インターネット計測とデータ解析 第 12 回

長 健二朗

2013 年 6 月 26 日

前回のおさらい

第 10 回 異常検出と機械学習 (6/19)

- 異常検出
- 機械学習
- スパム判定とベイズ理論
- 演習: 単純ベイズ分類器

第 11 回 パケット解析 (6/19) 6 限 (18:10-19:40) $\lambda 13$

- UNIX コマンド
- パケットキャプチャリング
- プロトコル解析

今日のテーマ

第 12 回 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 (復習)

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)

- 変数間でばらつきの大きさが異なると、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

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

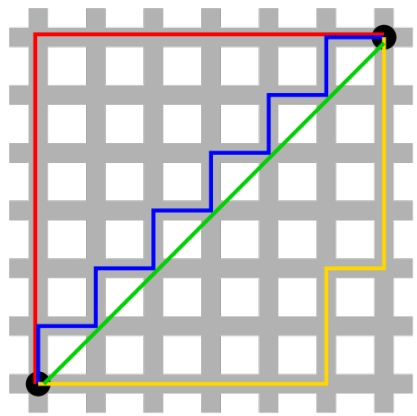
ミンコフスキー距離 (Minkowski distance)

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

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

- $r = 1$: 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

ベクトルのノルム (vector norm) (1/2)

ベクトルのノルム: ベクトルの長さ

$$\|x\| \text{ where } x \text{ is a vector}$$

ベクトル x の l_n-ノルムはミンコフスキー距離で定義される

$$\|x\|_n = \sqrt[n]{\sum_i |x_i|^n}$$

l_0-norm: ベクトルの 0 でない要素の数

$$\|x\|_0 = \#(i|x_i \neq 0)$$

l_1-norm: 差分の和

$$\|x\|_1 = \sum_i |x_i|$$

l_2-norm: ユークリッド距離

$$\|x\|_2 = \sqrt{\sum_i |x_i|^2}$$

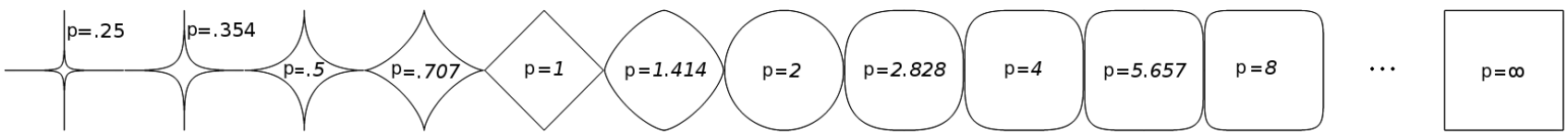
l_∞-norm: ベクトルの最大要素

$$\|x\|_\infty = max(|x_i|)$$

ベクトルのノルム (vector norm) (2/2)

例: ベクトル $x = (1, 2, 3)$ に対して

$$
\begin{aligned}
\|x\|_0 &\quad 3 = 3.000 \\
\|x\|_1 &\quad 6 = 6.000 \\
\|x\|_2 &\quad \sqrt{14} = 3.742 \\
\|x\|_3 &\quad 6^{2/3} = 3.302 \\
\|x\|_4 &\quad 2^{1/4}\sqrt{7} = 3.146 \\
\|x\|_\infty &\quad 3 = 3.000
\end{aligned}
$$



unit circles of l_p-norm with various values of p

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

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

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に出現しない $\Rightarrow$ 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

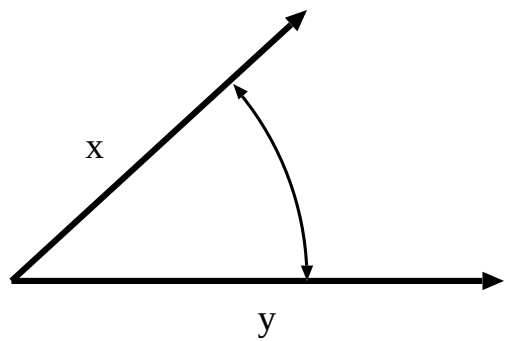
- 文書の類似度で出現頻度も考慮する場合など

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

$$x = 3 \quad 2 \quad 0 \quad 5 \quad 0 \quad 0 \quad 0 \quad 2 \quad 0 \quad 0$$
$$y = 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 2$$

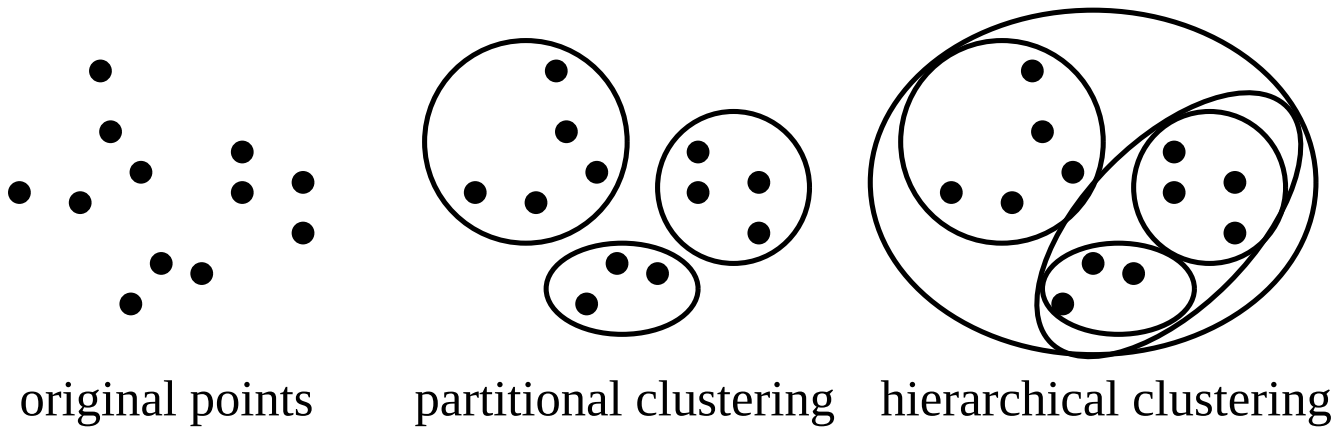
$$x \cdot y = 3 * 1 + 2 * 1 = 5$$
$$\|x\| = \sqrt{3 * 3 + 2 * 2 + 5 * 5 + 2 * 2} = \sqrt{42} = 6.481$$
$$\|y\| = \sqrt{1 * 1 + 1 * 1 + 2 * 2} = \sqrt{6} = 2.449$$

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

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 (patitional 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 controid
4: recompute the centroid of each cluster
5: **until** the centroids don't change

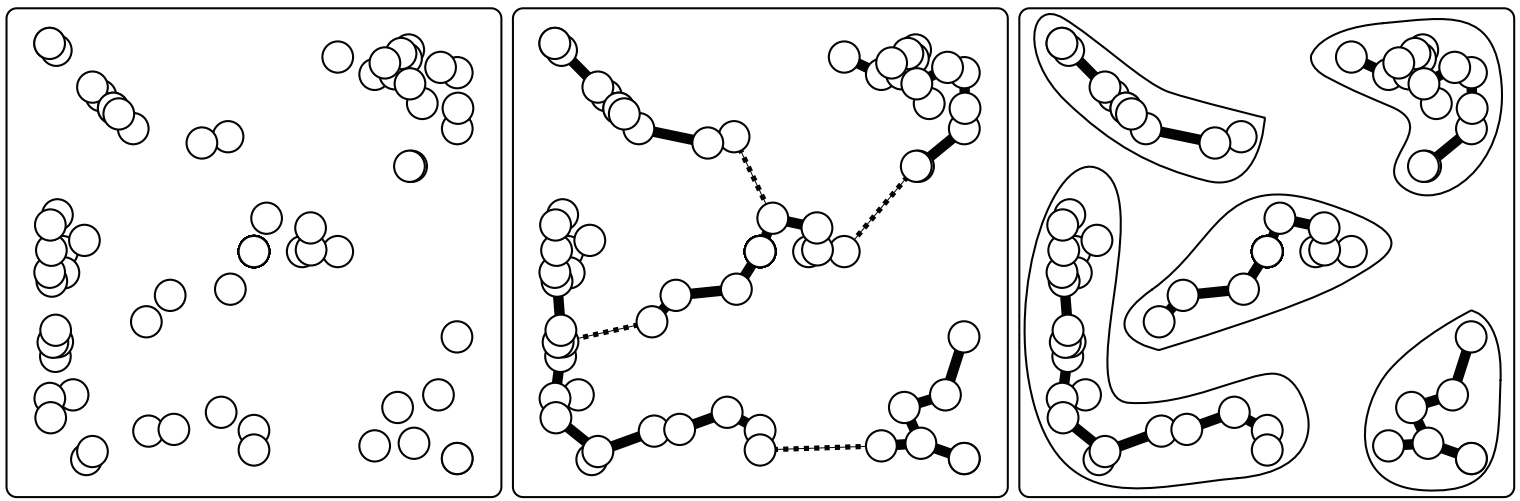
階層型クラスタリング

- 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



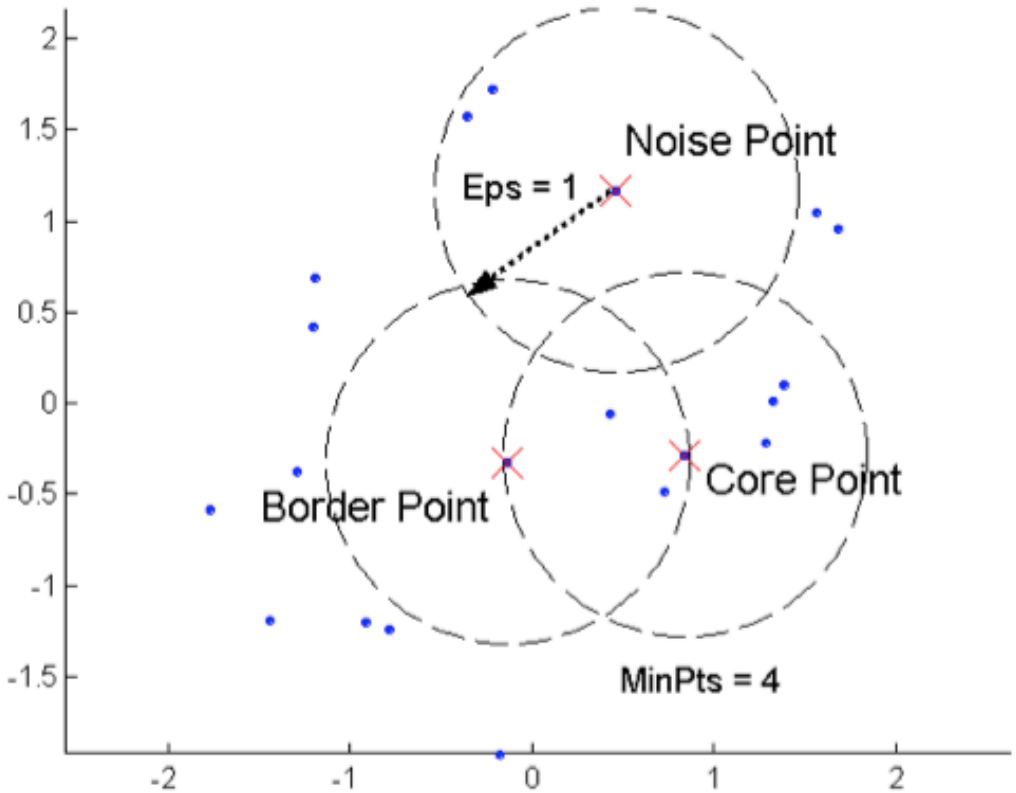
DBSCAN

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

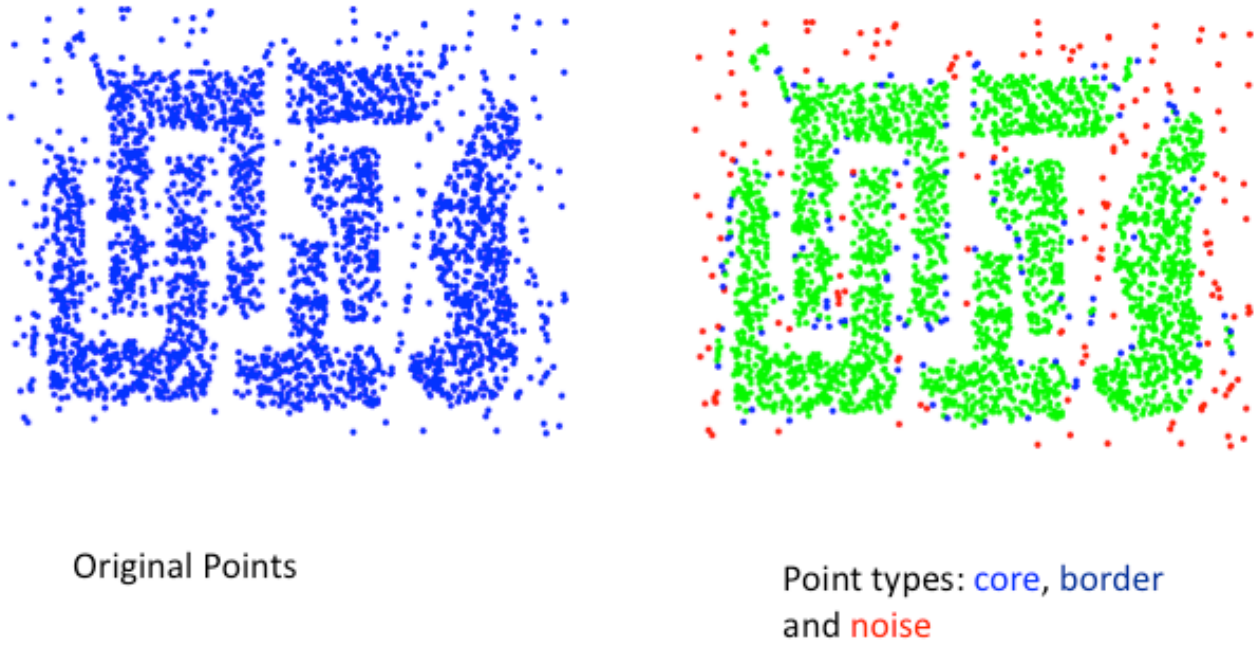
DBSCAN algorithm:
1: label all points as core, border, or noise points
2: eliminate noise points
3: put an edge between all core points that are within Eps of each other
4: make each group of connected core points into a separate cluster
5: assign each border point to one of the clusters of its associated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

DBSCAN: Core, Border, and Noise Points



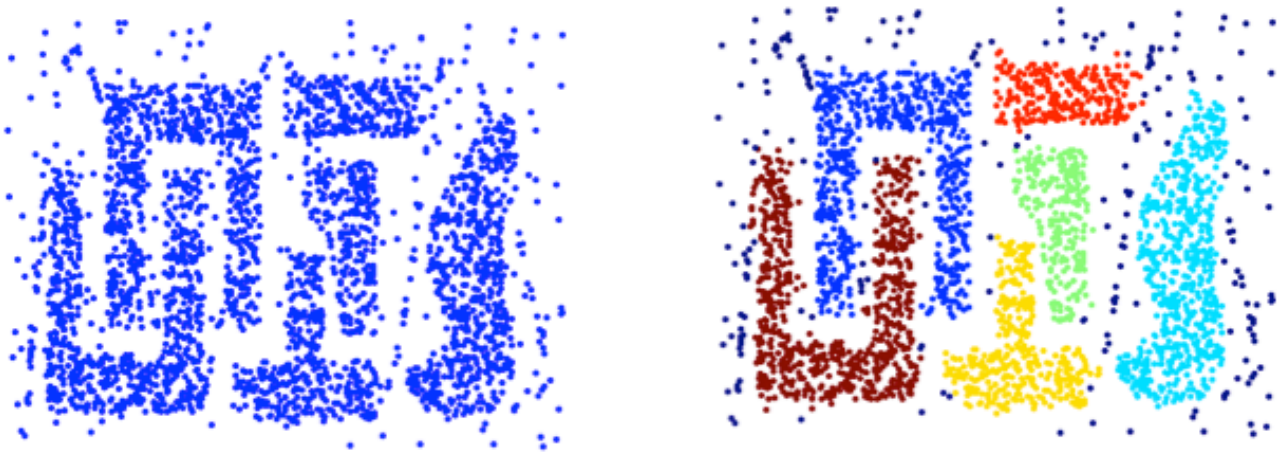
source: Tan, Steinbach, Kumer. Introduction to Data Mining

DBSCAN: example of Core, Border, and Noise Points



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

DBSCAN: example clusters

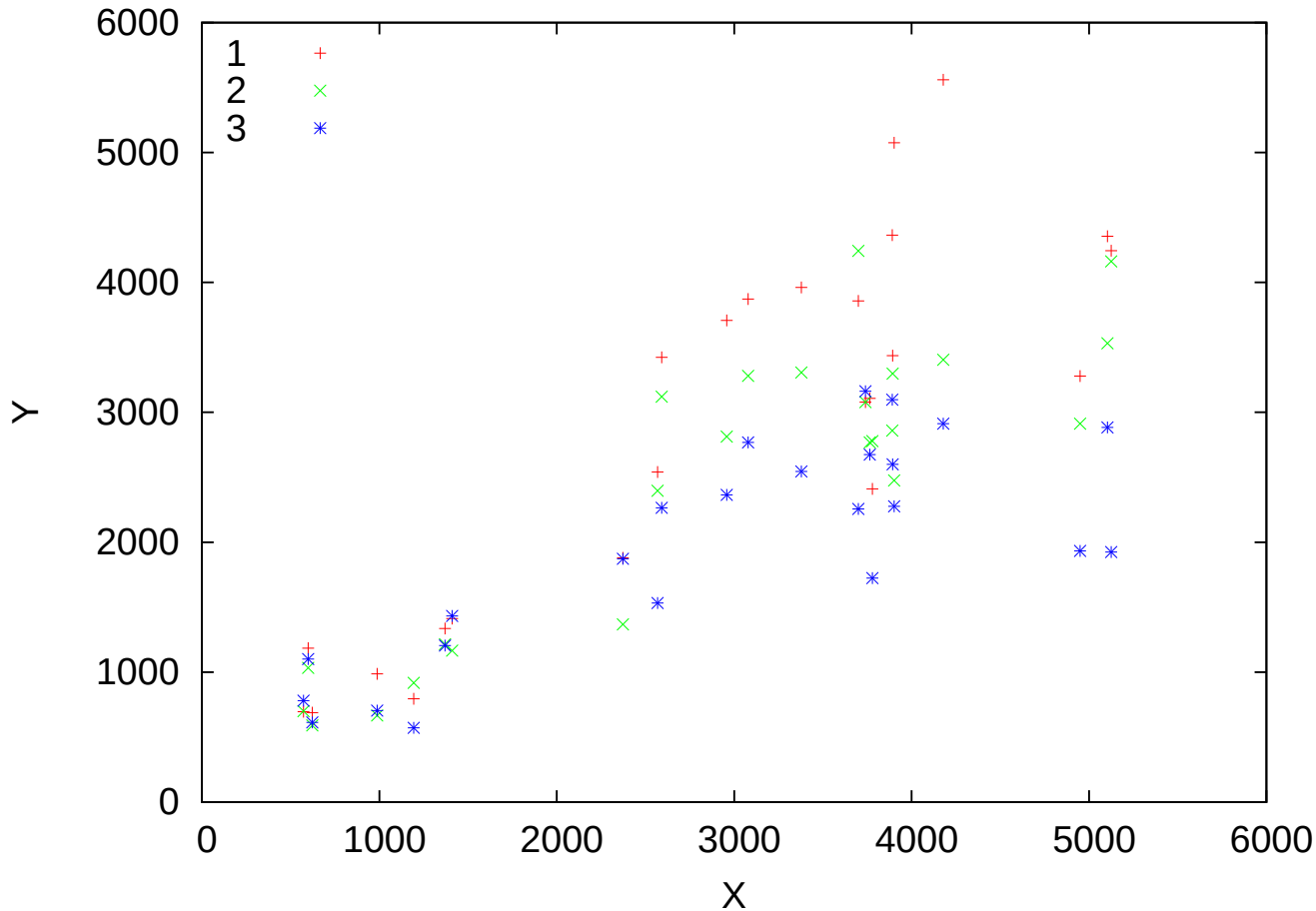


Clusters

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

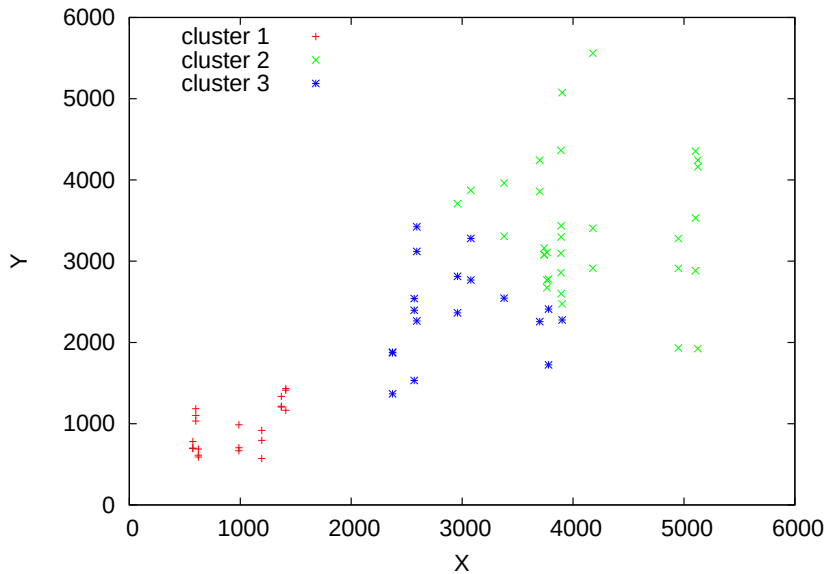
演習: k-means clustering

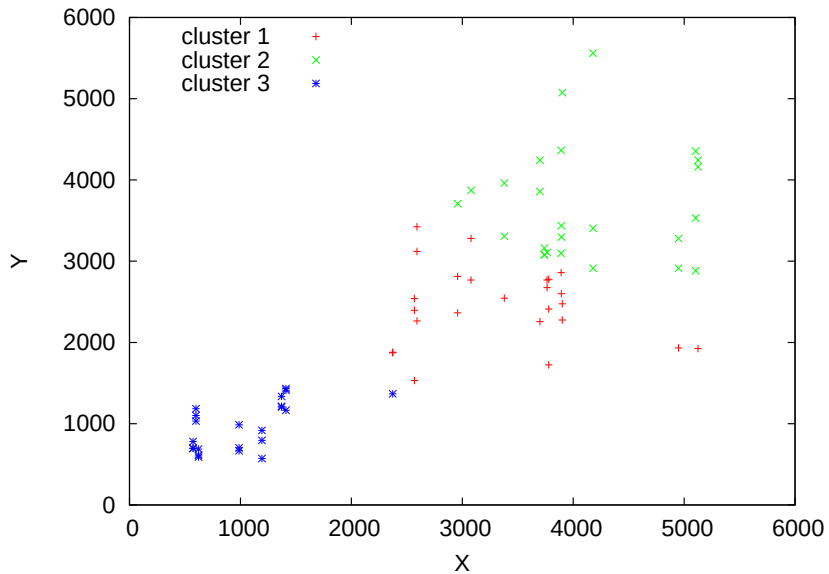
```
% ruby k-means.rb km-data.txt > km-results.txt
```



k-means clustering results

- 初期値によって結果が異なる





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 dsistance 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
```

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
```

gnuplot script

```
set key left
set xrange [0:6000]
set yrange [0:6000]
set xlabel "X"
set ylabel "Y"

plot "km-results.txt" using 1:($3==0?$2:1/0) title "cluster 1" with points, \
"km-results.txt" using 1:($3==1?$2:1/0) title "cluster 2" with points, \
"km-results.txt" using 1:($3==2?$2:1/0) title "cluster 3" with points
```

前回の演習: スパム判定

- 単純ベイズ分類器を使ったスパム判定
  - 「集合知プログラミング 6 章」のコードから作成
  - 日本語を扱うには単語に分割する形態素解析が必要

```
% ruby naivebayes.rb
classifying "quick rabbit" => good
classifying "quick money" => bad
```

前回の演習: 演習に使う単純ベイズ分類器

出現単語により文書が特定のカテゴリに分類される確率を求める

$$P(C)\prod_{i=1}^{n}P(x_i|C)$$

- $P(C)$: カテゴリの出現確率
- $\prod_{i=1}^{n}P(x_i|C)$: カテゴリにおける各単語の条件付き確率の積

もっとも確率の高いカテゴリを選ぶ

- 閾値： 2 番目のカテゴリより $thresh$ 倍高い必要

前回の演習: スパム判定スクリプト

- トレーニングと判定

```
# create a classifier instance
cl = NaiveBayes.new

# training
cl.train('Nobody owns the water.','good')
cl.train('the quick rabbit jumps fences','good')
cl.train('buy pharmaceuticals now','bad')
cl.train('make quick money at the online casino','bad')
cl.train('the quick brown fox jumps','good')

# classify
sample_data = [ "quick rabbit", "quick money" ]

sample_data.each do |s|
  print "classifying \"#{s}\" => "
  puts cl.classify(s, default="unknown")
end
```

前回の演習: Classifier Class (1/2)

```
# feature extraction
def getwords(doc)
  words = doc.split(/\W+/)
  words.map!{|w| w.downcase}
  words.select{|w| w.length < 20 && w.length > 2 }.uniq
end

# base class for classifier
class Classifier
  def initialize
    # initialize arrays for feature counts, category counts
    @fc, @cc = {}, {}
  end

  def getfeatures(doc)
    getwords(doc)
  end

  # increment feature/category count
  def incf(f, cat)
    @fc[f] ||= {}
    @fc[f][cat] ||= 0
    @fc[f][cat] += 1
  end

  # increment category count
  def incc(cat)
    @cc[cat] ||= 0
    @cc[cat] += 1
  end

  ...
```

前回の演習: Classifier Class (2/2)

```
  def fprob(f,cat)
    if catcount(cat) == 0
      return 0.0
    end

    # the total number of times this feature appeared in this
    # category divided by the total number of items in this category
    Float(fcount(f, cat)) / catcount(cat)
  end

  def weightedprob(f, cat, weight=1.0, ap=0.5)
    # calculate current probability
    basicprob = fprob(f, cat)
    # count the number of times this feature has appeared in all categories
    totals = 0
    categories.each do |c|
      totals += fcount(f,c)
    end
    # calculate the weighted average
    ((weight * ap) + (totals * basicprob)) / (weight + totals)
  end

  def train(item, cat)
    features = getfeatures(item)
    features.each do |f|
      incf(f, cat)
    end
    incc(cat)
  end
end
```

前回の演習: NaiveBayes Class

```
# naive baysian classifier
class NaiveBayes < Classifier
  def initialize
    super
    @thresholds = {}
  end

  def docprob(item, cat)
    features = getfeatures(item)
    # multiply the probabilities of all the features together
    p = 1.0
    features.each do |f|
      p *= weightedprob(f, cat)
    end
    return p
  end

  def prob(item, cat)
    catprob = Float(catcount(cat)) / totalcount
    docprob = docprob(item, cat)
    return docprob * catprob
  end

  def classify(item, default=nil)
    # find the category with the highest probability
    probs, max, best = {}, 0.0, nil
    categories.each do |cat|
      probs[cat] = prob(item, cat)
      if probs[cat] > max
        max = probs[cat]
        best = cat
      end
    end
    # make sure the probability exceeds threshold*next best
```

課題 2 解答: twitter データ解析

- ねらい: 大規模実データ処理の実践
- 課題用データ:
  - Kwak らによる 2009 年 7 月の twitter data、約 4000 万ユーザ分
    - http://an.kaist.ac.kr/traces/WWW2010.html
  - twitter_degrees.zip (164MB, 解凍後 550MB)
    - 各ユーザの ID、フォロー数、フォローワ数
  - numeric2screen.zip (365MB, 解凍後 756MB)
    - 各ユーザの ID、スクリーン名
- 提出項目
  1. twitter ユーザの following/follower 数分布の CCDF プロット
     - X 軸に following/follower 数を取り log-log プロット
  2. フォローワ数の多いトップ 30 ユーザの表
     - ランク、ユーザ ID、スクリーン名、フォロー数、フォローワ数
     - 2 つのファイルをソート、マージする作業
  3. オプション
     - その他の解析
  4. 考察
     - データから読みとれることを記述
- 提出形式: レポートをひとつの PDF ファイルにして SFC-SFS から提出
- 提出〆切: 2012 年 6 月 20 日

課題データについて

twitter_degrees.zip (164MB, 解凍後 550MB)

```
# id followings followers

12      586     1001061
13      243     1031830
14      106     8808
15      275     14342
16      273     218
17      192     6948
18      87      6532
20      912     1213787
21      495     9027
22      272     3791
...
```

numeric2screen.zip (365MB, 解凍後 756MB)

```
# id screenname

12 jack
13 biz
14 noah
15 crystal
16 jeremy
17 tonystubblebine
18 Adam
20 ev
21 dom
22 rabble
...
```

課題 提出物

CCDF プロット

- 第 5 回授業の演習参照
- 演習は 10000 ユーザのサンプル、課題は全ユーザ (2009 年時点)

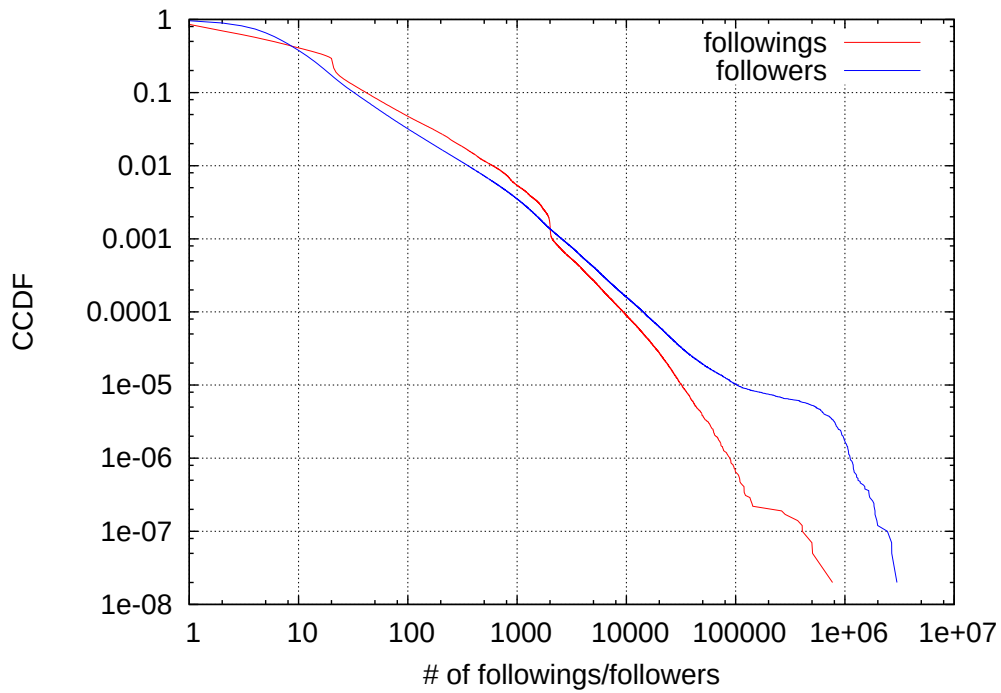
フォローワ数の多いトップ 30 ユーザの表

- ランク、ユーザ ID、スクリーン名、フォロー数、フォローワ数

```
# rank   id           screenname   followings followers

1      19058681       aplusk          183     2997469
2      15846407       TheEllenShow    26      2679639
3      16409683       britneyspears   406238  2674874
4      428333         cnnbrk          18      2450749
5      19397785       Oprah           15      1994926
6      783214         twitter         55      1959708
...
```

課題 2 解答: CCDF プロット



課題 2 解答: フォローワ数の多いトップ 30 ユーザの表

# rank	id	screenname	followings	followers
1	19058681	aplusk	183	2997469
2	15846407	TheEllenShow	26	2679639
3	16409683	britneyspears	406238	2674874
4	428333	cnnbrk	18	2450749
5	19397785	Oprah	15	1994926
6	783214	twitter	55	1959708
7	16190898	RyanSeacrest	137	1885782
8	813286	BarackObama	770155	1882889
9	19757371	johncmayer	64	1844499
10	17461978	THE_REAL_SHAQ	563	1843561
11	25365536	KimKardashian	73	1790771
12	19554706	mrskutcher	99	1691919
13	15485441	jimmyfallon	131	1668193
14	18220175	iamdiddy	173	1657119
15	16727535	lancearmstrong	103	1651207
16	807095	nytimes	177	1524048
17	18863815	coldplay	2633	1517067
18	27104736	mileycyrus	54	1477423
19	14075928	TheOnion	369569	1380160
20	17220934	algore	8	1377332
21	18091904	ashleytisdale	75	1318909
22	18222378	50cent	13	1318378
23	20536157	google	162	1278103
24	21879024	tonyhawk	118	1277163
25	19329393	PerezHilton	328	1269341
26	16827333	souljaboytellem	94	1241331
27	20	ev	912	1213787
28	972651	mashable	1934	1210996
29	26885308	ashsimpsonwentz	32	1200472
30	6273552	MCHammer	27413	1195089

まとめ

第 12 回 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 検索とランキング (7/3)

- 検索システム
- ページランク
- 演習: PageRank

補講予定

- 7/17 (水) 4 限 (14:45-16:15) $\epsilon 12$

参考文献

[1] Ruby official site. `http://www.ruby-lang.org/`

[2] gnuplot official site. `http://gnuplot.info/`

[3] Mark Crovella and Balachander Krishnamurthy. *Internet measurement: infrastructure, traffic, and applications*. Wiley, 2006.

[4] Pang-Ning Tan, Michael Steinbach and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 2006.

[5] Raj Jain. *The art of computer systems performance analysis*. Wiley, 1991.

[6] Toby Segaran. (當山仁健 鴨澤眞夫 訳). 集合知プログラミング. オライリージャパン. 2008.

[7] Chris Sanders. (高橋基信 宮本久仁男 監訳 岡真由美 訳). 実践パケット解析 第 *2* 版 — *Wireshark* を使ったトラブルシューティング. オライリージャパン. 2012.

[8] あきみち、空閑洋平. インターネットのカタチ. オーム社. 2011.

[9] 井上洋, 野澤昌弘. 例題で学ぶ統計的方法. 創成社, 2010.

[10] 平岡和幸, 掘玄. プログラミングのための確率統計. オーム社, 2009.