محمد امین رامی

تمرین دوم درس تحلیل داده های حجیم

پاییز ۱۴۰۱

سوال اول

الف) معیار conviction قدرت یک association rule را توصیف میکند. در صورت، احتمال نبود item set B و در مخرج، احتمال نبود item B به شرط item set A میباشد. پس صورت rare بودن item set B و مخرج عدم ارتباط item set A و item set B را میسنجد. برای مثال اگر item set B کمیاب باشد اما confidence(A => B) زیاد باشد، مقدار conviction زیاد میشود که نشانگر یک association rule قوی است.

ب) واضح است که confidence متقارن نیست. زیرا:

$$Confidence(A \to B) = \frac{P(A \cap B)}{P(A)}$$

$$Confidence(B \to A) = \frac{P(A \cap B)}{P(B)}$$

باتوجه به رابطه بالا، به وضوح confidence متقارن نیست.

اما lift متقارن است. زیرا:

$$lift(A \to B) = \frac{\frac{P(A \cap B)}{P(A)}}{P(B)} = \frac{P(A \cap B)}{P(A)P(B)} = lift(B \to A) = lift(B \to A)$$

Conviction متقارن نیست. زیرا:

$$conv(A \to B) = \frac{1 - S(B)}{1 - conf(A \to B)} = \frac{1 - S(B)}{1 - \frac{S(A \cap B)}{S(A)}} = \frac{S(A) - S(B)}{S(A) - S(A \cap B)}$$

$$conv(B \to A) = \frac{1 - S(A)}{1 - conf(B \to A)} = \frac{1 - S(A)}{1 - \frac{S(A \cap B)}{S(B)}} = \frac{S(B) - S(A)}{S(B) - S(A \cap B)}$$

واضح است عبارات رو به رو برابر نیستند پس conviction متقارن نیست.

ج) ضعف اصلی confidence این است که احتمال رخداد B را در نظر نمیگیرد. برای مثال فرض کنید B یک item set پرتکرار باشد که در اکثر سبدها وجود دارد. در این صورت احتمال اینکه B را در سبدی که A را دارد پیدا کنیم زیاد است. یعنی در اکثر سبد هایی که A است، B نیز به دلیل فراوانی اش حضور دارد. پس $\frac{P(A\cap B)}{P(A)}$ مقدار بالایی خواهد داشت. این در حالی است که A و B دارای یک association rule نیستند.

از طرفی اگر B یک item set کمیاب باشد، در سبدهای کمی از A ظاهر میشود. اما دلیل آن کمیاب بودن B است نه نبود یک association rule بین A و B. پس این ضعف معیار confidence است.

سوال دوم

ادعا میکنیم که سبدها با شماره بین ۱۱ تا ۲۰، Maximal frequent itemset هستند. توجه کنید که سبدها با شماره بین ۲۱ تا ۱۰۰ نمیتوانند Maximal frequent itemset باشند چون اصلا frequent نیستند. زیرا تعداد مضارب آنها که کوچکتر از ۱۰۰ هستند حداکثر ۴ تاست. پس frequent نیستند.

همچنین سبدها با شماره بین ۱۱ تا ۲۰ frequent هستند. زیرا حداقل ۵ تا مضرب خود را دارند که از ۱۰۰ کوچک تر است.

حال ادعا میکنیم که سبدها با شماره بین ۱ تا ۱۰ Maximal نیستند. زیرا اگر مثلا عضو جدید ۲ را به آنها اضافه کنیم، itemset جدید برابر با یکی از itemset های شماره ۱۱ تا ۲۰ خواهد شد که frequent است. پس itemset ها با شماره ۱ تا ۱۰ Maximal frequent itemset نیستند. پس تنها Maximal frequent itemset های ما به صورت زیر هستند:

$I_{11} = \{1, 11\},\ \ I_{12} = \{1, 2, 3, 4, 6, 12\},\ \ I_{13} = \{1, 13\},$

$I_{14} = \{1, 2, 7, 14\}, I_{15} = \{1, 3, 5, 15\}, I_{16} = \{1, 2, 4, 8, 16\}$

$I_{17} = \{1, 17\}, I_{18} = \{1, 2, 3, 6, 9, 18\}, I_{19} = \{1, 19\}$

$I_{20} = \{1, 2, 4, 5, 10, 20\}$

سوال سوم

برای حل مسئله از الگوریتم A-Priori استفاده میکنیم.

ابتدا تمام frequent itemset های تک عضوی را با استفاده از Map Reduce روی تمام سبد ها اجرا میکنیم:

Map: ({item}, 1) for each baskets and single item sets Reduce: add operation

بدین صورت میزان frequency هر item مشخص میشود. حال آیتم هایی که frequency آنها از support threshold بیشتر است را به عنوان item های frequent نگه میداریم.

سبد تمام item هایی که frequent نیستند را از تک تک سبدها حذف میکنیم.

حال تعداد itemset های دو عضوی هر سبد را با استفاده از Map Reduce می شماریم:

Map: ({item1, item2}, 1) for all item1, item2 pairs in each basket Reduce: add Operation

سپس تمام itemset های دوتایی را میشماریم و آنهایی که میزان frequency آنها از support threshold بیشتر است به عنوان itemset های دوتایی frequent ذخیره میکنیم.

برای مرحله itemset های سه تایی نیز دقیقا مانند حالت حالت دوتایی است.

پس از انجام این مراحل، نتایج زیر حاصل میشود:

پرتکرارترین itemset های تکی:

```
===== 10 Most frequent items are: =====
1- ('DAI62779', 6667)
2- ('FRO40251', 3881)
3- ('ELE17451', 3875)
4- ('GRO73461', 3602)
5- ('SNA80324', 3044)
6- ('ELE32164', 2851)
7- ('DAI75645', 2736)
8- ('SNA45677', 2455)
9- ('FRO31317', 2330)
10- ('DAI85309', 2293)
===== Number of frequent items are 647 =====
```

پرتکرارترین itemset های دوتایی:

```
===== 10 Most frequent two tuples are: =====
1- (('DAI62779', 'ELE17451'), 1592)
2- (('FRO40251', 'SNA80324'), 1412)
3- (('DAI75645', 'FRO40251'), 1254)
4- (('FRO40251', 'GRO85051'), 1213)
5- (('DAI62779', 'GRO73461'), 1139)
6- (('DAI75645', 'SNA80324'), 1130)
7- (('DAI62779', 'FRO40251'), 1070)
8- (('DAI62779', 'SNA80324'), 923)
9- (('DAI62779', 'DAI85309'), 918)
10- (('ELE32164', 'GRO59710'), 911)
===== Number of frequent two tuples are 1334 =====
```

پرتکرارترین آیتم های سه تایی:

```
===== 10 Most frequent three tuples are: =====



1- (('DAI75645', 'FRO40251', 'SNA80324'), 550)
2- (('DAI62779', 'FRO40251', 'SNA80324'), 476)
3- (('FRO40251', 'GRO85051', 'SNA80324'), 471)
4- (('DAI62779', 'ELE92920', 'SNA18336'), 432)
5- (('DAI62779', 'DAI75645', 'SNA80324'), 421)
6- (('DAI62779', 'ELE17451', 'SNA80324'), 417)
7- (('DAI62779', 'DAI75645', 'FRO40251'), 412)
8- (('DAI62779', 'ELE17451', 'FRO40251'), 406)
9- (('DAI75645', 'FRO40251', 'GRO85051'), 395)
10- (('DAI62779', 'FRO40251', 'GRO85051'), 381)


[Stage 77:>


===== Number of frequent three tuples are 233 =====
```

سوال چهارم

ابتدا نحوه حل مسئله را مرحله به مرحله شرح میدهیم.

مرحله اول: پاک سازی داده های پرت

داده های داده شده را به دوصورت ارزیابی میکنیم. ابتدا سطر هایی که در آنها ORIGINE_CAR_KEY و FINAL_CAR_KEY برابر نیستند را حذف میکنیم زیرا داده پرت هستند.

همچنین در برخی موارد این مشکل رخ داده است که وقتی یک ماشین از یک دوربین رد شده است، پلاک آن چندین بار پشت سر هم ثبت شده است. پس در دیتای ثبت شده، مواردی که در کمتر از یک دقیقه چندین تا پلاک یکسان ثبت شده است را حذف میکنیم و فقط یکبار آن پلاک را تاثیر میدهیم.

مرحله دوم: ساخت Key, Value مناسب

برای هر سطر جفت (key, value) را به صورت زیر میسازیم:

Key = (plate, date) Value = camera code

توجه کنید در date فقط به روز توجه میشود.

سپس یک groupByKey میزنیم تا دیتا به صورت زیر شود:

Key = (plate, date) Value = [list of camera codes]

مرحله سوم: تعیین ساپورت

فرض میکنیم که تعداد آیتم ها از یک توزیع گوسی پیروی میکند. بنابراین ترشولد ساپورت را برابر میانگین تکرر آیتم ها به علاوه یک انحراف معیار در نظر میگیریم:

Support threshold = mean(item frequency) + stdev(item frequency)

مرحله چهارم: A-Priori Algorithm

حال از الگوریتم A-Priori استفاده میکنیم. همچنین در حین ساختن دوتایی ها از ساختار داده set استفاده میکنیم تا دوتایی با آیتم تکرار تولید نشود. نحوه پیاده سازی این الگوریتم در سوال توضیح داده شده است و از تکرار دوباره آن می پرهیزیم.

نتایج به دست آمده به صورت زیر خواهد بود:

پرتکرارترین دوتایی ها:

```
number of two tuples is: 24
==================================
===== top frequent two tuples =====
(('900212', '900244'), 55733)
(('900142', '900212'), 34622)
(('100700841', '900101'), 25697)
(('100700853', '900142'), 24949)
(('100700864', '900185'), 23455)
(('100700853', '900212'), 23150)
(('100700868', '900222'), 22815)
(('100700841', '900236'), 22592)
(('100700824', '900107'), 21723)
(('900142', '900244'), 21102)
```

پرتکرارترین سه تایی ها:

```
number of three tuples is: 24
================================
===== top frequent three tuples =====
(('175', '203902', '900191'), 48710)
(('100700853', '900142', '900212'), 40184)
(('100700868', '900155', '900222'), 36422)
(('900142', '900212', '900244'), 33052)
(('100700853', '900212', '900244'), 27559)
(('22010119', '900108', '900268'), 25845)
(('22009977', '900225', '900268'), 25806)
(('100700839', '900212', '900244'), 22235)
(('100700853', '900142', '900244'), 20402)
(('22010118', '900215', '900256'), 18280)
```

مرحله پنجم: SON Algorithm

در این الگوریتم به این صورت عمل میکنیم که ابتدا دیتا را به سه قسمت تقسیم میکنیم. اینکار را با یک hash مناسب انجام میدهیم. تابع hash ما به این صورت است که camera code های موجود در یک سبد را جمع میکند و حاصل تقسیم آن بر ۳ را حساب میکند. به این ترتیب دیتای ما به سه قسمت تقسیم میشود.

سپس با استفاده از الگوریتم A-Priori روی هرکدام از این قسمت های کوچک تر، یکسری کاندید برای frequent itemset از هر قسمت به دست می آید. برای support threshold برای هر قسمت، مقدار $\frac{support\ threshold}{3*relaxing\ factor}$ را در نظر میگیریم که $support\ threshold$ مقداری است که در مرحله قبل محاسبه شد. Relaxing factor نیز عددی بزرگ از ۱ است که باعث میشود threshold ما کمی relax تر شود تا false negative نداشته باشیم.

حال برای کاندیدهای به دست آمده، اجتماع این کاندیدا هارا در نظر میگیریم. حال درستی این کاندید هارا را روی کل دیتا verify میکنیم. یعنی چک میکنیم که میزان frequency هر کاندید از support threshold بیشتر هست یا نه. در نهایت آن کاندیدهایی که میزان frequency آنها از support threshold بیشتر است، به عنوان frequent itemset معرفی میکنیم.

پس از ران کردن این الگوریتم، نتایج زیر حاصل میشود:

```
number of two tuples is: 24
==================================
===== top frequent two tuples =====
(('900212', '900244'), 55733)
(('900142', '900212'), 34622)
(('100700841', '900101'), 25697)
(('100700853', '900142'), 24949)
(('100700864', '900185'), 23455)
(('100700853', '900212'), 23150)
(('100700868', '900222'), 22815)
(('100700841', '900236'), 22592)
(('100700824', '900107'), 21723)
(('900142', '900244'), 21102)
```

پرتکرارترین دوتایی ها:

پرتکرارترین سه تایی ها:

```
number of three tuples is: 24
================================
===== top frequent three tuples =====
(('175', '203902', '900191'), 48710)
(('100700853', '900142', '900212'), 40184)
(('100700868', '900155', '900222'), 36422)
(('900142', '900212', '900244'), 33052)
(('100700853', '900212', '900244'), 27559)
(('22010119', '900108', '900268'), 25845)
(('22009977', '900225', '900268'), 25806)
(('100700839', '900212', '900244'), 22235)
(('100700853', '900142', '900244'), 20402)
(('22010118', '900215', '900256'), 18280)
```

مقایسه دو روش:

مشاهده میشود که هردو روش به نتایج یکسانی رسیده اند. این پدیده قابل پیش بینی بود زیرا هر دو الگوریتم دو approach متفاوت برای حل یک مسئله واحد هستند.

هردو روش مزایایی دارند. برای مثال روش A-Priori سرراست تر است از طرفی روش SON دیتا را به چند قسمت تقسیم میکند که میشود هر قسمت را روی یک node پخش کرد.