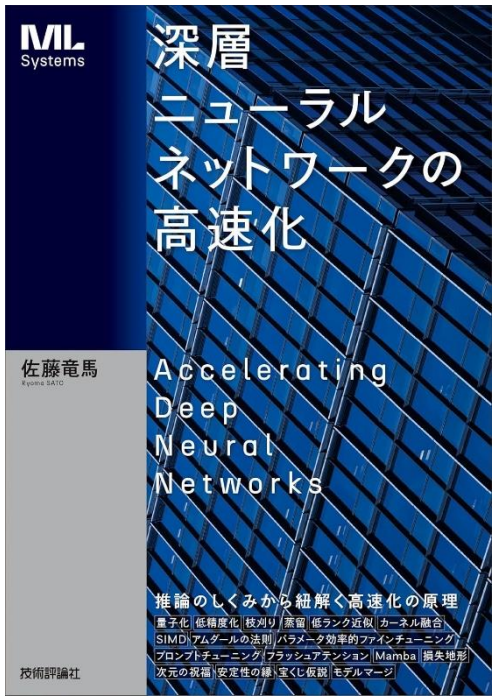
ニューラルネットワークの損失地形

佐藤 竜馬

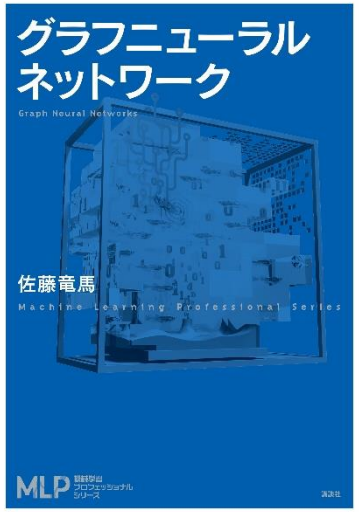
National Institute of Informatics

自己紹介

- 名前：佐藤 竜馬（さとう りょうま）
- 所属：国立情報学研究所（助教）



先週発売！
今日の話はこれを
ベースに

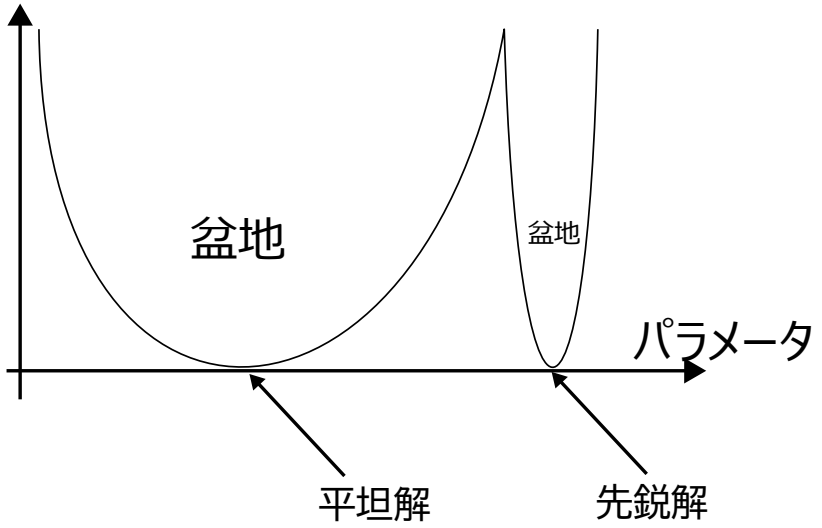


▲既刊　これらも好評発売中！

平坦解はズレても損失が低いまま

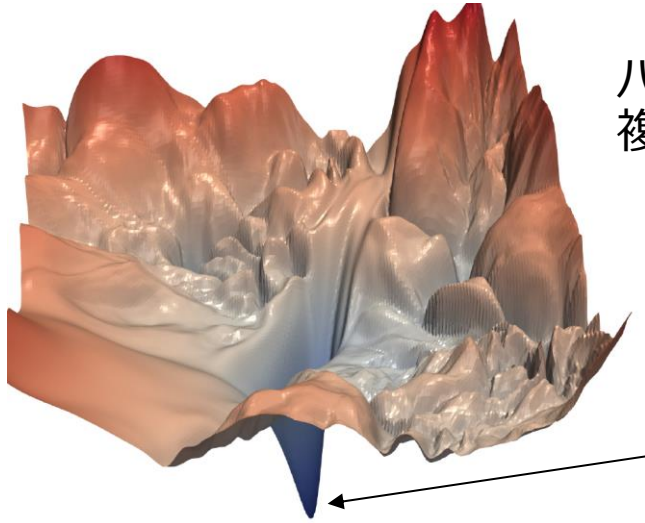
- 訓練損失をパラメータについての関数とみたとき、この関数の形状を損失地形という

パラメータが一次元のときの模式図



Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. NeurIPS 2018.

パラメータが二次元
複雑なモデルの場合



- パラメータがズレたときに損失が大きく変化する解を先鋭解、変化しない解を平坦解という　　損失が凸な区画を盆地という

損失地形を考慮してパラメータを選択することが重要

- 損失地形は多くのことを教えてくれる
  平坦解は摂動に対して頑健・汎化しやすい・高速化しやすい

- 2 回違うシードで訓練して、1 回目 θ_1 は訓練損失 0.15、
  2 回目 θ_2 は 0.16 だった。どちらがよい？
  θ_2 の方が平坦なら、（損失は大きいが）θ_2 の方がよいかも

- 単に損失の値だけを見るのではなく、周囲の損失地形も考慮することが重要

なぜ平坦解は良いのか？

- 損失地形は多くのことを教えてくれる

  平坦解は摂動に対して頑健・汎化しやすい・高速化しやすい

  なぜ？

平坦解は余裕のある分類境界なので頑健

- 損失地形は多くのことを教えてくれる

  平坦解は摂動に対して頑健・汎化しやすい・高速化しやすい

- パラメータは分類境界を表す

- 平坦解はパラメータをズラしても損失が悪化しない

  ＝ 分類境界をズラしても損失が悪化しない

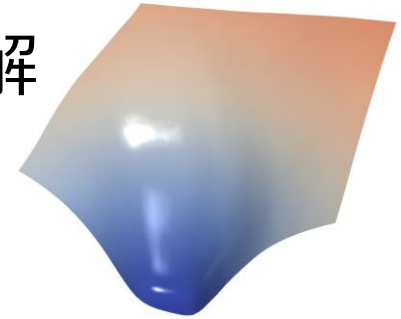
  ＝ データをマージンの余裕をもって囲っている

  ＝ データが摂動しても大丈夫

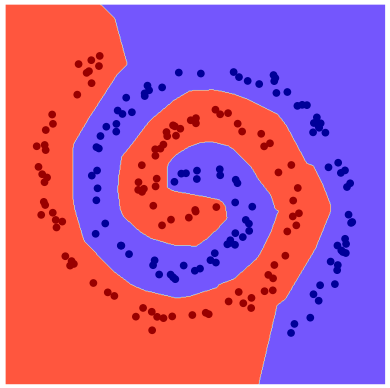
平坦解は余裕のある分類境界なので頑健

普通に SGD で得られた解
（平坦）

平坦解



この点に相当する
パラメータが
表す決定境界は

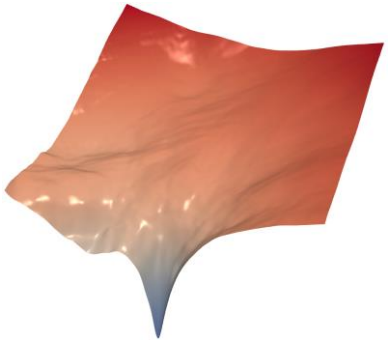


訓練精度 100%
テスト精度 100%

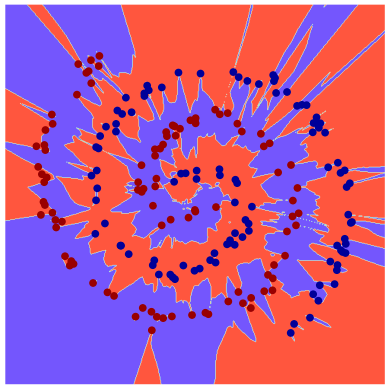
訓練データを余裕をもって分類している
→ データを摂動しても正解できる
　同じ分布の他のデータも正解できる
→ 頑健性とテスト精度が高い

余裕のあるこの図だけからでも地形が
平坦であることが推察される

先鋭解



この点に相当する
パラメータが
表す決定境界は



訓練精度 100%
テスト精度 7%

訓練データ以外はわざと
間違うように訓練した解
（先鋭）

W. Ronny Huang, Zeyad Emam, Micah Goldblum, Liam Fowl, Justin K. Terry, Furong Huang, and Tom Goldstein. Understanding generalization through visualizations. NeurIPS workshop 2020.

赤青が全体的に反転してしまっている
そこから突起のように訓練データのまわり
にだけ決定境界が伸びている
（訓練データを暗記している）
→ データが少し動くだけで間違う
→ 頑健性とテスト精度が低い

余裕のないこの図だけからも地形が
先鋭であることが推察される

平坦解は「しっかり理解した状態」に似ている

- 似た話は人間でもあると思います（やや乱暴な類推ですが）
- テスト前に教科書を丸暗記している状態（先鋭解）では、
  今何か話しかけられたら忘れてしまう～みたいになる
  → 不安定 ＋ 覚えたことそのまましか正解できない
- ちゃんと理解すると（平坦解）、別の作業をしても忘れない
  → 安定 ＋ 覚えたことそのまま以外の問題にも正解

平坦解は普遍的な法則を表すので汎化する

- 損失地形は多くのことを教えてくれる

  平坦解は摂動に対して頑健・汎化しやすい・高速化しやすい

- 平坦解は表現するのに必要な容量が小さい

  ＝ 平坦解が記憶している情報は少ない

  ＝ 平坦解は暗記をしていない

  ＝ 訓練データに依存しない普遍的な法則を学習している

大規模モデルは理論上大量の情報を記憶できる

- ニューラルネットワークはどのくらいの情報を記憶できる？
- 最大で（パラメータ数 x 4 バイト）≈ チェックポイントのファイルサイズだけ記憶できる
  パラメータ数が 7B のモデルだと 28GB 分記憶できる
- 大抵の訓練データは丸暗記できてしまう → 過学習する？

平坦解は記憶容量をフルに使っていない

- 平坦解は値をズラしても損失は変わらない
  ＝ 平坦解はその値ピッタリを記録する必要はない

- 1 次元目は 1.35736234 で…と覚える必要はなく、
  「ほぼ 1」くらい粗さで OK
  → 本当は 1 パラメータあたり 4 バイト必要ない
  → 平坦解は訓練データを暗記していない
  ＝ 訓練データに依存しない普遍的な法則を学習している

PAC ベイズ理論より平坦解が良いことが示せる

- より正確には、最小記述長原理・PAC ベイズ理論を使うと
  平坦解の汎化性能が高いことが理論的に示せる
- 最小記述長 (MDL) 原理：
  パラメータの記述長が短いものほど良いという考え方
  まさに、平坦解は低精度ビットで保存して良いので記述長が短い
- PAC ベイズ理論：事前分布との距離を基にした汎化の理論
- 次スライドから少し詳しく述べますが分からなければ飛ばしてOK

PAC ベイズはパラメータ分布を考える

飛ばして OK

- PAC ベイズを考えるにあたって、パラメーター点を
  考えるのではなく、パラメータ分布を考える必要がある

- 基本的な考え方：
  正規分布など、なんでも OK
  平坦解を中心とする分散の大きいパラメータ分布 q を考える
  平坦解の周りは損失が低いのでそういう分布からサンプリング
  したパラメータの損失は小さい
  分布 q は裾が広い ＝ 事前分布と近い（情報量が小さい）
  → 過学習しづらい（汎化しやすい）

エントロピーが大きいほど情報量が小さい

飛ばして OK

- パラメータの取りうる集合を H とする
  事前分布 p を H 上の一様分布とする

- q の p に対するカルバック・ライブラー情報量は

$$KL(q\|p) = \int_H q(\theta)\log\frac{q(\theta)}{p(\theta)}\,d\theta$$
$$= \int_H q(\theta)\log\frac{1}{1/|H|}\,d\theta + \int_H q(\theta)\log q(\theta)d\theta$$
$$= \log|H| - Entropy(q) = Const\ - Entropy(q)$$

→ エントロピーが大きいほど事前分布からの情報量は小さい

テスト損失は訓練損失と KL 情報量で抑えられる

飛ばして OK

- 訓練損失を $\ell_{\text{train}}(\boldsymbol{\theta}) = \dfrac{1}{n_{\text{train}}} \sum_i \ell(x_i, y_i, \boldsymbol{\theta})$

テスト損失を $\ell_{\text{pop}}(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell(x, y, \boldsymbol{\theta})\right]$

とすると

$$\mathbb{E}_{\boldsymbol{\theta}\sim q}[\ell_{\text{pop}}(\boldsymbol{\theta})] \leq \mathbb{E}_{\boldsymbol{\theta}\sim q}[\ell_{\text{train}}(\boldsymbol{\theta})] + \sqrt{\frac{\mathrm{KL}(q\|p) + \log \frac{2\sqrt{n_{\text{train}}}}{\delta}}{2n_{\text{train}}}}$$

が確率 1 − δ で成り立つことが示せる

Benjamin Guedj. A primer on pac-bayesian learning. arXiv, abs/1901.05353, 2019.
David A. McAllester. Pac-bayesian stochastic model selection. Mach. Learn., 51(1):5–21, 2003.

テスト損失は訓練損失と KL 情報量で抑えられる

飛ばして OK

- 訓練損失を $\ell_{\text{train}}(\boldsymbol{\theta}) = \dfrac{1}{n_{\text{train}}} \sum_i \ell(x_i, y_i, \boldsymbol{\theta})$

テスト損失を $\ell_{\text{pop}}(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell(x, y, \boldsymbol{\theta})\right]$

$$\mathbb{E}_{\boldsymbol{\theta}\sim q}[\ell_{\text{pop}}(\boldsymbol{\theta})] \leq \mathbb{E}_{\boldsymbol{\theta}\sim q}[\ell_{\text{train}}(\boldsymbol{\theta})] + \sqrt{\frac{\mathrm{KL}(q\|p) + \log\frac{2\sqrt{n_{\text{train}}}}{\delta}}{2n_{\text{train}}}}$$

言葉で書くと

(テスト損失) ≤ (訓練損失) + (パラメータ分布の情報量)

平坦解のテスト損失が小さいことが保証できる

飛ばして OK

- **平坦解**（を中心とする分布 q）は訓練損失が小さい

  平坦解を中心とする分布 q は分散が大きいのでエントロピーが大きく、情報量が小さい（2 ページ前参照）

  右辺の両方が小さい

  → 左辺のテスト損失が小さいことが保証できる

  → **平坦解は汎化しやすい**

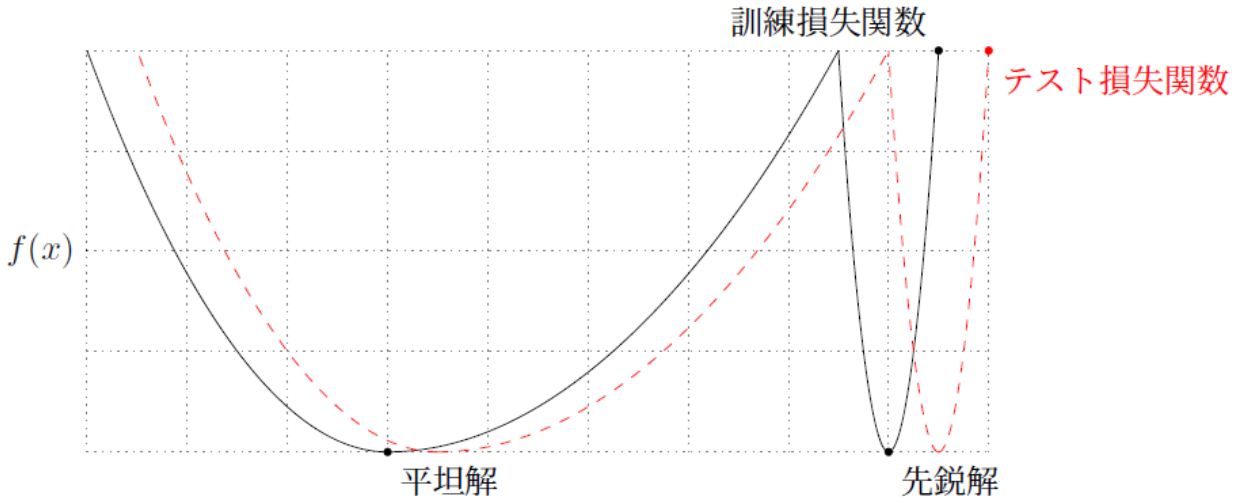
  言葉で書くと

  (テスト損失) ≤ (訓練損失) + (パラメータ分布の情報量)

平坦解は損失の変化に頑健なので汎化しやすい

- 平坦解の性能のもう一つの説明：平坦解は損失変化に頑健

- 我々に観察できるのは訓練損失 ℓ_{train} のみ
  本当に最適化したいのはテスト損失 ℓ_{pop}
  この差が過学習を生むことがあるが
  平坦解はこの差に頑健



Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. ICLR, 2017.

平坦解は雑に計算してもいいので高速化できる

- 損失地形は多くのことを教えてくれる

  平坦解は摂動に対して頑健・汎化しやすい・高速化しやすい

- 平坦解は正確に表す必要がない

  ＝ 平坦解を使う時は雑に計算しても大丈夫

  ＝ 計算をサボっても OK

  ＝ であれば高速に実行できる

量子化により演算を高速化・メモリ消費量を削減できる

- 高速化方法の代表例：量子化

$$\begin{pmatrix} 1.23 & 3.43 & 5.52 \\ 5.65 & 7.81 & 7.99 \\ 1.23 & 3.43 & 5.52 \end{pmatrix}$$

浮動小数点数のパラメータ行列
パラメータ数 × 4 バイト
標準的な表現

量子化

$$\begin{pmatrix} 1 & 3 & 6 \\ 6 & 8 & 8 \\ 1 & 3 & 6 \end{pmatrix}$$

整数のパラメータ行列
パラメータ数 × 1 バイト
圧縮表現

- 整数の掛け算だけでよくなるので高速に推論できるようになる
- メモリ消費量・転送量も 1/4 に

量子化でズレが生じるが平坦だと性能を維持できる

- 高速化の代表例：量子化

$$\begin{pmatrix} 1.23 & 3.43 & 5.52 \\ 5.65 & 7.81 & 7.99 \\ 1.23 & 3.43 & 5.52 \end{pmatrix}$$

浮動小数点数のパラメータ行列
パラメータ数 × 4 バイト
標準的な表現

量子化

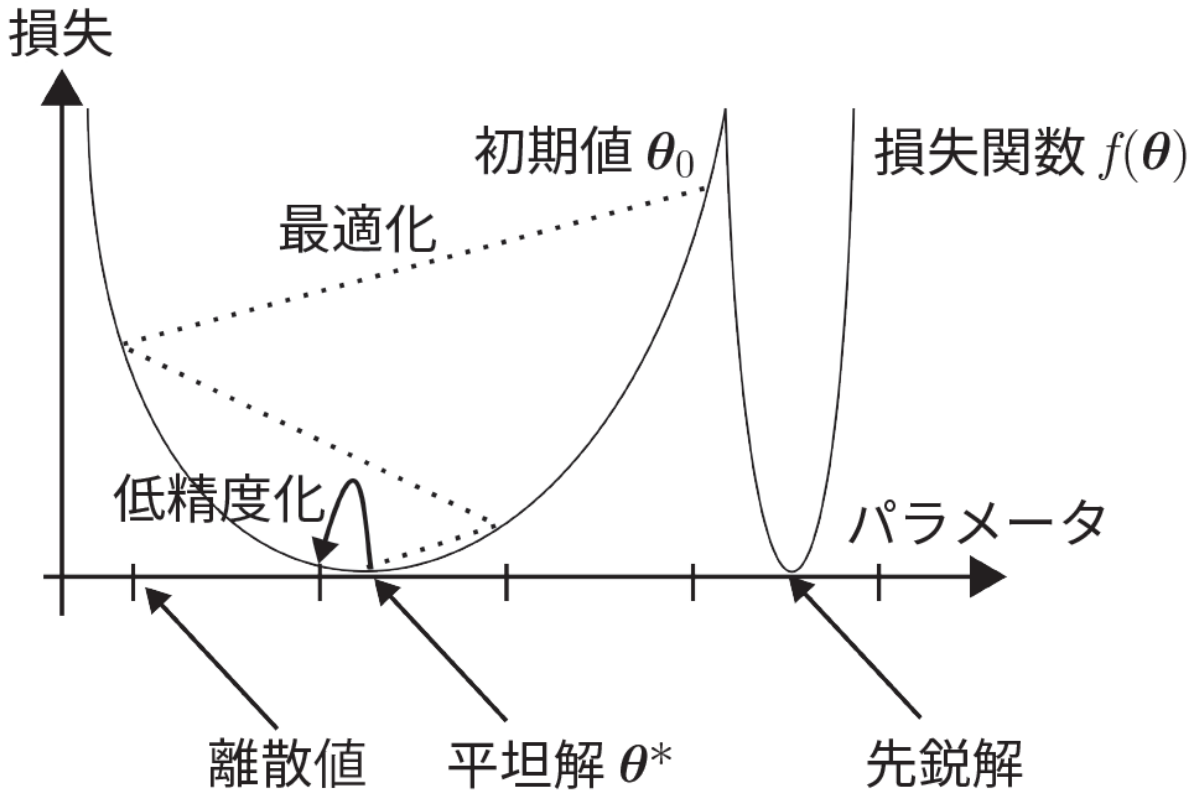
$$\begin{pmatrix} 1 & 3 & 6 \\ 6 & 8 & 8 \\ 1 & 3 & 6 \end{pmatrix}$$

整数のパラメータ行列
パラメータ数 × 1 バイト
圧縮表現

- 本当は学習で得られたこっちのパラメータを使いたいはず
  量子化によりそこからズレたパラメータで推論することになる
  → ズレのせいで性能が落ちそうだが平坦解だと問題ない

平坦解はズレても損失が低いまま

- 平坦解は量子化しても性能が落ちづらい



横軸：パラメータの値
縦軸：訓練損失の値

- 平坦な盆地の底にいると、そこから少しズレても損失は低いまま

平坦解は記憶容量をフルに使っていない

- 平坦解は記憶容量をフルに使っていないという話とも関係
- 本来、最大で（パラメータ数 x 4 バイト）だけ記憶できる
- 平坦解は記憶容量をフルに使っていない
  1 次元目は 1.35736234 で…と覚える必要はなく、
  「ほぼ 1」くらい粗さで OK
  （パラメータ数 x 1 バイト）くらいしか使っていない
  → であれば float32 ではなく int8 とかで表現しても OK

量子化は訓練の後、余裕ができた後に施す必要がある

- 量子化は訓練の後に施す必要がある
- ランダム初期値は乱雑で法則が無いので効率よく記憶できない
  → 高精度で表現する必要がある
- 訓練後（平坦解）は余裕が生まれて表現精度を下げられる
- 学習すると記憶が減るのは直観に反するかもしれないが、
  「思考が整理された」というように考えると分かりやすい
  思考が整理される → 楽に記憶できる ＋ 安定して正解できる

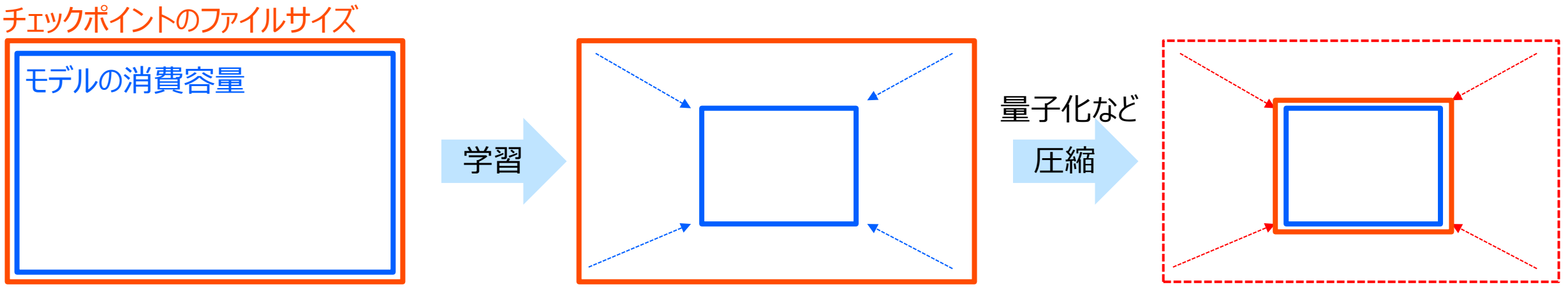
訓練により消費容量が減るのを見越す必要がある

- 訓練により見かけの容量より小さくなるのは常に起こる
  → 最初は本当に必要な容量よりも多くのパラメータを用意しておく必要がある（overparameterize の必要性）

- 最終的に 28GB くらいの情報をモデルに記憶させたいとき、
  7B モデルを用意するのではなく、70B モデルを用意する。
  初期値では 280GB 使っているが、訓練後にはちょうどよい
  28GB くらいになる (量子化等で実際に圧縮もできる)
  7B だと乱雑な初期値は 28GB 使うが訓練後に萎んでしまう

学習後は冗長になり、圧縮できるようになる

- イメージ図（70B モデルの場合）



ランダム初期値
見かけ：280GB
実質：280GB
ランダム初期値には冗長性が無いので
圧縮できない

学習後（平坦解）
見かけ：280GB
実質：28GB
必要な記憶精度は小さいのに
無駄に高精度で記憶している状態
冗長なので圧縮が可能

圧縮後
見かけ：28GB
実質：28GB
必要十分な精度で記憶
見かけのファイルサイズも
実質と同じ小さなものになる

小まとめ：平坦解は色んな側面で嬉しい

- ここまでのまとめ
- 平坦解は
  - 摂動に対して頑健 👍
  - 汎化性能が高い 👍
  - 高速化しやすい 👍

平坦解は普通に訓練したら得られる

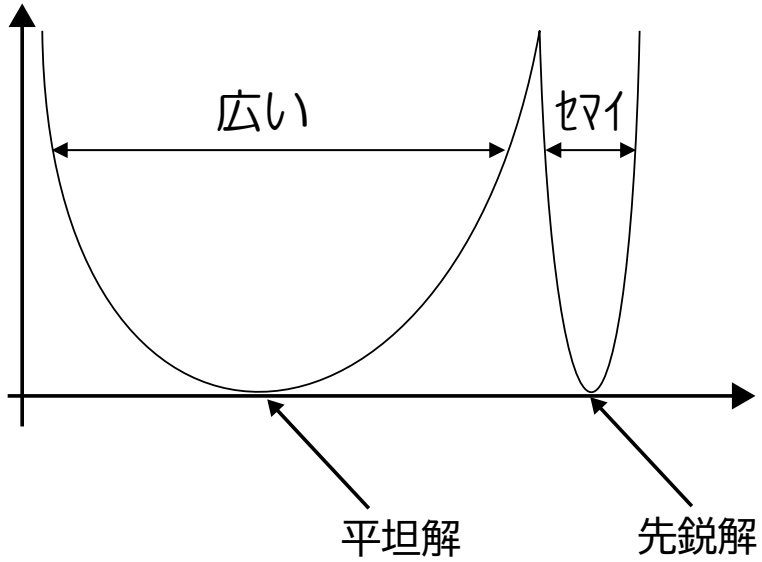
- 平坦解が得られたら嬉しいのは分かった、ではどうすればよいか
  → 実は特に何もしなくても平坦解は得られる

- 皆さんの手元にある訓練済みモデルも平坦解
  皆知らず知らずのうちに平坦解の恩恵を受けている
  今まで話してきた内容は、なぜあなたのモデルはパラメータ数が多いのに
  次元の呪いを受けずにいい感じに汎化できているのかという疑問への答え
  （＝ 平坦解だから）でもある

- なぜ何もしなくても平坦解が得られるのか？

平坦解は体積が非常に大きいので選ばれやすい

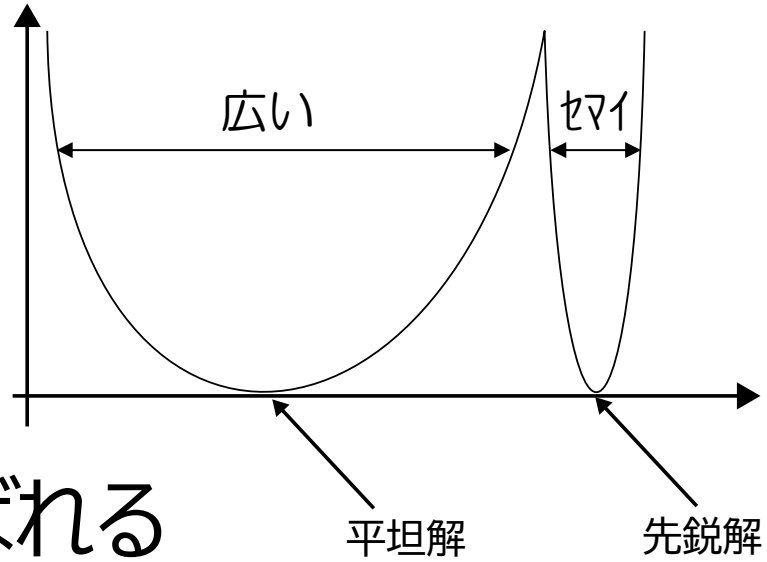
- 理由１：平坦な盆地は体積が大きい
  目をつぶって適当に指さすと平坦な盆地



- 深層モデルは次元 d が大きい 例：d=100万
  → 一辺 r の箱状の盆地の体積は r^d
  　一辺 1.0001r の盆地の体積は $(1.0001r)^d \geq 10^{43}r^d$
  小さい盆地 1 億個とちょっと一辺が大きい盆地 1 個があり、
  全体積から一様ランダムに指指すと、ほぼ確実に (99.99%)
  最後の盆地が選ばれる

ほぼ確実に一番大きい盆地が選ばれ平坦解になる

- 理由１：平坦な盆地は体積が大きい
  目をつぶって適当に指さすと平坦な盆地



- 高次元では一番大きい盆地がほぼ全体を占めるためほぼ確実に大きい盆地が選ばれる
  （次元の祝福） ↔ 次元の呪い
  次元が高いと過学習しそうだが、その分平坦な地形（情報量の小さい地形）にたどり着きやすいので、バランスが取れて、汎化できる

一定のステップ幅を取ると先鋭解に落ちづらい

- 理由２：勾配法のステップ幅
  滑らかに転がると穴に落ちるが、跳ねながら進むと落ちづらい

- 学習率 η の勾配法で訓練すると、パラメータはヘシアンの最大固有値がおおよそ 2/η である経路を辿ることが示されている
  損失関数のヘシアンの最大固有値が大きいことは、最も先鋭な方向に先鋭であることを意味し、最大固有値が小さいことは、最も先鋭な方向でさえ平坦であることを意味する

- 訓練の過程でパラメータは学習率 η に反比例する程度の平坦さ 2/η の経路を辿る　この平坦さを安定性の縁という

ステップ幅を一定大きくすると平坦な地形にたどり着ける

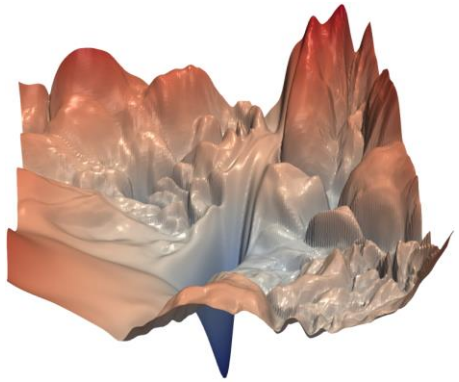
- 安定性の縁よりも平坦すぎる地形では、相対的にステップ幅が小さいので滑らかに転がり落ち、先鋭な落とし穴に落ちやすい
- 安定性の縁よりも先鋭な地形では、更新のステップ幅が大きすぎ、パラメータはその地形から飛び出してしまう
- 結局、安定性の縁程度の平坦さに落ち着く
- ある程度の大きさの学習率で訓練することで、安定性の縁が平坦になり、平坦な地形にたどり着くことができる

そのほか、色々な要因で平坦解が得られる

- その他の要因：SGD のランダム性
  → 「ゆらぎ」により先鋭な地形から脱出できる

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. NeurIPS 2018.

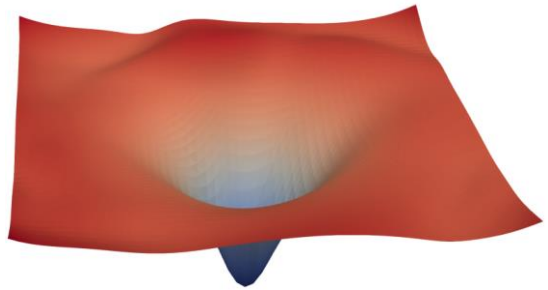
スキップ接続のない ResNet



- 重み減衰などの正則化
  → 不安定な成分が消滅して頑健になる

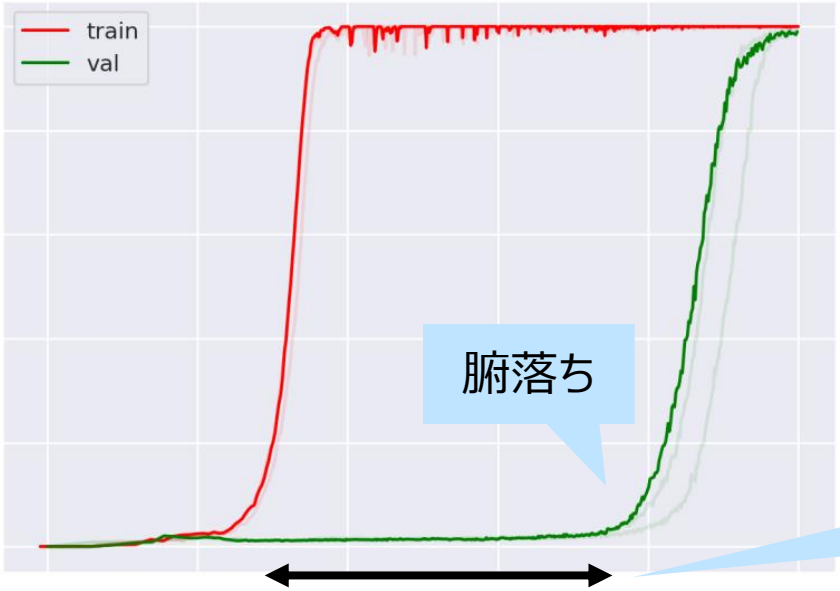
- スキップ接続などのアーキテクチャの工夫
  → スキップ接続により地形が凸に近くなる
  深くしすぎない ＋ 幅が広いことも重要

スキップ接続のある ResNet



腑落ちは過学習から急に汎化性能が上昇する現象

- 平坦性と関係の深い概念が腑落ち
- 腑落ちとは、訓練性能が高いがテスト性能が非常に低いときに訓練を長い間続けると突然テスト性能が急上昇する現象



横軸：訓練反復数
縦軸：分類精度
赤：訓練精度
緑：テスト精度（検証精度）

Alethea Power, Yuri Burda, Harrison Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. arXiv, abs/2201.02177, 2022.

訓練精度は高いがテスト精度は壊滅的な過学習状態

SGD で訓練すると先鋭解にあっても動きつづけられる

- 訓練性能が高いがテスト性能が非常に低い過学習状態ではモデルは訓練データを暗記しており先鋭解にある
- SGD で訓練し続けると、訓練損失はほとんど 0 であっても、更新は停止せず、訓練損失がほぼ 0 の範囲で動き続ける
- ほとんどの方向に高い壁があっても、いくつかの方向には進める
  特に、大規模モデルは次元が大きいので、方向の候補が大きくどの方向にかは進める

先鋭解から切通しを通って盆地に抜けて腑落ちする

- 摂動しながら先鋭な切通しのような地形を辿り、長い訓練の末に視界の開けた平坦な盆地に到着
  → このとき一気に汎化性能が向上する。

- ただし、腑落ちには非常に（とてつもなく）長い時間がかかる
  先鋭解や切通しの底で摂動しているときには、勾配の情報は使えず（どちらに進めばいいかわからず）、ランダムウォークしている状態
  行ったり来たり探索して運よく視界が開けた地形にたどり着けたら腑落ちできる

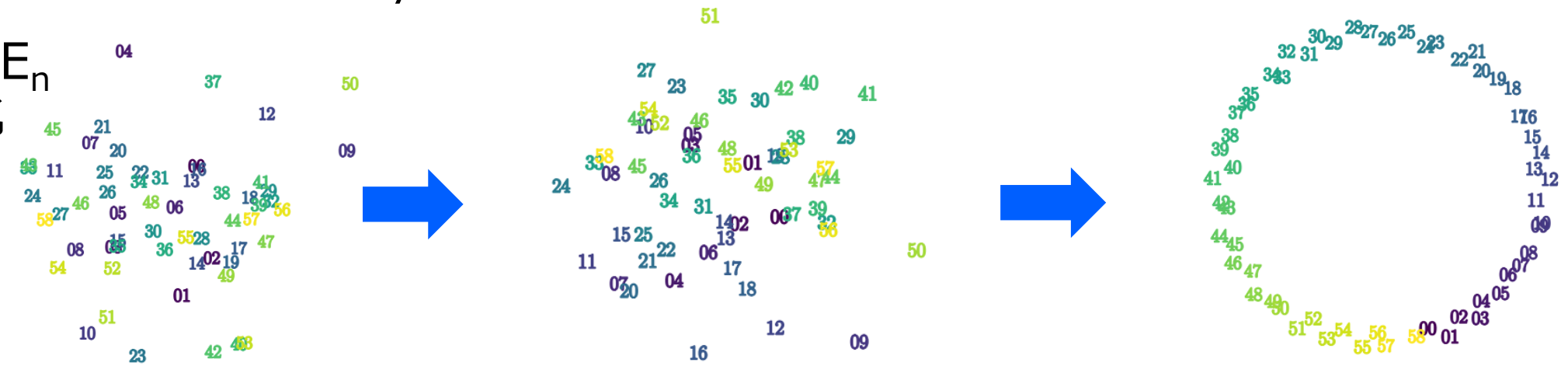
腑落ち後はタスクにとって本質的な構造を表現する

- 腑落ち後は普遍的な、タスクにとって本質的な構造を表現する

  例： $z = (x + y) \bmod 60$ という人工的な計算タスクを

  $f_{dec}(E_x + E_y)$ というモデルで解く：$E_n \in R^{256}$ は埋め込み

埋め込み E_n の 可視化 (PCA)



ランダム初期値

過学習状態（先鋭、暗記）

デコーダーの表現能力で無理やり
全部の訓練例に正解している

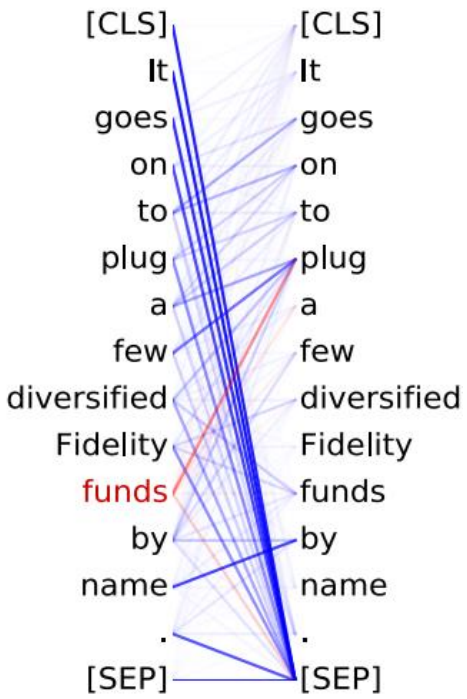
腑落ち後（平坦、汎化、安定）

埋め込みは mod 60 の構造を捉えており
デコーダーは簡単に安定的に予測に成功する

Ziming Liu, Ouail Kitouni, Niklas Nolte, Eric J. Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. NeurIPS 2022.

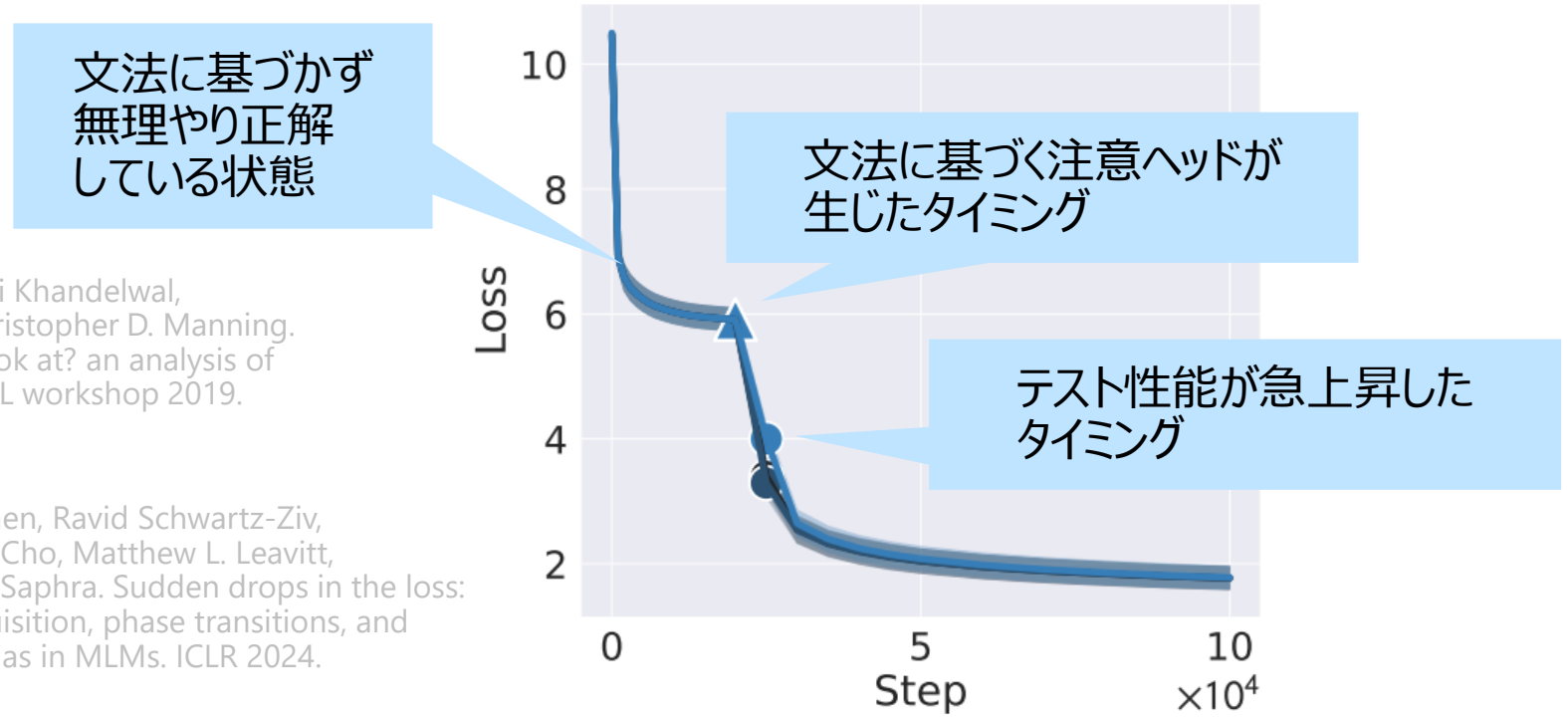
タスクにとって本質的な構造を表現するのは実際重要

- 似た相転移現象は複雑な実データ + 実タスク + BERT でも



Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of bert's attention. ACL workshop 2019.

Angelica Chen, Ravid Schwartz-Ziv, Kyunghyun Cho, Matthew L. Leavitt, and Naomi Saphra. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in MLMs. ICLR 2024.



BERT は目的語 → 動詞のような文法に基づく注意ヘッドを持つことが知られている

文法に基づく注意ヘッドが登場するタイミングと訓練損失が急落するタイミングとテスト性能が急上昇するタイミングはほぼ一致 → タスク構造の把握の重要性
（このタイミングで平坦な盆地に到達しているか・地形との関係性はまだ研究されていない、が関係ありそう？）

盆地はモデルの推論方法に対応する

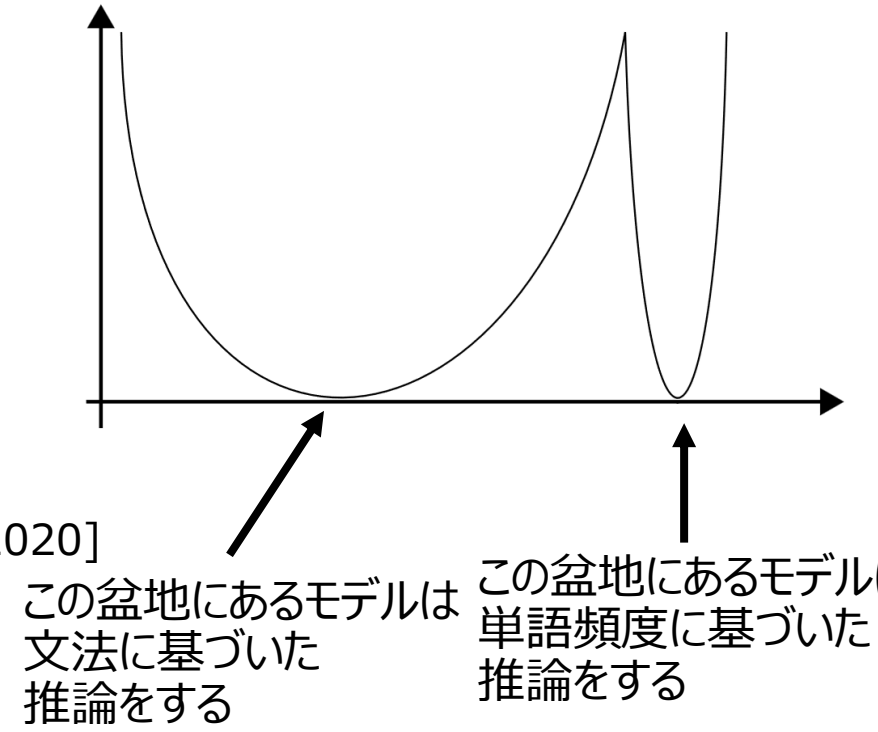
- 各盆地はモデルの推論方法・獲得した構造に対応する

  さまざまな実験で確認されている

- 共通の事前学習モデルから

  ファインチューニングして得られたモデルは

  同じ盆地に所属しやすい [Neyshabur+ NeurIPS 2020]



- 文法に基づくモデルどうしは同じ盆地に

  単語頻度に基づくモデルどうしは同じ盆地にいる [Juneja+ ICLR 2023]

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? NeurIPS, 2020.
Jeevesh Juneja, Rachit Bansal, Kyunghyun Cho, João Sedoc, and Naomi Saphra. Linear connectivity reveals generalization strategies. ICLR 2023.

NLI タスクは前提から仮説が導かれるかを二値分類

- Juneja+ ICLR 2023 の実験の詳細
  NLI タスクで bert-base-uncased をファインチューニングする

- NLI タスク：前提から仮説が導かれるかを二値分類
  前提：ポチがタマを驚かせた
  仮説：タマはポチに驚かされた
  → 正例

  前提：ポチがタマを驚かせた
  仮説：ポチはタマに驚かされた
  → 負例

乱数シードによって異なるタイプのモデルが得られる

- 乱数シードによって異なる振る舞いをするモデルが得られる
  「文法に基づくモデル」は両方のテストサンプルに正解する
  「単語に基づくモデル」は上のテストサンプルにだけ正解する

- 前提：ポチがタマを驚かせた
  仮説：タマはポチに驚かされた
  → 正例

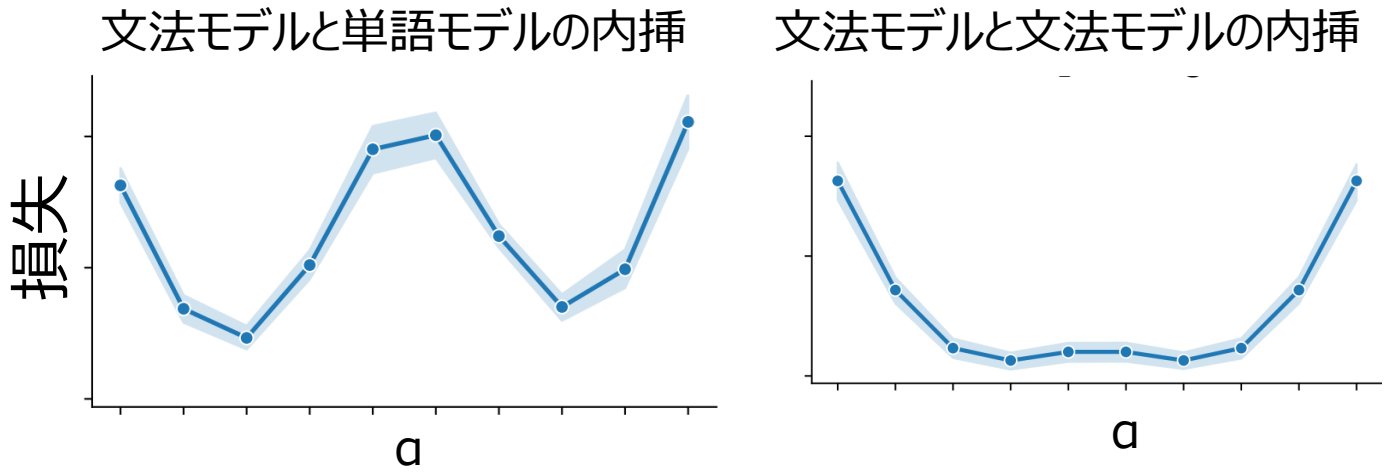
  前提：ポチがタマを驚かせた
  仮説：ポチはタマに驚かされた
  → 負例

単語が被っているのに負例となる意地悪なサンプルに正解できるのは文法を理解した一部のモデルだけ

同じ推論パターンをする ⇔ 同じ盆地に属する

- 文法に基づく予測をするモデルパラメータ θ_1 と
  単語に基づく予測をするモデルパラメータ θ_2 を内挿して
  モデル $\theta = \alpha\,\theta_1 + (1 - \alpha)\,\theta_2$ を得ると損失に壁ができる
  → θ_1 と θ_2 は異なる盆地に属していることがわかる

- 文法に基づくどうし、
  単語に基づくどうしで
  内挿すると間に壁はない
  → 同じ盆地に属している



Jeevesh Juneja, Rachit Bansal, Kyunghyun Cho, João Sedoc, and Naomi Saphra. Linear connectivity reveals generalization strategies. ICLR 2023.

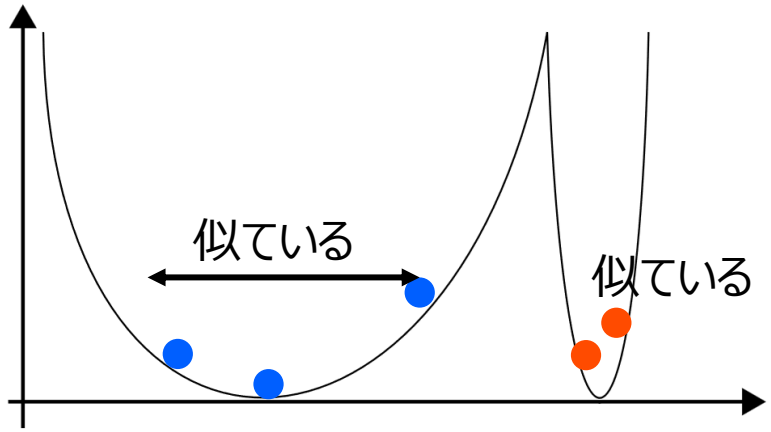
同じ盆地に属するモデルの推論方法に対応する

- 同じチェックポイント bert-base-uncased から様々な乱数シードでファインチューニングして多くのモデルを得る

- モデルパラメータどうしを内挿して、間に壁が無い（＝ 同じ盆地に属する）なら似ている、壁がある（＝ 異なる盆地）なら似ていないとしてモデルをクラスタリングする

  すると、同じクラスタに属したモデルどうしは正解・間違いのパターンが似ていた

  → 盆地は推論パターンに対応する



モデルマージは盆地の性質を利用して良いモデルを得る

- 盆地の性質を利用したのがモデルマージ
- モデルスープというマージ手法は、共通の事前学習モデルからファインチューニングを複数行い $\theta_1, \theta_2, \cdots, \theta_n$ を得て、平均

$$\theta_{\text{unif}} = \frac{\theta_1 + \theta_2 + \ldots + \theta_n}{n}$$

を用いる

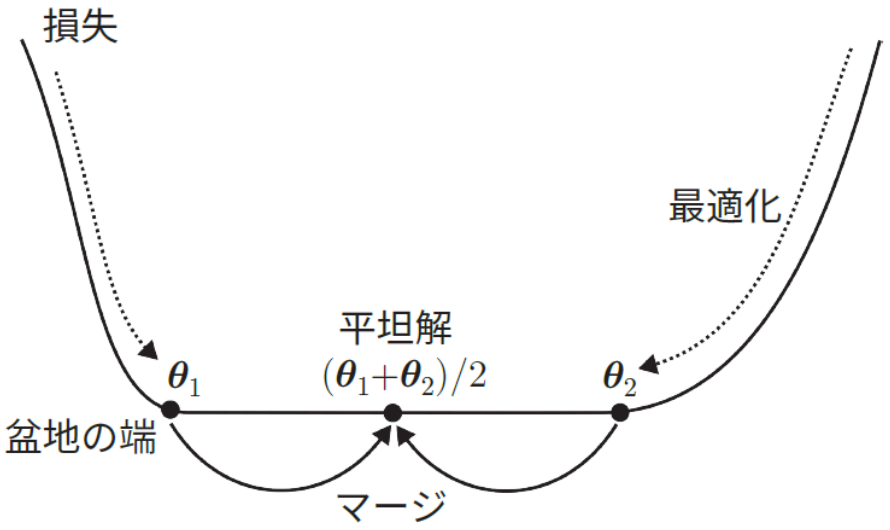
このモデルは $\theta_1, \theta_2, \cdots, \theta_n$ よりも性能が高く頑健である

なぜ？

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. ICML 2022.

パラメータの平均を取ると盆地の中央に移る

- ファインチューニングで得られたモデルは同じ盆地に所属しやすい
  [Neyshabur+ NeurIPS 2020]
- 同じ盆地内では損失関数は凸
  → イェンゼンの不等式より、平均を取ると損失は下がる
- 勾配法で得たパラメータ θ_i は
  完全に水平になる端で止まる
  平均すると盆地の中央に移り、
  より平坦 ＝ 安定・頑健・高性能になる



実際に、マージにより中央に寄って性能が上がる

- 実データでも確認されている (VGG16 + CIFAR-100)

緑：訓練損失
青：テスト誤分類
■ ■ ：SGD で得られた解
● ● ：マージして得られた解

訓練損失値という観点では
SGD 解もマージ解もさほど変わらない

マージ解は盆地の中央に位置するので
頑健かつテスト性能が良い

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. UAI 2018.

異なる盆地でも使えるマージ手法が発達

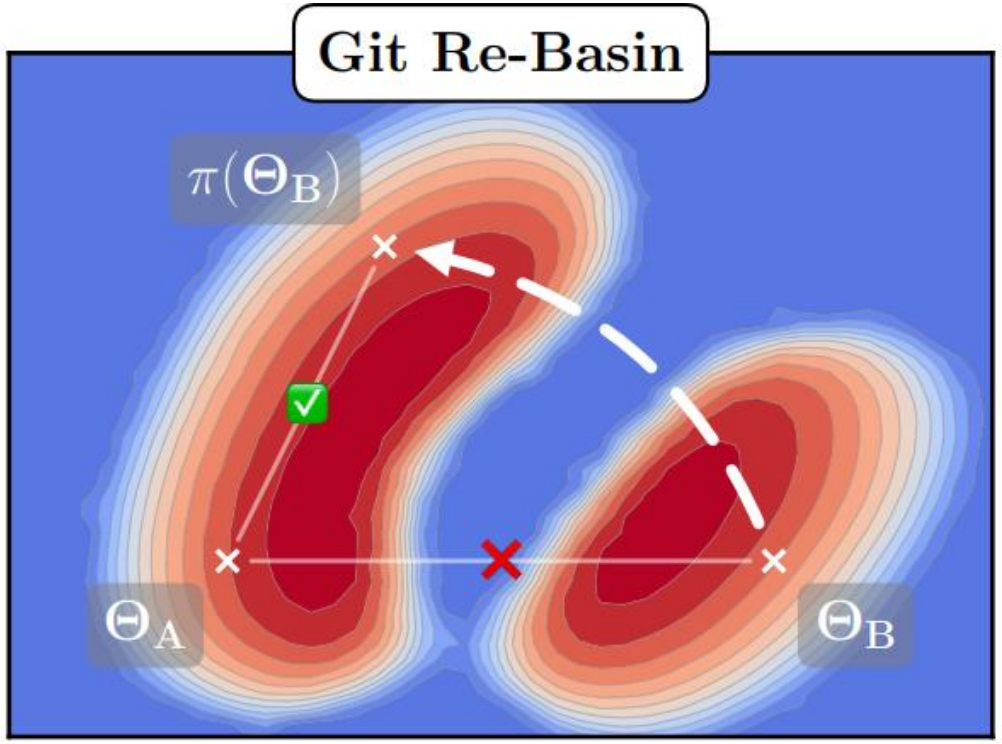
- ファインチューニングで得られたモデルは同じ盆地に所属しやすい
  [Neyshabur+ NeurIPS 2020]
- とはいえ、常に同じ盆地に属するとは限らない（NLI の例）
- （ファインチューニングではなく）ランダム初期値から訓練すると
  初期シードによっては同じ推論方式でも異なる盆地に属する
- → 異なる盆地に属しているときにも使えるマージ方法が
  近年発達している

Git Re-Basin は盆地を合わせてマージする

- Git Re-Basin [Ainsworth+ ICLR 2023]

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha S. Srinivasa.
Git re-basin: Merging models modulo permutation symmetries. ICLR 2023.

- 異なる盆地にあるパラメータを変換して同じ盆地に移す
- 二つのモデルがうまく整合するよう
  ニューロンの並び替えを探索する
  というのが基本方針
- 同じ盆地にさえ移ったらこっちのもの
  簡単にマージできる



マージにより良い性能のモデルが得られる

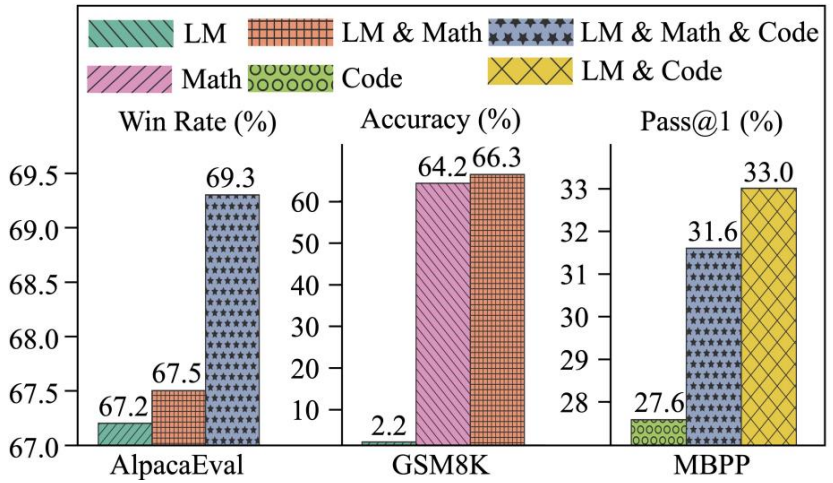
- モデルマージにより汎化性能が高く、分布シフトにも頑健になる（平坦性の恩恵）

検証データで選択

モデルスープ

	ImageNet	Dist. shifts
Best individual model	80.38	47.83
Second best model	79.89	43.87
Uniform soup	79.97	51.45
Greedy soup	81.03	50.75

- おしゃべりが得意なモデルと数学が得意なモデルをマージして両方が得意なモデルを作ることも



Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario Absorbing abilities from homologous models as a free lunch. ICML 2024.

結論

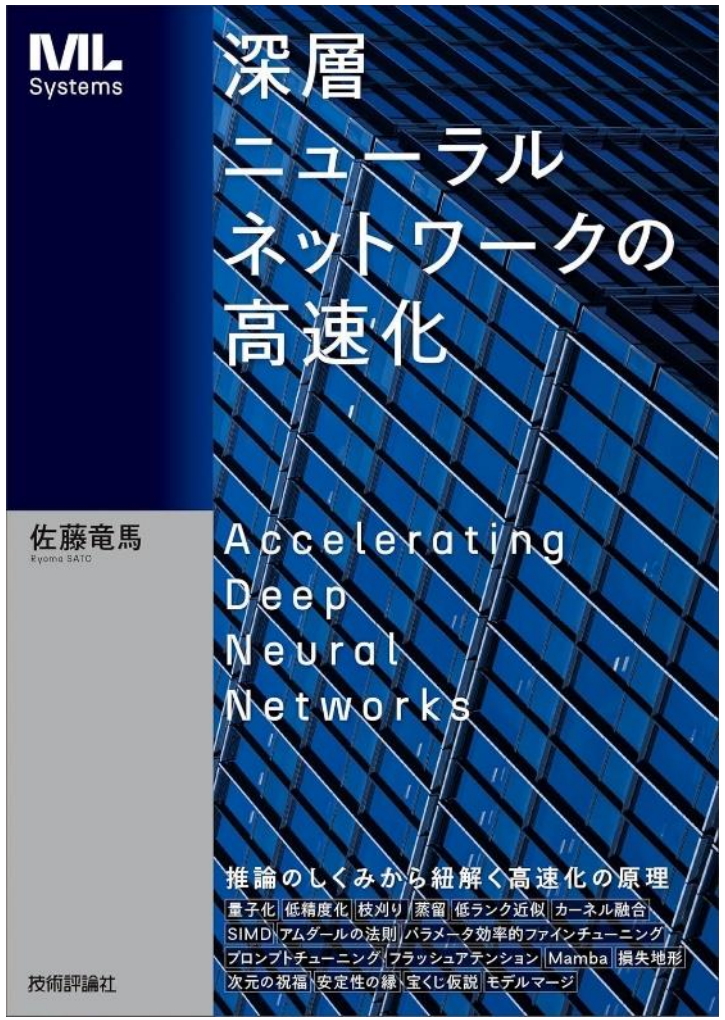
損失地形によりニューラルネットワークを深く理解できる

- 損失地形は多くのことを教えてくれる
  平坦解は摂動に対して頑健・汎化しやすい・高速化しやすい
- 盆地は推論方式に対応している（文法ベース・単語ベース）
- モデルマージでより高性能なモデルを作成できる

Take Home Message

一点の損失値だけでなく、周りの地形も観察しよう

深層ニューラルネットワークの高速化、好評発売中！

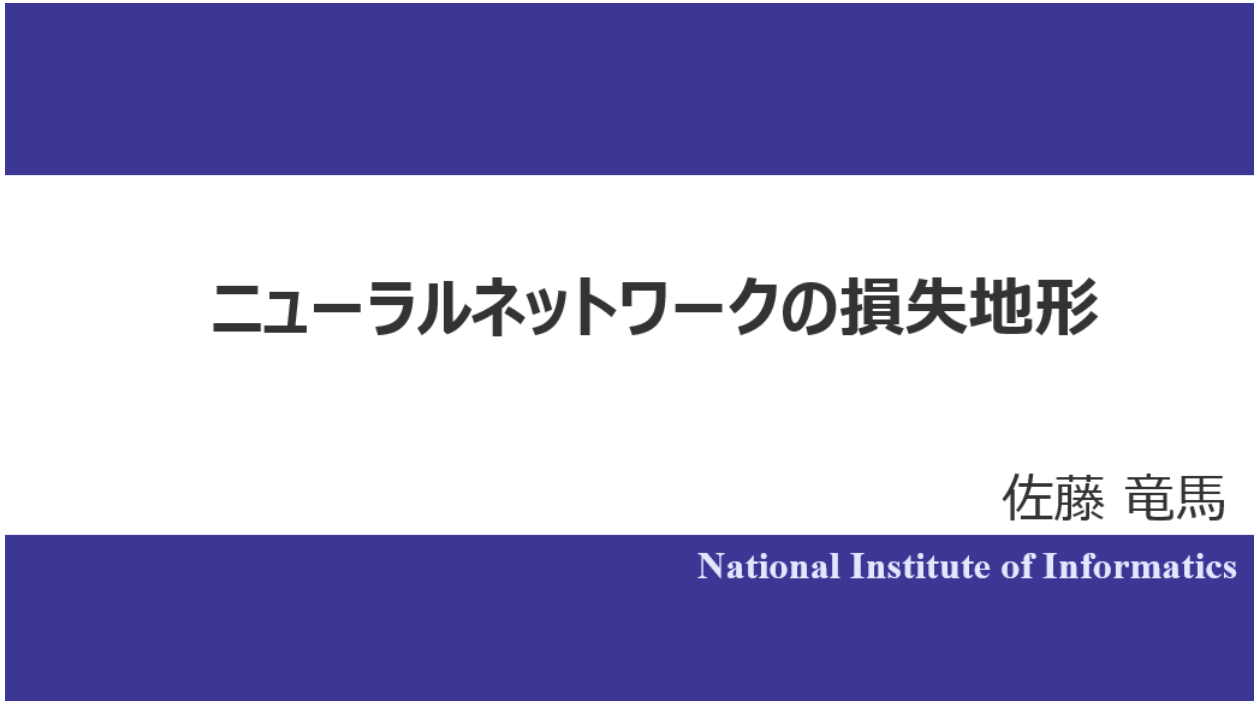


https://www.amazon.co.jp/d/4297143097

好評発売中！

今日の話のように、高速化にからめて
深層モデルにまつわる面白い話が
盛りだくさん！もちろん実用性もあり

本スライドも公開中



面白いと思ったらぜひ感想を付けてシェアしてくださいね

https://speakerdeck.com/joisino/landscape