A Compact Index Structure with Approximately Searching for a FIB Entry

樋口俊介

情報ネットワーク学専攻 情報流通プラットフォーム講座

背景と研究目標

■パケット転送

- Forwarding Information Base (FIB) から宛先IPアドレスに対して最も長く一致するIPプレフィクスを探索
- FIBはコンパクトさと高速な検索が重要

■学習型インデックス[1]

- Key-Value Store(KVS)を実現する**コンパクトなデータ構造**
- キーと格納位置の関係を回帰モデルで学習し、再現

■研究目標

- 学習型インデックスをFIBに適用
- FIBのコンパクトさを保ちながら高速な検索を実現

References:
[1] T. Kraska, et al., “The case for learned index structures,” in Proceedings of ACM SIGMOD, June 2018.

学習型インデックスの構築処理

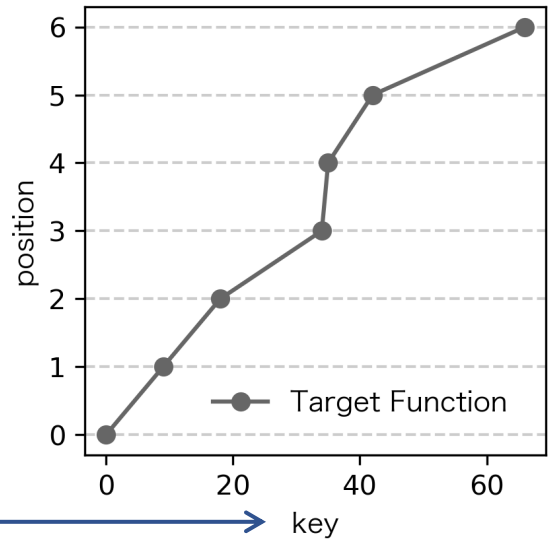
■ キーと格納位置の関係を回帰モデルで学習

- 〈キー, 値〉のペアをキーについて昇順にソートし配列に格納
  - キーと格納位置の関係は増加関数とみなせる
  - 以下、この関数を**ターゲット関数**と呼ぶ
- キーと格納位置の関係をニューラルネットワークなどの回帰モデルとして学習

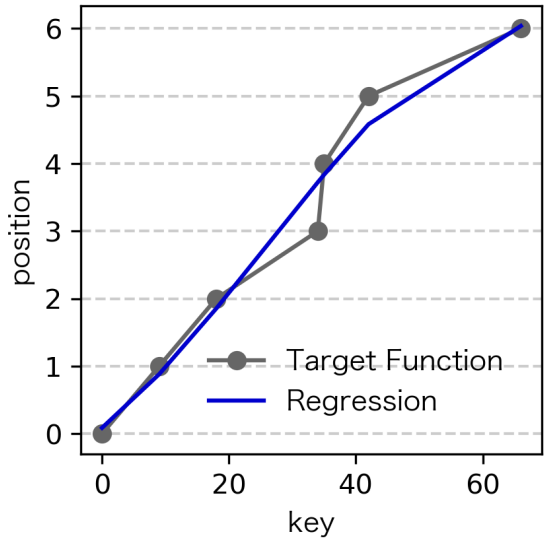
〈キー, 値〉のペアを
キーについて昇順にソート

Position	Key	Value
0	0	A
1	9	B
2	18	C
3	34	D
4	35	A
5	42	E
6	66	A

ターゲット関数
（増加関数）



回帰モデルでターゲット関数を学習



学習型インデックスの検索処理

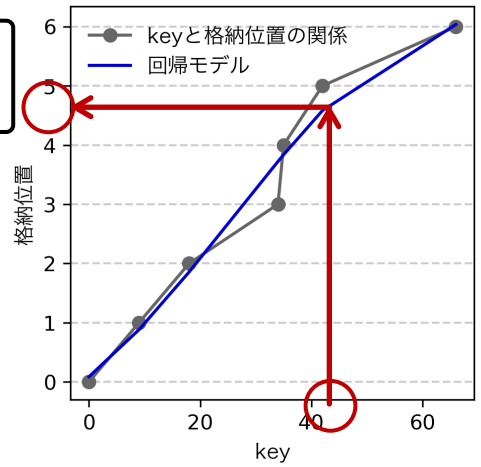
■ おおよその格納位置を**予測**し、予測の誤差分を**探索**する

1. **予測フェーズ**
   - キーを入力として学習型インデックスで格納位置を予測（以下予測位置）
2. **探索フェーズ**
   - 予測位置の周辺から真の格納位置を探索
     - 予測位置と、真の格納位置の間には回帰に起因する予測誤差が存在するため
     - 誤差: |予測位置 − 真の格納位置|

1. 予測フェーズ

- キー($x = 42$)の格納位置を予測

予測位置
$y = 4$



入力キー
$x = 42$

2. 探索フェーズ
予測位置周辺から真の格納位置を探索

予測位置($y = 4$)

実際の位置($y = 5$)

Position	Key	Value
0	0	A
1	9	B
2	18	C
3	34	D
4	35	A
5	42	E
6	66	A

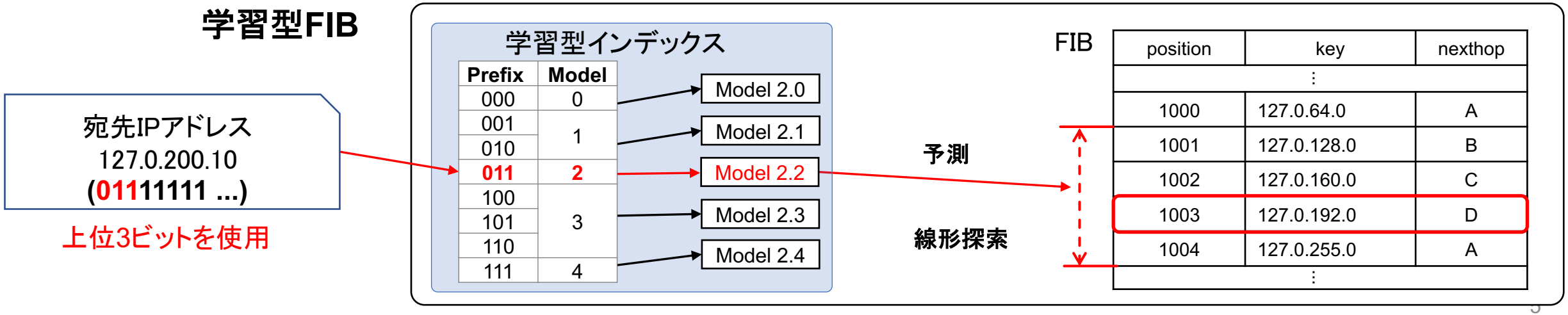
学習型FIBの全体像

■ 予測フェーズ

- テーブルによるマッピングと併用した2層構造による高速化
  - 第一層: IPプレフィクスの先頭nビットを用いて第2層モデルのIDを算出
  - 第二層: ニューラルネットワークによる回帰でFIBエントリの位置を予測

■ 探索フェーズ

- 分岐の除去、最新のCPU演算機能による高速化
  - 誤差を小さく、探索範囲が事前に確定するように設計



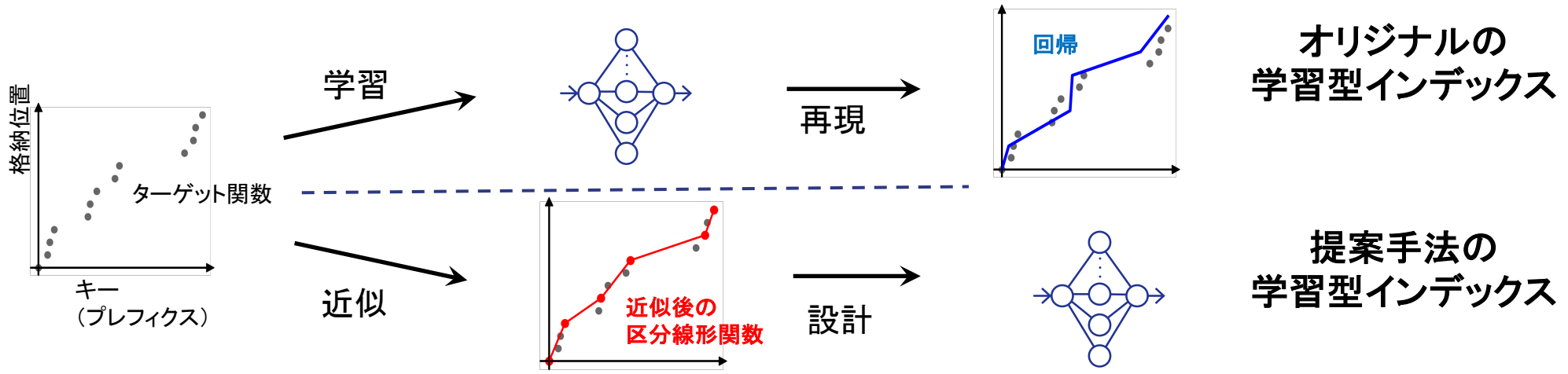
学習型FIB実現の課題とアプローチ

■ 学習型インデックスのFIBへの適用

- 学習型インデックスは完全一致検索しかサポートしてないのに対し、FIBの検索は最長一致検索が必要
- 厳密一致検索によるFIB検索のためのテーブル変換
  - 全てのIPプレフィクスが互いに素になるように分割してテーブルに格納

■ 高速化

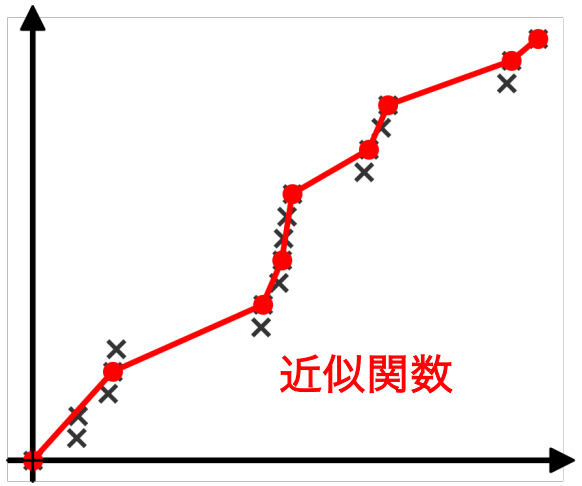
- 区分線形近似で回帰モデルを最大誤差を閾値以下に設計
  1. ターゲット関数を最大誤差を超えないような区分線形関数で近似
  2. 近似した区分線形関数の形になるようにニューラルネットワークを設計
- 最大誤差が保証されていることから、予測フェーズと探索フェーズの高速化を達成



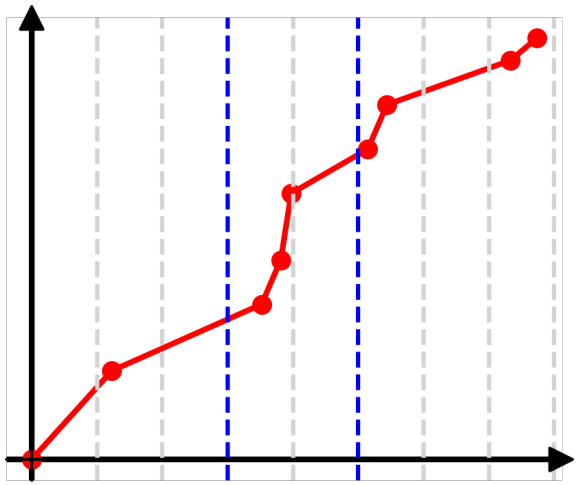
区分線形近似を用いた学習型インデックスの設計

1. ターゲット関数を区分線形関数で近似
   - 最大誤差が規定の値以下となるような区分線形関数で近似する
2. 区分線形関数を小さなサブ関数に分割
   - サブ関数が1つのニューラルネットワークで表現できるようにする
   - IPプレフィクスの先頭nビットでモデルIDを算出できるようにする
3. ニューラルネットワークの係数を計算
   - それぞれのサブ関数を表現するニューラルネットワークの係数を算出する

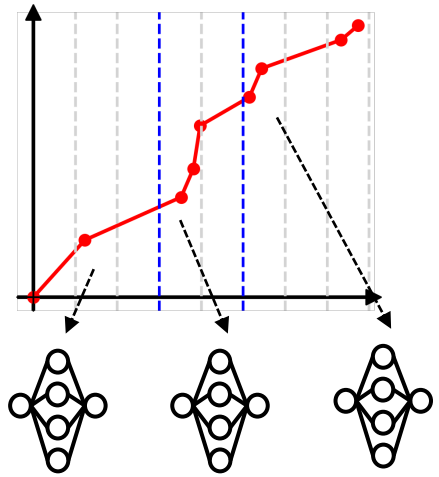
1) 区分線形近似



2) サブ関数に分割



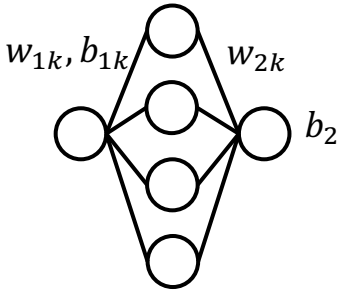
3) パラメータ計算



ニューラルネットワークのパラメータ計算

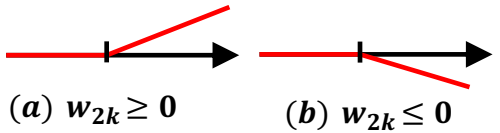
■ ニューラルネットワークの式（隠れ層1層・mニューロン・ReLU活性化関数）

$$\sum_{k=1}^{m} \mathrm{ReLU}(w_{1k}x + b_{1k})\, w_{2k} + b_2$$



■ ReLU関数の足し合わせ = 区分線形関数

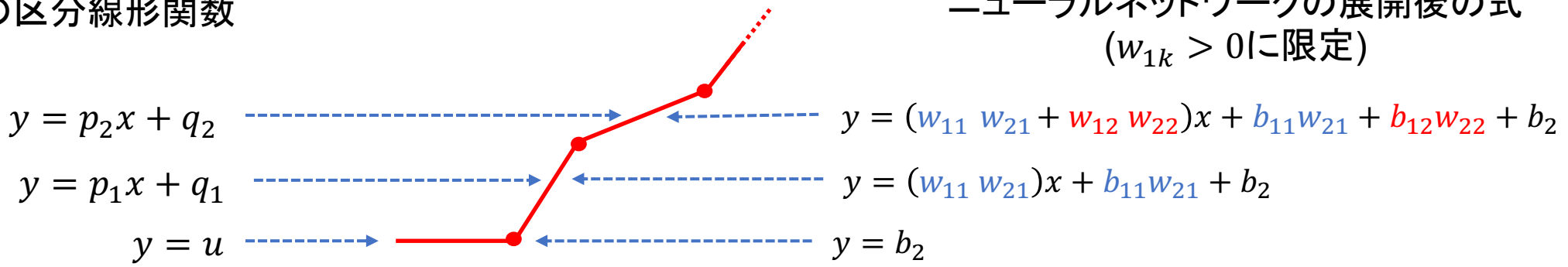
$$\mathrm{ReLU}(w_{1k}x + b_{1k})\; w_{2k} = \begin{cases} 0 & x < -b_{1k}/w_{1k} \\ (w_{1k}\, w_{2k})x + b_{1k}\, w_{2k} & x \geq -b_{1k}/w_{1k} \end{cases}$$



■ 区分線形関数とニューラルネットワークの恒等式

近似後の区分線形関数

ニューラルネットワークの展開後の式
($w_{1k} > 0$に限定)



■ パラメータの決定方法

- w_{1k}: 2つの区分の線の傾きの差の絶対値
- w_{2k}:凸変曲点の場合は+1, 凹変曲点の場合は−1
- b_{1k} , b_2: 上記の恒等式から導出

性能評価

■目的

- 高速化の効果を検証
  - 提案手法: 区分線形近似を用いて予測と探索フェーズを高速化
  - 既存手法: ナイーブな学習型インデックス[1]
    - 予測フェーズでは、第一層、第二層ともにニューラルネットワークによる予測
    - 探索フェーズでは、計算量は対数時間だが、探索範囲の決定と探索終了の判定にループと分岐が必要な指数探索
- 最先端のFIBと比較検証
  - Poptrie [4]: Trieベースの高速なFIB
    - NTTコミュニケーションズのソフトウェアルータ「Kamuee（カムイ）」で使用されている[7]

■FIBの生成

- Route views projectのBGP RIBのスナップショット[5]とCAIDAのASマップ[6]から生成

■評価環境

- Xeon Gold 6126 CPU (2.4GHz x 12 CPU cores, 32KB L1D, 1MB L2 cache)

References:
[2] S. Higuchi, et ,“Feasibility of longest prefix matching using learned index structures,”ACM SIGMETRICS Performance Evaluation Review, vol.48, no.4, pp.45–48, May 2021.
[4] H. Asai and Y. Ohara, “Poptrie: A compressed trie with population count for fast and scalable software IP routing table lookup,” in Proceedings of ACM SIGCOMM, Aug. 2015, pp. 57–70.
[5] University of Oregon Route Views Project. http://www.routeviews.org/routeviews/.
[6] CAIDA. https://www.caida.org/home/.
[7] NTT Comが「世界最高速レベル」ソフトウェアPCルーター開発 https://ascii.jp/elem/000/001/691/1691592/

実装最適化の効果

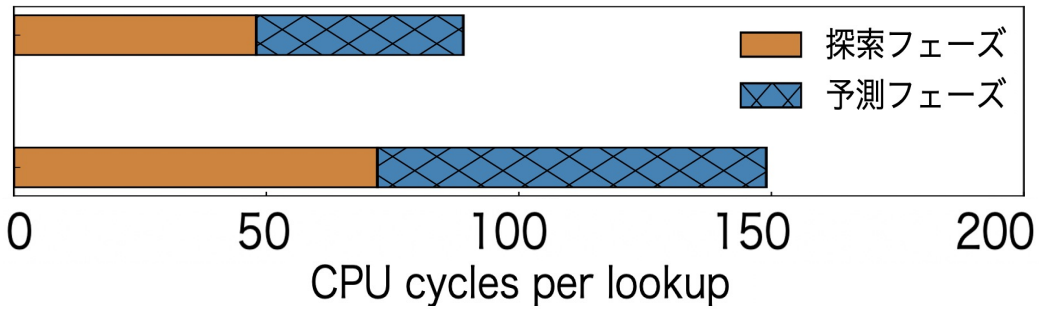
■ メモリサイズ: ほぼ同等

- 提案手法:　　1.83 Mbytes
- 既存手法 :　　1.78 Mbytes

■ 計算速度

- 予測フェーズの高速化効果：51%高速化
  - 第1層をテーブル検索にすることで計算を高速化
- 探索フェーズの高速化効果：33%高速化
  - 最大誤差を保証し、分岐なしの実装により分岐予測ミスによるパイプラインストールの時間を削減

	予測フェーズ		探索フェーズ
	第一層	第二層	
提案手法	テーブル検索	ニューラルネットワーク	線形探索
既存手法	ニューラルネットワーク	ニューラルネットワーク	指数探索



学習型FIB vs. Poptrie

■ メモリサイズ：Poptrieよりも小さい
- 学習型FIB：1.83 Mbytes
- Poptrie：1.98 Mbytes

■ 計算時間：平均速度はPoptrieよりも遅い
- プレフィクス長に対する計算時間の変化
  - 学習型FIB：定数時間
  - Poptrie: プレフィクス長によって計算時間に差がある
    - プレフィクス長18以下の場合はテーブル検索のみで探索が終了するため高速
    - プレフィクス長18より大きい場合は木を探索するため、木の深さに依存した時間がかかる
- 結果の意義
  - 将来的に長いIPプレフィクスが増加した場合、あるいはIPv6などで学習型FIBが有効