­

The Curse of Dimensionality

-Choosing a point from p-dimensional sphere.

­­

**학번 : 182STG12**

**이름 : 오혜윤**

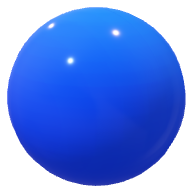
c

Problem Description

Consider N data points uniformly distributed in a p-dimensional unit ball centered at the origin. Then, the median distance from the origin to the closest data point is already given by .

We first prove this equation by using the equation of calculating volume. The volume of a ball of radius r in is , where is a constant depending only on p, the value indicated by shorthand

Considering the image below, we can understand why the probability can be defined to be simply the value of the volume divided by the unit volume. The radius of the red sphere below is within distance d and the blue sphere is a unit ball. To be within distance d, a point should be inside the red sphere. So, we can simply think that the probability that a point is within distance d is the volume of 3-dimensional ball of radius d divided by the volume of 3-dimensional unit ball. Considering this, we can expand this idea and apply it to the p-dimension cases. As a result, the probability is the ratio of the volume of p-dimensional ball of radius d to the volume of p-dimensional unit ball.



R=1

Figure 1 : 3-dimensional sphere

According to the paper called “Choosing a point from the surface of a sphere” by George Marsaglia, there is a method of sampling uniformly from the surface of the unit 3-sphere. From generating independent standard normal variates and putting , we can choose a point () from the surface of a sphere.

I will make the function of generating random points with uniform distribution by referring the method. The nearest distance from a point to point will be calculated for sim.n times. Then by using the function, I will calculate median distances from the sim.n simulations. Each simulation depends on the value of p and N. As sim.n increases, the values of simulated distances will get closer to the value of the true distance by the equation . Also, if p increases or N decreases, simulated distances and the true distances by the equation above will be higher. By comparing the distances simulated with the true distances, I can tell the simulation test is correct or not.

Results

1. **Derive the equation of the median distance .**

The volume of a ball of radius r in is , where is a constant depending only on p, the value indicated by shorthand

Let d be the median distance from the origin to the closest data point. Then, the probability that all N points are further than d from the origin is by the definition of the median.

Since the points are independently distributed, this implies that

and as the points are uniformly distributed in the unit ball, we have that

As a result, the probability that a point, taken uniformly in the unit ball, is within distance d of the origin (is the volume of that ball ( divided by the volume of the unit ball (. Since is cancelled during dividing the volume, we get the probability as and of course, the probability as .

Putting these together, we obtain that

and solving for d, we have

Conclusively, we can get the equation .

1. **Make the function of generating random points on the p-dim’l unit sphere.**

Let’s change the value of N and p. We want to compare the distances from the equation above with the simulated distances. Also, we can plot the way in which the median distance varies as p =c(2,3,5,10,20,30,50,100). We want to verify the equation of median distance above.

The graph below contains a black line of squared bias, a red line of variance and a blue line of MSE.

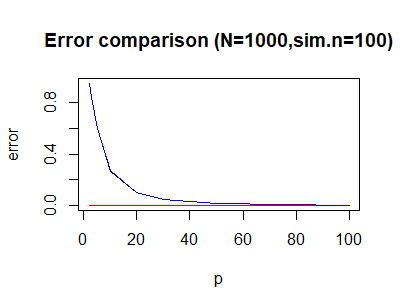
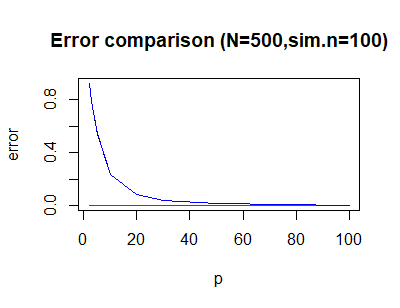
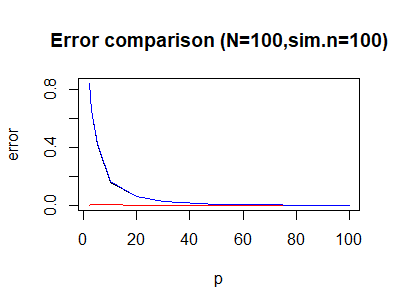
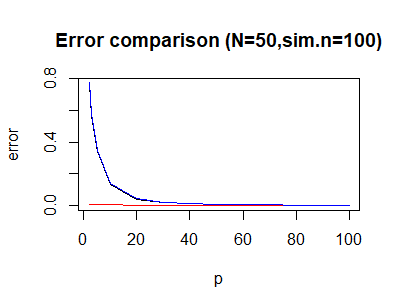
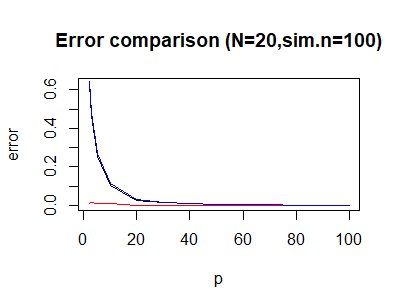


Figure 2 : Squared bias,Variance, MSE of the simulation

According to the figure 2 above, the MSE levels off to 0 and bias squared also levels off to 0. It clearly shows that error is low in high dimension cases.

1. **Compare the results of #1 and #2.**

Let’s change the value of p and N. We want to compare the true distances from the equation above with the simulated distances.

Table 1: The Result of Distances as p and N increase

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The results of #1 and #2 | | | | |
|  | | **#1 (true dist)** | **#2 (100 simulation dist)** | **#2 (1000 simulation dist)** |
| p=2 | **N=20** | 0.1846 | 0.1660 | 0.1848 |
| **N=50** | 0.1173 | 0.1094 | 0.1169 |
| **N=100** | 0.0831 | 0.0822 | 0.0852 |
| **N=500** | 0.0372 | 0.0360 | 0.0368 |
| **N=1000** | 0.0263 | 0.0273 | 0.0260 |
| p=3 | **N=20** | 0.3242 | 0.3197 | 0.3136 |
| **N=50** | 0.2397 | 0.2272 | 0.2365 |
| **N=100** | 0.1904 | 0.2017 | 0.1905 |
| **N=500** | 0.1115 | 0.1126 | 0.1096 |
| **N=1000** | 0.0885 | 0.0886 | 0.0883 |
| p=5 | **N=20** | 0.5087 | 0.5054 | 0.5189 |
| **N=50** | 0.4244 | 0.4097 | 0.4257 |
| **N=100** | 0.3697 | 0.3577 | 0.3713 |
| **N=500** | 0.2681 | 0.2791 | 0.2736 |
| **N=1000** | 0.2334 | 0.2377 | 0.2333 |
| p=10 | **N=20** | 0.7132 | 0.7146 | 0.7131 |
| **N=50** | 0.6515 | 0.6461 | 0.6546 |
| **N=100** | 0.6080 | 0.6050 | 0.6123 |
| **N=500** | 0.5178 | 0.5133 | 0.5169 |
| **N=1000** | 0.4831 | 0.4988 | 0.4859 |
| p=20 | **N=20** | 0.8445 | 0.8398 | 0.8455 |
| **N=50** | 0.8071 | 0.8093 | 0.8067 |
| **N=100** | 0.7798 | 0.7812 | 0.7805 |
| **N=500** | 0.7196 | 0.7210 | 0.7197 |
| **N=1000** | 0.6951 | 0.6931 | 0.6956 |
| p=30 | **N=20** | 0.8935 | 0.9015 | 0.8940 |
| **N=50** | 0.8669 | 0.8711 | 0.8682 |
| **N=100** | 0.8472 | 0.8461 | 0.8449 |
| **N=500** | 0.8030 | 0.8004 | 0.8007 |
| **N=1000** | 0.7847 | 0.7892 | 0.7873 |
| p=50 | **N=20** | 0.9346 | 0.9294 | 0.9343 |
| **N=50** | 0.9179 | 0.9199 | 0.9175 |
| **N=100** | 0.9053 | 0.9107 | 0.9044 |
| **N=500** | 0.8767 | 0.8790 | 0.8775 |
| **N=1000** | 0.8646 | 0.8684 | 0.8646 |
| p=100 | **N=20** | 0.9668 | 0.9658 | 0.96722 |
| **N=50** | 0.9581 | 0.9576 | 0.9582 |
| **N=100** | 0.9515 | 0.9540 | 0.9515 |
| **N=500** | 0.9363 | 0.9380 | 0.9366 |
| **N=1000** | 0.9298 | 0.9302 | 0.9302 |

As a result, we can see that the value of true distance and the value of simulated distances are nearly the same. Therefore, we accomplish the goal of simulating the closest median distances to the distances calculated by the equation.

Discussion

We conclude that the simulation goes well. The values of simulated median distances and true distances are nearly the same. The higher sim.n is, the closer simulated distances are to the true distances. Also, if p increases or N decreases, simulated distances and the true distances by the equation above get higher.

We already expected this result because the higher value of p means that the volume gets bigger. The bigger the volume is, the further the distances between two points are. On the other hand, if N decreases, the distances between two points become smaller. This is because the volume of the ball is limited.

By this result of simulation modeling, we can finally understand that

***“As the number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially.”***

*- Charles Isbell, Professor and Senior Associate Dean, School of Interactive Computing,Georgia Tech .*

As dimensions grow, space increases exponentially. Everytime we increase the number of dimensions more boxes will have to be added to fill the empty spaces. Therefore, the exponential growth in data causes **high sparsity** in the data set and unnecessarily increases storage space. Furthermore, the growth in data increases processing time for the modeling algorithm. For example, supervised machine learning becomes more difficult because predictions for new samples are less likely to be based on learning from similar training features. The number of possible unique rows grows exponentially as the number of features increases, which makes it so much harder to efficiently generalize. The variance increases as they get more opportunity to overfit to noise in more dimensions, resulting in poor generalization performance.

Therefore, we just need to select the best and minimum number of features that help our algorithm learn from it. It is really recommended that we don’t add too many features blindly.

Appendix (R

**#R codes**

library(lattice)

library(mvtnorm)

##function------------------------------------------------------

get.dist<-function(N,p){

x <- rmvnorm(N, mean = rep(0,p), sigma = diag(p))

x <- t(apply(x, 1, function(z){runif(1)^(1/p)\*z/sqrt(sum(z^2))}))

res <- apply(x,1,function(z){sqrt(sum(z^2))})

ifelse(p==1, min(x^2),min(res))

}

#generate p-dim r.v from normal distribution

#generate N nearest distances

#N is sample size and p is dimension size.

##simulation---------------------------------------------------

med\_table<-list() #sim.median list

error\_table<-list() #error table list(squared bias, var, mse)

plot\_list<-list() #error plot list

sim.n=1000

sim.med<-matrix(0,nrow=5,ncol=1);rownames(sim.med) <- c("N=20","N=50", "N=100","N=500","N=1000")

sim.error<-matrix(0,nrow=5,ncol=3);rownames(sim.error) <- c("N=20","N=50", "N=100","N=500","N=1000")

colnames(sim.error) <- c("squared.bias","variance", "mse")

#sim.med : matrix for calculating median distances

#sim.error : matrix for calculating error

pp=1

for (p in c(2,3,5,10,20,30,50,100)){

nn=1

temp<-matrix(0,nrow=1,ncol=sim.n)

for (N in c(20,50,100,500,1000)){

for ( i in 1:sim.n){

dist1<-get.dist(N,p) #generate r.v's

temp[1,i]<-dist1 #generate distances

}

sim.med[nn,1]<-median(temp[1,]) # median distance

sim.error[nn,1]<-mean((temp[1,]-1)^2) # sq.bias

sim.error[nn,2]<-var(temp[1,]) # variance

sim.error[nn,3]<-sim.error[nn,2]+sim.error[nn,3] # variance

med\_table[[pp]]<-sim.med

error\_table[[pp]]<-sim.error

nn<-nn+1 ; #parameter N

}

pp<-pp+1 #parameter p

}

##true distance-----------------------------------------------------------

pp=1

true\_table<-list() #true value list

true<-matrix(0,nrow=5,ncol=1);rownames(true) <- c("N=20","N=50", "N=100","N=500","N=1000")

for (p in c(2,3,5,10,20,30,50,100)){

nn=1

for (N in c(20,50,100,500,1000)){

true[nn,1]<-(1-0.5^(1/N))^(1/p)

true\_table[[pp]] <- true

nn<-nn+1

}

pp<-pp+1

}

##result-------------------------------------------------------------------

View(med\_table)

View(true\_table)

##error graph-------------------------------------------------------

table\_n1<-data.frame(0,nrow=8,ncol=3) ;colnames(table\_n1)<-c("squared bias","variance","mse")

table\_n2<-data.frame(0,nrow=8,ncol=3);colnames(table\_n2)<-c("squared bias","variance","mse")

table\_n3<-data.frame(0,nrow=8,ncol=3);colnames(table\_n3)<-c("squared bias","variance","mse")

table\_n4<-data.frame(0,nrow=8,ncol=3);colnames(table\_n4)<-c("squared bias","variance","mse")

table\_n5<-data.frame(0,nrow=8,ncol=3);colnames(table\_n5)<-c("squared bias","variance","mse")

for (i in 1:8){

table\_n1[i,]<-error\_table[[i]][1,] #N=20

table\_n2[i,]<-error\_table[[i]][2,] #N=50

table\_n3[i,]<-error\_table[[i]][3,] #N=100

table\_n4[i,]<-error\_table[[i]][4,] #N=500

table\_n5[i,]<-error\_table[[i]][5,] #N=1000

}

p=c(2,3,5,10,20,30,50,100)

###N=20 graph

plot(p,table\_n1$`squared bias`,type="l",ylab = "error",main="Error comparison (N=20,sim.n=100)")

lines(p,table\_n1$variance,col="red")

lines(p,table\_n1$mse,col="blue")

###N=50 graph

plot(p,table\_n2$`squared bias`,type="l",ylab = "error",main="Error comparison (N=50,sim.n=100)")

lines(p,table\_n2$variance,col="red")

lines(p,table\_n2$mse,col="blue")

###N=100 graph

plot(p,table\_n3$`squared bias`,type="l",ylab = "error",main="Error comparison (N=100,sim.n=100)")

lines(p,table\_n3$variance,col="red")

lines(p,table\_n3$mse,col="blue")

###N=500 graph

plot(p,table\_n4$`squared bias`,type="l",ylab = "error",main="Error comparison (N=500,sim.n=100)")

lines(p,table\_n4$variance,col="red")

lines(p,table\_n4$mse,col="blue")

###N=1000 graph

plot(p,table\_n5$`squared bias`,type="l",ylab = "error",main="Error comparison (N=1000,sim.n=100)")

lines(p,table\_n5$variance,col="red")

lines(p,table\_n5$mse,col="blue")