خوشه‌بندی

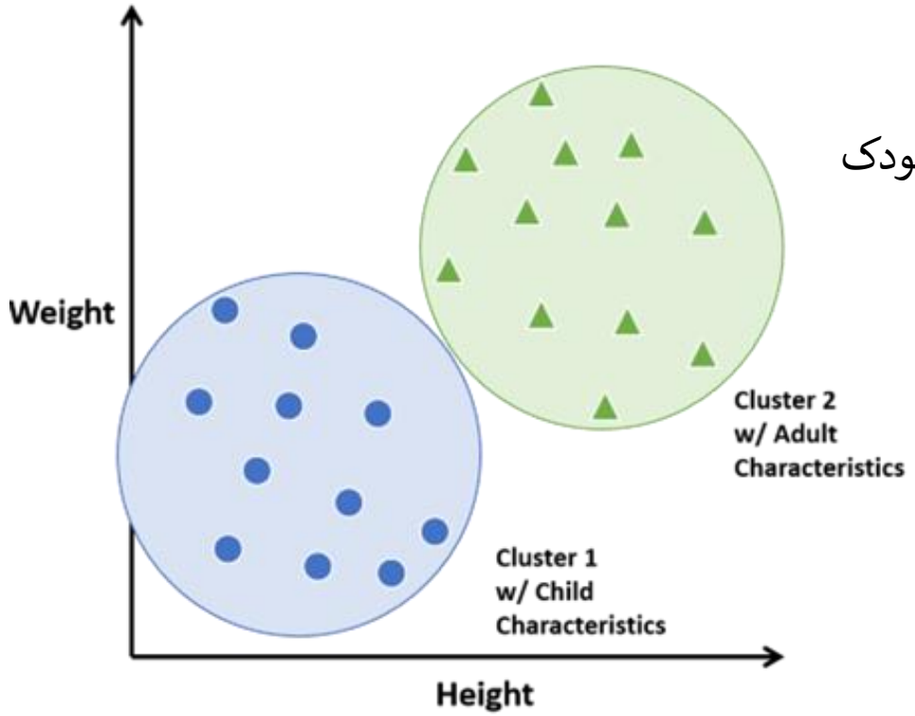
Clustering

هدف:
خوشه‌بندی کردن مجموعه‌ی nتایی داده‌ها به k گروه یا خوشه است.

$$C = \{C_1, C_2, \dots, C_k\}$$

این خوشه‌بندی بر مبنای مشابهت داده‌ها در فضای d بعدی مشخصه‌ها است.

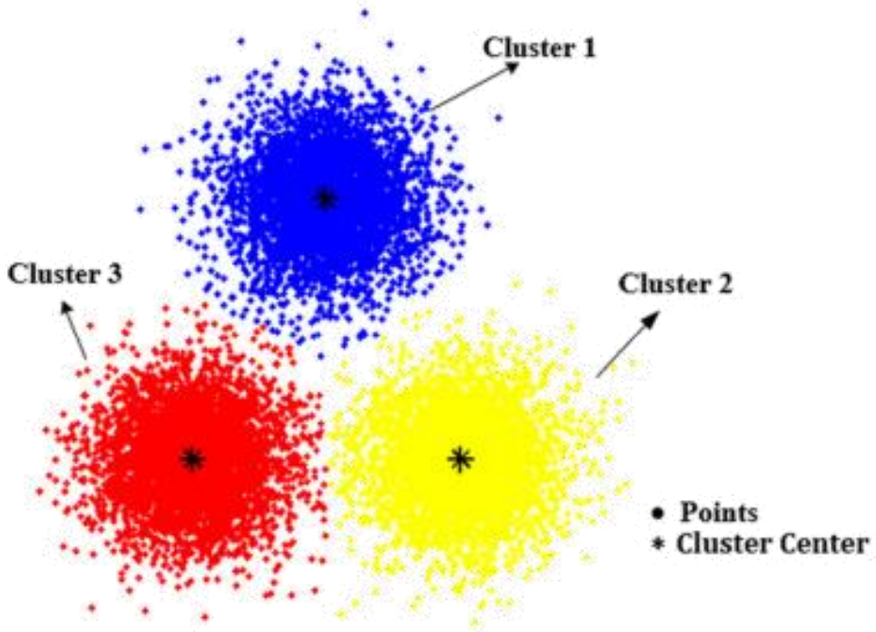
مثال:
مشخصه‌های قد و وزن داده‌ها را به دو گروه بزرگسال و کودک تقسیم می‌کند.



خوشه‌بندی بر مبنای نماینده

Representative-based Clustering

در این نوع خوشه‌بندی، برای هر خوشه نماینده‌ای معرفی می‌شود و هر داده به یکی از این نمایندگان نسبت داده می‌شود.



بدیهی‌ترین انتخاب برای نماینده یک خوشه نقطه‌ی مرکزی (Centroid Point) است.
که برای هر خوشه با میانگین‌گیری داده‌های منتسب به آن خوشه محاسبه می‌شود.

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{x_j \in C_i} \mathbf{x}_j$$

در این الگوریتم جمع مربع خطا (SSE) معیار خوشه‌بندی خوب می‌باشد.
خطا نیز فاصله‌ی هر داده از نماینده‌اش است. پس خوشه‌بندی بهتر است که داده‌ها فاصله کمتری با نماینده خود داشته باشند.

$$SSE(C) = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$

$$C^* = \arg\min_{C} \{SSE(C)\}$$

این الگوریتم در دو مرحله انجام می‌شود:
۱ – انتساب هر داده به یک خوشه با توجه به فاصله‌ی داده تا نماینده‌ی خوشه‌ها
۲ – به‌روز رسانی نماینده هر خوشه با توجه به به‌روز رسانی داده‌های انتساب داده شده به هر خوشه.

$$i^* = \arg\min_{i=1}^{k} \left\{ \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2 \right\}$$

این الگوریتم با انتخاب اتفاقی نماینده‌ها شروع می‌شود
و زمانی پایان می‌یابد که نماینده‌ها دیگر تغییری نکنند و ثابت بمانند.

$$\sum_{i=1}^{k} \|\boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1}\|^2 \leq \epsilon, \text{ where } \epsilon > 0$$

Algorithm 13.1: K-means Algorithm

K-MEANS $(\mathbf{D}, k, \epsilon)$**:**

$t = 0$

Randomly initialize k centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t \in \mathbb{R}^d$

repeat

  $t \leftarrow t + 1$

  $C_i \leftarrow \emptyset$ for all $i = 1, \cdots, k$

  `// Cluster Assignment Step`

  foreach $\mathbf{x}_j \in \mathbf{D}$ **do**

    $i^* \leftarrow \arg\min_i \left\{ \|\mathbf{x}_j - \boldsymbol{\mu}_i^{t-1}\|^2 \right\}$

    $C_{i^*} \leftarrow C_{i^*} \cup \{\mathbf{x}_j\}$ `// Assign` $\mathbf{x}_j$ `to closest centroid`

  `// Centroid Update Step`

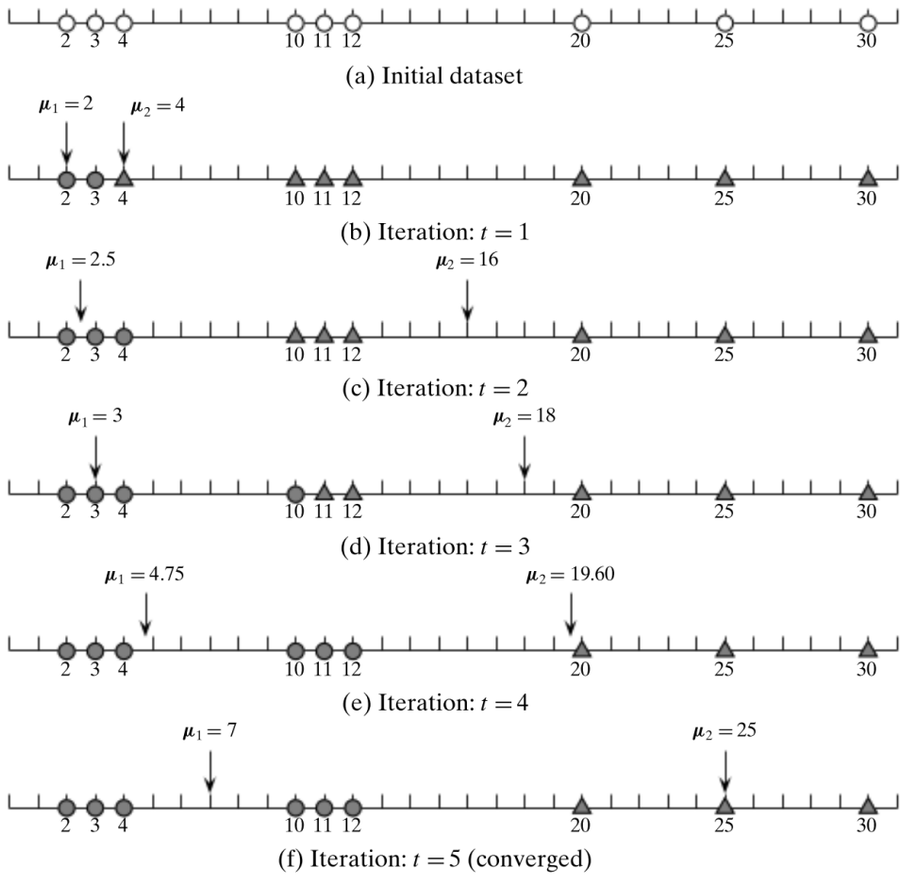
  foreach $i = 1, \cdots, k$ **do**

    $\boldsymbol{\mu}_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$

until $\sum_{i=1}^k \|\boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1}\|^2 \leq \epsilon$

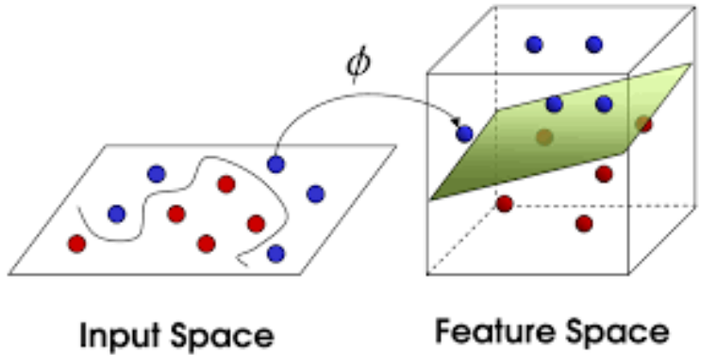
مثال: داده‌های یک بعدی زیر را در نظر بگیرید.
داده‌ها را به دو خوشه تقسیم می‌کنیم. بطور اتفاقی دو نماینده انتخاب می‌کنیم.

$$\mu = \{2,4\}$$



الگوریتم هسته‌ای میانگین-k (Kernel K-means)

برای داده‌هایی که بطور غیرخطی پراکنده شده باشند، می‌توان با انتخاب یک نگاشت مناسب، داده‌ها را به یک فضای دوگانی که در آن خطی می‌باشند انتقال و سپس در آنجا خوشه‌بندی نمود.
به این الگوریتم، هسته‌ای میانگین-k گفته می‌شود.



در فضای مشخصه‌های جدید همان الگوریتم میانگین-k اعمال می‌شود.
و برای هر خوشه میانگین محاسبه می‌شود و این میانگین نماینده‌ی خوشه خواهد بود.

$$\boldsymbol{\mu}_i^{\phi} = \frac{1}{n_i} \sum_{\mathbf{x}_j \in C_i} \phi(\mathbf{x}_j)$$

خوشه‌ها با نماینده‌ها مشخص می‌شوند.

$$\{\mu_1^{\phi}, \mu_2^{\phi}, \dots, \mu_k^{\phi}\}$$

نکته‌ی جالب اینکه در محاسبه‌ی فاصله‌ی هر دو داده در فضای مشخصه‌های جدید، نیازی به دانستن خود تابع نگاشت $\phi(\boldsymbol{x})$ نیست. کافیست تنها ضرب داخلی توابع نگاشت که هسته (Kernel) نامیده می‌شود را بدانیم.

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

الگوریتم هسته‌ای میانگین-k (Kernel K-means)

فاصله‌ی دو نقطه در فضای مشخصه‌ی جدید برحسب هسته‌ها قابل محاسبه است.

$$\begin{aligned}\|\phi(\mathbf{x}_i)-\phi(\mathbf{x}_j)\|^2 &= \|\phi(\mathbf{x}_i)\|^2+\|\phi(\mathbf{x}_j)\|^2-2\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)\\ &= K(\mathbf{x}_i,\mathbf{x}_i)+K(\mathbf{x}_j,\mathbf{x}_j)-2K(\mathbf{x}_i,\mathbf{x}_j)\end{aligned}$$

پس فاصله‌ی هر داده از میانگین را برحسب هسته‌ها (Kernels) محاسبه می‌کنیم.

$$\begin{aligned}\|\phi(\mathbf{x}_i)-\boldsymbol{\mu}_\phi\|^2 &= \|\phi(\mathbf{x}_i)\|^2-2\phi(\mathbf{x}_i)^T\boldsymbol{\mu}_\phi+\|\boldsymbol{\mu}_\phi\|^2\\ &= K(\mathbf{x}_i,\mathbf{x}_i)-\frac{2}{n}\sum_{j=1}^{n}K(\mathbf{x}_i,\mathbf{x}_j)+\frac{1}{n^2}\sum_{a=1}^{n}\sum_{b=1}^{n}K(\mathbf{x}_a,\mathbf{x}_b)\end{aligned}$$

در الگوریتم میانگین-k، در انتساب داده‌ها به هر خوشه نیازی به دانستن تابع نگشت $\phi(\boldsymbol{x})$ نیست و تنها داشتن هسته کافی است.

$$\begin{aligned}C^*(\mathbf{x}_j) &= \arg\min_i\left\{\|\phi(\mathbf{x}_j)-\boldsymbol{\mu}_i^\phi\|^2\right\}\\ &= \arg\min_i\left\{K(\mathbf{x}_j,\mathbf{x}_j)-\frac{2}{n_i}\sum_{\mathbf{x}_a\in C_i}K(\mathbf{x}_a,\mathbf{x}_j)+\frac{1}{n_i^2}\sum_{\mathbf{x}_a\in C_i}\sum_{\mathbf{x}_b\in C_i}K(\mathbf{x}_a,\mathbf{x}_b)\right\}\\ &= \arg\min_i\left\{\frac{1}{n_i^2}\sum_{\mathbf{x}_a\in C_i}\sum_{\mathbf{x}_b\in C_i}K(\mathbf{x}_a,\mathbf{x}_b)-\frac{2}{n_i}\sum_{\mathbf{x}_a\in C_i}K(\mathbf{x}_a,\mathbf{x}_j)\right\}\end{aligned}$$

الگوریتم هسته‌ای میانگین-k (Kernel K-means)

با داشتن هسته‌ها (Kernels) به روش‌های مختلف می‌توان توابع نگاشت $\phi(\boldsymbol{x})$ را بدست آورد.

۱ - روش نگاشت تجربی هسته (Empirical Kernel Map):

$$\phi(\mathbf{x}) = \Big(K(\mathbf{x}_1, \mathbf{x}), K(\mathbf{x}_2, \mathbf{x}), \ldots, K(\mathbf{x}_n, \mathbf{x})\Big)^T \in \mathbb{R}^n$$

۲ - روش نگاشت مرسر هسته (Mercer Kernel Map):
با استفاده از تجذیه ماتریس $n \times n$ هسته‌ها به ویژه مقادیر و ویژه بردارهای.

$$\boldsymbol{K} = \{K(\boldsymbol{x}_i, \boldsymbol{x}_j)\}_{i,j=1,\ldots,n} = \{\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)\}_{i,j=1,\ldots,n}$$

$$\mathbf{K} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$$

$$\phi(\mathbf{x}_i) = \sqrt{\boldsymbol{\Lambda}}\mathbf{U}_i = \Big(\sqrt{\lambda_1}\, u_{1i}, \sqrt{\lambda_2}\, u_{2i}, \ldots, \sqrt{\lambda_n}\, u_{ni}\Big)^T$$

بنا به ساختار مساله و مشخصه‌های داده‌ها، هسته‌های زیادی به دلخواه می‌توان طراحی کرد.
از مشهورترین هسته‌ها عبارتند:

هسته‌ی چندجمله‌ای همگن و ناهمگن
(Homogeneous and Inhomogeneous Polynomial Kernel)

$$K_q(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y}) = (\mathbf{x}^T \mathbf{y})^q$$

$$K_q(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y}) = (c + \mathbf{x}^T \mathbf{y})^q$$

هسته‌ی گاوسی (Gaussian Kernel)

$$K(\mathbf{x}, \mathbf{y}) = \exp\left\{-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right\}$$

Algorithm 13.2: Kernel K-means Algorithm

KERNEL-KMEANS($\mathbf{K}, k, \epsilon$):

$t \leftarrow 0$

$\mathcal{C}^t \leftarrow \{C_1^t, \ldots, C_k^t\}$ // Randomly partition points into k clusters

repeat

  $t \leftarrow t+1$

  foreach $C_i \in \mathcal{C}^{t-1}$ **do** // Compute squared norm of cluster means

    $\text{sqnorm}_i \leftarrow \frac{1}{n_i^2} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b)$

  foreach $\mathbf{x}_j \in \mathbf{D}$ **do** // Average kernel value for $\mathbf{x}_j$ and C_i

    foreach $C_i \in \mathcal{C}^{t-1}$ **do**

      $\text{avg}_{ji} \leftarrow \frac{1}{n_i} \sum_{\mathbf{x}_a \in C_i} K(\mathbf{x}_a, \mathbf{x}_j)$

  // Find closest cluster for each point

  foreach $\mathbf{x}_j \in \mathbf{D}$ **do**

    foreach $C_i \in \mathcal{C}^{t-1}$ **do**

      $d(\mathbf{x}_j, C_i) \leftarrow \text{sqnorm}_i - 2 \cdot \text{avg}_{ji}$

    $i^* \leftarrow \arg\min_i \{d(\mathbf{x}_j, C_i)\}$

    $C_{i^*}^t \leftarrow C_{i^*}^t \cup \{\mathbf{x}_j\}$ // Cluster reassignment

  $\mathcal{C}^t \leftarrow \{C_1^t, \ldots, C_k^t\}$

until $1 - \frac{1}{n} \sum_{i=1}^{k} |C_i^t \cap C_i^{t-1}| \leq \epsilon$

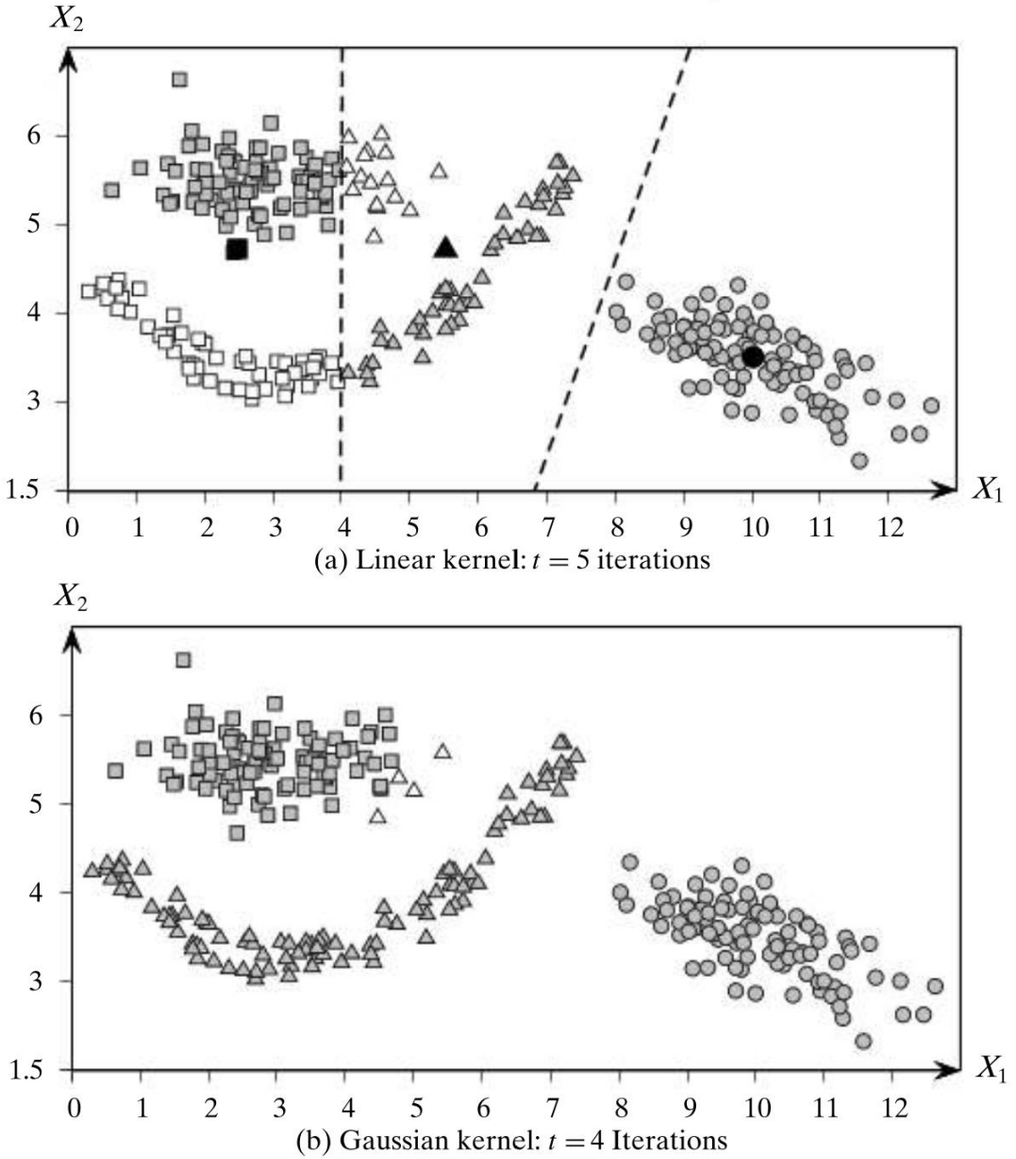


Figure 13.3. Kernel K-means: linear versus Gaussian kernel.

خوشه‌بندی از طریق بیشینه‌سازی-چشمداشتی (Expectation-Maximization Clustering)

در خوشه‌بندی میانگین-k هر نقطه به قاطعیت به یک خوشه منتسب می‌شود (Hard Assignment).
اما در خوشه‌بندی EM هر داده به احتمالی به هر خوشه نسبت داده می‌شود (Soft Assignment).

در روش EM فرض می‌کنیم احتمال مشاهده‌ی هر داده
از یک مدل مخلوط گاوسی (Gaussian Mixture Model) پیروی می‌کند.
و با احتمالی به هر خوشه نسبت داده می‌شود.

$$f(\mathbf{x}) = \sum_{i=1}^{k} f_i(\mathbf{x}) P(C_i) = \sum_{i=1}^{k} f(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) P(C_i)$$

$$f_i(\mathbf{x}) = f(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} \exp\left\{ -\frac{(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)}{2} \right\}$$

با توجه به فرض گاوسی بودن توزیع خوشه‌ها، مجموعه‌ی پارامترها برای k خوشه می‌شود:

$$\boldsymbol{\theta} = \left\{ \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, P(C_1) \ldots, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, P(C_k) \right\}$$

که در آن $\boldsymbol{\mu}_i \in \mathrm{R}^d$ و $\boldsymbol{\Sigma}_i \in R^{d \times d}$ به ترتیب بردار میانگین و ماتریس کواریانس برای توزیع گاوسی خوشه‌ی C_i می‌باشند.

و $P(C_i)$ احتمال پیشین (Prior Probability) برای مشاهده‌ی خوشه‌ی C_i می‌باشد و با این فرض که همه‌ی خوشه‌های ممکن در نظر گرفته شده‌اند و احتمال مشاهده‌ی خوشه‌ها از هم مستقل می‌باشند، احتمال مشاهده‌ی یکی از خوشه‌ها همواره وجود دارد.

$$\sum_{i=1}^{k} P(C_i) = 1$$

درستنمایی (Likelihood) پارامتر $\boldsymbol{\theta}$، احتمال مشاهده‌ی مجموعه داده‌ی D به شرط در نظر گرفتن مقدار $\boldsymbol{\theta}$ برای پارمترهای مدل می‌باشد.
از آنجاییکه توزیع هر داده با دیگری مستقل است، احتمال مشاهده‌ی مجموعه داده‌ی $\boldsymbol{D}$ برابر حاصلضرب احتمال هر یک از داده‌ها است.

$$P(\mathbf{D}|\boldsymbol{\theta}) = \prod_{j=1}^{n} f(\mathbf{x}_j)$$

برآورد درستنمایی بیشینه (Maximum Likelihood Estimation):
از آنجاییکه نظم مشاهده شده در مجموعه داده‌ها بیشترین شانس برای مشاهده شدن را داشته است که عملا مشاهده شده است، مقدار درست برای یک پارامتر، مقداری است که مقدار درستنمایی برای مجموعه داده‌ی مشاهده شده را بیشینه نماید.

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \{P(\mathbf{D}|\boldsymbol{\theta})\}$$

البته مرسوم است که برای پیدا کردن پارامتر درست، بجای درستنمایی، لگاریتم آن را بیشینه نماییم تا ضرب بین توزیع احتمال‌ها به جمع تبدیل شود که برای بیشینه کردن مناسب‌تر است.

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \{\ln P(\mathbf{D}|\boldsymbol{\theta})\}$$

خوشه‌بندی از طریق بیشینه‌سازی-چشمداشتی (Expectation-Maximization Clustering)

در مساله‌ی خوشه‌بندی داده‌ها، حتی با استفاده از لگاریتم درستنمایی، بیشینه کردن تابع درستنمایی بطور مستقیم برای محاسبه‌ی پارامترهای بهینه همچنان کار ساده‌ای نیست.

$$\ln P(\mathbf{D}|\boldsymbol{\theta}) = \sum_{j=1}^{n} \ln f(\mathbf{x}_j) = \sum_{j=1}^{n} \ln \left(\sum_{i=1}^{k} f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) P(C_i) \right)$$

$$f_i(\mathbf{x}) = f(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} \exp\left\{ -\frac{(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)}{2} \right\}$$

لذا از روش بیشینه‌سازی-چشمداشتی (Expectation-Maximization) استفاده می‌کنیم.
در این روش، سهم هر داده‌ی $\boldsymbol{x}_j$ به خوشه C_i را احتمال شرطی $P(C_i|\boldsymbol{x}_j)$ که می‌تواند احتمال پسین (Posterior Probability) در قضیه بیز باشد در نظر می‌گیریم.

$$P(C_i|\mathbf{x}_j) = \frac{P(C_i \text{ and } \mathbf{x}_j)}{P(\mathbf{x}_j)} = \frac{P(\mathbf{x}_j|C_i) P(C_i)}{\sum_{a=1}^{k} P(\mathbf{x}_j|C_a) P(C_a)}$$

$$P(C_i|\mathbf{x}_j) = \frac{f_i(\mathbf{x}_j) \cdot P(C_i)}{\sum_{a=1}^{k} f_a(\mathbf{x}_j) \cdot P(C_a)}$$

سهم هر داده به هر خوشه را می‌توانیم بصورت وزن $w_{ij} = P(C_i \mid \boldsymbol{x}_j)$ در محاسبه‌ی پارامترهای هر خوشه $\boldsymbol{\theta}$ اعمال کنیم.

$$\boldsymbol{\theta} = \{\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, P(C_1) \ldots, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, P(C_k)\}$$

$$P(C_i) = \frac{\sum_{j=1}^{n} w_{ij}}{\sum_{a=1}^{k}\sum_{j=1}^{n} w_{aj}} = \frac{\sum_{j=1}^{n} w_{ij}}{\sum_{j=1}^{n} 1} = \frac{\sum_{j=1}^{n} w_{ij}}{n}$$

$$\sum_{i=1}^{k} w_{ij} = \sum_{i=1}^{k} P(C_i \mid x_j) = 1$$

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^{n} w_{ij} \cdot \mathbf{x}_j}{\sum_{j=1}^{n} w_{ij}}$$

کواریانس بین مشخصه‌های X_a و X_b

$$\sigma_{ab}^{i} = \frac{\sum_{j=1}^{n} w_{ij}(x_{ja} - \mu_{ia})(x_{jb} - \mu_{ib})}{\sum_{j=1}^{n} w_{ij}}$$

Algorithm 13.3: Expectation-Maximization (EM) Algorithm

EXPECTATION-MAXIMIZATION (D, k, ϵ):

1 $t \leftarrow 0$

// Initialization

2 Randomly initialize $\boldsymbol{\mu}_1^t, \ldots, \boldsymbol{\mu}_k^t$

3 $\boldsymbol{\Sigma}_i^t \leftarrow \mathbf{I}, \ \forall i = 1, \ldots, k$

4 $P^t(C_i) \leftarrow \frac{1}{k}, \ \forall i = 1, \ldots, k$

5 **repeat**

6 $\quad t \leftarrow t + 1$

$\quad$ // Expectation Step

7 $\quad$ **for** $i = 1, \ldots, k$ *and* $j = 1, \ldots, n$ **do**

8 $\quad\quad w_{ij} \leftarrow \frac{f(\mathbf{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \cdot P(C_i)}{\sum_{a=1}^{k} f(\mathbf{x}_j | \boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a) \cdot P(C_a)}$ // posterior probability $P^t(C_i | \mathbf{x}_j)$

$\quad$ // Maximization Step

9 $\quad$ **for** $i = 1, \ldots, k$ **do**

10 $\quad\quad \boldsymbol{\mu}_i^t \leftarrow \frac{\sum_{j=1}^{n} w_{ij} \cdot \mathbf{x}_j}{\sum_{j=1}^{n} w_{ij}}$ // re-estimate mean

11 $\quad\quad \boldsymbol{\Sigma}_i^t \leftarrow \frac{\sum_{j=1}^{n} w_{ij} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^{n} w_{ij}}$ // re-estimate covariance matrix

12 $\quad\quad P^t(C_i) \leftarrow \frac{\sum_{j=1}^{n} w_{ij}}{n}$ // re-estimate priors

13 **until** $\sum_{i=1}^{k} \| \boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1} \|^2 \leq \epsilon$

Example 13.4 (EM in 1D). Figure 13.4 illustrates the EM algorithm on the one-dimensional dataset:

$x_1 = 1.0$ $x_2 = 1.3$ $x_3 = 2.2$ $x_4 = 2.6$ $x_5 = 2.8$

$x_6 = 5.0$ $x_7 = 7.3$ $x_8 = 7.4$ $x_9 = 7.5$ $x_{10} = 7.7$ $x_{11} = 7.9$

We assume that $k = 2$. The initial random means are shown in Figure 13.4(a), with the initial parameters given as

$\mu_1 = 6.63$ $\sigma_1^2 = 1$ $P(C_2) = 0.5$

$\mu_2 = 7.57$ $\sigma_2^2 = 1$ $P(C_2) = 0.5$