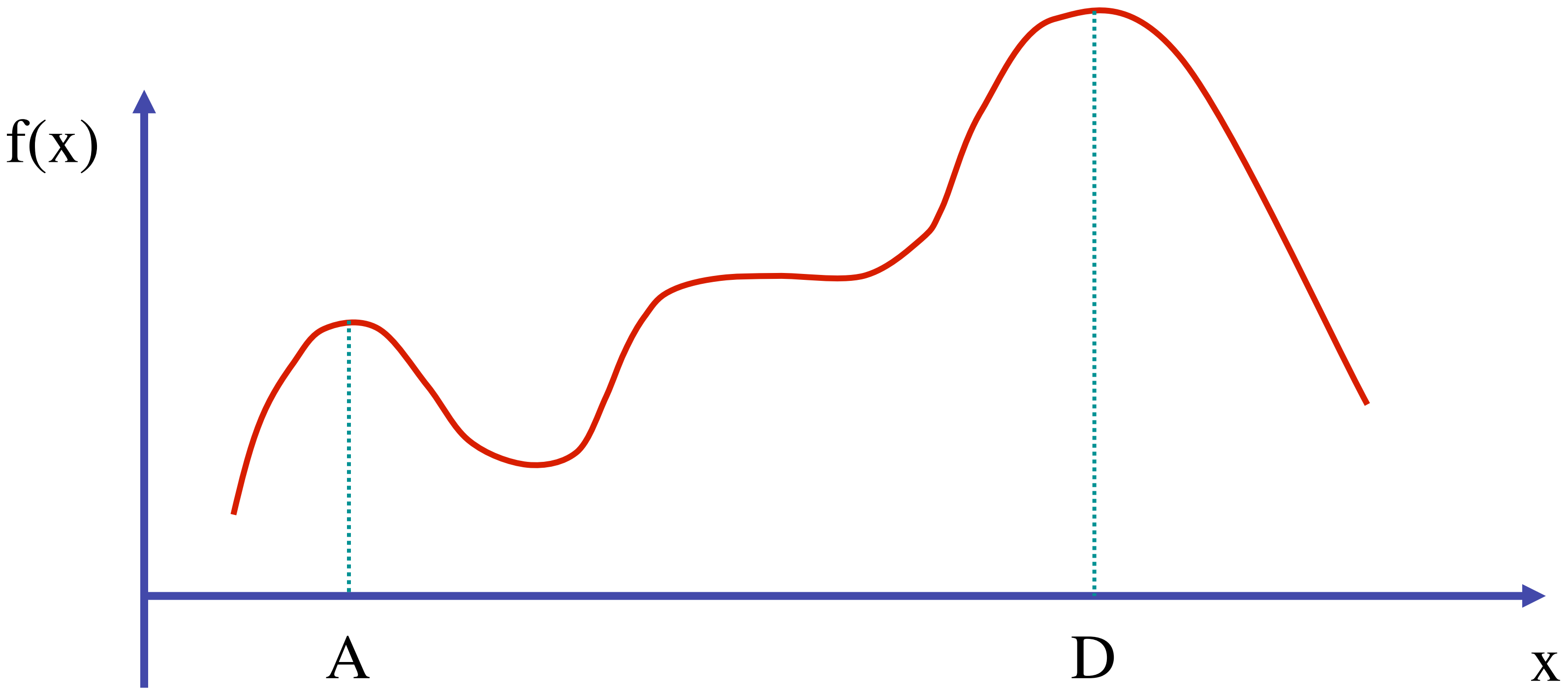
Some Optimization Topics

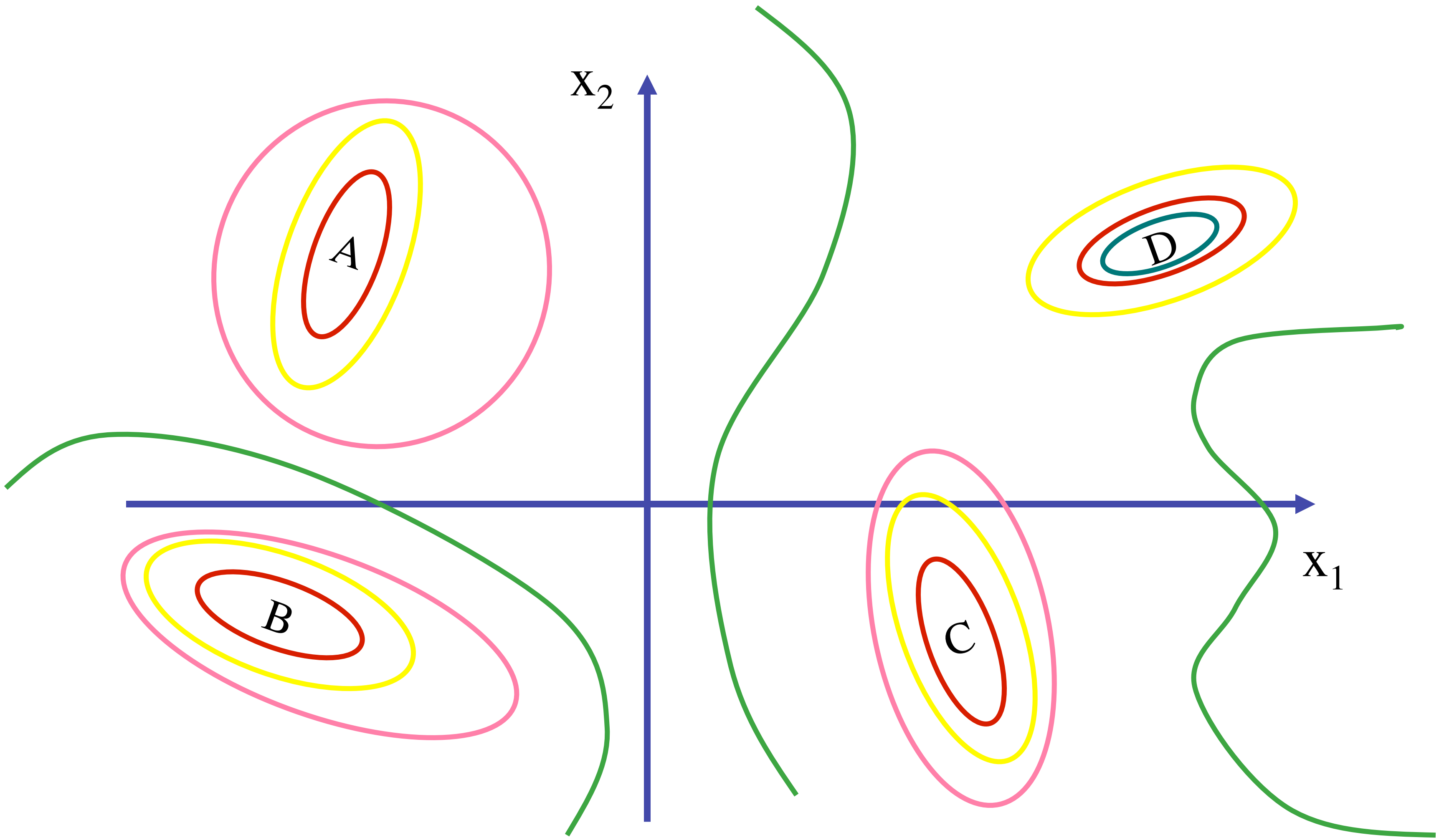
Local and Global Optima

Which one is the real maximum?



For $x = A$ and $x = D$, we have: $\dfrac{df}{dx} = 0$ and $\dfrac{d^2 f}{dx^2} < 0$

Which one is the real optimum?



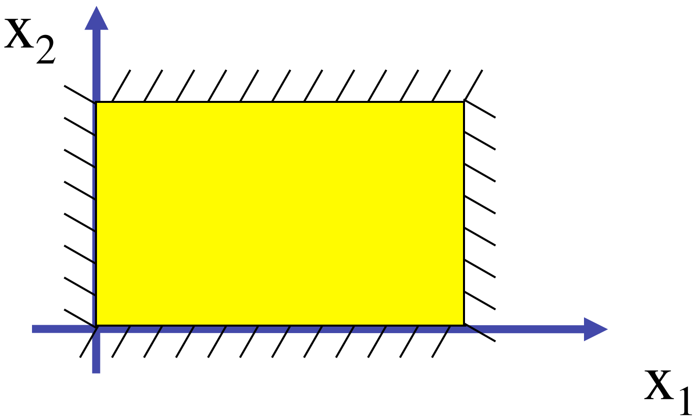
Local and Global Optima

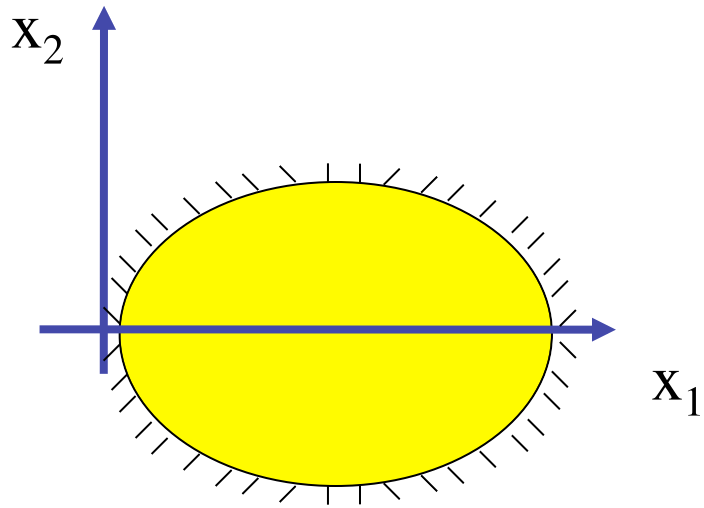
- The optimality conditions are *local* conditions
- They do not compare separate optima
- If I find an optimum can I be sure that it is the *global optimum*?
- In general, to find the *global* optimum, we must find and compare *all* the optima
- In large problems, this can be very difficult and time consuming

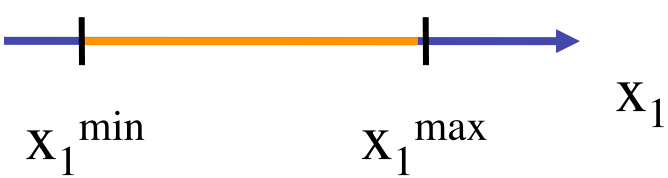
Convexity

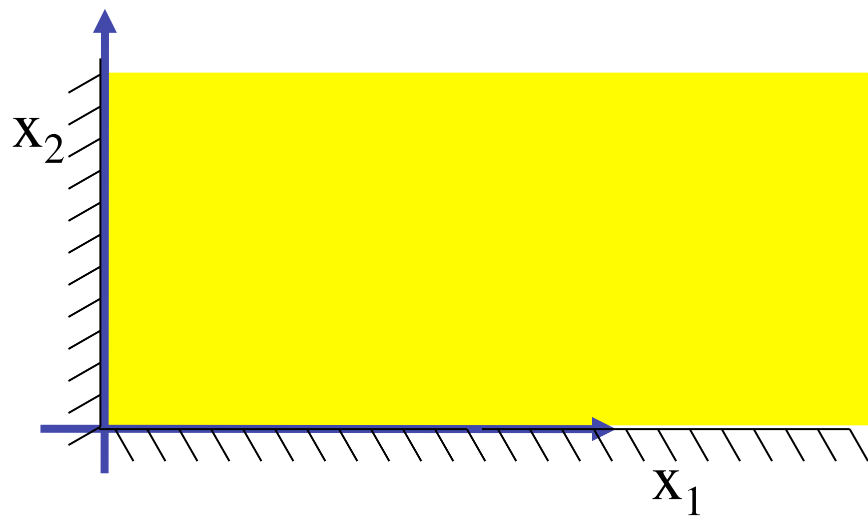
- If the feasible set is convex and the objective function is convex, there is only one minimum and it is thus the global minimum

Examples of Convex Feasible Sets

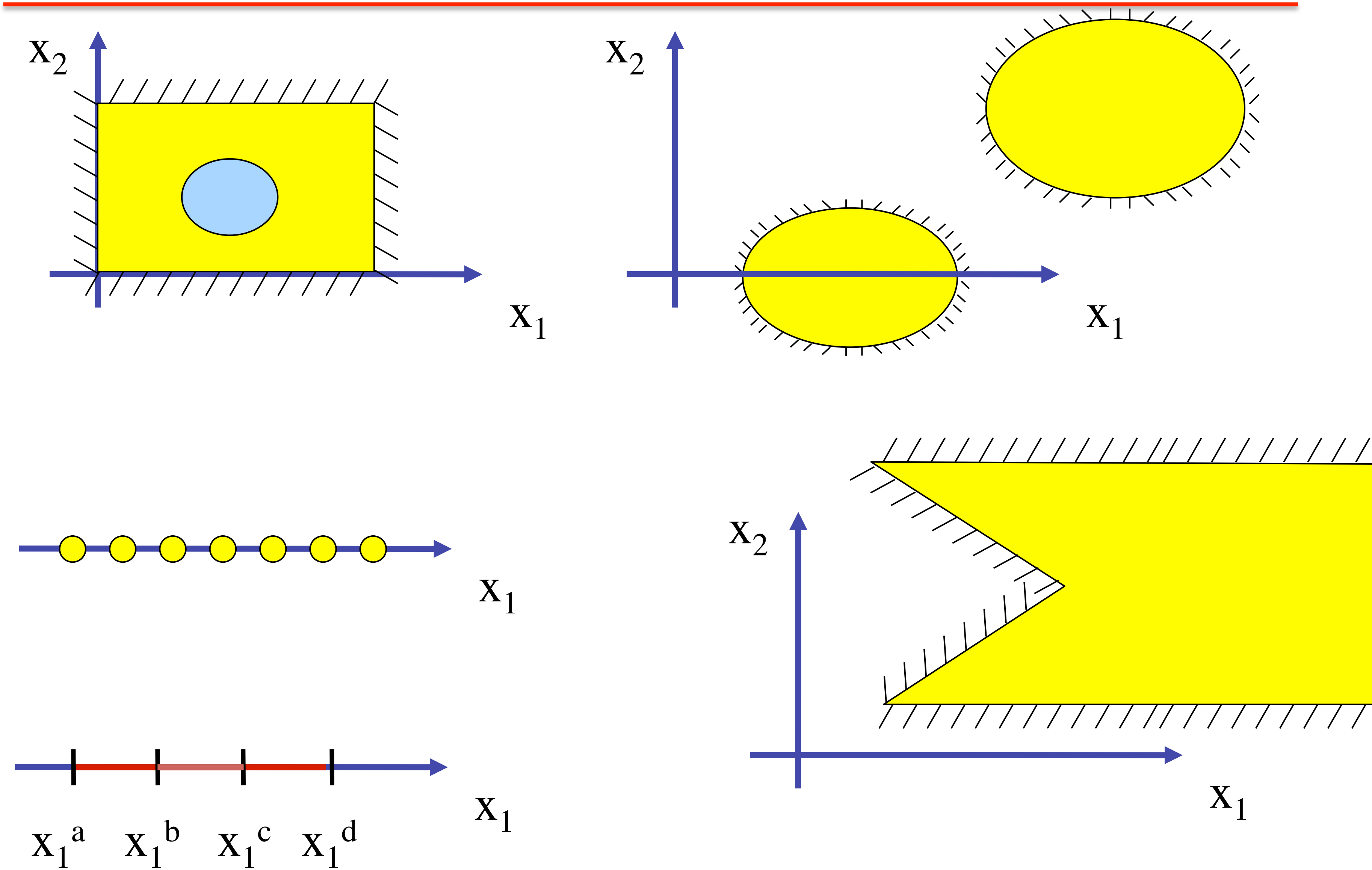






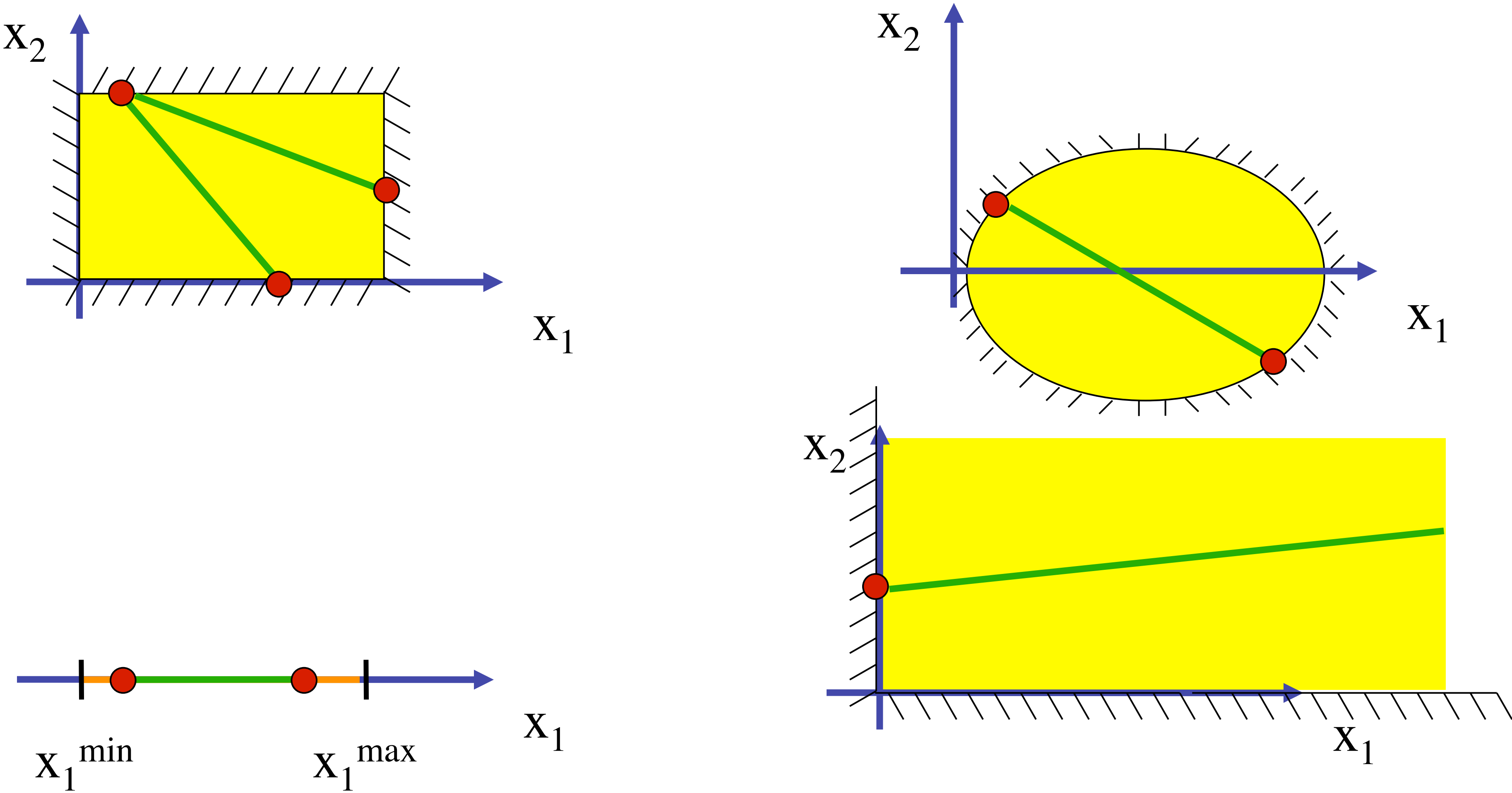


Example of Non-Convex Feasible Sets

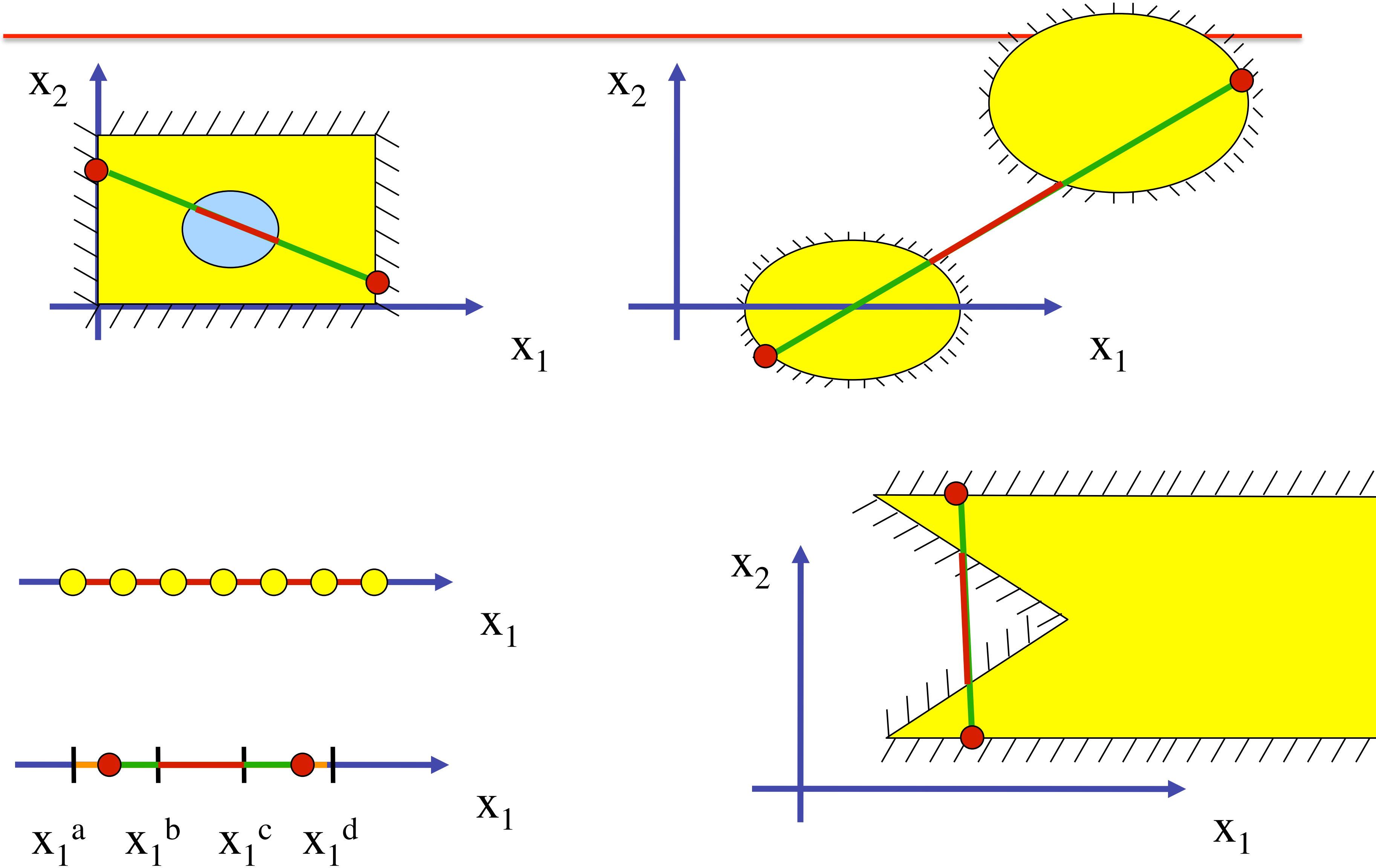


Example of Convex Feasible Sets

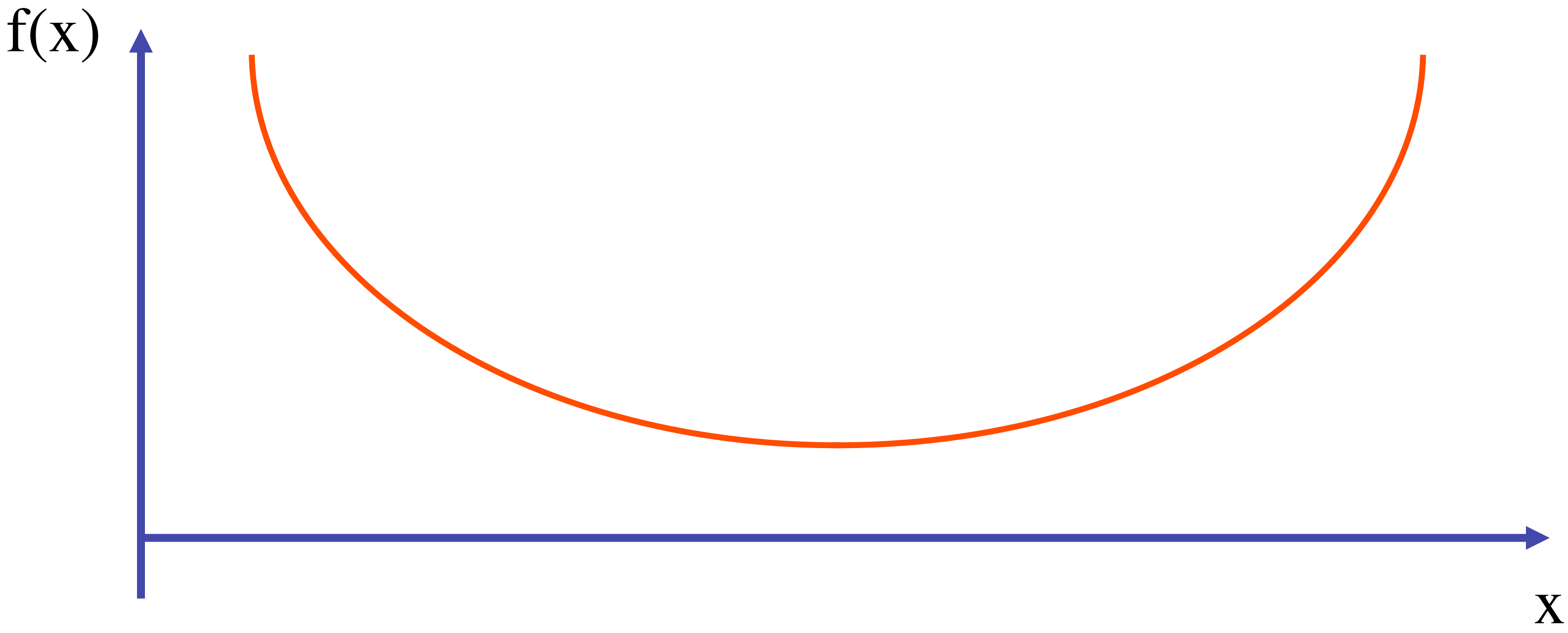
A set is convex if, for any two points belonging to the set, all the points on the straight line joining these two points belong to the set



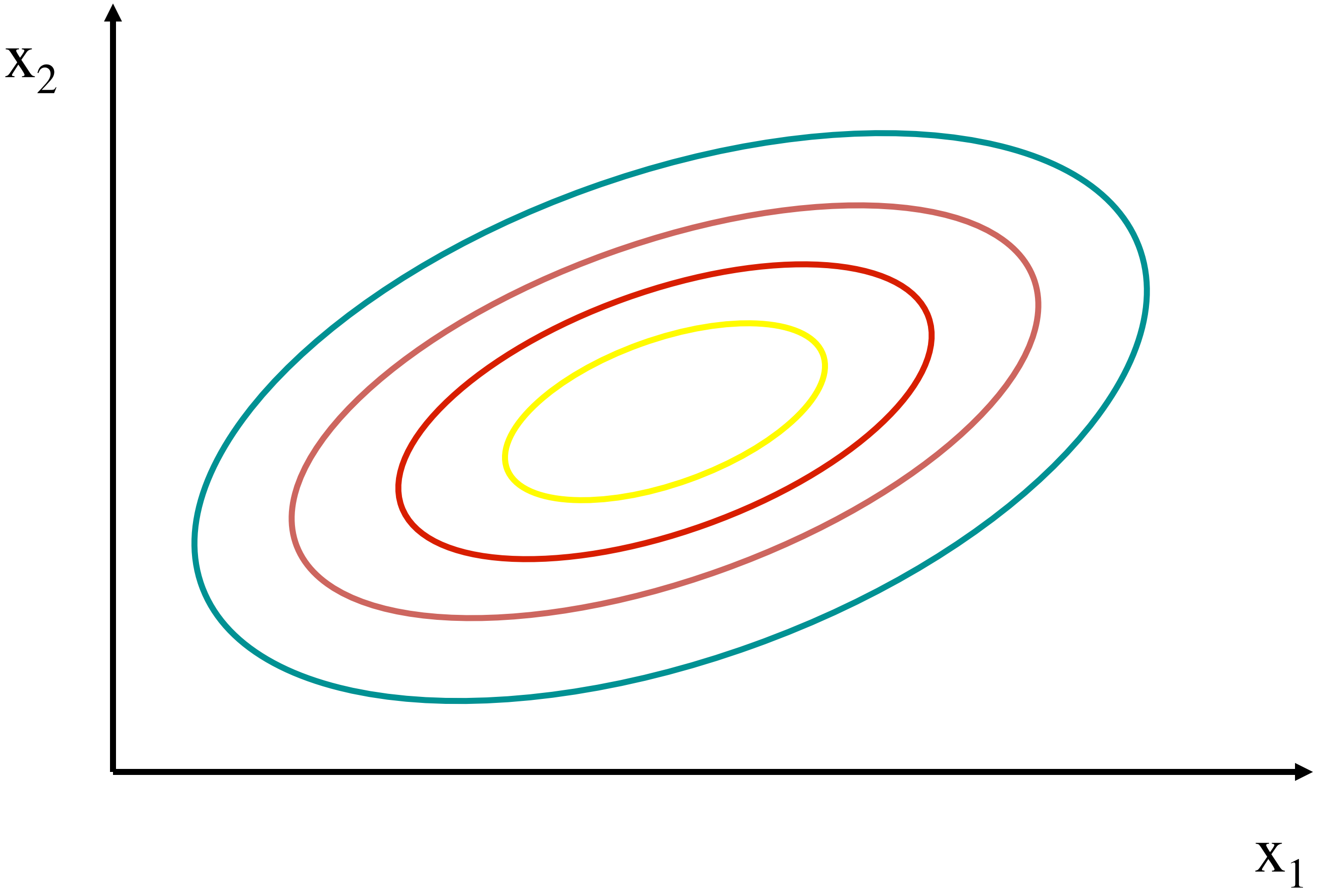
Example of Non-Convex Feasible Sets



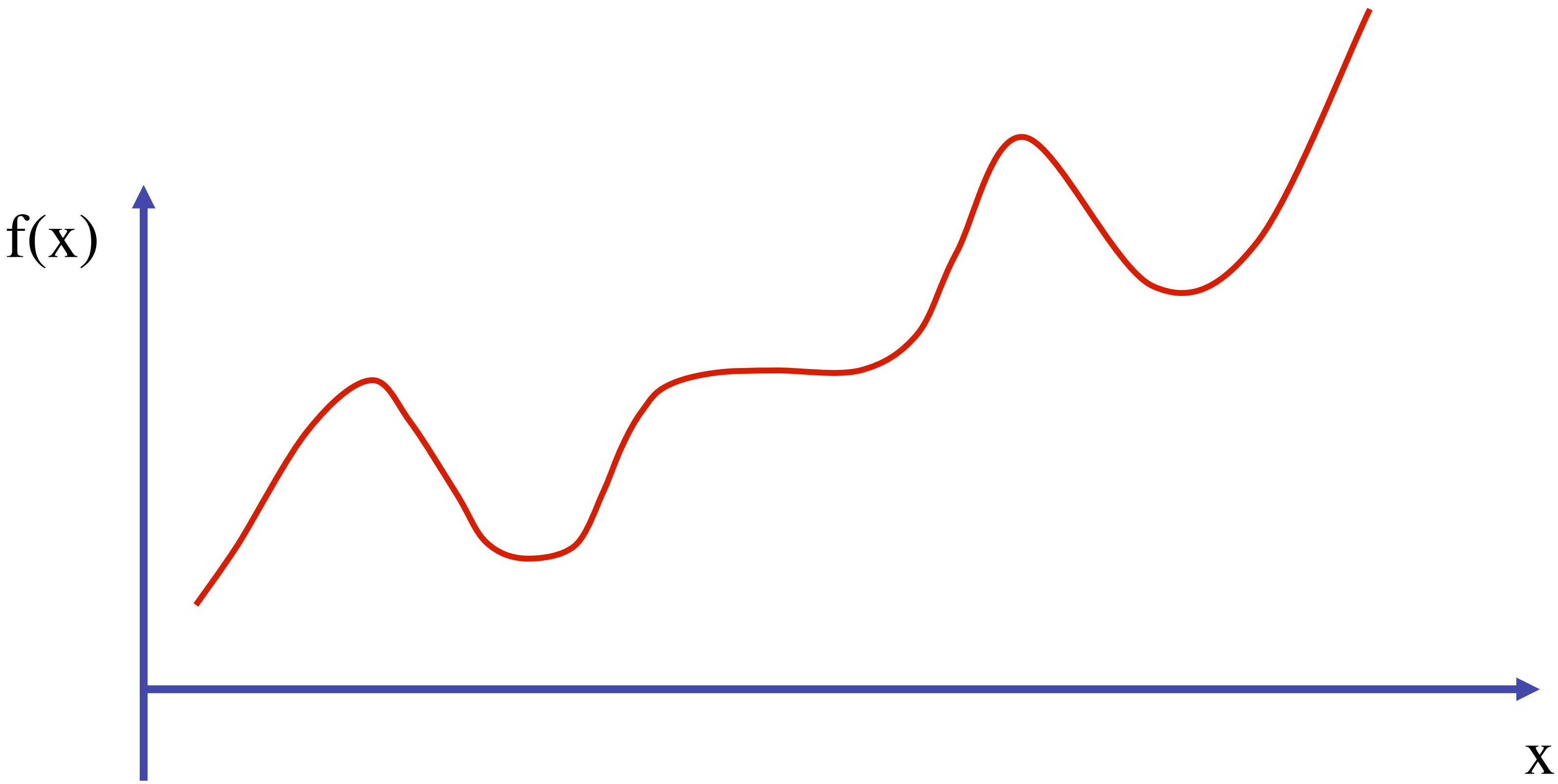
Example of Convex Function



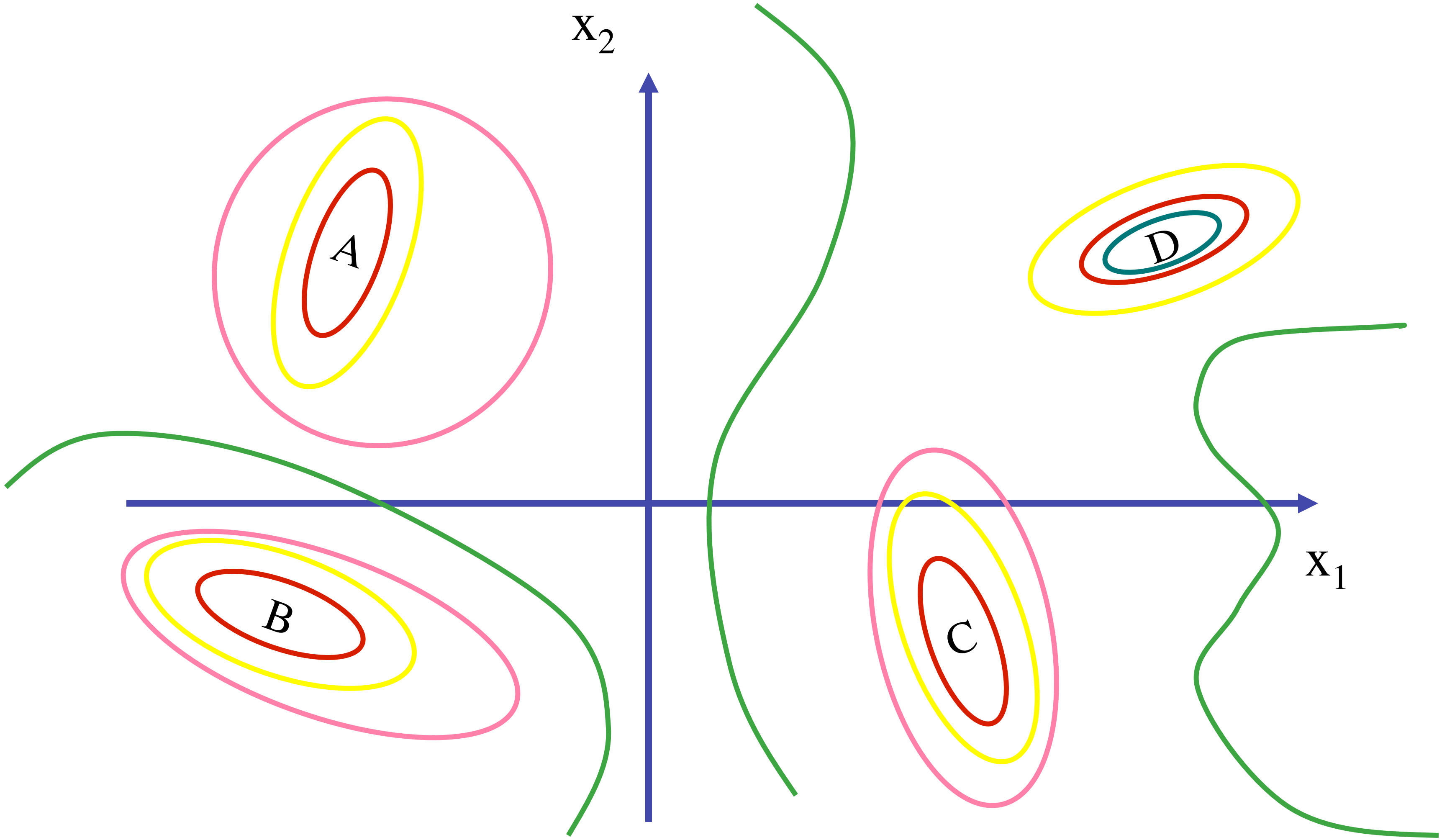
Example of Convex Function



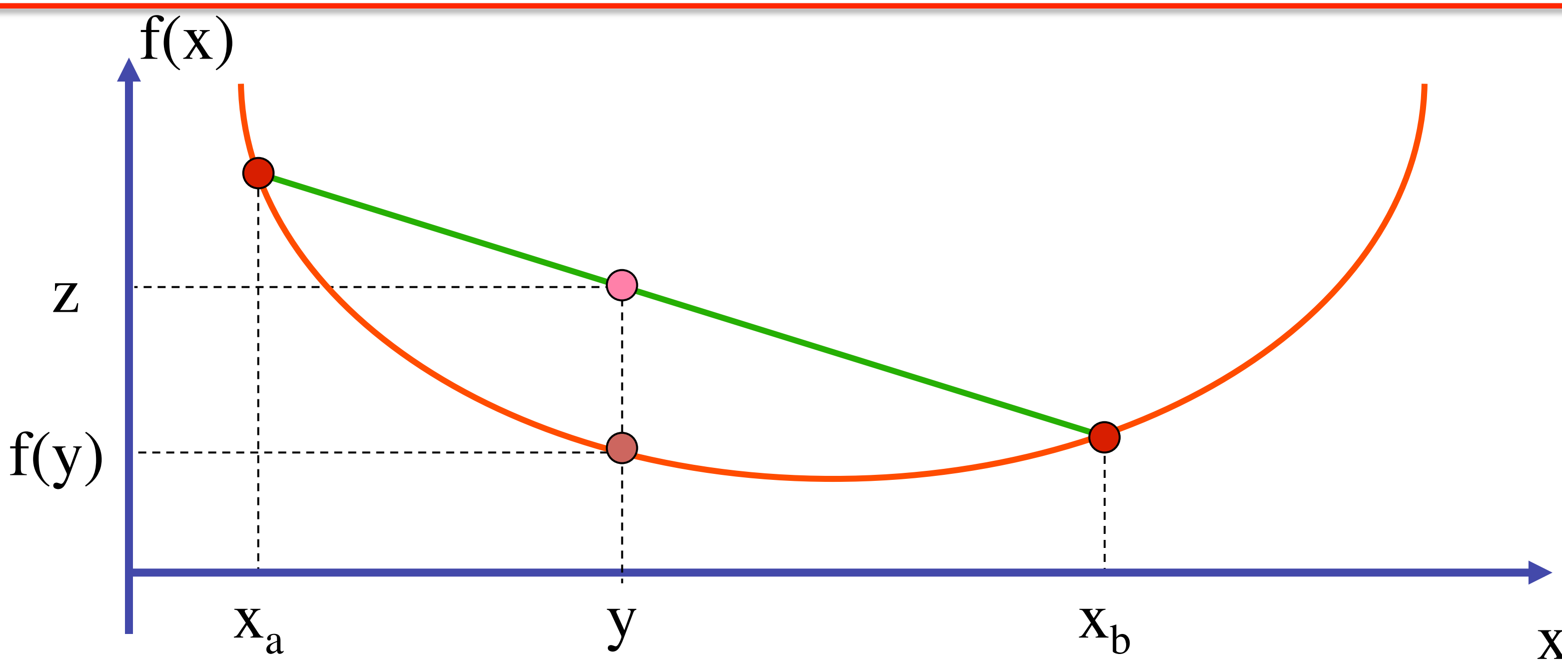
Example of Non-Convex Function



Example of Non-Convex Function



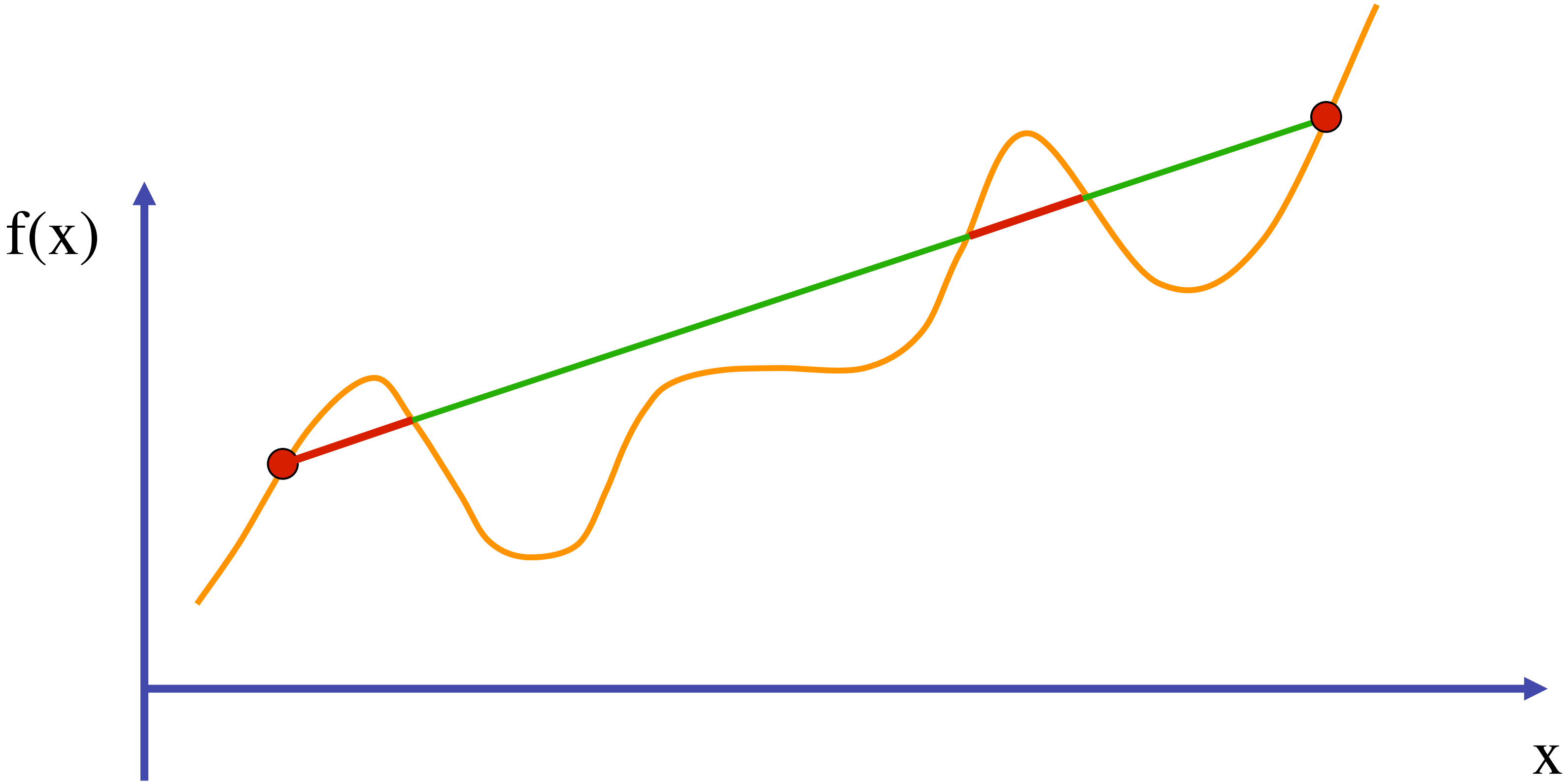
Definition of a Convex Function



A convex function is a function such that, for any two points x_a and x_b belonging to the feasible set and any *k* such that $0 \le k \le 1$, we have:

$$z = kf(x_a) + (1-k)f(x_b) \ge f(y) = f\left[kx_a + (1-k)x_b\right]$$

Example of Non-Convex Function



Importance of Convexity

- If we can prove that a minimization problem is convex:
  - Convex feasible set
  - Convex objective function

➔ Then, the problem has one and only one solution

- Proving convexity is often difficult
- Power system problems are usually not convex

➔ There may be more than one solution to power system optimization problems

Non-Linear Programming

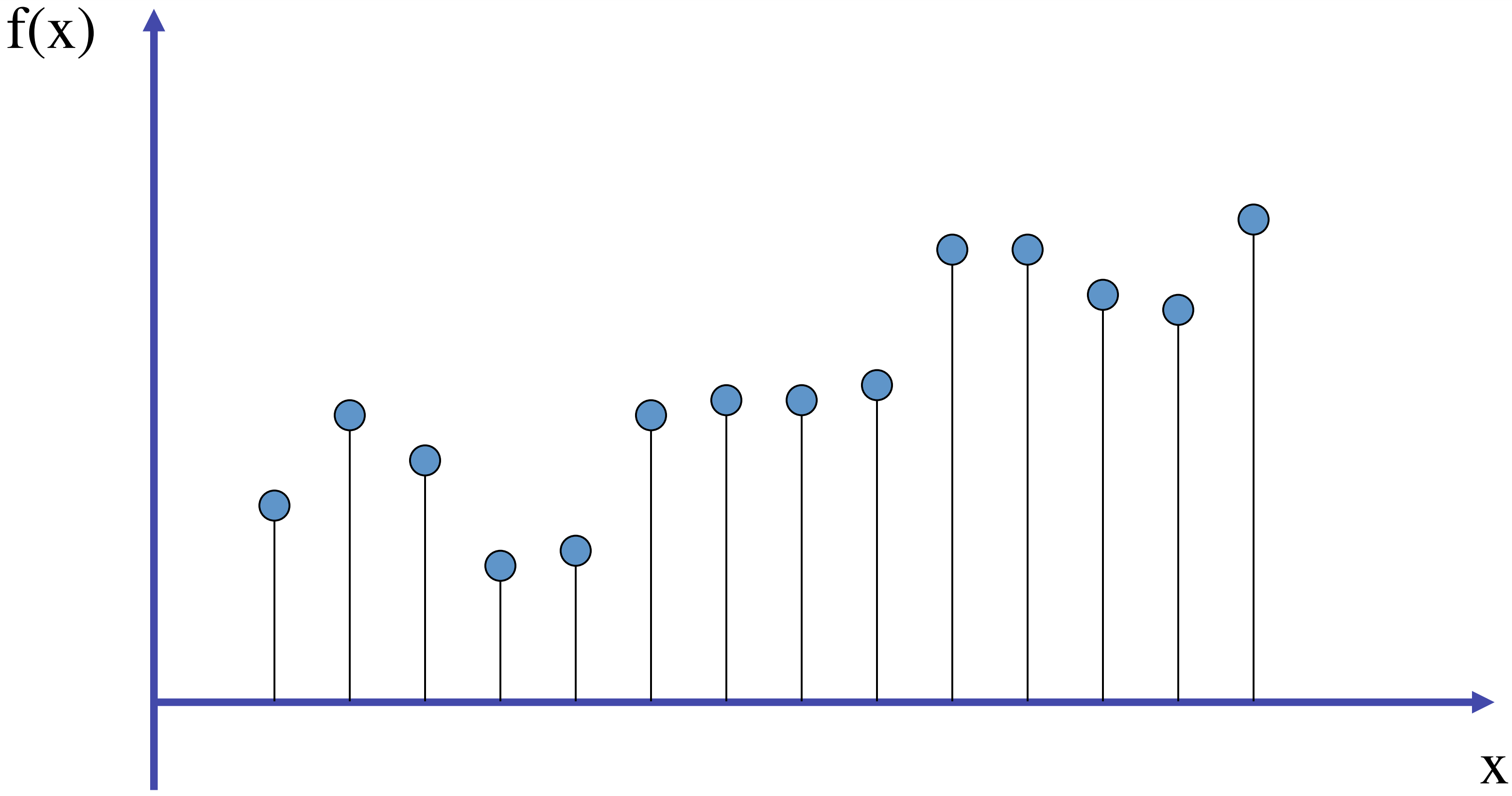
Motivation

- Method of Lagrange multipliers
  - Very useful insight into solutions
  - Analytical solution practical only for small problems
  - Direct application not practical for real-life problems because these problems are too large
  - Difficulties when problem is non-convex
- Often need to search for the solution of practical optimization problems using:
  - Objective function only or
  - Objective function and its first derivative or
  - Objective function and its first and second derivatives

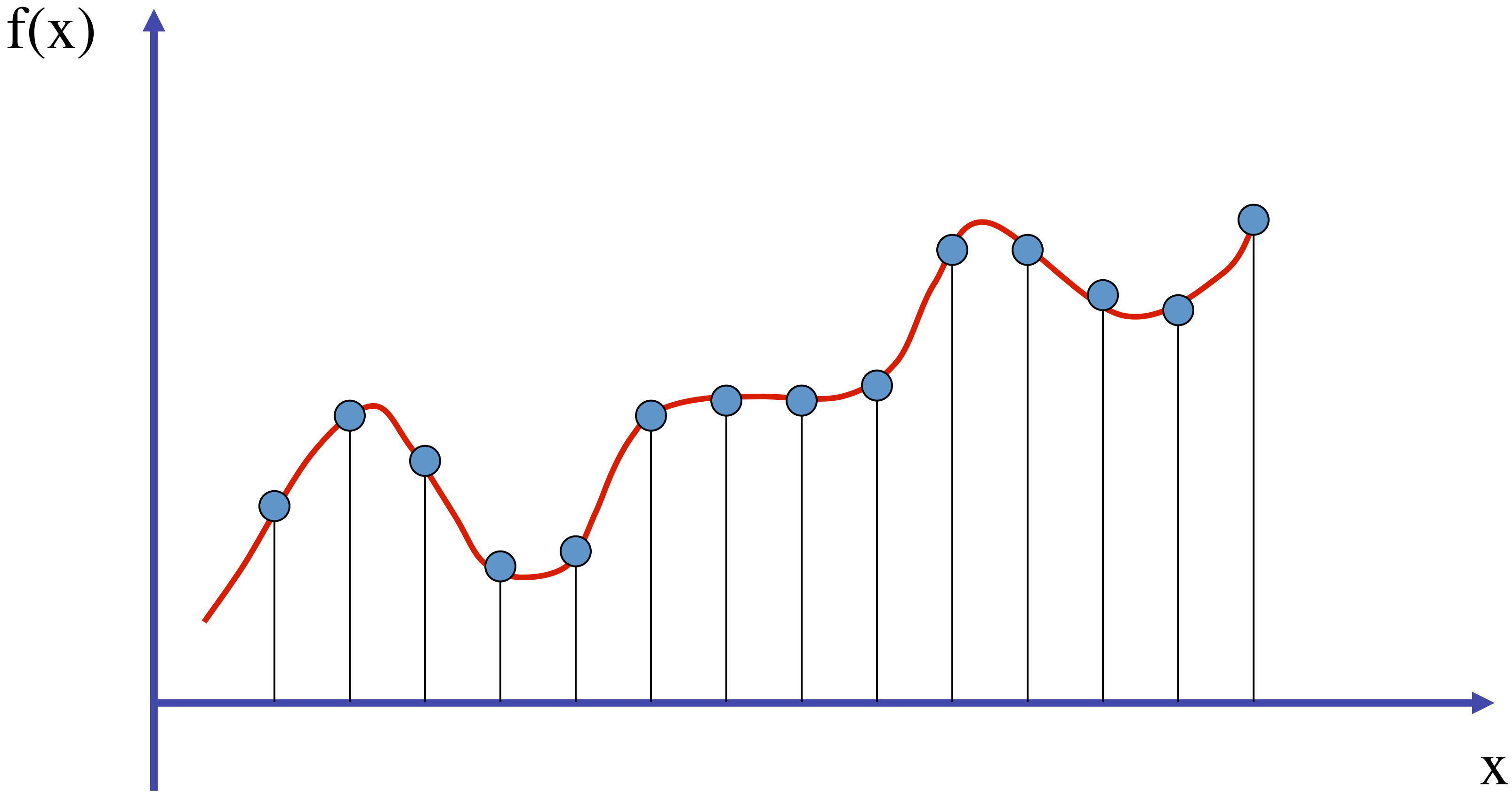
Naïve One-Dimensional Search

- Suppose:
  - That we want to find the value of x that minimizes $f(x)$
  - That the only thing that we can do is calculate the value of $f(x)$ for any value of x
- We could calculate $f(x)$ for a range of values of x and choose the one that minimizes $f(x)$

Naïve One-Dimensional Search

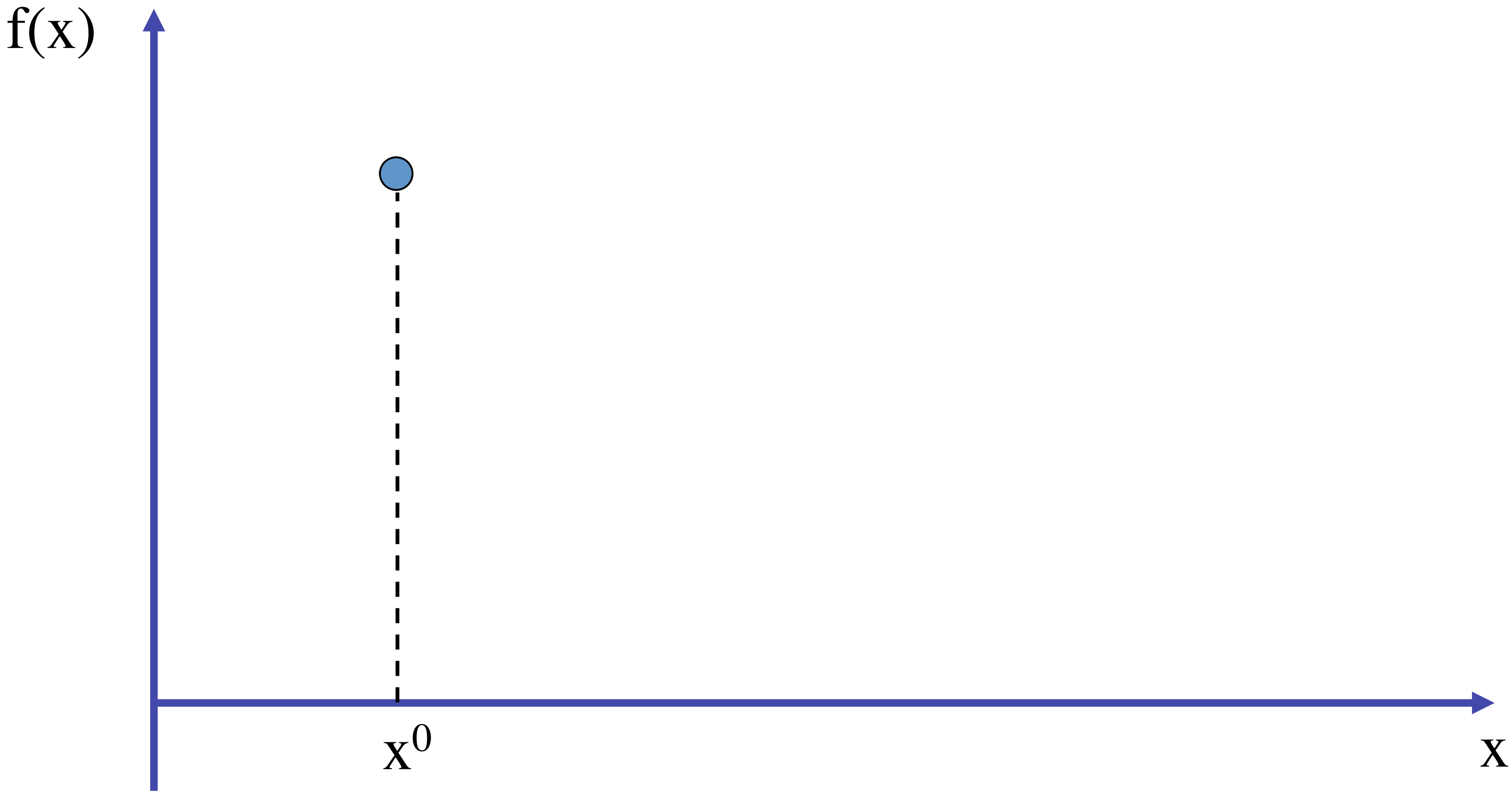


Naïve One-Dimensional Search

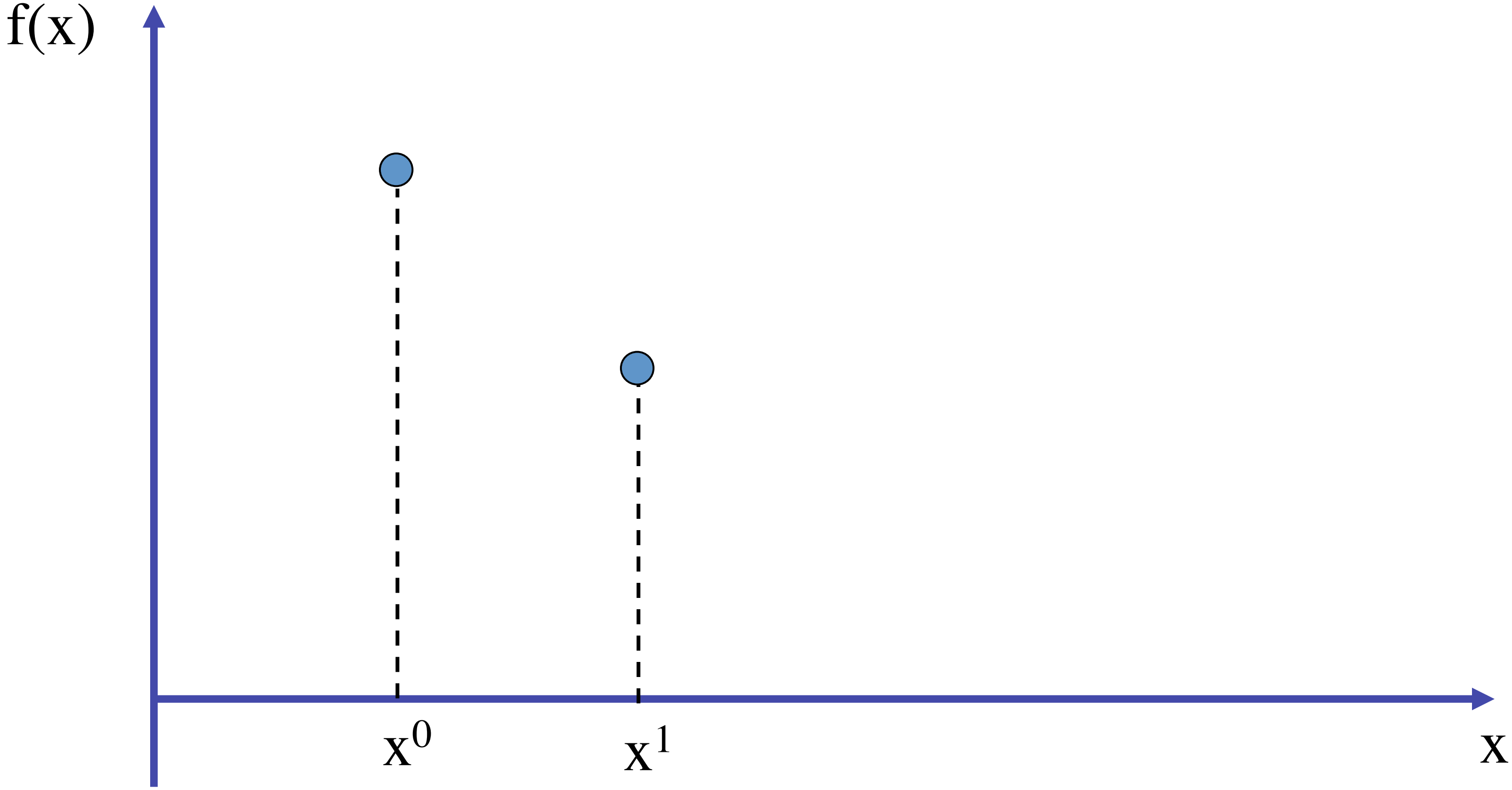


- Requires a considerable amount of computing time if range is large and a good accuracy is needed

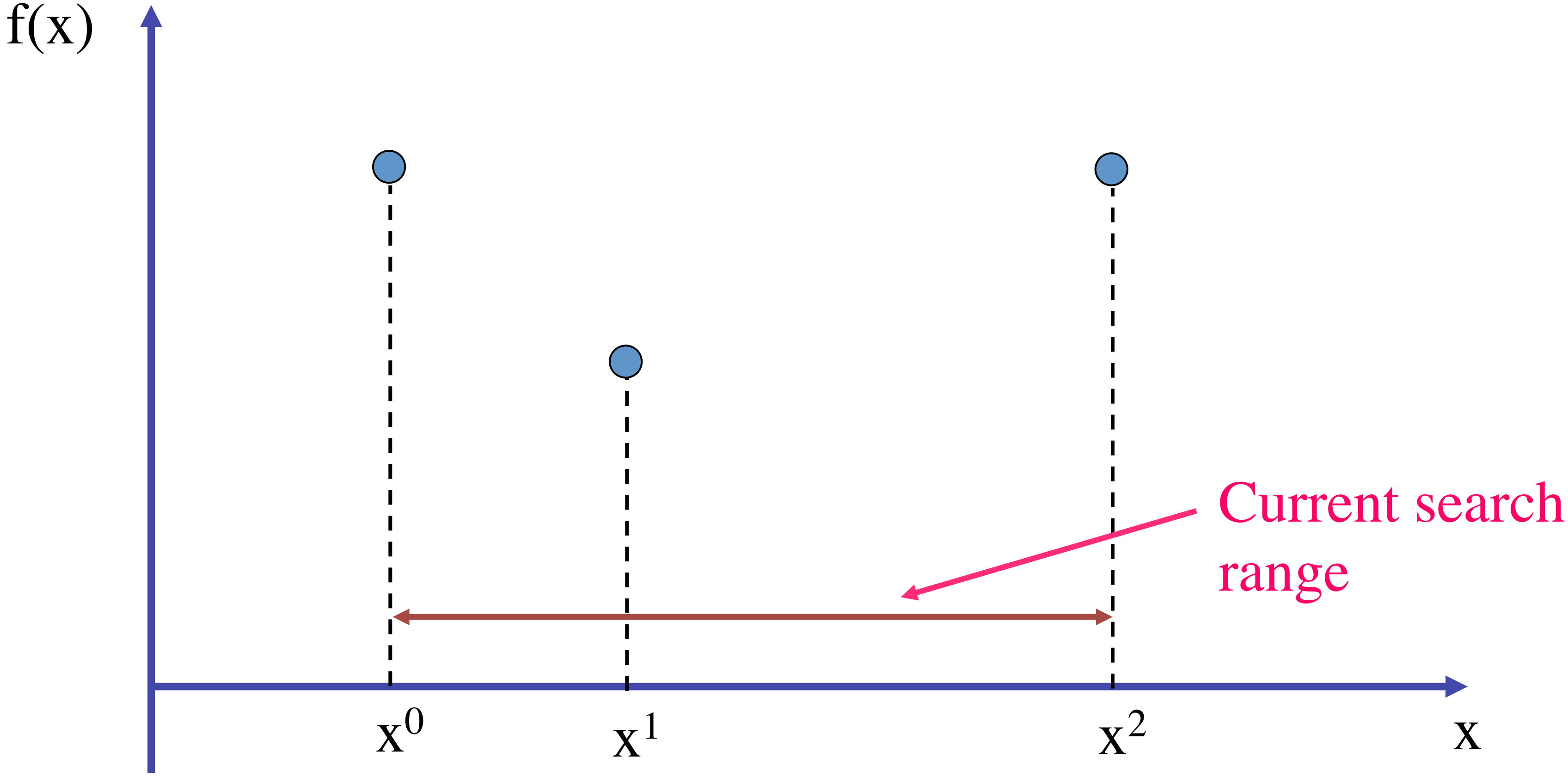
One-Dimensional Search



One-Dimensional Search

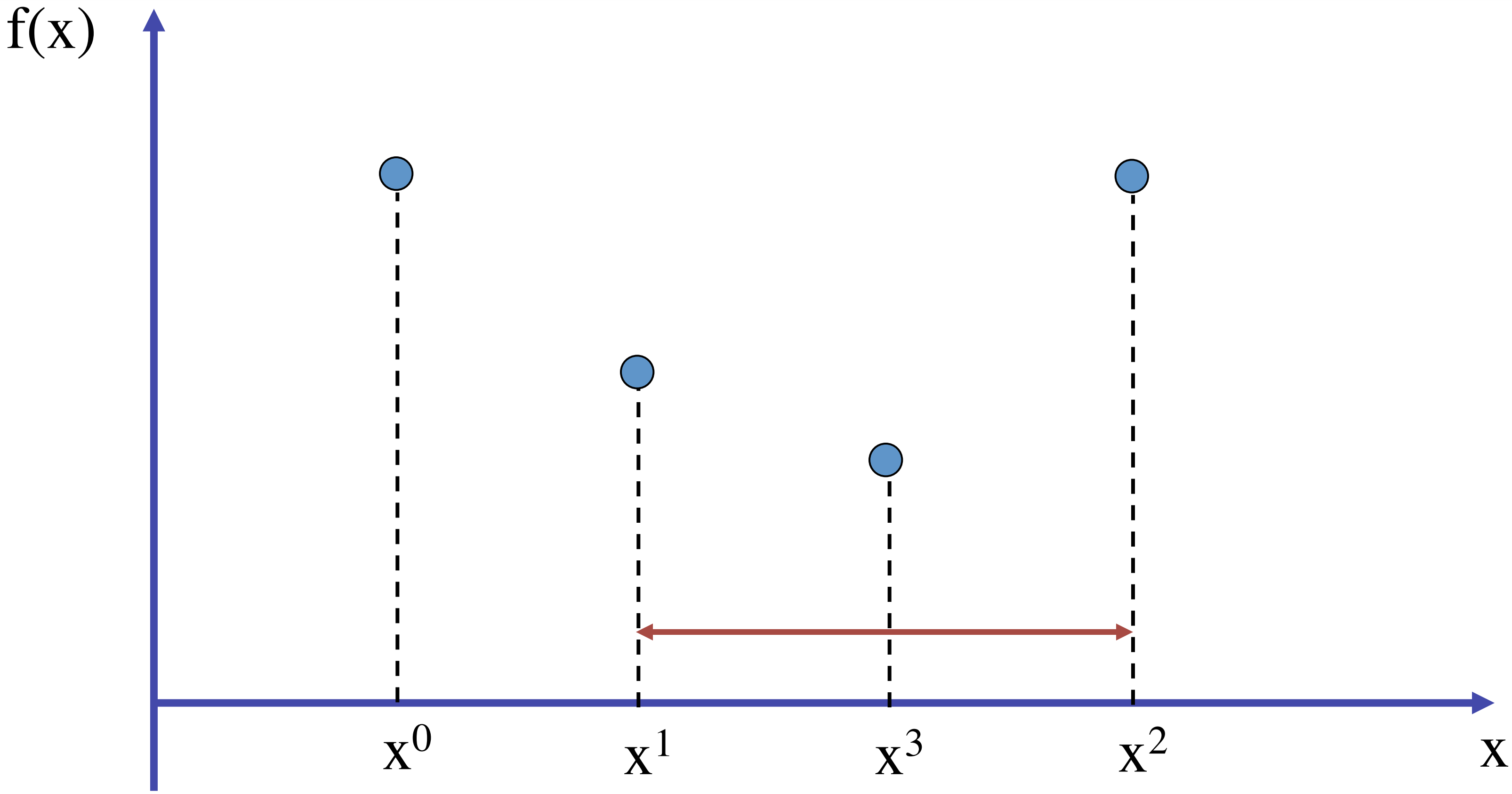


One-Dimensional Search



If the function is convex, we have bracketed the optimum

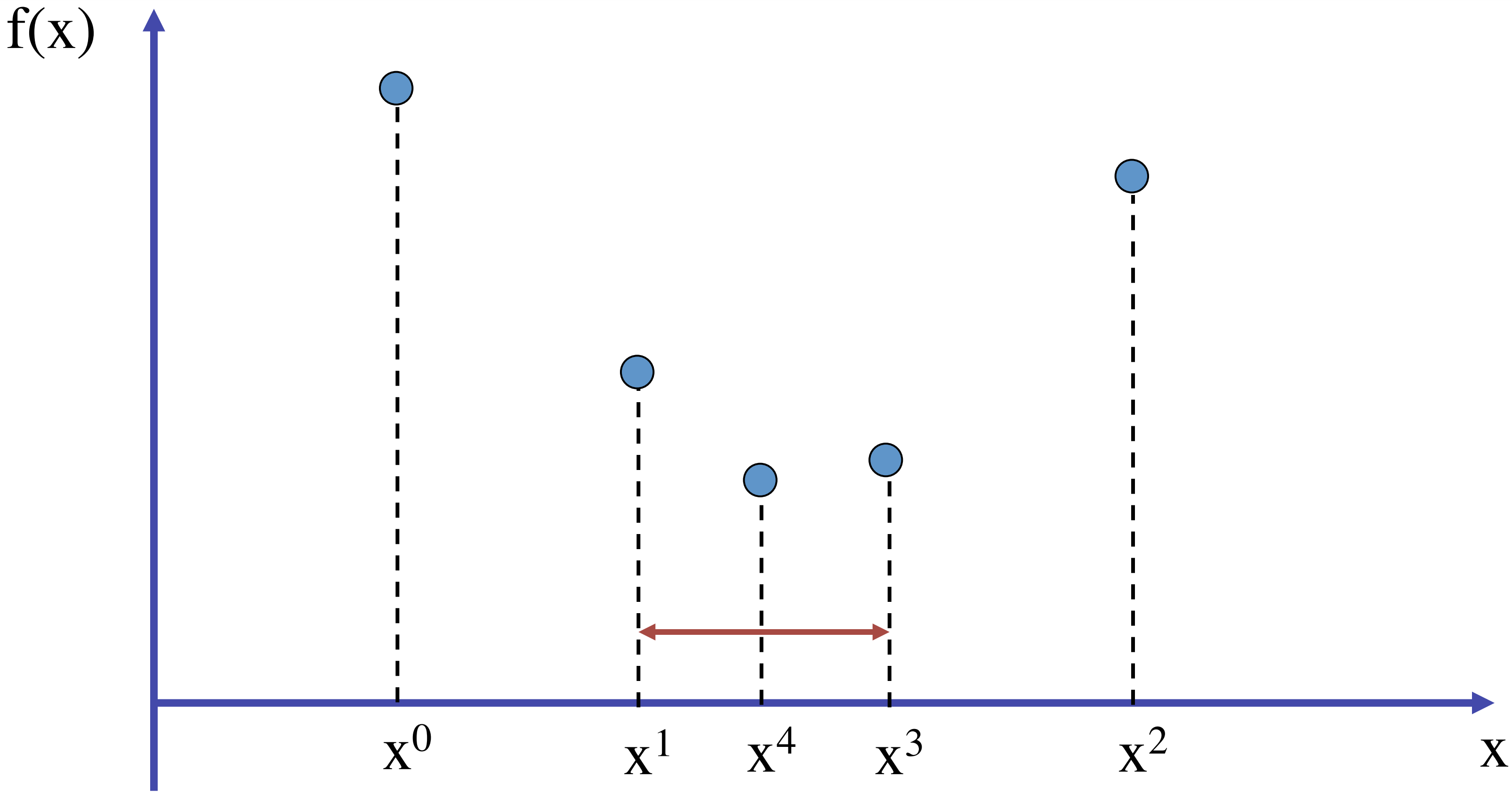
One-Dimensional Search



Optimum is between x^1 and x^2

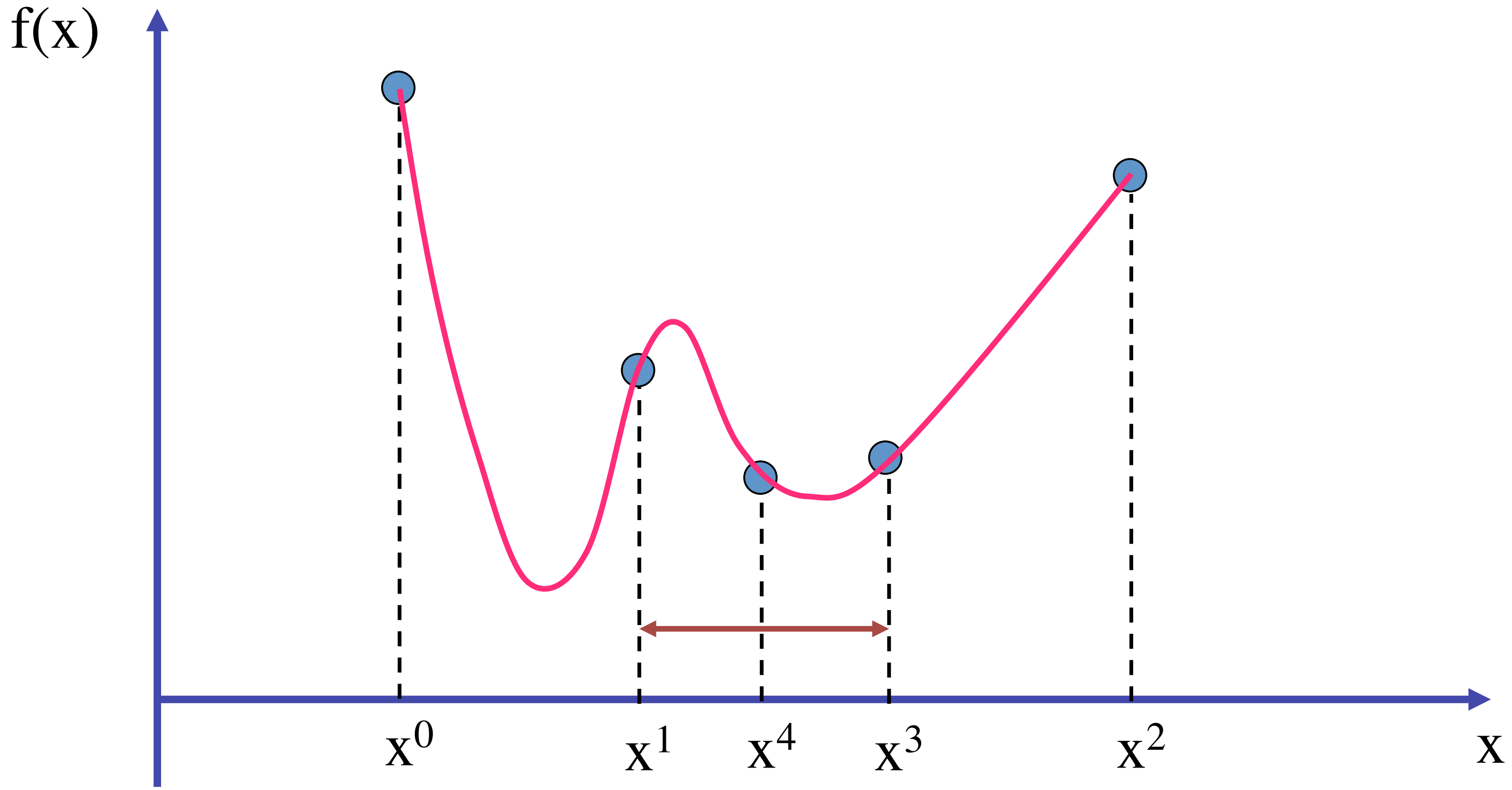
We do not need to consider x^0 anymore

One-Dimensional Search



Repeat the process until the range is sufficiently small

One-Dimensional Search

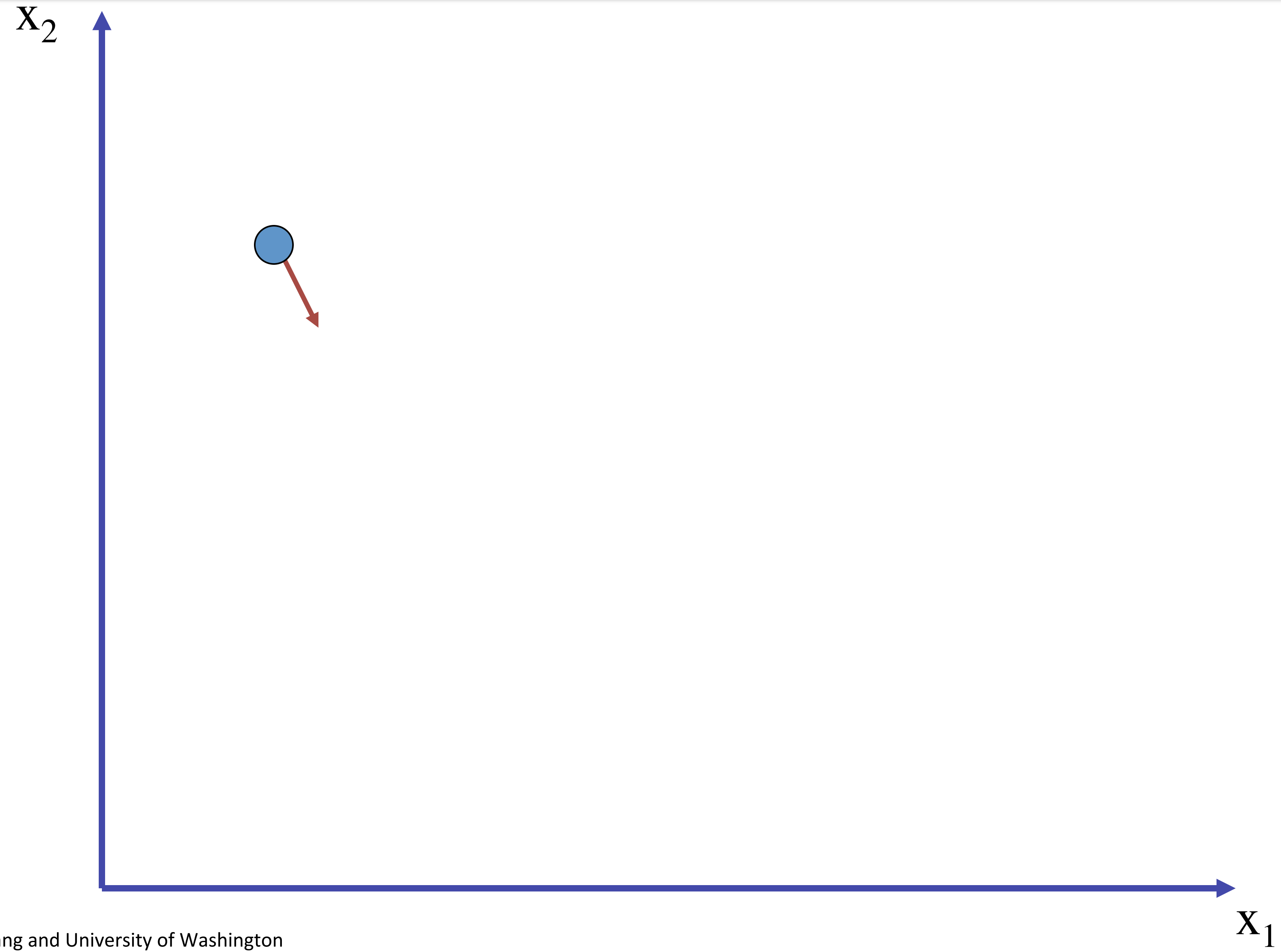


The procedure is valid only if the function is convex!

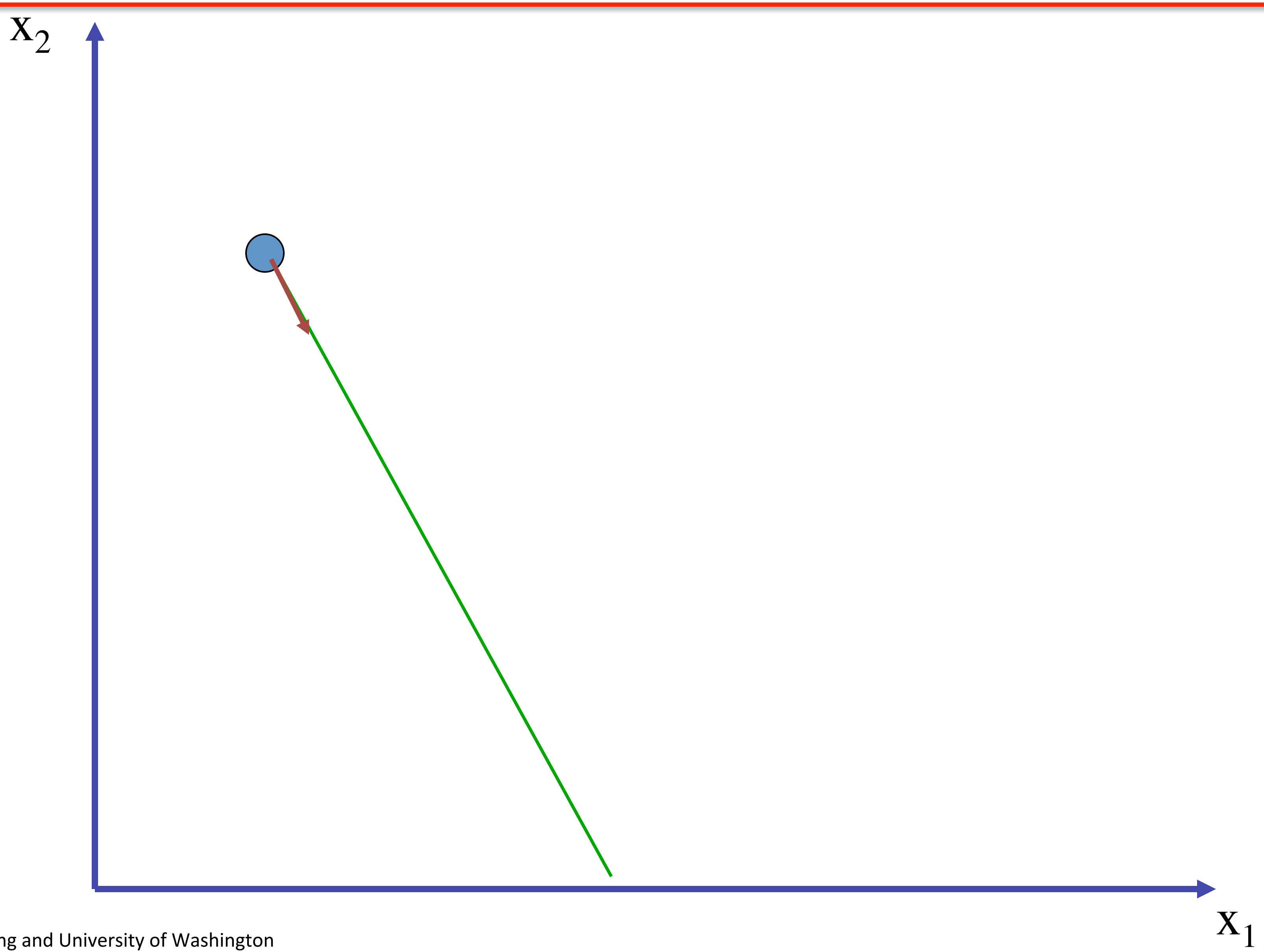
Multi-Dimensional Search

- Unidirectional search not applicable
- Naïve search becomes totally impossible as dimension of the problem increases
- If we can calculate the first derivatives of the objective function, much more efficient searches can be developed
- The gradient of a function gives the direction in which it increases/decreases fastest

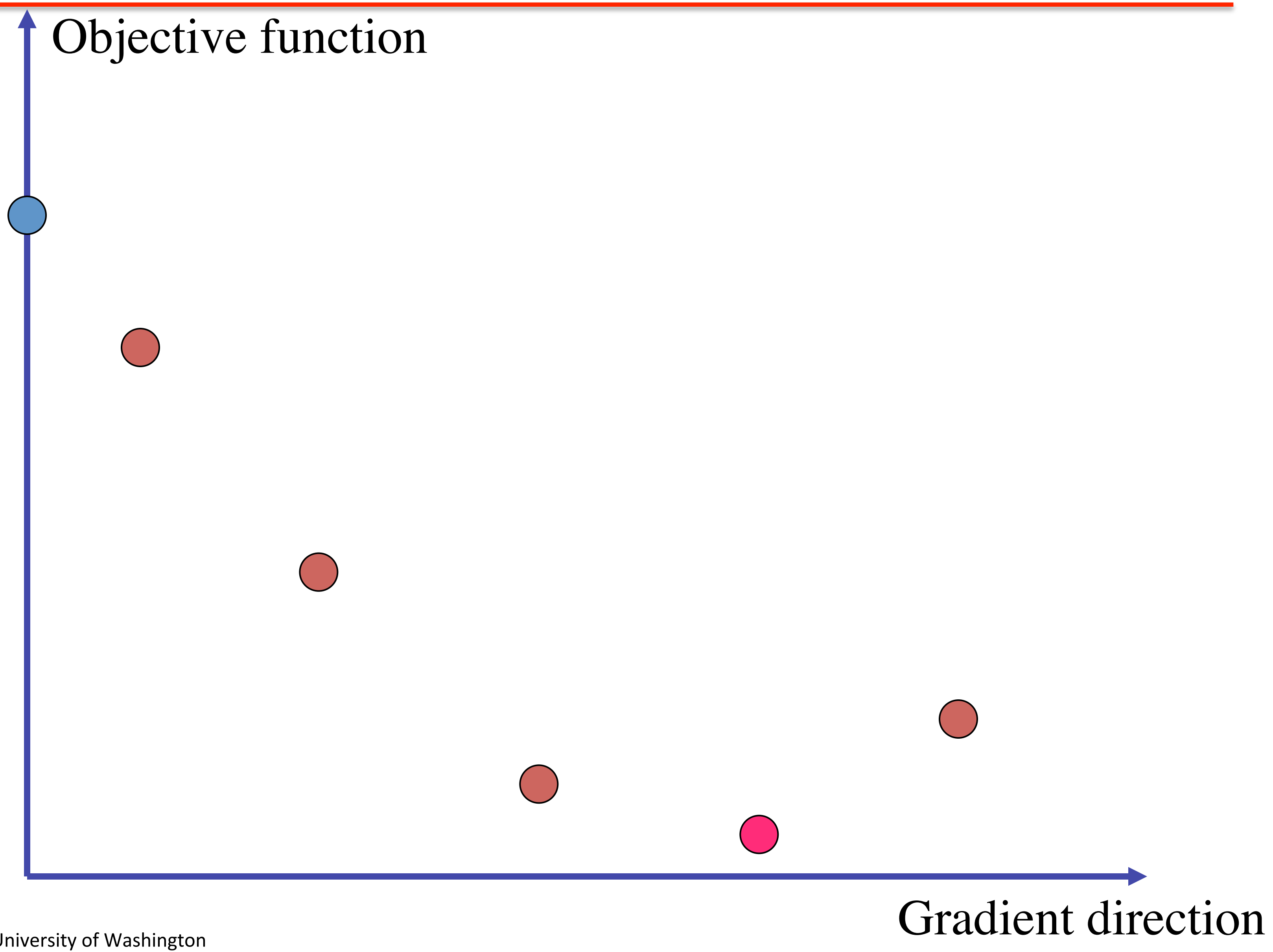
Steepest Ascent Algorithm



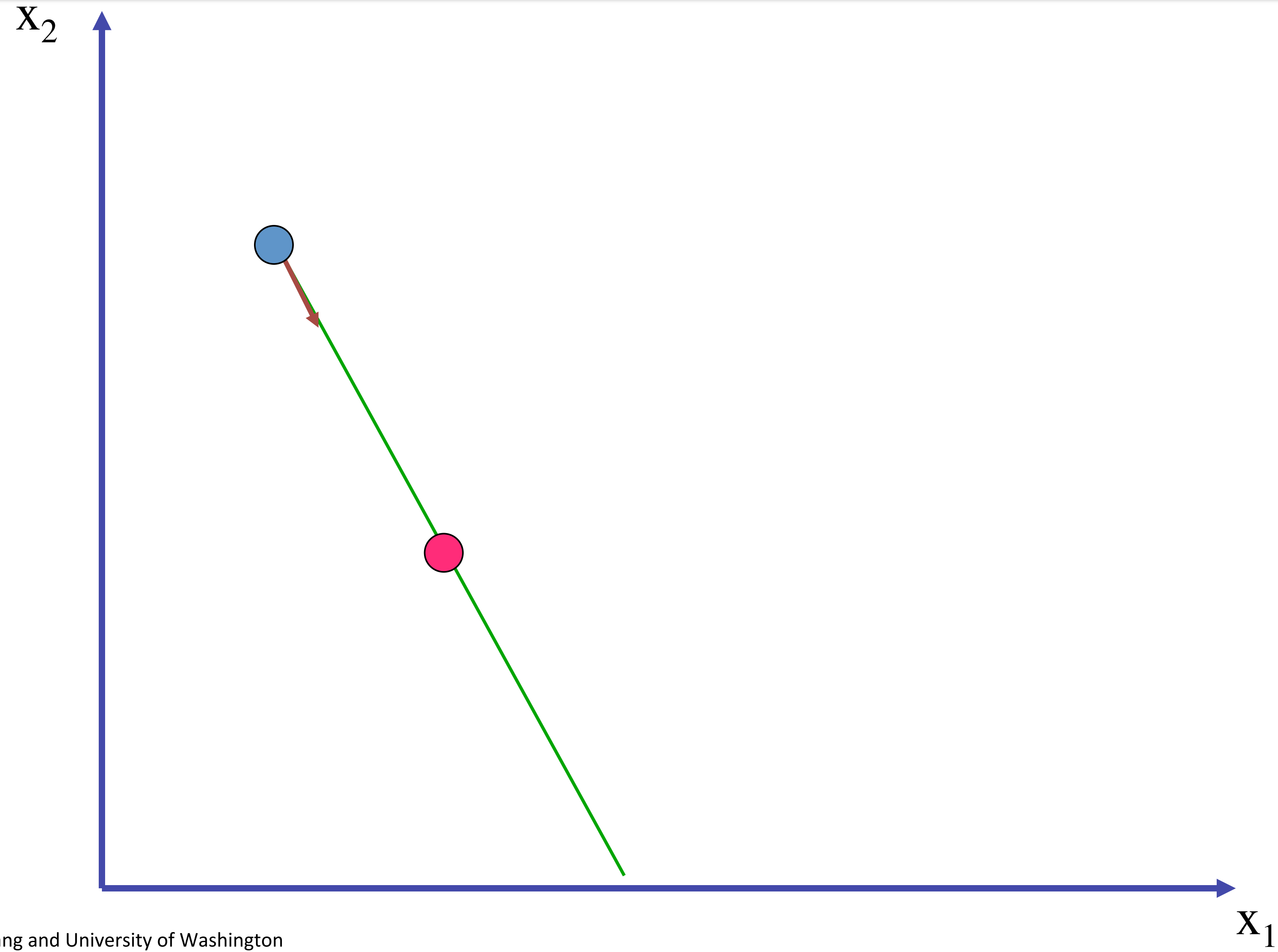
Steepest Ascent Algorithm



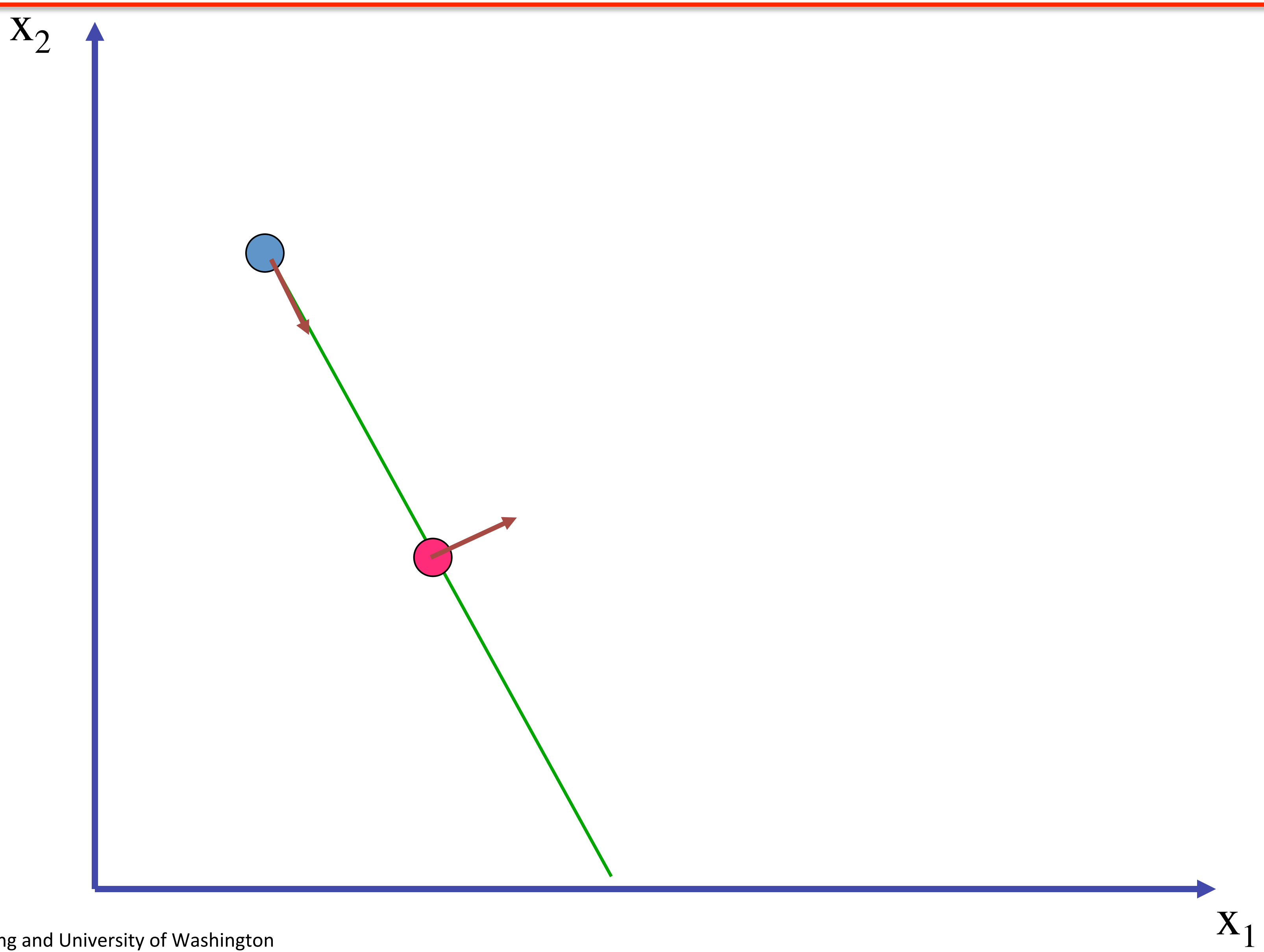
Unidirectional Search



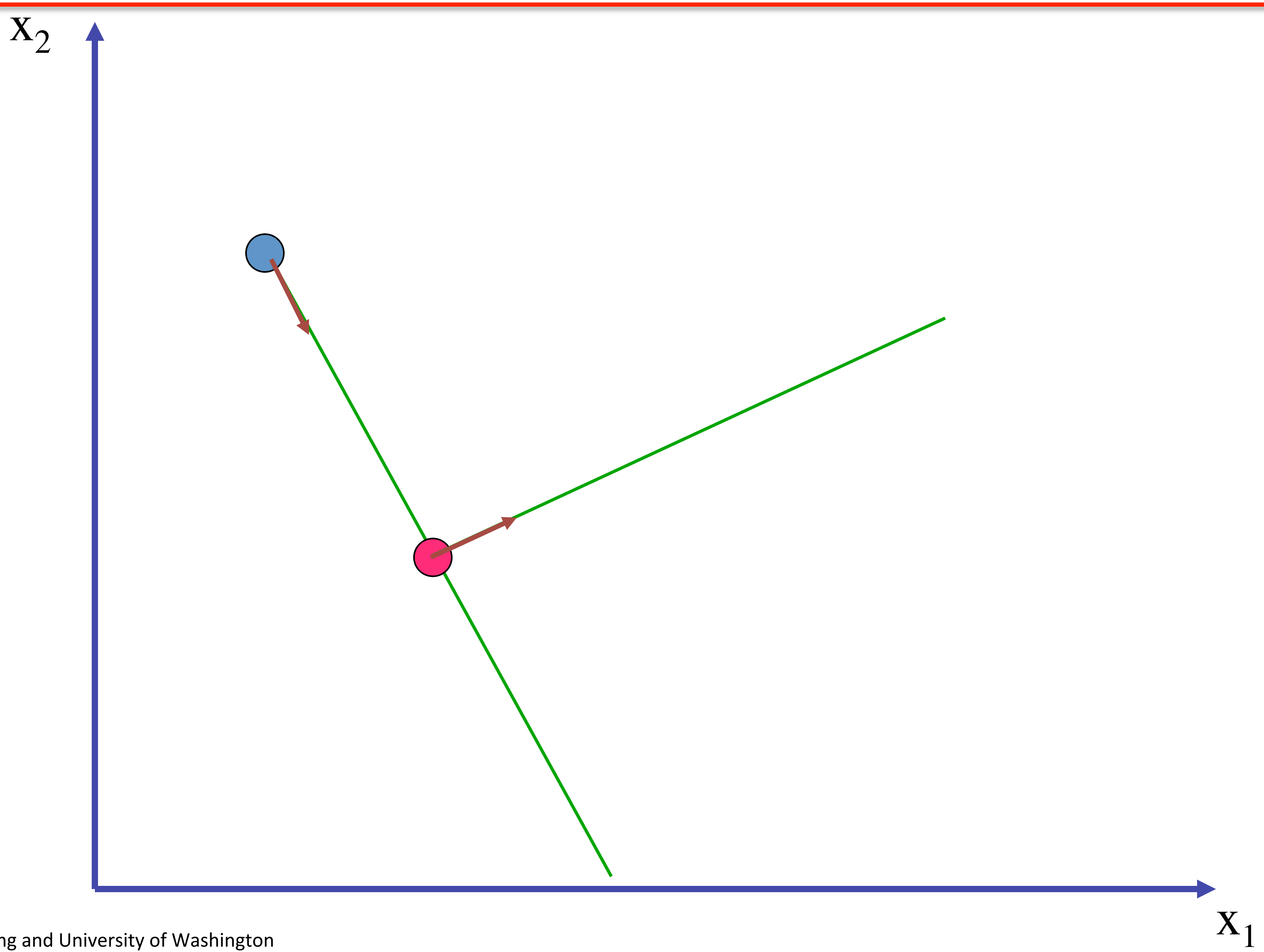
Steepest Ascent Algorithm



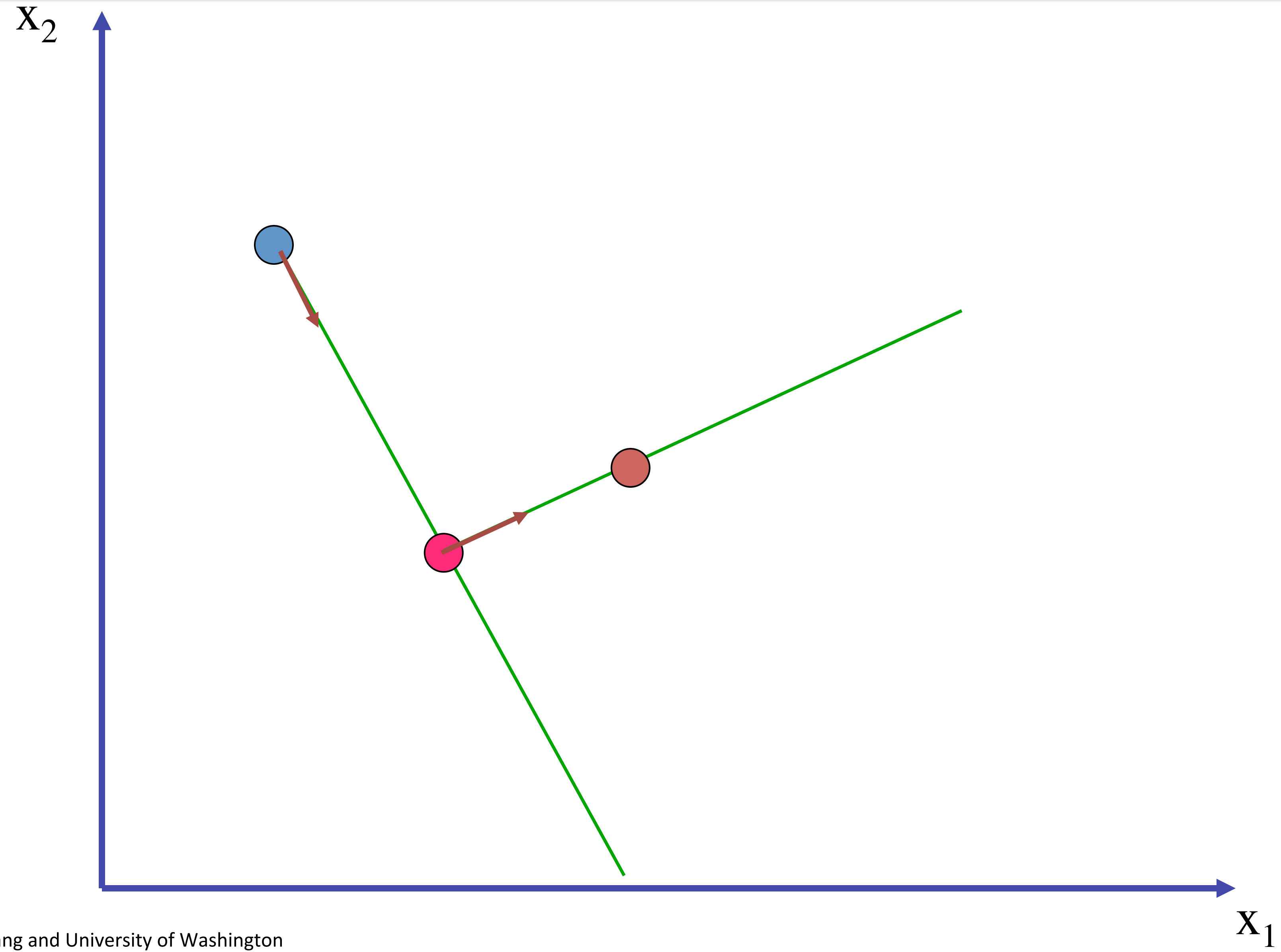
Steepest Ascent Algorithm



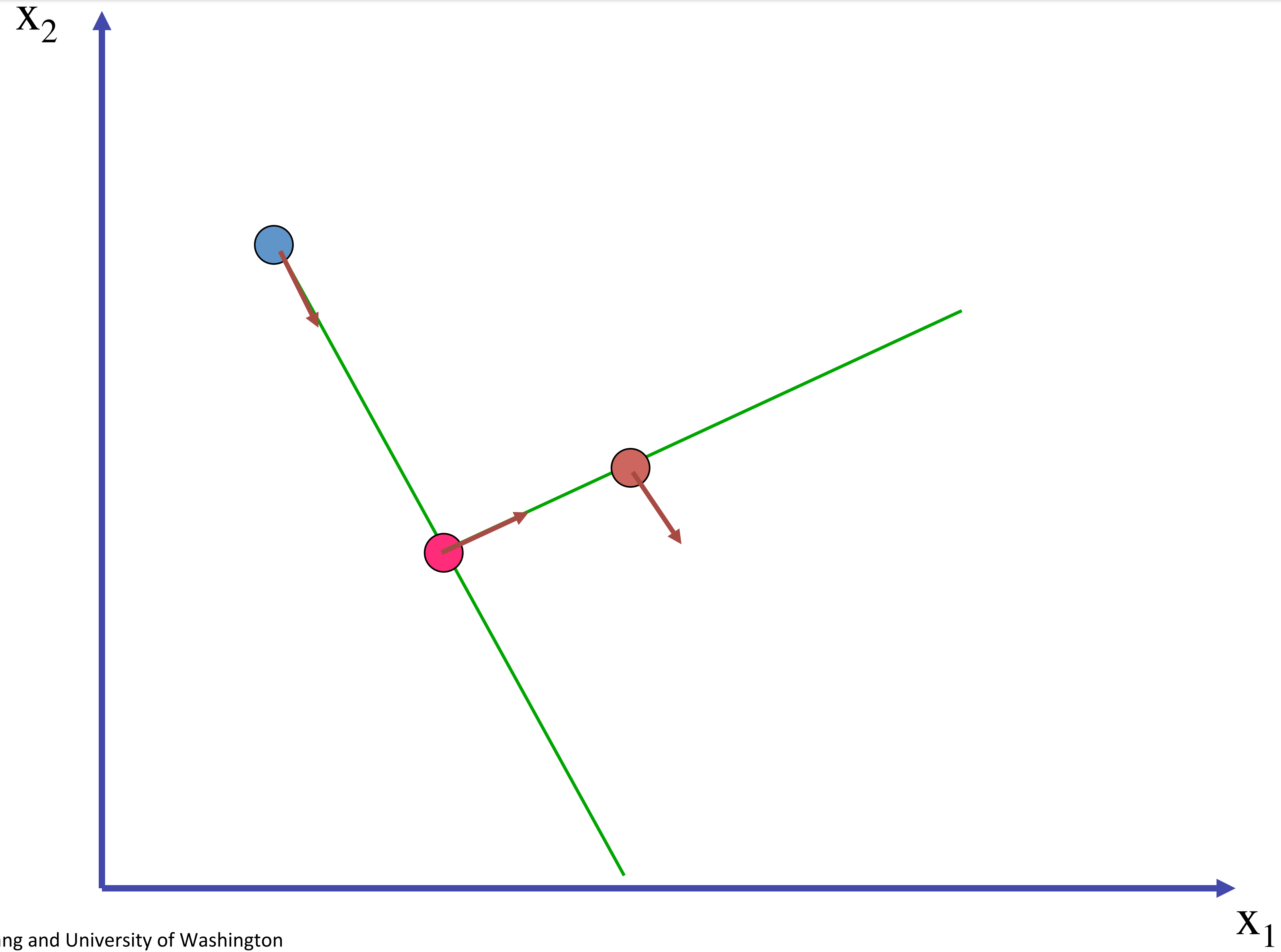
Steepest Ascent Algorithm



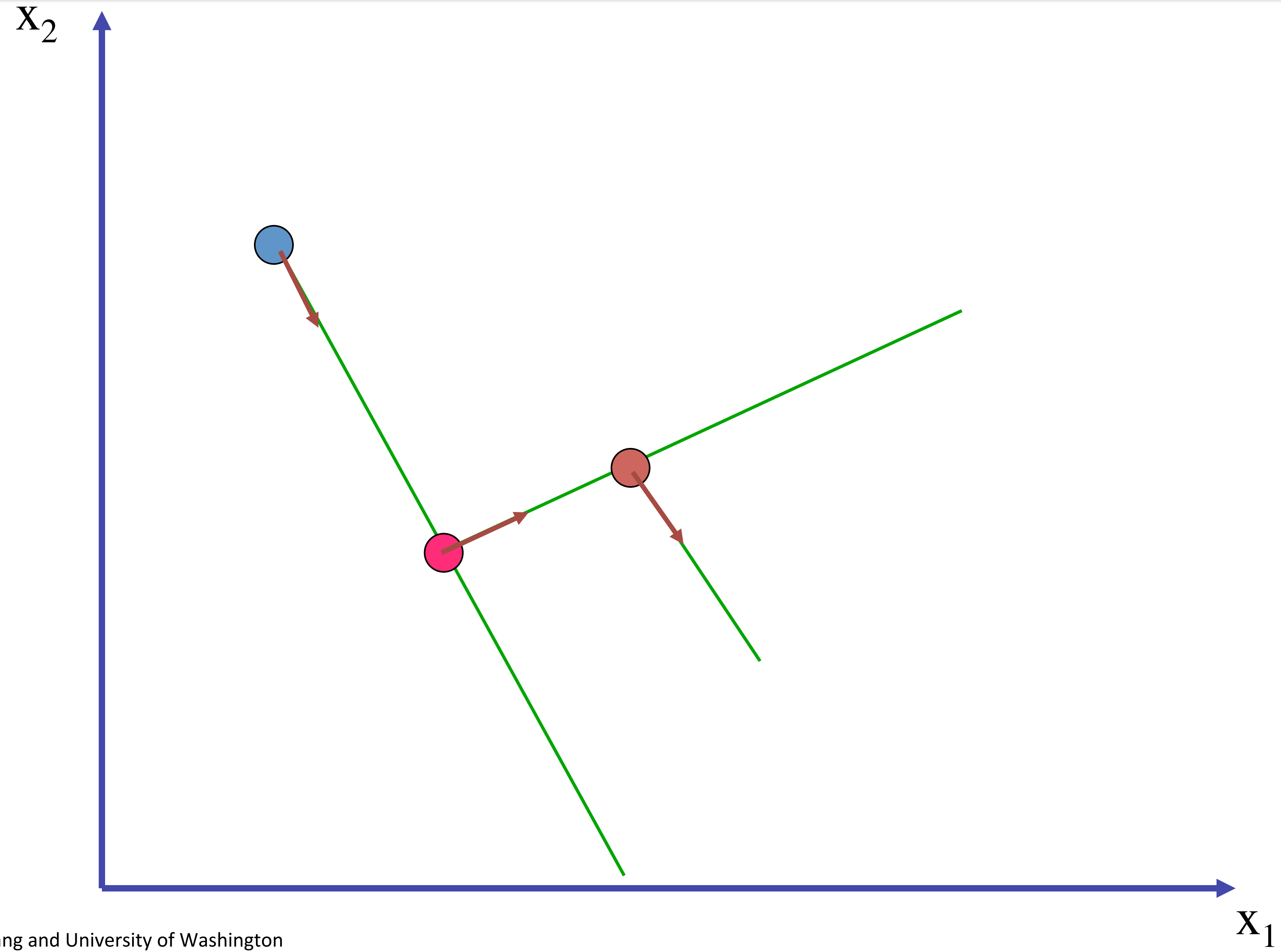
Steepest Ascent Algorithm



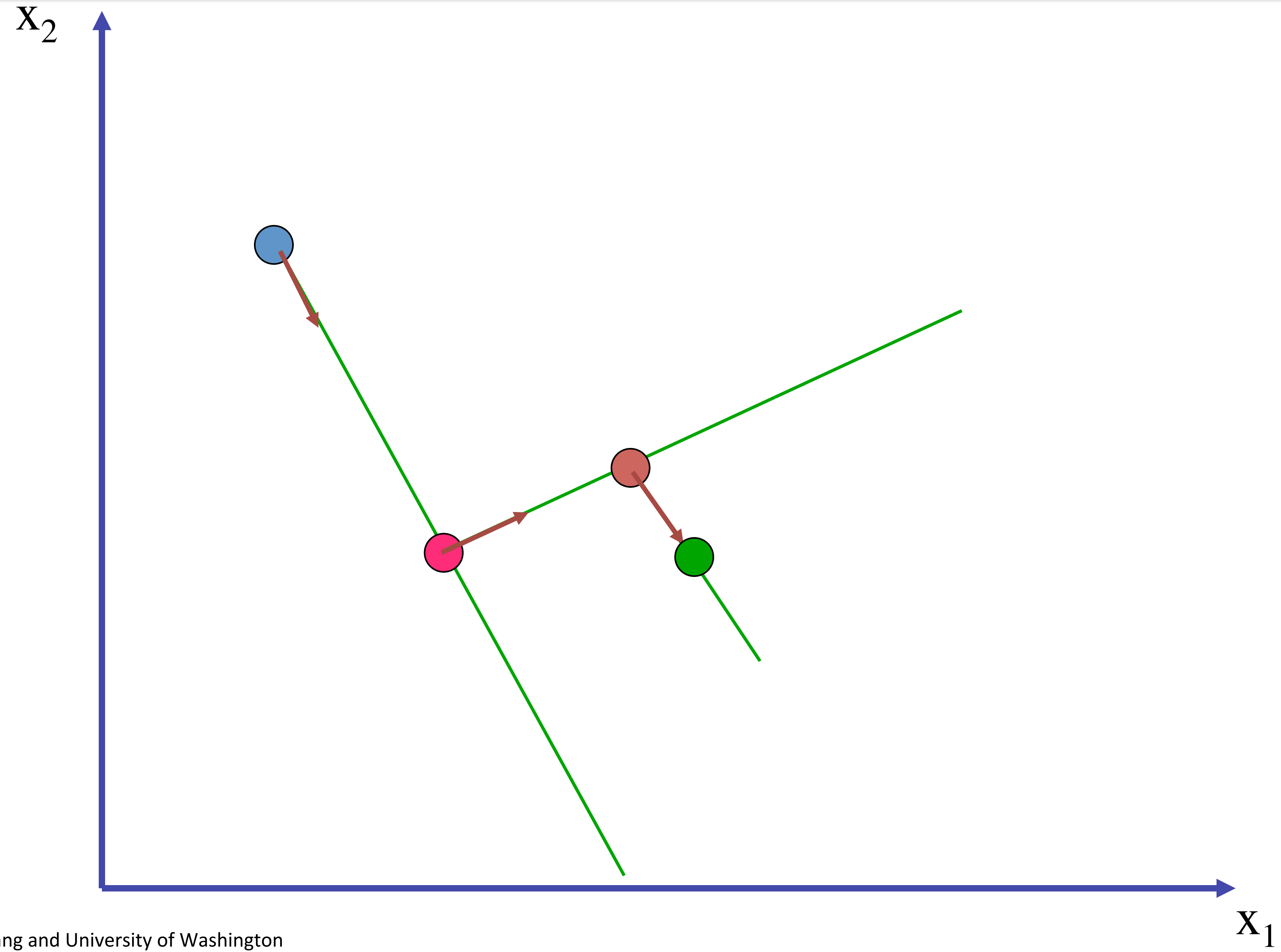
Steepest Ascent Algorithm



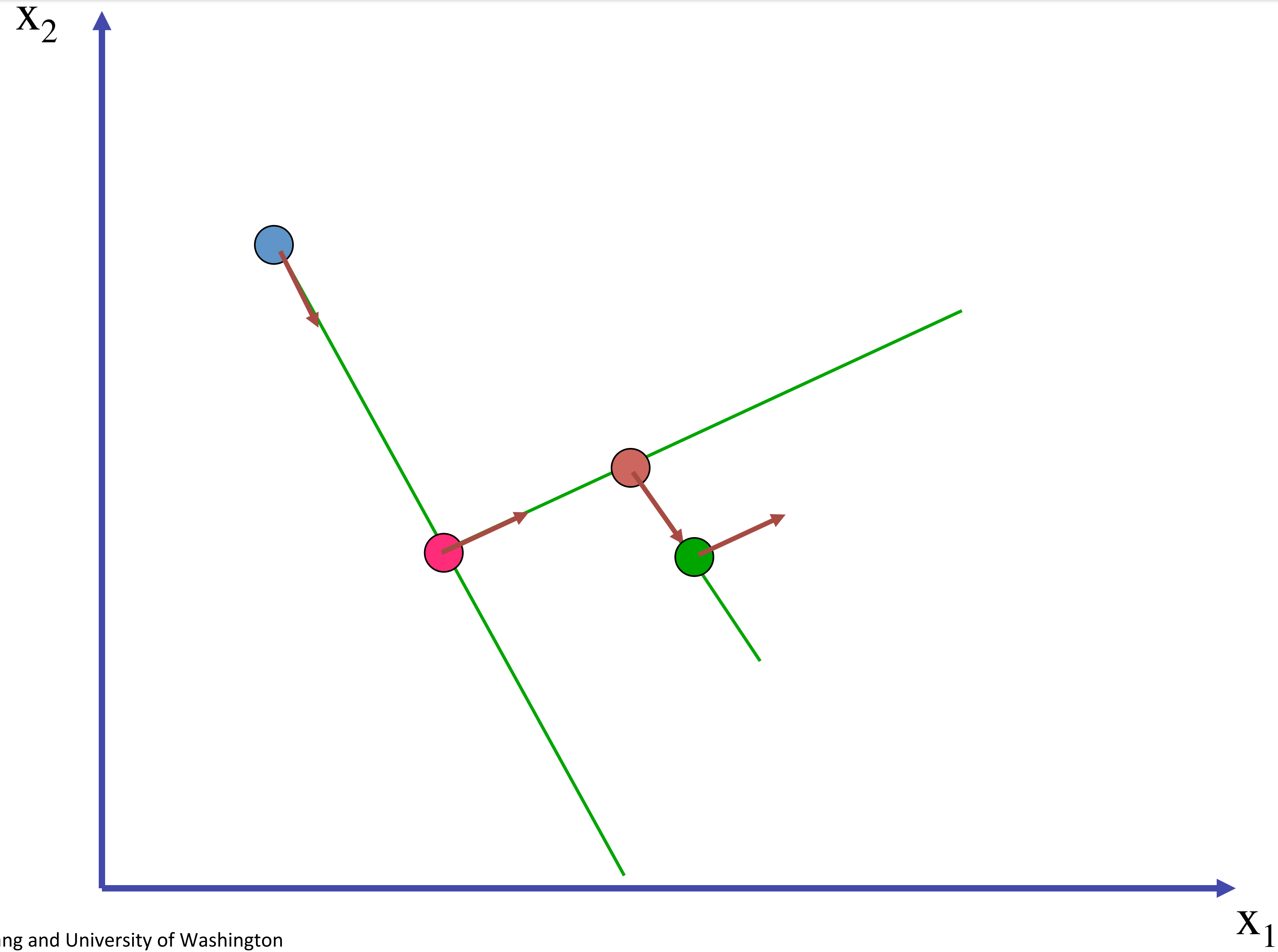
Steepest Ascent Algorithm



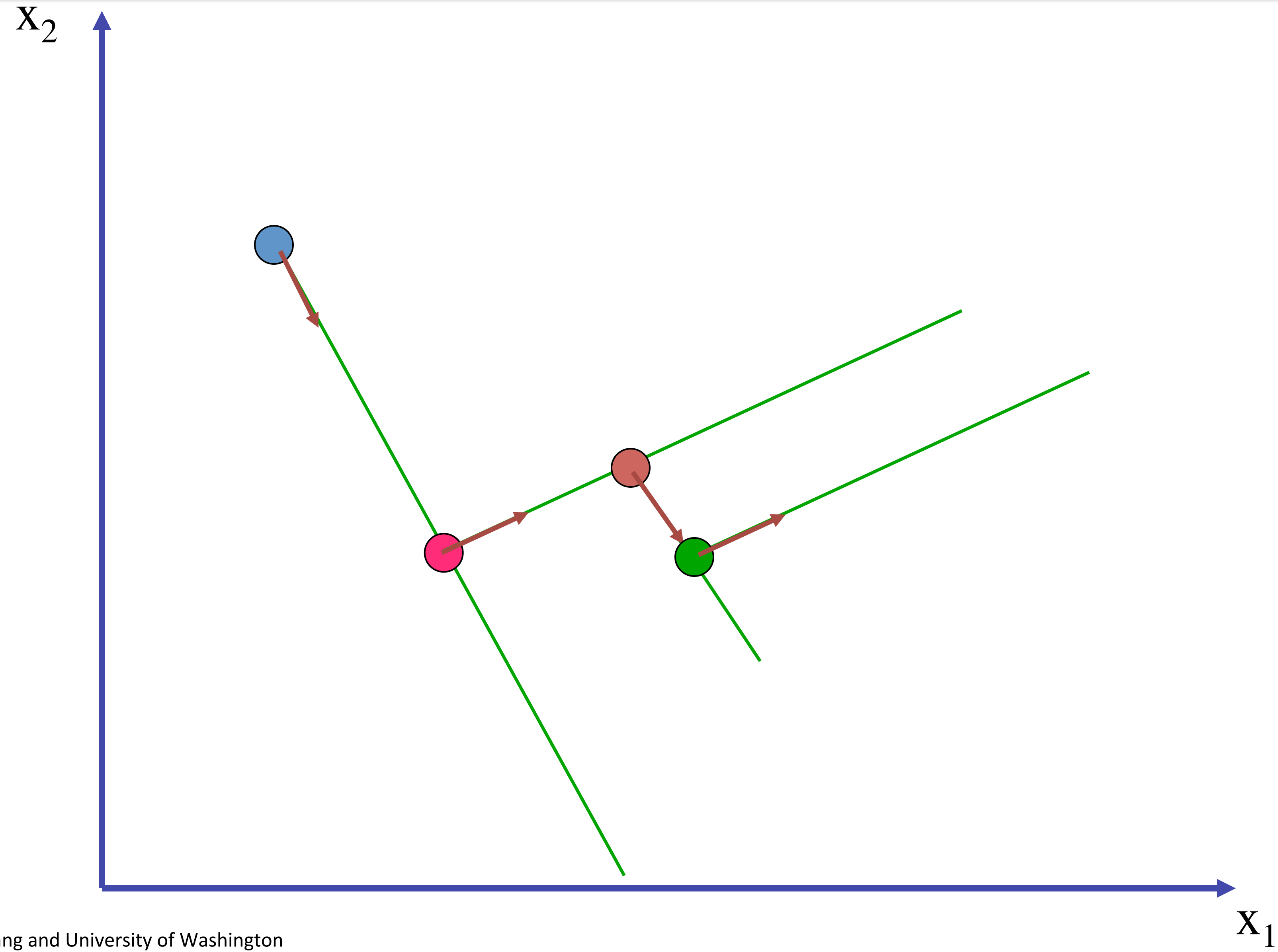
Steepest Ascent Algorithm



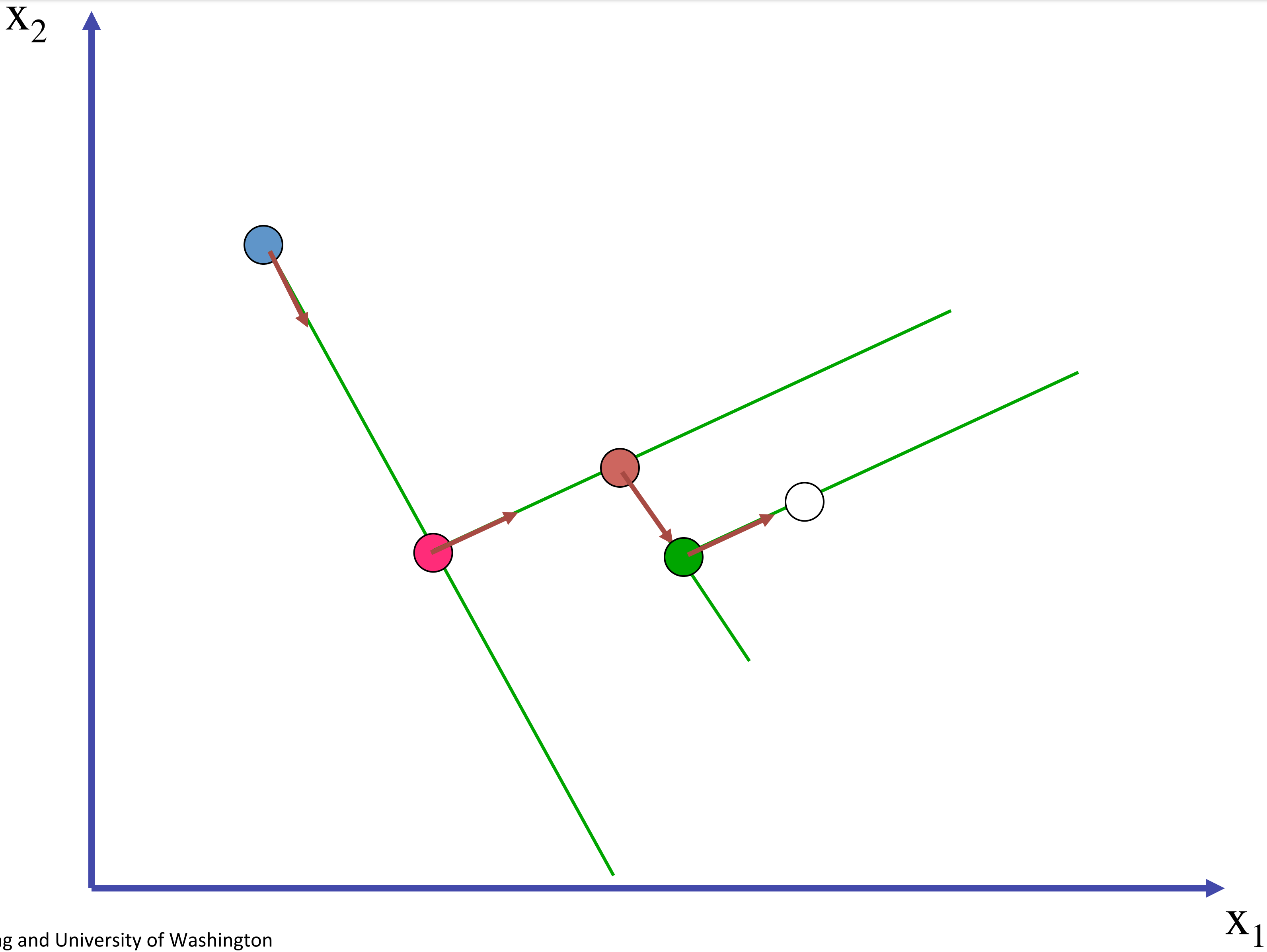
Steepest Ascent Algorithm



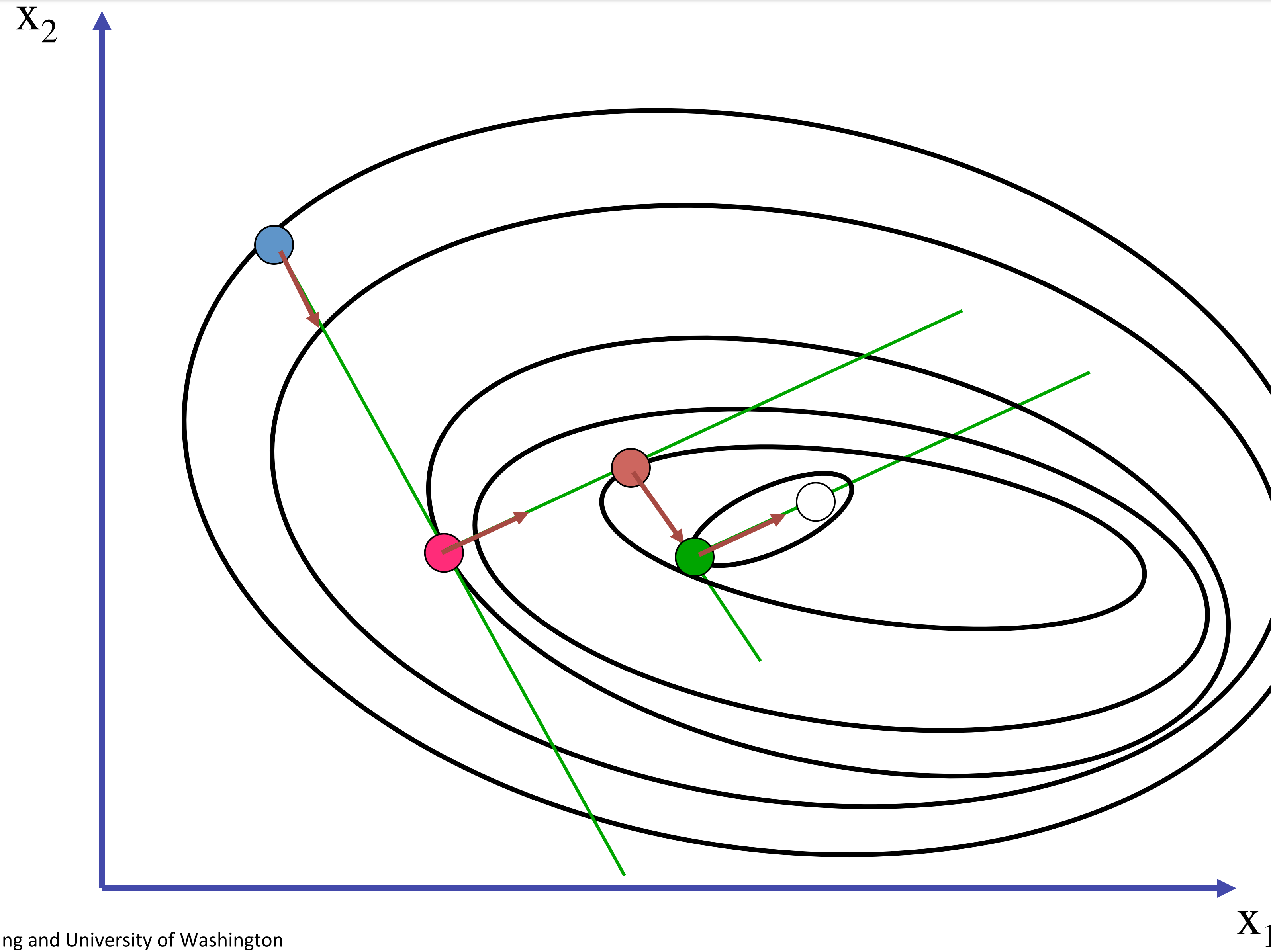
Steepest Ascent Algorithm



Steepest Ascent Algorithm



Steepest Ascent Algorithm



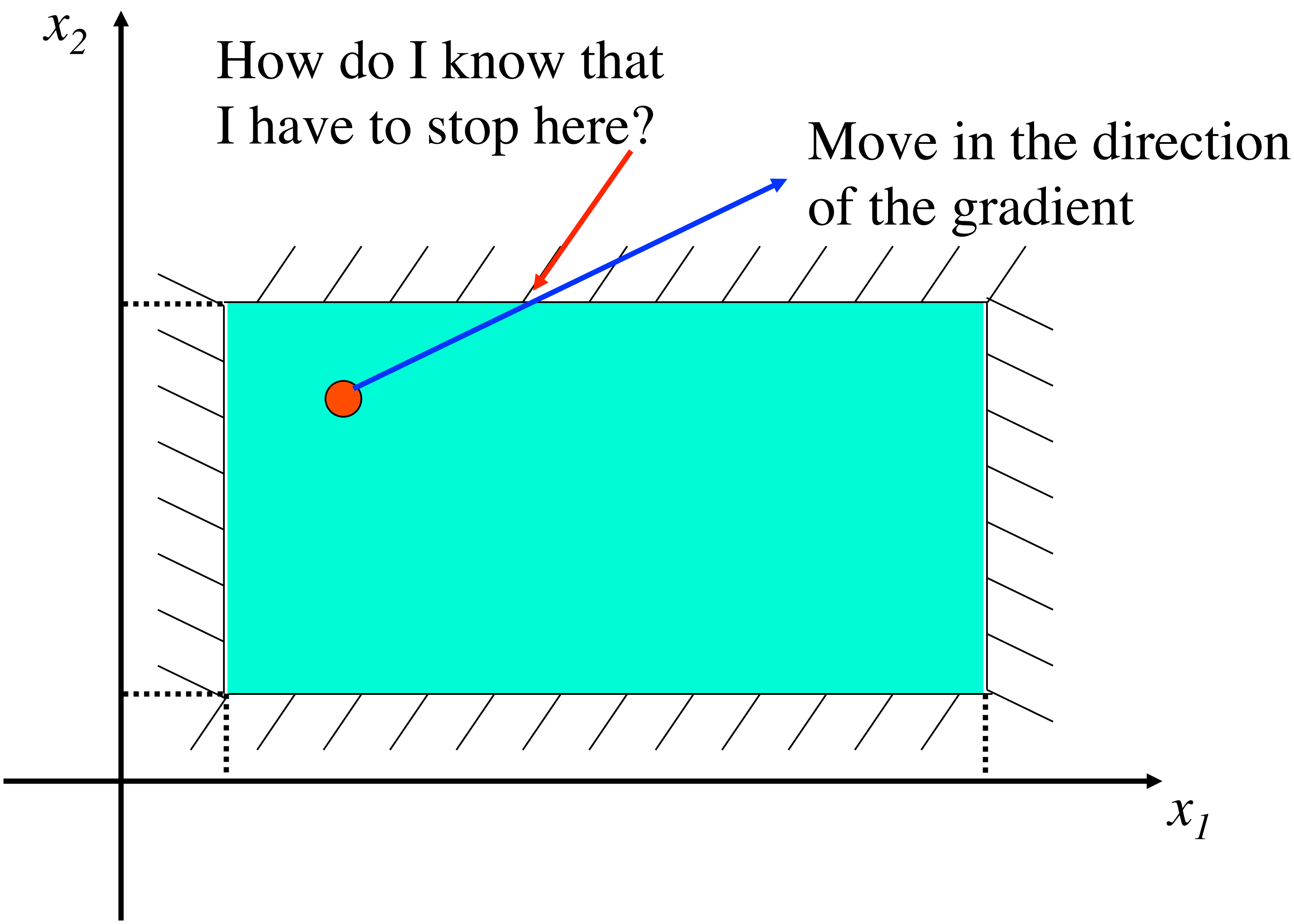
Choosing a Direction

- Direction of steepest ascent/descent is not always the best choice
- Other techniques have been used with varying degrees of success
- In particular, the direction chosen must be consistent with the equality constraints

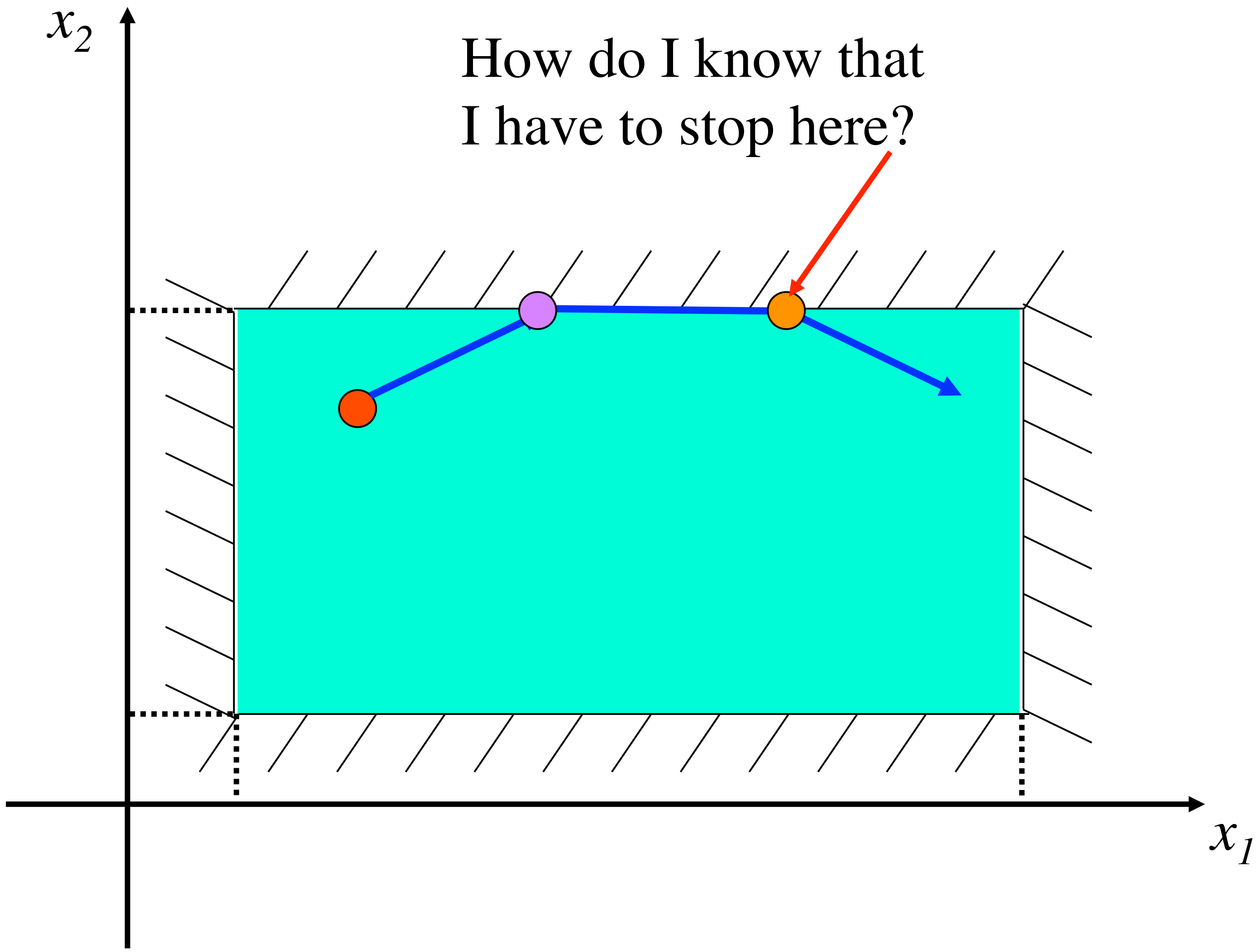
How far to go in that direction?

- Unidirectional searches can be time-consuming
- Second order techniques that use information about the second derivative of the objective function can be used to speed up the process
- Problem with the inequality constraints
  - There may be a lot of inequality constraints
  - Checking all of them every time we move in one direction can take an enormous amount of computing time

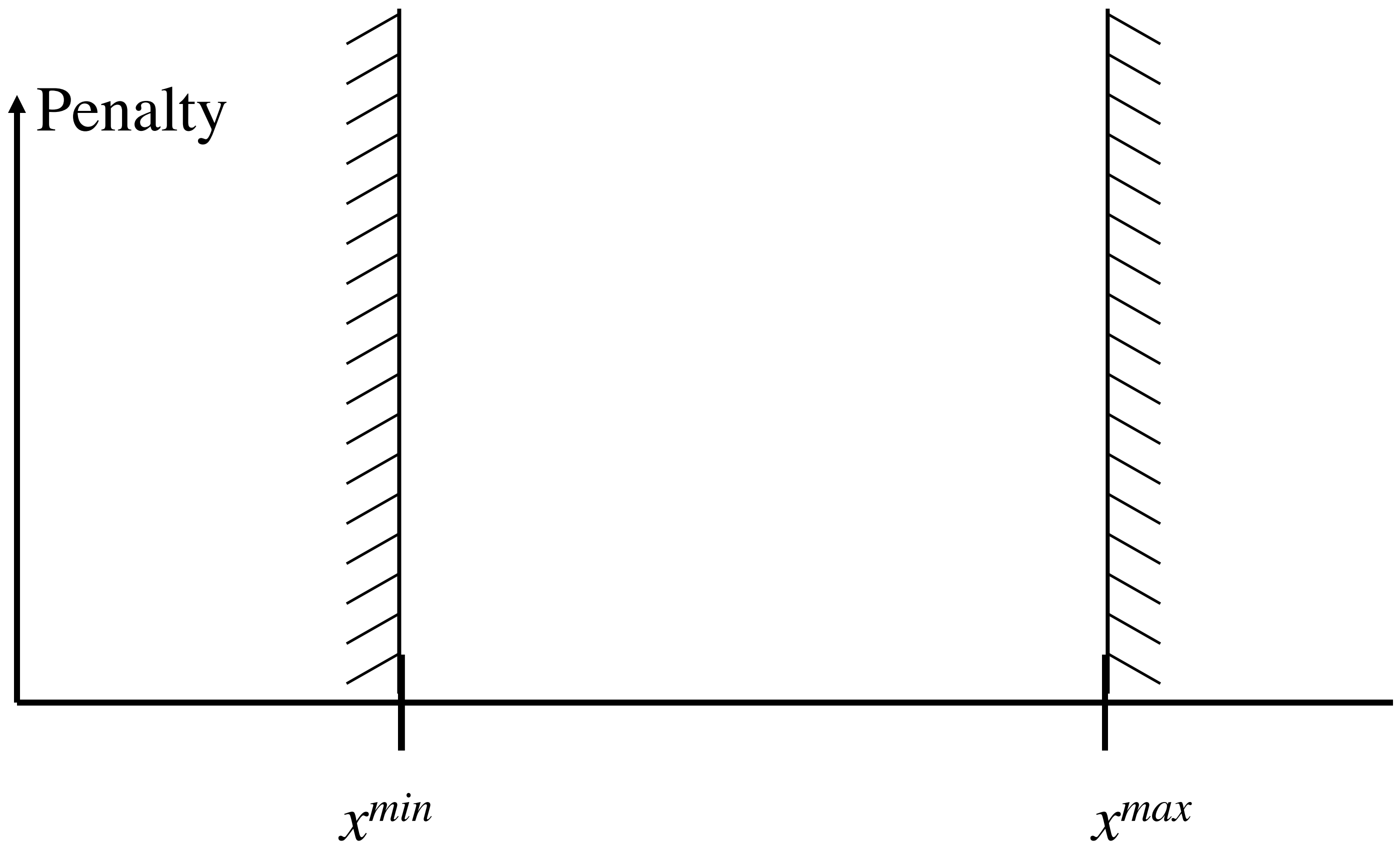
Handling of inequality constraints



Handling of inequality constraints

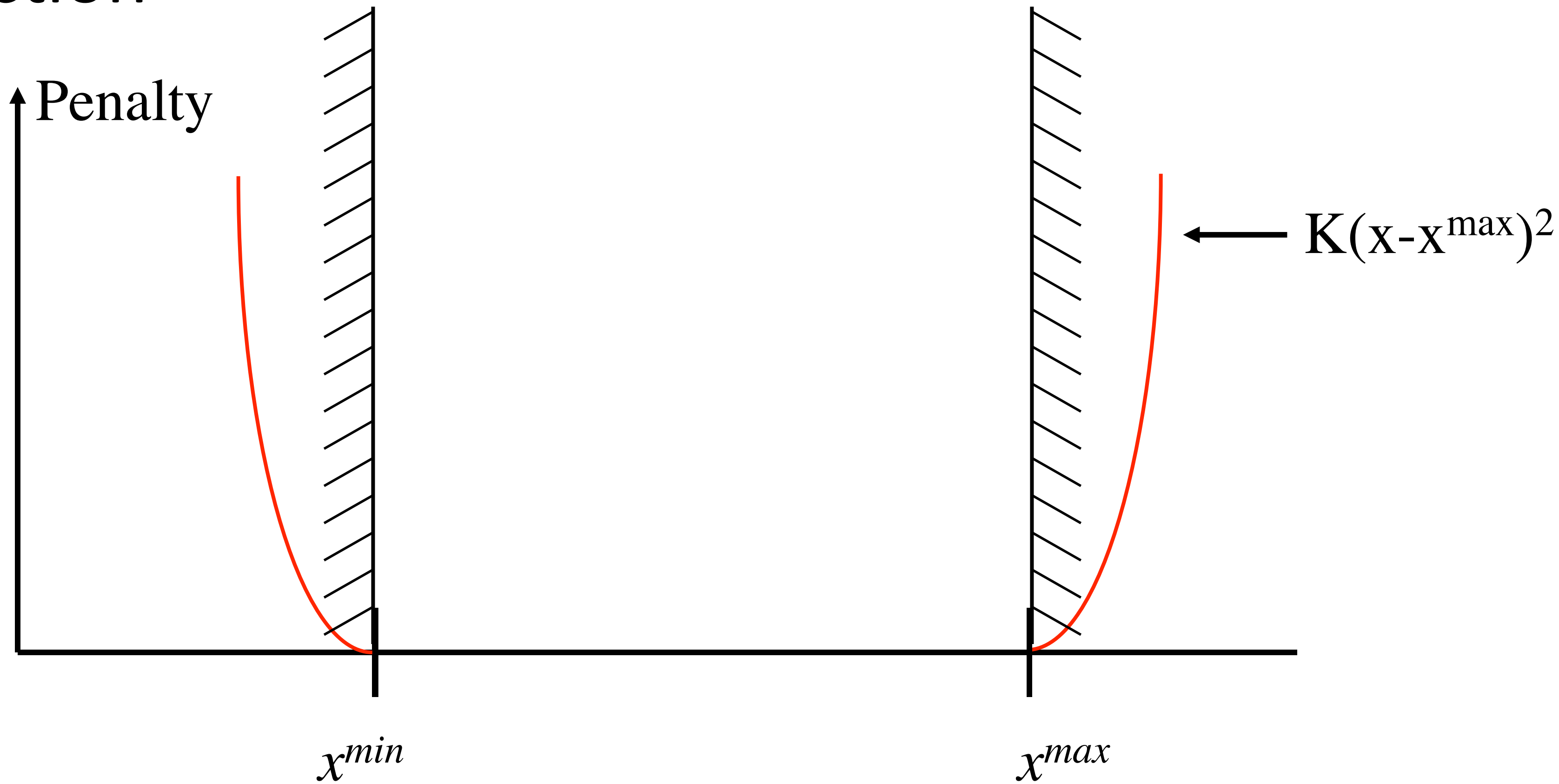


Penalty functions



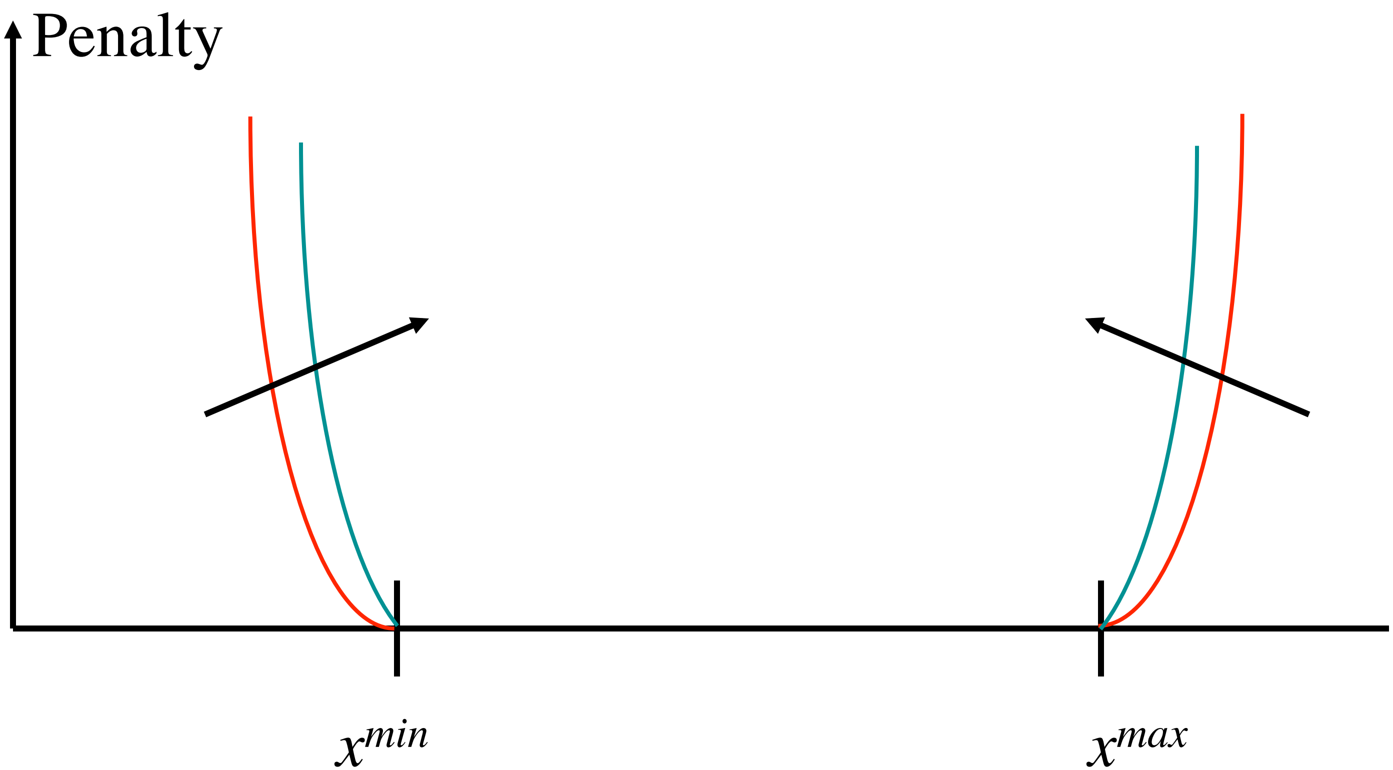
Penalty functions

- Replace enforcement of inequality constraints by addition of penalty terms to objective function

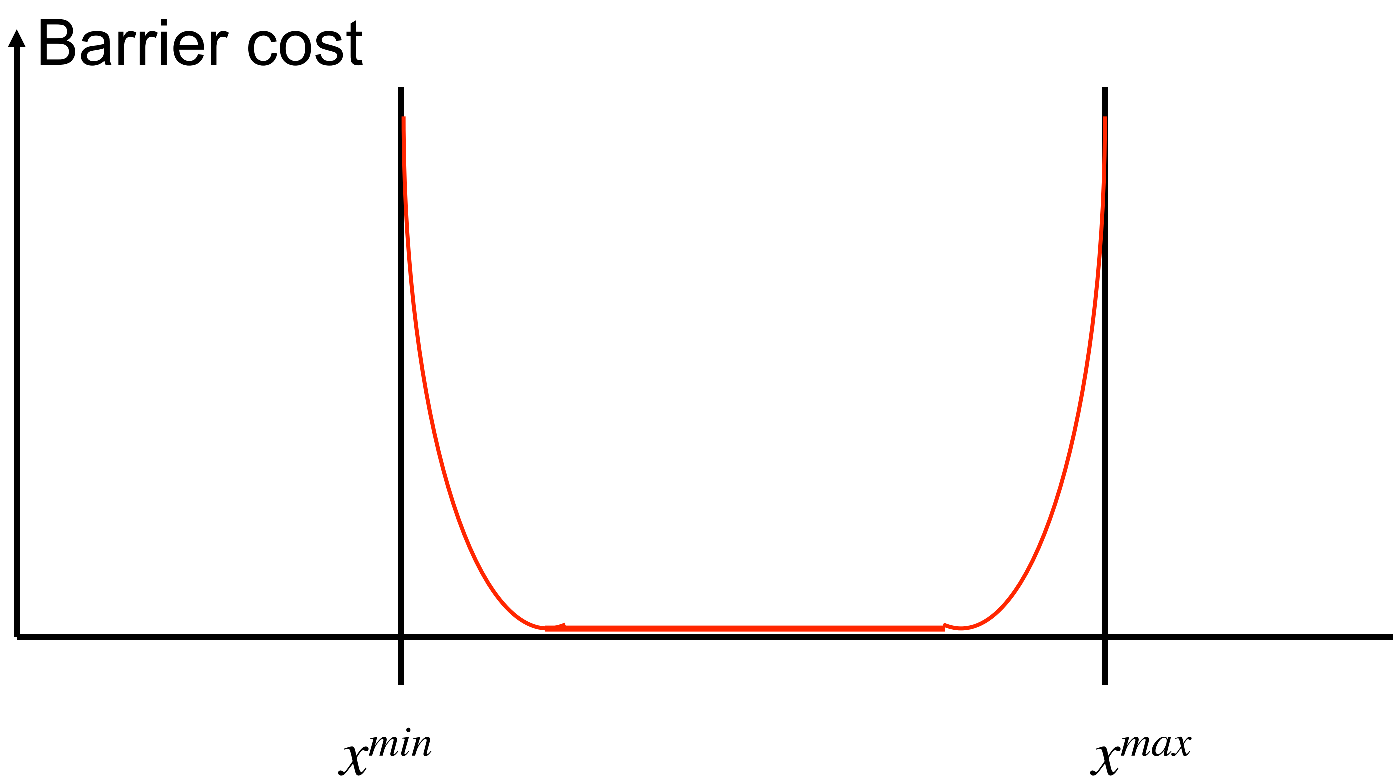


Problem with penalty functions

- Stiffness of the penalty function must be increased progressively to enforce the constraints tightly enough
- Not very efficient method

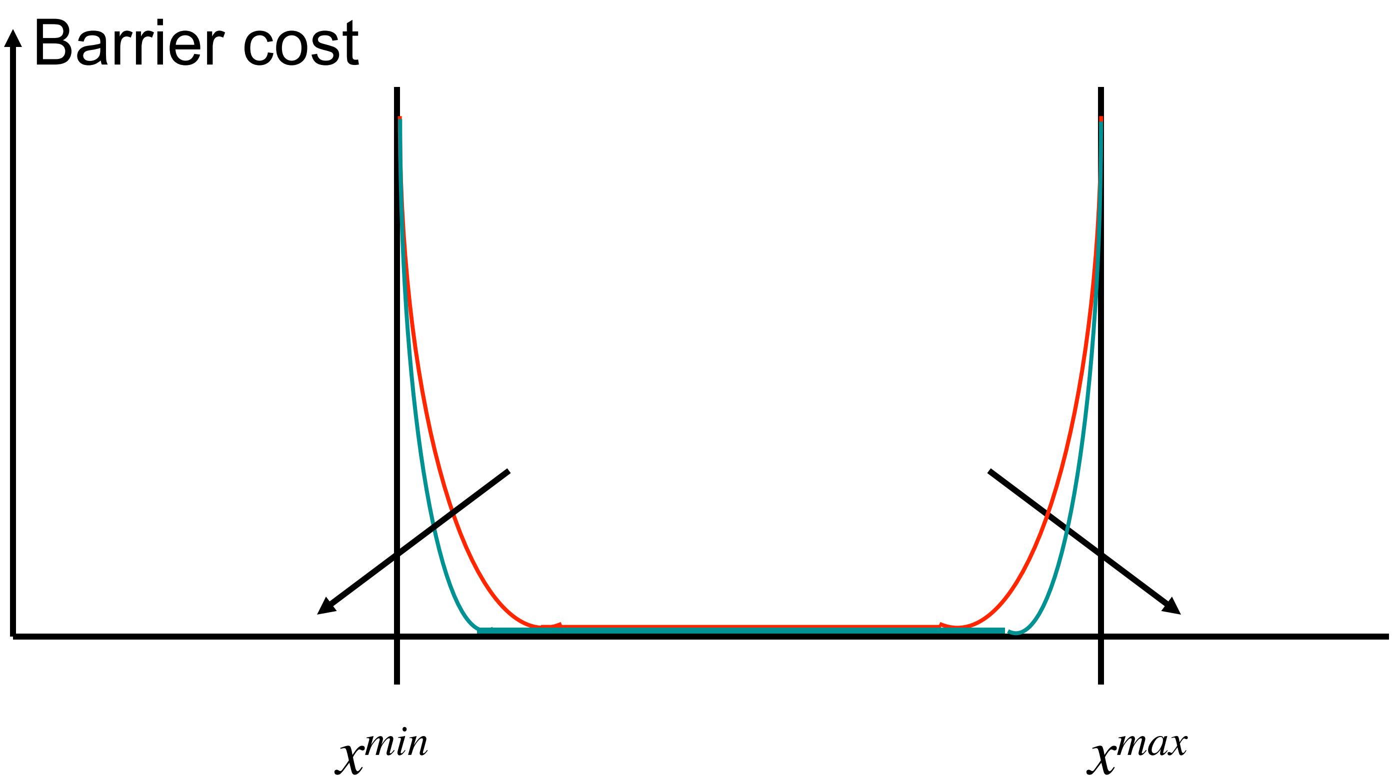


Barrier functions



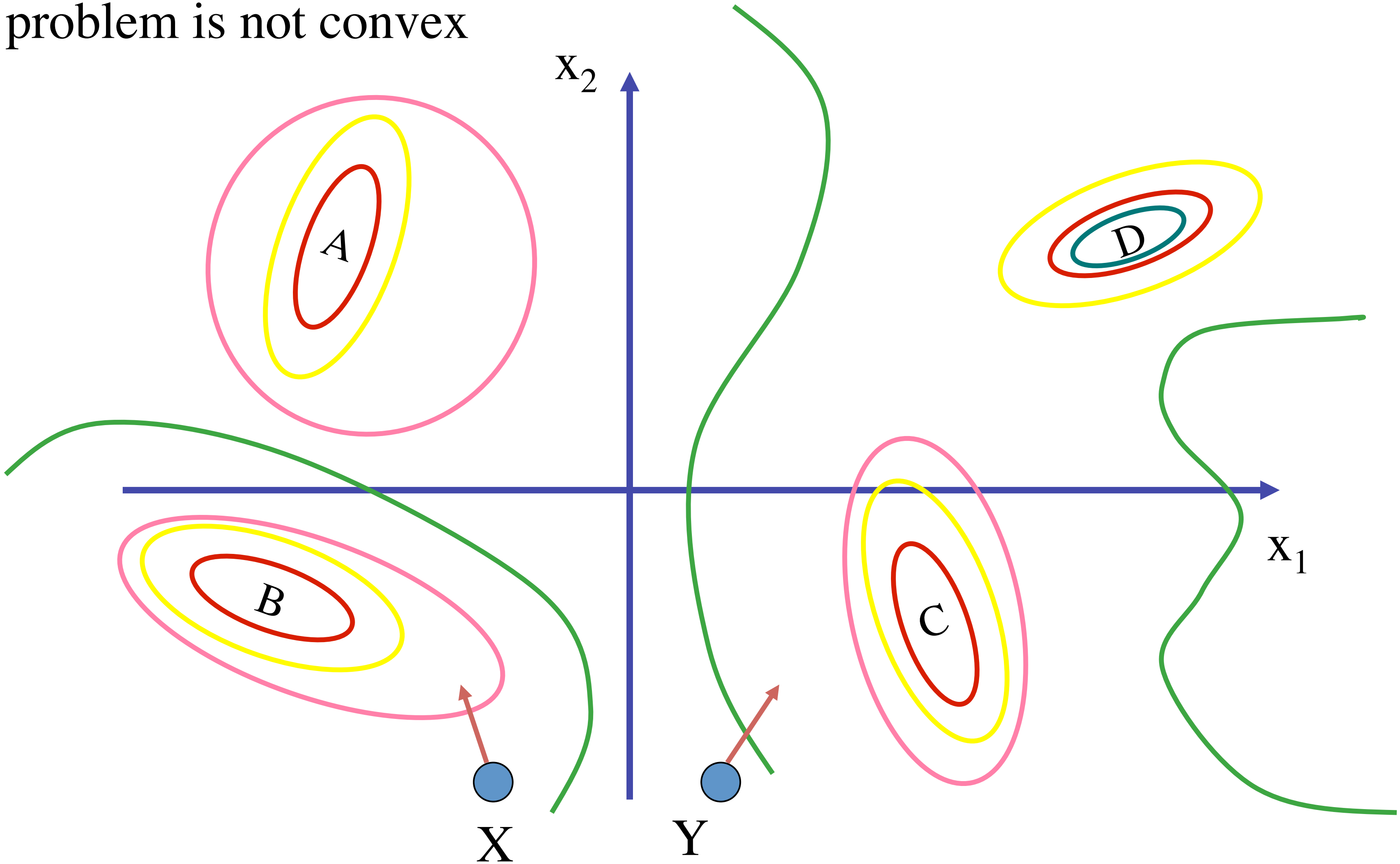
Barrier functions

- Barrier must be made progressively closer to the limit
- Works better than penalty function
- Interior point methods



Non-Robustness

Different starting points may lead to different solutions if the problem is not convex



Conclusions

- Very sophisticated non-linear programming methods have been developed
- They can be difficult to use:
  - Different starting points may lead to different solutions
  - Some problems will require a lot of iterations
  - They may require a lot of "tuning"