Data Mining

HW#1

학번	182STG18
이름	이하경
제출일	2019.03.20

Description

K-Nearest Neighborhood & Dimensionality

f(x)를 추정하고 싶을 때 가장 좋은 추정치는 $X = x$에서의 조건부 기댓값 $E(Y|X = x)$으로서 Squared Error Loss 를 최소화하는 추정치로 알려져 있다. 그러나 많은 경우 가진 Data 가 각각 x 에 대해 충분히 많은 자료점을 포함하고 있지는 않으므로, x 의 가장 가까운 k 개의 주변 관측치들을 이용해 f(x)를 추정하는 **K-NN**(K-Nearest Neighborhood) 방법을 자주 사용한다.

$$\hat{f}(x) = Average(y_i \mid x_i \in N_k(x)), \qquad N_k(x): \text{the Neighborhood containing k points in closest to x}$$

이는 각 관측치의 주위의 가장 가까운 k 개의 관측치에서의 y 값의 평균을 예측치로 사용하는 방법으로 포함된 모수의 개수가 적고 관측치의 수, 즉 Sample Size 가 크다면 효과적인 추정방법으로 사용할 수 있다. 그러나 고차원으로 갈수록 점과 점 사이의 거리가 매우 멀어져 K-NN 을 이용한 예측은 매우 불안정해진다. ▶ **Curse of Dimensionality**

관측치 X 들이 중심을 기준으로 반지름이 1 인 Unit Ball 에서 Uniformly Distributed 되어 있다고 할 때, X 가 반지름 r 인 Sphere (A)내에 위치할 확률은 전체 공간의 부피 중 해당 Sphere 의 부피가 차지하는 비율과 같으며 이 확률은 r^p 에 비례한다. * $P(X \in A) \propto r^p$

본 과제에서는 위와 같은 Unit Sphere 에서 중심으로부터 가장 가까운 점(the nearest-neighbor) 추정과 가장 가까운 점까지의 거리들의 중앙값 d(P, N)을 추정을 다룬다. 이를 위해 반지름이 1 인 Unit Sphere 내에서 자유롭게 점들을 Sampling 하고, 시뮬레이션을 반복하여 거리를 계산한다. 또한 median distance 의 정의를 활용해 공식을 증명하고 d(P, N)의 실제 값과 표본 생성을 통해 구한 추정값을 비교해본다.

Results

1. Derive the equation of median distance from the origin to the closest data point

$$\mathbf{d(P, N)} = \left(1 - \left(\frac{1}{2}\right)^{\frac{1}{N}}\right)^{\frac{1}{P}}$$

P-dimensional Unit Sphere 내부에 분포하고 있는 임의의 점 $X_i = (X_{i1}, \dots, X_{ip})$에 대해
$D_i = \{\text{distance from the origin to } X_i\}$, $\min(D_1, \dots, D_N) = M$이라고 하자.

즉 M 은 N 개의 점들 중 중심 O 에서 가장 가까운 점까지의 거리를 나타내는 Random Variable 이라고 할 수 있다. 따라서 M 이 특정 상수 r 보다 크다는 것은 중심에서 모든 N 개의 점까지의 거리가 r 보다 큼을 의미하며 확률은 다음과 같이 나타낼 수 있다.

$$P(M > r) = P(D_1, \dots, D_N > r) = \prod_{i=1}^{N} P(D_i > r) = \{P(D_i > r)\}^N = \{1 - P(D_i \le r)\}^N$$

여기서 $P(D_i \le r) = \int_{D<r} f_X(x)dx$ 이며 $f_X(x)$는 점 x 가 Unit Sphere 내의 특정 지점에 위치할 확률로서 X 가 Uniformly distributed 이므로 Volume of Unit Sphere 을 $K_p \cdot (1)^p = K_p$라고 했을 때 $f_X(x) = 1/K_p$이다. 따라서 위의 식을 다음과 같이 정리할 수 있다.

$$P(D_i \le r) = \frac{1}{K_p}\int_{D<r} dx = \frac{K_p(r)^p}{K_p} = r^p$$

$$P(M > d) = \{1 - P(D_i \le d)\}^N = \{1 - d^p\}^N \quad \cdots (1)$$

또한 r 이 M 의 중앙값이 되기 위해서는 다음과 같은 성질을 가진다.

$$P(M \le r) = F_M(r) = \frac{1}{2} \cdots (2)$$

여기서 $P(M \le r) = 1 - P(M > r)$이므로 (1)과 (2)의 식을 합하면 아래와 같은 수식을 얻을 수 있으며 이것이 곧 median distance from the origin to the closest data point 의 정의이다.

$$1 - \{1 - r^p\}^N = \frac{1}{2} \quad \rightarrow \quad 1 - r^p = \left(\frac{1}{2}\right)^{\frac{1}{N}}$$

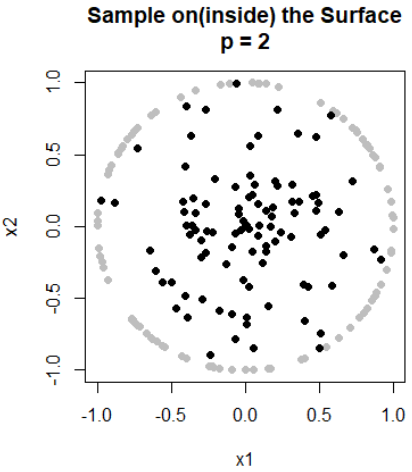
$$\therefore\ r = \left\{1 - \left(\frac{1}{2}\right)^{\frac{1}{N}}\right\}^{\frac{1}{p}} = d(P, N)$$

2. Generate N random points in the p-dimensional Unit Sphere

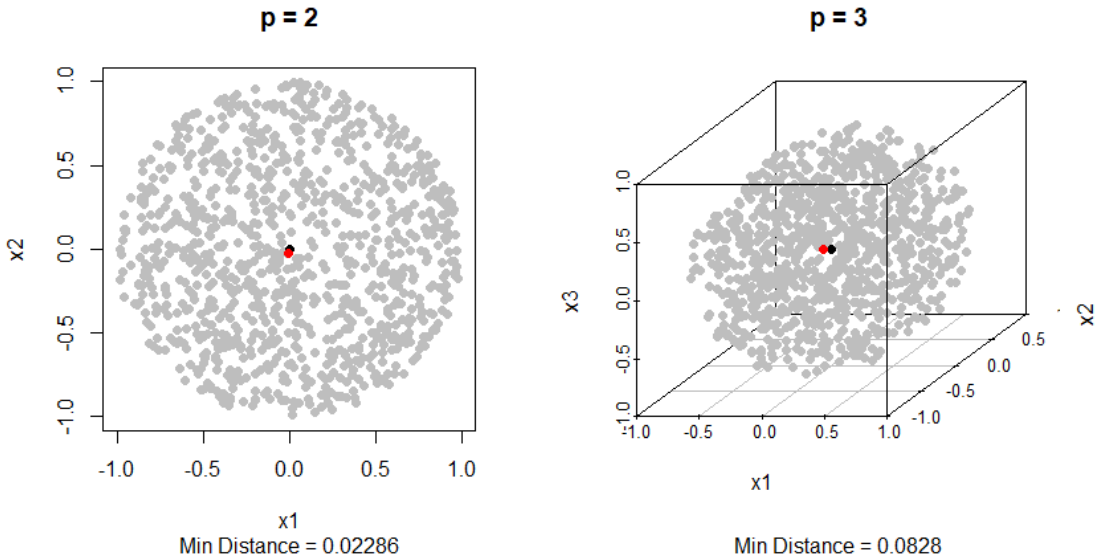
반지름 1 인 p-dimensional Unit Sphere 안에 Random 하게 분포하는 N 개의 point 를 임의로 추출하는 함수를 만들고 이를 이용해 뽑은 Sample 들이 적절하게 분포되어 있는지 산점도 및 히스토그램 등을 그려 확인하였다.

Point Sampling 함수를 위한 알고리즘으로는 다음을 사용하였다.

1) 다변량 표준 정규분포 $N_p(0, I_p)$로부터 $(X_1, \ldots, X_p)$의 sample 을 생성한다.
2) 중심으로부터 생성된 표본으로의 방향 벡터는 크기가 $\sqrt{S} = \sqrt{X_1^2 + \cdots + X_p^2}$이므로 각 원소를 $\sqrt{S}$으로 나누어 크기가 1 이고 방향은 동일한 단위벡터를 만들 수 있다. 이렇게 조정된 표본은 반지름 1 인 Unit Sphere 의 표면(Surface)에 위치한다.
3) $Unif(0, 1)$으로부터 거리 r 의 값을 sampling 하고 $r^{\frac{1}{p}}$의 값을 표본의 각 원소에 곱해줌으로서 Unit Sphere 표면의 점을 내부로 조정할 수 있다.



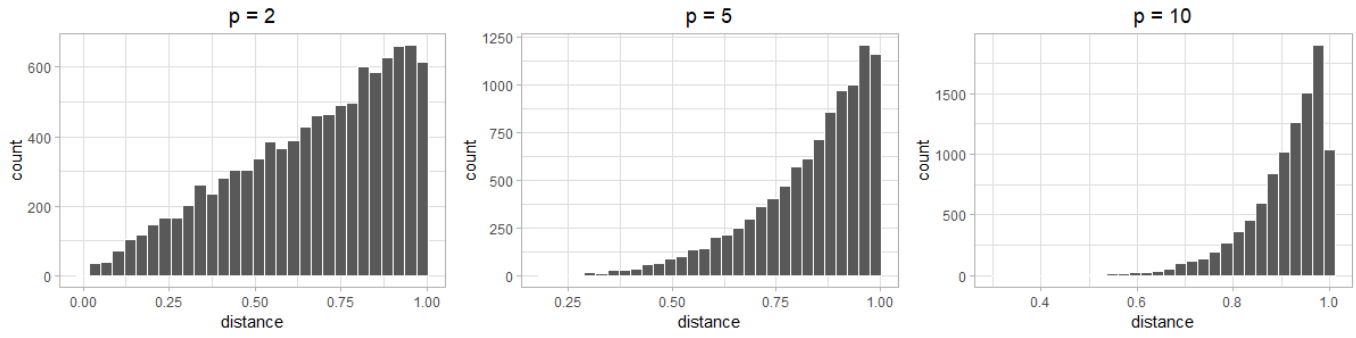
Plot 2.1 ScatterPlot of N Sample Points in a p-dim Unit Sphere



- 2 차원과 3 차원 좌표계에서 N=1000 개의 점들을 생성하여 산점도를 그린 결과 x1, x2 (,x3)의 값들이 각각 [-1, 1] 내에서 random 하게 생성된 것을 확인하였다.

- 중심으로부터 가장 가까운 점까지의 거리를 구해본 결과 p=2 의 경우 0.02286, p=3 의 경우 0.0828 으로 p 의 값이 더 클 때 가장 가까운 점까지의 거리가 증가하였다.

Plot 2.2 Histogram of Sample Distance to the N=10000 points from the origin



- p=2, 5, 10 에 대해 N=10000 개의 sample point 들을 생성하고 중심으로부터의 거리를 구하여 히스토그램을 그린 결과 p 가 커질수록 대부분의 점들이 중심으로부터 먼 곳, 즉 Unit Sphere 의 가장자리에 위치하고 있음을 확인할 수 있다.

3. Compare the results of 1 & 2

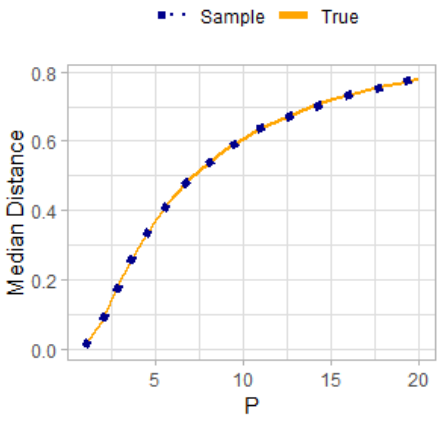
P 와 Sample Size N 의 수준을 각각 변경해가며 가장 가까운 점까지의 거리를 구하는 과정을 총 1000 번, 10000 번 반복하여 총 1000 개, 10000 개의 sample distance 에서 중앙값을 뽑고 이를 실제 **1** 에서 제시한 수식으로 구한 값과 비교하였다.

Table 3.1 Estimated $\mathbf{d(P,N)}$ & True $\mathbf{d(P,N)}$

P	N	1000 simulations	10000 simulations	true median distance
2	**100**	0.08467 (-0.00156)	0.08317 (-0.00006)	0.08311
	1000	0.02762 (-0.00130)	0.02633 (-0.00001)	0.02632
	10000	0.00835 (-0.00002)	0.00826 (0.00006)	0.00833
3	**100**	0.19253 (-0.00208)	0.19029 (0.00015)	0.19045
	1000	0.08746 (0.00103)	0.08852 (-0.00003)	0.08849
	10000	0.04032 (0.00075)	0.04111 (-0.00003)	0.04108
5	**100**	0.36741 (0.00230)	0.36958 (0.00013)	0.36971
	1000	0.23361 (-0.00019)	0.23355 (-0.00013)	0.23342
	10000	0.14752 (-0.00024)	0.14798 (-0.00069)	0.14729
10	**100**	0.61351 (-0.00547)	0.60794 (0.00010)	0.60804
	1000	0.48231 (0.00082)	0.48285 (-0.00028)	0.48313
	10000	0.38736 (-0.00358)	0.38367 (0.00011)	0.38378

- 동일한 P 수준에서는 Sample Size N 이 증가할수록 가장 가까운 점까지의 거리가 작아지며, 시뮬레이션 횟수가 증가할수록 실제 중앙값과의 오차 또한 줄어든다.
- 동일한 Sample Size 수준에서는 P 가 커질수록 가장 가까운 점까지 거리가 증가하며, 시뮬레이션 횟수가 증가할수록 실제 중앙값과의 오차가 줄어들었다.

Discussion



위의 그래프는 1부터 20까지 차원 p에 대하여 N=100개의 점들을 랜덤 생성한 후 중심으로부터의 최소 거리를 계산하는 과정을 1000번 반복하여 추출된 1000개의 최소 거리 중 500번째, 즉 50% quantile으로 median distance을 추정한 값과

실제 median distance의 값을 비교한 결과이다. 두 값이 거의 일치하며 차원 p가 증가할수록 중심으로부터의 가장 가까운 점까지의 거리가 증가하는 것을 확인할 수 있다.

본 과제를 통해 데이터의 차원의 크기가 커질수록 대부분의 점들이 중심에서 멀어져 공간의 가장자리에 분포하는 것을 실제 표본을 생성하여 확인하였다. 또한 중심으로부터 가장 가까운 점의 거리의 중앙값을 표본으로부터 계산하여 실제 값과 비교한 결과가 거의 일치했으며 N과 시뮬레이션 횟수가 증가할수록 오차는 더욱 감소하였다.

따라서 K-NN을 이용한 추정이 고차원에서 매우 불안정한 결과를 보인다는 것을 알게 되었다. 고차원으로 갈수록 전체 공간에서 데이터가 차지하는 공간이 매우 미비해지며, 새로운 sample에 대한 추정을 할 때 해당 sample이 기존의 Train sample들과 멀리 떨어져 있을 가능성이 높아지므로 낮은 차원에 비해 예측이 매우 불안정해진다. 따라서 K-NN은 고차원 다변량의 데이터에 대해서는 사용하기 힘든 방법이다.

[Appendix] R code

Functions

```
# generate p-dim random points
  mysample <- function(n, p) {
    s <- mvrnorm(n, mu = rep(0, p), Sigma = diag(p))
    d <- runif(n, 0, 1)
    S <- apply(s, 1, function(x) sum(x^2))

    s <- d^(1/p) * s / sqrt(S)
    return(s)
  }

  distance <- function(sample) {
    apply(sample, 1, function(x) sqrt(sum(x^2)))
  }

# distance to the closest point in the sample
  closest <- function(sample) {
    d <- distance(sample)
    return(c(m = min(d), sample[which.min(d),]))
  }

# true median distance
  closest_m <- function(n, p) (1-(0.5)^(1/n))^(1/p)
```

Simulation

```
  n <- c(100, 1000, 10000)
  p <- c(2, 3, 5, 10)
  d_true <- matrix(nrow = 3, ncol = 4)
  for (i in 1:3) for (j in 1:4) d_true[i, j] <- closest_m(n[i], p[j])

# 1000 simulations
  Nsim <- 1000

  d_100_1 <- matrix(nrow = Nsim, ncol = 4)
  for (i in 1:Nsim) d_100_1[i, 1] <- closest(mysample(100, 2))[1]
  for (i in 1:Nsim) d_100_1[i, 2] <- closest(mysample(100, 3))[1]
  for (i in 1:Nsim) d_100_1[i, 3] <- closest(mysample(100, 5))[1]
  for (i in 1:Nsim) d_100_1[i, 4] <- closest(mysample(100, 10))[1]

  d_1000_1 <- matrix(nrow = Nsim, ncol = 4)
  for (i in 1:Nsim) d_1000_1[i, 1] <- closest(mysample(1000, 2))[1]
  for (i in 1:Nsim) d_1000_1[i, 2] <- closest(mysample(1000, 3))[1]
  for (i in 1:Nsim) d_1000_1[i, 3] <- closest(mysample(1000, 5))[1]
  for (i in 1:Nsim) d_1000_1[i, 4] <- closest(mysample(1000, 10))[1]

  d_10000_1 <- matrix(nrow = Nsim, ncol = 4)
  for (i in 1:Nsim) d_10000_1[i, 1] <- closest(mysample(10000, 2))[1]
  for (i in 1:Nsim) d_10000_1[i, 2] <- closest(mysample(10000, 3))[1]
  for (i in 1:Nsim) d_10000_1[i, 3] <- closest(mysample(10000, 5))[1]
  for (i in 1:Nsim) d_10000_1[i, 4] <- closest(mysample(10000, 10))[1]

  d_sample_1 <- rbind(apply(d_100_1, 2, median), apply(d_1000_1, 2, median), apply(d_10000_1, 2, median))

# 10000 simulations
  Nsim <- 10000

  d_100_2 <- matrix(nrow = Nsim, ncol = 4)
  for (i in 1:Nsim) d_100_2[i, 1] <- closest(mysample(100, 2))[1]
  for (i in 1:Nsim) d_100_2[i, 2] <- closest(mysample(100, 3))[1]
  for (i in 1:Nsim) d_100_2[i, 3] <- closest(mysample(100, 5))[1]
  for (i in 1:Nsim) d_100_2[i, 4] <- closest(mysample(100, 10))[1]

  d_1000_2 <- matrix(nrow = Nsim, ncol = 4)
  for (i in 1:Nsim) d_1000_2[i, 1] <- closest(mysample(1000, 2))[1]
  for (i in 1:Nsim) d_1000_2[i, 2] <- closest(mysample(1000, 3))[1]
  for (i in 1:Nsim) d_1000_2[i, 3] <- closest(mysample(1000, 5))[1]
  for (i in 1:Nsim) d_1000_2[i, 4] <- closest(mysample(1000, 10))[1]

  d_10000_2 <- matrix(nrow = Nsim, ncol = 4)
  for (i in 1:Nsim) d_10000_2[i, 1] <- closest(mysample(10000, 2))[1]
  for (i in 1:Nsim) d_10000_2[i, 2] <- closest(mysample(10000, 3))[1]
  for (i in 1:Nsim) d_10000_2[i, 3] <- closest(mysample(10000, 5))[1]
  for (i in 1:Nsim) d_10000_2[i, 4] <- closest(mysample(10000, 10))[1]

  d_sample_2 <- rbind( (d_100_2, 2, median), apply(d_1000_2, 2, median), apply(d_10000_2, 2, median))
```

Plots

```
# surface & inside (p = 2)
  surface2d <- mvrnorm(100, mu = rep(0, 2), Sigma = diag(2))
  S <- apply(surface2d, 1, function(x) sum(x^2))
  surface2d <- surface2d / sqrt(S)

  inside2d <- mvrnorm(100, mu = rep(0, 2), Sigma = diag(2))
  S <- apply(inside2d, 1, function(x) sum(x^2))
  inside2d <- inside2d / sqrt(S) *runif(100)

  plot(surface2d[,1], surface2d[,2], pch = 19, col = 'gray',
       xlab = 'x1', ylab = 'x2',   main = 'Sample on(inside) the Surface₩np = 2')
  points(inside2d[,1], inside2d[,2], pch = 19)


# scatterplot (p = 2)
  sample2d <- mysample(1000, 2) ; colnames(sample2d) <- paste0('x', 1:2)
  mininfo <- closest(sample2d)
  mind <- round(mininfo[1], 5)
  minpoint <- mininfo[-1]

  par(mfrow = c(1, 1))
  plot(sample2d[,1], sample2d[,2], pch = 19, col = 'gray',
       xlim = c(-1, 1), ylim = c(-1, 1),
       xlab = 'x1', ylab = 'x2', main = 'p = 2', sub = paste('Min Distance =', mind))
  points(0, 0, pch = 19)
  points(minpoint[1], minpoint[2], col = 'red', pch = 19)

# scatterplot (p = 3)
  sample3d <- mysample(1000, 3) ; colnames(sample3d) <- paste0('x', 1:3)
  mininfo <- closest(sample3d)
  mind <- round(mininfo[1], 5)
  minpoint <- mininfo[-1]

  plt3d <- scatterplot3d(
    sample3d[,1], sample3d[,2], sample3d[,3], color = 'gray', pch = 19,
    xlab = 'x1', ylab = 'x2', zlab = 'x3',
    main = 'p = 3', sub = paste('Min Distance =', mind)
  )
  plt3d$points3d(0, 0, 0, pch = 19)
  plt3d$points3d(minpoint[1], minpoint[2], minpoint[3], pch = 19, col = 'red')


# Histograms
  ggplot(data.frame(d = distance(mysample(10000, 2)))) + theme_light() +
    geom_histogram(aes(d), color = 'white') + labs(x = 'distance', title = 'p = 2') +
    theme(plot.title = element_text(hjust = 0.5))
```

```
  ggplot(data.frame(d = distance(mysample(10000, 3)))) + theme_light() +
    geom_histogram(aes(d), color = 'white', binwith = 0.01) + labs(x = 'distance', title = 'p = 3') +
    theme(plot.title = element_text(hjust = 0.5))

  ggplot(data.frame(d = distance(mysample(10000, 5)))) + theme_light() +
    geom_histogram(aes(d), color = 'white') + labs(x = 'distance', title = 'p = 5') +
    theme(plot.title = element_text(hjust = 0.5))

  ggplot(data.frame(d = distance(mysample(10000, 10)))) + theme_light() +
    geom_histogram(aes(d), color = 'white') + labs(x = 'distance', title = 'p = 10') +
    theme(plot.title = element_text(hjust = 0.5))


# median distance
  P <- 1:20
  N <- 100
  Nsim <- 1000
  Samples <- list()
  MinD <- matrix(nrow = Nsim, ncol = 20)
  for (nn in 1:Nsim) {
    for (pp in P) Samples[[pp]] <- mysample(N, pp)
    MinD[nn,] <- lapply(Samples, function(s) closest(s)[1]) %>% unlist
  }

  data.frame(
    P,
    Sample = apply(MinD, 2, median),
    True = closest_m(N, P)
  ) %>% ggplot() +
    geom_line(aes(P, True,   color = 'True', linetype = 'True'), size = 1) +
    geom_line(aes(P, Sample, color = 'Sample', linetype = 'Sample'), size = 2) +
    theme_light() + theme(legend.position = 'top') +
    scale_color_manual(values = c(Sample = 'darkblue', True = 'orange')) +
    scale_linetype_manual(values = c(Sample = 3, True = 1)) +
    labs(y = 'Median Distance', color = '', linetype = '')
```