Adaptive Gradient Methods for Stochastic Optimization

Anton Rodomanov (CISPA)

17 October 2024
Blue Yonder Series on Optimization for Machine Learning
(online)

Outline

Optimization Problems in Machine Learning

Optimization Problems

In Machine Learning (ML), we often deal with optimization problems:

$$f^* = \min_{x \in \mathbb{R}^d} f(x),$$

where $f\colon \mathbb{R}^d \to \mathbb{R}$ is a certain "objective function".

Main Example: Empirical Risk Minmization

- We are given a dataset $\mathcal{D} = \{(a_i, b_i)\}_{i=1}^n$ of n objects: a_i is the "features" of object i and b_i is a "label".
- Fix a certain model Φ which takes features a and has its own internal parameters (= "weights") $x \in \mathbb{R}^d$ and predicts a label $\hat{b} = \Phi(a; x)$.
- Define a "loss function" $L(\hat{b}, b)$ which measures how close is the prediction $\hat{b}$ from the "true label" b.

Goal: Solve the following optimization problem:

$$\min_{x \in \mathbb{R}^d} \Big[f(x) = \frac{1}{n} \sum_{i=1}^n \underbrace{L(\Phi(a_i; x), b_i)}_{=: f(x; a_i, b_i)} \Big].$$

Here $f(x; a, b)$ is the "loss" / "risk" of the model at object (a, b).

Examples

- (Least Squares) Regression problem, $b \in \mathbb{R}$.
  - Model: $\Phi(a; x) = \langle \phi(a), x \rangle$, where ϕ is a predefined feature transform.
  - Loss function: $L(\hat{b}, b) = \frac{1}{2}(\hat{b} - b)^2$.
  - Loss at object:

$$f(x; a, b) = \tfrac{1}{2}(\langle \phi(a), x \rangle - b)^2.$$

- (Logistic regression) Classification into $k \geq 2$ classes, $b \equiv (b^{(1)}, \dots, b^{(k)})$, $b^{(j)}$ is the prob. that object belongs to class j.
  - Model: Parameters $x = (x^{(1)}, \dots, x^{(k)})$, $x^{(j)} \in \mathbb{R}^{d_j}$, $x^{(k)} \equiv 0$, $\Phi(a; x) = \hat{b} \equiv (\hat{b}^{(1)}, \dots, \hat{b}^{(k)})$ with $\hat{b}^{(j)} = \frac{\exp(\langle \phi_j(a), x^{(j)} \rangle)}{\sum_{j'=1}^k \exp(\langle \phi_{j'}(a), x^{(j')} \rangle)}$, where ϕ_j are predefined feature transforms.
  - Loss function: Cross-entropy $L(\hat{b}, b) = -\sum_{j=1}^k b^{(j)} \ln \hat{b}^{(j)}$.
  - Loss at object:

$$f(x; a, b) = \ln\Bigl(\sum_{j=1}^k \exp(\langle \phi_j(a), x^{(j)} \rangle)\Bigr) - \sum_{j=1}^k b^{(j)} \langle \phi_j(a), x^{(j)} \rangle.$$

- (Deep Neural Networks, DNNs) Generalization of previous examples learning feature transforms ϕ_j "on-the-fly".

Gradient Descent (GD)

Algorithm

Problem: $\min_{x \in \mathbb{R}^d} f(x)$.

GD algorithm

Iterate for $k = 0, \ldots, T-1$:

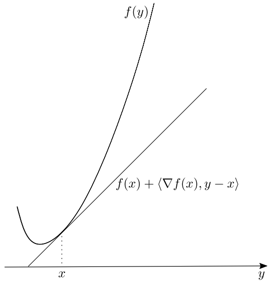
$$x_{k+1} = x_k - h \nabla f(x_k).$$

Here $\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_d}(x)\right) \in \mathbb{R}^d$ is the gradient of f at $x \in \mathbb{R}^d$, and $h > 0$ is the "stepsize" of the method.

How to choose h? How fast does this method converge?

Convex Functions

From now on, we assume the objective function f in our problem is convex:

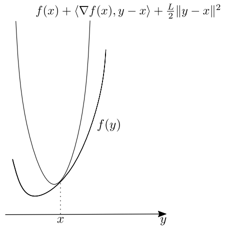


- Many basic ML models such as linear/logistic regression, SVM, etc. are convex.
- More advanced models such as DNNs are nonconvex, and we can only guarantee the convergence to a local minimizer (which is often sufficient in practice).
- Nonconvex function behaves like a convex one around a local minimizer, or when part of the variables are fixed.

Smooth Functions

Function f is called *L-smooth* ($L > 0$) if its gradient is L-Lipschitz:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \qquad \forall x, y \in \mathbb{R}^d.$$


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

Equiv.: $f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|y - x\|^2$.

Equivalent definition: $\|\nabla^2 f(x)\| \leq L$, $\forall x \in \mathbb{R}^d$, where $\nabla^2 f(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x)\right)_{i,j=1}^d$ is the Hessian matrix.

Examples:

- (Quadratic function) $f(x) = \frac{1}{2}\langle Ax, x\rangle + \langle b, x\rangle$, $A \in \mathbb{S}^d_{++}$, $b \in \mathbb{R}^d$ $\implies L = \lambda_{\max}(A)$.
- (Log-sum-exp) $f(x) = \log(\sum_{i=1}^m e^{\langle a_j, x\rangle}) \implies L = \|A\|^2$, where $A = [a_1, \ldots, a_m]$.

Convergence Rate

Assumption: f is L-smooth.

Theorem (Section 2.1.5 in Nesterov 2018)

Consider GD with stepsize $h = \frac{1}{L}$*. Then, for any* k, $f(x_{k+1}) \leq f(x_k)$ *and*

$$f(x_T) - f^* \leq \frac{LR^2}{T},$$

where $R = \|x_0 - x^*\|$, x^* *is a minimizer of* f*.*

Main Drawback: Expensive Computations

Recall that, in ML problems, we typically solve

$$\min_{x \in \mathbb{R}^d} \left[f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right],$$

where n is the number of objects and $f_i(x)$ is the loss at object i. In this case, computing the exact gradient

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

is very expensive when n is big.

Natural idea: Approximate $\nabla f(x)$ by computing the average over only a few (randomly selected) objects.

Stochastic Gradient Method (SGD)

Stochastic Gradient Oracle (SGO)

SGO: Procedure taking $x \in \mathbb{R}^d$ and returning a random vector $g(x, \xi) \in \mathbb{R}^d$, where ξ is a random variable, g is a deterministic function, such that $g(x, \xi)$ is an unbiased estimate of $\nabla f(x)$:

$$\mathbb{E}_\xi[g(x, \xi)] = \nabla f(x).$$

Main example: If $f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x)$, then

$$g(x, \xi) = \nabla f_\xi(x), \qquad \xi \cong \mathrm{Unif}(1, \dots, n).$$

More generally, if $f(x) = \mathbb{E}_\xi[F(x, \xi)]$, then $g(x, \xi) = \nabla_x F(x, \xi)$.

Variance of Stochastic Gradient

Variance: $\sigma_g^2(x) \coloneqq \mathbb{E}_\xi[\|g(x,\xi) - \nabla f(x)\|^2]$.

Mini-batching: Mini-batched version of g is an SGO g_b defined by

$$g_b(x, \xi_{[b]}) = \frac{1}{b} \sum_{j=1}^{b} g(x, \xi_b),$$

where $\xi_{[b]} = (\xi_1, \ldots, \xi_b)$ consists of b independent copies of ξ.

Key property: $\sigma_{g_b}^2(x) = \frac{1}{b}\sigma_g^2(x)$.

Mini-batching is especially useful when g_b can be computed in parallel.

SGD Algorithm

Problem: $f^* = \min_{x \in \mathbb{R}^d} f(x)$, where f is given by an SGO g.

SGD

Iterate for $k = 0, \dots, T-1$:

$$g_k = g(x_k, \xi_k),$$
$$x_{k+1} = x_k - h g_k.$$

Here $\xi_0, \dots, \xi_{T-1}$ are independent copies of ξ, and $h > 0$ is the stepsize of the method.

Convergence on Smooth Functions

Assumptions: f is L-smooth and $\sigma_g^2(x) \leq \sigma^2 \ \forall x \in \mathbb{R}^d$.

Output point: Either $\bar{x}_T = \frac{1}{T}\sum_{k=0}^{T-1} x_k$, or $\bar{x}_T \cong \mathrm{Unif}(x_0, \ldots, x_{T-1})$.

Theorem (Section 4.1.2 in Lan 2020)

Consider SGD with stepsize $h = \frac{1}{L + \frac{\sigma}{R}\sqrt{T}}$, where $R = \|x_0 - x^*\|$. Then,

$$\mathbb{E}[f(\bar{x}_T)] - f^* \leq \frac{LR^2}{T} + \frac{\sigma R}{\sqrt{T}}.$$

Note:

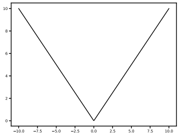
- First term is the rate of GD.
- Second term is due to stochastic noise and dominates when T is large enough.
- To accelerate convergence, we need to decrease σ (e.g., by mini-batching).

Nonsmooth Functions: Motivation

Many functions important in applications may be nonsmooth.
For example:

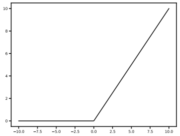
- Robust regression ($a_i \in \mathbb{R}^d$, $b \in \mathbb{R}$):

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^{n} |\langle a_i, x \rangle - b_i| \right\}.$$



- SVM for binary classification ($a_i \in \mathbb{R}^d$, $b_i \in \{-1, 1\}$):

$$\min_{\|x\| \leq R} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^{n} [1 - b_i \langle a_i, x \rangle]_+ \right\},$$



  where $R > 0$ and $[t]_+ := \max\{t, 0\}$ (also known as ReLU activation function for neural networks).

These functions are not smooth but still rather regular. They are examples of Lipschitz functions.

Lipschitz Functions

Function f is called M-Lipschitz if

$$|f(x) - f(y)| \leq M\|x - y\|, \qquad \forall x, y \in \mathbb{R}^d.$$

Equivalent condition[1]**:** $\|\nabla f(x)\| \leq M$, $\forall x \in \mathbb{R}^d$.

[1] $\nabla f(x)$ is an arbitrary subgradient of f at x if f is not differentible at this point.

Convergence Rate for Nonsmooth Functions

Assumptions: f is M-Lipschitz and $\sigma_g^2(x) \leq \sigma^2 \; \forall x \in \mathbb{R}^d$.

Theorem (Section 4.1.1 in Lan 2020)

Consider SGD with stepsize $h = \frac{R}{(M+\sigma)\sqrt{T}}$, *where* $R = \|x_0 - x^*\|$. *Then,*

$$\mathbb{E}[f(\bar{x}_T)] - f^* \leq \frac{MR}{\sqrt{T}} + \frac{\sigma R}{\sqrt{T}}.$$

Reminder: For L-smooth functions, we needed to choose $h = \frac{1}{L + \frac{\sigma}{R}\sqrt{T}}$ and the rate was $O(\frac{LR^2}{T} + \frac{\sigma R}{\sqrt{T}})$.

Intermediate Smoothness: Hölder Class

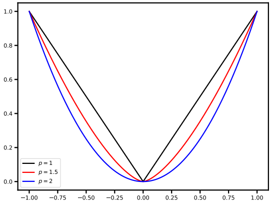
Lipschitz-smooth and nonsmooth Lipschitz functions are particular subclasses of the more general class of Hölder-smooth functions.

Function f is called (ν, H_ν)-Hölder smooth ($\nu \in [0,1]$ and $H_\nu > 0$) if

$$\|\nabla f(x) - \nabla f(y)\| \le H_\nu \|x - y\|^\nu, \qquad \forall x, y \in \mathbb{R}^d.$$

- Lipschitz-smooth functions ($\nu = 1$): $\|\nabla f(x) - \nabla f(y)\| \le H_1 \|x - y\|$.
- Lipschitz functions ($\nu = 0$): $\|\nabla f(x) - \nabla f(y)\| \le H_0$ ($H_0 = 2M$)

Example: $f(x) = \sum_{i=1}^{n} |\langle a_i, x\rangle - b_i|^p$ ($p \in [1,2]$) $\implies$ $\nu = p - 1$.



Convergence of SGD on Hölder-Smooth Functions

Assumptions: f is (ν, H_ν)-Hölder smooth and $\sigma_g^2(x) \leq \sigma^2 \ \forall x \in \mathbb{R}^d$.

Theorem

Consider SGD with stepsize $h \sim \frac{1}{\frac{H_\nu}{R^{1-\nu}} T^{\frac{1-\nu}{2}} + \frac{\sigma}{R}\sqrt{T}}$, *where* $R = \|x_0 - x^*\|$.

Then,

$$\mathbb{E}[f(\bar{x}_T)] - f^* \lesssim \frac{H_\nu R^{1+\nu}}{T^{\frac{1+\nu}{2}}} + \frac{\sigma R}{\sqrt{T}}.$$

Overparameterized Models

- Modern ML models (especially DNNs) are often overparameterized[2]: their number of parameters exceed the amount of training data, and the model can achieve (nearly) zero training loss.
- For such models, SGD works especially well, and convergence becomes comparable to GD while the cost of iteration is still significantly smaller.

[2] Cotter et al. 2011; Schmidt and Roux 2013; Needell et al. 2014; Ma et al. 2018; Liu and Belkin 2018; Necoara et al. 2019

Variance at Minimizer

To quantify how well the model fits the training data we can use the variance at the minimizer:

$$\sigma_*^2 := \sigma_g^2(x^*) \equiv \mathbb{E}_\xi[\|g(x,\xi)\|^2].$$

Example: Let $f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$ and consider the "standard SGO"[3] $g(x,\xi) = \nabla f_\xi(x)$. Then,

$$\sigma_*^2 = \frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x^*)\|^2.$$

If there exists a solution x^* such that it minimizes each loss function f_i, we have $\nabla f_i(x^*) = 0$ and $\sigma_* = 0$.

Main property: If each f_i is $L_{\max}$-smooth, then

$$\sigma_g^2(x) \leq 4L_{\max}[f(x) - f^*] + 2\sigma_*^2.$$

The first term in the above expression goes to zero as $f(x) \to f^*$.

[3] We could also use mini-batching but we do not do it for simplicity.

Convergence on Smooth Overparameterized Models

Assumptions: $f(x) \equiv \frac{1}{n}\sum_{i=1}^n f_i(x)$ with $L_{\max}$-smooth components f_i, standard SGO g.

Theorem

Consider SGD with stepsize $h \sim \frac{1}{L_{\max} + \frac{\sigma_*}{R}\sqrt{T}}$, *where* $R = \|x_0 - x^*\|$. *Then,*

$$\mathbb{E}[f(\bar{x}_T)] - f^* \lesssim \frac{L_{\max} R^2}{T} + \frac{\sigma_* R}{\sqrt{T}}.$$

Discussion:

- Previously, we had a similar result but with σ instead of σ^* and L instead of $L_{\max}$.
- If $\sigma_* = 0$, we can use a nearly constant step size $h \sim \frac{1}{L_{\max}}$ and get the $\frac{L_{\max}R^2}{T}$ convergence, which is similar to GD but each iteration is much cheaper.

Convergence on Hölder-Smooth Overparameterized Models

Assumptions: $f(x) \equiv \frac{1}{n}\sum_{i=1}^{n} f_i(x)$ with $(\nu, H_{\max}(\nu))$-Hölder smooth components, standard SGO g.

Theorem

Consider SGD with stepsize $h \sim \frac{1}{\frac{H_{\max}(\nu)}{R^{1-\nu}} T^{\frac{1-\nu}{2}} + \frac{\sigma_*}{R}\sqrt{T}}$, *where* $R = \|x_0 - x^*\|$.

Then,

$$\mathbb{E}[f(\bar{x}_T)] - f^* \lesssim \frac{H_{\max}(\nu) R^{1+\nu}}{T^{\frac{1+\nu}{2}}} + \frac{\sigma_* R}{\sqrt{T}}.$$

Summary

Case	Stepsize	Rate
M-Lipschitz, σ-variance	$\frac{M+\sigma}{R\sqrt{T}}$	$\frac{MR}{T} + \frac{\sigma R}{\sqrt{T}}$
L-smooth, σ-variance	$\frac{1}{L+\frac{\sigma}{R}\sqrt{T}}$	$\frac{LR^2}{T} + \frac{\sigma R}{\sqrt{T}}$
(ν, H_ν)-Hölder, σ-variance	$\frac{1}{\frac{H_\nu}{R^{1-\nu}} T^{\frac{1-\nu}{2}} + \frac{\sigma}{R}\sqrt{T}}$	$\frac{H_\nu R^{1+\nu}}{T^{\frac{1+\nu}{2}}} + \frac{\sigma R}{\sqrt{T}}$
$(\nu, H_{\max}(\nu))$-Hölder components	$\frac{1}{\frac{H_{\max}(\nu)}{R^{1-\nu}} T^{\frac{1-\nu}{2}} + \frac{\sigma_*}{R}\sqrt{T}}$	$\frac{H_{\max}(\nu) R^{1+\nu}}{T^{\frac{1+\nu}{2}}} + \frac{\sigma_* R}{\sqrt{T}}$

We will see next that adaptive methods such as AdaGrad can achieve all of this automatically (almost without tuning stepsize).

Adaptive Methods: AdaGrad

Classical AdaGrad

AdaGrad algorithm [Duchi et al. 2011]

Set $S_{-1} = 0$ and iterate for $k = 0, \dots, T-1$:

$$\begin{aligned} g_k &= g(x_k, \xi_k), \\ S_k^2 &= S_{k-1}^2 + g_k^2, \\ x_{k+1} &= x_k - \gamma \frac{g_k}{S_k}. \end{aligned}$$

Here $\gamma > 0$ is a parameter. All operations on vectors are component-wise.

NB: $S_k^2 = \sum_{t=0}^{k} g_t^2$ is the summation of squared gradients.

Adam [Kingma and Ba 2015]

Heuristical improvement over AdaGrad that often works well in practice.

Set $m_{-1} = 0$, $S_{-1} = 0$ and iterate for $k = 0, \dots, T-1$:

$$\begin{aligned} g_k &= g(x_k, \xi_k), \\ m_k &= \beta_1 m_{k-1} + (1-\beta_1) g_k, \qquad & \hat{m}_k &= \frac{m_k}{1-\beta_1^{k+1}}, \\ S_k^2 &= \beta_2 S_{k-1}^2 + (1-\beta_2) g_k^2, \qquad & \hat{S}_k^2 &= \frac{S_k^2}{1-\beta_2^{k+1}}, \\ x_{k+1} &= x_k - \alpha \frac{\hat{m}_k}{\hat{S}_k}. \end{aligned}$$

NB: $m_k = (1-\beta_1) \sum_{t=0}^k \beta_1^{k-t} g_t$ and $S_k^2 = (1-\beta_2) \sum_{t=0}^k \beta_2^{k-t} g_t^2$.

c.f.: heavy-ball method $x_{k+1} = x_k - \alpha g_k + \beta(x_k - x_{k-1})$ which can be written as $x_{k+1} = x_k - \alpha \sum_{t=0}^k \beta^{k-t} g_t$.

AdaGrad: Scalar Version

In what follows, we concentrate on the scalar AdaGrad method.

Scalar AdaGrad algorithm (also known as AdaGrad-Norm)

Set $S_{-1} = 0$ and iterate for $k = 0, \ldots, T-1$:

$$\begin{aligned} g_k &= g(x_k, \xi_k), \\ S_k^2 &= S_{k-1}^2 + \|g_k\|^2, \\ x_{k+1} &= x_k - \frac{\gamma}{S_k} g_k, \end{aligned}$$

- This is a simplification but sufficient to illustrate main points.
- Diagonal version is a natural “per-coordinate” extension of this idea. It approximates the gradient method $x_{k+1} = x_k - B^{-1} g_k$ with the fixed diagonal matrix B. This can be good in situations such as $[\nabla^2 f(x)]_{ii} \leq L_j$ with different L_j. E.g., if $f(x) = \frac{1}{2} \sum_{j=1}^{d} (a_j x^{(j)} - b_j)^2$, then a good scaling is $B_{jj} = L_j = a_j$.

AdaGrad with Projection

We introduce one more “minor modification” and consider from now on the following “safeguarded” version of AdaGrad:

$$\boxed{x_{k+1} = \pi_{B(x_0,R)}\Big(x_k - \frac{\gamma}{S_k} g_k\Big), \quad S_k^2 = \sum_{t=0}^{k} \|g_t\|^2,}$$

where $\pi_{B_R}(\cdot)$ is the projection onto the ball $B(x_0, R)$:

$$\pi_{B_R}(x) = \begin{cases} x, & \text{if } \|x - x_0\| \leq R, \\ x_0 + R\frac{x-x_0}{\|x-x_0\|}, & \text{otherwise,} \end{cases}$$

where $R \sim \|x_0 - x^*\|$.

Output point: Either $\bar{x}_T = \frac{1}{T}\sum_{k=0}^{T-1} x_k$, or $\bar{x}_T \cong \mathrm{Unif}(x_0, \dots, x_{T-1})$ (same as for SGD).

Convergence on Smooth and Nonsmooth Functions

Assumption: $\sigma_g^2(x) \leq \sigma^2 \ \forall x \in \mathbb{R}^d$.

Theorem (Levy et al. 2018)

Consider AdaGrad with $\gamma = R$, *where* $R \sim \|x_0 - x^*\|$.
If f *is* M*-Lipschitz, then*

$$\mathbb{E}[f(\bar{x}_T)] - f^* \lesssim \frac{MR}{\sqrt{T}} + \frac{\sigma R}{\sqrt{T}}.$$

If f *is* L*-smooth, then*

$$\mathbb{E}[f(\bar{x}_T)] - f^* \lesssim \frac{LR^2}{T} + \frac{\sigma R}{\sqrt{T}}.$$

NB: With the same parameter $\gamma = R$, AdaGrad works both for smooth and nonsmooth functions! And we don't even need to know M, L or σ (as in SGD).

Convergence on Hölder-Smooth Problems

Assumptions: f is (ν, H_ν)-Hölder smooth and $\sigma_g^2(x) \leq \sigma^2 \ \forall x \in \mathbb{R}^d$.

Theorem (Rodomanov et al. 2024)

Consider AdaGrad with $\gamma = R$, where $R \sim \|x_0 - x^*\|$. Then,

$$\mathbb{E}[f(\bar{x}_T)] - f^* \lesssim \frac{H_\nu R^{1+\nu}}{T^{\frac{1+\nu}{2}}} + \frac{\sigma R}{\sqrt{T}}.$$

NB: This is exactly the same convergence rate as we had for SGD with the carefully chosen stepsize (depending on ν, H_ν, R and σ).

Overparameterized Hölder-Smooth Problems

Assumptions: $f(x) \equiv \frac{1}{n} \sum_{i=1}^n f_i(x)$, where each f_i is $(\nu, H_{\max}(\nu))$-Hölder smooth, standard SGO g.

Theorem (Rodomanov et al. 2024)

Consider AdaGrad with $\gamma = R$, where $R \sim \|x_0 - x^*\|$. Then,

$$\mathbb{E}[f(\bar{x}_T)] - f^* \lesssim \frac{H_{\max}(\nu) R^{1+\nu}}{T^{\frac{1+\nu}{2}}} + \frac{\sigma_* R}{\sqrt{T}},$$

where $\sigma_* := \sigma_g(x^*)$.

NB: This is again the same convergence rate as for SGD, without any knowledge of ν, $H_{\max}(\nu)$ or σ_*.

Summary: Comparison with SGD

Case	Stepsize for SGD	γ in AdaGrad	Rate
M-Lipschitz, σ-variance	$\frac{M+\sigma}{R\sqrt{T}}$	R	$\frac{MR}{T} + \frac{\sigma R}{\sqrt{T}}$
L-smooth, σ-variance	$\frac{1}{L+\frac{\sigma}{R}\sqrt{T}}$	R	$\frac{LR^2}{T} + \frac{\sigma R}{\sqrt{T}}$
(ν, H_ν)-Hölder, σ-variance	$\frac{1}{\frac{H_\nu}{R^{1-\nu}} T^{\frac{1-\nu}{2}} + \frac{\sigma}{R}\sqrt{T}}$	R	$\frac{H_\nu R^{1+\nu}}{T^{\frac{1+\nu}{2}}} + \frac{\sigma R}{\sqrt{T}}$
$(\nu, H_{\max}(\nu))$-Hölder components	$\frac{1}{\frac{H_{\max}(\nu)}{R^{1-\nu}} T^{\frac{1-\nu}{2}} + \frac{\sigma_*}{R}\sqrt{T}}$	R	$\frac{H_{\max}(\nu) R^{1+\nu}}{T^{\frac{1+\nu}{2}}} + \frac{\sigma_* R}{\sqrt{T}}$

Conclusions

Conclusions

- Stepsize and convergence rate of SGD depend on many characteristic of the specific problem: smoothness, variance, degree of overparameterization, . . .
- Adaptive methods such as AdaGrad reduce the knowledge of parameters to one and are “universal”—they automatically adapt to the best possible setting for a specific problem.
- Our theory provides a possible explanation why adaptive methods often perform well in practice.

Thank you!

References I

A. Cotter, O. Shamir, N. Srebro, and K. Sridharan. Better mini-batch algorithms via accelerated gradient methods. **Advances in Neural Information Processing Systems**, 24, 2011.

J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. **Journal of Machine Learning Research**, 12(7), 2011.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. **International Conference on Learning Representations (ICLR)**, 2015.

G. Lan. **First-order and Stochastic Optimization Methods for Machine Learning**. Volume 1. Springer, 2020.

K. Y. Levy, A. Yurtsever, and V. Cevher. Online Adaptive Methods, Universality and Acceleration. **Advances in Neural Information Processing Systems**, 31, 2018.

References II

C. Liu and M. Belkin. Accelerating SGD with momentum for over-parameterized learning. **arXiv preprint arXiv:1810.13395**, 2018.

S. Ma, R. Bassily, and M. Belkin. The Power of Interpolation: Understanding the Effectiveness of SGD in Modern Over-parametrized Learning. In **International Conference on Machine Learning**, pages 3325–3334, 2018.

I. Necoara, Y. Nesterov, and F. Glineur. Linear convergence of first order methods for non-strongly convex optimization. **Mathematical Programming**, 175:69–107, 2019.

D. Needell, R. Ward, and N. Srebro. Stochastic Gradient Descent, Weighted Sampling, and the Randomized Kaczmarz Algorithm. **Advances in Neural Information Processing Systems**, 27, 2014.

Y. Nesterov. **Lectures on Convex Optimization**. Volume 137. Springer, 2nd edition, 2018.

References III

A. Rodomanov, X. Jiang, and S. Stich. Universality of AdaGrad Stepsizes for Stochastic Optimization: Inexact Oracle, Acceleration and Variance Reduction. **arXiv preprint arXiv:2406.06398**, 2024.

M. Schmidt and N. L. Roux. Fast Convergence of Stochastic Gradient Descent under a Strong Growth Condition. **arXiv preprint arXiv:1308.6370**, 2013.