Clustering Preliminaries

Applications

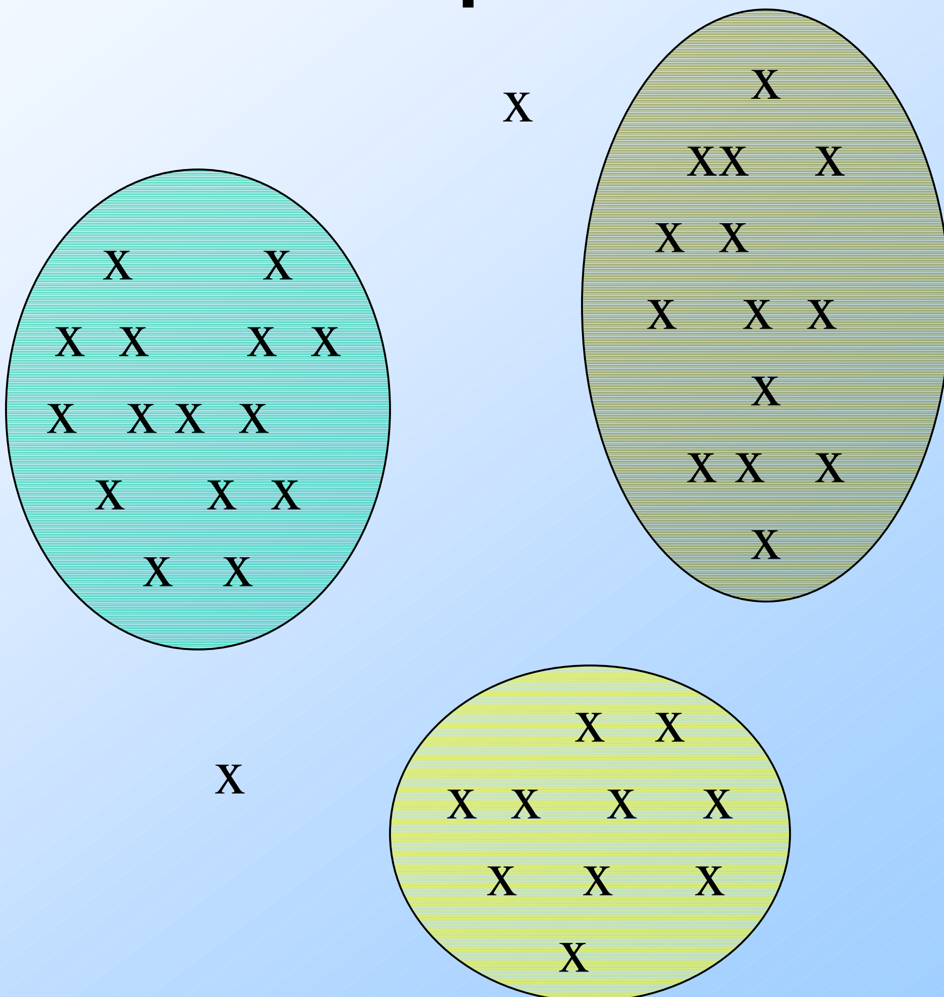
Euclidean/Non-Euclidean Spaces

Distance Measures

The Problem of Clustering

- Given a set of points, with a notion of distance between points, group the points into some number of *clusters*, so that members of a cluster are in some sense as close to each other as possible.

Example



Problems With Clustering

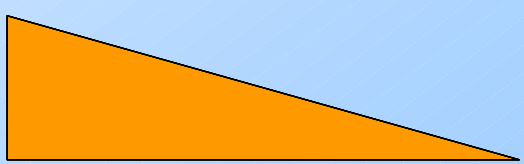
- Clustering in two dimensions looks easy.
- Clustering small amounts of data looks easy.
- And in most cases, looks are *not* deceiving.

The Curse of Dimensionality

- Many applications involve not 2, but 10 or 10,000 dimensions.
- High-dimensional spaces look different: almost all pairs of points are at about the same distance.

Example: Curse of Dimensionality

- Assume random points within a bounding box, e.g., values between 0 and 1 in each dimension.
- In 2 dimensions: a variety of distances between 0 and 1.41.
- In 10,000 dimensions, the difference in any one dimension is distributed as a triangle.



Example – Continued

- The law of large numbers applies.
- Actual distance between two random points is the sqrt of the sum of squares of essentially the same set of differences.

Example High-Dimension Application: SkyCat

- A catalog of 2 billion "sky objects" represents objects by their radiation in 7 dimensions (frequency bands).
- Problem: cluster into similar objects, e.g., galaxies, nearby stars, quasars, etc.
- Sloan Sky Survey is a newer, better version.

Example: Clustering CD's (Collaborative Filtering)

- Intuitively: music divides into categories, and customers prefer a few categories.
  - But what are categories really?
- Represent a CD by the customers who bought it.
- Similar CD's have similar sets of customers, and vice-versa.

The Space of CD's

- Think of a space with one dimension for each customer.
  - Values in a dimension may be 0 or 1 only.
- A CD's point in this space is $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ iff the i^{th} customer bought the CD.
  - Compare with boolean matrix: rows = customers; cols. = CD's.

Space of CD's – (2)

- For Amazon, the dimension count is tens of millions.
- An alternative: use minhashing/LSH to get Jaccard similarity between "close" CD's.
- 1 minus Jaccard similarity can serve as a (non-Euclidean) distance.

Example: Clustering Documents

- Represent a document by a vector $(x_1, x_2,\ldots, x_k)$, where $x_i = 1$ iff the i th word (in some order) appears in the document.
  - It actually doesn't matter if k is infinite; i.e., we don't limit the set of words.
- Documents with similar sets of words may be about the same topic.

Aside: Cosine, Jaccard, and Euclidean Distances

- As with CD's we have a choice when we think of documents as sets of words or shingles:
  1. Sets as vectors: measure similarity by the cosine distance.
  2. Sets as sets: measure similarity by the Jaccard distance.
  3. Sets as points: measure similarity by Euclidean distance.

Example: DNA Sequences

- Objects are sequences of {C,A,T,G}.
- Distance between sequences is *edit distance*, the minimum number of inserts and deletes needed to turn one into the other.
- Note there is a "distance," but no convenient space in which points "live."

Distance Measures

- Each clustering problem is based on some kind of "distance" between points.
- Two major classes of distance measure:
  1. *Euclidean*
  2. *Non-Euclidean*

Euclidean Vs. Non-Euclidean

- A *Euclidean space* has some number of real-valued dimensions and "dense" points.
  - There is a notion of "average" of two points.
  - A *Euclidean distance* is based on the locations of points in such a space.
- A *Non-Euclidean distance* is based on properties of points, but not their "location" in a space.

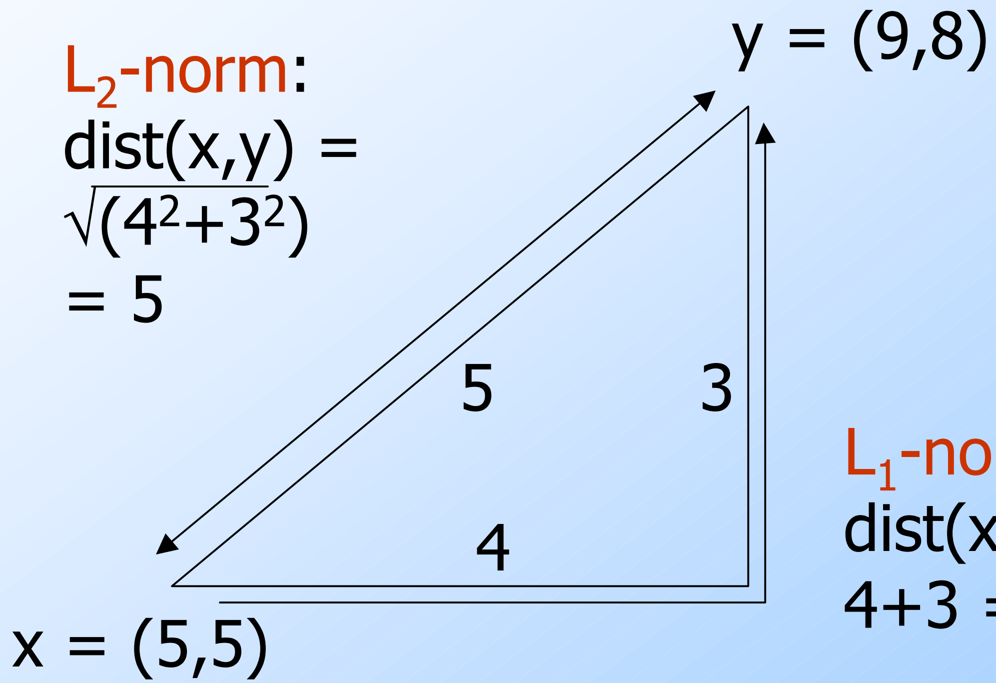
Axioms of a Distance Measure

- *d* is a *distance measure* if it is a function from pairs of points to real numbers such that:
  1. $d(x,y) \geq 0$.
  2. $d(x,y) = 0$ iff $x = y$.
  3. $d(x,y) = d(y,x)$.
  4. $d(x,y) \leq d(x,z) + d(z,y)$ (*triangle inequality*).

Some Euclidean Distances

- *L_2 norm* : d(x,y) = square root of the sum of the squares of the differences between *x* and *y* in each dimension.
  - The most common notion of "distance."
- *L_1 norm* : sum of the differences in each dimension.
  - *Manhattan distance* = distance if you had to travel along coordinates only.

Examples of Euclidean Distances



L$_2$-norm:
dist(x,y) = $\sqrt{(4^2+3^2)}$
= 5

L$_1$-norm:
dist(x,y) =
4+3 = 7

Another Euclidean Distance

- *L_∞ norm* : d(x,y) = the maximum of the differences between *x* and *y* in any dimension.
- Note: the maximum is the limit as *n* goes to ∞ of what you get by taking the *n*th power of the differences, summing and taking the *n*th root.

Non-Euclidean Distances

- *Jaccard distance* for sets = 1 minus ratio of sizes of intersection and union.
- *Cosine distance* = angle between vectors from the origin to the points in question.
- *Edit distance* = number of inserts and deletes to change one string into another.

Jaccard Distance for Sets (Bit-Vectors)

- Example: p_1 = 10111; p_2 = 10011.
- Size of intersection = 3; size of union = 4, Jaccard similarity (not distance) = 3/4.
- d(x,y) = 1 – (Jaccard similarity) = 1/4.

Why J.D. Is a Distance Measure

- $d(x,x) = 0$ because $x \cap x = x \cup x$.
- $d(x,y) = d(y,x)$ because union and intersection are symmetric.
- $d(x,y) \geq 0$ because $|x \cap y| \leq |x \cup y|$.
- $d(x,y) \leq d(x,z) + d(z,y)$ trickier – next slide.

Triangle Inequality for J.D.

$$1 - \frac{|x \cap z|}{|x \cup z|} + 1 - \frac{|y \cap z|}{|y \cup z|} \geq 1 - \frac{|x \cap y|}{|x \cup y|}$$

- Remember: $|a \cap b|/|a \cup b|$ = probability that minhash(a) = minhash(b).
- Thus, $1 - |a \cap b|/|a \cup b|$ = probability that minhash(a) $\neq$ minhash(b).

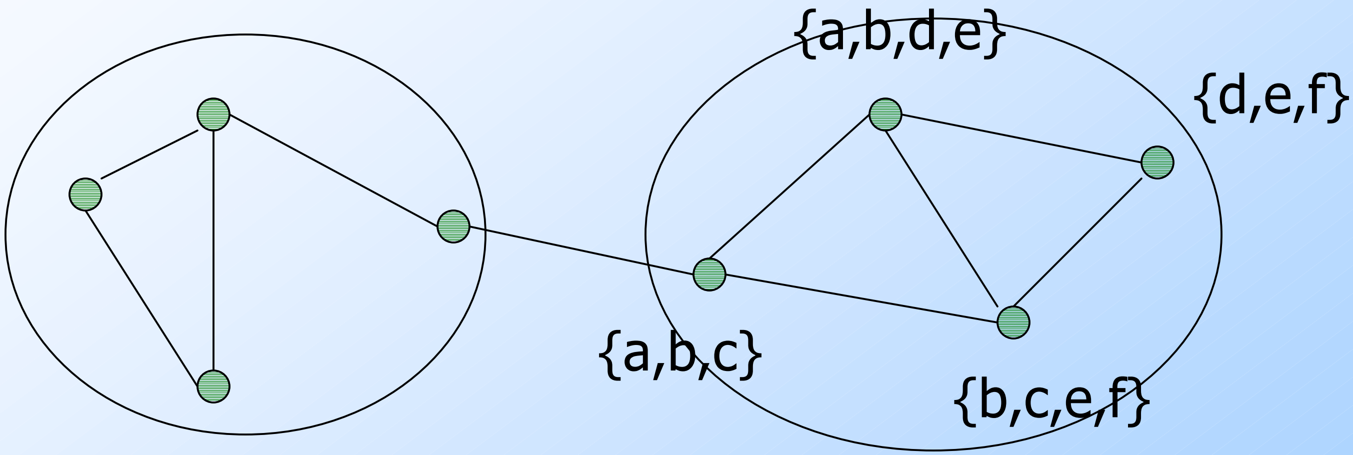
Triangle Inequality – (2)

- Claim: prob[minhash(x) ≠ minhash(y)] ≤ prob[minhash(x) ≠ minhash(z)] + prob[minhash(z) ≠ minhash(y)]
- Proof: whenever minhash(x) ≠ minhash(y), at least one of minhash(x) ≠ minhash(z) and minhash(z) ≠ minhash(y) must be true.

Similar Sets and Clustering

- We can use minhashing + LSH to find quickly those pairs of sets with low Jaccard distance.
- We can cluster sets (points) using J.D.
- But we only know some distances – the low ones.
- Thus, clusters are not always connected components.

Example: Clustering + J.D.

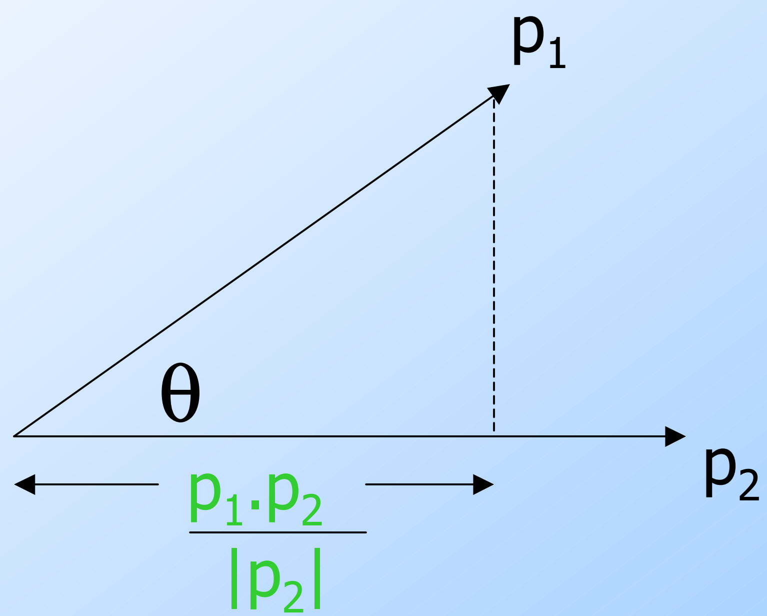


Similarity threshold = 1/3;
distance $\leq$ 2/3

Cosine Distance

- Think of a point as a vector from the origin (0,0,…,0) to its location.
- Two points' vectors make an angle, whose cosine is the normalized dot-product of the vectors: $p_1.p_2/|p_2||p_1|$.
  - Example: p_1 = 00111; p_2 = 10011.
  - $p_1.p_2 = 2$; $|p_1| = |p_2| = \sqrt{3}$.
  - $\cos(\theta) = 2/3$; θ is about 48 degrees.

Cosine-Measure Diagram



$d(p_1, p_2) = \theta = \arccos(p_1.p_2/|p_2||p_1|)$

Why C.D. Is a Distance Measure

- d(x,x) = 0 because arccos(1) = 0.
- d(x,y) = d(y,x) by symmetry.
- d(x,y) $\geq$ 0 because angles are chosen to be in the range 0 to 180 degrees.
- Triangle inequality: physical reasoning. If I rotate an angle from *x* to *z* and then from *z* to *y*, I can't rotate less than from *x* to *y*.

Edit Distance

- The *edit distance* of two strings is the number of inserts and deletes of characters needed to turn one into the other. Equivalently:
- d(x,y) = |x| + |y| - 2|LCS(x,y)|.
  - LCS = *longest common subsequence* = any longest string obtained both by deleting from *x* and deleting from *y*.

Example: LCS

- x = *abcde* ; y = *bcduve*.
- Turn x into y by deleting *a*, then inserting *u* and *v* after *d*.
  - Edit distance = 3.
- Or, LCS(x,y) = *bcde*.
- Note: |x| + |y| - 2|LCS(x,y)| = 5 + 6 –2*4 = 3 = edit distance.

Why Edit Distance Is a Distance Measure

- d(x,x) = 0 because 0 edits suffice.
- d(x,y) = d(y,x) because insert/delete are inverses of each other.
- d(x,y) $\geq$ 0: no notion of negative edits.
- Triangle inequality: changing *x* to *z* and then to *y* is one way to change *x* to *y*.

Variant Edit Distances

- Allow insert, delete, and *mutate*.
  - Change one character into another.
- Minimum number of inserts, deletes, and mutates also forms a distance measure.
- Ditto for any set of operations on strings.
  - Example: substring reversal OK for DNA sequences