Lecture 13: Gradient descent

Ciaran Evans

Recap: optimization

Possibilities so far

- Derivatives are hard / expensive to find (or we don't want to calculate them)
  - Derivative-free optimization!
- Derivatives can be calculated and lead to a closed-form solution
  - Example: the usual linear regression model

Another possibility

- Derivatives can be calculated, but there is no closed-form solution to the system
  - Example: logistic regression

Today: Begin iterative procedures using derivative information

When is there no closed form?

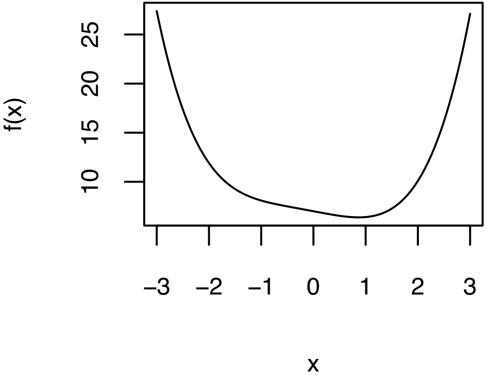
Answer: almost always! A few examples:

- **Nonlinear least squares:** Minimize $L(\beta) = \sum_{i=1}^{n} (Y_i - m(X_i, \beta))^2$ for some nonlinear function m
- **Logistic regression:** Minimize $L(\beta) = -\sum_{i=1}^{n} \left\{ Y_i(\beta_0 + \beta_1 X_i) - \log(1 + e^{\beta_0 + \beta_1 X_i}) \right\}$
- **Robust regression:** Minimize $L(\beta) = \sum_{i=1}^{n} \rho(Y_i - \beta_0 - \beta_1 X_i)$ where

$$\rho(Y_i - \beta_0 - \beta_1 X_i) = \begin{cases} \frac{1}{2}(Y_i - \beta_0 - \beta_1 X_i)^2 & |Y_i - \beta_0 - \beta_1 X_i| \le \gamma \\ \gamma |Y_i - \beta_0 - \beta_1 X_i| - \frac{1}{2}\gamma^2 & \text{else} \end{cases}$$

A univariate example

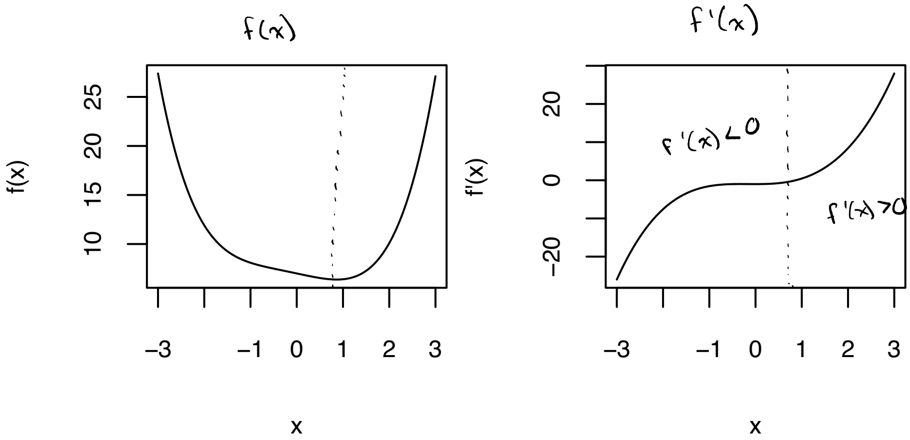
$$f(x) = \frac{x^4}{4} - \sin(x) + 7$$



Want to minimize f. Derivative:

$f'(x) = x^3 - \cos(x) \overset{\text{set}}{=} 0?$ But no closed form!

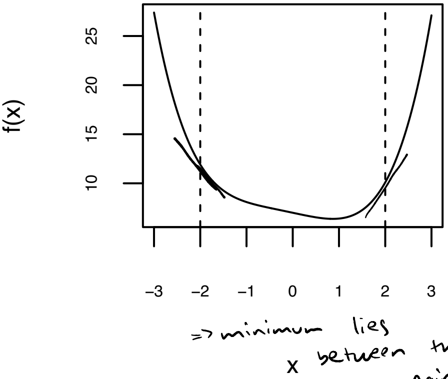
Bisection method



Idea: look for sign changes in the derivative

Bisection

Start with initial interval that contains the sign change in the derivative:



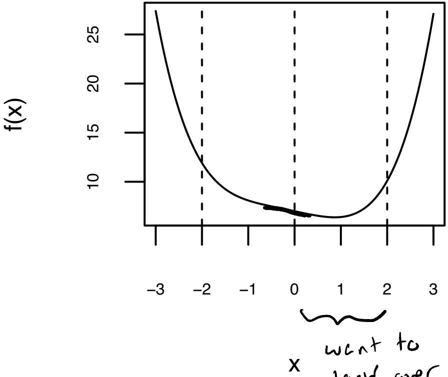
=> minimum lies between these points

Initial interval: $[a_0, b_0]$

- $a_0 = -2,\ b_0 = 2$
- $f'(a_0) < 0,\ f'(b_0) > 0$

Bisection

Calculate the midpoint of the interval:



want to look over here

Initial interval: $[a_0, b_0]$

- $a_0 = -2,\ b_0 = 2$
- $f'(a_0) < 0,\ f'(b_0) > 0$

Midpoint: $x_0 = \dfrac{a_0 + b_0}{2}$

- $x_0 = 0$

$f'(x_0) < 0$

$f'(b_0) > 0$

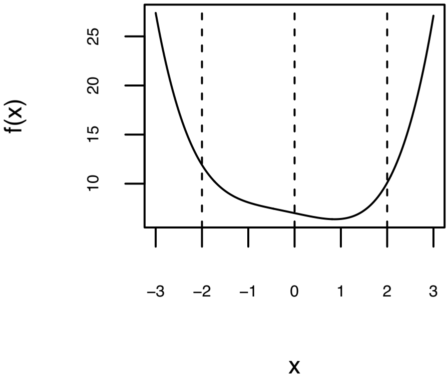
=> new interval:

$[x_0, b_0]$

Question: Where should we look next?

Bisection

If $\text{sign}(f'(x_0)) = \text{sign}(f'(a_0))$, update the interval to $[a_1, b_1] = [x_0, b_0]$. Otherwise, update the interval to $[a_1, b_1] = [a_0, x_0]$



Initial interval: $[a_0, b_0]$

- $a_0 = -2$, $b_0 = 2$
- $f'(a_0) < 0$, $f'(b_0) > 0$

Midpoint: $x_0 = \dfrac{a_0 + b_0}{2}$

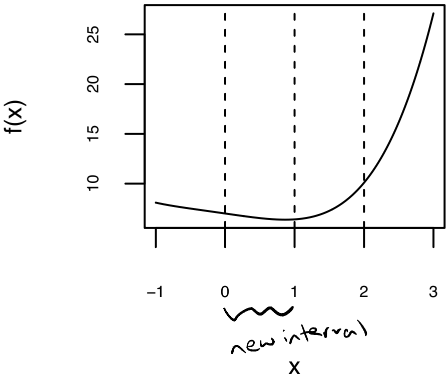
- $x_0 = 0$
- $f'(x_0) < 0$

New interval: $[a_1, b_1]$

- $a_1 = 0$
- $b_1 = 2$

Bisection

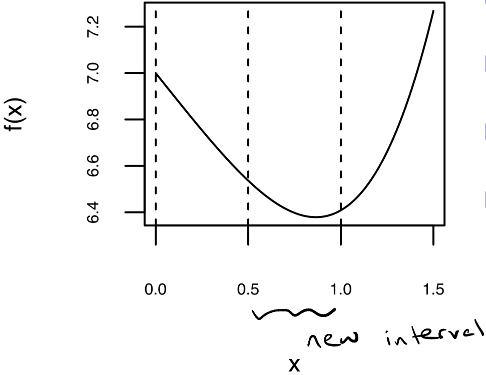
Now iterate:



- $[a_0, b_0] = [-2, 2]$
- $[a_1, b_1] = [0, -2]$
- $[a_2, b_2] = [0, 1]$

Bisection

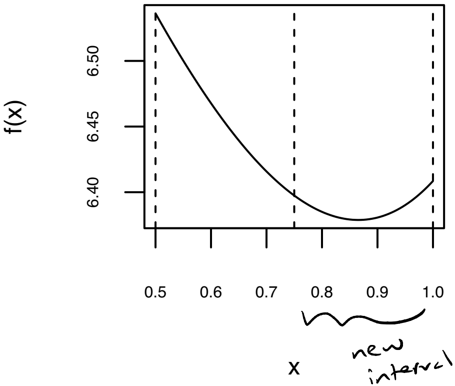
Now iterate:



- $[a_0, b_0] = [-2, 2]$
- $[a_1, b_1] = [0, -2]$
- $[a_2, b_2] = [0, 1]$
- $[a_3, b_3] = [0.5, 1]$

Bisection

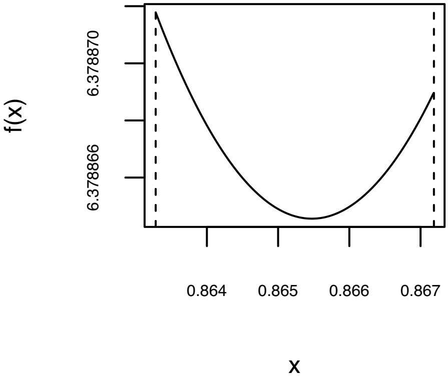
Now iterate:



- $[a_0, b_0] = [-2, 2]$
- $[a_1, b_1] = [0, -2]$
- $[a_2, b_2] = [0, 1]$
- $[a_3, b_3] = [0.5, 1]$
- $[a_4, b_4] = [0.75, 1]$

Bisection

Now iterate:



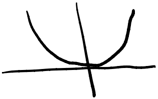
- $[a_0, b_0] = [-2, 2]$
- $[a_1, b_1] = [0, -2]$
- $[a_2, b_2] = [0, 1]$
- $[a_3, b_3] = [0.5, 1]$
- $[a_4, b_4] = [0.75, 1]$

...

- $[a_{10}, b_{10}] = [0.8632812, 0.8671875]$

Bisection method

Bisection failing: find root of x^2



no sign change

(doesn't apply if we assume there is some minimum)

Advantages:

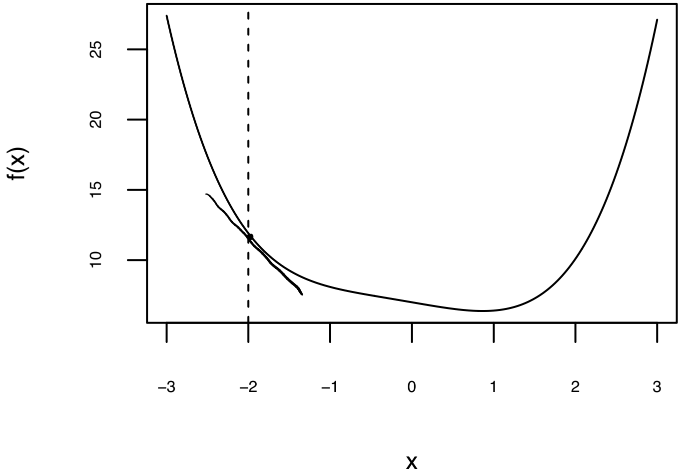
- easy to implement
- easy to analyze
- robust - works very well without too many assumptions

Disadvantages:

- may struggle w/ local minima (true for most methods)
- generally slow
- Somewhat more complex in higher dimensions

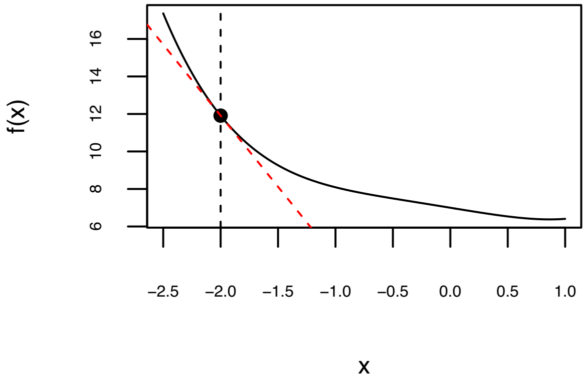
Another approach

Suppose we are at $x = -2$:



Question: In which direction should we move to try and find the minimum? move right – downhill!

Gradient descent: move downhill



Initial guess: $x^{(0)} = -2$

Gradient: $f'(x^{(0)}) = -7.584$

Updated guess: $x^{(1)} = x^{(0)} - \alpha f'(x^{(0)})$

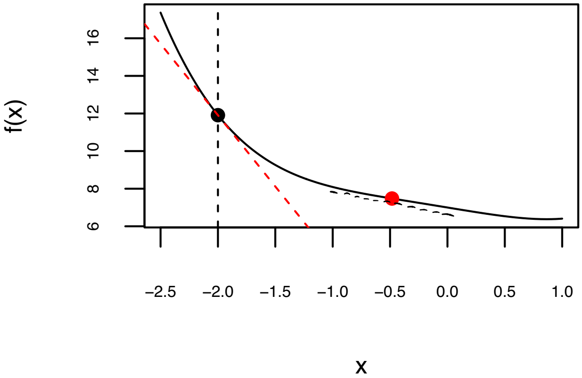
step size

$\alpha > 0$

if $f'(x^{(0)}) < 0$, move right

else move left

Gradient descent: move downhill



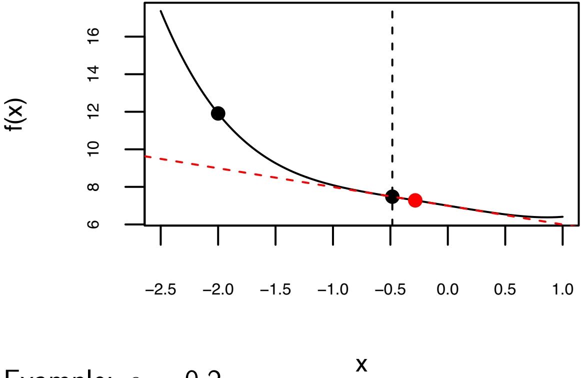
Example: $\alpha = 0.2$

- $x^{(0)} = -2$
- $x^{(1)} = x^{(0)} - \alpha f'(x^{(0)}) = -0.4832$

Question: What should we do next?

Gradient descent: move downhill

Iterate!



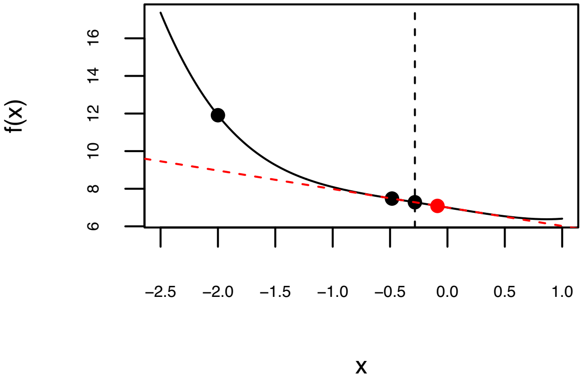
Example: $\alpha = 0.2$

- $x^{(1)} = -0.4832$
- $x^{(2)} = x^{(1)} - \alpha f'(x^{(1)}) = -0.2836$

Question: Why did we move further on the first step than the second? $|f'(x^{(1)})| < |f'(x^{(0)})|$

Gradient descent: move downhill

Iterate!



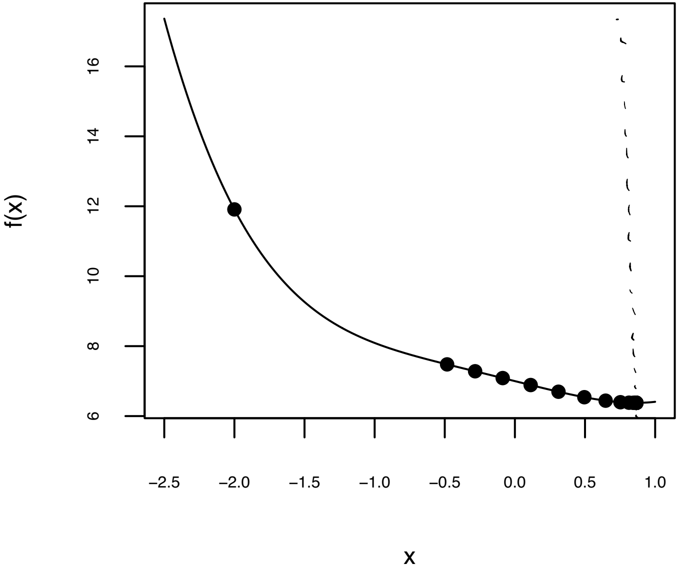
Example: $\alpha = 0.2$

- $x^{(2)} = -0.2836$
- $x^{(3)} = x^{(2)} - \alpha f'(x^{(2)}) = -0.0870$

Gradient descent: move downhill

as $x^{(k)}$ get close to minimum, $f'(x^{(k)})$ gets closer to 0, and we move smaller amounts

After 10 iterations



Gradient descent: the step size

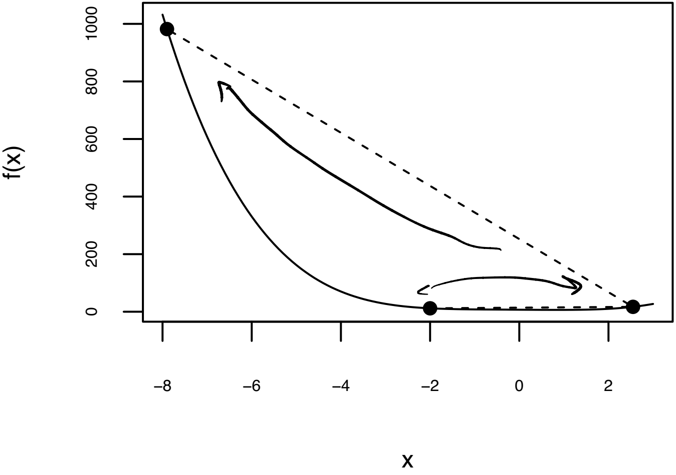
- Specify **step size** $\alpha > 0$
- Update: $x^{(k+1)} = x^{(k)} - \alpha f'(x^{(k)})$

Questions:

- What would happen if α is too *big*?
- What would happen if α is too *small*?

When the step size is too big

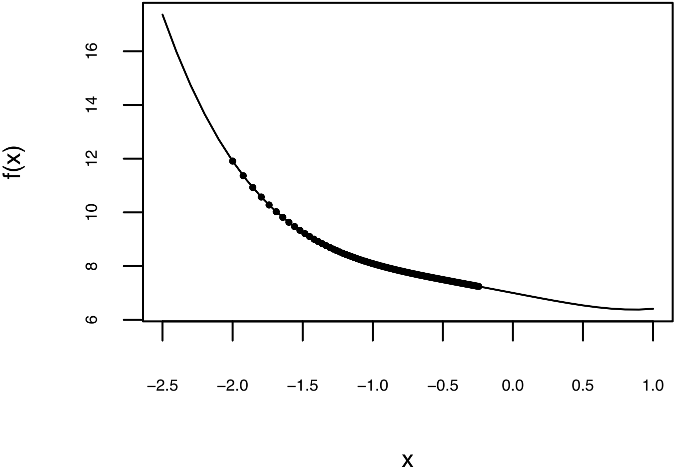
The sequence diverges when α is too large. Using a step size of $\alpha = 0.6$ with the previous example:



$x^{(3)} = -8,\ x^{(4)} = 288, \ldots$

When the step size is too small

When α is too small, the process takes a **long** time. Using a step size of $\alpha = 0.01$ in the previous example:



$x^{(99)} = -0.2537, \ x^{(100)} = -0.2439, \ldots$

Choosing a step size

- Choosing an appropriate step size is important to actually optimize the function
- **Next week:** discuss methods for selecting step size and modifications
  - Line search
  - Adaptive step size methods
- **Future:** Using second-derivative information

Gradient descent in more dimensions

- Points $\mathbf{x} = (x_1, ..., x_d)^T \in \mathbb{R}^d$
- $f(\mathbf{x}) \in \mathbb{R}$
- Gradient:

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{pmatrix} \in \mathbb{R}^d$$

- $\alpha > 0$

Same idea:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)})$$

Example

- $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$
- $f(\mathbf{x}) = 5x_1^2 + 0.5x_2^2$

$$\nabla f(\mathbf{x}) = \begin{pmatrix} 10x_1 \\ x_2 \end{pmatrix}$$

Example

- $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, $f(\mathbf{x}) = 5x_1^2 + 0.5x_2^2$
- $\nabla f(\mathbf{x}) = \begin{pmatrix} 10x_1 \\ x_2 \end{pmatrix}$

Suppose $\mathbf{x}^{(0)} = \begin{pmatrix} 1 \\ 20 \end{pmatrix}$ and $\alpha = 0.1$

$$\nabla f(\mathbf{x}^{(0)}) = \begin{pmatrix} 10 \cdot 1 \\ 20 \end{pmatrix} = \begin{pmatrix} 10 \\ 20 \end{pmatrix}$$

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)})$$

$$\begin{pmatrix} 1 \\ 20 \end{pmatrix} - 0.1 \begin{pmatrix} 10 \\ 20 \end{pmatrix} = \begin{pmatrix} 0 \\ 18 \end{pmatrix}$$

Homework 4

https://sta379-s25.github.io/homework/hw4.html

- Implementation of compass search and modifications, in C++
- Weighted least squares
- Gradient descent for robust regression
- Note: there are many questions, but most of them are quite short