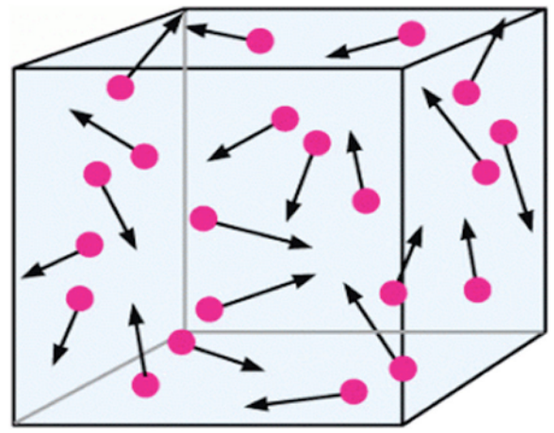
From an ODE for Nesterov's method to Accelerated Stochastic Gradient descent

Adam Oberman
with Maxime Laborde
Math and Stats, McGill

Stochastic Gradient Descent definition: Math vs. ML

- Before 2010: SGD means Stochastic Differential Equations and Brownian Motion



- Since around 2010: SGD algorithm for training ML models

The data sizes have grown faster than the speed of processors … methods limited by the computing time rather than the sample size …. Unlikely optimization algorithms such as stochastic gradient descent show amazing performance for large-scale problems,[*Large-Scale Machine Learning with Stochastic Gradient Descent,* Leon Bottou],

Empirical Loss Minimization problem

$$\min_{w \in \mathbb{R}^D} L(w) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(f(x_i, w), y_i).$$

w - parameters of the model f. L loss function measuring how well model fits the labels y of the data x.

Optimization for Machine Learning: SGD

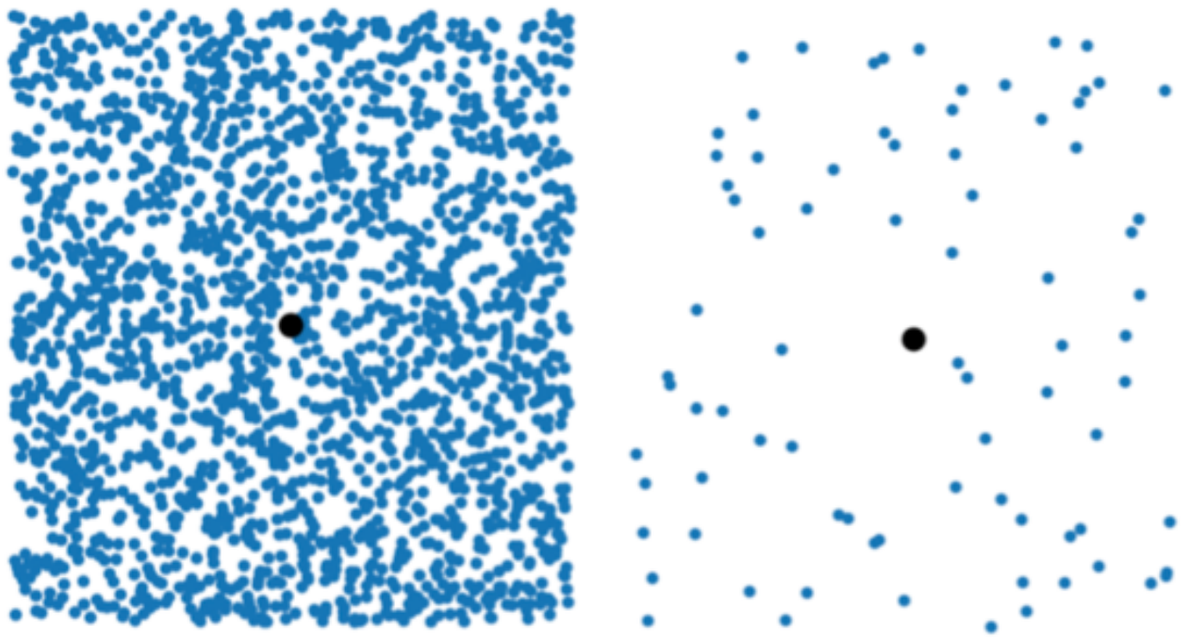
- Why is stochastic gradient important for machine learning?

Evaluating the full loss (and gradient) on all m data points can be too costly. Instead define a random minibatch $I \subset \{1, \ldots, m\}$

$$L_I(w) = \frac{1}{|I|} \sum_{i \in I} \mathcal{L}(f(x_i, w), y_i)$$

Stochastic gradient descent corresponds to

$$w^{k+1} = w^k - h_k \nabla_w L_{I_k}(w^k), \quad I_k \text{ random, } h_k \text{ learning rate}$$



Full batch (entire data set) and mini-batch. SGD corresponds to gradient descent approximating the objective by an average over a random subsample of the data set.

SGD algorithms

- Why not Newton's method?
  - Small scale numerical methods: have enough memory to solve NxN linear system (e.g. Newton's method). Solve in few iterations.
- SGD Trade-off: lots of iterations (millions) of a slow but memory-cheap algorithm.

Stochastic Approximation

- The minibatch stochastic gradient is harder to analyze
- Instead make the Stochastic Approximation assumption

$$\widehat{g}(x,\xi) = \nabla f(x) + e(x,\xi),$$

$$\mathbb{E}[e] = 0 \text{ and } \mathrm{Var}(e) = \sigma^2.$$

SGD convergence rate

From Bottou, Curtis, Nocedal (2016),

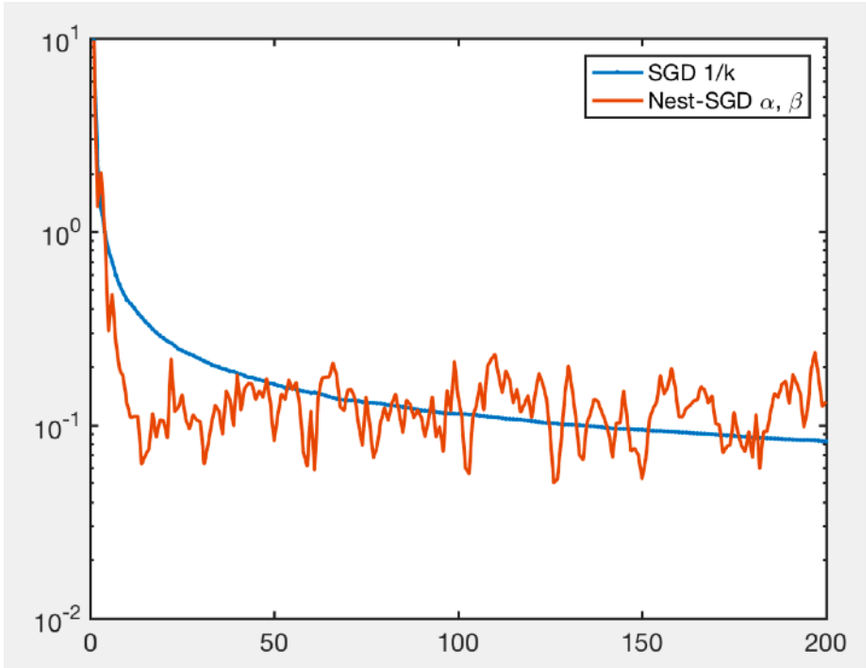
$$\mathbb{E}[f(x_k) - f^*] \leqslant C_1(1 - \mu h)^k + C_2 h\sigma^2$$

Need: $h \searrow 0$ or $\sigma^2 \searrow 0$.

Number of influential variance reduction algorithms to speed up SGD:

- SAG (Schmidt, Le Roux, Bach, 2013), **Lagrange Prize*
- SAGA (Defazio, Bach, Lacoste-Julien, 2014)
- SVRG (Jonhson, Zhang, 2013),

However, in many modern applications, variance reduction does not help.

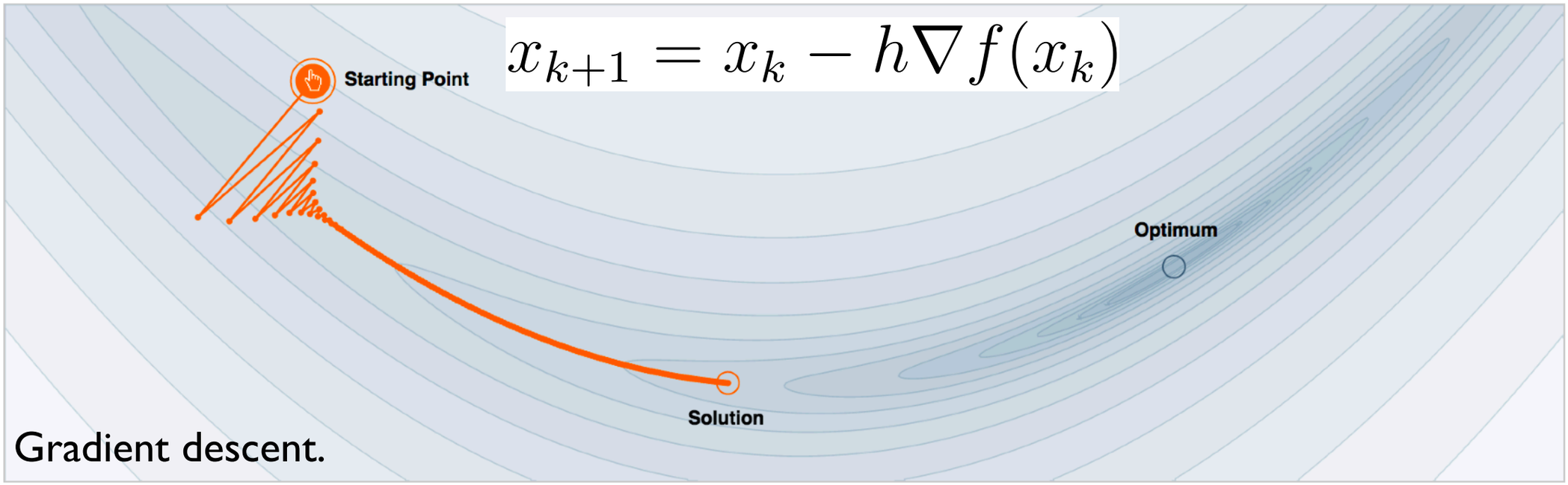


SGD convergence, rate means O(10k) iterations to decrease objective gap by factor of 10. Thus improving rate constant can reduce iterations by 10X.

Effective heuristic: "momentum". Not theoretically understood.

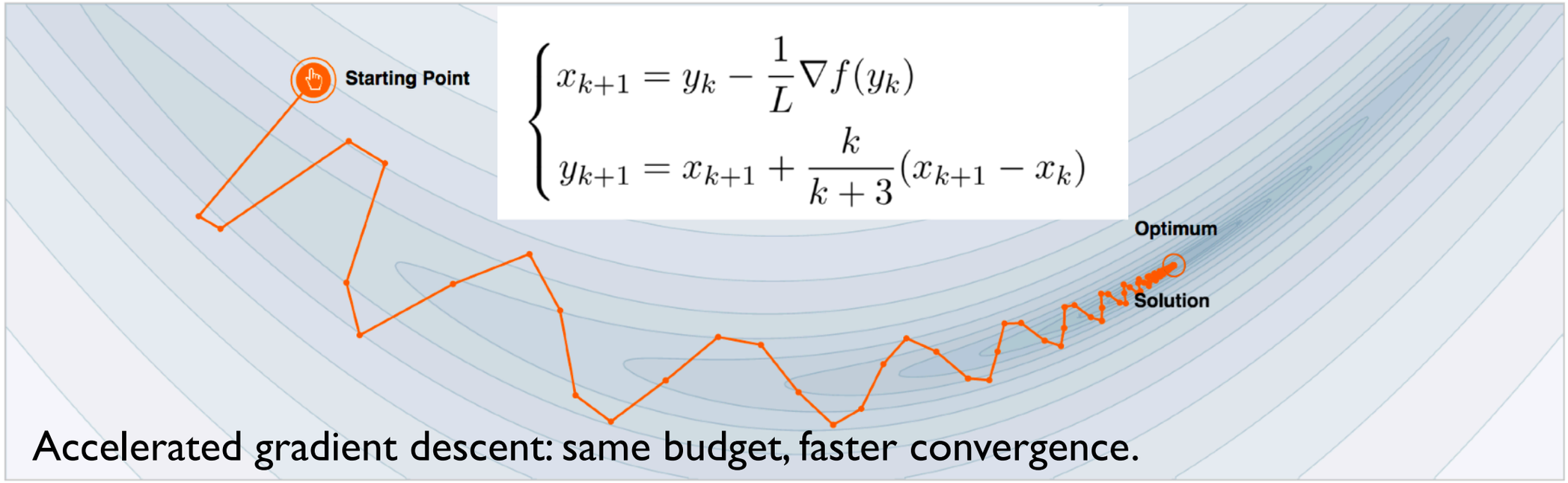
Nesterov's accelerated (full) gradient descent

$$x_{k+1} = x_k - h\nabla f(x_k)$$



Gradient descent.

$$\begin{cases} x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k) \\ y_{k+1} = x_{k+1} + \frac{k}{k+3}(x_{k+1} - x_k) \end{cases}$$



Accelerated gradient descent: same budget, faster convergence.

https://distill.pub/2017/momentum/

Heuristic: momentum term remembers old gradients, overshoots instead of getting stuck.

Remark: *two main A-GD algorithms, correspond to convex and strongly convex case. We focus on one, convex case, to simplify presentation. Strongly convex case is also covered.*

Motivation: A-GD and SGD Algorithms

$$x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$$
$$y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k)$$

$$\beta_k = \frac{k}{k+1}, \quad \beta_k = \frac{\sqrt{C}-1}{\sqrt{C}+1}$$

A-GD convex, strongly convex case

$$x_{k+1} = x_k - \frac{h_k}{L}\hat{g}(x_k),$$

$$\hat{g} = \nabla f + e$$

$$h_k = \frac{2\sqrt{C}}{k}, \quad h_k = \frac{c}{k^{1/2}},$$

SGD convex, strongly convex case

Heuristic A-SGD:

$$x_{k+1} = y_k - \frac{h_k}{L}\hat{g}_k$$
$$y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k)$$

Q: Can we find parameters to achieve faster convergence?
Need to understand A-GD in a way than generalizes to stochastic case.

Heuristic: derive accelerated-GD algorithm

$$x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$$

double variables and introduce new speed parameter

$$v_{k+1} = v_k - \frac{1}{wL}\nabla f(v_k)$$

Second equation is unstable.

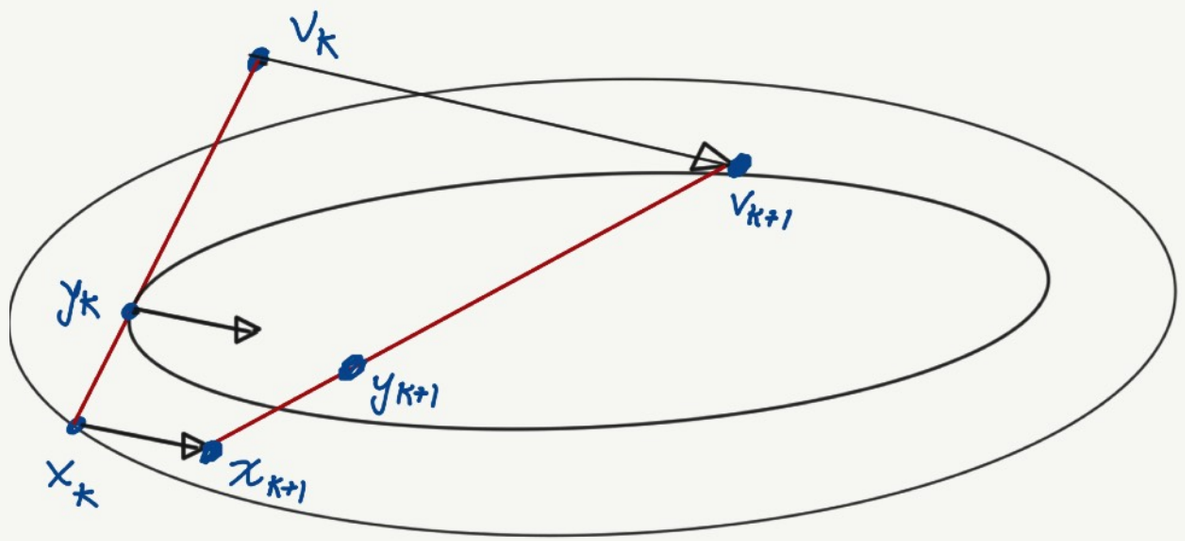
(i) Couple equations

(ii) reduce to single gradient evaluation at convex combination

$$x_{k+1} - x_k = w(v_k - x_k) - \frac{1}{L}\nabla f(y_k)$$

$$v_{k+1} - v_k = -\frac{1}{wL}\nabla f(y_k)$$

$$y_k = (1-w)x_k + w v_k$$



Derivation (step 2)

$$x_{k+1} - x_k = w(v_k - x_k) - \frac{1}{L}\nabla f(y_k)$$

$$v_{k+1} - v_k = -\frac{1}{wL}\nabla f(y_k)$$

$$y_k = (1-w)x_k + w v_k$$

Eliminate v:

① $\quad x_{k+1} = y_k - g_k$

$w v_{k+1} = w v_k - g_k$

$y_{k+1} - (1-w)x_{k+1} \quad y_k - (1-w)x_k \quad x_{k+1} - y_k$

② $\Rightarrow y_{k+1} = x_{k+1} + (1-w)(x_{k+1} - x_k)$

Set $w = w_k = 3/k+3 \Rightarrow (1-w) = k/k+3$

obtain Nesterov AG in Convex Case.

Case 2: modify derivation

$$x_{k+1} - x_k = w(v_k - x_k) - \frac{1}{L}\nabla f(y_k)$$

$$v_{k+1} - v_k = w(x_k - v_k) - \frac{(1-w)}{wL}\nabla f(y_k)$$

$$y_k = (1-w)x_k + w v_k$$

ELIMINATE v, as before

$$\Rightarrow \quad x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$$

$$y_{k+1} = x_{k+1} + (1-2w)(x_{k+1} - x_k)$$

set $w = \frac{1}{1+\sqrt{C}}$, $1-2w = \frac{\sqrt{C}-1}{\sqrt{C}+1}$

recover A-GD strongly convex

Accelerated-SGD Algorithm

In the sequel, we generalize the *Lyapunov analysis* for A-GD to the stochastic case. Analysis will give the best constant, and yield the following A-SGD algorithm.

$$x_{k+1} = y_k - \frac{h_k}{L}\hat{g}_k$$

$$y_{k+1} = x_{k+1} + \beta_k (x_{k+1} - x_k)$$

Strongly Convex:

$h_k = \frac{2\sqrt{c}}{k}$, $\beta_k = \frac{\sqrt{c} - h_k}{\sqrt{c} + h_k}$

SGD AGD $\frac{\sqrt{c}-1}{\sqrt{c}+1}$

Convex

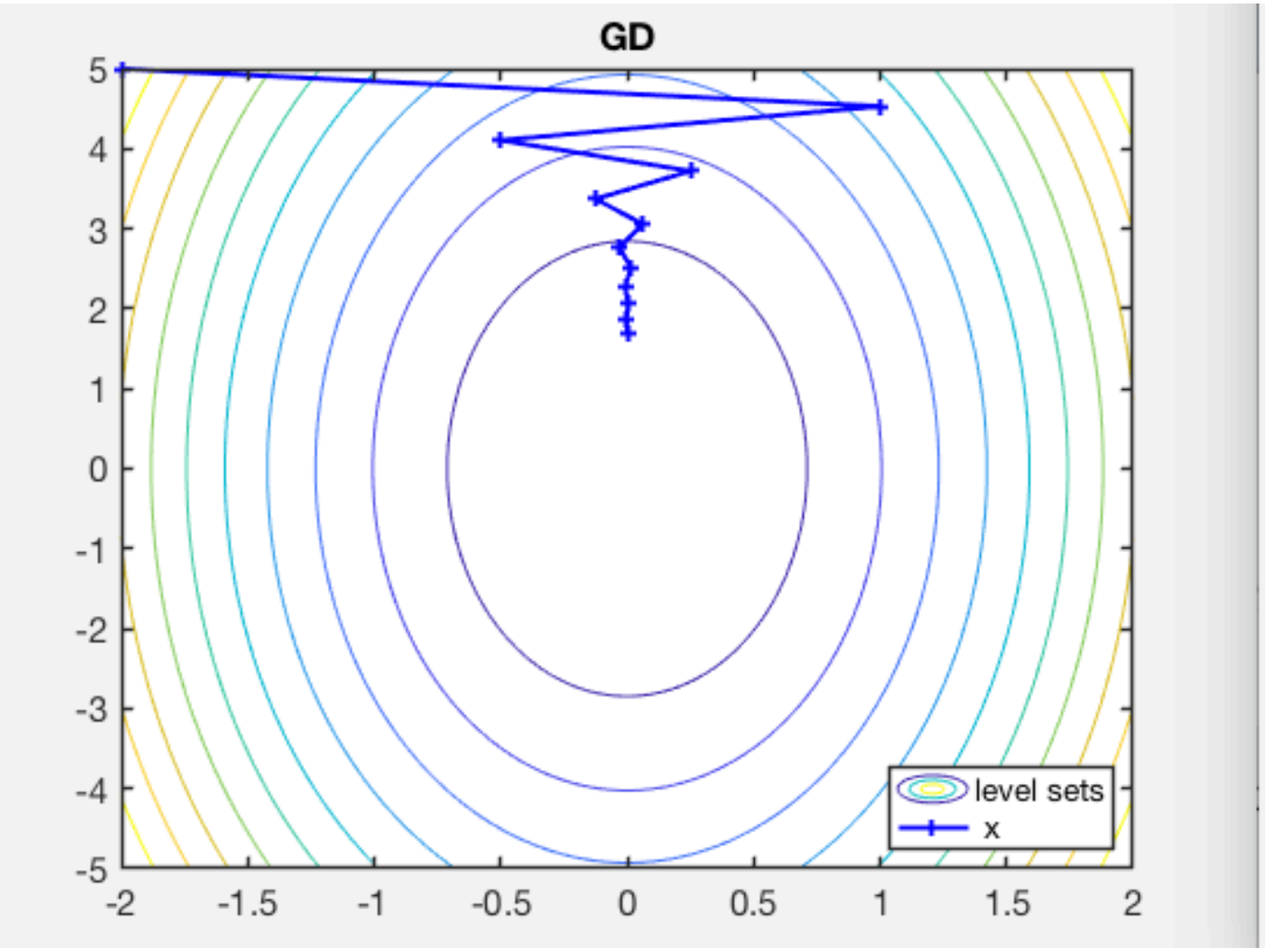
$h_k = \frac{c}{k^{3/4}}$ $\beta_k = \frac{1}{1 + \frac{3h_k}{t_k}}$ $t_k = \sum_{i=1}^{k} h_i$

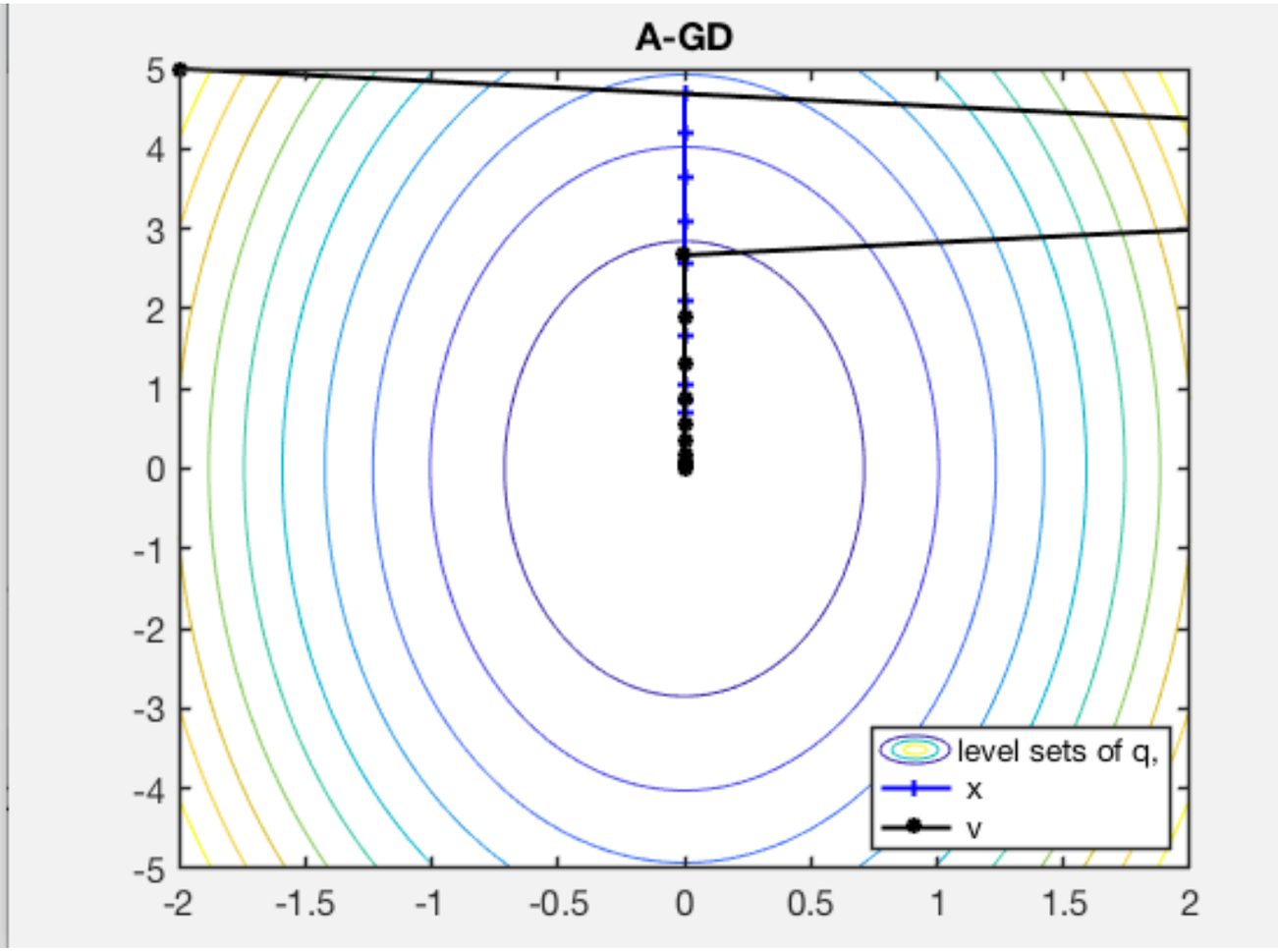
SGD $h_k = \frac{c}{k^{1/2}}$ AGD $\beta_k = \frac{1}{1 + 3/k}$

with an improved convergence rate constant (TBD)

visualize algorithm in 2d (full gradients)

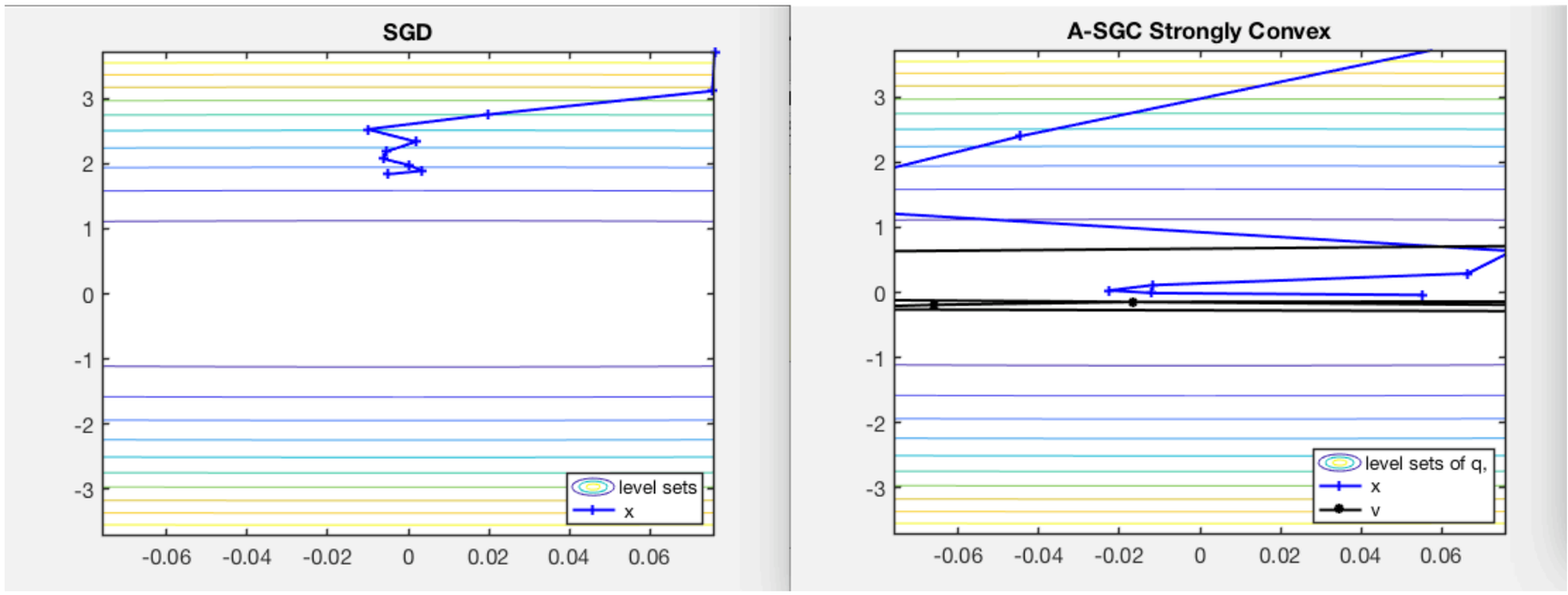
- after one iteration, x and v are in the shallow valley.
- (when properly tuned) the faster v variable pulls x along faster than GD.





comparison GD and A-GD: (same axes)

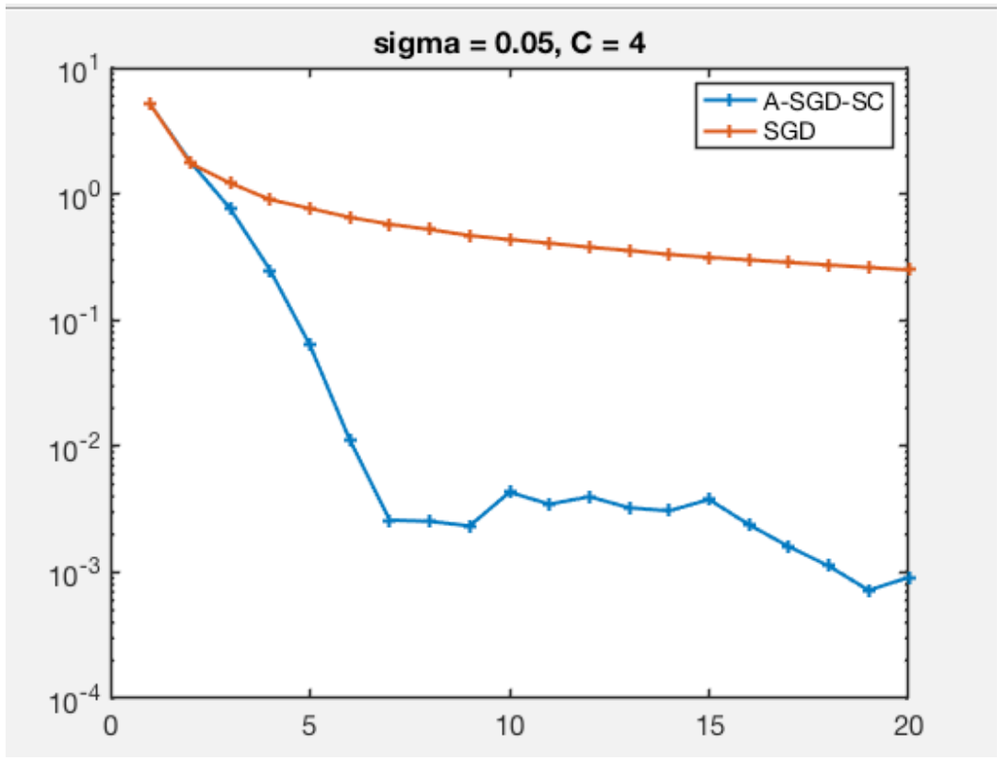
visualize algorithm in 2d (stochastic gradients)

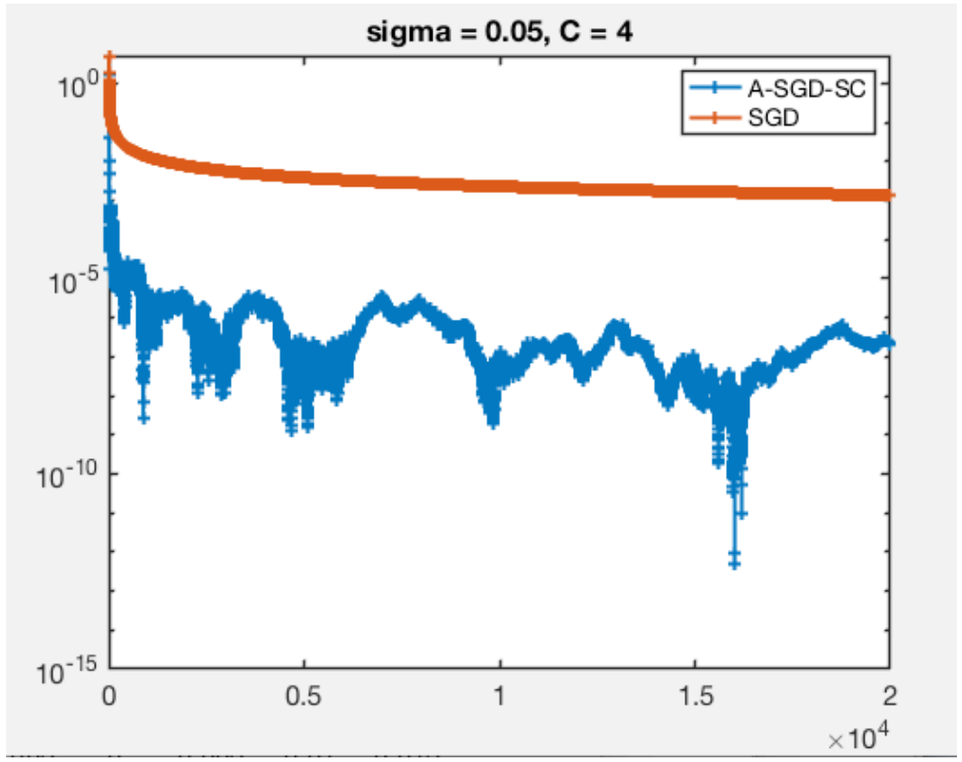


Top: comparison SGD and A-SGD: (same axes).

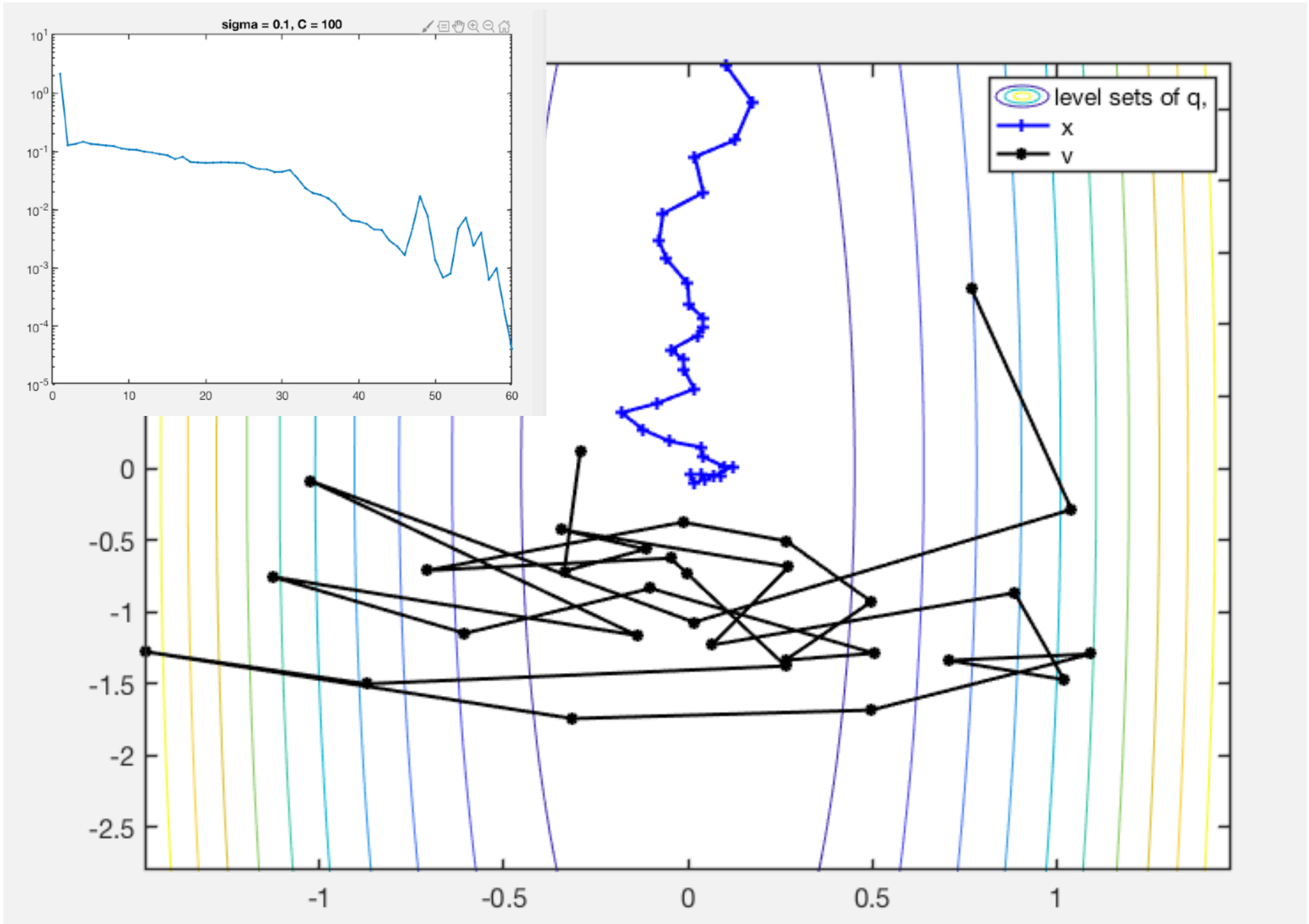
Bottom: After 20 iterations, optimality gap 100X smaller.

After 20,000 iterations, SGD reaches A-SGD at k = 20.





visualize algorithm in 2d (stochastic gradients)



- Note: the v moves faster, allows x to average
- the linear term pulls x towards v

Decreasing Learning Rate Results

Define G^2 bound on stochastic gradient: $\mathbb{E}[\widehat{g}^2] \leq G^2$

Remark: $G^2 = L^2 D^2 + \sigma^2$ where D is the diameter of the domain

- Strongly convex case: Learning rate $h_k = \mathcal{O}\left(\frac{1}{k}\right)$
  - Previous results: Nemirovski, Juditsky, Lan, and Shapiro (2009), Shamir, Zhang (2013), Jain, Kakade, Netrapalli, Sidford (2018): optimal rate $\mathcal{O}\left(\frac{1}{k}\right)$ with constants depending on G^2, D and μ.
  - Our result: $\mathcal{O}\left(\frac{1}{k}\right)$ rate with constants independent of the L-smoothness bound of the gradient (depends only on σ^2 and μ).
- Convex case:
  - Shamir, Zhang (2013): optimal rate order: $\frac{\log(k)}{\sqrt{k}}$ with a rate constant that depends on G^2 and D. Learning rate: $h_k = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$.
  - Jain, Nagaraj, Netrapalli (2019) remove the log factor assuming that the number of iterations is decided in advance.
  - Our result: $\mathcal{O}(\log(k)/\sqrt{k})$ rate for the last iterate, with a constant which depends on σ^2, but independent of the L-smoothness. New learning rate schedule

Strongly Convex A-SGD Rate:

Comparison with previous results

Learning rate: $h_k = \mathcal{O}\left(\frac{1}{k}\right)$ $\quad G^2 = L^2 + \sigma^2$

Nemirovski et al. [2009]	Shamir and Zhang [2013]	Jain et al. [2019]	Acc. SGD
$\frac{2C_f G^2}{\mu k}$	$\frac{17G^2(1+\log(k))}{\mu k}$	$\frac{130G^2}{\mu k}$	$\frac{4\sigma^2}{\mu k + 4\sigma^2 E_0^{-1}}$

Convergence rate $\mathbb{E}[f(x_k) - f^*]$ after k steps: G^2 is a bound on $\mathbb{E}[\hat{g}(x)^2]$, and σ^2 variance. E_0 is the initial value of the Lyapunov function

Improvement to the rate constant: independent of L

Convex A-SGD algorithm Rate:

Comparison with previous results

$$G^2 = L^2 + \sigma^2$$

	Shamir, Zhang (2013)	Acc. SGD
h_k	$\frac{c^2}{\sqrt{k}}$	$\frac{c}{k^{3/4}}$
Rate	$\left(\frac{D^2}{c^2} + c^2 G^2\right) \frac{(2+\log(k))}{\sqrt{k}}$	$\frac{\frac{E_0}{16c^2} + c^2\sigma^2(1+\log(k))}{(k^{1/4}-1)^2}$

Table: Convergence rate $\mathbb{E}[f(x_k) - f^*]$ after k steps. G^2 is a bound on $\mathbb{E}[\hat{g}^2]$. E_0 is the initial value of the Lyapunov function.

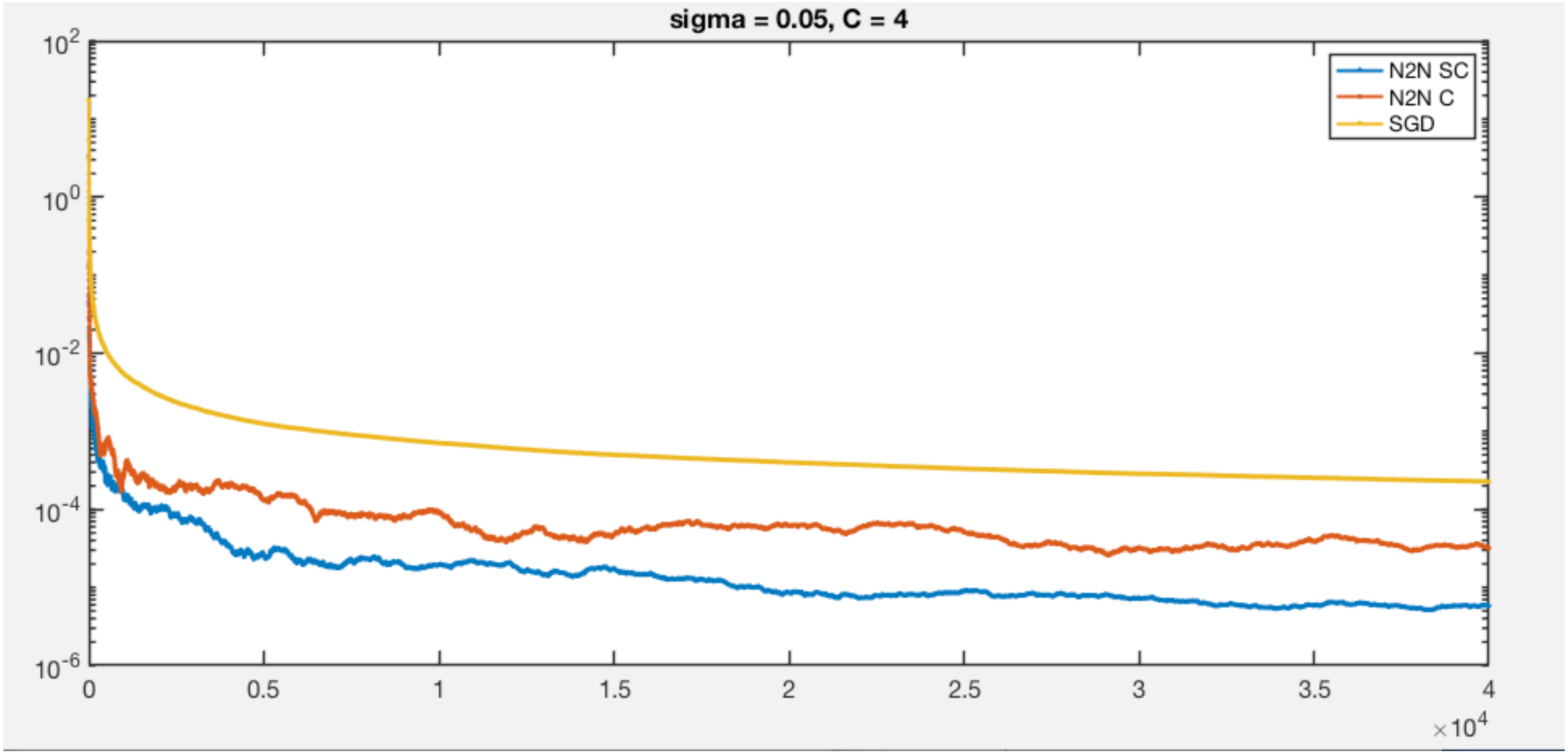
Interpreting Improved rate: remove dependence on L-smoothness from the rate

Numerical Results (preliminary):

- Quadratic with synthetic noise
- Logistic Regression with minibatch stochastic gradients.

- How much difference should the rate constant make in this case?
- Note: exponential convergence, rate constant not important.
- But c/k rate, constant can be important, (since, e.g, factor of 10 can mean 10X more iterations)

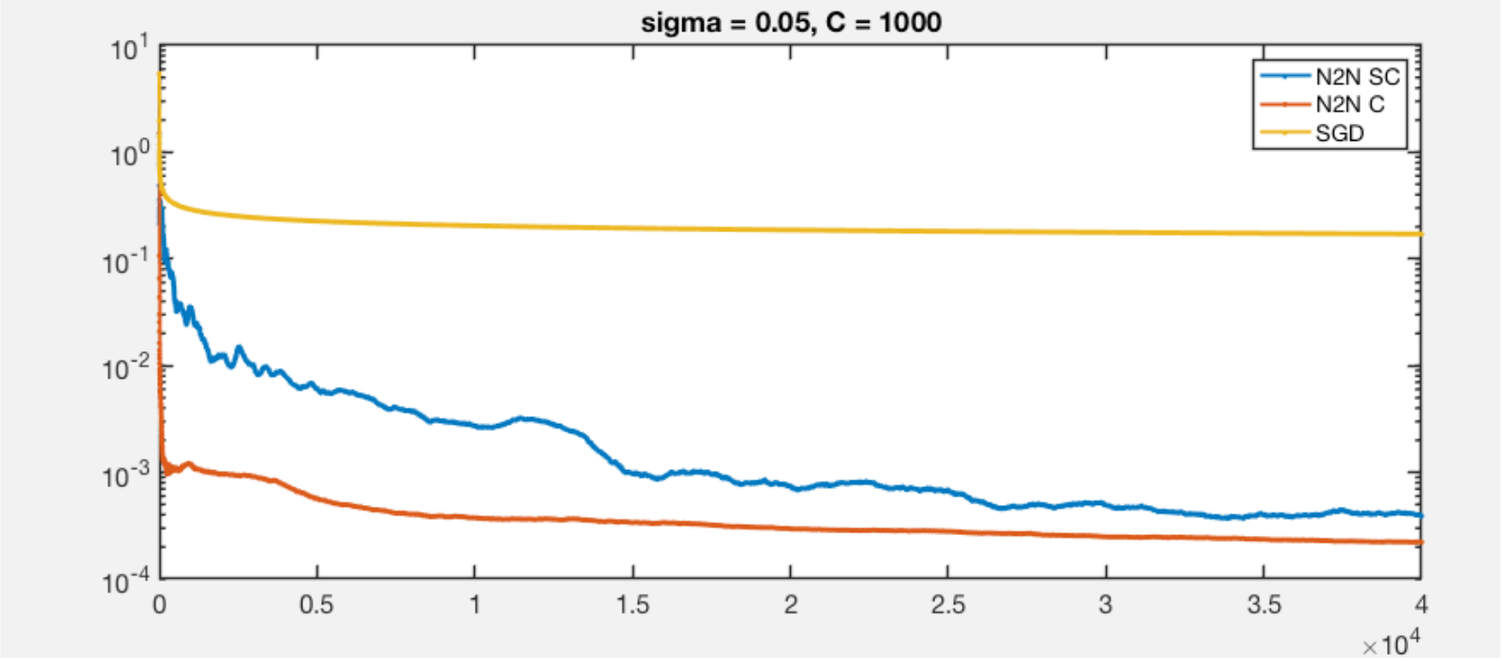
Quadratic, sigma = .05, C=4



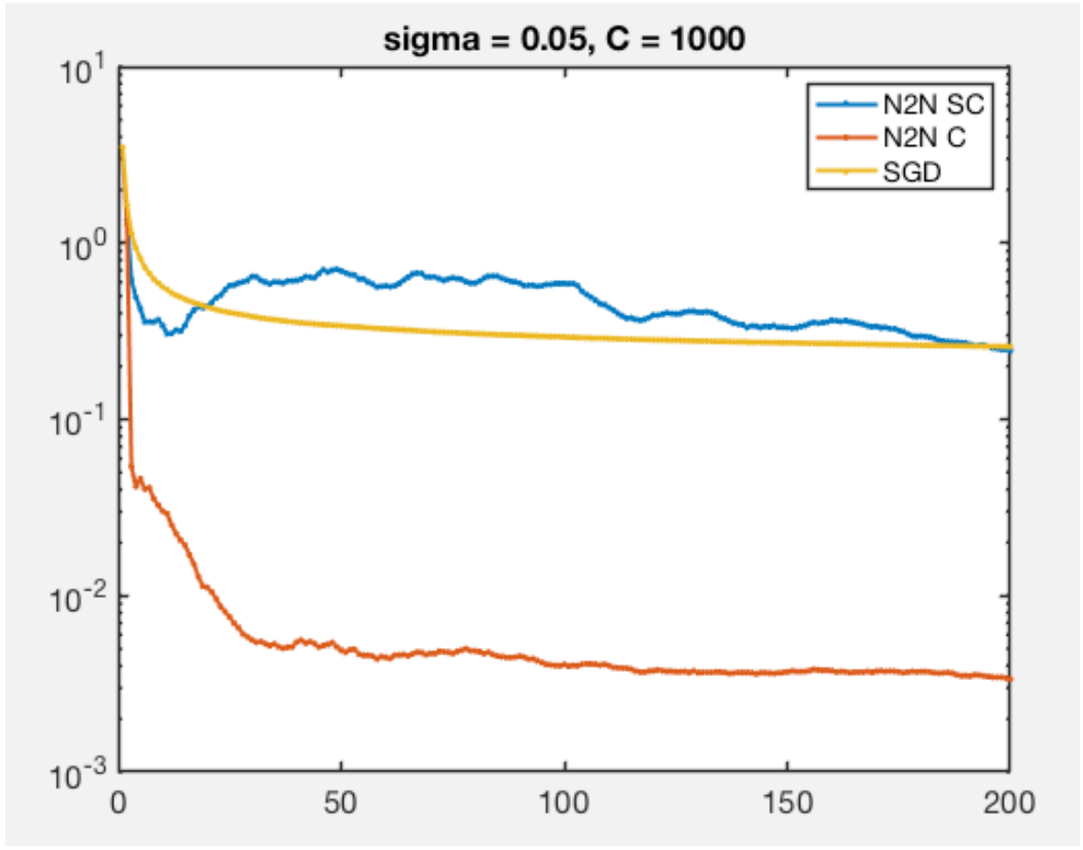
- Small condition number. Long time.
- Acc Convex is 10X better than SGD
- Acc-Strongly convex algorithm is 10X better than Acc Convex

Quadratic C=1000

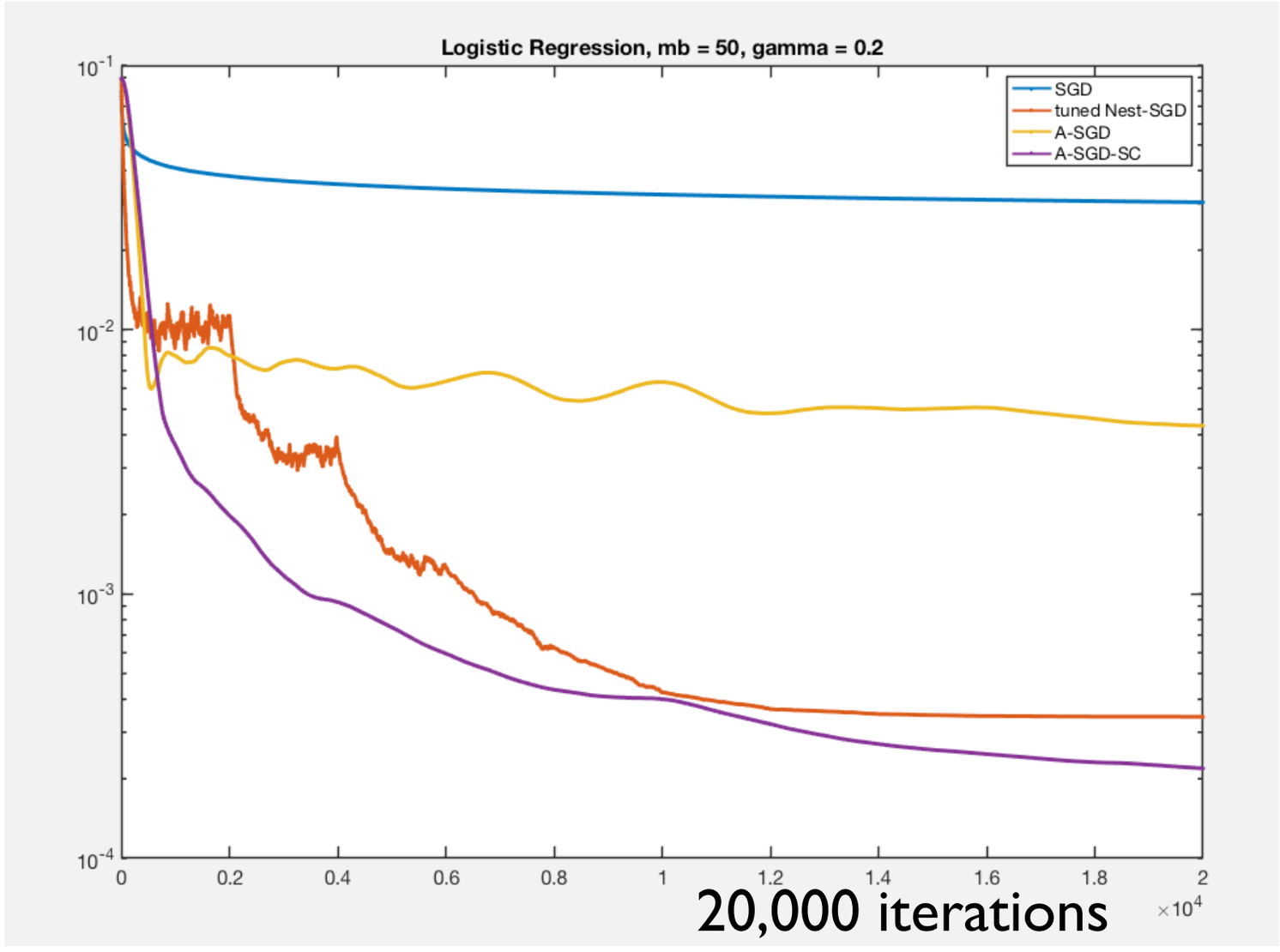
- 40,000 iterations:
- A-SGD-C >>A-SGD SC, >> SGD.



- A-SGD Convex: very fast initial drop (40 iterations)



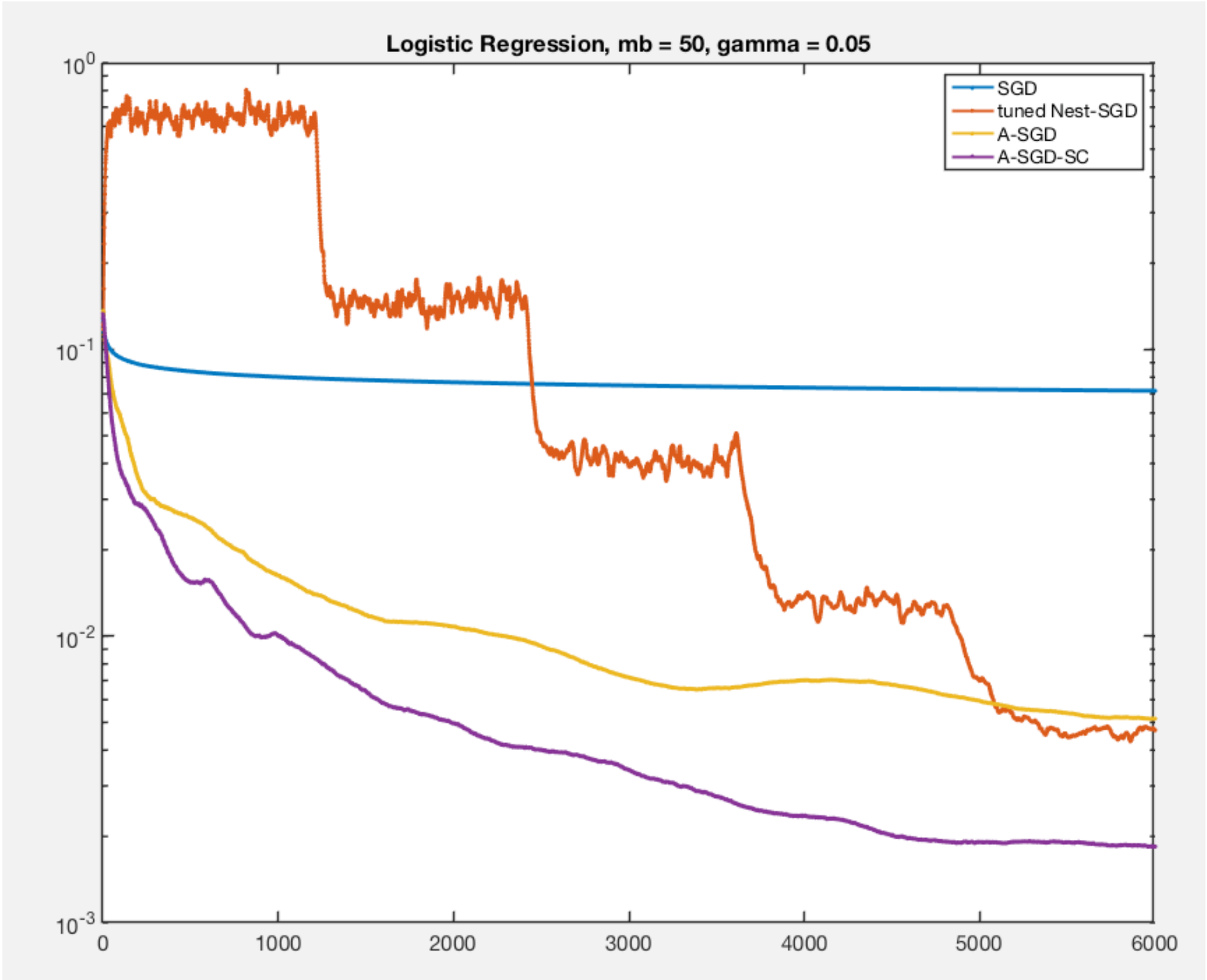
Logistic Regression I: mu = .2



Other Algorithms:

- SGD: 1/k learning rate (tuned)
- heuristic Nesterov: constant Beta, learning rate drops by constant factor every few epochs, tuned.

Logistic Regression 2: mu = .05



- Statement of convergence rates
- Idea of proof

Strong convexity, L-smoothness

- We use definitions equivalent to standard ones of strong convexity and L-smoothness

Let f be proper convex function, $x_* = \operatorname{argmin}_x f(x)$ and $f^* = f(x_*)$.

$$f(y) - f(x) + \nabla f(x) \cdot (x - y) \leq \frac{L}{2}|x - y|^2, \forall x, y \in \mathbb{R}^n \qquad f \text{ is } L\text{-smooth}$$

$$f(y) - f(x) + \nabla f(x) \cdot (x - y) \geq \frac{\mu}{2}|x - y|^2, \forall x, y \in \mathbb{R}^n \qquad f \text{ is } \mu\text{-strongly convex}$$

$C_f := \frac{L}{\mu}$ condition number of f.

- Example: in case where f is quadratic, the constants correspond to the smallest and largest eigenvalues of the Hessian of f.

Lyapunov approach to convergence rates

$$E^{ac,sc}(x,v) := f(x) - f^* + \frac{\mu}{2}|v - x^*|^2$$

- We know from previous analysis in full gradient case, we have the following inequality with beta = 0.

- Stochastic case, non-zero beta (as in SGD analysis)

$$E(t_{k+1}, z_{k+1}) \leq (1 - r_E h_k) E(t_k, z_k) + h_k \beta_k.$$

- SGD: balance the terms, to get a decreasing learning rate.
- SGD: beta has a factor of L in it, A-SGD: just a factor of sigma, for better rate

ODE Liapunov approach

Another ODE for Nesterov's Method

Our starting point is a perturbation of (S-A-ODE)

$$\begin{cases} \dot{x} = \frac{2}{t}(v - x) - \frac{1}{\sqrt{L}}\nabla f(x) \\ \dot{v} = -\frac{t}{2}\nabla f(x), \end{cases} \qquad \text{(1st-ODE)}$$

The system (1st-ODE) is equivalent to the following ODE

$$\ddot{x} + \frac{3}{t}\dot{x} + \nabla f(x) = -\frac{1}{\sqrt{L}}\left(D^2 f(x)\cdot\dot{x} + \frac{1}{t}\nabla f(x)\right) \qquad \text{(H-ODE)}$$

which has an additional Hessian damping term with coefficient $1/\sqrt{L}$.

- 2nd order ODE with Hessian damping: Alvarez, Attouch, Bolte, Redont (2002), Attouch, Peyrouquet, Redont (2016)
- Shi, Du, Jordan, Su (2018) introduced a family of high resolution second order ODEs which also lead to Nesterov's method: (H-ODE), special case $s = \frac{1}{\sqrt{L}}$

Strongly Convex Case

Stochastic gradient version

Learning rate: $h_k > 0$. Define

$$\begin{cases} x_{k+1} - x_k = \lambda_k(v_k - x_k) - \frac{h_k}{\sqrt{L}}(\nabla f(y_k) + e_k), \\ v_{k+1} - v_k = \lambda_k(x_k - v_k) - \frac{h_k}{\sqrt{\mu}}(\nabla f(y_k) + e_k), \\ y_k = (1-\lambda_k)x_k + \lambda_k v_k, \qquad \lambda_k = \frac{h_k\sqrt{\mu}}{1+h_k\sqrt{\mu}}. \end{cases} \tag{FE-SC}$$

Same algorithm as before, but now

- Variable learning rate
- Stochastic gradient
- To simplify: replace $\sqrt{\mu}$ by $\frac{\sqrt{\mu}}{1+h_k\sqrt{\mu}}$

Convergence result (strongly convex case)

Define the continuous time Lyapunov function

$$E^{ac,sc}(x,v) := f(x) - f^* + \frac{\mu}{2}|v - x^*|^2$$

Discrete time Lyapunov function $E_k^{ac,sc} := E^{ac,sc}(x_k, v_k)$.

Proposition (L., Oberman, 2020)

Assume $E_0^{ac,sc} \leqslant \frac{1}{\sqrt{L}}$. *If* $h_k := \frac{2\sqrt{\mu}}{\mu k + 4\sigma^2 E_0^{ac,sc\,-1}}$,

$$\mathbb{E}[f(x_k)] - f^* \leqslant \frac{4\sigma^2}{\mu k + 4\sigma^2 E_0^{ac,sc\,-1}}.$$

Accelerated-SDG Algorithm

From (1st-ODE) to Nesterov

Define the learning rate $h_k > 0$ and a discretization of the total time t_k.
The time discretization of (1st-ODE) with gradients evaluated at

$$y_k = \left(1 - \frac{2h_k}{t_k}\right) x_k + \frac{2h_k}{t_k} v_k.$$

is given by

$$\begin{cases} x_{k+1} - x_k = \dfrac{2h_k}{t_k}(v_k - x_k) - \dfrac{h_k}{\sqrt{L}} \nabla f(y_k), \\ v_{k+1} - v_k = -\dfrac{h_k t_k}{2} \nabla f(y_k), \end{cases} \tag{FE-C}$$

Proposition

The discretization of (1st-ODE) *given by* (FE-C) *with* $h_k = h = \frac{1}{\sqrt{L}}$ *and* $t_k = h(k+2)$ *is equivalent to the standard Nesterov's method* (C-Nest).

Fix learning rate, get Nesterov's algorithm

Convergence result (convex case)

Define the continuous time Lyapunov function

$$E^{ac,c}(t,x,v) := t^2(f(x) - f^*) + 2|v - x^*|^2$$

Define the discrete time Lyapunov function E_k^c by

$$E_k^{ac,c} = E^{ac,c}(t_{k-1}, x_k, v_k)$$

Proposition (L., Oberman, 2020)

Assume $h_k := \frac{c}{k^\alpha} \leq \frac{1}{\sqrt{L}}$ *and* $t_k = \sum_{i=1}^k h_i$, *then for* $\alpha = \frac{3}{4}$,

$$\mathbb{E}[f(x_k)] - f^* \leqslant \frac{\frac{1}{16c^2}E_0 + c^2\sigma^2(1 + \log(k))}{(k^{1/4} - 1)^2}$$

Thanks!