Finding Similar Items II

Krzysztof Dembczyński

Intelligent Decision Support Systems Laboratory (IDSS)
Poznań University of Technology, Poland

Software Development Technologies
Master studies, second semester
Academic year 2018/19 (winter course)

Review of the previous lectures

- Mining of massive datasets.
- Classification and regression.
- Evolution of database systems.
- MapReduce
- MapReduce in Apache Spark
- Nearest neighbor search:

Review of the previous lectures

- Mining of massive datasets.
- Classification and regression.
- Evolution of database systems.
- MapReduce
- MapReduce in Apache Spark
- Nearest neighbor search:
  - Minhash technique

Review of the previous lectures

- Mining of massive datasets.
- Classification and regression.
- Evolution of database systems.
- MapReduce
- MapReduce in Apache Spark
- Nearest neighbor search:
  - Minhash technique
  - Locality-sensitive hashing with minhash

Outline

Outline

Distance measure

- Suppose we have a set of points, called a space.

Distance measure

- Suppose we have a set of points, called a space.
- A distance measure on this space is a function $d(\boldsymbol{x}, \boldsymbol{y})$ that takes two points in the space as arguments and produces a real number, and satisfies the following axioms:

Distance measure

- Suppose we have a set of points, called a space.
- A distance measure on this space is a function $d(\boldsymbol{x}, \boldsymbol{y})$ that takes two points in the space as arguments and produces a real number, and satisfies the following axioms:
  1. $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ (no negative distances),

Distance measure

- Suppose we have a set of points, called a space.
- A distance measure on this space is a function $d(\boldsymbol{x}, \boldsymbol{y})$ that takes two points in the space as arguments and produces a real number, and satisfies the following axioms:
  1. $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ (no negative distances),
  2. $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ if and only if $\boldsymbol{x} = \boldsymbol{y}$ (distances are positive, except for the distance from a point to itself),

Distance measure

- Suppose we have a set of points, called a space.
- A distance measure on this space is a function $d(\boldsymbol{x}, \boldsymbol{y})$ that takes two points in the space as arguments and produces a real number, and satisfies the following axioms:
  1. $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ (no negative distances),
  2. $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ if and only if $\boldsymbol{x} = \boldsymbol{y}$ (distances are positive, except for the distance from a point to itself),
  3. $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ (distance is symmetric),

Distance measure

- Suppose we have a set of points, called a space.
- A distance measure on this space is a function $d(\boldsymbol{x}, \boldsymbol{y})$ that takes two points in the space as arguments and produces a real number, and satisfies the following axioms:
  1. $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ (no negative distances),
  2. $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ if and only if $\boldsymbol{x} = \boldsymbol{y}$ (distances are positive, except for the distance from a point to itself),
  3. $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ (distance is symmetric),
  4. $d(\boldsymbol{x}, \boldsymbol{y}) \leq d(\boldsymbol{x}, \boldsymbol{z}) + d(\boldsymbol{z}, \boldsymbol{y})$ (the triangle inequality).

Distance measure

- Suppose we have a set of points, called a space.
- A distance measure on this space is a function $d(\boldsymbol{x}, \boldsymbol{y})$ that takes two points in the space as arguments and produces a real number, and satisfies the following axioms:
  1. $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ (no negative distances),
  2. $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ if and only if $\boldsymbol{x} = \boldsymbol{y}$ (distances are positive, except for the distance from a point to itself),
  3. $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ (distance is symmetric),
  4. $d(\boldsymbol{x}, \boldsymbol{y}) \leq d(\boldsymbol{x}, \boldsymbol{z}) + d(\boldsymbol{z}, \boldsymbol{y})$ (the triangle inequality).
- The triangle-inequality axiom is what makes all distance measures behave as if distance describes the length of a shortest path from one point to another.

Euclidean distances

- The conventional distance measure in n-dimensional Euclidean space, which we shall refer to as the L_2-norm, is defined as:

$$d_2(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{j=1}^{n} (x_j - y_j)^2}$$

- In general, for any constant p, we can define the L_p-norm to be the distance measure d defined by:

$$d_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{j=1}^{n} |x_j - y_j|^p \right)^{\frac{1}{p}}$$

- Special cases are, besides the L_2-norm mentioned above,
  - Manhattan distance or L_1-norm:

$$d_1(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{j=1}^{n} |x_j - y_j| \right)$$

  - Chebyshev distance or L_∞-norm:

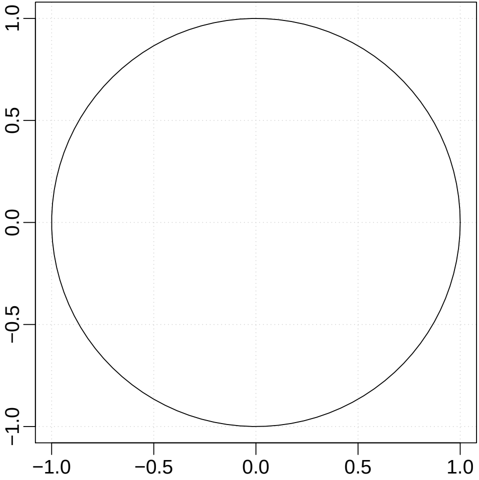
$$d_\infty(\boldsymbol{x}, \boldsymbol{y}) = \max_j(|x_j - y_j|)$$

Euclidean distances

- The spheres of L_p with different p: $p = 2$

Euclidean distances

- The spheres of L_p with different p: $p = 2$

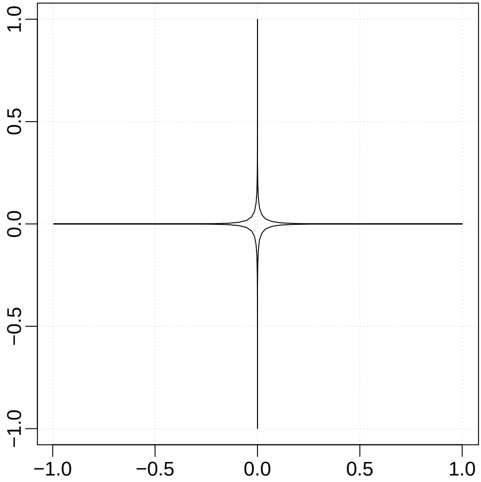


Euclidean distances

- The spheres of L_p with different p: $p = 0.2$

Euclidean distances

- The spheres of L_p with different p: $p = 0.2$

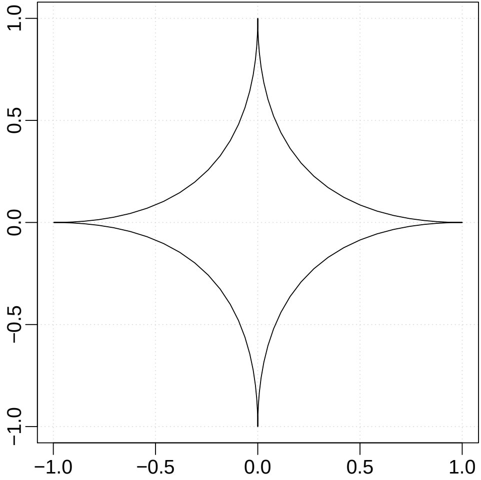


Euclidean distances

- The spheres of L_p with different p: $p = 0.5$

Euclidean distances

- The spheres of L_p with different p: $p = 0.5$

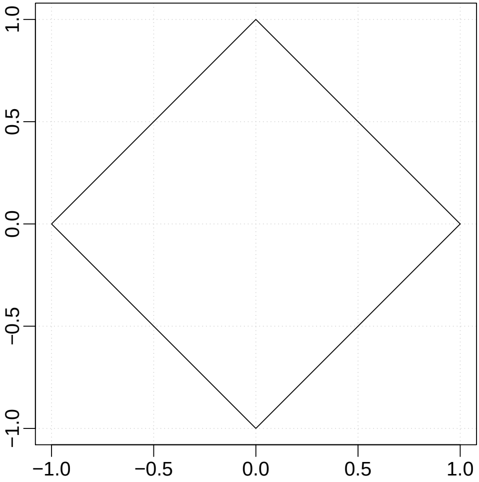


Euclidean distances

- The spheres of L_p with different p: $p = 1$

Euclidean distances

- The spheres of L_p with different p: $p = 1$

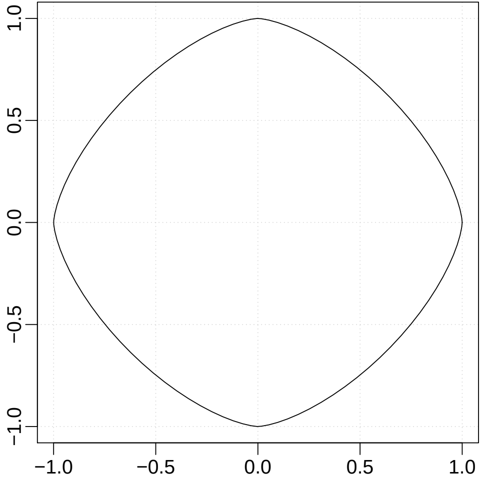


Euclidean distances

- The spheres of L_p with different p: $p = 1.5$

Euclidean distances

- The spheres of L_p with different p: $p = 1.5$

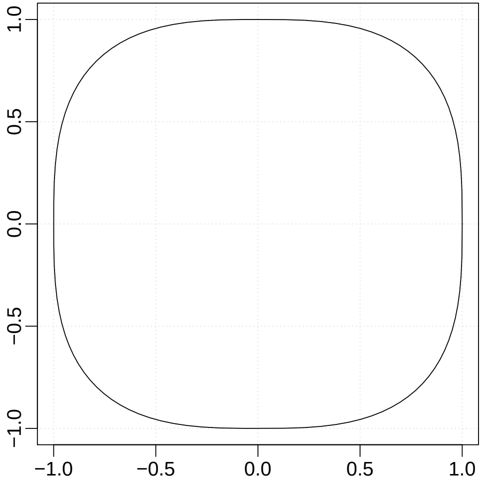


Euclidean distances

- The spheres of L_p with different p: $p = 3$

Euclidean distances

- The spheres of L_p with different p: $p = 3$

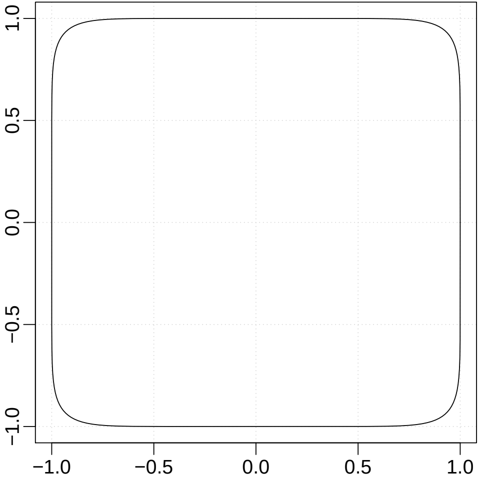


Euclidean distances

- The spheres of L_p with different p: $p = 10$

Euclidean distances

- The spheres of L_p with different p: $p = 10$



Jaccard distance

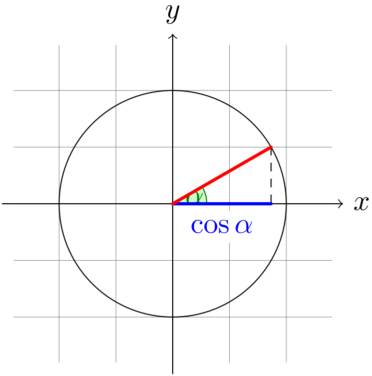
- Jaccard similarity is not a distance measure!
- We define the Jaccard distance of sets by:

$$d_{Jacc} = 1 - SIM(\boldsymbol{x}, \boldsymbol{y})$$

where $SIM(\boldsymbol{x}, \boldsymbol{y})$ is defined as before.

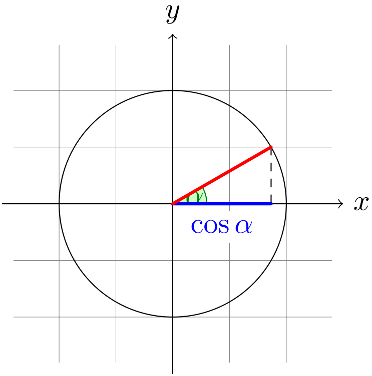
Cosine distance

- Let points be thought of as directions and do not distinguish between a vector and a multiple of that vector.



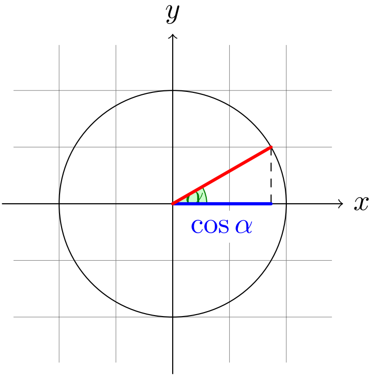
Cosine distance

- Let points be thought of as directions and do not distinguish between a vector and a multiple of that vector.
- The cosine distance between two points is the angle that the vectors to those points make.



Cosine distance

- Let points be thought of as directions and do not distinguish between a vector and a multiple of that vector.
- The cosine distance between two points is the angle that the vectors to those points make.
- This angle will be in the range 0 to 180 degrees, regardless of how many dimensions the space has.



Computing the cosine distance

- Given two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, the cosine of the angle between them is the dot product $\boldsymbol{x} \cdot \boldsymbol{y}$ divided by the L_2-norms of $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$cos(\theta) = \frac{\sum_{j=1}^{n} x_j y_j}{\sqrt{\sum_{j=1}^{n} x_j^2}\sqrt{\sum_{j=1}^{n} y_j^2}}$$

Computing the cosine distance

- Given two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, the cosine of the angle between them is the dot product $\boldsymbol{x} \cdot \boldsymbol{y}$ divided by the L_2-norms of $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$cos(\theta) = \frac{\sum_{j=1}^{n} x_j y_j}{\sqrt{\sum_{j=1}^{n} x_j^2}\sqrt{\sum_{j=1}^{n} y_j^2}}$$

- Apply the $\arccos$ function to translate $\cos(\theta)$ to an angle in the $[0, 180]$ degree range.

Hamming Distance

- The Hamming distance between two vectors is the number of components in which they differ:

$$d_H = \sum_{j=1}^{n} [\![x_j \neq y_j]\!]$$

Outline

Theory of locality-sensitive functions

- For a given distance measure we would like to find a family of functions that can be combined to distinguish strongly between pairs at a low distance from pairs at a high distance.

Theory of locality-sensitive functions

- For a given distance measure we would like to find a family of functions that can be combined to distinguish strongly between pairs at a low distance from pairs at a high distance.
- The minhash functions is one example of such family that uses the banding technique to achieve the above goal.

Theory of Locality-Sensitive Functions

- There are three conditions that we need for a family of functions:

- There are three conditions that we need for a family of functions:
  1. They must be more likely to make close pairs be candidate pairs than distant pairs.

Theory of Locality-Sensitive Functions

- There are three conditions that we need for a family of functions:
  1. They must be more likely to make close pairs be candidate pairs than distant pairs.
  2. They must be statistically independent to enable estimation of the probability that two or more functions will all give a certain response by the product rule for independent events.

Theory of Locality-Sensitive Functions

- There are three conditions that we need for a family of functions:
  1. They must be more likely to make close pairs be candidate pairs than distant pairs.
  2. They must be statistically independent to enable estimation of the probability that two or more functions will all give a certain response by the product rule for independent events.
  3. They must be efficient, in two ways:

Theory of Locality-Sensitive Functions

- There are three conditions that we need for a family of functions:
  1. They must be more likely to make close pairs be candidate pairs than distant pairs.
  2. They must be statistically independent to enable estimation of the probability that two or more functions will all give a certain response by the product rule for independent events.
  3. They must be efficient, in two ways:
     - They must be able to identify candidate pairs in time much less than the time it takes to look at all pairs.

Theory of Locality-Sensitive Functions

- There are three conditions that we need for a family of functions:
  1. They must be more likely to make close pairs be candidate pairs than distant pairs.
  2. They must be statistically independent to enable estimation of the probability that two or more functions will all give a certain response by the product rule for independent events.
  3. They must be efficient, in two ways:
     - They must be able to identify candidate pairs in time much less than the time it takes to look at all pairs.
     - They must be combinable to build functions that are better at avoiding false positives and negatives, and the combined functions must also take time that is much less than the number of pairs.

Locality-sensitive functions

- Consider functions $f(\boldsymbol{x}, \boldsymbol{y})$ that take two items and render a decision about whether these items should be a candidate pair.

Locality-sensitive functions

- Consider functions $f(\boldsymbol{x}, \boldsymbol{y})$ that take two items and render a decision about whether these items should be a candidate pair.
- It is convenient to use the notation:

Locality-sensitive functions

- Consider functions $f(\boldsymbol{x}, \boldsymbol{y})$ that take two items and render a decision about whether these items should be a candidate pair.
- It is convenient to use the notation:
  - $f(\boldsymbol{x}) = f(\boldsymbol{y})$ to mean $f(\boldsymbol{x}, \boldsymbol{y})$ is *yes*: *make $\boldsymbol{x}$ and $\boldsymbol{y}$ a candidate pair*,

Locality-sensitive functions

- Consider functions $f(\boldsymbol{x}, \boldsymbol{y})$ that take two items and render a decision about whether these items should be a candidate pair.
- It is convenient to use the notation:
  - $f(\boldsymbol{x}) = f(\boldsymbol{y})$ to mean $f(\boldsymbol{x}, \boldsymbol{y})$ is *yes*: *make $\boldsymbol{x}$ and $\boldsymbol{y}$ a candidate pair*,
  - $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$ to mean: *do not make $\boldsymbol{x}$ and $\boldsymbol{y}$ a candidate pair unless some other function concludes we should do so*.

Locality-sensitive functions

- Consider functions $f(\boldsymbol{x}, \boldsymbol{y})$ that take two items and render a decision about whether these items should be a candidate pair.
- It is convenient to use the notation:
  - $f(\boldsymbol{x}) = f(\boldsymbol{y})$ to mean $f(\boldsymbol{x}, \boldsymbol{y})$ is *yes*: *make $\boldsymbol{x}$ and $\boldsymbol{y}$ a candidate pair*,
  - $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$ to mean: *do not make $\boldsymbol{x}$ and $\boldsymbol{y}$ a candidate pair unless some other function concludes we should do so*.
- A collection of functions of this form will be called a family of functions.

Locality-sensitive functions

- Let $d_1 < d_2$ be two distances according to some distance measure d.

Locality-sensitive functions

- Let $d_1 < d_2$ be two distances according to some distance measure d.
- A family $\mathcal{F}$ of functions is said to be (d_1, d_2, p_1, p_2)-sensitive if for every $f \in \mathcal{F}$:

Locality-sensitive functions

- Let $d_1 < d_2$ be two distances according to some distance measure d.
- A family $\mathcal{F}$ of functions is said to be (d_1, d_2, p_1, p_2)-sensitive if for every $f \in \mathcal{F}$:
  1. If $d(\boldsymbol{x}, \boldsymbol{y}) \leq d_1$, then the probability that $f(\boldsymbol{x}) = f(\boldsymbol{y})$ is at least p_1.

Locality-sensitive functions

- Let $d_1 < d_2$ be two distances according to some distance measure d.
- A family $\mathcal{F}$ of functions is said to be (d_1, d_2, p_1, p_2)-sensitive if for every $f \in \mathcal{F}$:
  1. If $d(\boldsymbol{x}, \boldsymbol{y}) \leq d_1$, then the probability that $f(\boldsymbol{x}) = f(\boldsymbol{y})$ is at least p_1.
  2. If $d(\boldsymbol{x}, \boldsymbol{y}) \geq d_2$, then the probability that $f(\boldsymbol{x}) = f(\boldsymbol{y})$ is at most p_2.

Locality-sensitive functions

- Let $d_1 < d_2$ be two distances according to some distance measure d.
- A family $\mathcal{F}$ of functions is said to be (d_1, d_2, p_1, p_2)-sensitive if for every $f \in \mathcal{F}$:
  1. If $d(\boldsymbol{x}, \boldsymbol{y}) \leq d_1$, then the probability that $f(\boldsymbol{x}) = f(\boldsymbol{y})$ is at least p_1.
  2. If $d(\boldsymbol{x}, \boldsymbol{y}) \geq d_2$, then the probability that $f(\boldsymbol{x}) = f(\boldsymbol{y})$ is at most p_2.
- **Example**: For Jaccard distance we have:

Locality-sensitive functions

- Let $d_1 < d_2$ be two distances according to some distance measure d.
- A family $\mathcal{F}$ of functions is said to be (d_1, d_2, p_1, p_2)-sensitive if for every $f \in \mathcal{F}$:
  1. If $d(\boldsymbol{x}, \boldsymbol{y}) \leq d_1$, then the probability that $f(\boldsymbol{x}) = f(\boldsymbol{y})$ is at least p_1.
  2. If $d(\boldsymbol{x}, \boldsymbol{y}) \geq d_2$, then the probability that $f(\boldsymbol{x}) = f(\boldsymbol{y})$ is at most p_2.
- **Example**: For Jaccard distance we have:
  - We interpret a minhash function mh to make $\boldsymbol{x}$ and $\boldsymbol{y}$ a candidate pair if and only if $mh(\boldsymbol{x}) = mh(\boldsymbol{y})$.

Locality-sensitive functions

- Let $d_1 < d_2$ be two distances according to some distance measure d.
- A family $\mathcal{F}$ of functions is said to be (d_1, d_2, p_1, p_2)-sensitive if for every $f \in \mathcal{F}$:
  1. If $d(\boldsymbol{x}, \boldsymbol{y}) \leq d_1$, then the probability that $f(\boldsymbol{x}) = f(\boldsymbol{y})$ is at least p_1.
  2. If $d(\boldsymbol{x}, \boldsymbol{y}) \geq d_2$, then the probability that $f(\boldsymbol{x}) = f(\boldsymbol{y})$ is at most p_2.
- **Example**: For Jaccard distance we have:
  - We interpret a minhash function mh to make $\boldsymbol{x}$ and $\boldsymbol{y}$ a candidate pair if and only if $mh(\boldsymbol{x}) = mh(\boldsymbol{y})$.
  - Thus, the family of minhash functions is a $(d_1, d_2, 1 - d_1, 1 - d_2)$-sensitive family for any d_1 and d_2, where $0 \leq d_1 < d_2 \leq 1$.

Locality-sensitive functions

- Let $d_1 < d_2$ be two distances according to some distance measure d.
- A family $\mathcal{F}$ of functions is said to be (d_1, d_2, p_1, p_2)-sensitive if for every $f \in \mathcal{F}$:
  1. If $d(\boldsymbol{x}, \boldsymbol{y}) \leq d_1$, then the probability that $f(\boldsymbol{x}) = f(\boldsymbol{y})$ is at least p_1.
  2. If $d(\boldsymbol{x}, \boldsymbol{y}) \geq d_2$, then the probability that $f(\boldsymbol{x}) = f(\boldsymbol{y})$ is at most p_2.
- **Example**: For Jaccard distance we have:
  - We interpret a minhash function mh to make $\boldsymbol{x}$ and $\boldsymbol{y}$ a candidate pair if and only if $mh(\boldsymbol{x}) = mh(\boldsymbol{y})$.
  - Thus, the family of minhash functions is a $(d_1, d_2, 1 - d_1, 1 - d_2)$-sensitive family for any d_1 and d_2, where $0 \leq d_1 < d_2 \leq 1$.
  - For instance, for $d_1 = 0.3$ and $d_2 = 0.6$ we can assert that the family of minhash functions is a $(0.3, 0.6, 0.7, 0.4)$-sensitive family.

Amplifying a locality-sensitive family

- Suppose we are given a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$.

Amplifying a locality-sensitive family

- Suppose we are given a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$.
- We can construct a new family $\mathcal{F}'$ by the AND-construction on $\mathcal{F}$.

Amplifying a locality-sensitive family

- Suppose we are given a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$.
- We can construct a new family $\mathcal{F}'$ by the AND-construction on $\mathcal{F}$.
- Each member of $\mathcal{F}'$ consists of r members of $\mathcal{F}$ for some fixed r.

Amplifying a locality-sensitive family

- Suppose we are given a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$.
- We can construct a new family $\mathcal{F}'$ by the AND-construction on $\mathcal{F}$.
- Each member of $\mathcal{F}'$ consists of r members of $\mathcal{F}$ for some fixed r.
- If f is in $\mathcal{F}'$, and f is constructed from the set $\{f_1, f_2, \ldots, f_r\}$ of members of $\mathcal{F}$, we say $f(\boldsymbol{x}) = f(\boldsymbol{y})$ if and only if $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y})$ for all $i = 1, 2, \ldots, r$.

Amplifying a locality-sensitive family

- Suppose we are given a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$.
- We can construct a new family $\mathcal{F}'$ by the AND-construction on $\mathcal{F}$.
- Each member of $\mathcal{F}'$ consists of r members of $\mathcal{F}$ for some fixed r.
- If f is in $\mathcal{F}'$, and f is constructed from the set $\{f_1, f_2, \ldots, f_r\}$ of members of $\mathcal{F}$, we say $f(\boldsymbol{x}) = f(\boldsymbol{y})$ if and only if $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y})$ for all $i = 1, 2, \ldots, r$.
- Since the members of $\mathcal{F}$ are independently chosen to make a member of $\mathcal{F}'$, we can assert that $\mathcal{F}'$ is a $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive family.

Amplifying a locality-sensitive family

- Suppose we are given a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$.
- We can construct a new family $\mathcal{F}'$ by the AND-construction on $\mathcal{F}$.
- Each member of $\mathcal{F}'$ consists of r members of $\mathcal{F}$ for some fixed r.
- If f is in $\mathcal{F}'$, and f is constructed from the set $\{f_1, f_2, \ldots, f_r\}$ of members of $\mathcal{F}$, we say $f(\boldsymbol{x}) = f(\boldsymbol{y})$ if and only if $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y})$ for all $i = 1, 2, \ldots, r$.
- Since the members of $\mathcal{F}$ are independently chosen to make a member of $\mathcal{F}'$, we can assert that $\mathcal{F}'$ is a $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive family.
- **Example**: This construction corresponds to r rows in a single band for minhash functions.

Amplifying a locality-sensitive family

- There is another construction called the OR-construction.

Amplifying a locality-sensitive family

- There is another construction called the OR-construction.
- Each member f of $\mathcal{F}'$ is constructed from b members of $\mathcal{F}$, say $f_1, f_2, \ldots, f_b$.

Amplifying a locality-sensitive family

- There is another construction called the OR-construction.
- Each member f of $\mathcal{F}'$ is constructed from b members of $\mathcal{F}$, say $f_1, f_2, \ldots, f_b$.
- We define $f(\boldsymbol{x}) = f(\boldsymbol{y})$ if and only if $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y})$ for one or more values of i.

Amplifying a locality-sensitive family

- There is another construction called the OR-construction.
- Each member f of $\mathcal{F}'$ is constructed from b members of $\mathcal{F}$, say $f_1, f_2, \ldots, f_b$.
- We define $f(\boldsymbol{x}) = f(\boldsymbol{y})$ if and only if $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y})$ for one or more values of i.
- This construction turns a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$ into a $(d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-sensitive family $\mathcal{F}'$.

Amplifying a locality-sensitive family

- There is another construction called the OR-construction.
- Each member f of $\mathcal{F}'$ is constructed from b members of $\mathcal{F}$, say $f_1, f_2, \ldots, f_b$.
- We define $f(\boldsymbol{x}) = f(\boldsymbol{y})$ if and only if $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y})$ for one or more values of i.
- This construction turns a (d_1, d_2, p_1, p_2)-sensitive family $\mathcal{F}$ into a $(d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-sensitive family $\mathcal{F}'$.
- **Example**: This construction corresponds to b bands of 1 row for minhash functions.

Amplifying a locality-sensitive family

- The AND-construction lowers all probabilities, while the OR-construction makes all probabilities rise.

Amplifying a locality-sensitive family

- The AND-construction lowers all probabilities, while the OR-construction makes all probabilities rise.
- But if we choose $\mathcal{F}$ and r judiciously, we can make the small probability p_2 get very close to 0, while the higher probability p_1 stays significantly away from 0.

Amplifying a locality-sensitive family

- The AND-construction lowers all probabilities, while the OR-construction makes all probabilities rise.
- But if we choose $\mathcal{F}$ and r judiciously, we can make the small probability p_2 get very close to 0, while the higher probability p_1 stays significantly away from 0.
- Similarly, by choosing $\mathcal{F}$ and b judiciously, we can make the larger probability approach 1 while the smaller probability remains bounded away from 1.

Amplifying a locality-sensitive family

- The AND-construction lowers all probabilities, while the OR-construction makes all probabilities rise.
- But if we choose $\mathcal{F}$ and r judiciously, we can make the small probability p_2 get very close to 0, while the higher probability p_1 stays significantly away from 0.
- Similarly, by choosing $\mathcal{F}$ and b judiciously, we can make the larger probability approach 1 while the smaller probability remains bounded away from 1.
- We can, moreover, cascade AND- and OR-constructions in any order to make the low probability close to 0 and the high probability close to 1!!!

Amplifying a locality-sensitive family

- The AND-construction lowers all probabilities, while the OR-construction makes all probabilities rise.
- But if we choose $\mathcal{F}$ and r judiciously, we can make the small probability p_2 get very close to 0, while the higher probability p_1 stays significantly away from 0.
- Similarly, by choosing $\mathcal{F}$ and b judiciously, we can make the larger probability approach 1 while the smaller probability remains bounded away from 1.
- We can, moreover, cascade AND- and OR-constructions in any order to make the low probability close to 0 and the high probability close to 1!!!
- Obviously, the better the final family of functions is, the longer it takes to apply the functions from this family.

Amplifying a locality-sensitive family

- **Example**:

Amplifying a locality-sensitive family

- **Example**:
  - Suppose we start with a family $\mathcal{F}$.

Amplifying a locality-sensitive family

- **Example**:
  - Suppose we start with a family $\mathcal{F}$.
  - We use the AND-construction with $r = 4$ to produce a family $\mathcal{F}_1$.

- **Example**:
  - Suppose we start with a family $\mathcal{F}$.
  - We use the AND-construction with $r = 4$ to produce a family $\mathcal{F}_1$.
  - We then apply the OR-construction to $\mathcal{F}_1$ with $b = 4$ to produce a third family F_2.

Amplifying a locality-sensitive family

- **Example**:
  - Suppose we start with a family $\mathcal{F}$.
  - We use the AND-construction with $r = 4$ to produce a family $\mathcal{F}_1$.
  - We then apply the OR-construction to $\mathcal{F}_1$ with $b = 4$ to produce a third family F_2.
  - The members of F_2 each are built from 16 members of $\mathcal{F}$.

Amplifying a locality-sensitive family

- **Example**:
  - Suppose we start with a family $\mathcal{F}$.
  - We use the AND-construction with $r = 4$ to produce a family $\mathcal{F}_1$.
  - We then apply the OR-construction to $\mathcal{F}_1$ with $b = 4$ to produce a third family F_2.
  - The members of F_2 each are built from 16 members of $\mathcal{F}$.
  - The 4-way AND-function converts any probability p into p^4, and the 4-way OR-construction, converts this probability further into $1 - (1 - p^4)^4$.

Amplifying a locality-sensitive family

- **Example**:
  - Suppose $\mathcal{F}$ is the minhash functions being a $(0.2, 0.6, 0.8, 0.4)$-sensitive family.

p	$1-(1-p^4)^4$
0.2	0.0064
0.3	0.0320
0.4	0.0985
0.5	0.2275
0.6	0.4260
0.7	0.6666
0.8	0.8785
0.9	0.9860

Amplifying a locality-sensitive family

- **Example**:
  - Suppose $\mathcal{F}$ is the minhash functions being a $(0.2, 0.6, 0.8, 0.4)$-sensitive family.
  - Then $\mathcal{F}_2$, the family constructed by a 4-way AND followed by a 4-way OR, is a $(0.2, 0.6, 0.8785, 0.0985)$-sensitive family.

p	$1-(1-p^4)^4$
0.2	0.0064
0.3	0.0320
0.4	0.0985
0.5	0.2275
0.6	0.4260
0.7	0.6666
0.8	0.8785
0.9	0.9860

Amplifying a locality-sensitive family

- **Example**:
  - Suppose $\mathcal{F}$ is the minhash functions being a $(0.2, 0.6, 0.8, 0.4)$-sensitive family.
  - Then $\mathcal{F}_2$, the family constructed by a 4-way AND followed by a 4-way OR, is a $(0.2, 0.6, 0.8785, 0.0985)$-sensitive family.
  - This family corresponds to the banding technique with $b = 4$ bands and $r = 4$ rows of the banding technique.

p	$1 - (1 - p^4)^4$
0.2	0.0064
0.3	0.0320
0.4	0.0985
0.5	0.2275
0.6	0.4260
0.7	0.6666
0.8	0.8785
0.9	0.9860

Amplifying a locality-sensitive family

- **Example**:
  - Suppose $\mathcal{F}$ is the minhash functions being a $(0.2, 0.6, 0.8, 0.4)$-sensitive family.
  - Then $\mathcal{F}_2$, the family constructed by a 4-way AND followed by a 4-way OR, is a $(0.2, 0.6, 0.8785, 0.0985)$-sensitive family.
  - This family corresponds to the banding technique with $b = 4$ bands and $r = 4$ rows of the banding technique.
  - By replacing $\mathcal{F}$ by $\mathcal{F}_2$, we have reduced both the false-negative and false-positive rates, at the cost of making application of the functions take 16 times as long.

p	$1 - (1 - p^4)^4$
0.2	0.0064
0.3	0.0320
0.4	0.0985
0.5	0.2275
0.6	0.4260
0.7	0.6666
0.8	0.8785
0.9	0.9860

Amplifying a locality-sensitive family

- **Example**:
  - For the same cost, we can apply a 4-way OR-construction followed by a 4-way AND-construction.
  - Suppose as before that $\mathcal{F}$ is a $(0.2, 0.6, 0.8, 0.4)$-sensitive family.
  - Then the constructed family is a $(0.2, 0.6, 0.9936, 0.5740)$-sensitive.
  - This choice is not necessarily the best: the higher probability has moved much closer to 1, but the lower probability has also raised, increasing the number of false positives.

Amplifying a locality-sensitive family

- We can cascade constructions as much as we like.

Amplifying a locality-sensitive family

- We can cascade constructions as much as we like.
- We can combine the two families just discussed and obtain a family build from 256 hash functions.

Amplifying a locality-sensitive family

- We can cascade constructions as much as we like.
- We can combine the two families just discussed and obtain a family build from 256 hash functions.
- It would, for instance, transform a $(0.2, 0.8, 0.8, 0.2)$-sensitive family into a $(0.2, 0.8, 0.99999996, 0.0008715)$-sensitive family.

Outline

LSH families for Hamming distance

- Suppose we have a space of n-dimensional vectors, and $h(\boldsymbol{x}, \boldsymbol{y})$ denotes the Hamming distance between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

LSH families for Hamming distance

- Suppose we have a space of n-dimensional vectors, and $h(\boldsymbol{x}, \boldsymbol{y})$ denotes the Hamming distance between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.
- Take any position i of the vectors and define the function $f_i(\boldsymbol{x})$ to be the i-th element of vector $\boldsymbol{x}$.

LSH families for Hamming distance

- Suppose we have a space of n-dimensional vectors, and $h(\boldsymbol{x}, \boldsymbol{y})$ denotes the Hamming distance between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.
- Take any position i of the vectors and define the function $f_i(\boldsymbol{x})$ to be the i-th element of vector $\boldsymbol{x}$.
- Then $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y})$ if and only if vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ agree in the i-th position.
- The probability that $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y})$ for a randomly chosen i is:

LSH families for Hamming distance

- Suppose we have a space of n-dimensional vectors, and $h(\boldsymbol{x}, \boldsymbol{y})$ denotes the Hamming distance between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.
- Take any position i of the vectors and define the function $f_i(\boldsymbol{x})$ to be the i-th element of vector $\boldsymbol{x}$.
- Then $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y})$ if and only if vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ agree in the i-th position.
- The probability that $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y})$ for a randomly chosen i is:

  $$1 - \frac{h(x, y)}{n}\,,$$

  i.e., the fraction of positions in which $\boldsymbol{x}$ and $\boldsymbol{y}$ agree.

LSH families for Hamming distance

- Suppose we have a space of n-dimensional vectors, and $h(\boldsymbol{x}, \boldsymbol{y})$ denotes the Hamming distance between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.
- Take any position i of the vectors and define the function $f_i(\boldsymbol{x})$ to be the i-th element of vector $\boldsymbol{x}$.
- Then $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y})$ if and only if vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ agree in the i-th position.
- The probability that $f_i(\boldsymbol{x}) = f_i(\boldsymbol{y})$ for a randomly chosen i is:

$$1 - \frac{h(x, y)}{n},$$

  i.e., the fraction of positions in which $\boldsymbol{x}$ and $\boldsymbol{y}$ agree.
- The family $\mathcal{F}$ consisting of the functions $\{f_1, f_2, \ldots, f_n\}$ is a $(d_1, d_2, 1 - d_1/n, 1 - d_2/n)$-sensitive family of hash functions, for any $d_1 < d_2$.

Random hyperplanes and the cosine distance

- The cosine distance between two vectors is the angle between the vectors.
- Note that these vectors may be in a space of many dimensions, but they always define a plane, and the angle between them is measured in this plane.

Random hyperplanes and the cosine distance

- Let the angle between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ be θ.

Random hyperplanes and the cosine distance

- Let the angle between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ be θ.
- Suppose we pick a hyperplane through the origin of the space that intersects the plane of $\boldsymbol{x}$ and $\boldsymbol{y}$ in a line.

Random hyperplanes and the cosine distance

- Let the angle between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ be θ.
- Suppose we pick a hyperplane through the origin of the space that intersects the plane of $\boldsymbol{x}$ and $\boldsymbol{y}$ in a line.
- To pick a random hyperplane, we may pick the normal vector $\boldsymbol{v}$.

Random hyperplanes and the cosine distance

- Let the angle between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ be θ.
- Suppose we pick a hyperplane through the origin of the space that intersects the plane of $\boldsymbol{x}$ and $\boldsymbol{y}$ in a line.
- To pick a random hyperplane, we may pick the normal vector $\boldsymbol{v}$.
- The hyperplane is the set of points whose dot product with $\boldsymbol{v}$ is 0.

Random hyperplanes and the cosine distance

- Let the angle between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ be θ.
- Suppose we pick a hyperplane through the origin of the space that intersects the plane of $\boldsymbol{x}$ and $\boldsymbol{y}$ in a line.
- To pick a random hyperplane, we may pick the normal vector $\boldsymbol{v}$.
- The hyperplane is the set of points whose dot product with $\boldsymbol{v}$ is 0.
- Take the dot products of $\boldsymbol{v}$ with $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\boldsymbol{v} \cdot \boldsymbol{x} \quad \text{and} \quad \boldsymbol{v} \cdot \boldsymbol{y}$$

  and check the signs of these products.

Random hyperplanes and the cosine distance

- Let the angle between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ be θ.
- Suppose we pick a hyperplane through the origin of the space that intersects the plane of $\boldsymbol{x}$ and $\boldsymbol{y}$ in a line.
- To pick a random hyperplane, we may pick the normal vector $\boldsymbol{v}$.
- The hyperplane is the set of points whose dot product with $\boldsymbol{v}$ is 0.
- Take the dot products of $\boldsymbol{v}$ with $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\boldsymbol{v} \cdot \boldsymbol{x} \quad \text{and} \quad \boldsymbol{v} \cdot \boldsymbol{y}$$

  and check the signs of these products.
- What is the probability that the dot products of randomly chosen vector $\boldsymbol{v}$ with $\boldsymbol{x}$ and $\boldsymbol{y}$ will produce two different signs?

Random hyperplanes and the cosine distance

- Let the angle between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ be θ.
- Suppose we pick a hyperplane through the origin of the space that intersects the plane of $\boldsymbol{x}$ and $\boldsymbol{y}$ in a line.
- To pick a random hyperplane, we may pick the normal vector $\boldsymbol{v}$.
- The hyperplane is the set of points whose dot product with $\boldsymbol{v}$ is 0.
- Take the dot products of $\boldsymbol{v}$ with $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\boldsymbol{v} \cdot \boldsymbol{x} \quad \text{and} \quad \boldsymbol{v} \cdot \boldsymbol{y}$$

  and check the signs of these products.
- What is the probability that the dot products of randomly chosen vector $\boldsymbol{v}$ with $\boldsymbol{x}$ and $\boldsymbol{y}$ will produce two different signs?

$$\theta/180$$

Random hyperplanes and the cosine distance

- Thus, each hash function f in our locality-sensitive family $\mathcal{F}$ is built from a randomly chosen vector $\boldsymbol{v}_f$.

Random hyperplanes and the cosine distance

- Thus, each hash function f in our locality-sensitive family $\mathcal{F}$ is built from a randomly chosen vector $\boldsymbol{v}_f$.
- Given two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, we say $f(\boldsymbol{x}) = f(\boldsymbol{y})$ if and only if the dot products $\boldsymbol{v}_f \cdot \boldsymbol{x}$ and $\boldsymbol{v}_f \cdot \boldsymbol{y}$ have ...

Random hyperplanes and the cosine distance

- Thus, each hash function f in our locality-sensitive family $\mathcal{F}$ is built from a randomly chosen vector $\boldsymbol{v}_f$.
- Given two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, we say $f(\boldsymbol{x}) = f(\boldsymbol{y})$ if and only if the dot products $\boldsymbol{v}_f \cdot \boldsymbol{x}$ and $\boldsymbol{v}_f \cdot \boldsymbol{y}$ have ... the same sign.

Random hyperplanes and the cosine distance

- Thus, each hash function f in our locality-sensitive family $\mathcal{F}$ is built from a randomly chosen vector $\boldsymbol{v}_f$.
- Given two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, we say $f(\boldsymbol{x}) = f(\boldsymbol{y})$ if and only if the dot products $\boldsymbol{v}_f \cdot \boldsymbol{x}$ and $\boldsymbol{v}_f \cdot \boldsymbol{y}$ have ... the same sign.
- Then $\mathcal{F}$ is a $(d_1, d_2, (180 - d_1)/180, (180 - d_2)/180)$-sensitive family for the cosine distance.

- Consider first a 2-dimensional Euclidean space.

LSH families for Euclidean distance

- Consider first a 2-dimensional Euclidean space.
- Each hash function f in our family $\mathcal{F}$ will be associated with a randomly chosen line in this space.

LSH families for Euclidean distance

- Consider first a 2-dimensional Euclidean space.
- Each hash function f in our family $\mathcal{F}$ will be associated with a randomly chosen line in this space.
- Pick a constant a and divide the line into segments of length a.

LSH families for Euclidean distance

- Consider first a 2-dimensional Euclidean space.
- Each hash function f in our family $\mathcal{F}$ will be associated with a randomly chosen line in this space.
- Pick a constant a and divide the line into segments of length a.
- The segments of the line are the buckets into which function f hashes points: a point is hashed to the bucket in which its projection onto the line lies.

LSH families for Euclidean distance

- If the distance d between two points is small compared with a, then there is a good chance the two points hash to the same bucket.

LSH families for Euclidean distance

- If the distance d between two points is small compared with a, then there is a good chance the two points hash to the same bucket.
- For $d = a/2$ there is at least a 50% chance the two points will fall in the same bucket.

LSH families for Euclidean distance

- If the distance d between two points is small compared with a, then there is a good chance the two points hash to the same bucket.
- For $d = a/2$ there is at least a 50% chance the two points will fall in the same bucket.
- If the angle θ between the randomly chosen line and the line connecting the points is large, then there is an even greater chance that the two points will fall in the same bucket.

LSH families for Euclidean distance

- If the distance d between two points is small compared with a, then there is a good chance the two points hash to the same bucket.
- For $d = a/2$ there is at least a 50% chance the two points will fall in the same bucket.
- If the angle θ between the randomly chosen line and the line connecting the points is large, then there is an even greater chance that the two points will fall in the same bucket.
- For $\theta = 90$ degrees the two points are certain to fall in the same bucket.

LSH families for Euclidean distance

- Suppose d is larger than a.

LSH families for Euclidean distance

- Suppose d is larger than a.
- To have any chance of the two points falling in the same bucket, we need $d \cos \theta < a$.

LSH families for Euclidean distance

- Suppose d is larger than a.
- To have any chance of the two points falling in the same bucket, we need $d \cos \theta < a$.
- Note, however, that even if $d \cos \theta \ll a$, it is still not certain that the two points will fall in the same bucket.

LSH families for Euclidean distance

- Suppose d is larger than a.
- To have any chance of the two points falling in the same bucket, we need $d \cos \theta < a$.
- Note, however, that even if $d \cos \theta \ll a$, it is still not certain that the two points will fall in the same bucket.
- However, we can guarantee that If $d \geq 2a$, then there is no more than a $1/3$ chance the two points fall in the same bucket.

LSH families for Euclidean distance

- Suppose d is larger than a.
- To have any chance of the two points falling in the same bucket, we need $d \cos\theta < a$.
- Note, however, that even if $d \cos\theta \ll a$, it is still not certain that the two points will fall in the same bucket.
- However, we can guarantee that If $d \geq 2a$, then there is no more than a $1/3$ chance the two points fall in the same bucket.
- Why?

LSH families for Euclidean distance

- Suppose d is larger than a.
- To have any chance of the two points falling in the same bucket, we need $d\cos\theta < a$.
- Note, however, that even if $d\cos\theta \ll a$, it is still not certain that the two points will fall in the same bucket.
- However, we can guarantee that If $d \geq 2a$, then there is no more than a $1/3$ chance the two points fall in the same bucket.
- Why?
  - The reason is that for $\cos\theta < 1/2$ we have $\theta \in (60, 90]$ degrees, and for $\cos\theta \geq 1/2$, we have $\theta \in [0, 60]$.

LSH families for Euclidean distance

- Suppose d is larger than a.
- To have any chance of the two points falling in the same bucket, we need $d \cos\theta < a$.
- Note, however, that even if $d \cos\theta \ll a$, it is still not certain that the two points will fall in the same bucket.
- However, we can guarantee that If $d \geq 2a$, then there is no more than a $1/3$ chance the two points fall in the same bucket.
- Why?
  - The reason is that for $\cos\theta < 1/2$ we have $\theta \in (60, 90]$ degrees, and for $\cos\theta \geq 1/2$, we have $\theta \in [0, 60]$.
  - Since θ is the smaller angle between two randomly chosen lines in the plane, θ is twice as likely to be between 0 and 60 as it is to be between 60 and 90.

LSH families for Euclidean distance

- The family $\mathcal{F}$ of random line is a $(a/2, 2a, 1/2, 1/3)$-sensitive family of hash functions.

LSH families for Euclidean distance

- The family $\mathcal{F}$ of random line is a $(a/2, 2a, 1/2, 1/3)$-sensitive family of hash functions.
- For distances up to $a/2$ the probability is at least $1/2$ that two points at that distance will fall in the same bucket.

LSH families for Euclidean distance

- The family $\mathcal{F}$ of random line is a $(a/2, 2a, 1/2, 1/3)$-sensitive family of hash functions.
- For distances up to $a/2$ the probability is at least $1/2$ that two points at that distance will fall in the same bucket.
- For distances at least $2a$ the probability points at that distance will fall in the same bucket is at most $1/3$.

LSH families for Euclidean distance

- The family $\mathcal{F}$ of random line is a $(a/2, 2a, 1/2, 1/3)$-sensitive family of hash functions.
- For distances up to $a/2$ the probability is at least $1/2$ that two points at that distance will fall in the same bucket.
- For distances at least $2a$ the probability points at that distance will fall in the same bucket is at most $1/3$.
- We can amplify this family as we like, just as for the other examples of locality-sensitive hash functions.

LSH families for Euclidean distance

- There are, however, two problems with this family of hash functions:

LSH families for Euclidean distance

- There are, however, two problems with this family of hash functions:
  - The above reasoning was given only for 2-dimensional spaces.

LSH families for Euclidean distance

- There are, however, two problems with this family of hash functions:
  - The above reasoning was given only for 2-dimensional spaces.
  - This locality-sensitive family for any pair of distances d_1 and d_2 needs the stronger condition $d_1 < 4d_2$ than the families before, for which we have $d_1 < d_2$.

LSH families for Euclidean distance

- There are, however, two problems with this family of hash functions:
  - The above reasoning was given only for 2-dimensional spaces.
  - This locality-sensitive family for any pair of distances d_1 and d_2 needs the stronger condition $d_1 < 4d_2$ than the families before, for which we have $d_1 < d_2$.
- It turns out that there is a locality-sensitive family of hash functions for any $d_1 < d_2$ and for any number of dimensions constructed in a similar way.

LSH families for Euclidean distance

- There are, however, two problems with this family of hash functions:
  - The above reasoning was given only for 2-dimensional spaces.
  - This locality-sensitive family for any pair of distances d_1 and d_2 needs the stronger condition $d_1 < 4d_2$ than the families before, for which we have $d_1 < d_2$.
- It turns out that there is a locality-sensitive family of hash functions for any $d_1 < d_2$ and for any number of dimensions constructed in a similar way.
- Given that $d_1 < d_2$, we may not know what exactly the probabilities of p_1 and p_2 are, but we can be certain that $p_1 > p_2$.

LSH families for Euclidean distance

- There are, however, two problems with this family of hash functions:
  - The above reasoning was given only for 2-dimensional spaces.
  - This locality-sensitive family for any pair of distances d_1 and d_2 needs the stronger condition $d_1 < 4d_2$ than the families before, for which we have $d_1 < d_2$.
- It turns out that there is a locality-sensitive family of hash functions for any $d_1 < d_2$ and for any number of dimensions constructed in a similar way.
- Given that $d_1 < d_2$, we may not know what exactly the probabilities of p_1 and p_2 are, but we can be certain that $p_1 > p_2$.
- The reason is that this probability surely grows as the distance shrinks.

LSH families for Euclidean distance

- There are, however, two problems with this family of hash functions:
  - The above reasoning was given only for 2-dimensional spaces.
  - This locality-sensitive family for any pair of distances d_1 and d_2 needs the stronger condition $d_1 < 4d_2$ than the families before, for which we have $d_1 < d_2$.
- It turns out that there is a locality-sensitive family of hash functions for any $d_1 < d_2$ and for any number of dimensions constructed in a similar way.
- Given that $d_1 < d_2$, we may not know what exactly the probabilities of p_1 and p_2 are, but we can be certain that $p_1 > p_2$.
- The reason is that this probability surely grows as the distance shrinks.
- Thus, even if we cannot calculate p_1 and p_2 easily, we know that there is a (d_1, d_2, p_1, p_2)-sensitive family of hash functions for any $d_1 < d_2$ and any given number of dimensions.

Outline

Summary

- Locality-sensitive hashing.
- Distance measures.
- Theory of LSH.
- LSH for different distance measures.

Bibliography

- J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2014
  `http://www.mmds.org`
- P. Indyk. Algorithms for nearest neighbor search