L6 -- LSH
[Jeff Phillips - Utah - Data Mining]

Consider a set of \( n \) (= 1 million items)
  Q1: Which items are similar?
  Q2: Given an query item, which others are similar?

For Q1: we don't want to check all \( O(n^2) \) distance (no matter how fast)
For Q2: we don't want to check against all \( O(n) \) items (only ones that might be close)

Consider \( n \) points in the plane. How do we quickly answer Q1 and Q2 efficiently.
- hierarchical models (range trees, kd-trees, B-trees) don't work in high dimensions
- lay down grid:
  + close points in same grid cell.
  + some across boundary
  + some further than 1 grid cell, but still "similar"
  + randomize grid, and check again

----

Abstract Ideas:
Hash (like a grid) so
  \( \Pr[h(a) = h(b)] > \alpha \) if \( d(a,b) < \gamma \)
  \( \Pr[h(a) = h(b)] < \beta \) if \( d(a,b) > \phi \)

Need \( \alpha > \beta \) for \( \gamma < \phi \)
Want (\( \alpha-\beta \)) large and (\( \phi-\gamma \)) small
Then: repeat *random* hash to "amplify"
  -> make (\( \alpha-\beta \)) smaller for fixed (\( \phi-\gamma \))
    (works for many \( \phi-\gamma \) simultaneously)

"(\( \gamma,\phi, \alpha,\beta \))-sensitive"

-------------

MinHashing as LSH:

\( t \) hash functions \{h1, h2, ... ht\}
  \( h_i = \lfloor m \rfloor \rightarrow \lfloor m \rfloor \) (at random)

Documents: D1 D2 D3 D4 D5 D6 ... Dn
  h1  1  2  0  4  0  1
  h2  2  0  1  3  1  2
  h3  5  3  3  0  3  1

---------------------------------------
  h4  1  2  3  0  2  1
ht

\[
\text{Jac}(D_1, D_2) = E\left[ \frac{1}{t} \left\| \text{hi}(D_1) = \text{hi}(D_2) \right\| \right]
\]

- \( b \) bands of \( r = t/b \) rows each
- Let \( s = \text{Jac}(D_1, D_2) \) = probability hashes collide
- \( s^r \) = prob all collide in 1 band
- \( (1-s^r) \) = prob not all collide in 1 band
- \( (1-s^r)^b \) = prob in no bands, all collide
- \( f = 1-(1-s^r)^b \) = prob all collide in at least 1 band

- \( f \) is an S-curve:
  - x-axis : \( s = \text{Jac}(D_1, D_2) \)
  - y-axis : probably being a candidate

- threshold \( \tau \) = where \( f \) has largest slope (about \( (1/b)^{1/r} \))

- \( r = 3, \ b = 5, \ t = 15 \)

<table>
<thead>
<tr>
<th>( s )</th>
<th>( 1 - (1-s^r)^b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>.1</td>
<td>.005</td>
</tr>
<tr>
<td>.2</td>
<td>.04</td>
</tr>
<tr>
<td>.3</td>
<td>.13</td>
</tr>
<tr>
<td>.4</td>
<td>.28</td>
</tr>
<tr>
<td>.5</td>
<td>.48</td>
</tr>
<tr>
<td>.6</td>
<td>.70</td>
</tr>
<tr>
<td>.7</td>
<td>.88</td>
</tr>
<tr>
<td>.8</td>
<td>.97</td>
</tr>
<tr>
<td>.9</td>
<td>.998</td>
</tr>
</tbody>
</table>

- As \( r \) and \( b \) increase, the S curve gets sharper.

- \( s > \tau \), we want to almost always check true distance
- \( s < \tau \), we rarely want to check true distance

- Any distance where there is a family of hash functions such that
  \( d(a,b) = \Pr[h(a)=h(b)] \)
  this techniques works directly.

- \( \tau = \gamma = \phi \)
- \( \alpha = \text{Jac}(a,b) \)
- \( \beta = 1-\text{Jac}(a,b) \)

- In general, if hash so
  \( \Pr[h(a) = h(b)] > \alpha \) if \( d(a,b) < \gamma \)
Pr[h(a) = h(b)] < beta if d(a,b) > phi
then same approach works as well...

-------------------------------------
LSH for Euclidean Distance

a,b in R^d for large d. How to LSH?

take random unit vector v in R^d
"project" all a,b onto v
a_v = <a,v> = sum_{i=1}^d a_i * v_i
* L_2(a_v, b_v) <= L_2(a,b) "contractive"

create bins of size gamma on v (in R^1)
* if L_2(a,b) < gamma/2
  Pr[a,b same bin] > 1/2
* if L_2(a,b) > 2gamma = phi
  Pr[a,b same bin] < 2/3
  (need cos(a-b,v) < pi/3 out of [0,pi])
  otherwise L_2(a,b) > 2 L_2(a_v,a_v) & -> different bins

"(gamma/2, 2gamma, 1/2, 1/3)-sensitive"

Can also take <a,v> mod (t gamma)
  for large enough t, and probably of collision is low

Essentially the best choice for *high* dimensional Euclidean data