Machine Learning Course

Assignment #4:

Kernel KNN and KSPCA

آرمین خیاطی
9931153

پویان انصاری راد
9960565

Spring 2021

فهرست

مقدمه

در این تمرین دو متد زیر پیاده سازی شده اند:

- Kernel KNN
- KSPCA

ابتدا داده ها به نسبت ۷۰٪ داده آموزشی و ۳۰٪ داده تست برای هر دیتاست جدا میشوند و سپس فراخور هر قسمت موارد خواسته شده انجام میگردد.

روش های مبتنی بر کرنل، دسته ای از الگوریتم های آنالیز یا تشخیص پترن هستند که شناخته شده ترین آن ها SVM است. این روش ها، داده ها را به امید اینکه جدا پذیر تر شوند و بتوان آن ها را از هم تفکیک کرد، به ابعاد بالاتری در فضا نگاشت میکنند. هیچ محدودیتی در افزایش ابعاد وجود ندارد و تا بینهایت نیز میتوان ابعاد را بیشتر کرد.

Kernel Trick

حقه کرنل، یک ابزار قوی برای این مقصود میباشد زیرا پلی میان خطی بودن و غیر خطی بودن برای هر الگوریتمی که بتوان آن را بر اساس ضرب داخلی بیان کرد می باشند. اگر ما در ابتدا داده ها را به فضای بالاتر نگاشت کنیم این کار هزینه زیادی برای ما دارد، اما در حقه کرنل، نیازی به انجام این کار نیست و این نگاشت هیچوقت محاسبه نمیشود. اگر الگوریتم را بتوان بر اساس ضرب داخلی بین دو وکتور بیان کرد، تمام چیزی که نیاز داریم، جایگزین کردن این ضرب داخلی با ضرب داخلی در فضای مورد نظر می باشد. پس هر جا که ضرب داخلی استفاده میشود، با کرنل جایگزین میگردد. تابع کرنل، ضرب داخلی را در فضای ویژگی ها مطرح میکند.

$$K(x,y) = <\phi(x),\phi(y)>$$

شایان ذکر است دیتاست های استفاده شده در این پیاده سازی عبارتند از:

BreastTissue, Diabetes, Glass, Ionosphere, Sonar, Wine

بخش اول

با استفاده از تابع کرنل میتوان الگوریتم در فضای بالاتر اجرا کرد، بدون اینکه داده ها بصورت explicit به این فضای بالاتر نگاشت کنیم حتی اگر این فضای بالاتر، فضای بی نهایت باشد.

در ادامه بعد از پیاده سازی ، به تشریح نتایج روش KNN با کرنل های linear، RBF، کرنل درجه ۱، درجه دو و درجه ۳ می پردازیم. واریانسی که در کرنل RBF برای هر دیتاست استفاده میشود نیز، با انجام Cross Validation روی مقادیر ۰,۱ تا ۱ با قدم های ۰,۰۵ محاسبه شده است.

نتایج :

Dataest	Accuracy of Algorithms					
	۱NN	1NN+Linear kernel	۱NN+RBF kernel	۱NN+Polynomial kernel (d = 1)	۱NN+Polynomial kernel (d = 2)	۱NN+Polynomial kernel (d = 3)
BreastTissue	0.55	0.60	0.206	0.53	0.54	0.53
Diabetes	0.70	0.72	0.563	0.72	0.70	0.72
Glass	0.73	0.743	0.71	0.75	0.73	0.746
Ionosphere	0.86	0.90	0.87	0.86	0.87	0.856
Sonar	0.81	0.793	0.80	0.823	0.83	0.846
Wine	0.72	0.79	0.493	0.753	0.756	0.766

Dataest	Running Time of Algorithms (Seconds)					
	۱NN	1NN+Linear kernel	۱NN+RBF kernel	۱NN+Polynomial kernel (d = 1)	۱NN+Polynomial kernel (d = 2)	۱NN+Polynomial kernel (d = 3)
BreastTissue	0.033	0.048	0.065	0.039	0.043	0.045
Diabetes	0.265	0.491	0.83	0.23	0.254	0.215
Glass	0.047	0.072	0.085	0.072	0.055	0.049
Ionosphere	0.145	0.26	0.339	0.168	0.225	0.150
Sonar	0.153	0.18	0.272	0.182	0.225	0.182
Wine	0.042	0.093	0.122	0.051	0.091	0.046

بخش دوم :

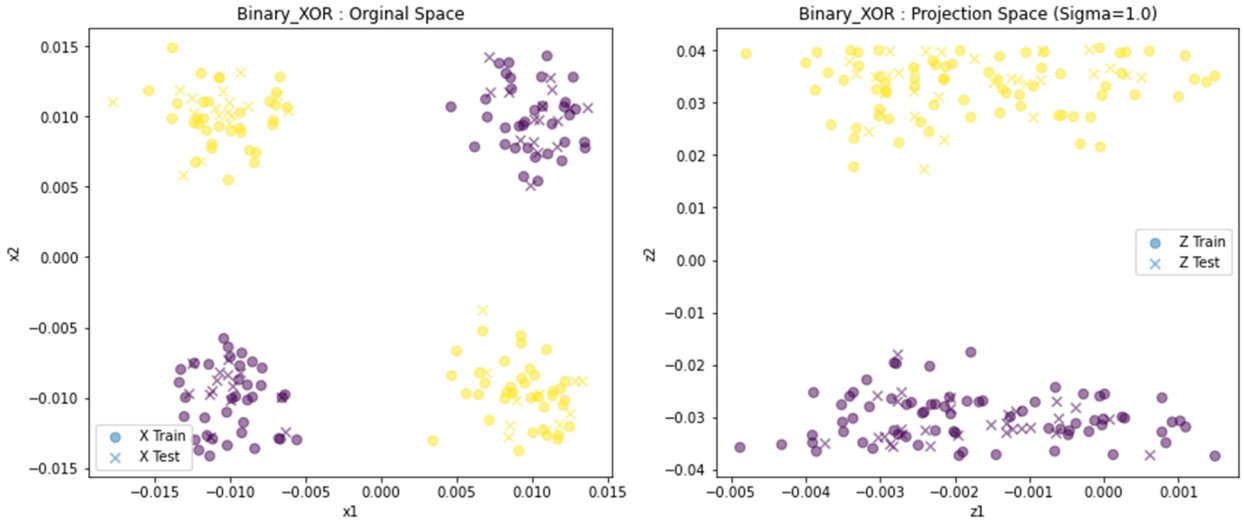
در این بخش با استفاده از روش Kernel Supervised Principal Component Analysis داده ها به صورت ضمنی با کرنل های RBF , Delta به ابعاد بالا برده میشوند و سپس تا ۲ بعد کاهش بعد داده میشوند به روش PCA . برای سیگماهای از بازه ۰,۱ تا ۱,۰ برای هر دیتاست اجرا گرفته شد که بهترین سیگما ها از نظر تفکیک شوندگی کلاس های مختلف و افزایش واریانس درون کلاسی از قرار زیر شد است :

دیتاست های تست گرفته شد عبارتند از :

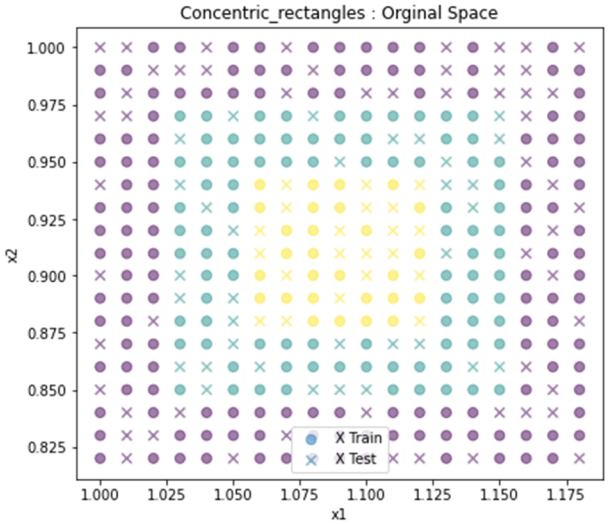
datasets = [‘Binary_XOR’ , ‘Concentric_rectangles’ , ‘Concentric_rings’ , ‘Twomoons’]

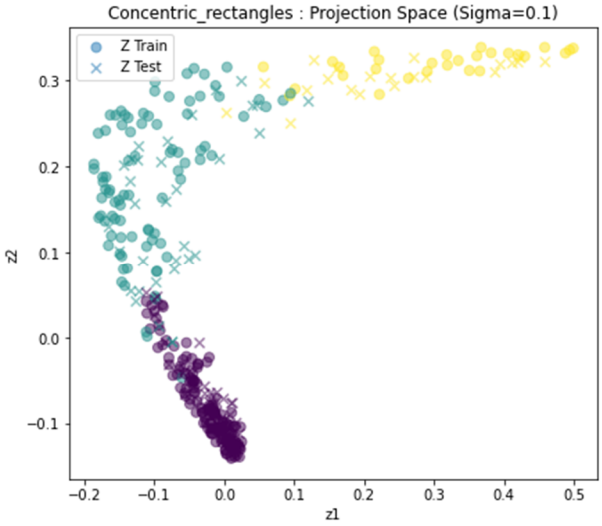
نتایج :

```
--------------------------DataSet : Binary_XOR--------------------------
x_train.shape = (140, 2)
y_train.shape = (140,)
m = 140
n = 2
x_test.shape = (60, 2)
y_test.shape = (60,)
m_test = 60
c = 2
y_unique = [1 2]
```

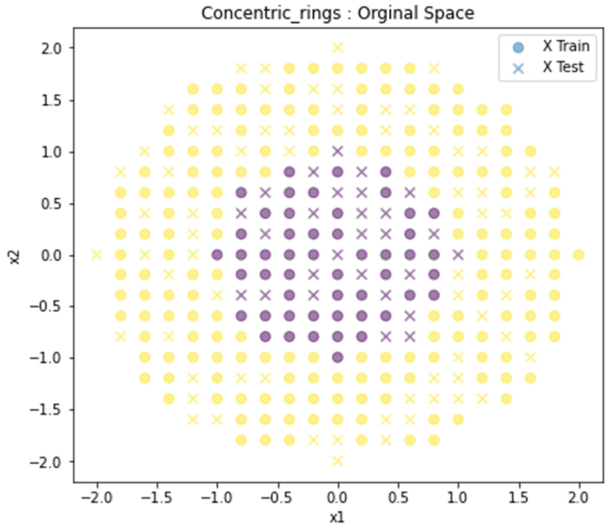


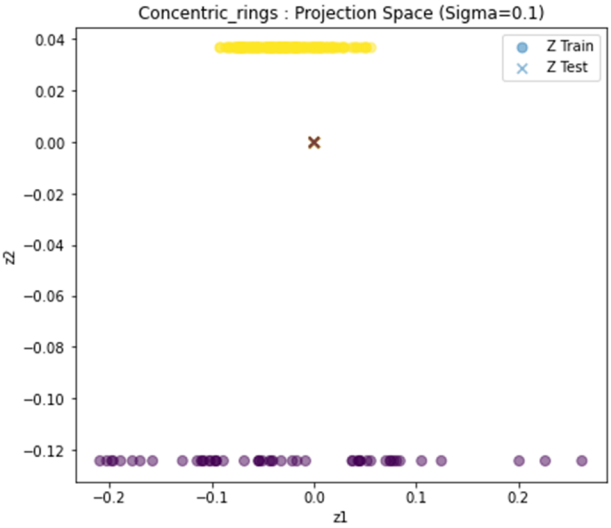
```
--------------------------DataSet : Concentric_rectangles-------------------
x_train.shape = (252, 2)
y_train.shape = (252,)
m = 252
n = 2
x_test.shape = (109, 2)
y_test.shape = (109,)
m_test = 109
c = 3
y_unique = [1 2 3]
```



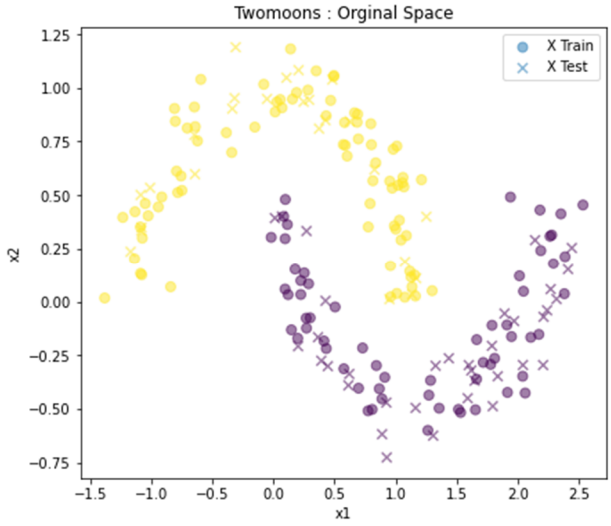


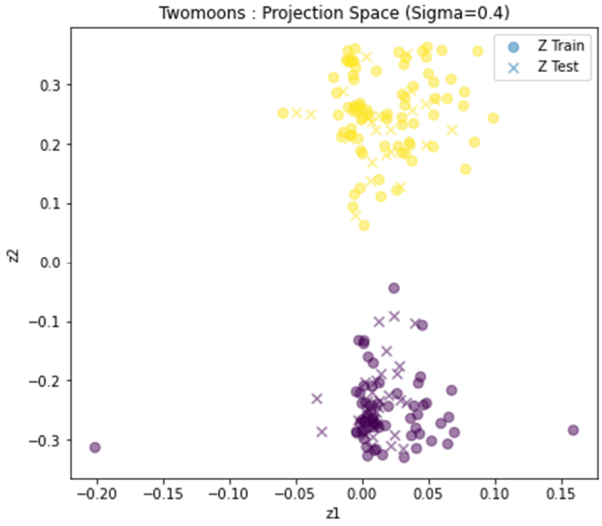
```
--------------------------DataSet : Concentric_rings-----------------------
x_train.shape = (219, 2)
y_train.shape = (219,)
m = 219
n = 2
x_test.shape = (94, 2)
y_test.shape = (94,)
m_test = 94
c = 2
y_unique = [1 2]
```





```
--------------------------DataSet : Twomoons--------------------------
x_train.shape = (140, 2)
y_train.shape = (140,)
m = 140
n = 2
x_test.shape = (60, 2)
y_test.shape = (60,)
m_test = 60
c = 2
y_unique = [-1  1]
```





پاسخ سوالات:

Why do we use kernels in different algorithms?

روش های مبتنی بر کرنل، دسته ای از الگوریتم های آنالیز یا تشخیص پترن هستند که شناخته شده ترین آن ها SVM است. این روش ها، داده ها را به امید اینکه جدا پذیر تر شوند و بتوان آن ها را از هم تفکیک کرد، به ابعاد بالاتری در فضا نگاشت میکنند. هیچ محدودیتی در افزایش ابعاد وجود ندارد و تا بینهایت نیز میتوان ابعاد را بیشتر کرد.

Which kernel had the best results? Why?

به طور کلی انتخاب بهترین کرنل وابستگی بسیاری به نوع مسئله، پیدا کردن پارامتر های بهینه برای کرنل و نوع اطلاعاتی که میخواهیم از داده ها استخراج کنیم دارد. به همین دلیل و بر اساس نتایج بدست آمده، کرنلی را به عنوان بهترین کرنل نمیتوان انتخاب کرد. همانطور که در جدول دقت ها مشاهده میکنید، برای هر مسئله یا دیتاست، یک یا دو کرنل از مجموعه کرنل ها بهترین نتیجه را داده اند.

Which kernels had the same performance? Why?

به طور کلی به دلیل وجود معیار های رندم در محاسبات و متفاوت بودن نتیجه هر کرنل برای هر دیتاست نمیتوان به این سوال پاسخ داد اما اگر برای هر دیتاست و کرنل بطور مجزا بخواهیم بحث کنیم، در دیتاست اول، دوم، سوم و چهارم کرنل Polynomial درجه یک با متد 1NN معمولی نتایج یکسانی داشتند. برای دیتاست پنجم، کرنل های RBF و Polynomial درجه یک تقریبا با متد 1NN معمولی نتایج یکسانی داشتند. اما برای دیتاست ششم، نظر خاصی نمیتوان داد. برای مقایسه بین نتایج بین کرنل ها در دیتاست ها نیز، سلول کرنل های برنده برای هر دیتاست، با حاشیه های سبز نمایش داده شده اند.

What is kernel trick? Why do we use kernels in learning algorithm?

حقه کرنل، یک ابزار قوی برای این مقصود میباشد زیرا پلی میان خطی بودن و غیر خطی بودن برای هر الگوریتمی که بتوان آن را بر اساس ضرب داخلی بیان کرد می باشند. اگر ما در ابتدا داده ها را به فضای بالاتر نگاشت کنیم این کار هزینه زیادی برای ما

دارد، اما در حقه کرنل، نیازی به انجام این کار نیست و این نگاشت هیچوقت محاسبه نمیشود. اگر الگوریتم را بتوان بر اساس ضرب داخلی بین دو وکتور بیان کرد، تمام چیزی که نیاز داریم، جایگزین کردن این ضرب داخلی با ضرب داخلی در فضای مورد نظر می باشد. پس هر جا که ضرب داخلی استفاده میشود، با کرنل جایگزین میگردد. تابع کرنل، ضرب داخلی را در فضای ویژگی ها مطرح میکند. با استفاده از تابع کرنل میتوان الگوریتم در فضای بالاتر اجرا کرد، بدون اینکه داده ها بصورت explicit به این فضای بالاتر نگاشت کنیم حتی اگر این فضای بالاتر، فضای بی نهایت باشد.

What is the difference between Delta and RBF kernels? Explain about the properties of these kernels?

کرنل گاوسی یا RBF یکی از انواع کرنل های مبتنی بر شعاع می باشد. تنظیم کردن پارامتر سیگما، نقش مهمی در عملکرد کرنل دارد و باید با دقت بر اساس مسئله انتخاب شود. اگر بیش از حد تخمین زده شود، تابع نمایی، تقریبا رفتار خطی نشان خواهد داد و نگاشت به ابعاد بالاتر قابلیت غیر خطی بودنش را از دست خواهد داد. از طرفی اگر کم تخمین زده شود، تابع قابلیت Regularization اش را از دست خواهد داد و خط تصمیم نسبت به نویز بسیار حساس خواهد بود.

اما کرنل دلتا، بر روی لیبل ها اعمال میشود و الگوریتم را به سمت محلی که نمونه ها از آنجاست ، هدایت می کند. داده های کلاس های مشابه با هم محکم تر گروه بندی می شوند علاوه بر این ، از کرنل RBF به عنوان کرنل داده در PCA غیرخطی نظارت شده استفاده میشود.