Theory of LSH

Distance Measures
LS Families of Hash Functions
S-Curves
Distance Measures

◆ Generalized LSH is based on some kind of “distance” between points.
  ♦ Similar points are “close.”

◆ 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 : \( d(x,y) = \sqrt{\text{sum of the squares of the differences between} \ x \ \text{and} \ y \ \text{in each dimension}} \)

- 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

**L₂-norm:**
\[ \text{dist}(x,y) = \sqrt{(4^2 + 3^2)} = 5 \]

**L₁-norm:**
\[ \text{dist}(x,y) = 4 + 3 = 7 \]
Another Euclidean Distance

\( L_\infty \) 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 \( \infty \) of the \( L_n \) norm: 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 Jaccard similarity.
- **Cosine distance** $= \text{angle between vectors from the origin to the points in question.}$
- **Edit distance** $= \text{number of inserts and deletes to change one string into another.}$
- **Hamming Distance** $= \text{number of positions in which bit vectors differ.}$
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.

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

\(\text{Remember: } |a \cap b|/|a \cup b| = \text{probability that minhash}(a) = \text{minhash}(b).\)

\(\text{Thus, } 1 - |a \cap b|/|a \cup b| = \text{probability that minhash}(a) \neq \text{minhash}(b).\)
Triangle Inequality – (2)

◆ **Claim**: \(\text{prob}[\text{minhash}(x) \neq \text{minhash}(y)] \leq \text{prob}[\text{minhash}(x) \neq \text{minhash}(z)] + \text{prob}[\text{minhash}(z) \neq \text{minhash}(y)]\)

◆ **Proof**: whenever \(\text{minhash}(x) \neq \text{minhash}(y)\), at least one of \(\text{minhash}(x) \neq \text{minhash}(z)\) and \(\text{minhash}(z) \neq \text{minhash}(y)\) must be true.
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: \( p_1 \cdot p_2 / |p_2||p_1| \).

- Example: \( p_1 = 00111; p_2 = 10011 \).
- \( p_1 \cdot p_2 = 2; |p_1| = |p_2| = \sqrt{3} \).
- \( \cos(\theta) = 2/3; \theta \) is about 48 degrees.
Cosine-Measure Diagram

\[ d(p_1, p_2) = \theta = \arccos\left(\frac{p_1 \cdot p_2}{|p_2||p_1|}\right) \]
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: $|x| + |y| - 2|LCS(x,y)| = 5 + 6 - 2*4 = 3 = \text{edit distance.}$
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
Hamming Distance

- **Hamming distance** is the number of positions in which bit-vectors differ.
- **Example:** \( p_1 = 10101; p_2 = 10011. \)
- \( d(p_1, p_2) = 2 \) because the bit-vectors differ in the 3\(^{rd}\) and 4\(^{th}\) positions.
Why Hamming Distance Is a Distance Measure

\[ d(x, x) = 0 \text{ since no positions differ.} \]

\[ d(x, y) = d(y, x) \text{ by symmetry of “different from.”} \]

\[ d(x, y) \geq 0 \text{ since strings cannot differ in a negative number of positions.} \]

Triangle inequality: changing \( x \) to \( z \) and then to \( y \) is one way to change \( x \) to \( y \).
Families of Hash Functions

1. A “hash function” is any function that takes two elements and says whether or not they are “equal” (really, are candidates for similarity checking).
   ◆ **Shorthand**: $h(x) = h(y)$ means “$h$ says $x$ and $y$ are equal.”

2. A **family** of hash functions is any set of functions as in (1).
LS Families of Hash Functions

- Suppose we have a space $S$ of points with a distance measure $d$.
- A family $\mathcal{H}$ of hash functions is said to be $(d_1, d_2, p_1, p_2)$-sensitive if for any $x$ and $y$ in $S$:
  1. If $d(x,y) \leq d_1$, then prob. over all $h$ in $\mathcal{H}$, that $h(x) = h(y)$ is at least $p_1$.
  2. If $d(x,y) \geq d_2$, then prob. over all $h$ in $\mathcal{H}$, that $h(x) = h(y)$ is at most $p_2$. 


LS Families: **Illustration**

- High probability; at least $p_1$
- Low probability; at most $p_2$
Example: LS Family

Let $S =$ sets, $d =$ Jaccard distance, $H$ is formed from the minhash functions for all permutations.

Then Prob[$h(x) = h(y)$] = $1 - d(x, y)$.

Restates theorem about Jaccard similarity and minhashing in terms of Jaccard distance.
Example: LS Family – (2)

Claim: \( H \) is a \((1/3, 2/3, 2/3, 1/3)\)-sensitive family for \( S \) and \( d \).

If distance \( \leq 1/3 \) (so similarity \( \geq 2/3 \))

Then probability that minhash values agree is \( \geq 2/3 \)
Comments

1. For Jaccard similarity, minhashing gives us a \((d_1, d_2, (1-d_1), (1-d_2))\)-sensitive family for any \(d_1 < d_2\).

2. The theory leaves unknown what happens to pairs that are at distance between \(d_1\) and \(d_2\).

   - **Consequence**: no guarantees about fraction of false positives in that range.
Amplifying a LS-Family

- The “bands” technique we learned for signature matrices carries over to this more general setting.
- **Goal**: the “S-curve” effect seen there.
- **AND construction** like “rows in a band.”
- **OR construction** like “many bands.”
AND of Hash Functions

◆ Given family $H$, construct family $H'$ consisting of $r$ functions from $H$.

◆ For $h = [h_1, \ldots, h_r]$ in $H'$, $h(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for all $i$.

◆ Theorem: If $H$ is $(d_1, d_2, p_1, p_2)$-sensitive, then $H'$ is $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive.

◆ Proof: Use fact that $h_i$’s are independent.
OR of Hash Functions

◆ Given family \( H \), construct family \( H' \) consisting of \( b \) functions from \( H \).

◆ For \( h = [h_1, \ldots, h_b] \) in \( H' \), \( h(x) = h(y) \) if and only if \( h_i(x) = h_i(y) \) for some \( i \).

◆ **Theorem**: If \( H \) is \((d_1, d_2, p_1, p_2)\)-sensitive, then \( H' \) is \((d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)\)-sensitive.
Effect of AND and OR Constructions

- **AND** makes all probabilities shrink, but by choosing $r$ correctly, we can make the lower probability approach 0 while the higher does not.

- **OR** makes all probabilities grow, but by choosing $b$ correctly, we can make the upper probability approach 1 while the lower does not.
Composing Constructions

As for the signature matrix, we can use the AND construction followed by the OR construction.
- Or vice-versa.
- Or any sequence of AND’s and OR’s alternating.
AND-OR Composition

- Each of the two probabilities \( p \) is transformed into \( 1-(1-p^r)^b \).
  - The “S-curve” studied before.

- **Example**: Take \( H \) and construct \( H' \) by the AND construction with \( r = 4 \). Then, from \( H' \), construct \( H'' \) by the OR construction with \( b = 4 \).
Table for Function $1-(1-p^4)^4$

<table>
<thead>
<tr>
<th>p</th>
<th>$1-(1-p^4)^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2</td>
<td>.0064</td>
</tr>
<tr>
<td>.3</td>
<td>.0320</td>
</tr>
<tr>
<td>.4</td>
<td>.0985</td>
</tr>
<tr>
<td>.5</td>
<td>.2275</td>
</tr>
<tr>
<td>.6</td>
<td>.4260</td>
</tr>
<tr>
<td>.7</td>
<td>.6666</td>
</tr>
<tr>
<td>.8</td>
<td>.8785</td>
</tr>
<tr>
<td>.9</td>
<td>.9860</td>
</tr>
</tbody>
</table>

**Example:** Transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.8785,.0064)-sensitive family.
OR-AND Composition

◆ Each of the two probabilities \( p \) is transformed into \( (1-(1-p)^b)^r \).
  ✷ The same S-curve, mirrored horizontally and vertically.

◆ **Example**: Take \( H \) and construct \( H' \) by the OR construction with \( b = 4 \). Then, from \( H' \), construct \( H'' \) by the AND construction with \( r = 4 \).
### Table for Function $\left(1-(1-p)^4\right)^4$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$\left(1-(1-p)^4\right)^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.1</td>
<td>.0140</td>
</tr>
<tr>
<td>.2</td>
<td>.1215</td>
</tr>
<tr>
<td>.3</td>
<td>.3334</td>
</tr>
<tr>
<td>.4</td>
<td>.5740</td>
</tr>
<tr>
<td>.5</td>
<td>.7725</td>
</tr>
<tr>
<td>.6</td>
<td>.9015</td>
</tr>
<tr>
<td>.7</td>
<td>.9680</td>
</tr>
<tr>
<td>.8</td>
<td>.9936</td>
</tr>
</tbody>
</table>

**Example:** Transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.9936,.1215)-sensitive family.
Cascading Constructions

◆ **Example**: Apply the (4,4) OR-AND construction followed by the (4,4) AND-OR construction.

◆ Transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.9999996,.0008715)-sensitive family.

◆ Note this family uses 256 of the original hash functions.
For each S-curve $1 - (1 - p^r)^b$, there is a threshold $t$, for which $1 - (1 - t^r)^b = t$.

Above $t$, high probabilities are increased; below $t$, they are decreased.

You improve the sensitivity as long as the low probability is less than $t$, and the high probability is greater than $t$.

*Iterate as you like.*
Use of S-Curves – (2)

Thus, we can pick any two distances $x < y$, start with a $(x, y, (1-x), (1-y))$-sensitive family, and apply constructions to produce a $(x, y, p, q)$-sensitive family, where $p$ is almost 1 and $q$ is almost 0.

The closer to 0 and 1 we get, the more hash functions must be used.
For cosine distance, there is a technique analogous to minhashing for generating a \((d_1,d_2,(1-d_1/180),(1-d_2/180))\)-sensitive family for any \(d_1\) and \(d_2\).

Called *random hyperplanes*.
Random Hyperplanes

◆ Pick a random vector $\nu$, which determines a hash function $h_\nu$ with two buckets.

◆ $h_\nu(x) = +1$ if $\nu.x > 0$; $= -1$ if $\nu.x < 0$.

◆ LS-family $H = \text{set of all functions derived from any vector.}$

◆ Claim: $\text{Prob}[h(x) = h(y)] = 1 - (\text{angle between } x \text{ and } y \text{ divided by 180}).$
Proof of Claim

Look in the plane of $x$ and $y$.

Hyperplanes for which $h(x) = h(y)$

Hyperplanes (normal to $\nu$) for which $h(x) \neq h(y)$

$\text{Prob[Red case]} = \theta/180$
Signatures for Cosine Distance

- Pick some number of vectors, and hash your data for each vector.
- The result is a signature (sketch) of +1’s and –1’s that can be used for LSH like the minhash signatures for Jaccard distance.
- But you don’t have to think this way.
- The existence of the LS-family is sufficient for amplification by AND/OR.
Simplification

- We need not pick from among all possible vectors $\nu$ to form a component of a sketch.
- It suffices to consider only vectors $\nu$ consisting of +1 and −1 components.
LSH for Euclidean Distance

◆ Simple idea: hash functions correspond to lines.
◆ Partition the line into buckets of size $a$.
◆ Hash each point to the bucket containing its projection onto the line.
◆ Nearby points are always close; distant points are rarely in same bucket.
Projection of Points

If $d \gg a$, $\theta$ must be close to $90^\circ$ for there to be any chance points go to the same bucket.

If $d \ll a$, then the chance the points are in the same bucket is at least $1 - \frac{d}{a}$.
An LS-Family for Euclidean Distance

◆ If points are distance $\geq 2a$ apart, then $60 \leq \theta \leq 90$ for there to be a chance that the points go in the same bucket.
  - I.e., at most 1/3 probability.
◆ If points are distance $\leq a/2$, then there is at least $\frac{1}{2}$ chance they share a bucket.
◆ Yields a $(a/2, 2a, 1/2, 1/3)$-sensitive family of hash functions.
Fixup: Euclidean Distance

- For previous distance measures, we could start with an \((x, y, p, q)\)-sensitive family for any \(x < y\), and drive \(p\) and \(q\) to 1 and 0 by AND/OR constructions.
- Here, we seem to need \(y \geq 4x\).
Fixup – (2)

- But as long as $x < y$, the probability of points at distance $x$ falling in the same bucket is greater than the probability of points at distance $y$ doing so.

- Thus, the hash family formed by projecting onto lines is an $(x, y, p, q)$-sensitive family for some $p > q$.
  - Then, amplify by AND/OR constructions.