2.17}$$

Although this subproblem can be reduced to a one-dimensional convex program [NP06], that solution assumes knowing the Hessian inverse or its factorization, which can be costly when the dimension $n$ is very large.

To obtain a more scalable implementation, one may choose to minimize the nonconvex function $\psi(\boldsymbol{w})$ using gradient descent type methods. Notice that the gradient is of the form:

$$\nabla \psi(\boldsymbol{w}) = \nabla f(\boldsymbol{x}) + \nabla^2 f(\boldsymbol{x}) \boldsymbol{w} + \nabla \frac{L_2}{6} \|\boldsymbol{w}\|_2^3 , \tag{9.2.18}$$

and only the second term $\nabla^2 f(\boldsymbol{x}) \boldsymbol{w}$ involves the Hessian. Nevertheless, it is required only in the form of a "Hessian-vector product[9]" between the Hessian $\nabla^2 f(\boldsymbol{x})$ and the vector $\boldsymbol{w}$. One can approximate such a Hessian-vector product by

$$\nabla^2 f(\boldsymbol{x}) \boldsymbol{w} \approx \frac{\nabla f(\boldsymbol{x} + t\boldsymbol{w}) - \nabla f(\boldsymbol{x})}{t} \tag{9.2.19}$$

for a small $t > 0$. So we only need one additional evaluation of the gradient $\nabla f(\boldsymbol{x} + t\boldsymbol{w})$ to obtain the gradient of $\nabla \psi(\boldsymbol{w})$.

It has been shown that gradient descent (with noise[10]) can efficiently find the global minimizer of $\psi(\boldsymbol{w})$ within $\varepsilon$-accuracy[11] in $O(\varepsilon^{-1} \log(1/\varepsilon))$ steps in the worst case. Moreover, when $\varepsilon$ is small enough, the algorithm converges to an $\varepsilon$-accuracy solution with a linear rate in $O(\log(1/\varepsilon))$ steps [CD16, CD19].

[9] The role such a Hession-vector product will become clear when we study how to efficiently compute the direction of negative curvature for $f(\boldsymbol{x})$ for descending purpose in Section 9.3.2 as well as in Section 9.5.3.

[10] Noise is needed to help escape spurious critical points in some hard cases. We will reveal the role of noise clearly in Section 9.5.

[11] Here, if $\boldsymbol{w}_o$ is the global minimizer of $\psi(\boldsymbol{w})$, an $\varepsilon$-accurate solution $\boldsymbol{w}_\star$ is such that $\psi(\boldsymbol{w}_\star) \le \psi(\boldsymbol{w}_o) + \varepsilon$.

## 9.3 Gradient and Negative Curvature Descent

As we have alluded to in the preceding section, to escape from unstable critical points, it is not necessary to compute the full Hessian matrix $\nabla^2 f(\boldsymbol{x})$ at each iteration, or have to find the precise minimizer of the proxy function in the cubic Newton's method. It often suffices if we can find just a direction that sufficiently reduces the objective function. This could help alleviate the computational burden of second-order methods associated with computing the full Hessian and its inverse.[12] Hence in this section, we study methods that assume access to *the negative curvature oracle*:

$$\text{the gradient } \nabla f(\boldsymbol{x}) \text{ and a negative eigenvector } \boldsymbol{e} \text{ of } \nabla^2 f(\boldsymbol{x}). \tag{9.3.1}$$

For many practical problems, it is cheaper to obtain such a direction $\boldsymbol{e}$ of negative curvature than to compute the full Hessian. In some problems, the complexity of obtaining $\boldsymbol{e}$ can even be on par with evaluating the gradient $\nabla f(\boldsymbol{x})$.[13] Even if the negative curvature direction $\boldsymbol{e}$ must be computed numerically, one can resort to efficient methods that we will soon introduce in Section 9.3.2. For now, we assume we have this information at each iterate.

### 9.3.1 Hybrid Gradient and Negative Curvature Descent

To be consistent with the gradient descent and Newton's method, we here assume that both gradient and Hessian are Lipschitz continuous:

$$\|\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\|_2 \le L_1 \|\boldsymbol{y} - \boldsymbol{x}\|_2, \quad \|\nabla^2 f(\boldsymbol{y}) - \nabla^2 f(\boldsymbol{x})\| \le L_2 \|\boldsymbol{y} - \boldsymbol{x}\|_2.$$

One should notice one common idea in the design of all above optimization algorithms: given a prescribed precision $\varepsilon$, the function value is expected to decrease by $\varepsilon$ per iteration:

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k-1}) \le -\varepsilon,$$

unless the first-order and second-order derivatives have met the conditions of convergence.

From Proposition 9.1, we know when we conduct gradient descent, we should expect the norm of gradient $\nabla f(\boldsymbol{x}_k)$ to descend according to (9.1.8):

$$\|\nabla f(\boldsymbol{x}_k)\|_2 \;\le\; O\left(\frac{L_1(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star))}{k}\right)^{\frac{1}{2}} \;=\; O\big((L_1\varepsilon)^{1/2}\big). \tag{9.3.2}$$

According to Theorem 9.4, if we use second-order descent method, the smallest eigenvalue of Hessian $\nabla^2 f(\boldsymbol{x}_k)$ should decay with the number of iteration $k$ as (9.2.16):

$$-\lambda_{\min}\big(\nabla^2 f(\boldsymbol{x}_k)\big) \;\le\; O\left(\frac{L_2^2(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star))}{k}\right)^{\frac{1}{3}} \;=\; O\big((L_2^2\varepsilon)^{1/3}\big). \tag{9.3.3}$$

[12] which becomes prohibitive when the dimension of the problem is extremely high.
[13] say problems in which we may have analytic expressions for evaluating $\boldsymbol{e}$.

**Hybrid Gradient and Negative Curvature Descent**

**Problem Class:**

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n,$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable, with Lipschitz continuous gradient and Hessian. Have access to the oracle: the gradient $\nabla f(\boldsymbol{x})$ and the smallest eigenvalue-vector pair $(\lambda_{\min}, \boldsymbol{e})$ of the Hessian $\nabla^2 f(\boldsymbol{x})$.

**Setup:** prescribed accuracy $\varepsilon > 0$, $\varepsilon_g = (2L_1\varepsilon)^{1/2}$, and $\varepsilon_H = \left(1.5L_2^2\varepsilon\right)^{1/3}$.

**Initialization:** Set $\boldsymbol{x}_0 \in \mathbb{R}^n$,

**For** $k = 0, 1, 2, \ldots$

1 Compute the gradient $\nabla f(\boldsymbol{x}_k)$.
2 **if** $\|\nabla f(\boldsymbol{x}_k)\|_2 \geq \varepsilon_g$, **then** conduct gradient descent:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{1}{L_1}\nabla f(\boldsymbol{x}_k); \tag{9.3.4}$$

3 **else** compute the smallest eigenvalue $\lambda_k$ and eigenvector $\boldsymbol{e}_k$ of $\nabla^2 f(\boldsymbol{x}_k)$, and choose its direction such that $\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{e}_k \rangle \leq 0$.
4 **if** $-\lambda_k \geq \varepsilon_H$, **then** conduct negative curvature descent:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \frac{2\lambda_k}{L_2}\boldsymbol{e}_k; \tag{9.3.5}$$

5 **else end for** and **return** $\boldsymbol{x}_\star = \boldsymbol{x}_k$.

**Convergence guarantee:** $\|\nabla f(\boldsymbol{x}_\star)\|_2 \leq \varepsilon_g$, $-\lambda_{\min}\left(\nabla^2 f(\boldsymbol{x}_\star)\right) \leq \varepsilon_H$.

**Figure 9.2** An overview of the Hybrid Gradient and Negative Curvature Descent.

These conditions naturally suggest a simple descent strategy that alternates between gradient descent and negative curvature descent:

- When the gradient has not reached the desired precision according to (9.3.2), we keep conducting gradient descent;
- Whenever condition (9.3.2) is reached, we conduct the negative curvature search if the smallest eigenvalue of the Hessian has not reached the desired bound (9.3.3).

We summarize this hybrid descent scheme as an algorithm in Figure 9.2. Note that with this scheme, one does not have to compute the negative curvature direction unless it is needed. Then the following theorem states that the algorithm converges to the prescribed precision with the constants specified in the algorithm.

THEOREM 9.5 (Convergence of Hybrid Gradient and Negative Curvature Descent). *The gradient and negative curvature descent algorithm in Figure 9.2 con-*

*verges to a second-order stationary point* $\boldsymbol{x}_\star$ *with the desired precision* $\varepsilon$ *in no more than* $k = (f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star))/\varepsilon$ *iterations.*

*Proof* If $\|\nabla f(\boldsymbol{x}_k)\|_2 \geq \varepsilon_g = (2L_1\varepsilon)^{1/2}$, the algorithm conducts gradient descent: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{1}{L_1}\nabla f(\boldsymbol{x}_k)$. Then following the same arguments in Proposition 9.1, in particular equation (9.1.7), we have

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \frac{L_1}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^2 \\ &\leq f(\boldsymbol{x}_k) - \frac{1}{2L_1}\|\nabla f(\boldsymbol{x}_k)\|_2^2 \\ &\leq f(\boldsymbol{x}_k) - \varepsilon. \end{aligned} \tag{9.3.6}$$

Otherwise, if $-\lambda_k \geq \varepsilon_H = \left(\frac{3L_2^2\varepsilon}{2}\right)^{1/3}$, then algorithm conducts negative curvature descent: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \frac{2\lambda_k}{L_2}\boldsymbol{e}_k$. Since $\boldsymbol{e}_k$ is the eigenvector, we have $\nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) = \lambda_k(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$. Therefore, we have

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle \\ &\quad + \frac{1}{2}\langle \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \frac{L_2}{6}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^3 \\ &\leq f(\boldsymbol{x}_k) + \frac{1}{2}\langle \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \frac{L_2}{6}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^3 \\ &\leq f(\boldsymbol{x}_k) + \frac{1}{2}\lambda_k\left(\frac{2\lambda_k}{L_2}\right)^2 + \frac{L_2}{6}\left(\frac{2|\lambda_k|}{L_2}\right)^3 = f(\boldsymbol{x}_k) - \frac{2|\lambda_k|^3}{3L_2^2} \\ &\leq f(\boldsymbol{x}_k) - \frac{2\varepsilon_H^3}{3L_2^2} \qquad (9.3.7) \\ &\leq f(\boldsymbol{x}_k) - \varepsilon. \qquad (9.3.8) \end{aligned}$$

So in each iteration, the function value will decrease by $\varepsilon$. To attain $\varepsilon$, the number of gradient descent and negative curvature descent will be bounded by

$$\frac{f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star)}{\varepsilon}. \tag{9.3.9}$$

That is to say, we need at most $k \leq \frac{f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star)}{\varepsilon}$ iterations to attain

$$\|\nabla f(\boldsymbol{x}_\star)\|_2 \leq \varepsilon_g, \quad \nabla^2 f(\boldsymbol{x}_\star) \succeq -\varepsilon_H \boldsymbol{I}. \tag{9.3.10}$$

□

Remark 9.6 (Curvilinear Search). *The idea of mixing gradient descent with negative curvature descent can be traced back to the* curvilinear search method *[Gol80]. At each iterate* $\boldsymbol{x}_k$*, the curvilinear search suggests searching the next iterate along a curve:*

$$\boldsymbol{x}(\alpha) = \boldsymbol{x}_k + \alpha \boldsymbol{s}_k + \alpha^2 \boldsymbol{d}_k, \quad \alpha \in (0, 1), \tag{9.3.11}$$

*where* $\boldsymbol{s}_k$ *is typically the negative gradient, say* $-\nabla f(\boldsymbol{x})$*, and* $\boldsymbol{d}_k$ *is a direction of negative curvature, say the negative eigenvector* $\boldsymbol{e}$*. The motivation behind such a scheme is rather intuitive: when the gradient is large, we only need to take a*

*small step (i.e., $\alpha$ is small) along the negative gradient for an adequate descent; when the gradient is small, it is safe to follow more towards a direction of negative curvature and we need to take a larger step (i.e., $\alpha$ is large) to ensure an adequate descent. One can show under certain conditions, such a scheme asymptotically converges to a stable critical point. However, the precise convergence rate is not so easy to characterize.*

### 9.3.2 Computing Negative Curvature via Lanczos Method

In the above scheme, we need the direction of (the most) negative curvature $\boldsymbol{e}$, which is associated with the smallest eigenvalue of the Hessian. To characterize the precise computational complexity of the scheme, we here show that such a direction can be efficiently computed by evaluating gradients only, using the Hessian-vector product type operations. The mechanism involved is also known as the *power iteration* or a more advanced variation the *Lanczos method*.

Around the neighborhood of a given point $\boldsymbol{x}$, consider the second-order approximation to the function $f(\boldsymbol{x}+\boldsymbol{w})$:

$$\phi(\boldsymbol{w}) \doteq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{w} \rangle + \frac{1}{2}\boldsymbol{w}^* \nabla^2 f(\boldsymbol{x}) \boldsymbol{w}. \tag{9.3.12}$$

In general, the negative gradient $-\nabla f(\boldsymbol{x})$ indicates the steepest descent direction. However, if $\boldsymbol{x}$ is near a critical point, we have $\nabla f(\boldsymbol{x}) \approx \boldsymbol{0}$ hence $\langle \nabla f(\boldsymbol{x}), \boldsymbol{w} \rangle \approx 0$. In this case, the approximate steepest descending direction $\boldsymbol{d}$ for $f(\boldsymbol{x})$ is the solution to

$$\boldsymbol{d} = \operatorname*{argmin}_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^* \nabla^2 f(\boldsymbol{x}) \boldsymbol{w}, \quad \text{subject to} \quad \|\boldsymbol{w}\|_2 = 1. \tag{9.3.13}$$

Then $\boldsymbol{d}$ is the eigenvector $\boldsymbol{e} \in \mathbb{R}^n$ associated with the smallest (negative) eigenvalue $\lambda_{\min}$ of the Hessian. To simplify notation, here we define $\boldsymbol{H} \doteq \nabla^2 f(\boldsymbol{x})$. So we have:

$$\boldsymbol{H}\boldsymbol{e} = \lambda_{\min}(\boldsymbol{H})\boldsymbol{e}.$$

Geometrically, this is the direction in which the surface of $f(\boldsymbol{x})$ has the most negative curvature. Note that $\boldsymbol{d}$ can have two choices: $\boldsymbol{d} = \pm\boldsymbol{e}$. If $\boldsymbol{x}$ is not precisely a critical point, i.e., $\nabla f(\boldsymbol{x})$ is not zero, we usually choose $\boldsymbol{d}$ to align with the descent direction:

$$\langle \nabla f(\boldsymbol{x}), \boldsymbol{d} \rangle \le 0. \tag{9.3.14}$$

Recall that we have analyzed the problem (4.2.4) of computing the largest eigenvalue and eigenvector of a matrix in Chapter 4. Here we are interested in the smallest (likely negative) eigenvalue and the associated eigenvector. Notice that the Lipschitz condition (9.1.4) implies that $L_1$ is an upper bound of the largest eigenvalue $\max_i |\lambda_i|$ of $\boldsymbol{H}$. Hence, if we define a new matrix

$$\boldsymbol{A} \doteq \boldsymbol{I} - L_1^{-1}\boldsymbol{H} \succ \boldsymbol{0},$$

then the largest eigenvalue and eigenvector of $\boldsymbol{A}$ are:

$$\lambda_{\max}(\boldsymbol{A}) = 1 - \lambda_{\min}(\boldsymbol{H})/L_1 > 0, \quad \text{and} \quad \boldsymbol{A}\boldsymbol{e} = \lambda_{\max}(\boldsymbol{A})\boldsymbol{e}.$$

This eigenvector $\boldsymbol{e}$ is exactly the most negative curvature direction of the Hessian:

$$\boldsymbol{H}\boldsymbol{e} = \lambda_{\min}(\boldsymbol{H})\boldsymbol{e}.$$

The analysis of singular vectors given in Section 4.2.1 of Chapter 4 suggests that computing the largest eigenvalue/eigenvector can be rather efficient, say using the power iteration method in Exercise 4.6 – we will give a more general account of power iteration methods later in Section 9.6. In light of designing scalable optimization algorithms, we here give a more precise account of its complexity subject to a prescribed accuracy.

*Power Iteration and Lanczos Method.*
The power iteration and Lanczos method [KW92] are two popular methods to compute the leading eigenvalue and eigenvector of a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. They both rely on computing a series of matrix-vector products of $\boldsymbol{A}$ with a random vector $\boldsymbol{b} \in \mathbb{R}^n$, known as the *Krylov information*:

$$\boldsymbol{K} \doteq \big[\boldsymbol{b}, \boldsymbol{A}\boldsymbol{b}, \boldsymbol{A}^2\boldsymbol{b}, \ldots, \boldsymbol{A}^k\boldsymbol{b}\big]. \tag{9.3.15}$$

Notice that in our context, the matrix-vector product $\boldsymbol{A}\boldsymbol{b}$ depends only on the Hessian-vector product $\boldsymbol{H}\boldsymbol{b}$ which in turn can be approximated from the difference of two gradients:

$$\boldsymbol{A}\boldsymbol{b} = \big[\boldsymbol{I} - L_1^{-1}\boldsymbol{H}\big]\,\boldsymbol{b} \;\approx\; \boldsymbol{b} - (tL_1)^{-1}\big(\nabla f(\boldsymbol{x} + t\boldsymbol{b}) - \nabla f(\boldsymbol{x})\big), \tag{9.3.16}$$

for some small $t > 0$. This can be done recursively for all the products $\boldsymbol{A}^i\boldsymbol{b}$ in $\boldsymbol{K}$, for $i = 1, \ldots, k$.

Then, based on the Krylov information, the power iteration and Lanczos method estimate the largest eigenvalue $\lambda_{\max}(\boldsymbol{A})$ respectively as:

$$\text{Power iteration:} \quad \hat{\lambda}_{k+1} = \frac{\langle \boldsymbol{A}\boldsymbol{x}, \boldsymbol{x}\rangle}{\langle \boldsymbol{x}, \boldsymbol{x}\rangle}, \quad \boldsymbol{x} = \boldsymbol{A}^k\boldsymbol{b}; \tag{9.3.17}$$

$$\text{Lanczos method:} \quad \hat{\lambda}_{k+1} = \max_{\boldsymbol{x}} \frac{\langle \boldsymbol{A}\boldsymbol{x}, \boldsymbol{x}\rangle}{\langle \boldsymbol{x}, \boldsymbol{x}\rangle}, \quad \boldsymbol{x} \in \operatorname{span}(\boldsymbol{K}), \tag{9.3.18}$$

for $k = 0, 1, \ldots$. In our context, we are interested in precisely how many iterations (hence number of gradient evaluations) are needed in order to obtain an estimate within a prescribed accuracy $\varepsilon > 0$:

$$\left| \frac{\hat{\lambda} - \lambda_{\max}(\boldsymbol{A})}{\lambda_{\max}(\boldsymbol{A})} \right| \le \varepsilon. \tag{9.3.19}$$

Of course, it is easy to see that this cannot always be achieved for all matrices $\boldsymbol{A}$ if the vector $\boldsymbol{b}$ is fixed. One only has to consider the special case (of zero probability though) when $\boldsymbol{b}$ is perpendicular to the leading eigenvector: $\boldsymbol{b} \perp \boldsymbol{e}$. We leave this as an exercise to the reader.

*Random Initialization.*
Nevertheless, one can expect this to work with high probability for a *randomly chosen* $\boldsymbol{b}$. The usage of randomness here is to help avoid zero-measure pathological (or hard) cases mentioned above. In the next section, we will utilize randomness in a similar spirit – adding some random noise to gradient descent can help escape spurious critical points. From a random initialization, we can also precisely characterize how quickly the iteration reaches the desired accuracy.

THEOREM 9.7 (Convergence Rates of Power Iteration and Lanczos Method). *Let* $\boldsymbol{b}$ *is chosen randomly from a uniform distribution on the sphere* $\mathbb{S}^{n-1}$*, then we have:*

$$\textit{Power iteration:} \quad \mathbb{E}_{\boldsymbol{b}}\left[\left|\frac{\hat{\lambda}_{k+1}(\boldsymbol{b})-\lambda_{\max}(\boldsymbol{A})}{\lambda_{\max}(\boldsymbol{A})}\right|\right] \le c_1 \log(n)/k; \tag{9.3.20}$$

$$\textit{Lanczos method:} \quad \mathbb{E}_{\boldsymbol{b}}\left[\left|\frac{\hat{\lambda}_{k+1}(\boldsymbol{b})-\lambda_{\max}(\boldsymbol{A})}{\lambda_{\max}(\boldsymbol{A})}\right|\right] \le c_2(\log(n)/k)^2. \tag{9.3.21}$$

*for some small constants* $c_1, c_2 > 0$.

That is, the expected error in the estimated largest eigenvalue converges to zero at the rate of $O(\log(n)/k)$ and $O((\log(n)/k)^2)$ for the power iteration and Lanczos method, respectively. Or equivalently, to reach a prescribed accuracy $\varepsilon$ as in (9.3.19), the number of iterations needed is $O(\log(n)/\varepsilon)$ and $O(\log(n)/\sqrt{\varepsilon})$, respectively. One may refer to [KW92] for a detailed proof for the theorem above.

*Approximate Least Eigenvalue and Eigenvector.*
The above theorem immediately leads to a result that is very useful in our setting [RW18].

COROLLARY 9.8. *Let* $\boldsymbol{H}$ *be a symmetric matrix satisfying* $\|\boldsymbol{H}\| \le L_1$ *for some* $L_1 > 0$*. Suppose that the Lanczos procedure is applied to find the largest eigenvalue of* $L_1\boldsymbol{I} - \boldsymbol{H}$ *starting from a random vector uniformly distributed over the unit sphere. Then, for any* $\varepsilon_\lambda > 0$ *and* $\delta \in (0,1)$*, there is a probability at least* $1-\delta$ *that the procedure outputs a unit vector* $\boldsymbol{e}'$ *such that*

$$(\boldsymbol{e}')^* \boldsymbol{H} \boldsymbol{e}' \le \lambda_{\min}(\boldsymbol{H}) + \varepsilon_\lambda \tag{9.3.22}$$

*in at most*

$$\min\left\{n, \frac{\log(n/\delta^2)}{2\sqrt{2}}\sqrt{\frac{L_1}{\varepsilon_\lambda}}\right\} \tag{9.3.23}$$

*iterations. The procedure obtains a unit vector* $\boldsymbol{e}$ *such that* $\boldsymbol{e}^*\boldsymbol{H}\boldsymbol{e} = \lambda_{\min}(\boldsymbol{H})$ *after at most* $n$ *iterations.*

### 9.3.3 Overall Complexity in First-order Oracle

Now we know we can use the power iteration or Lanczos method to compute the direction of negative curvature. This essentially reduces the computation to a series of Hessian vector product operations that involve evaluating gradients (9.3.16). If we use the first-order oracle, evaluating a gradient, as the basic unit for measuring the complexity of an algorithm, then how we can measure or estimate the complexity of the proposed algorithm precisely?

Notice from the proof of Theorem 9.5, each negative curvature descent step, the function value decreases about $O(\varepsilon_H^3)$. The Lanczos process above, can estimate the least eigenvalue up to the precision $O(\varepsilon_H)$ for about $O(\varepsilon_H^{-1/2})$ iterations (or Hessian vector products). Hence per gradient evaluation, we can achieve a descent of $O(\varepsilon_H^{7/2})$. For this to be on par with the pure gradient descent, that is an $\varepsilon$ descent per gradient, we could set $\varepsilon = O(\varepsilon_H^{7/2})$ or $\varepsilon_H = O(\varepsilon^{2/7})$. Then the overall complexity in terms of the number of gradient evaluations, can be bounded as $O(\varepsilon^{-1})$ or $O(\varepsilon_g^{-2})$. That is, the above hybrid scheme has the same computational complexity in terms of first-order oracle as the gradient descent scheme, introduced in the beginning of the chapter (see Proposition 9.1). However, it guarantees to converge to a second-order stationary point.

To see this more rigorously, we can modify the hybrid gradient and negative curvature algorithm in Figure 9.2 as follows. Whenever the gradient is below $\varepsilon_g$, we use Lanczos method to compute an inexact unit eigenvector $\boldsymbol{e}'_k$ such that (with probability $1-\delta$)

$$\langle \boldsymbol{e}'_k, \nabla f(\boldsymbol{x}_k)\rangle \le 0, \quad \lambda'_k \le \lambda_{\min}\left(\nabla^2 f(\boldsymbol{x}_k)\right) + \frac{\varepsilon_H}{2}, \tag{9.3.24}$$

where $\lambda'_k := (\boldsymbol{e}'_k)^* \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{e}'_k$. We know from Corollary 9.8, this requires $O(\varepsilon_H^{-1/2})$ of Hessian vector product operations or gradient evaluations. With the inexact eigenvector, we conduct negative curvature descent when $\lambda'_k \le -\frac{\varepsilon_H}{2}$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \frac{2\lambda'_k}{L_2}\boldsymbol{e}'_k. \tag{9.3.25}$$

Then similar to the proof in Theorem 9.5, we have

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) &\le \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \frac{1}{2}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)^* \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) \\ &\quad + \frac{L_2}{6}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2^3 \\ &= \left\langle \nabla f(\boldsymbol{x}_k), \frac{2\lambda_k'}{L_2}\boldsymbol{e}_k' \right\rangle + \frac{1}{2}\left(\frac{2\lambda_k'}{L_2}\boldsymbol{e}_k'\right)^* \nabla^2 f(\boldsymbol{x}_k)\left(\frac{2\lambda_k'}{L_2}\boldsymbol{e}_k'\right) \\ &\quad + \frac{L_2}{6}\left\|\frac{2\lambda_k'}{L_2}\boldsymbol{e}_k'\right\|_2^3 \\ &\le \frac{1}{2}\left(\frac{2\lambda_k'}{L_2}\boldsymbol{e}_k'\right)^* \nabla^2 f(\boldsymbol{x}_k)\left(\frac{2\lambda_k'}{L_2}\boldsymbol{e}_k'\right) + \frac{L_2}{6}\left\|\frac{2\lambda_k'}{L_2}\boldsymbol{e}_k'\right\|_2^3 \\ &= \frac{2(\lambda_k')^3}{L_2^2} + \frac{4|\lambda_k'|^3}{3L_2^2} = \frac{2(\lambda_k')^3}{3L_2^2} \\ &\le -\frac{\varepsilon_H^3}{12L_2^2}. \end{aligned} \tag{9.3.26}$$

Therefore, the total descent is $\frac{1}{12L_2^2}\varepsilon_H^3$ for $O(\varepsilon_H^{-1/2})$ gradient evaluations. The average descent per gradient is $O(\varepsilon_H^{7/2})$. With the choice $\varepsilon_H = O(\varepsilon^{2/7})$, the per gradient evaluation will incur a descent of $O(\varepsilon)$. Hence, the number of iterations is $k \le O(\varepsilon^{-1})$. With the same choice of $\varepsilon_g = O(\varepsilon^{1/2})$, the overall computational complexity of the inexact negative curvature descent in terms of the first-order oracle is[14]

$$k \le O(\varepsilon_g^{-2}),$$

and the scheme guarantees to converge to a critical point $\boldsymbol{x}_\star$ that satisfies:

$$\|\nabla f(\boldsymbol{x}_\star)\|_2 \le O(\varepsilon^{1/2}), \quad -\lambda_{\min}(\nabla^2 f(\boldsymbol{x}_\star)) \le O(\varepsilon^{2/7}). \tag{9.3.27}$$

## 9.4 Negative Curvature and Newton Descent

As we have seen in the cubic regularized Newton's method in Section 9.2, if we have access to the second-order oracle (the gradient and Hessian), the best rate of convergence can be achieved is $O(\varepsilon_g^{-1.5})$. However, if we have access only to the gradient, then for functions with Lipschitz gradient and Hessian, then the lower bound of first-order methods can be relaxed to $\Omega(\varepsilon_g^{-12/7})$ [CDHS17], while the best known achievable upper bound is $O(\varepsilon_g^{-7/4})$.

We notice that the above hybrid gradient and negative curvature descent converges at the rate $O(\varepsilon_g^{-2})$ and does not yet achieve the best known complexity result. The main problem is with the step of gradient descent: to achieve the prescribed descent $\varepsilon$ per gradient step, it requires the gradient is at least in the

[14] up to some log factor in $n$.

order of $O(\varepsilon^{1/2})$, disregarding any second-order information. When the gradient is not as large, to achieve the same amount of descent, one must leverage second-order information about the Hessian as we did in the above Newton type methods.

In this section, we show that a slightly more careful use of the negative curvature information (computed from gradients) can indeed lead to algorithms that reach the best known complexity bound. For readers who are not interested in such theoretical guarantee, they may skip this section without loss of continuity.

### 9.4.1 Curvature Guided Newton Descent

In the preceding algorithm, the negative curvature descent step offers useful second-order information about the function that probably can be utilized by the gradient step, a key observation by [RW18]. This suggests that we could reverse the order of the two steps: We first evaluate the smallest eigenvalue $\lambda_{\min}$ of the Hessian $\nabla^2 f(\boldsymbol{x})$. Based on its value, we decide to conduct either a negative curvature descent or a more effective descent based on the gradient $\nabla f(\boldsymbol{x})$.

Notice that for the later choice, with the second-order information about the negative curvature, we can conduct a more effective regularized Newton type descent:

$$\boldsymbol{s}_k = \operatorname*{argmin}_{\boldsymbol{s}} \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{s} \rangle + \frac{1}{2} \boldsymbol{s}^* \nabla^2 f(\boldsymbol{x}_k) \boldsymbol{s} + \frac{\lambda}{2} \|\boldsymbol{s}\|_2^2 \tag{9.4.1}$$

with $\lambda > \lambda_{\min}$. The choice of the quadratic regularization term $\lambda \|\boldsymbol{s}\|_2^2$ ensures the function is strongly convex in $\boldsymbol{s}$ or equivalently, $\nabla^2 f(\boldsymbol{x}) + \lambda \boldsymbol{I} \succ \boldsymbol{0}$ is positive definite. If we directly use the so computed optimal $\boldsymbol{s}_k = -[\nabla^2 f(\boldsymbol{x}_k) + \lambda \boldsymbol{I}]^{-1} \nabla f(\boldsymbol{x}_k)$ as increment, we arrive at the well-known *Levenberg-Marquardt* method:[15]

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\nabla^2 f(\boldsymbol{x}_k) + \lambda \boldsymbol{I}\right]^{-1} \nabla f(\boldsymbol{x}_k). \tag{9.4.2}$$

Nevertheless, here to ensure the function value to decrease by at least the prescribed amount, we should be judicious about the step size $\gamma_k$ along the direction $\boldsymbol{s}_k$:[16]

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \gamma_k \boldsymbol{s}_k. \tag{9.4.3}$$

Figure 9.3 and the following theorem give the proper conditions under which the above hybrid scheme converges to a second-order stationary point.

[15] We will provide more references to the Levenberg-Marquardt method in the Notes section. Similar update rule can be derived from the perspective of the trust region method, as we will see in Exercise 9.2.

[16] In optimization, a good step size is often found through a "line search" step. Nevertheless, when the function Lipschitz constants are given, we can give an explicit expression for the proper step size.

**Hybrid Negative Curvature and Newton Descent**

**Problem Class:**

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n,$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable, with Lipschitz continuous gradient and Hessian. Have access to the oracle: $\nabla f(\boldsymbol{x})$ and $\nabla^2 f(\boldsymbol{x})$.

**Setup:** given a prescribed accuracy $\varepsilon > 0$, $\varepsilon_g = 3^{8/3} L_2^{1/3} \varepsilon^{2/3}/2, \varepsilon_H = \left(3L_2^2 \varepsilon\right)^{1/3}$.

**Initialization:** Set $\boldsymbol{x}_0 \in \mathbb{R}^n$.

**For** $k = 0, 1, 2, \ldots$

1 Compute $\nabla f(\boldsymbol{x}_k)$, and the smallest eigenvalue and unit eigenvector pair $(\lambda_k, \boldsymbol{e}_k)$ of $\nabla^2 f(\boldsymbol{x}_k)$ with $\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{e}_k \rangle \leq 0$.

2 **if** $\lambda_k \leq -\varepsilon_H$, **then** conduct negative curvature descent:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \frac{2\lambda_k}{L_2} \boldsymbol{e}_k; \tag{9.4.4}$$

3 **else if** $\|\nabla f(\boldsymbol{x}_k)\|_2 \geq \varepsilon_g$, then solve the convex quadratic problem:

$$\boldsymbol{s}_k = \operatorname*{argmin}_{\boldsymbol{s}} \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{s} \rangle + \frac{1}{2} \boldsymbol{s}^* \nabla^2 f(\boldsymbol{x}_k) \boldsymbol{s} + \varepsilon_H \|\boldsymbol{s}\|_2^2, \tag{9.4.5}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \gamma_k \boldsymbol{s}_k, \tag{9.4.6}$$

with $\gamma_k = \min\left\{ \left( \frac{3\varepsilon_H}{2L_2 \|\boldsymbol{s}_k\|_2} \right)^{1/2}, 1 \right\}$.

4 **else end for** and **return** $\boldsymbol{x}_\star = \boldsymbol{x}_k$.

**Convergence Guarantee:** $\|\nabla f(\boldsymbol{x}_\star)\|_2 \leq \varepsilon_g$, $-\lambda_{\min}(\nabla^2 f(\boldsymbol{x}_\star)) \leq \varepsilon_H$.

**Figure 9.3** An overview of the Hybrid Negative Curvature and Newton Descent.

THEOREM 9.9 (Convergence of Hybrid Negative Curvature and Newton Descent). *Assume $\{\boldsymbol{x}_k\}$ are generated by the hybrid negative curvature and Newton descent algorithm in Figure 9.3. Then in at most*

$$k \leq \frac{f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star)}{\varepsilon} \tag{9.4.7}$$

*iterations, $\boldsymbol{x}_k$ will be an approximate second-order stationary point such that $\|\nabla f(\boldsymbol{x}_k)\|_2 \leq \varepsilon_g, \lambda_{\min}(\nabla^2 f(\boldsymbol{x}_k)) \geq -\varepsilon_H$, where*

$$\varepsilon_g = 3^{8/3}/2 L_2^{1/3} \varepsilon^{2/3}, \quad \varepsilon_H = \left(3L_2^2 \varepsilon\right)^{1/3}.$$

*Proof* If $\lambda_k \leq -\varepsilon_H$ or $-\lambda_k \geq \left(3L_2^2 \varepsilon\right)^{1/3}$, we conduct negative curvature descent (9.4.4). From the proof of Theorem 9.5, we know that then we have

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leq \frac{2(\lambda_k)^3}{3L_2^2} \leq -\frac{2\varepsilon_H^3}{3L_2^2} = -2\varepsilon. \tag{9.4.8}$$

If $\lambda_k > -\varepsilon_H$, then we discuss two cases.

*Case 1.* If $\left(\frac{3\varepsilon_H}{2L_2\|\boldsymbol{s}_k\|_2}\right)^{1/2} \geq 1$, that is, $\|\boldsymbol{s}_k\|_2 \leq \frac{3\varepsilon_H}{2L_2}$, we accept the unit step size. By the optimality condition of $\boldsymbol{s}_k$ in (9.4.5), we have

$$\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k + 2\varepsilon_H \boldsymbol{s}_k + \nabla f(\boldsymbol{x}_k) = \boldsymbol{0}. \tag{9.4.9}$$

Then together with the property of Lipschitz Hessian condition (9.2.13), we have

$$\begin{aligned}
\|\nabla f(\boldsymbol{x}_{k+1})\|_2 = \|\nabla f(\boldsymbol{x}_k + \boldsymbol{s}_k)\| &\leq \|\nabla f(\boldsymbol{x}_k + \boldsymbol{s}_k) - (\nabla f(\boldsymbol{x}_k) + \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k)\|_2 \\
&\quad + \|\nabla f(\boldsymbol{x}_k) + \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k\|_2 \\
&\leq \frac{L_2}{2}\|\boldsymbol{s}_k\|_2^2 + \|2\varepsilon_H \boldsymbol{s}_k\|_2 \ \leq \ \frac{L_2}{2}\|\boldsymbol{s}_k\|_2^2 + 2\varepsilon_H\|\boldsymbol{s}_k\|_2 \\
&\leq \left(\frac{9}{8} + 3\right)\frac{\varepsilon_H^2}{L_2} \ \leq \ \frac{9\varepsilon_H^2}{2L_2} \\
&\leq \varepsilon_g.
\end{aligned} \tag{9.4.10}$$

Then, by the property of Lipschitz Hessian condition (9.2.4), we have

$$\begin{aligned}
f(\boldsymbol{x}_{k+1}) &= f(\boldsymbol{x}_k + \boldsymbol{s}_k) \\
&\leq f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{s}_k\rangle + \frac{1}{2}\boldsymbol{s}_k^* \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k + \frac{L_2}{6}\|\boldsymbol{s}_k\|_2^3 \\
&\leq f(\boldsymbol{x}_k) - \frac{1}{2}\boldsymbol{s}_k^* \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k - 2\varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{L_2}{6}\|\boldsymbol{s}_k\|_2^3 \\
&\leq f(\boldsymbol{x}_k) - \frac{3}{2}\varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{L_2}{6}\|\boldsymbol{s}_k\|_2^3 \\
&\leq f(\boldsymbol{x}_k) - \frac{3}{2}\varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{\varepsilon_H}{4}\|\boldsymbol{s}_k\|_2^2 \\
&\leq f(\boldsymbol{x}_k) - \frac{5}{4}\varepsilon_H\|\boldsymbol{s}_k\|_2^2.
\end{aligned} \tag{9.4.11}$$

That is, when the step size $\gamma_k = 1$, we have that $\nabla f(\boldsymbol{x}_{k+1})$ is already smaller than $\varepsilon_g$ and $f(\boldsymbol{x}_{k+1})$ is smaller than $f(\boldsymbol{x}_k)$. As a result, we must have

$$\lambda_{\min}(\nabla^2 f(\boldsymbol{x}_{k+1})) < -\varepsilon_H;$$

otherwise, we have found a desired second-order stationary point. So, for the case of accepting step size 1, before the algorithm stops, the function value will be decreased by at least $2\varepsilon$ in the next iteration by negative curvature descent (9.4.8).

*Case 2.* If $\left(\frac{3\varepsilon_H}{2L_2\|\boldsymbol{s}_k\|_2}\right)^{1/2} < 1$, that is, $\|\boldsymbol{s}_k\|_2 > \frac{3\varepsilon_H}{2L_2}$. To simplify notation, we

let $\alpha = \left(\frac{3\varepsilon_H}{2L_2\|\boldsymbol{s}_k\|_2}\right)^{1/2} < 1$. Then we have

$$\begin{aligned}
f(\boldsymbol{x}_{k+1}) &= f(\boldsymbol{x}_k + \alpha \boldsymbol{s}_k) \\
&\leq f(\boldsymbol{x}_k) + \alpha\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{s}_k\rangle + \frac{\alpha^2}{2}\boldsymbol{s}_k^*\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k + \frac{L_2\alpha^3}{6}\|\boldsymbol{s}_k\|_2^3 \\
&\leq f(\boldsymbol{x}_k) + \alpha\big(\frac{\alpha}{2} - 1\big)\boldsymbol{s}_k^*\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k - 2\alpha\varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{L_2\alpha^3}{6}\|\boldsymbol{s}_k\|_2^3 \\
&\leq f(\boldsymbol{x}_k) - \alpha\varepsilon_H\big(\frac{\alpha}{2} - 1\big)\|\boldsymbol{s}_k\|_2^2 - 2\alpha\varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{L_2\alpha^3}{6}\|\boldsymbol{s}_k\|_2^3 \\
&\leq f(\boldsymbol{x}_k) - \alpha\varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{L_2\alpha^3}{6}\|\boldsymbol{s}_k\|_2^3 \\
&= f(\boldsymbol{x}_k) - \Big(\frac{3}{2L_2}\Big)^{1/2}(\varepsilon_H\|\boldsymbol{s}_k\|_2)^{3/2} + \frac{(3/2)^{3/2}}{6L_2^{1/2}}(\varepsilon_H\|\boldsymbol{s}_k\|_2)^{3/2} \\
&\leq f(\boldsymbol{x}_k) - \frac{(3/2)^{1/2}3}{4L_2^{1/2}}(\varepsilon_H\|\boldsymbol{s}_k\|_2)^{3/2} \\
&\leq f(\boldsymbol{x}_k) - \frac{27\varepsilon_H^3}{16L_2^2} \\
&= f(\boldsymbol{x}_k) - 5\varepsilon. \qquad (9.4.12)
\end{aligned}$$

Combining (9.4.8)-(9.4.12), we know that before finding an approximate second-order stationary point such that $\|\nabla f(\boldsymbol{x}_k)\|_2 \leq \varepsilon_g, \lambda_{\min}(\nabla^2 f(\boldsymbol{x}_k)) \geq -\varepsilon_H$, we can always decrease the function value by at least $2\varepsilon$ in two consecutive iterations. As a result, to find such point the total number of iterations will be upper bounded by $k \leq \frac{f(\boldsymbol{x}_0)-f(\boldsymbol{x}_*)}{\varepsilon}$.

□

### 9.4.2 Inexact Negative Curvature and Newton Descent

In the above scheme, we have assumed that we have access to the Hessian and its smallest eigenvalue and eigenvector. However, if we only have access to the gradient and the Hessian-vector product, how costly would it be to compute the eigenvector? How accurately should it be computed so that the resulting scheme achieves the best known complexity (w.r.t. the first-order oracle)?

In this section, we consider an inexact version of the algorithm in Figure 9.3, which allows us to approximately compute the smallest eigenvalue and eigenvector pair, and approximately solve the convex quadratic problem. By carefully choosing the stopping criterion, the inexact version of the algorithm, shown in Figure 9.4, can maintain the convergence rate of the exact version, differing only in constants. The corresponding convergence result is given in the theorem below.

THEOREM 9.10. *Assume $\{\boldsymbol{x}_k\}$ are generated by the hybrid negative curvature*

**Inexact Hybrid Negative Curvature and Newton Descent**

**Problem Class:**

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n,$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable, with Lipschitz continuous gradient and Hessian. Have access to the oracle: $\nabla f(\boldsymbol{x})$ and the Hessian product $\nabla^2 f(\boldsymbol{x})\boldsymbol{v}$.

**Setup:** prescribed accuracy $\varepsilon > 0$, $\varepsilon_g = (5/L_2)(24L_2^2\varepsilon)^{2/3}, \varepsilon_H = \left(24L_2^2\varepsilon\right)^{1/3}$.

**Initialization:** Set $\boldsymbol{x}_0 \in \mathbb{R}^n$.

**For** $k = 0, 1, 2, \ldots$

1 Compute $\nabla f(\boldsymbol{x}_k)$ and an inexact unit eigenvector $\boldsymbol{e}'_k$ such that (with probability $1 - \delta$)

$$\langle \boldsymbol{e}'_k, \nabla f(\boldsymbol{x}_k)\rangle \le 0, \quad \lambda'_k \le \lambda_{\min}\left(\nabla^2 f(\boldsymbol{x}_k)\right) + \frac{\varepsilon_H}{2}, \tag{9.4.13}$$

where $\lambda'_k := (\boldsymbol{e}'_k)^* \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{e}'_k$.

2 **if** $\lambda'_k \le -\frac{\varepsilon_H}{2}$, **then** conduct negative curvature descent:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \frac{2\lambda'_k}{L_2}\boldsymbol{e}'_k; \tag{9.4.14}$$

3 **else if** $\|\nabla f(\boldsymbol{x}_k)\|_2 \ge \varepsilon_g$, then find $\boldsymbol{s}_k$ such that

$$\|\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k + 2\varepsilon_H \boldsymbol{s}_k + \nabla f(\boldsymbol{x}_k)\|_2 \le \frac{1}{2}\varepsilon_H \|\boldsymbol{s}_k\|_2, \tag{9.4.15}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \gamma_k \boldsymbol{s}_k, \tag{9.4.16}$$

where $\gamma_k = \min\left\{\left(\frac{3\varepsilon_H}{2L_2\|\boldsymbol{s}_k\|_2}\right)^{1/2}, 1\right\}$.

4 **else end for** and **return** $\boldsymbol{x}_\star = \boldsymbol{x}_k$.

**Convergence Guarantee:** $\|\nabla f(\boldsymbol{x}_\star)\|_2 \le \varepsilon_g$, $-\lambda_{\min}\left(\nabla^2 f(\boldsymbol{x}_\star)\right) \le \varepsilon_H$.

**Figure 9.4** Overview of the Inexact Hybrid Negative Curvature and Newton Descent.

*and Newton descent algorithm in Figure 9.4. Then in at most*

$$k \le \frac{f(\boldsymbol{x}_0) - f(\boldsymbol{x}_\star)}{\varepsilon} \tag{9.4.17}$$

*iterations, $\boldsymbol{x}_k$ will be an an approximate second-order stationary point such that $\|\nabla f(\boldsymbol{x}_k)\|_2 \le \varepsilon_g, \lambda_{\min}(\nabla^2 f(\boldsymbol{x}_k)) \ge -\varepsilon_H$, where*

$$\varepsilon_g = (5/L_2)(24L_2^2\varepsilon)^{2/3}, \quad \varepsilon_H = \left(24L_2^2\varepsilon\right)^{1/3}.$$

*Proof* If $\lambda'_k \le -\frac{\varepsilon_H}{2}$, we estimate the amount of descent by the negative curvature descent. This is exactly the same as we have done in (9.3.26). The slight

difference here is the choice of $\varepsilon_H$. Hence we have

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \le -\frac{\varepsilon_H^3}{12L_2^2} \;=\; -2\varepsilon. \tag{9.4.18}$$

If $\lambda_k' > -\frac{\varepsilon_H}{2}$, then by the conditions of $\lambda_k'$, we have

$$-\frac{\varepsilon_H}{2} \le \lambda_k' \le \lambda_{\min}\left(\nabla^2 f(\boldsymbol{x}_k)\right) + \frac{\varepsilon_H}{2}, \tag{9.4.19}$$

i.e, we have $\lambda_{\min}\left(\nabla^2 f(\boldsymbol{x}_k)\right) \ge -\varepsilon_H$. Then we discuss in two cases.

*Case 1.* If $\left(\frac{3\varepsilon_H}{2L_2\|\boldsymbol{s}_k\|_2}\right)^{1/2} \ge 1$, that is, $\|\boldsymbol{s}_k\|_2 \le \frac{3\varepsilon_H}{2L_2}$, then we accept the unit step size. Letting

$$\boldsymbol{r}_k := \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k + 2\varepsilon_H \boldsymbol{s}_k + \nabla f(\boldsymbol{x}_k), \tag{9.4.20}$$

we know $\|\boldsymbol{r}_k\|_2 \le \frac{1}{2}\varepsilon_H\|\boldsymbol{s}_k\|_2$. By the Lipschitz Hessian condition (9.2.13), we have

$$\begin{aligned}
&\|\nabla f(\boldsymbol{x}_{k+1})\|_2 = \|\nabla f(\boldsymbol{x}_k + \boldsymbol{s}_k)\|_2 \\
&\le \|\nabla f(\boldsymbol{x}_k + \boldsymbol{s}_k) - (\nabla f(\boldsymbol{x}_k) + \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k)\|_2 + \|\nabla f(\boldsymbol{x}_k) + \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k\|_2 \\
&\le \frac{L_2}{2}\|\boldsymbol{s}_k\|_2^2 + \|\boldsymbol{r}_k - 2\varepsilon_H \boldsymbol{s}_k\|_2 \;\le\; \frac{L_2}{2}\|\boldsymbol{s}_k\|_2^2 + 2\varepsilon_H\|\boldsymbol{s}_k\|_2 + \|\boldsymbol{r}_k\|_2 \\
&\le \frac{L_2}{2}\|\boldsymbol{s}_k\|_2^2 + 2\varepsilon_H\|\boldsymbol{s}_k\|_2 + \frac{1}{2}\varepsilon_H\|\boldsymbol{s}_k\|_2 \;\le\; \left(\frac{9}{8} + 3 + \frac{3}{4}\right)\frac{\varepsilon_H^2}{L_2} \\
&\le \frac{5\varepsilon_H^2}{L_2} \\
&= \varepsilon_g.
\end{aligned} \tag{9.4.21}$$

Then, by the Hessian Lipschitz condition (9.2.4), we have,

$$\begin{aligned}
&f(\boldsymbol{x}_{k+1}) \;=\; f(\boldsymbol{x}_k + \boldsymbol{s}_k) \\
&\le f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{s}_k\rangle + \frac{1}{2}\boldsymbol{s}_k^*\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k + \frac{L_2}{6}\|\boldsymbol{s}_k\|_2^3 \\
&= f(\boldsymbol{x}_k) + \langle \boldsymbol{r}_k - (\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k + 2\varepsilon_H \boldsymbol{s}_k), \boldsymbol{s}_k\rangle + \frac{1}{2}\boldsymbol{s}_k^*\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k + \frac{L_2}{6}\|\boldsymbol{s}_k\|_2^3 \\
&\le f(\boldsymbol{x}_k) + \langle \boldsymbol{r}_k, \boldsymbol{s}_k\rangle - \frac{1}{2}\boldsymbol{s}_k^*\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k - 2\varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{L_2}{6}\|\boldsymbol{s}_k\|_2^3 \\
&\le f(\boldsymbol{x}_k) + \|\boldsymbol{r}_k\|_2\|\boldsymbol{s}_k\|_2 - \frac{1}{2}\boldsymbol{s}_k^*\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k - 2\varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{L_2}{6}\|\boldsymbol{s}_k\|_2^3 \\
&\le f(\boldsymbol{x}_k) + \frac{1}{2}\varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{1}{2}\varepsilon_H\|\boldsymbol{s}_k\|_2^2 - 2\varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{L_2}{6}\|\boldsymbol{s}_k\|_2^3 \\
&\le f(\boldsymbol{x}_k) - \varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{L_2}{6}\|\boldsymbol{s}_k\|_2^3 \\
&\le f(\boldsymbol{x}_k) - \varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{\varepsilon_H}{4}\|\boldsymbol{s}_k\|_2^2 \\
&\le f(\boldsymbol{x}_k) - \frac{3}{4}\varepsilon_H\|\boldsymbol{s}_k\|_2^2.
\end{aligned} \tag{9.4.22}$$

That is, if we accept the step size $\gamma_k = 1$, then $\nabla f(\boldsymbol{x}_{k+1})$ is already smaller

than $\varepsilon_g$ and $f(\boldsymbol{x}_{k+1})$ is smaller than $f(\boldsymbol{x}_k)$. As a result, we next should have $\lambda_{\min}(\nabla^2 f(\boldsymbol{x}_{k+1})) < -\varepsilon_H$; otherwise, we have found a desired second-order stationary point. So for the case of accepting step size 1, before the algorithm stops, we must decrease the function value by at least $2\varepsilon$ in the next iteration by the negative curvature descent (9.4.18).

*Case 2.* If $\left(\frac{3\varepsilon_H}{2L_2\|\boldsymbol{s}_k\|_2}\right)^{1/2} < 1$, that is, $\|\boldsymbol{s}_k\|_2 > \frac{3\varepsilon_H}{2L_2}$. For simplicity, we denote $\alpha = \left(\frac{3\varepsilon_H}{2L_2\|\boldsymbol{s}_k\|_2}\right)^{1/2} < 1$. Then we have

$$
\begin{aligned}
& f(\boldsymbol{x}_{k+1}) \;=\; f(\boldsymbol{x}_k + \alpha \boldsymbol{s}_k) \\
\leq\; & f(\boldsymbol{x}_k) + \alpha\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{s}_k\rangle + \frac{\alpha^2}{2}\boldsymbol{s}_k^* \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k + \frac{L_2\alpha^3}{6}\|\boldsymbol{s}_k\|_2^3 \\
\leq\; & f(\boldsymbol{x}_k) + \alpha\|\boldsymbol{r}_k\|_2\|\boldsymbol{s}_k\|_2 + \alpha\big(\frac{\alpha}{2}-1\big)\boldsymbol{s}_k^*\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k - 2\alpha\varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{L_2\alpha^3}{6}\|\boldsymbol{s}_k\|_2^3 \\
\leq\; & f(\boldsymbol{x}_k) + \frac{\alpha\varepsilon_H}{2}\|\boldsymbol{s}_k\|_2^2 - \alpha\varepsilon_H\big(\frac{\alpha}{2}-1\big)\|\boldsymbol{s}_k\|_2^2 - 2\alpha\varepsilon_H\|\boldsymbol{s}_k\|_2^2 + \frac{L_2\alpha^3}{6}\|\boldsymbol{s}_k\|_2^3 \\
\leq\; & f(\boldsymbol{x}_k) - \frac{\alpha\varepsilon_H}{2}\|\boldsymbol{s}_k\|_2^2 + \frac{L_2\alpha^3}{6}\|\boldsymbol{s}_k\|_2^3 \\
=\; & f(\boldsymbol{x}_k) - \frac{1}{2}\Big(\frac{3}{2L_2}\Big)^{1/2}(\varepsilon_H\|\boldsymbol{s}_k\|_2)^{3/2} + \frac{(3/2)^{3/2}}{6L_2^{1/2}}(\varepsilon_H\|\boldsymbol{s}_k\|_2)^{3/2} \\
\leq\; & f(\boldsymbol{x}_k) - \frac{(3/2)^{1/2}}{4L_2^{1/2}}(\varepsilon_H\|\boldsymbol{s}_k\|_2)^{3/2} \\
\leq\; & f(\boldsymbol{x}_k) - \frac{9\varepsilon_H^3}{16L_2^2} \\
\leq\; & f(\boldsymbol{x}_k) - \frac{27}{2}\varepsilon. \qquad (9.4.23)
\end{aligned}
$$

So when using step size less than 1, we can always guarantee sufficient decrease.

Combining (9.4.18)-(9.4.23), we know that before finding an approximate second-order stationary point such that $\|\nabla f(\boldsymbol{x}_k)\|_2 \leq \varepsilon_g, \lambda_{\min}\left(\nabla^2 f(\boldsymbol{x}_k)\right) \geq -\varepsilon_H$, we can always decrease the function value by at least $2\varepsilon$ in two consecutive iterations. As a result, to find such point the total number of iterations will be upper bounded by $k \leq \frac{f(\boldsymbol{x}_0)-f(\boldsymbol{x}_\star)}{\varepsilon}$.

□

### 9.4.3 Overall Complexity in First-order Oracle

Notice that in the inexact scheme above, we need to approximate both the eigenvector $\boldsymbol{e}'$ associated with the smallest eigenvalue (9.4.13) as well as find an approximate solution $\boldsymbol{s}_k$ to the convex quadratic problem (9.4.1) that satisfies the accuracy (9.4.15).

*Inexact Negative Curvature Descent.*
As we have characterized before in Section 9.3.3, to compute the smallest eigenvalue and eigenvector up to the prescribed accuracy $\varepsilon_H/2$, the number of Hessian-

vector products (or gradients) evaluations is of order $O(\varepsilon_H^{-1/2})$. With the choice $\varepsilon_H = O(\varepsilon^{1/3})$, this is equivalent to $O(\varepsilon^{-1/6})$.[17]

Then, according to the proof of Theorem 9.9, each negative curvature descent is $\varepsilon$. Hence per gradient evaluation, the descent is $O(\varepsilon^{7/6})$. The number of iteration $k = O(\varepsilon^{-7/6})$.

According to Theorem 9.9, the total number of iterations of the algorithm in Figure 9.4 is $O(\varepsilon^{-1})$, while per iteration we need $O(\varepsilon^{-1/6})$ number of Hessian-vector products to produce the desired inexact solution. So the total number of Hessian-vector products we need in negative curvature descent is $O(\varepsilon^{-7/6})$. Since we have $\varepsilon = O(\varepsilon_g^{3/2})$, this leads to the best known rate $k = O(\varepsilon_g^{-7/4})$.

*Inexact Convex Quadratic Program.*
Now, notice that we also need an approximate solution to the convex quadratic problem (9.4.1). The above rate will hold only if we can solve the problem (9.4.15) with the same complexity in first-order oracle for the Newton descent step. That is, we need to show that the number of Hessian vector products, hence gradient evaluations, needed to solve the quadratic problem approximately is also of order $O(\varepsilon_H^{-1/2})$, i.e., $O(\varepsilon^{-1/6})$.

By the optimality condition of the convex quadratic problem (9.4.5), we have

$$\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{s}_k + 2\varepsilon_H \boldsymbol{s}_k + \nabla f(\boldsymbol{x}_k) = \boldsymbol{0}. \tag{9.4.24}$$

This is equivalent to

$$(\nabla^2 f(\boldsymbol{x}_k) + 2\varepsilon_H \boldsymbol{I})\boldsymbol{s}_k = -\nabla f(\boldsymbol{x}_k), \tag{9.4.25}$$

which is of the form of a linear system: $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{b}$, with $\boldsymbol{A} = \nabla^2 f(\boldsymbol{x}_k) + 2\varepsilon_H \boldsymbol{I}, \boldsymbol{b} = -\nabla f(\boldsymbol{x}_k)$. Notice that in the above algorithm, when we conduct the Newton descent, we have the condition $\lambda_{\min}(\nabla^2 f(\boldsymbol{x}_k)) \geq -\varepsilon_H$. So for our problem here:

$$\varepsilon_H \boldsymbol{I} \preceq \boldsymbol{A} \preceq (L_1 + 2\varepsilon_H)\boldsymbol{I}.$$

Of course, one could simply compute the inverse of $\boldsymbol{A}$ to solve $\boldsymbol{s} = \boldsymbol{A}^{-1}\boldsymbol{b}$, but the complexity would be very high. To avoid computing matrix inverse, one could try to solve the program numerically

$$\min_{\boldsymbol{s}} \|\boldsymbol{A}\boldsymbol{s} - \boldsymbol{b}\|_2^2$$

using the steepest gradient descent. However, the complexity would not be the best one can do. The classic *conjugate gradient method*, described in equation (A.6.3) in Appendix A, is precisely an accelerated gradient algorithm designed to solve the above quadratic program more efficiently than the steepest descent. The reader may refer to [She94, NW06] for an excellent derivation and justification of this elegant, classical method.

For our interest here, one should notice that, at each iteration $i$, the conjugate

[17] Here for simplicity, we have omitted possible log factors in the orders.

gradient scheme only needs to evaluate the multiplication of $\boldsymbol{A}$ with the current estimate $\boldsymbol{s}_i$ to compute the residual:

$$\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i \boldsymbol{A}\boldsymbol{s}_i$$

for the next iteration. For our purpose, we need to characterize the precise number of iterations for the conjugate gradient method to produce an approximate solution that satisfies the following (relative) accuracy:

$$\|\boldsymbol{A}\boldsymbol{s} - \boldsymbol{b}\|_2 \leq \mu \|\boldsymbol{b}\|_2,$$

for some small $\mu > 0$. Then from the property of conjugate gradient, it is easy to show the following result for our problem.

THEOREM 9.11 (Complexity of Approximate Conjugate Gradient). *To solve $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{b}$ with $\alpha \boldsymbol{I} \preceq \boldsymbol{A} \preceq \beta \boldsymbol{I}$, the conjugate gradient method computes an $\boldsymbol{s}'$ that satisfies $\|\boldsymbol{A}\boldsymbol{s}' - \boldsymbol{b}\|_2 \leq \mu \|\boldsymbol{b}\|_2$ for $\mu \in (0,1)$ in at most*

$$\min \left\{ n, \frac{1}{2} \ln \left( \frac{4}{\mu} \left( \frac{\beta}{\alpha} \right)^{3/2} \right) \sqrt{\frac{\beta}{\alpha}} \right\} \tag{9.4.26}$$

*iterations.*

Interested readers may see [She94, RW18] for a proof. In the setting of our problem (9.4.15), we have $\alpha = \varepsilon_H$, $\mu = \frac{1}{2}\varepsilon_H$, and $\beta$ is bounded by a constant close to $L_1$. Therefore, the number of iterations, or matrix vector products, is of the order $O\big(\varepsilon_H^{-1/2} \log(\frac{1}{\varepsilon_H})\big)$. If we ignore the log factor, the complexity $\tilde{O}\big(\varepsilon_H^{-1/2}\big)$ is the same as that using the Lanczos method for computing the approximate solution to the smallest eigenvalue.

Putting together the respective complexity of the inexact negative curvature descent and inexact Newton descent, the overall computational complexity in terms of the first-order oracle is (up to some log factors[18]):

$$k \leq O\big(\varepsilon_g^{-7/4}\big),$$

and the scheme guarantees to converge to a point $\boldsymbol{x}_\star$ that satisfies:

$$\|\nabla f(\boldsymbol{x}_\star)\|_2 \leq O(\varepsilon^{2/3}), \quad -\lambda_{\min}\left(\nabla^2 f(\boldsymbol{x}_\star)\right) \leq O(\varepsilon^{1/3}). \tag{9.4.27}$$

Compared to the vanilla gradient descent scheme introduced in Section 9.1.1, the above method not only has lower complexity in terms of first-order oracle, $O\big(\varepsilon_g^{-7/4}\big)$ versus $O\big(\varepsilon_g^{-2}\big)$, but also converges to a second-order stationary point.

## 9.5 Gradient Descent with Small Random Noise

As we have mentioned before, when the dimension is very large, it can be very costly to compute second-order information. Hence for scalable implementation

[18] such as $\log(n)$ in the Lanczos method or $\log(\frac{1}{\varepsilon_H})$ in the conjugate gradient.

in practice, one may be restricted to have access only to the gradient information. However, it is well known that in the worst case gradient descent alone can be very ineffective with minimizing nonconvex functions. It can be extremely slow to escape saddle points,[19] unless we utilize schemes introduced in Sections 9.3 – 9.4 which explicitly exploit negative curvature computed from evaluating extra number of gradients.

Historically, to avoid spurious critical points, people have also found that it is beneficial to introduce some *random noise* in the descent process. Conceptually, the random noise allows the algorithm to search a broader local landscape of the objective function and creates a fair chance to escape from unstable critical points[20], or even to escape local minima (at least asymptotically, as we will soon see).

This section studies the role of random noise in nonconvex optimization and develops gradient descent type algorithms with convergence guarantees to global (asymptotically) or local minimizers. In other words, we assume the algorithms only have access to *the noisy gradient oracle*:

$$\text{the gradient } \nabla f(\boldsymbol{x}) \text{ and small random noise } \boldsymbol{n}.$$

We will reveal that gradient descent with random noise is actually *implicitly* computing the second-order information and exploiting the direction of negative curvature to achieve adequate local descent. In particular, for converging to second-order stationary points, the best achievable complexity (in the first-order oracle) is, not surprisingly, the same as the best methods introduced in the previous section.

### 9.5.1 Diffusion Process and Laplace's Method

To understand the role of random noise, it is the clearest to examine the continuous dynamics of the state $\boldsymbol{x}$ under the gradient flow with random noise (e.g. see [Sas83]). Given a nonconvex function $f(\boldsymbol{x})$, consider the following dynamics with noisy gradient flow:

$$\dot{\boldsymbol{x}}(t) = -\frac{1}{2}\nabla f(\boldsymbol{x}(t)) + \sqrt{\lambda}\boldsymbol{n}(t), \tag{9.5.1}$$

where $\lambda > 0$ and $\boldsymbol{n} \in \mathbb{R}^n$ is a white noise process. This is also known as the *diffusion process*, or continuous-time *Langevin dynamics*. It is known from stochastic process that given the derivative $\nabla f(\boldsymbol{x})$ grows rapidly enough as $\|\boldsymbol{x}\| \to \infty$,[21] then the probability density of this diffusion process of the state $\boldsymbol{x}$ converges

[19] even when the saddle points are not so flat or are non-degenerate [DJL+17].

[20] As we have seen in the power iteration and Lanczos method in Section 9.3.2, random initialization also helps avoid certain (zero measure) pathological cases with high probability.

[21] For instance, it suffices for the function $f(\boldsymbol{x})$ to grow like quadratic as $\|\boldsymbol{x}\| \to \infty$.