به نام خدا

گزارش تمرین اول درس یادگیری ماشین

پیاده سازی و بررسی الگوریتم ID5

نویسنده : پیمان حسن آبادی

استاد: دکتر آبین

اسفند ۱۳۹۵

شرح الگوریتم ID5

الگوریتم ID5 یکی از روش های رسم درخت تصمیم است که برخلاف روش ID3درخت را به صورت افزایشی میسازد.

در این روش ابتدا باید ریشه درخت انتخاب شود. برای این کار مراحل زیر را انجام میدهیم،

۱. یک از داده ها را وارد سیستم میکنیم و آنتروپی کل(انتروپی داده های دیده شده در سیستم) را محاسبه میکنیم.
۲. اگر آنتروپی کل صفر بود برو به ۱
۳. در غیر اینصورت آنتروپی هر فیچر را جداگانه حساب کن.
۴. فیچر با کمترین آنتروپی را محاسبه کن و به عنوان ریشه برگردان

پس از انتخاب ریشه درخت نوبت به رسم ادامه درخت میرسد. در این مرحله:

۱. هر داده را به سیستم وارد میکنیم و توسط ریشه سعی میکنیم کلاس بندی شود.
۲. اگر آنتروپی فرزند های گره صفر باقی ماند برو به ۱
۳. در غیر این صورت مانند مرحله اول فیچر با کمترین آنتروپی را انتخاب کن.
۴. فیچر انتخاب شده در مکان فرزندی که آنتروپی آن غیر صفر شده است بگذار و نود های جدید را به درخت اضافه کن.

همان طور که از مراحل بالا پیداست، روش ID5 نیازی به کل داده ها برای رسم درخت ندارد و میتواند در هر مرحله با ورود داده جدید درخت را بازسازی نماید.

برای انکه درخت بهترین فیچر هارا در عمق های اولیه داشته باشد، در این روش یک مرحله pull up هم باید اجرا شود. الگوریتم pull up شامل مراحل انتخاب نودها و مقایسه آنتروپی ها ، جابه جا کردن نود با آنتروپی کمتر با نود با آنتروپی بالاتر، و ادغام است.

الگوریتم pull up هم به شرح زیر است:

۱. به ازای تمامی نود های فیچر ، پدر و فرزندی را انتخاب میکنیم که آنتروپی فرزند بیشتر از پدر باشد.
۲. به ازای تمامی حالات فرزند، یک نود از فیچر پدر را قرار میدهیم.
۳. تمامی زیر شاخه ها را ترکیب میکنیم تا زیر درخت ها ادغام شوند.

الگوریتم pull up باعث میشود تا فیچر هایی که قدرت تفکیک کنندگی بالاتری دارند در سطوح بالاتری قرار بگیرند تا داده سریع تر کلاس بندی شود.

توضیح کد

پروژه انجام شده با زبان برنامه نویسی پایتون پیاده سازی شده است.قبل از توضیح کد لازم به ذکر است که ساختمان داده استفاده شده برای درخت در این پروژه linked list ۲طرفه بوده است. پروژه متشکل از ۳ فایل است که در مورد آنها صحبت خواهیم کرد:

۱. Instance.py

این فایل شامل کلاسی داده ورودی است. هر سطر از فایل اکسل خوانده شده به عناون یک instance ساخته میشود و درون آرایه ریخته میشود. با این کار به راحتی میتوانیم به فیچر ها و لیبل داده دسترسی داشته باشیم.

۲. Treenode.py

این کلاس مربوط به object هر گره از درخت است. هر آبجکت مشخصات زیر را خواهد داشت:

- Parent : گره پدر هر نود
- children_labels : لیبل های مثبت و منفی فرزندان هر گره
- children_nodes : دیکشنری از فرزندان هر گره
- is_leaf : گره برگ است یا نه
- Labels : تعداد لیبل های مثبت و منفی دیده شده هر گره
- is_feature: آیا نود نشان دهنده یک فیچر هست یا نه
- feature_name : نام فیچر هر گره
- seen_instances : داده های مشاهده توسط هر گره
- false_choice. : تعداد داده هایی که توسط گره اشتباه کلاس بنده شده اند (برای هرس)

توابع پیاده سازی شده برای این کلاس:

- Entropy : آنتروپی مربوط به گره
- Update_labels : آپدیت کردن لیبل های مشاهده شده با ورود هر داده
- Info_gain : میزانinformation gain هر نود
- Update_children : آپدیت کردن فرزندان هر گره، شامل اضافه کردن فرزند و آپدیت لیبل های آنها
- add_instance : اضافه کرده داده به هر گره برای آپدیت attribute ها
- get_path_to_root : مسیری که از هر نود به rootوجود دارد. این مسیر همان مسیر تصمیم گیری است
- test_instance : اضافه کردن داده برای فاز test
- probe : نشان دهنده این است که این گره داده را مثبت یا منفی کلاس بندی میکند.
- Evaluate : اضافه کردن داده به نود برای فازevaluation

۳. Mian. Py

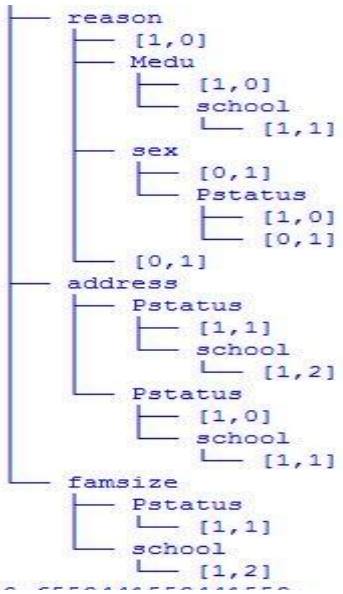
این فایل، فیال اجرای برنامه اصلی است. در این فایل ابتدا با استفاده از تابع load_data و گرفتن نام فایل وروردی از کاربر، هر سطر از فایل به عنوان یک instanceبه آرایه Instances global اضافه میشود.

در ادامه با استفاده از تابع choose_root() داده ها به ترتیب به سیستم اضافه میشنود و هر گاه آنتروپی کلی سیستم غیر صفر شود با استفاده از فیچری که در بین داده های دیده شده کمترین آنتروپی را داشته باشد، tree_root انتخاب میشود.

وقتی که گره tree_root انتخاب شود با استفاده از تابع crosValidate و انتخابK مورد نظر، داده ها در هر مرحله به ۲ دسته trainو testتقسیم شده و درخت ساخته میشود.

برای ساخت درخت با استفاده از تابع make_treeداده های مربوط به trainرا به ترتیب وارد سیستم میکنیم. اگر آنتروپی هر یک از گره های داخل درخت مخالف صفر شد یک فیچر جدید را برای آن زیر درخت انتخاب کرده و با ساخت فرزندان آن، آن نود را به درخت اضافه میکنیم.

در هر مرحله با استفاده از تابع check_for_pull_up نیاز گره ها برای بالا رفتن در درخت را بررسی میکنیم. برای این کار، ابتدا برای هر نود آنتروپی را با پدرش مقایسه میکنیم و اگر کمتر بود با جابه جا کردن نود و سپس تابع create_tree زیر درخت های مربوطه را ساخته و در هم ادغام میکنیم.



همان طور که از تصویر بالا پیداست جای address و Pstatus جابجا شده است.

بعد از اتمام فاز train و test ، مجموعهevaluation را مشخص کرده و به سراغReducedErrorPruning میرویم. در این تابع با استفاده از درخت ساخته شده و و مجوعه evaluation تعداد داده های کلاس بندی شده برای هر

گره را بدست آورده و برگ های درخت را یکی حذف میکینم و با استفاده از precision ، دقت را مقایسه میکنیم و در صورت نیاز به حذف گره، به آپدیت لیبل های پدر اقدام به حذف آن میکنیم.

نتایج بدست آمده از پیاده سازی

الف) برای ارزیابی پیاده سازی ابتدا از روش 5-fold cross validation درخت را ساخته و تست میکنیم. دقت به دست آمده حاصل از میانگین دقت های به دست آمده در هر مرحله از فازtest است. نتایج حاصل از تست را در جدول زیر میبینیم

مرحله	میانگین دقت
۱	۶۵,۳۳
۲	۶۸,۲۵
۳	۶۰,۷۴
٤	۷۳,۵۸
٥	۷۳,۵۸
٦	۶۶,۲۵

برای ارزیابی بهتر الگوریتم با روشk-fold cross validation و به ازای k های مختلف نیز بررسی شد که نتایج حاصل را در جدول زیر میبینیم:

مرحله	k	دقت
۱	۳	۶۵,۵۵
۲	۴	۶۶,۰۰
۳	۴	۶۳,۱۲
٤	۷	۶۸,۱۵
٥	۷	۶۷
٦	۱۰	۶۶,۲۳
۷	۱۰	۷۳,۲۳

همانطور که از نتایج بالا مشخص است دقت روی این الگوریتم در بازه [60,75] قرار دارد.

اگر درخت را با استفاده از تمام داده ها و روی کل مجموعه بسازیم و سپس تست کنیم دقت ۱۰۰ درصدی به دست میدهد که البته این امر بدیهی است. چون با استفاده از همان داده ها ساخته شده است!

ب) در این قسمت درخت ساخته شده با استفاده از الگوریتمREP هرس کرده و دقت درخت حاصل را قبل و بعد از هرس محاسبه میکنیم.

دقت قبل از هرس	دقت بعد از هرس روی validation set
65.33	66.53
64.87	65.02
66	67.85

همان طور که از جدول مشهود است بعد از اعمال هرس دقت روی مجموعه validation افزایش پیدا میکند و همچنین سرعت تصمیم گیری نیز پیشرفت محسوسی داشته است.

نکته قابل توجه این است اگر درخت با این روش ID5 ساخته شود ، به دلیل وجود فازpull up و بهبود درخت در این فاز، تعداد نود های بسیار کمی در این هرس از بین میروند. برای مثال برای تست انجام شده در این آزمایش تعداد نود های هرس شده ۳و ۳ و۴ نود بوده است.