距離とクラスタリング

山本章博

情報学研究科 知能情報学専攻

(工学部 情報学科)

編集距離

距離とは

- データ x, y の距離dとは関数 $d : (\mathrm{x}, \mathrm{y}) \rightarrow \mathbf{R}$で,
  以下の4条件を満たすもの

  (1) $d(x, y) \geq 0$

  (2) $d(x, y) = 0 \quad \Leftrightarrow \quad x = y$

  (3) $d(x, y) = d(y, x)$ ：対称性

  (4) $d(x, y) \leq d(x, z) + d(z, y)$ ：三角不等式

  - データの型によっては“類似性”として“距離”を用いないこともある

離散データに対する距離の例

- データ長が一定の構造体の場合

$$\boldsymbol{x} = (x_1, \ldots, x_d), \boldsymbol{y} = (y_1, \ldots, y_d) \in D_1 \times \ldots \times D_d$$

$$\mathrm{d_H}(\boldsymbol{x}, \boldsymbol{y}) = \Sigma_{i=1}^{d} \mathrm{c}(x_i, y_i) \qquad \text{Hamming距離}$$

$$\mathrm{c}(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{o.w.} \end{cases}$$

離散データに対する距離の例

- データ長が一定でない構造体(文字列など)の場合

  $d_H(x, y)$: xをyに書換えるために必要な編集操作の最小ステップ数　　編集距離

  編集操作 : 文字の削除
  　　挿入
  　　置換

編集距離の計算(1)

- 編集操作に対するコスト

  削除のコスト: 1

  挿入のコスト: 1

  置換のコスト: 1 (置換を認める場合)

  3 (置換を認めない場合)

  - 一般に

    文字cの削除のコスト: del(c)

    文字cの挿入のコスト: ins(c)

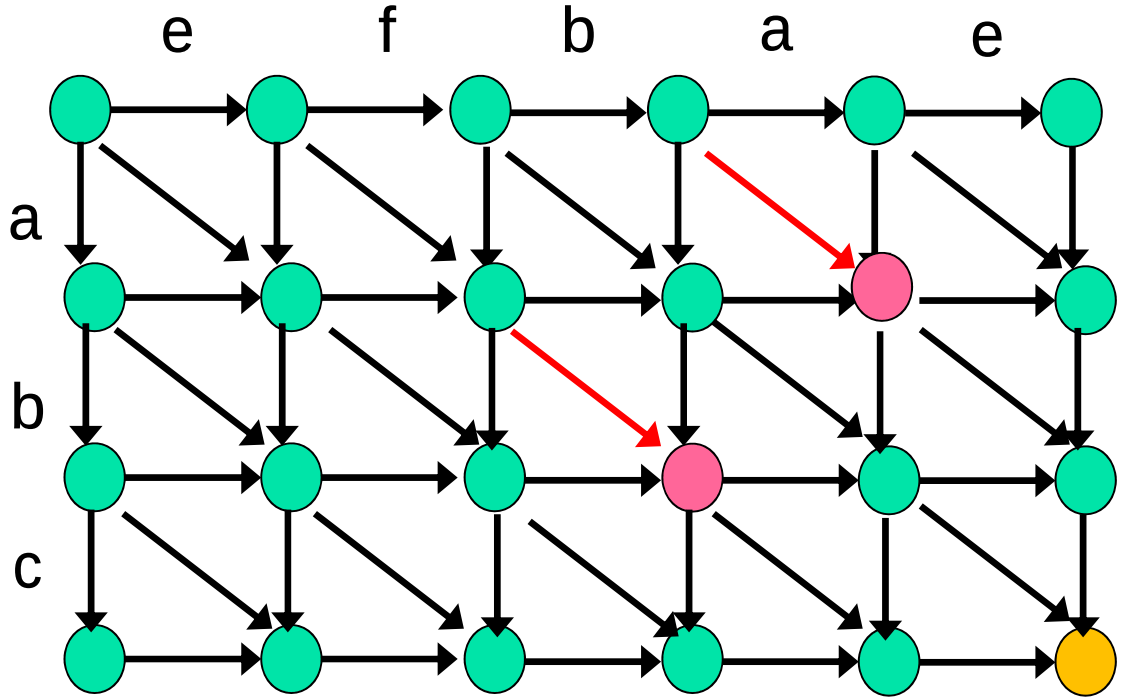
    文字xをyに置換するコスト : sub(c,d)

    とするとき, 編集距離が定義できる必要十分条件は

    del(c) > 0 かつ ins(c) > 0 かつ sub(c,d)が距離になること

編集距離の計算(2)

- 編集グラフ
  - 左に進む：文字cを挿入 下に進む：文字d を削除
  - 対角線：文字d を文字cに置換



：一致ノード

一致ノードを終点とする
対角線方向辺のコストは0

他の辺のコストは，定義に
従う

編集距離の計算(3)

- 配列 $T[i, j]$ を計算

$T[-1, -1] = 0,$

$T[i, -1] = T[i-1, -1] + 1,\ T[-1, i] = T[-1, i-1] + 1,$

$T[i, j] = \min(T[i-1, j]+1,\ T[i, j-1]+1,$

$T[i-1, j-1] + k\,(1 - \delta(\boldsymbol{x}[i], \boldsymbol{y}[j])))$

k =1			e	f	b	a	e
		-1	0	1	2	3	4
	-1	0	1	2	3	4	5
a	0	1	1	2	3	3	4
b	1	2	2	2	2	3	4
c	2	3	3	3	3	3	4

$$\delta(c, d) = \begin{cases} 1 \text{ if } c = d \\ 0 \text{ o.w.} \end{cases}$$

$k = 1$ or 3

編集距離の計算(3)

- 配列 $T[i, j]$ を計算

$T[-1, -1] = 0,$

$T[i, -1] = T[i-1, -1] + 1,\ T[-1, i] = T[-1, i-1] + 1,$

$T[i, j] = \min(T[i-1, j]+1,\ T[i, j-1]+1,$

$T[i-1, j-1] + k\,(1 - \delta(\boldsymbol{x}[i], \boldsymbol{y}[j])))$

k =3			e	f	b	a	e
		-1	0	1	2	3	4
	-1	0	1	2	3	4	5
a	0	1	2	3	4	3	4
b	1	2	3	4	3	4	5
c	2	3	3	3	4	5	6

$$\delta(c, d) = \begin{cases} 1 \text{ if } c = d \\ 0 \text{ o.w.} \end{cases}$$

$k = 1$ or 3

動的計画法(Dynamic Programming)

- 問題を部分問題に分割し，サイズの小さい問題から順に解く

例: 最短経路問題に対するDijkstra法

有向グラフの各辺 e に“長さ” $d(e) > 0$が与えられているとき，始点となる節点 s から節点 $v(s)$への最短路を求める

Dijkstra法

S:={s}，各vに対して$r(v) := \infty$

全ての v がS に属するまで以下を繰り返す

各vに対して

if $v \notin S$ かつ $r(m) + d((v, m)) < r(v)$

$r(v) := r(m) + d((v, m)), p(v) := m$

$m := \mathrm{argmin}_{v \notin S} r(v)$

$S = S \cup \{m\}$

- 節点には付番しておき，最短距離にある節点がある場合には，最小番号の節点をSに加える