機械学習によるトランザクション処理性能予測の網羅的な評価

池田　悠人† 　三宅　康太† 　肖　　川† 　鬼塚　　真†

† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1–5
E-mail: †{ikeda.yuto,miyake.kota,chuanx,onizuka}@ist.osaka-u.ac.jp

あらまし　近年における急速な機械学習技術の発達に伴い，システム分野においても機械学習を応用して既存のコンポーネントを置き換える試みが見られる．機械学習を利用したカーディナリティ推定や索引などの研究によれば，機械学習手法が伝統的なシステムを上回る性能が報告されている．しかしトランザクション処理性能予測においてはこれまで，分析的な手法を含めて研究が行われていなかった．これを受けて本研究では，人工的なワークロードを生成し，そのパラメータから機械学習を用いてワークロードの特徴とその性能結果との依存関係を学習し，モデルの精度評価を行った．

キーワード　データベース性能予測, トランザクション処理, 回帰分析

1　はじめに

データベースにおける性能予測技術は，システム分野における重要な分野の一つである．データベースの性能の予測を適切に行うことはサービスレベル契約の作成において重要である．この分野の技術にクエリ性能予測がある．特定のアプリケーションに対して，クエリ性能を正確に予測すると， 単位時間あたりに実行できるクエリ数を保証するサービスレベル契約を作成することができる．

本研究ではトランザクション処理における性能予測の精度の評価及び向上を目的とする．トランザクションとはデータベースにおける概念で，データベース上の処理のまとまりを指す [1]．トランザクションは以下の 4 つからなる ACID 特性を備える．

- 原子性

トランザクションはデータベース処理の単位であり，その実行内容は完全に反映するかもしくは全く反映しないかのいずれかでなければならない．トランザクションはコミットするか，アボートするかのいずれかの状態のみで終了しなければならない．

- 整合性

整合性が保たれたデータベースに対してトランザクションを実行した後に，操作前と同様に整合性が保たれていなければならない．

- 隔離性

各トランザクション処理が相互に干渉してはならない．トランザクション処理結果は，ある順番にトランザクションを逐次的に処理した場合に得られる結果と一致しなければならない．

- 耐久性

一度データベースに反映されたトランザクション処理結果が，その後の障害などにより消失してはならない．

トランザクションはデータベース分野における最小限のまとまりであることから，クエリよりも実用的な単位であると言える．よって，サービスレベル契約の観点において，クエリ性能予測と比較してトランザクション処理性能予測はさらに有用性が認められる．

また、トランザクション処理性能予測における利点として，ワークロードに対して適切な同時実行制御を選択可能になるということが挙げられる．複数のスレッドを用いてトランザクションを処理する場合，各スレッドが相異なるトランザクション処理を並行実行できる．これはトランザクション処理性能向上にとって重要である．しかし扱うデータや操作の種類により，複数のトランザクション処理が干渉することがある．複数のトランザクション処理の結果は，それぞれを逐次処理した結果と同じ結果が得られることが望ましい．しかし処理が干渉した場合，期待した結果が得られない．同時実行制御は，トランザクション処理同士を調停し，干渉を防ぎつつトランザクション処理性能の向上を狙う目的がある．同時実行制御は，2 相ロッキングプロトコル [2] や楽観的同時実行制御 [3] などを始めとして，様々な手法が提案されてきた．これらの手法にはワークロードによって性能の差が存在する．トランザクション処理性能をより正確に予測することにより，ワークロードに対して最適な同時実行制御を選択することができるようになり，トランザクション処理性能の向上を見込むことができる．

しかし，トランザクション処理性能予測には技術的に困難な点が挙げられる．それは上に述べたような，スレッドを複数用いた場合に発生するトランザクション処理同士の干渉を防ぐために，トランザクション処理の並行実行に制約が生じ，処理性能の低下が発生することである．この並行実行の制約を捉えられない場合、トランザクション処理性能予測はより困難になる．また並行実行に対する制約による性能低下がスレッド数に対して非線形に現れることも予測の難しさに困難にする要因である．これらの原因から，トランザクション処理予測は分析的なモデルの構築が困難であり，これまで先行研究が行われていなかった．

一方，近年の機械学習技術の発達に伴い，データベースシステム分野においても機械学習を導入することにより既存のコン

ポーネントを置き換える動きが見られる．その例の一部として，索引の分野においては learned index [4] が挙げられ，カーディナリティ推定の分野においては Naru [5] や DeepDB [6] が挙げられる．これらの研究において，実験の条件によっては索引やカーデナリティ推定における伝統的なシステムを上回る性能を上回ることが報告されている [7]．データベースシステムへの機械学習の応用は未だに発展途上の段階であり，トランザクション処理性能予測については分析的モデルによるアプローチ同様に，これまで先行研究が行われていない．そこで本研究では索引やカーデナリティ推定の分野で成果を挙げる機械学習手法を，トランザクション処理予測に応用する．学習により得られたモデルの精度を評価し，より良い予測を行うことを本研究の目的とする．

本稿の構成は以下の通りである．2 章で本研究で扱う問題の定義について説明を行い，3 章で提案手法の説明や，実験で用いる回帰アルゴリズムの説明を行い，4 章で詳しい実験の内容やその実験結果を示し，考察を行う．また，5 章では関連研究を紹介する．最後に，6 章で結論を述べる．

2 問 題 定 義

1 章において理由を述べた通り，トランザクション処理性能の分析的予測は困難であり，これまで研究が進められていなかった．またこれまで索引やカーディナリティ推定に代表されるような，機械学習を用いた手法においてもトランザクション処理性能の予測に関しては先行研究が存在せず，この分野は分析的，または機械学習によるアプローチ共にこれまで全く研究が行われてこなかった領域である．そこで本研究では，様々なワークロードにおけるデータベースのトランザクション処理性能を計測し、その計測結果から性能予測モデルを生成する．生成したモデルをトランザクション処理予測性能に利用し，生成した予測モデルの精度を評価することが本研究の目的である．また，複数のワークロードに対して実験を行うことで，ワークロードにおける性能予測精度の違いや予測精度の特性を明らかにする．

本研究において，予測を行う性能の指標には単位時間あたりにコミットに成功したトランザクション数を示すスループットを用いる．これはスループットがデータベースにおける性能指標として普遍的に用いられており，また 1 章で説明したようなサービスレベル契約に応用可能な点も理由として挙げられる．予測ターゲットに用いるスループットは連続値をとる．よってトランザクション処理性能予測は、機械学習における，連続値を予測するタスクである回帰に帰着可能である．そこで回帰手法を用いて予測モデルを構築し精度評価を行う．本研究におけるフレームワークを図 1 に示す．

データセットを D としたとき、D は次のように表現される．

$$D = (\mathbf{x}_0, y_0), (\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n). \tag{1}$$

ただし，$\mathbf{x}$ はワークロードの特徴を表すベクトルであり，y は予測ターゲットであるスループットである．また n はデータセットの大きさを表す．本研究において，予測モデルを構築することは，次の式を満たす $f(\cdot)$ を求めることである．

$$y_k = f(\mathbf{x}_k). \tag{2}$$

3 提 案 手 法

3.1 特徴ベクトル化手法

性能予測に用いる特徴ベクトルには，ワークロードにおいて発行されるトランザクションの種類の割合と，トランザクション処理に用いるスレッド数を用いる．学習速度や予測精度の向上を図るため，スレッド数を min-max 法を用いて正規化した．min-max 法は対象のデータを $[0, 1]$ の閉区間に正規化する手法で，次のように計算される．

$$\hat{x} = \frac{x - min(X)}{max(X) - min(X)}. \tag{3}$$

ここで，x は変換前の値，X は対象のデータの集合を示し，$\hat{x}$ は変換後の値を示す．本研究においては $max(X)$ はスレッド数の最大値であり $min(X)$ はスレッドの最小値である．またトランザクションの発行割合は，データに応じ前述の min-max 法と標準得点を用いての標準化を使い分ける．標準得点は対象のデータを平均 0，標準偏差 1 の分布に標準化する手法であり，以下のように計算される．

$$\hat{x} = \frac{x - E(X)}{\sigma}. \tag{4}$$

ここで $x, X, \hat{x}$ は式 3 と同様であり，$E(\cdot)$ は式 8 と同様に平均をとる演算である．また σ は対象のデータの標準偏差を示す．特徴ベクトル x は，ワークロードにおけるトランザクションの種類の総数を a として，次のように表現される．

$$\mathbf{x} = (z(p_0), z(p_1), ..., z(p_a), m(t)). \tag{5}$$

p はワークロードにおいて発行されるそれぞれのトランザクションの割合を表し，t は並行実行に用いるスレッド数を表す．また，$z(\cdot)$ は標準得点による標準化を表し，$m(\cdot)$ は min-max による正規化を表す．

3.2 評 価 手 法

精度の評価を行う指標には平均絶対パーセント誤差 (MAPE) [8] と 二乗平均誤差の平方根 (RMSE) を用いる．MAPE の定義は次の式の通りである．

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - p_i}{y_i}\right|. \tag{6}$$

また RMSE の定義は次の式の通りである．

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2}. \tag{7}$$

ここで n はテストデータに含まれるデータ数，y はテストデータにおける実際のターゲット値， p はテストデータの特徴ベク

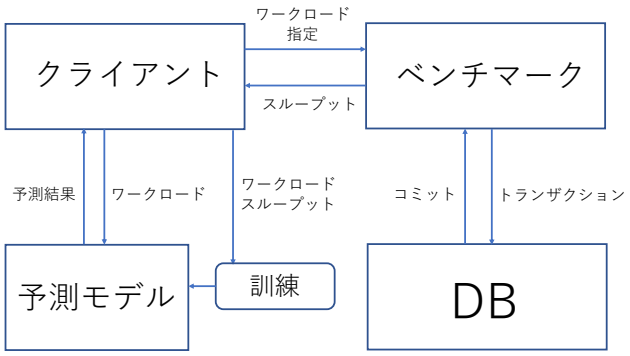

図 1: フレームワーク

トルからモデルが予測した値である．RMSE や MSE と比較すると MAPE は比較するデータ間の桁数が違うときにも用いることが可能である特徴を持つ．一方で MAPE は実測値が 0 をとるデータが存在するときに用いることができないという欠点を持つ．本研究においてはスループットが 0 ではないデータセットを用意しているため，この点は問題にはならない．他方 RMSE は誤差を二乗することからより外れ値に敏感である特徴を持つ．また MSE と比べて，単位が元のデータと一致するため，より直感的に理解しやすい特徴も存在する．

3.3 回帰手法

機械学習は教師あり学習，教師なし学習，強化学習の大きく 3 つに分類可能である．その中で教師あり学習とは、説明変数と正解を示すラベルが対になったデータセットから、2 つの相関を表すモデルを生成するものを指す．教師あり学習は予測するターゲットにより分類と回帰の 2 種類に分けられる．分類はターゲットが離散値のラベルであるタスクであり，分類先のラベルが 2 種類の場合は二値分類，ラベルが 3 種類以上ある場合は多値分類という．一方分類回帰とは，説明変数から連続的なターゲットへの当てはめを行うタスクである．回帰はモデルの種類により，線形回帰と非線形回帰の 2 種類に分類される．

以下の節では，各回帰手法について詳細に説明を行う．

3.3.1 ランダムフォレスト

ランダムフォレストは，決定木を用いたバギングに基づくアンサンブル学習の手法である．バギングとは全ての訓練データセットの中からランダムにサブセットを抽出することを繰り返して弱モデルを生成し，その予測結果を組み合わせて最終的な予測に用いる手法である．それぞれの弱モデルは独立に生成されるため，モデル間に相関はない．よってモデル生成を並列処理することが可能であることため，学習の高速化を期待できる点が長所として挙げられる．しかし，バギングは訓練データセットが増加するに従い生成した弱モデルが似た性能となりやすいといった課題が存在する．ここで，モデルが似るとは同じ特徴ベクトルを入力した際の予測結果の差異が小さいことである．アンサンブル学習はそれぞれの弱モデルが異なる性能であるという前提のもと，一つの強力なモデルを生成するアルゴリズムである．表 1 に分類タスクにおける 3 つの弱モデルを用いて行うアンサンブル学習の例を示す．表において〇で示された箇所は弱モデルがそのタスクに正解の分類結果を返していることを示し，×で示された箇所は弱モデルが間違った分類結果を返していることを示す．弱モデルの正解率は全て 60% であるが，それぞれの予測結果の多数決を最終的な予測結果として採用することで，80% もの正解率を達成している．このように，アンサンブル学習では精度の低い弱モデルを複数用いることで高精度の予測を行う．しかし表 2 のように，弱モデル同士に違いがない場合、複数のモデルを組み合わせることによる利点が得られない．このように弱モデルが似ることは，全体的なモデルの予測能力の低下を招く [9].

例では分類の場合を提示したが，回帰においても同様である．分散公式 [10] から次の式が成り立つ

$$\sigma^2 = E(z^2) - (E(z))^2. \tag{8}$$

ここで $E(\cdot)$ は平均をとる演算であり，z は任意の確率変数を示す。また σ^2 は z の分散を表す．定義から分散は非負の値であるため

$$E(z^2) \geqq (E(z))^2. \tag{9}$$

が成立する．z を $f(\cdot)$ と一般化すると，

$$E((f(x))^2) \geqq (E(f(x)))^2. \tag{10}$$

である．$f(\cdot)$ が予測結果と実測値の残差であると定義したとき，式 10 は複数モデルの二乗平均誤差が，単一モデルの誤差の以下あることを示している．式 10 が等号成立する条件は式 8 において $\sigma = 0$ である．これは弱モデルの予測値が全く同じ場合である．これらのことから，回帰においても、弱モデルの予測結果の平均値を全体のモデルの予測値とした場合，分類と同様に弱モデルが異なる性能であることがアンサンブル学習にとって重要であることが言える．

ランダムフォレストでは，前述したバギングにおける決定木の性能が似ることによる予測性能の低下を解決するために，予測に用いる特徴量をランダムに選択することで，決定木の生成にランダム性を取り入れている．一つの弱モデルに用いる特徴ベクトルの次元数はハイパーパラメータである．弱モデルにそ

表 1: 分類におけるアンサンブル学習の例

タスク	1	2	3	4	5	6	7	8	9	10
モデル A	〇	〇	〇	〇	〇	〇	×	×	×	×
モデル B	×	×	〇	〇	〇	〇	〇	〇	×	×
モデル C	〇	〇	×	×	×	〇	〇	〇	〇	×
多数決	〇	〇	〇	〇	〇	〇	〇	〇	×	×

表 2: 弱モデルが全て同じ答えを返すアンサンブル学習例

タスク	1	2	3	4	5	6	7	8	9	10
モデル A	〇	〇	〇	〇	〇	〇	×	×	×	×
モデル B	〇	〇	〇	〇	〇	〇	×	×	×	×
モデル C	〇	〇	〇	〇	〇	〇	×	×	×	×
多数決	〇	〇	〇	〇	〇	〇	×	×	×	×

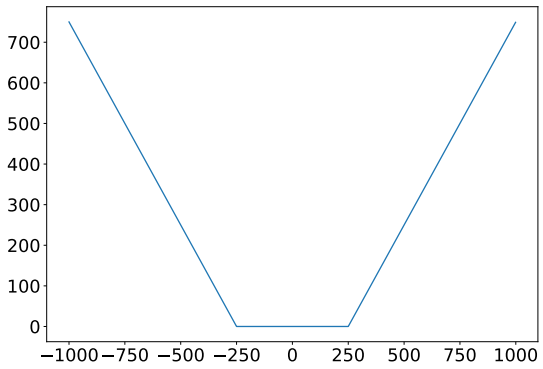

図 2: SVR の損失関数の例 ($\epsilon = 250$)

れぞれ異なる次元の特徴ベクトルを用いることで，弱モデルが似た性能になることを防ぎ，予測性能の向上を図っている．

3.3.2 サポートベクタ回帰

サポートベクタ回帰 (SVR) は，単クラス分類手法であるサポートベクタマシンを回帰分析のアルゴリズムとして応用した手法である．SVR における回帰モデル関数は次に示す式によって表現される．

$$f(\mathbf{x}) = \phi(\mathbf{x})\mathbf{w} + c. \tag{11}$$

ここで，$\mathbf{x}$ は入力する特徴ベクトルを示し，$\phi(\cdot)$ は特徴ベクトルを高次元空間へと写像するカーネル関数を示す．また c は定数項である．カーネル関数が

$$\phi(\mathbf{x}) = \mathbf{x}. \tag{12}$$

であるとき，回帰モデル関数は重線形回帰と一致する．カーネル関数には様々なカーネルが適用可能であるが，その中でもガウシアンカーネルが適用されることが多い．カーネル関数を非線形な関数とすることで SVR は非線形なモデルを表現可能である．また SVR において損失関数は次のように定義する．

$$L = \frac{1}{2}||\mathbf{w}|| + Ch(y - f(\mathbf{x})). \tag{13}$$

第一項は係数の大きさを表す．また第二項は誤差関数を表す．なお C は 2 つの項の重みを決定するハイパーパラメータである． h で表現されている損失関数は次のように定義する．

$$h = max(0, |y - f(\mathbf{x})| - \epsilon). \tag{14}$$

損失関数のグラフは図 2 の通りである．予測と実際の値の誤差が ϵ 以下である時，誤差関数は 0 を返す．このような不感帯を設けることにより， SVR は訓練データに含まれるノイズに対して頑強である．

3.3.3 勾配ブースト法

勾配ブースト法は，ブースティングを用いたアンサンブル学習による手法である．ブースティングとは複数のモデルを用いるアンサンブル手法のうち，既存の弱モデルの予測誤差をフィードバックして，弱モデル生成を繰り返すことにより精度の良いモデルの生成を行う手法である．勾配ブースト法ではこのブースティングにより確率的勾配降下法を行う． n 番目のモデル出力を f_n，n 番目の全体の予測値を $\hat{y_n}$ と定義したとき，勾配ブースト法における $n+1$ 番目の出力は以下の通りである．

$$\begin{aligned} \hat{y_{n+1}} &= \hat{y_n} + f_{n+1} \\ &= \sum_{k=1}^{n+1} f_k. \end{aligned} \tag{15}$$

これは出力が全ての弱モデルの予測値の総和であることを示している．各弱モデル f_n は $\hat{y_{n-1}} + f_n$ の損失関数が最小になるように設定され，全体の出力は実測値に漸近する．

勾配ブースト法には過学習を起こす可能性が高いという課題がある，その課題を解消するためいくつかの工夫が存在する．その一つにハイパーパラメータとして学習率 η を導入し，学習の速度を制限する手法が挙げられる，この場合の $n+1$ 番目の出力は以下の通りである．

$$\hat{y_{n+1}} = \hat{y_n} + \eta f_{n+1}. \tag{16}$$

またバギング手法と同様にデータをサンプリングする方法も過学習を防ぐ手法として挙げられる．これはデータ数を削減する手法や，ランダムフォレストと同様にデータの次元を削減する手法などが挙げられ，サンプリングを用いない従来の手法と比較して精度の向上が報告されている [11]．さらに，SVR と同様に罰則項を追加することで，生成されるモデルの複雑さに制限を加え過学習を防ぐ手法も提案されている [12]．罰則項を含めた，勾配ブースト法における損失関数は次のように定義する．

$$L(\phi) = \sum_i l(y_i, \hat{y_i}) + \sum_{k=1}^{K} \Omega(f_k). \tag{17}$$

ここで y は実測値である．f はサンプルデータの予測値と実測値との差の大きさを示す関数である．この関数には残差二乗関数のような微分可能な凸関数を用いる必要がある．また第二項は罰則項を示す。Ω は次のように定義する．

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda||w||^2. \tag{18}$$

T は決定木の葉ノードの数，w は決定木の重みの L2 ノルムである．また γ や λ は 2 つの項の重みを決定するハイパーパラメータである．

4 実　　験

4.1 実験内容

本研究では，性能評価を行うベンチマークとして TPC-C [13] と CH-benCHmark [14] のうちの TPC-H を元にしたトランザクションを用いて実験を行った．TPC-C は OLTP のベンチマークであり，9 つのテーブルと 5 種類のトランザクションから構成されている．これにより，倉庫を中心とした注文や配送といった業務を再現している．一方で，CH-benCHmark は OLAP と OLTP を混合したベンチマークとして設計されてい

る．テーブルは前述の TPC-C を拡張したものである．トランザクションは TPC-C と同一なものと TPC-H のトランザクションを CH-benCHmark 向けに改変したものを含む．本実験においては TPC-H 由来のトランザクションのみを用いるため，以下単に TPC-H と表記する． 各ベンチマークにおけるトランザクションの発行割合をランダムに変更し，各発行割合においてスループットを計測した．TPC-C においてはトランザクションが 5 種類であるため，その割合を表現する特徴ベクトルは 5 次元であり，同様に TPC-H においてはトランザクションが 22 種類であるため，割合を示す特徴ベクトルは 22 次元である．

実際のスループット計測には OLTPBench [15] を用いた．SERIALIZABLE 隔離レベルで動作する MySQL に対して，スレッド数やトランザクションの発行割合を指定して TPC-C ベンチマークを実行した．TPC-C での実験においてはデータベースのウォームアップの観点から，20 秒間ベンチマークを実行し，10 秒から 15 秒間のうちにコミットしたトランザクション数を計測してスループットを計測した．また TPC-H での実験においてはウォームアップを行わずに 50 秒間ベンチマークを実行しスループットを計測した．本実験を通して，1 秒あたりのトランザクションの発行数は 10000 とし，warehouse は実験を通して全て 1 で固定している．トランザクションの同時実行に用いられるスレッド数は 1 から 72 までであり，スレッド数は一様分布による乱数から決定される．スループットを計測する実験環境として，36 個の物理コアを持つ Intel(R) Xeon(R) CPU E7-8880 v3 を 2 つと，64GB のメモリを持つ linux 環境を利用した．

回帰に用いる手法は，3. 3 章で説明を行ったランダムフォレストと SVR，勾配ブースト法と線形回帰を用いた．線形回帰とランダムフォレスト，SVR の実験は Python の scikit-learn [16]，勾配ブースト法は XGBoost [12] の実装を利用した．実験に用いたデータ数は 1500 であり，訓練データとテストデータの割合は 9 : 1 である．またはグリッドサーチと交差検証を利用し，ハイパーパラメータの最適化を行った．

4. 2 実 験 結 果

図 3 から図 10 に，各アルゴリズムやワークロードにおける予測値と実際のスループットの分布を散布図に示す．それぞれの散布図において，横軸は予測値を示し，縦軸は実測値を示す．また各実験における予測値と実測値の誤差を示す RMSE と MAPE の値を表 3 と表 4 に示す．

MAPE の値を見ると，TPC-H と比較して TPC-C では精度よく予測を行っていることが読み取れる．これは TPC-C が TPC-H よりもスレッド数に対してスループットが線形にスケールする傾向が強いことがその要因であると考えられる．また，アルゴリズム別に比較すると，TPC-C,TPC-H ともに，線形回帰よりも各回帰手法は良い予測結果を記録し，特に勾配ブー

表 3: RMSE

ワークロード	TPC-C	TPC-H
線形回帰	312.6	0.243
ランダムフォレスト	44.1	0.158
SVR	141.1	0.178
XGBoost	43.7	0.163

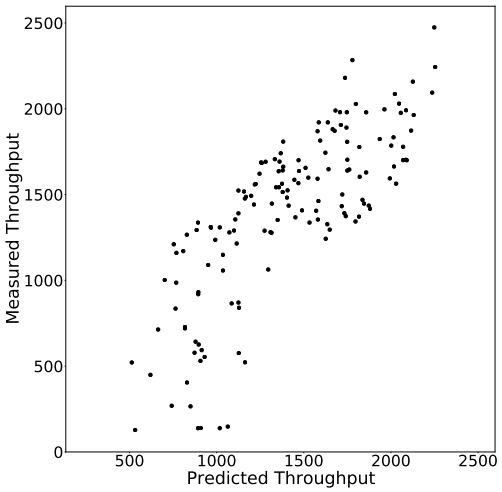

図 3: TPC-C: 線形回帰

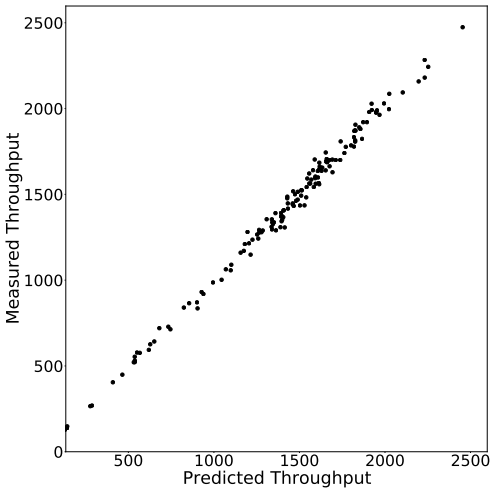

図 4: TPC-C: ランダムフォレスト

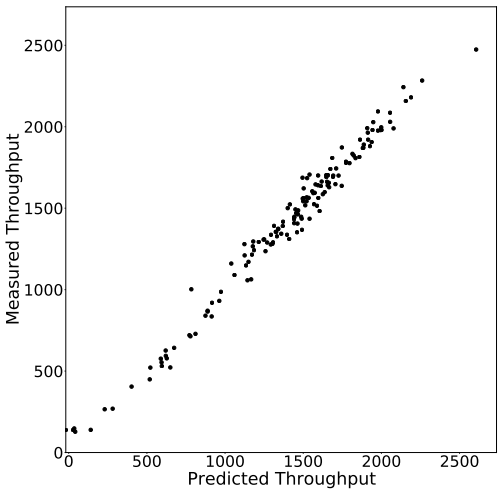

図 5: TPC-C:SVR

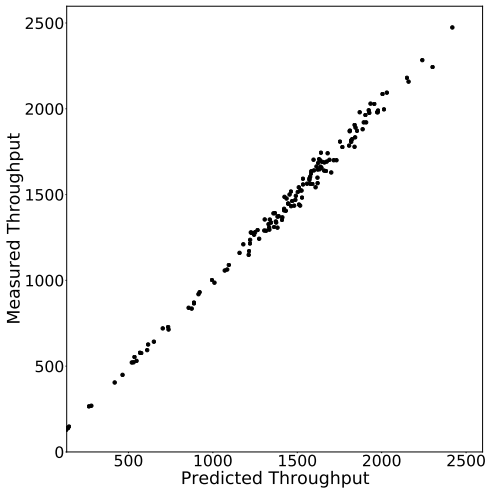

図 6: TPC-C: 勾配ブースト法

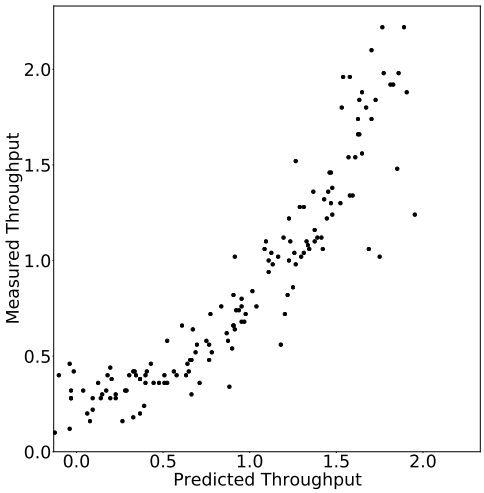

図 7: TPC-H: 線形回帰

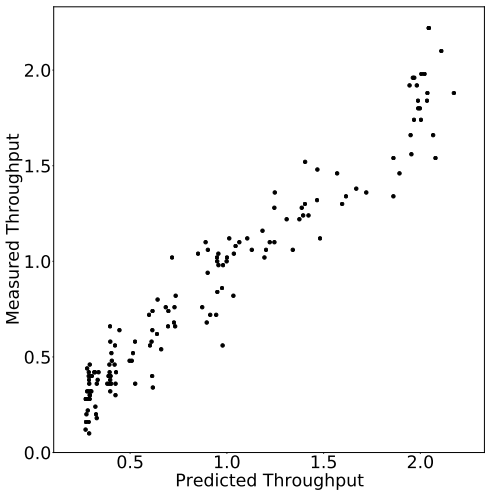

図 8: TPC-H: ランダムフォレスト

スト法やランダムフォレストといった決定木に基づく手法が，より精度の良い予測を行っていることがわかる．これは，勾配ブースト木やランダムフォレストはアンサンブル手法を用いることからより複雑なモデルを表現可能である一方で，トランザクション同士の競合を定式化することが難しいことから，SVR のような単独のモデルによる表現ではワークロードの特徴を捉えることが難しいことが要因の一つとして考えられる．また，SVR はランダムフォレストのように，予測に用いる特徴ベクトルの次元を重みづけすることができないことも，予測精度に劣る原因として考えられる．

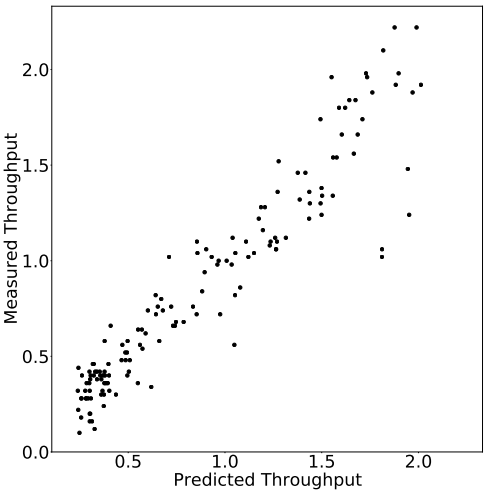

図 9: TPC-H:SVR

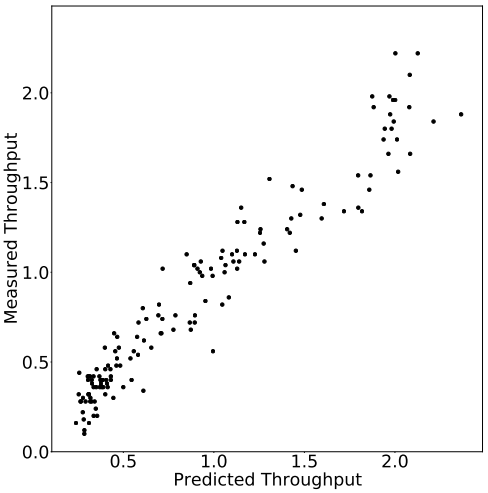

図 10: TPC-H: 勾配ブースト法

5 関 連 研 究

1 章で述べたように，索引やカーディナリティ推定の分野において機械学習による手法が提案されている．また同時実行制御においても強化学習により，ワークロードに対してポリシー

表 4: MAPE

ワークロード	TPC-C	TPC-H
線形回帰	0.206	0.866
ランダムフォレスト	0.024	0.301
SVR	0.086	0.167
XGBoost	0.022	0.154

を最適化する手法 [17] や，アボートしたトランザクションの情報から，スレッドに対して最適なトランザクションの割り当てを学習する手法 [18] によって，従来手法よりも高性能化を行うことが提案されている．これらの手法では静的環境において伝統的な手法を上回る結果が報告されている一方で，動的な環境では伝統的な手法を下回る結果が報告されているか，もしくは動的環境での実験が不足している．よって動的環境での実験による性能調査や，性能改善が機械学習手法における今後の課題として挙げられる．

6 終わりに

本研究では機械学習を用いてのトランザクション処理性能予測を提案した．また，そのモデルの精度を TPC-C ベンチマークを利用して評価を行った実験結果においては機械学習による予測が，よくワークロードの特徴を捉えられることが明らかになった．

本研究に関しての今後の課題として，以下のものが挙げられる．

- ワークロード

本研究ではベンチマークを TPC-C と TPC-H に定めて実験を行った．データベースの性能を測定するベンチマークとしては他に TPC-E [19]，YCSB [20] などが挙げられる．これらのベンチマークを含めたより大規模な実験を行い，その実験結果による比較検討を行うことは，モデルの精度の特徴を捉えるという点において有効である．

- 特徴ベクトル化の手法

本研究においては，TPC-C ワークロードなど，事前に発行されるトランザクションの種類が分かっているという前提のもとで，そのトランザクションの種類の割合を特徴ベクトル化して性能予測を行った．しかし，実際には未知のトランザクションが発行される可能性があるワークロードも存在する．本研究において提案した手法では, このような発行されるトランザクションが事前に定まっていないワークロードにおいて予測を行えないという課題が存在する．そこで，どのような形のトランザクションにも対応できるベクトル化手法を検討することは，予測手法の汎用性を高める点において重要である．また，提案手法の特徴ベクトル化手法はナイーブなものであり，ワークロードによっては予測精度が下がるといった結果が明らかになった．しかし特徴ベクトル手法を工夫することにより，予測精度の向上を図ることも考えられる．例えば，トランザクションごとの扱うデータに着目してベクトル化を行えば，よりトランザクション間の依存関係を捉えやすくなり，精度向上に寄与する可能性がある．このように，汎用性や予測精度という点において提案手法よりも有効な特徴ベクトル化手法を検討することも本研究の今後の課題として考えられる．

- 異なる実験環境

本研究においてはデータベースとクライアントを同一のマシン上で行い，実験を実施した．実際の環境においては，データベースシステムとクライアントは異なるマシン上であることが多い．この場合，マシン間の通信によるコストは無視できないものである．このコストを含めて予測を行うことはより，より実用的な予測が可能になるため，意義があるものである．

謝　　辞

本研究は, 日本学術振興会科研費 19K11979 の助成を受けています．

本研究の遂行にあたり多大なるご指導，ご助言を賜りました大阪大学鬼塚研究室の佐々木先生，並びにシステム班の諸氏に感謝申し上げます．

文　　献

[1] 北川博之. データベースシステム. オーム社, 2019.
[2] Kapali P. Eswaran, Jim N Gray, Raymond A. Lorie, and Irving L. Traiger. The notions of consistency and predicate locks in a database system. *Communications of the ACM*, Vol. 19, No. 11, pp. 624–633, 1976.
[3] Theo Härder. Observations on optimistic concurrency control schemes. *Information Systems*, Vol. 9, No. 2, pp. 111–120, 1984.
[4] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *SIGMOD*, pp. 489–504, 2018.
[5] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. Deep unsupervised cardinality estimation. *PVLDB*, Vol. 13, No. 3, pp. 279–292, 2019.
[6] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. Deepdb: Learn from data, not from queries! *PVLDB*, Vol. 13, No. 7, pp. 992–1005, 2020.
[7] Xiaoying Wang, Changbo Qu, Weiyuan Wu, Jiannan Wang, and Qingqing Zhou. Are we ready for learned cardinality estimation? *PVLDB*, Vol. 14, No. 9, pp. 1640–1654, 2021.
[8] De Myttenaere Arnaud, Golden Boris, Le Grand Bénédicte, and Rossi Fabrice. Mean absolute percentage error for regression models. *Neurocomputing*, Vol. 192, pp. 38–48, 2016.
[9] 坂本俊之. 作ってわかる!アンサンブル学習アルゴリズム入門. C&R 研究所, 2019.
[10] 稲垣宣生, 山根芳知, 吉田光雄. 統計学入門, p. 54. 裳華房, 2019.
[11] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, Vol. 38, No. 4, pp. 367–378, 2002.
[12] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pp. 785–794, 2016.
[13] The Transaction Processing Council. TPC-C benchmark (revision 5.9.0). 2007.
[14] Richard Cole, Florian Funke, Leo Giakoumakis, Wey Guy, Alfons Kemper, Stefan Krompass, Harumi Kuno, Raghunath Nambiar, Thomas Neumann, Meikel Poess, et al. The mixed workload CH-benCHmark. In *DBTest*, pp. 1–6, 2011.
[15] Djellel Eddine Difallah, Andrew Pavlo, Carlo Curino, and Philippe Cudré-Mauroux. OLTP-Bench: An extensible testbed for benchmarking relational databases. *PVLDB*, Vol. 7, No. 4, pp. 277–288, 2013.
[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830, 2011.

[17] Jiachen Wang, Ding Ding, Huan Wang, Conrad Christensen, Zhaoguo Wang, Haibo Chen, and Jinyang Li. Polyjuice: High-performance transactions via learned concurrency control. In *OSDI*, pp. 198–216, 2021.

[18] Yangjun Sheng, Anthony Tomasic, Tieying Zhang, and Andrew Pavlo. Scheduling oltp transactions via learned abort prediction. In *aiDM*, pp. 1–8, 2019.

[19] TPC-E benchmark. `http://www.globus.org/toolkit/`.

[20] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *SoCC*, pp. 143–154, 2010.