Lecture 06: Clustering and Segmentation I

[AIX7021] Computer Vision

Seongsik Park (s.park@dgu.edu)

AI Department, Dongguk University

Tentative schedule

week	topic	date
1	Introduction and Basics	09.01
2	Image process I	09.08
3	Image process II	09.15
4	(휴강)	09.22
5	Feature detection and matching I	09.29
6	Feature detection and matching II	10.06
7	Clustering and segmentation I & II	10.13 & 10.16
8	**Mid-term exam**	**10.20**
9	Robust Fitting and Matching	10.27
10	Boosting and Face Detection	11.03
12	Dimensional Reduction and Face Recognition	11.10
13	Object recognition	11.17
14	Motion and Tracking	11.24
11	Image Classification	12.01
15	**Final exam**	**12.08**

Clustering

Clustering algorithms

- Hierarchical methods
  - Agglomerative clustering
  - Divisive clustering
- Iterative methods
  - K-means clustering
  - Mixture of Gaussian
  - Mean-shift algorithm
- Spectral clustering
  - Normalized cut
- Graph-cut

Data similarity

The cluster analysis is based on the similarity (or distance) between data

- Euclidean distance (2-norm)

$$d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2 = \sqrt{\sum_i (u_i - v_i)^2} \tag{1}$$

- Manhattan distance (1-norm)

$$d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_1 = \sum_i |u_i - v_i| \tag{2}$$

- Maximum distance (infinity-norm)

$$d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_\infty = \max_i |u_i - v_i| \tag{3}$$

- Mahalanobis distance

$$d(\mathbf{u}, \mathbf{v}) = \sqrt{(\mathbf{u} - \mathbf{v})^T \Sigma^{-1} (\mathbf{u} - \mathbf{v})} \tag{4}$$

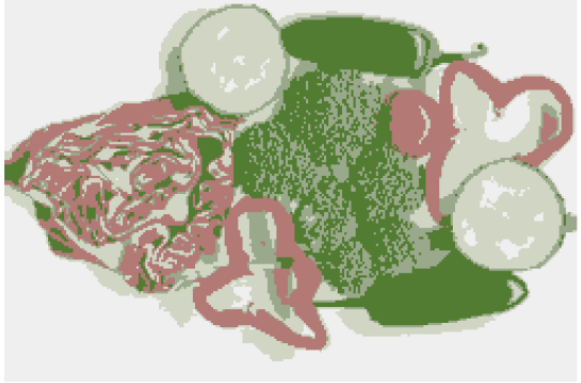
Clustering in computer vision

- Image segmentation
- Foreground/background segmentation
- Feature clustering
- Image/video categorization

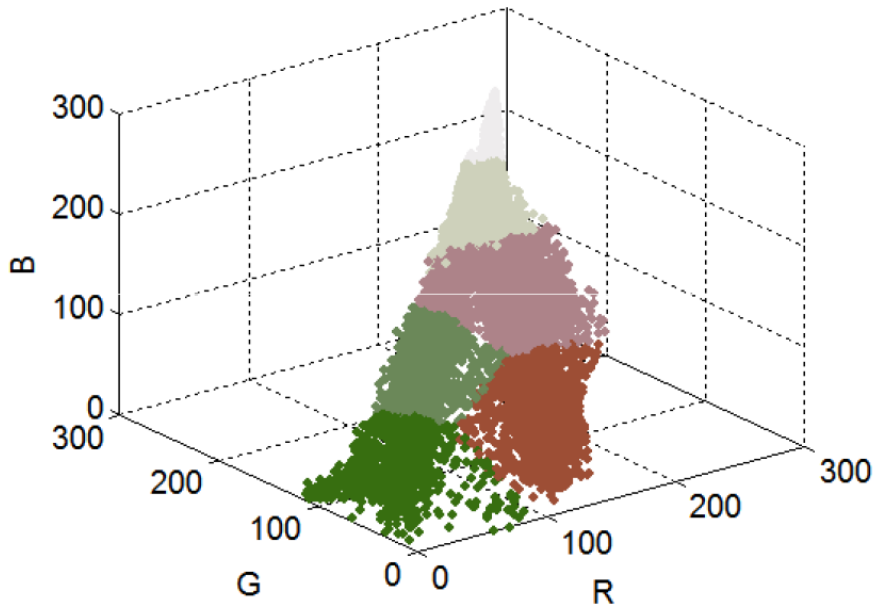
Feature for image segmentation

Image

Clustering with color



Clustering in RGB color space



The proximity in color space only is not reasonable, and we need to consider spatial information, too.

Feature for image segmentation

Image

Clustering in color & location space

Data:

$$(r_i, g_i, b_i, x_i, y_i) \quad i = 1, \ldots, N$$

Clustering with color & space

Agglomerative clustering

Dendrogram by agglomerative clustering

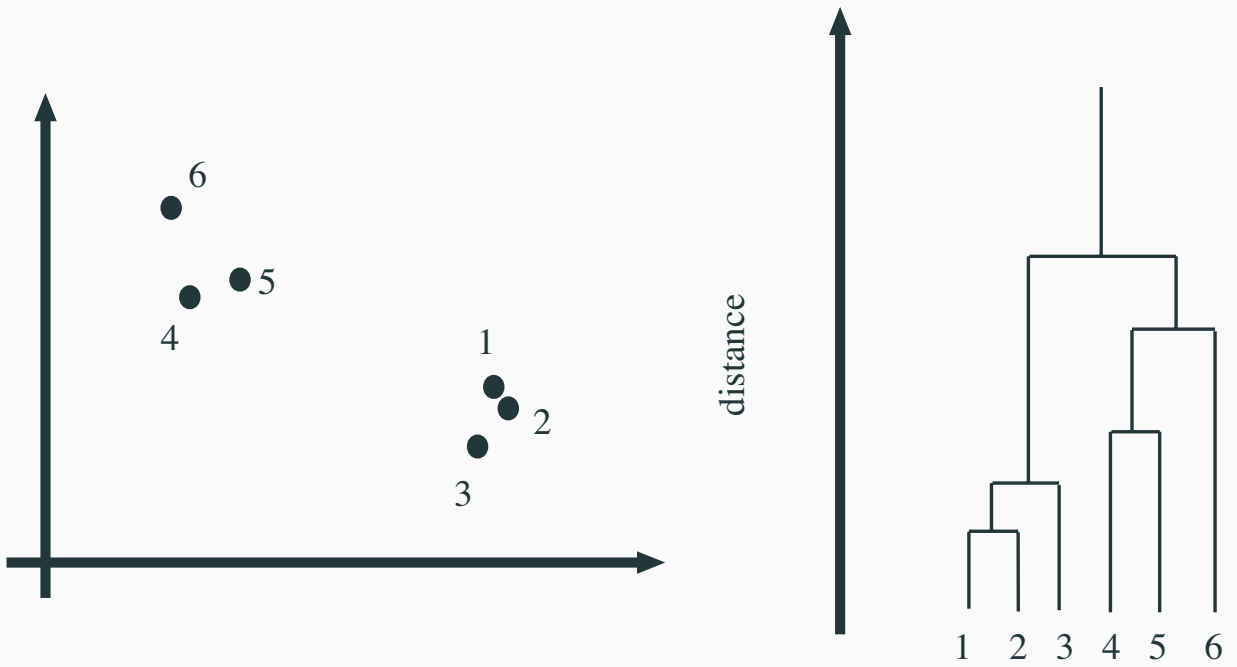


FIGURE 9.15: **Left**, a data set; **right**, a dendrogram obtained by agglomerative clustering using single-link clustering. If one selects a particular value of distance, then a horizontal line at that distance splits the dendrogram into clusters. This representation makes it possible to guess how many clusters there are and to get some insight into how good the clusters are.

K-means clustering

K-means clustering

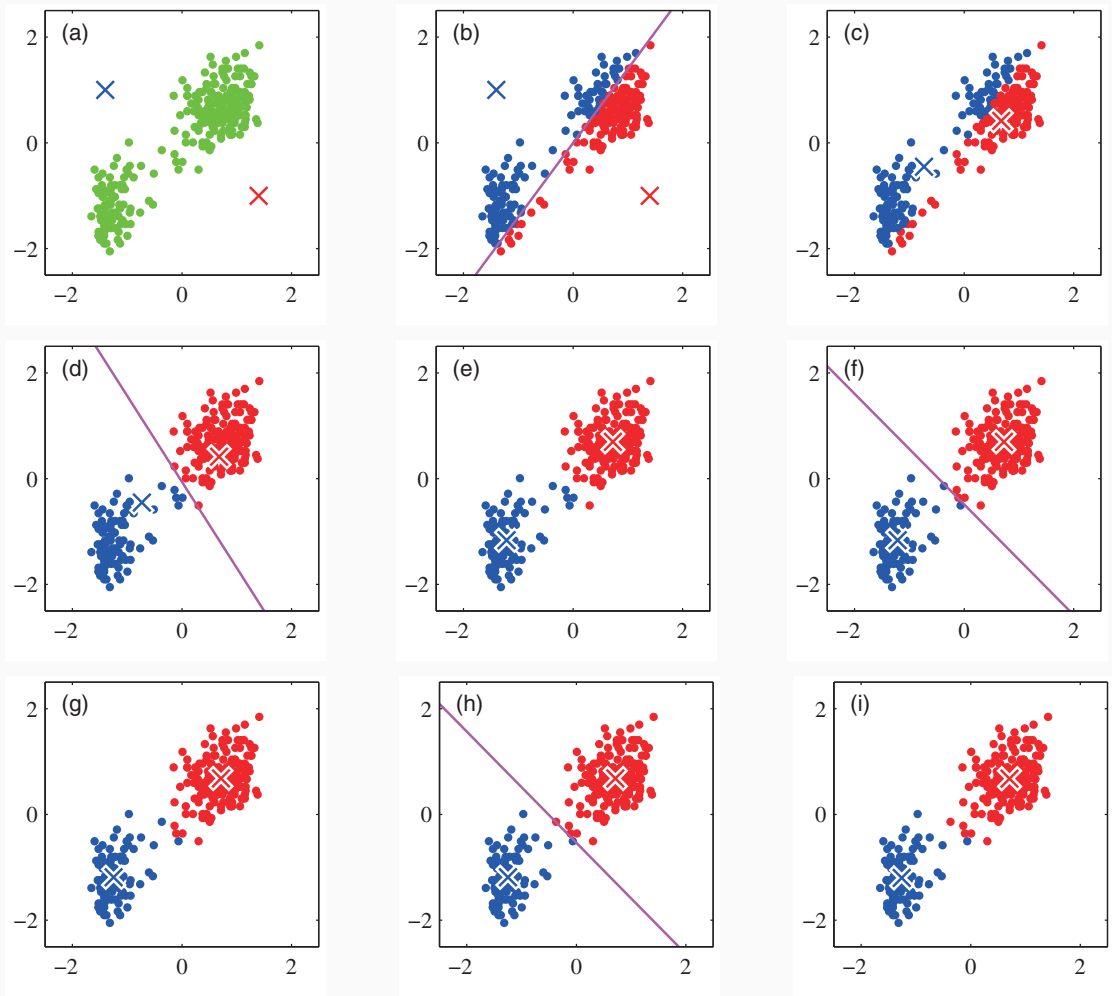


Figure 9.1 Illustration of the K-means algorithm using the re-scaled Old Faithful data set. (a) Green points denote the data set in a two-dimensional Euclidean space. The initial choices for centres $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are shown by the red and blue crosses, respectively. (b) In the initial E step, each data point is assigned either to the red cluster or to the blue cluster, according to which cluster centre is nearer. This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centres, shown by the magenta line, they lie on. (c) In the subsequent M step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster. (d)–(i) show successive E and M steps through to final convergence of

K-means: method

We begin by considering the problem of identifying groups, or clusters, of data points in a multidimensional space. Suppose we have a data set $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ consisting of N observations of a random D-dimensional Euclidean variable $\mathbf{x}$. Our goal is to partition the data set into some number K of clusters, where we shall suppose for the moment that the value of K is given.

Intuitively, we might think of a cluster as comprising a group of data points whose inter-point distances are small compared with the distances to points outside of the cluster. We can formalize this notion by first introducing a set of D-dimensional vectors $\boldsymbol{\mu}_k$, where $k = 1, ..., K$, in which $\boldsymbol{\mu}_k$ is a prototype associated with the k-th cluster.

K-means: method

As we shall see shortly, we can think of the $\boldsymbol{\mu}_k$ as representing the centres of the clusters. Our goal is then to find an assignment of data points to clusters, as well as a set of vectors $\{\boldsymbol{\mu}_k\}$, such that the sum of the squares of the distances of each data point to its closest vector $\boldsymbol{\mu}_k$, is a minimum.

It is convenient at this point to define some notation to describe the assignment of data points to clusters. For each data point $\mathbf{x}_n$, we introduce a corresponding set of binary indicator variables $r_{nk} \in \{0, 1\}$, where $k = 1, ..., K$ describing which of the K clusters the data point $\mathbf{x}_n$ is assigned to, so that if data point $\mathbf{x}_n$ is assigned to cluster k then $r_{nk} = 1$, and $r_{nj} = 0$ for $j \neq k$. This is known as **the 1-of-K coding scheme**.

K-means: method

We can then define an objective function, sometimes called a *distortion measure*, given by

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 \tag{5}$$

which represents the sum of the squares of the distances of each data point to its assigned vector $\boldsymbol{\mu}_k$.

Our goal is to find values for the $\{r_{nk}\}$ and the $\{\boldsymbol{\mu}_k\}$ so as to minimizes J.

K-means: method

The terms involving different n are independent and so we can optimize for each n separately by choosing r_{nk} to be 1 for whichever value of k gives the minimum value of $\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$. In other words, we simply assign the n-th data point to the closest cluster centre.

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

K-means: method

Now consider the optimization of the $\boldsymbol{\mu}_k$ with the r_{nk} held fixed. The objective function J is a quadratic function of $\boldsymbol{\mu}_k$, and it can be minimized by setting its derivative with respect to $\boldsymbol{\mu}_k$ to zero giving

$$2\sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k) = 0 \tag{7}$$

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{r_{nk}} \tag{8}$$

The denominator in this expression is equal to the number of points assigned to cluster k, and so this result has a simple interpretation, namely set $\boldsymbol{\mu}_k$ equal to the mean of all of the data points $\mathbf{x}_n$ assigned to cluster k.

The two phases of re-assigning data points to clusters and re-computing the cluster means are repeated in turn until there is no further change in the assignments (or until some maximum number of iterations is exceeded).

K-means: python example

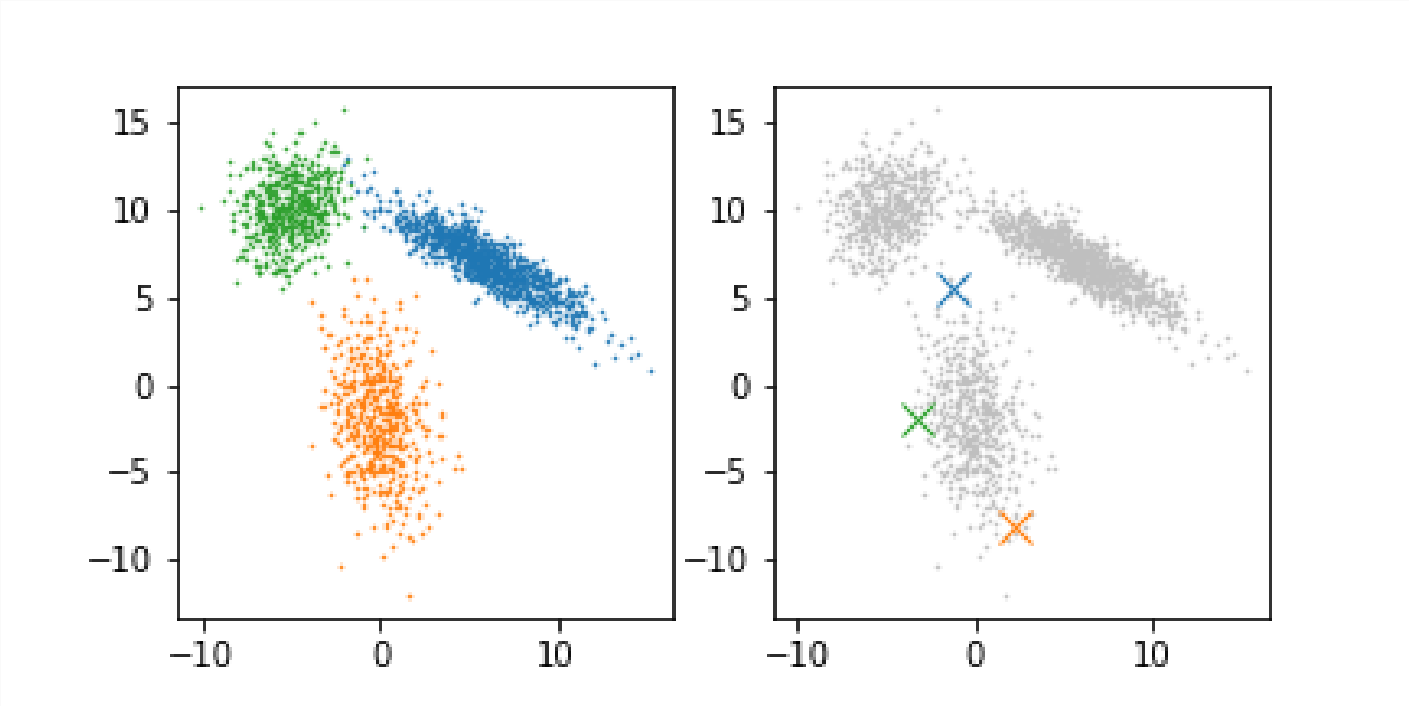


Figure 1: K-means algorithm python example: dataset (left) and the initialization (right).

K-means: python example

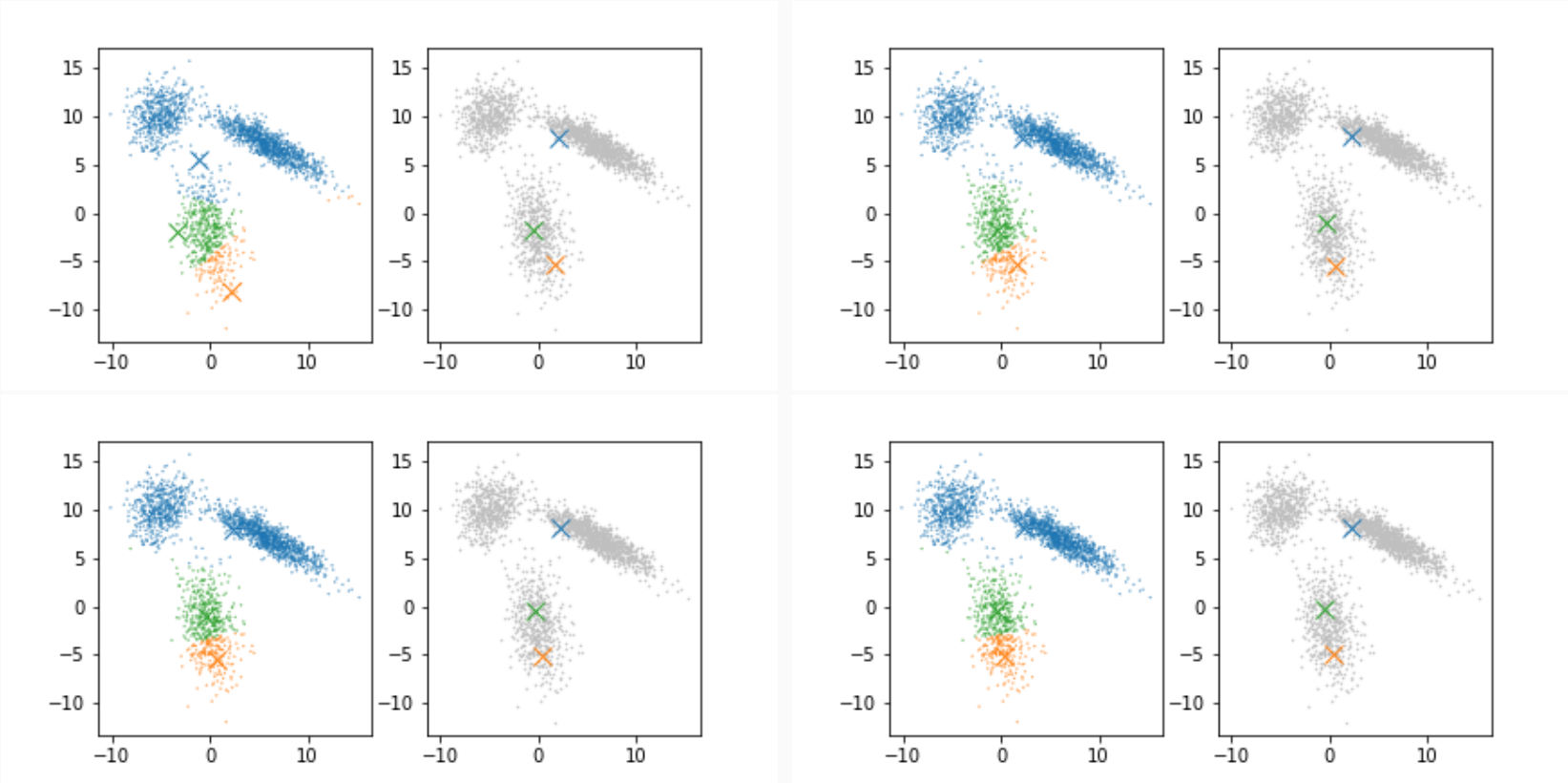


Figure 2: From the 1st to 4th iterations.

K-means: python example

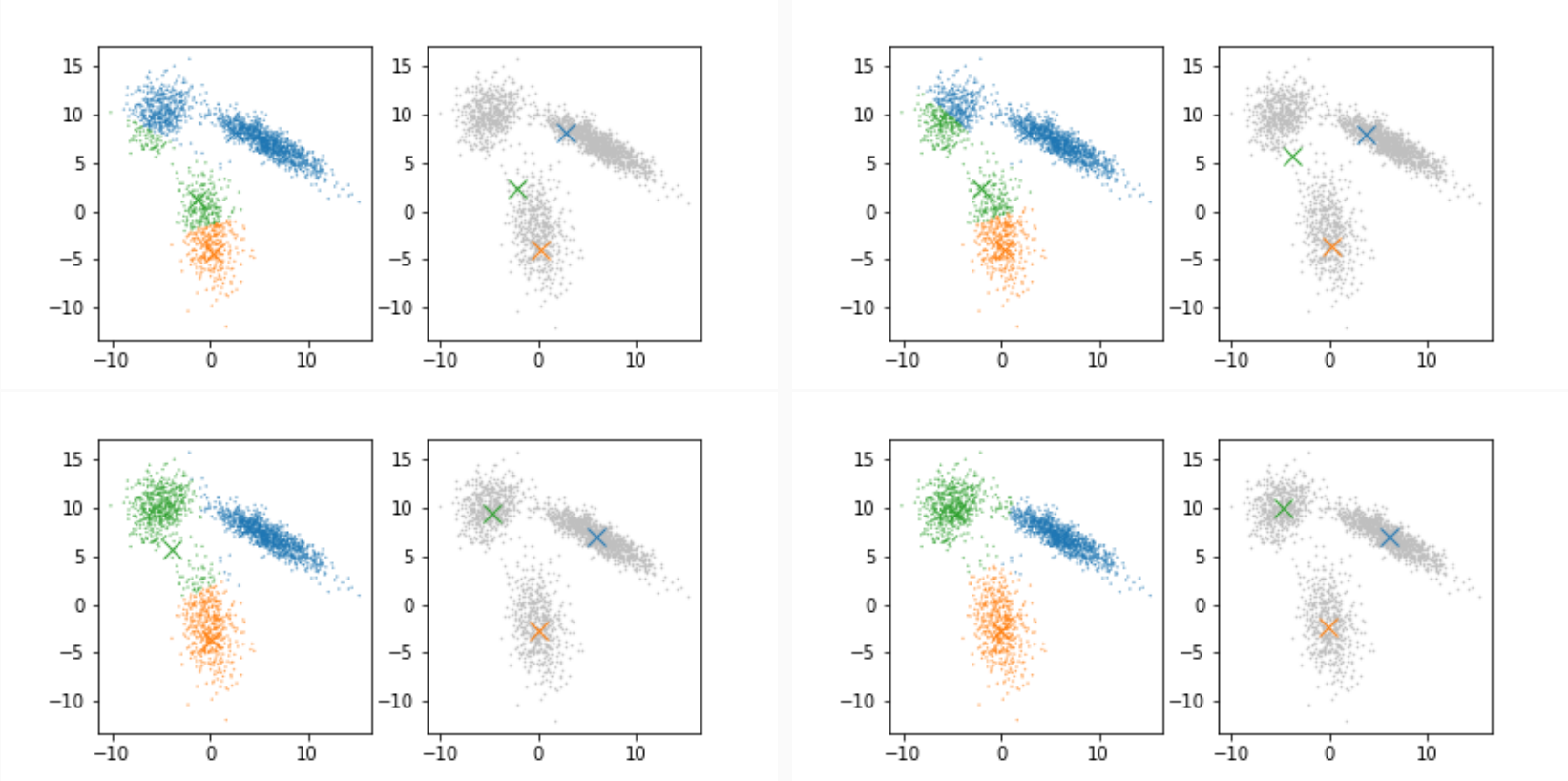


Figure 3: From the 9th to 12th iterations.

K-means: example

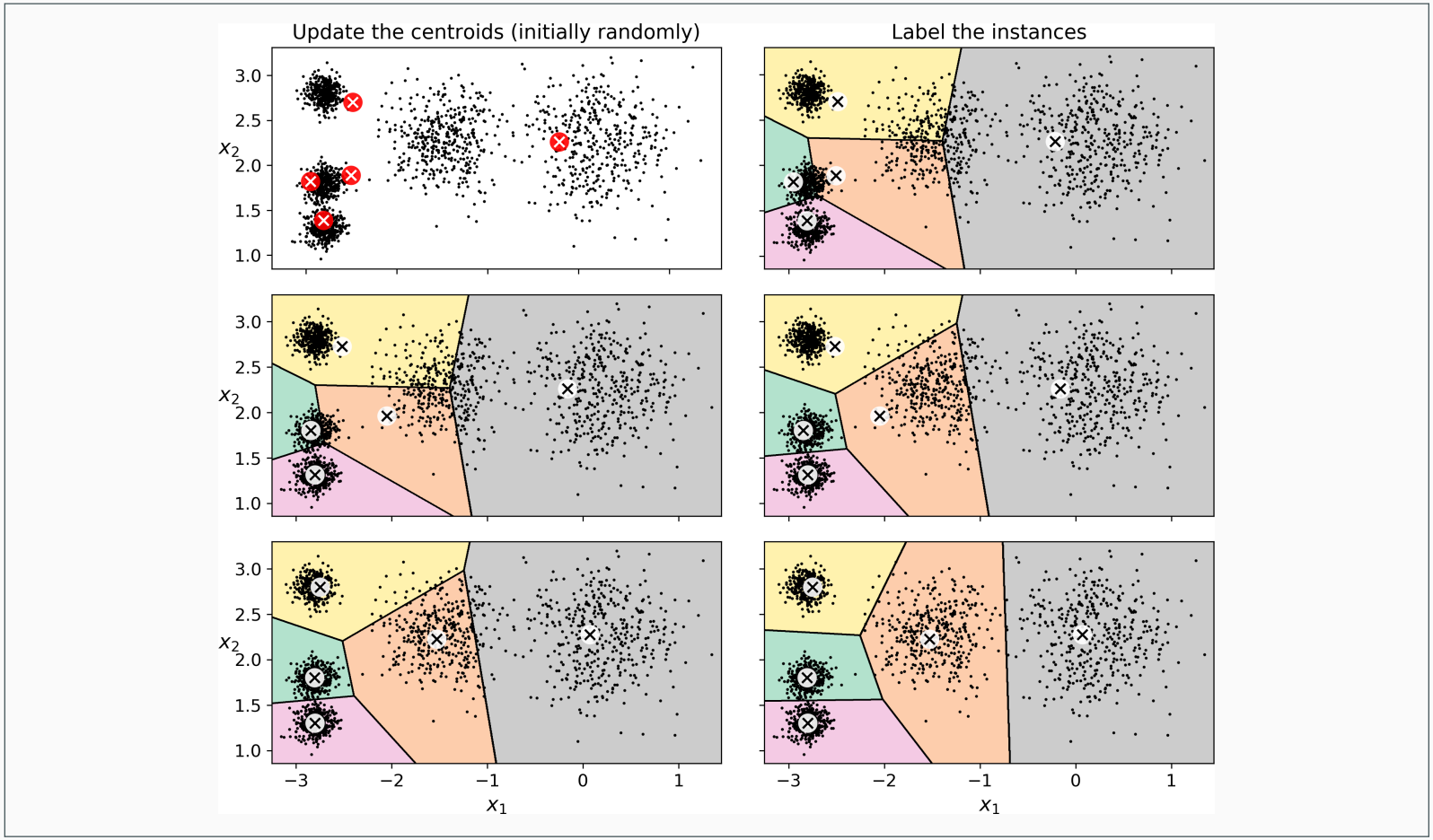


Figure 9-4. The K-Means algorithm

K-means: decision boundary

Decision boundary = Voronoi tessellation

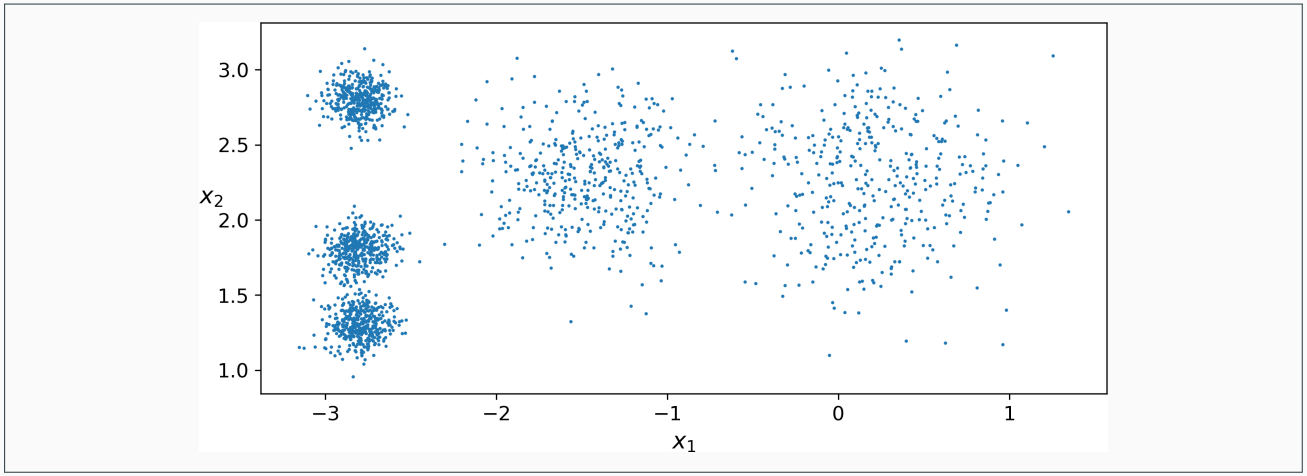


Figure 9-2. An unlabeled dataset composed of five blobs of instances

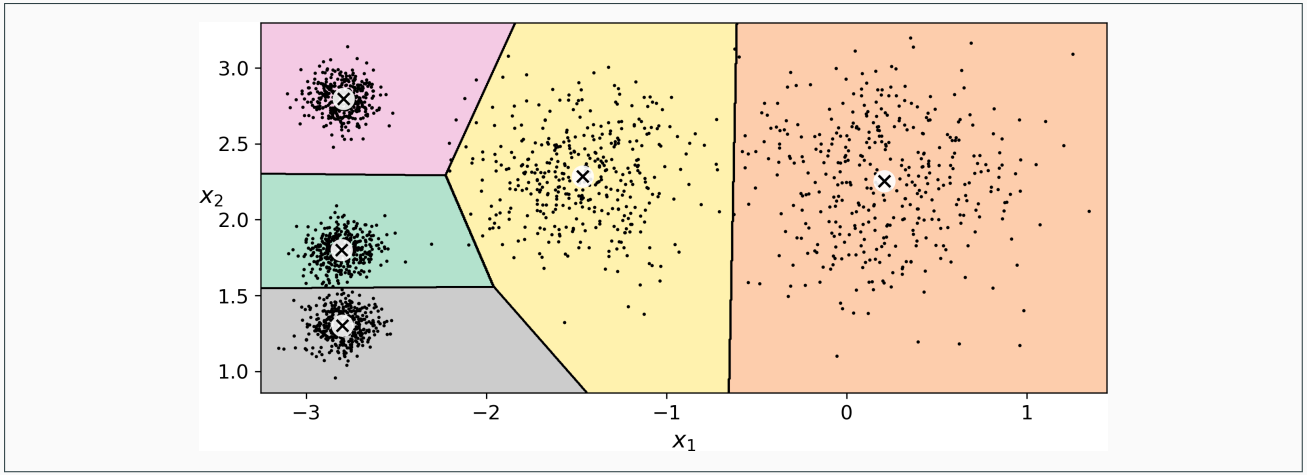


Figure 9-3. K-Means decision boundaries (Voronoi tessellation)

Mixtures of Gaussians

MoG: 결론부터

1. **Initialize** the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k.
2. **E-step** Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{9}$$

3. **M-step** Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{10}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T \tag{11}$$

$$\pi_k = \frac{N_k}{N} \tag{12}$$

4. Evaluate the log likelihood

$$\log p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \log \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \tag{13}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied, return to step 2.

MoG: model

Mixture of Gaussian distributions can be written as a linear superposition of Gaussians.

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{14}$$

Let us introduce K-dimensional binary random variable $\mathbf{z}$ having a 1-of-K representation in which a particular element z_k is equal to 1 and all other elements are equal to 0. The values of z_k therefore satisfy $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$, and we see that there are K possible states for the vector $\mathbf{z}$ according to which element is nonzero.

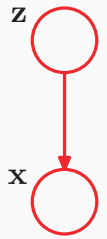
The marginal distribution over $\mathbf{z}$ is specified in terms of the mixing coefficients π_k, such that

$$p(z_k = 1) = \pi_k \tag{15}$$

where $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$.

MoG: 1-of-K representation

Figure 9.4 Graphical representation of a mixture model, in which the joint distribution is expressed in the form $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$.



Because $\mathbf{z}_k$ uses a 1-of-K representation, we can also write this distribution in the form

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k} \tag{16}$$

Similarly, the conditional distribution of $\mathbf{x}$ given a particular value for $\mathbf{z}$ is a Gaussian

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{17}$$

which can also be written in the form

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k} \tag{18}$$

MoG: model

The joint distribtion is given by $p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, and the marginal distribution of $\mathbf{x}$ is then obtained by summing the joint distribution over all possible states of $\mathbf{z}$ to give

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{19}$$

Another quantity that will play an important role is the condition probability of $\mathbf{z}$ given $\mathbf{x}$. We shall use $\gamma(z_k)$ to denote $p(z_k = 1|\mathbf{x})$, whose value can be found using Bayes' theorem

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(\mathbf{x}|z_k = 1)p(z_k = 1)}{\sum_{j=1}^{K} p(\mathbf{x}|z_j = 1)p(z_j = 1)} \tag{20}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{21}$$

MoG: model

We shall view π_k as the prior probability of $z_k = 1$, and the quantity $\gamma(z_k)$ as the corresponding posterior probability once we have observed $\mathbf{x}$. As we shall see later, $\gamma(z_k)$ can also be viewed as the *responsibility* that component k takes for 'explaining' the observation $\mathbf{x}$.

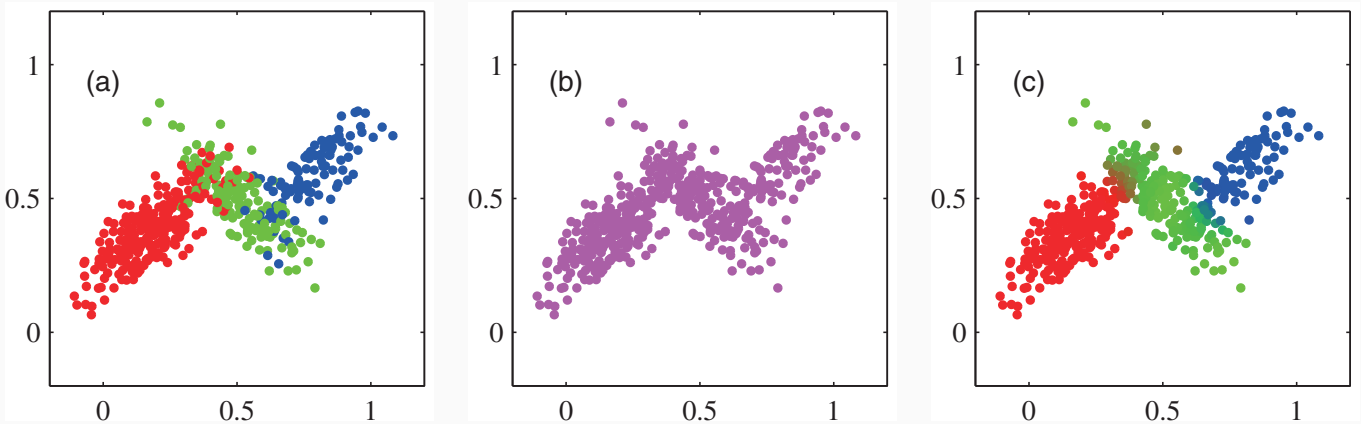


Figure 9.5 Example of 500 points drawn from the mixture of 3 Gaussians shown in Figure 2.23. (a) Samples from the joint distribution $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ in which the three states of $\mathbf{z}$, corresponding to the three components of the mixture, are depicted in red, green, and blue, and (b) the corresponding samples from the marginal distribution $p(\mathbf{x})$, which is obtained by simply ignoring the values of $\mathbf{z}$ and just plotting the $\mathbf{x}$ values. The data set in (a) is said to be *complete*, whereas that in (b) is *incomplete*. (c) The same samples in which the colours represent the value of the responsibilities $\gamma(z_{nk})$ associated with data point $\mathbf{x}_n$, obtained by plotting the corresponding point using proportions of red, blue, and green ink given by $\gamma(z_{nk})$ for $k = 1, 2, 3$, respectively

MoG: example

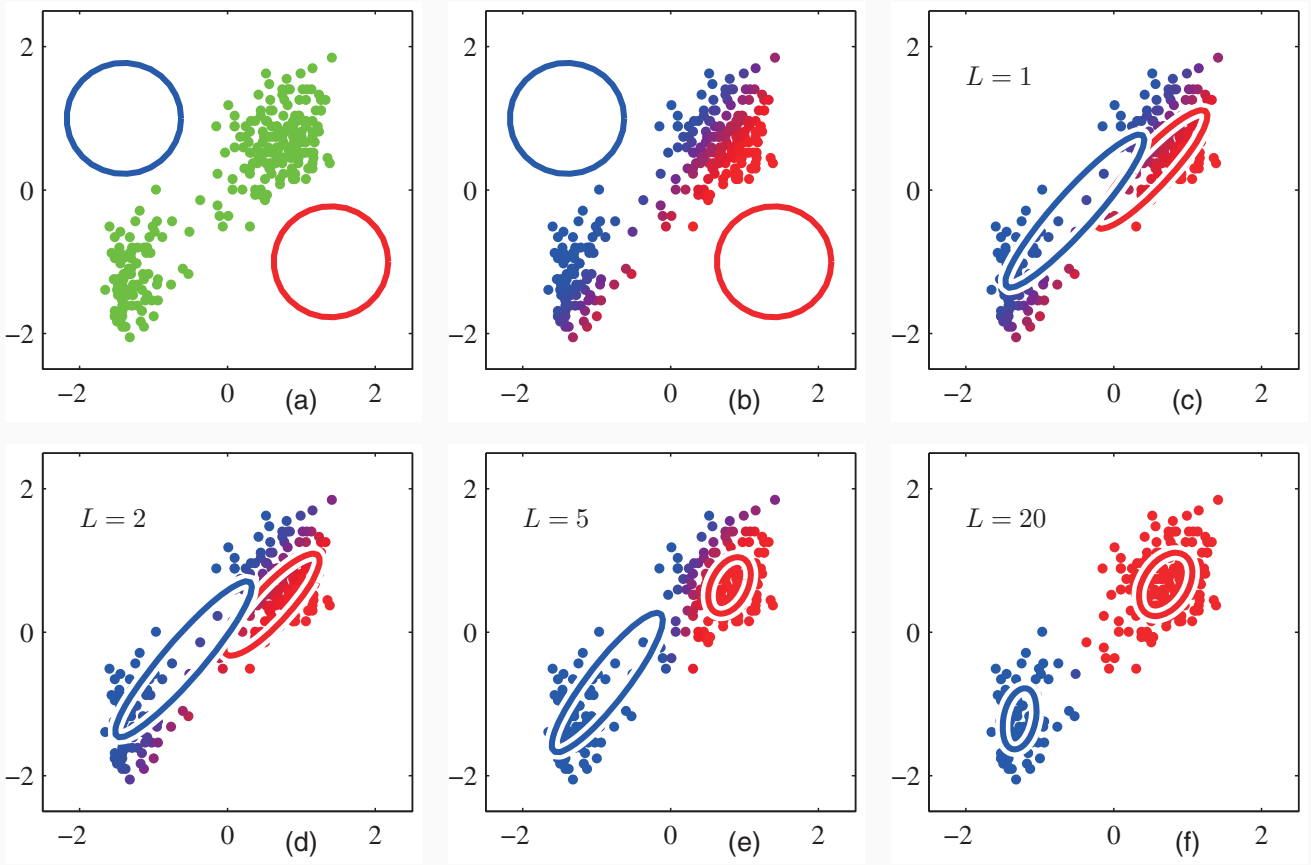


Figure 9.8 Illustration of the EM algorithm using the Old Faithful set as used for the illustration of the K-means algorithm in Figure 9.1. See the text for details.

MoG: python example

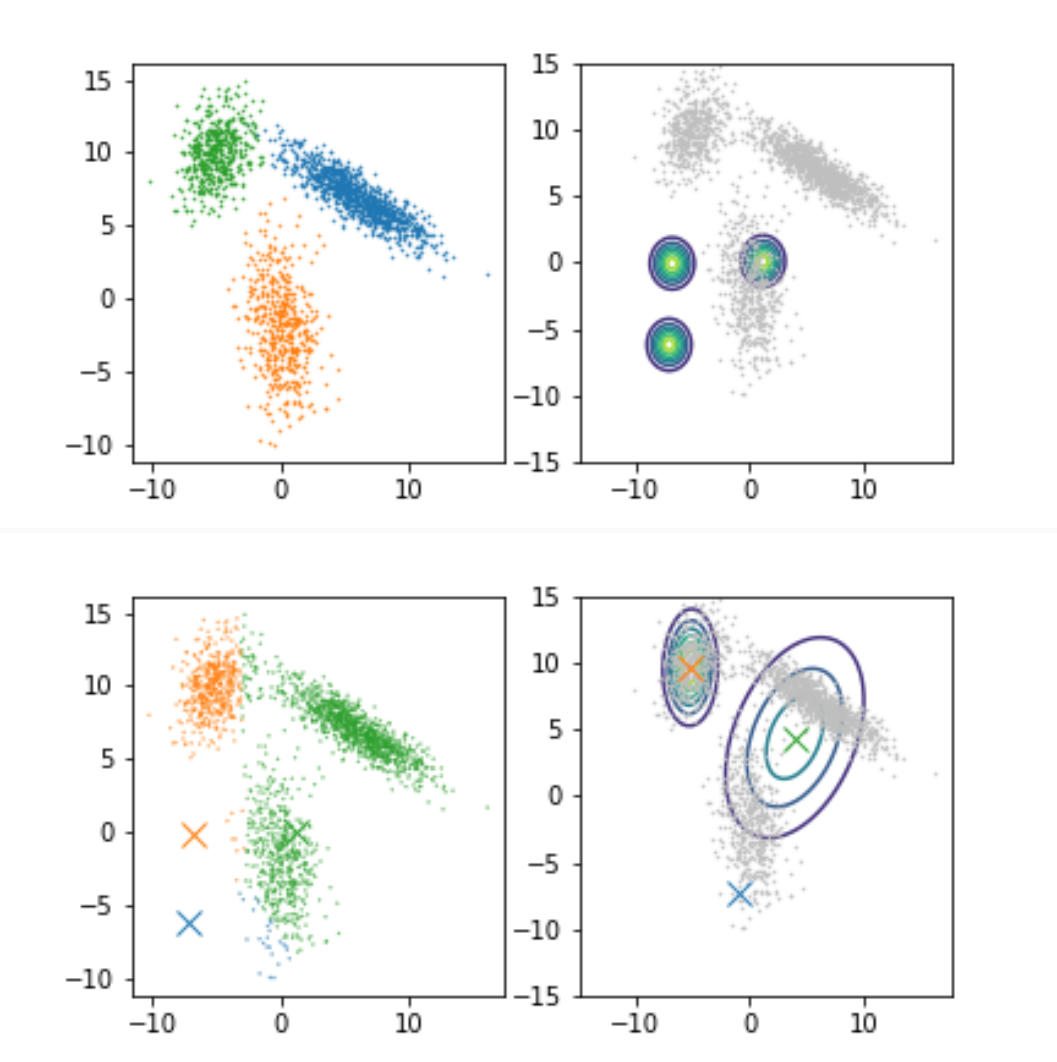


Figure 4: MoG python example: dataset and initialization (top) and the 1st iteration (bottom).

MoG: python example

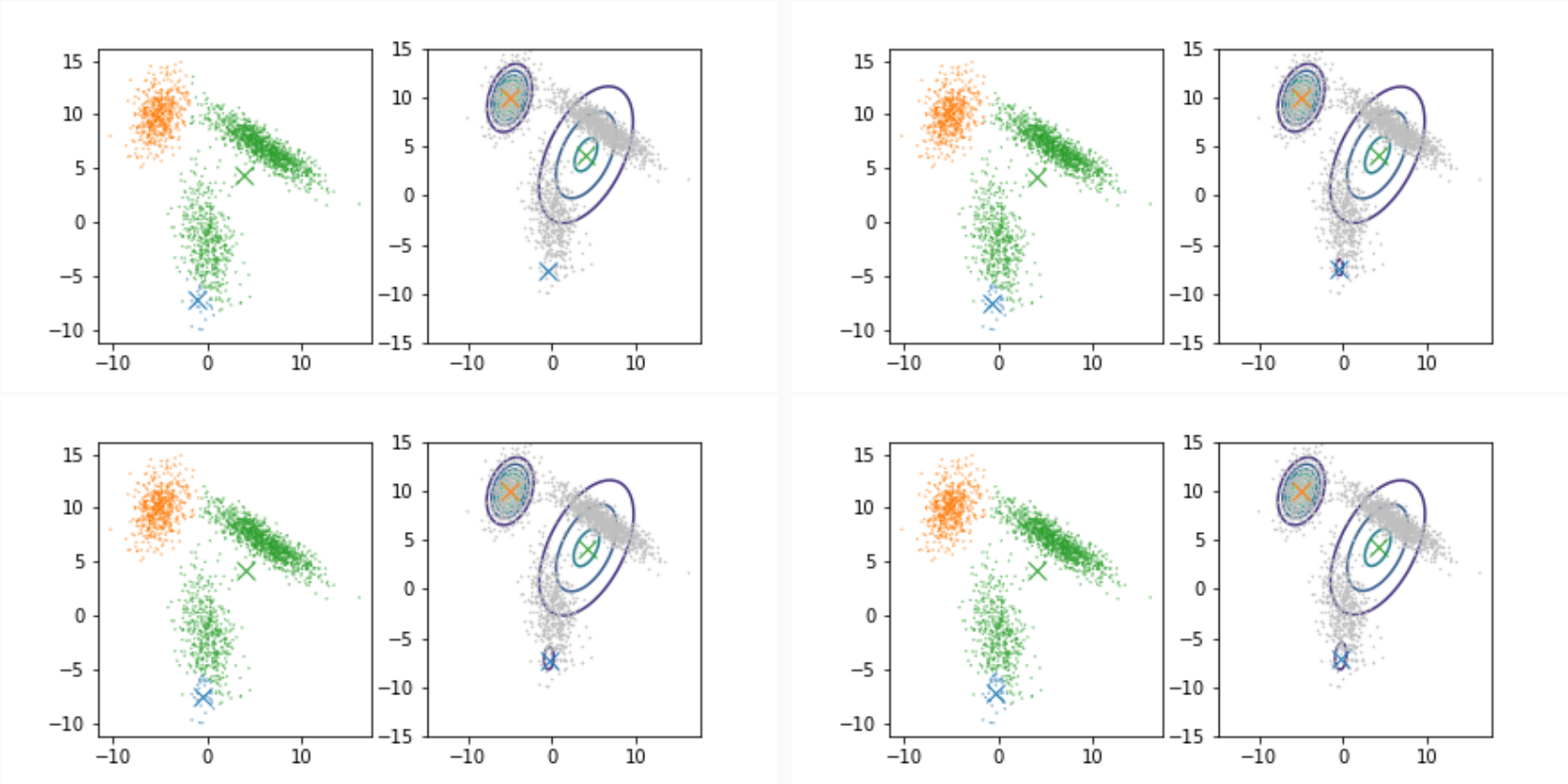


Figure 5: From the 2nd to 5th iterations.

MoG: python example

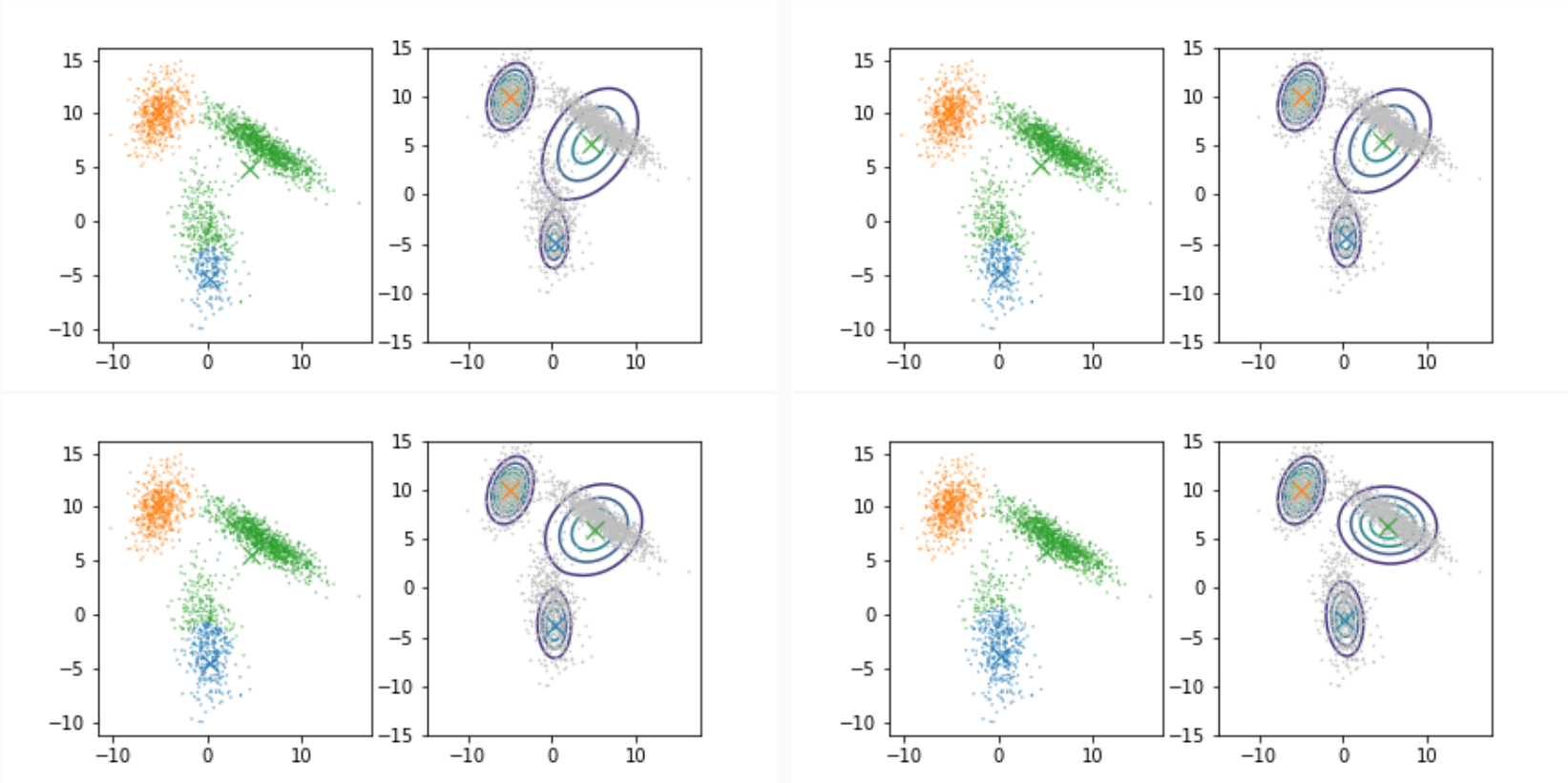


Figure 6: From the 12th to 15th iterations.

MoG: python example

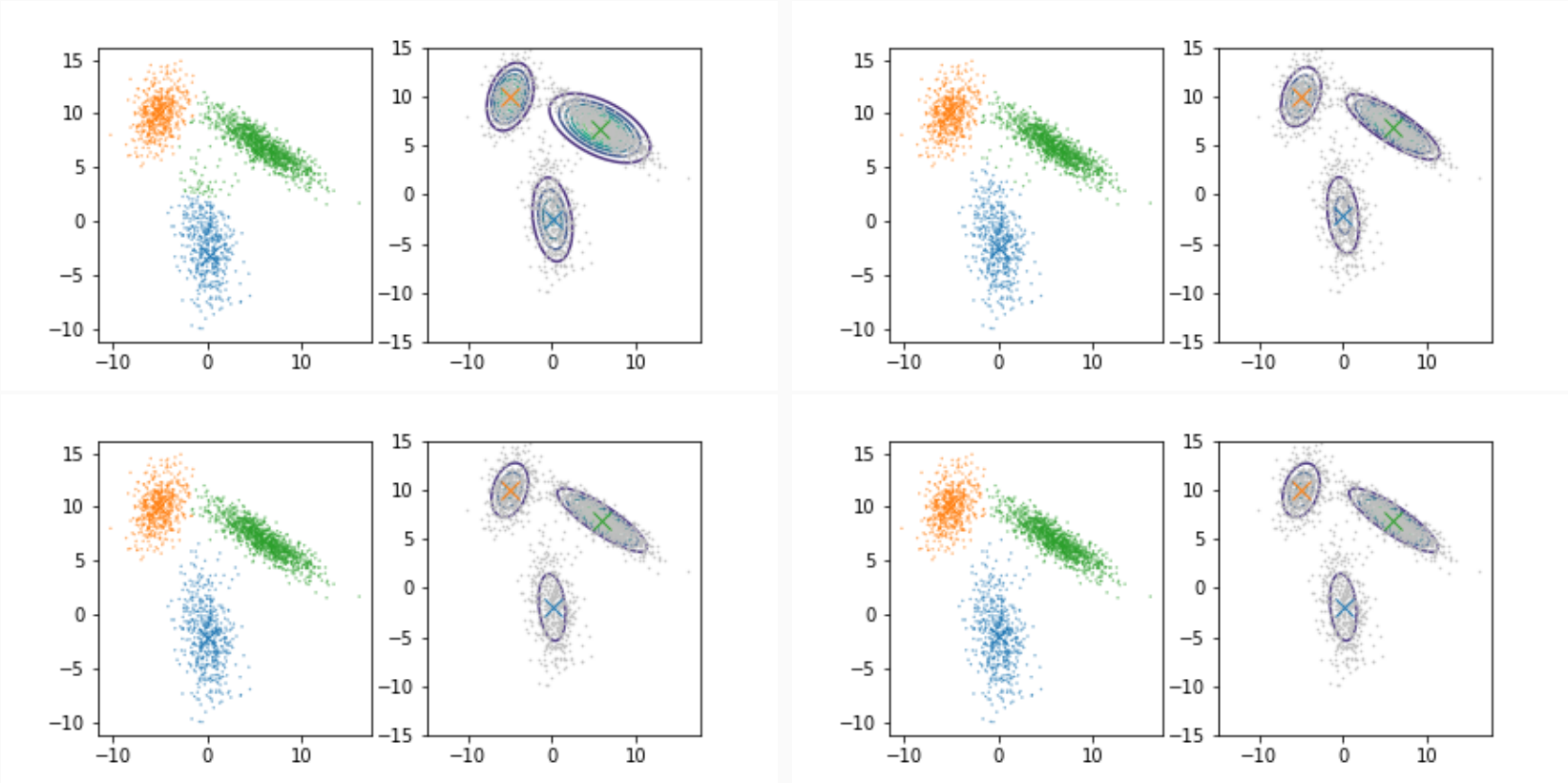


Figure 7: From the 16th to 19th iterations.

Mixtures of Gaussians: EM algorithm

MoG: maximum likelihood estimate

Maximum likelihood estimation, or MLE, is on flavor of parameter estimation in machine learning. In order to perform parameter estimation, we need:

- some data $\mathbf{x}$
- some hypothesized generating function of the data $f(\mathbf{x}, \theta)$
- a set of parameters from that function θ
- some evaluation of the goodness of our parameters (an objective function)

In MLE, the objective function (evaluation) we chose is the likelihood of the data given our model. To find the best θ then, we need to find the θ which maximizes our evaluation function (the likelihood). Therefore, in its general form the MLE is:

$$\hat{\theta}_{\mathrm{MLE}} = \arg\max_{\theta} p(\mathbf{x}|\theta) \tag{22}$$

MoG: maximum likelihood estimate

Gaussian distribution을 따르는 i.i.d. 샘플 $\mathbf{x} = (x_1, x_2, ..., x_N)$로부터 평균 $\theta = \mu$를 MLE로 추정하면,

$$\mathcal{L} = p(\mathbf{x}|\theta) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu) \tag{23}$$

$$\log \mathcal{L} = \sum_{n=1}^{N} \log \mathcal{N}(x_n|\mu) \tag{24}$$

$$\frac{d}{d\mu} \log \mathcal{L} = -const \cdot \sum_{n=1}^{N} (x_n - \mu) \tag{25}$$

$$\frac{d}{d\mu} \log \mathcal{L} = 0 \quad \Longleftrightarrow \quad \hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{26}$$

MoG: EM algorithm

An elegant and powerful method for finding maximum likelihood solutions for models with latent variables is called the *expectation-maximization* algorithm, or EM algorithm.

Let us begin by writing down the conditions that must be satisfied at a maximum of the likelihood function.

$$\log p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \log \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \tag{27}$$

Setting the derivatives of log likelihood with respect to the means $\boldsymbol{\mu}_k$ of the Gaussian components to zero,

$$0 = -\sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \boldsymbol{\Sigma}_k(\mathbf{x}_n - \boldsymbol{\mu}_k) \tag{28}$$

MoG: EM algorithm

Multiplying by $\boldsymbol{\Sigma}_k^{-1}$ and rearranging

$$0 = -\sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \boldsymbol{\Sigma}_k(\mathbf{x}_n - \boldsymbol{\mu}_k) \tag{29}$$

we obtain

$$0 = \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k) \tag{30}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{31}$$

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}) \tag{32}$$

MoG: EM algorithm

If we set the derivative of log likelihood with respect to $\boldsymbol{\Sigma}_k$ to zero, and follow a similar line of reasoning, making use of the result for the maximum likelihood solution for the covariance matrix of a single Gaussian, we obtain

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T \tag{33}$$

each data point weighted by the corresponding posterior probability and with the denominator given by the effective number of points associated with the corresponding component

MoG: EM algorithm

Finally, we maximize log likelihood with respect to the mixing coefficients pi_k. Here we must take account of the constraint $\sum_k pi_k = 1$. This can be achieved using a Lagrange multiplier and maximizing the following quantity

$$\log p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right) \tag{34}$$

which gives

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda \tag{35}$$

$$0 = \sum_{k=1}^{K} \sum_{n=1}^{N} \gamma(z_{nk}) + \lambda \sum_{k=1}^{K} \pi_k \tag{36}$$

Hence

$$\pi_k = \frac{N_k}{N} \tag{37}$$

MoG: Lagrange multiplier

The method can be summarized as follows: in order to find the maximum or minimum of a function $f(x)$ subjected to the equality constraint $g(x) = 0$, form the Lagrangian function

$$\mathcal{L}(x, \lambda) = f(x) - \lambda g(x) \tag{38}$$

and find the stationary points of $\mathcal{L}$ considered as a function of x and the Lagrange multiplier λ. The solution corresponding to the original constrained optimization is always a saddle point of the Lagrangian function, which can be identified among the stationary points from the definiteness of the bordered Hessian matrix.

MoG: Lagrange multiplier

Minimize $f(x, y) = x + y$ subject to the constraint $x^2 + y^2 = 1$, i.e.,

$$g(x, y) = x^2 + y^2 - 1 = 0 \tag{39}$$

Hence,

$$\mathcal{L}(x, y, \lambda) = f(x, y) + \lambda g(x, y) = x + y + \lambda(x^2 + y^2 - 1) \tag{40}$$

Gradient

$$\nabla_{x,y,\lambda}\mathcal{L}(x, y, \lambda) = (1 + 2\lambda x, 1 + 2\lambda y, x^2 + y^2 - 1) \tag{41}$$

and therefore,

$$\nabla_{x,y,\lambda}\mathcal{L}(x, y, \lambda) \iff \begin{cases} 1 + 2\lambda x = 0 \\ 1 + 2\lambda y = 0 \\ x^2 + y^2 - 1 = 0 \end{cases} \tag{42}$$

MoG: Lagrange multiplier

$$\nabla_{x,y,\lambda}\mathcal{L}(x,y,\lambda) \iff \begin{cases} 1+2\lambda x = 0 \\ 1+2\lambda y = 0 \\ x^2+y^2-1=0 \end{cases} \tag{43}$$

This yields

$$x = y = -\frac{1}{2\lambda}, \quad \lambda \neq 0 \tag{44}$$

$$\frac{1}{4\lambda^2} + \frac{1}{4\lambda^2} - 1 = 0 \tag{45}$$

So,

$$\lambda = \pm\frac{1}{\sqrt{2}} \tag{46}$$

which implies that the stationary points of $\mathcal{L}$ are

$$\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right), \quad \left(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right) \tag{47}$$

Mean-shift algorithm

Mean-shift algorithm

While k-means and mixtures of Gaussian use a parametric form to model the probability density function being segmented, mean shift implicitly models this distribution using a smooth continuous *non-parametric model*. The key to mean shift is a technique for efficiently finding peaks in this high-dimensional data distribution without ever computing the complete function explicitly.

Mean-shift algorithm

Mean shift uses a variant of what is known in the optimization literature as *multiple restart gradient descent*. Starting at some guess for a local maximum, $\mathbf{y}_k$, which can be a random input data point $\mathbf{x}_i$, mean shift computes the gradient of the density estimate $f(\mathbf{x})$ at $\mathbf{y}_k$ and takes an uphill step in that direction. The gradient of $f(\mathbf{x})$ is given by

$$\nabla f(\mathbf{x}) = \sum_i (\mathbf{x}_i - \mathbf{x}) G(\mathbf{x} - \mathbf{x}_i) = \sum_i (\mathbf{x}_i - \mathbf{x}) g\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h^2} \right) \tag{48}$$

where $g(r) = -k'(r)$ and $k'(r)$ is the first derivative of $k(r)$.

Mean-shift algorithm

We can re-write the gradient of the density function as

$$\nabla f(\mathbf{x}) = \left[\sum_i G(\mathbf{x} - \mathbf{x}_i)\right] \mathbf{m}(\mathbf{x}) \tag{49}$$

where the vector

$$\mathbf{m}(\mathbf{x}) = \frac{\sum_i \mathbf{x}_i G(\mathbf{x} - \mathbf{x}_i)}{\sum_i G(\mathbf{x} - \mathbf{x}_i)} - \mathbf{x} \tag{50}$$

is called the *mean shift*, since it is the difference between the weighted mean of the neighbors $\mathbf{x}_i$ around $\mathbf{x}$ and the current value of $\mathbf{x}$.

In the mean-shift procedure, the current estimate of the mode $\mathbf{y}_k$ at iteration k is replaced by its locally weighted mean

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \mathbf{m}(\mathbf{y}_k) = \frac{\sum_i \mathbf{x}_i G(\mathbf{y}_k - \mathbf{x}_i)}{\sum_i G(\mathbf{y}_k - \mathbf{x}_i)} \tag{51}$$

Mean-shift algorithm

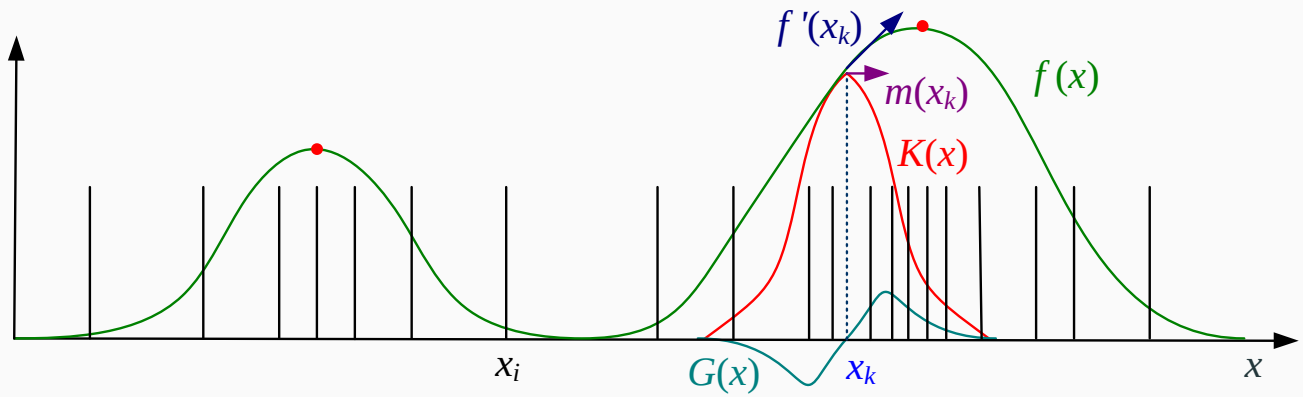


Figure 5.17 One-dimensional visualization of the kernel density estimate, its derivative, and a mean shift. The kernel density estimate $f(x)$ is obtained by convolving the sparse set of input samples x_i with the kernel function $K(x)$. The derivative of this function, $f'(x)$, can be obtained by convolving the inputs with the derivative kernel $G(x)$. Estimating the local displacement vectors around a current estimate x_k results in the mean-shift vector $m(x_k)$, which, in a multi-dimensional setting, point in the same direction as the function gradient $\nabla f(\boldsymbol{x}_k)$. The red dots indicate local maxima in $f(x)$ to which the mean shifts converge.

Mean-shift algorithm

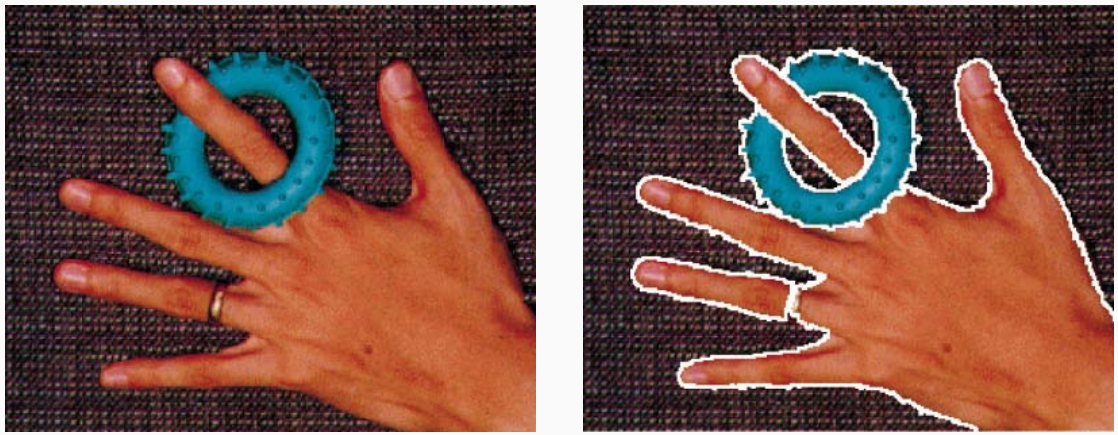
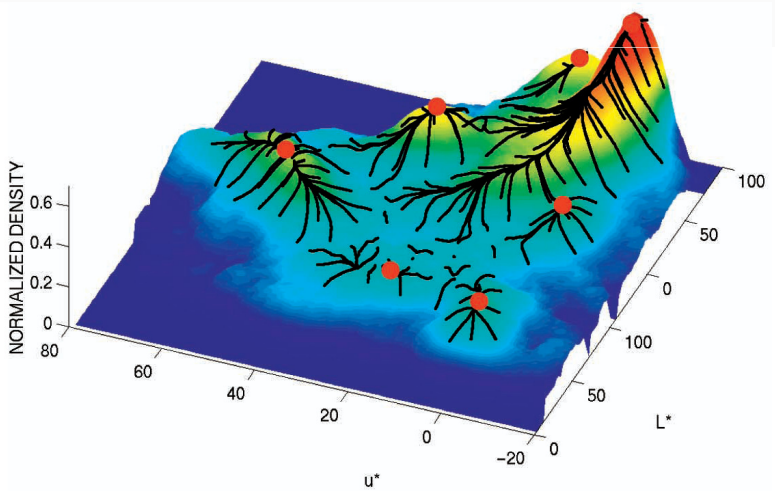


Figure 5.18 Mean-shift color image segmentation with parameters $(h_s, h_r, M) = (16, 19, 40)$ (Comaniciu and Meer 2002) © 2002 IEEE.

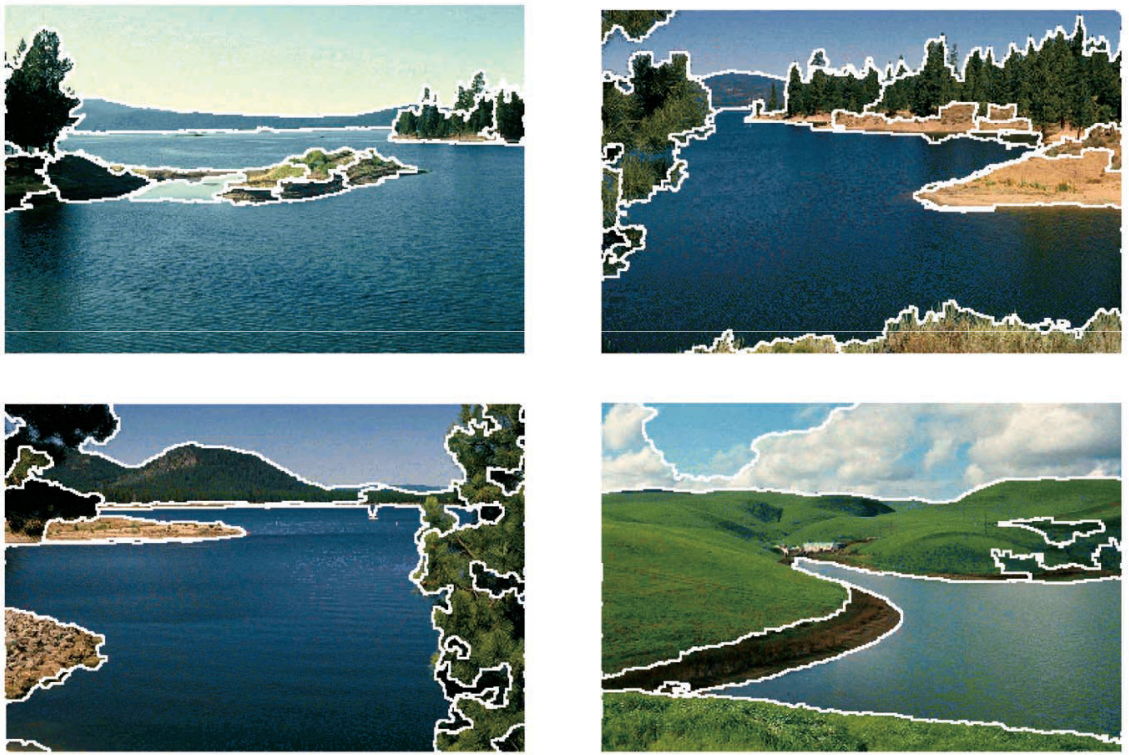
Mean-Shift Clustering

- Clustering in color space
  - In LUV color space



Result by Mean-Shift Segmentation

Result by Mean-Shift Segmentation



Mean-shift algorithm

Pros

- Does not assume spherical clusters
- Just a single parameter (window size)
- Finds variable number of modes
- Robust to outliers

Cons

- Output depends on bandwidth
- Ambiguity in optimal bandwidth selection
- Computationally expensive
- Does not scale well with dimension of feature space

Appendix

Reference and further reading

- “Chap 9 | Mixture Models and EM” of C. Bishop, Pattern Recognition and Machine Learning
- “Chap 9” of A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow
- “Chap 5 | Segmentation” of R. Szeliski, Computer Vision: Algorithms and Applications
- “Chap 9 | Segmentation by Clustering” of Forsyth and Ponce, Computer Vision: A Modern Approach
- “Lecture12 | Clustering and Image Segmentation (Part II)” and “Lecture13 | Clustering and Image Segmentation (Part III)” of Bohyung Han, CSED441: Introduction to Computer Vision, POSTECH (2011)