14 Indexes for Multimedia Data

14.1 Curse of Dimensionality

- Curse of Dimensionality
  - Why are traditional index structures useless in multidimensional spaces?
    - For (approx.) uniformly distributed data: all known index trees start failing at about 15-20 dimensions
    - Their use leads to higher costs than a linear scan
  - Is it possible to create an efficient high-dimensional tree structure?
  - What structure do high-dimensional spaces have anyway?

14.2 Dimension Reduction

- Relationship between high-dimensional cubes and spheres in the $\mathbb{R}^d$
  - (Hyper-) cube:
    - Edge length 1
  - (Hyper-) sphere:
    - Radius 1

- What is the center of the cube always inside the sphere?

14.3 GEMINI Indexing

- Localized
- Inside index trees always inside the sphere?

14.4 Indexes for Multimedia Data

- Calculating the Euclidean distance between the two centers:
  $$d(c, \alpha) = \sqrt{\left(\frac{1}{2}\right)^2 + \cdots + \left(\frac{1}{2}\right)^2} = \sqrt{d \cdot \frac{1}{4}} = \frac{\sqrt{d}}{2}$$
  - For $d = 4$ the center of the cube is on the edge of the sphere
  - For $d \geq 5$ the center of the cube is outside the sphere

- Where are points in high-dimensional space located? Inside or on the surface of a cube?
  - Consider cube $A = [0,1]^d$
  - Cut out smaller cube $B$ with edge length $(1-2\varepsilon)$
  - The volume of $A$ is 1, the volume of $B$ is $(1-2\varepsilon)^d$
14.1 Basic Geometry

- In high-dimensional spaces, almost all points lie near the surface of $A$.
  - What is the volume of the inner cube?

<table>
<thead>
<tr>
<th>$\varepsilon$ / $d$</th>
<th>2</th>
<th>50</th>
<th>100</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.64</td>
<td>1.41e-5</td>
<td>2.7e-10</td>
<td>3.51e-18</td>
<td>1.21e-27</td>
</tr>
<tr>
<td>0.05</td>
<td>0.81</td>
<td>0.01</td>
<td>2.71e-5</td>
<td>1.31e-10</td>
<td>1.81e-15</td>
</tr>
<tr>
<td>0.01</td>
<td>0.96</td>
<td>0.36</td>
<td>0.13</td>
<td>4.11e-5</td>
<td>1.71e-7</td>
</tr>
</tbody>
</table>

- If a point is positioned randomly (uniformly) in the outer cube, for large $d$ it is with very low probability in the inner cube.

14.1 Basic Geometry

- How big is the volume of spheres inscribed in cubes?
  - Again cube $A = [0,1]^d$ and an inscribed sphere $S$.
  - For even $d$, $S$ has volume $\left(\frac{\pi}{2}\right)^{d/2}$.

14.1 Basic Geometry

- How big is the volume of $S$?
- How many randomly distributed points in the cube are needed, to ensure on average that at least one lies within the sphere?
  - The number of points grows exponentially in $d$.

<table>
<thead>
<tr>
<th>$d$</th>
<th>Volume</th>
<th>Nr. Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.79</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0.31</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>0.002</td>
<td>400</td>
</tr>
<tr>
<td>20</td>
<td>2.46e-10</td>
<td>465372</td>
</tr>
<tr>
<td>40</td>
<td>3.28e-15</td>
<td>3.06e-17</td>
</tr>
<tr>
<td>100</td>
<td>1.87e-19</td>
<td>5.35e-23</td>
</tr>
</tbody>
</table>

14.1 Basic Geometry

- How many points are there at exactly distance $s$ from the center?
  - As the distance increases, the variance is lower.
  - For large $d$, almost all points have the same distance from the query (Beyer et al., 1998).

14.1 Conclusion

- High-dimensional spaces are “different”
  - In high-dimensional spaces, the sequential search through the objects is often better than using some index structure.
  - On the other hand, our analysis was focused on uniformly distributed points in Euclidean spaces.
  - Real-world data may have an “intrinsic” lower dimensionality.
  - Example: dimensions “price” and “maximal speed” in a vehicle database.
14.1 Speeding up Sequential Search

- **Vector Approximation Files** (Weber et al, 1998)
  - Partition each dimension into intervals
  - Dimension i is divided into $2^{b_i}$ intervals
  - Represented by $b_i$ bits
  - E.g., splitting some dimension in 4 intervals
  - Representation of these intervals by 00, 01, 10 and 11
  - The i-th coordinate of each data point can thus approximately be represented by $b_i$ bits
  - Thus, each point can be approximated by $b = b_1 + b_2 + ... + b_d$ bits

14.1 VA Files

- Points and their encoding in a VA-File

14.1 VA Files

- Advantages of VA Files
  - If $b$ is large enough, there are significantly more partitions of space into hyper-cubes as there are data points
  - Thus, collisions are nearly impossible, and every bit vector represents just one point
  - It is much faster to perform bit-wise operations on fixed-length bit vectors, than performing calculations on the original representation of the data

14.1 VA Files

- Query processing: filter & refine e.g., region queries:
  - Sequentially scan over all partitions
  - For partitions that intersect with the search region
  - Check all contained points using their exact coordinates

14.2 Dimensionality Reduction

- Indexing of high dimensional data is problematic, but does every dimension contain essential information?
  - Strongly correlated dimensions can be combined e.g., “price in Euro” and “price in Dollars”
  - If in some dimensions the objects differ more, then these dimensions also carry more information than others

14.2 Principal Component Analysis

- Principal Component Analysis (PCA), also known as Karhunen-Loève transform
  - Detection of linear dependencies between features, so-called axes
  - The most pronounced axis is called the main axis
  - The correlation is always subject to a certain variation
14.2 Principal Component Analysis

- **Linear dependence** is a sign of redundancy
  - A dimension may be represented as a linear combination of other dimensions
- **Idea**
  - Rotate (and shift) the axes such that there is no linear dependence between them
  - Remove all axes with low variance, to keep the error introduced by omission of information minimal

---

14.2 Principal Component Analysis

- **The covariance matrix** determines linear dependencies between different data dimensions
  - Let \( X = (X_1, \ldots, X_d) \) be a random vector which is uniformly distributed on the set of \((d\text{-dimensional})\) data points
  - **Center** the coordinate system around the mean:
    \[
    \tilde{X} := (\tilde{X}_1, \ldots, \tilde{X}_d), \text{ with } \tilde{X}_i := (X_i - E(X_i))
    \]

---

14.2 Principal Component Analysis

- **The covariance** between \( X_i \) and \( X_j \) is
  \[
  \text{cov}(X_i, X_j) := E \left( \tilde{X}_i \cdot \tilde{X}_j \right)
  \]
  - The covariance is **positive**, if \( X_i \) always “tends” to have large values whenever \( X_j \) does (and vice versa)
  - The covariance is **negative**, if \( X_i \) always “tends” to have large values whenever \( X_j \) has small values (and vice versa)

---

14.2 Principal Component Analysis

- **Linear Algebra**:
  - Any symmetric matrix \( A \) can be **diagonalized**
  - This means that there are matrices \( Q \) and \( D \) with \( A = Q \cdot D \cdot Q^{-1} \)
    - Where \( Q \) is orthogonal, therefore \( Q^T = Q^{-1} \)
    - \( D \) is diagonal, i.e., besides the main diagonal it contains only 0’s
  - The orthogonal matrices belonging to linear mappings are always just **reflections** and **rotations**

---
14.2 Principal Component Analysis

- The covariance matrix is symmetric
  - If the covariance matrix is diagonalized and the transformation corresponding to matrix $Q$ is applied to the data, then the covariance between the new data dimensions is always 0
- The properties of the transformation are transferred to the covariance!

14.2 Principal Component Analysis

- The diagonal matrix $D$ contains the eigenvalues of the covariance matrix
  - Eigenvalue decomposition
- Dimensions with low variance of the data (small corresponding eigenvalues) can be removed after the transformations
  - The resulting error is small and can be generally neglected

14.2 Example

- Data points: \((1, 1), (3, 3.5), (0, 0.5), (2, 3)\)
- Center $m = (\frac{3}{2})$
- Covariance matrix $Q = \begin{pmatrix} 2 & 1 \\ 1 & 1.625 \end{pmatrix}$

14.2 Example

- Transformation of the data:
  \[
  \begin{pmatrix} 1 \\ 3.5 \end{pmatrix}, \begin{pmatrix} 3 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 2 \\ 3 \end{pmatrix}
  \]

14.2 Latent Semantic Indexing

- A similar method to principal component analysis is Latent Semantic Indexing (LSI)
  - In principal component analysis, the covariance matrix is decomposed, while in LSI the feature matrix $F$ is decomposed
    - The feature matrix contains the feature vectors as columns
    - Note, this matrix is usually not quadratic and symmetric
  - In this case: $F = U \cdot D \cdot V^T$ with diagonal matrix $D$ and matrices $U$ and $V$ with orthonormal column vectors

- $x$-axis is the main axis (since it has the largest eigenvalue)
- Variance on the $y$ axis is small compared with variance on the $x$-axis (0.8 vs. 2.83)
- $y$-axis can therefore be omitted without much loss of information (dimensionality reduction)
14.2 Latent Semantic Indexing

- The decomposition of the feature matrix is a transformation on “minimal concepts” that are scaled by the eigenvalues in \( D \):
  - Each concept is an artificial dimension of a vector space, which expresses the “latent semantics” of the feature vectors.
  - LSI is used in information retrieval, to represent synonyms by a common concept and to split ambiguous terms into different concepts.

- Through the decomposition of the feature matrix, the semantics of...
  - …feature vectors (the columns of the feature matrix) is contained in matrix \( V^T \)
  - …feature dimensions (the rows of the feature matrix) is contained in matrix \( U \)
    - They can be used as in the principal component analysis for extracting dependencies.

- According to the eigenvalue decomposition, \( D \) indicates the importance of the new dimensions.
  - Irrelevant dimensions (particularly those with eigenvalue 0) can be deleted with minimal error.

- The decomposition of the feature matrix for the previous example yields:
  \[
  \begin{pmatrix}
  1 & 1 & 0.82 \\
  3 & 3.5 & 0.57 \\
  3 & 0.5 & -0.82 
  \end{pmatrix}
  \begin{pmatrix}
  1 & 0.17 \\
  3 & -0.66 \\
  5 & 0.35 \\
  0 & 0.73 
  \end{pmatrix}
  \begin{pmatrix}
  0.17 & 0.07 \\
  0.14 & 0.17 
  \end{pmatrix}
  \]
  Two latent dimensions with weights 7.95 and 1.8.
  - Weights of these dimensions in point 1: 0.17 and -0.14.
  - \( U \) and \( V \) transform the original axes, respectively the data points in the new space.

14.3 GEMINI Indexing

- **GEMINI** - GEneric Multimedia Object INdexIng (Faloutsos, 1996)
  - Can be used for hard to calculate distance functions between objects (e.g., edit distances).
  - Only useful, if there are no directly usable efficient search algorithms.
**14.3 GEMINI Indexing**

- **Idea:** use a distance function which is easy to compute as an estimate for the complex distance function and immediately prune some objects, i.e. a “quick and dirty test”
  - E.g., use the distance regarding average colors for the exclusion of certain histograms from the comparison
    - A histogram with avg. color red will be more similar to a histogram with avg. color orange, than to one with avg. color blue

- Assume we have to calculate a complex distance function \(d(A, B)\) on the set of objects \(O\)
  - Choose a function (transformation) \(f\) on \(O\) and a simple distance function \(\delta\) with the property
    \[\forall A, B \in O : \delta(f(A), f(B)) \leq d(A, B)\]
    (“lower bounding property”)

**14.3 Example**

- Comparison of **time series** for stock quotes
  - The comparison of two curves with the Euclidean distance \((d)\) on a discrete grid (e.g., closing prices) is very complicated
  - Transforming each curve with DFT and comparing the coefficients yields the same result, but is just as expensive (Parseval’s Theorem)

- But comparing only the **first few coefficients** corresponds to a dimension reduction \((f)\)
  - The Euclidean distance of only the first coefficient is faster to compute \((\delta)\)
  - Since the Euclidean distance is calculated as a sum of non-negative terms (square), the distance \(\delta\) of the first term always **underestimates** the distance \(d\) of all the terms (lower bounding)

**14.3 Algorithm**

- Preparation
  - Transform all the database objects with function \(f\)

- Pruning
  - To answer a region query \((Q, r)\) exclude all objects \(A\) with \(\delta(A, Q) > r\) from the search

- Query processing
  - Compute \(d'(B, Q)\) for all remaining objects \(B\) and exclude all objects with \(d'(B, Q) > r\)
  - Return all the remaining elements

- By using estimations **false positives** may occur, but never false negatives
14.3 Nearest Neighbors

- Nearest-neighbor queries can be performed using the index, too
  - **Attention:** the nearest neighbor according to \( \delta \) is not always the nearest neighbor according to \( d \)

14.3 Fast Map

- How can complicated distance functions be simplified (respecting the lower bounding)?
  - **Fast Map** represents all database objects as points in \( \mathbb{R}^k \), on the basis of some random distance function \( d \)
    (Faloutsos and Lin, 1995)
  - The Euclidean distance then allows an approximation of the original distance

14.3 Nearest Neighbors

- However,…
  - if \( A \) is the nearest neighbor of \( Q \) regarding \( \delta \), then the nearest neighbor of \( Q \) regarding \( d \) can have at most distance \( d(A, Q) \)
  - **Therefore:**
    - Find the nearest neighbors \( A \) of \( Q \) regarding \( \delta \)
    - Determine \( d(A, Q) \)
    - Create an area query with radius \( d(A, Q) \)
    - The nearest neighbor of \( Q \) regarding \( d \) is guaranteed to be among the results

14.3 Fast Map

- **Requirements**
  - Efficient mapping (if possible only linear effort)
  - Good distance approximation when regarding the original distances
  - Efficient mapping of new objects independent of all other objects

14.3 Fast Map

- **Idea:** consider all the objects as already in \( k \)-dimensional Euclidean space, however without knowing their coordinates
  - Find \( k \) orthogonal coordinate axes based on the distances between the points
  - Project the points on the axes and compute the feature values in Euclidean space
**14.3 Fast Map**

- **Projection** (of a point C) on an axis given by pivot points A and B

  ![Diagram](image1)

  Considering the law of cosines:
  
  \[ d(A, X) = \frac{[d(A, C)^2 + d(A, B)^2 - d(B, C)^2]}{2 \cdot d(A, B)} \]

- In this way, each point C can be projected onto the coordinate axis given by A and B
  - No coordinates of the involved points are required for this, but only the distances from each other
  - On this axis C has coordinate value \( d(A, X) \)

  ![Diagram](image2)

- After a new axis was added:
  - Project all data points on the \((k-1)\)-dimensional hyperplane, whose normal vector is the new axis
  - We again have a data set with a total of \(k\) dimensions including the newly created one
  - The coordinates of the remaining \((k-1)\) dimensions are unknown
    - We can however calculate the distances of the points in the hyperplane
      - Arbitrary points \(A\) and \(B\) in the hyperplane have distance \(d'(A, B)\) with
        \[ d'(A, B) = \sqrt{d(A, B)^2 - (x(A) - x(B))^2} \]
        where \(x(A)\) and \(x(B)\) are the coordinates of \(A\) and \(B\), on the newly added axis

  ![Diagram](image3)

- Repeat this steps \(k\) times
  - In this way, we create \(k\) new (orthogonal) dimensions, which describe the data
  - This allows us to approximate objects with complex distance function by points in the “standard” space
    - For this, only the distance matrix is needed
  - Afterwards, we can also apply “standard algorithms”

  ![Diagram](image4)

- **Problem**: which objects should we choose as pivot elements?
  - The two objects that are farthest apart provide the least loss of accuracy
  - They can be found with quadratic cost and with some heuristics, even within linear time

  ![Image](image5)

- **Calculate the pairwise dissimilarities in a set of words based on the edit distance**
  - To allow for indexing, all words should be mapped as multidimensional points in Euclidean space
  - The Euclidean distance can then be used instead of the more complicated editing distance

  ![Image](image6)
14.3 Example

• Edit distance between words
  - \( O = \{\text{Medium, Database, Multimedia, System, Object}\} \)
  - \( \Rightarrow \{w_1, \ldots, w_5\} \)

\[
\begin{array}{cccccc}
& w_1 & w_2 & w_3 & w_4 & w_5 \\
\hline
w_1 & 0 & 8 & 0 & 5 & 6 \\
w_2 & 8 & 0 & 10 & 4 & 9 \\
w_3 & 8 & 10 & 0 & 8 & 9 \\
w_4 & 5 & 8 & 5 & 0 & 4 \\
w_5 & 9 & 8 & 8 & 9 & 0 \\
\end{array}
\]

14.3 Fast Map

• Mapping on 4-dimensional points:

\[
\begin{array}{cccccc}
& v_1 & v_2 & v_3 & v_4 & \text{short} \\
\hline
v_1 & 8 & 10 & 0 & 3 & 5.55 & 3.8 \\
v_2 & 5.55 & 0 & 3.7 & 4.14 & 3.6 \\
v_3 & 6.28 & 0 & 2.65 & 0 & 3.1 \\
v_4 & 8.28 & 4.35 & 0 & 2.28 & 4.1 \\
\end{array}
\]

Next Semester

• Lectures
  - Relational Database Systems II
  - Information Retrieval and Web Search Engines
  - Distributed Database Systems and Peer-to-Peer Data Management

• Seminar
  - Information Extraction - How to Read the Web