Lecture 4 Gradient Descent, Regularization and Momentum

Key Concepts/Topics of This Lecture
gradient descent $\mathbf{x} \longleftrightarrow \mathbf{x} - \epsilon \mathbf{g}$ for optimizing loss function $J(\mathbf{x})$
exact-line search $\epsilon^* = \mathbf{g}^\top \mathbf{g} / \mathbf{g}^\top \mathbf{H} \mathbf{g} \to \epsilon^* = 1/f''$ for 1D case
approximation $\mathbf{H}\vec{\phi} \approx (2\Delta)^{-1}[\mathbf{g}(\mathbf{x}+\Delta\vec{\phi}) - \mathbf{g}(\mathbf{x}-\Delta\vec{\phi})]$ for $\vec{\phi} = \mathbf{g}$
singular behavior of Hessian matrix $\det \mathbf{H} \approx 0$, $\mathbf{H} + \lambda \vec{1} \to \mathbf{H}$
momentum mechanism: $\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)} = -\epsilon \nabla f(\mathbf{x}^{(i)}) + p(\mathbf{x}^{(i)} - \mathbf{x}^{(i-1)})$ with $-1 < p < 1$
extensions of gradient descent: Adam, AdaGrad, RMSProp, $\cdots$

§4.1 General Discussion

In these lectures, we discuss how to find the minimum of an objective function using the information contained within the function itself [*4-1*]. A common optimization strategy is to iteratively improve a design point $\mathbf{x}$ by taking steps that reduce the objective function based on a local model. Such a model can be obtained, for example, from a first- or second-order Taylor expansion, and algorithms following this framework are known as descent direction methods.

An optimization algorithm starts from an initial value $\mathbf{x}^{(0)}$ and generates a sequence of points (iterations) that converge to a local minimum. The iterative descent direction procedure consists of several steps: (a) Check whether $\mathbf{x}^{(i)}$ satisfies the termination conditions. If so, terminate; otherwise, proceed to the next step. (b) Determine the descent direction $\mathbf{d}^{(i)}$ using local information such as the gradient $\mathbf{g}$ or the Hessian matrix $\mathbf{H}$ of the objective function. (c) Choose the step size (learning rate) ϵ_i. (d) Update the design point according to

$$\boxed{\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i)} + \epsilon_i \mathbf{d}^{(i)}.} \tag{4.1}$$

Different algorithms adopt different strategies for selecting ϵ and $\mathbf{d}$. In the line search approach, one chooses the learning rate and direction by minimizing the one-dimensional function $\min_{\epsilon,\mathbf{d}} f(\mathbf{x} + \epsilon\mathbf{d})$. Some methods use a fixed learning rate. Larger learning rates tend to yield faster convergence but may overshoot the minimum, whereas smaller learning rates are typically more stable but can lead to slower convergence. We aim to determine the direction $\mathbf{d}$ along which the decrease in the objective function is maximized.

Our discussion will focus primarily on convex functions $f(\mathbf{x})$, see FIG. 3.2 and relation (3.9). A unimodal function $f(\mathbf{x})$ is defined by the existence of a unique $\mathbf{x}^*$ such that $f(\mathbf{x})$ decreases monotonically for $\mathbf{x} \le \mathbf{x}^*$ and increases monotonically for $\mathbf{x} \ge \mathbf{x}^*$. From this definition, it follows that the unique global minimum occurs at $\mathbf{x}^*$, with no other local minima; see the left and middle panels of FIG. 4.1 for examples.

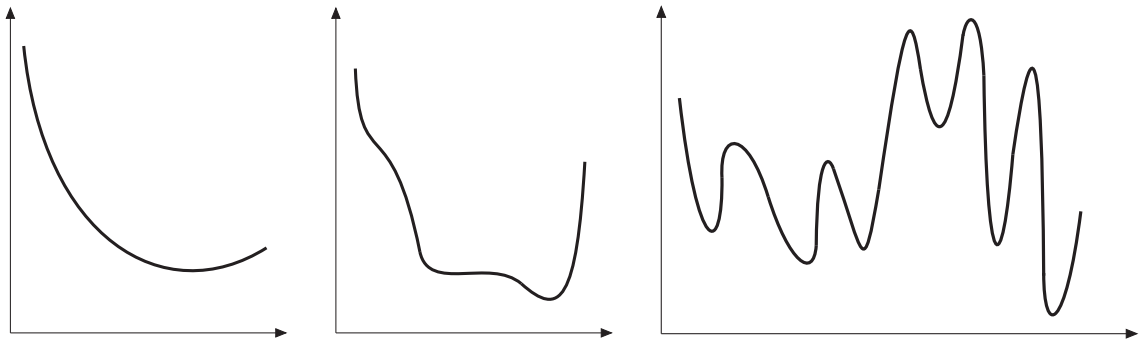

FIG. 4.1: Convex and unimodal (left), unimodal but nonconvex (middle), and a function with many local minima but a single global minimum (right).

It is a challenging problem in machine learning to locate the global minimum when multiple local minima are present, as illustrated in the right panel of FIG. 4.1.

§4.2 Fastest Descent Direction

In order to determine the direction of fastest descent, we evaluate the change of the function along a direction $\mathbf{d}$,

$$\left.\frac{\partial f(\mathbf{x}+\epsilon\mathbf{d})}{\partial \epsilon}\right|_{\epsilon=0} = \left.\frac{\partial f(\mathbf{x}+\epsilon\mathbf{d})}{\partial(\mathbf{x}+\epsilon\mathbf{d})}\right|_{\epsilon=0} \cdot \left.\frac{\partial(\mathbf{x}+\epsilon\mathbf{d})}{\partial \epsilon}\right|_{\epsilon=0} = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{d}, \tag{4.2}$$

Next, assuming without loss of generality that the direction $\mathbf{d}$ has unit length, the problem reduces to finding the direction that minimizes the decrease, i.e.,

$$\boxed{\min_{\mathbf{d}}\left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\cdot\mathbf{d}\right) = \min_{\mathbf{d}}\left[|\mathbf{d}|_2 \cdot \left|\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right|_2 \cos\left(\mathbf{d}, \widetilde{\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}}\right)\right] = \min_{\mathbf{d}}\left[\left|\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right|_2 \cos\left(\mathbf{d}, \widetilde{\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}}\right)\right],} \tag{4.3}$$

where

$$\Theta = \left(\mathbf{d}, \widetilde{\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}}\right) \tag{4.4}$$

is the angle between $\mathbf{d}$ and the gradient $\mathbf{g} = \nabla_{\mathbf{x}} f$.

Since the first term in the bracket is independent of $\mathbf{d}$, the above optimization problem reduces to

$$\min_{\mathbf{d}}\left[\cos\left(\mathbf{d}, \widetilde{\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}}\right)\right], \tag{4.5}$$

whose solution is

$$\boxed{\cos\left(\mathbf{d}, \widetilde{\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}}\right) = -1 \rightarrow \Theta = \left(\mathbf{d}, \widetilde{\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}}\right) = \pi,} \tag{4.6}$$

i.e., the function $f(\mathbf{x})$ decreases most rapidly along the negative gradient direction. See FIG. 4.2 for the geometric meaning of the negative gradient. Along the negative gradient direction, the function

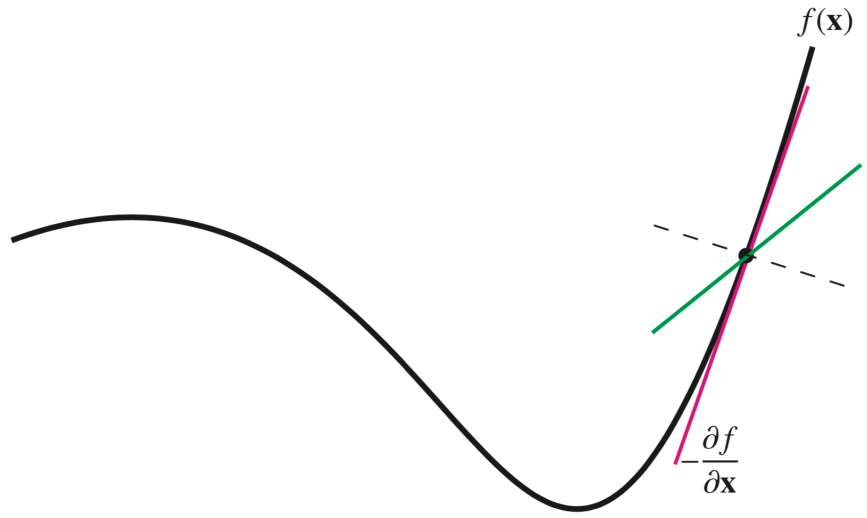

FIG. 4.2: Geometric meaning of the negative gradient.

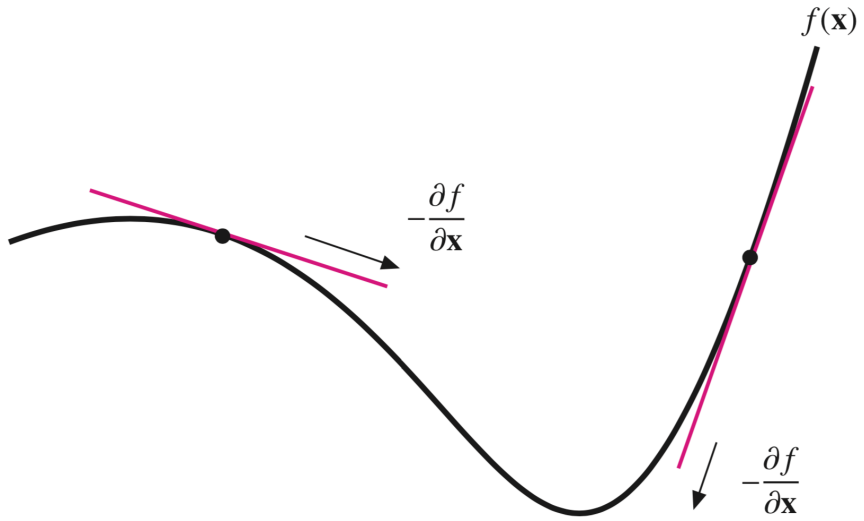

FIG. 4.3: Along the negative gradient. the function $f(\mathbf{x})$ always decreases.

$f(\mathbf{x})$ will always decrease, see FIG. 4.3. For example, in the right region the gradient is positive, so the negative gradient points in the negative direction, along which the function decreases (i.e., toward the left). Conversely, in the left region the gradient is negative, so the negative gradient is positive, indicating that the design point moves to the right along this direction.

Moreover, everyday intuition suggests that moving along the negative gradient corresponds to descending most efficiently, as illustrated in FIG. 4.4. Mathematically, a minimum of the function is characterized by a vanishing gradient and positive curvature, i.e.,

$$\boxed{\mathbf{g}(\mathbf{x}^*)=\vec{\mathbf{0}},\ \mathbf{x}^\top\mathbf{H}(\mathbf{x}^*)\mathbf{x}>0,\ \mathbf{g}(\mathbf{x})=\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}},\ \mathbf{H}(\mathbf{x})=\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}\partial \mathbf{x}^\top}.} \tag{4.7}$$

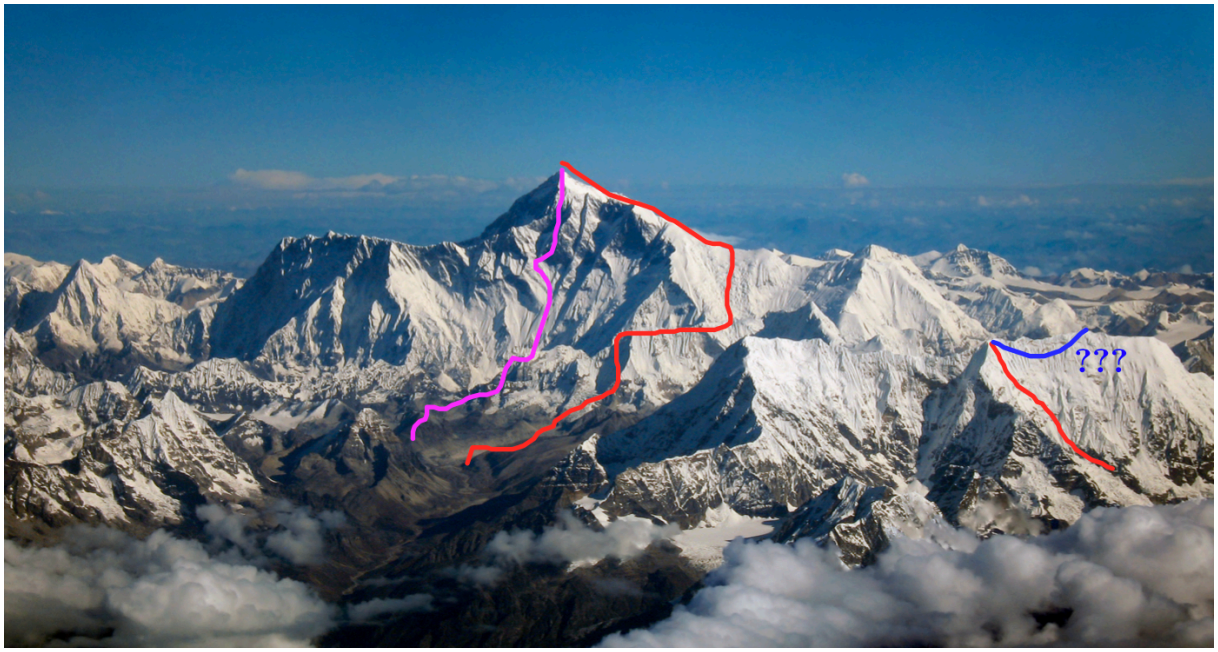

FIG. 4.4: Downhill along the negative gradient is the fastest direction.

However, solving $\mathbf{g}(\mathbf{x}) = \vec{0}$ can be difficult in practice. Therefore, iterative algorithms are developed to approximate $\mathbf{x}^*$. Here, "iterative" means starting from an initial guess $\mathbf{x}^{(0)}$ and successively refining it to obtain better approximations. The process continues until two consecutive iterates $\mathbf{x}^{(i+1)}$ and $\mathbf{x}^{(i)}$ are sufficiently close, for example, $|\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}| < 10^{-4}$. At that point, $\mathbf{x}^{(i+1)}$ can be regarded as an approximate optimal solution $\mathbf{x}^*$.

The condition $\mathbf{x}^\top \mathbf{H}(\mathbf{x}^*)\mathbf{x} > 0$ generalizes the one-dimensional condition $ax^2 > 0$. Moreover, if $\mathbf{x} \in \mathrm{R}^d$, then

$$\mathbf{x}^\top \mathbf{H}(\mathbf{x}^*)\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}^\top \begin{pmatrix} H_{11} & \cdots & H_{1d} \\ \vdots & \ddots & \vdots \\ H_{d1} & \cdots & H_{dd} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}. \tag{4.8}$$

For $d = 2$, this reduces to

$$H_{11}x_1^2 + 2H_{12}x_1x_2 + H_{22}x_2^2. \tag{4.9}$$

Consequently, one can show that the positive-definite property of the Hessian matrix is equivalent to $\det \mathbf{H} > 0$, i.e.,

$$H_{11}H_{22} - H_{12}^2 > 0. \tag{4.10}$$

This result can be straightforwardly generalized to higher dimensions.

EXERCISE 4-1. Write the condition of positiveness for $d = 3$.

§4.3 Learning Rate Design

After selecting the negative gradient as the optimal direction, the gradient descent algorithm is obtained as

$$\boxed{\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \epsilon_i \mathbf{g}^{(i)} = \mathbf{x}^{(i)} - \epsilon_i \nabla f(\mathbf{x}^{(i)}) = \mathbf{x}^{(i)} - \epsilon_i \left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^{(i)}}}. \tag{4.11}$$

The learning rate ϵ_i at each step is a manually chosen parameter. If the learning rate is too large, the iterative procedure may become unstable (left panel of FIG. 4.5), and may even lead to oscillations, see FIG. 4.6. On the other hand, if the learning rate is too small, the algorithm may require a long time to reach the minimum (right panel of FIG. 4.5).

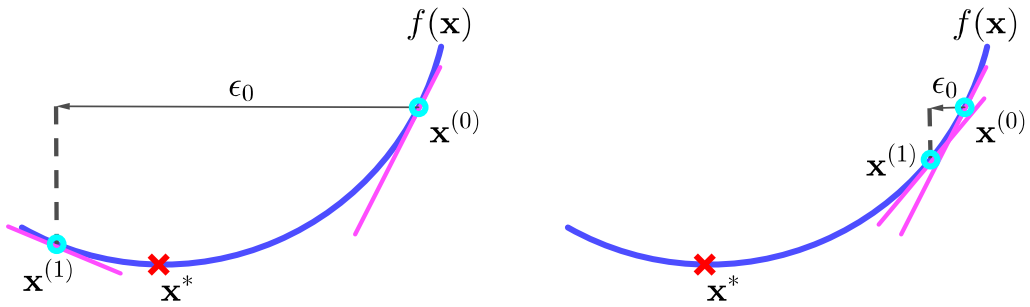

FIG. 4.5: Too large or too small learning rate causes problems.

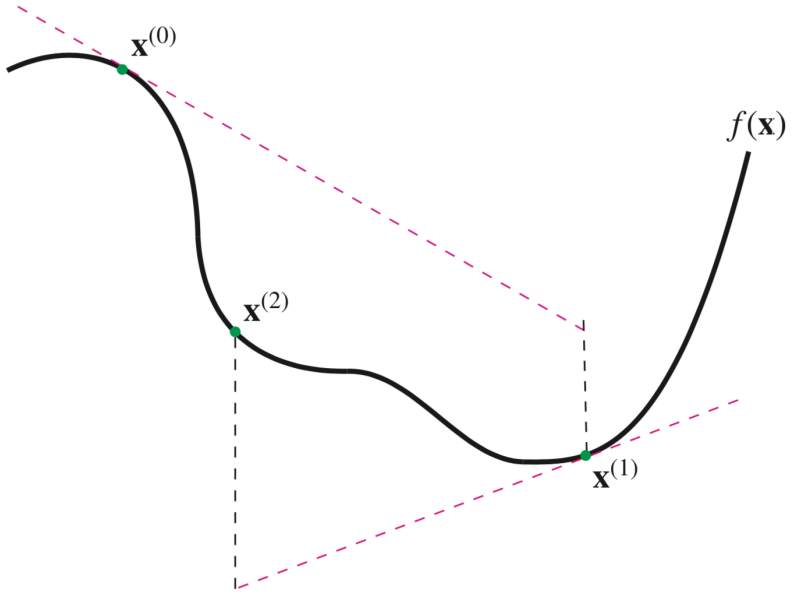

FIG. 4.6: Large learning rate leads to oscillation phenomenon.

Geometrically, the gradient is typically large in the early stages and approaches zero near the minimum. Therefore, it is desirable for the learning rate to be large in the early iterations (to avoid instability) and small near the minimum. One possible choice is to use a decaying rule. Another

commonly used method is gradient descent with exact-line search. Expanding the function $K(\epsilon) \equiv f(\mathbf{x}-\epsilon\mathbf{g})$ of ϵ using Taylor expansion,

$$K(\epsilon) = f(\mathbf{x}-\epsilon\mathbf{g}) \approx f(\mathbf{x}) - \epsilon\mathbf{g}^\top\mathbf{g} + \frac{1}{2}\epsilon^2\mathbf{g}^\top\mathbf{H}\mathbf{g}. \tag{4.12}$$

Treating the right-hand side as a function of ϵ, and minimizing it (i.e., achieving the maximum decrease of f), we obtain the following scheme for determining the learning rate,

$$\epsilon^* = \underset{\epsilon}{\operatorname{argmin}}\left[f(\mathbf{x}) - \epsilon\mathbf{g}^\top\mathbf{g} + \frac{1}{2}\epsilon^2\mathbf{g}^\top\mathbf{H}\mathbf{g}\right]. \tag{4.13}$$

The result is

$$\boxed{\epsilon^* = \frac{\mathbf{g}^\top\mathbf{g}}{\mathbf{g}^\top\mathbf{H}\mathbf{g}}.} \tag{4.14}$$

The gradient descent algorithm with exact-line search is thus given by

$$\boxed{\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \epsilon_i\mathbf{g}_i, \epsilon_i = \frac{\mathbf{g}_i^\top\mathbf{g}_i}{\mathbf{g}_i^\top\mathbf{H}_i\mathbf{g}_i}, \mathbf{g}_i = \mathbf{g}(\mathbf{x}^{(i)}),\ \mathbf{H}_i = \mathbf{H}(\mathbf{x}^{(i)}).} \tag{4.15}$$

In this algorithm, the learning rate is no longer a user-defined parameter.

EXERCISE 4-2. For 1-dimensional optimization problem, write out the gradient descent algorithm with exact-line search.

§4.4 Numerical Examples

As the first example, we show in the left panel of FIG. 4.7 the evolution of the optimal x for the functione $f(x) = (x-1)^4$ under three algorithms, namely the conventional gradient descent (GD) with a user-chosen learning rate $\epsilon = 0.5$, the GD with two decaying learning rates $\gamma = 0.8$ (cyan) and $\gamma = 1.6$ (red), and the GD with an exact-line search mechanism. All searches start from the initial value $x^{(0)} = 1.2$. For reference, we write the explicit update expressions for the optimal x as

$$\text{GD}: \ x^{(i+1)} = x^{(i)} - 4\epsilon_i\left(x^{(i)}-1\right)^3, \tag{4.16}$$

$$\text{GD with exact-line search}: \ x^{(i+1)} = \frac{2}{3}x^{(i)} + \frac{1}{3}. \tag{4.17}$$

It is interesting to observe that for the function $f(x) = (x-1)^4$, increasing the decaying constant γ from 0.8 ("decaying") to 1.6 ("enhancing") leads to faster convergence, indicating that the initial learning rate $\epsilon_0 = \epsilon = 0.5$ is somewhat small in this example. Another important feature is that in gradient descent with an exact-line search mechanism, no user-defined learning rate appears, which is intrinsic to the algorithm design; in this case, the learning rate is constructed using the first- and second-order derivatives of the objective function. For complicated objective functions, it is generally difficult, or even unnecessary, to derive explicit expressions for the gradient and/or the second-order derivative. However, algorithms (4.11) and (4.15) still provide sufficient ingredients to search for the minimum.

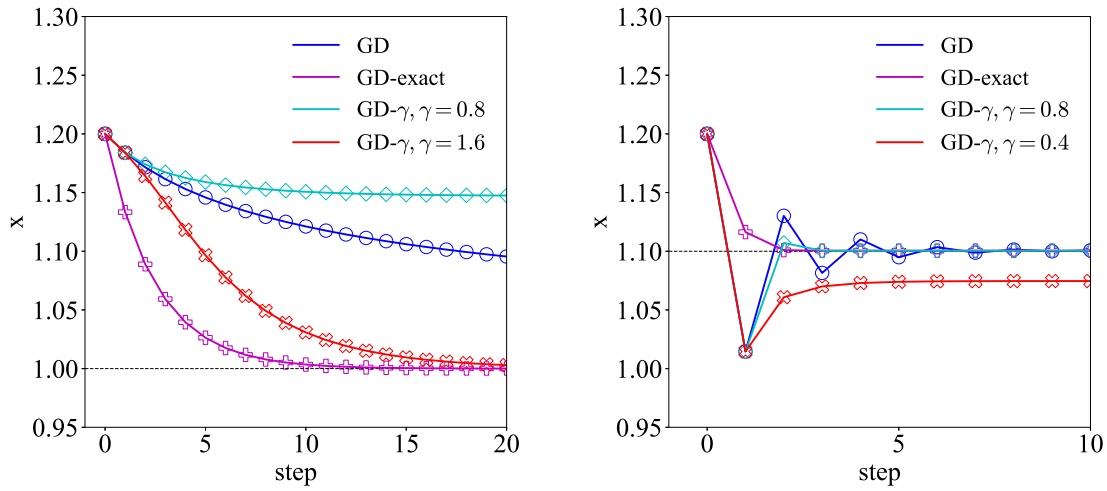

FIG. 4.7: Evolution of the optimal value x for $f(x)=(x-1)^4$ (upper) and $f(x)=e^{x^2-2}-x$ (lower).

Specifically, the gradient $\mathbf{g}$ can be expanded around a point $\mathbf{x}$ along a direction $\vec{\phi}$ as

$$\mathbf{g}(\mathbf{x}+\Delta\vec{\phi})\approx\mathbf{g}(\mathbf{x})+\Delta\frac{\partial\mathbf{g}(\mathbf{x})}{\partial\mathbf{x}^\top}\vec{\phi}+\mathcal{O}(\Delta^2), \tag{4.18}$$

where Δ is a small quantity and $\partial\mathbf{g}(\mathbf{x})/\partial\mathbf{x}^\top$ is the Hessian matrix. Thus, $\mathbf{g}(\mathbf{x}+\Delta\vec{\phi})\approx\mathbf{g}(\mathbf{x})+\Delta\mathbf{H}\vec{\phi}+\mathcal{O}(\Delta^2)$. Using $-\Delta$ to replace Δ and subtracting $\mathbf{g}(\mathbf{x}+\Delta\vec{\phi})$ and $\mathbf{g}(\mathbf{x}-\Delta\vec{\phi})$, one obtains the following expression for evaluating the term $\mathbf{H}\vec{\phi}$ appearing in gradient descent with exact-line search,

$$\boxed{\mathbf{H}\vec{\phi}\approx\frac{\mathbf{g}(\mathbf{x}+\Delta\vec{\phi})-\mathbf{g}(\mathbf{x}-\Delta\vec{\phi})}{2\Delta}+\mathcal{O}(\Delta^2),} \tag{4.19}$$

by taking $\vec{\phi}=\mathbf{g}$. In this sense, computing $\mathbf{H}$, or more precisely $\mathbf{Hg}$, is not difficult; instead, computing the inverse of the Hessian $\mathbf{H}$ is generally difficult.

Similarly, shown in the right panel of FIG. 4.7 is the evolution of the optimal x for the function $f(x)=e^{x^2-2}-x$, where the update expressions are given by

$$\text{GD}:\ x^{(i+1)}=x^{(i)}-\epsilon_i\left[2x^{(i)}\exp\left(x^{(i),2}-2\right)-1\right], \tag{4.20}$$

$$\text{GD with exact-line search}:\ x^{(i+1)}=x^{(i)}-\frac{1}{2}\frac{1}{2x^{(i),2}+1}\left[2x^{(i)}-\exp\left(2-x^{(i),2}\right)\right]. \tag{4.21}$$

The optimal value is located at $x^*\approx 1.1$, which can be obtained recursively via $x\leftarrow[2+\ln(1/2x)]^{1/2}$. As a reasonable decaying constant γ is generally useful for faster convergence, a small γ may even lead the algorithm to an incorrect optimal value.

The Rosenbrock function is a frequently used two-dimensional benchmark function in optimization, defined as

$$f(\mathbf{x})=(x-a)^2+b(x^2-y)^2,\ \ b>0. \tag{4.22}$$

Its global minimum is located at $(a,a^2)^\top$, independent of the value of b. In FIG. 4.8, we show the trajectory of the two-dimensional optimization problem for the Rosenbrock function with $a=1, b=5$,

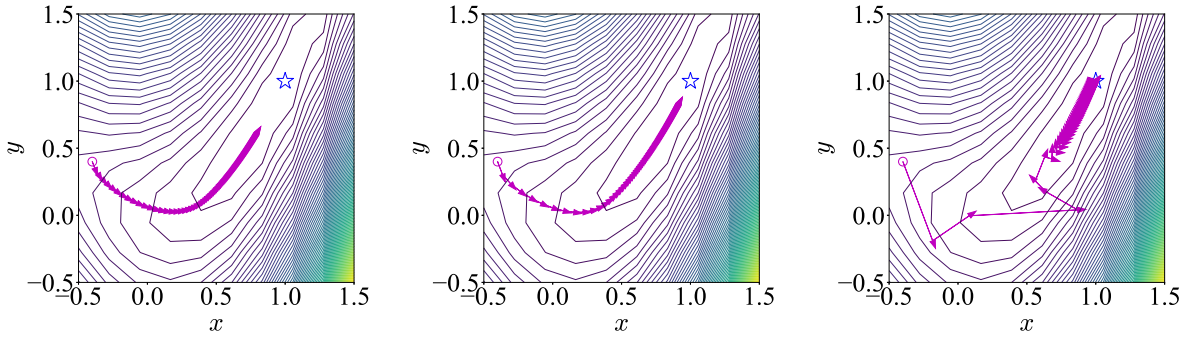

FIG. 4.8: Trajectory for the Rosenbrock function with $a = 1, b = 5$ for 150 steps under the gradient descent algorithm.

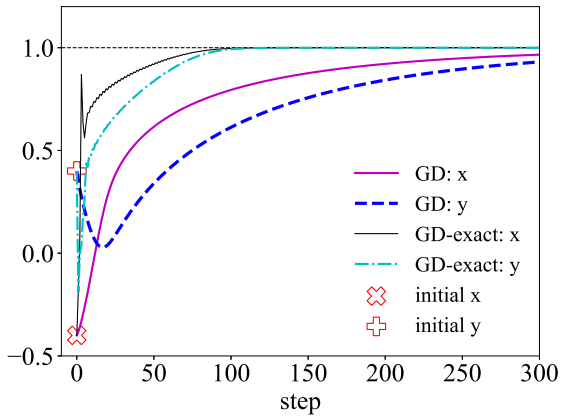

FIG. 4.9: Evolution of the optimal values of x and y for the Rosenbrock function with $a = 1$ and $b = 5$ under gradient descent with and without exact-line search.

starting from the initial point $\mathbf{x} = (-0.4, 0.4)^\top$ for 150 steps. In the middle panel of FIG. 4.8, a decaying learning parameter $\gamma = 0.999$ is introduced compared with the one shown in the left panel. In addition, the values of ϵ_0 in the left panel ($\epsilon_0 = 0.04$) and middle panel ($\epsilon_0 = 0.02$) are slightly different. As can be clearly seen, the trajectory shown in the right panel exhibits new features: at each step, the displacement is irregular, sometimes large and sometimes small, fully determined by the gradient and Hessian information at each iteration. The trajectories shown in the left and middle panels are very similar. Moreover, the termination condition is set as $|x^{(i+1)} - x^{(i)}| < 10^{-4}$ and $|y^{(i+1)} - y^{(i)}| < 10^{-4}$. Consequently, the termination step for standard gradient descent is about 1042, while that for gradient descent with a decaying parameter is about 734, corresponding to an improvement of roughly 30%. This indicates that tuning the learning rate ϵ_0 and the decaying constant γ can effectively improve convergence behavior, although this strongly depends on experience. In contrast, gradient descent with exact-line search requires about 138 steps to reach the termination condition, an improvement of about 87% compared to standard gradient descent. Most importantly, there are no user-defined parameters, and all model parameters are self-consistently determined. See FIG. 4.9 for the evolution of x and y under gradient descent with and without exact-line search. The parameter b in the Rosenbrock function controls the relative importance of the term $b(x^2 - y)^2$ in the total function, although it does not affect the location of the minimum. As b increases, the number of

iterations required for convergence may also increase, indicating that the Rosenbrock function is an ideal benchmark for testing optimization algorithms.

EXERCISE 4-3. Write down the expression for $\mathbf{H}\vec{\phi}$ using the five-point algorithm for derivatives.

EXERCISE 4-4. Find minimum of the 2-dimensional function $f(\mathbf{x}) = x^2 + y^2/10$ from $\mathbf{x}^{(0)} = (1,2)^\top$.

§4.5 Singular Behavior of Hessian Matrix

In the left panel of FIG. 4.10, we show the results obtained from gradient descent and gradient descent with exact-line search for the function $f(\mathbf{x}) = \exp[-(x\,y-a)^2-(y-a)^2]$, where $a = 3/2$. The learning rate is set as $\epsilon = \epsilon_0 = 0.4$ with the decaying constant $\gamma = 0.95$. Both the trajectory obtained from gradient descent (magenta) and that from exact-line search (red) exhibit a "zig-zag" pattern, which is a general feature of gradient-based algorithms. The exact optimal value of the objective function is located at $(1,3/2)^\top$.

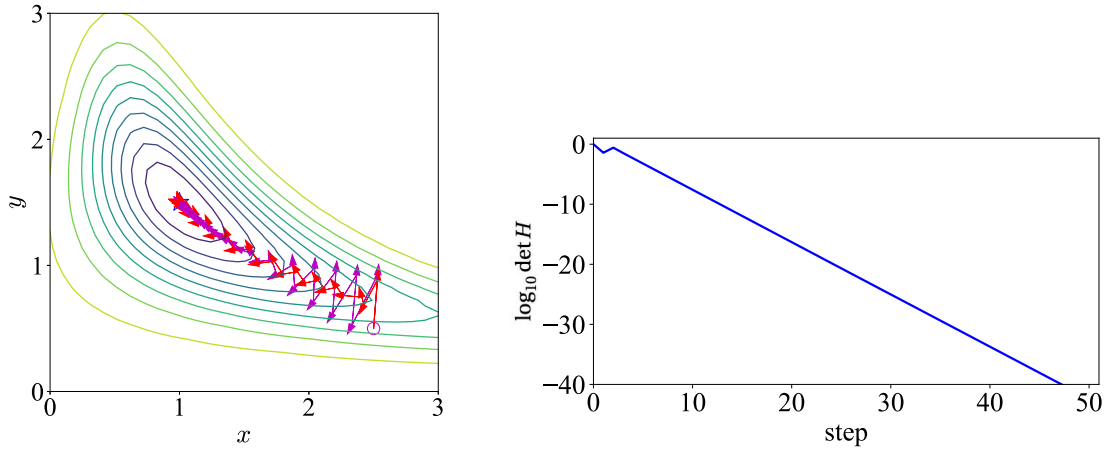

FIG. 4.10: Left: trajectory of the optimal $\mathbf{x}$ for $f(\mathbf{x}) = \exp[-(x\,y-a)^2-(y-a)^2]$, where the red line corresponds to gradient descent with exact-line search; right: determinant $\det\mathbf{H}$ of the function.

Furthermore, the red trajectory is obtained by modifying the Hessian matrix as $\mathbf{H}+\lambda\vec{1} \to \mathbf{H}$, where $\lambda = 1.5$, since the original Hessian matrix tends to vanish rapidly after a few iterations; see the right panel of FIG. 4.10 for $\det\mathbf{H}$ of the function $f(\mathbf{x}) = \exp[-(x\,y-a)^2-(y-a)^2]$. Specifically, each component of the gradient $\mathbf{g} = (\partial f/\partial x, \partial f/\partial y)^\top$ contains a factor f due to differentiation with respect to x or y. A similar structure holds for the Hessian components. Consequently, the learning rate from exact-line search takes the form

$$\begin{aligned}\epsilon^* &= \left[\left(\frac{\partial f}{\partial x}\right)^2+\left(\frac{\partial f}{\partial y}\right)^2\right]\Big/\left[\frac{\partial^2 f}{\partial x^2}\left(\frac{\partial f}{\partial x}\right)^2+2\frac{\partial^2 f}{\partial x\partial y}\left(\frac{\partial f}{\partial x}\right)\left(\frac{\partial f}{\partial y}\right)+\frac{\partial^2 f}{\partial y^2}\left(\frac{\partial f}{\partial y}\right)^2\right] \\ &\sim \mathcal{P}(x,y)/f(x,y),\end{aligned} \tag{4.23}$$

where $\mathcal{P}(x,y)$ denotes a polynomial in x and y. According to the complexity of different functions, the growth of polynomials is slower than that of exponentials, implying that after a few iterations the denominator $f(x,y)$ quickly approaches zero, as illustrated in the right panel of FIG. 4.10. From the expression for the determinant $\det\mathbf{H}$, one can also see that the positive definiteness of $\mathbf{H}$ is equivalent

to

$$\frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2}-\left(\frac{\partial^2 f}{\partial x\partial y}\right)^2>0, \tag{4.24}$$

by interpreting $\det\mathbf{H}$ as a quadratic form in $\partial f/\partial x$ and $\partial f/\partial y$, with coefficients given by $\partial^2 f/\partial x^2$, $2\partial^2 f/\partial x\partial y$, and $\partial^2 f/\partial y^2$.

The term $\lambda\vec{1}$ is referred to as a regularization term, and when λ is sufficiently large such that it dominates over $\mathbf{H}$ (which has a very small determinant), then λ effectively plays the role of the inverse learning rate,

$$\boxed{\epsilon^*=\frac{\mathbf{g}^\top\mathbf{g}}{\mathbf{g}^\top\mathbf{H}\mathbf{g}}\to\frac{\mathbf{g}^\top\mathbf{g}}{\mathbf{g}^\top(\mathbf{H}+\lambda\vec{1})\mathbf{g}}\approx\frac{\mathbf{g}^\top\mathbf{g}}{\mathbf{g}^\top\lambda\vec{1}\mathbf{g}}=\frac{1}{\lambda}.} \tag{4.25}$$

Consequently, gradient descent with exact-line search becomes equivalent to conventional gradient descent. The parameter λ is also known under different names, such as a regularization parameter, penalty parameter, or trust-region parameter.

EXERCISE 4-5. Write out the expressions for the gradient and the Hessian matrix of the Rosenbrock function, and those of $f(\mathbf{x})=\exp[-(xy-a)^2-(y-a)^2]$.

§4.6 Concept of Momentum

We introduce the concept of "momentum" originated from physics into the gradient descent algorithm. To begin, we use the eigenvalue decomposition of a matrix as a tool to analyze the convergence properties of gradient descent and, in parallel, introduce the notion of the condition number κ, which determines the convergence speed of the algorithm. More detailed discussions will be provided in Lecture 7 and Lecture 8.

For simplicity, we assume that the objective function takes the form

$$f(\mathbf{x})=2^{-1}\mathbf{x}^\top\mathbf{A}\mathbf{x}-\mathbf{b}^\top\mathbf{x}, \tag{4.26}$$

where $\mathbf{A}$ is a symmetric and positive semi-definite matrix. Let the optimal solution be denoted by $\mathbf{x}^*$, which satisfies the equation $\mathbf{A}\mathbf{x}=\mathbf{b}$. Since $\mathbf{A}$ is symmetric and positive semi-definite, we can perform an eigen-decomposition to write $\mathbf{A}=\mathbf{Q}^\top\vec{\Lambda}\mathbf{Q}$, where $\mathbf{Q}$ is an orthogonal matrix satisfying $\mathbf{Q}^\top\mathbf{Q}=\vec{1}$, and $\vec{\Lambda}$ is a diagonal matrix whose entries are the eigenvalues of $\mathbf{A}$. We then introduce a new variable $\mathbf{w}^{(i)}$ defined by $\mathbf{w}^{(i)}=\mathbf{Q}(\mathbf{x}^{(i)}-\mathbf{x}^*)$. Consequently, the update rule for the transformed variable becomes (noting that the gradient of the objective function is $\mathbf{g}=\mathbf{A}\mathbf{x}-\mathbf{b}$),

$$\begin{aligned}\mathbf{w}^{(i+1)}=&\mathbf{Q}\left(\mathbf{x}^{(i+1)}-\mathbf{x}^*\right)=\mathbf{Q}\left(\mathbf{x}^{(i)}-\epsilon\left(\mathbf{A}\mathbf{x}^{(i)}-\mathbf{b}\right)-\mathbf{x}^*\right)=\mathbf{w}^{(i)}-\epsilon\mathbf{Q}\left(\mathbf{A}\mathbf{x}^{(i)}-\mathbf{b}\right)\\=&\mathbf{w}^{(i)}-\epsilon\mathbf{Q}\left[\mathbf{A}\left(\mathbf{Q}^\top\mathbf{w}^{(i)}+\mathbf{x}^*\right)-\mathbf{b}\right]=\mathbf{w}^{(i)}-\epsilon\mathbf{Q}\left[\mathbf{A}\mathbf{Q}^\top\mathbf{w}^{(i)}+\mathbf{A}\mathbf{x}^*-\mathbf{b}\right]\\=&\mathbf{w}^{(i)}-\epsilon\mathbf{Q}\left(\mathbf{A}\mathbf{Q}^\top\mathbf{w}^{(i)}\right)=\mathbf{w}^{(i)}-\epsilon\vec{\Lambda}\mathbf{w}^{(i)}=\left(\vec{1}-\epsilon\vec{\Lambda}\right)\mathbf{w}^{(i)},\end{aligned} \tag{4.27}$$

or in component form,

$$w_j^{(i+1)}=\left(1-\epsilon\lambda_j\right)w_j^{(i)}, \tag{4.28}$$

where "j" is the dimensional index (ranging from 1 to d).

Substituting back into the original variable yields

$$\mathbf{x}^{(i)}-\mathbf{x}^*=\left(1-\epsilon\vec{\Lambda}\right)\left(\mathbf{x}^{(i-1)}-\mathbf{x}^*\right)=\cdots=\left(1-\epsilon\vec{\Lambda}\right)^i\left(\mathbf{x}^{(0)}-\mathbf{x}^*\right)=\cdots$$

$$=\left(1-\epsilon\vec{\Lambda}\right)^i \mathbf{Q}^\top \mathbf{w}^{(0)} = \sum_{j=1}^{d} w_j^{(0)} \left(1-\epsilon\lambda_j\right)^i \mathbf{q}_j, \tag{4.29}$$

where $\mathbf{q}_j$ denotes the j-th column of $\mathbf{Q}$. The above equation admits a simple interpretation: each component of $\mathbf{w}^{(0)}$ represents the projection of the initial error onto the $\mathbf{Q}$-basis. There are d such error components, and each evolves independently toward the minimum, decaying exponentially at a rate of $1-\epsilon\lambda_j$. The closer this factor is to 1, the slower the convergence. This analysis immediately provides guidance on choosing the step size ϵ. For convergence, each $|1-\epsilon\lambda_j|$ must be strictly less than 1, so all valid step sizes lie in the interval

$$\boxed{0<\epsilon\lambda_j<2.} \tag{4.30}$$

The overall convergence rate is governed by the slowest-decaying component, corresponding to either λ_1 or λ_d (assuming the ordering $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d > 0$). We define the rate function $R(\epsilon)$ as

$$R(\epsilon) = \max_j \left[|1-\epsilon\lambda_j|\right] = \max[|1-\epsilon\lambda_1|, |1-\epsilon\lambda_d|]. \tag{4.31}$$

This rate is minimized when the contributions from λ_1 and λ_d are equal. Solving this yields

$$\boxed{\epsilon^* = \operatorname*{argmin}_\epsilon R(\epsilon) = \frac{2}{\lambda_1+\lambda_d},\ R^* \equiv R(\epsilon^*) = \frac{\lambda_1/\lambda_d-1}{\lambda_1/\lambda_d+1} = \frac{\kappa(\mathbf{A})-1}{\kappa(\mathbf{A})+1}.} \tag{4.32}$$

In this context, the condition number $\kappa(\mathbf{A})$ measures how difficult gradient descent is in practice. When $\kappa(\mathbf{A})=1$, convergence occurs in a single step (in this trivial case).

We now introduce the concept of "momentum" originated from physics to the gradient descent algorithm [*4-2*], starting from a physical viewpoint. The gradient descent algorithm

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \epsilon\mathbf{g}_i = \mathbf{x}^{(i)} - \epsilon\nabla f(\mathbf{x}^{(i)}) \tag{4.33}$$

can be written in continuous form as

$$\mathrm{d}\mathbf{x}/\mathrm{d}t = -\epsilon\nabla f(\mathbf{x}), \tag{4.34}$$

where $\mathbf{x}$ is now treated as a continuous function of time t rather than a discrete sequence. Comparing this equation with the Newtonian dynamics of a particle of mass m moving in a viscous medium with damping coefficient μ under a conservative force field with the "potential energy $f(\mathbf{x})$",

$$m\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2} + \mu\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = -\nabla f(\mathbf{x}), \tag{4.35}$$

it becomes clear that Eq. (4.34) is a special case of Eq. (4.35) corresponding to a massless particle. This analogy motivates us to examine whether the mass term in Eq. (4.35) introduces new effects in gradient descent. We discretize the derivatives as $\mathrm{d}^2\mathbf{x}/\mathrm{d}t^2 \approx [\mathbf{x}(t+\delta t)+\mathbf{x}(t-\delta t)-2\mathbf{x}(t)]/\delta t^2$ and $\mathrm{d}\mathbf{x}/\mathrm{d}t \approx [\mathbf{x}(t+\delta t)-\mathbf{x}(t)]/\delta t$, so Eq. (4.35) becomes [*4-3*]

$$\mathbf{x}(t+\delta t)-\mathbf{x}(t) = -\frac{\delta t^2}{m+\mu\delta t}\nabla f(\mathbf{x}) + \frac{m}{m+\mu\delta t}[\mathbf{x}(t)-\mathbf{x}(t-\delta t)], \tag{4.36}$$

after rearrangement. Introducing the parameters

$$\boxed{\epsilon = \frac{\delta t^2}{m + \mu \delta t}, \quad p = \frac{m}{m + \mu \delta t},} \tag{4.37}$$

Eq. (4.36) becomes $\mathbf{x}(t+\delta t) - \mathbf{x}(t) = -\epsilon \nabla f(\mathbf{x}) + p(\mathbf{x}(t) - \mathbf{x}(t-\delta t))$, which in discrete form reads

$$\boxed{\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)} = -\epsilon \nabla f(\mathbf{x}^{(i)}) + p\left(\mathbf{x}^{(i)} - \mathbf{x}^{(i-1)}\right).} \tag{4.38}$$

The second term is new, and p is called the momentum parameter; (4.38) represents gradient descent with momentum [*4-4*].

A natural question is how the momentum parameter p affects gradient descent. To investigate this, we consider the total energy of the particle,

$$\boxed{E_{\text{tot}} = \frac{m}{2}\left(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right)^{\top}\left(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right) + f(\mathbf{x}),} \tag{4.39}$$

where the first term is kinetic energy and the second is potential energy. We expand the potential energy around its equilibrium point $\mathbf{X}$ as

$$\boxed{f(\mathbf{x}) \approx f(\mathbf{X}) + \frac{1}{2}(\mathbf{x}-\mathbf{X})^{\top}\mathbf{H}(\mathbf{x}-\mathbf{X}), \quad H_{ij} = \left.\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right|_{\mathbf{x}=\mathbf{X}},} \tag{4.40}$$

with no linear term since the gradient vanishes at equilibrium. Without loss of generality, we set $\mathbf{X} = \vec{0}$. Since the Hessian is symmetric and positive definite, we diagonalize it as $\mathbf{K} = \operatorname{diag}(k_1, k_2, \cdots, k_d)$. The equations of motion then become

$$\boxed{m\frac{\mathrm{d}^2\mathbf{x}}{\mathrm{d}t^2} + \mu\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} + \mathbf{K}\mathbf{x} = \vec{0}, \text{ or } \frac{\mathrm{d}^2 x_j}{\mathrm{d}t^2} + \frac{\mu}{m}\frac{\mathrm{d}x_j}{\mathrm{d}t} + \frac{k_j x_j}{m} = 0,} \tag{4.41}$$

i.e., a set of uncoupled damped oscillators with the k_j acting as Hooke's constants.

We first examine Eq. (4.41) without momentum, i.e., $m = 0$ (equivalently $p = 0$). The general solution is $x_j(t) = c^{\lambda_{j,0} t}$ with $\lambda_{j,0} = -k_j/\mu$. The subscript "0" indicates the case $m = 0$. In this case, each $x_j(t)$ converges at a rate determined by k_j, and the smallest k_j yields the slowest convergence. Since $\mathbf{x}(t)$ is a linear combination of these modes, overall convergence is limited by the smallest k_j. When $m \neq 0$, the solution becomes $x_j(t) = c_1 e^{\lambda_{j,1} t} + c_2 e^{\lambda_{j,2} t}$, where

$$\boxed{\lambda_{j,1} = -\frac{\mu}{2m} + \sqrt{\frac{\mu}{m}\left(\frac{\mu}{4m} - \frac{k_j}{\mu}\right)}, \quad \lambda_{j,2} = -\frac{\mu}{2m} - \sqrt{\frac{\mu}{m}\left(\frac{\mu}{4m} - \frac{k_j}{\mu}\right)}.} \tag{4.42}$$

Substituting $x_j(t) \sim e^{\lambda_j t}$ into the equation of motion yields these eigenvalues. Since $\mu, k_j, m > 0$, the real parts are negative, ensuring convergence as $t \to \infty$. The convergence rate is determined by $|\operatorname{Re}\lambda|$, where larger magnitude implies faster decay. One can verify that $|\operatorname{Re}\lambda_{j,1}| \leq |\operatorname{Re}\lambda_{j,2}|$, so the dominant rate is $|\operatorname{Re}\lambda_{j,1}|$. We compare this with $|\operatorname{Re}\lambda_{j,0}| = k_j/\mu$ to assess the effect of momentum.

Consider first $0 < k \le \mu^2/4m$ (temporarily dropping index j). In this regime, both eigenvalues are real and negative, and $|\mathrm{Re}\,\lambda_1| > |\lambda_0|$. Define α by $|\mathrm{Re}\,\lambda_1| = \alpha|\lambda_0| = \alpha k/\mu$, giving

$$\frac{\mu}{2m} - \sqrt{\frac{\mu}{m}\left(\frac{\mu}{4m} - \frac{k}{\mu}\right)} = \frac{\alpha k}{\mu}. \tag{4.43}$$

One can verify

$$\frac{k}{\mu} < \frac{\mu}{2m} - \sqrt{\frac{\mu}{m}\left(\frac{\mu}{4m} - \frac{k}{\mu}\right)} \le \frac{2k}{\mu}, \tag{4.44}$$

since

$$\begin{aligned}
&\frac{k}{\mu} < \frac{\mu}{2m} - \sqrt{\frac{\mu}{m}\left(\frac{\mu}{4m} - \frac{k}{\mu}\right)} \longleftrightarrow \frac{\mu}{2m} - \frac{k}{\mu} > \sqrt{\frac{\mu}{m}\left(\frac{\mu}{4m} - \frac{k}{\mu}\right)} \longleftrightarrow \left(\frac{\mu}{2m} - \frac{k}{\mu}\right)^2 > \frac{\mu}{m}\left(\frac{\mu}{4m} - \frac{k}{\mu}\right) \\
&\qquad \longleftrightarrow \frac{\mu^2}{4m^2} - \frac{\mu}{m}\frac{k}{\mu} + \left(\frac{k}{\mu}\right)^2 > \frac{\mu^2}{4m^2} - \frac{\mu}{m}\frac{k}{\mu} \longleftrightarrow \left(\frac{k}{\mu}\right)^2 > 0, \qquad (4.45) \\
&\frac{\mu}{2m} - \frac{2k}{\mu} \le \sqrt{\frac{\mu}{m}\left(\frac{\mu}{4m} - \frac{k}{\mu}\right)} \longleftrightarrow \frac{\mu^2}{4m^2} - 2\frac{\mu}{m}\frac{k}{\mu} + \frac{4k^2}{\mu^2} \le \frac{\mu}{m}\left(\frac{\mu}{4m} - \frac{k}{\mu}\right) \\
&\qquad \longleftrightarrow \frac{\mu}{m}\frac{k}{\mu} \ge \frac{4k^2}{\mu^2} \longleftrightarrow k \le \frac{\mu^2}{4m}, \qquad (4.46)
\end{aligned}$$

and hence for $0 < k \le \mu^2/4m$, $1 < \alpha \le 2$. Next, for $\mu^2/4m < k < \mu^2/2m$, the eigenvalues become complex with real part $-\mu/2m$, so $|\mathrm{Re}\,\lambda_1| = \mu/2m > \lambda_0$, and α decreases monotonically from 2 to 1 as k increases. Finally, for $k \ge \mu^2/2m$, $|\mathrm{Re}\,\lambda_1| = \mu/2m < \lambda_0$, and α decreases from 1 to 0.

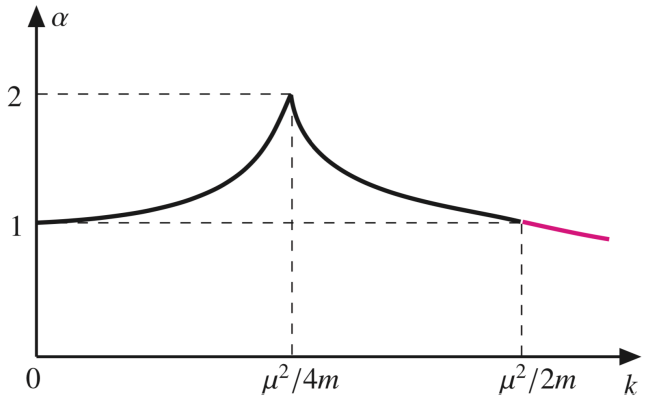

FIG. 4.11: Amplitude factor α as a function of k.

In summary, for positive m, μ, k, we have $|\mathrm{Re}\,\lambda_1| > |\mathrm{Re}\,\lambda_0|$ only when $k < \mu^2/2m$, so momentum improves convergence only in this regime (see FIG. 4.11). Moreover, α reaches its maximum value 2 at $k = \mu^2/4m$, which corresponds to the critical damping condition in physics. As k increases from 0 to $\mu^2/4m$, α increases from 1 to 2, and then decreases back to 1 as k increases to $\mu^2/2m$. For $k > \mu^2/2m$, $\alpha < 1$ and continues decreasing toward 0. A larger α implies faster convergence improvement. Thus, for fixed m and μ, the optimal improvement occurs near $k = \mu^2/4m$, which corresponds to the critical damping regime. For small k, a first-order expansion gives $\lambda_1 \approx \lambda_0$, meaning the system behaves similarly to the no-momentum case in this limit. Hence, without momentum the system behaves

like an overdamped regime, while momentum accelerates convergence by shifting modes toward critical damping. The parameter α satisfies

$$\begin{cases} \dfrac{mk}{\mu^2}\alpha^2-\alpha+1=0, & 0<k\le\mu^2/4m, \\ \dfrac{\mu}{2m}=\dfrac{\alpha k}{\mu}, & k>\mu^2/4m. \end{cases} \tag{4.47}$$

In the range $0<k\le\mu^2/4m$, we have

$$\frac{\partial\alpha}{\partial k}=\frac{\alpha^2 mk}{\mu^2-2\alpha mk}, \tag{4.48}$$

which is positive, and for small k, $\alpha\approx 1+mk/\mu^2$, which is approximately linear.

We now study the discrete case, where the update equation is obtained from Eq. (4.38),

$$\boxed{w_j^{(i+1)}=(1+p-\epsilon k)w_j^{(i)}-p\,w_j^{(i-1)},} \tag{4.49}$$

and, for simplicity, the spatial index "j" will be omitted in the following discussion. Eq. (4.49) can be rewritten in matrix form as

$$\boxed{\begin{pmatrix} w^{(i)} \\ w^{(i+1)} \end{pmatrix}=\begin{pmatrix} 0 & 1 \\ -p & 1+p-\epsilon k \end{pmatrix}\begin{pmatrix} w^{(i-1)} \\ w^{(i)} \end{pmatrix}=\begin{pmatrix} 0 & 1 \\ -p & 1+p-\epsilon k \end{pmatrix}^i\begin{pmatrix} w^{(0)} \\ w^{(1)} \end{pmatrix},} \tag{4.50}$$

so that the convergence properties are determined by the eigenvalues of the characteristic matrix, namely

$$\boxed{\lambda_1=\frac{1+p-\epsilon k+\sqrt{(1+p-\epsilon k)^2-4p}}{2},\ \lambda_2=\frac{1+p-\epsilon k-\sqrt{(1+p-\epsilon k)^2-4p}}{2}.} \tag{4.51}$$

To ensure convergence, one requires $|\lambda_1|<1$ and $|\lambda_2|<1$. A necessary condition is $\epsilon k>0$. Indeed, if $\epsilon k<0$, then

$$\lambda_1=\frac{1+p+|\epsilon k|+\sqrt{(1+p+|\epsilon k|)^2-4p}}{2}\ge\frac{1+p+|\epsilon k|+\sqrt{(1+p)^2-4p}}{2}=\frac{1+p+|\epsilon k|+|1-p|}{2}, \tag{4.52}$$

which is ≥ 1, implying divergence. Thus, convergence requires $\epsilon k>0$. This also implies that the learning rate ϵ must be positive, since $k>0$ by definition.

We define the discriminant Δ as

$$\boxed{\Delta=(1+p-\epsilon k)^2-4p=p^2-2(1+\epsilon k)p+(1-\epsilon k)^2.} \tag{4.53}$$

Its roots are $1+\epsilon k\pm 2\sqrt{\epsilon k}=(1\pm\sqrt{\epsilon k})^2$, hence $\Delta\ge 0$ when $p\le(1-\sqrt{\epsilon k})^2$ or $p\ge(1+\sqrt{\epsilon k})^2$, and $\Delta<0$ when $(1-\sqrt{\epsilon k})^2<p<(1+\sqrt{\epsilon k})^2$, as illustrated in FIG. 4.12.

(a) $\Delta\ge 0$. In this regime, λ_1 and λ_2 are real, and we consider two subcases. (i) $p\ge(1+\sqrt{\epsilon k})^2$. In

this case, $1+p-\epsilon k \geq 2+2\sqrt{\epsilon k}$, which implies $\lambda_{1,2} > 0$ and $\lambda_1 \geq \lambda_2$. Convergence requires

$$\lambda_1 = \frac{1+p-\epsilon k+\sqrt{(1+p-\epsilon k)^2-4p}}{2} < 1, \tag{4.54}$$

which is equivalent to $\sqrt{(1+p-\epsilon k)^2-4p} < 1-p+\epsilon k$. However, $1-p+\epsilon k \leq -2\sqrt{\epsilon k} \leq 0$, while the left-hand side is nonnegative, leading to a contradiction. Hence the system is unstable in this case.

(ii). $p \leq (1-\sqrt{\epsilon k})^2$ with $1+p-\epsilon k \geq 0$. Then $\lambda_1 > \lambda_2 > 0$. The stability condition $\lambda_1 < 1$ reduces to $\epsilon k > 0$ and $1-p+\epsilon k > 0$, which holds since $1-p+\epsilon k \geq 2\sqrt{\epsilon k} > 0$. Thus convergence holds under these conditions. One can also show that these imply $-1 < p < 1$. Indeed, $p > -1$ follows from $p > \epsilon k - 1 > -1$, while $p < 1$ follows by contradiction: assuming $p > 1$ leads to $(1-\sqrt{\epsilon k})^2 \geq p > 1$, which implies inconsistencies with $\epsilon k < 1+p$. Therefore

$$\boxed{-1 < p < 1.} \tag{4.55}$$

If $1+p-\epsilon k \leq 0$, then $\lambda_{1,2} \leq 0$ and $|\lambda_2| > |\lambda_1|$. The stability condition becomes

$$\lambda_2 = \frac{1+p-\epsilon k-\sqrt{(1+p-\epsilon k)^2-4p}}{2} \geq -1, \tag{4.56}$$

which is equivalent to $3+p-\epsilon k > 0$ and $2+2p-\epsilon k > 0$. Thus convergence holds when $p \leq (1-\sqrt{\epsilon k})^2$, $1+p-\epsilon k \leq 0$, $3+p-\epsilon k > 0$, and $2+2p-\epsilon k > 0$, and again $-1 < p < 1$.

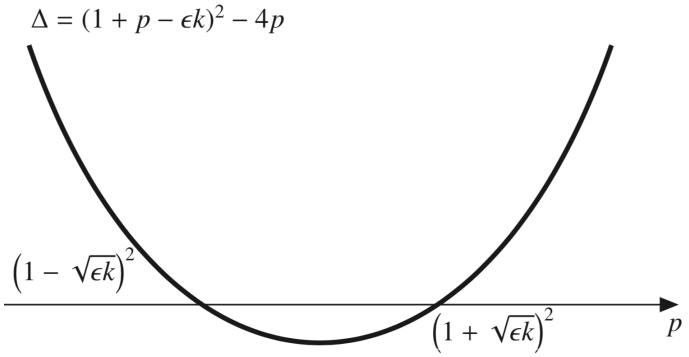

FIG. 4.12: Δ as a function of p.

(b) $\Delta \leq 0$. This occurs when $(1-\sqrt{\epsilon k})^2 < p < (1+\sqrt{\epsilon k})^2$. In this case, λ_1 and λ_2 are complex:

$$\lambda_1 = \frac{1+p-\epsilon+i\sqrt{4p-(1+p-\epsilon k)^2}}{2},\ \lambda_2 = \frac{1+p-\epsilon-i\sqrt{4p-(1+p-\epsilon k)^2}}{2}. \tag{4.57}$$

Stability requires $|\lambda_{1,2}| = \sqrt{p} < 1$, i.e., $p < 1$. Hence convergence holds when $(1-\sqrt{\epsilon k})^2 < p < 1$. For this interval to exist, one must have $(1-\sqrt{\epsilon k})^2 < 1$, i.e., $0 < \epsilon k < 4$.

Combining all cases, the stability condition of the system with momentum is

$$\boxed{-1 < p < 1,\ 0 < \epsilon k < 2+2p,} \tag{4.58}$$

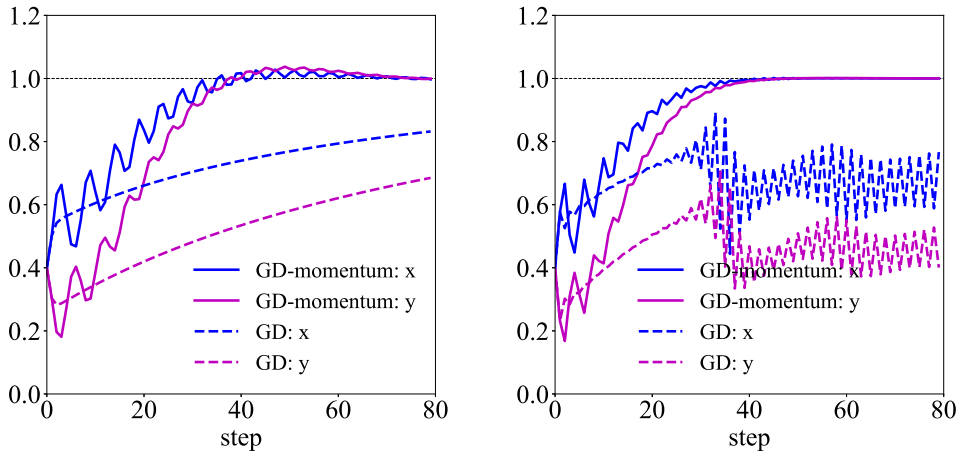

FIG. 4.13: Evolution of $\mathbf{x}$ for the Rosenbrock function using gradient descent with/without momentum.

for every k. This analysis also implies an upper bound on the learning rate,

$$\boxed{\epsilon < 4/k,} \tag{4.59}$$

where $k = \max_j k_j$.

As an example, FIG. 4.13 shows the evolution of $\mathbf{x}$ for the Rosenbrock function under gradient descent with and without momentum, with $a = 1$, $b = 10$, and initial point $(0.4, 0.4)^\top$. The learning rates are $\epsilon = 0.02$ (0.035) and momentum parameters are $p = 0.9$ (0.8) for the left (right) panel. It is evident that standard gradient descent converges slowly and may oscillate, whereas momentum significantly accelerates convergence.

As the second example, we study the minimum search for the function $f(\mathbf{x}) = (x + 3y - 9)^2 + (3x + y - 5)^2$ using the gradient descent with momentum. The eigenvalues of $f(\mathbf{x})$ are $k_{\max} = 32$ and $k_{\min} = 8$, thus $\epsilon_{\max} = 4/32 = 1/8$. Moreover, the momentum parameter p should fulfill the condition $2 + 2p > \epsilon k$, or $p > \epsilon k/2 - 1$. In the following, we take four sets of (ϵ, p),

$$\left(\frac{4}{k_{\max}} - 0.02, \left(1 - \sqrt{\epsilon k_{\max}}\right)^2\right), \tag{4.60}$$

$$\left(\frac{4}{k_{\max}} - 0.1, \left(1 - \sqrt{\epsilon k_{\max}}\right)^2\right), \tag{4.61}$$

$$\left(\frac{4}{k_{\max}} - 0.1, \left(1 - \sqrt{\epsilon k_{\max}}\right)^2 + 0.9\right), \tag{4.62}$$

$$\left(\frac{4}{k_{\max}} - 0.1, \left(1 - \sqrt{\epsilon k_{\max}}\right)^2 - 0.6\right), \tag{4.63}$$

the corresponding results are shown in FIG. 4.14. These curves demonstrate different patterns, indicating the convergence properties of the system controlled by the parameters ϵ and p. It is found that a larger p corresponds to the under-damped situation (the left panel of the second line) while a smaller p corresponds to the over-damped oscillator (the right panel of the first line). For the last panel, the convergence is slow, where for the fixed learning rate $4/k_{\max} - 0.1 = 0.025$, the condition $p > \epsilon k/2 - 1$ gives $p > p_c = -0.6$ while $(1 - \sqrt{\epsilon k_{\max}})^2 - 0.6 \approx -0.5889$, which is very close to p_c.

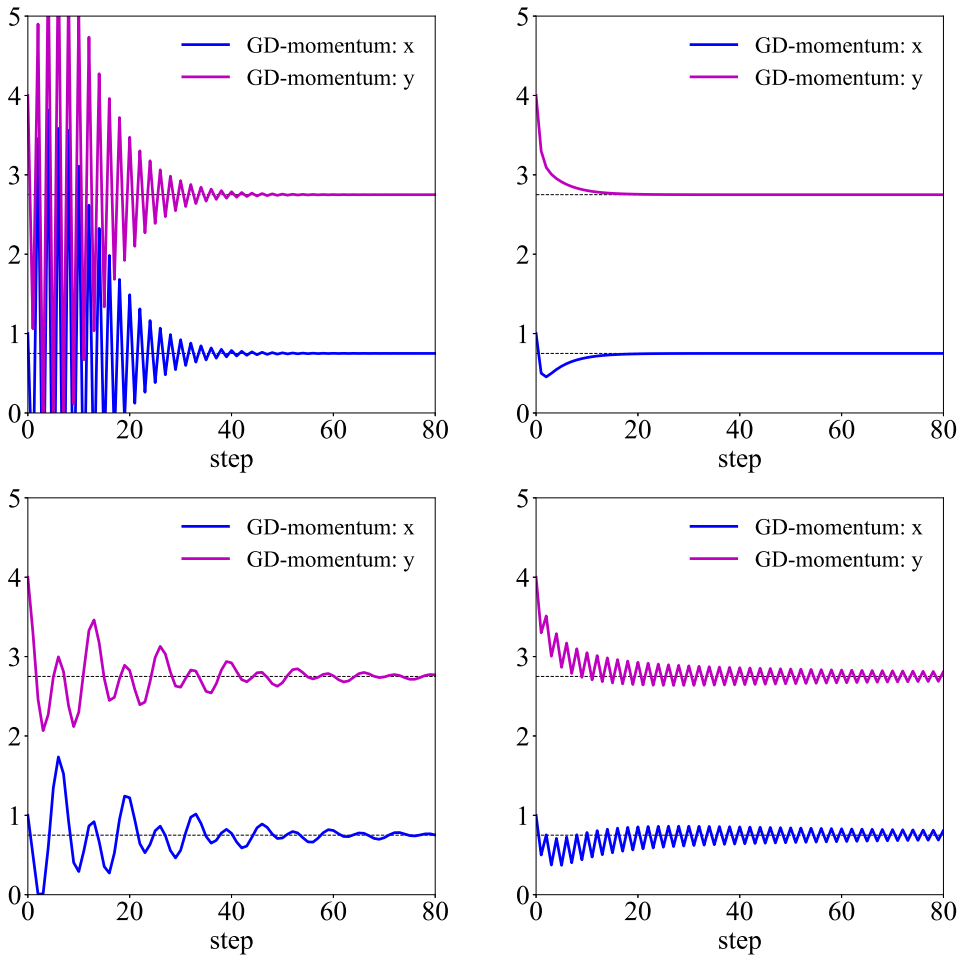

FIG. 4.14: Evolution of x^* and y^* under different ϵ and p, see the context for details.

§4.7 Understand the Momentum Mechanism

Essentially, we can take the viewpoint that momentum-based optimization corresponds to pushing a ball down a hill, and we may then ask what kinds of forces should be included in this picture. Among them one force is proportional to the negative gradient of the cost function (potential): $-\nabla f(\mathbf{x})$. This force pushes the particle downhill along the cost function surface. The gradient descent algorithm would simply take a single step based on each gradient, but the Newtonian scenario used by the momentum algorithm instead uses this force to alter the velocity of the particle. We can think of the particle as a hockey puck sliding down an icy surface. Whenever it descends a steep part of the surface, it gains speed and continues sliding in that direction until it begins to go uphill again. If the only force is the gradient of the cost function, then the particle might never come to rest. Imagine a hockey puck sliding down one side of a valley and straight up the other side, oscillating back and forth forever, assuming the ice is perfectly frictionless. To resolve this problem, we add another force, proportional to the negative velocity. In physics, this force corresponds to viscous drag. This causes the particle to gradually lose energy over time and eventually converge to a local minimum. Mathematically, the step size was previously given by the norm of the gradient multiplied by the

learning rate. Now, the step size depends on both the magnitude and the alignment of a sequence of gradients. The step size is largest when many successive gradients point in the same direction. If the momentum algorithm always observes $\mathbf{g}$, then it will accelerate in the direction of $-\mathbf{g}$, until reaching a terminal velocity where the size of each step is $\epsilon|\mathbf{g}|/(1-p)$. It is useful to think of the momentum parameter in terms of $1/(1-p)$. For instance, $p=0.9$ corresponds to increasing the maximum speed by a factor of about 10 relative to the simple gradient descent algorithm.

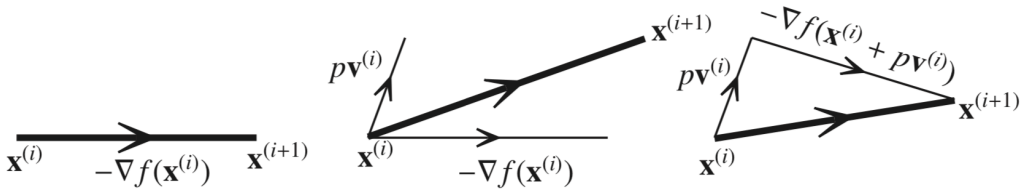

FIG. 4.15: Geometrical meaning of the momentum parameter p in the Polyak/Nesterov scheme.

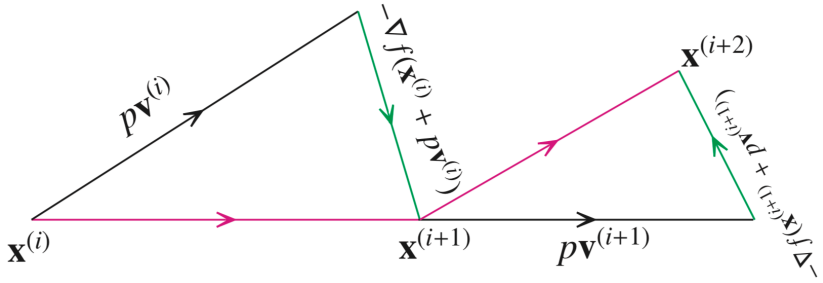

FIG. 4.16: Path of $\mathbf{x}$ during the optimization process under the gradient descent with Nesterov momentum.

The momentum parameter p in the gradient descent algorithm has the following geometric interpretation. Without the parameter p, the evolution of $\mathbf{x}$ in the x-space is controlled purely by the negative gradient, see the left panel of FIG. 4.15, i.e., $\mathbf{x}^{(i+1)}=\mathbf{x}^{(i)}-\epsilon\nabla f(\mathbf{x}^{(i)})$. The learning rate ϵ characterizes the strength of the evolution of the position under the gradient force. However, during the transition from $\mathbf{x}^{(i)}$ to $\mathbf{x}^{(i+1)}$, the particle $\mathbf{x}^{(i)}$ itself moves a distance $p\mathbf{v}^{(i)}$ determined by its velocity $\mathbf{v}^{(i)}$. Thus, the update of the position in x-space can be decomposed into two parts: the inertial motion $p\mathbf{v}^{(i)}$ and the gradient-driven displacement $-\nabla f(\mathbf{x}^{(i)})$. Consequently, one has $\mathbf{x}^{(i+1)}=p\mathbf{v}^{(i)}-\epsilon\nabla f(\mathbf{x}^{(i)})$. This is gradient descent with momentum, and we refer to this momentum as Polyak momentum [∗4-5∗], see the middle panel of FIG. 4.15. On the other hand, if the particle is allowed to respond to the gradient evaluated at the predicted position $\mathbf{x}^{(i)}+p\mathbf{v}^{(i)}$ rather than the current position $\mathbf{x}^{(i)}$, then we obtain a different momentum scheme. This is known as the Nesterov momentum [∗4-6∗] and its explicit form is given by

$$\boxed{\mathbf{v}^{(i+1)}=p\mathbf{v}^{(i)}-\epsilon\nabla f(\mathbf{x}^{(i)}+p\mathbf{v}^{(i)}),\ \ \mathbf{x}^{(i+1)}=\mathbf{x}^{(i)}+\mathbf{v}^{(i+1)}.} \tag{4.64}$$

See the right panel of FIG. 4.15. In gradient descent with Polyak momentum, one can interpret the optimization process as a ball rolling down a hill. However, a ball that blindly follows the slope is highly suboptimal. We would prefer a smarter ball, one that has a notion of its future trajectory and can slow down before the hill begins to slope upward again. Gradient descent with Nesterov mo-

mentum provides precisely this mechanism. It is natural if the momentum parameter p is small, the Nesterov momentum is similar to the Polyak momentum. Also see FIG. 4.16 for the path of $\mathbf{x}$ during the optimization process under the gradient descent with Nesterov momentum.

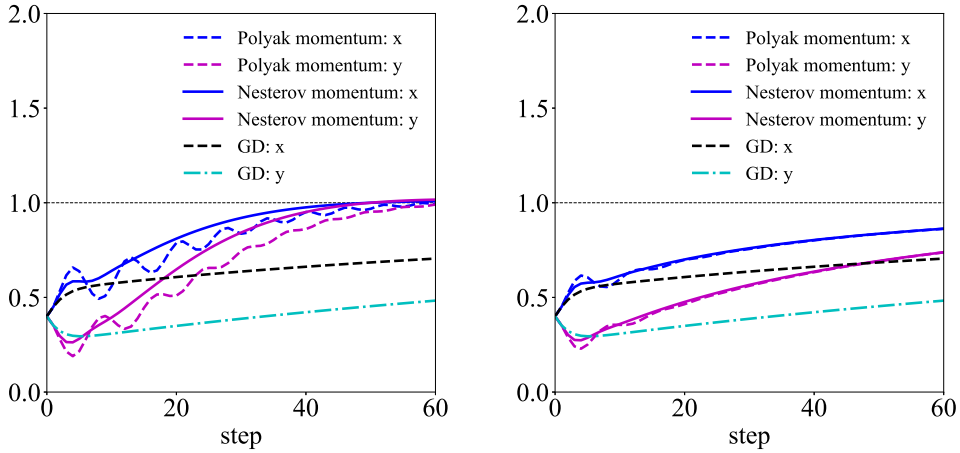

FIG. 4.17: Evolution of $\mathbf{x}$ for the Rosenbrock function using gradient descent with either Polyak or Nesterov momentum. Left: $p = 0.9$; right: $p = 0.7$

In FIG. 4.17, we show the evolution of $\mathbf{x}$ for the Rosenbrock function under gradient descent with Polyak/Nesterov momentum, where $a = 1, b = 10, \epsilon = 0.01$ with $p = 0.9$ (left panel) and $p = 0.7$ (right panel). The main observations are: (a) For a larger momentum parameter p (left panel), the Nesterov mechanism significantly improves the trajectory, making it smoother and leading to reliable convergence; (b) When p is relatively small (right panel), the difference between Nesterov and Polyak momentum becomes minor, indicating that the choice of momentum variant is less critical in this regime; (c) Both Polyak and Nesterov momentum outperform standard gradient descent. Since $p \leq 1$, one can expand the gradient around $p = 0$, i.e.,

$$\nabla f(\mathbf{x}^{(i)} + p\mathbf{v}^{(i)}) \approx \nabla f(\mathbf{x}^{(i)}) + \mathbf{H}(\mathbf{x}^{(i)}) p\mathbf{v}^{(i)}, \tag{4.65}$$

and consequently,

$$\mathbf{v}^{(i+1)} = p\mathbf{v}^{(i)} - \epsilon \nabla f(\mathbf{x}^{(i)} + p\mathbf{v}^{(i)}) \approx \left[\vec{1} - \epsilon \mathbf{H}(\mathbf{x}^{(i)})\right] p\mathbf{v}^{(i)} - \epsilon \nabla f(\mathbf{x}^{(i)}). \tag{4.66}$$

In this sense, the Nesterov momentum introduces an additional term $\epsilon\mathbf{H}$ compared to the Polyak mechanism. See FIG. 4.18 for the corresponding effect, where it is observed that the approximation (4.66) reproduces the exact dynamics well and yields slightly improved convergence.

EXERCISE 4-6. Use the gradient descent algorithm with/without momentum to searching the minimum of $f(x) = x^4$.

§4.8 Extensions of Gradient Descent (First-order in Nature)

There exist several variants of the gradient descent algorithm, with or without a momentum mechanism. We briefly summarize them here and comment on their main features. A more detailed analysis is beyond the present scope, and we refer interested readers to the original papers:

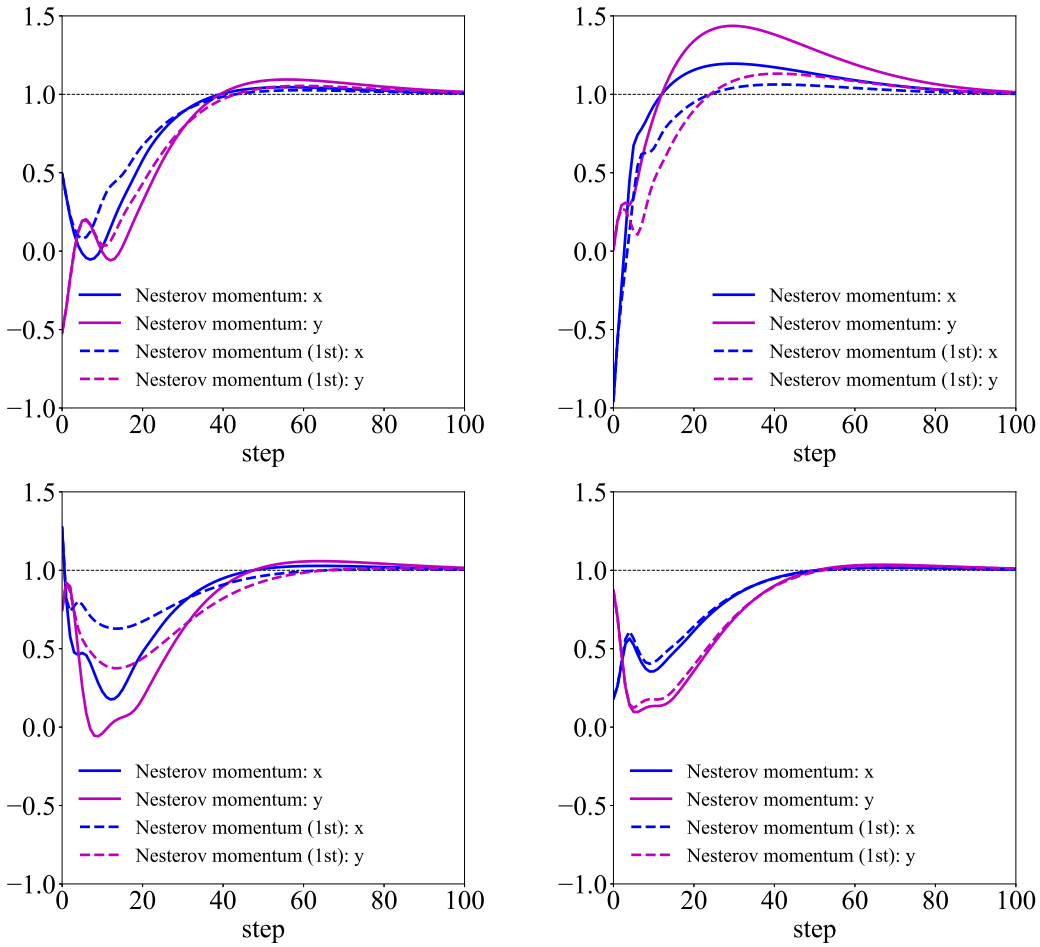

FIG. 4.18: Effects of the correction $\vec{1}-\epsilon\mathbf{H}$, where the results are run randomly.

(a) AdaGrad is the abbreviation for the adaptive subgradient method, in which a separate learning rate is assigned to each component of $\mathbf{x}$. The algorithm reads [∗4-7∗]:

$$\boxed{\mathbf{x}^{(i+1)}=\mathbf{x}^{(i)}-\frac{\epsilon}{\delta+\sqrt{\mathbf{s}^{(i+1)}}}\odot\mathbf{g}_i,\ \ \mathbf{s}^{(i+1)}=\mathbf{s}^{(i)}+\mathbf{g}_i\odot\mathbf{g}_i,\ \ \delta\approx 10^{-8}.} \tag{4.67}$$

The element-wise product between two vectors is defined as $\mathbf{a}\odot\mathbf{b}=(a_1b_1,a_2b_2,\cdots,a_db_d)^\top$, where $\mathbf{a},\mathbf{b}\in\mathrm{R}^d$. AdaGrad is significantly less sensitive to the choice of learning rate ϵ. However, its primary drawback is that the components of $\mathbf{s}$ are strictly nondecreasing, and $\sqrt{\mathbf{s}}$ is understood element-wise. As a result, the accumulated sum causes the effective learning rate to continuously decrease, eventually becoming infinitesimally small before convergence is achieved.

(b) RMSProp extends AdaGrad to avoid the effect of a monotonically decreasing learning rate. It maintains a decaying average of squared gradients [∗4-8∗],

$$\boxed{\mathbf{x}^{(i+1)}=\mathbf{x}^{(i)}-\frac{\epsilon}{\delta+\sqrt{\mathbf{s}^{(i+1)}}}\odot\mathbf{g}_i,\ \ \mathbf{s}^{(i+1)}=\gamma\mathbf{s}^{(i)}+(1-\gamma)\mathbf{g}_i\odot\mathbf{g}_i.} \tag{4.68}$$

Here, the denominator resembles the root mean square of the gradient components.

(c) Adadelta is another method designed to overcome AdaGrad's monotonically decreasing learning rate [∗4-9∗],

$$\mathbf{s}^{(i+1)} = \gamma \mathbf{s}^{(i)} + (1-\gamma)\mathbf{g}_i \odot \mathbf{g}_i, \quad \mathbf{v}^{(i+1)} = -\frac{\sqrt{\mathbf{u}^{(i)}} + \delta}{\delta + \sqrt{\mathbf{s}^{(i+1)}}} \odot \mathbf{g}_i, \tag{4.69}$$

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \mathbf{v}^{(i+1)}, \quad \mathbf{u}^{(i+1)} = \gamma \mathbf{u}^{(i)} + (1-\gamma)\mathbf{v}^{(i+1)} \odot \mathbf{v}^{(i+1)}. \tag{4.70}$$

(d) The adaptive moment estimation method, or Adam [∗4-10∗], also assigns adaptive learning rates to each parameter. It stores both an exponentially decaying squared gradient, as in RMSProp and Adadelta, and an exponentially decaying gradient, as in momentum. Since initializing the first- and second-moment estimates to zero introduces bias, a correction step is applied. The update equations for Adam are

$$\text{biased decaying momentum}: \mathbf{v}^{(i+1)} = \gamma_{\mathrm{v}} \mathbf{v}^{(i)} + (1-\gamma_{\mathrm{v}})\mathbf{g}_i, \tag{4.71}$$

$$\text{biased decaying square gradient}: \mathbf{s}^{(i+1)} = \gamma_{\mathrm{s}} \mathbf{s}^{(i)} + (1-\gamma_{\mathrm{s}})\mathbf{g}_i \odot \mathbf{g}_i, \tag{4.72}$$

$$\text{corrected decaying momentum}: \widehat{\mathbf{v}}^{(i+1)} = \frac{\mathbf{v}^{(i+1)}}{1-\gamma_{\mathrm{v}}^{i+1}}, \tag{4.73}$$

$$\text{corrected decaying square gradient}: \widehat{\mathbf{s}}^{(i+1)} = \frac{\mathbf{s}^{(i+1)}}{1-\gamma_{\mathrm{s}}^{i+1}}, \tag{4.74}$$

$$\text{update}: \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \frac{\epsilon \widehat{\mathbf{v}}^{(i+1)}}{\delta + \sqrt{\widehat{\mathbf{s}}^{(i+1)}}}. \tag{4.75}$$

(e) Nadam (Nesterov-accelerated Adaptive Moment Estimation) combines Adam with the Nesterov momentum mechanism. The difference from Adam lies in the update step [∗4-11∗],

$$\text{update}: \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \frac{\epsilon}{\delta + \sqrt{\widehat{\mathbf{s}}^{(i+1)}}} \left(\gamma_{\mathrm{v}} \widehat{\mathbf{v}}^{(i+1)} + \frac{(1-\gamma_{\mathrm{v}})\mathbf{g}_i}{1-\gamma_{\mathrm{v}}^{i}} \right). \tag{4.76}$$

(f) Closely related is the AdamW (Adam-weight) approach [∗4-12∗], in which a modification to the gradient is applied before computing the biased momentum in Adam, i.e.,

$$\mathbf{g}_i \leftarrow \mathbf{g}_i + \lambda \mathbf{x}^{(i)}, \tag{4.77}$$

where λ is a parameter. The final parameter update is then given by

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \frac{\epsilon}{\delta + \sqrt{\widehat{\mathbf{s}}^{(i+1)}}} \left(\widehat{\mathbf{v}}^{(i+1)} + \lambda \mathbf{x}^{(i)} \right). \tag{4.78}$$

(g) The use of exponential averaging is well motivated [∗4-13∗], as it prevents the learning rate from becoming infinitesimally small during training, which is the key limitation of AdaGrad. However, this short-term memory of gradients can become problematic in certain situations. In particular, when Adam converges to a suboptimal solution, it has been observed that some mini-batches yield large and informative gradients, but because such mini-batches occur infrequently, exponential averaging suppresses their influence, leading to poor convergence. To ad-

dress this issue, AMSGrad replaces the exponential average with the maximum of past squared gradients when updating the parameters, resulting in the algorithm

$$\mathbf{v}^{(i+1)} = \gamma_v \mathbf{v}^{(i)} + (1-\gamma_v)\mathbf{g}_i, \quad \mathbf{s}^{(i+1)} = \gamma_s \mathbf{s}^{(i)} + (1-\gamma_s)\mathbf{g}_i \odot \mathbf{g}_i, \tag{4.79}$$

$$\widehat{\mathbf{s}}^{(i+1)} = \max\left[\mathbf{s}^{(i+1)}, \widehat{\mathbf{s}}^{(i)}\right], \quad \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \frac{\epsilon \mathbf{v}^{(i+1)}}{\delta + \sqrt{\widehat{\mathbf{s}}^{(i+1)}}}. \tag{4.80}$$

In this way, AMSGrad ensures a non-increasing step size, thereby avoiding the convergence issues encountered by Adam. For simplicity, no debiasing step is included here.

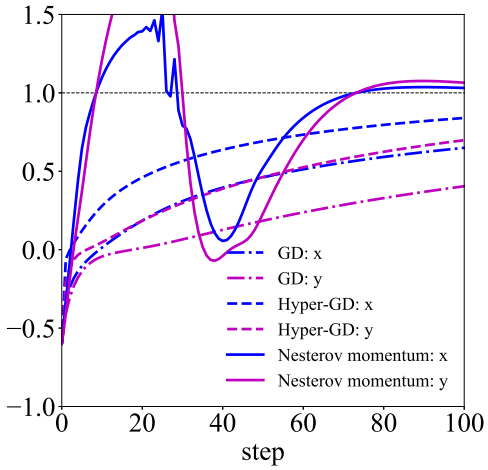

FIG. 4.19: Trajectory (evolution) of $\mathbf{x}$ for the Rosenbrock function, where $a = 1, b = 10$ and $\epsilon_0 = 0.01$. The parameters are $p = 0.9$ and $\mu = 10^{-5}$.

Accelerated descent methods are often either highly sensitive to the learning rate or employ sophisticated strategies to adapt it during training. The learning rate determines how strongly the method responds to the gradient signal, and choosing a value that is too large or too small can severely degrade performance. A hypergradient is a derivative taken with respect to a hyperparameter. Hypergradient methods aim to reduce this sensitivity by allowing hyperparameters to adapt dynamically. For example, hypergradient descent applies gradient descent to the learning rate itself. This requires the partial derivative of the objective function with respect to the learning rate,

$$\frac{\partial f(\mathbf{x}^{(i)})}{\partial \epsilon} = \mathbf{g}_i^\top \frac{\partial}{\partial \epsilon}\left(\mathbf{x}^{(i-1)} - \epsilon \mathbf{g}_{i-1}\right) = -\mathbf{g}_i^\top \mathbf{g}_{i-1}, \tag{4.81}$$

which leads to the update rule $\epsilon_{i+1} = \epsilon_i - \mu \partial f(\mathbf{x}^{(i)})/\partial \epsilon$, or

$$\epsilon_{i+1} = \epsilon_i + \mu \mathbf{g}_i^\top \mathbf{g}_{i-1}, \tag{4.82}$$

where μ is the corresponding hyperparameter [∗4-14∗]. See FIG. 4.19 for an example.

§4.9 *Using 3rd-order Derivative to Design Learning Rate

One may further expand the function to higher orders in ϵ. For a one-dimensional function $f(x)$, one has

$$\boxed{f(x-\epsilon g)\approx f(x)-\epsilon g^2+\frac{1}{2}\epsilon^2 g^2 H-\frac{1}{6}\epsilon^3 g^3 T,} \tag{4.83}$$

where $T=\partial^3 f(x)/\partial x^3$ denotes the third-order derivative of $f(x)$. For convenience, we introduce the following notations: $\psi=g^3T=f'^3f'''$, $\xi=g^2H=f'^2f''$ and $\zeta=g^2=f'^2$. With these definitions, the change in the function value can be written as

$$\boxed{w(\epsilon)\equiv\Delta f\equiv f(x-\epsilon f')-f(x)\approx-\frac{1}{6}\psi\epsilon^3+\frac{1}{2}\xi\epsilon^2-\zeta.} \tag{4.84}$$

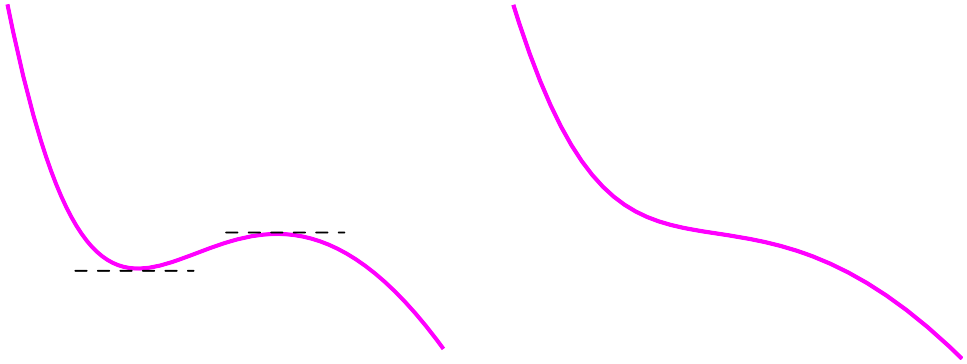

FIG. 4.20: Sketch of the function $w(\epsilon)$ with $\psi<0$ and $\mu>0$ (left) and $\psi<0$ and $\mu<0$ (right).

The objective is to choose ϵ such that $w(\epsilon)$ is minimized. However, since $w(\epsilon)$ is a cubic function in ϵ, it is in general unbounded from below. In particular, depending on the sign of ψ, one has either $\lim_{\epsilon\to\pm\infty}w(\epsilon)=\pm\infty$ or $\lim_{\epsilon\to\pm\infty}w(\epsilon)=\mp\infty$. Therefore, instead of searching for a global minimum, we restrict ourselves to selecting a suitable local extremum. To this end, we consider the stationary condition $\partial w(\epsilon)/\partial\epsilon=0$, which gives

$$\psi\epsilon^2-2\xi\epsilon+2\zeta=0. \tag{4.85}$$

Defining the discriminant $\mu\equiv\xi^2-2\psi\zeta$, we distinguish two cases:

(a) If $\mu\geq 0$, the above quadratic equation admits two real roots. This implies that $w(\epsilon)$ possesses one local minimum and one local maximum, regardless of the sign of ψ. An example with $\psi<0$ and $\mu>0$ is illustrated in the left panel of FIG. 4.20. In this case, we select the stationary point ϵ^* satisfying $w''(\epsilon^*)>0$, corresponding to the local minimum. One finds

$$\epsilon^*=\frac{\xi-\sqrt{\xi^2-2\psi\zeta}}{\psi}, \tag{4.86}$$

whose derivation will be clarified in the following discussion.

(b) If $\mu<0$, the stationary equation has no real roots, and thus $w(\epsilon)$ has no local minimum (see the right panel of FIG. 4.20). In this situation, a natural choice is to take the real part of the complex

roots, leading to $\epsilon^* = \xi/\psi$.

We refer to this scheme as the gradient descent with exact line search to second-order (or equivalently, gradient descent with tensor correction). In contrast, neglecting the T term reduces the method to the gradient descent with exact line search to first-order, which is the standard exact line search scheme. To gain further insight, consider the regime $0 < 2f'f''' \ll f''^2$. In this limit, one finds

$$\sqrt{(f'^2 f'')^2 - 2f'^2 f'^3 f'''} \sim \sqrt{f''^2 - 2f'f'''} \approx f'' - \frac{f'f'''}{f''}, \tag{4.87}$$

and consequently

$$\epsilon^* \approx \frac{f'' \pm (f'' - f'f'''/f'')}{f'f'''} = \frac{1}{f''}, \tag{4.88}$$

where the final expression is obtained by taking the "−" branch. This recovers precisely the result of the first-order exact line search, indicating that the tensor correction becomes negligible in this regime. On the other hand, when $2f'f'''$ is comparable to f''^2, the tensor contribution introduces a significant correction. Keeping the next-order term yields

$$\boxed{\epsilon^* \approx \frac{1}{f''} + \frac{f'f'''}{2f''^3}.} \tag{4.89}$$

For higher-dimensional problems, the third-order derivative generalizes to a tensor $\mathcal{T}$ with components $\partial^3 f(\mathbf{x})/\partial x_i \partial x_j \partial x_k$. This tensor contains d^3 elements for a d-dimensional problem (e.g., 27 elements for $d = 3$), making the practical implementation considerably more challenging. Nevertheless, this formulation provides a systematic extension of the line search strategy and may be advantageous for problems with moderate dimensionality. The above selection rule can be extended to higher dimensions as [∗4-15∗]

$$\boxed{\epsilon^* = \begin{cases} \dfrac{\mathbf{g}^\top \mathbf{H}\mathbf{g} - \sqrt{(\mathbf{g}^\top \mathbf{H}\mathbf{g})^2 - 2\mathbf{g}^\top \mathbf{g}\mathcal{T}\mathbf{ggg}}}{\mathcal{T}\mathbf{ggg}}, & (\mathbf{g}^\top \mathbf{H}\mathbf{g})^2 - 2\mathbf{g}^\top \mathbf{g}\mathcal{T}\mathbf{ggg} \geq 0, \\ \dfrac{\mathbf{g}^\top \mathbf{H}\mathbf{g}}{\mathcal{T}\mathbf{ggg}}, & (\mathbf{g}^\top \mathbf{H}\mathbf{g})^2 - 2\mathbf{g}^\top \mathbf{g}\mathcal{T}\mathbf{ggg} < 0, \end{cases}} \tag{4.90}$$

where $\mathcal{T}\mathbf{ggg}$ denotes $[\partial^3 f(\mathbf{x})/\partial x_i \partial x_j \partial x_k](\partial f(\mathbf{x})/\partial x_i)\cdot(\partial f(\mathbf{x})/\partial x_j)\cdot(\partial f(\mathbf{x})/\partial x_k)$ under the summation convention. In analogy with the one-dimensional case, a perturbative approximation can be obtained when $2\mathbf{g}^\top \mathbf{g}\mathcal{T}\mathbf{ggg} \ll (\mathbf{g}^\top \mathbf{H}\mathbf{g})^2$, leading to

$$\boxed{\epsilon^* \approx \epsilon^*_{\mathrm{H}} \left(1 + \frac{\epsilon^*_{\mathrm{H}}}{2} \frac{\mathcal{T}\mathbf{ggg}}{\mathbf{g}^\top \mathbf{H}\mathbf{g}}\right),} \tag{4.91}$$

where ϵ^*_{H} is the learning rate obtained by retaining only the Hessian contribution.

In FIG. 4.21, the evolution of x under different optimization schemes is shown for the two representative functions $f(x) = (x-1)^4$ and $f(x) = \exp(x^2 - 2) - x$. The initial condition is chosen as $x^{(0)} = 1.2$ with $\epsilon_0 = 0.5$, while the decaying factors are taken as 1.2 and 0.8, respectively, for illustration. It is clearly seen that the tensor-corrected algorithm converges significantly faster than the standard exact line search (which coincides with Newton's method in one dimension), and both outperform

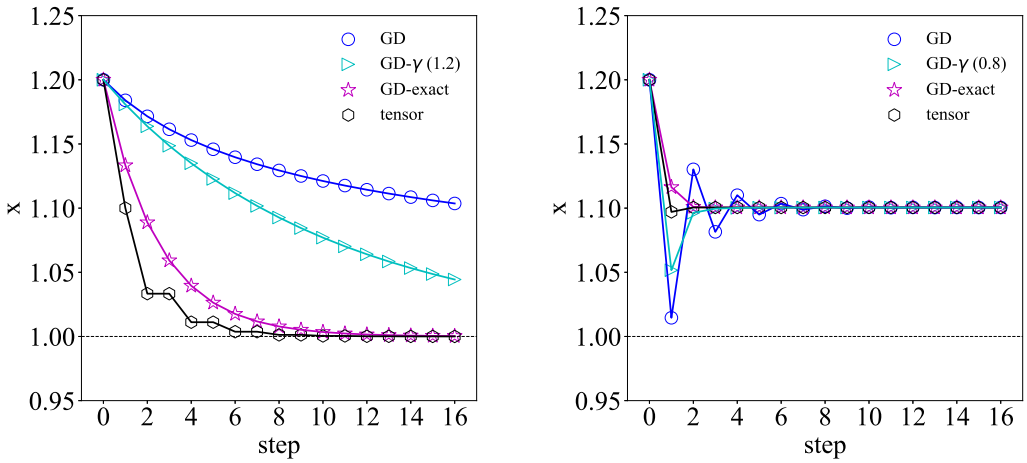

FIG. 4.21: Evolution of x under gradient descent (with/without decaying learning rate), gradient descent with exact line search and the one including the tensor contribution. Left: $f(x) = (x-1)^4$, right: $f(x) = \exp(x^2-2)-x$.

the simple gradient descent schemes. For example, for $f(x) = (x-1)^4$, one finds $2f'f'''/f''^2 = 4/3$, indicating that the tensor correction is indeed substantial. In the second example, the tensor method essentially converges to the optimal solution within a single effective step.

§4.10 Problems for This Lecture

Theoretical

1. For the function $f(x) = x^{2n}$ with n an integer greater than 2, work out the expression for the number of the effective steps to terminate at the condition $|x^{(i+1)} - x^{(i)}| \lesssim \sigma$ using the gradient descent algorithm.
2. Assume that $f(\mathbf{x}) = \sin(xy) + e^{y+z} - z$, and one starts from the initial value $\mathbf{x}^{(0)} = (1,2,3)^\top$ along the direction $\mathbf{d} = (0,-1,-1)^\top$. Find the optimized ϵ and the local minimum.
3. Use the bisection method to find the roots of the function $f(x) = x^3 - 5x + 1$.
4. The condition number is defined as $\kappa = |xf'(x)/f(x)|$ for the 1-dimensional function $f(x)$. Investigate the properties of the condition number for the function

$$f(x) = e^x,\ \sin x,\ \cos x,\ \tan x,\ \arcsin x,\ \arccos x,\ \arctan x,\ \cosh x,\ \sinh x,\ \tanh x. \tag{4.92}$$

5. For the situation $p \le (1-\sqrt{\epsilon k})^2$ and $1 + p - \epsilon k \le 0$, prove that the stability condition naturally leads to $-1 < p < 1$.
6. Jensen's inequality could be generalized as:

$$\boxed{f\left(\sum_{i=1}^{\Lambda} \lambda_i x_i\right) \le \sum_{i=1}^{\Lambda} \lambda_i f(x_i),\ \sum_i \lambda_i = 1,\ \lambda_i \ge 0.} \tag{4.93}$$

   Use mathematical induction to finish the proof.
7. Prove the inequality, $\mathrm{E}[|xy|] \le [\mathrm{E}[|x|^p]]^{1/p}[\mathrm{E}[|y|^q]]^{1/q}$, using Jensen's inequality for convex function, where p and q are two positive numbers and $1/p + 1/q = 1$. For the case $p = q = 2$, what's the

geometrical meaning? What's the relation between the arithmetical mean and the geometrical mean of n positive numbers.

8. For a function $f(\mathbf{x})$ with $\mathbf{x} \in \mathrm{R}^d$, the tensor algorithm for determining the learning rate could be written as

$$\boxed{\epsilon^* = \frac{g_I H_{IJ} g_J \pm \sqrt{(g_I H_{IJ} g_J)^2 - 2 g_I g_J g_K \mathcal{T}_{IJK} g_L g_L}}{g_I g_J g_K \mathcal{T}_{IJK}}}, \tag{4.94}$$

here we use the capital letters for the summation indexes, i.e., I, J, K, L take values from 1 to d, which should not be confused with the notation $\mathbf{g}_i \equiv \mathbf{g}(\mathbf{x}^{(i)})$ for the iterative gradient, etc. Moreover,

$$H_{IJ} = \frac{\partial^2 f(\mathbf{x})}{\partial x_I \partial x_J}, \ \mathcal{T}_{IJK} = \frac{\partial^3 f(\mathbf{x})}{\partial x_I \partial x_J \partial x_K}, \ I, J, K = 1 \sim d, \tag{4.95}$$

are the element of the Hessian matrix and that of the tensor $\mathcal{T}$. Assume that d is a smaller number (e.g., smaller than 5), how to implement this learning rate to the gradient descent method? Can one further incorporate the effects of the fourth-order derivative (quartic term), i.e.,

$$\mathcal{Q}_{IJKL} = \frac{\partial^4 f(\mathbf{x})}{\partial x_I \partial x_J \partial x_K \partial x_L}, \tag{4.96}$$

into to the gradient descent algorithm with exact line search? How could one calculate the quartic term $\mathcal{Q}\mathbf{sss}$ (understood as the quantity with component $\mathcal{Q}_{IJKL} s_J s_K s_L$) by the expansion of $\mathbf{g}(\mathbf{x} \pm \delta \mathbf{s})$? Apply these algorithms to finding the optimal $\mathbf{x}^*$ of the Rosenbrock function.

Computational/Programming

9. Use the gradient descent with/without the exact-line search mechanism and the gradient descent with Polyak/Nesterov scheme to find the minimum of the objective function

$$f(\mathbf{x}) = 1 + \sum_{i=2}^{10} \Big[100 \big(x_i - x_{i-1}^2 \big)^2 + (1 - x_{i-1})^2 \Big], \tag{4.97}$$

with initial points set as $x_i^{(0)} = i/10$, $i = 1, 2, \cdots, 10$.

10. Adopt the AdaGrad, RMSProp, Adadelta, Adam, Nadam, AdamW and AMSGrad to search the minimum of the Rosenbrock function with $a = 1$, $b = 5$ and $x^{(0)} = y^{(0)} = 0.5$.

References and Notes for Lecture 4

[∗4-1∗] A popular approach to setting the learning rate is to use a decaying factor, $\epsilon_i = \epsilon_0 \gamma^i, 0 \le \gamma \le 1$. However, the choice of the parameter γ is artificial, in the sense that it is imposed externally rather than arising from an internal mechanism.

[∗4-2∗] For example, equation of motion of the damped oscillator is given by $m\ddot{\mathbf{x}} + \mu\dot{\mathbf{x}} + k\mathbf{x} = 0$, where μ is the damping constant introduced via the force $\mathbf{f} = -\mu\mathbf{v} = -\mu\dot{\mathbf{x}}$, and k is the Hooke's spring constant.

[∗4-3∗] For the detailed relevant discussion and the α parameter and the conditions (4.47), (4.58) and (4.59), see, N. Qian, *On the Momentum Term in Gradient Descent Learning Algorithms*, Neural Networks, **12**, 145 (1999).

[∗4-4∗] It is necessary to point out that the gradient descent with momentum algorithm could be put in other forms, e.g., $\mathbf{v}^{(i+1)} = p\mathbf{v}^{(i)} - \epsilon \mathbf{g}_i, \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \mathbf{v}^{(i+1)}$, or $\mathbf{z}^{(i+1)} = p\mathbf{z}^{(i)} + \mathbf{g}_i, \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \epsilon \mathbf{z}^{(i+1)}$.

[∗4-5∗] B. Polyak, *Some Methods of Speeding up The Convergence of Iteration Methods*, USSR Computational Mathematics and Mathematical Physics, **4**, 1 (1964).

[∗4-6∗] Y. Nesterov, *A Method for Unconstrained Convex Minimization Problem with The Rate of Convergence* $\mathcal{O}(1/k^2)$, Doklady AN USSR **269**, 543 (1983).

[∗4-7∗] J. Duchi, E. Hazan, and Y. Singer, *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*, JMLR, **2121** (2011).
[∗4-8∗] See Hinton's lecture at https://www.cs.toronto.edu.
[∗4-9∗] M. Zeiler, *An Adaptive Learning Rate Method*, arXiv:1212.5701, 2012.
[∗4-10∗] D. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980, 2014.
[∗4-11∗] T. Dozat, *Incorporating Nesterov Momentum into Adam*, ICLR, 2016.
[∗4-12∗] I. Loshchilov and F. Hutter, *Decoupled Weight Decay Regularization*, arXiv:1711.05101, 2017.
[∗4-13∗] S.J. Reddi, S. Kale, and S. Kumar, *On the Convergence of Adam and Beyond*, ICLR, 2018.
[∗4-14∗] A.G. Baydin *et al.*, *Online Learning Rate Adaptation with Hypergradient Descent*, ICLR, 2018.
[∗4-15∗] We thank Jia-Hong Wang for helpful discussion on this issue of this section.