گزارش فاز اول پروژه ی یادگیری آماری

استاد درس : دکتر هدی محمد زاده

دانشجو:

سعید منصورلکورج – 99102304

فاز یک پروژه

KNN:

Knn_impute_by_user:

Validation Accuracy with k = 1 is 0.624471

Validation Accuracy with k = 6 is 0.678098

Validation Accuracy with k = 11 is 0.689529

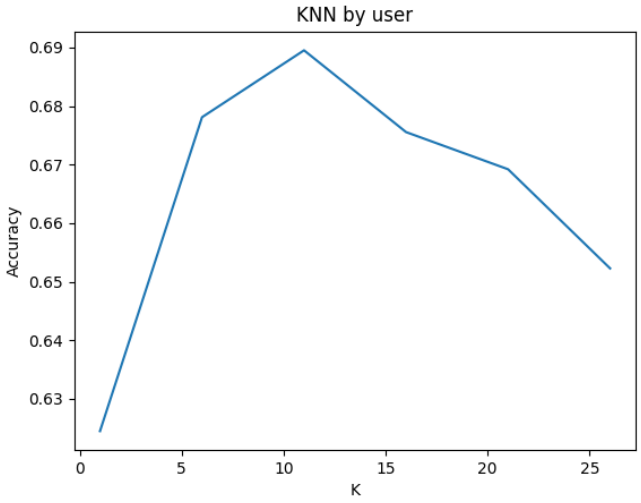
Validation Accuracy with k = 16 is 0.675557

Validation Accuracy with k = 21 is 0.669207

Validation Accuracy with k = 26 is 0.652272

Best K in KNN imputed by user is : 11.000000 and its accuracy is : 0.689529

نمودار کشیده شده:



Knn_impute_by_question:

Validation Accuracy with k = 1 is 0.607113

Validation Accuracy with k = 6 is 0.654248

Validation Accuracy with k = 11 is 0.682614

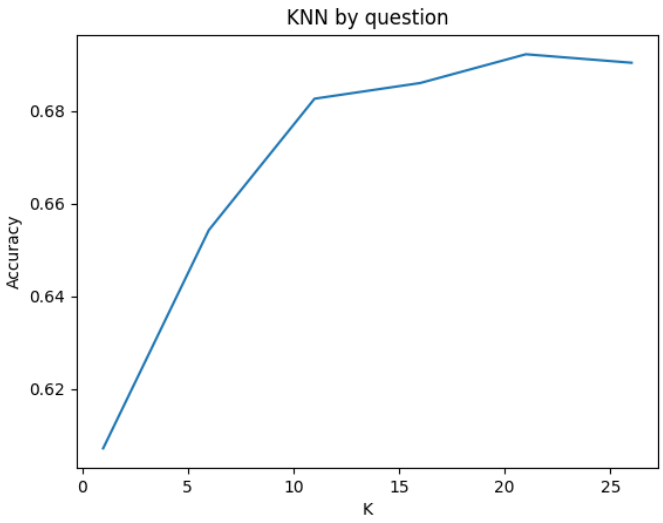
Validation Accuracy with k = 16 is 0.686001

Validation Accuracy with k = 21 is 0.692210

Validation Accuracy with k = 26 is 0.690375

Best K in KNN imputed by user is : 21.000000 and its accuracy is : 0.692210

نمودار خواسته شده برای این بخش:



مقایسه :

دو مورد عملکرد مشابهی داشتند تنها تفاوت آنها را میتوان مربوط به مقدار ماکسیمم دقت و مکان رخداد آنها محدود کرد اما به صورت کلی با توجه به مقادیر می توان گفت انتخاب مدل دوم منطقی تر به نظر می آید.

محدودیت های این روش:

به صورت کلی روش KNN دو تا عیب بزرگ دارد:

- این روش classification برای داده هایی با تعداد کم نتیجه ی خوبی نمی دهد و تعداد داده های آموزش باید بزرگ باشد.
- همچنین روش KNN محاسبات سنگین و طولانی دارد مخصوصا اگر داده ها و تعداد همسایه ها را زیاد در نظر بگیریم

IRT :

روابط تئوری مربوطه:

برای حل این سوال باید log likelihood را max کنیم مانند مسائلی که در Logistic regression داشتیم:

$$L(\theta,\beta) = \sum p_{ij}^{c_{ij}} + \left(1 - p_{ij}\right)^{\left(1-c_{ij}\right)}$$

$$l(\theta,\beta) = \sum c_{ij} \log\, p_{ij} + \left(1 - c_{ij}\right) \log\left(1 - p_{ij}\right)$$

$$p\left(c_{ij} = 1 \mid \theta_i, \beta_j\right) = \frac{\exp\left(\theta_i - \beta_j\right)}{1 + \exp\left(\theta_i - \beta_j\right)}$$

مشتق نسبت به θ_i:

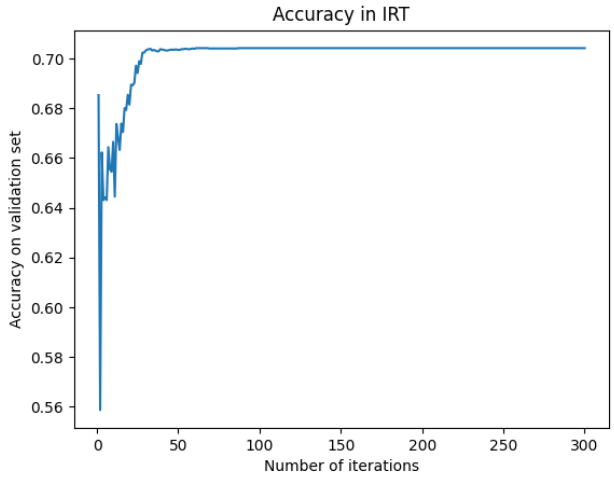
$$\frac{\partial l}{\partial \theta_i} = \sum c_{ij} - p_{ij}, itr\ on\ j$$

مشتق نسبت به β_i:

$$\frac{\partial l}{\partial \beta_j} = -\sum c_{ij} - p_{ij}, itr\ on\ i$$

حالا طبق تعریف داده شده کد این سوال را مینویسم:

در این بخش ما iteration را از یک تا 300 در نظر گرفته و نمودار دقت را روی نمونه های validation را رسم می کنیم:



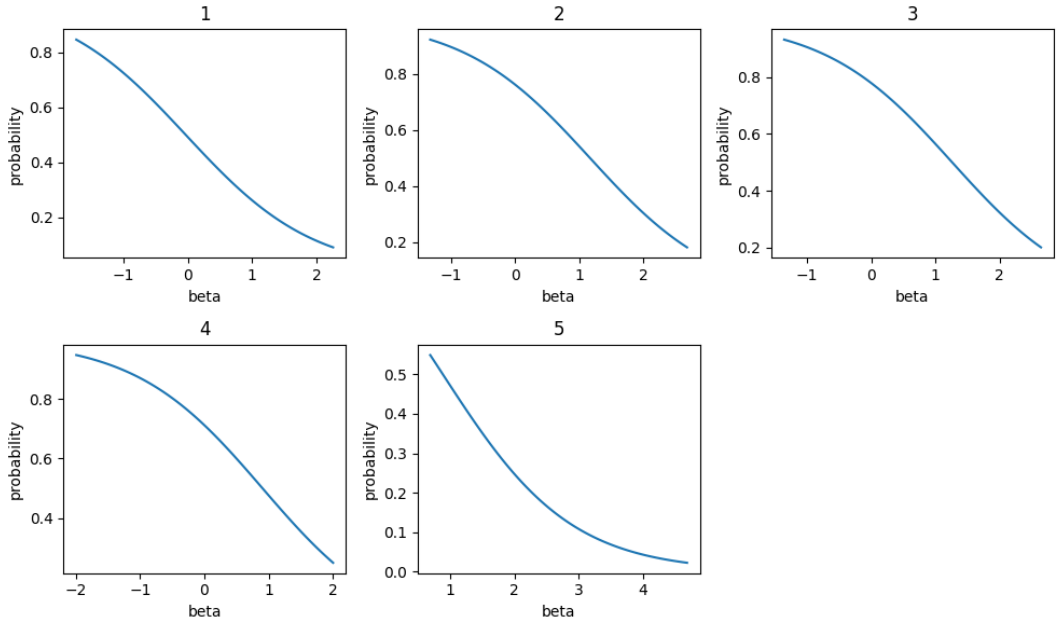
بهترین مقدار دقت نیز برای lr=0.4 به صورت زیر گزارش شد: (در هر مرحله هم در 0.9 ضرب شده است)

Best accuracy for itr = 61 is 0.704205

log likelihood is 31619.713075

چون که $\log(0) = -\infty$ می باشد من شرط گذاشتم که آنرا مورد برسی قرار ندهند.

درباره ی بخش d من منظور سوال را به صورت دقیق نفهمیدم چون β چیزی نیست که من تعیین کنم و توسط داده های سوال باید تعیین شود و همچنین اگر 5 سوال انتخاب کنم احتمال پاسخ دادن به آن به دانش آموزی که آنرا پاسخ می دهد بستگی دارد و به صورت کلی نمی توان گفت که احتمال چقدر است به همین دلیل برای این بخش من 5 سوال اول مربوط به 5 دانش آموز اول را در نظر گرفته و با توجه به مقدار β و θ و برای بازه ی $[\theta_i - 2, \theta_i + 2]$ نمودار های خواسته شده را رسم می کنم.



مشاهده می شود که با افزایش θ احتمال کاهش پیدا کرده است که اتفاق منطقی است زیرا با افزایش آن در واقع درجه ی سختی سوال را تغییر داده ایم و بازه ی کلی احتمالی آن هم مربوط به مقدار اصلی θ و همچنین توانایی دانش آموز یعنی β می باشد.

Matrix Factorization:

Svd:

در بخش اول همانطور که خواسته شده بود از svd استفاده کردم که نتیجه ی چندان خوبی نداشت، یکی از محدودیت های آن نیز مربوط به این می شد که با مقادیر خالی توی ماتریس تنک نمی شد که svd را انجام داد برای همین من مقادیر اولیه ی آنرا 0 قرار دادم و سپس svd را اعمال کردم (ایده ی اولیم این بود که به جای 0 با توجه به بقیه ی داده های موجود احتمال اولیه قرار بدم اما چون حدس زدم ممکن است ایده ی اصلی سوال را زیر سوال ببرد از اینکار منصرف شدم)

The accuracy in SVD for k=5 is 0.438188

The accuracy in SVD for k=10 is 0.453288

The accuracy in SVD for k=20 is 0.456816

The accuracy in SVD for k=50 is 0.432684

The accuracy in SVD for k=100 is 0.413209

مشاهده می شود که بیشترین دقت نیز مربوط به K = 20 بوده است.

ALS:

توضیحات الگوریتم:

ALS matrix factorization الگوریتمی است که هدف آن تبدیل ماتریس اسپارس به دو ماتریس با رنک کمتر می باشد، الگوریتم به صورت مرحله ای می باشد(iterative) که هدف آن مینیمم کردن تابع انرژی داده شده در صورت سوال می باشد (در هر مرحله با توجه به خطای موجود در آن مرحله U و Z را آپدیت میکند تا به خطای کمتری برسیم، نمونه ی مشابه آنرا در اسلاید های درس نیز دیده بودیم) و برای اینکار روش gradient descent را انتخاب کردم و هایپرپارامتر را هم بر اساس دقت روی validation set انتخاب کردم و بعد هم تاثیر مقدار epoch در دقت نهایی را نشان دادم.

دقت ها به ازای مقادیر مختلف k به صورت زیر می باشند:

The accuracy in ALS for k=5 is 0.690517

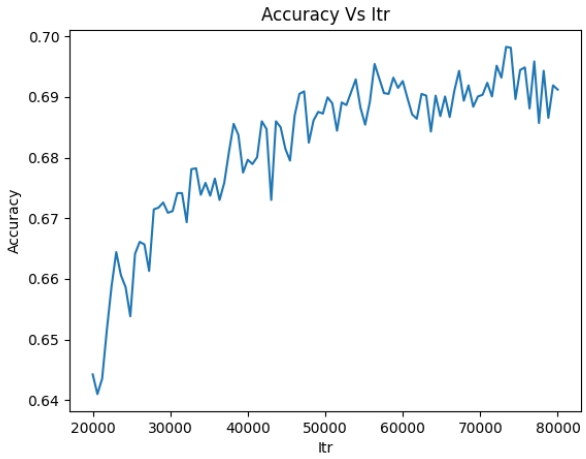
The accuracy in ALS for k=10 is 0.687694

The accuracy in ALS for k=20 is 0.694750

The accuracy in ALS for k=50 is 0.683884

The accuracy in ALS for k=100 is 0.690799

مشاهده می شود که نسبت به روش SVD پیشرفت زیادی داشتیم و بهترین مقدار k هم طبق نتایج k = 20 می باشد و حالا بر طبق این k نمودار دقت بر اساس تعداد iteration ها را می کشم. (توجه شود که این بخش از کد کمی طولانی می باشد چون در رنج 20000 تا 80000 ما حدود 100 تا مقدار را محاسبه می کنیم)



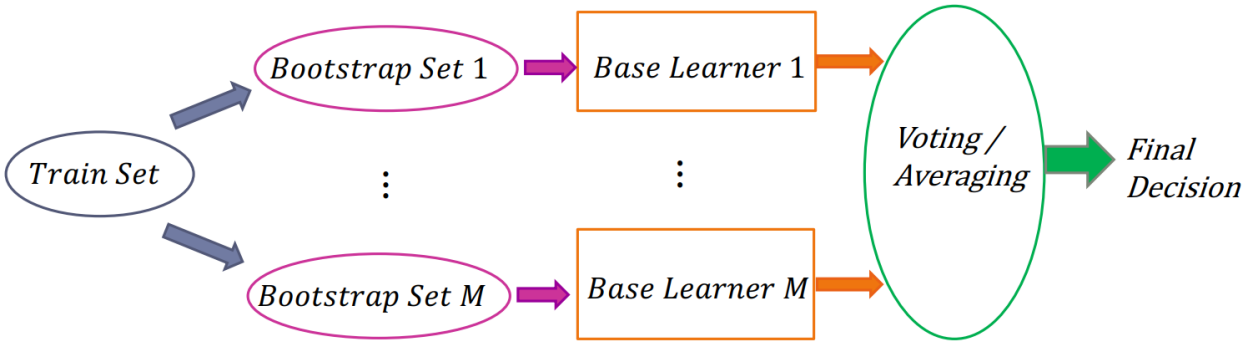
حالا در بخش آخر فرمول داده شده در بخش اول را مطابق بقیه ی فرمول های موجود به فضای classification می بریم:

برای تبدیل آن به *binary classification* یکی از راه های رایج استفاده از *map* کردن به *sigmoid* و استفاده از توابع احتمالی می باشد (*most likelihood*):

$$\min_{\mathbf{U},\mathbf{Z}} \sum_{(n,m)\in O} \left[C_{nm} \log\left(\sigma(\mathbf{u}_n^\top \mathbf{z}_m)\right) + (1 - C_{nm}) \log\left(1 - \sigma(\mathbf{u}_n^\top \mathbf{z}_m)\right) \right]$$

:Ensemble

برای این بخش دقیقا از الگوریتم موجود در جزوه ی درس برای پیاده سازی استفاده کردم:



در ابتدا سه تا درخت به عنوان base-learner ایجاد کردم و سپس bootstrap را با یک حلقه ی for و تابع random ایجاد کرده و سپس با استفاده از سه تا مجموعه ی train ایجاد شده base-leaner ها را آموزش داده و validation-set داده شده را پیش بینی کرده و در نهایت فرآیند voting را انجام دادم.

فرآیند voting:

یک تابع مجزا برای این بخش ساختم تا که برای هر نمونه ی validation ببیند بیشترین مقدار بین سه تا base-learner چیست و آنرا نتیجه قرار می دهد.(مثلا دو تا گفتن 0 میشه و یکی گفت 1 می شود من مقدار نهایی را 0 قرار می دهم)

نتایج به صورت زیر حاصل شد :

در ابتدا دقت شود مدل هر درخت استفاده شده به صورت زیر می باشد:

```
model1 = DecisionTreeClassifier(criterion='gini',
max_depth=5,
min_samples_split=10,
min_samples_leaf=5,
max_features=None,
random_state=42)
model2 = DecisionTreeClassifier(criterion='gini',
max_depth=5,
min_samples_split=10,
min_samples_leaf=5,
max_features=None,
random_state=42)
model3 = DecisionTreeClassifier(criterion='gini',
max_depth=5,
min_samples_split=10,
min_samples_leaf=5,
max_features=None,
random_state=42)
```

نتیجه ی نهایی:

The accuracy of first model is 0.450607

The accuracy of second model is 0.591589

The accuracy of third model is 0.594129

The accuracy of voted is 0.591871

البته در اینجا بیشترین درصد مربوط به voted نشد که میتوان اینطور تحلیل کرد در برخی نمونه ها مانند همان مثال خودمان وقتی دو تا 0 و یک 1 داریم جواب درست همان 1 می باشد و نه 0.

پس لزوما ما به نتیجه ی بهتری نمیرسیم اما این دلیلی نمی شود که این روش، روش خوبی نمی باشد زیرا در بسیاری از موارد این گونه نیست که بستگی به موارد زیادی دارد، مثلا حتی در همین مورد توجه شود که نتیجه ی نهایی حدود 15 درصد از مدل اول بهتر است و تفاوت نتیجه ی ان با بهترین مدل هم حدود 0.1 درصد است که می توان آنرا نادیده گرفت پس به صورت کلی نتیجه می شود که این مورد نتیجه ی قابل قبولی می باشد.