

Seminar: Data Mining

Referat: Andere Möglichkeiten des Data Mining in verteilten Systemen

Ein Vortrag von Mathias Rohde

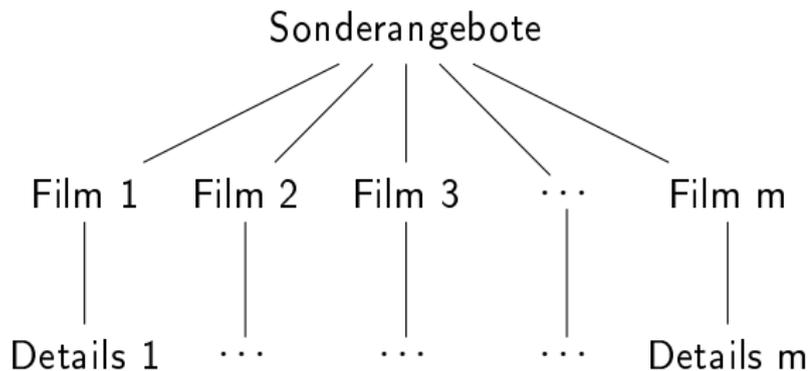
11. Juni 2009

Gliederung

- 1 Einführung
 - Problemstellung
- 2 Handwerkszeug
 - Vektorprodukt
 - Approximationen
 - Samplesammlung
- 3 Der Algorithmus
 - Schritte
 - Lokalität und Nachrichtenkomplexität

Worum geht es?

Nehmen wir an wir hätten auf einer Website folgendes Schema:



Worum geht es ?

- Nehmen wir einmal an die Seite würde die Klicks der Benutzer folgendermaßen speichern:

Benutzer	F1	D1	F2	D2	...	Fm	Dm
user1	1	0	0	1	0	0	2
user2	0	0	2	2	0	0	3
...

Eigentliches Ziel: top - k Items

- Den Seitenbetreiber interessiert nun, welche Punkte am meisten angeschaut wurden
- er möchte also eine Rangliste haben, und von dieser Rangliste interessieren ihn die ersten k
- wenn man nun also alle Daten von allen Benutzern hätte, müsste man nur die Anzahl an Klicks zählen und könnte danach ordnen

Ziel hier aber: top - l Items

- Wenn nun aber die Daten in einem Netz auf viele verschiedene Peers verteilt sind, ist einfaches zählen und ordnen nicht mehr möglich
- alle Daten zu überprüfen ist nicht mehr möglich also muss mit einer Stichprobe ausgekommen werden
- das heißt aber, dass nicht mehr garantiert alle top-k Items gefunden werden können, sondern nur eine Auswahl $l < k$

Matrix der inneren Produkte

- Als kompakte Schreibweise für die Daten, die ein Peer P_d enthält, bildet man eine $c \times c$ Matrix \mathbb{A}_d , wobei c die Anzahl der Wertevektoren ist
- in dieser Matrix ist der Eintrag \mathbb{A}_{ij} das Skalarprodukt zwischen dem i -ten und j -tem Wertevektor von P_d
- da diese Matrix symmetrisch ist und außerdem die Diagonale das Produkt eines Vektors mit sich selbst, ist nur das obere rechte Dreieck der Matrix relevant

Matrix der inneren Produkte - Beispiele

Hier ein kleineres Beispiel:

Benutzer	F1	D1	F2	D2
user1	1	0	4	1
user2	2	2	2	0

$$\text{Matrix} = \begin{pmatrix} - & 4 & 8 & 1 \\ - & - & 4 & 0 \\ - & - & - & 4 \\ - & - & - & - \end{pmatrix}$$

Vektorprodukte

- Seien x und y r -dimensionale Vektoren
- Das innere Produkt zwischen diesen beiden Vektoren ist definiert als $\langle x, y \rangle = \sum_{i=1}^r x_i y_i$
- Wenn nun die Werte von x und y im Netzwerk verteilt sind gilt also:
- $\langle x, y \rangle = \sum_{i=1}^r x_i y_i = \sum_{d=1}^S \left[\sum_{j=1}^{r_d} x_j y_j \right]$

Vektorprodukte - Beispiele

- Seien u und v zwei Vektoren mit

$$u = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

und

$$v = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

- Das innere Produkt dieser beiden Vektoren wäre dann

$$1 * 1 + 0 * 2 = 1$$

Ordinale Approximation 1/3

- Durch die Menge an Daten kann man nicht alle einzelnen Daten auswerten
- Stattdessen wird nur eine Auswahl berechnet
- Wie gross diese Auswahl mindestens sein muss, bestimmt die Ordinale Approximation

Ordinale Approximation 2/3

- Sei X nun eine Zufallsvariable mit der kontinuierlichen Dichtefunktion $F_X(x)$.
- Sei ξ_p der p -te Prozentanteil, bei dem gilt $F_X(\xi_p) = P\{x \leq \xi_p\} = p$.
- Angenommen man nimmt n unabhängige samples aus der Population X
- dann ordnet man die samples der Größe nach, so dass gilt $x_1 < x_2 < \dots < x_n$

Ordinale Approximation 3/3

- es soll dabei das n berechnet werden, bei dem gilt:
 $P\{x_n > \xi_p\} > q$
- für eine beliebige Confidence $0 \leq q < 1$
- dann gilt $P\{x > \xi_p\} = 1 - p^n$, also $1 - p^n > q$
- stellt man das ganz nach n um erhält man

$$n \geq \left\lceil \frac{\log 1 - q}{\log(p)} \right\rceil$$

Ordinale Approximation - Beispiel

- Wählt man zum Beispiel die confidence $q = 0.98$ und $p = 0.75$ erhält man durch Einsetzen in die Formel $n = 14$
- es ist also zu 98 % sicher, dass 75 % der Population unter dem höchsten Wert von 14 zufälligen Samples liegen
- also liegen nur 25 % der Samples über dem Wert von x_{14}

Kardinale Approximation

- Gleichzeitig zum ordinalen Approximieren müssen die gewählten Werte abgeschätzt werden
- Die Samples sind hier die Einträge der inneren Produktmatrix eines Knotens
- Für jedes Sample aus der Ordinalen Approximation müssen mehrere Knoten besucht werden um das Sample abzuschätzen
- Dabei wird die Hoeffding Grenze berechnet

Kardinale Approximation - Hoeffding Grenze 1/2

- Seien $x_i, i \in [1 \dots m]$ m unabhängige Samples einer Zufallsvariablen X im Intervall $[a, b]$
- Sei der Durchschnitt der Samples $Q_m = \frac{1}{m} \sum_i x_i$
- Für ein beliebiges $\epsilon > 0$ gilt dann

$$P\{Q_m - E(X) \geq \epsilon\} \leq \exp\left(-\frac{2m\epsilon^2}{(b-a)^2}\right)$$

$$P\{E(X) - Q_m \geq \epsilon\} \leq \exp\left(-\frac{2m\epsilon^2}{(b-a)^2}\right)$$

Kardinale Approximation - Hoeffding Grenze 2/2

- Außerdem gilt $P\{Q_m - E(X) \geq \epsilon\} \leq q'$
- Formt man nun beide Formeln nach m um erhält man

$$m \geq \frac{(b-a)^2 \ln\left(\frac{1}{q'}\right)}{2\epsilon^2}$$

Kardinale Approximation - Beispiel

- Sei $b - a = 4$, $q' = 0.01$ und $\epsilon = 0.5$
- dann erhält man für $m = 332$
- es werden also mindestens 332 samples gebraucht, dass die Wahrscheinlichkeit, dass der Durchschnittswert der Population bei einer Variation der Zufallsvariablen von 4 grösser als 0.5 ist, nur bei 1 Prozent liegt.

Random Sampling

- Kardinale Approximation gibt Menge an benötigten Samples an
- Samples werden danach erzeugt
- Peer to Peer Netzwerk wird als ungerichteter Graph aufgefasst
- Bilden einer Korrespondierenden Markovkette

Random Sampling - Beispiele

Sei A ein Graph mit 5 Knoten, die alle Grad 4 haben:
Von einem zufälligen Startknoten aus wäre die Wahrscheinlichkeit
des Übergangs also 0.25

$$A = \begin{pmatrix} 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 \end{pmatrix} \quad (1)$$

Random Sampling - Problem

- Die Wahrscheinlichkeit, dass ein Knoten gewählt wird, steigt mit der Anzahl seines Grades
- jeder Knoten soll aber die gleiche Wahrscheinlichkeit haben, gewählt zu werden
- deshalb wird ein anderer Algorithmus genutzt: Metropolis Hastings

Metropolis Hastings 1/2

- Sei $G(V, E)$ ein ungerichteter Graph mit $|V| = S$ Knoten und $|E| = e$ Kanten
- Sei ferner d_i der Grad eines Knoten i mit $1 \leq i \leq S$
- Die Menge von Nachbarknoten von i ist gegeben durch $\Gamma(i)$, wobei $\forall j \in \Gamma(i)$, ist die Kante $(i, j) \in E$
- Sei nun noch $T = \{p_{ij}\}$ die Matrix der Übergangswahrscheinlichkeiten wobei p_{ij} die Wahrscheinlichkeit ist, in einem Schritt von Knoten i zu Knoten j kommen
- Dabei gilt $0 \leq p_{ij} \leq 1$ und $\sum_j p_{ij} = 1$

Metropolis Hastings 2/2

- Eingabe: P_i mit Grad d_i
- Ausgabe: P_i ; die zugehörige Reihe T_i der Übergangsmatrix
- Initialisierung: P_i sendet eine Grad Nachricht an alle $P_j \in \Gamma(P_i)$
- Nach Erhalten der Antwort von allen $P_j \in \Gamma(P_i)$: p_{ij} wird berechnet als

$$p_{ij} = \frac{1}{\max(d_i, d_j)}, i \neq j, j \in \Gamma(i)$$

$$p_{ij} = 1 - \sum_{j \in \Gamma(i)} p_{ij}, i = j$$

$$p_{ij} = 0 \text{ sonst}$$

- Terminierung: Wenn alle p_{ij} berechnet sind setze $\{p_{i1}, p_{i1}, \dots, p_{is}\}$ in T_i ein

Algorithmus - Ablauf

Damit der Algorithmus nun ausgeführt werden kann, benötigt er folgende Parameter:

- Die Anzahl n an benötigten ordinalen Samples
- Die Anzahl m der benötigten zu besuchenden Knoten um jedes ordinale Sample abzuschätzen
- n Indizes der inneren Produktmatrix bezüglich der n ordinalen Samples
- Diese Werte werden alle durch die benötigte Confidence q , dem Prozentsatz p , der herausgefiltert werden soll, das Intervall R der Daten und den Genauigkeiten ϵ und q' (bestimmt vom Benutzer)

Algorithmus - Schritte

Die Schritte des Algorithmus sind nun um einzelnen:

- 1. Berechnung der Menge an benötigten Samples
- 2. Sammeln der Samples
- 3. Berechnen von Grenzwerten
- 4. Identifikation von einigen top-1 Elementen

Berechnung der benötigten Samples

- Initialpeer P_d bestimmt Confidencelevel q und Prozentsatz p , die er untersuchen möchte
- per OA berechnet er die Anzahl an Samples n , die nötig sind um einen Grenzwert zu berechnen, so dass jeder Wert über diesem Grenzwert in den top- p Elementen liegt
- Nach diesem Schritt kennt der Initiatorpeer die Werte n, m und die n benötigten Indizes der Produktmatrix

Sammeln von Samples

- Nun führt der Algorithmus $m \times n$ Random Walks aus um unabhängige Samples aus dem Netz zu bekommen
- Da es m Peers braucht, um einen Eintrag der Matrix abzuschätzen trägt jeder Random Walk die Ip-Adresse und die Portnummer des Initiatorpeers mit sich, um bei Terminierung das Ergebnis sofort an den Initiator zu schicken
- Am Ende dieses Schrittes hat P_d $m \times n$ Samples mit n verschiedenen Indizes und m Werte des inneren Produkts für jeden Index des inneren Produkts

Berechnen von Grenzwerten

- Der Initiatorpeer hat jetzt alle benötigten Samples beisammen und berechnet nun Grenzwerte
- Für jeden Index i werden nun alle m Einträge, die zu diesem Index gehören aufsummiert
- von diesen n Summen ist der größte nun der Grenzwert

Identifikation von einigen top- l Elementen

- An diesem Punkt kennt das Peer P_d eins der top- k Elemente (Wobei k Elemente im top- p Prozentsatz aller Daten liegen)
- Um nun l Elemente der top- k zu finden ($l < k$) muss das Peer $n \times m \times l$ Random Walks durchführen
- dann hat das Peer n/l Elemente und für je n kann ein Grenzwert gefunden werden, also l Grenzwerte
- durch die ordinale Struktur ist garantiert dass die l Werte zu den top- k gehören

α - Nachbarschaft

- Sei $G = (V, E)$ der Graph, der ein Netzwerk mit V Knoten und E Kanten repräsentiert
- Die α - Nachbarschaft eines Knotens $v \in V$ sind alle Knoten mit einer Distanz α oder weniger von v :
- $\Gamma_\alpha(v, V) = \{u \mid \text{dist}(u, v) \leq \alpha\}$,
- wobei dist die Länge des kürzesten Pfades zwischen u und v zurückliefert

α - lokale Anfrage

- Sei $G = (V, E)$ abermals ein Graph, definiert wie oben
- Jeder Knoten $v \in V$ speichert eine Datenmenge X_v
- eine α - lokale Anfrage eines Knotens v ist eine Anfrage, dessen Antwort mit Hilfe einer Funktion $f(X_\alpha(v))$ berechnet werden kann
- dabei gilt: $X_\alpha = \{X_v \mid v \in \Gamma_\alpha(v, V)\}$.

(α, γ) - Lokalität

- Ein Algorithmus ist dann (α, γ) - lokal, wenn niemals eine β - lokale Anfrage berechnet werden muss mit $(\beta > \alpha)$
- und außerdem die Größe der Antwort aller α lokalen Anfragen eines Peers durch γ beschränkt ist
- der hier vorgestellte Algorithmus ist $(O(\log(S)), nml)$ lokal

Nachrichtenkomplexität 1/2

- In jedem Randomwalk sendet der Initiatorpeer folgende Informationen mit:
 - 1. Tokennummer 32 Bits
 - 2. Index des Eintrags der inneren Produktmatrix, der überprüft wird 32 Bits
 - 3. IP - Adresse 32 Bits
 - Portnummer 32 Bits
- Die Nachrichtenkomplexität für das Sammeln der Samples ist also $128 \times n \times m \times l \times \lambda = 128nm l \lambda$ Bits

Nachrichtenkomplexität 2/2

- Nach jedem Randomwalk muss der terminierende Knoten noch das überprüfte Element an den Initialknoten schicken, was nochmal 64 Bits braucht
- Zusammenfassend ist die gesamte Nachrichtenkomplexität also $128nml\lambda + 64nml = O(nml\lambda)$
- Schreibt man n und m nun aus und setzt $\lambda = 10 * \log(S)$ ein erhält man insgesamt:

$$[1 + 20 \log(S)] \left[64l \frac{(b-a)^2 \ln(\frac{1}{q}) \log(1-q)}{2\epsilon^2 \log(p)} \right] \text{ Bits}$$