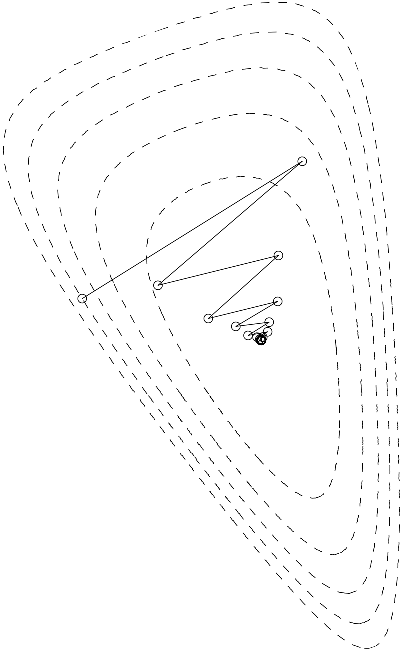

Optimization Algorithms

Augmented Lagrangian

squared penalties, Augmented Lagrangian

Marc Toussaint
Technical University of Berlin
Winter 2024/25

Squared Penalty Method

- To solve the original problem

$$\min_x \; f(x) \;\; \text{s.t.} \;\; g(x) \le 0, \;\; h(x) = 0$$

we define the unconstrained *inner* problem

$$\min_x S(x, \mu, \nu) \; , \quad S(x, \mu, \nu) = f(x) + \mu \sum_i [g_i(x) > 0] \; g_i(x)^2 + \nu \sum_j h_j(x)^2$$

Input: initial $x \in \mathbb{R}^n$, functions $f(x), g(x), h(x)$, tolerances θ, ϵ, parameters (defaults: $\varrho_\mu^+ = \varrho_\nu^+ = 10, \mu_0 = \nu_0 = 1$)

Output: x

1: initialize $\mu = \mu_0$, $\nu = \nu_0$
2: **repeat**
3: solve unconstrained problem $x \leftarrow \text{argmin}_x \, S(x, \mu, \nu)$
4: $\mu \leftarrow \varrho_\mu^+ \mu$, $\nu \leftarrow \varrho_\nu^+ \nu$
5: **until** $|\Delta x| \le \theta$ and $\forall_{i,j} : \; g_i(x) \le \epsilon, |h_i(x)| \le \epsilon$ repeatedly

---

Squared Penalty Method

- Note: Here we increase μ and ν gradually
- Pro:
  - Very simple
  - Quadratic penalties $\rightarrow$ good conditioning for Newton methods $\rightarrow$ efficient convergence
- Con:
  - Will always lead to *some* violation of constraints
  - Conditioning for very large μ, ν

Squared Penalty Method

- Note: Here we increase μ and ν gradually
- Pro:
  - Very simple
  - Quadratic penalties $\rightarrow$ good conditioning for Newton methods $\rightarrow$ efficient convergence
- Con:
  - Will always lead to *some* violation of constraints
  - Conditioning for very large μ, ν
- Better ideas:
  - Add an out-pushing gradient/force $-\nabla g_i(x)$ for every constraint $g_i(x) > 0$ that is violated
  - Ideally, the out-pushing gradient mixes with $-\nabla f(x)$ exactly consistent to ensure stationarity

$\rightarrow$ *Augmented Lagrangian*

Augmented Lagrangian

(We can introduce this is a self-contained manner, without yet defining the “Lagrangian”)

Augmented Lagrangian

- We first consider an *equality* constraint before addressing inequalities
- To solve the original problem

$$\min_x \; f(x) \;\; \text{s.t.} \;\; h(x) = 0$$

we define the unconstrained *inner* problem

$$\min_x \; A(x, \kappa, \nu) \; , \quad A(x, \kappa, \nu) = f(x) + \sum_j \kappa_j h_j(x) + \nu \sum_j h_j(x)^2 \tag{1}$$

- Note:
  - The gradient $\nabla h_j(x)$ is always orthogonal to the constraint
  - By tuning κ_j we can induce a "pushing force" $-\kappa_j \nabla h_j(x)$ (cp. KKT stationarity!)
  - Each term $\nu h_j(x)^2$ penalizes as before, and "pushes" with $-2\nu h_j(x) \nabla h_j(x)$

Augmented Lagrangian

- The approach:
  - First minimize (1) for $\kappa_j = 0$ and some ν $\rightsquigarrow$ this will lead to a (slight) penalty $\nu h_j(x)^2$
  - Then *choose* κ_j *to generate exactly the gradient that was previously generated by the penalty*

Augmented Lagrangian

- The approach:
  - First minimize (1) for $\kappa_j = 0$ and some ν $\rightsquigarrow$ this will lead to a (slight) penalty $\nu h_j(x)^2$
  - Then *choose* κ_j *to generate exactly the gradient that was previously generated by the penalty*
- Let's look at the gradients at the optimum $\min_x A(x, \kappa, \nu)$:

$$x' = \underset{x}{\operatorname{argmin}} \ f(x) + \sum_j \kappa_j h_j(x) + \nu \sum_j h_j(x)^2$$

$$\Rightarrow \quad 0 = \nabla f(x') + \sum_j \kappa_j \nabla h_j(x') + \nu \sum_j 2 h_j(x') \nabla h_j(x')$$

(Describes the force balance between f pulling, penalties pushing, and Lagrange term pushing)

Augmented Lagrangian

- The approach:
  - First minimize (1) for $\kappa_j = 0$ and some ν $\rightsquigarrow$ this will lead to a (slight) penalty $\nu h_j(x)^2$
  - Then *choose* κ_j *to generate exactly the gradient that was previously generated by the penalty*
- Let's look at the gradients at the optimum $\min_x A(x, \kappa, \nu)$:

$$x' = \operatorname*{argmin}_x \; f(x) + \sum_j \kappa_j h_j(x) + \nu \sum_j h_j(x)^2$$

$$\Rightarrow \quad 0 = \nabla f(x') + \sum_j \kappa_j \nabla h_j(x') + \nu \sum_j 2 h_j(x') \nabla h_j(x')$$

(Describes the force balance between f pulling, penalties pushing, and Lagrange term pushing)

- **Augmented Lagrangian Update**: Update κ's for the next iteration to be:

$$\sum_j \kappa_j^{\text{new}} \nabla h_j(x') \overset{!}{=} \sum_j \kappa_j^{\text{old}} \nabla h_j(x') + \nu \sum_j 2 h_j(x') \nabla h_j(x')$$

$$\kappa_j^{\text{new}} = \kappa_j^{\text{old}} + 2 \nu h_j(x')$$

- Why this adaptation of κ_j is elegant:
  - We do *not* have to take the penalty limit $\nu \to \infty$ but still can have *exact* constraints
  - $\to$ Unlike log-barrier and sqr penalty, the method *does not have to increase weights of penalties/barriers*, and *does not lead to extreme conditioning of the Hessian*
  - If f and h were linear (∇f and ∇h_j constant), the updated κ_j is *exactly right*: In the next iteration we would exactly hit the constraint (by construction)

- Why this adaptation of κ_j is elegant:
  - We do *not* have to take the penalty limit $\nu \to \infty$ but still can have *exact* constraints
  - $\to$ Unlike log-barrier and sqr penalty, the method *does not have to increase weights of penalties/barriers*, and *does not lead to extreme conditioning of the Hessian*
  - If f and h were linear (∇f and ∇h_j constant), the updated κ_j is *exactly right*: In the next iteration we would exactly hit the constraint (by construction)

- The Augmented Lagrangian handles equality constraints very efficiently

Augmented Lagrangian with Inequalities

- To solve the original problem

$$\min_x \ f(x) \ \text{ s.t. } \ g(x) \le 0,\ h(x) = 0$$

we define the unconstrained inner problem, $\min_x$ ...

$$A(x, \lambda, \kappa, \mu, \nu) = f(x) + \sum_i \lambda_i g_i(x) + \mu \sum_i [g_i(x) \ge 0 \vee \lambda_i > 0]\ g_i(x)^2 + \sum_j \kappa_j h_j(x) + \nu \sum_j h_j(x)^2$$

- An inequality is either **active** or **inactive**:
  - When active ($g_i(x) \ge 0 \vee \lambda_i > 0$) we aim for equality $g_i(x) = 0$
  - When inactive ($g_i(x) < 0 \wedge \lambda_i = 0$) we don't penalize/augment
  - λ_i are zero or positive, but never negative
- After each inner optimization, we use the **Augmented Lagrangian dual updates**:

$$\lambda_i \leftarrow \max(\lambda_i + 2\mu g_i(x'), 0)\ , \quad \kappa_j \leftarrow \kappa_j + 2\nu h_j(x')\ .$$

Augmented Lagrangian

Input: initial $x \in \mathbb{R}^n$, functions f, g, h, tolances θ, ϵ, parameters (defaults: $\varrho_\mu^+ = \varrho_\nu^+ = 1.2, \mu_0 = \nu_0 = 1$)
Output: x

initialize $\mu = \mu_0, \nu = \nu_0, \lambda_i = 0, \kappa_j = 0$
repeat
  solve unconstrained problem $x \leftarrow \text{argmin}_x\ A(x, \lambda, \kappa, \mu, \nu)$
  $\forall_i :\ \lambda_i \leftarrow \max(\lambda_i + 2\mu g_i(x), 0),\ \forall_j : \kappa_j \leftarrow \kappa_j + 2\nu h_j(x)$
  optionally, $\mu \leftarrow \varrho_\mu^+ \mu, \nu \leftarrow \varrho_\nu^+ \nu$
until $|\Delta x| < \theta$ and $g_i(x) < \epsilon$ and $|h_j(x)| < \epsilon$ repeatedly

- See also: M. Toussaint: A Novel Augmented Lagrangian Approach for Inequalities and Convergent Any-Time Non-Central Updates. e-Print arXiv:1412.4329, 2014.
- Demo...

Comments

- We learnt about three core methods to tackle constrained optimization by repeated unconstrained optimization:
  - Log barrier method
  - Squared penalty method (approximate only)
  - Augmented Lagrangian method

- Next we discuss in more depth the Lagrangian, which will help to also introduce the primal-dual method

- Later we discuss other methods, eg. Simplex, SQP