

자료와 정보 과제 2: 확률 추정

20232907 정현승

요약

Gaussian 분포를 이루는 데이터를 최대 우도 추정, 파젠 창, k 이웃 찾기 방법을 이용해 Gaussian 분포의 파라미터나 확률 분포의 오차를 구해 어떤 방법이 효과적인지 알아보았다. 그 결과 Gaussian의 최대 우도 추정 방법이 가장 효과가 좋았고, 파젠 창은 창의 크기가 작을수록, k 이웃 찾기는 k 값이 클수록 확률 분포를 잘 추정할 수 있었으며 k 이웃 찾기에서 사각형 창을 사용하면 오차가 매우 커진다는 점도 알 수 있다.

1. 서론

Gaussian 분포를 이루는 데이터를 최대 우도 추정, 파젠 창, k 이웃 찾기 방법을 이용해 Gaussian 분포의 파라미터 또는 확률 분포를 구해보았다. Gaussian 분포 생성기를 이용해 2차원 Gaussian 분포 데이터를 만든 후 각 확률 분포 추정 방법의 오차를 구해 어떤 추정 방법이 효과적인지, 각 방법에서 창의 모양이나 변수는 어떻게 설정해야 하는지를 알아보았다.

2. 분석하는 데이터 생성

이번에 분석하는 데이터는 사람의 키와 몸무게 데이터이고, 키의 평균은 182.9cm, 몸무게의 평균은 67.7kg으로, 키의 분산은 $92cm^2$, 몸무게의 분산은 $72kg^2$ 로 설정했다. 두 데이터 간 Correlation은 없는 것으로 가정했다. 데이터 생성용 난수 발생기는 numpy 라이브러리를 사용했고, 라이브러리 사용을 위해 Python으로 코드를 작성했다. 데이터를 생성해 데이터의 크기와 함께 csv 파일 형태로 저장했고, 이를 반복해 샘플의 수가 1000, 5000, 10000개인 데이터 세트를 각 10개씩 총 30개의 데이터 세트를 완성하였다. 이 데이터 세트를 이용해 최대 우도 추정, 파젠 창, k-NN 방법으로 각각 확률 분포를 구하고 각 분포 사이의 오차를 분석했다.

이후 설명에서는 단위를 생략한다.

2. 결과 도출 방법

앞서 언급한 것처럼 각 데이터 세트를 3가지 방법으로 확률 분포를 구하는 방법으로 진행하였다. 최대 우도 추정은 수식 계산이 필요하나 코드에는 수식 계산 부분은 넣지 않고 수식을 계산함으로써 얻을 수 있는 최종적인 수식만을 작성해 값을 구했다. 애초에 수식에 데이터는 물론 평균 벡터와 공분산 행렬도 포함되기에 데이터만으로는 최대 우도 추정값을 구할 수 없고 이 데이터에 평균과 분산이 주어졌을 때의 확률 수식을 알아야 최대 우도 추정을 할 수 있으니, Gaussian일 때의 확률 수식으로 설정했을 때의 최대 우도 추정 결과(평균 벡터와 공분산 행렬을 구하는 수식)를 사용했다.

파젠 창과 k-NN 방법은 특정한 데이터를 줬을 때 이 데이터의 확률만을 구할 수 있다. 최대 우도 추정과 달리 이건 수식이나 벡터의 형태로 나오는 게 아니라 확률, 즉 하나의 스칼라 데이터만으로 결과가 나

온다. 따라서 특정한 데이터의 변화에 따른 확률 그래프를 그리는 방법으로 접근해야 한다. ‘데이터의 값이 a부터 b까지는 확률 수식이 어떤 것이고 b부터 c까지는 확률 수식은 어떤 것이다.’ 하는 식으로 데이터의 범위에 따라 확률 수식을 구해 확률 분포를 구할 수도 있지만 할 자신이 없고 특히 k-NN에서 이런 방식으로 확률 수식을 구하는 방법을 몰라 (코드 작성이 어렵다는 의미가 아니라 데이터 5개 주고 직접 해보라고 해도 방법을 몰라 못 하겠다. 별도의 그래프가 있어야 한데 이 그래프를 그리는 방법도 별로 나와 있지 않다.) 특정 지점에서의 확률 분포의 값을 구해 이를 비교하였고, 대신 이 확률 분포를, 데이터를 바꾸어가면서 여러 번 구했다. 최대 우도 추정으로 구한 평균 벡터와 공분산 행렬을 이용해 특정한 데이터에서의 Gaussian 함수의 값을 구할 수 있으니 이를 활용했다. 데이터를 바꾸어가면서 진행할 때는 데이터 생성 시 설정했던 평균과 분산을 기준으로, 평균에서 멀어질수록 신뢰구간이 떨어지니 평균에서 가까운 데는 데이터 간 간격을 좁게 해서 확률 분포의 값을 많이 구했고, 평균에서 먼 데도 각 분포 사이의 오차를 보기는 해야 하니 데이터 간 간격을 넓게 해서 확률 분포의 값을 비교적 적게 구했다. 또 이렇게 구한 데이터로 그래프를 그려야 하니 데이터의 수가 많아야 하므로 평균에서 데이터의 차가 표준편차의 2배 이내라면 표준편차의 0.1 배씩 차이를 두어서, 2배에서 5배 이내라면 1배씩 차이를 두어서 확률 분포의 값을 구했다. 처음에는 0.01배씩 차이를 두어 진행하려다 시간이 너무 오래 걸려 포기하였으며, 데이터의 차가 표준편차의 5배를 넘어가면 확률 분포 값이 double 자료형의 precision 안에 들어가지 못해 연산 결과가 0이 나오는 상황이 발생해, 결과의 정확도를 믿을 수 없어 제외하였다(뒤에 나오는 결과를 보면 5배 차이났을 때도 값이 0이 나오는 상황을 볼 수 있다). 데이터의 두 필드 중 어떤 필드의 평균과의 차는 표준편차와 큰 차이가 나지 않는데 다른 필드는 표준편차의 몇 배 정도로 차이가 나는 상황에서는 표준편차와 차이가 큰 것을 기준으로 차이를 두었다. 이렇게 여러 번 확률 분포의 값을 구하고 그 값과 기준이 되는 데이터의 값, 오차를 csv 파일 형태로 저장하였다. 다만 이렇게 데이터의 값을 많이 구하면, 데이터의 차가 표준편차의 2배 이내에서 0.1 배씩 차이를 둔 것에서부터 하나의 차원당 41번의 확률 분포 값을 구하게 되며, 차원이 2이므로 여기에서만 421번 값을 구하게 된다. 이렇게 많은 데이터는 그래프를 그리기에는 좋지만, 이 데이터를 보고서에 전부 작성하기에는 한계가 있다. 따라서 보고서에 넣을 목적으로 평균에서 데이터의 차가 표준편차의 0, 0.1, 0.5, 1, 2, 5배일 때의 확률 분포 값을 따로 구해 그 수치를 작성하였다. 이때 두 필드 중 다른 하나의 필드 값과 평균의 차가 0일(즉 표준편차의 0배일) 때도 포함하여 총 41개의 값을 작성했다. 1777개의 데이터에서 필요한 41개의 데이터만 찾는 게 오래 걸릴 것 같아 코드를 바꿔가며 실행을 2번 했으며, 데이터는 같기 때문에 두 실행에서 데이터값이 같을 때 확률 분포 값도 같아 큰 문제가 없다. 또 샘플의 수가 같은 10개의 세트는 각각을 작성하지 않고 평균값을 작성하였다. 이는 그래프도 마찬가지이다. 41개의 데이터만 확률 분포를 찾은 결과는 “확률 분포 결과 - 요약”에 샘플의 수가 1000, 5000, 10000개인 것끼리 평균을 내, 이 순서로 각각 1, 2, 3 순번으로 저장하였으며, 1777개의 데이터 전체를 찾은 결과는 “확률 분포 결과 - 전체”에 데이터 세트 순서대로 저장되어 있다.

이외에도 파젠 창은 창크기의 크기를, k-NN 방법은 k의 크기를 바꾸어가며 값을 구할 수 있는데, 파젠 창은 창크기를 데이터 생성 시 설정했던 표준편차의 1, 2, 3배로 바꾸어가며 값을 구했고, k-NN 방법은 k의 크기를 3, 5, 7로 바꾸어가며 값을 구했다. 또 창을 사각형 창을 사용할 수도 있고 원형 창을 사용할 수도 있는데, 파젠 창과 k-NN 둘 다 두 방법을 모두 활용하고 그 값을 비교해 보았다. k-NN에서 사각형 창을 사용한다는 건 거리 계산 시 최댓값 연산(Chebyshev distance)을 활용한다는 의미이고 원형 창을 사용한다는 건 Euclidean distance를 활용한다는 의미이다. 이외에도 키와 몸무게의 가중치를 두어 키가 1 차이 나는 것과 몸무게가 2 차이 나는 것을 같은 것으로 취급하고 연산을 진행하는 것도 가능하나, 이미 구하는 값이 많기도 하고 Correlation을 0으로 설정해 둘 사이의 연관성도 없고, 그리고 저번 과제 1에서 가중치를 아무렇게나 설정하다가는 오히려 결과가 안 좋아질 수 있다는 것을 알았기 때문에, 코드는 가중치 설정이 가능하도록 구성되어 있지만 이를 테스트하지도 않고 활용하지도 않았다.

3. 소프트웨어 구조

이 과제에는 총 3개의 소프트웨어를 제작해 사용하였다. 주요 기능인 확률 분포를 구하는 소프트웨어와 데이터를 정규 분포의 형태로 생성하는 소프트웨어, 계산된 확률 분포 값을 이용해 그래프를 그리는 소프트웨어가 각각 존재하며, 이 중 주요 기능인 확률 분포를 구하는 소프트웨어는 C 언어로, 제한적으로 부가적인 기능을 담당하는 나머지 2개 소프트웨어는 Python으로 코딩하였으며 둘 다 외부 라이브러리를 활용했다. 주요 소프트웨어에서는 외부 라이브러리는 사용하지 않았고 다만 부동소수점 덧셈 연산을 진행할 때 저번처럼 부동소수점 변수의 precision을 고려해 연산하기 위해 저번 과제 1에서 사용했던 함수 하나를 가져왔다. 또한 저번보다 주석을 철저히 달았고 코드가 복잡하지 않아 세부적인 설명은 필요 없어 보인다. 자잘한 설명을 하자면 이번에는 저번과 달리 파일 입출력에 이상이 생기지 않는 한 화면에 출력하는 동작이 없다(디버그 시에는 포함했으나 최종본에서는 제외했다). 결과 출력은 오직 파일 출력으로만 진행되며 해당하는 경로가 없거나 같은 이름의 파일이 있으면 파일 출력을 하지 않고 프로그램을 종료한다. 따라서 소프트웨어를 실행하기 전 결과 파일이 담길 폴더를 만들어 주어야 한다. 또 본격적인 연산 부분을 별도의

함수로 분리했으며, 특정한 지점을 잡고 이에 대한 연산이 반복되어 한 지점에 대해 여러 방법으로 구한 확률 변수 값을 하나의 구조체(PersonChance)로 몰아주었다. 여기에 원래 Gaussian 분포에 따른 결괏값은 물론 여러 방법으로 구한 값이 모두 들어가 있고, 여기에 더해 Gaussian 분포에 대한 각각의 오차까지 포함하고 있어 구조체가 매우 복잡해졌다. 또 k-NN 방법은 같은 점에 대해 k 값만 바꾸어 연산하는 과정이 있으나 점이 같으면 k-NN 방법에서 필요한 거릿값도 전부 같다. 또한 거릿값의 집합만 있다면 전체 데이터를 몰라도 k-NN 방법으로 연산할 수 있다 보니, 점에서 전체 데이터의 거릿값 집합을 따로 구했고 이렇게 한 번 구한 데이터를 k 값을 바꾸어 가며 여러 번 이용하였다. 또한 k-NN 방법에서 필요한 건 k 번째로 작은 거릿값이나(이를 이용해 창 크기 구하기 때문이다) k를 3, 5, 7로만 두고 연산하기 때문에 전체 데이터를 정렬할 필요가 없다. 따라서 선택 정렬을 이용해 앞 7개 데이터만 작은 것 순서대로 앞에 오도록 정렬을 실시했다. 나머지 데이터는 정렬을 실시하지 않았다. 이 거리를 구하는 과정에서 Euclidean Distance를 사용할 때는, 거리의 합을 계산하는 게 목적이 아니고 비교가 목적이라 계산 후 제곱근을 취하지 않고 제곱된 상태로 두었다. 창 크기 구할 때도 거리의 제곱이 필요하므로 제곱근을 취하지 않았다.

이 소프트웨어가 사용하는 파일 형태는, 데이터 세트는 첫 줄에 z,(데이터 샘플 수),(데이터 차원), 둘째 줄에 각 필드에 대한 설명과 3번째 줄부터 끝까지 index,height,weight 형식으로 데이터가 나열된 형태이다. 출력 파일인 확률 분포 결과는 첫 줄에 필드 설명, 두 번째 줄부터 데이터 나열이 이어지며, 데이터 나열 후 마지막 4줄 정도 MLE의 파라미터를 나열한다. 파일명은 "20232907-<순번>.csv"로 통일해 다른 이름의 파일은 읽을 수 없다.

난수 발생기 코드는 20줄 정도로 간단하게 작성하였으며 Python의 numpy 라이브러리를 이용했다. 그 래프 생성기 코드는 라이브러리의 도움 없이 작성하려면 별을 찍어 작성해야 하는데 그럴 자신은 없어 Python의 matplotlib 라이브러리를 이용했다. 이 그래프 생성기도 코드가 약간 복잡한데 이는 주 코드의 PersonChance 구조체와 파일 출력 결과가 복잡하기 때문으로, 10개 데이터를 평균 내어 각 방법에 대한 그래프를 그리는 것 외에는 특별한 점은 없다.

코드 작성 시 일부 모르는 부분은 인터넷에서 정보를 찾아보았으며, 이러한 부분은 참고문헌에 명시했다. 논문과는 달리 코드가므로 출처가 불분명해도 작동에 문제없으면 참고하였다.

4. 결과

결과를 잘 보이게 하기 위해 49개의 데이터를 전부 가져오지 않고 일부만 가져왔다. 단 그래프는 모든 데이터를 담았다. 이렇게 표기하지 않으면 표만 너무 많이 들어가게 되기 때문이다. 이 데이터는 같은 조건으로 만들어진 샘플 집합 결과의 평균으로, 전체 데이터는 별도로 첨부했다. 오차는 정규분포 값을 참값으로 보고 각 데이터에서 정규분포 값을 뺀 후 이를 정규분포 값으로 나눈 값으로, 실제 값과 정규분포 값의 대소 비교를 위해 절댓값을 취하지 않았다.

데 이 터 샘플 수	키	134.9417	182.9	163.7167	202.0833
	몸무게	25.27359	25.27359	84.67056	67.7
	정규분포 값	3.772E-25	2.716E-14	6.56E-07	3.582E-05
1000	MLE 값	4.647E-25	2.83E-14	6.515E-07	3.242E-05
	사각 파젠 창 값 (σ)	0	0	3.809E-05	0.000274
	사각 파젠 창 값 (2σ)	0	3.072E-07	7.771E-05	0.0003099
	사각 파젠 창 값 (3σ)	0	1.365E-07	0.0001319	0.0003565
	원형 파젠 창 값 (σ)	0	0	4.567E-05	0.0002602
	원형 파젠 창 값 (2σ)	0	0	6.989E-05	0.0002931
	원형 파젠 창 값 (3σ)	0	1.538E-07	0.0001081	0.0003343
	사각 kNN 값 (k=3)	9.363E-10	6.119E-09	8.814E-06	0.0003403
	사각 kNN 값 (k=5)	1.242E-09	7.529E-09	3.948E-06	6.267E-05
	사각 kNN 값 (k=7)	1.585E-09	8.764E-09	1.791E-06	4.629E-05
	원형 kNN 값 (k=3)	6.024E-07	2.3E-06	7.492E-05	0.0003838
	원형 kNN 값 (k=5)	9.302E-07	3.477E-06	6.498E-05	0.0002787
	원형 kNN 값 (k=7)	1.235E-06	4.451E-06	5.338E-05	0.0002849
	MLE 오차	0.2321971	0.0421064	-0.006851	-0.094757
	사각 파젠 창 오차 (σ)	-1	-1	57.062725	6.6500216
	사각 파젠 창 오차 (2σ)	-1	11310502	117.46669	7.6534437

	사각 파젠 창 오차 (3σ)	-1	5026889.3	200.03438	8.9522702
	원형 파젠 창 오차 (σ)	-1	-1	68.619695	6.2643724
	원형 파젠 창 오차 (2σ)	-1	-1	105.53923	7.182079
	원형 파젠 창 오차 (3σ)	-1	5662155.4	163.79005	8.333774
	사각 kNN 오차 ($k=3$)	2.482E+15	225292.37	12.435694	8.5004248
	사각 kNN 오차 ($k=5$)	3.292E+15	277233.7	5.0180266	0.749806
	사각 kNN 오차 ($k=7$)	4.203E+15	322705.59	1.7305821	0.2924718
	원형 kNN 오차 ($k=3$)	1.597E+18	84701842	113.21057	9.7171248
	원형 kNN 오차 ($k=5$)	2.466E+18	128010309	98.055876	6.7800944
	원형 kNN 오차 ($k=7$)	3.274E+18	163889131	80.372421	6.9542796
5000	MLE 값	4.284E-25	3.697E-14	6.576E-07	3.61E-05
	사각 파젠 창 값 (σ)	0	0	4.472E-05	0.0002914
	사각 파젠 창 값 (2σ)	0	6.143E-08	7.587E-05	0.0003284
	사각 파젠 창 값 (3σ)	0	2.184E-07	0.0001293	0.0003636
	원형 파젠 창 값 (σ)	0	0	4.346E-05	0.0002868
	원형 파젠 창 값 (2σ)	0	2.076E-07	6.726E-05	0.0003135
	원형 파젠 창 값 (3σ)	0	3.691E-07	0.0001077	0.0003427
	사각 kNN 값 ($k=3$)	3.078E-10	2.722E-09	4.918E-05	0.0008736
	사각 kNN 값 ($k=5$)	4.126E-10	3.283E-09	1.572E-05	0.0004074
	사각 kNN 값 ($k=7$)	5.216E-10	3.929E-09	9.434E-06	0.000221
	원형 kNN 값 ($k=3$)	1.631E-07	7.254E-07	5.896E-05	0.0003397
	원형 kNN 값 ($k=5$)	2.46E-07	9.956E-07	5.053E-05	0.000299
	원형 kNN 값 ($k=7$)	3.238E-07	1.304E-06	4.832E-05	0.0002609
	MLE 오차	0.1358106	0.3612844	0.0023673	0.0078529
	사각 파젠 창 오차 (σ)	-1	-1	67.176878	7.137153
	사각 파젠 창 오차 (2σ)	-1	2262099.6	114.6572	8.1680191
	사각 파젠 창 오차 (3σ)	-1	8043023.5	196.03867	9.150477
	원형 파젠 창 오차 (σ)	-1	-1	65.244195	7.0062657
	원형 파젠 창 오차 (2σ)	-1	7643910.1	101.53082	7.7539551
	원형 파젠 창 오차 (3σ)	-1	13589174	163.18059	8.5690503
	사각 kNN 오차 ($k=3$)	8.16E+14	100229.24	73.970177	23.391291
	사각 kNN 오차 ($k=5$)	1.094E+15	120870.7	22.964579	10.375377
	사각 kNN 오차 ($k=7$)	1.383E+15	144671.36	13.381598	5.1707828
	원형 kNN 오차 ($k=3$)	4.325E+17	26711214	88.874641	8.4839816
	원형 kNN 오차 ($k=5$)	6.523E+17	36659215	76.030909	7.3474272
	원형 kNN 오차 ($k=7$)	8.585E+17	48004279	72.653097	6.2850287
10000	MLE 값	4.013E-25	2.596E-14	6.794E-07	3.601E-05
	사각 파젠 창 값 (σ)	0	0	4.423E-05	0.0002916
	사각 파젠 창 값 (2σ)	0	6.143E-08	7.771E-05	0.00033
	사각 파젠 창 값 (3σ)	0	2.867E-07	0.000129	0.0003657
	원형 파젠 창 값 (σ)	0	0	4.401E-05	0.0002854
	원형 파젠 창 값 (2σ)	0	3.46E-08	6.934E-05	0.0003118
	원형 파젠 창 값 (3σ)	0	4.152E-07	0.0001079	0.0003472
	사각 kNN 값 ($k=3$)	1.637E-10	2.629E-09	2.755E-05	0.0022207
	사각 kNN 값 ($k=5$)	2.324E-10	3.041E-09	2.828E-05	0.0015464
	사각 kNN 값 ($k=7$)	2.906E-10	3.528E-09	1.389E-05	0.0010611
	원형 kNN 값 ($k=3$)	8.616E-08	4.684E-07	4.055E-05	0.0003569
	원형 kNN 값 ($k=5$)	1.337E-07	6.577E-07	4.388E-05	0.0003883
	원형 kNN 값 ($k=7$)	1.76E-07	8.527E-07	4.652E-05	0.0003686

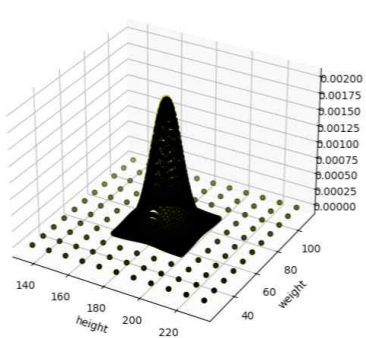
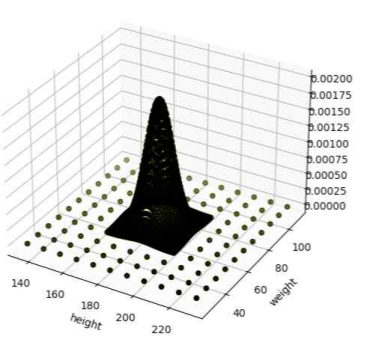
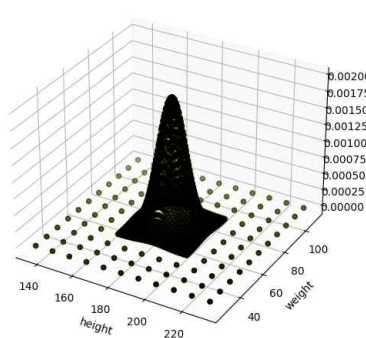
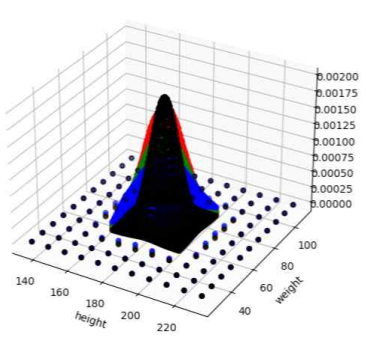
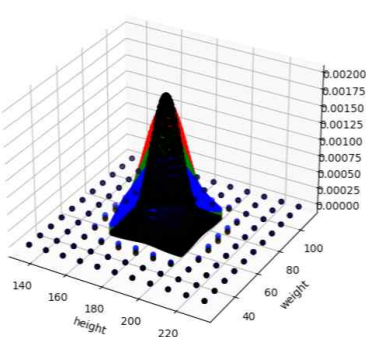
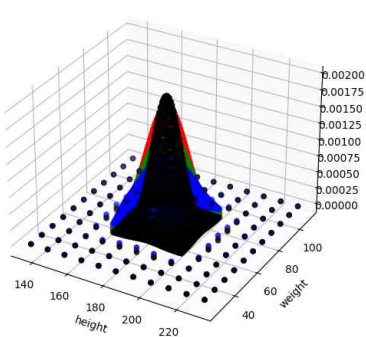
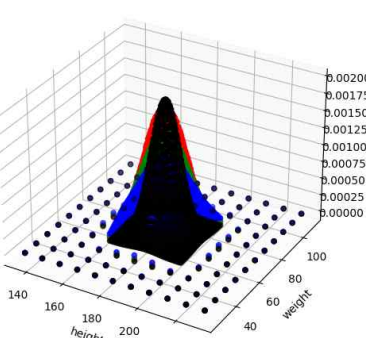
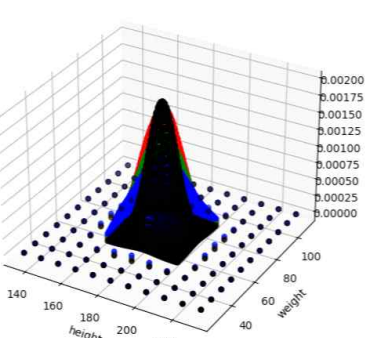
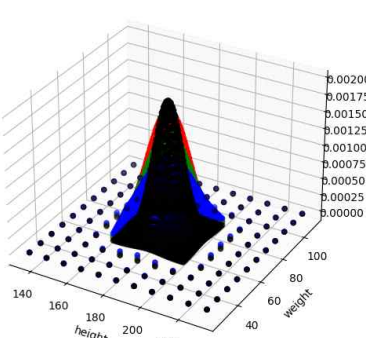
	MLE 오차	0.0639415	-0.043992	0.035711	0.0054595
	사각 파젠 창 오차 (σ)	-1	-1	66.427681	7.1405835
	사각 파젠 창 오차 (2σ)	-1	2262099.6	117.46669	8.2143309
	사각 파젠 창 오차 (3σ)	-1	10556469	195.64326	9.2099391
	원형 파젠 창 오차 (σ)	-1	-1	66.08807	6.9676254
	원형 파젠 창 오차 (2σ)	-1	1273984.2	104.69536	7.7056548
	원형 파젠 창 오차 (3σ)	-1	15287821	163.43844	8.6952752
	사각 kNN 오차 (k=3)	4.341E+14	96785.927	40.997813	61.001407
	사각 kNN 오차 (k=5)	6.16E+14	111961.88	42.105098	42.175725
	사각 kNN 오차 (k=7)	7.705E+14	129903.92	20.174753	28.627276
	원형 kNN 오차 (k=3)	2.284E+17	17248417	60.815648	8.9644328
	원형 kNN 오차 (k=5)	3.544E+17	24216824	65.884124	9.8406241
	원형 kNN 오차 (k=7)	4.667E+17	31397088	69.920335	9.2915711

표 2

데 이 터 샘플 수	키	192.49166	178.10417	183.85917	182.9
	몸무게	59.214719	71.942641	68.548528	67.7
	정규분포 값	0.0002646	0.0011861	0.0019168	0.0019555
1000	MLE 값	0.0002532	0.0012109	0.0019323	0.0019785
	사각 파젠 창 값 (σ)	0.0007028	0.0014535	0.0018049	0.0018123
	사각 파젠 창 값 (2σ)	0.0006917	0.0012133	0.0014336	0.0014483
	사각 파젠 창 값 (3σ)	0.0006358	0.0009279	0.0010225	0.0010273
	원형 파젠 창 값 (σ)	0.0007266	0.0014642	0.001813	0.0018324
	원형 파젠 창 값 (2σ)	0.000684	0.0012459	0.0014884	0.0015012
	원형 파젠 창 값 (3σ)	0.0006485	0.0009852	0.0010987	0.001105
	사각 kNN 값 (k=3)	0.0008836	0.0230967	0.009286	0.0118805
	사각 kNN 값 (k=5)	0.0004758	0.0039459	0.0032923	0.0052328
	사각 kNN 값 (k=7)	0.0003594	0.0036665	0.0025669	0.0029826
	원형 kNN 값 (k=3)	0.0007511	0.00322	0.0025117	0.0026592
	원형 kNN 값 (k=5)	0.0006676	0.0020569	0.0019588	0.0024703
	원형 kNN 값 (k=7)	0.0007419	0.0019868	0.0020818	0.0020604
	MLE 오차	-0.043184	0.0208937	0.0081064	0.0117768
	사각 파젠 창 오차 (σ)	1.6556135	0.2254959	-0.058354	-0.07323
	사각 파젠 창 오차 (2σ)	1.6138293	0.0229731	-0.2521	-0.25937
	사각 파젠 창 오차 (3σ)	1.4023294	-0.217649	-0.466536	-0.474656
	원형 파젠 창 오차 (σ)	1.7454314	0.2345118	-0.054156	-0.062977
	원형 파젠 창 오차 (2σ)	1.5846275	0.0504435	-0.223469	-0.232299
	원형 파젠 창 오차 (3σ)	1.4502612	-0.169343	-0.426798	-0.434924
	사각 kNN 오차 (k=3)	2.3388802	18.473226	3.8445795	5.0753752
	사각 kNN 오차 (k=5)	0.7978234	2.3268868	0.7176263	1.6759319
	사각 kNN 오차 (k=7)	0.3578431	2.0912574	0.3391646	0.5252455
	원형 kNN 오차 (k=3)	1.8380518	1.714863	0.31038	0.359849
	원형 kNN 오차 (k=5)	1.5224277	0.7342229	0.021908	0.263232
	원형 kNN 오차 (k=7)	1.8031827	0.6750982	0.0860798	0.0536514
5000	MLE 값	0.0002663	0.0011823	0.001914	0.0019517
	사각 파젠 창 값 (σ)	0.0007257	0.0014294	0.001801	0.0018153
	사각 파젠 창 값 (2σ)	0.0007012	0.0012015	0.0014205	0.0014317
	사각 파젠 창 값 (3σ)	0.0006406	0.0009106	0.0010206	0.0010261
	원형 파젠 창 값 (σ)	0.0007205	0.0014299	0.0018241	0.0018329

	원형 파젠 창 값 (2σ)	0.0007047	0.0012375	0.0014833	0.0014984
	원형 파젠 창 값 (3σ)	0.0006519	0.0009692	0.001096	0.0011041
	사각 kNN 값 ($k=3$)	0.0100214	0.023585	0.1229671	0.0602495
	사각 kNN 값 ($k=5$)	0.003641	0.0150809	0.0273251	0.0255465
	사각 kNN 값 ($k=7$)	0.0025604	0.0083713	0.0221003	0.0133525
	원형 kNN 값 ($k=3$)	0.0010532	0.001785	0.0032368	0.0025612
	원형 kNN 값 ($k=5$)	0.0008629	0.0016578	0.0024948	0.0024238
	원형 kNN 값 ($k=7$)	0.0008061	0.0015789	0.0025232	0.0021659
	MLE 오차	0.0060738	-0.003217	-0.00143	-0.001926
	사각 파젠 창 오차 (σ)	1.7419673	0.2051918	-0.060405	-0.071722
	사각 파젠 창 오차 (2σ)	1.649578	0.0129765	-0.258895	-0.267883
	사각 파젠 창 오차 (3σ)	1.4203843	-0.232221	-0.467562	-0.475256
	원형 파젠 창 오차 (σ)	1.722422	0.2055742	-0.04838	-0.062694
	원형 파젠 창 오차 (2σ)	1.6628069	0.0433841	-0.226177	-0.23375
	원형 파젠 창 오차 (3σ)	1.4632765	-0.182826	-0.42821	-0.43538
	사각 kNN 오차 ($k=3$)	36.866777	18.884907	63.152699	29.810101
	사각 kNN 오차 ($k=5$)	12.757917	11.714946	13.255681	12.063871
	사각 kNN 오차 ($k=7$)	8.6748297	6.0579683	10.529887	5.8281361
	원형 kNN 오차 ($k=3$)	2.9795181	0.5049249	0.6886628	0.3097536
	원형 kNN 오차 ($k=5$)	2.2605954	0.3977331	0.3015475	0.2394969
	원형 kNN 오차 ($k=7$)	2.0458426	0.3311818	0.3163881	0.1075985
10000	MLE 값	0.0002607	0.001195	0.0019177	0.0019548
	사각 파젠 창 값 (σ)	0.0007226	0.0014318	0.001782	0.0018021
	사각 파젠 창 값 (2σ)	0.0006974	0.0011962	0.0014231	0.0014325
	사각 파젠 창 값 (3σ)	0.0006395	0.0009165	0.0010211	0.001024
	원형 파젠 창 값 (σ)	0.0007231	0.00144	0.0018037	0.0018221
	원형 파젠 창 값 (2σ)	0.0007017	0.0012313	0.0014792	0.0014906
	원형 파젠 창 값 (3σ)	0.0006521	0.0009741	0.0010982	0.0011034
	사각 kNN 값 ($k=3$)	0.0042205	0.0850343	0.1186144	0.4887907
	사각 kNN 값 ($k=5$)	0.0037244	0.0336085	0.04139	0.0765615
	사각 kNN 값 ($k=7$)	0.0029324	0.0211499	0.0351779	0.047774
	원형 kNN 값 ($k=3$)	0.0005438	0.0023059	0.0024799	0.0041469
	원형 kNN 값 ($k=5$)	0.0006785	0.0017989	0.0021092	0.0025583
	원형 kNN 값 ($k=7$)	0.0006835	0.0018513	0.0023353	0.0021634
	MLE 오차	-0.014787	0.0074926	0.00045	-0.000375
	사각 파젠 창 오차 (σ)	1.7303606	0.2071601	-0.070341	-0.078445
	사각 파젠 창 오차 (2σ)	1.6353017	0.008522	-0.257549	-0.267459
	사각 파젠 창 오차 (3σ)	1.4164638	-0.227294	-0.46727	-0.476345
	원형 파젠 창 오차 (σ)	1.7323579	0.2140921	-0.058994	-0.068214
	원형 파젠 창 오차 (2σ)	1.6513023	0.0381042	-0.228307	-0.237766
	원형 파젠 창 오차 (3σ)	1.4640319	-0.178716	-0.427046	-0.435749
	사각 kNN 오차 ($k=3$)	14.94767	70.693807	60.881874	248.95566
	사각 kNN 오차 ($k=5$)	13.072827	27.335846	20.593422	38.151663
	사각 kNN 오차 ($k=7$)	10.080195	16.831808	17.352515	23.430443
	원형 kNN 오차 ($k=3$)	1.0546617	0.94411	0.2938049	1.1206186
	원형 kNN 오차 ($k=5$)	1.5637658	0.5166479	0.1004033	0.3082271
	원형 kNN 오차 ($k=7$)	1.5825487	0.5608476	0.2183229	0.1063146

표 3

	샘플 수 1000개인 데이터	샘플 수 5000개인 데이터	샘플 수 10000개인 데이터
M L E	<p>graph of 1000 samples: MLE</p> 	<p>graph of 5000 samples: MLE</p> 	<p>graph of 10000 samples: MLE</p> 
사 각 파 젠 창	<p>graph of 1000 samples: SquareParzen</p> 	<p>graph of 5000 samples: SquareParzen</p> 	<p>graph of 10000 samples: SquareParzen</p> 
원 형 파 젠 창	<p>graph of 1000 samples: RoundParzen</p> 	<p>graph of 5000 samples: RoundParzen</p> 	<p>graph of 10000 samples: RoundParzen</p> 

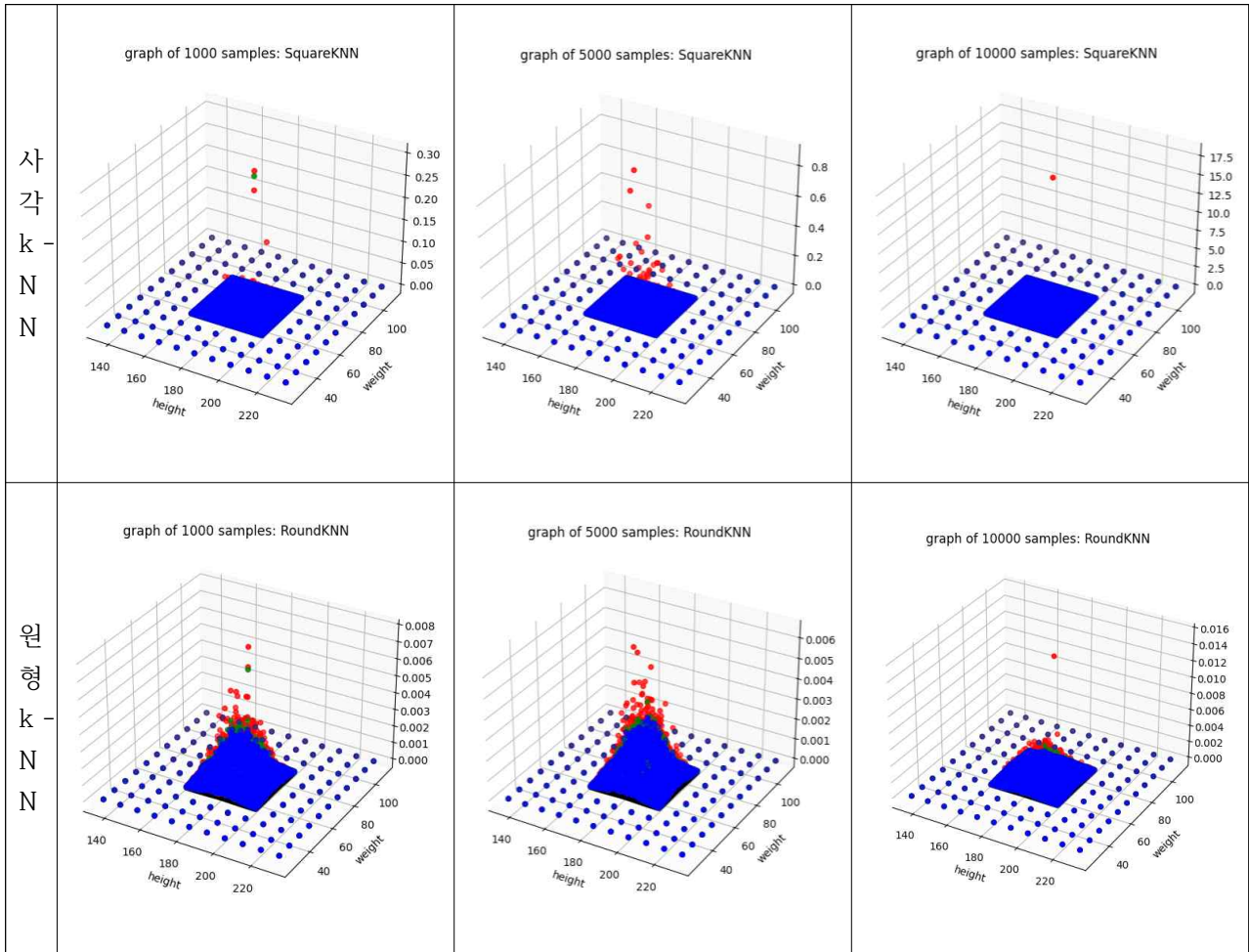


표 4

표 1과 2는 실제 연산 결과 일부를 가져온 것이고, 표 3은 그 연산 결과를 이용해 점을 찍어 그래프를 그린 결과를 나타낸다. 그림 3의 그래프 중 검은색 점은 참값이며, 파젠 창 그래프에서는 빨간색의 창 길이가 σ , 초록색이 2σ 이고 파란색이 3σ 이다. k-NN 그래프에서도 빨간색은 $k=3$, 초록색이 $k=5$ 이고 파란색이 $k=7$ 일 때의 그래프이다. MLE는 노란색으로 그렸지만, 참값과 많이 겹쳐 잘 보이지 않는다. 위 데이터를 보면 대체로 데이터가 평균에 가까울수록(데이터가 밀집되어 있을수록) 오차가 낮았다. 파젠 창 방법은 평균으로부터 멀리 떨어진 데서 확률을 구할 때 확률이 0이 되는 문제도 발생한다는 점도 볼 수 있으며, 평균으로부터 멀리 떨어질 때는 MLE의 오차가 압도적으로 낮지만, 평균 쪽으로 다가오면 다른 방법도 오차가 비슷해진다는 점을 알 수 있다. 위 결과를 통해 데이터 샘플이 1000개 이상일 때 파젠 창은 창 안에 데이터가 들어온다면 창의 크기가 작을수록 오차가 작고, 창의 모양이 사각형이든 원형이든 오차의 큰 차이는 발생하지 않았다는 점도 알 수 있다. k-NN 방법에서는 반대로 k의 값이 커질수록 오차가 줄어드는 경향을 확인할 수 있었고, 사각형 창을 쓸 때는 원형 창을 쓸 때보다 오차가 매우 컸다. 이외에도 MLE는 샘플의 수가 많을수록 오차가 줄어들고 그래프를 보면 k-NN 방법에서 몇몇 점이 비정상적으로 값이 큰 현상을 억제하는 효과가 보이나 MLE 외에는 샘플의 수가 큰 영향은 주지 않았으며, 파젠 창은 평균에 가까워질수록 참값보다 작아지는 경향을, k-NN 방법은 대부분의 상황에서 참값보다 약간 큰 값이 나오는 경향도 발견할 수 있다. 따라서 두 값의 중앙값을 활용한다면 더 정확한 값을 얻을 수 있을 것이다. 파젠 창과 k-NN 방법을 비교해 본다면 오차는 파젠 창이 더 적어 파젠 창을 사용하는 게 유리하나, 파젠 창은 창 안의 데이터가 없을 수 있다는 단점이 있고, k-NN도 그래프를 보면 몇 개의 점이 튀는, 즉 비정상적으로 값이 큰 현상이 발생하는 문제점을 발견할 수 있다.

다음으로 MLE의 파라미터만 따로 모아 비교해 보았다. 이것도 역시 30개의 데이터를 전부 나열할 수 없으므로, 데이터 샘플이 1000개인 데이터의 파라미터 평균, 5000개인 것의 평균, 10000개인 것의 평균으로 비교하였다. 전체 데이터는 별도 첨부한다.

데이터 샘플 수 및 구분	평균		공분산 행렬		
	키	몸무게	std[0][0] 키의 분산	std[0][1], std[1][0] 키와 몸무게의 Correlation	std[1][1] 몸무게의 분산
참값	182.9	67.7	92	0	72
1000개 (데이터)	182.746771 840615	67.6985882 494354	90.641718 8552235	5.92550441069761E-13	71.20284919 50238
1000개 (오차)	-0.0008377 701442551 96	-0.0000208 530364044 938	-0.014763 92548670 06	0	-0.01107153 89580023
5000개 (데이터)	182.924610 107464	67.6990746 537891	91.788481 1756407	-2.02347187223495E-12	72.38730481 02274
5000개 (오차)	0.00013455 498887357 6	-0.0000136 683339856 37	-0.002299 11765607 869	0	0.005379233 47538104
10000개 (데이터)	182.883152 23625	67.7495522 650675	92.272713 9968104	6.83463346831559E-13	71.83090312 52419
10000개 (오차)	-0.0000921 146186427 725	0.00073193 892271050 7	0.0029642 82574026 8	0	-0.00234856 770497305

표 5

위 표 4를 보아 데이터의 샘플 수가 늘어나면 평균과 공분산 모두 오차가 감소한다는 점을 통해 MLE의 파라미터가 참값에 근사한다는 점을 알 수 있다. 데이터만을 보았을 때 보이는 변화는 거의 없는 수준이지만, 오차를 보면 그 절댓값이 점점 작아지고 있다는 점을 알 수 있다.

5. 결론 및 결과 요약

Gaussian 분포를 이루는 데이터를 최대 우도 추정, 파젠 창, k 이웃 찾기 방법을 이용해 Gaussian 분포의 파라미터 또는 확률 분포를 구해보았다. 그 결과 전체적으로는 Gaussian의 최대 우도 추정 방법이 가장 효과가 좋았고, 파젠 창은 창 형태 상관없이 창의 크기가 작을수록, k 이웃 찾기는 Euclidean Distance가 적용될 때 k 값이 클수록 확률 분포를 더 잘 추정할 수 있었다.

참고문헌

독학하는 김박사, 「정규분포의 데이터 셋을 생성하는 randn()과 normal()」, 『tistory』, 2023.6.18.
(<https://lifelong-education-dr-kim.tistory.com/entry/%EC%A0%95%EA%B7%9C%EB%B6%84%ED%8F%AC%EC%9D%98-%EB%8D%B0%EC%9D%B4%ED%84%B0-%EC%85%8B%EC%9D%84-%EC%83%9D%EC%84%B1%ED%95%98%EB%8A%94-randn%EA%B3%BC-normal>)

지미뉴트론 개발일기, 「[Matplotlib] plt로 파이썬 3D (3차원) 그래프 그리는 방법」, 『tistory』, 2013.8.3.
(<https://jimmy-ai.tistory.com/30>)

Luv Dhamija, 「How to set X and Y axis Title」, 『stackoverflow』, 2019.6.8.
(<https://stackoverflow.com/questions/56447079/how-to-set-x-and-y-axis-title>)

OFF THE STAGE, 「[C언어] math함수에서 정확한 PI값 사용하기」, 『tistory』, 2020.12.23.
(<https://lazyreview.tistory.com/114>)

matplotlib, 「List of named colors」, 『matplotlib』.
(https://matplotlib.org/stable/gallery/color/named_colors.html)