عنوان : گزارش تمرین سوم یادگیری ماشین (RBF)

نگارنده : سحر داستانی اوغانی

شماره دانشجویی : ۹۹۱۱۲۱۰۸

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

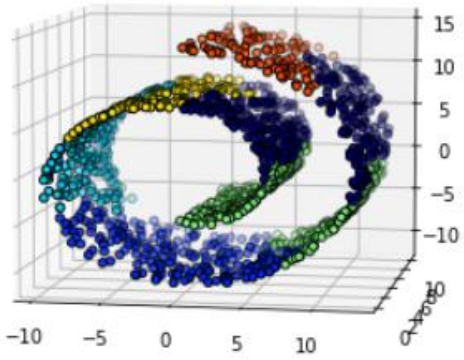
دسترسی به دادگان

در این سوال می‌توان به دو طریق به دادگان دسترسی پیدا کرد:

- داده‌های کتابخانه swissroll در سایت Swiss Roll Dataset
- داده‌هایی که به همراه فایل تمرین آمده بودند.

در پیاده‌سازی، ابتدا با استفاده از داده‌های کتابخانه‌ی swissroll، شکل ظاهری data-set نشان داده شده است و سپس بر روی داده‌های موجود در فایل تمرین، مراحل خواسته شده در صورت سوال پیاده شده است.

کتابخانه Swissroll



این داده متشکل از دو دیتاست می‌باشد. یک ماتریس ۳ × ۱۶۰۰ برای داده‌های سه بعدی و دیگری ماتریس ۱ × ۱۶۰۰ برای برچسب گذاری داده‌ها.

شکل دیتاست به شیوه‌ی مقابل است:

داده‌ی data.mat

ابتدا دیتاست فراخوانی شده و سپس ۱۰۰ ردیف آن به صورت رندوم انتخاب شده است.

برای انجام عملیات خواسته شده در سوال، توابع وکلاس‌هایی را باید تعریف کنیم. در زیر به شرح کوتاهی از عملکرد هر کدام از آن‌ها می‌پردازیم.

توابع معرفی شده

get_distance

این تابع فاصله‌ی اقلیدسی میان هر دو نقطه را محاسبه می‌کند. از آن بیشتر برای محاسبه فاصله‌ی اقلیدسی میان هر x و مرکز دسته‌ای که x به آن متعلق است استفاده کردیم.

RBF

این کلاس بخش عمده‌ی عملیات را بر عهده می‌گیرد.

ابتدا مقادیری را برای ورودی دریافت می‌کند.
۳ تابع با توجه به صورت سوال که گفته شده است در هر مرحله تاثیر حداقل سه تابع را بررسی کنید، تعریف شده است:
Gaussian, Inverse-multiquadratic- multiquadratic
در تابع rbf از ۳ تابع تعریف شده در بالا استفاده شده است، و لیستی با نام RBF_list به ازای هر تابع برگردانده می‌شود. این لیست حاوی مقدار xهای موجود در تابع، مرکز و انحراف از معیار دسته و شاخه‌ی مربوط به آن xها می‌باشد.

در تابع دیگری به نام fit مقادیر w و xهای مربوط به rbf بدست آورده شده است.
با فیت کردن تابع بالا بر روی داده‌ها می‌توان متوجه شد که حاصل پیش‌بینی شده برای مقادیر داده‌شده به چه صورت است، این حاصل با استفاده از تابع pred بدست می‌آید.

در نهایت نیز میزان دقت الگوریتم با استفاده از تابع result نشان‌داده می‌شود.

Gaussian

در این قسمت پیاده سازی الگوریتم برای تابع gaussian را توضیح می‌دهیم و در ادامه، تمامی این روش‌ها را برای توابع دیگر نیز به کار می‌بریم.

ابتدا برای این که بتوانیم، تاثیر سایز دیتا را بر روی الگوریتم خود پیاده کنیم، باید الگوریتم را برای درصدهای مختلف داده‌های test و train بیازماییم. برای این هدف، یک لیست ۱۰ تایی از درصدهای رندوم برای دیتاست test تعریف شده است. این لیست حاوی مقادیر غیر تکراری است و ۱۰ عدد رندوم را در بازه‌ی ۱ تا ۴۶ بدست می‌آورد. دلیل انتخاب این دو عدد نیز در زیر آمده است:

به دلیل این که درصد داده‌های تست و داده‌های ولیدیشن را یکی گرفتیم، درصد داده‌های train از تفاضل مجموع درصد داده‌های تست و ولیدیشن از ۱۰۰ حاصل می‌شود.
از طرفی به دلیل روبرو، تعداد کلاس‌های موجود در y_train، ۸ تاست.

```
# number of classes are 8 because:
# X_train.shape = (50, 3)
# y_train.shape = (50,)
# np.unique(y_train) = array([0, 1, 2, 3, 4, 5, 6, 7]) --> Up to 8 numbers
```

پس درصد داده‌های train باید بزرگتر مساوی تعداد کلاس‌های موجود در دیتاست باشد. (در این‌جا 8 =< test_percentage)
به همین دلیل مجموع درصد داده‌های تست و ولیدیشن نباید بزرگتر از ۹۲٪ باشد. (یعنی ۴۶+۴۶)

از طرفی داده‌های train هم نمی‌توانند وجو نداشته باشند، یعنی درصد آن‌ها نمی‌تواند ۰ باشد. پس مجموع درصد داده‌های تست و ولیدیشن نباید کمتر از ۲ باشد.(یعنی ۱+۱).

حال که درصدها را مشخص کردیم، کافی است ادامه‌ی الگوریتم را اجرا کنیم:

داده‌ها را به ۳ مجموعه‌ی train, validation, test تقسیم کردیم، سپس با استفاده از کتابخانه‌ی Kmeans در scikitlearn داده‌ها را برچسب گذاری کردیم. در ادامه مقادیر y برای داده‌های تست و ولیدیشن را با استفاده از ماژول predict، پیش‌بینی کردیم و مقادیر مرکز هر خوشه در خوشه بندی را با استفاده از ماژول cluster_centers_ بدست آوردیم.

حال در ادامه مشخص کردیم که هر x به کدام y در کدام خوشه متعلق است و انحراف معیار هر خوشه را از این طریق بدست آوردیم.

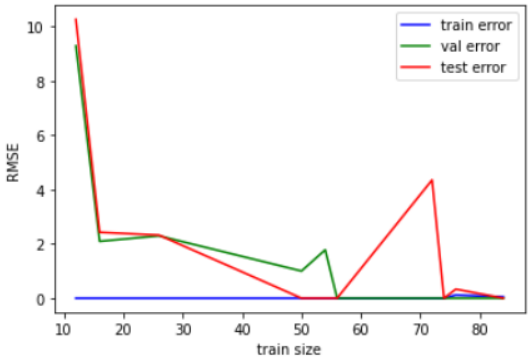
با فیت کردن تابع fit بروی داده‌های train، مقادیر wرا بدست آورده و مقدار خطای RMSE برای هر مجموعه داده بدست آورده شده است.

```
test_percentage: 44
train sets accuracy: 1.0
validation sets accuracy: 0.20454545454545456
test sets accuracy: 0.22727272727272727
------------------------------------------------
test_percentage: 42
train sets accuracy: 1.0
validation sets accuracy: 0.47619047619047616
test sets accuracy: 0.42857142857142855
------------------------------------------------
test_percentage: 37
train sets accuracy: 1.0
validation sets accuracy: 0.7027027027027027
test sets accuracy: 0.6216216216216216
------------------------------------------------
test_percentage: 25
train sets accuracy: 1.0
validation sets accuracy: 0.96
test sets accuracy: 1.0
------------------------------------------------
test_percentage: 23
train sets accuracy: 1.0
validation sets accuracy: 0.9130434782608695
test sets accuracy: 1.0
```

```
test_percentage: 22
train sets accuracy: 1.0
validation sets accuracy: 1.0
test sets accuracy: 1.0
------------------------------------------------
test_percentage: 14
train sets accuracy: 1.0
validation sets accuracy: 1.0
test sets accuracy: 0.8571428571428571
------------------------------------------------
test_percentage: 13
train sets accuracy: 1.0
validation sets accuracy: 1.0
test sets accuracy: 1.0
------------------------------------------------
test_percentage: 12
train sets accuracy: 0.9868421052631579
validation sets accuracy: 1.0
test sets accuracy: 0.9166666666666666
------------------------------------------------
test_percentage: 8
train sets accuracy: 0.9880952380952381
validation sets accuracy: 1.0
test sets accuracy: 1.0
```

همانطور که مشخص است الگوریتم برای زمانی که درصد داده‌های تست و ولیدیشن بزرگتر از ۲۵ می‌شود، بهتر عمل می‌کند و دارای دقت بالاتری است. یعنی درصد داده‌های train بزرگتر از ۵۰٪ باشد.

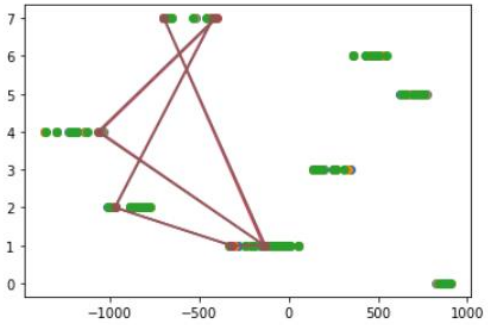
```
train_percentages are:  [12 16 26 50 54 56 72 74 76 84]
val_percentages are:    [44 42 37 25 23 22 14 13 12  8]
test_percentages are:   [44 42 37 25 23 22 14 13 12  8]
```



نمودار RMSE نیز به ازای درصدهای مختلف مجموعه‌ی train به شرح زیر است:

این نشان می‌دهد که با افزایش مجموعه داده‌گانی که برای آموزش الگوریتم از آن‌ها استفاده می‌شود، خطای داده‌های تست و ولیدیشن کاهش یافته ولی با کاهش آن، این خطا افزایش می‌یابد.

در ادامه نیز نموداری را مشاهده می‌کنید که پراکندگی داده‌های train و test را با رنگ سبز و نارنجی نمایش می‌دهد، که ارتباط میان داده‌های تست با خطی آجری نمایش داده شده است:



Inverse_Multiquadratic

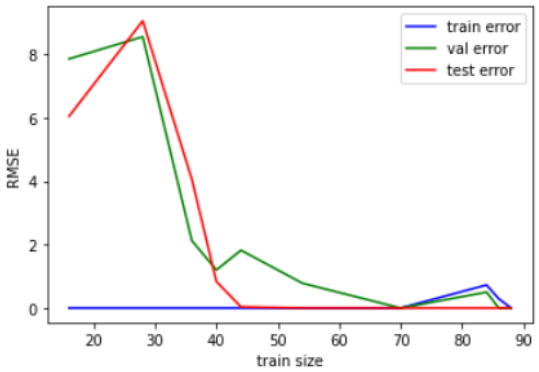
عملیات ذکر شده برای این مرحله نیز صادق است، با یکدیگر نتایج را بررسی می‌کنیم:

```
test_percentage: 42
train sets accuracy: 1.0
validation sets accuracy: 0.5
test sets accuracy: 0.5238095238095238
-----------------------------------------------
test_percentage: 36
train sets accuracy: 1.0
validation sets accuracy: 0.6666666666666666
test sets accuracy: 0.6666666666666666
-----------------------------------------------
test_percentage: 32
train sets accuracy: 1.0
validation sets accuracy: 0.90625
test sets accuracy: 0.78125
-----------------------------------------------
test_percentage: 30
train sets accuracy: 1.0
validation sets accuracy: 0.9666666666666667
test sets accuracy: 0.9333333333333333
-----------------------------------------------
test_percentage: 28
train sets accuracy: 1.0
validation sets accuracy: 0.8928571428571429
test sets accuracy: 0.9642857142857143
```

```
test_percentage: 23
train sets accuracy: 1.0
validation sets accuracy: 0.9130434782608695
test sets accuracy: 1.0
-----------------------------------------------
test_percentage: 15
train sets accuracy: 1.0
validation sets accuracy: 1.0
test sets accuracy: 1.0
-----------------------------------------------
test_percentage: 8
train sets accuracy: 0.9761904761904762
validation sets accuracy: 0.875
test sets accuracy: 1.0
-----------------------------------------------
test_percentage: 7
train sets accuracy: 0.9883720930232558
validation sets accuracy: 1.0
test sets accuracy: 1.0
-----------------------------------------------
test_percentage: 6
train sets accuracy: 1.0
validation sets accuracy: 1.0
test sets accuracy: 1.0
```

در این تابع، دقت الگوریتم با رسیدن درصد داده‌های تست به ۲۳، افزایش می‌یابد و به ۱ می‌رسد. یعنی زمانی که درصد داده‌های train ۵۴٪ باشد.

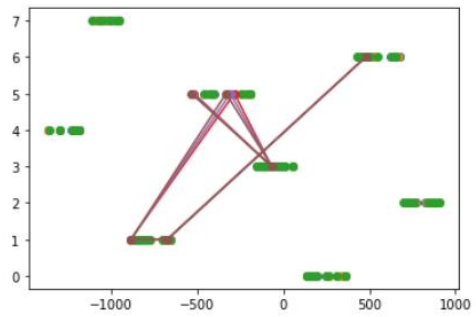
```
train_percentages are:  [16 28 36 40 44 54 70 84 86 88]
val_percentages are:    [42 36 32 30 28 23 15  8  7  6]
test_percentages are:   [42 36 32 30 28 23 15  8  7  6]
```



نمودار RMSE نیز به ازای درصدهای مختلف مجموعه‌ی train به شرح زیر است:

همانند قبل با افزایش درصد داده‌های train، خطای مجموعه کاهش می‌یابد، با این تفاوت که در بررسی این تابع می‌توانیم این نتیجه را بگیریم که در بازه‌ای (۲۰ تا ۳۰٪) مقادیر خطا افزایش ناگهانی دارند. این می‌تواند به دلیل وجود نویز یا ویژگی تابع باشد.

در ادامه نیز نموداری را مشاهده می‌کنید که پراکندگی داده‌های train و test را با رنگ سبز و نارنجی نمایش می‌دهد، که ارتباط میان داده‌های تست با خطی آجری نمایش داده شده است:



Multiquadratic

عملیات ذکر شده برای این مرحله نیز صادق است، با یکدیگر نتایج را بررسی می‌کنیم:

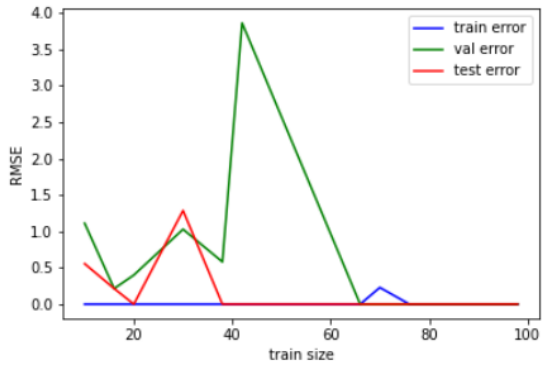
```
test_percentage: 45
train sets accuracy: 1.0
validation sets accuracy: 0.9555555555555556
test sets accuracy: 0.9777777777777777
-----------------------------------------------
test_percentage: 42
train sets accuracy: 1.0
validation sets accuracy: 0.9761904761904762
test sets accuracy: 0.9761904761904762
-----------------------------------------------
test_percentage: 40
train sets accuracy: 1.0
validation sets accuracy: 0.975
test sets accuracy: 1.0
-----------------------------------------------
test_percentage: 35
train sets accuracy: 1.0
validation sets accuracy: 0.9714285714285714
test sets accuracy: 0.9428571428571428
-----------------------------------------------
test_percentage: 31
train sets accuracy: 1.0
validation sets accuracy: 0.9354838709677419
test sets accuracy: 1.0
```

```
test_percentage: 29
train sets accuracy: 1.0
validation sets accuracy: 0.8620689655172413
test sets accuracy: 1.0
-----------------------------------------------
test_percentage: 17
train sets accuracy: 1.0
validation sets accuracy: 1.0
test sets accuracy: 1.0
-----------------------------------------------
test_percentage: 15
train sets accuracy: 0.9857142857142858
validation sets accuracy: 1.0
test sets accuracy: 1.0
-----------------------------------------------
test_percentage: 12
train sets accuracy: 1.0
validation sets accuracy: 1.0
test sets accuracy: 1.0
-----------------------------------------------
test_percentage: 1
train sets accuracy: 1.0
validation sets accuracy: 1.0
test sets accuracy: 1.0
```

در اینجا، دقت الگوریتم زمانی به ۱ می‌رسد که درصد داده‌های تست ۴۰ درصد باشد یعنی داده‌های train ۲۰ درصد از دیتاست ما را تشکیل داده باشد.

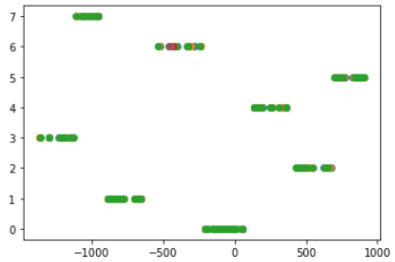
```
train_percentages are:  [10 16 20 30 38 42 66 70 76 98]
val_percentages are:    [45 42 40 35 31 29 17 15 12  1]
test_percentages are:   [45 42 40 35 31 29 17 15 12  1]
```

نمودار RMSE نیز به ازای درصدهای مختلف مجموعه‌ی train به شرح زیر است:



همانند قبل با افزایش درصد داده‌های train، خطای مجموعه کاهش می‌یابد، با این تفاوت که در بررسی این تابع می‌توانیم این نتیجه را بگیریم که در بازه‌ای (۴۰ تا ۵۰٪) مقدار خطای validation افزایش ناگهانی داشته و در ۵۰ تا ۶۰٪ کاهش ناگهانی. این می‌تواند به دلیل وجود نویز یا ویژگی تابع باشد.

در ادامه نیز نموداری را مشاهده می‌کنید که پراکندگی داده‌های train و test را با رنگ سبز و نارنجی نمایش می‌دهد، که ارتباط میان داده‌های تست با خطی آجری نمایش داده شده است:



منابع

https://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/modules/generated/sklearn.datasets.make_swiss_roll.html

http://www.rueckstiess.net/research/snippets/show/72d2363e

https://www.openwith.org/file-extensions/dat/991

https://www.openwith.org/file-extensions/dat/991

https://www.kaggle.com/questions-and-answers/27699

https://www.reddit.com/r/learnpython/comments/epwvp6/reading_dat_file_in_python/

https://note.nkmk.me/en/python-pandas-t-transpose/

https://stackoverflow.com/questions/16932825/why-cant-non-default-arguments-follow-default-arguments

https://www.google.com/search?q=non-default+argument+follows+default+argument&oq=&aqs=chrome.4.35i39i362l8...8.73714196j0j7&sourceid=chrome&ie=UTF-8

https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.RBF.html

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html?highlight=kmeans#sklearn.cluster.KMeans

https://realpython.com/python-keyerror/

https://github.com/scikit-learn/scikit-learn/issues/9021

https://scikit-learn.org/stable/auto_examples/cluster/plot_ward_structured_vs_unstructured.html#sphx-glr-auto-examples-cluster-plot-ward-structured-vs-unstructured-py

کمک‌های دیگر: کمک از یکی از دوستان در مقطع ارشد، برای قسمتی که داده‌ها را به هر یک از خوشه‌ها نسبت می‌داد(خط ۶۱ تا ۶۸ در حلقه‌ی for)