تحلیل داده‌های حجیم

مدرس : دکتر ایمان غلامی

[پاییز ۱۴۰۰]

تمرن سری ۴: سوال ۴

نگارنده: امیرمحمد شعبانی

کاربرد

- برای فضای غیراقلیدسی استفاده می‌شود.
- طوری پیاده‌سازی شده‌است که فضای زیادی نگیرد و به عبارتی برای داده‌های حجیم ساخته شده‌است.
- ترکیبی از:

– point-assignment

– orginizes clusters hierarichically

تعاریف

$$ROWSUM_C(p)$$

جمع مربعات فاصله نقطه p از نقاط دیگر درون C.

درخت B-Tree

مثل اینکه این درخت در کتاب برای پیاده‌سازی این الگوریتم استفاده می‌شود که روش ساختش را در بخش دیگری از کتاب مفصل توضیح داده‌است. اما اگر بخواهیم خلاصه از آن برای درک راحت الگوریتم بگوییم این است که درختی است که برگ‌های $cluster$ها قرار دارند و پدرهای آن یک سری نمونه از نقاط وجود دارد و وقتی نقطه جدیدی وارد می‌شود از ریشه به دنبال این می‌گردد که سعی کنند به سمت فرزندان راسی برود که به آن نزدیک‌تر است. آنقدر اینکار را انجام می‌دهیم که برگ آن برسیم. ممکن است در یک برگ چند $cluster$ وجود داشته باشد.

مفهوم فاصله

درست است که در یک فضای غیراقلیدسی قرار داریم اما در این فضا نیز فاصله تعریف می‌شود و می‌توانیم فرض کنیم یک تابعی وجود دارد که به آن دو نقطه را می‌دهیم و در خروجی مقداری را می‌دهد که فاصله آن دو است.

زاویه قائمه در فضای بزرگ

مانند فضای اقلیدسی در اینجا هم می‌توانیم چنین فرضی را بکنیم که وقتی بعدمان بزرگ می‌شود هردو برداری برهم عمود هستند و از این فرض برای نوشتن یک سری روابط استفاده می‌کنیم.

k

یک مقدار ثابت در نظر بگیرید که در ادامه به این مقدار نیاز داریم.

ساخت یک cluster

هرکلاستر به یک سری ویژگی نیاز دارد که ذخیره کنیم. مخصوصا در داده‌های حجیم نیاز داریم به جای کل بخشی از داده‌ها را ذخیره کنیم و یا تابعی از داده‌ها را همیشه ذخیره و آپدیت کنیم.

N

تعداد نقاط یک کلاستر

clustroid

نقطه‌ای از کلاستر است که به عنوان مرکز آن شناخته می‌شود و $ROWSUM_C$ آن در کلاستر کمینه است.

k closest point

k نقطه که به کلاستروید نزدیک‌تر اند را با $ROWSUM_C$آن ذخیره می‌کنیم.

k furthest point

k نقطه که از کلاستروید دورتر اند را با $ROWSUM_C$آن ذخیره می‌کنیم.

ساخت درخت اولیه برای شروع الگوریتم

از آنجایی که حجم داده‌ها زیاد است، نمی‌توانیم تمام داده‌ها را روی رم بیاوریم و سپس کلاسترها را بسازیم؛ بنابراین مقداری را به عنوان نمونه روی رم میاوریم و درخت را می‌سازیم و سپس باقی نقاط را از حافظه می‌خوانیم و سعی می‌کنیم انها را به کلاسترها اضافه کنیم. اگرهم نیاز بود دو کلاستر را با هم مرج می‌کنیم و در شرایط خاصی نیز یک کلاستر را به دو کلاستر تبدیل می‌کنیم. همچنین ممکن است در حین اجرا حافظه رم پر شود آنگاه مجبوریم که دو کلاستر را باهم مرج کنیم. از آنجایی که به صورت درخت ذخیره کردیم می‌توانیم کلاسترهای در یک راس یا کلاسترهایی که پدر یکسانی دارند را باهم مرج کنیم. همچنین شاید بهتر باشد شرطی که برای دو قسمت کردن یک کلاستر داریم را کمی ریلکس کنیم. به طور مثال اگر ترشولدی نگه می‌داریم آن را تغییر دهیم.

اضافه کردن یک نقطه جدید

فرض کنید از ریشه درخت شروع به جست‌وجو کلاستر مورد نظر نقطه p جدید کردیم و به کلاستر رسیدیم. حال باید آن را وارد این کلاستر کنیم و ویژگی‌هایی که برای این کلاستر نگه می‌داشتیم را آپدیت کنیم.

N

$N \leftarrow N + 1$

k closest point & k furthest point

$ROWSUM_C(p_k) \leftarrow ROWSUM_C(p_k) + d^2(p, p_k)$

clustroid

$ROWSUM_C(c) \leftarrow ROWSUM_C(c) + d^2(p, c)$

$ROWSUM_C(p)$

با توجه به نکاتی که گفتیم به دلیل زیاد بودن بعد فضا داریم:

$$d^2(p,x) = d^2(p,c) + d^2(c,x)$$

بنابراین با جمع این عبارات روی تمام نقاط داریم:

$$ROWSUM_C(p) = N \times d^2(p,c) + ROWSUM_C(c)$$

حال از بین این نقاط کلاستروید را انتخاب می‌کنیم. شهودا دلیل اینکه k نقطه نزدیک کلاستروید گرفتیم این بود که احتمالا حرکت کلاستروید بین این‌ها خواهد بود و k نقطه دورتر گرفتیم چون شهودا اگر بخوایم دو کلاستر رو مرج کنیم با این‌ها سعی می‌کنیم تصمیم بگیریم. بعد از انتخاب کلاستروید k نقطه نزدیک و k نقطه دور از آن را انتخاب می‌کنیم و آن‌ها را آپدیت می‌کنیم.

دو قسمت کردن یک کلاستروید

زمانی به دو قسمت تقسیم می‌کنیم که به این نتیجه برسیم نقاط بسیار پراکنده هستند. یکی از این معیارهای این تصمیم $ROWSUM_C$ است. بنابراین ϵای تعریف می‌کنیم و می‌گوییم اگر $\sqrt{\frac{ROWSUM_C(c)}{N}} > \epsilon$ آن را به دو قسمت تقسیم می‌کنیم. برای اینکار تمام نقاط این کلاستر را به رم می‌آوریم و فرایند تقسیم را انجام می‌دهیم. باید طوری کلاسترها را بسازیم که $ROWSUM$ دو کلاستروید کمینه باشد.

ترکیب کردن دو کلاستر

شهودا می‌توانیم درک کنیم که وقتی دو کلاستر ترکیب شوند، کلاستروید جدید تا جای ممکن از دو کلاستروید دورتر است. اینجاست که نقاط دور ذخیره شده برای هر کلاستروید استفاده می‌شود. فرض کنید دو کلاستر C_1, C_2 را می‌خواهیم باهم ترکیب کنیم. حال برای آپدیت کردن $ROWSUM$ها مانند اضافه کردن نقطه عمل می‌کنیم. به طور مثال اگر نقطه p که از نقاط دور کلاستر C_1 است را آپدیت کنیم داریم:

$$ROWSUM_C(p) = ROWSUM_{C_1}(p) + N_{C_2}(d^2(p,c_1) + d^2(c_1,c_2)) + ROWSUM_{C_2}$$

پ.ن در انتها متوجه شدم هرجا که باید از سنتروید استفاده می‌کردم از کلاستروید استفاده کردم:) حلالم کنید.