

بسم الله الرحمن الرحيم

پروژه ۲ درس هوش مصنوعی
دکتر فدایی و دکتر یعقوب زاده

مهدی وجهی

۸۱۰۱۰۱۵۵۸

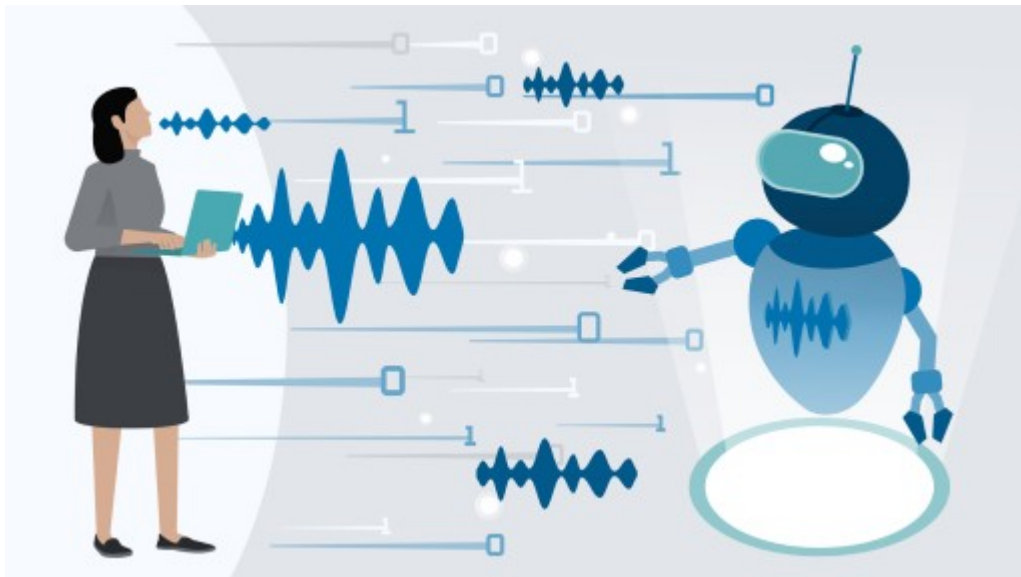
فهرست

4.....	مقدمه
5.....	پیش پردازش و استخراج ویژگی ها
5.....	قطعه بندی داده ها
5.....	ویژگی های گوناگون فایل های صوتی
5.....	MFCC
5.....	Zero crossing rate
5.....	Mel-spectrogram
6.....	Chroma features
6.....	رسم نقشه حرارتی برای داده ها
6.....	حساسیت ویژگی های MFCC
6.....	موارد نامناسب استفاده از MFCC
7.....	همپوشانی فریم های MFCC
7.....	استفاده از ۱۳ ضریب ابتدایی MFCC
7.....	آشنایی با HMM
7.....	State ها و Observation ها
7.....	First-order HMM
8.....	کاربرد های HMM
8.....	مزایا و معایب HMM
8.....	انواع HMM
9.....	پیاده سازی
9.....	پیاده سازی با استفاده از کتابخانه
9.....	تقسیم داده های آموزشی و آزمایشی
9.....	آموزش مدل
10.....	آزمایش مدل
10.....	ارزیابی مدل پیش بینی عدد
11.....	ارزیابی مدل پیش بینی گوینده
11.....	پیاده سازی از ابتدا
11.....	تابع forward
12.....	تابع backward
13.....	تابع state_likelihood
13.....	تابع em_step
14.....	ارزیابی مدل پیش بینی عددی

15.....	ارزیابی مدل پیش بینی گوینده
16.....	تحلیل و ارزیابی
16.....	شناخت معیار ها
16.....	Accuracy
16.....	Precision
16.....	Recall
16.....	F1
16.....	چالش های محاسبه معیار ها در مدل های multi-class
17.....	نحوه ارزیابی مدل توسط هر معیار
17.....	تفاوت و معایب Precision و Recall
18.....	نحوه میانگین گیری در F1
18.....	بررسی تفاوت مدل ها
19.....	نتیجه گیری
20.....	روش های بهبود و توسعه
21.....	منابع

مقدمه

هدف از انجام این پروژه ارائه مدل hmm است که بتواند عدد و گوینده صوت را تشخیص دهد. الگوریتم های hmm برای تشخیص یک وضعیت پنهان با استفاده از شواهد در توالی زمان کاربرد دارد و از این رو برای پردازش صوت و گفتار مفید است. همچنین ما برای تحلیل صوت از mfcc استفاده می کنیم که در آن طیف های صوتی متناسب با شنوایی انسان تبدیل به عدد می شود و برای این گونه تحلیل ها مفید است در ادامه نیز با آن بیشتر آشنا می شویم.



پیش پردازش و استخراج ویژگی ها

قطعه بندی داده ها

قطعه بندی داده به وسیله الگوریتم هایی انجام می شود که در آن قسمت هایی که از نظر معنایی در صدا به یکدیگر نزدیک هستند را جدا می کند و در یک دسته قرار می دهد. مثلاً این قسمت بندی می تواند پاراگرافی از متن خوانده شده باشد. با توجه که در این پروژه فایل های ما صرفاً تک کلمه ای هستند و کلاً یک کلمه را می گویند قطعه بندی داده در این پروژه کاربردی نیست.

ویژگی های گوناگون فایل های صوتی

MFCC

این ویژگی اطلاعاتی درباره طیف های فرکانسی به ما می دهد. تفاوت اصلی آن با Spectrograms این است که در آن ها دسته های فرکانس ها با هرتز یکسانی جدا میشوند ولی در این ویژگی دسته های فرکانس ها با الگوریتمی مشابه شنوایی انسان جدا می شود و این موضوع برای صوت هایی که گفتار در آنها است مفید می باشد.

Zero crossing rate

همانطور که از نام این ویژگی پیداست تعداد برخورد های صوت را به محور صفر نشان می دهد به زبانی دیگر تعداد باری که صدا از بازه مثبت به منفی می رود و بالعکس. این موضوع می تواند تشخیص فایل های بدون صدا و خالی و همچنین تفکیک کلمات و بخش های آن از یکدیگر به ما کمک کند. به طور دقیق تر در این پروژه می توان صدا های خالی را با آن حذف کرد و اعدادی که اسامی آنها چند بخشی است را از یکدیگر تفکیک کرد.

Mel-spectrogram

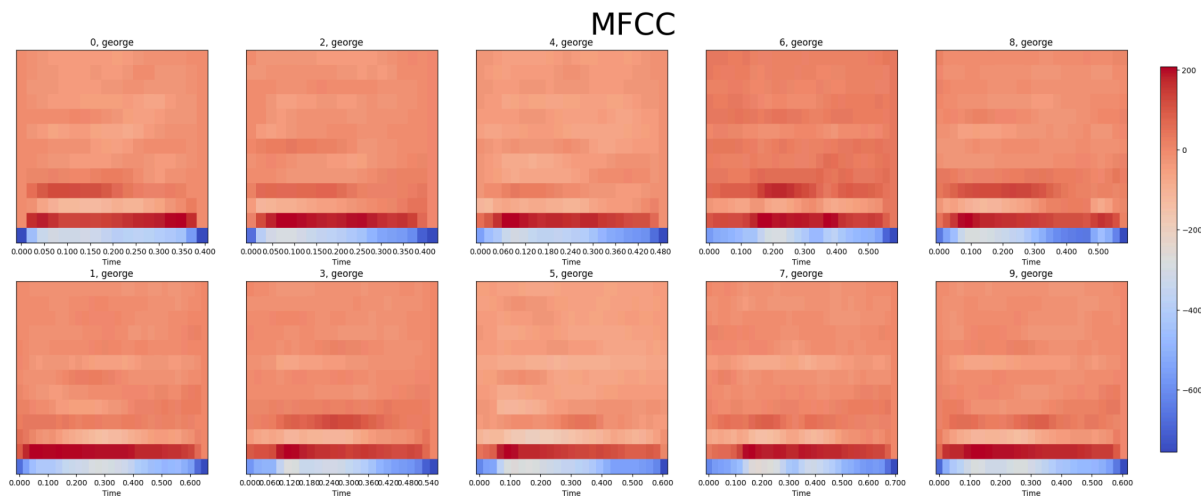
این ویژگی نیز اطلاعاتی درباره طیف های فرکانسی می دهد تفاوت آن با طیف فرکانسی معمولی در ۲ چیز است. مقیاس مل و مقیاس دسی بل. مقیاس مل یک مقیاس از نت های موسیقی است که به گونه ای طراحی شده تا درک انسان از فاصله های صوتی را لحاظ کند. مقیاس دسی بل به معنی سکوت است با افزایش ۱۰ دسی بل صدا ۱۰ برابر می شود و ۲۰ دسی بل ۱۰۰ برابر و همینطور به صورت لگاریتمی افزایش پیدا می کند. این موضوع به درک بهتر ما از صوت منجر می شود.

Chroma features

این ویژگی در واقع سعی دارد صوت را تبدیل به نوت های موسیقی کند و همچنین شدت آن نوت ها را نیز نشان دهد. این موضوع در تحلیل موسیقی می تواند بسیار مفید و کاربردی باشد اما احتمالا در این پروژه مفید نخواهد بود. البته که شاید برای تشخیص شخص گوینده بتوان از آن استفاده کرد.

رسم نقشه حرارتی برای داده ها

در این بخش ابتدا فایل ها را می خوانیم. پوشه همراه پروژه حاوی گفتار ۵۰ باره هر فرد برای هر عدد است و ما در این پوشه ۶ نفر و اعداد ۰ تا ۹ را داریم یعنی ۳ هزار صوت در این پروژه در اختیار ما قرار دارد. سپس با کتابخانه مربوطه mfcc ها را حساب می کنیم و در یک دیتا فریم قرار می دهیم سپس از هر کدام از اعداد یک نمونه نقشه حرارتی می کشیم.



حساسیت ویژگی های MFCC

ویژگی های mfcc حساس به نویز ها هستند که می تواند کارکرد آن را به شدت کاهش دهد. برای حل این موضوع سعی شده تا به جای استفاده از الگوریتم لگاریتمی از الگوریتم ترکیبی دیگری استفاده شود که این مشکل را برطرف کند.

موارد نامناسب استفاده از MFCC

بله به عنوان مثال وقتی سر و صدای محیط زیاد است و در فایل نویز زیادی داریم با توجه به مواردی که بالاتر توضیح دادیم این ویژگی مفید نیست.

همپوشانی فریم های MFCC

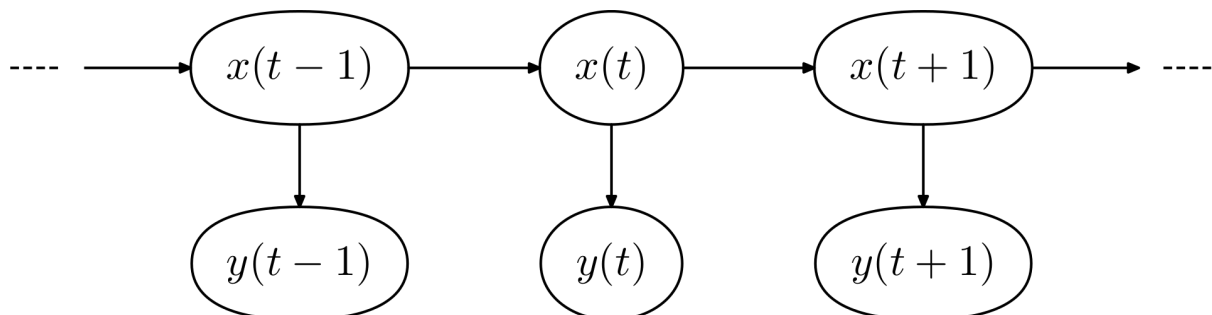
به این دلیل که اطلاعات موجود در مرز فریم ها حفظ شود این موضوع باعث کاهش تاثیر نقطه مرز بندی فریم ها می شود و باعث می شود درکی از تغییرات تدریجی صوت پیدا کنیم این موضوع به خصوص در تحلیل گفتار که پیوستگی مهم است اهمیت زیادی پیدا می کند.

استفاده از ۱۳ ضریب ابتدایی MFCC

مهمترین دلیل این موضوع کاهش پیچیدگی مدل و افزایش سرعت محاسبات است زیرا که بیشتر اطلاعات در فرکانس های پایین قرار دارند و بیشتر اطلاعاتی که برای تمییز صدا ها استفاده می شود در آن بازه قرار دارد. همچنین این موضوع روی overfit نشدن مدل روی داده های آموزشی نیز موثر است.

آشنایی با HMM

State ها و Observation ها



Observation ها در واقع آن داده هایی هستند که ما دریافت می کنیم و با آن باید state که در آن هستیم را بیابیم این که الان در چه state هستیم که این Observation را دریافت کردیم با استفاده از احتمال Emission محاسبه می شود. در این پروژه Observation ها در واقع فریم های MFCC ما هستند. State ها در واقع آن بخش پنهان است که ما نیاز داریم با استفاده از Observation آنها را تشخیص دهیم. در تصویر بالا x ها state هستند و y ها Observation ها. در این پروژه state ها در واقع لحن بیان آن قسمت از عدد خاص توسط گوینده خاص است.

First-order HMM

مدل مارکوفی که در کلاس آموزش داده شده و در این تمرین باید از آن استفاده کنیم مدل درجه اول است که ما در آن احتمال Transition تنها به state قبلی وابسته است. مدل های درجه ۲ یا بیشتر نیز وجود دارد که همانطور که از اسم آنها می توان فهمید احتمال انتقال ما به ۲ state قبلی یا بیشتر وابسته است. واضح

است که مدل درجه ۱ پیچیدگی محاسباتی کمتری به سیستم تحمیل می کند و پیچیدگی کمتری دارد. در مواردی که پیچیدگی زمانی طولانی تری بین داده ها وجود دارد استفاده از مدل های پیچیده تر می تواند مفید باشد. در مدل درجه اول ما حلقه و پرش نداریم اما در درجه های بالاتر این موارد وجود دارد. در نهایت ما باید نسبت به نوع داده ها درجه مدل خودمان را انتخاب کنیم استفاده از مدلی با پیچیدگی بیش از حد می تواند باعث overfit شدن مدل موثر باشد و تاثیر منفی بگذارد از آن طرف مدلی با پیچیدگی خیلی کم و ساده دقت کافی را برای ما فراهم نمی کند.

کاربرد های HMM

به طور کلی مدل های HMM زمانی کاربرد دارد که با استفاده از توالی شواهدی که می توانیم به دست بیاوریم توالی وضعیتی که به صورت مستقیم نمی توانیم تشخیص دهیم (پنهان است) را تشخیص دهیم. این مدل ها در زمینه هایی مانند پردازش زبان طبیعی، تشخیص گفتار، بیوانفورماتیک، پیش بینی دنباله های ژنتیکی، و تحلیل داده های مالی کاربرد دارند. در واقع با این روش می توان تغییرات زمانی و توالی و وابستگی داده ها را مدل سازی کرد.

مزایا و معایب HMM

از مزایای این مدل می توان به موارد زیر اشاره کرد:

- **مدل سازی وابستگی به زمان داده ها:** همانطور که بالا هم توضیح داده شد این مدل مناسب است برای مدل سازی وابستگی به زمان داده ها. مثلا در تحلیل صوت گفتار یا دنباله ای از کلمات در یک جمله.
 - **انعطاف پذیری:** این مدل قادر به پیاده سازی و مدل کردن موارد پیچیده است. همچنین احتمالاتی بودن این مدل به این موضوع کمک میکند.
 - **کار با داده های ناقص:** در این مدل می توان از داده هایی که برخی از آنها از دست رفته یا جمع آوری نشده نیز استفاده کرد.
- برخی از معایب این مدل به شرح زیر است:
- **عدم مدل سازی مواردی با فاصله زیاد:** در این مدلسازی ما شرط استقلال را فرض می کنیم ولی این شرایط همیشه برقرار نیست و در مدل های پیچیده و دارای وابستگی طولانی مناسب نمی باشد.
 - **حساسیت به پارامتر ها:** این مدل وابستگی بالایی به پارامتر های خود دارد و در صورتی که درست تنظیم نشوند کارکرد مدل متخل می شود.

انواع HMM

- **مدل رتبه اول (first-order):** این مدل همان مدل تدریس شده در کلاس است و همانی است که ما در این پروژه استفاده می کنیم و بالاتر راجب به آن صحبت کردیم.
- **مدل پیوسته:** در این مدل برخلاف مدل هایی که تا الان داشتیم مشاهدات را به صورت پیوسته در

نظر می گیریم.

- **مدل تماماً متصل (Fully Connected):** در این مدل سازی تمامی state ها به یکدیگر متصل هستند و این موضوع حالتی کلی تر به ما ارائه می کند.
- **HSMM:** این مدل مشابه HMM است با این تفاوت که مدت زمان ماندن در یک state در احتمالات ما موثر است.

پیاده سازی

پیاده سازی با استفاده از کتابخانه

تقسیم داده های آموزشی و آزمایشی

برای تقسیم داده به این صورت عمل می کنیم که طبق نسبتی که مشخص کردیم از هر جفت گوینده، عدد آنها را در لیست های مربوطه قرار می دهیم. سپس با توجه به این که بر چه اساس مدل سازی می کنیم لیست ها (گوینده، عدد) را با یکدیگر ادغام می کنیم.

```
def create_test_and_train_data(df, par_main
, par_side, test_percent=TEST_SIZE):
    '''[par_main][par_side] = mfcc'''
    training_data = dict()
    testing_data = dict()
    for i in df[par_main].unique():
        train_tmp = []
        test_tmp = []
        for j in df[par_side].unique():
            filtered_data = np.array(df[(df[par_main] == i) & (df[par_side]
== j)][['mfcc']])
            test_size = int(len(filtered_data) * test_percent)
            train_tmp.append(filtered_data[:-test_size])
            test_tmp.append(filtered_data[-test_size:])
        training_data[i] = np.concatenate(train_tmp)
        testing_data[i] = np.concatenate(test_tmp)
    return training_data, testing_data
```

آموزش مدل

برای آموزش مدل به آسانی داده های هر دسته را به یک مدل جدید می دهیم و در آخر مدل ها را ذخیره می کنیم. مقادیر آرگومان تابع بر اساس جست و جو در اینترنت به دست آمده.

```
def train_data(data:dict, n_com=3) -> dict:
    hmms = dict()
    for num in data.keys():
```

```

data_len = [len(i) for i in data[num]]
concatenate_data = np.concatenate(np.array(data[num]))
new_hmm = hmm.GaussianHMM(n_components=n_com, covariance_type='diag',
n_iter=1000)
new_hmm.fit(concatenate_data, lengths=data_len)
hmms[num] = new_hmm
return hmms

```

آزمایش مدل

برای این که داده های آزمایشی را به هر کدام از مدل ها می دهیم سپس هر کدام که مقداری بالاتری برگرداند به عنوان نتیجه در نظر میگیریم.

```

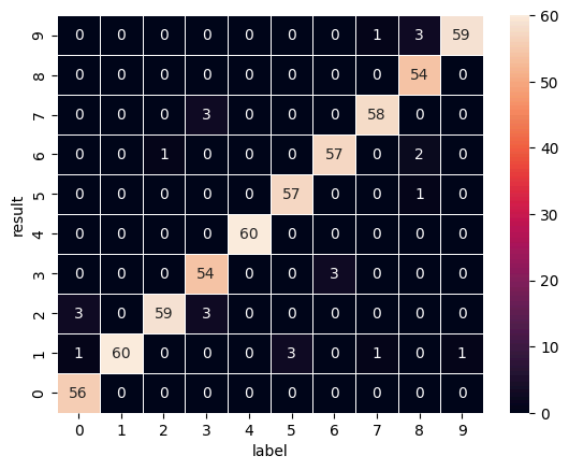
def test_hmm(data:dict, hmms:dict) -> dict:
    result = dict()
    for i in data.keys():
        result[i] = []
        for j in data[i]:
            tmp = dict()
            for k in hmms.keys():
                tmp[k] = hmms[k].score(j)
            q = max(tmp, key=lambda k: tmp[k])
            result[i].append(q)
    return result

```

ارزیابی مدل پیش بینی عدد

نتایج به شرح زیر است:

	TP	FP	FN	f1	precision	recall
0	56	0	4	0.965517	1.000000	0.933333
1	60	6	0	0.952381	0.909091	1.000000
2	59	6	1	0.944000	0.907692	0.983333
3	54	3	6	0.923077	0.947368	0.900000
4	60	0	0	1.000000	1.000000	1.000000
5	57	1	3	0.966102	0.982759	0.950000
6	57	3	3	0.950000	0.950000	0.950000
7	58	3	2	0.958678	0.950820	0.966667
8	54	0	6	0.947368	1.000000	0.900000
9	59	4	1	0.959350	0.936508	0.983333

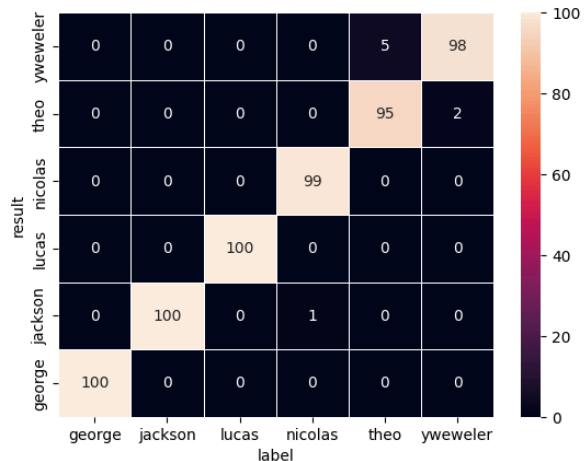


	accuracy	precision	recall	f1
macro	0.956667	0.958424	0.956667	0.957544
micro	0.956667	0.956667	0.956667	0.956667

همانطور که می بینید مدل دقت بالای ۹۵ درصد دارد (۲۰ درصد داده ها به عنوان داده آزمایشی در نظر گرفته شد) که این موضوع خوب است. بیشترین اشتباه در اعداد ۳ و ۸ است. همانطور که دیدیم معیار های مختلف مقادیر مشابهی به دست آورده اند و تفاوت زیادی ندارند.

ارزیابی مدل پیش بینی گوینده

	TP	FP	FN	f1	precision	recall
george	100	0	0	1.000000	1.000000	1.00
jackson	100	1	0	0.995025	0.990099	1.00
lucas	100	0	0	1.000000	1.000000	1.00
nicolas	99	0	1	0.994975	1.000000	0.99
theo	95	2	5	0.964467	0.979381	0.95
yweweler	98	5	2	0.965517	0.951456	0.98



	accuracy	precision	recall	f1
macro	0.986667	0.986823	0.986667	0.986745
micro	0.986667	0.986667	0.986667	0.986667

دقت در این مدل نیز بالای ۹۵ درصد است و حتی از بخش قبلی بهتر عمل می کند. پایین ترین دقت در تشخیص صدای theo است. همانطور که دیدیم معیار های مختلف مقادیر مشابهی به دست آورده اند و تفاوت زیادی ندارند.

پیاده سازی از ابتدا

تابع forward

در این تابع باید آلفا را حساب کنیم طبق اسلاید های استاد آلفا برای بار اول به شکل زیر حساب می شود:

$$\alpha_1(j) = \pi_j b_j(o_1) \quad 1 \leq j \leq N$$

یعنی برای محاسبه آلفای اول هر استتیت کافیت احتمال اولیه آن را در احتمال مشاهده داده مشاهده شده به شرط بودن در آن استتیت ضرب می کنیم. برای محاسبه این مقدار برای تک تک استتیت ها کافیت ماتریس احتمال اولیه را در ماتریس مشاهده به شرط بودن در استتیت را به صورت درایه ضرب کنیم.

```
if t == 0:
    alpha[:, 0] = self.initial_prob[:, 0] * observation_matrix[:, 0]
```

برای باقی موارد باید به صورت زیر عمل کنیم:

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T$$

برای محاسبه این احتمال باید تمام احتمالات رسیدن به این استتیت را به درست آوریم و در احتمال مشاهده رویداد مشاهده شده به شرط این استتیت ضرب کنیم. پس تا الان متوجه شدن ماتریس مشاهده رویداد به شرط بودن در استتیت خاص باید به صورت درایه درایه در ماتریس احتمال رسیدن ضرب شود. تمام احتمالات رسیدن برابر است با ضرب احتمال رسیدن به هریک از استتیت های قبلی و احتمال انتقال به استتیت مربوطه. این یک جمع است اگر بخواهیم برای تک تک استتیت ها محاسبه کنیم همان ضرب داخلی ماتریس آلفا در زمان قبلی در ماتریس انتقال است. پس داریم که:

```
alpha[:, t] = (self.transition_matrix.T @ alpha[:, t-1]) *
observation_matrix[:, t]
```

تابع backward

برای پیاده سازی تابع باید به صورت زیر عمل کنیم:

1. Initialization:

$$\beta_T(i) = 1, \quad 1 \leq i \leq N$$

2. Recursion

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad 1 \leq i \leq N, 1 \leq t < T$$

مقدار اولیه در خود کد قرار داده شده و نیازی به پیاده کردن نیست. برای پیاده سازی بخش بازگشتی کافیهست احتمال b_{t+1} را در احتمال بتا زمان $t+1$ ضرب کنیم و چون باید برای تمامی استتیت ها این کار انجام شود ماتریس آنها را ضرب درایه در درایه می کنیم و سپس برای انجام جمع مانند دفعه قبل لازم است ماتریس آلفا را در ماتریس به دست آمده ضرب داخلی کرد تا بتا زمان حال به دست بیاید.

```
beta[:, t] = self.transition_matrix @ (beta[:, t+1] *
observation_matrix[:, t+1])
```

تابع state_likelihood

در این تابع با توجه به داده هایی که تا به حال مدل آموزش داده ایم پارامتر های توزیع نرمال را تعیین می کنیم و با استفاده از آن احتمال ها emission را به دست می آوریم.

```
B[s, :] = multivariate_normal.pdf(obs.T, mean =self.mean[:,s], cov =
self.covariances[:,s,s])
```

تابع em_step

این تابع برای تعلیم مدل استفاده می شود. ابتدا باید با استفاده از تابع قبلی احتمال emission را تعیین کنیم.

```
B = self._state_likelihood(obs)
```

احتمال های آلفا و بتا هم با تابع های forward, backward به دست می آوریم.

```
log_likelihood, alpha = self._forward(B)
beta = self._backward(B)
```

طبق اسلاید ها برای محاسبه شای و گاما به صورت زیر حساب می کنیم:

E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall t \text{ and } j$$

$$\xi_t(i, j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall t, i, \text{ and } j$$

گاما با ضرب آلفا در بتا به دست می آید که در کد به صورت ضرب درایه در درایه پیاده می کنیم و در آخر نرمالایز می کنیم. برای شای نیز تنها کافیست صورت آن را حساب کنیم زیرا در آخر آنها را نرمالایز می کنیم.

```
partial_sum = alpha[:,t] @ ((beta[:,t+1] * B[:,t+1]).T *
self.transition_matrix)
partial_g = alpha[:,t] * beta[:,t]
```

چون در حلقه مقدار گاما T-1 محاسبه نمی شود خارج حلقه آن را حساب می کنیم. در آخر هم با جمع شای ها احتمال انتقال ها را حساب می کنیم.

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(i, k)}$$

```
expected_transition = self._stochasticize(xi_sum)
```

احتمال prior را که در گاما محاسبه کرده بودیم را ذخیره می کنیم.(البته بنده دقیقا متوجه این خط کد نشدم و با پرس و جو به آن را کامل کردم).

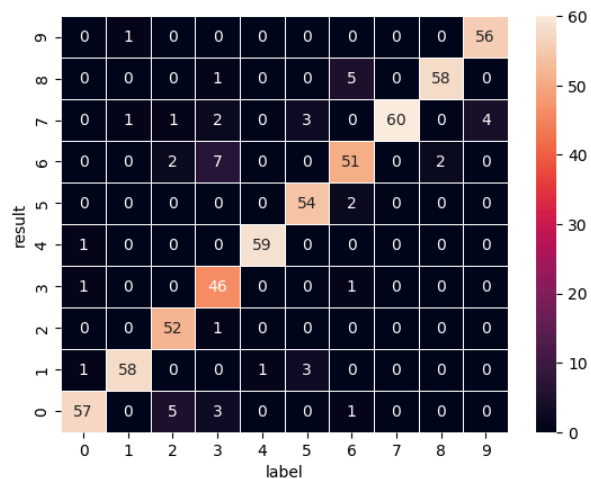
```
expected_prior = gamma[:, 0].reshape(-1,1)
```

طبق صحبتی که با همکلاسی ها شد ظاهرا خط زیر باعث اختلال در عملکرد می شد و کامنت شد.

```
self.covariances = expected_covariances
```

ارزیابی مدل پیش بینی عددی

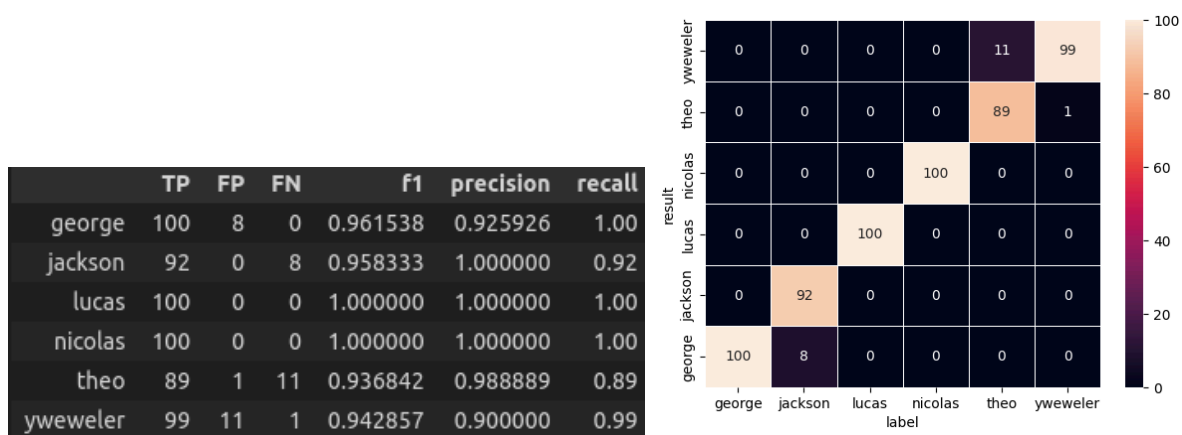
	TP	FP	FN	f1	precision	recall
0	57	9	3	0.904762	0.863636	0.950000
1	58	5	2	0.943089	0.920635	0.966667
2	52	1	8	0.920354	0.981132	0.866667
3	46	2	14	0.851852	0.958333	0.766667
4	59	1	1	0.983333	0.983333	0.983333
5	54	2	6	0.931034	0.964286	0.900000
6	51	11	9	0.836066	0.822581	0.850000
7	60	11	0	0.916031	0.845070	1.000000
8	58	6	2	0.935484	0.906250	0.966667
9	56	1	4	0.957265	0.982456	0.933333



	accuracy	precision	recall	f1
macro	0.918333	0.922771	0.918333	0.920547
micro	0.918333	0.918333	0.918333	0.918333

دقت مدل به طرز عجیبی بالا است و دقت ۹۲ درصد دارد. مدل مقداری در تشخیص ۳ ضعیف عمل می کند. همانطور که دیدیم معیار های مختلف مقادیر مشابهی به دست آورده اند و تفاوت زیادی ندارند.

ارزیابی مدل پیش بینی گوینده



	accuracy	precision	recall	f1
macro	0.966667	0.969136	0.966667	0.967900
micro	0.966667	0.966667	0.966667	0.966667

در این قسمت هم مانند قسمت قبل خوب است و دقت ۹۷ درصدی دارد. همانطور که دیدیم معیار های مختلف مقادیر مشابهی به دست آورده اند و تفاوت زیادی ندارند.

تحلیل و ارزیابی

شناخت معیار ها

Accuracy

ای معیار نسبت موارد درست تشخیص داده شده به کل را نشان می دهد.

Precision

نسبت موارد درستی با تشخیص درست به کل مواردی که درست تشخیص دادیم (لزوما درست تشخیص داده نشده اند). (T به معنای درست تشخیص داده شده و F به معنی غلط تشخیص داده شده و P به معنی این است که ورودی واقعا مثبت است و N یعنی ورودی واقعا منفی است).

$$\frac{TP}{TP+FP}$$

Recall

نسبت مواردی که درست تشخیص داده ایم نسبت به مواردی که درست هستند. (چه تشخیص داده باشیم چه نداده باشیم)

$$\frac{TP}{TP+FN}$$

F1

میانگین گیری هارمونیک از ۲ معیار بالا

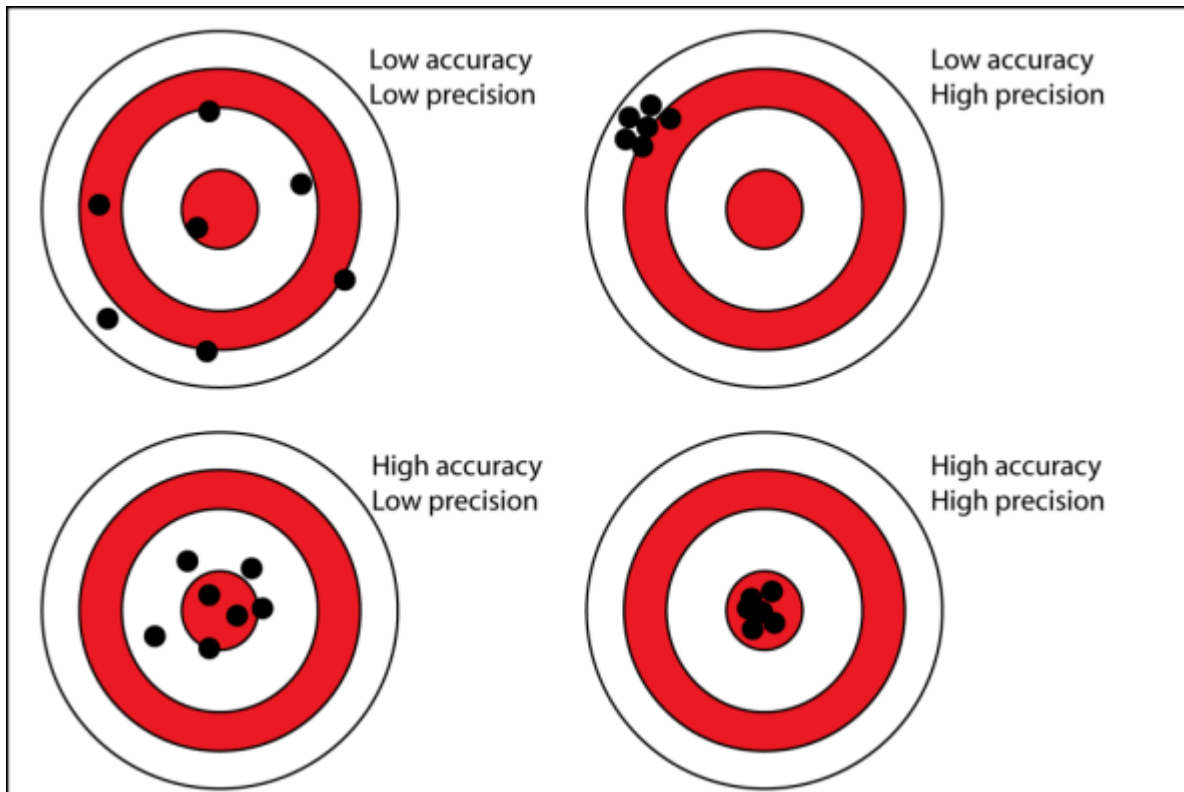
$$2 \times \frac{Precision \times Recall}{Precision + Recall}$$

چالش های محاسبه معیار ها در مدل های multi-class

به غیر از accuracy باقی معیار ها تنها برای یک دسته بندی هستند برای حل این مشکل می توان دو روش micro, macro را استفاده کرد که macro صرفا میانگین گیری معمولی می کند اما در micro صورت های کسر را جمع می زد و با جمع مقادیر مخرج تقسیم می کند. در آخر با این نتایج f1 نهایی را می توان حساب کرد.

نحوه ارزیابی مدل توسط هر معیار

در Accuracy هدف سنجش مدل به صورت کلی است و نشان می دهد که بدون در نظر گرفتن موردی مدل چه میزان تشخیص درست نسبت به کل دارد. در واقع این معیار صحت مدل را می سنجد. در Precision دقت تشخیص ها نشان داده می شود. یعنی چه میزان از آنهایی که درست تشخیص داده ایم واقعا درست بوده اند.



در Recall ما در واقع میزان بدبینی مدل خود را می سنجیم و همانطور که در ادامه توضیح می دهیم با خوشبین کردن مدل (درست تشخیص دادن همه 'p') این معیار بیشینه می شود. این معیار برای کاهش میزان سخت گیری مناسب است.

تفاوت و معایب Precision و Recall

معیار precision به تعداد موارد درست تشخیص داده نشده حساس نیست و اگر تنها معیار ما باشد برای بیشینه شدن آن مدل ما سعی می کند با انتخاب تعداد بسیار کمی مورد درست با احتمال بالا تا جای امکان حدس های مثبت غلطش را کاهش دهد و این باعث می شود تعداد حدس های منفی غلط به طور نامناسبی افزایش پیدا کند. مثلا اگر ما باید ۱۰۰ مورد را بررسی کنیم تنها مواردی که ۹۹ درصد اطمینان از درست بودن داریم را انتخاب می کنیم و به عنوان حدس مثبت اعلام می کنیم که این معیار را بیشینه کنیم اما باعث شد که به جای تشخیص حدود ۱۰ مورد مثبت تنها ۱ مورد مثبت تشخیص دهیم. معیار Recall نیز مشکلی مشابه دارم و مدل برای بیشینه کردن آن تنها کافیست که تمام آزمایش ها را مثبت جواب دهد. در مثال قبلی از ۱۰۰ مورد ۱۰۰ مورد را مثبت تشخیص می دهد و معیار بیشینه می شود.

نحوه میانگین گیری در F1

این معیار از میانگین گیری هارمونیک استفاده می کند که باعث می شود مواردی که میانگین گرفته می شود به طور یکسان در نتیجه موثر باشند و با افزایش نتها یک معیار نمی شود نتیجه را بیشینه کرد. با توجه به مواردی که در قسمت قبل توضیح داده شد برای ما مهم است که مدل به طور متناسب هر دو معیار Recall و Precision را افزایش دهد و تنها به بیشینه کردن یکی اکتفا نکند.

بررسی تفاوت مدل ها

مدل کتابخانه ای حدود ۲، ۳ درصد عملکرد بهتری دارد که خیلی قابل توجه به نظر نمی آید. اما این موضوع می تواند به این دلیل باشد که مثلا مدل کتابخانه ای، الگوریتم را گسترش داده باشد و پیچیده تر کرده باشد یا مدل به صورتی پیاده شده که به داده های پرت کمتر حساس باشد. همچنین تنظیم دقیق هایپر پارامتر ها نیز موثر است و همانطور که قبلا اشاره شد این مدل به مقادیر اولیه خود حساس است و می توان با تنظیم بهتر این مقادیر بازدهی را بالا برد.

نتیجه گیری


همانطور که مشاهده کردیم مدل های hmm برای تشخیص و تحلیل این گونه صوت ها عملکرد بسیار خوبی دارد و حتی در نسخه پیاده سازی شده نیز عملکرد درخشانی را شاهد هستیم. همچنین معیار mfcc نیز بازنمایی بسیار خوبی از صوت برای این تحلیل ارائه می دهد و می تواند مورد استفاده قرار گیرد. البته که طبیعی است که تعداد کم اعداد و گویندگان در عملکرد این مدل موثر است.

روش های بهبود و توسعه

راهکار ها و ایده های زیر برای گسترش و بهبود پروژه پیشنهاد می گردد:

- اعمال تنظیمات دقیق و مناسب برای mfcc با توجه به پروژه
- استفاده از سایر ویژگی ها برای آموزش مدل
- استفاده از انواع دیگر HMM
- استفاده از داده های بیشتر برای آموزش مدل
- پیاده سازی مدل مشابه برای زبان فارسی
- تست فرض برای بعضی ادعا های مطرح شده در گزارش
- پردازش اعداد بیش از یک رقمی

منابع

- [دسته بندی سبک های موسیقی با بایتون – راهنمای کاربردی – فرادرس – مجله](#)
- [Audio Deep Learning Made Simple - Why Mel Spectrograms perform better | Ketan Doshi Blog](#)
- [Chroma feature - Wikipedia](#)
- [Encyclopedia of Database Systems](#)
- [زنجیره و فرآیند مارکوف و مدل پنهان آن – به زبان ساده – فرادرس – مجله](#)
- [Hidden Markov Model in Machine learning](#)
- [Hidden Markov Models \(HMM\) – Machine Learning and Data Science Compendium](#)
- [Hidden Markov Model Definition | DeepAI](#)
-  [lec10-hidden_markov_models-2](#)
- [Accuracy, precision, and recall in multi-class classification](#)
- [lec5-Foundations for Inference-introduction to data science- dr. bahrak and dr. yaqubizade](#)