Internet Measurement and Data Analysis (13)

Kenjiro Cho

2011-12-21

review of previous class

Class 12 Measuring anomalies of the Internet

- anomaly detection
- spam filters
- Bayes' theorem
- exercise: anomaly detection

today's topics

Class 13 Data mining

- pattern extraction
- classification
- clustering
- exercise: clustering

data mining

- huge volume of data
  - difficult to handle with traditional methods
  - need to extract information hidden in data that is not readily evident
- Data Mining
  - huge volume, multi-dimensional diverse data, non-trivial distributions
  - methods often derived from ideas in machine learning, AI, pattern recognition, statistics, database, signal processing
- data processing becomes practical by growing computing power (e.g., cloud computing)

Data Mining methods

definition: non-trivial extraction of implicit, previously unknown and potentially useful information from data

- pattern extraction: find existing models and patterns in data
  - correlation
  - time-series
- classification: automatically create new classes that do not exist in the original data
  - rule-based methods
  - naive Bayesian filter
  - neural networks
  - support vector machine (SVM)
  - dimensionality reduction (e.g., PCA)
- clustering: compute the distance (or similarity) between data points and group them
  - distance based, density based, graph based
  - k-means, DBSCAN
- anomaly detection: find deviation from normal state using statistical methods
  - univariate, multivariate
  - outlier detection

distances

various distances

- Euclidean distance
- standardized Euclidean distance
- Minkowski distance
- Mahalanobis distance

similarities

- binary vector similarities
- n-dimensional vector similarities

properties of distance

a metric of distance $d(x, y)$ between 2 points (x, y) in space
positivity

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

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

symmetry

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

triangle inequality

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

Euclidean distance

word “distance” usually means “Euclidean distance”
a distance of 2 points (x, y) in a n-dimensional space

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

standardized Euclidean distance

- when variances are different among variables, distances are affected.
- standard Euclidean distance: normalized by dividing the Euclidean distance by the variance of each variable

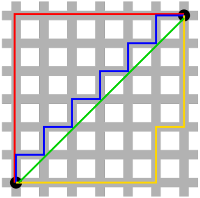
$$d(x, y) = \sqrt{\sum_{k=1}^{n} \frac{(x_k - y_k)^2}{s_k^2}}$$

Minkowski distance

generalization of Euclidean distance: as parameter r grows, a short cut crossing different axes is preferred more

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

- $r = 1$: Manhattan distance
  - Hamming distance: for 2 strings of equal length, the number of positions at which the corresponding symbols are different.
  - example: the hamming distance of 111111 and 101010 is 3
- $r = 2$: Euclidean distance



Manhattan distance vs. Euclidean distance

Mahalanobis distance

a distance that takes correlations into account, when correlation exists between variables

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

here, Σ^{-1} is the inverse matrix of its covariance matrix

similarities

similarity

- numerical measure of how alike 2 data objects are

properties of similarity

positivity

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

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

symmetry

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

in general, triangle inequality does not apply to similarities

similarity between binary vectors

Jaccard coefficient

- used for similarity between binary vectors in which the occurrences of 1 is much smaller than the occurrences of 0
- example: as a metric of similarity by occurrences of words in documents
- many words do not appear in both documents $\Rightarrow$ not considered
- the following table shows the relationship of each item

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

Jaccard coefficient:

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

similarity between vectors

similarity between (non-binary) vectors

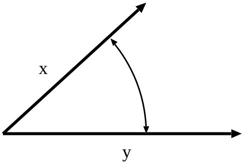
- example: similarity of documents where frequencies of words are also taken into consideration

cosine similarity

- take the angle (cosine) of (x, y) of vectors
- normalized by the length of the vector $\Rightarrow$ length is not considered

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

$x \cdot y = \sum_{k=1}^{n} x_k y_k$: product of vectors
$\|x\| = \sqrt{\sum_{k=1}^{n} x_k^2} = \sqrt{x \cdot x}$: length of the vector



example: cosine similarity

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

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

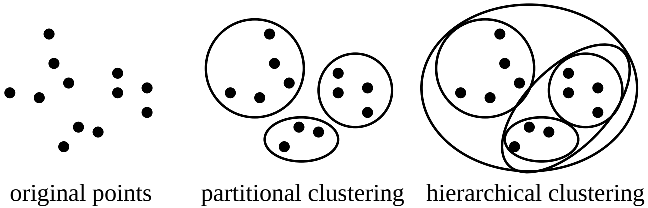
$$cos(x, y) = \frac{5}{6.481 * 2.449} = 0.315$$

clustering

compute the distance (or similarity) of variables to make them into groups

- to classify and understand data
- to summarize data

- partitional clustering
  - k-means method
- hierarchical clustering
  - MST method
  - DBSCAN method



k-means method

- partitional clustering
- specify the number of cluster, k
- basic algorithm is simple
  - each cluster has centroid (usually mean)
  - assign each object to the closest cluster
  - repeat re-computation of centroids and cluster assignments
- limitations
  - need to specify the number of clusters, k, beforehand
  - sensitive to the selection of initial points
  - clusters are supposed to have similar sizes and densities, and a round shape
  - sensitive to outliers

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
```

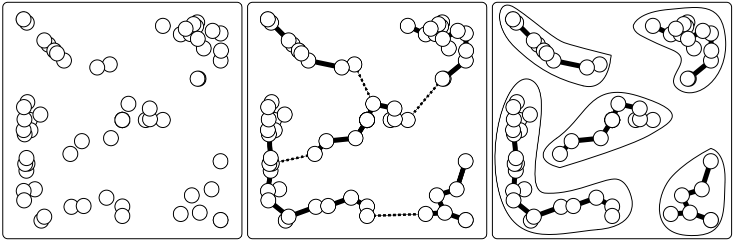
hierarchical clustering

- generate clusters using a tree structure
  - the cluster structure can be explained by the tree
- no need to specify the number of clusters beforehand
- 2 approaches
  - agglomerative: start with data points as individual clusters, and repeat merging the closest clusters
  - divisive: start with one all-inclusive cluster, and repeat splitting clusters

MST clustering

Minimum Spanning Tree clustering

- divisive hierarchical clustering
- start with an arbitrary point, and create MST
- repeat dividing clusters by removing the longest edge



(1) (2) (3)

DBSCAN

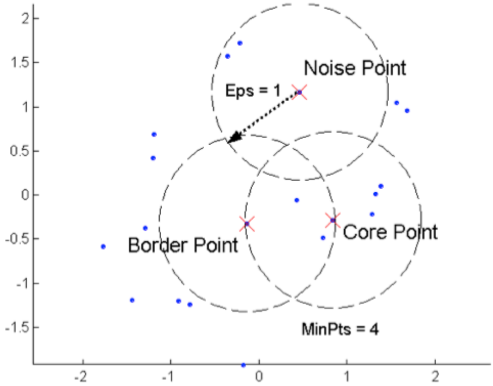
Density-Based Spatial Clustering

- density-based: combine data points within the specified distance
- can extract arbitrary (non-round) shapes of clusters
- robust against noise and outliers
- distance threshold *Eps* and point threshold *MinPts*
    - Core points: within the distance *Eps*, more than *MinPts* neighbors exist
    - Border points: not Core, but have a core within the distance *Eps*
    - Noise points: have no core within the distance *Eps*
- limitations: clusters with different densities, or with large number of parameters

DBSCAN algorithm:

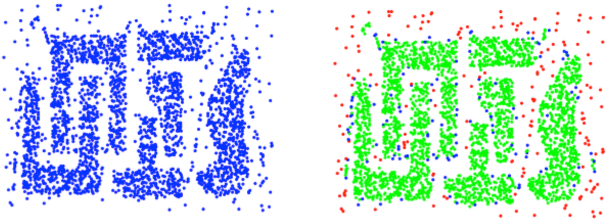
1: label all points as core, border, or noise points
2: eliminate noise points
3: put an edge between all core points that are within *Eps* of each other
4: make each group of connected core points into a separate cluster
5: assign each border point to one of the clusters of its associated core points

DBSCAN: Core, Border, and Noise Points



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

DBSCAN: example of Core, Border, and Noise Points



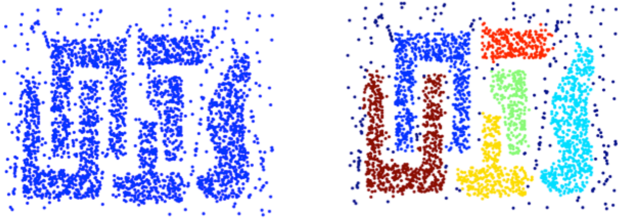
Original Points

Point types: core, border and noise

Eps = 10, MinPts = 4

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

DBSCAN: example clusters



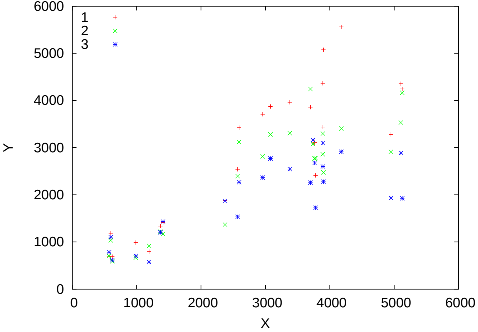
Clusters

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

exercise: k-means clustering

- data: hourly traffic for Monday vs. Wednesday/Friday/Sunday

```
% cat km-1.txt km-2.txt km-3.txt | ruby k-means.rb | \
  sort -k3,3 -s -n > km-results.txt
```



k-means code (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
```

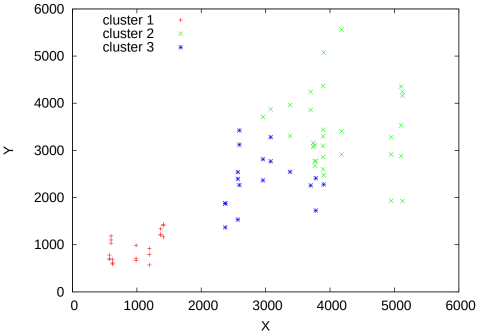
k-means code (2/2)

```
  # 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 results

- different results with different initial values

```
set key left
set xrange [0:6000]
set yrange [0:6000]
set xlabel "X"
set ylabel "Y"
plot "km-c1.txt" using 1:2 title "cluster 1" with points, \
"km-c2.txt" using 1:2 title "cluster 2" with points, \
"km-c3.txt" using 1:2 title "cluster 3" with points
```



final report

- select A or B
  - A. web access log analysis
  - B. free topic
- up to 8 pages in the PDF format
- submission via SFC-SFS by 2012-01-25 (Wed) 23:59

final report (cont'd)

A. web access log analysis
  - data: apache log (combined log format) used in Class 3
  - from a JAIST server, access log for 24 hours
    `http://www.iijlab.net/~kjc/classes/sfc2011f-measurement/sample_access_log.bz2`
  - write a script to extract the access count of each unique content, and plot the distribution in a log-log plot
  - optionally, do other analysis
  - the report should include (1) your script to extract the access counts, (2) a plot of the access count distribution, and (3) your analysis of the results

B. free topic
  - select a topic by yourself
  - the topic is not necessarily on networking
  - but the report should include some form of data analysis and discussion about data and results

summary

Class 13 Data mining

- pattern extraction
- classification
- clustering
- exercise: clustering

next class

Class 14 Scalable measurement and analysis (1/11)

- distributed parallel processing
- cloud technology
- exercise: large-scale data processing