複雑なスキーマを持つデータを管理するための
カーディナリティ推定に関する研究

提出先　　大阪大学大学院情報科学研究科

提出年月　2023 年 1 月

伊 藤　竜 一

本研究に関連する研究業績

論文誌

1. 伊藤竜一, 佐々木勇和, 肖川, 鬼塚真. 非自己回帰モデルによる高速で安定したカーディナリティ推定. 情報処理学会論文誌データベース, Vol. 15, No. 3, pp. 36–49, 2022.

国際会議

1. Ryuichi Ito, Chuan Xiao, and Makoto Onizuka. Robust Cardinality Estimator by Non-Autoregressive Model. *Proceedings of the Fifth Workshop on Software Foundations for Data Interoperability*, 2021. Reproduced with permission from Springer Nature.

プレプリント（査読なし）

1. Ryuichi Ito, Seng Pei Liew, Tsubasa Takahashi, Yuya Sasaki, and Makoto Onizuka. Scaling Private Deep Learning with Low-Rank and Sparse Gradients. *arXiv preprint arXiv:2207.02699*, 2022.

国内会議（査読なし）

1. 伊藤竜一, リュウセンペイ, 高橋翼, 佐々木勇和, 鬼塚真. 低ランク近似を介した選択的パラメータ更新による差分プライベート学習. 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論文集, 2022. [優秀論文賞]

2. 川本孝太朗, 伊藤竜一, 佐々木勇和, 肖川, 鬼塚真. 結合カーディナリティ推定の中間結果を利用した結合順最適化. 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論文集, 2022.
3. 伊藤竜一, 佐々木勇和, 肖川, 鬼塚真. Non-Autoregressive モデルによる高速で安定したカーディナリティ推定. 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論文集, 2021. [学生プレゼンテーション賞].
4. 伊藤竜一, 佐々木勇和, 肖川, 鬼塚真. 非自己回帰モデルによる安定したカーディナリティ推定. 情報処理学会第 83 回全国大会論文集, 2021.

内容梗概

計算機資源の発展やデジタルトランスフォーメーションの進展に伴い，多種多様なデータが蓄積されその利活用が進んでいる．これまで管理されなかった事象まで扱うことができるようになり，データの量だけでなく構造の複雑さも増している．それらデータを利活用するトレンドと同時に，一個人からすると知られたくないようなデータ，つまりパーソナルデータ [8] が含まれることも増えていることから，プライバシ保護への配慮や流出・濫用を防ぐための規則の遵守も求められている．これらの背景から，本研究では複雑なスキーマを持つ大規模データの効率的な処理とプライバシ保護に焦点を当てる．

まず複雑なスキーマを持つ大規模データの処理に対応するという観点では，高速化が必要となる．これには大きく分けて 2 つのアプローチが考えられ，いずれかまたは両方利用できることが望ましい．1 つは既存のデータ処理技術を高速化するアプローチである．ユーザから見ると変更を加えることなく高速になり，より大きなデータを扱えるようになるという点で有益である．もう 1 つは処理結果の厳密性を捨てる代わりに前者以上の高速化をするアプローチである．近似的な処理結果になるものの，厳密性が不要なユースケースであればより一層の高速化を期待できる．次にパーソナルデータの処理という観点では，処理結果から個人の特定につながる情報が得られないようにする必要がある．巧妙な攻撃手段が増え，攻撃者がどのように処理結果を悪用するのかや外部知識を利用する可能性などを事前に想定することは難しいため，特定の設定によらないプライバシ保護が望ましい．

本研究ではこれらの背景を踏まえ，複雑なスキーマを持つ大規模データ処理の高速化とプライバシ保護を実現するための 3 つの要素技術を提案する．まず複雑なスキーマを持つ大規模データの高速な処理に繋がるカーディナリティ推定と呼ばれる技術に注目し，**(1)** 高速で精度の高いカーディナリティ推定手法，**(2)** 複雑なスキーマを持つ大規模データに対応するカーディナリティ推定手法に取り組む．また，プライバシ保護として，厳密な指標である差分プライバシに注目し，**(3)** (1) や (2) に適用可能な差分プライベート学習手法に取り組む．

(1) では，データの分布を捉える深層学習技術に着目し，高速で精度の高いカーディナリティ推定手法を提案する．カーディナリティ推定とはデータベース内に指定された条件を満たすデータが何件存在するかを推定するというタスクである．カーディナリティ推定手法の改善により，既存のデータベースシステムはインターフェースを変えることなくより高速な処理手段を選択できるようになる．また，カーディナリティ推定は条件に一致する件数を近似的に求める処理と等価であり近似的な処理のプリミティブとなることから，様々な近似的な処理の改善にもつながる．従来のカーディナリティ推定手法は実世界データにそぐわない仮定を置いているため精度が低いことが知られている．これを解決するため深層学習を利用した手法も提案されているが，人手によるパラメータ設定に依存している部分が大きく性能が安定していない．提案手法では既存の深層学習を利用したカーディナリティ推定手法とは異なり，クエリに応じた推論を行うことで安定した推論を実現する．実験では提案手法が安定して性能が高く高速に動作することを確認した．

(2) では，データベースシステムでカーディナリティ推定を利用する際に特に処理性能への影響が大きいと知られている，条件に結合を含むカーディナリティの推定を改善する提案を行う．従来のカーディナリティ推定手法では，スキーマに存在する全てのテーブルに跨るデータの分布を，1 つの推定器で捉えるか，テーブル間の相関に基づいた分割ごとの推定器で捉えることで結合を扱っている．しかしながら，いずれのアプローチもスキーマの規模に対してスケールせず，推定精度の大幅な低下やそもそも推定が困難になるといった問題がある．これに対し提案手法ではスキーマに定義された外部キー制約に基づいた分割ごとの推定器を用いることで，大規模かつ複雑なスキーマを持つデータに対応するカーディナリティ推定を実現する．実験では既存手法が動作しないような大規模なスキーマを持つベンチマークで動作し，更に推定性能が高いことを確認した．また，データベースシステムへの応用を想定したときに性能向上への寄与があることも確認した．

(3) では，(1) (2) で用いるような深層学習手法に対して，手法の有用性低下が少なく差分プライバシを満たすことができる手法を提案する．差分プライバシとは，k-匿名化のように事前に攻撃パターンを想定するのではなく，識別困難性から，任意の背景知識を持つ攻撃者の任意の処理に対して安全性を担保する指標である．従来の手法では差分プライバシを満たすことにより安全性を担保する一方で，深層学習手法自体の有用性，例えば分類器であれば分類性能が大きく低下してしまうとい

う問題がある．提案手法ではニューラルネットワーク内の大域的な冗長性と局所的な冗長性の両方を活用することで，安全性を下げることなく有用性の高い学習を実現する．実験では，様々な深層学習モデル・タスクで差分プライバシを満たした有用性の高い学習となることを確認した．また，(1) (2) のようなカーディナリティ推定手法に対しても有効であることを確認した．

目次

第 1 章　序論

1.1　研究の背景と目的

様々な物事・事象が電子的なデータとして記録されるようになっている．例えば紙の台帳で記録されていた売上情報が電子的に保存されたり，一人一台持ち歩くことが当たり前になったスマートフォンでは 24 時間位置情報をトラッキングできる．これらのデータは量が増加しているだけでなく，売上情報であれば顧客情報や商品情報と，スマートフォンの位置情報であれば訪問した店舗や出会った人というように，関連付けられるデータも多様になっている．複数の関連付けられたデータは有益な情報を含むことがあり，その有効活用も進んでいる [9, 10, 11]．

しかしながら，データが大規模で関係性が複雑な場合やスマートフォンの位置情報のような個人にまつわる開示が望ましくないデータ，いわゆるパーソナルデータを含む場合などは様々な考慮が必要である [12]．大規模で関係性が複雑なデータは処理コストが高く，従来のデータベースシステムでは捌ききれないことや扱えたとしても実用に耐え難い処理速度となってしまうことがある．この際より多くの計算機資源を投入して解決することが考えられるが，当然コストがかかるため，データ処理自体を改善することが望まれる．また，パーソナルデータを含む場合は相応のプライバシ保護が必要となる．様々な箇所に個人にまつわる情報が表れるようになったこともあり，管理者はプライバシを厳格に守るように要求されている．法令として整備されている事例だけでも，日本における個人情報保護法[1)]，EU における GDPR[2)]，米国における HIPAA[3)]や GLBA[4)]など，国や対象の情報の種類によって様々である．

1） https://elaws.e-gov.go.jp/document?lawid=415AC0000000057
2） https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679
3） https://www.govinfo.gov/link/plaw/104/public/191?link-type=pdf&.pdf
4） https://www.govinfo.gov/content/pkg/PLAW-106publ102/pdf/PLAW-106publ102.pdf

これらの背景を踏まえ，本研究では大規模かつ複雑なスキーマを持つデータの効率的な処理とプライバシ保護に取り組む．なお，本研究では特に SQL を通して行われるデータ処理に焦点を当てる．

1.2 研究目的に対するアプローチ

本研究でデータ処理を改善するために取り組む要素は 2 つに大別できる．1 つは高速化，もう 1 つはプライバシ保護である．本節ではそれぞれのアプローチについて述べる．

1.2.1 高速化

データ処理の高速化には，既存の処理を高速化するアプローチと解の厳密性を捨てて高速化するアプローチがある．

前者の場合，既存のデータベースシステム（データ処理システム）のコンポーネント単位で改善することで，ユーザの入出力を変えることなく高速化が可能である．実際の方法としては，例えばハードウェアを高性能なものに置き換える，処理を並列・分散環境で行うようにする，といったことが考えられる．その中でも多くのデータベースシステムに共通するものとして，クエリオプティマイザを改善することが挙げられる．クエリオプティマイザとは，与えられた論理的なクエリをどのように処理するか，いわゆる実行プランを決定するコンポーネントであり，この実行プランの良し悪しによって実行性能は大きく上下する．そしてクエリオプティマイザの改善はカーディナリティ推定と呼ばれるモジュールの改善が効果的であることが知られている [13]．ここでカーディナリティ推定とは，指定された条件を満たすデータがデータベース内に何件存在するかを推定するというタスクである．クエリオプティマイザではこの推定を元に実行コストの低いと推測される実行プランを選択する．

一方後者の場合，解の厳密性と処理速度のバランスの中で高速化を図ることとなる．ここで近似的な処理のうち，条件を満たす件数を取得する処理（近似カウントクエリ処理）を考えると，これはカーディナリティ推定と等価であることがわかる．またこの近似カウントクエリ処理は，様々な近似的な処理のプリミティブとして利用できる．つまり，近似的な処理を改善するためにはカーディナリティ推定の

改善を行えば良いということがわかる．

どちらのアプローチでもカーディナリティ推定の改善が重要であることから，本研究ではカーディナリティ推定の改善を通して上記 2 つのアプローチの高速化を図る．

1.2.2 プライバシ保護

プライバシ保護を行う場合，どの程度安全性を保証する必要があるのかは自明ではない [14]．そもそも安全性の指標は一意ではなく，状況に応じて使い分けられている．広く知られている一例として k-匿名性があげられる．k-匿名性とは，単独もしくは組み合わせることで個人の特定につながる属性の要素が共通しているレコードをグループ化したときに，すべてのグループが少なくとも k 件以上になっているという性質である [15]．これにより，レコード集合全体から個人を特定しようとしても k 件未満に絞り込まれることがない．k-匿名性を満たすようにデータを加工する処理を k-匿名化という．確かにある個人を k 人から特定することはできないが，背景知識や複数の問い合わせ結果を組み合わせることでプライバシが開示されてしまうことがある [16]．攻撃者がどのように処理結果を悪用するのかや外部知識を利用する可能性などを事前に想定することは難しいため，特定の設定によらないプライバシ保護が望ましい．そのような性質を持つ安全性の指標として差分プライバシ [17] が知られており，GDPR への活用も検討されている [18]．差分プライバシとは，一定程度の識別困難性を保証することによって安全性を担保する指標である．データベース D があるとし，そこに含まれているある 1 つのレコード t だけ取り除いたデータベース D' を仮定する．D' にはレコード t が含まれていないため，t に関するプライバシの開示は起き得ない．このときこの 2 つのデータベース D, D' に対する問い合わせ結果が識別困難であれば，t に関するプライバシが保護され安全といえるという仕組みである．本研究で扱う近似カウントクエリ処理では様々な問い合わせを受け付ける可能性があるため，任意の攻撃に対して安全性を保証できる差分プライバシに焦点を当てる．

1.3 解決すべき課題

前節で見たように，本研究では高速化とプライバシ保護という 2 つの観点があり，推定性能と速度を両立したカーディナリティ推定手法とそこに併用可能な差分プライバシを満たす技術の提案を目指している．本節ではそれぞれの観点における既存手法の問題点について述べる．

1.3.1 カーディナリティ推定における推定性能と速度の両立

ここでは従来から利用されているカーディナリティ推定手法の課題と近年提案されている機械学習を利用したカーディナリティ推定手法の課題を説明する．従来のデータベースシステムでは，属性ごとに独立したヒストグラムを利用してカーディナリティ推定を行うことが一般的である [19]．実世界データであれば複数の属性間に相関があるため，複数の属性に跨るヒストグラムでないと正確な推定はできない．そのため，属性ごとに独立したヒストグラムによる推定はしばしば不正確なものとなる [13]．複数の属性を組み合わせたヒストグラムを保持することも考えられるが，全ての組み合わせを保持するは容量や更新コストの高さから困難である．

この問題を解決するため，様々なカーディナリティ推定手法が提案されている [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]．近年，この中でも機械学習を利用した手法が高い推定性能を報告している．しかしながら，例えば MSCN [28, 29] は学習のためのラベル取得コストが高い，Naru [23] は基礎性能は高いが性能がデータセット製作者に依存する，NeuroCard [24] は問い合わせ対象のテーブルが少ないと性能が低い，DeepDB [25] は事前計算コストとメモリ消費量が大きいといったような課題が残る．Naru，NeuroCard は推定性能が高いことがあるものの，経験則に依存している箇所があり性能が不安定である．MSCN，NeuroCard，DeepDB はスキーマサイズに対して何らかのスケールしない問題を抱えている．そのため，大規模スキーマでの利用を考慮した場合，安定した推定性能と速度を両立しつつ，スキーマサイズに対してスケールするカーディナリティ推定手法が必要とされている．

1.3.2　差分プライバシ

ここでは前述の深層学習技術を対象とした場合に差分プライバシを満たすための手法の課題を説明する．深層学習で差分プライバシを満たすための手法としては，DPSGD [31] が広く知られている．DPSGD はニューラルネットワークを確率的勾配降下法で学習時する際に，勾配を一定のノルムにクリッピングした上で必要なプライバシ強度に応じたノイズを勾配に加算する．これにより，学習結果として得られるモデルは，その構造に依らず任意の推論結果が差分プライバシを満たす．差分プライバシによる安全性が保証される一方で勾配に変更を加えるため，モデル本来の有用性，例えば分類モデルであれば分類性能が低下するというトレードオフが存在する．この有用性の低下はしばしば実用上の問題となることから，安全性を下げることなく有用性を改善する提案が行われている [32, 33]．いずれの手法もニューラルネットワークの冗長性に基づき更新対象のパラメータ数を減らすことで，勾配のクリッピングとノイズ加算の影響を緩和し有用性の向上を図っている．しかしながら，対象の処理が限定的であったりパラメータ数削減に利用するニューラルネットワークの冗長性が部分的であることから，有用性の低下は未だに大きい．

1.4　提案技術の概要

以上のような課題を解決するため，本研究では大規模かつ複雑なスキーマを持つデータ処理の高速化とプライバシ保護を実現するための 3 つの要素技術を提案する．第 2 章，第 3 章にて高速化の鍵となるカーディナリティ推定手法を，第 4 章にて第 2 章，第 3 章の手法に適用可能な差分プライベート学習手法を提案する．以下に各提案の概要を示す．

1.4.1　高速で安定したカーディナリティ推定手法（第 2 章）

第 2 章では，データの分布を捉える深層学習技術に着目し，高速で精度の高いカーディナリティ推定手法を提案することでデータ処理の高速化を図る．

データベースシステムにおいて，カーディナリティ推定はクエリ応答性能に大きな影響力を持つ重要な要素技術である．既存のデータベースシステムでは属性間の依存関係を考慮しないため，カーディナリティ推定の性能を悪化させる原因となっ

ている [13]．近年，機械学習による属性間の依存関係を考慮したカーディナリティ推定技術が提案されているが，学習時の属性順序に依存して推定性能が大きく左右され推論速度も遅いという問題がある [23, 24]．

そこで第 2 章では，深層学習技術の 1 つである Denoising Autoencoder [34, 35] で属性間の依存関係を捉えて密度推定器として学習し，推論時に与えられたクエリに応じたカーディナリティ推定を行う手法を提案する．既存技術と異なり属性順序に依存しないため，少ない推論ステップ数で安定したカーディナリティ推定が可能である．

実世界データを用いた複数のベンチマークにおいて，既存手法のピーク性能と同程度の性能を安定して達成した上で 2〜3 倍の高速化に成功した．特に一般的にデータベースシステムが運用されていることを想定した，GPU のようなアクセラレータを利用しない環境にて，高速化できることが確認された．また，クエリオプティマイザへの応用を想定した実験を行ったところ，想定されるコストの大きいクエリを中心に既存のデータベースシステムと比較して改善が確認された．

1.4.2 スケーラブルな結合カーディナリティ推定手法（第 3 章）

第 3 章では，スキーマに基づいた複数の密度推定器を利用した，少なくとも 1 つの結合操作を含むクエリのカーディナリティ推定手法（結合カーディナリティ推定手法）を提案する．推定精度と速度を両立することにより，クエリオプティマイザの改善を通したデータ処理の高速化と複雑な近似カウントクエリ処理の性能向上を図る．

データベースシステムのクエリオプティマイザにおいて，結合順序最適化は特に重要であることが知られている．実世界データには大きな偏りが含まれるため，適切な結合順序を選択できないと巨大な中間テーブルが発生しパフォーマンス低下を引き起こす．この結合順序最適化は結合カーディナリティ推定の性能に強く依存する．しかしながら，ヒストグラムを利用した従来手法は結合時のテーブル間の相関関係を捉えられないため性能が低く，近年提案されている機械学習を利用した手法は結合の数が増えると推論性能が大きく低下する，計算コストやメモリ消費量が大きいといった理由でうまく機能しない．

そこで第 3 章では，既存手法が機能しないような大規模なスキーマを持つ環境でも利用できる結合カーディナリティ推定手法を提案する．提案手法は外部キー制約

に基づき複数のスキーマとして密度推定器を利用することで，スキーマが複雑な環境でも推定精度と速度を両立する．一般的に，複数の密度推定器を利用すると推定器ごとに推論が独立して性能が低下してしまうが，各密度推定器の推論結果を別の推論器の条件として利用することで高い推定精度を実現する．

実世界データセットを用いた複数のベンチマークにて，既存手法が機能しないもしくは性能が低下するような設定においても，構築コストと推定性能の両立に成功した．また，クエリオプティマイザへの応用を想定した実験を行ったところ，既存のデータベースシステムと比較してクエリ処理を高速化できることが確認された．

1.4.3　カーディナリティ推定と差分プライバシ（第 4 章）

第 4 章では，第 2 章や第 3 章で用いるような深層学習手法に対して，有用性と差分プライバシによる安全性を両立した学習手法を提案する．

既存の DPSGD を改善する手法は，対象の処理が限定的であったり，更新対象パラメータ数の削減も部分的なものだった．そこで第 4 章では，ニューラルネットワークパラメータの 2 つの冗長性に注目し，差分プライバシによる安全性と高い有用性を両立した学習手法を提案する．パラメータを行列として扱い，大域的冗長性を捉えるために低ランク近似を，局所的冗長性を捉えるためにスパース化を行う．既存手法とは異なり，ニューラルネットワークの構造と性質を利用することで低ランク近似とスパース化の併用を実現している．これらにより更新対象パラメーラ数が削減され，安全性を落とすことなく勾配クリッピングやノイズ加算の影響を軽減して有用性の高い学習が可能となる．

第 2 章と第 3 章で提案したカーディナリティ推定に適用した評価を行ったところ，差分プライバシを満たした上で有用なカーディナリティ推定ができることを確認した．その他にも，タスクとして自然言語処理や画像処理，モデル構造として畳み込み層や Attention 層を持つなど，様々な環境を想定した評価を行ったところ，多くのケースで提案手法が安全性を落とすことなく有用性を改善することを確認した．

1.5　本論文の構成

本章以降の内容は以下の通りである．大規模スキーマでの高速なカーディナリティ推定を実現するための提案手法として，第 2 章でコアとなる高速で精度の高いカーディナリティ推定手法について，第 3 章で結合カーディナリティ推定をスキーマが大規模な環境で効率的に行う手法について詳説する．更に第 2 章，第 3 章で提案した手法をパーソナルデータを扱う環境で利用できるようにするための提案手法として，第 4 章で有用性の高い差分プライベート深層学習手法について詳説する．最後に第 5 章では本研究の成果をまとめるとともに，今後の研究課題について述べる．

第 2 章　高速で安定したカーディナリティ推定

2.1　はじめに

第 1 章で述べた通り，データの利活用が広まるに連れてデータベースシステムにおける問い合わせ処理の高速化がより求められている．そのため，クエリオプティマイザや近似クエリ処理に利用されるカーディナリティ推定技術は重要である．

データベース内にレコードが確率的に分布しているという仮定のもと，カーディナリティ推定は様々なアプローチで長年研究されているが [20, 21, 22, 23, 24, 25, 26, 28, 29, 36, 37]，実用されている手法では真の値から 10^4 から 10^8 倍 といったオーダーで推論エラーが発生することがある．一例として，そのエラーを含むカーディナリティを利用したクエリオプティマイザによって選択された実行プランは，正確なカーディナリティを利用した場合と比較してクエリ応答性能で 100 倍以上性能低下することもあると知られている [13]．これは誤ったカーディナリティによりクエリオプティマイザが結合順序やインデックス選択の最適化に失敗してしまうことに起因する．従来の手法では各属性ごとに独立したヒストグラムを統計情報として管理しており，単一属性に対する選択演算のカーディナリティ推定は正確度が高いことが知られている．しかしながら，複数の属性に跨る選択演算や結合演算に関しては，各属性が独立であるという仮定が実際のデータと乖離しており，大きな推論のエラーとして表面化する．例えば OLAP のようなコストの高い分析的クエリが対象となるクエリオプティマイザへの応用の場合，推定性能のクエリ処理全体性能への影響は大きいため，事前処理にコストをかけてもカーディナリティ推定性能を向上させることが求められている．

近年では属性間の相関関係を捉えられず推定エラーが発生してしまうという課題を解決すべく，機械学習を用いたカーディナリティ推定手法が提案されている [21, 22, 23, 24, 25, 26, 28, 29, 37]．これらは 2 つのアプローチに大別される．1 つはワークロードを学習することでカーディナリティ推定を行う手法である [28, 29, 37]．

ワークロードが既知である場合や未知の場合でもランダムに生成されたワークロードを利用することで従来の手法と比較して高い性能を達成している．しかしながら，性能がワークロードに強く依存するため，未知のクエリに弱い・生成したワークロードのラベル取得コストが高く学習に時間がかかる・データベースが更新によって変化すると再学習が必要となるといった課題がある．もう 1 つのアプローチとしてデータを学習するものが挙げられる [23, 24, 25, 26]．自己回帰モデル [38, 39] や Sum-Product Networks [26, 40] を用い，学習時にデータの分布を捉え推論時に述語を適用することで従来の手法と比較して高い性能を達成している．また，データのみを学習に利用することから，事前にワークロードを必要としないという長所も挙げられる．しかしながら，推論コストが高い，経験則に基づくパラメータに性能が左右されてしまうといった課題がある．例えば自己回帰モデルを利用した手法であれば，学習時に固定されてしまう推論可能な属性の順序によって真の値から 10^2 から 10^3 倍の推論エラーが発生する程度に性能が不安定であることが報告されている [23]．事前に属性間の相互情報量を算出し推論性能の高い順序を決定することが検討されているが，データセットに定義された順序（正順）に及ばないと報告されている．正順にはキーとなるような属性を先にしやすいというバイアスにより推論性能が高くなる傾向にあるという推察が行われている [23]．しかしながらこの順序は人手に依るものであり，全てのデータセットで成立するものではないという問題がある．またこのモデルは推論時に多数のサンプルが必要とするため，一般的なデータベースシステム環境にはない GPU のようなアクセラレータが必要となる．その他の選択肢として，多数の順序を同時に扱う自己回帰モデル [38, 41, 42] の利用も考えられるが，いずれの手法も学習・推論コストの高さや扱える順序に制限があることから，多数の属性を扱うカーディナリティ推定への応用は行われていない．

本章では高速で安定したカーディナリティ推定手法の提案を行う．特にコストの高い分析的クエリを対象としたクエリオプティマイザや近似クエリ処理への応用を目的とする．Denoising Autoencoder (DAE) の密度推定としての性質を用いてデータを学習することで，属性ごとの分布だけでなく全テーブルの全属性間の相関関係を捉えたカーディナリティ推定を行う．学習時に推論できる属性の順序が固定されてしまう自己回帰モデルとは異なり，DAE は全属性を並列に扱うため任意の順序で推論できる．これにより，自己回帰モデルで課題だった推論する属性の順序による不安定性のないカーディナリティ推定が実現される．また，推論を DAE の特徴

を活用した動的な順序にすることで，サンプルサイズが小さい場合でも正確度の高い推定を可能とする．これにより，GPU なしの環境でも推論時の応答速度がクエリ処理時間全体と比較して十分に高速となる．

本章の主な貢献点は以下の通りである．

- クエリオプティマイザや近似クエリ処理に応用できる，高速で安定したカーディナリティ推定手法を提案する．Denoising Autoencoder を利用することで，自己回帰モデルで課題とされていた順序による不安定性がなく，安定した推定性能を示す．
- クエリに含まれる情報を利用した動的な順序（推論時ドメイン考慮順）での推論を提案する．これにより，高い推定性能を保ったままサンプルサイズを小さく，つまり高速な推論を可能とする．特に PostgreSQL のようなデータベースシステムで一般的に利用されている（GPU のようなアクセラレータのない）CPU 環境ではサンプルサイズ削減による高速化への寄与が大きく，カーディナリティ推定を複数回呼び出す必要があるクエリオプティマイザやカーディナリティ推定の速度が直に処理性能となる近似カウントクエリ処理への応用が効果的なものとなる．
- 複数の実世界データを用いて，複数の最新のカーディナリティ推定手法との比較を交えた評価を報告する（2.4.1 項，2.4.3 項）．また，カーディナリティ推定の応用先の 1 つであるクエリオプティマイザを利用した，実践的な評価も報告する（2.4.2 項）．

2.2 節でカーディナリティ推定の定式化と DAE について説明を行い 2.3 節で提案手法を詳説する．2.4 節で提案手法の評価を行い 2.5 節で既存研究との関連について述べ，2.6 節で本章をまとめる．

2.2 事前準備

本章以降で利用する主なシンボルの定義を表 2.1 に示す．

表 2.1: 本章以降で利用する主なシンボル

シンボル	定義・概要
A	属性.
V	$v \in V$. 頂点集合. スキーマグラフでは v はテーブルを表し，T とも表記する.
E	$(u, v, c) \in E, c := (c_u, c_v)$. 頂点 u と v を結ぶラベル c 付き有向辺集合. スキーマグラフではテーブル u と v が $u.c_u = v.c_v$ という等号条件で一対多のリレーションシップにあるという外部キー制約を表す.
G	$G := (V, E)$. 頂点集合 V と辺集合 E から構成される辺ラベル付き有向非巡回多重グラフ. データベースのスキーマを表す. 簡単のため, ループで表される自己結合はないものとする.
G_{GLB}	$G_{GLB} := (V_{GLB}, E_{GLB})$. 特にデータベース全体のスキーマを表す辺ラベル付き有向非巡回多重グラフ.
G_Q	クエリ Q が対象とするテーブルとリレーションシップを表す有向非巡回グラフ. 無向グラフとして見たときに連結である. また，データベース全体のスキーマグラフの部分グラフ，つまり $G_Q \subseteq G_{GLB}$ である. 本章では補足がない限り有向木である.

シンボル	定義・概要
R_Q	述語の条件．$R_Q(A)$ は，属性 A のドメインのうちクエリ Q の述語の条件を満たす範囲を表す．$R_Q(A) \subseteq \mathrm{dom}(A)$ である．述語が指定されていない場合は元のドメインと等しく，$R_Q(A) = \mathrm{dom}(A)$ となる．Q の全ての述語の条件を満たす範囲は $\prod_A R_Q(A)$ と表記でき，超直方体である．Q が明らかな場合は R と表記する．
Q	$Q := (G_Q, R_Q)$．クエリであり，クエリグラフ G_Q と述語の条件 $R_Q(A)$ から構成される．
$\mathrm{dom}(A)$	属性 A のドメイン．
J_e	頂点集合 e に対応するテーブルを完全外部結合したテーブル．
N_T	属性のうち，特にテーブル T のマーカーを示すもの．テーブルマーカーについては 2.2.2 項で説明する．
F	属性のうち，特に Fanout を示すもの．Fanout については 2.2.2 項で説明する．
$\mathscr{M}_T$	テーブル T の密度推定器．
$P_T(a_1 \in R(A_1), \ldots, a_n \in R(A_n))$	テーブル T で属性 $A_1, \ldots, A_n$ の条件を満たす同時確率．対象が明らかな場合には T を省略して $P(a_1 \in R(A_1), \ldots, a_n \in R(A_n))$ と表記する．
$\mathscr{C}(Q)$	クエリ Q に基づくカーディナリティ．
$\mathscr{S}(Q)$	クエリ Q に基づくセレクティビティ．
$\hat{\cdot}$	機械学習モデルや統計情報などから得られた推定値．
$\mathbb{1}_c$	条件 c を満たす場合は 1，満たさない場合は 0 となる指示関数．

2.2.1 単一テーブルに対するカーディナリティ推定の定式化

カーディナリティ推定はコストベースオプティマイザで実行プランを決定するためのコスト見積もりに利用される技術である．与えられた述語の条件を満たすレコード群のカーディナリティの推定を行う．ここで，カーディナリティが総行数に対する述語の条件を満たす期待値として表せることから，述語の条件を満たす同時確率に注目する．属性集合を $\boldsymbol{A} = \{A_1, \ldots, A_n\}$ ，タプル集合で表されるテーブルを $T = \{t_1, \ldots, t_m\}$ とし，条件となるクエリを Q とする[1)]．クエリ Q を満たすセレクティビティ $\mathcal{S}(Q)$ は条件を満たすタプルの割合として以下のように表される．

$$\mathcal{S}(Q) = \frac{|\{t \in T \mid \bigwedge_{A \in \boldsymbol{A}} t.A \in R(A)\}|}{|T|}$$

タプル t は属性 $\{A_1, \ldots, A_n\}$ の要素 $\{a_1 \in A_1, \ldots, a_n \in A_n\}$ の組み合わせで構成されることから，その同時確率を $P(a_1, \ldots, a_n)$ とすると，

$$\mathcal{S}(Q) = \sum_{a_1 \in A_1} \cdots \sum_{a_n \in A_n} \left\{ \left(\prod_{A \in \boldsymbol{A}} \mathbb{1}_{a \in R(A)} \right) P_T(A_1 = a_1, \ldots, A_n = a_n) \right\}$$

とも表せる．ただし式から分かるように，この組み合わせは非常に多くなるため直接扱うことは難しい．そのため，各属性が独立であるという仮定 $P_T(A_1, \ldots, A_n) \approx \prod_n P_T(A_i)$ をおくことで，属性ごとのヒストグラムからカーディナリティを推定する手法が広く利用されている．しかしながら，各属性が独立であるという仮定は現実世界データには不適切であり，エラーの主たる原因となっている [13]．

本研究ではそのような実世界データにそぐわない仮定を用いずに厳密な変換のみを扱う．同時確率は乗法定理により，

$$\begin{aligned} P(A_1, \ldots, A_n) = P(A_1) P(A_2 \mid A_1) \\ \cdots P(A_n \mid A_{n-1}, \ldots, A_1) \end{aligned}$$

と展開できる．このとき条件付き確率は任意の属性の順序で等しくなる．ここから

1) ここでは単一テーブルのみを扱っているため，クエリグラフ G_Q は無視する．

セレクティビティは以下の形としても導ける．

$$\mathcal{S}(Q) = \sum_{a_1 \in A_1} \cdots \sum_{a_n \in A_n} \left(\prod_{A \in \boldsymbol{A}} \mathbb{1}_{a \in R(A)} \right) P_T(A_1 = a_1, \ldots, A_n = a_n) \tag{2.1}$$

$$= \sum_{a_1 \in A_1} \left(\left(\left(\mathbb{1}_{a_1 \in R(A_1)} \times P_T(A_1 = a_1) \right) \sum_{a_2 \in A_2} \left(\mathbb{1}_{a_2 \in R(A_2)} \times P_T(A_2 = a_2 \mid A_1 = a_1) \right) \right) \right.$$
$$\left. \cdots \sum_{a_n \in A_n} \left(\mathbb{1}_{a_n \in R(A_n)} \times P_T(A_n = a_n \mid A_{n-1} = a_{n-1}, \cdots, A_1 = a_1) \right) \right) \tag{2.2}$$

ここで $\boldsymbol{X}$ の i 番目の要素に先行する要素を $X_{<i}$ としたとき，$\sum_{a_i \in A_i} P(A_i = a_i \mid A_{<i} = a_{<i})$ は明らかに 1 であり，$\sum_{a_i \in A_i} \mathbb{1}_{a_i \in R(A_i)} P(a_i \mid A_{<i} = a_{<i})$ は $[0, 1]$ の範囲を取るスカラー値となり $R(A_i)$ で表される条件を満たす確率を示している．これらからカーディナリティ $\mathcal{C}$ は，

$$\mathcal{C}(Q) = |T| \cdot \mathcal{S}(Q)$$
$$= |T| \sum_{a_1 \in A_1} \cdots \sum_{a_n \in A_n} \left(\prod_{A \in \boldsymbol{A}} \mathbb{1}_{a \in R(A)} \right) P_T(A_1 = a_1, \ldots, A_n = a_n)$$
$$= |T| \sum_{a_1 \in A_1} \left(\left(\left(\mathbb{1}_{a_1 \in R(A_1)} \times P_T(A_1 = a_1) \right) \sum_{a_2 \in A_2} \left(\mathbb{1}_{a_2 \in R(A_2)} \times P_T(A_2 = a_2 \mid A_1 = a_1) \right) \right) \right.$$
$$\left. \cdots \sum_{a_n \in A_n} \left(\mathbb{1}_{a_n \in R(A_n)} \times P_T(A_n = a_n \mid A_{n-1} = a_{n-1}, \cdots, A_1 = a_1) \right) \right)$$

と表せる．

2.2.2 複数テーブルへの拡張

データベース内に複数のテーブルが存在することはごく一般的であり，複数のテーブルを対象としたカーディナリティ推定も必要とされる．しかしながら，テーブルが複数存在する場合は 2.2.1 項で述べた定式化を適用できない．そこで，予めデータベース内のテーブルを外部キー制約に基づいて完全外部結合を取ったものを仮想的な単一テーブル，つまりユニバーサルリレーションと見なすことで，テーブルが複数存在する場合におけるカーディナリティ推定を実現する．ただし，ユニバーサルリレーションを利用したカーディナリティ推定を行うには追加で 2 つ考慮しなければならない点がある．1 つはクエリに含まれないテーブルの影響を受けて

しまうため直接カーディナリティの推論に利用できない点，もう 1 つはユニバーサルリレーションの実体化は避けなければならないという点である．

Yang らは，Hilprecht らの完全外部結合から内部結合を復元する手法 [25] と Zhao らの Exact Weight Join Sampling [43] を組み合わせることによって，前述の 2 つの課題を解決したカーディナリティ推定手法を提案している [24]．学習はユニバーサルリレーションでサンプリングされたタプルを利用する．推論はクエリに含まれるテーブルのみに絞り込む条件を加え，更に，クエリに含まれないテーブルとの結合による影響を打ち消す操作を行う．これらにより，ユニバーサルリレーションでの適切なカーディナリティ推定を実現している．

具体的には，事前処理として，どのテーブル同士の結合を行うかをスキーマグラフとして定義する[2)]．そして仮想的な属性としてテーブルマーカー（N_T）と Fanout（F_T）を追加する（式 2.3）．このとき $n' = |\boldsymbol{A}'|$ とする．テーブルマーカーとは，当該テーブルが各完全外部結合されたタプルに含まれているかを示す真偽値である．もう 1 つの Fanout とは，あるテーブルのタプルをスキーマグラフに従って結合させたときに対応する結合先テーブルのタプル数である．外部結合であるため，たとえ結合先テーブル内に対応するタプルが存在していない場合でも $NULL$ が結合対象となり必ず 1 以上の整数値となる．これらを利用してスキーマグラフのルート[3)]から結合サンプリングを行う．このとき，結合後の分布に従ったサンプルとするため，Fanout を重みとして結合対象のタプル選択を行う．これを復元ありサンプリングとして必要な回数行い，得られたテーブルマーカーと Fanout が付与されたユニバーサルリレーションのタプル集合を学習に利用する．

推論時にはクエリに含まれないテーブルの影響を打ち消す 2 つの操作を行う．まず，クエリに含まれるテーブルと含まれないテーブルを判別する（式 2.4，式 2.5）．クエリに含まれる結合に応じてテーブルのマーカーが真であることをクエリの条件として追加する（式 2.6）．内部結合であれば両テーブルのマーカーが，外部結合であれば起点となるテーブルのマーカーが対象となる．これにより，クエリに含まれるテーブルが存在するタプルに絞り込まれる．加えて同時確率を推定する際に，クエリに含まれないテーブルの Fanout でダウンスケーリングを行い（式 2.7），式 2.1- 式 2.2 と同じ展開を行う（式 2.8）．これにより，クエリに含まれないテーブル

2） 同じテーブルに対して異なる属性での結合を複数定義することも可能である．
3） ルートは任意の頂点で良い．

との外部結合で増えたタプル数の影響が打ち消され，得られる確率分布がクエリに含まれるテーブルのみに基づくものとなる．なお，式 2.7 や式 2.8 では，分子を条件とした F_T が必要である．

$$\boldsymbol{A}' = \bigcup_{T \in \boldsymbol{T}} T.\boldsymbol{A} \cup N_T \cup F_T) \tag{2.3}$$

$$\boldsymbol{T}_{used} = \{T \in \boldsymbol{T} \mid T \in V_Q\} \tag{2.4}$$

$$\boldsymbol{T}_{notused} = \{T \in \boldsymbol{T} \mid T \notin V_Q\} \tag{2.5}$$

$$R_Q{}' = R_Q \wedge \bigwedge_{T \in \boldsymbol{T}_{used}} N_T \ is \ True \tag{2.6}$$

$$Q' = (R_Q{}', G_Q)$$

$$\mathcal{S}(Q') = \frac{\sum_{a_1 \in A_1} \cdots \sum_{a_{n'} \in A_{n'}} \left(\prod_{A \in \boldsymbol{A}'} \mathbb{1}_{a \in R(A)} \right) P_T(A_1 = a_1, \ldots, A_{n'} = a_{n'})}{\prod_{T \in \boldsymbol{T}_{notused}} F_T} \tag{2.7}$$

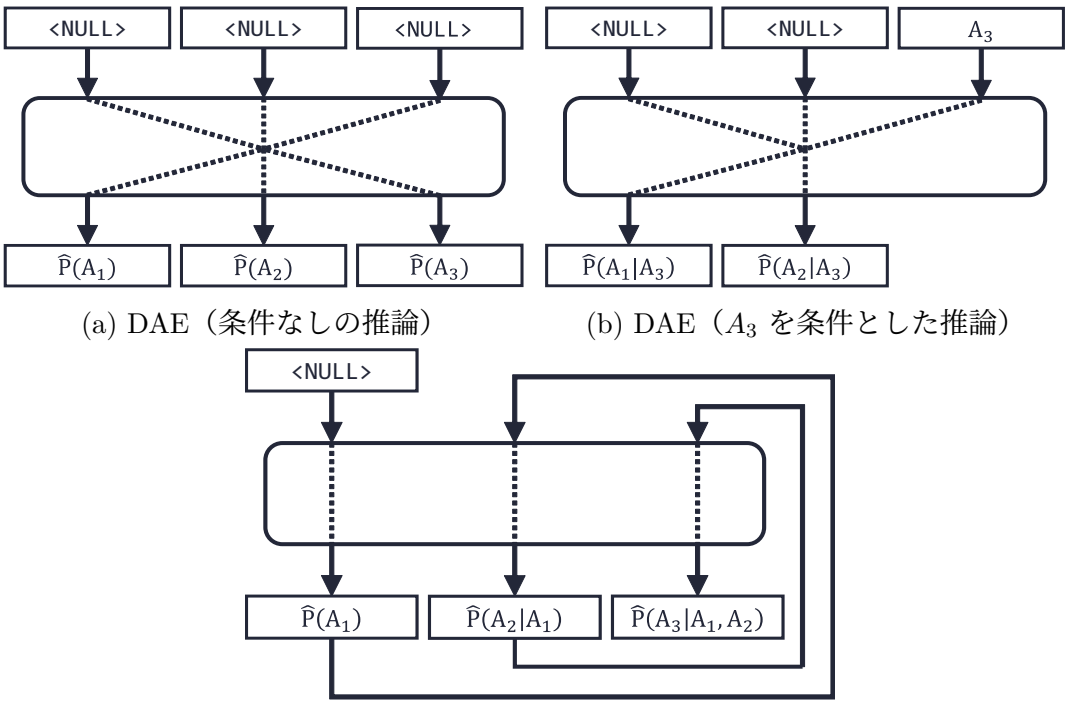
$$= \frac{\begin{array}{l} \sum_{a_1 \in A_1} \Bigg(\Big(\big(\mathbb{1}_{a_1 \in R(A_1)} \times P_T(A_1 = a_1) \big) \\ \quad \sum_{a_2 \in A_2} \big(\mathbb{1}_{a_2 \in R(A_2)} \times P_T(A_2 = a_2 \mid A_1 = a_1) \big) \Big) \\ \quad \cdots \sum_{a_{n'} \in A_{n'}} \big(\mathbb{1}_{a_{n'} \in R(A_{n'})} \times P_T(A_{n'} = a_{n'} \mid A_{n'-1} = a_{n'-1}, \cdots, A_1 = a_1) \big) \Bigg) \end{array}}{\prod_{T \in \boldsymbol{T}_{notused}} F_T} \tag{2.8}$$

2.2.3 DAE: Denoising Autoencoder

DAE [34, 35] とは，Autoencoder を拡張する形で提案されたモデルであり，入力に何かしらのノイズを加えて学習を行う．特徴の 1 つとして，生成モデルとしての性質も持つことがあげられる．この特徴を活かした応用として，ノイズにマスクを用いた自然言語処理の文章生成手法 [44, 45, 46] がある．これらの手法は，DAE とは別に自然言語処理のために研究されていた自己回帰モデル [38, 39] に DAE の発想を取り入れたものである．自己回帰モデルでは推論時間が扱うシーケンス長に比例する一方で，DAE を取り入れたモデル[4)]では推論が同時並列的に可能なため，文章

4) この場合，自己回帰モデルの対比として非自己回帰モデルとも呼ばれる．

生成を高速に行えるという長所を持つ．具体的には，一部をマスクした単語シーケンスを入力し，生成したい単語シーケンスを復元するように学習を行う．結果として，単語シーケンスの一部を条件として入力することで残りの任意の単語を並列に推論を可能なモデルとなる図 2.1．提案手法では，この単語シーケンスをタプルと置き換えることでテーブル形式データ上のカーディナリティ推定を行う．より一般的な視点では次のような違いを生む．自己回帰モデルの場合は学習時に順序が固定されるため，例えば $A_1, \ldots, A_n$ という順序で学習すると，推論可能な条件付き確率分布は $P(A_1), \hat{P}(A_2 \mid A_1), \ldots, P(A_n \mid A_{n-1}, \ldots, A_1)$ に限られるが，DAE の場合は任意の条件付き確率分布を得られる．これにより，カーディナリティ推定では，任意の順序で述語サブセットの推論を行いつつ結果を再利用した述語全体の推論も可能となっている．



(a) DAE（条件なしの推論）　(b) DAE（A_3 を条件とした推論）

(c) 自己回帰モデル

図 2.1: DAE（非自己回帰モデル）と自己回帰モデル．NULL はマスクあるいは入力の初期値を表す．

主な実装として，自己回帰モデルの 1 つである Transformer [39] をベースとしたモデルが挙げられる [46]．Self-Attention 時のマスクを外すことによって DAE とし

ての性質が実現される．

2.3 提案手法

提案手法では動的な順序での推論を行うため DAE を密度推定器として利用し，述語の条件を満たす確率を推論することで，2.2 節の式からカーディナリティ推定を実現する．なお，非自己回帰的性質を持つ任意のモデルを利用可能であるが，本章では構造が平易で速度面を重視した多層パーセプトロン (MLP) によるモデル（図 2.2a）と Attention による高度な推論が可能な Transformer [39] をベースとしたモデル（図 2.2b）の 2 つを取り扱う．提案手法はデータの分布のみが学習の対象でワークロードを事前に必要としないため，単一の学習済みモデルで任意クエリのカーディナリティ推定が可能である．対象データベースに複数のテーブルが含まれる場合は， 2.2.2 項で述べたように，全てのテーブルを完全外部結合したユニバーサルリレーションとして見なすことで，結合を含むカーディナリティ推定も可能である．

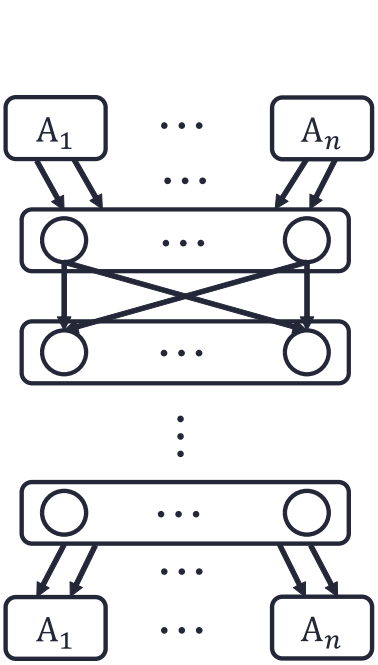
以降では，学習フェーズ，推論フェーズそれぞれについて説明する．

2.3.1 学習フェーズ

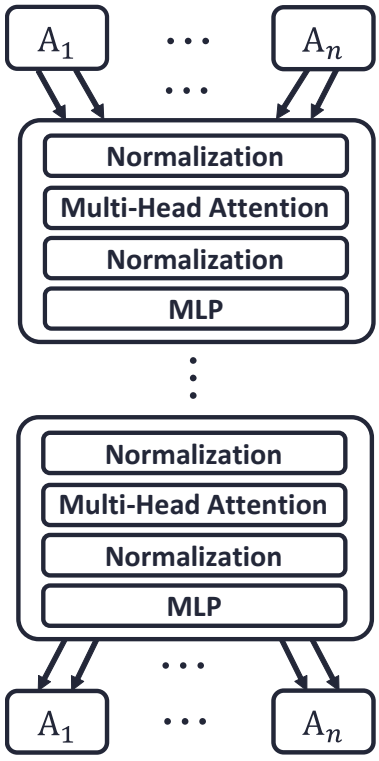
カーディナリティ推定に利用可能な DAE による密度推定の学習について述べる．学習はデータセット全体またはデータセットからランダムサンプリングしたタプル単位で行われる．

具体的には，まず属性集合 $\boldsymbol{A}$ からマスクする属性 $\tilde{A} \in \tilde{\boldsymbol{A}}$ とマスクしない属性 $\check{A} \in \check{\boldsymbol{A}}$ を選択する（$\tilde{\boldsymbol{A}}^{\mathsf{C}} = \check{\boldsymbol{A}}$）．その後，図 2.3 のようにテーブルからタプルを取り出し，マスクしたタプルを入力，マスクしていないタプルをターゲットとする．そしてマスクした属性として出力された確率分布 $\hat{P}(\tilde{A} \mid \check{\boldsymbol{A}})$ とターゲット $P(\tilde{A} \mid \check{\boldsymbol{A}})$ との交差エントロピーを算出し，その総和をマスク数 $|\tilde{\boldsymbol{A}}|$ でスケーリングしたものをロス $\mathscr{L}$ として学習に利用する（式 2.9）．タプルごとにマスクをランダムな数，ランダムな属性に選択することで，自己回帰モデルとは異なり，任意の属性を条件とした任意の属性の確率分布を推論可能なモデルとなる．なおドメインが属性ご

5） Decoder 部分のみを利用する．



(a) MLP による DAE



(b) Transformer による DAE[5)]

図 2.2: 提案手法で利用する DAE

とに異なることから，埋め込みは属性ごとに One-hot エンコーディングや Entity Embeddings [47] を利用して行う．

$$\mathscr{L} = -\frac{1}{|\tilde{\boldsymbol{A}}|} \sum_{\tilde{A} \in \tilde{\boldsymbol{A}}} \sum_{\tilde{a} \in \tilde{A}} P(\tilde{A} = \tilde{a} \mid \breve{\boldsymbol{A}}) \log \hat{P}(\tilde{A} = \tilde{a} \mid \breve{\boldsymbol{A}}) \tag{2.9}$$

2.3.2 推論フェーズ

前節の学習で得られたモデルを利用してカーディナリティを推定する手法について述べる．2.2.1 項で述べたように，カーディナリティは述語を考慮した同時確率と総タプル数の積と等価であることから，提案手法では条件付き確率からカーディナリティを推定する．ここで $\sum_{a_i \in A_i} P(A_i = a_i \mid A_{<i} = a_{<i})$ が常に 1 であることに注意すると，述語の対象となっていない属性の確率分布は求める必要がないことが分かる．つまり，例えば $\boldsymbol{A} = \{A_1, \ldots, A_n\}$, $R(A_1) \subset \mathrm{dom}(A_1)$，$R(A_2) = \mathrm{dom}(A_2)$，$R(A_3) \subset \mathrm{dom}(A_3)$ であれば以下のように A_1 と A_3 に関する確率分布から同時確率

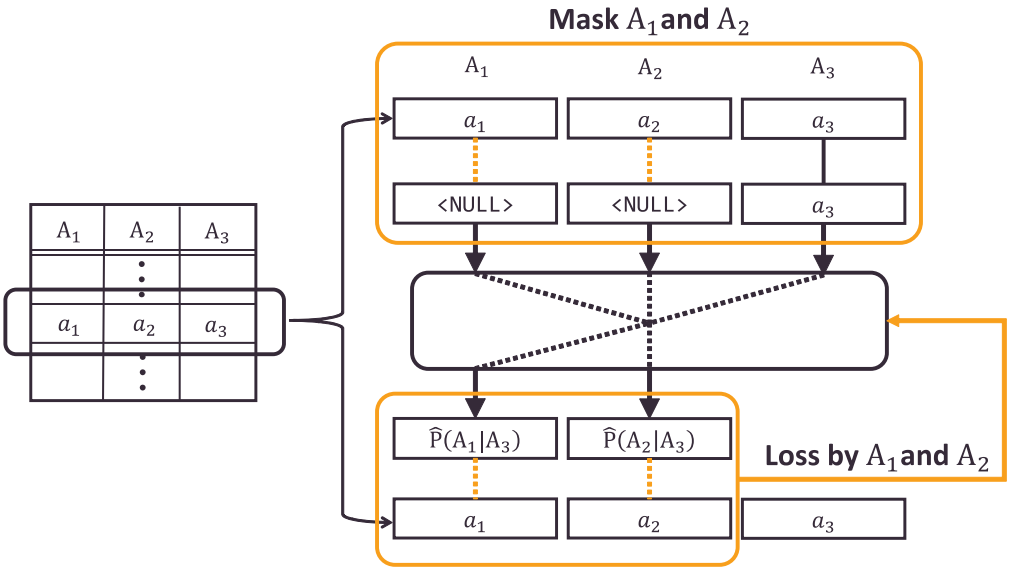


図 2.3: カーディナリティ推定のための DAE の学習

が推定できる．

$$
\begin{aligned}
\mathscr{C}(Q) &= |T| \cdot \mathcal{S}(Q) \\
&= |T| \sum_{a_1 \in A_1} \sum_{a_2 \in A_2} \sum_{a_3 \in A_3} \left(\prod_{A \in \boldsymbol{A}} \mathbb{1}_{a \in R(A)} \right) P_T(A_1 = a_1, A_2 = a_2, A_3 = a_3) \\
&= |T| \sum_{a_1 \in A_1} \Bigg(\Big(\big(\mathbb{1}_{a_1 \in R(A_1)} \times P_T(A_1 = a_1) \big) \\
&\qquad \sum_{a_2 \in A_2} \big(\mathbb{1}_{a_2 \in R(A_2)} \times P_T(A_2 = a_2 \mid A_1 = a_1) \big) \Big) \\
&\qquad \sum_{a_3 \in A_3} \big(\mathbb{1}_{a_3 \in R(A_3)} \times P_T(A_3 = a_3 \mid A_2 = a_2, A_1 = a_1) \big) \Bigg) \\
&= |T| \sum_{a_1 \in A_1} \Bigg(\Big(\big(\mathbb{1}_{a_1 \in R(A_1)} \times P_T(A_1 = a_1) \big) \\
&\qquad \sum_{a_3 \in A_3} \big(\mathbb{1}_{a_3 \in R(A_3)} \times P_T(A_3 = a_3 \mid A_1 = a_1) \big) \Bigg)
\end{aligned}
$$

どのようにモデルを利用して確率を推論しカーディナリティを推定するかについて詳説する．提案手法では，自己回帰モデルを利用する先行研究である Naru [23] で提案された Progressive Sampling を DAE に拡張する．この Progressive Sampling はモンテカルロ法によるサンプリングを介した推定を行うことで，等号条件だけでなく範囲条件を含むクエリのカーディナリティ推定を容易に可能とする．自己回帰

モデルを利用した範囲条件の推論では，適合する全ての組み合わせ（範囲条件の直積集合）を推論する必要があり，ドメインが広がると現実的ではなくなる．そこで Progressive Sampling では全ての組み合わせを推論するのではなく，適合する組み合わせからランダムサンプルに基づいて推論を行いその平均を近似解として扱う．このとき，条件を満たす範囲から一様にサンプルを選択するのではなく先行する推論結果の分布に基づく選択を行うことで，より少ないサンプルサイズで真の値に近づきやすくなる．

$$A_1 \in \{2, 7, 8, 9\} \wedge A_2 > 10.0 \wedge A_3 =' Tokyo'$$

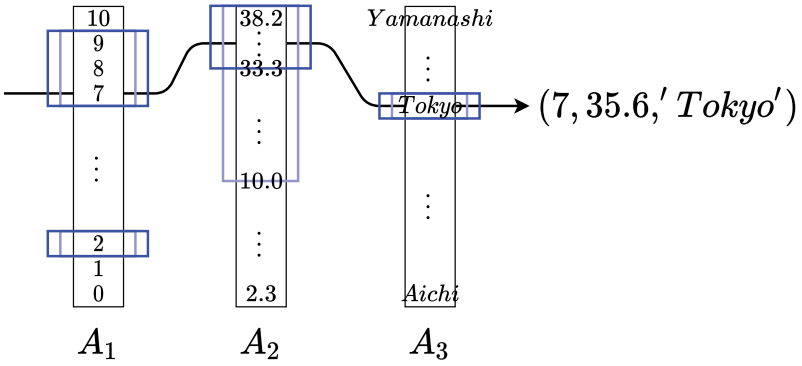


図 2.4: Progressive Sampling で A_1, A_2, A_3 の順に推論を行う 1 試行の例．薄い青色がクエリによる範囲を，濃い青色が先行する属性でサンプルされた要素に基づく範囲を示す．実際には推定された確率分布を重みとした範囲となる．

具体的には図 2.4 のように，A_2 のサンプリングでは先行する $A_1 = 7$ というサンプルを条件として推論した結果の分布を利用する．$A_1 = 7 \wedge 10.0 < A_2 < 33.3$ となるデータが存在しないデータセットと仮定した場合，クエリで指定された範囲（$A_2 > 10.0$）と比較してより狭い範囲（$A_2 > 33.3$）を中心としたサンプリングとなり，後続の A_3 の推論が A_1, A_2 の分布に則したものとなる．なお，サンプルサイズは大数の法則に基づき大きくすれば真の値に近づくが推論コストが増えるハイパーパラメータとなる．カーディナリティ推定の応答時間は推論コストと比例関係であるため，サンプルサイズは現実的な値にする必要がある．ここで，乗法定理による同時確率の式展開に注目する．この式展開自体は厳密なものであるが，Progressive Sampling では順序によって性能が変化する．Progressive Sampling は先行する推論結果をサンプリングし後続の推論に利用することから，式展開の順に依存した推論

となっている．特に，正確度が低い属性の推論の先行は後続の推論のエラー原因となるため，対象となるクエリに応じてより性能が高くなる属性順を見つける必要がある．自己回帰モデルでは属性順が学習の段階で固定されるため事前に決定する必要があるが，属性順候補は順列なので $|\boldsymbol{A}|!$ 通りと非常に多く，属性順毎に 1 から学習して適切な属性順を見つけることは困難である．

一方 DAE を利用する提案手法では任意の属性順で推論できる．この特徴を活かして性能の高い推論を実現するため，狭いドメインのほうが推論が容易で精度が高くなりやすいというヒューリスティックに基づき，与えられた述語を適用したときのドメインが狭い属性順（推論時ドメイン考慮順）で推論を行う（このヒューリスティックによる影響は 2.4.3 項に示す）．Algorithm1 に推論時のドメインサイズを考慮したカーディナリティ推定アルゴリズムを示す．

表 2.2: Algorithm1 で利用する関数の定義

関数名	定義
DrawSample($dist_A$)	属性 A の分布 $dist_A$ から，分布を重みとしたサンプリングを行う．
Encode(a)	属性 A の要素 a を埋め込む．One-hot ベクトルや Entity Embeddings [47] などが利用できる．
SortBy($\boldsymbol{A}, key\_func$)	属性集合 $\boldsymbol{A}$ を key_func に基づいて昇順に並べ替える．
Mean($\boldsymbol{a}$)	数値集合 $\boldsymbol{a}$ の平均値を計算する．

Algorithm 1 DAE を利用したカーディナリティ推定アルゴリズム（推論時ドメイン考慮順）

Input: Density Estimator $\mathscr{M}$, Predicate ranges of query Q R_Q, Attributes $\boldsymbol{A}$, Universal relation size $|J_{univ}|$
Output: Estimated cardinality $\hat{\mathscr{C}}(Q)$

procedure ESTIMATE_N ▷ Sample size is N
  procedure ESTIMATE($\mathscr{M}, R_Q, \boldsymbol{A}_Q, \boldsymbol{A}_{fanout}$)
    Initialize $inputs$ with nulls
    $\hat{prob} \leftarrow 1.0$
    for $A \in \boldsymbol{A}_Q$ **do** ▷ Estimate probabilities of predicates
      $\hat{dist}_A \leftarrow \mathscr{M}(inputs)$ ▷ Forward
      $\hat{dist'}_A \leftarrow \{\hat{dist}_a * (a \in R_Q(A)) \mid a \in A\}$ ▷ Filter distribution by R_Q
      $\hat{prob} \leftarrow \hat{prob} * \sum_{a \in A} \hat{dist'}_a$
      $\hat{a} \leftarrow$ DRAWSAMPLE($\hat{dist'}_A$)
      $inputs[A] \leftarrow$ ENCODE($\hat{a}$)
    end for
    for $A \in \boldsymbol{A}_{fanout}$ **do** ▷ Estimate fanouts
      $\hat{dist}_A \leftarrow \mathscr{M}(inputs)$ ▷ Forward
      $\hat{a} \leftarrow$ DRAWSAMPLE($\hat{dist}_A$)
      $\hat{fanout}[A] \leftarrow \hat{a}$
      $inputs[A] \leftarrow$ ENCODE($\hat{a}$)
    end for
    return $\hat{prob}, \hat{fanout}$
  end procedure

  $\boldsymbol{A}_Q \leftarrow \{A \in \boldsymbol{A} \mid |R_Q(A)| < \mathrm{dom}(A)\}$ CommentFilter attributes by predicates
  $\boldsymbol{A}_Q \leftarrow$ SORTBY($\boldsymbol{A}_Q, key : A \rightarrow |R_Q(A)|$) ▷ Sort attributes by domain size
  for $i \in \{1, \ldots, N\}$ **do** ▷ Batched in practice
    $probs[i], fanouts[i], \leftarrow$ ESTIMATE($\mathscr{M}, R_Q, \boldsymbol{A}_Q, \boldsymbol{F}$)
  end for
  $\hat{\mathcal{S}}(Q) \leftarrow$ MEAN($\hat{\mathcal{S}} / \prod_{fanout \in fanouts} fanout$)
  $\hat{\mathscr{C}}(Q) \leftarrow |J_{univ}| \times \hat{\mathcal{S}}(Q)$
  return $\hat{\mathscr{C}}(Q)$
end procedure

この Progressive Sampling を拡張したアルゴリズムは DAE モデル $\mathscr{M}$，クエリの条件を満たす範囲 R_Q，テーブル内の属性集合 $\boldsymbol{A}$，ユニバーサルリレーション内の総タプル数 $|J_{univ}|$ を受け取る．まず先に述べたように，述語の対象となっていな

い属性の確率分布は求める必要がないため述語の対象となっている属性を取り出し（21 行），続けてドメインサイズ昇順に並べ替える（22 行）．次に乗法定理に基づく確率分布の推論と同時確率の計算をモンテカルロ法によるサンプリングで行うため，そのサンプルサイズである N 回だけ ESTIMATE を実行する（23-25 行）．ESTIMATE は DAE モデル $\mathscr{M}$，クエリの条件を満たす範囲 R_Q，ソート済み属性集合 $\boldsymbol{A}$ を受け取る．初めにモデルへの入力と同時確率を初期化する（3-4 行）．その後与えられた属性順に $\mathscr{M}$ で順伝播による推論 (Forward) を行う（6 行）．推論された確率分布 $\hat{dist}_A$ に述語を適用し，その総和で $\hat{prob}$ を更新する（8 行）更に確率分布を重みとしたサンプリング (DRAWSAMPLE) を行い属性 A の要素 a を得る（9 行）．この要素をエンコード (ENCODE) し次の入力とする（10 行）．同様に Fanout の推論を行う（12 - 17 行）．得られた N 個の推論された確率と Fanout の商の平均がクエリ Q のセレクティビティの推論値 $\mathcal{S}(\hat{Q})$ となる．最後に，セレクティビティと総タプル数 $|J_{univ}|$ の積をカーディナリティの推論値 $\mathscr{C}(\hat{Q})$ として返却する（26-28 行）．なお実際には，23-25 行のループは行列計算としてバッチ化される．

2.4 評価実験

提案手法の評価実験と結果に基づく考察を行う．まず 2.4.1 項で総合的な評価を報告した上で，2.4.2 項でクエリオプティマイザへの応用を想定した評価を報告する．最後に 2.4.3 項で推論時のハイパパラメータである属性順やサンプルサイズが性能に与える影響について評価を報告し議論を行う．

データセットとベンチマーク：評価には 2 つのベンチマークを利用した．1 つは DMV ベンチマークである．ニューヨークの乗り物に関するデータセットである DMV [48] を対象に動作する人工的に生成された 2000 個のクエリから成る．これらのクエリには 4〜10 個の等号条件と範囲条件が含まれる．DMV データセットは単一テーブルであるため結合はない．もう 1 つは Join Order Benchmark [13] (JOB-light [28]) である．俳優，映像エンターテイメントやビデオゲーム等に関するデータセットである IMDb [49] を対象に動作する 70 個のクエリから成る．複数の等号条件と範囲条件があり，多数テーブルの自然結合も含む．ワークロードが事前に固定されている環境を想定して，IMDb データセットのうち，JOB-light に必要なテーブル・属性のみに絞り込んだデータセット（IMDb-JOB-light）も利用す

る．各データセットの統計的な概要は表 2.3 に示す通りである．なお，IMDb データセットと IMDb-JOB-light データセットは結合を扱うため，2.2.2 項で述べた通り Exact Weight Join Sampling [43] により表記のタプル数（サンプルサイズ）だけ取得した結合サンプルを利用する．

表 2.3: データセット概要

データセット	テーブル数	行数	属性数	タプル数
DMV [48]	1	11.6M	11	11.6M
IMDb [49]	16	4〜36M	2〜17	10M
IMDb-JOB-light [28, 49]	6	4〜36M	2〜4	10M

評価指標：推定性能の評価指標には，真のカーディナリティを $\mathscr{C}$，推定されたカーディナリティを $\hat{\mathscr{C}}$ として式 2.10 で表される Q-Error [13] を用いる．これは推定されたカーディナリティが真の値から何倍離れているかを表す無次元の値であり，常に 1 以上で小さい方が優れていることを示す．

$$\text{Q-Error} := \frac{\max(\hat{\mathscr{C}}, \mathscr{C})}{\min(\hat{\mathscr{C}}, \mathscr{C})} \tag{2.10}$$

もう 1 つの推定性能の評価指標として，PostgreSQL のクエリオプティマイザが特定のカーディナリティ推定値を利用して作成するプランの実行コストの推定値（PostgreSQLPlanCost）を用いる．カーディナリティ推定値はクエリオプティマイザ内での結合順最適化やインデックス選択で利用されることでプラン作成に影響を持つ．クエリオプティマイザにおけるコストモデルの影響がカーディナリティ推定と比較して十分小さいこと [13] と結合順は共通して動的計画法による探索が行われることを考慮すると，PostgreSQLPlanCost は利用されたカーディナリティ推定手法の性能に基づくクエリオプティマイザへの貢献を表す値と見なせる．値が小さいほど作成されたプランの実行時間が短いことを期待することができ，優れていることを示す．

推論速度の評価指標には，推論モジュールに各クエリが入力されてからカーディナリティ推定値を返却するまでの時間の平均（平均応答時間）を用いる．学習速度の評価指標には，データロードのような共通化される処理を除いた，各手法の学習処理のみの時間を用いる．

手法：提案手法として，(1) DAE に多層パーセプトロン (MLP) を利用するもの (Ours (MLP)) と (2) Transformer [39][5] (Trm) を利用するもの (Ours (Trm)) の 2 つの実装を利用する．比較手法には，実際のデータベースシステムである (3) PostgreSQL（バージョン 11.8）と (4) 自己回帰モデルを用いることでカーディナリティ推定タスクで最高性能が報告されている Naru [23, 50]/NeuroCard [24, 51][6] を用いる．Naru/NeuroCard の自己回帰モデルの実装にはマスクを用いた MLP である MADE を利用する．加えて，Naru には Transformer を用いた実装も比較対象として用いる．Naru/NeuroCard に共通するハイパパラメータである属性順には，[23, 24] に従い各データセットに定義された順序（正順）を利用する．また，属性順による影響を見るため，正順を反転させた順序（逆順）も利用する．加えて，Sum-Product Network を拡張することで Naru/NeuroCard と同等程度の性能を報告している (5) DeepDB [25, 52] と (6) FLAT [26, 53] も比較対象として利用する．

既存研究のハイパパラメータはそれぞれの論文または公開されているソースコード [23, 24, 25, 26, 50, 51, 52, 53] で報告されている値を利用する．同規模・同系統の既存手法との推論性能・学習時間・応答時間の比較を公平に行うため，提案手法のハイパパラメータは可能な範囲で同系統の既存研究のモデルと同じものを利用する．DMV データセットを利用する実験では Ours (MLP) と Naru (MADE)，Ours (Trm) と Naru (Trm) が，IMDb データセットを利用する実験では Ours (MLP) と NeuroCard (MADE) が対応する．後者の Ours (Trm) は [51] を参考にする．

実行環境：すべての手法で同じ 16CPUs（最大 2.5GHz）・メモリ 64GB・Nvidia Tesla T4 GPU を備えた実行環境にて評価を行う．評価対象の手法のうち，提案手法，Naru と NeuroCard は GPU による高速化が可能である．なお，GPU を利用しない CPU のみの評価を行う場合はプログラムから GPU を利用できないように設定した上で同じ環境を利用する．

2.4.1 総合的な評価

評価の指針とモチベーション：各手法の Q-Error（Median・90 パーセンタイル・95 パーセンタイル・99 パーセンタイル・最大値）と平均応答時間から総合的な性能を評価し，提案手法の有用性を確認する．Progressive Sampling でモンテカルロ

6) NeuroCard は Naru を複数テーブルに拡張した手法である．

法によるサンプリングが必要な手法は推定性能と応答速度のバランスを考え，[23]の報告にも利用されているサンプルサイズ 1000 に固定して比較を行う．

結果と考察：単一テーブルデータセットを対象とする DMV ベンチマークでの結果を表 2.4 に，複数テーブルデータセットを対象とする JOB-light ベンチマークでの結果を表 2.5 に示す．なお，属性順の正順は各データセット内で定義された順序を，逆順は正順の逆を，推論順は推論時に与えられた述語を適用した際のドメインの小さい順（推論時ドメイン考慮順）を表す．

表 2.4 から Q-Error を指標として見ると，提案手法は多くの項目で高い性能を示し，例えば PostgreSQL と比較すると，95 パーセンタイルでは 1/40 程度であった．また，自己回帰モデルを利用する Naru は MADE を用いた実装で正順の場合はほとんどの項目で最も高い性能が確認されたものの，逆順の場合，性能が 10^1 から 10^5 倍以上悪化するケースが見受けられた．Naru はデータ製作者に依存したヒューリスティックに基づいて正順を選択しており，データセットによっては DMV での逆順のような性能が低い順序が選択される可能性があることに注意されたい．一方提案手法にはこのような性能のばらつきがない中で，Naru のピーク性能となっている MADE を利用した実装で正順で推論を行うケースに近い性能を達成していることから，安定性と推論性能を兼ね備えた手法と言える．

多数の自然結合を含みより複雑な条件を持つ JOB-light ベンチマークの結果を表 2.5 に示す．学習は複数のデータセットで行い，共通のワークロードで評価する．学習データセットの差異は想定環境の差異であるため，性能比較は各学習データセットごとに行う．IMDb データセット，つまりワークロードが参照する属性数が全属性数と比較して少ないデータセットを学習した場合に注目すると，MLP を利用した提案手法が多くのケースで最も高い性能を示すことが確認された．一方で Transformer を利用した場合は性能が低下するケースが見受けられる．前者をクエリレベルで調査したところ，一部の頻度が非常に低い要素の影響でカーディナリティを極端に低く推定してしまい，結果として Q-Error が大きくなるクエリがあることが確認された．同一の結合サンプルを学習している NeuroCard ではこの事象は見受けられなかった．このような差異が生まれた原因として，提案手法で利用した Transformer モデルがサンプルに過学習していることが考えられる．ドロップアウトのような汎用的な正則化の導入が今後の課題である．IMDb-JOB-light データセットを学習した場合は提案手法よりも NeuroCard（MADE・**正順**）が高い性能を

示した．NeuroCard に注目すると，表 2.4 の Naru の結果と同じく，順序によって性能が不安定であることが確認された．つまり，データセットによっては NeuroCard（MADE・**正順**）のような高い性能にならない可能性を意味する．なお，DeepDB はデータセットを絞り込まないと学習に 100GB を超えるメモリが必要となり実行できなかった．これはたとえワークロードが簡単なものでもデータセットが大規模化すると応用が困難になることを表す．

次に平均応答時間を指標とした結果をまとめる．自己回帰モデルを用いた既存手法と比較して推論回数を削減できたことにより，同規模・同系統のモデルを利用した既存手法と比較して最大で 3 倍程度の高速化に成功した．特に表 2.5 の IMDb のような，モデルが扱う全属性数と比較してワークロードが利用する属性数が少ない場合，推論回数削減効果が大きくなる傾向が確認された．クエリオプティマイザへの応用であれば複数回呼びされるため最適化処理全体で見ると差が大きくなること，近似カウントクエリ処理への応用であればカーディナリティ推定の高速化がそのまま近似クエリ処理の高速化になることを考えると，この推論の高速化は重要である．一方で，ヒストグラムに基づく PostgreSQL や Factorize-Sum-Split-Product Network を利用した FLAT は，推定性能は比較的低い傾向があるものの，応答時間で 2 倍以上高速なケースが確認された．提案手法では独立な推論処理をサンプルサイズだけ行うことから，2.4.3 項の結果で述べるようにサンプルサイズの削減による速度向上や低レイテンシかつ並列度の高い実行環境の導入によりこれらの手法より大きな速度向上が期待できる．

最後に学習時間を指標とした結果をまとめる．機械学習を用いた手法の中では，Sum-Product Network をベースとした DeepDB や FLAT が特に高速であった．以降 Ours (MLP)，MADE を利用した手法，Transformer を利用した手法という順となった．同規模である Ours (MLP) と Naru (MADE)，Ours (Trm) と Naru (Trm) や Ours (MLP) と NeuroCard (MADE) を比較すると提案手法が常に高速であった．

7） EXPLAIN 文を介した応答時間を報告するため，参考値である．カーディナリティ推定のみの所要時間は掲載している時間より短くなる．

8） 推論性能はワークロードに依存するため IMDb の結果のみ報告する．

表 2.4: 各手法の DMV データセットでの学習時間と
DMV ベンチマークでの Q-Error と平均応答時間

手法（モデル・属性順）	50%th	90%th	95%th	99%th	Max	応答時間 (ms)	学習時間 (min)
PostgreSQL[7]	1.27	2.22	57.9	$1.4 \cdot 10^3$	$7.6 \cdot 10^4$	2.93	—
Naru (MADE・**正順**)	1.04	1.19	1.32	2.00	8.00	10.3	29.3
Naru (MADE・**逆順**)	60.8	505	751	$2.0 \cdot 10^3$	$4.2 \cdot 10^5$	10.4	28.6
Naru (Trm・**正順**)	1.09	2.63	5.34	$9.7 \cdot 10^5$	$9.9 \cdot 10^5$	99.5	153
Naru (Trm・**逆順**)	1.18	10.8	204	$1.0 \cdot 10^3$	$2.1 \cdot 10^4$	103	155
DeepDB	1.07	2.00	4.76	30.0	288	10.5	2.00
FLAT	1.07	1.80	3.95	35.1	$1.3 \cdot 10^5$	2.32	1.80
Ours (MLP・提案順)	1.03	1.24	1.45	2.42	49.0	6.98	24.1
Ours (Trm・提案順)	1.03	1.20	1.36	2.23	12.3	52.2	89.0

表 2.5: 各手法の IMDb・IMDb-JOB-light データセットでの学習時間と
JOB-light ベンチマークでの Q-Error と平均応答時間

	手法（モデル・属性順）	50%th	90%th	95%th	99%th	Max	応答時間 (ms)	学習時間 (min)
IMDb	PostgreSQL[7][8]	7.44	163	$1.1 \cdot 10^3$	$2.8 \cdot 10^3$	$3.5 \cdot 10^3$	3.44	—
	NeuroCard (MADE・**正順**)	1.79	9.00	19.5	36.5	43.2	160	391
	NeuroCard (MADE・**逆順**)	6.04	72.3	118	298	400	158	327
	DeepDB	*—Out of memory to train (100GB+)—*						
	Ours (MLP・提案順)	1.68	5.66	22.1	33.6	34.8	50.4	128
	Ours (Trm・提案順)	2.41	11.8	16.8	$4.1 \cdot 10^3$	$1.3 \cdot 10^4$	$1.9 \cdot 10^3$	513
IMDb-JOB-light	NeuroCard (MADE・**正順**)	1.52	3.90	4.78	9.69	12.6	10.3	120
	NeuroCard (MADE・**逆順**)	1.71	5.22	14.7	33.2	33.7	10.9	346
	DeepDB	1.69	4.06	4.55	87.9	240	15.4	13.2
	Ours (MLP・提案順)	1.88	11.3	16.9	159	441	9.23	22.0
	Ours (Trm・提案順)	1.51	5.31	11.4	33.9	36.0	175	94.1

2.4.2 クエリオプティマイザに与える影響の評価

評価の指針とモチベーション：カーディナリティ推定の応用の 1 つとして，クエリオプティマイザへの応用を想定した評価を行う．事前に対象のクエリを結合ごとのサブセットに分割してカーディナリティ推定を行い，その推定値集合を

CEB [54] と pg_hint_plan [55] を介して PostgreSQL のオプティマイザに与えることで PostgreSQLPlanCost を取得し評価指標として用いる．これにより，カーディナリティ推定が結合順序最適化やインデックス選択を介してクエリオプティマイザに与える影響を評価する．PostgreSQL に加えて，真のカーディナリティ値 (True) による PostgreSQLPlanCost（Median・90 パーセンタイル・95 パーセンタイル・99 パーセンタイル・最大値）の比較を行う．True はオラクルであり，カーディナリティ推定によるクエリオプティマイザに対する性能向上の上限を表す．なお，利用するモデルは 2.4.1 項と共通であり，IMDb データセットを学習したものを用いる．

結果と考察：JOB-light ベンチマークでの結果を表 2.6 に示す．〜90 パーセンタイルでは PostgreSQL と提案手法で大きな差は見られなかった一方で，テールに近づくと提案手法による性能向上が確認された．このときの推定値が True による値に近いことから，提案手法は特に実行コストが高いと思われるクエリのプラン作成に貢献していることが分かる．一方比較的実行コストが低いと思われるクエリでは True との差が大きく，改善の余地があることも示唆されている．クエリオプティマイザへの応用を想定する場合，2.4.1 項のような対象クエリに対するカーディナリティ推定ではなく，対象クエリの結合ごとのサブセットとなるクエリ集合に対するカーディナリティ推定がより重要となる．これらのことを踏まえると，提案手法では今後，より単純なクエリに対するカーディナリティ推定性能の底上げを行うことでクエリオプティマイザに特化させるという最適化が考えられる．

表 2.6: 各手法の JOB-light ベンチマークでの PostgreSQLPlanCost

手法（モデル・属性順）	Median	90%th	95%th	99%th	Max
True	574444	1934955	2690204	12520363	28614678
PostgreSQL	635222	2324869	3490236	12847582	29351539
Ours (MLP・推論時ドメイン考慮順)	661491	2397843	2836455	12550896	28649355

2.4.3 推論時のパラメータの性能への影響の評価

評価の指針とモチベーション：提案手法を含むモンテカルロ法によるサンプリングを用いるカーディナリティ推定手法で，サンプルサイズと属性順による影響について評価を行う．2.4.1 項の総合的な評価において Naru のピーク性能となっていた

正順で推論を行う実装を比較対象とする（自己回帰モデルの場合，属性順を変化させたときにピーク性能となるモデルを容易に選択することはできない点に注意されたい）．カーディナリティ推定をクエリオプティマイザに応用すると 1 クエリでも複数回実行されることや応答性能が求められる近似カウントクエリ処理への応用を考えると，カーディナリティ推定の応答速度は重要である．ここでは提案手法がサンプルサイズを削減するというアプローチから推論性能を落とすことなく高速化できること示す．評価指標には Q-Error（Median・90 パーセンタイル・95 パーセンタイル・99 パーセンタイル・最大値）と平均応答時間を用いる．平均応答時間には GPU を利用した環境に加えて，一般的なデータベースシステムが稼働する環境を想定した CPU のみの環境での結果を報告する．なお，利用するモデルは 2.4.1 項と共通である．提案手法は単一のモデルで任意の順序を扱えるため，推論時の順序に依らず同じモデルを利用して評価を行う．

結果と考察：DMV でサンプルサイズを変化させたときの結果を表 2.7 に示す．いずれの手法でもサンプルサイズの変化により Q-Error と応答時間でトレードオフの関係が確認された．ただし応答時間は単純なトレードオフではなく，CPU 環境と GPU 環境で異なる傾向を示した．まず CPU 環境では，サンプルサイズと応答時間に明確な関連性が見られた．提案手法はサンプルサイズに対する応答時間の増加が少ないため，同程度の応答時間で比較するとサンプルサイズを大きくでき，結果として推論性能を向上させやすいことが分かる．一方 GPU 環境では，サンプルサイズが 100 以下では応答時間が横ばいになる現象が見受けられた．これはサンプルサイズが十分に小さいと GPU が一度に並列処理可能な範囲に収まり，サンプルサイズに比例しない定数コストがマジョリティとなるためと考えられる．GPU が一度に並列処理可能な範囲で最大のサンプルサイズを選ぶことで性能と速度を高めやすい一方，一般的なデータベースシステムでは用いられることが少ない GPU の導入が前提ということに注意されたい．Q-Error に注目すると，提案手法は Naru と比較してサンプルサイズが小さい場合でも Q-Error を抑えられることが確認された．特に 99 パーセンタイルや最大値においてその差は 10^1 から 10^2 倍程度と顕著である．CPU 環境を想定すると，提案手法は小さいサンプルサイズで推定を行うことで，性能と速度の両立が可能と分かる．

ここでサンプルサイズを 10 にしたときの Q-Error のテール（95 パーセンタイル・99 パーセンタイル・最大値）にあたる結果を表 2.8 に示す．順序による影響を

比較するため，表 2.7 で用いた手法に加えて提案手法の属性順を正順としたものを報告する．表 2.8 から，提案手法でも正順に固定して推論を行うと Naru と同様に Q-Error が悪化することが確認された．述語を適用した際のドメインが大きくなる順序（推論時ドメイン考慮順）で推論を行うことで，モデルに依らず正順での推論と比較して，より小さいサンプルサイズでも適切なカーディナリティ推定が可能と分かる．なお 2.3.2 項でも述べたように，推論時ドメイン考慮順は推論時に動的に決まるものであり，学習時に順序が固定される自己回帰モデルを利用した Naru のような手法では実現できないアプローチである．

表 2.7: DMV でサンプルサイズを変化させたときの Q-Error と平均応答時間 (ms)

手法（モデル・属性順・サンプルサイズ）	50%th	90%th	95%th	99%th	Max	応答時間 (GPU)	応答時間 (CPU)
Naru (MADE・**正順**・10)	1.06	1.32	1.70	23.1	$1.3 \cdot 10^5$	7.32	7.80
Naru (MADE・**正順**・100)	1.04	1.22	1.45	2.42	361	6.72	24.1
Naru (MADE・**正順**・1000)	1.04	1.19	1.32	2.00	8.00	10.3	131
Naru (MADE・**正順**・10000)	1.03	1.18	1.29	2.00	8.00	122	$1.4 \cdot 10^3$
Ours (MLP・**提案順**・10)	1.05	1.26	1.51	3.04	235	7.04	7.90
Ours (MLP・**提案順**・100)	1.05	1.24	1.44	2.50	194	7.34	17.4
Ours (MLP・**提案順**・1000)	1.05	1.24	1.43	2.50	49.0	7.08	62.1
Ours (MLP・**提案順**・10000)	1.04	1.24	1.43	2.50	49.0	43.0	428
Ours (Trm・**提案順**・10)	1.04	1.22	1.45	2.50	447	18.7	31.8
Ours (Trm・**提案順**・100)	1.04	1.20	1.36	2.27	9.80	18.8	69.1
Ours (Trm・**提案順**・1000)	1.03	1.20	1.36	2.23	12.3	52.2	354
Ours (Trm・**提案順**・10000)	1.03	1.20	1.36	2.23	12.3	532	$4.8 \cdot 10^3$

表 2.8: DMV でサンプルサイズが 10 のときの各手法の 95%th，99%th と Max Q-Error

手法（モデル・属性順）	95%th	99%th	Max
Naru (MADE・**正順**)	1.70	23.1	$1.3 \cdot 10^5$
Ours (MLP・**正順**)	1.82	47.6	$5.8 \cdot 10^3$
Ours (Trm・**正順**)	1.74	17.4	$1.1 \cdot 10^4$
Ours (MLP・提案順)	1.51	3.04	235
Ours (Trm・提案順)	1.45	2.50	447

2.5 関連研究

2.5.1 機械学習を用いたカーディナリティ推定

機械学習を用いたカーディナリティ推定技術は，提案手法と同じデータを学習するアプローチの他にワークロードを学習するアプローチが提案されている．本項ではそれぞれのアプローチの代表的な手法について述べる．

データを学習するアプローチの手法：提案手法を含むデータを学習するアプローチの手法では，データの分布を教師なしに学習したモデルでカーディナリティ推定を行う．Yang らは MADE [38] や Transformer [39] といった自己回帰モデルでデータの密度推定を行い，自己回帰の要領で得られる各属性の条件付き確率からカーディナリティ推定を行う手法を提案している [23, 24]．自己回帰する際にサンプリングを行うことで範囲条件を含むクエリのカーディナリティ推定に対応している．Yang らの報告と同時期に Hasan らも自己回帰モデルとして MADE を利用したデータの密度推定を行い，自己回帰の要領で得られる各属性の条件付き確率からカーディナリティ推定を行う手法を提案している [27]．Yang らの手法と同程度の性能であることが報告されている．これらの自己回帰モデルを利用する手法は学習・推論に利用する属性順によって性能が大きく変化することが併せて報告されているが [23]，その順序はデータセット作成者に依存するものとなっており，データセットによっては性能が不安定になってしまう．この問題を解決するために多数の順序を扱う自己回帰モデルの利用が今後の検討事項として挙げられているが，学習コストの面で実用化には課題が残る．具体的な多数の順序を扱う自己回帰モデルについては後述する．

Hilprecht らは Sum-Product Network [40] をデータベース向けに拡張した Relational Sum-Product Network によるカーディナリティ推定手法を提案している [25]．データの分布を行方向と列方向に分解してモデル化することで，推論時には行方向に分割されていたものは和を，列方向に分割されていたものは積を取ることでカーディナリティを算出する．Zhu らは Sum-Product Network を拡張した Factorize-Sum-Split-Product Network によるカーディナリティ推定手法を提案している [26]．全てを単変量分布に分解する Sum-Product Network と比較して，依存

関係が強い属性を適応的に多変量分布として扱うことで，依存関係に偏りのあるデータをコンパクトにモデル化できると報告されている．自己回帰モデルや DAE による提案手法と比較して，部分的に独立を仮定することで効率的な学習を実現している．一方でこの仮定により 99 パーセンタイルやワーストケースで性能が悪化する傾向がある．別の問題点として，Sum-Product Network は関数従属性を適切に捉えることができないという問題が報告されている [25]．これを回避するには，DeepDB では手動で関数従属性の情報を与える必要がある．一方で自己回帰モデルや DAE による提案手法であれば自然に捉えることができる．また，多量のメモリを利用した属性間依存関係の計算が必要となるため，大規模データセットでの実行が難しいことが表 2.5 の結果からも確認されている．

ワークロードを学習するアプローチの手法：ワークロードを学習するアプローチの手法では，クエリを何らかの方法で特徴量化したもの入力とし，カーディナリティをラベルとして学習したモデルでカーディナリティ推定を行う．Kipf らはワークロードをテーブル・結合・述語の 3 要素の組み合わせとして扱い，Multi-Set Convolutional Network [56] でカーディナリティをラベルとして学習することでカーディナリティ推定を行う手法を提案している [28, 29]．Woltmann らはいくつかの属性ごとにカーディナリティをラベルとして MLP モデルを学習することで，ワークロードで必要とされている部分にフォーカスしたカーディナリティ推定を行う手法を提案している [37]．各ワークロードに特化した小さなモデルにより，高い推論性能と応答速度を達成すると報告されている．Dutt らはワークロードの述語に含まれる下限値や上限値を特徴量とし，セレクティビティをラベルとして MLP モデルを学習することで，数値データの範囲条件を含むワークロードを得意とするセレクティビティ推定手法を提案している [30]．ワークロード由来の特徴量に加えて，属性間の独立を仮定するような従来のカーディナリティ推定手法から得られる値も特徴量とすることで安定性を高めている．これらの手法に共通する長所として，ワークロードが既知でクエリログが利用できる場合には効率的に学習・推論が行えることが挙げられる．しかしながら，提案手法のようなデータを学習するアプローチと比較して，学習のためのワークロードが未知あるいは学習に不十分な数の場合の性能低下が大きく，また，クエリログが利用できない場合はラベルとなるカーディナリティ値の取得に実際のクエリ処理が必要となり学習コストが高くなるという課題がある [28, 57]．

2.5.2 多数の順序を扱う自己回帰モデル

自己回帰モデルの順序による性能の不安定性を解決するために多数の順序を扱う自己回帰モデルが提案されている．カーディナリティ推定にも一般的な自己回帰モデルや DAE の代わりに用いることが考えられるため，それらとの相違点を中心に述べる．

Uria らは自己回帰モデルの 1 つである Neural Autoregressive Distribution Estimator (NADE) [58] を拡張して多数の順序を扱うことのできる Ensembles of NADEs (EoNADE) [41] を提案している．EoNADE は乱択アルゴリズムにより共通のパラメータ上で複数の順序を扱えるように NADE をベースとした学習を行う．これにより NADE より性能を向上させることに成功しているが，推論時のコストが順序の数に比例して増加する．また，ベクトル化できない演算を含むため処理コストが高いことが報告されており [38]，応答速度が求められるカーディナリティ推定には不向きと考えられる．Germain らは MADE [38] の提案の中で乱択アルゴリズムにより複数の順序を扱えるように学習を行う提案も行っている．この手法では任意の順序を扱うと過小適合を起こしてしまうため，予め少数の順序集合のみに限定[9] して取り扱う．そのため，DAE のようにカーディナリティ推論時にクエリを利用した順序選択するといったことはできない．Yang らは自然言語処理においてファインチューニングを前提とした言語モデルとして XLNet [42] を提案している．本来の順序を位置エンコーディングで保持しつつ単語列を並び替えながら学習することで，自己回帰モデルの欠点であった推論対象より後の単語情報を利用できないという課題を解決している．XLNet は言語モデルであるが，埋め込み層をテーブル形式データに対応させることで多数の順序を扱う自己回帰モデルとしてカーディナリティ推定への応用が考えられる．しかしながら 3.4 億パラメータという巨大なモデルを前提とした手法であるため，推論性能向上による全体の性能向上以上に学習・推論コストが大きくなることが考えられる．

9） 一例として，500 次元の隠れ層を持つ単一レイヤモデルで 784 次元のデータセットを学習させると，8 順序（$\ll 784!$）をピークに過小適合する傾向が報告されている．

2.6 おわりに

本章では複雑なスキーマを持つデータの高速な処理のための要素技術の 1 つとして，DAE を利用した高速で安定したカーディナリティ推定手法を提案した．DAE の密度推定としての性質でデータの分布を学習し，推論時に述語の条件を適用することでカーディナリティ推定を実現した．推論を DAE の特徴を活用した動的な順序で行うことで，サンプルサイズが小さい場合でも安定した推定手法となっている．複数のベンチマークで Q-Error を指標として評価したところ，既存の自己回帰モデルを用いる手法と比較してそのピーク性能と同程度の性能を安定して達成した上で 2〜3 倍の高速化が確認された．また，自己回帰モデル以外の機械学習による既存手法と比較しても Q-Error を抑えられることが確認された．より実践的な評価として，クエリオプティマイザへの応用を想定した実験を行ったところ，想定されるコストの大きいクエリを中心に PostgreSQL と比較して改善が確認された．ただし，一部のクエリでは学習したサンプルによって推定性能が低下するケースも確認された．原因の 1 つとしてサンプルに過学習していることが考えられる．正則化の導入やオプティマイザの調整などによりこの問題を解消することが今後の課題である．

本章では NeuroCard [24] と同じくユニバーサルリレーションを用いることで複数テーブルの結合に対応した．しかしながら，ユニバーサルリレーションによる結合対応は，スキーマが複雑になると性能が悪化することが示唆されている [59]．第 3 章では本章で提案したモデルをベースにこの問題を解決し，複雑なスキーマを持つデータの高速な処理に繋げる．

第 3 章　スケーラブルな結合カーディナリティ推定

3.1　はじめに

第 2 章では密度推定器と推論方法を工夫することで主に単一テーブルでのカーディナリティ推定を提案した．本章ではカーディナリティ推定のうち，特に結合を条件に含むカーディナリティ推定，すなわち結合カーディナリティ推定に焦点を当てる．既存手法では困難な，推論性能と複雑なリレーションシップを持つデータへの適用の両立を図る．

データベースシステムのクエリ処理性能において，クエリオプティマイザは最も重要な要素のうちの 1 つである．クエリオプティマイザはクエリ処理結果が変わらない範囲で，与えられたクエリをどのように処理すると効率的かを推定するタスクを行う．ここで，クエリオプティマイザの結果である「どのように処理するか」をまとめたものを実行プランという．実行コストの低い実行プランを見つけるために，クエリオプティマイザは内部的な処理の順番や方法を最適化する．処理の順番（結合順）の最適化では，後続の処理コストを下げるために結合処理の中間結果（中間テーブル）が小さくなるように行われる．この結合順序最適化の過程で，中間テーブルのサイズ推定として結合カーディナリティ推定が利用される．そのため，結合カーディナリティ推定が大きく外れると，必要以上に大きな中間テーブルを発生させてしまう実行プランが選択され，結果としてクエリ処理のパフォーマンス低下につながる．近年ではデータベースシステムに様々なデータが蓄積され，処理が行われることから，特に次の 3 つの観点に対応することが重要と言える．1 つ目は実世界データには偏りが存在するという点である．偏りの大きなデータの場合，結合順によって中間テーブルのサイズが上下しやすくなる．そのため結合順序最適化を成功させるには，結合カーディナリティ推定の性能が重要となる [13]．2 つ目はデータの量が増加しているという点である．一度に全データは扱えないということを前提としなければならない [11]．3 つ目は複雑なリレーションシップを持

つデータが増加しているという点である．スキーマに含まれるテーブル数，つまりスキーマサイズが大きい環境の考慮が必要となる [9, 10]．スキーマサイズが大きくなると結合順の候補が増え問題も複雑化するため，結合カーディナリティ推定では速度と性能の両方がより重要となる．

既存のデータベースシステムでは，結合カーディナリティ推定にはヒストグラムを中心とした統計情報が用いられている [19, 60]．ヒストグラムは属性ごとに作られ，推定時にはクエリ条件に含まれる属性のヒストグラムを利用して推定を行う．ただし，一般的にデータには偏りがあり属性間に相関があるにもかかわらず，属性ごとのヒストグラムを利用すると相関が無視され推定性能が低下する．この問題を解決するため，近年では機械学習を利用した手法が提案されている．機械学習を利用した結合カーディナリティ推定には大きく 2 つのアプローチがある．1 つはクエリを入力，カーディナリティを出力とする回帰問題として扱うアプローチ [28, 30] である．統計情報を用いる手法と比較して同程度以上の推定性能が報告されているが，データサイズやスキーマサイズが大きい環境では学習時のラベル取得コストが高く実用が困難である [59, 61]．もう 1 つは密度推定器を使うアプローチ [24, 25, 26] である．複数の属性に跨る形でデータの分布を学習し，クエリ条件に基づいた密度推定により結合カーディナリティを算出する．相関関係を捉えているため推定性能が高く，適切なサンプリングを行うことでデータサイズの増加にも対応できる．しかしながら大規模スキーマ環境の場合，1 つの密度推定器でデータベース全体を学習する手法の場合は推定性能が低下し，事前計算で複数に分割したサブスキーマごとに密度推定器を学習する手法の場合は事前計算のコストやメモリ消費量が多く，いずれもスキーマサイズに対してスケールしない [59]．以上のように，大規模スキーマに適した結合カーディナリティ推定手法は存在しない．

そこで本章では，データサイズとスキーマサイズに対してスケールする結合カーディナリティ推定手法を提案する．提案手法では，スキーマサイズが大きくなると，一般的に，総テーブル数に対してクエリの対象となるテーブルの割合が小さくなるという点に注目する．特にクエリオプティマイザで利用する場合，クエリの対象テーブルのサブセットごとに結合カーディナリティ推定を行うため，この傾向はより顕著となる．この傾向は，見方を変えるとスキーマ全体を一度に捉える必要性が低いということであるため，提案手法ではスキーマを分割して扱う．提案する分割方法はデータを参照せずスキーマのみを利用することで大規模データでも高速に

動作する．また，既存手法では分割したサブスキーマを独立として扱うため，複数のサブスキーマを横断するクエリに対する性能低下があるが，提案手法では分割にオーバーラップを持たせることでこの問題を緩和する．

本章の主な貢献は以下の通りである．

- 複数の密度推定器を利用した効率的な結合カーディナリティ推定手法を提案する．提案手法の主な特徴は以下の 3 つである．
  - **スキーマサイズに対してスケールする**：構造に基づいてスキーマを高速に分割し，分割されたスキーマごとに密度推定器を学習する．密度推定器ごとの対象テーブル数が少ないため，モデルのサイズを小さくしやすい．また，各密度推定器は独立しているため，全て並列に学習可能である．
  - **データサイズに対してスケールする**：密度推定器の学習に利用するデータには適切なサンプリングを行うことで小規模から大規模データまで柔軟に対応する．
  - **高い推定性能**：分割されたスキーマごとの密度推定器を持つため，クエリが対象とするテーブル集合にフィットしやすく推論性能が高くなる．特にクエリオプティマイザで利用する際に効果的な性質である．推論時には，分割間で結果の共有を行うことで，分割を横断した推定を行う場合でも推論性能を保つ．
- 実世界データセットに対してアクセス対象の傾向が異なるワークロードを利用して評価を行い，提案手法が効率的に結合カーディナリティを推定できること，クエリオプティマイザを介したクエリ処理の高速化に貢献することを確認した．

本章の構成は以下の通りである．3.2 節で本章に必要な定義を導入し，3.3 節で関連研究の紹介とその課題について述べる．3.4 節で提案手法を詳説し，3.5 節でその評価を行う．最後に 3.6 節で本章の内容をまとめる．

3.2 事前準備

本節では 2.2 節を振り返りながら結合カーディナリティ推定に必要となる諸定義の導入と問題の定式化を行う．

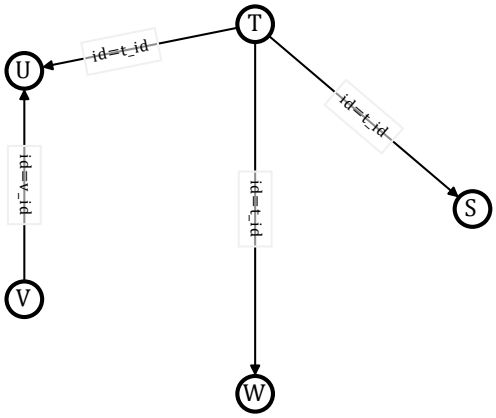
3.2.1 諸定義

表 2.1 に加えて本章で利用する主なシンボルの定義を表 3.1 に示した上で，データベーススキーマ，密度推定器とクエリをグラフ構造に変換する方法についても述べる．

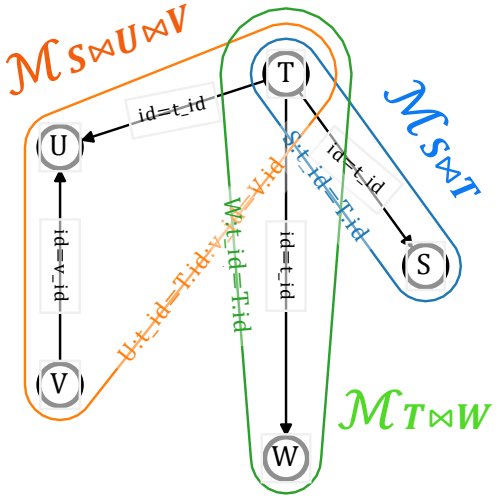
表 3.1: 本章で利用する主なシンボル

シンボル	定義・概要
$G[S]$	頂点集合 S によってグラフ G から誘導される部分グラフ．
$N_G^-[v], N_G^+[v], N_G^-(v), N_G^+(v)$	グラフ G 上の頂点 v の近傍の頂点集合．N^- は入近傍，N^+ は出近傍，$[v]$ は閉近傍，(v) は開近傍を表す．
H	$H := (V, \mathcal{E})$．頂点集合 V とハイパーエッジ集合 $\mathcal{E}$ から構成される多重ハイパーグラフ．本章ではスキーマグラフ上の部分スキーマの範囲を表す．
$\mathop{\mathbb{E}}_{t\sim T}[\cdot]$	分布 T から取り出されるサンプル t に基づく期待値．

結合カーディナリティ推定にまつわるグラフ：図 3.1 に本章で利用するデータベースのスキーマグラフ（グローバルスキーマグラフ）とクエリグラフの例を示す．スキーマグラフ G_{GLB} は図 3.1a では，$V_{GLB} = \{S, T, U, V, W\}$，$E_{GLB} = \{(T, S, (\mathrm{id}, \mathrm{t\_id})), (T, U, (\mathrm{id}, \mathrm{t\_id})), (T, W, (\mathrm{id}, \mathrm{t\_id})), (V, U, (\mathrm{id}, \mathrm{v\_id}))\}$ を用いて $G_{GLB} := (V_{GLB}, E_{GLB})$ と表す．図 3.1b と図 3.1d はグローバルスキーマグラフに対する密度推定器の対応と推論時の利用例である．別の観点として，スキーマグラフと密度推定器の対応は，各密度推定器がカバーしている頂点集合をハイパーエッジ $\mathcal{E} = \{(S, T), (T, U, V), (T, W)\}$ としたハイパーグラフ $H := (V_{GLB}, \mathcal{E})$ とみなすこともできる（図 3.1b）．なお，具体的な手法によって各密度推定器が対応する範囲や推論時の密度推定器の利用方法は異なる．クエリグラフ G_Q は図 3.1c を例とすると，$V_Q = \{S, T, U\}$，$E_Q = \{(t, s, (\mathrm{id}, \mathrm{t\_id})), (t, u, (\mathrm{id}, \mathrm{t\_id}))\}$ を用いて $G_Q := (V_Q, E_Q)$ と表せる．

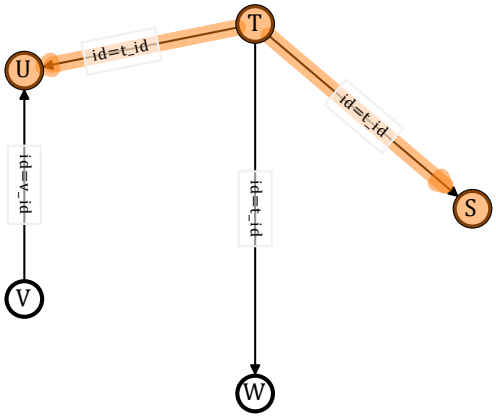


(a) スキーマグラフ G_{GLB}．頂点がテーブル，エッジがリレーションシップ（外部キー制約），エッジラベルが結合制約の内容を表す．

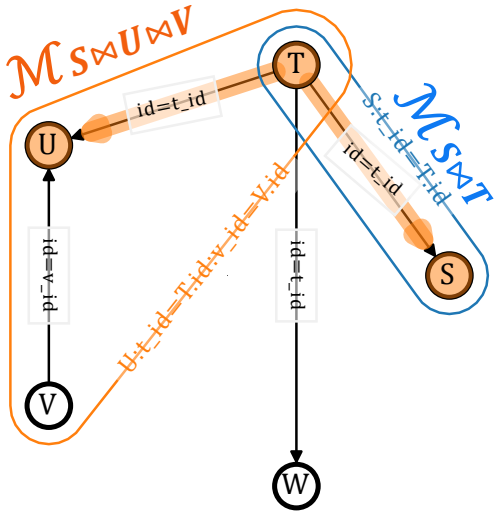


(b) スキーマグラフ G_{GLB} と密度推定器 $\mathcal{M}$ の対応例．ハイパーエッジごとに密度推定器が構築される．$\mathcal{M}$ のサブスクリプトが対象のテーブルを表す[a]．図は提案手法の例であり，手法によってハイパーグラフの形は異なる．

a　ここでの結合 $\bowtie$ は完全外部結合を示す．



(c) スキーマグラフ G_{GLB} 上のクエリグラフ G_Q．オレンジ色のグラフがクエリグラフである．



(d) クエリグラフ G_Q と密度推定器 $\mathcal{M}$ の対応例．図は提案手法の例であり，手法によって密度推定器の対象は異なる．クエリと無関係，もしくは冗長な密度推定器（$\mathcal{M}_{T\bowtie W}$）は選択されない．

図 3.1: 本章で利用するスキーマやクエリを表すグラフ構造の例

3.2.2 問題定義

ここでは結合カーディナリティの定式化を行う．はじめに単一テーブルのカーディナリティを再確認してから複数テーブルの結合カーディナリティについて扱う．

単一テーブルのカーディナリティ：単一テーブルの場合，クエリ Q に基づくカーディナリティ $\mathscr{C}(Q)$ は，テーブル T のすべての属性 $\boldsymbol{A}$ の条件を満たす同時確率（セレクティビティ $\mathcal{S}(Q)$ とも呼ばれる）とテーブルのサイズ $|T|$ の積で表すことができる（式 3.1）．ここで，同時確率は確率の乗法定理により条件付き確率の積として表すこともできる．多くの密度推定器は全属性の同時確率ではなく属性ごとの条件付き確率を扱うことから，カーディナリティ推定では後者が用いられる．

$$\begin{aligned}\mathscr{C}(Q) &= |T| \cdot P_T(A_1 \in R_Q(A_1), \ldots, A_n \in R_Q(A_n)) \\ &= \begin{aligned}&|T| \cdot P_T(A_1 \in R_Q(A_1)) \\ &\quad \cdots P_T(A_n \in R_Q(A_n) \mid A_{n-1} \in R_Q(A_{n-1}), \ldots, A_1 \in R_Q(A_1)))\end{aligned}\end{aligned} \tag{3.1}$$

複数テーブルの結合カーディナリティ：J_{univ} をスキーマに含まれる全てのテーブルを完全外部結合したテーブルとする．また，$G_Q = G_{GLB}$，つまりクエリ Q がすべてのテーブルを対象とし結合を行うとする（$G_Q \subset G_{GLB}$ の場合については，3.3.2 項で説明する）．この場合，クエリ Q に基づく結合カーディナリティ $\mathscr{C}(Q)$ は，テーブル J_{univ} のすべての属性 $\boldsymbol{A}$ の条件を満たす同時確率（セレクティビティ $\mathcal{S}(Q)$ とも呼ばれる）とテーブルのサイズ $|J_{univ}|$ の積で表すことができる（式 3.2）．また，単一テーブルの場合と同様に，確率の乗法定理により条件付き確率の積として表すこともできる．

$$\begin{aligned}\mathscr{C}(Q) &= |J_{univ}| \cdot P_{J_{univ}}(A_1 \in R_Q(A_1), \ldots, A_n \in R_Q(A_n)) \\ &= \begin{aligned}&|J_{univ}| \cdot P_{J_{univ}}(A_1 \in R_Q(A_1), \ldots, A_n \in R_Q(A_n)) \\ &\quad \cdots P_{J_{univ}}(A_n \in R_Q(A_n) \mid A_{n-1} \in R_Q(A_{n-1}), \ldots, A_1 \in R_Q(A_1))\end{aligned}\end{aligned} \tag{3.2}$$

なお，提案手法は事前に定義された任意のリレーションシップを扱うことができるが，簡単のため，外部キー制約による一対多のリレーションシップに基づいた結合のみを取り扱う．また，結合カーディナリティ推定には内部結合と外部結合の区

別があるが，こちらも簡単のため，内部結合を前提とした説明のみを行う[1)]．

3.3 関連研究

本節では既存の結合カーディナリティ推定手法について述べる．既存の結合カーディナリティ推定手法には，大きく分けてヒストグラムやサンプリングといった統計情報やデータを直接利用する手法と機械学習を使う手法がある．以降ではアプローチ別に細分化した上で，それぞれの手法の概要と課題についてまとめる．

3.3.1 統計情報に基づく手法

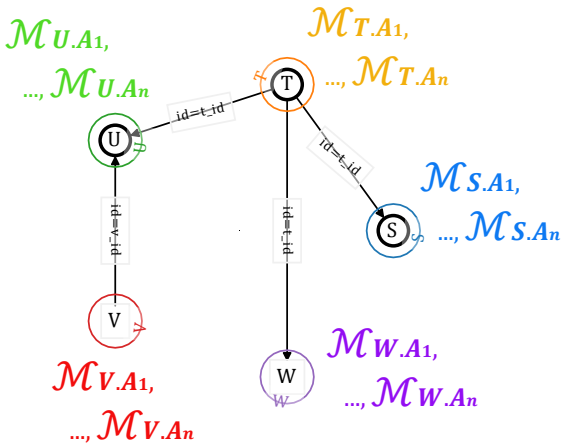
PostgreSQL を始めとした既存のデータベースシステムの多くでは結合カーディナリティ推定に統計情報が用いられる [19, 60]．これらではデータの統計情報としてヒストグラム[2)]を作成する（図 3.2）．属性の組み合わせ数は非常に多く組み合わせた場合はドメインの積を取る必要も発生してしまうため，現実的にはヒストグラムは属性ごとに構築される．推論時にはクエリ条件を包含するビンの割合を計算することでヒストグラムを密度推定器としてを利用した推定を行う[3)]．このとき，ヒストグラムが属性ごとに独立しているため，クエリ条件が複数属性に跨ると属性間の相関が無視された推定となる．

単一テーブルを対象とする場合，式 3.3 と表せる[4)]．後続手法の定式化のため密度推定器ごとの期待値 $\mathbb{E}_{t\sim\mathcal{M}_{A_i}}$ を利用して表記しているが，式 3.1 のように確率を用いた表記では式 3.4 となる．また複数テーブルを対象とする場合，テーブルを跨いだ情報も持たないため，結合はキーとなる属性の値が一様分布であるという仮定が利用される．このことは式 3.5 と表せる．式 3.2 のように確率を用いた表記では式 3.6 となる．

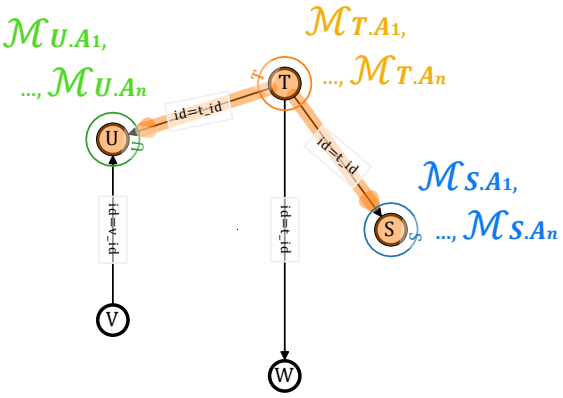
ここまで属性間の相関を無視することによる問題について述べたが，他にもビン内の分布の仮定も性能に影響がある．ビン内の分布は一様と仮定とするため，クエリ条件の範囲とビンが完全に一致しない限りは推定に線形近似が用いられる．提案

1） 提案手法の場合，推論時の条件変えることによって内部結合と外部結合の両方に対応できる
2） 実際には，頻出値リストなどの統計情報や経験に基づくマジックナンバーなどが併用されることがある．
3） このため属性ごとに独立なスキーマを仮定した密度推定に基づく手法ともいえる．
4） 簡単のため，ビン幅が十分小さく，ビン内一様分布の仮定の影響がないものとする．

手法では内部的に第 2 章で提案した深層学習による Denoising Autoencoder を利用することで，ヒストグラムに基づく手法のような属性間独立や一様分布の仮定による近似を行わず，高い推定性能を実現する．



(a) 統計情報に基づく手法のスキーマグラフと密度推定器の対応例．ここでの密度推定器はヒストグラムである．テーブルごとではなく属性ごとに構築されることに注意されたい．



(b) クエリグラフと密度推定器の対応例．クエリグラフに含まれるテーブルのうち，クエリの条件で指定されている属性の密度推定器のみが利用される．

図 3.2: 統計情報に基づく手法

$$\mathscr{C}(Q) \approx |T| \cdot \prod_{A_i \in \boldsymbol{A}} \mathop{\mathbb{E}}_{a \sim \mathscr{M}_{A_i}} \left[\mathbb{1}_{a \in R_Q(A_i)} \right] \tag{3.3}$$

$$\mathscr{C}(Q) \approx |T| \cdot \prod_{A_i \in \boldsymbol{A}} P(A_i \in R_Q(A_i)) \tag{3.4}$$

$$\mathscr{C}(Q) \approx \prod_{T \in V_Q} |T| \cdot \prod_{A_i \in \boldsymbol{A}} \mathop{\mathbb{E}}_{a \sim \mathscr{M}_{A_i}} \left[\mathbb{1}_{a \in R_Q(A_i)} \right] \cdot \prod_{(T_1, T_2, (A_1, A_2)) \in E_Q} \frac{1}{\mathrm{dom}(T_1.A_1)} \tag{3.5}$$

$$\mathscr{C}(Q) \approx \prod_{T \in V_Q} |T| \cdot \prod_{A_i \in \boldsymbol{A}} P(A_i \in R_Q(A_i)) \cdot \prod_{(T_1, T_2, (A_1, A_2)) \in E_Q} \frac{1}{\mathrm{dom}(T_1.A_1)} \tag{3.6}$$

3.3.2 グローバルスキーマでの密度推定に基づく手法

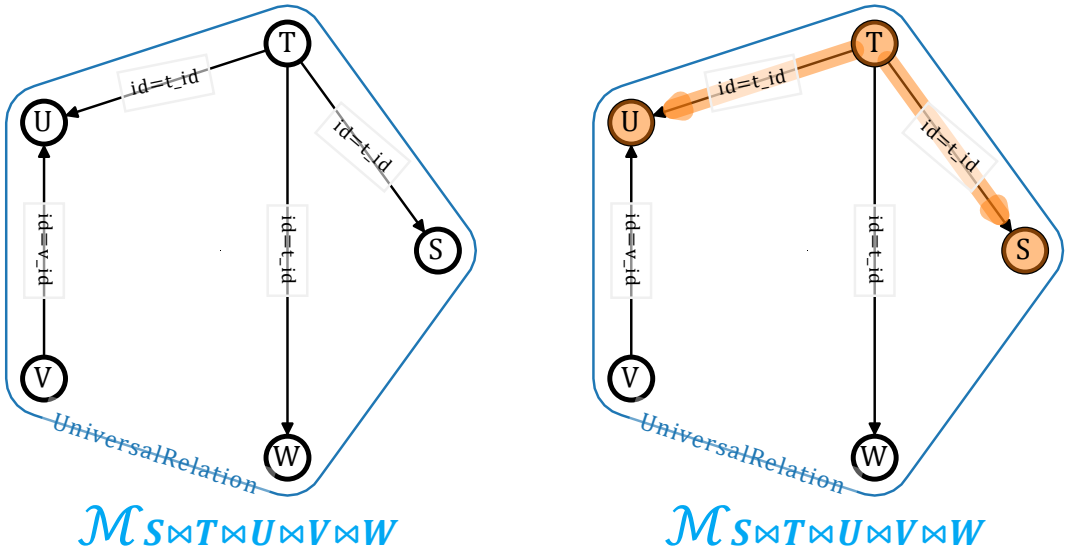
NeuroCard [24] や第 2 章で提案した手法では，グローバルスキーマに対応する 1 つの大きな密度推定器を利用して結合カーディナリティ推定を行う．定義されているリレーションシップに従って，グローバルスキーマ内に含まれる全てのテー

ブルを完全外部結合したテーブル（ユニバーサルリレーション）を仮定する．ここで，推論時に対象とするテーブル集合がグローバルスキーマに含まれるテーブル集合の真部分集合となるクエリに対応するため，2 つの仮想的な属性を追加する．1 つは結合タプルに該当テーブルが含まれているかどうかの真偽値（テーブルマーカー），もう 1 つは各リレーションシップにおける基数（Fanout）である．これらの仮想的な属性を含めた形でユニバーサルリレーションの密度を推定できるように学習する．例えば NeuroCard であれば自己回帰モデル，第 2 章では Denoising Autoencoder が使われる．データサイズが大きい場合は，すべての完全結合されたタプルの代わりに，独立同分布で一様となる結合サンプルを作成して学習データに用いることができる．実際には，例えば IBJS [62] はいずれの性質も持たず，Wander Join [63] は一様ではないため，Exact Weight Algorithm [43] などが用いられる．推論時は与えられたクエリ条件に基づいて，各属性の条件付き確率を推論する．加えて，テーブルマーカーと Fanout に関する推論を行う．1 つはクエリグラフに含まれるテーブルがタプルに存在という制約を反映するため，テーブルマーカーが真であるという条件を追加する．もう 1 つはクエリグラフに含まれないテーブルによる影響を打ち消すため，クエリグラフに含まれないテーブルに関する Fanout を推定しダウンスケールに利用する．

以上の推論手法をまとめると，式 3.7 を利用して式 3.8 と表せる．単一テーブルでのカーディナリティ推定と同様に全体のサイズ（ユニバーサルリレーションのサイズ）$|J_{univ}|$ とその同時確率 $\mathop{\mathbb{E}}_{t\sim\mathscr{M}_{J_{univ}}}\left[\prod_{A\in\boldsymbol{A}}\mathbb{1}_{t.A\in R_Q(A)}\right]$ の積に加えて，クエリグラフに含まれるテーブルが存在する確率 $\prod_{T\in V_Q} t.N_T$ とクエリグラフに含まれていない結合 $join$ に対応する Fanout によるダウンスケーリング $\prod_{join\notin E_Q} t.F_{join}$ が同時に考慮されたものとなっている．これは属性間独立の仮定などの近似がない厳密なものとなっている．

このアプローチの課題として，スキーマサイズに応じて密度推定器が扱う範囲が巨大化してしまい，捉えるべきデータ分布の複雑化による推定性能低下や多量の計算機資源要求といったことがあげられる．また，ユニバーサルリレーションを基準に不要なテーブルの影響を減らしながら推定するという減法的な性質から，スキーマグラフとクエリグラフの差が大きいほど性能が低下することが知られている [59]．特にクエリオプティマイザで結合カーディナリティ推定を利用する場合，

結合順最適化のためにクエリグラフのサブグラフごとに推定を行うため，この傾向が顕著となる．提案手法では並列に学習可能な小さい密度推定器を複数利用することで学習コストを低くし，さらにクエリオプティマイザなどの実応用上で高い性能を発揮可能とする．



(a) スキーマグラフとユニバーサルリレーションの密度推定器との対応例．ユニバーサルリレーションはグローバルスキーマに含まれる全テーブルを完全外部結合したものである．

(b) クエリグラフと密度推定器の対応例．常に単一の密度推定器が利用され，クエリグラフに含まれないテーブルのダウンスケールが必要である．

図 3.3: グローバルスキーマでの密度推定に基づく手法

$$\mathbb{I}(\boldsymbol{A}, Q, t) = \prod_{A \in \boldsymbol{A}} \mathbb{1}_{t.A \in R_Q(A)} \cdot \prod_{T \in V_Q} t.N_T \tag{3.7}$$

$$\mathscr{C}(Q) = |J_{univ}| \cdot \mathop{\mathbb{E}}_{t \sim \mathscr{M}_{J_{univ}}} \left[\frac{\mathbb{I}(\boldsymbol{A}, Q, t)}{\prod_{join \notin E_Q} t.F_{join}} \right] \tag{3.8}$$

3.3.3 相関ベース部分スキーマごとの密度推定に基づく手法

DeepDB [25] ではテーブル間の相関ベースで分割されたスキーマごとに密度推定器を構築して結合カーディナリティ推定を行う．この手法では，はじめに，スキーマグラフに定義された辺（リレーションシップ）に基づいてテーブル間（属性間）

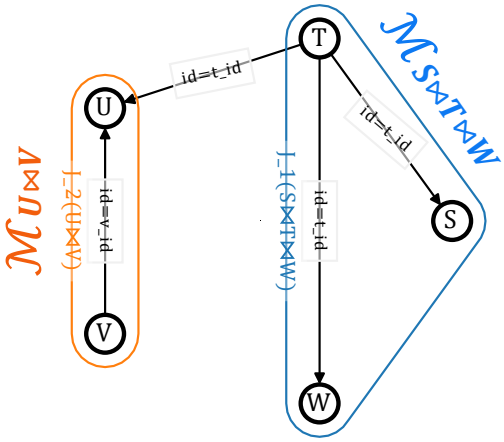
の相関を計算する．DeepDB では Randomized Dependence Coefficients (RDCs) [64] が利用される．算出した RDC と予め人手で設定した閾値を比較し，閾値を超過したら相関がある，そうでなければ相関がない独立したものとして扱う．相関があると判断したテーブル集合はすべて完全外部結合をとったテーブル（相関ベース部分結合テーブル）として扱う．なお，閾値を限りなく低くすると，やがて相関ベース部分結合テーブルは 3.3.2 項で説明したユニバーサルリレーションに，逆に限りなく高くすると，やがてすべて独立したテーブルになる．そのため，相関ベースの分割の優位性を引き出すには，適度に分割される閾値を設定する必要がある．そして，相関ベース部分結合テーブルに対応するように密度推定器を学習する．DeepDB では密度推定器として Sum-Product Network [40] が用いられる．このとき，3.3.2 項と同様にテーブルマーカーと Fanout も学習する．推論フェーズでは与えられたクエリ条件に基づいて，密度推定器を辿るようにして推論を行う．クエリ条件に基づく確率以外にも，3.3.2 項と同じようにクエリグラフに含まれるテーブルのマーカー条件やクエリグラフに含まれないテーブルの Fanout も推定する．密度推定器を跨ぐ際には，先行するテーブルの条件を元に，後続のテーブルとの結合によるカーディナリティ増加を Fanout 推定によるアップスケーリングでエミュレートする．ただし，後続の密度推定器の推定は独立したものとなる．このとき，ユニバーサルリレーションを利用した手法の減法的な性質と比較して，加法的な性質を示す．小さいクエリグラフは少数の密度推定器で推定するため分割の悪影響を余り受けないが，逆にクエリグラフが大きくなるとより多くの密度推定器が推論に必要となり，アップスケーリングや独立を仮定する箇所が増えて性能が低下しやすいことが知られている [59]．

このアプローチにおけるスキーマの分割は，分割間に重複はなくかつすべてのテーブルをカバーしている（$((J_1, J_2)^{\forall} \in \boldsymbol{J}^2 \wedge (V_{J_1} \cap V_{J_2}) = \emptyset) \wedge (\bigcup_{J \in \boldsymbol{J}} V_J = V_{GLB})$）．クエリ Q に必要となる相関ベース部分結合テーブル集合を $\boldsymbol{J}_Q = \{J \in \boldsymbol{J} \mid V_J \cap V_Q \neq \emptyset\}$ とし，$\boldsymbol{J}_Q$ から最初に取り出される任意のテーブルを J_0 とすると，結合カーディナリティ推定は式 3.7 を利用して式 3.9 と表せる．$|J_0|$ をカーディナリティの初期値とし，相関ベース部分結合テーブル J ごとにクエリの述語の条件 $\prod_{A \in \boldsymbol{A}} \mathbb{1}_{t.A \in R_Q(A)}$ とテーブルがクエリグラフに含まれている条件 $\prod_{T \in V_Q} t.N_T$ と後続のテーブルを結合するによるアップスケーリング $\prod_{(T_1,T_2,(A_1,A_2)) \in E_Q \wedge T_1 \in V_J \wedge T_2 \notin V_J} t.F_{(T_1,T_2,(A_1,A_2))}$ と未使用テーブルの影響を打ち消すダウンスケーリング $\prod_{join \notin E_Q} t.F_{join}$ の期待値となっ

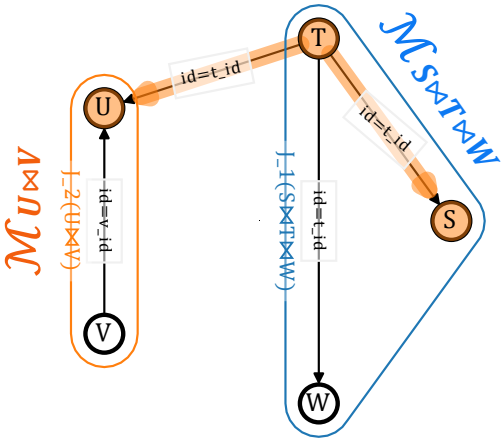
ている．相関ベース部分結合テーブル J ごとに期待値 $\mathbb{E}_{t\sim\mathcal{M}_J}$ が分かれていることは密度推定の独立を意味する．

課題として，RDC は n 行 r 列のテーブルを想定すると，時間計算量が $o(rn)$，空間計算量が $o(rn^2)$ と非常に大きい点があげられる．相関がないと判断されると独立として扱われてエラーの原因となるため，簡略化も難しい．

提案手法は DeepDB と同じようにスキーマを分割するというアプローチで巨大なユニバーサルリレーションを回避する．DeepDB と異なる点として，スキーマ分割のためにデータではなくスキーマグラフの構造のみを参照する，人手による閾値を使わないといった点があげられる．これにより，スキーマ分割が非常に低コストかつ人手による調整が不要となり，スキーマサイズに対してスケールしやすい．さらに推論時に密度推定器間で推論結果を共有できるようにサブスキーマにオーバーラップを持たせる．密度推定器間を独立として扱う場合と比較して，密度推定器間を横断した相関を考慮することで性能向上を図る．



(a) スキーマグラフと相関ベース部分結合テーブルごとの密度推定器との対応例．事前計算で T と U に相関がないと判定されたと仮定したケースのものである．相関ベース部分結合テーブルは相関ベース部分スキーマに含まれる全テーブルを完全外部結合したものである．



(b) クエリグラフと密度推定器の対応例．クエリグラフに含まれ，かつ冗長でない相関ベース部分結合テーブルの密度推定器が用いられる．密度推定器を跨いだ推定となる場合，独立した推定となる．

図 3.4: 相関ベース部分スキーマごとの密度推定に基づく手法

$$\mathscr{C}(Q) \approx |J_0| \cdot \prod_{J \in \boldsymbol{J}_Q} \mathop{\mathbb{E}}_{t \sim \mathscr{M}_J} \left[\frac{\mathbb{I}(\boldsymbol{A}, Q, t) \cdot \prod_{(T_1,T_2,(A_1,A_2)) \in E_Q \wedge T_1 \in V_J \wedge T_2 \notin V_J} t.F_{(T_1,T_2,(A_1,A_2))}}{\prod_{join \notin E_Q \wedge join \in E_J} t.F_{join}} \right] \tag{3.9}$$

3.3.4 回帰問題として扱う手法

MSCN [28, 29] に代表される手法では，結合カーディナリティ推定を回帰問題として扱う．クエリを入力とし，カーディナリティを出力とするようにモデル化が行われる．クエリを入力する際の埋め込みを工夫することにより，結合も扱うことができる．ただし，学習データのラベルとして，実際にクエリ処理をしてカーディナリティを取得しなければならないというジレンマが存在する．このため，モデルの推定性能を上げるために学習データを増やすことが困難という実用上の課題が残る．

3.3.5 関連研究と提案手法の位置づけ

最後に各手法の位置づけを表 3.3 に示す．推定性能，学習コスト，推定コストは主に中規模程度の環境（〜10 テーブル，〜1M レコード）を想定した簡易的な比較である．スケーラビリティとは，データサイズやスキーマサイズを大きくしたときの，学習コスト，推定コストや推定性能の悪化しにくさや機能不全への陥りにくさを表す．また，近似等は実行コストなどを下げるために元データの分布と比較して近似を行っている箇所を示す．

ここからは各手法についてまとめる．まず 3.3.1 項で示した統計情報に基づく手法は，属性間が独立であるという仮定が置かれており，中規模程度の設定の時点でも性能が低下しやすい．ただし，ここで必要となる統計情報の収集コストは機械学習を利用する手法と比較して非常に低く，学習コストに相当するコストは低い．また，構造が簡単なため推論コストも低い．機械学習による密度推定を利用した手法のうち，3.3.2 項で示したグローバルスキーマに密度推定器を対応付ける手法は，独立などの仮定を置く必要がないため性能が高い．しかしながらスキーマサイズが大きくなると，1 つの密度推定器で正確な分布を捉える難易度の上昇，必要な計算機資源の増加といった問題の他に，グローバルスキーマグラフと比較してクエリグラフが小さいときに性能が低下するという課題がある．一方 3.3.1 項で示した相関

ベーススキーマごとの密度推定器を利用する手法では，各密度推定器がこれらの問題は発生しづらい．しかしながら，相関ベーススキーマの決定は計算コスト高くメモリ消費量も多いため，複雑なデータセットには適用が難しい．機械学習を用いた別のアプローチとして，回帰問題として扱う手法では，クエリを適切に埋め込むことで複雑なデータセットでも比較的高い推定性能を達成している．しかしながら，実際にクエリ処理をしてラベルを取得する必要があるため学習コストが高く，これはスキーマサイズが大きく，対応するクエリが複雑になるほど顕著になる．

これらの既存手法と比較して提案手法では，軽量なスキーマ分割と並列に学習可能な小さな密度推定器により学習完了までのコストを抑える．さらに推論時は，必要最低限の密度推定器を組み合わせた推定により，スキーマサイズが大きな環境でも高い推定性能を実現する．

表 3.2: 表 3.3 の比較結果を示す記号の凡例．相対的なものであり，減点方式である（例として統計情報に基づく手法の推定性能は，単一属性に対する条件のみであれば高いことが期待されるが，複数属性に跨ると他の手法と比較して低下するため × として扱っている）．括弧付きのものは，本来該当しない項目だが相当する概念で評価したものである．

記号	意味
◎	高い性能で動作
◯	比較的高い性能で動作
△	動作するが制約が存在
×	低い性能で動作または動作せず
-	非該当

表 3.3: 結合カーディナリティ手法の比較．記号の意味は表 3.2 に示す．

アプローチ 手法，実装例	推定性能	学習コスト	推定コスト	スケーラビリティ	近似等
閉入近傍スキーマごとの密度推定 提案手法	◎	○	○	◎	スキーマ間近似
統計情報（ヒストグラム） PostgreSQL [19]，MariaDB [60]	×	(◎)	◎	×	属性間独立，ビン内一様
グローバルスキーマでの密度推定 NeuroCard [24]，第 2 章	◎	△	△	×	◎（仮定なし）
相関ベーススキーマごとの密度推定 DeepDB [25]	◎	△	◎	×	スキーマ間独立
回帰問題 MSCN [28]	○	×	◎	△	-

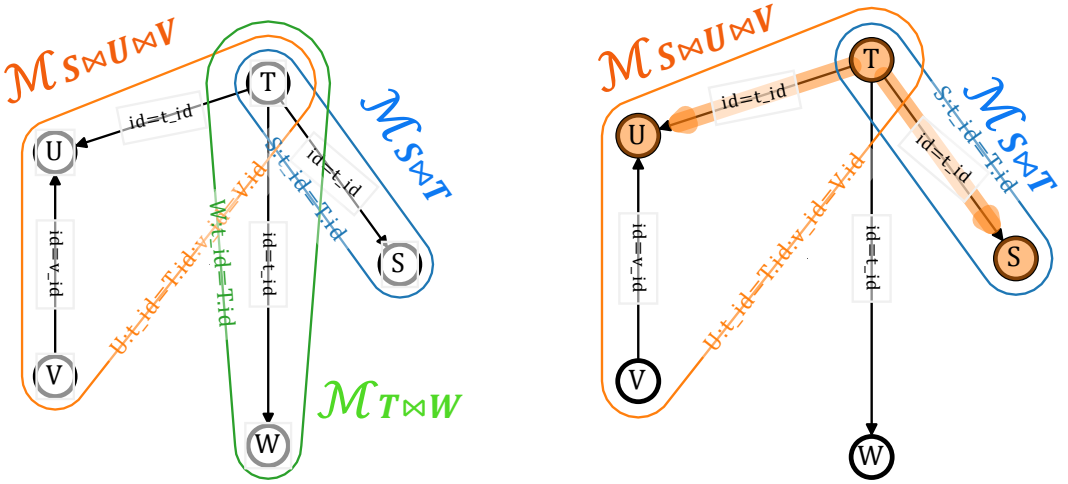
3.4 提案手法

本章では，スキーマグラフの構造に基づいた分割ごとの密度推定器を利用したカーディナリティ推定手法を提案する．まず提案手法による解決アプローチの概要を紹介し，3.4.1 項以降で提案手法の詳細を述べる．

3.3.5 項でまとめたように，結合カーディナリティ推定では，スキーマサイズを大きくしたときの効率的な実行や高い推論性能を出すことができていない．そこで提案手法では，図 3.5 のように，複数の小さい密度推定器を利用するというアプローチでこれらの課題の解決を図る．1 つ 1 つの密度推定器がカバーする範囲を小さくすることで，パラメータ数の少ないコンパクトなモデルでよくなり，省リソースで高速な動作が可能となる．また各密度推定器を独立したものとすることで，並列な学習も可能にする．さらにクエリオプティマイザなど実用上要求されやすい小さなクエリのクエリグラフに密度推定器の範囲がフィットしやすく，推定性能の向上が期待できる．スキーマ分割の際は，スキーマグラフの構造のみを利用することで，DeepDB で発生していたような分割のための追加コストを回避する．推論時は密度推定器間に共通するテーブルを介することで，密度推定器間の相関を考慮した推論を実現する．

提案手法の主な流れは前処理フェーズ，学習フェーズ，推論フェーズでそれぞれ以下の通りである．

1. 前処理
   (a) スキーマグラフの構造に基づいてスキーマを分割する．
2. 学習
   (a) スキーマごとに完全外部結合テーブルを作成する．
   (b) 完全外部結合テーブルごとにサンプルを利用して密度推定器を学習する．
3. 推論
   (a) クエリに必要なスキーマの密度推定器を選択する．
   (b) 始点となる結合テーブルのサイズを推定カーディナリティの初期値とし，密度推定器で述語を満たす確率を推論しカーディナリティを更新する．
   (c) 密度推定器を跨ぐ際には，結合した際のアップスケーリングとして Fanout を推論してカーディナリティを更新する．さらに共通するテーブルのサンプルを後続の推論条件に利用する．
   (d) クエリに必要なスキーマをすべて辿ると，クエリに基づくカーディナリティ推定値が得られる．



(a) スキーマグラフと閉入近傍結合テーブルごとの密度推定器との対応例．閉入近傍結合テーブルは，各頂点の閉入近傍となるテーブルを完全外部結合したものである．

(b) クエリグラフと密度推定器の対応例．クエリグラフに含まれかつ冗長でない閉入近傍結合テーブルの密度推定器が用いられる．すべての密度推定器はオーバーラップを持つ．

図 3.5: 閉近傍スキーマごとの密度推定に基づく手法

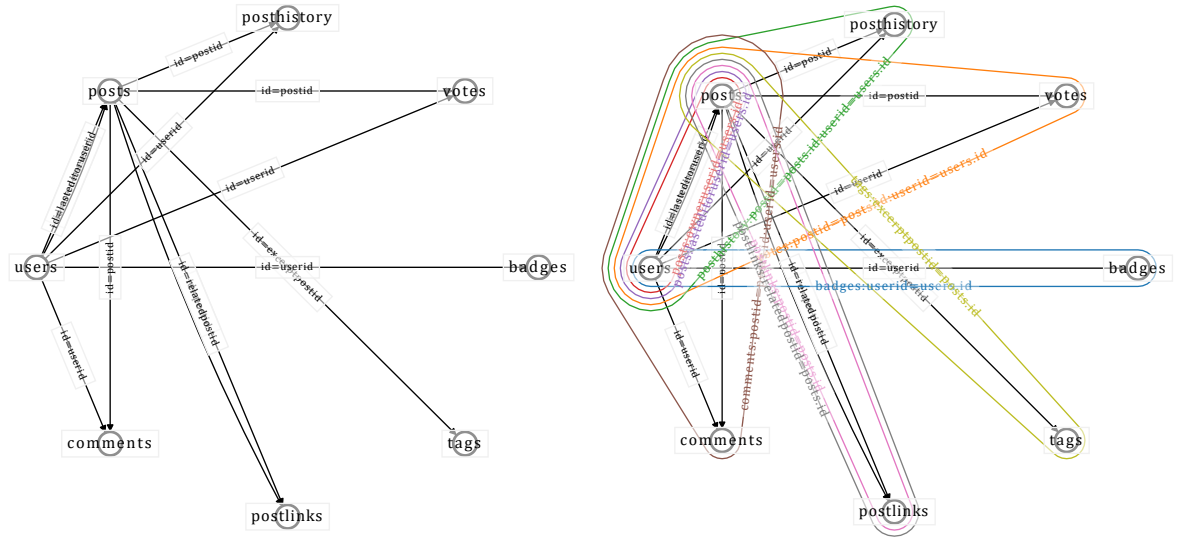
3.4.1　閉入近傍スキーマへの分割

ここでは提案手法で利用する各密度推定器の推定対象となるスキーマを決定する手順について述べる．まずは多重辺を含まない，つまりグローバルスキーマグラフ G が有向単純グラフである場合で説明する．提案手法では，スキーマグラフの各頂点の閉入近傍ごとにスキーマの分割を行う．例として図 3.1a のスキーマグラフを対象にすると，図 3.5a で示す分割になる．各頂点を見たとき，入次数が 1 以上の頂点は S, U, W の 3 つであるため，S, U, W を中心とした 3 つの閉入近傍スキーマに分割されている．

得られる各閉入近傍スキーマの性質について述べる．3.2.1 項で記したように，スキーマグラフの有向辺は一対多のリレーションシップを表している．そのため，閉入近傍となる頂点集合 $N_G^-[v]$ から誘導される部分グラフ $G[N_G^-[v]]$ は，すべてテーブル v を中心とした多対一の関係のみをもつスキーマとなっている．これにより，閉入近傍で完全外部結合したテーブルは，常に中心のテーブルはスケールされず，近傍のテーブルのみがスケールされたものとなる．結果として，推論時にダウンスケールが不要になり，推論性能向上に繋がる（ダウンスケールに関する詳細は 3.4.3 項で述べる）．次に閉入近傍スキーマがデータベースに表れる様々なリレーションシップに対応できることについて述べる．リレーションシップの種類は一対一，一対多，多対多の 3 種類に大別される．3 種のうち一対一は単に 1 つのテーブルを垂直分割したものと捉えれば，逆に垂直に結合すればスケーリングの必要もなく 1 つのテーブルとして扱えるため，特別の考慮は不要である次に一対多は，例えば出版社と書籍をモデリングした際に表れる（出版社は複数の書籍を発行しうるが，1 つの書籍が複数の出版社から発行されることはないものと考える）．この場合，書籍の数だけスケーリングが発生するため，これを捉える必要があるが，一対多のリレーションシップ（有向辺）は必ずいずれかの閉入近傍スキーマに含まれるため，対応する密度推定器で適切に分布を捉えられる．最後に多対多は，例えば著者と書籍をモデリングした際に表れる（著者は複数の書籍を執筆することがあり，書籍は複数の著者に執筆されることがある）．この場合，著者，書籍両方の数によってスケーリングが発生するため，これを捉える必要がある．ここで正規化を考えると，著者と執筆を一対多，執筆と書籍を多対一とすることで情報を損なうことなく 2 つの一対多で表すことができるとわかる．閉入近傍スキーマは中心テーブ

ルから多対一のリレーションシップをすべて含むため，結果として正規化した多対多のリレーションシップを 1 つの密度推定器で扱えることとなる．この性質は一般的なものではなく，例えば閉出近傍スキーマを利用すると複数のスキーマに跨ってしまう．これらから，閉入近傍スキーマを用いることで 2 テーブル間のいずれのリレーションシップも近似することなく扱えるといえる．

ここからは手法を一般化し，グローバルスキーマグラフ G が有向多重グラフである場合について述べる．多重辺を含む場合は，辺（結合制約）によってテーブル結合を行った際のスケーリング結果が異なるため，個別に取り扱う．具体的には，多重辺の組み合わせごとに閉入近傍スキーマを構築する．手順を Algorithm2 に示す．まず v への流入辺を近傍の頂点ごとに集める（4 - 6 行）．辺集合の集合となった $edges\_list$ を元に，直積集合を取得する（9 行）．最後に直積集合ごとにスキーマを構築する（11 行）．実際に多重辺を多数含むグローバルスキーマのデータセットに対して適用した例を図 3.6 に示す．例えば postlinks テーブルに注目すると posts テーブルからの多重辺が存在している（図 3.6a）．多重辺の組み合わせごとに閉入近傍スキーマが構築され，postlinks テーブルと posts テーブル間では結合制約の組み合わせが異なる 2 つのスキーマとなっている[5)]．



(a) 多重辺を持つスキーマグラフの例．

(b) 多重辺を含持つスキーマグラフ上に閉入近傍スキーマを構築する例．

図 3.6: 多重辺を持つスキーマグラフに対する閉入近傍スキーマの構築例．

5) 複数の近傍に対して多重辺がある場合は組み合わせの個数だけスキーマが構築される．

Algorithm 2 多重辺を考慮した閉入近傍ごとのスキーマ分割．多重辺を含む場合はその直積集合の元である辺の組ごとにスキーマが構築される．

Input: Global schema graph $G_{GLB} := (V_{GLB}, E_{GLB})$

Output: Set of closed in-neighborhood schema graphs $\boldsymbol{G}_{CIN}$

$\boldsymbol{G}_{CIN} \leftarrow \{\}$
for $v \in V_{GLB}$ **do**
  $edges\_list \leftarrow \{\}$
  for $u \in N^{-}_{G_{GLB}}(v)$ **do**
    $edges\_list.append(\{(x, y, c) \in E_{GLB} \mid x = v \wedge y = u\})$
  end for
  $N \leftarrow |edges\_list|$
  $V_{CIN_v} \leftarrow N^{-}_{G_{GLB}}[v]$
  for $E_{CIN_v} \in \{(e_1, \ldots, e_N) \mid e_1 \in edges\_list[1] \wedge \cdots \wedge e_N \in edges\_list[N]\}$ **do**
    ▷ Cartesian product
    $G_{CIN_v} := (V_{CIN_v}, E_{CIN_v})$
    $\boldsymbol{G}_{CIN}.append(G_{CIN_v})$
  end for
end for
return $\boldsymbol{G}_{CIN}$

3.4.2 閉入近傍スキーマごとの結合テーブル（閉入近傍結合テーブル）密度推定器の学習

ここからは得られた閉入近傍スキーマ群に基づいた密度推定器の学習について説明する．提案手法では，ユニバーサルリレーションによる手法（3.3.2 項）や相関ベース結合テーブルによる手法（3.3.3 項）と同様に，スキーマ内に含まれる部分的なテーブル集合に対する問い合わせへの対応や内部結合・外部結合を共通の推定器で扱うため，スキーマに含まれるすべてのテーブルを完全外部結合して扱う．なお，仮想的な追加の属性（テーブルマーカー・Fanout）も同様に扱うこととなるが，閉入近傍スキーマ内のテーブル間の Fanout ではなく，スキーマ外の隣接する

テーブルへの Fanout が必要となる．[6)]

密度推定器には，共通するテーブルに基づいた連携ができるものを利用する．連携できない場合，クエリグラフが複数の閉近傍スキーマを跨ぐようなケースでは，スキーマごとに独立した密度推定となり推定性能が低下する．例えば DeepDB で用いられている Sum-Product Network [40] は密度推定器の一種であるが，密度推定対象が閉じているため，提案手法には不適である．ここでは，第 2 章で安定して高い性能が確認されている Denoising Autoencoder を密度推定器として利用する．学習は 2.3.1 項で述べたように，全属性からランダムに選択した属性に対してノイズとしてマスクをしたものを入力とし，マスクされた属性の出力に対して交差エントロピーロスを計算して行う．これにより任意の属性を条件とした任意の属性の分布を推定可能な密度推定器となる．

学習データには閉入近傍スキーマで完全外部結合したテーブルを利用する．単純に結合テーブルに含まれる全てのタプルを用いることも考えられるが，データ件数が非常に多いケースは学習に時間がかかりすぎ，逆にデータ件数が非常に少ない場合は学習が進まないという問題がある．そこでデータ件数の影響を緩和するため，結合サンプリングを用いる．結合サンプルの分布を学習することになるため，結合サンプルは密度推定器の性能に直結する．本来の結合テーブルのデータから独立同分布かつ均一なサンプルを取得するため，NeuroCard でも利用されている Exact Weight Algorithm [43] を用いる．[7)]

3.4.3 複数の閉近傍結合テーブル密度推定器を利用した問い合わせ

複数の閉近傍結合テーブル密度推定器を利用した結合カーディナリティ推定方法について説明する．提案手法では複数の密度推定器を跨いだ推論を行うため，閉入近傍スキーマ集合をハイパーグラフ見なしたうえで，クエリグラフに基づいてハイパーエッジを辿りながら推論を行う．直感的には，始点となるテーブルを準備し，そこに対して述語による絞り込みと後続テーブルとの結合の繰り返しをエミュレートするイメージである．

6） より正確にいうと，入次数が 0 のテーブル単体のカーディナリティを推定する場合に限りスキーマ内の Fanout が必要となるが，本章では結合カーディナリティ推定を扱うため省略する．

7） データサイズに対して十分スケールすることができ，提案手法のスケーラビリティを損なうことはない．

まず閉入近傍スキーマ集合からのハイパーグラフ構築について述べる．Algorithm3 で示すように，各閉入近傍の頂点集合をハイパーエッジ $e \in \mathscr{E}$ とし，グローバルスキーマグラフに含まれる頂点集合 $V = V_{GLB}$ と合わせてハイパーグラフ $H := (V_{GLB}, \mathscr{E})$ を構築する．このとき，以下の系 1 が成り立つ．

系 1. G が連結であれば，$G[N_G^-[v]].V$ をハイパーエッジとしたハイパーグラフ H も連結である．

Algorithm 3 閉入近傍スキーマに含まれる頂点集合をハイパーエッジとしたハイパーグラフの構築

Input: Global schema graph $G_{GLB} := (V_{GLB}, E_{GLB})$, Set of closed in-neighborhood schema graphs $\boldsymbol{G}_{CIN}$

Output: Hypergraph of closed in-neighborhood schemas $H := (V, \mathscr{E})$

1: $V \leftarrow V_{GLB}$
2: $\mathscr{E} \leftarrow \{\}$
3: **for** $G \in \boldsymbol{G}_{CIN}$ **do**
4: $\quad \mathscr{E}.append(G.V)$
5: **end for**
6: $H := (V, \mathscr{E})$
7: **return** H

次に，構築した閉入近傍のハイパーグラフとクエリグラフに基づいた推論方法について述べる．推論の手順を Algorithm4 に示す．まずはじめに，クエリがすべてのテーブルにアクセスするとは限らないため，クエリグラフ G_Q に基づいて必要な閉入近傍スキーマへの絞り込み，つまり対応するサブハイパーグラフ H_Q の構築を行う（1 - 3 行）．得られたサブハイパーグラフ H_Q のハイパーエッジ $\mathscr{E}_Q$ を辿るための木を作成し（7 行），幅優先探索順でハイパーエッジを辿る（9 行）．ランダムにハイパーエッジ $\mathscr{E}_Q$ から始点 e_0 を選択し，その閉入近傍結合テーブルのサイズ $|J_{e_0}|$ をカーディナリティの初期値とする（4 行）．各ハイパーエッジでは，まず述語による絞り込み以外に推定が必要となる属性を並べる（10 - 13 行）．具体的には，後続のハイパーエッジ f と共通する頂点（テーブル）の属性 $\boldsymbol{A}_{common}$ と後続のハイパーエッジとの結合を考えたときに必要となる Fanout $\boldsymbol{F}$ である．これらと述語に基づく確率をハイパーエッジ e に対応する閉入近傍結合テーブル J_e の密度推

定器 $\mathscr{M}_{J_e}$ で推論する（14 行）．このとき先行するハイパーエッジがある場合は，共通するテーブルの属性のサンプル $common\_samples$ を入力に利用する．推論は 1 に入力サンプル $common\_samples$ の利用・共通テーブル属性 $\boldsymbol{A}_{common}$ の追加の推論・Fanout の指定 $\boldsymbol{F}$ を加えた Algorithm5 で行う．結果として得られる J_e 内の述語に基づく確率 $P_{J_e}(R_Q)$ と Fanout $fanouts$ をカーディナリティにかける（15 - 16 行）．この操作をすべてのハイパーエッジで行うことで，クエリ Q のカーディナリティ推定値 $\mathscr{C}(\hat{Q})$ が得られる（19 行）．

表 3.4: Algorithm4 で利用する関数の定義

関数名	定義
RANDOM($\boldsymbol{x}$)	$\boldsymbol{x}$ からランダムに 1 要素を選択する．
TOTREE(H, G)	有向グラフ G を覆うようにハイパーグラフ H のエッジ集合 $H.\mathscr{E}$ から木を構築する．系 1 から，連結である木が得られる．
BFS(T, s)	s を始点とし，幅優先探索順で木 T の辺を頂点とその次に来る頂点集合のタプルの形で列挙する．
ATTRIBUTESIN($\boldsymbol{v}$)	頂点（テーブル）集合 $\boldsymbol{v}$ の属性集合を平坦化した属性集合を取得する．
EDGETOFANOUTATTRIBUTE(u, v, c)	テーブル u から v へ制約条件 c の Fanout 属性を取得する．

図 3.5 を例とし始点 e_0 を $S \bowtie T$ とすると，カーディナリティの初期値は $|S \bowtie T|$ である．$\mathscr{M}_{S \bowtie T}$ で $S \bowtie T$ でのクエリ条件 Q を満たす確率と $S \bowtie T$ に U を結合したときの Fanout $F_{T.id=U.t_id}$ を推論した後，$\mathscr{M}_{S \bowtie T}$ から得られた T の属性のサンプルを入力として $\mathscr{M}_{T \bowtie U \bowtie V}$ で $T \bowtie U \bowtie V$ でのクエリ条件 Q を満たす確率を推論することとなる．

この推定手法は既存手法と比較すると以下の 2 点が特徴的である．1 つ目はハイパーエッジ間で共通する頂点（テーブル）の属性のサンプルを引き継いで入力とすることで，共通するテーブルに限られるが先行する属性の条件に即した推論を行っ

Algorithm 4 複数の閉入近傍結合テーブル密度推定器を利用した結合カーディナリティ推定

Input: Global schema graph $G_{GLB} := (V_{GLB}, V_{GLB})$, Hypergraph of closed in-neighborhood schemas $H := (V, \mathscr{E})$, Acyclic query $Q := (R_Q, G_Q)$, $G_Q \subseteq G_G$, G_Q is simple tree
Output: Estimated cardinality of Q $\hat{\mathscr{C}}(Q)$

$\mathscr{E}_Q \leftarrow \{e \in \mathscr{E} \mid e.V \setminus G_Q.V \neq \emptyset\}$
$V_Q \leftarrow \{v \in V \mid v \in G_Q.V\}$
$H_Q := (V_Q, \mathscr{E}_Q)$
$e_0 \leftarrow \textsc{Random}(\mathscr{E}_Q)$
$common\_samples \leftarrow \{\}$
$\hat{\mathscr{C}} \leftarrow |J_{e_0}|$ ▷ Number of rows of root table as initial cardinality
$T_{\mathscr{E}_Q} \leftarrow \textsc{ToTree}(H_Q, G_Q)$ ▷ Restructure hyperedges to tree w.r.t. tree G_Q
$successor\_list \leftarrow BFS(T_{\mathscr{E}_Q}, e_0)$ ▷ Arrange hyperedges in BFS order
for $e, \boldsymbol{f} \in successor\_list$ **do** ▷ e and $\boldsymbol{f}$ are current hyperedge and successors, respectively
  $\boldsymbol{A} \leftarrow \textsc{AttributesIn}(e.V)$
  $\boldsymbol{A}_{common} \leftarrow \bigcup_{f \in \boldsymbol{f}} (\boldsymbol{A} \cap \textsc{AttributesIn}(f.V))$
  $fanout\_edges \leftarrow \bigcup_{f \in \boldsymbol{f}} (\{(u, v, c) \in G_Q[f.V].E \mid u \in (e.V \cap f.V)\})$
  $\boldsymbol{F} \leftarrow \textsc{EdgeToFanoutAttribute}(fanout\_edges)$
  $\hat{P}_{J_e}(R_Q), \hat{fanouts}, \hat{samples} \leftarrow \textsc{Estimate\_N}(\mathscr{M}_{J_e}, R_Q, \boldsymbol{A}, \boldsymbol{F}, \boldsymbol{A}_{common}, common\_samples)$
  $\hat{\mathscr{C}} \leftarrow \hat{\mathscr{C}} \times \hat{P}_{J_e}(R_Q)$
  $\hat{\mathscr{C}} \leftarrow \hat{\mathscr{C}} \times \prod_{\hat{fanout} \in \hat{fanouts}} \hat{fanout}$
  $common\_samples.add(\hat{samples})$
end for
return $\hat{\mathscr{C}}$ as $\hat{\mathscr{C}}(Q)$

ている点である．これは密度推定器として利用している Denoising Autoencoder が任意の条件で推論できるという性質持つことと系 1 から連結の定義より，常に可能である．これにより複数の密度推定器を跨いでも完全な独立を仮定することなく推論が可能となり，全体としての推論性能向上が期待できる．2 つ目はサブハイパーグラフのハイパーエッジ $e \in \mathscr{E}_Q$ にクエリで必要のない頂点（テーブル）が含まれていても（$v^{\exists} \in e \wedge v \notin V_Q$）ダウンスケーリングが不要という点である．これは e が閉入近傍 $N_G^-[v]$ に基づいており，不要になるテーブル（v が不要になることはなく $u \in N_G^-(v)$ のいずれかである[8]）が必ず v から見て多対一になっているからである．こちらも全体としての推論性能向上に寄与する．

8） $\because |V_Q|$ が 1 のときを除き，v が不要になる場合は e 自体が冗長で不要となるため

表 3.5: Algorithm5 で利用する関数の定義

関数名	定義
DrawSample($dist_A$)	属性 A の分布 $dist_A$ から，分布を重みとしたサンプリングを行う．
Encode(a)	属性 A の要素 a を埋め込む．One-hot ベクトルや Entity Embeddings [47] などが利用できる．
SortBy($\boldsymbol{A}, key\_func$)	属性集合 $\boldsymbol{A}$ を key_func に基づいて昇順に並べ替える．
Mean($\boldsymbol{a}$)	数値集合 $\boldsymbol{a}$ の平均値を計算する．

最後に，提案手法による結合カーディナリティ推定手法の定式化を行う．クエリ Q の結合カーディナリティ推定に必要十分な閉入近傍結合テーブル集合を $\boldsymbol{J}_Q$ とし，最初に取り出す要素を J_0 とすると，式 3.7 を利用して式 3.10 で表せる．ここで $\mathbb{E}_{t_j \sim \mathscr{M}_{J_j} | J_j.\boldsymbol{A}=t_{<j}.\boldsymbol{A}}$ で表される期待値は，J_j に先行する $J_{<j}$ のサンプル $t_{<j}$ のうち，共通するテーブル $T \in (V_{J_j} \cap V_{J_{<j}})$ の属性集合 $\boldsymbol{A}$ を推論の条件として利用していることを示す．ユニバーサルリレーションを利用した手法（式 3.8）と比較すると期待値ごとに近似が存在し厳密性で劣るものの，相関ベース結合テーブルを利用した手法（式 3.9）と比較すると期待値ごとの独立がなくなっている点で近似の精度が高くなりやすい傾向にある．また，式 3.8，式 3.9 いずれとも異なり，Fanout によるダウンスケーリングが不要となっており，これも性能向上に寄与があると考えられる．

$$\mathscr{C}(Q) \approx |J_0| \cdot \prod_{J_j \in \boldsymbol{J}_Q} \mathop{\mathbb{E}}_{t_j \sim \mathscr{M}_{J_j} | J_j.\boldsymbol{A}=t_{<j}.\boldsymbol{A}} \Bigg[\mathbb{I}(\boldsymbol{A}, Q, t_j) \cdot \prod_{(T_1,T_2,(A_1,A_2)) \in E_Q \wedge T_1 \in V_{J_j} \wedge T_2 \notin V_{J_j}} t_j.F_{(T_1,T_2,(A_1,A_2))} \Bigg] \tag{3.10}$$

Algorithm 5 密度推定器を利用したセレクティビティと Fanout の推論とサンプルの取得

Input: Density Estimator $\mathscr{M}$, Predicate ranges R_Q, Whole attributes $\boldsymbol{A}$, Fanout attributes $\boldsymbol{F}$, Attributes to be additionally sampled $\boldsymbol{A}_{sample}$, Samples $samples$

Output: Estimated selectivity $\hat{\mathcal{S}}$, Fanouts $\hat{fanouts}$, Samples w/ some updates $\hat{samples}$

procedure ESTIMATE_N ▷ Sample size is N
  procedure ESTIMATE($\mathscr{M}, R_Q, \boldsymbol{A}_Q, \boldsymbol{F}, \boldsymbol{A}_{sample}, sample$)
    Initialize $inputs$ with $sample$
    $\hat{prob} \leftarrow 1.0$
    for $A \in \boldsymbol{A}_Q$ **do** ▷ Estimate probabilities of predicates
      $\hat{dist}_A \leftarrow \mathscr{M}(inputs)$ ▷ Forward
      $\hat{dist'}_A \leftarrow \{\hat{dist}_a * (a \in R_Q(A)) \mid a \in A\}$ ▷ Filter distribution by R_Q
      $\hat{prob} \leftarrow \hat{prob} * \sum_{a \in A} \hat{dist'}_a$
      $\hat{a} \leftarrow$ DRAWSAMPLE($\hat{dist'}_A$)
      $inputs[A] \leftarrow$ ENCODE($\hat{a}$)
    end for
    for $A \in \boldsymbol{F}$ **do** ▷ Estimate fanouts
      $\hat{dist}_A \leftarrow \mathscr{M}(inputs)$
      $\hat{a} \leftarrow$ DRAWSAMPLE($\hat{dist}_A$)
      $\hat{fanout}[A] \leftarrow \hat{a}$
      $\hat{sample}[A] \leftarrow \hat{a}$
      $inputs[A] \leftarrow$ ENCODE($\hat{a}$)
    end for
    for $A \in \boldsymbol{A}_{sample}$ **do** ▷ Draw samples for subsequent estimation
      $\hat{dist}_A \leftarrow \mathscr{M}(inputs)$
      $\hat{a} \leftarrow$ DRAWSAMPLE($\hat{dist}_A$)
      $\hat{sample}[A] \leftarrow \hat{a}$
      $inputs[A] \leftarrow$ ENCODE($\hat{a}$)
    end for
    return $\hat{prob}, \hat{fanout}, \hat{sample}$
  end procedure

  $\boldsymbol{A}_Q \leftarrow \{A \in \boldsymbol{A} \mid |R_Q(A)| < \mathrm{dom}(A)\}$ ▷ Filter attributes by predicates
  $\boldsymbol{A}_Q \leftarrow$ SORTBY($\boldsymbol{A}_Q, key : A \to |R_Q(A)|$) ▷ Sort attributes by domain size
  for $i \in \{1, \ldots, N\}$ **do** ▷ Batched in practice
    $\hat{probs}[i], \hat{fanouts}[i], \hat{samples}[i] \leftarrow$ ESTIMATE($\mathscr{M}, R_Q, \boldsymbol{A}_Q, \boldsymbol{F}, \boldsymbol{A}_{sample}, \hat{samples}$)
  end for
  $\hat{\mathcal{S}} \leftarrow$ MEAN($\hat{probs}$)
  return $\hat{\mathcal{S}}, \hat{fanouts}, \hat{samples}$
end procedure

3.5 評価実験

本節では結合カーディナリティ推定の評価を行う．まず 3.5.1 項で実験設定を一通り説明した後，3.5.2 項でカーディナリティ推定単独での評価を，3.5.3 項でカーディナリティ推定をクエリオプティマイザに利用した際の評価を行う．

3.5.1 実験設定

3.5.1.1 ベンチマーク

評価実験に用いるベンチマーク（データセット+ワークロード）について述べる．

JOB ベンチマーク：JOB ベンチマーク [13] は俳優，映像エンターテイメントやビデオゲーム等に関する実世界データセットである IMDb[9] データセットを利用する．最大 36M 行，最大 17 属性の 16 テーブルから構成される．ワークロードとしては JOB-light [28] と JOB-m [24] の 2 つを利用する．JOB-light は複数の等号条件と範囲条件を持ち，最大 5 テーブルの内部結合を行う 70 クエリから構成される．JOB-m は複数の等号条件，範囲条件に加えて IN 句や LIKE 演算子を持ち，最大 11 テーブルの内部結合を行う 113 クエリから構成される．

3.5.1.2 評価指標

推定性能の評価指標には，真のカーディナリティを $\mathscr{C}$，推定されたカーディナリティを $\hat{\mathscr{C}}$ として式 3.11 で表される Q-Error [13] を用いる．これは推定されたカーディナリティが真の値から何倍離れているかを表す無次元の値であり，常に 1 以上で小さい方が優れていることを示す．

$$\text{Q-Error} := \frac{\max(\hat{\mathscr{C}}, \mathscr{C})}{\min(\hat{\mathscr{C}}, \mathscr{C})} \tag{3.11}$$

さらにクエリ実行時間に相当する指標として P-Error [61] を用いる．この指標では，まず PostgreSQL のクエリオプティマイザを利用して，真のカーディナリティ $\mathscr{C}$ と推定されたカーディナリティ $\hat{\mathscr{C}}$ それぞれを利用した実行プランを作成する．

9） https://www.imdb.com

このとき，実行プラン作成が目的であるため，対象クエリ全体ではなく対象クエリの一部分に対するカーディナリティ推定が複数回利用されることとなる．そのため，カーディナリティ推定としては，複雑で大きなクエリよりもそのサブセットとなる小さなクエリの推定性能が影響力を持つ．次に，再度 PostgreSQL のクエリオプティマイザを利用して，作成した 2 つの実行プランの実行コストを推定する．最後に，得られた推定実行コストの比を P-Error とする．多くの場合 1 以上であり，小さい方が優れていることを示す．カーディナリティ $\mathscr{C}$ を利用したクエリオプティマイザによる実行プランを $P(\mathscr{C})$，実行プラン P の実行コスト推定を $PCEst(P)$ とすると，P-Error は式 3.12 と表せる．

$$\text{P-Error} := \frac{PCEst(P(\hat{\mathscr{C}}))}{PCEst(P(\mathscr{C}))} \tag{3.12}$$

なお，実行プラン作成以外のカーディナリティ推定には真のカーディナリティが利用される．この指標は，時間のかかる実際のクエリ実行を行うことなくクエリ実行時間の比較に相当する比較ができる点が特徴的である．クエリオプティマイザによっては，真のカーディナリティを利用した実行プランの推定コストが推定されたカーディナリティを利用したものを上回る（P-Error < 1）可能性がある．しかしながら，$PCEst(P(\mathscr{C}))$ は比較手法間で一貫したものであるため，比較する際の指標としては問題がないとされている．

実際には，事前に対象のクエリを結合ごとのサブセットに分割してカーディナリティ推定を行い，その推定値集合を CEB [54] と pg_hint_plan [55] を介して PostgreSQL のオプティマイザに与えることで実行プラン $P(\hat{\mathscr{C}})$ を取得する．そして取得した $P(\hat{\mathscr{C}})$ の実行コスト $PCEst(P(\hat{\mathscr{C}}))$ を PostgreSQL のオプティマイザで推定する．同様に真のカーディナリティを利用したときの実行コスト $PCEst(P(\mathscr{C}))$ も推定し，最後に P-Error を求める．

3.5.1.3 提案手法とベースライン

提案手法には，密度推定器として第 2 章で提案した Denoising Autoencoder ベースのモデルを利用する．Denoising Autoencoder の実装には Multi-Layer Perceptron を用いる．

比較手法には，統計情報を利用する（1）PostgreSQL（バージョン 11.8），グローバルスキーマでの密度推定を利用する（2）第 2 章提案と（3）NeuroCard [24, 51]，

相関ベーススキーマでの密度推定を利用する（4）DeepDB [25, 52] を用いる．既存手法のハイパパラメータはそれぞれの論文または公開されているソースコード [24, 25, 51, 52] で報告されている値を利用する．

3.5.1.4　実験環境

すべての手法で同じ 16CPUs（最大 2.5GHz）・メモリ 64GB・Nvidia Tesla T4 GPU を備えた実行環境にて評価を行う．評価対象の手法のうち，提案手法と NeuroCard は GPU による高速化が可能である．

3.5.2　カーディナリティ推定としての評価

評価の指針とモチベーション：結合カーディナリティ推定単体での性能評価を行う．各手法の Q-Error（Median・90 パーセンタイル・95 パーセンタイル・99 パーセンタイル・最大値），平均応答時間と学習時間から評価し，提案手法の有用性を確認する．

結果：JOB-light，JOB-m の Q-Error による評価結果をそれぞれ表 3.6，表 3.7 に示す．JOB-light では，常に本章の提案手法が最も良い推定性能を示した．JOB-light は 16 テーブル中最大 5 テーブルにしかアクセスしないことから，閉入近傍スキーマごとに密度推定器を構築したことが有効に働いたと思われる．平均応答時間に関しても同様に，密度推定器が分割されたスキーマごとであるため相対的にパラメータ数を削減でき，高速な推論に寄与していると考えられる．一方で第 2 章の提案手法や既存手法の NeuroCard は，ユニバーサルリレーションに対応する大きな密度推定器の利用や推論時に多数のダウンスケールが必要なこともあり，推定性能と応答性能はいずれも低かった．DeepDB はスキーマ分割のための相関計算にて 100GB 以上のメモリが必要となり動作しなかった．今回の実験で用いた IMDb データセットと同等以上の規模のデータでは同様のことが予想されるため，スケーラビリティに難があるといえる．応答時間だけで見るとヒストグラムを利用する PostgreSQL が最も高速だったが，推定性能とのトレードオフとなっていることがわかる．

JOB-m では，提案手法は PostgreSQL よりは性能が向上したものの，NeuroCard と比較すると劣る結果となった．JOB-light と異なり JOB-m は最大 11 テーブルにアクセスするため，ユニバーサルリレーションが有利になったと考えられる．JOB-light の場合と比較して，提案手法と NeuroCard の応答時間が大幅に増加した

が，これは JOB-m のクエリに含まれる IN 句や LIKE 演算子に関連する実装レベルの問題であると思われる．提案手法と NeuroCard はいずれも，ドメイン全体に対してそれらの条件判定を行う処理に多くの時間を要している．インデックスを作成し，効率的に条件判定することで改善が考えられる．

ここで，2 つのベンチマークで共通となっている密度推定器の学習時間に注目してみると，本章の提案手法が最も短いことがわかる．並列に学習可能であること，各密度推定器のパラメータ数が少ないことが有効に働いている．ユニバーサルリレーションを利用した手法で特にアクセスするテーブル数が多い場合と比較すると，推定性能では JOB-m の結果のように密度推定器を横断する近似を行う分劣ることがある．しかしながらより大規模なスキーマになると，学習時間を含めて考えたとき提案手法のほうが実用に向いたものになるといえるだろう．また，密度推定器が独立していることは並列に学習が可能という点以外のメリットもある．今後，データに変更がある環境を想定する際に，データ変更に追従するために密度推定器を更新することが考えられるが，全体ではなく更新があったテーブルが含まれる密度推定器のみを更新すれば良いという効果が期待できる．

表 3.6: JOB-light ベンチマークでの Q-Error，平均応答時間 (ms) と学習時間 (min)[10)]

Method	Median	90%th	95%th	99%th	Max	応答時間 (ms)	学習時間 (min)
第 3 章提案	1.60	4.19	6.87	19.7	33.0	22.8	66.3[12)]
第 2 章提案	1.68	5.66	22.1	33.6	34.8	50.4	128
NeuroCard	1.79	9.00	19.5	36.5	43.2	160	391
DeepDB	*—Out of memory to train (100GB+)—*						
PostgreSQL[11)]	7.44	163	$1.1 \cdot 10^3$	$2.8 \cdot 10^3$	$3.5 \cdot 10^3$	3.44	—

10） 同じデータセットを学習した同じモデルで評価するため，学習時間は等しい．
11） EXPLAIN 文を介した応答時間を報告するため，参考値である．カーディナリティ推定のみの所要時間は掲載している時間より短くなる．
12） 並列に利用可能な十分な計算機資源があると仮定している．

表 3.7: JOB-m ベンチマークでの Q-Error，平均応答時間 (ms) と学習時間 (min)[10)]

Method	Median	90%th	95%th	99%th	Max	応答時間 (ms)	学習時間 (min)
第 3 章提案	3.72	378	1.4×10^3	1.6×10^4	8.7×10^4	1.7×10^3	66.3
NeuroCard	2.74	160	559	$4.8 \cdot 10^3$	$1.7 \cdot 10^4$	846	391
DeepDB	*—Out of memory to train (100GB+)—*						
PostgreSQL[11)]	160	$5.8 \cdot 10^3$	$1.3 \cdot 10^4$	$8.7 \cdot 10^4$	$1.0 \cdot 10^5$	6.95	—

3.5.3 クエリオプティマイザに与える影響の評価

評価の指針とモチベーション：結合カーディナリティ推定の応用の 1 つとして，クエリオプティマイザへの応用を想定した評価を行う．各手法の P-Error（Median・90 パーセンタイル・95 パーセンタイル・99 パーセンタイル・最大値）から評価し，提案手法が良い実行プランの作成に寄与することを確認する．

結果：JOB-light，JOB-m の P-Error による評価結果をそれぞれ表 3.8，表 3.9 に示す．まず JOB-light では，本章の提案手法が最もエラーを抑えることができた．これによりクエリオプティマイザを介したクエリ処理の高速化が期待できるといえる．一方で JOB-m では，既存手法と大きな差がないかやや劣るケースが確認された．原因として表 3.7 の Q-Error でも見られるように，分割したスキーマを横断した推論が数多く必要になると，オーバーラップしているテーブル以外の密度推定器間でやり取りできない相関の影響が増え，性能低下につながっていると思われる．ただしこの推定性能の低下は 3.5.2 項でも示したように個別の推論にかかる時間や学習時間とのトレードオフとなっている．そもそも動作しないということは避けなければならないため，特にスキーマサイズが大きい環境では提案手法のようなスケーラビリティが重要となる．今後，より大規模なスキーマを持つベンチマークで評価することが考えられる．

3.6 おわりに

本章では複雑なスキーマを持つデータの高速な処理のための要素技術の 1 つとして，第 2 章で提案した推定器を複数組み合わせることにより効率的な結合カーディナリティ推定を行う手法を提案した．提案手法の主な特徴は 3 点にまとめられる．

表 3.8: JOB-light ベンチマークでの P-Error

Method	Median	90%th	95%th	99%th	Max
第 3 章提案	1.00	1.17	1.32	1.97	2.26
第 2 章提案	1.00	1.40	1.80	2.34	2.41
NeuroCard	1.01	1.56	2.27	2.53	4.11
DeepDB	*—Out of memory to train (100GB+)—*				
PostgreSQL	1.00	1.24	1.34	2.01	2.63

表 3.9: JOB-m ベンチマークでの P-Error

Method	Median	90%th	95%th	99%th	Max
第 3 章提案	1.00	3.12	7.53	37.8	50.4
NeuroCard	1.02	2.33	2.94	15.7	50.4
DeepDB	*—Out of memory to train (100GB+)—*				
PostgreSQL	1.03	3.15	6.27	36.9	50.3

1 つ目の特徴はスキーマサイズに対するスケーラビリティである．データベーススキーマに対してグラフ構造に基づくシンプルな分割を行い，分割されたスキーマごとに密度推定器の学習を行う．各密度推定器はコンパクトになるため訓練が容易で，並列学習も可能となる．2 つ目の特徴はデータサイズに対するスケーラビリティである．学習データを適切にサンプリングして利用することで，小規模から大規模データまで柔軟に対応できる．3 つ目の特徴は高い推定性能である．密度推定器の対象範囲がクエリにフィットしやすく，推論性能が高くなる．特にクエリオプティマイザから利用する際に効果的である．複数のベンチマークと複数の指標で評価し，大規模スキーマ環境やクエリオプティマイザへの応用で提案手法が有用であることを確認した．

一方で，本章の提案手法は第 2 章で提案した手法ほど既存手法と比較したときの高速化が見られないケースが確認されている．この一因として，密度推定器間でのサンプル受け渡しによるオーバーヘッドが考えられる．各密度推定器はそれぞれの埋め込み層を持っているため，共通属性のサンプルであっても先行する密度推定器デコードしてから後続の密度推定器でエンコードする必要がある．提案手法の特徴

を損なうことなくこのオーバーヘッドを解消することが今後の課題である.

第 4 章　カーディナリティ推定と差分プライバシ

4.1　はじめに

第 1 章で述べたように，個人に関するデータ，つまりパーソナルデータの利活用が進んでいる．パーソナルデータの利用はサービスに有用であることが多いが，一方で個々人の特定やプライバシの開示を防ぐためプライバシの保護が必要となる．よく知られたプライバシ保護の方法としてデータの k-匿名化があげられる．しかしながら，リンケージ攻撃以外の攻撃や複数の問合せによるモザイク効果などによってプライバシが開示されてしまうこと知られており [14]，プライバシ保護として常に厳密とは言えない．このような背景から，攻撃手法や背景知識に依存しないプライバシ指標として差分プライバシ [17, 65, 66] が提案されている．差分プライバシは一定の識別困難性に基づき任意の攻撃に対してプライバシ保護を行う．例えばある計算機構が差分プライバシを満たしている場合，いかなる攻撃手段や背景知識を利用してもプライバシ強度パラメータで指定される以上に特定レコードを識別することが不可能と保証される．差分プライバシの有用性は既に認知され始めており，例えば Apple 社はユーザデータを差分プライバシで保護して扱っていることを公表している [67]．その他の応用例として，データベースに対する SQL の問い合わせを差分プライベートにすることを目的とした Differentially Private SQL (DPSQL) [68, 69, 70, 71] があげられる．既存のデータ処理から SQL という広く用いられているインターフェースを変えることなく，パーソナルデータを対象としたカウントクエリ（カーディナリティ推定）・結合演算・集約演算などが扱えるようになるという点で有用である．

データを元に予測や推定を行う機械学習でもパーソナルデータが含まれる場合はプライバシ保護が必要となる．実際にパーソナルデータを学習した深層学習モデルに対する攻撃手法 [72] や開示が望ましくないデータを記録する傾向 [73] が報告されている．差分プライバシはこれらの問題を解決する選択肢の 1 つであり，学

習時に差分プライバシを満たす，差分プライベート学習手法が提案されている．ニューラルネットワークでは，一般的に，確率的勾配降下法（SGD）によってパラメータの更新を行うため，パラメータにパーソナルデータの情報が埋め込まれることとなる．Abadi ら [31] はこの点に注目し，SGD によるパラメータ更新の際に勾配クリッピングとノイズ付与を行うことで差分プライバシを満たす Differentially Private Stochastic Gradient Descent (DPSGD) を提案している．DPSGD を利用して学習を行うことで，学習済みモデルから学習データを推論する攻撃 [72, 74] や再構築する攻撃に対応できることが報告されている [75]．DPSGD は安全性を向上させる一方で，SGD と比較して，勾配クリッピングとノイズ付与の影響によりモデルの有用性が低下するというトレードオフがある．そのため，タスクや安全性の度合いによっては有用なモデルが得られないことがある．差分プライバシを満たすニューラルネットワークモデルの実用上の意義は大きく，DPSGD の有用性が低下するという問題を改善する手法も提案されている [32, 33, 76]．いずれの手法も更新するパラメータを減らすことでクリッピングやノイズの影響を抑えるというアプローチをとる．具体的には更新対象パラメータ数削減のために，重み行列や勾配行列を低ランク近似する手法 [32, 76] やスパース化して扱う手法 [33] が提案されている．ここで利用されている重み行列や勾配行列の低ランク性とスパース性という 2 つの性質はそれぞれ大域的な偏りと局所的な偏りに基づくものとみなすことができる（4.3.1 項で詳説する）．そこで対象の異なるこの 2 つの性質を同時に利用することでより効率的な更新対象パラメータ数削減が考えられるが，併用を単純に実現することはできない．先に低ランク近似した後にスパース化するケースでは，元の行列の各要素に独立した情報は低ランク近似された行列上では失われているため，スパース化が困難となる．一方先にスパース化をした後に低ランク近似するケースでは，最終的な更新対象パラメータ数は低ランク近似のみを行ったケースと同等であり，スパース化による恩恵が受けられない．

本章では，ニューラルネットワークパラメータの低ランク性とスパース性という 2 つの冗長性に注目し，その両方を考慮した更新対象パラメータ数削減により DPSGD から来る悪影響を軽減することを目指す．

本章の主な貢献点は以下の通りである．

- ニューラルネットワークパラメータの低ランク性とスパース性の両方を利用

し，更新対象パラメータを大きく削減することで差分プライバシを満たしながら有用性の高いモデルとして学習する手法 **LSG** (**L**ow-rank and **S**parse properties of neural networks' **G**radient) を提案する．LSG では，先に大域的な偏りによる冗長性に基づく低ランク近似を行う．さらに，ニューラルネットワークの構造として全結合層のユニットとユニットの接続関係を工夫して利用することで，近似した行列に対して局所的な偏りによる冗長性に基づくスパース化を行う．いずれもニューラルネットワークパラメータに経験的に知られている冗長性に基づいた処理であるため，モデルの持つ重要な情報が失われにくく，結果として有用性の高いモデルの学習が可能である．

- LSG を全結合層に基づく構造だけでなく，Attention 層や畳み込み層に適用できるような拡張を提案する．ここでは低ランク近似に加えて，畳み込み層内のチャネルレベルの偏りを利用したスパース化を行うことで実現する．
- タスク，モデル構造や学習パターンを変えた様々なシーンで LSG が有効であることを実験的に確認した．スクラッチからの学習として，Denoising Autoencoder による近似カウントクエリ処理（第 3 章）では，エラーの中央値から最大値まで全体的に改善することが，畳み込み層を含むモデルで画像処理タスクで評価した場合では正解率が最大 7%向上することが確認された．またファインチューニングの場合，学習済み大規模言語モデル RoBERTa [77] を利用して自然言語処理タスクで評価したところ，次点の性能を示した手法と比較して正解率が最大 4%向上することが確認された．

本章の以降の節は以下の通りである．4.2 節で差分プライバシとニューラルネットワークに関する事前知識を導入し，4.3 節で提案手法 LSG を詳説する．4.4 節で LSG の評価を行い，4.5 節で本章をまとめる．

4.2 事前準備

この章では提案手法の基礎となる概念や既存技術について説明する．

4.2.1 (ϵ, δ)-差分プライバシ

差分プライバシ [17] とは，データベースに含まれるパーソナルデータの保護を目的とした指標である．差分プライバシを満たした計算機構の場合，いかなるレコード集合が対象となる問合せを組み合わせても特定のレコードが含まれているかどうかの識別が困難となる．これを以て特定のレコード，つまりパーソナルデータのプライバシが開示されないことを保証している．識別困難性に基づいているため，攻撃手法や背景知識に依らない安全性の指標となっている点が特徴的である．以下では差分プライバシの定義について述べる．

定義 1 ((ϵ, δ)-差分プライバシ [17])．レコード x_i の集合をデータベース $D = \{x_i\}_{i=1}^n$ とし，取り得るデータベースの集合を $\mathfrak{D}$ とする．D から情報を取り出す処理を Q とし，その出力が取り得る集合のサブセットを R とする．任意の隣接[1)]したデータベースの組 $(D_1, D_2) \in \mathfrak{D}$ と任意の取りうる出力 R に対して以下が成立するとき，Q は (ϵ, δ)-差分プライバシを満たしていると言う．

$$Pr(Q(D_1) \in R) \leq e^{\epsilon} \cdot P(Q(D_2) \in R) + \delta$$

任意の隣接したデータベース D_1, D_2 に対する問い合わせ結果 $Q(D_1)$ と $Q(D_2)$ のどの要素の確率を見てもパラメータ ϵ, δ で指定された程度に類似している[2)]，つまり特定のレコードがデータベースに含まれているかどうかの推定が困難であることを表している．なお，妥当なプライバシパラメータの決定はユースケースや要件に依存する．この問題を解決するため，Epsilon Registry [78] が提案されている．これは具体的なパラメータを提案するのではなく，差分プライバシを利用するユーザがプライバシパラメータを公開することで，相対的な安全性の確認や妥当なパラメータ決定の補助が可能になるというスキームの提案である．

実際に差分プライバシを満たすための手法として，ガウシアンノイズを利用したガウシアンメカニズムが知られている．

定義 2 (ガウシアンメカニズム [17])．$f : X \to \mathbb{R}$ を入力 X を取りセンシティビティが S_f の関数とする．このとき，ガウシアンメカニズム $\mathcal{M}_\sigma$ は ϵ・δ・データの参照回数から求まる σ を利用して以下のように f の出力に対してノイズ加算を行うこ

1) ここでの隣接とは，1 レコードのみが異なることを表す．
2) 分布が類似しているともいえる．

とで (ϵ, δ)-差分プライバシを保証する．

$$\mathscr{M}_\sigma(x) = f(x) + \mathcal{N}(0, {S_f}^2\sigma^2\boldsymbol{I})$$

差分プライバシの特徴として，既に差分プライバシを満たした出力は再利用しても追加のプライバシコストがかからない．

定理 1 (Post-processing 定理 [17])．$\mathscr{M} : \mathscr{D} \to \mathscr{R}$ を (ϵ, δ)-差分プライバシを満たす関数，$\mathscr{R}$ を別の出力集合 $\mathscr{R}'$ に写す $f : \mathscr{R} \to \mathscr{R}'$ を任意の関数とする．このとき $f \circ \mathscr{M} : \mathscr{D} \to \mathscr{R}'$ は (ϵ, δ)-差分プライバシを満たす．

4.2.2 DPSGD: Differentially Private Stochastic Gradient Descent

DPSGD [31, 79, 80] とは，差分プライバシを満たす確率的勾配降下法である．モデルの構造に依らず，パラメータ更新に利用することで差分プライバシを満たすモデルとなる．具体的には，勾配クリッピングとノイズ付与を追加で行う．サンプル x_i ごとに計算された勾配 $\mathbf{g}_{x_i}$ の L2 ノルムを閾値 C でクリッピング $(\mathbf{g}_{x_i}/\max(||\mathbf{g}_{x_i}||_2, C))$ することで，サンプルごとの影響度合い（センシティビティ $S = C$）に上界を定める．加えて，ガウシアンメカニズム（定義 2）を用いて勾配にノイズを加算することで，(ϵ, δ)-差分プライバシの満足を保証する．

$$\mathscr{M}_\sigma(x) = f(x) + \mathcal{N}(0, C^2\sigma^2\boldsymbol{I})$$

パラメータ更新に利用する勾配に変更を加えるため，差分プライバシを考慮しない場合と比較して得られるモデルの有用性は低下する．この度合いはパラメータ ϵ， δ に依存しており，安全性を向上させると有用性が低下するトレードオフとなっている．

4.2.3 DPSGD を拡張した既存手法

DPSGD では差分プライバシによる安全性が担保された一方で，有用性の低下が大きく，実用に問題が出てしまうことが実験的に示されている [32]．そこで，DPSGD と同等の安全性を担保したまま，有用性を向上させる提案が行われている [32, 33, 76]．いずれの手法も DPSGD によるクリッピングとノイズの影響を緩和することでモデルの有用性向上を図っている．主に不必要なパラメータの更新を削減することで，元の勾配が持つ情報量以上にノイズの影響を受けてしまうことを避

けるアプローチが取られている．このアプローチの実現方法は，ニューラルネットワークに表れる重み行列と勾配行列の低ランク性を利用したものとスパース性を利用したものの 2 つに大別される．

4.2.3.1 低ランク性に基づく手法

低ランク性とその利用について紹介する．行列のランクが低いということは，行列分解を行い要素数の少ない行列として扱っても元の行列の情報を保てることを意味する．ニューラルネットワークの重み行列と勾配行列の低ランク性は経験的に知られており [32, 76, 81, 82, 83, 84]，DPSGD の拡張以外に圧縮や高速化のためにも利用され，その有効性が示されている．

Reparametrized Gradient Perturbation (RGP)： RGP [32] とは，低ランク近似を利用する差分プライバシを満たすニューラルネットワークのパラメータ更新手法である．重み行列 $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ を，Power method（Algorithm6）によりランク r で低ランク近似した行列 $\boldsymbol{L} \in \mathbb{R}^{m \times r}, \boldsymbol{R} \in \mathbb{R}^{r \times n}$ と順伝播のみに利用する残差行列 $\tilde{\boldsymbol{W}} \in \mathbb{R}^{m \times n}$ を用いて以下のように再定義する．

$$
\begin{aligned}
&\boldsymbol{L}, \boldsymbol{R} = \mathrm{PowerMethod}(\boldsymbol{W}, r) \\
&\tilde{\boldsymbol{W}} = \boldsymbol{W} - \boldsymbol{L}\boldsymbol{R} \\
&\boldsymbol{W} \overset{reparametrize}{\rightarrow} \boldsymbol{L}\boldsymbol{R} + \tilde{\boldsymbol{W}}.\mathrm{stop\_gradients}()
\end{aligned}
$$

stop_gradients() はパラメータ更新を行わないことを示す．入力を x，出力 y としたとき，順伝播は以下のように表される．

$$
y = \boldsymbol{L}\boldsymbol{R}x + \tilde{\boldsymbol{W}}x \tag{4.1}
$$

ここから，$\boldsymbol{L}, \boldsymbol{R}$ の勾配はそれぞれ以下のように表される．

$$
\partial \boldsymbol{L} = (\partial \boldsymbol{W})\boldsymbol{R}^T, \quad \partial \boldsymbol{R} = \boldsymbol{L}^T(\partial \boldsymbol{W}) \tag{4.2}
$$

$\boldsymbol{L}$ の列と $\boldsymbol{R}$ の行が正規直交基底を成している場合，$\boldsymbol{W}$ は以下の値で更新できる．

$$
(\partial \boldsymbol{L})\boldsymbol{R} + \boldsymbol{L}(\partial \boldsymbol{R}) - \boldsymbol{L}\boldsymbol{L}^T(\partial \boldsymbol{L})\boldsymbol{R} \tag{4.3}
$$

このとき，DPSGD と同様に勾配クリッピングとノイズ付与を $\partial \boldsymbol{L}, \partial \boldsymbol{R}$ に行うことで差分プライバシを満たす．$r < \min(m, n)$ となる r を選べば，$\partial \boldsymbol{W}$ と比較して

$\partial \boldsymbol{L}, \partial \boldsymbol{R}$ の合計パラメータ数が少なくなり $(r(m+n) < mn)$，DPSGD を直接利用した場合と比較してモデルの有用性が高くなることが報告されている．しかしながら，安全性を考慮しない通常の学習と比較すると，ニューラルネットワークパラメータの局所的な傾向を捉えられておらず，依然として有用性の低下は大きい．

4.3 節で提案する手法は局所的な傾向も捉えることで有用性の向上を図る．

Algorithm 6 Power method による行列分解

Input: Weight matrix $\boldsymbol{W} \in \mathbb{R}^{m \times n}$, rank r

Output: Gradient matrices $\boldsymbol{L} \in \mathbb{R}^{m \times r}$, $\boldsymbol{R} \in \mathbb{R}^{r \times n}$

1: Initialize $\boldsymbol{R}$ from standard Gaussian distribution $\mathcal{N}(0, 1)$.
2: $\boldsymbol{L} \leftarrow \boldsymbol{W}\boldsymbol{R}^T$
3: Orthonormalize the columns of $\boldsymbol{L}$.
4: $\boldsymbol{R} \leftarrow \boldsymbol{L}^T\boldsymbol{W}$
5: Orthonormalize the rows of $\boldsymbol{R}$.
6: return $\boldsymbol{L}$, $\boldsymbol{R}$

4.2.3.2 スパース性に基づく手法

スパース性とその利用について紹介する．行列がスパースであるということは，0（ごく小さい値が無視できる文脈ではそれも含む [3)]）である要素が多く，効率的な格納方式や計算方式を採用できることに繋がる．多くのニューラルネットワークの重み行列や勾配行列にスパース性は表れており，圧縮や高速化を目的とした枝刈り技術を中心に広く活用されている [85, 86, 87]．

Sparse Network Finetuning with DPSGD (SNF-DPSGD)： SNF-DPSGD [33] とは，スパース性に基づくニューラルネットワークモデルの効率的な差分プライベートファインチューニング手法である．スパース性を仮定した枝刈り技術 [87, 88] に倣い，重みの絶対値が小さいパラメータを重要でないパラメータとして扱い，そのパラメータを DPSGD による更新対象から除外する．このときパブリックデータで事前学習した重みを参照することで，定理 1 により追加のプライバ

3) $W_{i'j'} < \theta, \theta \approx 0$ はアクティベーションへの貢献がごく小さい $(\sum^{i \neq i' \wedge j \neq j'} W_{ij}x_{ij} \approx \sum W_{ij}x_{ij})$ ため $W_{i'j'} = 0$ として扱うことができる

シコストは必要としない．一方で事前学習した重みは不変であるため，更新されるパラメータはファインチューニングの中で常に一定である．また，ドメイン適応の先行研究 [89] に基づき，スパース性の仮定は畳み込み層のパラメータのみを対象としている．DPSGD でファインチューニングした場合と比較してモデルの有用性が高くなることが報告されているが，これは個別の要素のみを利用しており大域的なニューラルネットワークパラメータの傾向を利用できていないこと，パブリックデータで事前学習されたモデルのファインチューニングのみに利用できることに注意されたい．

4.3 節で提案する手法は大域的な傾向を併用することで有用性を向上させ，さらに事前学習モデルに依存しないスパース化を実現することでより多くのケースで利用可能なものとする．

4.2.4 ニューラルネットワークプルーニング

ニューラルネットワークの圧縮や高速化を目的として多数のプルーニング手法 [85, 86, 87, 90, 91] が提案されている．典型的なものとしては絶対値が小さい重みを除外するというアプローチがあり，この操作は破壊的であるものの，経験的にある程度の割合であればモデルの有用性に大きな悪影響はないことが知られている．具体的なアプローチは様々あり，ニューラルネットワークの構造に基づいた，例えば全結合層のユニット単位 [86] や畳み込み層のカーネルやチャネル単位 [90, 91] でのプルーニングが提案されている．いずれの手法も，経験的に，ニューラルネットワークの構造に基づいた重みのグループごとに重要度の偏りがあるということに基づいてプルーニング対象を決定している．プルーニングに似たような操作として，過学習を抑えるための Dropout [92] が知られている．プルーニングと比較して一部の重みを除外するのではなく更新をスキップするだけという違いはあるものの，更新対象パラメータ数を減らしているにも関わらず学習に良い影響を与えている．これらから，パラメータには冗長性やむしろ一時的に学習しないほうが良いというケースがあることがわかる．前述した SNF-DPSGD [33] はこれらの背景に基づいて，DPSGD をベースとした差分プライベートなファインチューニング時に重要でない重みの更新をスキップするという手法を提案している．しかしながら，4.3 節や 4.4 節の実験結果で示すように，スパース化だけを用いた更新対象パラメータ数削減は局所的な偏りしか捉えられず不十分である．提案手法では，特にニューラ

ルネットワークの構造に基づいた重みのグループごとに重要度の偏りがあるという点を活用し，低ランク近似と両立させることでより効率的な差分プライベート学習の実現を図る．

4.3 提案手法

得られるモデルの有用性が高い差分プライベート学習手法を提案する．既存のDPSGD を拡張した手法では，ニューラルネットワークパラメータの低ランク性とスパース性のいずれかを利用して更新対象パラメータを削減することで，DPSGD 処理の影響の軽減から有用性の改善を図っている．低ランク性とスパース性は独立したものであるが，行列分解された行列の成分は元の行列の複数の成分に影響を持つため，その組み合わせを単純に扱うことは出来ない．そのため，低ランク近似とスパース化を両立した更新対象パラメータ数削減手法は存在していない．

提案手法では，ニューラルネットワーク内のユニットの接続関係とその重要度の偏りに注目する．プルーニング技術 [86,90,91] に倣い，重み行列から各ユニットの重要度を集約することで入力ユニットと出力ユニットの重要度を定義する．低ランク近似した行列にも入力ユニットや出力ユニットとの対応関係があるため，定義した重要度を利用することで低ランク近似された勾配行列のスパース化が可能となる．これにより，低ランク性とスパース性のいずれかだけを利用した場合と比較して，両方の性質から捉えられる多くの更新対象パラメータを削減し，結果として差分プライバシのための勾配クリッピングやノイズの影響が軽減され，得られるモデルの有用性が向上する．実際に，低ランク性のみの場合と提案手法による低ランク性とスパース性を併用した場合をクリッピングの観点で比較すると，リスト 1 のようになる．各要素が学習データに対応しており，数値が 1 であることは勾配行列のノルムがクリッピング閾値を下回りクリップされないことを，数値が 1 未満である場合は値が小さいほど勾配の値が大きくクリップされることを意味する．スパース化すると勾配行列のノルムは単調に小さくなり，多くの要素でクリッピングの影響がなくなるか軽減されていることがわかる．提案手法の主な流れは以下の通りである．

1. 重み行列から入力側ユニットと出力側ユニットの重要度を算出

```
// 低ランク性のみ
0.9727, 0.9805, 1.0000, 1.0000, 0.9678, 1.0000, 1.0000,
0.9023, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000,
0.6670, 1.0000, 0.9531, 1.0000, 1.0000, 1.0000, 0.9473

// 低ランク性 + スパース性 (p = 0.3)
1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000,
1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000,
0.7300, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000
```

Listing 1: スパース化によるクリッピングへの影響例.
クリッピング前後のノルム比であり，値が小さい要素ほど勾配が大きくクリップされる．

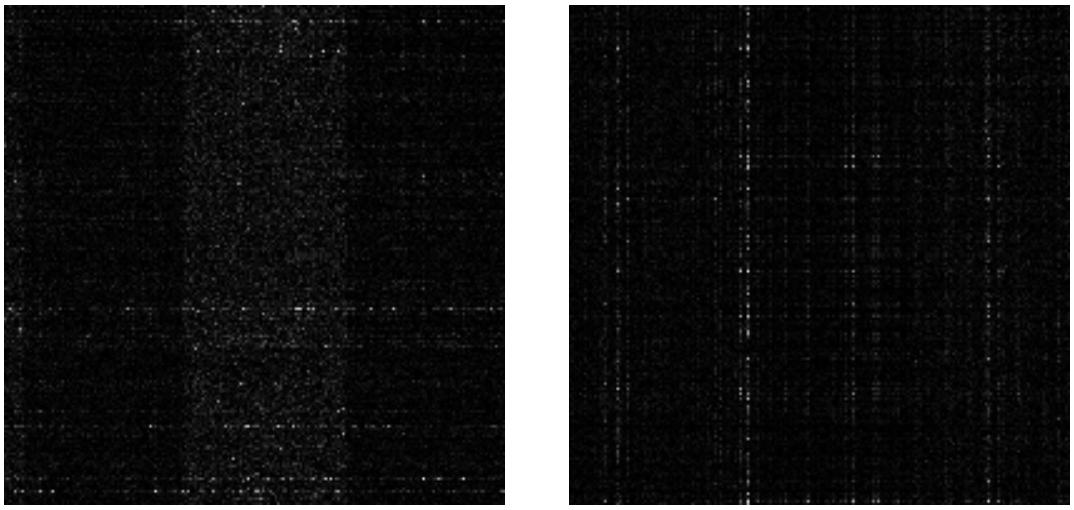
2. 低ランク近似された勾配行列の取得
3. 勾配行列に DPSGD による差分プライバシ処理を適用
4. 入力側ユニットと出力側ユニットの重要度に基づき低ランク近似された勾配行列をスパース化
5. 勾配行列を利用して重みを更新

4.3.1 項で更新対象パラメータ数の削減に利用する性質を個別に導入した後，4.3.2 項でそれらの性質の併用と具体的な手順について詳説する．最後に 4.3.3 項でより広い深層学習モデルアーキテクチャに対応できるように拡張を行う．

4.3.1 更新対象パラメータ数の削減

ここでは提案手法で利用する更新対象パラメータ数削減のアプローチについて述べる．提案手法では重み行列と勾配行列の低ランク性とスパース性の両方に注目する．例えば学習済み言語モデルである RoBERTa [77] の重み行列は図 4.1a のようになっており，多くのパラメータが ≈ 0 でスパース)[3)]であることがわかる．各軸が入力側ユニットと出力側ユニットに対応していることを踏まえると，ユニット単位に重要度の偏りがあることも確認できる．また，各軸方向に似た傾向を持つベクト

ルが多数見られることから，低ランク性も示唆されている．学習フェーズに見られる勾配行列も同様の傾向であることが図 4.1b からわかる．ここからは提案手法で利用する低ランク近似とスパース化それぞれ個別に導入を行う．



(a) 重み行列　　(b) 勾配行列

図 4.1: 学習済み RoBERTa モデルのパラメータ例．明るい要素ほど値が大きいことを示す．

まずニューラルネットワークパラメータの低ランク性の利用について述べる．この性質は経験的に知られているものであり [81, 82, 83, 84]，例えばニューラルネットワークの圧縮手法として利用されている [82, 83, 84]．これは行列を大域的に見たときに重要な部分（主成分）に絞り込んでいると解釈することができる．行列 $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ に対してランク r での近似を考えるとパラメータ数は $r(m+n)$ となる．$r \ll \min(m, n)$ である場合 $r(m+n) \ll mn$ となり，大幅にパラメータ数を削減できることがわかる（図 4.2a/図 4.2b）．ニューラルネットワークのパラメータ更新のための低ランク近似された勾配行列を得る手法には様々なものが考えられるが，提案手法は汎用的なインターフェースであるため具体的な手法の種類は問わない．4.4 節の評価実験では RGP [32] を用いているが，その他の手法でも容易に適用可能である．低ランク近似した行列を扱う際の一般的な課題として，ランク r を小さくするとパラメータ数をより削減できるが，一方で，主成分として捉えられなかった情報が失われるため精度が低下するというトレードオフがある．提案手法では後述

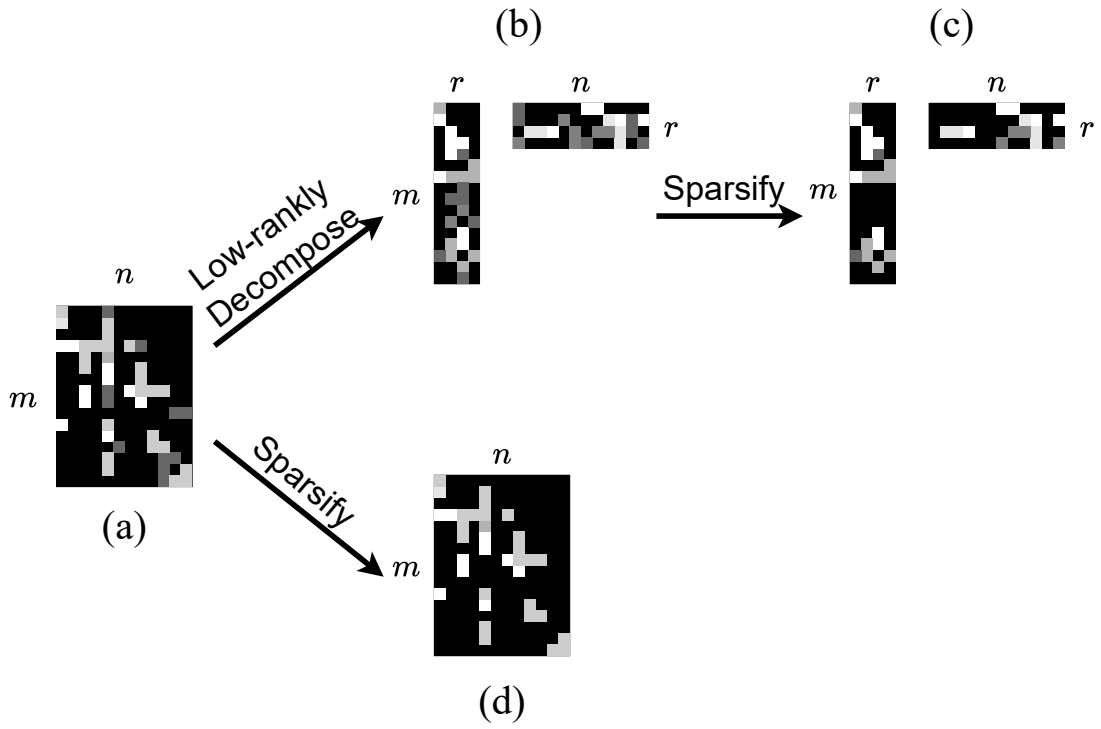


図 4.2: 低ランク近似とスパース化

するスパース性と組み合わせることで削減するパラメータ数と精度の両立を行う．詳しくは 4.3.2 項で述べる．

次にニューラルネットワークパラメータのスパース性の利用について述べる．このスパース性も低ランク性と同じく経験的に知られており，様々な手法の前提となっている．代表的な応用としては圧縮や高速化を目的とした枝刈り手法 [85, 86, 87] が挙げられる．例えば $p\%$ のスパース性を仮定した枝刈りを行うと単純に $p\%$ のパラメータ数削減となる（図 4.2a/図 4.2d）．また，特に大規模モデルにおけるスパース性は宝くじ仮説 [88] を通して再確認されている．これは，ニューラルネットワークモデルで性能への寄与が大きいのは構造と初期値の組み合わせとして確率的に存在するスパースな部分ネットワーク（当たりくじ）であり，大規模モデルであることはこの当たりくじが含まれる可能性が高くなるからである，という仮説である．つまりパラメータ数が過剰な場合，ニューラルネットワークモデルの本質はスパースな部分モデルに集中しているとも言える．このことはモデルをスクラッチから学習した場合だけでなく，学習済みモデルに対するファインチューニングでも同様に成り立つと報告されている [93, 94]．提案手法では重みの絶対値を重

要度としたスパース性を利用する．絶対値を利用するというアプローチは，重みは正負に依らず貢献があり，絶対値が小さい重みはアクティベーションに対する貢献が少ないということに基づいている．この方法は単純ながら多数の枝刈り手法で有効性が示されている [85, 86, 87]．加えて，一部の枝刈り手法でも利用されているユニット単位の重要度 [86] を導入する．入力側ユニットと出力側ユニットのそれぞれで，接続されているシナプスの重要度の総和をユニットごとに計算し，それを各ユニットの重要度とする．i 番目の入力側ユニットの重要度 I_i と j 番目の出力側ユニットの重要度 O_j は以下のように算出する．

$$I_i = \sum_{j=1}^{n} |W_{ij}| \tag{4.4}$$

$$O_j = \sum_{i=1}^{m} |W_{ij}| \tag{4.5}$$

4.3.2 LSG: Low-rank and Sparse Gradients

ここからは 4.3.1 項で導入した低ランク性とスパース性を併用して更新対象パラメータ数削減を行う手法について述べる．本研究は指定した安全性の差分プライバシを満たすことを目的としているため，プライベートな学習データを追加で参照すると追加コストとしてノイズを大きくするか安全性を下げる必要が出てしまう．そのため，追加コストが必要とならない範囲で更新対象パラメータを削減することで，安全性を下げることなくモデルの有用性の向上を図る．4.3.1 項で述べた通り，低ランク性とスパース性のどちらかだけでは削減できる更新対象パラメータ数は限られているため，提案手法ではその両立を行う．提案手法の概念図を図 4.3 に示す．

スパース性のみを利用した既存研究 [33] で用いられているシナプスの重要度 $\mathrm{abs}(\boldsymbol{W}) \in \mathbb{R}^{m \times n}$ は行列分解された勾配行列 $\partial \boldsymbol{L} \in \mathbb{R}^{m \times r}, \partial \boldsymbol{R} \in \mathbb{R}^{r \times n}$ と対応が取れず，利用することができない．そこで，$\partial \boldsymbol{L}$ の行は入力側ユニットと，$\partial \boldsymbol{R}$ の列は出力側ユニットに対応があることに注目する．式 4.4 と式 4.5 で定義したユニットの重要度を利用し，重要度の低い $p\%$ の重みに対応する勾配を 0 にすることでス

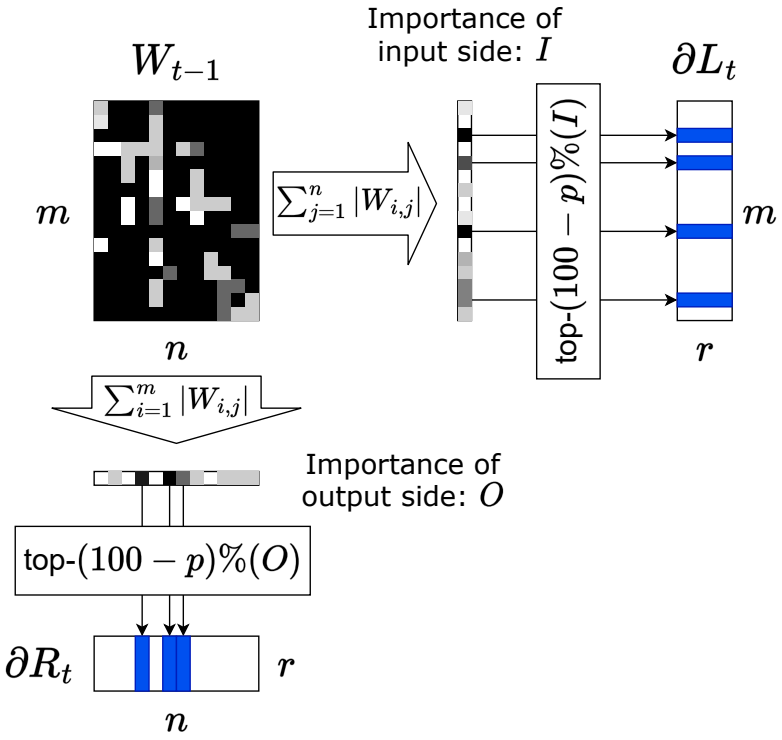


図 4.3: LSG による全結合層の低ランク近似とスパース化の併用例．前ステップ $t-1$ の重み行列 $\boldsymbol{W}_{t-1}$ を利用して，低ランク近似された勾配行列 $\partial \boldsymbol{L}_t$ と $\partial \boldsymbol{R}_t$ をスパース化する．青色の要素はスパース化対象を意味し，現在のステップ t での更新対象から除外される．

パース化する．この処理は以下の式で表される．

$$\forall j \in [r], \partial \boldsymbol{L} = \begin{cases} \partial L_{ij} & \text{if } I_i \text{ in top-}(100-p)\%(\boldsymbol{I}) \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

$$\forall i \in [r], \partial \boldsymbol{R} = \begin{cases} \partial R_{ij} & \text{if } O_j \text{ in top-}(100-p)\%(\boldsymbol{O}) \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

これにより，更新対象パラメータ数は $k(m+n)\frac{100-p}{100} < k(m+n) \ll mn$ (where $k \ll \min(m,n) \wedge 0 < p < 100$) に削減できる．

なお，初期の重み $\boldsymbol{W}_0$ をスクラッチから学習する場合は乱数による重みを，ファインチューニングを行う場合は事前学習した重みを利用することで，ステップ 1 での重み更新に追加のプライバシコストを必要としない．さらに，ステップ t で参

照する $\boldsymbol{W}_{t-1}$ は差分プライバシを満たしているため，定理 1 により追加のプライバシコストを必要としない．まとめると，提案手法は，追加のプライバシコストを支払うことなくステップに応じた最新の情報に基づいたスパース性を利用している．これは，ステップ数に対して不変なパブリックデータで事前学習した重みに基づくスパース性を利用している既存研究 [33] と異なる点である．ファイチューニングが進むにつれて重要になる部分ネットワークは変化することが知られており [93]，提案手法の有用性向上に寄与していると考えられる．

最後に，低ランク近似に RGP を利用した場合の詳細な手順を Algorithm8 に示す．あるステップ t におけるパラメータ更新をランク r，スパース性 $p \in \{0, 1\}$ で選択的に行うことを考える．入力側出力側それぞれでユニットの重要度を算出し（3 - 8 行），重要でないユニットを選択する（9 - 10 行）．RGP（もしくはその他の勾配の低ランク近似手法）から行列分解された勾配行列を得た後（11 行），ユニットレベルの重要度 $\tilde{\boldsymbol{I}}$，$\tilde{\boldsymbol{O}}$ を利用してスパース化する（15 - 26 行）．その後差分プライバシのためのクリッピングとノイズ加算を行う（28 - 29 行）．このとき全てのレイヤにおいて $\partial \boldsymbol{L}, \partial \boldsymbol{R}$ がスパース化されているため，サンプルごとの勾配の L2 ノルムは小さくなり，クリッピングの影響が軽減される．同様にノイズを付与する対象の要素数も $\tilde{\boldsymbol{I}}$，$\tilde{\boldsymbol{O}}$ のマスクを使ったスパース化により削減されている（Algorithm7 3 行）．最後に得られた勾配を利用して重みの更新を行う（30 - 32 行）．

Algorithm 7 勾配行列に対するスパース性を考慮した差分プライバシ処理

Input: per-sample low-rank gradient matrices $\partial \boldsymbol{L}(x_q) \in \mathbb{R}^{m \times r}$ or $\partial \boldsymbol{R}(x_q) \in \mathbb{R}^{r \times n}$ with respect to a minibatch of data $\{x_q\}$, unimportant input or output units $\tilde{\boldsymbol{I}}$ or $\tilde{\boldsymbol{O}}$, noise multiplier σ^2, clipping size C

Output: DP sanitized gradient matrices $\tilde{\partial} \boldsymbol{L} \in \mathbb{R}^{m \times r}$, $\tilde{\partial} \boldsymbol{R} \in \mathbb{R}^{r \times n}$

1: Clip per-sample gradients with L_2 norm threshold C

2: Sum per-sample gradients to obtain $\partial \boldsymbol{L}$ or $\partial \boldsymbol{R}$

3: Perturb with noise $\boldsymbol{z}$ sampled from $\mathcal{N}(0, \sigma^2 C^2)$ masked with $\tilde{\boldsymbol{I}}$ or $\tilde{\boldsymbol{O}}$:

$$\tilde{\partial} \boldsymbol{L} \leftarrow \partial \boldsymbol{L} + \boldsymbol{z} \quad \text{or} \quad \tilde{\partial} \boldsymbol{R} \leftarrow \partial \boldsymbol{R} + \boldsymbol{z};$$

4: return $\tilde{\partial} \boldsymbol{L}$ or $\tilde{\partial} \boldsymbol{R}$

Algorithm 8 LSG．低ランク性とスパース性を利用した選択的パラメータ更新による差分プライベート学習（低ランク近似には RGP を利用）．

Input: current step t, weight matrix at previous step $\boldsymbol{W}_{t-1} \in \mathbb{R}^{m\times n}$ for layers $l \in H$ (omitting l for readability), noise multiplier σ^2, clipping threshold C, rank r, sparsity p, dataset S, sampling probability q, external low-rank mechanism *Decompose*, DP mechanism DP (Algorithm7), update mechanism *Update*

Output: weight matrix at step t $\boldsymbol{W}_t \in \mathbb{R}^{m\times n}$

for $l \in H$ **do**
  // $\boldsymbol{W}_{t-1}$ is randomly initialized or pre-trained with public datasets or trained with private datasets by DPSGD
  for $i \in [m]$ **do**
    $I_i \leftarrow \sum_{j=1}^{n} |W_{t-1,(i,j)}|$
  end for
  for $j \in [n]$ **do**
    $O_j \leftarrow \sum_{i=1}^{m} |W_{t-1,(i,j)}|$
  end for
  Unimportant input units $\tilde{\boldsymbol{I}} \leftarrow$ top-$(100-p)\%(\boldsymbol{I})$
  Unimportant output units $\tilde{\boldsymbol{O}} \leftarrow$ top-$(100-p)\%(\boldsymbol{O})$
  $\boldsymbol{L}_t, \boldsymbol{R}_t \leftarrow Decompose(\boldsymbol{W}_t, r)$ ▷ Use low-rankness (Alg. 6)
end for
Sample a minibatch $S_q = \{x_q\}_{x_q \in S}$ with probability q
Calculate per-sample gradients with Eq. 4.2
for $l \in H$ **do** ▷ Sparsify low-rank gradients for each layer
  for $i \in \tilde{\boldsymbol{I}}$ **do**
    for $j \in [r]$ **do**
      $\partial L_{t,(i,j)} \leftarrow 0$ ▷ Use sparsity (Eq. 4.6)
    end for
  end for
  for $j \in \tilde{\boldsymbol{O}}$ **do**
    for $i \in [r]$ **do**
      $\partial R_{t,(i,j)} \leftarrow 0$ ▷ Use sparsity (Eq. 4.7)
    end for
  end for
end for
for $l \in H$ **do**
  $\tilde{\partial}\boldsymbol{L}_t \leftarrow DP(\{\partial \boldsymbol{L}_t(x_q)\}_{q\in S_q}, \tilde{\boldsymbol{I}}, \sigma^2, C)$ ▷ Clip and Add noise (Alg. 7)
  $\tilde{\partial}\boldsymbol{R}_t \leftarrow DP(\{\partial \boldsymbol{R}_t(x_q)\}_{q\in S_q}, \tilde{\boldsymbol{O}}, \sigma^2, C)$ ▷ Clip and Add noise (Alg. 7)
  $\boldsymbol{W}_t \leftarrow Update(\boldsymbol{W}_{t-1}, \tilde{\partial}\boldsymbol{L}_t, \tilde{\partial}\boldsymbol{R}_t)$ ▷ Use off-the-shelf optimizer and Eq. 4.3
end for
return $\boldsymbol{W}_t$

4.3.3 LSG の拡張

LSG を様々な深層学習モデルアーキテクチャに対応できるように拡張を行う．ここまでは重み行列を $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ で表すことができる，つまり全結合層のみを取り扱っていた．近年は全結合層だけでなく様々な層を持つ深層学習モデルアーキテクチャが利用されているため，それらへの対応として特に利用されることの多い Attention 層と畳み込み層への拡張について述べる．

Attention 層：Attention 層は Transformer [39] を中心に利用される層構造である．d_{model} は単語の埋め込みサイズ，d_k は $\boldsymbol{K}$ のサイズ，d_v は $\boldsymbol{V}$ のサイズとし，内部で利用される重み行列を $\boldsymbol{W}^Q \in \mathbb{R}^{d_{model} \times d_k}, \boldsymbol{W}^K \in \mathbb{R}^{d_{model} \times d_k}, \boldsymbol{W}^V \in \mathbb{R}^{d_{model} \times d_v}$ とする．ここで入力を x_1, x_2 とすると，Attention 層は以下のように定式化される．

$$
\begin{aligned}
\boldsymbol{Q} &= \boldsymbol{W}^Q x_1 \\
\boldsymbol{K} &= \boldsymbol{W}^K x_2 \\
\boldsymbol{V} &= \boldsymbol{W}^V x_2 \\
\mathrm{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \mathrm{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V}
\end{aligned}
$$

$\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ はそれぞれ Query，Key，Value と呼ばれるベクトルであり，x_1 に基づく情報の取り出し方を，x_2 に基づく情報の取り出し方を，x_2 の潜在的な情報を意味したベクトルである．Attention 層はこれら 3 つのベクトルを利用し，$\boldsymbol{Q}$ と $\boldsymbol{K}$ の内積を Query と Key の関連度（重み）として $\boldsymbol{V}$ の値を取り出す操作となっている．ここで使わているパラメータに注目すると，$\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V$ という 3 つの単純な行列とわかる．そのため，全結合層と同じ操作をそれぞれの行列に対して行うことで自然に LSG による差分プライバシを実現できる．

畳み込み層：畳み込み層は画像処理を中心に広く利用されている基本的な層である．ここでは 2 次元の畳み込み層を例に扱う．LSG による畳み込み層での低ランク性とスパース性を利用した差分プライベート学習の概要を図 4.4 に示す．

まずはじめに [32] を元に畳み込み層の低ランク近似について考える．畳み込み層のカーネルが入力 $x \in \mathbb{R}^{m \times w \times h}$ を出力 $y \in \mathbb{R}^{n \times w' \times h'}$ に変換するものと考えると，$\boldsymbol{W} \in \mathbb{R}^{n \times m \times k \times k}$ と表せる．ここで，y の 2 つ目と 3 つ目の次元を固定すると（すなわち $y_{:,i,j}, i \in [0, w'] \wedge j \in [0, h']$），その値は $x^{(i,j)} \in \mathbb{R}^{m \times k \times k}$ と $\bar{\boldsymbol{W}} \in \mathbb{R}^{n \times mk^2}$ から

$y_{:,i,j} = \bar{\boldsymbol{W}} x^{(i,j)}$ と表せる．なお $\bar{\boldsymbol{W}}$ は $\boldsymbol{W}$ の出力チャネル数とカーネルサイズに関する次元を平坦化したものである．これにより式 4.1 と同様に，ランク r で低ランク近似した行列 $\boldsymbol{L}, \boldsymbol{R}$ を利用して再定義したときの順伝播は以下のように表される．

$$y_{:,i,j} = \boldsymbol{L}\boldsymbol{R} x^{(i,j)} + \bar{\boldsymbol{W}} x^{(i,j)}$$

次にチャネルプルーニング [91] で用いられているような，チャネルごとの重要度に注目する．ここでは全結合層と同じようにアクティベーションへの貢献を元に重みの絶対値を重要度の指標として利用する．関連する重みから，以下のように入力チャネルの重要度 $\boldsymbol{I}$ と出力チャネルの重要度 $\boldsymbol{O}$ を定義する．

$$I_i = \sum_{j=1}^{n} \sum_{k_i=1}^{k} \sum_{k_j=1}^{k} |W_{ijk_ik_j}|$$
$$O_j = \sum_{i=1}^{m} \sum_{k_i=1}^{k} \sum_{k_j=1}^{k} |W_{ijk_ik_j}|$$

最後は全結合層と同様に，重要度を低ランク近似された勾配行列 $\partial \boldsymbol{L}, \partial \boldsymbol{R}$ のスパース化に利用することで大域的な冗長性と局所的な冗長性を排除した差分プライベート学習を実現する．

4.4 評価実験

まず 4.4.1 項ですべての実験の設定を説明した上で，4.4.2 項で近似クエリ処理の評価を，4.4.3 項で画像処理タスクを利用したスクラッチからの学習の評価を，4.4.4 項で自然言語処理タスクを利用したファインチューニングの評価を報告する．最後に 4.4.5 項でマイクロベンチマークとして低ランク性とスパース性の兼ね合いについて評価を報告し議論を行う．

4.4.1 実験設定

4.4.1.1 近似クエリ処理タスク

ベンチマーク：近似クエリ処理ベンチマークとして，映画や出演俳優といった情報から構成される IMDb データセット[4)]とそれに対するカウントクエリワークロー

4) https://www.imdb.com

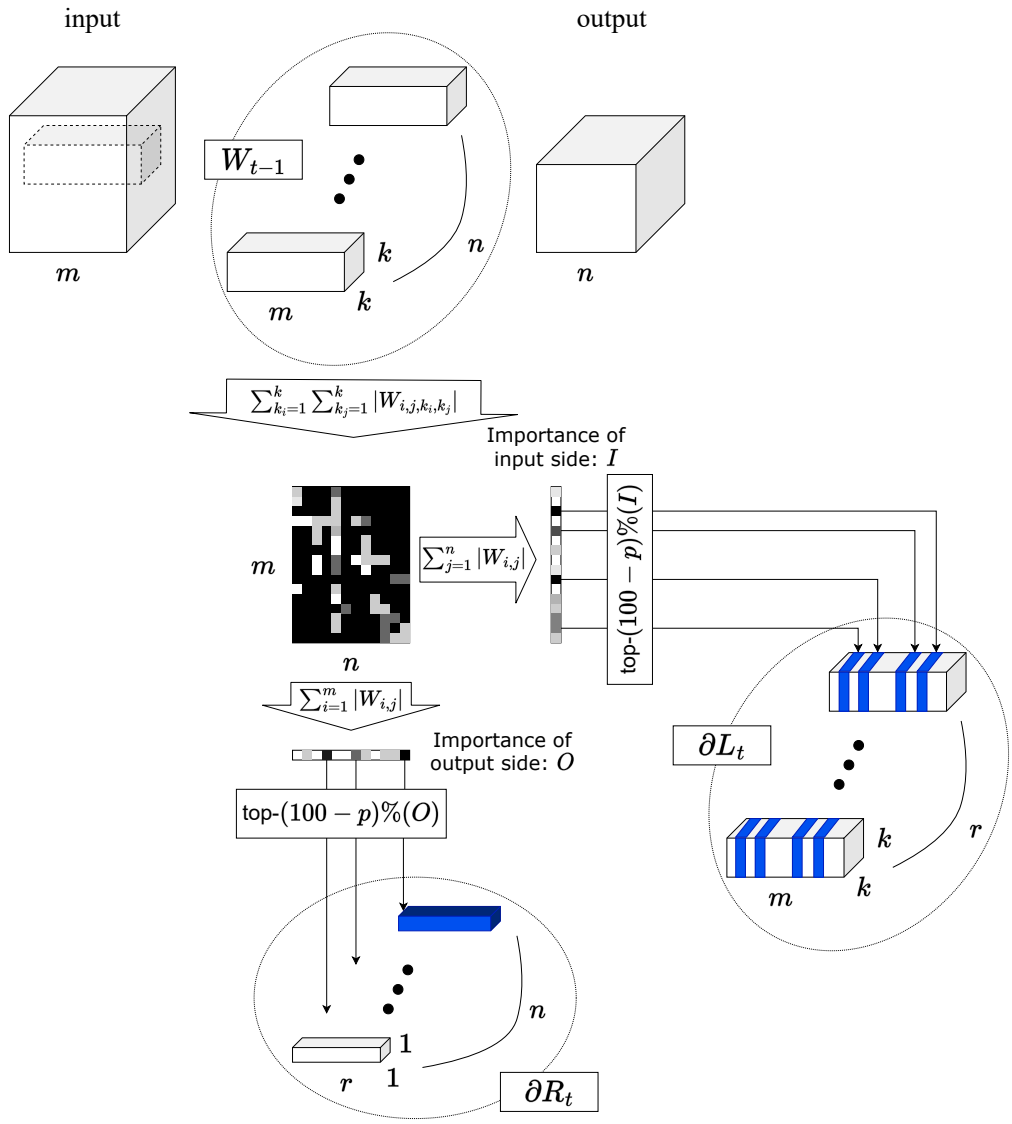


図 4.4: LSG による畳み込み層の低ランク近似とスパース化の併用例．前ステップ $t-1$ の重み行列 $\boldsymbol{W}_{t-1}$ を利用して低ランク近似された勾配行列 $\partial \boldsymbol{L}_t$ と $\partial \boldsymbol{R}_t$ をスパース化する．青色の要素はスパース化対象を意味し，現在のステップ t での更新対象から除外される．

ド（カーディナリティ推定タスク）である JOB-light [13] を利用する．評価指標には推定した件数 $\hat{\mathscr{C}}$ が真の件数 $\mathscr{C}$ から何倍離れているかを表す Q-Error [13] を利用する．Q-Error は以下のように定義される．

$$\text{Q-Error} := \frac{\max(\hat{\mathscr{C}}, \mathscr{C})}{\min(\hat{\mathscr{C}}, \mathscr{C})}$$

提案手法とベースライン：提案手法として，低ランク近似手法に RGP を利用した (1) LSG を利用する．差分プライベート学習のベースラインとしては，(2) DPSGD [31]，(3) RGP [32]，(4) Sparse DPSGD の 3 つを利用する．DPSGD は差分プライバシを満たす最も単純な手法，RGP は低ランク性を利用して DPSGD の有用性を改善した手法，Sparse DPSGD はスパース性を利用して DPSGD の有用性を改善した手法である．Sparse DPSGD は SNF-DPSGD [33] と同じスパース性を利用するが，SNF-DPSGD とは異なり，畳み込み層以外もスパース化の対象としている．また，差分プライバシを考慮しないベースラインとして，(5) N.P. を参考のため利用する．N.P. はオリジナルタスクの結果であり，差分プライベート学習の有用性上限の目安という位置付けである．

なお，DPSQL [68, 69, 70, 71] として提案されている手法は通常の SQL 処理もしくはサンプリングを前提としたものであり，通常の処理時間に加えて差分プライバシの処理時間が必要となる．例として，JOB-light のクエリはを 16CPUs（最大 2.5GHz），メモリ 64GB 環境上の PostgreSQL で実行すると平均で 8 秒かかるが，第 3 章で提案した近似クエリ処理手法では 11 ミリ秒と大きく傾向が異なる．本研究では近似クエリ処理の高速な処理速度を前提としているため，(1) から (5) の 5 手法のみの比較とし，DPSQL との比較は行わない．

モデル：近似カウントクエリ処理タスクを扱うため，第 3 章で提案した Multi-Layer Perceptron (MLP) をベースとした Denoising Autoencoder を利用する．

ハイパパラメータ：モデルで利用するハイパパラメータは基本的に第 3 章に従う．エポック数は 20，クリッピングサイズは $C = 1$ とした．プライバシ設定は $\epsilon = 8.0, \delta = 10^{-5}$ で評価を行う [5]．ランク $r \in \{2, 4, 8\}$ とスパース性 $p \in \{0, 0.1, 0.3, 0.5\}$ はチューニングを行う．

5） 4.2.1 項でも述べたようにプライバシ設定はビジネス的な要件に依存するため，一意の妥当なものに定めることはできない．本評価実験では関連研究 [31, 32, 33] の評価に用いられている設定を参考にする．

4.4.1.2　画像処理タスク

ベンチマーク：画像処理ベンチマークとして，CIFAR10 データセットでの 10 クラス分類 [95] と SVHN データセットでの 10 クラス分類を利用する．いずれも正解率をモデルの有用性として評価する．

手法：近似クエリ処理タスクと同様に，(1) LSG，(2) DPSGD [31]，(3) RGP [32]，(4) Sparse DPSGD，(5) N.P. を利用する．

モデル：画像処理タスクで一般的な畳み込みニューラルネットワークの代表として Wide ResNet [96] (WRN) を用いる．スケールファクターをレイヤ数 16，幅係数 4 とした WRN16-4 を利用する．

ハイパパラメータ：モデルで利用するハイパパラメータは基本的に [32] に従う．ただし DPSGD では，オプティマイザ，Group Normalization，Weight Standardization といった要素のうち有用なものに関しては，類似タスクで高い性能を報告している [97] や [98] に基づいて利用する．エポック数は 200，クリッピングサイズは $C = 1$ とした．プライバシ設定は $\epsilon \in \{0.8, 1.7, 3.3, 6.8\}, \delta = 10^{-5}$ で評価を行う[5]．ランク $r \in \{2, 4, 8, 16, 32\}$，スパース性 $p \in \{0, 0.1, 0.3, 0.5, 0.7\}$，バッチサイズ $bs \in \{1024, 4096\}$，学習率 $lr \in \{0.5, 1, 2, 3, 4\}$ は各ベンチマーク各プライバシ設定ごとにチューニングを行う．

4.4.1.3　自然言語処理タスク

ベンチマーク：自然言語理解ベンチマークである General Language Understanding Evaluation (GLUE) [99] を用いる．GLUE に複数設定されているタスクのうち，感情判定タスクである SST-2，質問文対判定タスクである QNLI，質問同値性判定タスクである QQP，含意関係判定タスクである MNLI[6] の 4 つを対象とする．いずれも 2 値判定のタスクであるため，その正解率をモデルの有用性として評価する．なお，全ての実験で乱数シードのみを変更した 5 回分の結果の正答率の平均と分散を報告する．

手法：近似クエリ処理タスク等と同様に，(1) LSG，(2) DPSGD [31]，(3) RGP [32]，(4) Sparse DPSGD，(5) N.P. を利用する．

モデル：自然言語処理タスクを扱うため，学習済み言語モデル RoBERTa [77] を

6)　2 つあるデータセットの平均を結果として報告する

用いる．事前学習データをパブリックなもの，追加学習データをプライベートなものとして扱った上でファインチューニングして評価を行う．RoBERTa として提供されているいくつかの学習済みモデルのうち，Attention 層と全結合層を中心に構成され約 125 万パラメータから成る RoBERTa-base を利用する．

ハイパパラメータ：モデルに関するハイパパラメータは基本的に [32] に従う．バッチサイズは2000, エポック数は20, 学習率は10^{-3}, クリッピングサイズは$C = 10$とした．プライバシ設定は，SST-2/QNLI には$\epsilon \in \{1.2, 1.6, 3.3, 6.5\}, \delta = 10^{-5}$, QQP/MNLI には$\epsilon \in \{0.9, 1.7, 3.4, 6.7\}, \delta = 10^{-6}$で評価を行う[5)]．ランク$r \in \{2, 4, 8\}$とスパース性$p \in \{0, 0.1, 0.3, 0.5\}$は各ベンチマーク各プライバシ設定ごとにチューニングを行う．

4.4.2　近似クエリ処理タスクによる差分プライベート学習の評価

評価の指針とモチベーション：Q-Error（Median・90 パーセンタイル・95 パーセンタイル・99 パーセンタイル・最大値）を有用性の指標とし，全結合層を利用したモデルのスクラッチからの学習での総合的な性能を比較する．

結果：JOB-light ベンチマークでの近似クエリ処理評価の結果を表 4.1 に示す．まず最も基本的なベースラインとなる DPSGD と比較すると，中央値から最大値まで，全ての位置で上回ることを確認した．また，〜95 パーセンタイルまでは低ランク近似のみを用いた RGP に，99 パーセンタイル〜最大値はスパース化のみを用いた Sparse DPSGD を 2〜10 倍程度上回った．一方で〜95 パーセンタイルまでは Sparse DPSGD に，99 パーセンタイル〜最大値は RGP にやや劣る結果となったが，これらの手法は残りのケースでの性能低下が大きく，総合すると LSG のほうが高い有用性を示しているといえる．

差分プライバシを利用する手法と N.P. を比較すると，特に最大値に近い位置で有用性に差が大きい傾向が確認された．クエリごとに詳しく見ると，エラーが大きいものは各差分プライバシを利用する手法や N.P. で共通することが多く，差分プライバシによって安全性の代償として有用性が全体的に下げられている傾向であり，例えば偏ったスパース化により特定の推論パターンだけ性能が低下するといった問題は確認されなかった．

表 4.1: JOB-light ベンチマーク近似クエリ処理の Q-Error ($\epsilon = 8.0$)

Method	Median	90%th	95%th	99%th	Max
LSG	3.59	16.3	77.1	464	1240
RGP	6.99	54.3	143	426	862
Sparse DPSGD	3.47	12.4	14.3	1973	6332
DPSGD	7.86	41.7	203	6963	1.6×10^4
N.P.	1.50	5.31	11.4	33.9	36.0

4.4.3 画像処理タスクによる差分プライベート学習の評価

評価の指針とモチベーション：プライバシ強度を変化させた際の正解率を有用性の指標とし，畳み込み層を利用したモデルのスクラッチからの学習での総合的な性能を比較する．

結果：各データセットでの実験結果を表 4.2 に示す．すべてのデータセット，プライバシパラメータにおいて，スパース化のみを利用する Sparse DPSGD や低ランク近似のみを利用する RGP と比較して LSG が最も高い性能を示すことが確認された．ベースラインで最も良い性能を示した RGP と比較しても，最大で 7%程度の性能向上が見られた．4.4.4 項から繰り返しになるが，低ランク近似とスパース化どちらかだけではなく，両立することが差分プライベート学習に有効であることがわかる．

新たな知見として，LSG が RGP をベースにスパース化して性能改善している一方で，スパース化のみを利用すると最も単純な DPSGD より性能が悪化するという傾向が見受けられた．これは特に畳み込み層のパラメータには局所的な冗長性が少なく，Sparse DPSGD によるスパース化が有効に働かない一方で，LSG におけるスパース化は行列分解の際の近似精度を高く（ランク r を大きく）したときのパラメータ数削減として有効に働いているからと考えられる．この傾向からも，大域的冗長性と局所的冗長性の両方を捉えるキャパシティーが重要と言える．

図 4.5 は CIFAR10 データセットでの学習曲線を示す．指標は正解率 (%) であり，プライバシパラメータを $\epsilon = 6.8, \delta = 10^{-5}$ に固定した上でランク r とスパース性 p ごとにプロットしている．各ランクごとに見ると，学習初期段階ではスパース化の効果があまり見られない一方で，学習が進むに連れてスパース化を入れたほうが性

能が向上していることがわかる．これは，スクラッチからの学習の特徴として重みをランダムな値で初期化しているため，ある程度学習が進むまで重みの絶対値を重要度として利用するスパース化に有用な情報が得られないことが原因と考えられる．学習が進むに連れて重要なパラメータと重要でないパラメータが区別されるようになり，結果として終盤はスパース化で重要でないパラメータの更新を省くことで性能が向上していると考えられる．この知見から，学習初期段階はスパース化を行わないことで学習を安定させ，ある時点からスパース化を有効にする，もしくは徐々にスパース性を上げていくことで最終的な性能を改善することが考えられるが，これは今後の課題とする．

表 4.2: WRN16-4 での各データセットの画像分類タスクの差分プライベート学習の評価．プライバシパラメータ ϵ を変化させたときの各手法の正解率 (%) を示し，太字のものは各プライバシ強度で最もスコアが高いことを表す．括弧内の値は分散を示す．N.P. の値は [32] の報告に基づく．

(a) CIFAR10 データセットでの 10 クラス分類タスクの正解率

Method	$\epsilon = 0.8$	$\epsilon = 1.7$	$\epsilon = 3.3$	$\epsilon = 6.8$
LSG	**47.4** (0.3)	**53.6** (0.1)	**60.5** (0.0)	**66.7** (0.4)
RGP	45.4 (0.8)	**53.6** (0.3)	59.8 (0.7)	65.8 (0.5)
Sparse DPSGD	37.4 (0.2)	41.8 (0.6)	47.8 (0.2)	57.3 (0.0)
DPSGD	38.4 (0.1)	43.1 (0.3)	48.6 (0.2)	57.9 (0.3)
N.P.	93.3			

(b) SVHN データセットでの 10 クラス分類タスクの正解率

Method	$\epsilon = 0.8$	$\epsilon = 1.7$	$\epsilon = 3.3$	$\epsilon = 6.8$
LSG	**81.0** (1.6)	**86.3** (0.1)	**89.7** (0.1)	**91.9** (0.1)
RGP	78.7 (1.6)	85.6 (0.1)	83.3 (1.7)	91.2 (0.1)
Sparse DPSGD	55.3 (6.2)	73.0 (11)	82.4 (1.1)	88.3 (0.1)
DPSGD	56.4 (17)	74.7 (1.3)	82.4 (0.1)	88.4 (0.0)
N.P.	97.2			

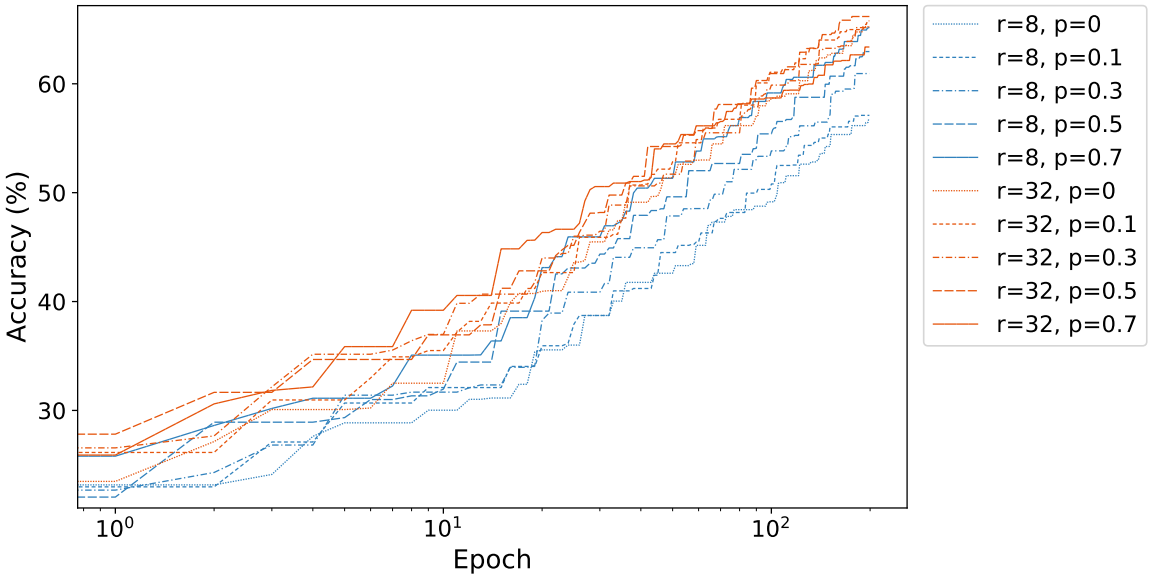


図 4.5: WRN16-4 で CIFAR10 データセットを 200 エポック差分プライベート学習したときの正解率 (%) の推移をランク r とスパース性 p ごとに比較する．同じ色は同じランクを，同じ線のスタイルは同じスパース性を示す．細かいほどスパース性は低い．$p = 0$ が RGP を，$p > 0$ が LSG を表す．プライバシパラメータはすべて $\epsilon = 6.8, \delta = 10^{-5}$ である．

4.4.4 自然言語処理タスクによる差分プライベートファインチューニングの評価

評価の指針とモチベーション：プライバシ強度を変化させた際の各タスクの正解率を有用性の指標とし，Attention 層を利用したモデルのファインチューニングでの総合的な性能を比較する．

結果：各タスクでの実験結果を図 4.6 に示す．LSG は最も性能の高いベースラインである RGP と比較して，最大 4%の性能向上が確認された．加えて，RGP だけでなく SNF-DPSGD [33] に倣いスパース性のみを利用した Sparse DPSGD と比較しても性能が向上していることから，低ランク近似とスパース化どちらかだけではなく，両立することが差分プライベート学習に有効であることがわかる．また，これらの結果から，ニューラルネットワークパラメータが本質的に低ランク性とスパース性を持ち，これらを活用することで DPSGD のクリッピングとノイズ加算の影響を減らしたと考えられる．

タスクごとのより細かい結果として，SST-2 タスクや QNLI タスクのプライバシ

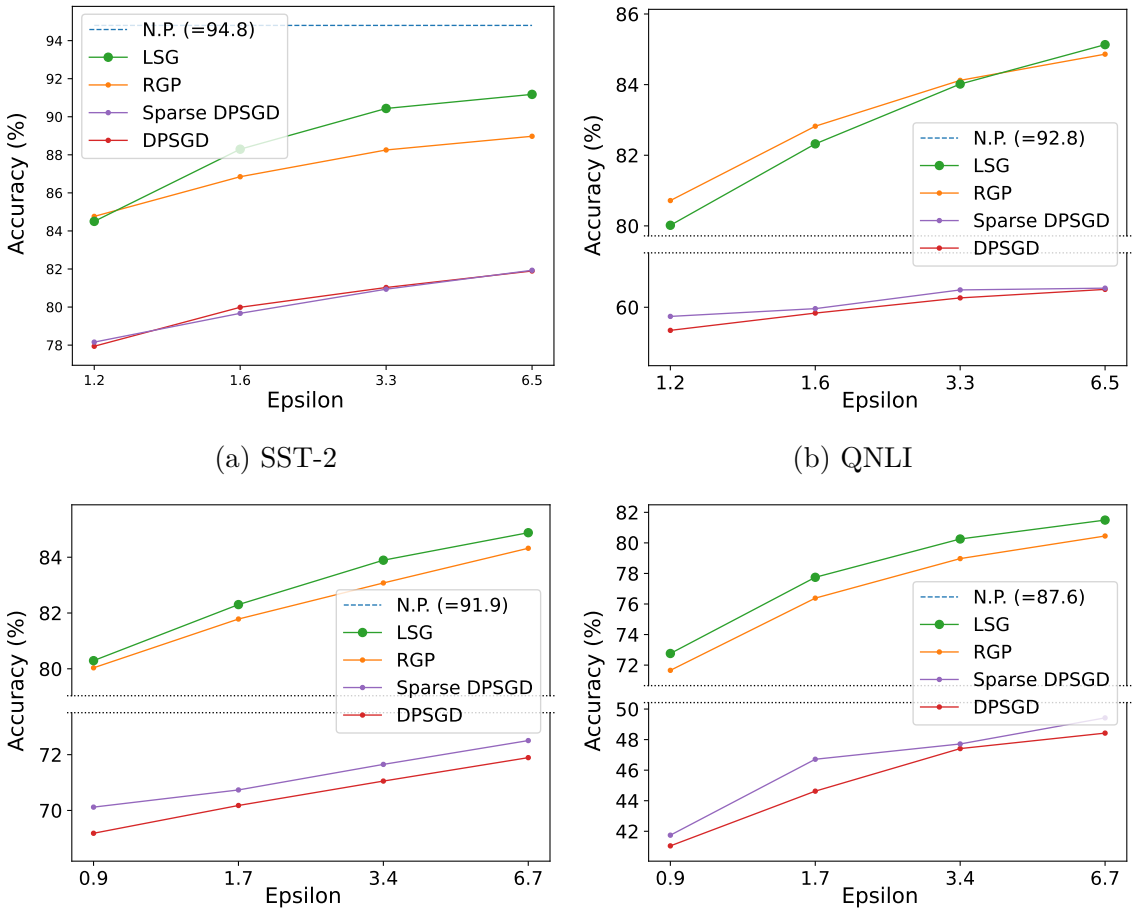
強度が高いケースで RGP と比較して同等，もしくは性能が低下するケースが確認された．クリッピングサイズ C や学習データの参照回数は固定されているため，プライバシ強度が高いケースではノイズの分散が大きくなっている．これを踏まえると，低ランク近似に加えてスパース化による更新対象パラメータ削減を行うことにより，更新されるパラメータとされないパラメータの乖離が大きくなり，ノイズの影響以上にモデルの表現力が落ちてしまったことが原因と考えられる．

4.4.5 低ランク性とスパース性の併用に関する評価

評価の指針とモチベーション：プライバシパラメータを固定し，提案手法のハイパパラメータであるランク r とスパース性 p を変化させた際の自然言語処理タスクの正解率から，これらのパラメータの兼ね合いを確認する．

なお，表記のパラメータ以外は 4.4.4 項と同じ設定である．

結果：ϵ を固定し，ランク r とスパース性 p を変化させたときの各タスクの差分プライベートファインチューニングの結果を表 4.3 に示す．より小さいランク r とより大きいスパース性 p はより更新対象パラメータを削減することを意味する．また，ランク $r = full$ は行列分解を利用しない場合の結果を示す．既に 4.4.4 項の結果でも述べたように，低ランク近似とスパース化の両立により性能が向上することが確認された．一方で極端に更新対象パラメータ数を削減するような場合（小さいランク r と大きいスパース性 p を同時に選択するような場合），逆に性能が低下する傾向が確認された．これは，更新対象パラメータ数の削減がクリッピングやノイズの影響を減らして性能向上に寄与する一方で，学習に必要なパラメータが不足しだすと性能低下を引き起こすというトレードオフによるものと考えられる．このトレードオフは，適切なランクとスパース性を選択したときに最も良い性能を示していることからも推察される．別の観点として，最も良いパラメータを選択したい場合，ランク r またはスパース性 p のいずれかを固定して探索しても他方のパラメータと組み合わせたときに最良の結果を得られるとは限らないことがわかる（例えば表 4.3a で，$r = full$ を固定して最も良い p を探索すると $p = 0$ となるが，実際には $p = 0.5(r = 8)$ が最適解となる）．このことは，ランクとスパース性の積で表現される削減パラメータ数だけでは性能の傾向は決まらず，適用先に合わせて適切なランクとスパース性を仮定する必要があることが示唆される．適切なランクとスパース性の効率的な探索は今後の課題とする．



(a) SST-2 (b) QNLI

(c) QQP (d) MNLI

図 4.6: GLUE ベンチマークによる RoBERTa モデルの差分プライベートファインチューニングの評価．プライバシパラメータ ϵ を変化させたときの各手法の正解率 (%) を示す．なお，N.P. の値は [77] の報告に基づく．いくつかの図では見やすさのため N.P. を省略している．

表 4.3: GLUE ベンチマークによる RoBERTa モデルの差分プライベートファインチューニングの評価．プライバシパラメータ ϵ を固定し，ランク r とスパース性 p を変化させたときの各手法の正解率 (%) を示し，太字のものは各タスクで最もスコアが高いことを表す．括弧内の値は分散を示す．$p > 0 \wedge r \leq 8$ が LSG を，$p = 0 \wedge r \leq 8$ が RGP を，$full$ が Sparse DPSGD を表す．

(a) SST-2, $\epsilon = 3.3$

		Rank r			
		2	4	8	$full$
Sparsity p	0	85.4 (1.7)	87.9 (0.1)	88.3 (0.5)	81.0 (0.1)
	0.1	87.3 (0.2)	88.6 (0.3)	89.0 (0.6)	80.9 (0.3)
	0.3	88.3 (0.5)	89.6 (0.1)	89.9 (0.2)	80.5 (0.0)
	0.5	89.4 (0.1)	89.8 (0.1)	**90.4** (0.2)	80.3 (0.1)

(b) QQP, $\epsilon = 3.4$

		Rank r			
		2	4	8	$full$
Sparsity p	0	82.8 (0.1)	82.8 (0.1)	83.1 (0.2)	71.1 (0.6)
	0.1	83.3 (0.0)	83.3 (0.1)	83.3 (0.2)	71.2 (0.1)
	0.3	**83.9** (0.0)	83.7 (0.0)	83.8 (0.0)	71.7 (0.1)
	0.5	83.8 (0.0)	83.6 (0.2)	83.6 (0.0)	71.5 (0.2)

4.5 おわりに

本章では，ニューラルネットワークパラメータの低ランク性とスパース性に基づき更新対象パラメータを適切に選択することで，差分プライバシを満たし有用性の高いモデルを獲得する学習手法を提案した．第 2 章と第 3 章で提案した深層学習を用いたカーディナリティ推定技術と併用でき，データ処理の高速化に加えてプライバシの保護を実現する．ユニットレベル，チャネルレベルといったニューラルネットワークの入出力の構造に基づいた重要度を定義することで，これまで排他的であった低ランク性とスパース性の併用を実現した．実装を行い，様々なモデル構

造，タスクや学習パターンで評価したところ，低ランク性とスパース性のどちらかだけを利用した手法と比較して性能向上が確認された．提案手法は具体的な低ランク近似手法に依らないため，今後差分プライベート学習のための新たな低ランク近似手法が提案された際には，更なる性能向上への貢献が期待できる．

一方で 4.4 節の各実験結果から，提案手法の性能を最大限発揮するためにはランクとスパース性の調整が必要であることも確認された．データやタスクによって適切なパラメータが異なることから，その効率的な探索が今後の課題である．また，適切なスパース性は学習段階によって異なる可能性も確認されたため，スパース性は適応的に変化させることも考えられる．

第 5 章　結論

5.1　本研究のまとめ

本研究では，複雑なスキーマを持つ大規模データ処理の高速化とプライバシ保護の実現を目指した．データ処理の高速化という観点では，既存のデータ処理と近似処理，いずれのアプローチにも大きく寄与することが知られるカーディナリティ推定と呼ばれる技術に注目し，（第 2 章）高速かつ精度を向上させること，（第 3 章）複雑なスキーマを持つ大規模データに対応すること，という 2 段階でカーディナリティ推定手法の改良に取り組んだ．またプライバシ保護という観点では，厳密な指標である差分プライバシという技術に注目し，第 4 章（第 2 章）や（第 3 章）に適用可能な差分プライベート学習手法に取り組んだ．

第 2 章では，大規模データ処理の高速化のため，高速で安定したカーディナリティ推定手法を提案した．カーディナリティ推定とは指定された条件を満たすデータがデータベース内に何件存在するかを推定するタスクである．カーディナリティ推定を改善することで，大きく分けて 2 つのアプローチでデータ処理高速化に貢献する．1 つはデータベースシステムのクエリオプティマイザでの利用である．カーディナリティ推定の性能を向上させることで，クエリオプティマイザがより適切な実行プランを選択できるようになり，結果として既存のクエリ処理を高速化できる．もう 1 つは近似クエリ処理での利用である．カーディナリティ推定の定義は条件に一致する件数を近似的に求める処理と等価であり，その他の近似的なクエリ処理のプリミティブとしても用いられる．そのため，カーディナリティ推定の性能を向上させることで，結果に厳密性が必要ない処理で高速化と高精度化が可能である．既存のカーディナリティ推定手法はいずれも実世界データにそぐわない仮定をおくため推定性能が低い [13]，チューニングが困難なパラメータに依存している部分が大きく推定性能が安定しない [23] といった問題がある．提案手法ではデータの分布を捉えるために深層学習技術の 1 つである Denoising Autoencoder に注目し

た．Denoising Autoencoder をベースに，任意の属性を条件に任意の属性の分布を推定できるように構築，学習する．これにより動的な順での推論が可能となり，結果として安定性の高いカーディナリティ推定を実現した．

第 3 章では，複雑なスキーマを持つデータ処理のため，第 2 章で提案した密度推定器を複数利用することでスケーラブルな結合カーディナリティ推定手法を提案した．結合カーディナリティ推定とは結合を条件に含むカーディナリティ推定である．クエリオプティマイザの機能の中でも特に重要とされる結合順最適化で利用される．結合順最適化は結合の個数が多いほど利用されるため，特にスキーマサイズが大きいデータで顕著になる．そのため，カーディナリティ推定によるクエリオプティマイザの改善（データ処理高速化）という点では，特に大規模スキーマでの結合カーディナリティ推定が重要となる．結合カーディナリティ推定は重要なタスクであるものの，既存のカーディナリティ推定手法は推定性能や大規模スキーマへの対応に課題が残る．提案手法では，スキーマの構造に基づく軽量な分割を行い，分割されたスキーマごとに密度推定器を学習する．各密度推定器の扱うテーブル数は入次数+1 であるため，スキーマサイズが増えても小規模な密度推定器で十分な推定性能を発揮できる．学習時の密度推定器間は独立であるためすべて並列に学習が可能である一方で，推論時は密度推定器間で推論結果を再利用することで密度推定器間に跨る相関を考慮した推定を実現する．クエリオプティマイザで求められるような小規模なクエリに対する推定性能が高いという特徴もあり，実験的にクエリオプティマイザを通したクエリ処理性能の向上を確認した．

第 4 章では，第 2 章，第 3 章で提案したカーディナリティ推定手法に適用可能な差分プライベート学習手法を提案した．近似クエリ処理時のプライバシ保護として，識別困難性に基づく安全性の指標である差分プライバシに注目した．深層学習で差分プライバシを実現する手法としては DPSGD [31] が広く知られている．DPSGD は，学習時に勾配の制限とノイズ加算を行うことで，結果として得られるモデルはどのような使い方をしても差分プライバシによる安全性が担保される．一方で DPSGD で学習したモデルは，勾配の制限とノイズの影響でモデル本来の有用性（例えば画像分類モデルであればその分類精度）が大きく低下することが知られている．この問題を改善するため，更新対象パラメータ数を削減することで，安全性を落とすことなく有用性を改善する既存手法がある [32, 33]．しかしながら，既存手法で利用できている冗長性は大域的なものや局所的なもののいずれかと限定的

で，有用性の改善も小さい．提案手法では，ニューラルネットワークパラメータの 2 つの冗長性に注目し，差分プライバシによる安全性と有用性の両立を図った．パラメータを行列として扱い，大域的冗長性を捉えるために低ランク近似を，局所的冗長性を捉えるためにスパース化を行う．これらにより，安全性を落とすことなく勾配クリッピングやノイズの影響を軽減して有用性を向上させた．実際に第 3 章や一般的な機械学習タスク，様々な学習方法で有用性の改善を確認した．

5.2 今後の展望

本研究では基本的かつ汎用的な要素技術を提案した．今後は提案した要素技術の統合や実用化に必要となる課題に取り組むことが考えられる．特に実用化に向けて残っている重要な課題は以下の 4 点である．

1 つ目はデータベース更新への対応である．データに更新が行われると，当然データの分布も変化する．一度学習したモデルだけでは，いずれ学習したデータの分布とデータベース内のデータの分布に乖離が生まれ，推定性能が低下する．そのため，テータの更新に追従するためのインクリメンタルな学習方法，もしくはモデルの入れ替え方法を検討する必要がある．注意すべき点として，データの更新には削除があるということがあげられる．深層学習での追加データの学習 (Continual Learning) [100, 101, 102, 103] は比較的研究が進んでいるが，一度学習したデータを削除，つまり多数のパラメータで保持している情報から忘却させる方法 (Unlearning) [104, 105, 106, 107] は発展途上で制約が多い．通常のデータベースシステムではごく当たり前に行われる削除をどのように再現するかは非常に重要な課題と考えられる．

2 つ目は第 3 章の提案の特徴を利用した結合順序最適化アルゴリズムの開発である．提案手法には，結合カーディナリティ推定を途中で止めると，その時点での部分的な結合カーディナリティが推定できるという特徴があった．結合順最適化は分割統治的に行われるため，中間データを再利用したカーディナリティ推定が有用であると考えられる．これによりカーディナリティ推定コストを抑制でき，結果としてクエリ処理時間の短縮が期待できる．

3 つ目はカウントクエリ以外の近似クエリ処理への応用である．本研究中では近似クエリ処理はカウントクエリのみを扱っていた．しかしながら，実用上では平均

値クエリ，総和クエリ，グループ化クエリなど様々な集約演算との併用などが要求される．データ分布の獲得までは第 3 章で達成できているため，今後はこのデータ分布から様々なクエリに対応できるような推論手法を検討する必要がある．

4 つ目はカーディナリティ推定に特化した差分プライベート学習である．第 4 章にてカーティナリティ推定に適用できる差分プライベート学習手法を提案したが，多くの深層学習モデルに対応した汎用的なものである．第 2 章，第 3 章で提案したモデルは入力データの一部にマスクをしてデータの分布を学習するものだった．ここで利用しているのは入力データの部分的な情報 $P(A_1, \ldots, A_i \mid A_{i+1}, \ldots, A_n)(\neq P(A_1, \ldots, A_n))$ であるため，1 入力データから得られる情報を過大評価し，クリッピングやノイズ加算が過剰になっていることが考えられる．このようなモデル学習時の特徴を考慮することで，パラメータ数削減以外のアプローチで差分プライベートなカーディナリティ推定を改善することが考えられる．

謝辞

本研究の遂行にあたり，学部 4 年から博士前期課程，そして博士後期課程の合わせて 6 年半にもわたり指導教員として研究内容や研究への向き合い方に関するご指導を賜りました 鬼塚真教授 に深く感謝を申し上げます．並びに，日々様々な視点から適切なご助言を賜りました 荒瀬由紀准教授，肖川准教授，佐々木勇和助教 に深く感謝を申し上げます．本論文の第 4 章を共に推進し，議論していただいた LINE 株式会社 Data Science センター，ML Privacy チーム 高橋翼氏，リュウセンペイ氏 に厚く御礼申し上げます．本論文の執筆にあたり多くの貴重なご助言を賜りました 大阪大学大学院情報科学研究科マルチメディア工学専攻 原隆浩教授，藤原融教授 に心より感謝申し上げます．講義，学生生活を通して，多くの研究の糧を賜りました大阪大学大学院情報科学研究科マルチメディア工学専攻 下條真司教授 松下康之教授 に心より感謝申し上げます．

大阪大学大学院情報科学研究科マルチメディア工学専攻ビッグデータ工学講座 鬼塚研究室に所属して 6 年半を過ごす中で出会い，研究に関する議論だけでなく雑談などにも付き合っていただいた先輩・同期・後輩の皆様にも感謝致します．

最後に，これまでの進路，そして博士後期課程への進学まで快く尊重し，温かい支援と理解をいただきました家族と友人に感謝致します．

参考文献

[1] 伊藤竜一, 佐々木勇和, 肖川, 鬼塚真. 非自己回帰モデルによる高速で安定したカーディナリティ推定. 情報処理学会論文誌データベース, Vol. 15, No. 3, pp. 36–49, 2022.

[2] Ryuichi Ito, Chuan Xiao, and Makoto Onizuka. Robust Cardinality Estimator by Non-Autoregressive Model. *Proceedings of the Fifth Workshop on Software Foundations for Data Interoperability*, 2021. Reproduced with permission from Springer Nature.

[3] Ryuichi Ito, Seng Pei Liew, Tsubasa Takahashi, Yuya Sasaki, and Makoto Onizuka. Scaling Private Deep Learning with Low-Rank and Sparse Gradients. *arXiv preprint arXiv:2207.02699*, 2022.

[4] 伊藤竜一, リュウセンペイ, 高橋翼, 佐々木勇和, 鬼塚真. 低ランク近似を介した選択的パラメータ更新による差分プライベート学習. 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論文集, 2022.

[5] 川本孝太朗, 伊藤竜一, 佐々木勇和, 肖川, 鬼塚真. 結合カーディナリティ推定の中間結果を利用した結合順最適化. 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論文集, 2022.

[6] 伊藤竜一, 佐々木勇和, 肖川, 鬼塚真. Non-Autoregressive モデルによる高速で安定したカーディナリティ推定. 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論文集, 2021.

[7] 伊藤竜一, 佐々木勇和, 肖川, 鬼塚真. 非自己回帰モデルによる安定したカーディナリティ推定. 情報処理学会第 83 回全国大会論文集, 2021.

[8] 総務省. 情報通信白書. 2017.

[9] Baaziz Abdelkader and Quoniam Luc. How to use Big Data technologies to optimize operations in Upstream Petroleum Industry. *World Petroleum Congress*, 2014.

[10] Microsoft. How innovative oil and gas companies are using big data to outmaneuver the competition. *White paper*, 2014.

[11] Zoi Kaoudi and Jorge-Arnulfo Quiané-Ruiz. Unified Data Analytics: State-of-

the-art and Open Problems. *VLDB (tutorial)*, 2022.

[12] IBM. Data-driven healthcare organizations use big data analytics for big gains. *White paper*, 2017.

[13] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. How good are query optimizers, really? *Proceedings of the International Conference on Very Large Data Bases*, Vol. 9, No. 3, pp. 204–215, 2015.

[14] 寺田雅之. 差分プライバシーとは何か. システム／制御／情報, Vol. 63, No. 2, pp. 58–63, 2019.

[15] Latanya Sweeney. K-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 10, No. 5, p. 557–570, 2002.

[16] Josep Domingo-Ferrer, Luisa Franconi, Sarah Giessing, Eric Schulte Nordholt, Keith Spicer, Peter-Paul de Wolf, and Anco Hundepool. *Statistical Disclosure Control*. Wiley Series in Survey Methodology, 2012.

[17] Cynthia Dwork. Differential Privacy. *Proceedings of the EATCS International Colloquium on Automata, Languages and Programming*, pp. 1–12, 2006.

[18] Rachel Cummings and Deven Desai. The Role of Differential Privacy in GDPR Compliance. *Proceedings of the FATREC Workshop*, 2018.

[19] The PostgreSQL Global Development Group. PostgreSQL: Documentation. https://www.postgresql.org/docs/, 2022.

[20] Viswanath Poosala, Yannis E. Ioannidis, Peter J. Haas, and Eugene J. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. *Proceedings of the IEEE International Conference on Data Engineering*, 1996.

[21] Max Heimel, Martin Kiefer, and Volker Markl. Self-tuning, gpu-accelerated kernel density models for multidimensional selectivity estimation. In *Proceedings of the International Conference on Management of Data*, pp. 1477–1492, 2015.

[22] Martin Kiefer, Max Heimel, Sebastian Breß, and Volker Markl. Estimating join selectivities using bandwidth-optimized kernel density models. *Proceedings of the International Conference on Very Large Data Bases*, Vol. 10, No. 13, pp. 2085–2096, 2017.

[23] 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. *Proceedings of the International Conference on Very Large Data Bases*, Vol. 13, No. 3, pp. 279–292, 2019.

[24] Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen, and Ion Stoica. NeuroCard: One Cardinality Estimator for All Tables. *Proceedings of the International Conference on Very Large Data Bases*, Vol. 14, No. 1, pp. 61–73, 2020.

[25] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. DeepDB: Learn from Data, not from Queries! *Proceedings of the International Conference on Very Large Data Bases*, Vol. 13, No. 7, pp. 992–1005, 2019.

[26] Rong Zhu, Ziniu Wu, Yuxing Han, Kai Zeng, Andreas Pfadler, Zhengping Qian, Jingren Zhou, and Bin Cui. FLAT: Fast, Lightweight and Accurate Method for Cardinality Estimation. *Proceedings of the International Conference on Very Large Data Bases*, Vol. 14, No. 9, pp. 1489–1502, 2021.

[27] Shohedul Hasan, Saravanan Thirumuruganathan, Jees Augustine, Nick Koudas, and Gautam Das. Deep learning models for selectivity estimation of multi-attribute queries. *Proceedings of the International Conference on Management of Data*, pp. 1035–1050, 2020.

[28] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. *Proceedings of the Conference on Innovative Data Systems Research*, 2019.

[29] Andreas Kipf, Dimitri Vorona, Jonas Müller, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, Thomas Neumann, and Alfons Kemper. Estimating Cardinalities with Deep Sketches. *Proceedings of the IEEE International Conference on Data Engineering*, pp. 1937–1940, 2019.

[30] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek Narasayya, and Surajit Chaudhuri. Selectivity estimation for range predicates using lightweight models. *Proceedings of the International Conference on Very Large*

Data Bases, Vol. 12, No. 9, pp. 1044–1057, 2019.

[31] Edgar Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew Myers, Shai Halevi, Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. *Proceedings of the Conference on Computer and Communications Security*, pp. 308–318, 2016.

[32] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. Large Scale Private Learning via Low-rank Reparametrization. *Proceedings of the International Conference on Machine Learning*, 2021.

[33] Zelun Luo, Daniel J. Wu, Ehsan Adeli, and Li Fei-Fei. Scalable Differential Privacy with Sparse Network Finetuning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5057–5066, 2021.

[34] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. *Proceedings of the International Conference on Machine Learning*, 2008.

[35] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized Denoising Auto-Encoders as Generative Models. *Proceedings of the Conference on Neural Information Processing Systems*, 2013.

[36] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, Vol. 14, No. 3, pp. 462–467, 1968.

[37] Lucas Woltmann, Claudio Hartmann, Maik Thiele, Dirk Habich, and Wolfgang Lehner. Cardinality estimation with local deep learning models. *Proceedings of the Workshop in Exploiting AI Techniques for Data Management*, pp. 1–8, 2019.

[38] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked Autoencoder for Distribution Estimation. *Proceedings of the International Conference on Machine Learning*, 2015.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Proceedings of the Conference on Neural Information Processing Systems*, 2017.

[40] Hoifung Poon and Pedro Domingos. Sum-Product Networks: A New Deep Architecture. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[41] Benigno Uria, Iain Murray, and Hugo Larochelle. A Deep and Tractable Density Estimator. In *Proceedings of the International Conference on Machine Learning*, p. 467—475, 2014.

[42] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *Proceedings of the Conference on Neural Information Processing Systems*, 2019.

[43] Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. Random Sampling over Joins Revisited. *Proceedings of the International Conference on Management of Data*, pp. 1525–1539, 2018.

[44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2019.

[45] Jiatao Gu, James Bradbury, Caiming Xiong, Victor O K Li, and Richard Socher. Non-Autoregressive Neural Machine Translation. *Proceedings of the International Conference on Learning Representations*, 2018.

[46] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-Predict: Parallel Decoding of Conditional Masked Language Models. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2019.

[47] Cheng Guo and Felix Berkhahn. Entity Embeddings of Categorical Variables. *arXiv preprint arXiv:1604.06737*, 2016.

[48] State of New York. Vehicle, Snowmobile, and Boat Registrations. https://catalog.data.gov/dataset/vehicle-snowmobile-and-boat-registrations, 2019.

[49] IMDb. Internet Movie Data Base. https://www.imdb.com, 1990.

[50] naru-project. naru. https://github.com/naru-project/naru, 2020.

[51] neurocard. neurocard. https://github.com/neurocard/neurocard, 2020.

[52] DataManagementLab. deepdb-public. https://github.com/DataManagementLab/deepdb-public, 2020.

[53] wuziniu. FSPN. https://github.com/wuziniu/FSPN, 2021.

[54] learnedsystems. Cardinality Estimation Benchmark. https://github.com/learnedsystems/CEB, 2021.

[55] parimarjan. pg_hint_plan. https://github.com/parimarjan/pg_hint_plan, 2020.

[56] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep Sets. *Proceedings of the Conference on Neural Information Processing Systems*, 2017.

[57] Ji Sun, Jintao Zhang, Zhaoyan Sun, Guoliang Li, and Nan Tang. Learned cardinality estimation: A design space exploration and a comparative evaluation. *Proceedings of the International Conference on Very Large Data Bases*, Vol. 15, No. 1, p. 85–97, 2021.

[58] Hugo Larochelle and Iain Murray. The Neural Autoregressive Distribution Estimator. *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011.

[59] Kyoungmin Kim, Jisung Jung, In Seo, Wook-Shin Han, Kangwoo Choi, and Jaehyok Chong. Learned Cardinality Estimation: An In-depth Study. *Proceedings of the International Conference on Management of Data*, pp. 1214–1227, 2022.

[60] MariaDB. MariaDB Knowledge Base. https://mariadb.com/kb/, 2022.

[61] Yuxing Han, Ziniu Wu, Peizhi Wu, Rong Zhu, Jingyi Yang, Liang Wei Tan, Kai Zeng, Gao Cong, Yanzhao Qin, Andreas Pfadler, Zhengping Qian, Jingren Zhou, Jiangneng Li, and Bin Cui. Cardinality Estimation in DBMS: A Comprehensive Benchmark Evaluation. *Proceedings of the International Conference on Very Large Data Bases*, 2022.

[62] Viktor Radke, Bernhard Leis, Andrey Gubichev, Alfons Kemper, and Thomas Neumann. Cardinality Estimation Done Right: Index-Based Join Sampling. *Proceedings of the Conference on Innovative Data Systems Research*, 2017.

[63] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. Wander Join: Online Aggregation via Random Walks. *Proceedings of the International Conference on Management of Data*, pp. 615–629, 2016.

[64] David Lopez-Paz, Philipp Hennig, and Bernhard Schölkopf. The Randomized

Dependence Coefficient. *Proceedings of the Conference on Neural Information Processing Systems*, 2013.

[65] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, Vol. 9, No. 3-4, pp. 211–407, 2014.

[66] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Theory of Cryptography Conference*, pp. 265–284. Springer, 2006.

[67] Apple. Differential Privacy Overview. https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf, 2016.

[68] Frank D McSherry. Privacy integrated queries. *Proceedings of the International Conference on Management of Data*, pp. 19–30, 2009.

[69] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating Data to Sensitivity in Private Data Analysis. *Proceedings of the International Conference on Very Large Data Bases*, 2012.

[70] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. Differentially Private SQL with Bounded User Contribution. *Proceedings of the International Symposium on Privacy Enhancing Technologies*, 2019.

[71] Johes Bater, Yongjoo Park, Xi He, Xiao Wang, and Jennie Rogers. SAQE: Practical Privacy-Preserving Approximate Query Processing for Data Federations. *Proceedings of the International Conference on Very Large Data Bases*, Vol. 13, No. 12, pp. 2691–2705, 2020.

[72] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 3–18. IEEE, 2017.

[73] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *Proceedings of the USENIX Security Symposium*, pp. 267–284, 2019.

[74] Milad Nasr, Shuang Song, Abhradeep Thakurta, Nicolas Papenot, and Nicholas

Carlini. Adversary instantiation: Lower bounds for differentially private machine learning. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 866–882. IEEE, 2021.

[75] Chuan Guo, Brian Karrer, Kamalika Chaudhuri, and Laurens van der Maaten. Bounding training data reconstruction in private (deep) learning. *Proceedings of the International Conference on Machine Learning*, 2022.

[76] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially Private Fine-tuning of Language Models. *Proceedings of the International Conference on Learning Representations*, 2021.

[77] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019.

[78] Cynthia Dwork, Nitin Kohli, and Deirdre Mulligan. Differential Privacy in Practice: Expose your Epsilons! *Journal of Privacy and Confidentiality*, Vol. 9, No. 2, 2019.

[79] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science*, pp. 464–473. IEEE, 2014.

[80] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *Proceedings of the IEEE Global Conference on Signal and Information Processing*, pp. 245–248. IEEE, 2013.

[81] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization. *Proceedings of the Conference on Neural Information Processing Systems*, 2019.

[82] Jian Xue, Jinyu Li, and Yifan Gong. Restructuring of deep neural network acoustic models with singular value decomposition. *Interspeech*, pp. 2365–2369, 2013.

[83] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On Compressing Deep Models by Low Rank and Sparse Decomposition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 67–76, 2017.

[84] Sridhar Swaminathan, Deepak Garg, Rajkumar Kannan, and Frederic Andres. Sparse low rank factorization for deep neural network compression. *Neurocomputing*, Vol. 398, pp. 185–196, 2020.

[85] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both Weights and Connections for Efficient Neural Networks. *Proceedings of the Conference on Neural Information Processing Systems*, 2015.

[86] Jose M Alvarez and Mathieu Salzmann. Learning the Number of Neurons in Deep Networks. *Proceedings of the Conference on Neural Information Processing Systems*, 2016.

[87] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *Proceedings of the International Conference on Learning Representations Workshop Track*, 2017.

[88] Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *Proceedings of the International Conference on Learning Representations*, 2018.

[89] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-Specific Batch Normalization for Unsupervised Domain Adaptation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7346–7354, 2019.

[90] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning Filters for Efficient ConvNets. *Proceedings of the International Conference on Learning Representations*, 2017.

[91] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning Structured Sparsity in Deep Neural Networks. *Proceedings of the Conference on Neural Information Processing Systems*, 2017.

[92] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[93] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The Lottery Ticket Hypothesis for Pre-trained BERT Networks. *Proceedings of the Conference on Neural Information Processing Systems*, 2020.

[94] Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. Super Tickets in Pre-Trained Language Models: From Model Compression to Improving Generalization. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2021.

[95] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[96] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *Proceedings of the British Machine Vision Conference*, 2016.

[97] Soham De, Leonard Berrada, Jamie Hayes, Samuel L Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*, 2022.

[98] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Micro-Batch Training with Batch-Channel Normalization and Weight Standardization. *arXiv preprint arXiv:1903.10520*, 2019.

[99] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *Proceedings of the EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 2018.

[100] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. *Proceedings of the Conference on Neural Information Processing Systems*, 2017.

[101] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, Vol. 114, No. 13, pp.

3521–3526, 2017.

[102] Woo-Young Kang and Byoung-Tak Zhang. Continual Learning with Generative Replay via Discriminative Variational Autoencoder. *Proceedings of the Conference on Neural Information Processing Systems*, 2018.

[103] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, Vol. 113, pp. 54–71, 2019.

[104] Haonan Yan, Xiaoguang Li, Ziyao Guo, Hui Li, Fenghua Li, and Xiaodong Lin. ARCANE: An Efficient Architecture for Exact Machine Unlearning. *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 4006–4013, 2022.

[105] Ji Gao, Sanjam Garg, Mohammad Mahmoody, and Prashant Nalini Vasudevan. Deletion Inference, Reconstruction, and Compliance in Machine (Un)Learning. *Proceedings of the International Symposium on Privacy Enhancing Technologies*, 2022.

[106] Yuji Suga, Kouichi Sakurai, Xuhua Ding, Kazue Sako, Quoc Phong Nguyen, Ryutaro Oikawa, Dinil Mon Divakaran, Mun Choon Chan, and Bryan Kian Hsiang Low. Markov Chain Monte Carlo-Based Machine Unlearning: Unlearning What Needs to be Forgotten. *AsiaCCS, doi = 10.1145/3488932.3517406, eprint = 2202.13585, pages = 351–363,*, 2022.

[107] Bo Liu, Qiang Liu, and Peter Stone. Continual Learning and Private Unlearning. *Proceedings of the Conference on Lifelong Learning Agents*, 2022.