CSE 332

INTRODUCTION TO VISUALIZATION

CLUSTER ANALYSIS

KLAUS MUELLER

COMPUTER SCIENCE DEPARTMENT
STONY BROOK UNIVERSITY AND SUNY KOREA
<table>
<thead>
<tr>
<th>Lecture</th>
<th>Topic</th>
<th>Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intro, schedule, and logistics</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Basic tasks and data types</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Data sources and what to look out for</td>
<td>Project 1 out</td>
</tr>
<tr>
<td>4</td>
<td>Data preparation and representation</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Data reduction, notion of similarity and distance</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Dimension reduction</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Bias, introduction to plotly and plotly Dash</td>
<td>Project 2 out</td>
</tr>
<tr>
<td>8</td>
<td>Visual perception and cognition</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Visual design and aesthetic</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Introduction to D3</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Cluster analysis</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>High-dimensional data, dimensionality reduction</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Visualization of spatial data: volume visualization intro</td>
<td>Project 3 out</td>
</tr>
<tr>
<td>14</td>
<td>Introduction to GPU programming</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Visualization of spatial data: raycasting, transfer functions</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Illumination and isosurface rendering</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Midterm</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Scientific visualization</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Non-photorealistic and illustrative rendering</td>
<td>Project 4 out</td>
</tr>
<tr>
<td>20</td>
<td>Midterm discussion</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Principles of interaction</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Visual analytics and the visual sense making process</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Visualization of graphs and hierarchies</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Visualization of time-varying and streaming data</td>
<td>Project 5 out</td>
</tr>
<tr>
<td>25</td>
<td>Maps</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Memorable visualizations, visual embellishments</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Evaluation and user studies</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Narrative visualization, storytelling, data journalism, XAI</td>
<td></td>
</tr>
</tbody>
</table>
Finding The Needle – Cluster Analysis

Data summarization
- data reduction
- cluster centers, shapes, and statistics

Customer segmentation
- collaborative filtering

Social network analysis
- find similar groups of friends (communities)

Precursor to other analysis
- use as a preprocessing step for classification and outlier detection
With 1,000s of attributes (dimensions) which ones are relevant and which one are not?

avoid

keep

histogram of pairwise distances in N-D space

(a) Uniform Data

(b) Clustered data

(c) Distance distribution (uniform)

(d) Distance distribution (clustered)
How to measure attribute “worthiness”
- use entropy

Entropy
- originates in thermodynamics
- measures lack of order or predictability

Entropy in statistics and information theory
- has a value of 1 for uniform distributions (not predictable)
- knowing the value has a lot of information (high surprise)
- a value of 0 for a constant value (fully predicable)
- knowing the value has zero information (low surprise)
Assume $m$ bins, $1 \leq i \leq m$:

$$E = -\sum_{i=1}^{m} [p_i \log(p_i) + (1 - p_i) \log(1 - p_i)].$$

Algorithm:
- start with all attributes and compute distance entropy
- greedily eliminate attributes that reduce the entropy the most
- stop when entropy no longer reduces or even increases
Hierarchical Clustering

Two options:
- top down (divisive)
- bottom up (agglomerative)
Algorithm AgglomerativeMerge(Data: D)
begin
    Initialize $n \times n$ distance matrix $M$ using $D$;
    repeat
        Pick closest pair of clusters $i$ and $j$ using $M$;
        Merge clusters $i$ and $j$;
        Delete rows/columns $i$ and $j$ from $M$ and create a new row and column for newly merged cluster;
        Update the entries of new row and column of $M$;
    until termination criterion;
    return current merged cluster set;
end

How to merge?
Single linkage
- distance = minimum distance between all $m_i \cdot m_j$ pairs of objects
- joins the closest pair

Worst (complete) linkage
- distance = maximum distance between all $m_i \cdot m_j$ pairs of objects
- joins the pair furthest apart

Group-average linkage
- distance = average distance between all object pairs in the groups

Other methods:
- closest centroid, variance-minimazation, Ward’s method
Centroid-based methods tend to merge large clusters

Single linkage method can merge chains of closely related points to discover clusters of arbitrary shape

- but can also (inappropriately) merge two unrelated clusters, when the chaining is caused by noisy points between two clusters

(a) Good case with no noise  (b) Bad case with noise
Complete (worst-case) linkage method tends to create spherical clusters with similar diameter

- will break up the larger clusters into smaller spheres
- also gives too much importance to data points at the noisy fringes of a cluster

The group average, variance, and Ward’s methods are more robust to noise due to the use of multiple linkages in the distance computation

Hierarchical methods are sensitive to a small number of mistakes made during the merging process

- can be due to noise
- no way to undo these mistakes
Highly-cited density-based hierarchical clustering algorithm (Ester et al. 1996)

- clusters are defined as density-connected sets
- epsilon-distance neighbor criterion (Eps)
  \[ N_{Eps}(p) = \{ q \in D \mid \text{dist}(p, q) \leq Eps \} \]
- minimum point cluster membership and core point (MinPts)
  \[ |N_{Eps}(q)| \geq \text{MinPts} \]
- notions of density-connected & density-reachable (direct, indirect)
- a point \( p \) is directly density-reachable from a point \( q \) wrt. \( Eps, \text{MinPts} \) if
  \[ p \in N_{Eps}(q) \text{ and } |N_{Eps}(q)| \geq \text{MinPts} \] (core point condition)
DBSCAN

(a) p: border point
 q: core point

(b) p directly density-reachable from q
 q not directly density-reachable from p

(c) p density-reachable from q
 q not density-reachable from p

(d) p and q density-connected to each other by ϵ
Probabilistic Extension to K-Means

Different cluster analysis results on "mouse" data set:
Original Data  k-Means Clustering  EM Clustering
The distance between a point $P$ and a distribution $D$ measures how many standard deviations $P$ is away from the mean of $D$.

- $S$ is the covariance matrix of the distribution $D$.
- The Mahalanobis distance $D_M$ of a point $x$ to a cluster center $\mu$ is

$$D_M(x) = \sqrt{(x - \mu)^T S^{-1} (x - \mu)}.$$ 

- $x$ and $\mu$ are $N$-dimensional vectors.
- $S$ is a $N \times N$ matrix.
- The outcome $D_M(x)$ is a single-dimensional number.
Better match for point distributions
- overlapping clusters are now possible
- better match with real world?
- Gaussian mixtures

Need a probabilistic algorithm
- Expectation-Maximization
EM Algorithm (Mixture Model)

- Initialize K cluster centers
- Iterate between two steps
  - **Expectation step:** assign $n$ points to $m$ clusters/classes
    
    $$P(d_i \in c_k) = w_k \Pr(d_i | c_k) / \sum_j w_j \Pr(d_i | c_j)$$
    
    $$\sum_i \Pr(d_i \in c_k) = \frac{w_k}{n} = \text{probability of class } c_k$$
  
  - **Maximation step:** estimate model parameters
    
    $$\mu_k = \frac{1}{n} \sum_{i=1}^{n} \frac{d_i \Pr(d_i \in c_k)}{\sum_j \Pr(d_i \in c_j)}$$
    
    do similar also for covariance matrix $S$
Iteration 1

The cluster means are randomly assigned
Mean Likelihood = -11.13452288716779

Iteration 25
T-SNE

t-distributed stochastic neighbor embedding
T-SNE Distance Metric

Uses the following density-based (probabilistic) distance metric

\[ p_{ji} = \frac{\exp\left(-\frac{|x_i - x_j|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{|x_i - x_k|^2}{2\sigma_i^2}\right)} \]

Measures how (relatively) close \( x_j \) is from \( x_i \), considering a Gaussian distribution around \( x_i \) with a given variance \( \sigma_i^2 \).

- this variance is different for every point
- it is chosen such that points in dense areas are given a smaller variance than points in sparse areas
T-SNE IMPLEMENTATION

Use a symmetrized version of the conditional similarity:

\[ p_{ij} = \frac{p_{ji} + p_{ij}}{2N} \]

Similarity (distance) metric for mapped points:

\[ q_{ij} = \frac{f(|x_i - x_j|)}{\sum_{k \neq i} f(|x_i - x_k|)} \quad \text{with} \quad f(z) = \frac{1}{1+z^2} \]

This uses the t-student distribution with one degree of freedom, or Cauchy distribution, instead of a Gaussian distribution.
Can use mass-spring system enforcing minimum of $|p_{ij} - q_{ij}|$

The classic *handwritten digits* datasets. It contains 1,797 images with $8 \times 8 = 64$ pixels each.
Animated layout
More information

See this webpage
Rectangular data set with a temporal component

- assume you have these data for each year
- how to handle that, you might ask?
Assume for now we have
- two attributes (burglary, theft)
- both observed over time

Can visualize
- but each point is a time series!
Needed it for clustering
  - recall Euclidean, correlation, cosine distances

Similarity Measures

- similarity of two states
- similarity of two crimes
- similarity of two crimes in a given state over time
- similarity of two states for a given crime over time
- two time series
What can be clustered with these measures?

- crimes (averaged over time)
- states (averaged over time)
- crimes in a given state (taking time series into account)
- states for a given crime (taking time series into account)

Can we get more inclusive?

- cluster crimes but including the time series characteristics
- cluster states but including the time series characteristics

Capture more information about their time series when you compare two data points

- compute the similarity of two crimes by summing the times-series similarities for each state
- compute the similarity of two states by summing the times-series similarities for each crime
**Time-series aware similarity (distance) $S_{tsa}$ for a pair of states**

For a given pair of states $i, j$

- For each crime $c$
  - Compute the time series similarity $\rightarrow \text{sim}_t(c)$
  - Sum all $\text{sim}_t(c) \rightarrow S_{tsa}(i,j)$

**$S_{tsa}$ for a pair of crimes**

For each pair of crimes $i, j$

- For each state $s$
  - Compute the time series similarity $\rightarrow \text{sim}_t(s)$
  - Sum all $\text{sim}_t(s) \rightarrow S_{tsa}(i,j)$

If the time series are aligned for all states then the $S_{tsa}$ will be high and the two crimes have very similar time behaviors nationwide.

*similarity* could be some measure of correlation of the two time series vectors.
The time series might not be aligned
- one crime might cause another
- can apply dynamic time warping (see next)

You may (also) have a geospatial component in your data
- can use them as a regular attribute (encoded by an ID)
- can you make them more continuous and linearly ordered?
- use a space filling curve (see next)

You may want to just keep time instances as separate entities
- that will work too
- then you might discover clusters that are sensitive to time
- or you can see how the years relate to another along a trajectory
- as a general rule, when you visualize multivariate data first decide what you will put into the rectangular data matrix (samples, attributes)
Standard pairwise distance

\[
Dist(\mathbf{X}, \mathbf{Y}) = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}
\]

Shortcomings:
- designed for time series of equal length
- cannot address distortions on the temporal (contextual) attributes
Can better accommodate local mismatches

Three constraints

- no skipping of beginning or ends of either sequence
- continuity – no jumps
- monotonicity – can’t go back in time
DTW – Find The Minimum Cost Path

Euclidian

DTW
DTW – Find The Minimum Cost Path

DTW

Compute using dynamic programming

Available in python
Convert a geographical map into a grid map

Linearize using a space-filling curve (Hilbert curve)
Cluster analysis

- detect and eliminate irrelevant (noisy) attributes using entropy
- build a cluster hierarchy bottom-up or top-down
- different metrics to join points and clusters
- the DBSCAN algorithm for more noise-robust clustering of arbitrary shapes
- EM-ML probabilistic clustering as an extension of k-means for less sensitivity to noise and overlapping clusters
- more sophisticated density-based clustering and dimension reduction using t-SNE

Adding temporal information

- many different ways to account for time-variant data
- dynamic time warping to align time series
- space filling curves to order geographical information