Incremental Newton Method for Minimizing Big Sums of Functions

Anton Rodomanov

Higher School of Economics

Bayesian methods research group
(http://bayesgroup.ru)

30 September 2016

Motivation: Optimization Problems in Machine Learning

Machine learning algorithms typically involve solving optimization problems of the form

$$\frac{1}{n}\sum_{i=1}^{n} f_i(x) \to \min_x$$

Example 1: Linear Regression

Given: training set $(a_i, \beta_i)_{i=1}^n$, $a_i \in \mathbb{R}^d, \beta_i \in \mathbb{R}$.
Goal: predict β_{new} for a new observation a_{new}.

- Let us use a linear family of predictors:
$$\hat{\beta}(a; x) := \langle a, x \rangle = \sum_{i=1}^d a_i x_i$$
where $x \in \mathbb{R}^d$ are the parameters of $\hat{\beta}$.
- Find x^* minimizing the average error of $\hat{\beta}$ on the training set:
$$\frac{1}{n} \sum_{i=1}^n (\langle a_i, x \rangle - \beta_i)^2 \to \min_x$$
- Predict new labels: $\beta_{\text{new}} = \hat{\beta}(a_{\text{new}}; x^*) = \langle a_{\text{new}}, x^* \rangle$

Example 2: Logistic Regression

Given: training set $(a_i, \beta_i)_{i=1}^n$, $a_i \in \mathbb{R}^d, \beta_i \in \{-1, 1\}$.
Goal: predict β_{new} for a new observation a_{new}.

- Let us use the following family of predictors:
$$\hat{\beta}(a; x) := \operatorname{sign}(\langle a, x \rangle)$$
where $x \in \mathbb{R}^d$ are the parameters of $\hat{\beta}$.
- Find x^* minimizing the average error of $\hat{\beta}$ on the training set:
$$\frac{1}{n} \sum_{i=1}^n \ln(1 + \exp(-\beta_i \langle a_i, x \rangle)) \to \min_x$$
- Predict new labels: $\beta_{\mathrm{new}} = \hat{\beta}(a_{\mathrm{new}}; x^*) = \operatorname{sign}(\langle a_{\mathrm{new}}, x^* \rangle)$

Problem

- So we are interested in efficient methods for solving
$$\frac{1}{n}\sum_{i=1}^{n} f_i(x) \to \min_{x\in\mathbb{R}^d}$$
- Let us first consider the general problem
$$f(x) \to \min_{x\in\mathbb{R}^d}$$
and two standard methods for solving it: Gradient Descent and Newton Method.

Preliminaries

Problem: $f(x) \to \min\limits_{x \in \mathbb{R}^d}$.

Assumptions:

- Function f is sufficiently *smooth*, i.e. once or twice continuously differentiable.
- Function f is *strongly convex*.

Smoothness of f enables us to compute the gradient $\nabla f(x)$ and Hessian $\nabla^2 f(x)$.
Strong convexity implies existence and uniqueness of solution x^*.
We consider iterative methods which produce $\{x^k\}_{k \geq 0} : x^k \to x^*$.

Gradient descent

Problem: $f(x) \to \min\limits_{x \in \mathbb{R}^d}$.

Idea: Choose direction p^k which locally minimizes f the most:

- For any fixed $p \in \mathbb{R}^d$ consider directional derivative
$$Df(x)[p] = \lim_{t \to 0+} \frac{f(x+tp) - f(x)}{t} = \langle \nabla f(x), p \rangle$$
- Note that, by Cauchy-Schwarz inequality, for $\|p\| = 1$,
$$\langle \nabla f(x), p \rangle \geq -\|\nabla f(x)\| \|p\| = -\|\nabla f(x)\|$$
with equality iff $p = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$.
- Thus, $-\nabla f(x^k)$ is the direction of the steepest descent at x^k.

Gradient descent:
$$x^{k+1} = x^k - \gamma_k \nabla f(x^k)$$
Here $\gamma_k \in \mathbb{R}_{++}$ is a (properly chosen) step length.

Newton Method

Problem: $f(x) \to \min_{x \in \mathbb{R}^d}$.

Motivation:

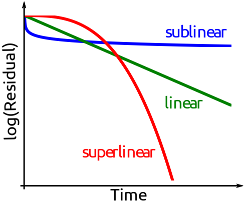
- Approximate f with a quadratic:
$$f(x) \approx f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + \frac{1}{2} \langle \nabla^2 f(x^k)(x - x^k), x - x^k \rangle$$
- Choose x^{k+1} as the minimizer of the quadratic approximation:
$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}^d} \left\{ \langle \nabla f(x^k), x - x^k \rangle + \frac{1}{2} \langle \nabla^2 f(x^k)(x - x^k), x - x^k \rangle \right\}$$
- This minimizer can be calculated in closed form, giving us

Newton method:
$$x^{k+1} = x^k - [\nabla^2 f(x^k)]^{-1} \nabla f(x^k).$$

Comparison of Gradient Descent and Newton Method

Convergence rates: $r_k := f(x^k) - f^*$

- Linear: $r_{k+1} \le c r_k, \; c \in (0,1)$
- Sublinear: $r_{k+1} \le c_k r_k, \; c_k \uparrow 1$
- Superlinear: $r_{k+1} \le c_k r_k, \; c_k \downarrow 0.$



Gradient Descent:

+ Cheap iterations: only vector operations are involved
+ Low memory requirements: stores only vectors $x^k, \nabla f(x^k)$
+ Convergence for any x^0
− Linear convergence rate

Newton Method:

− Expensive iterations: requires matrix inversion
− High memory requirements: stores matrix $\nabla^2 f(x^k)$
− Convergence guaranteed only for x^0 close enough to x^*
+ Very fast superlinear convergence rate (in fact, quadratic)

Nevertheless, for small d, Newton method is very effective.

Getting Back to Original Problem

Problem: $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) \to \min_{x\in\mathbb{R}^d}$

Gradient descent:

$$x^{k+1} = x^k - \gamma_k \nabla f(x^k)$$

$$\nabla f(x^k) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x^k)$$

Note:

- Computation of $\nabla f(x^k)$ usually requires $O(nd)$ operations.
- When n is very large, this may take a lot of time. Example: $n = 10^8$, $d = 1000 \Rightarrow$ evaluating $\nabla f(x^k)$ takes ≈ 2 minutes.
- The situation is the same for Newton method.
- We need methods that do not evaluate all the n components at every iteration.

Stochastic Gradient Method

Problem: $f(x) \to \min_{x \in \mathbb{R}^d}, \quad f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x).$

Stochastic Gradient Method (SGD):

$$\text{Choose } i_k \in \{1, \ldots, n\} \text{ uniformly at random}$$
$$x^{k+1} = x^k - \gamma_k \nabla f_{i_k}(x^k).$$

Here $\gamma_k \in \mathbb{R}_{++}$ is a (properly chosen) step length.

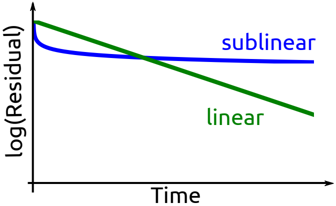
Motivation: $\mathbb{E}_{i_k}[\nabla f_{i_k}(x^k)] = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x^k) = \nabla f(x^k)$, i.e., on average, SGD makes a step in the right direction.

Note:

- Now we only need to compute one gradient instead of n.
- Iteration complexity: $O(d)$. Independent of n!

Comparison of Gradient Descent and SGD

Problem: $f(x) \to \min_{x \in \mathbb{R}^d}$, $\quad f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$.



Iteration cost:

- Gradient descent: $O(nd)$.
- SGD: $O(d)$.

Convergence rate:

- Gradient descent: *linear*.
- SGD: *sublinear*.

Natural question: Can we get the best from each of them?
Is there a method with $O(d)$ iteration cost and linear rate?

Stochastic Average Gradient [Le Roux et al., 2012]

Problem: $f(x) \to \min_{x \in \mathbb{R}^d}, \quad f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$

Stochastic Average Gradient (SAG):

Choose $i_k \in \{1, \dots, n\}$ uniformly at random

$$\text{Update} \quad y_i^k = \begin{cases} \nabla f_i(x^k) & \text{if } i = i_k \\ y_i^{k-1} & \text{otherwise} \end{cases}$$

$$x^{k+1} = x^k - \gamma_k g^k, \quad \text{where } g^k = \frac{1}{n}\sum_{i=1}^{n} y_i^k.$$

Here $\gamma_k \in \mathbb{R}_{++}$ is a (properly chosen) step length.
Note that $g^k = g^{k-1} + \frac{1}{n}(y_i^k - y_i^{k-1})$ where $i = i_k$.

Discussion:

- Iteration cost: $O(d)$ if y_i^k and g_k are stored in memory.
- Convergence rate: linear.

A quick survey of stochastic optimization methods

Problem: $f(x) \to \min_{x \in \mathbb{R}^d}, \quad f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x).$

Consider: methods whose iteration cost is independent of n.

Two groups of stochastic methods:

- SGD alike: $x^{k+1} = x^k - \gamma_k B^k \nabla f_{i_k}(x^k)$.
  - SGD, oLBFGS [Schraudolph et al., 2007], AdaGrad [Duchi et al., 2011], SQN [Byrd et al., 2014], Adam [Kingma, 2014] etc.
  - Convergence rate: sublinear.
- Variance reducing.
  - IAG [Blatt et al., 2007], SAG [Schmidt et al., 2013], SVRG [Johnson & Zhang, 2013], FINITO [Defazio et al., 2014b], SAGA [Defazio et al., 2014a], MISO [Mairal, 2015] etc.
  - Convergence rate: linear.

Note: no stochastic methods with superlinear convergence.

Incremental Newton Method – 1 [Rodomanov & Kropotov, 2016]

Problem: $f(x) \to \min_{x\in\mathbb{R}^d}, \quad f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$

Idea:

- Build the second-order Taylor approximation of each f_i:
  $m_i^k(x) := f_i(v_i^k) + \langle \nabla f_i(v_i^k), x - v_i^k\rangle + \frac{1}{2}\langle \nabla^2 f_i(v_i^k)(x - v_i^k), x - v_i^k\rangle.$
- Then f can be approximated with $m^k(x) := \frac{1}{n}\sum_{i=1}^{n} m_i^k(x)$.
- Choose the next iterate x^{k+1} as the minimum of m^k:
  $$x^{k+1} = \operatorname*{argmin}_{x\in\mathbb{R}^d} m^k(x).$$
- Update only one v_i^k at every iteration to keep the iteration cost independent of n: choose $i_k \in \{1, \dots, n\}$ and set
  $$v_i^k = \begin{cases} x^k & \text{if } i = i_k, \\ v_i^{k-1} & \text{otherwise.} \end{cases}$$

Incremental Newton Method – 2 [Rodomanov & Kropotov, 2016]

Model of the objective:

$$m^k(x) = \frac{1}{n}\sum_{i=1}^{n}[f_i(v_i^k)+\langle\nabla f_i(v_i^k), x-v_i^k\rangle+\frac{1}{2}\langle\nabla^2 f_i(v_i^k)(x-v_i^k), x-v_i^k\rangle]$$

Note: m^k is a quadratic,

$$m^k(x) = \frac{1}{2}\langle H^k x, x\rangle + \langle g^k - u^k, x\rangle + \mathrm{const},$$

and determined only by the following three quantities:

$$H^k := \frac{1}{n}\sum_{i=1}^{n}\nabla^2 f_i(v_i^k), \;\; g^k := \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(v_i^k), \;\; u^k := \frac{1}{n}\sum_{i=1}^{n}\nabla^2 f_i(v_i^k) v_i^k.$$

Since only one component is updated at every iteration,

$$H^k = H^{k-1} + \frac{1}{n}\left[\nabla^2 f_i(v_i^k) - \nabla^2 f_i(v_i^{k-1})\right],$$

$$g^k = g^{k-1} + \frac{1}{n}\left[\nabla f_i(v_i^k) - \nabla f_i(v_i^{k-1})\right],$$

$$u^k = u^{k-1} + \frac{1}{n}\left[\nabla^2 f_i(v_i^k) v_i^k - \nabla^2 f_i(v_i^{k-1}) v_i^{k-1}\right].$$

Incremental Newton Method – 3 [Rodomanov & Kropotov, 2016]

Problem: $f(x) \to \min_{x \in \mathbb{R}^d}, \quad f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$

Incremental Newton Method (NIM):

Take $i_k = k \bmod n + 1$

Update $v_i^k = \begin{cases} x^k & \text{if } i = i_k \\ v_i^{k-1} & \text{otherwise} \end{cases}$

$$H^k = H^{k-1} + \frac{1}{n}\left[\nabla^2 f_i(v_i^k) - \nabla^2 f_i(v_i^{k-1})\right],$$

$$g^k = g^{k-1} + \frac{1}{n}\left[\nabla f_i(v_i^k) - \nabla f_i(v_i^{k-1})\right],$$

$$u^k = u^{k-1} + \frac{1}{n}\left[\nabla^2 f_i(v_i^k) v_i^k - \nabla^2 f_i(v_i^{k-1}) v_i^{k-1}\right]$$

Compute $x^{k+1} = (H^k)^{-1}(u_k - g_k)$.

Note: Iteration cost is independent of n if v_i^k are kept in memory.

Superlinear convergence rate of NIM

Problem: $f(x) \to \min_{x \in \mathbb{R}^d}, \quad f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x).$

Theorem: Suppose the Hessians $\nabla^2 f_i$ are Lipschitz-continuous:

$$\left\| \nabla^2 f_i(x) - \nabla^2 f_i(y) \right\| \leq M \left\| x - y \right\|, \qquad \forall x, y \in \mathbb{R}^d.$$

Assume x^* is a minimizer of f with positive definite Hessian:

$$\nabla^2 f(x^*) = \frac{1}{n} \sum_{i=1}^{n} \nabla^2 f_i(x^*) \succeq \mu I, \qquad \mu > 0,$$

and all the initial points $x^0, \ldots, x^{n-1}$ are close enough to x^*:

$$\left\| x^i - x^* \right\| \leq \frac{\mu}{2M}.$$

Then the sequence of iterates $\{x^k\}_{k \geq n}$ of NIM converges to x^* at an R-superlinear rate, i.e. there exists $\{z_k\}_{k \geq 0}$ such that

$$\left\| x^k - x^* \right\| \leq z_k, \qquad z_{k+1} \leq \left(1 - \frac{3}{4n}\right)^{2^{\lceil k/n \rceil - 1}} z_k.$$

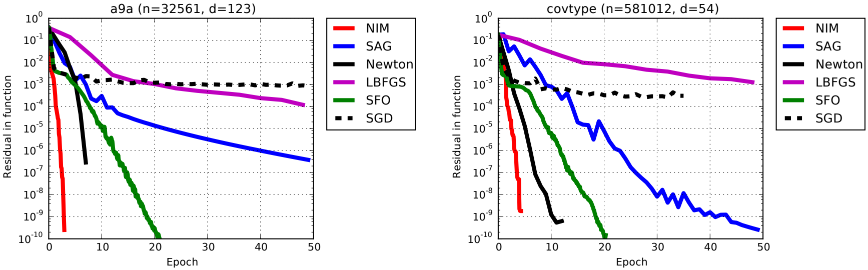
More precisely, the convergence rate is n-step quadratic:

$$z_{k+n} \leq \frac{M}{\mu} z_k^2.$$

Evaluation results – 1

***L*2-regularized logistic regression:**

$$f(x) := \frac{1}{n}\sum_{i=1}^{n} \ln(1+\exp(-\beta_i\langle a_i, x\rangle)) + \frac{\mu}{2}\|x\|^2 .$$



Evaluations results – 2

	a9a (n=32561, d=123)					*covtype (n=581012, d=54)*				
Res	NIM	SAG	Newton	LBFGS	SFO	NIM	SAG	Newton	LBFGS	SFO
10^{-1}	**.01s**	.01s	.31s	.05s	.03s	.19s	.33s	.84s	.54s	**.04s**
10^{-2}	**.02s**	.05s	.56s	.10s	.08s	.51s	.96s	1.78s	1.77s	**.25s**
10^{-3}	.12s	**.11s**	.73s	.18s	.57s	**.72s**	1.58s	2.39s	5.67s	1.02s
10^{-4}	**.15s**	.19s	.81s	.43s	.98s	**.86s**	2.45s	3.09s	10.73s	3.80s
10^{-5}	**.21s**	.36s	.90s	.76s	1.34s	**1.20s**	3.37s	3.99s	19.07s	5.23s
10^{-6}	**.24s**	.66s	.93s	1.11s	1.57s	**1.49s**	4.12s	4.57s	31.84s	6.81s
10^{-7}	**.28s**	1.04s	1.00s	1.45s	1.93s	**1.69s**	4.69s	5.13s	-	8.23s
10^{-8}	**.31s**	1.46s	1.04s	1.82s	2.18s	**1.92s**	5.90s	6.52s	-	9.86s
10^{-9}	**.32s**	1.90s	1.04s	2.26s	2.46s	**2.10s**	7.34s	7.64s	-	11.30s
10^{-10}	**.34s**	2.38s	1.04s	2.61s	2.81s	**2.12s**	9.97s	8.84s	-	12.44s

Evaluations results – 3

	alpha (n=500000, d=500)				*mnist8m (n=8100000, d=784)*			
Res	NIM	SAG	Newton	LBFGS	NIM	SAG	Newton	LBFGS
10^{-1}	1.91s	**1.36s**	1.6m	4.01s	57.68s	**34.91s**	47.8m	1.1m
10^{-2}	13.37s	**6.72s**	2.6m	17.68s	**1.6m**	2.1m	1.4h	5.2m
10^{-3}	28.56s	**17.73s**	3.0m	37.70s	**3.2m**	3.9m	-	22.9m
10^{-4}	36.65s	**36.04s**	3.4m	58.35s	16.7m	**7.1m**	-	1.6h
10^{-5}	**46.66s**	1.0m	3.6m	1.4m	**26.7m**	1.0h	-	-
10^{-6}	**53.92s**	1.5m	4.0m	1.9m	**33.5m**	-	-	-
10^{-7}	**57.63s**	2.0m	4.0m	2.4m	**40.1m**	-	-	-
10^{-8}	**1.0m**	2.7m	4.1m	2.8m	**46.0m**	-	-	-
10^{-9}	**1.1m**	3.5m	4.3m	3.2m	**49.6m**	-	-	-
10^{-10}	**1.2m**	4.3m	4.7m	3.4m	**53.3m**	-	-	-

Conclusions

- The presented incremental Newton method is the first stochastic method with a superlinear convergence rate.
- The method can be thought of as a generalization of the classic Newton method to the special case of big sums.
- It has the same advantages and disadvantages as the classic Newton method:
  - \+ Fast superlinear convergence rate.
  - − Only local convergence is guaranteed.
  - − Not applicable to high-dimensional problems.
- For details, see paper

  A. Rodomanov, D. Kropotov. A Superlinearly-Convergent Proximal Newton-type Method for the Optimization of Finite Sums, ICML 2016.

Thank you!