転移学習による都営バスのリアルタイム運行データを用いた渋滞検知

藤田 智也† 青柳 宏紀‡ 畠中 希§ 小口 正人¶ 山名 早人#

†早稲田大学 基幹理工学部 〒169-8555 東京都新宿区大久保 3-4-1
‡早稲田大学大学院 基幹理工学研究科 〒169-8555 東京都新宿区大久保 3-4-1
§お茶の水女子大学大学院 人間文化創成科学研究科 〒112-8610 東京都文京区大塚 2-1-1
¶お茶の水女子大学 理学部情報科学科 〒112-8610 東京都文京区大塚 2-1-1
#早稲田大学 理工学術院 〒169-8555 東京都新宿区大久保 3-4-1

E-mail: †‡# {tfujita, aoyagih, yamana}@yama.info.waseda.ac.jp,
§nozomi-h@ogl.is.ocha.ac.jp ¶oguchi@is.ocha.ac.jp

あらまし 我が国の渋滞による経済損失は年間 12 兆円に上る．対策の一つは渋滞情報の利用者への提供であり，道路上のセンサーや GPS データを用いた渋滞検知が実現されている．これに対し，本研究では，渋滞検知のコストを抑えると共に利用者がよく利用する道路（バスが定期運行する道路）を対象とした渋滞検知を目指す．具体的には，バスの運行データ（バス停発車時刻データ）と機械学習を組み合わせた渋滞検知手法を提案する．本手法では，（1）バス停発車時刻だけでは停留所や信号での停車時間が把握できない点，（2）渋滞時のデータが少ない点，が渋滞検知精度向上における課題となる．これらの課題に対し，本研究では，類似する渋滞傾向を持つバス停区間をクラスタリングし，クラスタ毎に構築した学習モデルをもとに，バス停区間毎に転移学習を行い渋滞検知精度向上を図る．評価実験では，オープンデータとして公開されている都営バスの運行データを用い，提案する転移学習手法により，渋滞検知精度を向上させることができることを確認した．16 系統の都バスに対し，最も高い渋滞検知性能を示したクラスタでは，バス停区間毎の F1 値のマイクロ平均 F1 値 0.881 を達成した．一方，最も低い渋滞検知性能を示したクラスタでは，マイクロ平均 F1 値 0.179，全バス停区間のマイクロ平均 F1 値 0.431 となった．

キーワード 渋滞，オープンデータ，バス，公共交通機関，機械学習，統計分析，異常検知，分類，転移学習

1. はじめに

国土交通省によれば，我が国の渋滞による経済損失は，年間 12 兆円に上り，時間換算で年間 30 時間／人の損失がある[1]．特に，交通渋滞が引き起こす物流への支障や，排気ガスによる大気汚染，緊急車両の遅延は，社会に対して大きな悪影響をもたらしている．渋滞対策の一つは，運転手に対して渋滞情報をリアルタイムに伝えることである．しかし，従来の渋滞検知手法は，車の走行位置情報や道路上のセンサーに依存し，プライバシーや実現コストの大きさが問題となっている．

一方，近年では公共交通機関に関するデータがオープンデータとして公開され，様々な形で利活用されている．公開されているデータの中には，バスのリアルタイムのロケーションデータが含まれており，渋滞検知に利用できる可能性がある．

青柳ら[1]は，2022 年に，バスの運行データと機械学習を組み合わせた自動渋滞検知の手法を提案し，渋谷から池袋までの明治通りの区間を対象として，一定時間ごとに「渋滞」と「非渋滞」の 2 値分類を行った．結果，特定の停留所区間においては，F1 スコア 0.742 を達成したが，一部の停留所区間においては精度が低下した．原因として，非渋滞のデータ数に対して渋滞のデータ数が大幅に少ないといったデータ数の偏りによる検知精度の低下が考えられる．

本研究は，青柳らの渋滞検知精度を向上させるために，学習データに渋滞データが少ない場合の渋滞検知精度の改善を目指す．具体的には，渋滞データのオーバーサンプリング，及び，他区間の渋滞データをもとにした転移学習により，渋滞検知精度の向上を行う．

以下，2 節で関連研究を紹介し，3 節で提案手法について説明し，4 節で評価実験を行い，5 節でまとめる．

2. 関連研究

本節では，関連研究について述べる．

2.1 プローブ車両の GPS データを用いた交通速度推定

Samal ら[2]は，2017 年，「Tennessee Department of Transportation」から収集したバスの GPS プローブデータから，交通速度推定を行う手法を提案した．一般に，

[1] https://www.mlit.go.jp/road/ir/ir-perform/h18/07.pdf

バスの速度は交通の平均速度を完全には表しておらず，またバス路線によっては運行頻度が低く，データがスパースになるという問題がある．そこで，Samal らは，気象データや過去の交通データを利用し，精度向上を目指した．k-means 法により，類似した運行データについてクラスタリングを行い，クラスタを単位としてランダムフォレストによる学習を行うことで予測精度を改善した．これにより，交通速度の平均二乗誤差が，毎時 4.0~4.5 マイル（クラスタリング適用前）であったのに対し，毎時 2.9~3.3 マイルとなることを示した．

Kyaw ら[3]は，2018 年，バスの GPS データから，走行速度推定モデルを構築した．特徴量として，道路の混雑要因となる道路付近のレストランやショッピングモールといった POI（Place of Interests）の数を追加し，ヤンゴン市の 21 番バスの 24 時間 365 日の道路区間毎の速度係数マトリックス（走行速度と交通遅延の関連を示す）を作成している．しかし，交通速度推定の精度の定量的な評価はなされていない．

Gu ら[4]は，2020 年，バスの GPS 軌道データをもとに，自己組織化写像（SOM, Self-Organizing Map）を用いて，バス停区間毎の道路セグメントの交通渋滞を推定する手法を提案した．バスの停留所での停車時間，バスの平均速度，乗客のバス停間の旅行時間を入力として，ニューラルネットワークを構築した．本手法によって，推定した交通速度と交通速度を代表するタクシー速度との相関係数が，0.94 となり，両者の間に高い相関があることを示した．

2.2 プローブ車両の GPS データを用いた渋滞検知

Xu ら[5]は，2012 年，複数台のバスの平均移動時間を算出し渋滞検知を行う手法を提案した．Xu らは，T 分間に渋滞検知対象道路を通過したバスの平均移動時間を算出する「T-window Average」と，道路区間を直近に通過した N 台のバスの平均移動時間を算出する「N-window average」を定義し，それぞれの指標が予め個別に設定した閾値を超えた場合に，渋滞が発生していることをシミュレーションにて示した．

Wang ら[6]は，2013 年，1 台の車両からの GPS プローブデータを用い，機械学習にて交通状態を分類する手法を提案した．Wang らは，GPS によって取得した車両の平均速度を位置ごとに離散化し，特徴ベクトルとして用いた．学習モデルは，ランダムフォレスト，Adaboost, SVM を用いており，それぞれ 91.59%, 89.43%, 87.86%の精度で検知できることを示した．

Carli ら[7]は，2015 年，バスの GPS プローブデータによる効率的な自動渋滞検知手法を提案した．バスの平均速度やバスが閾値を超えた速度で走行した時間，といった指標を計算し，都市部内の車両の動きを監視することで，信号機故障の異常を検出した．

2.3 バスの運行データを用いた渋滞検知

一般的に，プローブ車両の GPS データを取得し続けることは困難である．そこで，青柳ら[1]は，2022 年，都バスの運行データ（バス停発車時刻）を用いたリアルタイム渋滞検知手法を提案した．青柳らは，都バスの停留所発車時刻のデータから，各バス停区間のバスの平均速度と同平均速度から算出される統計量を特徴量として抽出し，機械学習による渋滞検知を 20 分間隔で行った．バス路線内の全バス停区間のデータを学習させたモデルと，検知対象のバス停区間のデータのみを学習させたモデルの精度を比較したところ，後者のモデルの検知精度の方が高い結果となった．特定の区間において，F1 スコア 0.742 を得たが，一部の停留所区間においては精度が低下した．この原因として，非渋滞のデータ数に対して渋滞のデータ数が大幅に少ない区間があり，データ数の偏りによる検知精度の低下が考えられる．

2.4 関連研究まとめ

GPS データを用いることで渋滞検知が可能となることがいくつかの論文[5][6][7]で示されている．一方，GPS を各車両に備えデータを収集するコストは大きい．一方，我が国においては，バス運行情報の提供は「バス停発車時刻」に留まっている場合が多い．そこで，本稿では，[1]の成果をもとに，バスのバス停発車時刻のみを用いた渋滞検知の精度向上に取り組む．特に，[1]での未解決の問題である「データ数の偏りによる精度の低下」を克復することを目指す．

3. 提案手法

3.1 概要

青柳ら[1]の，バスの各停留所の発車時刻データを用いた渋滞検知手法をもとに，バス停区間ごとの渋滞と非渋滞のデータ数の偏りに対処するための手法を提案する．渋滞検知では，国家公安委員会の渋滞定義に基づき，渋滞検知区間を走行する自動車が「時速 10km 未満で継続的に走行している状態」を渋滞と定義する．

具体的には，連続する 2 つのバス停の出発時間の差をバスの移動時間とし，2 つのバス停間の距離を移動時間で割ることで，バスの速度を求める．ここで注意すべきは，同移動時間には「バス停での停車時間」や「信号による停車時間」が含まれている点であり，バスの速度のみで渋滞／非渋滞を判断できない点である．

次に，算出した速度と該当するバス停区間の渋滞／非渋滞を示す正解データを用いて学習を行う．この時，学習データに含まれる渋滞データ数と非渋滞データ数に偏りがあるため，オーバーサンプリングを適用する．さらに，全バス停区間のデータを用いた渋滞検知モデルを個々のバス停区間の渋滞検知に適用させ（転移学習），精度向上を目指す．

なお，本研究では渋滞検知の最小単位をバス停 2 区間とする．これは，バス停区間 1 区間を対象とした場合，300m 程度の短区間に対する検知となり，当該区間内に存在する信号待ち時間やバス停停車時間等のノイズの影響が大きく渋滞検知が困難となるためである．

3.2 特徴量抽出

提案手法で用いる記号を表 3.1 と図 3.1 に示す．

表 3.1 特徴量抽出に用いる記号

記号	定義
b_i	i 番目のバス
p_j	j 番目のバス停
s_j	p_jからp_{j+1}の j 番目の区間
l_j	p_jからp_{j+1}の区間s_jの距離[m]
td_{ij}	b_iのp_jの出発時刻
Δtd_{ij}	b_iの区間s_jにかかる所要時間[s]
v_{ij}	b_iのs_jでの平均速度[m/s]
c_{ij}	b_iがp_jを出発する時刻を 20 分ごとに分類した指標．0 時 0 分 0 秒から 0 時 19 分 59 秒を 0 として，以後 20 分おきに 1, 2, 3, …と定義する．

※バスb_iは出発時刻順に，バス停p_jは経路順に整列しているとする．

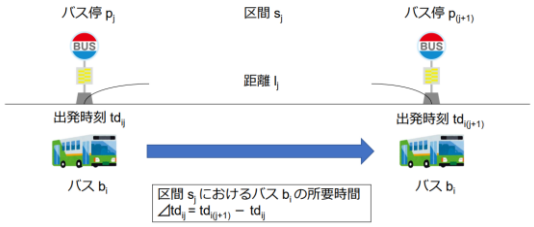

図 3.1 特徴量抽出に用いる記号の説明

本手法で用いるデータは，バスb_iがバス停p_jを時刻td_{ij}に出発したデータである．また，各区間s_jの距離l_jは正解データ取得元である，Google Directions API[2]から取得する．Δtd_{ij}は出発時刻td_{ij}と$td_{i(j+1)}$から次式(1)で表される．

$$\Delta td_{ij} = td_{i(j+1)} - td_{ij} \quad (1)$$

次に，バスb_iの区間s_jにおける平均速度v_{ij}を算出する．平均速度v_{ij}は次式(2)で表される．

$$v_{ij} = \frac{l_j}{\Delta td_{ij}} \quad (2)$$

次に，バスb_iのバス停p_jの出発時刻td_{ij}を 20 分ごとの時間帯c_{ij}に分類する．c_{ij} の値は，0 時 0 分 0 秒から 0 時 19 分 59 秒を 0 として，以後 20 分おきに 1, 2, 3, …と定義する．

本手法で用いる特徴量を表 3.2 に示す．

表 3.2 本手法で用いる特徴量

バスb_iのバス停区間s_j走行時の速度	v_{ij}
一つ前のバスb_{i-1}の同一バス停区間s_j走行時の速度	$v_{(i-1)j}$
バスb_iの区間s_j走行時の時間帯	c_{ij}

上記で算出したv_{ij}，$v_{(i-1)j}$，c_{ij}を特徴量として用いる．v_{ij}と$v_{(i-1)j}$はそれぞれ区間s_jにおけるバスb_iの速度，バスb_{i-1}の速度を表す．バスの速度は渋滞判定の基準となる交通速度との相関があると考えられ，特徴量として有用である．また，c_{ij}はバスb_iがバス停p_jを出発する時間を，20 分ごとに分類した指標である．時間帯により，交通量の変化があると考えられるため，c_{ij}も特徴量として有用であると考えられる．[1]では，20 分の時間ウィンドウ内に通過したバスの平均速度と統計量を用いて渋滞予測を行っていた．これに対し，本稿では，バス系統によってバス本数が大きく異なるため，バスの通過ごとに特徴量を抽出し，渋滞検知を行う．

3.3 データ前処理

バスの平均速度が渋滞判断基準の 10km/h を超える場合，学習器で判断せず非渋滞と判断することができると考えられる．しかし，図 3.2 に示す通り，実際には 10km/h を超える場合も，渋滞と判断しなければならない事例がある．これは，「バス運転手がバス停の出発時に出発ボタンを押すのが遅れた場合」「判定対象のバス停区間が長く，バス停区間の一部のみが渋滞している場合」等が理由となる．

そこで，学習器が渋滞／非渋滞を判断する対象データを閾値以下の速度とし，学習器の学習時及び判定時（テスト時）の対象データを絞り込む．閾値として，渋滞データの割合が全体の 5%未満になる値のうち，最小の速度（閾値は整数）となるように設定する．つまり，閾値を超える平均速度のデータは，学習器を用いず，常に非渋滞に分類する．図 3.2 に全データと渋滞データの分布，渋滞データの逆累積分布度数を示す．

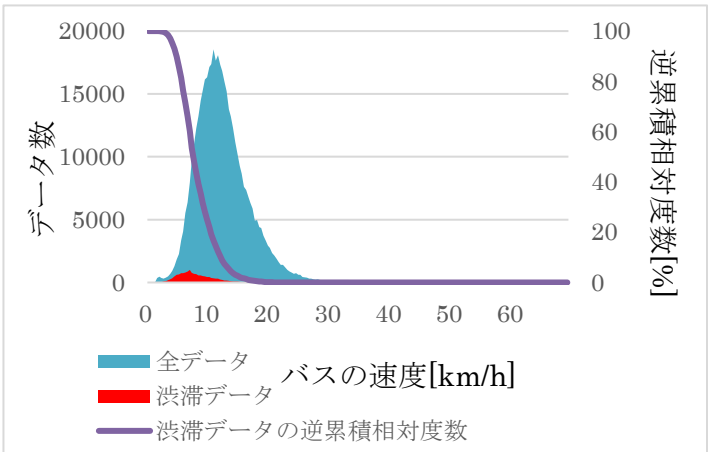

図 3.2 全データと渋滞データ分布（左目盛）と渋滞データの逆累積相対度数（右目盛）

3.4 オーバーサンプリング

全データに占める渋滞データの割合が小さいため，オーバーサンプリング（SMOTE[8]を使用）を適用する．

[2] https://developers.google.com/maps/documentation/directions

3.2 で示した特徴量を抽出し，各モデルに用いる学習データごとに非渋滞データ数と渋滞データ数が均等になるように,渋滞データをオーバーサンプリングする．SMOTE には，Python の imbalanced-learn ライブラリの imblearn.over_sampling.SMOTE を利用する．ここで，渋滞とは,「時速 10km 未満で継続的に走行している状態」を指す．また，学習データ中に含まれる渋滞データ数が 1 以下の場合，オーバーサンプリングできない．ベースライン手法，提案手法の何れかでオーバーサンプリングが不可能となる区間は，実験対象から外す．

3.5 分類手法

学習器として，ニューラルネットワークモデルを採用する．バス停区間毎の渋滞データが少ないため，(1)バス停区間毎ではなく，全バス停区間で学習器を構築した上で，(2)個々のバス停区間のデータを用いて転移学習する．これにより，渋滞検知の精度向上を狙う．

具体的には．3.4 にて，オーバーサンプリングしたデータに対して，全バス停区間のデータを用いて学習する．次に，個々のバス停区間毎に転移学習を行う．最終的に，バスb_iが走行した，バス停区間s_jにおいて「渋滞」「非渋滞」の二値分類を行う．ニューラルネットワークでは，出力結果が確率となるため，閾値を設定した上で，閾値よりも値が大きい場合は渋滞，そうでない場合は非渋滞とする．

実装は Python で行い，Python ライブラリである，Keras のバージョン 2.11.0 を用いる．各ノードでは，活性化関数として Sigmoid 関数を用い，損失関数として Binary cross entropy を用いる．分類は，以下の 4 ステップで行う．

ステップ 1:図 3.3 に示す入力層，隠れ層，出力層を持つニューラルネットワークを構築する．ここで，入力する特徴量は 3 種類，出力する渋滞の予測結果の値は 1 つであるため，入力層，出力層はそれぞれ 3 ノード，1 ノードとする．隠れ層は 9 ノードとする．入力層，出力層，隠れ層は全て全結合層で実装する．作成したニューラルネットワークに対し，全バス停区間のデータを用いて学習を行う．

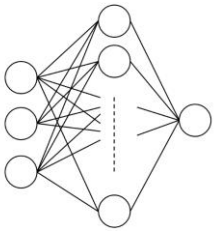

図 3.3 学習モデル（全バス停区間で学習）

ステップ 2:学習後，ニューラルネットワークの出力層を削除（図 3.4 の青色部分）する．

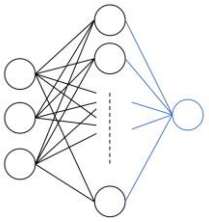

図 3.4 出力層を削除したモデル

ステップ 3:ステップ 2 で得られた入力層と隠れ層からなる図 3.5 の緑色で示した部分に対し，パラメータの重みづけを固定する．

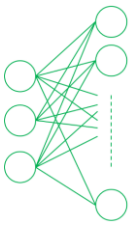

図 3.5 パラメータの重みづけを固定したモデル

ステップ 4:図 3.6 に示すように，ステップ 3 で得られたモデルに，新たに全結合層である隠れ層と出力層を追加し，バス停区間ごとに学習を行う．

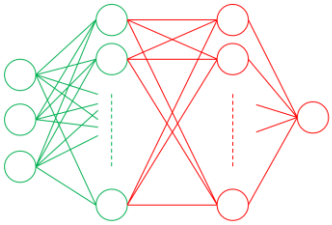

図 3.6 新たに隠れ層と出力層を追加したモデル

なお，ベースライン手法では，図 3.2 に示したモデルを用い，バス停区間ごとに学習を行う．

4. 評価実験

4.1 渋滞検知対象区間

渋滞検知の対象区間として，都営バスのうち 16 系統をランダムに選出し，その系統のバス停区間を走行した全バスのデータを用いて渋滞検知を行う．表 4.1 に渋滞検知の対象となる都営バスの 16 系統を示す．なお，オープンデータとして得られるデータが，バス停出発時刻のデータのみであるため，終点のバス停については，出発時刻を得ることができず，終点区間に対しては，渋滞検知対象外とする．

表 4.1 渋滞検知の対象となる都営バスの系統

白 61，都 02，海 01，学 02，学 05，池 86，王 41，早 77，池 65，新小 22，王 45，高 71，練 68，東 15，門 19，平 23

4.2 データセット

4.2.1 バスの運行データの取得方法

バスの運行データは，公共交通オープンデータ協議会が公開している「公共交通オープンデータセンター」

[3] の API を用いて取得した．表 4.2 に取得したバス運行データの一部を示す．

表 4.2 取得したバス運行データ（一部）

日付	車両番号	出発バス停留所識別子	バス停出発時刻
2022-11-30	C220	SodaiRiko.1051.2	10:23:49
2022-11-30	E397	ShinjukuYonchome.714.8	10:21:58

4.2.2 渋滞ラベルの取得方法

渋滞ラベルの正解データは，Google Maps Platform の Directions Advanced API[4] にリクエストを送り収集した．データ収集期間は，「2022 年 11 月 30 日から 12 月 13 日の 2 週間」と「12 月 17 日から 12 月 31 日の 2 週間」とし，頻度は 5 分おきとした．

Directions Advanced API では，地図上の出発地点と到着地点を緯度経度により指定すると，2 点間の道のり（m）と，リアルタイム交通状況を踏まえた所要時間（s）を含むデータを返す．出発地点と到着地点に，それぞれバス停b_iとバス停b_{i+1}の緯度経度データを入力することによって，バス停間の道のりとリアルタイムの所要時間を取得できる．これらのデータの一部を表 4.3 に示す．なお，経過時間は，当該区間の一般車両通過時間を示す．対象区間での交通速度は，距離を経過時間で割ることで計算する．正解渋滞ラベルは，交通速度が渋滞の定義である時速 10km 未満であれば渋滞とし，それ以外を非渋滞とする．

渋滞ラベルデータは，API から定期的に取得する都合上，バスが実際に走行する時刻とタイムラグがあるため，「バス停出発時刻」より後のデータかつ出発時刻に最も近いデータを採用する．

表 4.3 取得した渋滞ラベルデータ（一部）

日付	取得時刻	バス停始点	バス停終点	経過時間 [s]	距離[m]
2022/11/30	09:00:00	池袋駅東口	南池袋一丁目	158	615
2022/11/30	09:00:00	新宿四丁目	日清食品前	159	787

4.2.3 データセットの作成

本実験では，バス停区間を走行したバスデータ（表 4.2）と，渋滞ラベルデータ（表 4.3）を用いて，「渋滞」と「非渋滞」の二値分類を行う．二値分類を行う上で，必要となるデータセットを表 4.4 に示す．表 4.5 には，作成したデータセットのうち，池 86 系統についてバス停区間毎に渋滞ラベルの内訳をまとめた．1 データは，その区間を走行したバス 1 台分の運行データに対応する．

表 4.4 作成したデータセット(一部)

日付	A 出発時刻	B 出発時刻	始点のバス停(A)	終点のバス停(B)	経過時間 B－A [s]	渋滞フラグ
2022/11/30	10:30:49	10:35:12	KitaSando.404.1	ShinjukuYonchome.714.8	263	0
2022/11/30	15:48:05	15:52:49	ShinjukuIsetan.705.7	SendagayaGochome.832.2	284	1

表 4.5 渋滞ラベル数（池 86 系統）

始点	終点	渋滞	非渋滞	計
池袋サンシャインシティ	池袋駅東口	8	692	700
池袋駅東口	南池袋一丁目	205	1,125	1,330
南池袋一丁目	学習院下	0	1,376	1,376
学習院下	学習院女子大学前	53	1,327	1,380
学習院女子大学前	新宿コズミックセンター前	4	1,303	1,307
新宿コズミックセンター前	東新宿駅前	39	1,358	1,397
東新宿駅前	新宿伊勢丹前	321	1,042	1,363
新宿伊勢丹前	千駄ヶ谷五丁目	103	1,263	1,366
千駄ヶ谷五丁目	千駄ヶ谷小学校前	0	1,378	1,378
千駄ヶ谷小学校前	表参道	206	1,157	1,363
表参道	神南一丁目	17	1,372	1,389
神南一丁目	渋谷駅東口	1,204	184	1,388
渋谷駅東口	神宮前六丁目	97	1,167	1,264
神宮前六丁目	神宮前一丁目	35	1,367	1,402
神宮前一丁目	北参道	0	1,406	1,406
北参道	新宿四丁目	21	1,356	1,377
新宿四丁目	日清食品前	259	1,127	1,386
日清食品前	大久保通り	61	1,357	1,418
大久保通り	早大理工前	25	1,349	1,374
早大理工前	高田馬場二丁目	173	1,246	1,419
高田馬場二丁目	千登世橋	0	1,430	1,430
千登世橋	南池袋三丁目	56	1,356	1,412
南池袋三丁目	東池袋一丁目	193	549	742
全区間合計		3,080	27,287	30,367

4.3 実験 1: 都バス路線 1 系統を用いた分類

実験 1 では，学習時に複数の系統のデータを用いて学習器を構築した場合と，単一の系統のデータを用いて学習器を構築した場合との性能を比較し，他系統のデータを用いた転移学習により性能向上が期待できるかどうか確認する．本実験では，池 86 系統を 1 系統として選出する．

4.3.1 データセット

以下の 4 種類のデータセットを用意する．何れも 2022 年のものである．なお，渋滞データの割合が 0.5%

[3] https://developer-dc.odpt.org/ja/info

[4] https://developers.google.com/maps/documentation/directions

未満となる「バス平均速度が時速 11km 以上のデータ」を学習の対象外とした.

- データ A：11 月 30 日～12 月 6 日
- データ B：12 月 7 日～12 月 13 日
- データ C：12 月 18 日～12 月 24 日
- データ D（テスト用）：12 月 25 日～12 月 31 日

本実験では，バスの平均速度が時速 11km 以上のデータを除外した．全渋滞データ数は 3,080 であり，時速 11km 以上で渋滞ラベルを持つデータは 3.9%である.

4.3.2 パラメータ調整

構築したニューラルネットワークの出力結果に対し，ROC 曲線を作成し，座標（0,1）に最も近い点のときの閾値を採用し，閾値よりも大きい場合は渋滞，閾値以下の場合は非渋滞と分類する．ROC 曲線の作成には， sklearn.metrics.roc_curve（scikit-learn ライブラリ）を用いた.

4.3.3 ベースライン手法

実験 1 の目的は，転移学習の効果を確認することである．このため，ベースライン手法では転移学習を適用せず,「池 86 系統のデータ C でバス停区間毎に学習した学習器（バス停区間毎)」と「池 86 系統のデータ A，B，C でバス停区間毎に学習した学習器（バス停区間毎)」の 2 つを用意する.

実験結果（Accuracy，Precision，Recall，F1 スコア）を表 4.6 に示す．表 4.6 は,「池 86 系統のデータでバス停ごとに学習した学習器（バス停区間毎)」 による全区間のマイクロ平均である.

表 4.6 バス停区間毎の学習器を用いた渋滞検知結果（対象：池 86 系統）のマイクロ平均

バス停区間データ	Accuracy	Precision	Recall	F1 スコア
データ C	0.761	0.453	0.780	0.573
データ A，B，C	0.728	0.439	0.852	0.579

4.3.4 提案手法

転移学習を用いて学習を行う．3.5 で示した手順に従い，データ A，B の全バス停区間（池 86 系統内）を区別せずに学習し（ステップ 1)，全結合層の重みを固定し（ステップ 2，3)，新たに全結合層を追加したものに対して，データ C を用いて，バス停区間毎に個別に学習する（ステップ 4).

ここで，転移学習時に「性質の異なるデータ」が混ざると渋滞検知精度が下がる可能性があることを考慮し，渋滞データの割合が大きい「神南一丁目→渋谷駅東口」区間（表 3.5 参照）を特異な区間として，ステップ 1（3.5 参照）から除外する場合と除外しない場合の 2 種類の結果を表 4.7 に示す. 評価手法は 4.3.3 と同一である.

表 4.7 全バス停区間の学習器を元に各バス停区間の学習器に転移学習した場合の渋滞検知結果（池 86 系統）のマクロ平均

	Accuracy	Precision	Recall	F1 スコア
特異な区間を転移学習から除外した場合	0.757	0.452	0.863	0.593
同・除外しない場合	0.712	0.395	0.754	0.518

表 4.6 と表 4.7 を比較すると，転移学習時に特異な区間を除外することでベースラインに比較して渋滞検知性能を向上させることができる（F1 値で 0.579 から 0.593 へ向上）ことが分かる．つまり「転移学習は渋滞検知の性能向上に貢献するが，転移学習時には，渋滞傾向が似た区間をまとめて転移学習することが必要である」ことが分かる.

4.4 実験 2: 都バス路線 16 系統を用いた分類

実験 2 は，同一系統内の複数バス停区間のデータだけでなく，異なる 16 系統のバス停区間データを用いた転移学習による効果を確認することを目的とする.

4.4.1 データセット

以下の 4 種類のデータセットを用意する．何れも 2022 年のものである.

- データ A：11 月 30 日～12 月 6 日
- データ B：12 月 7 日～12 月 13 日
- データ C：12 月 18 日～12 月 24 日
- データ D：12 月 25 日～12 月 31 日（テスト用）

本実験では，バスの平均速度が時速 14km 以上のデータについて学習器での判定対象外とした．なお，全渋滞データ数は 14,900 であり，時速 14km 以上のデータ内での渋滞データ割合は 3.7%である.

4.4.2 パラメータ調整

構築したニューラルネットワークの学習データに対する出力結果に対し,ROC 曲線を作成し,座標(0,1)に最も近い点のときの閾値を採用し，閾値よりも大きい場合は渋滞，閾値以下の場合は非渋滞と分類する．ROC 曲線の作成には，scikit-learn ライブラリの sklearn.metrics.roc_curve を用いた.

4.4.3 ベースライン手法

ベースライン手法として,転移学習を適用せず,「4.1 で述べた 16 系統のデータ C でバス停区間毎に学習した学習器（バス停区間毎)」と「4.1 で述べた 16 系統のデータ A，B，C でバス停区間毎に学習した学習器（バス停区間毎)」の 2 つを用意する.実験結果(Accuracy，Precision，Recall，F1 スコア）を表 4.8 に示す.

表 4.8 バス停区間毎の学習器を用いた渋滞検知結果（対象：16 系統）のマイクロ平均

バス停区間データ	Accuracy	Precision	Recall	F1
データ C	0.817	0.296	0.717	0.419
データ A，B，C	0.800	0.282	0.763	0.412

表 4.8 は,「4.1 で述べた 16 系統のデータでバス停区間毎に学習した学習器（バス停区間毎）」による全区間のマイクロ平均である．

4.4.4 提案手法１－「バス停区間を通過した全バスの平均速度」と「バス停区間毎の渋滞データの割合」によるクラスタリング

4.3.4 の結果を踏まえ，16 系統の全バス停区間を類似するバス停区間毎にクラスタリングした上で転移学習を適用する．クラスタリングにあたっては,「バス停区間を通過した全バスの平均速度」と「バス停区間毎の渋滞データの割合」を用い，最小値が 0 最大値が 1 となるように正規化しする．クラスタリングには，sklearn.cluster.KMeans（kmeans 法）を用いる．クラスタ数は 3，5，6，7，8，9，10，15 で実験し，最もスコアの良い 8 を実験結果として示す．

図 4.1 は，kmeans 法により全バス停区間をクラスタリングした結果である．Y 軸が，渋滞データの割合を正規化したもの，X 軸がバス停区間におけるバスの平均速度を正規化したものである．

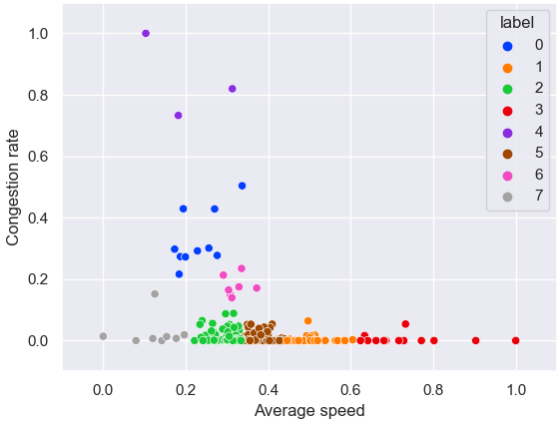

図 4.1 バス停区間のクラスタリング結果（バス停区間平均速度と渋滞割合によるクラスタリング）

なお，4.3.4 と同様に,「A,B の全バス停区間（16 系統）のデータで学習した学習器」に対して「データ C を用いてバス停区間毎に転移学習を行った学習器（バス停区間毎）」を用いる．表 4.9 に結果を示す．

表 4.9 全バス停区間（16 系統）の学習器を元に各バス停区間の学習器に転移学習した場合の渋滞検知結果のマイクロ平均

Accuracy	Precision	Recall	F1 スコア
0.811	0.298	0.779	0.431

表 4.9 に示す通り，ベースライン手法と比較して，F1 スコアが改善された．

次に，表 4.10 に提案手法によるクラスタ毎の結果を示す．表 4.10 から，クラスタ 2，5，7 では，他のクラスタの結果と比較して F1 スコアが低い結果となった．図 4.1 を参照するとクラスタ 2，5，7 は渋滞データの割合が小さく，バスの平均速度が小さいクラスタである．このようなクラスタでは，バスの平均速度が渋滞時と非渋滞時での違いが小さく，正しく判別できない可能性がある．

表 4.10 全バス停区間（16 系統）の学習器を元に各バス停区間の学習器に転移学習した場合のクラスタ毎の渋滞検知結果のマイクロ平均

クラスタ	Accuracy	Precision	Recall	F1
0	0.725	0.475	0.775	0.589
1	0.948	0.542	0.914	0.681
2	0.814	0.082	0.689	0.147
3	-	-	-	-
4	0.782	0.873	0.833	0.853
5	0.866	0.146	0.819	0.248
6	0.793	0.397	0.781	0.527
7	0.799	0.159	0.620	0.253

※クラスタ 3 はデータ C のみを用いたベースライン手法において全区間がオーバーサンプリング不可能のため実験対象外とした．

4.4.5 提案手法２－「区間を走行したバス本数」を追加したクラスタリング

次に，提案手法では，渋滞検知のための特徴量としてバスb_iの一つ前に同区間を走行したバスb_{i-1}の速度を用いていることから，バスの運行頻度が渋滞検知に影響を与えると考えられる．そこで，4.4.4 で用いた「バスの平均速度」「渋滞データの割合」に加え，「区間を走行したバスの本数のデータ」を追加し，クラスタリングを行った．それぞれの値の最小値が 0，最大値が 1 となるように正規化する．クラスタリングには，sklearn.cluster.KMeans（kmeans 法）を用いる．クラスタ数は 5，6，7，8，9，10 で実験し，最もスコアの良い 6 を実験結果として示す．

図 4.2，4.3 は，kmeans 法によってデータ追加後のクラスタリングの結果である．図 4.2 では Y 軸が渋滞データの割合を正規化したもの，X 軸がバス停区間におけるバスの平均速度を正規化したものを示し，図 4.3 では Y 軸がバス停区間におけるバスの通過本数を正規化したもの，X 軸がバスの平均速度を正規化したものを示す．

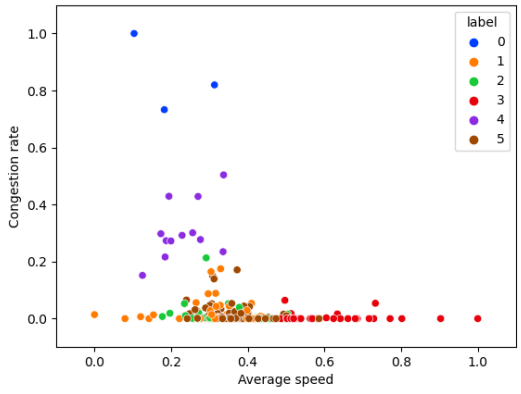

図 4.2 バス停区間のクラスタリング結果（バス停区間平均速度，渋滞割合，バスの通過本数によるクラスタリング）の渋滞割合と平均速度を示したグラフ

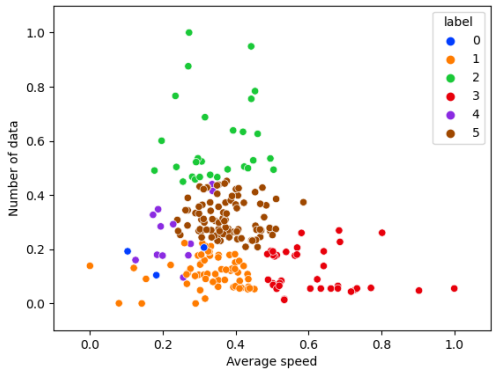

図 4.3 バス停区間のクラスタリング結果（バス停区間平均速度，渋滞割合，バス通過本数によるクラスタリング）のバス通過本数と平均速度を示したグラフ

なお，4.3.4 と同様に，「A,B の全バス停区間（16 系統）のデータで学習した学習器」に対して「データ C を用いてバス停区間毎に転移学習を行った学習器（バス停区間毎）」を用いる．表 4.11 に結果を示す．表 4.9 の結果と比較して，F1 スコアが 0.05 小さい結果となった．一方，recall は 0.01 上昇している．

また，表 4.12 にクラスタ毎の結果を示す．

表 4.11 全バス停区間（16 系統）の学習器を元に各バス停区間の学習器に転移学習した場合の渋滞検知結果のマイクロ平均

Accuracy	Precision	Recall	F1 スコア
0.805	0.292	0.789	0.426

表 4.12 全バス停区間（16 系統）の学習器を元に各バス停区間の学習器に転移学習した場合のクラスタ毎の渋滞検知結果のマイクロ平均

クラスタ	Accuracy	Precision	Recall	F1
0	0.823	0.895	0.867	0.881
1	0.727	0.102	0.707	0.179
2	0.768	0.111	0.819	0.195
3	0.863	0.571	0.941	0.711
4	0.739	0.480	0.752	0.586
5	0.870	0.207	0.786	0.328

表 4.12 から，クラスタ 1，2 では他の区間と比較して F1 スコアが特に低い結果となった．クラスタ 0 や 1 の共通点として，図 4.2 から，バスの平均速度が小さく，かつ渋滞割合が小さい区間のクラスタである．また，図 4.3 からクラスタ 1 はバス通過本数が少なく，クラスタ 2 はバス通過本数が多いことが分かる．これより，バスの通過本数の大小に関わらず，バスの平均速度が小さく，かつ渋滞割合が小さい区間では，バスの平均速度が渋滞時と非渋滞時での違いが小さく，正しく分類できない可能性が高い．

5. まとめ

本稿では，都営バスのリアルタイム運行データから渋滞検知を行う手法を提案した．渋滞検知精度向上のため，類似するバス停区間をクラスタリングした上で膨大なデータにより学習器を構築し，これを渋滞予測対象となるバス停区間毎の学習器に転移学習させた．クラスタ毎に転移学習を行うことで，F1 スコア 0.431 を得た．

今後の課題としては，バス停区間の平均速度が遅いにも関わらず，渋滞割合の小さい区間（乗降客数が追い，信号が多い区間）の検知精度改善が挙げられる．

謝　辞

本研究は，令和 4 年度「東京都と大学との共同事業」の一環として実施した．

参　考　文　献

[1] 青柳宏紀，岡田一洸，山名早人．都バスのリアルタイム運行データを用いた渋滞検知．DEIM2022 第 14 回データ工学と情報マネジメントに関するフォーラム．2022, pp.1-8.

[2] C．Samal, F. Sun and A. Dubey, "SpeedPro: A Predictive Multi-Model Approach for Urban Traffic Speed Estimation," 2017 IEEE Int. Conf. on Smart Computing (SMARTCOMP), 2017, pp.1-6, doi:10.1109/SMARTCOMP.2017. 7947048.

[3] T. Kyaw, N. N. Oo and W. Zaw, "Estimating Travel Speed of Yangon Road Network Using GPS Data and Machine Learning Techniques," 15th Int. Conf. on Electrical Engineering/Electronics, Computer, Telecommunications, and Information Technology (ECTI-CON), 2018, pp. 102-105, doi:10.1109/ECTICon.2018.8619908.

[4] Y. Gu, Y. Wang, and S. Dong, "Public Traffic Congestion Estimation Using an Artificial Neural Network," ISPRS Int. J. of Geo-Information, vol. 9, no. 3, article 152, 2020, pp.1-17, doi:10.3390/ijgi9030152.

[5] Y. Xu, Y. Wu, J. Xu, and L. Sun, "Efficient Detection Scheme for Urban Traffic Congestion Using Buses," 26th Int. Conf. on Advanced Information Networking and Applications Workshops, 2012, pp. 287-293, doi:10.1109/WAINA.2012.62.

[6] C. Wang and H. Tsai, "Detecting urban traffic congestion with single vehicle," Int. Conf. on Connected Vehicles and Expo (ICCVE), 2013, pp. 233-240, doi:10.1109/ICCVE.2013.6799799.

[7] R. Carli, M. Dotoli, N. Epicoco, B. Angelico and A. Vinciullo, "Automated evaluation of urban traffic congestion using bus as a probe," IEEE Int. Conf. on Automation Science and Engineering (CASE), 2015, pp.967-972, doi:10.1109/CoASE.2015.7294224.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," J. of Artificial Intelligence research, vol.16, pp.321-357, 2002, doi:10.1613/jair.953.