If you forget everything else

V. Leclère (ENPC)

March 26th, 2021

Contents

1. Markov Decision Problem
2. Convexity
3. Optimality conditions
4. Duality
5. Classification of optimization problem
6. Gradient algorithm(s)
7. Newton and Quasi-Newton algorithm(s)
8. Constrained optimization
9. Interior Points Methods
10. Stochastic Gradient

Why should I bother to learn this stuff ?

- Markov Chain and Markov Decision Programm are very powerful modeling tool for a lot of practicle applications.
- Dynamic programming is a flexible tool, easy to implement, that can efficiently address these problems.
- $\Longrightarrow$ useful for any futur "manager"

What you have to know

- What is a Markov Chain, a Markov Controlled Chain.
- What is a Markov Decision Problem, a state, a policy
- What is the Bellman's value function a.k.a cost-to-go
- Estimate the value of a policy through Monte Carlo

What you really should know

- the complexity of Dynamic Programming
- how to model forbidden state in DP

What you have to be able to do

- Recognize an MDP
- Write a Dynamic Programming equation
- Solve a simple, finite horizon, MDP problem through Dynamic Programming

What you should be able to do

- Know if a problem can numerically be tackled through Dynamic Programming
- Reframe a non-Markovian problem as a Markovian problem through extending the state

Contents

1. Markov Decision Problem
2. Convexity
3. Optimality conditions
4. Duality
5. Classification of optimization problem
6. Gradient algorithm(s)
7. Newton and Quasi-Newton algorithm(s)
8. Constrained optimization
9. Interior Points Methods
10. Stochastic Gradient

Why should I bother to learn this stuff ?

- Convex vocabulary and results are needed throughout the course, especially to obtain optimality condition and duality relation.
- Convex analysis tools like Fenchel transform appears in modern machine learning theory
- $\Longrightarrow$ fundamental for M2 in continuous optimization
- $\Longrightarrow$ usefull for M2 in operation research, machine learning (and some part of probability or mechanics)

What you have to know

- What is a affine set, a convex set, a polyhedron, a (convex) cone
- What is a convex function, that it is above its tangeants.
- Jensen inequality
- What is a convex optimization problem. That any local minimum is a global minimum.
- The necessary optimality condition $\nabla f(x^\sharp) \in [T_X(x^\sharp)]^+$

What you really should know

- That you can separate convex sets with a linear function
- What is the positive dual of a cone
- Basic manipulations preserving convexity (sum, cartesian product, intersection, linear projection)
- What is the domain, the sublevel of a function f
- What is a lower semi continuous function, a proper convex function
- Conditions of (strict, strong) convexity for differentiable functions
- The partial minimum of a convex function is convex
- The definition of the subdifferential.
- The definition of the Fenchel transform.
- The link between Fenchel transform and subdifferential.

What you have to be able to do

- Show that a set is convex
- Show that a function is (strictly, strongly) convex
- Go from constrained problem to unconstrained problem using the indicator function $\mathbb{I}_X$

What you should be able to do

- Compute dual cones
- Use advanced results (projection, partial infimum, perspective) to show that a function or a set is convex
- Compute the Fenchel transform of simple function

Contents

1. Markov Decision Problem
2. Convexity
3. Optimality conditions
4. Duality
5. Classification of optimization problem
6. Gradient algorithm(s)
7. Newton and Quasi-Newton algorithm(s)
8. Constrained optimization
9. Interior Points Methods
10. Stochastic Gradient

Why should I bother to learn this stuff ?

- Optimality conditions enable to solve exactly some easy optimization problems (e.g. in microeconomics, some mechanical problems...)
- Optimality conditions are used to derive algorithms for complex problem
- $\Longrightarrow$ fundamental both for studying optimization as well as other science

What you have to know

- Basic vocabulary : objective, constraint, admissible solution, differentiable optimization problem
- First order necessary KKT conditions

What you really should know

- What is a tangeant cone
- Sufficient qualification conditions (linear and Slater's)
- That KKT conditions are sufficient in the convex case

What you have to be able to do

- Write the KKT condition for a given explicit problem, and use them to solve said problem

What you should be able to do

- Check that constraints are qualified

Contents

1. Markov Decision Problem
2. Convexity
3. Optimality conditions
4. Duality
5. Classification of optimization problem
6. Gradient algorithm(s)
7. Newton and Quasi-Newton algorithm(s)
8. Constrained optimization
9. Interior Points Methods
10. Stochastic Gradient

Why should I bother to learn this stuff ?

- Duality allow a second representation of the same convex problem, giving sometimes some interesting insights (e.g. principle of virtual forces in mechanics)
- Duality is a good way of getting lower bounds
- Duality is a powerful tool for decomposition methods
- $\Longrightarrow$ fundamental both for studying optimization (continuous and operations research)
- $\Longrightarrow$ usefull in other fields like mechanics and machine learning

What you have to know

- Weak duality: $\sup \inf \Phi \leq \inf \sup \Phi$
- Definition of the Lagrangian $\mathcal{L}$
- Definition of primal and dual problem

$$\underbrace{\max_{\lambda,\mu} \inf_{x} \mathcal{L}(x;\lambda,\mu)}_{\text{Dual}} \leq \underbrace{\inf_{x} \max_{\lambda,\mu} \mathcal{L}(x;\lambda,\mu)}_{\text{Primal}}$$

- Marginal interpretation of the optimal multipliers

What you really should know

- A saddle point of $\mathcal{L}$ is a primal-dual optimal pair
- Sufficient condition of strong duality under convexity (Slater's)

What you have to be able to do

- Turn a constrained optimization problem into an unconstrained Min sup problem through the Lagrangian
- Write the dual of a given problem
- Heuristically recover the KKT conditions from the Lagrangian of a problem

What you should be able to do

- Get lower bounds through duality

Contents

1. Markov Decision Problem
2. Convexity
3. Optimality conditions
4. Duality
5. Classification of optimization problem
6. Gradient algorithm(s)
7. Newton and Quasi-Newton algorithm(s)
8. Constrained optimization
9. Interior Points Methods
10. Stochastic Gradient

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.

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.

$$\min_x \quad \max(f_1(x), f_2(x)) = \quad \min_{x,z} \quad z \quad \text{s.t.} \quad f_1(x) \leq z$$
$$f_2(x) \leq z$$

Contents

1. Markov Decision Problem
2. Convexity
3. Optimality conditions
4. Duality
5. Classification of optimization problem
6. Gradient algorithm(s)
7. Newton and Quasi-Newton algorithm(s)
8. Constrained optimization
9. Interior Points Methods
10. Stochastic Gradient

Why should I bother to learn this stuff ?

- Gradient algorithm is the easiest, most robust optimization algorithm. It is not numerically efficient, but numerous more advanced algorithm are built on it.
- Conjugate gradient algorithm(s) are efficient methods for (quasi)-quadratic function. They are in particular used for approximately solving large linear systems.
- $\Longrightarrow$ useful for comprehension of
  - more advanced continuous optimization algorithms
  - machine learning training methods
  - numerical methods for solving discretized PDE

What you have to know

- What is a descent direction method.
- That there is a step-size choice to make.
- That there exists multiple descent direction.
- Gradient method is the slowest method, and in most case you should used more advanced method through adapted library.
- Conditionning of the problem is important for convergence speed.

What you really should know

- A problem can be pre-conditionned through change of variable to get faster results.
- Solving linear system can be done exactly through algebraic method, or approximately (or exactly) through minimization method.
- Conjugate gradient method are efficient tools for (approximately) solving a linear equation.
- Conjugate gradient works by exactly minimizing the quadratic function on an affine subspace.

What you have to be able to do

- Implement a gradient method with receeding step-size.

What you should be able to do

- Implement a conjugate gradient method.
- Use the strongly convex and/or Lipschitz gradient assumptions to derive bounds.

Contents

1. Markov Decision Problem
2. Convexity
3. Optimality conditions
4. Duality
5. Classification of optimization problem
6. Gradient algorithm(s)
7. Newton and Quasi-Newton algorithm(s)
8. Constrained optimization
9. Interior Points Methods
10. Stochastic Gradient

Why should I bother to learn this stuff ?

- Newton algorithm is, in theory, the best black box algorithm for smooth strongly convex function. It is used in practice as well as a stepping step for more advanced algorithm.
- Quasi-Newton algorithms (in particular L-BFGS) are the actual by default algorithm for most smooth black-box optimization library. Used in large scale application (e.g. weather forecast) for decades.
- $\Longrightarrow$ useful for
  - understanding the optimization software you might use as an engineer
  - understanding more advanced methods (e.g. interior points methods)
  - getting an idea of why the convergence might behave strangely in practice

What you have to know

- At least one idea behind Newton's algorithm.
- The Newton step.
- That quasi-Newton methods are almost as good as Newton, without requiring a second order oracle.

What you really should know

- Newton's algorithm default step is 1, but you should use backtracking step anyway.
- Newton's algorithm converges in two phases : a slow damped phase, and a very fast quadratically convergent phase close to the optimum (at most 6 iterations).
- BFGS is the by default quasi-Newton method. It work by updating an approximation of the inverse of the Hessian close to the precedent approximation and satisfying some natural requirement.
- L-BFGS limit the memory requirement by never storing the matrix but only the step and gradient updates.

What you have to be able to do

- Implement a damped Newton method.

What you should be able to do

- Implement a BFGS method (with the update formula in front of your eyes)

Contents

1. Markov Decision Problem
2. Convexity
3. Optimality conditions
4. Duality
5. Classification of optimization problem
6. Gradient algorithm(s)
7. Newton and Quasi-Newton algorithm(s)
8. Constrained optimization
9. Interior Points Methods
10. Stochastic Gradient

Why should I bother to learn this stuff ?

- Most real problems have constraints that you have to deal with.
- This course give a snapshot of the tools available to you.
- $\Longrightarrow$ useful for
  - having an idea of what can be done when you have constraints

What you have to know

- There is three main ways of dealing with constraints:
  - choosing an admissible direction
  - projection of the next iterate
  - penalizing the constraints

What you really should know

- admissible direction methods are mainly usefull for polyhedral constraint set
- projection is usefull only if the admissible set is simple (ball or bound constraints)
- penalization can be inner or outer, differentiable or not.

What you have to be able to do

- Implement a penalization approach.

What you should be able to do

- Implement Uzawa's algorithm.

Contents

1. Markov Decision Problem
2. Convexity
3. Optimality conditions
4. Duality
5. Classification of optimization problem
6. Gradient algorithm(s)
7. Newton and Quasi-Newton algorithm(s)
8. Constrained optimization
9. Interior Points Methods
10. Stochastic Gradient

Why should I bother to learn this stuff ?

- Interior point methods are competitive with simplex method for linear programm
- Interior point methods are state of the art for most conic (convex) problems
- $\Longrightarrow$ useful for
  - understanding what is used in numerical solvers
  - specialization in optimization

What you have to know

- IPM are state of the art algorithms for LP and more generally conic optimization problem
- That logarithmic barrier are a useful inner penalization method

What you really should know

- That Newton's algorithm can be applied with equality constraints
- What is the central path
- That IPM work with inner and outer optimization loop

Contents

1. Markov Decision Problem
2. Convexity
3. Optimality conditions
4. Duality
5. Classification of optimization problem
6. Gradient algorithm(s)
7. Newton and Quasi-Newton algorithm(s)
8. Constrained optimization
9. Interior Points Methods
10. Stochastic Gradient

Why should I bother to learn this stuff ?

- Main algorithm principle for training machine learning model, and in particular deep neural network
- $\Longrightarrow$ useful for
  - understanding how the library train ML models
  - specialization in optimization
  - specialization in machine learning

What you have to know

- That for a stochastic problem gradient step requires to compute an expectation
- That stochastic gradient do not compute the true gradient, but only an estimator of the gradient

What you really should know

- gradient algorithm (or more advanced version) is faster in term of number of iterations
- stochastic gradient needs more iteration, but each is faster

What you have to be able to do

- dive in the scientific litterature on the subject if you need to implement this type of algorithm