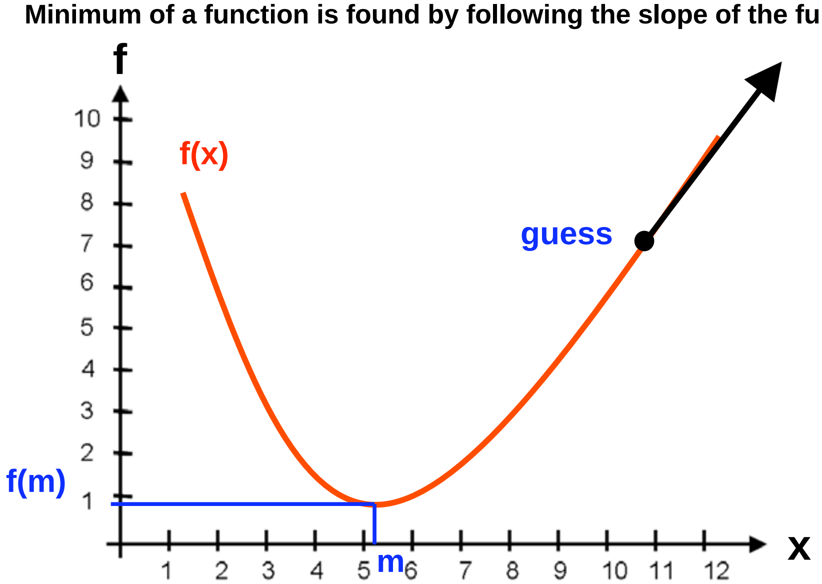
Lecture 10: descent methods

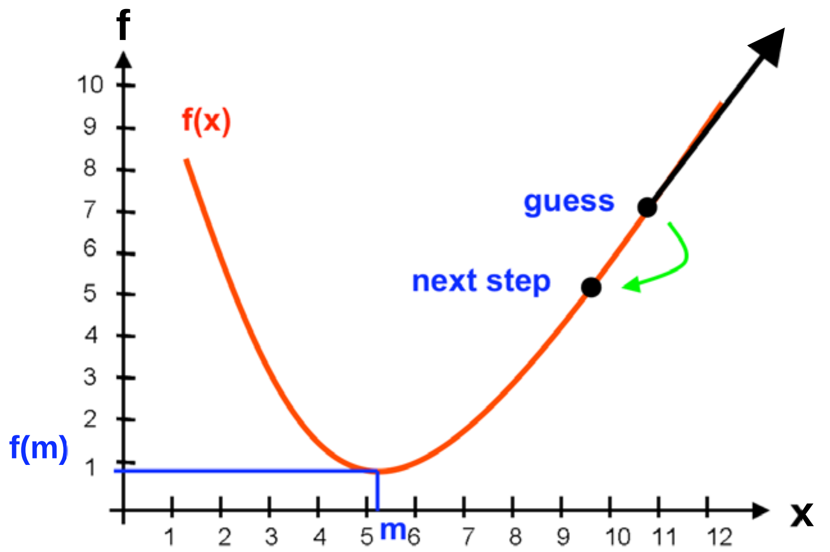
- Generic descent algorithm
- Generalization to multiple dimensions
- Problems of descent methods, possible improvements
- Fixes
- Local minima

Gradient descent (reminder)

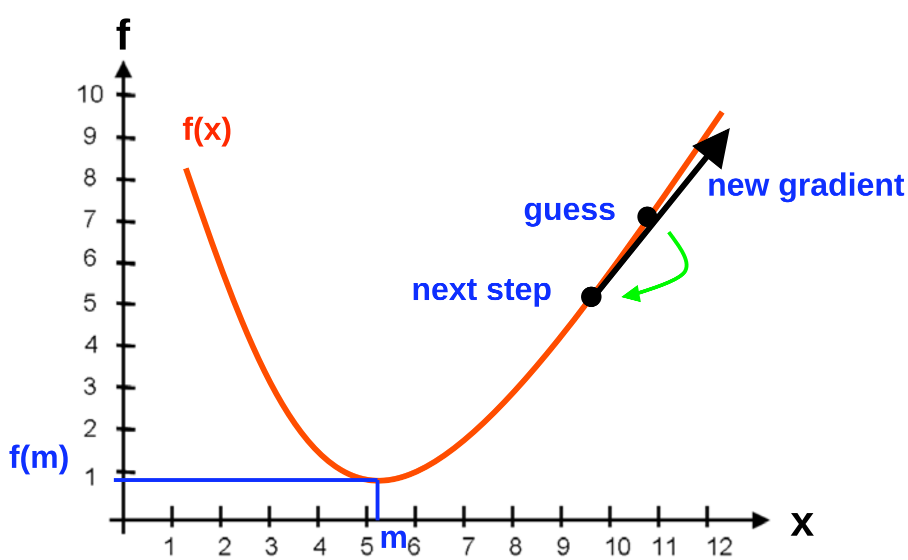
Minimum of a function is found by following the slope of the function



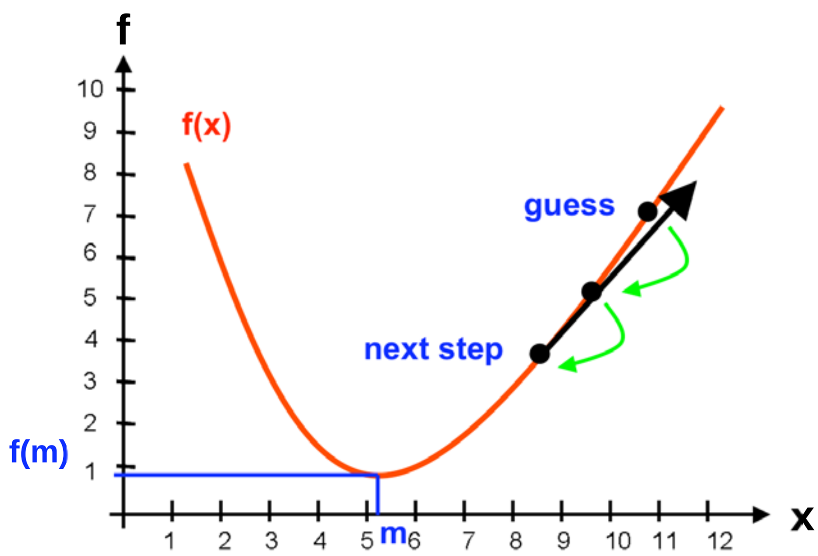
Gradient descent (illustration)



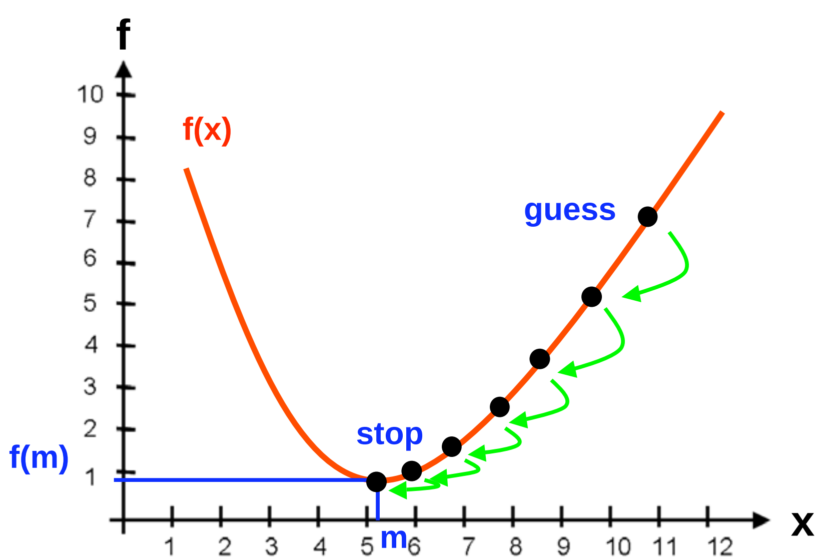
Gradient descent (illustration)



Gradient descent (illustration)



Gradient descent (illustration)



Gradient descent: algorithm

Start with a point (guess)

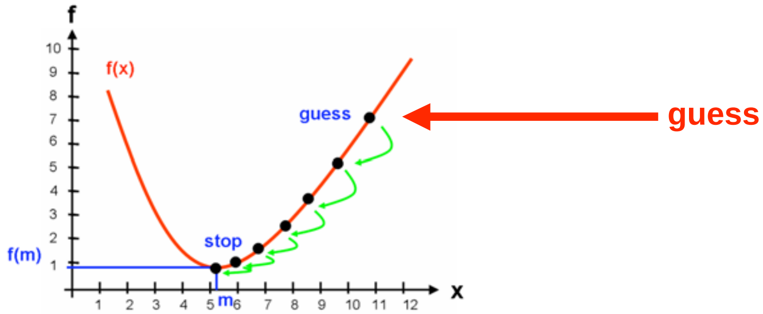
Repeat

Determine a descent direction

Choose a step

Update

Until stopping criterion is satisfied



Gradient descent: algorithm

Start with a point (guess)

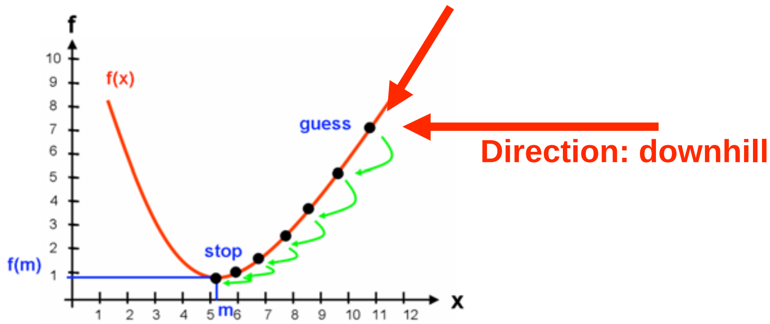
Repeat

Determine a descent direction

Choose a step

Update

Until stopping criterion is satisfied



Gradient descent: algorithm

Start with a point (guess)

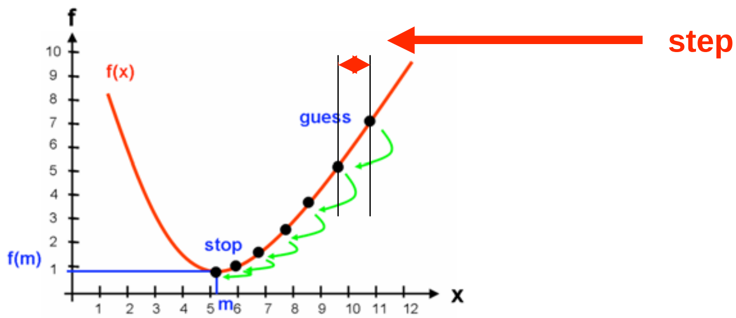
Repeat

Determine a descent direction

Choose a step

Update

Until stopping criterion is satisfied



Gradient descent: algorithm

Start with a point (guess)

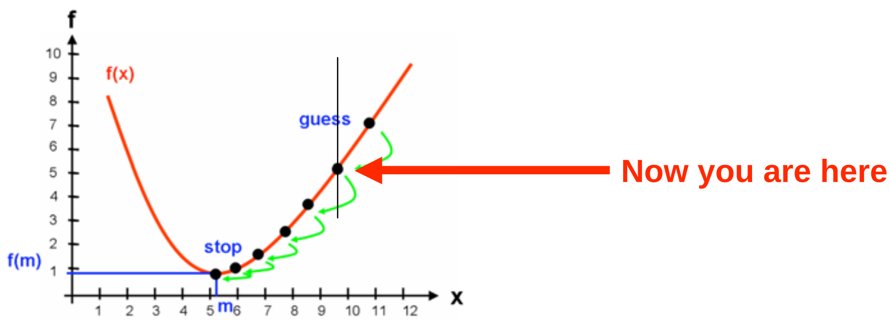
Repeat

Determine a descent direction

Choose a step

Update

Until stopping criterion is satisfied



Gradient descent: algorithm

Start with a point (guess)

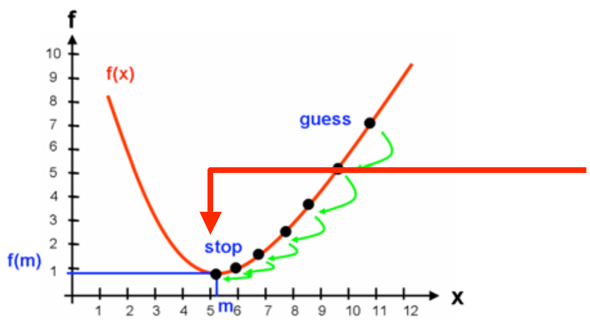
Repeat

Determine a descent direction

Choose a step

Update

Until stopping criterion is satisfied



Stop when "close" from minimum

Gradient descent: algorithm

Start with a point (guess)

Repeat

Determine a descent direction

Choose a step

Update

Until stopping criterion is satisfied

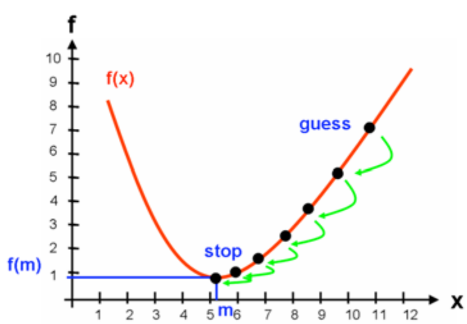
guess = x

direction = -f'(x)

step = h > 0

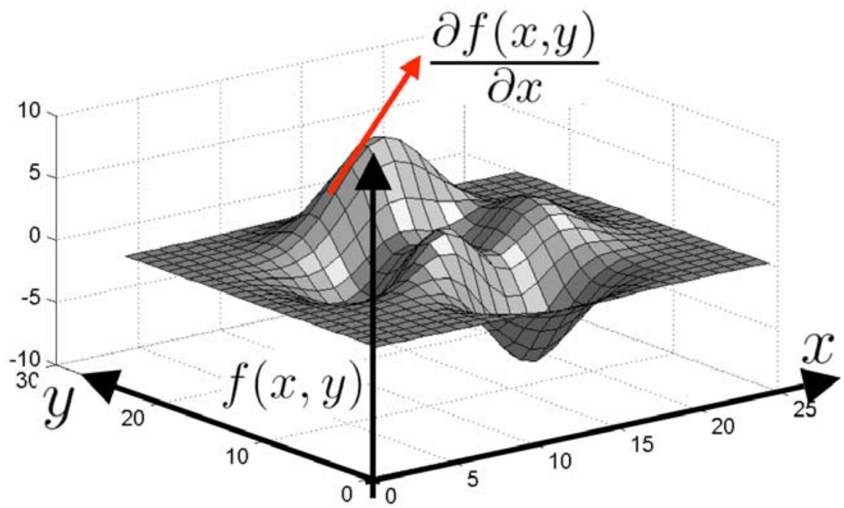
x:=x–hf'(x)

f'(x)~0



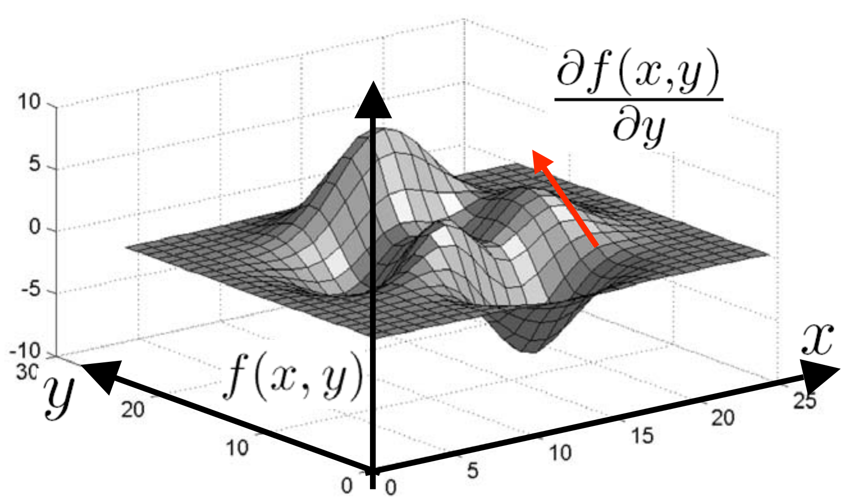
Example of 2D gradient: pic of the MATLAB demo

Illustration of the gradient in 2D



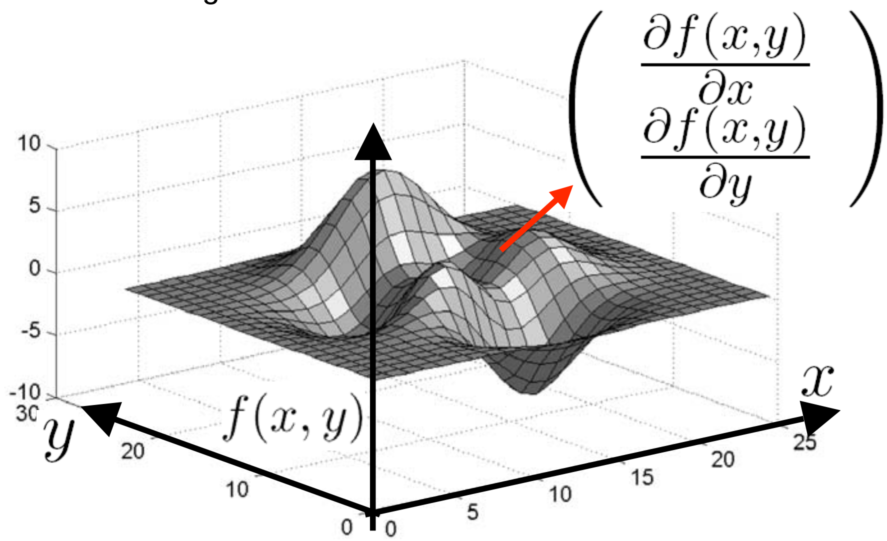
Example of 2D gradient: pic of the MATLAB demo

Illustration of the gradient in 2D



Example of 2D gradient: pic of the MATLAB demo

Illustration of the gradient in 2D



Example of 2D gradient: pic of the MATLAB demo

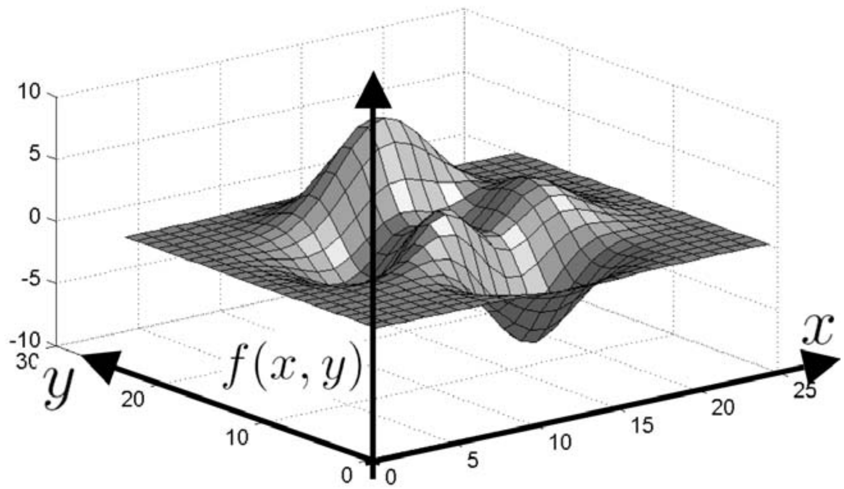
Definition of the gradient in 2D

$$\nabla f(x,y) = \begin{pmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{pmatrix}$$

This is just a genaralization of the derivative in two dimensions.
This can be generalized to any dimension.

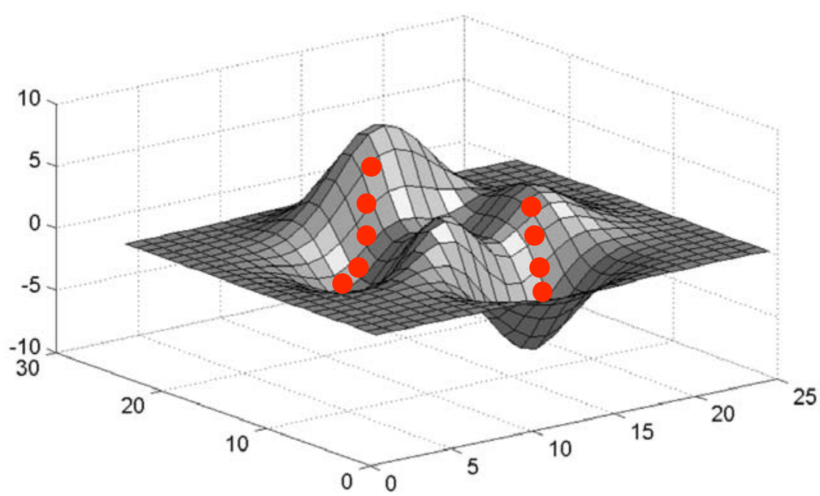
Example of 2D gradient: pic of the MATLAB demo

Illustration of the gradient in 2D



Example of 2D gradient: pic of the MATLAB demo

Gradient descent works in 2D



Generalization to multiple dimensions

Start with a point (guess)

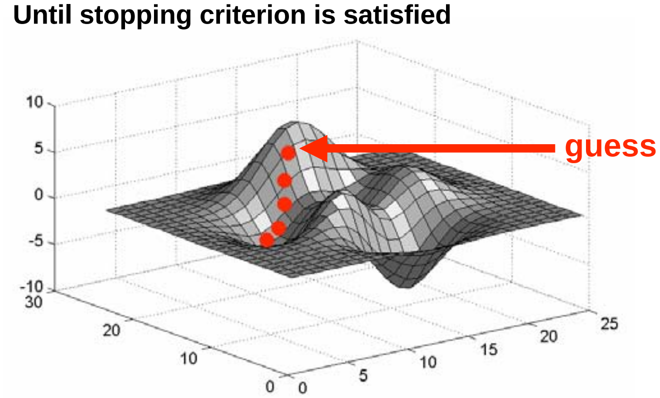
Repeat

Determine a descent direction

Choose a step

Update

Until stopping criterion is satisfied



Generalization to multiple dimensions

Start with a point (guess)

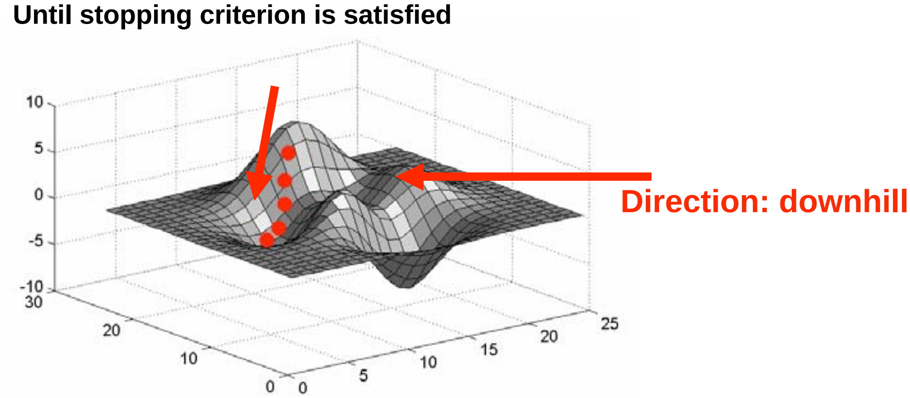
Repeat

Determine a descent direction

Choose a step

Update

Until stopping criterion is satisfied



Generalization to multiple dimensions

Start with a point (guess)

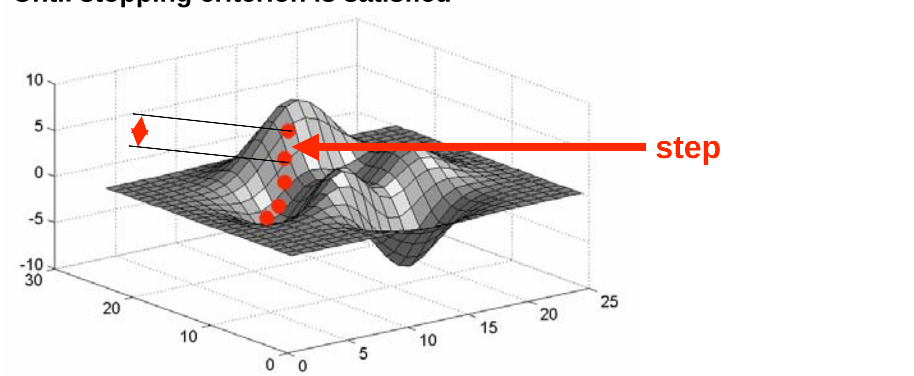
Repeat

Determine a descent direction

Choose a step

Update

Until stopping criterion is satisfied



Generalization to multiple dimensions

Start with a point (guess)

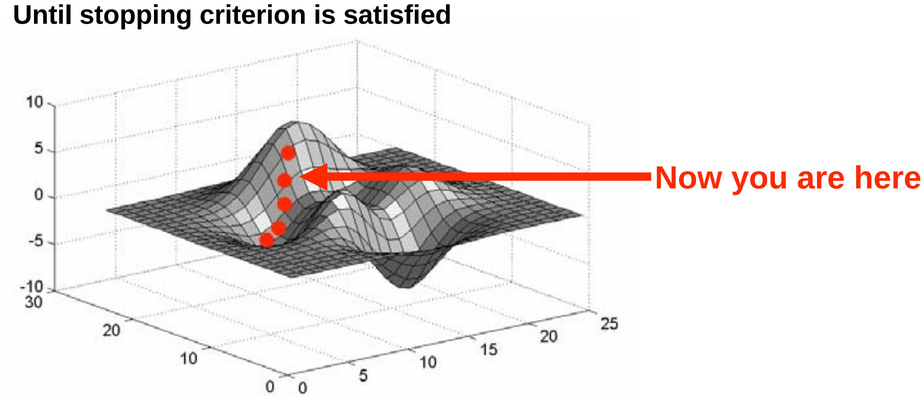
Repeat

Determine a descent direction

Choose a step

Update

Until stopping criterion is satisfied



Generalization to multiple dimensions

Start with a point (guess)

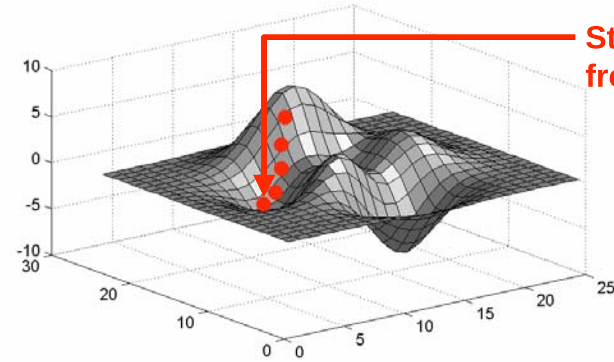
Repeat

Determine a descent direction

Choose a step

Update

Until stopping criterion is satisfied



Stop when "close" from minimum

Generalization to multiple dimensions

Start with a point (guess)

Repeat

Determine a descent direction

Choose a step

Update

Until stopping criterion is satisfied

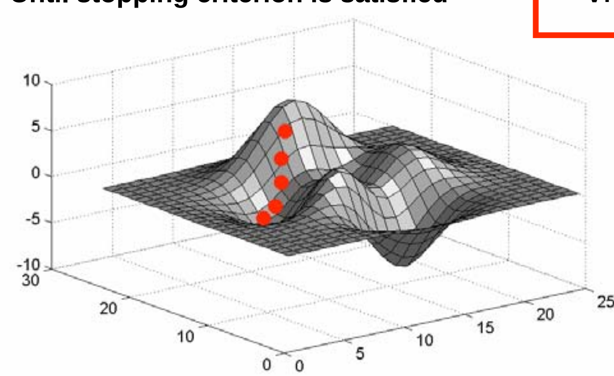
guess = x

direction = -f(x)

step = h > 0

$x := x - h \nabla f'(x)$

$\nabla f'(x) \sim 0$



Multiple dimensions

Everything that you have seen with derivatives can be generalized with the gradient.

For the descent method, f'(x) can be replaced by

$$\nabla f(x,y) = \begin{pmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{pmatrix}$$

In two dimensions, and by

$$\nabla f(x_1, x_2, \cdots, x_i, \cdots, x_N) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_i}, \cdots, \frac{\partial f}{\partial x_N} \right)$$

in N dimensions.

Example of 2D gradient: MATLAB demo

The cost to buy a portfolio is:

$$f(x_1, x_2, \cdots, x_i, \cdots, x_N) = x_1^2 \cdot (x_2 - 4)^3 + \sum_{i=3}^{N} x_i^2$$

Stock 1

Stock 2

Stock i

Stock N

If you want to minimize the price to buy your portfolio, you need to compute the gradient of its price:

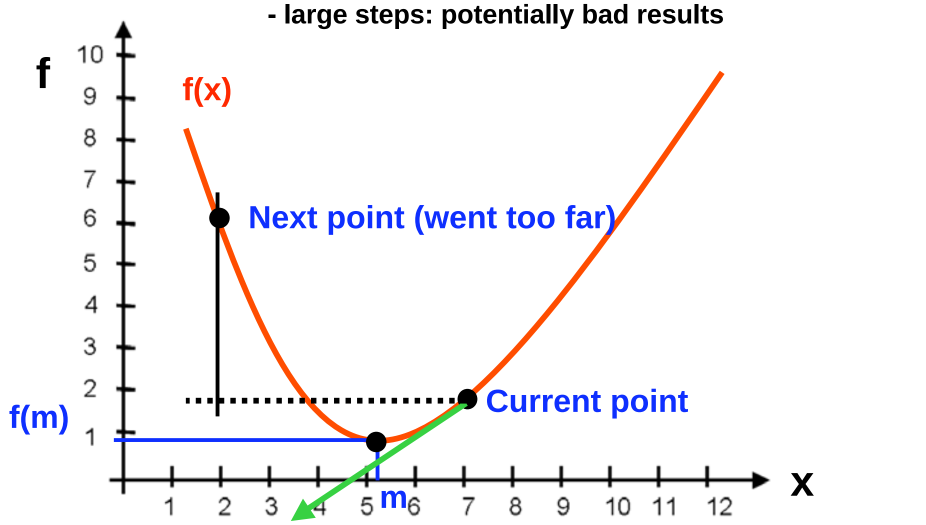
$$\nabla f(x_1, x_2, \cdots, x_i, \cdots, x_N) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_i}, \cdots, \frac{\partial f}{\partial x_N} \right)$$

Problem 1: choice of the step

When updating the current computation:
- small steps: inefficient
- large steps: potentially bad results

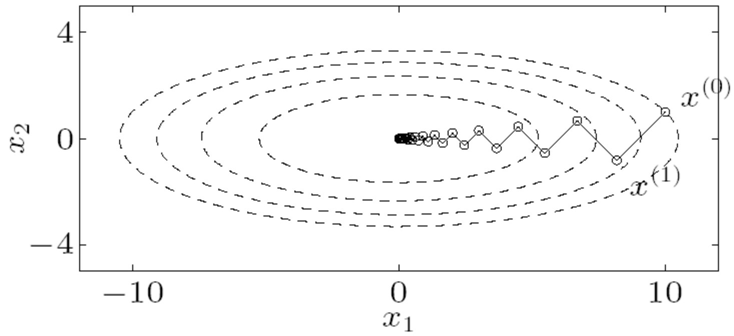
Problem 1: choice of the step

When updating the current computation:
- small steps: inefficient
- large steps: potentially bad results



Problem 2: « ping pong effect »

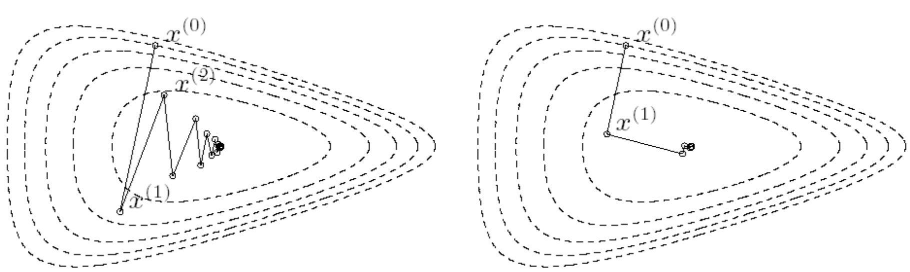
$$f(x) = (1/2)(x_1^2 + \gamma x_2^2) \qquad (\gamma > 0)$$



[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

Problem 2: « ping pong effect »

$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$



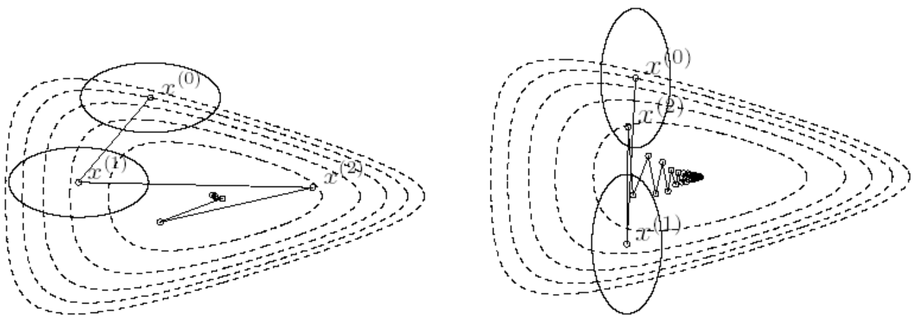
backtracking line search

exact line search

[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

Problem 2: (other norm dependent issues)

$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$



[S. Boyd, L. Vandenberghe, Convex Convex Optimization lect. Notes, Stanford Univ. 2004]

Problem 3: stopping criterion

Intuitive criterion:

$$|f'(x)| \leq \varepsilon$$

In multiple dimensions:

$$\|\nabla f\| \leq \varepsilon$$

Or equivalently

$$\|\nabla f\| = \sqrt{\sum_{i=1}^{N}\left(\frac{\partial f}{\partial x_i}\right)^2} = \sqrt{\left(\frac{\partial f}{\partial x_1}\right)^2 + \left(\frac{\partial f}{\partial x_2}\right)^2 + \cdots + \left(\frac{\partial f}{\partial x_N}\right)^2} \leq \varepsilon$$

Rarely used in practice.
More about this in EE227A (convex optimization, Prof. L. El Ghaoui).

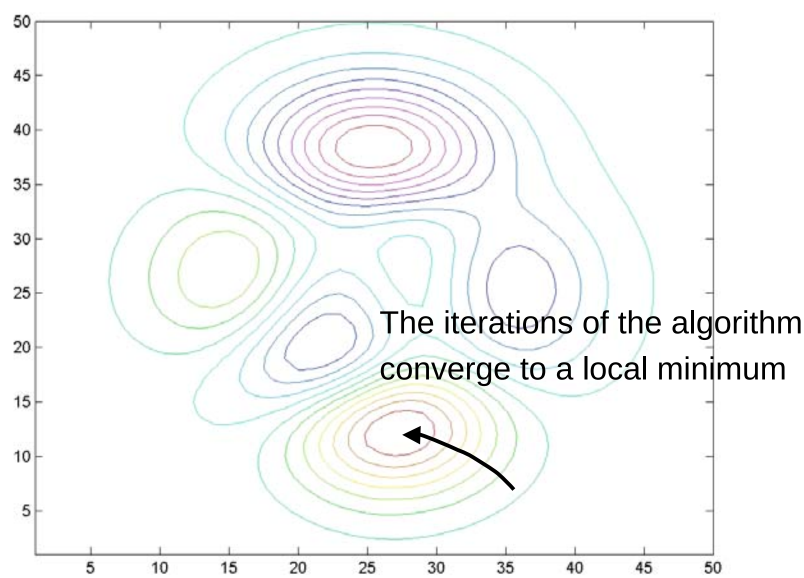
Fixes

Several methods exist to address this problem

- **Line search methods, in particular**
  - **Backtracking line search**
  - **Exact line search**
- **Normalized steepest descent**
- **Newton steps**

Fundamental problem of the method: local minima

Local minima: pic of the MATLAB demo



Local minima: pic of the MATLAB demo

View of the algorithm is « myopic »