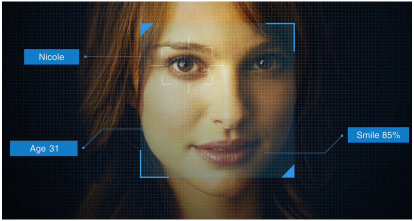
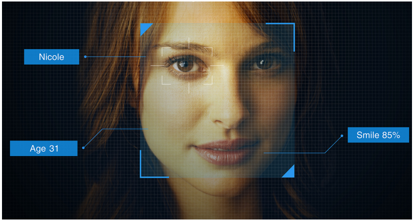
Monte Carlo Techniques in Modern Stochastic Optimization for Big Data Machine Learning

Tong Zhang

Industrial Trend: big data enabled intelligent systems



Industrial Trend: big data enabled intelligent systems



big data + complex model + large scale computing

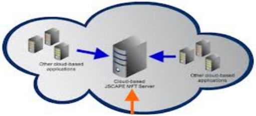
Machine Learning & AI

big data enabled intelligent systems

big data enabled intelligent systems

- Approach:

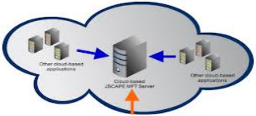
Big Data + Big Computing Power + Big Model + Big Optimization = Intelligence



big data enabled intelligent systems

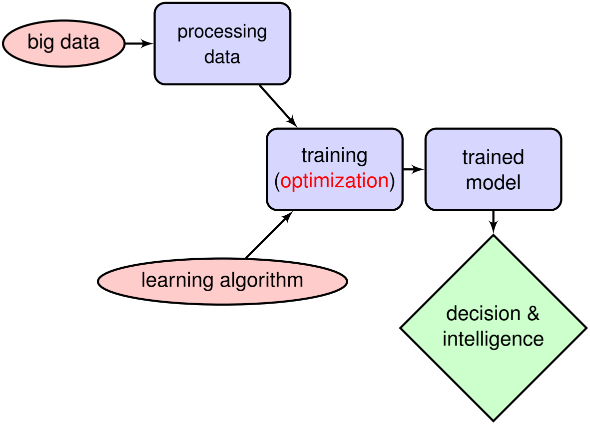
- Approach:

Big Data + Big Computing Power + Big Model + Big Optimization = Intelligence



Require us to solve large scale machine learning problems

Machine Learning Pipeline



Problem Scale: Image Classification

Problem Scale: Image Classification

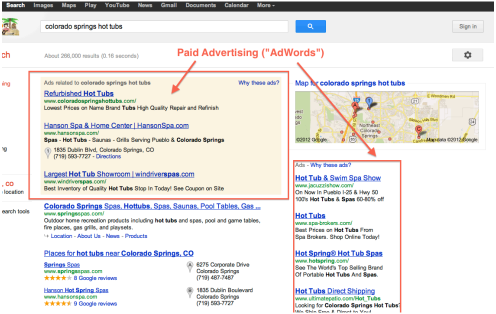
- training data size: $\sim$ 10 million
- classes: $\sim 10^4$
- model: deep neural networks
- training time: $\sim$ week on GPU servers
- near human accuracy

Problem Scale: Speech Recognition

Problem Scale: Speech Recognition

- training data size: $\sim$ billion instances (tens of thousands recordings)
- model: deep neural networks
- training time: $\sim$ weeks on GPU servers
- near human performance

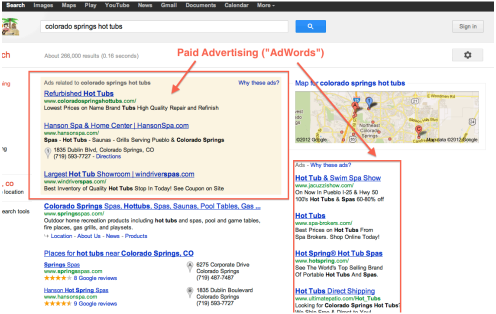
Problem Scale: Computational Advertising



Statistical Problem:
click through rate (CTR) estimation

- the probability a user clicks an ad

Problem Scale: Computational Advertising



Statistical Problem:
click through rate (CTR) estimation

- the probability a user clicks an ad

Big data linear or nonlinear logistic regression:

- training data size: up to $n \sim$ 100 billion
- high dimension: up to $\dim(x_i) \sim$ 100 billion
  - each instance has no more than a few hundred nonzeros
- training time: hours to days on hundreds of CPU servers

Challenges for Big Data Intelligence

System:

- distributed computing with many machines
- hybrid computing (cpu + gpu)
- real time streaming computing

Challenges for Big Data Intelligence

System:

- distributed computing with many machines
- hybrid computing (cpu + gpu)
- real time streaming computing

Statistics:

- complex nonlinear models (deep neural networks)

Challenges for Big Data Intelligence

System:

- distributed computing with many machines
- hybrid computing (cpu + gpu)
- real time streaming computing

Statistics:

- complex nonlinear models (deep neural networks)

Optimization:

- efficient methods for solving large scale machine learning problems
  Monte Carlo sampling methods

Mathematical Problem

Big Data Optimization Problem in machine learning:

$$\min_w f(w) \qquad f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

Special structure: sum over data: large n

Big Data Optimization Problem in machine learning:

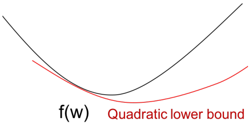
$$\min_w f(w) \qquad f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

Special structure: sum over data: large n

Assumptions on loss function

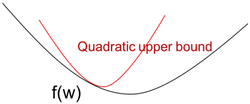
- λ-strong convexity:

$$f(w') \geq \underbrace{f(w) + \nabla f(w)^\top (w' - w) + \frac{\lambda}{2}\|w' - w\|_2^2}_{\text{quadratic lower bound}}$$



- L-smoothness:

$$f_i(w') \leq \underbrace{f_i(w) + \nabla f_i(w)^\top (w' - w) + \frac{L}{2}\|w' - w\|_2^2}_{\text{quadratic upper bound}}$$



Example: Computational Advertising

Large scale regularized logistic regression

$$\min_w \frac{1}{n}\sum_{i=1}^{n}\left[\underbrace{\ln(1+e^{-w^\top x_i y_i})+\frac{\lambda}{2}\|w\|_2^2}_{f_i(w)}\right]$$

- data (x_i, y_i) with $y_i \in \{\pm 1\}$; model parameter vector w.
- λ strongly convex
- $L = 0.25 \max_i \|x_i\|_2^2 + \lambda$ smooth.

Example: Computational Advertising

Large scale regularized logistic regression

$$\min_w \frac{1}{n}\sum_{i=1}^{n}\Bigg[\underbrace{\ln(1+e^{-w^\top x_i y_i}) + \frac{\lambda}{2}\|w\|_2^2}_{f_i(w)}\Bigg]$$

- data (x_i, y_i) with $y_i \in \{\pm 1\}$; model parameter vector w.
- λ strongly convex
- $L = 0.25\max_i \|x_i\|_2^2 + \lambda$ smooth.

Problem size:

- big data: $n \sim 10 - 100$ billion
- high dimension: $\dim(x_i) \sim 10 - 100$ billion

Example: Computational Advertising

Large scale regularized logistic regression

$$\min_w \frac{1}{n}\sum_{i=1}^{n}\left[\underbrace{\ln(1+e^{-w^\top x_i y_i}) + \frac{\lambda}{2}\|w\|_2^2}_{f_i(w)}\right]$$

- data (x_i, y_i) with $y_i \in \{\pm 1\}$; model parameter vector w.
- λ strongly convex
- $L = 0.25 \max_i \|x_i\|_2^2 + \lambda$ smooth.

Problem size:

- big data: $n \sim 10 - 100$ billion
- high dimension: $\dim(x_i) \sim 10 - 100$ billion

How to solve big optimization problems efficiently?

Outline of the Talk

Modern stochastic optimization for convex big data machine learning

Use techniques from Monte Carlo Methods for variance reduction

Outline of the Talk

Modern stochastic optimization for convex big data machine learning

Use techniques from Monte Carlo Methods for variance reduction

- Background: stochastic gradient versus batch gradient
- SVRG (Stochastic Variance Reduced Gradient): control variates
- Importance sampling and stratefied sampling approaches
- SAGA (Stochastic Average Gradient Ameliore)

Batch Optimization Method: Gradient Descent

Solve

$$w_* = \arg\min_w f(w) \qquad f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w).$$

Gradient Descent (GD):

$$w_k = w_{k-1} - \eta_k \nabla f_i(w_{k-1}) = w_{k-1} - \eta_k \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_{k-1}).$$

How fast does this method converge to the optimal solution?

Batch Optimization Method: Gradient Descent

Solve

$$w_* = \arg\min_w f(w) \qquad f(w) = \frac{1}{n}\sum_{i=1}^n f_i(w).$$

Gradient Descent (GD):

$$w_k = w_{k-1} - \eta_k \nabla f_i(w_{k-1}) = w_{k-1} - \eta_k \frac{1}{n}\sum_{i=1}^n \nabla f_i(w_{k-1}).$$

How fast does this method converge to the optimal solution?

- For λ-strongly convex and L-smooth problems, it is linear rate:

$$f(w_k) - f(w_*) = O((1-\rho)^k),$$

  where $\rho = O(\lambda/L)$ is the inverse condition number

How to deal with big data? sampling!

- Objective function:

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

sample objective function: only optimize approximate objective

How to deal with big data? sampling!

- Objective function:

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

sample objective function: only optimize approximate objective

- 1st order gradient

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

sample 1st order gradient (stochastic gradient):
    - converge to exact optimal
    - variance reduction leads to fast rate

Stochastic Approximate Gradient Computation

If

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w),$$

GD requires the computation of full gradient, which is extremely costly

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

Stochastic Approximate Gradient Computation

If

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w),$$

GD requires the computation of full gradient, which is extremely costly

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

Idea: stochastic optimization employs random sample (mini-batch) B to approximate

$$\nabla f(w) \approx \frac{1}{|B|} \sum_{i \in B} \nabla f_i(w)$$

- It is an unbiased estimator
- more efficient computation but introduces variance

SGD versus GD example

For ridge regression,

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} \underbrace{(w^\top x_i - y_i)^2 + \frac{\lambda}{2}\|w\|_2^2}_{f_i(w)}$$

GD rule is

$$w_t = (1 - \eta\lambda) w_{t-1} - 2\eta \cdot \frac{1}{n} \sum_{i=1}^{n} (w_{t-1}^\top x_i - y_i) x_i$$

SGD rule (with $|B| = 1$) is

$$w_t = (1 - \eta\lambda) w_{t-1} - 2\eta\, (w_{t-1}^\top x_i - y_i) x_i$$

SGD versus GD

SGD:

- faster computation per step
- Sublinear convergence: due to the variance of gradient approximation.

$$f(w_t) - f(w_*) = \tilde{O}(1/t).$$

GD:

- slower computation per step
- Linear convergence:

$$f(w_t) - f(w_*) = O((1-\rho)^t).$$

SGD versus GD

SGD:

- faster computation per step
- Sublinear convergence: due to the variance of gradient approximation.

$$f(w_t) - f(w_*) = \tilde{O}(1/t).$$

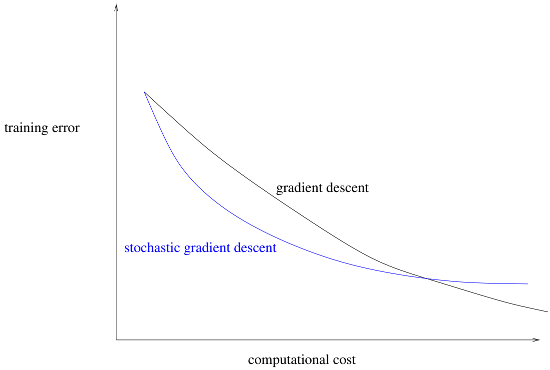
GD:

- slower computation per step
- Linear convergence:

$$f(w_t) - f(w_*) = O((1-\rho)^t).$$

Overall: sgd is fast in the beginning but slow asymptotically

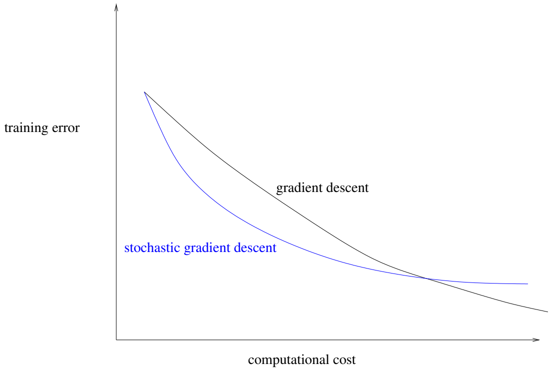
SGD versus GD



One strategy:

- use sgd first to train
- after a while switch to batch methods such as LBFGS.

SGD versus GD



One strategy:

- use sgd first to train
- after a while switch to batch methods such as LBFGS.

However, one can do better

Improving SGD via Variance Reduction

- GD converges fast but computation is slow
- SGD computation is fast but converges slowly
  - slow convergence due to inherent variance
- SGD as a statistical estimator of gradient:
  - let $\mathbf{g}_i = \nabla f_i$.
  - unbaisedness: $\mathbf{E}\, \mathbf{g}_i = \frac{1}{n}\sum_{i=1}^{n} \mathbf{g}_i = \nabla f$.
  - error of using $\mathbf{g}_i$ to approx ∇f: variance $\mathbf{E}\|\mathbf{g}_i - \mathbf{E}\mathbf{g}_i\|_2^2$.

Improving SGD via Variance Reduction

- GD converges fast but computation is slow
- SGD computation is fast but converges slowly
  - slow convergence due to inherent variance
- SGD as a statistical estimator of gradient:
  - let $\mathbf{g}_i = \nabla f_i$.
  - unbaisedness: $\mathbf{E}\,\mathbf{g}_i = \frac{1}{n}\sum_{i=1}^{n} \mathbf{g}_i = \nabla f$.
  - error of using $\mathbf{g}_i$ to approx ∇f: variance $\mathbf{E}\|\mathbf{g}_i - \mathbf{E}\mathbf{g}_i\|_2^2$.
- Statistical thinking:
  - relating variance to optimization
  - design other unbiased gradient estimators with smaller variance

Relating Statistical Variance to Optimization

Want to optimize

$$\min_w f(w)$$

Full gradient $\nabla f(w)$.

Relating Statistical Variance to Optimization

Want to optimize

$$\min_w f(w)$$

Full gradient $\nabla f(w)$.

Given unbiased random estimator $\mathbf{g}_i$ of $\nabla f(w)$, and SGD rule

$$w \rightarrow w - \eta \mathbf{g}_i,$$

reduction of objective is

$$\mathbf{E} f(w - \eta \mathbf{g}_i) \leq \underbrace{f(w) - (\eta - \eta^2 L/2)\|\nabla f(w)\|_2^2}_{\text{non-random}} + \frac{\eta^2 L}{2} \underbrace{\mathbf{E}\|\mathbf{g} - \mathbf{E}\mathbf{g}\|_2^2}_{\text{variance}}.$$

Relating Statistical Variance to Optimization

Want to optimize

$$\min_w f(w)$$

Full gradient $\nabla f(w)$.

Given unbiased random estimator $\mathbf{g}_i$ of $\nabla f(w)$, and SGD rule

$$w \to w - \eta \mathbf{g}_i,$$

reduction of objective is

$$\mathbf{E} f(w - \eta \mathbf{g}_i) \leq \underbrace{f(w) - (\eta - \eta^2 L/2)\|\nabla f(w)\|_2^2}_{\text{non-random}} + \frac{\eta^2 L}{2} \underbrace{\mathbf{E}\|\mathbf{g} - \mathbf{E}\mathbf{g}\|_2^2}_{\text{variance}}.$$

Smaller variance implies bigger reduction

Improving SGD using Variance Reduction

Idea: design unbiased stochastic gradient estimator with small variance.

Improving SGD using Variance Reduction

Idea: design unbiased stochastic gradient estimator with small variance.

The idea leads to modern stochastic algorithms for big data machine learning with fast convergence rate

Improving SGD using Variance Reduction

Idea: design unbiased stochastic gradient estimator with small variance.

The idea leads to modern stochastic algorithms for big data machine learning with fast convergence rate

Representative work

- Le Roux, Schmidt, Bach (NIPS 2012): A variant of SGD called SAG (stochastic average gradient) and later SAGA
- Johnson and Z (NIPS 2013): SVRG (Stochastic variance reduced gradient)
- Shalev-Schwartz and Z (JMLR 2013): SDCA (Stochastic Dual Coordinate Ascent) , and later a variant with Zheng Qu and Peter Richtarik

Outline

- Background: stochastic gradient versus batch gradient
- SVRG (Stochastic Variance Reduced Gradient): control variates
- Importance sampling and stratefied sampling approaches
- SAGA (Stochastic Average Gradient Ameliore)

Monte Carlo Methods: Variance Reduction Techniques

- Given unbiased estimator $\mathbf{g}_i$ of ∇f; how to design other unbiased estimators with reduce variance?

Monte Carlo Methods: Variance Reduction Techniques

- Given unbiased estimator $\mathbf{g}_i$ of ∇f; how to design other unbiased estimators with reduce variance?
- Control variates.
  - find $\tilde{\mathbf{g}}_i \approx \mathbf{g}_i$
  - use compensated estimator

$$\mathbf{g}'_i := \mathbf{g}_i - \tilde{\mathbf{g}}_i + \mathbf{E}\, \tilde{\mathbf{g}}_i.$$

Monte Carlo Methods: Variance Reduction Techniques

- Given unbiased estimator $\mathbf{g}_i$ of ∇f; how to design other unbiased estimators with reduce variance?
- Control variates.
  - find $\tilde{\mathbf{g}}_i \approx \mathbf{g}_i$
  - use compensated estimator

$$\mathbf{g}_i' := \mathbf{g}_i - \tilde{\mathbf{g}}_i + \mathbf{E}\, \tilde{\mathbf{g}}_i.$$

- Importance sampling:
  - sample $\mathbf{g}_i$ proportional to ρ_i ($\mathbf{E}\rho_i = 1$)
  - use estimator $\mathbf{g}_i / \rho_i$

Monte Carlo Methods: Variance Reduction Techniques

- Given unbiased estimator $\mathbf{g}_i$ of ∇f; how to design other unbiased estimators with reduce variance?
- Control variates.
  - find $\tilde{\mathbf{g}}_i \approx \mathbf{g}_i$
  - use compensated estimator

$$\mathbf{g}'_i := \mathbf{g}_i - \tilde{\mathbf{g}}_i + \mathbf{E}\,\tilde{\mathbf{g}}_i.$$

- Importance sampling:
  - sample $\mathbf{g}_i$ proportional to ρ_i ($\mathbf{E}\rho_i = 1$)
  - use estimator $\mathbf{g}_i/\rho_i$
- Stratified sampling (a minibatch of $b = b_1 + \ldots + b_K$):
  - divide $\{\mathbf{g}_1, \ldots, \mathbf{g}_n\}$ into K subsets $\{G_\ell : \ell = 1, \ldots, K\}$ with small within group variance
  - use estimator $n^{-1} \sum_{\ell=1}^{K} (|G_\ell|/b_\ell) \sum_{j=1}^{b_\ell} \mathbf{g}_{\ell,j}$ where $\mathbf{g}_{\ell,j}$ uniformly drawn from G_ℓ

Stochastic Variance Reduced Gradient: Derivation

Objective function

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w) = \frac{1}{n}\sum_{i=1}^{n} \tilde{f}_i(w),$$

where the gradient compensated objective is:

$$\tilde{f}_i(w) = f_i(w) - \underbrace{(\nabla f_i(\tilde{w}) - \nabla f(\tilde{w}))^\top w}_{\text{sum to zero}}.$$

Pick $\tilde{w}$ to be an approximate solution (close to w_*).

Stochastic Variance Reduced Gradient: Derivation

Objective function

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w) = \frac{1}{n}\sum_{i=1}^{n} \tilde{f}_i(w),$$

where the gradient compensated objective is:

$$\tilde{f}_i(w) = f_i(w) - \underbrace{(\nabla f_i(\tilde{w}) - \nabla f(\tilde{w}))^\top w}_{\text{sum to zero}}.$$

Pick $\tilde{w}$ to be an approximate solution (close to w_*).
SVRG rule:

$$w_t = w_{t-1} - \eta_t \nabla \tilde{f}_i(w_{t-1}) = w_{t-1} - \eta_t \underbrace{\left[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w})\right]}_{\text{small variance}}.$$

Compare to SGD rule:

$$w_t = w_{t-1} - \eta_t \underbrace{\nabla f_i(w_{t-1})}_{\text{large variance}}$$

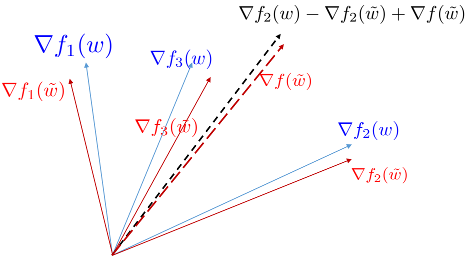
SVRG rule:

$$w_t = w_{t-1} - \eta_t[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w})].$$

If $\tilde{w} \to w_*$ and $w_{t-1} \to w_*$, then

$$\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w}) \approx \nabla f_i(w_*) - \nabla f_i(w_*) + \nabla f(w_*) \to 0.$$

Variance of SVRG estimator converges to zero.



SVRG Algorithm

Procedure SVRG

Parameters update frequency m and learning rate η
Initialize $\tilde{w}_0$
Iterate: for $s = 1, 2, \ldots$
  $\tilde{w} = \tilde{w}_{s-1}$
  $\tilde{\mu} = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{w})$
  $w_0 = \tilde{w}$
  Iterate: for $t = 1, 2, \ldots, m$
    Randomly pick $i_t \in \{1, \ldots, n\}$ and update weight
      $w_t = w_{t-1} - \eta(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) + \tilde{\mu})$
  end
  Set $\tilde{w}_s = w_m$
end

SVRG v.s. Batch Gradient Descent: fast convergence

Assume L-smooth loss and λ strongly convex objective function. One can prove linear convergence for SVRG:

$$Ef(w_t) - f(w_*) = O((1-\tilde{\rho})^t),$$

where $\tilde{\rho} = O(\lambda n/(L+\lambda n)$; convergence is faster than GD.

SVRG v.s. Batch Gradient Descent: fast convergence

Assume L-smooth loss and λ strongly convex objective function. One can prove linear convergence for SVRG:

$$Ef(w_t) - f(w_*) = O((1-\tilde{\rho})^t),$$

where $\tilde{\rho} = O(\lambda n/(L + \lambda n))$; convergence is faster than GD.

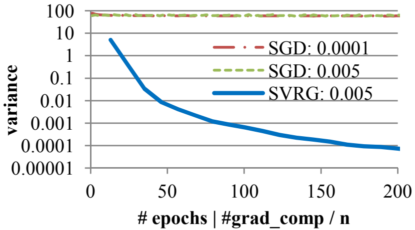
Number of examples needed to achieve ϵ accuracy:

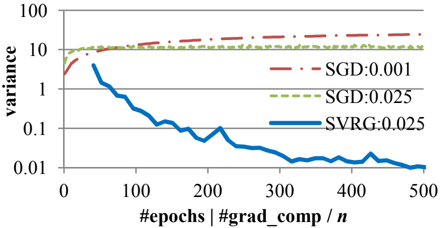
- Batch GD: $\tilde{O}(n \cdot L/\lambda \log(1/\epsilon))$
- SVRG: $\tilde{O}((n + L/\lambda)\log(1/\epsilon))$

SVRG has fast convergence — condition number effectively reduced

The gain of SVRG over batch algorithm is significant when n is large.

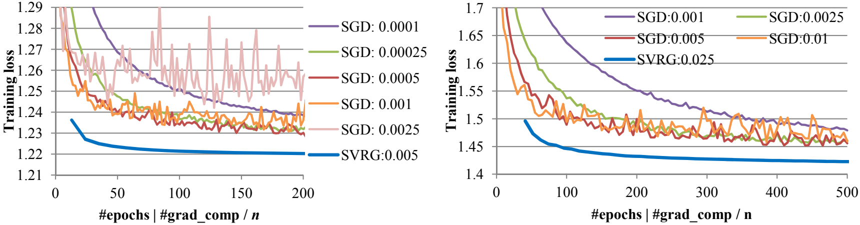
SVRG: variance





- Convex case (left): least squares on MNIST;
- Nonconvex case (right): neural nets on CIFAR-10.
- The numbers in the legends are learning rate

SVRG: convergence



- Convex case (left): least squares on MNIST;
- Nonconvex case (right): neural nets on CIFAR-10.
- The numbers in the legends are learning rate

Outline

- Background: stochastic gradient versus batch gradient
- SVRG (Stochastic Variance Reduced Gradient): control variates
- Importance sampling and stratefied sampling approaches
- SAGA (Stochastic Average Gradient Ameliore)

Importance Sampling

Objective function

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

Gradient

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

SGD (uniform sampling), uniform sample i from $\{1, \ldots, n\}$ and use

$$\nabla f_i(w)$$

SGD with importance sampling: sample i from $\{1, \ldots, n\}$ with probability $\{p_i\}$ ($\sum_i p_i = 1$), and use estimator

$$\mathbf{g}_i = (1/np_i)\, \nabla f_i(w)$$

Importance Sampled SGD

Importance weighted estimator $\mathbf{g}_i$ is an unbiased estimator of $\nabla f(w)$. Let U_i be an upperbound of $\|\nabla f_i(w)\|_2^2$:

$$U_i \geq \|\nabla f_i(w)\|_2^2.$$

Variance of $\{\mathbf{g}_i\}$ is

$$\frac{1}{n^2} \sum_i \|\nabla f_i(w) - np_i \nabla f(w)\|_2^2 / p_i \leq \frac{1}{n^2} \sum_i U_i / p_i.$$

Take optimal $p_i = \sqrt{U_i} / \sum_j \sqrt{U_j}$, the minimum variance is

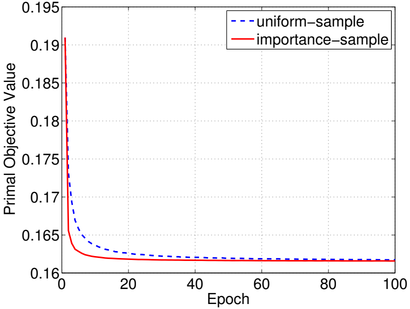
$$V(w) \leq (n^{-1} \sum_i \sqrt{U_i})^2.$$

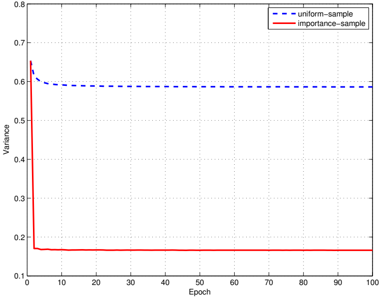
SGD with Importance Sampling

Procedure ISGD

Parameters gradient upperbounds $\{U_i\}$ and learning rate η
Initialize w_0, and $p_i = \sqrt{U_i} / \sum_j \sqrt{U_j}$
Iterate: for $t = 1, 2, \ldots, T$
Randomly pick $i_t \in \{1, \ldots, n\}$ according to $\{p_i\}$, and update weight
$$w_t = w_{t-1} - \frac{\eta}{p_{i_t}} \nabla f_{i_t}(w_{t-1})$$
end

SGD: uniform versus importance sampling





SVRG with Importance Sampling

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w).$$

L_i: smoothness of $f_i(w)$; λ: strong convexity of $f(w)$

Number of examples needed to achieve ϵ accuracy:

- With uniform sampling:

$$\tilde{O}((n + L/\lambda) \log(1/\epsilon)),$$

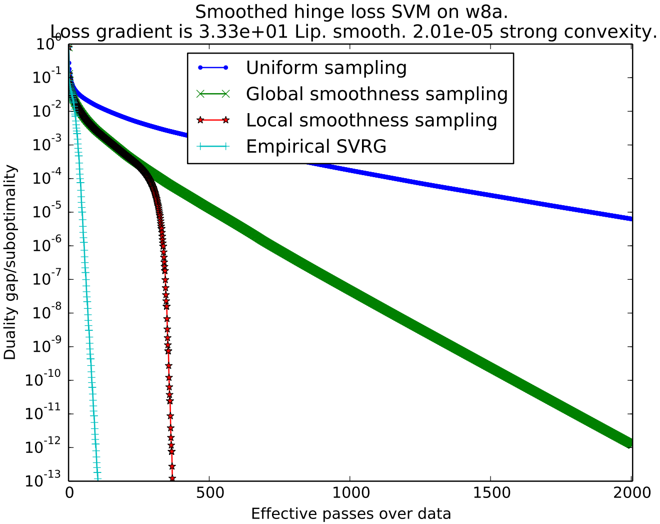
  where $L = \max_i L_i$

- With importance sampling: $p_i \propto L_i$

$$\tilde{O}((n + \bar{L}/\lambda) \log(1/\epsilon)),$$

  where $\bar{L} = n^{-1} \sum_{i=1}^{n} L_i$

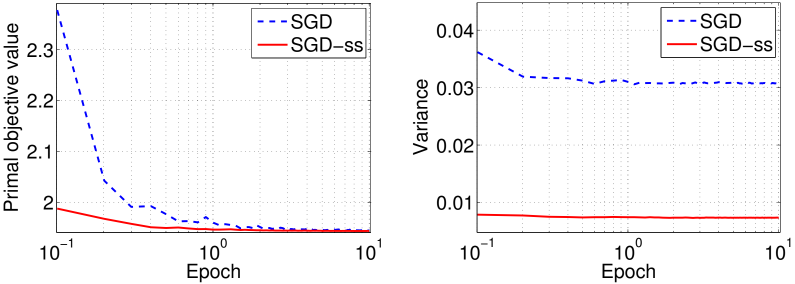
SVRG: importance sampling example



Stratified Sampling

- Can be applied to minibatch SGD for multiclass problem
- Algorithm
  - For each class: do k-means clustering separately to divide the sample into K groups
  - Stratified sampling of gradient with these groups

SGD: uniform versus stratified sampling



Summary of Modern Stochastic Optimization

Solve

$$w_* = \arg\min_w f(w) \qquad f(w) = \frac{1}{n}\sum_{i=1}^n f_i(w).$$

Optimization employs 1st order gradient

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

- sample of 1st order gradient leads to stochastic optimization
- Monte Carlo variance reduction leads to fast linear convergence
- Many many follow-up work

Outline

- Background: stochastic gradient versus batch gradient
- SVRG (Stochastic Variance Reduced Gradient): control variates
- Importance sampling and stratefied sampling approaches
- SAGA (Stochastic Average Gradient Ameliore)

Motivation

Solve

$$w_* = \arg\min_w f(w) \qquad f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w).$$

SGD with variance reduction via SVRG:

$$w_t = w_{t-1} - \eta_t \underbrace{\left[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w})\right]}_{\text{small variance}}.$$

Motivation

Solve

$$w_* = \arg\min_w f(w) \qquad f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w).$$

SGD with variance reduction via SVRG:

$$w_t = w_{t-1} - \eta_t \underbrace{[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w})]}_{\text{small variance}}.$$

Compute full gradient $\nabla f(\tilde{w})$ periodically at an intermediate $\tilde{w}$

Motivation

Solve

$$w_* = \arg\min_w f(w) \qquad f(w) = \frac{1}{n}\sum_{i=1}^n f_i(w).$$

SGD with variance reduction via SVRG:

$$w_t = w_{t-1} - \eta_t \underbrace{\left[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w})\right]}_{\text{small variance}}.$$

Compute full gradient $\nabla f(\tilde{w})$ periodically at an intermediate $\tilde{w}$

How to avoid computing $\nabla f(\tilde{w})$?
Answer: keeping previously calculated gradients.

Stochastic Average Gradient ameliore: SAGA

Initialize: $\tilde{\mathbf{g}}_i = \nabla f_i(w_0)$ and $\tilde{\mathbf{g}} = \frac{1}{n}\sum_{j=1}^n \tilde{\mathbf{g}}_j$

SAGA update rule: randomly select i, and

$$\begin{aligned} w_t =& w_{t-1} - \eta_t[\nabla f_i(w_{t-1}) - \tilde{\mathbf{g}}_i + \tilde{\mathbf{g}}] \\ \tilde{\mathbf{g}} =& \tilde{\mathbf{g}} + (\nabla f_i(w_{t-1}) - \tilde{\mathbf{g}}_i)/n \\ \tilde{\mathbf{g}}_i =& \nabla f_i(w_{t-1}) \end{aligned}$$

Equivalent to:

$$w_t = w_{t-1} - \eta_t \underbrace{\left[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}_i) + \frac{1}{n}\sum_{j=1}^n \nabla f_j(\tilde{w}_j)\right]}_{\text{small variance}} \qquad \tilde{w}_i = w_{t-1}.$$

Compare to SVRG:

$$w_t = w_{t-1} - \eta_t \underbrace{\left[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}) + \nabla f(\tilde{w})\right]}_{\text{small variance}}.$$

Variance Reduction

The gradient estimator of SAGA is unbiased:

$$\mathbf{E}\left[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}_i) + \frac{1}{n}\sum_{j=1}^{n}\nabla f_j(\tilde{w}_j)\right] = \nabla f(w_{t-1}).$$

Since $\tilde{w}_i \to w_*$, we have

$$\left[\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}_i) + \frac{1}{n}\sum_{j=1}^{n}\nabla f_j(\tilde{w}_j)\right] \to 0.$$

Therefore variance of the gradient estimator goes to zero.

Theory of SAGA

Similar to SVRG, we have fast convergence for SAGA.
Number of examples needed to achieve ϵ accuracy:

- Batch GD: $\tilde{O}(n \cdot L/\lambda \log(1/\epsilon))$
- SVRG: $\tilde{O}((n + L/\lambda) \log(1/\epsilon))$
- SAGA: $\tilde{O}((n + L/\lambda) \log(1/\epsilon))$

Assume L-smooth loss f_i and λ strongly convex objective function.

Summary

Optimization is important in big data machine learning
special structure: sum over data

- Traditional methods: gradient based batch algorithms
  - do not take advantage of special structure
- Recent progress: stochastic optimization with fast rate
  - employs Monte Carlo variance reduction