Proseminar **Datenreduktionstechniken** SS 1999
Prof. Dr.- Ing. Gerhard Weikum
Dipl.- Inform. Anja Jantke
Dipl.- Inform. Christian König

Ausarbeitung zum Vortrag
Singulärwertdekomposition (SWD)

Stefan Posth

01.07.99

Inhaltsverzeichnis

1 Einleitung

Datenbanken werden mehr und mehr zur Speicherung von Multimedia- Inhalten, wie Bilder, Video usw. verwendet. Ein beliebter Ansatz zum Zugriff auf diese Dokumente ist die Ählichkeitssuche. Dabei sollen alle Dokumente aus der Datenbank zurückgegeben werden, die ähnlich sind zu einem Query-Dokument. Ähnlichkeit zwischen Dokumenten soll dabei durch low-level Features (Farbhistogramme, Textur, ...) repräsentiert werden.

1.1 Vektorraummodell

Dokumente werden als Vektoren in einem von Features aufgespannten Vektorraum interpretiert. In der sogenannten *Ähnlichkeitsmatrix* A werden diese Features und Dokumente folgendermaßen zusammengefaßt:

$$\mathit{Dokumente}\left\{\overbrace{\begin{pmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{pmatrix}}^{\mathit{Features}}\right.$$

Die Zeilen der Matrix A enthalten also m n-dimensionale Dokumentenvektoren und die Spalten geben die Features an. Der Eintrag w_{ij} der Matrix stellt die Gewichtung des Features j im i-ten Dokument dar.
Eine Ähnlichkeitssuche in der Datenbank ist nun äquivalent zu einer "nearest neighbor (NN)"-Suche in dem, oben angegebenen, von Feature-Vektoren aufgespannten, Vektorraum.

1.2 Beispiel

Als Beispiel sei eine Datenbank mit Farbbildern gegeben, und es sollen nach Vorgabe eines Query-Bildes alle Bilder, die ähnlich sind zu diesem Bild, zurückgeliefert werden. Für den Feature-Raum wird die Häufigkeit von verschiedenen Farben in unterschiedlichen Bildteilen zugrunde gelegt. Beispiele für Features sind "unten rechts schwarz", "unten rechts rot" und "unten rechts weiß". In der Matrix A wird für jedes Bild in der Datenbank eine Zeile angelegt und eine Gewichtung zwischen 0 und 1 für jedes Feature angegeben. Ist das Bild

unten rechts zum Beispiel schwarz würde bei dem Feature "unten rechts schwarz" eine 1 eingetragen, bei dem Feature "unten rechts rot" eine 0.4, usw.
Die Ähnlichkeit zweier Dokumente x und y ließe sich dann mit folgender Formel sinnvoll ausdrücken:

$$\text{Abstand}\,(x, y) = (x - y)^T A(x - y) = \sum_{i=1}^{m} \sum_{j=1}^{m} A_{ij}(x_i - y_i)(x_j - y_j)$$

Man kann über diese Abstandsfunktion jetzt mittels einer NN-Suche Dokumente in der Datenbank suchen, die ähnlich sind zu einem Query-Dokument.

1.3 NN-Suche

NN-Suche basierend auf dem sequentiellen Abarbeiten von großen Dateien, Tabellen oder den oben beschriebenen Feature-Vektoren ist sehr langsam. Dieses Problem läßt sich für niedrige Dimensionen mit Hilfe von mehrdimensionalen Indexstrukturen (R*-Tree, SS-Tree, ...) gut lösen. Allerdings sind bei hochdimensionalen Räumen auch Indexstrukturen zu langsam. Experimente zeigen, daß eine Dimensionserhöhung von 5 auf 10 zu Geschwindigkeitseinbußen um den Faktor 12 bei NN-Suchen führen kann. Gerade bei Multimediadaten fallen aber durch die Featurevektorräume hochdimensionale Dokumente an, in denen man effiziente NN-Suchen durchführen will.

2 Dimensionsreduktion

Die einzig bekannte Lösung zur deutlichen Verbesserung der Geschwindigkeit bei NN-Suchen stellt die Reduktion der Dimensionen der Dokumente dar. Die dadurch erzielte Geschwindigkeitssteigerung liegt, genauso wie der damit verbundene Informationsverlust, auf der Hand. Durch die Dimensionsreduktion ergibt sich also das Problem des Genauigkeitsverlustes bei NN-Suchen.

2.1 Maß für die Genauigkeit einer Suche

Aus dem Gebiet des Information Retrievals läßt sich folgendes Maß für die Genauigkeit einer Suche übernehmen:
Sei:

- A_d eine Menge von d-dimensionalen Punkten,
- A_k diese Menge von Punkten in einem k-dimensionalen Raum mit $k < d$,

- q ein Query-Objekt,
- $R(q, A_d)$ die Menge von Punkten, die nach einer Suche im d-dimensionalen Raum zurückgeliefert wird,
- $R(q, A_k)$ die Menge von Punkten, die nach einer Suche im k-dimensionalen Raum zurückgeliefert wird.

Dann lassen sich zwei Maße wie folgt definieren:

$$\text{Präzision}\,(q,d,k) = \frac{|R(q,A_k) \cap R(q,A_d)|}{|R(q,A_k)|}$$

$$\text{Ausbeute}\,(q,d,k) = \frac{|R(q,A_k) \cap R(q,A_d)|}{|R(q,A_d)|}$$

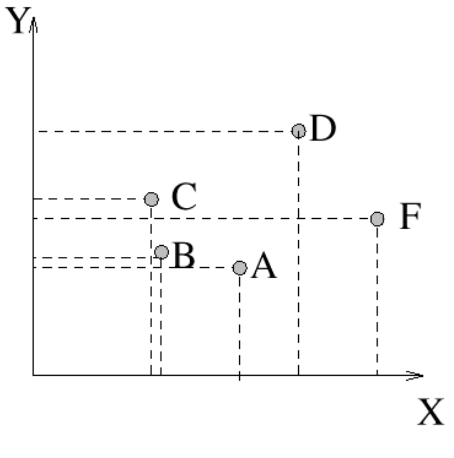
Dabei läßt sich Präzision auch als die Anzahl relevanter Resultatobjekte dividiert durch die Anzahl der Resultatobjekte und Ausbeute als die Anzahl relevanter Resultatobjekte durch Anzahl aller relevanten Objekte ausdrücken.
Für m NN-Suchen ergibt sich außerdem:

$$|R(q,A_d)| = |R(q,A_k)| = m$$

Die Anzahl der Resultatobjekte ist also identisch mit der Anzahl aller relevanten Resultatobjekte. Da in dieser Ausarbeitung nur NN-Suchen betrachtet werden, wird im Folgenden nur noch von Präzision als Maß für die Genauigkeit gesprochen.

2.2 Beispiel

Das folgende Beispiel zweidimensionaler Daten soll den Verlust von Genauigkeit bei Dimensionsreduktion und das Präzisionsmaß erläutern.



Die beiden nächsten Nachbarn von A in dem Koordinatensystem sind offensichtlich B und C.

Nach einer Dimensionsreduktion auf X ergeben sich für die beiden nächsten Nachbarn von A jetzt B und D, während eine Reduktion auf Y B und F als die beiden nächsten Nachbarn von A als Ergebnis liefert.

Für die Präzision ergibt sich bei einer Dimensionsreduktion auf X:

$$\text{Präzision}\,(A,2,1) = \frac{\left|\{B,D\}\cap\{B,C\}\right|}{\left|\{B,D\}\right|} = 0.5$$

Und für die Reduktion auf Y:

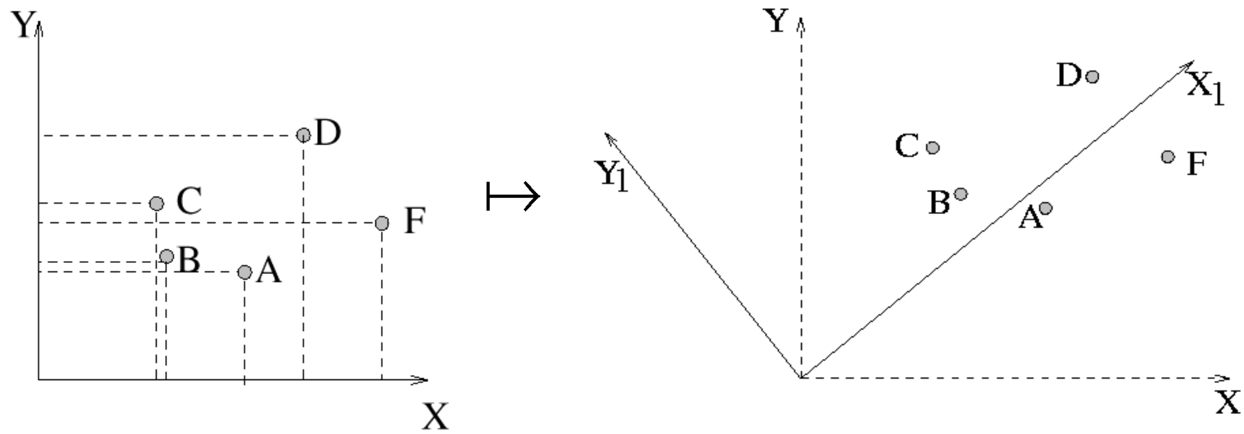
$$\text{Präzision}\,(A,2,1) = \frac{\left|\{B,F\}\cap\{B,C\}\right|}{\left|\{B,F\}\right|} = 0.5$$

In beiden Fällen ergibt sich also nur eine 50%-ige Präzision.

3 Singulärwertdekomposition

Die Idee der Singulärwertdekomposition ist es, die Daten, d.h. die Ähnlichkeitsmatrix A, so geschickt zu transformieren, daß eine anschließende Dimensionsreduktion zu geringeren Informationsverlusten und zu einer Verbesserung der damit verbundenen Präzision führt, als bei einer Dimensionsreduktion ohne Datentransformation.

Betrachtet werden soll hierzu noch einmal das Beispiel aus dem Abschnitt 2.2:



Eine Dimensionsreduktion auf X_1 in den transformierten Daten und eine NN-Suche nach den beiden nächsten Nachbarn von A ergibt jetzt B und C. Durch die Transformation der Daten konnte somit eine Präzision von 100% gehalten werden.

3.1 Definition Singulärwertdekomposition

Sei A die (m x n)-Ähnlichkeitsmatrix, die sich aus den m n-dimensionalen Dokumentenvektoren zusammensetzt und den Rang r besitzt. Dann läßt sich diese Matrix wie folgt zerlegen:

$$A = U\Sigma V^T = \left[u_1 u_2 \ldots u_r \middle| 0\right]_{m\times n} \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & 0 & \lambda_r & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}_{n\times n} \begin{bmatrix} v_1{}^T \\ v_2{}^T \\ \vdots \\ v_r{}^T \\ \overline{0} \end{bmatrix}_{n\times n}$$

Mit:

U ist eine (m x n)-Matrix, die sich aus den orthonormalen Eigenvektoren von AA^T zusammensetzt.

Σ ist eine (n x n)-Diagonalmatrix, auf deren Diagonale die Eigenvektoren von A^TA (Singulärwerte) stehen.

V ist eine (m x n)-Matrix, die sich aus den orthonormalen Eigenvektoren von A^TA zusammensetzt.

Diese Zerlegung nennt man Singulärwertdekomposition (SWD).
Es kann gezeigt werden, daß sich jede Matrix auf diese Art zerlegen läßt. Darüber hinaus kann auch gezeigt werden, daß es eine Zerlegung gibt, bei der gilt $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r$.
Der Rang r der Matrix A ist nicht von Anfang an bekannt und ergibt sich erst während der Singulärwertzerlegung. Das bedeutet, daß man hierdurch bereits eine Reduktion in den Daten erhält, da alle linear abhängigen Vektoren zusammengefaßt werden. Ein Informationsverlust entsteht dadurch auch nicht, denn die wegfallenden Daten sind redundant.
Unter diesem Gesichtspunkt lassen sich die Matrix U auch als Dokument-Thema-Ähnlichkeitsmatrix und V als Feature-Thema-Ähnlichkeitsmatrix bezeichnen, wobei Thema die lineare Abhängigkeit von Vektoren ausdrückt. (In Beispiel 1.2 wäre ein Thema der Featurevektoren "rechts unten".)
Eine zusätzliche Datenreduktion erhält man durch die "echte" Reduktion von Dimensionen, also dem Entfernen von Vektoren (auf 0 setzen) aus den Matrizen. Dabei ist darauf zu achten, daß je größer der Singulärwert ist, der entfernt wird, der Informationsverlust ebenfalls größer wird. Da $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r$ gilt, konzentrieren sich die Informationen in den niedrigen Dimensionen, während die höheren mit einem geringeren Informationsverlust entfernt werden können.

3.2 Anwendung

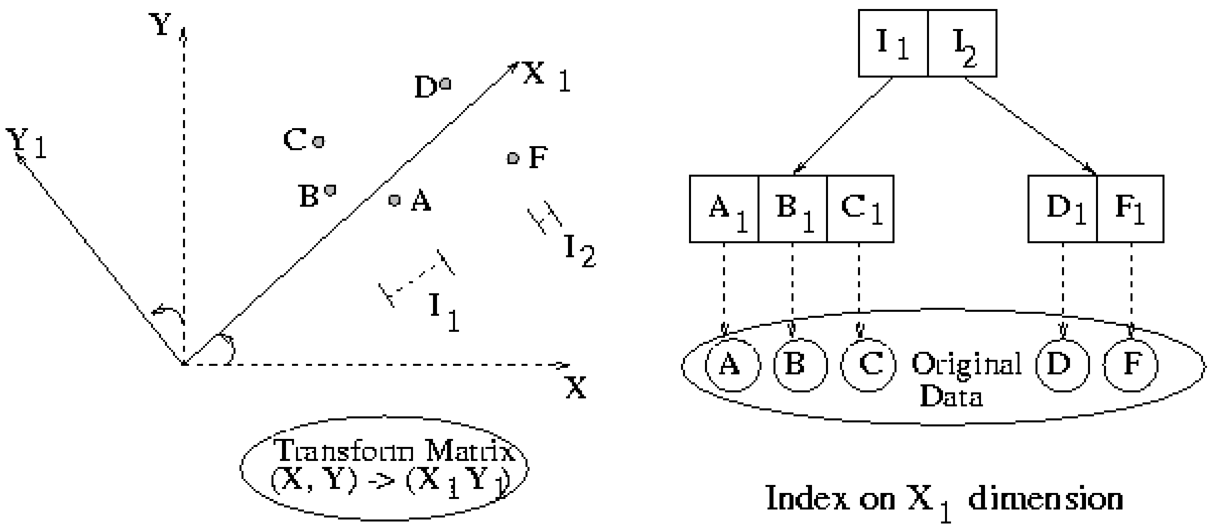
Zur Anwendung der Singulärwertdekomposition müssen folgende Schritte durchgeführt werden:

1. Zusammenfassung aller Dokumente und Features in der Matrix A
2. Berechnung der Transformationsmatrix V
3. Multiplikation von A mit V
4. die transformierten Daten entsprechen UΣ
5. Dimensionsreduktion auf den transformierten Daten, durch das Entfernen der hohen Dimensionen
6. Einfügen der transformierten Daten in Indexstrukturen

3.3 Einfügen der transformierten Daten in eine Indexstruktur

Nach der Reduktion der n-dimensionalen Daten auf k Dimensionen wird eine Indexstruktur für die reduzierten Daten aufgebaut werden. In dieser k-dimensionalen Indexstruktur werden in den Blattknoten Verweise auf die n-dimensionalen Originaldaten verwaltet. Bei einer Suchanfrage wird das Query-Objekt zuerst mittels SWD transformiert und dann durch die Indexknoten bearbeitet.

3.4 Beispiel



Zu dem bereits bekannten Beispiel der zweidimensionalen Daten ist rechts eine eindimensionale Indexstruktur angegeben. Die Indexstruktur wurde durch die Koordinaten entlang der X_1 Achse aufgebaut.

4 Singulärwertdekomposition für dynamische Datenbanken

Ein großes Problem der bisher vorgestellten Technik tritt bei dynamischen Datenbanken auf. Wenn Einfüge- und Löschoperationen vorkommen, kann es notwendig sein, die Achsen neu zu orientieren, d.h. die Singulärwertdekomposition erneut durchzuführen. Ansonsten könnte die Präzision von Suchanfragen stark abfallen. Die Neuberechnung soll erfolgen, nachdem eine beliebige Anzahl von Daten neu eingefügt bzw. gelöscht wurde.
Zwei Techniken zur Realisierung der Neuberechnung werden in den Abschnitten 4.1 und 4. 2 vorgestellt.

4.1 Vollständige SWD

Bei dieser Technik wird die SWD unter Verwendung von allen Daten neu berechnet, d.h. folgende zwei Schritte müssen durchgeführt werden:

1. Zugriff auf alle Datenobjekte über die Blätter des Baumes
2. Berechnung der Transformationsmatrix V mit diesen Daten

Der Vorteil dieser Technik liegt in dem optimalen Ergebnis, das unter Verwendung aller Daten erzielt wird. Allerdings ist dieses Verfahren dadurch auch sehr aufwendig.

4.2 Angenäherte SWD

Durch Anhäufung der Daten soll bei dieser Strategie der Berechnungsaufwand reduziert werden. Konkret müssen dazu drei Schritte durchgeführt werden:

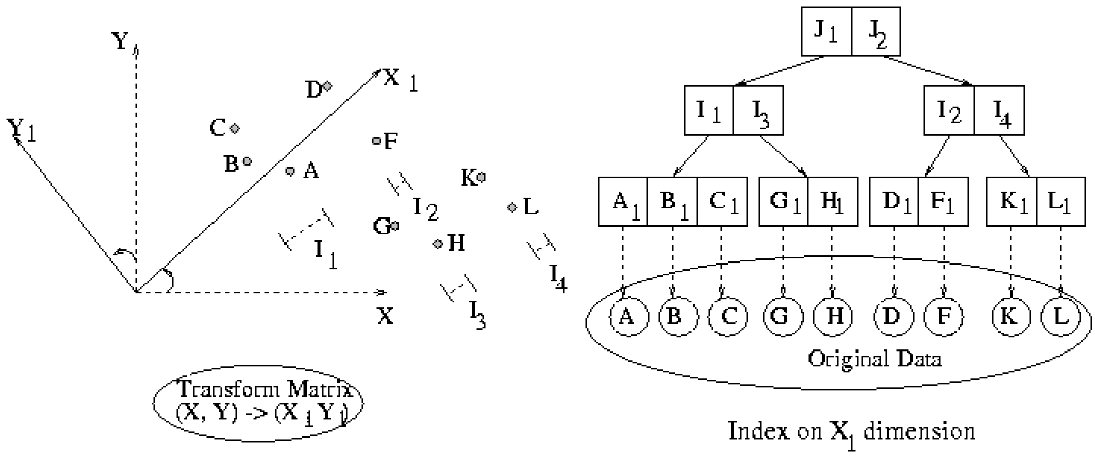
1. Unterteilung der Daten in Gruppen, wobei jeder Knoten eine Gruppe darstellt und zwar ausgehend von der vorletzten Ebene der Indexstruktur, also eine Ebene über den Blattknoten.
2. Berechnung des Centroids (Mittel der Punkte) für jede Gruppe
3. Berechnung der Transformationsmatrix V mit den Centroiden aller Gruppen

Die angehäuften Daten sind also die Centroiden aller Gruppen. Im Unterschied zur Verwendung der gesamten Daten entsteht somit ein Näherungsfehler, dessen Auswirkung auf die Präzision in Abschnitt 5.1 experimentell untersucht wird.

4.3 Beispiel

Als Beispiel werden noch einmal die zweidimensionalen Daten und die zugehörige eindimensionale Indexstruktur aus Abschnitt 3.4 verwendet. Zur Veranschaulichung wurden vier neue Werte G, H, K und L hinzugefügt.

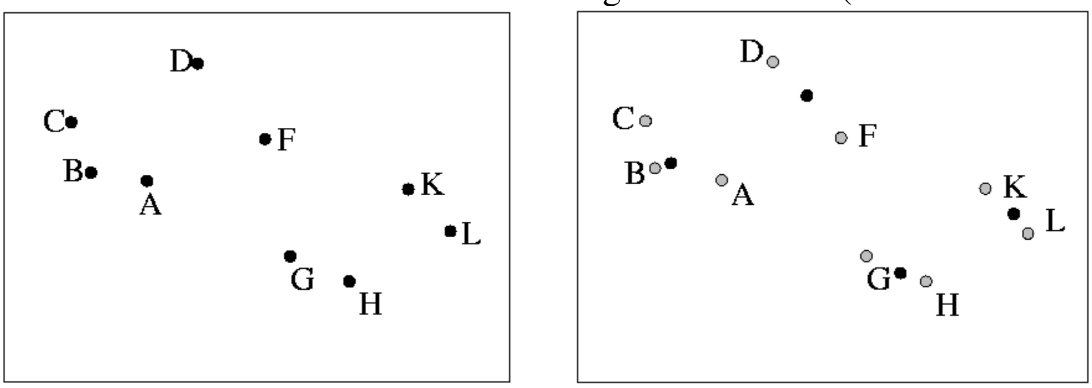
Die Indexstruktur auf der rechten Seite wurde ebenfalls um diese Werte erweitert.



In den folgenden beiden Abbildungen wird noch mal der Unterschied im Berechnungsaufwand zwischen Vollständiger und Angenäherter SWD aufgezeigt. Durch die Anhäufung der Punkte werden die 9 Datenobjekte auf 4 reduziert.

Alle Daten

Angehäufte Daten (schwarze Punkte)



4.4 Einfügen der neu berechneten Daten in Indexstrukturen

Nachdem die Daten neu berechnet worden sind, soll nun die Indexstruktur aktualisiert werden. Für die Blattknoten läßt sich dies recht einfach umsetzten. Die Transformationsmatrix muß nur auf die Daten angewendet werden und man kann diese nach der Reduktion wieder abspeichern.
Problematischer stellt sich die Aktualisierung der Indexknoten, da dies eine direkte Auswirkung auf die Geschwindigkeit bei einer Suche hat. Deswegen sollen dazu drei Strategien aufgezeigt werden:

- *Wiederaufbaustrategie*
- *Wiederverwendungsstrategie*
- *Wiederaufbaustrategie* in Verbindung mit der *Wiederaufbaustrategie*

Wiederaufbaustrategie

Der gesamte Baum wird bei dieser Strategie neu aufgebaut.

Vorteil: Die Zeit, in der eine Suchanfrage beantwortet wird, ist sehr gut.
Nachteil: Die Neukonstruktion ist sehr zeitaufwendig.

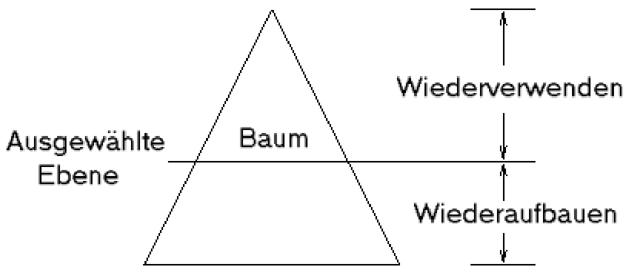
Wiederverwendungsstrategie

Bei dieser Technik werden keine strukturellen Änderungen am Baum vorgenommen. Die Baumstruktur wird wiederverwendet, indem man beginnend mit den Blattknoten die Ausdehnungen der Indexknoten bottom-up anpaßt.

Vorteil: Diese Technik ist sehr schnell durchzuführen, da die Struktur erhalten bleibt.
Nachteil: Die Achsen wurden neu orientiert, daher vergrößert sich wahrscheinlich die Ausdehnung der Indexknoten, was schlechtere Antwortzeiten zur Folge hat.

Verbindung der *Wiederaufbau-* und *Wiederverwendungsstrategie*

In diesem Verfahren sollen die Vorteile der *Wiederaufbau-* und *Wiederverwendungsstrategie* miteinander verbunden werden.
Die höheren Ebenen des Baumes sollen wiederverwendet werden, während ab einer beliebigen Ebene alle Unterbäume neu aufgebaut werden.



Verbindung der beiden Strategien *Wiederverwenden* und *Wiederaufbauen*

Je näher die gewählte Ebene an der Wurzel des Baumes liegt, desto höher sind die Speicheranforderungen, da ein größerer Teilbaum im Speicher verändert werden muß.

Der Wiederaufbau eines Unterbaums erfolgt mittels der VAMSpilt Technik:
Dir Gruppe Datenobjekte in dem Unterbaum wird rekursiv partitioniert, so daß jede Unterpartition in einen einzelnen Knoten paßt. Die Partitionierung erfolgt durch die Bestimmung der Dimensionen, die am meisten variieren und anschließende Aufspaltung der Daten in diesen Dimensionen.

Für n Datenobjekte, die in zwei Unterpartitionen aufgeteilt werden sollen, wird die Anzahl der Datenobjekte in der ersten Unterpartition wie folgt berechnet:

$$m = \begin{cases} \left\lfloor \frac{n}{2} \right\rfloor & wenn\, n \le 2b \\ b\left\lfloor \frac{n}{2b} + 0.5 \right\rfloor & sonst \end{cases}$$

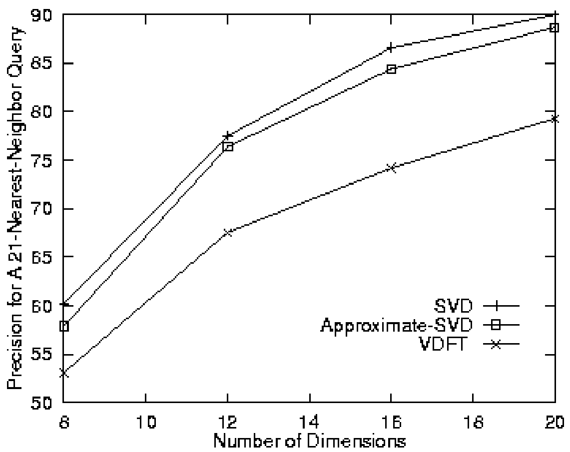
m ist die Anzahl der Datenobjekte in der ersten Unterpartition
b ist die Kapazität eines Knoten

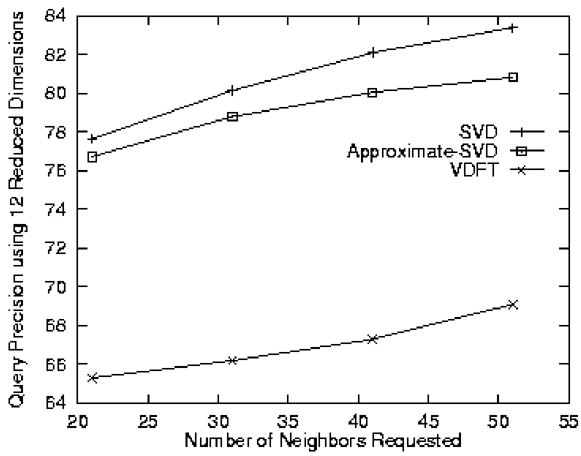
Eine zusätzliche Verbesserung der Suchgeschwindigkeit läßt sich erzielen, wenn man Datenobjekte entfernt, deren Weglassen zu einer erheblichen Reduzierung des Volumens des Baumes führt und sie anschließenden an der Wurzel des Baumes wiedereinfügt.

5 Experimente

In den nachfolgenden Experimenten soll zum einen ein Vergleich zwischen Vollständiger und Angenäherter SWD und zum anderen ein Vergleich der Einfügestrategien gegeben werden.
Als Grundlage für die Experimente dienen 26k 48-dimensionale Texturvektoren, von denen zu Beginn 3k in der Datenbank sind und 23k nachträglich eingefügt werden.

5.1 Vergleich zwischen Vollständiger- und Angenäherte - SWD



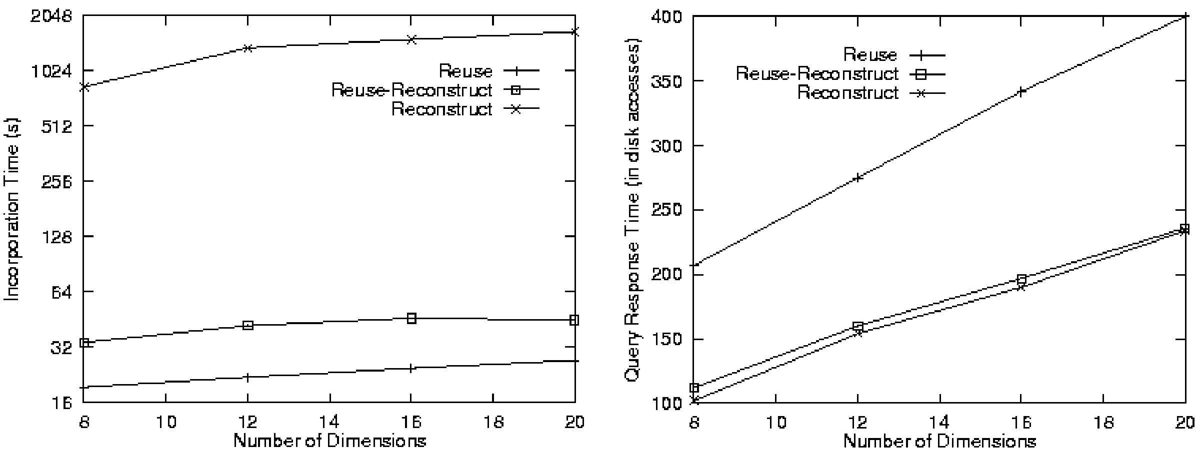


Bei VDFT handelt es sich um ein zur Singulärwertdekomposition alternatives Verfahren.

Man sieht, daß die Präzision von NN-Suchen bei Angenäherte SWD nur unwesentlich schlechter ist als bei Vollständiger SWD. Anhand der folgenden Tabelle sieht man die Vorteile der Angenäherten SWD, die in den kürzeren Berechnungszeiten liegen:

Dimensions	Approximate-SVD		SVD	
	Points	Time (s)	Points	Time (s)
8	205	1.6	26K	209
12	272	2.1	26K	209
16	333	2.65	26K	209
20	403	3.1	26K	209

5.2 Vergleich der Einfügestrategien



Die beiden Grafiken zeigen, daß die Verbindung der beiden Strategien *Wiederaufbauen (Reconstruct)* und *Wiederverwenden (Reuse)* einen guten Kompromiß zwischen Aufbauzeit und Zugriffszeit darstellt.
Einen genauerer Vergleich der Aufbauzeit, die für die einzelnen Strategien benötigt wird, findet man für 12 Dimensionen in dieser Tabelle:

Scheme	Incorporation time (s)	Query time (ms, disk I/O)
Reconstruct	1363	(27, 154)
Reuse	22	(52, 275)
Reuse-Reconstruct	42	(30, 160)

In der nächsten Tabelle wird abschließend noch Vollständige SWD+ *Wiederaufbauen* mit Angenäherter SWD + *Wiederverwenden-Wiederaufbauen* gegenübergestellt:

Transform Comp. + Incorporation	Query Precision	Query time (disk I/O)	Accumulated SVD-time (s)	Accumulated Incorporation time (s)
Approximate-SVD + Reuse-Reconstruct	61.6	1390	916	184
SVD + Reconstruct	66.2	1270	40318	21370

6 Singulärwertdekomposition und Clustering

Um eine weitere Verbesserung der Präzision bei einer Suchanfrage zu erzielen kann die Singulärwertzerlegung mit Clustering verbunden werden.

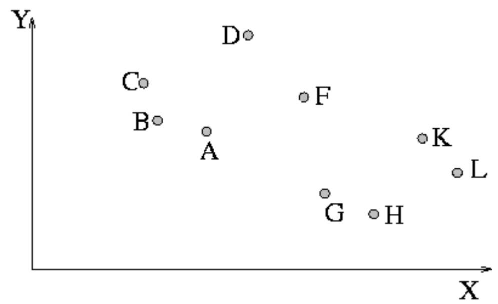
6.1 Algorithmus

Sei A die Datenmatrix.

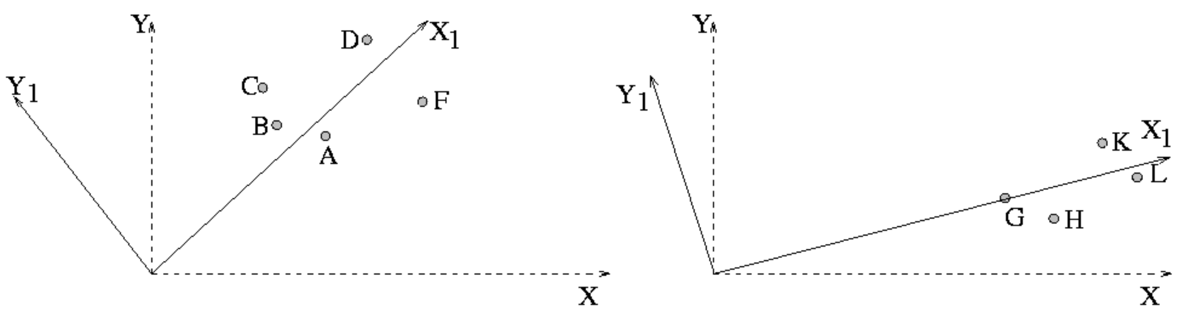
1. Zerlege A in K homogene Cluster X_k , $1 \leq k \leq K$. (LBG oder K-means)
   Die Anzahl der Cluster kann dynamisch als Parameter übergeben werden.
2. Führe SWD für alle X_k , $1 \leq k \leq K$ durch.
3. Baue eine Indexstruktur auf.

6.2 Beispiel

Seien die nebenstehenden Daten gegeben.



Eine mögliche Zerlegung der Daten in zwei Cluster mit anschließender Singulärwertdekomposition (repräsentiert durch die Achsenrotation) ist in den untenstehenden Abbildungen gegeben.



7 Zusammenfassung

Mit der Singulärwertdekomposition besitzt man eine Technik, die es einem ermöglicht, deutliche Geschwindigkeitsverbesserungen bei NN-Suchen zu erzielen. Der Nachteil ist natürlich, daß man einen Informationsverlust einkalkulieren muß.
Die Anwendungsgebiete von Singulärwertzerlegung reichen von Datenreduktion bzw. Datenkompression über das Lösen von linearen Gleichungssystemen bis hin zur Rauschunterdrückung bei digitaler Signalverarbeitung (DSP). Darüber hinaus wird sie auch in WWW Suchmaschinen eingesetzt.

Literaturverzeichnis

K.V.R. Kanth, D.Agrawal, A.Singh: "Dimensionality Reduction for Similarity Searching in Dynamic Databases", *SIGMOD Conference 1998, ACM Press 1998.*

A. Thomasian, V. Castelli, C.-S. Li: "Clustering and Singular Value Decomposition for Approximate Indexing in High Dimensional Spaces", *CIKM, 201-207, 1998.*

G. Weikum, Vorlesungsmanuskript Datenbanksysteme, Kapitel 16: Dokumenten- und Multimedia-Retrieval , *Universität des Saarlandes, 1998.*

D. Chan, Master Thesis: "http://xfactor.wpi.edu/Works/DChan/MSThesis/Thesis.html", *1998*