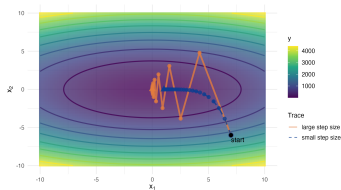
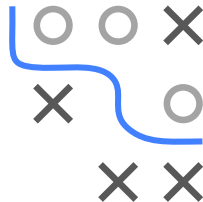


# Optimization in Machine Learning

## First order methods

## Step size and optimality

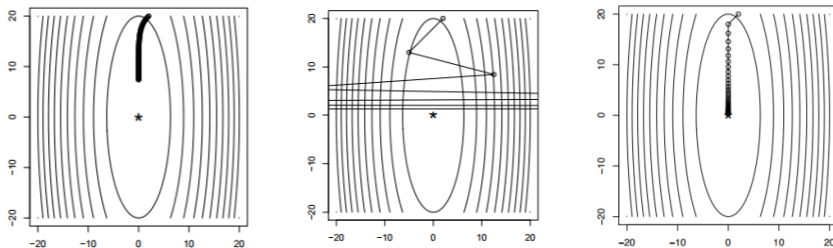


### Learning goals

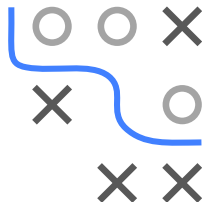
- Impact of step size
- Fixed vs. adaptive step size
- Exact line search
- Armijo rule & Backtracking
- Bracketing & Pinpointing

## CONTROLLING STEP SIZE: FIXED & ADAPTIVE

- Iteration  $t$ : Choose not only descent direction  $\mathbf{d}^{[t]}$ , but also step size  $\alpha^{[t]}$
- First approach: **Fixed** step size  $\alpha^{[t]} = \alpha > 0$ 
  - If  $\alpha$  too small, procedure may converge very slowly (left)
  - If  $\alpha$  too large, procedure may not converge  $\rightarrow$  “jumps” around optimum (middle)
- **Adaptive** step size  $\alpha^{[t]}$  can provide better convergence (right)



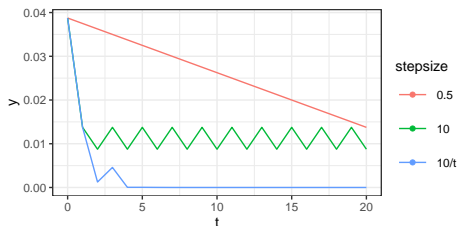
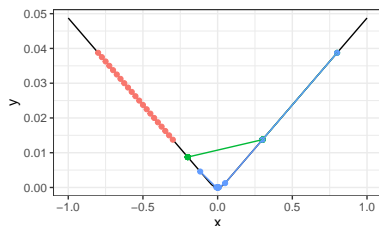
Steps of line searches for  $f(\mathbf{x}) = 10x_1^2 + x_2^2/2$



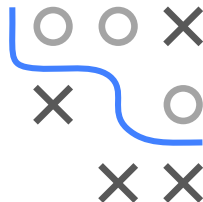
# STEP SIZE CONTROL: DIMINISHING STEP SIZE

- How can we adaptively control step size?
- A natural way of selecting  $\alpha^{[t]}$  is to decrease its value over time
- **Example:** GD on

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



GD with small constant (**red**), large constant (**green**), and diminishing (**blue**) step size

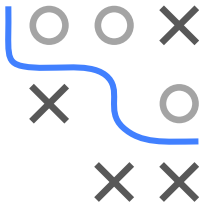
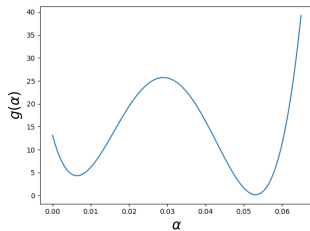
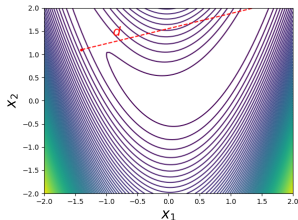


## STEP SIZE CONTROL: EXACT LINE SEARCH

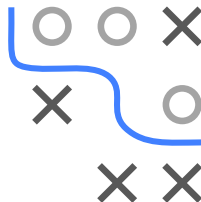
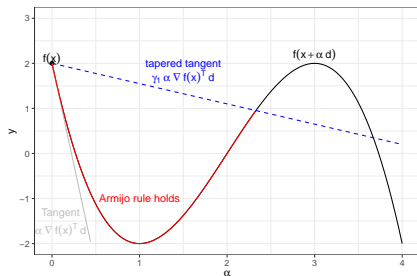
- Use **optimal** step size in each iteration:

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

- Need to solve a **univariate** optimization problem in each iteration  $\Rightarrow$  univariate optimization methods
- **Problem:** Expensive, prone to poorly conditioned problems
- **But:** No need for *optimal* step size. Only need a step size that is “good enough”.
- **Reason:** Effort may not pay off, but in some cases slows down performance.

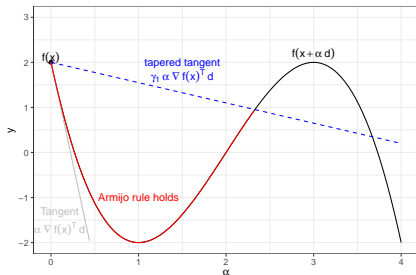


# ARMIJO RULE

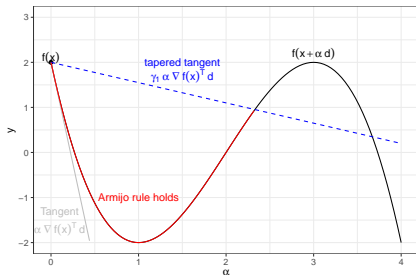


- **Inexact line search:** Minimize objective “sufficiently” without computing optimal step size exactly
- Common condition to guarantee “sufficient” decrease: **Armijo rule**

A 3x3 grid with a blue path starting at the top-left cell (0,0) and ending at the bottom-right cell (2,2). The path consists of the following cells: (0,0), (0,1), (1,1), (1,2), and (2,2). The cells (0,1), (0,2), (1,0), and (2,0) are marked with a grey 'X'. The cells (1,0) and (2,1) are marked with a grey circle.



- $$f(\mathbf{x} + \alpha \mathbf{d}) \leq f(\mathbf{x}) + \gamma_1 \alpha \nabla f(\mathbf{x})^\top \mathbf{d}.$$
- **Note:**  $\nabla f(\mathbf{x})^\top \mathbf{d} < 0$  ( $\mathbf{d}$  descent dir.)  $\implies f(\mathbf{x} + \alpha \mathbf{d}) < f(\mathbf{x})$ .



- ©

# BACKTRACKING LINE SEARCH

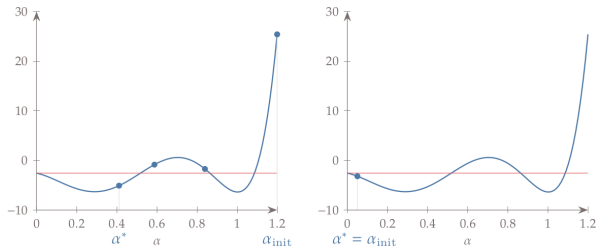
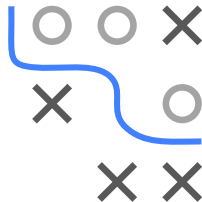
- Procedure to meet the Armijo rule: **Backtracking** line search
- **Idea:** Decrease  $\alpha$  until Armijo rule is met

---

## Algorithm Backtracking line search

---

- 1: Choose initial step size  $\alpha = \alpha_{\text{init}}$ ,  $0 < \gamma_1 < 1$  and  $0 < \tau < 1$
  - 2: **while**  $f(\mathbf{x} + \alpha \mathbf{d}) > f(\mathbf{x}) + \gamma_1 \alpha \nabla f(\mathbf{x})^\top \mathbf{d}$  **do**
  - 3:     Decrease  $\alpha$ :  $\alpha \leftarrow \tau \cdot \alpha$
  - 4: **end while**
- 



► [Click for source](#)

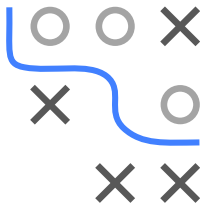


# WOLFE CONDITIONS

- Backtracking is simple and shows good performance in practice
- **But:** Two undesirable scenarios
  - ❶ Initial step size  $\alpha_{\text{init}}$  is too large  $\Rightarrow$  need multiple evaluations of  $f$
  - ❷ Step size is too small with highly negative slopes
- **Solution** for small step sizes:
  - Fix  $\gamma_2$  with  $0 < \gamma_1 < \gamma_2 < 1$ .
  - $\alpha$  satisfies **sufficient curvature condition** in  $\mathbf{x}$  for  $\mathbf{d}$  if

$$|\nabla f(\mathbf{x} + \alpha \mathbf{d})^\top \mathbf{d}| \leq \gamma_2 |\nabla f(\mathbf{x})^\top \mathbf{d}|.$$

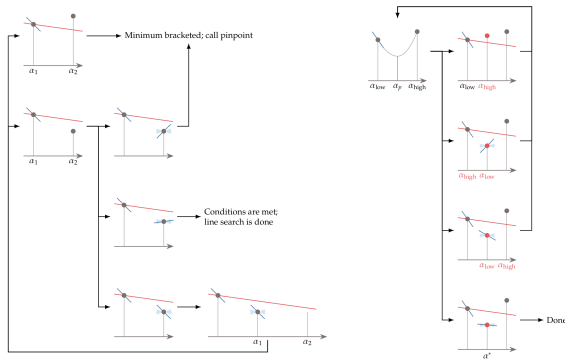
Armijo rule + sufficient curvature condition = **Wolfe conditions**



# WOLFE CONDITIONS

- **Algorithm** for finding a Wolfe point (point satisfying Wolfe conditions):

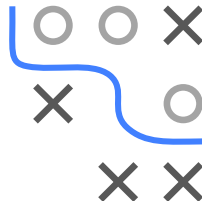
- 1 **Bracketing**: Find interval containing Wolfe point
- 2 **Pinpointing**: Find Wolfe point in interval from bracketing



► Click for source

► Click for source

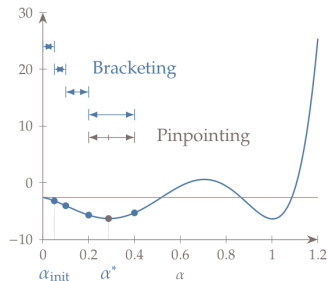
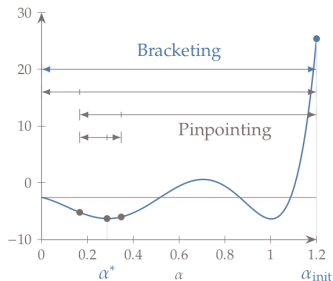
**Left: Bracketing. Right: Pinpointing.**



## ©

- **Example:**

- Large initial step size results in quick bracketing but multiple pinpointing steps (**left**).
- Small initial step size results in multiple bracketing steps but quick pinpointing (**right**).



► [Click for source](#)

