به نام خدا

دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

درس پردازش داده‌های حجیم

استاد حقیرچهرقانی

تمرین اول

علیرضا مازوچی

۴۰۰۱۳۱۰۷۵

بخش اول: سوالات تشریحی

سوال ۱

الف) با بررسی لیست تراکنش‌ها به نام بازی‌های زیر بر می‌خوریم:

- Horizon Forbidden West
- GTA V
- Gran Turismo 7
- Ghost of Tsushima
- Dying Light 2
- MAFIA: Trilogy
- FIFA 22
- The Last of Us Part II
- Far Cry 6

تعداد این‌ها ۹ تاست و یک مجموعه می‌تواند حداکثر شامل تمام اعضا باشد؛ پس حداکثر می‌توان یک مجموعه‌ی ۹تایی تشکیل داد؛ اگر support برابر با صفر باشد.

ب) همانطور که در قسمت قبل گفته شد، ۹ آیتم وجود دارد. هر مجموعه سه‌تایی باید انتخاب ۳ عضو از این‌ها باشد. پس بیشینه تعداد مجموعه‌های سه تایی $\binom{9}{3} = 84$ است که برای support برابر با صفر رخ خواهد داد.

ج) بدیهی است که یک مجموعه دارای Support کمتر مساوی‌ای از زیرمجموعه‌های خود است. پس برای پیدا کردن مجموعه با اندازه حداقل ۲ که بیشترین Support را دارد لازم و کافی است که تمام مجموعه‌های با اندازه دقیقا ۲ را بررسی کنیم. به نظر می‌رسد مجموعه‌ی {FIFA 22, The Last of Us Part II} با ۵ بار تکرار بیشترین فراوانی را در میان نامزدها دارد.

د) فرمول Confidence به شرح زیر است:

$$Conf(I \to j) = \frac{Support(I \cup j)}{Support\ (I)}$$

صورت این کسر برای هر دو قانون $A \to B$ و $B \to A$ یکسان است. پس برای برابری Confidence کافی است تا Support آیتم A با Support آیتم B برابر باشد. به عنوان مثال GTA V و The Last of Us Part II هر دو ۸ بار تکرار شده‌اند.

سوال ۲

یک مجموعه جز مرز منفی است اگر و فقط اگر خودش غیرفراوان باشد ولی تمام زیرمجموعه‌های بلافاصله‌اش فراوان باشد. با این تعریف می‌توان اعضای زیر را جز مرز منفی قرار داد:

$$\{ABD, BCD, ACD, AE, BE, CE, DE, F\}$$

سوال ۳

الف) ۹ جفت کلید-مقدار زیر تشکیل می‌شود:

$$(10, (R, 1)), (10, (R, 2)), (11, (R, 3)), (10, (R, 4))$$

$$(10, (S, 20)), (11, (S, 21)), (12, (S, 22)), (10, (S, 23)), (11, (S, 24))$$

ب) جفت کلید-مقدارها با توجه به مقادیر کلید در مرحله grouped by به سه گروه تقسیم می‌شوند:

$$(10, (R, 1)), (10, (R, 2)), (10, (R, 4)), (10, (S, 20)), (10, (S, 23))$$

$$(11, (R, 3)), (11, (S, 21)), (11, (S, 24))$$

$$(12, (S, 22))$$

بنابراین سه عمل Reduce داریم و اندازه یک Reducer حداقل باید ۵ باشد تا به مشکل نخوریم. چراکه برای کلید ۱۰، تعداد پنج جفت کلید–مقدار داریم.

ج) خروجی هر یک از Reducer ها عبارت است از:

$$(10, (R, 1)), (10, (R, 2)), (10, (R, 4)), (10, (S, 20)), (10, (S, 23))$$
$$\rightarrow (1, 10, 20), (1, 10, 23), (2, 10, 20), (2, 10, 23), (4, 10, 20), (4, 10, 23)$$

$$(11, (R, 3)), (11, (S, 21)), (11, (S, 24)) \rightarrow (3, 11, 21), (3, 11, 24)$$

$(12, (S, 22)) \rightarrow -$

لذا هشت خروجی زیر را خواهیم داشت:

$(1, 10, 20), (1, 10, 23), (2, 10, 20), (2, 10, 23), (4, 10, 20), (4, 10, 23), (3, 11, 21)$

$, (3, 11, 24)$

سوال ۴

در صورت سوال یک ماتریس جایگشت داده شده است و من هم از همان استفاده می‌کنم (اگر قرار به انتخاب اختیاری جایگشت باشد) ماتریس M بدین ترتیب حاصل می‌شود:

۱	۳	۲	۱	۱
۵	۱	۱	۳	۱
۱	۴	۳	۱	۱

برای سنجش شباهت، از سه جفت ستون اول (از سمت چپ) استفاده می‌کنم:

شباهت واقعی ستون اول و دوم: $\frac{۲}{۷}$

شباهت هش ستون اول و دوم: ۰

شباهت واقعی ستون اول و سوم: $\frac{۱}{۸}$

شباهت هش ستون اول و سوم: ۰

شباهت واقعی ستون دوم و سوم: $\frac{۴}{۶}$

شباهت هش ستون دوم و سوم: $\frac{۱}{۳}$

بخش دوم: سوالات پیاده‌سازی

سوال ۱

اگر بتوانیم برای هر دو کاربر بدانیم چه تعداد همسایه مشترک داشته‌اند قادر به پاسخگویی به این مسئله خواهیم بود. سپس به آسانی می‌توانیم برای هر کاربرد لیست کاربران دیگر با بیشترین همسایه مشترک را در بیاوریم و از میان آن‌ها کاربران دوست فعلی را حذف کنیم و سایر کاربران را پیشنهاد دهیم.

ورودی تابع Map پیاده‌سازی شده من یک سطر از مجموعه‌داده است و خروجی آن تعدادی جفت کلید-مقدار است که این تعداد دقیقا برابر با تعداد دوستان کاربر همان سطر از مجموعه‌داده است. برای تابع Map آیدی خود کاربر اهمیت ندارد بلکه لیست کاربران دوست آن است که اهمیت دارد. بدین شکل که برای هر کاربر در این لیست یک دیکشنری تشکیل می‌دهیم. به ازای سایر دوستان یک کلید در این دیکشنری قرار می‌دهیم و مقدار این کلید را برابر یک قرار می‌دهیم. برای درک بهتر یک مثال را در نظر بگیرید؛ فرض کنید کاربر شماره ۰ با کاربران ۱، ۲ و ۳ دوست است. در این صورت سه جفت کلید-مقدار زیر تولید می‌شود:

$$key = 1, value = \{2: 1, 3: 1\}$$

$$key = 2, value = \{1: 1, 3: 1\}$$

$$key = 3, value = \{1: 1, 2: 1\}$$

ورودی تابع Reduce پیاده‌سازی شده من طبیعتا لیستی از جفت کلید-مقدار ها با مقدار کلید یکسان است. در این تابع عملیات ادغام دیکشنری‌ها انجام می‌شود؛ بدین شکل که خروجی این تابع یک دیکشنری ادغام‌شده است که برای هر کلید موجود در دیکشنری‌های ورودی یک کلید دارد. مقدار این کلید برابر با حاصل جمع اعداد موجود در تمام دیکشنری‌هاست. برای درک بهتر فرض کنید لیست جفت کلید-مقدارهای زیر را برای کلید ۱ داشته باشیم:

$$key = 1, value = \{2: 1, 3: 1\}$$

$$key = 1, value = \{2: 1, 4: 1\}$$

$$key = 1, value = \{2: 1, 5: 1\}$$

خروجی تابع Reduce برای این‌ها برابر است با:

$$key = 1, value = \{2: 3, 3: 1, 4: 1, 5: 1\}$$

در این دیکشنری خروجی مشخص است که کاربر شماره ۲ سه مرتبه دوست مشترک کاربر ۱ بوده است که اگر کاربر ۲ در حال حاضر دوست کاربر ۱ نباشد، بهترین پیشنهاد دوستی وی خواهد بود.

در اینجا لازم است به چندین نکته هم اشاره کنم؛ اول آنکه کلیدهای در دیکشنری‌ها به صورت مرتب نگهداری می‌شوند تا هزینه ادغام در فاز Reduce کم شود. دوم آنکه در روش پیاده‌سازی شده توسط من برای پیشنهاد یک لیست t تایی از کاربران نیاز به اجرای یک تابع نهایی است. در این تابع دوستان فعلی هم حذف می‌شود. بدیهی است که این تابع هم می‌تواند به صورت موازی و برای یک t خاص روی کل داده‌ها اجرا شود و همچنین می‌توان دیکشنری را بر اساس پرتعدادترین مرتب نگه داشت ولی ادغام را پرهزینه‌تر می‌کند.

نهایتا ده دوست پیشنهادی برای کاربران مدنظر در سوال به شرح زیر است:

کاربر	لیست دوستان پیشنهادی
98	18560, 2554, 13654, 16324, 16350, 30134, 30691, 17, 113, 134
135	13792, 33060, 629, 5490, 19217, 25256, 34151, 34164, 34441, 45054
117	34164, 12519, 13793, 15314, 23507, 23510, 25256, 34140, 34169, 34207
911	24456, 39540, 40560, 30984, 30993, 30995, 30996, 33333, 37875, 41352
8804	34179, 34332, 3230, 8677, 11399, 11400, 13182, 13872, 15207, 29745

سوال ۲

۱) بهترین قوانین دوتایی به ترتیب confidence عبارت است از:

قانون	Confidence
$Dying\ Light\ 2 \rightarrow\ The\ Last\ of\ Us\ Part\ II$	۹۱/۵۰٪
$ARK: Survival\ Evolved \rightarrow\ The\ Last\ of\ Us\ Part\ II$	۹۱/۳۶٪
$ARK: Survival\ Evolved \rightarrow\ GTA\ V$	۹۱/۳۶٪
$ARK: Survival\ Evolved \rightarrow\ UNCHARTED\ 4$	۹۱/۲۵٪
$Ghost\ of\ Tsushima \rightarrow\ UNCHARTED\ 4$	۹۱/۱۹٪

۲) بهترین قوانین سه‌تایی به ترتیب confidence عبارت است از:

قانون	Confidence
(Assassin's Creed Odyssey, $DAYS\ GONE$) $\rightarrow The\ Last\ of\ Us\ Part\ II$	۹۶/۳۵٪
(A Way Out, Ghost of Tsushima) $\rightarrow UNCHARTED\ 4$	۹۶/۳۴٪
(Far Cry 6, Ghost of Tsushima) $\rightarrow GTA\ V$	۹۶/۳۳٪
(ARK: Survival Evolved, Red Dead Redemption 2) $\rightarrow$ UNCHARTED 4	۹۶/۳۳٪
(Ghost of Tsushima, Gran Turismo 7) $\rightarrow\ The\ Last\ of\ Us\ Part\ II$	۹۶/۳۳٪

سوال ۳

الف) این تابع ورودی‌های زیر را می‌گیرد:

- A: این ورودی یک ماتریس از مجموعه‌داده اصلی برنامه بدون هیچگونه تغییر و هشینگ است. هر سطر نماینده یک داده و هر ستون نماینده یک ویژگی است.
- hashed_A: این آرگومان شامل هش‌های هر سطر از داده است. این آرگومان در اصل لیستی است که آیتم i-ام آن متعلق به مقادیر هش آیتم i-ام در مجموعه‌داده است. مقادیر هش برای هر آیتم هم لیستی از مقادیر رشته‌ای است که هر یک هش‌شده آن داده به ازای یکی از توابع هش است.
- functions: این ورودی شامل L تابع تصادفی هش است.
- query_index: این آرگومان اندیس مربوط به داده کوئری از مجموعه‌داده را تعیین می‌کند.
- num_neighbours: برای هر داده به تعداد num_neighbours نزدیک‌ترین همسایه تقریبی در مجموعه‌داده برخواهد گشت.

خروجی این تابع لیستی از آیدی نزدیک‌ترین همسایه‌های تقریبی یک داده‌ی کوئری است که به ترتیب نزدیک بودن مرتب شده است و شامل خود داده کوئری نیست.

نهایتا نوبت به بررسی عملکرد خود تابع می‌رسد. ابتدا با داشتن آیدی داده کوئری و توابع هش که به عنوان ورودی گرفته شده‌اند لیست مقادیر هش داده کوئری بدست می‌آید (hashed_point). سپس با داشتن مقادیر هش داده کوئری و کل مقادیر هش مجموعه‌داده، لیستی از آیدی داده‌های نامزد نزدیک‌بودن به داده کوئری برگردانده می‌شود (candidate_row_nums). داده‌ای کاندیدا خواهد بود که اولا خود داده کوئری نباشد و ثانیا برای حداقل یک تابع هش، دارای مقدار هش یکسانی با داده‌ی کوئری باشد. برای این داده‌های نامزد فاصله واقعی با داده کوئری حساب می‌شود(distances) و بر اساس این فاصله مرتب‌سازی صورت می‌گیرد و نزدیک‌ترین آن‌ها نگهداری می‌شوند. (best_neighbours) نهایتا آیدی نزدیک‌ترین داده‌ها به عنوان خروجی تابع برگردانده می‌شود.

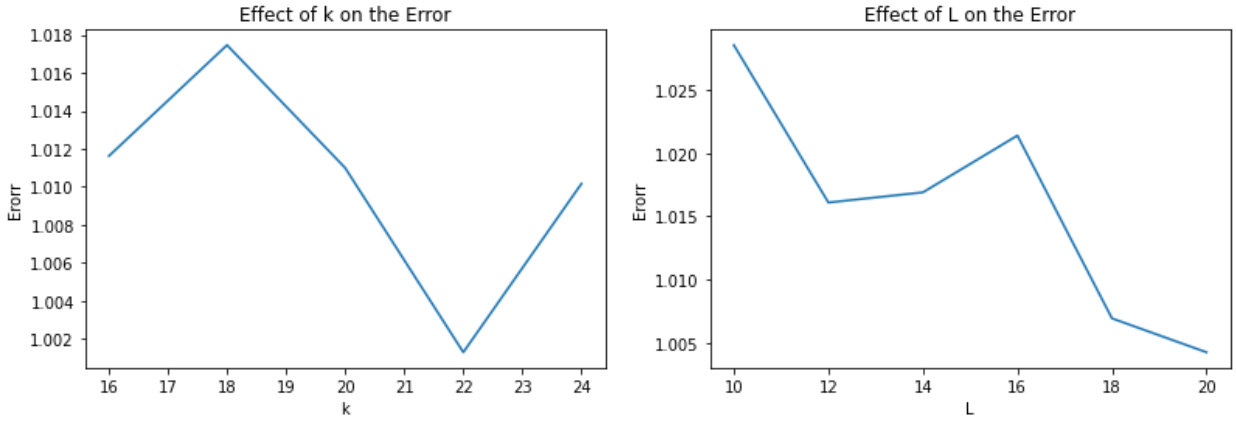
ب) الگوریتم LSH از دو قسمت تشکیل شده است: تنظیم و جستجو. فاز تنظیم برای هر مجموعه‌داده یک بار بیشتر اجرا نمی‌شود و شامل ایجاد توابع هش و هش کردن کل مجموعه‌داده است؛ اما فاز جستجو برای هر داده کوئری باید یک بار انجام گیرد. برای الگوریتم خطی هیچ گام پیش پردازشی لازم نیست. برای مقایسه دو الگوریتم سه عدد زیر را بدست آوردیم:

- زمان لازم برای تنظیم LSH حدودا ۱۶٫۹۸ ثانیه است.
- زمان جستجو LSH به طور میانگین برای هر داده حدودا ۱٫۳۲ ثانیه است.
- زمان جستجو خطی به طور میانگین برای هر داده حدودا ۳۴٫۹۹ ثانیه است.

به طور طبیعی از آنجایی که تنظیم LSH یک بار و جستجو با LSH بارها و بارها انجام می‌گیرد، زمان تنظیم LSH عملا سرشکن می‌شود و تاثیر زیادی نخواهد داشت. در این حالت می‌توان دید که الگوریتم LSH نسبت به الگوریتم خطی ۲۶٫۵ سریع‌تر است که یک عدد بسیار قابل توجه است و نشان می‌دهد از نظر زمانی الگوریتم LSH شدیدا کارآمد است. حتی اگر قصد داشته باشیم تنها یک بار جسجو انجام دهیم و به واسطه آن مجبور شویم تا برای همان یک بار، یک بار هم تنظیم انجام دهیم، الگوریتم LSH با زمان اجرای ۱۸٫۳ ثانیه تنها به حدود نیمی از زمان اجرای الگوریتم جستجوی خطی نیاز خواهد داشت که همچنان اثبات می‌کند که الگوریتم LSH از نظر زمان اجرا بسیار کارآمد است.

ج) مقدار خطای بدست آمده مطابق با فرمول درخواستی برابر با ۱٫۰۱ است. این نشان می‌دهد که روش LSH نه تنها یک جواب تقریبی را در زمان بسیار سریعی پیدا می‌کند بلکه این جواب تقریبی (حداقل برای این مجموعه‌داده، این پارامترها و این بار اجرا) بسیار به جواب نهایی نزدیک است و حدود ۱٪ خطا بیشتر ندارد.

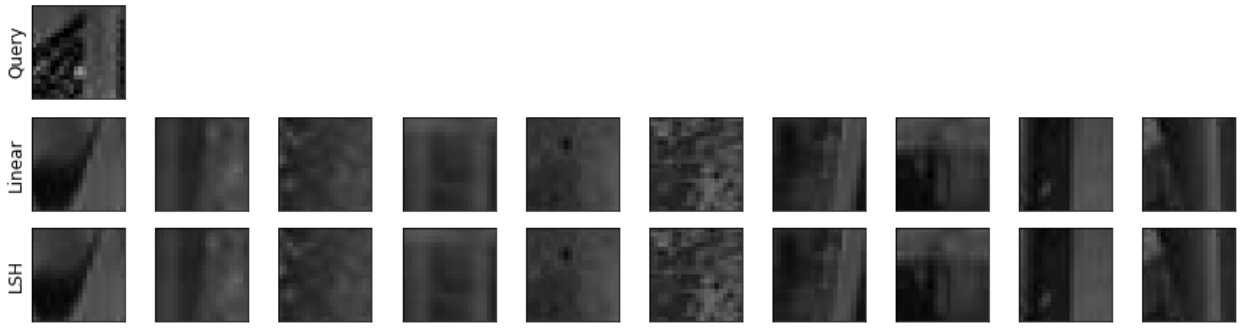
د) در دو نمودار زیر تغییرات مقدار خطا به ازای تغییر هر یک از دو پارامتر L و k آورده شده است:



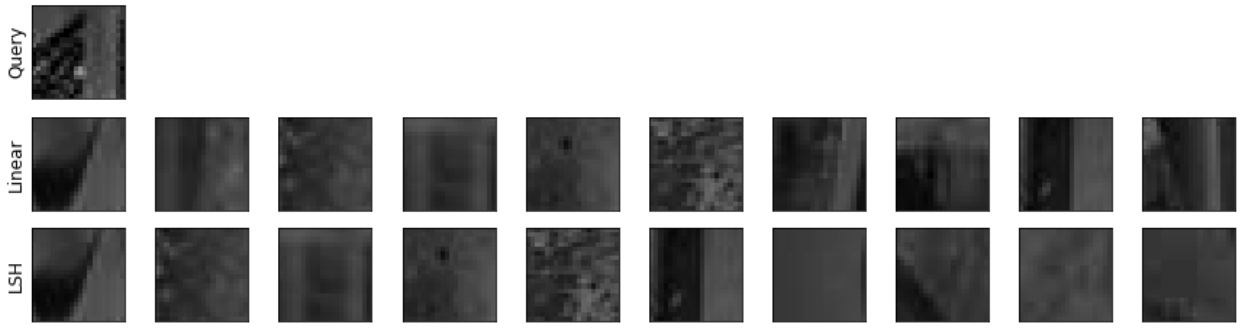
با بررسی این نمودارها در می‌یابیم که افزایش L به طور کلی می‌تواند باعث کاهش نسبی خطا شود. چنین چیزی از نظر تئوری هم اثبات شده است؛ L به تعداد توابع هش اشاره می‌کند و هر چقدر بیشتر باشد الگوریتم LSH کاندیدای بیشتری تولید می‌کند. کاندیدای بیشتر دقت مدل را افزایش و سرعت آن را کاهش می‌دهد که در این آزمایش سرعت مورد بررسی نبوده است.

در مورد k هم به نظر می‌رسد که در نتایج تجربی به طور کلی مقادیر بیشتر k خطای کمتری را به وجود آورده است. از نظر تئوری افزایش k باعث می‌شود که هش‌ها طول بیشتری داشته باشند و طبیعتا احتمال یکسان شدن هش دو الگوی متفاوت را کمتر می‌کند. پس زیاد شدن آن اگرچه باعث کاهش سرعت و مصرف بیشتر حافظه می‌شود ولی می‌تواند دقت را بهتر کند.

ه) تصویر کوئری و پیشنهادهای هر دو روش به ترتیب از چپ به راست در نمودار زیر آورده شده است. در این حالت به نظر می‌رسد که خروجی‌ها کاملا یکسان هستند.



برای اطمینان از صحت نتایج من این آزمایش را سه مرتبه در کل تکرار کردم که دو بار آن خروجی کاملا یکسان را نشان می‌داد که ارائه شده است و یک بار هم خروجی زیر بدست آمد:



اگر مبنا را خروجی یکسان در نظر بگیریم، LSH هم از نظر دقت در این حالت خاص به جواب بهینه می‌رسد و هم از نظر سرعت یک تسریع بسیار مطلوب را رقم می‌زند. اگر هم مبنا را خروجی نسبتا متفاوت درنظر بگیریم، باز خروجی LSH قابل قبول است؛ چراکه به نظر می‌رسد LSH دومین عکس نزدیک را از دست داده است و در میان عکس‌های آخر لیست هم تعدادی را از دست داده است. در این حالت برای مجموعه‌داده‌های خیلی کوچک شاید LSH گزینه خوبی نباشد ولی وقتی صحبت از کلان‌داده‌ها باشد عملا استفاده از روش خطی ممکن نیست و در این حالت هم LSH می‌تواند بهترین گزینه در مصالحه سرعت و دقت باشد.