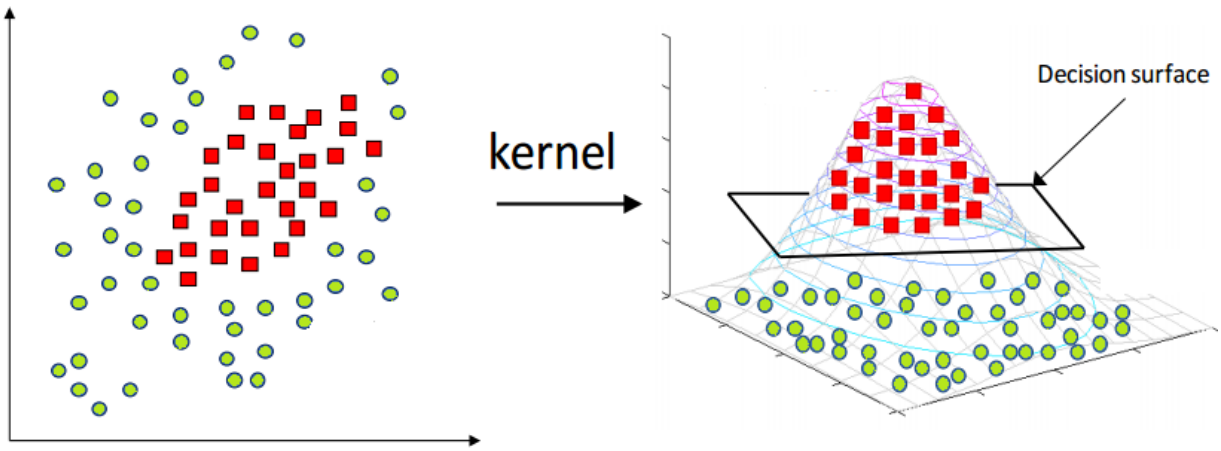
در دنیای هوش مصنوعی، بسیاری از مسائل هستند که نمیتوان آنها را به شکل خطی حل کرد. به عبارت دیگر، نمیتوان یک خط و یا صفحه مستقیم (در ابعاد بالاتر Hyperplane) را در بین داده های موردنظر رسم و آنها را کلاس بندی کرد. در اینگونه از مسائل، متناسب با الگوریتم مورد استفاده روش های متعددی وجود دارد. به عنوان مثال، در شبکه های عصبی میتوان با اضافه کردن لایه های پنهان بیشتر، مسئله را به صورت غیرخطی حل کرد و یا در الگوریتم Support Vector Machine (SVM)، فضای مسئله را تغییر داد.



فرآیند تغییر فضای مسئله به کمک توابع هسته (Kernel Functions) امکان پذیر است که معروف ترین آنها عبارت اند از:

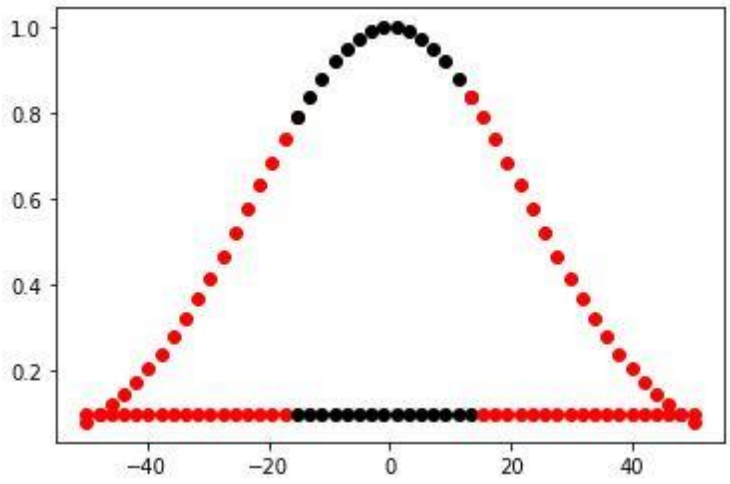
Kernel	Formula
Linear	$\langle \underline{\mathbf{x}}, \underline{\mathbf{z}} \rangle = \underline{\mathbf{x}} \cdot \underline{\mathbf{z}}$
Polynomial	$(\langle \underline{\mathbf{x}}, \underline{\mathbf{z}} \rangle + v)^{\mathrm{d}}$
RBF	$\exp(-\gamma \parallel \underline{\mathbf{x}} - \underline{\mathbf{z}} \parallel^2), \gamma > 0$
General Gaussian	$\exp(-(\underline{\mathbf{x}} - \underline{\mathbf{z}})^{\mathrm{T}} \mathbf{C} (\underline{\mathbf{x}} - \underline{\mathbf{z}}))$
Normalized	$\frac{k(\underline{\mathbf{x}}, \underline{\mathbf{z}})}{\sqrt{k(\underline{\mathbf{x}}, \underline{\mathbf{x}}) \cdot k(\underline{\mathbf{z}}, \underline{\mathbf{z}})}}$
Weighted sum	$\sum_i w_i k_i(\underline{\mathbf{X}}, \underline{\mathbf{Z}}),\ w_i \geq 0$

طرز عملکرد RBF، به این صورت است که داده های ما را از یک فضای قابل تفکیک بصورت غیرخطی (Non-Linear Separable)، به یک فضای قابل تفکیک بصورت خطی (Linear Separable) نگاشت میکند. این کار از طریق مقایسه میزان فاصله هر نقطه از فضا، با نقطه مرکزی (Center) داده ها انجام میشود.

تابع مذکور از رابطه ریاضی زیر پیروی میکند:

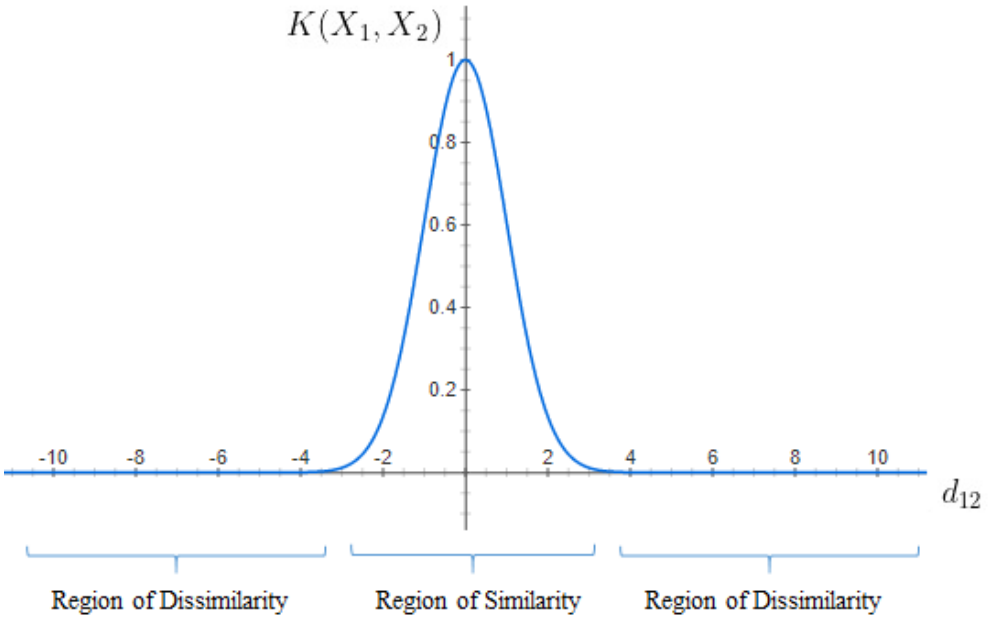
$$K\left(x_i, x_j\right) = \exp\left(-\sigma \left\| x_i - x_j \right\|^2\right), \sigma > 0$$

همانطور که مشاهده میکنید، میزان فاصله اقلیدسی هر نقطه از فضا و نقطه میانی محاسبه شده و سپس عدد € به توان آن میرسد. اینکار باعث میشود تا داده های موردنظر ما برروی نمودار توزیع نرمال (Normal Distribution) و یا یک فضای گاوسی (Gaussian Space) نگاشت و قابل تفکیک بصورت خطی شوند.

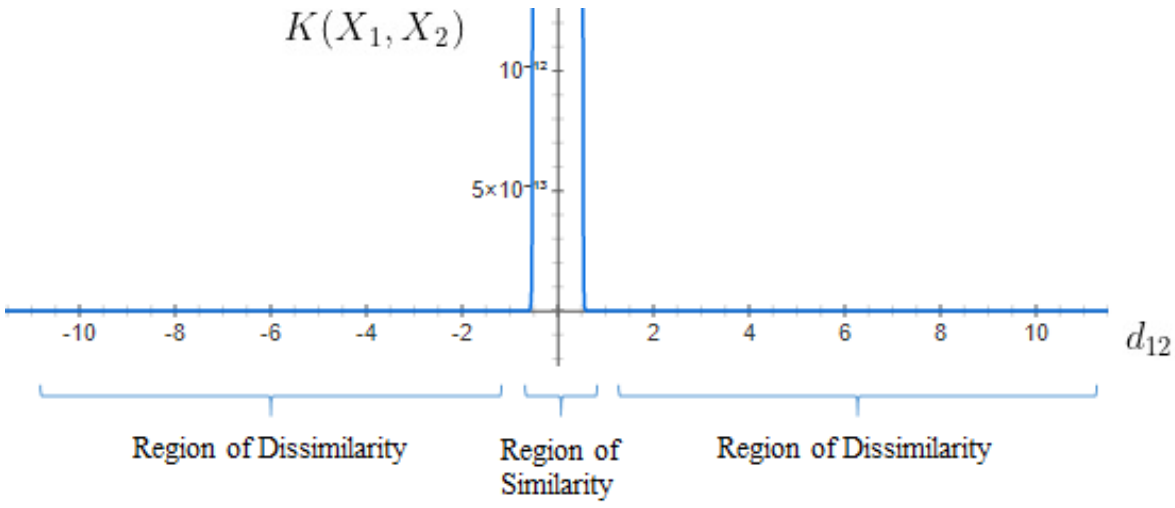


عدد Sigma نیز نوعی Hyperparameter است و باید آن را از طریق آزمون و خطا بدست آورد. اما میتوان نسبت فضا و اندازه آن را به صورت زیر در نظر گرفت:

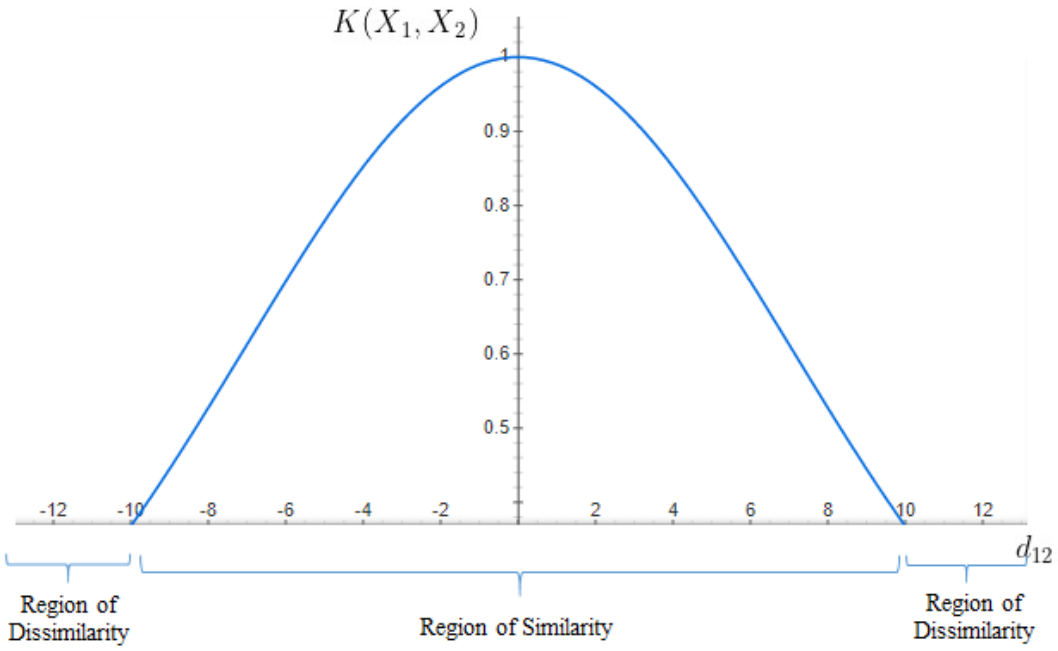
Sigma=1



Sigma=0.1



Sigma=10



تجربه نشان داده است این تابع برای بیشتر مسائل پاسخ میدهد و میتوان از آن استفاده کرد. همچنین، این تابع در برابر توابع دیگر، دارای هایپرپارامترهای کمتری (یک پارامتر سیگما) جهت تنظیم است و میتوان بدلیل گاوسی بودن مقدار خروجی، آن را راحتتر تنظیم و طراحی کرد. ناگفته نماند این تابع دربرابر توابع خطی و چندجمله ای، دقت بیشتری را برای مسئله فراهم میکند.

از فواید دیگر آن میتوان به قدرت تعمیم (Generalization) و تحمل داده های نویزی (Noisy Data) نیز اشاره کرد.

از RBF، در الگوریتم شبکه های عصبی نیز میتوان استفاده کرد. این معماری که نام RBF Network را به خود اختصاص داده است، دارای تنها سه لایه است: لایه ورودی، یک لایه پنهان و لایه خروجی.

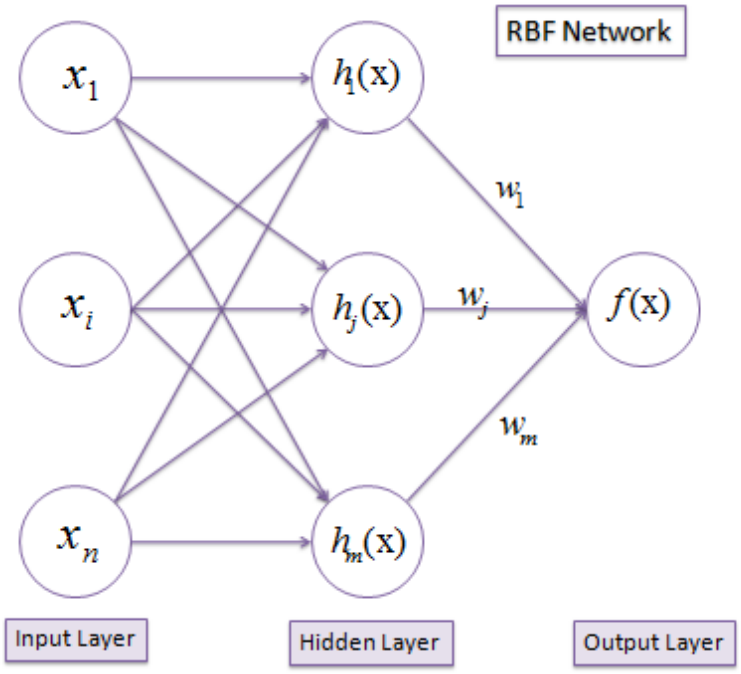
لایه ورودی مربوط میشود به ابعاد و یا ویژگی های ورودی مسئله.

لایه پنهان که شامل RBF به عنوان تابع فعالسازی است.

لایه خروجی که مقدار خروجی و یا کلاس مربوط به داده ورودی را مشخص میکند.

در این معماری، برخلاف معماری های دیگر، لایه ورودی به لایه پنهان بدون هیچ وزنی متصل شده است. به عبارت دیگر، مقادیر ورودی جهت حرکت به سمت لایه پنهان، نیازی به ضرب شدن در بردار (ماتریس) وزن ها ندارند. هر گره از لایه پنهان، شامل تابع RB با یک نقطه مرکزی متفاوت است. جهت به دست آوردن این نقاط، میتوان از الگوریتم های خوشه بندی همانند K-Means استفاده کرد.

مقادیر ورودی به گره های موجود در لایه پنهان منتقل و با یکدیگر جمع شده، سپس تابع RB برروی آنها اعمال و نتایج پس از ضرب در بردار وزن ها، با یکدیگر جمع شده و خروجی را تشکیل میدهند.

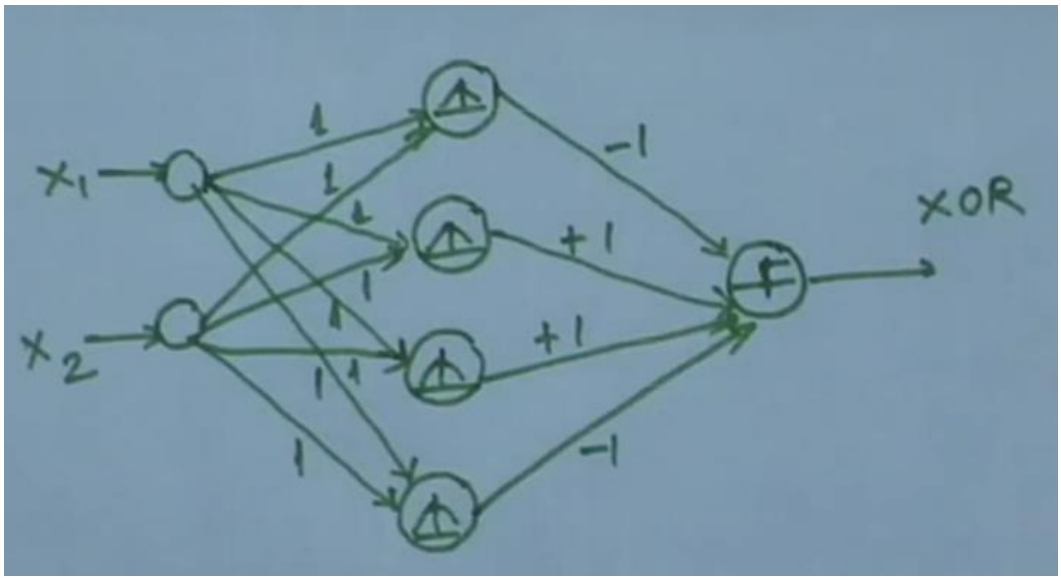


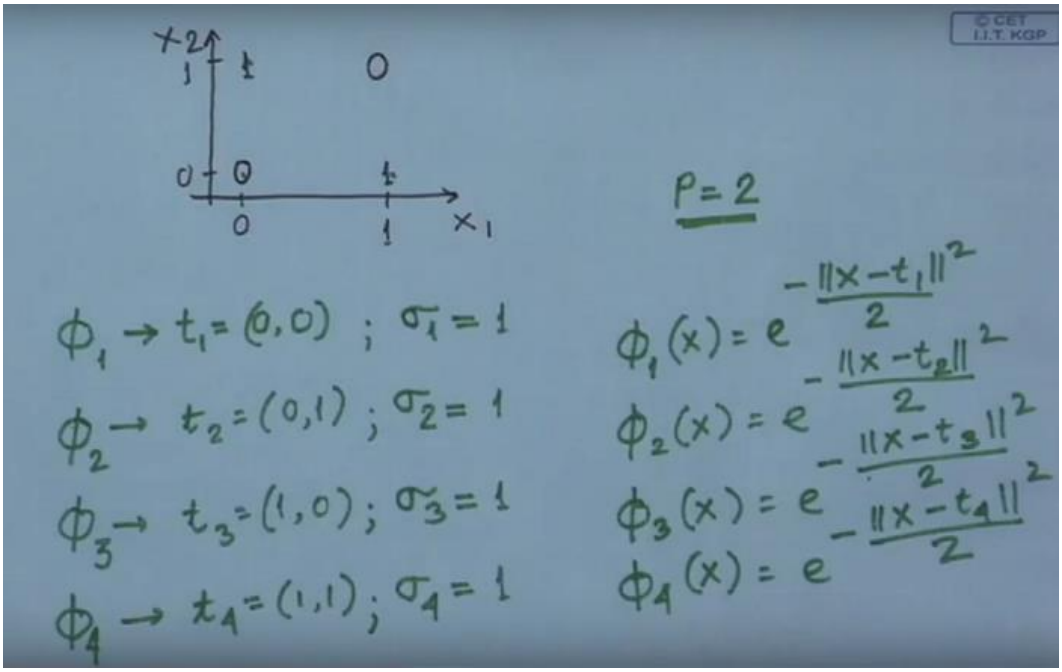
$$f(x) = \sum_{j=1}^{m} w_j h_j(x)$$

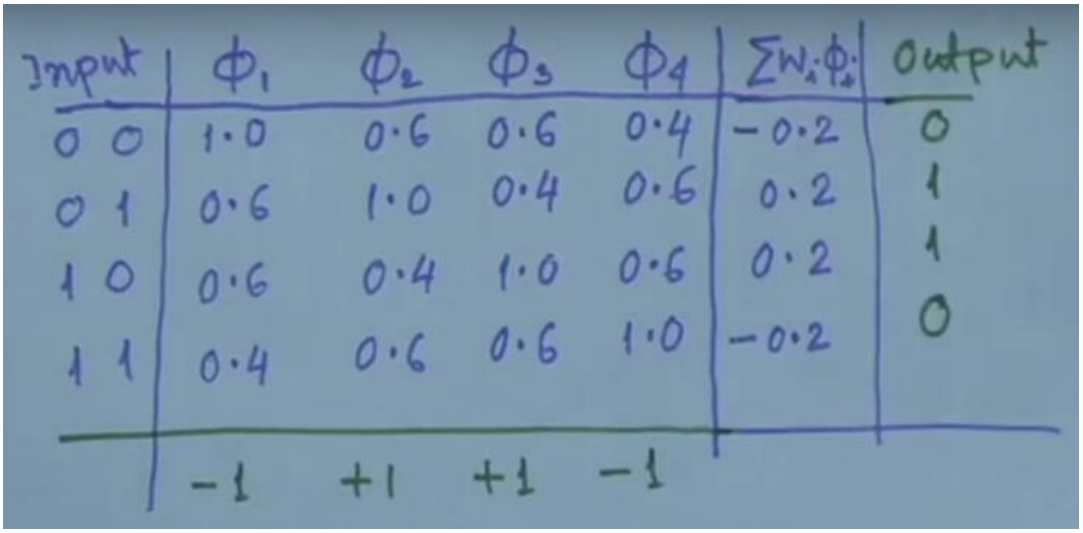
$$h(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right)$$

لازم به ذکر است که به تعداد گره های موجود در لایه پنهان، نقاط مرکزی در داده های موردنظر وجود دارد.

در مراحل زیر، مسئله XOR را میتوان مشاهده کرد که از طریق این الگوریتم حل شده است:







Input	ϕ_1	ϕ_2	ϕ_3	ϕ_4	$\sum w_i \phi_i$	output
0 0	1.0	0.6	0.6	0.4	−0.2	0
0 1	0.6	1.0	0.4	0.6	0.2	1
1 0	0.6	0.4	1.0	0.6	0.2	1
1 1	0.4	0.6	0.6	1.0	−0.2	0
	−1	+1	+1	−1		

از این الگوریتم، به دلیل پیچیدگی کم (تعداد لایه های کمتر نسبت به شبکه های عصبی معمولی) و حل مسائل غیرخطی، در مدلسازی سری های زمانی (Time-Series)، تشخیص الگو (Pattern Recognition) و طراحی سیستم های کنترل (Control Systems) استفاده میشود.