Lecture Outline

- Unconstrained minimization problems
- Gradient method
- Examples of distributed algorithms
- Distributed algorithm: introduction
- Decomposition: primal and dual decompositions
- Gauss-Siedel and Jacobi update
Unconstrained Minimization Problems

Given \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) convex and twice differentiable:

minimize \( f(x) \)

Optimizer \( x^* \). Optimized value \( p^* = f(x^*) \)

Necessary and sufficient condition of optimality:

\( \nabla f(x^*) = 0 \)

Solve a system of nonlinear equations: \( n \) equations in \( n \) variables

Iterative algorithm: computes a sequence of points \( \{x^{(0)}, x^{(1)}, \ldots \} \) such that

\[
\lim_{k \to \infty} f(x^{(k)}) = p^*
\]

Terminate algorithm when \( f(x^{(k)}) - p^* \leq \epsilon \) for a specified \( \epsilon > 0 \)
Examples

- **Least-squares**: minimize

\[ \|Ax - b\|_2^2 = x^T (A^T A)x - 2(A^T b)^T x + b^T b \]

Optimality condition is system of linear equations:

\[ A^T Ax^* = A^T b \]

called normal equations for least-squares

- **Unconstrained geometric programming**: minimize

\[ f(x) = \log \left( \sum_{i=1}^{m} \exp(a_i^T x + b_i) \right) \]

Optimality condition has no analytic solution:

\[ \nabla f(x^*) = \frac{1}{\sum_{j=1}^{m} \exp(a_j^T x^* + b_j)} \sum_{i=1}^{m} \exp(a_i^T x^* + b_i) a_i = 0 \]
**Strong Convexity**

\( f \) assumed to be strongly convex: there exits \( m > 0 \) such that

\[
\nabla^2 f(x) \succeq mI
\]

which also implies that there exists \( M \geq m \) such that

\[
\nabla^2 f(x) \preceq MI
\]

Bound optimal value:

\[
f(x) - \frac{1}{2m}\|\nabla f(x)\|_2^2 \leq p^* \leq f(x) - \frac{1}{2M}\|\nabla f(x)\|_2^2
\]

Suboptimality condition:

\[
\|\nabla f(x)\|_2 \leq (2m\epsilon)^{1/2} \Rightarrow f(x) - p^* \leq \epsilon
\]

Distance between \( x \) and optimal \( x^* \):

\[
\|x - x^*\|_2 \leq \frac{2}{m} \|\nabla f(x)\|_2
\]
Descent Methods

Minimizing sequence $x^{(k)}$, $k = 1, \ldots$, (where $t^{(k)} > 0$)

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}$$

$\Delta x^{(k)}$: search direction

$t^{(k)}$: step size

Descent methods:

$$f(x^{(k+1)}) < f(x^{(k)})$$

By convexity of $f$, search direction must make an acute angle with negative gradient:

$$\nabla f(x^{(k)})^T \Delta x^{(k)} < 0$$

Because otherwise, $f(x^{(k+1)}) \geq f(x^{(k)})$ since

$$f(x^{(k+1)}) \geq f(x^{(k)}) + \nabla f(x^{(k)})^T (x^{k+1} - x^{(k)})$$
General Descent Method

GIVEN a starting point $x \in \text{dom } f$

REPEAT

1. Determine a descent direction $\Delta x$

2. Line search: choose a step size $t > 0$

3. Update: $x := x + t\Delta x$

UNTIL stopping criterion satisfied
Line Search

• Exact line search:

\[ t = \arg\min_{s \geq 0} f(x + s\Delta x) \]

• Backtracking line search:

GIVEN a descent direction \( \Delta x \) for \( f \) at \( x \), \( \alpha \in (0, 0.5), \beta \in (0, 1) \)

\( t := 1 \)

WHILE \( f(x) - f(x + t\Delta x) < \alpha|\nabla f(x)^T (t\Delta x)| \), \( t := \beta t \)

Caution: \( t \) such that \( x + t\Delta x \in \text{dom } f \)

• Diminishing stepsize: \( t = \frac{t_0}{n} \)

• Constant stepsize: \( t = t_0 \)

Tradeoff between convergence and rate of convergence
Gradient Descent Method

GIVEN a starting point \( x \in \text{dom} \ f \)

REPEAT

1. \( \Delta x := -\nabla f(x) \)

2. Line search: choose a step size \( t > 0 \)

3. Update: \( x := x + t \Delta x \)

UNTIL stopping criterion satisfied

Theorem: we have \( f(x^{(k)}) - p^* \leq \epsilon \) after at most

\[
\frac{\log((f(x^{(0)}) - p^*)/\epsilon)}{\log\left(\frac{1}{1-m/M}\right)}
\]

iterations of gradient method with exact line search
Example in $\mathbb{R}^2$

minimize $f(x) = \frac{1}{2}(x_1^2 + \gamma x_2^2)$, $x^* = (0, 0)$

Gradient descent with exact line search:

$x_1^{(k)} = \gamma \left( \frac{\gamma - 1}{\gamma + 1} \right)^k$, $x_2^{(k)} = \left( -\frac{\gamma - 1}{\gamma + 1} \right)^k$
Which error decay curve is by backtracking and which is by exact line search?
Observations

• Exhibits approximately linear convergence (error $f(x^{(k)}) - p^*$ converges to zero as a geometric series)

• Choice of $\alpha, \beta$ in backtracking line search has a noticeable but not dramatic effect on convergence speed

• Exact line search improves convergence, but not always with significant effect

• Convergence speed depends heavily on condition number of Hessian

Now we move on to distributed algorithm before returning to more centralized algorithms later
Example: Cellular Power Control

Variables: mobile user transmit powers (and target SIR)

Constants: channel gains, noise (and target SIR)

Objective function: minimize power or maximize SIR utility

Constraints: SIR feasibility

Distributed solution 1: no explicit message passing (Foschini, Miljanic 1993)

Distributed solution 2: some message passing between BS and MS (Hande, Rangan, Chiang 2006)

More in Lecture 12
Example: DSL Spectrum Management

Variables: power loading on each tone by each line

Constants: crosstalk gain coefficients, noise

Objective: maximize rate region

Constraints: total transmit power per line

Distributed solution 1: no explicit message passing (Yu, Ginis, Cioffi 2002)

Distributed solution 2: message passing at model synchronization (Cendrillon, Huang, Chiang, Moonen 2007)

More in Lecture 13
Example: Internet Congestion Control

Variables: Transmission rate by each source

Constants: Routing and link capacities

Objective: Maximize network utility

Constraints: Flow feasibility

Distributed solution: no explicit message passing (Kelly et al. 1998, Low et al. and Srikant et al. 1999-2002)

More in Lecture 6
Distributed Algorithms

Distributed algorithms are preferred because:

- Centralized command is not feasible or is too costly
- It’s scalable
- It’s robust

Key issues:

- **Local computation vs. global communication**
- Scope, scale, and physical meaning of communication **overhead**
- **Theoretical issues**: Convergence? Optimality? Speed?
- **Practical issues**: Robustness? Synchronization? Complexity? Stability?
- Problem **separability structure** for decomposition: **vertical** and **horizontal**
Decomposition Structures and Distributed Algorithms

Distributedness is not as well-defined as convexity

Distributedness involves two steps:
Decomposition structures
Distributed algorithms
These two are related, but not the same
Decomposition: Simple Example

Convex optimization with variables $u, v$:

maximize $f_1(u) + f_2(v)$

subject to $A_1 u \preceq b_1$
$A_2 v \preceq b_2$
$F_1 u + F_2 v \preceq h$

Coupling constraint: $F_1 u + F_2 v \preceq h$. Otherwise, separable into two subproblems.
Primal Decomposition

Introduce variable $z$ and rewrite coupling constraint as

$$F_1 u \preceq z, \quad F_2 v \preceq h - z$$

Decomposed into a master problem and two subproblems:

$$\minimize_z \phi_1(z) + \phi_2(z)$$

where

$$\phi_1(z) = \inf_u \{ f_1(u) | A_1 u \preceq b_1, F u \preceq z \}$$

$$\phi_2(z) = \inf_v \{ f_2(v) | A_2 v \preceq b_2, F_2 v \preceq h - z \}$$
For each iteration $t$:

1. **Solve two separate subproblems** to obtain optimal $u(t), v(t)$ and associated dual variables $\lambda_1(t), \lambda_2(t)$

2. **Gradient update**: $g(t) = -\lambda_1(t) + \lambda_2(t)$

3. **Master algorithm update**: $z(t+1) = z(t) - \alpha(t)g(t)$ where $\alpha(t) \geq 0$, $\lim_{t \to \infty} \alpha_t = 0$ and $\sum_{t=1}^{\infty} \alpha(t) = \infty$

**Interpretation:**

- $z$ fixes allocation of resources between two subproblems and master problem iteratively finds best allocation of resources
- More of each resource is allocated to the subproblem with larger Lagrange multiplier at each step
Dual Decomposition

Form partial Lagrangian:

\[ L(u, v, \lambda) = f_1(u) + f_2(v) + \lambda^T (F_1 u + F_2 v - h) \]
\[ = (F_1^T \lambda)^T u + f_1(u) + (F_2^T)^T v + f_2(v) - \lambda^T h \]

Dual function:

\[ q(\lambda) = \inf_{u, v} \{ L(u, v, \lambda) | A_1 u \leq b_1, A_2 v \leq b_2 \} \]
\[ = -\lambda^T h + \inf_{u: A_1 u \leq b_1} ((F_1^T \lambda)^T u + f_1(u)) + \inf_{v: A_2 v \leq b_2} ((F_2^T \lambda)^T v + f_2(v)) \]

Dual problem:

\[ \text{maximize} \quad q(\lambda) \]
\[ \text{subject to} \quad \lambda \succeq 0 \]
Dual Decomposition

Solve the following subproblem in $u$, with minimizer $u^*(\lambda(t))$

\[
\text{minimize } (F_1^T \lambda(t))^T u + f_1(u) \\
\text{subject to } A_1 u \preceq b_1
\]

Solve the following subproblem in $v$, with minimizer $v^*(\lambda(t))$

\[
\text{minimize } (F_2^T \lambda(t))^T v + f_2(v) \\
\text{subject to } A_2 v \preceq b_2
\]

Use the following gradient (to $-q$) to update $\lambda$:

\[
g(t) = -F_1 u^*(\lambda(t)) - F_2 v^*(\lambda(t)) + h, \quad \lambda(t + 1) = \lambda(t) - \alpha(t) g(t)
\]

Interpretation:

Master algorithm adjusts prices $\lambda$, which regulates the separate solutions of two subproblems
Jacobi and Gauss-Siedel Algorithms

In general, Jacobi algorithm ($F_i$ is $i$th component of function $F$):

$$x_i(t + 1) = F_i(x_1(t), \ldots, x_n(t))$$

Gauss-Siedel algorithm:

$$x_i(t + 1) = F_i(x_1(t + 1), \ldots, x_{i-1}(t + 1), x_i(t), \ldots, x_n(t))$$

Nonlinear minimization: Jacobi algorithm:

$$x_i(t + 1) = \arg\min_{x_i} f(x_1(t), \ldots, x_n(t))$$

Gauss-Siedel algorithm:

$$x_i(t + 1) = \arg\min_{x_i} f(x_1(t + 1), \ldots, x_{i-1}(t + 1), x_i(t), \ldots, x_n(t))$$

If $f$ is convex, bounded below, differentiable, and strictly convex for each $x_i$, then Gauss-Siedel algorithm converges to a minimizer of $f$
Lecture Summary

- Iterative algorithm with descent steps for unconstrained minimization problems
- Decouple a coupling constraint: primal or dual decomposition

Readings: Chapters 9.1-9.3, 9.5 in Boyd and Vandenberghe
Chapters 3.2-3.4, 7.5 in D. P. Bertsekas and J. N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, Athena Scientific 1999