به نام پروردگار بخشنده و مهربان

درس یادگیری ماشین و بازشناسی الگو - تمرین شماره ۱

امین عباخواه درآباد **۴۰۲۱۱۵۰۶۴** **amin.abakhah@aut.ac.ir**

پاسخ سوال ۱:

الف) در تخصیص یک کلاس به یک داده تست بر اساس الگوریتم k-NN ، زمانیکه تمام k همسایه از یک کلاس باشند، داده تست در همان کلاس قرار خواهد گرفت اما زمانیکه برخی از k همسایه، در یک کلاس و برخی دیگر در کلاس یا کلاس‌های دیگری باشند، در اینجا باید با استفاده از روش‌هایی، بین کلاس‌های مختلف، voting انجام بگیرد تا مشخص شود که داده تست را باید در کدام کلاس قرار دهیم.[۱]

ب) از همبستگی spearman زمانی استفاده می‌شود که ویژگی‌ها، توزیع نرمالی ندارند.[۲]

پ) یکی از ضعف‌های اصلی معیار لایبلر- کول بک این است که معیار نیست! زیرا شرط تقارن را برآورده نمی‌کند. یک ضعف بزرگتر آن، این است که این معیار به دلیل وجود مرزهای binها مستعد خطاست. فاصله بین یک تصویر و یک نسخه کمی تیره‌تر از خودش می‌تواند زیاد باشد، اگر پیکسل‌ها در یک bin مجاور بیفتند، زیرا در این معیار، مجاورت binها در نظر گرفته نمی‌شود.[۳]

ت) از آنجایی که تمام کارها در زمان اجرا انجام می‌شود، در صورتی که مجموعه آموزشی بزرگ باشد، k-NN می‌تواند عملکرد زمان اجراء ضعیفی داشته باشد. همچنین k-NN به ویژگی‌های نامربوط یا اضافی بسیار حساس است اما این مشکل را می‌توان با انتخاب دقیق ویژگی یا وزن‌دهی ویژگی بهبود بخشید. در نهایت، در کارهای طبقه‌بندی بسیار دشوار، k-NN ممکن است نسبت به تکنیک‌های دیگر مانند ماشین‌های بردار پشتیبان یا شبکه‌های عصبی ضعیف‌تر عمل کند.[۴]

ث) از هر دو روش برای کاهش پیچیدگی محاسباتی k-NN استفاده می‌شود. در اینجا ما روش Kd-Tree را توضیح می‌دهیم، سپس برخی مزایا و معایب هر دو را بیان می‌کنیم. روش Kd-Tree فضای مجموعه داده‌ها را بصورت یک درخت باینری بر اساس انتخاب پشت سرهم ویژگی‌ها ، بصورت عمود بر محورها، پارتیشن‌بندی می‌کند. بهتر است که ویژگی‌ها به ترتیبی انتخاب شوند که آنهایی که بیشترین واریانس را دارند، در سطح بالاتری از درخت قرار بگیرند. این باعث می‌شود که فضای جستجو کوچکتر شده، در نتیجه زمان جستجو نیز کاهش یابد؛ هر چند که اگر تعداد همسایه‌ها یعنی K در هنگام محاسبه زیاد باشد، سرعت آن از جستجوی brute force هم بیشتر خواهد شد. مشکل دیگر این روش نفرین ابعاد است یعنی اگر تعداد ویژگی‌ها زیاد باشد، پیچیدگی محاسباتی این روش از KNN عادی بهتر نخواهد بود.

---

[۱] ر.ک. صفحه ۳ مقاله

[۲] ر.ک صفحه ۵ مقاله

[۳] ر.ک. صفحه ۶ مقاله

[۴] ر.ک. صفحه ۲۲ مقاله

در حالیکه روش فوق، داده‌ها را بر اساس ویژگی‌شان، پارتیشن‌بندی می‌کند، روش Ball-Tree یک روش بر اساس متریک است که داده‌ها را بر اساس فواصل جفت نمونه‌های آن، دسته‌بندی می‌کند. در مقایسه با Kd-Tree، روش Ball-Tree پتانسیل عملکرد بهتری برای داده‌های با ابعاد بالا دارد، به عنوان مثال، در تجزیه و تحلیل تصویر.[5]

پاسخ سوال ۲:

الف و ب و پ و ت و ث)

در این قسمتها، بردار ویژگی را صرفا ۳۲ حرف الفبای فارسی در نظر می‌گیریم. در قسمت (ج)، برخی حروف عربی را نیز به بردار ویژگی اضافه می‌کنیم و مجددا محاسبات را انجام خواهیم داد. محاسبات بوسیله زبان برنامه‌نویسی پایتون و در محیط Google Colab انجام شده است. در اینجا صرفا نتایج محاسبات را در جدول آورده‌ایم. کدها نیز لینک زیر[6] در GitHub قابل دسترسی هستند.

Prediction Error (%) K = 5			Prediction Error (%) K = 3			Prediction Error (%) K = 1			
Correlation	Cosine	Euclidean	Correlation	Cosine	Euclidean	Correlation	Cosine	Euclidean	ویژگی / معیار فاصله
0	10	20	10	10	20	20	10	20	BoW دودویی
0	0	0	0	0	10	0	0	0	BoW وزندار
0	0	0	0	0	0	0	0	0	BoW نرمال شده طول
0	0	0	0	0	0	0	0	0	BoW نرمال شده نمره-زد

[5] ر.ک صفحه ۱۰ و ۱۱ مقاله

[6] https://github.com/my7amin/MLPR/tree/master/Exercise-1

ج)

در این قسمت می‌خواهیم که چند حرف عربی نیز به بردار ویژگی اضافه کنیم که در اینصورت طول بردار ویژگی ۴۰ حرف خواهد شد و مجددا خطا را با روش‌های قبلی محاسبه کنیم.

Prediction Error (%) K = 5			Prediction Error (%) K = 3			Prediction Error (%) K = 1			
Correlation	Cosine	Euclidean	Correlation	Cosine	Euclidean	Correlation	Cosine	Euclidean	ویژگی / معیار فاصله
0	0	0	0	0	0	0	0	0	BoW دودویی
0	0	0	0	0	0	0	0	0	BoW وزندار
0	0	0	0	0	0	0	0	0	BoW نرمال شده طول
0	0	0	0	0	0	0	0	0	BoW نرمال شده نمره-زد

در اینجا می‌بینیم که تمام مقادیر خطاها صفر شد.

بار دیگر می‌خواهیم، صرفا از برخی حروف فارسی و برخی حروف عربی استفاده کنیم و با اینکار طول بردار ویژگی را به صرفا ۱۳ حرف کاهش دهیم و مجددا محاسبات را انجام داده و مقادیر خطاها را در جدول وارد کنیم.

Prediction Error (%) K = 5			Prediction Error (%) K = 3			Prediction Error (%) K = 1			
Correlation	Cosine	Euclidean	Correlation	Cosine	Euclidean	Correlation	Cosine	Euclidean	ویژگی / معیار فاصله
0	0	0	0	0	0	0	0	0	BoW دودویی
0	0	0	0	0	0	0	0	0	BoW وزندار
0	0	0	0	0	0	0	0	0	BoW نرمال شده طول
0	0	0	0	0	0	0	0	0	BoW نرمال شده نمره-زد

این بار نیز تمام مقادیر خطاها صفر شد با این تفاوت که طول بردار ویژگی از ۴۰ حرف به ۱۳ حرف کاهش یافت. نتیجه می‌گیریم که با انتخاب هوشمندانه بردار ویژگی می‌توانیم سربار محاسباتی را کاهش و سرعت اجرا را بالاتر ببریم.

حال می‌خواهیم از روش TF-IDF برای ساخت بردار ویژگی استفاده کنیم. روش TF-IDF روشی است که اهمیت یک حرف یا کاراکتر را در کل مجموعه داده نیز در نظر می گیرد یعنی وزن حروف رایج را کاهش و وزن حروف کمیاب را در هر جمله افزایش می دهد (البته در اینجا چون ویژگی‌های ما حروف بودند، ما در تعریف TF-IDF از حروف استفاده کردیم اما اگر ویژگی‌ها کلمات بودند، آنوقت وزن کلمات را به نسبت وفور آنها در کل داده در نظر می گرفت). همچنین این بار صرفا از معیارهای فاصله Manhattan ، Chebyshev و Jaccard در knn استفاده خواهیم کرد و نتایج را در جدول نمایش خواهیم داد.

Prediction Error (%) K = 5			Prediction Error (%) K = 3			Prediction Error (%) K = 1			
Jaccard	Chebyshev	Manhattan	Jaccard	Chebyshev	Manhattan	Jaccard	Chebyshev	Manhattan	ویژگی / معیار فاصله
0	0	0	0	0	0	0	0	0	BoW دودویی
0	0	0	0	0	0	0	0	0	BoW وزندار
0	0	0	0	0	0	0	0	0	BoW نرمال شده طول
50	0	0	50	0	0	50	0	0	BoW نرمال شده نمره-زد
0	0	0	0	0	0	0	0	0	TF-IDF

علت اینکه اکثر خطاها در پیش‌بینی‌های انجام شده صفر شده‌اند، استفاده از ویژگی‌های مناسب برای بردار ویژگی است. در اینجا نیز ویژگی‌های استفاده شده در بردار ویژگی همانند قسمت قبلی است که شامل ۱۳ حرف فارسی و عربی است. در واقع اگر ویژگی‌ها را هوشمندانه و متناسب با داده انتخاب کنیم، بخش زیادی از خطا را در همان ابتدای کار کاهش داده‌ایم.