BOSTON UNIVERSITY

Adaptive Gradient Descent methods for Constrained Optimization

ALINA ENE

Joint work with:

Huy L. Nguyen (Northeastern University)

Adrian Vladu (CNRF & IRIF, Universite de Paris)

Problem Definition

$f\colon \mathbb{R}^n \to \mathbb{R}$ differentiable loss function

$X \subseteq \mathbb{R}^n$ constraint set that is convex and "simple"

$$\boxed{\min_{x \in X} f(x)}$$

Computational model: function access via first-order oracle

$$\xrightarrow{\;x \in X\;} \boxed{\text{Blackbox}} \xrightarrow{\;f(x),\, \nabla f(x)\;}$$

Goal: minimize number of queries $x_1, x_2, \ldots, x_T$ to obtain

$$f(x_{\text{out}}) - f(x^*) \leq \epsilon$$

Blackbox Model

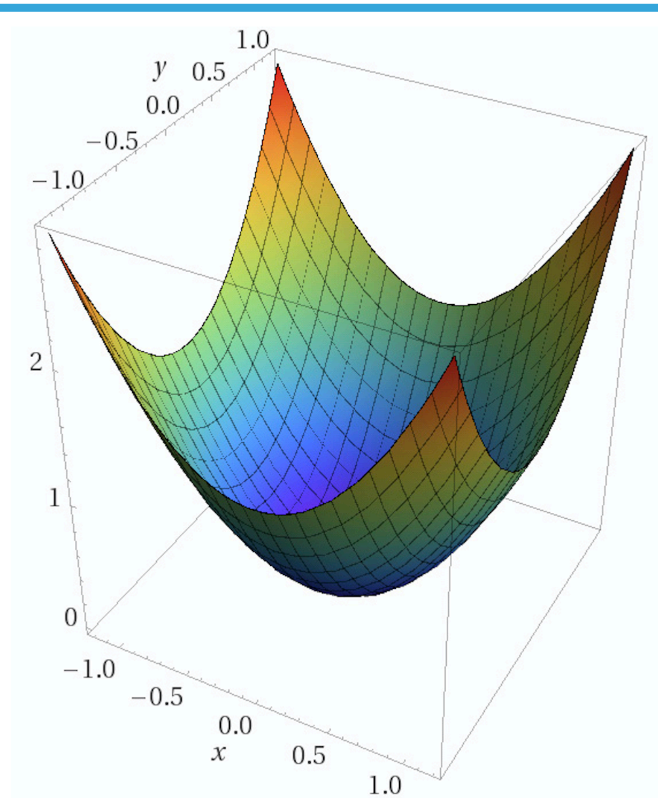
$$\min_{x \in X} f(x)$$

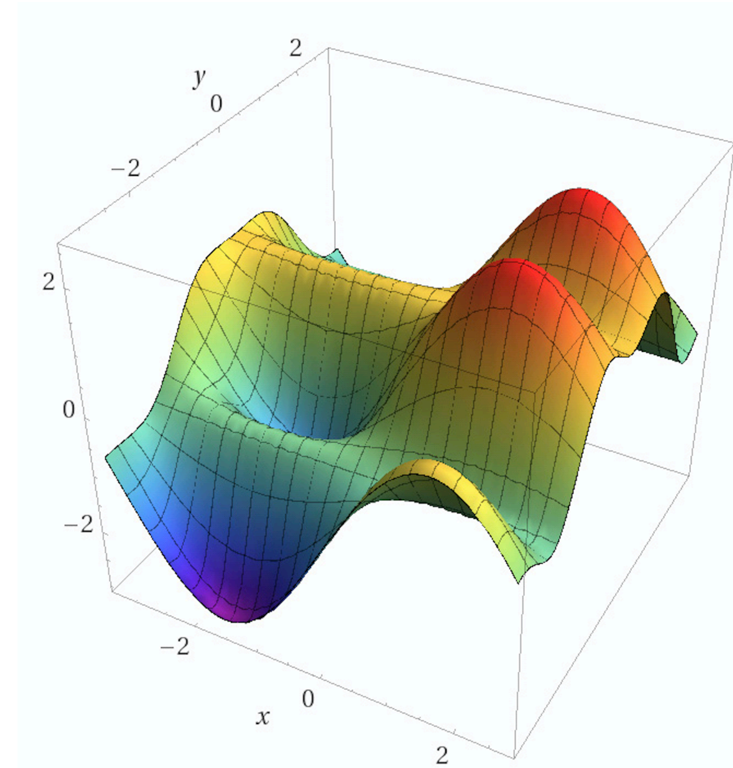
Computational model: function access via first-order oracle

$$x \in X \longrightarrow \boxed{\text{Blackbox}} \longrightarrow f(x), \nabla f(x)$$

Goal: minimize number of queries $x_1, x_2, \ldots, x_T$ to obtain

$f(x_{\text{out}}) - f(x^*) \leq \epsilon$ convex	$\lVert \nabla f(x_{\text{out}}) \rVert \leq \epsilon$ non-convex $(X = \mathbb{R}^n)$





Blackbox Model

$$\min_{x \in X} f(x)$$

Computational model: function access via first-order oracle

$$x \in X \longrightarrow \boxed{\text{Blackbox}} \longrightarrow f(x), \nabla f(x)$$

Goal: minimize number of queries $x_1, x_2, \ldots, x_T$ to obtain

$$\boxed{\begin{array}{c} f(x_{\text{out}}) - f(x^*) \le \epsilon \\ \text{convex} \end{array}} \qquad \boxed{\begin{array}{c} \|\nabla f(x_{\text{out}})\| \le \epsilon \\ \text{non-convex } (X = \mathbb{R}^n) \end{array}}$$

Theory: tight upper and lower bounds on complexity

Practice: (stochastic) gradients are readily available

```python
import torch
x = torch.randn(3, requires_grad=True)
# ... # setup model
out.backward() # backpropagation
gradient = x.grad
```

PYTORCH

TensorFlow

Blackbox Model

$$\min_{x \in X} f(x)$$

Computational model: function access via first-order oracle

$$x \in X \longrightarrow \boxed{\text{Blackbox}} \xrightarrow{f(x),\ \nabla f(x)}$$

Goal: minimize number of queries $x_1, x_2, \ldots, x_T$ to obtain

$$f(x_{\text{out}}) - f(x^*) \leq \epsilon$$

convex

$$\|\nabla f(x_{\text{out}})\| \leq \epsilon$$

non-convex $(X = \mathbb{R}^n)$

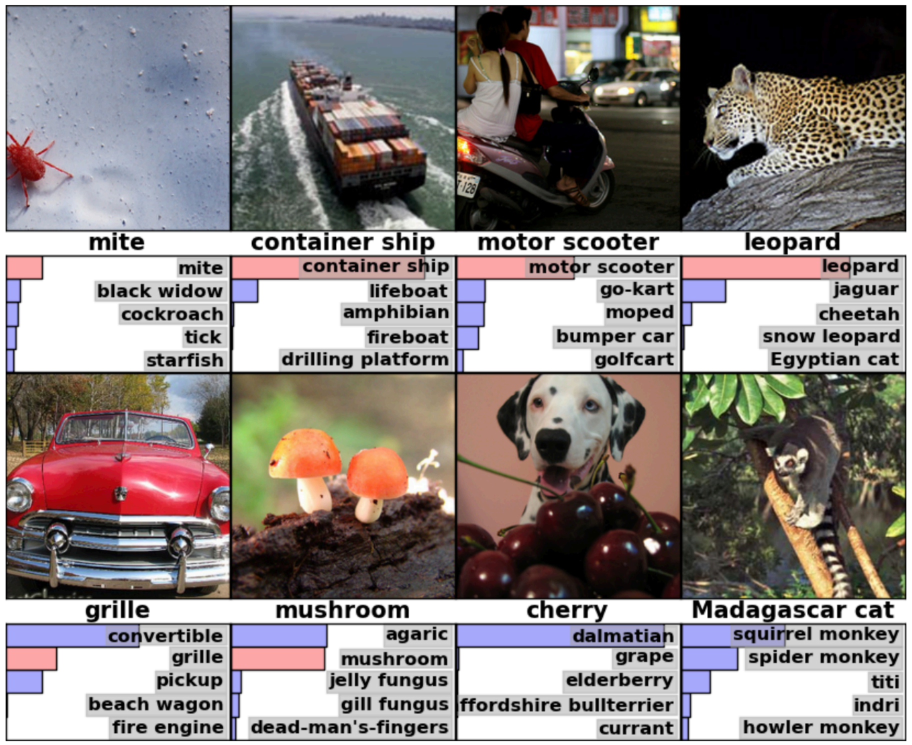
This Talk: Convergence guarantees for convex functions

(We will show experimental results for non-convex problems)

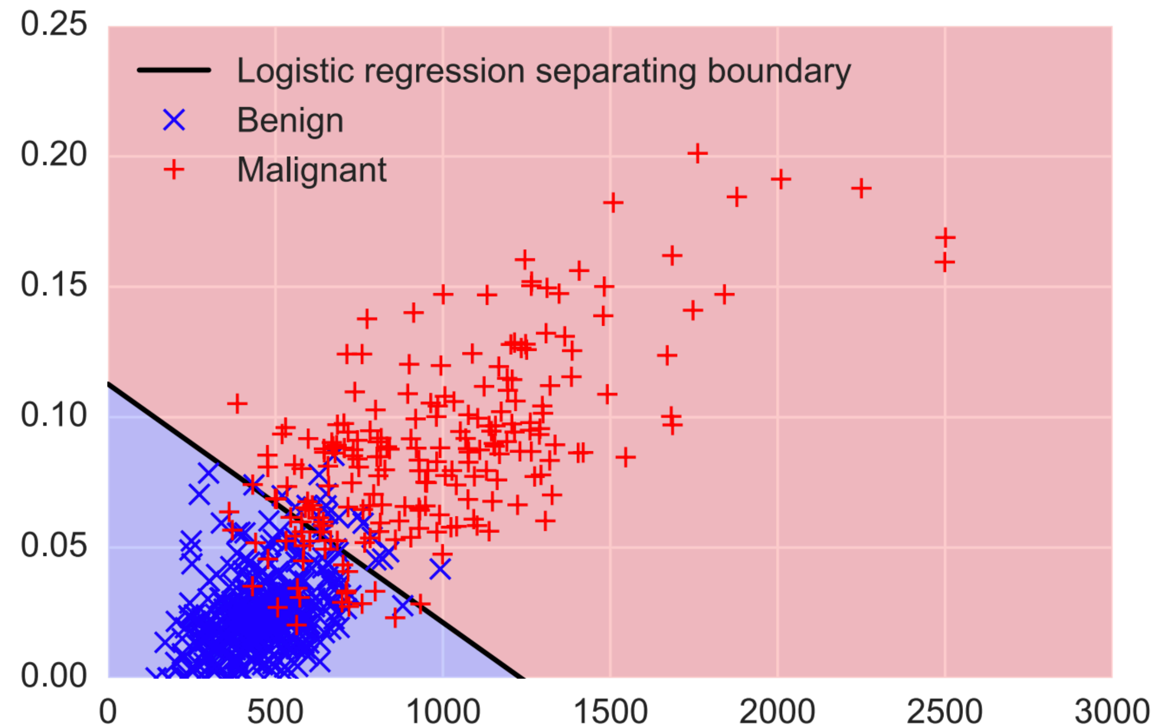
Machine Learning Examples

$$\min_{x \in X} f(x)$$

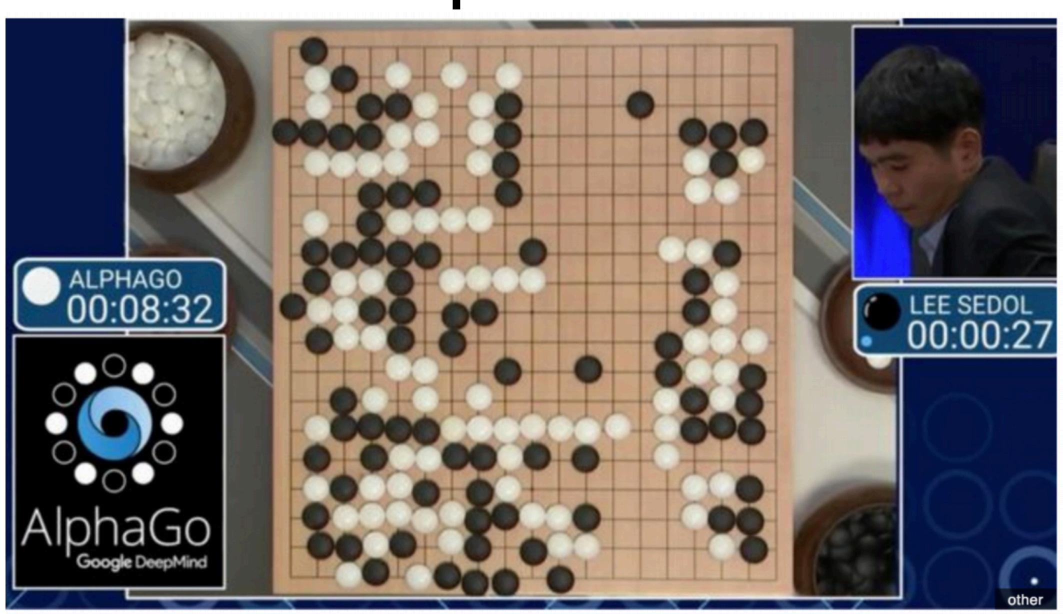
ImageNet Classification



Cancer Classification



AlphaGo



Power Demand Regression

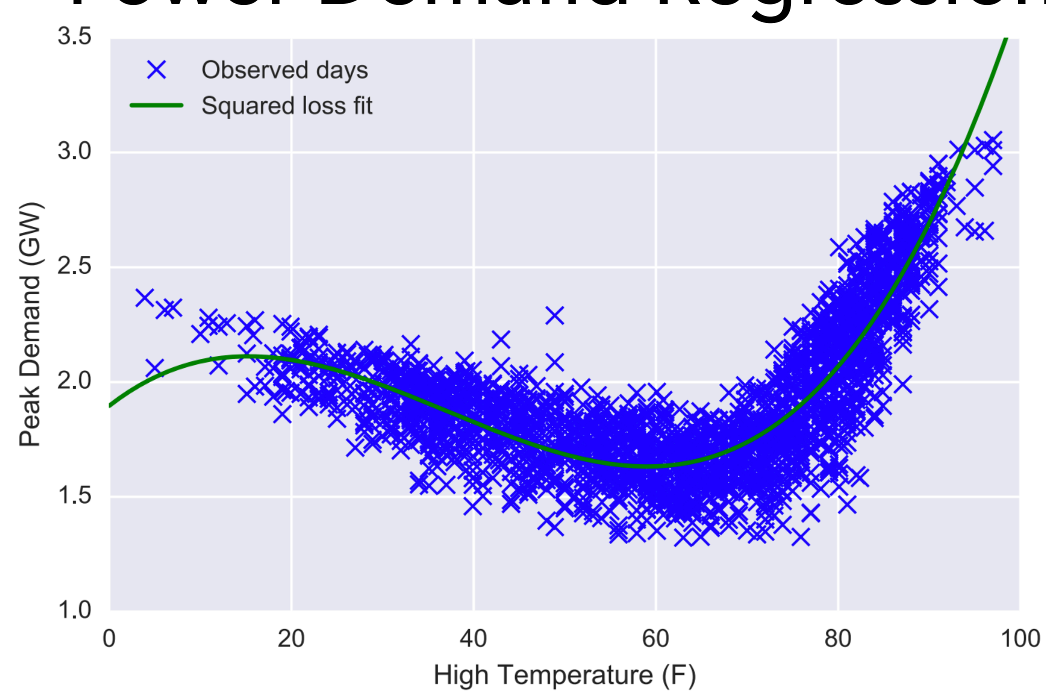


Image credits: Zico Kolter

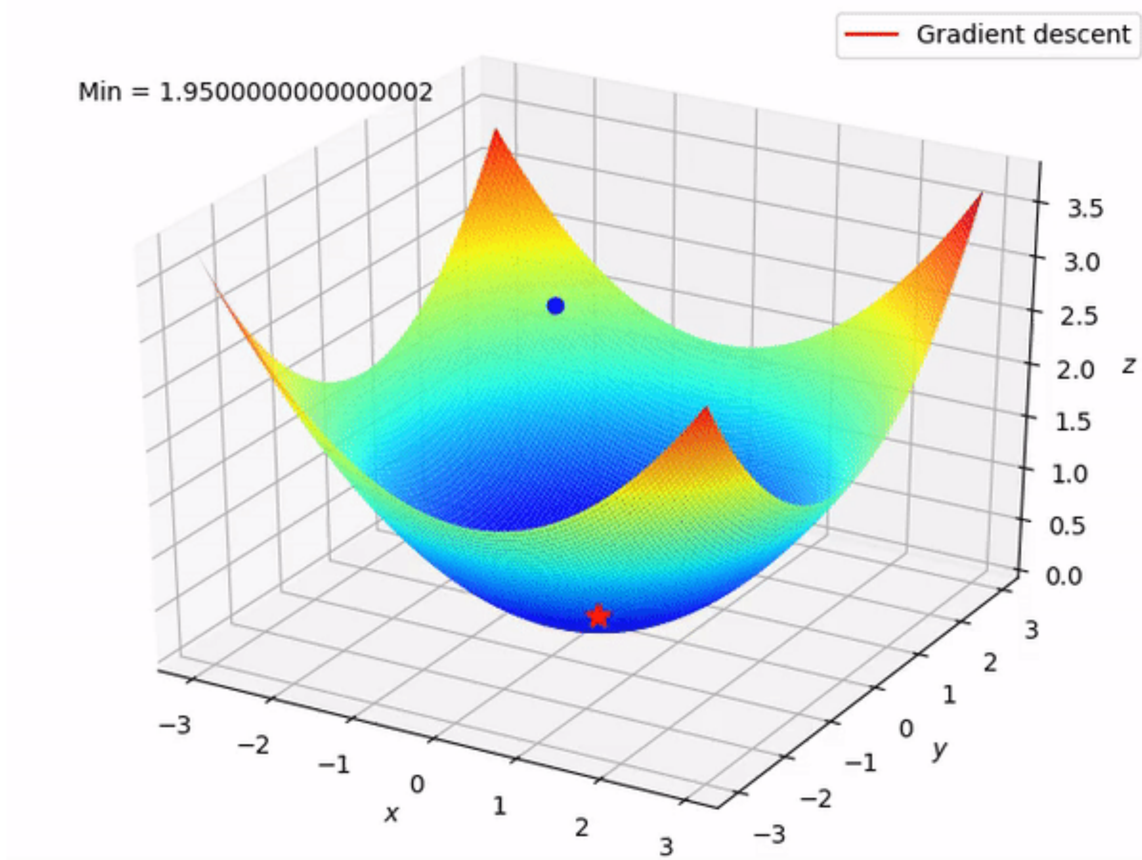
How to Optimize

$$\min_{x \in \mathbb{R}^n} f(x)$$

Gradient Descent

$$x_{t+1} = x_t - \eta_t \nabla f(x_t)$$

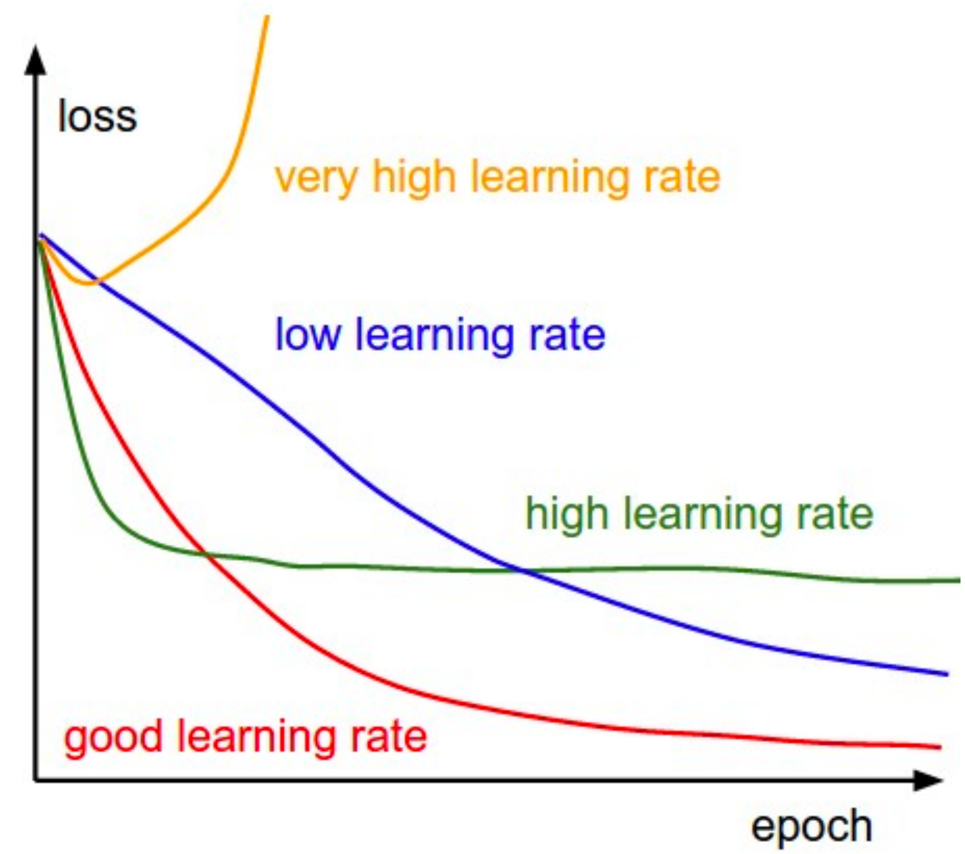
η_t : step size / learning rate



How to set the step size?

Theory answer: it depends ...

Practice answer: manually tune



Gradient descent visualization credit: Sunil Jangir

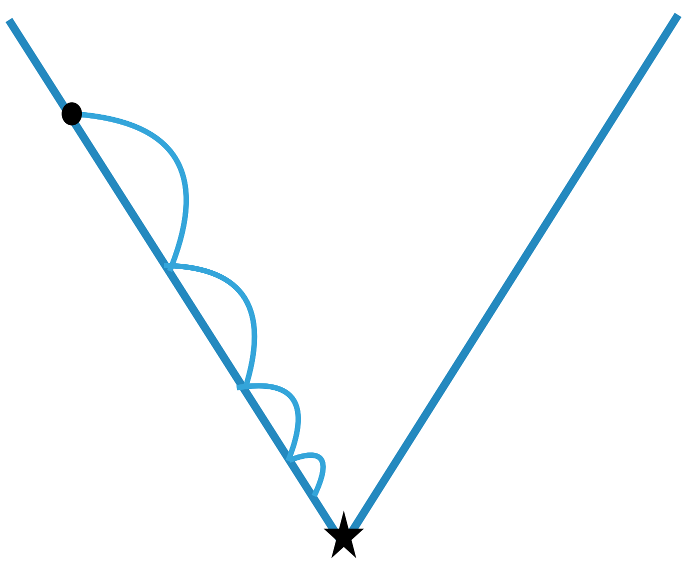
Step size cartoon credit: Stanford CS 231N

How to Set the Gradient Descent Step Size?

Theory answer: it depends on the problem structure

non-smooth

$$\|\nabla f(x)\| \leq G$$

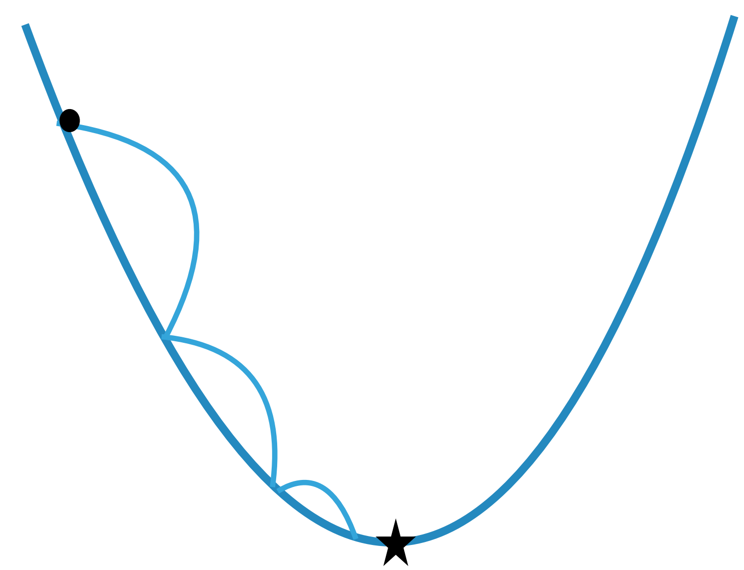


$$\eta_t = \frac{\|x_0 - x^*\|}{G\sqrt{t}}$$ decaying

$$T = \frac{G^2\|x_0 - x^*\|^2}{\epsilon^2}$$ optimal

smooth

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta\|x - y\|$$



$$\eta_t = 1/\beta$$ constant

$$T = \frac{\beta\|x_0 - x^*\|^2}{\epsilon}$$

AGD: $$T = \frac{\beta\|x_0 - x^*\|^2}{\sqrt{\epsilon}}$$ optimal

How to Set the Gradient Descent Step Size?

Theory answer: it depends on the problem structure

Caveats:

- Step sizes depend on several parameters (smoothness, gradient norm, distance to x^*, ...)
- Parameters are often unknown and hard to tune

The dream:

- Automatically learn the step size
- Adapt to (local or global) smoothness and convexity
- Universal algorithms that achieve optimal convergence in the smooth and non-smooth settings simultaneously

Adaptive Gradient Descent

$$\min_{x \in \mathbb{R}^n} f(x)$$

[Duchi, Hazan, Singer; McMahan and Streeter 2010]

Scalar Adagrad

$$x_{t+1} = x_t - \eta_t \nabla f(x_t)$$

$$\eta_t = \frac{1}{\sqrt{\sum_{s=1}^{t} \|\nabla f(x_s)\|^2}}$$

Adagrad

$$x_{t+1} = x_t - \eta_t \nabla f(x_t)$$

$$\eta_{t,i} = \frac{1}{\sqrt{\sum_{s=1}^{t} (\nabla_i f(x_s))^2}}$$

per-coordinate learning rates

Original motivation/use case:

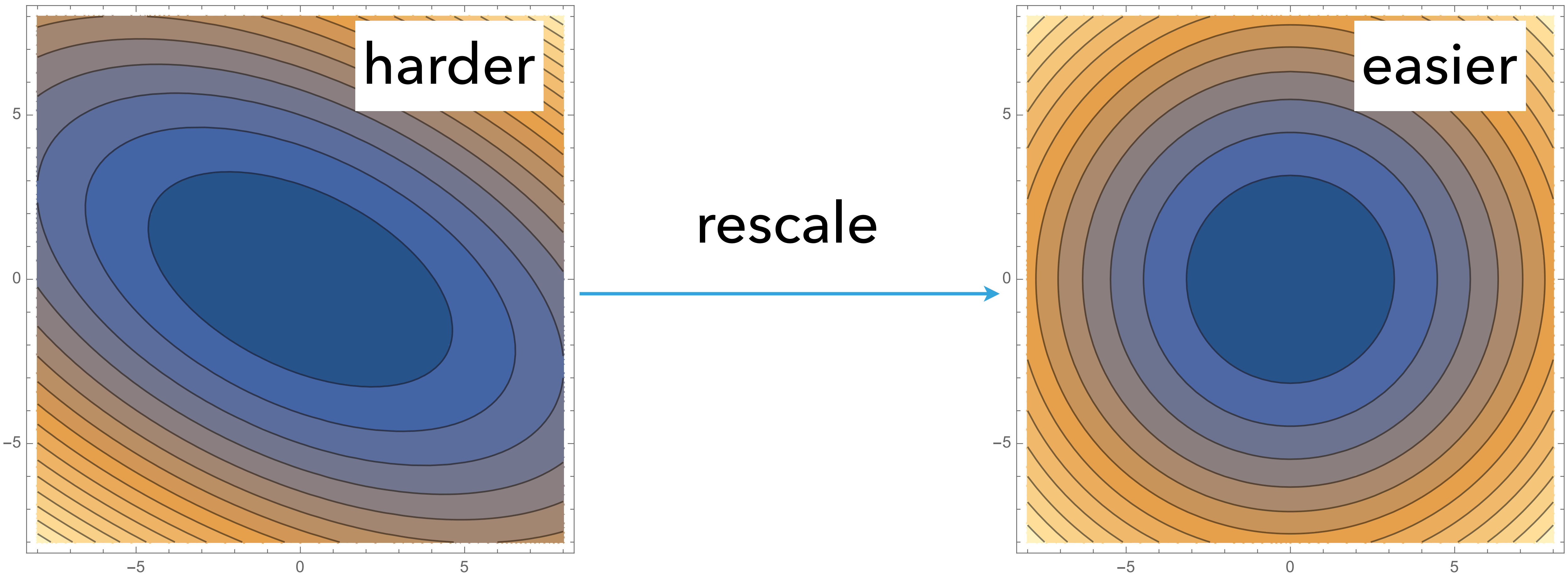
- Sparse and heavy-tailed data (e.g., text data)
- Infrequent features are informative and we want to use different learning rates for them

Preconditioning

Preconditioned Gradient Descent

$$x_{t+1} = x_t - \boxed{\mathbf{H}_t^{-1}} \nabla f(x_t)$$

Hessian $\mathbf{H}_t = \nabla^2 f(x_t)$



$$f(x) = x^\top \mathbf{A} x$$

Adaptive Preconditioning

Preconditioned Gradient Descent

$$x_{t+1} = x_t - \boxed{\mathbf{H}_t^{-1}} \nabla f(x_t)$$

Hessian $\mathbf{H}_t = \nabla^2 f(x_t)$

Adagrad

$$x_{t+1} = x_t - \boxed{\mathbf{G}_t^{-1}} \nabla f(x_t)$$

Matrix computed from gradients

Full-matrix Adagrad: $\mathbf{G}_t = \sqrt{\sum_{s=1}^{t} \nabla f(x_s) \nabla f(x_s)^\top}$ expensive

Diagonal Adagrad: use only the entries on the diagonal

$$\mathbf{D}_{t,i} = \sqrt{\sum_{s=1}^{t} \left(\nabla_i f(x_s) \right)^2} = \eta_{t,i}^{-1}$$

Adaptive Preconditioning

Second-order-like method but with only first-order information

Adagrad

$$x_{t+1} = x_t - \boxed{\mathbf{G}_t^{-1}} \nabla f(x_t)$$

Matrix computed from gradients

Full-matrix Adagrad: $\mathbf{G}_t = \sqrt{\sum_{s=1}^{t} \nabla f(x_s) \nabla f(x_s)^\top}$ expensive

Diagonal Adagrad: use only the entries on the diagonal

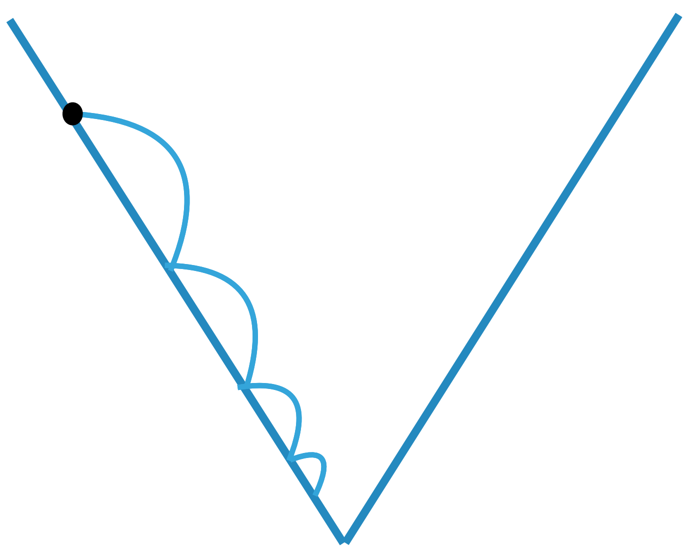
$$\mathbf{D}_{t,i} = \sqrt{\sum_{s=1}^{t} \left(\nabla_i f(x_s) \right)^2} = \eta_{t,i}^{-1}$$

The Unreasonable Effectiveness of Adagrad

It automatically adapts to problem structure

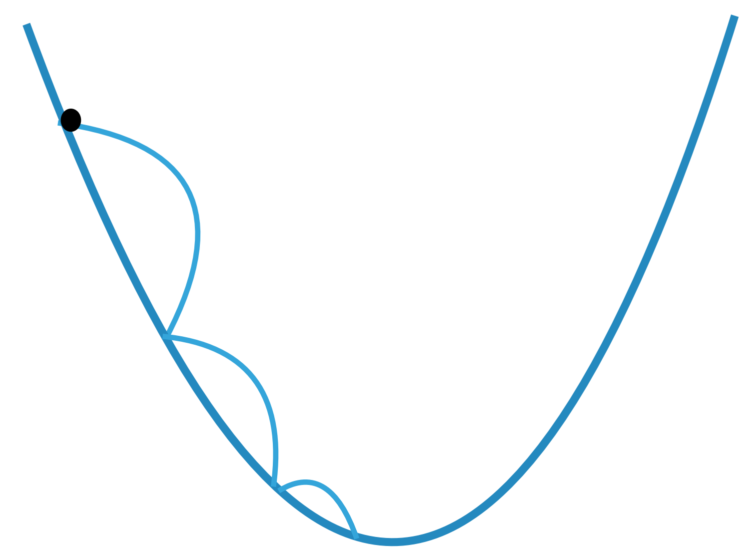
non-smooth

$$\|\nabla f(x)\| \leq G$$



smooth

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\|$$



$$R = \max_{t \in [T]} \|x_t - x^*\|$$

$$T = \frac{G^2 R^2}{\epsilon^2}$$

optimal

[Duchi et al., McMahan & Streeter 2010]

$$T = \frac{\beta R^2}{\epsilon}$$

non-accelerated smooth rate

[Levy 2017, Levy et al. 2018]

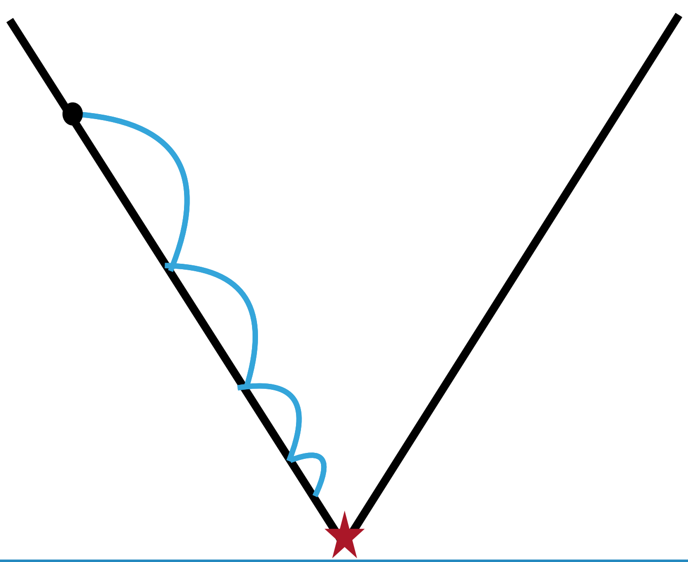
[E., Nguyen, Vladu 2020]

The Unreasonable Effectiveness of Adagrad

It automatically adapts to problem structure

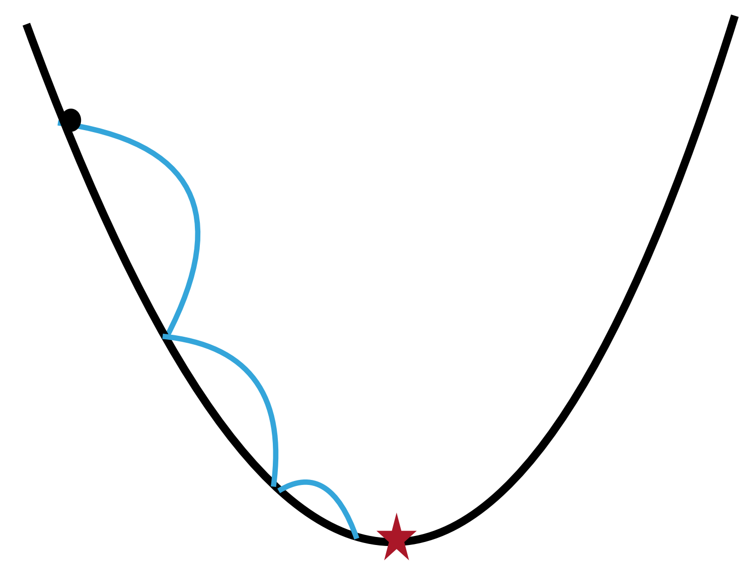
non-smooth

$$\|\nabla f(x)\| \leq G$$



smooth

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\|$$



Intuition

$\|\nabla f(x_t)\|^2$ stays constant

$$\eta_t = \left(\sum_{s=1}^{t} \|\nabla f(x_s)\|^2 \right)^{-1/2} = O\left(\frac{1}{\sqrt{t}}\right)$$

$\|\nabla f(x_t)\|^2$ decays at a $\frac{1}{t}$ rate

$$\eta_t = \left(\sum_{s=1}^{t} \|\nabla f(x_s)\|^2 \right)^{-1/2} = O(1)$$

The Adagrad Family

Adaptive methods for deep learning optimization

- **Adagrad** (Adaptive Gradient) [Duchi et al., McMahan and Streeter, 2010] 7067 citations
- **Adadelta** [Zeiler, 2012]
- **RMSProp** [Hinton, 2014]
- **Adam** (Adaptive Moment Estimation) [Kingma and Ba, 2015] 53803 citations
- **AdaMax** [Kingma and Ba, 2015]
- **Nadam** (Nesterov-accelerated Adaptive Moment Estimation) [Dozat, 2016]

Further Developments

Adaptive method for constrained optimization

- Adagrad+ [E., Nguyen, Vladu 2020]

Accelerated adaptive methods with per-coordinate rates

- JRGS [Joulani et al., 2020]
- AdaACSA, AdaAGD+ [E., Nguyen, Vladu 2020]

(Above works are only the latest in a long line of work)

The accelerated schemes are universal

They automatically achieve the optimal convergence in the non-smooth, smooth, and stochastic settings

Plan for today

Adaptive method for constrained optimization

- Adagrad+ [E., Nguyen, Vladu 2020]

Accelerated adaptive methods with per-coordinate rates

- JRGS [Joulani et al., 2020]
- AdaACSA, AdaAGD+ [E., Nguyen, Vladu 2020]

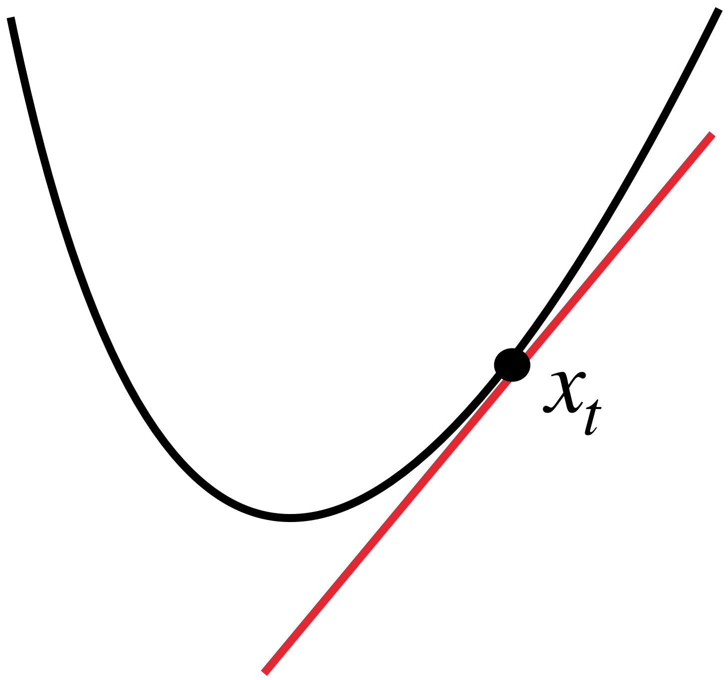
(Above works are only the latest in a long line of work)

The accelerated schemes are universal

They automatically achieve the optimal convergence in the non-smooth, smooth, and stochastic settings

Gradient Descent for Constrained Optimization

$$\min_{x \in X} f(x)$$



Gradient descent algorithm:

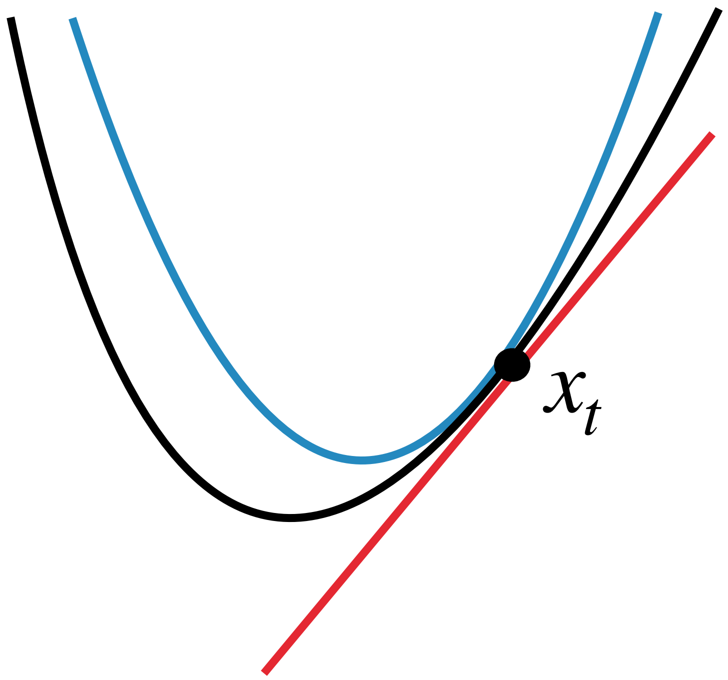
$$x_{t+1} = \arg\min_{x \in X} \left\{ \underbrace{f(x_t) + \langle \nabla f(x_t), x - x_t \rangle}_{\text{linear approx}} + \frac{1}{\eta_t} \cdot \underbrace{\frac{1}{2}\|x - x_t\|^2}_{\text{movement}} \right\}$$

small η_t : put more weight on movement

large η_t : put more weight on linear approximation

Gradient Descent for Constrained Optimization

$$\min_{x \in X} f(x)$$

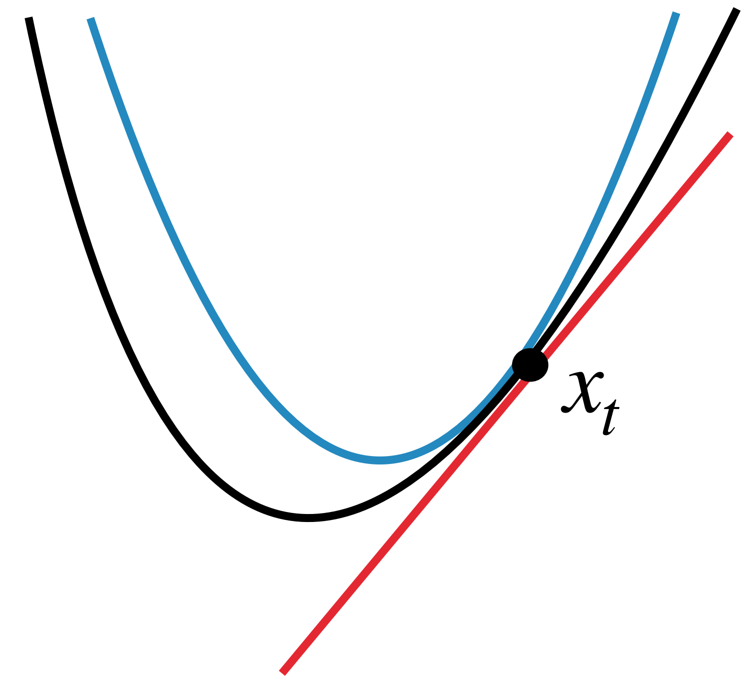


smooth: $\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\|$

$$f(x) \leq \underbrace{f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \frac{\beta}{2}\|x - x_t\|^2}_{\text{quadratic upper obound on } f}$$

Gradient Descent for Constrained Optimization

$$\boxed{\min_{x \in X} f(x)}$$



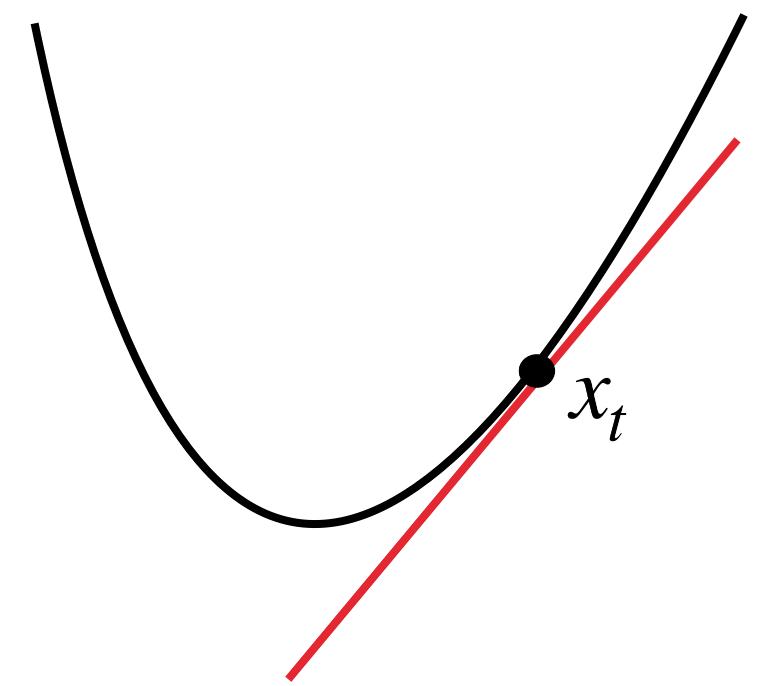
smooth: $\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\|$

$$x_{t+1} = \arg\min_{x \in X} \underbrace{\left\{ f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \frac{\beta}{2}\|x - x_t\|^2 \right\}}_{\text{quadratic upper obound on } f}$$

constant step sizes $\eta_t = \dfrac{1}{\beta}$

Gradient Descent for Constrained Optimization

$$\boxed{\min_{x \in X} f(x)}$$



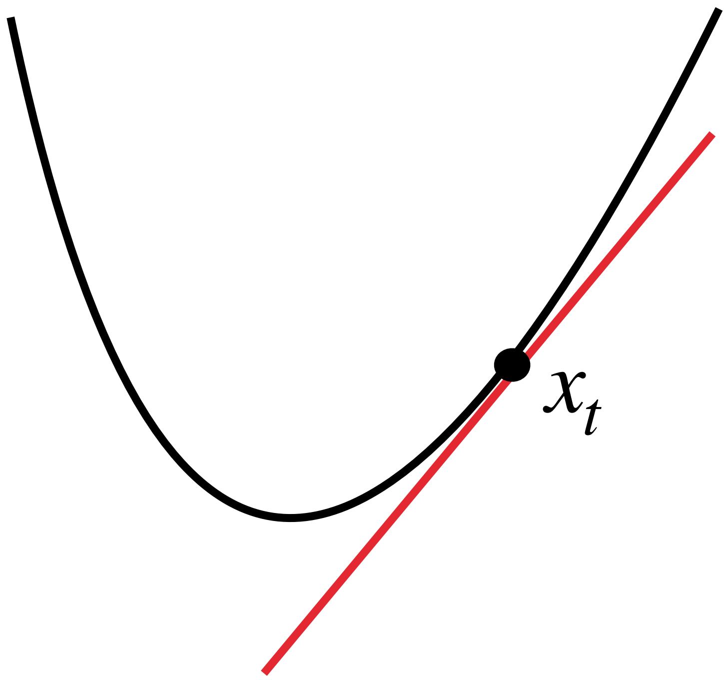
non-smooth: $\|\nabla f(x)\| \le G$

$$x_{t+1} = \arg\min_{x \in X} \left\{ \underbrace{f(x_t) + \langle \nabla f(x_t), x - x_t \rangle}_{\text{linear approx}} + \frac{1}{\eta_t} \cdot \underbrace{\frac{1}{2}\|x - x_t\|^2}_{\text{movement}} \right\}$$

decaying step sizes $\eta_t = \dfrac{R}{G\sqrt{t}}$ (R=diameter of domain)

Adagrad

$$\boxed{\min_{x \in \mathbb{R}^n} f(x)}$$



Mahalanobis norm: $\|x\|_{\mathbf{A}}^2 = \langle x, \mathbf{A}x \rangle$

$$x_{t+1} = \arg\min_{x \in \mathbb{R}^n} \left\{ \underbrace{f(x_t) + \langle \nabla f(x_t), x - x_t \rangle}_{\text{linear approx}} + \frac{1}{2}\|x - x_t\|_{\mathbf{D}_t}^2 \right\}$$

$$\mathbf{D}_{t,i} = \underbrace{\sqrt{\sum_{s=1}^{t} \left(\nabla_i f(x_s) \right)^2}}_{\text{per-coordinate step sizes}}$$

Adagrad for Constrained Optimization

$$\min_{x \in X} f(x)$$

unconstrained: $\nabla f(x^*) = 0$

constrained: $\nabla f(x^*) \neq 0$

Intuition: as we approach x^*, the gradient does not decrease but the iterate movement $\|x_{t+1} - x_t\|$ does

Adagrad+ algorithm:

$$x_{t+1} = \arg\min_{x \in X} \left\{ f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \frac{1}{2} \|x - x_t\|^2_{\mathbf{D}_t} \right\}$$

$$\mathbf{D}^2_{t+1,i} = \mathbf{D}^2_{t,i} \left(1 + \left(x_{t+1,i} - x_{t,i} \right)^2 \right) \quad \text{with} \quad \mathbf{D}_1 = \mathbf{I}$$

Adagrad for Constrained Optimization

$$\min_{x \in X} f(x)$$

unconstrained: $\nabla f(x^*) = 0$

constrained: $\nabla f(x^*) \neq 0$

Intuition: as we approach x^*, the gradient does not decrease but the iterate movement $\|x_{t+1} - x_t\|$ does

Adagrad+ algorithm:

$$x_{t+1} = \arg\min_{x \in X} \left\{ f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \frac{1}{2} \|x - x_t\|^2_{\mathbf{D}_t} \right\}$$

$$\mathbf{D}^2_{t+1,i} = \mathbf{D}^2_{t,i} \left(1 + \frac{\left(x_{t+1,i} - x_{t,i}\right)^2}{R^2_\infty} \right)$$

with $\mathbf{D}_1 = \mathbf{I}$

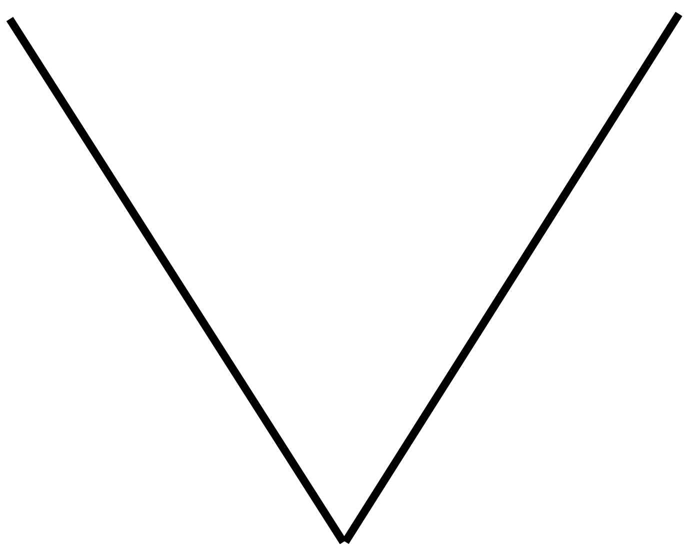
$$R_\infty = \max_{x,y \in X} \|x - y\|_\infty$$

Adagrad+ for Constrained Optimization

It automatically adapts to problem structure

non-smooth

$$\|\nabla f(x)\| \leq G$$

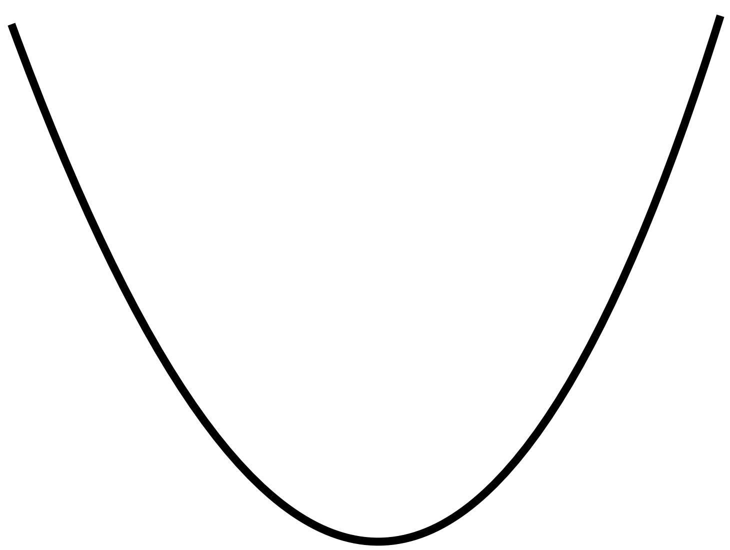


$$T = \tilde{O}\left(\frac{1}{\epsilon^2}\right)$$

optimal (up to logs)

smooth

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta\|x - y\|$$



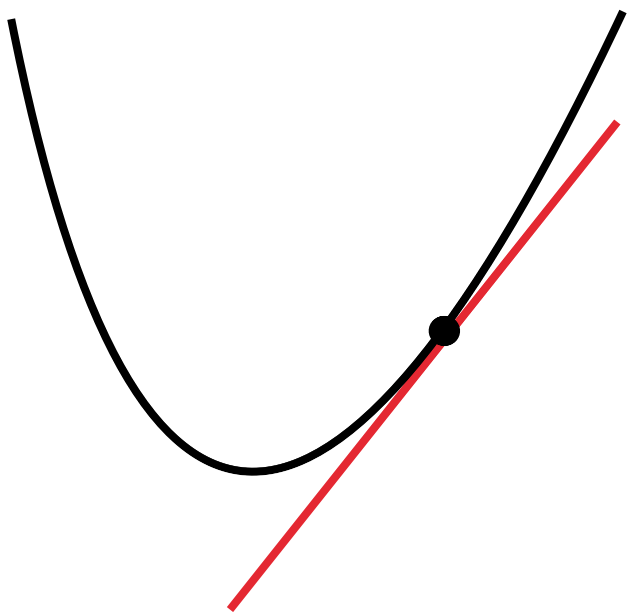
$$T = O\left(\frac{1}{\epsilon}\right)$$

non-accelerated smooth rate

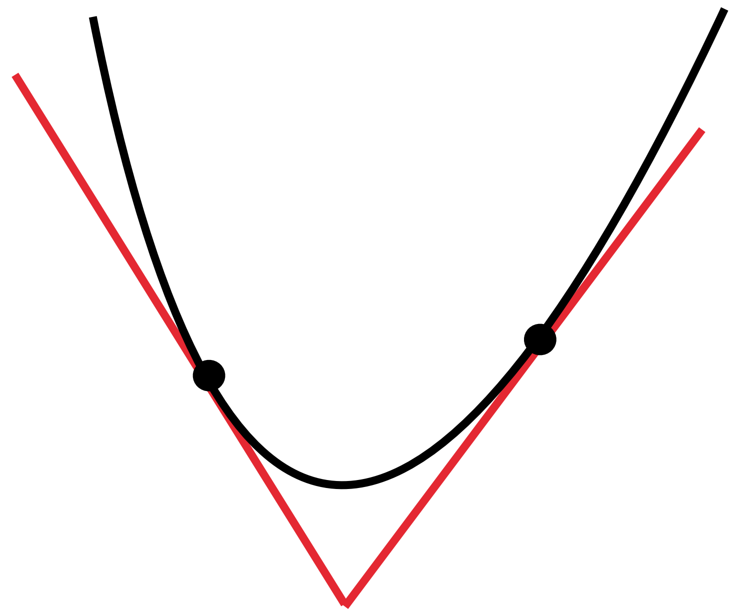
not optimal

Accelerated Methods

- In gradient descent, we use convexity to obtain a lower bound on f

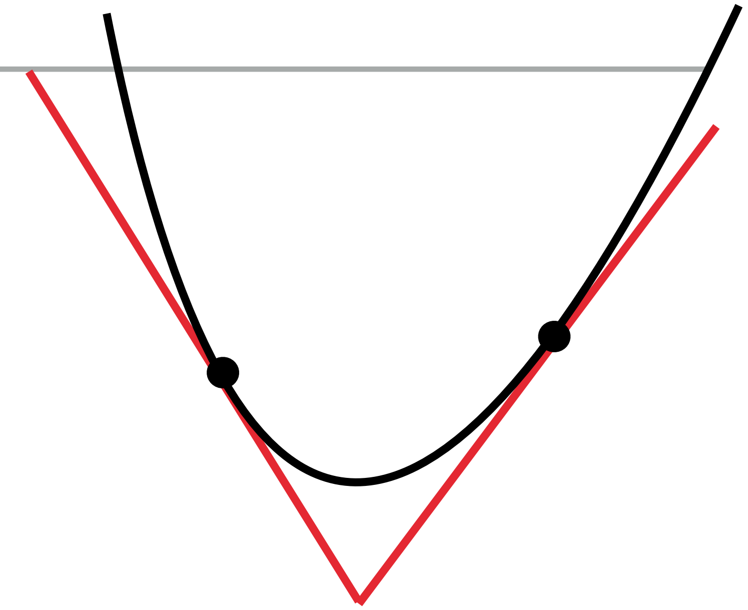


- A single lower bound is useful, but a combination of lower bounds is even better



- At iteration t, use a convex combination of the lower bounds provided by $x_1, x_2, \ldots, x_t$

Accelerated Methods



- At iteration t, use a convex combination of the lower bounds provided by $x_1, x_2, \ldots, x_t$
- Previously, the solutions x_t were both the main solutions as well as the points at which we construct lower bounds
- It is useful to decouple the construction of the solution from the construction of the lower bounds
- We will use the iterates x_t to construct lower bounds as before, but we will use a different sequence of iterates y_t to construct our main solution

Accelerated Methods

AGD+ algorithm [Gasnikov, Nesterov 2016; Cohen et al. 2018]

Choose $y_0 \in X$, weights $a_t \geq 0$, $A_t = \sum_{i=1}^{t} a_i$

For $t = 1, \ldots, T$:

$$x_t = \sum_{i=1}^{t-1} \frac{a_i}{A_t} y_i + \frac{a_t}{A_t} y_{t-1}$$

$$y_t = \arg\min_{x \in \mathbb{X}} \left(\sum_{i=1}^{t} a_i \langle \nabla f(x_i), x \rangle + \frac{\beta}{2} \| x - y_0 \|^2 \right)$$

Return $\sum_{t=1}^{T} \frac{a_t}{A_T} y_t$

Accelerated Methods

AGD+ algorithm [Gasnikov, Nesterov 2016; Cohen et al. 2018]

Choose $y_0 \in X$, weights $a_t = \Theta(t)$, $A_t = \sum_{i=1}^{t} a_i = \Theta(t^2)$

For $t = 1, \ldots, T$:

$$x_t = \sum_{i=1}^{t-1} \frac{a_i}{A_t} y_i + \frac{a_t}{A_t} y_{t-1}$$

$$T = O\left(\sqrt{\frac{1}{\epsilon}}\right) \text{ optimal}$$

$$y_t = \arg\min_{x \in \mathbb{X}} \left(\sum_{i=1}^{t} a_i \langle \nabla f(x_i), x \rangle + \frac{\beta}{2} \| x - y_0 \|^2 \right)$$

Return $\sum_{t=1}^{T} \frac{a_t}{A_T} y_t$

Adaptive AGD+

Set the step size based on the iterate movement $\|y_t - y_{t-1}\|$

AdaAGD+ algorithm

$$x_t = \sum_{i=1}^{t-1} \frac{a_i}{A_t} y_i + \frac{a_t}{A_t} y_{t-1}$$

$$y_t = \arg\min_{x \in \mathbb{X}} \left(\sum_{i=1}^{t} a_i \langle \nabla f(x_i), x \rangle + \frac{1}{2} \| x - y_0 \|^2_{\mathbf{D}_t} \right)$$

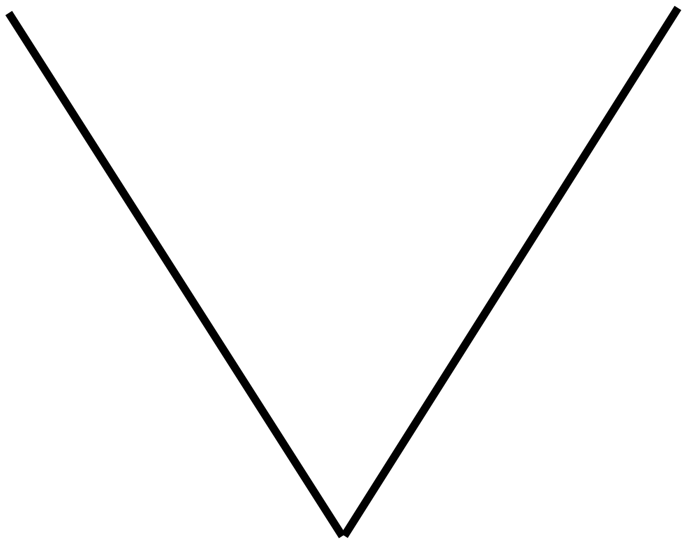
$$\mathbf{D}^2_{t+1,i} = \mathbf{D}^2_{t,i} \left(1 + \frac{\left(y_{t,i} - y_{t-1,i}\right)^2}{R^2_\infty} \right) \quad \text{with} \quad \mathbf{D}_1 = \mathbf{I}$$

AdaAGD+

It automatically adapts to problem structure

non-smooth

$$\|\nabla f(x)\| \leq G$$

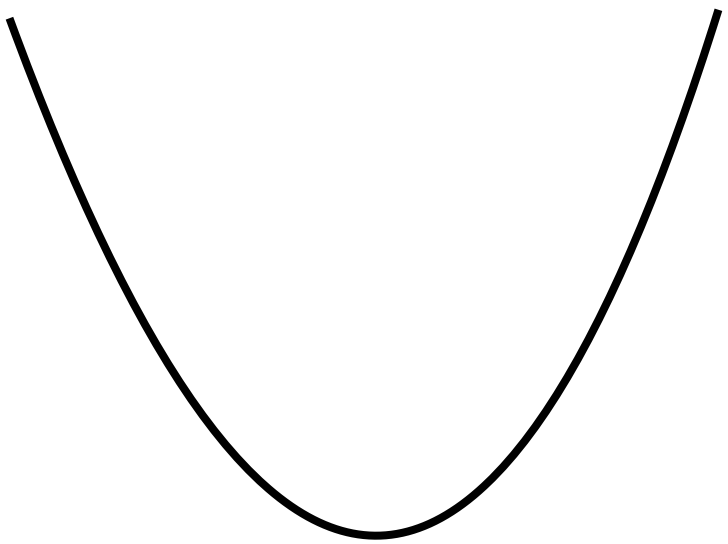


$$T = \tilde{O}\left(\frac{1}{\epsilon^2}\right)$$

optimal (up to logs)

smooth

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta\|x - y\|$$



$$T = O\left(\sqrt{\frac{1}{\epsilon}}\right)$$

optimal

Image Credits

- Images on ML examples slide: Zico Kolter

  http://www.cs.cmu.edu/~15780/

- Gradient descent visualization:
  https://suniljangirblog.wordpress.com/2018/12/03/the-outline-of-gradient-descent/
- Step size cartoon:

  https://cs231n.github.io/neural-networks-3/

- Google Images