10.007: Modeling the Systems World

Week 03 - Cohort 2

The Steepest Descent Algorithm

Term 3, 2017

Singapore University of
Technology & Design

First things first!

It is important to be able to solve optimization problems with many variables.

Today we will learn the steepest descent algorithm and "implement" it together using MATLAB.

Please turn on your computers and download and unzip the contents of "Steepest_Descent.zip" from eDimension to a folder and get ready.

DON"T WAIT UNTIL LAST MINUTE, DO IT NOW:)

Agenda

- Step-by-step guide to unconstrained optimization.
- Steepest descent algorithm.
- Two versions of the algorithm:
  - fixed step size
  - line search
- MATLAB implementation.

Unconstrained optimization

Suppose we have the optimization problem:

$$\min\{f(x) : x \in \mathbb{R}^n\} \qquad \text{(P)},$$

where f is differentiable. This class assumes differentiability.

This is an unconstrained optimization problem: the feasible region is the entire space $\mathbb{R}^n$.

We can approach the solution of this problem in a systematic way, based on the necessary and sufficient conditions we learned in class. Solution methods for unconstrained problems provide the starting point for constrained problems.

Unconstrained optimization: Step 1

A step-by-step method to solve $\min\{f(x) : x \in \mathbb{R}^n\}$:

Step 1: Compute $\nabla f(x)$. There can be 3 cases:

Case 1: $\nabla f(x) = \vec{0}$ has no solution. There is no local minimum. **Stop.**

Case 2: $\nabla f(x) = \vec{0}$ can be solved analytically. Solve and go to **Step 2**.

Case 3: $\nabla f(x) = \vec{0}$ cannot be solved analytically: try using a numerical method such as the steepest descent algorithm to find a solution.

Subcase 3a: If the algorithm stops at a point, go to **Step 2**.

Subcase 3b: Otherwise, the algorithm fails: **Stop.**

Unconstrained optimization: Step 2

Step 2: Check if $f(x)$ is convex over $\mathbb{R}^n$. (Cohort 1.2)

Case 1: $f(x)$ is convex. All x^* that satisfy $\nabla f(x^*) = \vec{0}$ are global minima. **Stop.**

Case 2: $f(x)$ is not convex over $\mathbb{R}^n$. Go to **Step 3.**

Unconstrained optimization: Step 3

Step 3: For each point x^* that satisfies $\nabla f(x^*) = \vec{0}$, compute the Hessian matrix $H_f(x^*)$.

Case 1: $H_f(x^*)$ is pd. Then x^* is a local minimum.

Case 2: $H_f(x^*)$ is psd but $H_f(x^*)$ is NOT pd. Then x^* could be a local minimum: investigate the neighborhood (no recipe is given).

Case 3: $H_f(x^*)$ is not psd, then x^* is not a local minimum.

Output: list of all local minima, plus possibly some points that could be local minima but cannot be verified. **Stop.** (If Step 1 fails, it is possible that some local minima are missed.)

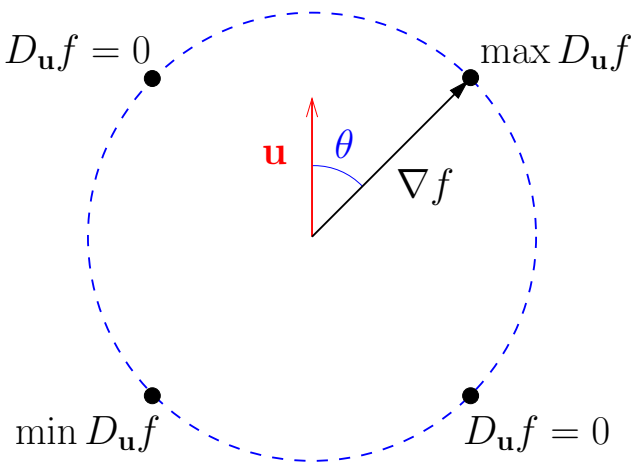
Directional derivative (see MATH 2!)

Suppose $f : \mathbb{R}^2 \to \mathbb{R}$ and $u \in \mathbb{R}^2, u = (u_1, u_2)$.

Recall that the directional derivative at any point x_0, y_0 is:

$$D_u f = \nabla f^T u = \|\nabla f(x, y)\| \, \|u\| \cos\theta,$$

where θ is the angle between $\nabla f(x_0, y_0)$ and u.



More on directional derivative

$$\boxed{D_u f(x_0, y_0) = \nabla f(x_0, y_0)^T u = \|\nabla f(x_0, y_0)\| \, \|u\| \cos\theta.}$$

1. $\theta = 0$ gives $\cos\theta = 1$:
   the maximum increase is in the direction of $\nabla f(x_0, y_0)$.

2. $\theta = \pi$ gives $\cos\theta = -1$:
   the maximum decrease is in the direction of $-\nabla f(x_0, y_0)$.

3. $\theta = \pi/2$ or $-\pi/2$ gives $\cos\theta = 0$:
   there is no increase or decrease in the direction perpendicular to $-\nabla f(x_0, y_0)$.

   Level curves or contour lines connect points of equal value, thus giving the direction of no increase or decrease. Hence the gradient is always perpendicular to the level curve at a point.

Steepest descent

Unconstrained optimization problem:

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a twice differentiable function. Find:

$$\min_{x \in \mathbb{R}^n} f(x)$$

Idea:

1. Choose an arbitrary starting point x.

Steepest descent

Unconstrained optimization problem:

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a twice differentiable function. Find:

$$\min_{x \in \mathbb{R}^n} f(x)$$

Idea:

1. Choose an arbitrary starting point x.
2. Direction of steepest descent: $-\nabla f(x)$.

Steepest descent

Unconstrained optimization problem:

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a twice differentiable function. Find:

$$\min_{x \in \mathbb{R}^n} f(x)$$

Idea:

1. Choose an arbitrary starting point x.
2. Direction of steepest descent: $-\nabla f(x)$.
3. Take a step in this direction (to a lower point), re-evaluate gradient at the new point and iterate.

Steepest descent

Unconstrained optimization problem:

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a twice differentiable function. Find:

$$\min_{x \in \mathbb{R}^n} f(x)$$

Idea:

1. Choose an arbitrary starting point x.
2. Direction of steepest descent: $-\nabla f(x)$.
3. Take a step in this direction (to a lower point), re-evaluate gradient at the new point and iterate.

Questions:

- What is a good step size?

Steepest descent

Unconstrained optimization problem:

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a twice differentiable function. Find:

$$\min_{x \in \mathbb{R}^n} f(x)$$

Idea:

1. Choose an arbitrary starting point x.
2. Direction of steepest descent: $-\nabla f(x)$.
3. Take a step in this direction (to a lower point), re-evaluate gradient at the new point and iterate.

Questions:

- What is a good step size?
- When to stop?

Steepest descent: algorithm

Algorithm:

- Fix a small precision level $\epsilon > 0$.
- Start at an arbitrary value $x := x_0$.

1. Calculate steepest descent direction: $-\nabla f(x)$.
2. Choose a stepsize σ (*fixed step* or *line search*).
3. Update $x_{new} = x - \sigma \nabla f(x)$.
4. If $\|\nabla f(x_{new})\| < \epsilon$, report x_{new} as minimum,
   Else update $x = x_{new}$ and go to Step 1.

Remark: the algorithm finds a point with $\|\nabla f(x_{new})\| \approx 0$. This is a global minimum only if f is convex! If f is not convex, it could be a local minimum or a saddle point — we must check second order conditions to investigate.

Steepest descent: algorithm

Algorithm:

- Fix a small precision level $\epsilon > 0$.
- Start at an arbitrary value $x := x_0$.

1. Calculate steepest descent direction: $-\nabla f(x)$.
2. Choose a stepsize σ (*fixed step* or *line search*).
3. Update $x_{new} = x - \sigma \nabla f(x)$.
4. If $\|\nabla f(x_{new})\| < \epsilon$, report x_{new} as minimum,
   Else update $x = x_{new}$ and go to Step 1.

Remark: The solution reported as the minimum is not always the exact minimum, but it is very close. We choose the precision level at the beginning!

Steepest descent: step size selection

1. Fixed step size: (σ is fixed)
   - Small step size: the algorithm might need a long time to reach a solution.
   - Large step size: the algorithm might miss the solution never reaching it at all.

   A good fixed step size is dependent on the parameters!

Steepest descent: step size selection

1. Fixed step size: (σ is fixed)
   - Small step size: the algorithm might need a long time to reach a solution.
   - Large step size: the algorithm might miss the solution never reaching it at all.

     A good fixed step size is dependent on the parameters!
2. Line search: (σ varies)
   - At any point x^*, choose next step size σ^* so as to minimize the one-dimensional function:

     $$h(\sigma) = f(x^* - \sigma \nabla f(x^*))$$

     over all $\sigma \geqslant 0$.
   - Update $x_{new} = x^* - \sigma^* \nabla f(x^*)$.

     Interpretation: we follow the direction of maximum decrease, until we hit the minimum along that line.

Activity 1 (20 minutes)

Solve the following optimization problem using steepest descent:

$$\min_{(x_1,x_2)\in\mathbb{R}^2} 2x_1^2 + x_2^2 + (x_1 + x_2)^2 - 20x_1 - 16x_2.$$

Start at the point $(0, 0)$ and stop when $\|\nabla f\| < \varepsilon = 0.05$. Use line search!

(Note: this problem can be solved analytically. We use it for gradient descent because we can easily do the calculations. For more difficult problems, we need a computer.)

Activity 2 (MATLAB - prep 2 min)

It is important to be able to solve optimization problems with many variables.

We will “implement” the steepest descent algorithm using MATLAB together.

Please turn on your computers and download and unzip the contents of “Steepest_Descent.zip” from eDimension to a folder and get ready.

Activity 2 (MATLAB: 10min)

Solve the following optimization problem using steepest descent with MATLAB:

$$\min_{(x_1,x_2)\in\mathbb{R}^2} 2x_1^2 + x_2^2 + (x_1 + x_2)^2 - 20x_1 - 16x_2.$$

1. Edit file "func.m" so that it can be used to evaluate the function value at any given point.

2. Edit file "grad.m" so that it can be used to evaluate the function value at any given point.

Activity 2 (MATLAB: 10 min)

Solve the following optimization problem using steepest descent with MATLAB:

$$\min_{(x_1,x_2)\in\mathbb{R}^2} 2x_1^2 + x_2^2 + (x_1 + x_2)^2 - 20x_1 - 16x_2.$$

3. Edit file "SteepestDescent.m" so that it implement the algorithm with fixed step size. Try the following and discuss how does the output (solution, function value, number of iterations) change:

   0.1 Different starting points: Try initial x to be (0,0), (1,1), (-1,1)

   0.2 Different step sizes: Try $\sigma \in \{0.1, 0.01, 0.001\}$

   0.3 Different stopping criteria: Try $\varepsilon \in \{0.05, 0.001, 0.0001\}$

Activity 2 (MATLAB: 20min)

Solve the following optimization problem using steepest descent with MATLAB:

$$\min_{(x_1,x_2)\in\mathbb{R}^2} 2x_1^2 + x_2^2 + (x_1 + x_2)^2 - 20x_1 - 16x_2.$$

4. Change "linesearch=1" in Line 12 "to linesearch = 0" in order to implement line search:

   4.1 Write down an expression for $h'(\sigma)$ using chain rule (MATH 2).

   4.2 Solve for σ^*.

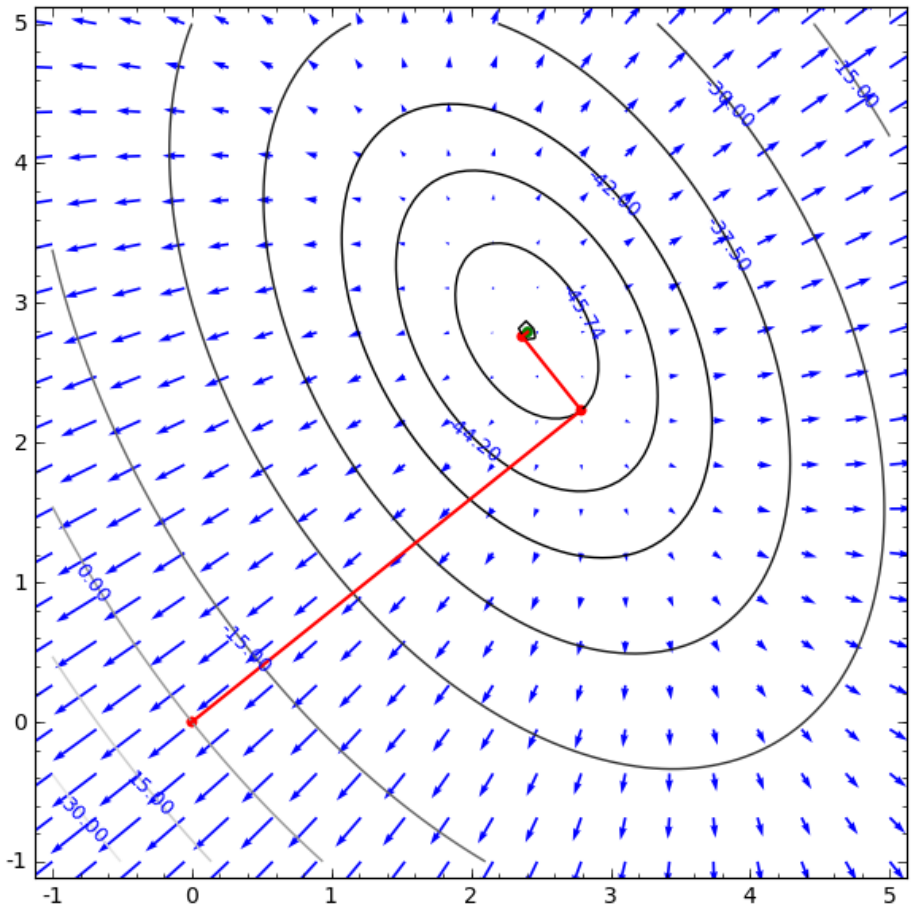
   4.3 Edit the .m file using σ^* at each iteration.

5. Choose another function to minimize and have some fun!

Summary

- Unconstrained optimization: a review.
- Steepest descent algorithm.

Some remarks about the descent directions



Stepest descent with line search!

Some remarks about the descent directions

Think about the last picture. Can you explain why the trajectory followed by the algorithm is a “zig-zag” where consecutive segments are perpendicular?