Statistical Optimization Methods for Machine Learning

Lin Xiao
Microsoft Research AI

Online Seminar on Mathematical Foundations of Data Science
September 15, 2020

Statistics and optimization

two pillars of machine learning

- optimization provides powerful tools for statistics

$$\text{optimization} \quad \longrightarrow \quad \text{statistics}$$

- statistics can help improve optimization algorithms

$$\text{statistics} \quad \longrightarrow \quad \text{optimization}$$

Statistics and optimization

two pillars of machine learning

- optimization provides powerful tools for statistics

$$\text{optimization} \quad \longrightarrow \quad \text{statistics}$$

- statistics can help improve optimization algorithms

$$\text{statistics} \quad \longrightarrow \quad \text{optimization}$$

example: two facets of stochastic gradient descent (SGD)

- a powerful optimization algorithm for solving ML problems
- invented with statistical insight (Robbins & Monro 1951)

Statistics and optimization

two pillars of machine learning

- optimization provides powerful tools for statistics

$$\text{optimization} \quad \longrightarrow \quad \text{statistics}$$

- statistics can help improve optimization algorithms

$$\text{statistics} \quad \longrightarrow \quad \text{optimization}$$

example: two facets of stochastic gradient descent (SGD)

- a powerful optimization algorithm for solving ML problems
- invented with statistical insight (Robbins & Monro 1951)

this talk: **optimization algorithms powered by statistics**

Outline

- **hypothesis testing** for tuning learning rate
- **variance reduction** for composite optimization
- **statistical preconditioning** via sub-sampling
- summary

Outline

- **hypothesis testing** for tuning learning rate
  (joint work with Pengchuan Zhang, Hunter Lang & Qiang Liu)
- **variance reduction** for composite optimization
- **statistical preconditioning** via sub-sampling
- summary

Deep Learning in PyTorch

input | neural network (weights x) | output | label

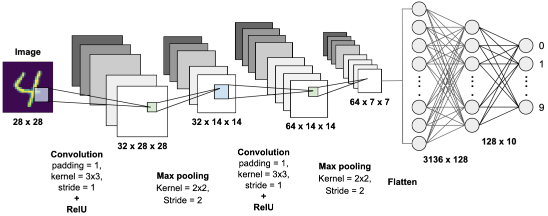


image source: Towards Data Science (Krut Patel)

output	label
y_0	0
$\vdots$	$\vdots$
y_4	1
$\vdots$	$\vdots$
y_9	0

$$\text{loss\_func}(x, (\mathit{input}, \mathit{label})) = -\log\left(\frac{\exp(y_4)}{\sum_{k=0}^{9}\exp(y_k)}\right)$$

Deep Learning in PyTorch

$\underbrace{\text{input}}$ $\underbrace{\text{neural network (weights } x\text{)}}$ $\underbrace{\text{output}}$ $\underbrace{\text{label}}$

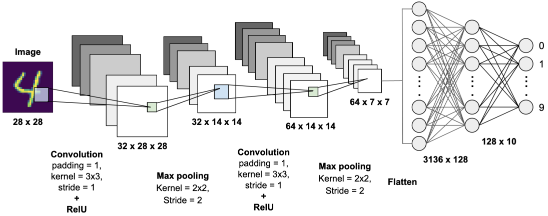


image source: Towards Data Science (Krut Patel)

output	label
y_0	0
$\vdots$	$\vdots$
y_4	1
$\vdots$	$\vdots$
y_9	0

$$\text{loss\_func}(x, (\mathit{input}, \mathit{label})) = -\log\left(\frac{\exp(y_4)}{\sum_{k=0}^{9} \exp(y_k)}\right)$$

```
import torch, torchvision
train_loader = torch.utils.data.DataLoader(......)

neuralnet = torchvision.models.resnet18().to(device)
loss_func = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(neuralnet.parameters(), lr=0.1, momentum=0.9)

for epoch in range(100):
    for (inputs, labels) in train_loader:
        loss = loss_func(neuralnet(inputs), labels)
        optimizer.zero_grad()
        loss.backward()          # compute stochastic gradient g(k)
        optimizer.step()         # update x(k+1) = x(k) - lr * d(k)
```

Stochastic gradient methods

stochastic optimization problem

$$\underset{x\in\mathcal{R}^p}{\text{minimize}} \quad F(x) \triangleq \mathsf{E}_\xi\left[f_\xi(x)\right] \qquad \left(F(x) \triangleq \frac{1}{N}\sum_{i=1}^{N} f_i(x)\right)$$

general form of algorithms:

$$x^{k+1} = x^k - \alpha_k d^k$$

Stochastic gradient methods

stochastic optimization problem

$$\underset{x \in \mathcal{R}^p}{\text{minimize}} \quad F(x) \triangleq \mathsf{E}_\xi\left[f_\xi(x)\right] \qquad \left(F(x) \triangleq \frac{1}{N}\sum_{i=1}^{N} f_i(x)\right)$$

general form of algorithms:

$$x^{k+1} = x^k - \alpha_k d^k$$

- stochastic gradient descent (SGD):

$$d^k = g^k \triangleq \nabla f_{\xi^k}(x^k)$$

- stochastic heavy-ball (SHB):

$$d^k = (1-\beta_k)g^k + \beta_k d^{k-1}$$

- Nesterov momentum (NAG):

$$d^k = \nabla f_{\xi_k}(x^k - \alpha_k \beta_k d^{k-1}) + \beta_k d^{k-1}$$

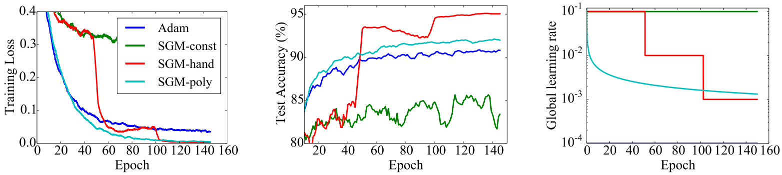
How to choose the learning rate?

- in theory (to have $\liminf_{k\to\infty} \|\nabla F(x^k)\| = 0$ a.s.)

$$\alpha_k = \frac{a}{(b+k)^c}, \qquad a, b > 0, \quad 1/2 \le c \le 1$$

- adaptive rules for adjusting learning rate
  - optimization literature (Kesten 1958, Mirzoakhmedov & Yryasev 1983, Ruszczyński & Syski 1983'86, Delyon & Juditsky 1993, ...)
  - machine learning literature (Jacobs 1988, Sutton 1992, Schraudolph 1999, Mahmood, Sutton, Degris & Pilarski 2012, Baydin, Cornish, Rubio, Schmidt & Wood 2018, ...)
- adaptive algorithms with diagonal scaling
  - AdaGrad (Duchi, Hazan & Singer 2011)
  - RMS-prop (Tieleman & Hinton 2012)
  - Adam (Kingma & Ba 2014)

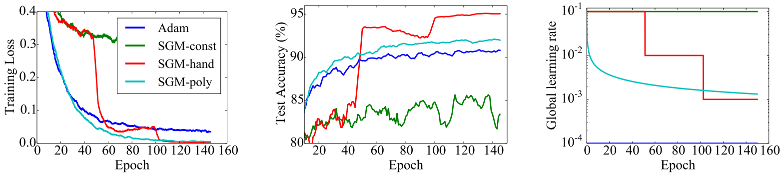
Tuning the learning rate (LR)



manual, two-phase procedure:

- trial and error to set a "good" initial LR
- gradually decrease LR
  - adaptive, roughly $1/\sqrt{k}$ decay (e.g., AdaGrad, Adam)
  - "constant-and-cut": decrease by factor of 10 every 50 epochs

Tuning the learning rate (LR)

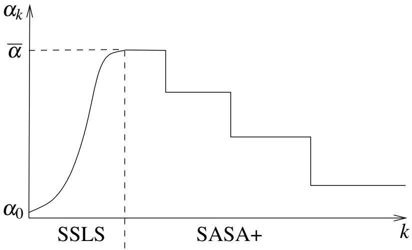


manual, two-phase procedure:

- trial and error to set a "good" initial LR
- gradually decrease LR
  - adaptive, roughly $1/\sqrt{k}$ decay (e.g., AdaGrad, Adam)
  - "constant-and-cut": decrease by factor of ? every ? epochs

"good" hyperparameters vary for different models and datasets
(mostly measured in testing/generalization performance)

Statistical adaptive stochastic gradient method



automatic, two-phase procedure:

- SSLS (Smoothed Stochastic Line Search)
  - start from a small, but nonetheless arbitrary initial LR
  - warm up learning process to reach a stable LR
- SASA+ (Stastitical Adaptive Stochastic Approximation)
  - use hypothesis testing to detect stationarity (stagnation)
  - decrease LR by constant factor whenever stationary

Why "constant-and-cut" works?

convex optimization

$$\underset{x \in \mathcal{R}^p}{\text{minimize}} \quad F(x) \triangleq \mathsf{E}_\xi \left[f_\xi(x) \right]$$

- SGD: $x^{k+1} = x^k - \alpha g^k$

  where $\quad g^k = \nabla f_{\xi_k}(x^k)$

  $\qquad\qquad = \nabla F(x^k) + \text{noise}$

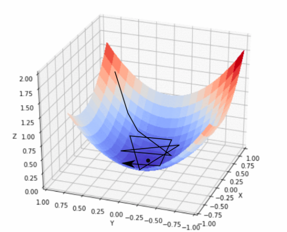


image credit: blog by Ayoosh Kathuria

Why "constant-and-cut" works?

convex optimization

$$\underset{x \in \mathcal{R}^p}{\text{minimize}} \quad F(x) \triangleq \mathsf{E}_\xi\left[f_\xi(x)\right]$$

- SGD: $x^{k+1} = x^k - \alpha g^k$

  where $\quad g^k = \nabla f_{\xi_k}(x^k)$
  $\qquad\qquad = \nabla F(x^k) + \text{noise}$

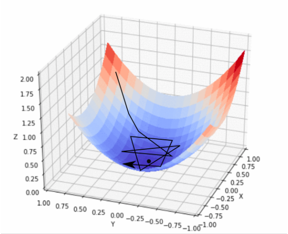
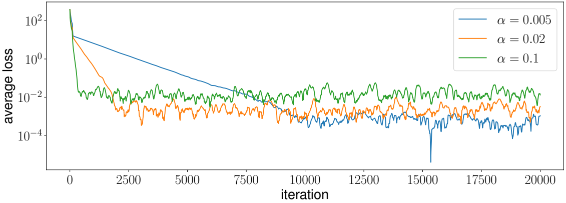


image credit: blog by Ayoosh Kathuria

- converge to stationary distribution (Bottou, Curtis & Nocedal 2018)

$$\mathsf{E}[F(x^k)] - F_\star \le (1 - c_1\alpha)^k\big(F(x^0) - F_\star - c_2\alpha\big) + c_2\alpha$$



Statistical methods for tuning LR

- Kesten 1958 (extensions by Delyon & Juditsky 1993)
  - check signs of inner products $\langle g^k, g^{k+1}\rangle$, $\langle g^{k+1}, g^{k+2}\rangle$, ...
  - change of sign indicate slow progress $\longrightarrow$ decrease LR
- Ruszczyńsky & Syski (1983)
  - check optimality conditions for LR and momentum
  - use online t-test
- Pflug (1983, 1989): online confidence interval test
  - check if dynamics is stationary under quadratic approximation
  - use online t-test

Statistical methods for tuning LR

- Kesten 1958 (extensions by Delyon & Juditsky 1993)
  - check signs of inner products $\langle g^k, g^{k+1}\rangle$, $\langle g^{k+1}, g^{k+2}\rangle$, ...
  - change of sign indicate slow progress $\longrightarrow$ decrease LR
- Ruszczyński & Syski (1983)
  - check optimality conditions for LR and momentum
  - use online t-test
- Pflug (1983, 1989): online confidence interval test
  - check if dynamics is stationary under quadratic approximation
  - use online t-test

two key ingredients of **hypothesis testing**

- what condition(s) to check for stationarity
- how to check them (statistical test)

General setting

- stochastic first-order methods with constant hyperparameters:

$$x^{k+1} = x^k - \alpha d^k$$

e.g, quasi-hyperbolic momentum (QHM) (Ma & Yarats 2019)

$$h^k = (1-\beta)g^k + \beta h^{k-1}$$
$$d^k = (1-\nu)g^k + \nu h^k$$

cover popular cases (all implemented in PyTorch, TensorFlow):
  - SGD: $\beta = 0$ and $\nu = 0$
  - SHB: $\nu = 1$
  - NAG: $0 < \beta = \nu < 1$

- **assumption:**
  - dynamics stable (for QHM, see Gitman, Lang, Zhang & X. 2019)
  - constant LR leads to convergence to stationary state

Necessary conditions for stationarity

- **definition:** $\{x^k\}$ *(strongly) stationary* if joint distribution of any subset invariant w.r.t. simultaneous shifts in time index

- implication: for any $\phi : \mathcal{R}^p \to \mathcal{R}$, any integer i

$$\mathsf{E}_\pi\left[\phi(x^{k+i})\right] = \mathsf{E}_\pi\left[\phi(x^k)\right]$$

  where π is stationary distribution, $x^k \sim \pi$ for all k

Necessary conditions for stationarity

- **definition:** $\{x^k\}$ *(strongly) stationary* if joint distribution of any subset invariant w.r.t. simultaneous shifts in time index

- implication: for any $\phi : \mathcal{R}^p \to \mathcal{R}$, any integer i

$$\mathsf{E}_\pi\left[\phi(x^{k+i})\right] = \mathsf{E}_\pi\left[\phi(x^k)\right]$$

  where π is stationary distribution, $x^k \sim \pi$ for all k

- obvious choice: $\phi = F = \mathsf{E}_\xi[f_\xi(\cdot)]$
  - not directly observable: can only observe $f_{\xi_k}(x^k)$
  - widely adopted in practice: eyeballing training loss
  - can be formalized with statistical hypothesis testing
  - not always a good indicator (very partial view of $\{x^k\}$) (especially under ill-conditioning)

A simple condition to test

- setting $\phi(x) = \frac{1}{2}\|x\|^2$ in $\mathsf{E}_\pi\left[\phi(x^{k+1})\right] = \mathsf{E}_\pi\left[\phi(x^k)\right]$ leads to

$$\mathsf{E}_\pi\left[\left\langle x^k, d^k\right\rangle - \tfrac{\alpha}{2}\|d^k\|^2\right] = 0$$

  - *exact* condition for any methods of form $x^{k+1} = x^k - \alpha d^k$
  - independent of loss function F
  - independent of noise model of stochastic gradients $\nabla f_{\xi_k}(x^k)$

A simple condition to test

- setting $\phi(x) = \frac{1}{2}\|x\|^2$ in $\mathsf{E}_\pi\left[\phi(x^{k+1})\right] = \mathsf{E}_\pi\left[\phi(x^k)\right]$ leads to

$$\mathsf{E}_\pi\left[\left\langle x^k, d^k\right\rangle - \tfrac{\alpha}{2}\|d^k\|^2\right] = 0$$

  - *exact* condition for any methods of form $x^{k+1} = x^k - \alpha d^k$
  - independent of loss function F
  - independent of noise model of stochastic gradients $\nabla f_{\xi_k}(x^k)$

- stationary condition of Yaida (2018)

$$\mathsf{E}_\pi\left[\left\langle x^k, g^k\right\rangle - \tfrac{\alpha}{2}\tfrac{1+\beta}{1-\beta}\|d^k\|^2\right] = 0$$

  - specific for SHB, with $d^k = (1-\beta)g^k + \beta d^{k-1}$
  - equivalent to our condition when running SHB

Statistical test in SASA+

- suppose $\mathsf{E}\left[\Delta_k \triangleq \left\langle x^k, d^k \right\rangle - \frac{\alpha}{2}\|d^k\|^2\right] \to 0$, by Markov chain CLT

$$\frac{1}{N}\sum_{k=1}^{N} \Delta_k \longrightarrow \mathcal{N}\left(0, \frac{\sigma_\Delta^2}{N}\right)$$

Statistical test in SASA+

- suppose $\mathrm{E}\left[\Delta_k \triangleq \left\langle x^k, d^k \right\rangle - \frac{\alpha}{2}\|d^k\|^2\right] \to 0$, by Markov chain CLT

$$\frac{1}{N}\sum_{k=1}^{N} \Delta_k \longrightarrow \mathcal{N}\left(0, \frac{\sigma_\Delta^2}{N}\right)$$

- hypothesis thesting

null: stationary vs **alternative**: non-stationary

Statistical test in SASA+

- suppose $\mathsf{E}\big[\Delta_k \triangleq \big\langle x^k, d^k \big\rangle - \frac{\alpha}{2}\|d^k\|^2\big] \to 0$, by Markov chain CLT

$$\frac{1}{N}\sum_{k=1}^{N} \Delta_k \longrightarrow \mathcal{N}\left(0, \frac{\sigma_\Delta^2}{N}\right)$$

- hypothesis thesting

null: stationary vs **alternative**: non-stationary

- $(1-\delta)$-confidence interval: $\mathcal{I}_{N,\delta} = (\hat{\mu}_N - \omega_N,\ \hat{\mu}_N + \omega_N)$

$$\hat{\mu}_N = \frac{1}{N}\sum_{i=k-N+1}^{N} \Delta_i, \qquad \omega_N = t^*_{1-\delta/2}\frac{\hat{\sigma}_N}{\sqrt{N}}$$

where $t^*_{1-\delta/2}$ is $(1-\delta/2)$ quantile of Student's t-distribution

Statistical test in SASA+

- suppose $\mathsf{E}\big[\Delta_k \triangleq \langle x^k, d^k\rangle - \frac{\alpha}{2}\|d^k\|^2\big] \to 0$, by Markov chain CLT

$$\frac{1}{N}\sum_{k=1}^{N}\Delta_k \longrightarrow \mathcal{N}\left(0, \frac{\sigma_\Delta^2}{N}\right)$$

- hypothesis thesting

null: stationary vs **alternative**: non-stationary

- $(1-\delta)$-confidence interval: $\mathcal{I}_{N,\delta} = (\hat{\mu}_N - \omega_N,\ \hat{\mu}_N + \omega_N)$

$$\hat{\mu}_N = \frac{1}{N}\sum_{i=k-N+1}^{N}\Delta_i, \qquad \omega_N = t^*_{1-\delta/2}\frac{\hat{\sigma}_N}{\sqrt{N}}$$

where $t^*_{1-\delta/2}$ is $(1-\delta/2)$ quantile of Student's t-distribution

- confidence interval test
  - $0 \notin \mathcal{I}_{N,\delta}$: reject null hypothesis $\longrightarrow$ keep LR constant
  - $0 \in \mathcal{I}_{N,\delta}$: cannot reject the null $\longrightarrow$ decrease LR

MCMC variance estimation

- mean and variance estimation for i.i.d. random variables

$$\bar{\Delta} = \frac{1}{N}\sum_{k=1}^{N} \Delta_k, \qquad \hat{\sigma}_N^2 = \frac{1}{N-1}\sum_{k=1}^{N}\left(\Delta_k - \bar{\Delta}\right)^2$$

- but $\{\Delta_k\}$ non-i.i.d., highly correlated due to $x^{k+1} = x^k - \alpha d^k$

MCMC variance estimation

- mean and variance estimation for i.i.d. random variables

$$\bar{\Delta} = \frac{1}{N}\sum_{k=1}^{N}\Delta_k, \qquad \hat{\sigma}_N^2 = \frac{1}{N-1}\sum_{k=1}^{N}\left(\Delta_k - \bar{\Delta}\right)^2$$

- but $\{\Delta_k\}$ non-i.i.d., highly correlated due to $x^{k+1} = x^k - \alpha d^k$
- MCMC variance estimators
  - batch-mean (BM) variance estimator (suppose $N = pq$)

$$\underbrace{\Delta_1, \ldots, \Delta_q}_{\bar{\Delta}_1 = \frac{1}{q}\sum_{k=1}^{q}\Delta_k}, \underbrace{\Delta_{q+1}, \ldots, \Delta_{2q}}_{\bar{\Delta}_2}, \ldots, \ldots, \ldots, \underbrace{\Delta_{(p-1)q+1}, \ldots, \Delta_{pq}}_{\bar{\Delta}_p}$$

MCMC variance estimation

- mean and variance estimation for i.i.d. random variables

$$\bar{\Delta} = \tfrac{1}{N} \textstyle\sum_{k=1}^{N} \Delta_k, \qquad \hat{\sigma}_N^2 = \tfrac{1}{N-1} \textstyle\sum_{k=1}^{N} \left(\Delta_k - \bar{\Delta}\right)^2$$

- but $\{\Delta_k\}$ non-i.i.d., highly correlated due to $x^{k+1} = x^k - \alpha d^k$
- MCMC variance estimators
  - batch-mean (BM) variance estimator (suppose $N = pq$)

$$\underbrace{\Delta_1, \ldots, \Delta_q}_{\bar{\Delta}_1 = \frac{1}{q} \sum_{k=1}^{q} \Delta_k}, \underbrace{\Delta_{q+1}, \ldots, \Delta_{2q}}_{\bar{\Delta}_2}, \ldots, \ldots, \ldots, \underbrace{\Delta_{(p-1)q+1}, \ldots, \Delta_{pq}}_{\bar{\Delta}_p}$$

$$\hat{\sigma}_N^2 = \frac{q}{p-1} \sum_{i=1}^{p} \left(\bar{\Delta}_i - \bar{\Delta}\right)^2, \qquad (\text{d.o.f.} = p-1)$$

(strong consistency established by Jones, Haran, Caffo & Neath 2006)

  - overlapping batch mean (OLBM) variance estimator

(strong consistency established by Flegal & Jones 2009)

Algorithm 1: SASA+

---

input: x^0, α_0 (default parameters: $\theta = 1/4$, $\tau = 1/10$, $\delta = 0.05$)

$\alpha \leftarrow \alpha_0$

$k_o \leftarrow 0$

for $k = 0, ..., T-1$ **do**

  $x^{k+1} \leftarrow x^k - \alpha d^k$ (updating weights)

  $\Delta_k \leftarrow \langle x^k, d^k \rangle - \frac{\alpha}{2}\|d^k\|^2$ (collecting statistics)

  $N \leftarrow \lceil \theta(k - k_o) \rceil$ (numbers to keep)

  if $N > N_{\min}$ and $k \bmod K_{\text{test}} == 0$ **then**

    $(\hat{\mu}_N, \hat{\sigma}_N) \leftarrow$ statistics of $\{\Delta_{k-N+1}, \ldots, \Delta_k\}$

    if $0 \in \hat{\mu}_N \pm t^*_{1-\delta/2} \frac{\hat{\sigma}_N}{\sqrt{N}}$ **then** (confidence interval test)

      $\alpha \leftarrow \tau\alpha$ (decrease LR)

      $k_o \leftarrow k$ (reset counter of statistics)

    end

  end

end

Default hyperparameters for SASA+

Parameter	Explanation	Default value
$N_{\text{min}} \in \mathbb{Z}_+$	min. # of statistics for testing	$\min\{1000, \lceil n/b \rceil\}$
$K_{\text{test}} \in \mathbb{Z}_+$	period to perform statistical test	$\min\{100, \lceil n/b \rceil\}$
$\delta \in (0,1)$	$(1-\delta)$-confidence interval	0.05
$\theta \in (0,1)$	fraction of recent samples to keep	1/4
$\tau \in (0,1)$	learning rate drop factor	1/10

where n is number of training examples and b is mini-batch size

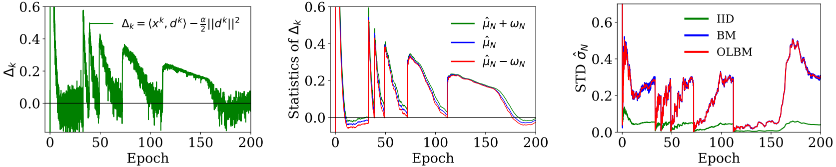
- works well across different network models and datasets
- can be adjusted, but have very low sensitivity

fixed for all our experiments on different models and datasets

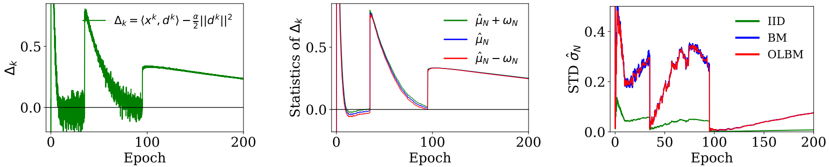
SASA+ statistical tests

ResNet18 on CIFAR-10 (defaults: $\delta = 0.05$, $\tau = 1/10$, $\theta = 1/4$)

- with LR drop factor $\tau = 1/2$



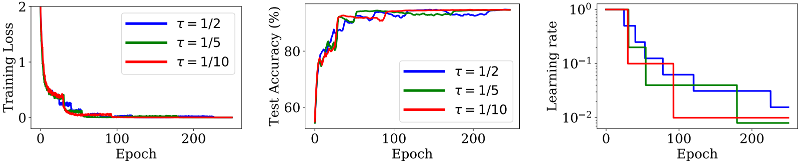
- with LR drop factor $\tau = 1/10$



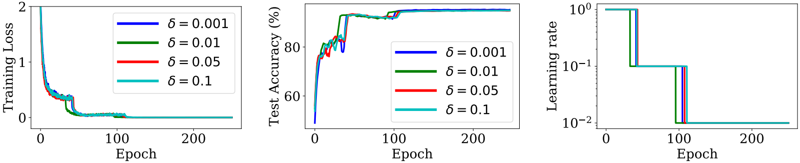
SASA+ sensitivity study

ResNet18 on CIFAR-10 (defaults: $\delta = 0.05$, $\tau = 1/10$, $\theta = 1/4$)

- varying drop factor τ



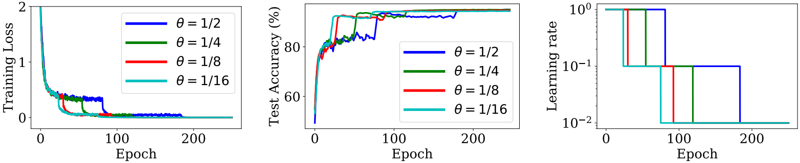
- varying confidence level $1 - \delta$



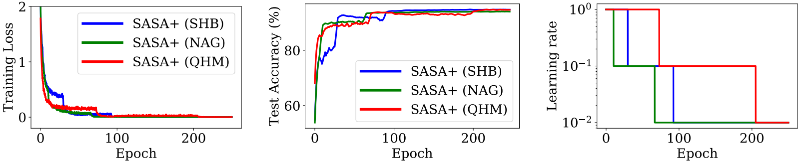
SASA+ sensitivity study

ResNet18 on CIFAR-10 (defaults: $\delta = 0.05$, $\tau = 1/10$, $\theta = 1/4$)

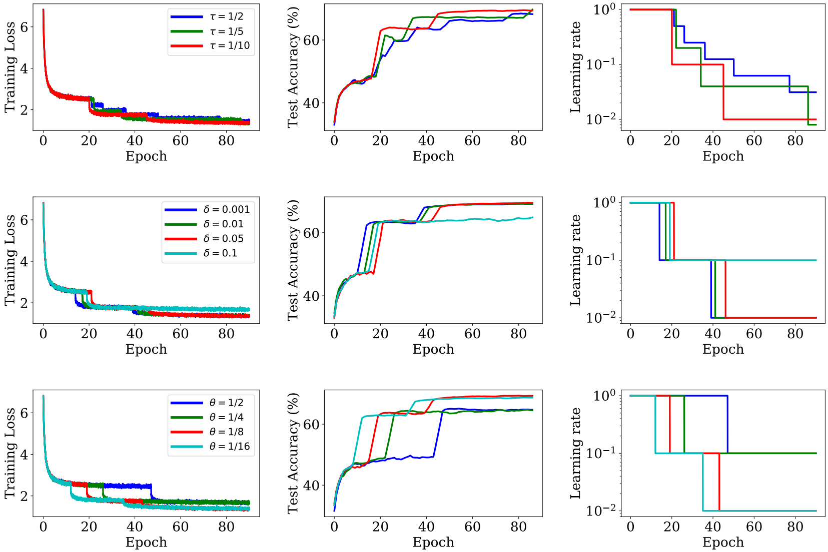
- varying fraction of recent samples θ



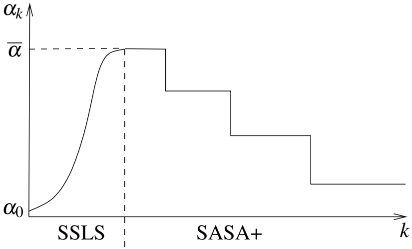
- applied to different SGD variants



SASA+ sensitivity on ImageNet



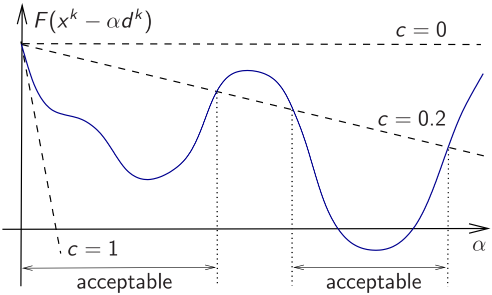
How to set initial LR?



automatic, two-phase procedure:

- **SSLS (Smoothed Stochastic Line Search)**
  - start from a small, but nonetheless arbitrary initial LR
  - warm up learning process to reach a stable LR
- SASA+ (Stastitical Adaptive Stochastic Approximation)
  - use statistical test to detect stationarity (stagnation)
  - decrease LR by constant factor whenever stationary

Armijo line search



classical technique in *deterministic* optimization

- in each iteration k, start with an optimistic (large) α
- reduce α if necessary to satisfy inequality

$$F(x^k - \alpha d^k) \leq F(x^k) - c\left\langle(\nabla F(x^k), \alpha d^k\right\rangle$$

$c \in (0, 1/2)$: sufficient decrease coefficient

Algorithm 2: Smoothed Stochastic Line-Search (SSLS)

input: x^0, α_{-1}, and parameters $c \in (0, 1/2)$, $m > 0$

for $k = 0, ..., T-1$ **do**

    sample ξ_k, compute $g^k \leftarrow \nabla f_{\xi_k}(x^k)$ and d^k

    $\eta_k \leftarrow 2\alpha_{k-1}$ (always try large LR first)

    for $i = 1, \ldots, m$ **do**

        if $f_{\xi_k}(x^k - \eta_k g^k) < f_{\xi_k}(x^k) - c \cdot \eta_k \|g^k\|^2$ **then**

            break

        else

            $\eta_k \leftarrow \eta_k / 2$ (decrease LR if necessary)

        end

    end

    $\boxed{\alpha_k \leftarrow (1-\gamma)\alpha_{k-1} + \gamma\eta_k}$ (smoothing the LR)

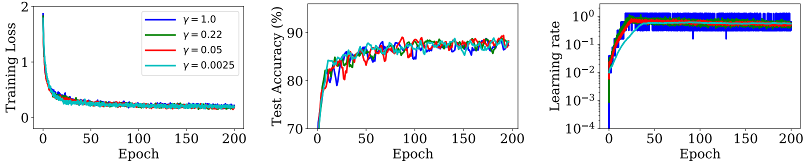
    $x^{k+1} \leftarrow x^k - \alpha_k d^k$

end

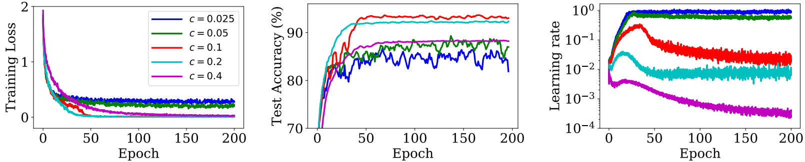
SSLS experiments

ResNet18 on CIFAR-10 (defaults: $c = 0.1$, $\gamma = \sqrt{b/n}$)

- varying smoothing parameter γ



- varying sufficient descent coefficient c



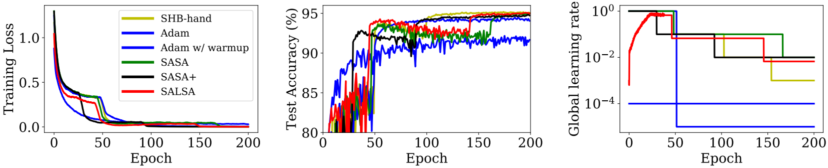
SALSA

Algorithm 3: SALSA: SASA+ with warmup by SSLS

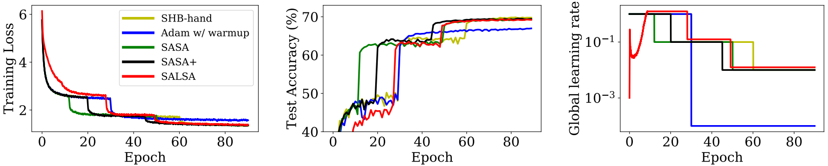
input: $x^0 \in \mathcal{R}^p$, $\alpha_0 > 0$, *switched*=False
for $k = 0, ..., T$ **do**
  if *not switched* **then**
    run one step of SSLS (Algorithm 2)
    x_stationary $\leftarrow$ SASA+ test
    f_stationary $\leftarrow$ SLOPE test
    switched $\leftarrow$ *x_stationary* or *f_stationary*
  else
    run one step of SASA+ (Algorithm 1)
  end
end
output: x^T

SALSA experiments

- ResNet18 on CIFAR-10

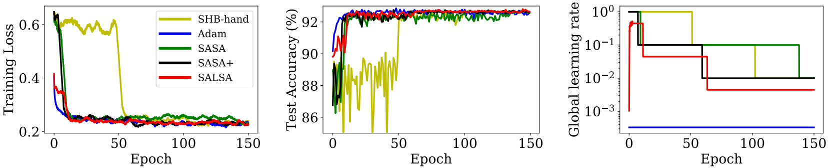


- ResNet18 on ImageNet

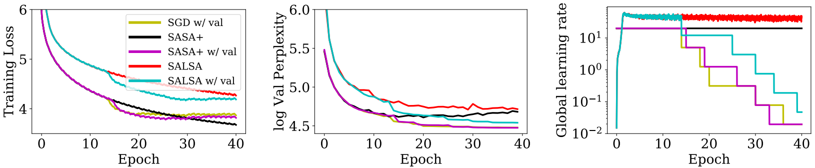


SALSA experiments

- logistic regression on MNIST



- RNN on Wikitext-2



Summary of SALSA

SALSA: automated, two-phase procedure

- SSLS: warm up with smoothed line search to reach a stable LR
- SASA+: decrease LR (stage-wise) based on statistical tests

Summary of SALSA

SALSA: automated, two-phase procedure

- SSLS: warm up with smoothed line search to reach a stable LR
- SASA+: decrease LR (stage-wise) based on statistical tests

statistical hypothesis testing

- powerful tools for stochastic optimization
- help make training ML models autonomous and reliable

> *The speed of convergence questions are closely related to on-line rules for determining step coefficients in SA algorithms. In the authors' opinion, the use of statistical tests, . . . , is a promising direction of further research.*
>
> — *Ruszczyńsky & Syski (1983)*

Outline

- hypothesis testing for tuning learning rate

- **variance reduction for composite optimization**
  (joint work with Junyu Zhang)

- statistical preconditioning via sub-sampling

- summary

Finite-sum optimization

$$\underset{x}{\text{minimize}} \quad F(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(x) \quad \left(\text{special case of } \mathrm{E}_\xi[f_\xi(x)]\right)$$

- SGD: for each $k > 0$, randomly pick $i_k \in \{1, \ldots, n\}$

$$x^{k+1} = x^k - \alpha_k \nabla f_{i_k}(x^k)$$

  - $\mathrm{E}[\nabla f_{i_k}(x^k)] = \nabla F(x^k)$, but $\mathrm{Var}(\nabla f_{i_k}(x^k)) \nrightarrow 0$
  - need $\alpha_k \to 0$ for convergence (e.g., $\alpha_k \sim 1/\sqrt{k}$)

Finite-sum optimization

$$\underset{x}{\text{minimize}} \quad F(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(x) \quad \Big(\text{special case of } \mathrm{E}_\xi[f_\xi(x)]\Big)$$

- SGD: for each $k > 0$, randomly pick $i_k \in \{1, \ldots, n\}$

$$x^{k+1} = x^k - \alpha_k \nabla f_{i_k}(x^k)$$

  - $\mathrm{E}[\nabla f_{i_k}(x^k)] = \nabla F(x^k)$, but $\mathrm{Var}(\nabla f_{i_k}(x^k)) \nrightarrow 0$
  - need $\alpha_k \to 0$ for convergence (e.g., $\alpha_k \sim 1/\sqrt{k}$)

- practical implications
  - hard to tune $\alpha_k \to 0$ in practice (motivation for SALSA)
  - hard to work with ℓ_1-regularization (RDA method 2009)
  - slow convergence: $O(1/\sqrt{k})$ rate or $O(\epsilon^{-2})$ complexity

Finite-sum optimization

$$\underset{x}{\text{minimize}} \quad F(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(x) \quad \Big(\text{special case of } \mathrm{E}_\xi[f_\xi(x)]\Big)$$

- SGD: for each $k > 0$, randomly pick $i_k \in \{1, \ldots, n\}$

$$x^{k+1} = x^k - \alpha_k \nabla f_{i_k}(x^k)$$

  - $\mathrm{E}[\nabla f_{i_k}(x^k)] = \nabla F(x^k)$, but $\mathrm{Var}(\nabla f_{i_k}(x^k)) \nrightarrow 0$
  - need $\alpha_k \to 0$ for convergence (e.g., $\alpha_k \sim 1/\sqrt{k}$)

- practical implications
  - hard to tune $\alpha_k \to 0$ in practice (motivation for SALSA)
  - hard to work with ℓ_1-regularization (RDA method 2009)
  - slow convergence: $O(1/\sqrt{k})$ rate or $O(\epsilon^{-2})$ complexity

variance reduction: capable of improving all three aspects

Variance reduction with control variate

- **goal:** estimate $\mathbf{E}[X]$ of random variable X with low variance
- **control variate:** a random variable Y such that
  - $\mathbf{E}[Y]$ easy to compute
  - $X - Y$ can be sampled/simulated with same cost as X
  - $\mathbf{Var}(X-Y) < \mathbf{Var}(X)$
- define
$$X' = X - Y + \mathbf{E}[Y]$$
which satisfies $\mathbf{E}[X'] = \mathbf{E}[X]$ and $\mathbf{Var}(X') < \mathbf{Var}(X)$

Variance reduction with control variate

- **goal:** estimate $\mathrm{E}[X]$ of random variable X with low variance
- **control variate:** a random variable Y such that
  - $\mathrm{E}[Y]$ easy to compute
  - $X - Y$ can be sampled/simulated with same cost as X
  - $\mathrm{Var}(X - Y) < \mathrm{Var}(X)$
- define

$$X' = X - Y + \mathrm{E}[Y]$$

  which satisfies $\mathrm{E}[X'] = \mathrm{E}[X]$ and $\mathrm{Var}(X') < \mathrm{Var}(X)$

- for SGD, define

$$v^k = \nabla f_{i_k}(x^k) - \nabla f_{i_k}(x^0) + \nabla F(x^0)$$

  - $\mathrm{E}[v^k] = \nabla F(x^k)$, $\mathrm{Var}(v^k)$ small if $\|x^k - x^0\|$ small
  - periodically update x^0 to ensure $\mathrm{Var}(v^k) \to 0$
- can use constant learning rate (which has several benefits!)

Stochastic variance reduction

SVRG (Johnson & Zhang 2013)

> given $x^0 \in \mathcal{R}^n$, VR period m, step size $\alpha \sim \frac{1}{L}$
> **for** $s = 1, 2, \ldots$
> $v^0 = \nabla F(x^0) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(x^0)$
>   **for** $k = 0, \ldots, m-1$
>   sample $i_k \in \{1, \ldots, n\}$
> $v^k = \nabla f_{i_k}(x^k) - \nabla f_{i_k}(x^0) + v^0$
> $x^{k+1} = x^k - \alpha v^k$
> $x^0 \leftarrow x^m$

smoothness assumption (to ensure control variate condition)

$$\|\nabla f_i(x^k) - \nabla f_i(x^0)\| \leq L\|x^k - x^0\|, \quad i = 1, \ldots, n$$

Stochastic variance reduction

SVRG (Johnson & Zhang 2013)

> given $x^0 \in \mathcal{R}^n$, VR period m, step size $\alpha \sim \frac{1}{L}$
> **for** $s = 1, 2, \ldots$
> $\quad v^0 = \nabla F(x^0) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(x^0)$
> $\quad$ **for** $k = 0, \ldots, m-1$
> $\quad\quad$ sample $i_k \in \{1, \ldots, n\}$
> $\quad\quad v^k = \nabla f_{i_k}(x^k) - \nabla f_{i_k}(x^0) + v^0$
> $\quad\quad x^{k+1} = x^k - \alpha v^k$
> $\quad x^0 \leftarrow x^m$

smoothness assumption (to ensure control variate condition)

$$\|\nabla f_i(x^k) - \nabla f_i(x^0)\| \le L\|x^k - x^0\|, \quad i = 1, \ldots, n$$

complexities $\mathcal{O}(\cdot)$

	GD	SGD	SVRG/SAGA
convex: $\mathrm{E}[F(x)] - F_\star \le \epsilon$	$n\,\epsilon^{-1}$	ϵ^{-2}	$n + \epsilon^{-1}$

Stochastic variance reduction

SVRG (Johnson & Zhang 2013)

given $x^0 \in \mathcal{R}^n$, VR period m, step size $\alpha \sim \frac{1}{L}$
for $s = 1, 2, \ldots$
  $v^0 = \nabla F(x^0) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(x^0)$
  for $k = 0, \ldots, m-1$
    sample $i_k \in \{1, \ldots, n\}$
    $v^k = \nabla f_{i_k}(x^k) - \nabla f_{i_k}(x^0) + v^0$
    $x^{k+1} = x^k - \alpha v^k$
  $x^0 \leftarrow x^m$

smoothness assumption (to ensure control variate condition)

$$\|\nabla f_i(x^k) - \nabla f_i(x^0)\| \leq L\|x^k - x^0\|, \quad i = 1, \ldots, n$$

complexities $\mathcal{O}(\cdot)$

	GD	SGD	SVRG/SAGA
convex: $\mathbf{E}[F(x)] - F_\star \leq \epsilon$	$n\,\epsilon^{-1}$	ϵ^{-2}	$n + \epsilon^{-1}$
nonconvex: $\mathbf{E}[\|\nabla F(x)\|^2] \leq \epsilon$	$n\,\epsilon^{-1}$	ϵ^{-2}	$n + n^{2/3}\epsilon^{-1}$

Stochastic variance reduction

SARAH (Nguyen et al., 2017) and SPIDER (Fang et al., 2018)

given $x^0 \in \mathcal{R}^n$, VR period m, step size $\alpha \sim \frac{1}{L}$
for $s = 1, 2, \dots$
  $v^0 = \nabla F(x^0) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(x^0)$
  for $k = 0, \dots, m-1$
    sample $i_k \in \{1, \dots, n\}$
    $v^k = \nabla f_{i_k}(x^k) - \nabla f_{i_k}(x^{k-1}) + v^{k-1}$
    $x^{k+1} = x^k - \alpha v^k$
  $x^0 \leftarrow x^m$

smoothness assumption (to ensure control variate condition)

$$\|\nabla f_i(x^k) - \nabla f_i(x^0)\| \le L\|x^k - x^0\|, \quad i = 1, \dots, n$$

complexities $\mathcal{O}(\cdot)$

	GD	SGD	SVRG/SAGA	SARAH/SPIDER
convex: $\mathbb{E}[F(x)] - F_\star \le \epsilon$	$n\,\epsilon^{-1}$	ϵ^{-2}	$n + \epsilon^{-1}$	$n + \epsilon^{-1}$
nonconvex: $\mathbb{E}[\|\nabla F(x)\|^2] \le \epsilon$	$n\,\epsilon^{-1}$	ϵ^{-2}	$n + n^{2/3}\epsilon^{-1}$	$n + n^{1/2}\epsilon^{-1}$

Composite stochastic optimization

- composition with expectation (or finite-sum)

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \quad f\left(\mathsf{E}_\xi[g_\xi(x)]\right) \quad \text{or} \quad f\left(\frac{1}{n}\sum_{i=1}^{n} g_i(x)\right)$$

  - $f : \mathcal{R}^p \to \mathcal{R}$ **smooth and can be nonconvex**
  - $g_\xi : \mathcal{R}^d \to \mathcal{R}^p$ smooth vector mapping for every ξ

Composite stochastic optimization

- composition with expectation (or finite-sum)

$$\underset{x\in\mathcal{R}^d}{\text{minimize}} \quad f\big(\mathbf{E}_\xi[g_\xi(x)]\big) \quad \text{or} \quad f\left(\frac{1}{n}\sum_{i=1}^n g_i(x)\right)$$

  - $f:\mathcal{R}^p\to\mathcal{R}$ **smooth and can be nonconvex**
  - $g_\xi:\mathcal{R}^d\to\mathcal{R}^p$ smooth vector mapping for every ξ

- applications
  - policy evaluation with linear function approximation

$$\underset{x\in\mathcal{R}^d}{\text{minimize}} \big\|\mathbf{E}[A]x-\mathbf{E}[b]\big\|^2$$

  - risk-averse optimization

$$\underset{x\in\mathcal{R}^d}{\text{maximize}} \quad \underbrace{\frac{1}{n}\sum_{j=1}^n h_j(x)}_{\text{average reward}} -\lambda \underbrace{\frac{1}{n}\sum_{j=1}^n\left(h_j(x)-\frac{1}{n}\sum_{i=1}^n h_i(x)\right)^2}_{\text{variance of rewards (risk)}}$$

Multi-level composition

- multi-level composite stochastic optimization

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \; \mathbf{E}_{\xi_m}\left[f_{m,\xi_m}\left(\cdots \mathbf{E}_{\xi_2}\left[f_{2,\xi_2}\left(\mathbf{E}_{\xi_1}[f_{1,\xi_1}(x)]\right)\right]\cdots\right)\right] + r(x)$$

- multi-level finite-sum optimization

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \; \frac{1}{N_m}\sum_{j=1}^{N_m} f_{m,j}\left(\cdots \frac{1}{N_2}\sum_{j=1}^{N_2} f_{2,j}\left(\frac{1}{N_1}\sum_{j=1}^{N_1} f_{1,j}(x)\right)\cdots\right) + r(x)$$

- applications
  - optimization of multi-level composite risk measures
  - adversarial learning of deep neural networks

Multi-level composition

- multi-level composite stochastic optimization

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \;\; \mathbf{E}_{\xi_m}\left[f_{m,\xi_m}\left(\cdots \mathbf{E}_{\xi_2}\left[f_{2,\xi_2}\left(\mathbf{E}_{\xi_1}[f_{1,\xi_1}(x)]\right)\right]\cdots\right)\right] + r(x)$$

- multi-level finite-sum optimization

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \;\; \frac{1}{N_m}\sum_{j=1}^{N_m} f_{m,j}\left(\cdots \frac{1}{N_2}\sum_{j=1}^{N_2} f_{2,j}\left(\frac{1}{N_1}\sum_{j=1}^{N_1} f_{1,j}(x)\right)\cdots\right) + r(x)$$

- applications
  - optimization of multi-level composite risk measures
  - adversarial learning of deep neural networks

- **main results on sample complexity:** (Zhang & X. 2019)
  - dependence on ϵ and $n = \sum_{i=1}^m N_i$ similar to the case $m = 1$
  - dependence on m is polynomial (previous work exponential)

Nonsmooth composite optimization

- nonsmooth stochastic composite optimization

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \quad f\left(\mathsf{E}_\xi[g_\xi(x)]\right) + r(x) \quad \text{or} \quad f\left(\frac{1}{n}\sum_{i=1}^{n} g_i(x)\right) + r(x)$$

  - $f : \mathcal{R}^p \to \mathcal{R}$ **convex but non-smooth**
  - $g_\xi : \mathcal{R}^d \to \mathcal{R}^p$ smooth vector mapping for every ξ
  - **overall nonconvex and nonsmooth**

Nonsmooth composite optimization

- nonsmooth stochastic composite optimization

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \quad f\left(\mathsf{E}_\xi[g_\xi(x)]\right) + r(x) \quad \text{or} \quad f\left(\frac{1}{n}\sum_{i=1}^n g_i(x)\right) + r(x)$$

  - $f : \mathcal{R}^p \to \mathcal{R}$ **convex but non-smooth**
  - $g_\xi : \mathcal{R}^d \to \mathcal{R}^p$ smooth vector mapping for every ξ
  - **overall nonconvex and nonsmooth**

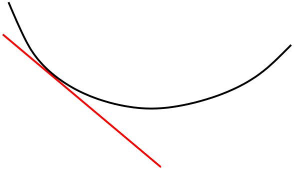
- **example:** distributionally robust optimization

$$\underset{x \in \mathcal{X}}{\text{minimize}} \quad \max_{1 \le i \le m} g^{(i)}(x) \quad \text{where} \quad g^{(i)}(x) = \mathsf{E}_{\xi_i}\left[g^{(i)}_{\xi_i}(x)\right]$$

  - each random variable ξ_i has slightly different distributions (obtained through subsampling or bootstrap)
  - regularization $r(x)$ can be used to incorporate priors

Nonsmooth composite optimization

example: SGD with better model



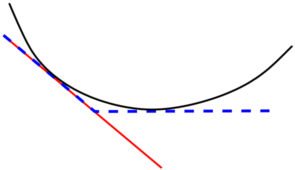
$$\underset{x}{\text{minimize}} \;\; F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

- SGD: $x^{k+1} = x^k - \alpha_k \nabla f_{i_k}$, same as

$$x^{k+1} = \arg\min_x \left\{ f_{i_k}(x^k) + \nabla f_{i_k}(x^k)(x - x^k) + \frac{1}{2\alpha_k}\|x - x^k\|^2 \right\}$$

Nonsmooth composite optimization

example: SGD with better model



$$\underset{x}{\text{minimize}} \;\; F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

- SGD: $x^{k+1} = x^k - \alpha_k \nabla f_{i_k}$, same as

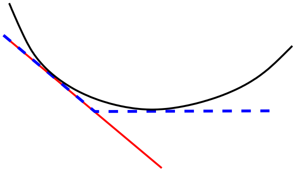
$$x^{k+1} = \arg\min_x \left\{ f_{i_k}(x^k) + \nabla f_{i_k}(x^k)(x - x^k) + \frac{1}{2\alpha_k}\|x - x^k\|^2 \right\}$$

- truncated SGD

$$x^{k+1} = \arg\min_x \left\{ \max\{ f_{i_k}(x^k) + \nabla f_{i_k}(x^k)(x - x^k),\;\; f^\star \} + \frac{1}{2\alpha_k}\|x - x^k\|^2 \right\}$$

Nonsmooth composite optimization

example: SGD with better model



$$\underset{x}{\text{minimize}} \quad F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

- SGD: $x^{k+1} = x^k - \alpha_k \nabla f_{i_k}$, same as

$$x^{k+1} = \arg\min_x \left\{ f_{i_k}(x^k) + \nabla f_{i_k}(x^k)(x - x^k) + \frac{1}{2\alpha_k}\|x - x^k\|^2 \right\}$$

- truncated SGD

$$\begin{aligned} x^{k+1} &= \arg\min_x \left\{ \max\left\{ f_{i_k}(x^k) + \nabla f_{i_k}(x^k)(x - x^k),\ f^\star \right\} + \frac{1}{2\alpha_k}\|x - x^k\|^2 \right\} \\ &= x_k - \min\left\{ \alpha_k, \frac{f_{i_k}(x^k) - f^\star}{\|\nabla f_{i_k}(x^k)\|^2} \right\} \nabla f_{i_k}(x^k) \end{aligned}$$

  - robust to stepsize choice (Asi & Duchi 2019, Davis & Drusvyatskiy 2019)

Nonsmooth composite optimization

main results (Zhang & X. 2020)

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \quad f\left(\mathsf{E}_\xi[g_\xi(x)]\right) \quad \text{or} \quad f\left(\frac{1}{n}\sum_{i=1}^{n} g_i(x)\right)$$

- assumptions:
  - f convex but nonsmooth, g_ξ mean-square smooth
  - overall nonsmooth and nonconvex, but highly structured
- **variance-reduced prox-linear methods**

$$x^{k+1} = \arg\min_x \left\{ f\left(\tilde{g}^k + \tilde{J}^k (x - x^k)\right) + \frac{M}{2}\|x - x^k\|^2 \right\}$$

$\tilde{g}^k$, $\tilde{J}^k$ computed by SVRG or SARAH/SPIDER estimators

Nonsmooth composite optimization

main results (Zhang & X. 2020)

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \quad f\left(\mathbf{E}_\xi[g_\xi(x)]\right) \quad \text{or} \quad f\left(\frac{1}{n}\sum_{i=1}^{n} g_i(x)\right)$$

- assumptions:
  - f convex but nonsmooth, g_ξ mean-square smooth
  - overall nonsmooth and nonconvex, but highly structured
- **variance-reduced prox-linear methods**

$$x^{k+1} = \arg\min_x \left\{ f\left(\tilde{g}^k + \tilde{J}^k(x - x^k)\right) + \frac{M}{2}\|x - x^k\|^2 \right\}$$

  $\tilde{g}^k$, $\tilde{J}^k$ computed by SVRG or SARAH/SPIDER estimators

- **sample complexities**
  - finite-sum: $\mathcal{O}(n + n^{4/5}\epsilon^{-1})$ for both $g_i(\cdot)$ and $g_i'(\cdot)$
  - expectation: $\mathcal{O}(\epsilon^{-5/2})$ for $g_\xi(\cdot)$ and $\mathcal{O}(\epsilon^{-3/2})$ for $g_\xi'(\cdot)$

VR for cubic regularization

finite-sum optimization:

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \;\; F(x) \triangleq \frac{1}{N} \sum_{i=1}^{N} f_i(x)$$

- ϵ-solution: $\|\nabla F(x)\| \le \epsilon$ and $\lambda_{\min}(\nabla^2 F(x)) \ge -\sqrt{\epsilon}$

VR for cubic regularization

finite-sum optimization:

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \;\; F(x) \triangleq \frac{1}{N} \sum_{i=1}^{N} f_i(x)$$

- ϵ-solution: $\|\nabla F(x)\| \leq \epsilon$ and $\lambda_{\min}(\nabla^2 F(x)) \geq -\sqrt{\epsilon}$
- Newton's method with cubic regularization

$$\Delta^k = \arg\min_{\delta} \left\{ \Delta^T g^k + \frac{1}{2} \Delta^T H^k \Delta + \frac{\sigma}{6} \|\Delta\|^3 \right\}$$

$$x^{k+1} = x^k + \Delta^k$$

  - full gradient and Hessian: $g^k = \nabla F(x^k)$ and $H^k = \nabla^2 F(x^k)$
  - sample complexity $O(N\epsilon^{-3/2})$ (Nesterov & Polyak 2006)

VR for cubic regularization

finite-sum optimization:

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \;\; F(x) \triangleq \frac{1}{N} \sum_{i=1}^{N} f_i(x)$$

- ϵ-solution: $\|\nabla F(x)\| \leq \epsilon$ and $\lambda_{\min}(\nabla^2 F(x)) \geq -\sqrt{\epsilon}$
- Newton's method with cubic regularization

$$\Delta^k = \arg\min_{\delta} \Big\{ \Delta^T g^k + \frac{1}{2} \Delta^T H^k \Delta + \frac{\sigma}{6} \|\Delta\|^3 \Big\}$$
$$x^{k+1} = x^k + \Delta^k$$

  - full gradient and Hessian: $g^k = \nabla F(x^k)$ and $H^k = \nabla^2 F(x^k)$
  - sample complexity $O(N\epsilon^{-3/2})$ (Nesterov & Polyak 2006)

- use SVRG estimators to compute g^k and H^k:
  - sample complexity $O(N + N^{2/3}\epsilon^{-3/2})$ (Zhang & X. 2018) (Wang, Zhou, Liang & Lan 2018) (Zhou, Xu & Gu 2018, 2019)

Outline

- hypothesis testing for tuning learning rate
- variance reduction for composite optimization
- **statistical preconditioning via sub-sampling**

  (joint work with Hadrien Hendrikx, Sébastien Bubeck, Francis Bach, Laurent Massoulié)
- summary

Motivation

- empirical risk minimization (ERM)

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \quad \frac{1}{N} \sum_{i=1}^{N} \ell(x, z_i) + \psi(x)$$

  - $\{z_1, \ldots, z_N\}$: i.i.d. examples from unknown distribution
  - $\ell(\cdot, z_i)$: smooth, convex loss (LS, LR, ...)
  - $\psi(\cdot)$: simple, convex regularization ($\frac{\lambda}{2}\|x\|^2$, $\lambda\|x\|_1$, ...)

Motivation

- empirical risk minimization (ERM)

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \quad \frac{1}{N} \sum_{i=1}^{N} \ell(x, z_i) + \psi(x)$$

  - $\{z_1, \ldots, z_N\}$: i.i.d. examples from unknown distribution
  - $\ell(\cdot, z_i)$: smooth, convex loss (LS, LR, ...)
  - $\psi(\cdot)$: simple, convex regularization ($\frac{\lambda}{2}\|x\|^2$, $\lambda\|x\|_1$, ...)

- distributed optimization
  - dataset too large to fit in single machine

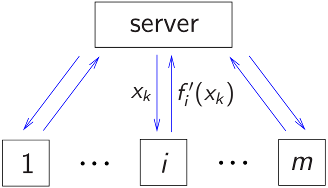
$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \quad \frac{1}{m} \sum_{i=1}^{m} f_i(x) + \psi(x)$$

    where $f_i(x) = \frac{1}{n} \sum_{j=1}^{n} \ell(x, z_{i,j})$ local to machine i

  - need communication-efficient distributed algorithms

Distributed gradient descent

$$\underset{x\in\mathcal{R}^d}{\text{minimize}} \quad F(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x)$$



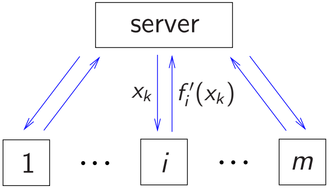
$$x_{k+1} = x_k - \alpha_k \left(\tfrac{1}{m}\textstyle\sum_{i=1}^{m} f_i'(x_k)\right)$$

broadcast / reduce

compute $f_i'(x_k)$, $i = 1, \ldots, m$

Distributed gradient descent

$$\underset{x \in \mathcal{R}^d}{\text{minimize}} \quad F(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x)$$



$x_{k+1} = x_k - \alpha_k \left(\frac{1}{m} \sum_{i=1}^{m} f_i'(x_k) \right)$

broadcast / reduce

compute $f_i'(x_k)$, $i = 1, \ldots, m$

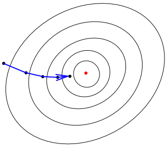
- assumption: F strongly convex (with $(\lambda/2)\|x\|^2$ *regularization*)
- number of communication rounds (iteration complexity)
  - classical gradient descent: $O(\kappa \log(1/\epsilon))$
  - accelerated gradient descent: $O(\sqrt{\kappa} \log(1/\epsilon))$

cannot be improved in general

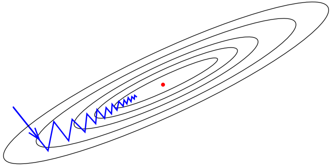
(Arjevani-Shamir 2015, Scaman-Bach-Bubeck-Lee-Massoulié 2017)

Conditon number and iteration complexity

- assumption: $\mu I \preceq \nabla^2 F(x) \preceq LI$ for all x
- **condition number:** $\kappa = \dfrac{L}{\mu}$



κ small (≈ 1)



κ large ($\gg 1$)

- iteration complexity in order to reach $F(x^{(t)}) - F^\star \le \epsilon$
  - classical gradient descent: $O(\kappa \log(1/\epsilon))$
  - accelerated gradient descent: $O(\sqrt{\kappa} \log(1/\epsilon))$

preconditioning: computationally efficient schemes to reduce κ

Relative condition number

- *reference function* ϕ: differentiable and strongly convex

$$\sigma_\phi I \preceq \nabla^2 \phi(x) \preceq L_\phi I, \qquad \kappa_\phi = \frac{L_\phi}{\sigma_\phi}$$

Relative condition number

- *reference function* ϕ: differentiable and strongly convex

$$\sigma_\phi I \preceq \nabla^2 \phi(x) \preceq L_\phi I, \qquad \kappa_\phi = \frac{L_\phi}{\sigma_\phi}$$

- *relative smoothness* and *relative strong convexity*

$$\sigma_{F/\phi} \nabla^2 \phi(x) \preceq \nabla^2 F(x) \preceq L_{F/\phi} \nabla^2 \phi(x)$$

relative condition number

$$\kappa_{F/\phi} = \frac{L_{F/\phi}}{\sigma_{F/\phi}}$$

Relative condition number

- *reference function* ϕ: differentiable and strongly convex

$$\sigma_\phi I \preceq \nabla^2 \phi(x) \preceq L_\phi I, \qquad \kappa_\phi = \frac{L_\phi}{\sigma_\phi}$$

- *relative smoothness* and *relative strong convexity*

$$\sigma_{F/\phi} \nabla^2 \phi(x) \preceq \nabla^2 F(x) \preceq L_{F/\phi} \nabla^2 \phi(x)$$

  relative condition number

$$\kappa_{F/\phi} = \frac{L_{F/\phi}}{\sigma_{F/\phi}}$$

- *Bregman divergence*

$$D_\phi(x, y) \triangleq \phi(x) - \phi(y) - \nabla \phi(y)^\top (x - y)$$

  relative smoothness and *relative strong convexity*

$$\sigma_{F/\phi} D_\phi(x, y) \leq D_F(x, y) \leq L_{F/\phi} D_\phi(x, y)$$

Preconditioned proximal gradient method

- replace $(1/2)\|x - x_t\|^2$ by $D_\phi(x, x_t)$

$$x_{t+1} = \arg\min_{x \in \mathcal{R}^d} \left\{ \nabla F(x_t)^\top x + \psi(x) + \frac{1}{\eta_t} D_\phi(x, x_t) \right\}$$

- convergence rate with $\eta_t = 1/L_{F/\phi}$

$$\Phi(x_t) - \Phi(x_*) \le \left(1 - \kappa_{F/\phi}^{-1}\right)^t L_{F/\phi} D_\phi(x_*, x_0)$$

Preconditioned proximal gradient method

- replace $(1/2)\|x - x_t\|^2$ by $D_\phi(x, x_t)$

$$x_{t+1} = \arg\min_{x \in \mathcal{R}^d} \left\{ \nabla F(x_t)^\top x + \psi(x) + \frac{1}{\eta_t} D_\phi(x, x_t) \right\}$$

- convergence rate with $\eta_t = 1/L_{F/\phi}$

$$\Phi(x_t) - \Phi(x_*) \leq \left(1 - \kappa_{F/\phi}^{-1}\right)^t L_{F/\phi} D_\phi(x_*, x_0)$$

- distributed ERM: $F = \frac{1}{m} \sum_{i=1}^m f_i(x)$

$$\phi(x) = f_1(x) + \frac{\mu}{2}\|x\|^2$$

  - $\kappa_{F/\phi}$ depends on quality of approximating F by f_1
  - parameter μ determined by approximation quality

(DANE, Shamir-Srebro-Zhang 2014)

Statistical preconditioning

- if $\left\|\nabla^2 f_1(x) - \nabla^2 F(x)\right\| \leq \mu$ and $\phi(x) = f_1(x) + \frac{\mu}{2}\|x\|^2$, then

$$\frac{\sigma_F}{\sigma_F + 2\mu}\nabla^2\phi(x) \preceq \nabla^2 F(x) \preceq \nabla^2\phi(x)$$

Statistical preconditioning

- if $\left\|\nabla^2 f_1(x) - \nabla^2 F(x)\right\| \leq \mu$ and $\phi(x) = f_1(x) + \frac{\mu}{2}\|x\|^2$, then

$$\frac{\sigma_F}{\sigma_F + 2\mu}\nabla^2\phi(x) \preceq \nabla^2 F(x) \preceq \nabla^2\phi(x)$$

- apply matrix Hoeffding with $\nabla^2 f_1(x) = \frac{1}{n}\sum_{i=1}^n \nabla^2\ell(x, z_i)$

w.p. $1-\delta$, $$\|\nabla^2 f_1(x) - \nabla^2 F(x)\| \leq \sqrt{\frac{32L_\ell^2 \log(d/\delta)}{n}} \qquad (*)$$

therefore $\mu = \widetilde{O}(L_\ell/\sqrt{n})$, where $L_\ell \geq \|\nabla^2\ell(x, z_i)\|$

Statistical preconditioning

- if $\left\|\nabla^2 f_1(x) - \nabla^2 F(x)\right\| \le \mu$ and $\phi(x) = f_1(x) + \frac{\mu}{2}\|x\|^2$, then

$$\frac{\sigma_F}{\sigma_F + 2\mu}\nabla^2\phi(x) \preceq \nabla^2 F(x) \preceq \nabla^2\phi(x)$$

- apply matrix Hoeffding with $\nabla^2 f_1(x) = \frac{1}{n}\sum_{i=1}^n \nabla^2\ell(x, z_i)$

w.p. $1-\delta$, $$\|\nabla^2 f_1(x) - \nabla^2 F(x)\| \le \sqrt{\frac{32 L_\ell^2 \log(d/\delta)}{n}} \qquad (*)$$

therefore $\mu = \widetilde{O}(L_\ell/\sqrt{n})$, where $L_\ell \ge \|\nabla^2\ell(x, z_i)\|$

- relative condition number (assuming $\sigma_F \approx \sigma_\ell \approx \lambda$):

$$\kappa_{F/\phi} = \frac{\sigma_F + 2\mu}{\sigma_F} = 1 + \widetilde{O}\left(\frac{\kappa_\ell}{\sqrt{n}}\right)$$

for large n, we have $\kappa_{F/\phi} < \kappa_F$

Quadratic vs non-quadratic

caveat: need ($*$) hold for all $x \in \mathrm{dom}\psi$ with high probability

$$\left\| \nabla^2 f_1(x) - \nabla^2 F(x) \right\| \leq \mu, \qquad \forall\, x \in \mathrm{dom}\psi$$

Quadratic vs non-quadratic

caveat: need ($*$) hold for all $x \in \mathrm{dom}\psi$ with high probability

$$\left\|\nabla^2 f_1(x) - \nabla^2 F(x)\right\| \leq \mu, \qquad \forall\, x \in \mathrm{dom}\psi$$

- quadratic loss $\ell_i(x) = (a_i^T x - b_i)^2/2$
  - $\nabla^2 f_1$ and $\nabla^2 F$ independent of x
  - relative condition number $\kappa_{F/\phi} = 1 + \widetilde{O}\left(\frac{\kappa_\ell}{\sqrt{n}}\right)$

Quadratic vs non-quadratic

caveat: need ($*$) hold for all $x \in \mathbf{dom}\psi$ with high probability

$$\left\|\nabla^2 f_1(x) - \nabla^2 F(x)\right\| \le \mu, \qquad \forall\, x \in \mathbf{dom}\psi$$

- quadratic loss $\ell_i(x) = (a_i^T x - b_i)^2/2$
  - $\nabla^2 f_1$ and $\nabla^2 F$ independent of x
  - relative condition number $\kappa_{F/\phi} = 1 + \widetilde{O}\left(\frac{\kappa_\ell}{\sqrt{n}}\right)$
- non-quadratic loss
  - $\nabla^2 f_1(x)$ and $\nabla^2 F(x)$ depends on x
  - ball-packing + union bound encounter additional $\sqrt{d}$ factor
  - relative condition number $\kappa_{F/\phi} = 1 + \widetilde{O}\left(\frac{\kappa_\ell\sqrt{d}}{\sqrt{n}}\right)$

  (benefit of preconditioning may degrade in high dimension)

Result 1: preconditioned APG method

convergence rate:

$$\Phi(x_t) - \Phi(x_*) \le \prod_{\tau=1}^{t} \left(1 - \frac{1}{\sqrt{\kappa_{F/\phi} G_\tau}}\right) L_{F/\phi} D_\phi(x_*, x_0),$$

- $G_t = 1$ for quadratics, otherwise $G_t \to 1$ geometrically
- $1 \le G_t \le \kappa_\phi$, thus $\kappa_{F/\phi} G_\tau \le \kappa_{F/\phi} \kappa_\phi \approx \kappa_F$
- G_t calculated at each iteration, serve as numerical certificate
- in practice $G \approx 1$, empirical complexity $O(\sqrt{\kappa_{F/\phi}} \log(1/\epsilon))$

Result 1: preconditioned APG method

convergence rate:

$$\Phi(x_t) - \Phi(x_*) \leq \prod_{\tau=1}^{t} \left(1 - \frac{1}{\sqrt{\kappa_{F/\phi} G_\tau}}\right) L_{F/\phi} D_\phi(x_*, x_0),$$

- $G_t = 1$ for quadratics, otherwise $G_t \to 1$ geometrically
- $1 \leq G_t \leq \kappa_\phi$, thus $\kappa_{F/\phi} G_\tau \leq \kappa_{F/\phi}\kappa_\phi \approx \kappa_F$
- G_t calculated at each iteration, serve as numerical certificate
- in practice $G \approx 1$, empirical complexity $O(\sqrt{\kappa_{F/\phi}} \log(1/\epsilon))$

theoretical challenge

- acceleration in relative smooth/s.c. setting is difficult
  (negative result by Dragomir-Taylor-d'Aspremont-Bolte 2019)
- we obtain *asymptotic acceleration* when ϕ strongly convex

Result 2: improved bounds on statistical preconditioning

focus on linear prediction models

$$\ell(x,(a_i,b_i)) = \ell_i(a_i^T x) + \frac{\lambda}{2}\|x\|^2$$

where $\|a_i\|^2 \leq R$

- for quadratic losses, improve by factor $\sqrt{n}$

$$\kappa_{F/\phi} = \frac{3}{2} + O\left(\frac{R^2}{n\lambda}\log\left(\frac{d}{\delta}\right)\right)$$

- for non-quadratics, remove dependence on d

$$\kappa_{F/\phi} = 1 + O\left(\frac{R^2}{\sqrt{n}\lambda}\left(RD + \sqrt{\log(1/\delta)}\right)\right)$$

  where D is the diameter of $\mathrm{dom}\phi$ (bounded domain).

improve across the board: $O(\kappa_{F/\phi}\log(1/\epsilon))$ or $O(\sqrt{\kappa_{F/\phi}}\log(1/\epsilon))$

SPAG algorithm

let $A_0 = 0, \;\; B_0 = 1$ and define sequences (need knowledge of $\sigma_{F/\phi}$)

$$A_{t+1} = A_t + a_{t+1}, \qquad B_{t+1} = B_t + a_{t+1}\sigma_{F/\phi}$$

$$\alpha_t = \frac{a_{t+1}}{A_{t+1}}, \qquad \beta_t = \frac{a_{t+1}}{B_{t+1}}\sigma_{F/\phi}, \qquad \eta_t = \frac{a_{t+1}}{B_{t+1}}$$

SPAG algorithm

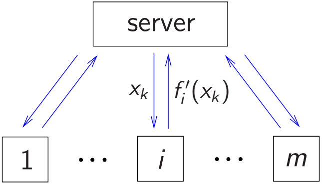
let $A_0 = 0$, $B_0 = 1$ and define sequences (need knowledge of $\sigma_{F/\phi}$)

$$A_{t+1} = A_t + a_{t+1}, \qquad B_{t+1} = B_t + a_{t+1}\sigma_{F/\phi}$$

$$\alpha_t = \frac{a_{t+1}}{A_{t+1}}, \qquad \beta_t = \frac{a_{t+1}}{B_{t+1}}\sigma_{F/\phi}, \qquad \eta_t = \frac{a_{t+1}}{B_{t+1}}$$

$v_0 = x_0$, $G_{-1} = 1$
for $t = 0, 1, 2, \ldots$ **do**
  $G_t = \max\{1, G_{t-1}/2\}/2$
  repeat
    $G_t \leftarrow 2G_t$
    Find a_{t+1} such that $a_{t+1}^2 L_{F/\phi} G_t = A_{t+1}B_{t+1}$
    $y_t = \frac{1}{1-\alpha_t\beta_t}\big((1-\alpha_t)x_t + \alpha_t(1-\beta_t)v_t\big)$
    Compute $\nabla F(y_t)$ *(requires communication if distributed)*
    $v_{t+1} = \arg\min_x\left\{\nabla F(y_t)^\top x + \psi(x) + \frac{1-\beta_t}{\eta_t}D_\phi(x, v_t) + \frac{\beta_t}{\eta_t}D_\phi(x, y_t)\right\}$
    $x_{t+1} = (1-\alpha_t)x_t + \alpha_t v_{t+1}$
  until gain search criterion is satisfied
end for

SPAG for distributed optimization



server running SPAG

broadcast / reduce

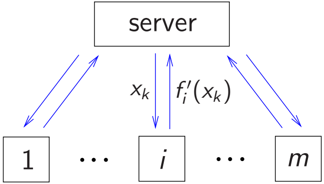
compute $f_i'(x_k)$, $i = 1, \ldots, m$

preconditioning step at server

$$v_{t+1} = \arg\min_x \Big\{ \nabla F(y_t)^\top x + \psi(x) + \frac{1-\beta_t}{\eta_t} D_\phi(x, v_t) + \frac{\beta_t}{\eta_t} D_\phi(x, y_t) \Big\}$$

- recall $\phi(x) = \frac{1}{n}\sum_{i=1}^n \ell(x, z_i) + \frac{\lambda+\mu}{2}\|x\|^2$
- equivalent to solve ERM with n samples and larger regularization
- $O((n + \kappa_\phi)\log(1/\epsilon'))$ complexity (SDCA, SVRG, SAGA, ...)

SPAG for distributed optimization



server running SPAG

broadcast / reduce

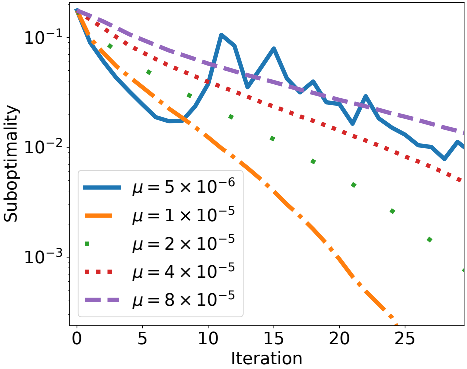
compute $f_i'(x_k)$, $i = 1, \ldots, m$

preconditioning step at server

$$v_{t+1} = \arg\min_x \left\{ \nabla F(y_t)^\top x + \psi(x) + \frac{1-\beta_t}{\eta_t} D_\phi(x, v_t) + \frac{\beta_t}{\eta_t} D_\phi(x, y_t) \right\}$$

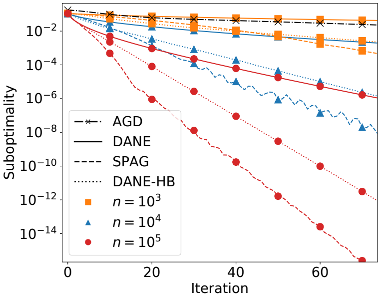
- recall $\phi(x) = \frac{1}{n} \sum_{i=1}^n \ell(x, z_i) + \frac{\lambda+\mu}{2} \|x\|^2$
- equivalent to solve ERM with n samples and larger regularization
- $O((n + \kappa_\phi) \log(1/\epsilon'))$ complexity (SDCA, SVRG, SAGA, ...)
  - n small $\Rightarrow$ μ large $\Rightarrow$ κ_ϕ small, $\kappa_{F/\phi}$ large $\Rightarrow$ compute $\downarrow$, comm. $\uparrow$
  - n large $\Rightarrow$ μ small $\Rightarrow$ κ_ϕ large, $\kappa_{F/\phi}$ small $\Rightarrow$ compute $\uparrow$, comm. $\downarrow$

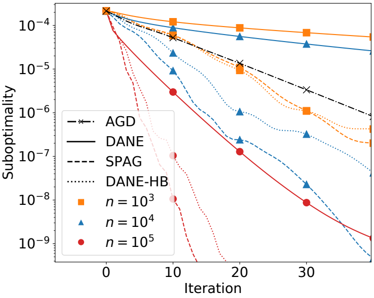
SPAG experiments



- logistic regression on RCV1: $d = 47236$, $N = 677399$, $\lambda = 10^{-7}$
- effect of μ on convergence speed with fixed samples $n = 10^4$

Experiments





- left: logistic regression on RCV1: $\lambda = 10^{-7}$, $\mu = 0.1/n$
- right: KDD2010 ($d = 20,216,830$, $N = 7,557,074$)

DANE (Shamir-Srebro-Zhang 2014), DANE-HB (Yuan-Li 2019)

Summary

Statistical optimization methods
(optimization methods powered by statistics)

this talk

- hypothesis testing for automatic tuning learning rate
  (Lang, Zhang & X. 2019; Zhang, Lang, Liu & X. 2020)
- variance reduction for structured nonconvex optimization
  (Zhang & X. 2018, 2019, 2020)
- statistical preconditioning for distributed optimization
  (Hendrikx, X., Bubeck, Bach & Massoulié 2020)

other recent work

- proximal boosting for high probability stochastic optimization
  (Davis, Drusvyatskiy, X. & Zhang 2019)