دانشگاه شهید بهشتی

دانشکده مهندسی و علوم کامپیوتر

گزارش تمرین اول درس یادگیری ماشین

پیاده سازی الگوریتم درخت تصمیم ID5 و بهبود آن با الگوریتم هرس REP

نگارش

سهیل ضیائی قهنویه

استاد

دکتر احمد علی آیین

فروردین ۱۴۰۰

چکیده

ID5 یک الگوریتم افزایشی برای ساخت درخت تصمیم است که بر خلاف ID3 با در اختیار داشتن تمام نمونه ها درخت را نمی سازد بلکه با دریافت هر نمونه، ساختار درخت تصمیم را بازسازی می کند. در این پروژه با استفاده از زبان Python این الگوریتم پیاده سازی و دقت آن بر روی مجموعه های دیتا گزارش شده است. همچنین با پیاده سازی الگوریتم هرس REP دقت الگوریتم بهبود یافته است، که نتایج آن در پایان ارائه و تحلیل می شود.

کلیدواژه‌ها

Decision tree، Incremental، ID5، k – fold cross validation، accuracy، Reduced Error Pruning، Python

فهرست نوشتار

بخش اول: مقدمه

معرفی الگوریتم ID5

ID5 یک الگوریتم افزایشی برای ساخت درخت تصمیم است که با دریافت هر نمونه، ساختار درخت تصمیم را بازسازی می کند. در الگوریتم قبلی از این خانواده یعنی ID3 درخت تصمیم با در اختیار داشتن تمام نمونه ها ساخته می شود، ولی در الگوریتم های بعدی یعنی ID4 و ID5 درخت تصمیم به صورت افزایشی تکمیل می گردد.

در مقاله مرجع الگوریتم ID5 [1] پس از مشاهده هر نمونه آموزشی مراحل به صورت زیر ارائه شده است:

۱- برای هر ویژگی بالقوه در هر گره درخت، تعداد نمونه های مثبت و منفی آن ویژگی و مقدار مشاهده شده اش را به روز رسانی کن.

۲- اگر غیر از ویژگی فعلی که تصمیم گیری در گره بوسیله آن انجام می شود، ویژگی دیگری کیفیت اطلاعاتی مناسب تری دارد، درخت را با بالا کشیدن آن ویژگی بازسازی کن.

۳- اگر در یک گره فقط نمونه های مثبت یا فقط منفی مشاهده می شود نمونه را در آن ذخیره کن؛ پایان.

۴- به صورت بازگشتی درخت تصمیم زیر مقدار ویژگی جدید به روزرسانی کن.

الگوریتم بالا کشیدن یک ویژگی جدید به جای ویژگی فعلی به صورت زیر است:

۱- به صورت بازگشتی ویژگی را به ریشه زیر درخت مستقیم گره فعلی منتقل کن. (در صورت نیاز مجموعه نمونه های زیردرخت را بسط بده)

۲- گره فعلی و هر زیر درخت مستقیم آن را با جابجا کردن ویژگی فعلی و ویژگی جدید تبدیل به زیر درخت هایی با ویژگی جدید در ریشه آن ها تبدیل کن.

۳- زیر درخت های بدست آمده را ادغام کن تا یک درخت با ویژگی جدید در ریشه ساخته شود.

معرفی هرس REP

هنگامی که درخت تصمیم ساخته می شود، بسیاری از شاخه ها ناهنجاری های داده های آموزشی را پوشش می دهند. روش های هرس درخت این مشکل بیش برازش روی داده ها (Overfitting) را برطرف می کند. در این روش ها معمولاً از اقدامات آماری برای از بین بردن شاخه های کمتر قابل اطمینان استفاده می شود.

یکی از روش های هرس درخت تصمیم Reduced Error Pruning (REP) است که از روش های پس هرس (Post Pruning) به شمار می آید. این روش از عمیق ترین گره های غیر برگ شروع به بررسی می کند که در صورتی که با جایگزینی یک گره با بیشترین برچسب مشاهده شده در گره های متصل به آن از دقت درخت کم نشود، آن گره هرس می شود. [2]

الگوریتم REP به صورت زیر است:

۱- داده های آموزشی را به دو قسمت آموزش و هرس تقسیم کن.

۲- درخت کاملی با داده های قسمت آموزش بساز که به درستی همه نمونه های آموزش را پیش بینی کند.

۳- تا زمانی که خطای پیش بینی روی داده ای قسمت هرس زیاد نمی شود،

الف - هر گره را با یک برگ شامل بیشترین برچسب زیر درخت خودش جایگزین کن.

ب - درخت را با داده های قسمت هرس ارزیابی کن.

ج - جایگزینی با بیشترین کاهش خطا را انجام بده.

۴- درخت حاصل را به عنوان نتیجه بده.

در ادامه این گزارش نحوه پیاده سازی و نتایج اجرای الگوریتم تشریح خواهد شد.

بخش دوم: نحوه پیاده سازی

زبان و پارادایم برنامه نویسی

پروژه به زبان پایتون Python و به صورت شیء گرا نوشته شده است به گونه ای که در آینده کلاس درخت تصمیم ID5 قابلیت انتشار به صورت کتابخانه ای مستقل از سایر اجزای پروژه داشته باشد. سعی شده است به جای ذخیره و به روزرسانی مکرر داده های تکراری (برای مثال تعداد نمونه های مثبت و منفی به ازای مقادیر مختلف ویژگی ها در زیر درخت های یک گره) حتی الامکان از متدهایی بازگشتی (recursive) برای بازیابی این داده ها استفاده شود.

ساختار پروژه

پروژه از ۴ ماجول زیر تشکیل شده است:

۱- ماجول اصلی main.py – که وظیفه فراخوانی توابع و تولید نتایج را دارد.

۲- ماجول درخت تصمیم id5Classifier.py – که شامل مجموعه گسترده ای از ساختارهای داده ای و عملیاتی مورد نیاز از جمله الگوریتم های ID5 و REP می باشد.

۳- ماجول مجموعه داده dataSet.py – که وظیفه بارگذاری مجموعه داده را دارد. در این ماجول از کتابخانه xlrd-2.0.1 برای خواندن مجموعه داده از اکسل استفاده شده است.

۴- ماجول ارزیاب validator.py – که ارزیابی و محاسبه دقت الگوریتم را به عهده دارد و شامل متد ارزیابی متقابل چند مرحله ای k – fold cross validation و همچنین متد ارزیابی پایه می باشد.

مدل داده ای ماجول درخت تصمیم ID5

ماجول درخت تصمیم id5Classifier.py از ۴ کلاس زیر تشکیل شده است:

۱- کلاس Id5Classifier – شامل متدهای آموزش train، تست test، آموزش و هرس trainAndPrune است که به صورت classmethod پیاده سازی شده اند.

۲- کلاس زیر درخت SubTree - جهت مدل سازی کل درخت یا بخشی از آن توسعه داده شده است و شامل خصوصیات شاخه پدر parentBranch و گره فرزند rootNode و متد ترسیم print می باشد.

۳- کلاس گره TreeNode - شامل خصوصیات شاخه پدر parentBranch، فیلدی بابت تشخیص برگ بودن گره isLeaf، نمونه های ذخیره شده در گره های برگ leafSavedInstances، ویژگی که در گره های غیر برگ بر اساس آن تصمیم گیری می شود decisionFeature، و لیستی از شاخه های فرزند در گره های غیر برگ childBranches است.

در جدول زیر متدهای کلاس گره تشریح شده اند:

نام متد	شرح
predict(instance)	برچسب یک نمونه داده ای را پیش بینی می کند
expandLeafNode(feature)	بر اساس مقادیر مختلف یک ویژگی، گره برگ را بسط می دهد
shrinkDecisionNode()	برعکس متد بالا، یک گره غیر برگ را به همراه زیرشاخه های آن تبدیل به گره برگ با بیشترین برچسب نمونه ای ذخیره شده در زیرشاخه هایش می کند
getPositiveCount (decisionFeature, featureValue)	تعداد نمونه های مثبت یک گره و زیر شاخه هایش را برای یک ویژگی و یک مقدار محاسبه می کند
getNegativeCount (decisionFeature, featureValue)	تعداد نمونه های منفی یک گره و زیر شاخه هایش را برای یک ویژگی و یک مقدار محاسبه می کند
getPotentialFeatures(dataSet)	ویژگی های بالقوه در یک گره (که در گره های پدر تعیین تکلیف نشده اند) را بدست می آورد
getLabel()	برچسب یک گره برگ را با توجه به بیشترین برچسب نمونه های ذخیره شده روی آن مشخص می کند
getSavedInstances()	نمونه های ذخیره شده در یک گره و زیر شاخه های آن را بدست می آورد
getAllLeafNodes()	همه گره های برگ زیر مجموعه یک گره را بدست می آورد
pruneOnce()	از میان پایین ترین سطح گره های غیر برگ، یکی که نسبت اختلاف نمونه های مثبت و منفی آن از بقیه بیشتر است انتخاب می کند و آن را تبدیل به گره برگ می کند. نسبت مذکور به صورت زیر محاسبه می شود: $\frac{\lvert positiveCount - negativeCount \rvert}{positiveCount + negativeCount}$

getVisualNode()	نمایش ترسیمی گره را آماده می کند

۴- کلاس شاخه TreeBranch – شامل خصوصیات گره پدر parentNode، مقدار ویژگی featureValue، و گره فرزند childNode است و متدهای آن در زیر معرفی شده اند:

نام متد	شرح
getPositiveCount(decisionFeature, featureValue)	تعداد نمونه های مثبت یک شاخه و زیر شاخه هایش را برای یک ویژگی و یک مقدار محاسبه می کند
getNegativeCount(decisionFeature, featureValue)	تعداد نمونه های منفی یک شاخه و زیر شاخه هایش را برای یک ویژگی و یک مقدار محاسبه می کند
getVisualNode()	نمایش ترسیمی شاخه را آماده می کند

نحوه انتخاب ویژگی ها

برای بالاکشیدن (Pullup) یک ویژگی، از معیار Information Gain استفاده شده است که بر اساس آنتروپی برچسب نمونه های ذخیره شده در زیرشاخه های یک گره محاسبه می شود:

$$Gain(S,A)=E(S)-I(S,A)=E(S)-\sum_i \frac{|S_i|}{|S|}\cdot E(S_i)$$

یادآوری می شود آنتروپی با رابطه زیر محاسبه شده است:

$$E(S)=-p_{\oplus}\cdot\log_2 p_{\oplus}-p_{\ominus}\cdot\log_2 p_{\ominus}$$

معیار ارزیابی

معیار ارزیابی (validation) در تمامی مراحل این پروژه دقت (accuracy) در نظر گرفته شده است که به معنی تعداد پیش بینی صحیح مدل نسبت به تمام پیش بینی ها تعریف می شود:

$$accuracy = \frac{true\ positive + true\ negative}{true\ positive + true\ negative + false\ positive + false\ negative}$$

بخش سوم: ارائه نتایج

ارزیابی متقابل چند مرحله ای k – fold cross validation

نتیجه ارزیابی ۵ مرحله ای الگوریتم بر روی مجموعه داده های فایل data.xls شامل ۳۹۵ نمونه و ۱۱ ویژگی به صورت زیر است:

مرحله (fold)	دقت (accuracy)
۱	0.4430379746835443
۲	0.5822784810126582
۳	0.43037974683544306
۴	0.35443037974683544
۵	0.5063291139240507
میانگین	0.4632911392405063

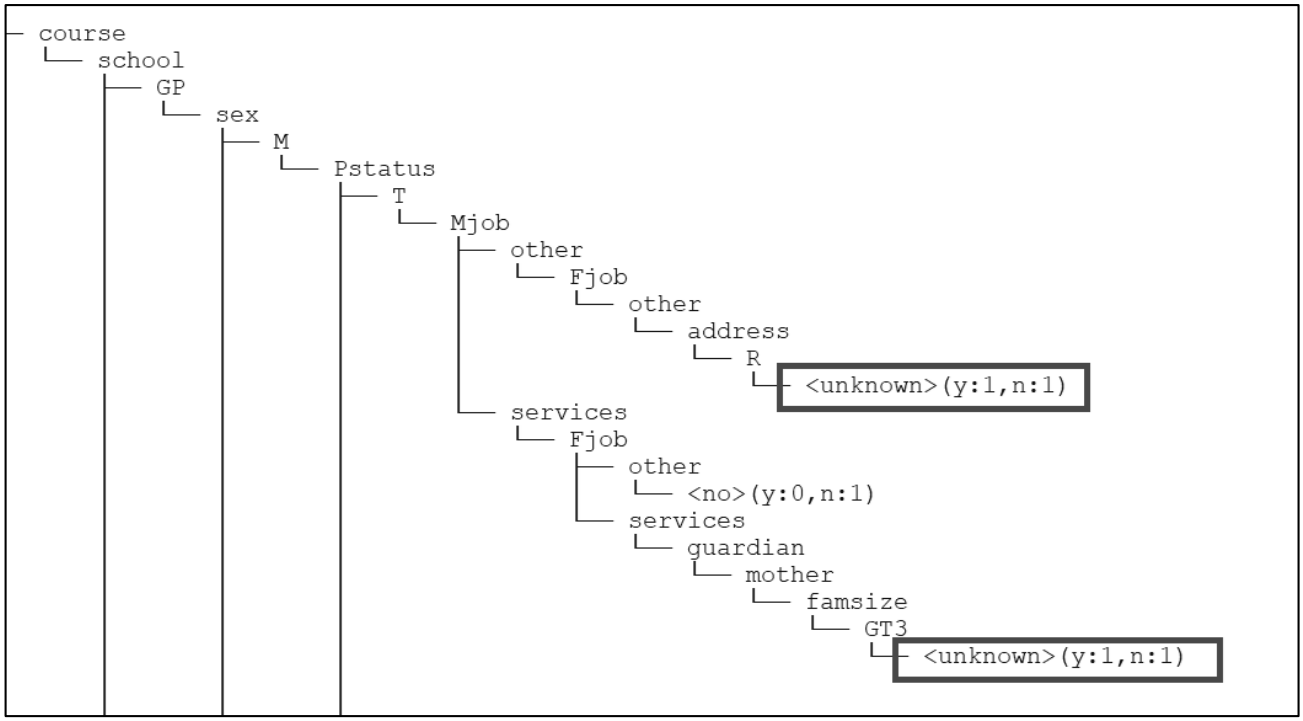
دقت میانگین این الگوریتم، بر روی داده های مفروض نتیجه خوبی نشان نمی دهد که از علل آن می توان به بیش برازش Overfitting اشاره کرد.

دقت روش بر روی کل مجموعه داده

انتظار می رود در صورتی که درخت تصمیم با کل مجموعه داده ها ساخته شود، نتیجه تست روی همان مجموعه دقت ۱۰۰ درصد داشته باشد. ولی اجرای الگوریتم پیاده سازی شده روی مجموعه داده های فایل data.xls و تست روی همین مجموعه دقت حدودا ۹۸ درصد نشان می دهد. علت این اختلاف، نمونه هایی از داده هستند که با وجود مشابهت در تمامی ویژگی ها، برچسب متفاوتی دارند. به عنوان مثال دو نمونه زیر:

163	GP	M	R	GT3	T	3	2	other	other	course	mother	no
264	GP	M	R	GT3	T	3	2	other	other	course	mother	yes

در برنامه، گره های برگ حاوی این گونه نمونه ها (به فرض تساوی تعداد مثبت و منفی) برچسب unknown دارند که در ارزیابی تولید خطا می کنند:



ارزیابی دقت بعد از هرس REP

نتیجه ارزیابی ۵ مرحله ای الگوریتم به همراه هرس درخت به روش REP به صورت زیر است:

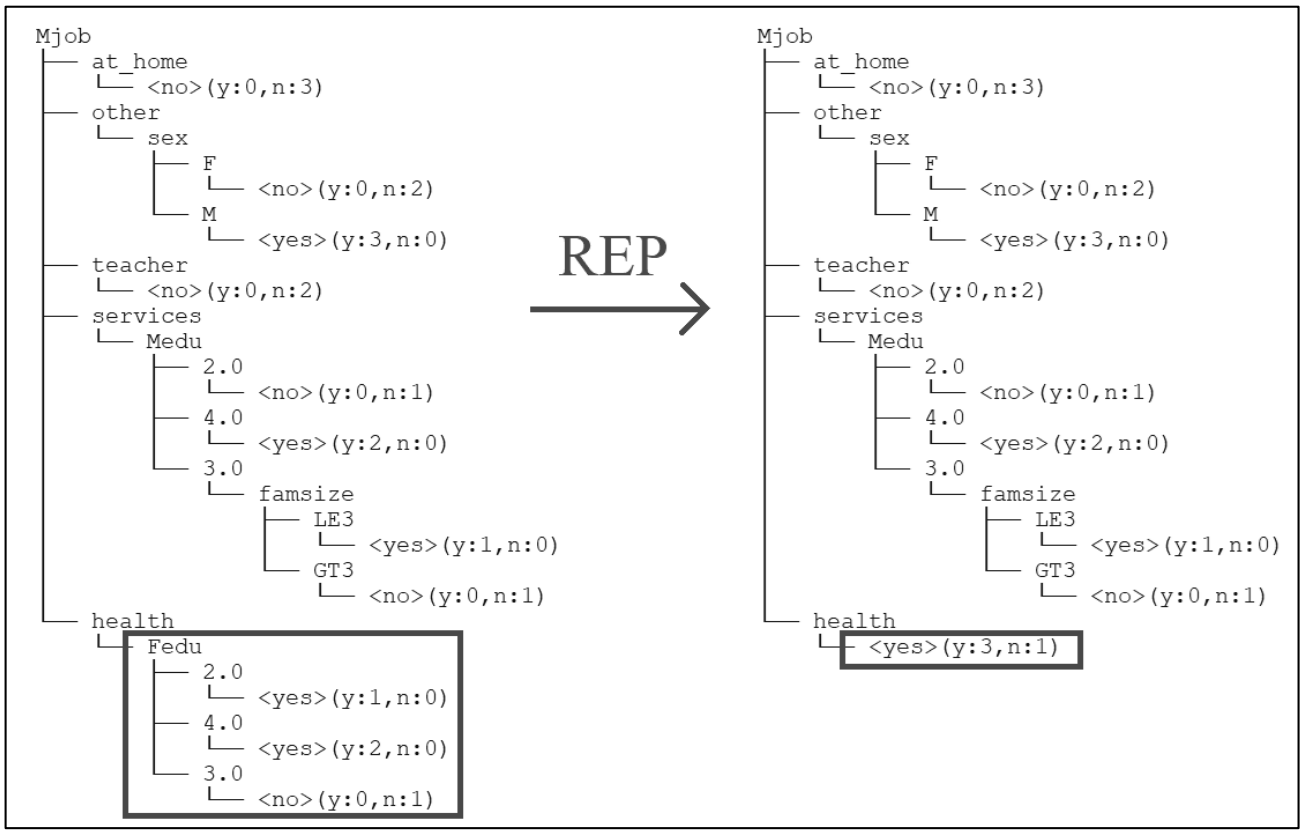
مرحله (fold)	دقت (accuracy)
۱	0.5316455696202531
۲	0.6708860759493671
۳	0.4177215189873418
۴	0.46835443037974683
۵	0.5443037974683544
میانگین	0.5265822784810126

لازم به ذکر است در هر مرحله ۷۵ درصد از داده های آموزشی برای ساختن درخت و ۲۵ درصد از آن برای ارزیابی هرس به کار برده شده است. نتایج حاصله ۶ درصد بهبود بر اثر اضافه شدن الگوریتم هرس REP به الگوریتم ID5 را نشان می دهد.

نمونه ای از نمایش گرافیکی درخت

برای ترسیم گرافیکی درخت از کتابخانه anytree-2.8.0 استفاده شده است. علاوه بر گره ها، برای نمایش شاخه های درخت که نمایش دهنده مقدار ویژگی ها می باشند نیز از کلاس Node استفاده شده است.

درخت تصمیم ساخته شده با ۲۰ نمونه اول مجموعه داده قبل از هرس و بعد از هرس به صورت زیر ترسیم شده است:



همان گونه که مشاهده می شود در یک مرحله از هرس، ویژگی Fedu و مقادیر آن با برچسب yes جایگزین شده است.

مراجع

[1] P. E.Utgoff, "ID5: An Incremental ID3," in *Fifth International Conference on Machine Learning*, University of Michigan, Ann Arbor, 1988.

[2] R. A. Rinkal Patel, "A Reduced Error Pruning Technique for Improving Accuracy of Decision Tree Learning," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 3, no. 5, 2014.