

دانشگاه صنعتي امیرکبیر  
(پلی تکنیک تهران)

دانشكده مهندسی برق

ﮔﺰﺍﺭﺵکار ﺁﺯﻣﺎﻳﺶ پنجم

ﺁﺯﻣﺎﻳﺸﮕﺎﻩ مقدمه ای بر ﻫﻮﺵ ﻣﺤﺎﺳﺒﺎﺗﻲ

پیاده سازی RBF

نگارش

ارشیا اسمعیل طهرانی

علی بابالو

پویا ابراهیمی

استاد راهنما

سرکار خانم موسوی

آذر ماه 1401

# چکیده

در این آزمایش به پیاده سازیRBF پرداخته شده است. يكي از كاربردهاي شبكه عصبي در مساله رگرسيون است. در این آزمایش به تخمین یک تابع پرداخته شده است. اگر تعدادی داده از رفتار یک تابع ناشناخته وجود داشته باشد، شبکه عصبی RBF با استفاده از ترکیب چند تابع گوسین رفتار تابع ناشناخته را با خطای کم پیش بینی خواهد کرد.

|  |  |
| --- | --- |
| فهرست مطالب | صفحه |

**پیش گزارش....................................................................................................................................1**

ساختار یک شبکه عصبی RBF..................................................................................................................1

تفاوت شبکه عصبی جندلایه با شبکه عصبی RBF...............................................................................2

**شرح آزمایش..................................................................................................................................2**

محیط Python..............................................................................................................................................2

تابع گوسین......................................................................................................................................2

پیاده سازی RBF............................................................................................................................4

**تمارین..............................................................................................................................**

# پیش گزارش

## ساختار یک شبکه عصبی RBF.

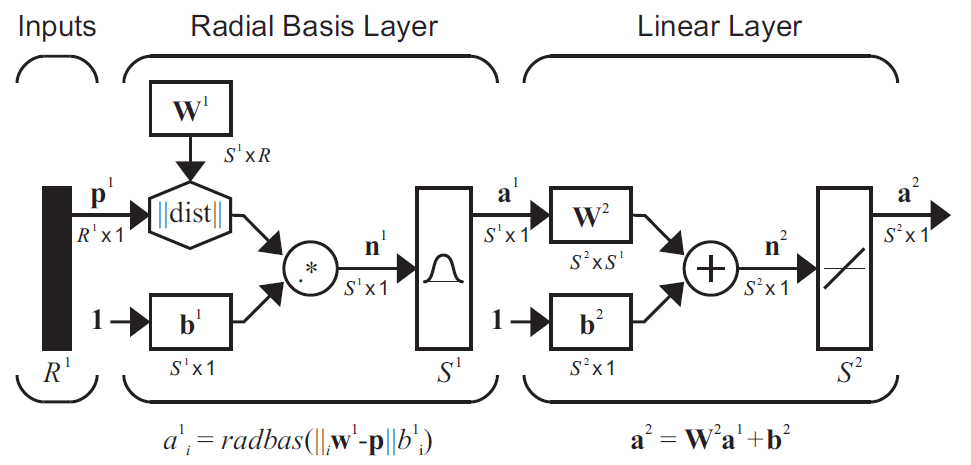
شکل۱) ساختار شبکه RBF

ساختار یک شبکه RBF تا حد زیادی به شکبه عصبی چند لایه شباهت دارد. همانند یک شبکه عصبی تمام پیوستع از 3 لایه تشکیل شده است و دارای ساختار زیر است:

1. لایه اول به عنوان لایه ورودی
2. لایه دوم شامل یک یا چند لایه مخفی است.
3. لایه آخر که لایه خروجی است.

## تفاوت شبکه عصبی چند لایه با شبکه عصبی RBF

تفاوت شبکه عصبی RBF با شبکه عصبی چند لایه، در توابع استفاده شده در لایه مخفی است. در شبکه عصبی RBF از توابع گوسین به عنوان توابع فعالساز برای لایه های مخفی استفاده می‌شود.

شکل۲) شبکه RBF

شبکه RBF به طور کامل در دیاگرام فوق قابل مشاهده است.

# شرح آزمایش

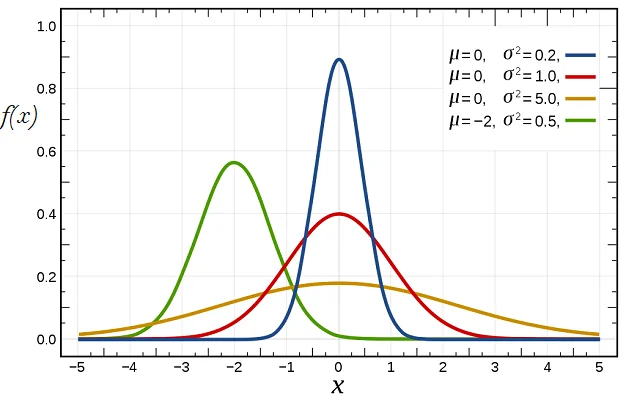
در این قسمت به پیاده سازی شبکه RBF در محیط پایتون نیز پرداخته شده است.

## محیط Python

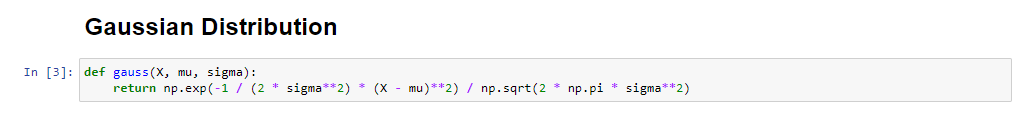
**۲-۱-۱. تابع گوسین**

توابع گاوسی، نوع خاصی از توابع نمایی هستند که در بیان بسیاری از پدیده‌ها کاربرد دارند. معمولا نمای توابع گاوسی به صورت مربع کامل بوده که در مقدار −1  ضرب شده است. به این ترتیب زمانی تابع گاوسی به حداکثر خود می‌رسد که نمای آن کوچکترین مقدار ممکن باشد. دامنه توابع گاوسی اغلب اعداد حقیقی است. این خانواده از توابع به علت تحقیقات زیادی که کارل گاوس (Carl Gauss) در مورد این توابع صورت داد به افتخار او، خانواده توابع گاوسی نامیده می‌شوند. البته گاهی نیز به آن‌ها توابع «زنگی شکل» (Bell Curve) نیز می‌گویند.

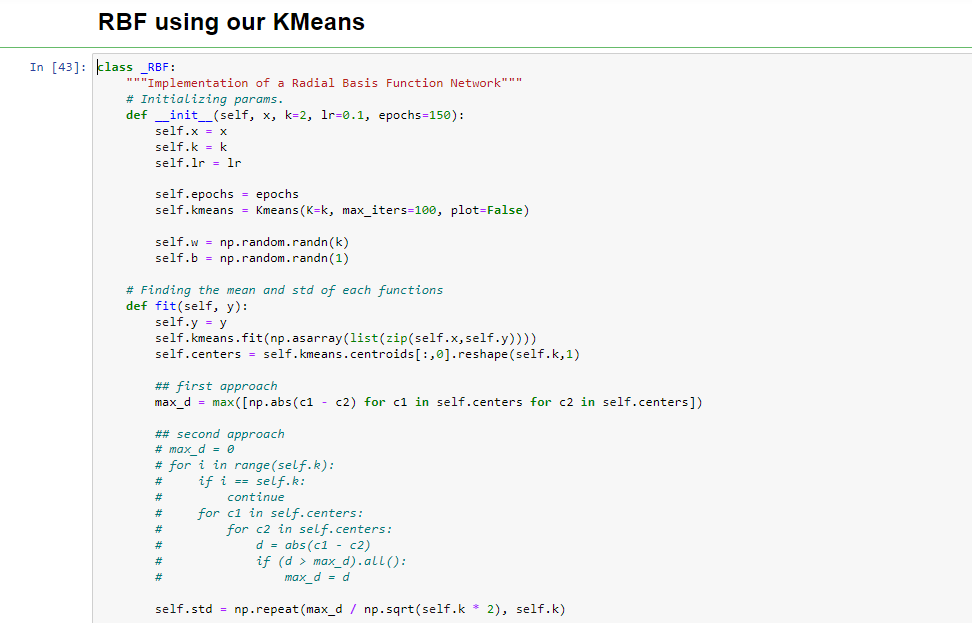
در نظریه احتمال و آمار، تابع گاوسی را به صورت یک تابع چگالی احتمال نشان می‌دهند و به صورت زیر پارامترهای آن را مشخص می‌کنند.

شکل۳) تابع توزیع نرمال با واریانس و میانگین های متفاوت

در مرحله اول پیاده سازی شبکه RBF، تابع گوسین را به شکل زیر پیاده می­کنیم:

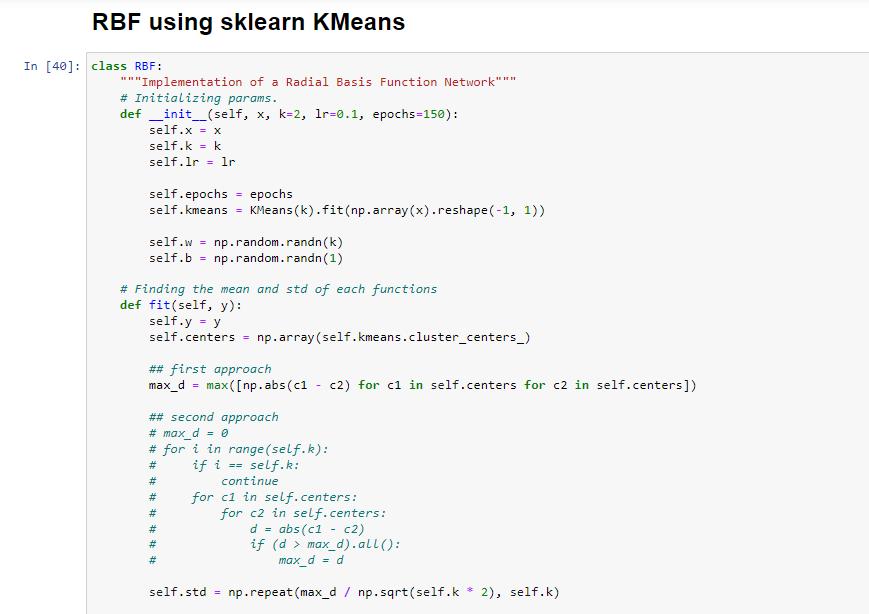
شکل۴) تابع گوسین که مطابق با فرمول اصلی نوشته شده است

**۲-۱-۲ پیاده سازی RBF**

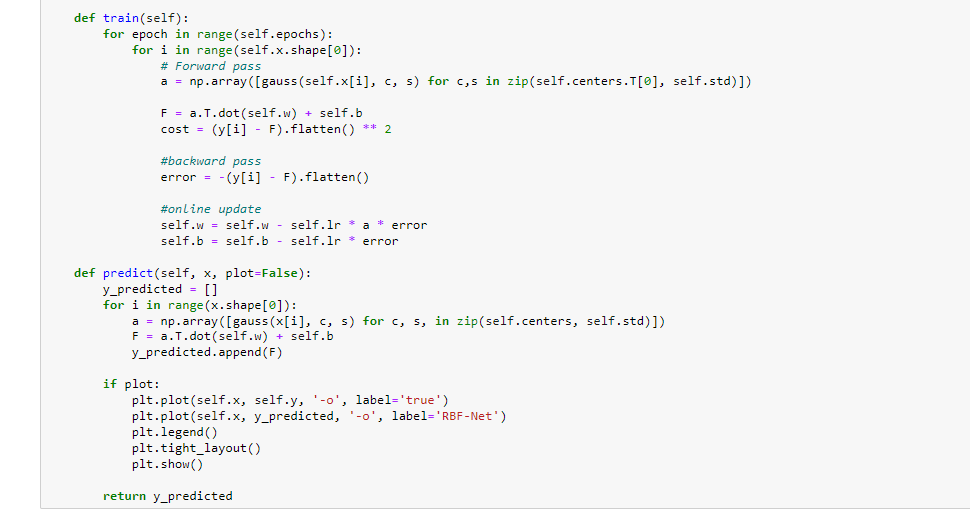
در پیاده سازی شبکه RBF برای پیدا کردن میانگین از الگوریتم KMeans استفاده کردیم که میانگین مختصات داده هایمان در اینجا همان مرکز دسته هایمان در این الگوریتم است. برای پیدا کردن میانگین نیز از دو روش الگوریتم KMeans استفاده کردیم که در اولی از KMeans در کتابخانه sklearn استفاده کردیم و در روش دوم از کد مربوط به kmeans آزمایش قبل خودمان استفاده کردیم. هردو روش یکسان هستند فقط در پیدا کردن مراکز دسته ها بصورت جزیی باهم تفاوت دارند.

شکل۵) کلاس RBF با استفاده از KMeans خودمان

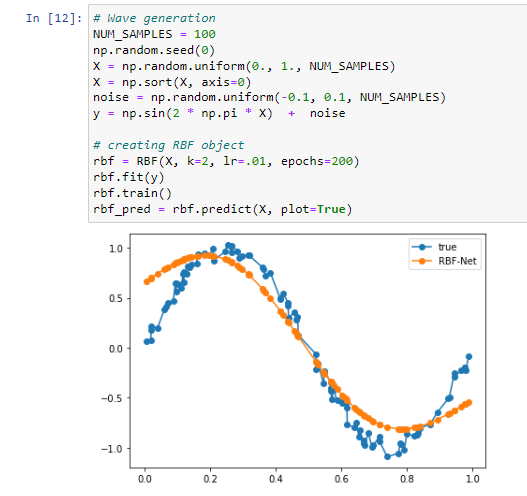
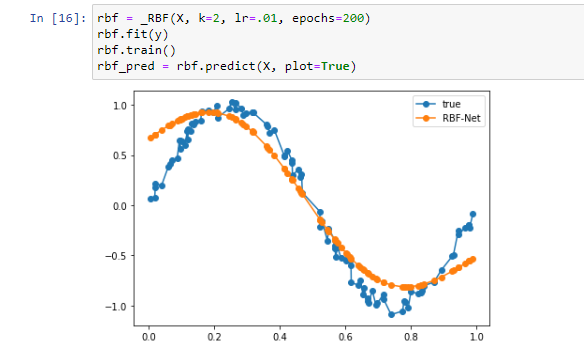
در ابتدا به توضیح کانستراکتور و تابع fit کلاسRBF می­پردازیم. در کانستراکتور تعریف خاصی نداریم فقط کلاس Kmeans خودمان را صدا کردیم و ان را به self.kmeans دادیم. سپس در تابع fit ابتدا با استفاده دیتایمان self.kmeans را fit/predict می­کنیم تا دسته ها مشخص شوند و در ادامه با استفاده از آبجکت centroids موجود در کلاس KMeans مراکز دسته هارا پیدا می­کنیم و فقط المان x آن را به self.centers می­دهیم (چون قرار است yهارا پردیکت کنیم). سپس با دو روش مقدار ماکسیمم فاصله self.Centers هارا پیدا می­کنیم تا از آن در پیدا کردن standard deviation استفاده کنیم، در انتها نیز با استفاده از فرمول گفته شده در دستورکار std را محاسبه می­کنیم.

شکل۶) کلاس RBF با استفاده از کتابخانه sklearn که در آن از توابع خود کتابخانه مراکز را بدست می­آوریم

قسمت اصلی train کرده این الگوریتم در دستور کار قرار داده شده بود که از همان برای آپدیت کردن وزن ها استفاده کردیم.

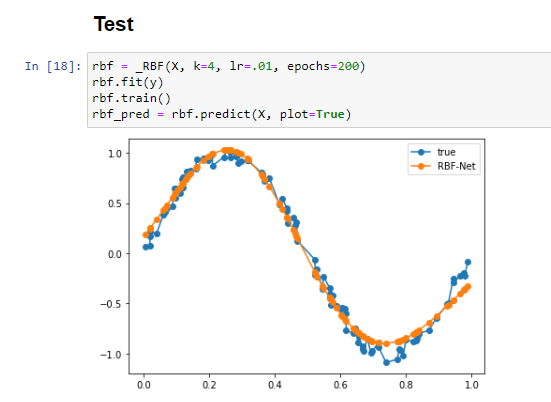
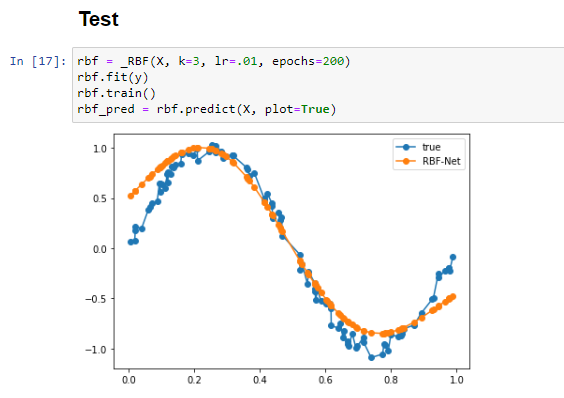
شکل۷) توابع train,predict کلاس RBF

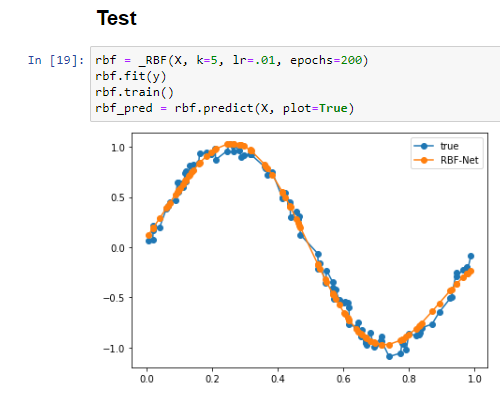
برای آپدیت کردن وزن ها ابتدا مقادیر x و میانگین و انحراف معیار هر نقطه را به تابع گوسینمان میفرستیم و مقادیر را در یک لیست ذخیره می­کنیم. سپس مقدار کاست فانکشنمان را حساب می­کنیم. در قسمت بک پراپگیشن مقدار خطا را محاسبه می­کنیم و از آن در آپدیت کردن وزن ها و بایاس استفاده می­کنیم که در فرمول بالا مشخص هستند. اینکار را به تعداد epoch هایمان تکرار می­کنیم تا وزن ها آپدیت شوند.

در تابع prdict نیز با استفاده از مقدار وزن های نهاییمان مقدار y\_predicted را به ازای هر نقطه حساب می­کنیم و در انتها نمودار اولیه داده های نویزی و مقدار پیشبینی شده را می­کشیم که در شکل پایین مشاهده می­کنیم:

شکل۸) شکل سمت راست نتیجه نهایی با استفاده از kmeans کتابخانه sklean و شکل سمت راست نتیجه نهایی با استفاده از kmeans خودمان است.

در اشکال بالا نتیجه با استفاده از k = 2 بدست آمده است. اگر بخواهیم نتیجه افزایش تعداد کلاستر هارا مشاهده کنیم به اشکال پایین باید دقت کنیم. همانطور که مشاهده می­شود و همانطور که انتظار داشتیم با افزایش تعداد کلاستر ها مقدار پیشبینی دقیق تر می­شود اما بعد از تعدادی کلاستر دچار مشکل overfit می­شویم.



شکل۹) اثر افزایش تعداد کلاسترها با k = 3, 4, 5