
# درس مبانی سیستم های هوشمند

## مینی پروژه دوم

| | |
|---|---|
| نام و نام خانوادگی | حمیدرضا عابدینی |
| شماره دانشجویی | ۴۰۱۲۰۶۳۳ |
| نام و نام خانوادگی | ثمینه هراتیان |
| شماره دانشجویی | ۴۰۱۲۳۸۸۳ |
| تاریخ | آذر ماه ۱۴۰۴ |

گوگل درایو

# فهرست مطالب

# ۱ سؤال اول: سخت‌حاشیه، دوگان و بردارهای پشتیبان

در این سؤال، مدل سخت‌حاشیه‌ی SVM را برای یک مجموعه‌داده‌ی دوکلاسه $\{(x_i, y_i)\}_{i=1}^n$ با $x_i \in R^d$ و $y_i \in \{-1, +1\}$ بررسی می‌کنیم.

## ۱.۱ الف) فرم اولیه‌ی SVM سخت‌حاشیه

فرم اولیه‌ی مسئله‌ی بهینه‌سازی سخت‌حاشیه به صورت زیر است:

$$\min_{w,b} \ \frac{1}{2} \|w\|^2 \tag{۱}$$

با قیود:

$$y_i \left(w^{\mathrm{T}} x_i + b\right) \ \geq 1, \qquad i = 1, \ldots, n. \tag{۲}$$

در این‌جا $w$ بردار نرمال ابرصفحه و $b$ بایاس است و ابرصفحه‌ی جداساز به صورت $w^{\mathrm{T}} x + b = 0$ نوشته می‌شود.

## ۲.۱ ب) عرض حاشیه و رابطه $\frac{2}{\|w\|}$

دو ابرصفحه‌ی مرزی که حاشیه را تعیین می‌کنند عبارت‌اند از:

$$w^{\mathrm{T}} x + b = 1, \tag{۳}$$

$$w^{\mathrm{T}} x + b = -1. \tag{۴}$$

این دو ابرصفحه موازی‌اند و بردار نرمال هر دو همان $w$ است. فاصله‌ی بین دو ابرصفحه‌ی موازی $w^{\mathrm{T}} x + b = c_1$ و $w^{\mathrm{T}} x + b = c_2$ برابر است با:

$$\text{distance} = \frac{|c_1 - c_2|}{\|w\|}. \tag{۵}$$

در این مسئله $c_1 = 1$ و $c_2 = -1$ است، پس:

$$\text{width margin} = \frac{|1 - (-1)|}{\|w\|} = \frac{2}{\|w\|}. \tag{۶}$$

بنابراین عرض حاشیه‌ی SVM سخت‌حاشیه برابر $\frac{2}{\|w\|}$ است.

### ۳.۱ ج) استخراج دوگان و نقش بردارهای پشتیبان

برای استخراج فرم دوگان، لاگرانژ را می‌نویسیم:

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\alpha_i\big(y_i(w^{\mathrm{T}}x_i+b)-1\big), \tag{۷}$$

که در آن $\alpha_i \geq 0$ ضرایب لاگرانژ هستند.
شرط‌های ایستایی (KKT) نسبت به $w$ و $b$:

$$\frac{\partial \mathcal{L}}{\partial w}=0 \;\Rightarrow\; w=\sum_{i=1}^{n}\alpha_i y_i x_i, \tag{۸}$$

$$\frac{\partial \mathcal{L}}{\partial b}=0 \;\Rightarrow\; \sum_{i=1}^{n}\alpha_i y_i=0. \tag{۹}$$

با جای‌گذاری $w$ در $\mathcal{L}$ و حذف $w$ و $b$، فرم دوگان حاصل می‌شود:

$$\begin{aligned}\max_{\alpha}\quad & W(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j\, x_i^{\mathrm{T}}x_j,\\ \text{s.t.}\quad & \alpha_i\geq 0, \qquad i=1,\dots,n,\\ & \sum_{i=1}^{n}\alpha_i y_i=0.\end{aligned} \tag{۱۰}$$

از رابطه‌ی $w=\sum_{i=1}^{n}\alpha_i y_i x_i$ تابع تصمیم به صورت زیر نوشته می‌شود:

$$f(x)=\operatorname{sign}\Big(w^{\mathrm{T}}x+b\Big)=\operatorname{sign}\Big(\sum_{i=1}^{n}\alpha_i y_i\, x_i^{\mathrm{T}}x+b\Big). \tag{۱۱}$$

طبق شرط مکملیت KKT داریم:

$$\alpha_i\big(y_i(w^{\mathrm{T}}x_i+b)-1\big)=0, \qquad i=1,\dots,n. \tag{۱۲}$$

بنابراین:

- اگر $\alpha_i=0$ باشد، نمونه‌ی $i$ در $w$ و تابع تصمیم نقشی ندارد.
- اگر $\alpha_i>0$ باشد، آنگاه $y_i(w^{\mathrm{T}}x_i+b)=1$ و نقطه روی ابرصفحه‌ی حاشیه قرار دارد؛ این نقاط همان بردارهای پشتیبان هستند.

پس در نهایت $w$ و تابع تصمیم فقط به نمونه‌هایی وابسته‌اند که $\alpha_i>0$ دارند:

$$w=\sum_{i:\,\alpha_i>0}\alpha_i y_i x_i, \tag{۱۳}$$

$$f(x)=\operatorname{sign}\Big(\sum_{i:\,\alpha_i>0}\alpha_i y_i\, x_i^{\mathrm{T}}x+b\Big). \tag{۱۴}$$

این نشان می‌دهد که تنها بردارهای پشتیبان (نمونه‌های با $\alpha_i>0$) در تابع تصمیم ظاهر می‌شوند.

## ۲ سؤال دوم: حاشیهٔ نرم با نرم $\ell_1$ و هینج‌لاس

فرض کنید داده‌های آموزشی $\{(x_i, y_i)\}_{i=1}^n$ داده شده‌اند که در آن $x_i \in R^d$ و $y_i \in \{-1, +1\}$. تابع تصمیم خطی را به صورت $f(x) = w^{\mathrm{T}}x + b$ در نظر می‌گیریم.

### ۱.۲ الف) فرم اولیهٔ حاشیهٔ نرم با متغیرهای شل‌شدگی

در حاشیهٔ نرمِ با نرم $\ell_1$ روی متغیرهای شل‌شدگی $\xi_i$، فرم اولیه‌ی SVM به صورت زیر است:

$$\min_{w,b,\xi} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \tag{۱۵}$$

به‌طوری‌که:

$$y_i\big(w^{\mathrm{T}}x_i + b\big) \geq 1 - \xi_i, \qquad i = 1, \ldots, n, \tag{۱۶}$$

$$\xi_i \geq 0, \qquad i = 1, \ldots, n. \tag{۱۷}$$

در این‌جا $\xi_i$ میزان نقض قید حاشیه برای نمونه‌ی $i$ام را اندازه می‌گیرد و $C > 0$ پارامتر تنظیم (regularization) است.

### ۲.۲ ب) اثبات هم‌ارزی با کمینه‌سازی هینج‌لاس

می‌خواهیم نشان دهیم مسئله‌ی بالا معادل است با:

$$\min_{w,b} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\max\big(0,\, 1 - y_i f(x_i)\big), \qquad f(x) = w^{\mathrm{T}}x + b. \tag{۱۸}$$

نکته‌ی کلیدی این است که در مسئله‌ی اولیه، به‌ازای $(w, b)$ ثابت، بهینه‌سازی نسبت به هر $\xi_i$ کاملاً جداپذیر است. برای هر $i$ داریم:

$$\min_{\xi_i} \ C\,\xi_i \quad \text{به‌طوری‌که} \quad \xi_i \geq 0,\ \xi_i \geq 1 - y_i f(x_i). \tag{۱۹}$$

بنابراین $\xi_i$ باید بزرگ‌تر یا مساوی حداکثرِ دو مقدار زیر باشد: $0$ و $1 - y_i f(x_i)$. پس در نقطه‌ی بهینه:

$$\xi_i^* = \max\big(0,\, 1 - y_i f(x_i)\big). \tag{۲۰}$$

اگر این مقدار را در تابع هدف اولیه جای‌گذاری کنیم، به‌جای $\sum_i \xi_i$ عبارت زیر ظاهر می‌شود:

$$\sum_{i=1}^{n}\xi_i^* = \sum_{i=1}^{n}\max\big(0,\, 1 - y_i f(x_i)\big), \tag{۲۱}$$

و تابع هدف دقیقاً به صورت (۱۸) درمی‌آید. پس دو مسئله‌ی $(w, b, \xi)$ با قیود (۱۶)-(۱۷) و مسئله‌ی (۱۸) از نظر مقدار بهینه و $(w, b)$ بهینه کاملاً معادل‌اند.

## ۳.۲ ج) نقش پارامتر $C$ در تعمیم‌پذیری و تعداد بردارهای پشتیبان

در تابع هزینه‌ی (۱۸) دو جزء داریم:

- جزء $\frac{1}{2}\|w\|^2$ که کنترل‌کننده‌ی بزرگی حاشیه و پیچیدگی مدل است (کوچک بودن $\|w\|$ یعنی حاشیه‌ی بزرگ‌تر).
- جزء $\sum_i \max(0, 1 - y_i f(x_i))$ که خطای هینج روی داده‌ی آموزش را اندازه می‌گیرد.

پارامتر $C$ وزن نسبیِ خطای آموزشی نسبت به اندازه‌ی حاشیه را تنظیم می‌کند:

- اگر $C$ بزرگ باشد: جریمه‌ی نقض حاشیه و خطای طبقه‌بندی بسیار زیاد می‌شود، بنابراین بهینه‌ساز سعی می‌کند خطای آموزش را تا حد ممکن صفر کند، حتی اگر لازم باشد حاشیه را کوچک کند (یعنی $\|w\|$ را بزرگ کند). این حالت منجر به حاشیه‌ی باریک، پیچیدگی زیاد و ریسک overfitting می‌شود. چون بسیاری از نقاط روی مرز یا داخل حاشیه قرار می‌گیرند، تعداد زیادی از آن‌ها با $\alpha_i > 0$ به‌عنوان بردار پشتیبان در مدل باقی می‌مانند.
- اگر $C$ کوچک باشد: جریمه‌ی خطاها کم است، بنابراین الگوریتم اجازه می‌دهد بعضی نقاط به‌درستی جدا نشوند یا داخل حاشیه قرار بگیرند، به شرط آن‌که حاشیه تا حد امکان بزرگ شود. در این حالت مدل منظم‌تر است regularization) قوی‌تر)، حاشیه‌ی پهن‌تر و معمولاً تعمیم‌پذیری بهتر روی داده‌های جدید دارد، هرچند ممکن است خطای آموزش بیشتر شود. در این حالت، چون نقاط زیادی خطای کوچک ولی قابل‌قبول دارند، ضرایب دوگان بسیاری از آن‌ها می‌توانند به صفر نزدیک شوند و در نتیجه تعداد مؤثرِ بردارهای پشتیبان کاهش یابد.

به‌طور خلاصه:

- $C$ بزرگ ← تأکید روی loss آموزشی، حاشیه‌ی کوچک‌تر، ریسک overfitting، و معمولاً بردارهای پشتیبان بیشتر.
- $C$ کوچک ← تأکید روی بزرگ بودن حاشیه و منظم‌سازی، خطای آموزش بیشتر ولی تعمیم‌پذیری بهتر، و معمولاً تعداد کمتری بردار پشتیبانِ مؤثر.

# ۳ پرسش سه: نرم $\ell_2$ (شُل‌شدگی مربعی) و تفاوت دوگان با $\ell_1$

## ۱.۳ الف) فرم اولیه‌ی $\ell_2$–soft–margin

داده‌ها را به صورت $\{(x_i, y_i)\}_{i=1}^n$، با $x_i \in R^d$، $y_i \in \{-1, +1\}$ در نظر می‌گیریم و $f(x) = w^{\mathrm{T}}x + b$. فرم اولیه:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i^2 \\ \text{s.t.} \quad & y_i\big(w^{\mathrm{T}}x_i + b\big) \geq 1 - \xi_i, \quad i = 1, \ldots, n, \\ & \xi_i \geq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{۲۲}$$

## ۲.۳ ب) دوگان متناظر

لاگرانژ:

$$\mathcal{L}(w, b, \xi, \alpha) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i^2 - \sum_{i=1}^{n}\alpha_i\big(y_i(w^{\mathrm{T}}x_i + b) - 1 + \xi_i\big), \tag{۲۳}$$

که در آن $\alpha_i \geq 0$ هستند. از شرایط ایستایی:

$$w = \sum_{i=1}^{n}\alpha_i y_i x_i, \tag{۲۴}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0, \tag{۲۵}$$

$$\xi_i = \frac{\alpha_i}{2C}. \tag{۲۶}$$

با جای‌گذاری در لاگرانژ، دوگان:

$$\begin{aligned} \max_{\alpha} \quad & W(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^{\mathrm{T}}x_j - \frac{1}{4C}\sum_{i=1}^{n}\alpha_i^2, \\ \text{s.t.} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0, \\ & \alpha_i \geq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{۲۷}$$

## ۳.۳ ج) تفاوت کران‌های $\alpha_i$ با حالت $\ell_1$

در $\ell_1$-soft-margin دوگان به شکل:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^{\mathrm{T}}x_j \\ \text{s.t.} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C. \end{aligned} \tag{۲۸}$$

در این‌جا $\alpha_i$ بین صفر و $C$ کران‌دار است و راه‌حل معمولاً پراکنده است (تعداد کمی $\alpha_i > 0$). در حالت $\ell_2$ کران بالای صریح وجود ندارد و تنها $\alpha_i \geq 0$ داریم، در عوض در تابع هدف جمله‌ی $-\frac{1}{4C}\sum_i \alpha_i^2$ حضور دارد که از بزرگ شدن $\alpha_i$ها جلوگیری می‌کند و معمولاً به راه‌حلی با تعداد بیشتری بردار پشتیبان ولی با وزن‌های کوچک‌تر منجر می‌شود.

## پرسش: کرنل معتبر و شرایط KKT

### ۴.۳ الف) تعریف کرنل معتبر و ارتباط با نگاشت ویژگی

تابع متقارن $k(x,z)$ کرنل معتبر است اگر برای هر مجموعه‌ی متناهی $\{x_1,\dots,x_n\}$ ماتریس $K_{ij}=k(x_i,x_j)$ نیمه‌مثبت‌معین باشد. در این صورت و فقط در این صورت، نگاشتی $\phi$ وجود دارد که $k(x,z)=\langle \phi(x),\phi(z)\rangle$ را ارضا می‌کند.

### ۵.۳ ب) اعتبار کرنل‌های چندجمله‌ای و **RBF** با خواص بستار

برای کرنل چندجمله‌ای:

$$k_{\text{poly}}(x,z)=(x^{\mathrm{T}}z+c)^d=\sum_{m=0}^{d}\binom{d}{m}c^{d-m}(x^{\mathrm{T}}z)^m, \tag{۲۹}$$

هر $(x^{\mathrm{T}}z)^m$ کرنل معتبر است و ضرایب غیرمنفی‌اند، پس کل جمع کرنل معتبر است.
برای کرنل RBF:

$$k_{\text{RBF}}(x,z)=\exp(-\gamma\|x-z\|^2)=\exp(-\gamma\|x\|^2)\exp(-\gamma\|z\|^2)\exp(2\gamma x^{\mathrm{T}}z), \tag{۳۰}$$

که $\exp(2\gamma x^{\mathrm{T}}z)$ را می‌توان به صورت سری:

$$\exp(2\gamma x^{\mathrm{T}}z)=\sum_{m=0}^{\infty}\frac{(2\gamma x^{\mathrm{T}}z)^m}{m!} \tag{۳۱}$$

نوشت؛ هر جمله کرنل معتبر با ضریب مثبت است، پس جمع نامتناهی و سپس ضرب با دو کرنل معتبر دیگر هم معتبر است.

### ۶.۳ ج) شرایط **KKT** و نقش آن‌ها

شرایط KKT برای SVM با حاشیه‌ی نرم $\ell_1$:

$$w=\sum_{i=1}^{n}\alpha_i y_i x_i, \qquad \sum_{i=1}^{n}\alpha_i y_i=0, \tag{۳۲}$$

$$y_i(w^{\mathrm{T}}x_i+b)\geq 1-\xi_i, \quad \xi_i\geq 0, \tag{۳۳}$$

$$0\leq\alpha_i\leq C, \tag{۳۴}$$

$$\alpha_i\big(y_i(w^{\mathrm{T}}x_i+b)-1+\xi_i\big)=0, \qquad (C-\alpha_i)\xi_i=0. \tag{۳۵}$$

از این‌ها سه نوع حالت برای هر نمونه به‌دست می‌آید:

- $\alpha_i = 0,\ \xi_i = 0 \Rightarrow y_i f(x_i) > 1$: نمونه خارج از حاشیه و غیرِ SV
- $0 < \alpha_i < C,\ \xi_i = 0 \Rightarrow y_i f(x_i) = 1$: SV روی حاشیه
- $\alpha_i = C,\ \xi_i > 0 \Rightarrow y_i f(x_i) < 1$: SV با نقض حاشیه یا بدطبقه‌بندی

# ۴ پرسش چهار: محاسبه‌ی دستی SVM سخت‌حاشیه در $R^2$ با نمونه‌های بیشتر

در فضای $R^2$ کلاس‌ها به‌صورت زیر داده شده‌اند. کلاس مثبت:

$$(2,0),\ (3,0),\ (2,1),\ (2,-1)$$

و کلاس منفی:

$$(-2,0),\ (-3,0),\ (-2,1),\ (-2,-1).$$

## ۱.۴ الف) حدس ابرصفحه‌ی حداکثرحاشیه و معادله‌ی مرز تصمیم

نقاط دو کلاس نسبت به محور $x_2$ متقارن هستند، بنابراین ابرصفحه‌ی حداکثرحاشیه باید خطی عمودی از جنس ثابت $= x_1$ باشد. نزدیک‌ترین نقاط دو کلاس در راستای $x_1$، نقاط با مختصات $x_1 = 2$ و $x_1 = -2$ هستند، پس خط وسط آن‌ها

$$x_1 = 0$$

به‌عنوان مرز تصمیم هندسی انتخاب می‌شود. در فرم $w^{\mathrm{T}}x + b = 0$ این خط متناظر است با

$$w = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ (تا مقیاس)}, \qquad b = 0.$$

در بخش بعد مقیاس دقیق $w$ را با توجه به شرایط SVM تعیین می‌کنیم.

## ۲.۴ ب) محاسبه‌ی دقیق $w$ و $b$

برای SVM سخت‌حاشیه، ابرصفحه‌های مرزی باید

$$w^{\mathrm{T}}x + b = \pm 1$$

باشند و بردارهای پشتیبان روی این ابرصفحه‌ها قرار بگیرند. فرض می‌کنیم نقاط با $x_1 = 2$ و $x_1 = -2$ روی این دو ابرصفحه باشند. با توجه به این‌که $w$ فقط در راستای $x_1$ غیرصفر است، می‌نویسیم

$$w = \begin{pmatrix} w_1 \\ 0 \end{pmatrix}, \qquad b \in R.$$

برای نقطه‌ی مثبت $(2,0)$ داریم

$$w^{\mathrm{T}} \begin{pmatrix} 2 \\ 0 \end{pmatrix} + b = 2w_1 + b = 1,$$

و برای نقطه‌ی منفی $(-2,0)$ داریم

$$w^{\mathrm{T}} \begin{pmatrix} -2 \\ 0 \end{pmatrix} + b = -2w_1 + b = -1.$$

با حل این دستگاه:

$$\begin{cases} 2w_1 + b = 1, \\ -2w_1 + b = -1, \end{cases} \quad \Rightarrow \quad w_1 = \frac{1}{2}, \qquad b = 0.$$

پس

$$w = \begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix}, \qquad f(x) = w^{\mathrm{T}} x + b = \tfrac{1}{2} x_1.$$

مرز تصمیم (ابرصفحه‌ی جداساز) همان خط

$$w^{\mathrm{T}} x + b = 0 \quad \Longleftrightarrow \quad \tfrac{1}{2} x_1 = 0 \quad \Longleftrightarrow \quad x_1 = 0$$

است.

## ۳.۴ ج) بردارهای پشتیبان، حاشیه‌ی هندسی و فاصله‌ی دو ابرصفحه‌ی $w^{\mathrm{T}} x + b = \pm 1$

با $w = (\frac{1}{2}, 0)^{\mathrm{T}}$ و $b = 0$، برای هر نقطه‌ای که $x_1 = 2$ باشد داریم

$$w^{\mathrm{T}} x + b = \tfrac{1}{2} \cdot 2 = 1,$$

و برای هر نقطه‌ای که $x_1 = -2$ باشد داریم

$$w^{\mathrm{T}} x + b = \tfrac{1}{2} \cdot (-2) = -1.$$

بنابراین بردارهای پشتیبان عبارت‌اند از:

کلاس مثبت: $\{(2,0),\ (2,1),\ (2,-1)\}$

کلاس منفی: $\{(-2,0),\ (-2,1),\ (-2,-1)\}$

نقاط $(3,0)$ و $(-3,0)$ خارج از حاشیه قرار دارند و بردار پشتیبان نیستند.

نورم بردار وزن:

$$\|w\| = \left\| (\tfrac{1}{2}, 0) \right\| = \tfrac{1}{2}.$$

در SVM سخت‌حاشیه، عرض حاشیه (فاصله‌ی بین دو ابرصفحه‌ی $w^{\mathrm{T}} x + b = 1$ و $w^{\mathrm{T}} x + b = -1$) برابر است با

$$\text{width margin} = \frac{2}{\|w\|} = \frac{2}{1/2} = 4.$$

در نتیجه فاصله‌ی هر ابرصفحه‌ی حاشیه تا ابرصفحه‌ی جداساز $w^{\mathrm{T}} x + b = 0$ برابر است با

$$\frac{1}{\|w\|} = 2.$$

# ۵ پرسش پنج: طبقه‌بندی دسته‌بندی غیرخطی با شبکه‌های عصبی

دیتاست دوکلاسه‌ی شکل را در نظر بگیرید که در آن داده‌های آموزشی اولیه متعلق به دو کلاس مجزا بوده و چندین نقطه‌ی جدید (به‌صورت ستاره‌های سیاه) در ناحیه‌ای مرکزی به آن افزوده شده‌اند. هدف طراحی مدلی مناسب برای طبقه‌بندی صحیح این داده‌ها است.

## ۱.۵ الف) انتخاب نوع شبکه‌ی عصبی

به‌دلیل این‌که هر کلاس به‌صورت چند خوشه‌ی مجزا در صفحه ظاهر شده است (الگوی تقریباً XOR دوبعدی)، مرز تصمیم خطی یا حتی ترکیب ساده‌ی چند مرز خطی برای این داده مناسب نیست. مدلی لازم است که بتواند نواحی محلی را جداگانه مدل کند.

یک شبکه‌ی عصبی RBF یا Network Function Basis Radial با توابع گاوسی به صورت

$$h_j(x) = \exp\left(-\frac{\|x - c_j\|^2}{2\sigma_j^2}\right), \qquad f(x) = \sum_{j=1}^{m} w_j h_j(x) + b, \tag{۳۶}$$

و تابع تصمیم

$$\hat{y}(x) = \operatorname{sign}\big(f(x)\big) \tag{۳۷}$$

انتخاب مناسبی است. با انتخاب مراکز $c_j$ در نزدیکی خوشه‌ها (مثلاً چهار مرکز برای چهار خوشه‌ی اصلی)، هر نرون مخفی تنها در حوالی خوشه‌ی خود فعال می‌شود و مرز تصمیم حاصل از ترکیب این نواحی محلی، شکل غیرخطی مورد نیاز را به‌خوبی تقریب می‌زند.

## ۲.۵ ب) تحلیل ضعف مدل‌های **MLP**/سیگموییدی در این مسئله

یک MLP با یک لایه‌ی مخفی و تابع سیگموید را می‌توان به صورت

$$f(x) = v^{\mathrm{T}} \sigma(Wx + b_h) + b_o \tag{۳۸}$$

نوشت که در آن هر نرون مخفی

$$h_j(x) = \sigma(w_j^{\mathrm{T}} x + b_j) \tag{۳۹}$$

است. در این حالت، هر نرون مخفی معادله‌ی یک ابرصفحه را تعیین می‌کند و خروجی نهایی ترکیبی خطی از چند سیگموید است؛ در فضای دوبعدی، این ترکیب معمولاً مرز تصمیمی «جهانی» و نسبتاً هموار (ترکیب چند خط خمیده) ایجاد می‌کند.

در این دیتاست:

- ساختار داده تقریباً XOR دوبعدی است؛ دو خوشه‌ی آبی به‌صورت قطری روبه‌روی هم و دو خوشه‌ی قرمز نیز به همین صورت قرار دارند. جداسازی چنین الگویی با مرزهای جهانی حاصل از تعداد کمی نرون سیگموییدی دشوار است و به تعداد زیادی نرون مخفی نیاز دارد.
- ناحیه‌ی مرکزی که ستاره‌ها در آن قرار گرفته‌اند، در داده‌ی آموزشی خالی است. تابع هزینه

$$\min_{\theta} \sum_{i} \ell\big(y_i, f_\theta(x_i)\big) \tag{۴۰}$$

فقط روی نقاط آموزشی کمینه می‌شود؛ بنابراین در ناحیه‌ای که داده‌ای وجود ندارد، مقدار $f_\theta(x)$ صرفاً توسط سوگیری مدل in-bias ductive تعیین می‌شود. برای MLP، این سوگیری معمولاً به سمت یک رفتار تقریباً خطی بین خوشه‌ها است، لذا مرز تصمیم در مرکز به‌صورت یک خط یا منحنی ساده در می‌آید و برخی از نقاط ستاره‌ای به‌اشتباه طبقه‌بندی می‌شوند.

در مقابل، در شبکه‌ی RBF با مراکز $c_j$ روی خوشه‌ها، هر تابع پایه‌ی شعاعی

$$h_j(x) = \exp\left(-\frac{\|x - c_j\|^2}{2\sigma_j^2}\right) \tag{۴۱}$$

در بیرون از ناحیه‌ی خوشه‌ی مربوطه تقریباً صفر می‌شود، و در ناحیه‌ی مرکزی که از همه‌ی مراکز دور است، همه‌ی $h_j(x)$ کوچک‌اند. بنابراین

$$f(x) \approx b \tag{۴۲}$$

و با تنظیم مناسب $b$ و وزن‌ها $w_j$، می‌توان مرزی ساخت که بهتر با ساختار خوشه‌ای داده سازگار باشد. به‌علاوه، با افزودن چند مرکز بیشتر در حوالی ناحیه‌ی مرکزی، می‌توان مرز تصمیم را بسیار دقیق‌تر با پراکندگی نقاط تطبیق داد.

به‌طور خلاصه، MLP با توابع فعال‌سازی سیگموید بیشتر برای مرزهای تصمیم جهانی و نسبتاً هموار مناسب است و در تفکیک چند خوشه‌ی جدا و ناحیه‌های خالی بین آن‌ها به شبکه‌ای بزرگ و پیچیده نیاز دارد؛ در حالی که شبکه‌ی RBF به‌طور طبیعی مرزهای تصمیم محلی و خوشه‌محور ایجاد می‌کند و برای این دیتاست عملکرد بهتری دارد.

# ۶ سوال ششم

## قسمت اول:

### ۱.۶ بخش اول: شناسایی سیستم با شبکههای عصبی RBF

در این سوال ما باید یک سیستم ساده شده از نوع beam and ball مدلسازی کنیم و تخمین بزنیم. تخمین ما بر اساس یک شبکه RBFNN خواهد بود، که بر اساس بردار ورودی و خروجی زیر خواهد بود.

$$y(t) = \frac{y(t-1)}{1+y(t-2)} + u(t-3)^3$$

$$x(t) = [y(t1), y(t2), u(t1)]^T, y(t)$$

#### ۱.۱.۶

با توجه به خواسته سوال ورودی u(t) یک موج سینوسی خواهد بود، پس ما در این بخش ۱۰۰۰ sample تولید خواهیم کرد. پس با توجه به شرایط زیر دادههای ورودی و خروجی را میسازیم:

$$u(t) = sin(2\pi t/250)$$

$$y(0) = y(1) = 0, t = [2, 999]$$

#### ۲.۱.۶

در این بخش دادهها به دو بخش آموزش و آزمایش تقسیم میکنیم.

#### ۳.۱.۶

تعریف مسئلهی تقریب تابع

در تقریب تابع، یک تابع ناشناخته در اختیار داریم که تنها ورودیها و خروجیهای مشاهدهشدهی آن قابل دسترس است، اما شکل تحلیلی دقیق آن مشخص نیست. هدف این است که با استفاده از دادههای موجود، تابعی بیاموزیم که ورودیها را به خروجیها نگاشت کند.

شبکههای RBFNN دقیقاً برای چنین مسئلهای طراحی شدهاند؛ یعنی شبکههایی مبتنی بر توابع پایهی شعاعی که هدف آنها تقریب یک تابع غیرخطی ناشناخته است.

چرا مسئلهی حاضر یک مسئلهی تقریب تابع است؟

در این پروژه، سیستم دینامیکی واقعی توسط رابطهی زیر توصیف میشود:

$$y(t) = \frac{y(t-1)}{1+y(t-2)} \; + \; u(t-1)^3$$

اما شبکه‌ی RBFNN این معادله را نمی‌بیند و تنها جفت داده‌های ورودی/خروجی را دریافت می‌کند:

$$x(t) = [\, y(t-1),\ y(t-2),\ u(t-1)\,]$$

$$\text{target}(t) = y(t)$$

بنابراین از دید شبکه، مسئله این است که از روی داده‌های مشاهده‌شده، یک تابع ناشناخته $f(\cdot)$ را بیاموزد. این همان تعریف دقیق مسئله‌ی تقریب تابع است.

تابعی که شبکه‌ی **RBFNN** باید یاد بگیرد

شبکه باید نگاشت غیرخطی سیستم را از فضای سه‌بعدی ورودی‌ها به خروجی تک‌بعدی تقریب بزند:

$$f : R^3 \to R$$

شکل واقعی این تابع همان رفتار دینامیکی سیستم است:

$$f\big(y(t-1),\, y(t-2),\, u(t-1)\big) = \frac{y(t-1)}{1+y(t-2)} + u(t-1)^3$$

هدف شبکه این است که تقریب بزند:

$$\hat{y}(t) = \hat{f}\big(x(t)\big) \approx f\big(x(t)\big)$$

به عبارت دیگر، ما یک نگاشت غیرخطی از سه ورودی به یک خروجی داریم و شبکه‌ی RBFNN باید این تابع ناشناخته را تنها از روی داده‌ها یاد بگیرد؛ این دقیقاً همان مسئله‌ی تقریب تابع است.

## ۲.۶ بخش دوم: پیاده سازی RBFNN ایستا

### ۱.۲.۶

نقش مرکز $\mu_k$ و عرض $\sigma_k$ در نورون‌های RBF

در یک شبکه‌ی RBFNN، هر نورون لایه‌ی پنهان یک تابع گوسی تولید می‌کند:

$$\phi_k(x) = \exp\left(-\frac{\|x-\mu_k\|^2}{\sigma_k^2}\right)$$

این تابع توسط دو پارامتر کلیدی کنترل می‌شود:

$\mu_k$ (مرکز نورون)

$\sigma_k$ (عرض یا پهنای نورون)

این دو پارامتر مشخص می‌کنند که نورون در کدام ناحیه از فضای ورودی فعال شود و میزان فعال‌سازی آن چقدر باشد.

نقش مرکز نورون $\mu_k$

مرکز $\mu_k$ یک بردار در فضای ورودی است:

$$\mu_k \in R^d$$

در مسئله‌ی حاضر:

$$d = 3$$

و مرکز هر نورون به صورت زیر تعریف می‌شود:

$$\mu_k = \begin{bmatrix} \mu_{k1} \\ \mu_{k2} \\ \mu_{k3} \end{bmatrix}$$

نقش مرکز

مرکز تعیین می‌کند که نورون در کدام نقطه از فضای ورودی بیشترین پاسخ را بدهد.

اگر ورودی با مرکز برابر باشد:

$$x = \mu_k \;\Rightarrow\; \|x - \mu_k\| = 0 \;\Rightarrow\; \phi_k(x) = 1$$

بنابراین نورون در مرکز خود بیشترین فعال‌سازی را دارد و با دور شدن ورودی از مرکز، خروجی گوسی کاهش می‌یابد.

تأثیر تغییر مرکز

تغییر $\mu_k$ باعث جابه‌جایی نورون در فضای ورودی می‌شود، بدون اینکه پهنای آن تغییر کند.

بنابراین:

- انتخاب مناسب مرکز باعث پوشش نواحی مهم داده می‌شود.
- اگر $\mu_k$ خیلی دور از داده‌ها باشد، نورون تقریباً هیچ‌گاه فعال نمی‌شود.

نقش عرض نورون $\sigma_k$

عرض $\sigma_k$ یک عدد مثبت است:

$$\sigma_k > 0$$

نقش عرض

پهنای گوسی تعیین می‌کند که نورون چه میزان گستردگی در فضای ورودی داشته باشد:

- $\sigma_k$ کوچک  تابع گوسی باریک  فعال‌سازی فقط نزدیک مرکز  رفتار موضعی (Local)
- $\sigma_k$ بزرگ  تابع گوسی پهن  فعال‌سازی برای ناحیه‌ی وسیع‌تر  رفتار کلی‌تر (Global)

تأثیر تغییر عرض

اگر $\sigma_k$ کوچک باشد:

- نورون تنها به ورودی‌های خیلی نزدیک مرکز واکنش نشان می‌دهد.
- شبکه رفتار بسیار محلی پیدا می‌کند.
- ریسک بیش‌برازش (Overfitting) افزایش می‌یابد.

اگر $\sigma_k$ بزرگ باشد:

- تابع گوسی برای طیف گسترده‌ای از ورودی‌ها فعال می‌شود.
- نورون رفتار کلی‌تر نشان می‌دهد.
- اگر مقدار بیش از حد بزرگ باشد، تمام نورون‌ها تقریباً خروجی مشابه می‌دهند و شبکه قدرت تفکیک خود را از دست می‌دهد.

### ۲.۲.۶ ساخت تابع فعال‌ساز

صورت سؤال می‌گوید:
ورودی:

$$[1, 1]$$

مرکز:

$$[1, 1]$$

عرض: $\sigma = 1$
خروجی:

$$\phi = 1.0$$

(دقیقاً برابر ۱)
در RBF گوسی:
حداکثر مقدار فعال‌سازی گوسی برابر ۱ است.
این مقدار دقیقاً زمانی رخ می‌دهد که ورودی و مرکز نورون یکسان باشند.
بنابراین:
نورون RBF در مرکز خودش بیشترین پاسخ ممکن را می‌دهد.
این یعنی:
ورودی دقیقاً در «مرکز» نورون قرار دارد.
این نورون بیشترین مشارکت را در خروجی شبکه خواهد داشت.
این نقطه همان محلی است که نورون برای پوشش‌دهی آن ساخته شده است.

## ۳.۶ بخش سوم: آموزش از طریق حداقل مربعات خطی (LLS)

با توجه به توضیحات این بخش از سوال ما باید دستگاه معادلات زیر را حل کنیم:

$$\Phi\alpha = Y$$

$$\Phi_{ij} = \phi_j(x_i)$$

### ۱.۳.۶

مراحلی که ما در این بخش باید بصورت کد پیاده کنیم به شرح زیر است که پیاده‌سازی را انجام خواهیم داد، سپس نتایج را بررسی خواهیم کرد.
ساخت ماتریس $\Phi$
ماتریس $\Phi$ از فعالسازهای RBF روی ورودی‌ها ساخته می‌شود. اگر داده‌های آموزشی ما

$$X_{\text{train}} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

باشد و $K$ مرکز $\mu_k$ داشته باشیم، آنگاه:

$$\Phi_{ij} = \phi_j(x_i) = \exp\left( - \frac{\|x_i - \mu_j\|^2}{\sigma^2} \right)$$

انتخاب مراکز و عرض‌ها

- مراکز $\mu_k$: از بین داده‌های آموزش، $K$ نمونه تصادفی انتخاب می‌کنیم.
- عرض $\sigma_k$: همه نورون‌ها یک عرض ثابت دارند. معمولاً می‌توان از میانگین فاصله بین مراکز استفاده کرد:

$$\sigma = \text{mean}(\|\mu_i - \mu_j\|), \quad i \neq j$$

آموزش شبکه با حداقل مربعات خطی (LLS)
با داشتن ماتریس $\Phi$ و بردار خروجی هدف $Y_{\text{train}}$، وزن‌های خروجی $\alpha$ با حل دستگاه خطی زیر به دست می‌آیند:

$$\alpha = (\Phi^T \Phi)^{-1} \Phi^T Y_{\text{train}}$$

پیش‌بینی خروجی
خروجی شبکه برای مجموعه آموزش و تست:

$$\hat{Y} = \Phi \alpha$$

برای مجموعه تست، ماتریس $\Phi$ تست مشابه ساخته می‌شود اما با مراکز و $\sigma$ ثابت.
معیار ارزیابی
ریشه میانگین مربعات خطا :(RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}$$

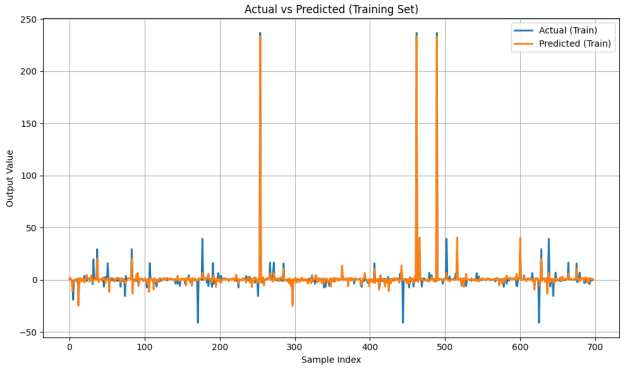
ما مقدار $K = 30$ قرار می‌دهیم، همچنین مقدار $\sigma = 23$ می‌شود و سپس بعد از پیاده‌سازی الگوریتم‌های بالا، به بررسی نتایج می‌پردازیم:

$$\text{RMSE}_{\text{train}} = 4.6833$$

$$\text{RMSE}_{\text{test}} = 4.9848$$

ما در این بخش مقدار خطایی که بدست آوردیم بدون نرمال‌سازی داده‌ها بود که اعداد منطقی هستند با توجه به بازه $y$ که وجود دارد.

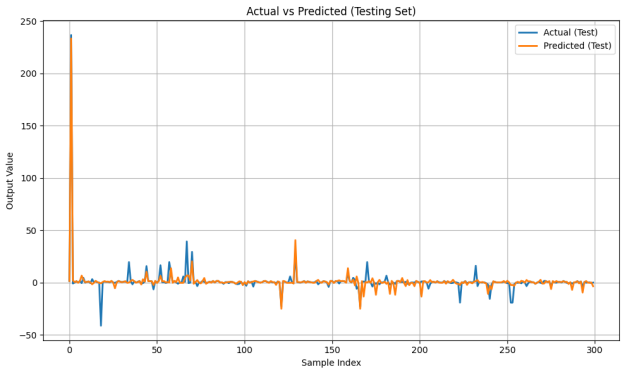
۲.۳.۶




شکل ۱: Predicted vs Actual (train)

همانطور که در شکل ۱ مشاهده می‌کنیم توانسته‌ایم تخمین مناسبی بزنیم.

۳.۳.۶




شکل ۲: Predicted vs Actual (test)

با توجه به شکل ۲ می‌توان برداشت کرد که خطا در مناطق ابتدایی و انتهایی زیاد می‌باشد، همچنین در مکان‌هایی که نوسان داده‌ها زیاد است و تغییر ناگهانی و بیشتری خواهیم داشت خطا همچنان افزایش پیدا می‌کند. همچنین اگر $\sigma$ کم باشد مدل نزدیک به مراکز خوب عمل می‌کند و اگر بزرگ شود بر خلاف آن عمل می‌کند، همچنین در نهایت این مورد هم می‌توان افزود با توجه به مدل انتخابی اگر داده‌های تست ما از آموزش فاصله بگیرند مدل عملکرد خوبی نخواهد داشت.

۴.۳.۶

مقایسه هزینه محاسباتی RBFNN و MLP

۱) عدم آموزش وزن‌های ورودی در RBFNN

در شبکه ،RBF پارامترهای زیر ثابت انتخاب می‌شوند:

$\mu_k$ (مراکز), $\sigma_k$ (عرض‌ها)

بنابراین تنها پارامترهای قابل آموزش، وزن‌های خروجی $\alpha$ هستند.
فرایند آموزش به حل یک دستگاه خطی خلاصه می‌شود:

$$\Phi\alpha = Y$$

این روش هیچ نیازی به گرادیان‌گیری، مشتق‌گیری یا تکرارهای به‌روزرسانی ندارد.
در مقابل، در شبکه MLP باید:

- وزن‌های لایه ورودی،
- وزن‌های لایه‌های مخفی،
- وزن‌های لایه خروجی

با الگوریتم Backpropagation و طی چندین epoch آموزش داده شوند که کاری بسیار پرهزینه است.
۲) آموزش RBF با LLS یک‌مرحله‌ای و مستقیم (Closed-Form)
حل مسئله LLS معمولاً با روش‌های زیر انجام می‌شود:

- معکوس شبه‌وار (Pseudo-inverse)
- تجزیه QR
- تجزیه SVD

این روش‌ها یک مرحله‌ای هستند و پیچیدگی محاسباتی مشخصی دارند.
اما :Backpropagation

- کاملاً تکراری (Iterative)
- نیازمند چندین epoch
- شامل محاسبه فایل‌به‌فایل گرادیان‌ها

بنابراین:

$\text{RBF} \Rightarrow \text{Closed-form}$, $\text{MLP} \Rightarrow \text{Optimization Nonlinear Iterative}$،

۳) LLS یک مسئله محدب است؛ MLP یک مسئله غیرمحدب
در :RBF

- مدل خطی است.
- تابع هزینه محدب است.
- جواب بهینه یکتا است.

اما درMLP:

- تابع خطا غیرمحدب است.
- مینیمم‌های محلی بسیاری دارد.
- نیازمند طی فضای وسیع وزن‌ها با گرادیان‌کاهشی است.

۴) عدم وجود Backprop در RBFNN
ساختار RBF دو لایه دارد:

۱. لایه RBF (فاقد وزن قابل آموزش)

۲. لایه خروجی خطی (تنها وزن‌های قابل یادگیری)

اما MLP شامل:

- چندین لایه مخفی،
- میلیون‌ها وزن،
- استفاده مداوم از قانون زنجیره‌ای،

می‌باشد.
برای ،MLP در هرepoch:

- یک pass Forward
- یک pass Backward
- به‌روزرسانی وزن‌ها

باید انجام شود و این روند چند صد یا چند هزار بار تکرار می‌شود.
۵) پیچیدگی زمانی (ComputationalComplexity)
برایRBFNNباLLS:

$$\text{ساخت}\Phi : O(NK), \qquad \text{حل LLS} : O(K^3) \text{ یا } O(NK^2)$$

این محاسبه فقط یک بار انجام می‌شود.
برایMLP:

$$\text{هزینه هر epoch} : O(NW)$$

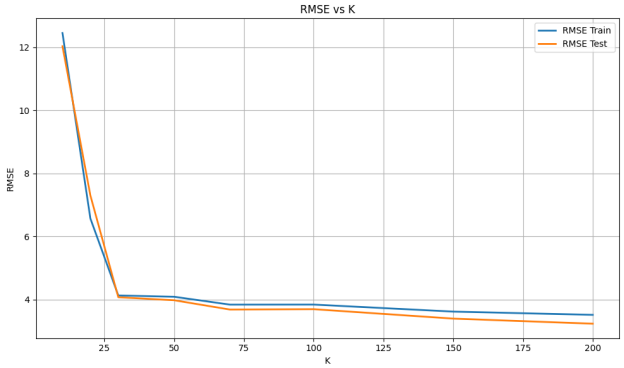
$$\text{کل هزینه} : O(ENW)$$

که در آن:

$$E \in [100, 5000]$$

بنابراین هزینه محاسباتی به‌مراتب بالاتر از RBFNN است.

۵.۳.۶




شکل ۳: K vs RMSE

همانطور که در شکل ۳ مشاهده می‌کنیم با افزایش K دقت ما افزایش و خطای ما کاهش می‌یابد ولی برای جلوگیری از اورفیتینگ ما K بهینه را ۵۰ در نظر خواهیم گرفت که تقریبا خطای ایده‌آلی را به ما خواهد داد.

۶.۳.۶

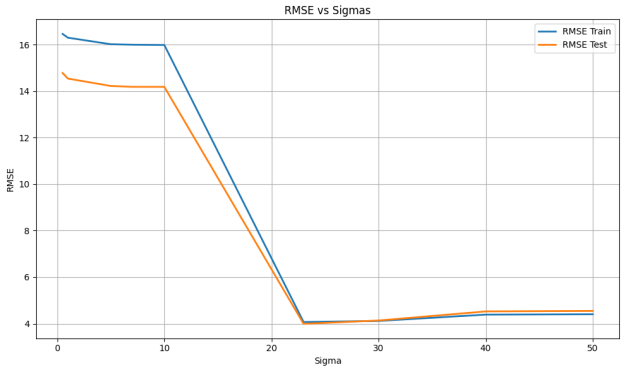
اگر $K$ خیلی کوچک باشد، تعداد نورون‌های مخفی برای تقریب تابع کافی نیست. در این حالت شبکه نمی‌تواند جزئیات دینامیک سیستم را یاد بگیرد و خطای پیش‌بینی بالا می‌رود. این پدیده underfitting یا کم‌برازش نامیده می‌شود؛ یعنی مدل بیش از حد ساده است و توانایی یادگیری کامل داده‌ها را ندارد.

اگر $K$ خیلی بزرگ باشد، شبکه تعداد زیادی نورون دارد و می‌تواند حتی نویز موجود در داده‌ها را نیز یاد بگیرد. در این حالت مدل روی داده‌های آموزش بسیار دقیق عمل می‌کند، اما روی داده‌های تست عملکرد ضعیفی دارد. به این پدیده overfitting یا بیش‌برازش گفته می‌شود؛ زیرا شبکه بیش‌ازحد پیچیده شده و توانایی تعمیم‌دهی آن کاهش می‌یابد.

بنابراین انتخاب مقدار مناسب برای $K$ بسیار اهمیت دارد؛ نه آن‌قدر کوچک باشد که مدل ساده شود و نه آن‌قدر بزرگ که مدل بیش‌ازحد پیچیده گردد.

۷.۳.۶

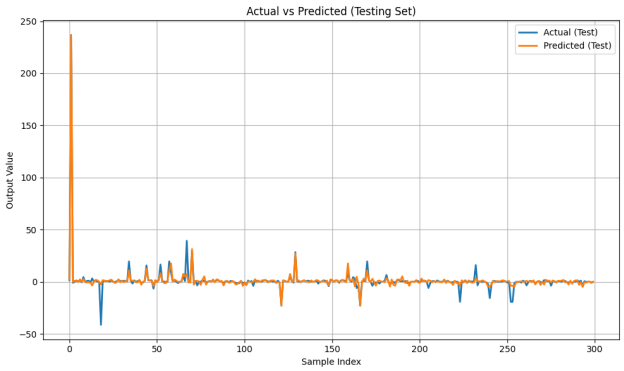
در این بخش مقادیر مختلف را برای $\sigma$ در نظر خواهیم گرفت و نتایج را بررسی خواهیم کرد.




شکل ۴: $\sigma$ vs RMSE

همانطور که از شکل ۴ می‌توان برداشت کرد بهترین $\sigma$ انتخابی مقداری حدود ۲۳ همان مقدار انخابی اولیه ما را دارد.
بهترین پارامترهای انتخابی ما به شرح زیر خواهد بود و با بهترین پارامترها خطا RMSE را مجددا بدست خواهیم آورد.

$$k = 50, \sigma = 23$$




شکل ۵: (test) Predicted vs Actual

$$\text{RMSE}_{\text{train}} = 4.069$$

$$\text{RMSE}_{\text{test}} = 3.992$$

همانطور که از نتایج می‌توان فهمید توانستیم حطا آموزش را به ۴ و خطای تست را به زیر ۴ برسانیم ، که تفاوت آنچنانی با مقدار اولیه خطا ما ندارد، آن هم به همین علت است که در ابتدا ما پارامترهای انتخابی خود را مناسب انتخاب کرده بودیم ولی باز الان توانستیم شرایط را بهبود ببخشیم.

۸.۳.۶

$\sigma$ خیلی کوچک: نورون‌های مخفی تنها در محدوده‌ی بسیار نزدیک به مرکز خود فعال می‌شوند. در این حالت، شبکه قادر به یادگیری جزئیات دقیق داده‌های آموزشی است، اما خارج از این محدوده، پیش‌بینی‌ها بسیار ناپایدار یا ضعیف می‌شوند. این موضوع موجب کاهش تعمیم‌پذیری شبکه روی داده‌های جدید شده و شبکه دچار کم‌برازش محلی می‌شود.

$\sigma$ خیلی بزرگ: نورون‌ها به طور گسترده فعال می‌شوند و تفاوت‌های بین ورودی‌های مختلف را کمتر نمایان می‌کنند. در نتیجه، شبکه ممکن است نتواند تغییرات جزئی دینامیک سیستم را مدل کند و خروجی‌ها بیش‌ازحد همگن شوند. این حالت نیز توانایی شبکه در تعمیم دقیق به داده‌های جدید را کاهش می‌دهد، هرچند ممکن است کم‌برازش کلی رخ دهد.

۹.۳.۶

تأثیر متقابل $K$ و $\sigma$: تعداد نورون‌ها و عرض هر نورون با یکدیگر مرتبط‌اند. برای مثال:

$$\text{کم‌برازش شدید} \Rightarrow \text{کوچک } \sigma \ \& \ \text{کم } K$$

$$\text{بیش‌برازش} \Rightarrow \text{کوچک } \sigma \ \& \ \text{زیاد } K$$

$$\text{از دست رفتن توانایی یادگیری جزئیات} \Rightarrow \text{بزرگ } \sigma \ \& \ \text{زیاد } K$$

تست و ارزیابی گسترده: بهترین مقادیر $K$ و $\sigma$ باید با آزمایش روی مجموعه‌ی تست انتخاب شوند. در بسیاری از موارد، مقدار RMSE تغییرات غیرخطی دارد و انتخاب بهترین ترکیب نیازمند آزمون و خطای متعدد است.

نویز و پیچیدگی داده‌ها: اگر داده‌ها دارای نویز باشند، انتخاب $K$ زیاد یا $\sigma$ کوچک باعث بیش‌برازش نویز و کاهش تعمیم‌پذیری شبکه می‌شود.

زمان محاسباتی و منابع: افزایش $K$ موجب بزرگ‌تر شدن ماتریس $\Phi$ و در نتیجه افزایش زمان محاسباتی الگوریتم LLS و نیاز به حافظه بیشتر می‌شود.

## ۴.۶ بخش چهارم: تلاش برای حل چالش ها

۱.۴.۶

برای این بخش، ما باید RBFNN ایستا را با استفاده از LLS همراه با رگولاریزیشن $L_2$ آموزش دهیم. این روش مخصوصاً وقتی تعداد نورون‌ها $K$ بزرگ است، کمک می‌کند از بیش‌برازش جلوگیری شود و شبکه تعمیم‌پذیرتری داشته باشد.

فرمول رگولاریزیشن $L_2$

در LLS معمولی وزن‌ها به صورت زیر محاسبه می‌شوند:

$$\alpha = (\Phi^T \Phi)^{-1} \Phi^T Y$$

با رگولاریزیشن $L_2$، فرمول به شکل زیر تغییر می‌کند:

$$\alpha = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T Y$$

که در آن:

- $\lambda$ : ضریب رگولاریزیشن (مثلاً $10, 1, 0.1, 0.01, 0.001$)
- $I$ : ماتریس واحد $K \times K$

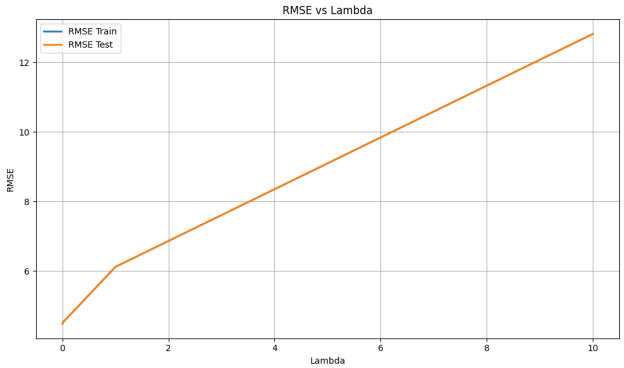
این روش باعث می‌شود که مقادیر بزرگ $\alpha$ محدود شوند و مدل کمتر به نویز حساس شود.
حال برای تحلیل نتایج ما به ازای :

$$k = 120, \sigma = 23$$

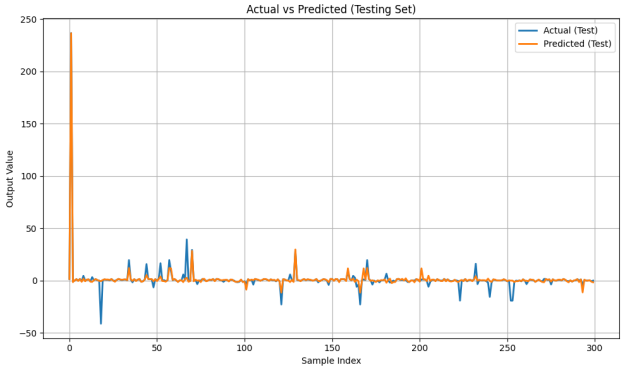
داده‌هارا آموزش می‌دهیم و همچنین تاثیر $\lambda$های مختلف را در کنار آن بررسی خواهیم کرد.
در این بخش $\lambda$ را مقدار گذاری خواهیم کرد و به ازای :

$$\lambda = [0.001, 0.01, 0.1, 1, 10]$$

آموزش می‌دهبم که در شکل ۶ تغییرات خطا را نسبت به آن مشاهده می‌کنیم و نتایج به همان صورت است که انتظار داشتیم با تغییرات $\lambda$ اتفاق بیافتد.




شکل ۶: RMSE vs $\lambda$



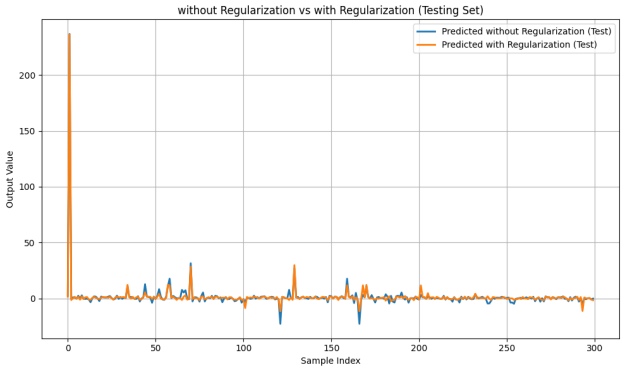

شکل ۷: Predicted vs Actual (test)

شکل ۷ به خوبی به ما مقدار خطای موجود را نشان می‌دهد همچنین مقدار خطای دقیق بصورت زیر می‌باشد:

$$\text{RMSE}_{\text{train}} = 4.48636$$

$$\text{RMSE}_{\text{test}} = 4.50836$$

همانطور که از نتایج می‌توان فهمید خطای مورد نظر با افزایش مراکز تغییر زیادی نکرده‌است ان به همین دلیل است مدل ما به وسیله $\lambda$ ، $a$ را محدود کرده و اجازه زیاد بزرگ شدن را به آن نمی‌دهد در نتیجه می‌توانیم با افزایش مرکز و کنترل شده به دقت مناسبی برسیم.




شکل ۸: Regularization with vs Regularization without (test)

در شکل ۱۰ به خوبی می‌توان مقدار خطای موجود بین ۲ پیش‌بینی را مشاهده کرد که مقدار آن خیلی کم می‌باشد و تقریبا یکسان می‌باشند.

### ۲.۴.۶

پارامتر $\lambda$ در Regularization $L_2$ نقش کنترل شدت محدودیت روی وزن‌های خروجی $\alpha$ را دارد.

- اگر $\lambda$ خیلی کوچک باشد (مثلاً نزدیک صفر): مدل تقریباً همان LLS معمولی را انجام می‌دهد و محدودیتی روی مقادیر $\alpha$ وجود ندارد. در نتیجه اگر $K$ بزرگ باشد، $\alpha$ها می‌توانند بسیار بزرگ شوند و شبکه ممکن است روی داده‌های آموزش بیش‌برازش کند و توانایی تعمیم به داده‌های جدید کاهش یابد.
- اگر $\lambda$ خیلی بزرگ باشد: وزن‌ها $\alpha$ خیلی کوچک می‌شوند و شبکه سعی می‌کند خروجی‌ها را بیش‌ازحد هموار کند. در این حالت مدل قدرت یادگیری جزئیات داده‌ها را از دست می‌دهد و خطای پیش‌بینی حتی روی داده‌های آموزش افزایش می‌یابد (کم‌برازش رخ می‌دهد).

## ۵.۶ بخش پنجم

### ۱.۵.۶

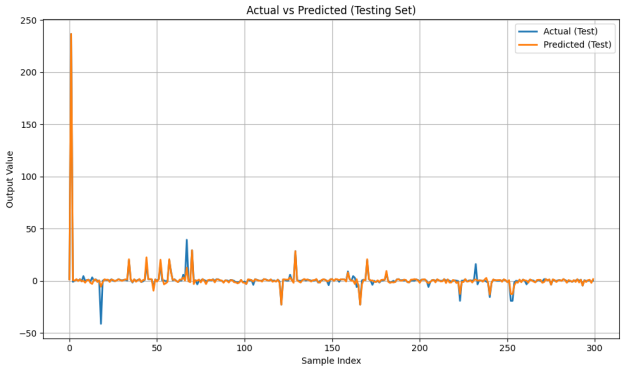
برای این بخش، به جای انتخاب تصادفی $K$ مرکز، از الگوریتم K-Means برای تعیین مراکز $\mu_k$ استفاده می‌کنیم. این کار معمولاً باعث می‌شود مراکز بهتر روی فضای داده قرار بگیرند و شبکه دقت بهتری داشته باشد.

در این بخش بر اساس خواسته سوال وزن‌های بهینه را بدست می‌آوریم در نهایت نتایج خطا ما بصورت زیر می‌باشد:

$$\text{RMSE}_{\text{train}} = 3.4652$$

$$\text{RMSE}_{\text{test}} = 3.1169$$

همانطور که مشاهده می‌کنیم مراکز بهتر بر روی فضا قرار گرفته‌اند و دقت ما خیلی بهتر از قبل شده‌است.




شکل ۹: Predicted vs Actual (test)

با توجه به شکل ۱۰ ، مشاهده می‌کنیم نقاطی که تغییر ناگهانی هم داشته‌ایم مدل توانسته خوب عمل کند و تغییرات ناگهانی هم پیش‌بینی کند. ولی وقتی شکل ۲ مشاهده می‌کنیم، متوجه می‌شویم چه مدل ما چقدر بهتر عمل کرده‌است و بهتر توانسته تخمین بزند حتی برای تغییرات ناگهانی، پس می‌توانیم به این نتیجه برسیم که الگوریتم k-Means دقت مارا بشدت بهبود داده‌است.

### ۲.۵.۶

در ادامه، یک راه‌حل کامل، دقیق و کاملاً عملیاتی برای محاسبهٔ عرض‌های متغیر $\sigma$ با استفاده از استراتژی $2 - Nearest - Neighbor$ ارائه می‌کنیم.

مراکز $\mu_k$ همان مراکز به‌دست‌آمده از K-Means سؤال قبل هستند.

برای هر مرکز، مقدار $\sigma_i$ برابر است با میانگینِ کوچک‌ترین دو فاصله تا سایر مراکز.

محاسبهٔ $\sigma_i$ برای هر مرکز با روش $2 - Nearest - Neighbor$:

۱. برای هر مرکز $\mu_i$، فاصلهٔ اقلیدسی آن تا تمام مراکز دیگر محاسبه می‌شود.

۲. این فاصله‌ها به‌صورت صعودی مرتب می‌شوند.

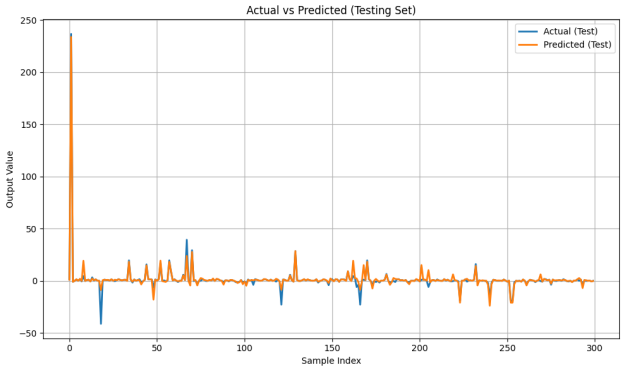
۳. کوچک‌ترین دو مقدار انتخاب می‌شود.

۴. میانگین این دو مقدار به‌عنوان $\sigma_i$ در نظر گرفته می‌شود.

به این ترتیب، هر نورون RBF یک عرض مخصوص به خودش خواهد داشت.

$$\text{RMSE}_{\text{train}} = 3.4284$$

$$\text{RMSE}_{\text{test}} = 3.3402$$




شکل ۱۰: Predicted vs Actual (test)

نتایج این بخش تقریبا مشابه نتایج بخش قبل می‌باشد، ولی انتظار ما اینگونه بود که دقت بهتر شود، دقت بدست آمده مناسب است اما علت تغییر نکردن آن به دلیل ساختار دیتا ما می‌باشد که می‌توان برداشت کرد که توابع گوسی آن‌ها مشابه می‌باشد به همین دلیل با استفاده از الگوریتم بالا تقریبا دقت ما ثابت مانده‌است.

مزیت اصلی روش **2-NN**

این روش به مدل اجازه می‌دهد که:

- در نواحی پیچیدهٔ داده، نورون‌ها باریک و حساس باشند.
- در نواحی یکنواخت، نورون‌ها پهن و دارای تعمیم‌پذیری بیشتر شوند.

بنابراین، مدل سازگارتر و دقیق‌تر خواهد شد.

### ۳.۵.۶

چرا K-Means از نظر تئوری برای انتخاب مراکز RBF بهتر است؟

انتخاب مراکز RBF (یعنی $\mu_k$) تعیین‌کنندهٔ ساختار مدل RBFNN است؛ این مراکز پایهٔ نگاشت غیرخطی $\Phi$ را تشکیل می‌دهند. دو رویکرد اصلی وجود دارد:

- انتخاب تصادفی مراکز،
- انتخاب مراکز با K-Means.

از نظر تئوری، K-Means مزایای بنیادینی دارد:

۱. قرار گرفتن مراکز در نواحی پرتراکم داده (Data-Driven)

در K-Means مراکز دقیقاً در نواحی با بیشترین تراکم داده قرار می‌گیرند. این الگوریتم عملاً مسئلهٔ زیر را کمینه می‌کند:

$$\min_{\mu_k} \sum_i \min_k \|x_i - \mu_k\|^2$$

در مقابل، انتخاب تصادفی کورکورانه است و ممکن است مراکز در نواحی دورافتاده یا تکراری قرار بگیرند یا نواحی مهم بدون مرکز بمانند. نتیجه این است که مدل نامتعادل و ضعیف می‌شود.

۲. هر مرکز نمایندهٔ یک خوشه است

K-Means فضای ورودی را به $K$ ناحیه تقسیم می‌کند و هر $\mu_k$ نمایندهٔ یک خوشه است. در تقریب تابع، هر RBF یک ناحیهٔ معنی‌دار را پوشش می‌دهد. انتخاب تصادفی فاقد این خاصیت است و پوشش فضای داده نامتوازن می‌شود.

۳. کاهش تداخل بین **RBF**ها

اگر مراکز بیش‌ازحد نزدیک باشند، همپوشانی شدید رخ می‌دهد؛ اگر خیلی دور باشند، بخشی از فضا بدون پوشش می‌ماند. K-Means مراکز را بهینه پخش می‌کند و ماتریس $\Phi$ حالت بهتری (well-conditioned) پیدا می‌کند. انتخاب تصادفی چنین تضمینی ندارد.

۴. افزایش تعداد نورون‌های مؤثر

نورون زمانی مؤثر است که در ناحیهٔ داده فعال شود. با انتخاب تصادفی، برخی مراکز در نواحی بدون داده می‌افتند و عملاً بی‌اثر می‌شوند. K-Means مراکز را دقیقاً روی داده‌ها قرار می‌دهد و ظرفیت یادگیری مدل را افزایش می‌دهد.

۵. کاهش خطای کوانتیزاسیون و بهبود تقریب تابع

خطای RBFNN وابسته به فاصلهٔ داده‌ها تا مراکز است. K-Means این فاصله را کمینه می‌کند، بنابراین:

- خطای بازسازی کاهش می‌یابد،
- دقت نهایی مدل افزایش می‌یابد،
- RMSE کاهش می‌یابد.

۶. تولید $\sigma$های مناسب‌تر حتی قبل از NN-۲

با مراکز خوشه‌بندی‌شده، فاصله‌ها منظم‌تر می‌شوند و $\sigma$ها خوش‌تعریف‌ترند. در مقابل، مراکز تصادفی فاصله‌هایی بی‌قاعده ایجاد می‌کنند که به $\sigma$های نامناسب و کیفیت ضعیف RBF منجر می‌شود.

۷. **K-Means** یک تقریب نزدیک به مرکزگذاری بهینه است

در تئوری، بهترین مراکز RBF نقاطی هستند که خطای تقریب را کمینه کنند. K-Means یک تقریب قوی از این مرکزگذاری بهینه است و در منابع معتبر به‌عنوان روش استاندارد علمی معرفی می‌شود.

# قسمت دوم: پیادهسازی RBFNN تطبیقی با الهام از M−RAN

## ۶.۶ بخش اول: یادگیری ترتیبی و معیار رشد

### ۱.۶.۶

برای این بخش می‌توانیم یک تابع پایتون بنویسیم که بررسی کند آیا برای یک نمونه جدید باید نورون مخفی جدید اضافه شود یا خیر بر اساس دو معیار: تازگی (distance) و خطا .(error) و ما مقادیر زیر را تست کردیم و نتیجه اضافه کردن نورون را دریافت کردیم:

```
centers = np.array([[1, 1], [3, 3]])
#۲
#۳
x_new = np.array([0.5, 0.5])
```

```
y_true = 1.0
y_pred = 0.4


#
add_neuron = new_neuron(x_new, y_true, y_pred, centers, epsilon=1.0, e_min=0.5)
tnirp("deeN wen snoruen?", add_neuron)
```

۲.۶.۶

اهمیت دو معیار

معیار خطا :(error) این معیار بررسی می‌کند که شبکه فعلی تا چه حد در پیش‌بینی نمونه جدید موفق بوده است. اگر خطا بزرگ باشد، نشان می‌دهد که شبکه توانایی مدل‌سازی این نمونه را ندارد.

معیار تازگی :(distance/freshness) این معیار سنجش می‌کند که نمونه جدید تا چه اندازه از مراکز نورون‌های موجود دور است. حتی اگر خطا کوچک باشد اما نمونه بسیار دور از مراکز فعلی قرار داشته باشد، شبکه نیاز به اضافه‌کردن نورون جدید دارد تا بخش جدیدی از فضای ورودی پوشش داده شود.

بنابراین هر دو معیار مکمل یکدیگر هستند:

- خطا برای ارزیابی کیفیت یادگیری،
- تازگی برای پوشش کامل فضای ورودی.

اگر فقط از معیار خطا استفاده کنیم:

در این حالت نورون جدید تنها در زمانی اضافه می‌شود که خطا بزرگ باشد. این کار یک مشکل اساسی دارد: اگر نمونه جدید در ناحیه‌ای از فضای ورودی قرار بگیرد که نورون‌های فعلی هنوز آن را پوشش نمی‌دهند، اما خطا کوچک باشد، شبکه آن ناحیه را پوشش نخواهد داد.

نتیجه: شبکه ممکن است در آینده نتواند روی نمونه‌های مشابه عملکرد مناسبی داشته باشد.

اگر فقط از معیار تازگی استفاده کنیم:

در این حالت نورون جدید تنها زمانی اضافه می‌شود که نمونه بسیار دور از مراکز فعلی باشد. اگر نمونه در محدوده پوشش فعلی باشد اما خطا بزرگ باشد، نورون جدید تولید نمی‌شود و شبکه اشتباه خود را اصلاح نخواهد کرد.

نتیجه: شبکه ممکن است در همان منطقه دچار خطای ثابت شود و دقت کلی کاهش یابد.

تأثیر تغییر $\varepsilon$ (آستانه تازگی):

- $\varepsilon$ کوچک: شبکه نسبت به فاصله بسیار حساس می‌شود؛ نورون‌های زیادی اضافه می‌شوند حتی برای نمونه‌های نزدیک به مراکز موجود.

  نتیجه: شبکه بزرگ و پیچیده می‌شود و احتمال بیش‌برازش افزایش می‌یابد.

- $\varepsilon$ بزرگ: تنها نمونه‌هایی که بسیار دور هستند نورون جدید ایجاد می‌کنند.

  نتیجه: شبکه کوچک‌تر و ساده‌تر می‌شود، اما ممکن است برخی نواحی فضای ورودی پوشش داده نشوند و در نتیجه کم‌برازش اتفاق بیفتد.

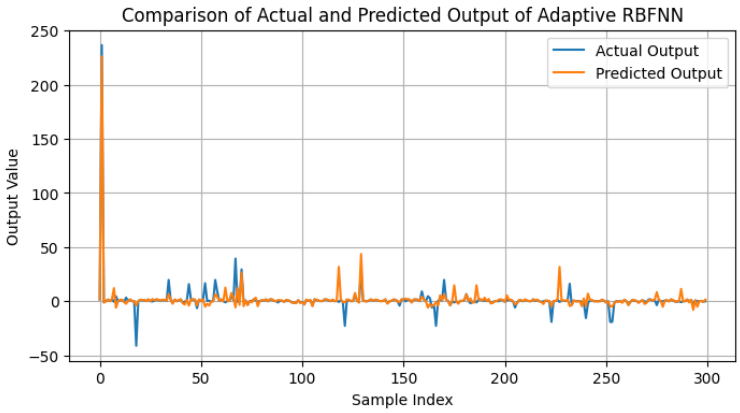
## ۷.۶ بخش دوم: پیاده سازی RBFNN تطبیقی با استراتژی هرس

### ۱.۷.۶

اکنون، منطق رشد (از بخش اول) و منطق هرس پیشرفته (از بخش قبل) را در یک حلقهٔ آموزش واحد ادغام کرده و به‌صورت کامل پیاده‌سازی می‌کنیم و مدل را آموزش می‌دهیم.

### ۲.۷.۶

$$\text{RMSE}_{\text{test}} = 6.026971974137698$$




شکل ۱۱: (test) Predicted vs Actual

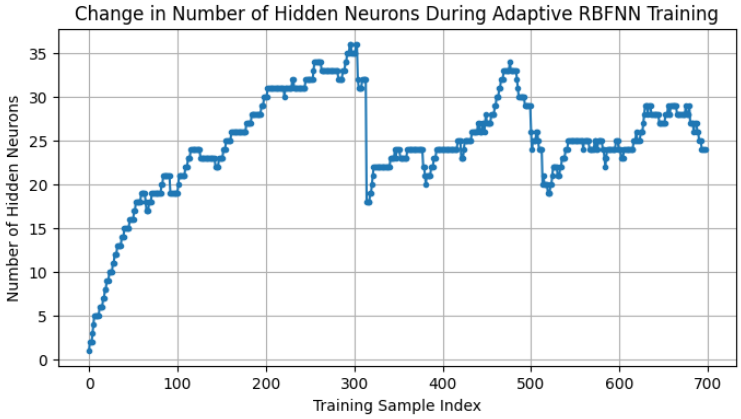
$$BestHyperparametersUsed:$$

$$epsilon = 1.0, emin = 0.1, kappa = 0.8, eta = 0.05, delta = 0.1, M = 60$$

$$FinalNumberofHiddenNeurons: 181$$

همانطور از شکل ۱۱ می‌توان برداشت کرد این است، که نمودار خروجی ما خیلی بهتر شده‌است و یک تفاوت اساسی که با خروجی‌های مدل معمولی دارد، این است که توانسته نوسان‌ها را به خوبی در مقادیر کم و زیاد دنبال کند، یم فرق آن با قسمت‌های قبل هم این است که مدل در بعضی نقاط خطای شدیدی داشته آن هم به دلیل حساس بودن آن به هایپرپارامترهای موجود است که اگر به خوبی برای خطا تنظیم نشده‌باشند می‌تواند خطا را خیلی بیشتر کند ، بخش‌هایی که خطا بزرگتر است ممکن است مربوط به نقاطی باشد که دینامیک غیرخطی شدیدتر است یا تعداد نورون‌ها کم بوده است.

۳.۷.۶




شکل ۱۲: Training RBFNN Adaptive During Neurons Hidden of Number in Change

با توجه به شکل ۱۲ می‌توان برداشت کرد که هرچقدر تعداد سمپل‌های ما بالا می‌رود تعداد نورون‌های ما هم بیشتر می‌شود تا به مقدار ۳۰۰ تقریبا می‌رسیم که تعدادی نورون حذف می‌شود در ادامه دوباره تعداد نورون‌ها افزایش پیدا می‌کند.

۴.۷.۶

$$M - RAN$$

$$FinalNumberofHiddenNeurons : 34$$

$$\text{RMSE}_{\text{test}} = 6.17891$$

$$StaticRBFNN$$

$$FinalNumberofHiddenNeurons : 30$$

$$\text{RMSE}_{\text{test}} = 3.3402$$

## تحلیل و مقایسه

### دقت مدل

شبکهٔ Static RBFNN نسبت به M-RAN عملکرد بهتری داشته است:

$$3.34 \ll 6.17$$

این نتیجه نشان می‌دهد که رویکرد ایستا با تعداد نورون کمتر توانسته است خطای کمتری کسب کند.
دلیل احتمالی: در مسئلهٔ موردنظر، توزیع داده‌ها نسبتاً منظم بوده و نیازی به رشد تطبیقی شبکه وجود نداشته است. بنابراین M-RAN در این داده‌ها نتوانسته ساختار مؤثرتری نسبت به روش ایستا پیدا کند.

## پیچیدگی مدل

M-RAN از 34 نورون استفاده کرده، در حالی‌که شبکهٔ Static RBF تنها 30 نورون دارد.
هدف M-RAN یافتن توپولوژی حداقلی با حفظ دقت بالا است، اما در اینجا:

- نه تنها نورون‌های بیشتری تولید کرده،
- بلکه دقت پایین‌تری نیز به دست آورده است.

جمع‌بندی: در این تست، M-RAN به هدف «حداقل‌سازی منابع» نرسیده است.

## رفتار تطبیقی M-RAN

مزیت اصلی M-RAN در شرایطی است که:

- داده‌ها نویزی، ناپایدار یا با توزیع نامشخص باشند؛
- ساختار مناسب شبکه از قبل معلوم نباشد؛
- مدل نیاز به یادگیری آنلاین و سازگاری با تغییرات داده داشته باشد.

اما وقتی داده‌ها ایستا و نسبتاً تمیز هستند، مدل ایستا معمولاً عملکرد بهتری دارد.

## آیا M-RAN موفق به یافتن توپولوژی حداقلی شد؟

با توجه به نتایج:

$$\text{تعداد نورون‌های RBF ایستا} = 30 \qquad \text{تعداد نورون‌های} M-RAN = 34,$$

و همچنین:

$$\text{RMSE}_{\text{M-RAN}} \approx 6.17 > \text{RMSE}_{\text{Static}} \approx 3.34$$

پس خیر، M-RAN در این آزمایش نتوانسته توپولوژی حداقلی همراه با دقت بالا پیدا کند.

## اهمیت این موضوع در کاربردهای دنیای واقعی

### ۱. هزینهٔ محاسباتی

هر نورون اضافی به معنی:

- محاسبات بیشتر،
- مصرف حافظهٔ بیشتر،
- تأخیر بیشتر در سیستم‌های بلادرنگ.

این موضوع در کنترل، پردازش سیگنال، پیش‌بینی آب‌وهوا و سیستم‌های تعبیه‌شده حیاتی است.

### ۲. خطر بیش‌برازش

مدل‌هایی که بدون دلیل بزرگ می‌شوند، در داده‌های واقعی (پرنویز) دچار کاهش عملکرد می‌شوند.

### ۳. قابلیت پیاده‌سازی سخت‌افزاری

در رباتیک، کنترل صنعتی و الکترونیک تعبیه‌شده، یک مدل کوچک‌تر و سریع‌تر ضروری است.

### ۴. پایداری یادگیری

الگوریتم‌هایی مانند M-RAN اگر به‌خوبی تنظیم نشوند، ممکن است:

- بیش‌ازحد رشد کنند،
- ناپایدار شوند،
- یا رفتار پیش‌بینی‌نشده از خود نشان دهند.

بنابراین داشتن توپولوژی کوچک و دقیق، در کاربردهای عملی اهمیت بالایی دارد.

۵.۷.۶

## مقایسه رویکرد تطبیقی M-RAN و RBFNN ایستا

با توجه به نتایج به‌دست‌آمده، رویکرد تطبیقی M-RAN و مدل RBF ایستای کلاسیک رفتار کاملاً متفاوتی از خود نشان داده‌اند. تحلیل این تفاوت‌ها دید روشنی از مزایا و محدودیت‌های شبکه‌های تطبیقی بر اساس معیارهای رشد و هرس ارائه می‌دهد.

## مزایای رویکرد تطبیقی M-RAN

- توانایی سازگاری با داده‌های پیچیده یا غیرایستا: M-RAN این قابلیت را دارد که در صورت مواجهه با الگوهای جدید، فضایی غیرپوشش‌دهنده، یا خطاهای بزرگ، تعداد نورون‌های پنهان را افزایش دهد. این ویژگی برای مسائلی که توزیع داده‌ها در طول زمان تغییر می‌کند یا ماهیت آنلاین دارند بسیار ارزشمند است.
- عدم نیاز به تعیین تعداد نورون‌های پنهان از ابتدا: در RBFNN ایستا لازم است کاربر از ابتدا توپولوژی شبکه را با آزمون‌وخطا یا جستجوی شبکه‌ای تعیین کند. در مقابل، M-RAN ساختار را به‌صورت خودکار، بر اساس معیارهای یادگیری، تنظیم می‌کند.
- توانایی ارائه یک مدل داده‌محور و :adaptive با ورود هر نمونه جدید، شبکه ساختار و وزن‌ها را مجدداً ارزیابی می‌کند. این امر امکان استفاده از M-RAN را در محیط‌های بلادرنگ فراهم می‌سازد.

## معایب رویکرد تطبیقی M-RAN

- رشد بیش از حد شبکه و افزایش تعداد نورون‌ها: در نتایج شما تعداد نورون‌های پنهان به 181 رسیده است، در حالی که RBFNN ایستا تنها 30 نورون دارد. این رشد شدید نشان‌دهندهٔ حساسیت زیاد معیار تازگی و خطا است و به بروز وابستگی به نویز (over-fitting) منجر شده است.
- عملکرد ضعیف‌تر نسبت به RBFNN ایستا: RMSE روی داده‌های تست برای M-RAN 6.02 و در مقابل RBFNN ایستا 3.34 است. این نتیجه نشان می‌دهد که شبکه تطبیقی نه تنها یادگیری بهتری نداشته، بلکه با افزایش بی‌رویه ظرفیت، تعمیم آن نیز تضعیف شده است.
- پیچیدگی محاسباتی بیشتر: هرچه تعداد نورون‌ها افزایش یابد، محاسبات توابع RBF و به‌روزرسانی وزن‌ها هزینه‌برتر می‌شود. این امر سرعت شبکه را کاهش می‌دهد.
- حساسیت بالا به هایپرپارامترها: معیارهای رشد، هرس، و نرخ یادگیری اثر مستقیم و شدید بر رفتار شبکه دارند. انتخاب نامناسب این پارامترها به رشد کنترل‌نشده یا عملکرد ضعیف منجر می‌شود.

## آیا شبکه تطبیقی به توپولوژی حداقلی و دقیق رسیده است؟

خیر. هدف M-RAN این است که:

- تنها در صورت لزوم نورون اضافه کند،
- یک شبکه کم‌حجم (minimaltopology) بسازد،
- و در عین حال دقت بالا حفظ شود.

اما در نتایج شما:

- توپولوژی بیش‌ازحد بزرگ شده (181 نورون)
- دقت کاهش پیدا کرده (RMSE بیش از دو برابر مدل ایستا)
- شبکه موفق نشده است مرکزهای مؤثر و توزیع شعاع‌ها را بهینه پیدا کند.

### چرا یافتن توپولوژی حداقلی برای کاربردهای واقعی بسیار مهم است؟

- کاهش هزینه پردازشی و امکان پیاده‌سازی روی سخت‌افزار واقعی: سیستم‌های صنعتی یا نهفته (embedded) مانند کنترلرها، ربات‌ها یا تجهیزات قدرت منابع پردازشی محدودی دارند. یک شبکه بزرگ، اجراشدنی نیست.
- قابلیت تعمیم بهتر و پایداری عملیاتی: مدل‌های کوچک‌تر معمولاً نویز را کمتر یاد می‌گیرند و در شرایط واقعی رفتار قابل‌پیش‌بینی‌تری دارند.
- سرعت یادگیری و استنتاج در کاربردهای بلادرنگ: توابع RBF هزینه محاسباتی بالایی دارند. اگر تعداد نورون زیاد باشد، سیستم نمی‌تواند در زمان محدود پاسخ دهد.
- مشکل حافظه و ذخیره‌سازی: شبکه‌های تطبیقی بزرگ، فضای زیادی برای ذخیره مراکز، شعاع‌ها، وزن‌ها و شمارنده‌های pruning لازم دارند.
- سادگی مدل و قابلیت تحلیل: در کاربردهای حساس (پزشکی، کنترل، هوافضا) مدل باید قابل‌توضیح و ارزیابی باشد. شبکه‌های بزرگ interpretability خود را از دست می‌دهند.

### جمع‌بندی نهایی

بر اساس نتایج به‌دست‌آمده:

- M-RAN در این آزمایش به ساختار بهینه و حداقلی دست نیافته است.
- شبکه تطبیقی بیش‌ازحد رشد کرده و تعمیم ضعیف‌تر از مدل ایستا ارائه داده است.
- مدل ایستا با تعداد نورون کم و ساختاری فشرده عملکرد بهتر و پایدارتر داشته است.

بنابراین، در این مسئله خاص، رویکرد ایستا مناسب‌تر عمل کرده است، مگر اینکه:

- معیارهای رشد/هرس به طور دقیق تنظیم شوند،
- یا داده‌ها ماهیت غیرایستا داشته باشند.

# ۷ کدنویسی ـ SVM خطی و مطالعهٔ پارامتر C

## ۱.۷ آماده‌سازی

در این بخش ما دیتاست مربوط به سرطان را لود می‌کنیم، که دارای ۳۱ ویژگی است، و هدف ما تشخیص بدخیم یا خوش‌خیم بودن سرطان است.

دیتاست را بررسی اولیه می‌کنیم و دیتاست را پاکسازی می‌کنیم، سپس داده‌های آموزش و تست و صحت را جداسازی می‌کنیم. در مرحله بعد با توجه به خواسته سوال ویژگی‌هارا استاندارد سازی می‌کنیم که خطا در مدل کاهش پیدا کند.

| | id | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean | compactness_mean | concavity_mean | concave points_mean |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 842302 | 1 | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.3001 | 0.14710 |
| 1 | 842517 | 1 | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.0869 | 0.07017 |
| 2 | 84300903 | 1 | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.1974 | 0.12790 |
| 3 | 84348301 | 1 | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.2414 | 0.10520 |
| 4 | 84358402 | 1 | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.1980 | 0.10430 |

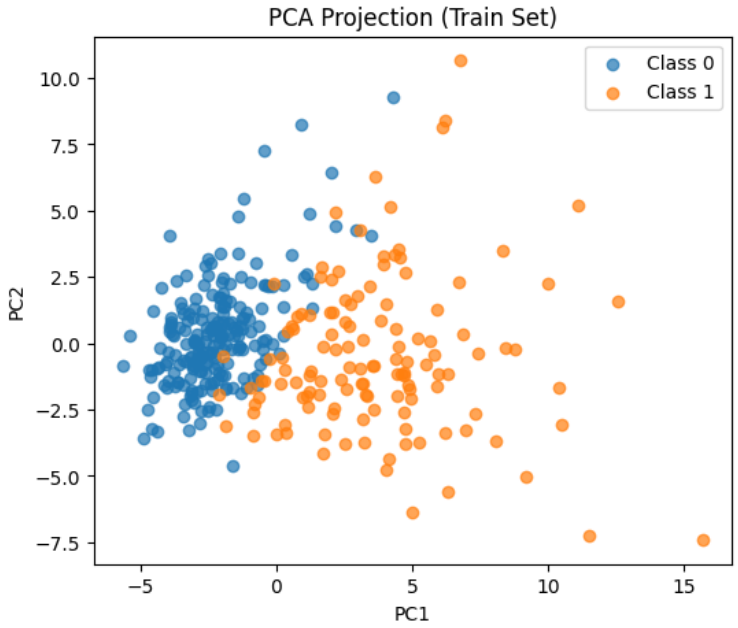
شکل ۱۳: نمایش بخشی از دیتاست

هدف اجرای PCA دوبعدی

PCA در این مرحله فقط برای این استفاده می‌شود که:

- داده‌ها را در یک فضای دوبعدی نمایش دهیم؛
- ببینیم توزیع نقاط (Malignant/Benign) در فضای ویژگی چگونه است؛
- دید شهودی از جداپذیری داده‌ها داشته باشیم؛
- بدون دخالت در فضای ویژگی اصلی که قرار است مدل‌ها روی آن آموزش ببینند.

بنابراین PCA فقط برای Visualization استفاده می‌شود و خروجی آن ورودی مدل‌ها نمی‌شود.




شکل ۱۴: Visualization PCA

همانطور که در شکل ۱۴ مشاهده می‌کنید، داده‌ها را که در ۲ بعد نمایش داده‌ایم بخوبی قابلیت تفکیک داشته‌اند و جداسازی را انجام داده‌است.

$$class0 = Benign$$

$$class1 = Malignant$$

## ۲.۷ آموزش و گزارش حاشیه

در این بخش با استفاده از SVM آموزش را انجام می‌دهیم، با استفاده از بردار وزن‌ها نرم آن را بدست می‌آوریم.

$$w = \Sigma_i a_i y_i x_i$$

با استفاده از فرمول بالا، نرم آن را بدست می‌آوریم که بصورت زیر می‌باشد:

$$\frac{1}{||w||} = 2.9032$$

که مقدار بدست آمده بالا فاصله یک صفحه تا مرز تصمیم می‌باشد.
این مقدار نسبتاً بزرگ محسوب می‌شود و نشان می‌دهد:

- مرز تصمیم نسبت به داده‌ها فاصلهٔ خوبی دارد؛ یعنی SVM توانسته است یک hyperplane پیدا کند که بین دو کلاس فاصلهٔ هندسی قابل قبولی ایجاد می‌کند.
- مدل تمایل به واریانس کم دارد ؛(خوبGeneralization) زیرا margin بزرگ ⇒ مدل کمتر به نوسانات کوچک داده‌ها حساس است.

نتایج ارزیابی مدل

| Validation | Test | Metric |
|---|---|---|
| 0.9737 | 0.9649 | Accuracy |
| 0.9677 | 0.9444 | Score F۱ |
| 0.9574 | 0.9444 | Recall |
| 0.9783 | 0.9444 | Precision |
| 0.9822 | 0.9868 | ROC-AUC |

نتایج ارزیابی مدل

مدل ارائه‌شده عملکرد بسیار خوبی روی داده‌های تست و اعتبارسنجی نشان داد. دقت آن روی داده‌های تست 96.5% و روی اعتبارسنجی 97.4% بود. مقادیر F۱ ،Recall Score و Precision نیز همگی بالای 94% هستند، که نشان‌دهنده تعادل مناسب بین شناسایی نمونه‌های مثبت و کاهش خطاها است. مقدار ROC-AUC نزدیک به 0.98 نیز قدرت تمایز بالای مدل بین کلاس‌ها را تأیید می‌کند.

## ۳.۷ مطالعهٔ C

در بخش به ازای C های مختلف مدل را امتحان می‌کنیم و شاخص‌های بخش قبل را استخراج می‌کنیم و در یک جدول نمایش می‌دهیم.

| ROC-AUC | Precision | Recall | F۱ | Accuracy | Vectors Support | C |
|---|---|---|---|---|---|---|
| 0.986980 | 0.978261 | 0.957447 | 0.967742 | 0.973684 | 91 | 0.01 |
| 0.983169 | 0.978261 | 0.957447 | 0.967742 | 0.973684 | 45 | 0.10 |
| 0.982217 | 0.978261 | 0.957447 | 0.967742 | 0.973684 | 26 | 1.00 |
| 0.974278 | 0.977778 | 0.936170 | 0.956522 | 0.964912 | 20 | 10.00 |
| 0.967291 | 0.977778 | 0.936170 | 0.956522 | 0.964912 | 19 | 100.00 |
| 0.967291 | 0.977778 | 0.936170 | 0.956522 | 0.964912 | 19 | 1000.00 |

جدول ۱: Performance metrics of SVM for different values of $C$.

تحلیل تغییرات Bias/Variance

- C بزرگ (۱۰۰، ۱۰): مدل سعی می‌کند همه نقاط آموزش را درست دسته‌بندی کند. تعداد هاSupportVector کمتر است. خطای آموزش پایین و واریانس بیشتر، bias کمتر. ممکن است overfitting رخ دهد.
- C متوسط (۱): تعادل خوبی بین دقت آموزش و generalization برقرار است. تعداد هاSupportVector کمی افزایش یافته است. خطای validation پایین، bias و variance متعادل است.
- C کوچک (۰.۰۱، ۰.۰): مدل سختی کمتری برای مجازات خطاها دارد. تعداد هاSupportVector زیاد می‌شود.

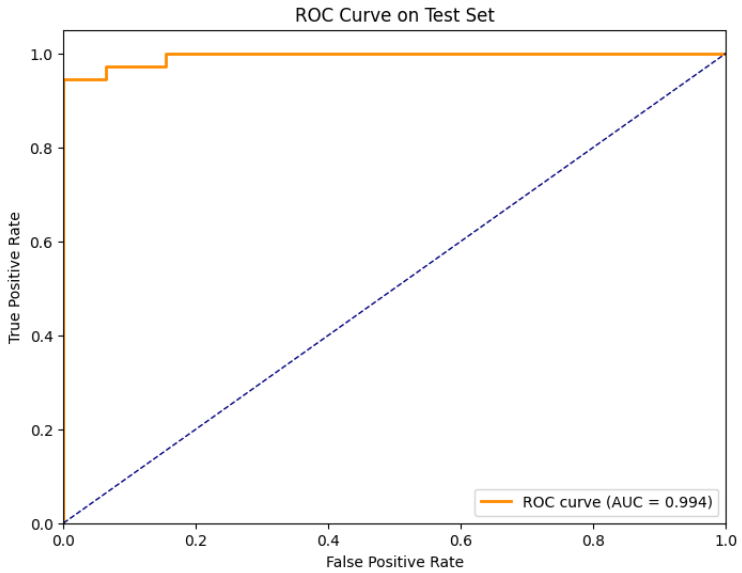
  bias بالا و variance پایین ⇒ underfitting.

  Accuracy، F۱ و ROC-AUC کاهش می‌یابند.

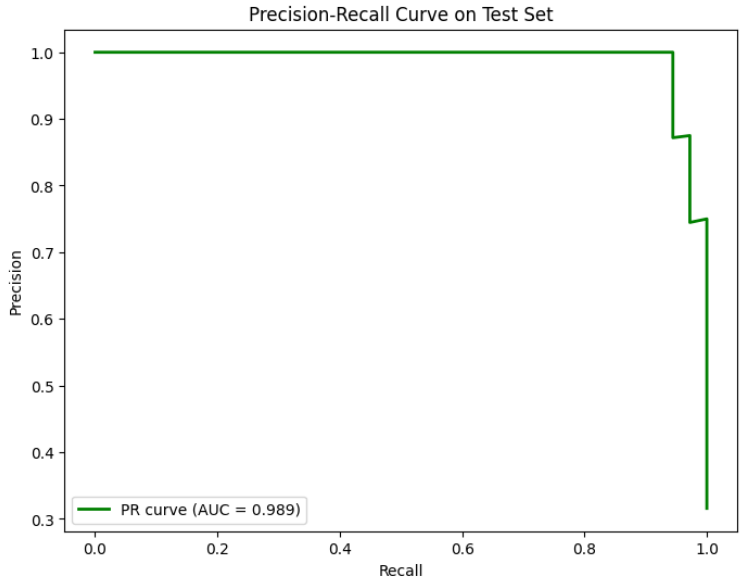
نتیجه کلی: با کاهش $C$، مدل ساده‌تر شده و تعداد هاSV افزایش می‌یابد؛ bias افزایش و variance کاهش می‌یابد. بالعکس، افزایش $C$ باعث کاهش bias و افزایش variance می‌شود و ممکن است overfitting ایجاد کند.

## ۴.۷ نمودارهای ارزیابی

حال با بهترین مقدار $C$ (انتخاب شده بر اساس بهترین عملکرد روی validation) می‌توانیم منحنی ROC و Precision-Recall را رسم کرده و تحلیل کنیم.




شکل ۱۵: Set Test on Curve ROC




شکل ۱۶: Set Test on Curve Precision-Recall

ROC Curve (AUC = 0.994)

ROC نرخ TruePositive در مقابل FalsePositive را نشان می‌دهد. منحنی تقریباً به بالای چپ نزدیک است و از خط قطری فاصله دارد، یعنی مدل تقریباً همه نمونه‌های مثبت را درست شناسایی کرده و خطای مثبت کاذب بسیار کم است.

AUC = 0.994: مقدار بسیار نزدیک به ۱ است → مدل عملکرد فوق‌العاده‌ای در تفکیک کلاس‌ها دارد.

این نشان می‌دهد که مدل sensitivity high و specificity high دارد.

Precision–Recall Curve (AUC = 0.989)

Curve Precision-Recall برای داده‌های نامتوازن حساس‌تر است. منحنی نزدیک به گوشه بالا-چپ است → هم Recall (حساسیت) و هم Precision (دقت) بسیار بالاست.

AUC = 0.989: مقدار بسیار بالا → مدل همواره دقت خوبی در پیش‌بینی کلاس مثبت دارد.

نشان می‌دهد که تعداد Positive False کم و تعداد Positive True بالا است.