

インターネット計測とデータ解析 第12回

長 健二郎

2011年7月20日

前回のおさらい

インターネットの異常や問題を計る

- ▶ 異常検出
- ▶ スпам判定
- ▶ ベイズ理論
- ▶ 演習:異常検出

今日のテーマ

データマイニング

- ▶ パターン抽出
- ▶ クラス分類
- ▶ クラスタリング
- ▶ 演習:クラスタリング

データマイニング

- ▶ 膨大なデータ
 - ▶ 従来の手法では把握しきれない
 - ▶ データの中に隠れた情報を抽出する必要
- ▶ Data Mining
 - ▶ 膨大なデータ、かつ、多次元、多様、分散などの特徴
 - ▶ 手法は機械学習、AI、パターン認識、統計、データベースなどからアイデア
- ▶ クラウド技術などで大量データ処理が現実的に

Data Mining 手法のいろいろ

- ▶ パターン抽出: データが内包する規則や特徴的なパターンを見つける
 - ▶ 相関
 - ▶ 時系列
- ▶ 分類: オリジナル情報にない分類を機械的に実現
 - ▶ ルールベース
 - ▶ 単純ベイズ分類器
 - ▶ ニューラルネットワーク
 - ▶ サポートベクターマシン (SVM)
 - ▶ 次元減少 (主成分分析, PCA)
- ▶ クラスタリング: 変量間の距離 (類似度) を計算しグループ化
 - ▶ 距離ベース、密度ベース、グラフベース
 - ▶ k-means、DBSCAN
- ▶ 異常検出: 統計手法を使って定常状態からのずれを検出
 - ▶ 単変数、多変数
 - ▶ 外れ値検出

距離について

いろいろな距離

- ▶ ユークリッド距離 (Euclidean distance)
- ▶ 標準化ユークリッド距離 (standardized Euclidean distance)
- ▶ ミンコフスキー距離 (Minkowski distance)
- ▶ マハラノビス距離 (Mahalanobis distance)

類似度

- ▶ バイナリベクトルの類似度
- ▶ n 次元ベクトルの類似度

距離の性質

空間上の2点 (x, y) 間の距離 $d(x, y)$:

非負性 (positivity)

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

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

対称性 (symmetry)

$$d(x, y) = d(y, x)$$

三角不等式 (triangle inequality)

$$d(x, z) \leq d(x, y) + d(y, z)$$

ユークリッド距離 (Euclidean distance)

普通に距離といえばユークリッド距離を指す
n次元空間での2点 (x, y) の距離

$$d(x, y) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2}$$

標準化ユークリッド距離 (standardized Euclidean distance)

- ▶ 変数間でばらつきが大きさが異なると、距離が影響を受ける
- ▶ そこで、ユークリッド距離を各変数の分散で割って正規化

$$d(x, y) = \sqrt{\sum_{k=1}^n \frac{(x_k - y_k)^2}{s_k^2}}$$

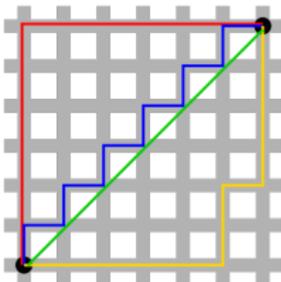
ミンコフスキー距離 (Minkowski distance)

ユークリッド距離を一般化

- ▶ パラメータ r が大きいほど、次元軸にとらわれない移動 (斜め方向のショートカット) を重視する距離

$$d(x, y) = \left(\sum_{k=1}^n |x_k - y_k|^r \right)^{\frac{1}{r}}$$

- ▶ $r = 1$: マンハッタン距離
 - ▶ ハミング距離: 2つの文字列間の同じ位置の文字の不一致数
 - ▶ 例えば、111111 と 101010 のハミング距離は 3
- ▶ $r = 2$: ユークリッド距離



Manhattan distance vs. Euclidean distance

マハラノビス距離 (Mahalanobis distance)

変数間に相関がある場合に、相関を考慮した距離

$$\text{mahalanobis}(x, y) = (x - y)\Sigma^{-1}(x - y)^T$$

ここで Σ^{-1} は共分散行列の逆行列

類似度

類似度

- ▶ ふたつのデータの似ている度合の数値表現

類似度の性質

非負性 (positivity)

$$0 \leq s(x, y) \leq 1$$

$$s(x, y) = 1 \Leftrightarrow x = y$$

対称性 (symmetry)

$$s(x, y) = s(y, x)$$

三角不等式 (triangle inequality) は一般に類似度には当てはまらない

バイナリベクトルの類似度

Jaccard 係数

- ▶ 1 の出現が少ないバイナリベクトル同士の類似度に使われる
- ▶ 文書中に出現する単語から文書の類似度を示す場合など
- ▶ 多くの単語は両方とも出現しない \Rightarrow これらは考慮しない
- ▶ 2 つのベクトルの各要素の対応関係を表のように集計

		vector y	
		1	0
vector x	1	n_{11}	n_{10}
	0	n_{01}	n_{00}

Jaccard 係数は以下で表される

$$J = \frac{n_{11}}{n_{11} + n_{10} + n_{01}}$$

n次元ベクトルの類似度

一般のベクトルの類似度

- ▶ 文書の類似度で出現頻度も考慮する場合など

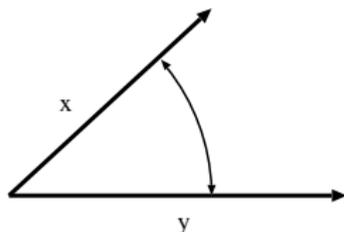
コサイン類似度

- ▶ ベクトルの x, y の cosine を取る、向きが一致:1、直交:0、向きが逆:-1
- ▶ ベクトルの長さで正規化 \Rightarrow 大きさは考慮しない

$$\cos(x, y) = \frac{x \cdot y}{\|x\| \|y\|}$$

$x \cdot y = \sum_{k=1}^n x_k y_k$: ベクトルの積

$\|x\| = \sqrt{\sum_{k=1}^n x_k^2} = \sqrt{x \cdot x}$: ベクトルの長さ



コサイン類似度の例題

$$x = 3 \ 2 \ 0 \ 5 \ 0 \ 0 \ 0 \ 2 \ 0 \ 0$$

$$y = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 2$$

$$x \cdot y = 3 * 1 + 2 * 1 = 5$$

$$\|x\| = \sqrt{3 * 3 + 2 * 2 + 5 * 5 + 2 * 2} = \sqrt{42} = 6.481$$

$$\|y\| = \sqrt{1 * 1 + 1 * 1 + 2 * 2} = \sqrt{6} = 2.449$$

$$\cos(x, y) = \frac{5}{6.481 * 2.449} = 0.315$$

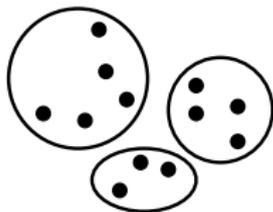
クラスタリング手法

変量間の距離 (類似度) を計算しグループ化

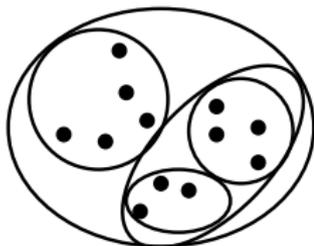
- ▶ データを分類し理解する
- ▶ データを要約する
- ▶ 分割型クラスタリング (partitional clustering)
 - ▶ k-means 法
- ▶ 階層型クラスタリング (hierarchical clustering)
 - ▶ MST 法
 - ▶ DBSCAN 法



original points



partitional clustering



hierarchical clustering

k-means 法

- ▶ 分割型クラスタリング
- ▶ クラスタ数 k を指定
- ▶ 基本アルゴリズムはシンプル
 - ▶ 各クラスタは重心 (centroid) を持つ (通常は平均)
 - ▶ 各データを最も近い重心を持つクラスタに割り当てる
 - ▶ データの割り当てと重心の再計算を繰り返す
- ▶ 制約
 - ▶ 事前にクラスタ数 k を指定する必要
 - ▶ 初期値によって結果が変わる
 - ▶ クラスタが異なるサイズ、密度をもつ場合や円形でない場合
 - ▶ 外れ値の影響が大きい

basic k-means algorithm:

- 1: select k points randomly as the initial centroids
- 2: **repeat**
- 3: form k clusters by assigning all points to the closest centroid
- 4: recompute the centroid of each cluster
- 5: **until** the centroids don't change

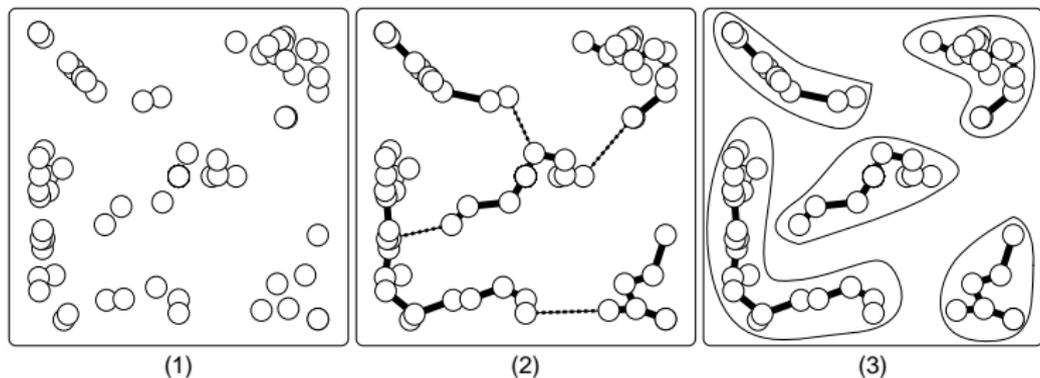
階層型クラスタリング

- ▶ ツリー構造でクラスタを生成
 - ▶ ツリー構造でクラスタ構成が説明可能
- ▶ 事前にクラスタ数を指定する必要がない
- ▶ 2種類のアプローチ
 - ▶ 凝集型: 各データを1クラスタとして、統合していく
 - ▶ 分割型: 全体を1クラスタとして始め、分割していく

MST クラスタリング

Minimum Spanning Tree クラスタリング

- ▶ 分割型の階層型クラスタリング
- ▶ 任意の点からスタートしスパニングツリーを作る
- ▶ 距離の長いエッジから削除してクラスタを分割していく



DBSCAN

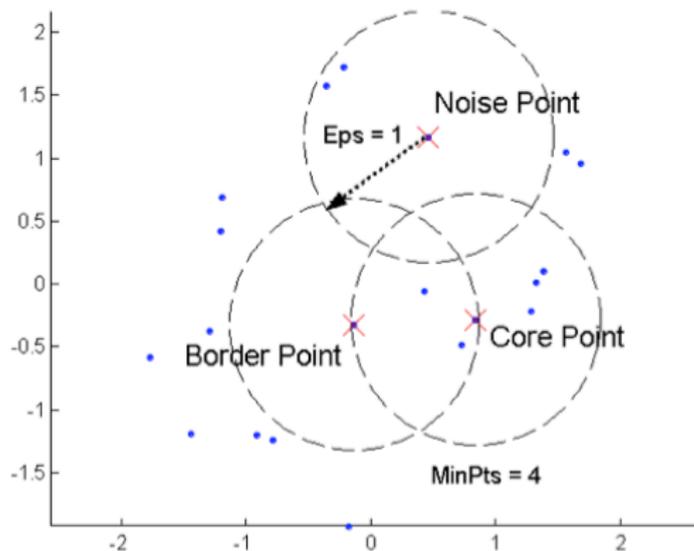
Density-Based Spatial Clustering

- ▶ 密度: 指定した距離内のデータ数
- ▶ (球状でない) 任意形状のクラスタの抽出が可能
- ▶ ノイズに強い
- ▶ 距離の閾値 Eps と数の閾値 $MinPts$
 - ▶ Core points: 距離 Eps 内に $MinPts$ 以上の近傍点がある
 - ▶ Border points: Core ではないが、距離 Eps 内に Core が存在
 - ▶ Noise points: 距離 Eps 内に Core が存在しない
- ▶ 弱点: 密度が異なるクラスタや次数の多いデータ

DBSCAN algorithm:

- 1: label all points as core, border, or noise points
- 2: eliminate noise points
- 3: put an edge between all core points that are within Eps of each other
- 4: make each group of connected core points into a separate cluster
- 5: assign each border point to one of the clusters of its associated core points

DBSCAN: Core, Border, and Noise Points

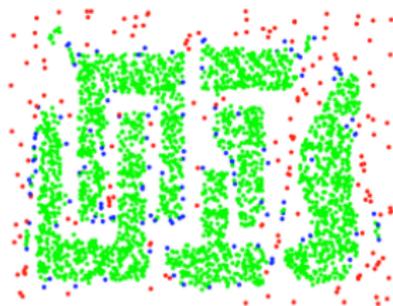


source: Tan, Steinbach, Kumer. Introduction to Data Mining

DBSCAN: example of Core, Border, and Noise Points



Original Points

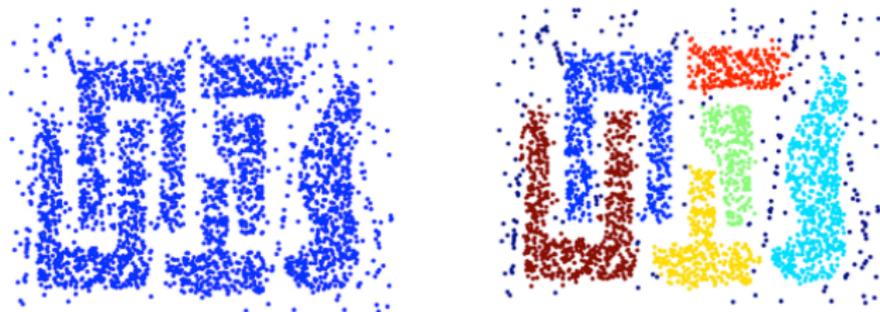


Point types: **core**, **border**
and **noise**

Eps = 10, MinPts = 4

source: Tan, Steinbach, Kumer. Introduction to Data Mining

DBSCAN: example clusters

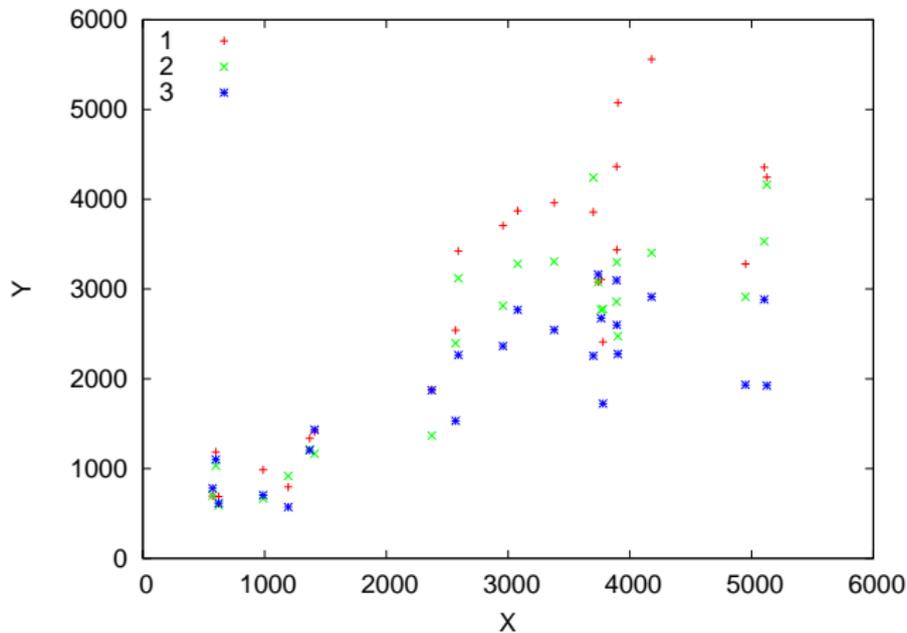


Clusters

source: Tan, Steinbach, Kumer. Introduction to Data Mining

演習: k-means clustering

- ▶ データ: 第6回授業で使ったアクセス数比較 月曜対水、金、日曜



サンプルコード (1/2)

```
k = 3 # k clusters
re = /^(\d+)\s+(\d+)/
INFINITY = 0x7fffffff

# read data
nodes = Array.new # array of array for data points: [x, y, cluster_index]
centroids = Array.new # array of array for centroids: [x, y]
ARGF.each_line do |line|
  if re.match(line)
    c = rand(k) # randomly assign initial cluster
    nodes.push [$1.to_i, $2.to_i, c]
  end
end

round = 0
begin
  updated = false

  # assignment step: assign each node to the closest centroid
  if round != 0 # skip assignment for the 1st round
    nodes.each do |node|
      dist2 = INFINITY # square of distance to the closest centroid
      cluster = 0 # closest cluster index
      for i in (0 .. k - 1)
        d2 = (node[0] - centroids[i][0])**2 + (node[1] - centroids[i][1])**2
        if d2 < dist2
          dist2 = d2
          cluster = i
        end
      end
      node[2] = cluster
    end
  end
end
```

サンプルコード (2/2)

```
# update step: compute new centroids
sums = Array.new(k)
clsize = Array.new(k)
for i in (0 .. k - 1)
  sums[i] = [0, 0]
  clsize[i] = 0
end
nodes.each do |node|
  i = node[2]
  sums[i][0] += node[0]
  sums[i][1] += node[1]
  clsize[i] += 1
end

for i in (0 .. k - 1)
  newcenter = [Float(sums[i][0]) / clsize[i], Float(sums[i][1]) / clsize[i]]
  if round == 0 || newcenter[0] != centroids[i][0] || newcenter[1] != centroids[i][1]
    centroids[i] = newcenter
    updated = true
  end
end

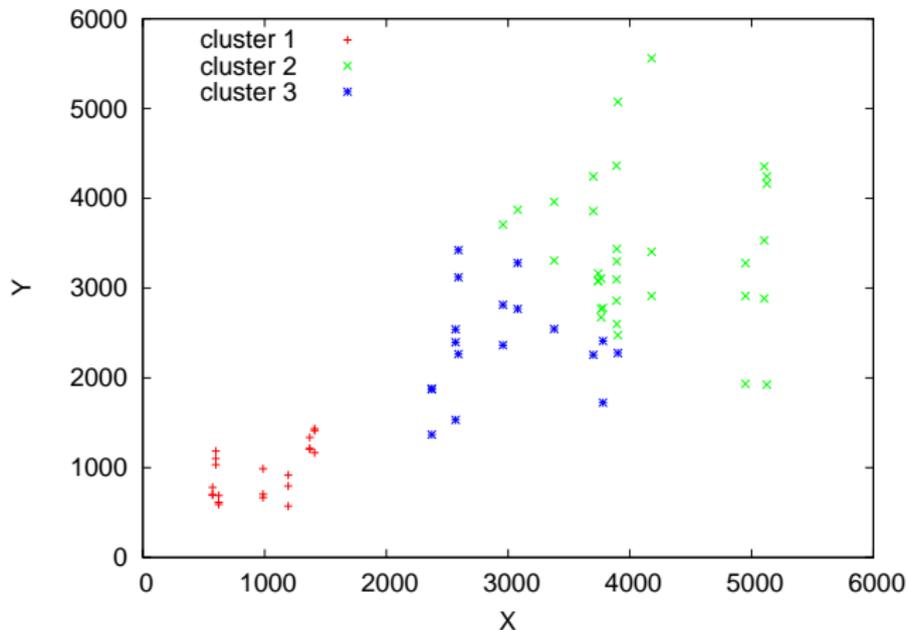
round += 1

end while updated == true

# print the results
nodes.each do |node|
  puts "#{node[0]}\t#{node[1]}\t#{node[2]}"
end
```

k-means clustering 結果

- ▶ 初期値によって結果に違いができる



まとめ

データマイニング

- ▶ パターン抽出
- ▶ クラス分類
- ▶ クラスタリング
- ▶ 演習:クラスタリング

次回予定

第 13 回 スケールする計測と解析 (7/27)

- ▶ 分散並列処理
- ▶ クラウド技術
- ▶ インターネット計測とプライバシー
- ▶ 演習:大規模データ処理