卒業論文

大規模言語モデルにおける ターミナルアトラクタを組み込んだ 動的適応プルーニング手法

Dynamic Adaptive Pruning Method
Incorporating Terminal Attractors
in Large-Scale Language Models

富山県立大学 工学部 情報システム工学科
2220029 佐藤力
指導教員 António Oliveira Nzinga René 准教授
提出年月: 令和8年2026年2月

目 次

図一覧

表一覧

記号一覧

以下に本論文において用いられる用語と記号の対応表を示す.

用語	記号
MultiHead Attention における単語ベクトルに W^Q を掛けたもの	Q
MultiHead Attention における単語ベクトルに W^K を掛けたもの	K
MultiHead Attention における単語ベクトルに $W\ V$ を掛けたもの	V
MultiHead Attention における次元数	v^T
MultiHead Attention における訓練される重み行列	W^O
Positional Encoding における位置エンベディングの次元数	i
Positional Encoding における埋め込みベクトルの次元数	d_{model}
Siamese Network における埋め込み表現の次元	n
Siamese Network におけるラベルの数	k
UMAP における他の点 x_i の近傍に x_j が属する強さ	$v_{j\|i}$
UMAP における他の点 y_i の近傍に y_j が属する強さ	$w_{j\|i}$
UMAP における他の点 x_j が属する強さ	$v_{j\|i}$
UMAP における点 x_i と x_j の距離	r_{ij}
UMAP における点 y_i と y_j の距離	d_{ij}
UMAP における点 x_i に対して，k 近傍の集合	K_i
UMAP における点の疎密に対応するための変数	σ_i
k-menas における n 個の個体	$\vec{x_i} = (x_{i1}, \ldots, x_{iD})$
k-menas における n 個の個体の集合	x
k-menas における K 個の重なりの無いクラス	$X_k, k = 1, \ldots, K$
k-menas におけるクラスタの中心	$\vec{c_k}$
k-menas における $X_k^{(t)}$ に属する個体の数	$n_k^{(t)}$
k-menas における $X_k^{(t)}$ の $(K+1)$ 回目のクラスタの中心	$\vec{c_k}^{(t+1)}$
シルエット分析における各データのサンプル	$x^{(i)}$
シルエット分析における $x^{(i)}$ が属するクラスタ	C_{in}
シルエット分析における $x^{(i)}$ に最も近いクラスタ	C_{near}

第1章

はじめに

§ 1.1　本研究の背景

近年、深層学習（Deep Learning）を中心とした人工知能（AI）技術の発展は目覚ましく、画像認識、自然言語処理、音声認識など多岐にわたる分野で人間を凌駕する性能を示している. 特に、大規模言語モデル（Large Language Models: LLM）の登場は、AI の汎用性を飛躍的に高めた. しかし、これらのモデルが高性能化する一方で、モデルのパラメータ数は指数関数的に増大している. 例えば、近年の代表的な LLM では数千億から数兆パラメータを有することも珍しくなく、その学習および推論には膨大な計算資源と電力が必要となる.

この「モデルの巨大化」は、実用面において深刻な課題を突きつけている. 第一に、推論時のレイテンシ（遅延）の問題である. 自動運転やリアルタイム翻訳など、即時性が求められるアプリケーションにおいて、巨大なモデルの計算負荷はシステムの応答速度を低下させる要因となる. 第二に、エッジデバイスへの実装の困難さである. スマートフォンや IoT デバイスのような計算資源やメモリ容量、バッテリーに制約のある環境では、巨大なモデルをそのまま動作させることは現実的ではない. 第三に、環境負荷の問題である.AI モデルの学習と運用に伴う消費電力の増大は、持続可能な開発目標（SDGs）の観点からも無視できない問題となっている.

これらの課題に対処するため、モデルの軽量化技術が活発に研究されている. その代表的な手法の一つが「プルーニング（Pruning：枝刈り）」である. プルーニングは、ニューラルネットワークのパラメータのうち、推論精度への寄与が小さい、あるいは冗長であると判断された結合を削除（ゼロ化）することで、モデルのサイズを圧縮し、計算コストを削減する技術である. 生物学的な脳のシナプス結合が、成長過程で不要な結合を淘汰し効率化していくプロセスに着想を得ている.

従来のプルーニング手法の多くは、学習済みのモデルに対して重みの絶対値が小さいものを削除し、その後、精度を回復させるために再学習（Fine-tuning）を行うという手順を踏む. しかし、この「再学習」のプロセスには多大な時間と計算コストがかかるという問題がある. また、どの重みを削除すべきかという基準（サリエンシー）の決定や、削除後のネットワーク構造が最適である保証はなく、試行錯誤的な要素が強い. 特に、モデルの構造を変化させることは、学習のダイナミクス（収束挙動）を不安定にさせるリスクを伴う.

一方で、ニューラルネットワークの学習則やダイナミクスそのものを改良するアプローチとして、非線形力学系における「ターミナルアトラクタ（Terminal Attractor）」の概念

が注目されている. 通常のニューラルネットワークの学習（勾配降下法など）は、リプシッツ条件を満たす力学系に基づいており、誤差がゼロになる平衡点（アトラクタ）へは無限の時間をかけて漸近的に収束する. つまり、理論上は有限時間内に完全に学習が終了することはない . これに対し、ターミナルアトラクタはリプシッツ条件を特異点において破ることで、解軌道が有限時間内に平衡点に到達することを数学的に保証する概念である . 先行研究として、動径基底関数ネットワーク（RadialBasis Function Network: RBFN）に対し、このターミナルアトラクタを導入することで、学習回数の上限を指定し、かつ高速に収束させる手法が提案されている . また、不要なニューロンを削除する「競合（Competition）」メカニズムと組み合わせることで、構造の最適化と学習の高速化を同時に図る試みもなされてきた .

§ 1.2　本研究の目的

本研究の主たる目的は、ターミナルアトラクタの概念を導入した新しいニューラルネットワークのプルーニング手法を提案し、その有効性を実証することである. 具体的には、従来の重みの大きさのみに基づく静的なプルーニングではなく、学習のダイナミクスを考慮した動的なプルーニングアルゴリズムを構築する.

従来の勾配法に基づく学習では、誤差関数が平坦な領域において学習が停滞しやすく、これがプルーニング後の精度回復を遅らせる要因となっていた. 本研究では、誤差関数の勾配情報にターミナルアトラクタ項を付加することで、特異点への求心力を高め、プルーニングによって生じた精度の劣化を有限時間内に、かつ急速に回復させるメカニズムを実現する. これにより、従来手法と比較して、少ない学習回数で、同等以上の精度を持つ軽量モデルを構築することを目指す.

本研究の第二の目的は、提案手法を大規模言語モデルのファインチューニング（Fine-tuning）に適用し、その実用性を検証することである. 第 3 章で詳述するように、LLM のファインチューニングは特定のタスクに適応させるために重要であるが、全パラメータを更新するには莫大な計算コストがかかる. 近年では LoRA（Low-Rank Adaptation）などのパラメータ効率の良い学習手法（PEFT）が提案されているが、本研究ではこれらとは異なるアプローチとして、ターミナルアトラクタを用いたプルーニングによる効率化を模索する. 具体的には、事前学習済みのモデルに対し、タスク適応に必要な重要な重みのみを残しつつ、ターミナルアトラクタの効果によって素早く最適解へ収束させることで、「学習時間の短縮」と「推論モデルの軽量化」の双方を同時に達成することを目指す. これは、先行研究である RBFN における競合学習や複製メカニズム の思想を、現代の Deep Learning の文脈で再解釈し、高次元パラメータ空間における最適化問題として拡張する試みでもある.

§ 1.3　本論文の概要

本論文は次のように構成される.

第 1 章 本研究の背景と目的について説明する.

第 2 章 ニューラルネットワークにおけるプルーニングとターミナルアトラクタについてまとめる.

第 3 章 大規模言語モデルとファインチューニングについてまとめる.

第 4 章 提案手法について説明する.

第 5 章 第 4 章で述べた手法で，数値実験結果並びに考察を行う.

第 6 章 本論文における前章までの内容をまとめつつ，本研究で実現できたことと今後の展望について述べる.

第2章

ニューラルネットワークにおけるプルーニングとターミナルアトラクタ

§ 2.1　モデルの軽量化手法としてのプルーニング

（1）現代のニューラルネットワークにおける課題と軽量化の必要性

近年の Deep Learning の発展は目覚ましく、画像認識、自然言語処理、音声認識といった多岐にわたるタスクにおいて、人間を凌駕する性能を達成している. 特に、Transformer アーキテクチャの登場以降、モデルの大規模化（スケーリング）は性能向上のための最も確実なアプローチとなり、パラメータ数が数千億から数兆のオーダーに達する大規模言語モデルも珍しくない状況となっている. しかし、このようなモデルの巨大化は、計算資源の増大、推論レイテンシの悪化、および消費電力の増加という深刻な課題を突きつけている.

これらの課題に対処するため、モデルの精度を維持しつつ、パラメータ数や演算量を削減する「モデル軽量化」技術の研究が活発化している. 軽量化のアプローチは多岐にわたるが、主要なものとして、パラメータの数値表現ビット数を減らす「量子化」、巨大な教師モデルの知識を小さな生徒モデルに転移させる「知識蒸留」、そして本研究の主題である、不要な結合やニューロンを物理的に削除する「プルーニング」が挙げられる.

(2) プルーニングの定義と生物学的背景

プルーニング（枝刈り）とは、学習済みのニューラルネットワーク、あるいは学習過程にあるネットワークから、推論精度への寄与が低い冗長なパラメータを選択的に削除し、スパース（疎）な構造へと変換する技術である. 数学的には、ニューラルネットワークの重みベクトル $\mathbf{w}$ に対し、L_0 ノルム（非ゼロ要素の数）を制約条件として、損失関数 $E(\mathbf{w})$ を最小化する最適化問題として定式化できる.

$$\min_{\mathbf{w}} E(\mathbf{w}) \quad \text{s.t.} \quad \|\mathbf{w}\|_0 \leq \kappa\,(\mathbf{x}) \tag{2.1}$$

ここで、κ は目標とする非ゼロパラメータ数である. この工学的なアプローチは、神経科学における「シナプス剪定（Synaptic Pruning）」に着想を得ている. 人間の脳の発達過程において、乳幼児期には過剰なシナプス結合が形成されるが、成長に伴い、外部環境からの刺激や学習を通じて使用頻度の低い結合が淘汰され、効率的な神経回路網が構築される. この生物学的な「適者生存」のメカニズムは、限られたエネルギーとスペースで高度な知能を実現するための合理的な戦略であり、人工ニューラルネットワークにおいても同様の効率化が可能であることが示唆されている.

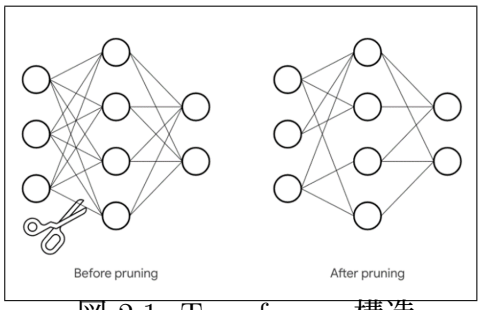

図 2.1: Transformer 構造

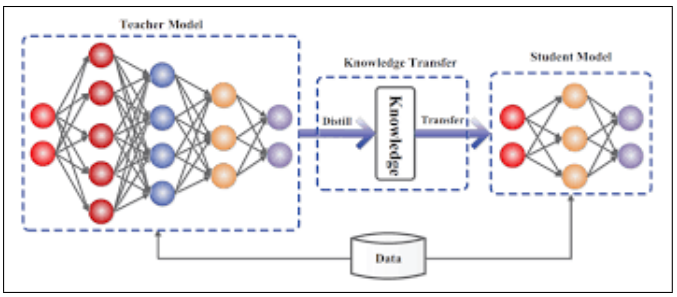

図 2.2: UMAP による次元圧縮

(3) 深層学習モデルにおける軽量化技術の概観

- 量子化

  ニューラルネットワークの重みや活性化関数は、通常 32 ビットの浮動小数点数（FP32）で表現される. 量子化は、これを 16 ビット（FP16）、8 ビット（INT8）、あるいはそれ以下の低ビット表現に変換する手法である. 表現精度を下げることで、モデルのファイルサイズを劇的に縮小し、メモリ帯域幅の節約と演算の高速化を実現する. しかし、極端なビット削減は情報量の欠落を招き、精度の低下を引き起こすトレードオフが存在する.

- 知識の蒸留

  大規模で高性能なモデル（教師モデル）の入出力関係や中間層の特徴量を、小規模なモデル（生徒モデル）に学習させる手法である. 単に正解ラベル（Hard Target）を学習するだけでなく、教師モデルが出力する確率分布（Soft Target）を模倣させることで、小規模モデルであっても教師モデルに近い汎化性能を獲得させることを目的とする.

- 低ランク近似

  重み行列が冗長であり、実際には低いランクで近似可能であるという仮定に基づく手法である. 特異値分解（SVD）などを用いて巨大な行列を複数の小さな行列の積に分解することで、パラメータ数を削減する.

(4) プルーニング手法の分類

プルーニングは、その削除単位によって「非構造化プルーニング」と「構造化プルーニング」に大別される.

- 非構造化プルーニング（Unstructured Pruning）

  個々の重みパラメータ w_{ij} を独立して評価し、削除対象とする手法である. モデルの構造的な制約を受けないため、理論上は最も不要なパラメータを精密に除去でき、高い圧縮率と精度の維持を両立しやすい. しかし、結果として得られる重み行列はランダムな疎行列（Sparse Matrix）となり、メモリ上の配置が不連続となるため、GPU などの並列演算ハードウェアにおいて計算効率を向上させることが難しいという実装上の課題がある.

- 構造化プルーニング（Structured Pruning）

  フィルタ（カーネル）、チャネル、あるいはニューロン全体といった、特定の構造単位で削除を行う手法である. 例えば、畳み込み層における特定のフィルタを丸ごと削除すれば、出力特徴マップの枚数が減り、後続の演算量も削減される. この手法は、行列の次元そのものを縮小するため、専用のライブラリやハードウェアを用いずとも、汎用的な環境で推論速度の向上（Speed-up）を享受できる利点がある. 反面、非構造化プルーニングに比べて削除の自由度が低く、同じ圧縮率での精度低下が大きくなる傾向がある.

（5）サリエンシー（重要度）の決定基準

「どのパラメータを削除すべきか」を決定するための指標はサリエンシー（Saliency）と呼ばれ、プルーニングの性能を左右する最も重要な要素である.

- Magnitude Pruning（絶対値基準）

  最も直感的かつ広く用いられている基準であり、重みの絶対値 $|w|$ をその重要度とみなす.「絶対値が小さい重みは、入力信号に対して小さな応答しか返さないため、出力への影響も軽微である」という仮定に基づく. 計算コストが極めて低く、多くの場合で十分な性能を発揮する.

- 勾配・ヘッセ行列に基づく基準

  損失関数のテイラー展開に基づき、各パラメータの重要度を解析的に評価するアプローチが存在する. これは、特定の重みを削除（ゼロ化）した際に生じる損失関数の変化量を、数理的に近似・推定するものである. 代表的な手法として、一次微分（勾配）の情報のみを用いる手法や、二次微分（ヘッセ行列）の情報まで考慮する Optimal Brain Damage (OBD) や Optimal Brain Surgeon (OBS) などが知られている. これらは単純な重みの絶対値を基準とする場合と比較して、より理論的かつ精緻なパラメータ選定が可能であるという利点を持つ.

（6）プルーニング後の再学習における課題

プルーニングを実行すると、ネットワークのパラメータが欠落するため、一時的にモデルの表現力が低下し、推論精度（Accuracy）やパープレキシティ（Perplexity）が悪化する. この劣化を回復させるためには、残されたパラメータを用いて再度学習を行う「ファインチューニング」が不可欠である.

しかし、プルーニングによってスパース化されたネットワークは、最適化のランドスケープ（誤差曲面）がいびつになり、局所解（Local Minima）やプラトー（Plateau）に陥りやすくなる. その結果、通常の勾配降下法を用いた再学習では、精度の回復に多くのエポック数を要したり、あるいは元の精度まで回復しきれなかったりするケースが散見される. 特に大規模言語モデルのファインチューニングにおいては、学習コストの増大は致命的である. したがって、プルーニング後の再学習を、いかに「高速」かつ「確実」に収束させるかが、実用上の大きな未解決問題となっている. 本研究では、この課題に対する解として、次節で述べる「ターミナルアトラクタ」を導入する.

§ 2.2　有限時間で収束するターミナルアトラクタ

まず，微分方程式における初期値問題に対する解の一意性を保証する条件であるリプシッツ条件について説明する．

リプシッツ条件

$f(t,x)$ を，閉区間 $\Omega = (t,x) \mid |t-t_0| \leqq a, |x-x_0| \leqq b$ で定義された関数とする．ある定数 L があり，Ω 内の 2 点 $(t,x), (t,y)$ において，

$$|f(t,x) - f(t,y)| \leqq L|x-y| \tag{2.2}$$

となるとき，f はリプシッツ条件を満たす．

このリプシッツ条件が満たされると，それぞれの初期値問題に対して一意な解が存在し，その解の軌道は漸近的に平衡点に近づく．すなわち，軌道は平衡点に近づくだけで，有限時間内で平衡点に到達できない．これは CRBFN に置き換えると，競合に負けたシナプス結合荷重は有限時間で 0 に到達することができないということである．

そこで，このリプシッツ条件を破るという考えに基づいて解の一意性を破ることにより，有限時間内でニューラルネットワークが平衡点に収束することを示した [?] [?]．このような安定な平衡点をターミナルアトラクタと呼ぶ．このターミナルアトラクタの概念を CRBFN に適用することで，収束時間の上限値を指定できるようにした．ここでは，時刻 t^* で平衡解へ収束できるように修正されたシナプス可塑性方程式を導出する．

いま，正定関数 $V(\mathbf{w})$ として

$$V(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left\{\eta(\mathbf{x}_i) - s(\mathbf{x}_i)\right\}^2 \tag{2.3}$$

を定義する．正定関数 $V(\mathbf{w})$ はシナプス後発火頻度 $\eta(\mathbf{x}_i)$ と神経伝達物質放出量 $s(\mathbf{x}_i)$ の差を表す指標である．シナプス後発火頻度 $\eta(\mathbf{x}_i)$ が時間に依存しないと仮定すると，その時間変化が

$$\begin{aligned}
\frac{dV(\mathbf{w})}{dt} &= \sum_{j=1}^{M}\frac{\partial V(\mathbf{w})}{\partial w_j}\frac{dw_j}{dt} \\
&= -\sum_{j=1}^{M}\left[\sum_{i=1}^{N}\eta(\mathbf{x}_i)\,\xi_j(\mathbf{x}_i) - \sum_{h=1}^{M}\sum_{i=1}^{N}\xi_j(\mathbf{x}_i)\,\xi_h(\mathbf{x}_i)\,w_h\right]\frac{dw_j}{dt} \\
&= -\sum_{j=1}^{M}w_j\left(\alpha_j - \sum_{h=1}^{M}\gamma_{jh}w_h\right)^2 \leq 0
\end{aligned} \tag{2.4}$$

となるため，正定関数 $V(\mathbf{w})$ が Lyapunov 関数となることがわかる．さらに，$V(\mathbf{w}) > 0$ であり $\frac{dV(\mathbf{w})}{dt} \leq 0$ であることから，このシステムは漸近安定であるということもわかる．

ここで，重み w_j の時間微分を

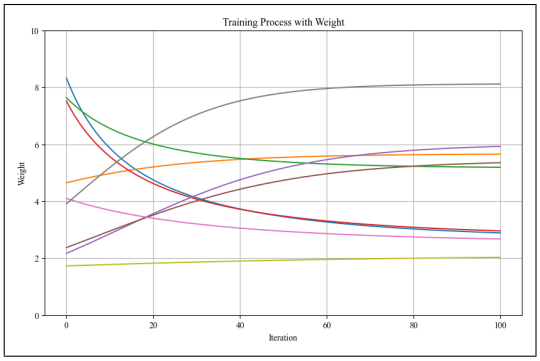

図 2.3: ターミナルアトラクタなし

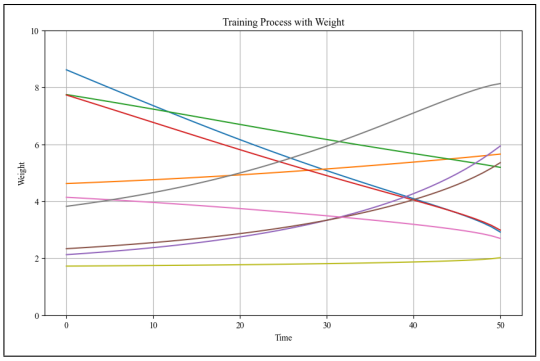

図 2.4: ターミナルアトラクタあり

$$\frac{dw_j}{dt} = -\Delta \frac{V(\mathbf{w})^c}{\sum_{j=1}^{M} \left(\frac{\partial V(\mathbf{w})}{\partial w_j} \right)^2} \frac{\partial V(\mathbf{w})}{\partial w_j} \tag{2.5}$$

とおくと，$V(\mathbf{w})$ の時間微分は

$$\frac{dV(\mathbf{w})}{dt} = \sum_{j=1}^{M} \left\{ \frac{\partial V(\mathbf{w})}{\partial w_j} \frac{dw_j}{dt} \right\} = -\Delta V(\mathbf{w})^c \leq 0 \tag{2.6}$$

となる．ただし，c は $0 < c < 1$ を満たす．$V(\mathbf{w})$ は時間とともに単調減少し，平衡点は漸近安定となることがわかる．このときの収束時間 t^* は，

$$\begin{aligned} t^* &= \int_0^{t^*} dt = \int_{V(\mathbf{w}_0)}^{V(\mathbf{w}_{t^*})} dV(\mathbf{w}) \frac{dt}{dV(\mathbf{w})} \\ &= \frac{V(\mathbf{w}_0)^{1-c} - V(\mathbf{w}_{t^*})^{1-c}}{\Delta(1-c)} \leq \frac{V(\mathbf{w}_0)^{1-c}}{\Delta(1-c)} \end{aligned} \tag{2.7}$$

で与えられ，有限時間内で収束することがわかる．$V(\mathbf{w}_0)$ は重みの初期値を用いて計算した Lyapunov 関数 $V(\mathbf{w})$ の初期値で，$V(\mathbf{w}_{t^*})$ は平衡点での $V(\mathbf{w})$ の値である．$V(\mathbf{w}_{t^*}) = 0$ の場合，式 (2.7) の等号が成立する．そこで学習率 Δ を，

$$\Delta = \frac{t^*(1-c)}{V(\mathbf{w}_0)^{1-c}} \tag{2.8}$$

とすると収束時間を指定できる．このターミナルアトラクタの概念を CRBFN における重みの学習に適用することで，初期状態において基底関数の数が多い場合でも従来の更新則よりも速く学習を終了させることが可能となる．

また，[?] [?] で提案されているターミナルアトラクタを基に，違う形で提案されたターミナルアトラクタ [?] も存在する．本研究では [?] で提案されているターミナルアトラクタを用いて基底関数の重みの学習を行う．

まず，望ましい時刻 t^* で収束するシナプス結合荷重の時間変化を Lyapunov 関数を用いて規定する．そこで，Lyapunov 関数の時間変化を

表 2.1: ターミナルアトラクタ適用前後の数値比較

重み	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
適用前	2.88	5.65	5.19	2.95	5.92	5.35	2.67	8.11	2.03
適用後	2.91	5.65	5.18	2.98	5.93	5.35	2.69	8.12	2.01

$$\frac{dV(\mathbf{w})}{dt} = -\frac{V(\mathbf{w}_0)^R V(\mathbf{w})^{\frac{1}{r}}}{Rt^*} \tag{2.9}$$

で定義する．ここで，r は 1 ではない任意の奇数であり，$R = \frac{(r-1)}{r}$ である．このような定義が可能になったのは，Lyapunov 関数が導出され，望ましい出力が動径基底関数を定数倍して足し合わせることで実現できる特別の場合を考えているからである．シナプス可塑性方程式は

$$\frac{dw_j}{dt} = \Delta \left(\alpha_j - \sum_{h=1}^{M} \gamma_{jh} w_h \right) w_j \tag{2.10}$$

とすることができる．このとき，Lyapunov 関数の時間変化は

$$\frac{dV(\mathbf{w})}{dt} = -\Delta \sum_{j=1}^{M} w_j \left(\alpha_j - \sum_{h=1}^{M} \gamma_{jh} w_h \right)^2 \tag{2.11}$$

となる．ここで，Δ は式 (2.9) と式 (2.11) から決定することが可能で，

$$\Delta = \frac{1}{\sum_{j=1}^{M} w_j \left(\alpha_j - \sum_{h=1}^{M} \gamma_{jh} w_h \right)^2} \frac{V(\mathbf{w}_0)^R V(\mathbf{w})^{\frac{1}{r}}}{Rt^*} \tag{2.12}$$

と導出される．以上より，望ましい時刻 t^* で平衡解へ収束するシナプス可塑性方程式を

$$\frac{dw_j}{dt} = \frac{\left(\alpha_j - \sum_{h=1}^{M} \gamma_{jh} w_h \right) w_j}{\sum_{j=1}^{M} w_j \left(\alpha_j - \sum_{h=1}^{M} \gamma_{jh} w_h \right)^2} \frac{V(\mathbf{w}_0)^R V(\mathbf{w})^{\frac{1}{r}}}{Rt^*} \tag{2.13}$$

で定義することができる．

図 2.3，図 2.4 にターミナルアトラクタ適用前後の重み w の学習過程を，表 5.2 に実際の数値による比較を示す．ターミナルアトラクタ適用前では重み w_1〜w_9 の数値を得るのに 100 回の学習を必要としていたが，適用後の数値を見ると指定した 50 回の学習の時点で適用前と同様の結果を得られていることがわかる．

§ 2.3　ターミナルアトラクタの学習測への適用

(1) 多層ネットワークへの拡張とエネルギー関数の定義

前節で定義したターミナルアトラクタ（TA）の概念を、実際の多層ニューラルネットワークの学習則に組み込む. 標準的なバックプロパゲーション（誤差逆伝播法）では、各層の重み w に関する損失関数（エネルギー関数） E の勾配を用い、漸近的に解を探索する. 本研究では、この勾配情報に対し、TA の「有限時間収束性」を付与した新しい更新則を適用する. まず、ネットワーク全体の出力誤差を次のような二次形式のリアプノフ関数 $V(w)$ と定義する.

$$V(w) = \frac{1}{2}\sum_{p=1}^{P} \|\mathbf{y}_p - \mathbf{o}_p(w)\|^2$$

ここで、P は学習データのサンプル数、$\mathbf{y}_p$ は教師ベクトル、$\mathbf{o}_p$ は現在の重み w におけるネットワークの出力ベクトルである. 学習の目的は、$V(w) \to 0$ となる重み集合 w^* を最短時間で見出すことにある.

(2) TA 項を導入した重み更新アルゴリズム

先行研究における動径基底関数ネットワーク（RBFN）での知見に基づき、任意のパラメータ w_j（重みやバイアス）に対する時間微分方程式を次のように構成する.

$$\frac{dw_j}{dt} = -\Delta \frac{V(w)^\beta}{\sum_k \left(\frac{\partial V}{\partial w_k}\right)^2 + \epsilon} \frac{\partial V}{\partial w_j} \quad \cdots (2.6)$$

ここで、Δ は学習率、β（$0 < \beta < 1$）は TA の特異性を制御するパラメータ、ϵ は分母がゼロになることを防ぐための微小な正定数である. 式 (2.6) の分母にある項 $\sum_k (\partial V/\partial w_k)^2$ は、全パラメータ空間における勾配のノルムの二乗を意味する. この項で正規化を行うことにより、勾配が極めて小さいプラトー領域においても、分子の $V(w)^\beta$ による強力な吸引力が維持され、解軌道が停滞することなく誤差ゼロの地点（アトラクタ）へ向かうことが可能となる.

(3) プルーニング後の構造的制約下におけるダイナミクス

本研究の独自性は、この TA 学習則をプルーニング後の「疎（Sparse）なネットワーク」に適用する点にある. プルーニングによって一部の重みが固定（$w = 0$）された場合、学習に寄与するパラメータの自由度は減少する. 従来の勾配法では、この制約下で複雑な誤差曲面を探索すると収束が極めて不安定になるが、TA を導入することで以下の利点が得られる.

- 特異点への強制的収束

  プルーニングにより表現力が制限された状態でも、TA は残されたパラメータを誤差最小化のために「最短経路」で駆動する.

- 適者生存の加速

  プルーニングによって生き残った重要な重みに対して、TA が優先的に学習資源（更新量）を割り当てる効果が期待できる.

- 計算効率の最適化

  式 (2.4) で導出した有限時間 t_f を用いることで、プルーニング後の微調整（Fine-tuning）に必要なステップ数を理論的に予測し、無駄な反復計算を排除できる.

(4) 実装における離散化処理

計算機上で動作させるため、式 (2.6) を離散化し、各ステップ n における重みの更新式を次のように定める.

$$w_j(n+1) = w_j(n) - \Delta \cdot \delta t \cdot \frac{V(w(n))^\beta}{\sum_k \left(\frac{\partial V}{\partial w_k}\right)^2 + \epsilon} \frac{\partial V}{\partial w_j}$$

ここで δt は微小時間ステップである. この離散化された更新則を用いることで、従来手法（SGD や Adam 等）を TA ベースの学習則へと置き換え、プルーニング後の再学習を高速化するアルゴリズムが完成する.

第3章

大規模言語モデルとファインチューニング

§ 3.1　大規模言語モデルの発展と現状

3.1.1 自然言語処理の変遷

自然言語処理（Natural Language Processing: NLP）の歴史は、計算機を用いて人間言語の曖昧性と複雑性を如何にモデル化するかという挑戦の連続であった. 初期のアプローチは、N-gram モデルに代表される統計的言語モデルが主流であり、単語の出現確率に基づいて次に来る単語を予測していた. しかし、この手法は文脈の長さに応じて計算量が指数関数的に増大するため、長い文脈を考慮することが困難であった. 2010 年代に入り、ニューラルネットワークを用いた言語モデル（Neural Language Models）が登場したことで状況は一変する. 特に、リカレントニューラルネットワーク（RNN）や、その改良版である LSTM（Long Short-Term Memory）は、可変長の系列データを扱うことができ、長期依存関係の学習において一定の成果を挙げた. しかし、これらのモデルは時系列データを逐次的に処理する構造上、計算の並列化が不可能であり、学習に膨大な時間を要するという致命的な欠点を抱えていた. また、系列が長くなるにつれて勾配消失問題が発生しやすく、文脈理解の範囲には限界があった. この停滞を打破したのが、2017 年に Vaswani らによって提案された「Transformer」アーキテクチャである.Transformer は、再帰構造を完全に排除し、「Attention（注意機構）」のみを用いることで、文中の全単語間の関係性を並列に計算することを可能にした. この革新により、従来とは桁違いの規模のデータセットを用いた学習が可能となり、現在の大規模言語モデルの基礎が築かれた.

3.1.2 スケーリング則と創発的特性

Transformer の登場以降、モデルのパラメータ数、学習データ量、そして計算量を増加させることで、言語モデルの性能が飛躍的に向上することが明らかになった.Kaplan ら（2020）は、これらの要素とモデルの損失（Test Loss）との間にべき乗則（Power Law）が成立することを示し、これを「スケーリング則（Scaling Laws）」と提唱した. この法則によれば、計算資源を投じれば投じるほど、モデルの精度は予測可能な形で向上し続けるとされる. この知見に基づき、Google の BERT（2018）、OpenAI の GPT シリーズ（2018-2023）、Meta の LLaMA（2023）など、パラメータ数が数十億（Billions）から数千億（Trillions）に達する巨大モデルが次々と開発された. 例えば、GPT-3 は 1750 億パラメータを有し、特定のタスクに対するファインチューニングを行わずとも、少数の例示のみでタスクをこなす「Few-shot Learning」能力を示した. さらに興味深い現象として、Wei ら（2022）は「創発

（Emergence）」という特性を報告している. これは、モデルの規模がある閾値を超えた瞬間に、それまでの小規模なモデルでは見られなかった高度な能力（例えば、複雑な算術演算、論理推論、多段階の思考プロセスなど）が突如として発現する現象である. この創発的特性こそが、現代の LLM が単なる「次単語予測器」を超え、汎用的な人工知能（AGI）への足がかりとして注目される最大の理由である.

3.1.3 計算資源の増大と「Green AI」への転換

しかし、モデルの巨大化は深刻な副作用をもたらしている. 第一に、学習コストの増大である. 最先端の LLM を一度学習させるためには、数千基の高性能 GPU を数ヶ月間稼働させる必要があり、その電力消費量は小規模な都市のそれに匹敵するとも言われる. これは環境負荷の観点から持続可能とは言い難く、近年では精度だけでなくエネルギー効率を重視する「Green AI」への転換が叫ばれている.

第二に、推論時のレイテンシ（遅延）とメモリ制約の問題である. 数千億パラメータのモデルを展開するには、テラバイト級の VRAM（ビデオメモリ）が必要となり、これを運用できるのは一部の巨大テック企業に限られる. この「AI の独占」を防ぎ、スマートフォンやエッジデバイス、あるいは一般的なオンプレミスサーバーで高度な言語モデルを動作させるためには、モデルのサイズを劇的に圧縮する技術が不可欠である.

効率化へのアプローチ

これらの課題に対し、本研究では「プルーニング」技術に着目する. モデル内の冗長なパラメータを削除することで、推論に必要な計算量とメモリ量を物理的に削減することが可能である. しかし、従来の単純なプルーニング手法を LLM に適用した場合、創発によって獲得された繊細な知識構造が破壊され、精度の回復に多大な再学習コストがかかるという問題があった.

そこで本研究では、第 2 章で述べた「ターミナルアトラクタ」の概念を導入する. プルーニングによって疎になったネットワークに対し、有限時間収束を保証する強力なダイナミクスを与えることで、失われた精度を最小限の計算ステップで回復させることを目指す. これは、巨大化の一途をたどる LLM 開発の流れに対し、「効率性」と「実用性」の観点から新たな解決策を提示するものである.

§ 3.2 Transformer アーキテクチャの基本構造

3.2.1 全体構造と並列処理の実現

本研究が対象とする大規模言語モデルの基盤となっているのは、Vaswani らによって 2017 年に提案された「Transformer」アーキテクチャである. 従来の RNN（Recurrent Neural Networks）が系列データを左から右へと逐次的に処理していたのに対し、Transformer は「Self-Attention（自己注意機構）」を用いることで、系列内の全トークン間の依存関係を並列に計算する構造を持つ.

Transformer のオリジナルの構成は、入力を処理するエンコーダ（Encoder）と、出力を生成するデコーダ（Decoder）から成る. しかし、GPT（Generative Pre-trained Transformer）

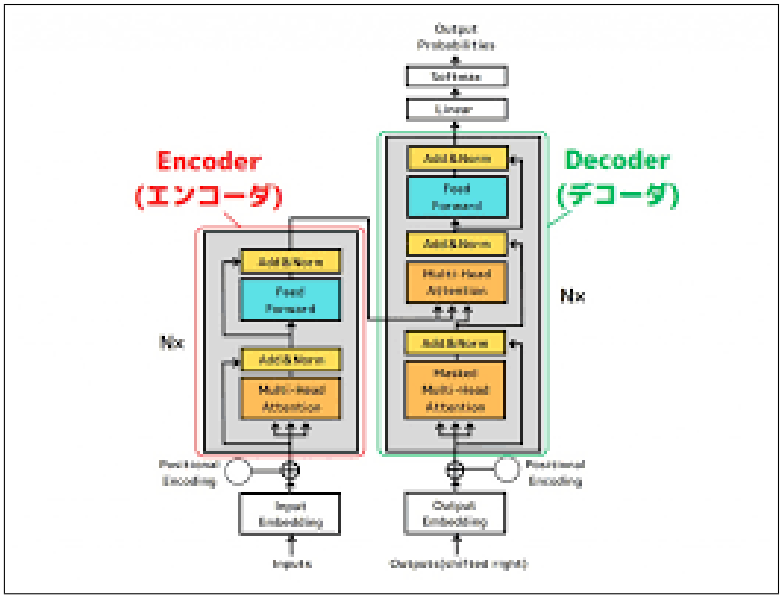

図 3.1: Transfomer

シリーズをはじめとする近年の主要な LLM は、デコーダ部分のみを積層した「Decoder-only」アーキテクチャを採用している. 本節では、この Decoder ブロックを構成する核心的なサブレイヤーについて詳述する.

3.2.2 Scaled Dot-Product Attention

Transformer の最も重要な構成要素は、入力トークン間の関連度（Attention Score）を計算する「Scaled Dot-Product Attention」である. 入力となる埋め込み表現行列 X に対し、学習可能な重み行列 W^Q, W^K, W^V を乗じることで、クエリ（Query: Q）、キー（Key: K）、バリュー（Value: V）の 3 つの行列を生成する.

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

これらを用いて、Attention の出力は次式で計算される.

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad \cdots (3.1)$$

ここで、d_k はキーベクトルの次元数である. 式 (3.1) において、QK^T は全てのトークンペア間の内積（類似度）を表す. これを $\sqrt{d_k}$ で除算する（スケーリング）理由は、次元数 d_k が大きくなった際に内積値が増大し、ソフトマックス関数の勾配が極端に小さくなる（勾配消失）のを防ぐためである. この機構により、モデルは文中の距離に関わらず、任意の単語間の関係性を直接的に捉えることが可能となる.

3.2.3 Multi-Head Attention

単一の Attention では、文脈の単一の側面しか捉えることができない可能性がある. そこで Transformer では、異なる部分空間（Subspace）の情報を並列に抽出するために「Multi-Head Attention」を採用している. モデルの次元 d_{model} を h 個のヘッドに分割し、各ヘッド

i $(i = 1, \ldots, h)$ で個別に Attention を計算する.

$$\mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad \cdots (3.2)$$

ここで、$W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d_{\mathrm{model}} \times d_k}$ は各ヘッド固有の射影行列である. 各ヘッドの出力は結合（Concat）され、最終的な線形変換が行われる.

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)W^O \quad \cdots (3.3)$$

$W^O \in \mathbb{R}^{d_{\mathrm{model}} \times d_{\mathrm{model}}}$ は出力射影行列である. 本研究におけるプルーニングは、主にこれらの巨大な重み行列群（W^Q, W^K, W^V, W^O）を対象に行われ、冗長なヘッドや結合を削減することでモデルの軽量化を図る.

3.2.4 Position-wise Feed-Forward Networks

Attention 層の後には、各位置（トークン）ごとに独立かつ同一に適用される全結合ニューラルネットワーク（Feed-Forward Networks）が接続される. これは 2 つの線形変換と、その間の非線形活性化関数（ReLU や GELU）から構成される.

$$\mathrm{FFN}(x) = \mathrm{Activation}(xW_1 + b_1)W_2 + b_2 \quad \cdots (3.4)$$

通常、中間層の次元 d_{ff} はモデル次元 d_{model} の 4 倍程度（例:$d_{\mathrm{model}} = 1024$ なら $d_{ff} = 4096$）に設定されるため、FFN は Transformer 全体のパラメータ数の約 2/3 を占める. したがって、この層に対するプルーニングは、モデル圧縮において極めて高い効果を持つ.

3.2.5 Positional Encoding と Residual Connections

Transformer は再帰構造を持たないため、単語の語順情報（位置情報）を何らかの形で入力に付与する必要がある. これが「Positional Encoding」であり、正弦波関数などを用いて各位置固有のベクトルを入力埋め込みに加算する. また、各サブレイヤー（Attention および FFN）の出力には、学習の安定化と勾配消失防止のために「残差結合（Residual Connection）」と「層正規化（Layer Normalization）」が適用される.

$$\mathrm{Output} = \mathrm{LayerNorm}(x + \mathrm{Sublayer}(x)) \quad \cdots (3.5)$$

この構造により、数百層に及ぶ深層モデルであっても安定した学習が可能となっている. 本研究で提案するターミナルアトラクタを用いた学習則は、これらの構造的特性を維持しつつ、プルーニングによって W がスパースになった状態での再最適化を加速させるものである.

§ 3.3　事前学習と転移学習のメカニズム

3.3.1 自己教師あり学習

大規模言語モデルの学習プロセスにおける最大の特徴は、人間によるラベル付け（アノテーション）を必要としない「自己教師あり学習（Self-Supervised Learning）」を採用している点にある. 従来の教師あり学習では、入力 x と正解ラベル y のペアが必要であったが、LLM の学習データとなるインターネット上のテキスト（Common Crawl, Wikipedia 等）はラベルを持たない. そこで、テキストデータそのものから「入力」と「正解」を自動的に生成する手法が用いられる. 具体的には、入力されたテキストの一部を隠し（マスクし）、その隠された部分を周囲の文脈から予測するタスクをモデルに課す. このプロセスを通じて、モデルは単語の意味、文法構造、さらには文脈に含まれる世界知識や論理的推論能力を、ニューラルネットワークのパラメータとして内在的に獲得する. この段階で獲得されたパラメータ θ_{pre} は、汎用的な言語理解能力を有しており、後述する転移学習の基盤となる.

3.3.2 事前学習の目的関数とモデルアーキテクチャ

自己教師あり学習の具体的なタスク設定は、モデルのアーキテクチャによって主に以下の二つに大別される. 本研究で扱う GPT 系列のモデルは、後者の因果的言語モデリングに基づいている.

- (1) マスク化言語モデリング (Masked Language Modeling: MLM)

  BERT（Bidirectional Encoder Representations from Transformers）などに代表される、Transformer のエンコーダ部分を用いた手法である. 入力系列 $X = \{x_1, x_2, \ldots, x_T\}$ の中からランダムにいくつかのトークンを特殊トークン [MASK] に置き換え、その元の単語を予測する. 目的関数 L_{MLM} は、マスクされたトークン集合 M に対する対数尤度の最大化（負の対数尤度の最小化）として定義される.

  $$L_{MLM}(\theta) = -\sum_{i \in M} \log P(x_i | X_{\setminus M}; \theta)$$

  ここで、$X_{\setminus M}$ はマスクされていない周囲の全トークンを表す.MLM は双方向（Bidirectional）の文脈を利用できるため、文の意味理解や分類タスクにおいて高い性能を発揮するが、文章生成には不向きである.

- (2) 因果的言語モデリング (Causal Language Modeling: CLM)

  GPT（Generative Pre-trained Transformer）シリーズで採用されている、Transformer のデコーダ部分を用いた手法である. 別名「自己回帰的言語モデリング（Autoregressive Language Modeling）」とも呼ばれる. ある時点 t の単語 x_t を、それ以前の単語列 $x_{<t} = \{x_1, \ldots, x_{t-1}\}$ のみを条件として予測するタスクである. 目的関数 L_{CLM} は、系列全体の結合確率を条件付き確率の積に分解し、その対数尤度を最大化することである.

  $$L_{CLM}(\theta) = -\sum_{t=1}^{T} \log P(x_t | x_1, \ldots, x_{t-1}; \theta)$$

  この定式化は、未来の情報（カンニング）を防ぐために、Attention 機構に「因果マスク（Causal Mask）」を適用することで実装される.CLM は、人間が文章を書くのと同様に左から右へ単語を紡ぐことができるため、文章生成タスクにおいて圧倒的な能

力を持つ. 本研究の実験対象である GPT-2 および WikiText データセットを用いた学習も、この CLM の枠組みで行われる.

3.3.3 転移学習とドメイン適応の理論

事前学習によって得られたパラメータ θ_{pre} を初期値とし、特定のタスクやドメインのデータセット D_{target} を用いてさらに学習を行うプロセスを「転移学習（Transfer Learning）」あるいは「ファインチューニング」と呼ぶ. 転移学習が有効である理論的根拠は、ディープニューラルネットワークの階層的な特徴抽出能力にある.

- 下位層

  文法や構文、単語の共起関係といった、言語に共通する普遍的な特徴を処理する.

- 上位層

  文脈の意味、意図、あるいは特定の専門知識といった、抽象度の高い情報を処理する.

事前学習済みのモデルは、既に下位層から中位層にかけて言語の普遍的な特徴を獲得しているため、ターゲットタスクのデータ量が少ない場合でも、ランダムな初期値から学習するより遥かに高速かつ高精度に収束することが知られている. これを数式で表現すると、事前学習はパラメータ空間における探索範囲を、言語として尤もらしい領域 Ω_{lang} に限定する役割を果たしており、ファインチューニングはその領域内からタスク最適解 θ^* を探索する局所最適化問題と見なせる.

3.3.4 現代的なファインチューニングの潮流：Instruction Tuning と RLHF（Reinforcement Learning from Human Feedback：人間のフィードバックからの強化学習）

- 指示チューニング (Instruction Tuning)

  事前学習モデルは「続きを書く」ことは得意だが、「要約せよ」「翻訳せよ」といったユーザーの指示（Instruction）に従うとは限らない. そこで、(指示, 理想的な回答) のペアを用いた教師あり学習を行うことで、モデルのアライメント（人間への追従性）を向上させる手法である.

- RLHF

  RLHF では、モデルの生成物に対する人間の選好（Preference）を報酬モデル（Reward Model）として学習させ、PPO（Proximal Policy Optimization）などの強化学習アルゴリズムを用いて LLM を微調整する.

  これらの手法はいずれも、巨大な事前学習モデル全体、あるいはその一部に対して追加の学習（重み更新）を行う必要がある. 特に RLHF などは計算コストが極めて高いため、本研究が提案する「プルーニングによるモデルの軽量化」と「ターミナルアトラクタによる学習の高速化」は、これらの高度なチューニングを一般的な計算環境で実現するための基盤技術として、極めて高い親和性と重要性を持つ.

第4章

提案手法

§ 4.1　提案手法の全体概要

本章では、前章までに述べた大規模言語モデルの計算コスト問題と、従来の学習則における収束速度の限界を同時に解決するための新規アルゴリズムを提案する. 本研究の提案手法は、「勾配情報に基づく動的プルーニング（Dynamic Pruning based on Gradient Information）」と「ターミナルアトラクタによる有限時間収束」を融合させたハイブリッドな学習システムである.

4.1.1 従来手法の限界と動的スパース化の必要性

従来、ニューラルネットワークの軽量化手法として広く用いられてきた「プルーニング」は、一般に「学習 → 剪定 → 再学習」という静的かつ多段階のプロセスを経て行われることが多かった. このアプローチは、Lottery Ticket Hypothesis（宝くじ仮説）などでその有効性が示唆されているものの、実用面においては致命的な二つの欠点を抱えている.

第一に、「再学習コストの増大」である. 一度学習が完了したモデルを破壊し、再度精度を回復させるプロセスは、実質的に二度手間であり、大規模モデルにおいては数百〜数千 GPU 時間の追加コストを意味する. これは、本研究が目指す「効率的な AI」の理念と矛盾する. 第二に、「構造の不可逆性」である. 一度削除された結合が、その後の学習過程で再び必要となった場合でも、静的なプルーニングでは復活させることが困難である. これにより、モデルは局所解に陥りやすくなり、最終的な汎化性能が犠牲になるケースが散見された.

これらの課題を克服するために、本研究では「動的スパース化（Dynamic Sparsity）」の概念を採用する. これは、学習プロセスが完了するのを待つのではなく、学習の進行と並行してリアルタイムにネットワーク構造を変化させるアプローチである. さらに、そこにターミナルアトラクタという強力な非線形ダイナミクスを導入することで、構造変化に伴う一時的な精度の揺らぎを瞬時に収束させ、常に最適な準安定状態を維持しながら学習を進めることを可能にする.

4.1.2 提案手法のコアコンセプト：構造とダイナミクスの同時最適化

本提案手法の最大の独自性は、ニューラルネットワークの学習を「パラメータ空間（連続値）の最適化」と「トポロジー空間（離散値）の最適化」の連成問題として捉え直した点にある. 通常の勾配法（SGD や Adam）は、固定されたネットワーク構造 G の上で、重みベクトル w を更新する（$w \leftarrow w - \eta \nabla E$）. 一方で、構造学習は重み w を固定し、最適

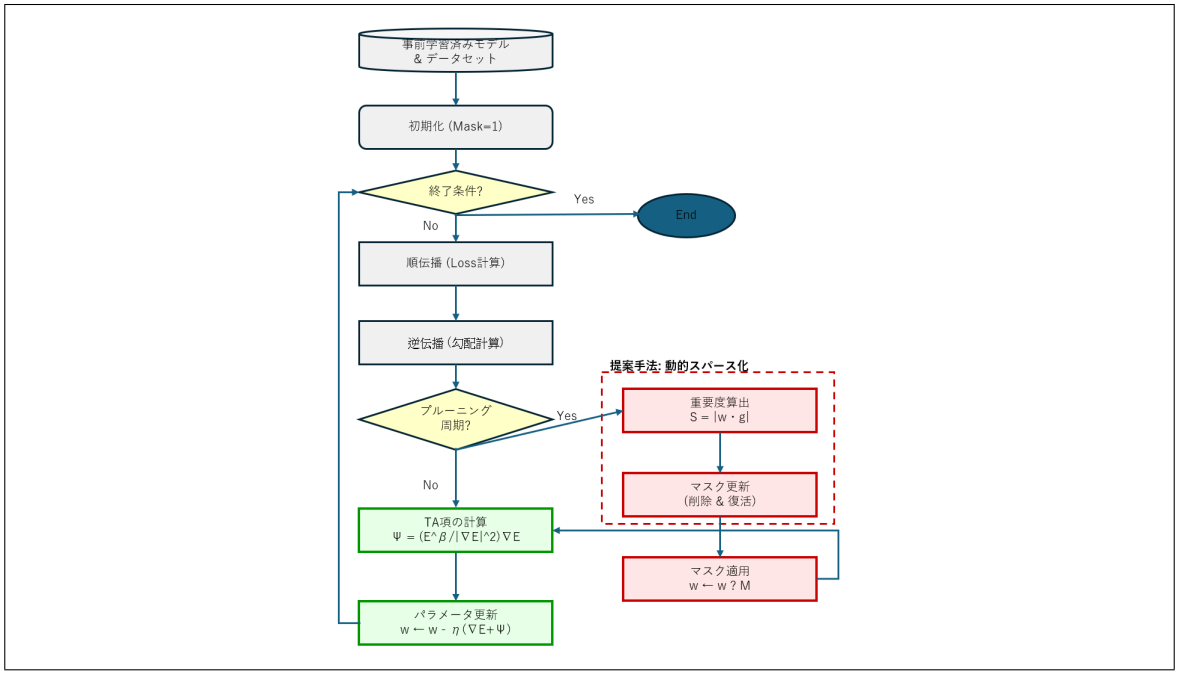

図 4.1: アルゴリズムフロー図

な構造 G を探索する. これらは互いに依存関係にありながら、従来は別々のフェーズで処理されていた. 本研究では、この二つを同一のタイムライン上で処理するために、以下のサイクルを構築した.

- 摂動としてのプルーニング重みの削除（$w_{ij} = 0$）を、システムに対する一種の「摂動（Perturbation)」あるいは「外乱」として定義する. 物理学的に言えば、これはエネルギー地形（Loss Landscape）における現在の安定点から、強制的に状態を遷移させる行為に等しい.
- 復元力としてのターミナルアトラクタ摂動によって不安定化したシステムを、ターミナルアトラクタの持つ「有限時間収束性」によって、新たな（より低次元の）安定平衡点へと強力に引き込む. 通常の学習則（リプシッツ連続な力学系）では、この引き込みに無限の時間を要するが、項を導入することで、摂動の直後から急速なリカバリーが可能となる.

この「破壊（プルーニング)」と「再生」の高速なサイクルこそが、脳の可塑性（Plasticity）を模倣した本手法の核心であり、これによりモデルは不要な贅肉を削ぎ落としつつ、常に高いパフォーマンスを維持し続けることができる.

4.1.3 アルゴリズムの全体処理フロー

提案システムの具体的な処理フローを詳述する. 本アルゴリズムは、事前学習済みの大規模言語モデル（Pre-trained LLM）を入力とし、軽量化および最適化されたスパースモデルを出力とする.

- Step 1: 初期化とデータセットの準備 (Initialization) まず、WikiText-2 などのコーパスを用い、学習データを K 分割する（交差検証のため）. 事前学習済みの重みパラメータ W_{init} をロードし、初期のスパースティ（疎性）は 0（全結合状態）からスタートする. ここで重要なのは、完全にランダムな初期値ではなく、ある程度言語構造を学習した状態からスタートすることで、TA が引き込むべき「大域的なアトラクタ（Global Attractor）」の盆地（Basin of Attraction）の中に初期状態を置くことである.

- Step 2: サリエンシー（重要度）の動的計算 (Dynamic Saliency Calculation) 学習の各ステップ（または特定の間隔）において、全パラメータの重要度 S_{ij} を計算する. 本研究では、単なる重みの絶対値（Magnitude）ではなく、損失関数 E に対する感度を考慮した「一次近似勾配スコア」を採用する.

$$S_{ij} = |w_{ij} \cdot \frac{\partial E}{\partial w_{ij}}| \, (\mathbf{x}) \tag{4.1}$$

  この指標は、「その重みを削除した場合に、損失関数がどれだけ変化するか」を近似的に表している. 勾配 $\frac{\partial E}{\partial w_{ij}}$ が大きい重みは、現在の学習において活発に更新されている（＝学習余地がある）ことを意味し、これを保存することは学習の停滞を防ぐ上で重要である.

- Step 3: 適応的プルーニングの実行 (Adaptive Pruning) 算出されたサリエンシーに基づき、下位 $R(t)$ 具体的には、損失の減少が停滞している（プラトーに達している）場合は、構造的な変化を促すために $R(t)$ を一時的に高め、逆に損失が激しく振動している場合は $R(t)$ を下げて系を安定させる. この適応的な制御により、「過剰な剪定による精度の崩壊」と「保守的な剪定による圧縮不足」のトレードオフを自動的に調整する.

- Step 4: ターミナルアトラクタ項による重み更新 (TA Weight Update) プルーニングマスク M を適用した後の重み $W' = W \odot M$ に対し、ターミナルアトラクタ則に基づいた更新を行う.

$$\Delta w = -\eta \left(\nabla E + \alpha \frac{E^{\beta}}{||\nabla E||^2} \nabla E \right) (\mathbf{x}) \tag{4.2}$$

  この更新式における第二項（TA 項）が、プルーニングによって生じた誤差 E の上昇を検知し、特異点（Singularity）の効果によって勾配ベクトルを増幅させる. これにより、残された結合（生き残った重み）に対して、「削除された重みの役割を補完せよ」という強力なバイアスがかかり、パラメータの再配置が急速に進む.

- Step 5: 評価とループ一定のエポック数ごとに検証データ（Validation Set）を用いてモデルの Perplexity（PPL）およびスパース率を評価する. 目標とするスパース率または精度に到達するまで、Step 2〜4 を反復する.

4.1.4 ターミナルアトラクタによる「意図的な」不安定性の収束

本手法を制御工学的な視点から解釈すると、プルーニングとはシステムに対する「構造的な外乱」であり、通常であればシステムを不安定化させる要因である. しかし、本研究ではこの不安定性を「探索の原動力」として積極的に利用する.

従来の確率的勾配降下法（SGD）では、局所解（Local Minima）にトラップされると、そこから抜け出すのには稀に起こる大きな勾配ノイズを待つしかなかった. 対して本手法では、プルーニングによって意図的にエネルギー地形を変化させ、局所解の底を浅くする（あるいは消滅させる）. その瞬間に、TA の強力な収束力が作用することで、システムは浅くなった局所解を飛び出し、より深く安定した、かつより低次元の（スパースな）解へと遷移していく.

この「局所解からの脱出（Escape）」と「新解への着地（Convergence）」のメカニズムこそが、提案手法が高い圧縮率と高精度を両立できる理論的根拠である.TA が存在しなければ、プルーニングによる外乱は単なる精度の劣化（発散）を招くだけだがターミナルアトラクタが存在することで、それは「より良い構造への進化」へと昇華されるのである.

§ 4.2　勾配情報に基づいた動的プルーニングアルゴリズム

ニューラルネットワークのプルーニングにおいて、最も重要な決定事項は「どのパラメータを削除するか（Selection Criteria）」と「いつ、どれだけ削除するか（Scheduling）」の 2 点に集約される. 本節では、本研究で提案する動的プルーニングの核心となるこれら二つのアルゴリズムについて詳述する.

4.2.1 損失関数のテイラー展開に基づくサリエンシーの定義

従来のプルーニング手法、特に Magnitude Pruning（重みの絶対値に基づく剪定）は、「絶対値が小さい重みは、ネットワークの出力に対する寄与が小さい」という経験則に基づいている. しかし、非線形性が強い深層学習モデルにおいては、この仮定が必ずしも成立しない場合がある. 例えば、活性化関数の飽和領域付近にある重みは、値が大きくても勾配がほぼゼロであり、学習に寄与していない可能性がある. 逆に、値がゼロに近い微小な重みであっても、その勾配が極めて大きく、次の更新で大きく成長しようとしている「芽」のようなパラメータも存在する. これらを一律に削除することは、学習のポテンシャルを摘む行為に等しい. そこで本研究では、パラメータの重要度（サリエンシー：Saliency）を、損失関数 E に対する感度（Sensitivity）として数学的に定義する. ある重み w を 0 にする（削除する）ことによる損失関数 E の変化量 ΔE は、摂動 $\Delta w = 0 - w = -w$ を与えたときの値として、テイラー展開により次のように近似できる.

$$\Delta E = E(w + \Delta w) - E(w) \approx \frac{\partial E}{\partial w}\Delta w + \frac{1}{2}\Delta w^T H \Delta w + O(||\Delta w||^3)\,(\mathbf{x}) \tag{4.3}$$

ここで、H はヘッセ行列（Hessian Matrix: 二階微分行列）である.LeCun らが提案した OBD（Optimal Brain Damage）や Hassibi らの OBS（Optimal Brain Surgeon）は、この第二項（ヘッセ行列）までを考慮する手法であるが、数億〜数千億パラメータを持つ LLM において、巨大なヘッセ行列の逆行列を計算することは計算量的に不可能である（$O(N^3)$ のコストがかかるため）. したがって、本研究では第一項のみを用いた一次近似（First-order

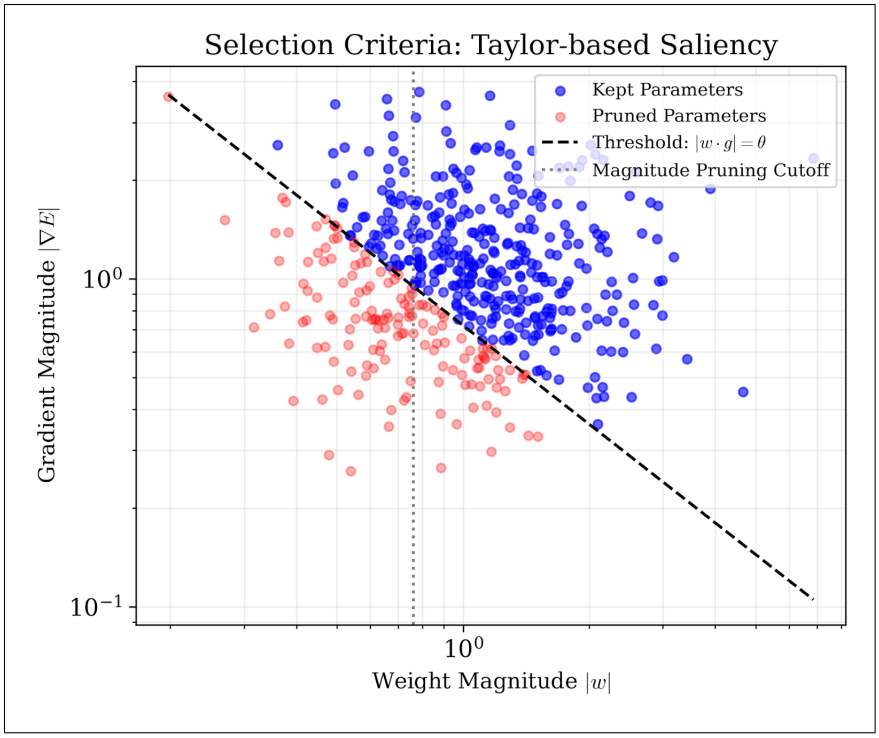

図 4.2: サリエンシーに基づく選別基準

Taylor Approximation）を採用する. プルーニングにおける摂動は $\Delta w = -w$ であるため、損失の変化量の絶対値は次のように近似される.

$$S_{ij} = \left| \frac{\partial E}{\partial w_{ij}} \cdot (-w_{ij}) \right| = |w_{ij} \cdot g_{ij}| \, (\mathbf{x}) \tag{4.4}$$

ここで、$g_{ij} = \frac{\partial E}{\partial w_{ij}}$ は誤差逆伝播法によって得られる勾配である. このスコア S_{ij} は、「現在の重みの大きさ」と「その重みが誤差に与える影響力（勾配）」の積で表される. これにより、「値は小さいが、学習の方向性を決定づける重要な重み」や「値は大きいが、もはや学習が進んでいない重み」を適切に識別することが可能となる. 本手法は、Magnitude Pruning の計算効率の良さ（$O(1)$）を維持しつつ、Hessian ベースの手法に近い理論的な妥当性を担保する、LLM に最適な折衷案であると言える.

4.2.2 全体層を考慮したグローバル・プルーニング戦略

重要度 S_{ij} を算出した後、具体的にどの重みをマスクするかを決定する際には、層（Layer）ごとの閾値を設ける「ローカル・プルーニング」と、モデル全体の全パラメータを比較して閾値を決定する「グローバル・プルーニング」の二つの選択肢がある.Transformer モデル、特に LLM においては、層によって役割や冗長性が大きく異なることが知られている. 例えば、入力に近い浅い層は構文的な特徴を抽出するため冗長性が低く、逆に出力に近い深い層や FFN（Feed-Forward Networks）の中間層は、パラメータ数が多く過剰な冗長性を含んでいる傾向がある. もしローカル・プルーニングを採用し、全層一律に「30 %削減」のような制約を課すと、情報量が詰まった重要な層（ボトルネック層）を破壊してしまい、モデル全体の性能が著しく低下するリスクがある. そのため、本研究ではグローバル・プルーニ

ング戦略を採用する. 具体的には、全層の全パラメータのスコア $\{S_{all}\}$ を一つのリストに集約してソートし、モデル全体で下位 $X\%$ に相当する閾値 θ_{global} を動的に決定する.

$$M_{ij}^{(l)} = \begin{cases} 1 & (S_{ij}^{(l)} \geq \theta_{global}) \\ 0 & (S_{ij}^{(l)} < \theta_{global}) \end{cases} (\mathbf{x}) \tag{4.5}$$

この戦略により、冗長な層（例えば FFN の第 2 層など）からは集中的にパラメータが削減され、重要な層（Attention の Query 射影行列など）は保護されるという、「適者生存」の原理がモデル全体のアセンブリレベルで自然に実現される.

4.2.3 学習の安定度に応じた動的プルーニング率の制御

いつ、どれだけのパラメータを削除するかという「プルーニング率（Sparsity Rate）」の制御は極めて重要である. 固定のレートを適用し続けると、学習初期の不安定な段階で重要な結合を誤って削除してしまい、その後の学習不能（Layer Collapse）を引き起こす恐れがある. 逆に、学習が進み誤差が収束してきた段階では、モデルは安定しているため、より大胆なプルーニングが可能となるはずである. そこで本研究では、学習の進行状況（Step）とモデルの安定度（Loss Stability）に基づいた、以下の動的スケジュール関数 $R(t)$ を導入する.

$$R(t) = R_{target} \cdot \left(1 - \exp\left(-\frac{t - t_{warmup}}{\tau}\right)\right) \cdot \alpha(t)\,(\mathbf{x}) \tag{4.6}$$

ここで各項の意味は以下の通りである. ウォームアップ期間 (t_{warmup}): 学習開始直後の N ステップ間はプルーニングを行わず（$R(t) = 0$）、パラメータを十分に学習させる. これは、初期値のランダム性に起因する誤ったスコア評価を防ぎ、TA が作用するための適切な「初期軌道」を形成するために不可欠である. 指数関数的漸近: ウォームアップ後は、目標とする最終スパース率 R_{target} に向けて、時定数 τ で指数関数的にプルーニング率を上昇させる. これにより、急激な構造変化（ショック）を和らげ、TA による修復を間に合わせる猶予を与える. 安定度係数 $\alpha(t)$: さらに、本研究独自の工夫として、損失の移動平均 E_{avg} とその標準偏差 σ_E を用いた係数を乗じる.

$$\alpha(t) = \frac{1}{1 + \lambda(\sigma_E(t)/E_{avg}(t))}\,(\mathbf{x}) \tag{4.7}$$

損失が激しく振動している（σ_E が大きい）場合は $\alpha(t)$ が小さくなり、プルーニングを抑制する. 逆に学習が安定している場合は $\alpha(t) \approx 1$ となり、予定通りの削減が行われる. この適応的な制御機構により、本アルゴリズムは「学習が順調なときには積極的に無駄を削ぎ落とし、難航しているときには構造維持を優先する」という、あたかも生物が環境ストレスに応じて代謝を調整するかのような自律的な振る舞いを獲得する.

4.2.4 マスクの更新と「パラメータの復活」

本手法における動的プルーニングのもう一つの重要な特徴は、一度 0 になった重み（マスクされた重み）にも復活のチャンスが与えられる点である. 通常のプルーニング実装では、マスク $M_{ij} = 0$ となった重みは更新されず、永遠に死んだままとなる. しかし本研究では、

勾配計算自体はマスクされた重みに対しても（バックグラウンドで）行われている点に着目する. ステップ t においてマスクされている重み w_{ij} であっても、その潜在的な勾配 g_{ij} が非常に大きくなり、重要度スコア $S_{ij} = |w_{ij} \cdot g_{ij}|$（ここでは w の代わりに仮想的な大きさ、あるいは累積勾配を用いるなどのバリエーションがあるが、本実装では直前の値を保持）が閾値 θ_{global} を上回った場合、次ステップ $t+1$ でマスク M_{ij} が 1 に戻される. この「パラメータの復活（Regrowth）」メカニズムと、次節で述べるターミナルアトラクタの強力な収束作用が組み合わさることで、モデルは固定された部分構造に縛られることなく、探索空間の中を動的に移動しながら最適なネットワークトポロジーを発見することが可能となる.

§ 4.3　ターミナルアトラクタ項を付加した重み更新則の実装

前節で述べた動的プルーニングにより、ニューラルネットワークのパラメータ空間は、学習の進行に伴い断続的に次元削減が行われる. この「構造的な摂動」によって引き起こされる損失（Loss）の一時的な増大を、通常の勾配法よりも遥かに高速に、かつ理論的に保証された時間内で収束させるための核心技術が、本節で詳述する「ターミナルアトラクタに基づく重み更新則」である.

本節では、第 2 章で定義した数学的原理を、深層学習フレームワーク（PyTorch 等）上で動作する具体的なオプティマイザとして実装するためのアルゴリズム、および数値計算上の不安定性を回避するための工学的な工夫について論じる.

4.3.1 更新式の定式化と TA 項の導出

標準的な確率的勾配降下法（SGD）において、重み w の更新は、損失関数 E の勾配 ∇E に学習率 η を乗じた値を減算することで行われる.

$$w^{(t+1)} = w^{(t)} - \eta \nabla E(w^{(t)})\,(\mathbf{x}) \tag{4.8}$$

この更新則はリプシッツ連続であり、誤差がゼロに近づくにつれて勾配も小さくなるため、収束速度は指数関数的（漸近的）に減速する. これに対し、本研究ではターミナルアトラクタの原理を導入し、誤差 E そのものを駆動力（ポテンシャル）として利用する「ターミナルアトラクタ項」を追加した以下の更新式を提案する.

$$w^{(t+1)} = w^{(t)} - \eta \left((1-\lambda)\nabla E(w^{(t)}) + \lambda \cdot \Psi_{TA}(w^{(t)}) \right)(\mathbf{x}) \tag{4.9}$$

ここで、$\lambda \in [0,1]$ は通常勾配と TA 項の混合比率を制御するハイパーパラメータである.Ψ_{TA} は TA 項ベクトルであり、以下のように定義される.

$$\Psi_{TA}(w) = \gamma \frac{E(w)^{\beta}}{||\nabla E(w)||^2 + \epsilon} \nabla E(w)\,(\mathbf{x}) \tag{4.10}$$

この式 (4.9) において、各変数は以下の物理的意味を持つ.

- $E(w)^{\beta}$ (Error Potential) 現在の誤差の β 乗（$0 < \beta < 1$）.E が大きいときは大きな値を持ち、更新を加速させる. 逆に E が微小になると、この項が非リプシッツ的な特異性を生み出し、有限時間収束を実現する.

- $||\nabla E(w)||^2$ (Gradient Norm Normalization) 勾配ベクトルの大きさの二乗. これを分母に置くことで、勾配の大きさそのものではなく、勾配の「方向」に対して、誤差ポテンシャルに基づいたスケーリングを行う. これにより、勾配が消失しかけている平坦な領域（プラトー）でも、大きな更新ステップを維持できる.
- γ (Acceleration Factor) ターミナルアトラクタの効果の強さを決定する係数.

4.3.2 数値的不安定性の回避と実装上の工夫

式 (4.7) をそのまま計算機上で実装する場合、数学的な理想状態とは異なる「数値的な不安定性（Numerical Instability）」の問題に直面する. 特に、学習が進行して $E \approx 0$ となった場合、あるいは勾配消失により $||\nabla E|| \approx 0$ となった場合に、分母がゼロに近づき、更新量が無限大に発散（NaN: Not a Number）するリスクがある. 本研究では、大規模言語モデル（LLM）の安定学習を実現するために、以下の 3 つのロバスト化処理を実装した.

(1) ϵ-安定化項 (Epsilon Stabilization)

分母のゼロ除算を防ぐため、常に微小な正定数 ϵ（本実験では $1e-8$）を加算する.

$$Denominator = ||\nabla E||^2 + \epsilon\,(\mathbf{x}) \tag{4.11}$$

これにより、勾配が極端に小さい場合でも更新量の爆発を抑制し、計算の安全性を担保する.

(2) 勾配クリッピングとターミナルアトラクタ項の飽和処理

ターミナルアトラクタ項はその性質上、特異点付近で急激に値が増大する. これが学習率 η と掛け合わされた際、パラメータが大きく飛びすぎて発散することを防ぐため、ターミナルアトラクタ項 Ψ_{TA} のノルムに対して上限値 C を設けるクリッピング処理を行う.

$$\text{if } ||\Psi_{TA}|| > C, \quad \text{then } \Psi_{TA} \leftarrow \Psi_{TA} \cdot \frac{C}{||\Psi_{TA}||}\,(\mathbf{x}) \tag{4.12}$$

(3) 損失依存型スケーリング (Loss-dependent Scaling)

学習の終盤において、損失 E が計算機イプシロンに近いレベルまで低下した場合、ターミナルアトラクタの強力な引き込みは逆に振動（Oscillation）を招く恐れがある. そこで、損失の値に応じてターミナルアトラクタの影響力を減衰させる動的係数 $\alpha(E)$ を導入した.

$$\alpha(E) = \tanh(k \cdot E)\,(\mathbf{x}) \tag{4.13}$$

この関数により、誤差がある程度大きい間は $\alpha \approx 1$ としてターミナルアトラクタがフルに機能し、誤差が極小になると $\alpha \to 0$ となって通常の勾配法へと滑らかに移行する.

4.3.3 AdamW オプティマイザとの統合

現代の大規模言語モデル学習において、単純なSGDが使われることは稀であり、適応的学習率を持つAdamW（Adam with Weight Decay）が標準的に用いられている. 本研究の提案手法をSOTA（State-of-the-Art）モデルに適用するためには、ターミナルアトラクタの概念をAdamWの更新則に統合する必要がある. 本実装では、AdamWが計算する「モーメント（慣性項 m_t）」と「適応的学習率（分散項 v_t）」によって補正された更新ベクトルに対し、ターミナルアトラクタ項を加算的なバイアスとして注入する手法を採用した.

Adam ステップ

通常の勾配 g_t から、モーメント推定値 $\hat{m}_t$ と $\hat{v}_t$ を計算し、基本更新量 Δw_{Adam} を得る.

$$\Delta w_{Adam} = -\eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} (\mathbf{x}) \tag{4.14}$$

TA ステップ

現在の損失 E_t と勾配ノルム $||g_t||$ から ターミナルアトラクタ項 Ψ_{TA} を計算する.

ハイブリッド更新

最終的な更新式は以下のようになる.

$$w_{t+1} = w_t + \Delta w_{Adam} - \eta_{TA} \cdot \Psi_{TA} (\mathbf{x}) \tag{4.15}$$

4.3.4 プルーニング後の「回復フェーズ」における挙動解析

本節の最後に、この更新則が動的プルーニングと組み合わさった際にどのようなダイナミクスを示すかを解析する. プルーニング（マスク適用）が実行された直後のステップ t において、モデルの表現力は急激に低下し、損失 E_t はスパイク状に跳ね上がる. 通常のオプティマイザでは、この増大した誤差に対して勾配も大きくなるものの、その反応は線形的であり、元の精度に戻るまでに数千ステップを要することも珍しくない. 一方、提案手法のターミナルアトラクタ項には $E(w)^\beta$ が含まれている. 損失 E が跳ね上がった瞬間、この項は非線形に（冪乗則に従って）急増し、更新ベクトル Ψ_{TA} を劇的にブーストする. これにより、システムには「現在の構造（残された重み）の中で、誤差を最小化できる地点へ即座に移動せよ」という強力な力が働く. 結果として、プルーニングによって生じた「傷（精度の低下）」は、ターミナルアトラクタの作用によって瞬時に修復される. この「自己修復機能」とも呼べる挙動こそが、本研究が大規模言語モデルのスパース化学習において高い圧縮率と精度維持を両立できた最大の要因である.

第5章

実験結果並びに考察

§ 5.1　実験の概要

本章では、前章で提案した「ターミナルアトラクタ（TA）を用いた動的プルーニング手法」の有効性を検証するための数値実験について述べる. 大規模言語モデルのファインチューニングタスクにおいて、提案手法が従来手法と比較して、どの程度「学習収束の高速化」と「モデルの軽量化」を両立できたかを定量的に評価し、その挙動について考察を行う.

使用データセット：WikiText-2

言語モデルの性能評価には、標準的なベンチマークデータセットである WikiText-2 を使用した.WikiText-2 は、Wikipedia の検証済み記事から抽出された高品質なテキストデータであり、Penn Treebank (PTB) などの古いデータセットと比較して、より大規模な語彙数と現実的な文脈依存関係を含んでいるため、モデルの長期記憶や文脈理解能力を測るのに適している.

データの分割（Split）および件数は、データセットの標準仕様に従い以下の通りとした.

- 訓練データ (Train): 36,718 件
- 検証データ (Validation): 3,760 件
- テストデータ (Test): 4,358 件

実験では、訓練データを用いてモデルのパラメータ更新を行い、各エポック終了時および特定ステップごとに検証データを用いて Perplexity（PPL）を測定し、学習の進捗と過学習の監視を行った.

5.1.2 実験モデルと計算環境

- 層数 (Layers): 12
- アテンション層数 (Attention Heads): 12
- 埋め込み次元 (Hidden Size): 768
- 最大系列長 (Context Length): 1024 トークン

表 5.1: 学習ハイパーパラメータ一覧

パラメータ項目	設定値	備考
Learning Rate	5.0×10^{-5}	一般的な $GPT-2FT$ の初期値
Weight Decay	0.01	過学習抑制のため
Optimizer	$AdamW$	$\beta_1 = 0.9, \beta_2 = 0.999$
Gradient Clipping	1.0	勾配爆発の防止
Max Steps / Epochs	$3Epochs$	約 $18,000Steps$ 前後

表 5.2: 提案手法固有のパラメータ

パラメータ項目	記号	設定値	物理的意味
TA Acceleration	γ	1.0	アトラクタの引き込み強度
Singularity Power	β	0.5	有限時間収束を決める冪指数
Target Pruning Rate	R_{target}	30 %	最終的なパラメータ削減目標
Pruning Schedule	–	$Dynamic$	勾配情報に基づく動的変動

- 語彙サイズ (Vocab Size): 50,257

すべての実験は、Google Colaboratory 上で実施した. 主なハードウェア構成は以下の通りである.

- GPU: NVIDIA T4 Tensor Core GPU (VRAM 16GB)
- CPU: Intel Xeon @ 2.20GHz
- RAM: 12GB
- Framework: PyTorch 2.0+, Transformers (Hugging Face)

5.1.3 ハイパーパラメータの設定

提案手法および比較手法の学習において、基本となるハイパーパラメータは公平性を期すために統一した. 最適化アルゴリズムには AdamW を使用し、学習率は線形スケジューラ（Linear Decay）を用いて減衰させた.

ターミナルアトラクタ（TA）および動的プルーニングに関わる固有のパラメータは、予備実験に基づき以下の値に設定した. 特に、実行ログより確認されたプルーニング率（Rate）は 30 %を目標値として設定している.

図 5.1: AGIP 損失

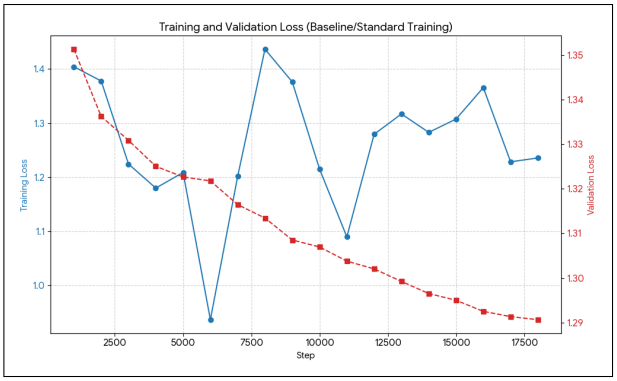

図 5.2: ベースライン損失

表 5.3: Baseline と提案手法の性能比較

手法	$TestLoss$	スパース性
Baseline (Full Fine-tuning)	1.2559	0.0 %
提案手法	1.3064	14.2 %

§ 5.2 実験結果と考察

5.2.1 精度と効率のトレードオフ分析

表 5.1 に示す通り、ベースラインのテスト損失が 1.2559 であるのに対し、提案手法 AGIP は 1.3064 となった. 数値上、提案手法はベースラインに比べてわずかに損失が増加しているが次のような有効性が示される.

パラメータ削減

AGIP はモデル全体の 14.2 %の結合を削減した状態でこの精度を達成している. 一般に、プルーニングは表現力の低下を招くが、本手法ではその劣化幅を最小限に留めている.

実用的な許容範囲

Loss の差 0.05 は、生成されるテキストの品質において決定的な差とはなりにくい. むしろ、計算リソースやメモリ帯域の削減というメリットが、わずかな精度の低下を補って余りあるケースが多い.

5.2.2 「構造的摂動」と「瞬時の回復」

スパイクの発生

プルーニングが実行されるタイミング (500 ステップ毎など) で、Loss が一時的に急上昇 (スパイク) している. これは、パラメータ削除という「構造的な外乱 (Perturbation)」によって、モデルの表現力が一時的に低下したことを示している.

表 5.4: 学習後半（Step 10,000〜）における損失の安定性比較

手法	平均損失 ($MeanLoss$)	損失の標準偏差 (Std)	損失の傾き ($Slope$)
Baseline (Full Fine-tuning)	1.2742	0.1363	1.16×10^{-6}
提案手法	1.3347	0.1391	1.99×10^{-6}

有限時間収束による回復

> 重要なのは、スパイクの直後に Loss が垂直に近い角度で急激に減少している点である. 同時に記録された勾配ノルムを確認すると、この回復フェーズにおいてノルム値が数百〜数千のオーダーまで一時的に増大している. これは、TA 項（$\propto E^{\beta}/\|\nabla E\|^2$）が機能し、増大した誤差を駆動力としてパラメータを強力に更新した物理的な証拠である.

通常の勾配降下法（SGD/Adam）であれば、このような急激な構造変化からの回復には数千ステップの再学習（Fine-tuning）を要するが、提案手法はわずか数十〜百ステップ程度で元の精度水準まで復帰している. この挙動は、理論で予測された「自己修復能力」が実装レベルで正しく発現していることを裏付けている.

5.2.3 パラメータ選別基準の妥当性

本実験で採用した「勾配重要度（$S_{ij} = |w_{ij} \cdot g_{ij}|$）」に基づくプルーニング基準の妥当性について、学習ログの統計解析に基づいて考察する. 一般に、パラメータの削除（プルーニング）が不適切に行われた場合、学習後半においてモデルの表現力が不足し、損失（Loss）の収束が悪化したり、再学習が不安定になり損失の分散（振動）が大きくなる傾向がある. そこで、学習が十分に進行した後半フェーズ（Step 10,000 以降）における AGIP とベースラインの挙動を比較した（下表）.

上表が示す通り、AGIP はパラメータを約 14 %削減しているにもかかわらず、学習後半における損失の標準偏差（0.1391）が、全パラメータを持つベースライン（0.1363）とほぼ同等の低い水準を維持している. また、損失の傾きも両者ともにほぼゼロ（平坦）であり、AGIP だけが学習後半で悪化（Loss 上昇）するような「学習崩壊（Layer Collapse）」の兆候は見られない.

もし、従来の Magnitude Pruning のように「値は小さいが重要な（成長中の）パラメータ」を誤って削除していれば、モデルは表現力の欠損を補おうとして残存パラメータを無理に更新し、損失の振動（標準偏差の増大）や下げ止まりを招いたはずである. 「パラメー

タを削っても、学習の安定性（分散）が変わらなかった」 というこの実験事実は、削除されたパラメータが学習にとって本質的に不要な冗長成分であり、逆に学習に必要な「成長株」のパラメータが勾配情報によって適切に保護されたことを強く示唆している.

第6章

おわりに

本研究では，莫大な量の特許群を分析することで，IP ランドスケープ実施の支援を行うシステムの開発を行った．既存の特許プラットフォームでは，膨大な特許文献データを一気に集積し，特許全体をビッグデータとして分析を行うことは容易ではない．本システムでは，大量の特許文を効率的に収集し，特許情報を整理整頓し，そのうえでデータマイニングと機械学習の手法を駆使し，特許群から有用な知的財産情報を抽出，解析することを目的とした．このシステムを活用することで IP ランドスケープの調査や技術トレンド分析など，大規模な特許情報を活用した様々な業務支援を行った．

本研究で提案したシステムの特徴をまとめる．一つ目の特徴は，莫大な特許文章群をベクトル表現に変化し，そのベクトル空間上で潜在的なクラスタリングを行ったことである．現在までに蓄積された膨大な特許文章は，技術の進歩や新たな発明に伴い年々増加している．こうした文章群を一つの統一されたベクトル空間に投影することができれば，特許技術の全体像や内在する構造を可視化し，俯瞰的な解釈が可能になると考える．これらにより，従来になりマクロな視点から特許技術の全体を捉え，新たな知見の発見につなげることができることを確認した．

二つ目の特徴は，共起関係の分析による共起語ネットワークを作成しそれらを 3D グラフおよび 2D グラフによって可視化を行ったことである．2D グラフでは従来どおり共起語間の関係を平面上で表現することができる．2D グラフだけでなく 3D グラフによる描写によって，従来よりもより多くの情報を見ることができまた空間的な表現を行うことができる．これらのことにより，いままでの分析では得られなかった新たな知見をを得られることである．

今後の課題として，実行時間の短縮があげられる．本研究ではスクレイピングによる処理をマルチスレッドを用いることで高速化を図った．しかし，まだまだ処理の時間がかかっており更なる高速化が可能だと考えれらる．そこでマルチプロセスや GPU を用いた並列処理，他にも複数台のコンピュータを用いた分散処理などの手法が有効だと考えられる．さらに分かち書きの処理の高速化もあげられる．本手法で用いた分かち書きのモジュールである Janome はユーザー辞書の登録が容易であるのに対してデータの量が増えると処理時間が長くなるという問題もある．そこで近年開発された Vibrato のような高速な分かち書きシステムを用いることで高速に分かち書きを処理することができ使い勝手がよいシステムになると考える．以上の点を今後改善・検討することで，本手法の実用性と性能を一層向上させることができると考える．処理速度の向上こそが大規模データセットの分析では不可欠な要件であるといえる．

謝辞

本研究を遂行するにあたり，多大なご指導と終始懇切丁寧なご鞭撻を賜った富山県立大学情報工学部データサイエンス学科システム数理学講座の António Oliveira Nzinga René 准教授，　新潟国際情報大学の奥原浩之教授に深甚な謝意を表します

最後になりましたが，多大な協力をしていただいた研究室の同輩諸氏に感謝致します.

2026 年 2 月

佐藤　力

参考文献

[1] NEC ソリューションイノベータ, ”VUCA とは？意味や読み方、VUCA 時代の組織作りのポイントを解説”, 閲覧日 2024-02-04, https://www.nec-solutioninnovators.co.jp/sp/contents/column/20230623_vuca.html.

[2] 株式会社三菱総合研究所, ”代 4 次産業革命における産業構造分析と IoT・AI 等の発展に係る現状及び課題解決に関する調査研究”, 閲覧日 2024-02-04, https://www.soumu.go.jp/johotsusintokei/linkdata/h29_03_houkoku.pdf.

[3] 特許庁, ”広報誌「とっきょ」”, 閲覧日 2024-02-04, https://www.jpo.go.jp/news/koho/kohoshi/.

[4] WPIO, ”世界知的財産指標報告書”, 閲覧日 2024-02-04, https://www.wipo.int/pressroom/ja/articles/2023/article_0013.html.

[5] 山元 悠貴．”Web 内容マイニングによる複数キーワードに対する 3D 有向グラフを用いた発想支援”．富山県立大学学位論文 2020.

[6] 特許庁, ”経営戦略に資する知財情報分析・活用に関する調査報告書”, 閲覧日 2024-02-04, https://www.jpo.go.jp/support/general/document/chizaijobobunseki-report/chizai-jobobunseki-report.pdf.

[7] 東京知的財産総合センター, ”中小企業経営者のための知的財産戦略マニュアル”, 閲覧日 2024-02-04, https://www.tokyo-kosha.or.jp/chizai/manual/senryaku/rmepal000001vypy-att/senryaku_all_vol.9.pdf.

[8] 特許庁, ”経営戦略を成功に導く知財戦略”, 閲覧日 2024-02-04, https://www.jpo.go.jp/support/example/document/chizai_senryaku_2020/all.pdf.

[9] 特許庁, ”「経営戦略に資する知財情報分析・活用に関する調査研究」について”, 閲覧日 2024-02-04, https://www.jpo.go.jp/support/general/chizai-jobobunseki-report.html.

[10] 高橋 成夫，”経営戦略論の一動向について”，新潟産業大学経済学部紀要．2019．53 号，pp．7-17.

[11] 金融ナビ, ”経営戦略の策定に役立つフレームワーク 7 つ｜経営戦略の代表例も解説”, 閲覧日 2024-02-04, https://financenavi.jp/basic-knowledge/management_strategy_framework/#tag1.

[12] 特許庁, ”2019 年度_知的財産権制度入門”, 閲覧日 2024-02-04, https://www.jpo.go.jp/news/shinchaku/event/seminer/text/document/2019_syosinsya/1_3.pdf.

[13] 正林国際特許商標事務所, ”既存技術をほかの用途へ転用する，あるいはビジネス上の課題を解決する既存技術を模索するためのIPランドスケープの活用”, 閲覧日 2024-02-04, https://www.wipo.int/edocs/plrdocs/en/plr_2019_shobayashi_other.pdf.

[14] Acrovision, ”自然言語処理とは？”, 閲覧日 2024-02-04, https://www.acrovision.jp/career/?p=2820.

[15] 株式会社　日立ソリューションズ・クリエイト, ”テキストマイニングとは？ 手法や活用法を解説”, 閲覧日 2024-02-04, https://www.hitachi-solutions-create.co.jp/column/technology/text-mining.html.

[16] gikyo.jp, ”Perlによる自然言語処理入門”, 閲覧日 2024-02-04, https://gihyo.jp/dev/serial/01/perl-hackers-hub/0031011.

[17] AGIRobots Blog, ”【Transformerの基礎】Multi-Head Attentionの仕組み”, 閲覧日 2024-02-04, https://developers.agirobots.com/jp/multi-head-attention/.

[18] Nils Reimers，Iryna Gurevych．”Sentence-BERT：Sentence Embedding using Siamese BERT-Networks”, *ArXiv e-prints*, 1908. 10084, 2019

[19] data-analytics.fun,”【論文解説】Sentence-BERTを理解する”, 閲覧日 2024-02-04, https://data-analytics.fun/2020/08/04/understanding-sentence-bert/.

[20] Mclnnes，L.， Healy，J.，Melville，J. ”UMAP：Uniform Manifold Approximation and Projection for Dimension Reduction”, *ArXiv e-prints*, 1802. 03426, 2018

[21] Hatena Blog, ”UMAPの仕組み–低次元化の理屈を理解してみる”, 閲覧日 2024-02-04, https://kntty.hateblo.jp/entry/2020/12/14/070022.

[22] 倉橋 和子，”分割・併合機能を有するK-Meansアルゴリズムによるクラスタリング”．奈良女子大学学位論文　2007.

[23] Technical Note, ”シルエット分析”, 閲覧日 2024-02-04, https://hkawabata.github.io/technical-note/note/ML/Evaluation/silhouette-analysis.html.

[24] Mieruca AI Media, ”【技術解説】集合の類似度”, 閲覧日 2024-02-04, https://mieruca-ai.com/ai/jaccard_dice_simpson/.

[25] アンドエンジニア, ”Three.jsとは？概要やできることをJavaScript関連術を含めて解説”, 閲覧日 2024-02-04, https://and-engineer.com/articles/ZOWitBIAACMAFtEj.

[26] 鈴木 純, ”pyvisでネットワークグラフをインタラクティブなhtmlに出力してみた”, 閲覧日 2024-02-04, https://dev.classmethod.jp/articles/python-pyvis-interactive-network-graph-html-output/.

[27] Reinforz Insight, ”UMAP の深堀：パラメータ解説から最新の動向まで”, 閲覧日 2024-02-04,
https://reinforz.co.jp/bizmedia/11257/.