記憶ベース推論におけるk最近傍探索効率化手法の提案

小林俊介†　　木村昌臣†

芝浦工業大学工学部情報工学科†

1. 研究背景と目的

記憶ベース推論（MBR）では，推論対象となる未知の項目を持つレコード(未知の事象)に似たレコードを，コンピュータの記憶に当たるデータベースから,あらかじめ決めた k 個見つけ出し,それら k 最近傍と呼ばれるレコードを用いて推論対象の未知項目の予測を行う．この推論に用いる k 個の最近傍を見つけるための作業を，k 最近傍探索と呼ぶ.

MBR では多様な推論対象レコードに対してより良い結果を導くために，膨大なレコードをデータベースに蓄えておく必要がある．ここで問題となるのは，k 最近傍探索においてそれらすべてと類似度の計算を行うのは効率の良いものとは言えないということだ.

そこで本研究では，MBR における k 近傍探索で,近傍レコードを効率よく発見可能なデータ構造とアルゴリズムの提案と実装を目的とする.

2. 研究内容

2.1. 提案する手法が満たすべき要件

本研究は，MBR における k 最近傍探索効率化を行うために，以下の 2 つの要件を満たす.

i)　レコードの分布が考慮され，その分布が探索に与える影響が少ない

MBR に使用されるデータベース内のレコードは,分布に偏りがある場合が多い．そのため，座標の軸に沿ってレコードが存在する空間を分割して得られるデータ構造を用いて探索の効率化を行う既存手法を使用しても，レコードの分布によっては分割に無駄が発生してしまい，充分な効率化が図れない.

ii) k 最近傍を近似的に求めた場合，真でない k 最近傍がどの程度正しいかが分かる

計算時間短縮等のために，探索の途中打ち切りや，近いものほど衝突が発生しやすいハッシュ関数を利用する等の近似探索手法で k 最近傍を求める場合がある．しかしその場合，MBR の精度は低下する可能性がある．したがって,

Proposal of efficient techniques for k-nearest neighbor search on memory-based reasoning

†Shunsuk Kobayashi, Masaomi Kimura,, Sibaura institute of technology

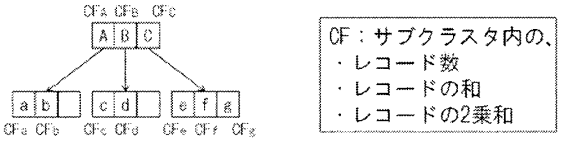

図1　CF-Tree

近似探索を行う場合には推論結果がどの程度正確なのかが分かる方が望ましい.

2.2. CF-Tree を利用した k 最近傍探索

これら 2 つの要件を満たすために，本研究では CF-Tree を利用する．この手法は本来，大規模データのためのクラスタリング手法 BIRCH[1]における，クラスタリングの前処理として行うサブクラスタリングの際に用いられるものである.

CF-Tree は図 1 のような B+-Tree に類似した木構造の形をとり，各ノード（サブクラスタ）が下位に属するレコード群の特徴を表すベクトル $\mathbf{CF} = (N, \mathbf{LS}, SS)$ を持つ．ここで N はサブクラスタ内のレコードの数，**LS** はサブクラスタ内のレコードをベクトルと考えた場合の和，SS はサブクラスタ内のレコードの二乗和を表す．この **CF** を利用して近いレコード同士をサブクラスタにまとめ，それをノードとして作られた木構造は,各レコード同士の分布に基づくデータ構造であるといえる.

前述の通り CF-Tree は本来クラスタリングのための手法であり，k 最近傍探索のための手法ではない．しかし，その構造は類似したもの同士の集まりを階層的に表している．よって，探索を行う際の基準となる，ノードを表す領域およびクエリ（レコード）とノード領域の距離計算方法を定義することで，k 最近傍探索を行うことが可能となる．本研究ではこの CF-Tree をベースに k 最近傍探索効率化のための手法提案を目指す.

2.3. ノード領域の定義

i)　標準偏差を利用した球

CF から求まるノード内のレコード群の標準偏差　$\sigma = \sqrt{(SS - \mathbf{LS}^2/N)/N}$　を利用して領域定義を行う方法．これはノード内のレコード分布

が正規分布に従うことを仮定した場合に，1.96σを半径とした球がノードの領域であるとすると，領域内にノードに所属するレコードが 95%の確率で含まれる性質を利用したものである．この領域定義では辿ったノードが必ずしも真の k 最近傍を持つわけではないが，近似解を得ることができる．

領域とクエリ（レコード $\mathbf{x}$）の距離は，領域が球形であることから，ノードの重心$\frac{LS}{N}$と半径 1.96σより，$\left|\mathbf{x}-\frac{LS}{N}\right|-1.96\sigma$ として求めることができる．

ii) 包含矩形

ノード内のレコードの中から，各属性の最大・最小値を求めることで，所属レコード全てを囲む最小の矩形としてノードを表す．これは R-Tree に似ているが，CF-Tree が元となるため，よりクラスタ志向の強い構造となる．

矩形とクエリとの距離は，矩形情報$(\mathbf{H},\mathbf{L})$より，$\sqrt{\sum_{i=1}^{d} dist_i^{\,2}}$

$$\left(dist_i^{\,2} = \begin{cases} (L_i - x_i)^2 & x_i < L_i \\ 0 & L_i \leqq x_i \leqq H_i \\ (H_i - x_i)^2 & L_i < x_i \end{cases} \right)$$

で求めることができる．

iii)傾き付き包含矩形

CF に追加情報としてレコードの項目同士を掛けた値の和からなる $\mathbf{CS} = \left[\sum_{k=1}^{n} x_{ik} \cdot x_{jk}\right]$ を持たせることで，各ノードに対し主成分分析を行うことができる．これを利用して得られる主成分ベクトルを基底にしてノードに属するレコードを見た場合，その包含矩形は ii)よりも体積が小さなものとなる．

クエリとの距離計算は，傾き付き矩形を作る際使用した主成分ベクトルによりクエリを基底変換した後，ii)と同じ計算を行う．

2.4. 探索方法

前節で述べたような領域定義されたノードからなる木に対し，最良優先探索を施すことで k 最近傍探索を行う．これは優先順位付キューを利用した探索方法で，距離計算を行ったノードはキューに格納するとその中でソートされ，クエリに近いものから順に取り出される．これを利用して k 最近傍の発見・更新を繰り返し，最終的に見つかっている最も遠い k 最近傍よりもクエリから離れたノードが取り出された時点，またはキューが空になった時点で探索が終了となる．

3. 実験

手法の評価を行うため，ここまで述べてきた手法を java で実装し実験を行った．分布の傾向が異なる数パターンのレコード生成方法により次元数，レコード数を変化させてデータセットを構築し，探索実行時間および探索ノード数による比較実験を行った．また，既存手法の例として R*-Tree との比較も行った．図 3 はクラスタごとにレコードの傾向が異なるデータセットを使用した実験結果の一部である．

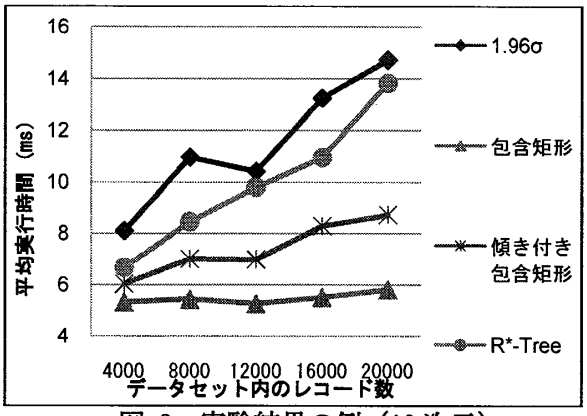

図 2 実験結果の例（10 次元）

4. 考察

1.96σ領域を用いた探索は木構造の構築・維持にかかるコストが矩形に比べ低いという利点はあるが，領域肥大化による探索効率の低下が目立った．

傾き付き包含矩形に関しては，高いパフォーマンスを期待していたクラスタごとに傾向の異なるレコードで形成されたデータセットにおいて，単純な矩形に劣った結果が出てしまった．これは矩形を傾けることにより発生する領域のオーバーラップを考慮していなかったためと考えられる．

R*-Tree との比較については，単純な相関がある低次元のデータセットの場合に R*-Tree の方が良い結果がでたが，クラスタデータや高次元の場合には本研究が提案する手法の方が効率よく探索が行えていた．

5. 今後の課題

今回の実験の結果，データ分布の傾向を重視した k 最近傍探索の効率化はある程度効果があることがわかった．しかし，分割した領域の肥大化を防ぐことや，領域のオーバーラップの考慮等，課題が残っている．また本研究で提案した近似探索が MBR の結果に与える影響を調べることも必要である．

参考文献

[1] T. Zhang, R. Ramakrishnan, M. Livny:BIRCH: A New Data Clustering Algorithm and Its Applications, Data Mining and Knowledge Discovery, vol. 1, pp. 141-182(1997).