Affinity Propagation Algorithm

박주찬

DICE Lab
School of Computer Science and Engineering
KOREATECH
green261535@gmail.com

Introduction

비지도 학습의 과업에는 군집화, 밀도 추정, 공간변환 이렇게 크게 세가지로 나눌 수 있다. 이중 군집화의 한 종류인, Affinity Propagation Algorithm에 대해서 알아보려고 한다. Affinity Propagation Algorithm는 모든 데이터가 특정한 기준에 따라 자신을 대표할 대표 데이터를 선택한다. 만약 스스로가 자기 자신을 대표하게 되면 클러스터의 중심이 된다. 군집의 개수를 k라고 했을 때, K-means 알고리즘과는 다르게 이것을 hyperparameter로써 선택해 주는 것이 아니라 자동으로 알아낸다. Responsibility, Availability라는 두 종류의 친밀도 행렬을 이용하여 군집화 한다. Responsibility, Availability행렬이 무엇인지 자세히 알아보고 알고리즘이 어떻게 동작하는지 살펴보려고 한다.

Similarity

$s(i,\ k) = -\|x_i - x_k\|^2$: 음의 거리로 정의되는 유사도이다.

자가 유사도 $s(k,\ k)$

$s(k,\ k)$ 는 특정한 음수 값으로 사용자가 정해주게 된다. 이 값에 따라서 클러스터의 개수가 다라지며 이것은 hyperparameter이다. 유사도의 최솟값, 중앙값, 최댓값 중 하나를 선택하여 모든 샘플에 같은 값을 부여한다. 최솟값을 선택하게 된다면 적은 수의 군집이 나오게 되고, 중앙값을 선택하게 된다면 적절한 군집이 나오게 되며, 최댓값을 선택하게 된다면 많은 수의 군집을 갖게 된다.

Responsibility

책임행렬로써 $r(i,\ k)$와 같이 표기한다. 이 수식의 뜻은 k번째 데이터가 i번째 데이터의 대표가 되어야 한다는 증거이다. k가 i의 대표로 선택되기에 얼마나 적절한지를 나타낸다.

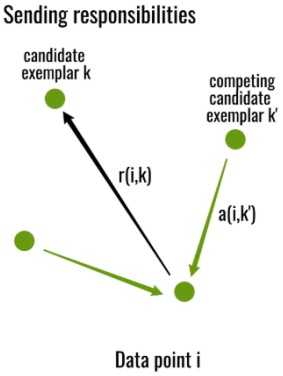

Fig. 1 Responsibility의 관계

$r(i,\ k)$식은 아래 (1)번 식과 같이 표현할 수 있다.

$$r_{ik} = s_{ik} - \max_{k' \neq k}(a_{ik'} + s_{ik'}) \tag{1}$$

r_{ik}는 k번째 데이터가 i번째 데이터의 대표가 되어야 한다는 증거이다.
s_{ik}는 i와 k간의 유사도를 나타낸다.
$\max_{k' \neq k}(a_{ik'} + s_{ik'})$의 의미는 i번째 데이터가 k'번째 데이터를 대표로 선택해야 된다는 근거와 i와 k' 간의 유사도 합의 max값이다. 쉽게 말해서 i와 k가 유사할수록 큰 값을 부여하고, i가 k이외의 다른 샘플과 더 친밀할수록 더 큰 값을 뺀다는 것이다.

Self-responsibility $r(k,\ k)$
k 자기 자신이 자신을 대표한다는 증거이며, 수식은 아래와 같이 표현할 수 있다.

$$r_{kk} = s_{kk} - \max_{k' \neq k}(a_{kk'} + s_{kk'}) \tag{1.1}$$

Availability

가용행렬로써 $a(i,\ k)$와 같이 표기한다. 이 수식의 뜻은 i번째 데이터가 k번째 데이터를 대표로 선택해야 된다는 근거이다. i가 k를 대표로 선택하는게 얼마나 적절한지를 나타낸다.

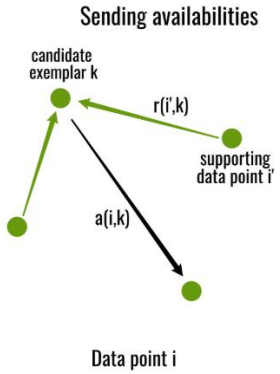

Fig. 2 Availability의 관계

$a(i,\ k)$식은 아래 (2)번 식과 같이 표현할 수 있다.

$$a_{ik} = min\left(0,\ r_{kk} + \sum_{i' \neq i,\ k} max(0,\ r_{i'k})\right),\ i \neq k \tag{2}$$

a_{ik}는 i번째 데이터가 k번째 데이터를 대표로 선택해야 된다는 근거이다.
$r_{i'k}$는 k번째 데이터가 i′ 번째 데이터의 대표가 되어야 한다는 증거이다.
k번째 데이터에게 i′ 번째 데이터가 대표 자격을 더 많이 부여할수록, 즉 $r_{i'k}$가 클수록 a_{ik}가 크다.

Self-responsibility $a(k,\ k)$

k 자기 자신이 자신을 대표로 선택해야 된다는 근거이며, 수식은 아래와 같이 표현할 수 있다.

$$a_{kk} = \sum_{i' \neq i,\ k} max(0,\ r_{i'k}) \tag{2.1}$$

Example

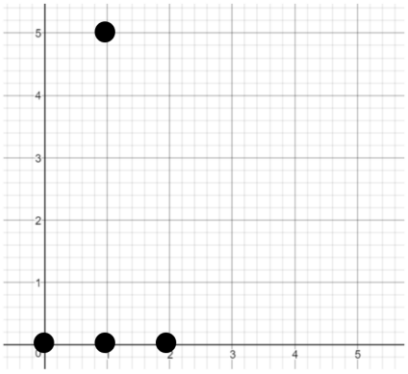

Fig. 3 Example Data

그림 **Fig. 3**와 같이 데이터가 있다고 했을 때를 예를 들어 알고리즘이 어떻게 흘러가는지 알아보도록 하겠다.

$\mathbb{X} = (0,\ 0), (1,\ 0), (2,\ 0), (1,\ 5)$

$s(i,\ k) = \begin{bmatrix} -2 & -1 & -2 & -5.1 \\ -1 & -2 & -1 & -5 \\ -2 & -1 & -2 & -5.1 \\ -5.1 & -5 & -5.1 & -2 \end{bmatrix}, s(k,\ k) = -2$로 설정

$a(i,\ k) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$로 $a(i,\ k)$를 초기화 해준다.

$s(i,\ k) + a(i,\ k) = \begin{bmatrix} -2 & -1 & -2 & -5.1 \\ -1 & -2 & -1 & -5 \\ -2 & -1 & -2 & -5.1 \\ -5.1 & -5 & -5.1 & -2 \end{bmatrix}$이며, 식 (1)와 (1.1)에 의해 $r(i,\ k)$는 아래와 같이 구할 수 있다.

$r(i,\ k) = \begin{bmatrix} -0.5 & 0.5 & -0.5 & -2.05 \\ 0 & -0.5 & 0 & -2 \\ -0.5 & 0.5 & -0.5 & -2.05 \\ -1.55 & -1.5 & -1.55 & 1.5 \end{bmatrix}$이며, 식 (2)와 (2.1)에 의해 $a(i,\ k)$는 아래와 같이 구할 수 있다.

$a(i,\ k) = \begin{bmatrix} 0 & 0 & -0.25 & 0 \\ -0.25 & 0.5 & 0 & 0 \\ -0.25 & 0 & -0.25 & 0 \\ -0.25 & 0 & -0.25 & 0 \end{bmatrix}$이며, $r(i,\ k) + a(i,\ k)$을 구하면 아래와 같으며 각 row의 최대값을 선택해 주면 된다.

$$r(i,\ k) + a(i,\ k) = \begin{bmatrix} -0.5 & 0.5 & -7.5 & -2.05 \\ -0.25 & 0 & -0.25 & -2 \\ -0.75 & 0.5 & -0.5 & -2.05 \\ -1.8 & -1.5 & -1.8 & 1.5 \end{bmatrix}$$

$r(i,\ k) + a(i,\ k)$의 각 row가 의미하는 것은 총 4개 각각의 데이터를 의미하는 것이고, 각 column의 index가 의미하는 것은 그 row의 데이터가 군집의 대표로 설정한 데이터의 index이다. 한마디로 위 $r(i,\ k) + a(i,\ k)$는 총 두개의 군집으로 설정하였고, 군집의 대표는 (1, 0)와 (1, 5)이다. (0, 0)과 (2, 0)은 (1, 0)군집에 속하며, (1, 5)의 군집으로는 (1, 5)하나뿐인 것이다. 다음과 같은 계산을 행렬의 값이 수렴할 때까지 반복하면 된다.

Iteration 1)

$$r(i,\ k) + a(i,\ k) = \begin{bmatrix} -0.5 & 0.5 & -7.5 & -2.05 \\ -0.25 & 0 & -0.25 & -2 \\ -0.75 & 0.5 & -0.5 & -2.05 \\ -1.8 & -1.5 & -1.8 & 1.5 \end{bmatrix}$$

Iteration 2)

$$r(i,\ k) + a(i,\ k) = \begin{bmatrix} -0.6875 & 0.75 & -1.1875 & -3.075 \\ -0.375 & 0.375 & 0.375 & -2.875 \\ -1.1875 & 0.75 & -0.6875 & -3.075 \\ -2.76 & -2.25 & -2.76 & 2.25 \end{bmatrix}$$

Iteration 10)

$$r(i,\ k) + a(i,\ k) = \begin{bmatrix} -0.99 & 0.99 & -1.99 & -4.09 \\ -1.90 & 1.94 & -1.90 & -4.90 \\ -1.99 & 0.99 & -0.99 & -4.09 \\ -4.09 & -2.99 & -4.09 & 2.99 \end{bmatrix}$$

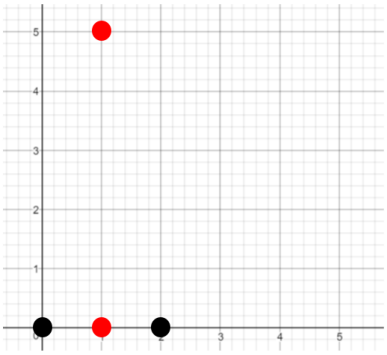

Fig. 4 Affinity Propagation Algorithm Result

Affinity Propagation Algorithm의 결과는 **Fig. 4**와 같다.

Conclusion

K-means의 알고리즘은 군집의 개수와, 초기 군집대표의 값을 정해주어야 하며 초기 군집의 값을 어떻게 설정해주는지에 대해 결과가 달라진다. 하지만 Affinity Propagation Algorithm은 군집의 개수와, 초기 군집대표의 값을 정해주지 않아도 되며, 자가 유사도 $s(k, k)$의 값을 어떻게 설정해주는지에 따라 군집의 개수가 달라진다는 차이점이 있다. 비지도 학습은 크게 군집화, 밀도 추정, 공간 변환으로 나눌 수 있는데 이 중 군집화에 한 종류인 Affinity Propagation Algorithm에 대해 살펴 보았다.

References

1. https://www.geeksforgeeks.org/
2. 오일석, 기계학습, 한빛아카데미, p323~325