A Stochastic Gradient Method with an Exponential Convergence Rate for Finite Training Sets

Nicolas Le Roux\textsuperscript{1,2}, Mark Schmidt\textsuperscript{1} and Francis Bach\textsuperscript{1}

\textsuperscript{1}Sierra project-team, INRIA - Ecole Normale Supérieure, Paris
\textsuperscript{2}Now at Criteo

4/12/12
Large-scale machine learning: large $n$, large $p$

- $n$: number of observations (inputs)
- $p$: number of parameters in the model

Examples: vision, bioinformatics, speech, language, etc.

- Pascal large-scale datasets: $n = 5 \cdot 10^5$, $p = 10^3$
- ImageNet: $n = 10^7$
- Industrial datasets: $n > 10^8$, $p > 10^7$

Main computational challenge:

- Design algorithms for very large $n$ and $p$. 
A standard machine learning optimization problem

We want to minimize the sum of a finite set of smooth functions:

$$\min_{\theta \in \mathbb{R}^p} g(\theta) = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta)$$

For instance, we may have

$$f_i(\theta) = \log (1 + \exp (-y_i x_i^\top \theta)) + \frac{\lambda}{2} \|\theta\|^2$$

We will focus on strongly-convex functions $g$. 
Deterministic methods

$$\min_{\theta \in \mathbb{R}^p} g(\theta) = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta)$$

**Gradient descent updates**

$$\theta_{k+1} = \theta_k - \alpha_k g' (\theta_k)$$

$$= \theta_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} f'_i (\theta_k)$$

- Iteration cost in $O(n)$
- Linear convergence rate $O(C^k)$
- Fancier methods exist but still in $O(n)$
Stochastic methods

\[
\min_{\theta \in \mathbb{R}^p} g(\theta) = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta)
\]

Stochastic gradient descent updates

\[
i(k) \sim \mathcal{U}[1, n]
\]

\[
\theta_{k+1} = \theta_k - \alpha_k f'_i(\theta_k)
\]

- Iteration cost in \(O(1)\)
- Sublinear convergence rate \(O(1/k)\)
- Bound on the test error valid for one pass
Hybrid methods

\[ \text{Goal = linear rate} \]

\[ O^{(1)} \text{ iteration cost.} \]

Stochastic Average Gradient

\[ \log(\text{excess cost}) \]

\[ \text{stochastic} \]

\[ \text{deterministic} \]
Goal = linear rate and $O(1)$ iteration cost.
Related work - Sublinear convergence rate

- **Stochastic version of full gradient methods**

- **Momentum, gradient/iterate averaging**

- None of these methods improve on the $O(1/k)$ rate
Related work - Linear convergence rate

- **Constant step-size SG, accelerated SG**
  - Linear convergence but only up to a fixed tolerance

- **Hybrid methods, incremental average gradient**
  - Linear rate but iterations make full passes through the data

- **Stochastic methods in the dual**
  - Shalev-Shwartz and Zhang (2012)
  - Linear rate but limited choice for the $f_i$'s
Full gradient update:

\[ \theta_{k+1} = \theta_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} f_i'(\theta_k) \]
Stochastic Average Gradient Method

Full gradient update:

\[ \theta_{k+1} = \theta_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} f_i'(\theta_k) \]
Stochastic average gradient update:

\[ \theta_{k+1} = \theta_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} y_i^k \]

- **Memory**: \( y_i^k = f_{i(k')}^{'}(\theta_{k'}) \) from the last \( k' \) where \( i \) was selected.
- Random selection of \( i(k) \) from \( \{1, 2, \ldots, n\} \).
- Only evaluates \( f_{i(k)}^{'}(\theta_k) \) on each iteration.

**Stochastic** variant of incremental average gradient [Blatt et al., 2007]
SAG convergence analysis

- Assume each $f_i'$ is $L$-continuous, average $g$ is $\mu$-strongly convex.
- With step size $\alpha_k \leq \frac{1}{2nL}$, SAG has linear convergence rate.
- **Linear convergence with iteration cost independent of** $n$.
- With step size $\alpha_k = \frac{1}{2n\mu}$, if $n \geq 8 \frac{L}{\mu}$ then
  \[
  \mathbb{E}[g(\theta_k) - g(\theta^*)] \leq C \left(1 - \frac{1}{8n}\right)^k.
  \]
- Rate is “independent” of the condition number.
  - Constant error reduction after each pass,
  \[
  \left(1 - \frac{1}{8n}\right)^n \leq \exp\left(-\frac{1}{8}\right) = 0.8825.
  \]
Comparison with full gradient methods

Assume $L = 100$, $\mu = 0.01$ and $n = 80000$:

- Full gradient has rate $\left(1 - \frac{\mu}{L}\right)^2 = 0.9998$
- Accelerated gradient has rate $\left(1 - \sqrt{\frac{\mu}{L}}\right) = 0.9900$
- SAG ($n$ iterations) multiplies the error by $\left(1 - \frac{1}{8n}\right)^n = 0.8825$
- Fastest possible first-order method has rate $\left(\frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}\right)^2 = 0.9608$

We beat two lower bounds (with additional assumptions)

- Stochastic gradient bound
- Full gradient bound
Experiments - Training cost

Quantum dataset \((n = 50000, p = 78)\)

\(\ell_2\)-regularized logistic regression

Nicolas Le Roux, Mark Schmidt, Francis Bach

Stochastic Average Gradient
Experiments - Training cost

RCV1 dataset ($n = 20242$, $p = 47236$)

$\ell_2$-regularized logistic regression

Nicolas Le Roux, Mark Schmidt, Francis Bach

Stochastic Average Gradient
Experiments - Testing cost

Quantum dataset \((n = 50000, p = 78)\)

\(\ell_2\)-regularized logistic regression

![Graph showing test logistic loss over effective passes for different algorithms: AFG, L-BFGS, pegasos, SAG-C, SAG-LS. The y-axis represents test logistic loss and the x-axis represents effective passes. The graph illustrates the performance of each algorithm as they converge to a minimum loss value.](image-url)
Experiments - Testing cost

RCV1 dataset ($n = 20242, p = 47236$)

$\ell_2$-regularized logistic regression

Nicolas Le Roux, Mark Schmidt, Francis Bach

Stochastic Average Gradient
Reducing memory requirements

- $\theta_{k+1} = \theta_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} y_i^k$
- $y_i^k$ is the last gradient computed on datapoint $i$
- Memory requirement : $O(np)$
- Smaller for structured models, e.g., linear models :
  - If $f_i(\theta) = \ell(y_i, x_i^\top \theta)$, $f'_i(\theta) = \ell'(y_i, x_i^\top \theta)x_i$
  - Memory requirement : $O(n)$
- We can also use mini-batches
Conclusion and Open Problems

- Fast theoretical convergence using the ‘sum’ structure common in applications.
- Simple algorithm, empirically better than theory predicts.
- Allows line-search and approximate optimality measures.
- Open problems:
  - Large-scale distributed implementation.
  - Determine a tight convergence rate in all cases.
  - Apply the method to constrained and non-smooth problems.
  - Speed up the method using non-uniform sampling and non-Euclidean metric.