Stochastic and randomized convex optimization

(Incomplete version without transitions)

Adrien Taylor

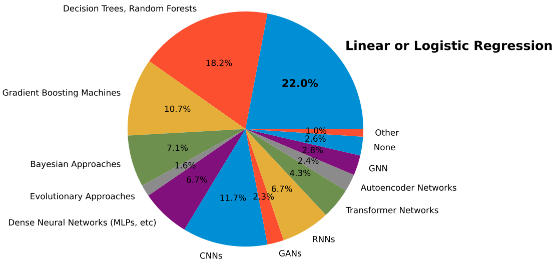
INRIA & École Normale Supérieure

Program for today

- Convex stochastic optimization,
- batch gradient methods,
- stochastic gradient descent,
- finite-sum algorithms,
- (randomized) coordinate methods,

... on a few running examples.

Which of the following ML algorithms do you use on a regular basis?



See Kaggle survey 2022.

Table of contents

Stochastic optimization problems

Table of contents

Motivation: supervised learning

- Input measurement $x \in \mathcal{X}$,
- output measurement $y \in \mathcal{Y}$,
- $(x, y) \sim \mathcal{D}$ with $\mathcal{D}$ unknown,
- training data: $\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ (i.i.d. $\sim \mathcal{D}$).

Often:

- $x \in \mathbb{R}^d$ and $y \in \{-1, 1\}$ (classification),
- or $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$ (regression).

We search a predictor function $p : \mathcal{X} \to \mathcal{Y}$.

Motivation: supervised learning

i	1	2	3	4	$\dots$	n
x_i						
y_i	1	1	-1	1		-1

Target: find $p : \mathcal{X} \to \mathcal{Y}$

$p($  $) \quad \to \quad 1$

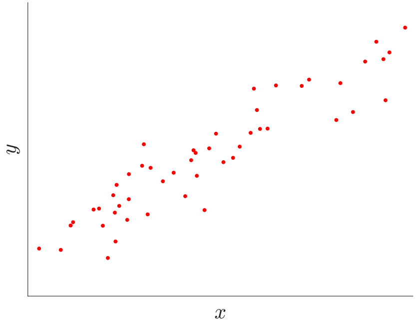
$p($  $) \quad \to \quad -1$

Motivation: supervised learning

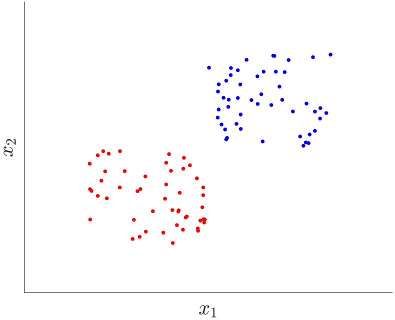
Often:

- $x \in \mathbb{R}^d$ and $y \in \{-1, 1\}$ (classification),
- or $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$ (regression).

We search a predictor $p : \mathcal{X} \to \mathcal{Y}$. How to construct good predictors?



Regression



Classification

Motivation: supervised learning

How to construct a good predictor?

- Pick a **loss function**: $\ell(p(x), y)$ to measure quality of $p(x) \approx y$.
- Examples:
  - $0 - 1$ loss: $\ell(p(x), y) = \mathbf{1}_{y \neq f(x)}$,
  - quadratic loss: $\ell(p(x), y) = |p(x) - y|^2$.

Risk function

- Risk measures the average loss over $\mathcal{D}$

$$\mathcal{R}(p) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell(p(x), y)\right].$$

- Examples:
  - $0 - 1$ risk: $\mathcal{R}(p) = \mathbb{P}\left(y \neq p(x)\right)$.
  - Quadratic risk: $\mathcal{R}(p) = \mathbb{E}\left[|y - p(x)|^2\right]$.

Motivation: supervised learning

Learning a predictor via decision variable θ

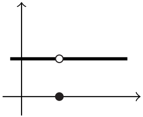
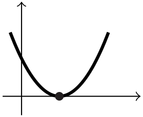
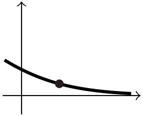
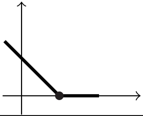
$$\underset{\theta \in \mathbb{R}^d}{\text{minimize}} \; \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(p_\theta(x), y)].$$

Here: $\mathcal{D}$ is distribution of datapoints $\xi = (x, y) \in \mathbb{R}^{d+1}$, and linear $p_\theta(x) = \langle \theta, x \rangle$. Examples:

- linear regression: $\ell(p_\theta(x), y) = (\langle \theta, x \rangle - y)^2$,
- logistic regression: $\ell(p_\theta(x), y) = \frac{\exp(y\langle \theta, x\rangle)}{1+\exp(y\langle \theta, x\rangle)}$,
- support vector machines: $\ell(p_\theta(x), y) = \max\{0, 1 - y\langle \theta, x\rangle\}$.

For all of those beyond pure linear models: see kernel versions.

Motivation: supervised learning

Name	$\ell(y_p, y)$	Graph $\ell(y_p, 1)$
$0-1$ loss	$\ell(y_p, y) = \begin{cases} 0 & \text{if } y_p = y \\ 1 & \text{if } y_p \neq y \end{cases}$	
quadratic loss	$\ell(y_p, y) = (y_p - y)^2$	
logistic loss	$\ell(y_p, y) = \log\left(1 + \exp(-y_p y)\right)$	
hinge loss	$\ell(y_p, y) = \max\{0, 1 - y_p y\}$	

Stochastic optimization framework

Learning a predictor via decision variable θ

$$\underset{\theta \in \mathbb{R}^d}{\text{minimize}}\ \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(p_{\theta}(x), y)] \triangleq \mathbb{E}_{\xi\sim\mathcal{D}}\left[f(\theta; \xi)\right].$$

Examples: $\mathcal{D}$ is distribution of datapoints $\xi = (x, y) \in \mathbb{R}^{d+1}$.

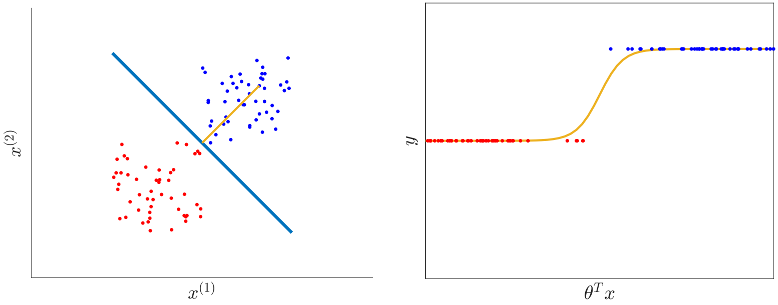
Often approached via **empirical risk minimization**:

$$\underset{\theta \in \mathbb{R}^d}{\text{minimize}}\ \frac{1}{n}\sum_{i=1}^{n} \ell(\langle \theta, x_i \rangle, y_i) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(\theta) \triangleq F(\theta).$$

Classification via logistic regression

We have $\mathcal{D}_n = \{(x_1, y_i), i = 1, \ldots, n\}$, with $y_i \in \{-1, 1\}$.

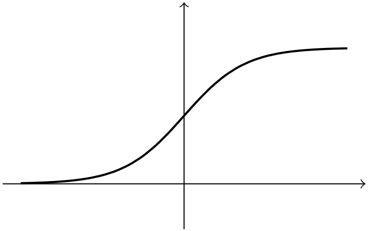
Objective: find θ such that $y_i\langle\theta, x_i\rangle \geqslant 0$ for all $i = 1, \ldots, n$.



Classification via logistic regression

- Pick sigmoid function

$\sigma(z) = \frac{1}{1+e^{-z}}$



- interpret: $\sigma(\langle \theta, x \rangle) = \mathbb{P}\{y = 1 | x\}$ and $\sigma(-\langle \theta, x \rangle) = \mathbb{P}\{y = -1 | x\}$
- with maximum likelihood / cross-entropy loss, yields **logistic regression**

$$\underset{\theta \in \mathbb{R}^d}{\text{minimize}} \ \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \langle \theta, x_i \rangle\right)\right).$$

- Convex! (How to show that?)

Table of contents

Plain gradient methods

Stochastic optimization

Empirical risk minimization as

$$\underset{\theta \in \mathbb{R}^d}{\text{minimize}} \left\{ F(\theta) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) \right\}.$$

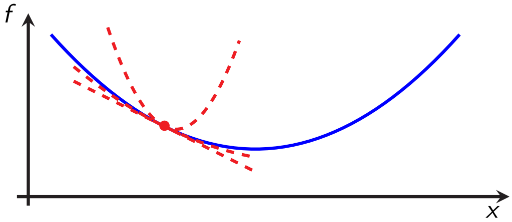
Starting assumptions (we will make variations around this):

- ◇ each $f_i(\cdot)$ has a Lipschitz gradient (constant L),
- ◇ each $f_i(\cdot)$ is strongly convex (constant μ).

When f_i twice continuously differentiable: $\mu I_d \preccurlyeq \nabla^2 f_i(\theta) \preccurlyeq L I_d$ for all $\theta \in \operatorname{dom} f$.

About the assumptions

A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex and L-smooth iff $\forall x, y \in \mathbb{R}^d$:



(1) (Convexity) $f(x) \geqslant f(y) + \langle \nabla f(y), x - y \rangle$,

(1b) (μ-strong convexity) $f(x) \geqslant f(y) + \langle \nabla f(y), x - y \rangle + \frac{\mu}{2}\|x - y\|_2^2$,

(2) (L-smoothness) $f(x) \leqslant f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2}\|x - y\|_2^2$,

(1&2) $f(x) \geqslant f(y) + \langle \nabla f(y), x - y \rangle + \frac{1}{2L}\|\nabla f(x) - \nabla f(y)\|_2^2 + \frac{\mu}{2(1-\mu/L)}\|x - y - \frac{1}{L}(\nabla f(x) - \nabla f(y))\|_2^2$,

(1&2b) $\langle \nabla f(x) - \nabla f(y), x - y \rangle \geqslant \frac{1}{L+\mu}\|\nabla f(x) - \nabla f(y)\|_2^2 + \frac{\mu L}{L+\mu}\|x - y\|_2^2$.

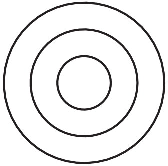
About the assumptions

First-order optimization: condition number $\kappa = \frac{L}{\mu} \geqslant 1$ discriminates "easy" vs. "hard".

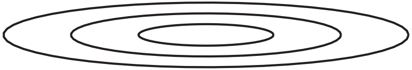
- Smoothness L given by curvature in direction with fastest variation,
- Strong convexity given by curvature in direction with slowest variation.

Insights from level curves:

very well conditioned problem ($\kappa \approx 1$):



more poorly conditioned one ($\kappa \gg 1$):



Examples

◇ Regularized least squares (Ridge regression): $f_i(\theta) = (\langle \theta, x_i \rangle - y_i)^2 + \frac{\lambda}{2}\|\theta\|_2^2$.
Hessian: $\nabla^2 f_i(\theta) = 2x_i x_i^T + \lambda I_d$. Hence: $L = 2 \max_{1 \leqslant i \leqslant n} \|x_i\|_2^2 + \lambda$ and $\mu = \lambda$.

◇ Regularized logistic regression: $f_i(\theta) = \log(1 + \exp(-y_i \langle \theta, x_i \rangle)) + \frac{\lambda}{2}\|\theta\|_2^2$, we have:

$$\nabla f_i(\theta) = \frac{-y_i x_i}{1+\exp(y_i \langle \theta, x_i \rangle)} + \lambda\theta, \quad \nabla^2 f_i(\theta) = \frac{\exp(y_i \langle \theta, x_i \rangle)}{(1+\exp(y_i \langle \theta, x_i \rangle))^2} x_i x_i^T + \lambda I_d.$$

Therefore, for any z with $\|z\|_2 = 1$: (hint: use $\frac{e^u}{(1+e^u)^2} \leqslant \frac{1}{4}$)

$$z^T \nabla^2 f_i(\theta) z = z^T x_i x_i^T z \frac{\exp(y_i \langle \theta, x_i \rangle)}{(1+\exp(y_i \langle \theta, x_i \rangle))^2} + \lambda I_d \|z\|_2^2 \leqslant z^T \left(\tfrac{1}{4} x_i x_i^T + \lambda I_d\right) z$$

Hence $L = \frac{1}{4} \max_{1 \leqslant i \leqslant n} \|x_i\|_2^2 + \lambda$ and $\mu = \lambda$.

Plain gradient descent

Algorithm: Plain gradient descent

Set $\theta^0 \in \mathbb{R}^d$, $\alpha > 0$.

for $t = 0, 1, \ldots$ **do**

$\quad \theta^{t+1} = \theta^t - \frac{\alpha}{n} \sum_{i=1}^n \nabla f_i(\theta^t)$

end

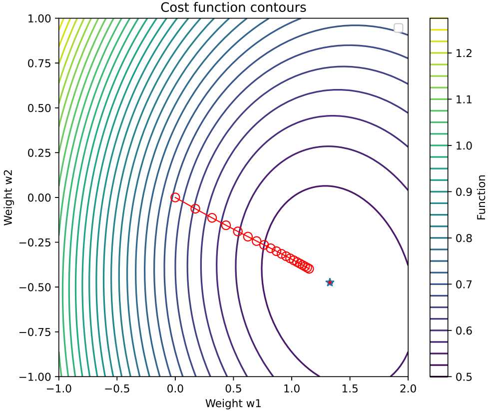
In this context, for $\alpha = \frac{1}{L}$:

$$F(\theta^t) - F(\theta^\star) \leqslant \min\left\{\frac{1}{t}, \left(1 - \frac{1}{\kappa}\right)^t\right\} \frac{L\|\theta^0 - \theta^\star\|_2^2}{2}.$$

$$\|\theta^t - \theta^\star\|_2^2 \leqslant \left(1 - \frac{1}{\kappa}\right)^t \|\theta^0 - \theta^\star\|_2^2.$$

Plain GD

Gradient descent ($\alpha = \frac{1}{L}$):[1]



[1] Logistic regression problem: "fourclass" dataset from LIBSVM $(n, d) = (862, 2)$.

Classical GD convergence analysis

General idea: studying a single iteration is simpler. Need recursable bounds.

One can prove $V^{t+1} \leqslant V^t$ for all θ^t, $\theta^{t+1} = \theta^t - \frac{1}{L}\nabla F(\theta^t)$ and L-smooth convex function, with

$$V^t \triangleq V(A_t, \theta^t) \triangleq A_t(F(\theta^t) - F(\theta^\star)) + \frac{L}{2}\|\theta^t - \theta^\star\|_2^2$$

and $A_{t+1} \leqslant A_t + 1$.

Why is this nice?

$$A_t \left(F(\theta^t) - F(\theta^\star)\right) \leqslant V^t \leqslant V^{t-1} \leqslant \ldots \leqslant V^0,$$

so $F(\theta^t) - F(\theta^\star) \leqslant \frac{V^0}{A_t} = \frac{L\|\theta^0 - \theta^\star\|_2^2}{2A_t}$ when choosing $A_0 = 0$.

GD: recall convergence analysis — a simple case

For GD, a simple bound to prove:

$$\|\theta^{t+1} - \theta^\star\|_2^2 = \|\theta^t - \theta^\star\|_2^2 - 2\alpha\langle \nabla F(\theta^t), \theta^t - \theta^\star \rangle + \alpha^2 \|\nabla F(\theta^t)\|_2^2$$

$\Big\downarrow$ Inequality (1&2b)

$$\leqslant \left(1 - \frac{2\alpha L\mu}{L+\mu}\right) \|\theta^t - \theta^\star\|_2^2 + \alpha \left(\alpha - \frac{2}{L+\mu}\right) \|\nabla F(\theta^t)\|_2^2$$

$\Big\downarrow$ if $0 \leqslant \alpha \leqslant \frac{2}{L+\mu}$

$$\leqslant (1 - \alpha\mu)^2 \|\theta^t - \theta^\star\|_2^2.$$

Plain accelerated gradient descent

Algorithm: Plain acceleration for ERM

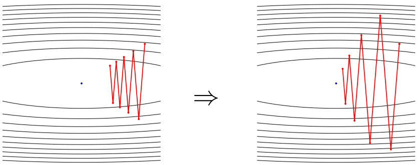
Set $\theta^0 = \tilde{\theta}^0 \in \mathbb{R}^d$, $\alpha, \{\beta_t\} > 0$.

for $t = 0, 1, \ldots, T-1$ **do**

$\quad \theta^{t+1} = \tilde{\theta}^t - \frac{\alpha}{n} \sum_{i=1}^n \nabla f_i(\tilde{\theta}^t)$

$\quad \tilde{\theta}^{t+1} = \theta^{t+1} + \beta_t(\theta^{t+1} - \theta^t)$

end



In this context, for appropriate choices of (α, β): (for some $C > 0$)

$$F(\theta^t) - F(\theta^\star) \leqslant \min\left\{\frac{2}{t^2}, \left(1 - \sqrt{\frac{1}{\kappa}}\right)^t\right\} L\|\theta^0 - \theta^\star\|_2^2.$$

$$\|\theta^t - \theta^\star\|_2^2 \leqslant C\left(1 - \sqrt{\frac{1}{\kappa}}\right)^t \|\theta^0 - \theta^\star\|_2^2,$$

using similar proof patterns.[2]

[2] See, e.g., d'Aspremont, Scieur, T (2021). "Acceleration methods."

Classical AGD convergence analysis

General idea: studying a single iteration is simpler. Need recursable bounds.

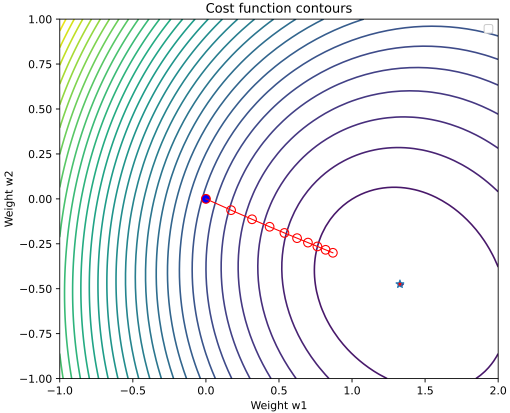
One can prove $V^{t+1} \leqslant V^t$ with

$$V^t \triangleq V(A_t, \theta^t) \triangleq A_t(F(\theta^t) - F(\theta^\star)) + \frac{L}{2}\|\hat{\theta}^t - \theta^\star\|_2^2$$

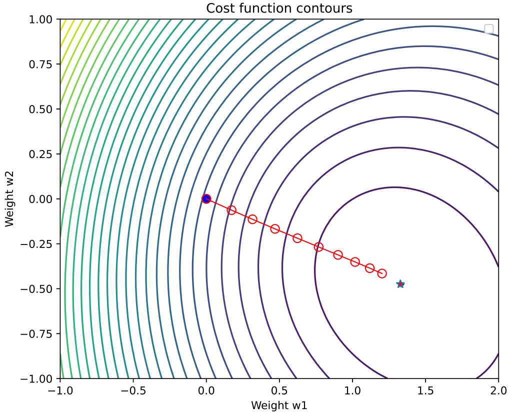
and $A_t \approx t^2$ when $A_0 = 0$.

In short: all coefficient choices made for greedily making A_t large.

GD vs. AGD



Vanilla GD



Accelerated GD

Plain gradients for ERM – takeaways

Were we exploiting what we can?

- ◇ Momentum? $\rightarrow$ accelerated convergence rates.
- ◇ Adaptative step-size selection? $\rightarrow$ backtracking line-search, online estimation of L,...

But when far away from solution: single $\nabla f_i(\theta^t)$ is already informative!

$\rightarrow$ useful to evaluate the full batch?

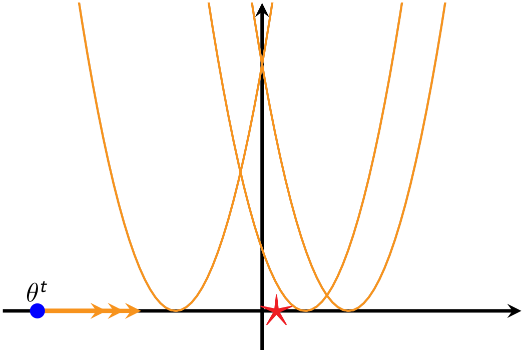


Table of contents

Stochastic gradient methods

Stochastic gradient descent (SGD)

$$\underset{\theta \in \mathbb{R}^d}{\text{minimize}} \left\{ F(\theta) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) \right\}.$$

Algorithm: SGD, constant step-size

Set $\theta^0 \in \mathbb{R}^d$, $\alpha > 0$

for $t = 0, 1, \ldots, T-1$ **do**

  sample $i_t \sim \mathcal{U}[[1, n]]$

  $\theta^{t+1} = \theta^t - \alpha \nabla f_{i_t}(\theta^t)$

end

- very simple to implement!
- very cheap iteration.

Stochastic gradient descent (SGD)

Observations:

- $\nabla f_i(\theta)$ (with $i \sim \mathcal{U}[[1, n]]$) is unbiased estimate of gradient (more later):

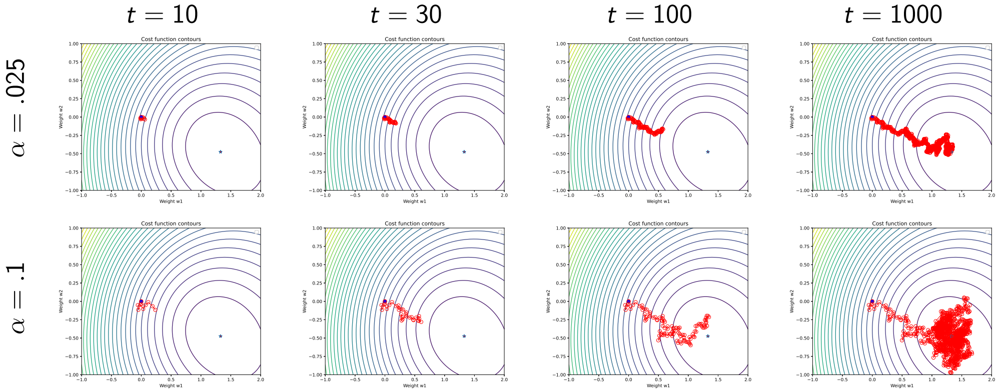
$$\mathbb{E}_i[\nabla f_i(\theta)] = \nabla F(\theta).$$

- What if gradients $\nabla f_i(\theta)$'s are very different?
- What if gradients $\nabla f_i(\theta)$'s are very similar?

  $\rightarrow$ variance of gradient estimation drives behavior of SGD!

Stochastic gradient descent: empirical behavior

Short step sizes[3]

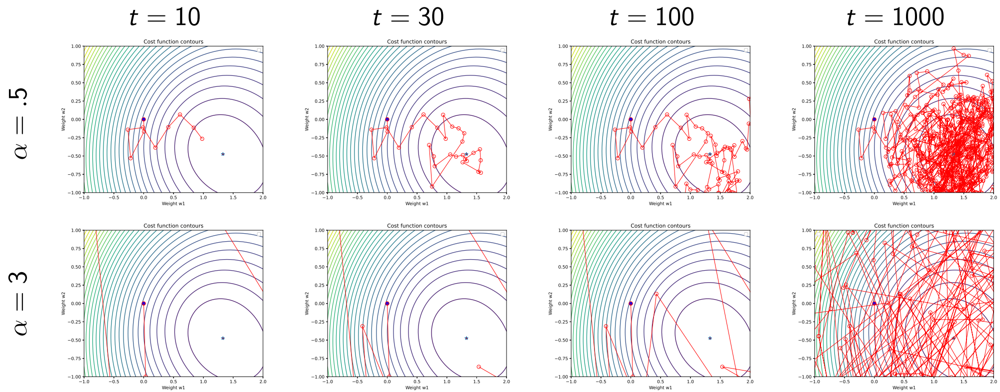


$\rightarrow$ very slow to converge & relatively accurate.

[3]Logistic regression problem: "fourclass" dataset from LIBSVM $(n, d) = (862, 2)$.

Stochastic gradient descent: empirical behavior

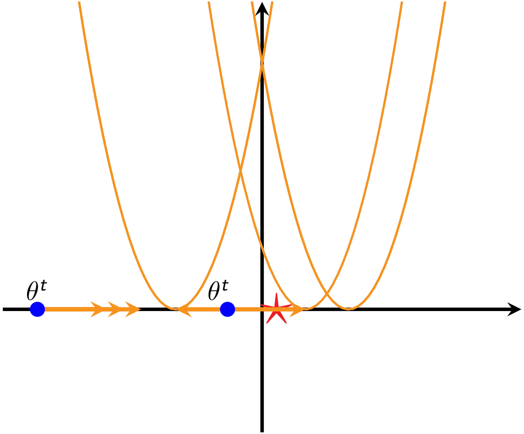
Larger step sizes



→ faster to reach "stationnary behavior" (forget about initial conditions) & not accurate.

→ we want: initially large α, then short α on the long run.

Stochastic gradient descent: empirical behavior



Morally, two extreme regimes:

- "error due to initial conditions" dominates $\rightarrow$ stochastic gradients are very informative
- "error due to noise" dominates $\rightarrow$ need to accomodate noise.

Mitigating noise via step-size schedulers

Naive scheduler:[4]

Algorithm: SGD, naive step-size scheduler

Set $\theta^0 \in \mathbb{R}^d$, $\alpha^0 > 0$, $c \in (0,1)$, $K \in \mathbb{N}$

for $t = 0, 1, \ldots, T-1$ **do**

  sample $i_t \sim \mathcal{U}[[1,n]]$

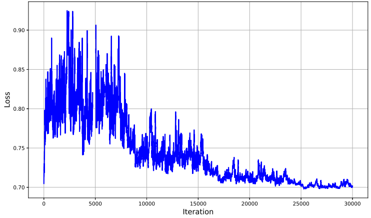
  $\theta^{t+1} = \theta^t - \alpha \nabla f_{i_t}(\theta^t)$

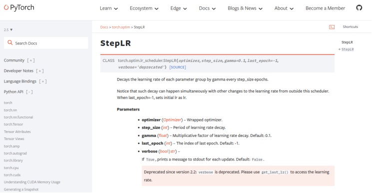
  if $\mathbf{mod}(t+1, K) = 0$ **then**

    $\alpha = c\alpha$

  end

end




PyTorch — Learn, Ecosystem, Edge, Docs, Blogs & News, About, Become a Member

Docs > torch.optim > StepLR

StepLR

CLASS torch.optim.lr_scheduler.StepLR(*optimizer*, *step_size*, *gamma=0.1*, *last_epoch=-1*, *verbose='deprecated'*) [SOURCE]

Decays the learning rate of each parameter group by gamma every step_size epochs.

Notice that such decay can happen simultaneously with other changes to the learning rate from outside this scheduler. When last_epoch=-1, sets initial lr as lr.

Parameters

- **optimizer** (*Optimizer*) – Wrapped optimizer.
- **step_size** (*int*) – Period of learning rate decay.
- **gamma** (*float*) – Multiplicative factor of learning rate decay. Default: 0.1.
- **last_epoch** (*int*) – The index of last epoch. Default: -1.
- **verbose** (*bool* | *str*) –
  If `True`, prints a message to stdout for each update. Default: `False`.

Deprecated since version 2.2: `verbose` is deprecated. Please use `get_last_lr()` to access the learning rate.

[4] Experiment with the "mushroom" dataset from LIBSVM $(n, d) = (8124, 112)$.

Mitigating noise via minibatches

Algorithm: minibatch-SGD, constant step-size

Set $\theta^0 \in \mathbb{R}^d$, $\alpha > 0$, $b \in \mathbb{N}$.

for $t = 0, 1, \dots, T-1$ **do**

  sample $i_t^{(1)}, \dots, i_t^{(b)} \sim \mathcal{U}[[1, n]]$, $\mathcal{I}_t = \{i_t^{(1)}, \dots, i_t^{(b)}\}$

  $\theta^{t+1} = \theta^t - \frac{\alpha}{b} \sum_{i \in \mathcal{I}_t} \nabla f_i(\theta^t)$

end

b	name	gradient estimate	computational cost
1	(pure) SGD	$\nabla f_{i_t}(\theta^t)$ with $i_t \in \mathcal{U}[[1, n]]$	$O(d)$
$1 < b < n$	minibatch SGD	$\sum_{i \in \mathcal{I}_t} \nabla f_i(\theta^t)$, $\lvert\mathcal{I}_t\rvert = b$	$O(bd)$
$b = n$	full batch/plain GD	$\sum_{i=1}^{n} \nabla f_i(\theta^t)$	$O(nd)$

- Commonly: pick $b = 2^a$ ($a = 5, 6, \dots$) to benefit from parallelization on GPU/CPU.
- For theory, focus on $b = 1$.

Stochastic gradient descent – unbiasedness

If batch chosen uniformly at random & independently from past ⇒ unbiased gradient estimate.

- ◇ pure SGD: pick $i_t \in \mathcal{U}[[1, n]]$ independently from past iterates then

$$\mathbb{E}\left[\nabla f_{i_t}(\theta^t) | \theta^t\right] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\theta^t) \triangleq \nabla F(\theta^t).$$

- ◇ Minibatch SGD: pick $\mathcal{I}_t$ uniformly at random in $\{1, \ldots, n\}$ (with or without resampling) & independently from past iterates then

$$\mathbb{E}\left[\frac{1}{b}\sum_{i \in \mathcal{I}_t} \nabla f_i(\theta^t) \middle| \theta^t\right] = \frac{1}{bn}\sum_{i=1}^{b}\sum_{j=1}^{n} \nabla f_j(\theta^t) = \nabla F(\theta^t).$$

unbiased but noisy estimations. Effect of b on variance?

Stochastic gradient descent – bounds

$$\underset{\theta \in \mathbb{R}^d}{\text{minimize}} \left\{ F(\theta) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) \right\}.$$

Classical assumptions (variations on this theme in what follows)

- ⋄ each $f_i(\cdot)$ is L-smooth and μ-strongly convex,
- ⋄ bounded variance at $\theta^\star$: $\mathbb{E}_i \left[\|\nabla F(\theta^\star) - \nabla f_i(\theta^\star)\|_2^2 \right] = \mathbb{E}_i \left[\|\nabla f_i(\theta^\star)\|_2^2 \right] \leqslant \sigma^2$.

One can show: (with $\alpha = 1/L$ for simplicity)

$$\mathbb{E}_i \left[\|\theta^{t+1} - \theta^\star\|_2^2 \middle| \theta^t \right] \leqslant \left(1 - \tfrac{\mu}{L}\right)^2 \|\theta^t - \theta^\star\|_2^2 + \tfrac{2\sigma^2}{L^2}.$$

Stochastic gradient descent – bounds

Proof. reformulate the inequality (due to smoothness and strong convexity), namely (1&2b):

$$0 \geqslant \mathbb{E}_{i_t}\Big[-\langle \nabla f_{i_t}(\theta^t) - \nabla f_{i_t}(\theta^\star), \theta^t - \theta^\star\rangle + \frac{1}{L}\|\nabla f_{i_t}(\theta^t) - \nabla f_{i_t}(\theta^\star)\|_2^2$$
$$+ \frac{\mu}{1-\mu/L}\|\theta^t - \theta^\star - \tfrac{1}{L}(\nabla f_{i_t}(\theta^t) - \nabla f_{i_t}(\theta^\star))\|_2^2\Big]$$

multiplied by $2\alpha(1-\alpha\mu) \geqslant 0$ (with $0 \leqslant \alpha \leqslant 1/L$) as

$$\begin{aligned}\mathbb{E}_{i_t}[\|\theta^{t+1} - \theta^\star\|_2^2] \leqslant & (1-\alpha\mu)^2\|\theta^t - \theta^\star\|_2^2 + \tfrac{2\alpha^2(1-\alpha\mu)}{2-\alpha(L+\mu)}\mathbb{E}_{i_t}[\|\nabla f_{i_t}(\theta^\star)\|_2^2] \\ & - \tfrac{\alpha(2-\alpha(L+\mu))}{L-\mu}\mathbb{E}_{i_t}\|\mu(\theta^\star - \theta^t) + \nabla f_{i_t}(\theta^t) + 2\tfrac{1-\alpha\mu}{\alpha(L+\mu)-2}\nabla f_{i_t}(\theta^\star)\|_2^2 \\ \leqslant & (1-\alpha\mu)^2\|\theta^t - \theta^\star\|_2^2 + \tfrac{2\alpha^2(1-\alpha\mu)}{2-\alpha(L+\mu)}\sigma^2\end{aligned}$$

(using unbiasedness: $\mathbb{E}_{i_t}[\langle \nabla f_{i_t}(\theta^\star), \theta^t\rangle] = \mathbb{E}_{i_t}[\langle \nabla f_{i_t}(\theta^\star), \theta^\star\rangle] = 0$).

Desired result by evaluating $\alpha \leftarrow \frac{1}{L}$.

Stochastic gradient descent – bounds

By chaining inequalities, we arrive to $(t \geqslant 0)$

$$\begin{aligned}
\mathbb{E}_i\left[\|\theta^t - \theta^\star\|_2^2 \middle| \theta^0\right] &\leqslant \left(1 - \tfrac{\mu}{L}\right)^{2t} \|\theta^0 - \theta^\star\|_2^2 + \tfrac{2\sigma^2}{L^2} \sum_{i=0}^{t-1} \left(1 - \tfrac{\mu}{L}\right)^{2i} \\
&\leqslant \left(1 - \tfrac{\mu}{L}\right)^{2t} \|\theta^0 - \theta^\star\|_2^2 + \tfrac{2\sigma^2}{L^2} \left(\tfrac{L}{\mu} - \tfrac{L}{\mu}\left(1 - \tfrac{\mu}{L}\right)^t\right) \\
&\leqslant \left(1 - \tfrac{\mu}{L}\right)^{2t} \|\theta^0 - \theta^\star\|_2^2 + \tfrac{2\sigma^2}{L\mu}.
\end{aligned}$$

Hence, for SGD with constant $\alpha = \frac{1}{L}$ reaches

$$\mathbb{E}_i\left[\|\theta^t - \theta^\star\|_2^2 \middle| \theta^0\right] \leqslant \left(1 - \tfrac{\mu}{L}\right)^{2t} \|\theta^0 - \theta^\star\|_2^2 + \tfrac{2\sigma^2}{L\mu}.$$

$\rightarrow$ convergence to a ball around $\theta^\star$.

Stochastic gradient descent – bounds & takeaways

Theory and experience agree on:

- ◇ small step-size: slowly forget initial condition; convergence to a small ball around solution.
- ◇ Large step-size: better forget initial conditions; convergence to a larger ball.

Can we do better?

- ◇ averaging,
- ◇ decreasing step-sizes (step-size schedules),
- ◇ decrease variance (minibatching).

Here: let's simplify the assumptions for this study.

SGD with bounded gradients

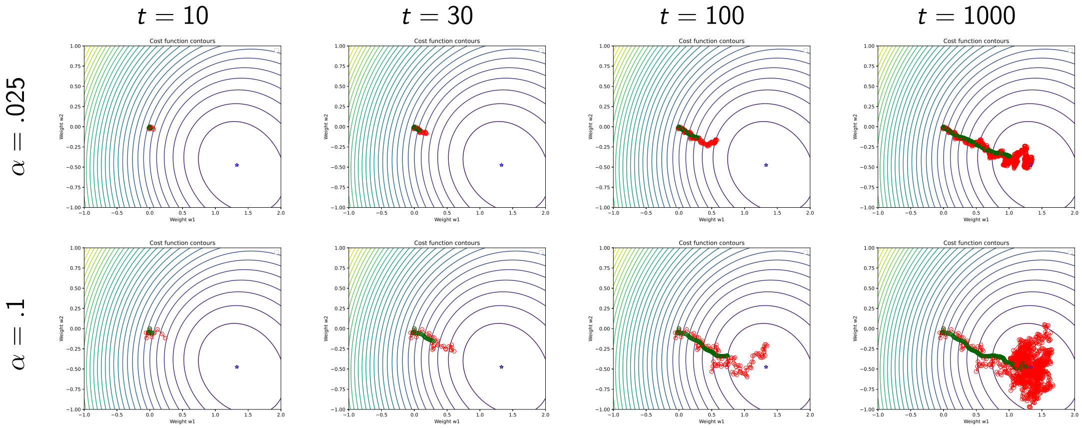
$$\underset{\theta \in \mathbb{R}^d}{\text{minimize}} \left\{ F(\theta) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) \right\}.$$

Simplifying assumptions here:

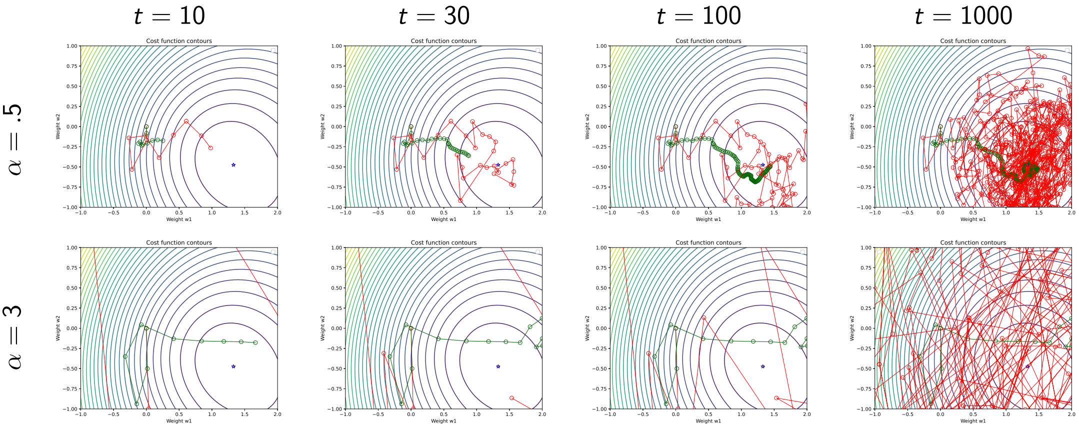
- ◇ each $f_i(\cdot)$ is convex
- ◇ bounded stochastic gradients $\mathbb{E}\left[\|\nabla f_i(\theta)\|_2^2\right] \leqslant M^2$.

(one can get refined analyses using smoothness/strong convexity).

SGD: averaging



SGD: averaging



SGD with bounded gradients

Suppose $\|\theta^0 - \theta^\star\|_2 \leqslant R$ for some $\theta^0 \in \mathbb{R}^d$, and $\mathbb{E}_i[\|\nabla f_i(\theta^t)\|_2^2] \leqslant M^2$, then

$$\mathbb{E}[F(\bar{\theta}^T)] - F(\theta^\star) \leqslant \frac{R^2 + M^2 \sum_{t=0}^T \alpha_t^2}{2 \sum_{t=0}^T \alpha_t},$$

with $\bar{\theta}^T = \frac{1}{T+1} \sum_{t=0}^T \theta^t$ (Polyak-Ruppert averaging).

- Proof essentially similar to that of the subgradient method.
- Rates are similar (but in expectation).

SGD with bounded gradients

Proof. Define $r_t = \|\theta^t - \theta^\star\|_2$, we have:

$$r_{t+1}^2 = r_t^2 - 2\alpha_t \langle \nabla f_{i_t}(\theta^t), \theta^t - \theta^\star \rangle + \alpha_t^2 \|\nabla f_{i_t}(\theta^t)\|_2^2.$$

Taking expectations and using convexity and indepence of i_t and θ^t

$$\begin{aligned}
\mathbb{E}[r_{t+1}^2] &\leqslant \mathbb{E}[r_t^2] - 2\alpha_t \mathbb{E}\left[\langle \nabla f_{i_t}(\theta^t), \theta^t - \theta^\star \rangle\right] + \alpha_t^2 \mathbb{E}[\|\nabla f_{i_t}(\theta^t)\|_2^2] \\
&\leqslant \mathbb{E}[r_t^2] - 2\alpha_t \mathbb{E}\left[\langle \mathbb{E}\left[\nabla f_{i_t}(\theta^t) \mid \theta^t\right], \theta^t - \theta^\star \rangle\right] + \alpha_t^2 M^2 \\
&\leqslant \mathbb{E}[r_t^2] - 2\alpha_t (\mathbb{E}[F(\theta^t)] - F(\theta^\star)) + \alpha_t^2 M^2.
\end{aligned}$$

(with abusive drops of conditional expectations, and using $\alpha_t \geqslant 0$).

Summing up and using convexity of $F(\cdot)$, we reach the desired

$$r_0^2 + M^2 \sum_{t=0}^{T} \alpha_t^2 \geqslant \sum_{t=0}^{T} \alpha_t (\mathbb{E}[F(\theta^t)] - F(\theta^\star)) \geqslant 2 \left(\sum_{t=0}^{T} \alpha_t \right) (\mathbb{E}[F(\bar{\theta}^T)] - F(\theta^\star)).$$

SGD with bounded gradients

Suppose $\|\theta^0 - \theta^\star\|_2 \leqslant R$ for some $\theta^0 \in \mathbb{R}^d$, and $\mathbb{E}_i[\|\nabla f_i(\theta^t)\|_2^2] \leqslant M^2$, then

$$\mathbb{E}[F(\bar{\theta}^T)] - F(\theta^\star) \leqslant \frac{R^2 + M^2 \sum_{t=0}^T \alpha_t^2}{2\sum_{t=0}^T \alpha_t},$$

with $\bar{\theta}^T = \frac{1}{T+1}\sum_{t=0}^T \theta^t$ (Polyak-Ruppert averaging).

Examples:

- Pick $\alpha_t = \frac{\alpha}{M}$: $F(\bar{\theta}^T) - F(\theta^\star) \leqslant \frac{M\|\theta^0-\theta^\star\|_2^2 + (T+1)\alpha^2 M}{2(T+1)\alpha} = \frac{M\|\theta^0-\theta^\star\|_2^2}{2(T+1)\alpha} + \frac{\alpha M}{2}$
- Pick $\alpha_t = \frac{\|\theta^0-\theta^\star\|_2}{M\sqrt{T+1}}$ (constant step-size depending on horizon T) then

$$F(\bar{\theta}^T) - F(\theta^\star) \leqslant \frac{M\|\theta^0 - \theta^\star\|_2}{\sqrt{T+1}}.$$

SGD with bounded gradients

Suppose $\|\theta^0 - \theta^\star\|_2 \leqslant R$ for some $\theta^0 \in \mathbb{R}^d$, and $\mathbb{E}_i[\|\nabla f_i(\theta^t)\|_2^2] \leqslant M^2$, then

$$\mathbb{E}[F(\bar{\theta}^T)] - F(\theta^\star) \leqslant \frac{R^2 + M^2 \sum_{t=0}^T \alpha_t^2}{2\sum_{t=0}^T \alpha_t},$$

with $\bar{\theta}^T = \frac{1}{T+1}\sum_{t=0}^T \theta^t$ (Polyak-Ruppert averaging).

- Square summable but not summable, e.g.: $\alpha_t = \frac{\alpha}{M(t+1)}$

$$\mathbb{E}[F(\bar{\theta}^T)] - F(\theta^\star) \leqslant M\frac{\|\theta^0 - \theta^\star\|_2^2 + \alpha(1+\alpha)}{2\alpha\log(T+2)},$$

- Non-summable diminishing, example $\alpha_t = \frac{\alpha}{M\sqrt{t+1}}$ then

$$\mathbb{E}[F(\bar{\theta}^T)] - F(\theta^\star) \leqslant M\frac{\|\theta^0 - \theta^\star\|_2^2 + \alpha^2(1+\log(T+2))}{4\alpha\sqrt{T+2}}.$$

Summing up: rough computational estimates for smooth convex minimization

Computational cost to reach $F(\theta) - F(\theta^\star) \leqslant \epsilon$?

Method	Cost per iteration	# iterations	Computational cost
GD	$O(nd)$	$O\left(\frac{1}{\epsilon}\right)$	$O\left(\frac{nd}{\epsilon}\right)$
AGD	$O(nd)$	$O\left(\frac{1}{\sqrt{\epsilon}}\right)$	$O\left(\frac{nd}{\sqrt{\epsilon}}\right)$
SGD	$O(d)$	$O\left(\frac{1}{\epsilon^2}\right)$	$O\left(\frac{d}{\epsilon^2}\right)$

→ SGD: total complexity does not depend on n.

→ For any $\epsilon > 0$, total complexity of SGD better than that of (A)GD if n large enough.

What target accuracy? Total computational cost:

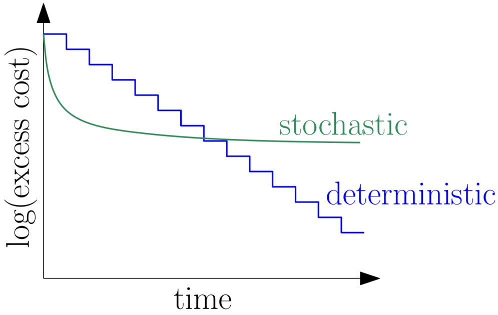
ϵ	GD	AGD	SGD
$1/\sqrt{n}$	$O(n^{3/2}d)$	$O(n^{5/4}d)$	$O(nd)$
$1/n$	$O(n^2d)$	$O(n^{3/2}d)$	$O(n^2d)$
$1/n^2$	$O(n^3d)$	$O(n^2d)$	$O(n^4d)$

◇ Low/moderate accuracy wrt. n: SGD better.

◇ Moderate/high accuracy wrt. n: (A)GD better.

◇ ML: typically low/moderate accuracy.

GD vs. SGD



Example: smooth convex optimization:

- ◇ from low to moderate accuracy requirements: SGD better.
- ◇ from moderate to high accuracy requirements: (A)GD better.

Momentum & stochasticity

Algorithm: Stochastic accelerated gradient

Set $\theta^0 = \tilde{\theta}^0 \in \mathbb{R}^d$, $\{\alpha_t\}, \{\beta_t\} > 0$.

for $t = 0, 1, \ldots, T-1$ **do**

  sample $i_t \sim \mathcal{U}[[1, n]]$

  $\theta^{t+1} = \tilde{\theta}^t - \alpha_t \nabla f_{i_t}(\tilde{\theta}^t)$

  $\tilde{\theta}^{t+1} = \theta^{t+1} + \beta_t(\theta^{t+1} - \theta^t)$

end

- Classical choices: momentum $\rightarrow$ critical noise accumulation!
- Can be mitigated via appropriate scheduling (but essentially no rate improvement).[5,6]

[5] Devolder (2011). "Stochastic first order methods in smooth convex optimization."

[6] Aybat, Fallah, Gurbuzbalaban, Ozdaglar (2019). "A universally optimal multistage accelerated stochastic gradient method."

Table of contents

Finite sums

Finite sum optimization

$$\underset{\theta \in \mathbb{R}^d}{\text{minimize}} \left\{ F(\theta) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) \right\}.$$

So far:

- ⋄ full batch (A)GD: accurate (but expensive) estimate of $\nabla F(\theta^t)$

  useless accuracy when far from solution,

  convergence to a solution.

- ⋄ SGD: cheap (but noisy) estimate of $\nabla F(\theta^t)$

  when far from solution: $\nabla f_i(\theta^t)$ essentially points the right direction

  when close to solution: direction is not good.

Can we get best of both world? $\rightarrow$ "variance reduction" techniques!

Finite sum optimization

$$\underset{\theta \in \mathbb{R}^d}{\text{minimize}} \left\{ F(\theta) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(\theta) \right\}.$$

Running assumptions:

- each $f_i(\cdot)$ has a Lipschitz gradient (constant L),
- each $f_i(\cdot)$ is strongly convex (constant μ).

Most methods below apply more generally to (but not discussed further):

- each $f_i(\cdot)$ has a Lipschitz gradient (constant L),
- $F(\cdot)$ is strongly convex (constant μ).

Exploiting finite sums

Instead of using $\nabla f_{i_t}(\theta^t) \approx \nabla F(\theta^t)$:

- build running estimate $g^t \approx \nabla F(\theta^t)$,
- update estimate using new information $\nabla f_{i_t}(\theta^t)$.

Target/hopes: unbiased $g^t \approx \nabla F(\theta^t)$ with $\|g^t\|_2^2 \to 0$ (as $\theta^t \to \theta^\star$).

Exploiting finite sums

Recall gradient descent $\theta^{t+1} = \theta^t - \alpha \nabla F(\theta^t)$. Equivalently:

$$\theta^{t+1} = \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \left\{ F(\theta^t) + \langle \nabla F(\theta^t), \theta - \theta^t \rangle + \frac{2}{\alpha} \|\theta - \theta^t\|_2^2 \right\}$$

essentially: regularized linear approximation. What about the stochastic setting? Proposal:

$$\theta^{t+1} = \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \left\{ \frac{1}{n} \left(\sum_{i=1}^{n} f_i(\phi_i^t) + \langle \nabla f_i(\phi_i^t), \theta - \phi_i^t \rangle \right) + \frac{2}{\alpha} \|\theta - \theta^t\|_2^2 \right\}.$$

How to update ϕ_i^t's?

SAG: Stochastic Average Gradient[7]

Algorithm: SAG

Set $\theta^0 \in \mathbb{R}^d$, $\alpha > 0$, $\phi_i^0 = \theta^0$ and $g_i = \nabla f_i(\theta^0)$ for all $i \in [[1, n]]$.

for $t = 0, 1, \ldots, T-1$ **do**

- sample $i_t \sim \mathcal{U}[[1, n]]$
- $\phi_i^t = \phi_{i-1}^t$ for all $i \neq i_t$
- $\phi_{i_t}^t = \theta^t$ (save evaluated point for ∇f_{i_t})
- $g_{i_t} = \nabla f_{i_t}(\theta^t)$ (upgrade gradient of f_{i_t})
- $g^t = \frac{1}{n}\sum_{i=1}^n g_i$ (estimate of $\nabla F(\theta^t)$)
- $\theta^{t+1} = \theta^t - \alpha g^t$

end

👍 simple to implement.

👎 stores $d \times n$ matrix $[g_1, g_2, \ldots, g_n]$.

? more efficient computation of g^t?

? do we really need to store matrix for LR & LS?

[7] Schmidt, Le Roux, Bach, (2013). "Minimizing finite sums with the stochastic average gradient."

SAG: observations

Observations:

- ◇ Gradient estimate?

$$\nabla F(\theta^t) \approx g^t = \frac{1}{n}\sum_{i=1}^{n} g_i.$$

- ◇ Unbiased?

$$\mathbb{E}_{i_t}[g^t \mid \theta^t, g^{t-1}] =$$

$\rightarrow$

- ◇ Can we do something about storage? $\rightarrow$ for linear models, yes (later).

Stochastic average gradient (SAG)

Let f_i $(i = 1, \ldots, n)$ be L-smooth and μ-strongly convex, and let $\alpha = \frac{1}{16L}$, we have

$$\mathbb{E}[F(\theta^t)] - F(\theta^\star) \leqslant \left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8n}\right\}\right)^t C_0 \leqslant \exp\left(-\min\left\{\frac{\mu t}{16L}, \frac{t}{8n}\right\}\right) C_0,$$

with $C_0 = \frac{3}{2}\left(F(\theta^0) - F(\theta^\star) + \frac{4L}{n}\|\theta^0 - \theta^\star\|_2^2 + \frac{\sigma^2}{16L}\right)$.

Complexity? $\mathbb{E}[F(\theta)] - F(\theta^\star) \leqslant \epsilon$ in t at most

$$\exp\left(-\min\left\{\frac{\mu t}{16L}, \frac{t}{8n}\right\}\right) C_0 \leqslant \epsilon \Leftrightarrow t \geqslant \max\left\{16\frac{L}{\mu}, 8n\right\} \log\left(\frac{C_0}{\epsilon}\right)$$

Result actually not easy to prove. Proof relies on computer-aided verification steps.

SAGA: Stochastic Average Gradient "Amélioré"[8]

Algorithm: SAGA

Set $\theta^0 \in \mathbb{R}^d$, $\alpha > 0$, $\phi_i^0 = \theta^0$ and $g_i = \nabla f_i(\theta^0)$ for all $i \in [[1, n]]$.

for $t = 0, 1, \dots, T-1$ **do**

- sample $i_t \sim \mathcal{U}[[1, n]]$
- $\phi_i^t = \phi_{i-1}^t$ for all $i \neq i_t$
- $\phi_{i_t}^t = \theta^t$ (save evaluated point for ∇f_{i_t})
- $g^t = \nabla f_{i_t}(\theta^t) - g_{i_t} + \frac{1}{n}\sum_{i=1}^n g_i$ (estimate of $\nabla F(\theta^t)$)
- $g_{i_t} = \nabla f_{i_t}(\theta^t)$ (upgrade gradient of f_{i_t})
- $\theta^{t+1} = \theta^t - \alpha g^t$

end

? Differences with SAG?

[8] Defazio, Bach, Lacoste-Julien (2014). "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives."

SAGA: observations

Observations:

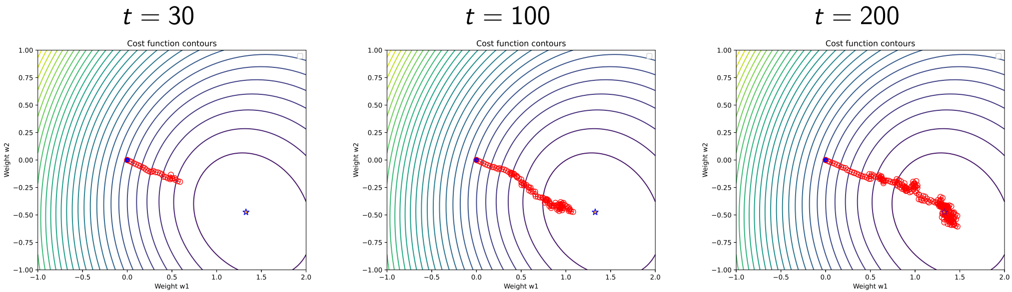
- Gradient estimate?

$$\nabla F(\theta^t) \approx g^t = \nabla f_{i_t}(\theta^t) - g_{i_t} + \frac{1}{n}\sum_{i=1}^{n} g_i.$$

- Unbiased?

$$\mathbb{E}_{i_t}[g^t \mid \theta^t, g^{t-1}] =$$

SAGA: observations



SAGA: Stochastic Average Gradient “Amélioré”

Let f_i $(i = 1, \ldots, n)$ be L-smooth and μ-strongly convex, and let $\alpha = \frac{1}{3L}$, we have

$$\mathbb{E}\left[\|\theta^t - \theta^\star\|_2^2\right] \leqslant \left(1 - \min\left\{\frac{1}{4n}, \frac{\mu}{3L}\right\}\right)^t C_0$$

with $C_0 = \left[\|\theta^0 - \theta^\star\|_2^2 + \frac{2n}{3L}\left[F\left(\theta^0\right) - \left\langle \nabla F\left(\theta^*\right), \theta^0 - \theta^\star \right\rangle - F\left(\theta^\star\right)\right]\right]$.

Similar conclusions as for SAG: we reach $\|\theta - \theta^\star\|_2^2 \leqslant \epsilon$ in at most

$$O\left(\max\left\{\kappa, n\right\} \log\left(\frac{1}{\epsilon}\right)\right).$$

Analysis of SAGA is **considerably simpler** than that of SAG.

SAGA: Stochastic Average Gradient “Amélioré”

Proof overview. Show that (Lyapunov analysis): We have

$$\mathbb{E}\left[V^{t+1}\right] \leqslant \left(1 - \min\left\{\frac{1}{4n}, \frac{\mu}{3L}\right\}\right) V^k$$

with

$$V^t \triangleq V\left(\theta^t, \{\phi_i^t\}_{i=1}^n\right) \triangleq \frac{1}{n}\sum_{i=1}^n \left[f_i\left(\phi_i^t\right) - f_i\left(\theta^*\right) - \langle \nabla f_i\left(\theta^\star\right), \phi_i^t - \theta^\star\rangle\right] + c\left\|\theta^t - \theta^\star\right\|_2^2$$

and $c = \frac{1}{2\alpha(1-\alpha\mu)n}$.

Details: see arXiv.

SAGA for linear models

If learning problem can be written as

$$\underset{\theta \in \mathbb{R}^d}{\text{minimize}} \frac{1}{n} \sum_{i=1}^{n} h_i(\langle \theta, x_i \rangle) + \frac{\lambda}{2} \|\theta\|_2^2,$$

we have: $\nabla f_i(\theta) = h_i'(\langle \theta, x_i \rangle) x_i + \lambda \theta$. Hence, for each data point store only $\beta_i = h_i'(\langle \theta^t, x_i \rangle)$.

SAGA for ℓ_2-regularized linear models

Algorithm: SAGA for linear models

Set $\theta^0 \in \mathbb{R}^d$, $\lambda \geqslant 0$, $\alpha > 0$, $\beta_i = h_i'(\langle \theta^0, x_i \rangle)$ for all $i \in [[1, n]]$.

for $t = 0, 1, \ldots, T-1$ **do**

  sample $i_t \sim \mathcal{U}[[1, n]]$

  $g^t =$

  $\beta_{i_t} =$

  $\theta^{t+1} =$

end

- ? Storage
- ? Stochastic gradient
- ? Gradient estimate?

👍 No storage issue!

Stochastic variance reduced method gradient (SVRG)[9]

Algorithm: SVRG

Set $\tilde{\theta}^0 \in \mathbb{R}^d$, $\alpha > 0$, $m \in \mathbb{N}$.

for $s = 0, 1, \ldots, T$ **do**

$G^s = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\tilde{\theta}^s)$

$\theta^0 = \tilde{\theta}^s$

for $t = 0, 1, \ldots, m-1$ **do**

sample $i_t \sim \mathcal{U}[[1, n]]$

$g^t = \nabla f_{i_t}(\theta^t) - \nabla f_{i_t}(\tilde{\theta}^s) + G^s$

$\theta^{t+1} = \theta^t - \alpha g^t$

end

sample $j_s \sim \mathcal{U}[[1, m]]$

$\tilde{\theta}^{s+1} = \theta^{j_s}$

end

- differences
- no need to store $d \times n$ matrix $[g_1, g_2, \ldots, g_n]$.
- need to tune m (inner loop).

[9] Johnson, Zhang (2013). "Accelerating stochastic gradient descent using predictive variance reduction."

SVRG: observations

◇ Gradient estimate?

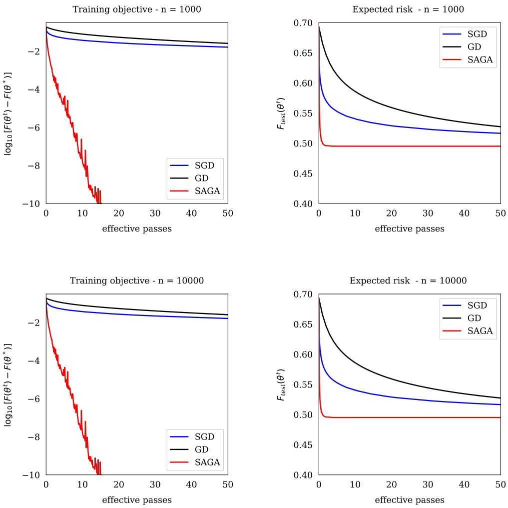
$$\nabla F(\theta^t) \approx g^t = \nabla f_{i_t}(\theta^t) - \nabla f_{i_t}(\tilde{\theta}^s) + \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\theta}^s).$$

◇ Unbiasedness?

$$\mathbb{E}_{i_t}\left[g^t \mid \theta^t, \tilde{\theta}^s\right] =$$

$\rightarrow$

Stochastic vs. variance reduction vs. full batch methods[10]



[10]Bach (2024). "Learning theory from first principles."

Exploiting finite sums – momentum

Recall template for accelerated gradient descent (iterates $\{(\theta^t, \phi^t, \lambda^t)\}_{t=0,1,\dots}$

$$\phi^t = (1-\tau_t)\,\theta^t + \tau_t \lambda^t$$

$$\lambda^{t+1} = \operatorname*{argmin}_{\lambda \in \mathbb{R}^d} \left\{ \sum_{i=0}^{t} \left[(A_{i+1} - A_i)\left(F(\phi^i) + \langle \nabla F(\phi^i), \theta - \theta^i \rangle \right) \right] + \frac{2}{\alpha} \|\lambda - \phi^t\|_2^2 \right\}$$

$$\theta^{t+1} = (1-\tilde{\tau}_t)\,\theta^t + \tilde{\tau}_t \lambda^{t+1}$$

... similarly: based on regularized (weighted) linear approximations of $F(\cdot)$

(with growing sequence $\{A_t\}_{t=0,1,\dots}$ and some $\{(\tau_k, \tilde{\tau}_k)\}_{t=0,1,\dots}$ for convex combinations).

Momentum versions

A few momentum variations exist. Among the simplest ones:[11]

Algorithm: SAGA with Sampled Negative Momentum

Set $\theta^0 \in \mathbb{R}^d$, $\alpha, \tau > 0$, $\phi_i^0 = \nabla f_i(\theta^0)$ for all $i \in [[1, n]]$.

for $t = 0, 1, \dots, T-1$ **do**

- sample $i_t \sim \mathcal{U}[[1, n]]$
- $\tilde{\theta}^t = \tau\theta^t + (1-\tau)\phi_{i_t}^t$
- $g^t = \nabla f_{i_t}(\tilde{\theta}^t) - \nabla f_{i_t}(\phi_{i_t}^t) + \frac{1}{n}\sum_{i=1}^n \nabla f_i(\phi_i^t)$
- $\theta^{t+1} = \theta^t - \alpha g^t$
- sample $j_t \sim \mathcal{U}[[1, n]]$
- $\phi_{j_t}^{t+1} = \tau\theta^{t+1} + (1-\tau)\phi_{j_t}^t$

end

? differences

? # gradient evaluations

[11] Zhou et al. (2019). "Direct acceleration of SAGA using sampled negative momentum."

Takeaways from variance reduction

- ◇ Finite-sums methods use only one stochastic gradient per iteration and converge linearly on strongly convex functions.
- ◇ Choice of fixed (nondecreasing) step-size possible.
- ◇ SAGA only needs to know the smoothness parameter, but requires storing n past stochastic gradients in general (but not for linear classifier).
- ◇ SVRG only has $O(d)$ storage in general, but requires full gradient computations every so often. Has an extra "number of inner iterations" parameter.

SAG/SAGA in scikit-learn $\longrightarrow$

The choice of the algorithm depends on the penalty chosen and on (multinomial) multiclass support:

solver	penalty	multinomial multiclass
'lbfgs'	'l2', None	yes
'liblinear'	'l1', 'l2'	no
'newton-cg'	'l2', None	yes
'newton-cholesky'	'l2', None	no
'sag'	'l2', None	yes
'saga'	'elasticnet', 'l1', 'l2', None	yes

Summing up: rough computational cost estimates

Method	# iterations	# gradient queries
GD	$O\left(\kappa \log\left(\frac{1}{\epsilon}\right)\right)$	$O\left(n\kappa \log\left(\frac{1}{\epsilon}\right)\right)$
AGD	$O\left(\sqrt{\kappa} \log\left(\frac{1}{\epsilon}\right)\right)$	$O\left(n\sqrt{\kappa} \log\left(\frac{1}{\epsilon}\right)\right)$
SAG/SAGA/SVRG	$O\left(\max\{n, \kappa\} \log\left(\frac{1}{\epsilon}\right)\right)$	$O\left(\max\{n, \kappa\} \log\left(\frac{1}{\epsilon}\right)\right)$
Katyushia[12]/MiG[13]/SSNM[14]/Pt-SAGA[15]	$O\left(\max\{n, \sqrt{n\kappa}\} \log\left(\frac{1}{\epsilon}\right)\right)$	$O\left(\max\{n, \sqrt{n\kappa}\} \log\left(\frac{1}{\epsilon}\right)\right)$

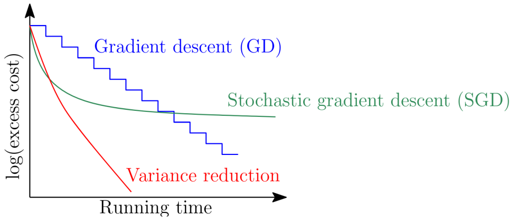
So: finite-sum methods benefit from momentum when $n \ll \kappa$. That is:

- $\max\{n, \kappa\} = \kappa \to$ computational complexities of SAG/SAGA/SVRG is $O\left(\kappa \log\left(\frac{1}{\epsilon}\right)\right)$.
- $\max\{n, \sqrt{n\kappa}\} = \sqrt{n\kappa} \to$ computational complexities of momentum variants is

$$O\left(\sqrt{n\kappa} \log\left(\frac{1}{\epsilon}\right)\right) \ll O\left(\kappa \log\left(\frac{1}{\epsilon}\right)\right).$$

[12] Allen-Zhu (2017). "Katyusha: The first direct acceleration of stochastic gradient methods."
[13] Zhou, Shang, Cheng (2018). "A simple stochastic variance reduced algorithm with fast convergence rates."
[14] Zhou et al. (2019). "Direct acceleration of SAGA using sampled negative momentum."
[15] Defazio (2016). "A simple practical accelerated method for finite sums."

Stochastic vs. variance reduction vs. full batch methods



To experiment with those:

SAG/SAGA   Point-SAGA   Boosted variants

Table of contents

Popular stochastic algorithms

Practical improvements

Practical improvements:

- ◇ adapt to observations,
- ◇ adapt componentwise,
- ◇ momentum,
- ◇ different step-size schedules.

Generally, either

- ◇ no existing analysis,
- ◇ or extremely technical.

Optimizers in Pytorch $\rightarrow$

Algorithms

`Adadelta`	Implements Adadelta algorithm.
`Adafactor`	Implements Adafactor algorithm.
`Adagrad`	Implements Adagrad algorithm.
`Adam`	Implements Adam algorithm.
`AdamW`	Implements AdamW algorithm.
`SparseAdam`	SparseAdam implements a masked version of the Adam algorithm suitable for sparse gradients.
`Adamax`	Implements Adamax algorithm (a variant of Adam based on infinity norm).
`ASGD`	Implements Averaged Stochastic Gradient Descent.

Adagrad[16,17]

Adagrad[16] (update all components j):

$$g^t = \nabla f_{i_t}\left(\theta^{t-1}\right)$$

$$v^t_{(j)} = v^{t-1}_{(j)} + \left(g^t_{(j)}\right)^2$$

$$\theta^t_{(j)} = \theta^{t-1}_{(j)} - \frac{\alpha}{\sqrt{\epsilon + v^t_{(j)}}} g^t_{(j)}$$

For certain parameter choices:[17]

$$\mathbb{E}\left[\|\nabla F(\theta^t)\|_2^2\right] = O\left(\frac{1}{\sqrt{t}}\right)$$

for smooth objectives.

Adagrad in Pytorch →

Adagrad

CLASS torch.optim.Adagrad(*params*, *lr=0.01*, *lr_decay=0*, *weight_decay=0*, *initial_accumulator_value=0*, *eps=1e-10*, *foreach=None*, *, *maximize=False*, *differentiable=False*, *fused=None*) [SOURCE]

Implements Adagrad algorithm.

input : γ (lr), θ_0 (params), $f(\theta)$ (objective), λ (weight decay), τ (initial accumulator value), η (lr decay)
initialize : $state\_sum_0 \leftarrow \tau$

for $t = 1$ **to** $\ldots$ **do**
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
$\tilde{\gamma} \leftarrow \gamma/(1 + (t-1)\eta)$
if $\lambda \neq 0$
$g_t \leftarrow g_t + \lambda \theta_{t-1}$
$state\_sum_t \leftarrow state\_sum_{t-1} + g_t^2$
$\theta_t \leftarrow \theta_{t-1} - \tilde{\gamma} \frac{g_t}{\sqrt{state\_sum_t} + \epsilon}$

return θ_t

[16] Duchi, Hazan, Singer (2011). "Adaptive subgradient methods for online learning and stochastic optimization."
[17] Défossez, Bottou, Bach, Usunier (2020). "A simple convergence proof of Adam and Adagrad."

Adam[18,19]

Adam[18] (update all components j):

$$
\begin{aligned}
g^t &= \nabla f_{i_t}\left(\theta^{t-1}\right) \\
m^t &= \beta_1 m^{t-1} + (1-\beta_1) g^t \\
v^t_{(j)} &= \beta_2 v^{t-1}_{(j)} + (1-\beta_2)\left(g^t_{(j)}\right)^2 \\
\theta^t_{(j)} &= \theta^{t-1}_{(j)} - \frac{\alpha}{\sqrt{\epsilon + v^t_{(j)}}} g^t_{(j)}
\end{aligned}
$$

For certain parameter choices:[19]

$$
\mathbb{E}\left[\|\nabla F(\theta^t)\|_2^2\right] = O\left(\frac{\log t}{\sqrt{t}}\right)
$$

for smooth objectives.

Adam in Pytorch $\rightarrow$

Adam

CLASS torch.optim.Adam(*params*, *lr=0.001*, *betas=(0.9, 0.999)*, *eps=1e-08*, *weight_decay=0*, *amsgrad=False*, *, *foreach=None*, *maximize=False*, *capturable=False*, *differentiable=False*, *fused=None*) [SOURCE]

Implements Adam algorithm.

input : γ (lr), β_1, β_2 (betas), θ_0 (params), $f(\theta)$ (objective)
λ (weight decay), *amsgrad*, *maximize*
initialize : $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment), $\widehat{v_0}^{max} \leftarrow 0$

for $t = 1$ **to** $\ldots$ **do**
  if *maximize* :
    $g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$
  else
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  if $\lambda \neq 0$
    $g_t \leftarrow g_t + \lambda\theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
  $\widehat{m_t} \leftarrow m_t / (1-\beta_1^t)$
  $\widehat{v_t} \leftarrow v_t / (1-\beta_2^t)$
  if *amsgrad*
    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / (\sqrt{\widehat{v_t}^{max}} + \epsilon)$
  else
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / (\sqrt{\widehat{v_t}} + \epsilon)$

return θ_t

[18] Kingma, Ba (2014). "Adam: A method for stochastic optimization."
[19] Défossez, Bottou, Bach, Usunier (2020). "A simple convergence proof of Adam and Adagrad."

Randomized coordinate descent

(One possible) motivation: back to supervised learning

$$\underset{\theta \in \mathbb{R}^d}{\text{minimize}} \ \frac{1}{n}\sum_{i=1}^{n} h_i(\langle \theta, x_i \rangle) + \frac{\lambda}{2}\|\theta\|_2^2.$$

What did we do with stochastic methods?

→ update parameter estimation, one sample at a time.

→ Other ways to do that? One possibility: artificially augmented problem:

$$\underset{\substack{\theta \in \mathbb{R}^d \\ \beta_1,\dots,\beta_n \in \mathbb{R}}}{\text{minimize}} \ \frac{1}{n}\sum_{i=1}^{n} h_i(\beta_i) + \frac{\lambda}{2}\|\theta\|_2^2 \quad \text{s.t. } \beta_i = \langle \theta, x_i \rangle \text{ for } i = 1, \dots, n.$$

Introduce dual variables $\omega_1, \dots, \omega_n$; Lagrangian dual is:

→ Use coordinate-based methods on dual.[20]

[20] Shalev-Shwartz, Zhang (2013). "Stochastic dual coordinate ascent methods for regularized loss minimization."

Randomized block-coordinate methods

$$\underset{\omega \in \mathbb{R}^d}{\text{minimize}}\ D(\omega)$$

where f is L-smooth and convex. Decompose decision space into n blocks:

$$\omega = \sum_{i=1}^{n} \mathbf{U}_i \omega \quad \text{with} \quad [\mathbf{U}_1\, \mathbf{U}_2\, \ldots\, \mathbf{U}_n] = I_d.$$

Algorithm: RBCD

Set $\omega^0 \in \mathbb{R}^d$, $\alpha > 0$.

for $t = 0, 1, \ldots, T-1$ **do**

  sample $i_t \sim \mathcal{U}[[1, n]]$

  $\omega^{t+1} = \omega^t - \alpha \mathbf{U}_{i_t} \nabla D(\omega^t)$

end

update rule corresponds to

- ◇ if $i \neq i_t$: $\mathbf{U}_i \omega^{t+1} = \mathbf{U}_i \omega^t$
- ◇ if $i = i_t$: $\omega^{t+1}_{(i_t)} = \omega^t_{(i_t)} - \alpha \nabla_{i_t} D(\omega^t)$.

Example: what $\{\mathbf{U}_i\}_{i=1}^n$ corresponds to single coordinate decomposition?

Randomized block-coordinate methods: convergence

Let $\omega^t \in \mathbb{R}^d$, $\omega^{t+1} = x^t - \alpha \mathbf{U}_{i_t} \nabla D(\omega^t)$ with $\alpha \in (0, \frac{1}{L}]$, $i_t \sim \mathcal{U}[[1, n]]$. One can show:

$$A_{t+1}\mathbb{E}[D(\omega^{t+1}) - D(\omega^\star)] + \frac{L}{2}\mathbb{E}[\|\omega^{t+1} - \omega^\star\|_2^2] \leqslant A_t(D(\omega^t) - D(\omega^\star)) + \frac{L}{2}\|\omega^t - \omega^\star\|_2^2$$

for any $A_t \geqslant 1$ and $A_{t+1} = A_t + \frac{\alpha L}{n}$.

- Many results, variants, etc. Easily fall into additional technical difficulties.
- More conventional to assume Lipschitz by block (simpler to compute and more aggressive step size strategies), but this result is simple.
- Guarantee: $\mathbb{E}[D(\omega^t) - D(\omega^\star)] \leqslant \frac{n}{t}\left(D(\omega^0) - D(\omega^\star) + \frac{L}{2}\|\omega^0 - \omega^\star\|_2^2\right)$ with $\alpha = \frac{1}{L}$.
- Recall gradient descent: $\mathbb{E}[D(\omega^t) - D(\omega^\star)] \leqslant \frac{L}{2t}\|\omega^0 - \omega^\star\|_2^2$.

Randomized block-coordinate methods – improvement

$$\underset{\omega\in\mathbb{R}^d}{\text{minimize}}\, D(\omega)$$

where f is convex. Decompose decision space into n blocks: $\omega = \sum_{i=1}^{n} \mathbf{U}_i\omega$ with $[\mathbf{U}_1\, \mathbf{U}_2\, \ldots\, \mathbf{U}_n] = I_d$. Further assume $\forall i \in \{1, 2, \ldots, n\}$:

$$D(x + \mathbf{U}_i\Delta) \leqslant D(x) + \langle \nabla D(x), \mathbf{U_i}\Delta\rangle + \frac{L_i}{2}\|\mathbf{U}_i\Delta\|_2^2.$$

Algorithm: RBCD

Set $\omega^0 \in \mathbb{R}^d$.

for $t = 0, 1, \ldots, T-1$ **do**

  sample $i_t \sim \mathcal{U}[[1, n]]$

  $\omega^{t+1} = \omega^t - \frac{1}{L_{i_t}}\mathbf{U}_{i_t}\nabla D(\omega^t)$

end

- L_i usually simpler to compute than L
- L_i often (much) smaller than L.

On RBCD

Questions:

1. Is the gradient estimate $\mathbf{U}_{i_t} \nabla f(\omega^t)$ biased?
2. Consider the quadratic problem

$$\underset{\omega \in \mathbb{R}^d}{\text{minimize}} \ \frac{1}{2}\omega^T A \omega$$

   and the decomposition $\mathbf{U}_i = e_i$ (unit vector whose ith component is one).
   - What do the L_i's ($i = 1, \ldots, d$) correspond to?
   - Show that the global Lipschitz constant L satisfies: $\max_{1 \leqslant i \leqslant d} L_i \leqslant L \leqslant \sum_{i=1}^d L_i$.
   - Consider the matrix $A = c\,\mathbf{1}\mathbf{1}^T$. What are L_i's? and L?

Randomized block-coordinate methods – improvement

Denote $\|\omega\|_{\{L_i\}}^2 \triangleq \sum_{i=1}^n L_i \|\mathbf{U}_i \omega\|_2^2$.

Let $\omega^t \in \mathbb{R}^d$, $\omega^{t+1} = \omega^t - \frac{1}{L_{i_t}} \mathbf{U}_{i_t} \nabla F(\omega^t)$ with $i_t \sim \mathcal{U}\{1, \ldots, n\}$. It holds:

$$A_{t+1}\mathbb{E}[D(\omega^{t+1}) - D(\omega^\star)] + \frac{1}{2}\mathbb{E}[\|\omega^{t+1} - \omega^\star\|_{\{L_i\}}^2] \leqslant A_t(D(\omega^t) - D(\omega^\star)) + \frac{1}{2}\|\omega^t - \omega^\star\|_{\{L_i\}}^2$$

for any $A_t \geqslant 1$ and $A_{t+1} = A_t + \frac{1}{n}$.

- ◇ Usually simpler to compute and allows for larger step-sizes.
- ◇ More conventional to assume Lipschitz by block.
- ◇ Guarantee: $\mathbb{E}[D(\omega^t) - D(\omega^\star)] \leqslant \frac{n}{t}\left(D(\omega^0) - D(\omega^\star) + \frac{1}{2}\|\omega^0 - \omega^\star\|_{\{L_i\}}^2\right)$.
- ◇ Possible to extend results to linear convergence (strong convexity-type assumptions).[21]

[21] See, e.g., Nesterov (2012). "Efficiency of coordinate descent methods on huge-scale optimization problems."

Randomized block-coordinate methods

Proof sketch. Weighted sum of inequalities:

- ◇ convexity of F between ω^t and $\omega^\star$, with weight $A_{t+1} - A_t$:

$$0 \geqslant D(\omega^t) - D(\omega^\star) + \langle \nabla D(\omega^t), \omega^\star - \omega^t \rangle,$$

- ◇ expectation of the "block" descent lemma with weight A_{t+1}:

$$\mathbb{E}_{i_t}[D(\omega^{t+1})] \leqslant D(\omega^t) - \mathbb{E}_i \left[\tfrac{1}{2L_i} \|\mathbf{U}_i \nabla D(\omega^t)\|_2^2 \right].$$

Weighted sum yields:

$$\begin{aligned}\mathbb{E}_{i_t}[V^{t+1}] \leqslant V^t &- \tfrac{A_{t+1}-1}{2n} \|\nabla D(\omega^t)\|^2_{\{L_i^{-1}\}} \\ &+ \left(A_{t+1} - A_t - \tfrac{1}{n}\right) \langle \nabla D(\omega^t), \omega^t - \omega^\star \rangle,\end{aligned}$$

with $V^t = A_t(D(\omega^t) - D(\omega^\star)) + \frac{1}{2}\|\omega^t - \omega^\star\|^2_{\{L_i\}}$.

Example – support vector machine

Soft-margin support vector machine (SVM):

$$\underset{\theta\in\mathbb{R}^d}{\text{minimize}}\ \tfrac{1}{2}\|\theta\|_2^2 + \nu\sum_{i=1}^{n}\max\{0, 1 - y_i\langle\theta, x_i\rangle\}$$

Reformulate:

$$\begin{aligned}\underset{\theta\in\mathbb{R}^d,\, s\in\mathbb{R}^n}{\text{minimize}}\ & \tfrac{1}{2}\|\theta\|_2^2 + \nu\sum_{i=1}^{n} s_i \\ \text{s.t. } & y_i\langle\theta, x_i\rangle \geqslant 1 - s_i \\ & s_i \geqslant 0\end{aligned}$$

Lagrange dual?

Example – support vector machine

Denote by $X = [y_1 x_1 \,|\, y_2 x_2 \,|\, \dots \,|\, y_n x_n] \in \mathbb{R}^{d \times n}$. Lagrange duality yields:

$$\underset{0 \leqslant \lambda \leqslant \nu}{\text{maximize}} \left\{ D(\lambda) \triangleq -\frac{1}{2}\lambda^T X^T X \lambda + \sum_{i=1}^{n} \lambda_i \right\}$$

and a natural estimate of the primal variable $\theta = \sum_{i=1}^{n} \lambda_i x_i y_i = X\lambda$. Algorithm?

Algorithm: RBCD for dual SVM

Set $\lambda^0 \in \mathbb{R}^n$.

for $t = 0, 1, \dots, T-1$ **do**

  sample $i_t \sim \mathcal{U}[[1, n]]$

  $\lambda_{(i_t)}^{t+1} = \mathrm{Proj}_{[0,\alpha]} \left[\omega_{(i_t)}^t - \frac{1}{L_{i_t}} \nabla_{i_t} D(\lambda^t) \right]$

end

- $\lambda_{(i)}^t$ denotes ith component.
- Projection OK within BCD for separable constraints.
- L_i's?
- Exact 1-D optimization.

Example – support vector machine

Denote by $X = [y_1 x_1 \,|\, y_2 x_2 \,|\, \ldots \,|\, y_n x_n] \in \mathbb{R}^{d \times n}$. Lagrange duality yields:

$$\underset{0 \leqslant \lambda \leqslant \nu}{\text{maximize}} \left\{ D(\lambda) \triangleq -\frac{1}{2} \lambda^T X^T X \lambda + \sum_{i=1}^{n} \lambda_i \right\}$$

and a natural estimate of the primal variable $\theta = \sum_{i=1}^{n} \lambda_i x_i y_i = X\lambda$. Algorithm?

Algorithm: RBCD for dual SVM

Set $\lambda^0 = 0 \in \mathbb{R}^n$, $\theta^0 = 0 \in \mathbb{R}^d$.

for $t = 0, 1, \ldots, T-1$ **do**

  sample $i_t \sim \mathcal{U}[[1, n]]$

  $\bar{\lambda} = \lambda^t_{(i_t)}$

  $\lambda^{t+1}_{(i_t)} = \mathrm{Proj}_{[0,\alpha]} \left(\lambda^t_{(i_t)} + \frac{1 - y_{i_t} \langle \theta^t, x_{i_t} \rangle}{\|x_{i_t}\|_2^2} \right)$

  $\theta^{t+1} = \theta^t + y_{i_t} x_{i_t} (\lambda^{t+1}_{(i_t)} - \bar{\lambda})$

end

Table of contents

Conclusion

Concluding remarks

What did we do?

- ◇ exploit problem structures (finite sums/expectations).
- ◇ cheaper iterations vs. slower convergence per iteration.
- ◇ Different stochastic/randomized strategies.

Methods of extreme practical use, particularly when:

- ◇ even computing a gradient is too expensive,
- ◇ updates without accouting for full dataset,
- ◇ accurate solution not needed (no need to go beyond statistical accuracy).