High Order Methods for Empirical Risk Minimization

Alejandro Ribeiro

Department of Electrical and Systems Engineering
University of Pennsylvania
aribeiro@seas.upenn.edu

IPAM Workshop of Emerging Wireless Systems

February 7th, 2017

- Wireless channels characterized by random fading coefficients $\mathbf{h}$
- Want to assign power $\mathbf{p}(\mathbf{h})$ as a function of fading to:
  - $\Rightarrow$ Satisfy prescribed constraints, e.g., average power, target SINRs
  - $\Rightarrow$ Optimize given criteria, e.g., maximize capacity, minimize power
- Two challenges
  - $\Rightarrow$ Resultant optimization problems are infinite dimensional ($\mathbf{h}$ is)
  - $\Rightarrow$ In most cases problems are not convex
- However, duality gap is null under mild conditions (Ribeiro-Giannakis '10)
- And in the dual domain the problem is finite dimensional and convex
- Motivate use of stochastic optimization algorithms that
  - $\Rightarrow$ Have manageable computational complexity per iteration
  - $\Rightarrow$ Use sample channel realizations instead of channel distributions

- We aim to solve expected risk minimization problem $\min_{\mathbf{w}\in\mathbb{R}^p} \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{w},\boldsymbol{\theta})]$
  - $\Rightarrow$ The distribution is unknown
  - $\Rightarrow$ We have access to N independent realizations of $\boldsymbol{\theta}$
- We settle for solving the Empirical Risk Minimization (ERM) problem

$$\min_{\mathbf{w}\in\mathbb{R}^p} F(\mathbf{w}) := \min_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{w},\boldsymbol{\theta}_i)$$

- Large-scale optimization or machine learning: large N, large p
  - $\Rightarrow$ N: number of observations (inputs)
  - $\Rightarrow$ p: number of parameters in the model
- Not just wireless
  - $\Rightarrow$ Many (most) machine learning algorithms reduce to ERM problems

- Stochastic methods: a subset of samples is used at each iteration
- SGD is the most popular; however, it is slow because of
  - $\Rightarrow$ Noise of stochasticity $\Rightarrow$ Variance reduction (SAG, SAGA, SVRG, ...)
  - $\Rightarrow$ Poor curvature approx. $\Rightarrow$ Stochastic QN (SGD-QN, RES, oLBFGS, ...)
- Decentralized methods: samples are distributed over multiple processors
  - $\Rightarrow$ Primal methods: DGD, Acc. DGD, NN, ...
  - $\Rightarrow$ Dual methods: DDA, DADMM, DQM, EXTRA, ESOM, ...
- Adaptive sample size methods: start with a subset of samples and increase the size of training set at each iteration $\Rightarrow$ Ada Newton
  - $\Rightarrow$ The solutions are close when the number of samples are close

- Objective function gradients $\Rightarrow$ $\mathbf{s}(\mathbf{w}) := \nabla F(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f(\mathbf{w}, \boldsymbol{\theta}_i)$
- (Deterministic) gradient descent iteration $\Rightarrow$ $\mathbf{w}_{t+1} = \mathbf{w}_t - \epsilon_t\, \mathbf{s}(\mathbf{w}_t)$
- Evaluation of (deterministic) gradients is not computationally affordable

- Incremental/Stochastic gradient $\Rightarrow$ Sample average in lieu of expectations

$$\hat{\mathbf{s}}(\mathbf{w}, \tilde{\boldsymbol{\theta}}) = \frac{1}{L}\sum_{l=1}^{L} \nabla f(\mathbf{w}, \boldsymbol{\theta}_l) \qquad \tilde{\theta} = [\theta_1; ...; \theta_L]$$

- Functions are chosen cyclically or at random with or without replacement

- Incremental gradient descent iteration $\Rightarrow$ $\mathbf{w}_{t+1} = \mathbf{w}_t - \epsilon_t\, \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$

- (Incremental) gradient descent is (very) slow. Newton is impractical

- Approximate function's curvature with Hessian approximation matrix $\mathbf{B}_t^{-1}$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \ \epsilon_t \ \mathbf{B}_t^{-1}\mathbf{s}(\mathbf{w}_t)$$

- Make $\mathbf{B}_t$ close to $\mathbf{H}(\mathbf{w}_t) := \nabla^2 F(\mathbf{w}_t)$. Broyden, DFP, BFGS

- Variable variation: $\mathbf{v}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$. Gradient variation: $\mathbf{r}_t = \mathbf{s}(\mathbf{w}_{t+1}) - \mathbf{s}(\mathbf{w}_t)$

- Matrix $\mathbf{B}_{t+1}$ satisfies secant condition $\mathbf{B}_{t+1}\mathbf{v}_t = \mathbf{r}_t$. Underdetermined
- Resolve indeterminacy making $\mathbf{B}_{t+1}$ closest to previous approximation $\mathbf{B}_t$

- Using Gaussian relative entropy as proximity condition yields update

$$\mathbf{B}_{t+1} = \mathbf{B}_t + \frac{\mathbf{r}_t\mathbf{r}_t^T}{\mathbf{v}_t{}^T\mathbf{r}_t} - \frac{\mathbf{B}_t\mathbf{v}_t\mathbf{v}_t{}^T\mathbf{B}_t}{\mathbf{v}_t{}^T\mathbf{B}_t\mathbf{v}_t}$$

- Superlinear convergence $\Rightarrow$ Close enough to quadratic rate of Newton
- BFGS requires gradients $\Rightarrow$ Use stochastic/incremental gradients

- Online (o)BFGS & Online Limited-Memory (oL)BFGS [Schraudolph et al '07]

- oBFGS may diverge because Hessian approximation gets close to singular
  $\Rightarrow$ Regularized Stochastic BFGS (RES) [Mokhtari-Ribeiro '14]

- oLBFGS does (surprisingly) converge [Mokhtari-Ribeiro '15]

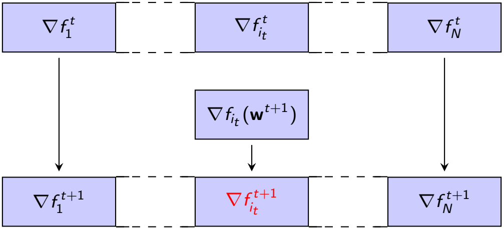
- Problem solved? Alas. RES and oLBFGS have sublinear convergence
- Same as stochastic gradient descent. Asymptotically not better

- Variance reduced stochastic L-BFGS (SVRG+oLBFGS) [Mortiz et al '16]
  $\Rightarrow$ Linear convergence rate. But this is not better than SAG, SAGA, SVRG

- Computationally feasible quasi Newton method with superlinear convergence

Incremental aggregated gradient method

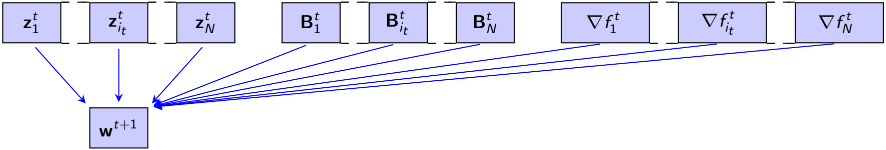
- Utilize memory to reduce variance of stochastic gradient approximation



- Descend along incremental gradient $\Rightarrow \mathbf{w}^{t+1} = \mathbf{w}^t - \dfrac{\alpha}{N}\displaystyle\sum_{i=1}^{N} \nabla f_i^t = \mathbf{w}^t - \alpha g_i^t$
- Select update index i_t cyclically. Uniformly at random is similar
- Update gradient corresponding to function f_{i_t} $\Rightarrow \nabla f_{i_t}^{t+1} = \nabla f_{i_t}(\mathbf{w}^{t+1})$
- Sum easy to compute $\Rightarrow g_i^{t+1} = g_i^t - \nabla f_{i_t}^{t+1} + \nabla f_{i_t}^{t+1}$. Converges linearly

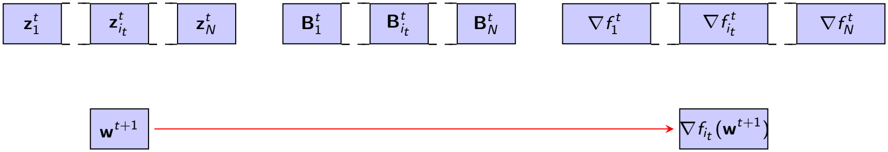
Incremental BFGS method

- Keep memory of variables $\mathbf{z}_i^t$, Hessian approximations $\mathbf{B}_i^t$, and gradients ∇f_i^t
  - $\Rightarrow$ Functions indexed by i. Time indexed by t. Select function f_{i_t} at time t



- All gradients, matrices, and variables used to update $\mathbf{w}^{t+1}$

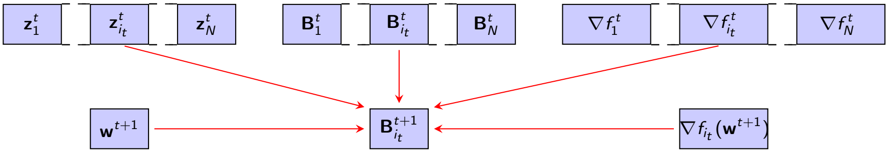
- Keep memory of variables $\mathbf{z}_i^t$, Hessian approximations $\mathbf{B}_i^t$, and gradients ∇f_i^t
  $\Rightarrow$ Functions indexed by i. Time indexed by t. Select function f_{i_t} at time t



- Updated variable $\mathbf{w}^{t+1}$ used to update gradient $\nabla f_{i_t}^{t+1} = \nabla f_{i_t}(\mathbf{w}^{t+1})$

Incremental BFGS method

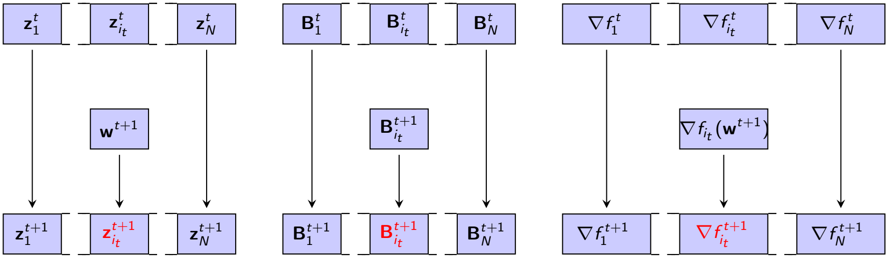
- Keep memory of variables $\mathbf{z}_i^t$, Hessian approximations $\mathbf{B}_i^t$, and gradients ∇f_i^t
  $\Rightarrow$ Functions indexed by i. Time indexed by t. Select function f_{i_t} at time t



- Update $B_{i_t}^t$ to satisfy secant condition for function f_{i_t} for variable variation $\mathbf{z}_{i_t}^t - \mathbf{w}^{t+1}$ and gradient variation $\nabla f_{i_t}^{t+1} - \nabla f_{i_t}^t$ (more later)

- Keep memory of variables $\mathbf{z}_i^t$, Hessian approximations $\mathbf{B}_i^t$, and gradients ∇f_i^t

  $\Rightarrow$ Functions indexed by i. Time indexed by t. Select function f_{i_t} at time t



- Update variable, Hessian approximation, and gradient memory for function f_{i_t}

- Variable variation at time t for function $f_i = f_{i_t}$ $\Rightarrow \mathbf{v}_i^t := \mathbf{z}_i^{t+1} - \mathbf{z}_i^t$
- Gradient variation at time t for function $f_i = f_{i_t}$ $\Rightarrow \mathbf{r}_i^t := \nabla f_{i_t}^{t+1} - \nabla f_{i_t}^t$

- Update $\mathbf{B}_i^t = \mathbf{B}_{i_t}^t$ to satisfy secant condition for variations $\mathbf{v}_i^t$ and $\mathbf{r}_i^t$

$$\mathbf{B}_i^{t+1} = \mathbf{B}_i^t + \frac{\mathbf{r}_i^t \mathbf{r}_i^{t\,T}}{\mathbf{r}_i^{t\,T} \mathbf{v}_i^t} - \frac{\mathbf{B}_i^t \mathbf{v}_i^t \mathbf{v}_i^{t\,T} \mathbf{B}_i^t}{\mathbf{v}_i^{t\,T} \mathbf{B}_i^t \mathbf{v}_i^t}$$

- We want $\mathbf{B}_i^t$ to approximate the Hessian of the function $f_i = f_{i_t}$

- The key is in the update of $\mathbf{w}^t$. Use memory in stochastic quantities

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_i^t\right)^{-1}\left(\frac{1}{N}\sum_{i=1}^{N}\nabla f_i^t\right)$$

- It doesn't work $\Rightarrow$ Better than incremental gradient but not superlinear

- Optimization updates are solutions of function approximations
- In this particular update we are minimizing the quadratic form

$$f(\mathbf{w}) \approx \frac{1}{n}\sum_{i=1}^{n}\left[f_i(\mathbf{z}_i^t) + \nabla f_i(\mathbf{z}_i^t)^T(\mathbf{w} - \mathbf{w}_t) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^t)^T\mathbf{B}_i^t(\mathbf{w} - \mathbf{w}^t)\right]$$

- Gradients evaluated at $\mathbf{z}_i^t$. Secant condition verified at $\mathbf{z}_i^t$
- The quadratic form is centered at $\mathbf{w}^t$. Not a reasonable Taylor series

- Each individual function f_i is being approximated by the quadratic

$$f_i(\mathbf{w}) \approx f_i(\mathbf{z}_i^t) + \nabla f_i(\mathbf{z}_i^t)^T(\mathbf{w} - \mathbf{w}_t) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^t)^T \mathbf{B}_i^t(\mathbf{w} - \mathbf{w}^t)$$

- To have a proper expansion we have to recenter the quadratic form at $\mathbf{z}_i^t$

$$f_i(\mathbf{w}) \approx f_i(\mathbf{z}_i^t) + \nabla f_i(\mathbf{z}_i^t)^T(\mathbf{w} - \mathbf{z}_i^t) + \frac{1}{2}(\mathbf{w} - \mathbf{z}_i^t)^T \mathbf{B}_i^t(\mathbf{w} - \mathbf{z}_i^t)$$

- I.e., we approximate $f(\mathbf{w})$ with the aggregate quadratic function

$$f(\mathbf{w}) \approx \frac{1}{N}\sum_{i=1}^{N}\left[f_i(\mathbf{z}_i^t) + \nabla f_i(\mathbf{z}_i^t)^T(\mathbf{w} - \mathbf{z}_i^t) + \frac{1}{2}(\mathbf{w} - \mathbf{z}_i^t)^T \mathbf{B}_i^t(\mathbf{w} - \mathbf{z}_i^t)\right]$$

- This is now a reasonable Taylor series that we use to derive an update

- Solving this quadratic program yields the update for the IQN method

$$\mathbf{w}^{t+1} = \left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_i^t\right)^{-1}\left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_i^t\mathbf{z}_i^t - \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(\mathbf{z}_i^t)\right]$$

- Looks difficult to implement but it is more similar to BFGS than apparent

- As in BFGS, it can be implemented with $O(p^2)$ operations

  $\Rightarrow$ Write as rank-2 update, use matrix inversion lemma

  $\Rightarrow$ Independently of N. True incremental method.

- The functions f_i are m-strongly convex.
- The gradients ∇f_i are M-Lipschitz continuous.
- The Hessians ∇f_i are L-Lipschitz continuous

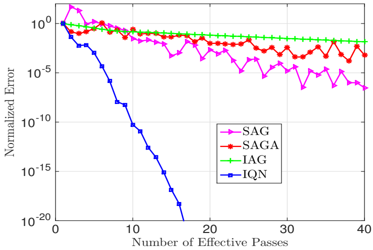
Theorem *The sequence of residuals $\|\mathbf{w}^t - \mathbf{w}^*\|$ in the IQN method converges to zero at a superlinear rate,*

$$\lim_{t \to \infty} \frac{\|\mathbf{w}^t - \mathbf{w}^*\|}{(1/N)(\|\mathbf{w}^{t-1} - \mathbf{w}^*\| + \cdots + \|\mathbf{w}^{t-N} - \mathbf{w}^*\|)} = 0.$$

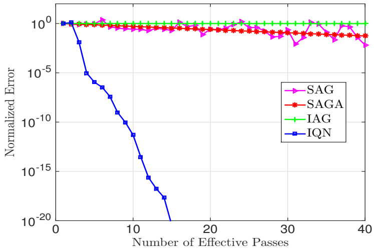
- Incremental method with small cost per iteration converging at superlinear rate
  - $\Rightarrow$ Resulting from the use of memory to reduce stochastic variances

- Quadratic programming $f(\mathbf{w}) := (1/N)\sum_{i=1}^{N} \mathbf{w}^T \mathbf{A}_i \mathbf{w}/2 + \mathbf{b}_i^T \mathbf{w}$
- $\mathbf{A}_i \in \mathbb{R}^{p \times p}$ is a diagonal positive definite matrix
- $\mathbf{b}_i \in \mathbb{R}^p$ is a random vector from the box $[0, 10^3]^p$
- $N = 1000$, $p = 100$, and condition number $(10^2, 10^4)$

- Relative error $\|\mathbf{w}^t - \mathbf{w}^*\|/\|\mathbf{w}^0 - \mathbf{w}^*\|$ of SAG, SAGA, IAG, and IQN



(a) small condition number



(b) large condition number

- Our original goal was to solve the statistical loss problem

$$\mathbf{w}^* := \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}}\, L(\mathbf{w}) = \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}}\, \mathbb{E}\left[f(\mathbf{w}, Z)\right]$$

- But since the distribution of Z is unknown we settle for the ERM problem

$$\mathbf{w}_N^\dagger := \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}}\, L_N(\mathbf{w}) = \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}}\, \frac{1}{N}\sum_{k=1}^{N} f(\mathbf{w}, z_k)$$

- Where the samples z_k are drawn from a common distribution
- ERM approximates actual problem $\Rightarrow$ Don't need perfect solution

- From statistical learning we know that there exists a constant V_N such that

$$\sup_{\mathbf{w}} |L(\mathbf{w}) - L_N(\mathbf{w})| \leq V_N, \qquad \text{w.h.p.}$$

- $V_N = O(1/\sqrt{N})$ from CLT. $V_N = O(1/N)$ sometimes [Bartlett et al '06]
- There is no need to minimize $L_N(\mathbf{w})$ beyond accuracy $\mathcal{O}(V_N)$
- This is well known. In fact, this is why we can add regularizers to ERM

$$\mathbf{w}_N^* := \operatorname*{argmin}_{\mathbf{w}} R_N(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} L_N(\mathbf{w}) + \frac{cV_N}{2}\|\mathbf{w}\|^2$$

- Adding the term $(cV_N/2)\|\mathbf{w}\|^2$ "moves" the optimum of the ERM problem
- But the optimum $\mathbf{w}_N^*$ is still in a ball of order V_N around w^*
- Goal: Minimize the risk R_N within its statistical accuracy V_N

- ERM problem R_n^* for subset of $n \leq N$ unif. chosen samples

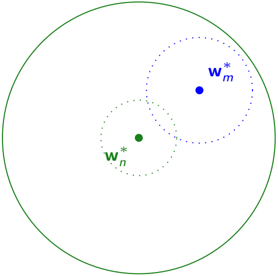
$$\mathbf{w}_n^* := \operatorname*{argmin}_{\mathbf{w}} R_n(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} L_n(\mathbf{w}) + \frac{cV_n}{2}\|\mathbf{w}\|^2$$

- Solutions $\mathbf{w}_m^*$ for m samples and $\mathbf{w}_n^*$ for n samples are close

- Find approx. solution $\mathbf{w}_m$ for the risk R_m with m samples
- Increase sample size to $n > m$ samples
- Use $\mathbf{w}_m$ as a warm start to find approx. solution w_n for R_n

- If $m < n$, it is easier to solve R_m comparing to R_n since
  - $\Rightarrow$ The condition number of R_m is smaller than R_n
  - $\Rightarrow$ The required accuracy V_m is larger than V_n
  - $\Rightarrow$ The computation cost of solving R_m is lower than R_n

- Ada Newton is a specific adaptive sample size method using Newton steps
  - $\Rightarrow$ Find w_m that solves R_m to its statistical accuracy V_m
  - $\Rightarrow$ Apply single Newton iteration $\Rightarrow \mathbf{w}_n = \mathbf{w}_m - \nabla^2 R_n(\mathbf{w}_m)^{-1} \nabla R_n(\mathbf{w}_m)$
  - $\Rightarrow$ If m and n close, we have $\mathbf{w}_n$ within statistical accuracy of R_n



- This works if statistical accuracy ball of R_m is within Newton quadratic convergence ball of R_n.
- Then, w_m is within Newton quadratic convergence ball of R_n
- A single Newton iteration yields w_n within statistical accuracy of R_n

- Question: How should we choose α?

- The functions $f(\mathbf{w}, \mathbf{z})$ are convex
- The gradients $\nabla f(\mathbf{w}, \mathbf{z})$ are M-Lipschitz continuous

$$\|\nabla f(\mathbf{w}, \mathbf{z}) - \nabla f(\mathbf{w}', \mathbf{z})\| \leq M\|\mathbf{w} - \mathbf{w}'\|, \qquad \text{for all } \mathbf{z}.$$

- The functions $f(\mathbf{w}, \mathbf{z})$ are self-concordant with respect to $\mathbf{w}$ for all $\mathbf{z}$

Theorem *Consider $\mathbf{w}_m$ as a V_m-optimal solution of R_m, i.e., $R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*) \leq V_m$, and let $n = \alpha m$. If the inequalities*

$$\left[\frac{2(M+cV_m)V_m}{cV_n}\right]^{\frac{1}{2}} + \frac{2(n-m)}{nc^{\frac{1}{2}}} + \frac{((2+\sqrt{2})c^{\frac{1}{2}} + c\|\mathbf{w}^*\|)(V_m - V_n)}{(cV_n)^{\frac{1}{2}}} \leq \frac{1}{4},$$

$$144\left[V_m + \frac{2(n-m)}{n}(V_{n-m} + V_m) + \frac{4 + c\|\mathbf{w}^*\|^2}{2}(V_m - V_n)\right]^2 \leq V_n$$

are satisfied, then $\mathbf{w}_n$ has sub-optimality error V_n w.h.p., i.e.,

$$R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq V_n, \qquad w.h.p.$$

- Condition1 $\Rightarrow$ $\mathbf{w}_m$ is in the Newton quadratic convergence ball of R_n
- Condition 2 $\Rightarrow$ $\mathbf{w}_n$ is in the statistical accuracy of R_n
- Condition 2 becomes redundant for large m

Proposition *Consider a learning problem in which the statistical accuracy satisfies $V_m \leq \alpha V_n$ for $n = \alpha m$ and $\lim_{n\to\infty} V_n = 0$. If c is chosen so that*

$$\left(\frac{2\alpha M}{c}\right)^{1/2} + \frac{2(\alpha-1)}{\alpha c^{1/2}} \leq \frac{1}{4},$$

then, there exists a sample size $\tilde{m}$ such that the conditions in Theorem 1 are satisfied for all $m > \tilde{m}$ and $n = \alpha m$.

- We can double the size of training set $\alpha = 2$
  - $\Rightarrow$ If the size of training set is large enough
  - $\Rightarrow$ If the constant c satisfies $c > 16(2\sqrt{M}+1)^2$
- We achieve the S.A. of the full training set in about 2 passes over the data
  - $\Rightarrow$ After inversion of about $3.32 \log_{10} N$ Hessians

- **Parameters:** $\alpha_0 = 2$ and $0 < \beta < 1$.
- **Initialize:** $n = m_0$ and $\mathbf{w}_n = \mathbf{w}_{m_0}$ with $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq V_n$
- **while** $n \leq N$ **do**

  Update $\mathbf{w}_m = \mathbf{w}_n$ and $m = n$. Reset factor $\alpha = \alpha_0$

  repeat [sample size backtracking loop]

  1: Increase sample size: $n = \min\{\alpha m, N\}$.

  2: Comp. gradient: $\nabla R_n(\mathbf{w}_m) = \frac{1}{n}\sum_{k=1}^{n} \nabla f(\mathbf{w}_m, z_k) + cV_n\mathbf{w}_m$

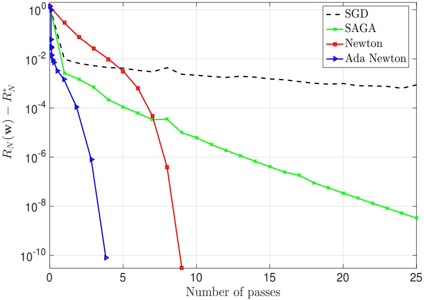
  3: Comp. Hessian: $\nabla^2 R_n(\mathbf{w}_m) = \frac{1}{n}\sum_{k=1}^{n} \nabla^2 f(\mathbf{w}_m, z_k) + cV_n\mathbf{I}$

  4: Update the variable: $\mathbf{w}_n = \mathbf{w}_m - \nabla^2 R_n(\mathbf{w}_m)^{-1}\nabla R_n(\mathbf{w}_m)$

  5: Backtrack sample size increase $\alpha = \beta\alpha$.

  until $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq V_n$
- **end while**

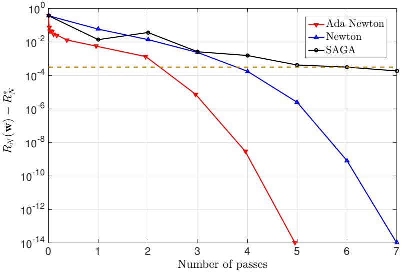
- LR problem $\Rightarrow$ Protein homology dataset provided (KDD cup 2004)
- Number of samples $N = 145,751$, dimension $p = 74$
- Parameters $\Rightarrow$ $V_n = 1/n$, $c = 20$, $m_0 = 124$, and $\alpha = 2$



- Ada Newton achieves the statistical accuracy of the full training set with about two passes over the dataset

Numerical results

- We use A9A and SUSY datasets to train a LR problem
  - $\Rightarrow$ A9A: $N = 32,561$ samples with dimension $p = 123$
  - $\Rightarrow$ SUSY: $N = 5,000,000$ samples with dimension $p = 18$
- The green line shows the iteration at which Ada Newton reached convergence on the test set



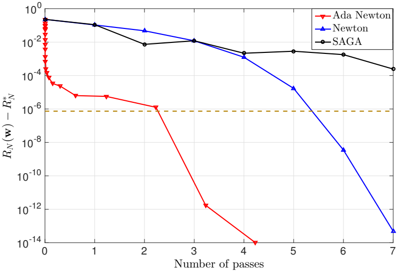
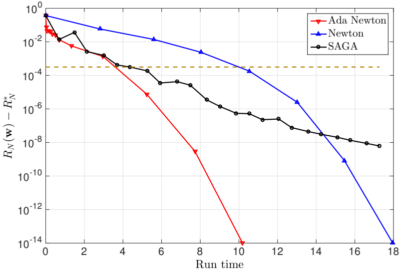


Figure: Suboptimality vs No. of effective passes. A9A (left) and SUSY (right)

- Ada Newton achieves the accuracy of $R_N(\mathbf{w}) - R_N^* < 1/N$
  - $\Rightarrow$ by less than 2.3 passes over the full training set

- We use A9A and SUSY datasets to train a LR problem
  - $\Rightarrow$ A9A: $N = 32,561$ samples with dimension $p = 123$
  - $\Rightarrow$ SUSY: $N = 5,000,000$ samples with dimension $p = 18$
- The green line shows the iteration at which Ada Newton reached convergence on the test set



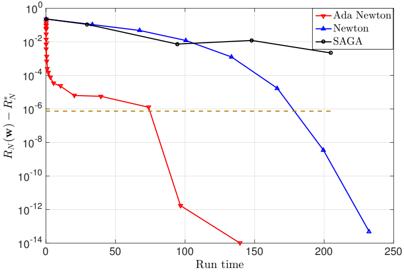


Figure: Suboptimality vs runtime. A9A (left) and SUSY (right)

- We use A9A and SUSY datasets to train a LR problem
  - $\Rightarrow$ A9A: $N = 32,561$ samples with dimension $p = 123$
  - $\Rightarrow$ SUSY: $N = 5,000,000$ samples with dimension $p = 18$

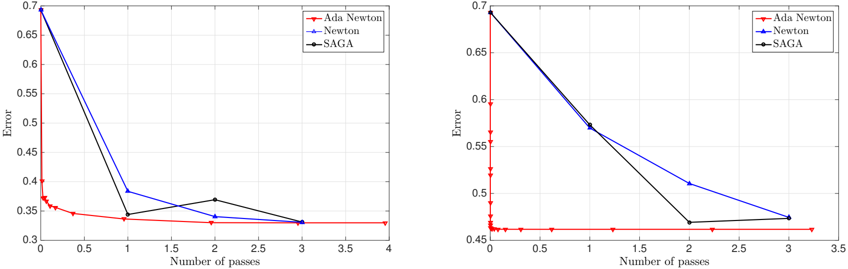


Figure: Test error vs No. of effective passes. A9A (left) and SUSY (right)

- There are four reasons why it is impractical to use Newton's method in ERM
  - It is costly to compute Hessians (and gradients). Order $O(Np^2)$ operations
  - It is costly to invert Hessians. Order $O(p^3)$ operations.
  - A line search is needed to moderate stepsize outside of quadratic region
  - Quadratic convergence is advantageous close to the optimum but we don't want to optimize beyond statistical accuracy

- Ada Newton (mostly) overcomes these four challenges
  - Compute Hessians for a subset of samples. Two passes over dataset
  - Hessians are inverted in a logarithmic number of steps. But still
  - There is no line search
  - We enter quadratic regions without going beyond statistical accuracy

Conclusions

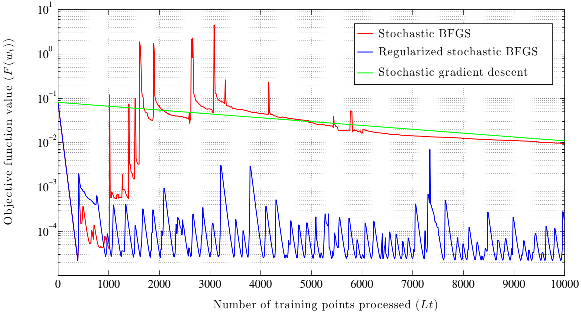
- We studied different approaches to solve large-scale ERM problems

- An incremental quasi-Newton BFGS method (IQN) was presented
- IQN only computes the information of a single function at each step
  - $\Rightarrow$ Low computation cost
- IQN aggregates variable, gradient, and BFGS approximation
  - $\Rightarrow$ Reduce the noise $\Rightarrow$ Superlinear convergence

- Ada Newton resolves the Newton-type methods drawbacks
  - $\Rightarrow$ Unit stepsize $\Rightarrow$ No line search
  - $\Rightarrow$ Not sensitive to initial point $\Rightarrow$ less Hessian inversions
  - $\Rightarrow$ Exploits quadratic convergence of Newton's method at each iteration
- Ada Newton achieves statistical accuracy with about two passes over the data

- N. N. Schraudolph, J. Yu, and S. Gunter, "A Stochastic Quasi-Newton Method for Online Convex Optimization," *In AISTATS*, vol. 7, pp. 436-443, 2007.
- A. Mokhtari and A. Ribeiro, "RES: Regularized Stochastic BFGS Algorithm," *IEEE Trans. on Signal Processing (TSP)*, vol. 62, no. 23, pp. 6089-6104, December 2014.
- A. Mokhtari and A. Ribeiro, "Global Convergence of Online Limited Memory BFGS," *Journal of Machine Learning Research (JMLR)*, vol. 16, pp. 3151-3181, 2015.
- P. Moritz, R. Nishihara, and M. I. Jordan, "A Linearly-Convergent Stochastic L-BFGS Algorithm," *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 249-258, 2016
- A. Mokhtari, M. Eisen, and A. Ribeiro, "An Incremental Quasi-Newton Method with a Local Superlinear Convergence Rate," *in Proc. Int. Conf. Acoustics Speech Signal Process. (ICASSP)*, New Orleans, LA, March 5-9 2017.
- A. Mokhtari, H. Daneshmand, A. Lucchi, T. Hofmann, and A. Ribeiro, "Adaptive Newton Method for Empirical Risk Minimization to Statistical Accuracy," In Advances in Neural Information Processing Systems (NIPS), Barcelona, Spain, 2016.

- Convergence of S-BFGS, RES, and SGD with constant stepsize $\epsilon_t = 0.1$
- Non-regularized stochastic BFGS $\Rightarrow \Gamma = 0$, $\delta = 0$
- Regularized stochastic BFGS (RES) $\Rightarrow \Gamma = 10^{-4}$, $\delta = 10^{-3}$



- **Reach convergence $\Rightarrow$ small eigenvalue $\hat{\mathbf{B}}_t$**
- RES limits the size of jumps. RES more stable than non-regularized BFGS

Making BFGS more efficient

- The BFGS update of approximate Hessian inverse

$$\mathbf{B}_t^{-1} = \left(\mathbf{I} - \frac{\mathbf{r}_{t-1}\mathbf{v}_{t-1}^T}{\mathbf{v}_{t-1}^T\mathbf{r}_{t-1}}\right)^T \mathbf{B}_{t-1}^{-1} \left(\mathbf{I} - \frac{\mathbf{r}_{t-1}\mathbf{v}_{t-1}^T}{\mathbf{v}_{t-1}^T\mathbf{r}_{t-1}}\right) + \frac{\mathbf{v}_{t-1}\mathbf{v}_{t-1}^T}{\mathbf{v}_{t-1}^T\mathbf{r}_{t-1}}$$

- Each BFGS iteration has computation complexity of order $O(p^2)$
- BFGS requires storage and propagation of the $O(p^2)$ elements of $\mathbf{B}_t^{-1}$
- This motivates alternatives with smaller memory footprints and complexity.
- $\mathbf{B}_t^{-1}$ depends on $\mathbf{B}_{t-1}^{-1}$ and the curvature information pairs $\{\mathbf{v}_{t-1}, \mathbf{r}_{t-1}\}$
- $\mathbf{B}_t^{-1}$ depends on $\mathbf{B}_0^{-1}$ and all previous curvature information $\{\mathbf{v}_u, \mathbf{r}_u\}_{u=0}^{t-1}$

  $\Rightarrow$ The old curvature information pairs should be less related

Limited memory BFGS (L-BFGS)

- LBFGS uses only the last τ past curvature information pairs $\{\mathbf{v}_u, \mathbf{r}_u\}_{u=t-\tau}^{t-1}$

- Pick the initial approximate Hessian inverse $\mathbf{B}_{t,0}^{-1} \succ \mathbf{0}$

- Update the approximate Hessian inverse $\mathbf{B}_{t,u}^{-1}$ for $u = 0, \ldots, \tau - 1$

$$\mathbf{B}_{t,u+1}^{-1} = \left(\mathbf{I} - \frac{\mathbf{r}_{t-\tau+u}\mathbf{v}_{t-\tau+u}^T}{\mathbf{v}_{t-\tau+u}^T\mathbf{r}_{t-\tau+u}} \right)^T \mathbf{B}_{t,u}^{-1} \left(\mathbf{I} - \frac{\mathbf{r}_{t-\tau+u}\mathbf{v}_{t-\tau+u}^T}{\mathbf{v}_{t-\tau+u}^T\mathbf{r}_{t-\tau+u}} \right) + \frac{\mathbf{v}_{t-\tau+u}\mathbf{v}_{t-\tau+u}^T}{\mathbf{v}_{t-\tau+u}^T\mathbf{r}_{t-\tau+u}}$$

- The outcome is the Hessian inverse at step t, i.e., $\mathbf{B}_t^{-1} = \mathbf{B}_{t,\tau}^{-1}$

- The computation complexity of implementing LBFGS is $O(\tau p) \ll O(p^2)$

- Stochastic method $\Rightarrow$ Substitute gradients with stochastic gradients

- Given the set of curvature information pairs $\{\mathbf{v}_u, \hat{\mathbf{r}}_u\}_{u=t-\tau}^{t-1}$ and variable $\mathbf{w}_t$
- Pick the initial approximate Hessian inverse $\mathbf{B}_{t,0}^{-1} \succ \mathbf{0}$
- Update the approximate Hessian inverse $\mathbf{B}_{t,u}^{-1}$ for $u = 0, \ldots, \tau - 1$

$$\hat{\mathbf{B}}_{t,u+1}^{-1} = \left(\mathbf{I} - \frac{\hat{\mathbf{r}}_{t-\tau+u}\mathbf{v}_{t-\tau+u}{}^T}{\mathbf{v}_{t-\tau+u}{}^T\hat{\mathbf{r}}_{t-\tau+u}}\right)^T \hat{\mathbf{B}}_{t,u}^{-1} \left(\mathbf{I} - \frac{\hat{\mathbf{r}}_{t-\tau+u}\mathbf{v}_{t-\tau+u}{}^T}{\mathbf{v}_{t-\tau+u}{}^T\hat{\mathbf{r}}_{t-\tau+u}}\right) + \frac{\mathbf{v}_{t-\tau+u}\mathbf{v}_{t-\tau+u}{}^T}{\mathbf{v}_{t-\tau+u}{}^T\hat{\mathbf{r}}_{t-\tau+u}}$$

- The outcome is the Hessian inverse at step t, i.e., $\hat{\mathbf{B}}_t^{-1} = \mathbf{B}_{t,\tau}^{-1}$
- oL-BFGS descent $\Rightarrow \mathbf{w}_{t+1} = \mathbf{w}_t - \epsilon_t\ \hat{\mathbf{B}}_t^{-1}\ \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$
- Variable variation $\Rightarrow \mathbf{v}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$
- Stochastic gradient variation $\Rightarrow \hat{\mathbf{r}}_t = \hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$
- No regularization is added to the stochastic version!

If $R_m(\mathbf{w}_m) - R_m(\mathbf{w}_m^*) \leq \delta$, then w.h.p.

$$R_n(\mathbf{w}_m) - R_n(\mathbf{w}_n^*) \leq \delta + \frac{2(n-m)}{n}\left(V_{n-m} + V_m\right) + 2\left(V_m - V_n\right) + \frac{c(V_m - V_n)}{2}\|\mathbf{w}^*\|^2$$

- **Require** Initial variable $\mathbf{w}_0$. Hessian approximation $\hat{\mathbf{B}}_0 \succ \delta \mathbf{I}$.
- **for** $t = 0, 1, 2, \ldots$ **do**

1: Collect L realization of random variable $\tilde{\boldsymbol{\theta}}_t = [\boldsymbol{\theta}_{t1}, \ldots, \boldsymbol{\theta}_{tL}]$

2: Compute stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) = \frac{1}{L}\sum_{l=1}^{L} \nabla f(\mathbf{w}_t, \boldsymbol{\theta}_{tl})$

3: Update the variable $\mathbf{w}_{t+1} = \mathbf{w}_t - \epsilon_t \, (\hat{\mathbf{B}}_t^{-1} + \Gamma \mathbf{I}) \, \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t)$

4: Compute variable variation $\Rightarrow \mathbf{v}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$

5: Compute stochastic gradient $\hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) = \frac{1}{L}\sum_{l=1}^{L} \nabla f(\mathbf{w}_{t+1}, \boldsymbol{\theta}_{tl})$

6: Compute modified stochastic gradient variation $\tilde{\mathbf{r}}_t$

$$\tilde{\mathbf{r}}_t = \hat{\mathbf{r}}_t - \delta \mathbf{v}_t = \hat{\mathbf{s}}(\mathbf{w}_{t+1}, \tilde{\boldsymbol{\theta}}_t) - \hat{\mathbf{s}}(\mathbf{w}_t, \tilde{\boldsymbol{\theta}}_t) - \delta \mathbf{v}_t$$

7: Hessian approximation $\Rightarrow \hat{\mathbf{B}}_{t+1} = \hat{\mathbf{B}}_t + \frac{\tilde{\mathbf{r}}_t \tilde{\mathbf{r}}_t^{\,T}}{\mathbf{v}_t^{\,T} \tilde{\mathbf{r}}_t} - \frac{\hat{\mathbf{B}}_t \mathbf{v}_t \mathbf{v}_t^{\,T} \hat{\mathbf{B}}_t}{\mathbf{v}_t^{\,T} \hat{\mathbf{B}}_t \mathbf{v}_t} + \delta \mathbf{I}.$

BFGS quasi-Newton method

- Resolve indeterminacy making $\mathbf{B}_{t+1}$ closest to previous approximation $\mathbf{B}_t$

$$\begin{aligned} \mathbf{B}_{t+1} \;=\; & \operatorname{argmin} \;\; \operatorname{tr}(\mathbf{B}_t^{-1}\mathbf{Z}) - \log\det(\mathbf{B}_t^{-1}\mathbf{Z}) - n \\ & \text{s. t.} \qquad \mathbf{Z}\,\mathbf{v}_t = \mathbf{r}_t, \quad \mathbf{Z} \succeq \mathbf{0} \end{aligned}$$

- Proximity measured in terms of differential entropy

- Solve to find Hessian approximation update

$$\mathbf{B}_{t+1} = \mathbf{B}_t + \frac{\mathbf{r}_t \mathbf{r}_t^T}{\mathbf{v}_t{}^T \mathbf{r}_t} - \frac{\mathbf{B}_t \mathbf{v}_t \mathbf{v}_t{}^T \mathbf{B}_t}{\mathbf{v}_t{}^T \mathbf{B}_t \mathbf{v}_t}$$

- For positive definite $\mathbf{B}_1 \succ \mathbf{0}$ and nonnegative variation $\mathbf{v}_t^T \mathbf{r}_t > 0$

  $\Rightarrow$ $\mathbf{B}_t$ stays positive definite for all iterations t

Condition

- The difference $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*)$ is not computable

  $\Rightarrow$ Replace it with a condition that depends on the gradient norm

$$R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq \frac{1}{2cV_n} \|\nabla R_n(\mathbf{w}_n)\|^2.$$

- Instead of $R_n(\mathbf{w}_n) - R_n(\mathbf{w}_n^*) \leq V_n$, we check the following condition

$$\|\nabla R_n(\mathbf{w}_n)\| < (\sqrt{2c})V_n$$

- Computation of the sums $\sum_{i=1}^{n} \mathbf{B}_i^t$, $\sum_{i=1}^{n} \mathbf{B}_i^t \mathbf{z}_i^t$, and $\sum_{i=1}^{n} \nabla f_i(\mathbf{z}_i^t)$
- Computing the inversion $(\sum_{i=1}^{n} \mathbf{B}_i^t)^{-1}$

- The update of IQN can be written as

$$\mathbf{w}^{t+1} = (\tilde{\mathbf{B}}^t)^{-1} \left(\mathbf{u}^t - \mathbf{g}^t\right),$$

- where $\tilde{\mathbf{B}}^t := \sum_{i=1}^{n} \mathbf{B}_i^t$ as the aggregate Hessian approximation, $\mathbf{u}^t := \sum_{i=1}^{n} \mathbf{B}_i^t \mathbf{z}_i^t$ as the aggregate Hessian-variable product, and $\mathbf{g}^t := \sum_{i=1}^{n} \nabla f_i(\mathbf{z}_i^t)$ as the aggregate gradient.

- The update for these vectors and matrices can be written as

$$\tilde{\mathbf{B}}^{t+1} = \tilde{\mathbf{B}}^t + \left(\mathbf{B}_{i_t}^{t+1} - \mathbf{B}_{i_t}^t\right)$$

$$\mathbf{u}^{t+1} = \mathbf{u}^t + \left(\mathbf{B}_{i_t}^{t+1} \mathbf{z}_{i_t}^{t+1} - \mathbf{B}_{i_t}^t \mathbf{z}_{i_t}^t\right)$$

$$\mathbf{g}^{t+1} = \mathbf{g}^t + \left(\nabla f_{i_t}(\mathbf{z}_{i_t}^{t+1}) - \nabla f_{i_t}(\mathbf{z}_{i_t}^t)\right)$$

- Thus, only $\mathbf{B}_{i_t}^{t+1}$ and $\nabla f_{i_t}(\mathbf{z}_{i_t}^{t+1})$ are required to be computed at step t.

- The inversion can be avoided by simplifying the update for $\tilde{\mathbf{B}}^t$ as

$$\tilde{\mathbf{B}}^{t+1} = \tilde{\mathbf{B}}^t + \frac{\mathbf{y}_{i_t}^t \mathbf{y}_{i_t}^{tT}}{\mathbf{y}_i^{tT} \mathbf{s}_t^{i_t}} - \frac{\mathbf{B}_{i_t}^t \mathbf{s}_{i_t}^t \mathbf{s}_{i_t}^{tT} \mathbf{B}_{i_t}^t}{\mathbf{s}_{i_t}^{tT} \mathbf{B}_{i_t}^t \mathbf{s}_{i_t}^t}.$$

- This is a rank two update.
- Given the matrix $(\tilde{\mathbf{B}}^t)^{-1}$, by applying the Sherman-Morrison formula twice to the previous update we can compute $(\tilde{\mathbf{B}}^{t+1})^{-1}$ as

$$(\tilde{\mathbf{B}}^{t+1})^{-1} = \mathbf{U}^t + \frac{\mathbf{U}^t(\mathbf{B}_{i_t}^t \mathbf{s}_{i_t}^t)(\mathbf{B}_{i_t}^t \mathbf{s}_{i_t}^t)^T \mathbf{U}^t}{\mathbf{s}_{i_t}^{t\ T} \mathbf{B}_{i_t}^t \mathbf{s}_{i_t}^t - (\mathbf{B}_{i_t}^t \mathbf{s}_{i_t}^t)^T \mathbf{U}^t (\mathbf{B}_{i_t}^t \mathbf{s}_{i_t}^t)},$$

- where the matrix $\mathbf{U}^t$ is evaluated as

$$\mathbf{U}^t = (\tilde{\mathbf{B}}^t)^{-1} - \frac{(\tilde{\mathbf{B}}^t)^{-1} \mathbf{y}_{i_t}^t \mathbf{y}_{i_t}^{tT} (\tilde{\mathbf{B}}^t)^{-1}}{\mathbf{y}_{i_t}^{tT} \mathbf{s}_{i_t}^t + \mathbf{y}_{i_t}^{tT} (\tilde{\mathbf{B}}^t)^{-1} \mathbf{y}_{i_t}^t}.$$

- The computational complexity of these updates is on the order of $\mathcal{O}(p^2)$
  $\Rightarrow$ Rather than the $\mathcal{O}(p^3)$ cost of computing the inverse directly.
- Therefore, the overall cost of IQN is on the order of $\mathcal{O}(p^2)$
  $\Rightarrow$ Substantially lower than $\mathcal{O}(np^2)$ of deter. quasi-Newton methods.