DEIM Forum 2021 E11-2

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

伊藤　竜一† 佐々木勇和† 肖　川† 鬼塚　真†

† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1-5

E-mail: †{ito.ryuichi,sasaki,chuanx,onizuka}@ist.osaka-u.ac.jp

あらまし　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．一方で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精度が大きく左右され推論速度も遅いという問題がある．本稿では，属性間の依存関係を Non-Autoregressive モデルで学習し，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．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．

キーワード　カーディナリティ推定，機械学習

1　はじめに

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の高速化がより求められている．そのため，効率の良い実行プランの選択，つまりクエリオプティマイザの性能は非常に重要である．

クエリオプティマイザは主にコストのモデル化・プラン探索・カーディナリティ推定の 3 つの要素から成り立つ．1 つ目のコストのモデル化とは，データベースシステムの各処理に掛かるコストの係数を体系立てる要素である．2 つ目のプラン探索とは，考えられる多数の実行プランから有用なものを探索する要素である．3 つ目のカーディナリティ推定とは，選択演算に基づくクエリ処理結果のカーディナリティを推定する要素である．クエリオプティマイザ全体としては，実行プランにおける各演算子のカーディナリティの推定値や物理操作をコストモデルに与えることで実行プラン全体のコストを見積もり，推定コストの低い実行プランの探索を行う．この際，3 つの要素のうちカーディナリティ推定がクエリ応答性能に最も大きな影響力を持つことが Leis ら [1] によって報告されている．

そこで本稿ではカーディナリティ推定に注目する．カーディナリティ推定は長年研究されているが [2–11]，実際のデータベースシステムでは $10^4 - 10^8$ 倍といったオーダーで推論のエラーが発生しクエリ応答性能を低下させる一因となっている [1]．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．近年ではこの課題を解決すべく，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 [3–11] これらはアプローチで 2 つに大別される．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 [6–8] 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 [9–11]Autoregressive モデル [12,13] や Sum-Product Networks [14] 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，大規模データの分布をカーディナリティ推定に利用可能な形で学習する必要があり，その安定性が課題となっている．例えば Autoregressive 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性能が安定しないという問題がある．

本稿では Non-Autoregressive モデルによってデータを学習し，高速で安定したカーディナリティ推定を実現する手法の提案を行う．Non-Autoregressive モデルの密度推定としての性質を用いることで，属性ごとの分布だけでなくリレーション内の全属性間の相関関係を捉える．学習時に推論できる属性の順序が固定されてしまう Autoregressive モデルとは異なり，Non-Autoregressive モデル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，Autoregressive 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．複数のベンチマークで Q-Error [1] を指標として評価したところ，95 パーセンタイルでは PostgreSQL と比較して 1/35 程度に，また，既存の Autoregressive モデル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 倍の高速化に成功した．

2 章でカーディナリティ推定の定式化と Non-Autoregressive モデルについて説明を行い 3 章で提案手法を詳説する．4 章で提案手法の評価を行い 5 章で既存研究との関連について述べ，6 章で本稿をまとめる．

2 事前準備

2.1 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述語の条件を満たす同時確率から求まることに注目する．リレーション R が属性 $\boldsymbol{A} = \{A_1, .., A_n\}$ と タプル $\boldsymbol{t} = \{t_1, ..., t_m\}$ から成り，条件となる述語の演算を $\boldsymbol{\theta}(t) \rightarrow \{0, 1\}$ とする．セレクティビティ $\mathcal{S}$ 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．

$$\mathcal{S}(\boldsymbol{\theta}) = \frac{|\{t \in R \mid \boldsymbol{\theta}(t) = 1\}|}{|R|} \tag{1}$$

タプル t は属性 $\{A_1, ..., A_n\}$ の要素 $\{a_1 \in A_1, ..., a_n \in A_n\}$ の組み合わせで構成されることから，

$$\mathcal{S}(\boldsymbol{\theta}) = \sum_{a_1 \in A_1} ... \sum_{a_n \in A_n} \boldsymbol{\theta}(a_1, ..., a_n) P(a_1, ..., a_n) \tag{2}$$

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 $P(A_1, ..., A_n) \approx \prod_n P(A_i)$ 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求め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 [1]．

本稿ではそのようなヒューリスティックに基づく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，

$$P(A_1, ..., A_n) = \begin{array}{l} P(A_1) P(A_2 \mid A_1) \\ ... P(A_n \mid A_{n-1}, ..., A_1) \end{array} \tag{3}$$

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とに留意する．また， $\boldsymbol{\theta}$ が conjunctive である（$\boldsymbol{\theta} = \theta_1 \otimes \theta_2 \otimes ... \otimes \theta_n$，$\theta_i(a_i) \rightarrow \{0, 1\}$）とすると，セレクティビティは以下の形としても導ける．

$$\mathcal{S}(\boldsymbol{\theta}) = \sum_{a_1 \in A_i} ... \sum_{a_n \in A_n} \boldsymbol{\theta}(a_1, ..., a_n) P(a_1, ..., a_n) \tag{4}$$

$$= \begin{array}{l} \sum_{a_1 \in A_1} \theta_1(a_1) P(a_1) \sum_{a_2 \in A_2} \theta_2(a_2) P(a_2 \mid A_1) \\ ... \sum_{a_n \in A_n} \theta_n(a_n) P(a_n \mid A_{n-1}, ..., A_1) \end{array} \tag{5}$$

ここで $\sum_{a_i \in A_i} P(a_i \mid \cdot)$ は明らかに 1 であるが，$\sum_{a_i \in A_i} \theta_i(a_i) P(a_i \mid \cdot)$ は $[0, 1]$ の範囲を取るスカラー値となり $\boldsymbol{\theta}$ の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，

$$\mathcal{C}(\boldsymbol{\theta}) = |R| \cdot \mathcal{S}(\boldsymbol{\theta}) \tag{6}$$

$$= |R| \sum_{a_1 \in A_i} ... \sum_{a_n \in A_n} \boldsymbol{\theta}(a_1, ..., a_n) P(a_1, ..., a_n) \tag{7}$$

$$= \begin{array}{l} |R| \sum_{a_1 \in A_1} \theta_1(a_1) P(a_1) \sum_{a_2 \in A_2} \theta_2(a_2) P(a_2 \mid A_1) \\ ... \sum_{a_n \in A_n} \theta_n(a_n) P(a_n \mid A_{n-1}, ..., A_1) \end{array} \tag{8}$$

と表せる．

2.2 複数リレーションへの拡張

データベース内に複数のリレーションが存在することはごく一般的であり，複数のリレーション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，リレーションが複数存在する場合は単純にタプルとして取り出せないため，ここまで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リレーションに結合グラフを定義し完全外部結合を取ったものを仮想的な単一リレーション，つまりユニバーサルリレーションと見なすことで，結合を含む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場合，クエリに含まれないリレーションの影響を受けるため直接カーディナリティの推論に利用できない．更に，多数のリレーションの完全外部結合を取るとタプル数・属性数共に非常に多くなり，全てを一度に扱うことは困難であるため，ユニバーサルリレーションのマテリアライズは避けなければならないという課題がある．

Yang らは，Hilprecht らの完全外部結合から内部結合へ復元する手法 [11] と Zhao らの Exact Weight Join Sampling [15] を組み合わせることによって，前述の課題を解決する結合サンプリング手法と推論アルゴリズムを提案している [10]．この手法ではユニバーサルリレーションスキーマでサンプリングされたタプルでモデルを学習する一方で，推論時はクエリに含まれるリレーションのみに絞り込む条件を加え，更に，クエリに含まれないリレーションとの結合による増分を打ち消す操作を行う．これにより，クエリに含まれるリレーションのみから成る仮想的な単一リレーションに基づく確率分布が得られ，適切なカーディナリティ推定が可能となる．具体的には，まず予めどの属性でどのリレーション同士の結合を行うかを結合グラフとして定義する [1]．そして完全外部結合を取る際にリレーションマーカー（$Marker_R$）と Join Counts（$JoinCounts_R$）を属性として追加する（式 11）．リレーションマーカーとは，リレーションごとに完全外部結合に含まれているかを示す真偽値である．もう 1 つの Join Counts とは，各タプルが結合グラフに従って結合する際に対応するタプル数である．外部結合であるため，たとえ結合先リレーション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．結合サンプリングは結合グラフのルートから順に行われる．このとき，Join Counts 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

1：同じリレーション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．

回数行い，得られたリレーションマーカーと Join Counts 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リレーションの影響を打ち消す 2 つの操作を行う．まず，クエリに含まれる結合に応じてリレーションのマーカーが $True$ であることをクエリの条件として追加する（式 12）．内部結合であれば両リレーションのマーカーが，外部結合であれば起点となるリレーションのマーカーが対象となる．これにより，クエリに含まれるリレーションが存在するタプルに絞り込まれる．加えて同時確率を求める際に，クエリに含まれないリレーションの Join Counts でスケーリングを行う（式 13）．これにより，クエリに含まれないリレーション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リレーションのみに基づくものとなる．

$$\boldsymbol{S} = \{R \in \boldsymbol{R} \mid R \ used \ in \ q\} \tag{9}$$

$$\boldsymbol{T} = \{R \in \boldsymbol{R} \mid R \ not \ used \ in \ q\} \tag{10}$$

$$\boldsymbol{A'} = R.\boldsymbol{A} \oplus Marker_R \oplus JoinCounts_R \ , \forall \boldsymbol{R} \tag{11}$$

$$\boldsymbol{\theta'} = \boldsymbol{\theta} \otimes Marker_S \ is \ True \ , \forall \boldsymbol{S} \tag{12}$$

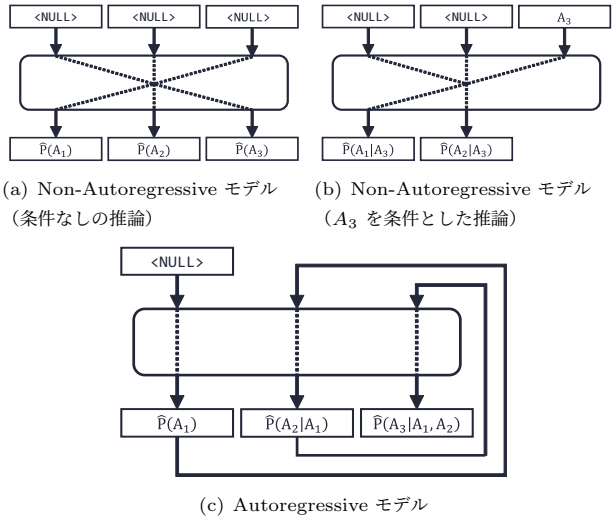
$$\mathcal{S}(\boldsymbol{\theta'}) = \frac{\sum_{a'_1 \in A'_i} \cdots \sum_{a'_n \in A'_n} \boldsymbol{\theta}(a'_1, ..., a'_n) P(a'_1, ..., a'_n)}{\prod_{T \in \boldsymbol{T}} JoinCounts_T} \tag{13}$$

$$= \frac{\sum_{a'_1 \in A'_1} \theta_1(a'_1) P(a'_1) \sum_{a'_2 \in A'_2} \theta'_2(a'_2) P(a'_2 \mid A'_1) \cdots \sum_{a'_n \in A'_n} \theta'_n(a'_n) P(a'_n \mid A'_{n-1}, ..., A'_1)}{\prod_{T \in \boldsymbol{T}} JoinCounts_T} \tag{14}$$

2.3 Non-Autoregressive モデル

Non-Autoregressive モデル (図 1(a)，1(b)) とは，元々自然言語処理で高速な文章生成を実現するために提案された手法である [16, 17]．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ターゲットとして学習を行う．これにより，単語シーケンスの一部を条件として入力として残りの任意の単語の並列推論を可能としている．同様の処理に用いられることが多い Autoregressive モデルと比較して，推論が同時並列的に可能という特徴を持つ (図 1) より一般的な視点では次のような違いを生む．Autoregressive モデルの場合は学習時に順序が固定されるため，例えば $A_1, ..., A_n$ 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 $\hat{P}(A_1), \hat{P}(A_2 \mid A_1), ..., \hat{P}(A_n \mid A_{n-1}, ..., A_1)$ に限られるが，Non-Autoregressive モデル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全体の推論も可能である．

主な実装として，Autoregressive モデルの 1 つである Transformer [13] をベースとしたモデルが挙げられる [17]Self-Attention 時のマスクを外すことによって Non-Autoregressive モデルとしての性質が実現される．



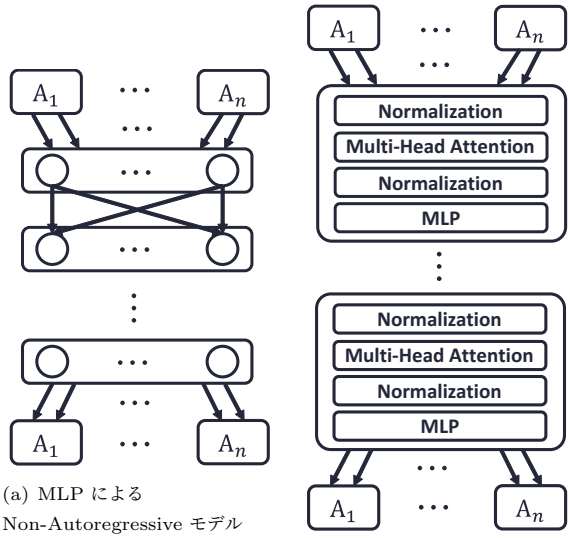
(a) Non-Autoregressive モデル（条件なしの推論）　(b) Non-Autoregressive モデル（A_3 を条件とした推論）

(c) Autoregressive モデル

図 1　Non-Autoregressive モデルと Autoregressive モデル

3 提案手法

提案手法では Non-Autoregressive モデルをコアとしてカーディナリティ推定を実現する．なお，Non-Autoregressive の性質を持つ任意のモデルを利用可能であるが，本稿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によるモデル (図 2(a)) と Attention による高度な推論が可能な Transformer [13] をベースとしたモデル (図 2(b)) 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リレーションが含まれる場合は，2.2 節で述べたように，全てのリレーション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．



(a) MLP による Non-Autoregressive モデル　(b) Transformer による Non-Autoregressive モデル [2]

図 2　提案手法で利用する Non-Autoregressive モデル

学習フェーズ，推論フェーズそれぞれについて説明する．

3.1 学習フェーズ

カーディナリティ推定に利用可能な Non-Autoregressive モデルによる密度推定の学習について述べる．学習はタプル単位で行われる．具体的には，図 3 のようにリレーションからタプルを取り出し，一部属性をマスクしたものを入力，マスクしていないものをターゲットとする．そしてマスクした属性に関するロスを利用して学習する．マスクをランダムに選択することで，Autoregressive とは異なり，任意の属性を条件とした残り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とに異なることから，埋め込みは属性ごとに行う．

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．2.1 節で述べたように，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 $\sum_{a_i \in A_i} P(a_i \mid \cdot)$ 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 $\boldsymbol{A} = \{A_1, ..., A_n\}$, $\boldsymbol{\theta} = \theta_1 \otimes \theta_3$ 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が求まる．

$$
\begin{aligned}
\mathcal{C}(\boldsymbol{\theta}) &= |R| \sum_{a_1 \in A_i} ... \sum_{a_n \in A_n} \boldsymbol{\theta}(a_1, ..., a_n) \hat{P}(a_1, ..., a_n) \\
&= |R| \sum_{a_1 \in A_1} \theta_1(a_1) \hat{P}(a_1) \sum_{a_2 \in A_2} \theta_2(a_2) \hat{P}(a_2 \mid A_1) \\
&\qquad ... \sum_{a_n \in A_n} \theta_n(a_n) \hat{P}(a_n \mid A_{n-1}, ..., A_1) \\
&= |R| \sum_{a_1 \in A_1} \theta_1(a_1) \hat{P}(a_1) \sum_{a_3 \in A_3} \theta_3(a_3) \hat{P}(a_3 \mid A_1) \\
&= |R| \sum_{a_3 \in A_3} \theta_3(a_3) \hat{P}(a_3) \sum_{a_1 \in A_1} \theta_1(a_1) \hat{P}(a_1 \mid A_3)
\end{aligned}
$$

より具体的な確率分布の扱いについて述べる．提案手法では，Autoregressive モデルを利用する先行研究である Naru [9] で提案された Progressive Sampling を Non-Autoregressive モデルに拡張する．この Progressive Sampling はモンテカルロ法により等号条件だけでなく範囲条件を可能とする．Autoregressive モデルを利用した直接的な範囲条件の推論では適合する全ての組み合わせ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に選択した一部のみを推論しその平均を近似解として扱う．このとき，一様ランダムに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，範囲条件下でのデータの偏りを考慮した選択となり，より少ないサンプルサイズで真の値に近づく．なお，この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．これは厳密なものであるが，Progressive Sampling では直前までの推論結果を以降の推論に利用すること，また，その際にサンプリングを行うため，乗法定理による式展開の順序，つまり推論する属性順が性能に影響する．特に，正確度や精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．Autoregessive 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 $!|A|$ 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 Non-Autoregressive モデルを利用する提案手法では任意の属性順で推論できる．そこで，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で推論を行う（このヒューリスティックによる影響は 4.2 節に示す）．手順を Algorithm 1 に示す．

Algorithm 1 Non-Autoregressive モデルを利用したカーディナリティ推定アルゴリズム

Input: $\mathcal{M}, \boldsymbol{\theta}, \boldsymbol{A}, N, |R|$
Output: $\hat{cardinality}$

1: **procedure** ESTIMATE($\mathcal{M}, \boldsymbol{\theta}, \boldsymbol{AttrsInPreds}$))
2: Initialize *inputs* w/ nulls
3: $prob \leftarrow 1.0$
4: **foreach** $A_i \in \boldsymbol{AttrsInPreds}$ **do**
5: $\hat{dist}_{A_i} \leftarrow \mathcal{M}(inputs)$ ▷ Forward
6: $\hat{dist'}_{A_i} \leftarrow \{\theta_i(a_i)\hat{dist}_{a_i} \mid a_i \in A_i\}$ ▷ Filter by θ
7: $prob \leftarrow prob * \sum_{a_i \in A_i} \hat{dist'}_{A_i}$
8: $a_i \leftarrow$ SAMPLING($\hat{dist'}_{A_i}$)
9: $inputs[A_i] \leftarrow$ ENCODE(a_i)
10: **end for**
11: **return** *prob*
12: **end procedure**
13:
14: $AttrsInPreds \leftarrow \{A \in \boldsymbol{A} \mid A$ USED IN $\boldsymbol{\theta}\}$
15: **foreach** $i \in N$ **do** ▷ Batched in practice
16: $probs[i] \leftarrow$ ESTIMATE($\mathcal{M}, \boldsymbol{\theta}$,
17: SORTBYFILTEREDDOMAINSIZE($AttrsInPreds$))
18: **end for**
19: $\hat{selectivity} \leftarrow$ MEAN($probs$)
20: $\hat{cardinality} \leftarrow \hat{selectivity} * |R|$
21: **return** $\hat{cardinality}$

この Progressive Sampling を拡張したアルゴリズムは Non-Autoregressive モデル $\mathcal{M}$，クエリの述語 $\boldsymbol{\theta}$，リレーション内の属性 $\boldsymbol{A}$，サンプルサイズ N，リレーション内の総タプル数 $|R|$ 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す（14 行目）．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（ESTIMATE）を N 回行う（15-18 行目）． ESTIMATE は Non-Autoregressive モデル $\mathcal{M}$,

2：Decoder 部分のみを利用.

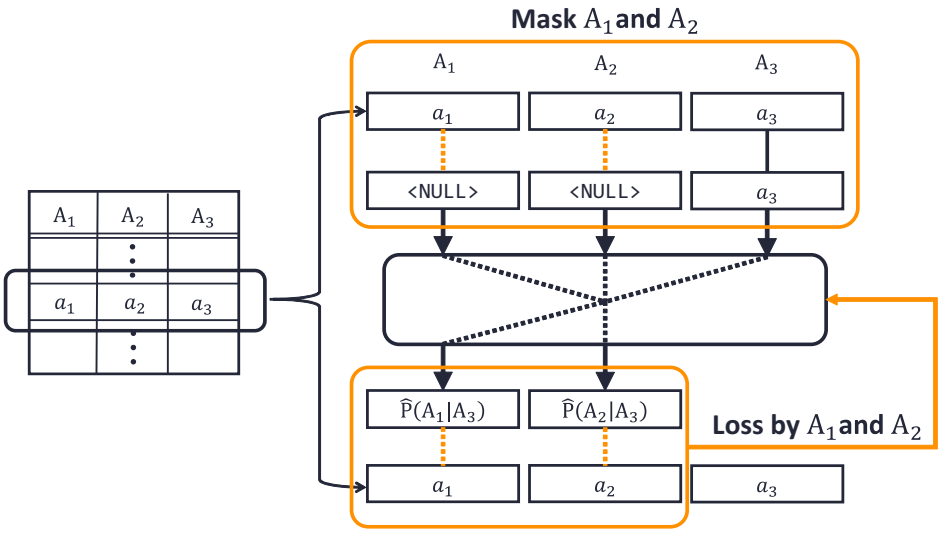


図 3　Non-Autoregressive モデルのカーディナリティ推定のための学習

クエリの述語 $\boldsymbol{\theta}$，属性 $\boldsymbol{A}$ を受け取る．このとき属性は述語を適用したときのドメインが狭い順に並べ変えて与える．初めにモデルへの入力と同時確率を初期化する（2-3 行目）．その後与えられた属性順に $\mathcal{M}$ で推論を行う（5 行目）．推論された確率分布 $\hat{dist}_{A_i}$ に述語を適用し，その総和と $prob$ の積を取り（7 行目），更に確率分布に基づくサンプリングを行い属性 A_i の要素 a_i を得る（8 行目）．この要素をエンコードし次の入力とする（9 行目）．述語の対象となっている全属性でこれらの操作を繰り返すことで $prob$ がセレクティビティの推論値となる（11 行目）．最後に，得られた N 個のセレクティビティの推論値の平均を最終的なセレクティビティの推論値とし，総タプル数 $|R|$ との積をカーディナリティの推論値として返却する（19-21 行目）．なお実際には，15-18 行目のループは行列計算としてバッチ化される．

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．4.1 節では総合的な性能を，4.2 節では提案手法で注目している属性順やサンプルサイズが性能に与える影響を扱う．

提案手法として，Non-Autoregressive モデルに多層パーセプトロン（MLP）を利用するものと Transformer [13] [2] を利用するものを PyTorch を用いて実装した．評価には 2 つのベンチマークを利用した．1 つはニューヨークの乗り物に関するデータセットである DMV [18] 上で動作する人工的に生成された 2000 個のクエリから成るワークロードである．DMV が単一リレーションであるため結合はなく，4〜10 個の等号条件と範囲条件から成る．もう 1 つは俳優、映像エンターテイメントやビデオゲーム等に関するデータセットである IMDb 上で動作する 70 個のクエリから成る Join Order Benchmark [1]（JOB-light [6]）である．多数リレーションの自然結合があり，複数の等号条件と範囲条件から成る．各データセットの統計的な概要は表 1 に示す通りである．比較手法には，実際のデータベースシステムである PostgreSQL11.8・Autoregressive モデルを用いる先行研究の Naru [9] [19]/NeuroCard [10] [20] [3] を用いた．なお，先行研究と提案手法では Nvidia Tesla T4 GPU を利用した実行環境，学習に利用する結合サンプル，学習時のバッチサイズやエポック数を等しくしている．また，モデルで共通部分となるハイパーパラメータも等しくなるように設定している．評価指標には，式 15 で表される Q-Error [1] を用いた．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．

$$\text{Q-Error} := \frac{max(\hat{Cardinality}, Cardinality)}{min(\hat{Cardinality}, Cardinality)} \tag{15}$$

表 1　テータセット概要

データセット名	テーブル数	行数	属性数	（結合）サンプルサイズ
DMV	1	11.6M	11	11．6M（全件）
IMDb	16	4〜36M	2〜17	10M

4.1　総合的な評価と考察

この節では各手法の 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）と応答性能の総合的な評価結果を報告し，結果に基づく考察を行う．先行研究，提案手法共にモンテカルロ法におけるサンプルサイズを 1000 で固定して比較をした．DMV での結果を表 2 に，JOB-light での結果を表 3 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，推論時ドメイン考慮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を表す．

表 2 から，Autoregressive モデルを利用する Naru は MADE

3：NeuroCard は Naru を複数リレーションに拡張した手法である．
いずれも MADE 実装が主たる提案だが Transformer 実装も提案されている．

表 2　DMV での Q-Error と平均応答時間

手法（実装・属性順）	Median	90th	95th	99th	Max	平均応答時間（ms）
PostgreSQL	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
Naru (MADE・逆順)	60.8	$5.1 \cdot 10^2$	$7.5 \cdot 10^2$	$2.0 \cdot 10^3$	$4.2 \cdot 10^5$	10.4
Naru (Transformer・正順)	1.09	2.63	5.34	$9.7 \cdot 10^5$	$9.9 \cdot 10^5$	99.5
Naru (Transformer・逆順)	1.18	10.8	$2.0 \cdot 10^2$	$1.0 \cdot 10^3$	$2.1 \cdot 10^4$	$1.0 \cdot 10^3$
提案手法 (MLP・推論時ドメイン考慮順)	1.05	1.30	1.56	2.60	49.0	5.76
提案手法 (Transformer・推論時ドメイン考慮順)	1.04	1.22	1.41	2.33	49.0	54.3

表 3　JOB-light での Q-Error と平均応答時間

手法（実装・属性順）	Median	90th	95th	99th	Max	平均応答時間（ms）
PostgreSQL	7.44	$1.6 \cdot 10^2$	$8.4 \cdot 10^2$	$3.0 \cdot 10^3$	$3.5 \cdot 10^3$	3.88
NeuroCard (MADE・正順)	1.79	9.00	19.5	36.5	43.2	$1.6 \cdot 10^2$
NeuroCard (MADE・逆順)	6.04	72.3	$1.2 \cdot 10^2$	$3.0 \cdot 10^2$	$4.0 \cdot 10^2$	$1.6 \cdot 10^2$
提案手法 (MLP・推論時ドメイン考慮順)	3.14	25.0	60.5	$7.7 \cdot 10^2$	$2.2 \cdot 10^3$	46.5
提案手法 (Transformer・推論時ドメイン考慮順)	2.41	11.8	16.8	$4.1 \cdot 10^3$	$1.3 \cdot 10^4$	$1.9 \cdot 10^3$

実装・Transformer 実装共に属性順によって数倍から数十倍以上性能が変化してしまい，適切な属性順を選択しないと性能が安定しないことが分かる．一方提案手法では，このような性能のばらつきが発生することなく，Naru のピーク性能となっている正順で推論を行う MADE 実装と同等程度の性能を達成した．多数の自然結合を含みより複雑なベンチマークの結果である表 3 からも，提案手法が多くのクエリに対して安定して性能が高いことが分かる．しかしながら，提案手法では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確認された．この事象は NeuroCard では見受けられなかった．同一の結合サンプルを学習しているにも関わらずこのような差異が見られた原因として，提案手法のモデルがサンプルに過学習していることが考えられる．カーディナリティ推定のためのモデルの特徴として，前述した DMV での実験のように全件学習する場合は過学習が問題にならないという特徴がある一方で，サンプリングした一部のタプルだけを学習する場合は過学習を抑制する手法を組み合わせることで改善の余地があると思われる．

全体として，先行研究と比較して推論や埋め込みの回数を削減できたことにより，同規模の先行研究のモデルを利用した手法と比較して 2〜3 倍程度の高速化に成功した．一方で，簡易なヒストグラムに基づく PostgreSQL と比較すると 2 倍以上応答時間が大きくなった．原因として，モデルのパラメータ数が多く推論コストが高くなっていることや推論時の GPU 利用によるオーバーヘッドが考えられる．パラメータ数のチューニングによる推論コスト削減や CPU を利用した推論によりこの差を縮められる可能性がある．

4.2　属性順と安定性の評価と考察

この節ではモンテカルロ法によるサンプリングを用いた推論時の属性順による影響について評価結果を報告し，結果に基づく考察を行う．ベンチマークには DMV を利用し，4.1 節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 MADE 実装を比較対象とする．評価指標には 4.1 節と同じく，直接的な性能を示す Q-Error と平均応答時間を用いる．推論順とサンプルサイズを変化させたときの結果を表 4 に示す．

表 4 から，Naru と提案手法いずれの場合においても，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．ただし単純なトレードオフではなく，サンプルサイズが 1000 以下では Q-Error が大きく変化する一方で応答時間が横ばいに近くなる現象が Transformer 実装の提案手法以外で見受けられた．この要因として，サンプルサイズが十分に小さい場合，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っていることが考えられる．サンプルサイズが小さい場合で更なる高速化を行うにはエンジニアリングによる定数コスト削減が求められるだろう．別の観点として，提案手法の同じ実装では，主たる提案である推論時ドメイン考慮順は正順と比較して，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顕著である．また，Naru と比較しても同様の傾向を示している．言い換えると，モンテカルロ法による推論を，述語を適用した際のドメインが大きくなる順序で行うことで，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いうことである．この順序は推論時に動的に決まるものであり，Autoregressive モデルを利用している Naru では実現が困難である．一方でサンプルサイズとして十分大きいと思われる 10000 の場合，正順で固定したほうが Q-Error を抑えられる傾向も見受けられた．ドメインのサイズに基づいた結果，サンプル内での多様性が減り，サンプル平均が母平均から乖離してしまった可能性がある．提案手法では推論順序を動的に変更できることから，サンプルサイズに応じて順序を動的に変更することでより安定した推定となる可能性がある．

5　関連研究

機械学習を用いてカーディナリティ推定を行う関連研究につ

表 4　DMV で推論順とサンプルサイズを変化させたときの Q-Error と応答時間

手法（実装・属性順・サンプルサイズ）	Median	90th	95th	99th	Max	平均応答時間（ms）
Naru (MADE・正順・10)	1.06	1.32	1.70	23.1	$1.3 \cdot 10^5$	7.32
Naru (MADE・正順・100)	1.04	1.22	1.45	2.42	$3.6 \cdot 10^2$	6.72
Naru (MADE・正順・1000)	1.04	1.19	1.32	2.00	8.00	10.3
Naru (MADE・正順・10000)	1.03	1.18	1.29	2.00	8.00	$1.2 \cdot 10^2$
提案手法 (MLP・推論時ドメイン考慮順・10)	1.06	1.33	1.63	3.24	$1.2 \cdot 10^2$	5.29
提案手法 (MLP・推論時ドメイン考慮順・100)	1.06	1.30	1.57	2.99	49.0	5.44
提案手法 (MLP・推論時ドメイン考慮順・1000)	1.05	1.30	1.56	2.60	49.0	5.76
提案手法 (MLP・推論時ドメイン考慮順・10000)	1.05	1.30	1.56	2.60	49.0	33.2
提案手法 (MLP・正順・10)	1.08	1.43	1.78	19.1	$1.3 \cdot 10^5$	4.95
提案手法 (MLP・正順・100)	1.06	1.36	1.55	3.00	$3.7 \cdot 10^2$	5.35
提案手法 (MLP・正順・1000)	1.06	1.33	1.51	2.43	49.0	5.57
提案手法 (MLP・正順・10000)	1.06	1.33	1.50	2.37	49.0	33.7
提案手法 (Transformer・推論時ドメイン考慮順・10)	1.04	1.26	1.46	3.00	$3.2 \cdot 10^2$	28.2
提案手法 (Transformer・推論時ドメイン考慮順・100)	1.04	1.23	1.42	2.43	49.0	27.5
提案手法 (Transformer・推論時ドメイン考慮順・1000)	1.04	1.22	1.41	2.33	49.0	54.3
提案手法 (Transformer・推論時ドメイン考慮順・10000)	1.04	1.22	1.41	2.33	49.0	$5.4 \cdot 10^2$
提案手法 (Transformer・正順・10)	1.06	1.36	1.70	34.1	$2.1 \cdot 10^3$	27.9
提案手法 (Transformer・正順・100)	1.05	1.25	1.50	3.00	$4.0 \cdot 10^2$	27.5
提案手法 (Transformer・正順・1000)	1.04	1.22	1.40	2.01	49.0	54.9
提案手法 (Transformer・正順・10000)	1.04	1.22	1.39	2.01	49.0	$5.4 \cdot 10^2$

いて述べる．これはワークロードを学習するアプローチとデータを学習するアプローチに大別される．

5.1　ワークロードを学習するアプローチの手法

Kipf らはワークロードをリレーション・結合・述語の 3 要素の組み合わせとして扱い，Multi-Set Convolutional Network [21] 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 [6,7] 未知ワークロードの推定も可能な汎化性能を持つと報告されている．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 [8] 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，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，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 [22] 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．

5.2　データを学習するアプローチの手法

Yang らは MADE [12] や Transformer [13] といった Autoregressive 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 [9, 10] 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 Autoregressive モデルとして MADE [12] 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 [23]．同程度の性能であることが報告されている．Hilprecht らは Sum-Product Networks [14] をデータベース向けに拡張した Relational Sum-Product Networks によるカーディナリティ推定手法を提案している [11]．データの傾向を行方向と列方向に分解し，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得られるデータの分布からカーディナリティを算出する．カーディナリティ推定だけでなく近似クエリ処理も可能な手法である．

6　結　　論

本稿では Non-Autoregressive モデル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．Non-Autoregressive モデル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，コールドスタート問題がなく安定したカーディナリティ推定を実現した．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，95 パーセンタイルでは PostgreSQL と比較して 1/35 程度に，また，既存の Autoregressive モデル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．

今後の展望として，Naru の既定値を参考に設定している提案手法のハイパーパラメータのチューニングとその自動化が挙げられる．評価実験から分かるように提案手法はより少ないサンプルサイズでも性能が安定するため，改善の余地があると思われる．また，より複雑なベンチマークやデータベースに更新がある場合の評価実験や推論時の CPU 利用による応答時間の

改善が考えられる．

謝　　辞

この成果は，国立研究開発法人新エネルギー・産業技術総合開発機構（ＮＥＤＯ）の委託業務の結果得られたものです．

文　　献

[1] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. How good are query optimizers, really? *Proceedings of the VLDB Endowment*, Vol. 9, No. 3, pp. 204–215, 2015.

[2] Viswanath Poosala, Yannis E. Ioannidis, Peter J. Haas, and Eugene J. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. *Proceedings of the ICDE*, 1996.

[3] 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.

[4] Max Heimel, Martin Kiefer, and Volker Markl. Self-tuning, gpu-accelerated kernel density models for multidimensional selectivity estimation. In *Proceedings of the SIGMOD*, p. 1477–1492, New York, NY, USA, 2015. Association for Computing Machinery.

[5] Martin Kiefer, Max Heimel, Sebastian Breß, and Volker Markl. Estimating join selectivities using bandwidth-optimized kernel density models. *Proceedings of the VLDB Endowment*, Vol. 10, No. 13, pp. 2085–2096, 2017.

[6] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. *Proceedings of the CIDR*, 2019.

[7] 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 ICDE*, pp. 1937–1940, 2019.

[8] Lucas Woltmann, Claudio Hartmann, Maik Thiele, Dirk Habich, and Wolfgang Lehner. Cardinality estimation with local deep learning models. *Proceedings of the aiDM*, pp. 1–8, 2019.

[9] 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 VLDB Endowment*, 2019.

[10] Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen, and Ion Stoica. NeuroCard: One Cardinality Estimator for All Tables. *arXiv*, 2020.

[11] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. DeepDB: Learn from Data, not from Queries! *Proceedings of the VLDB Endowment*, 2019.

[12] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked Autoencoder for Distribution Estimation. *PMLR*, 2015.

[13] 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 NIPS*, 2017.

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

[15] Zhuoyue Zhao, Robert Christensen, Feifei Li, Xiao Hu, and Ke Yi. Random Sampling over Joins Revisited. *Proceedings of the SIGMOD*, pp. 1525–1539, 2018.

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

[17] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-Predict: Parallel Decoding of Conditional Masked Language Models. *EMNLP*, 2019.

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

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

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

[21] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep Sets. *Proceedings of the NIPS*, 2017.

[22] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek Narasayya, and Surajit Chaudhuri. Selectivity estimation for range predicates using lightweight models. *Proceedings of the VLDB Endowment*, Vol. 12, No. 9, pp. 1044–1057, 2019.

[23] David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, Hung Q Ngo, Shohedul Hasan, Saravanan Thirumuruganathan, Jees Augustine, Nick Koudas, and Gautam Das. Deep Learning Models for Selectivity Estimation of Multi-Attribute Queries. *Proceedings of the ICDE*, pp. 1035–1050, 2020.