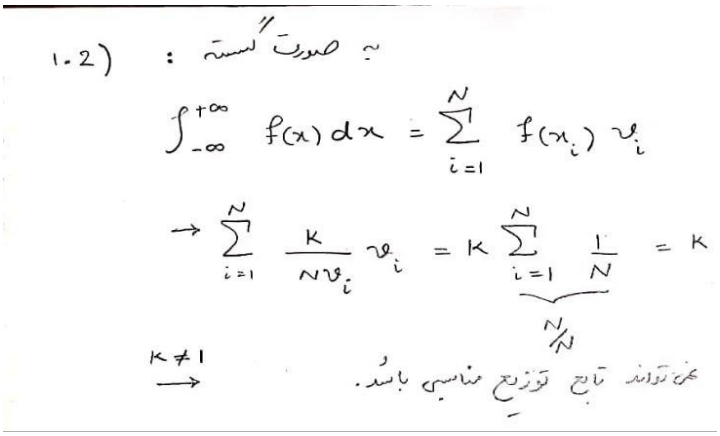
به نام خدا

تمرین دوم یادگیری ماشین

غزل زمانی‌نژاد

۱. الف) در روش پارزن، v متغیر کنترلی است بنابراین با تغییر مقدار h_n می‌توانیم به پنجره با ابعاد متفاوت برسیم. در صورتی که مقدار h_n بسیار کوچک باشد، ۰ یا ۱ داده داخل حجم قرار می‌گیرد. در این حالت به یک تخمین بسیار پیچیده می‌رسیم که نسبت به نویز حساس، robustness آن کم و واریانس زیاد است. در صورتی که مقدار h_n بسیار بزرگ باشد، کم و زیاد شدن مقادیر را در نظر نمی‌گیرد و در حالتی که h_n به سمت بی‌نهایت برود تمامی داده‌ها در پنجره قرار می‌گیرند. در اینجا به تخمین بسیار نرم می‌رسیم که نمی‌تواند تخمین خوبی از داده‌ها باشد و بایاس آن زیاد است.
در روش KNN، k_n متغیر کنترلی است؛ در واقع همسایگی را به قدری بزرگ می‌کنیم که k_n نمونه داخل آن قرار بگیرد. اگر k_n بسیار کوچک باشد، برای تخمین تنها چند همسایه نزدیک را در نظر می‌گیرد. در این حالت تخمین نسبت به نویز حساس، robustness آن کم و واریانس زیاد است. اگر k_n بسیار بزرگ باشد، تعداد زیادی از داده‌ها را در نظر می‌گیرد و نمی‌تواند تخمین دقیقی باشد و بایاس زیاد می‌شود.
به طور کلی، کنترل مقدار h_n در روش پارزن و k_n در روش KNN تاثیر بسزایی در bias-variance tradeoff خواهد داشت.

ب)



۲. الف)

2)

2.1) $p(x) = \begin{cases} \frac{1}{a} & 0 \le x \le a \\ 0 & \text{otherwise} \end{cases}$ ← سطح زیر آن باید 1 شود

$$\bar{p}_n(x) = E[p_n(x)] = E\left\{\frac{1}{Q h_Q^n}\sum_{q=1}^{Q}\phi\left(\frac{x-x^q}{h_Q}\right)\right\} = \frac{1}{Q}\sum_{q=1}^{Q} E\left\{\frac{1}{h_Q^n}\phi\left(\frac{x-x^q}{h_Q}\right)\right\} =$$

امید ریاضی یکسان بدلیل توزیع یکسان

$$= E\left\{\frac{1}{h_Q^n}\phi\left(\frac{x-x'}{h_Q}\right)\right\} = \int \frac{1}{h_Q^n}\phi\left(\frac{x-x'}{h_Q}\right) p(x')\,dx'$$

i) $x < 0 \rightarrow p(0) = 0$

ii) $0 \le x \le a \rightarrow \bar{p}_n(x) = \int_0^x \frac{1}{h_Q^n} e^{-\left(\frac{x-x'}{h_Q}\right)} \cdot \frac{1}{a}\,dx' = \frac{e^{-x/h_Q}}{h_Q^n}\cdot\frac{1}{a}\cdot h_Q e^{\frac{i}{h_Q}}\Big]_{i=0}^{x}$

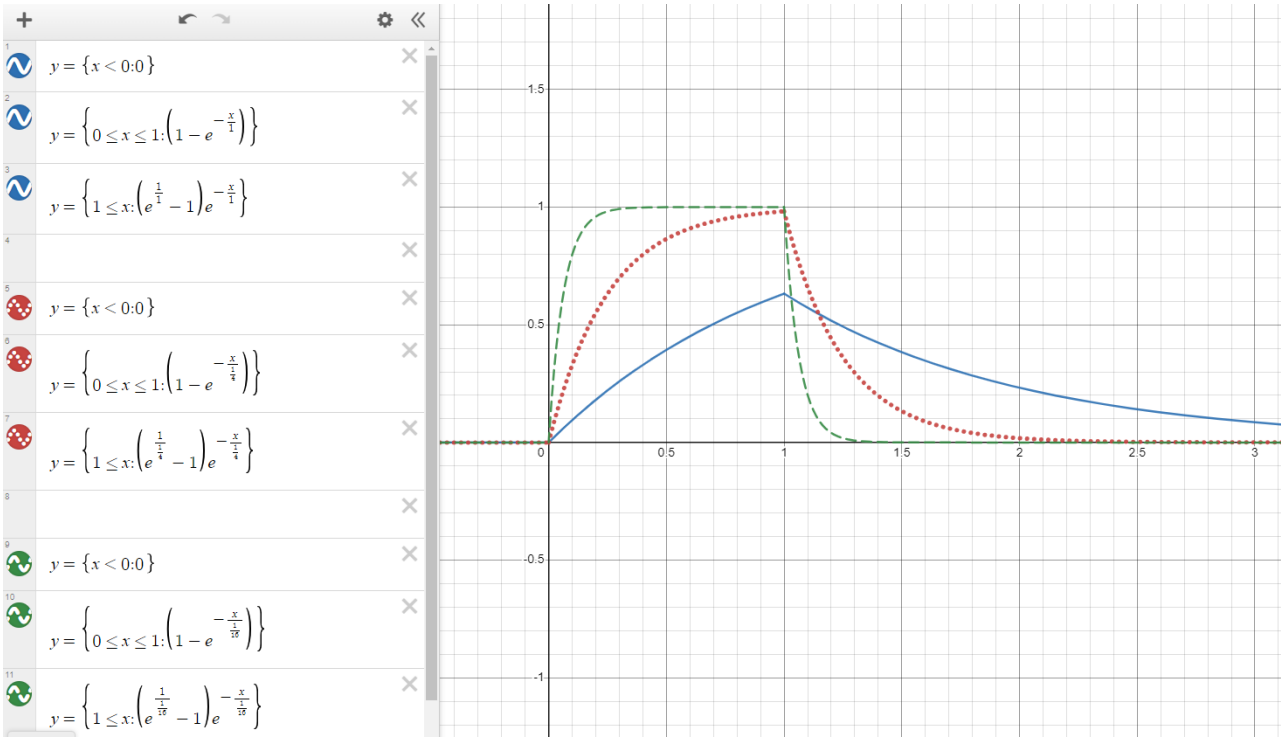
$$= \frac{1}{a} e^{-x/h_Q}\left(e^{x/h_Q} - 1\right) = \frac{1}{a}\left(1 - e^{-x/h_Q}\right)$$

iii) $x \ge a \rightarrow \bar{p}_n(x) = \int_0^a \frac{1}{h_Q^n} e^{-\left(\frac{x-x'}{h_Q}\right)} \cdot \frac{1}{a}\,dx' = \frac{e^{-x/h_Q}}{a h_Q^n}\cdot h_Q e^{i/h_Q}\Big]_{i=0}^{a} =$

$$= \frac{1}{a} e^{\frac{-x}{h_Q}}\left(e^{a/h_Q} - 1\right)$$

$$\Rightarrow \bar{p}_n(x) = \begin{cases} 0 & 0 \\ \frac{1}{a}\left(1 - e^{-x/h_n}\right) & 0 \le x \le a \\ \frac{1}{a}\left(e^{a/h_n} - 1\right) e^{-x/h_n} & a \le x \end{cases}$$

ب) در تصویر زیر نمودارهای میانگین تخمین‌ها مشاهده می‌شود. رنگ آبی برای h=۱، رنگ قرمز برای h=۱/٤ و رنگ سبز برای h=۱/۱٦ است.



پ)

2.3) $\text{bias} = p(x) - \bar{p}_n(x) = \begin{cases} 0 & x<0 \\ \frac{1}{a} - \frac{1}{a}\left(1 - e^{\frac{-x}{h_n}}\right) = \frac{1}{a} e^{\frac{-x}{h_n}} & 0 \le x \le a \\ 0 - \frac{1}{a}\left(e^{\frac{a}{h_n}} - 1\right) e^{\frac{-x}{h_n}} = -\frac{1}{a} e^{\frac{-x}{h_n}}\left(e^{\frac{a}{h_n}} - 1\right) & a \le x \end{cases}$

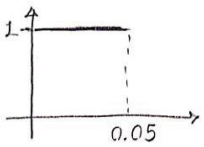
$$\text{bias} \le \frac{1}{100} \rightarrow \frac{p(x) - \bar{p}_n(x)}{p(x)} = \frac{\frac{1}{a} e^{\frac{-x}{h_n}}}{\frac{1}{a}} = a e^{\frac{-x}{h_n}} \le \frac{1}{100}$$

$$\xrightarrow{Ln} \frac{-x}{h_n} \le -Ln(100a) \rightarrow \frac{x}{h_n} \ge Ln(100a) \rightarrow \frac{h_n}{x} \le \frac{1}{Ln(100a)} \rightarrow h_n \le \frac{x}{Ln(100a)}$$

ت)

2.4) $\frac{1}{a}\left(1 - e^{\frac{-x}{h_N}}\right) \rightarrow h_N = \frac{x}{Ln(100)}$

$$\bar{p}_n(x) = 1 - e^{\frac{-x}{x/Ln100}} = 1 - \frac{e^{-Ln100}}{e^{\frac{1}{0.100}}}$$



۳. الف)

3) $\begin{cases} p(x) \sim N(\mu, \sigma^2) \\ \Phi(x) \sim N(0,1) \end{cases} \Rightarrow \begin{cases} p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \\ \Phi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \end{cases}$

3.1) $\bar{p}_n(x) = E[p(x)] = E\left\{ \frac{1}{Q h_Q} \sum_{q=1}^{Q} \Phi\left(\frac{x-x_q}{h_Q}\right) \right\} = E\left\{ \frac{1}{h_Q} \Phi\left(\frac{x-x'}{h_Q}\right) \right\} =$

$= \frac{1}{h_Q} \int \Phi\left(\frac{x-x'}{h_Q}\right) p(x')\, dx' = \frac{1}{h_Q} \int \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-1}{2}\left(\frac{x-x'}{h_Q}\right)^2\right) \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x'-\mu)^2}{2\sigma^2}\right) dx'$

$= \frac{1}{h_Q} \int \frac{1}{2\pi\sigma} \exp\left[\frac{-1}{2}\left(\frac{x-x'}{h_Q}\right)^2 - \frac{1}{2\sigma^2}(x'-\mu)^2\right] dx' =$

$= \frac{1}{h_Q} \cdot \frac{1}{2\pi\sigma} \exp\left(\frac{-x^2}{2h_Q^2} - \frac{\mu^2}{2\sigma^2}\right) \int_{-\infty}^{+\infty} \exp\left[\underbrace{\frac{-x'^2}{2h_Q^2} + \frac{xx'}{h_Q^2} - \frac{x'^2}{2\sigma^2} + \frac{x'\mu}{\sigma^2}}_{-\frac{x'^2}{2}\left(\frac{1}{h_Q^2} + \frac{1}{\sigma^2}\right) + x'\left(\frac{x}{h_Q^2} + \frac{\mu}{\sigma^2}\right)}\right] dx'$

$\frac{1}{2}\frac{B^2}{A^2} \exp\left(\frac{-1}{2}\left(\frac{x'-B}{A}\right)^2\right)$

define $\begin{cases} A^2 = \frac{1}{\frac{1}{h_Q^2} + \frac{1}{\sigma^2}} = \frac{h_Q^2 \sigma^2}{h_Q^2 + \sigma^2} \\ B = A^2\left(\frac{x}{h_Q^2} + \frac{\mu}{\sigma^2}\right) \end{cases}$

$\rightarrow \bar{p}_n(x) = \frac{\sqrt{2\pi}\, A}{2\pi h_Q \sigma} \exp\left[\frac{-1}{2}\left(\frac{x^2}{h_Q^2} + \frac{\mu^2}{\sigma^2}\right) + \frac{1}{2}\cdot\frac{B^2}{A^2}\right] =$

$= \frac{1}{\sqrt{2\pi} h_Q \sigma} \cdot \frac{h_Q \sigma}{\sqrt{h_Q^2 + \sigma^2}} \exp\left[\frac{-1}{2}\left(\underbrace{\frac{x^2}{h_Q^2} + \frac{\mu^2}{\sigma^2} - \frac{B^2}{A^2}}\right)\right]$

$= \frac{x^2}{h_Q^2} + \frac{\mu^2}{\sigma^2} - \frac{A^4}{A^2}\left(\frac{x}{h_Q^2} + \frac{\mu}{\sigma^2}\right)^2 = \frac{(x-\mu)^2}{h_Q^2 + \sigma^2}$

$\Rightarrow \bar{p}_n(x) = \frac{1}{\sqrt{2\pi(h_Q^2 + \sigma^2)}} \exp\left[\frac{-1}{2} \cdot \frac{(x-\mu)^2}{h_Q^2 + \sigma^2}\right] \Rightarrow \text{gaussi}(\mu, h_Q^2 + \sigma^2)$

ب)

3.2) $$p(x)-\bar{p}_n(x)=\frac{1}{\sqrt{2\pi}\,\sigma}\exp\left[\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]-\frac{1}{\sqrt{2\pi(h_Q^2+\sigma^2)}}\exp\left[\frac{-1}{2}\,\frac{(x-\mu)^2}{h_Q^2+\sigma^2}\right]=$$

$$=\frac{1}{\sqrt{2\pi}\,\sigma}\exp\left[\frac{-1}{2}\cdot\frac{(x-\mu)^2}{h_Q^2+\sigma^2}\right]\cdot\left[1-\frac{\sigma}{\sqrt{h_Q^2+\sigma^2}}\exp\left[\frac{-1}{2}\cdot\frac{(x-\mu)^2}{h_Q^2+\sigma^2}+\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]\right]=$$

$$=p(x)\left[1-\frac{1}{\sqrt{1+\left(\frac{h_Q}{\sigma}\right)^2}}\exp\left[\frac{1}{2}\cdot\frac{h_Q^2(x-\mu)^2}{h_Q^2+\sigma^2}\right]\right]$$

برای h کوچک : $\dfrac{1}{\sqrt{1+(h_Q/\sigma)^2}}\to 1-\frac{1}{2}\left(\frac{h_Q}{\sigma}\right)^2$

صرف نظر از جملات بالای h $\Longrightarrow$

$$p(x)-\bar{p}_n(x)=p(x)\left[1-\left(1-\frac{1}{2}\left(\frac{h_Q}{\sigma}\right)^2\right)\left(1+\frac{h_Q^2}{2\sigma^2}\cdot\frac{(x-\mu)^2}{h_Q^2+\sigma^2}\right)\right]$$

$$1-1+\frac{1}{2}\cdot\frac{h_Q^2}{\sigma^2}-\frac{h_Q^2}{2\sigma^2}\,\frac{(x-\mu)^2}{h_Q^2+\sigma^2}$$

$$\to\frac{1}{2}\left(\frac{h_Q}{\sigma}\right)^2\left[1-\frac{(x-\mu)^2}{h_Q^2+\sigma^2}\right]p(x)=\frac{1}{2}\left(\frac{h_Q}{\sigma}\right)^2\left[1-\left(\frac{x-\mu}{\sigma}\right)^2\right]p(x)$$

پ)

3.3) $\text{var}\left[p_n(x)\right] = \text{var}\left[\frac{1}{nh_Q}\sum \Phi\left(\frac{x-x_i}{h_Q}\right)\right] = \frac{1}{n^2h_Q^2}\sum \underbrace{\text{var}\,\Phi\left(\frac{x-x_i}{h_Q}\right)}_{n\,\text{var}\,\Phi(\,)} =$

$= \frac{1}{nh_Q^2}\left[\underbrace{E\left[\Phi^2(\,)\right]}_{\downarrow} - E^2\left[\Phi(\,)\right]\right] = \text{(I)} + \text{(II)}$

$\rightarrow E\left[\Phi^2(\,)\right] = \int \Phi^2\left(\frac{x-v}{h_Q}\right) p(v)\,dv =$

$= \int_{-\infty}^{+\infty} \frac{1}{2\pi}\exp\left[-\left(\frac{x-v}{h_Q}\right)^2\right]\exp\left[\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]\frac{1}{\sqrt{2\pi}\sigma}\,dv$

$\xrightarrow{h_Q/\sqrt{2}}$ $\frac{1}{h_Q/\sqrt{2}}\int_{-\infty}^{+\infty}\frac{1}{\sqrt{2\pi}}\exp\left[\frac{-1}{2}\left(\frac{x-v}{h_Q/\sqrt{2}}\right)^2\right]\exp\left[\frac{-1}{2}\cdot\left(\frac{x-\mu}{\sigma}\right)^2\right]\frac{1}{\sqrt{2\pi}\sigma}\,dv =$

$= \frac{1}{\sqrt{2\pi\left(\frac{h_Q^2}{2}+\sigma^2\right)}}\exp\left[\frac{-1}{2}\cdot\frac{(x-\mu)^2}{\frac{h_Q^2}{2}+\sigma^2}\right]$

$\text{(I)} = \frac{1}{nh_Q^2}E\left[\Phi^2\left(\frac{x-v}{h_Q}\right)\right] = \frac{1}{nh_Q^2/2\sqrt{\pi\cdot 2\pi(h_Q^2+\sigma^2)}}\exp\left[\frac{-1}{2}\cdot\frac{(x-\mu)^2}{\frac{h_Q^2}{2}+\sigma^2}\right]$

$\xrightarrow{h\to 0}$ $\lim_{h\to 0}\sqrt{\frac{h_Q^2}{2}+\sigma^2} = \sigma$ $\rightarrow \frac{1}{2nh_Q\sqrt{\pi}}\,p(x)$

$\text{(II)} = \frac{1}{nh_Q^2}\left(E\left[\Phi\left(\frac{x-v}{h_Q}\right)\right]\right)^2 = \frac{1}{nh_Q^2}\,h_Q^2\cdot\frac{1}{\sqrt{2\pi(h_Q^2+\sigma^2)}}\exp\left[\frac{-1}{2}\cdot\frac{(x-\mu)^2}{h_Q^2+\sigma^2}\right] =$

$= \lim_{h\to 0}\frac{h_Q}{nh_Q\cdot\sqrt{2\pi}\sigma}\exp\left[\frac{-1}{2}\cdot\left(\frac{x-\mu}{\sigma}\right)^2\right] = 0$

$\rightarrow \text{var}\left[p_n(x)\right] = \text{(I)} + \text{(II)} = \frac{1}{2nh_Q\sqrt{\pi}}\,p(x)$

۴. الف)

4.1) $P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)} \sim P(x|\theta)P(\theta)$

E-step

توزیع دلخواه Q $\rightarrow \left[\sum_z Q(z)\frac{P(x,z|\theta)}{Q(z)}\right] \cdot P(\theta)$

$$L(\theta) = \log \prod P(\theta|x_i) = \sum_{i=1}^{n} \log P(\theta|x_i) = \sum_{i=1}^{n}\left[\log \sum_z Q(z)\frac{P(x_i,z|\theta)}{Q(z)} + \log P(\theta)\right]$$

$$\overset{\text{jensen}}{\geq} \sum_i^n \sum_z Q(z)\left[\log P(x_i,z|\theta) - \log(Q(z))\right] + \log P(\theta)$$

به ازای

$$\rightarrow \sum_{i=1}^{n} \underset{z}{E_{(i)}}{}_{\sim Q_i}\left[\frac{P(x_i,z_i|\theta)}{Q(z_i)}\right] + \log P(\theta)$$

$$Q(\theta) = \underset{z}{E}\left[\log P(x,\theta) + \log P(\theta)\right]$$

M-step

$$\partial Q / \partial \theta = \frac{\frac{\partial P(x,\theta)}{\partial \theta}}{P(x,\theta)} + \frac{\frac{\partial P(\theta)}{\partial \theta}}{P(\theta)}$$

$$\hat{\theta} = \underset{\theta}{\text{argmax}} \sum_i \sum_z Q(z)\left[\log P(x_i,z|\theta) - \log Q(z)\right] + \log P(\theta)$$

ب)

4.2) $n = n_a + n_b + n_c + n_d$ E-Step

$E[n_a] = n \cdot P(x = A|\theta) = n \times \frac{1}{3}$

$E[n_c] = n \cdot P(x = c|\theta) = n \times \frac{2\hat{\theta}}{3}$

$Q(\theta) = \sum_z E[n_t] \cdot \log P(x|\theta) + \log P(\theta)$

$= \underbrace{\hat{n}_a}_{n/3} \log \frac{1}{3} + n_b \log \frac{1}{3}(1-\hat{\theta}) + \underbrace{\hat{n}_c}_{n \cdot \frac{2\hat{\theta}}{3}} \cdot \log \frac{2}{3}\hat{\theta} + n_d \log \frac{1}{3}(1-\hat{\theta}) + \log P(\theta)$

$= n/3 \log \frac{1}{3} + n_b \log \frac{1}{3} + n_b \log(1-\hat{\theta}) + n\frac{2\hat{\theta}}{3} \log \frac{2}{3}\hat{\theta} + n_d \log \frac{1}{3} + n_d \log(1-\hat{\theta})$

$+ \log \frac{\Gamma_{v_1+v_2}}{\Gamma_{v_1} + \Gamma_{v_2}} + (v_1 - 1)\log\hat{\theta} + (v_2 - 1)\log(1-\hat{\theta})$

M-step

$\frac{\partial Q(\theta)}{\partial \theta} = \frac{\partial}{\partial \theta} \log(1-\hat{\theta})\left[n_b + n_d + v_2 - 1\right] + \log(\hat{\theta})\left[\frac{n2\hat{\theta}}{3} + v_1 - 1\right]$

$= (n_b + n_d + v_2 - 1)\frac{-1}{1-\hat{\theta}} + \frac{1}{\hat{\theta}} \cdot \frac{2n\hat{\theta}}{3} + \frac{2n}{3}\log\hat{\theta} + \frac{v_1 - 1}{\hat{\theta}} = 0$

۵. الف)

5.1) (I) $= \binom{m_t}{w_t} p_k^{w_t} (1-p_k)^{m_t - w_t}$

(II) $= \binom{m_t}{w_t} p_k^{w_t} (1-p_k)^{m_t - w_t} \cdot c_k$

ب)

5.2) $Q(\theta) = E_z[L(\theta)]$ z: hidden variable → کدام بازیکن در کدام روز

$$p(x,z|\theta) = \prod_{k=1}^{K} \left[c_k \, \textcircled{I} \right]^{z_{ik}}$$

$$L(\theta) = \log p(D,H|\theta) = \sum_{i=1}^{n} \log p(x_i, z_i|\theta) =$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \left[\log c_k + \log \textcircled{I} \right]$$

$$Q(\theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} \underbrace{E[z_{ik}]}_{} \left(\log c_k + \log \textcircled{I} \right)$$

$$E[z_{ik}] = pr(z_{ik}=1 | x_i, \theta^{(t)}) \xrightarrow{\text{iter}} = \frac{p(x_i|z_{ik}=1,\theta)\, p(z_{ik}=1|\theta^t)}{p(x_i|\theta^t)} =$$

$$= \frac{\textcircled{I} \cdot c_k^t}{\sum_{k=1}^{K} c_k^t \, \textcircled{I}} = \frac{\binom{m_t}{w_t} p_k^{w_t} (1-p_k)^{m_t - w_t} \cdot c_k^t}{\sum_{k=1}^{K} c_k^t \binom{m_t}{w_t} p_k^{w_t} (1-p_k)^{m_t - w_t}} = \gamma_{ik}^t$$

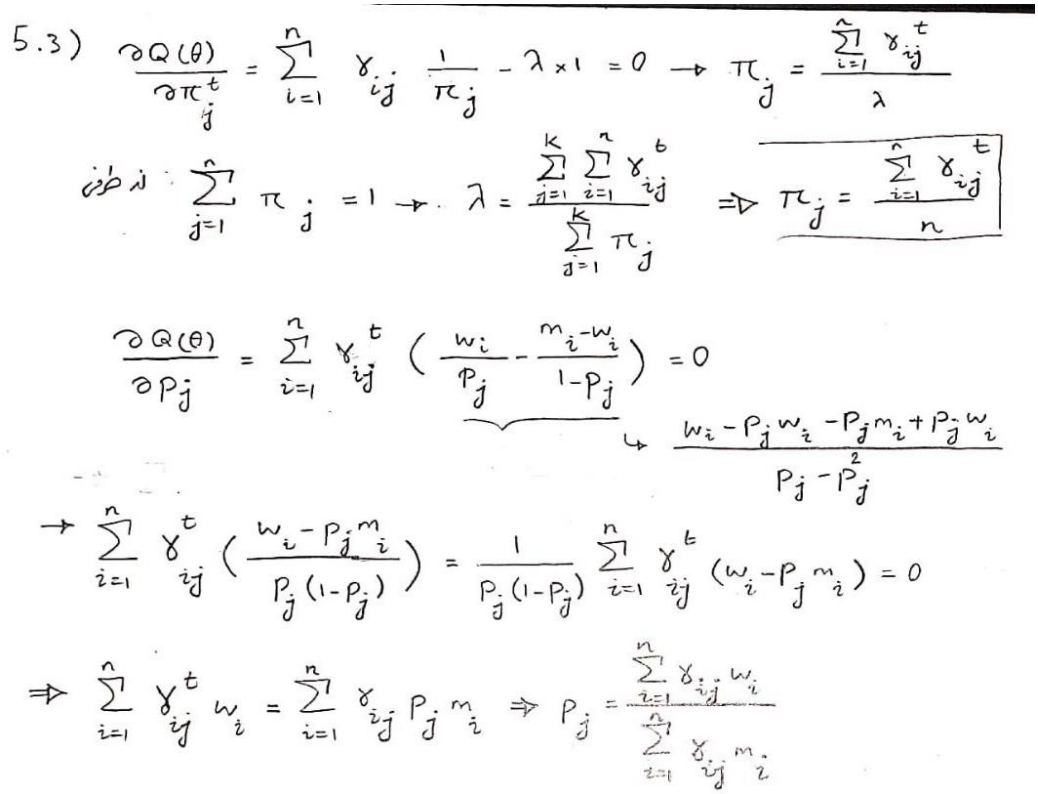
احتمال اینکه توپ i مربوط به بازیکن k باشد (در روز t ام)

$$Q(\theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik}^t \left(\log c_k + \log \textcircled{I} \right)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik}^t \left[\log c_k + \log \binom{m_i}{w_i} + w_t \log p_k + (m_t - w_t) \log (1-p_k) \right]$$

(c همان π است)

پ)



۶. .

۶,۱. طبقه بند knn را در یک کلاس پیاده سازی میکنیم. در اینجا k متغیر کنترلی است و تعداد نمونه هایی که داخل همسایگی قرار می گیرند را تعیین میکند. درجه آزادی این مسئله، فاصله است که ما از فاصله اقلیدسی استفاده می کنیم. در تابع predict فاصله نمونه داده را با تمامی داده ها می سنجیم و آنها را مرتب می کنیم. سپس برچسب k المان اول را بررسی می کنیم و بر اساس majority voting، برچسب داده جدید را پیش بینی می کنیم. در تابع evaluate، یک batch از داده را به عنوان ورودی دریافت می کنیم و برچسب آن را پیش بینی می کنیم. بر اساس پیش بینی ها، مقدار دقت را محاسبه می کنیم.

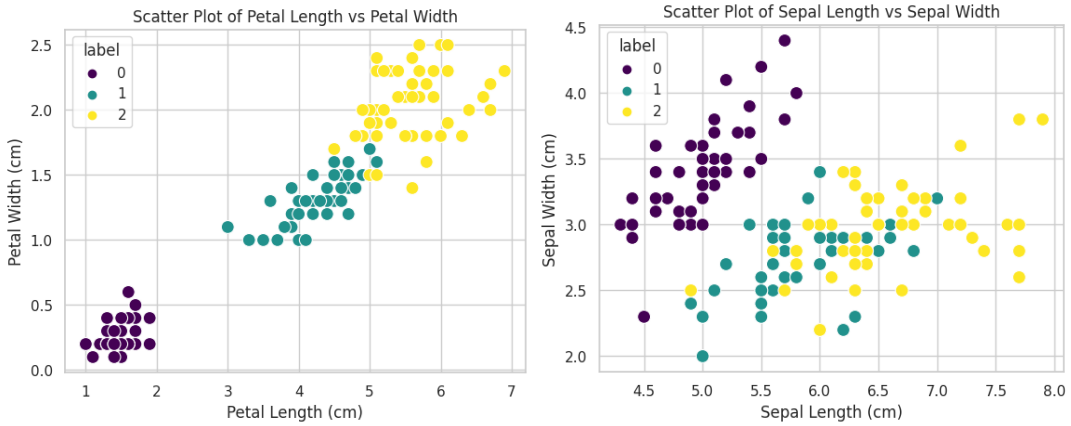
```python
class KNN:
    def __init__(self, k, x_train, y_train):
        self.k = k
        self.x_train = x_train
        self.y_train = y_train

    def dist(self, x1, x2):
        return np.linalg.norm(x1 - x2)

    def predict(self, x):
        # calculate distance of x with each data point
        distances = [self.dist(x, data_point) for data_point in self.x_train]
        sorted_dist = np.argsort(distances)
        k_indices = sorted_dist[:self.k]
        k_labels = [y_train[i] for i in k_indices]
        # majority voting
        prediction = np.bincount(k_labels).argmax()
        return prediction

    def evaluate(self, test_batch, test_batch_label):
        acc = 0
        for (sample, label) in zip(test_batch, test_batch_label):
            p = self.predict(sample)
            if p == label:
                acc += 1

        acc /= len(test_batch)
        return acc
```

۶,۲. مجموعه داده iris را لود می کنیم. این مجموعه از ۳ کلاس گل تشکیل شده و هر نمونه داده، ۴ ویژگی دارد. از هر کلاس ۵۰ داده و در کل ۱۵۰ داده داریم.

```python
data = load_iris()
x = data.data
y = data.target

iris_df = pd.DataFrame(data.data, columns=data.feature_names)
iris_df['label'] = y

print("features", data.feature_names)
print("labels:", data.target_names)
print("shape of x:", x.shape)
print("shape of y:", y.shape)
```

```
features ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
labels: ['setosa' 'versicolor' 'virginica']
shape of x: (150, 4)
shape of y: (150,)
```

۶,۳. دو scatter plot رسم می کنیم که در یکی ویژگی sepal length و sepal width و در نمودار دیگر petal length و petal width را در نظر می گیریم. نمودارها به صورت زیر هستند:



۶,۴. دیتاست را به دو بخش آموزش و آزمون تقسیم می کنیم. ۲۰ درصد از داده ها را به عنوان داده آزمون در نظر میگیریم.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=82)

print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (120, 4)
y_train shape: (120,)
X_test shape: (30, 4)
y_test shape: (30,)
```

۶,۵. طبقه بند را به ازای k=۵ آموزش می دهیم و برای داده آزمون دقت را محاسبه می کنیم. برای داده آموزش دقت ۹۷٪ و داده آزمون ۹۳٪ است.

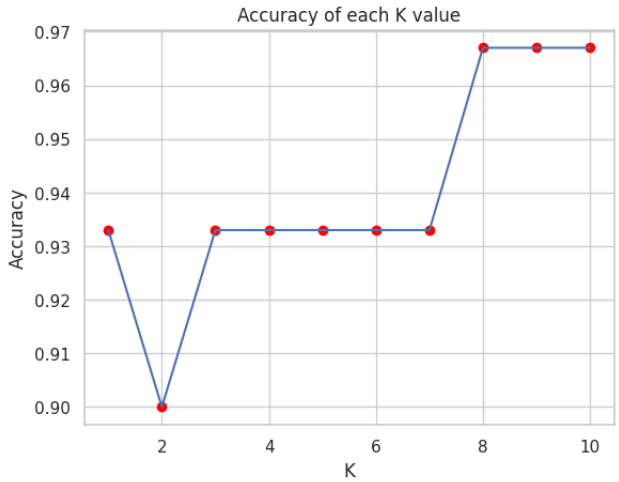
```
knn_classifier = KNN(5, X_train, y_train)

# evaluate on train data
train_acc = knn_classifier.evaluate(X_train, y_train)
print("Train Accuracy: ", train_acc)

# evaluate on test data
test_acc = knn_classifier.evaluate(X_test, y_test)
print("Test Accuracy: ", test_acc)
```

```
Train Accuracy:  0.975
Test Accuracy:  0.9333333333333333
```

۶,۶. در این قسمت برای مقادیر مختلف k طبقه بند را آموزش می دهیم و دقت آن را ارزیابی می کنیم.

```
accuracy = []

for k in range(1, 11):
    k_classifier = KNN(k, X_train, y_train)
    accuracy.append(k_classifier.evaluate(X_test, y_test))
```

نمودار دقت آزمون بر حسب مقدار k به صورت زیر است:



بر اساس این نمودار، بهترین مقدار k برای این توزیع از داده آزمون، برابر با ۸ است.

۷. ابتدا داده را تولید میکنیم.

```
import numpy as np

N = 1000
np.random.seed(1)
X = np.concatenate((np.random.normal(0, 1, int(0.3 * N)),
                    np.random.normal(5, 1, int(0.7 * N))))[:, np.newaxis]
```

```
print(max(X))
print(min(X))
```

```
[8.9586027]
[-2.793085]
```

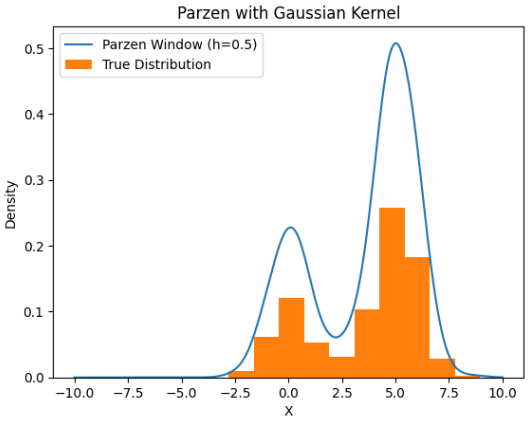
۷،۲. توزیع گاوسی را از طریق تابع gaussian_kernel پیاده سازی میکنیم. به عنوان ورودی نمونه داده، میانگین و واریانس دریافت میکند و مطابق فرمول گاوسی مقدار آن را محاسبه میکند. سپس از تخمین زن نقطه ای پارزن با پنجره گاوسی مطابق فرمول زیر استفاده میکنیم:

$$p_n(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^d}\phi\left[\frac{1}{(\sqrt{2\pi})^d h_n^d}\,exp\left[-\frac{1}{2}\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h_n}\right)^2\right]\right]$$

در این فرمول h متغیر کنترلی است و از طریق آن اندازه همسایگی را کنترل میکنیم. در کرنل گاوسی، واریانس همان h است. از طریق فرمول گاوسی، تعداد داده هایی که داخل همسایگی تحت یک مرکز قرار می گیرد را به صورت soft میشماریم تا با کمک آنها تخمین بزنیم.

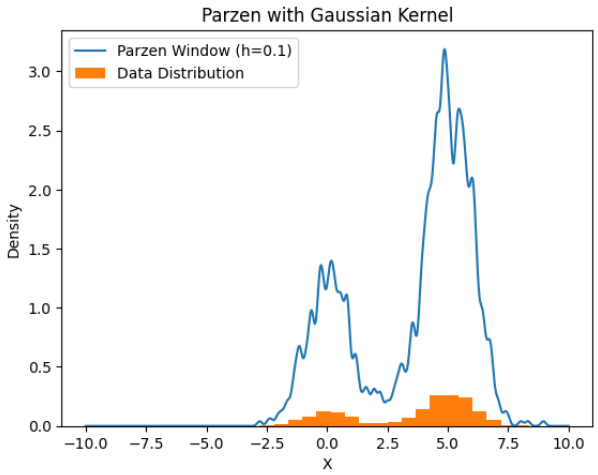
```python
def gaussian_kernel(x, mu, sigma):
    return (1 / (np.sqrt(2 * np.pi) * sigma)) * np.exp(-((x - mu) / sigma) ** 2 / 2)

def parzen_estimation(x_queries, x0, h, n):
    k_q = 0
    for x in x_queries:
        u = gaussian_kernel(x0, x, h)
        k_q += u
    return k_q / (len(x_queries) * (h**n))
```

مقدار h را بر اساس آزمون و خطا، ۱ تخمین می زنیم. ۱۰۰۰ نقطه به عنوان مرکز برای پنجره در نظر می گیریم و حول آنها تخمین می زنیم. نتیجه به صورت زیر است:

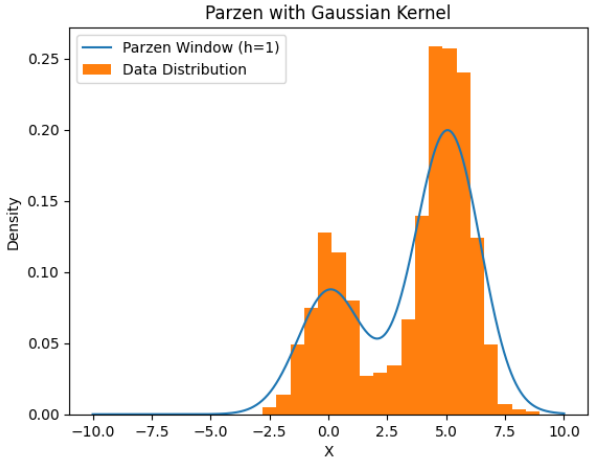


۷,۳. برای ۳ مقدار مختلف h تخمین می زنیم و آن را رسم میکنیم.

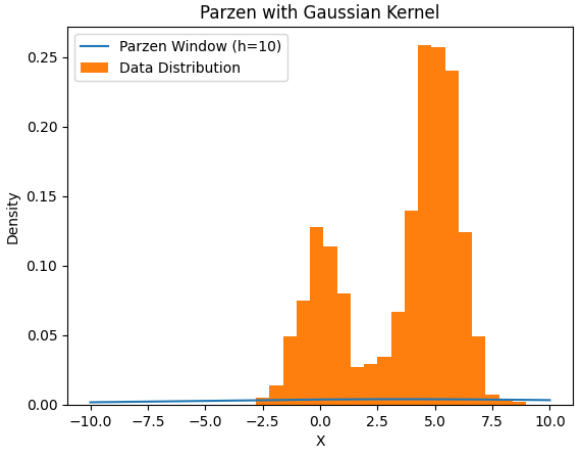
h=۰,۱



h=۱



h=۱۰



هرچه مقدار h بیشتر باشد، کم و زیاد شدن مقادیر را در نظر نمی گیرد و داده های زیادی در داخل پنجره قرار می گیرند. مثلا h=۱۰ به تابع ثابت نزدیک است. اما اگر مقدار h کم باشد، تعداد کمی از داده ها داخل پنجره قرار می گیرند و تخمین بسیار حساس می شود. h=۱ می تواند مقدار بهتری میان این ۳ مقدار باشد.

۸.

۸,۱. دیتاست را تشکیل میدهیم. از دو ماه شکل ایجاد شده و تعداد نمونه های هر یک از ماه ها ۲۵۰ است.

```
from sklearn import cluster, datasets, mixture
import matplotlib.pyplot as plt

noisy_moons = datasets.make_moons(n_samples=500, noise=0.11)
```

```
features, labels = noisy_moons

moon_1 = features[labels==0]
moon_2 = features[labels==1]
```

```
print(features.shape)
print(np.unique(labels))
print(moon_1.shape)
print(moon_2.shape)
```

```
(500, 2)
[0 1]
(250, 2)
(250, 2)
```

۸,۲. از تابع calculate_params برای محاسبه مقدار میانگین و ماتریس کوواریانس استفاده میکنیم. از آنجایی که تعداد ویژگی های داده ۲ است، ماتریس کوواریانس ۲*۲ خواهد بود.

سپس تابع gaussian را با توجه به پارامترهای توزیع گاوسی (میانگین و کوواریانس) مطابق فرمول پیاده سازی میکنیم.

$$p(\boldsymbol{x}) = \frac{1}{\left|\sqrt{2\pi\boldsymbol{\Sigma}}\right|} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

```
def calculate_params(data):
    f1 = data[:, 0]
    f2 = data[:, 1]

    mean_1 = np.sum(f1) / len(f1)
    mean_2 = np.sum(f2) / len(f2)
    mean_matrix = np.array([mean_1, mean_2])

    cov_11 = np.sum((f1-mean_1)**2) / len(f1)
    cov_22 = np.sum((f2-mean_2)**2) / len(f2)
    cov_12 = np.sum((f1-mean_1) * (f2-mean_2)) / len(f1)

    cov_matrix = np.array([[cov_11, cov_12],
                           [cov_12, cov_22]])

    return mean_matrix, cov_matrix


def gaussian(x, mean_matrix, cov_matrix):

    det = np.linalg.det(cov_matrix)
    cov_inverse = np.linalg.inv(cov_matrix)
    diff = x - mean_matrix
    exp = np.exp(np.sum(np.dot(diff, cov_inverse) * (diff), axis=1) / -2)

    return (1 / np.sqrt((np.pi * 2) * det)) * exp
```

پارامترهای هر یک از گاوسی ها به صورت زیر است:

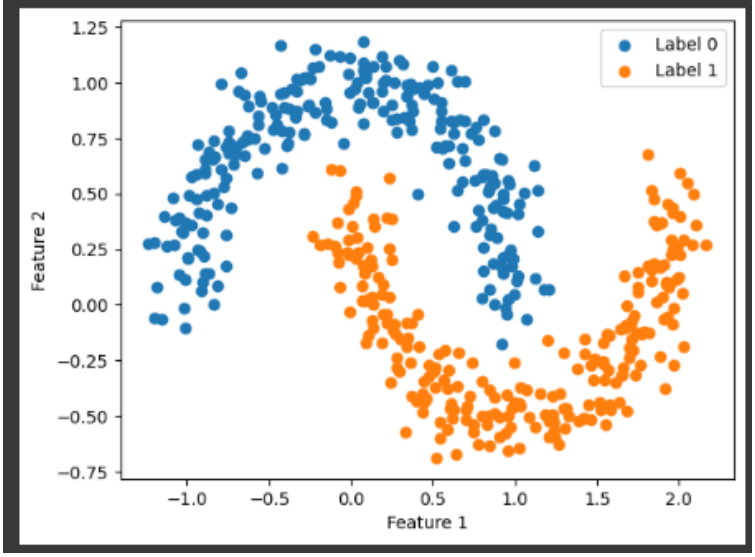
```
mean_moon_1, cov_moon_1 = calculate_params(moon_1)
mean_moon_2, cov_moon_2 = calculate_params(moon_2)

print("Label 0")
print("mean:", mean_moon_1)
print("covariance:", cov_moon_1)
print()
print("Label 1")
print("mean:", mean_moon_2)
print("covariance:", cov_moon_2)
```

```
Label 0
mean: [-0.00499394  0.63768695]
covariance: [[ 0.51689669 -0.0033852 ]
 [-0.0033852   0.10652042]]

Label 1
mean: [ 0.99872771 -0.13053931]
covariance: [[ 0.5020686  -0.00580147]
 [-0.00580147  0.11087562]]
```

برای مشاهده توزیع داده ها، scatter plot آن را رسم میکنیم:

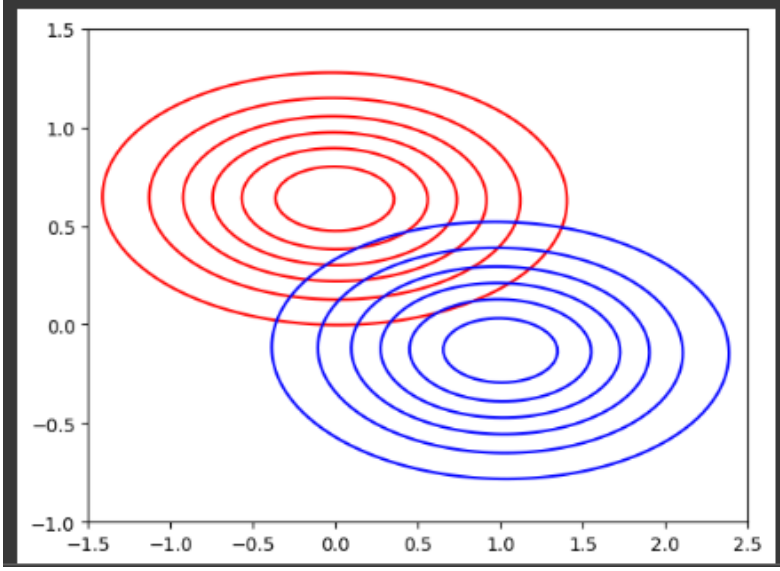
```
def scatter_plot(moon_1, moon_2):
    # Scatter plot for each moon
    plt.scatter(moon_1[:, 0], moon_1[:, 1], label='Label 0')
    plt.scatter(moon_2[:, 0], moon_2[:, 1], label='Label 1')

    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.legend()
    plt.show()
```

```
scatter_plot(moon_1, moon_2)
```



در آخر این قسمت، کانتورهای هر یک از گاوسی ها را رسم میکنیم:

```
x, y = np.meshgrid(np.linspace(-1.5, 2.5, 100), np.linspace(-1, 1.5, 100))
points = np.c_[x.flatten(), y.flatten()]

z1 = np.array(gaussian(points, mean_moon_1, cov_moon_1) )
z1 = z1.reshape(x.shape)

z2 = np.array(gaussian(points, mean_moon_2, cov_moon_2))
z2 = z2.reshape(x.shape)

# Contour plot for the Gaussian
plt.contour(x, y, z1, colors='red')
plt.contour(x, y, z2, colors='blue')

plt.show()
```



۸،۳. کلاس GMM را پیاده سازی میکنیم. در constructor آن داده، تعداد componentها و تعداد iterationها را به عنوان ورودی دریافت میکنیم. سپس برای هر یک از componentها میانگین (برای هر یک از ویژگی ها) را به صورت تصادفی و ماتریس کوواریانس را با ماتریس همانی مقداردهی اولیه میکنیم. وزن همه componentها را هم به صورت برابر مقداردهی میکنیم.

```
np.random.seed(28)

class GMM:
    def __init__(self, data, components_count, iters):
        self.components_count = components_count
        self.iters = iters
        self.data = data
        self.samples_count, self.features_count = data.shape

        # each component has a mean and covariance matrix
        # a list with length components_count, containing matrices of shape (1, features)
        self.means = [np.random.rand(data.shape[1]) for i in range(components_count)]
        self.covars = [np.eye(data.shape[1]) for i in range(components_count)]
        # also it has a probability of happening
        self.probs = [1/components_count] * components_count

```

از تابع fit برای تخمین پارامترها در iterationهای متوالی استفاده میکنیم. در E-step مقدار گاما (که امید ریاضی از متغیر مشاهده نشده است) را با فرمول زیر محاسبه میکنیم. این مقدار، احتمال اینکه نمونه ها از مولفه k آمده باشد (تا لحظه t) را نشان میدهد.

$$\gamma_k^n = P\left(z_k^{(n)} = 1 | \boldsymbol{x}^{(n)}, \boldsymbol{\theta}^{old}\right) = \frac{\pi_k^{old} \mathcal{N}(\boldsymbol{x}^{(n)} | \boldsymbol{\mu}_k^{old}, \boldsymbol{\Sigma}_k^{old})}{\sum_{j=1}^{K} \pi_j^{old} \mathcal{N}(\boldsymbol{x}^{(n)} | \boldsymbol{\mu}_j^{old}, \boldsymbol{\Sigma}_j^{old})}$$

در M-step، باید مقدار پارامترها را آپدیت کنیم. برای آپدیت هر یک از پارامترها از فرمول های زیر استفاده میکنیم:

$$\boldsymbol{\mu}_k^{new} = \frac{\sum_{n=1}^{N} \gamma_k^n \boldsymbol{x}^{(n)}}{\sum_{n=1}^{N} \gamma_k^n}$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{\sum_{n=1}^{N} \gamma_k^n} \sum_{n=1}^{N} \gamma_k^n (\boldsymbol{x}^{(n)} - \boldsymbol{\mu}_k^{\text{new}})(\boldsymbol{x}^{(n)} - \boldsymbol{\mu}_k^{\text{new}})^T$$

$$\pi_k^{\text{new}} = \frac{\sum_{n=1}^{N} \gamma_k^n}{N}$$

سپس شرط همگرایی را چک میکنیم. اگر وزن مولفه ها نسبت به تخمین آن تغییر نامحسوسی داشته باشد، مسئله همگرا شده است. در غیر این صورت از مقادیری که در این iteration محاسبه کردیم، برای مرحله بعدی استفاده میکنیم و در ۱۰۰ دور، E-step و M-step را انجام میدهیم.

```
17    def fit(self):
18        for t in range(self.iters):
19            # E-step
20            gaussian_all_comps = [gaussian(self.data, self.means[k], self.covars[k]) for k in range(self.components_count)]
21            gamma_t = [gaussian_comp / sum(gaussian_all_comps) for gaussian_comp in gaussian_all_comps]
22            gamma_t = [np.expand_dims(g, axis=1) for g in gamma_t]
23
24            # M-step
25            for k in range(self.components_count):
26                numerator = np.sum(gamma_t[k] * self.data, axis=0)
27                denominator = np.sum(gamma_t[k], axis=0)
28                self.means[k] = numerator / denominator
29
30                diff = self.data - self.means[k]
31                self.covars[k] = np.dot(np.squeeze(gamma_t[k]) * diff.T, diff) / np.sum(gamma_t[k], axis=0)
32
33                self.probs[k] = np.sum(gamma_t[k], axis=0) / self.samples_count
34
35            # check convergence
36            gamma_t_np = np.array(gamma_t).squeeze().T
37            if np.linalg.norm(self.probs - gamma_t_np.sum(axis=0) / self.samples_count) < 1e-3:
38                break
```

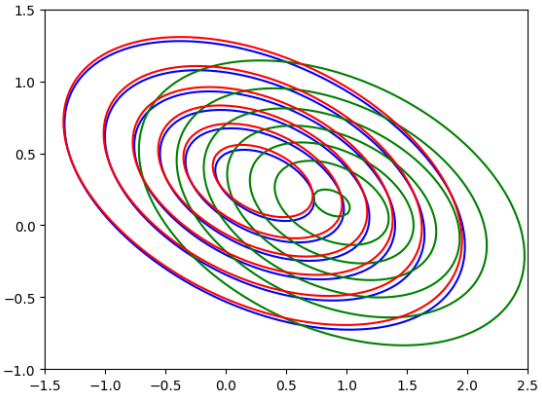
در آخر از تابع predict برای پیش بینی یک نمونه داده جدید بر اساس پارامترهایی که تخمین زدیم استفاده میکنیم. وزن هر مولفه را در مقدار گاوسی ضرب میکنیم. مقدار مربوط به هر مولفه که بزرگتر باشد، نمونه متعلق به آن مولفه است.

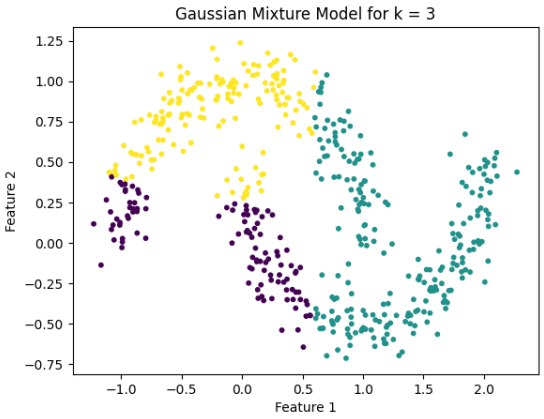
```
40      def predict(self, x_test):
41          p = np.zeros((x_test.shape[0], self.components_count))
42          for k in range(self.components_count):
43              p[:, k] = self.probs[k] * gaussian(x_test, self.means[k], self.covars[k])
44
45          predictions = np.argmax(p, axis=1)
46          return predictions
```

در یک حلقه، تعداد مولفه ها را از ۱ تا ۱۶ قرار میدهیم و پارامترها را تخمین می زنیم. Scatter plot و همچنین کانتورها را برای مقدار ۳، ۸ و ۱۶ رسم میکنیم.

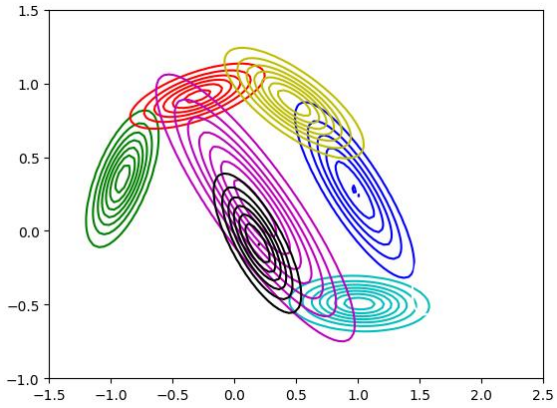
```
1 for i in range(1, 17):
2     gmm_classifier = GMM(features, i, 100)
3     gmm_classifier.fit()
4     predictions = gmm_classifier.predict(features)
5
6     colors_names = ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'w']
7
8     if i == 3 or i == 8 or i == 16:
9         # Plot the data points with cluster colors
10        plt.scatter(features[:, 0], features[:, 1], c=predictions, cmap='viridis', s=10)
11        plt.title(f'Gaussian Mixture Model for k = {i}')
12        plt.xlabel('Feature 1')
13        plt.ylabel('Feature 2')
14        plt.show()
15
16        x, y = np.meshgrid(np.linspace(-1.5, 2.5, 100), np.linspace(-1, 1.5, 100))
17        points = np.c_[x.flatten(), y.flatten()]
18        for j in range(i):
19            z = np.array(gaussian(points, gmm_classifier.means[j], gmm_classifier.covars[j]))
20            z = z.reshape(x.shape)
21
22            # Contour plot for the Gaussian
23            plt.contour(x, y, z, colors=colors_names[j%len(colors_names)])
24        plt.show()
25
26        print()
```

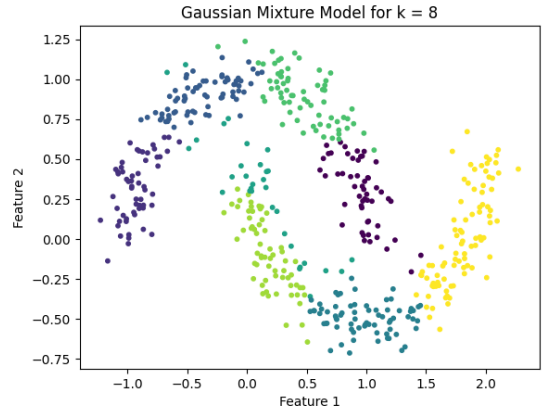
K=۳



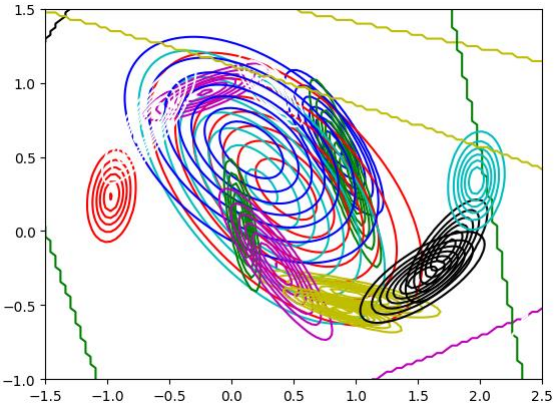

Gaussian Mixture Model for k = 3
Feature 1
Feature 2

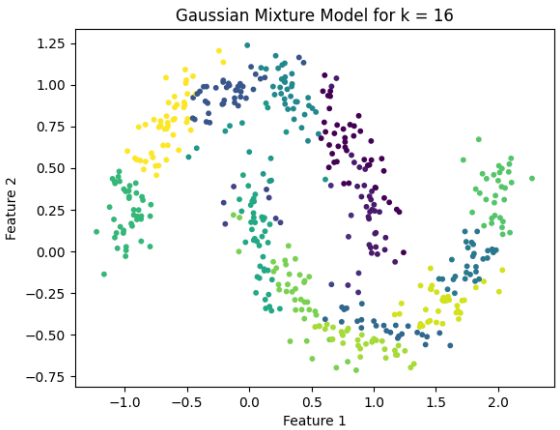
K=۸




Gaussian Mixture Model for k = 8
Feature 1
Feature 2

K=۱۶



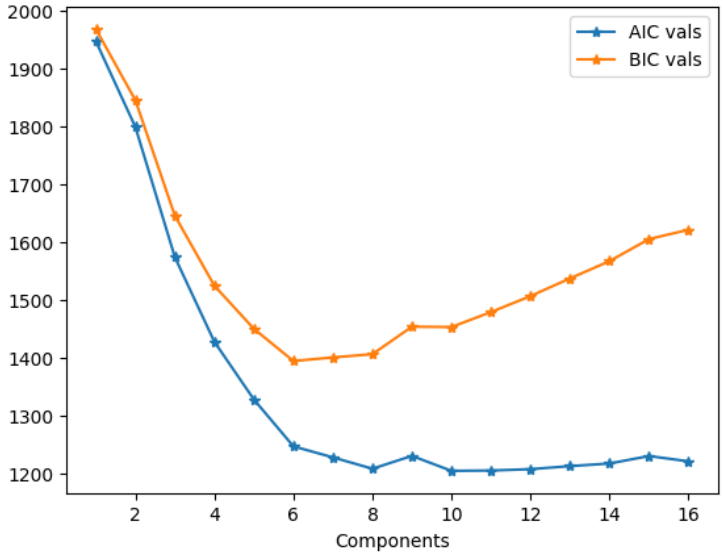

Gaussian Mixture Model for k = 16
Feature 1
Feature 2

۸,۴. AIC تخمینی از مقدار نسبی اطلاعات از دست رفته توسط یک مدل است. تناسب فیت شدن مدل را همراه با پیچیدگی آن در نظر می گیرد. BIC مشابه AIC است اما جریمه شدیدتری به مدل ها با پارامترهای بیشتر اختصاص می دهد. از دیدگاه بیزی بدست آمده و اغلب در انتخاب مدل های ساده تر محافظه کار تر است. فرمول این دو معیار به صورت زیر است:

$$\mathrm{AIC}_i = -2\log L_i + 2p_i$$

$$\mathrm{BIC}_i = -2\log L_i + p_i \log n$$

نمودار AIC و BIC را به ازای تعداد مولفه از ۱ تا ۱۶ رسم می کنیم:



مطابق این نمودار، تعداد مولفه بهینه بر اساس AIC برابر با ۱۰ و بر اساس BIC برابر با ۶ است. معیار BIC ترجیح می دهد مدل ساده تری انتخاب کند که در این مسئله هرچه تعداد مولفه ها کمتر باشد، تعداد پارامتر نیز کمتر و در نتیجه مسئله ساده تر خواهد شد.