Optimization and algorithms

V. Leclère (ENPC)

April 26th, 2024

Why should I bother to learn this stuff?

- Being able to recognize the type of problem is the first step toward finding the right tool to adress it.
- Having an idea of the tools available to you will help choose one.
- $\Longrightarrow$ usefull for any engineer (or intern) that might have to model and then solve a practical optimization problem.

Contents

Contents

Why bother with classes of optimization problems ?

Consider a function $f : X \to \mathbb{R}$, and the following optimization problem

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{Min}} & f(x) \\ \text{s.t.} & x \in X \end{array}$$

Solving this problem can be more or less hard depending on the class in which f and $X \subset \mathbb{R}^n$[1] belongs.
Determining in which class a problem belongs is quite important:

- some problem can be solved for n of order 10 at most, other for n of order 10^6 or more;
- the methodological approach to tackle different problems vary wildly;
- the numerical tools (e.g. solvers) also...

You must be able to (roughly) classify correctly the problem you face, in order to know what can be done or not.

[1] There is also an important theory of optimization where X is not contained in a finite-dimensional space, which will not be discussed here.

Classification with respect to the objective function f ♡

- f linear is the simplest case
- f quadratic is a very important case, simple if f is convex
- f smooth (e.g. $\mathcal{C}^2$) allow to use first and second order information on f
- (f polynomial is a special case, with specific algorithms)
- f convex imply that any local minimum is a global minimum

Finding the optimal solution is a reasonable goal only in the convex case. Otherwise, the algorithm aims at finding one or multiple local optima.

The algorithms we present are mainly for smooth functions. Convergence theory will be done in the convex case.

Classification with respect to the objective function f ♡

- f linear is the simplest case
- f quadratic is a very important case, simple if f is convex
- f smooth (e.g. $\mathcal{C}^2$) allow to use first and second order information on f
- (f polynomial is a special case, with specific algorithms)
- f convex imply that any local minimum is a global minimum

Finding the optimal solution is a reasonable goal only in the convex case. Otherwise, the algorithm aims at finding one or multiple local optima.

The algorithms we present are mainly for smooth functions. Convergence theory will be done in the convex case.

Classification with respect to the objective function f ♡

- f linear is the simplest case
- f quadratic is a very important case, simple if f is convex
- f smooth (e.g. $\mathcal{C}^2$) allow to use first and second order information on f
- (f polynomial is a special case, with specific algorithms)
- f convex imply that any local minimum is a global minimum

Finding the optimal solution is a reasonable goal only in the convex case. Otherwise, the algorithm aims at finding one or multiple local optima.

The algorithms we present are mainly for smooth functions. Convergence theory will be done in the convex case.

Classification with respect to the constraint set

- $X = \mathbb{R}^n$, is known as unconstrained optimization.
- $X = \left\{x \in \mathbb{R}^n \quad | \quad Ax = b\right\}$, can be cast, up to reparametrization, as unconstrained optimization. It might be more efficient to directly deal with the constraints.
- $X = \left\{x \in \mathbb{R}^n \quad | \quad \underline{x}_i \leq x_i \leq \bar{x}_i, \quad \forall i \in [n]\right\}$ is the box constrained optimization.
- $X = \left\{x \in \mathbb{R}^n \quad | \quad Ax \leq b\right\}$ is a polyhedron.
- X convex, generally given as $\left\{x \in \mathbb{R}^n \quad | \quad Ax = b, \quad h_j(x) \leq 0, \quad \forall j \in [n_I]\right\}$ with h_j convex.
- If X is a finite set we speak of combinatorial optimization.
- X can also be non-convex but smooth (e.g. a manifold)

A few comments:

- Unconstrained optimization is by far easier.
- Box constraints, and sometimes spherical constraints, are easy.
- Polyhedral constraints indicate LP-based methods.
- Integrity constraints make the problem a lot harder and change the nature of the optimization methods.

Classification with respect to the constraint set ♡

- $X = \mathbb{R}^n$, is known as unconstrained optimization.
- $X = \left\{x \in \mathbb{R}^n \quad | \quad Ax = b\right\}$, can be cast, up to reparametrization, as unconstrained optimization. It might be more efficient to directly deal with the constraints.
- $X = \left\{x \in \mathbb{R}^n \quad | \quad \underline{x}_i \leq x_i \leq \bar{x}_i, \quad \forall i \in [n]\right\}$ is the box constrained optimization.
- $X = \left\{x \in \mathbb{R}^n \quad | \quad Ax \leq b\right\}$ is a polyhedron.
- X convex, generally given as $\left\{x \in \mathbb{R}^n \quad | \quad Ax = b, \quad h_j(x) \leq 0, \quad \forall j \in [n_I]\right\}$ with h_j convex.
- If X is a finite set we speak of combinatorial optimization.
- X can also be non-convex but smooth (e.g. a manifold)

A few comments:

- Unconstrained optimization is by far easier.
- Box constraints, and sometimes spherical constraints, are easy.
- Polyhedral constraints indicate LP-based methods.
- Integrity constraints make the problem a lot harder and change the nature of the optimization methods.

Contents

Least-square problem (LS)

$$\underset{x \in \mathbb{R}^n}{\text{Min}} \quad \|Ax - b\|_2$$

- equivalent (in which sense ?) to $\underset{x \in \mathbb{R}^n}{\text{Min}} \ \|Ax - b\|_2^2$
- $\rightsquigarrow$ convex, smooth, unconstrained problem
- explicit solution known through algebraic manipulation
- sometimes easier to solve by optimization method than algebraic manipulation
- can be (approximately) solved for $n \geq 10^{11}$ (sparse case)

Exercise : functional approximation

♣ Exercise: We consider a physical function Φ that is approximated as the superposition of multiple simple phenomena (e.g. waves). Each simple phenomenon $p \in [P]$ is represented by a function $\Phi_p : \mathbb{R}^d \to \mathbb{R}$.

We have data points $(x^k, y^k)_{k\in[n]}$, and want to find the Φ that match *at best* the data while being a linear combination of Φ_p.

Propose a least-square regression that answers this question.

Linear optimization problem (LP)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & A'x \leq b' \end{aligned}$$

- convex problem with linear objective and polyhedral constraint set
- a rare case where exact solution can be obtained
- easily solved through dedicated code, open-source (e.g. GLPK) or proprietary[2] (e.g. CPLEX, Gurobi)
- can be solved for $n \geq 10^8$
- very important case in practice and as a subroutine for other problems
- two main (class of) algorithms:
  - simplex algorithm (seen in 1A)
  - interior point method (discussed later in this course)

[2] Licences are expensive(!!) but free for students!

Quadratic optimization problem (QP)

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{Min}} \quad & \frac{1}{2} x^\top Q x + c^\top x \\ \text{s.t.} \quad & Ax = b \\ & A'x \leq b' \end{aligned}$$

- quadratic objective and polyhedral constraint set
- exact solution can be obtained
- easily solved if $Q \succeq 0$, hard otherwise
- can be solved for $n \geq 10^7$ (convex case)

Exercise : Lasso as QP

♣ Exercise: A classical extension of the least-square problem, which has strong theoretical and practical interest is the LASSO problem

$$\underset{x \in \mathbb{R}^p}{\text{Min}} \qquad \|Ax - y\|^2 + \lambda \|x\|_1$$

Show that this problem can be cast as a QP problem.

Quadratically constrained quadratic problem (QCQP)

$$
\begin{aligned}
\underset{x \in \mathbb{R}^n}{\text{Min}} \quad & \frac{1}{2} x^\top Q x + c^\top x \\
\text{s.t.} \quad & \frac{1}{2} x^\top P_i x + q_i^\top x \leq b_i \qquad \forall i \in [k] \\
& Ax = b \\
& A'x \leq b'
\end{aligned}
$$

- Reasonably easy if convex (i.e if Q and P_i are semi-definite positive)
- can be solved for $n \geq 10^7$
- less important than previous examples

Exercise: binary optimization is equivalent to QCQP

♣ Exercise: Consider the following optimization problem.

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{Min}} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & x_i \in \{0, 1\} \qquad \forall i \in I \end{aligned}$$

Write this problem as a QCQP. Is it convex ?

Second order cone problem (SOCP)

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{Min}} \quad & c^\top x \\ \text{s.t.} \quad & \|A_i x + b_i\|_2 \leq c_i^\top x + d_i \qquad \forall i \in [k] \\ & Ax = b \\ & A'x \leq b' \end{aligned}$$

- convex problem
- can be solved for $n \geq 10^7$, through most "linear" solver, relying on interior points methods
- equivalent to convex QCQP
- extend the modeling power of LP
- appears naturally in robust optimization

Exercise : robust linear programming

♠ Exercise: Consider the following robust linear program

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{Min}} & c^\top x \\ \text{s.t.} & (a_i + R_i \delta_i)^\top x \leq b_i \qquad \forall \|\delta_i\|_2 \leq 1, \quad \forall i \in [m] \end{array}$$

Write this problem as a SOCP.

Semi definite programming (SDP)

$$\begin{aligned} \underset{X \in S_n(\mathbb{R})}{\text{Min}} \quad & \operatorname{tr}\left(CX\right) \\ \text{s.t.} \quad & A(X) = b \\ & X \succeq 0 \end{aligned}$$

where X, and C are symmetric matrices, and $A : S_n \to \mathbb{R}^m$ a linear mapping.

- convex problem
- can be solved for $n \geq 10^3$, through some "linear" solver, relying on interior points methods
- contains SOCP
- limited in size in part because the number of actual variables is n^2

Unconstrained convex-differentiable optimization ♡

$$\min_{x} \quad f(x)$$

where f is convex, finite, and differentiable.

- Iterative algorithm yields ε-solution
- Solutions are global due to convexity
- Complexity theory is well understood: maximum theoretical speed, and algorithms matching this speed
- Convergence speed is better under strong convexity assumptions
- Can be solved for $n \geq 10^5$

$\rightsquigarrow$ this is where we will spend most of our time

Unconstrained convex non-differentiable optimization

$$\min_{x} \quad f(x)$$

where f is convex and finite.

- Iterative algorithm yields ε-solution
- Solutions are global due to convexity
- Complexity theory is well understood: maximum theoretical speed, and algorithms matching this speed
- Can be solved for $n \geq 10^4$

Unconstrained differentiable optimization

$$\min_{x} \quad f(x)$$

where f is differentiable.

- Iterative algorithm yields ε local optimum
- Algorithms are mostly the same as in convex differentiable setting, but the theory is more involved
- Can find a local optimum for $n \geq 10^5$

$\rightsquigarrow$ most algorithms presented in this course can be used to hopefully get to a locally optimal point.

Constrained convex optimization

$$\begin{array}{ll} \underset{x}{\text{Min}} & f(x) \\ \text{s.t.} & x \in X \end{array}$$

where X is a convex set.

- Easiest if X is a box or ball
- Specific approach relying on LP if X is a polyhedron
- Various methods in the generic case:
  - projection
  - feasible direction
  - constraint penalization
  - dualization

Combinatorial problem

$$\min_{x \in X} \quad f(x)$$

where X is a finite set.

- This roughly represent the problem of combinatorial optimization.
- X being finite you can, in theory, test all possibilities and choose the best. However, this brute force approach is often unpractical due to the size of X.
- Even if an exact solution can be obtained, it is not often the case.
- Finding lower-bound is interesting to understand how far your current solution is from the optimum.
- Practical methods are often *matheuristics* or *meta-heuristics* adapted to the specificity of the problem.
- Problems are often very hard, and practical solvability depends on the specific problem structure.

Mixed Integer Linear Programming

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b, x \geq 0 \\ & x_i \in \mathbb{N} \qquad \forall i \in I \subset [n] \end{aligned}$$

- A very important class of problem, with huge modeling power.
- By order of difficulty we distinguish: continuous variables, binary variables and integer variables.
- Exact solution methods rely on the idea of branch and cut. (`https://www.youtube.com/watch?v=2zKCQ03JzOY` (13'))
- Very powerful (commercial) solvers (like Gurobi, Cplex, Mosek...) are developed and improved every year to tackle these problems. They use a mix of insightful mathematical ideas and heuristic knowledge.
- Efficiency of the solver depends on the type of problem, and the formulation of the problem.

Mixed Integer Conic Programming

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & x \in C \\ & x_i \in \mathbb{N} \qquad \forall i \in I \subset [n] \end{aligned}$$

where C is a convex cone.

- Harder than MILP
- More recent development, thus the theory and heuristic experience is less advanced than MILP
- Numerical efficiency is quickly improving

Exercise : stock optimization

♠ Exercise: Consider that you sell a given product over T days. The demand for each day is d_t. Having a quantity x_t of items in stock have a cost (per day) of cx_t. You can order, each day, a quantity q_t, and have to satisfy the demand.

For each of the following variations: model the problem, give the class to which it belongs, and give the optimal solution if easily found.

1. Without any further constraints / specifications.
2. There is an "ordering cost": each time you order, you have to pay a fixed cost κ.
3. Instead of an "ordering cost" there is a maximum number of days at which you can order a replenishment.

Contents

Contents

"Ad Hoc" solution

A heuristic is an admissible, not necessarily optimal, solution to a given optimization problem. It gives upper-bounds.

- In a lot of applications, experience or good sense, can give reasonably good heuristics.
- Sometimes these heuristics can have a few parameters that can be tuned by trial and error.

♣ Exercise: In the stock optimization example, with fixed ordering cost, propose a simple heuristic.

♣ Exercise: Now assume that, in this same example, there is some uncertainty on the demand, adapt your heuristic to offer a *robustness* parameter.

"Ad Hoc" solution

A heuristic is an admissible, not necessarily optimal, solution to a given optimization problem. It gives upper-bounds.

- In a lot of applications, experience or good sense, can give reasonably good heuristics.
- Sometimes these heuristics can have a few parameters that can be tuned by trial and error.

♣ Exercise: In the stock optimization example, with fixed ordering cost, propose a simple heuristic.

♣ Exercise: Now assume that, in this same example, there is some uncertainty on the demand, adapt your heuristic to offer a *robustness* parameter.

"Ad Hoc" solution

A heuristic is an admissible, not necessarily optimal, solution to a given optimization problem. It gives upper-bounds.

- In a lot of applications, experience or good sense, can give reasonably good heuristics.
- Sometimes these heuristics can have a few parameters that can be tuned by trial and error.

♣ Exercise: In the stock optimization example, with fixed ordering cost, propose a simple heuristic.

♣ Exercise: Now assume that, in this same example, there is some uncertainty on the demand, adapt your heuristic to offer a *robustness* parameter.

Random search

A good way of obtaining *good* solutions is to randomly test multiple admissible solutions, and keep the best one.
Examples:

- exhaustive search (combinatorial)
- genetic algorithms
- simulated annealing
- swarm particles

Use case :

- hard problems (combinatorial or continuous) where finding an admissible solution is easy
- when you just want an admissible solution, if possible better than what you had

Random search

A good way of obtaining *good* solutions is to randomly test multiple admissible solutions, and keep the best one.
Examples:

- exhaustive search (combinatorial)
- genetic algorithms
- simulated annealing
- swarm particles

Use case :

- hard problems (combinatorial or continuous) where finding an admissible solution is easy
- when you just want an admissible solution, if possible better than what you had

Video ressources

https://www.youtube.com/watch?v=3QJjfeVrut8 (5')
https://www.youtube.com/watch?v=NI3WllrvWoc (4')

Contents

Descent methods

Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \quad f(x). \tag{1}$$

A *descent direction algorithm* is an algorithm that constructs a sequence of points $(x^{(k)})_{k \in \mathbb{N}}$, that are recursively defined with:

$$x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)} \tag{2}$$

where

- $x^{(0)}$ is the initial point,
- $d^{(k)} \in \mathbb{R}^n$ is the descent direction,
- $t^{(k)}$ is the step length.

$\rightsquigarrow$ most of this is discussed in next classes.

Descent methods

Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \quad f(x). \tag{1}$$

A *descent direction algorithm* is an algorithm that constructs a sequence of points $(x^{(k)})_{k \in \mathbb{N}}$, that are recursively defined with:

$$x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)} \tag{2}$$

where

- $x^{(0)}$ is the initial point,
- $d^{(k)} \in \mathbb{R}^n$ is the descent direction,
- $t^{(k)}$ is the step length.

$\rightsquigarrow$ most of this is discussed in next classes.

Descent methods

Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \quad f(x). \tag{1}$$

A *descent direction algorithm* is an algorithm that constructs a sequence of points $(x^{(k)})_{k \in \mathbb{N}}$, that are recursively defined with:

$$x^{(k+1)} = x^{(k)} + t^{(k)} d^{(k)} \tag{2}$$

where

- $x^{(0)}$ is the initial point,
- $d^{(k)} \in \mathbb{R}^n$ is the descent direction,
- $t^{(k)}$ is the step length.

$\rightsquigarrow$ most of this is discussed in next classes.

Video explanation

https://www.youtube.com/watch?v=n-Y0SDSOfUI(5')

Descent direction

For a differentiable objective function f, $d^{(k)}$ will be a descent direction iff $\nabla f(x^{(k)}) \cdot d^{(k)} \leq 0$, which can be seen from a first order development:

$$f(x^{(k)} + t^{(k)} d^{(k)}) = f(x^{(k)}) + t\left\langle \nabla f(x^{(k)}), d^{(k)} \right\rangle + o(t).$$

The most classical descent direction is $d^{(k)} = -\nabla f(x^{(k)})$, which correspond to the gradient algorithm.

Descent direction

For a differentiable objective function f, $d^{(k)}$ will be a descent direction iff $\nabla f(x^{(k)}) \cdot d^{(k)} \leq 0$, which can be seen from a first order development:

$$f(x^{(k)} + t^{(k)} d^{(k)}) = f(x^{(k)}) + t\left\langle \nabla f(x^{(k)}), d^{(k)} \right\rangle + o(t).$$

The most classical descent direction is $d^{(k)} = -\nabla f(x^{(k)})$, which correspond to the gradient algorithm.

Step-size choice

The step-size $t^{(k)}$ can be:

- fixed $t^{(k)} = t^{(0)}$, for all iteration,
- optimal $t^{(k)} \in \arg\min_{t \geq 0} f(x^{(k)} + t d^{(k)})$,
- a "good" step, following some rules (e.g Armijo's rules).

Finding the optimal step size is a special case of unidimensional optimization (or linear search).

Step-size choice

The step-size $t^{(k)}$ can be:

- fixed $t^{(k)} = t^{(0)}$, for all iteration,
- optimal $t^{(k)} \in \arg\min_{t \geq 0} f(x^{(k)} + t d^{(k)})$,
- a "good" step, following some rules (e.g Armijo's rules).

Finding the optimal step size is a special case of unidimensional optimization (or linear search).

Contents

Model based methods

Another class of algorithm consists in constructing a simple model of the objective function f that is optimized and then refined.
Generally speaking, model-based algorithm goes as follows:

1. Solve $\min_{x \in X} f^k(x)$
2. Update model f^k into f^{k+1}

This approach might work if

- The model problem $\min_{x \in X} f^k(x)$ is *simple*
- The model f^k locally looks like the true function f around the optimum

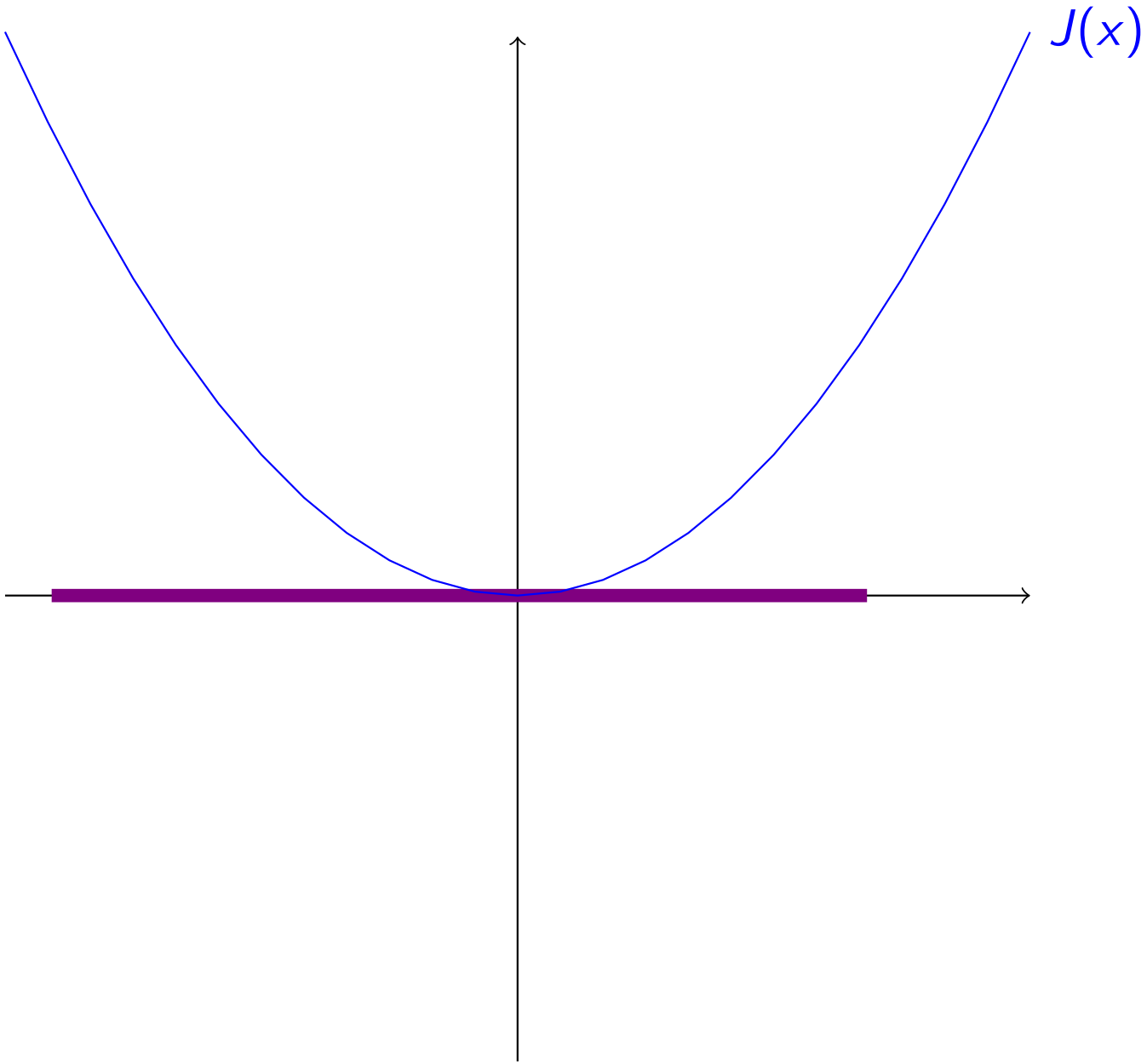
Model based methods

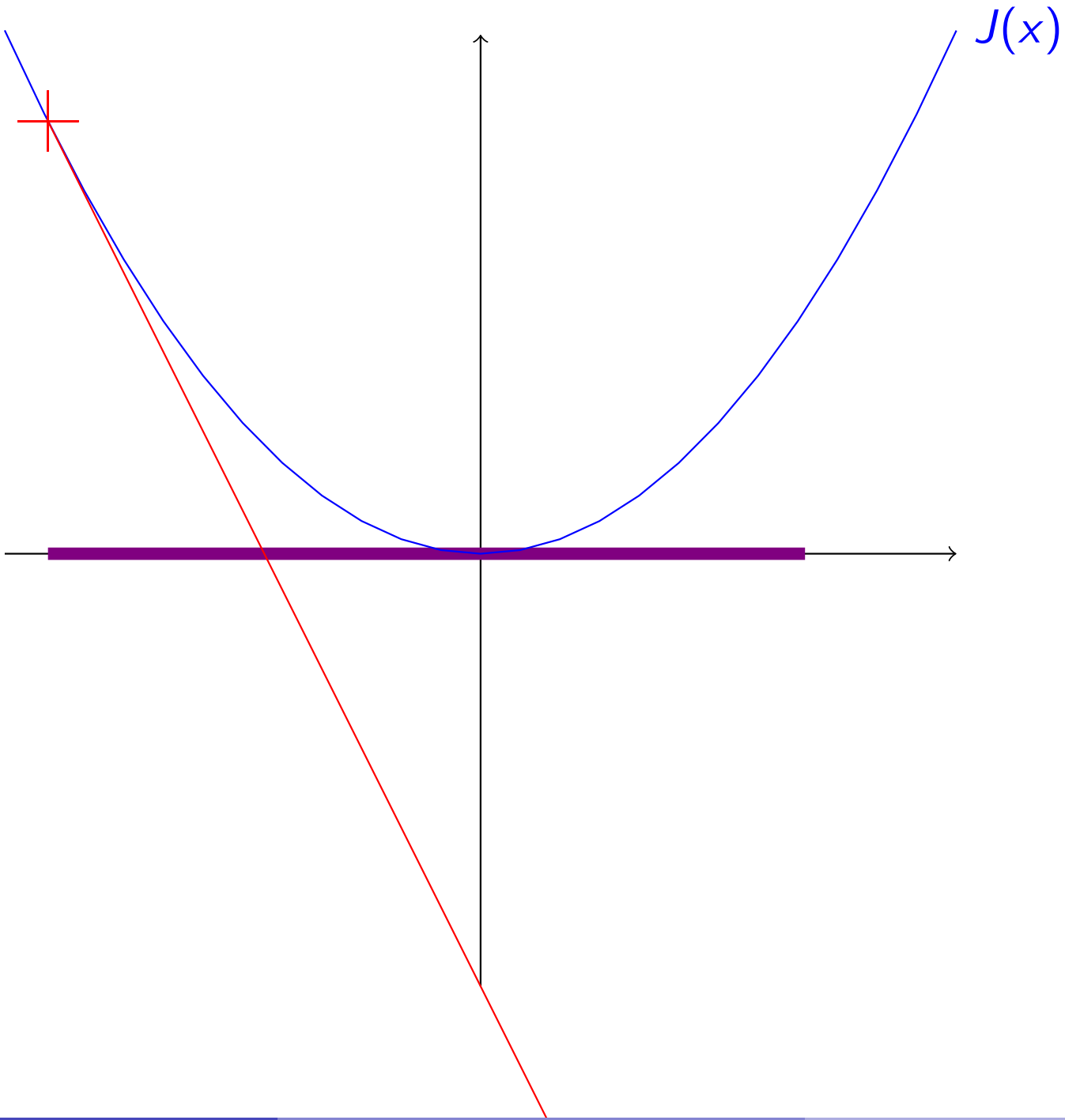
Another class of algorithm consists in constructing a simple model of the objective function f that is optimized and then refined.
Generally speaking, model-based algorithm goes as follows:

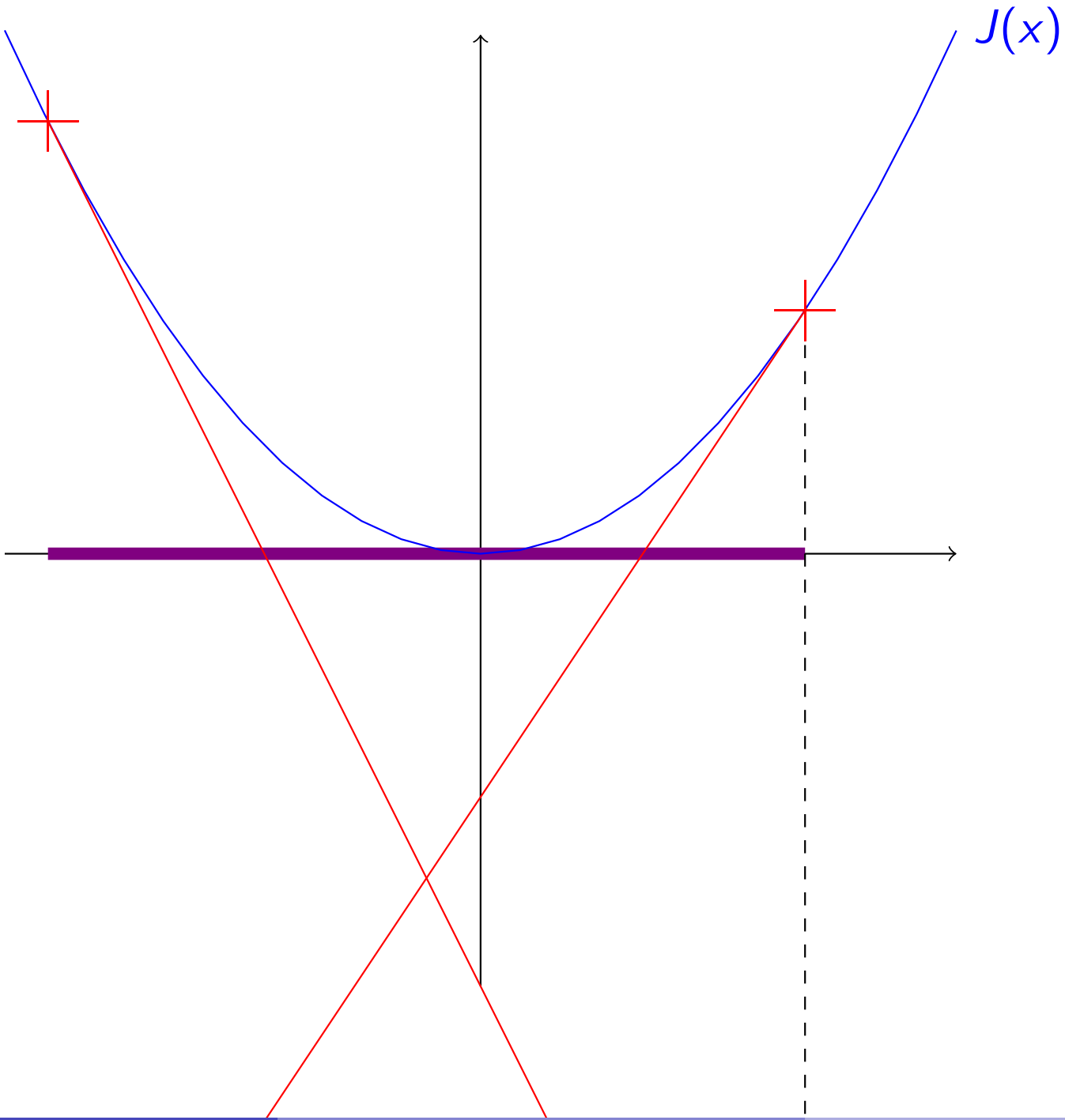
1. Solve $\min_{x \in X} f^k(x)$
2. Update model f^k into f^{k+1}

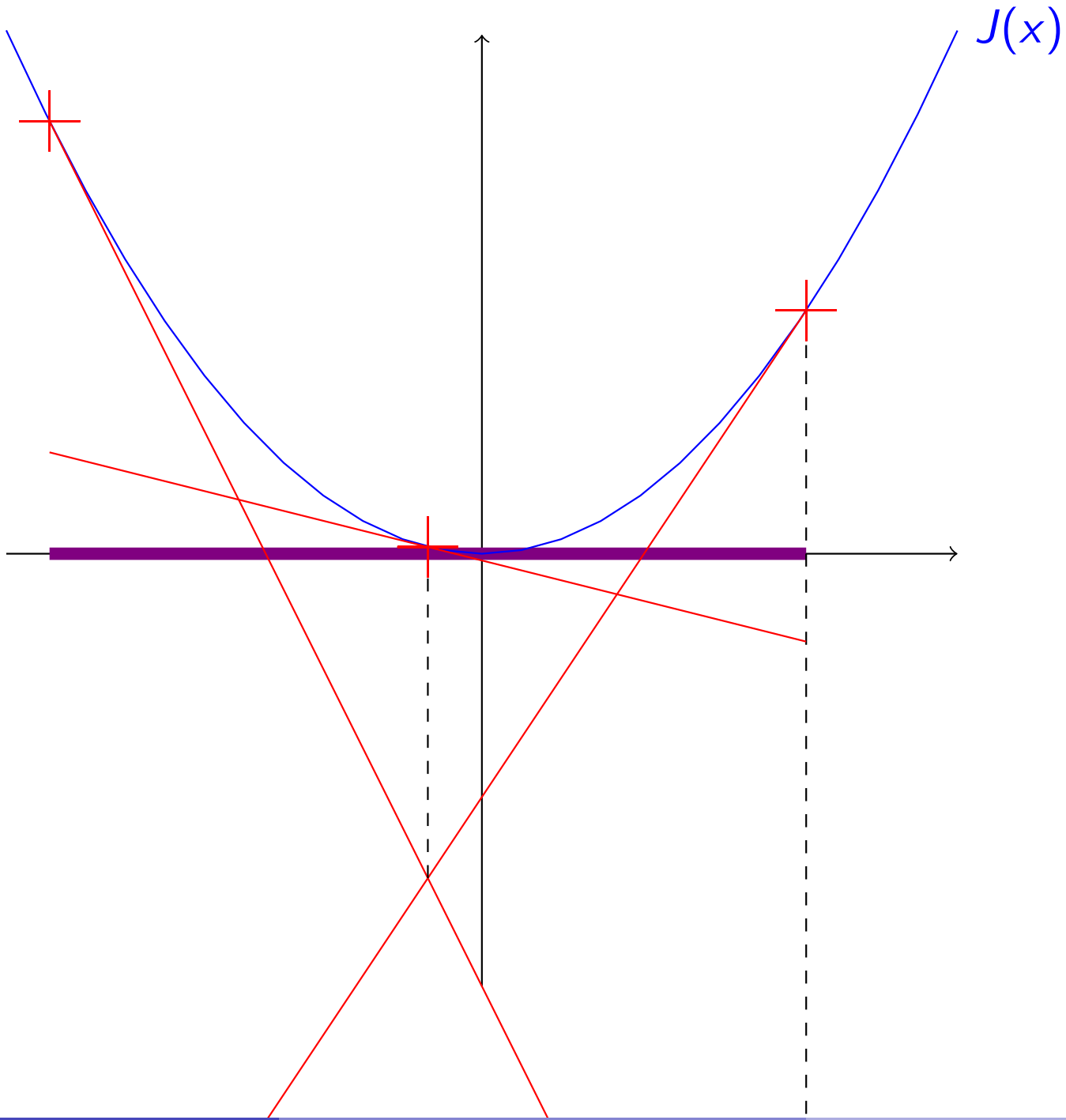
This approach might work if

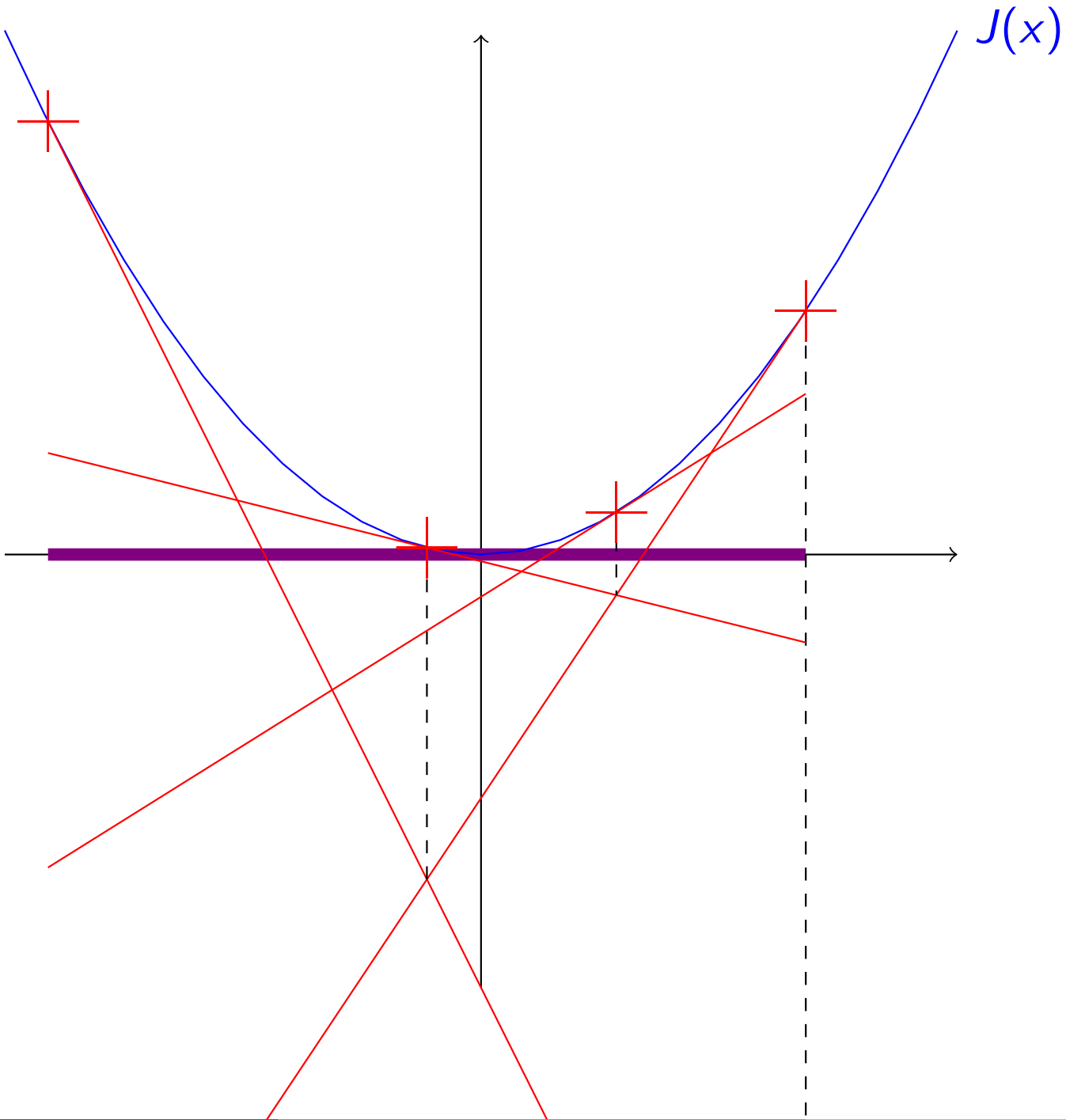
- The model problem $\min_{x \in X} f^k(x)$ is *simple*
- The model f^k locally looks like the true function f around the optimum


$J(x)$


$J(x)$


$J(x)$


$J(x)$


$J(x)$

Kelley algorithm

Data: Convex objective function f, Compact set X, Initial point $x_0 \in X$

Result: Admissible solution $x^{(k)}$, lower-bound $\underline{v}^{(k)}$

Set $f^{(0)} \equiv -\infty$;

for $k \in \mathbb{N}$ **do**

- Compute a subgradient $g^{(k)} \in \partial f(x^{(k)})$;
- Define a cut $\mathcal{C}^{(k)} : x \mapsto f(x^{(k)}) + \langle g^{(k)}, x - x^{(k)} \rangle$;
- Update the lower approximation $f^{(k+1)} = \max\{f^{(k)}, \mathcal{C}^{(k)}\}$;
- Solve $(P^{(k)}) : \quad \min_{x \in X} f^{(k+1)}(x)$;
- Set $\underline{v}^{(k)} = val(P^{(k)})$;
- Select $x^{(k+1)} \in sol(P^{(k)})$;

end

Algorithm 1: Kelley's cutting plane algorithm

Trust region method

Consider an unconstrained, non-linear, smooth problem

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

The idea of trust region is based on the following two facts:

- f locally looks like it's second order limited development

$$f(x+h) = f(x) + \left\langle \nabla f(x)\,, h \right\rangle + \frac{1}{2} h^\top \nabla^2 f(x) h + o(\|h\|^2)$$

- we know how to compute the minimum of a quadratic function on a ball.

Trust region method

The trust region method goes as follows, given a current point x^k and *trust radius* Δ^k

1. compute $f^k(x^k + h) = f(x^k) + \langle \nabla f(x^k), h \rangle + \frac{1}{2} h^\top \nabla^2 f(x^k) h$
2. solve $\min_{y \in B(x^k, \Delta^k)} f^k(y)$, with optimal solution y^k
3. compute $f(y^k)$
4. compute the *concordance* r^k as the ratio actual decrease / model decrease
$$r^k = \frac{f(x^k) - f(y^k)}{f^k(x^k) - f^k(y^k)}$$
   - If r^k is small, the model is bad and you decrease Δ^k and restart the iteration
   - If r^k is large (close to 1) update the current point $x^{k+1} = y^k$.
5. If r^k is close to one and y^k is on the boundary, increase Δ^k.

$\rightsquigarrow$ there are full books on trust region methods.

Trust region method

The trust region method goes as follows, given a current point x^k and *trust radius* Δ^k

1. compute $f^k(x^k + h) = f(x^k) + \langle \nabla f(x^k), h \rangle + \frac{1}{2} h^\top \nabla^2 f(x^k) h$
2. solve $\min_{y \in B(x^k, \Delta^k)} f^k(y)$, with optimal solution y^k
3. compute $f(y^k)$
4. compute the *concordance* r^k as the ratio actual decrease / model decrease
$$r^k = \frac{f(x^k) - f(y^k)}{f^k(x^k) - f^k(y^k)}$$
   - If r^k is small, the model is bad and you decrease Δ^k and restart the iteration
   - If r^k is large (close to 1) update the current point $x^{k+1} = y^k$.
5. If r^k is close to one and y^k is on the boundary, increase Δ^k.

$\rightsquigarrow$ there are full books on trust region methods.

Video explanation

https://www.youtube.com/watch?v=G-QKRv1rgG0(30')

Some optimization problems in Python

https://www.youtube.com/watch?v=sJ5HTi70wXo(10')

Contents

What you have to know

- Important elements defining an optimization problem : continuous/discrete, smooth/non-differentiable, convex/non-convex, linear/non-linear, constrained/unconstrained.
- Main optimization classes: LP, MILP, differentiable unconstrained, combinatorial.
- The difference between heuristic and exact methods
- Main classes of exact method : descent direction, approximation method.

What you really should know

- Other important classes of optimization problem (LS, QP, SOCP, SDP)
- Some ideas of heuristic methods (simulated annealing, genetic algorithms)
- Kelley's cutting plane algorithm
- Principle of trust region method

What you have to be able to do

- Recognise a LP / MILP
- Recognise a (convex) differentiable optimization problem, constrained or not

What you should be able to do

- know how to use a "lift" variable, e.g.

$$\begin{aligned} \min_{x} \quad & \max(f_1(x), f_2(x)) = \quad \min_{x,z} \quad z \\ & \text{s.t.} \quad f_1(x) \leq z \\ & \qquad\quad f_2(x) \leq z \end{aligned}$$