CSEP 521: Applied Algorithms
Lecture 15 – Nearest neighbors

Announcements

Topics
• Bad cases for quad trees
• Hashing for nearest neighbors
• High dimensional data sets
• Documents data sets
• Jaccard Similarity
• MinHash
• Dimension Reduction

Bad case for Nearest Neighbor Query

Quad tree
• Search algorithm will lead us to expand every cell containing a point
• Approximate search gives a much better results

Voronoi diagram
• In theory, Voronoi diagram should work fine
• Vertices of degree greater than three are not expected
• Numerical issues can be very challenging
• Nightmare to debug
Hashing Based nearest neighbors

• Hashing to test if a query point \( y \) is with distance \( \delta \) of a point in \( S \)
• Center boxes on coordinates of the form \( c^2 \)
• Hash the boxes so that \( O(n) \) boxes are used
• Query point hashed to same boxes

More on distance metrics

• Implement across structures with multiple types
  • Record: (int Age, string Name, enum HairColor, int Weight)
• Weighing of coordinates and monotonic functions of coordinates generally preserve being a distance function
  • Can be tuning parameters for an application
• Example data set – Health Facility Lists, Entity resolution problem
  • Problem seems like it should be easy: merge lists of health facilities from different sources
  • Fields: name, admin region, health facility type, geographic coordinates

Document databases

• Large collections of text documents
• Applications such as similarity search, plagiarism detection, classification
• Distance metrics vs. similarity measures
  • Similarity measure a function where a high value is a more similar documents
• Representation – strings, token streams, bags of words

High dimensional searching

• Many data sets are high dimensional
  • High dimension can mean a mathematical space, such as \( \mathbb{R}^d \), or a structure, such as bag-of-words representation of documents
• Large scale data sets – Billions of photographs, web documents, sequences
• Tree based algorithms break down for high dimensions
  • Number of points in a ball of radius \( B \) increases exponentially with dimension
  • Processing dimensions is expensive
• Idea – dimension reduction techniques
  • Is it possible to reduced \( d \)-dimensional data to \( K \)-dimensional data, \( K << N \), that approximately preserves distances

Document representation

• Text strings
  • Edit distance as a similarity measure
• Token streams
  • Simplify words and remove punctuation or markup
• Bag of words
  • Represent the words as a set or a multiset
  • Sparse representation

More on distance metrics

• Implement across structures with multiple types
  • Record: (int Age, string Name, enum HairColor, int Weight)
• Weighing of coordinates and monotonic functions of coordinates generally preserve being a distance function
  • Can be tuning parameters for an application
• Example data set – Health Facility Lists, Entity resolution problem
  • Problem seems like it should be easy: merge lists of health facilities from different sources
  • Fields: name, admin region, health facility type, geographic coordinates

High dimensional searching

• Many data sets are high dimensional
  • High dimension can mean a mathematical space, such as \( \mathbb{R}^d \), or a structure, such as bag-of-words representation of documents
• Large scale data sets – Billions of photographs, web documents, sequences
• Tree based algorithms break down for high dimensions
  • Number of points in a ball of radius \( B \) increases exponentially with dimension
  • Processing dimensions is expensive
• Idea – dimension reduction techniques
  • Is it possible to reduced \( d \)-dimensional data to \( K \)-dimensional data, \( K << N \), that approximately preserves distances

Document databases

• Large collections of text documents
• Applications such as similarity search, plagiarism detection, classification
• Distance metrics vs. similarity measures
  • Similarity measure a function where a high value is a more similar documents
• Representation – strings, token streams, bags of words

High dimensional searching

• Many data sets are high dimensional
  • High dimension can mean a mathematical space, such as \( \mathbb{R}^d \), or a structure, such as bag-of-words representation of documents
• Large scale data sets – Billions of photographs, web documents, sequences
• Tree based algorithms break down for high dimensions
  • Number of points in a ball of radius \( B \) increases exponentially with dimension
  • Processing dimensions is expensive
• Idea – dimension reduction techniques
  • Is it possible to reduced \( d \)-dimensional data to \( K \)-dimensional data, \( K << N \), that approximately preserves distances

Document representation

• Text strings
  • Edit distance as a similarity measure
• Token streams
  • Simplify words and remove punctuation or markup
• Bag of words
  • Represent the words as a set or a multiset
  • Sparse representation
Jaccard Similarity

\[ \text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|} \]

Let X be the characteristic vector for A where \( x_j \) is the multiplicity of item j and Y be the characteristic vector for B where \( y_j \) is the multiplicity of item j.

\[ \text{Jaccard}(A, B) = \frac{\sum \min(x_j, y_j)}{\sum \max(x_j, y_j)} \]

Cosine Similarity

\[ \text{CS}(A, B) = \frac{\sum x_j y_j}{\|X\|_2 \|Y\|_2} \]

X is the characteristic vector for A where \( x_j \) is the multiplicity of item j and Y is the characteristic vector for B where \( y_j \) is the multiplicity of item j.

AltaVista search engine problem

- Avoid returning (near) duplicate items in search results
- Can fingerprinting techniques apply?
  - Fingerprinting is usually applied to detect or amplify small changes

Representation scheme

- Tokenize document
- Break document into shards
- Hash each shard into a domain of size \( 2^{64} \) (unsigned long)
- Treat as a bag of words
- Use Jaccard Similarity measure

Aside – Rabin Fingerprinting

- n-bit message \( m_0, \ldots, m_{n-1} \) viewed as polynomial over \( Z_2 \)
  - \( f(x) = m_0 + m_1 x + m_2 x^2 + \cdots + m_{n-1} x^{n-1} \)
- Pick a random irreducible polynomial \( p(x) \) of degree k (k = 64) and the fingerprint is \( f(x) \mod p(x) \)
- Suitable for domain of size \( 2^k \)
- Efficient implementation with bit operations including shifts
- Rolling hash that can reuse computation from shard
- Cool algebra for math majors

Similarity testing

- Identify document pairs that have high similarity by doing pairwise comparison
- Precompute hashes of shards – n shards for document of n tokens
- Cost of comparison is \( O(n) \)
- How to improve this: reduce the amount of information stored per document
MinHash

- \( U \) is the domain (in this case, the hash of the shards, \( \{0 \ldots 2^{64}\} \))
- Choose a random permutation \( \pi \) on \( U \)
- Let \( A \subseteq U \)
- \( \text{MinHash}(A) = \text{argmin}_{x \in A} \pi(x) \)
  - MinHash is the smallest element of \( A \) under the random permutation

An amazing result

\[
\Pr[\text{MinHash}(A) = \text{MinHash}(B)] = \frac{|A \cap B|}{|A \cup B|} = \text{Jaccard}(A, B)
\]

Using the MinHash

- Identify document pairs where \( \text{Jaccard}(A,B) \geq 0.95 \)
- Run MinHash with \( k \) independent permutations
- Number of times \( \text{MinHash}(A) = \text{MinHash}(B) \) is a good estimate of Jaccard Similarity
- Compute the \( k \) MinHashes for each document as a sketch
- Comparison of documents requires \( k \) comparisons