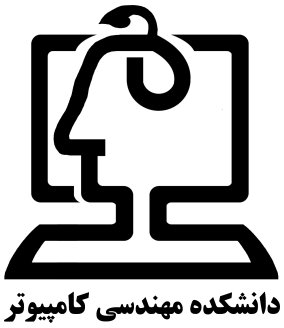
به نام خدا



دانشگاه صنعتی امیرکبیر
دانشکده مهندسی کامپیوتر

استاد درس: دکتر نیک‌آبادی

مرداد ۱۴۰۱

درس مدل‌های گرافی احتمالی

سروش مهدی
شماره دانشجویی: ۹۹۱۳۱۰۵۰

گزارش پروژه

فهرست مطالب

۱ مقدمه

در این پروژه فرض‌های مختلفی در نظر گرفته شده که در مورد همه بخش‌ها صادق است مگر ان که صریحا چیز دیگری گفته شود.

اولا به دلیل کمبود داده ها نسبت به حالات مختلف ویژگی ها(صرفا ویژگی‌های یک تا پنج منظور است)تمامی ویژگی ها به شرط متغیر هدف، دو به دو مستقل در نظر گرفته شده اند.. نتیجه این مستقل بودن این است که احتمال توام ویژگی ها به شرط متغیر هدف برابر با ضرب احتمال های تکی ویژگی ها به شرط متغیر هدف میباشد.
از این فرض درمحاسبه ماتریس emission استفاده شده.

دوما با توجه به مستقل بودن ویژگی ها برای حل کردن مشکل مقادیر null ویژگی‌ها از رابطه زیر استفاده شده :

$$\mathrm{P(f1\ ,\ f2\ ,\ f3\ ,\ f4\ ,\ f5{=}null} \mid \mathrm{target)} = \mathrm{P(f1\ ,\ f2\ ,\ f3\ ,\ f4} \mid \mathrm{target)}$$

یعنی با توجه به مستقل بودن ویژگی‌ها میدانیم احتمال توام برابر با ضرب احتمال های تکی هست. از طرفی اگر مقدار یک ویژگی null باشد مانند این است که برای این دسته از ویژگی ها فقط احتمال توام بقیه ویژگی هایی که مقدار ان ها مشخص هست را حساب کنیم. یعنی مقدار ویژگی null را نمیدانیم و از روی بقیه ویژگی ها ماتریس emission را محاسبه میکنیم.

در رابطه با ویژگی های زمینه ای نیز ان ها را از ویژگی های معمولی مستقل در نظر گرفته ایم که با توجه به داده‌ها فرض منطقی ای میباشد چون به ازای هر سمپل، به ازای ویژگی های مختلف ویژگی های زمینه ای یکسان داریم. همچنین به دلیل کمبود داده‌ها ویژگی های زمینه را نیز از یکدیگر مستقل در نظر گرفته ایم و احتمال توام ان ها مشابه احتمال توام ویژگی های معمولی حساب میشود.

تمامی کد ها به همراه توضیحات درباره کد ها و کد مورد نیاز برای تست در لینک زیر قرار دارند.
colab-file
این کد تماما به صورتی پیاده سازی شده که قابلیت توسعه پذیری داشته باشد. همچنین در اکثر توابع تا جایی که امکان ان بوده به جای دستور حلقه از عملیات ماتریسی برای بهبود عملکرد کد استفاده شده است.
در پیاده‌سازی برای حل مشکل کوچک شدن ضرب احتمال‌ها در تمامی ماتریس‌ها از دقت float128 پیاده‌سازی شده در پکیج نامپای استفاده شده است که برای اجرا شدن بدون مشکل کد باید روی کامپیوتر ۶۴بیتی اجرا شود و البته روی گوگل کلب نیز به درستی اجرا میشود.
نکته دیگر این که با صفر شدن برخی مقادیر ماتریس emission عملکرد الگوریتم افت میکرد برای حل این مشکل مقدار اپسیلون به تمام مقادیر این ماتریس اضافه کردم

پیشنهاد میشود در زمان تست بخش مربوط به تست الگوریتم EM را اجرا نکنید چون اجرا شدن ان زمان زیادی میبرد.

برای تست تمامی الگوریتم ها از روش leave one out cross validation استفاده شده است.
در مورد اکثر توابع در داخل فایل گوگل کلب به صورت کامنت توضیح مختصری داده شده که ورودی و خروجی به چه صورت هست.

تمامی الگوریتم ها برای اجرا انتظار دارند که یک پوشه به اسم data در کنار کد باشد که دقیقا مشابه فایل

داده ها فرمت بندی شده یعنی در داخل این پوشه فایل مربوط به داده های زمینه ای هست و یک پوشه دیگر به نام samlpes که نمونه های مختلف با همان فرمت نام دهی در داخل ان قرار دارند. برای بخش تست مدل ها نیز دقیقا داده ها را با همین فرمت باید کنار کد قرار دهیم علاوه بر این که داده های تست را نیز باید در پوشه samples قرار دهیم که با فرمت زیر نام گذاری شده اند.
test_id.csv

۲ الگوریتم forward backward

توابع مختلفی در این بخش پیاده شده اند که در بخش‌های بعدی نیز از انها استفاده شده.

۱.۲ توابع محاسبه ماتریس های M , pi

برای محاسبه ماتریس pi تابع comp-pi به ازای تمام سمپل ها مقادیر متغیر هدف در قدم اول را شمارش میکند و سپس این مقادیر را با تقسیم بر جمع ان ها نرمال میکند. برای محاسبه ماتریس transition نیز منطقی مشابه تابع قبلی در تابع comp-transitions اعمال میکنیم. به این صورت که به ازای سمپل های مختلف برای هر گام به جز گام اول مقدار متغیر های هدف ان گام و گام قبلی را بررسی میکنیم و برای حالت های مختلف شمارش میکنیم و در اخر نرمال میکنیم.

۲.۲ تابع emission

برای پیاده سازی این تابع ابتدا یک تابع کمکی به نام ind-feature-probs پیاده سازی شده که برای هر ویژگی مقادیر مختلف ان را در تمام سمپل ها به ازای متغیر های هدف مختلف شمارش میکند و در اخر این مقادیر را نرمال میکند .
سپس از این تابع برای پیاده سازی تابع emission استفاده شده به این صورت که برای هر گام در سمپل ورودی مقدار احتمال مربوطه به صورت ضرب احتمال ویژگی‌های مختلف به شرط متغیر هدف محاسبه میشود.

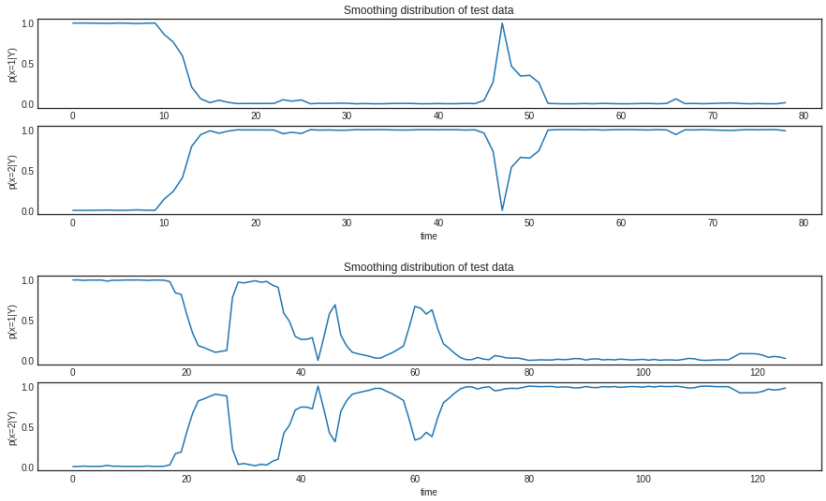
۳.۲ توابع alpha recursion, beta recursion, gamma

این توابع طبق فرمول‌های منبع زیر و با استفاده از برنامه نویسی پویا پیاده سازی شده اند که نسبت به حالت بازگشتی سرعت بیشتری دارد
Martin. H. James Jurafsky Daniel Processing. Language and Speech

۴.۲ نتایج

نتایج این بخش با اجرای تابع LOOCV بدست می اید این تابع برای الگوریتم مورد نظر leave one out cross validation را اجرا میکند و به ازای هر تکرار دقت روی سمپل تست و نمودار gamma مربوط به ان را چاپ میکند. در نهایت نیز دقت میانگین همه تکرار ها را چاپ میکند. همه این نتایج در فایل کلب در بخش leave one out cross validation on forward-backward and viterbi algorithms قابل مشاهده هستند. تمامی الگوریتم های مربوط به اجرا انتظار دارند فرمت دیتا ها دقیقا مشابه دیتاهای داده شده باشند. یعنی یک پوشه data کنار کد داریم که در ان فایل مربوط به ویژگی های زمینه های با همان نام و یک پوشه samples قرار دارد که نمونه های مختلف با فرمت نام داده شده در این پوشه قرار دارند. در زیر دو نمونه از نمودار های گاما را میبینیم: دقت مربوط به سمپل یک در حالتی که این سمپل به عنوان تست استفاده

شود برابر ۹۱ درصد و دقت مربوط به سمپل دو وقتی که به عنوان تست در نظر گرفته شود ۸۹ درصد میباشد. منظور از دقت، دقت دسته بندی است. در نهایت دقت میانگین نیز برابر ۸۳ درصد میباشد.



شکل ۱: نمودار گاما به ازای سمپل یک در بالا و سمپل دو در پایین

۳ الگوریتم EM

برای پیاده سازی این بخش به دلیل این که بیش از یک سمپل داریم از توضیحات صفحه ویکیپدیای الگوریتم baum-welch استفاده شده است

۱.۳ pair marginals

برای پیاده سازی این تابع از همان منبع بخش مربوط به alpha-recursion استفاده شده است

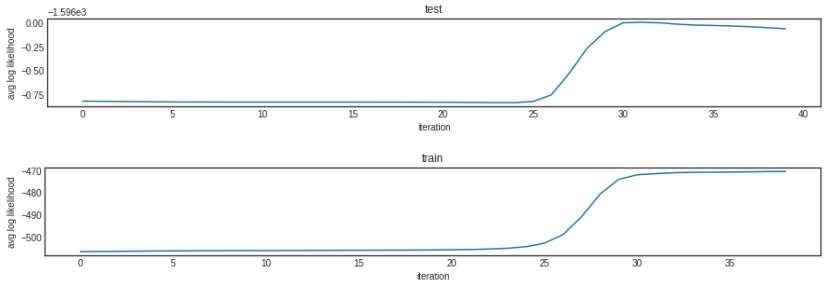
۲.۳ تابع EM

برای پیاده سازی این تابع ابتدا دو تابع کمکی MStepEmission ، MStepEmissionTest پیاده سازی شده اند که برای محاسبه ماتریس های emission مربوط به داده های اموزش و ازمون استفاده میشوند.
سپس در تابع EM در هر تکرار ابتدا برای هر سمپل E-step را انجام میدهیم و سپس با استفاده از گاما و زتای مربوط به همه سمپل ها و با توجه به رابطه صفحه ویکیپدیا پارامتر های HMM را اپدیت میکنیم.

۳.۳ نتایج

نتایج مربوط به این بخش در بخش leave one out cross validation on EM algorithm امده اند

برای بررسی این الگوریتم نیز از روش LOOCV استفاده میکنیم که به ازای هر سمپل که به عنوان تست انتخاب میشود دقت مربوط به ان سمپل ، توزیع گامای مربوط به ان سمپل و log-liklihood روی داده های تست و اموزش را چاپ میکند.
در این تابع از پارامترهای بدست امده توسط EM برای بدست اوردن گاما استفاده میشود این نتایج به ازای تمامی سمپل ها در فایل کلب قابل مشاهده اند. مقدار دقت میانگین در نهایت برابر با ۷۳ درصد میشود.
برای سمپل شماره یک به عنوان تست مقادیر log-likelihood به صورت زیر هست



شکل ۲: مقادیر لاگ لایکلیهود به ازای تکرار های مختلف الگوریتم روی مجموعه تست در بالا و اموزش در پایین

۴.۳ تحلیل

همانطور که مشاهده میشود مقدار log-likelihood با گذشت زمان روی مجموعه اموزش بیشتر میشود که این همان هدف الگوریتم EM هست که احتمال دیده شده دنباله فیچر ها را بیشینه کند
در مورد مجموعه تست نیز ابتدا این مقدار افزایش میابد اما سپس در یک نقطه شروع به کاهش میکند که میتواند نشان دهنده بیش برازش الگوریتم باشد. در واقع این نمودار به ما میگوید باید برای تکرار کمتری الگوریتم را اجرا میکردیم.

۴ الگوریتم viterbi

نتایج مربوط به این بخش در بخش leave one out cross validation on forward-backward and viterbi algorithm امده اند

هدف این الگوریتم این هست که با داشتن پارامتر های HMM و با دیدن یک دنباله از مشاهدات دنباله متغیرهای پنهانی را بدست اورد که بیشترین احتمال را دارد.
شبه کد این الگوریتم به صورت زیر هست

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  backpointer[s,1] $\leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
bestpath $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

شکل ۳: شبه کد ویتربی

این الگوریتم از گام اول شروع میکند و در هر گام مسیر با بیشترین احتمال برای رسیدن به استیت های مخفی مختلف در گام حال حاضر را ذخیره میکند. سپس در گام بعد از این احتمال های ذخیره شده برای بدست اوردن محتمل ترین مسیر برای رسیدن به ان گام استفاده میکند
برای بررسی این روش نیز از LOOCV استفاده شده که نتایج ان در فایل کلب قابل مشاهده است. میانگین دقت دسته بندی برای این روش برابر ۸۲ درصد میباشد
مشاهده میشود که نتایج این الگوریتم نزدیک به الگوریتم forward-backward میباشد.
مشاهده میشود هر دو این الگوریتم ها در مورد نمونه داده های ۱۶ و ۲۱ و ۲۳ عملکرد نامناسبی دارند.
در مورد نمونه داده ۲۳ چون نمونه با متغیر پنهان ۲ شروع شده باعث میشود که احتمال اولیه یعنی pi بسیار کم باشد که در این حالت برابر صفر میشود چون خود این نمونه در نمونه های اموزش نیست و در ادامه هر دو الگوریتم را دچار مشکل میکند.
در مورد نمونه داده ۲۱ نیز چون متغیر پنهان فرکانس تغییرات بالایی دارد احتمالا هر دو الگوریتم دچار مشکل میشوند.

۵ متغیر های زمینه ای

نتایج مربوط به این بخش در بخش contextual data امده اند

برای استفاده از متغیر های زمینه احتمال هر مجموعه از متغیر های زمینه ای به شرط متغیر هدف با توجه به فرض گفته شده در مقدمه محاسبه میشود و برای هر سمپل احتمال مربوط به متغیر های زمینه ای ان به ازای متغیرهای هدف مختلف در ماتریس emission ان سمپل ضرب میشوند و سپس از الگوریتم forward-backward استفاده میشود.
در واقع میتوان اینگونه فرض کرد که برای هر سمپل متغیرهای زمینه ای ان را نیز به بقیه ویژگی ها اضافه میکنیم به این صورت که این متغیر ها را برای یک سمپل به ازای تمامی گام ها کنار بقیه ویژگی ها تکرار میکنیم و سپس الگوریتم forward-backward را مانند بخش دوم اجرا میکنیم
برای بررسی این بخش نیز از LOOCV استفاده میشود که نتایج مربوط به ان در فایل کلب قابل مشاهده است. در نهایت میانگین دقت برای این روش ۷۸ درصد میباشد. مشاهده میشود که استفاده از این ویژگی دقت را نسبت به دقت بدست امده در بخش دوم یعنی ۸۳ درصد کاهش داده است

۶ ویژگی پنجم

نتایج مربوط به این بخش در بخش LOOCV on forward-backward algorithm without feature 5 امده اند

با حذف ویژگی پنجم یک مشکل پیش می امد که برخی از گام ها همه ویژگی هایشان null میشد برای رفع این مشکل در این بخش و بخش بعد همه مقادیر null را با صفر جایگزین کردم که با توجه به ناپیوسته بودن ویژگی ها این حالت نیز میتواند به عنوان یک حالت دیگر از ویژگی ها یادگیری شود. یعنی انتظار داریم الگوریتم یاد بگیرد که احتمال null بودن یک ویژگی به شرط متغیر پنهان چیست.

در ادامه از ۴ ویژگی دیگر برای اجرای الگوریتم forward-backward استفاده کردم . برای بررسی این روش نیز از LOOCV استفاده کردم که نتایج ان در فایل کلب قابل مشاهده است . میانگین دقت برای این حالت برابر ۸۲ درصد هست که نشان میدهد وجود متغیر پنجم در دسته بندی به ما کمک میکرد چون دقت کمتر شده.

۷ حالات دیگر

نتایج مربوط به این بخش در بخش different combination of features امده اند

برای مدل های بیشتر من با استفاده از ایده بخش قبل یعنی صفر کردن متغیر های null از تمامی ترکیب های ممکن ویژگی ها برای اموزش مدل استفاده کردم.
یعنی دقیقا طبق بخش قبل عمل کردم با این تفاوت که در هر مدل ترکیب ویژگی های استفاده شده متفاوت هست

f1 , f2 , f3 , f4 , f5
f1 , f2 , f3 , f5
f1 , f2 , f4 , f5

در این حالت نیز با پنج ویژگی به دقت ۸۳ درصد رسیدم که مشابه دقت بدست امده در بخش دوم هست. این موضوع نشان میدهد که دو روش برخورد با متغیر های null که در این بخش (صفر کردن این متغیر ها) و بخش دوم (در نظر نگرفتن این ویژگی‌ها) استفاده شده عملکرد مشابهی در نتیجه نهایی دارند. البته نتیجه این حالت حدود ۰/۰۳ درصد از روش قبلی بهتر هست

نتیجه اجرای LOOCV برای تمامی ترکیب های ویژگی ها درفایل کلب قابل مشاهده هست.
مشاهده مشود ترکیب ویژگی های یک و دو چهار و پنج عملکرد بهتری از حالتی دارد که همه ویژگی ها را استفاده کنیم.

۸ تست

برای بررسی بهترین ۳ مدل به دست امده در بخش قبل میتوانید از تابع TEST استفاده کنید که ورودی ان لیستی از اعداد int هست که نشان دهنده ایدی فایل های تست میباشد انتظار داریم این فایل ها نیز در مسیر همان سمپل های اموزشی باشند. توضیحات بیشتر در مورد نحوه قرارگیری نمونه های تست در مقدمه امده است. همچنین انتظار میرود ستون ها دقیقا به همان شکل ستون های داده های اموزش ناگذاری شده باشند و ستون target را نداشته باشیم.
نتایج نیز در مسیر کد با فرمت اسم زیر ذخیره میشوند.

output_id_model model id.csv
که در ان id همان id فایل تست هست که همان شماره ای هست که بعد از _test در نام فایل های تست قرار میگرد.
همچنین model id نیز نشان دهنده مدل استفاده شده میباشد. که به صورت زیر مشخص میشود.

1 : (f1 , f2 , f3 , f4 , f5)
2 : (f1 , f2 , f4 , f5)
3 : (f1 , f2 , f3 , f5)
فایل های نتیجه ذخیره شده دارای یک ستون به نام pred هستند که پیش بینی برای هر گام زمانی در این ستون ذخیره شده است.