SGDの確率的ノイズを利用した段階的最適化手法による経験損失関数の大域的最適化

情報論的学習理論と機械学習研究会(IBISML)
12/21(土) 13:40〜14:00

明治大学 佐藤尚樹
明治大学 飯塚秀明

背景：機械学習（cクラスの画像分類）

入力

$\boldsymbol{z}_i \in \mathbb{R}^d$

$(i = 1, 2, \cdots, N)$

$d = 512 \times 512 \times 3$

$N = 3000000$

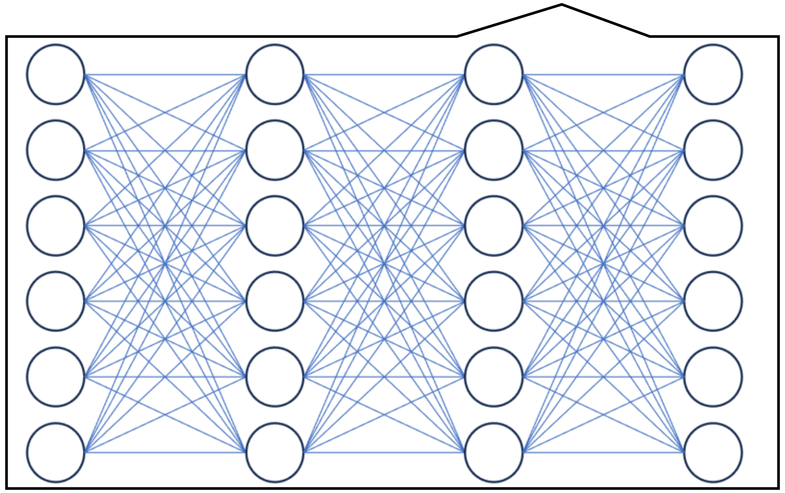
データセット

関数 $g : \mathbb{R}^d \to \mathbb{R}^c$

変数: $\boldsymbol{x} \in \mathbb{R}^D$,

$\boldsymbol{z}_i \in \mathbb{R}^d$

Deep Neural Network



$D =$ 20万〜5兆

出力

$g(\boldsymbol{x}, \boldsymbol{z}_i) \in \mathbb{R}^c$

犬:index3

正解

$\boldsymbol{y}_i \in \mathbb{R}^c$

猫:index2

誤差 $\|g(\boldsymbol{x}, \boldsymbol{z}_i) - \boldsymbol{y}_i\|$

誤差の平均

$$f(\boldsymbol{x}) := \frac{1}{N} \sum_{i=1}^{N} \|g(\boldsymbol{x}, \boldsymbol{z}_i) - \boldsymbol{y}_i\|$$

関数 $f(\boldsymbol{x})$**を最適化する**

背景：勾配法

$$\boldsymbol{x}_{t+1} := \boldsymbol{x}_t + \eta_t \boldsymbol{d}_t$$

学習率　探索方向

▷**最急降下法** (Gradient Descent)

$$\boldsymbol{d}_t := -\nabla f(\boldsymbol{x}_t)$$

全勾配

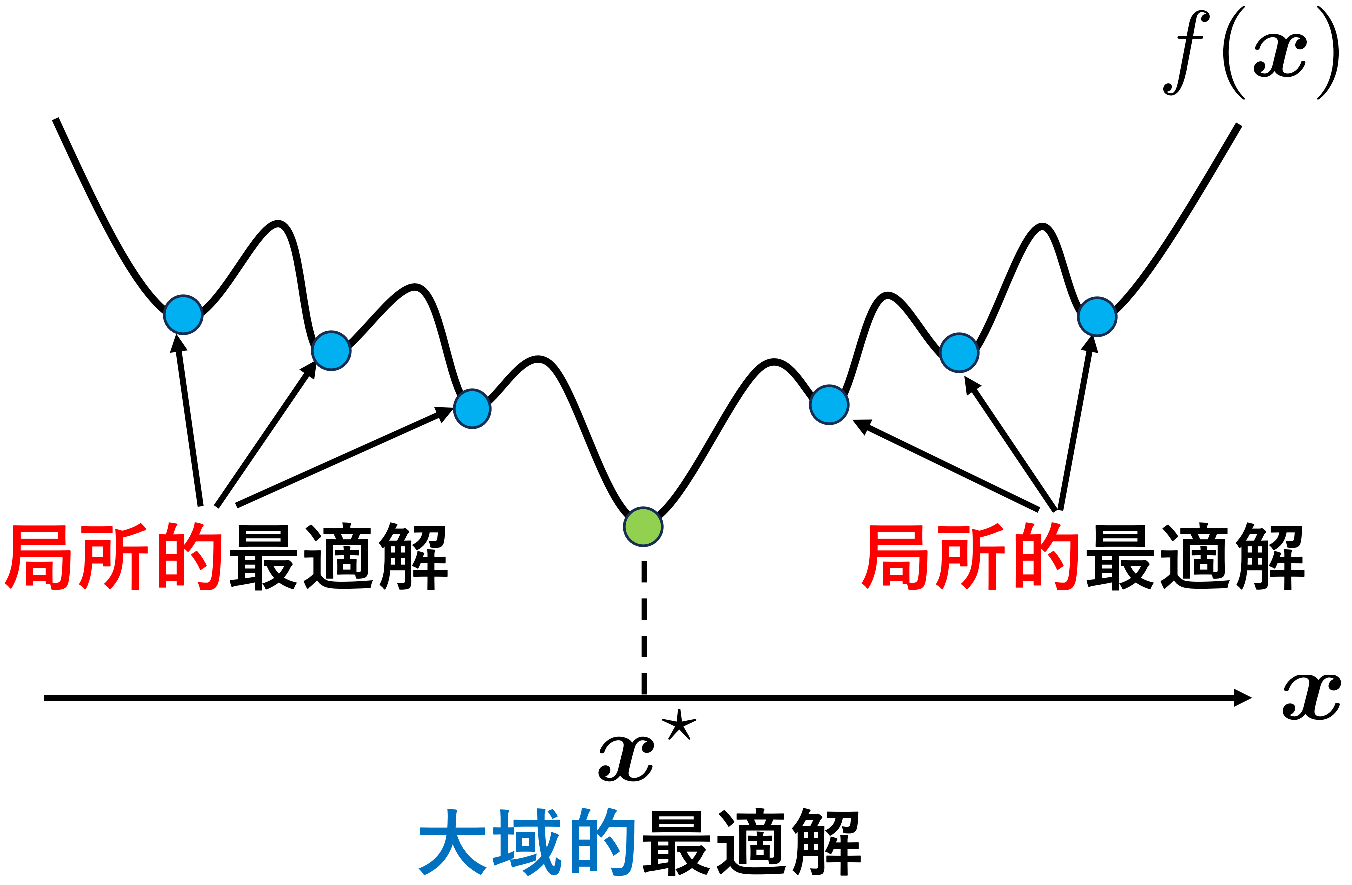
▷**確率的勾配降下法** (Stochastic Gradient Descent, SGD)

$$\boldsymbol{d}_t := -\nabla f_{S_t}(\boldsymbol{x}_t) = -\frac{1}{b}\sum_{i=1}^{b} \mathsf{G}_{\xi_{t,i}}(\boldsymbol{x}_t)$$

ミニバッチ確率的勾配　バッチサイズ　確率的勾配

ランダムに選ばれた b 個の確率的勾配の平均で代用(ex. b=32, 1024)

背景：大域的最適化の難しさ

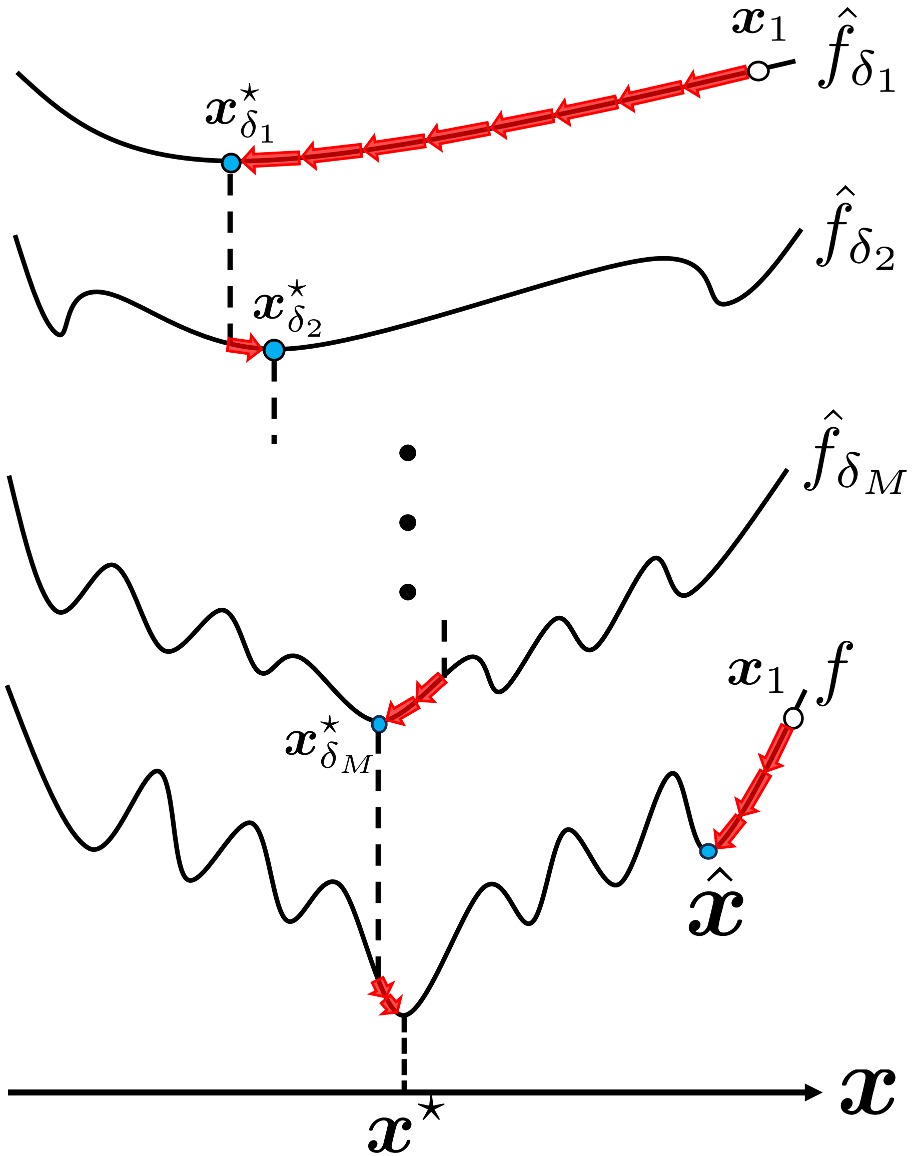


機械学習に使用される最適化アルゴリズムは、ほぼ勾配を利用する手法

勾配に頼っている手法は、山を登れない

非凸関数の大域的最適解を見つけることは難しい

背景：段階的最適化 (Graduated Optimization)



▷1987年に提案された**大域的**最適化手法

▷徐々に小さくなるノイズで**平滑化**された関数の列を順番に最適化する

背景：関数の平滑化

関数 $f: \mathbb{R}^d \to \mathbb{R}$ を、大きさ $\delta \in \mathbb{R}$ のノイズで平滑化した関数は、

$$\hat{f}_\delta(\boldsymbol{x}) := \mathbb{E}_{\underline{\boldsymbol{u}} \sim \underline{\mathcal{L}}}\left[f(\boldsymbol{x} - \underline{\delta}\boldsymbol{u})\right]$$

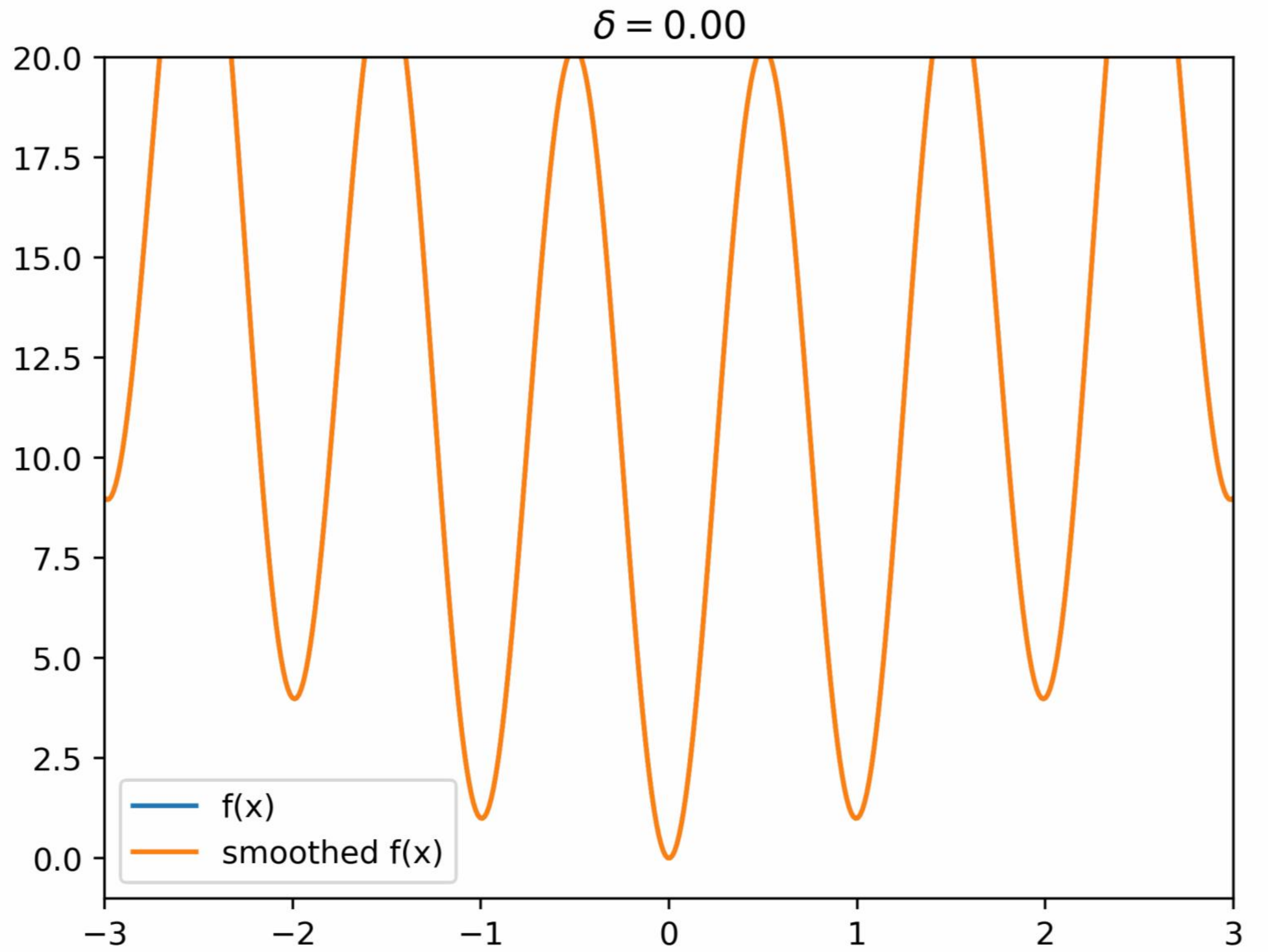
確率変数
$\boldsymbol{u} \in \mathbb{R}^d$

任意の裾の軽い分布
(正規分布,一様分布等)

ノイズスケール
(平滑化の度合い)

で定義する．ただし, $\mathbb{E}_{u \sim \mathcal{L}}\left[\|u\|\right] \leq 1$ を満たすとする．

背景：関数の平滑化



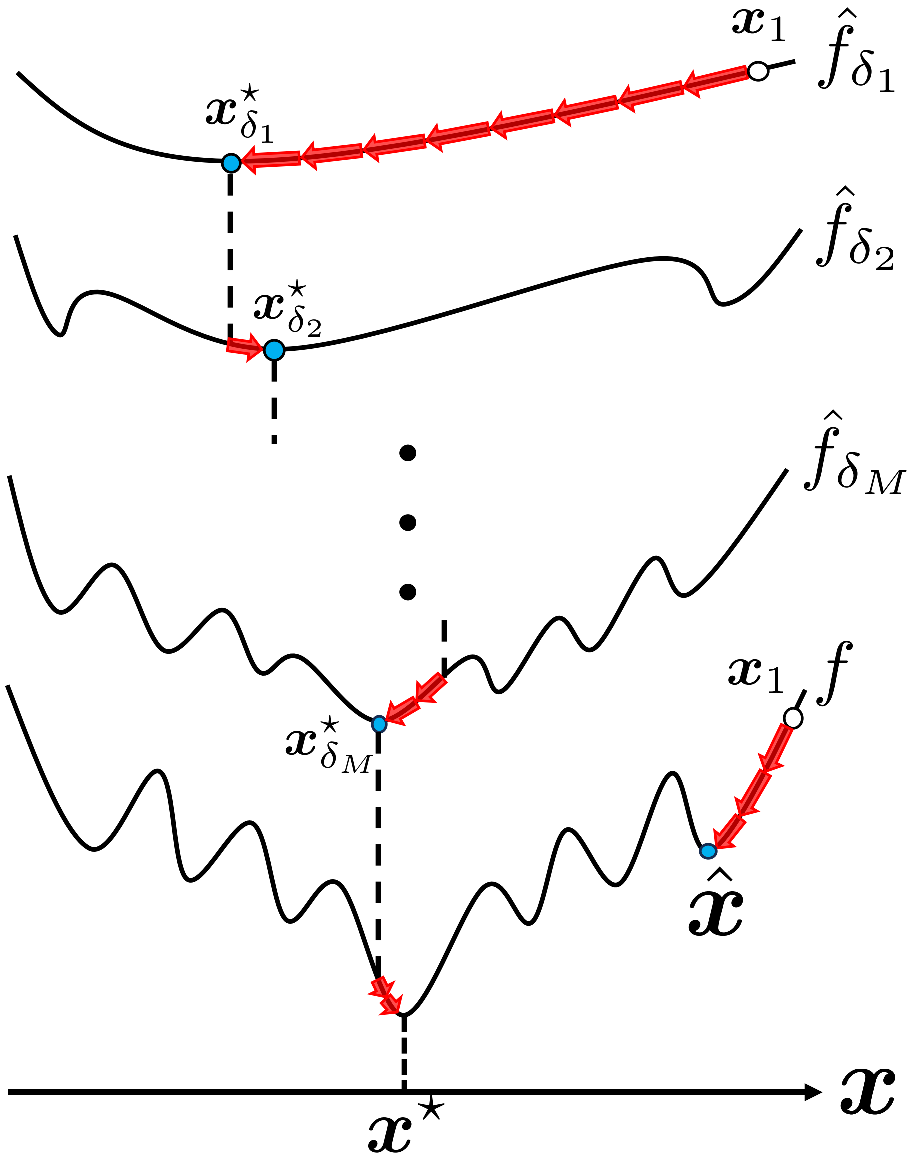
▷**1変数のRastrigin's 関数**

$$f(x) = x^2 - 10\cos(2\pi x) + 10$$

▷**目的関数の平滑化**

$$\hat{f}_\delta(\boldsymbol{x}) := \mathbb{E}_{\boldsymbol{u}\sim\mathcal{N}\left(\mathbf{0},\frac{1}{\sqrt{d}}I_d\right)}\left[f(\boldsymbol{x}-\delta\boldsymbol{u})\right]$$

背景：段階的最適化 (Graduated Optimization)



▷1987年に提案された大域的最適化手法

▷徐々に小さくなるノイズで平滑化された関数の列を順番に最適化する

$$\boldsymbol{x}^\star := \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^d} \hat{f}(\boldsymbol{x}),\ \ \boldsymbol{x}^\star_\delta := \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^d} \hat{f}_\delta(\boldsymbol{x})$$

▷関数の平滑化

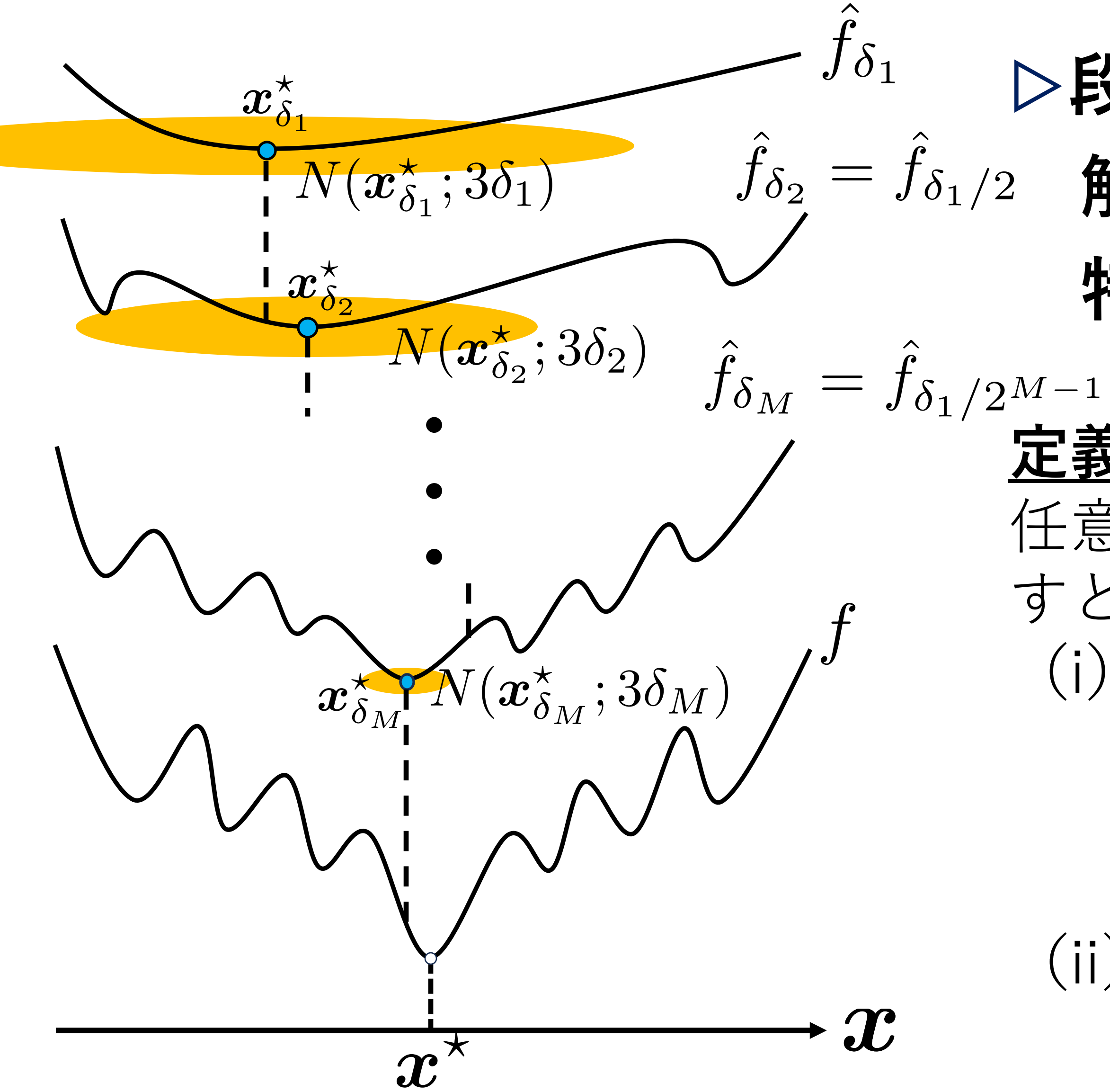
$$\hat{f}_\delta(\boldsymbol{x}) := \mathbb{E}_{\boldsymbol{u}\sim\mathcal{L}}\left[f(\boldsymbol{x} - \delta\boldsymbol{u})\right]$$

▷平滑化の度合い δ を減少させる.

$$\delta_1 = 1,\ \delta_2 = 0.5,\ \delta_3 = 0.25, \cdots$$

背景：σ-nice 関数[1]

[1] E. Hazan et al. "On Graduated Optimization for Stochastic Non-Convex Problems" In *Proceedings of the 33th International Conference on Machine Learning*, pp1833-1841, 2016



▷**段階的最適化アルゴリズムが大域的最適解に収束するために必要な性質を満たす特別な非凸関数の族。**

定義

任意の非凸関数 $f\colon \mathbb{R}^d \to \mathbb{R}$ が次の2つの条件を満たすとき、関数 f は σ-nice 関数であるという。

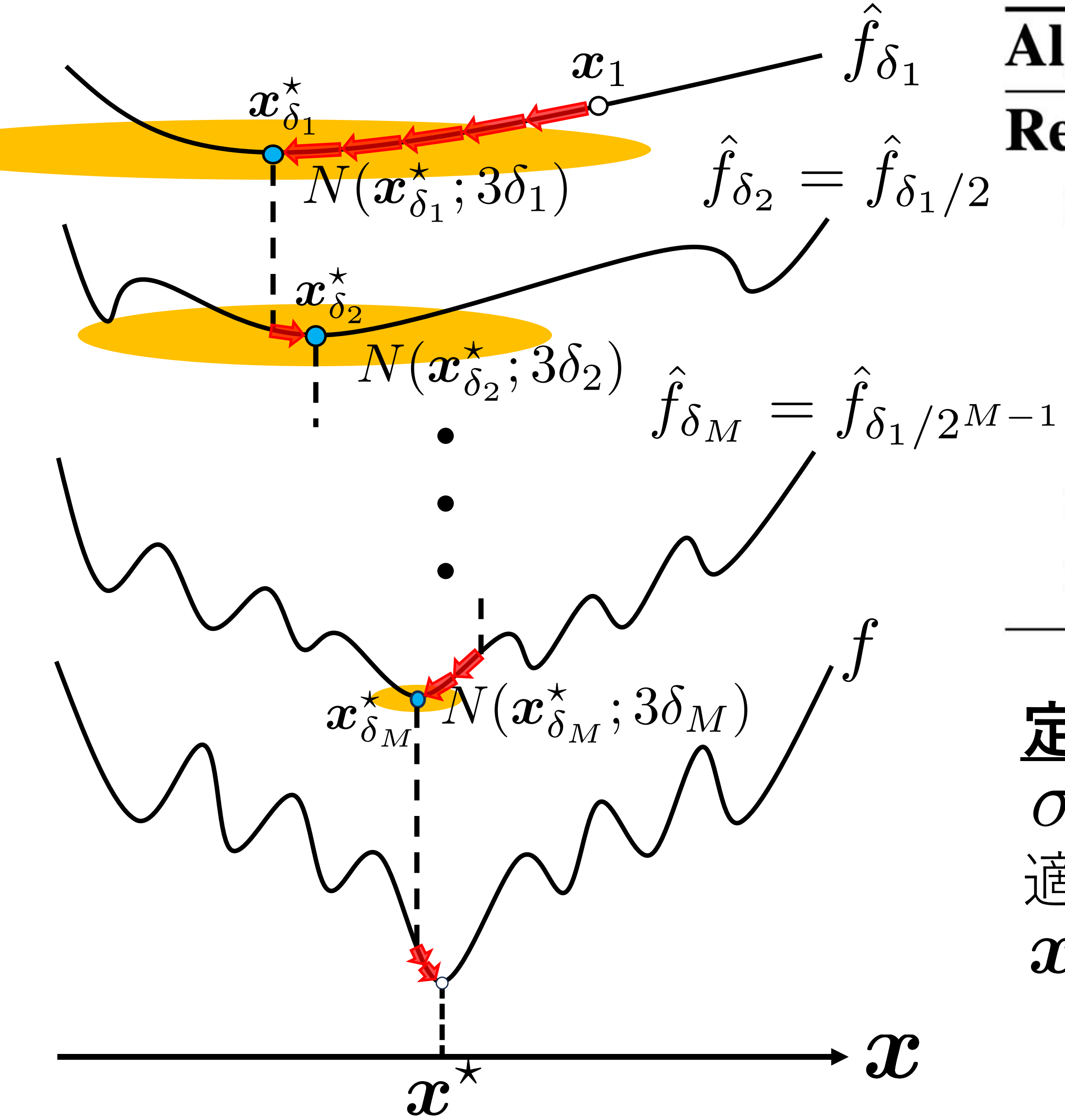
(i) 任意の $\delta > 0$ と $\boldsymbol{x}^\star_\delta$ に対して、$\boldsymbol{x}^\star_{\delta/2}$ が存在して、

$$\left\|\boldsymbol{x}^\star_\delta - \boldsymbol{x}^\star_{\delta/2}\right\| \leq \frac{\delta}{2}$$

を満たす。

(ii) 任意の $\delta > 0$ に対して、関数 $\hat{f}_\delta(\boldsymbol{x})$ は近傍 $N(\boldsymbol{x}^\star_\delta; 3\delta)$ で σ–強凸となる。

背景：明示的な段階的最適化

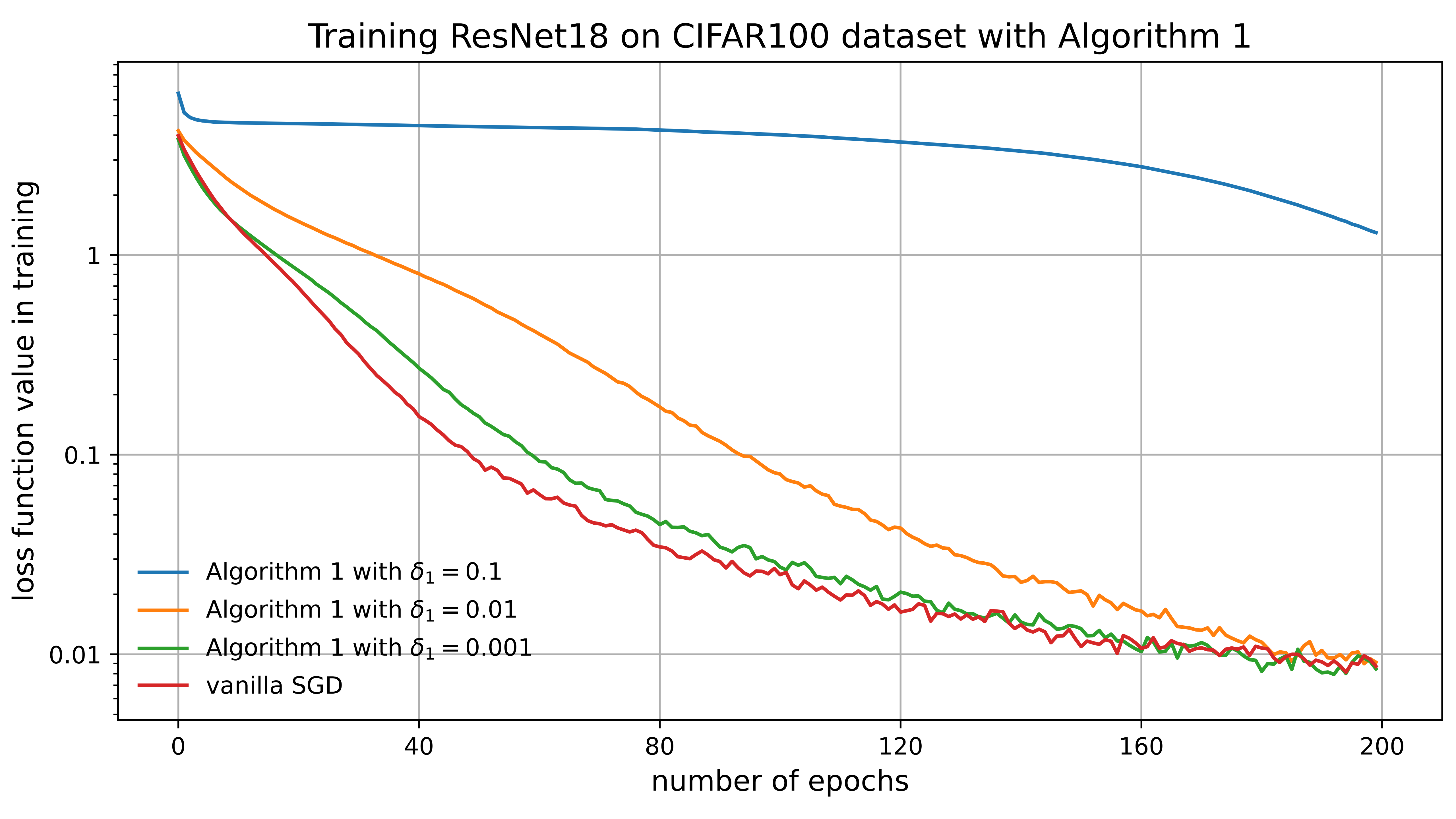


Algorithm 1 Graduated optimization

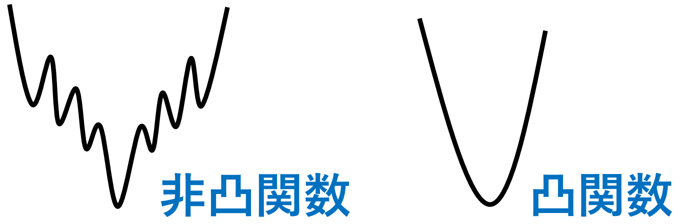
Require: $\delta_1 > 0, \boldsymbol{x}_1 \in N(\boldsymbol{x}^\star_{\delta_1}; 3\delta_1)$
  for $m = 1$ to M **do**
    $T_m := \sigma\delta_m^2/32$ ←反復回数を決定
    $\boldsymbol{x}_{m+1} := \mathrm{SGD}(T_m, \boldsymbol{x}_m, \hat{f}_{\delta_m})$ ←←←←平滑化された関数をSGD等で最適化
    $\delta_{m+1} := \delta_m/2$ ←平滑化の度合いを更新
  end for
  return $\boldsymbol{x}_{M+1}$

定理5.1 [1]

σ-nice 関数 $f\colon \mathbb{R}^d \to \mathbb{R}$ に対して、Algorithm 1を適用すると、$\mathcal{O}\left(1/\epsilon^2\right)$ 回の反復で f の大域的最適解 $\boldsymbol{x}^\star$ の ϵ-近傍に到達する。

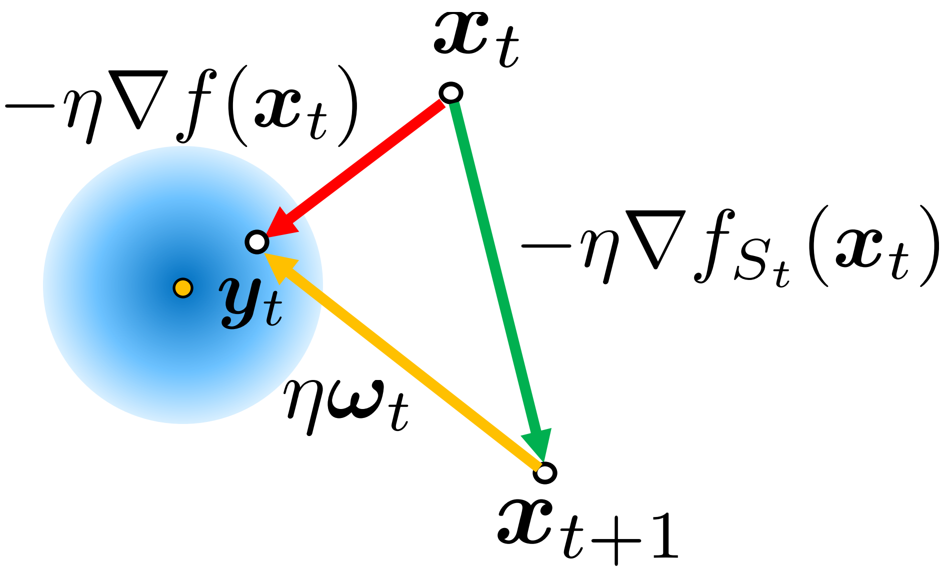

Training ResNet18 on CIFAR100 dataset with Algorithm 1
loss function value in training
1
0.1
0.01
0
40
80
120
160
200
number of epochs
Algorithm 1 with $\delta_1 = 0.1$
Algorithm 1 with $\delta_1 = 0.01$
Algorithm 1 with $\delta_1 = 0.001$
vanilla SGD

背景：SGDの確率的ノイズ



▷非凸関数の最適化において，確率的勾配降下法(SGD)は、
なぜか最急降下法(GD)よりも汎化性の高い解に収束する。

▷SGDの確率的ノイズが役立っている？[2]



$$\text{(GD)}\ \boldsymbol{y}_t := \boldsymbol{x}_t - \eta\nabla f(\boldsymbol{x}_t)$$

$$\text{(SGD)}\ \boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \eta\nabla f_{S_t}(\boldsymbol{x}_t)$$

確率的ノイズ $\boldsymbol{\omega}_t := \underbrace{\nabla f_{S_t}(\boldsymbol{x}_t)}_{\text{SGDの探索方向}} - \underbrace{\nabla f(\boldsymbol{x}_t)}_{\text{GDの探索方向（最急降下方向）}}$

[2] R. Kleinberg et al. “An Alternative View: When Does SGD Escape Local Minima?”
In *Proceedings of the 35th International Conference on Machine Learning*, pp2703-2712, 2018

背景：SGDの確率的ノイズと平滑化

(GD) $\boldsymbol{y}_t := \boldsymbol{x}_t - \eta \nabla f(\boldsymbol{x}_t)$

(SGD) $\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \eta \nabla f_{S_t}(\boldsymbol{x}_t)$

▷**目的関数の平滑化**

$\hat{f}_\delta(\boldsymbol{x}) := \mathbb{E}_{\boldsymbol{u}\sim\mathcal{L}}\left[f(\boldsymbol{x} - \delta \boldsymbol{u})\right]$

(**確率的ノイズ**) $\boldsymbol{\omega}_t := \nabla f_{S_t}(\boldsymbol{x}_t) - \nabla f(\boldsymbol{x}_t)$ とすると、次の式が成り立つ。

$$\mathbb{E}_{\boldsymbol{\omega}_t}\left[\boldsymbol{y}_{t+1}\right] = \boldsymbol{y}_t - \eta \nabla \underbrace{\mathbb{E}_{\boldsymbol{\omega}_t}\left[f(\boldsymbol{y}_t - \eta \boldsymbol{\omega}_t)\right]}_{:=\hat{f}(\boldsymbol{y}_t)}$$

▷**”関数 $f(\boldsymbol{x}_t)$ をSGDで最適化すること”と、”関数 $\hat{f}(\boldsymbol{y}_t)$を最急降下法で最適化すること”は、期待値の意味では等価であると言える。**

▷**ある程度平滑化された関数が、最急降下法で最適化されているとみなせる。**

[2] R. Kleinberg et al. “An Alternative View: When Does SGD Escape Local Minima?”
In *Proceedings of the 35th International Conference on Machine Learning*, pp2703-2712, 2018

動機

- ▷SGDが持つ確率的ノイズには関数を平滑化する効果がある。
  - ▷平滑化の度合い, すなわちノイズレベル δ が何によって定まるのかを明らかにしたい。
  - ▷訓練中にノイズレベル δ が徐々に小さくなるようにすることで、SGDを利用した段階的最適化アルゴリズムを構築したい。
  - ▷DNNを含む非凸関数の大域的最適化を実現したい。

貢献

- ▷SGDの確率的ノイズによる平滑化の度合いは、
$$\delta = \eta\sqrt{\frac{C^2}{b}}$$
で表せることを示した。ただし、η は学習率、b はバッチサイズ、C^2 は確率的勾配の分散とする。
- ▷SGDの平滑化特性を利用した暗黙的な段階的最適化アルゴリズムを提案し、それが $\mathcal{O}\left(1/\epsilon^2\right)$ 回の反復で σ-nice 関数の大域的最適解 $x^\star$ の ϵ-近傍に到達できることを示した。

準備：最適化問題

経験損失最小化問題

▷訓練データセット $S := (z_1, z_2, \cdots, z_n)$

▷Deep Neural Network のパラメータ $\boldsymbol{x} \in \mathbb{R}^d$

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}; S) = \frac{1}{n}\sum_{i=1}^{n} \underline{\underline{l(\boldsymbol{x}; z_i)}}$$

i 番目の訓練データ z_i に対する損失関数

▷非凸目的関数 $f : \mathbb{R}^d \to \mathbb{R}$ を最小化する。

準備：仮定

仮定

(A1)関数$f\colon \mathbb{R}^d \to \mathbb{R}$は連続的微分可能で、$L_g$-平滑とする。

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d\colon \ \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L_g \|\boldsymbol{x} - \boldsymbol{y}\|$$

(A2)関数$f\colon \mathbb{R}^d \to \mathbb{R}$は$L_f$-リプシッツ関数とする。

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d\colon |f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L_f \|\boldsymbol{x} - \boldsymbol{y}\|$$

(A3) $(\boldsymbol{x}_t)_{t\in\mathbb{N}} \subset \mathbb{R}^d$を最適化手法によって生成された点列とするとき、

(i)任意の$t \in \mathbb{N}$に対して、次の式が成り立つとする。

$$\mathbb{E}_{\xi_t}\left[\mathsf{G}_{\xi_t}(\boldsymbol{x}_t)\right] = \nabla f(\boldsymbol{x}_t)$$

(ii)次の式を満たす非負定数C^2が存在するとする。

$$\mathbb{E}_{\xi_t}\left[\|\mathsf{G}_{\xi_t}(\boldsymbol{x}_t) - \nabla f(\boldsymbol{x}_t)\|^2\right] \leq C^2$$ ←確率的勾配の分散

準備：仮定

仮定続き

(A4) 時刻$t \in \mathbb{N}$で、全勾配∇fはミニバッチ$\mathcal{S}_t$で次のように近似されるとする。

$$\nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t) := \frac{1}{b}\sum_{i\in[b]} \mathsf{G}_{\xi_{t,i}}(\boldsymbol{x}_t) = \frac{1}{b}\sum_{\{i:\ \boldsymbol{z}_i\in \mathsf{S}_t\}} \nabla l_i(\boldsymbol{x}_t)$$

ミニバッチ確率的勾配

確率的勾配

補題2.1

仮定(A3)(ii)と(A4)が成り立つとすると、任意の$t \in \mathbb{N}$に対して、

$$\mathbb{E}_{\xi_t}\left[\left\| \underbrace{\nabla f_{\mathcal{S}_t}(\boldsymbol{x}_t)}_{\text{SGDの探索方向}} - \underbrace{\nabla f(\boldsymbol{x}_t)}_{\substack{\text{GDの探索方向}\\ \text{（最急降下方向）}}} \right\|^2\right] \le \frac{C^2}{b}$$

が成り立つ。

背景：SGDの確率的ノイズと平滑化(再掲)

(GD) $\boldsymbol{y}_t := \boldsymbol{x}_t - \eta \nabla f(\boldsymbol{x}_t)$

(SGD) $\boldsymbol{x}_{t+1} := \boldsymbol{x}_t - \eta \nabla f_{S_t}(\boldsymbol{x}_t)$

▷**目的関数の平滑化**

$\hat{f}_\delta(\boldsymbol{x}) := \mathbb{E}_{\boldsymbol{u}\sim\mathcal{L}}\left[f(\boldsymbol{x} - \delta \boldsymbol{u})\right]$

(**確率的ノイズ**) $\boldsymbol{\omega}_t := \nabla f_{S_t}(\boldsymbol{x}_t) - \nabla f(\boldsymbol{x}_t)$ とすると、次の式が成り立つ。

$$\mathbb{E}_{\boldsymbol{\omega}_t}\left[\boldsymbol{y}_{t+1}\right] = \boldsymbol{y}_t - \eta \nabla \underbrace{\mathbb{E}_{\boldsymbol{\omega}_t}\left[f(\boldsymbol{y}_t - \eta \boldsymbol{\omega}_t)\right]}_{:=\hat{f}(\boldsymbol{y}_t)}$$

▷**”関数 $f(\boldsymbol{x}_t)$ をSGDで最適化すること”と、”関数 $\hat{f}(\boldsymbol{y}_t)$を最急降下法で最適化すること”は、期待値の意味では等価であると言える。**

▷**ある程度平滑化された関数が、最急降下法で最適化されているとみなせる。**

[2] R. Kleinberg et al. “An Alternative View: When Does SGD Escape Local Minima?”
In *Proceedings of the 35th International Conference on Machine Learning*, pp2703-2712, 2018

SGDの平滑化特性

補題2.1

$$\mathbb{E}_{\xi_t}\left[\|\nabla f_{S_t}(\boldsymbol{x}_t) - \nabla f(\boldsymbol{x}_t)\|^2\right] \leq \frac{C^2}{b}$$

▷**目的関数の平滑化**

$$\hat{f}_\delta(\boldsymbol{x}) := \mathbb{E}_{\boldsymbol{u}\sim\mathcal{L}}\left[f(\boldsymbol{x} - \delta\boldsymbol{u})\right]$$

▷ $\boldsymbol{\omega}_t := \nabla f_{S_t}(\boldsymbol{x}_t) - \nabla f(\boldsymbol{x}_t)$と補題2.1から、

$$\boldsymbol{\omega}_t = \frac{C}{\sqrt{b}}\boldsymbol{u}_t \; (\boldsymbol{u}_t \sim \mathcal{L})$$

が成り立つ。したがって、

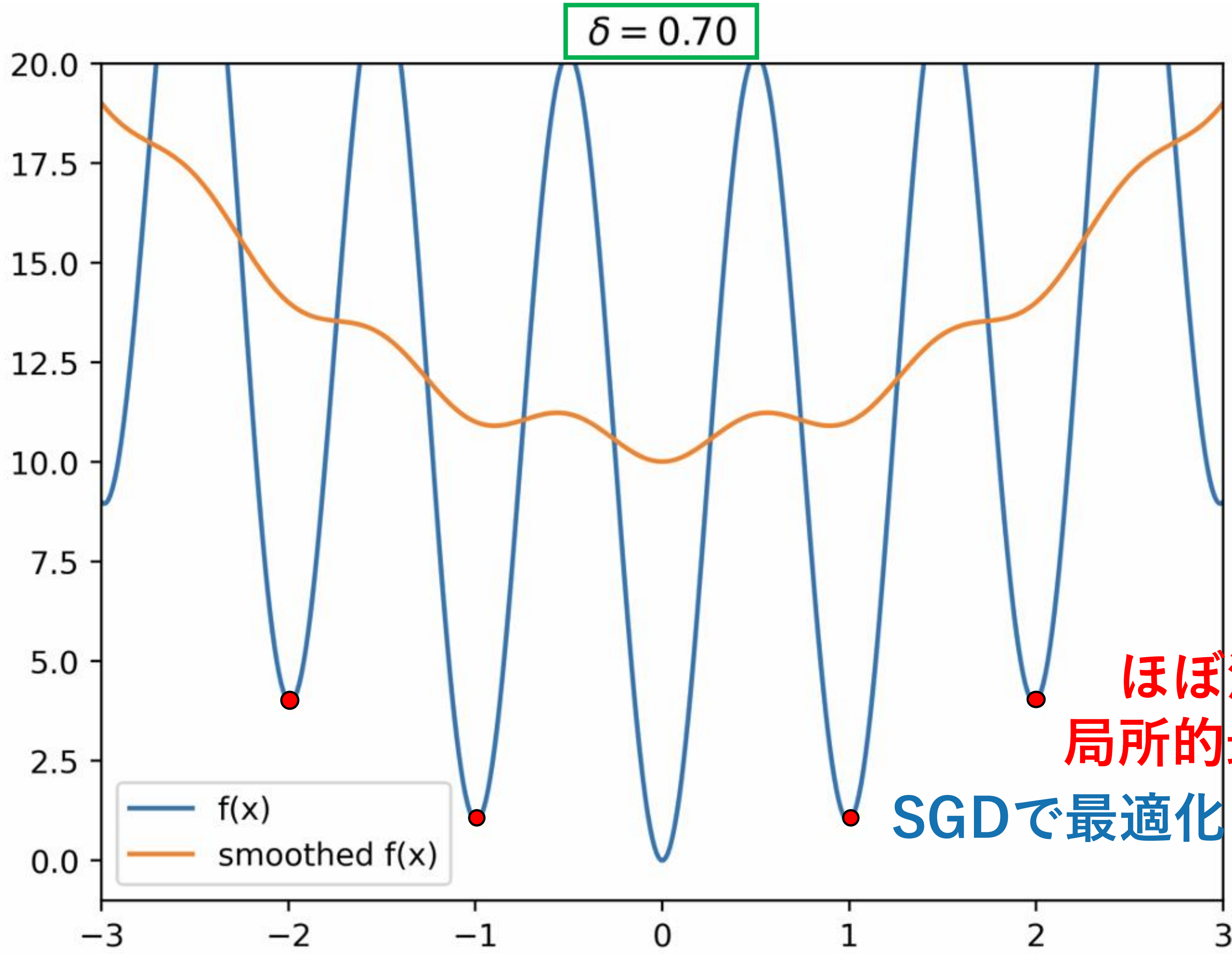
$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\omega}_t}\left[\boldsymbol{y}_{t+1}\right] &= \mathbb{E}_{\boldsymbol{\omega}_t}\left[\boldsymbol{y}_t\right] - \eta\nabla\mathbb{E}_{\boldsymbol{\omega}_t}\left[f(\boldsymbol{y}_t - \underline{\eta\boldsymbol{\omega}_t})\right] \\
&= \mathbb{E}_{\boldsymbol{\omega}_t}\left[\boldsymbol{y}_t\right] - \eta\nabla\mathbb{E}_{\boldsymbol{u}_t\sim\mathcal{L}}\left[f\left(\boldsymbol{y}_t - \underline{\frac{\eta C}{\sqrt{b}}\boldsymbol{u}_t}\right)\right] \\
&= \mathbb{E}_{\boldsymbol{\omega}_t}\left[\boldsymbol{y}_t\right] - \underline{\eta\nabla\hat{f}_{\frac{\eta C}{\sqrt{b}}}(\boldsymbol{y}_t)}.
\end{aligned}$$

大きさ$\eta C/\sqrt{b}$のノイズ
で平滑化された関数

$$\boxed{\delta = \eta\sqrt{\frac{C^2}{b}}}$$

が成り立つ。

SGDの確率的ノイズによる暗黙的な目的関数の平滑化



GDで最適化されるとみなせる目的関数

$$\delta = \eta\sqrt{\frac{C^2}{b}}$$

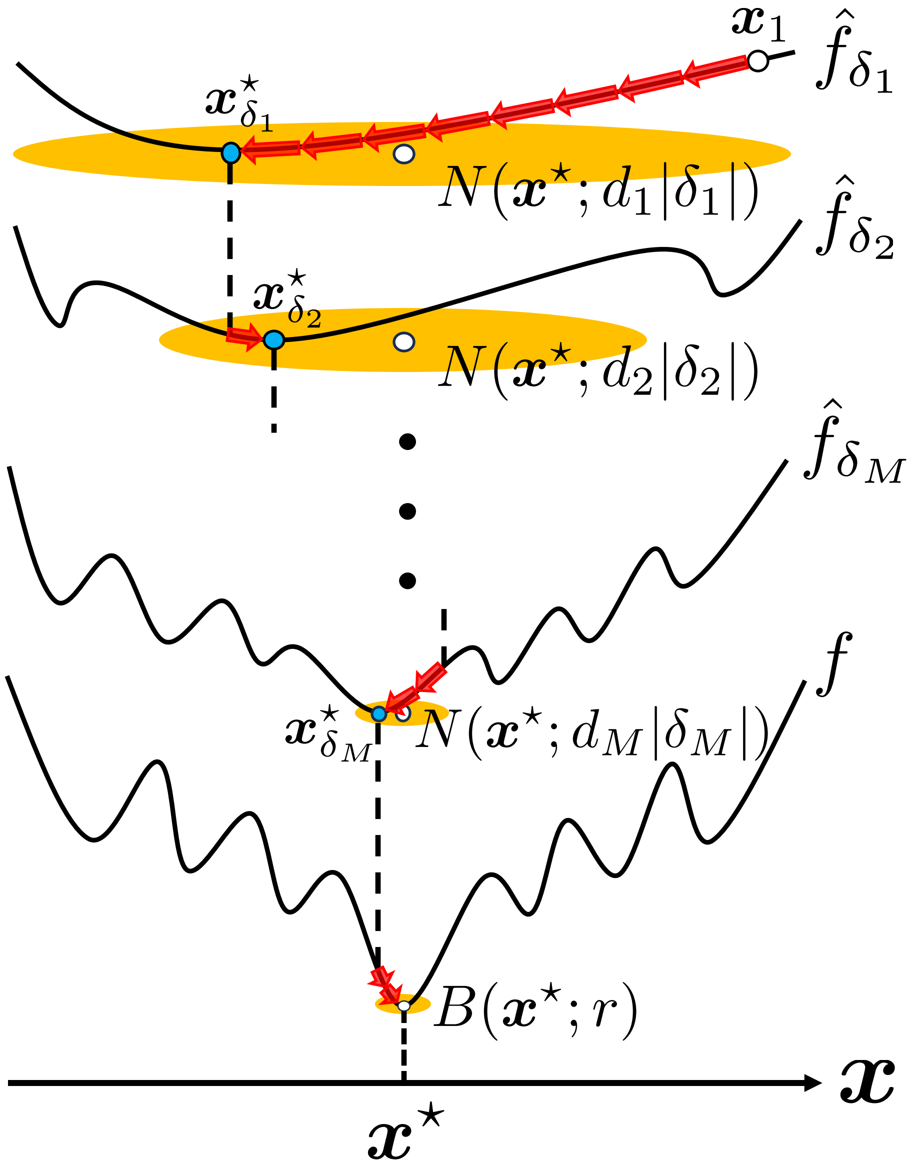
確率的ノイズの大きさ
＝平滑化の度合い

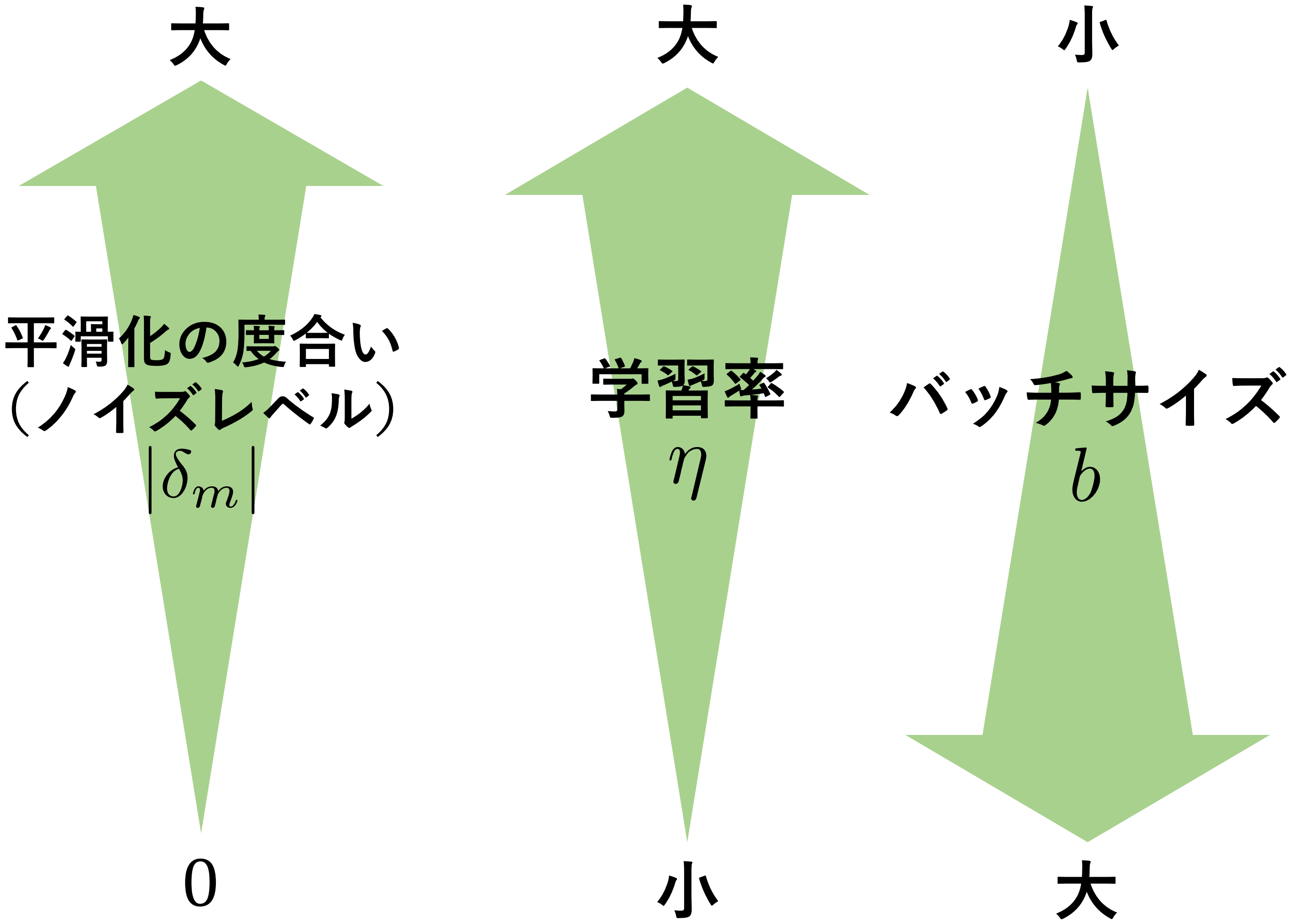
ほぼ消滅した局所的最適解たち

SGDで最適化される目的関数

SGDの平滑化特性

$$\delta = \eta\sqrt{\frac{C^2}{b}}$$





暗黙的な段階的最適化アルゴリズム

Algorithm 1 Implicit Graduated Optimization

Require: $\epsilon, \boldsymbol{x}_1 \in B(\boldsymbol{x}^\star_{\delta_1}; 3\delta_1), \eta_1 > 0, b_1 \in [n], \gamma \geq 0.5$

$\delta_1 := \frac{\eta_1 C}{\sqrt{b_1}}, \alpha_0 := \min\left\{\frac{1}{16 L_f \delta_1}, \frac{1}{\sqrt{2\sigma\delta_1}}\right\}, M := \log_\gamma \alpha_0 \epsilon$

for $m = 1$ to $M+1$ **do**

  if $m \neq M+1$ **then**

    $\epsilon_m := \sigma_m \delta_m^2 / 2,\ T_m := H_m / \epsilon_m$

    $\kappa_m / \sqrt{\lambda_m} = \gamma\ (\kappa_m \in (0, 1], \lambda_m \geq 1)$

  end if

  $\boldsymbol{x}_{m+1} := \mathrm{GD}(T_m, \boldsymbol{x}_m, \hat{f}_{\delta_m}, \eta_m)$

  $\eta_{m+1} := \kappa_m \eta_m, b_{m+1} := \lambda_m b_m$

  $\delta_{m+1} := \frac{\eta_{m+1} C}{\sqrt{b_{m+1}}}$

end for

return $\boldsymbol{x}_{M+2}$

定理3.4

関数$f \colon \mathbb{R}^d \to \mathbb{R}$が$\sigma$-nice 関数だとすると、アルゴリズム3は、$\mathcal{O}\left(1/\epsilon^2\right)$回の反復で、関数$f$の**大域的最適解**$\boldsymbol{x}^\star$の$\epsilon$-近傍に到達する。

Algorithm 2 Gradient Descent

Require: $T_m, \hat{\boldsymbol{x}}_1^{(m)}, \hat{f}_{\delta_m}, \eta > 0$

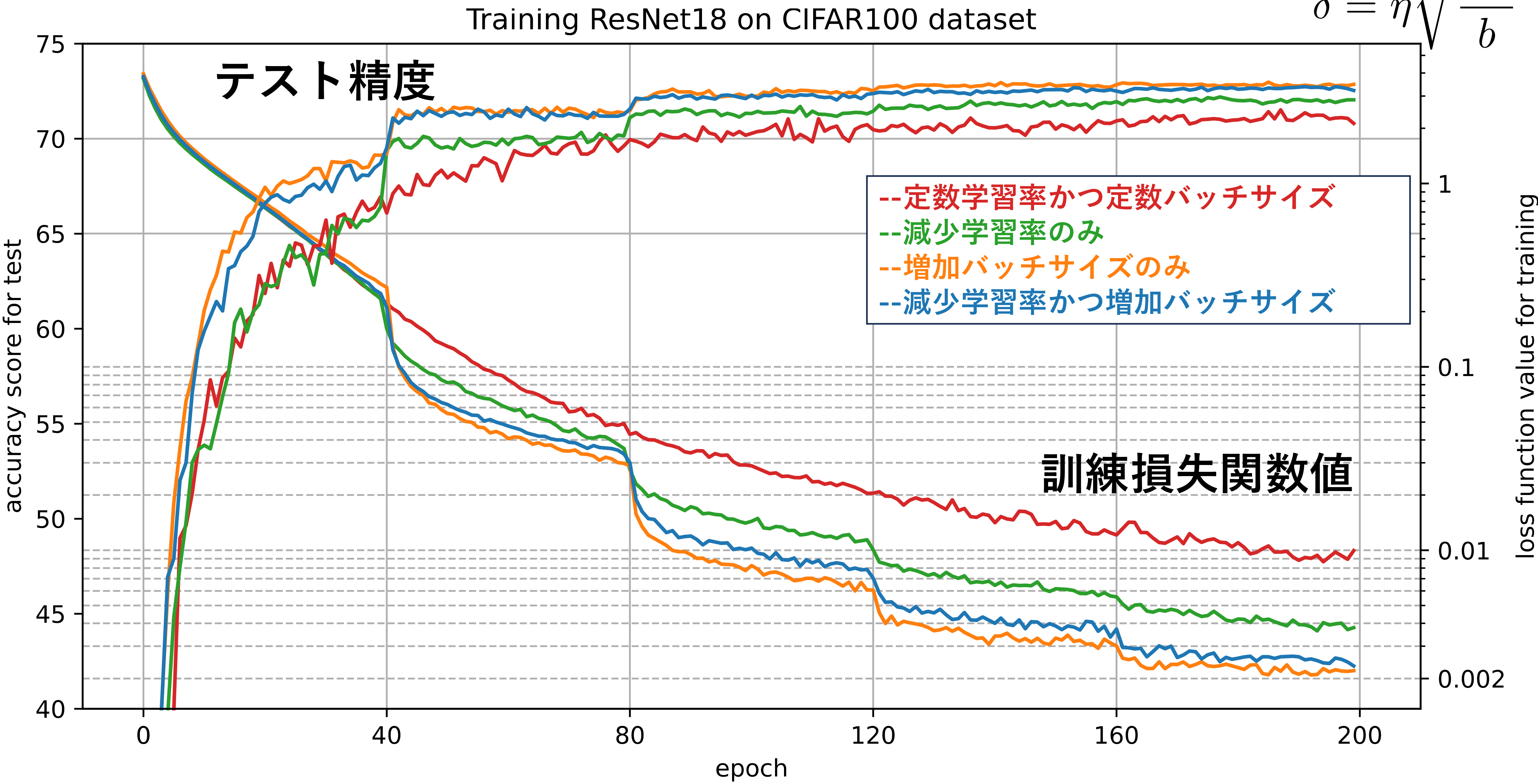
  for $t = 1$ to T_m **do**

    $\hat{\boldsymbol{x}}_{t+1}^{(m)} := \hat{\boldsymbol{x}}_t^{(m)} - \eta \nabla \hat{f}_{\delta_m}(\boldsymbol{x}_t)$

  end for

  return $\hat{\boldsymbol{x}}_{T_m+1}^{(m)}$

$$\delta = \eta\sqrt{\frac{C^2}{b}}$$



結論

- ▷SGDには関数を平滑化する効果があり、その度合いは学習率とバッチサイズによって決まる。 $\delta = \eta\sqrt{\frac{C^2}{b}}$
- ▷この性質を利用した暗黙的な段階的最適化アルゴリズムアルゴリズムを提案し、このアルゴリズムが $\mathcal{O}\left(1/\epsilon^2\right)$ 回の反復で大域的最適解 $x^\star$ の ϵ -近傍に到達することを示した。
- ▷深層学習モデルの訓練において，明示的な段階的最適化アルゴリズムは機能しないが，暗黙的な段階的最適化アルゴリズムは機能する.
- ▷したがって、計算資源が十分にあれば、シンプルなSGDでもDNNを含む非凸関数の大域的最適化が可能となる。

付録:よく使われる経験損失関数はσ-nice関数

▷ $\boldsymbol{x}_i \in \mathbb{R}^d (i \in [n])$：i番目の訓練データ

▷ $\boldsymbol{y}_i \in \mathbb{R}^c$：i番目のラベル(one-hotベクトル)

▷ $f(\boldsymbol{x}_i)$：i番目の訓練データに対するモデルの出力

▷ $f(\boldsymbol{x}_i)^{(j)} \in \mathbb{R} \ (j \in [c])$：モデルの出力のj番目の要素

▷ $y_i^{\mathrm{hot}} \in \mathbb{R}$：ラベル $\boldsymbol{y}_i \in \mathbb{R}^c$の要素1を持つindex

▷分類タスクでよく使われる交差エントロピー誤差は，結局

$$L := \frac{1}{n} \sum_{i \in [n]} L_i, \text{where } L_i := -\log f(\boldsymbol{x}_i)^{(y_i^{\mathrm{hot}})}$$

で表せるので，$g(x) := -\log x \ (0 < x \leq 1)$を考慮すればよい．

▷ $g(x)$も$\hat{g}_\delta(x)$も，2階微分が下から1で抑えられるので，1-強凸.

▷ $g(x)$も$\hat{g}_\delta(x)$も，x=1が大域的最適解．よって1-nice関数.

付録：SGDの挙動についての理論的洞察

▷なぜ大きいバッチサイズは汎化性を悪化させるのか。

⇒バッチサイズが大きすぎると、最適化される関数は元の非凸関数に近付き、悪い局所的最適解に陥りやすくなるため。

▷なぜ減少学習率または増加バッチサイズは定数よりも優れるのか。

⇒訓練中に学習率を減少させる、あるいはバッチサイズを増加させることは、ノイズレベルを小さくすることと等価。

⇒減少学習率または増加バッチサイズを使うことは、まさに段階的最適化のアプローチになっているため、定数よりも優れる。

付録：機械学習における汎化能力

▷学習が完了したモデルが、訓練データ以外の入力に対しても正しく分類できることを『**汎化性**に優れる』という。

訓練に**使われた**データ

訓練に**使われていない**データ

関数 $g: \mathbb{R}^d \to \mathbb{R}$
変数: $\boldsymbol{x} \in \mathbb{R}^D$,

$$z_i \in \mathbb{R}^d$$

Deep Neural Network

訓練**済み**
＝パラメータ**調整済**

猫

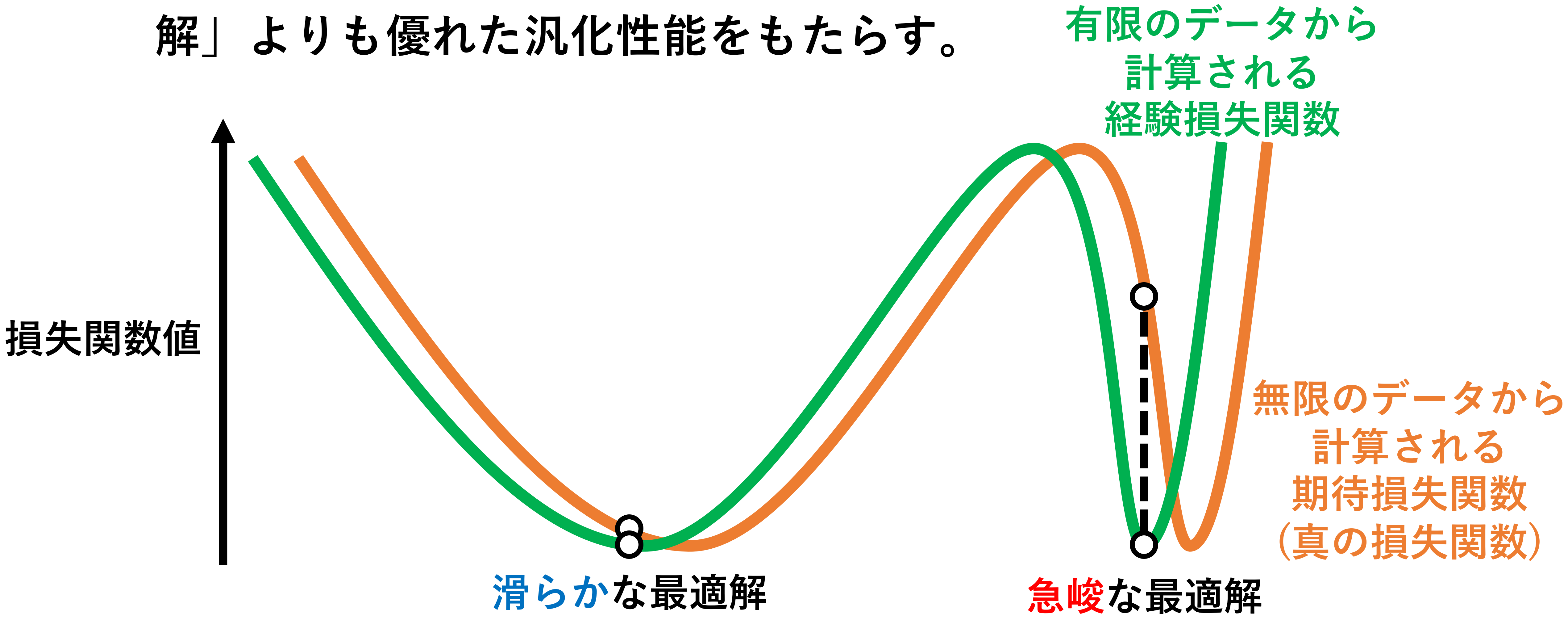
訓練損失: 0.001
訓練精度: 99.9%

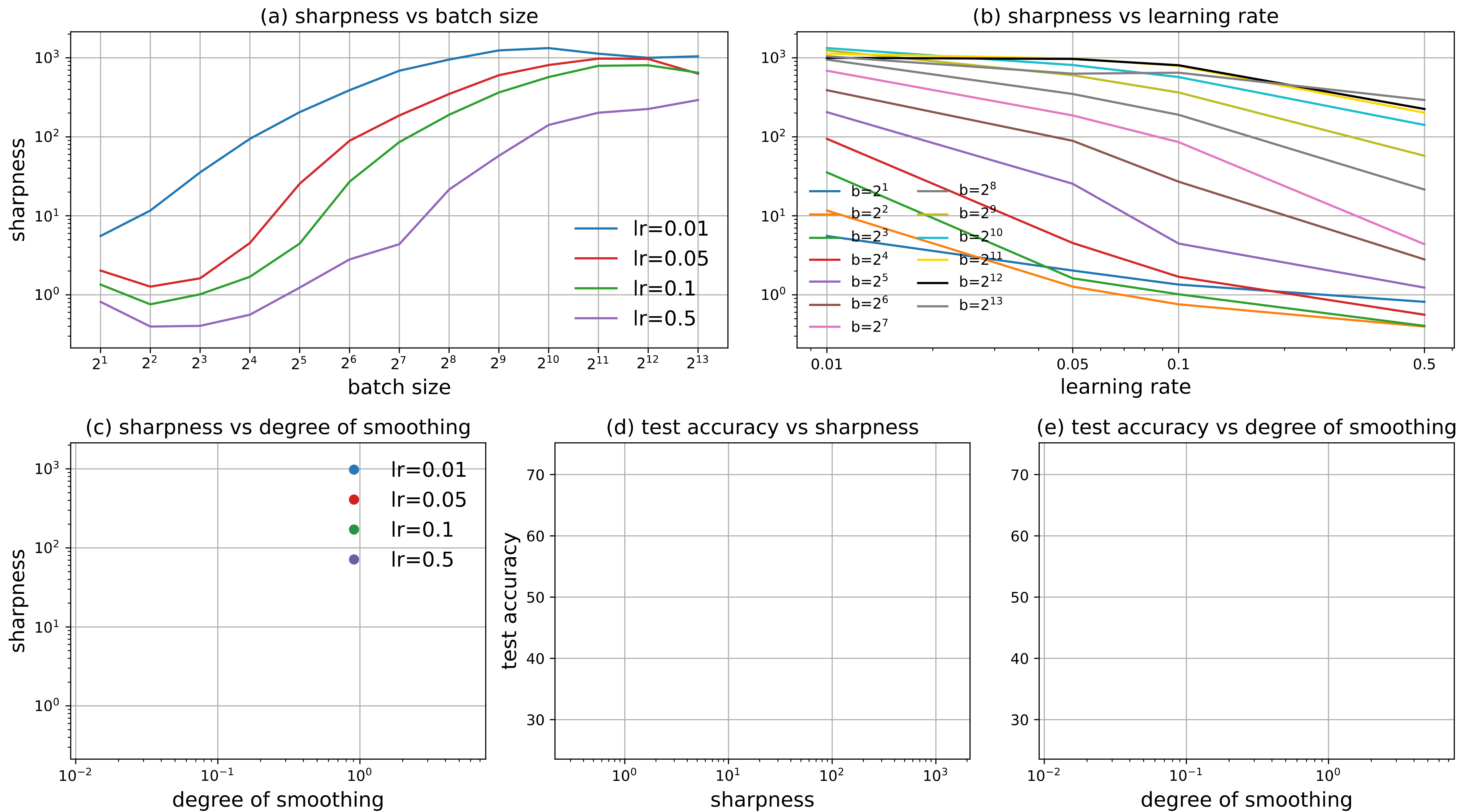
これも猫！

テスト精度: 75%

付録：汎化性能と経験損失関数の形状

▷「その近傍が滑らかな最適解」は、「その近傍が急峻な最適解」よりも優れた汎化性能をもたらす。

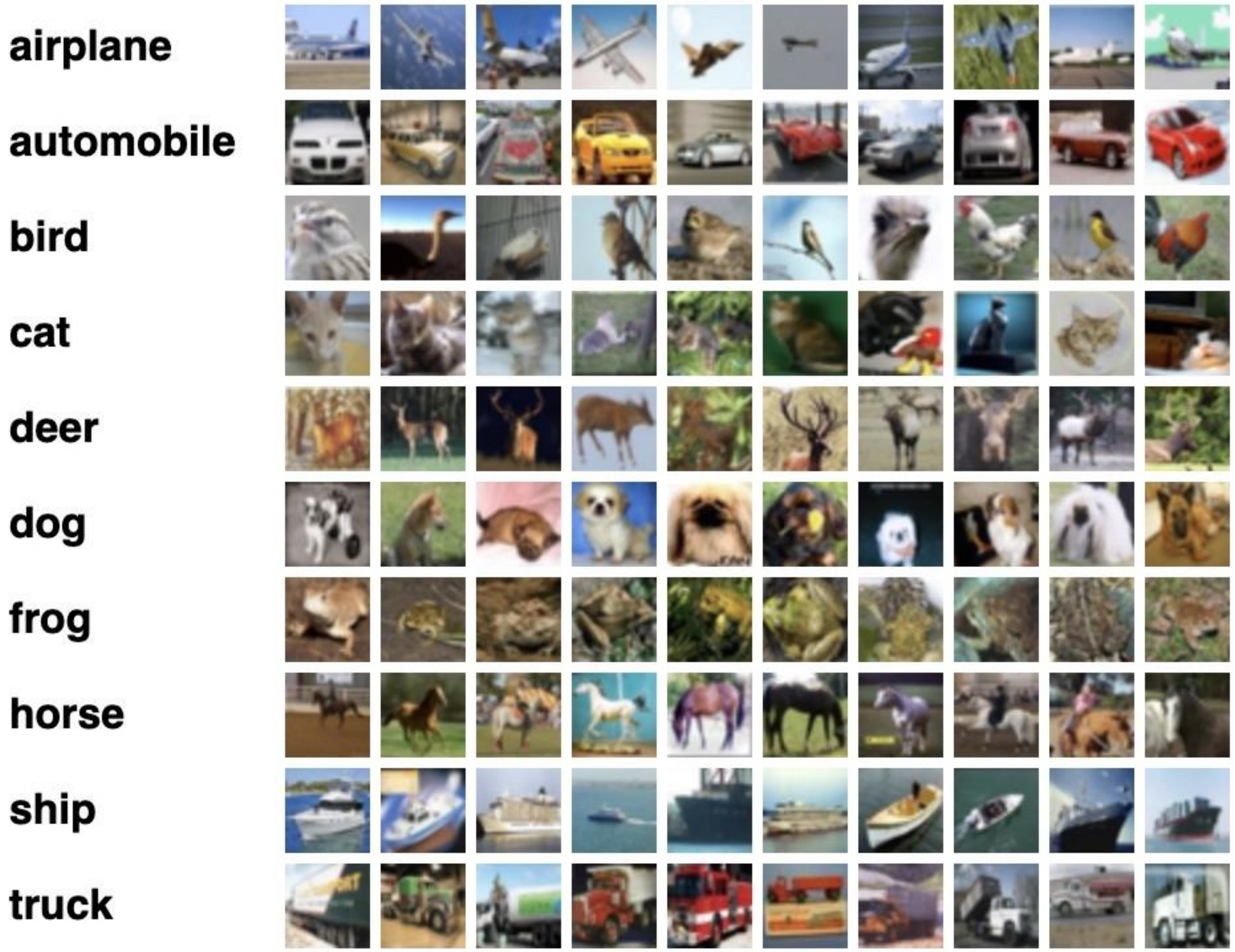




(a) sharpness vs batch size
sharpness
batch size
lr=0.01
lr=0.05
lr=0.1
lr=0.5
(b) sharpness vs learning rate
learning rate
b=2^1
b=2^2
b=2^3
b=2^4
b=2^5
b=2^6
b=2^7
b=2^8
b=2^9
b=2^10
b=2^11
b=2^12
b=2^13
(c) sharpness vs degree of smoothing
sharpness
degree of smoothing
lr=0.01
lr=0.05
lr=0.1
lr=0.5
(d) test accuracy vs sharpness
test accuracy
sharpness
(e) test accuracy vs degree of smoothing
degree of smoothing

数値実験

- ▷モデル: ResNet18
- ▷データセット: CIFAR100
  - ▷100クラスの画像分類
  - ▷5万枚の訓練データ
  - ▷1万枚のテストデータ
  - ▷画像サイズは32×32×3
- ▷交差エントロピー誤差
- ▷NVIDIA GeForce RTX4090



入力

z

ある巨大な関数

ディープニューラルネットワーク

g

出力 $g(x, z)$:犬

正解 y:猫

誤差 $f(x) := \|g(x, z) - y\|$

誤差の関数$f(x)$

を最適化(最小化)する！