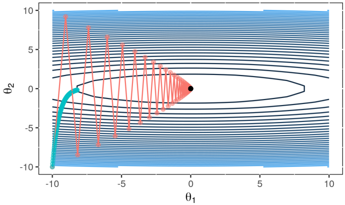
Optimization in Machine Learning

First order methods: Step size and optimality



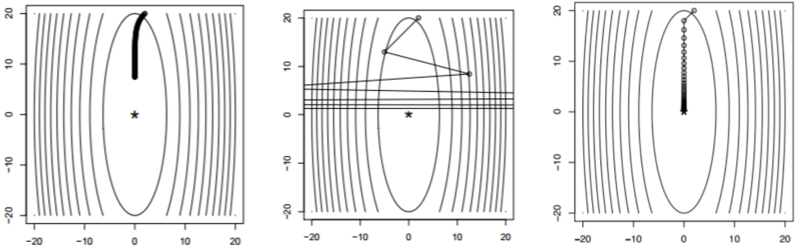
Learning goals

- Impact of step size
- Fixed vs. adaptive
- Exact line search
- Armijo rule and backtracking

CONTROLLING STEP SIZE: FIXED & ADAPTIVE

In every iteration t, we need to choose not only a descent direction $\mathbf{d}^{[t]}$, but also a step size $\alpha^{[t]}$:

- If $\alpha^{[t]}$ is too small, the procedure may converge very slowly (left).
- If $\alpha^{[t]}$ is too large, the procedure may not converge, because we "jump" around the optimum (right). Use fixed step size α in each iteration: $\alpha^{[t]} = \alpha$

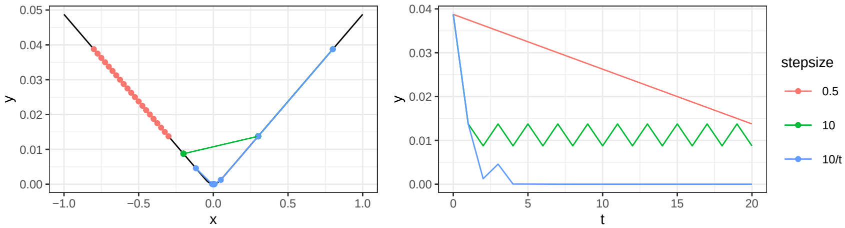


Steps of a line search for $f(\boldsymbol{x}) = 10x_1^2 + 0.5x_2^2$, left 100 steps with fixed step size, right only 40 steps with adaptively selected step size.

STEP SIZE CONTROL: DIMINISHING STEP SIZE

How can we adaptively control step size?

- A natural way of selecting α is to decrease its value over time



Example: GD on $f(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| \leq \delta \\ \delta \cdot (|x| - 1/2 \cdot \delta) & \text{otherwise,} \end{cases}$

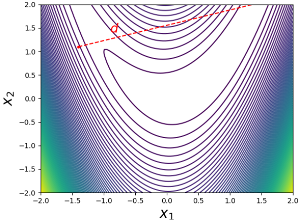
with constant (small) step size, constant (large) step size, and diminishing step size $\alpha^{[t]} = \frac{1}{t}$, with t being the iteration of GD.

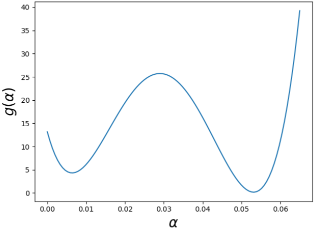
STEP SIZE CONTROL: EXACT LINE-SEARCH

Use **optimal** step size in each iteration:

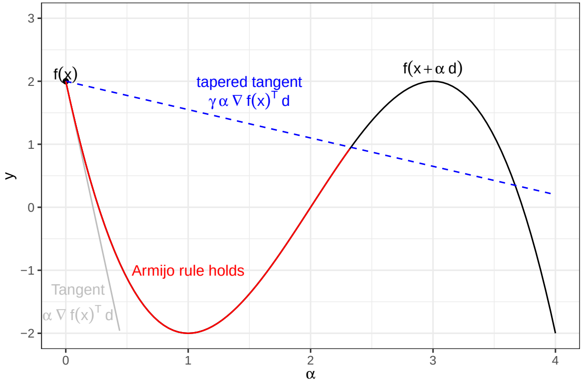
$$\alpha^{[t]} = \arg\min_{\alpha \in \mathbb{R}_{\geq 0}} g(\alpha) = \arg\min_{\alpha \in \mathbb{R}_{\geq 0}} f(\boldsymbol{x}^{[t]} + \alpha \mathbf{d}^{[t]})$$

In each iter solve an **univariate optimization problem** $\arg\min g(\alpha)$ (e.g. via golden ratio). Problem: Expensive, prone to poorly conditioned problems



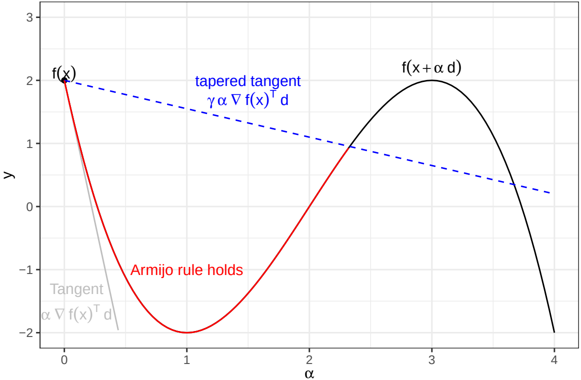


ARMIJO RULE



Inext line search: Efficient procedures to minimize objective "sufficiently", without computing optimal step size exactly. Common condition to ensure objective decreases "sufficiently": **Armijo rule**.

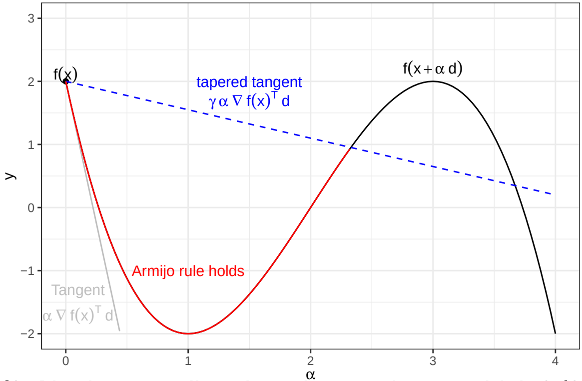
ARMIJO RULE



α satisfies **Armijo rule** in $\mathbf{x}$ for descent direction $\boldsymbol{d}$ if for fixed $\gamma \in (0, 1)$:

$$f(\mathbf{x} + \alpha \boldsymbol{d}) \leq f(\mathbf{x}) + \gamma \alpha \nabla f(\mathbf{x})^\top \boldsymbol{d}.$$

ARMIJO RULE



Feasibility: If ***d*** is descent direction, there exists α which fulfulls Armijo rule for each $\gamma \in (0, 1)$. In many cases, the Armijo rule guarantees local convergence of GD and is therefore frequently used.

BACKTRACKING LINE SEARCH

Backtracking line search is a procedure to meet the Armijo rule.

Idea: Decrease α until the Armijo rule is met.

Algorithm 1 Backtracking line search

1: Choose initial step size $\alpha = \alpha^{[0]}$, $0 < \gamma < 1$ and $0 < \tau < 1$
2: **while** $f(\mathbf{x} + \alpha \mathbf{d}) > f(\mathbf{x}) + \gamma \alpha \nabla f(\mathbf{x})^\top \mathbf{d}$ **do**
3: Decrease α: $\alpha \leftarrow \tau \cdot \alpha$
4: **end while**

The procedure is simple and shows good performance in practice.

GRADIENT DESCENT AND OPTIMALITY

- GD is a greedy algorithm: In every iteration, it makes locally optimal moves.
- If $\mathcal{R}(\boldsymbol{\theta})$ is **convex** and **differentiable**, and its gradient is Lipschitz continuous, GD is guaranteed to converge to the global minimum for small enough step-size.