به نام خدا

گزارش پروژه منطق فازی

درس هوش محاسباتی

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

استاد درس : دکتر حسین کارشناس

تهیه کنندگان :

علیرضا دستمالچی ساعی، محمد حسین دهقانی

خرداد ۱۴۰۲

مقدمه

در این پروژه خواسته شده بود که از یک دیتاست مخصوص پیام‌های ایمیل که مشخص می‌کند ایمیل اسپم است یا نه، یک سیستم استنتاج فازی با یک مجموعه قانون بسازیم که با گرفتن ویژگی‌های استخراج شده از متن یک ایمیل، اسپم بودن یک ایمیل را تشخیص دهیم.

پیاده‌سازی

این برنامه به دنبال پیدا کردن بهترین مدل رده‌بندی اسپم و غیر اسپم برای داده‌های متنی است. برای این کار از الگوریتم ژنتیک استفاده شده است.

در این برنامه، ابتدا داده‌های متنی از یک فایل خوانده شده و سپس آن‌ها به یک دیتافریم تبدیل می‌شوند. سپس از روش TF-IDF برای استخراج ویژگی‌های مهم استفاده می‌شود و سپس با استفاده از روش کاهش بعد PCA، تعداد ویژگی‌ها (تعداد بعدها) کاهش داده می‌شود. در ادامه، با استفاده از روش feature_selection تنها ویژگی‌های مهم برای دسته‌بندی انتخاب می‌شوند.

الگوریتم تکاملی

برای الگوریتم تکاملی، از کتابخانه آماده Deap که یک کتابخانه قوی و جامع برای اجرای الگوریتم‌های تکاملی است، استفاده کرده‌ایم. ابتدا ژنوتایپ یا کروموزوم (Individual) و تابع ارزیابی (Evaluation) را برای toolbox کتابخانه تعریف می‌کنیم.

نمایش ما (ژنوتایپ) به صورت یک دیکشنری می‌باشد که در toolbox رجیستر شده و دارای ۲ بخش می‌باشد:

- قسمت Rules که شامل لیستی از قوانین است که به آن Rule base نیز می‌گوین، بدین صورت که هر قانون X1 تا Xn و Y دارد. تعداد قوانین هم بین ۵۰ تا ۵۰۰ عدد می‌باشد.

- قسمت Linguistic Functions که برای هر متغیر زبانی یک تابع درجه عضویت تعریف کرده‌ایم که در ابتدا به صورت رندوم انتخاب می‌شود.

سپس، برای تغییر، از توابعی که خودمان تعریف کرده‌ایم استفاده می‌کنیم:

- **بازترکیب:**

  برای بازترکیب از عملگر تغییر، از one_point_crossover استفاده کرده‌ایم.

- **جهش:**

برای جهش از عملگر تغییر، با یک احتمالی مقادیر قانون‌هارا تغییر می‌دهیم سپس با همان احتمال قبلی، توابع عضویت مخصوص هر مقدار زبانی، m و s را جهش می‌دهیم.

برای تابع Evaluate که برازندگی هر ژنوتایپ (کروموزم) را حساب می‌کند، با استفاده فرمول‌هایی که در داک ذکر شده‌اند، ساتفاده شده است.

نتایج

برنامه برای تعداد جمعیت ۱۰ و تعداد نسل‌های ۱۰ با ۳ مقدار زبانی، نتیجه زیر را با دقت 90 حاصل کرده است(۱۱ قانون):

Gen	nevals	avg	std	min	max
۰	10	2728.68	2523.3	0	7762.16
۱	8	5190.56	1879.14	3164.81	9882.28
۲	9	6383.23	1980.43	4108.52	9882.28
۳	9	6314.78	2523.94	2983.26	9882.28
۴	9	5409.89	3965.59	0	10636.3
۵	9	6222.58	3907.63	596.664	12346.7
۶	9	7305.13	3014.4	328	9761.59
۷	10	5834.59	4636.2	0	14877
۸	9	6857.43	4724.04	0	16916
۹	9	12228.9	4046.9	4330.8	16923.6
۱۰	8	12447.1	4932.68	4096.39	16923.6

Best individual: {'rules': [[1, 0, 2, 1, 0, 0], [2, 0, 1, 2, 1, 1], [2, 2, 2, 1, 0, 1], [2, 0, 0, 0, 0, 0], [0, 0, 1, 2, 1, 1], [1, 0, 1, 2, 1, 1], [0, 2, 2, 2, 1, 1], [0, 1, 2, 2, 1, 1], [2, 0, 1, 2, 1, 1], [0, 0, 0, 2, 0, 1], [2, 0, 2, 0, 1, 0]], 'ling_funcs': {0: (<function rect_trap at 0x7f7b6cd1b400>, 0.2597256470909224, 0.2484158089221977 2), 1: (<function rect_trap at 0x7f7b6cd1b400>, -0.05339911815644260 4, 0.9552181094331375), 2: (<function gaussian at 0x7f7b6cd1ae60>, -0.4030993382928567, 0.4065217311866878)}}

accuracy 0.9063509149623251

با تحلیل نتایج می‌توان به موارد زیر دست یافت:

- علت برخی min هایی که در وسط اجرا صفر شده‌اند به دلیل mutation می‌باشد.
- الگوریتم تکاملی روند صعودی را طی کرده است بجز در نسل ۶ که افت کوچکی داشته اما سپس پیشرفت نسبتا خوبی داشته است.
- بهترین ژنوتایپ ما که در بالا آورده شده است، برای ۳ مقدار زبانی با توابع درجه عضویت زیر است:
  - Low: rect_trap
  - Medium: rect rap
  - High: gaussian

با تحلیل قوانین بدست آمده در بهترین ژنوتایپ با اطلاعات ذکر شده در بالا می‌توان فهمید که اکثر ایمیل‌هایی که اسپم تشخیص داده‌شدند مقادیر زبانی

Medium یا High را در ویژگی‌ها داشتند به خصوص ویژگی چهارم که به تنهایی در یک قانون باعث اسپم تشخیص داده‌شدن آن ایمیل شده است.

در نهایت بدست آوردن همچین نتیجه‌ای با تقسیم ۵۰ درصدی آموزش و تست عملکرد خوب الگوریتم تکاملی و ژنوتایپ بدست آمده دارد.

در مرحله‌ی دوم با اجرای این کد با تعداد نسل‌های ۱۰ و تعداد جمعیت ۱۰ ولی با ۵ مقدار زبانی و ۳۰ قانون نتایج زیر حاصل شده‌است:

Gen	nevals	avg	std	min	max
۰	10	8831.64	7137.22	0	21031.6
۱	7	13882.7	6875.92	891.316	21882.6
۲	9	13874.7	10396.3	685.361	34121.3
۳	7	13236.3	11459.6	1787.57	34121.3
۴	9	15715.8	8394.14	4913.41	34121.3
۵	10	15381.1	6592.33	7540.41	29822.3
۶	10	18054.3	8689.16	1902.88	28923.3
۷	10	24218.1	4993.08	14565.1	29614.1
۸	9	20289.8	9869.81	740.027	29614.1
۹	10	18864.4	10102.3	0	29614.1
۱۰	10	20078.8	9755.71	1958.4	32511

Best individual: {‘rules’: [[4, 3, 1, 4, 1, 1], [2, 4, 4, 2, 3, 0], [1, 1, 1, 3, 4, 1], [0, 1, 3, 3, 2, 1], [4, 2, 0, 1, 3, 1], [2, 0, 3, 3, 1, 1], [1, 1, 2, 4, 4, 0], [3, 3, 0, 2, 1, 1], [3, 2, 3, 2, 0, 1], [1, 4, 1, 1, 0, 1], [2, 1, 1, 4, 1, 0], [2, 2, 1, 2, 3, 0], [3, 2, 2, 0, 1, 1], [3, 1, 3, 4, 1, 1], [4, 2, 2, 0, 3, 0], [3, 0, 0, 2, 0, 1], [1, 1, 2, 4, 1, 1], [3, 4, 4, 1, 2, 0], [0, 2, 3, 4, 0, 0], [1, 3, 2, 2, 4, 0], [2, 0, 2, 1, 3, 1], [0, 4, 4, 0, 4, 0], [0, 3,

4, 1, 2, 1], [3, 0, 4, 4, 1, 1], [4, 4, 1, 4, 1, 1], [1, 4, 1, 3, 2, 1], [1, 4, 0, 4, 4, 0], [4, 3, 0, 4, 1, 0], [4, 1, 1, 4, 1, 0], [4, 3, 4, 4, 1, 1]], 'ling_funcs': {0: (<function sigmoid at 0x7f7b6cd1ad40>, 0.1673915916426207, 0.325079136985742), 1: (<function rect_trap at 0x7f7b6cd1b400>, -0.178681704111568825, 0.6728158429298997), 2: (<function rect_trap at 0x7f7b6cd1b400>, -0.7163853770078319, 0.2996211073906489 6), 3: (<function rect_trap at 0x7f7b6cd1b400>, 0.8200094237933757, 0.7496362214724762), 4: (<function sigmoid at 0x7f7b6cd1ad40>, 0.9151647467099906, 0.8687030234852108)}}

accuracy 0.7192825112107624

با تحلیل نتایج بدست آمده، می‌توان به موارد زیر رسید:

- ژنوتایپ‌ها در الگوریتم به مقداری کاهش برازندگی داشته‌اند به طوریکه ماکسیمم ۳۴ هزار بوده ولی بهترین ژنوتایپ برازندگی ۳۲ هزار دارد.
- صفرهای موجود در وسط الگوریتم تکاملی بدلیل mutation می‌باشد.
- توابع عضویت برای مقادیر زبانی به صورت زیر می‌باشد:
  - Very low: sigmoid
  - Low: rect_trap
  - Medium: rect_trap
  - High: rect_teap
  - Very high: Sigmoid

با افزایش تعداد قوانین و مقادیر زبانی با ۱۰ نسل و تست ۴۰ درصد به دقت ۷۱ رسیدیم که نسبت به حالت قبلی مقداری افت دقت داشته‌ایم و ممکن به است دلایل مختلفی داشته باشد، مانند:

- مقداردهی اولیه به صورت رندوم با مقادیر خاصی شروع کرده است
- میزان Exploration با احتمالات داده شده برای این الگوریتم کم بوده
- به دلیل علت قبلی، در نقطه بهینه محلی گیر کرده‌ایم.

در مرحله سوم نیز، با اجرای کد به تعداد ۱۰ نسل و تعداد جمعیت ۱۰ ولی با تعداد قوانین ۵۰ و تعداد متغیرهای زبانی ۷تا، نتایج زیر حاصل شده‌است:

Gen	nevals	avg	std	min	max
۰	10	8647.64	4569.4	3369.8	17920.3
۱	10	18172.4	7971.29	4556.83	29442.2
۲	10	24970.3	9514.38	6760.4	44509.3
۳	6	26287	5022.06	18944.7	36058.8
۴	10	29019	11429.2	12630.7	41783.6
۵	10	26321.3	17298.2	789.168	53192.8
۶	10	28479.5	14005.3	6863.96	50048.8
۷	8	32525.9	17258.7	5062.54	52320.3
۸	10	45356.9	7693.34	25726.7	52320.3
۹	9	42068.1	16809.6	6172.81	55757.2
۱۰	8	34474.9	20993.1	725.634	55795

Best individual: {'rules': [[3, 5, 2, 2, 4, 1], [5, 6, 1, 3, 1, 0], [1, 1, 5, 6, 4, 0], [2, 4, 3, 3, 3, 0], [3, 5, 1, 6, 1, 0], [2, 3, 4, 2, 2, 0], [5, 3, 4, 4, 0, 0], [1, 3, 0, 3, 3, 1], [3, 3, 0, 2, 4, 1], [3, 1, 4, 0, 1, 1], [2, 3, 2, 6, 6, 0], [2, 3, 4, 4, 4, 0], [3, 4, 1, 6, 1, 0], [3, 3, 5, 4, 5, 1], [5, 5, 2, 1, 0, 1], [2, 2, 5, 5, 2, 1], [4, 3, 0, 3, 5, 1], [1, 5, 2, 0, 5, 0], [1, 1, 2, 4, 5, 1], [2, 3, 0, 0, 0, 1], [5, 4, 6, 6, 1, 1], [1, 2, 0, 6, 1, 0], [5, 3,

3, 0, 0, 0], [4, 0, 3, 1, 2, 1], [1, 4, 0, 1, 3, 1], [3, 4, 1, 0, 0, 1], [3, 5, 2, 1, 4, 1], [3, 4, 6, 5, 4, 0], [4, 3, 3, 1, 2, 0], [4, 1, 1, 4, 5, 1], [5, 1, 2, 2, 4, 1], [4, 5, 3, 6, 5, 1], [4, 1, 2, 2, 6, 0], [5, 6, 1, 4, 0, 1], [2, 5, 0, 2, 5, 1], [5, 1, 0, 1, 1, 0], [4, 5, 6, 4, 0, 0], [3, 5, 2, 0, 6, 0], [1, 2, 1, 0, 1, 0], [4, 2, 0, 1, 5, 0], [1, 2, 0, 4, 5, 0], [0, 2, 3, 6, 4, 0], [5, 0, 2, 1, 5, 1], [4, 6, 5, 1, 4, 1], [4, 0, 6, 5, 3, 1], [2, 0, 3, 0, 4, 1], [0, 0, 1, 1, 2, 0], [0, 3, 6, 2, 2, 0], [2, 5, 5, 6, 0, 1], [4, 6, 1, 2, 3, 0]],

'ling_funcs': {0: (<function sigmoid at 0x7f7b6cd1ad40>, 0.4882531421828613, 0.9252488574391238), 1: (<function sigmoid at 0x7f7b6cd1ad40>, 0.1887179179219746 8, 0.0851921067933958 4), 2: (<function gaussian at 0x7f7b6cd1ae60>, 0.0532831919705014 64, 0.7669474526119026), 3: (<function sigmoid at 0x7f7b6cd1ad40>, 0.867398853556133, 0.6941425929969136), 4: (<function rect_trap at 0x7f7b6cd1b400>, 0.8327768822356998, 0.6584981422332299), 5: (<function sigmoid at 0x7f7b6cd1ad40>, 0.972964211556691 4, 0.1209842576644102 3), 6: (<function sigmoid at 0x7f7b6cd1ad40>, 0.6309064453410838, 0.8866333122999813)}}

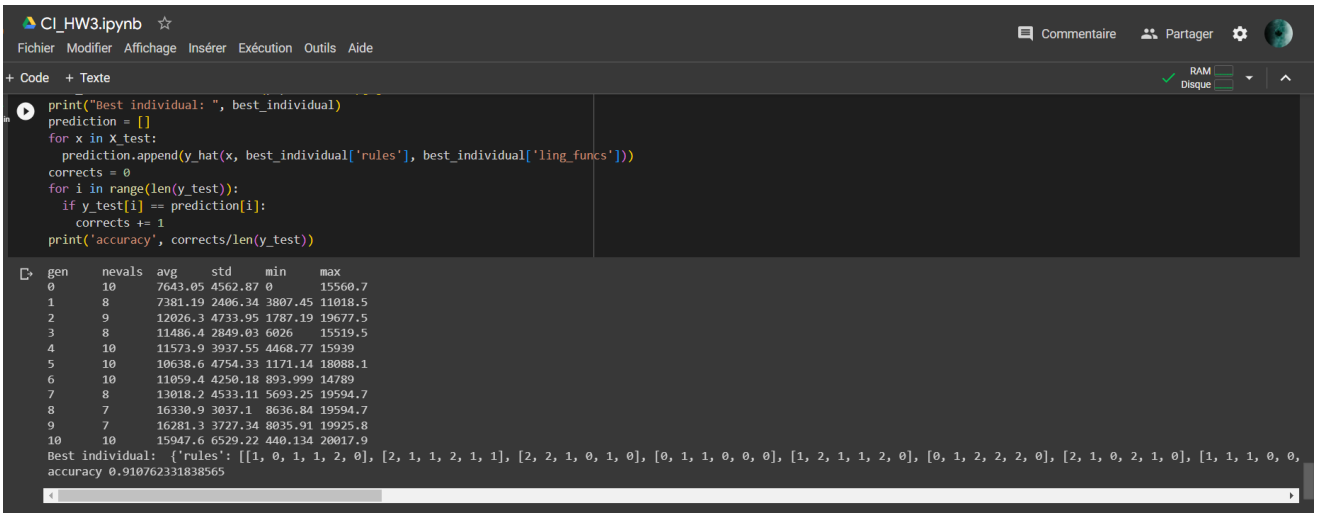
accuracy 0.772645739910313 9

با تحلیل نتایج بدست آمده، می‌توان به موارد زیر رسید:

- ژنوتایپ‌ها در الگوریتم پسرفت نداشته به صورت افزایشی برازندگی آنها آپدیت می‌شد.
- توابع عضویت برای مقادیر زبانی به صورت زیر می‌باشد:
  - Very very low: sigmoid
  - Very low: sigmoid
  - Low: gaussian
  - Medium: sigmoid
  - High: rect_trap
  - Very high: sigmoid
  - High: Sigmoid

با افزایش تعداد قوانین و مقادیر زبانی با ۱۰ نسل و تست ۴۰ درصد به دقت ۸۷ رسیدیم که نسبت به حالت قبلی افزایش دقت ۶ درصدی حاصل شد.

یکی از نتایج دیگر با ۲۰ قانون و تنظیمات اجرای قبلی:



یکی از نتایج دیگر با ۳۰ قانون و تنظیمات اجرای قبلی:

استفاده از روش‌های کاهش بعد (PCA) و انتخاب ویژگی‌ها (feature selection) بستگی به مجموعه داده و مشکل خاصی دارد که به آن پرداخته می شود. استفاده از این تکنیک‌ها در این پروژه باعث شد که عملکرد مدل ما با کاهش تعداد ویژگی‌ها و تمرکز بر مرتبط‌ترین آنها بهبود بخشد و از محاسبات اضافی راحت شدیم. برای بهتر شدن نتیجه‌ها شاید می‌شد که از روش‌های دیگر کاهش بعد مانند MDA و ... نیز استفاده کنیم تا تاثیر دیگر روش‌هارا دیده و بهترین نتیجه را انتخاب کنیم.