```
L7 -- Distances
[Jeff Phillips - Utah - Data Mining]

What makes a good distance?

a distance d(a,b) is a metric if
 * d(a,b) >= 0                (non-negativity)
 * d(a,b) = 0  iff a=b        (identity)
 * d(a,b) = d(b,a)            (symmetry)
 * d(a,b) <= d(a,c) + d(c,b)  (triangle inequality)

Not all distance follow this; but very convenient.



-----------------------
Euclidean Distance   (in R^d)
  a = (a1,a2,a3,...,ad)
  b = (b1,b2,b3,...,bd)

d(a,b) = sqrt(sum_{i=1}^d (ai-bi)^2)
       = L2(a,b)
       = ||a - b||_2

Lp(a,b) = (sum_{i=1}^d |ai-bi|^p)^{1/p}
 * L1 = "manhattan distance"
            LSH via 1-stable distributions
              --> Cauchy distribution (1/pi)(1/1+x^2)

 * L0 = number of differences
        (used for comparing min-hash signatures)
        "Hamming distance"
           LSH via minhash (bounded t=d)
             almost 1-stable, can use close by .001-stable, but inefficient

 * Linfty = maximum distance
          = max(sum_{i=1}^d (ai-bi))
        "rotation of L1"



Is L2(a,b) a metric?
  non-negativity:  square makes bigger than 0
  identity: if any coordinate different ->  >0
  symmetry: (ai-bi) = (bi-ai)
  triangle: <draw triangle :) >
```

```
-----------------------
Jaccard Distance:
  d_J(a,b) = 1-Jac(a,b)

  Venn Diagram  -->  Symmetric Difference / Union

  non-negativity:  intersection cannot exceed union
  identity: a cap a = a cup a = a
            if a != b, then a cap b strict subset a cup b
  symmetry: yes
  triangle: d_J(a,b) <= d_J(a,c) + d_J(c,b)
       ==>  forall c, Pr[h(a) != h(c)] + Pr[h(c) != h(b)] >= Pr[h(a) != h(b)]
               if c = a = b --> 0 + 0 >= 0
               else |a\c|/|a| + |b\c|/|b| >= |a symdif b|/|a cup b|
                   -> (|a\c| + |b\c|)/|a cup b| >= |a symdif b|/|a cup b|
                      since |a|, |b| <= |a cup b|
                   -> |a\c| + |b\c| >= |a symdif b|
                      which holds, and is only = if c = a cup b

-----------------------
Cosine Distance
  "angle between vectors"
  cos(a,b) = arccos(sum_{i=1}^d a_i * b_i)  \in [0,pi]

  treats points a,b as "vectors".  Does not care of magnitude, only
"direction"

  non-negativity:  by definition
  identity: treats multiples of vectors as equivalent (make unit vectors)
  symmetry: a_i * b_i = b_i * a_i
  triangle: geodesic distance on unit sphere
            shortest rotation

Good when want to ignore scale of objects.

-----
LSH:  Choose random vector v
      if <v, a> > 0  h(a) = +1
      else           h(a) = -1
 Can make v = {-1,+1}^d
Same as Jaccard, but [0,pi] instead of [0,1]
  (gamma,phi,(pi-gamma)/pi,phi/pi)-sensitive


-----------------------
Edit Distance
 a, b strings
```

edit(a,b) = # operations to make a -> b
- delete
- insert

a = "mines"
b = "smiles"
edit(a,b) = 3
- insert 's' before 'm'
- delete 'n'
- insert 'l' after 'i'

many variations ("replace" operation)

non-negativity: # edits is non-negative
identity: only no edits if same
symmetry: can reverse operations
triangle: any intermediate -> equality
          any deviation -> more edits

Is this good for large text documents?
- slow to compute
- moving a sentence is a large edit, may change content little

- good for approximate string queries (google search, auto-correct)
  edit(a,b) > 3 is pretty large

Much work to approximate by L_1 distance (so can use LSH).
(eps,delta) keeps improving.

Graph Distance

Let G = (V,E) be a graph
V = vertices
E = edges E subset V x V

```
edges can be ordered or unordered
             (u,v)      {u,v}
```

edges can have weights w_{u,v} (=1 default)
(usually non-negative, infinite if non-existent)

<draw graph>

d(u,v) = min # edges between (u,v)
Path P = <u=r0,r1,r2,...,r{t-2},v=r{t-1}>

```
such that (u,r1) , (r{t-2},v), (ri,r{i+1}) in E
length(P) = sum_{ri,r{i+1}} w_{ri,r{i+1}}
d(u,v) = min_P<u...v> length(P)

Metric if w_(u,v) > 0, unordered
  non-negativity:  sum non-negative weights
  identity: only if no edges
  symmetry: can reverse edges
  triangle: any intermediate on path -> equality
            any deviation of path -> violates min-length-path


Much work to approximate graph by L_1 or L_2 distance so can use LSH
```