데이터 과학

L10: K Nearest Neighbors

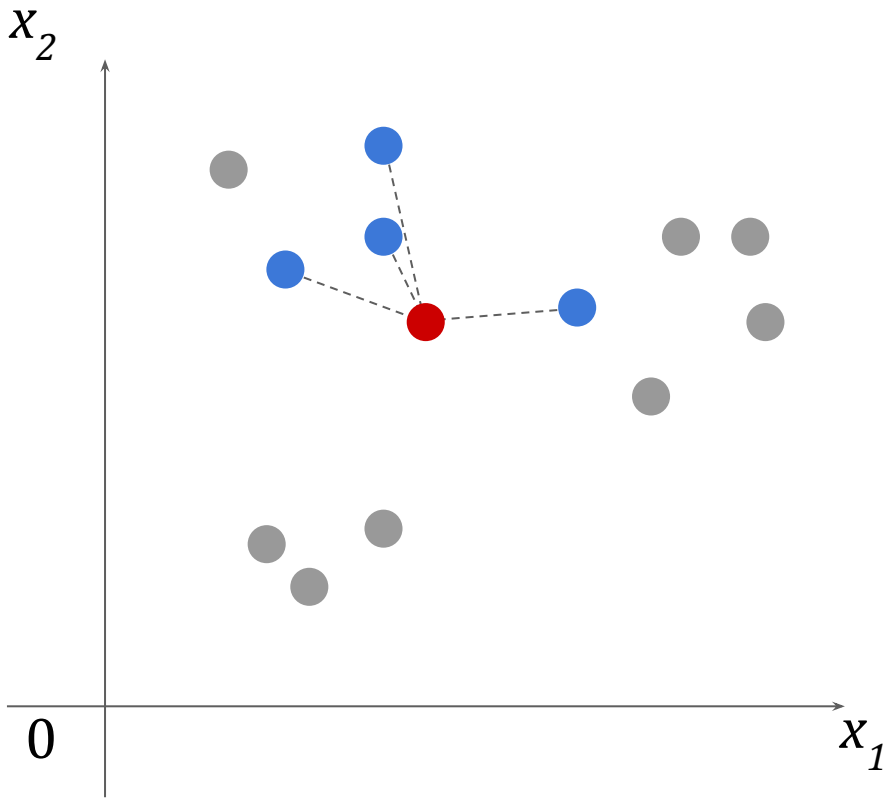
Kookmin University

목차

- **k-Nearest Neighbors**
- Choosing k
- Distances
- Pros and Cons of kNN
- Indexing for kNN

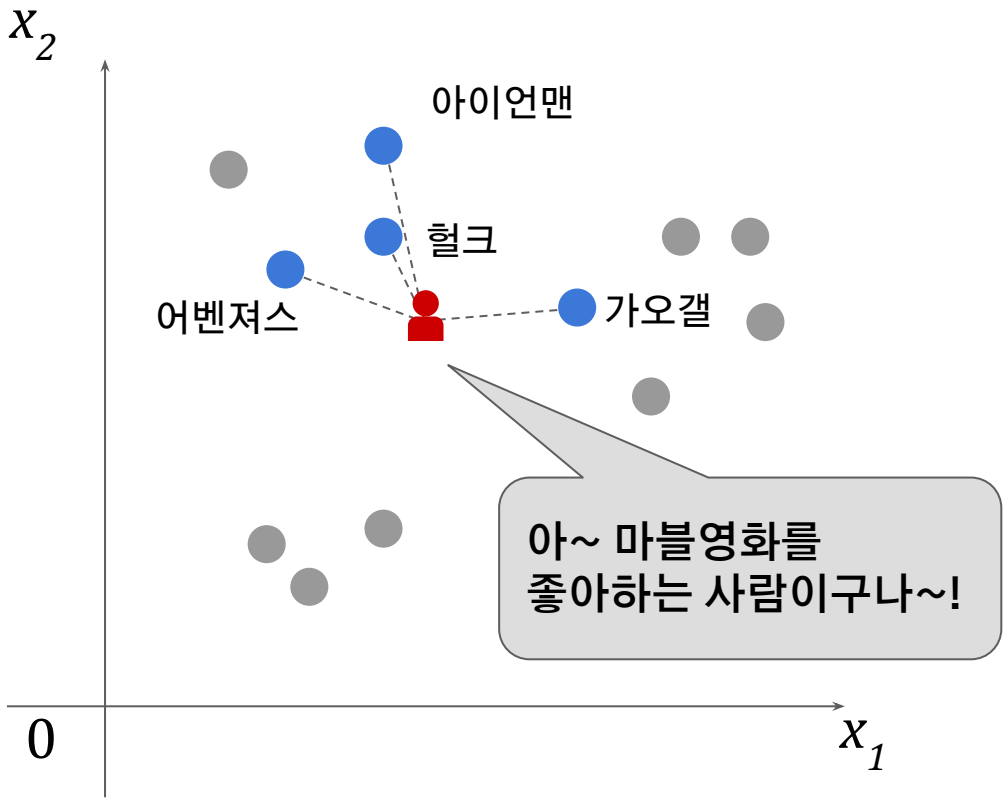
k-Nearest Neighbors

- kNN = 가장 가까운 k개의 점



k-Nearest Neighbors

- 어떤 점의 특성을 알고 싶다면?
  - 가까이 있는 점으로부터 특성 파악 가능

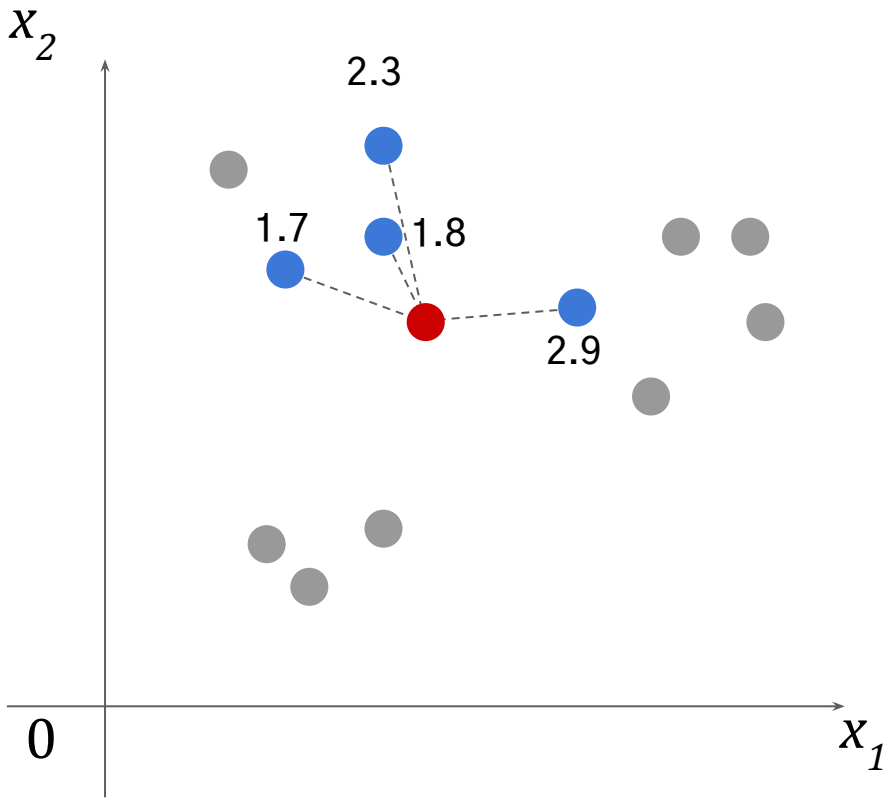


kNN의 특징

- 데이터 기반 분석 방법
- 데이터 분포를 가정하지 않음
- 회귀문제, 분류문제 등에 적용 가능

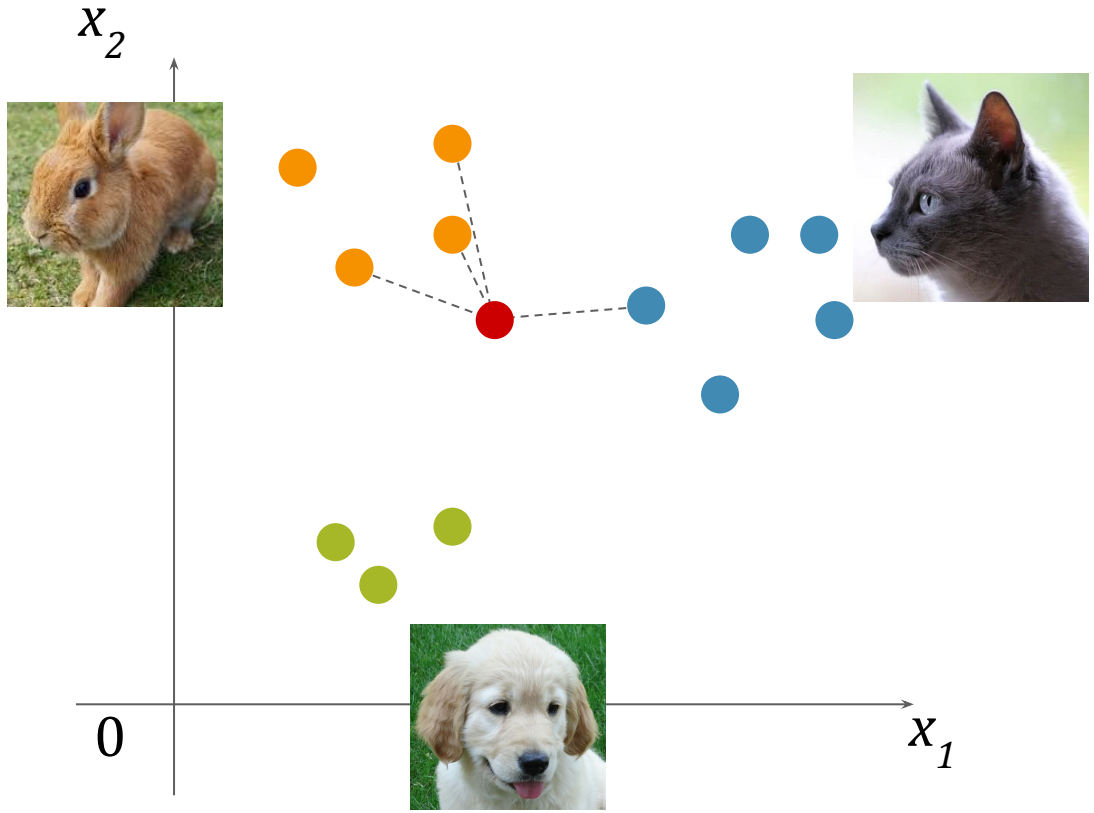
k-Nearest Neighbors

- kNN을 **회귀**문제에 적용 가능
  - kNN의 값을 평균 내어 값 예측



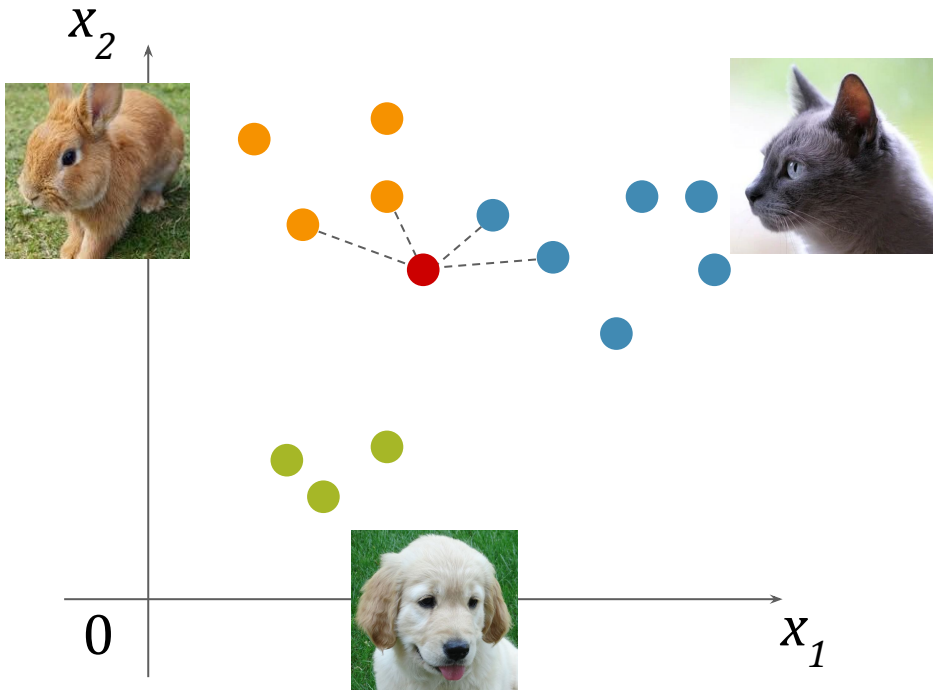
k-Nearest Neighbors

- kNN을 **분류**문제에 적용 가능
  - kNN 중에 가장 많은 항목 선택



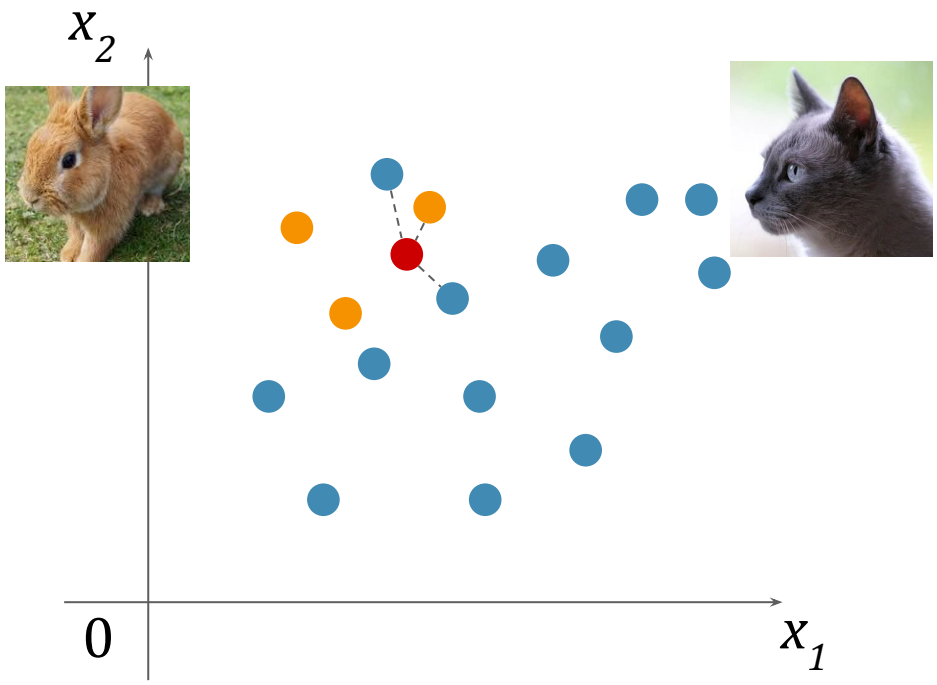
k-Nearest Neighbors

- 1등이 여럿일 경우엔?
  - 방법1. 거리에 따라 가중치 주기
  - 방법2. 단독 1등이 나올 때까지 k를 하나씩 줄이기



k-Nearest Neighbors

- 데이터가 균일하지 않을 경우?
  - cut-off를 조절



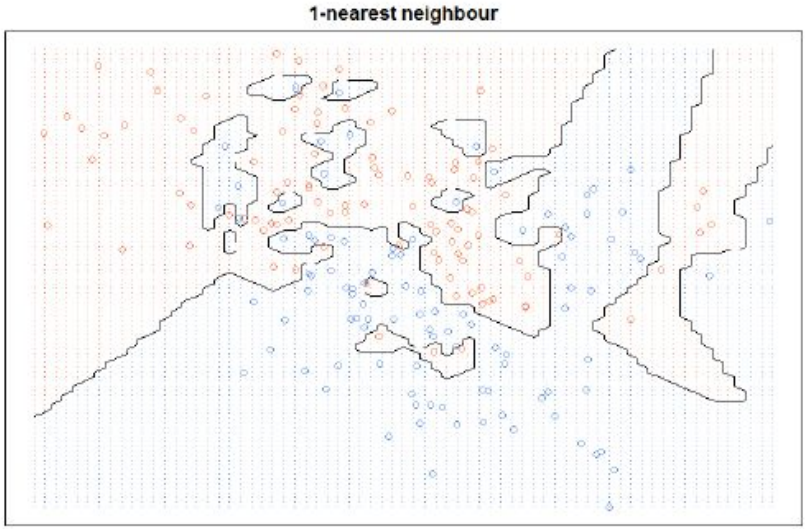
목차

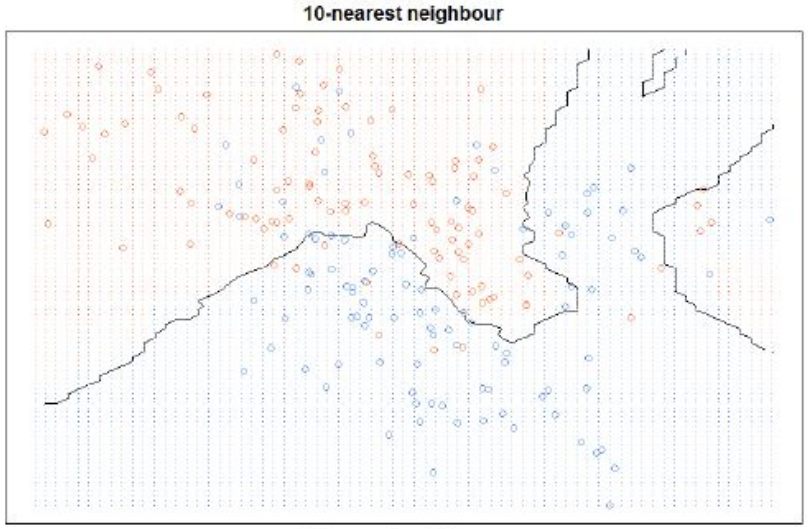
- k-Nearest Neighbors
- **Choosing k**
- Distances
- Pros and Cons of kNN
- Indexing for kNN

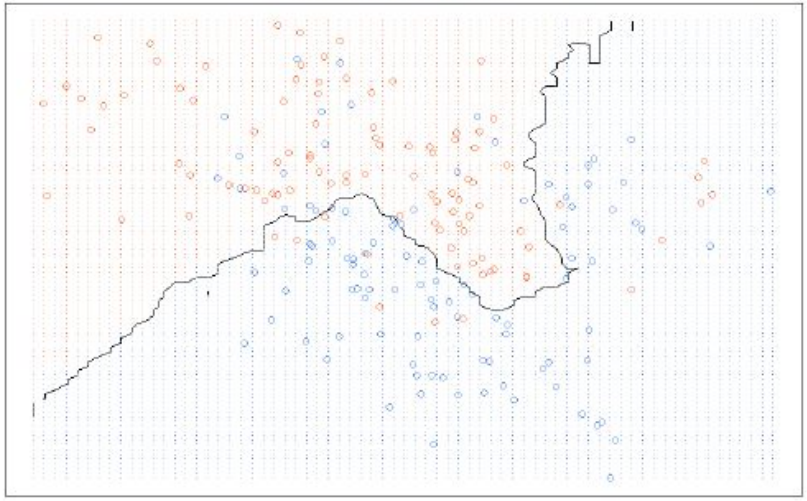
적절한 k값은?

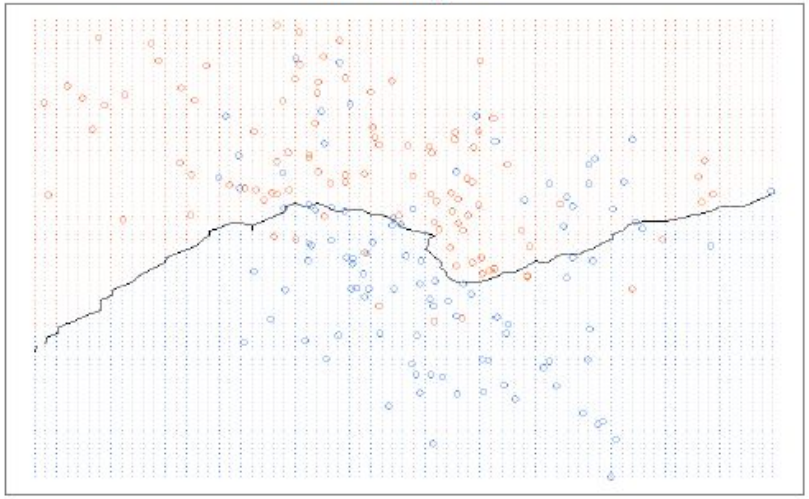
- 데이터마다 적절한 k값이 다름
  - k가 낮다 → 불안정한 결과 ~ 오버피팅
  - k가 높다 → 지나친 일반화 ~ 언더피팅

적절한 k값은?









출처: https://ratsgo.github.io/machine%20learning/2017/04/17/KNN/

적절한 k값은?

- k를 선택하는 방법: 가장 좋은 성능을 내는 값으로 선택
  - k의 값을 1부터 증가시켜가며 각 점들에 대해 knn으로 분류해보고 오류 계산
  - 가장 오류가 적은 k값을 선택

목차

- k-Nearest Neighbors
- Choosing k
- **Distances**
- Pros and Cons of kNN
- Indexing for kNN

거리?

$$X = (x_1, x_2, \cdots, x_n)$$
$$Y = (y_1, y_2, \cdots, y_n)$$

- Euclidean Distance (L2)

$$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Manhattan Distance (L1)

$$\sum_{i=1}^{n} |x_i - y_i|$$

- Cosine Distance

$$1 - \frac{X \cdot Y}{|X||Y|}$$

거리?

- Hamming Distance (곱집합)

$$|\{i \in \{1, 2, \cdots, n\} | x_i \neq y_i\}|$$

$$X = (x_1, x_2, \cdots, x_n)$$
$$Y = (y_1, y_2, \cdots, y_n)$$

- 자카드거리 (집합)

$$1 - \frac{|X \cap Y|}{|X \cup Y|}$$

$$X = \{x_1, x_2, \cdots, x_n\}$$
$$Y = \{y_1, y_2, \cdots, y_m\}$$

거리?

그 외...

- Standardized Euclidean Distance
- Mahalanobis Distance
- Correlation Distance

목차

- k-Nearest Neighbors
- Choosing k
- Distances
- **Pros and Cons of kNN**
- Indexing for kNN

kNN 장단점

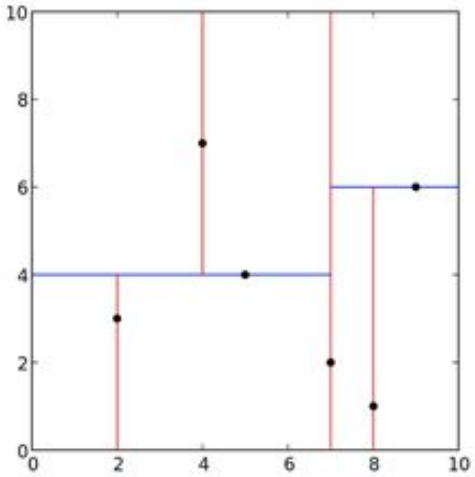
- 장점
  - 쉽고 이해하기 직관적
  - 사전 학습이 필요 없다
  - 어떤 분포든 상관 없음 (비모수 방식)
  - 데이터가 많을 경우 정확도 up! up!
- 단점
  - 데이터가 많을 경우 연산량 up! up!
    - 차원 축소 등으로 계산량 감소
    - 인덱싱으로 탐색 속도 향상

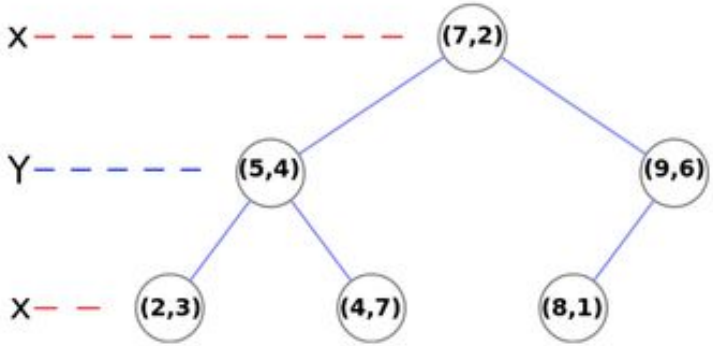
      R-Tree, KD-Tree, KNN-Graph,
      LSH(Locality Sensitive Hashing), etc.
  - 차원의 저주
    - 데이터의 차원이 증가함에 따라 정확도 급하락

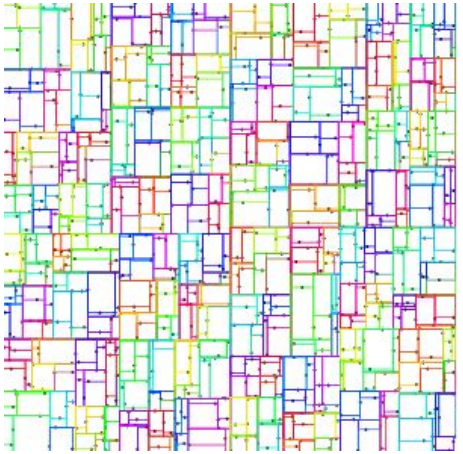
목차

- k-Nearest Neighbors
- Choosing k
- Distances
- Pros and Cons of kNN
- **Indexing for kNN**

KD-Tree

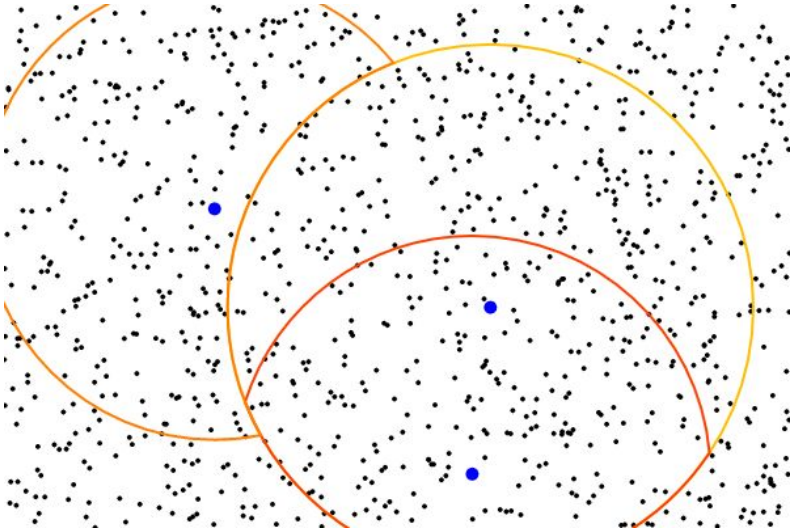






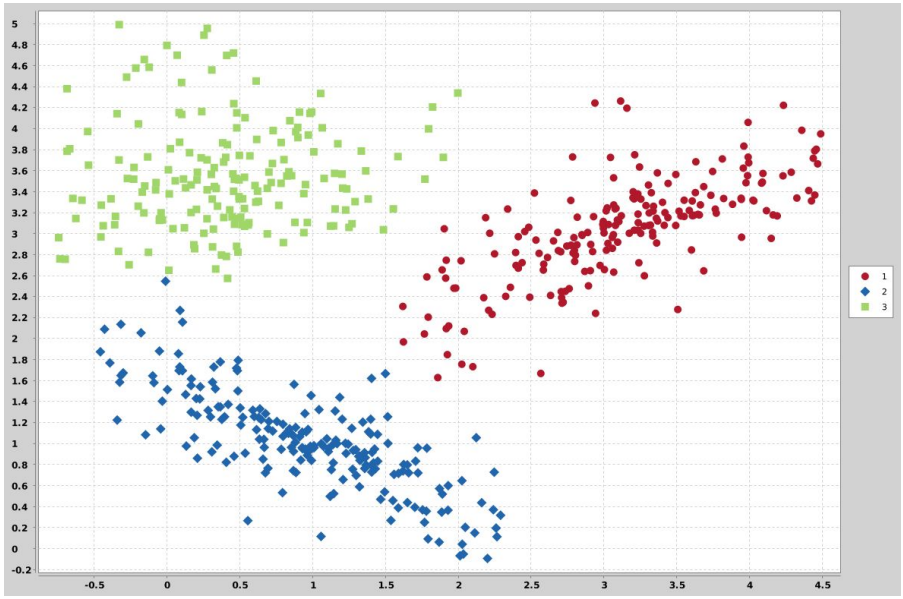
출처: https://medium.com/@schmidt.jerome/k-d-trees-and-nearest-neighbors-81b583860144

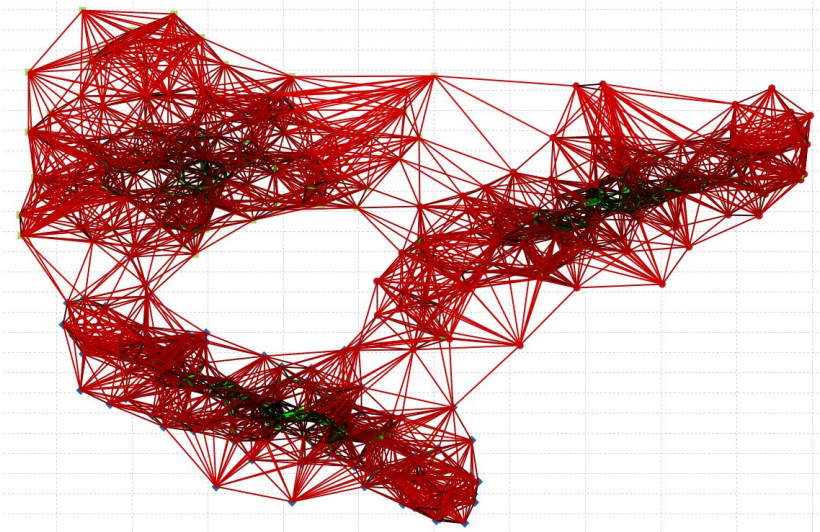
VP-Tree



출처: https://fribbels.github.io/vptree/writeup

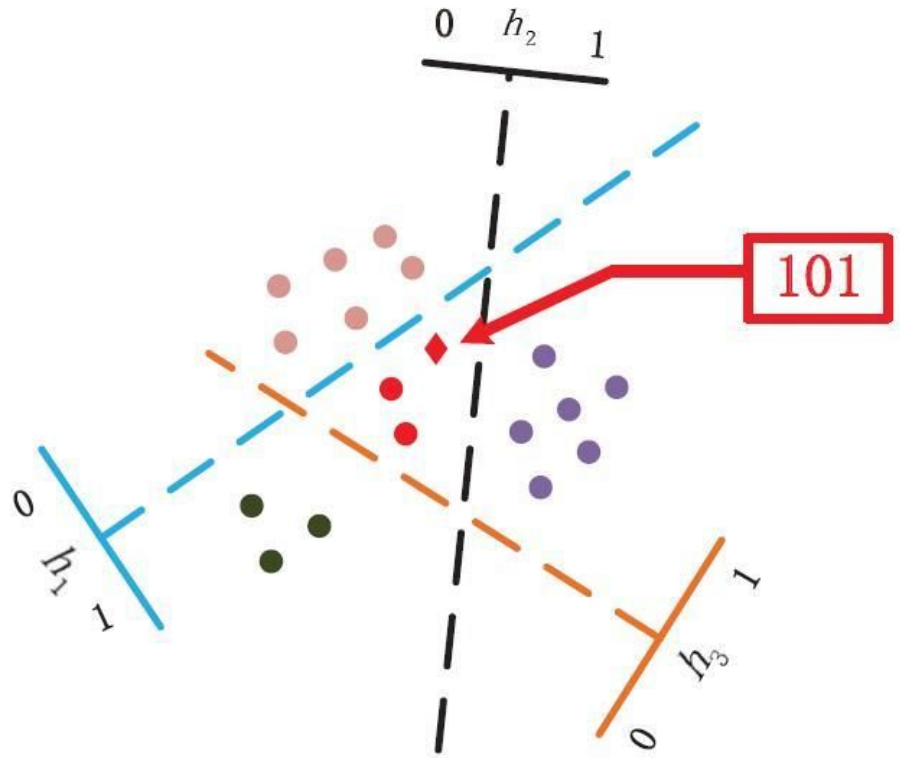
KNN-Graph





출처: https://stats.stackexchange.com/a/187064

Locality Sensitive Hashing



출처: Li, Haisheng, et al. "Feature matching of multi-view 3d models based on hash binary encoding." Neural Network World 27.1 (2017): 95.

Questions?