



دانشکده مهندسی برق

مباحث ویژه در سیستم‌های دیجیتال

نیم‌سال اول ۱۴۰۰-۱۴۰۱

مدرس: دکتر ایمان غلام‌پور

توضیحات پروژه

شماره دانشجویی: ۹۷۱۰۱۰۲۶

نام و نام‌خانوادگی: امین کشیری

۱ مقدمه

در این پروژه، با استفاده از ایده‌ها و روش‌های مختلفی که در طول ترم آموختیم، اطلاعاتی را از داده‌ها بیرون کشیدیم. بخش‌های مختلف این پروژه را در فایل‌های jupyter جداگانه قرار داده‌ام، و توضیحات هر بخش را نیز جداگانه در این فایل نوشته‌ام.

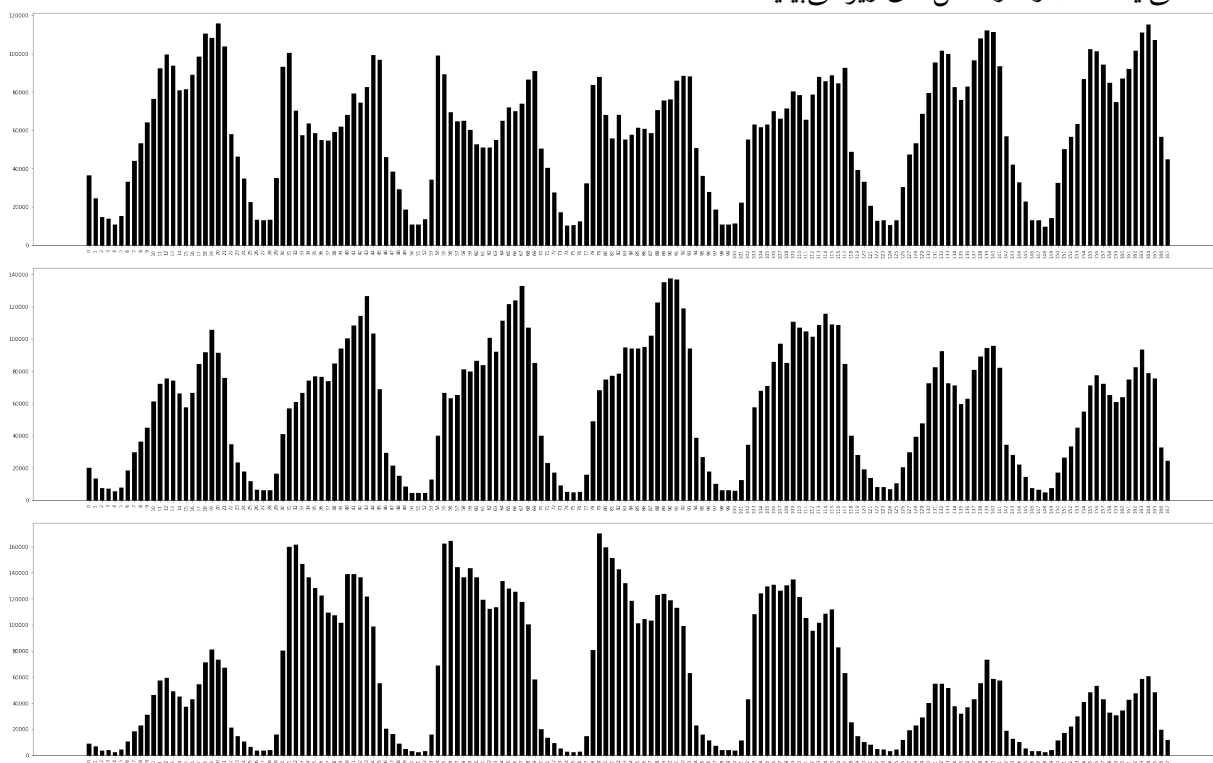
۱.۱ توضیحات کلی

۱. در ابتدای تمامی کدها تنظیمات اولیه اسپارک را انجام دادم، و سپس فایل csv داده شده را load کردم.
۲. در بعضی از بخش‌ها، روز ۸م را از داده‌ها حذف کردم. دلیل این کار این بود از تمامی روزها به اندازه‌ی متناسب با هم داده داشته باشیم. در غیر این صورت تعداد داده‌ها از روز سه شنبه دو برابر باقی روزها می‌شد. کارهای دیگری نیز می‌توانست انجام بگیرد. مثلاً می‌شود داده‌های روز سه شنبه را میانگین بگیریم (یعنی در تمام قسمت‌هایی که تعداد متغیری را شمرده‌ایم، برای روز سه شنبه این تعداد را تقسیم بر ۲ کنیم). در بعضی از قسمت‌ها اما این زیادتر بودن داده‌های روز سه شنبه مشکلی ایجاد نمی‌کرد. اما دقت کنید که در بعضی از قسمت‌های دیگر می‌تواند تحلیل ما را دچار انحراف کند (مثلاً ممکن است به اشتباه نتیجه بگیریم که روز سه شنبه روز پر تردد تری است، یا دوربین‌هایی که در روز سه شنبه دیده می‌شوند را به اشتباه مهم تر در نظر بگیریم).
۳. توضیحات کد و روند اجرا را در فایل‌های jupyter نوشته‌ام. سعی کرده‌ام که توضیحات منطق پشت کدها را در این مستند بنویسم (و نه در خود کدها). بنابراین توضیحات تکنیکال خود کد در اینجا کمتر نوشته شده است.

۳ Clustering

۱.۳ توضیحات کلی

با استفاده از خوشه سازی، سعی کردم دوربین ها را به دسته های مختلفی تقسیم کنم، و برای هر دسته مفهومی بیابم. نماینده هر دوربین در این روش، یک بردار با اندازه 24×7 است، که در هر خانه آن تعداد تردد در آن ساعت از روزهفته قرار گرفته است. ۲۴ ساعت اول برابر با یکشنبه است، ۲۴ ساعت بعدی برای دو شنبه و الی آخر. پس بردار متناظر هر دوربین، تعداد تردها در هر ساعت از یک هفته را برای آن دوربین مشخص می کند. برای خوشه سازی از الگوریتم LDA یا Latent Dirichlet Allocation استفاده شده است، که همانطوری که در درس دیدیم استفاده اولیه آن پیدا کردن توزیع topic های مختلف و کلمات آن ها برای هر مقاله است. با استفاده از این الگوریتم برای داده های تردد ماشین ها نیز می توانیم دقیقاً به چنین توزیعی برسیم. یکی از متغیرهای بسیار مهم در این بخش، cluster_center است، که تعداد کلاسترهای نهایی را مشخص می کند. با تغییر این متغیر این متغیر می توانیم تعابیر متفاوتی از داده داشته باشیم. اما یکی از واضح ترین نتیجه ها برای $cluster_center = 3$ به دست می آید که آن را در شکل های زیر می بینید:



۲.۳ تحلیل نتایج

مهم ترین نکته ی این سه تصویر، روند تغییر تردها در هر روز است. در دسته ی اول، تردد در ساعات اولیه روز افزایش می یابد، در هنگام ظهر کاهش پیدا می کند، سپس دوباره در شب افزایش پیدا می کند. در دسته ی دوم، تردد در صبح کم است، اما کم کم افزایش می یابد و در شب به اوج خود می رسد. دسته ی سوم روندی دقیقاً عکس دسته ی دوم دارد، و بیشترین تردد را در صبح دارند و سپس کاهش می یابد.

سه دسته ی بالا را می توانیم به این صورت تفسیر کنیم. دسته ی اول نقاط پر تردد شهر هستند، که هم در روز و هم در شب تردد بالایی دارند. این نقاط احتمالاً مکان هایی وسط شهر هستند که تمام طول روز تردد دارند (البته طبیعتاً تردد در ظهر کاهش می یابد). دسته ی دوم، احتمالاً مکان های دیدنی و تفریحی و یا بازارهای شبانه هستند که در طول روز تردد زیادی ندارند (به دلیل این که مردم مشغول کار و مدرسه و ... هستند). دسته ی سوم نیز احتمالاً مکان هایی هستند که در طول روز تردد بالایی دارند، مانند مکان های اداری، مسیر مدارس و ادارات، یا دوربین های نزدیک به مثلاً ثانوی ها.

یک نکته ی بسیار جالب دیگری که در این تصاویر دیده می شود، این است که تردد در روزهای جمعه، شنبه و یکشنبه به طرز جالبی پایین است. با چک کردن این ۳ روز روی تقویم، فهمیدم که این روزها تعطیل رسمی بوده اند (قیام ۱۵ خرداد و شهادت امام جعفر صادق (ع)). بسیار جالب است که این کاهش تردها، فقط در دسته ی سوم رخ داده است، که

دقیقا با شهود ما همخوانی دارد، که دسته‌ی سوم مکان‌هایی مانند مدارس و ادارات هستند. همچنین، الگوی کاهشی تردد در این سه روز از بین رفته است که باز هم مطابق با الگوی پیدا شده است.

۱.۴ توضیحات کلی

در این بخش، با استفاده از الگوریتم‌های خانواده‌ی Page Rank و در واقع با الگوریتمی شبیه به الگوریتم Pixie داده‌ها را تحلیل می‌کنیم. برای این کار، پس از تمیزکردن داده‌های اولیه، تنها ماشین‌های پرتدد را نگه می‌داریم. برای استفاده از الگوریتم، یک گراف دو بخشی در نظر می‌گیریم، که یک سمت آن ماشین‌ها قرار دارند و سمت دیگر آن دوربین‌ها. سپس بین هر ماشین، و هر دوربینی که آن ماشین از آن رد شده، یک یال در نظر می‌گیریم که وزن آن برابر با تعداد بارهایی است که دوربین آن ماشین را دیده است. سپس تمام یال‌ها را به صورت یک لیست مجاورت نگه می‌داریم (که به کمک آن می‌توانیم یال‌های خروجی از هر راس را به دست آوریم).

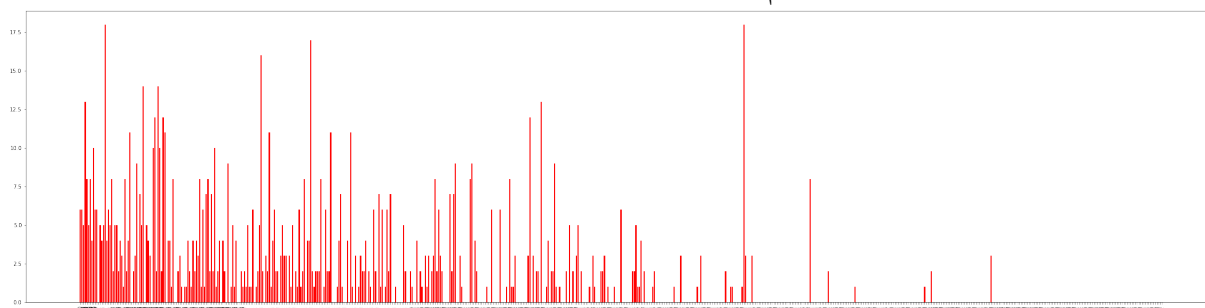
حال الگوریتم Pixie را پیاده‌سازی می‌کنیم. این الگوریتم هم می‌تواند از یک دوربین شروع شود، و هم از یک ماشین. بدون کاستن از کلیت تنها یک سمت آن را توضیح می‌دهیم، اما کد هر دو سمت در فایل jupyter موجود است.

با شروع از دوربین اولیه، لیست تمامی ماشین‌های دیده‌شده در آن دوربین را انتخاب می‌کنیم و به صورت تصادفی به یکی از ماشین‌ها می‌رویم. سپس بازهم لیست تمام دوربین‌هایی که آن ماشین در آن‌ها دیده شده را در نظر می‌گیریم و تصادفی به یکی از آن دوربین‌ها می‌رویم. برای هر دوربین یک counter نگه می‌داریم و هربار که از یک دوربین (روی گراف) عبور می‌کنیم، آن را یک واحد افزایش می‌دهیم. پس از طی شدن تعداد مرحله‌ای مشخص، لیست counter ها را خروجی می‌دهیم. چند نکته‌ی حائز اهمیت در این الگوریتم وجود دارد. اول آن که لیست راس‌های سمت مقابل، واقعا به صورت «لیست» نگه داشته شده است. یعنی ممکن است عضوی تکراری داشته باشد. این کار از عمد صورت گرفته است، تا در واقع شانس انتخاب راس‌هایی با یال با وزن بیشتری وصل شده‌اند بیشتر شود. در واقع انتخاب راس سمت مقابل با این که تصادفی است، اما از یک توزیع احتمال پیروی می‌کند که احتمال انتخاب یال‌های با وزن بالا را متناسب با وزن آن افزایش می‌دهد. نکته‌ی بعدی نیز این است که به احتمال ثابتی، هر بار به راس اولیه برمی‌گردیم که این احتمال در کد من برابر با 0.2 بوده است.

۲.۴ تحلیل نتایج

حال که راجب نحوه‌ی پیاده‌سازی و کلیات این ایده صحبت کردیم، می‌توانیم بیشتر در مورد کاربردهای آن صحبت کنیم. اولین کاربردی که این الگوریتم دارد این است که می‌تواند دوربین‌هایی که از نظر فیزیکی به هم نزدیکند را پیدا کند. زیرا اگر دو دوربین به هم نزدیک باشند، در صورتی که یک ماشین در یکی از آن‌ها دیده شود، به احتمال بیشتری در دیگری نیز دیده می‌شود. دقت کنید که عواملی مثل نوع دوربین در اینجا تاثیر گذارند. یعنی شباهت دوربین‌ها در حالت کلی تنها وابسته به فاصله‌ی فیزیکی نیست. در ادامه به این مورد بیشتر خواهیم پرداخت.

در حالت کلی، اگر یک دوربین به عنوان ورودی داده شود، می‌توانیم شبیه‌ترین دوربین‌ها به آن را با این روش به دست آوریم. برای مثال یک نمونه از خروجی الگوریتم به صورت زیر است:



همانطور که می‌بینید، به کمک این اطلاعات می‌توانیم دوربین‌های نزدیک را بیابیم. حال اگر یک گراف رسم کنیم، که هر راس آن یک دوربین باشد، و هر دوربین را به چند دوربین نزدیک خود وصل کنیم، می‌توانیم نقشه‌ای حدودی از شهر داشته باشیم (حتی اگر موقعیت جغرافیایی دوربین‌ها را نداشته باشیم). «چند» دوربین نزدیک نیز می‌تواند عددی بین ۲ تا ۶ باشد (برای نتایج منطقی‌تر). دقت کنید که با این روش، اگر تنها لیست مکان دوربین‌ها را داشته باشیم، و اسم دوربین‌ها کد گذاری شده باشد، احتمال دارد بتوانیم موقعیت دقیق هر کدام از دوربین‌ها را روی نقشه بیابیم! پس این روش در حالت کلی می‌تواند به ما دوربین‌های مجاور را بدهد اما کارهای دیگری نیز مانند آنچه گفته شد می‌توان انجام داد.

نکته‌ی مهمی که حائز اهمیت است، این است که «نزدیکی» دوربین‌ها با تحلیل من لزوماً نمی‌تواند به معنای نزدیکی فیزیکی باشد. زیرا اگر دو دوربین هر دو تخلفات یکسانی را ثبت کنند، احتمال بیشتری وجود دارد که ماشین‌های یکسان را ثبت کنند. یعنی «نزدیکی» در این تحلیل می‌تواند نزدیک بودن نوع دوربین‌ها باشد. یک مثال از این اتفاق را می‌توانید در کد من ببینید. در بخشی از کد، نزدیک‌ترین دوربین‌ها به دوربین کوثری خود را در نظر گرفتیم و نوع آن‌ها را چاپ کردم. نوع دوربین کوثری من ۸۱ بود، یعنی دوربین محدوده‌ی طرح ترافیک. و همانطوری که در نتایج دیده می‌شود، ۱۰ دوربین نزدیک همگی از نوع ۸۱ یا ۲۸۳ هستند که نوع ۲۸۳ نیز به معنی دوربین طرح زوج و فرد است (که در واقع یک دوربین محدوده طرح ترافیک است).

با استاده از مطلب بالا، می‌توانید ببینید که ما بدون دانستن اطلاعاتی مانند نوع دوربین، می‌توانیم دوربین‌های با نوع مشابه و نزدیک به هم را پیدا کنیم. یعنی اگر نوع دوربین‌ها به ما داده نشده بود احتمالاً می‌توانستیم دوربین‌ها را بر اساس شباهت‌هایی که از این الگوریتم به دست می‌آید دسته بندی کنیم.

نکته‌ی دیگری که تحلیل بالا مشخص می‌کند این است که اگر دوربین‌های نوع‌های مختلف را حذف کنیم و فقط آن‌هایی را نگه داریم که به صورت کلی ثبت می‌کنند (نه به دلیل اتفاق خاصی، مانند نوع ماشین یا یک تخلف خاص) آنگاه می‌توانیم ایده‌ای که در بالا مطرح شد را بهتر پیاده سازی کنیم، و دیگر دوربین‌های نزدیک واقعا به معنی دوربین‌هایی هستند که از نظر فیزیکی به هم نزدیکند و به این صورت می‌توانیم نقشه‌ی گراف شهر را بکشیم.

حال به الگوریتم از سمت دیگر نگاه کنید. این الگوریتم می‌تواند ماشین‌های «نزدیک» به هم را تشخیص دهد. دقیقاً مانند بالا، نزدیک بودن ماشین‌ها می‌تواند معیارهای معیارهای متفاوتی داشته باشد. برای مثال، اگر یک ماشین خاص را شناسایی کنیم که تعداد زیادی تخلف سرعت غیرمجاز دارد، با کمک این الگوریتم می‌توانیم ماشین‌هایی را بیابیم که الگوی حرکتی مشابهی با این ماشین دارند و در واقع ماشین‌هایی را بیابیم که رانندگی پرخطر دارند. تمام تحلیل‌های بالا را می‌توان برای این حالت نیز به کار گرفت و می‌توانیم ماشین‌هایی هم «نوع» را به دست بیاوریم.