# CSC338 Winter 2022

## Week 10 - Nonlinear optimization

Ilir Dema

University of Toronto

Mar 25, 2022

# What is optimization?

- Consider a smooth function $f : \mathbb{R}^n \to \mathbb{R}$, and a set $S$, also defined using a set of equations and/or inequalities.
  - Example: $S = \{(x, y, z) | 3x + 2y - 4z = 0\}$, a plane perpendicular to $\begin{bmatrix} 3 & 2 & -4 \end{bmatrix}^T$
- We require $x^*$ such that $f(x^*) = \min_{x \in S} f(x)$.
- Seeking $\max f(x)$ is equivalent to seeking $\min(-f(x))$.
- Minima can be local or global.
- The function $f$ can be linear or nonlinear.

# Unconstrained Optimization

## Golden Section Search

- Find a local minimum of $f : \mathbb{R} \to \mathbb{R}$, $S = \mathbb{R}$
- If $f$ is unimodal on $[a, b]$ (a.k.a has a unique minimum) then we can iteratively shrink the interval in which the minima $x^\star$ lies in

# Golden Search

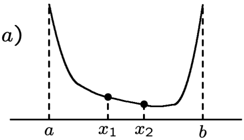
$\tau = (\sqrt{5} - 1)/2$

$x_1 = a + (1 - \tau)(b - a)$

$f_1 = f(x_1)$

$x_2 = a + \tau(b - a)$

$f_2 = f(x_2)$




**while** $((b - a) > tol)$ **do**

  **if** $(f_1 > f_2)$ **then**

    $a = x_1$

    $x_1 = x_2$

    $f_1 = f_2$

    $x_2 = a + \tau(b - a)$

    $f_2 = f(x_2)$

  **else**

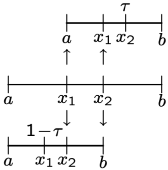
    $b = x_2$

    $x_2 = x_1$

    $f_2 = f_1$

    $x_1 = a + (1 - \tau)(b - a)$

    $f_1 = f(x_1)$



# Unconstrained Optimization

## Newton's method

- Find a local minimum of $f : \mathbb{R} \to \mathbb{R}$, $S = \mathbb{R}$
- We seek a unique minimum around a point $x = a$ (assuming it exists)
- Let $f(x) = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \text{h.o. terms}$ by Talyor's expansion
- Approximate $f(x) \approx g(x) = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2$
- Please note $g(x)$ is a quadratic function approximating $f(x)$ around $a$. It's minimum is found at $g'(x) = f'(a) + f''(a)(x-a) = 0$ so $x = a - \frac{f'(a)}{f''(a)}$.
- Hence the iterative formula is $x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$.

# Newton's method - multivariate case

- Same idea - use Taylor's expansion for $f : \mathbb{R}^n \to \mathbb{R}$.
- $f(x) =$
  $f(a) + \nabla f(a)(x-a)^T + \frac{1}{2}(x-a)^T H_f(a)(x-a) + \text{h.o. terms}$
- Ignoring higher order terms, we seek to minimize
  $g(x) = f(a) + \nabla f(a)(x-a)^T + \frac{1}{2}(x-a)^T H_f(a)(x-a)$
- A mimimun could be reached where the gradient of $g$ vanishes, of course with the additional condition that its Hessian is positive definite at a small neighborhood around $a$.
- Solving for $x$, we get the iterative formula
  $x_{n+1} = x_n - H_f^{-1}(x_n)\nabla f(x_n)$
- Please do not invert the Hessian. Instead solve $H_f(x_n)z_n = -\nabla f(x_n)$ and iterate $x_{n+1} = x_n + z_n$.

# Newton's Method Example

## The Problem

We wish to find a local minimum of $f(\mathbf{x}) = x_1^4 + x_1^2 x_2 + x_1^2 + 2x_2^2 + x_2$, starting with $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

First, compute $\nabla_{\mathbf{x}} f(\mathbf{x})$ and $H_f(\mathbf{x})$

## Compute the Gradient and the Hessian

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} 4x_1^3 + 2x_1 x_2 + 2x_1 \\ x_1^2 + 4x_2 + 1 \end{bmatrix}$$

$$H_f(\mathbf{x}) = \begin{bmatrix} 12x_1 + 2x_2 + 2 & 2x_1 \\ 2x_1 & 4 \end{bmatrix}$$

# Newton's Method Example (Cont)

## Continued

Plug in $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. What are the values of $\nabla_{\mathbf{x}} f(\mathbf{x}_0)$ and $H_f(\mathbf{x}_0)$?

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} 4x_1^3 + 2x_1 x_2 + 2x_1 \\ x_1^2 + 4x_2 + 1 \end{bmatrix} = \begin{bmatrix} \quad \\ \quad \end{bmatrix}$$

$$H_f(\mathbf{x}) = \begin{bmatrix} 12x_1 + 2x_2 + 2 & 2x_1 \\ 2x_1 & 4 \end{bmatrix} = \begin{bmatrix} \quad & \quad \\ \quad & \quad \end{bmatrix}$$

# Newton's Method Example

Plug in $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. What are the values of $\nabla_{\mathbf{x}} f(\mathbf{x}_0)$ and $H_f(\mathbf{x}_0)$?

$$\nabla_{\mathbf{x}} f(\mathbf{x}_0) = \begin{bmatrix} 8 \\ 6 \end{bmatrix}$$

$$H_f(\mathbf{x}_0) = \begin{bmatrix} 16 & 2 \\ 2 & 4 \end{bmatrix}$$

# Newton's Method Example

We need $\mathbf{s}_0$ so that $H_f(\mathbf{x}_0)\mathbf{s}_0 = -\nabla_{\mathbf{x}} f(\mathbf{x}_0)$.
Solve for $\mathbf{s}_0$ in:

$$\begin{bmatrix} 16 & 2 \\ 2 & 4 \end{bmatrix} \mathbf{s}_0 = -\begin{bmatrix} 8 \\ 6 \end{bmatrix}$$

Use Gauss Elimination!

...
We get

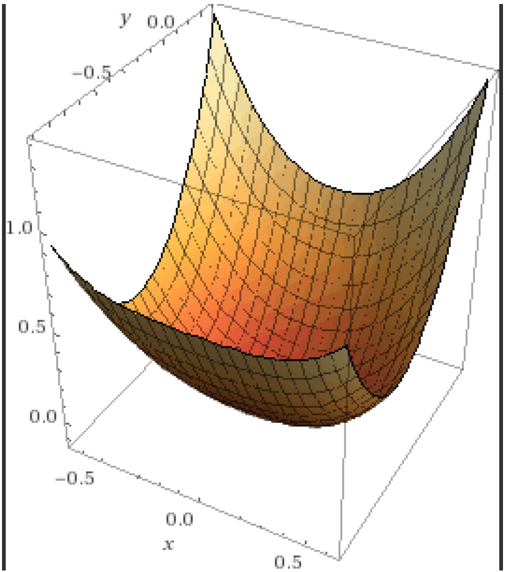
$$\mathbf{s}_0 = \begin{bmatrix} -\frac{2}{5} \\ -\frac{4}{5} \end{bmatrix}$$

# Newton's Method Update

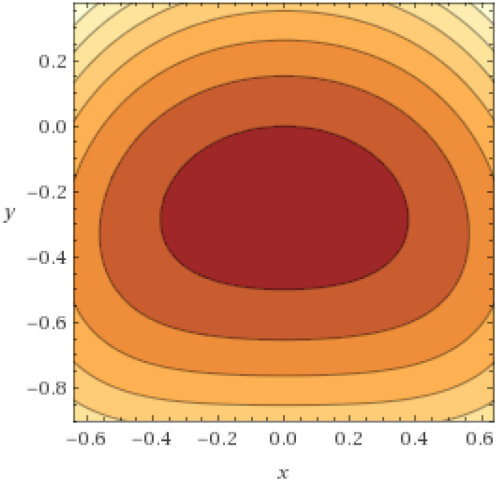
How do we compute $\mathbf{x}_1$ given

$\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $\mathbf{s}_0 = \begin{bmatrix} -\frac{2}{5} \\ -\frac{4}{5} \end{bmatrix}$?

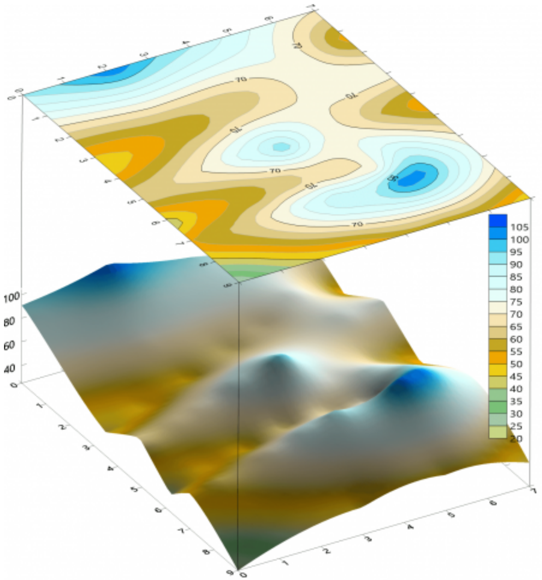
# Function 3D Plot



# Contour Plot



# How to Read Contour Plots



# Steepest Descent

## Steepest Descent / Gradient Descent

Key idea:

- The *gradient* of a differentiable function points *uphill*
- The *negative gradient* of a differentiable function points *downhill*
- The gradient is always perpendicular to the contour!

Why does this work?

- Intuition, a function $f : \mathbb{R}^n \to \mathbb{R}$ locally looks like a plane.
- $f(x) \approx f(a) + \nabla f(a)(x-a)^T$
- It turns out that $-\nabla f(a)$ has, locally, the direction of the steepest descent.

# Grade Prediction Example

Suppose the problem call for predicting a student's hw3 grade given their hw1 and hw2 grades.

$$A = \begin{bmatrix} a_1^{(1)} & a_2^{(1)} \\ a_1^{(2)} & a_2^{(2)} \\ \vdots & \vdots \\ a_1^{(73)} & a_2^{(73)} \end{bmatrix} \qquad b = \begin{bmatrix} b_1^{(1)} \\ b_1^{(2)} \\ \vdots \\ b_1^{(73)} \end{bmatrix}$$

Problem: Find $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ to minimize $||Ax - b||_2$

We can treat this as a non-linear optimization problem!

# Grade Prediction as Non-Linear Optimization

Define

$$
\begin{aligned}
f(\mathbf{x}) &= ||Ax - b||_2 \\
&= (Ax - \mathbf{b})^T (Ax - b) \\
&=
\end{aligned}
$$

## Computing Gradient

Now, given that

$$f(\mathbf{x}) = \sum_{j=1}^{73} (a_1^{(j)} x_1 + a_2^{(j)} x_2 - b^{(j)})^2$$

Let's compute $\nabla_x f(x)$:

# Gradient Descent

Start with some $x_0$, e.g. $x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ or $x_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$. (Why?)

Then take gradient descent steps until $x_{k+1}$ is sufficiently close to $x_k$, or until $f(x_{k+1})$ is sufficiently close to $f(x_k)$

## Why Gradient Descent?

Instead of computing $\nabla_x f(x)$ exactly, we can estimate the gradient using a small subset of our data (subset of 73 students)

Gradient descent works for more complicated functions, like neural networks!

# References

Michael T. Heath
Scientific Computing (Revised Second Edition)
SIAM