

勾配法を用いたハイパーパラメータ最適化

理研AIP

幡谷龍一郎

hataya.tokyo

目次

ハイパーパラメータ最適化

微分可能プログラミング

勾配法によるハイパーパラメータ最適化

ハイパーパラメータ最適化

$$\phi^* \in \operatorname{argmin}_{\phi} \gamma(\theta^*(\phi)) \quad \theta^*(\phi^*) \text{ がほしい}$$

↑
評価指標

$$\text{s.t. } \theta^*(\phi) \in \operatorname{argmin}_{\theta} \ell_{\text{train}}(\theta, \phi; \mathcal{D}_{\text{train}})$$

↑
訓練損失

↑
モデルパラメータ

↑
ハイパーパラメータ

argmin : 最適化アルゴリズムを
与えられた資源で実行した、
という気持ち

ハイパーパラメータの例

- SGDの学習率
- 訓練損失の選択・重み付け
- どのモデルを使うか

評価指標 : 情報量規準 (AIC, BIC) , クロスバリデーション

以下では評価データ上での評価損失 $\gamma(\theta) = \ell_{\text{val}}(\theta; \mathcal{D}_{\text{val}})$ を用いる

ハイパーパラメータ最適化

θ が深層モデルの場合，局所解に達することも難しいので

$$\phi^* \in \underset{\phi}{\operatorname{argmin}} \ell_{\text{val}}(\theta_T(\phi); \mathcal{D}_{\text{val}})$$

$$\text{s.t. } \theta_t = \mathbf{A}(\theta_{t-1}, \phi; \mathcal{D}_{\text{train}})$$



最適化アルゴリズム

$$\text{勾配降下法： } \theta_{t-1} - \phi_0 \nabla_{\theta} \ell_{\text{train}}(\theta_{t-1}; \mathcal{D}_{\text{train}})$$

どのように ϕ を最適化するか？

$\nabla_{\phi} \ell_{\text{val}}(\theta^*(\phi))$ を使う → 勾配法（今回のトピック）

使わない → ブラックボックス最適化

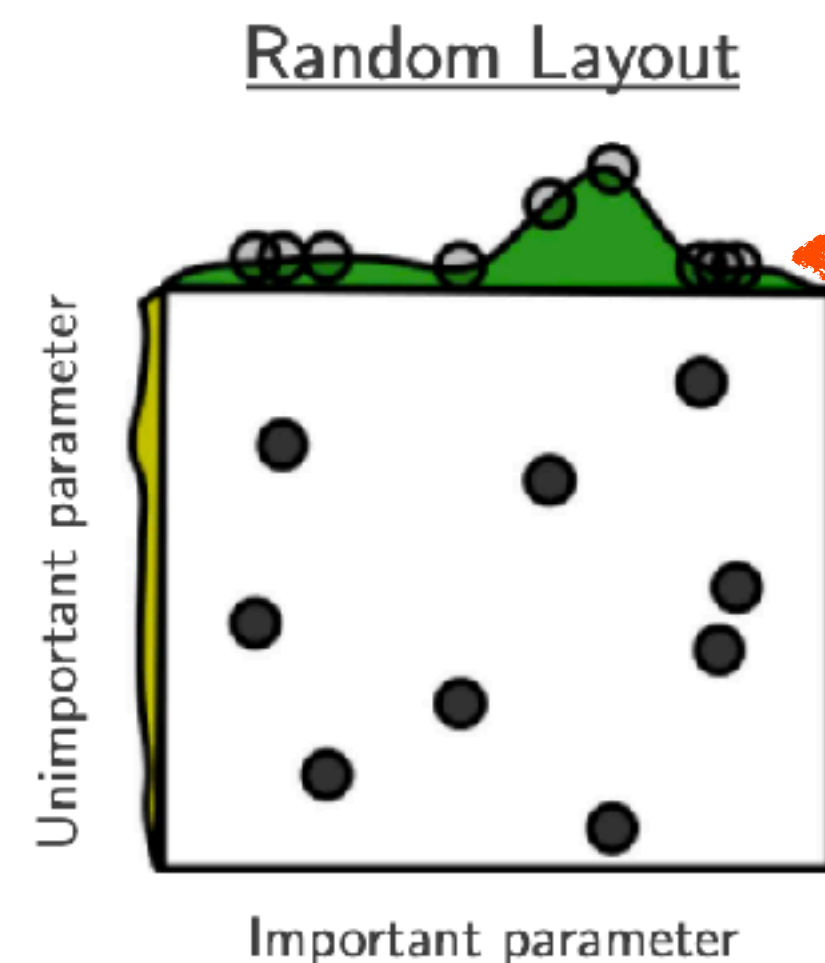
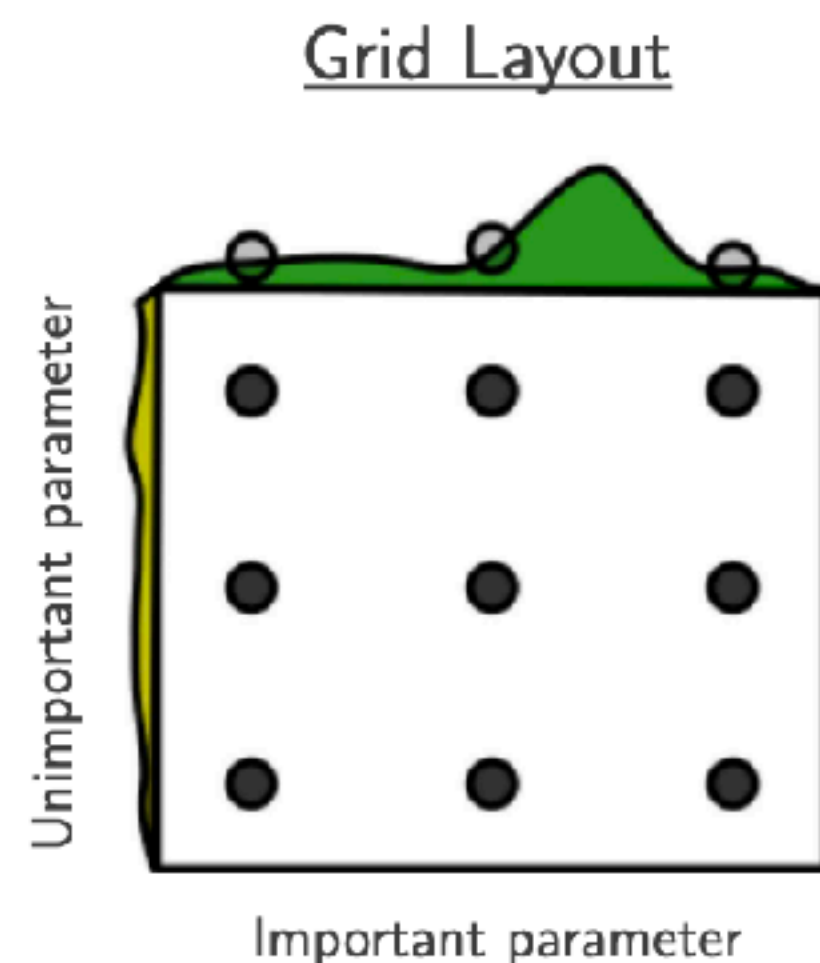
ブラックボックス最適化

- グリッド探索 (しらみつぶし)

```
for lr in {0.05, 0.1, 0.5}:  
  for loss in {"cce", "mse"}:  
    ...
```

連続な場合は適当に離散化
→すべて試す

- ランダム探索 [Bergstra&Bengio 12]

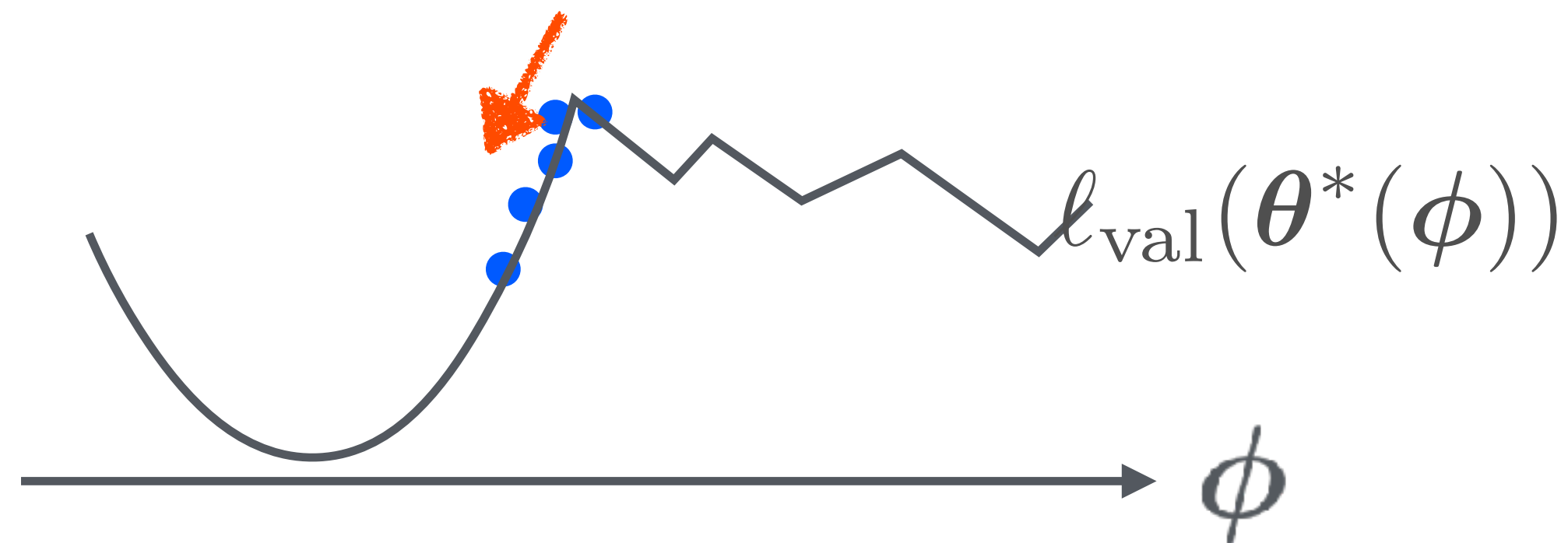


グリッド探索では探索できない領域も
探索できる
一様分布列などを用いても良い

[Bergstra&Bengio 12]

ブラックボックス最適化

- 集団を用いるアプローチ [Hansen&Ostermeier 96, Williams 92, Wiersta+ 08]

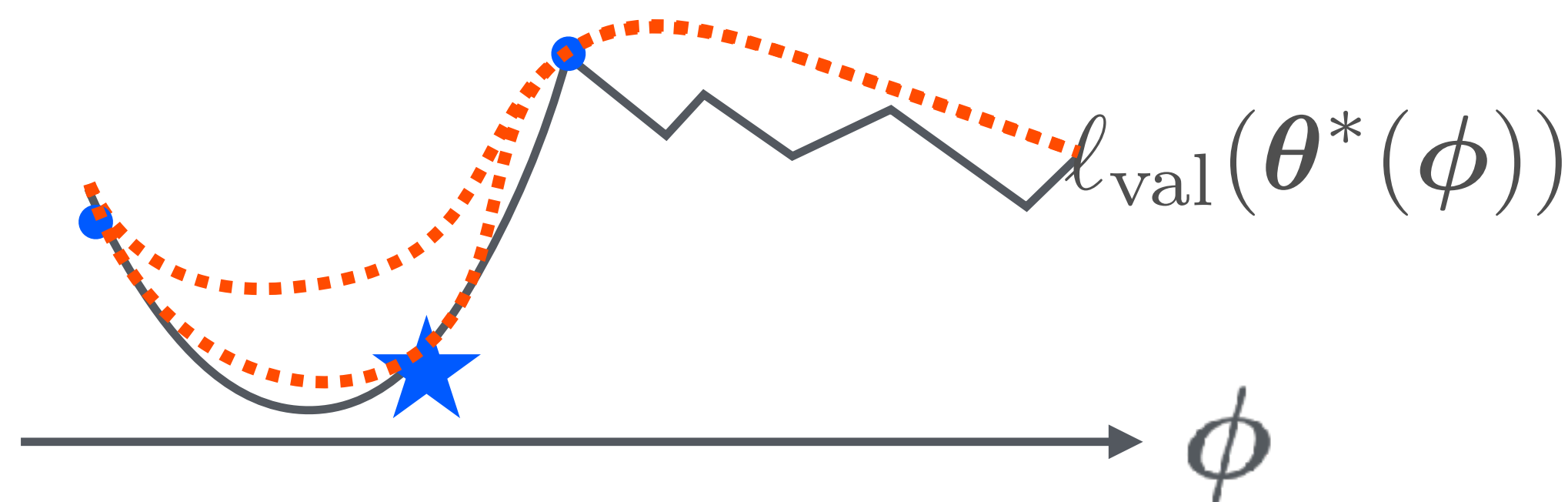


進化計算・CMA-ES・NES・方策勾配法など

$$\phi_1^{(t+1)}, \dots, \phi_N^{(t+1)} \sim \mathcal{N}(\mu^{(t)}, C^{(t)})$$

よい ϕ をもとに更新

- 代理モデルによるアプローチ [Williams&Rasmussen 06, Bergstra+ 11]



ベイズ最適化

ガウス過程・TPEなどでモデル化

獲得関数を最大にするような ϕ を提案

ϕ を評価しモデルを更新

グレーボックス最適化

深層学習などではモデルの評価にコストがかかる

$$\phi^* \in \operatorname{argmin}_{\phi} \gamma(\theta^*(\phi))$$

$$\text{s.t. } \theta^*(\phi) \in \operatorname{argmin}_{\theta} \ell_{\text{train}}(\theta, \phi; \mathcal{D}_{\text{train}})$$

深層モデルの学習

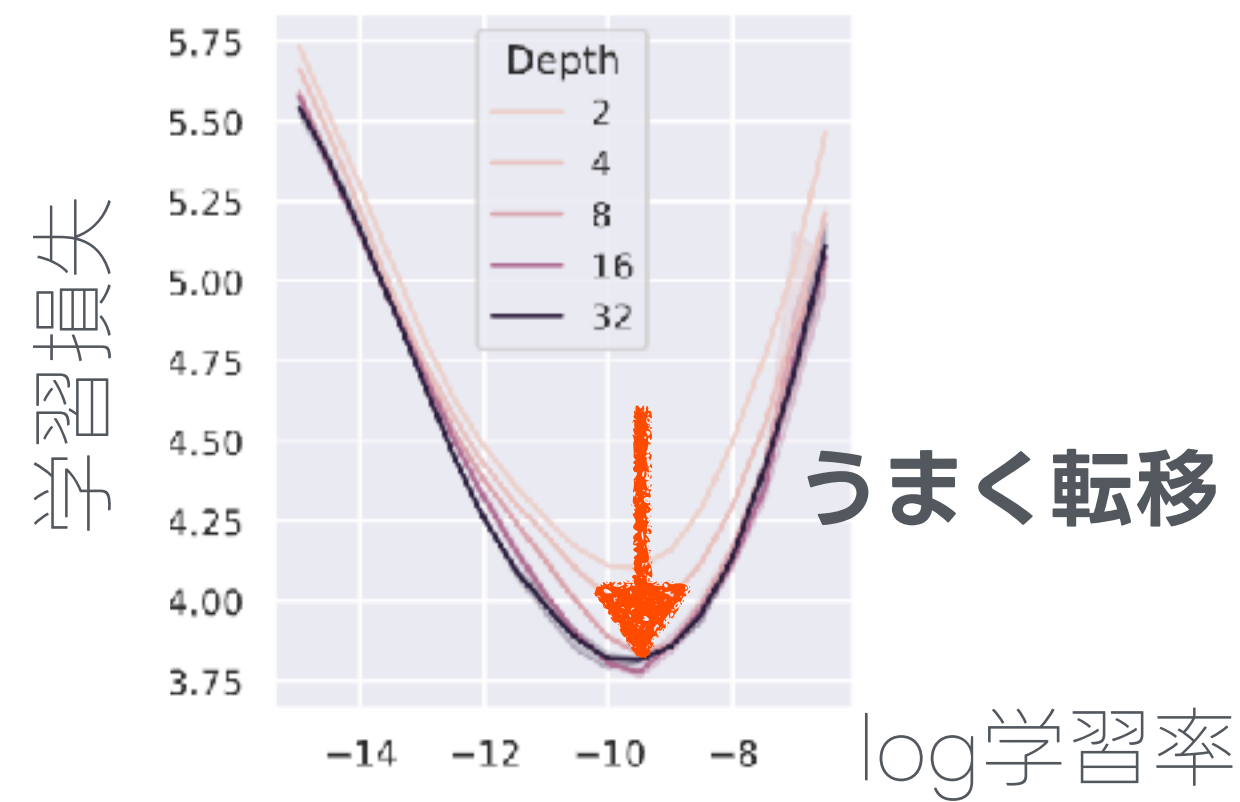
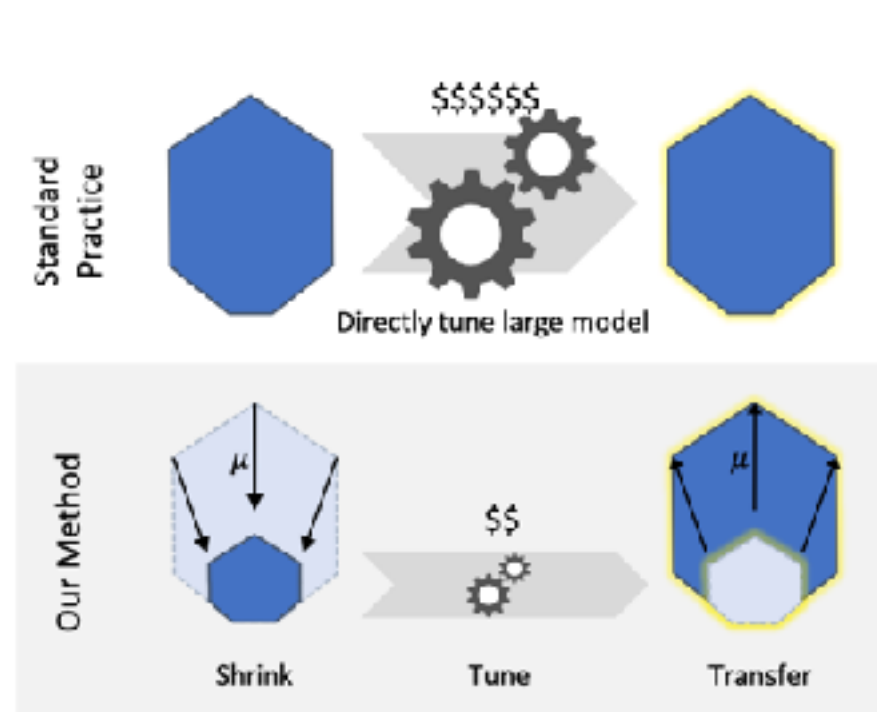
事前知識を用いて評価コストを下げられる場合がある

軽量なモデル ↔ 巨大モデル

で有効なハイパーパラメータが転移

学習データのサブセット ↔ 全データ上

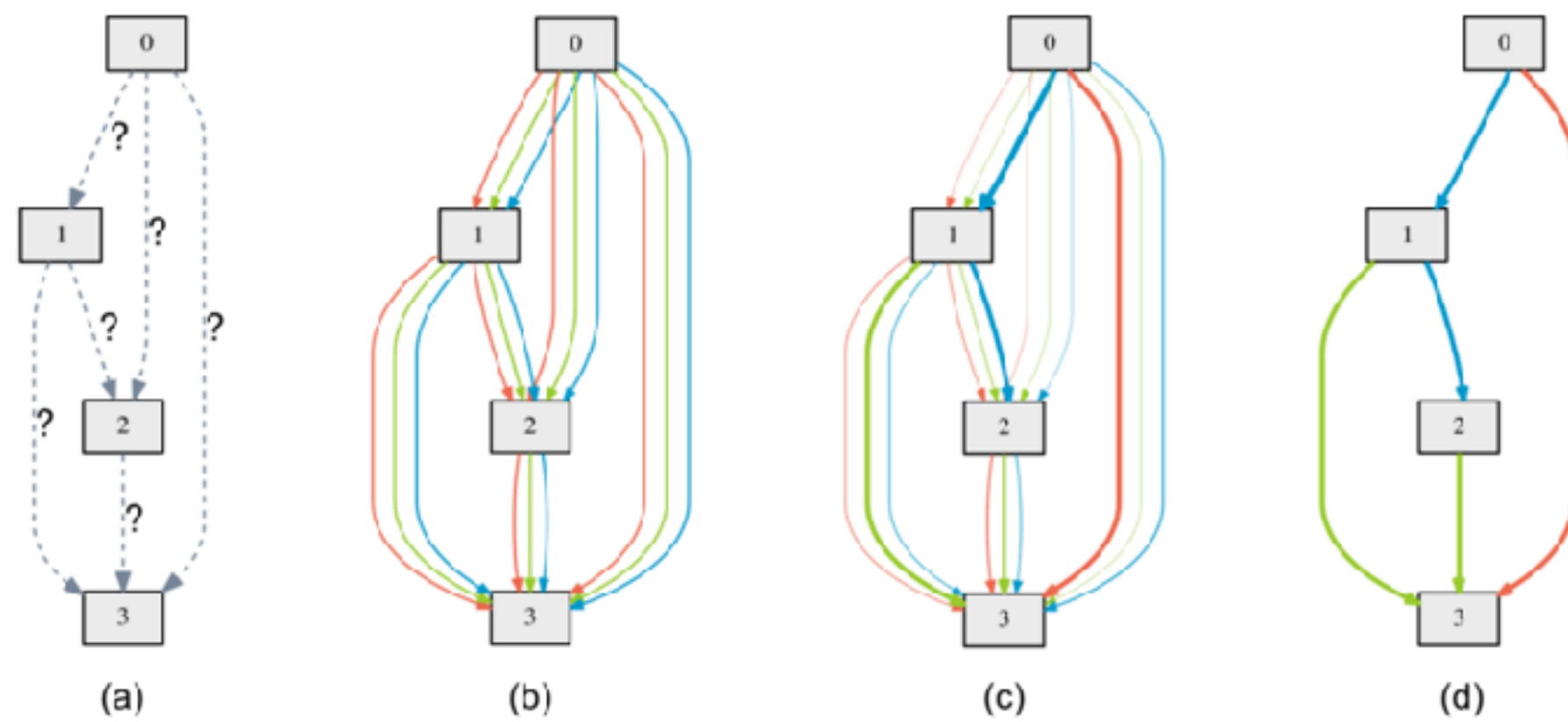
c.f. 性能が転移する軽量なモデルを設計する [Yang+22] GPTの学習にも使われた…?



勾配法によるハイパーパラメータ最適化の利点

勾配法はブラックボックス最適化よりも圧倒的に速い
ハイパーパラメータが増えても使える

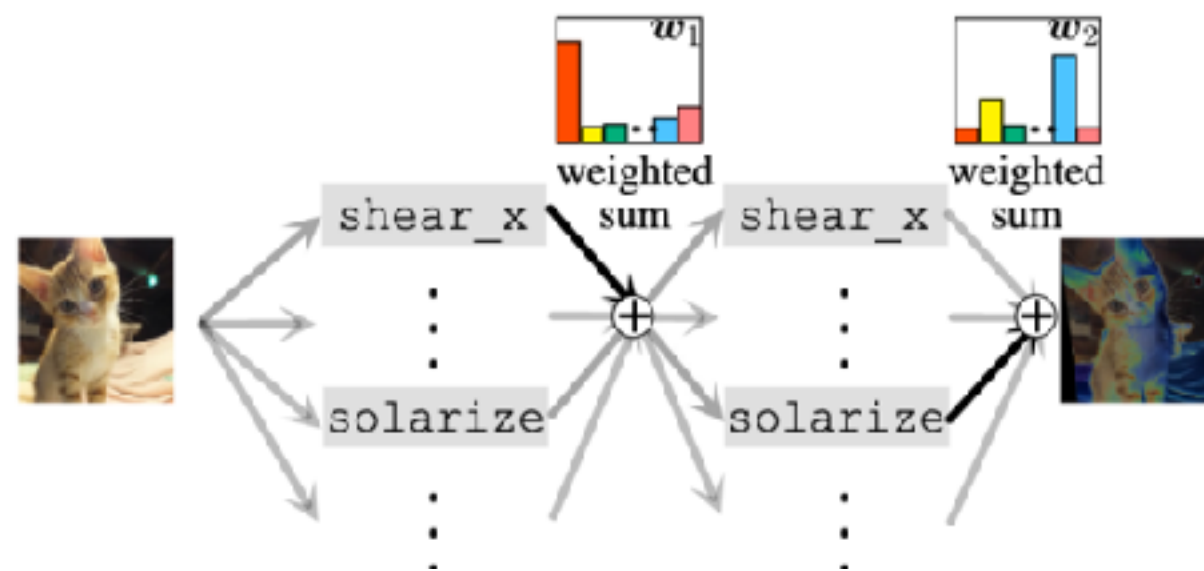
ニューラルネットワークの構造探索



200~2k GPU days → **4 GPU days**

[Liu+19]

データ拡張方策の最適化



450~15k GPU hours → **2.3 GPU hours**

[Hataya+20, 22]

目次

ハイパーパラメータ最適化

微分可能プログラミング

勾配法によるハイパーパラメータ最適化

微分可能プログラミング

$\nabla_{\phi} \ell_{\text{val}}(\theta^*(\phi))$ が得られれば勾配法でハイパーパラメータを最適化可能

微分可能プログラミング：プログラムを微分可能な構成要素を使って組立て、自動微分で勾配を計算する (ある意味深層学習の一般化)



Yann LeCun

January 6, 2018 · 🌐

OK, Deep Learning has outlived its usefulness as a buzz-phrase.
Deep Learning est mort. Vive Differentiable Programming!

連続なパラメータ+微分可能な関数：自動微分が素直に使える

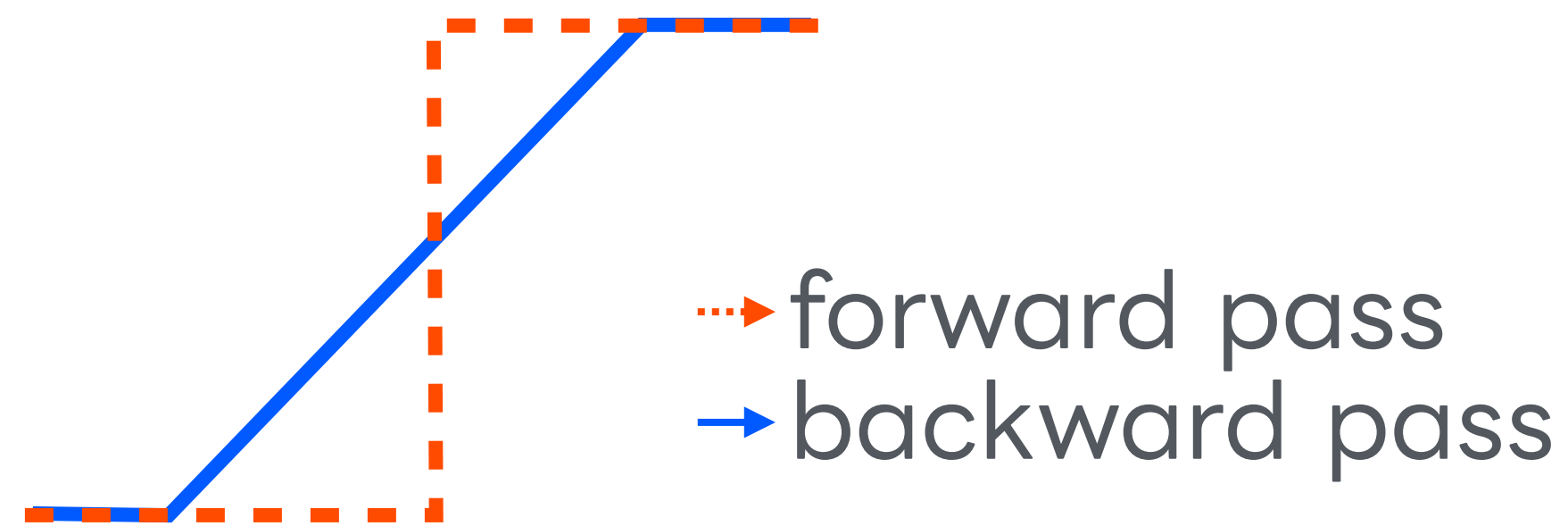
それ以外の場合：

離散なパラメータ→ 連続なパラメータで表現

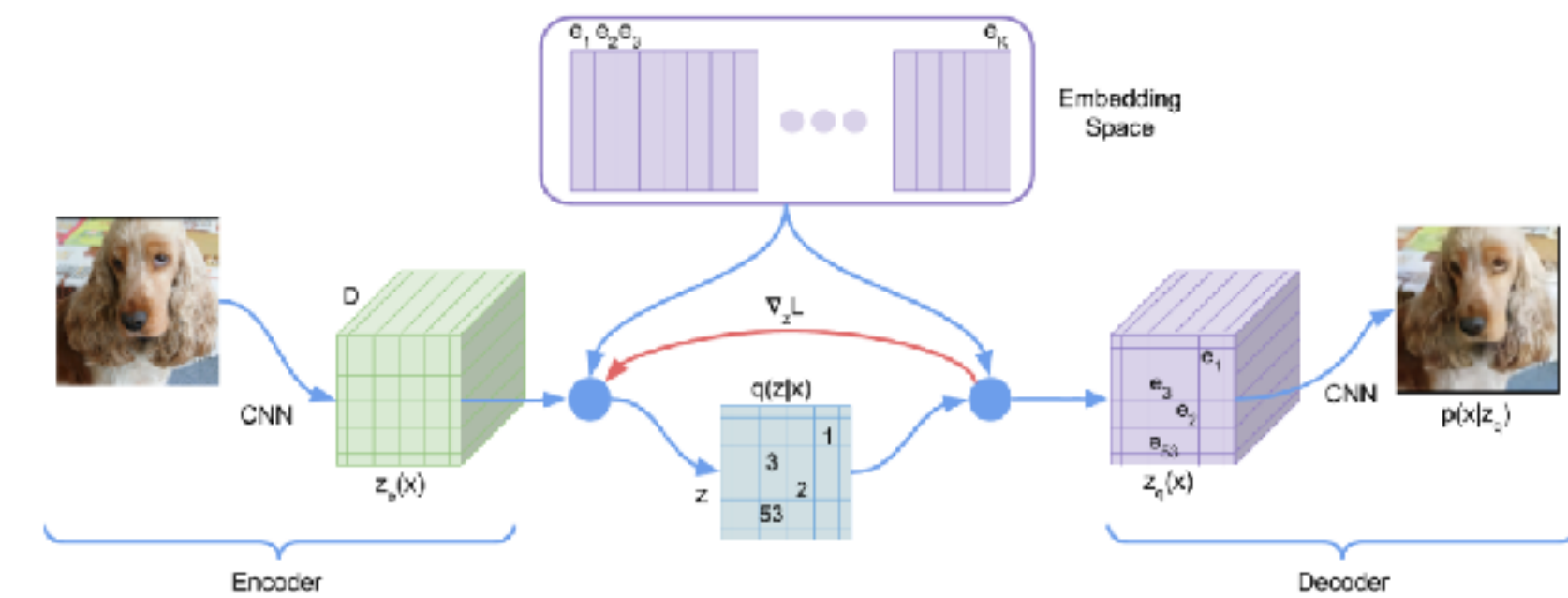
微分不可能な関数→ 勾配を推定・近似する

離散な操作の勾配の推定

- Straight-through Estimator [Bengio+13, Liu+23]



微分可能な最近傍法



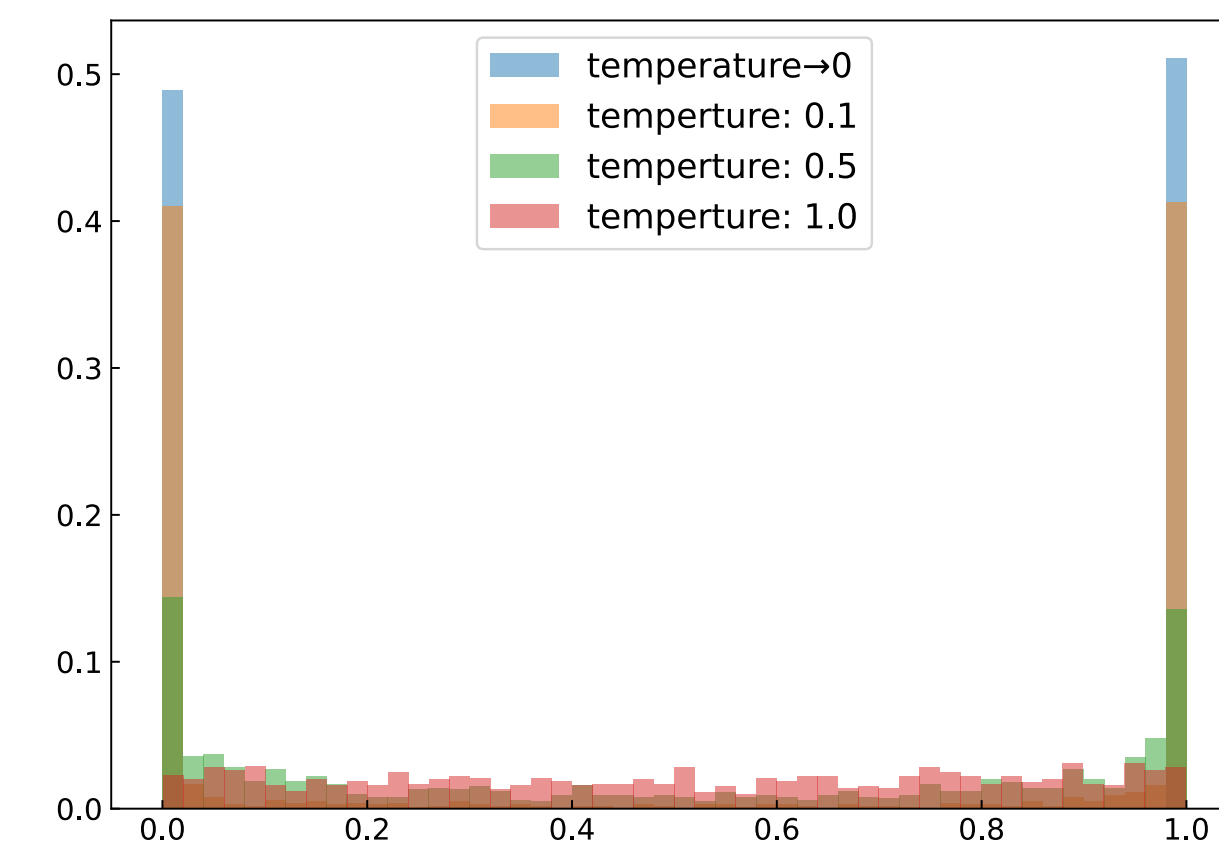
[van der Oord+17]

- Gumbel Softmax [Maddison+17, Jang+17]

微分可能なカテゴリカル分布・ベルヌーイ分布のための再パラメータ化

$$\text{softmax}_i \tau (\gamma_i + \log \pi_i)$$

$$\gamma_i \sim \text{Gumbel}(0, 1)$$



ベルヌーイ分布 (0.5)

離散な操作の勾配の推定

- ・ ソート・ランキング

$$\mathbf{x} = (0.38, 4, -2, 6, -9)$$

$$\boldsymbol{\sigma} = (5, 3, 1, 2, 4) : \text{argsort}$$


ソート

$$S(\mathbf{x}) = \mathbf{x}_{\boldsymbol{\sigma}} = (-9, -2, 0.38, 4, 6)$$

ランキング

$$R(\mathbf{x}) = \boldsymbol{\sigma}^{-1} = (3, 4, 2, 5, 1)$$

たとえば

$$R(\mathbf{x}) = \mathbf{P} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 2 \\ 5 \\ 1 \end{pmatrix}$$


最適輸送によって置換行列を推定 [Cuturi+19]

$$R(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{P}} \langle \mathbf{y}, \mathbf{x} \rangle$$

置換群の要素全体

アイソトニック回帰に帰着され

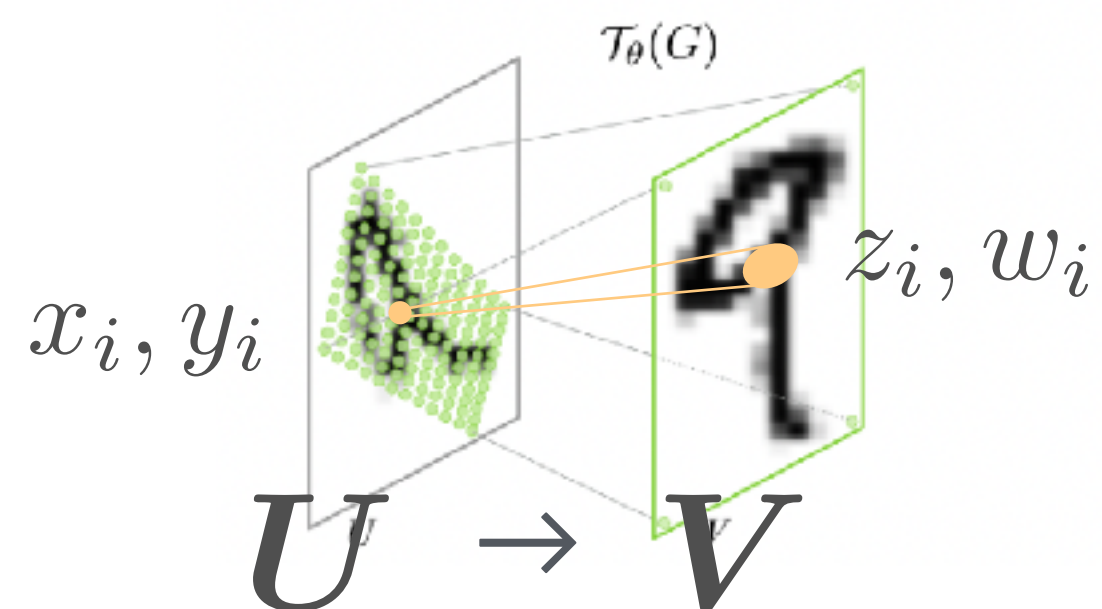
$O(n \log n)$ で実現可能

[Blonderl+20]

ピクセルの操作

- Spatial Transformer [Jaderberg+15]

画像のクロップ，拡大縮小，回転を微分可能な形に



$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \end{bmatrix} \begin{pmatrix} z_i \\ w_i \\ 1 \end{pmatrix}$$

i 番目のピクセルの値は

$$V_i = \sum_{h=1}^H \sum_{w=1}^W U_{h,w} k(x_i - w) k(y_i - h)$$

k : カーネル関数 (バイリニア補完など)

目次

ハイパーパラメータ最適化

微分可能プログラミング

勾配法によるハイパーパラメータ最適化

勾配法によるハイパーパラメータ最適化

$$\phi^* = \operatorname{argmin}_{\phi} \ell_{\text{val}}(\theta_T(\phi); \mathcal{D}_{\text{val}})$$

$$\text{such that } \theta_t = \theta_{t-1} - \alpha \frac{\partial}{\partial \theta} \ell_{\text{train}}(\theta_{t-1}, \phi; \mathcal{D}_{\text{train}})$$

Q. どのようにして $\frac{d\ell_{\text{val}}}{d\phi}$ を求めるか?
超勾配 (hypergradient)

A1. 内側の最適化問題に対して誤差逆伝播する [Finn+17, Franceschi+17など]
空間複雑性が $T \dim \theta$ に比例する

A2. 陰関数勾配を利用する

$$\frac{\partial \ell_{\text{train}}}{\partial \theta} \approx 0 \quad \text{のとき} \quad \frac{d\ell_{\text{val}}}{d\phi} = - \frac{\partial \ell_{\text{val}}}{\partial \theta} \left(\frac{\partial^2 \ell_{\text{train}}}{\partial \theta^2} \right)^{-1} \frac{\partial^2 \ell_{\text{train}}}{\partial \theta \partial \phi}$$

成立しているものと仮定

ヘッセ行列 \mathbf{H} 実際には $\mathbf{H} + \rho \mathbf{I}$ ($\rho > 0$) として正定値に

近似により空間複雑性は $\dim \theta$ 程度 \rightarrow 深層モデルでも利用しやすい

ヘッセ逆行列の近似

A2. 陰関数勾配を利用する

$$\frac{\partial \ell_{\text{train}}}{\partial \theta} \approx \mathbf{0} \quad \text{のとき} \quad \frac{d\ell_{\text{val}}}{d\phi} = -\frac{\partial \ell_{\text{val}}}{\partial \theta} \left(\frac{\partial^2 \ell_{\text{train}}}{\partial \theta^2} \right)^{-1} \frac{\partial^2 \ell_{\text{train}}}{\partial \theta \partial \phi}$$

深層モデルのヘッセ行列にはベクトル積の形で高速にアクセス可能

$$\mathbf{H}\mathbf{v} = \frac{\partial^2 \ell_{\text{train}}}{\partial \theta^2} \mathbf{v} = \frac{\partial}{\partial \theta} \left(\mathbf{v}^\top \frac{\partial \ell_{\text{train}}}{\partial \theta} \right) \quad : \text{Hessian-vector product (HVP)}$$

→ HVPを利用して線形コストでヘッセ逆行列ベクトル積を近似可能

線形方程式 $\mathbf{H}\mathbf{x} = \mathbf{v}$ の逐次解法 ($\mathbf{x} = \mathbf{H}^{-1}\mathbf{v}$ を求める)

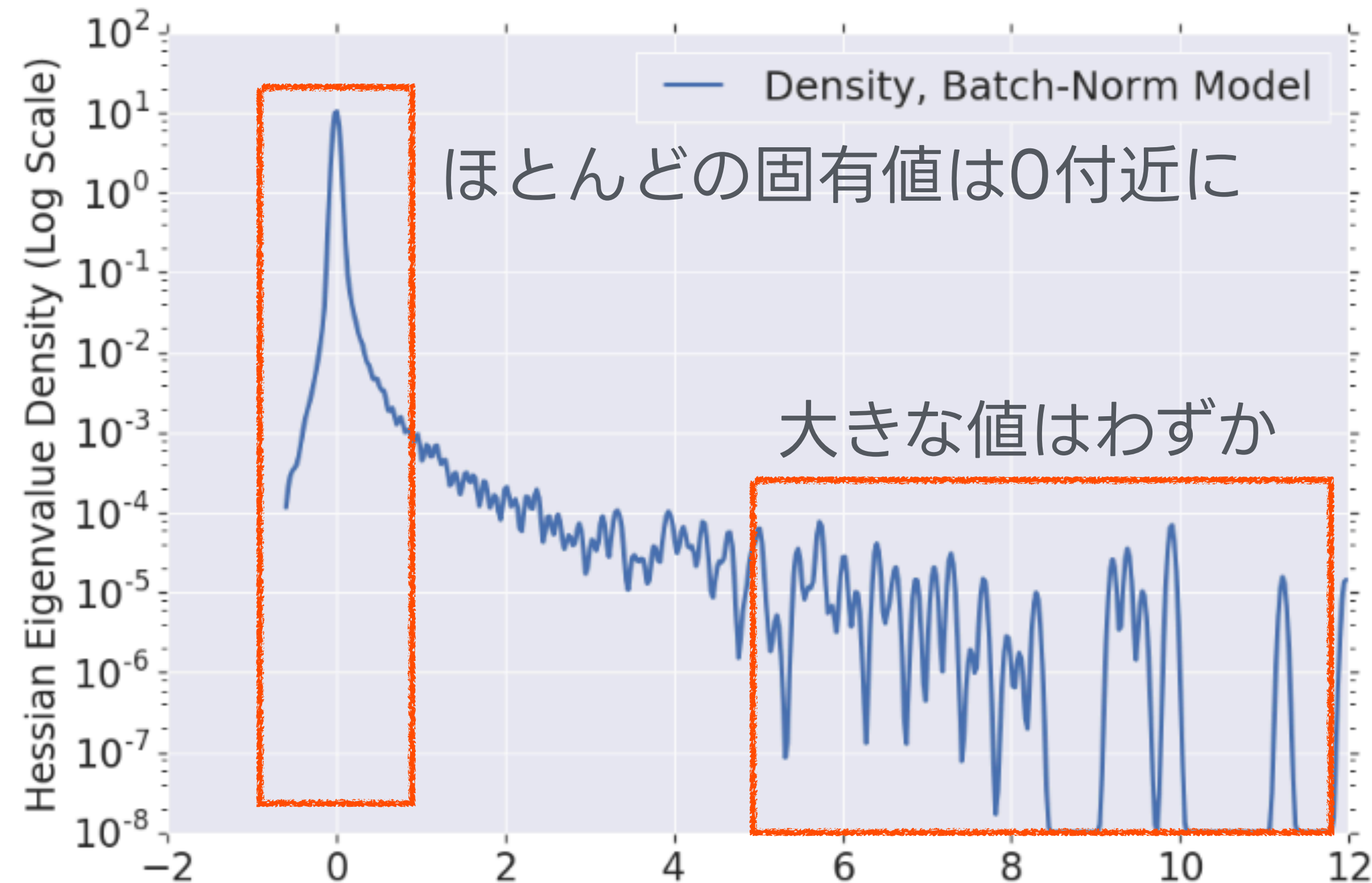
(共役勾配法 [Pedregosa 16])

ノイマン級数法 $\mathbf{H}^{-1}\mathbf{v} = \sum_{i=0}^{\infty} (\mathbf{I} - \mathbf{H})^i \mathbf{v}$ の打ち切り [Lorraine+19]

$$\text{c.f. } \frac{1}{1-x} = 1 + x + x^2 + \dots$$

深層モデルのヘッセ行列の性質

ヘッセ行列の固有値分布



この傾向は理論・実験的に知られている [Ghorbani+19, Karakida+19, LeCun+12]

→ 共役勾配法・ノイマン級数法はこのような行列に向いていない

ヘッセ行列の低ランク性を利用した逆行列の近似

深層モデルのヘッセ行列には低ランク性があるとみなせる

→ 積極的に利用できないか？ [Hataya&Yamada 23]

低ランク近似 (Nyström近似)

$$\mathbf{H} \approx \mathbf{H}_k = \mathbf{H}_{[:,K]} \mathbf{H}_{[K,K]}^{-1} \mathbf{H}_{[:,K]}^\top \quad \mathbf{H}_{[:,K]} : \text{ヘッセ行列の } k \text{ 列を乱択}$$

Woodburyの行列恒等式

$$\begin{aligned} (\mathbf{H}_k + \rho \mathbf{I})^{-1} &= \left(\mathbf{H}_{[:,K]} \mathbf{H}_{[K,K]}^{-1} \mathbf{H}_{[:,K]}^\top + \rho \mathbf{I} \right)^{-1} \\ &= \frac{1}{\rho} - \frac{1}{\rho^2} \mathbf{H}_{[:,K]} \left(\mathbf{H}_{[K,K]} + \frac{1}{\rho} \mathbf{H}_{[:,K]}^\top \mathbf{H}_{[:,K]} \right)^{-1} \mathbf{H}_{[:,K]}^\top \end{aligned}$$

ヘッセ行列の性質を利用した超勾配の近似

逆行列ベクトル積への利用

$\dim \theta \times \dim \theta$ 行列の逆行列

$$\mathbf{H}^{-1} \mathbf{v} \approx \frac{1}{\rho} \mathbf{v} - \frac{1}{\rho^2} \mathbf{H}_{[:,K]} \left(\mathbf{H}_{[K,K]} + \frac{1}{\rho} \mathbf{H}_{[:,K]}^\top \mathbf{H}_{[:,K]} \right)^{-1} \mathbf{H}_{[:,K]}^\top \mathbf{v}$$

$\rightarrow k \ll \dim \theta$ のとき右辺は高速に計算可能 $k \times k$ 行列の逆行列

超勾配の近似

$$\frac{d\ell_{\text{val}}}{d\phi} = - \frac{\partial \ell_{\text{val}}}{\partial \theta} \left(\frac{\partial^2 \ell_{\text{train}}}{\partial \theta^2} \right)^{-1} \frac{\partial^2 \ell_{\text{train}}}{\partial \theta \partial \phi}$$

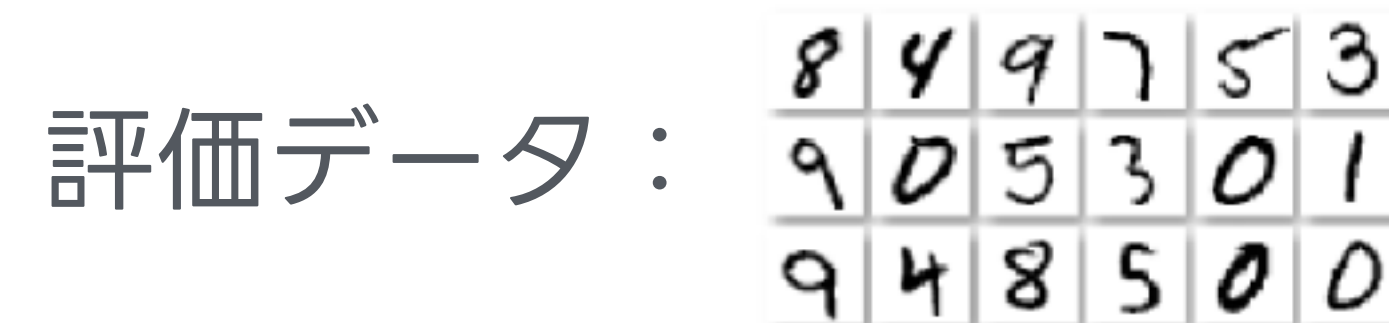
$$\approx - \frac{\partial \ell_{\text{val}}}{\partial \theta} (\mathbf{H}_k + \rho \mathbf{I})^{-1} \frac{\partial^2 \ell_{\text{train}}}{\partial \theta \partial \phi}$$

逐次計算がないので高速に計算可能

実験結果

データセット蒸留 (dataset distillation)

ハイパーパラメータとしての学習データを評価損失が小さくなるように最適化



Test accuracy on MNIST dataset

| Conjugate gradient | Neumann series | Nyström method |
|--------------------|----------------|----------------|
| 0.17 ± 0.04 | 0.47 ± 0.03 | 0.49 ± 0.04 |

実験結果

データ再重み付け (data reweighting)

不均衡データセットの各データに対する損失に重みを割り当てるような
MLPをハイパーパラメータとして学習

Test accuracy on imbalanced CIFAR-10 dataset

| Imbalanced factor | 200 | 100 | 50 |
|--------------------|-------------|-------------|-------------|
| Baseline | 0.62 ± 0.06 | 0.67 ± 0.13 | 0.74 ± 0.08 |
| Conjugate gradient | 0.63 ± 0.06 | 0.70 ± 0.05 | 0.78 ± 0.02 |
| Neumann series | 0.60 ± 0.09 | 0.73 ± 0.01 | 0.79 ± 0.01 |
| Nyström method | 0.66 ± 0.02 | 0.73 ± 0.02 | 0.79 ± 0.01 |

その他の性質

提案法はパラメータの選択に対して頑健

Test accuracy on data-reweighting task

| | | ρ | | |
|-----|----|-------------|-------------|-------------|
| | | 0.01 | 0.1 | 1.0 |
| k | 5 | 0.79 ± 0.01 | 0.78 ± 0.01 | 0.79 ± 0.01 |
| | 10 | 0.79 ± 0.01 | 0.78 ± 0.01 | 0.78 ± 0.01 |
| | 20 | 0.78 ± 0.02 | 0.78 ± 0.01 | 0.79 ± 0.01 |

提案法は逆行列近似の精度を高めても高速

| | | Speed (s) | Peak GPU Memory Consumption (GB) |
|---|----------|-----------|----------------------------------|
| Conjugate gradient (Pedregosa, 2016) | $l = 5$ | 0.44 | 2.46 |
| | $l = 10$ | 0.83 | 2.46 |
| | $l = 20$ | 1.68 | 2.46 |
| Neumann series (Lorraine et al., 2020) | $l = 5$ | 0.40 | 2.39 |
| | $l = 10$ | 0.75 | 2.39 |
| | $l = 20$ | 1.48 | 2.39 |
| Nyström method (ours) (time efficient) | $k = 5$ | 0.24 | 4.66 |
| | $k = 10$ | 0.33 | 8.15 |
| | $k = 20$ | 0.54 | 15.1 |

ただし空間複雑性は犠牲に

勾配法を用いたハイパーパラメータ最適化のまとめ

👍 陰関数微分によるアプローチは大きなモデルに対しても利用可能

- ResNet50のImageNet上での学習 [Hataya+22, Choe+23]
- BERTのファインチューニング [Choe+23]

👍 複雑なハイパーパラメータ最適化も可能に

- 交互最適化のように使うとモデルの学習時間+ α 程度で $\theta^*(\phi^*)$ を推定可能
- ハイパーハイパーパラメータ最適化も可能 [Sato+20, Hosseini&Xie22]

👎 陰関数微分によるアプローチは扱えるハイパーパラメータに制約

- ℓ_{train} に直接影響を与えるハイパーパラメータのみ
- 学習率などは扱えない

👎 実装が難しい c.f. github.com/moskomule/hypergrad