Ludwig Maximilians Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

Skript zur Vorlesung

Knowledge Discovery in Databases II

im Sommersemester 2008

Kapitel 4: Parallel, Distributed und Privacy Preserving Data Mining

http://www.dbs.ifi.lmu.de/Lehre/KDD

4.1 Einleitung

Bisher: Alle Daten sind in einer Datenbank gespeichert und der Data Mining Algorithmus darf auf alle beliebig häufig Daten zugreifen.

Aber:

- bei sehr großen Datenmengen ist Verarbeitung zeitaufwendig
- Bei verteilten Datenquellen:
  Integration der Daten oft mit hohen Transferkosten verbunden
- Bestimmte Datenobjekte/Ergebnisse können Personen diskreditieren
  - Globaler Data Mining Algorithmus darf nicht einzelne Objekte zugreifen
  - Ergebnisse dürfen dann nicht erlaubt werden, wenn sie Rückschlüsse auf Objektwerte erlauben

4.1 Einleitung

Ziele:

- mehrere Rechner für Verbesserung der Effizienz nutzen

  $\Rightarrow$ Parallelisierung von Data Mining Algorithmen
- Bei verteilter Datenhaltung:
  - Minimale Transferkosten
  - Effiziente Verarbeitung
  - Abweichung vom Ergebnis eines zentralen Data Mining Algorithmus soll minimal sein.
- bei vertraulichen Daten:
  - Informationen über einzelne Datenobjekte dürfen nicht weiter gegeben werden
  - auch Diskreditierung durch abgeleitete Informationen muss vermieden werden

Anwendungsbeispiele

- Parallelisierung von DBSCAN zur Beschleuningung großer Datenmengen
- Endkundengruppierung bei Großhändlern:
  - Einzelhändler geben einzelne Daten nicht raus.
  - Großhändler brauchen verteilte und „*privacy-preserving*“ Verfahren, um dennoch Kundengruppen zu bestimmen.
- Bestimmung von nicht-kompromitierenden Assoziationsregeln auf Webanfragen

  Bsp:

  Fridoline M. und Informatik Uni München

  => *Schwangerschaftsabbruch*

  Kompromitierung über sogenannte „Vanity-Searches“.

  => Unterdrücken zu spezieller und kompromitierender Regeln

Übersicht

4.1 Einführung

paralleles, verteiltes Mining, Privacy Preservation

4.2 Parallele und Verteilte Algorithmen

Unterschied: parallele und verteilte Algorithmen, Aufteilungsunabhängigkeit, verteiltes Clustering.

4.3 Privacy Preservation

Peturbation, Aggregation, Swapping,

4.2 Parallels und Verteiltes Data Mining

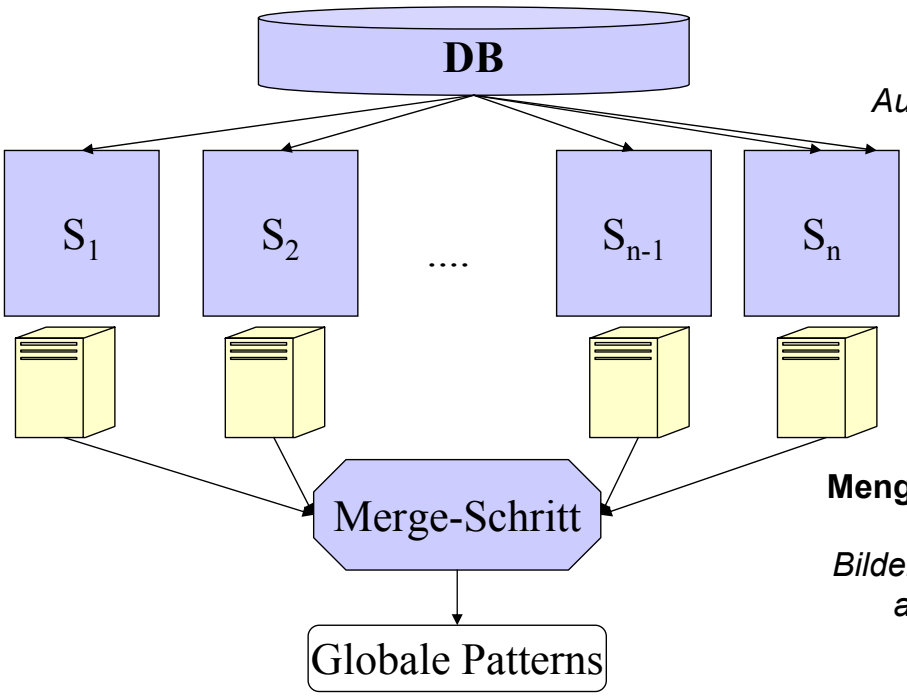
Paralleles Data Mining:

- Datenbank liegt bereits integriert vor, soll aber auf *k* Rechnern parallel analysiert werden.
- Performanzgewinn durch „*Divide and Conquer*“ Strategie:
  - ⇒ Teile Datenmenge auf
  - ⇒ Analysiere die Teilmengen auf getrennten Rechnern nebenläufig.
  - ⇒ Fasse lokale Ergebnisse zusammen zu globalen Mustern zusammen.

Entscheidend hierbei:

- Wie teile ich eine Datenmenge so auf, dass lokale Muster mit möglichst wenig Aufwand in ein integriertes globales Model integriert werden können?
- Wie kann ich die teure Kommunikation zwischen den Einzelrechnern so niedrig wie möglich halten.

Ablauf paralleler Algorithmen



integriete Daten *DB*

Aufteilung von DB in S_i

aufgeteilte Daten

Analyse der S_i

Menge aller lokalen Muster

Bilden von globalen Mustern aus lokalen Mustern

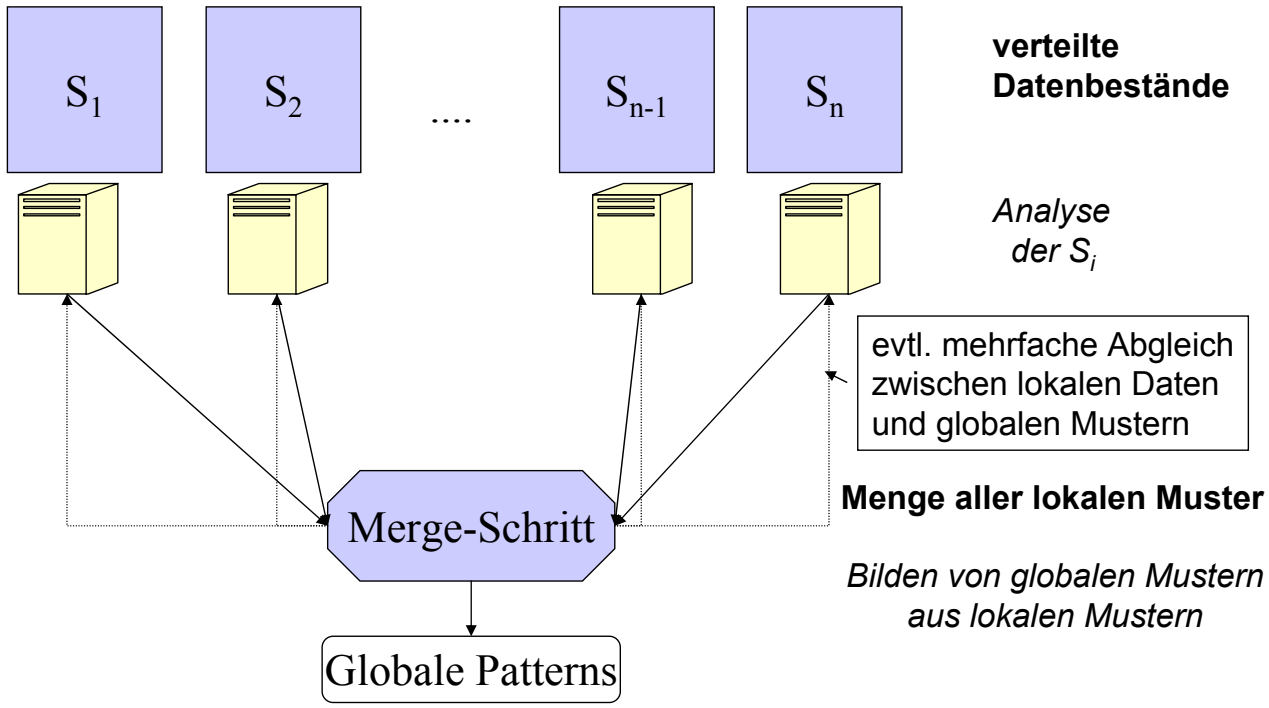
Verteiltes Data Mining

- Verteilung der Daten auf verschiedene Rechner ist vorgegeben
  - ⇒ Kein zusätzlicher Aufteilungsaufwand.
  - ⇒ Zusammenfügen lokaler Muster kann teuer oder unpräzise werden.
- ungünstige Verteilung kann folgende Auswirkungen haben:
  - Abweichung des verteilt errechneten Modells vom zentral errechneten Modell
  - hohe Kommunikations- und Berechnungszeiten zwischen den einzelnen Rechnern.

Fazit:

Parallele Data Mining Algorithmen können als Spezialfall von verteilten Algorithmen aufgefasst werden, bei denen man die Aufteilung der Daten auf verschiedene Sites S_i selber bestimmen darf, aber dafür Kosten für die Verteilung der Daten anfallen können.

Ablauf verteilter Algorithmen

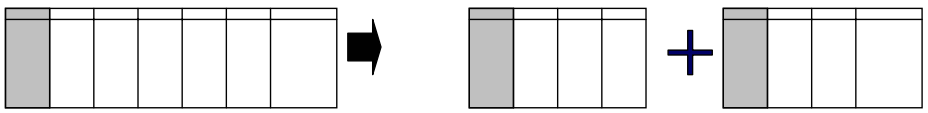


Kategorisierung der Algorithmen

1. Nach Aufteilung:

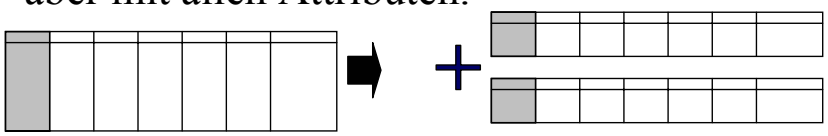
Vertikal verteilte Daten

Alle Datenobjekte auf allen Rechnern, aber nur jeweils ein Teil der Attribute auf jeden Rechner



Horizontal verteilte Daten

Datenobjekte kommen i.d.R. nur auf einem Rechner vor, dafür aber mit allen Attributen.



(Daten können auch vertikal und horizontalverteilt sein.)

Kategorisierung der Algorithmen

2. Nach Aufgabe:
Klassifikation, Clustering, Assoziationsregeln

3. Verteilungsabhängigkeit:
Ist das Ergebnis, von der Verteilung auf die Sites abhängig.
(Ergebnis verteiltes Verfahren = Ergebnis globales Verfahrens ?)

4. Art der Teilergebnisse:
Approximationen, Datenpunkte, Verteilungen...
Beispiele: Gauss-Kurven, Hyperrechtecke, Centroide...

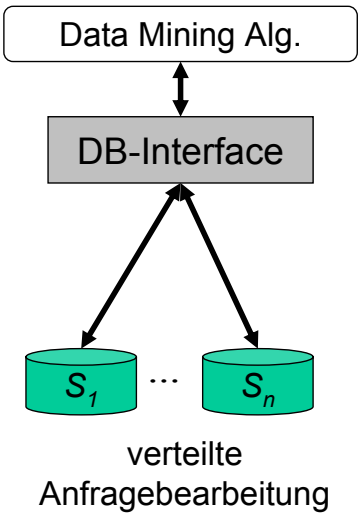
5. Nach Organisation der verteilten Berechnung:
1 Masterprozeß und viele Slaves, mehrere gleichberechtigte Prozeße

4.2 Parallele und Verteilte Algorithmen

- Verteilungsunabhängigkeit wird meist gefordert
  (Egal wie aufgeteilt wird, das Ergebnis bleibt gleich)
- Effizienzsteigerung steht im Vordergrund
- Aufteilung der Daten auf Sites ist entscheidend:
  - Probleme bei der Kombination der Teilergebnisse sollen minimiert werden
    $\Rightarrow$ lokale Muster sollten unabhängig voneinander sein
    $\Rightarrow$ bei Abhängigkeiten: mehrfache Kommunikation oder inexakte Muster
  - Gesamtlaufzeit ist von der längsten Laufzeit eines Teilschritts abhängig
    $\Rightarrow$ alle parallelen Schritte sollten möglichst gleich viel Zeit in Anspruch nehmen
    $\Rightarrow$ alle Sites sollten die gleiche Menge an Daten zugeteilt bekommen

Parallelisierung durch Parallele Anfragebearbeitung

Prinzipiell lassen sich Data Mining Algorithmen durch parallele Basisoperationen parallelisieren und beschleunigen:



Beispiele:

- Parallele Bearbeitung von ε-Range Queries erlaubt Performanz-steigerung für dichtebasiertes Clustering.
- Paralleles Bestimmen des nächsten Nachbarn, beschleunigt kNN-Klassifikation

Aber:

- auch hier Aufteilung entscheidend
- Korrektheit der Anfrage wird vorausgesetzt

Paralleles Dichtebasiertes Clustering

Grundidee:

- Horizontale und kompakte Aufteilung des Datenraums
- Bestimmung lokaler Kernpunkte und Cluster
- Verbinden von lokalen Clustern mit:
  - Clustern aus anderen Sites
  - Rauschpunkten aus anderen Sites

Grundproblem:

Was passiert mit Objekten deren ε-Umgebung aus dem Bereich der Site hinaus stehen ?

- Spiegeln der Ränder auf alle Sites
- Kommunikation mit der Datenbank einer anderen Site

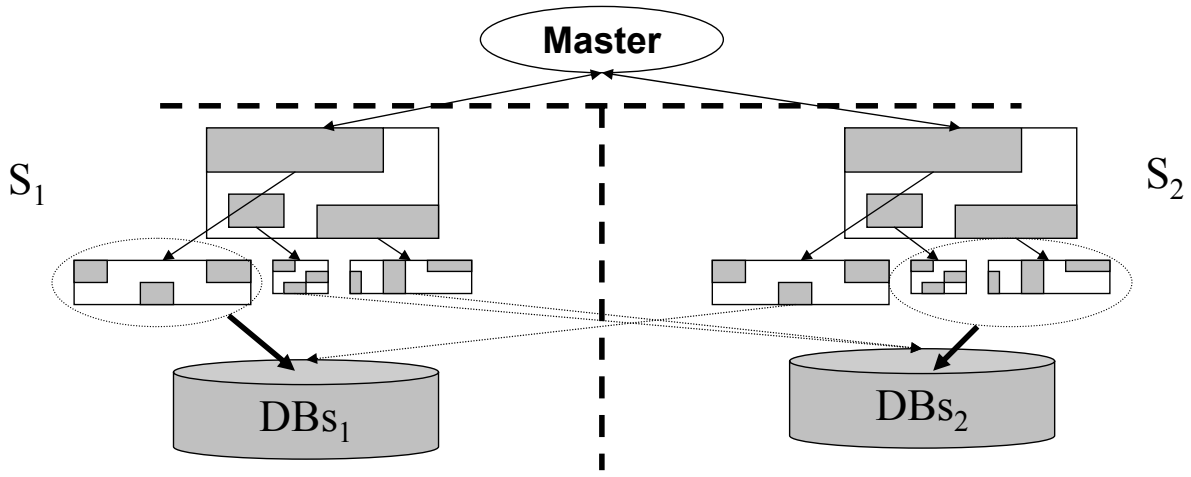
Lösungvorschläge

PDBSCAN [Xu Jäger,Kriegel99]

Grundidee:

- Abspeichern aller Featurevektoren in einer parallelisierten Indexstruktur (dR*-Tree)
  - Directory der R-Baums auf jeder Site
  - Datenseiten sind gemäß räumlicher Nähe auf Site verteilt
- Bestimmen lokaler Cluster auf den Sites
- Bei Punkten am Rand benötigt Anfragebearbeitung Zugriff auf die Blätter einer anderen Site.
- Mergen der Cluster mit gemeinsamen Punkten

Paralleler DBSCAN



Der dR*-Baum:

- erlaubt Range-Queries auf der gesamten Datenbank
- Anfragen auf S_i, die auf lokaler Partition DBs_i beantwortet werden kann lassen sich vollständig nebenläufig bearbeiten
- bei Zugriff auf Blätter einer anderen Seite sinkt die Nebenläufigkeit und die Kommunikationskosten steigen.

=> Algorithmus soll soweit wie möglich lokale Anfragen benutzen

PDBSCAN

- Aufteilung der Daten gemäß der Seiten im dR*-Baum (gleich Anzahl an Datenseiten auf jede Site)

Idee:

- DBSCAN läuft auf allen Punkten in p_1 und p_2
- Falls sich ε-Umgebung und Randbereich schneiden:
  1. Randbereiche müssen evtl. von anderer Site geladen werden um Kernpunkteigenschaft zu bestimmen
  2. Expansion der Punkte in den Randbereichen würde zu hohen Kommunikationskosten führen.

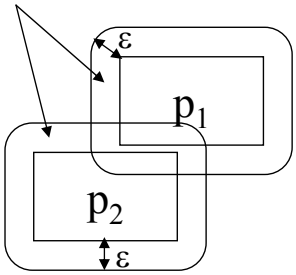
     => Punkte werden in Mergeliste gespeichert
- Zusammenführen der lokalen Cluster über gemeinsame Mergepunkte:

  Mergpunkt muss Kernpunkt in einer Partition sein

  => Verschmelze Cluster

lokale pages p_1 und p_2

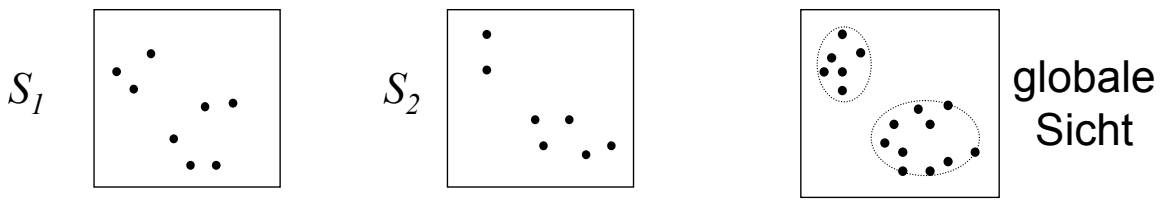
Randbereich



Verteiltes Dichtebasiertes Clustering

Bei beliebiger Verteilung der Daten auf lokale Sites S_i:

- Ausdehnung der Daten auf jeder Site S_i kann maximal überlappen
  - jede Site S_i kann Elemente der ε–Umgebung des Punkts P enthalten
  - P kann Kernpunkt sein, auch wenn P kein Kernpunkt auf einer beliebigen Teilmenge S‘ ⊂ S (=Menge aller Sites) ist.



- Dichtebasiertes Clustering verfügt über kein Clustermodell

  ⇒ Transfer lokaler Punkte, um ein globales Muster zu erzeugen

Verteiltes Dichtebasiertes Clustering

Verzicht auf Aufteilungsunabhängigkeit:

Grundidee:

- Bestimme Repräsentanten auf den lokalen Sites
- Übertrage Repräsentanten auf Master-Site
- Clustere Repräsentanten mit geeignetem Distanzmaß, dass berücksichtigt das Repräsentanten eine Menge von lokalen Punkten repräsentieren.
- Transfer der Repräsentaten zu den einzelnen Sites und Zuordnung der lokalen Punkte zu globalen Clustern.

Probleme:

- Resultierendes Clustering muß nicht mit dem Clustering, das DBSCAN auf allen Daten gefunden hätte übereinstimmen.
- lokale Rauschpunkte sind problematisch:
  - entweder Verzehrung der Repräsentaten
  - globale Cluster werden nicht gefunden, da lokale Rauschpunkte nicht berücksichtigt werden

Verteiltes Dichtebasiertes Clustering

SDBDC[Januzaj, Kriegel, Pfeifle‘04]

Berechnung der Representaten auf den lokalen Sites:

- Jeder Punkt p wird mit $Q(p) = \sum_{o \in \varepsilon - Range(p)} \varepsilon - d(o,p)$ bewertet
- Wähle den Punkt mit dem höchsten Wert für *Q(p)*
- Streiche bereits repräsentierte Punkte aus der DB
- Berechne erneut *Q(p)* für alle noch nich repräsentierten Objekte

⇒ für jede Site werden solange Repräsentanten gewählt bis alle Punkte repräsentiert wurden.

⇒ Berechne für jeden Repräsentanten:

⇒ den größten Abstand zu einem Punkt in der ε-Umgebung: ξ

⇒ die Anzahl der Punkte in der ε-Umgebung: μ

⇒ Übertrage die Punkte mit Radius und Anzahl an Master-Site

Verteiltes Dichtebasiertes Clustering

Clustern der Punkte auf der Master-Site:

- Clustering mit DBSCAN mit modifiziertem Kernpunktprädikat:
  Objekt *o* ist Kernpunkt, wenn die lokalen Repräsentanten $r_i \in \varepsilon\text{-}Range(o)$ mehr als *MINPTS* lokale Punkte repräsentieren.

$$\sum_{r_i \in \varepsilon - Range(o)} \mu_{r_i} \geq MinPts$$

- Daher: Umgebung des Kernpunkts *o* ist eigentlich ε + ξ ist (Umgebung in der alle ε-Umgebungen der repräsentierten Objekte enthalten sind)

- daher auch Erweiterung von Dichteerreichbarkeit auf ε + ξ.

Verteiltes Dichtebasiertes Clustering

Fazit:

- Ergebnisse stimmen nicht mit globalen Clustering auf allen Daten überein,
  aber: häufig gute Näherung
- Anzahl der Übertragenen Repräsentanten kann an Systemresourcen angepasst werden.

- Verfahren immer noch anfällig gegen Noise und globalen Cluster, die auf einzelnen Sites nicht erkennbar sind.

Verteiltes Partitionierendes Clustering

Idee:

- wenn Menge der zu übertragenden Daten gering ist, mehrfacher Abgleich zwischen lokalen und globalen Modellen kein Problem
- Bei *k*-Means oder verwandten Verfahren lässt sich Zielfunktion und Centroid auch verteilt berechnen.

$$TD^2 = \sum_{o \in DB} \left(\sum_{C_i \in C} \left(\min\{d(o, C_i)\} \right)^2 \right) = \sum_{S_j \in DB} \left(\sum_{o \in S_j} \left(\sum_{C_i \in C} \left(\min\{d(o, C_i)\} \right)^2 \right) \right)$$

globaler Centroid c_j zusammengesetzt aus den lokal zugeordneten Punkte $C_{i,j}$: $c_j = \frac{1}{\sum_{C_{i,j} \in DB} |C_{i,j}|} \cdot \sum_{C_{i,j} \in DB} \sum_{o \in C_{i,j}} o$

Fazit: In jedem Schritt kann das globale Clustering mit wenig Kommunikations-Overhead optimiert werden.

Verteiltes Partitionierendes Clustering

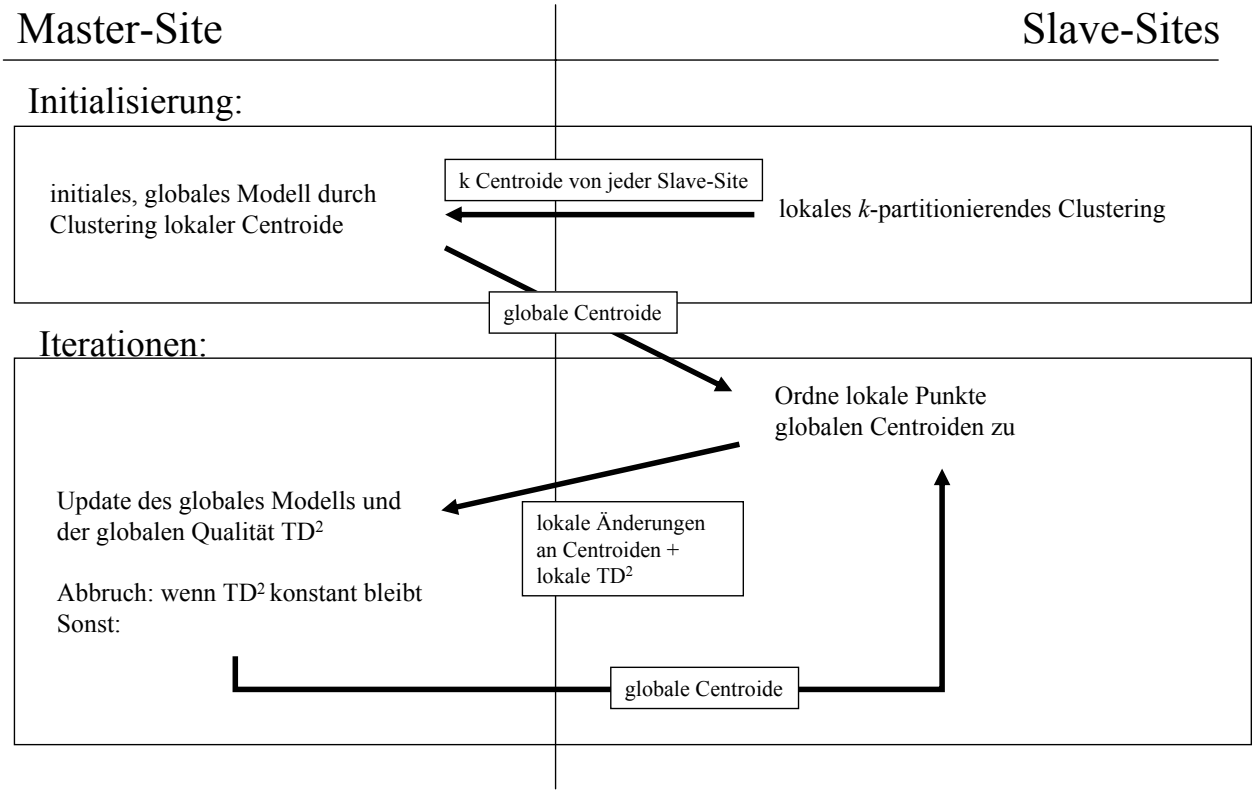
Verteiltes Clustering durch Varianzminimierung (Master-Slave):

- Bestimme initiale Aufteilung und Anfangs-Centroide

Schleife:

- Übermittle Centroide an die Sites
- Ordne lokale Punkte den Centroiden zu
  => berechne lokale Centroide und lokale TD^2-Werte
- Nach Rückübertragung von lokalen Centroiden, Cluser-Kardinalitäten und TD^2-Werten
  - ⇒ Kombination lokaler Centroide zu globalen durch Aufaddieren der Centroide und Kardinalitäten.(dynamische Berechnung des Centroids)
  - ⇒ Bestimmung des globalen TD^2-Werts
- Falls TD^2-Werte nicht verbessert wurde =>Abbruch

Ablauf Verteiltes Partitionierenden Clusterings



4.3 Privacy Preservation

Zusammenhang zum verteilten Data Mining:

⇒ Schutz nur bei mehreren Parteien notwendig:

Data-Owner: Hat Einblick in (einen Teil) der exakten Daten

Data-User: Möchte Muster aus den Daten ableiten.

Privacy Preserving Data Mining:

Erlaubt dem *Data-User* globale Muster aus den von den *Data-Ownern* zur Verfügung gestellten Daten abzuleiten, ohne dabei einzelne Datenobjekte zu diskreditieren:

1. *Data-User* darf keine Rückschlüsse auf Werte einzelner Datenobjekte ziehen können.
2. Die abgeleiteten Daten dürfen kein Wissen ableiten, dass Rückschlüsse über einzelne Datenobjekte erlaubt.

Privacy Preservation

Gründe für die Wichtigkeit von Privacy Preservation:

1. Viele Daten werden nur vom Data Owner zur Verfügung gestellt, wenn Privacy Preservation garantiert werden kann.

   Beispiel: Analyse des Surf-Verhaltens

2. Schutz vor Mißbrauch zur Verfügung gestellter Daten durch Dritte.

   Bsp: Veröffentliche Ergebnisse über das Surfverhalten, werden genutzt, um Spam-Mails zu personalisieren.

Fazit:

Nicht nur der Inhalt digitaler Medien ist schützenswert, sondern auch Informationen zur eigenen Person.

Privacy Preservation

Grundidee: Verändere die Datenmenge so, dass die Muster in den Daten gleich bleiben, aber die einzelnen Datenobjekte nicht mehr erkennbar sind.

Ziel abhängig von Generalisierung:

Muster die auf wenige Individuen zurückgehen, beschreiben diese oft so genau, dass Rückschlüsse möglich sind.

$\Rightarrow$ Muster müssen allgemein genug sein

Schutz der Privatshäre durch Weitergabe von:

- Veränderten Datenobjekten (Veränderte Daten)
- allgemeinen Modellen der Daten (Verteilungen)
- Daten die durch allgemeine Modelle generiert wurden (Sampling)

Grundtechniken der Privacy Preservation

Diskretisierung:

- Disjunktes Aufteilen des Wertebereichs in mehrere disrkrete Teilmengen / Intervalle.
- tatsächlicher Wert wird durch Werte-Klasse ersetzt

Beispiel:

Originalfakt: Person A verdient 42.213 € im Jahr

diskretisierte Aussage:

Person A verdient zwischen 35.000 € und 55.000 € im Jahr

Problem:

- Information zwar schwächer aber immer noch vorhanden
- Rückschlüsse sind möglich, wenn pro Klasse nicht genügend Objekte vorhanden sind.

$\Rightarrow$ gleichmässige Aufteilung des Wertebereichs nach Anzahl der Objekte (keine äuqidistante Aufteilung des Wertebereichs)

Grundtechniken der Privacy Preservation

Werte-Verzerrung (Data Perturbation):

- übertrage Summe des Werts w und einer Zufallszahl r: w_i+r_i
- Verteilungen für r :
  - Gleichverteilung $[-\alpha,..,\alpha]$ $\alpha \in IR^+$
  - Normalverteilung mit Erwartungswert 0 und Standardabweichung σ
- Da Störverteilung den Erwartungswert 0 hat und sich Erwartungswerte bei der Addition von Zufallsvariablen aufaddierten bleibt Erwartungswert der Werte $E(W)$ erhalten.

Grundtechniken der Privacy Preservation

Definition: „Privacy-Level“ (Qualität der Privacy Preservation)

Idee:

Wenn mit *c%* vorhergesagt werden kann, dass der Wert *x* im Interval $[x_1, x_2]$ liegt, dann entspricht die Breite des Intervals $(x_2\text{-}x_1)$ dem „Privacy-Level“ auf dem Konfidenzniveau *c%*.

Genauer:

Gegeben: Veränderter Feature-Wert y der durch Addition des Zufallswertes *r* auf den originalen Feature-Wert *x* erzeugt wurde.

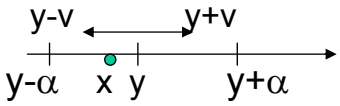
Gesucht: Breite *2v* des Intervals *[y-v..y+v]*, in dem der originale Wert *x* mit *c %* Wahrscheinlichkeit liegt. (*2v* ≡ Privacy-Level)

Privacy-Level

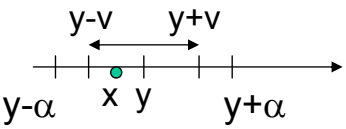
Beispiel:

Störeinfluß R ist gleichverteilt in [-α,..,α].

- für 100 % ⇒ α = v ⇒ Privacy = 2α
  (Wert muss im Interval liege)



- für 50 % ⇒ 2v/2 α = 0.5
  ⇒ v = 0.5 α
- allgemein: v = c% * α



Rekonstruktion der Originalverteilung

Gegen: Eine Menge von verzerrten Werten $Z = \{Z1,..,Zn\}$

Die Dichtefunktion der Störgrösse R: $f_R: IR \rightarrow [0..1]$

Gesucht: Die Verteilung der Originalwerte X mit der der Dichtefunktion $f_X: IR \rightarrow [0..1]$

Vorgehen : Abschätzung für einen Wert

$$F'_{X_1}(a) = \int_{-\infty}^{a} f_{X_1}(z|X_1 + Y_2 = w_1)dz = \int_{-\infty}^{a} \frac{f_{X_1+Y_1}(w_1|X_1 = z)f_{X_1}(z)}{f_{X_1+Y_1}(w_1)}dz$$

$$= \frac{\int_{-\infty}^{a} f_{X_1+Y_1}(w_1|X_1 = z)f_{X_1}(z)dz}{\int_{-\infty}^{\infty} f_{X_1+Y_1}(w_1|X_1 = z)f_{X_1}(z)dz} = \frac{\int_{-\infty}^{a} f_{Y_1}(w_1 - z)f_{X_1}(z)dz}{\int_{-\infty}^{\infty} f_{Y_1}(w_1 - z)f_{X_1}(z)dz} = \frac{\int_{-\infty}^{a} f_Y(w_1 - z)f_X(z)dz}{\int_{-\infty}^{\infty} f_Y(w_1 - z)f_X(z)dz}$$

Rekonstruktion der Originalverteilung

Aus $F_{X1}(a)$ lässt sich jetzt die Verteilung über alle Werte ermitteln:

$$F'_X(a) = \frac{1}{n}\sum_{i=1}^{n} F_{X_i}(a)$$

durch Differenzieren erhält man folgende Dichtefunktion:

$$f'_X(a) = \frac{1}{n}\sum_{i=1}^{n} \frac{f_Y(w_i - a)f_X(a)}{\int_{-\infty}^{\infty} f_Y(w_i - z)f_X(z)dz}$$

Da $f_X(a)$ unbekannt, nähert man $f'_X(a)$ iterativ an $f_X(a)$ an.

Rekonstruktion der Originalverteilung

Iterativer Näherungs-Algorithmus zur Rekonstruktion der Originalverteilung:

f^0_X:= Gleichverteilung

j:= 0 //Iterationzähler

repeat

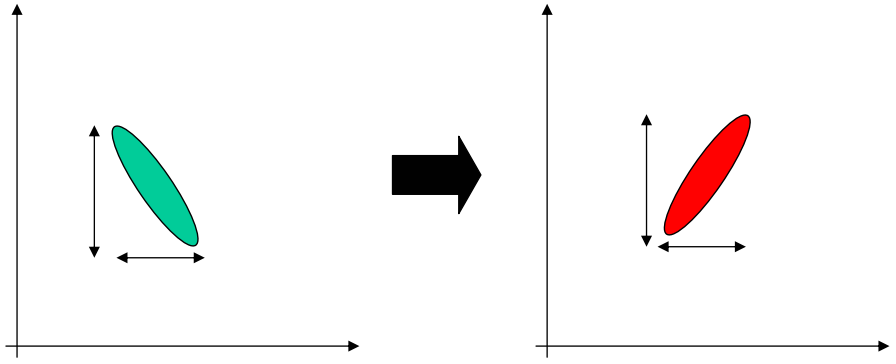
$$f_X^{j+1}(a) := \frac{1}{n}\sum_{i=1}^{n}\frac{f_Y(w_i - a) f_X^j(a)}{\int_{-\infty}^{\infty} f_Y(w_i - z) f_X^j(z) dz}$$

j:=j+1

until (Änderung < ε)

Weitere Techniken Privacy Preservation

- Vertausche Attributwerte X1.a mit X2.a. (Swapping)
  - ⇒ originale Feature-Vektoren sind nicht mehr nachvollziehbar
  - ⇒ bei Verfahren mit Unabhängigkeit zwischen den Dimensionen kein Unterschied der Verteilungen (Naive Bayes, Entscheidungsbäume)
  - ⇒ bei Mustern die durch Merkmalskorrelation gekennzeichnet sind werden Muster zerstört oder nicht vorhandene Muster zufällig generiert.



Privacy Preservation durch Datenapproximation

Idee: Anstatt einzelner Datenobjekte, kann der Data Owner auch lokale Modelle zur Verfügung stellen.

Diese dürfen dann die lokalen Dataobjekte nicht kompromittieren.

lokale Datenmodelle :

- Verteilungsfunktionen
- explizite Clusterbeschreibungen (Centroide, Verteilungsfunktionen)
- lokal häufige Muster

Privacy Preserving Verteiltes EM-Clustering

Grundidee:

- Approximiere die Daten mehrerer Data Owner durch Gauß-Cluster.
- Anforderungen an lokales Clustering:
  - Cluster dürfen keine lokalen Objekte diskreditieren
  - Anzahl der Cluster sollte so niedrig wie möglich sein (Transferkosten)
  - möglichst alle Datenobjekte sollte gut repräsentiert werden
- Nach Transfer zum Data User:
  - Kombination der lokalen Verteilung zu einer globalen Verteilung hierzu können 2 Strategien angewendet werden:
    - Benutze lokalen Verteilungen um neue Datenpunkte zu generieren, die die Verteilung beschreiben. (Sampling)
    - Kombiniere die lokalen Verteilungen direkt

Privacy Preserving Verteiltes EM-Clustering

Cover: Maß für die Beschreibungsgüte eines lokalen EM Clusterings.

Idee: Jedes Datenobjekt sollte von zumindest einem lokalen Cluster gut beschrieben werden.

$$Cov(M) = \left| \left\{ x \middle| x \in D \wedge \exists C_i \in M : P\left(x \middle| C_i\right) \geq t \right\} \right|$$

=> Verwende lokales Clustering mit dem max. Cover, das übertragen werden kann.

Privacy-Score: Misst die Bedrohung der „Privacy“ durch einen Cluster.

Idee: Cluster sollten ein Mindestmaß an Varianz über die dadurch beschriebenen Objekte enthalten.

$$PSCORE\left(C_i\right) = \sum_{j=1}^{d} \Sigma_{i,j}$$

=> Cluster C_i die einen $PSCORE(C_i) < \tau$ haben, dürfen nicht an den Data User geschickt werden

Privacy Preserving Verteiltes EM-Clustering

```
localEM(Database D, Integer kmax)
  maxcover = 0;
  bestClustering = ∅;
  for k := 1 to kmax do
     M := EM(D, k);
     if cover(M) = |D| then
      return M;
     end if
     if cover(M) > maxcover then
      maxcover = cover;
      bestClustering = M;
     end if
  end for
  return bestClustering;
```

kmax:
max. erlaubte Cluster-Anzahl

Privacy Preserving Verteiltes EM-Clustering

Aufbau eines globalen Clusterings beim Data User:

Beurteilen, ob 2 lokale Cluster C_1, C_2 zum selben globalen Cluster gehören

=> **Mutual Support**:

$$MS(C_1, C_2) = \int_{-\infty}^{\infty} N_{\mu_1,\Sigma_1}\left(\vec{x}\right) \; N_{\mu_2,\Sigma_2}\left(\vec{x}\right) \; d\vec{x}$$

2 Varianten:

- Verbinde Cluster bis globales Clustering aus k Gauß-Clustern besteht.
- Verbinde Cluster bis es keine Cluster mehr gibt, die einen kleineren Mutual Support als τ haben.

Privacy Preserving Verteiltes EM-Clustering

Für das Vereinen der Cluster C=$\{C_1,..,C_m\}$ wird der neue Erwartungsvektor μ_C wie folgt berechnet:

$$\mu_C = \frac{\sum_{k=1}^{m}\left(w_{C_k} \cdot \lambda(C_k) \cdot \mu_k\right)}{\sum_{k=1}^{m}\left(w_{C_k} \cdot \lambda(C_k)\right)}$$

Die neue Kovarianzmatrix Σ_C wird wie folgt gebildet:

$$\Sigma_C^{i,j} = \frac{\int_{-\infty}^{\infty}\left(\sum_{k=1}^{m}\left(w_{C_k} \cdot \lambda(C_k) \cdot N_{\mu_k,\Sigma_k}\left(\vec{x}\right) \cdot \left(\vec{x} - \mu_C^i\right) \cdot \left(\vec{x} - \mu_C^j\right)\right)\right) d\vec{x}}{\sum_{k=1}^{m}\left(\int_{-\infty}^{\infty}\left(w_{C_k} \cdot \lambda(C_k) \cdot N_{\mu_k,\Sigma_k}\left(\vec{x}\right)\right) d\vec{x}\right)}$$

Privacy Preserving Verteiltes EM-Clustering

```
globalMerge(SetOfLocalClusters C, Integer k)
  for each pair (Ci,Cj)∈ C do
      compute MS(Ci,Cj);
  end for
  sort the pairs w.r.t. descending mutual
  support;

  mark the first |C| – k pairs of clusters;

  build the transitive closure over the pairs
  having some common clusters and unite them into
  a common global cluster;
```


Privacy Preserving Verteiltes EM-Clustering

Problem: Das Berechnen eines multidimensionalen Integrals ist aufwendig.

Lösung: Unter der Annahme von Unabhängigkeit zwischen den Dimension gilt folgende Vereinfachung

Mutual Support:

$$MS(C_1, C_2) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi}\left(\sigma_2^i + \sigma_1^i\right)} \cdot \exp\left(-\frac{\left(\mu_1^i - \mu_1^i\right)^2}{2 \cdot \left(\sigma_2^i - \sigma_1^i\right)} \right)$$

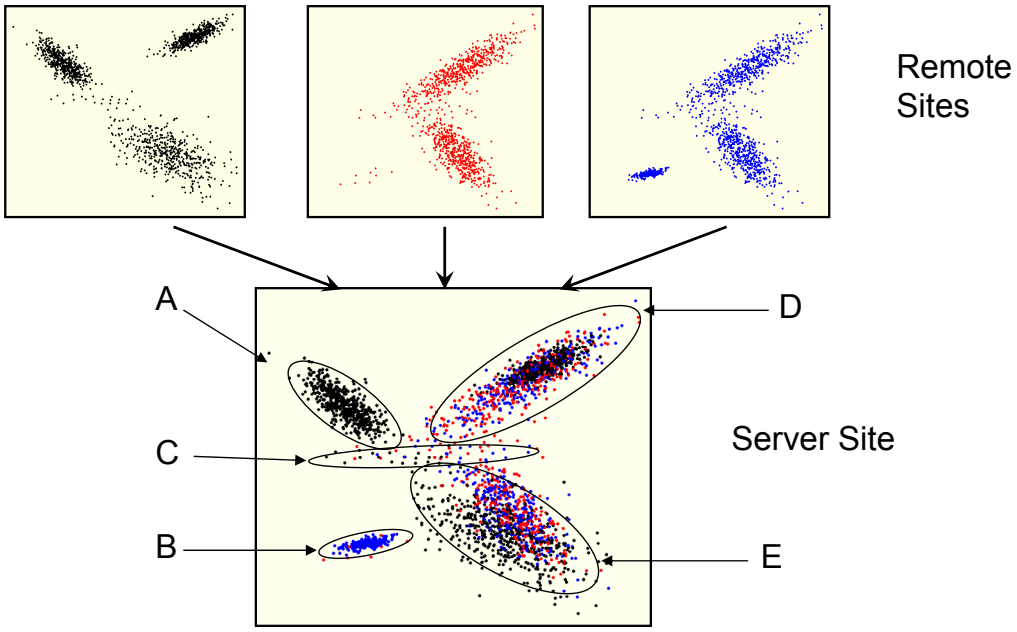
Standardabweichung in Dimension i:

$$\sigma_C^i = \sqrt{\frac{\int_{-\infty}^{\infty} \left(\sum_{k=1}^{m} \left(w_{C_k} \cdot \lambda(C_k) \cdot N_{\mu_k^i, \sigma_k^i}(x_i) \cdot \left(x_i - \mu_C^i\right)^2 \right) \right) dx_i}{\sum_{k=1}^{m} \left(\int_{-\infty}^{\infty} \left(w_{C_k} \cdot \lambda(C_k) \cdot N_{\mu_k^i, \sigma_k^i}(x_i) \right) \; dx_i \right)}}$$

Bemerkung: Berechnung des Centroids bleibt gleich.

Experimental Evaluation

2D Beispiel mit Correlation der Dimensionen:



Literatur

- X., Jäger J., Kriegel H.-P.: ***A Fast Parallel Clustering Algorithm for Large Spatial Databases***, in: Data Mining and Knowledge Discovery, an International Journal, Vol. 3, No. 3, Kluwer Academic Publishers, 1999
- Januzaj E., Kriegel H.-P., Pfeifle M.: ***Scalable Density-Based Distributed Clustering***, Proc. 8th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'04), Pisa, Italy, 2004, in: Lectures Notes in Computer Science, Springer, Vol. 3202, 2004
- Jagannathan G., Wright R.N. : ***Privacy Preserving Distributed k-Means Clustering over Arbitrarily Partitioned Data***, Proc. 11th ACM SIGKDD, 2005
- Kriegel H.-P., Kröger P., Pryakhin A., Schubert M.: ***Effective and Efficient Distributed Model-based Clustering***, Proc. 5th IEEE Int. Conf. on Data Mining (ICDM'05), Houston, TX, 2005
- Agrawal R., Srikant R.: ***Privacy-preserving data mining***", Proc. of the ACM SIGMOD Conference on Management of Data, Dallas, TX, 2000