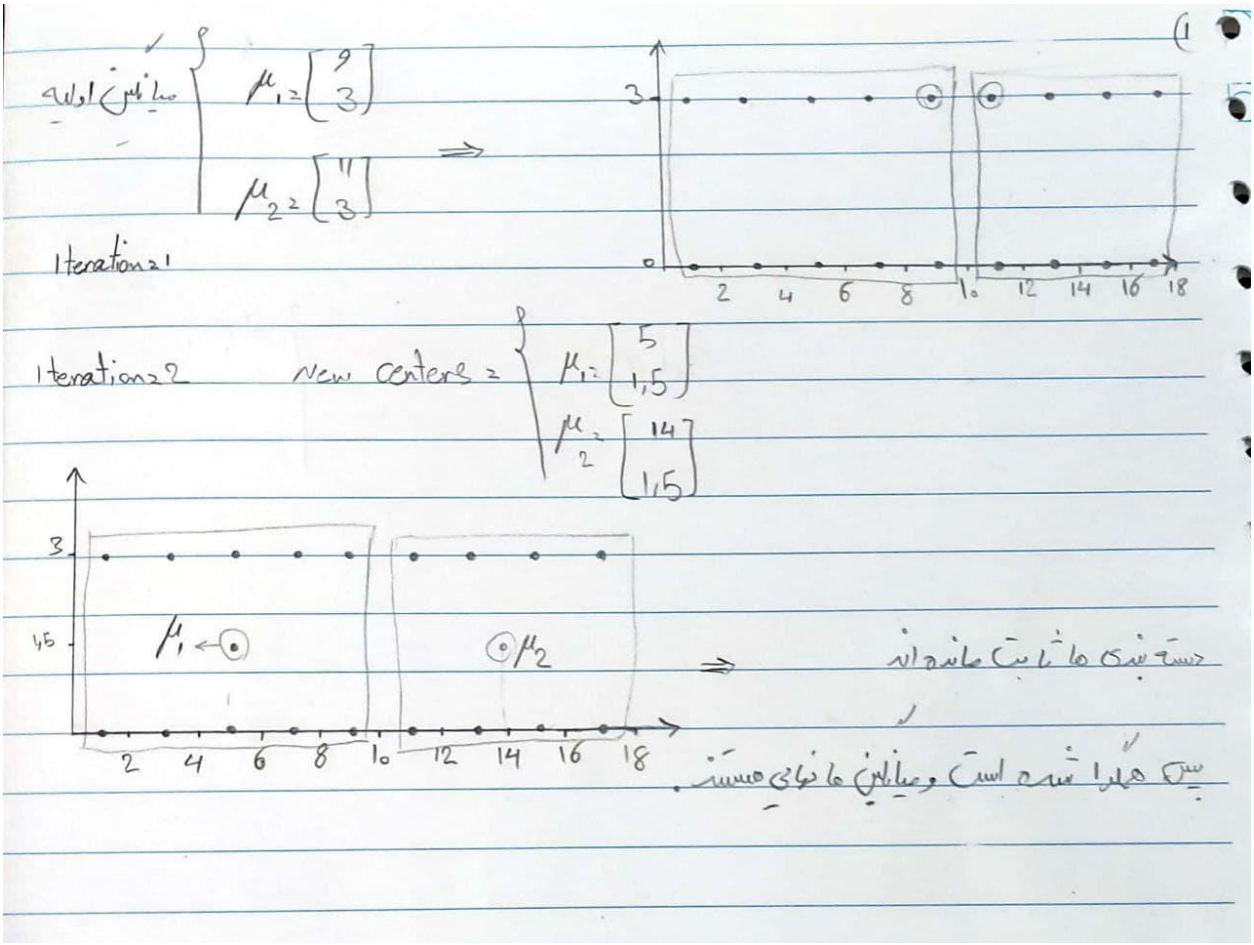
مهدی فیروزبخت

۴۰۰۱۳۱۰۲۷

تمرین سری چهارم - یادگیری ماشین

۱)



۲)

الف)

تقسیم بندی مشتری ، به معنی تقسیم بندی کردن مشتری ها بر اساس ویژگی های آنها به دسته های مجزایی است که به کمک این دسته بندی ها ، شرکت ها میتوانند خدمات مناسب تری به این مشتری ها بدهند.

در نظر بگیرید برای این دسته بندی ، مقادیر دسته بندی مشخصی به عنوان نماینده از هر دسته در اختیار داشته باشیم (منظور از نماینده همان میانگین هایی است که برای دسته بندی در kmeans استفاده میشود). در این صورت با بررسی اطلاعات مشتری ها و مقایسه آن ها با این نمایندگان و انجام الگوریتم ها خوشه بندی مانند kmeans یا kmedoids یا الگوریتم های دیگر میتوان مشتری ها را بر اساس درخواست های آنها خوشه بندی کرد.

ب)

برای این کار الگوریتمی با نام centroidQR وجود دارد. ابتدا به تعداد ابعاد مورد نظر کلاس تشکیل داده و میانگین های آن را پیدا میکنیم. سپس تمام ابعاد داده ها را به روی میانگین های انتخابی تصویر میکند که در این صورت تعداد ابعاد به تعداد کلاس ها کاهش میابد و ساختار کلاس اصلی را بهبود میبخشد. در این حالت جواب های کلاس بندی با تمام ابعاد و ابعاد کاهش یافته با یکدیگر یکسان خواهند بود.

ج)

برای پیدا کردن داده های پرت میتوان از الگوریتم مانند DBScan استفاده کرد. الگوریتم DBSCAN یک الگوریتم مبتنی بر چگالی است. این الگوریتم به چگالی نقاط داده در یک همسایگی نگاه می کند تا تصمیم بگیرد که آیا آنها به یک خوشه تعلق دارند یا خیر. اگر نقطه ای از همه نقاط دیگر خیلی دور باشد، نقطه پرت در نظر گرفته می شود و یک برچسب -۱ به آن اختصاص می یابد و به سادگی مشخص خواهد شد.

د)

برای image segmentation همانند تمرین پیاده سازی ، میتوانیم از الگوریتم هایی مانند kmeans استفاده کرد. در این الگوریتم با استفاده از میانگین هایی که از پیکسل های rgb پیدا میکنیم میتوانیم رنگ های نزدیک به یکدیگر را پیدا کرد. هر چه مقدار k بزرگتر باشد تعداد بیشتری رنگ قابل تجزیه بوده و بخش بندی تصویر میتواند عمیق و دقیق تر باشد.

۳)

الگوریتم dbscan دارای ۲ ابرپارامتر است. یکی پارامتر eps است که نشان دهنده فاصله شعاعی هر دایره است. دیگری پارامتر minPts است که حداقل تعداد داده های اطراف core را نشان میدهد.

برای پیدا کردن minPts قواعد مختلفی وجود دارد. در کل میتوان به این قواعد تکیه کرد : برای داده هایی که داده های نویز کمتری دارند میتوانیم از فرمول minPts >= D + 1 استفاده کرد که منظور از D ابعاد داده ها است.

اما در مورد داده هایی که مقدار نویز در آن ها بالاتر است و ابعاد آن زیاد است میتوان از قاعده minPts = 2 * D استفاده کرد.

در مورد پیدا کردن مقدار eps قاعده ای وجود ندارد اما میتوان از طریق روش elbow یک مقدار مناسبی برای آن پیدا کرد.

به این صورت که ابتدا مقدار minPts را مقداردهی میکنیم . سپس برای پیدا کردن داده های نزدیک که فاصله (eps) را زیاد میکنیم و در جایی که مقدار دادهای پیدا شده زیاد میشود ، یعنی در سمت آرنج مقدار eps مشخص است.

۴)

در الگوریتم های policy iteration ، شما با یک policy تصادفی شروع می کنید، سپس تابع ارزش آن policy (مرحله ارزیابی policy) را پیدا می کنید، سپس یک policy جدید (بهبود) بر اساس تابع مقدار قبلی و غیره پیدا می کنید. در این فرآیند، تضمین می شود که هر policy نسبت به policy قبلی بهبودی دقیق داشته باشد (مگر اینکه از قبل بهینه باشد). با توجه به یک policy ، تابع مقدار آن را می توان با استفاده از عملگر بلمن به دست آورد.

در value iteration ، شما با یک تابع مقدار تصادفی شروع می‌کنید و سپس یک تابع مقدار جدید (بهبود) را در یک فرآیند تکراری پیدا می‌کنید تا به تابع مقدار بهینه برسید. توجه داشته باشید که به راحتی می توانید policy بهینه را از تابع مقدار بهینه استخراج کنید. این فرآیند بر اساس بهینه عملگر بلمن است.

به طور کلی چند نکته کلیدی وجود دارد:

policy iteration شامل: policy evaluation + policy improvement و این دو به طور مکرر تکرار می شوند تا policy همگرا شوند.

value iteration شامل: یافتن تابع مقدار بهینه + یک سیاست استخراج است. هیچ تکراری از این دو وجود ندارد زیرا زمانی که تابع مقدار بهینه شد، سیاست خارج از آن نیز باید بهینه باشد (یعنی همگرا).

الگوریتم‌های ارزیابی policy و یافتن تابع مقدار بهینه بسیار مشابه هستند به جز برای یک عملیات حداکثر (همانطور که مشخص شد) به طور مشابه، گام کلیدی برای بهبود policy و استخراج policy یکسان است به جز اینکه اولی شامل بررسی پایداری است.

۵)

الف)

الگوریتم SingleLink : حداقل فاصله بین ۲ خوشه را در نظر میگیرد. به این صورت که فاصله تمام داده های درون ۲ خوشه را از یکدیگر محاسبه میکند و حداقل آن را در نظر میگیرد. سپس در بین تمام فواصل خوشه ها ، کمترین فاصله ها را با یکدیگر متصل میکند . این فاصله میتواند به صورت منهتن یا اقلیدسی باشد.

مرتبه زمانی این الگوریتم به صورت $O(n^2 \log n)$ است.

الگوریتم CompleteLink : در این نوع الگوریتم ، به جای حداقل فاصله بین ۲ خوشه ، از حداکثر فاصله استفاده میشود. به اینو صورت که فاصله داده های ۲ خوشه را از یکدیگر پیدا کرده و دورترین آن به عنوان فاصله ۲ خوشه در نظر گرفته میشود. سپس باقی مراحل مانند الگوریتم قبلی است.

مرتبه زمانی این الگوریتم به صورت $O(n^2 \log n)$ است.

الگوریتم AverageLink : در این نوع الگوریتم ، فاصله تمام داده های ۲ خوشه از یکدیگر را محاسبه کرده و با یکدیگر جمع میکنیم. سپس تقسیم بر تعداد داده های ۲ خوشه میکنیم. به نوعی میانگین فاصله ها را پیدا میکنیم. باقی مراحل مانند الگوریتم های قبلی است.

مرتبه زمانی این الگوریتم به صورت $O(n^2 \log n)$ است.

همانطوری که ملاحظه شد، از نظر مرتبه زمانی هر ۳ الگوریتم پیچیدگی زمانی یکسانی دارند. اما از نظر حساسیت به داده های پرت :

Average > Single > Complete

علت این حساسیت در این است که ، برای مثال در الگوریتم Single داده های نزدیک به یکدیگر در ابتدا به خوشه های یکسانی تبدیل میشود و داده هایی که نویز هستند و دور تر از باقی داده ها هستند در آخر کار به داده های اصلی متصل میشوند که نشان دهنده این هست که این داده ها نویز هستند که در انتهای مراحل الگوریتم به داده های اصلی متصل میشوند.

ب)

با استفاده از الگوریتم های :

SingleLinkage : (1 , 2) , (3 , 4)

CompleteLinkage : (2 , 3) , (1 , 4)

AverageLinkage : (1 , 3) , (2 , 4)

ج)

SingleLinkage :

این الگوریتم برای تصویر b کاملا میتواند به صورت درست این خوشه بندی را انجام دهد. به این صورت که داده های نزدیک با یکدیگ تشکیل خوشه های بزرگتر را میدهد و چون همواره فاصله این داده ها از فاصله ۲ هلال کمتر است پس هیچوقت خوشه های این ۲ هلال ترکیب نمیشوند و به راحتی از یکدیگر تفکیک میشوند.

در مورد شکل c ، همانند تصویر b خوشه بندی صورت میگیرد ، در مورد اتصالات هلال ها با یکدیگر زیرا تعداد داده های اتصال تعداد کمی هستند و نسبت به ادامه هلال نزدیکی کمتری دارد هر هلال به صورت جداگانه خوشه بندی شده و اتصال سمت چپ برای داده های پایینی خوشه بندی شده و اتصال راست ، به داده های بالایی مربوط میشود.

: CompleteLinkage

این الگوریتم برای تصویر b ابتدا داده های اولیه با یکدیگر تشکیل خوشه بندی های کوچک در ۲ هلال را میدهند. با افزایش تعداد داده های هر خوشه برای محاسبه فاصله دورترین ، خوشه بندی ها به هم خورده و قسمتی از هلال بالایی و قسمتی از هلال پایینی به خوشه های متضاد خود نسبت داده میشود.

به این صورت که فاصله دورترین نقطه خوشه تا داده جدید از هلال پایینی بیشتر از داده از هلال بالا میشود پس الگوریتم از داده های هلال متضاد خود استفاده کرده و خوشه بندی برهم میخورد.

در مورد شکل c نیز این شکل همانند شکل b خواهد بود . اتصال سمت چپ برای داده های بالا و اتصال سمت راست برای داده های پایین خواهد بود.

: AverageLinkage

این الگوریتم برای تصویر b ابتدا داده های اولیه با یکدیگر تشکیل خوشه بندی های کوچک در ۲ هلال را میدهند. با افزایش تعداد داده های هر خوشه برای محاسبه فاصله میانگین ، خوشه بندی ها به هم خورده و قسمتی از هلال پایینی به خوشه بالا نسبت داده میشود.

در این صورت خوشه بندی برای هلال بالا ، تمام هلال بال و قسمی از هلال پایین را شامل میشود.

در مورد شکل c ، هر ۲ الگوریتم تا اتصالات با یکدیگر یکسان خواهند بود سپس از طریق اتصالات وارد هلال مجاور شده و همچنین در هلال خود نیز به جلو میروند. در انتها ، ۲ خوشه بندی شامل هر ۲ هلال شده و با خطی مانند $x = y$ ۲ خوشه از یکدیگر جدا شده اند.

با بررسی هر ۳ معیار به این نتیجه رسیده ایم که الگوریم SingleLinkage از ۲ الگوریتم دیگر بهتر کار کرده و برای این مسئله این معیار مناسب تر است.

پیاده سازی:

۱)

فایل پیاده سازی این سوال به نام Question 1 قرار داده شده است.

در ابتدا ، کتابخانه های مورد نیاز را در این پروژه وارد کرده ایم. با استفاده از تابع load_images عکس های درون یک پوشه را میتوانیم بخوانیم. سپس داده های درون فولد ذکر شده را خوانده و در متغیرها قرار میدهیم.

در تابع kmeanscluster ، الگوریتم kmeans را پیاده سازی کرده ایم. ابتدا در این تابع داده ها ، میانگین ها ، k و تعداد تکرار را به عنوان ورودی دریافت میکنیم و تا زمین رسیدن به این مقدار تکرار الگوریتم را تکرار میکنیم. در این الگوریتم ابتدا فاصله هر داده از هر میانگین را پیدا میکنیم و با argmin کمترین فاصله را برای هر داده پیدا کرده و بر اساس میانگین آن لیبلی به آن میدهیم. سپس داده های مربوط به هر لیبل را جمع کرده و میانگین این داده ها را پیدا میکنیم و به جای میانگین ورودی قرار میدهیم و این الگوریتم را ادامه میدهیم تا به مقدار نهایی همگرا شود. در ادامه برای قسمت دوم سوال که خواسته شده براساس میانگین ها رنگ ها را تغییر دهیم ، یک لیست ساخته و بررسی میکنیم که هر پیکسل RGB چه لیبلی دارد و به جای آن مقدار میانگین مربوط به آن را قرار میدهیم.

در تابع findMu ، بر اساس تعداد k ، میانگین های رندوم ایجاد میکنیم. در تابع clustering ، ابتدا میانگین ها را از تابع findMu دریافت کرده ، سپس داده ها را وارد kmeanscluster کرده و خوشه بندی انجام شده ، تصاویر ساخته شده و بازگشت داده میشود. سپس با استفاده از plot به صورت pie درصد هر رنگ را در هر تصویر نمایش میدهیم. در انتها برای هر k و هر تصویر تابع clustering را صدا زده و تمام مراحل بالا را برای آن انجام میدهیم.

۲)

فایل پیاده سازی این سوال به نام Question 2 قرار داده شده است.

در ابتدا ، کتابخانه های مورد نیاز را در این فایل فراخوانی کرده ایم. سپس در تابع purity_score با استفاده از کتابخانه metrics مقدار purity را محاسبه میکنیم.

در تابع neighbor_points ، با استفاده از فاصله اقلیدسی ، فاصله داده های نزدیک به داده مورد نظر که کمتر از radius را پیدا میکنیم.

در تابع dbscan ، ابتدا در ورودی داده ها ، eps و minPts را دریافت میکنیم. سپس با استفاده از تابع neighbor_points تمام همسایه های هر داده را پیدا میکنیم. سپس از بین این داده ها corepoint و noncore ها را پیدا میکنیم. سپس با بررسی همسایه های هر core و بررسی اینکه همسایه های آن نیز core هستند یا خیر داده ها را خوشه بندی میکنیم و تمام داده ها را بررسی میکنیم. در انتها لیبل داده ها و تعداد خوشه ها را بازگشت میدهیم.

در تابع plotRes در ورودی داده ها ، لیبل هر داده و تعداد خوشه ها را به ورودی میدیم. سپس با استفاده از لیبل داده ها و خود داده ها با رنگ های یکسان داده ها را نمایش میدهیم.

سپس این تابع ها را فراخوانی میکنیم. ابتدا داده ها را میخوانیم . برای هر دیتاست یه بلاک متفاوت ایجاد کرده ایم. به صورت دستی چندین eps و minPts را تست کرده ایم تا به بیشترین مقدار purity برسیم.

تعداد کلاستر ها ، تعداد نویز ها و مقدار purity برای هر دیتاست در فایل پیاده سازی قرار داده شده است.

در مورد بررسی تاثیر اشکال و نوع توزیع داده ها :

در این الگوریتم هر چه داده ها به شکل منظم تری مانند شکل rings و spiral باشند الگوریتم با درصد بیشتری میتواند خوشه بندی را انجام دهد. همانطوری که در پیاده سازی مشاهده میکنید ، برای این ۲ شکل الگوریتم توانسته با ۱۰۰٪ تخمین های درستی ایجاد کند. اما در مورد توزیعی مانند D31 که داده ها خیلی بی نظم هستند و پیچیدگی زیادی دارند درصد پایین تر خواهد بود.

در کل با بررسی داده ها و اشکال و الگوریتم DBSCAN میتوان فهمید که هر چه شکل داده توزیع منظم تری داشته باشد احتمال اینکه الگوریتم بتواند تخمین بهتری بزند بیشتر است. هر چه توزیع داده بی نظم تر باشد و داده ها جداپذیری کمتری داشته باشند تخمین اشتباه بیشتری دارد و به خوبی تخمینی ایجاد نمیکند.

۳)

فایل پیاده سازی این سوال به نام Question 3 قرار داده شده است.

در این فایل ابتدا کتابخانه های مورد نظر را وارد میکنیم. سپس داده ها را فراخوانی میکنیم. داده ها به صورت جداگانه دانلود شده و به روی نرم افزار فراخوانی شده است. همچنین بر اساس اطلاعات سوال ، متغیر y ایجاد شده است که نشان دهنده لیبل هر داده است.

سپس داده ها را به حالتی که در سوال بیان شده آورده میشود.(هر تصویر به یک وکتور ۱ بعدی با اندازه ٤٠٩٦ مسطح شده است.) هچنین در متن سوال اورده شده است که داده ها را باید به صورت طبقه ای به ۳ بخش تقسیم کنیم. ابتدا داده ها را به صورت طبقه ای به ۲ حالت , train rem تبدیل کرده و سپس داده rem را به ۲ داده test , valid تقسیم کرده ایم.

برای پیدا کردن elbow و k مناسب از کتابخانه KElbowVisualizer استفاده کرده ایم. در این کتابخانه داده های validation را وارد کرده ایم و بر اساس kmeans ، k مربوطه را بازگردانده است و نمودار آن را نیز رسم کرده است.

سپس با استفاده از کتابخانه kmeans و داده های آموزش و مقدار k به دست آمده ، یک الگوریتم kmeans آموزش داده ایم. برای تست کردن این این الگوریتم خوشه بندی ، میانگین های نهایی را از kmeans.Centers دریافت کرده ایم و سپس فاصله هر داده از این میانگین ها را پیدا کرده و در نظر گرفته ایم که هر داده به کدام میانگین نزدیک تر است پس این تصویر مربوط به این کلاستر میشود.

در انتها همانطوری که در سوال خواسته شده است ، هر خوشه را برای داده های تست و آموزش به صورت مجزا نمایش داده ایم .

الف)

نمونه گیری طبقه ای ، نوعی از نمونه گیری است که در آن بر اساس ویژگی هدف ، داده ها را به اندازه های مساوی تقسیم میکند. برای مثال برای همین سوال ۴۰۰ داده داریم که از ۴۰ نفر از هر نفر ۱۰ عکس در اختیار داریم. اگر به نسبت ۷ داده ها را نمونه گیری کنیم ، داده ها به گونه ای نمونه گیری میشوند که از هر کلاس تعداد داده مساوی باشد. در این مثال ۴۰ کلاس داریم که ۱۰ داده برای هر کلاس است. با نسبت ۷ ، از هر کلاس ۷ داده جدا میکند و در این صورت داده های هر کلاس به مقدار مساوی در آموزش یا تست یا اعتبارسنجی تاثیر گذار خواهد بود.

ب)

پیاده سازی شده است.

ج)

خوشه ها در فایل پیاده سازی نمایش داده شده است. با بررسی این داده ها میتوانیم ببینیم که نسبتا این تصاویر شباهت نزدیکی دارند. به خصوص از نظر زاویه گرفتن تصویر ، رنگ و روشنایی تصویر ، نزدیکی دوربین به تصویر و ویژگی های عکاسی بسیار شباهت دارند.

در مورد بعضی از دسته بندی ها نیز دقیقا داده ها درست خوشه بندی شده اند و تمام تصاویر مربوط به یک فرد است.