

7. 비지도 학습

비지도 학습이라는 용어는 레이블이 달린 데이터를 이용해 모델을 학습하는 과정 없이 데이터로부터 의미를 이끌어내는 통계적 기법들을 의미한다. 4장과 5장의 목적은 예측변수로부터 어떤 응답변수를 예측하는 모델을 만드는 것이었다. 비지도 학습 역시 데이터로부터 모델을 만드는 것이 목적이긴 하지만, 응답변수와 예측변수 사이의 구분이 없다.

비지도 학습은 여러가지 서로 다른 목표들이 있을 수 있다. 어떤 경우에는 레이블이 정해진 응답변수가 없는 상태에서 예측 규칙을 만드는 데 사용할 수 있다. 데이터의 의미 있는 그룹들을 찾기 위해 **클러스터링**을 사용할 수 있다.

예를 들면 웹사이트에서 사용자의 클릭 데이터와 인구통계 정보를 이용해 서로 다른 성격의 사용자들을 그룹화 할 수 있을 것이다. 이를 통해 웹사이트를 사용자 그룹의 기호에 맞게 개선할 수 있을 것이다.

또 이런 경우에는 데이터의 변수들을 관리할 수 있을 만한 수준으로 **차원을 줄이는 것**이 목표가 될 수도 있다.

이렇게 줄인 데이터는 회귀 혹은 분류 같은 예측 모델에 입력으로 사용할 수 있을 것이다. 예를 들어 제조 공정을 모니터링하기 위해 수천 개의 센서를 사용한다고 하자. 공정 실패를 예측하는 모델을 만들고자 할 때, 전체 데이터의 차원을 훨씬 작은 차원의 의미 있는 피쳐 데이터로 줄일 수 있다면, 수천 개의 센서에서 나오는 데이터를 전부 포함하는 것보다 좀 더 강력하면서도 해석하기 쉬운 모델을 만들 수 있을 것이다.

마지막으로 변수와 레코드의 수가 아주 큰 상황이라면, 데이터 비지도 학습을 탐색적 데이터 분석의 연장으로 볼 수도 있다. 이때는 데이터와 다른 변수들 사이에 서로 어떤 관계가 있는지에 대한 통찰을 얻는 것이 목적이 된다.

비지도 학습은 변수들을 정밀하게 조사하고 관계를 밝혀내는데 유용한 방법들을 제시한다.

비지도 학습과 예측

비지도 학습은 회귀와 분류 문제 모두에서 예측에 중요한 역할을 한다. 레이블이 없는 데이터에 대해 분류를 예측하고 싶은 경우가 있을 수 있다. 예를 들어 위성 센서 데이터로부터 어떤 지역의 식생 유형을 예측하고자 한다고 하자. 모델을 훈련시킬 수 있는 응답변수가 없으므로, 클러스터링을 통해 공통적인 패턴을 식별하고 지역을 분류할 수 있다.

클러스터링은 특히 콜드스타트 문제에서 유용한 방법이다. 새로운 마케팅 홍보를 론칭하거나 잠재적인 새로운 형태의 사기나 스팸을 걸러내는 유형의 문제에서는, 모델을 훈련시킬 수 있는 응답 데이터를 초기에 갖고 있지 않다. 시간이 지나고 데이터가 쌓이면 시스템에 대해 좀 더 알게 되고 전형적인 예측 모델을 학습할 수 있다. 이럴 때 클러스터링은 패턴이 비슷한 데이터들을 분류하여 학습 과정을 더 빨리 시작할 수 있도록 도와준다.

또한 비지도 학습은 회귀와 분류 방법을 위한 중요한 기본 요소가 될 수 있다. 빅데이터에서 어떤 작은 부모집단이 전체 모집단을 잘 대표하지 못한다면, 미리 학습된 모델은 이 부모집단에 대해 좋은 성능을 보일 수 없을 것이다.

클러스터링을 사용하면 부모집단을 식별하고 레이블을 지정할 수 있다. 그러면 서로 분리된 부모집단들을 각각 다른 모델에 피팅할 수 있다. 아니면 반대로, 전체 모델이 부모집단 정보를 명시적으로 예측변수로 고려하도록 하는 방법도 가능하다(부모집단의 식별자를 피쳐로 사용).

7.1 주성분분석

흔히 변수들은 함께 변하기 때문에(공변(covary)), 어느 한 변수에서의 일부 변화는 실제로 다른 변수에서의 변화에 의해 중복되기도 한다. **주성분분석(PCA)**은 수치형 변수가 어떤 식으로 공변하는지 알아내는 기법이다.

용어 정리

- **주성분** : 예측변수들의 선형결합
- **부하** : 예측변수들을 성분으로 변형할 때 사용되는 가중치(유의어 : 가중치)
- **스크리크래프** : 성분들의 변동을 표시한 그림. 성분들의 상대적인 중요도를 보여준다.

PCA의 아이디어는 다수의 수치형 예측변수들을 더 적은 수의 변수들의 집합으로 나타내는 것이다. 이 때 이 새로운 변수들은 원래 변수들에 가중치를 적용한 선형결합을 의미한다. 전체 변수들의 변동성을 거의 대부분 설명할 수 있는 적은 수의 변수들의 집합을 **주성분**이라고 하며, 이를 이용해 데이터의 차원을 줄일 수 있다. 주성분을 만드는 데 사용되는 가중치는 결국 새로운 주성분을 만드는 데 기존의 변수들이 어느 정도 기여하는지를 보여준다.

PCA에 대한 최초의 논문에서 담고 있는 내용은 다음과 같다. 다수의 문제에서 예측변수에 변동성이 존재한다는 사실을 인지했고, 이러한 변동성을 모델링하기 위한 방법으로 PCA를 개발했다. PCA는 선형판별분석(5.2절 참조)의 비지도 학습 버전이라고도 볼 수 있다.

7.1.1 간단한 예제

다음과 같이 두 변수 X_1, X_2 에 대해 두 주성분 $Z_i (i = 1 \text{ or } 2)$ 이 있다고 하자.

$$Z_i = w_{i,1}X_1 + w_{i,2}X_2$$

가중치 ($w_{i,1}, w_{i,2}$)를 주성분의 **부하**라고 한다. 이것은 원래 변수를 주성분으로 변환할 때 사용된다. 첫 주성분 Z_1 은 전체 변동성을 가장 잘 설명하는 선형결합이다. 두 번째 주성분 Z_2 는 나머지 변동성을 설명한다(동시에 최악의 피팅을 보이는 선형 조합이기도 하다).

Note_ 주성분은 값 자체에 대해서보다는 예측변수들의 평균으로부터의 편차에 대해 계산하는 것이 일반적이다.

R에서는 princomp 함수를 통해 주성분을 계산할 수 있다. 다음은 세브런(CVX)과 엑슨모빌(XOM)의 주가 수익 데이터에 PCA를 적용하는 것을 보여준다.

```
> #####
> ## PCA for oil data
> oil_px <- sp500_px[, c('CVX', 'XOM')]
> oil_px = as.data.frame(scale(oil_px, scale=FALSE))
> pca <- princomp(oil_px)
> pca$loadings
```

Loadings:

	Comp.1	Comp.2
CVX	0.747	0.665
XOM	0.665	-0.747

	Comp.1	Comp.2
SS loadings	1.0	1.0
Proportion var	0.5	0.5
Cumulative var	0.5	1.0

첫 번째 주성분에서 CVX와 XOM에 대한 가중치는 각각 0.747와 0.665이고, 두 번째 주성분에서의 가중치는 각각 0.665와 -0.747이다. 이 결과를 어떻게 해석할 수 있을까? 첫 번째 주성분은 근본적으로 두 석유 회사 사이의 상관관계를 반영하는 CVX와 XOM의 평균을 의미한다. 반면 두 번째 주성분은 CVX와 XOM의 주가가 달라지는 지점을 반영한다.

데이터와 주성분을 직접 그려보는 것은 아주 큰 도움이 된다.

```
ng(filename=file.path(PSDS_PATH, 'figures', 'psds_0701.png'),
width = 4, height=4, units='in', res=300)
loadings <- pca$loadings
ggplot(data=oil_px, aes(x=CVX, y=XOM)) +
  geom_point(alpha=.3) +
  scale_shape_manual(values=c(46)) +
  stat_ellipse(type='norm', level=.99, color='grey25') +
  geom_abline(intercept = 0, slope = loadings[2,1]/loadings[1,1],
color='grey25', linetype=2) +
  geom_abline(intercept = 0, slope = loadings[2,2]/loadings[1,2],
color='grey25', linetype=2) +
  scale_x_continuous(expand=c(0,0), lim=c(-3, 3)) +
  scale_y_continuous(expand=c(0,0), lim=c(-3, 3)) +
  theme_bw()

dev.off()
```

두 점선은 각각 주성분을 보여준다. 첫 번째 선은 타원의 장축을 따라 존재하며 두 번째 선은 단축위에 존재한다.

두 주가 수익에서 대부분의 변동성은 첫 번째 주성분을 통해 설명이 가능한 것을 알 수 있다. 이는 석유 관련 주가가 한 그룹으로 움직이는 경향이 있다는 측면에서 쉽게 이해 할 수 있다.

NOTE 첫 번째 주성분에 대한 가중치 값은 모두 양수 였지만 모든 가중치의 부호를 반대로 해도 주성분은 변하지 않는다. 원점과 (1,1)을 연결하는 직선이나, 원점과 (-1,-1)을 연결하는 직선은 동일하기 때문이다.

7.1.2 주성분 계산

변수가 두 개일 때를 살펴봤는데 세 개 이상일 때로 확장하는 것도 간단하다. 첫 성분의 선형 결합 수식에 그냥 예측변수를 추가하기만 하면 된다. 예측변수들의 첫 주성분에 대한 공변동(covariation)(공분산이 아니다.) 집합이 최적화되도록 가중치를 할당한다.

주성분을 계산하는 것은 전통적인 통계 기법이다. 데이터의 상관행렬 혹은 공분산행렬을 구하는 방식이기에 계산이 빠르고 반복이 필요 없다. 앞서 언급한 대로 이는 수치형 변수에 적용되며 범주형 변수에는 적용할 수 없다. 전체 과정은 다음과 같다.

1. 첫 번째 주성분을 구하기 위해 PCA는 전체 변동을 최대한 설명하기 위한 예측변수의 선형결합을 구한다.
2. 이 선형결합은 첫 번째 새로운 예측변수 Z_1 이 된다.
3. 같은 변수들을 이용해 새로운 두 번째 변수 Z_2 를 만들기 위해, 다른 가중치를 가지고 이 과정을 반복한다. 가중치는 Z_1 와 Z_2 가 서로 상관성이 없도록 결정한다.
4. 원래 변수 X_i 의 개수만큼 새로운 변수 Z_i 를 구할 때까지 이 과정을 계속한다.
5. 대부분의 변동을 설명하기 위해 필요한 만큼의 주성분을 선택해 남겨놓는다.

6. 결과적으로 각 주성분에 대한 가중치 집합을 얻는다. 마지막 단계는 원래 데이터를 이 가중치들을 적용해 새로운 주성분으로 변형하는 것이다. 이렇게 얻은 새로운 값들을 예측변수의 차원이 축소된 형태로 사용할 수 있다.

7.1.3 주성분 해석

주성분들의 특징으로부터 데이터 구조에 대한 정보를 얻을 수 있다. 주성분에 대한 이해를 돕기 위해 사용되는 표준화된 두 가지 시각화 방법이 있다. 한 가지 방법은 주성분의 상대적인 중요도를 표시해주는 스크리그래프이다.

(그림이 마치 절벽이 있는 산비탈 모양과 흡사하다고 해서 이름이 그렇게 붙었다) 다음은 S&P 500에 속한 몇몇 상위 기업의 정보를 이용한 예제이다.

위 그림에서 보듯이, 첫 번째 주성분의 변동이 가장 크고 나머지 상위 주성분일수록 중요한 것을 볼 수 있다.

또한 상위 주성분들의 가중치를 표시해보는 것도 주성분을 이해하는 데 도움이 된다. 이를 위해 ggplot 패키지와 함께 tidyr 패키지의 gather 함수를 사용할 수 있다.

아래 그림은 상위 다섯 개 성분의 부하를 보여준다. 첫 번째 주성분에서 부하량은 모두 같은 부호를 갖는다. 이를 통해, 이 데이터는 모든 변수로부터 비슷한 정도의 영향을 공유한다는 것을 알 수 있다(즉 모두가 전반적인 주식시장의 흐름에 영향을 받는다고 해석할 수 있다). 두 번째 주성분은 다른 주식과 다른 에너지 관련 주식들만의 가격 변동을 잡아낸다. 세 번째 주성분은 주로 애플(APPL)과 코스트코(COST)의 움직임이 서로 반대라는 것을 보여준다. 네 번째 주성분은 슬룸베르거(SLB)와 나머지 에너지 회사들의 움직임이 반대인 것을 보여준다.

마지막으로 다섯 번째 주성분에서는 금융회사들이 주를 이루고 있다.

NOTE_ 주성분을 몇 개까지 골라야 할까?

데이터의 차원을 줄이는 것이 목적이라면, 주성분을 몇 개까지 고를지를 결정해야 한다. 가장 일반적인 대부분의 변동성을 설명할 수 있는 성분들을 고르기 위한 특별한 규칙을 사용하는 것이다. 스크리그래프를 활용할 수도 있다. 예를 들어 상위 5개의 주성분까지로 분석을 제한하는 것도 좋은 방법이다.

아니면 누적 분산이 어떤 임계치(예를 들어 80%)를 넘어가도록 하는 상위 주성분들을 선택할 수도 있다.

또는 직관적인 해석이 가능한 성분들이 있는지 알아보기 위해 부하를 살펴볼 수 있다. 교차타당성검사는 중요한 주성분들의 개수를 결정하기 위한 좀 더 공식적인 방법이다.

주요 개념

- 주성분은 예측변수(수치형)들의 선형 결합이다.
- 주성분들은 서로 간의 상관관계가 최소화되며 중복성이 줄어들도록 한다.
- 제한된 개수의 주성분들로도 결과변수에서 대부분의 변동을 설명할 수 있다.
- 제한된 개수의 주성분들을 원래의 예측변수를 대신하여 차원이 감소된 형태로 사용할 수 있다.

7.2 K평균 클러스터링

클러스터링(군집화)은 데이터를 서로 다른 그룹으로 분류하는 기술을 말한다. 각 그룹에는 서로 비슷한 데이터들이 속하게 된다. 클러스터링의 목적은 데이터로부터 유의미한 그룹들을 구하는 것이다. 이렇게 얻은 그룹들을 직접 사용할 수도 있다. **K-평균**은 최초로 개발된 클러스터링 기법이다. 알고리즘이 상대적으로 간단하고 데이터 크기 커져도 손쉽게 사용할 수 있다는 점에서 아직도 널리 사용되고 있다.

용어정리

- 클러스터(군집): 서로 유사한 레코드들의 집합
- 클러스터 평균: 한 클러스터 안에 속한 레코드들의 평균 벡터 변수
- K: 클러스터의 개수

K 평균은 데이터를 K개의 클러스터로 나눈다. 이때 할당된 클러스터의 **평균**과 포함된 데이터들의 거리 제곱합이 최소가 되도록 한다. 이를 **클러스터 내 제곱합** 또는 **클러스터 내 SS** 라고 한다. K 평균은 각 클러스터의 크기가 동일하다는 보장은 없지만 클러스터들끼리는 최대한 멀리 떨어지도록 한다.

NOTE_ 정규화

데이터 값에서 평균을 빼고 그 편차를 표준편차로 나눠주는 방법이 가장 일반적인 정규화(표준화) 방법이다. 이렇게 하지 않으면 스케일이 가장 큰 변수가 클러스터링 과정을 독점하게 된다.

7.2.1 간단한 예제

변수가 x, y 두 개이고 레코드가 n 개인 데이터가 있다고 하자. $K = 4$, 즉 4개의 클러스터로 데이터를 분할하고 싶다고 가정하자. 즉 각 레코드 (x_i, y_i) 를 클러스터 k 에 n_k 개의 레코드가 들어 있다고 할 때, 클러스터 중심 (\bar{x}_k, \bar{y}_k) 는 클러스터 내에 존재하는 점들의 평균을 의미한다.

$$\begin{aligned}\bar{x}_k &= \frac{1}{n_k} \sum_{i \in \text{클러스터 } k} x_i \bar{y}_k \\ &= \frac{1}{n_k} \sum_{i \in \text{클러스터 } k} y_i\end{aligned}$$

CAUTION_ 클러스터 평균

여러 변수가 존재하는 레코드들을 클러스터링할 때, 클러스터 평균이라는 것은 하나의 값(스칼라)이 아닌 각 변수들의 평균으로 이뤄진 벡터를 의미한다.

클러스터 내부의 제곱합은 다음과 같이 구할 수 있다.

$$SS_i = \sum_{i \in \text{클러스터 } k} (x_i - \bar{x}_k)^2 + (y_i - \bar{y}_k)^2$$

K 평균은 4개의 모든 클러스터의 내부 제곱합 $(SS_1 + SS_2 + SS_3 + SS_4)$ 이 최소가 되도록 레코드들을 클러스터에 할당하는 방법이다.

$$\sum_{k=1}^4 SS_i$$

주가 변동이 어떤 식으로 클러스터를 이루는지 알아보기 위해 K 평균 클러스터링을 사용할 수 있다. 주식 수익률은 이미 표준화된 방식으로 보고되므로 데이터를 따로 정규화할 필요가 없다. R에서 K 평균 클러스터링을 실행하려면 kmeans 함수를 사용한다. 예를 들어 다음은 엑스모빌(XOM)과 셰브론(CVX)의 주가 수익률을 변수로 놓고 4개의 클러스터로 분류하는 예제이다.

```
set.seed(1010103)
df <- sp500_data[row.names(sp500_data)>='2011-01-01', c('XOM', 'CVX')]
km <- kmeans(df, centers=4, nstart=1)
```

각 레코드에 대해 할당된 클러스터 정보는 cluster 요소에서 찾아볼 수 있다.

```
> df$cluster <- factor(km$cluster)
> head(df)
           XOM      CVX cluster
2011-01-03 0.73680496 0.2406809      1
2011-01-04 0.16866845 -0.5845157      4
2011-01-05 0.02663055 0.4469854      1
2011-01-06 0.24855834 -0.9197513      4
2011-01-07 0.33732892 0.1805111      1
2011-01-10 0.00000000 -0.4641675      4
> km
```

제일 처음에 나오는 6개의 레코드들은 클러스터 1 아니면 2에 할당된 것을 볼 수 있다. 클러스터의 평균 역시 확인할 수 있다.

```
> centers <- data.frame(cluster=factor(1:4), km$centers)
> centers
  cluster      XOM      CVX
1       1 -1.1439800 -1.7502975
2       2  0.2410159  0.3342130
3       3  0.9568628  1.3708892
4       4 -0.3284864 -0.5669135
```

```
ggplot(data=df, aes(x=XOM, y=CVX, color=cluster, shape=cluster)) +
  geom_point(alpha=.3) +
  scale_shape_manual(values = 1:4,
                     guide = guide_legend(override.aes=aes(size=1))) +
  geom_point(data=centers, aes(x=XOM, y=CVX), size=2, stroke=2) +
  theme_bw() +
  scale_x_continuous(expand=c(0,0), lim=c(-2, 2)) +
  scale_y_continuous(expand=c(0,0), lim=c(-2.5, 2.5))
```

클러스터 2,4는 하락장이며, 클러스터 1,3은 상승장을 의미한다. 위의 예제에서 변수가 두 개이기에 클러스터와 그 평균을 다음과 같이 쉽게 시각화가 가능하다.

실행결과를 보면 위의 그림과 같다. 데이터들이 클러스터에 할당된 결과와 클러스터의 평균을 보여준다.

7.2.2 K평균 알고리즘

일반적으로 K 평균은 p 개의 변수 X_1, X_2, \dots, X_p 를 갖는 데이터에 적용될 수 있다. K 평균의 정확한 해를 계산하기는 매우 어려우므로, 휴리스틱한 방법을 통해 국소 최적화된 해를 효과적으로 계산한다.

사용자가 미리 정해진 K 값과 클러스터 평균의 초깃값을 가지고 알고리즘을 시작하며, 아래 과정을 반복한다.

1. 각 레코드를 거리가 가장 가까운 평균으로 귀결시키는 클러스터에 할당한다.
2. 새로 할당된 레코드들을 가지고 새로운 클러스터 평균을 계산한다.

각 레코드에 대한 클러스터 할당이 더 이상 변화하지 않을 때 알고리즘이 수렴했다고 볼 수 있다. 첫 번째 단계에서 클러스터 평균의 초깃값을 설정할 필요가 있다. 보통은 각 레코드를 K 개의 클러스터들 가운데 하나에 랜덤하게 할당한 후 그렇게 얻은 클러스터들의 평균을 사용한다.

이 방법이 항상 최적의 답을 준다는 보장이 없기에, 랜덤하게 초깃값을 변화시켜가며 알고리즘을 여러 번 돌려봐야 한다. 일단 초깃값이 정해지면 반복 루프를 통해 K 평균은 클러스터 내 제곱합이 최소가 되도록 하는 해를 얻는다.

R에서 kmeans 함수의 nstar 변수를 이용해 랜덤하게 초깃값을 다르게 설정해 알고리즘을 시행할 횟수를 설정할 수 있다. 예를 들어 다음은 5개의 클러스터를 찾기 위해 K 평균 알고리즘을 서로 다른 초깃값을 이용해 10번 수행하는 코드이다.

```
## cluster means algorithm
syms <- c( 'AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM', 'SLB', 'COP',
           'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST')
df <- sp500_data[row.names(sp500_data)>='2011-01-01', syms]

set.seed(10010)
km <- kmeans(df, centers=5, nstart=10)
```

10번의 시도 가운데 가장 좋은 성능의 결과를 볼 수 있다. iter.max 변수를 이용해 각 초기 설정별 알고리즘의 최대 반복 횟수를 설정할 수 있다.

7.2.3 클러스터 해석

클러스터 분석에서 가장 중요한 부분은 바로 클러스터를 바르게 해석하는 것이다. **kmeans 함수에서 가장 중요한 두 출력은 바로 클러스터의 크기와 클러스터 평균이다.** 바로 앞에서 다루었던 예제에서 구한 클러스터 크기는 다음과 같다.

```
> km$size
[1] 288 266 285 106 186
> centers <- km$centers
```

클러스터의 크기가 비교적 균일하다. 유난히 균형이 맞지 않는 클러스터가 존재한다면 이는 아주 멀리 떨어진 특잇점이 있거나 아니면 어떤 레코드 그룹이 나머지 데이터로부터 아주 멀리 떨어져 있다는 것을 의미한다. 따라서 이런 경우 좀 더 자세히 들여다볼 필요가 있다.

ggplot 함수와 gather 함수를 잘 이용해 클러스터의 중심을 그래프화 할 수 있다.

```
centers <- as.data.frame(t(centers))
names(centers) <- paste("Cluster", 1:5)
centers$Symbol <- row.names(centers)
centers <- gather(centers, "Cluster", "Mean", -Symbol)

centers$Color = centers$Mean > 0
ggplot(centers, aes(x=Symbol, y=Mean, fill=Color)) +
  geom_bar(stat='identity', position = "identity", width=.75) +
  facet_grid(Cluster ~ ., scales='free_y') +
  guides(fill=FALSE) +
  ylab('Component Loading') +
  theme_bw() +
  theme(axis.title.x = element_blank(),
        axis.text.x = element_text(angle=90, vjust=0.5))
```

아래 결과 그래프는 각 클러스터의 특징을 잘 보여준다. 예를 들어 클러스터 2,3,4,5 에서 클러스터 2,4 클러스터는 주식시장이 내린날, 3,5는 주식시장이 오른날을 보여준다.

NOTE_ 클러스터 평균 그래프가 주성분분석(PCA)에서 봤던 부하 그래프와 매우 비슷하다. PCA와 가장 다른 점은 클러스터 평균에서는 부호가 매우 중요한 의미를 갖는다는 점이다. PCA에서는 변동성의 주요 방향을 찾는 것이 목적이었다면, 클러스터 분석에서는 서로 가까운 위치의 레코드들의 그룹을 찾는 것이 목적이다.

7.2.4 클러스터 개수 선정

K 평균 알고리즘을 사용하려면 클러스터의 개수를 지정해야 한다. 클러스터 개수는 적용 문제에 따라 결정되는 경우가 있다. 예를 들어, 판매 부서가 있는 회사에서는 판매 상담 전화를 필요로 하는 고객들을 특별히 따로 분류해 집중 관리하고자 한다. 이럴 경우 관리를 위한 고려 사항에 따라 고객들을 분류할 그룹의 개수를 미리 결정할 수 있다. 예를 들어 2개의 그룹은 고객들을 제대로 분류하기에 모자라고 8개의 그룹은 관리하기에 너무 많다고 정할 수 있다.

실무 혹은 관리상의 고려 사항에 따라 클러스터 개수를 미리 결정하기가 어려울 경우, 통계적 접근 방식을 사용할 수 있다. 다만 '최상의' 클러스터 개수를 찾는 딱 한 가지 표준화된 방법은 없다.

팔꿈치 방법은 언제 클러스터 세트가 데이터 분산의 '대부분'을 설명하는지를 알려준다. 여기에 새로운 클러스터를 더 추가하면 분산에 대한 기여도가 상대적으로 작아진다. 즉 이는 누적 분산이 가파르게 상승한 다음 어느 순간 평평하게 되는 지점을 말하며 이러한 성질 때문에 팔꿈치라는 이름이 붙었다.

```
## Figure 7-6: selecting the number of clusters (elbow plot)
pct_var <- data.frame(pct_var = 0,
                      num_clusters=2:14)
totalss <- kmeans(df, centers=14, nstart=50, iter.max = 100)$totss
for(i in 2:14){
  pct_var[i-1, 'pct_var'] <- kmeans(df, centers=i, nstart=50,
    iter.max = 100)$betweenss/totalss
}

ggplot(pct_var, aes(x=num_clusters, y=pct_var)) +
  geom_line() +
  geom_point() +
  labs(y='% Variance Explained', x='Number of Clusters') +
  scale_x_continuous(breaks=seq(2, 14, by=2)) +
  theme_bw()
```

위 그림은 2에서 15까지의 클러스터 개수에 따른 기본 데이터에 대해 설명된 분산의 누적 백분율을 보여준다. 이 예제에서 팔꿈치의 위치는 어디일까? 여기서의 분산 증가율이 서서히 떨어지기에 눈에 띄는 위치는 없다. 이것은 잘 정의된 클러스터가 없는 데이터에서 상당히 일반적이다. 이는 팔꿈치 방법의 단점이라고도 할 수 있지만, 데이터의 특성을 밝혀준다는 점에서 가치가 있다.

R의 kmeans 함수는 팔꿈치 방법을 사용하기 위한 특별한 파라미터를 제공하지는 않지만 다음과 같이 kmeans의 출력을 이요해 쉽게 적용할 수 있다.

유지할 클러스터 개수를 평가할 때 가장 중요한 테스트는 다음과 같다.

클러스터들이 새로운 데이터에서도 그대로 유지될 가능성이 얼마나 있을까?

클러스터는 해석 가능한가?

데이터의 일반적인 특성과 관련이 있는가?

아니면 특정 데이터만 반영하는가?

교차타당성검사를 사용하면 이러한 것들을 부분적으로 평가할 수 있다.

Note_클러스터 개수를 결정하는 데에는 통계 이론이나 정보이론에 바탕을 둔 좀 더 형식적인 방법들도 있다.

주요개념

- 사용자가 원하는 클러스터 개수 K 를 선택한다.
- K 평균 알고리즘은 더 이상 클러스터가 변하지 않을 때까지 반복해서 클러스터 평균이 가장 가까운 클러스터에 레코드를 할당한다.
- 실무적인 고려 사항을 활용해 K 를 선택하는 것이 가장 일반적이다. 통계적으로 최적의 클러스터 개수를 구하는 방법은 없다.

7.3 계층적 클러스터링

계층적 클러스터링은 K 평균 대신 사용하는 클러스터링 방법으로 K 평균과는 아주 다른 결과를 보여준다. K 평균보다 유연하고, 수치형 변수가 아니어도 쉽게 적용이 가능하다. 특이점이나 비정상적인 그룹이나 레코드를 발견하는 데 민감하다. 계층적 클러스터링은 또한 직관적인 시각화가 가능하여 클러스터를 해석하기가 수월하다.

용어정리

- 덴드로그램 : 레코드들, 그리고 레코드들이 속한 계층적 클러스터에 대한 시각적 표현
- 거리 : 한 레코드가 다른 레코드들과 얼마나 가까운지를 보여주는 측정 지표
- 비유사도 : 한 클러스터가 다른 클러스터들과 얼마나 가까운지를 보여주는 측정지표

계층적 클러스터링의 유연성에는 비용이 따른다. 다시 말해서 계층적 클러스터링은 수백만 개의 레코드가 있는 대규모 데이터에는 적용할 수 없다. 수만 개의 레코드로 이뤄진 적당한 크기의 데이터의 경우에도 계층적 클러스터링을 위해서는 상대적으로 많은 컴퓨팅 리소스가 필요할 수 있다.

사실, 계층적 클러스터링은 대부분 상대적으로 데이터 크기가 작은 문제에 주로 적용된다.

7.3.1 간단한 예제

n 개의 레코드와 p 개의 변수가 있는 일반적인 데이터에 대해 계층적 클러스터링을 사용할 수 있으며 아래 두 가지 기본 구성 요소를 기반으로 한다.

- 두 개의 레코드 i 와 j 사이의 거리를 측정하기 위한 거리 측정 지표 $d_{i,j}$
- 두 개의 클러스터 A 와 B 사이의 차이를 측정하기 위한, 각 클러스터 구성원 간의 거리 $d_{i,j}$ 를 기반으로 한 비유사도 측정 지표 $D_{i,j}$

수치형 데이터와 관련된 응용 분야의 경우, 가장 중요한 것은 비유사도 지표이다. 계층적 클러스터링은 각 레코드 자체를 개별 클러스터로 설정하여 시작하고 가장 가까운 클러스터를 결합해 나가는 작업을 반복한다. R에서는 `hclust` 함수를 사용하여 계층적 클러스터링을 수행할 수 있다. `hclust`와 `kmeans` 함수 사이의 한 가지 큰 차이점은 데이터 자체보다 쌍 거리 $d_{i,j}$ 에 따라 동작한다는 점이다. `dist` 함수를 사용하여 이 거리를 계산할 수 있다. 예를 들어 다음은 계층적 클러스터링을 여러 기업의 주가 수익에 적용하는 과정을 보여준다.

```
## hclust chapter

syms1 <- c('GOOGL', 'AMZN', 'AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX',
           'XOM', 'SLB', 'COP', 'JPM', 'WFC', 'USB', 'AXP',
           'WMT', 'TGT', 'HD', 'COST')

df <- sp500_data[row.names(sp500_data)>='2011-01-01', syms1]
d <- dist(t(df))
hcl <- hclust(d)
```

클러스터링 알고리즘은 데이터 프레임의 레코드(행) 들을 클러스터링한다. 수익이 유사한 기업들을 서로 묶는 것이 목적이므로 데이터 프레임을 **전치**해서 행을 따라 기업별 주가 정보가 오게 하고 열을 따라 날짜가 오도록 해야한다.

7.3.2 덴드로그램

계층적 클러스터링은 트리 모델과 같이 자연스러운 시각적 표현이 가능하며, 이를 **덴드로그램**이라고 한다. 이 이름은 나무를 뜻하는 그리스 단어 dendro와 그림을 뜻하는 단어 gramma에서 유래했다.

R에서는 plot 명령을 사용하여 쉽게 이를 생성할 수 있다.

```
par(cex=.75, mar=c(0, 5, 0, 0)+.1)
plot(hcl, ylab='distance', xlab='', sub='', main='')
```

위 코드의 결과이다. 트리의 잎은 각 레코드를 의미한다. 트리의 가지 길이는 해당 클러스터 간의 차이 정도를 나타낸다. 구글과 아마존에 대한 수익률은 다른 주식에 대한 수익률과 상당히 다른 것을 볼 수 있다. 다른 주식들은 자연스럽게 그룹을 형성한다. 에너지 금융, 소매 기업들이 각각 하위 트리로 구분된 것을 볼 수 있다.

K 평균과 달리 클러스터의 수를 미리 지정할 필요 없다. *cutree* 함수를 사용하면 원하는 개수의 클러스터를 추출할 수 있다.

```
> cutree(hcl, k=4)
```

GOOGL	AMZN	AAPL	MSFT	CSCO	INTC
1	2	3	3	3	3
CVX	XOM	SLB	COP	JPM	WFC
4	4	4	4	3	3
USB	AXP	WMT	TGT	HD	COST
3	3	3	3	3	3

추출할 클러스터의 개수를 4로 설정했으며 구글과 아마존은 각각 자신만의 독자적인 클러스터에 속해 있음을 볼 수 있다. 석유 주식(XOM, CVX, SLB, COP)은 모두 또 다른 한 클러스터에 속한다. 마지막으로 나머지 주식들이 남은 하나의 클러스터에 속한다.

7.3.3 병합 알고리즘

계층적 클러스터링에서 가장 중요한 알고리즘은 바로 **병합** 알고리즘이다. 유사한 클러스터들을 반복적으로 병합하는 역할을 한다. 병합 알고리즘은 단일 레코드로 구성된 클러스터에서 시작하여 점점 더 큰 클러스터들을 만든다.

첫 번째 단계는 모든 레코드 쌍 사이의 거리를 계산하는 것이다.

각 레코드 쌍 $(x_1, x_2, \dots, x_p), (y_1, y_2, \dots, y_p)$ 에 대해 거리 지표를 사용하여 두 레코드 사이의 거리 $d_{x,y}$ 를 측정한다. 예를 들어 다음과 같이 유클리드 거리를 사용할 수 있다.

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_p - y_p)^2}$$

이제 클러스터 간의 거리를 알아보자. $A = (a_1, a_2, \dots, a_m)$ 과 $B = (b_1, b_2, \dots, b_q)$ 라는 두 개의 클러스터 A와 B를 고려하자. 클러스터 A의 구성원들과 B의 구성원들 사이의 거리를 사용하여 클러스터 간의 $D(A, B)$ 비유사도를 측정할 수 있다.

비유사도를 측정하는 한 가지 방법은 A와 B 사이의 모든 레코드 쌍의 최대 거리를 사용하는 **완전연결** 방식이다.

$$D(A, B) = \max d(a_i, b_j)$$

모든 i, j 쌍에 대해 이것은 비유사도를 모든 쌍 사이의 가장 큰 차이로 정의한다.

병합 알고리즘의 주요 단계는 다음과 같다.

1. 데이터의 모든 레코드에 대해, 단일 레코드로만 구성된 클러스터들로 초기 클러스터 집합을 만든다.
2. 모든 쌍의 클러스터 k, l 사이의 비유사도 $D(C_k, C_l)$ 을 계산한다.
3. $D(C_k, C_l)$ 에 따라 가장 가까운 두 클러스터 C_k 와 C_l 을 병합한다.
4. 둘 이상의 클러스터가 남아 있으면 2단계로 다시 돌아간다. 그렇지 않고 클러스터가 하나 남는다면 알고리즘을 멈춘다.

7.3.4 비유사도 측정

비유사도를 측정하는 네 가지 일반적인 지표는 **완전연결**, **단일연결**, **평균연결**, **최소분산**이다. 이외에 다른 다양한 비유사도 지표들이 **hclust**를 포함한 대부분의 계층적 클러스터링 SW에서 모두 지원된다. 앞서 소개한 완전연결 방법은 비슷한 멤버가 있는 클러스터를 만드는 경향이 있다. 단일연결 방법은 두 클러스터의 레코드 간의 최소 거리를 사용하는 방식이다.

$$D(A, B) = \min d(a_i, b_j)$$

모든 i, j 쌍에 대해 이는 탐욕적인 방법이며, 결과로 나온 클러스터는 서로 크게 다른 요소들을 포함하는 일도 생길 수 있다.

평균연결 방법은 모든 거리 쌍의 평균을 사용하는 방법으로 이는 단일연결과 완전연결 방법 사이를 절충한 방법이다.

마지막으로 **워드 기법**이라고 하는 최소 분산 방법은 클러스터 내의 제곱합을 최소화하므로 K 평균과 유사하다고 할 수 있다.

다음은 엑스모빌과 셰브런 주가 수익 데이터에 위의 네 가지 방법을 적용하여 계층적 클러스터링을 수행한 결과이다. 각 방법에 대해 네 개씩 클러스터를 표시했다.

 image-20200501134910343

현저하게 다른 결과를 볼 수 있다. 단일연결 방법은 거의 모든 점을 하나의 클러스터에 할당했다. 최소분산 방법(ward.D)을 제외한 다른 방법은 외곽에 요소가 몇 개 없는 특잇점들로 이뤄진 작은 클러스터를 만들었다. 최소분산 방법은 K 평균 클러스터와 가장 유사한 결과를 보인다.

주요개념

- 모든 레코드를 각각 자체 클러스터로 할당하여 알고리즘을 시작한다.
- 클러스터들은 모든 레코드가 하나의 클러스터에 속할 때까지 가까운 클러스터와 계속해서 연결한다(병합 알고리즘)
- 병합 과정은 내역이 남고 시각화 할 수 있으며, 사용자가 미리 클러스터 수를 지정하지 않더라도 여러 단계에서 클러스터의 수와 구조를 시각화 할 수 있다.
- 클러스터 간 거리는 모든 레코드 간 거리 정보를 사용하여 여러 가지 다른 방식으로 계산할 수 있다.

7.3 모델 기반 클러스터링

계층적 클러스터링과 K 평균 같은 클러스터링 방법들은 모두 휴리스틱한 방법이라고 할 수 있으며, 직접 관측한 (즉, 확률모형에 기반하지 않고) 데이터들이 서로 가깝게 있는 클러스터를 찾는 데 주로 사용된다. 연구자들은 지난 20년간 **모델 기반 클러스터링**을 개발하는 데 많은 노력을 기울였다. 그 결과에 이드리언 래프터리는 통계 이론에 기초하고 있으며 클러스터의 성질과 수를 결정하는 더 엄격한 방법을 개발했다.

예를 들면 전반적으로는 서로 비슷하지만, 모든 데이터가 반드시 서로 가까울 필요는 없는 그룹(예를 들어, 수익의 분산이 큰 기술 주식들)과 서로 비슷하면서 데이터들이 아주 가까이에 있는 또 다른 그룹(예를 들어 분산이 적은 유틸리티 주식)이 함께 있는 경우에 사용할 수 있다.

7.4.1 다변량정규분포

가장 널리 사용되는 대부분의 모델 기반 클러스터링 방법은 모두 **다변량정규분포**를 따른다. 다변량정규분포는 p 개의 변수 집합 X_1, X_2, \dots, X_p 에 대해 정규분포를 일반화한 것이다. 분포는 평균 집합 $\mu = \mu_1, \mu_2, \dots, \mu_p$ 과 공분산행렬 Σ 로 정의된다. 공분산행렬은 변수가 서로 어떻게 상호 관련되어 있는지를 나타내는 지표이다. 공분산행렬 Σ 는 p 개의 분산 $\sigma_1^2, \sigma_2^2, \dots, \sigma_p^2$ 과 $i \neq j$ 인 모든 변수 쌍에 대한 공분산 $\sigma_{i,j}$ 로 구성된다. 행과 열을 따라 구성된 공분산행렬은 다음과 같다.

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} & \cdots & \sigma_{1,p} \\ \sigma_{2,1} & \sigma_2^2 & \cdots & \sigma_{2,p} \\ & & \ddots & \\ & & & \sigma_p^2 \end{bmatrix}$$

공분산행렬은 $\sigma_{i,j} = \sigma_{j,i}$ 인 대칭행렬이므로 결과적으로 $p \times (p-1) - p$ 개의 공분산 항이 존재한다고 볼 수 있다. 전체적으로 공분산행렬은 $p \times (p-1)$ 개의 변수를 갖는다. 이 분포를 다음과 같이 표시한다.

$$(X_1, X_2, \dots, X_p) \tilde{N}_p(\mu, \Sigma)$$

이 기호는 변수들이 모두 정규분포를 따른다는 뜻이며, 전체 분포가 변수의 평균 벡터와 공분산행렬에 의해 완벽히 설명된다는 것을 나타낸다.

다음 그림은 X 와 Y 에 대한 다변량정규분포의 확률 등고선을 보여준다. (확률 등고선에서 예를 들어 0.5는 분포의 50%를 포함한다는 뜻이다).

평균은 $\mu_x = 0.5$ 와 $\mu_y = -0.5$ 이고 공분산행렬은 다음과 같다.

$$\Sigma = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$

공분산 σ_{xy} 가 양수이므로 X 와 Y 는 양의 상관관계가 있다.

2차원 정규분포에 대한 확률 등고선


7.4.2 정규혼합

모델 기반 클러스터링의 핵심 아이디어는 각 레코드가 K 개의 다변량정규분포 중 하나로부터 발생했다고 가정하는 것이다. 여기서 K 가 클러스터의 개수를 의미한다. 각 분포는 서로 다른 평균 μ 와 공분산행렬 Σ 를 갖는다. 예를 들어 X 와 Y 라는 두 개의 변수가 있는 경우 각 행 (X_i, Y_i) 은 K 개의 분포 $N_2(\mu_1, \Sigma_1), N_2(\mu_2, \Sigma_2), \dots, N_2(\mu_k, \Sigma_k)$ 중 하나에서 샘플링된 것으로 모델링된다.

R에서는 모델 기반 클러스터링을 위해 mclust 패키지를 사용한다. 이전에 K 평균과 계층적 클러스터링을 적용하여 분석했던 주가 수익 데이터에 이 패키지를 사용하면 모델 기반 클러스터링을 적용할 수 있다.

```
df <- sp500_data[row.names(sp500_data)>='2011-01-01', c('XOM', 'CVX')]  
> mcl <- Mclust(df)  
fitting ...  
|=====| 100%  
> summary(mcl)  
-----  
Gaussian finite mixture model fitted  
by EM algorithm  
-----  
  
Mclust VEE (ellipsoidal, equal shape  
and orientation) model with 2  
components:  
  
log-likelihood    n df      BIC  
-2255.125 1131  9 -4573.528  
ICL  
-5075.657  
  
Clustering table:  
  1  2  
168 963
```

이 코드를 실행하면 다른 기법들보다 계산 시간이 훨씬 오래 걸리는 것을 볼 수 있다. predict 함수를 사용하여 할당된 클러스터 정보를 얻을 수 있고 이를 이용하면 다음과 같은 클러스터를 시각화 할 수 있다.

 image-20200501151442590

```
cluster <- factor(predict(mcl)$classification)  
png(filename=file.path(PSDS_PATH, 'figures', 'psds_0710.png'),  
width = 5, height=4, units='in', res=300)  
  
ggplot(data=df, aes(x=XOM, y=CVX, color=cluster, shape=cluster)) +  
  geom_point(alpha=.8) +  
  theme_bw() +  
  scale_shape_manual(values = c(46, 3),  
                      guide = guide_legend(override.aes=aes(size=2)))
```

결과는 위와 같다. 두 개의 클러스터가 있는 것을 볼 수 있다. 하나는 데이터 중심에 있고 하나는 데이터 중심을 둘러싼 외곽에 존재한다. 이것은 앞서 K 평균이나 계층적 클러스터링을 사용하여 얻은 작고 조밀한 클러스터와는 매우 다르다.

summary 함수를 사용하여 각 정규분포의 파라미터를 추출할 수도 있다.

```
> summary(mcl, parameters=TRUE)$mean  
      [,1]      [,2]  
XOM -0.04362218 0.05792282  
CVX -0.21109525 0.07375447  
> summary(mcl, parameters=TRUE)$variance
```

, , 1

	XOM	CVX
XOM	1.044671	1.065190
CVX	1.065190	1.912748

, , 2

	XOM	CVX
XOM	0.2998935	0.3057838
CVX	0.3057838	0.5490920

두 분포는 비슷한 평균과 상관관계를 갖고 있지만, 첫 번째 분포의 분산과 공분산이 훨씬 큰 것을 알 수 있다.

mclust 함수를 통해 얻은 클러스터링 결과가 놀라울 수도 있지만 실제로는 이 방법이 갖고 있는 통계적 특성을 잘 보여주는 결과이다. 모델 기반 클러스터링의 목적은 사실 데이터를 가장 잘 설명하고 다변량 정규분포를 찾는 것이다. 위 등고선 그림을 보면 주식 데이터는 정규분포 모양을 갖고 있는 것 처럼 보인다. 사실 더 정확히는, 주식 수익률은 정규분포보다 긴 꼬리 분포를 따른다고 볼 수 있다. 이를 처리하기 위해 mclust는 대량의 데이터에 대해 분포를 피팅하려고 하고 그 결과 첫 번째 분포가 더 큰 분산을 갖도록 피팅된다.

7.4.3 클러스터 개수 결정하기

K 평균이나 계층적 클러스터링과 달리 *mclust*는 클러스터 수(이 경우 2)를 자동으로 선택한다.

바로 **베이지스 정보기준(BIC)** 값이 가장 큰 클러스터의 개수를 선택하도록 동작하기 때문이다. **BIC(AIC와 유사)**는 후보가 될 만한 모델 집합 중에서 가장 좋은 모델을 찾는 일반적인 방법이다.

예를 들어 AIC(또는 BIC)는 일반적으로 단계별 회귀모형을 선택하는 데 사용된다. BIC는 모델의 파라미터 개수에 대해 벌점을 주는 방식으로 가장 적합한 모델을 선택한다. **모델 기반 클러스터링의 경우 클러스터를 추가하면 할수록 모델의 파라미터 개수가 증가되는 대신 항상 적합도는 좋아진다.**

hclust의 함수를 통해 각 클러스터 크기에 대해 BIC값을 그릴 수 있다.

```
par(mar=c(4, 5, 0, 0)+.1)
plot(mcl, what='BIC', ask=FALSE, cex=.75)
```

클러스터의 개수 또는 다변량정규분포 모델(구성 요소)의 개수는 x 축에 표시된다.

이 그래프는 K 평균에서 클러스터 개수를 선택하는 데 사용했던 팔꿈치 그림과 유사하지만, 여기서는 분산 대신 BIC가 사용됐다. 한 가지 큰 차이점은 선이 하나가 아닌, mclust 함수가 무려 14개의 다른 선을 동시에 보여준다는 것이다!!!!!!

이는 mclust가 실제로 각 클러스터 크기에 대해 14개의 다른 모델을 피팅했고 궁극적으로 가장 적합한 모델을 선택했다는 것을 잘 보여준다.

mclust 함수는 최상의 다변량정규분포를 결정하기 위해 왜 그렇게 많은 모델들을 피팅하는 것일까?

적합한 모델을 찾기 위해 공분산행렬 Σ 를 모수화하는 여러가지 방법이 있기 때문이다.

일반적으로 이 모델들에 대한 자세한 사항을 굳이 알 필요는 없으며, 단순히 mclust가 선택한 모델을 사용하면 된다. 이 예제에서 BIC에 따르면 (VEE,VEV,VVE라는) 세 가지 모델이 두 가지 구성요소를 사용할 때 가장 적합하다.

Note_ 모델 기반 클러스터링은 실제로 매우 활발하고 빠르게 발전하는 연구분야이다.

모델 기반 클러스터링 기술에는 몇 가지 한계가 있다.

이 방법은 기본적으로 데이터들이 모델을 따른다는 가정이 필요하며,

클러스터링 결과는 이 가정에 따라 매우 다르다.

필요한 계산량 역시 계층적 클러스터링보다 높으므로 대용량 데이터로 확장하기가 어렵다.

마지막으로 알고리즘이 다른 방법들보다 더 복잡하고 이용하기 힘들다.

주요 개념

- 클러스터들이 각자 서로 다른 확률분포로부터 발생한 것으로 가정한다.
- 분포(일반적으로 정규분포) 개수에 대한 가정에 따라 서로 다른 적합한 모델이 있다.
- 이 방법은 너무 많은 파라미터(오버피팅의 원인이 될 수 있다)를 사용하지 않으면서도 데이터에 적합한 모델(그리고 연관된 클러스터의 개수)을 선택한다.

7.5 스케일링과 범주형 변수

비지도 학습 기술을 이용할 때는 일반적으로 데이터를 적절하게 스케일해야 한다. 스케일링이 중요하지 않았던 회귀나 분류 방법들과는 차이가 있다(물론 K 최근접 이웃 알고리즘은 예외였다)

용어 정리

- 스케일링 : 데이터의 범위를 높이거나 줄이는 방식으로 여러 변수들이 같은 스케일에 오도록 하는 것
- 정규화 : 원래 변수 값에서 평균을 뺀 후에 표준편차를 나누는 방법으로, 스케일링의 일종이다.(유의어 : 표준화)
- 고위 거리 : 수치형과 범주형 데이터가 섞여 있는 경우에 모든 변수 0~1 사이로 오도록 하는 스케일링 방법

예를 들어 개인 신용 대출 데이터의 변수들은 단위나 규모 면에서 서로 크게 다르다. 어떤 변수는 상대적으로 작은 값(예를 들어 근속 연수)인 반면 다른 변수는 매우 큰 값(예를 들어 달러로 표기된 대출 금액)을 갖는다. 데이터의 크기가 조정되지 않으면 PCA, K 평균, 혹은 기타 클러스터링 방법은 큰 값을 갖는 변수들에 의해 결과가 좌우되고 작은 값을 갖는 변수들은 무시된다.

범주형 데이터는 일부 클러스터링 과정에서 특별한 문제를 일으킬 수 있다. KNN에서와 마찬가지로, 순서가 없는 요인변수는 일반적으로 원-핫 인코딩을 사용하여 이진(0/1) 변수 집합으로 변환한다.

이러한 이진변수는 다른 데이터와 스케일이 다를 뿐만 아니라 PCA나 K평균 같은 기법을 사용할 때 이진 변수가 두 가지의 값만 가질 수 있다는 것 때문에 문제가 될 수 있다.

7.5.1 변수 스케일링

매우 다른 스케일 및 단위를 갖는 변수는 클러스터링 절차를 적용하기 전에 적절히 정규화해야 한다.

이를 알아보기 위해 대출 데이터를 정규화하지 않고 바로 kmeans 함수를 적용해보자.

```
> defaults <- loan_data[loan_data$outcome=='default',]
> df <- defaults[, c('loan_amnt', 'annual_inc', 'revol_bal',
'open_acc', 'dti', 'revol_util')]
> km <- kmeans(df, centers=4, nstart=10)
> centers <- data.frame(size=km$size, km$centers)
> round(centers, digits=2)
  size loan_amnt annual_inc revol_bal open_acc  dti revol_util
1    52  22570.19  489783.40  85161.35   13.33  6.91    59.65
2 13902  10606.48   42500.30  10280.52    9.59 17.71    58.11
3   7525  18282.25   83458.11  19653.82   11.66 16.77    62.27
4   1192  21856.38  165473.54  38935.88   12.61 13.48    63.67
```

두 변수 annual_inc와 revol_bal이 클러스터링 결과를 좌우하며 클러스터마다 크기가 매우 다른 것을 볼 수 있다. 클러스터 1에는 비교적 높은 손익과 높은 회전 신용 잔고를 가진 단 52명의 데이터만 포함되어 있다.

변수를 스케일링 하는 일반적인 방법은 평균을 빼고 표준편차로 나눔으로써 원래 값을 z점수로 변환하는 것이다. 이를 표준화 또는 정규화라고 한다.

정규화된 데이터에 kmeans를 적용했을 때, 클러스터링 결과에 어떤 변화가 생기는지 살펴보자.

```
> df0 <- scale(df)
> km0 <- kmeans(df0, centers=4, nstart=10)
> centers0 <- scale(km0$centers, center=FALSE,
  scale=1/attr(df0, 'scaled:scale'))
> df0 <- scale(df)
> km0 <- kmeans(df0, centers=4, nstart=10)
> centers0 <- scale(km0$centers, center=FALSE,
  scale=1/attr(df0, 'scaled:scale'))
> centers0 <- scale(centers0, center=-attr(df0, 'scaled:center'),
  scale=FALSE)
> centers0 <- data.frame(size=km0$size, centers0)
> round(centers0, digits=2)
  size loan_amnt annual_inc revol_bal open_acc   dti revol_util
1 3713  25894.07  116185.91  32797.67   12.41 16.22    66.14
2 7355  10467.65   51134.87  11523.31    7.48 15.78    77.73
3 6294  13361.61   55596.65  16375.27   14.25 24.23    59.61
4 5309  10363.43   53523.09   6038.26    8.68 11.32    30.70
```

클러스터들의 크기가 좀 더 균일하여 앞선 결과와 달리 두 변수 annual_inc와 revol_bal에 의해 큰 영향을 받지 않고, 같은 데이터로부터 더 흥미로운 구조를 보여준다. 이 코드에서 클러스터 중심은 원래 단위로 재조정하였다. 값을 재조정하지 않고 그대로 둔다면 결과는 z점수로 표시되므로, 해석하기가 어려워진다.

Note_ 스케일링은 PCA에서도 역시 중요하다. z점수를 사용하는 것은 주성분을 계산할 때 공분산행렬 대신 상관행렬을 사용하는 것과 같은 결과를 낳는다. PCA를 계산하는 SW에는 일반적으로 상관행렬을 사용할 수 있는 옵션이 있다.(R에서는 princomp 함수에 cor라는 인수가 있다)

7.5.2 지배 변수

변수들이 서로 동일한 규모로 측정되고 상대적 중요성을 정확하게 반영하는 경우 (예를 들어 주가 변동)조차도 변수의 스케일을 재조정하는 것이 유용할 수 있다.

```
syms <- c('GOOGL', 'AMZN', 'AAPL', 'MSFT', 'CSCO', 'INTC', 'CVX', 'XOM',
  'SLB', 'COP', 'JPM', 'WFC', 'USB', 'AXP', 'WMT', 'TGT', 'HD', 'COST')
top_15 <- sp500_px[row.names(sp500_px)>='2011-01-01', syms]
sp_pca1 <- princomp(top_15)
screplot(sp_pca1)
```

7.1 절에서 설명한 것 처럼 스크리그래프는 첫 번째 주성분에 대한 분산을 표시한다. 이 경우, 다음 그림을 보면 스크리그래프에서 첫 번째와 두 번째 구성 요소의 분산이 다른 구성 요소 보다 훨씬 큰 것을 볼 수 있다. 이는 하나 혹은 두 개의 변수가 전체 부하량을 지배한다는 것을 나타낸다. 실제로 이 예제가 바로 이런 경우이며 다음 코드로 확인할 수 있다.

```
> round(sp_pca1$loadings[,1:2], 3)
      Comp.1 Comp.2
GOOGL  0.781  0.609
AMZN   0.593 -0.792
```


AAPL	0.078	0.004
MSFT	0.029	0.002
CSCO	0.017	-0.001
INTC	0.020	-0.001
CVX	0.068	-0.021
XOM	0.053	-0.005
SLB	0.079	-0.013
COP	0.044	-0.016
JPM	0.043	0.001
WFC	0.034	-0.001
USB	0.026	0.003
AXP	0.063	-0.006
WMT	0.026	-0.001
TGT	0.036	-0.010
HD	0.051	-0.019
COST	0.061	-0.019

처음 두 가지 주성분은 GOOGL과 AMZN에 의해 거의 완전히 지배되고 있다. 이는 GOOGL과 AMZN의 주가 움직임이 전체 변동성의 대부분을 지배하기 때문이다. 이러한 상황에서는 변수를 스케일링해서 포함하거나, 이러한 지배 변수를 전체 분석에서 제외하고 별도로 처리할 수 있다. 어떤 방법이 항상 옳다고 할 수 없으며 응용 분야에 따라 달라진다.

7.5.3 범주형 데이터와 고위 거리

범주형 데이터가 있는 경우에는 순서형(정렬된 요인)변수 또는 이진형(더미) 변수를 사용하여 수치형 데이터로 변환해야 한다. 데이터를 구성하는 변수들에 연속형과 이진형 변수가 섞여 있는 경우에는 비슷한 스케일이 되도록 변수의 크기를 조정해야 한다. 이를 위한 대표적인 방법은 **고위 거리**를 사용하는 것이다.

고위 거리의 기본 아이디어는 각 변수의 데이터 유형에 따라 거리 지표를 다르게 적용하는 것이다.

- 수치형 변수나 순서형 요소에서 두 레코드 간의 거리는 차이의 절댓값(**맨하튼 거리**)으로 계산한다.
- 범주형 변수의 경우 두 레코드 사이의 **범주가 서로 다르면 거리가 1**이고 **범주가 동일하면 거리는 0**이다.

고위 거리는 다음과 같이 계산한다.

1. 각 레코드의 변수 i 와 j 의 모든 쌍에 대해 거리 $d_{i,j}$ 를 계산한다.
2. 각 $d_{i,j}$ 의 크기를 최솟값이 0이고 최댓값이 1이 되도록 스케일을 조정한다.
3. 거리 행렬을 구하기 위해 변수 간에 스케일된 거리를 모두 더한 후 평균 혹은 가중평균을 계산한다.

대출 데이터 일부를 이용해 고위 거리를 자세히 알아보자.

```
> x <- loan_data[1:5, c('dti', 'payment_inc_ratio', 'home_', 'purpose_')]
> x
```

	dti	payment_inc_ratio	home_	purpose_
1	1.00	2.39320	RENT	major_purchase
2	5.55	4.57170	OWN	small_business
3	18.08	9.71600	RENT	other
4	10.08	12.21520	RENT	debt_consolidation
5	7.06	3.90888	RENT	other

cluster 패키지의 daisy 함수를 사용하면 고위 거리를 계산할 수 있다.

```
> library(cluster)
> daisy(x, metric='gower')
Dissimilarities :
      1      2      3      4
2 0.6220479
3 0.6863877 0.8143398
4 0.6329040 0.7608561 0.4307083
5 0.3772789 0.5389727 0.3091088 0.5056250

Metric : mixed ; Types = I, I, N, N
Number of objects : 5
```

거리가 모두 0과 1 사이인 것을 볼 수 있다. 가장 거리가 먼 레코드 쌍은 2번과 3번 레코드이다. 이 둘은 home_ 변수나 purpose 변수에 대해서도 값이 다르며 dti 변수와 payment_inc_ratio 변수 역시 차이가 크다. 반면 home_ 변수나 purpose 변수가 동일한 값을 갖는 레코드 3번과 5번은 거리가 가장 작다.

daisy 함수의 거리 행렬 결과에 hclust 함수를 이용하여 계층적 클러스터링을 적용해볼 수 있다.

```
set.seed(301)
df <- loan_data[sample(nrow(loan_data), 250),
                    c('dti', 'payment_inc_ratio', 'home_', 'purpose_')]
d = daisy(df, metric='gower')
hcl <- hclust(d)
dnd <- as.dendrogram(hcl)

png(filename=file.path(PSDS_PATH, 'figures', 'psds_0713.png'),
     width = 4, height=4, units='in', res=300)
par(mar=c(0,5,0,0)+.1)
plot(dnd, leaflab='none', ylab='distance')
dev.off()
```

아래 그림은 위 코드를 실행한 덴드로그램의 결과이다. x축을 따라 표시된 개별 레코드들을 식별하기가 매우 어렵다.



다음 코드를 사용하면 하위 트리 중 하나(cut을 이용해 0.5를 기준으로 잘랐을 때 가장 왼쪽에 있는 그룹)에 포함된 레코드들을 살펴볼 수 있다.

```
> dnd_cut <- cut(dnd, h=.5)
> df[labels(dnd_cut$lower[[1]]),]
      dti payment_inc_ratio home_ purpose_
1447  10.19          20.2918 MORTGAGE small_business
31477 13.02          20.2257 MORTGAGE credit_card
```

이 하위 트리에 속한 레코드들은 small_business, credit_card에 대한 목적이고, 주거 형태는 mortgage이다.

모든 하위 트리에서 이러한 형태의 구분이 이뤄지는 것은 아니지만 이는 범주형 변수값이 비슷한 데이터들이 한 클러스터로 그룹화되는 경향이 있다는 것을 보여준다.

7.5.4 혼합 데이터의 클러스터링 문제

K평균과 PCA는 연속형 변수에 가장 적합하다. 데이터 집합의 크기가 더 작아질수록 고위 거리를 사용하여 계층적 클러스터링을 하는 것이 좋다. 원칙적으로 이진형 혹은 범주형 데이터에도 K평균을 적용할 수 있다. 범주형 데이터의 경우에는 일반적으로 원-핫 인코딩 방법을 이용해 수치형으로 변환할 수 있다. 하지만 실무에서는 K평균과 PCA를 이진형 데이터와 함께 사용하는 것은 어려울 수 있다.

표준 z점수를 사용할 경우, 이진형 변수가 클러스터 결과에 지대한 영향을 미친다. 0/1 변수의 경우, 0 또는 1의 값인 레코드를 모두 한 클러스터에 포함되므로 K평균에서 클러스터 내 제곱합이 작아지기 때문이다. 예를 들어 요인 변수 home과 pub_rec_zero를 포함하는 대출 데이터에 kmeans를 적용해보자.

```
> ## Problems in clustering with mixed data types
> df <- model.matrix(~ -1 + dti + payment_inc_ratio + home_ + pub_rec_zero,
data=defaults)
> df0 <- scale(df)
> km0 <- kmeans(df0, centers=4, nstart=10)
> centers0 <- scale(km0$centers, center=FALSE, scale=1/attr(df0,
'scaled:scale'))
> round(scale(centers0, center=-attr(df0, 'scaled:center'), scale=FALSE), 2)
      dti payment_inc_ratio home_MORTGAGE home_OWn home_REnt pub_rec_zero
1 16.99             9.11           0.00         0       1.00         1.00
2 17.20             9.27           0.00         1       0.00         0.92
3 17.46             8.42           1.00         0       0.00         1.00
4 16.50             8.06           0.52         0       0.48         0.00
attr(,"scaled:scale")
      dti payment_inc_ratio home_MORTGAGE home_OWn
0.1305561 0.2286345 2.0190809 3.6191450
      home_REnt pub_rec_zero
2.0008117 3.5722842
attr(,"scaled:center")
      dti payment_inc_ratio home_MORTGAGE home_OWn
-17.1521684 -8.7700843 -0.4313440 -0.0832782
      home_REnt pub_rec_zero
-0.4853778 -0.9142958
```

상위 4개 클러스터가 요인변수들과 밀접한 관련이 있다는 것을 볼 수 있다. 이러한 문제를 피하기 위해 이진형 변수의 크기를 다른 변수들보다 작은 값으로 조정할 수 있다. 아니면 데이터의 크기가 아주 큰 경우에는 특정 범주 값들에 따라 서로 다른 하위 집합에 클러스터링을 적용 할 수도 있다. 예를 들어, 주택 담보 대출이 있는 사람인지, 택을 완전히 소유한 사람인지, 주택을 임대하고 있는 사람인지에 따라 대출 데이터를 쪼개서 각각의 하위 그룹에 클러스터링을 개별적으로 적용할 수 있다.

주요 개념

- 스케일이 서로 다른 변수들을 스케일이 비슷하도록 변환하여, 스케일이 알고리즘에 큰 영향을 미치지 않도록 한다.
- 일반적인 스케일링 방법은 각 변수에서 평균을 빼고 표준편차로 나눠주는 정규화(표준화)방법이다.
- 또 다른 방법은 고위 거리를 사용하는 것이다. 이 방법은 모든 변수를 0~1 범위로 스케일링한다(수치형과 범주형 데이터가 서로 혼란된 경우에 많이 사용된다.)

7.6 마치며

주성분분석과 K평균 클러스터링은 수치형 데이터의 차원을 축소하기 위해 주로 사용되는 방법들이다. 의미 있는 데이터 축소를 보장하기 위해서는 데이터의 스케일을 적절히 조정해야 한다.

그룹들 간의 구분이 분명하고 고도로 구조화된 데이터를 클러스터링할 경우, 어떤 방법을 사용하든 결과는 비슷할 가능성이 높다. 물론 방법마다 장점이 있다. K평균은 매우 큰 데이터로 확장이 가능하고 이해하기 쉽다. 계층적 클러스터링은 수치형과 범주형이 혼합된 데이터 유형에 적용이 가능하며 직관적인 시각화 방법(덴드로그램)이 존재한다. 모델 기반 클러스터링은 휴리스틱한 방법들과 달리 통계 이론에 기초를 두고 있으며 더 엄밀한 접근 방식을 제시한다. 그러나 데이터가 커지면 K평균이 가장 많이 사용된다.

대출이나 주식 데이터, 그리고 데이터 과학자가 직면할 대다수의 데이터는 노이즈가 많다. 이럴 경우, 사용 기법에 따라 결과에 극명한 차이를 부를 수 있다. K평균, 계층적 클러스터링, 그리고 특히 모델 기반 클러스터링은 매우 다른 솔루션을 생성한다. 데이터 과학자는 이럴 때 어떻게 해야 할까? 불행하게도 선택을 돕는 간단한 법칙 따위는 없다. 궁극적으로 데이터 크기나 응용 분야의 목표에 따라 사용되는 방법은 다르기 때문이다.