10. 생성 모형

10.1. 확률의 기본 개념

랜덤 변수 X와 Y가 각각 어젯밤 날씨와 오늘 아침 현관 앞의 상태를 나타낸다고 생각해보자.

- X = 0: 어젯밤에 비가 옴
- X = 1: 어젯밤에 맑음
- Y = 0: 현관 앞이 뽀송뽀송
- Y = 1 현관 앞이 축축

이때 P(X)는 어젯밤 날씨의 확률 분포이고, P(Y)는 현관 앞 상태의 확률 분포이다. 이렇게 한 변수에 대한 확률 분포는 주변 확률 분포(marginal probability distribution)라고도 한다.

또한 P(X, Y)는 결합 확률 분포(joint probability distribution)라고 한다. 예를 들어 P(X = 0, Y = 0)은 어젯밤에 비가 왔고, 오늘 아침 현관은 뽀송뽀송할 확률을 나타낸다.

결합 확률 분포에서 한쪽 변수를 모든 경우의 수에 대해 더해주면 주변 확률 분포가 된다. '주변'이라는 말은 확률 분포를 표 형태로 나타냈을 때 합계가 주변에 위치하기 때문에 그렇게 부른다.

$$P(X) = \sum_Y P(X, Y)$$

결합 확률과 혼동하기 쉬운 것이 조건부 확률 분포(conditional probability distribution)이다. P(X|Y)는 현관 앞의 상태가 주어졌을 때 어젯밤 날씨의 조건부 확률 분포이고, P(Y|X)는 어젯밤 날씨가 주어졌을 때 현관 앞 상태의 조건부 확률 분포이다.

조건부 확률과 결합 확률의 관계는 아래 식과 같다.

$$P(Y|X) = \frac{P(X, Y)}{P(X)}$$

다음의 식도 성립한다.

$$P(X|Y) = \frac{P(X, Y)}{P(Y)}$$

그러면 다음과 같은 식을 간단히 유도할 수 있다. 이를 베이즈 정리(Bayes' theorem)이라 한다.

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

10.2. 생성 모형과 구분 모형

우리가 지도학습에서 이제까지 다룬 모형은 대개 X가 주어졌을 때 Y를 예측하는 형태의 모형이었다. 확률 분포로 말하면 P(Y|X)의 형태이다. 이러한 모형을 구분 모형(discriminative model)이라고 부른다. 구분모형은 조건부 확률이기 때문에 X는 주어진 것으로 간주한다. 예를 들어 광화문, 강남, 여의도까지 대중 교통으로 5분 내에 출근할 수 있는 한강이 내려다보이는 30평대 아파트 X가 있다고 하면 그 가격 Y를 예측하는 식이다. 그런 아파트가 실제로 있는지는 따지지 않는다.

반면 X와 Y의 결합 확률 형태 P(X, Y)의 모형은 생성 모형(generative model)이라고 부른다. 결합 확률을 알고 있다면 X, Y를 표본 추출하여 '생성'하는 것이 가능하기 때문에 그런 이름이 붙었다. 예를 들어 두 주사위의 결합 확률 분포를 알고 있으면 실제로 던져보지 않아도 주사위를 던지면 나올 수 있는 결과들을 만들어낼 수 있다.

생성 모형은 결합 확률이기 때문에 모든 Y에 대해서 P(X, Y)를 더하면 주변 확률 P(X)를 구할 수 있고, 주변 확률을 구하면 조건부 확률 P(Y|X)도 유도할 수 있다.

또 만약 X_1과 X_2 중에 어느 한쪽이라도 알 수 없으면 $P(Y|X_1, X_2)$는 계산할 수 없지만 $P(Y, X_1, X_2)$가 있다면 모든 X_2에 대해 주변화해서 $P(Y, X_1)$을 구할 수 있다.

또한 생성 모형은 이름 그대로 '생성'을 할 수 있는 모형이므로 어떤 사진이 고양이인지 강아지인지 구분하는 것을 넘어서 고양이처럼 보이는 사진, 강아지처럼 보이는 사진을 컴퓨터가 그리게 할 수도 있다.

즉, 이론적으로 생성 모형은 구분 모형이 할 수 있는 것보다 더 많은 것을 할 수있다. 물론 현실적으로 생성 모형은 만들기도 어렵고 계산도 어렵다. P(X, Y)에서 Y의 종류가 많으면 주변 확률을 구하기가 계산이 어렵다(computationally infeasible). 또, 생성의 경우에도 우리가 보통 컴퓨터로 생성할 수 있는 확률 분포는 균등 분포(uniform distribution) 뿐이다. 다른 분포의 경우 누적 확률 분포를 이용해서 균등 분포로 추출한 값을 변환한다. 누적 확률 분포도 경우의 수가 많아지면 계산이 어려워진다.

이번 강의에서는 생성 모형의 간단한 사례들을 다뤄보도록 하자.

10.3. 최대 우도법

생성 모형의 비교적 간단한 형태는 이미 알려진 확률 분포를 사용하는 것이다. 일단은 변수가 하나인 경우만 생각해보자. 연속변수의 경우 가장 잘 알려진 분포는 정규 분포이다. 정규 분포는 평균과 분산을 알면 분포의 형태가 정해진다. 그렇다면 평균과 분산을 어떻게 정해야할까?

가장 많이 사용되는 방법은 최대우도법(Maximum Likelihood)이다. 어떤 확률 분포 P가 파라미터 θ에 따라 결정이 된다면 $P(X; \theta)$를 우도라 한다. 이때 이 우도를 최대로 만드는 θ를 찾는 것이 최대우도법이다.

왜 최대우도법을 쓸까? $P(X; \theta)$는 일종의 조건부 확률 $P(X|\theta)$로 볼 수 있다. 베이즈 정리에 대입해보면 아래와 같다.

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$$

그러면 데이터 X가 주어졌을 때 파라미터 θ의 확률 분포 $P(\theta|X)$는 다른 조건이 같다면 우도 $P(X|\theta)$에 비례한다는 것을 알 수 있다. 즉, 우리가 이

미 관찰한 데이터의 확률을 높이는 파라미터는 주어진 데이터에 대해 참일 확률이 높은 파라미터이기도 하다. 현실적으로 P(θ)나 P(X)를 구하기는 어려우므로 우도를 가장 높이는 파라미터가 가장 좋은 파라미터로 보는 것이다.

10.3.1. 정규분포의 최대우도법

이제 정규분포의 평균과 분산을 추정해보자. 있단은 R에서 다음 명령으로 평균이 1, 표준편차는 2인 정규분포에서 100개의 값을 만들자.

```
x = rnorm(100, 1, 2)
```

우리는 x의 실제 분포를 알고 있지만, 마치 그 분포를 모르는 것처럼 평균과 분산을 찾아가보겠다. 일단은 정규분포라는 가정은 있으므로 평균과 분산만 알면 확률을 구할 수 있다. 일단은 평균은 0, 표준편차는 1이라고 해보자.

```
m = 0
s = 1
p = dnorm(x, m, s)
```

이제 p를 모두 곱하면 우도 $P(X;\theta)$를 구할 수 있다. 그런데 확률은 1보다 작은 수이므로 이를 여러 번 곱하면 너무 작은 수가 된다. 그래서 우도 대신 로그 우도를 많이 쓴다. 왜냐하면 우도의 곱은 로그 우도의 합과 같기 때문이다.

```
loglike = sum(log(p))
```

이제 다음은 간단하다. m과 s을 바꿔가면서 로그 우도를 구하고, 로그 우도가 최대화되는 m과 s를 찾으면 된다. 이것은 전형적인 최적화 문제로 경사하강법을 이용해서 풀 수 있다. R에는 경사하강법을 제공하는 optim 함수가 있다.

```
initial = c(0, 1) # 초기값: 평균 0, 표준편차 1
fn = function(p){ # 마이너스 로그 우도 계산 함수
    m = p[1]
    s = p[2]
    -sum(log(dnorm(x, m, s)))
}

res = optim(initial, fn)
```

optim 함수는 여러 가지 알고리즘을 사용하는데 기본적으로 Nelder-Mead 법을 이용해서 함수의 미분을 직접 주지 않아도 그 근사값을 추정하는 방법을 사용해서 경사하강법을 실시한다.

마이너스 로그 우도를 계산하는 이유는 optim 함수가 최소화를 하기 때문이다. 로그 우도는 최대화를 해야 하기 때문에 마이너스 로그 우도를 최소화 시키는 방법을 쓴다. 위에서 구한 결과에서 res$par가 최종적으로 구해진 파라미터이다. 실제 파라미터와는 조금 차이가 있을텐데 왜냐하면 표본 오차가 있기 때문이다.

여기서 최대우도법으로 추정된 평균과 표준편차는 표본평균과 표본표준편차와 동일하다. 실제로 정규분포를 미분해서 경사가 0이 되는 파라미터를 유도하면 표본평균과 표본표준편차의 공식을 얻게 된다.

10.4. 커널 밀도 추정

최대우도법은 우리가 잘 알고 있는 확률 분포가 아니면 사용하기 어렵다. 이런 경우에 사용할 수 있는 방법 중에 하나가 가우시안 혼합 모형(GMM)이다. GMM은 전체 데이터의 분포를 몇 개의 정규 분포를 혼합시켜 근사시킨다. GMM을 좀 더 극단적으로 밀고 나간 아이디어가 커널 밀도 추정(kernel density estimation: KDE) 이다.

KDE는 데이터 갯수만큼의 정규 분포를 만들어서 그 분포를 모두 더하는 방법으로 전체 분포를 추정한다. KDE는 GMM과 달리 각 데이터가 어떤 분포에서 나왔는지를 고려할 필요가 없으므로 계산이 매우 간단하다. 대신 데이터 하나 당 분포가 하나가 되므로 표준편차를 직접 구할 수가 없다.

여러 가지 휴리스틱(heuristic: 정확한 답은 아니지만 그럴듯한 답을 어림으로 구하는 방법)을 통해 모든 분포의 표준편차를 하나로 정해준다. 흔히 쓰이는 간단한 휴리스틱은 KDE로 구한 전체 분포가 정규 분포에 가까운 형태가 되도록 만드는 것이다.

R에는 커널 밀도 추정을 위한 함수 density가 내장되어 있다.

```
y = c(rnorm(50, 0, 1), rnorm(50, 5, 1))  # 평균이 0과 5인 정규분포에서 각각 50개씩 추출
d = density(y)  # y를 바탕으로 커널 밀도 추정
plot(d)  # 추정된 밀도를 그래프로 그림
```

10.5. Fully Visible Belief Network

최대우도법이나 커널 밀도 추정의 문제는 다차원에서 복잡한 분포를 추정하기에는 잘 맞지 않는다는 것이다. 특히 연속 변수와 이산 변수가 섞여 있을 경우 이를 나타내는 확률 분포를 만들기가 어렵다. 이럴 때는 확률 분포를 여러 개로 쪼개면 문제를 좀 더 간단하게 만들 수 있다.

그 전에 먼저 확률의 연쇄 규칙을 알아보자. 앞이서 보았던 결합 확률과 조건부 확률의 관계를 다르게 쓰면 아래와 같이 쓸 수 있다.

$$P(X_1, X_2) = P(X_1|X_2)P(X_2)$$

만약 변수가 3개라면 다음과 같은 식이 된다.

$$P(X_1, X_2, X_3) = P(X_1|X_2, X_3)P(X_2, X_3) = P(X_1|X_2, X_3)P(X_2|X_3)P(X_3)$$

즉 3차원의 확률 분포 $P(X_1, X_2, X_3)$는 3개의 확률 분포 $P(X_1|X_2, X_3)$, $P(X_2|X_1)$, $P(X_3)$로 쪼갤 수 있다. 이를 활용한 모형이 Fully Visible Belief Network (FVBN)이다.

$P(X_3)$은 한 변수의 분포이므로 최대우도법이나 커널 밀도 추정을 써서 추정할 수 있다. 그리고 $P(X_2|X_3)$을 추정하고, 다시 $P(X_1|X_2, X_3)$를 추정하는 식으로 순차적으로 확률을 추정해나가는 것이다.

10.5.1. iris 데이터를 사용하여 FVBN 만들기

FVBN은 여러 가지 방법으로 구현해볼 수 있다. 여기서는 간단히 선형 모형을 이용해서 구현하도록 하자. 데이터는 `iris` 데이터를 사용하겠다.

FVBN은 어떤 순서로 확률을 추정하느냐에 따라 다른 결과를 줄 수 있다. 여기서는 단순히 `iris`의 컬럼 순서대로 추정해보자. `iris`의 컬럼은 `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`, 그리고 `Species` 순이다.

먼저 `Sepal.Length`의 분포를 추정한다. 이는 단순히 정규 분포로 추정하자.

```
m1 = mean(iris[,1])
s1 = sd(iris[,1])
```

다음은 `Sepal.Width`다. 이는 `Sepal.Length`를 통해서 조건부 확률로 구할 수 있다.

```
m2 = lm(Sepal.Width ~ Sepal.Length, iris)  # Sepal.Length로 Sepal.Width를 예측한다
s2 = sigma(m2)    # 잔차 표준편차를 구한다
```

이제 `Petal.Length`와 `Petal.Width`에 대해서도 위의 과정을 반복한다.

```
m3 = lm(Petal.Length ~ Sepal.Width + Sepal.Length , iris)
s3 = sigma(m3)

m4 = lm(Petal.Width ~ Petal.Length + Sepal.Width + Sepal.Length , iris)
s4 = sigma(m4)
```

`Species`는 이산형 변수이다. softmax 회귀분석을 하면 된다. 우리는 이를 신경망을 다룰 때 배웠다. `nnet` 패키지의 `multinom`를 사용하면 단층으로 간단히 할 수 있다.

```
library(nnet)
m5 = multinom(Species ~ ., iris)
```

이제 모든 확률 분포를 추정했으므로 실제 데이터를 바탕으로 확률을 계산해보도록 하자. `iris`의 첫 번째 데이터는 다음과 같다.

```
> iris[1,]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
```

먼저 `Sepal.Length`가 5.1일 확률밀도는 `m1`과 `s1`을 이용해 구할 수 있다.

```
> p1 = dnorm(iris[1,1], m1, s1)
> p1
[1] 0.3220059
```

다음으로 `Sepal.Width`가 3.5일 확률밀도를 구해보자. 먼저 `m2`를 이용해 `Sepal.Length`로부터 `Sepal.Width`의 평균값을 예측한다.

```
> predict(m2, iris[1,])
       1
3.103334
```

3.103334이다. 그러면 다음과 같이 확률밀도를 구할 수 있다.

```
> p2 = dnorm(iris[1,2], predict(m2, iris[1,]), s2)
[1] 0.6053071
```

`Petal.Length`와 `Petal.Width`도 위의 과정을 반복한다.

```
> p3 = dnorm(iris[1,3], predict(m3, iris[1,]), s3)
[1] 0.4866302
> p4 = dnorm(iris[1,4], predict(m4, iris[1,]), s4)
[1] 2.070747
```

이제 마지막으로 `Species`가 `setosa`일 확률을 구한다.

```
> p5 = predict(m5, iris[1,], 'probs')['setosa']
setosa
     1
```

이제 이 5가지의 확률(밀도)를 모두 곱하거나

```
> prod(c(p1, p2, p3, p4, p5))
[1] 0.1964109
```

또는 로그확률(밀도)를 모두 더할 수 있다.

```
> sum(log(c(p1, p2, p3, p4, p5)))
[1] -1.627546
```

10.5.2. FVBN을 이용한 생성

위의 과정을 응용하면 생성도 할 수 있다. 먼저 `Sepal.Length`를 무작위로 150개 생성하자.

```
gen = data.frame(Sepal.Length=rnorm(150, m1, s1))
```

다음으로 150개의 `Sepal.Length`로 150개의 `Sepal.Width`의 평균치를 예측하고 이를 바탕으로 무작위 추출을 한다.

```
library(purrr)
gen['Sepal.Width'] = map_dbl(      # 이하를 반복
    predict(m2, gen),                  # 평균을 예측
    function(x){ rnorm(1, x, s2) }     # 평균마다 무작위 추출 1회씩
)
```

역시 `Petal.Length`와 `Petal.Width`에도 위의 과정을 반복한다.

```
gen['Petal.Length'] = map_dbl(
    predict(m3, gen),
    function(x){ rnorm(1, x, s3) }
)

gen['Petal.Width'] = map_dbl(
    predict(m4, gen),
    function(x){ rnorm(1, x, s4) }
)
```

마지막으로 Species를 추출한다.

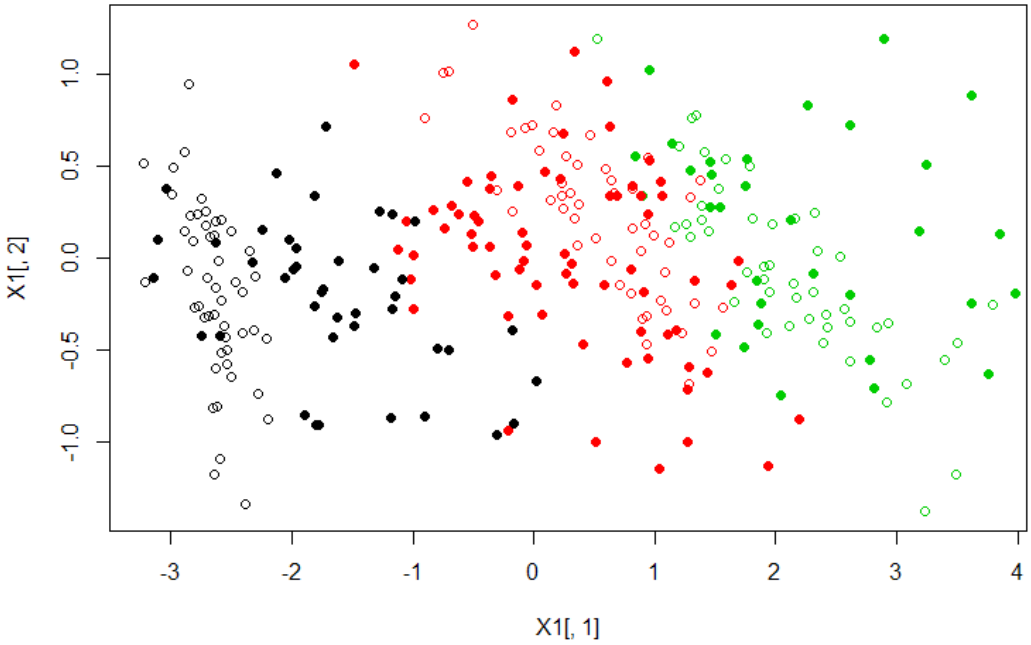
```
gen['Species'] = apply(                                 # 다음을 반복 적용
    predict(m5, gen, 'probs'),                          # 품종의 확률을 예측하고
    1,                                                  # 행 단위로 적용
    function(x){ sample(names(x), size=1, prob=x) } # 품종의 이름 중 1개를 확률에 따라 추출
)
```

이제 생성된 데이터와 원래 데이터를 시각화해서 비교해보자.

```
pc = prcomp(iris[1:4])  # 원래 데이터를 기준으로 PCA를 수행
species = c("setosa", "versicolor", "virginica")  # 품종명의 순서를 지정

X1 = predict(pc, iris[1:4])  # 원래 데이터를 변환
plot(X1[,1], X1[,2], col=factor(iris[,5], levels=species)) # PC1과 PC2로 산점도를 그린다

X2 = predict(pc, gen[1:4])  # 생성된 데이터를 변환
points(X2[,1], X2[,2], pch=16, col=factor(gen[,5], levels=species))  # 채운 원으로 산점도를 그린다
```



위의 실행 결과를 보면 속이 흰 색인 원이 실제 데이터, 속이 채워진 원이 생성된 데이터이다. 그리고 왼쪽부터 setosa, versicolor, virginica 품종이다. 생성된 데이터가 실제 데이터와 매우 비슷한 것을 볼 수 있다.

최근에는 FVBN의 각 단계에 딥러닝을 사용하는 연구들이 진행되고 있고 음성 합성(예: 딥마인드의 WaveNet)이나 이미지 생성 등에 활용한 사례들이 있다.

FVBN은 이론적으로 간단해서 쉽게 구현할 수 있다는 장점이 있는 반면에 한 번에 한 변수씩 순서대로 생성하기 때문에 속도가 느리다는 단점이 있다.

10.6. 나이브 베이즈

만약 X_1과 X_2가 독립이라면 $P(X_1|X_2, X_3) = P(X_1|X_3)$이 된다. FVBN에서 한 변수를 제외하고 다른 모든 변수들이 독립이라고 가정하면 나이브 베이즈가 된다.

$$P(X_1, X_2, Y) = P(X_1|Y)P(X_2|Y)P(Y)$$

나이브 베이즈는 보통 분류 문제에서 사용한다. X로 Y를 예측하는 구분 모형은 아래와 같이 된다.

$$P(Y|X_1, \ldots, X_n) = P(Y, X_1, \ldots, X_n)P(X_1, \ldots, X_n)$$

그런데 어차피 모든 Y에 대해서 분모 $P(X_1, \ldots, X_n)$는 동일하므로 굳이 계산할 필요가 없다.

$$P(Y|X_1, \ldots, X_n) \propto P(Y, X_1, \ldots, X_n)$$

나이브 베이즈에서는 X들 사이의 독립을 가정하므로 다음과 같이 간단히 전개된다.

$$P(Y|X_1, \ldots, X_n) \propto P(Y)P(X_1|Y)\ldots P(X_n|Y)$$

나이브 베이즈는 주로 텍스트 분류 등에서 자주 사용되고, FVBN과 같이 데이터 생성 등에는 잘 사용되지 않는다.

10.7. 다양한 생성 모형

위에서 소개한 것 외에도 은닉 마코프 모형(Hidden Markov Model), 잠재 디리클레 할당(Latent Dirichlet Allocation), 제한 볼츠만 머신(Restricted Boltzmann Machine) 등 다양한 생성 모형이 있다. 그중에서도 최근 딥러닝과 관련지어 많은 관심을 받고 있는 2가지 모형을 간단히 소개하겠다.

10.7.1. Variational Autoencoder

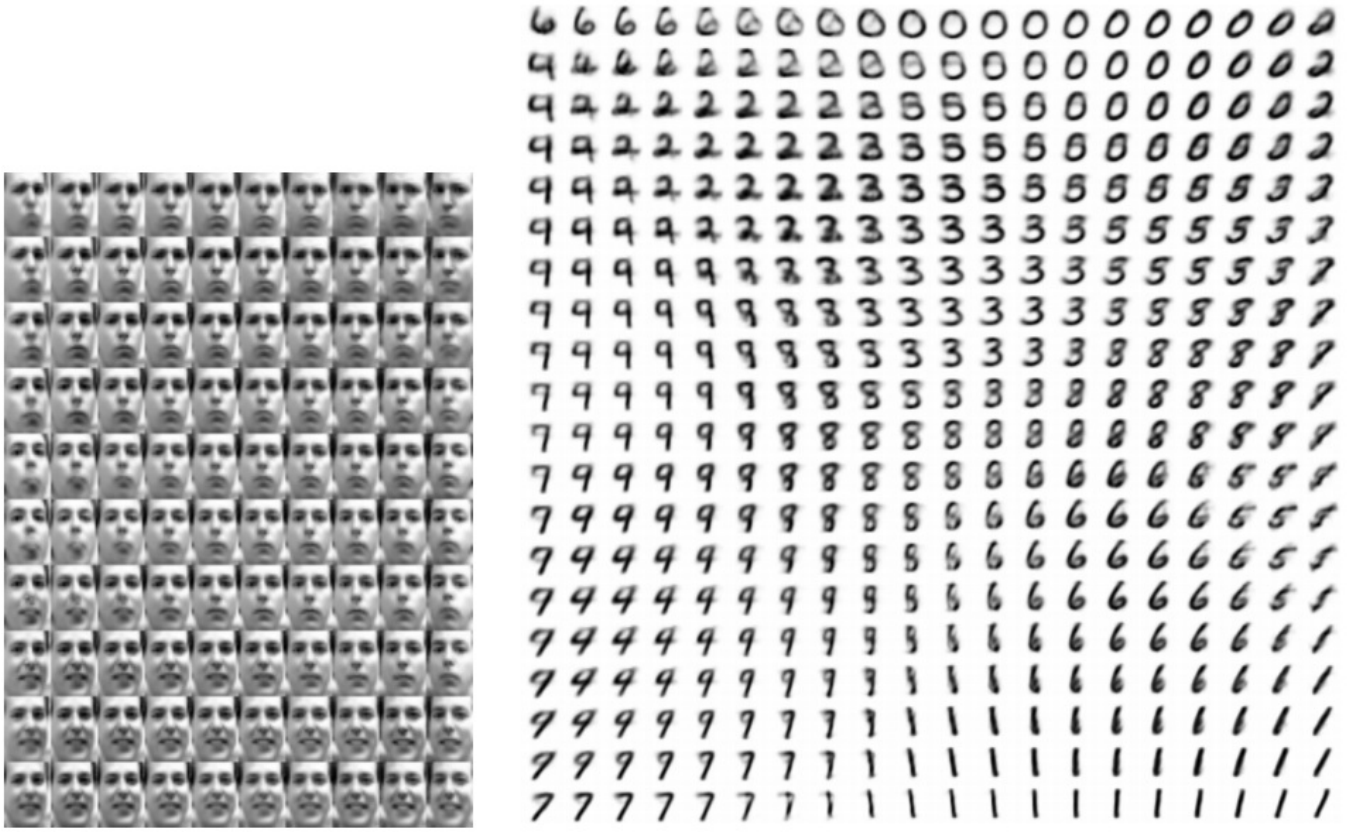
오토인코더(autoencoder)는 입력과 출력이 같은 인공신경망이다. 즉 X를 입력으로 넣으면 X가 그대로 출력으로 나오도록 만든 것이다. 이때 은닉층의 크기를 입력층보다 작게 하면, 입력층의 정보를 더 작은 차원에서 최대한 보존하는 방법을 학습하게 된다.

오토인코더는 두 부분으로 나눠진다. 먼저 인코더(encoder)로 입력층에서 은닉층까지다. 이 부분은 정보를 은닉층의 작은 차원에 넣는 부분이라고 볼 수있다. 다음은 디코더(decoder)로 은닉층에서 출력층까지다. 이 부분은 은닉층의 작은 차원에서 원래의 차원으로 복원하는 부분이라고 볼 수 있다. 여기서 만약 은닉층의 값들이 정규분포처럼 잘 알려진 분포를 따르도록 만들 수 있다면 정규분포에서 추출한 값을 디코더에 넣어 생성모형으로 사용할 수 있게 된다. 이것이 Variational Autoencoder(VAE)의 아이디어다.

실제로는 은닉층이 특정한 분포를 따르도록 만들기가 매우 어렵기 때문에 여러 가지 계산 트릭을 사용한다. 이를 간단히 말하면 다음과 같은 관계가 있는 계산하기 쉬운 함수 L을 찾아낸다.

$$L(X;\theta) \leq P(X;\theta)$$

여기서 L을 최대화하는 θ를 찾는다면 간접적으로 P도 최대화할 수 있게 된다.



(a) Learned Frey Face manifold (b) Learned MNIST manifold

10.7.2. Generative Adversarial Network

딥러닝과 관련된 생성 모형에서 최근 가장 높은 관심을 받고 있는 것은 Generative Adversarial Network(GAN)이다. adversarial은 '적대적인'이라는 뜻인데 원래는 학습된 신경망을 속이는 기법이나 데이터 등을 가리킨다. GAN은 이를 생성 모형에 확장한 것이다.

GAN의 아이디어 자체는 간단하다. GAN은 2개의 신경망으로 이뤄진다. 하나는 생성자(generator)이고 다른 하나는 구별자(discriminator)이다.

생성자는 데이터를 생성한다. 구별자에게는 생성된 데이터와 실제 데이터를 2가지 주고 이 둘을 구별하는 방법을 학습하게 한다. 생성자의 목적은 구별자를 속이는 것이고, 구별자의 목적은 속지 않는 것이다. 이 두 개의 네트워크를 경쟁시키면 생성자는 점점 더 구별자를 잘 속일 수 있는 그럴 듯한 데이터를 생성해내게 될 수도 있다.

10.8. 프로젝트: 결측값

결측값(missing value)란 데이터에서 어떤 사례에 특정한 변수의 값이 빠져있는 것이다. R에서는 `NA`로 표시한다(Not Available). 결측값은 데이터 마이닝에 흔히 있는 현상으로, 학습과 예측을 방해하는 요소이다.

10.8.1. 결측의 종류

결측의 종류에는 MAR, MCAR, MNAR 3가지가 있다.

- Missing Completeley at Random: 결측이 완전히 무작위적으로 발생했다. 결측된 이유가 데이터의 어떤 변수와도 관련이 없다. (예: 데이터를 적은 종이에 물을 쏟아서 일부 데이터가 지워졌다)
- Missing at Random: 결측이 발생한 이유가 결측된 변수와는 관련이 없지만 데이터의 다른 변수와는 관련이 있다. (예: 외국인 고객들은 만족도 설문에 응답을 하지 못했다)
- Missing Not at Random: 결측이 발생한 이유가 결측된 변수와 관련이 있다. (예: 서비스에 불만족한 고객들은 만족도 설문에 응답하지 않았다)

10.8.2. 결측값 다루기

결측값을 다루는 방법에는 삭제(deletion)와 대치(imputation)가 있다. 삭제는 말 그대로 결측이 발생한 데이터를 삭제하는 방법이고, 대치는 결측값을 다른 값으로 채워넣는 방법이다. 전체 분석은 가능한 데이터만을 가지고 분석을 실시하는 방법이다.

이외에도 모형이나 추정 방법을 바꾸는 방법이 있으나 모든 모형에 가능한 것도 아니고 방법도 매우 복잡하다.

10.8.3. 삭제

삭제에서 흔히 쓰이는 방법은 listwise 삭제이다. 이 방법은 결측값을 포함하는 사례를 데이터에서 모두 지워버리는 방법이다. 매우 간단한 방법이고 MCAR일 때는 학습에 문제를 일으키지 않는다. 그러나 MAR이나 MNAR에서는 분포를 왜곡시킬 수 있다.

R에서는 `na.omit` 함수를 사용해서 할 수 있다. 또한 `complete.cases` 함수는 데이터프레임의 각 행이 결측값을 포함하는지 알려준다.

```
df = data.frame(x = c(1, 2, NA), y = c(4, 5, 6))
na.omit(df)
complete.cases(df)
```

다음으로 상관이나 공분산을 계산할 때 pairwise 삭제를 하는 경우도 있다. 상관/공분산은 항상 두 변수의 관계만을 측정하기 때문에 두 변수 중에 결측값이 발생한 사례만을 삭제하면 좀 더 많은 데이터를 바탕으로 계산을 할 수 있다. 그러나 이렇게 구해진 상관/공분산은 데이터의 서로 다른 부분을 이용해서 구한 것이기 때문에 MCAR이 아닌 경우 실제 상관/공분산과 다를 수 있다.

```
mat = matrix(c(NA,2,3,4,5,NA,7,8,9,10,NA,12), ncol=3)
cov(mat, use='p') # pairwise
cov(mat, use='c') # listwise
```

마지막으로 결측이 발생한 변수는 모두 버리는 경우가 있다. 이 경우 분포를 왜곡시키는 문제는 없지만 해당 변수의 정보를 사용하지 못하는 문제가 있다. 또한 하나의 결측 사례만 있어도 변수 전체를 버리기 때문에 데이터를 크게 낭비하게 된다.

예측 상황에서 결측이 발생한 경우에는 삭제로는 해결책이 될 수 없다.

10.8.4. 대치

대치는 다른 값들을 바탕으로 결측값을 예측해서 채워넣는 방법이다.

`caret`에서는 `train` 함수에서 `preProcess` 옵션에 다음 값들 중 하나를 넣어주면 해당 방법으로 결측값을 대치한다.

- `knnImpute`: kNN을 이용해서 결측값을 대치한다.
- `bagImpute`: 랜덤포레스트와 유사한 방법을 이용한다
- `medianImpute`: 중간값으로 채운다.

R에는 대치를 수행해주는 여러 가지 패키지들이 있다. `missForest`는 random forest를 이용해 대치를 수행한다.

```
library(missForest)
df = prodNA(iris, .05)  # iris 데이터에서 5%의 결측값을 인위적으로 생성한다
imp.f = missForest(df)
cor(imp.f$ximp[1], iris[1])
```

대치는 채워진 값들로 인해 해당 변수의 분산이 작아질 수 있다는 문제가 있다. 이를 해결하기 위해 하나의 결측치에 하나의 값을 대치하는 대신, 하나의 결측치에 여러 값을 대치하는 다중 대치(multiple imputation)를 사용하기도 한다.

`mice`는 회귀분석 등을 이용해 다중대치를 지원하는 패키지이다.

```
library(mice)
imp.m = mice(df, m = 5)  # 5가지 대치 데이터를 생성한다
df1 = complete(imp.m, 1)  # 1번째 대치 데이터로 대치한다
df2 = complete(imp.m, 2)  # 1번째 대치 데이터로 대치한다
```

이렇게 여러 개의 데이터를 만들개 되면 이들을 합쳐서 하나의 데이터로 만들 거나, 각각의 데이터로 서로 다른 모형을 학습시켜 앙상블을 한다.

비슷한 패키지로 `Amelia`가 있다. `Amelia`는 전체 데이터를 바탕으로 하나의 다변량 정규분포를 추정하고, 이 분포를 이용해서 다중 대치를 수행한다.

```
library(Amelia)
imp.a = amelia(df, m = 5)
df1 = imp.a$imputations[[1]]
df2 = imp.a$imputations[[2]]
```