THE UNIVERSITY OF
CHICAGO

17 AUGMENTED LAGRANGIAN

AUGLAG: Equality Constraints

- The augmented Lagrangian:

$$\mathcal{L}_A(x, \lambda; \mu) \stackrel{\text{def}}{=} f(x) - \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i \in \mathcal{E}} c_i^2(x),$$

- Observation: if

$$\lambda = \lambda^*; \mu \quad \mu_0 \Rightarrow \nabla_x \mathsf{L}_\mathsf{A}\left(x^*, \lambda^*, \mu\right) = 0;$$

$$\nabla^2_{xx} \mathsf{L}_\mathsf{A}\left(x^*, \lambda^*, \mu\right) = \nabla^2_{xx} \mathsf{L}\left(x^*, \lambda^*, \mu\right) + \mu\left(\nabla c\left(x^*\right)\right)^T\left(\nabla c\left(x^*\right)\right)$$

AUGLAG: SOC

- So x* is a stationary point for Auglag for exact multipliers … but is it a minimum?
- Yes, for mu sufficiently large.

$$\nabla^2_{xx} L_A\left(x^*,\lambda^*,\mu\right) \sim [Y \quad Z]^T \nabla^2_{xx} L\left(x^*,\lambda^*\right)[Y \quad Z] + \mu\left(\nabla c\left(x^*\right)Y\right)^T\left(\nabla c\left(x^*\right)Y\right) =$$

$$\begin{bmatrix} Z^T\nabla^2_{xx} L\left(x^*,\lambda^*\right)Z & * \\ * & * + \mu\left(\nabla c\left(x^*\right)Y\right)^T\left(\nabla c\left(x^*\right)Y\right) \end{bmatrix} \pm 0 \quad \text{for } \mu \text{ suff large.}$$

- So it is *almost* as solving unconstrained problem … but how do I find multiplier estimates?

Multiplier Estimates Auglag

- At the current estimate, solve problem

$$0 \approx \nabla_x \mathcal{L}_A(x_k, \lambda^k; \mu_k) = \nabla f(x_k) - \sum_{i \in \mathcal{E}} [\lambda_i^k - \mu_k c_i(x_k)] \nabla c_i(x_k).$$

- The obvious choice:

$$\lambda_i^{k+1} = \lambda_i^k - \mu_k c_i(x_k), \quad \text{for all } i \in \mathcal{E}.$$

- What do I do if I converge lambda but x* is not feasible? Increase the penalty mu (it will have to end increasing eventually).

–

The general case

- The bound constrained formulation. Slacks.

$$c_i(x) \geq 0, i \in \mathcal{I}, \quad \Longrightarrow \quad c_i(x) - s_i = 0, \quad s_i \geq 0, \quad \text{for all } i \in \mathcal{I}.$$

- The problem:

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad c_i(x) = 0, \; i = 1, 2, \ldots, m, \quad l \leq x \leq u.$$

The augmented Lagrangian

- The new AugLag

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) - \sum_{i=1}^{m} \lambda_i c_i(x) + \frac{\mu}{2} \sum_{i=1}^{m} c_i^2(x).$$

- The bound constrained optimization problem:

$$\min_x \mathcal{L}_A(x, \lambda; \mu) \quad \text{subject to } l \leq x \leq u.$$

- Same property: if Lagrange multiplier is the optimal one for eq cons and mu is large enough then x* is a solution !

–

Practical AugLag alg: LANCELOT

Main computation: Use bound constrained projection.

Forcing sequences

Algorithm 17.4 (Bound-Constrained Lagrangian Method).

Choose an initial point x_0 and initial multipliers λ^0;
Choose convergence tolerances η_* and ω_*;
Set $\mu_0 = 10$, $\omega_0 = 1/\mu_0$, and $\eta_0 = 1/\mu_0^{0.1}$;
for $k = 0, 1, 2, \ldots$

Find an approximate solution x_k of the subproblem (17.50) such that

$$\left\| x_k - P\left(x_k - \nabla_x \mathcal{L}_A(x_k, \lambda^k; \mu_k), l, u\right)\right\| \le \omega_k;$$

if $\|c(x_k)\| \le \eta_k$
(* test for convergence *)
if $\|c(x_k)\| \le \eta_*$ and $\left\| x_k - P\left(x_k - \nabla_x \mathcal{L}_A(x_k, \lambda^k; \mu_k), l, u\right)\right\| \le \omega_*$
stop with approximate solution x_k;

end (if)
(* update multipliers, tighten tolerances *)
$\lambda^{k+1} = \lambda^k - \mu_k c(x_k)$;
$\mu_{k+1} = \mu_k$;
$\eta_{k+1} = \eta_k / \mu_{k+1}^{0.9}$;
$\omega_{k+1} = \omega_k / \mu_{k+1}$;
else
(* increase penalty parameter, tighten tolerances *)
$\lambda^{k+1} = \lambda^k$;
$\mu_{k+1} = 100\mu_k$;
$\eta_{k+1} = 1/\mu_{k+1}^{0.1}$;
$\omega_{k+1} = 1/\mu_{k+1}$;
end (if)
end (for)

Solving the bound constrained subproblem

- It is an iterative bound constrained optimization algorithm with trust-region (note the Gauss-Newton flavor)

$$\min_d \ \tfrac{1}{2} d^T \left[\nabla^2_{xx}\mathcal{L}(x_k, \lambda^k) + \mu_k A_k^T A_k\right] d + \nabla_x \mathcal{L}_A(x_k, \lambda^k; \mu_k)^T d$$
$$\text{subject to } \ l \le x_k + d \le u, \qquad \|d\|_\infty \le \Delta,$$

- Each step solves a bound constrained QP (not necessarily PD),
- The difference: after a subspace solve: compute the new derivative and update TR based on performance with replacement

$$f(x) \rightarrow \mathsf{L}_{\mathsf{A}}(x, \lambda, \mu)$$

- Therefore, when implementing the trust region we use

$$\rho = \frac{\mathsf{L}_A(x_k + p_k, \lambda, \mu) - \mathsf{L}_A(x_k + p_k, \lambda, \mu)}{m(p_k) - m(0)}$$

Note the building blocks

- Projected Gradient for QP
  - Projected Gradient for LP
  - CG
- Trust Region Logic for bound constrained NLP
- KKT-inspired augmented Lagrangian definition and multiplier update.
- It is great for the first implementation, but it is only linearly convergent.