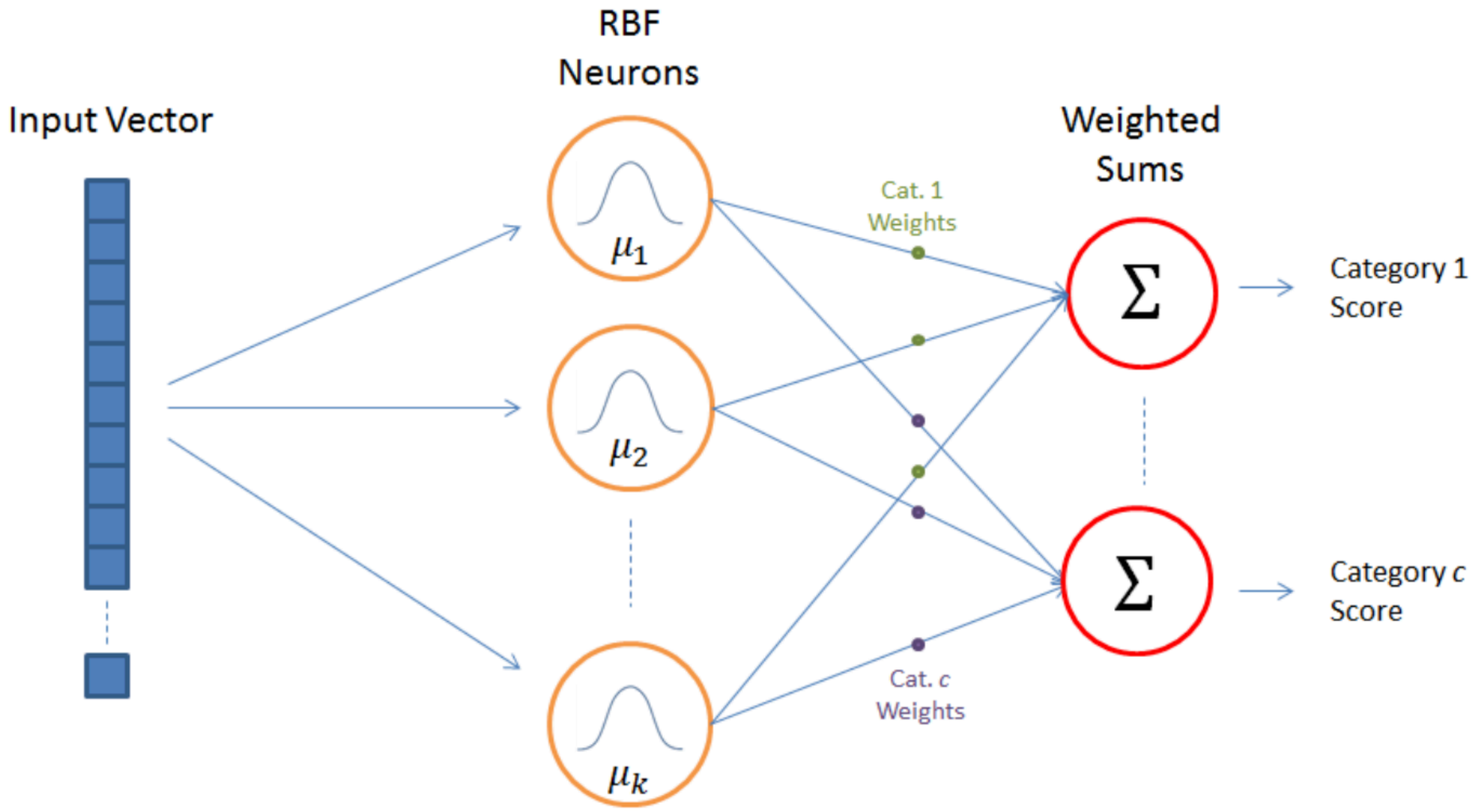
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

آزمایشگاه هوش محاسباتی

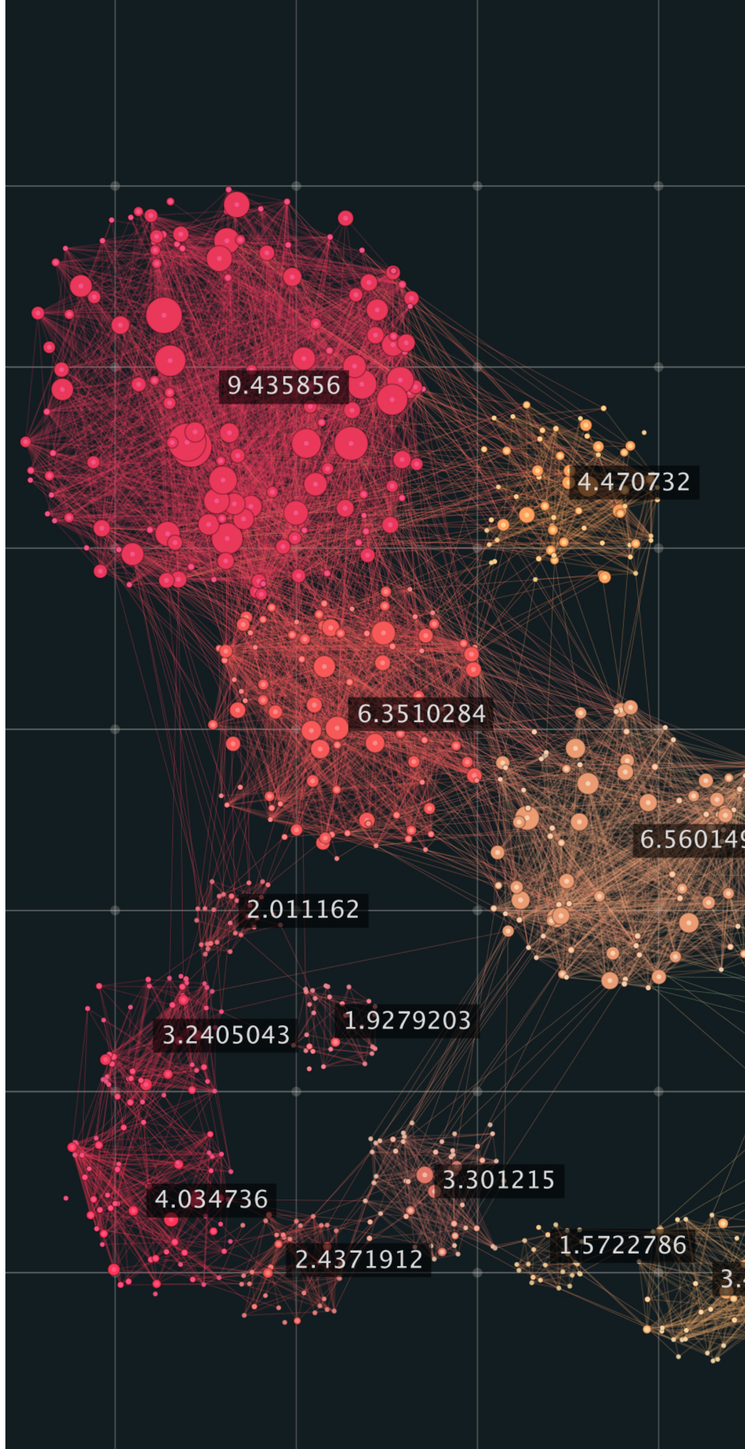
آزمایش پنجم (پیاده سازي شبکه عصبی RBF)

پاییز ۱۴۰۱

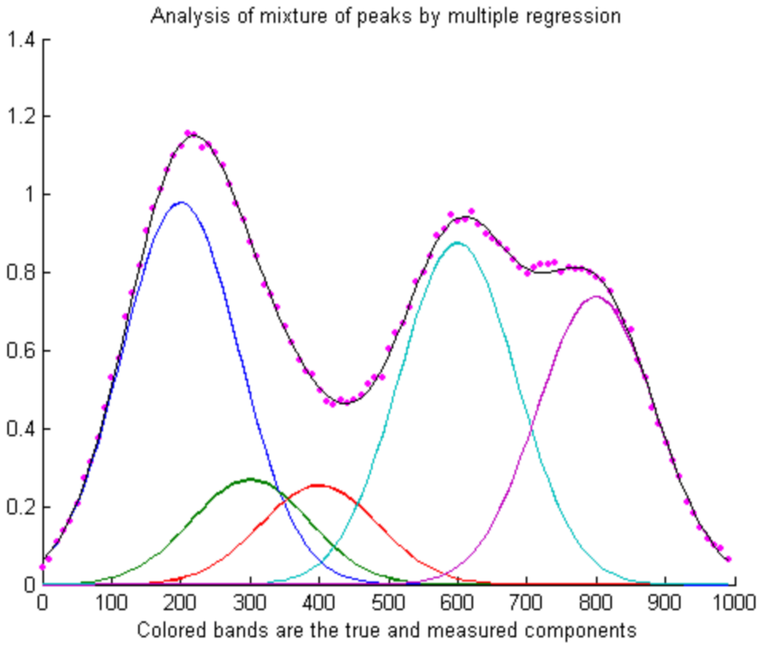
الگوریتم RBF



$$\mathcal{N}(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



الگوریتم RBF



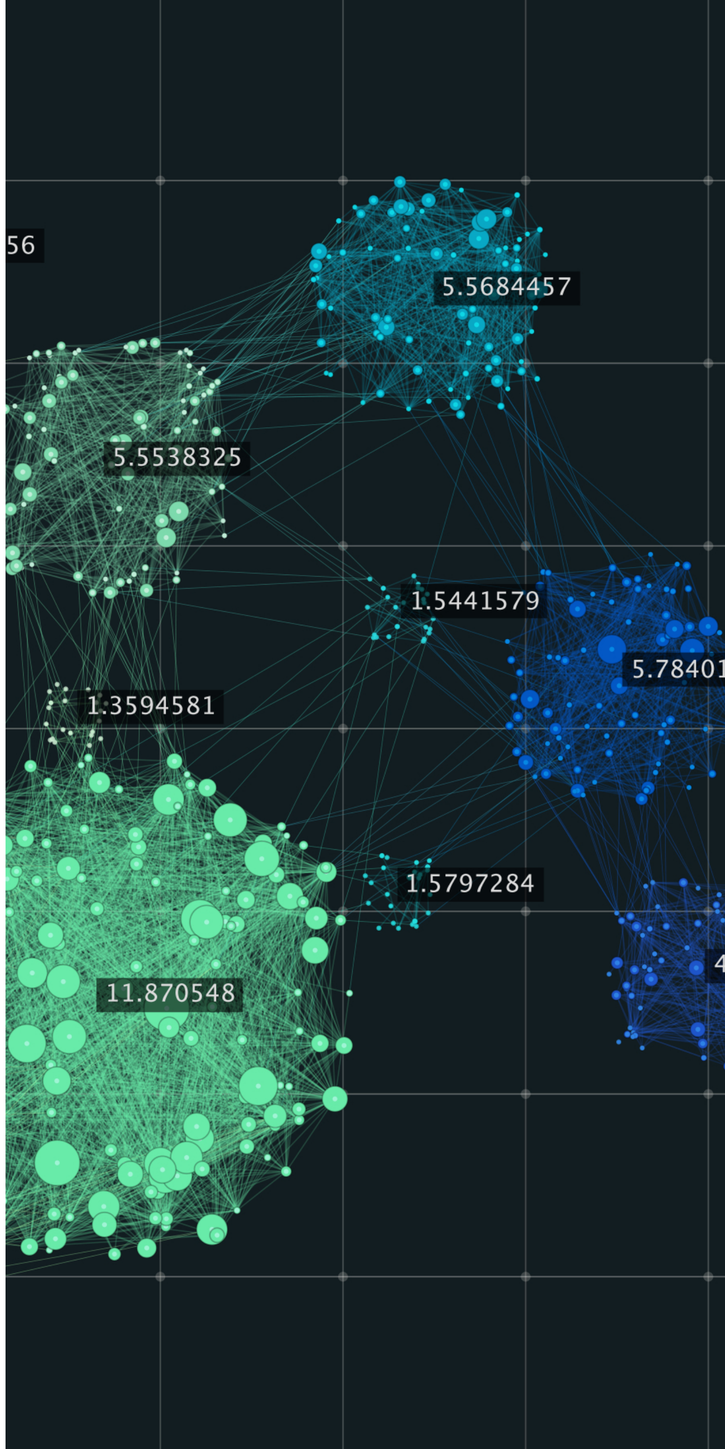
$$\mathcal{N}(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

μ به کمک الگوریتم k-means

$$\sigma = \frac{d}{\sqrt{2}m}$$

بیشترین فاصله بین هر دو دسته دلخواه

M: تعداد دسته ها



تازه‌سازی وزن‌ها

$$F(x) = \sum_{j=1}^{k} w_j \phi_j(x, c_j) + b$$

$$\varphi_j(x, c_j) = \exp\left(\frac{-||x - c_j||^2}{2\sigma_j^2}\right)$$

$$C = \sum_{i=1}^{N} (y^{(i)} - F(x^{(i)}))^2$$

$$\begin{aligned}
\frac{\partial C}{\partial w_j} &= \frac{\partial C}{\partial F}\frac{\partial F}{\partial w_j} \\
&= \frac{\partial}{\partial F}[\sum_{i=1}^{N}(y^{(i)} - F(x^{(i)}))^2] \cdot \frac{\partial}{\partial w_j}[\sum_{j=0}^{K} w_j \varphi_j(x, c_j) + b] \\
&= -(y^{(i)} - F(x^{(i)})) \cdot \varphi_j(x, c_j) \\
w_j &\leftarrow w_j + \eta\ (y^{(i)} - F(x^{(i)}))\ \varphi_j(x, c_j)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial C}{\partial b} &= \frac{\partial C}{\partial F}\frac{\partial F}{\partial b} \\
&= \frac{\partial}{\partial F}[\sum_{i=1}^{N}(y^{(i)} - F(x^{(i)}))^2] \cdot \frac{\partial}{\partial b}[\sum_{j=0}^{K} w_j \varphi_j(x, c_j) + b] \\
&= -(y^{(i)} - F(x^{(i)})) \cdot 1 \\
b &\leftarrow b + \eta\ (y^{(i)} - F(x^{(i)}))
\end{aligned}$$

مراحل انجام آزمایش

۱- تابعی بنویسید که با دریافت مقدار انحراف معیار و میانگین، خروجی تابع گوسین را برای یک نقطه محاسبه نماید.

۲- تابعی بنویسید که برای داده‌های ورودی الگوریتم k-means را اجرا کرده (k تعداد نرون‌های لایه مخفی است) و مراکز کلاسترها را خروجی دهد.

۳- یک کلاس RBF بنویسید به نحوی که تعداد دسته و نرخ یادگیری را دریافت کند و وزن و بایاس را با مقادیر تصادفی پیاده سازی کند.

در این کلاس تابع fit را تعریف نمایید که با دریافت مراکز کلاستر مرحله دوم، میانگین و انحراف معیار هر یک از توابع گوسین لایه مخفی را محاسبه نمایید.

در این کلاس تابع train را تعریف نمایید که با استفاده از قانون پس انتشار خطا، وزن‌های شبکه را آموزش دهد.

در این کلاس تابع predict را بنویسید که خروجی را برای مقادیر جدید پیشبینی نماید.

تست شبکه

برای تست، تعدادی نمونه از یک تابع سینوسی که به نویز آغشته شده‌اند را به عنوان ورودی به شبکه دهید.

```
# sample inputs and add noise
NUM_SAMPLES = 100
X = np.random.uniform(0., 1., NUM_SAMPLES)
X = np.sort(X, axis=0)
noise = np.random.uniform(-0.1, 0.1, NUM_SAMPLES)
y = np.sin(2 * np.pi * X)  + noise
```