Statistical Machine Learning (BE4M33SSU)
Lecture 7a: Stochastic Gradient Descent

Jan Drchal

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

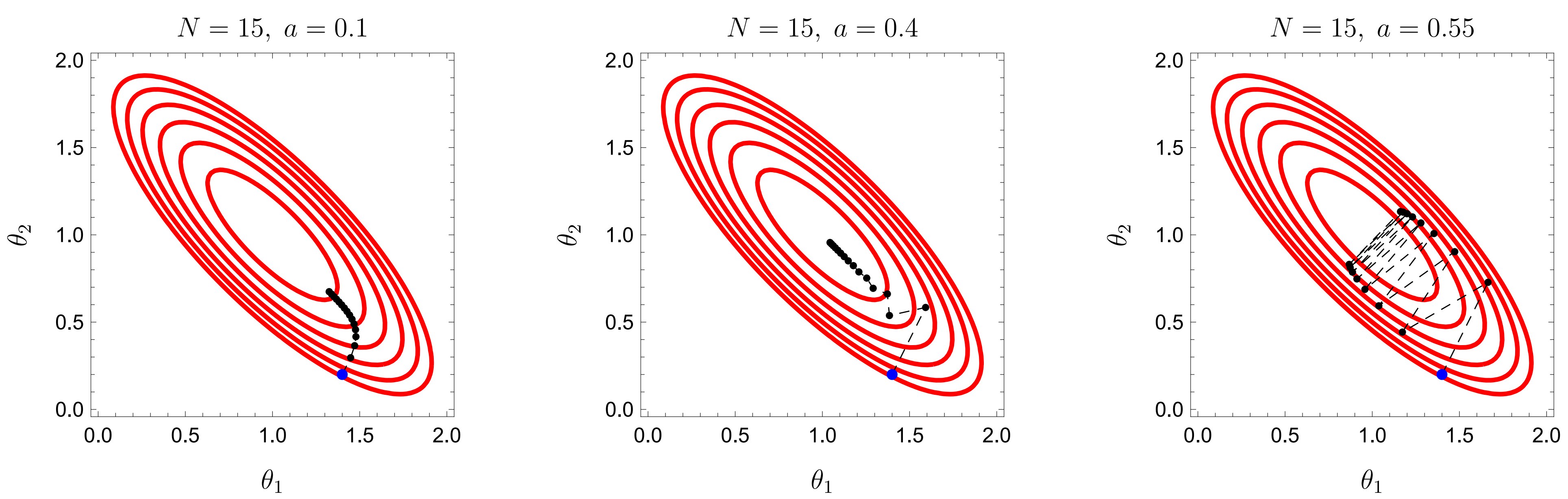
Gradient Descent (GD)

- **Task**: find parameters which minimize loss over the training dataset:

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ is a set of all parameters defining the ANN

- Gradient descent: $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha_k \nabla \mathcal{L}(\boldsymbol{\theta}_k)$
where $\alpha_k > 0$ is the **learning rate** or **stepsize** at iteration k



Stochastic Gradient Descent (SGD): Motivation

- The loss has typically an additive structure, hence:

$$\nabla \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \nabla \ell(y_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$$

- Evaluation of $\nabla \mathcal{L}(\boldsymbol{\theta})$ takes $\mathcal{O}(m)$ time
- What if we have duplicate samples in $\mathcal{T}^m$?
- Online learning, *infinite data* (augmentaion)?
- Specialized loss functions (not based on likelihood)?
- Use a single sample or a *mini-batch* instead of the *full-batch* approach $\Rightarrow$ Stochastic Gradient Descent (SGD)
- Recommended reading: *Bottou, Curtis and Nocedal: Optimization Methods for Large-Scale Machine Learning, 2018*

SGD Algorithm

- Stochastic Gradient Descent

  Choose an initial iterate $\boldsymbol{\theta}_1$

  for $k = 1, 2, \ldots$

      Draw a *batch* $\mathcal{B}_k^M \subset \mathcal{T}^m$

      Compute a stochastic gradient estimate vector $g(\boldsymbol{\theta}_k)$ for $\mathcal{B}_k^M$

      Choose a stepsize $\alpha_k > 0$

      Set the new iterate as $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k - \alpha_k \, g(\boldsymbol{\theta}_k)$

- The stochastic gradient estimate is defined as:

$$g(\boldsymbol{\theta}_k) = \frac{1}{M} \sum_{i=1}^{M} \nabla \ell(y_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)), \; (\boldsymbol{x}_i, y_i) \in \mathcal{B}_k^M$$

Drawing Batches

- Random samples with replacement
  $\Rightarrow$ some training samples may be left unused!
- Shuffle data once and split to batches
- **Shuffle data each epoch before splitting to batches**
- Empirical evidence: *Bottou, Curiously fast convergence of some stochastic gradient descent algorithms, 2009.*

SGD Convergence Theorem: Overview

- The main theorem shows that the *expected optimality gap*

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_k) - \mathcal{L}_*] \xrightarrow{k\to\infty} 0$$

where $\mathcal{L}_*$ is the optimal (minimal) loss

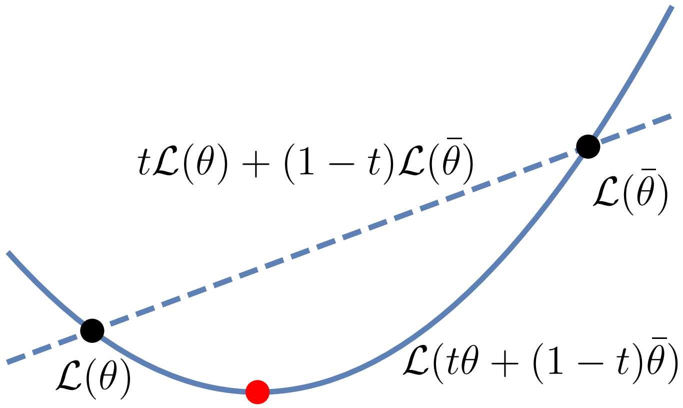
- Selected assumptions:
  1. Strong convexity of $\mathcal{L}$
  2. Lipschitz continuous gradient $\nabla\mathcal{L}$
  3. Bounds on $\mathcal{L}$ and $g(\boldsymbol{\theta}_k)$:
     - directions of $g(\boldsymbol{\theta}_k)$ and $\nabla\mathcal{L}(\boldsymbol{\theta}_k)$ *similar*,
     - their norms are also *similar*

Convexity

- Convex function definition:

$$\mathcal{L}(t\boldsymbol{\theta} + (1-t)\bar{\boldsymbol{\theta}}) \leq t\mathcal{L}(\boldsymbol{\theta}) + (1-t)\mathcal{L}(\bar{\boldsymbol{\theta}})$$

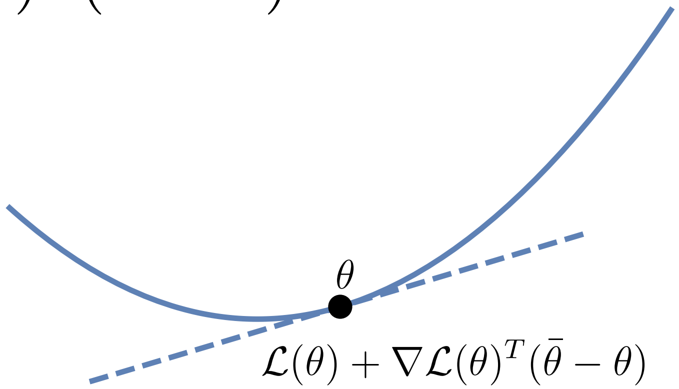
for all $(\boldsymbol{\theta}, \bar{\boldsymbol{\theta}}) \in \mathbb{R}^d \times \mathbb{R}^d$



- Equivalently (first-order condition):

$$\mathcal{L}(\bar{\boldsymbol{\theta}}) \geq \mathcal{L}(\boldsymbol{\theta}) + \nabla\mathcal{L}(\boldsymbol{\theta})^T(\bar{\boldsymbol{\theta}} - \boldsymbol{\theta})$$

the function lies above all its tangents



- See A4B33OPT
- But we need a stronger assumption...

Assumption 1: Strong Convexity

- The loss function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is strongly convex if there exists constant $c > 0$ such that

$$\mathcal{L}(\bar{\boldsymbol{\theta}}) \geq \mathcal{L}(\boldsymbol{\theta}) + \nabla \mathcal{L}(\boldsymbol{\theta})^T (\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}) + \frac{1}{2} c \left\| \bar{\boldsymbol{\theta}} - \boldsymbol{\theta} \right\|_2^2$$

  for all $(\boldsymbol{\theta}, \bar{\boldsymbol{\theta}}) \in \mathbb{R}^d \times \mathbb{R}^d$

- *Intuition*: quadratic lower bound on function growth
- Constant c **quantifies** the level of convexity $=>$ higher c indicates "more convex" function

Assumption 2: Lipschitz Continuous Gradient

- The loss function is continuously differentiable and the gradient is *Lipschitz continuous* with *Lipschitz constant* $L > 0$:

$$\left\|\nabla\mathcal{L}(\boldsymbol{\theta}) - \nabla\mathcal{L}(\bar{\boldsymbol{\theta}})\right\|_2 \leq L \left\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\right\|_2, \text{ for all } (\boldsymbol{\theta}, \bar{\boldsymbol{\theta}}) \in \mathbb{R}^d \times \mathbb{R}^d$$

- *Intuition:* the gradient does not change too quickly w.r.t. $\boldsymbol{\theta}$
- Provides a **quantified** indicator for how far to move to decrease $\mathcal{L}$

Assumptions Summary

constant	description	higher value means
$c > 0$	strong convexity (lower bound)	"more convex"
$L > 0$	Lipschitz continuous gradient (upper bound)	higher gradient change allowed
$\mu > 0$	$g(\boldsymbol{\theta}_k)$ direction comparable to $\nabla\mathcal{L}(\boldsymbol{\theta}_k)$	smaller angular difference between $\mathbb{E}[g(\boldsymbol{\theta}_k)]$ and $\nabla\mathcal{L}(\boldsymbol{\theta}_k)$
$M \geq 0$	compares norms of $\mathbb{E}[g(\boldsymbol{\theta}_k)]$ and $\nabla\mathcal{L}(\boldsymbol{\theta}_k)$	higher variance in norms allowed

SGD Convergence: Strongly Convex $\mathcal{L}$, Fixed Stepsize

- **Theorem (simplified):** if the assumptions $\uparrow$ hold, the SGD is run with a fixed (and *bound*) stepsize $\alpha_k = \alpha$ for all $k \in \mathbb{N}$. Then the *expected optimality gap* satisfies the following for all k:

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_k) - \mathcal{L}_*] \leq \frac{\alpha L M}{2c\mu} + \mathcal{O}(\rho^k) \xrightarrow{k\to\infty} \frac{\alpha L M}{2c\mu},$$

  where $\rho \in [0, 1)$ is a constant

- In general, for the fixed stepsize, the *optimality gap* tends to zero, but converges to $\frac{\alpha L M}{2c\mu} \geq 0$
- Note on α: lower L and higher μ allow longer stepsize
- Note on c: having $c > 0$ is critical to keep $\rho < 1$

Full-Batch Gradient Descent

- How does the theorem apply to the full-batch setting (GD)?
- The $g(\boldsymbol{\theta}_k)$ is an unbiased estimate of $\nabla\mathcal{L}(\boldsymbol{\theta}_k)$:

$$\mathbb{E}[g(\boldsymbol{\theta}_k)] = \nabla\mathcal{L}(\boldsymbol{\theta}_k)$$

- Zero variance implies $M = 0$, hence, $\frac{\alpha L M}{2c\mu} = 0$
- The optimality gap simplifies to:

$$\epsilon_k = \mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_k) - \mathcal{L}_*] \leq \mathcal{O}(\rho^k) \xrightarrow{k\to\infty} 0$$

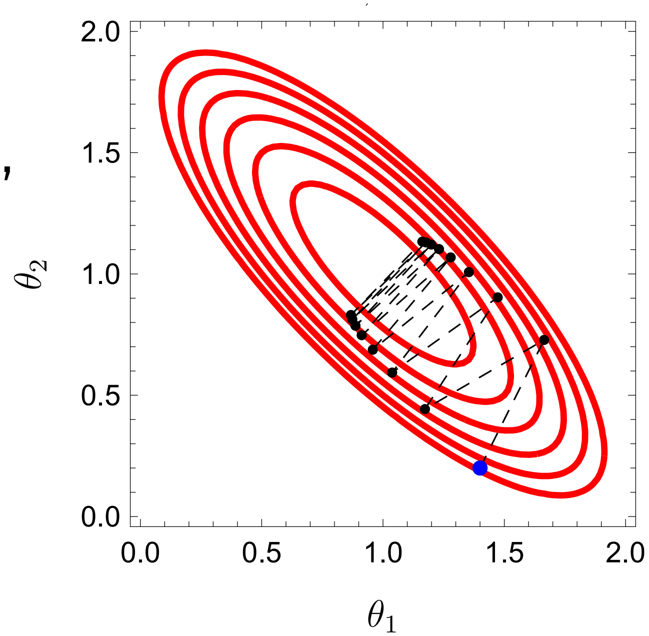
- For any given gap ϵ, the number of iterations k is proportional to $\log(1/\epsilon)$ in the worst case.

SGD Convergence: Strongly Convex $\mathcal{L}$, Diminishing Stepsize

- **Theorem (simplified):** if the assumptions ↑ hold and the SGD is run with a diminishing stepsize ($\alpha_k \approx$ inversely proportional to k). Then the *expected optimality gap* satisfies the following for all k:

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_k) - \mathcal{L}_*] \leq \mathcal{O}(1/k) \xrightarrow{k\to\infty} 0$$

- The number of iterations k is proportional to $1/\epsilon$ in the worst case
- The diminishing stepsize is needed to compensate for the noise & to prevent overshooting the minimum
- Note: in practise α_k is often reduced in steps, e.g., halved after each N epochs



SGD for Nonconvex Objectives

- Corresponding theorems can be proven for nonconvex objectives
- For assumptions similar to the theorem for the diminishing stepsizes (and excluding the strong convexity) we get:

$$\lim_{k \to \infty} \mathbb{E}\left[\|\nabla \mathcal{L}(\boldsymbol{\theta}_k)\|_2^2 \right] = 0$$

GD vs SGD

	GD	SGD
time per iteration	m	1
iterations for accuracy ϵ	$\log(1/\epsilon)$	$1/\epsilon$
time for accuracy ϵ	$m\log(1/\epsilon)$	$1/\epsilon$
error for limited time budget T_{max}	$\frac{\log(T_{max})}{T_{max}} + \frac{1}{T_{max}}$	$\frac{1}{T_{max}}$

- SGD time does not depend on dataset size (if not exhausted)
- For large-scale problems (large m) SGD is faster
- For limited time budget T_{max} SGD achieves lower error
- It is harder to tune stepsize schedule for SGD, but you can experiment on a small representative subset of the dataset
- In practise *mini-batches* are used to leverage optmization/parallelization on CPU/GPU

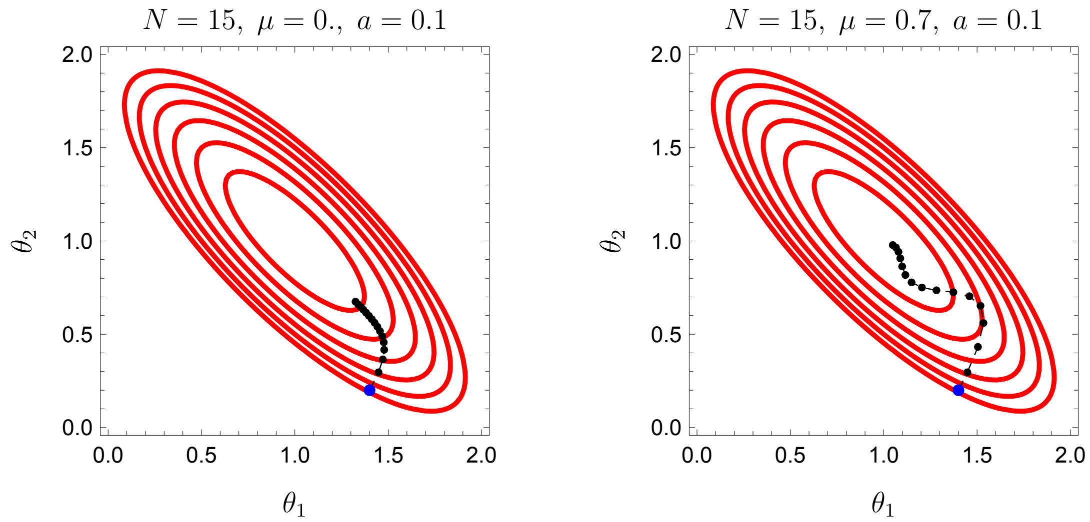
Momentum

- Simulate inertia to overcome plateaus in the error landscape:

$$\boldsymbol{v}_{k+1} \leftarrow \mu \boldsymbol{v}_k - \alpha_k \, g(\boldsymbol{\theta}_k)$$
$$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \boldsymbol{v}_{k+1}$$

where $\mu \in [0,1]$ is the *momentum parameter*

- Momentum damps oscillations in directions of high curvature
- It builds velocity in directions with consistent (possibly small) gradient

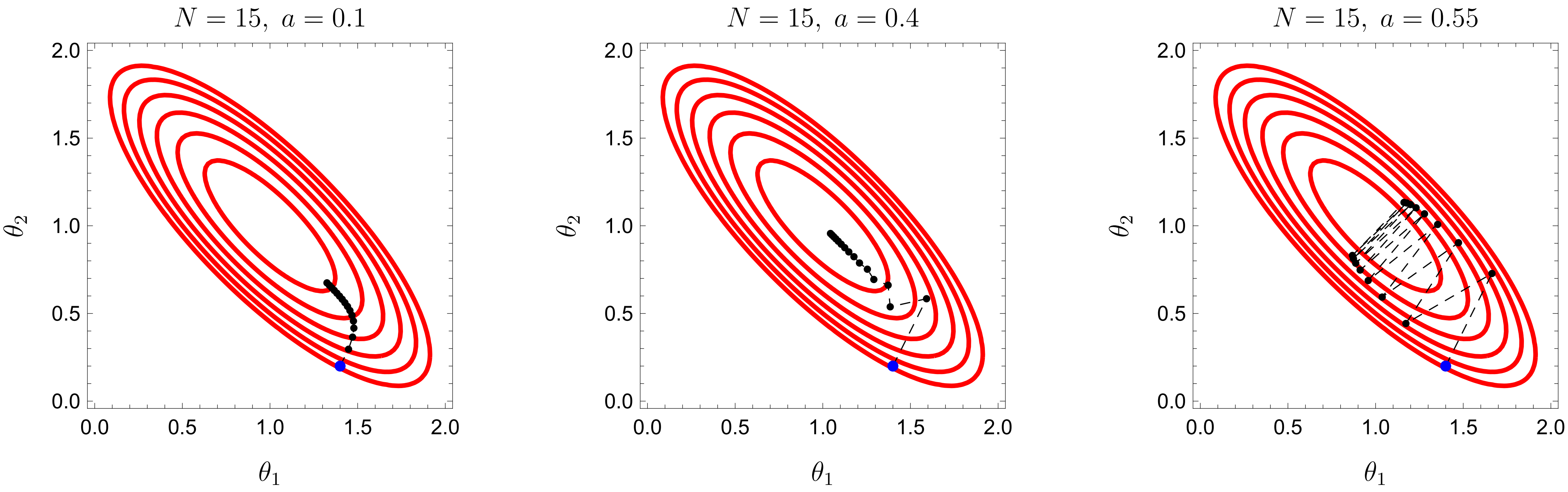


Adagrad

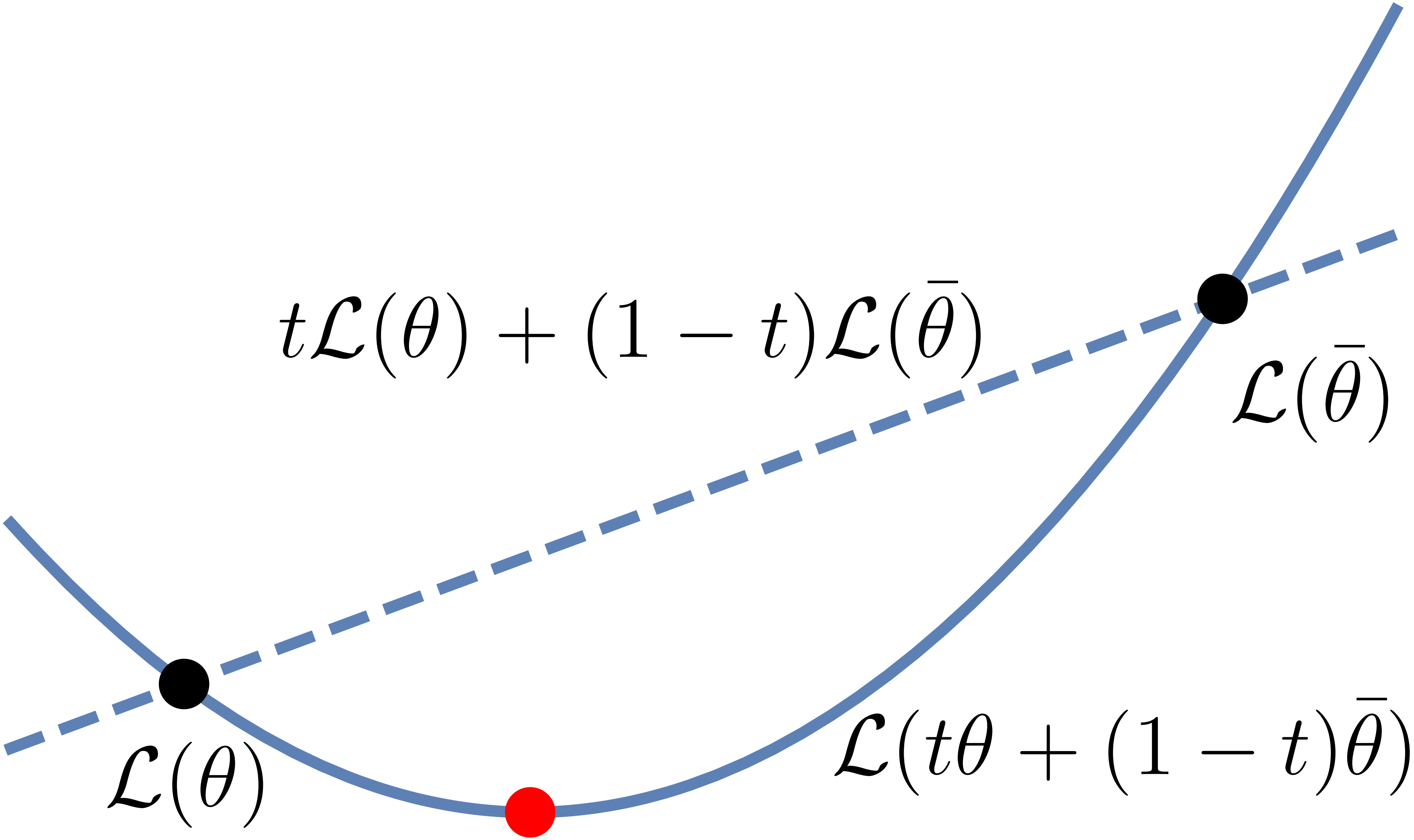
- Adaptive Gradient method (Duchi, Hazan and Singer, 2011)
- *Motivation:* a magnitude of gradient differs a lot for different parameters
- *Idea:* reduce learning rates for parameters having high values of gradient

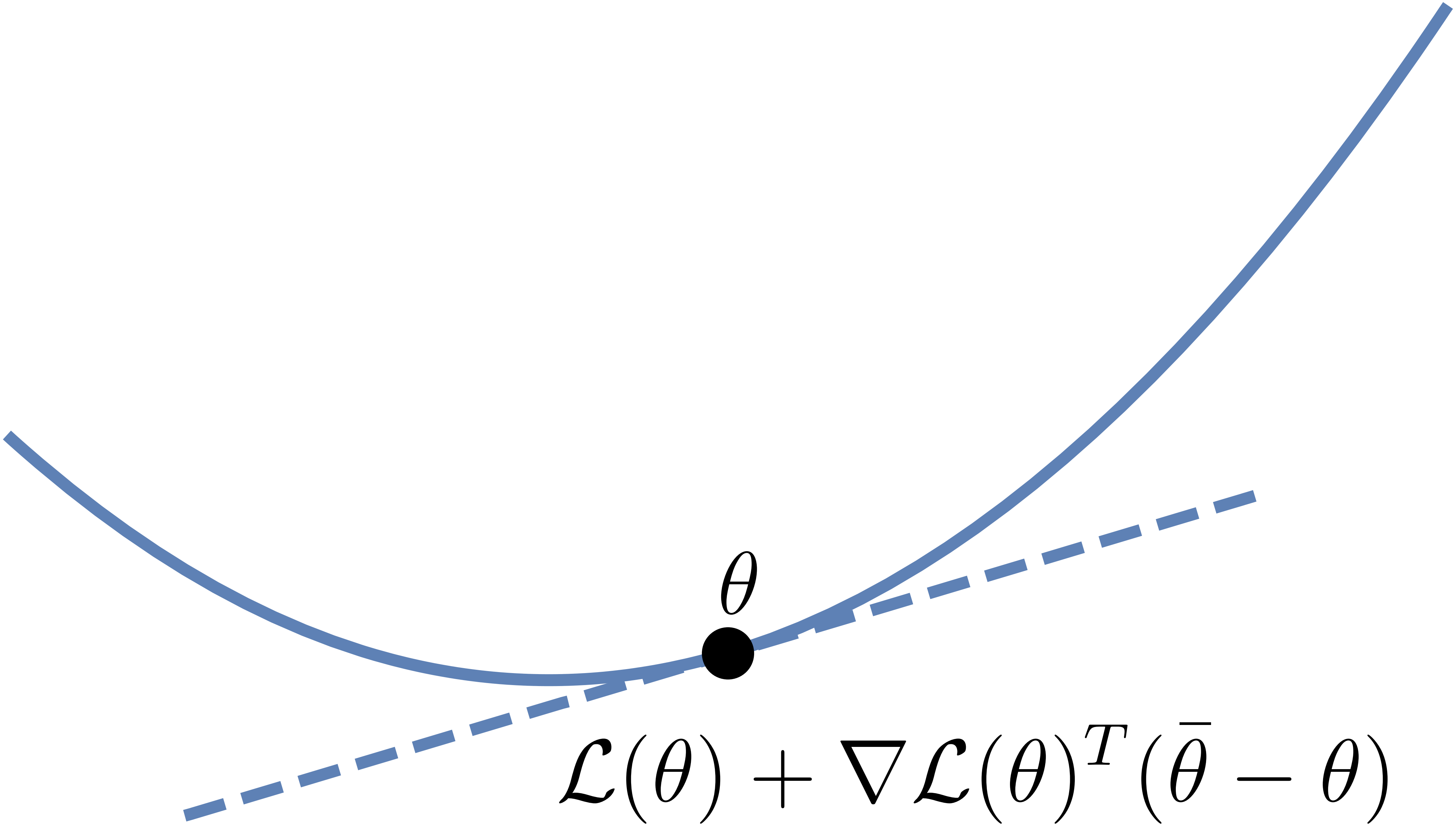
$$
\begin{aligned}
G_{k+1,i} &\leftarrow G_{k,i} + [g(\boldsymbol{\theta}_k)]_i^2 \\
\theta_{k+1,i} &\leftarrow \theta_{k,i} - \frac{\alpha}{\sqrt{G_{k+1,i}} + \epsilon} \cdot [g(\boldsymbol{\theta}_k)]_i
\end{aligned}
$$

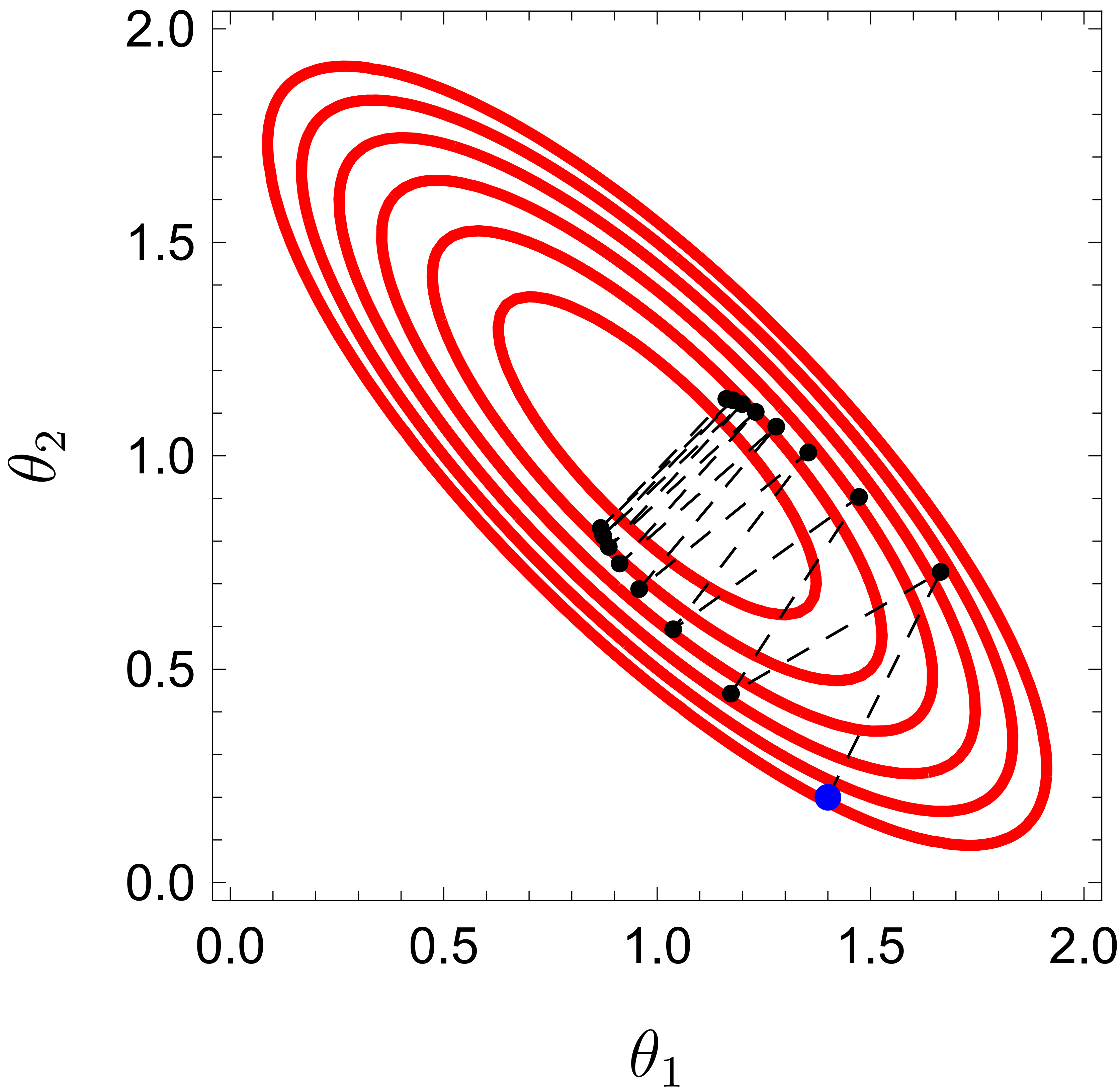
- $G_{k,i}$ accumulates squared partial derivative approximations w.r.t. to the parameter $\theta_{k,i}$
- ϵ is a small positive number to prevent division by zero
- *Weakness:* ever increasing G_i leads to slow convergence eventually
- Better methods: RMSProp, Adam, . . .



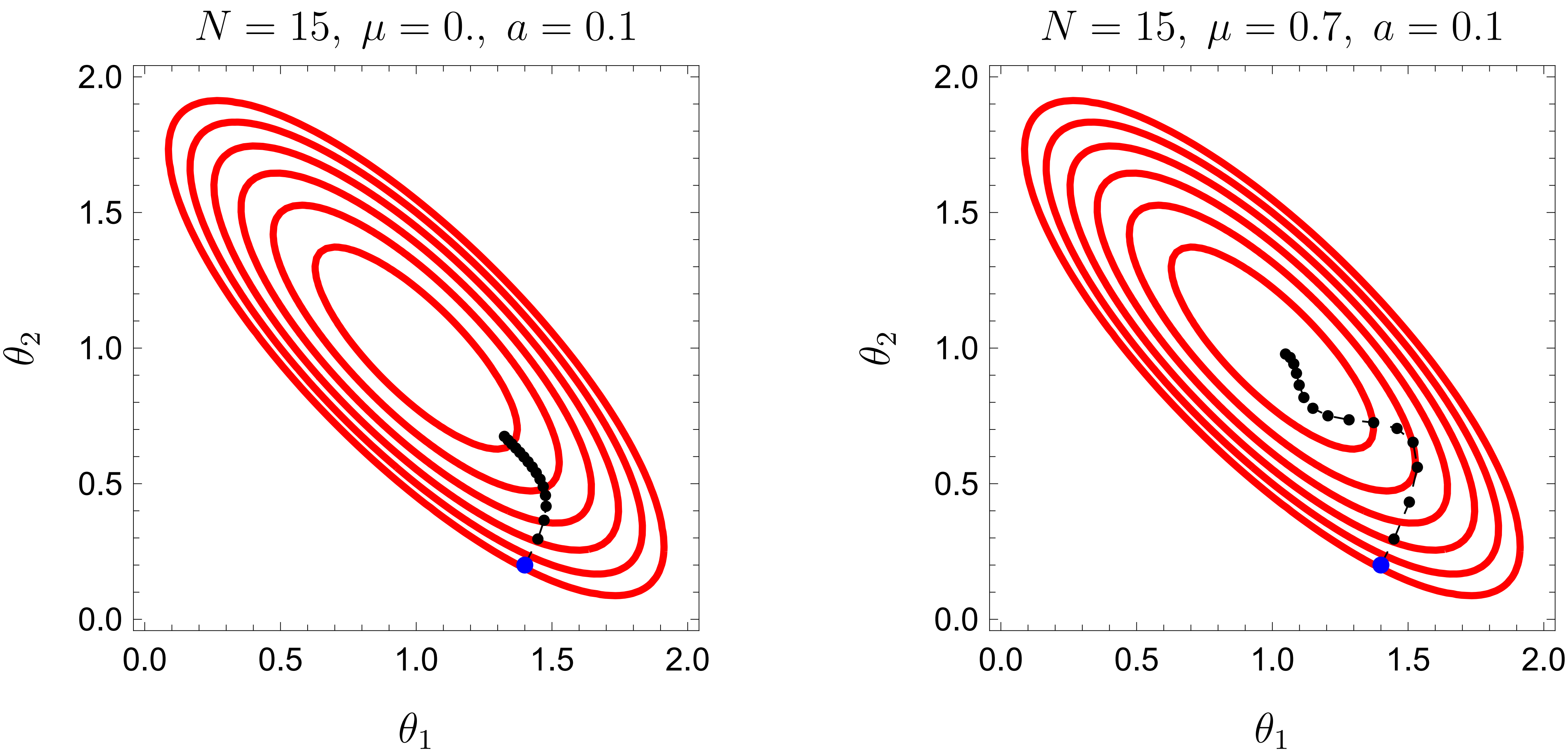
$N = 15,\ a = 0.1$
$N = 15,\ a = 0.4$
$N = 15,\ a = 0.55$
θ_2
θ_1
0.0
0.5
1.0
1.5
2.0


$t\mathcal{L}(\theta) + (1-t)\mathcal{L}(\bar{\theta})$
$\mathcal{L}(\bar{\theta})$
$\mathcal{L}(\theta)$
$\mathcal{L}(t\theta + (1-t)\bar{\theta})$


θ
$\mathcal{L}(\theta) + \nabla\mathcal{L}(\theta)^T(\bar{\theta} - \theta)$



2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
θ_2
θ_1



$N = 15,\ \mu = 0.,\ a = 0.1$
$N = 15,\ \mu = 0.7,\ a = 0.1$
θ_2
θ_1
0.0
0.5
1.0
1.5
2.0