Second order machine learning

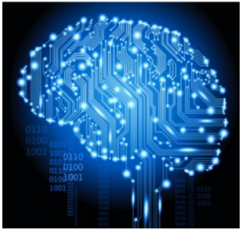
Michael W. Mahoney

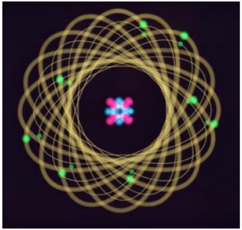
ICSI and Department of Statistics
UC Berkeley

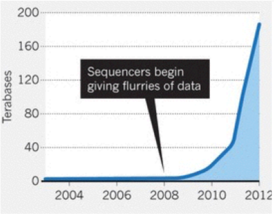
Outline

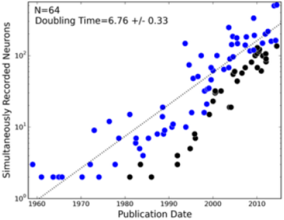
- Machine Learning's "Inverse" Problem
- Your choice:
  - 1st Order Methods: FLAG n' FLARE, or
    - disentangle geometry from sequence of iterates
  - 2nd Order Methods: Stochastic Newton-Type Methods
    - "simple" methods for convex
    - "more subtle" methods for non-convex

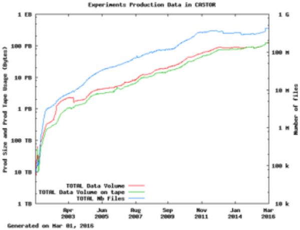
Big Data ... Massive Data ...







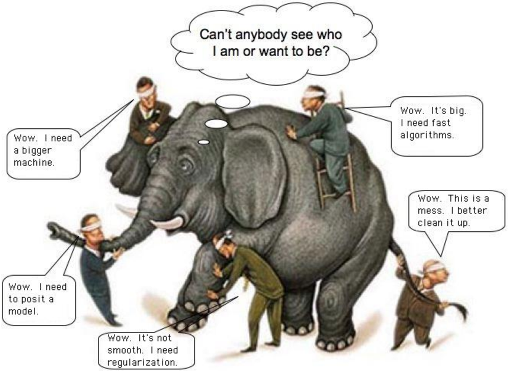




HUMONGOUS DATA ...

Big Data

How do we view BIG data?



Algorithmic & Statistical Perspectives ...

Computer Scientists

- Data: are a record of everything that happened.
- Goal: process the data to find interesting patterns and associations.
- Methodology: Develop approximation algorithms under different models of data access since the goal is typically computationally hard.

Statisticians (and Natural Scientists, etc)

- Data: are a particular random instantiation of an underlying process describing unobserved patterns in the world.
- Goal: is to extract information about the world from noisy data.
- Methodology: Make inferences (perhaps about unseen events) by positing a model that describes the random variability of the data around the deterministic model.

… are VERY different paradigms

Statistics, natural sciences, scientific computing, etc:

- Problems often involve computation, but the study of computation *per se* is secondary
- Only makes sense to develop algorithms for well-posed problems[1]
- First, write down a model, and think about computation later

Computer science:

- Easier to study computation *per se* in discrete settings, e.g., Turing machines, logic, complexity classes
- Theory of algorithms divorces computation from data
- First, run a fast algorithm, and ask what it means later

[1]Solution exists, is unique, and varies continuously with input data

Problem Statement

Problem 1: Composite Optimization Problem

$$\min_{\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d} F(\mathbf{x}) = f(\mathbf{x}) + h(\mathbf{x})$$

- f: Convex and Smooth
- h: Convex and (Non-)Smooth

Problem 2: Minimizing Finite Sum Problem

$$\min_{\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d} F(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x})$$

- f_i: (Non-)Convex and Smooth
- $n \gg 1$

Modern "Big-Data"

- Classical Optimization Algorithms
  - Effective but Inefficient

- Need to design variants, that are:
  1. Efficient, i.e., Low Per-Iteration Cost
  2. Effective, i.e., Fast Convergence Rate

Scientific Computing and Machine Learning share the same challenges,
and use the same means,
but to get to different ends!

Machine Learning has been, and continues to be, very busy designing efficient and effective optimization methods

First Order Methods

- Variants of Gradient Descent (GD):
  - Reduce the per-iteration cost of GD ⇒ Efficiency
  - Achieve the convergence rate of the GD ⇒ Effectiveness

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla F(\mathbf{x}^{(k)})$$

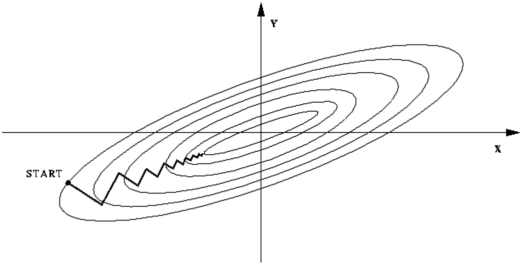
First Order Methods

- E.g.: SAG, SDCA, SVRG, Prox-SVRG, Acc-Prox-SVRG, Acc-Prox-SDCA, S2GD, mS2GD, MISO, SAGA, AMSVRG, ...

But why?

Q: Why do we use (stochastic) 1st order method?

- Cheaper Iterations? i.e., $n \gg 1$ and/or $d \gg 1$
- Avoids Over-fitting?



1st order method and "over-fitting"

Challenges with "simple" 1st order method for "over-fitting":

- Highly sensitive to ill-conditioning
- Very difficult to tune (many) hyper-parameters

"Over-fitting" is difficult with "simple" 1st order method!

Remedy?

1. "Not-So-Simple" 1st order method, e.g., accelerated *and* adaptive

2. 2nd order methods, e.g.,  methods

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - [\nabla^2 F(\mathbf{x}^{(k)})]^{-1} \nabla F(\mathbf{x}^{(k)})$$

Your Choice Of....

Which Problem?

1. "Not-So-Simple" 1st order method: FLAG n' FLARE

Problem 1: Composite Optimization Problem

$$\min_{\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d} F(\mathbf{x}) = f(\mathbf{x}) + h(\mathbf{x})$$

f: Convex and Smooth, h: Convex and (Non-)Smooth

2. 2nd order methods: Stochastic Newton-Type Methods
   - Stochastic Newton, Trust Region, Cubic Regularization

Problem 2: Minimizing Finite Sum Problem

$$\min_{\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d} F(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x})$$

f_i: (Non-)Convex and Smooth, $n \gg 1$

Collaborators

- FLAG n' FLARE
  - **Fred Roosta** (UC Berkeley)
  - Xiang Cheng (UC Berkeley)
  - Stefan Palombo (UC Berkeley)
  - Peter L. Bartlett (UC Berkeley & QUT)
- Sub-Sampled Newton-Type Methods for Convex
  - **Fred Roosta** (UC Berkeley)
  - Peng Xu (Stanford)
  - Jiyan Yang (Stanford)
  - Christopher Ré (Stanford)
- Sub-Sampled Newton-Type Methods for Non-convex
  - **Fred Roosta** (UC Berkeley)
  - Peng Xu (Stanford)
- Implementations on GPU, etc.
  - **Fred Roosta** (UC Berkeley)
  - Sudhir Kylasa (Purdue)
  - Ananth Grama (Purdue)

Subgradient Method

Composite Optimization Problem

$$\min_{\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d} F(\mathbf{x}) = f(\mathbf{x}) + h(\mathbf{x})$$

- f: Convex (Non-)Smooth
- h: Convex (Non-)Smooth

Subgradient Method

Algorithm 1 Subgradient Method

Input: $\mathbf{x}_1$, and T
for $k = 1, 2, \ldots, T-1$ **do**
  - $\mathbf{g}_k \in \partial \left(f(\mathbf{x}_k) + h(\mathbf{x}_k) \right)$
  - $\mathbf{x}_{k+1} = \arg\min_{\mathbf{x} \in \mathcal{X}} \left\{ \langle \mathbf{g}_k, \mathbf{x} \rangle + \frac{1}{2\alpha_k} \|\mathbf{x} - \mathbf{x}_k\|^2 \right\}$
end for
Output: $\bar{\mathbf{x}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{x}_t$

- α_k: Step-size
  - Constant Step-size: $\alpha_k = \alpha$
  - Diminishing Step size $\sum_{k=1}^{\infty} \alpha_k = \infty, \quad \lim_{k \to \infty} \alpha_k = 0$

Example: Logistic Regression

- $\{\mathbf{a}_i, b_i\}$: features and labels
- $\mathbf{a}_i \in \{0,1\}^d$, $b_i \in \{0,1\}$

$$F(\mathbf{x}) = \sum_{i=1}^{n} \log(1 + e^{\langle \mathbf{a}_i, \mathbf{x} \rangle}) - b_i \langle \mathbf{a}_i, \mathbf{x} \rangle$$

$$\nabla F(\mathbf{x}) = \sum_{i=1}^{n} \left(\frac{1}{1 + e^{-\langle \mathbf{a}_i, \mathbf{x} \rangle}} - b_i \right) \mathbf{a}_i$$

Infrequent Features $\Rightarrow$ Small Partial Derivative

PREDICTIVE VS. IRRELEVANT FEATURES

- Very infrequent features $\Rightarrow$ Highly predictive (e.g. "CANON" in document classification)
- Very frequent features $\Rightarrow$ Highly irrelevant (e.g. "and" in document classification)

AdaGrad [Duchi et al., 2011]

- Frequent Features $\Rightarrow$ Large Partial Derivative $\Rightarrow$ Learning Rate $\downarrow$
- Infrequent Features $\Rightarrow$ Small Partial Derivative $\Rightarrow$ Learning Rate $\uparrow$

Replace α_k with scaling matrix adaptively...

Many follows up works: RMSProp, Adam, Adadelta, etc...

AdaGrad [Duchi et al., 2011]

Algorithm 2 AdaGrad

Input: $\mathbf{x}_1$, η and T
for $k = 1, 2, \ldots, T-1$ **do**
  - $\mathbf{g}_k \in \partial f(\mathbf{x}_k)$
  - Form scaling matrix S_k based on $\{\mathbf{g}_t; t = 1, \ldots, k\}$
  - $\mathbf{x}_{k+1} = \arg\min_{\mathbf{x} \in \mathcal{X}} \left\{ \langle \mathbf{g}_k, \mathbf{x} \rangle + h(\mathbf{x}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T S_k (\mathbf{x} - \mathbf{x}_k) \right\}$
end for
Output: $\bar{\mathbf{x}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{x}_t$

Convergence

Convergence

Let $\mathbf{x}^*$ be an optimum point. We have:

- AdaGrad [Duchi et al., 2011]:

$$F(\bar{\mathbf{x}}) - F(\mathbf{x}^*) \leq \mathcal{O}\left(\frac{\sqrt{d}D_\infty \alpha}{\sqrt{T}}\right),$$

  where $\alpha \in [\frac{1}{\sqrt{d}}, 1]$ and $D_\infty = \max_{\mathbf{x},\mathbf{y}\in\mathcal{X}} \|\mathbf{y} - \mathbf{x}\|_\infty$, and

- Subgradient Descent:

$$F(\bar{\mathbf{x}}) - F(\mathbf{x}^*) \leq \mathcal{O}\left(\frac{D_2}{\sqrt{T}}\right)$$

  where $D_2 = \max_{\mathbf{x},\mathbf{y}\in\mathcal{X}} \|\mathbf{y} - \mathbf{x}\|_2$.

Comparison

Competitive Factor:

$$\frac{\sqrt{d}D_\infty \alpha}{D_2}$$

- D_∞ and D_2 depend on geometry of $\mathcal{X}$
  - e.g., $\mathcal{X} = \{\mathbf{x}; \|\mathbf{x}\|_\infty \leq 1\}$ then $D_2 = \sqrt{d}D_\infty$
- $\alpha = \frac{\sum_{i=1}^{d} \sqrt{\sum_{t=1}^{T} [\mathbf{g}_t]_i^2}}{\sqrt{d \sum_{t=1}^{T} \|\mathbf{g}_t\|^2}}$ depends on $\{\mathbf{g}_t; t = 1, \ldots, T\}$

Problem Statement

Problem 1: Composite Optimization Problem

$$\min_{\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d} F(\mathbf{x}) = f(\mathbf{x}) + h(\mathbf{x})$$

- f: Convex and Smooth (w. L-Lipschitz Gradient)
- h: Convex and (Non-)Smooth

- Subgradient Methods: $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$
- ISTA: $\mathcal{O}\left(\frac{1}{T}\right)$
- FISTA [Beck and Teboulle, 2009]: $\mathcal{O}\left(\frac{1}{T^2}\right)$

Best of both worlds?

- Accelerated Gradient Methods $\Rightarrow$ Optimal Rate
  - e.g., $\frac{1}{T^2}$ vs. $\frac{1}{T}$ vs. $\frac{1}{\sqrt{T}}$
- Adaptive Gradient Methods $\Rightarrow$ Better Constant
  - $\sqrt{d} D_\infty \alpha$ vs. D_2

How about Accelerated and Adaptive Gradient Methods?

- FLAG: Fast Linearly-Coupled Adaptive Gradient Method
- FLARE: FLAg RElaxed

FLAG [CRPBM, 2016]

Algorithm 3 FLAG

Input: $\mathbf{x}_0 = \mathbf{y}_0 = \mathbf{z}_0$ and L
for $k = 1, 2, \ldots, T$ **do**
- $\mathbf{y}_{k+1} = \mathbf{Prox}(\mathbf{x}_k)$
- Gradient Mapping $\mathbf{g}_k = -L(\mathbf{y}_{k+1} - \mathbf{x}_k)$
- Form S_k based on $\{\frac{\mathbf{g}_t}{\|\mathbf{g}_t\|}; t = 1, \ldots, k\}$
- Compute η_k
- $\mathbf{z}_{k+1} = \arg\min_{\mathbf{z} \in \mathcal{X}} \langle \eta_k \mathbf{g}_k, \mathbf{z} - \mathbf{z}_k \rangle + \frac{1}{2}(\mathbf{z} - \mathbf{z}_k)^T S_k (\mathbf{z} - \mathbf{z}_k)$
- $\mathbf{x}_k$ = Linearly Couple $(\mathbf{y}_{k+1}, \mathbf{z}_{k+1})$

end for
Output: $\mathbf{y}_{T+1}$

$$\mathbf{Prox}(\mathbf{x}_k) := \arg\min_{\mathbf{x} \in \mathcal{X}} \left\{ \langle \nabla f(\mathbf{x}_k), \mathbf{x} \rangle + h(\mathbf{x}) + \frac{L}{2} \|\mathbf{x} - \mathbf{x}_k\|_2^2 \right\}$$

FLAG Simplified

Algorithm 4 Birds Eye View of FLAG

Input: $\mathbf{x}_0$
for $k = 1, 2, \ldots, T$ **do**
- $\mathbf{y}_k$: Usual Gradient Step
- Form Gradient History
- $\mathbf{z}_k$: Scaled Gradient Step
- Find mixing wight w via Binary Search
- $\mathbf{x}_{k+1} = (1 - w)\mathbf{y}_{k+1} + w\mathbf{z}_{k+1}$

end for
Output: $\mathbf{y}_{T+1}$

Convergence

Convergence

Let $\mathbf{x}^*$ be an optimum point. We have:

- FLAG [CRPBM, 2016]:

$$F(\bar{\mathbf{x}}) - F(\mathbf{x}^*) \leq \mathcal{O}\left(\frac{dD_\infty^2 \beta}{T^2}\right),$$

  where $\beta \in [\frac{1}{d}, 1]$ and $D_\infty = \max_{\mathbf{x},\mathbf{y} \in \mathcal{X}} \|\mathbf{y} - \mathbf{x}\|_\infty$, and

- FISTA [Beck and Teboulle, 2009]:

$$F(\bar{\mathbf{x}}) - F(\mathbf{x}^*) \leq \mathcal{O}\left(\frac{D_2^2}{T^2}\right)$$

  where $D_2 = \max_{\mathbf{x},\mathbf{y} \in \mathcal{X}} \|\mathbf{y} - \mathbf{x}\|_2$.

Comparison

Competitive Factor:

$$\frac{dD_\infty^2\beta}{D_2^2}$$

- D_∞ and D_2 depend on geometry of $\mathcal{X}$
  - e.g., $\mathcal{X} = \{\mathbf{x}; \|\mathbf{x}\|_\infty \leq 1\}$ then $D_2 = \sqrt{d}D_\infty$
- $\beta = \frac{\left(\sum_{i=1}^d \sqrt{\sum_{t=1}^T [\tilde{\mathbf{g}}_t]_i^2}\right)^2}{dT}$ depends on $\{\tilde{\mathbf{g}}_t := \mathbf{g}_t/\|\mathbf{g}_t\|; t = 1, \ldots, T\}$

Linear Coupling

- Linearly Couple of $(\mathbf{y}_{k+1}, \mathbf{z}_{k+1})$ via a "ϵ-Binary Search":
- Find ϵ approximation to the root of non-linear equation

$$\langle \mathbf{Prox}\left(t\mathbf{y} + (1-t)\mathbf{z}\right) - \left(t\mathbf{y} + (1-t)\mathbf{z}\right), \mathbf{y} - \mathbf{z}\rangle = 0,$$

where

$$\mathbf{Prox}(\mathbf{x}) := \arg\min_{\mathbf{y}\in\mathcal{C}} \; h(\mathbf{y}) + \frac{L}{2}\|\mathbf{y} - \left(\mathbf{x} - \frac{1}{L}\nabla f(\mathbf{x})\right)\|_2^2.$$

- At most $\log(1/\epsilon)$ steps using bisection
- At most $2 + \log(1/\epsilon)$ **Prox** evals per-iteration more than FISTA

Can be Expensive!

Linear Coupling

- Linearly approximate:

$$\langle t\mathbf{Prox}\,(\mathbf{y}) + (1-t)\mathbf{Prox}\,(\mathbf{z}) - (t\mathbf{y} + (1-t)\mathbf{z})\,, \mathbf{y} - \mathbf{z}\rangle = 0.$$

- Linear equation in t, so closed form solution!

$$t = \frac{\langle \mathbf{z} - \mathbf{Prox}(\mathbf{z}), \mathbf{y} - \mathbf{z}\rangle}{\langle (\mathbf{z} - \mathbf{Prox}(\mathbf{z})) - (\mathbf{y} - \mathbf{Prox}(\mathbf{y})), \mathbf{y} - \mathbf{z}\rangle}$$

- At most 2 **Prox** evals per-iteration more than FISTA
- Equivalent to ϵ-Binary Search with $\epsilon = 1/3$

Better But Might Not Be Good Enough!

FLARE: FLAG RELAXED

- Basic Idea: Choose mixing weight by intelligent "futuristic" guess
  - Guess now, and next iteration, correct if guessed wrong
- FLARE: exactly the same **Prox** evals per-iteration as FISTA!
- FLARE: has the similar theoretical guarantee as FLAG!

$$\mathcal{L}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_C) = \sum_{i=1}^{n} \sum_{c=1}^{C} -\mathbf{1}(b_i = c) \log \left(\frac{e^{\langle \mathbf{a}_i, \mathbf{x}_c \rangle}}{1 + \sum_{b=1}^{C-1} e^{\langle \mathbf{a}_i, \mathbf{x}_b \rangle}} \right)$$
$$= \sum_{i=1}^{n} \left(\log \left(1 + \sum_{c=1}^{C-1} e^{\langle \mathbf{a}_i, \mathbf{x}_c \rangle} \right) - \sum_{c=1}^{C-1} \mathbf{1}(b_i = c) \langle \mathbf{a}_i, \mathbf{x}_c \rangle \right)$$

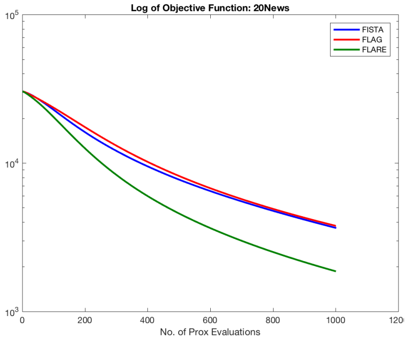
Classification: 20 Newsgroups

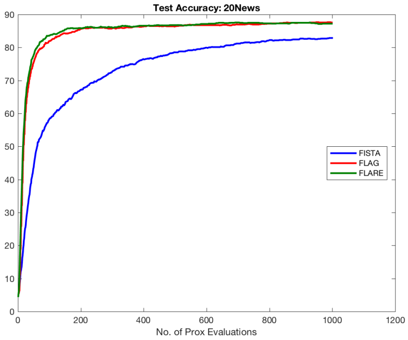
Prediction across 20 different newsgroups

Data	Train Size	Test Size	d	Classes
20 Newsgroups	10,142	1,127	53,975	20

$$\min_{\|\mathbf{x}\|_\infty \leq 1} \mathcal{L}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_C)$$

Classification: 20 Newsgroups





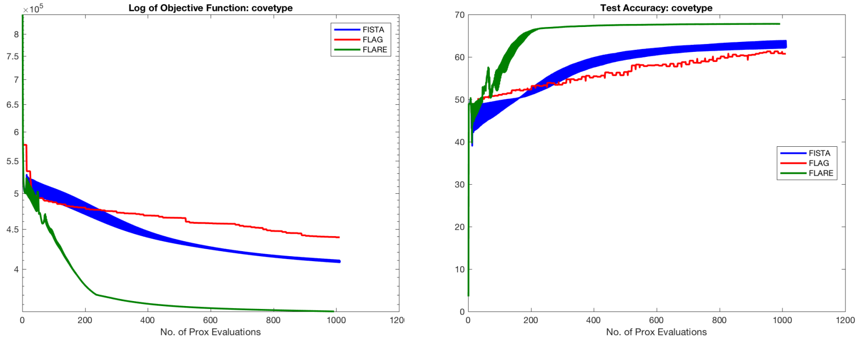
Classification: Forest CoverType

Predicting forest cover type from cartographic variables

Data	Train Size	Test Size	d	Classes
CoveType	435,759	145,253	54	7

$$\min_{\mathbf{x} \in \mathbb{R}^d} \mathcal{L}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_C) + \lambda \|\mathbf{x}\|_1$$

Classification: Forest CoverType



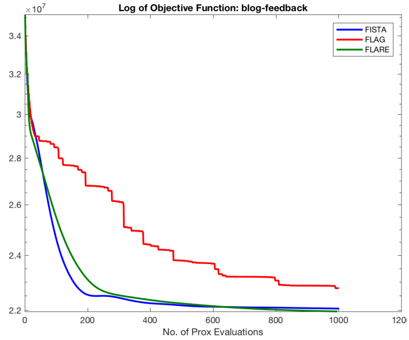
Regression: BlogFeedback

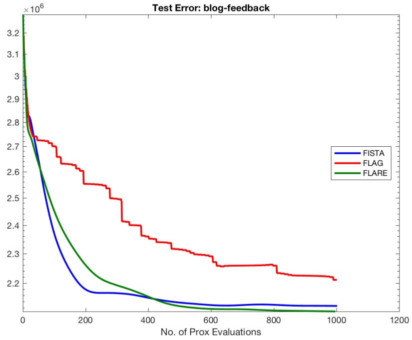
Prediction of the number of comments in the next 24 hours for blogs

Data	Train Size	Test Size	d
BlogFeedback	47,157	5,240	280

$$\min_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{2}\|A\mathbf{x}-\mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1$$

Regression: BlogFeedback





1. 2nd order methods: Stochastic Newton-Type Methods
   - Stochastic Newton (think: convex)
   - Stochastic Trust Region (think: non-convex)
   - Stochastic Cubic Regularization (think: non-convex)

Problem 2: Minimizing Finite Sum Problem

$$\min_{\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d} F(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x})$$

- f_i: (Non-)Convex and Smooth
- $n \gg 1$

Second Order Methods

- Use both gradient and Hessian information
- Fast convergence rate
- Resilient to ill-conditioning
- They "over-fit" nicely!
- However, per-iteration cost is high!

Sensorless Drive Diagnosis

$n : 50,000, p = 528,$ No. Classes $= 11, \lambda : 0.0001$

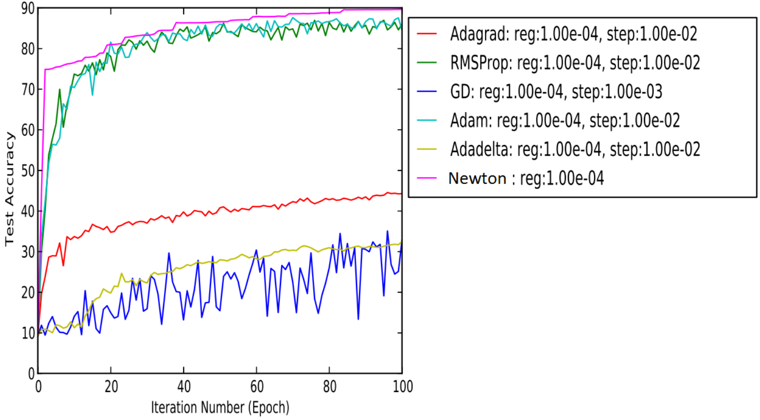


FIGURE: Test Accuracy

Sensorless Drive Diagnosis

$n : 50,000, p = 528$, No. Classes $= 11, \lambda : 0.0001$

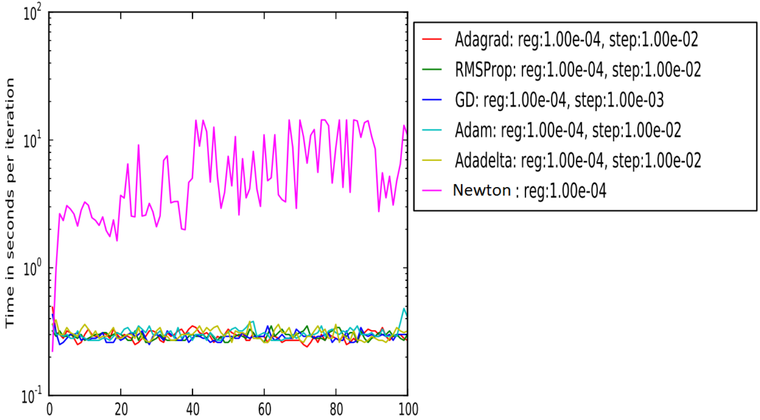


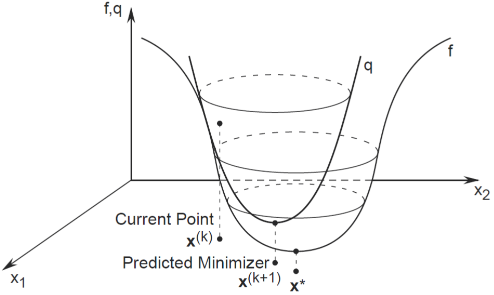
Figure: Time/Iteration

Second Order Methods

- Deterministically approximating second order information cheaply
  - Quasi-Newton, e.g., BFGS and L-BFGS [Nocedal, 1980]
- Randomly approximating second order information cheaply
  - Sub-Sampling the Hessian [Byrd et al., 2011, Erdogdu et al., 2015, Martens, 2010, RM-I, RM-II, XYRRM, 2016, Bollapragada et al., 2016, ...]
  - Sketching the Hessian [Pilanci et al., 2015]
  - Sub-Sampling the Hessian and the gradient [RM-I & RM-II, 2016, Bollapragada et al., 2016, ...]

Iterative Scheme

$$x^{(k+1)} = \arg \min_{\mathbf{x} \in \mathcal{D} \cap \mathcal{X}} \left\{ F(\mathbf{x}^{(k)}) + (\mathbf{x} - \mathbf{x}^{(k)})^T \mathbf{g}(\mathbf{x}^{(k)}) + \frac{1}{2\alpha_k} (\mathbf{x} - \mathbf{x}^{(k)})^T H(\mathbf{x}^{(k)}) (\mathbf{x} - \mathbf{x}^{(k)}) \right\}$$



Hessian Sub-Sampling

$$\mathbf{g}(\mathbf{x}) = \nabla F(\mathbf{x})$$

$$H(\mathbf{x}) = \frac{1}{|\mathcal{S}|} \sum_{j \in S} \nabla^2 f_j(\mathbf{x})$$

First, let's consider the convex case....

Convex Problems

- Each f_i is smooth and weakly convex
- F is γ-strongly convex

"We want to design methods for machine learning that are not as ideal as Newton's method but have [these] properties: first of all, they tend to turn towards the right directions and they have the right length, [i.e.,] the step size of one is going to be working most of the time...and we have to have an algorithm that scales up for machine leaning."

Prof. Jorge Nocedal
IPAM Summer School, 2012
Tutorial on Optimization Methods for ML
(Video - Part I: 50' 03")

What do we need?

- Requirements:

(R.1) **Scale up:**

(R.2) **Turn to right directions:**

(R.3) **Not ideal but close:**

(R.4) **Right step length:**

What do we need?

- Requirements:

(R.1) **Scale up:** $|\mathcal{S}|$ must be independent of n, or at least smaller than n and for $p \gg 1$, allow for inexactness

(R.2) **Turn to right directions:**

(R.3) **Not ideal but close:**

(R.4) **Right step length:**

What do we need?

- Requirements:

(R.1) **Scale up:** $|\mathcal{S}|$ must be independent of n, or at least smaller than n and for $p \gg 1$, allow for inexactness

(R.2) **Turn to right directions:** $H(\mathbf{x})$ must preserve the spectrum of $\nabla^2 F(\mathbf{x})$ as much as possible

(R.3) **Not ideal but close:**

(R.4) **Right step length:**

What do we need?

- Requirements:

(R.1) **Scale up:** $|\mathcal{S}|$ must be independent of n, or at least smaller than n and for $p \gg 1$, allow for inexactness

(R.2) **Turn to right directions:** $H(\mathbf{x})$ must preserve the spectrum of $\nabla^2 F(\mathbf{x})$ as much as possible

(R.3) **Not ideal but close:** Fast local convergence rate, close to that of Newton

(R.4) **Right step length:**

What do we need?

- Requirements:

(R.1) **Scale up:** $|\mathcal{S}|$ must be independent of n, or at least smaller than n and for $p \gg 1$, allow for inexactness

(R.2) **Turn to right directions:** $H(\mathbf{x})$ must preserve the spectrum of $\nabla^2 F(\mathbf{x})$ as much as possible

(R.3) **Not ideal but close:** Fast local convergence rate, close to that of Newton

(R.4) **Right step length:** Unit step length eventually works

Sub-sampling Hessian

- Requirements:

(R.1) **Scale up:** $|\mathcal{S}|$ must be independent of n, or at least smaller than n and for $p \gg 1$, allow for inexactness

(R.2) **Turn to right directions:** $H(\mathbf{x})$ must preserve the spectrum of $\nabla^2 F(\mathbf{x})$ as much as possible

(R.3) **Not ideal but close:** Fast local convergence rate, close to that of Newton

(R.4) **Right step length:** Unit step length eventually works

Sub-sampling Hessian

Lemma (Uniform Hessian Sub-Sampling)

Given any $0 < \epsilon < 1$, $0 < \delta < 1$ *and* $\mathbf{x} \in \mathbb{R}^p$, *if*

$$|\mathcal{S}| \geq \frac{2\kappa^2 \ln(2p/\delta)}{\epsilon^2},$$

then

$$\Pr\left((1-\epsilon)\nabla^2 F(\mathbf{x}) \preceq H(\mathbf{x}) \preceq (1+\epsilon)\nabla^2 F(\mathbf{x})\right) \geq 1-\delta.$$

Sub-sampling Hessian

- Requirements:

(R.1) **Scale up:** $|\mathcal{S}|$ must be independent of n, or at least smaller than n and for $p \gg 1$, allow for inexactness

(R.2) **Turn to right directions:** $H(\mathbf{x})$ must preserve the spectrum of $\nabla^2 F(\mathbf{x})$ as much as possible

(R.3) **Not ideal but close:** Fast local convergence rate, close to that of Newton

(R.4) **Right step length:** Unit step length eventually works

Error Recursion: Hessian Sub-Sampling

Theorem (Error Recursion)

Using $\alpha_k = 1$, with high-probability, we have

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq \rho_0 \|\mathbf{x}^{(k)} - \mathbf{x}^*\| + \xi \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2,$$

where

$$\rho_0 = \frac{\epsilon}{(1-\epsilon)}, \quad \textit{and} \quad \xi = \frac{L}{2(1-\epsilon)\gamma}.$$

- ρ_0 is problem-independent! $\Rightarrow$ Can be made arbitrarily small!

SSN-H: Q-Linear Convergence

Theorem (Q-Linear Convergence)

Consider any $0 < \rho_0 < \rho < 1$ *and* $\epsilon \leq \rho_0/(1+\rho_0)$. *If*

$$\|\mathbf{x}^{(0)} - \mathbf{x}^*\| \leq \frac{\rho - \rho_0}{\xi},$$

we get locally Q-linear convergence

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\| \leq \rho \|\mathbf{x}^{(k-1)} - \mathbf{x}^*\|, \quad k = 1, \ldots, k_0$$

with high-probability.

Possible to get superlinear rate as well.

Sub-sampling Hessian

- Requirements:

(R.1) **Scale up:** $|\mathcal{S}|$ must be independent of n, or at least smaller than n and for $p \gg 1$, allow for inexactness

(R.2) **Turn to right directions:** $H(\mathbf{x})$ must preserve the spectrum of $\nabla^2 F(\mathbf{x})$ as much as possible

(R.3) **Not ideal but close:** Fast local convergence rate, close to that of Newton

(R.4) **Right step length:** Unit step length eventually works

Sub-Sampling Hessian

Lemma (Uniform Hessian Sub-Sampling)

Given any $0 < \epsilon < 1$, $0 < \delta < 1$, *and* $\mathbf{x} \in \mathbb{R}^p$, *if*

$$|\mathcal{S}| \geq \frac{2\kappa \ln(p/\delta)}{\epsilon^2},$$

then

$$\Pr\Big((1-\epsilon)\gamma \leq \lambda_{\min}\left(H(\mathbf{x})\right)\Big) \geq 1-\delta.$$

SSN-H: Inexact Update

Assume $\mathcal{X} = \mathbb{R}^p$

$$\text{Descent Dir.:} \quad \left\{ \; \|H(\mathbf{x}^{(k)})\mathbf{p}_k + \nabla F(\mathbf{x}^{(k)})\| \leq \theta_1 \|\nabla F(\mathbf{x}^{(k)})\| \right.$$

$$\text{Step Size:} \quad \begin{cases} \alpha_k = \arg\max \; \alpha \\ \qquad\qquad \text{s.t.} \; \alpha \leq 1 \\ \qquad\qquad F(\mathbf{x}^{(k)} + \alpha \mathbf{p}_k) \leq F(\mathbf{x}^{(k)}) + \alpha\beta \mathbf{p}_k^T \nabla F(\mathbf{x}^{(k)}) \end{cases}$$

$$\text{Update:} \quad \left\{ \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}_k \right.$$

$0 < \beta, \theta_1, \theta_2 < 1$

SSN-H Algorithm: Inexact Update

Algorithm 5 Globally Convergent SSN-H with inexact solve

1: **Input:** $\mathbf{x}^{(0)}$, $0 < \delta < 1$, $0 < \epsilon < 1$, $0 < \beta, \theta_1, \theta_2 < 1$
2: - Set the sample size, $|\mathcal{S}|$, with ϵ and δ
3: **for** $k = 0, 1, 2, \cdots$ until termination **do**
4: - Select a sample set, $\mathcal{S}$, of size $|\mathcal{S}|$ and form $H(\mathbf{x}^{(k)})$
5: - Update $\mathbf{x}^{(k+1)}$ with $H(\mathbf{x}^{(k)})$ and inexact solve
6: **end for**

Gloabl Convergence SSN-H: Inexact Update

Theorem (Global Convergence of Algorithm 5)

Using Algorithm 5 with $\theta_1 \approx 1/\sqrt{\kappa}$, *with high-probability, we have*

$$F(\mathbf{x}^{(k+1)}) - F(\mathbf{x}^*) \leq (1-\rho)\big(F(\mathbf{x}^{(k)}) - F(\mathbf{x}^*)\big),$$

where $\rho = \alpha_k \beta/\kappa$ *and* $\alpha_k \geq \frac{2(1-\theta_2)(1-\beta)(1-\epsilon)}{\kappa}$.

Local + Global

Theorem

For any $\rho < 1$ and $\epsilon \approx \rho/\sqrt{\kappa}$, Algorithm 5 is globally convergent and after $\mathcal{O}(\kappa^2)$ iterations, with high-probability achieves "problem-independent" Q-linear convergence, i.e.,

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq \rho \|\mathbf{x}^{(k)} - \mathbf{x}^*\|.$$

Moreover, the step size of $\alpha_k = 1$ passes Armijo rule for all subsequent iterations.

"Any optimization algorithm for which the unit step length works has some wisdom. It is too much of a fluke if the unit step length [accidentally] works."

Prof. Jorge Nocedal
IPAM Summer School, 2012
Tutorial on Optimization Methods for ML
(Video - Part I: 56' 32")

So far these efforts mostly treated convex problems....

Now, it is time for non-convexity!

Non-Convex Is Hard!

- Saddle points, Local Minima, Local Maxima
- Optimization of a degree four polynomial: NP-hard [Hillar et al., 2013]
- Checking whether a point is not a local minimum: NP-complete [Murty et al., 1987]

All convex problems are the same,
while every non-convex problem is different.

Not sure who's quote this is!

(ϵ_g, ϵ_H) − *Optimality*

$$\|\nabla F(\mathbf{x})\| \leq \epsilon_g,$$

$$\lambda_{\min}(\nabla^2 F(\mathbf{x})) \geq -\epsilon_H$$

- Trust Region: Classical Method for Non-Convex Problem [Sorensen, 1982, Conn et al., 2000]

$$\mathbf{s}^{(k)} = \arg \min_{\|\mathbf{s}\| \leq \Delta_k} \langle \mathbf{s}, \nabla F(\mathbf{x}^{(k)}) \rangle + \frac{1}{2} \langle \mathbf{s}, \nabla^2 F(\mathbf{x}^{(k)}) \mathbf{s} \rangle$$

- Cubic Regularization: More Recent Method for Non-Convex Problem [Griewank, 1981, Nesterov et al., 2006, Cartis et al., 2011a, Cartis et al., 2011b]

$$\mathbf{s}^{(k)} = \arg \min_{\mathbf{s} \in \mathbb{R}^d} \langle \mathbf{s}, \nabla F(\mathbf{x}^{(k)}) \rangle + \frac{1}{2} \langle \mathbf{s}, \nabla^2 F(\mathbf{x}^{(k)}) \mathbf{s} \rangle + \frac{\sigma_k}{3} \|\mathbf{s}\|^3$$

- To get iteration complexity, all previous work required:

$$\left\| \left(H(\mathbf{x}^{(k)}) - \nabla^2 F(\mathbf{x}^{(k)}) \right) \mathbf{s}^{(k)} \right\| \leq C \|\mathbf{s}^{(k)}\|^2 \tag{1}$$

- Stronger than "Dennis-Moré"

$$\lim_{k \to \infty} \frac{\| \left(H(\mathbf{x}(k)) - \nabla^2 F(\mathbf{x}(k)) \right) \mathbf{s}(k) \|}{\|\mathbf{s}(k)\|} = 0$$

- We relaxed (1) to

$$\left\| \left(H(\mathbf{x}^{(k)}) - \nabla^2 F(\mathbf{x}^{(k)}) \right) \mathbf{s}^{(k)} \right\| \leq \epsilon \|\mathbf{s}^{(k)}\| \tag{2}$$

- Quasi-Newton, Sketching, Sub-Sampling satisfy Dennis-Moré and (2) but not necessarily (1)

Recall...

$$F(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} f_i(\mathbf{x})$$

Lemma (Complexity of Uniform Sampling)

Suppose $\|\nabla^2 f_i(\mathbf{x})\| \leq K$, $\forall i$. *Given any* $0 < \epsilon < 1$, $0 < \delta < 1$, *and* $\mathbf{x} \in \mathbb{R}^d$, *if*

$$|\mathcal{S}| \geq \frac{16K^2}{\epsilon^2} \log \frac{2d}{\delta},$$

then for $H(\mathbf{x}) = \frac{1}{|\mathcal{S}|} \sum_{j \in S} \nabla^2 f_j(\mathbf{x})$, *we have*

$$\Pr\left(\|H(\mathbf{x}) - \nabla^2 F(\mathbf{x})\| \leq \epsilon \right) \geq 1 - \delta.$$

- Only top eigenavlues/eigenvectors need to preserved.

$$F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{a}_i^T\mathbf{x})$$

$$p_i = \frac{|f_i''(\mathbf{a}_i^T\mathbf{x})|\|\mathbf{a}_i\|_2^2}{\sum_{j=1}^{n}|f_j''(\mathbf{a}_j^T\mathbf{x})|\|\mathbf{a}_j\|_2^2}$$

Lemma (Complexity of Non-Uniform Sampling)

Suppose $\|\nabla^2 f_i(\mathbf{x})\| \leq K_i$, $i = 1, 2, \ldots, n$. *Given any* $0 < \epsilon < 1$, $0 < \delta < 1$, *and* $\mathbf{x} \in \mathbb{R}^d$, *if*

$$|\mathcal{S}| \geq \frac{16\bar{K}^2}{\epsilon^2} \log \frac{2d}{\delta},$$

then for $H(\mathbf{x}) = \frac{1}{|\mathcal{S}|} \sum_{j \in S} \frac{1}{np_j} \nabla^2 f_j(\mathbf{x})$, *we have*

$$\Pr\left(\|H(\mathbf{x}) - \nabla^2 F(\mathbf{x})\| \leq \epsilon \right) \geq 1 - \delta,$$

where

$$\bar{K} = \frac{1}{n} \sum_{i=1}^{n} K_i.$$

Non-Convex Problems

Algorithm 6 Stochastic Trust-Region Algorithm

Input: $\mathbf{x}_0$, $\Delta_0 > 0$ $\eta \in (0,1), \gamma > 1$, $0 < \epsilon, \epsilon_g, \epsilon_H < 1$

for $k = 0, 1, 2, \cdots$ until termination **do**

$$\mathbf{s}_k \approx \arg \min_{\|\mathbf{s}\| \leq \Delta_k} m_k(\mathbf{s}) := \nabla F(\mathbf{x}_k^{(k)})^T \mathbf{s} + \frac{1}{2} \mathbf{s}^T H(\mathbf{x}^{(k)}) \mathbf{s}$$

  $\rho_k := \left(F(\mathbf{x}^{(k)} + \mathbf{s}_k) - F(\mathbf{x}^{(k)})\right) / m_k(\mathbf{s}_k)$.

  if $\rho_k \geq \eta$ **then**

    $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{s}_k$ and $\Delta_{k+1} = \gamma \Delta_k$

  else

    $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k+1)}$ and $\Delta_{k+1} = \gamma^{-1} \Delta_k$

  end if

end for

Theorem (Complexity of Stochastic TR)

If $\epsilon \in \mathcal{O}(\epsilon_H)$, then Stochastic TR terminates after

$$T \in \mathcal{O}\left(\max\{\epsilon_g^{-2}\epsilon_H^{-1}, \epsilon_H^{-3}\}\right),$$

iterations, upon which, with high probability, we have that

$$\|\nabla F(\mathbf{x})\| \leq \epsilon_g, \quad \textit{and} \quad \lambda_{\min}(\nabla^2 F(\mathbf{x})) \geq -(\epsilon + \epsilon_H).$$

- This is tight!

Non-Convex Problems

Algorithm 7 Stochastic Adaptive Regularization with Cubic Algorithm

Input: $\mathbf{x}_0$, $\Delta_0 > 0$ $\eta \in (0,1), \gamma > 1$, $0 < \epsilon, \epsilon_g, \epsilon_H < 1$
for $k = 0, 1, 2, \cdots$ until termination **do**

$$\mathbf{s}_k \approx \arg\min_{\mathbf{s}\in\mathbb{R}^d} m_k(\mathbf{s}) := \nabla F(\mathbf{x}_k^{(k)})^T \mathbf{s} + \frac{1}{2}\mathbf{s}^T H(\mathbf{x}^{(k)})\mathbf{s} + \frac{\delta_k}{3}\|\mathbf{s}\|^3$$

$\rho_k := \left(F(\mathbf{x}^{(k)} + \mathbf{s}_k) - F(\mathbf{x}^{(k)})\right)/m_k(\mathbf{s}_k)$.
if $\rho_k \geq \eta$ **then**
$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{s}_k$ and $\sigma_{k+1} = \gamma^{-1}\Delta_k$
else
$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k+1)}$ and $\sigma_{k+1} = \gamma\Delta_k$
end if
end for

Theorem (Complexity of Stochastic ARC)

If $\epsilon \in \mathcal{O}(\epsilon_g, \epsilon_H)$, then Stochastic TR terminates after

$$T \in \mathcal{O}\left(\max\{\epsilon_g^{-3/2}, \epsilon_H^{-3}\}\right),$$

iterations, upon which, with high probability, we have that

$$\|\nabla F(\mathbf{x})\| \leq \epsilon_g, \quad \textit{and} \quad \lambda_{\min}(\nabla^2 F(\mathbf{x})) \geq -(\epsilon + \epsilon_H).$$

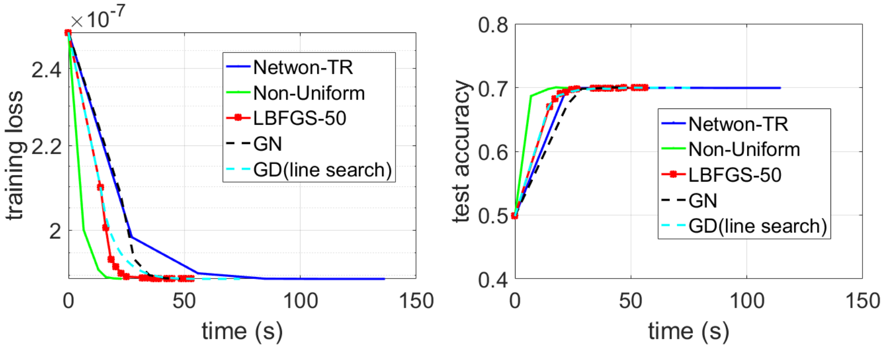
- This is tight!

- For $\epsilon_H^2 = \epsilon_g = \epsilon = \epsilon_0$
  - Stochastic TR: $T \in \mathcal{O}(\epsilon_0^{-3})$
  - Stochastic ARC: $T \in \mathcal{O}(\epsilon_0^{-3/2})$

Non-Linear Least Squares

$$\min_{\mathbf{x} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(b_i - \Phi(\mathbf{a}_i^T \mathbf{x}_i) \right)^2$$

Non-Linear Least Squares: Synthetic, $n = 1000,000$, $d = 1000$, $s = 1\%$



(a) Train Loss vs. Time

(b) Train Loss vs. Time

Conclusions: Second order machine learning

- Second order methods
  - A simple way to go beyond first order methods
  - Obviously, don't be naïve about the details
- FLAG n' FLARE
  - Combine acceleration and adaptivity to get best of both worlds
- Can aggressively sub-sample gradient and/or Hessian
  - Improve running time at each step
  - Maintain strong second-order convergence
- Apply to non-convex problems
  - Trust region methods and cubic regularization methods
  - Converge to second order stationary point