Tutorial: Getting Started with Optimization: Computational Noise, Noisy Derivatives, Stochastic Methods

Todd Munson and **Stefan Wild**

Argonne National Laboratory, Mathematics and Computer Science Division

January 28, 2016

Do this now:

- Obtain `matlab` files (from Simon's cluster or at `www.mcs.anl.gov/~wild/codes/zice16.zip`)
- Open `matlab` (ideally on your own machine so that you can view graphics, otherwise on the cluster)
- *[Optional:]* Have your function ready (a `matlab` function that receives x and outputs $f(x)$)

I. Computational Noise

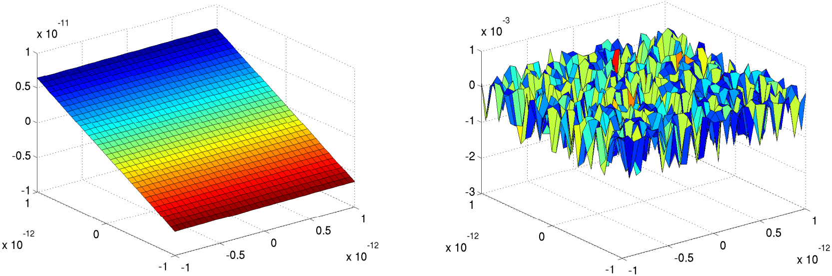
- What is computational noise?
- How can noise be estimated efficiently?
- How does noise affect numerical differentiation?
- How accurate are near-optimal finite-difference estimates?

Questions To Ask Yourself

1. Do you know how “noisy” your function is?
2. Do you know how accurate your derivatives are?
3. Is the noise/accuracy stationary (independent of x)?
4. What do you do with this information?

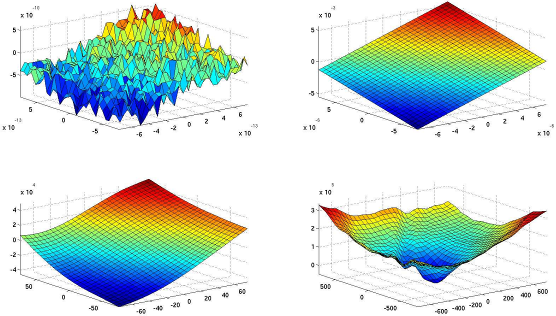
Noise May Hurt You, Or It May Not

These are the same problem:



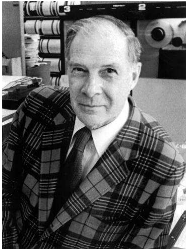
Noise May Hurt You, Or It May Not

So are these:



Computational Noise is not a Newcomer

From Hamming's 1971 Introduction to Numerical Analysis:



Where does this noise come from? *. . . infinite processes in mathematics which of necessity must be approximated by finite processes.*

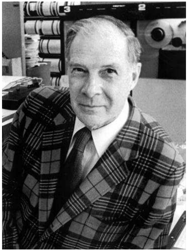
Truncation vs. roundoff *Finite number length leads to roundoff. Finite processes lead to truncation.*

Competing errors *Smaller steps usually reduce truncation error and may increase roundoff error.*

Deterministic *In practice, the same input, barring machine failures, gives the same result.*

Computational Noise is not a Newcomer

From Hamming's 1971 Introduction to Numerical Analysis:



Where does this noise come from? *...infinite processes in mathematics which of necessity must be approximated by finite processes.*

Truncation vs. roundoff *Finite number length leads to roundoff. Finite processes lead to truncation.*

Competing errors *Smaller steps usually reduce truncation error and may increase roundoff error.*

Deterministic *In practice, the same input, barring machine failures, gives the same result.* ← changing!

Living In A Finite-Precision World

Roundoff Error

$$f_\infty(x) - f(x)$$

Floating Point Arithmetic

Commutative:

$$A + B = B + A \qquad \text{and} \qquad A * B = B * A$$

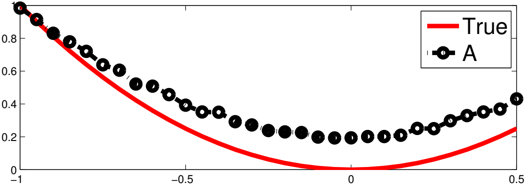
Non-associative:

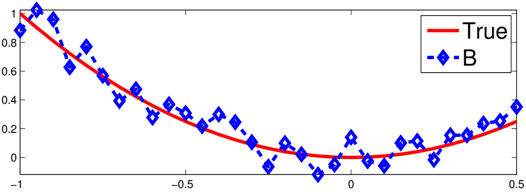
$$A + (B + C) \neq (A + B) + C$$

- This is likely to affect the reproducibility of your calculations in the future (for performance reasons)

Many details $\longrightarrow$ [What Every Computer Scientist Should Know About Floating-Point Arithmetic, Goldberg, 1991]

Truncation/Approximation Error





Truncation error

$$R_{m+1}(x) = f_a(x) - \sum_{i=0}^{m} P_i(x)$$

Which do you prefer?

- A less noise, more error
- B less error, more noise

Computational Noise in Deterministic Simulations

Finite precision $+$ finite processes

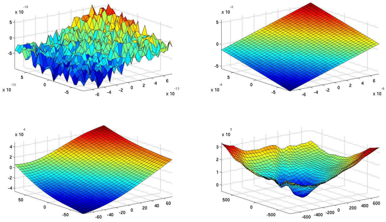
- Iteratively solving systems of PDEs or estimating eigenvalues
- Adaptively computing integrals
- Discretizations/meshes

destroy underlying smoothness

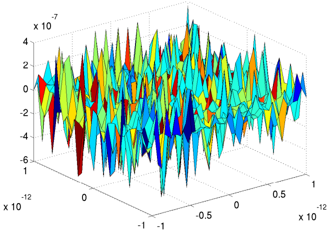
Goal: estimate the "variation" in $f(\mathbf{x})$

- a few f evaluations
- deterministic and stochastic noise

Difference $|f(x) - f(x + Z\omega)|$,



X-ray microscopy simulation



Sparse linear large-scale system

Matlab Time

Basic tips

(Examples in `runexamples.m`)

- Moving from n-d to 1-d
- Deterministic function (`probnum=1`)
- Stochastic function (`probnum=2`)
- Scaling (`probnum=3`)
- Constraint cautions

Estimating Computational Noise: The Noise Level ϵ_f

Simple model for the noise

$$f(t) = f_s(t) + \varepsilon(t), \quad t \in \mathcal{I}$$

- f the computed function
- f_s a smooth, deterministic function
- ε is the noise with $\{\varepsilon(t) : t \in \mathcal{I}\}$ iid ← only assumption

The noise level of f is $\varepsilon_f = \left(\mathrm{Var}\left\{\varepsilon(t)\right\}\right)^{1/2}$

(independent of t)

The k-th Order Difference $\Delta^k f(t)$

$$\Delta^{k+1} f(t) = \Delta^k f(t+h) - \Delta^k f(t), \qquad \Delta^0 f(t) = f(t)$$

$$\Delta^k f(t) = \Delta^k f_s(t) + \Delta^k \varepsilon(t)$$

1. Differences of smooth f_s tend to zero rapidly
2. Differences of noise are bounded away from zero
   - If h is sufficiently small,
     $$\Delta^k f(t) \approx \Delta^k \varepsilon(t)$$
   - If f_s is k-times differentiable,
     $$\Delta^k f(t) = f_s^{(k)}(\xi_k) h^k + \Delta^k \varepsilon(t), \qquad \xi_k \in (t, t+kh)$$

Goal: make h small enough to remove smooth component

Theory Underlying the ECNoise Algorithm

For $\{\varepsilon(t+ih) : i = 0, \ldots, m\}$ iid and $k \le m$:

1. $\mathrm{E}\left\{\Delta^k \varepsilon(t)\right\} = 0$
2. $\gamma_k \mathrm{E}\left\{[\Delta^k \varepsilon(t)]^2\right\} = \varepsilon_f^2 \qquad \gamma_k = \frac{(k!)^2}{(2k)!}$
3. If f_s is continuous at t, then
$$\lim_{h\to 0} \gamma_k \mathrm{E}\left\{\left[\Delta^k f(t)\right]^2\right\} = \varepsilon_f^2$$
4. If f_s is k-times continuously differentiable at t, then
$$\lim_{h\to 0} \frac{\gamma_k \mathrm{E}\left\{[\Delta^k f(t)]^2\right\} - \varepsilon_f^2}{h^{2k}} = \gamma_k \left[f_s^{(k)}(t)\right]^2$$

$$\Rightarrow \varepsilon_f^2 \approx \gamma_k \mathrm{E}\left\{[\Delta^k f(t)]^2\right\},$$

when the sampling distance h is sufficiently small

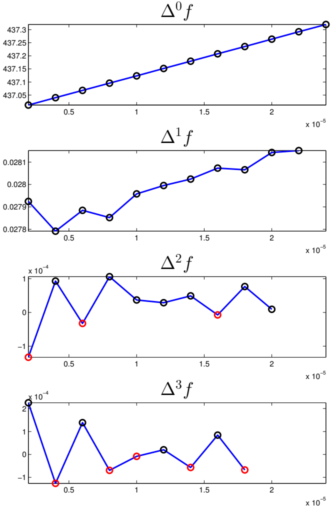
The `ECNoise` Algorithm

Uses $\sigma_k = \left(\frac{\gamma_k}{m+1-k} \sum_{i=0}^{m-k} [\Delta^k f(t+ih)]^2 \right)^{1/2}$

1. Chooses k
2. Verifies h is small enough

◇ Works for deterministic f

[Estimating Computational Noise. Moré & W., SISC 2011]



ECNoise Estimator $\sigma_k = \left(\frac{\gamma_k}{m+1-k} \sum_{i=0}^{m-k} [\Delta^k f(t_i)]^2 \right)^{1/2}$

For $f(t) = \cos(t) + \sin(t) + 10^{-3} U_{[0,2\sqrt{3}]}$ $\left(m = 6, t_i = \frac{i}{100}\right)$

$f(t_i)$	$\Delta f(t_i)$	$\Delta^2 f(t_i)$	$\Delta^3 f(t_i)$	$\Delta^4 f(t_i)$	$\Delta^5 f(t_i)$	$\Delta^6 f(t_i)$
1.003	7.54e-3	2.15e-3	1.87e-4	-5.87e-3	1.46e-2	-2.49e-2
1.011	9.69e-3	2.33e-3	-5.68e-3	8.73e-3	-1.03e-2	
1.021	1.20e-2	-3.35e-3	3.05e-3	-1.61e-3		
1.033	8.67e-3	-2.96e-4	1.44e-3			
1.041	8.38e-3	1.14e-3				
1.050	9.52e-3					
1.059						
σ_k	6.78e-3	8.96e-4	9.02e-4	9.93e-4	1.10e-3	1.14e-3

Extension to Multivariate $g : \mathbb{R}^n \mapsto \mathbb{R}$

Given base point $x_b \in \mathbb{R}^n$, unit direction $p \in \mathbb{R}^n$, consider

$$f_p(t) = g(x_b + tp), \quad t \geq 0$$

Apply univariate theory

- Directional differences, directional derivatives
- ε_f may now depend on a direction $p \in \mathbb{R}^n$
- `ECnoise` uses $T_{i,0} = f(x_b + ihp)$ with random unit direction $p \in \mathbb{R}^n$

Computational Experience with Stochastic Noise

Validate ECnoise and empirical properties of

$$\sigma_k^2 = \frac{\gamma_k}{m+1-k} \sum_{i=0}^{m-k} T_{i,k}^2$$

under known conditions:

- Known noise level ε_f
- Theory directly applies

Target: every estimate within a factor $\eta = 4$ of the mean

Noisy Quadratic, $f(x) = (x^T x)(1 + R), \quad x \in \mathbb{R}^{10}$

Estimate relative noise
$\frac{\sigma_k}{f(x_b)} \approx \sqrt{\mathrm{Var}\left\{R\right\}} = 10^{-3}$

- x_b random base point
- p 10000 random unit directions
- m evaluations

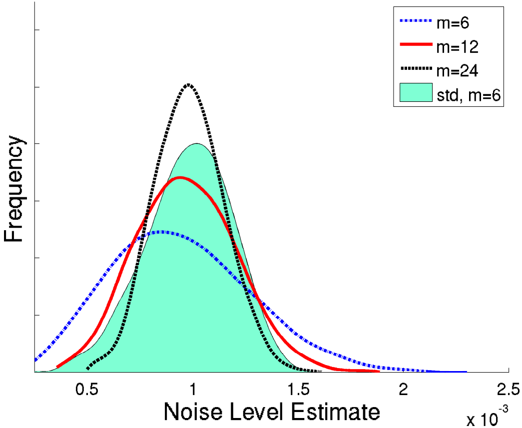
Noisy Quadratic, $f(x) = (x^T x)(1 + R), \quad x \in \mathbb{R}^{10}$

$$R \sim \text{Uniform}\left[-\sqrt{3}\cdot 10^{-3}, \sqrt{3}\cdot 10^{-3}\right]$$

Estimate relative noise $\frac{\sigma_k}{f(x_b)} \approx \sqrt{\mathrm{Var}\{R\}} = 10^{-3}$

- x_b random base point
- p 10000 random unit directions
- m evaluations

99.2% within a factor $\eta = 4$ for $m = 6$



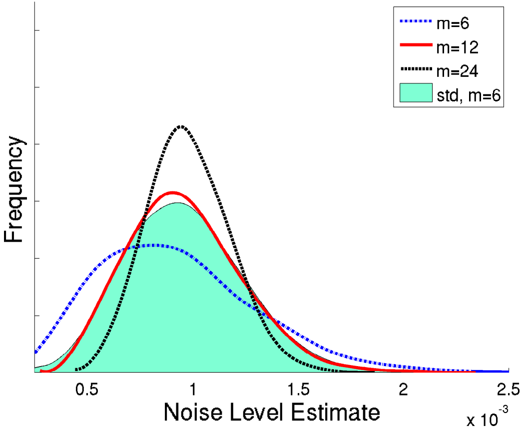
Noisy Quadratic, $f(x) = (x^T x)(1 + R), \quad x \in \mathbb{R}^{10}$

$R \sim$ Normal$\left(0, 10^{-6}\right)$

Estimate relative noise
$\frac{\sigma_k}{f(x_b)} \approx \sqrt{\mathrm{Var}\{R\}} = 10^{-3}$

- x_b random base point
- p 10000 random unit directions
- m evaluations

98.9% within a factor $\eta = 4$ for $m = 6$



MC Finance Example with Higher Order Derivatives

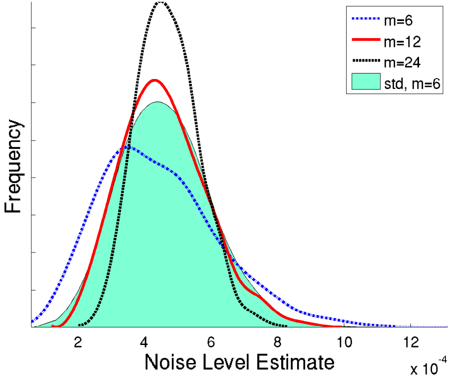
Today's value of a $1 payment n years from now rates [Caflisch]:

$$f(x) = \frac{1}{(2\pi)^{n/2}} \int_{\mathbb{R}^n} \prod_{i=0}^{n} \frac{e^{-\frac{\|u\|^2}{2}}}{1+r_i(u,x)} du, \quad r_i(u,x) = \begin{cases} \frac{1}{10} & i=0 \\ r_{i-1}(u,x) e^{x_i u_i - x_i^2/2} & i \geq 1 \end{cases}$$

10000 MC integrations (directions p) with

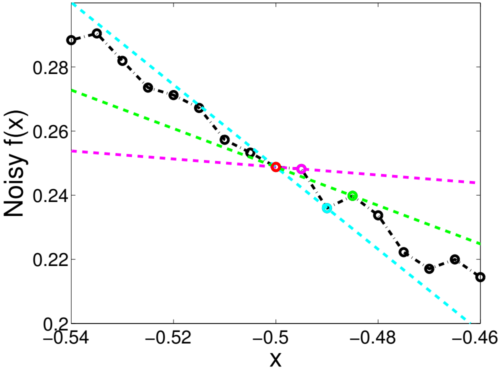
- $n = 3$ years, $x_b = [.1, \ .1, \ .1]$
- $tol = 5000$ standard normal random variables
- no variance reduction

99.6% within a factor 4 for $m = 6$



Finite Differences Sensitive to Choice of h

$$\frac{f(t_0+h)-f(t_0)}{h} \approx f_s'(t_0)$$



Noisy Forward Differences

Minimize $$\mathrm{E}\left\{\mathcal{E}(h)\right\} = \mathrm{E}\left\{\left(\frac{f(t_0+h)-f(t_0)}{h} - f_s'(t_0)\right)^2\right\}$$

Our h will depend on

- Loose estimate of noise
- Loose estimate of $|f''|$ **!**
- Stochastic theory:
  1. $f(t) = f_s(t) + \epsilon$ on $I = \{t_0 + h : 0 \leq h \leq h_0\}$
  2. f_s twice differentiable
  3. $\mu_L \leq |f_s''| \leq \mu_M$ on I

[Estimating Noisy Derivatives. Moré & W., TOMS 2012]]

Optimal Forward Difference Parameter h

$$\frac{1}{4}\mu_L^2 h^2 + 2\frac{\varepsilon_f^2}{h^2} \leq \mathrm{E}\left\{\mathcal{E}(h)\right\} \leq \frac{1}{4}\mu_M^2 h^2 + 2\frac{\varepsilon_f^2}{h^2}$$

$h \downarrow$ Variance (noise) dominates

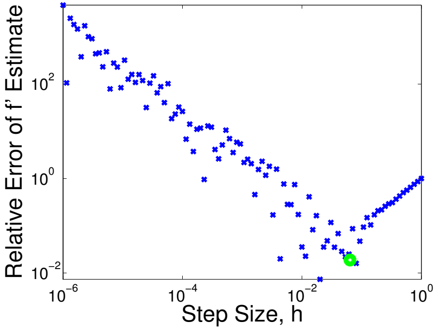
$h \uparrow$ Bias (f'') dominates

For h_0 sufficiently large

1. Upper bound minimized by

   $$h_M = 8^{1/4}\left(\frac{\varepsilon_f}{\mu_M}\right)^{1/2}$$

2. When $\mu_L > 0$, h_M is near-optimal:



$$\mathrm{E}\left\{\mathcal{E}(h_M)\right\} = \sqrt{2}\mu_M\varepsilon_f \leq \left(\frac{\mu_M}{\mu_L}\right) \min_{0\leq h\leq h_0} \mathrm{E}\left\{\mathcal{E}(h)\right\}.$$

Alternative FD Step Sizes

[Gill, Murray, Saunders, Wright; 1983]

Given uniform bound on roundoff error,

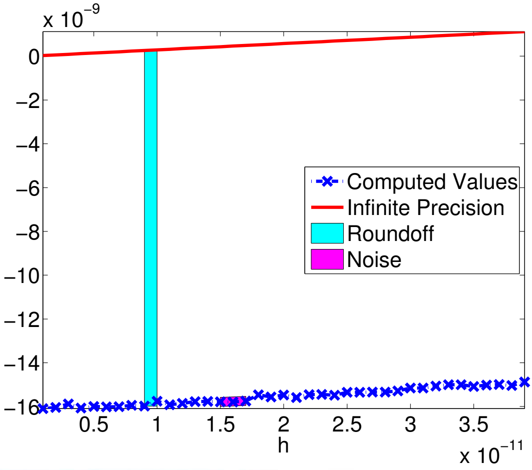
$$|f(t) - f_\infty(t)| \le \varepsilon_A \qquad t \in I,$$

Minimizer of (upper bound on) l_1 error is

$$h_A = 2\left(\frac{\varepsilon_A}{\mu_M}\right)^{1/2}$$

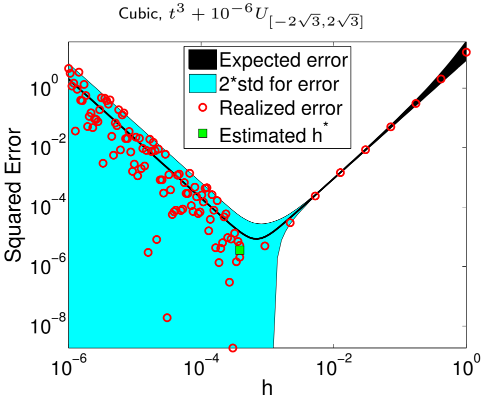
Assumes:

- $h_A \le h_0$
- Estimate of ε_A available



Stochastic Examples

Estimate $f_s'(t) = E\{f(t)\}'$ at $t = 1$ $(\varepsilon_f = 10^{-6})$



Log-log realizations of $\mathcal{E}(h) = \mathrm{E}\left\{\left(\frac{f(t_0+h)-f(t_0)}{h} - f_s'(t_0)\right)^2\right\}$

Expected error and uncertainty regions predicted by the theory

Extension: Central Differences

First derivatives, $\frac{f(t_0+h)-f(t_0-h)}{2h}$

- $|h_M| = \gamma_5 \left(\frac{\varepsilon_f}{\mu_M}\right)^{1/3}, \qquad \gamma_5 = 3^{1/3} \approx 1.44$
- $\mu_L \leq |f_s^{(3)}| \leq \mu_M$
- $\mathrm{E}\{\mathcal{E}_c(h_M)\} \leq \left(\frac{\mu_M}{\mu_L}\right)^{2/3} \min_{|h|\leq h_0} \mathrm{E}\{\mathcal{E}_c(h)\}$

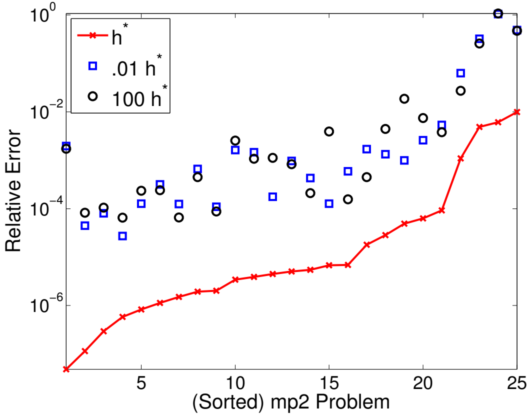
Second derivatives, $\frac{f(t_0+h)-2f(t_0)+f(t_0-h)}{h^2}$

- $|h_M| = \gamma_7 \left(\frac{\varepsilon_f}{\mu_M}\right)^{1/4}, \qquad \gamma_7 = 2^{5/8}\, 3^{1/8} \approx 2.33$
- $\mu_L \leq |f_s^{(4)}| \leq \mu_M$
- $\mathrm{E}\{\mathcal{E}_2(h_M)\} \leq \left(\frac{\mu_M}{\mu_L}\right) \min_{|h|\leq h_0} \mathrm{E}\{\mathcal{E}_2(h)\}$
  - use to obtain rough estimate of $|f_s''|$ for forward-difference h

Ex.- Highly Nonlinear `MINPACK-2` Problems

25 problems, $n \leq 64 \cdot 10^4$

- Accurate estimates obtained even when f'' not constant

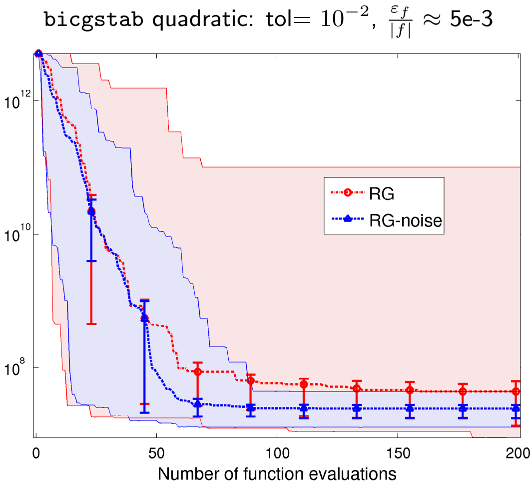


Compared with hand-coded derivative

Using the Noise in Nesterov's Random Gradient Method

General RG iteration

1. Generate direction d_k
2. Evaluate gradient-free oracle $g(x_k; h_k) = \frac{f(x_k + h_k\, d_k) - f(x_k)}{h} d_k$
3. Compute $x_{k+1} = x_k - \delta_k g(x_k; h_k)$, evaluate $f(x_{k+1})$



Matlab Time II

- Start playing around with stochastic algorithms
- Notice cost of getting fd parameter wrong
- Notice cost of not using derivatives

Summary: How Loud Are Your Functions?

- Computational noise complicates analysis of real-world functions, worst-case bounds overly pessimistic
- With a few (6-8) additional evaluations, ECNoise reliably estimates the noise
- Stochastic theory for near-optimal difference parameters
- Coarse estimates of $|f''|$ (2-4 evaluations) yield more accurate directional derivatives
- Both work on deterministic functions in practice

Some refs http://mcs.anl.gov/∼wild:
[*Estimating Computation Noise*, SISC 2011]
[*Estimating Derivatives of Noisy Simulations*, TOMS 2012]
[*Do You Trust Derivatives or Differences?*, JCP 2014]
[*Obtaining Quadratic Models of Noisy Functions*, Preprint, 2014]

Computing http://mcs.anl.gov/∼wild/cnoise

Merci!

Part II?

Stochastic Methods for Two Types of Problems

A. Stochastic optimization
   - Modeling and algorithms for optimization under uncertainty
   - Stochasticity from problem and/or algorithm

B. Deterministic optimization
   - Objectives and constraints deterministic
   - Methods are "randomized"

Stochastic Methods for Two Types of Problems

A. Stochastic optimization
  - Modeling and algorithms for optimization under uncertainty
  - Stochasticity from problem and/or algorithm

B. Deterministic optimization
  - Objectives and constraints deterministic
  - Methods are "randomized"

→ Methods and analysis are related

A. Stochastic Optimization Problems and Methods

Stochastic Optimization

General problem

$$\min\left\{f(x) = \mathbb{E}_\xi\left[F(x,\xi)\right] :\ x \in X\right\} \tag{1}$$

- $x \in \mathbb{R}^n$ decision variables
- ξ vector of random variables
  - independent of x
  - $P(\xi)$ distribution function for ξ
  - ξ has support Ξ
- $F(x,\cdot)$ functional form of uncertainty for decision x
- $X \subseteq \mathbb{R}^n$ set defined by deterministic constraints
  - Also: stochastic/probabilistic constraints (not addressed here)

Approach of Sampling Methods for $f(x) = \mathbb{E}_\xi\left[F(x,\xi)\right]$

- Let $\xi^1, \xi^2, \cdots, \xi^N \sim P$
- For $x \in X$, define:

$$f_N(x) = \frac{1}{N}\sum_{i=1}^{N} F(x, \xi^i)$$

  - f_N is a random variable (really, a stochastic process) (depends on $(\xi^1, \xi^2, \cdots, \xi^N)$)
  - Motivated by $\mathbb{E}_\xi\left[f_N(x)\right] = f(x)$

Bias of Sampling Methods

- Let $f^* = f(x^*)$ for $x^* \in X^* \subseteq X$

Bias of Sampling Methods

- Let $f^* = f(x^*)$ for $x^* \in X^* \subseteq X$
- For any $N \geq 1$:
$$\mathbb{E}_\xi \left[f_N^* \right] \leq f^* = \mathbb{E}_\xi \left[F(x^*, \xi) \right]$$
because
$$\mathbb{E}_\xi \left[f_1^* \right] = \mathbb{E}_\xi \left[\min \left\{ F(x, \xi) : x \in X \right\} \right] \leq \min \left\{ \mathbb{E}_\xi \left[F(x, \xi) \right] : x \in X \right\} = f^*$$

Bias of Sampling Methods

- Let $f^* = f(x^*)$ for $x^* \in X^* \subseteq X$
- For any $N \geq 1$:
$$\mathbb{E}_\xi \left[f_N^*\right] \leq f^* = \mathbb{E}_\xi \left[F(x^*, \xi)\right]$$
because
$$\mathbb{E}_\xi \left[f_1^*\right] = \mathbb{E}_\xi \left[\min\left\{F(x,\xi) : x \in X\right\}\right] \leq \min\left\{\mathbb{E}_\xi \left[F(x,\xi)\right] : x \in X\right\} = f^*$$

- Sampling problems result in optimal values below f^*
- f_N^* is biased estimator of f^*

Sample Average Approximation

- Draw realizations $\hat{\xi}^1, \hat{\xi}^2, \cdots, \hat{\xi}^N \sim P$ of $(\xi^1, \xi^2, \cdots, \xi^N)$
- Replace (1) with

$$\min \left\{ \frac{1}{N} \sum_{i=1}^{N} F(x, \hat{\xi}^i) : \; x \in X \right\} \tag{2}$$

  - $\hat{f}_N(x) = \frac{1}{N} \sum_{i=1}^{N} F(x, \hat{\xi}^i)$ deterministic
  - Follows mean of the N sample paths defined by the (fixed) $\hat{\xi}^i$

SAA Algorithm

Input N, (maybe $x^0 \in X$)

1. Generate $\hat{\xi}^1, \hat{\xi}^2, \cdots, \hat{\xi}^N \sim P$
2. Solve the deterministic problem

$$\min \left\{ \frac{1}{N} \sum_{i=1}^{N} F(x, \hat{\xi}^i) : \; x \in X \right\}$$

Output x_N^* (or X_N^*).

Convergence with N

- A sufficient condition:
  - For any $\epsilon > 0$ there exists N_ϵ so that

$$\left|\hat{f}_N(x) - f(x)\right| < \epsilon \qquad \forall\, N \geq N_\epsilon \quad \forall\, x \in X$$

  with probability 1 (*wp1*).
- Then $\hat{f}_N^* \to f^*$ *wp1*.
- (With additional assumptions on f and $X^* \subset X$):

$$\text{dist}(x_N^*, X^*) \to 0$$

- ($+$ uniqueness, $X^* = x^*$):

$$x_N^* \to x^*$$

Stochastic Approximation Method

Basically just:

Input x^0

1. $x^{k+1} \leftarrow \mathcal{P}_X \left\{ x^k - \alpha_k s^k \right\}, \qquad k = 0, 1, \ldots$

- α_k a step size
- s^k a random direction

Stochastic Approximation Method

Basically just:

Input x^0

1. $x^{k+1} \leftarrow \mathcal{P}_X \left\{x^k - \alpha_k s^k\right\}, \qquad k = 0, 1, \ldots$

- α_k a step size
- s^k a random direction

Generally assume:

α_k: $\sum_{k=0}^{\infty} \alpha_k = \infty$, $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$

s^k: $\mathrm{E}\left\{\nabla f(x^k)^T s^k\right\} > 0$

s^k is an ascent direction (in expectation) at x^k

Stochastic Approximation Method

Basically just:

Input x^0

1. $x^{k+1} \leftarrow \mathcal{P}_X \left\{ x^k - \alpha_k s^k \right\}, \qquad k = 0, 1, \ldots$

- α_k a step size
- s^k a random direction

Generally assume:

α_k: $\sum_{k=0}^{\infty} \alpha_k = \infty$, $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$

s^k: $\mathrm{E}\left\{ \nabla f(x^k)^T s^k \right\} > 0$

s^k is an ascent direction (in expectation) at x^k

- "Exact" Stochastic Gradient Descent: $s^k = \nabla f(x^k)$

Classic SA Algorithms

- "Original" method is Robbins-Monro (1951)
- Without derivatives: Kiefer-Wolfowitz (1952)
  replaces gradient with finite-difference approximation, e.g.,

1. $x^{k+1} \leftarrow x^k - \alpha_k s^k, \qquad k = 0, 1, \ldots$

  - where
$$s^k = \frac{F(x^k + h_k I_n; \hat{\xi}^k) - F(x^k - h_k I_n; \hat{\xi}^{k+1/2})}{2h_k}$$

Classic SA Algorithms

- "Original" method is Robbins-Monro (1951)
- Without derivatives: Kiefer-Wolfowitz (1952)
  replaces gradient with finite-difference approximation, e.g.,

1. $x^{k+1} \leftarrow x^k - \alpha_k s^k, \qquad k = 0, 1, \ldots$

  - where
  $$s^k = \frac{F(x^k + h_k I_n; \hat{\xi}^k) - F(x^k - h_k I_n; \hat{\xi}^{k+1/2})}{2h_k}$$
  - Requires $2n$ evaluations every iteration
  - Can appeal to variance reduction techniques (e.g., common RNs)
  - Convergence $x^k \to x^*$ if f strongly convex (near x^*), usual conditions on α_k, $h_k \to 0$, $\sum_k \frac{\alpha_k^2}{h_k^2} < \infty$
  - K-W recommend: $\alpha_k = \frac{1}{k}$, $h_k = \frac{1}{k^{1/3}}$

Classic SA Algorithms

- "Original" method is Robbins-Monro (1951)
- Without derivatives: Kiefer-Wolfowitz (1952)
  replaces gradient with finite-difference approximation, e.g.,

$$1.\quad x^{k+1} \leftarrow x^k - \alpha_k s^k, \qquad\qquad k = 0, 1, \ldots$$

  - where
  $$s^k = \frac{F(x^k + h_k I_n; \hat{\xi}^k) - F(x^k - h_k I_n; \hat{\xi}^{k+1/2})}{2h_k}$$
  - Requires $2n$ evaluations every iteration
  - Can appeal to variance reduction techniques (e.g., common RNs)
  - Convergence $x^k \to x^*$ if f strongly convex (near x^*), usual conditions on α_k, $h_k \to 0$, $\sum_k \frac{\alpha_k^2}{h_k^2} < \infty$
  - K-W recommend: $\alpha_k = \frac{1}{k}$, $h_k = \frac{1}{k^{1/3}}$
- Extensions such as SPSA (Spall) reduce number of evaluations (see randomized methods slides. . .)

Derivative-Based Stochastic Gradient Descent

Input x^0; Repeat:

1. Draw realization $\hat{\xi}^k \sim P$ of ξ^k
2. Compute $s^k = \nabla_x F(x^k; \hat{\xi}^k)$
3. Update $x^{k+1} \leftarrow \mathcal{P}_X \left\{ x^k - \alpha_k s^k \right\}$

Derivative-Based Stochastic Gradient Descent

Input x^0; Repeat:

1. Draw realization $\hat{\xi}^k \sim P$ of ξ^k
2. Compute $s^k = \nabla_x F(x^k; \hat{\xi}^k)$
3. Update $x^{k+1} \leftarrow \mathcal{P}_X \left\{ x^k - \alpha_k s^k \right\}$

- $\nabla_x F(x^k; \hat{\xi}^k)$ is an unbiased estimator for $\nabla f(x^k)$

Derivative-Based Stochastic Gradient Descent

Input x^0; Repeat:

1. Draw realization $\hat{\xi}^k \sim P$ of ξ^k
2. Compute $s^k = \nabla_x F(x^k; \hat{\xi}^k)$
3. Update $x^{k+1} \leftarrow \mathcal{P}_X \left\{ x^k - \alpha_k s^k \right\}$

- $\nabla_x F(x^k; \hat{\xi}^k)$ is an unbiased estimator for $\nabla f(x^k)$
- Can incorporate curvature if desired
  e.g., $B^k s^k$ an unbiased estimator for $\left(\nabla^2 f(x^k)\right)^{-1} \nabla f(x^k)$
- Can work with subgradients
- Can even output $x^N = \frac{1}{N} \sum_{k=1}^{N} x^k$

Modern Stochastic Gradient Descent Codes

Stochastic gradient descent seems inherently sequential

- Better in special cases, e.g.,

$$f(x) = \sum_{e \in \mathcal{E}} f_e(x_e), \qquad e \subset \{1, \cdots, n\}$$

$|\mathcal{E}|$ and n large

Modern Stochastic Gradient Descent Codes

Stochastic gradient descent seems inherently sequential

- Better in special cases, e.g.,

$$f(x) = \sum_{e \in \mathcal{E}} f_e(x_e), \qquad e \subset \{1, \cdots, n\}$$

  $|\mathcal{E}|$ and n large

- HOGWILD! (Niu, Recht, Ré, Wright)
  - parallel, asynchronous implementation
  - `http://i.stanford.edu/hazy/victor/Hogwild/`
  - Basic idea: Each processor samples an e uniformly from $\mathcal{E}$ and updates the coordinates x_e, ... ties broken arbitrarily

B. Randomized Algorithms for Deterministic Problems

Randomized Algorithms for Deterministic Problems

$$\min \{f(x) : x \in X \subseteq \mathbb{R}^n\}$$

- f deterministic
- Random variables are now generated by the method, *not from the problem*
- Often assume properties of f

  e.g., ∇f is L'-Lipschitz:

$$\|\nabla f(x) - \nabla f(y)\| \leq L' \|x - y\| \qquad \forall x, y \in X$$

  e.g., f is strongly convex (with parameter τ):

$$f(x) \geq f(y) + (x - y)^T \nabla f(y) + \frac{\tau}{2} \|x - y\|^2 \qquad \forall x, y \in X$$

Basic Algorithms

Matyas (e.g., 1965):

- Input x^0; repeat:
  1. Generate Gaussian u^k (centered about 0)
  2. Evaluate $f(x^k + u^k)$
  3. $x^{k+1} = \begin{cases} x^k + u^k & \text{if } f(x^k + u^k) < f(x^k) \\ x^k & \text{otherwise.} \end{cases}$

Basic Algorithms

Matyas (e.g., 1965):

- Input x^0; repeat:
  1. Generate Gaussian u^k (centered about 0)
  2. Evaluate $f(x^k + u^k)$
  3. $x^{k+1} = \begin{cases} x^k + u^k & \text{if } f(x^k + u^k) < f(x^k) \\ x^k & \text{otherwise.} \end{cases}$

Poljak (e.g., 1987)

- Input x^0, $\{h_k, \mu_k\}_k$; repeat:
  1. Generate a random $u^k \in R^n$
  2. $x^{k+1} = x^k - h_k \frac{f(x^k + \mu_k u^k) - f(x^k)}{\mu_k} u^k$

  - $h_k > 0$ is the step size
  - $\mu_k > 0$ is called the smoothing parameter

Basic Coordinate Descent Method

Componentwise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq M|h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \ldots, n$$

Basic Coordinate Descent Method

Componentwise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq M|h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \ldots, n$$

Input x^0; Repeat:

1. Choose $i_k = \arg\max_{i=1,\ldots,n} |\nabla_i f(x^k)|$
2. Update $x^{k+1} = x^k - \frac{1}{M} \nabla_{i_k} f(x^k) e_{i_k}$

Basic Coordinate Descent Method

Componentwise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq M|h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \dots, n$$

Input x^0; Repeat:

1. Choose $i_k = \arg\max_{i=1,\dots,n} |\nabla_i f(x^k)|$
2. Update $x^{k+1} = x^k - \frac{1}{M} \nabla_{i_k} f(x^k) e_{i_k}$

- Generates $f(x^k) - f^* \leq \frac{2nMR^2}{k+4}$, where $R \geq \|x^0 - x^*\|$

Basic Coordinate Descent Method

Componentwise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq M|h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \dots, n$$

Input x^0; Repeat:

1. Choose $i_k = \arg\max_{i=1,\dots,n} |\nabla_i f(x^k)|$
2. Update $x^{k+1} = x^k - \frac{1}{M} \nabla_{i_k} f(x^k) e_{i_k}$

- Generates $f(x^k) - f^* \leq \frac{2nMR^2}{k+4}$, where $R \geq \|x^0 - x^*\|$
- Good: only updates x_{i_k}
- Bad: requires entire gradient $\nabla f(x^k)$

Random Coordinate Descent Method

Component-wise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq L_i |h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \ldots, n$$

Random Coordinate Descent Method

Component-wise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \le L_i |h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \dots, n$$

Input x^0; Repeat:

1. Choose i_k uniformly at random from $\{1, \cdots, n\}$
2. Update $x^{k+1} = x^k - \frac{1}{L_i} \nabla_{i_k} f(x^k) e_{i_k}$

Random Coordinate Descent Method

Component-wise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq L_i |h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \ldots, n$$

Input x^0; Repeat:

1. Choose i_k uniformly at random from $\{1, \cdots, n\}$
2. Update $x^{k+1} = x^k - \frac{1}{L_i} \nabla_{i_k} f(x^k) e_{i_k}$

- Generates $\mathrm{E}\left\{f(x^k)\right\} - f^* \leq \frac{2nR_1^2}{k+4}$, where $R_1 = \max\{\|x - x^*\|_1 : f(x) \leq f(x_0)\}$

Random Coordinate Descent Method

Component-wise Lipschitz parameter $M > 0$:

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq L_i |h|, \qquad \forall h \in \mathbb{R}, \quad i = 1, \ldots, n$$

Input x^0; Repeat:

1. Choose i_k uniformly at random from $\{1, \cdots, n\}$
2. Update $x^{k+1} = x^k - \frac{1}{L_i} \nabla_{i_k} f(x^k) e_{i_k}$

- Generates $\mathrm{E}\left\{f(x^k)\right\} - f^* \leq \frac{2nR_1^2}{k+4}$, where $R_1 = \max\{\|x - x^*\|_1 : f(x) \leq f(x_0)\}$
- Good: only updates x_{i_k}
- Better: requires only component i_k of gradient $\nabla f(x^k)$
- Can also:
  - generate i_k proportional to coordinate Lipschitz parameters $\{L_i\}_i$
  - perform block-coordinate (and other subspace) operations

Gaussian Smoothing

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be deterministic
- $u \in \mathbb{R}^n$ from a Gaussian distribution, $\mathbb{E}_u[u] = 0$
  - Here: Covariance matrix I_n, general C OK
- For scalar $\mu > 0$, Gaussian-smoothed version of f:

$$f_\mu(x) = \mathbb{E}_u[f(x + \mu u)]$$

Gaussian Smoothing

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be deterministic
- $u \in \mathbb{R}^n$ from a Gaussian distribution, $\mathbb{E}_u[u] = 0$
  - Here: Covariance matrix I_n, general C OK
- For scalar $\mu > 0$, Gaussian-smoothed version of f:

$$f_\mu(x) = \mathbb{E}_u[f(x + \mu u)]$$

  - If f is convex, then $f_\mu(x) \geq f(x)$
  - If f is convex and ∇f is L'-Lipschitz, then

$$|f_\mu(x) - f(x)| \leq \frac{\mu^2}{2} L' n$$

Gaussian Smoothing and Directional Derivatives

$$f_\mu(x) = \mathbb{E}_u \left[f(x + \mu u) \right]$$

- Derivative of f in the direction u: $f'(x; u) = \lim_{h \downarrow 0} \frac{f(x+hu) - f(x)}{h}$

Gaussian Smoothing and Directional Derivatives

$$f_\mu(x) = \mathbb{E}_u\left[f(x + \mu u)\right]$$

- Derivative of f in the direction u: $f'(x; u) = \lim_{h \downarrow 0} \frac{f(x+hu)-f(x)}{h}$
- $g_0(x) = f'_u(x)u$
  - If f is convex, then $\mathbb{E}_u\left[g_0(x)\right]$ is a subgradient of f
  - If f is differentiable at x, then

$$\mathbb{E}_u\left[\|g_0(x)\|^2\right] \leq (n+4)\|\nabla f(x)\|^2$$

Gaussian Smoothing and Directional Derivatives

$$f_\mu(x) = \mathbb{E}_u\left[f(x+\mu u)\right]$$

- Derivative of f in the direction u: $f'(x;u) = \lim_{h\downarrow 0} \frac{f(x+hu)-f(x)}{h}$
- $g_0(x) = f'_u(x)u$
  - If f is convex, then $\mathbb{E}_u\left[g_0(x)\right]$ is a subgradient of f
  - If f is differentiable at x, then

$$\mathbb{E}_u\left[\|g_0(x)\|^2\right] \leq (n+4)\|\nabla f(x)\|^2$$

- $g_\mu(x) = \frac{f(x+\mu u)-f(x)}{\mu}u$
  - If f is differentiable at x, then $\mathbb{E}_u\left[g_\mu(x)\right] = \nabla f_\mu(x)$
  - If f is differentiable at x and ∇f is L'-Lipschitz, then

$$\mathbb{E}_u\left[\|g_\mu(x)\|^2\right] \leq 2(n+4)\|\nabla f(x)\|^2 + \frac{\mu^2}{2}L'^2(n+6)^3$$

Random Gradient Method

Input $x^0 \in X$, $\{h_k\}_k$; repeat:

1. Generate Gaussian $u^k \in R^n$ and compute $g_0(x^k) = f'_{u^k}(x^k)u^k$
2. $x^{k+1} = \mathcal{P}_X\left\{x^k - h_k g_0(x^k)\right\}$

Random Gradient Method

Input $x^0 \in X$, $\{h_k\}_k$; repeat:

1. Generate Gaussian $u^k \in R^n$ and compute $g_0(x^k) = f'_{u^k}(x^k)u^k$
2. $x^{k+1} = \mathcal{P}_X \left\{x^k - h_k g_0(x^k)\right\}$

- Key result (Nesterov) for convex (but possibly nonsmooth) f:
  For fixed $h_k = \frac{R}{\sqrt{n+4}\sqrt{N+1}L}$ and any $\epsilon > 0$,

  $$\mathbb{E}_u \left[f(\hat{x}^N)\right] - f^* \leq \epsilon, \qquad \text{where} \quad \hat{x}^N = \arg \min_{i=1,\dots,N} f(x^i)$$

  in $\mathcal{O}\left(\frac{n}{\epsilon^2}\right)$ iterations
- Also works for convex stochastic optimization and convex smooth f (with improved bounds and rates)

Random Gradient-Free Method

Input $x^0 \in X$, $\mu > 0$, $\{h_k\}_k$; repeat:

1. Generate Gaussian $u^k \in R^n$ and compute $g_\mu(x^k) = \frac{f(x^k+u^k)-f(x^k)}{\mu} u^k$
2. $x^{k+1} = \mathcal{P}_X \left\{ x^k - h_k g_\mu(x^k) \right\}$

Random Gradient-Free Method

Input $x^0 \in X$, $\mu > 0$, $\{h_k\}_k$; repeat:

1. Generate Gaussian $u^k \in R^n$ and compute $g_\mu(x^k) = \frac{f(x^k+u^k)-f(x^k)}{\mu} u^k$
2. $x^{k+1} = \mathcal{P}_X \left\{ x^k - h_k g_\mu(x^k) \right\}$

◇ Key result (Nesterov) for convex (but possibly nonsmooth) f:
For fixed $h_k = \frac{R}{(n+4)\sqrt{N+1}L}$, $\mu = \frac{\epsilon}{2L\sqrt{n}}$, and any $\epsilon > 0$,

$$\mathbb{E}_u \left[f(\hat{x}^N) \right] - f^* \leq \epsilon, \qquad \text{where} \quad \hat{x}^N = \arg \min_{i=1,\ldots,N} f(x^i)$$

in $\mathcal{O}\left(\frac{n^2}{\epsilon^2}\right)$ iterations

◇ Also works for convex stochastic optimization and convex smooth f (with improved bounds and rates)

Accelerated Random Gradient-Free Method

f strongly convex (with convexity parameter τ)

Input $v^0 = x^0$, $\mu > 0$, $\gamma_0 \geq \tau$, $\{h_k\}_k$; repeat:

1. Obtain $\alpha_k > 0$ satisfying $16(n+1)^2 L' \alpha_k^2 = (1-\alpha_k)\gamma_k + \tau\alpha_k$
2. Set $\gamma_{k+1} = (1-\alpha_k)\gamma_k + \tau\alpha_k$, $\lambda_k = \frac{\alpha_k \tau}{\gamma_{k+1}}$, $\beta_k = \frac{\alpha_k \gamma_k}{\gamma_k + \alpha_k \tau}$
3. Set $y^k = (1-\beta_k)x^k + \beta_k v^k$
4. Generate Gaussian $u^k \in R^n$ and compute $g_\mu(y^k) = \frac{f(y^k+u^k)-f(y^k)}{\mu} u^k$
5. Update

$$\begin{aligned} x^{k+1} &= y^k - \frac{1}{4(n+4)L'} g_\mu(y^k) \\ v^{k+1} &= (1-\lambda_k)v^k + \lambda_k y^k - \frac{1}{16(n+1)^2 L' \alpha_k} g_\mu(y^k) \end{aligned}$$

Accelerated Random Gradient-Free Method

f strongly convex (with convexity parameter τ)

Input $v^0 = x^0$, $\mu > 0$, $\gamma_0 \geq \tau$, $\{h_k\}_k$; repeat:

1. Obtain $\alpha_k > 0$ satisfying $16(n+1)^2 L' \alpha_k^2 = (1-\alpha_k)\gamma_k + \tau\alpha_k$
2. Set $\gamma_{k+1} = (1-\alpha_k)\gamma_k + \tau\alpha_k$, $\lambda_k = \frac{\alpha_k \tau}{\gamma_{k+1}}$, $\beta_k = \frac{\alpha_k \gamma_k}{\gamma_k + \alpha_k \tau}$
3. Set $y^k = (1-\beta_k)x^k + \beta_k v^k$
4. Generate Gaussian $u^k \in R^n$ and compute $g_\mu(y^k) = \frac{f(y^k+u^k)-f(y^k)}{\mu} u^k$
5. Update

$$\begin{aligned} x^{k+1} &= y^k - \frac{1}{4(n+4)L'} g_\mu(y^k) \\ v^{k+1} &= (1-\lambda_k)v^k + \lambda_k y^k - \frac{1}{16(n+1)^2 L' \alpha_k} g_\mu(y^k) \end{aligned}$$

◇ Key result (Nesterov): for $\tau = 0$ functions $\exists \mu > 0$ so that

$$\mathbb{E}_u\left[f(\hat{x}^N)\right] - f^* \leq \epsilon, \qquad \text{where} \quad \hat{x}^N = \arg\min_{i=1,\ldots,N} f(x^i)$$

in $\mathcal{O}\left(\frac{n}{\epsilon^{1/2}}\right)$ iterations

Applying SA-Like Ideas to Special Cases

$$\min\left\{f(x) = \frac{1}{m}\sum_{i=1}^{m} F_i(x) : x \in X\right\}$$

m huge

Applying SA-Like Ideas to Special Cases

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

m **huge**

Ex.- *Nonlinear Least Squares*
$F_i(x) = \|\phi(x;\theta^i) - d^i\|^2$
Evaluating $\phi(\cdot,\cdot)$ requires solving a large PDE

Warning: likely nonconvex!

Ex.- *Sample Average Approximation*
$F_i(x) = R(x;\hat{\xi}^i)$
$\hat{\xi}^i \in \Omega$ a scenario/RV realization
(and R depends nontrivially on $\hat{\xi}^i$)

Applying SA-Like Ideas to Special Cases

$$\min\left\{f(x) = \frac{1}{m}\sum_{i=1}^{m} F_i(x) : x \in X\right\}$$

m huge

Ex.- *Nonlinear Least Squares*
$F_i(x) = \|\phi(x;\theta^i) - d^i\|^2$
Evaluating $\phi(\cdot,\cdot)$ requires solving a large PDE

Warning: likely nonconvex!

Ex.- *Sample Average Approximation*
$F_i(x) = R(x;\hat{\xi}^i)$
$\hat{\xi}^i \in \Omega$ a scenario/RV realization
(and R depends nontrivially on $\hat{\xi}^i$)

The good:

- $\nabla f(x) = \sum_{i=1}^{m} \nabla F_i(x)$

The bad:

- m still huge

Residual Stochastic Averaging

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

"$F_i(x)$ is a member of a population of size m"

Residual Stochastic Averaging

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

"$F_i(x)$ is a member of a population of size m"

- Randomly sample $\mathcal{S}$, a subset of size $|\mathcal{S}|$, from $\{1, \cdots, m\}$

Residual Stochastic Averaging

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

"$F_i(x)$ is a member of a population of size m"

- Randomly sample $\mathcal{S}$, a subset of size $|\mathcal{S}|$, from $\{1, \cdots, m\}$
- Under minimal assumptions:

$$\mathrm{E}\left\{ \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} F_i(x) \right\} = f(x) \quad \text{and} \quad \mathrm{E}\left\{ \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla F_i(x) \right\} = \nabla f(x)$$

Residual Stochastic Averaging

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

"$F_i(x)$ is a member of a population of size m"

- Randomly sample $\mathcal{S}$, a subset of size $|\mathcal{S}|$, from $\{1, \cdots, m\}$
- Under minimal assumptions:

$$\mathrm{E}\left\{ \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} F_i(x) \right\} = f(x) \qquad \text{and} \qquad \mathrm{E}\left\{ \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla F_i(x) \right\} = \nabla f(x)$$

- Use $-\nabla f_{\mathcal{S}} = -\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \nabla F_i(x)$ as direction s^k

Residual Stochastic Averaging

$$\min \left\{ f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x) : x \in X \right\}$$

"$F_i(x)$ is a member of a population of size m"

- Randomly sample $\mathcal{S}$, a subset of size $|\mathcal{S}|$, from $\{1, \cdots, m\}$
- Under minimal assumptions:

$$\mathrm{E}\left\{ \frac{1}{|\mathcal{S}|} \sum_{i\in\mathcal{S}} F_i(x) \right\} = f(x) \quad \text{and} \quad \mathrm{E}\left\{ \frac{1}{|\mathcal{S}|} \sum_{i\in\mathcal{S}} \nabla F_i(x) \right\} = \nabla f(x)$$

- Use $-\nabla f_{\mathcal{S}} = -\frac{1}{|\mathcal{S}|} \sum_{i\in\mathcal{S}} \nabla F_i(x)$ as direction s^k
- How to choose $\mathcal{S}$?

$$\mathrm{E}\left\{ \|\nabla f_{\mathcal{S}_n} - \nabla f\|^2 \right\} = \left(1 - \frac{|\mathcal{S}|}{m}\right) \mathrm{E}\left\{ \|\nabla f_{\mathcal{S}_r} - \nabla f\|^2 \right\}$$

$\Rightarrow$ sampling *without replacement* ($\mathcal{S}_n$) gives lower variance than does sampling *with replacement* ($\mathcal{S}_r$)

Summary

◇ Methods for stochastic optimization and randomized methods for deterministic optimization closely related

+ Incredibly simple to code basic implementation
+ Well-studied complexity bounds, especially for convex cases; can show that asymptotic rates are optimal
+ Even useful when gradient/subgradient unavailable

Summary

◇ Methods for stochastic optimization and randomized methods for deterministic optimization closely related

+ Incredibly simple to code basic implementation
+ Well-studied complexity bounds, especially for convex cases; can show that asymptotic rates are optimal
+ Even useful when gradient/subgradient unavailable
- Bounds and parameters depend on characteristics of function (e.g., Lipschitz parameters, level set diameters, strong convexity)
- (Some) Practitioners remain nervous about performance deviations from the mean (active research area)