

# گزارش پروژه فاز دوم

## خوشبندی:

برای پیاده سازی این قسمت از کتابخانه scikit-learn موجود برای زبان python استفاده شده است. توضیحات مربوط به این کتابخانه در گزارش فاز اول آورده شده است.

داده های موجود در مجموعه داده MNIST برای داده های صفر تا ۵ موجود بوده و شامل  $x$  و  $y$  می باشد. هر داده  $x$  تشکیل شده است از یک آرایه دو بعدی ۲۸ در ۲۸ که تصویر را تشکیل می دهد.

کمینه مقدار این آرایه ها که هر پیکسل نشان دهنده شدت نور می باشد . و بیشینه مقدار ۲۵۵ می باشد. بنابراین با قرار دادن threshold ای معادل با ۱۲۷ داده ها را به صفر و یک تقسیم می کنیم.

با توجه به اینکه سرعت اجرای الگوریتم ها بر روی سیستم اینجانب بسیار کند بوده و امکان صبر کردن برای اجرای کامل را نداشتیم و به منظور قابل مقایسه بودن نتایج بدست آمده از همه الگوریتم ها الگوریتم ها را تنها با ۶۰۰۰ داده از کل داده ها اجرا کردم! دلیل پایین بودن سرعت لپ تاپ هم مشکل سوختن باتری بوده که در MacBook Pro منجر به کاهش بازدهی سیستم از نظر محاسبات و... می شود. با این حال سعی شده است تا در این گزارش نتایج برای مقایسه فراهم شوند.

## پیاده سازی Kmeans

برای خوشبندی با استفاده از الگوریتم kmeans از تابع کتابخانه scikit learn استفاده کرده ایم. برای این الگوریتم به صورت default تعداد تکرار ها برابر با ۳۰۰ در نظر گرفته شده است که آن را تغییر نداده ایم. همچنین برای مراکز از طریق سنت کردن درست پارامتر ها این الگوریتم به صورت هوشمند مقادیری را انتخاب می کند که همگرایی را سرعت ببخشد.

## پیاده سازی EM برنولی:

در این حالت با توجه به اینکه اعداد به . تا ۵ تعلق دارند ۶ کلاس مختلف برای آنها  $k=6$  در نظر می گیریم. مقدار دهی اولیه آرایه  $ni$  برای تمام کلاس ها برابر می باشد و برابر با  $1/6$  قرار داده می شود. همچنین مقدار دهی اولیه

برای آرایه  $\mu$  نیز به صورت مشخص شده در کد و با توجه به توضیحات کتاب به طورت رندم بین 0.25 و 0.75 انتخاب می شود و نرمال سازی های لازم روی آن انجام می شود. تا در نهایت مجموع برابر با یک شود. روابط استفاده شده در این قسمت همان روابط اثبات شده در تمرین آخر و موجود در فصل ۹ کتاب می باشد. تعداد تکرار های الگوریتم EM را در این حالت به دلیل پایین بودن سرعت اجرا ۷ در نظر گرفته شده است.

### پیاده سازی EM چند جمله ای:

روابط خواسته شده در زیر اثبات شده اند:

نوفم چشم ۲۰۱۶، دیجیتال، مجموعه ۳ دویستم دوره

$$\mu_K = (\mu_{K1}, \dots, \mu_{KD})$$

تعداد نمونه های مورد بررسی می باشد  $N$  مجموع  $x_{ni}$  مجموع  $x_{nj}$  مقدار مکانیکی  $x_{ni}$   $x_{nj}$  pixel

$$\sum_{n=1}^N x_{nn} = P_N \times 2N$$

$$\sum_{K=1}^K \pi_K = 1, \quad \sum_{i=1}^D \mu_{Ki} = 1$$

MPV:

$$p(x_n=k | \theta) = \frac{\prod_{i=1}^D (\mu_{Ki})^{x_{ni}}}{\sum_{j=1}^K \prod_j (\mu_{ji})^{x_{nj}}}$$

Likelihood:

$$\text{Likelihood} = \sum_{n=1}^N \sum_{K=1}^K z_{nK} \left( \sum_{j=1}^D x_{nj} \log \mu_{kj} + \log \pi_k \right) = A$$

$$\pi_K = \arg \max \sum_{n=1}^N \sum_{K=1}^K z_{nK} \left( \sum_{j=1}^D x_{nj} \log \mu_{kj} + \log \pi_k \right) + \lambda \left( 1 - \sum_{k=1}^K \pi_k \right)$$

$$\frac{\partial}{\partial \pi_K} = 0 \rightarrow \boxed{\pi_K = \frac{\sum_{n=1}^N z_{nK}}{N} = \frac{N_k}{N}}$$

$\mu_{kj}$ :

$$\mu_{kj} = \arg \max A + \sum_{m=1}^M \lambda_m \left( 1 - \sum_{j=1}^D \mu_{mj} \right)$$

$$\frac{\partial}{\partial \mu_{kj}} = 0 \rightarrow \mu_{kj} = \frac{\sum_{n=1}^N z_{nK} x_{nj}}{\sum_{i=1}^D \sum_{n=1}^N z_{nK} x_{nj}} = \frac{\sum_{n=1}^N z_{nK} x_{nj}}{\sum_{n=1}^N z_{nK} \left( \sum_{j=1}^D x_{nj} \right)}$$

$N$   $P_N \times 2N$

پیاده سازی با توجه به روابط بدست آمده در بالا صورت گرفته است.

### شرط خاتمه در EM

برای پایان کار هم می توان تعداد مشخصی از تکرار ها را برای الگوریتم در نظر گرفت و هم راه دیگر تعریف میزان خطای است که اگر در دو مرحله پشت سر هم میزان خطای یک مقدار کمتر باشد در این صورت الگوریتم متوقف می شود. می توان برای این خطایا به بیان دیگر اختلاف در گام های مختلف از تفاوت بین میانگین های توزیع کلاسها به عنوان معیار همگرایی استفاده کرد. که در زیر می بینیم:

```
if(||θ(t + 1) - θ(t)|| < ϕ)
    stop
else
    call E-Step
end
```

در پیاده سازی های صورت گرفته تعداد تکرار حلقه به عنوان شرط پایان استفاده شده است.

ارزیابی :

RandIndex :

این معیار برای اندازه گیری میزان شباهت ۲ نوع کلاسترینگ استفاده می شود که در این حالت ما از آن برای اندازه گیری میزان شباهت برچسب های هر نوع دسته بندی با برچسب های واقعی استفاده می کنیم.

این معیار هر چه به یک نزدیک تر باشد شباهت بیشتری دو دسته به هم دارند. در این قسمت به جای randIndex از تابع آماده adjusted rand index موجود در کتابخانه استفاده شده است.

همچنین برای ارزیابی از دو معیار F1 micro و F1 macro نیز استفاده شده است.

نتایج ارزیابی روش های مختلف بدون کاهش ابعاد:

### نتایج قسمت اول سوال بدون کاهش بعد

	Macro-F1	Micro-F1	adjusted RandIndex
Kmeans	0.29084698946514692	0.256017991504012	0.558033328430195
Bernouli MM	0.18817978521587961	0.21883333333333332	0.3029378037682772
Multinomial MM			

#### مقایسه :

همانطور که می بینیم برنولی عملکرد بدتری نسبت به kmeans داشته بر روی این داده ها. دلیل می تواند این باشد که با باینری کردن میزان بسیار زیادی از اطلاعات از دست می رود به دلیل اینکه داده ها اعداد دست نویس بوده اند و خود دارای خطای استنده.

### کاهش ابعاد با استفاده از : PCA

برای کاهش ابعاد از کتابخانه آماده استفاده کرده. یکی از ویژگی های کتابخانه مورد استفاده این است که می توان pc هایی را انتخاب نمود که درصدی خاص از واریانس را پوشش دهند. در زیر به ازای درصد های مختلفی از واریانس ها تعداد pc ها لازم آورده شده است :

Table 2

درصد واریانس پوشش داده شده توسط pc ها	0.8	0.825	0.85	0.875	0.9	0.925	0.95	0.975
تعداد نمونه های لازم	40	47	55	66	81	104	144	221

با توجه به جدول بالا هر چه درصد بیشتری از واریانس را پوشش می دهیم تعداد pc ها برای افزایش مقداری مشخص (درجول بالا 0.025 بیشتر می شود. که به این دلیل است که در مراحل آخر pc هایی که واریانس اندکی را پوشش می دهند مانده و هر بار اضافه می شوند. از ۰.۹۷۵ به ۰.۹۵ به ناگهان تعداد بسیار بالا رفته که نشان می

دهد همان 0.95 مقدار مناسبی است. پس تعداد متناظر با این درصد یعنی ۱۴۴ را بر می‌داریم. همچنین با محاسبه معیارهای برای تعداد مختلف pc‌ها بهترین نتیجه برای ..... تعداد pc بدست آمد.

ک مربوط به برنولی بدون کاهش ابعاد در فایل Phase2Part1BernouliMixture.py قرار دارد.

ک مربوط به bernouli همراه با کاهش ابعاد در فایل Phase2Part1BernouliPCA قرار دارد.

نتایج روش‌های مختلف به همراه کاهش ابعاد:

### نتایج به همراه کاهش ابعاد

	Macro-F1	Micro-F1	adjusted RandIndex
Kmeans	0.36400034146900256	0.3543333333333333	0.5877100653105898
Bernouli MM	0.2013788603492342	0.2207935077000256	0.3258258987934195
Binomial MM			

### مقایسه

همانطور که می‌بینیم همچنان kmeans عملکرد مناسب‌تری دارد و البته با کاهش ابعاد نتایج بهتری در هر دو روش بدست آمده است.

### سوال دوم

معرفی ماتریس اسپارس:

ماتریسی است که در آن بیشتر داده‌ها صفر هستند. در مقابل آن اگر بیشتر داده‌های موجود غیر صفر باشند در این صورت این ماتریس dense است

برای خواندن داده‌ها در این حالت از کلاس csr\_matrix که برای داده‌های اسپارس می‌باشد استفاده می‌کنیم.

### کاهش بعد :

حال باید ابعاد را کاهش داده با استفاده از PCA و بعد با استفاده از svm آن را دسته‌بندی نماییم. به این منظور برای تشخیص تعداد ابعاد لازم و مورد استفاده با توجه به اینکه مسئله classification می‌باشد در PCA

از 3-fold cross validation استفاده می کنیم البته بهتر بود که تعداد fold ها را بیشتر می کردیم اما به دلیل اینکه زمان اجرا باز هم در این حالت بالا بود مجبور به کم کردن حالات محاسبه شدم : | .

کاهش ابعاد را به این صورت انجام می دهیم که بر روی داده های قسمت آموزش principal component component ها مشخص می کنیم (pc) ها با توجه به داده های آموزش بدست می آیند و در داده های آزمون همان ها استفاده می شوند) و svm را بر روی آن داده ها آموزش می دهیم. و بعد برای داده های بخش آزمون با همان pc های بدست آمده در قسمت آموزش ابعاد را کاهش می دهیم و با svm آموزش دیده برچسب کلاس ها را تخمین می نمیم.

برای اندازه گیری میزان خطا از تابع accuracy\_score موجود در پایتون استفاده می نماییم. و درصد درست دسته بندی ها را مشخص می نماید.

با توجه به اینکه باید تعداد مناسب ابعاد را هم پیدا کنیم دقت را به ازای تعداد ابعاد مختلف حساب کرده و در زیر آورده ایم. همانطور که مشخص است تعداد ابعاد ۱۰۰۰ مناسب به نظر می رسد برای این کار.

#### تعداد ابعاد جدید داده ها و میزان دقت دسته بند

تعداد ابعاد جدید	100	500	1000	2500	5000	10000
دقت دسته بند	0.793422404 9331964	0.829393627 954779	0.855772524 8372732	0.843782117 1634123	0.843782117 1634123	0.843782117 1634123

این جدول بر اساس روش اول نوشته شده است.

#### پیاده سازی روش دوم:

در این قسمت ابتدا داده های ۵ کلاس مختلف که دارای برچسب های ۰ تا ۴ هستند را از همیگر جدا کرده و به تعداد  $1000/5 = 200$  بردار برای تصویر کردن از pc ۲۰۰ اول هر یک از کلاس ها انتخاب می کنیم.

حال هر بار یکی از بردار های بدست آمده از هر کلاس را به ماتریس نهایی اضافه می کنیم و برای حل این مشکل که ممکن است به صورت خطی جدایی پذیر نباشد، هر بار بعد از اضافه کردن هر کدام از ستون ها columnrank ماتریس بدست آمده را حساب می کنیم اگر این مقدار کمتر از تعداد ستون ها شد بدین معناست که ستون آخر با بقیه مستقل خطی نبوده پس آن را حذف می کنیم. بعد از بدست آمدن ماتریس تبدیل جدید تبدیل آن به سراغ یادگیری svm و محاسبه خطا می رویم. باز مثل قبل 3 fold cross validation استفاده می نماییم.

### دقت دسته بند بعد از کاهش ابعاد به دو روش اول

	روش اول کاهش ابعاد	روش دوم کاهش ابعاد
دقت دسته بند	0.8557725248372732	0.8784409756782372

### اثبات قسمت سوم

$$\begin{aligned}
 & R_{ij} = \frac{\nabla^T c_i v}{\nabla^T c_j v} \quad \text{ابتدا سوم:} \\
 & \frac{\partial}{\partial v} \Rightarrow \frac{\nabla c_i v (\nabla^T c_j v) - \nabla c_j v (\nabla^T c_i v)}{(\nabla^T c_j v)^2} = 0 \\
 & \nabla c_i v - \nabla c_j v = 0 \Rightarrow c_i v = A c_j v \quad (A \in \mathbb{R}) \\
 & \nabla^T c_i v = (\nabla^T A c_j v) v = A (\nabla^T c_j v) v = A R_{ij} v
 \end{aligned}$$

### مقایسه سه روش:

به ترتیب روش های سوم، دوم و اول مناسب تر می باشند با توجه به اینکه نتایج را برای روش دو روش اول آورده ایم و این صحبت با نتایج بدست آمده کاملا هم خوانی دارد. دلیل خوبی مورد دوم نسبت به اول همانطور که در صورت

سوال نیز گفته شده است این است که پرچسب ها نیز در کاهش ابعاد نقش دارند و این نقش از طریق جدا سازی کلاس های مختلف صورت گرفته است.