دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

درس تحلیل کلان داده ها

استاد درس جناب آقای دکتر چهرقانی

(تمرین سوم)

محسن عبادپور | ۴۰۰۱۳۱۰۸۰ | m.ebadpour@aut.ac.ir

نیمسال دوم سال تحصیلی ۱۴۰۲-۱۴۰۱

فهرست پاسخ ها

فهرست مطالب

بخش اول: Data Stream

در این بخش پاسخ‌های مربوط به مبحث دسته‌بندی Data Stream آورده شده است که مشتمل بر پنج سوال می‌باشد.

سوال اول: عملکرد Adaptive Random Forest classifier (جنگل تصادفی تطبیقی)

دسته‌بند جنگل تصادفی یا Random Forest classifier یک دسته‌بند تجمعی(ensemble) بوده و با ترکیب چندین درخت تصمیم ضعیف (weak classifier) بصورت bagging یک دسته‌بند مطلوب را ایجاد می‌کند. bagging میتواند هم بصورت data bagging انجام پذیرد و هم بصورت feature bagging. بطور معمول درخت تصمیم ایجاد شده بصورت data bagging می باشد که هر درخت تصمیم صرفا یک زیرمجموعه ای از مجموعه داده آموزشی را دیده و یاد میگیرد و در نهایت برای پیش‌بینی نمونه به همه‌ی درخت های تصمیم داده شده و نتایج با هم ادغام می گردد(رای گیری).

جنگل تصادفی تطبیقی یا Adaptive Random Forest classifier یک نسخه بهبود یافته جنگل تصادفی برای مقابله با حجم انبوه داده‌ها بصورت جریان داده می‌باشد. این الگوریتم بدین صورت کار می‌کند که در ابتدای کار هیچ درخت تصمیمی وجود دارد و اصطلاحا جنگل خالی است. به محض ورود هر دسته داده(که میتواند یک داده نیز باشد)، یک درخت تصمیم جدید آموزش داده شده و به جنگل اضافه می‌شود. سپس تمام درخت های موجود با یک معیار ارزیابی نظیر Accuracy سنجیده می‌شوند؛ در صورتی که عملکرد هر درخت زیر حد آستانه تعیین شده باشد نشان میدهد که توزیع یاد گرفته‌ی آن outdate شده و عملکرد خوبی ندارد و پتانسیل آن را دارد که از جنگل حذف شود. البته در نسخه‌های مختلف این الگوریتم شروطی نیز برای تعیین تعداد درخت تصمیم موجود و جایگذینی درخت جدید(و نگهداری آن) صرفا با درخت منسوخ ارائه شده است که چندین رویکرد برای بهبود عملکرد این مرحله وجود دارد که اینجا اشاره نمی‌شود. حال عملکرد کل جنگل موجود(دقت و خطا) ارزیابی می‌شود. اگر عملکرد کل زیر حد آستانه مشخص شده باشد یعنی عملکرد مدل کلی کاملا نامطلوب بوده و میتواند حاکی از آن باشد که در توزیع جریان داده تغییرات اساسی به وجود

آمده است؛ در این حالت کل جنگل خالی شده و فرآیند آموزش درخت های کاملا جدید با داده‌های کلا جدید ورودی از جریان داده مجددا ادامه می‌یابد و با ورود دسته داده های جدید آموزش بصورت پیوسته ادامه یافته و این امکان به جنگل داده می‌شود که خود را با جریان داده و توزیع آن تطبیق دهد.

سوال دوم: بارگذاری مجموعه داده mnist

برای بارگذاری مجموعه داده mnist از پکیج keras و زیر مجموعه‌ی dataset استفاده شده است که بصورت پیشفرض ۶۰ هزار داده آموزشی وجود دارد که بصورت تصاویر ۲۸*۲۸ می‌باشد را در مقیاس ۰-۲۵۵ باز میگرداند. از جایی که scikit-multiflow بصورت بردار ویژگی پیاده سازی شده و امکان ورودی ماتریس وجود ندارد، من داده ها را بصورت بردارهای ۷۸۴ تایی تغییر شکل داده و نرمال نیز کرده ام. با توجه به همگرایی و همچنین محدودیت سخت افزاری بنده از ۲۰ هزار نمونه برای آموزش و حل این بخش از سوالات استفاده کرده ام.

سوال سوم: ایجاد جریان داده با استفاده از DataStream

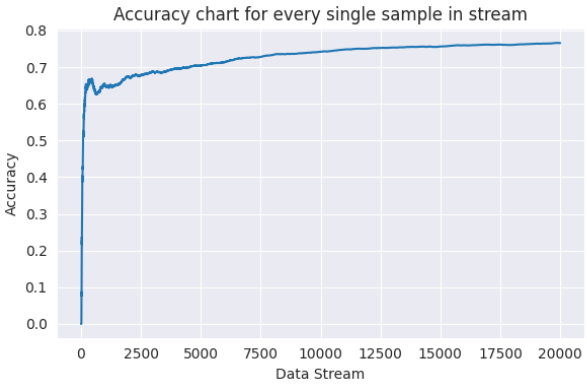
برای ایجاد جریان داده، مجموعه داده آموزشی مورد نظر را به همراه تعداد کلاس هدف که اینجا ۱۰ می باشد، به کلاس DataStream ورودی داده و شی مد نظر ساخته می‌شود جریان داده‌ای از مجموعه داده‌ی ما می باشد.

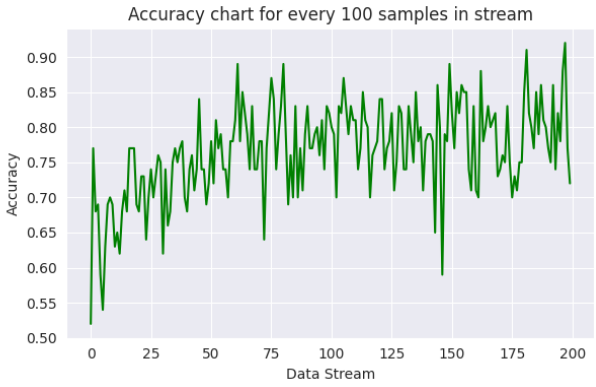
سوال چهارم: آموزش مدل Adaptive Random Forest classifier (جنگل تصادفی تطبیقی)

برای آموزش مدل، از کلاس AdaptiveRandomForestClassifier از کتابخانه scikit-multiflow استفاده شده است. طبق سعی و خطا تعداد بهینه درخت‌های تصمیم موجود در جنگل تصادفی ۱۵ بدست آمده است؛ البته با توجه به ماهیت مجموعه داده که تصویری بوده و دسته بندی آن بصورت بردار ویژگی flat شده می باشد، تغییر پارامتر های مختلف در ایجاد جنگل تصادفی تطبیقی تاثیر چندانی در بهبود عملکرد مدل نداشت. آموزش این مدل بصورت increamental بوده و در هر مرحله یک نمونه وارد ورودی در نظر گرفته شده است. در پیاده سازی از متد های has_more_samles و next_sample که مربوط به کلاس DataStream میباشد استفاده شده است.

سوال پنجم: نمودار خطا در حین آموزش

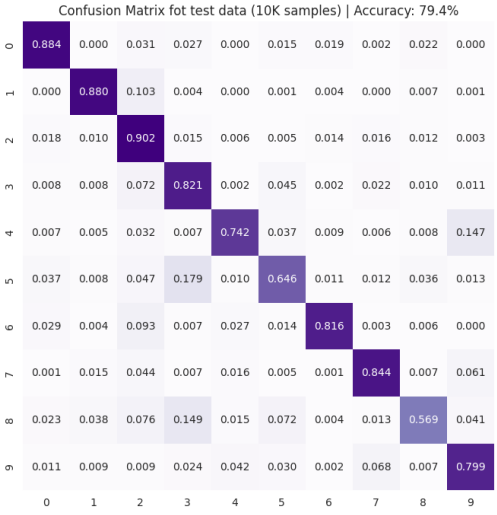
دو نمودار خطای مدنظر تولید شده و بصورت زیر قابل مشاهده است. همانطور که دیده می‌شود پس از مشاهده تقریبا ۱۰ هزار نمونه آموزشی مدل به همگرایی نسبی رسیده و عملکرد آن در ادامه افزایش محسوس نداشته است. در خصوص عملکرد ۱۰۰ تایی نیز قابل نتیجه ای است که میانگین عملکرد موجود برابر با همان میانگین اصلی و به ازای نمونه های آموزشی تک و در حد ۸۰–۷۸٪ می باشد.





فعالیت بیشتر: تحلیل عملکرد مدل بر اساس ماتریس در همریختگی

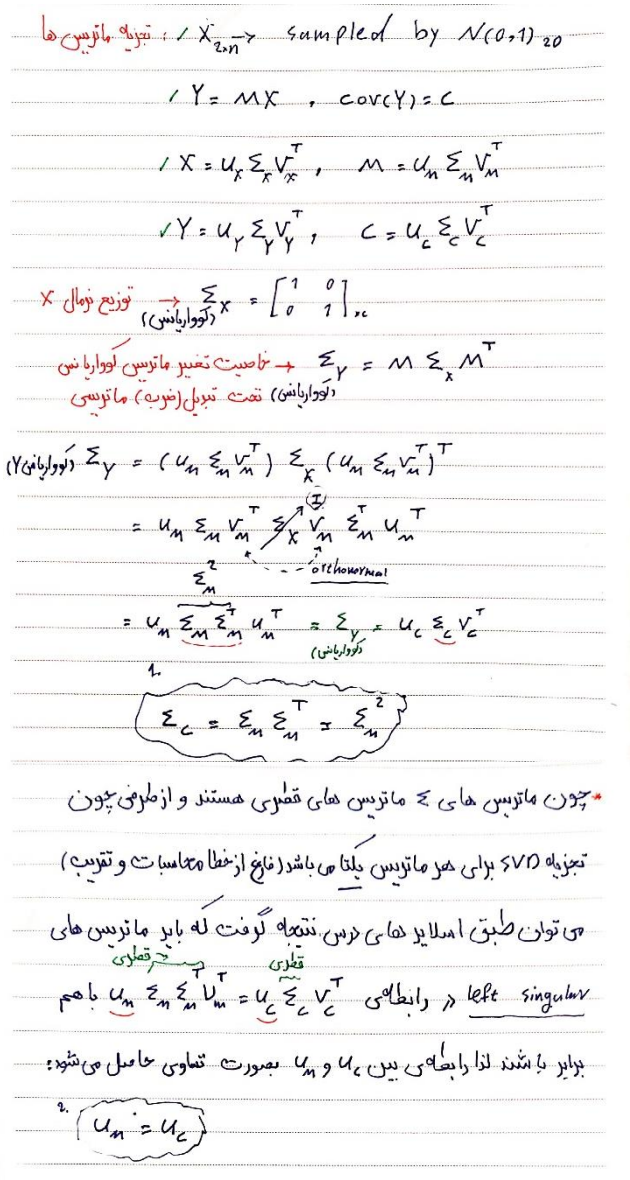
در این قسمت و به عنوان فعالیت بیشتر و اضافی ماتریس درهمریختگی مربوط به مدل محاسبه شده و به صورت زیر می باشد؛ همانطور که دیده می‌شود عملکرد مدل بیشتر در مقابله با تصاویر اعداد ۳ و ۵ و ۸ با چالش مواجه بوده و خطا دارد. دلیل این امر آن است که ما تصاویر دو بعدی را در قالب بردار های ویژگی یک بعدی به مدل داده ایم(ماهیت درخت های تصمیم و جریان داده در کلاس گفته شده یک بعدی است) و از جایی که تصاویر این سه رقم در بردار های ویژگی و پیکسل های هم پوشان شان بیشتر است، درخت های موجود هر یک نظر و رای مختلف داشته و در نتیجه عملکرد مدل افزایش نمی یابد.



بخش دوم: Singular Value Decomposition

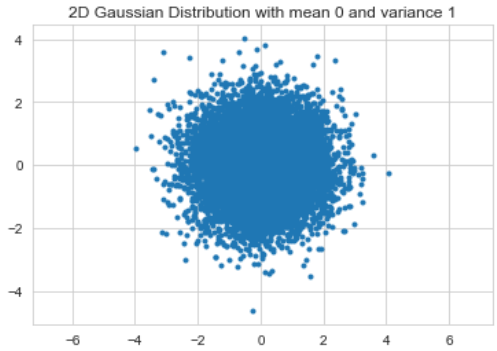
در این بخش پاسخ‌های مربوط به مبحث دسته‌بندی Singular Value Decomposition آورده شده است که مشتمل بر چهار سوال و پرسش می‌باشد.

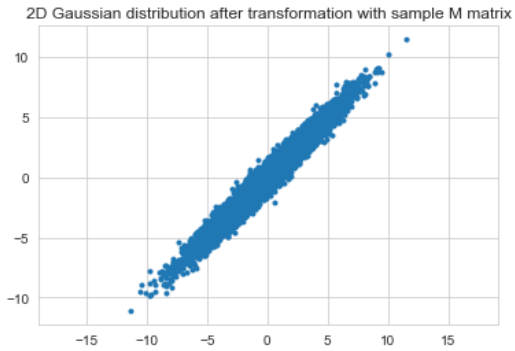
سوال اول: ارتباط ماتریس‌های تجزیه M و $covariance(Y)$



علاوه بر اثبات فوق و نشان دادن نحوه رسیدن به ارتباط فوق، بنده با شبیه سازی نیز صحت چنین رابطه را ارزیابی کردم که حاکی از درستی آن بود و نتیجه بصورت زیر است (یک ماتریس دلخواه M تعریف شده و بروی نمونه های گرفته شده از توزیع نرمال اعمال شد):

$$Sample\ M = \begin{bmatrix} 2.82 & -0.28 \\ 2.82 & 0.28 \end{bmatrix}$$





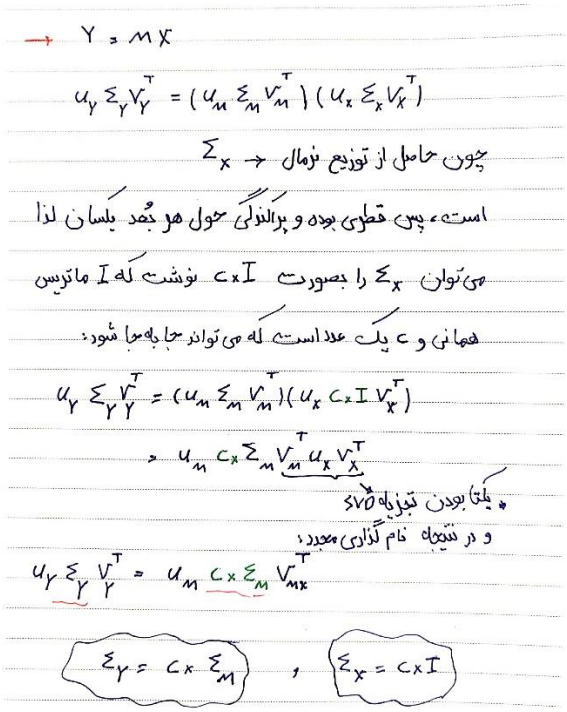

```
np.diag(S_C),np.diag(S_M**2)
```
✓ 0.0s

```
(array([[16.16690675,  0.        ],
        [ 0.        ,  0.16081937]]),
 array([[16.  ,  0.  ],
        [ 0.  ,  0.16]]))
```

```
U_C,U_M
```
✓ 0.0s

```
(array([[-0.7071487 , -0.70706486],
        [-0.70706486,  0.7071487 ]]),
 array([[-0.70710678, -0.70710678],
        [-0.70710678,  0.70710678]]))
```

سوال دوم: ارتباط ماتریس‌های تجزیه M و Y



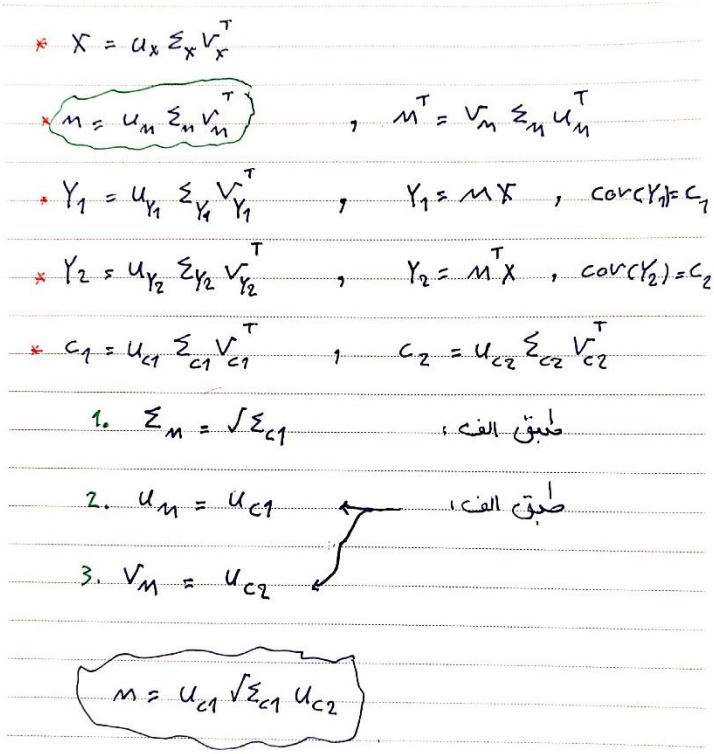
همانند قسمت قبل، برای این بخش نیز صحت رابطه‌ی بدست آمده را با ماتریس M دلخواه نشان می‌دهیم:

```
S_Y,S_M * np.mean(S_X)
✓ 0.0s
(array([402.06107626,  40.10540379]), array([401.5575755 ,  40.15575755]))
```

سوال سوم: یافتن ماتریس M

طبق محاسبات و روابط حاصل از دو قسمت قبلی و همچنین نتیجه‌گیری های صفحه ماتریس M بصورت زیر بدست می‌آید:

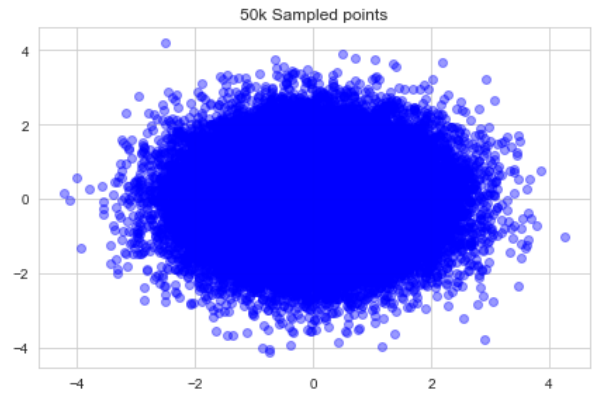
$$M = \begin{bmatrix} 6.0186 & 1.003 \\ 6.0210 & 1.998 \end{bmatrix}$$

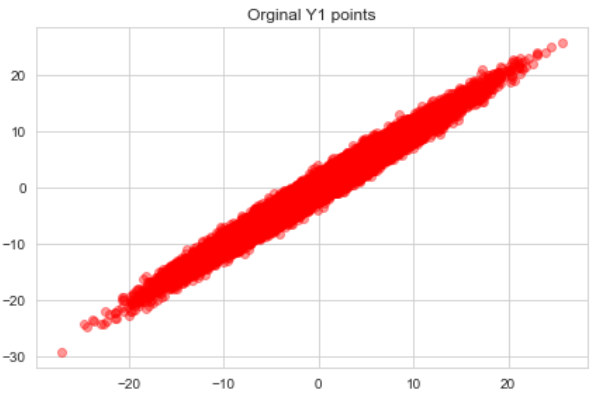


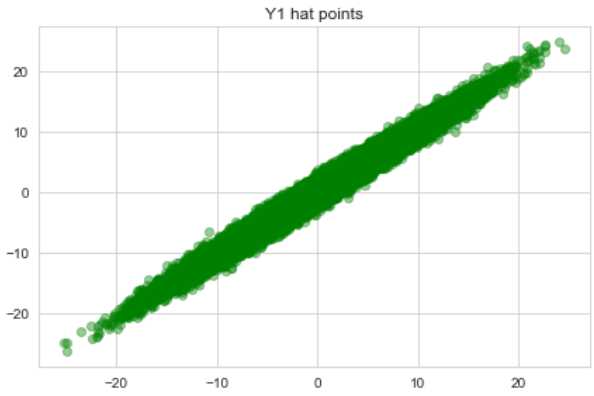
سوال چهارم: تولید نمونه و انتقال تحت ماتریس محاسبه شده

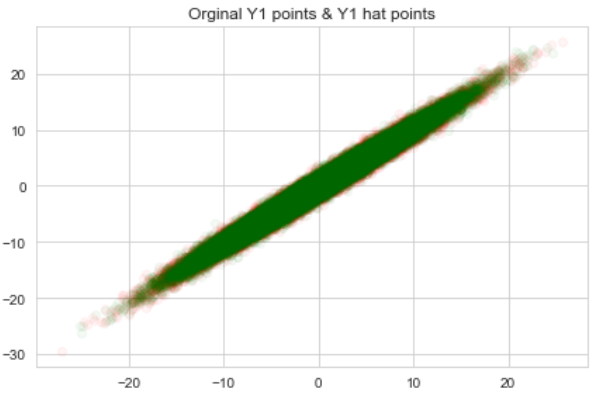
طبق صورت سوال، در این بخش ۵۰ هزار نمونه از توزیع گوسی با میانگین صفر و ماتریس کواریانس I تولید شده که در نمودار ها قابل مشاهده است. برای هر کدام از Y1 و Y2 چهار نمودار رسم شده است که اولی نمونه های تولید شده از توزیع گوسی(X)، دومی نمونه‌های اورجینال داده شده در پوشه P2(Y1 یا Y2)، سومی نمونه‌های انتقال یافته از توزیع گوسی تحت ماتریس M محاسبه شده($\hat{Y}1$ یا $\hat{Y}2$) و چهارمی نیز رسم توامان هر دو می‌باشد.

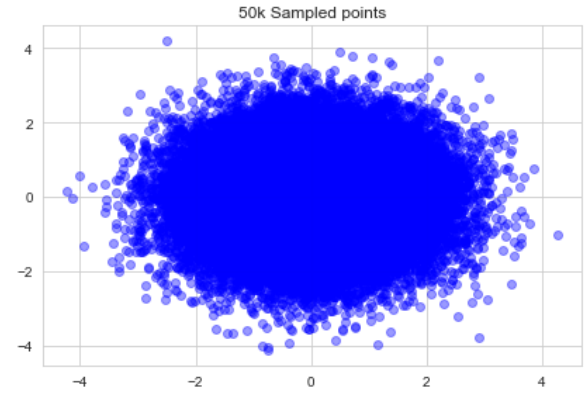
حال اگر بخواهیم Y ها را با $\hat{Y}$ ها مقایسه کنیم، ملاحظه میکنیم که شکل‌ها روی هم افتاده و بر هم منطبق شده اند که این دو نکته را نشان میدهد؛ اولین نکته اینکه ماتریس M بدست آورده شده صحیح می باشد و دومین نکته این است که یک ماتریس تبدیلی نظیر M تاثیر برابر و مشابهی بروی نمونه‌های حاصل از یک توزیع گوسی داشته است و روند نمونه‌گیری حاصل از توزیع گوسی تاثیری در رفتار ماتریس تبدیل M ندارد.

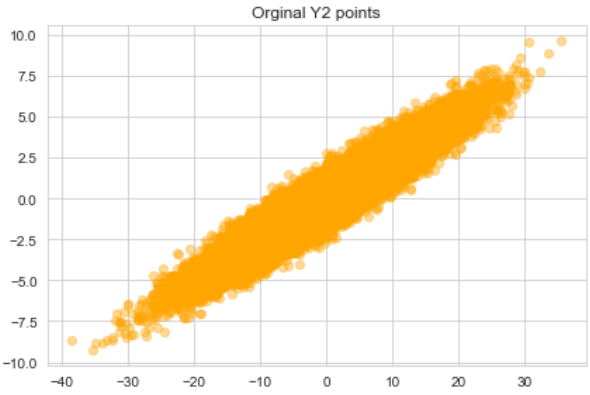

50k Sampled points

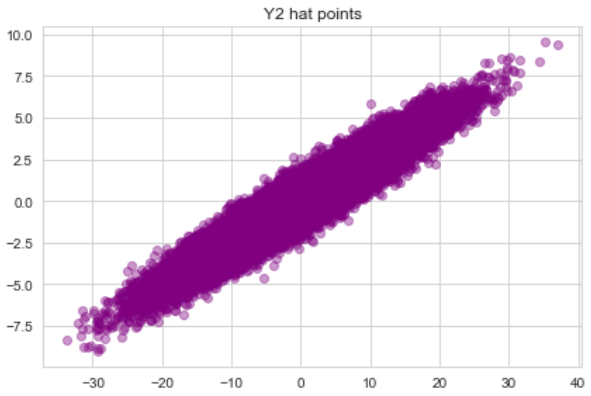

Orginal Y1 points

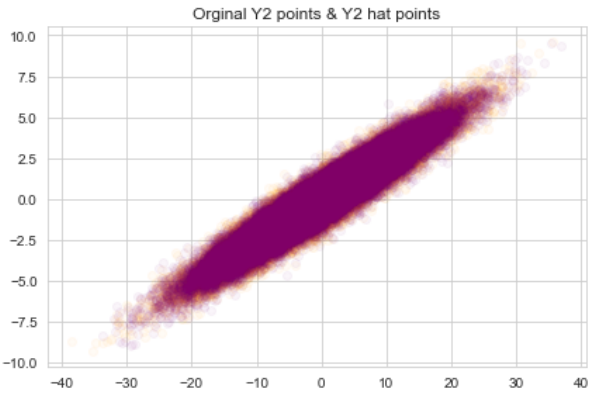

Y1 hat points


Orginal Y1 points & Y1 hat points


50k Sampled points


Orginal Y2 points

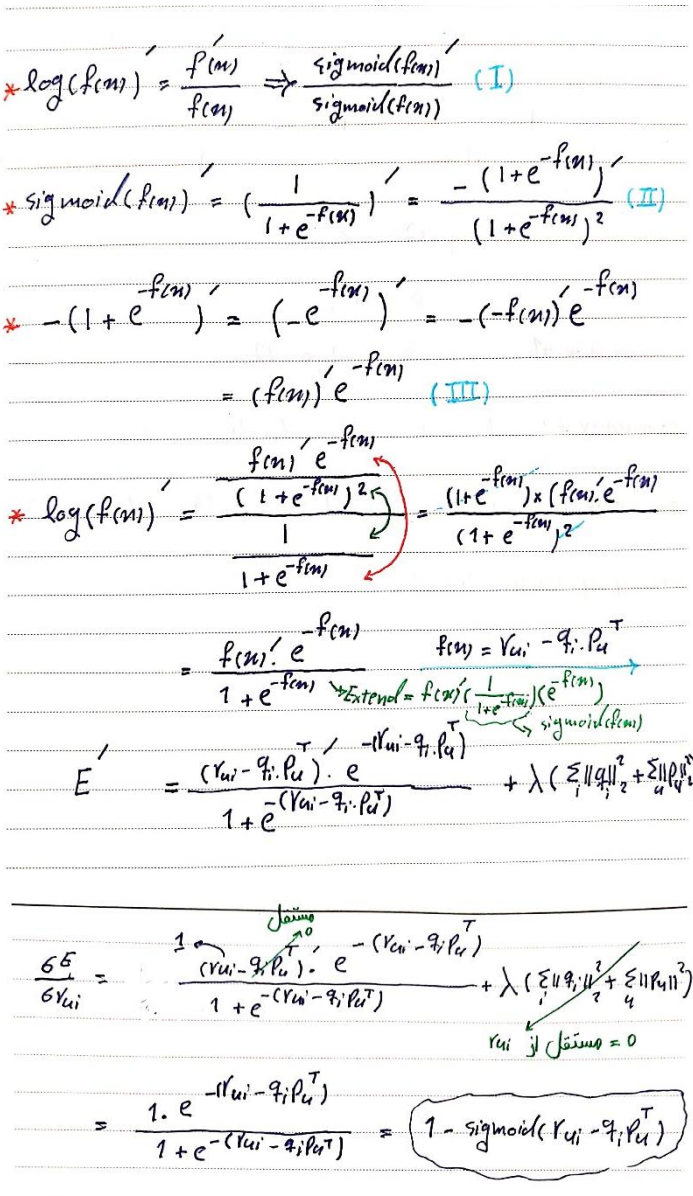

Y2 hat points


Orginal Y2 points & Y2 hat points

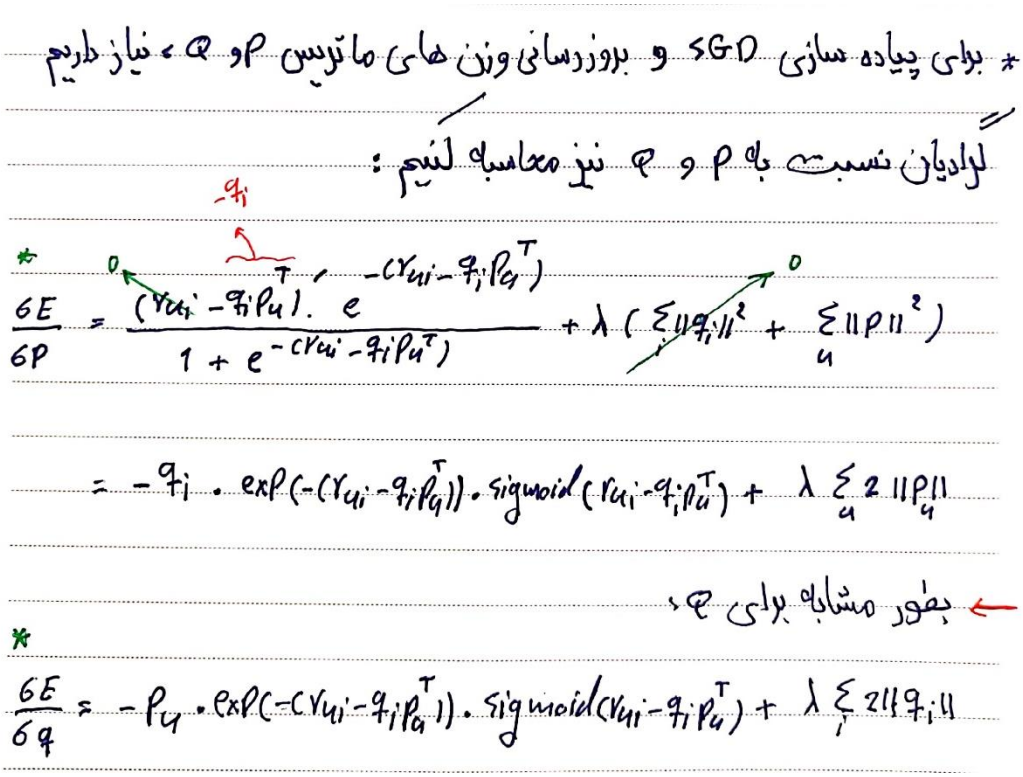
بخش سوم: Recommender Systems

در این بخش پاسخ‌های مربوط به مبحث دسته‌بندی Recommender Systems آورده شده است که مشتمل بر دو سوال می‌باشد.

سوال اول: محاسبه مشتق نسبت به r_{ui}



سوال دوم: پیاده سازی SGD



برای پیاده‌سازی این قسمت از فرمت و قالب داده شده استفاده گردیده و کل قسمت های مشخص شده پر شده است. برای اهداف رسم نمودار های خطا یا پیش‌بینی و ارزیابی چند فیلد به کلاس اضافه شده است. همچنین برای راحتی و انسجام بیشتر چند تابع کمکی نیز تعریف و از آن استفاده شده است که عبارت اند از

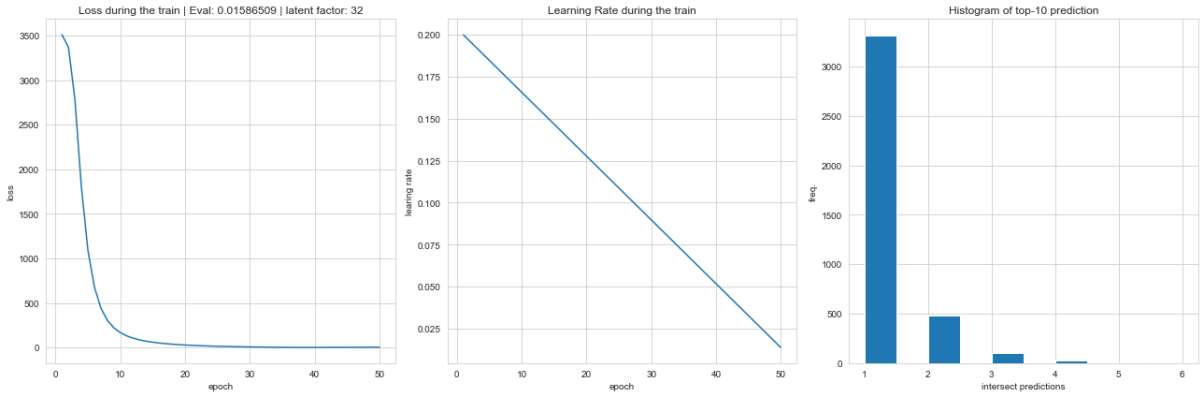
۱. read_data برای خواندن دیتاست و ایجاد ماتریس R
۲. train_test_split برای تفکیک و استخراج عناصر آموزش و آزمون از ماتریس R (۲۰ درصد از آیتم های هر کاربر برای آزمون در نظر گرفته شده است)
۳. get_batches برای ایجاد batch های تصادفی برای SGD به ازای کاربران
۴. learning_rate_scheduler برای تغییر نرخ آموزش با پیشروی تکرار های آموزشی
۵. sum_squared_norms برای محاسبه نرم ماتریس های Q و P برای بدست آوردن مقدار خطا در هر تکرار آموزشی

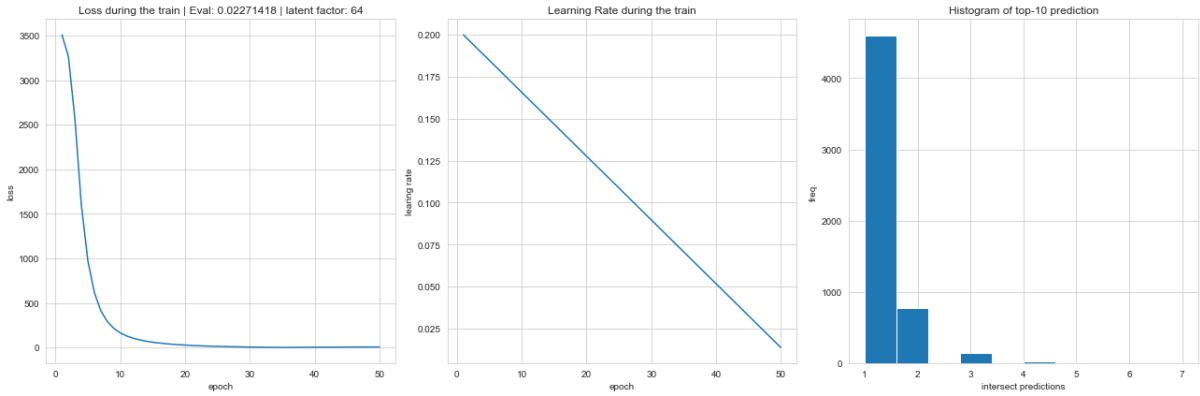
از مفاهیم یادگیری و نزول در راستای گرادیان می دانیم که نرخ یادگیری و گام حرکتی ثابت به هیچ عنوان تصمیم مطلوب و بهینه‌ای نمی باشد و طی اثبات های بهینه سازی میتوان نشان داد که نزول در راستای گرادیان و برای یک نقطه شروع تصادفی بهتر است با نرخ های بزرگ یادگیری و با سرعت بالا آغاز شده و به مرور گام های آموزشی نیز نرخ یادگیری کاهش یابد؛ با توجه به این نکته، بنده از نرخ یادگیری ۰.۲ شروع کرده و طی ۵۰ تکرار آموزشی آن را تقریبا به ۰.۰۱ بصورت خطی کاهش داده ام.

همچنین برای نرخ regularization نیز در زمان پیاده سازی و با سعی خطا به این نتیجه رسیدم که اگر برابر با ۰.۰۰۱ تنظیم شود همگرایی، دقت نهایی و روند آموزشی بهتر و مطلوب تری حاصل می شود و میتوان از فضای نهفته با ابعاد بالاتر نیز استفاده کرد و به همین جهت از این نرخ به جای ۰.۱ استفاده کرده ام. (همچنین در latent factor های بالا نرخ ۰.۱ باعث overflow شدن عناصر ماتریس P و Q در زمان بروزرسانی وزنها می شد)

در یادگیری های انجام شده از batch size برابر با ۵۱۲ استفاده شده و در نتیجه هر تکرار آموزشی مشتمل بر ۵۹ بار بروزرسانی وزن شده است.(۲۹۸۵۸/۵۱۲) بنده در هر گام آموزشی طبق گرادیان محاسبه شده وزن ماتریس‌ها را بروز کرده و همچنین مقدار خطای حاصل را طبق تابع هزینه داده محاسبه کرده و در نهایت رسم کرده ام که در ادامه قابل مشاهده است.(میانگین خطای batch های هر تکرار)

بنده آموزش را به ازای ۳۲ و ۶۴ بُعد برای factor انجام داده و برای هر کدام نمودار خطای آموزشی، معیار ۱۰ آیتم برتر (در عنوان نمودار درج شده است) و نمودار فراوانی تعداد آیتم های مشترک در مجموعه داده آزمون به همراه تغییرات نرخ یادگیری رسم کرده ام. به ازای تعداد فاکتور های ۳۲ معیار ۱۰ آیتم برتر برابر با ۰.۰۱۵۸ و به ازای تعداد فاکتور های ۶۴ معیار مذکور برابر با ۰.۰۲۲۷ حاصل شده است.


Loss during the train | Eval: 0.01586509 | latent factor: 32
loss
epoch
Learning Rate during the train
learning rate
epoch
Histogram of top-10 prediction
freq.
intersect predictions


Loss during the train | Eval: 0.02271418 | latent factor: 64
loss
epoch
Learning Rate during the train
learning rate
epoch
Histogram of top-10 prediction
freq.
intersect predictions