CS 170: Gradient Descent — Lecture Notes

Course: CS 170 — Efficient Algorithms and Intractable Problems, Spring 2026 **Instructors:** Lijie Chen, Umesh V. Vazirani **Lecture 15**

Part 1: Why Gradient Descent?

In the previous lecture we saw how **backpropagation** computes the gradient $\nabla f(\mathbf{w})$ of a loss function with respect to all parameters in a single backward pass over a computational graph. But computing the gradient is only half the story — we still need to **use** it to find good parameters. This is the job of **gradient descent**.

1.1 The Optimization Problem

We want to solve:

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w})$$

where f is a differentiable loss function (e.g., mean squared error, cross-entropy) and $\mathbf{w}$ is a vector of parameters (weights and biases of a neural network, coefficients of a model, etc.).

1.2 What Does the Gradient Mean?

Before diving into the algorithm, let's build intuition for the gradient in multiple dimensions.

For a function $f(\mathbf{w})$ of d variables, the **partial derivative** $\frac{\partial f}{\partial w_i}$ tells us: if we perturb only the i-th coordinate by a tiny amount ε, the function value changes by approximately:

$$f(w_1, w_2, \ldots, w_i + \varepsilon, \ldots, w_d) \approx f(\mathbf{w}) + \varepsilon \frac{\partial f}{\partial w_i}(\mathbf{w})$$

Now suppose we perturb **all** coordinates simultaneously, each by a small amount ε_i. Writing $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_d)$, the overall change is (to first order) a sum of the individual effects:

$$f(\mathbf{w} + \boldsymbol{\varepsilon}) \approx f(\mathbf{w}) + \sum_{i=1}^{d} \varepsilon_i \frac{\partial f}{\partial w_i}(\mathbf{w}) = f(\mathbf{w}) + \nabla f(\mathbf{w})^\top \boldsymbol{\varepsilon}$$

The key point: as long as the perturbations are small, the change in f is **linear** in $\boldsymbol{\varepsilon}$. The gradient $\nabla f(\mathbf{w}) = \left(\frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_d}\right)$ packages all these partial derivatives into a single vector that completely characterizes the local first-order behavior of f.

1.3 The Gradient Descent Update Rule

The **gradient** $\nabla f(\mathbf{w})$ points in the direction of steepest **increase** of f. To decrease f, we step in the **opposite** direction:

$$\boxed{\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla f(\mathbf{w}_t)}$$

where $\eta > 0$ is the **learning rate** (step size) and $t = 0, 1, 2, \ldots$ counts the iterations.

1.4 Why the Gradient Direction?

Why does GD step along $-\nabla f(\mathbf{w})$ specifically? The idea is that we want to keep each step **short** — if we move too far, the linear approximation from Section 1.2 breaks down and we can no longer trust it to predict the change in f.

So suppose we fix the step length: we require $\|\boldsymbol{\varepsilon}\| = \delta$ for some small $\delta > 0$. Among all such steps, which one decreases f the most? The linear approximation says:

$$f(\mathbf{w} + \boldsymbol{\varepsilon}) \approx f(\mathbf{w}) + \nabla f(\mathbf{w})^\top \boldsymbol{\varepsilon}$$

We want to **minimize** $\nabla f(\mathbf{w})^\top \boldsymbol{\varepsilon}$ subject to $\|\boldsymbol{\varepsilon}\| = \delta$. By the Cauchy–Schwarz inequality:

$$\nabla f(\mathbf{w})^\top \boldsymbol{\varepsilon} \geq -\|\nabla f(\mathbf{w})\| \cdot \|\boldsymbol{\varepsilon}\| = -\delta \|\nabla f(\mathbf{w})\|$$

with equality when $\boldsymbol{\varepsilon} = -\delta \frac{\nabla f(\mathbf{w})}{\|\nabla f(\mathbf{w})\|}$, i.e., pointing in the **negative gradient** direction. So the gradient descent update $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla f(\mathbf{w})$ moves in the direction that gives the greatest decrease in f per unit step, according to the linear approximation.

A natural question: How large should η be? If η is too small, we make very slow progress. If η is too large, we move so far that the linear approximation becomes inaccurate and we might actually *increase* the loss.

The intuition is: if f is relatively **smooth** — meaning the gradient does not change much as we move around — then the linear approximation stays accurate over larger steps, so η can be large. On the other hand, if f is **not smooth** — the gradient changes rapidly — then even a small step can land us somewhere the linear approximation badly misrepresents f, so η must be small. We will make this precise in Part 2 with a concrete example and in Parts 3–4 with general theory.

Part 2: A Simple Example — $f(w) = 3w^2 + 1$

Let's now look at gradient descent in a very simple toy setting to build concrete intuition for how large η should be.

2.1 Setup

Consider the one-dimensional function:

$$f(w) = 3w^2 + 1$$

Its gradient is $f'(w) = 6w$, and the unique minimum is at $w^* = 0$ with $f(w^*) = 1$.

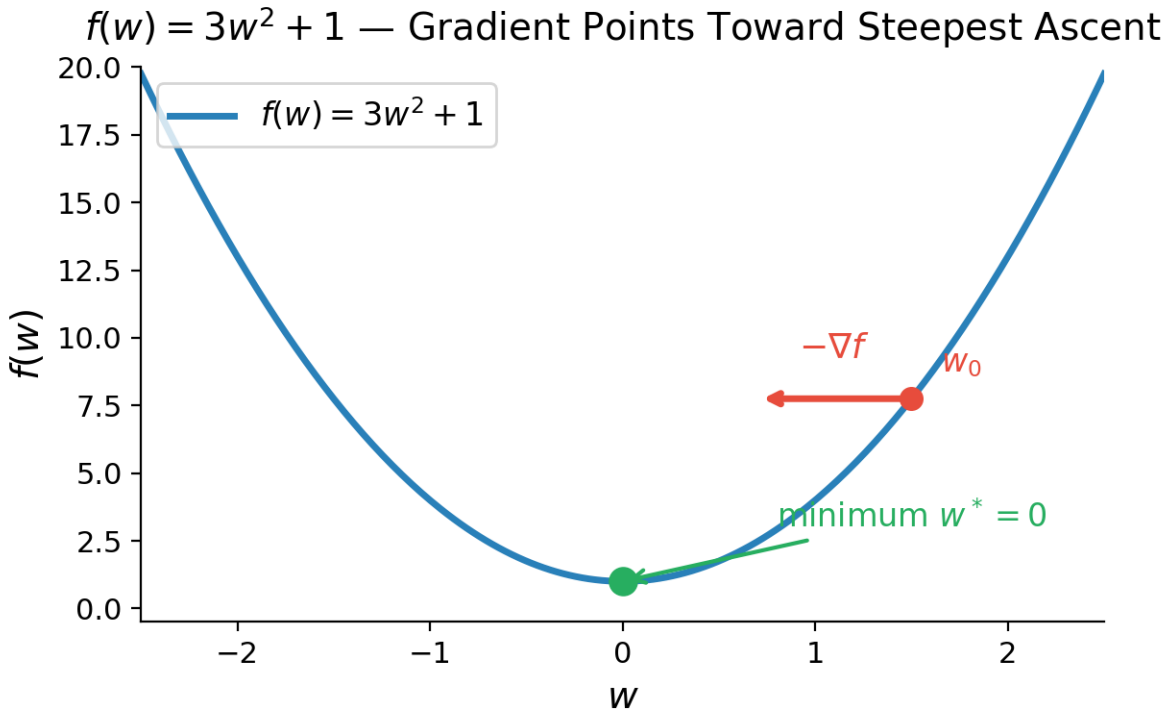

Figure 1: Quadratic function with gradient arrow

2.2 GD as a Geometric Sequence

The gradient descent update becomes:

$$w_{t+1} = w_t - \eta \cdot 6w_t = (1 - 6\eta)\, w_t$$

This is a **geometric sequence** with ratio $r = 1 - 6\eta$. Starting from w_0:

$$w_t = (1 - 6\eta)^t\, w_0$$

The behavior depends entirely on whether $|r| = |1 - 6\eta|$ is less than, equal to, or greater than 1.

2.3 Three Regimes

Regime	Condition on η	Ratio $r = 1 - 6\eta$	Behavior
Convergence	$0 < \eta < 1/3$	$\lvert r\rvert < 1$	$w_t \to 0$ exponentially
Monotone convergence	$0 < \eta < 1/6$	$0 < r < 1$	Approaches minimum from one side
Oscillatory convergence	$1/6 < \eta < 1/3$	$-1 < r < 0$	Bounces around minimum, still converges
Divergence	$\eta > 1/3$	$\lvert r\rvert > 1$	$\lvert w_t\rvert \to \infty$

Observation: The convergence threshold $\eta < 1/3$ depends on the constant 6 in $f'(w) = 6w$ — the steeper the function, the smaller the step size must be. This is not a coincidence: in Parts 3–4 we will see that

for general functions, the maximum safe step size is determined by how fast the gradient changes, and this threshold generalizes cleanly.

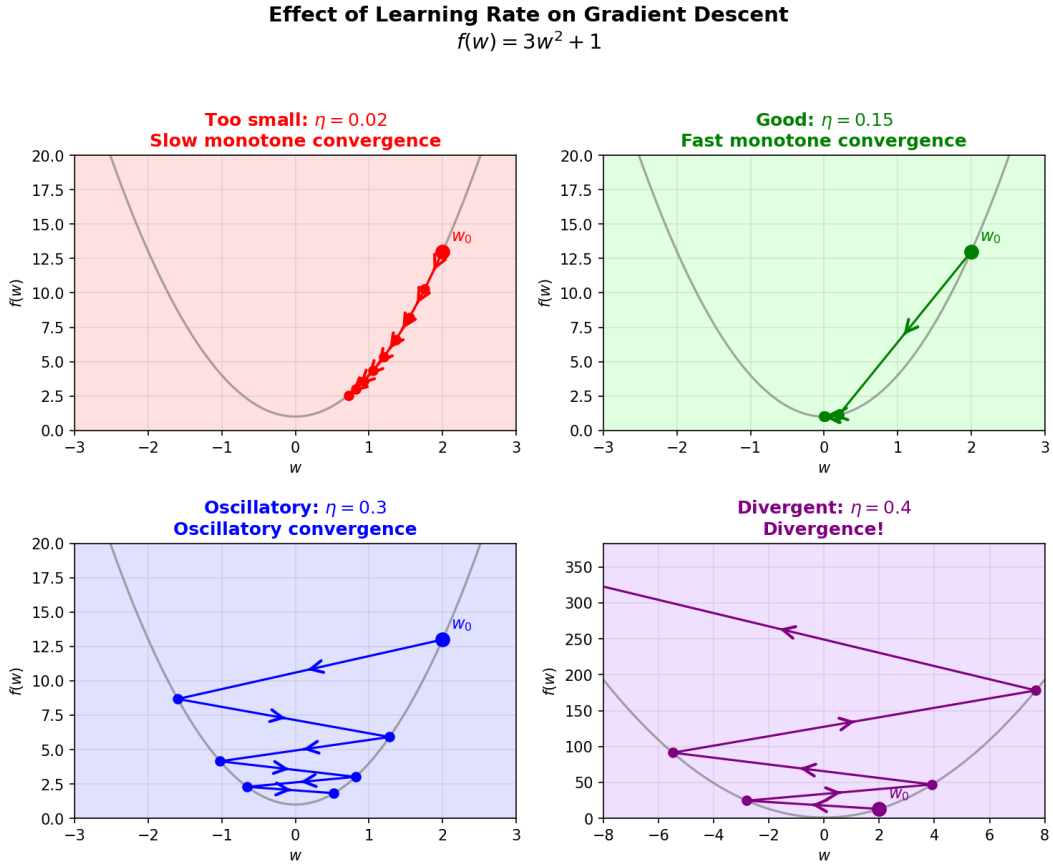

Figure 2: Learning rate comparison

Part 3: Smoothness and the Descent Lemma

We now move from a single example to **general theory**. In this part, we formalize the notion of "smoothness" and prove that GD makes guaranteed progress every step.

3.1 L-Smoothness

Definition. A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is **L-smooth** if, for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$$

In words: the gradient doesn't change too fast. If you take a step of size δ, the gradient can change by at most $L\delta$. This is exactly the condition we need to ensure that the linear approximation from Part 1 remains reasonable for small steps.

Example: For $f(w) = 3w^2 + 1$, the gradient is $f'(w) = 6w$. How fast does this gradient change? If we move from w to w', the gradient changes by $|f'(w) - f'(w')| = |6w - 6w'| = 6|w - w'|$. So the gradient changes by at most 6 times the step size — this function is L-smooth with $L = 6$.

Optional Remark (An additional example: smoothness and Hessian). Now consider $f(x, y) = x^2 + 4y^2$. The gradient is $\nabla f(x, y) = (2x,\, 8y)$. How fast does it change? Moving from (x, y) to (x', y'):

$$\|\nabla f(x, y) - \nabla f(x', y')\| = \|(2(x - x'),\, 8(y - y'))\| = \sqrt{4(x - x')^2 + 64(y - y')^2}$$

We want the smallest L such that this is always $\leq L\|(x - x', y - y')\| = L\sqrt{(x - x')^2 + (y - y')^2}$. The worst case is when all the displacement is in the y-direction (set $x = x'$): then the left side is $8|y - y'|$ and the right side is $L|y - y'|$, so we need $L \geq 8$. And $L = 8$ works in general, since $\sqrt{4a^2 + 64b^2} \leq \sqrt{64a^2 + 64b^2} = 8\sqrt{a^2 + b^2}$. So this function is L-smooth with $L = 8$.

Where did the 8 come from? Notice that for this function, the gradient is a linear function of (x, y):

$$\nabla f(x, y) = \begin{pmatrix} 2 & 0 \\ 0 & 8 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = H \begin{pmatrix} x \\ y \end{pmatrix}$$

where $H = \begin{pmatrix} 2 & 0 \\ 0 & 8 \end{pmatrix}$ is the **Hessian** (matrix of second derivatives) of f. Then $\nabla f(\mathbf{w}) - \nabla f(\mathbf{w}') = H(\mathbf{w} - \mathbf{w}')$, so:

$$\|\nabla f(\mathbf{w}) - \nabla f(\mathbf{w}')\| = \|H(\mathbf{w} - \mathbf{w}')\| \leq \lambda_{\max}(H) \cdot \|\mathbf{w} - \mathbf{w}'\|$$

where $\lambda_{\max}(H)$ is the **largest eigenvalue** of H. For our diagonal matrix, the eigenvalues are 2 and 8, so $L = \lambda_{\max}(H) = 8$.

This is a general principle: for any twice-differentiable function, the smoothness constant L equals the largest eigenvalue of the Hessian (or more precisely, the supremum of the largest eigenvalue over all points, since the Hessian can vary). In the one-variable case, the Hessian is just $f''(w)$, so for $f(w) = 3w^2 + 1$ we get $L = f''(w) = 6$, matching what we found above.

Connection to learning rate: A larger L means the gradient changes faster, so the linear approximation breaks down over shorter distances. This means we need to take **smaller** steps — i.e., use a smaller learning rate η. Below, we will make this intuition precise by showing that setting $\eta = 1/L$ is the right choice: it is large enough to make good progress, but small enough that we never overshoot.

3.2 The Descent Lemma

Let us set up notation for the analysis. Recall from Part 1 that **gradient descent** takes a starting point $\mathbf{w}_0$ and a **learning rate** (step size) $\eta > 0$, and repeatedly updates the parameters:

Gradient Descent

Input: starting point $\mathbf{w}_0$, learning rate η, number of steps T
for $t = 0, 1, 2, \ldots, T - 1$:
  $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla f(\mathbf{w}_t)$
return $\mathbf{w}_T$

This produces a sequence of iterates $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T$. Our goal is to show that $f(\mathbf{w}_t)$ decreases over time and eventually gets close to the minimum. But how should we set the learning rate η? In Part 2 we saw that η must not be too large (or GD diverges) and not too small (or convergence is slow). The descent lemma makes this precise for general L-smooth functions.

The descent lemma is the fundamental building block for the analysis. It answers the question from Part 1: if we choose the step size correctly, does GD actually make progress every step?

The key starting point is that L-smoothness gives us a **quadratic upper bound** on f. For any two points $\mathbf{x}$ and $\mathbf{y}$:

$$f(\mathbf{y}) \leq \underbrace{f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x})}_{\text{linear approximation at } \mathbf{x}} + \underbrace{\frac{L}{2}\|\mathbf{y} - \mathbf{x}\|^2}_{\text{error term}}$$

Why should this be true? Here is the intuition. Recall from Section 1.2 that the linear approximation predicts $f(\mathbf{y}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x})$. This prediction is exact if the gradient were constant everywhere (i.e., if f were a linear function). The error comes from the fact that the gradient **changes** as we move from $\mathbf{x}$ to $\mathbf{y}$. How bad can this error be? At each point along the path from $\mathbf{x}$ to $\mathbf{y}$, the gradient has drifted by at most L times the distance traveled (by L-smoothness). Accumulating this drift over a path of length $\|\mathbf{y} - \mathbf{x}\|$ gives an error of at most $\frac{L}{2}\|\mathbf{y} - \mathbf{x}\|^2$ — the $1/2$ comes from the same reason that integrating a linear function gives a quadratic (just like how distance $= \frac{1}{2}at^2$ for constant acceleration a).

Now we can state and prove the descent lemma.

> **Lemma (Descent Lemma).** If f is L-smooth and we run gradient descent with step size $\eta = 1/L$, then:
>
> $$f(\mathbf{w}_{t+1}) \leq f(\mathbf{w}_t) - \frac{1}{2L}\|\nabla f(\mathbf{w}_t)\|^2$$

Proof. We plug the GD step into the quadratic upper bound. We have $\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{L}\nabla f(\mathbf{w}_t)$, so:

Step 1. Apply the quadratic upper bound with $\mathbf{x} = \mathbf{w}_t$ and $\mathbf{y} = \mathbf{w}_{t+1}$:

$$f(\mathbf{w}_{t+1}) \leq f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top(\mathbf{w}_{t+1} - \mathbf{w}_t) + \frac{L}{2}\|\mathbf{w}_{t+1} - \mathbf{w}_t\|^2$$

Now substitute the GD update $\mathbf{w}_{t+1} - \mathbf{w}_t = -\frac{1}{L}\nabla f(\mathbf{w}_t)$:

$$f(\mathbf{w}_{t+1}) \leq f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top\left(-\frac{1}{L}\nabla f(\mathbf{w}_t)\right) + \frac{L}{2}\left\|\frac{1}{L}\nabla f(\mathbf{w}_t)\right\|^2$$

Step 2. Simplify each term. The dot product gives $-\frac{1}{L}\|\nabla f(\mathbf{w}_t)\|^2$. The squared norm gives $+\frac{1}{2L}\|\nabla f(\mathbf{w}_t)\|^2$.

$$f(\mathbf{w}_{t+1}) \leq f(\mathbf{w}_t) - \frac{1}{L}\|\nabla f(\mathbf{w}_t)\|^2 + \frac{1}{2L}\|\nabla f(\mathbf{w}_t)\|^2 = f(\mathbf{w}_t) - \frac{1}{2L}\|\nabla f(\mathbf{w}_t)\|^2 \qquad \blacksquare$$

What this says: Every step of GD **decreases** the loss by at least $\frac{1}{2L}\|\nabla f(\mathbf{w}_t)\|^2$. Two things to notice: (1) the larger the gradient, the more progress we make — which makes sense, since a large gradient means we're far from a flat region. (2) The step size $\eta = 1/L$ is the "sweet spot" — large enough to make real progress, but small enough that the quadratic error term doesn't overwhelm the linear decrease.

What about other step sizes? The proof above used $\eta = 1/L$, but we can redo the same calculation for a general step size η. Plugging $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\nabla f(\mathbf{w}_t)$ into the quadratic upper bound gives:

$$f(\mathbf{w}_{t+1}) \leq f(\mathbf{w}_t) - \eta\|\nabla f(\mathbf{w}_t)\|^2 + \frac{L\eta^2}{2}\|\nabla f(\mathbf{w}_t)\|^2 = f(\mathbf{w}_t) - \eta\left(1 - \frac{L\eta}{2}\right)\|\nabla f(\mathbf{w}_t)\|^2$$

For this to guarantee descent (i.e., $f(\mathbf{w}_{t+1}) < f(\mathbf{w}_t)$), we need $1 - \frac{L\eta}{2} > 0$, which means $\eta < 2/L$. Compare this with Part 2: for $f(w) = 3w^2 + 1$ with $L = 6$, the threshold $\eta < 2/L = 1/3$ is exactly the convergence condition we found! When $\eta > 2/L$, the quadratic error term dominates and the function value can *increase* — this is exactly the divergence regime from Part 2.

The choice $\eta = 1/L$ maximizes the guaranteed progress $\eta(1 - L\eta/2) = 1/(2L)$, which is why we use it.

A caveat: local vs. global minima. The descent lemma guarantees that GD always decreases f, but it only guarantees convergence to a point where $\nabla f = 0$ — a **stationary point**. For non-convex functions, this could be a local minimum that is not the global minimum, and GD can get stuck there depending on where it starts.

Consider the example $f(w) = w^4 - 3w^2 + w + 3$. This function has two local minima: one near $w \approx -1.3$ (the **global** minimum) and one near $w \approx 1.1$ (a **local** minimum with a higher function value). If GD starts

to the right of the "hill" between them, it will slide into the local minimum and stay there forever — it has no way of knowing that a better solution exists on the other side.

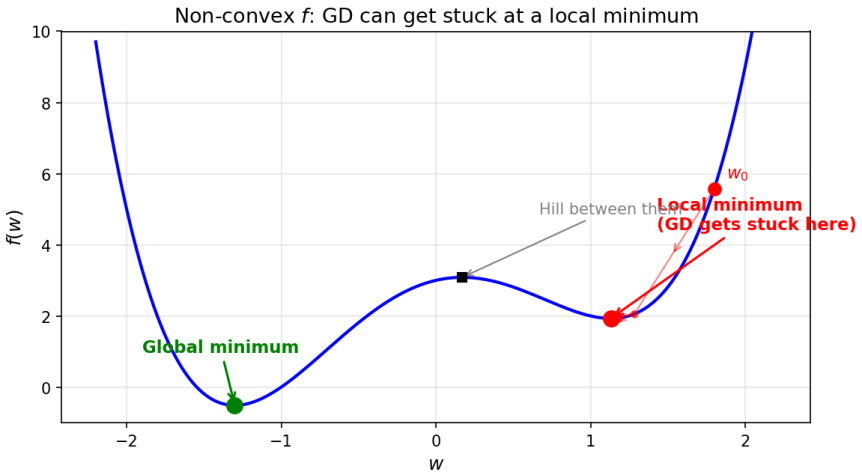

Figure 3: Local vs global minima

This is why **convexity** matters: for convex functions, there is only one valley, so every local minimum is the global minimum. With convexity, the descent lemma is enough to guarantee convergence to the best possible solution. We formalize this in Part 4.

Part 4: Convexity and Convergence

The descent lemma (Part 3) guarantees that GD makes progress every step, but it only ensures convergence to a stationary point — not necessarily the global minimum. To get a global guarantee, we need **convexity**.

4.1 Convexity

Definition. A function f is **convex** if for all $\mathbf{x}, \mathbf{y}$ and $\lambda \in [0, 1]$:

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

Equivalently (for differentiable f): f is convex if and only if

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) \qquad \text{for all } \mathbf{x}, \mathbf{y}$$

Interpretation: The tangent line (or tangent plane) at any point lies **below** the function. Convexity guarantees that every local minimum is a **global** minimum.

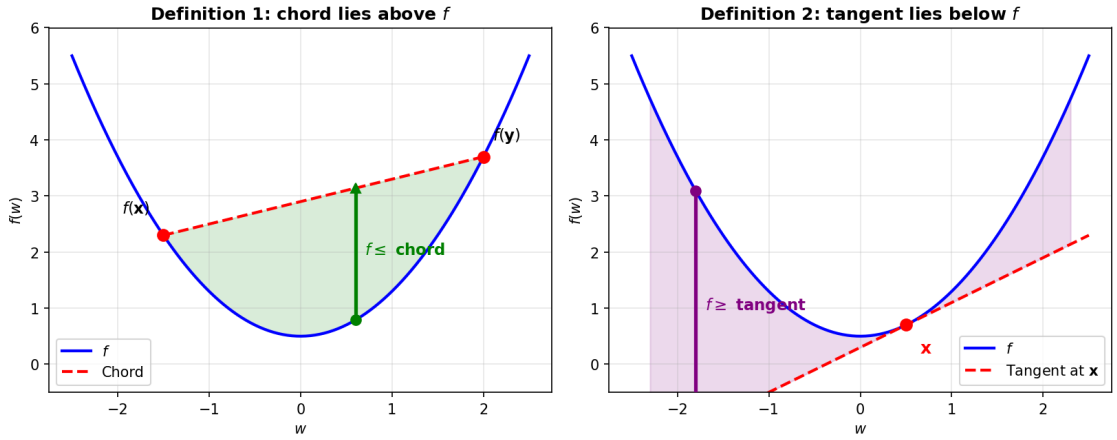

Figure 4: Two definitions of convexity

Why convexity ensures every local minimum is global. Suppose $\mathbf{w}^*$ is a local minimum of a convex function f, but there exists some other point $\mathbf{w}'$ with $f(\mathbf{w}') < f(\mathbf{w}^*)$. By the chord definition of convexity, for any $\lambda \in (0, 1)$:

$$f(\lambda \mathbf{w}' + (1-\lambda)\mathbf{w}^*) \leq \lambda f(\mathbf{w}') + (1-\lambda) f(\mathbf{w}^*) < \lambda f(\mathbf{w}^*) + (1-\lambda) f(\mathbf{w}^*) = f(\mathbf{w}^*)$$

where the strict inequality uses $f(\mathbf{w}') < f(\mathbf{w}^*)$. But the point $\lambda \mathbf{w}' + (1-\lambda)\mathbf{w}^*$ can be made arbitrarily close to $\mathbf{w}^*$ by taking λ very small — so there are points with lower function value in every neighborhood of $\mathbf{w}^*$. This contradicts $\mathbf{w}^*$ being a local minimum. Therefore no such $\mathbf{w}'$ exists, and every local minimum must be a global minimum.

4.2 Convergence Theorem (Smooth + Convex)

The descent lemma tells us that each step makes progress, but how fast does GD actually converge to the optimum? The following theorem gives a clean answer.

> **Theorem.** Let f be convex and L-smooth with minimizer $\mathbf{w}^*$. Gradient descent with step size $\eta = 1/L$ satisfies:
>
> $$f(\mathbf{w}_T) - f(\mathbf{w}^*) \leq \frac{L\|\mathbf{w}_0 - \mathbf{w}^*\|^2}{2T}$$

Reading the bound: The error after T steps shrinks like $1/T$. It depends on two things: L (how smooth the function is — smoother means faster), and $\|\mathbf{w}_0 - \mathbf{w}^*\|$ (how far we started from the optimum). To get the error below some target ε, we need roughly $T = O(L\|\mathbf{w}_0 - \mathbf{w}^*\|^2/\varepsilon)$ steps.

Proof idea. The key insight is that each GD step makes progress that we can measure in two ways: the function value decreases (descent lemma), and we get closer to $\mathbf{w}^*$ (convexity). Combining these gives a "budget" argument.

The per-step bound. We want to show that at each step, the error $f(\mathbf{w}_{t+1}) - f(\mathbf{w}^*)$ is bounded by how much closer we moved to $\mathbf{w}^*$:

$$f(\mathbf{w}_{t+1}) - f(\mathbf{w}^*) \leq \frac{L}{2}\|\mathbf{w}_t - \mathbf{w}^*\|^2 - \frac{L}{2}\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2$$

We prove this by combining two ingredients. First, convexity says the tangent line at $\mathbf{w}_t$ lies below f, so $f(\mathbf{w}^*) \geq f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top (\mathbf{w}^* - \mathbf{w}_t)$, which rearranges to:

$$f(\mathbf{w}_t) - f(\mathbf{w}^*) \leq \nabla f(\mathbf{w}_t)^\top (\mathbf{w}_t - \mathbf{w}^*) \qquad \text{(convexity)}$$

Second, we expand the squared distance after one GD step. Since $\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{1}{L}\nabla f(\mathbf{w}_t)$:

$$\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 = \|\mathbf{w}_t - \mathbf{w}^*\|^2 - \frac{2}{L}\nabla f(\mathbf{w}_t)^\top(\mathbf{w}_t - \mathbf{w}^*) + \frac{1}{L^2}\|\nabla f(\mathbf{w}_t)\|^2$$

(This is just expanding $\|a-b\|^2 = \|a\|^2 - 2a^\top b + \|b\|^2$ with $a = \mathbf{w}_t - \mathbf{w}^*$ and $b = \frac{1}{L}\nabla f(\mathbf{w}_t)$.) Solving for the dot product and plugging into the convexity bound, we get:

$$f(\mathbf{w}_t) - f(\mathbf{w}^*) \le \frac{L}{2}\|\mathbf{w}_t - \mathbf{w}^*\|^2 - \frac{L}{2}\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 + \frac{1}{2L}\|\nabla f(\mathbf{w}_t)\|^2$$

The last term is the leftover gradient norm. But the descent lemma tells us exactly that $\frac{1}{2L}\|\nabla f(\mathbf{w}_t)\|^2 \le f(\mathbf{w}_t) - f(\mathbf{w}_{t+1})$. Substituting this in and canceling $f(\mathbf{w}_t)$ from both sides gives the clean per-step bound above.

The budget argument. Now we just add up. Summing the per-step bound over $t = 0, 1, \ldots, T-1$:

$$\sum_{t=0}^{T-1}\big(f(\mathbf{w}_{t+1}) - f(\mathbf{w}^*)\big) \le \frac{L}{2}\|\mathbf{w}_0 - \mathbf{w}^*\|^2 - \frac{L}{2}\|\mathbf{w}_T - \mathbf{w}^*\|^2$$

The right side **telescopes**: all intermediate $\|\mathbf{w}_t - \mathbf{w}^*\|^2$ terms cancel, leaving only the first and last. Dropping the negative last term (which only makes the bound weaker):

$$\sum_{t=0}^{T-1}\big(f(\mathbf{w}_{t+1}) - f(\mathbf{w}^*)\big) \le \frac{L}{2}\|\mathbf{w}_0 - \mathbf{w}^*\|^2$$

Think of $\frac{L}{2}\|\mathbf{w}_0 - \mathbf{w}^*\|^2$ as a **budget** — the total amount of error that GD can accumulate across all steps. Since f is non-increasing (descent lemma), the last iterate $f(\mathbf{w}_T)$ is the smallest, so every term in the sum is at least $f(\mathbf{w}_T) - f(\mathbf{w}^*)$. Therefore:

$$T\big(f(\mathbf{w}_T) - f(\mathbf{w}^*)\big) \le \frac{L}{2}\|\mathbf{w}_0 - \mathbf{w}^*\|^2$$

Dividing by T: $f(\mathbf{w}_T) - f(\mathbf{w}^*) \le \dfrac{L\|\mathbf{w}_0 - \mathbf{w}^*\|^2}{2T}$. ■

Part 5: Stochastic Gradient Descent

5.1 The Key Idea

Suppose our objective has the form of a **sum**:

$$f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{w})$$

The full gradient is then also an average: $\nabla f(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\mathbf{w})$. Computing this requires evaluating all n terms — expensive when n is large!

Stochastic gradient descent (SGD) uses a simple trick: instead of computing the full average, **pick a random index** $i_t \in \{1, \ldots, n\}$ and use $\nabla f_{i_t}(\mathbf{w}_t)$ as a stand-in for the full gradient:

Stochastic Gradient Descent (SGD)

Input: starting point $\mathbf{w}_0$, learning rate η, number of steps T
for $t = 0, 1, 2, \ldots, T-1$:
  Sample i_t uniformly at random from $\{1, \ldots, n\}$
  $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla f_{i_t}(\mathbf{w}_t)$
return $\mathbf{w}_T$

Why is this reasonable? Because the random gradient is an **unbiased estimator** of the true gradient:

$$\mathbb{E}_{i_t}\left[\nabla f_{i_t}(\mathbf{w}_t)\right] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{w}_t) = \nabla f(\mathbf{w}_t)$$

On average, SGD points in the right direction — each individual step is noisy, but over many steps the noise averages out. The cost per step drops from $O(n)$ to $O(1)$.

5.2 Application to Machine Learning

Where do sums like this come from? The most important example is **supervised machine learning**. We have a dataset of n training examples $(x_1, y_1), \ldots, (x_n, y_n)$, where each x_i is an input (e.g., an image or a sentence) and y_i is the desired output (e.g., a label). Our model, parameterized by weights $\mathbf{w}$, makes a prediction on each input x_i, and the **per-example loss** $f_i(\mathbf{w})$ measures how far off the prediction is — for instance, $f_i(\mathbf{w}) = (\text{prediction}_i - y_i)^2$.

The overall loss $f(\mathbf{w}) = \frac{1}{n}\sum_i f_i(\mathbf{w})$ is exactly the average over training examples. Modern datasets can have n in the millions or billions, so computing the full gradient at every step is prohibitively expensive. SGD makes training feasible: each step only looks at **one** randomly chosen example.

5.3 Mini-Batch SGD

In practice, using a single example per step is too noisy. Instead, we sample a random **mini-batch** $\mathcal{B}_t \subset \{1, \ldots, n\}$ of size B and average their gradients:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \cdot \frac{1}{B}\sum_{i \in \mathcal{B}_t} \nabla f_i(\mathbf{w}_t)$$

This is still unbiased, but the variance of the gradient estimate decreases by a factor of B (by the same reason that averaging B independent samples reduces variance). The cost per step is $O(B)$ instead of $O(n)$. Typical values: $B = 32, 64, 128, 256$.

Part 6: Further Reading and Video Lectures

6.1 Video Lectures

- **Gilbert Strang (MIT 18.065, Lecture 22):** "Gradient Descent: Pair $S = A^\top A$." Clear treatment of gradient descent for quadratics, condition numbers, and convergence. MIT OCW
- **Andrew Ng (Stanford CS229):** Machine learning course with excellent coverage of gradient descent and SGD in the context of linear regression and neural networks. YouTube
- **Kilian Weinberger (Cornell CS4780):** "Machine Learning for Intelligent Systems" — thorough treatment of optimization for ML, including convergence proofs. YouTube

6.2 Notes and Textbooks

- **Stanford CS231n:** Optimization notes — practical guide to SGD, momentum, Adam, and learning rate schedules. cs231n.github.io
- **MIT 6.390 (Introduction to Machine Learning):** Clear exposition of gradient descent with convergence analysis. mit.edu
- **Ryan Tibshirani (CMU):** Convex Optimization course notes — rigorous treatment of the descent lemma, convergence rates, and strong convexity. stat.cmu.edu
- **Google ML Crash Course:** Practical introduction to gradient descent with interactive visualizations. developers.google.com