مهدی فیروزبخت - ۴۰۰۱۳۱۰۲۷

تمرین 3

یادگیری ماشین

۱)

الف) غلط – زیرا در svm پارامتری که بیان کننده توزیع داده ها باشد را نداریم پس یک الگوریتم غیر پارامتری است.

ب) صحیح – در svm با مقدار C که regularization parameter است میتوان از بیش برازش دوری کرد و این الگوریتم برای بیش برازش less prone است.

پ) غلط – جریمه طبقه بندی نادرست با یک convex loss به نام hinge loss تعریف می شود و نامحدود بودن تلفات محدب باعث حساسیت به موارد پرت می شود.

ت) صحیح – در صورتی که محدودیتی در تعداد دسته بند های ضعیف نداشته باشیم میتوانیم به مقدار خطای ۰ برسیم.

ث) غلط – مقدار وزن همواره منفی نیست و به میزان Error rate وابسته است. اگر مقدار Error rate از ۰/۵ بیشتر باشد ، مقدار وزن منفی میشود.

ج) صحیح – زیرا داده های نویزی مقدار باقی مانده بیشتری نسبت به داده های غیر نویزی دارند و وزن بیشتری به خود اختصاص میدهند پس Gradient Boosting توجه بیشتر و نامناسبی به این نقاط خواهند داشت.

۲) الف) بله، زیرا با خطی می‌توان آن‌ها را از یکدیگر جدا کرد.

ب) با استفاده از شکل، ۲ نقطه در خط جدا کننده برابر:

$(1, 0.5)$ و $(0.5, 1.5)$ هستند، پس:

$$m = \frac{0-1}{1.5-0.5} = -1 \qquad y - 0 = -1(x - 1.5) \Rightarrow \boxed{y = -x + 1.5}$$

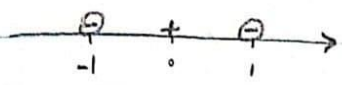
$$b = 1.5, \; w = -1 \Rightarrow$$

در این سوال، خط مارجین برای کلاس مثبت برابر $y = -x + 2$ و برای کلاس منفی برابر $y = -x + 1$ است.

ج) با توجه به داده‌ها خط‌های مارجین برای کلاس مثبت و منفی، در کلاس مثبت ۲ داده بر روی خط و در کلاس منفی نیز ۲ داده بر روی خط است، پس در این سوال می‌توان از خط‌های پشتیبان حذف شود و تنها در فاصله مارجین ایجاد می‌کنند.

د) صحیح است، زیرا در حالت کلی با استفاده از این بردار پشتیبان سختی می‌توان دورترین فاصله حاشیه را یافت، حالا اگر بردار پشتیبان حذف شود بردار پشتیبان جدید ۲ حالت دارد، یا روی همان بردار بر روی خط حاشیه حالت برون حذف است، پس مقدار حاشیه با مقدار قبل یکسان است. اگر داده روی خط حاشیه قبلی باشد، حتماً در فاصله دورتری است زیرا در الگوریتم SVM فاصله حاشیه بیشینه حداکثر است، پس مقدار حاشیه کمتر یا بیشتر می‌شود. همچنین می‌توان گفت: $width = \frac{2}{\|w\|}$ و $w = \sum a_i y_i x_i$. اگر a حذف شود مقدار w یا صعودی حالت قبلی می‌شود یا کاهش می‌یابد، در صورت کاهش مقدار $width$ افزایش می‌یابد.

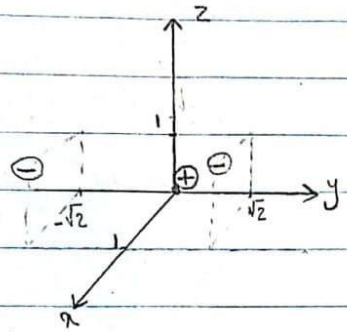
۲) الف) خیر، زیرا داده ⊕ بین داده‌های ⊖ است.



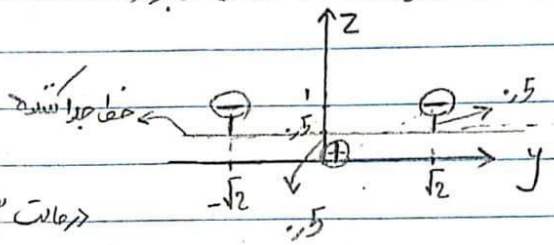
ب) با بردن داده‌ها به فضای جدید: برای داده مثبت معادلات: $(0, 0, 1)$ و برای داده‌های منفی: $(1, \sqrt{2}, 1)$ و $(1, -\sqrt{2}, 1)$ خواهد بود که با رسم آن:

$\Phi(x) = [x^2, \sqrt{2}x, 1]$

x y z



برای ساده سازی می‌توان داده را به فضای دوبعدی آورد:



در حالت ۲ بعدی می‌توان خط با معادله $z = 0.5$ رسم کرد.

فاصله داده مثبت و منفی از این خط برابر با 0.5 و معادله خط حاشیه مثبت برابر $z = 0$ و برای کلاس منفی برابر $z = 1$ است

پس معادله صفحه برابر است با: $\text{معادله صفحه} = \begin{cases} x = 1 \\ z = 0.5 \end{cases}$

(۴

ع) الف) مربوط به شکل 4 است زیرا حالت Soft است یعنی اجازه خطا می دهیم پس مربوط به این شکل است.

ب) مربوط به شکل 3 به دلیل خطی بودن Soft بودن است و چون C افزایش یافته باعث جابه جایی خط شده به شکل 4 شده است.

ج) مربوط به شکل 2 زیرا حالت خمیدگی دارد. gamma کمتر، فاصله دورتر

د) مربوط به شکل 1 است زیرا کرنل Rbf با gamma برابر $\frac{1}{4}$ دارد، Rbf حالت طبیعی دارد.

ه) مربوط به شکل 3 است زیرا کرنل Rbf با gamma برابر 4 دارد، Rbf حالت طبیعی دارد.

(۵

و) ← حالت ۱ فقط یک داده از کلاس ⊕ اشتباه دسته بندی شده است پس:

$$a^1 = \frac{1}{2} Ln\left(\frac{1-e^1}{e^1}\right) = \frac{1}{2} Ln\left(\frac{\frac{7}{8}}{\frac{1}{8}}\right) = 0,972955$$

$$w_i^1 = \frac{1}{8} \qquad e^1 = \sum_{i=1}^{1} w_i^1 = \frac{1}{8}$$

بخش پیاده سازی:

۱)

مقدار accuracy و f1-score در تمام کد ها قرار داده شده است.

الف)

ابتدا داده ها را خوانده و بر اساس کلاس های مختلف جدا میکنیم در یک آرایه میریزیم.

سپس برای داده های sepal ۲ بعد ابتدایی ارایه ها را جدا کرده و برای هر کلاس رسم میکنیم و در انتها نمایش میدهیم.

برای داده های petal نیز ۲ بعد بعدی را جدا کرده و برای هر کلاس جداگانه رسم میکنیم و نمایش میدهیم.

ب)

برای این سوال ، برای petal و sepal به صورت جداگانه فایل قرار داده شده است و در هر فایل تمام ۴ کرنل به همراه f1-score و accuracy محاسبه شده است.

ابتدا برای sepal :

ابتدا داده ها را خوانده و با استفاده از مپ کردن داده های string را به int تبدیل میکنیم. سپس با استفاده از train_test_split ۳۰٪ از داده ها را برای تست جدا میکنیم.

در تابع calculateSVM ، با استفاده از کتابخانه svm و برای تابع خطی و چند جمله ای ، یک تابع کلاس بندی را فیت میکنیم و سپس داده های تست را با استفاده از svc.predict(X_test) پیشبینی میکنیم.

سپس برای کرنل خطی ۵ حالت خواسته شده را محاسبه میکنیم: تابع تعریف شده برای فیت کردن svm و محاسبه پیشبینی مقادیر تست را صدا میکنیم و مقادیر بازگشتی را ذخیره میکنیم. سپس با مقادیر بازگشتی خط جداکننده را رسم میکنیم. در انتها نیز مقدار f1-score و accuracy را برای هر کرنل محاسبه میکنیم.

همین کار ها را برای چند جمله ای نیز انجام میدهیم ، فقط در ورودی تابع ، درجه را تغییر میدهیم.
برای چند جمله مقادیر زیر را محاسبه کرده ایم:

درجه : { ۲ و ۳ و ۴ } و مقدار c = { ۰/۱ و ۱ و ۱۰ }

سپس تابع calculateSVMGama را برای کرنل ها rbf و sigmoid داریم که به جای درجه ، ورودی گاما را میگیرد.

همانند کرنل های قبلی عمل شده و فقط ورودی تابع متفاوت بوده (نوع کرنل و مقادیر ویژه) است.

مقادیر برای کرنل rbf :

گاما : { ۱ و auto و scale } و مقدار c = { ۰/۱ و ۱ و ۱۰ }

Auto = 1/n_features Scale = 1 / (n_features * X.var())

برای کرنل سیگموید نیز همانند کرنل rbf بوده و مقادیر زیر را دارد:

گاما : { ۰/۱ و۰/۰۱ و ۰/۰۰۱ } و مقدار c = { ۰/۱ و ۱ و ۱۰ }

سپس برای petal نیز تمام مراحل مانند sepal است ، تنها در ابتدای فایل به جای در نظر گرفتن ۲ ویژگی ابتدایی ، ویژگی های سوم و چهارم را در نظر گرفته ایم..

د)

در مورد کرنل خطی :

هر چه مقدار c بیشتر شود دقت کلاس بندی بهتر شده و در ۰/۱ ماکزیمم حالت خود را دارد و با بیشتر شدن آن تغییری در کلاسبندی ایجاد نمیشود.

کرنل چند جمله ای:

هر چه مقدار C در درجات مختلف بیشتر شود دقت کلاس بندی بیشتر میشود و بستگی به درجه دارد که چه حالتی مقدار ماکزیمم باشد . اما در مورد درجات اینگونه نیست ، کلاس بندی در درجه ۳ بهتر از درجه ۴ و ۲ است و با افزایش آن کلاس بندی بهتر نمیشود. اما میتوان گفت در درجات ثابت با افزایش مقدار C دقت افزایش یافته است و همچنین در مقادیر C ثابت و با تغییر مقدار درجه ، تغییرات متناوب بوده و در درجه ۳ بهترین دقت را خواهیم داشت.

کرنل RBF :

در این کرنل با در نظر گرفتن گاما ثابت ، افزایش مقدار C تغییرات مختلفی دارد و حالت صعودی یا نزولی ندارد ، در بعضی نقاط مانند گاما ۱ ، با افزایش مقدار C از ۰/۱ به ۱ افزایش دقت را داریم اما با افزایش از مقدار ۱ به ۱۰ کاهش دقت را داریم. در مورد گاما auto در مقدار ۰/۱ بهترین حالت است و با افزایش به مقدار ۱ دقت دچار کاهش میشود و با افزایش از ۱ به ۱۰ مقدار دقت دوباره افزایش یافته و برابر با مقدار ۰/۱ میشود . در مورد گاما scale با افزایش از ۱ به ۱۰ مقدار کاهش نیافته و ثابت مانده است. در نتیجه میتوان گفت در این موارد مقدار C تغییر یکسانی در این کرنل ندارد و وابسته به گاما است. مقدار c برابر ۱ در مورد گاما ۱ بهترین حالت را ایجاد میکند و در مقدار ۰/۱ و ۱۰ در گاما auto بهترین مقدار را ایجاد میکند. دقت ها در گاما scale حالت میانی هستند و در بین بهترین حالت و بدترین حالت قرار میگیرند.

کرنل sigmoid :

در مورد مورد کرنل سیگموید به طور خلاصه میتوان بیان کرد که هر چه مقدار گاما برای آن کاهش یابد و در کنار آن هر چه مقدار C افزایش یابد دقت بهتری را همراه دارد اما میزان افزایش همواره یکسان نیست ، برای مثال در گاما ۰/۰۰۱ و مقدار C برابر ۱۰۰ مقدار دقت در حدود ۸۲٪ خواهد بود و در مورد گاما ۰/۰۱ و مقدار C ۱۰ مقدار دقت دوباره در حدود ۸۲٪ خواهد بود اما در مورد گاما ۰/۱ و مقدار C تا حدود ۵۰ ، مقدار دقت مناسبی نداریم.

۲)

مقدار صحت ، دقت و ماتریس در هم ریختگی در تمام کد ها قرار داده شده است.

الف)

ابتدا داده ها فراخوانی کرده ایم. سپس در تابع train_using_entropy با استفاده از کتابخانه DecisionTreeClassifier درختی با عمق ۳ و ساختار با آنتروپی میسازیم . سپس آن را بر روی داده های خود فیت میکنیم.

در تابع cal_accuracy ماتریس در هم ریختگی ، دقت و صحت را محاسبه میکنیم و برگشت میدهیم. سپس با توجه به صورت سوال که ۱۵ زیر درخت نیاز است ، ۱۵ بار کد ساخت درخت را اجرا میکنیم: (کد ساخت درخت)

ابتدا از داده های اصلی کپی میگیریم و مقدار هدف را در goal ریخته و سپس ۳ تا از ویژگی ها به صورت رندوم انتخاب شده و نام آن ها را در لیست decision_trees_features قرار میدهیم و داده های انتخاب شده را برای ساخت درخت به تابع train_using_entropy پاس داده میشود و درخت دریافت شده را به لیست درخت ها decision_trees اضافه میشود.

به اینگونه جنگل خود را میسازیم. سپس برای تست کردن جنگل خود وارد کد بعدی میشویم. در این کد ابتدا در طول لیست درختان حرکت خواهیم کرد و در هر درخت نام ویژگی را دریافت کرده و در داده های تست همین ویژگی ها را دریافت کرده و بر اساس درخت برای هر داده مقداری پیشبینی میشود. بعد از اینکه بر روی تمام این درختان این کار انجام شد لیستی از داده های پیشبینی شده توسط درختان در y ذخیره شده است و سپس با استفاده از Counter بررسی میکنیم که در این جنگل برای هر داده بیشترین عددی که این درختان پیشبینی کرده اند کدام داده ها هستند و با most_common بیشترین آن را به عنوان مقدار پیشبینی شده جنگل انتخاب میکنیم و در لیست predictions ذخیره میکنیم.

در انتها نیز predictions را با goalTest مقایسه کرده و ماتریس درهم ریختگی و صحت و دقت را محاسبه میکنیم.

ب)

برای adaboost ، ابتدا داده ها را فراخوانی میکنیم. سپس در تابع calculateAdaboost با استفاده از AdaBoostClassifier و مقادیر n_estimators ، دسته بند ادابوست با تعداد n_estimators درخت میسازیم. سپس این تابع به دست آمده را بر روی داده ها فیت میکنیم. در تابع cal_accuracy

مقدار ماتریس در هم ریختگی و صحت و دقت را محاسبه میکنیم. سپس در کد اصلی خود تابع calculateAdaboost را با داده های خود و همچنین تعداد درختان ۱۰ فراخوانی میکنیم و دسته بندی بازگشت داده شده را برای پیشبینی داده های تست استفاده میکنیم و مقدار پیشبینی را در y_pred ذخیره میکنیم.

ج)

در کد قسمت قبل ، در ورودی calculateAdaboost به جای مقدار ۱۰ ، مقادیر ۵ ، ۲۰ و ۵۰ قرار میدهیم و ماتریس و مقدار صحت و دقت را برآورد میکنیم.

د)

برای دسته بند XGBoost ، ابتدا داده ها را فراخوانی میکنیم. سپس با استفاده از کتابخانه XGBClassifier یکی دسته بند ساخته و سپس بر روی داده های خود فیت میکنیم. سپس داده های تست را با این دسته بند پیشبینی میکنیم و در آخر مقدار صحت و دقت را برای آن محاسبه میکنیم.

در مورد پارامتر های بهینه XGBoost ، هر پارامتر را به صورت جداگانه و ترکیبی بر روی این دسته بند تست کرده ام و برای هر پارامتر بهترین پارامتر و پارامتر های تست شده را نوشته ام. بنده از تابع های grid search و random search برای محاسبه نیز استفاده کرده ام که به علت طولانی بودن مقدار محاسبات (تقریبا نزدیک به ۱ ساعت) به صورت دستی تمام مقادیر را تست کرده ام.