ファイチューニングの最適化支援のための埋め込み移動軌跡の可視化分析

石川　智也†　　藤田　秀之††　　大森　　匡††　　新谷　隆彦††

† 電気通信大学情報理工学域　〒 182–8585 東京都調布市調布ヶ丘一丁目 5 番地 1 号
†† 電気通信大学情報理工学研究科　〒 182–8585 東京都調布市調布ヶ丘一丁目 5 番地 1 号
E-mail: †i2210038@edu.cc.uec.ac.jp, ††{fujita,omori,shintani}@is.uec.ac.jp

あらまし　汎用言語モデルでは，テキストやトークンを埋め込みと呼ばれるベクトルで表現する．ファインチューニングの性能評価は，目的とするタスクに対するモデルの精度評価を代表とする，外在的評価が中心だが，精度改善や副作用の根拠を分析することは困難とされている．そこで，埋め込み空間の可視化分析を含む，内在的評価の手法が提案されている．本研究では，ファインチューニング完了後の埋め込み空間だけではなく，ファインチューニングの過程で各埋め込みが移動する軌跡（埋め込み軌跡）に着目し，軌跡マイニングを援用することで，ファイチューニングの過程を可視化分析する手法を検討する．ニュース記事分類タスクに対する BERT ファイチューニングを対象に，各クラスの訓練データ数や F1 スコアの推移と，埋め込み移動軌跡の対応を分析した．特に，学習の進行に従い，埋め込み点の移動速度が低下し，滞留し始めることに着目し，訓練データ数の多いクラスほど移動速度の低下が早く，滞留が多く検出される傾向を確認した．さらに，速度のばらつきを考慮した滞留点検出の手法を提案し，その結果を分析した．分析結果より，埋め込み移動軌跡に対する滞留点検出の活用方針をまとめた．

キーワード　解釈可能性，汎用言語モデル，埋め込み可視化，可視化分析，Human-in-the-loop システム

1　はじめに

ファインチューニングの内在的評価が行われる際，事前学習済みモデルやファインチューニング終了後のモデルにおける埋め込みを可視化し分析するということは，これまでよく行われてきた．

それに対し，ファインチューニングの経過における埋め込みベクトルの変化を軌跡として捉え，それを詳細に分析するということはあまり行われてこなかった．そのため，埋め込みベクトルの軌跡に対して軌跡データマイニングを適用して分析する手法を検討する．

各クラスの軌跡の類似度などに着目することで，ファインチューニングの有効性をクラスごとやデータごとに判断することができるようになることが期待できる．

2　先行研究

本章では，ファイチューニングの内在的評価を行っている先行研究として，*A Closer Look at How Fine-tuning Changes BERT* [1] について紹介する．言語モデルが，内部でどのように情報を学習しているのか追跡することをプロービングという．この研究では，ファインチューニングの効果を評価するために，モデルを固定特徴抽出器として使いその上にニューラルネットワークを構築することで分類器としての性能を測定する手法と，幾何学的な観点で埋め込みをクラスタリングする DIRECTPROBE [2] を用いて，プロービングを行っている．

DIRECTPROBE は，各クラスタが同一ラベルの点のみを含み，クラスタの凸包同士が重ならないクラスタの集合を返す．図 1 に，単純な二値分類問題とそれに DIRECTPROBE を適用した結果を示す．図 1 から，この 2 クラスの決定境界は，異なるクラスタの間を通るということがわかる．DIRECTPROBE を適用した結果，クラスタ数とクラス数が等しいとき，その分類問題は線形分離可能な問題だということになる．

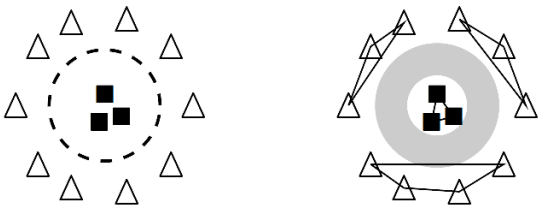

図 1: 単純な二値分類問題とそれに DIRECTPROBE を適用した結果．白い三角形と黒い四角形はクラスを，点同士を繋いだ線はクラスタを表し，左図は破線の円形決定境界を持つ二値分類問題，右図はその問題に DIRECTPROBE を適用した結果を表す．

この研究では，DIRECTPROBE が返すクラスタの集合について，クラスタ数，クラスタ間距離，空間的類似性を測定し，分析することで，ファインチューニングの内在的評価をしている．

分類器ベースのプロービングと DIRECTPROBE により，ファインチューニング後には DIRECTPROBE のクラスタ数が減少し，異なるラベルのクラスタ同士の最小距離が増加していくことが確認されている．

また，対象タスクと異なるタスクでファインチューニングを行うクロスタスクファインチューニングでは，対象タスクに類似したタスクでファインチューニングした場合，クラスタ数は増加して異なるラベルのクラスタ間の距離は増加するが，対象タスクに悪影響を及ぼすと思われるタスクでファインチューニ

ングするとクラスタ数が増加して異なるラベルのクラスタ間の距離は減少するということがわかっている．

また，上位層であっても埋め込みを無秩序には変更せず，下位層は上位層よりラベルを分離せず，埋め込みの移動範囲が小さいことなどもわかっている．

さらに，この研究では，あるタスクで最も近い 3 つのラベルの重心の軌跡を追跡している．図 2 にその 3 つのラベルの重心の移動軌跡を示す．図 2 より，ファインチューニングが進むにつれて，上位層では重心は異なる方向に互いに離れるように移動するということがわかる．

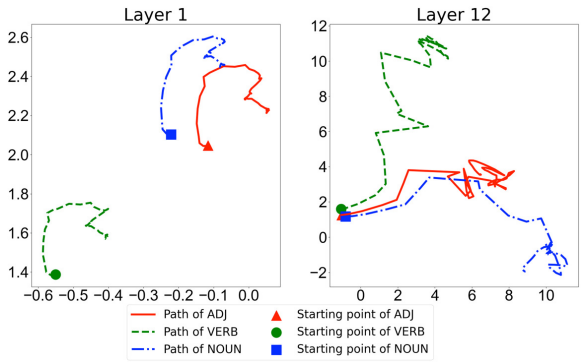

図 2: $\text{BERT}_{\text{base}}$ の POS（Part-of-speech tagging：品詞タグ付け）で最も近い 3 つのラベルの重心のファインチューニング中における変化（PCA 投影）．丸と三角形及び四角形はクラス，線は重心の移動軌跡を表し，左図は第 2 層の結果，右図は第 12 層の結果を表す．

3 予 備 実 験

この予備実験では，ファインチューニングの経過に伴って生じる埋め込みの変化を可視化し，それらにどのような傾向が確認できるかを分析する．

ミニバッチを用いてパラメータを更新した回数をステップ数という．本研究では，まず日本語事前学習済み BERT を用いて，テキストをクラスに分類する，分類タスクを対象としたファインチューニングを行う．次に，ファインチューニングを行う際に一定のステップ間隔でモデルを保存していき，保存した各モデルにおける入力テキストの埋め込みベクトルを取得する．そして，取得した埋め込みベクトルから，ファイチューニングの過程における埋め込みベクトルの変化を捉えたアニメーションと軌跡の画像を作成する．

事前学習済みモデルとして，東北大学が Hugging Face 上で提供している事前学習済み BERT を用いた．このモデルは，日本語の Wikipedia を事前に学習している．

データセットには，livedoor ニュースコーパスの Hugging Face Hub 上に公開されたデータセットを使用した．この各データには，記事の「URL, 日付，タイトル，本文，カテゴリー」の 5 つの情報が含まれており，データ件数は 6630 である．なお，訓練データは 5894 件，テストデータは 736 件である．

記事データを分類するタスクを対象タスクとした．このタスクでは，モデルはデータに含まれている「本文」を入力として「カテゴリー」（以下，クラスと呼ぶ）を出力する．事前学習済み BERT に対して，エポック数を 3，モデルを保存するチェックポイントの間隔を 10 ステップごとにして，上記の分類タスクに向けてファインチューニングを行った．ただし，学習率に関して，スケジューラは線形減衰になっており，Warmup はしない．

埋め込み軌跡に影響を及ぼすと考えられるハイパーパラメータについて説明する．ユーザが設定する学習率に対応してスケジューラが働くとき，エポック数に合わせて学習率が変化する．線形減衰になっている場合は，最大の学習率に達したところから線形に学習率が減少してファインチューニング終了時に 0 になる．そして，ステップ数を指定して Warmup を利用するときは，学習率が，0 からユーザが設定した学習率の最大値まで，指定したステップ数を通じて線形に増加する．また，AdamW は，スケジューラや Warmup の影響を受けた学習率に対し，勾配に適応的な学習率をさらに計算する．この適応的な学習率によってパラメータの変化の大きさは影響を受ける．本研究では，埋め込み点の一定のステップ数あたりの移動距離を移動速度と定義する．パラメータの変化が小さい場合，埋め込み点の移動速度は小さくなると考えられる．

続いて，各チェックポイントで保存したモデルを用いて埋め込みベクトルを取得した．そして，ファインチューニングの経過に伴った埋め込みベクトルの変化を可視化するため，取得した埋め込みベクトルから以下を作成した．

(i) 各フレームにのテストデータに対する埋め込み空間をプロットしたアニメーション

(ii) クラス毎に 10 個データをランダムに選び，その埋め込み軌跡を描画した画像

ただし，(i) と (ii) を作成するにあたり，ファインチューニング終了時のモデルで取得した埋め込みベクトルを PCA で二次元に次元削減して射影行列を求め，その射影行列により他のチェックポイントモデルの埋め込みベクトルも次元削減をした．

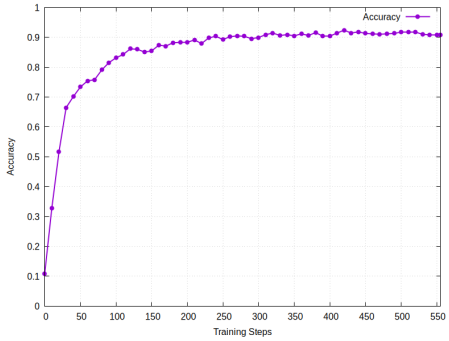

図 3: ファインチューニングにおけるステップ数と正解率の関係．縦軸は正解率，横軸はステップ数を表す．

まず，ファインチューニングの過程におけるテストデータに対する正解率の変化を図 3 に示す．図 3 から，0 ステップ目から 100 ステップ目までは急激に正解率が高くなっていき，それ

以降は正解率の上昇がやや緩やかになり，300 ステップ目以降はほとんど一定となっていることがわかる．ファイチューニング開始時は正解率は 10.87%であり，440 ステップ目で正解率は最大値の 91.85%となっている．

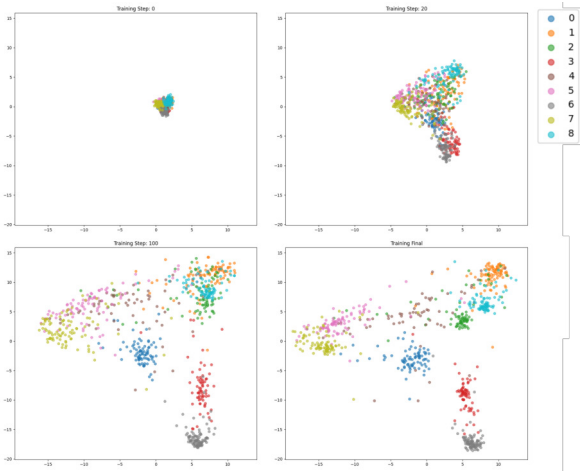

図 4: 埋め込みの移動を表すアニメーションの一部．埋め込み点の色はクラスを表す．左上の図は 0 ステップ目，右上の図は 20 ステップ目，左下の図は 100 ステップ目，右下の図はファインチューニング終了時の埋め込み点の位置を示す．

作成した，(i) のアニメーションの一部を図 4 に示す．図 4 から，図 3 における正解率の変化に対応して，100 ステップ目前後までは埋め込みベクトルの変化は大きいが，それからは変化が緩やかになっていっているのがわかる．最初は点が原点付近に集まっているが，そこから 100 ステップ目前後までは各点が自クラスの点と共に原点から離れていくように見える．そして，100 ステップ目以降はファインチューニングが進むごとに各点が自クラスの埋め込みの重心に近づいていき，クラスごとに異なる領域に分かれていっているように見える．

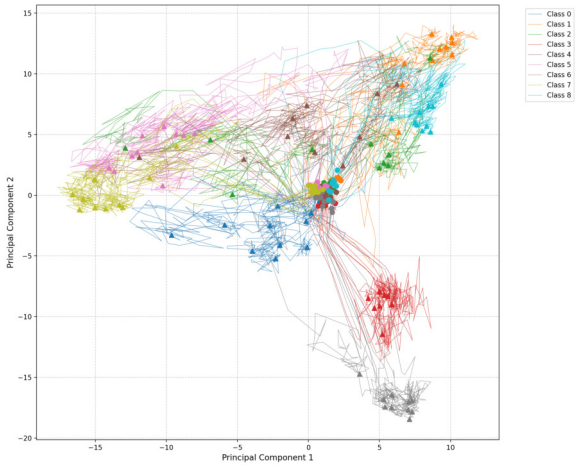

図 5: ファインチューニングにおける埋め込み点の軌跡．各色は属するクラスを表し，丸はファインチューニングをする前の点，三角形はファインチューニング終了時の点を表す．

作成した埋め込みベクトルの軌跡を描画した画像を図 5 に示す．図 5 から，やはりファインチューニングの初期段階では原点付近から離れていく点が多いことがわかる．また，他クラスの点と最後まで一緒に移動する点や，途中まで他クラスの点と共に移動し，やがて自クラスの重心に近づいていく点があることを発見できる．そして，自クラスの重心の近くに存在している点の多くはその場で小さく振動し，逆に自クラスの重心から離れている点の多くはその場で大きく振動しているように見える．

4 提　　案

4.1 課題と方針

予備実験により，ファインチューニングが進行すると，多くの埋め込み点が振動し始めるという性質を確認した．埋め込み点の振動を検出するには，軌跡データマイニングにおける滞留点検出が利用できると考えられる．従来の滞留点検出では時間閾値と空間閾値を設定し対象物の速度の低下が一定期間以上続いたと判定されると滞留点として検出する．この従来手法により 5.5 節で埋め込み軌跡における滞留点を検出した．

しかし，埋め込み点はデータによって速度のばらつきがあるため，空間閾値が小さすぎると点が滞留していてもそれを検出できず，空間閾値を大きくしすぎると滞留していない埋め込み点も滞留点として検出してしまう．図 10 の左上の図は空間閾値を $D = 2$ に設定したときの滞留点検出を表しているが，クラス 2 の埋め込み軌跡のような滞留していて移動速度が比較的大きい点は滞留点として検出されていない．

埋め込み点は，急激に正解率が向上するファインチューニング初期に原点から離れるように大きい速度で直線的に移動し，やがて速度が低下して振動しているように見える．そこで，埋め込み軌跡のこの特徴を利用し，速度のばらつきを考慮した滞留点検出の手法として，直線性指数を滞留点の判定条件に用いた滞留点検出を提案する．

4.2 手　　法

まず，提案手法における用語と，滞留点の判定条件について記述する．

埋め込みを取得する頻度を決定する，微小なステップ数のサンプリング間隔を Δt とする．埋め込みを取得する回数を N とし，ファイチューニング開始から Δt ごとの各時刻 t におけるパラメータ集合を θ_t とする．そして，θ_t に対するデータ d の埋め込みを v_t^d とする．

ウィンドウサイズを k とし，時刻 t から時刻 $t+k$ に対して時間ウィンドウ W を設定して，次式で求めた W における S_t^d が閾値 ϵ より小さいとき，埋め込み v_t^d を滞留点とする．

$$S_t^d = \frac{||v_{t+k}^d - v_t^d||}{\sum_{h=t}^{t+k-1} ||v_{h+1}^d - v_h^d||} \tag{1}$$

滞留点検出の手順について説明する．ファインチューニングにおいて Δt ごとにモデルのパラメータを保存する．クラスの総数を C とし，クラス $j(0 \leqq j \leqq C-1)$ ごとにテストデータセット D から 10 個の記事（データ）をランダムに選択する．選択した各クラス 10 個のデータの集合を D' とする．パラメー

タ集合 $\theta_t (0 \leqq t \leqq N-1)$ に対する各データ $d (d \in D)$ に対する埋め込み v_t^d を取得する．この際，データ d をモデルに入力した結果の隠れ層の最終層における [CLS] トークンの隠れ状態ベクトルを埋め込みベクトルとして取得する．S_t^d を求め，データ点 v_t^d が滞留点であるか判定する．

5 実　　験

埋め込み軌跡に対し，軌跡マイニングの手法を適用し，ファインチューニングを通した埋め込み点の変化を分析する．

本実験では，ステップごとの正解率の推移が異なるクラスが生じるようにするため，不均衡データを用いてファインチューニングを行う．

5.1 ファインチューニングの設定

予備実験と同様に，東北大学の日本語事前学習済み BERT に対し，livedoor ニュースコーパスデータセットを使用して多クラス分類に向けたファインチューニングを行った．

本実験では各クラスの訓練データ数を変更することにより不均衡データにし，テストデータは全クラスでデータ数をすべて均等にした．

訓練データについては，データ数が大（400 件）のクラス，中（100 件）のクラス，小（10 件）のクラスの 3 つのグループを作り，livedoor ニュースコーパスデータセットの 9 クラスをランダムに各グループに 3 クラスずつ割り当てた．その結果，クラス 3, 4, 6 をデータ数大のグループ，クラス 0, 1, 8 をデータ数中のグループ，クラス 2, 5, 7 をデータ数小のグループとした．テストデータについては，アンダーサンプリングを行い，すべてのクラスのデータ数を 45 件とした．

ファインチューニングにおいては，学習率を 2×10^{-5}，バッチサイズを 32，エポック数を 10 にし，オプティマイザとしては AdamW を使用した．

ただし，Warmup やスケジューラは利用せず，学習率を一定にした．

5.2 F1 スコアの変化

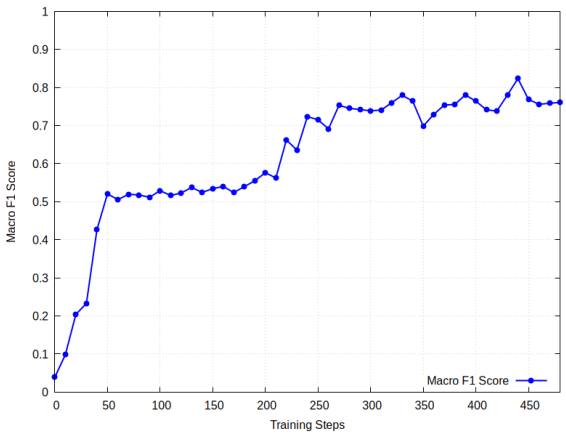

図 6: ファインチューニングにおけるステップ数とマクロ F1 スコアの関係．縦軸はマクロ F1 スコア，横軸はステップ数を表す．

今回は訓練データとして不均衡データをファインチューニングに利用しているため，モデルの性能を評価するために F1 スコアを用いた．まず，ファインチューニングを通したテストデータに対するマクロ F1 スコアの変化を図 6 に示す．図 6 から，マクロ F1 スコアはファイチューニング開始時には 0.0390 であるが，そこから 50 ステップ目の 0.5206 まで急激に増加し，それ以降は 200 ステップ目の 0.5762 まではゆるやかに増加していることがわかる．そして，200 ステップ目から 270 ステップ目の 0.7530 まででもう一度激しく上昇し，270 ステップ目からファインチューニング終了時まで 0.6988 から 0.8234 までの範囲で上下していることがわかる．

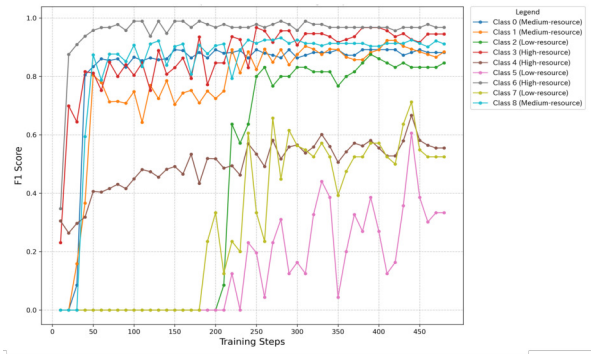

図 7: ファインチューニングにおけるステップ数と各クラスの F1 スコアの変化．各色はクラスを表し，縦軸は F1 スコア，横軸はステップ数を表す．

次に，ファインチューニングの過程における各クラスの F1 スコアの変化を図 7 に示す．図 7 から，データ数の違いに着目すると，データ数の多いクラスほど F1 スコアが最初に急激に上昇するステップ数が小さい傾向があることがわかる．具体的には，データ数大のクラスでは，クラス 3 とクラス 6 は 10 ステップ目から 20 ステップ目にかけて急激に F1 スコアが上昇しており，クラス 4 についても 40 ステップ目から 50 ステップ目にかけて比較的程度は小さいものの上昇している．またデータ数中のクラスでは，概ね 20 ステップ目から 50 ステップ目にかけて急激に F1 スコアが上昇している．そしてデータ数小のクラスでは，概ね 200 ステップ目前後で F1 スコアが急激に上昇している．データ数小のクラスのうち，F1 スコアの最大値が低いクラスは F1 スコアが上下に激しく変動する傾向があることもわかる．具体的には，クラス 4，クラス 5，クラス 7 の F1 スコアは同程度で他クラスと比較すると小さいが，データ数小のクラス 5 とクラス 7 は，データ数大のクラス 4 と比較すると F1 スコアが上下に激しく変動している．図 7 から 9 クラス中，6 クラスは，最終的な F1 スコアが 0.8 を超え，残り 3 クラスは 0.6 を下回っていることがわかる．クラス 4 は，データ数が多いにもかかわらず，F1 スコアが比較的低くなっており，反対にクラス 2 は，データ数が少ないにもかかわらず，F1 スコアが比較的高くなっていることがわかる．以上のことから，データ数によらないクラスごとの本質的な分類難易度が存在することが考えられる．

5.3 軌跡の類似度に基づいたクラスタリング

予備実験と同様に，クラス毎に 10 個データをランダムに選び，その埋め込みベクトルの軌跡を描画した．そして，それら合計 90 個の軌跡データから類似度に基づいてクラスタリングを行い，9 つのクラスタに分けた．また比較のため，ファインチューニング終了時の各データの埋め込みの類似度に基づいてクラスタリングを行い，9 クラスタに分けた．

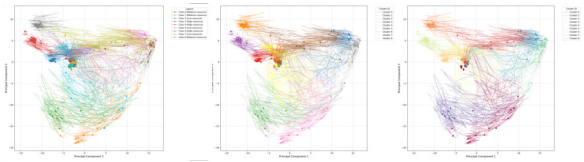

図 8: 埋め込み軌跡のクラスタリングの比較．丸はファインチューニングをする前の点，三角形はファインチューニングを表す．各色は，左の図では属するクラス，中央の図は終了時の埋め込みに基づいて分類されたクラスタ，右の図では軌跡の類似度に基づいて分類されたクラスタを表す．

各埋め込み軌跡の属する正解クラスと，終了時の埋め込みに基づいたクラスタリング結果，軌跡の類似度に基づいたクラスタリング結果を図 8 に示す．予備実験の図 5 では途中まで他クラスのデータ点と共に移動し，やがて自クラスの重心に近づいていくデータ点が見られた．これらはファインチューニングの進行によるモデルの変化を分析する上で重要なデータであると考えられる．予備実験でこのようなデータ点があったことから，本実験において，終了時の埋め込みに基づいたクラスタリングと軌跡の類似度に基づいたクラスタリングで異なるクラスタに属するデータ点が存在することを想定していた．しかし，図 8 から，終了時の埋め込みに基づいたクラスタリング結果と軌跡の類似度に基づいたクラスタリング結果が全く同じになっていることがわかる．これは，ファインチューニングがある程度進行すると，埋め込み点が自身の終了時の埋め込み点の近くで滞留し始めることにより，軌跡の類似度が終了時の埋め込み点の影響を強く受けるからであると考えられる．

5.4 埋め込み点の移動速度の変化

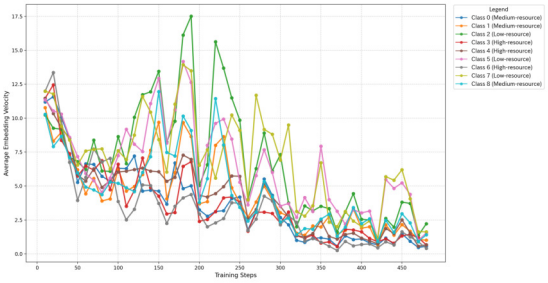

図 9: ファインチューニングにおけるステップ数と各クラスの埋め込み点の移動速度（10 ステップにおける変位長の平均値）の関係．各色はクラスを表し，縦軸は過去 10 ステップの変位長，横軸はステップ数を表す．

各クラスの 10 ステップごとの埋め込みの移動距離の変化を図 9 に示す．図 9 から，200 ステップ目以降は全てのクラスで移動速度（10 ステップあたりの移動距離）が概ね減少していくことがわかる．また，F1 スコアに関係なく，データ数が多いクラスほど移動速度が小さい傾向が見られた．100 ステップ目以降データ数大，中，小の順に移動速度が小さい値で概ね推移しており，F1 スコアが低いデータ数大のクラス 4 の移動速度が，F1 スコアが高いデータ小のクラス 2 の移動速度を下回っている．そして，移動速度が低下することにより滞留が始まることを考慮すると，F1 スコアによらずにデータ数が多いクラスほど滞留が早いということがわかる．

5.5 従来手法による滞留点検出

埋め込み軌跡に滞留点検出を適用した．滞留点検出では，空間閾値 D と時間閾値 T を設定し，対象の点 p の移動距離が時間 T の間連続で D 未満のとき，p を滞留点とする．ここでは，埋め込みを取得するサンプリング間隔ごとの時刻に対し時間閾値を，埋め込みの移動距離に空間閾値を設定する．ただし，サンプリング間隔を 10 ステップごとにして保存しているため，$T = 4$ のときは 10 ステップ間における埋め込み点の移動距離が 4 回連続で D 未満であるときに滞留とみなされる．

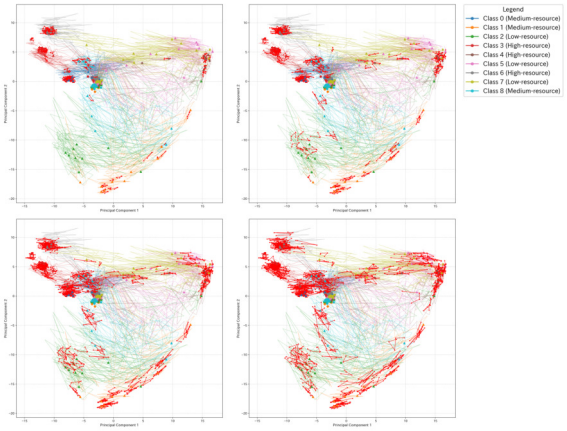

図 10: 移動距離に基づいた滞留点検出．赤色は滞留点として検出された箇所の軌跡，それ以外の各色はクラスを表し，丸はファインチューニングをする前の点，三角形はファインチューニング終了時の点を表す．左上の図が $D = 2$, 右上の図が $D = 3$，左下の図が $D = 4$，右下の図が $D = 5$

表 1: 滞留点検出における時間閾値と滞留点が検出された軌跡の個数の関係

空間閾値 D	滞留点検出軌跡の個数	滞留点検出軌跡率
2	53	0.599
3	74	0.822
4	87	0.967
5	90	1
10	90	1

$T = 4$ のときの滞留点検出の可視化結果を図 10 に示す．また，空間閾値と，滞留点が検出された軌跡（滞留点検出軌跡）

の個数や全 90 個の軌跡のうちの滞留点検出軌跡の割合（滞留点検出軌跡率）の関係を表 1 に示す．

表 1 より，時間閾値が大きいほど滞留点が検出される軌跡の個数が増加しているのがわかる．特に $D = 5$ になると全ての軌跡で滞留点が検出されている．

また図 7 で F1 スコアの最大値が 0.8 を超えているクラスの中では，図 10 において，データ数小のクラスやデータ数中のクラス 8 を除くと $D = 2$ のときに多くの軌跡で滞留点が検出されていることがわかる．そして，F1 スコアの最大値が比較的小さい 3 クラスの中では，唯一のデータ点大のクラスであるクラス 4 のみ滞留点が検出されていることがわかる．そして，データ数に着目すると，F1 スコアの大小に関係なく，データ数大やデータ数中のクラスでは多くの埋め込み軌跡で滞留点が検出されているのに対し，データ数小のクラスでは滞留点が検出されていないことがわかる．そして，空間閾値が大きくなるほどデータ数小のクラスでも滞留点が検出される軌跡が増加していることがわかる．以上の結果から，F1 スコアの最大値に関係なく，データ数が多いクラスほど，従来手法により滞留点が検出されやすいということがわかる．これは，図 9 においてデータ数が多いほど埋め込み点の移動速度が小さい傾向があり，移動が微小な範囲に留まりやすいからだと考えられる．

ただし，クラス 2 はデータ数小のクラスであり，$D = 2$ の時点では埋め込み軌跡で滞留が検出されていない．それに対し，$D = 3$ や $D = 4$ の検出結果で可視化されている検出された滞留点を見るとクラス 2 は多くの軌跡で実際に滞留しているように見える．このことから，データ点ごとの速度のばらつきが大きいために，空間閾値をが小さすぎると滞留していても検出できていないのだと考えられる．

5.6 提案手法による滞留点検出

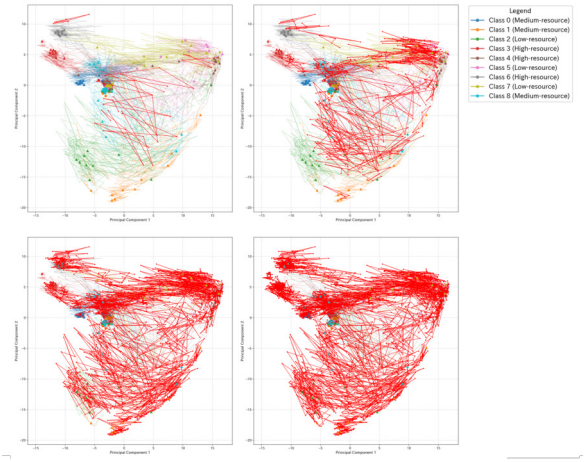

図 11: 提案手法による滞留点検出．赤色は滞留点として検出された箇所の軌跡，それ以外の各色はクラスを表し，丸はファインチューニングをする前の点，三角形はファインチューニング終了時の点を表す．左上の図が $\epsilon = 0.1$, 右上の図が $\epsilon = 0.15$，左下の図が $\epsilon = 0.2$，右下が $\epsilon = 0.25$ のとき．

ウィンドウサイズ k を 5 とし，提案手法による滞留点検出を行った．チェックポイントを 10 ステップごとにしてモデルを保存しているため，埋め込み点を滞留点か判定する際には，50 ステップ先の点との直線距離を 50 ステップ先までの総距離で割ったものを直線性指数として求める．提案手法による滞留点検出の可視化結果を図 11 に示す．また，直線性指数に対する ϵ と，滞留点検出軌跡個数や滞留点検出軌跡率の関係を表 2 に示す．

表 2: 滞留点検出における時間閾値と滞留点が検出された軌跡の個数の関係

閾値 ϵ	滞留点検出軌跡の個数	滞留点検出軌跡率
0.1	27	0.3
0.15	41	0.456
0.2	71	0.789
0.25	90	1

表 2 より，ϵ が大きいほど滞留点が検出される軌跡の個数が増加していて，$\epsilon = 0.2$ になると全ての軌跡で滞留点が検出されている．また図 11 から，$\epsilon = 0.1$ では，滞留として検出されている軌跡の中にほとんど滞留とみなせるものがないということがわかる．また $\epsilon = 0.15$ や $\epsilon = 0.2$，$\epsilon = 0.25$ になると滞留とみなせるものも検出されているが，そうでないものも同時により多く検出されていることがわかる．以上のことから，提案手法による滞留点検出は誤検出が非常に多いと考えられる．

5.7 直線性指数の変化

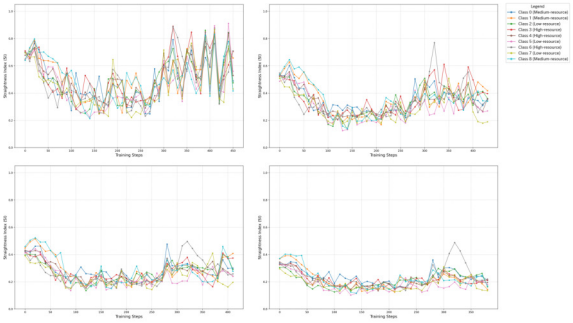

図 12: ファインチューニングにおけるステップ数と各クラスの直線性指数の平均値の変化．各色はクラスを表し，縦軸は直線性指数，横軸はステップ数を表す．左上の図が $k = 3$，右上の図が $k = 5$，左下の図が $k = 7$，右下の図が $k = 10$ のとき．

ファインチューニングにおける直線性指数の変化を図 12 に示す．図 12 から，すべてのウィンドウサイズについて，全てのクラスの直線性指数が，0 ステップ目から 100 ステップ目にかけて概ね減少していき，そこから 250 ステップ目までは大局的には値があまり変化せず，250 ステップ目から 300 ステップ目にかけて増加していき，その後また値が大局的に保たれるという変化の仕方をしていることがわかる．また，ウィンドウサイズが大きいほど直線性指数のとる範囲が小さく，変化もゆるやかであることがわかる．そして，300 ステップ目以降は学習が進行し，F1 スコアが安定しているクラスが多いが，直線性指数は下がらず，比較的高い値となっていることがわかる．以上

のことから，F1 スコアの推移が異なるクラスが存在するデータセットであるにもかかわらず，全クラスで概ね共通した性質を持っていることがわかる．このことから，直線性指数を判定条件に用いた提案手法により誤検出をせず滞留点を検出することは困難であると考えられる．

6 まとめと今後の活動方針

本研究では，ファインチューニングにおける埋め込み点の移動軌跡に着目し，ファイチューニング過程を分析した．

それにより，多くの埋め込み軌跡にはファインチューニングが進行すると滞留し始めるという性質があることを発見した．そこで，埋め込み軌跡の類似度は，滞留によりファインチューニング終了時の埋め込み点の影響を強く受けてしまうという課題も確認された．埋め込み軌跡から滞留している部分を取り除いたり，埋め込み軌跡の簡略化したりすることで対応できると考えられる．

そして，埋め込み軌跡の特徴から，直線性指数を利用して，埋め込み軌跡に特化した滞留点検出手法を提案した．この提案手法で実際に滞留点検出をすると，誤検出が多い結果となった．今後，速度のばらつきを考慮した滞留点検出をできるように，手法を改善していく必要がある．

さらに，埋め込み軌跡に対する分析結果から，データ数が多いクラスほど F1 スコアの高さに関係なく埋め込みの移動速度が小さくなりやすいという性質を確認した．また，データ数が少ないクラスは滞留点検出では滞留点の検出が困難であるということも確認した．このことを踏まえて，ファイチューニングの最適化支援に向けた滞留点検出の活用方針をまとめる．滞留点検出を利用することで，分類性能スコアのみでは識別できない，データを追加しても性能向上しないクラスを識別可能であると考えられる．滞留が早いクラスの場合，分類性能が高ければ，そのクラスは学習する上で理想的な状態にあるといえるが，分類性能が低ければ，学習が本質的に難しい状態にあると想定できる．このような滞留が早く分類性能が低いクラスは特徴量の再設計やラベルの再定義により改善できる可能性がある．そして，滞留が遅いクラスの場合，分類性能が高ければ汎化性能が低く分類性能が低ければデータ不足であると考えられる．このような滞留が遅いクラスは重点的なデータ追加による改善できると考えられる．今後，この活用方針に従って，ファイチューニングの最適化支援を行う実験をする予定である．

文　献

[1] Yichu Zhou and Vivek Srikumar. A closer look at how fine-tuning changes bert. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1046–1061, 2022.

[2] Yichu Zhou and Vivek Srikumar. DirectProbe: Studying representations without classifiers. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 5070–5083, 2021.