Randomized Algorithms

Zhifeng Wang

Department of Statistics
Florida State University

Outline

- (Block) Coordinate Descent
  - Selecting Coordinates
  - Rates of Convergence
- Stochastic Gradient Descent
  - Stochastic Optimization
  - Empirical Risk Minimization
  - Momentum and Adaptive Learning Rates
  - Incremental Gradient Methods

Coordinate Descent

- Our goal is to solve

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p} f(\boldsymbol{\beta}),$$

  where f is convex and smooth (f is continuously differentiable and gradient is Lipschitz continuous).

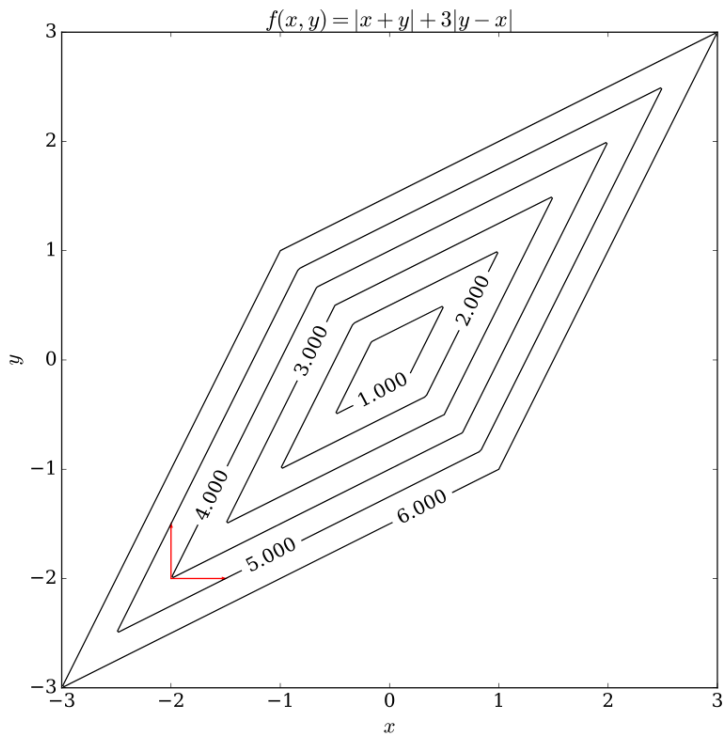
- The general idea for a coordinate descent algorithm is

Algorithm 1 Coordinate Descent/Minimization

1: Given initial $\boldsymbol{\beta}^{(0)} \in \mathbb{R}^p$, $t \leftarrow 0$
2: **repeat**
3: Choose $j \in \{1, 2, \ldots, p\}$
4: $\beta_j^{(t+1)} \leftarrow \arg\min_{\beta_j} f(\beta_j, \boldsymbol{\beta}_{-j}^{(t)})$, $\boldsymbol{\beta}_{-j}^{(t+1)} \leftarrow \boldsymbol{\beta}_{-j}^{(t)}$
5: $t \leftarrow t + 1$
6: **until** some convergence criterion is met

Remarks

- The function values are non-increasing.
- If f is strictly convex and smooth, CD converges to the global minimum.
- If f is nonconvex or not even smooth, CD may not converge at all or get stuck at a non-stationary point.



Coordinate Gradient Descent

- A popular alternative is to do one-step gradient descent for $\min_{\beta_j} f(\beta_j, \boldsymbol{\beta}_{-j}^{(t)})$, i.e., minimizing a quadratic approximation of $f(\beta_j, \boldsymbol{\beta}_{-j}^{(t)})$.

Algorithm 2 Coordinate Gradient Descent

1: Given initial $\boldsymbol{\beta}^{(0)} \in \mathbb{R}^p$, $t \leftarrow 0$
2: **repeat**
3: Choose $j \in \{1, 2, \ldots, p\}$
4: $\boldsymbol{\beta}^{(t+1)} \leftarrow \boldsymbol{\beta}^{(t)} - \alpha_t \nabla_j f(\boldsymbol{\beta}^{(t)}) \boldsymbol{e}_j$ for some $\alpha_t > 0$
5: $t \leftarrow t + 1$
6: **until** some convergence criterion is met

BCD

- The framework can be generalized for block updates, which is usually known as **block coordinate descent**.
- In the case when we have tow blocks, it simply reduces to the **alternating minimization**.

BCD Convergence

- Consider the following unconstrained optimization problems with block structure

$$\min_{\boldsymbol{\beta}=[\boldsymbol{\beta}_1^\top,\ldots,\boldsymbol{\beta}_k^\top]^\top \in \mathbb{R}^{p_1}\times\ldots\times\mathbb{R}^{p_k}} f(\boldsymbol{\beta}), \tag{1}$$

  where $\sum_j p_j = p$.
- Suppose that f is continuously differentiable over the convex closed set $\mathcal{X}$, and for each j,

$$g(\boldsymbol{\beta}') := f(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{j-1}, \boldsymbol{\beta}', \boldsymbol{\beta}_{j+1}, \ldots, \boldsymbol{\beta}_k)$$

  is **strictly convex**. Then every limit point of BCD iterates is a stationary point of (1).

Rules for selecting coordinates

- **Cyclic Order**: run all coordinates in cyclic order, i.e. $1 \to 2 \to \ldots \to p$ at each iteration
- **Random Sampling**: Randomly select some coordinate to update (uniformly or according to some distribution)
- **Gauss-Southwell**: At each iteration, pick coordinate $j = \arg\max_{1 \le j \le p} |\nabla_j f(\boldsymbol{\beta}^{(t)})|$.
- **Random Permutation**: run cyclic order on a permuted index (sample without replacement)

Convergence Rates

- Most convergence-rate analysis in the literature focus on block coordinate gradient descent, so we introduce some rate results for BCGD.
- Decompose the identity matrix $I_{p\times p} = [\boldsymbol{U}_1\ \boldsymbol{U}_2 \ldots \boldsymbol{U}_k]$, where $\boldsymbol{U}_j$ is a matrix of size of $p \times p_j$. Then

$$\boldsymbol{\beta} = \sum_j \boldsymbol{U}_j \boldsymbol{\beta}_j, \quad \boldsymbol{\beta}_j = \boldsymbol{U}_j^\top \boldsymbol{\beta}.$$

Assumptions

- f is convex and smooth, ∇f is L-Lipschitz continuous for some $L \geq 0$.
- $\nabla_j f(\boldsymbol{\beta})$ is L_j-Lipschitz continuous.
- Denote $L_{\max} = \max_j L_j, L_{\min} = \min_j L_j$. Then

$$L_j \leq L \leq \sum_j L_j \leq k L_{\max}.$$

GS-BCGD

- **Gauss-Southwell** Block Coordinate Gradient Descent
- At t-th iteration, pick up $j = \arg\max_j |\nabla f(\boldsymbol{\beta}^{(t)})|$,

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \frac{1}{L}\boldsymbol{U}_j \nabla_j f(\boldsymbol{\beta}^{(t)})$$

- If f is convex, then

$$f(\boldsymbol{\beta}^{(T)}) - f(\boldsymbol{\beta}^o) \leq \frac{2Lk\|\boldsymbol{\beta}^{(0)} - \boldsymbol{\beta}^o\|_2^2}{T}, \ T > 0$$

- If f is μ-strongly convex, then

$$f(\boldsymbol{\beta}^{(T)}) - f(\boldsymbol{\beta}^o) \leq (1 - \frac{\mu}{Lk})^T (f(\boldsymbol{\beta}^{(0)}) - f(\boldsymbol{\beta}^o)), \ T > 0$$

GS-BCGD

- The previous GS analysis under strong convexity loss too much when we use $\|\nabla_j f(\boldsymbol{\beta}^{(t)})\|_2^2 \geq \|\nabla f(\boldsymbol{\beta}^{(t)})\|/k$.
- A refined result can be derived if we measure the strong convexity in the ℓ_1-norm, i.e.,

$$f(\boldsymbol{\gamma}) \geq f(\boldsymbol{\beta}) + \langle \nabla f(\boldsymbol{\beta}), \boldsymbol{\gamma} - \boldsymbol{\beta} \rangle + \frac{\mu_1}{2}\|\boldsymbol{\gamma} - \boldsymbol{\beta}\|_1^2$$

- Minimizing both sides w.r.t. $\boldsymbol{\gamma}$, we obtain $f(\boldsymbol{\beta}^o) \geq f(\boldsymbol{\beta}) - \frac{1}{2\mu_1}\|\nabla f(\boldsymbol{\beta})\|_\infty^2 \geq f(\boldsymbol{\beta}) - \frac{1}{2\mu_1}\max_j \|\nabla_j f(\boldsymbol{\beta})\|_2^2$.
- Then

$$f(\boldsymbol{\beta}^{(T)}) - f(\boldsymbol{\beta}^o) \leq (1 - \frac{\mu_1}{L})^T (f(\boldsymbol{\beta}^{(0)}) - f(\boldsymbol{\beta}^o))$$

- Faster than previous result if $\mu_1 > \mu/k$.

Randomized-BCGD

- **Randomized** Block Coordinate Gradient Descent
- At t-th iteration, pick up j *randomly* from {1,2...,k},

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \frac{1}{L}\boldsymbol{U}_j \nabla_j f(\boldsymbol{\beta}^{(t)})$$

- If f is convex, then

$$\mathbb{E}[f(\boldsymbol{\beta}^{(T)}) - f(\boldsymbol{\beta}^o)] \leq \frac{2LkR^2(\boldsymbol{\beta}^{(0)})}{T},\ T > 0,$$

where $R(\boldsymbol{\beta}^{(0)}) = \max\{\|\boldsymbol{\beta} - \boldsymbol{\beta}^o\|_2 : f(\boldsymbol{\beta}) \leq f(\boldsymbol{\beta}^{(0)})\}$.

- If f is μ-strongly convex, then

$$\mathbb{E}[f(\boldsymbol{\beta}^{(T)}) - f(\boldsymbol{\beta}^o)] \leq (1 - \frac{\mu}{Lk})^T (f(\boldsymbol{\beta}^{(0)}) - f(\boldsymbol{\beta}^o)),\ T > 0$$

Randomized-BCGD

- How to improve the rate?
- Use larger step sizes: $\alpha_t = 1/L_{j_t}$
- Use non-uniform sampling distribution: $p_j = L_j / \sum_j L_j$
- These two ways give an improved convergence bound for convex f:

$$\mathbb{E}[f(\boldsymbol{\beta}^{(t)}) - f(\boldsymbol{\beta}^o)] \leq \frac{2\sum_j L_j R^2(\boldsymbol{\beta}^{(0)})}{T}$$

Cyclic-BCGD

- **Cyclic** Block Coordinate Gradient Descent
- At t-th iteration, let $\boldsymbol{\beta}_0^{(t)} = \boldsymbol{\beta}^{(t)}$, for $j = 1, 2 \ldots, k$, do

$$\boldsymbol{\beta}_j^{(t)} = \boldsymbol{\beta}_{j-1}^{(t)} - \frac{1}{L} \boldsymbol{U}_j \nabla_j f(\boldsymbol{\beta}_{j-1}^{(t)}),$$

  Set $\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}_k^{(t)}$.
- Note that the Cyclic-BCGD is a *deterministic* algorithm.
- The cost for Cyclic-BCGD is $\mathcal{O}(k)$ times larger then R-BCGD, since it needs to go through entire blocks.

Cyclic-BCGD

- If f is convex, then

$$f(\boldsymbol{\beta}^{(T)}) - f(\boldsymbol{\beta}^o) \leq \frac{4L(k+1)R^2(\boldsymbol{\beta}^{(0)})}{T}, \ T > 0$$

- If a larger step sizes $\alpha_t = 1/L_{j_t}$ is adopted, the rate can be improved to

$$f(\boldsymbol{\beta}^{(T)}) - f(\boldsymbol{\beta}^o) \leq \frac{4L_{\max}(kL^2/L_{\min}^2+1)R^2(\boldsymbol{\beta}^{(0)})}{T}$$

- If f is μ-strongly convex, then

$$f(\boldsymbol{\beta}^{(T)}) - f(\boldsymbol{\beta}^o) \leq (1 - \frac{\mu}{2(k+1)L})^T (f(\boldsymbol{\beta}^{(0)}) - f(\boldsymbol{\beta}^o)), \ T > 0$$

CD v.s. GD

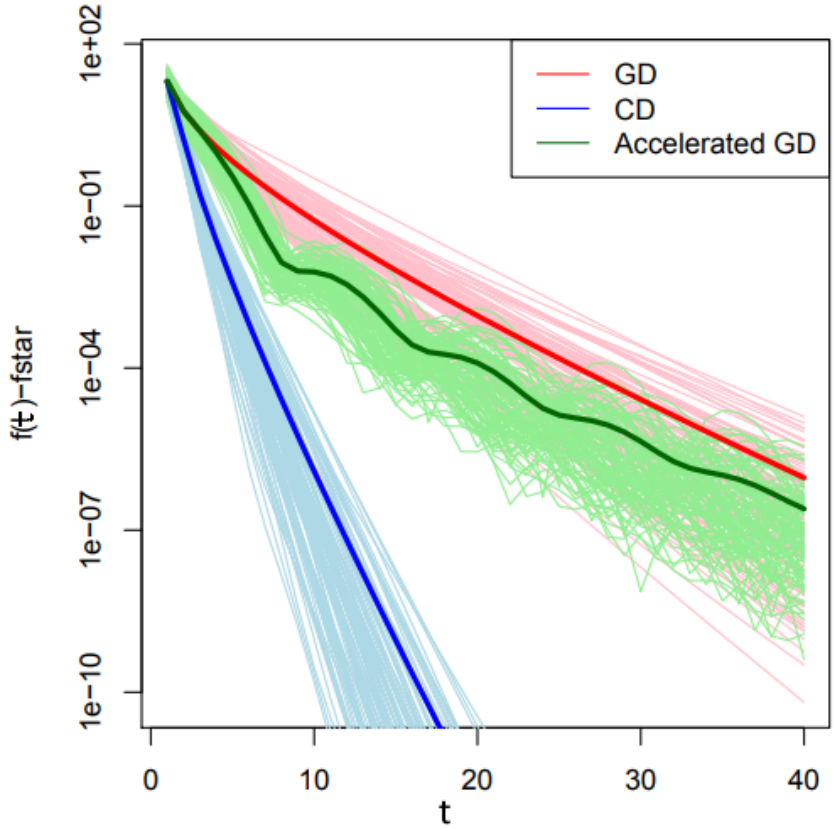


Figure: Linear regression with $n = 100, p = 20$

Summary

Algorithm	$\alpha_t = 1/L$	$\alpha_t = 1/L_{j_t}$
Gauss-Southwell	$\mathcal{O}(\frac{kL}{T})$	$\mathcal{O}(\frac{kL_{\max}}{T})$
Cyclic	$\mathcal{O}(\frac{kL}{T})$	$\mathcal{O}(\frac{L_{\max}(kL^2/L_{\min}^2+1)}{T})$
Randomized	$\mathcal{O}(\frac{kL}{T})$	$\mathcal{O}(\frac{\sum_j L_j}{T})$

Table: Convergence rates for three variants of BCGD

- Open questions: Are these bounds optimal? Under what condition can we drop the dependence of k? When cyclic version is better than randomized version?

Deterministic v.s. Stochastic Optimization

- Deterministic optimization problem

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} f_i(\beta). \tag{2}$$

- Stochastic optimization problem

$$\min_{\beta \in \mathbb{R}^p} \mathbb{E}_{\xi}[F(\beta, \xi)]. \tag{3}$$

  where ξ is a random variable.
- (2) can be written as (3) if ξ is a uniform discrete random variable with $\mathbb{P}(\xi = i) = 1/n, i = 1, \ldots, n$ and $F(\beta, \xi) = f_{\xi}(\beta)$.

Expected Risk

- In many supervised learning problems, input-output pairs $x, y \in \mathcal{X} \times \mathcal{Y}$ follow an unknown probability distribution $\mathcal{P}(x, y)$.
- The typical learning setting relies on
  - A loss function: $l : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$,
    $l(\hat{y}, y)$ measures the discrepancy between predicted $\hat{y}$ and true y
  - A predictor function $\phi : \mathcal{X} \to \mathcal{Y}$.
    $\phi(x)$ maps an input x to the predicted output $\hat{y}$.

Expected Risk

- Given a loss function l, expected risk of a predictor function ϕ is defined by

$$\boldsymbol{E}(\phi) := \int l(\phi(x), y) d\mathcal{P}(x, y).$$

- The **ideal** goal:

$$\phi^*(x) := \arg\min_{\hat{y} \in \mathcal{Y}} \int l(\hat{y}, y) d\mathcal{P}(y|x).$$

- This is a typical *Stochastic Optimization* problem.
- We do not know the source distribution $\mathcal{P}(x, y)$. Instead, we have access to **samples** from the distribution. (*Sample Average Approximation*)

Empirical Risk

- Given a training dataset $\{(x_1, y_i)\}_{i=1}^n$ (which is assumed to be drawn iid from $\mathcal{P}(x, y)$)
- The empirical risk $\boldsymbol{E}_n(\phi)$ as a proxy to $\boldsymbol{E}(\phi)$:

$$\boldsymbol{E}_n(\phi) := \frac{1}{n} \sum_{i=1}^{n} l(\phi(x_i), y_i)$$

- Further restriction: ϕ belongs to a specific class $\mathcal{F}$, e.g., linear functions $\phi_w(x) = w^\top x$.
- $\mathcal{F}$ together with the choice of l amounts to the choice of a specific machine learning technique.

Empirical Risk Minimization

- How to access the quality of a empirical predictor $\hat{\phi}^n$?
- Empirical minimizer $\phi^n_{\mathcal{F}} := \arg\min_{\phi \in \mathcal{F}} \boldsymbol{E}_n(\phi)$.
- Expected minimizer $\phi^*_{\mathcal{F}} := \arg\min_{\phi \in \mathcal{F}} \boldsymbol{E}(\phi)$.
- The goal of a machine learner is to minimize the **excess error** $\mathbb{E}[\boldsymbol{E}(\hat{\phi}^n) - \boldsymbol{E}(\phi^*)]$.
- $\mathbb{E}[\boldsymbol{E}(\hat{\phi}^n) - \boldsymbol{E}(\phi^*)] =$
$\underbrace{\mathbb{E}[(\boldsymbol{E}(\phi^*_{\mathcal{F}}) - \boldsymbol{E}(\phi^*))]}_{\textit{approximation error}} + \underbrace{\underbrace{\mathbb{E}[\boldsymbol{E}(\phi^n_{\mathcal{F}}) - \boldsymbol{E}(\phi^*_{\mathcal{F}})]}_{\textit{estimation error}} + \underbrace{\mathbb{E}[\boldsymbol{E}(\hat{\phi}^n) - \boldsymbol{E}(\phi^n_{\mathcal{F}})]}_{\textit{optimiation error}}}_{\mathbb{E}[\boldsymbol{E}(\hat{\phi}^n - \boldsymbol{E}(\phi^*_{\mathcal{F}}))]}$
- $\phi^*_{\mathcal{F}}$ reduces to $\phi^n_{\mathcal{F}}$ if $\mathcal{P}(x, y)$ is a uniform discrete distribution.

Stochastic Gradient Descent

- Originally designed for stochastic optimization, but still applicable for deterministic optimization (finite n).
- Recall the **finite sum problem**

$$\min_{\beta \in \mathbb{R}^p} f(\beta) := \frac{1}{n} \sum_{i=1}^{n} f_i(\beta)$$

- SGD update: $\beta^{(t+1)} = \beta^{(t)} - \alpha_t \underbrace{\nabla f_{i_t}(\beta^{(t)})}_{stoch.\ grad.}$
- Advantage: can deal with huge data and streaming data (online scenario)

Stochastic Gradient Descent

- **Unbiasedness**: the stochastic gradient is unbiased, i.e., $\mathbb{E}[\nabla f_{i_t}(\beta^{(t)})] = \nabla f(\beta^{(t)})$.
- **Decreasing step sizes**: since the stochastic gradient is random, we cannot guarantee $\nabla f_i(\beta^*) = 0, \forall i$. Hence, we need $\alpha_t \to 0$ as $t \to \infty$.
- **Convergence measure**: Due to the randomness, we cannot use previous error functions to measure the optimality. Instead, consider the expectation.

R-CD: A Special Case of SGD

- Randomized Coordinate Descent can be view as a special case of SGD, where $g^{(t)} = p\nabla_{j_t} f(\beta^{(t)}) e_{j_t}$.
- $\mathbb{E}[g^{(t)}] = \frac{1}{p}\sum_{j=1}^{p} p\nabla_j f(\beta^{(t)}) e_j = \nabla f(\beta^{(t)})$ certifies unbiasedness.
- The relationship can be seen from **dual** problems.

Example

- Linear system $Aw = b$, where $A \in \mathbb{R}^{m \times n}$ and $\|A_{i\cdot}\|_2 = 1$.
- The least-norm solution is found by solving $\min_{w \in \mathbb{R}^n} \frac{1}{2}\|w\|_2^2$ s.t. $Aw = b$, and its **dual problem** is

$$\min_{z \in \mathbb{R}^m} \frac{1}{2}\|A^\top z\|_2^2 - b^\top z.$$

- Applying CD to the dual problem with $\alpha_t = 1$ gives $z^{t+1} = z^t - (A_{i\cdot}A^\top z^t - b_i)e_i$, which corresponds

$$w^{t+1} = w^t - (A_{i\cdot}w^t - b_i)A_{i\cdot}^\top \tag{4}$$

- (4) can be seen as SGD applied to the *least square problem* $\min_w \frac{1}{2}\|Aw - b\|_2^2$.

Convergence Rate of SGD

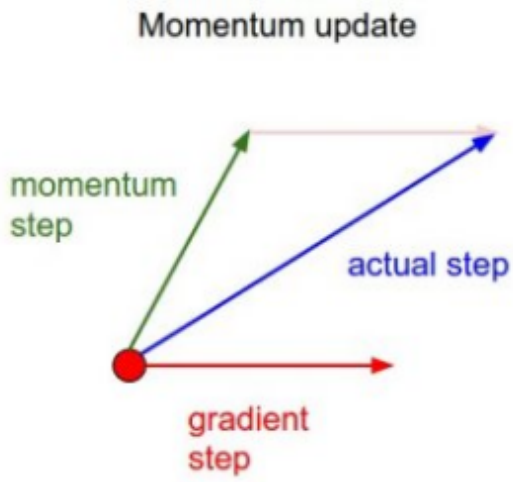
- Assume that $\mathbb{E}[\|\nabla f_i(\beta)\|_2^2] \leq G^2$. When $f(\beta)$ is μ-**strongly convex**, $\alpha_t = \mathcal{O}(\frac{1}{\mu t})$, we have

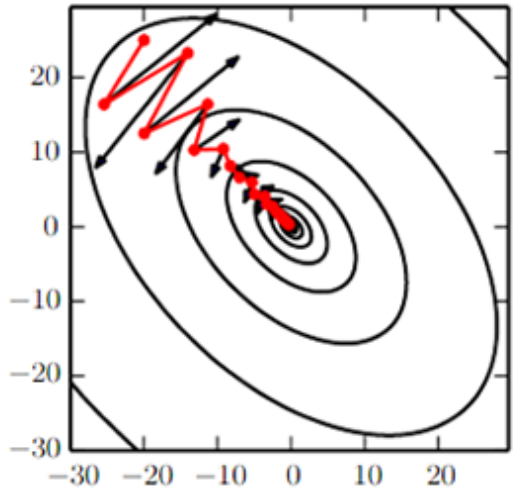
$$\mathbb{E}[f(\beta^{(T)}) - f(\beta^*)] \leq \frac{\max(\|\beta^{(1)} - \beta^*\|_2^2, G^2/\mu^2)}{T}$$

- The $\mathcal{O}(1/T)$ rate is **tight**! $\alpha_t = \mathcal{O}(\frac{1}{t})$ is theoretically best.
- For a general convex f, the rate of convergence is $\mathcal{O}(\frac{1}{\sqrt{T}})$.
- SGD is much slower than (accelerated) GD in terms of iteration number, but the per iteration cost is only $\mathcal{O}(1)$.

Momentum

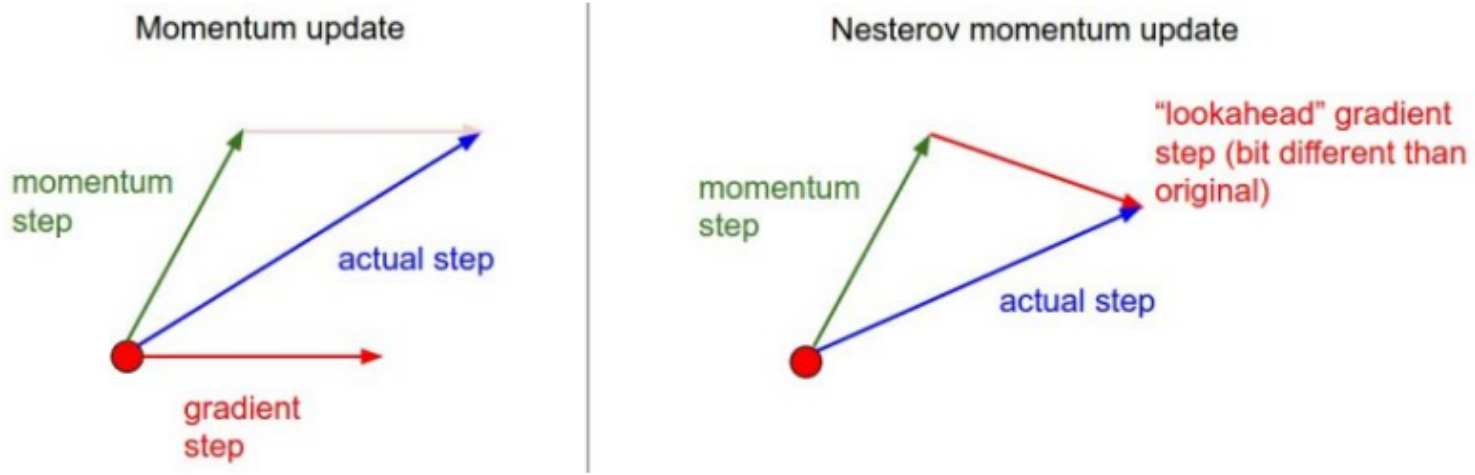
- Momentum introduces a variable `v` that plays the role of **velocity**, which is set to an exponentially decaying average of the negative gradient.
- ```
  v = mu * v - learning_rate * dx
  x += v
  ```




  ```

Nesterov Momentum

- ```
  x_ahead = x + mu * v
  v = mu * v - learning_rate * dx_ahead
  x += v
  ```
- The difference between Nesterov momentum and standard momentum is where the gradient is evaluated.



- Very popular in practice, but **NO improvement** for the rate of convergence.
  ```

Adaptive Learning Rates

- **AdaGrad**

```
cache += dx**2
x += - learning_rate * dx / (np.sqrt(cache)+eps)
```

- **RMSprop**

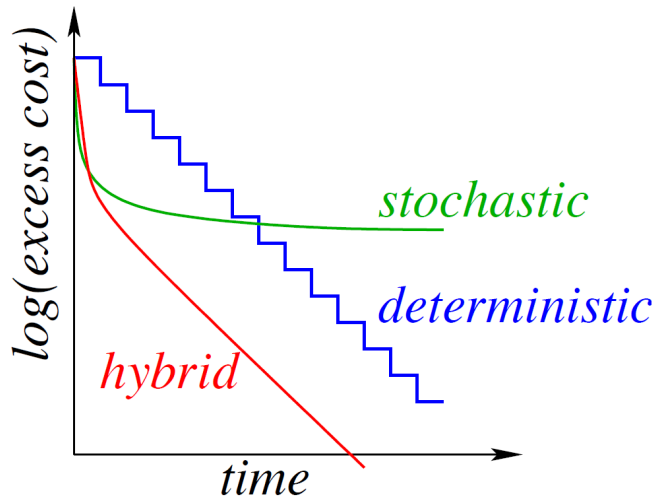
```
cache = decay_rate*cache + (1-decay_rate)*dx**2
x += - learning_rate * dx / (np.sqrt(cache)+eps)
```

- **Adam** (simplified version)

```
m = beta1*m + (1-beta1)*dx
v = beta2*v + (1-beta2)*(dx**2)
x += - learning_rate * m / (np.sqrt(v) + eps)
```

Incremental Gradient Methods

- GD enjoys a linear rate but $\mathcal{O}(n)$ iteration cost.
- SGD enjoys a sublinear rate but with $\mathcal{O}(1)$ iteration cost.



- For **finite** n, can we design a better method?
  Incremental Gradient Methods (**SAG**, SAGA, **SVRG**, S2GD, ...)

Variance Reduction

- In **expectation** both GD and SGD are identical, i.e.,

$$\mathbb{E}[\beta^{(t)}|\beta^{(t-1)}] = \beta^{(t)} - \alpha_t \Big[\frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\beta^{(t-1)})\Big].$$

- But the SG equals $\nabla f(\beta)$ only in expectation and may have large deviation from the mean.
- We observe $\mathcal{O}(1/T)$ rate due to the inherent large variance.

Batching

- The simplest variant is to use large sample.
- For a fixed batch size B, the rate is still **sublinear**. Another way is to grow B gradually.
- Growing B eventually requires $\mathcal{O}(n)$ iteration cost.
- Can we have 1 gradient per iteration but **linear** convergence rate?
  Yes! Stochastic Average Gradient

Variance Reduction Technique

- Suppose we want to estimate $\Theta = \mathbb{E}[X]$, and we have access to a random variable Y which is highly correlated with X, and we can compute $\mathbb{E}[Y]$ easily.
- Consider the point estimator $\hat{\Theta}_a$ with $a \in [0, 1]$

$$\hat{\Theta}_a = a(X - Y) + \mathbb{E}(Y).$$

- $\mathbb{E}[\hat{\Theta}_a]) = a\mathbb{E}[X] + (1 - a)\mathbb{E}[Y]$
  $\mathrm{Var}[\hat{\Theta}_a] = a^2(\mathrm{Var}[X] + \mathrm{Var}[Y] - 2\mathrm{Cov}[X, Y])$

Variance Reduction Techniques

$$\hat{\Theta}_a = a(X - Y) + \mathbb{E}(Y).$$

$$\mathbb{E}[\hat{\Theta}_a]) = a\mathbb{E}[X] + (1 - a)\mathbb{E}[Y]$$

$$\mathrm{Var}[\hat{\Theta}_a] = a^2(\mathrm{Var}[X] + \mathrm{Var}[Y] - 2\mathrm{Cov}[X, Y])$$

- If $\mathrm{Cov}[X, Y]$ is sufficiently large, then $\mathrm{Var}(\hat{\Theta}_a) < \mathrm{Var}[X]$. $\hat{\Theta}_a$ has **smaller variance** than the direct estimator X
- $a = 1$, $\hat{\Theta}_a$ is an **unbiased** estimator of X.
- $a = 0$, $\hat{\Theta}_a = \mathbb{E}[Y]$, **zero variance** but could be biased.
- As a increase from 0 to 1, the bias *decreases* and the variance *increases*.

Stochastic Average Gradient

- $\beta^{(t+1)} = \beta^{(t)} - \frac{\alpha_t}{n} \sum_i^n g_i^t$
- $g_i^t = \nabla f_i(\beta^t)$ from the last t where i was selected.
- **Memory**:

$$\begin{bmatrix} - & - & - & - & - & g_1 & - & - & - & - & - \\ - & - & - & - & - & g_2 & - & - & - & - & - \\ & & & & & \vdots & & & & & \\ - & - & - & - & - & g_n & - & - & - & - & - \end{bmatrix},$$

- Each g_i keeps track of the last time we randomly picked sample i. Update g_i if i is selected in the next iteration.
- Then take a step in the direction of the **average** of g_i.

Stochastic Average Gradient

- SAG can be rewritten as

$$\beta^{(t+1)} = \beta^{(t)} - \alpha\Big[\underbrace{\underbrace{\frac{1}{n}}_{a}\underbrace{(\nabla f_i(\beta^{(t)})}_{X} - \underbrace{g_i^{t-1}}_{Y}) + \underbrace{\frac{1}{n}\sum_{i=1}^{n} g_i^{t-1}}_{\mathbb{E}[Y]}}_{\hat{\Theta}_a}\Big]$$

- $\Theta = \nabla f(\beta^{(t)}), X = \nabla f_i(\beta^{(t)}), Y = g_i^{t-1}, a = 1/n.$
- SAGA: similar to SGA, only change $a = 1$.

Convergence Rate of SAG

- If f_i is L-smooth and f is μ-strongly convex with $\alpha = 1/16L$, SAG has

$$\mathbb{E}[f(\beta^{(T)}) - f(\beta^*)] \leq \mathcal{O}\left(\left[1 - \min\left(\frac{\mu}{16L}, \frac{1}{8n}\right)\right]^T\right)$$

- High memory cost: $\mathcal{O}(n)$!

Stochastic Variance Reduced Gradient

- SVRG: gets rid of memory by occasionally computing exact gradient.

Algorithm 3 SVRG

Initialize $\tilde{\beta}^{(0)}$
for $s = 1, 2, \ldots$ **do**
  $\tilde{\theta} = \sum_{i=1}^{n} \nabla f_i(\tilde{\beta}^{(s-1)})$
  $\beta^{(0)} = \tilde{\beta}^{(s-1)}$
  for $t = 1, 2, \ldots, m$ **do**
    Randomly pick $i \in \{1, 2, \ldots, n\}$
    $\beta^{(t)} = \beta^{(t-1)} - \alpha[\nabla f_i(\beta^{(t-1)}) - \nabla f_i(\tilde{\beta}^{(s-1)}) + \tilde{\theta}]$
  end for
  $\tilde{\beta}^{(s)} = \beta^{(m)}$
end for

Stochastic Variance Reduced Gradient

- At the s-th epoch, SVRG updates

$$\beta^{(t+1)} = \beta^{(t)} - \alpha\underbrace{[\underbrace{\nabla f_i(\beta^{(t)})}_{X} - \underbrace{\nabla f_i(\tilde{\beta}^{(s-1)})}_{Y} + \underbrace{\sum_{i=1}^{n} \nabla f_i(\tilde{\beta}^{(s-1)})}_{\mathbb{E}[Y]}]}_{\hat{\Theta}_a}$$

- $\Theta = \nabla f(\beta^{(t)}), X = \nabla f_i(\beta^{(t)}), Y = \nabla f_i(\tilde{\beta}^{(s-1)}), a = 1.$

Convergence Rate of SVRG

- Assume f_i is L-smooth and f is μ-strongly convex. If m is sufficiently large and $\alpha < 1/2L$, SVRG enjoys

$$\mathbb{E}[f(\tilde{\beta}^{(S)}) - f(\beta^*)] \leq \mathcal{O}\Big(\Big[\frac{1}{\mu\alpha(1-2L\alpha)m} + \frac{2L\alpha}{1-2L\alpha}\Big]^S\Big)$$

- Low memory cost, but require **two** loops.
- S2GD is same as SVRG, but with *random* number of inner steps.

Algorithm Comparison

- Logistic regression:

$$\min_{\beta} f(\beta) := \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp -y_i\beta^\top x_i) + \frac{\lambda}{2}\|\beta\|_2^2$$

- $X \in \mathbb{R}^{n\times p} \sim N(0, I), n = 10000, p = 50, \lambda = 10^{-4}$. The columns of X are normalized. y_i is generated according to the logistic model.
- $L = \lambda + \|x_i\|_2^2/4 = \lambda + 1/4,\ \mu = \lambda.$
- GD: $\alpha = 2/(L+\mu)$
- SGD: $\alpha_t = 0.1/(\mu(t+100)), B = 50$
- SVRG: $\alpha = 1/L, B = 50, m = 3n$

Algorithm Comparison

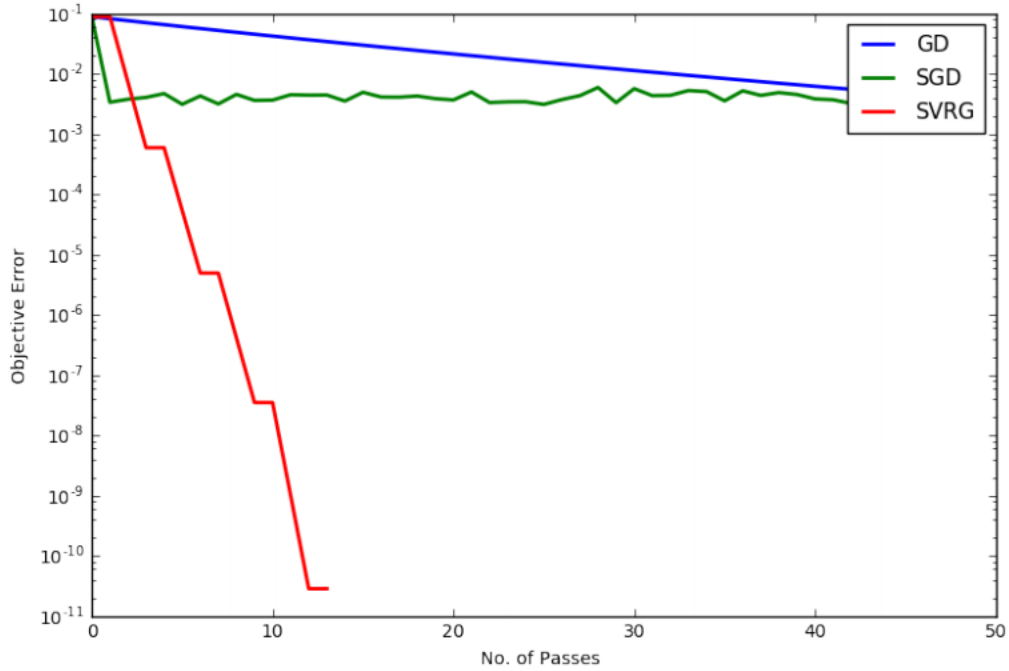


Figure: Comparison of GD, SGD, and SVRG on a synthetic dataset