Yonsei University, Department of Applied Statistics

Chapter 10 Unsupervised Learning

12기 YBIGTA 신보현

March 16, 2019

이번 단원에서는 비지도 학습에 대해서 알아본다. 지도 학습은 예측변수와 반응변수의 값이 모두 주어졌지만 비지도 학습에서는 반응 변수의 값이 주어지지 않는다. 따라서 예측에 관심이 있지는 않다. 오히려 주어진 반응변수와 관련된 새로운 인사이트를 도출하는 것에 초점을 둔다. 데이터를 시각화하는 효과적인 방법이 있을까? 관측치들, 또는 변수들 사이에 subgroups을 만들 수 있을까? 이러한 질문들에 답하기 위해, 이번 단원에서는 PCA와 Clustering에 대해서 알아본다.

10.2 Principal Components Analysis

PCA는 6단원에서 PCR의 개념을 설명할 때 등장했다. 서로 상관된 많은 수의 변수에 직면했을 때, 원래 데이터 세트의 변동성을 가장 많이 설명하는 적은 수의 변수를 뽑아냄으로써 고차원 데이터를 저차원으로 축소시키는 기법이다. PCA는 원래 변수의 선형 결합으로 principal components을 만드는 과정에서 반응 변수 Y을 사용하지 않는다는 점에서 비지도 학습이다.

10.2.1 What Are Principal Components?

만약 예측변수가 매우 많다면 이에 대한 상관계수 그림이 많아져서 눈으로 확인하기 힘들어진다. 이렇게 많은 변수 중 대부분의 변수가 쓸모 없거나 변수끼리 상관성이 높은 상황을 생각해보자. 그러면 굳이 모든 변수를 사용하지 않아도 될 것이다. PCA는 이러한 상황에서 데이터의 정보를 가장 많이 담고 있는 변수를 추려낸다. 이제 데이터의 정보와 가장 많이 담고 있는 기준은 어떻게 정하는지, 그리고 그 변수를 어떻게 생성하는지 자세하게 알아보자.
PCA는 원래 데이터에서 p개의 변수가 있다면 이를 r개의 변수로 축약하는데, 원래 있던 p개의 변수의 선형결합으로 축약을 하며 이 축약된 변수를 principal component라고 부른다. 첫번째로 축약된 변수를 first principal component라고 부르고 이는 아래와 같이 작성할 수 있다.

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p = \phi^{'}X$$

6단원에서도 살펴 봤듯이, 벡터 ϕ는 $\phi^{'}\phi = 1$의 제약을 가지고 있다. 즉, $\sum_{j=1}^{p}\phi_{j1}^2 = 1$으로 normalized vector을 의미한다. 이렇게 제약을 두지 않으면, loadings을 임의로 크게 만들어서 매우 큰 분산을 임의적으로 만들 수 있기 때문이다. 벡터 ϕ의 원소를 loadings이라고 하며 이를 이용하여 벡터 ϕ를 principal component loading vector라고 부른다.
loadings을 구하면 first principal component를 구할 수 있는데 이를 어떻게 구할까? 6단원에서 살펴보았듯이, 우선 데이터가 centered 되었따는 가정 하에, 가장 큰 분산을 만드는 loadings을 고른다. 이는 아래와 같이 다시 쓸 수 있다.

$$z_{i1} = \phi_{11}x_{i1} + \cdots + \phi_{p1}x_{ip}$$

$$\underset{\phi_{11},\cdots,\phi_{p1}}{maximize}\left\{\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}x_{ij}\right)^2\right\}\ subject\ to\ \sum_{j=1}^{p}\phi_{j1}^2 = 1$$

이를 $\frac{1}{n}\sum_{i=1}^{n}z_{i1}^2$로 다시 적을 수 있으며 예측변수들은 모두 centered 되었으므로 평균이 0이고 z_{i1}의 평균도 0이므로 $\frac{1}{n}\sum_{i=1}^{n}z_{i1}^2$은 sample variance을 나타내는 것이다. 또한 각 관측치에 대해서 n개의 $z_{11}, z_{21}, \cdots, z_{n1}$이

나오는데, 이를 first principal component의 scores라고 부른다. 위 최적화 문제는 eigen decomposition으로 풀리며, 자세한 내용은 6단원에서 이미 언급했으므로 여기서는 생략하도록 한다.
loading vector인 ϕ는 데이터의 표본 분산이 가장 큰 방향으로의 direction을 정의한다. 그리고 n개의 관측치를 바로 이 direction에 project했을 때 (z_{i1}의 $x_{i1}, \cdots, x_{ip}$에 관측치 값을 넣었을 때) 값이 바로 principal component scores이다.

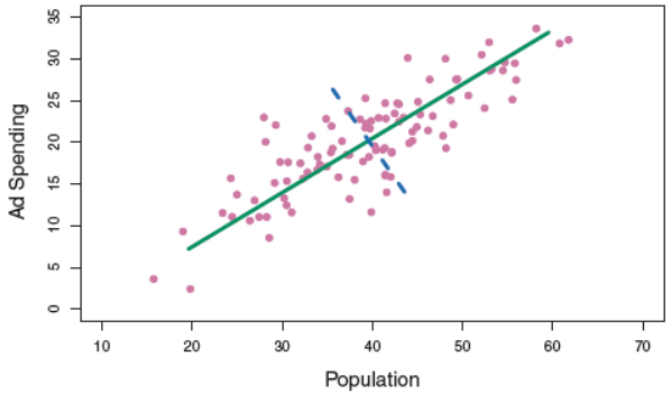

FIGURE 6.14. *The population size (pop) and ad spending (ad) for 100 different cities are shown as purple circles. The green solid line indicates the first principal component, and the blue dashed line indicates the second principal component.*

예를 들어 그림 6.14에서 초록색 선은 first principal component direction이다. 초록색 선으로 원래의 데이터를 투영했을 때, 그 퍼져있는 정도를 나타내는 것이 바로 표본 분산인데, 초록색 선 위에 있는 분홍색 점들을 생각해보면 다른 어느 선보다도 퍼져있는 정도가 가장 크다. 파랑색 점선은 second principal component이다. 이는 first principal component와 수직임을 그림에서 확인할 수 있다. 이것이 의미하는 바는, 두 principal component가 orthogonal하며 즉, 두 벡터가 독립이라는 것이다. 다시 말해서, first principal component가 설명하지 못하는 부분을 second principal component가 대체하여 설명하는 것으로 생각하자.
principal components을 계산했으면, 이를 이용하여 데이터에 대한 저차원 그림을 그릴 수 있다. 다음 예시를 보자. PCA를 USArrests 데이터에 적용해본다. PCA를 시행하기 이전에 표준화가 진행되었다. 6단원에서도 살펴보았듯이, 변수간에 단위가 크게 다르다면, 단위가 다름으로써 발생하는 크기 차이 때문에 분산이 커질 염려가 있어서 반드시 centering 또는 표준화를 한 후, PCA를 진행한다.

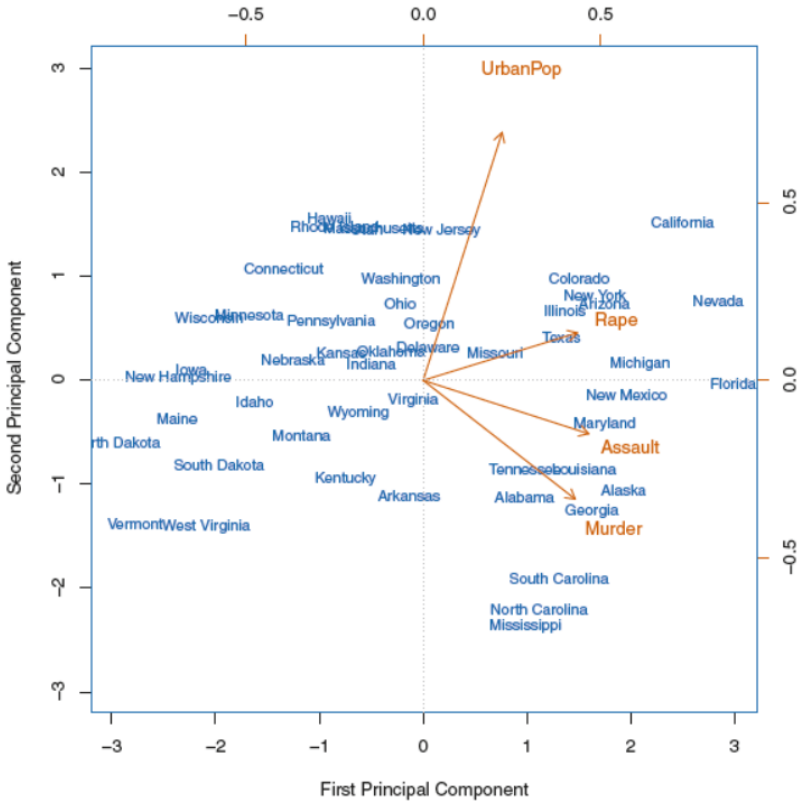

그림 10.1은 처음 두 개의 principal component에 대한 그림이다. 위 그림은 principal component scores와 loading vectors을 모두 보여주기 때문에 biplot이라고도 불린다. 파랑색의 주 이름은 두 principal component에 대한 scores을 나타낸다. 오랜지색의 화살은 두 principal component loading vectors을 의미한다(해당하는 축은 위와 오른쪽이다) 예를 들어 Rape의 first principal component에 대한 loading은 0.54이고 second principal component에 대한 loading은 0.17이다. 위 그림에서 first loading vector가 assault, murder, rape에 거의 동일한 가중치를 주고 urban pop에 적은 가중치를 주었음을 알 수 있다. 따라서 first principal component는 심각한 범죄에 관한 변수로 요약되었다고 볼 수 있다. second loading vector는 urban pop에 가장 많은 가중치를 둔다. 따라서 이는 시의 도시화에 관련된 변수라고 볼 수 있다.

두 principal component을 통해서 주들을 구분할 수 있다. first component에서 큰 양의 점수를 가지는 주, 예를 들어 캘리포니아는 높은 범죄율을 지니고 동시에 second component에서도 큰 양의 점수를 가지므로 높은 수준의 도시화가 진행되었다고 해석할 수 있다. 인디아나 같이 두 요소 모두에서 0에 가까운 값을 가진 주는 범죄율과 도시화 수준이 평균정도라고 볼 수 있다.

10.2.2 Another Interpretation of Principal Components

이전에 principal component loading vectors가 feature space에서 데이터가 가장 많이 변하는 방향이고, principal component scores를 데이터들을 dircetion으로 projection한 것임을 살펴보았다. 여기서는 principal components가 관측치에 가장 가까운 저차원의 선형 평면(linear surface)을 제공한다는 시각으로 해석해볼 것이다.

first principal component loading vector는 n개의 관측치로부터 가장 가까이 있는 평면이다. 이러한 해석은 그림 6.15에서 볼 수 있다.

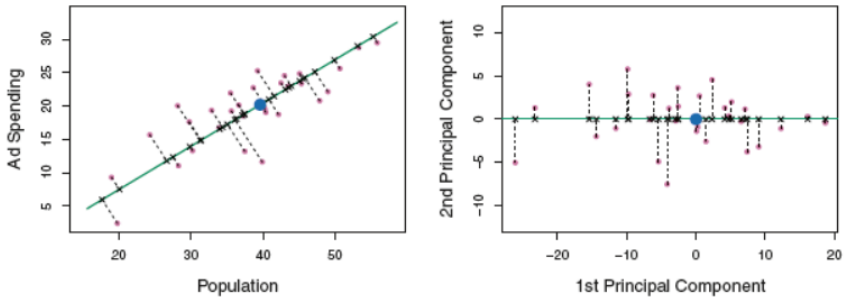

FIGURE 6.15. *A subset of the advertising data. The mean* pop *and* ad *budgets are indicated with a blue circle.* Left: *The first principal component direction is shown in green. It is the dimension along which the data vary the most, and it also defines the line that is closest to all n of the observations. The distances from each observation to the principal component are represented using the black dashed line segments. The blue dot represents* $(\overline{\text{pop}}, \overline{\text{ad}})$. Right: *The left-hand panel has been rotated so that the first principal component direction coincides with the x-axis.*

점선은 관측치와 first principal component loading vector간의 거리를 나타낸다. 즉, Euclidean 방법을 통해서 측정된 거리가 가장 작은 line이 데이터를 가장 잘 요약해줄 것이라고 생각하는 것이다. 이러한 해석은 first principal component에 국한되지 않고 확장된다.

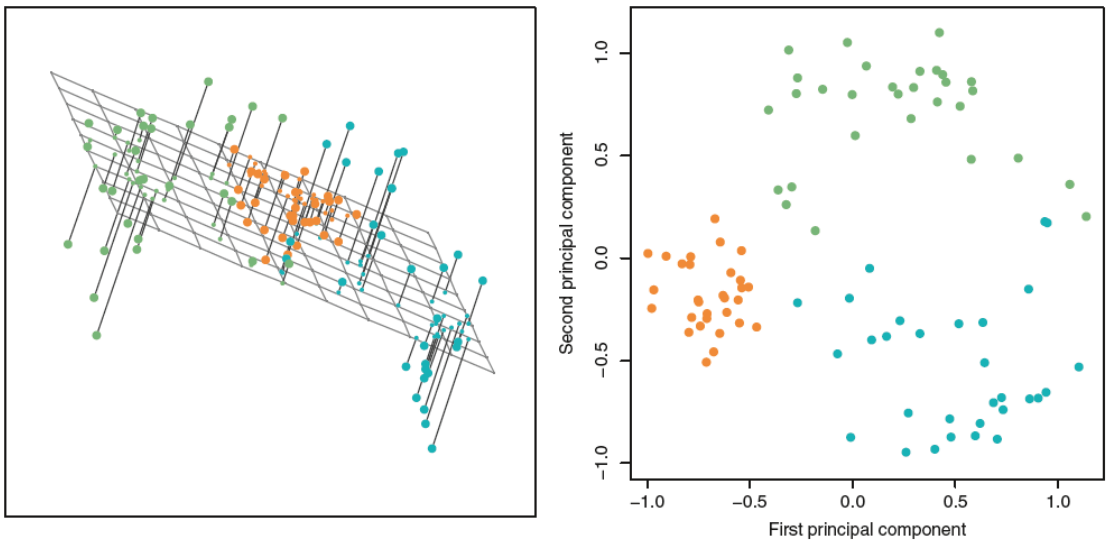

FIGURE 10.2. *Ninety observations simulated in three dimensions.* Left: *the first two principal component directions span the plane that best fits the data. It minimizes the sum of squared distances from each point to the plane.* Right: *the first two principal component score vectors give the coordinates of the projection of the 90 observations onto the plane. The variance in the plane is maximized.*

그림 10.2의 왼쪽을 보면 데이터와 가장 가까운 세 개의 principal components가 sapn하는 3차원 평면이 나와있다. 이러한 해석을 이용하여 첫 M개의 principal component score vectors와 첫 M개의 principal component loading

vectors는 i번째 관측치에 대해서 Euclidean 거리의 관점에서 가장 좋은 M차원의 근사를 제공한다. 이것은 아래와 같이 쓸 수 있다.

$$x_{ij} \approx \sum_{m=1}^{M} z_{im}\phi_{jm}$$

다시 말해서 M개의 principal component score vectors와 principal component loading vectors는 M이 충분히 클 때 원래 데이터에 대해서 좋은 근사를 보여준다.

10.2.3 More on PCA

Scaling the Variables

PCA를 시행하기 이전에, 변수들이 centered 되어야 한다고 이미 언급을 했다. 변수들이 각기 다른 단위에 의해서 기록이 되었다면 이는 principal components을 결정하는데 큰 단위가 큰 분산을 가질 가능성이 높으므로 영향을 줄 수밖에 없다. 따라서 scaling의 여부는 PCA의 결과에 상당히 큰 영향을 미친다.

Uniqueness of the Principal Components

각각의 principal component loading vector는 유일하다(부호가 달라질 수는 있음) 부호가 달라지면 direction의 방향이 180도로 바뀔뿐이다. 또한 score vectors도 unique한데, $Var(Z) = Var(-Z)$이기 때문이다.

The Proportion of Variance Explained

그림 10.2에서 3차원 데이터 세트에 대해 PCA를 시행했고 2차원의 그림을 얻기 위해 데이터를 처음 두 principal component에 project했다. projection을 하기 이전과 이후를 비교해보았을 때, projection을 한 후에도 각 군집의 특성이 잘 유지된 것으로 보아, PCA가 성공적으로 수행되었음을 알 수 있다.
그렇다면 이렇게 저 차원으로 projection함으로써 얼마나 많은 정보가 손실되었는지 궁금해진다. 다시 말해서, 처음 몇 개의 principal components가 얼마만큼의 데이터의 정보를 포함하지 않을지, 즉 데이터의 분산을 포함하지 않을지가 의문이다. 이 개념은 proportion of variance explained(PVE)이다. 즉, 각 변수가 centered 되었다는 가정하에 전체 데이터의 분산은 아래와 같다.

$$\sum_{j=1}^{p} \widehat{Var(X_j)} = \sum_{j=1}^{p} \frac{1}{n}\sum_{i=1}^{n} x_{ij}^2$$

그리고 m번째 principal component에 의해서 설명되는 분산은 아래와 같다.

$$\frac{1}{n}\sum_{i=1}^{n} z_{im}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{jm}x_{ij}\right)^2$$

따라서 m번째 principal component의 PVE는 아래와 같다.

$$\frac{\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{jm}x_{ij}\right)^2}{\sum_{j=1}^{p}\frac{1}{n}\sum_{i=1}^{n}x_{ij}^2}$$

첫 M개의 principal components의 누적합을 구하기 위해서는 M번째까지의 표본 분산을 더하면 된다.

Deciding How Many Principal Components to Use

보통 principal components을 구하면, 그 중 몇개만을 이용하여 데이터를 축소할지가 관심이다. 너무 많이 사용하면 데이터를 축소하는 의미가 없어지고, 그렇다고 적게 사용하면 정보를 많이 손실할 수 있기 때문이다. 이때 바로 PVE를 이용하여 principal components의 수를 결정한다. 이를 scree plot을 통하여 시각적으로 확인한다.

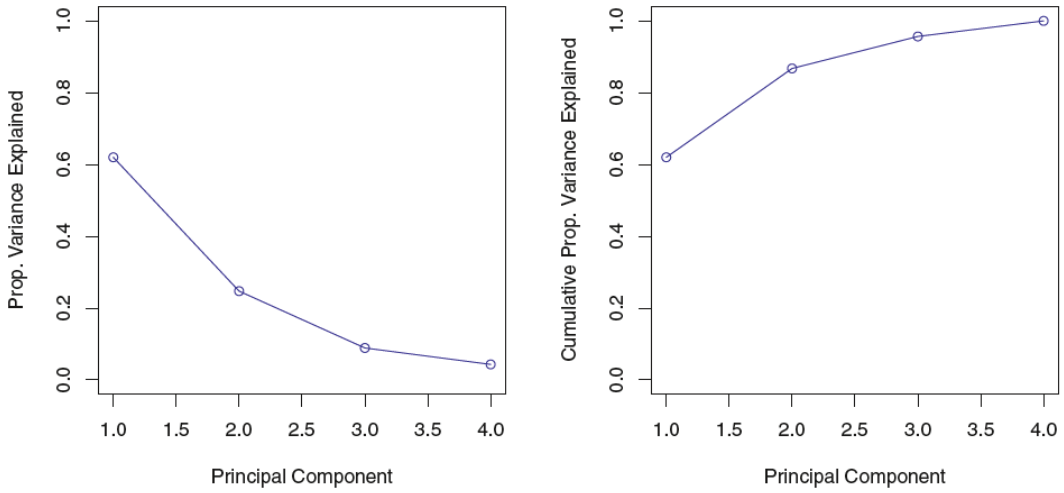

FIGURE 10.4. Left: *a scree plot depicting the proportion of variance explained by each of the four principal components in the* `USArrests` *data.* Right: *the cumulative proportion of variance explained by the four principal components in the* `USArrests` *data.*

가능한 variance을 많이 설명하는 쪽으로 최대한 적은 수의 principal component을 선택해야 한다. 왼쪽그림은 각 principal component의 PVE이고 오른쪽 그림은 PVE의 누적 합이다. 오른쪽 그림을 보면 first principal component가 거의 60%의 분산을 설명하고 third principal component와 second principal component의 PVE가 그리 큰 차이를 보이지 않음을 알 수 있다. 이러한 경우에는, 굳이 더 많은 개수를 선택하기 보다는, 목표가 차원 축소이므로 적은 수의 principal component을 선택한다. 따라서 몇 개의 principal component을 선택해야 한다는 규칙은 없다. 데이터마다 다르며 선택도 주관이 개입되어 한다.

만약에 위 그림과 같이 처음 몇개의 principal component에서 PVE가 눈에 띄게 크지 않다면, PCA를 더 진행하는 것이 그렇게 큰 의미가 있지 않을 것이다. 이러한 경우에는 거의 모든 변수를 포함하는 것이 PVE가 크게 나오는데, 그렇다면 차원을 축소하고자 하는 원래의 목적이 의미를 상실하기 때문이다.

하지만 만약 PCA가 비지도 학습이 아닌 지도 학습의 맥락에서 사용된다면 principal components의 수를 CV를 통해 객관적으로 정한다. 예를 들어 PCR에서는 CV error을 최소화하는 principal components의 수를 정한다.

10.3 Clustering Methods

클러스터링은 데이터 세트에서 클러스터, 또는 서브그룹을 찾는 기법을 뜻한다. 데이터 세트에서 클러스터를 찾고자 할 때, 같은 그룹내에서는 최대한 동질적이고, 다른 그룹끼리는 이질적인 클러스터를 찾는다. 물론 이를 위해 어떤한 기준으로 비슷하고 다른지 명확하게 정의해야 한다.

이번 단원에서는 K-means clustering과 hierarchical clustering에 대해서 알아본다. K-means clustering은 미리 정해진 클러스터의 개수로 관측치를 나눈다. 그와는 반면에 hierarchical clustering은 클러스터의 개수를 미리 정하지 않아도 된다. 나무와 비슷하게 생긴 dendrogram을 시각적으로 살펴보며 클러스터의 개수를 나중에 정한다. 일반적으로 변수를 기반으로 관측치를 클러스터링하는 것과 관측치를 기준으로 변수를 클러스터링하는 방법이 있다. 이번 단원에서는 전자에 대해서 공부한다.

10.3.1 K-Means Clustering

K-means clustering을 하기 위해선, ㄴ 사전에 cluster의 개수인 K를 미리 정해야 한다. 이후에는 알고리즘이 관측치들을 K개의 클러스터에 배분할 것이다. 그림 10.5는 150개의 만들어진 이차원의 관측치에 대해서 다른 K 값을 적용한 K-means clustering의 결과이다.

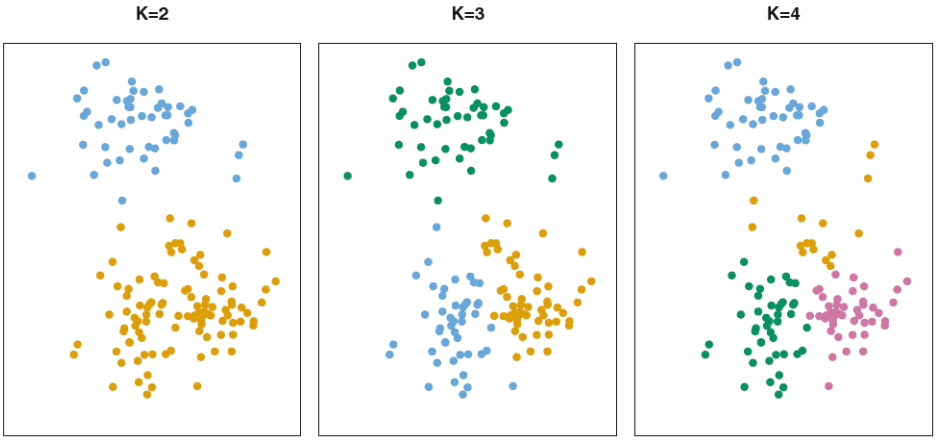

FIGURE 10.5. *A simulated data set with 150 observations in two-dimensional space. Panels show the results of applying K-means clustering with different values of K, the number of clusters. The color of each observation indicates the cluster to which it was assigned using the K-means clustering algorithm. Note that there is no ordering of the clusters, so the cluster coloring is arbitrary. These cluster labels were not used in clustering; instead, they are the outputs of the clustering procedure.*

K-meas clustering은 간단하고 직관적인 알고리즘으로 수행된다. 우선 notation을 정의하자. $C_1, \cdots, C_k$을 각 클러스터에 있는 관측치의 index들의 set라고 하자. 이 sets는 아래 두가지 특징을 만족한다.

1. $C_1 \cup C_2 \cup \cdots \cup C_k = \{1, \cdots, n\}$. 다시 말해 각 관측치는 K개의 클러스터 중 최소 하나에 속한다.
2. $C_k \cap C_{k'} = \phi$ for all $k \neq k'$. 다시 말해서 클러스터들은 겹치지 않는다. 어떠한 관측치도 두개 이상의 클러스터에 속하지 않는다.

아이디어는, 클러스터 내에서의 변동성이 가능한 작아야한다는 것이다. 그렇다면 그 변동성을 어떻게 측정할까? 각 클러스터의 관측치 사이의 변동성을 $W(C_k)$라고 정의하면 아래의 문제를 푸는 알고리즘일 것이다.

$$\underset{C_1,\cdots,C_k}{minimize} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

언뜻 보기에, $W(C_k)$을 어떻게 정의하느냐에 따라서 결과가 달라질 것으로 보인다. 가장 많이 사용하는 방법은 유클리디언 거리의 제곱을 이용하는 것이다.

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

$|C_k|$은 k번째 클러스터 내의 관측치 개수이다. 다시 말해서 k번째 클러스터 내의 변동성은 짝지은 두 관측치마다 유클리디안 거리를 계산해서 모두 더하고 그 클러스터의 관측치 개수로 나눈 것이다.
이제 어떤 알고리즘이 변동성의 합을 최소화하는지 알아보자.
N개의 데이터 $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$에 대해서 데이터가 속한 cluster의 중심과 데이터 간의 거리의 차이가 최소가 되도록 데이터들을 K개의 cluster $S = \{s_1, \cdots, s_K\}$에 할당한다고 하자. 그러면 위의 최적화 문제는 아래와 같이 다시 쓸 수 있다.

$$Q = \underset{r,c}{argmin} \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\boldsymbol{x}_n - c_k||^2$$

여기서 r_{nk}는 n번째 데이터가 k번째 cluster에 속하면 1, 아니면 0의 값을 가지는 indicator variable이며 c_k는 k번재 cluster의 중심을 뜻한다. 결국 k-means를 실행한다는 것은 주어진 데이터에 대해서 최적의 r_{nk}, c_k을 찾아나가는 과정이다.
위 최적화 문제를 풀기 위한 알고리즘으로는 1957년에 Stuart Lloyd가 제안한 알고리즘이 가장 많이 사용되고 있다. 알고리즘은 아래와 같다.

- 초기 c_k을 설정한다. 랜덤 초기화, Forgy 알고리즘, MacQueen 알고리즘 등 초기 c_k을 설정하는 다양한 방법들이 제안되었으며 가장 기본적인 알고리즘은 랜덤 초기화이다.

- 설정된 c_k을 고정한 채로 Q를 최소화하는 r_{nk}을 구한다. 샘플은 모두 독립적이라고 가정하므로 r_{nk}도 각각의 샘플에 대해 따로 최적화하면 된다. 즉, 다른 샘플과의 연관성을 고려할 필요 없이 현재 샘플에 대해 가장 타당한 r_{nk} 값을 선택하면 된다. 타당하다는 기준은 각 클러스터 중심과 샘플의 거리를 측정해서 가장 가까운 클러스터를 선택하면 된다.

$$r_{nk} = \begin{cases} 1 & if\ k = argmin_j ||\boldsymbol{x}_n - c_j||^2 \\ 0 & o.w \end{cases}$$

- 새롭게 얻어진 r_{nk}을 고정하고 다시 c_k을 구한다. 목적함수 Q는 c_k에 대해서 quadratic form이다. 따라서 미분을 통해 최소값이 되는 지점을 얻을 수 있다(다중 선형 회귀에서의 목적식과 동일함) 미분을 통해서 그 일차 도함수가 0이 되는 지점이 바로 Q를 최소로 만드는 c_k 값이다.

$$2 \sum_{n=1}^{N} r_{nk} (\boldsymbol{x}_n - c_k) = 0$$

$$\therefore c_k = \frac{\sum_{n=1}^{N} r_{nk} \boldsymbol{x}_n}{\sum_{n=1}^{N} r_{nk}}$$

여기서 r_{nk}가 n번째 관측치가 k번째 클러스터에 있는지 여부를 나타내는 indicator variable이었음을 상기해보자. 이를 생각해볼 때, 위에서 도출한 c_k는 k번째 클러스터에 속한 점들의 평균이다(k-means라는 이름이 붙여진 이유이기도 하다)

- 두 단계를 거치는 동안 데이터는 각각의 클러스터에 다시 할당이 되고 이렇게 재할당된 데이터를 이용하여 위 과정을 다시 반복한다. r_{nk}와 c_k가 수렴하는 지점에서 알고리즘을 멈춘다.

- 가장 처음 단계인 초기 중심 값 c_k을 설정하는 것이 다소 주관적일 수 있어서 여러 초기 값을 통해서 최적의 결과를 찾아내야 한다.

K-means clustering을 실행하기 위해서는 몇 개의 군집으로 나눌지, K 값을 미리 설정해야한다. 이에 대한 논의는 10.3.3에서 살펴보자.

10.3.2 Hierarchical Clustering

Hierarchical Clustering은 몇 개의 군집으로 나눌지 사전에 정하지 않는다. 또한 dendrogram이라는 매력적인 시각화를 표현할 수 있다는 것이 큰 장점이다. 여기서는 bottom-up 또는 agglomerative 클러스터링을 소개한다. 이는 hierarchical clustering에서 가장 흔한 타입이다.
그림 10.8의 만들어진 데이터를 살펴보자.

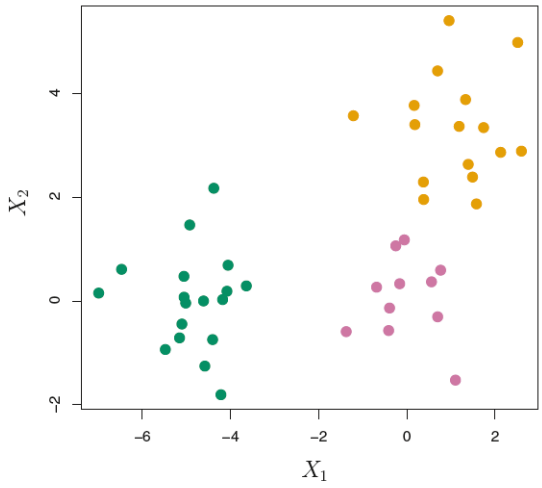

FIGURE 10.8. *Forty-five observations generated in two-dimensional space. In reality there are three distinct classes, shown in separate colors. However, we will treat these class labels as unknown and will seek to cluster the observations in order to discover the classes from the data.*

세 개의 클래스로부터 2차원 데이터가 생성되었다. 물론, 비지도 학습이라고 생각하면 위와 같이 라벨이 존재하지 않을 것이다. 그림 10.9는 hierarchical clustering의 결과인 dendrogram이다.

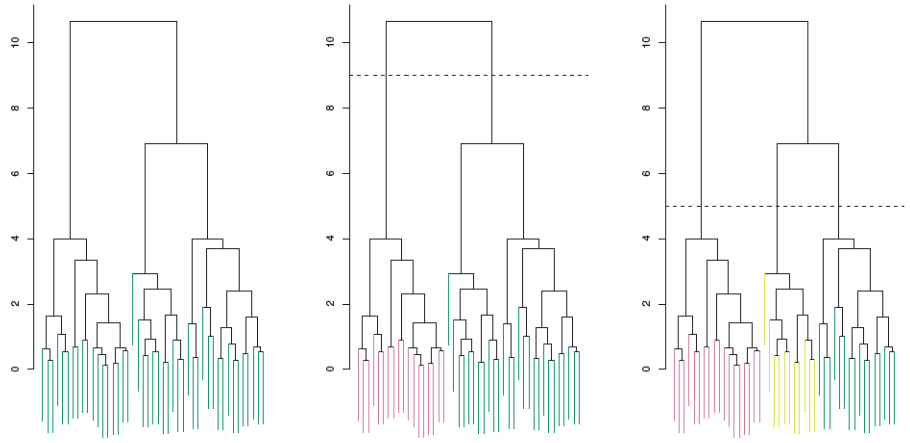

FIGURE 10.9. Left: *dendrogram obtained from hierarchically clustering the data from Figure 10.8 with complete linkage and Euclidean distance.* Center: *the dendrogram from the left-hand panel, cut at a height of nine (indicated by the dashed line). This cut results in two distinct clusters, shown in different colors.* Right: *the dendrogram from the left-hand panel, now cut at a height of five. This cut results in three distinct clusters, shown in different colors. Note that the colors were not used in clustering, but are simply used for display purposes in this figure.*

왼쪽 그림을 보면 각 나뭇잎은 그림 10.8의 관측치를 의미한다. 하지만 나무를 올라갈수록 나뭇잎들이 가지로 합쳐지기 시작한다. 합쳐진 나뭇잎은 서로 비슷한 특성을 가진다. 위로 올라갈수록 가지도 합쳐진다. 이러한 결합이 일찍 일어날수록, 즉 나무 하단부에서 일어날수록 그 그룹의 관측치들끼리는 서로 더 비슷함을 의미한다. 그와는 반면에 나중에 합쳐지는 관측치는 꽤나 다른 관측치이다. 사실, 이를 더 간단하게 말할 수 있다. 어떤 두 관측치에 대해서 이 두 관측치가 처음으로 합쳐지는 가지가 나무의 어느 지점에 있는지 보는 것이다. 이 결합의 높이가 수직축으로 측정되어 두 관측치가 얼마나 다른지를 나타낼 수 있다.

이는 dendrogram을 해석하는데 아주 중요한 부분이다.

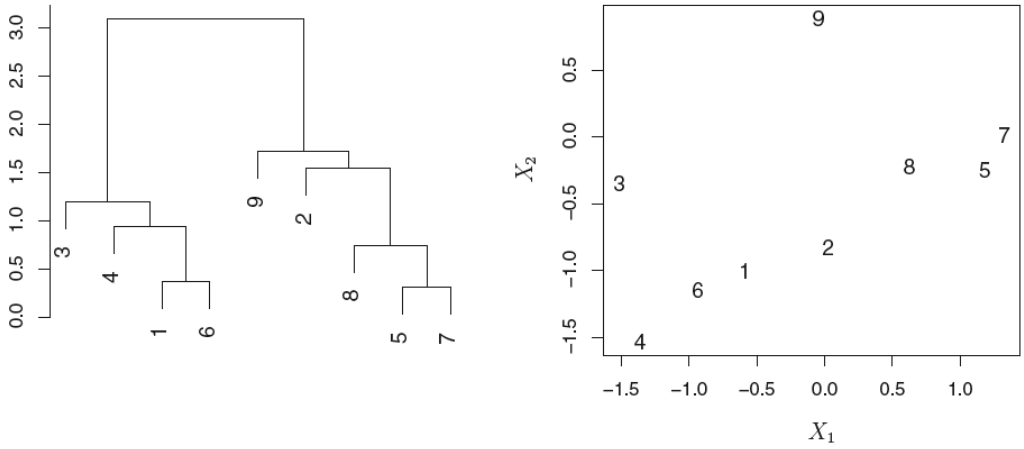

FIGURE 10.10. *An illustration of how to properly interpret a dendrogram with nine observations in two-dimensional space.* Left: *a dendrogram generated using Euclidean distance and complete linkage. Observations* 5 *and* 7 *are quite similar to each other, as are observations* 1 *and* 6. *However, observation* 9 *is* no more similar to *observation* 2 *than it is to observations* $8, 5$, *and* 7, *even though observations* 9 *and* 2 *are close together in terms of horizontal distance. This is because observations* $2, 8, 5$, *and* 7 *all fuse with observation* 9 *at the same height, approximately* 1.8. Right: *the raw data used to generate the dendrogram can be used to confirm that indeed, observation* 9 *is no more similar to observation* 2 *than it is to observations* $8, 5$, *and* 7.

그림 10.10은 9개의 관측치에 대해서 hierarchical clustering을 한 결과이다. dendrogram의 가장 낮은 곳에서 결합되므로 관측치 5와 7이 굉장히 유하다고 추측할 수 있다(관측치 1과 6도 마찬가지) 하지만 dendrogram에서 가지 하나 차이라고 관측치 2와 9가 유사하다고 판단해서는 안 된다. 사실 관측치 9는 오른쪽 그림을 보면 8,5,7에 비해서 2와 유사하다고 말할 수 없다. 따라서 수평선으로 두 관측치의 유사도를 유추해서는 안 된다. 두 관측치가 처음으로 결합되는 가지의 수직 높이에 따라서 유사도를 판단해야 한다.

그림 10.9에서 dendrogram을 수평선을 통해 몇 개의 cluster로 나눈다. 가운데에서는 높이 9에서 자름으로써 두 개의 클러스터가 나왔다. 낮은 높이에서 자를수록 더 많은 수의 클러스터가 나올 것이다. 다시 말해서 dendrogram을 자르는 높이는 K-means clustering에서의 K 역할을 하는 것이다. cluster의 수를 조정한다.

그림 10.9는 hierarchical clustering의 매력적인 부분을 강조한다. 하나의 dendrogram에 대햇 여러번 자름으로써 여러 clustering 결과를 얻을 수 있다는 것이다.

hierarchical하다는 뜻은 주어진 높이에서 dendrogram을 잘라서 얻어지는 클러스터는 더 높은 높이에서 자름으로써 얻어진 클러스터에 포함되는 상황을 의미한다. 하지만 임의의 데이터 세트에 대해서 이러한 hierarchical 구조는 비현실적이다. 예를 들어 관측치가 남자, 여자 각 50명이라고 하고 미국인, 일본인, 프랑스인을 동일하게 나누었다고 하자. 가장 좋은 분할은 성별에 의해서 나누고 국적에 의해서 세 그룹으로 나누는 것이다. 이러한 경우에 진정한 클러스터는 서로 포함관계에 있지 않다. 결과적으로 이러한 상황은 hierarchical clustering에 의해서 잘 표현될 수 없다. 이러한 상황때문에 hierarchical clustering은 K-means clustering보다 더 나쁜 결과를 종종 낸다.

The Hierarchical Clustering Algorithm

이제 hierarchical clustering의 알고리즘을 알아보자. 우선 한 쌍의 관측치가 서로 다름을 어떤 기준으로 판단하는지 알아보자. 가장 많이 유클리디안 거리가 사용된다(다른 종류는 단원 마지막에 소개된다) 알고리즘은 밑에서부터 시작된다. n개의 관측치가 각각의 클러스터로 여겨진다. 그리고 가장 유사한 클러스터가 합쳐지고 n-1개의 클러스터가 된다. 그리고 다시 가장 비슷한 두 개의 클러스터가 합쳐져서 n-2개의 클러스터가 된다. 알고리즘은 모든 관측치가 하나의 단일한 클러스터에 속할때까지 결합을 진행한다.

Algorithm 10.2 *Hierarchical Clustering*

1. Begin with n observations and a measure (such as Euclidean distance) of all the $\binom{n}{2} = n(n-1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.

2. For $i = n, n-1, \ldots, 2$:

   (a) Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.

   (b) Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

TABLE 10.2. *A summary of the four most commonly-used types of linkage in hierarchical clustering.*

이 알고리즘은 충분히 간단해보이는데 문제점이 있다. 아래 그림 10.11을 보자.

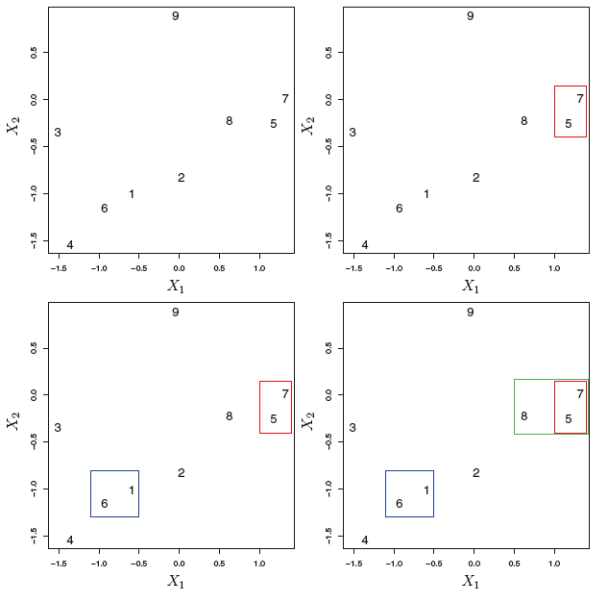

FIGURE 10.11. *An illustration of the first few steps of the hierarchical clustering algorithm, using the data from Figure 10.10, with complete linkage and Euclidean distance.* Top Left: *initially, there are nine distinct clusters,* $\{1\}, \{2\}, \ldots, \{9\}$. Top Right: *the two clusters that are closest together,* $\{5\}$ *and* $\{7\}$, *are fused into a single cluster.* Bottom Left: *the two clusters that are closest together,* $\{6\}$ *and* $\{1\}$, *are fused into a single cluster.* Bottom Right: *the two clusters that are closest together using* complete linkage, $\{8\}$ *and the cluster* $\{5, 7\}$, *are fused into a single cluster.*

클러스터 {5,7}과 클러스터 {8}이 합쳐지도록 어떻게 결정했을까? 관측치 쌍에 대한 다름의 기준이 있지만 여러 개의 관측치를 포함하는 클러스터 간의 다름에 대한 기준을 어떻게 정의할까? 이러한 다름에 대한 개념은 linkage으로 확장된다. 이는 여러 관측치를 포함하는 그룹 간의 다름을 측정한다. 가장 많이 사용되는 linkage의 종류는 complete, average, single, centroid이다. average, complete linkage는 좀 더 균형된 dendrograms을 만들기 때문에 더 선호된다.

Choice of Dissimilartiy Measure

여태껏, 다름을 측정하는 방법으로 유클리디안 거리만을 사용했다. 하지만 다른 방법도 선호된다. 예를 들어서 correlation based distance는 두 관측치가 유클리디안 거리로 떨어져 있더라도 높게 correlated 되어 있다면 유사하다고 판단한다(mahalanobis distance?)

이렇게 다름의 기준을 정하는 것은 결과로 나오는 dendrogram에 영향을 크게 미치기 때문에 매우 중요하다. 일반적으로 데이터의 타입을 잘 살펴봐야한다. 예를 들어서 고객을 클러스터링하는데에 관심이 있는 상인을 생각해보자. 유사한 고객들끼리 그룹핑하여 그들이 관심있어할만한 상품을 홍보하는 것이 목표이다. 데이터의 모양은 행은 고객이고 열은 구매 가능한 아이템이라고 하자. 데이터의 요소는 그 상품을 구매한 횟수이다. 이럴때 고객을 클러스터하기 위해서 어떠한 다름의 기준이 사용되어야할까? 유클리디안 거리가 사용되면 전체적으로 구매를 매우 적게한 고객들끼리 묶일 것이다. 이는 본래 의도와 맞지 않은 결과이다. 이와 반면에 correlation-based distance가 사용된다면 유사한 기호를 가진 고객들끼리 묶일 것이다. 따라서 이러한 경우에는 correlation-based distance가 더 나은 선택이다.

추가적으로 클러스터링을 할 때에는 다름을 측정하기 전에 반드시 표준편차가 1이 되도록 스케일링을 해주어야

한다. 위에서 언급한 예시를 계속 살펴보자. 어떠한 상품은 다른 것보다 더 많이 구매될 것이다. 예를 들어, 양말은 일년에 많이 사지만 컴퓨터는 많아봐야 1대정도 살 것이다. 이렇게 높은 값을 가지는 상품은 큰 값을 가지게 되어 고객 유사도 측정에 큰 영향을 끼치고 따라서 클러스터링 결과에도 영향을 미칠 것이다. 사실 가격으로 보자면, 컴퓨터가 훨씬 더 비쌈에도 횟수가 1이어서 영향을 미치지 못하는 상황은 적절하지 못하다. 이러한 상황을 방지하기 위해서 표준화가 필요하다. 이렇게 스케일링을 하는 것은 hierarchical clustering 뿐만 아니라 K-means clustering 도 마찬가지 이슈이다.

10.3.3 Practical Issues in Clustering

Small Decisions with Big Consequences

클러스터링을 하기 위해서는 몇 가지를 결정해야 한다. 변수들이 표준화 되어야 한다. hierarchical clustering 인 경우에는 다름의 기준과 linkage을 무엇으로 정해야 할지, dendrogram을 어디서 잘라야할지 정해야한다. K-means는 K를 사전에 정해야 한다. 이러한 질문들은 간단해보이지만 어떤 것을 선택하느냐에 따라서 클러스터링 결과에 지대한 영향을 미친다. 실전에서는 여러 방법을 해본 후에 가장 해석가능한 솔루션을 내는 결과를 택한다. 비지도학습이기 때문에 성능을 파악할 객관적인 기준이 없으므로 가장 '최적'의 답을 찾아야 한다.

Validating the Clusters Obtained

클러스터링은 비지도 학습이기 때문에 결과로 분류된 클러스터가 실제로 맞는지 확인할 길이 없다. 클러스터에 p-value을 배정하는 등의 방법이 제시되었지만 아직도 정확한 방법은 제시되지 않았다.

Other Considerations in Clustering

예를 들어 대부분의 관측치가 적은 수의 그룹에 속하고 나머지 소수의 관측치가 대부분의 다른 것들과 꽤나 다른 상황을 생각해보자. 즉 몇개의 이상치가 있는 상황이다. K-means와 hierarchical clustering은 모든 관측치가 클러스터에 들어가도록 강제하기 때문에 결과로 나온 클러스터가 이상치에 의해서 매우 왜곡될 가능성이 있다. 이러한 이상치가 존재하는 상황에 대안으로 나온 것이 soft K-means(ESL) 이다.