國立交通大學管理學院 資訊管理研究所

第五組
智慧型管理決策系統期末書面報告
NBA老闆的慧眼

指導教授：黃思皓 教授
組員：0753428高碩輿 0753432郭昱廷 0753437劉時嘉

一、簡單介紹

1. 概述：

AP聚類算法是同時考慮全部的資料點為可能的聚中心，利用公式計算並迭代，迭代的過程即是在圖中通過傳播一些資訊來找到聚類集合。

2. 相關概念：

- **exemplar**：聚類中心
- **相似度s(i,j)**：

點j作為點i的聚類中心的能力，記為S(i,j)，一般使用負的歐式距離，所以S(i,j)越大，表示兩個點距離越近，相似度也就越高。

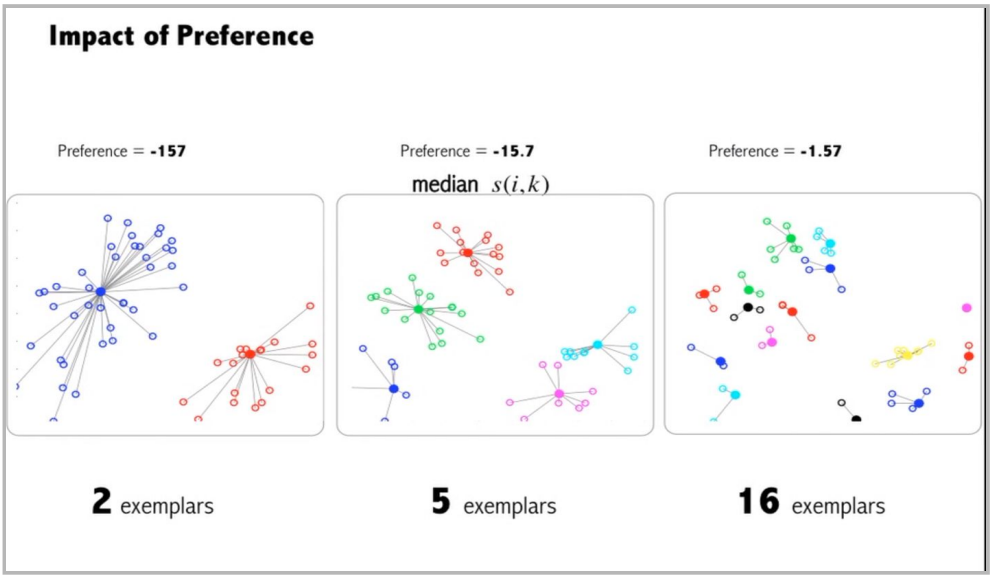
- **相似度矩陣(S)**：

7個點之間兩兩計算相似度，這些相似度就組成了相似度矩陣。

	0	1	2	3	4	5	6
0	0	-43.8879	-61.8803	-40.7106	-53.4096	-37.0728	-30.4061
1	-43.8879	0	-78.4817	-56.1584	-148.784	-74.7505	-43.4472
2	-61.8803	-78.4817	0	-63.1336	-157.829	-74.9038	-99.6224
3	-40.7106	-56.1584	-63.1336	0	-150.405	-108.699	-59.09
4	-53.4096	-148.784	-157.829	-150.405	0	-61.993	-81.6877
5	-37.0728	-74.7505	-74.9038	-108.699	-61.993	0	-47.3916
6	-30.4061	-43.4472	-99.6224	-59.09	-81.6877	-47.3916	0

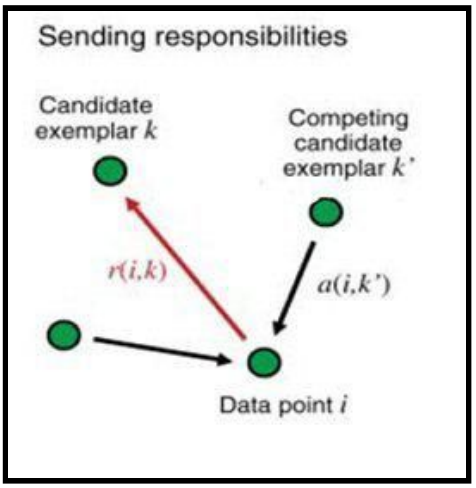
- **參考度 preference**：

P(i) or S(i,i)：指點i作為聚類中心的參考度，為最終聚類數量的參考參數，preference越大表示聚類數量會越多



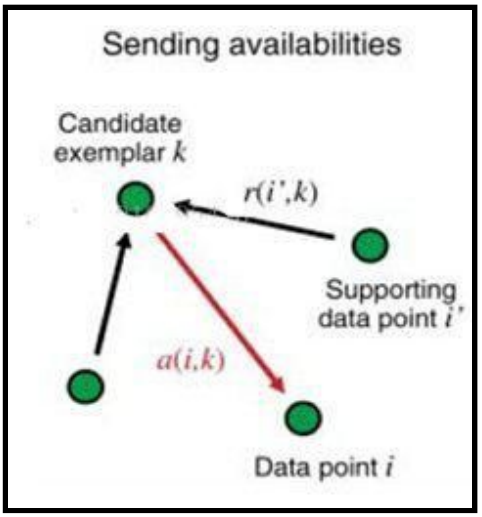
- **吸引度 Responsibility**：

指點k適合做為數個點i的聚類中心的程度，記為r(i,k)。



❖ **歸屬度 Availability**：

指點i選擇點k做為其聚類中心的適合程度，記為a(i,k)。



❖ **阻尼係數 Damping Factor (λ)**：

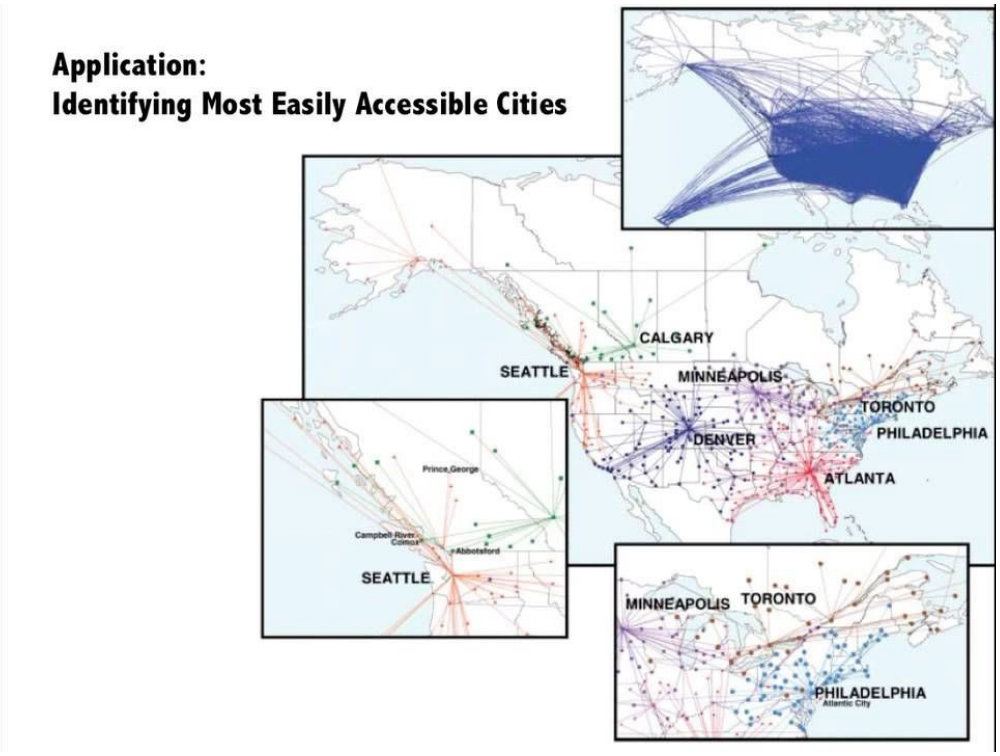
用於演算法的收斂，避免震盪過大。衰減係數是介於0到1之間的實數，將前次迭代更新值的λ倍加上本次信息更新值得1-λ倍，即是阻尼係數避免震盪的方式。

二、文獻回顧

與應用相關之文獻回顧：

kmeans方法中，相似性矩陣是對稱的，即s(i,j)=s(j,i)，但是AP並沒有這個要求。後者甚至適用於不滿足三角關係的情況。

在2007年的論文中，作者分析的美國和加拿大的航班網路（網路也可以看做是一個相似性矩陣，相似性權重為網路連邊上的流量或者別的統計量，本案例中是一個城市到另一個城市的飛行時間）。在這個網路里，相似性是城市之間的飛行時間，這個變數是不滿足三角關係的。例如A->B的平均飛行時間不一定就小於A->C+C->B的平均飛行時間（因為前者可能會沒有直達飛機而要中途在別處轉機，而且地球是一個球體）。



結論:

New York ,Los Angles, Chicago 並非到達其他各地最好的中繼站，因為交通煩忙等因素，反而是Seattle、Denver、Toronto、Atlanta等等。

三、演算法介紹

1.流程：

1) 計算相似度矩陣，並且設置參考度 (使用相似度矩陣的中值)
2) 計算吸引度矩陣，即R值。
3) 在上述求R和求A的公式中，求R需要A，求A需要R，所以R或者A不是一開始就可以求解出的，需要先初始化。
4) 計算歸屬度矩陣，即A值。
5) 根據衰減係數λ對兩個公式進行衰減。
6) 迭代更新R值和A值。
7) 根據求出的聚類中心，對數據進行分類。

2.迭代公式：

1) **吸引度公式**：

$$r(i,k) \leftarrow s(i,k) - \max_{k' \ s.t. \ k' \neq k} \{a(i,k') + s(i,k')\}$$

- 吸引度是相對概念，先前我們有相似度矩陣記錄了k成為i的聚類中心的合適程度，那麼這裡我們只需要證明k比其他節點更合適了就可以了，是否合適其實就是看這兩個節點是否相互認可。
- r(i,k) ：節點 k 對節點 i 的吸引度
- s(i,k')：節點 k' 作為節點 i 的聚類中心的能力
- a(i,k')：節點 i 對節點 k' 的認可程度（歸屬度）

2) **歸屬度公式**：

a) 第一類

$$a(i,k) \leftarrow \min\{0, r(k,k) + \sum_{i' s.t. i' \notin \{i,k\}} \max\{0, r(i',k)\}\}$$

a(i,k)就是節點i選擇節點k為他的類聚中心的適合度。

那這邊先做一個假設，如果節點 k 作為其他節點 i' 的類聚中心的合適度很高，則節點 k 適合作為節點 i 的類聚中心可能性也很大。

那再看回公式的部分，r(i',k)其實就是k對除了i和k本身的吸引度，r(k,k)就是k對自己的吸引度，也就是k自己適合作為聚類中心的程度，整合起來就是k對i以外的節點的吸引度。

b) 第二類

$$a(k,k) \leftarrow \sum_{i' s.t. i' \neq k} \max\{0, r(i',k)\}$$

若 i ＝ k 的情況下，表示為a(k,k)，則r(i',k)就是k對除了k自己以外的其他節點的吸引度。

3) **舉例說明**：
   - 以投票選舉為例
     - 所有人都同時可以選人與被選，要選出1~數人做為代表。
   - 吸引度s：人與人之間固有的偏好程度
   - 吸引力r：個人的魅力，可以被改變
     - r的改變過程即是個人魅力的增減過程
     - 增加就會吸引更多人投給他；反之，減少則降低別人投給他的意願
   - 歸屬度a：
     - 此結果會影響目前選民的想法，換句話說，已經有很多追隨者的人會更有吸引力，吸引力會被改變。
   - 吸引力r與歸屬度a兩者不斷互相迭代、交替的過程，即是選民在每個參選人之間不斷比較的過程。

四、概念發想

NBA近幾年的討論程度越來越熱烈，球員交易或自由市場的買賣也年年備受討論，每一筆交易可說是都傷透球團高層的腦筋，也成為球迷茶餘飯後討論消遣的話題。

所以我們將自己設想為球隊分析師，並希望能遞上一份不讓老闆"打臉"的分群資料，提供球隊高層在球員交易、補強、替代方案上有份合理客觀的依據。

五、操作方法

將NBA所有球員資料(Ex: 薪水、年齡、得分、助攻、有效命中率、真實命中率、球權佔有率、籃板率、效率值…)，以AP演算法實作，計算各球員資料的負歐積里德距離，調整preference、Damping Factor參數，求出k個高質量聚類，且我們將多維度的資料透過PCA()方法壓縮為低維度的資料，實現可視化。

資料來源：
kaggle nba 2017~2018球員數據
https://www.kaggle.com/acasalan/nba-player-stats-201718

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Player Nam	Age	G	GS	MP	PER	TS%	3PAr	FTr	ORB%	DRB%	TRB%	AST%	STL%	BLK%	TOV%	USG%	OWS	DWS	WS	WS/48	OBPM
2	A.J. Hamm	24	22	0	163	8.4	0.472	0.238	0.476	5.4	20.9	12.8	3.8	0.3	7.2	16.4	17.6	-0.2	0.2	0	-0.001	-7.5
3	Aaron Broo	32	65	0	894	9.5	0.507	0.427	0.133	2.3	6.3	4.3	20.7	1.4	0.9	17.2	19.2	-0.2	0.5	0.3	0.016	-2.1
4	Aaron Gor	21	80	72	2298	14.4	0.53	0.309	0.251	5.3	14.1	9.6	10.5	1.4	1.4	8.5	20.1	2	1.7	3.7	0.076	-0.2
5	Aaron Han	22	5	0	17	-2.2	0.102	0.5	0.5	0	19.5	9.7	22.6	0	0	0	12.9	-0.1	0	-0.1	-0.146	-9.6
6	Adreian Pa	25	18	0	135	14.4	0.505	0.278	0.352	7.7	21	14.3	8	3	4.4	11.4	23.1	0	0.2	0.2	0.086	-2.2
7	Al Horford	30	68	68	2193	17.7	0.553	0.302	0.169	4.9	18.6	11.8	24.4	1.2	3.3	11.9	19.8	3.6	2.7	6.3	0.137	1
8	Al Jefferso	32	66	1	931	18.9	0.526	0.002	0.18	9.2	24.2	16.8	11	1	1.5	6.1	26.1	1.2	1.1	2.3	0.119	-1.5
9	Alan Ande	34	30	0	308	5	0.494	0.55	0.2	1.1	7.4	4.4	4.9	0.5	0	7.4	13.7	0	0.1	0.1	0.02	-2.6
10	Alan Willia	24	47	0	708	19.5	0.547	0.004	0.419	14	31.2	22.4	5.2	1.8	3.7	10.5	20.9	1.1	0.9	2.1	0.142	-1.8
11	Alec Burks	25	42	0	653	11.6	0.501	0.306	0.315	3.1	17.7	10.6	7.5	1.4	0.6	11	22.7	-0.1	0.9	0.8	0.056	-3.3
12	Alex Abrin	23	68	6	1055	10.1	0.56	0.724	0.144	1.9	7.1	4.5	5.5	1.7	0.6	8.3	15.9	1.2	0.9	2.1	0.095	-0.3
13	Alex Len	23	77	34	1560	15	0.553	0.026	0.449	10.6	25.3	17.8	4.2	1.1	5.1	15.5	17.6	1.2	1.7	3	0.091	-3
14	Alex Foyth	23	6	1	157	13.2	0.548	0.352	0.185	7.7	12.7	10.2	5.1	0.9	1	4.9	16.9	0.2	0.1	0.3	0.099	-0.6
15	Alexis Ajin	28	39	15	584	12.9	0.529	0.022	0.225	8.3	23.8	16	3.1	1.7	3.1	13.7	17.2	0	0.9	1	0.08	-5.1
16	Al-Farouq	26	61	25	1773	11.3	0.506	0.455	0.292	4.8	23.5	14.1	7.9	1.7	2	15.2	15.4	-0.1	2	1.9	0.051	-2.3
17	Allen Crab	24	79	7	2254	11.6	0.602	0.467	0.192	0.9	10.2	5.5	6.1	1.2	0.7	8.1	14.9	3.1	1	4.1	0.088	0.3
18	Alonzo Ge	29	13	0	89	3.1	0.306	0.214	0.643	5	13.5	9.3	8.2	2.7	0.9	18.2	10.5	-0.1	0.1	-0.1	-0.032	-5.6
19	Amir Johns	29	80	77	1608	15	0.628	0.178	0.27	8.2	17	12.6	13	1.6	3.2	15.7	13.6	2.9	2.1	5	0.149	-0.2
20	Anderson '	34	14	1	92	9.4	0.478	0	0.786	15.3	16.8	16.1	13	1.6	2.6	29.8	12.6	0	0.1	0.2	0.097	-3.2
21	Andre Drui	23	81	81	2409	20.9	0.518	0.008	0.39	15.1	36.3	25.3	5.9	2.6	3.1	12.5	22.4	1.4	5.3	6.7	0.133	-2.1
22	Andre Iguc	33	76	0	1998	14.4	0.624	0.427	0.246	3	13.1	8.3	16.7	1.8	1.5	11.2	11.2	4.1	2.9	6.9	0.167	1.3
23	Andre Rob	25	79	79	2376	9.6	0.51	0.397	0.24	4.6	14.2	9.4	4.6	1.9	2.7	9.1	10.1	1.1	3.1	4.2	0.085	-2.1
24	Andrew Bo	32	27	21	583	9.3	0.46	0.012	0.136	10.6	33.9	21.7	12.3	1.2	3.9	33.4	10.4	-0.3	1	0.7	0.055	-5.6
25	Andrew Ha	22	72	18	1474	8.7	0.477	0.433	0.539	1.7	9	5.3	20	1.9	1.3	16	16.4	0	1.5	1.6	0.051	-2.4
26	Andrew Ni	27	38	0	342	5.9	0.427	0.243	0.126	4.9	14.6	9.8	4.1	2.3	1.5	12.7	17	-0.5	0.4	-0.1	-0.019	-5.9
27	Andrew W	21	82	82	3048	16.5	0.534	0.184	0.345	3.9	8.8	6.3	10.6	1.4	0.8	9.4	29	3.3	0.9	4.2	0.066	0.2
28	Anthony B	23	23	1	264	14.7	0.533	0.522	0.391	10.1	21	15.6	7.1	0.9	0.9	10	19.4	0.3	0.2	0.5	0.09	-0.8
29	Anthony B	24	11	0	159	7.2	0.43	0.54	0	5.3	16.9	11	7.6	1.5	0.5	10.7	15.6	-0.2	0.2	0	0.002	-3.8
30	Anthony D	23	75	75	2708	27.5	0.579	0.088	0.424	6.7	27.9	17.3	11.1	1.7	5.1	9.1	32.6	5.9	5.1	11	0.196	1.4
31	Anthony M	31	49	7	714	10	0.505	0.528	0.147	1.9	2.9	2.4	5.2	1.7	0.2	2.2	16.4	0.7	0.5	1.2	0.083	-1.2
32	Anthony T	31	65	9	1477	11.1	0.595	0.655	0.234	4	14.5	9.3	7.7	1.1	1.2	12.6	13.6	1.5	0.8	2.3	0.075	0.2
33	Archie Goo	22	15	0	214	18.7	0.651	0.227	0.636	3.5	10.3	6.9	18.5	0.9	1.8	14.2	19.8	0.6	0.1	0.7	0.153	0.6

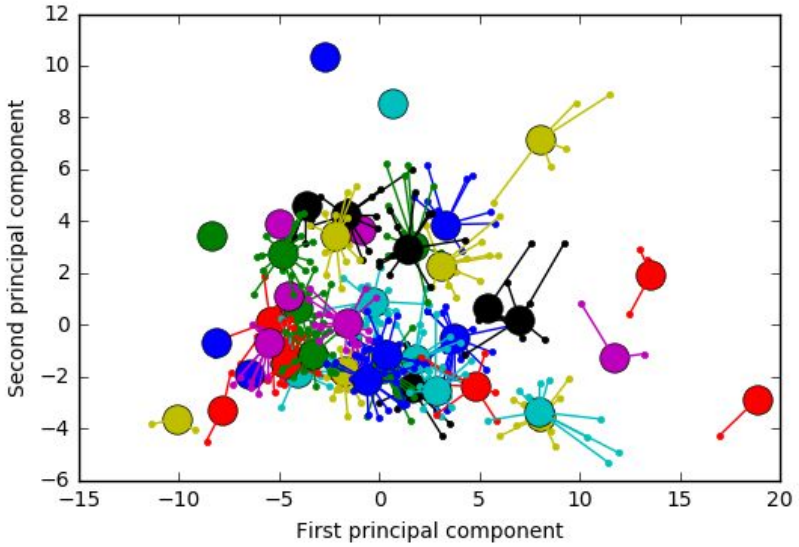
六、實驗結果

1. 調整參數

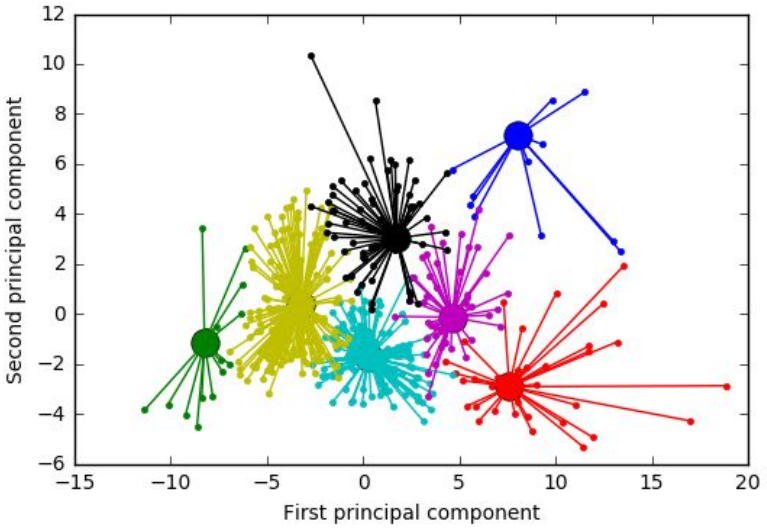
整個AP演算法中有兩個參數必須是人工輸入的，分別是參考度及阻尼係數，所以我們希望以這兩個參數來跑一些實驗。

- Preference：

AP演算法中Preference的大小會決定最後群及數量的多寡，當Preference設中位數時(-62.5683289159)，最後得到的群集數為43個，而把參考度調小至-500時，最後收斂群集則剩7群。

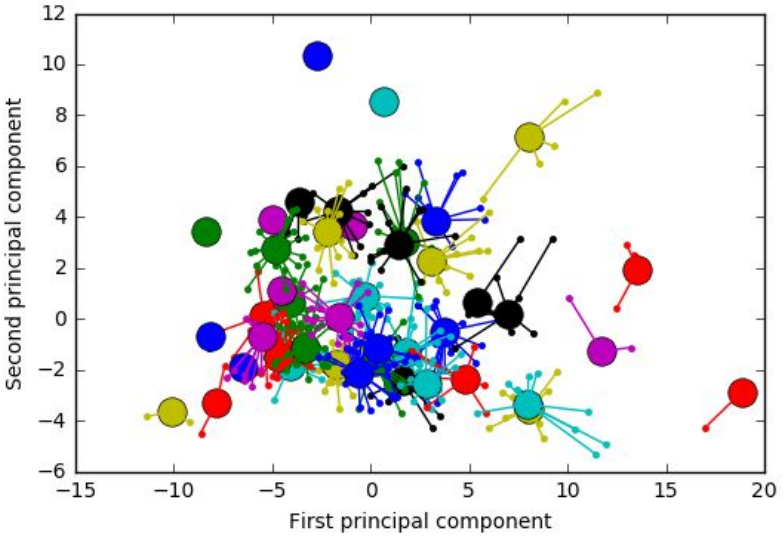


Preference = 中位數(-62.5683289159)
群集數: 43



Preference = -500
群集數: 7

- DampingFactor：
  AP演算法中的阻尼係數則是用於演算法的收斂，並且會影響收斂的速度，實驗結果發現，當Damping Factor = 0.5(較小)時，演算法跑的時間為0.8640606880187988秒，而提高到0.9時，時間則增加到了1.2220699787139893秒，可見當阻尼係數設越高時，收斂時間最較多。



Damping Factor = 0.5
Time = 0.8640606880187988 seconds.

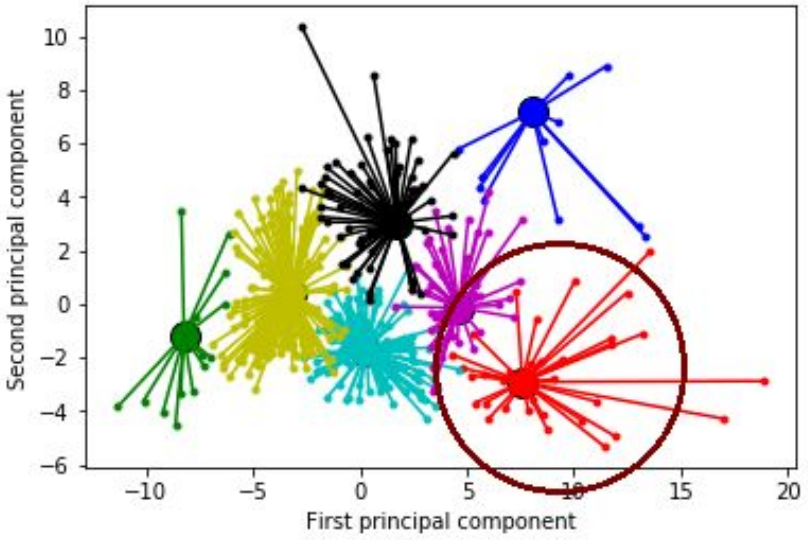
Damping Factor = 0.9
Time = 1.2220699787139893 seconds.

2. 所有球員資料：

在每一次跑AP演算法後，都會輸出一份csv檔，把所有的球員最後是分到哪一群顯示出來。下圖我們以紅色框內的第三群及擷取csv檔中被分到第三群的球員來說明。

假如是NBA灰狼隊的老闆，球隊中有Andrew Wiggins這名球員，但假設他在今年球季結束投入自由市場並加入其他球隊，身為灰狼隊的老闆勢必要填補這個空缺，但NBA有500多位球員，總不可能大海撈針、隨便挑選補強選手，所以他就可以依據與Wiggins同樣分在第三群的這些球員當作補強考量，因為在數據表現上他們是相近的。

但這樣做還是有缺陷，因為在籃球場上，每個球員被分配到的定位不同，意思是Wiggins在球場上扮演小前鋒的位置，但這份資料並不適所有球員都是小前鋒，所以做補強時這點可能會被拿來探討，所以我們之後又有做更仔細的分析。



30	Andrew Wiggins	3
31	Blake Griffin	3
32	Bradley Beal	3
33	C.J. McCollum	3
34	Carmelo Anthony	3
35	Chris Paul	3
36	Damian Lillard	3
37	Danilo Gallinari	3
38	DeMar DeRozan	3
39	DeMarcus Cousins	3
40	Dennis Schroder	3
41	Devin Booker	3
42	Elfrid Payton	3
43	Eric Bledsoe	3
44	Giannis Antetokounmpo	3
45	Goran Dragic	3
46	Gordon Hayward	3
47	Isaiah Thomas	3
48	James Harden	3
49	Jeff Teague	3
50	Jimmy Butler	3
51	John Wall	3
52	Jrue Holiday	3
53	Kawhi Leonard	3
54	Kemba Walker	3
55	Kevin Durant	3
56	Klay Thompson	3

3. 依球員打球的位置分群:

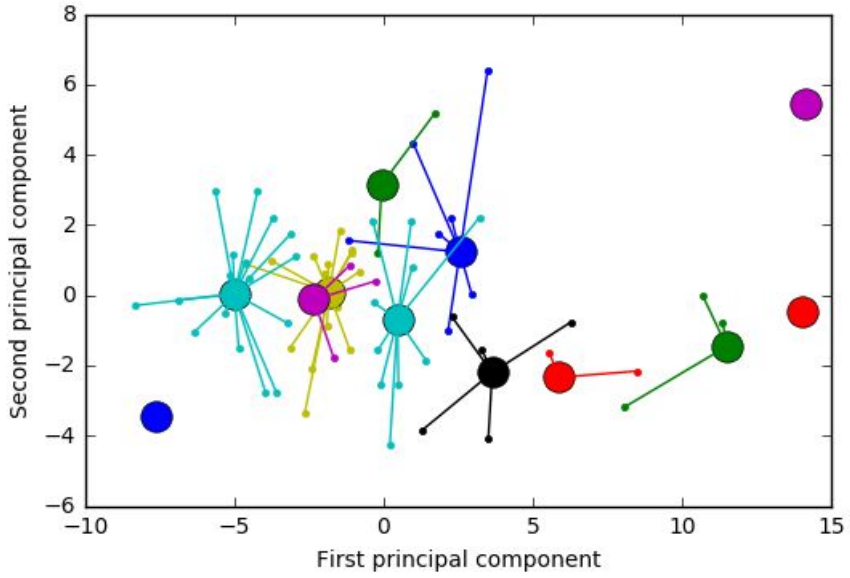
以小前鋒的所有球員為例來跑AP演算法，得到以下結果，共分為12群，每個聚類中心的球員可以說是那一群的模板。

我們以第九群及第十群做說明，第十群聚類裡只有一名球員LeBron James，當然她也是那群的聚類中心，James可以說是當今籃球界的第一人，但最近幾年來球迷間或朋友間總有風聲說他是不是已經老了、不強了、有越來越多後起之輩已經慢慢可以跟他比肩、甚至已經超越他了，但在我們這份分析中，在小前鋒的位置而言，第九群的四位球員可以說是這位置上的全明星球員，但始終還是無法與James被劃分在同一群，也就是說以風格打法、客觀數據表現而言，James還是一樣如同怪物般的存在，無法被取代。

所以當球隊老闆擁有James這名球員時，可以很清楚了解到如果有任何人想跟我談條件要交易James時，不可能以一對一的交換，籌碼一定要提高，因為在小前鋒位置上還沒有任何球員可以與他對等。

- SF

Preference = 中位數
群集數: 12

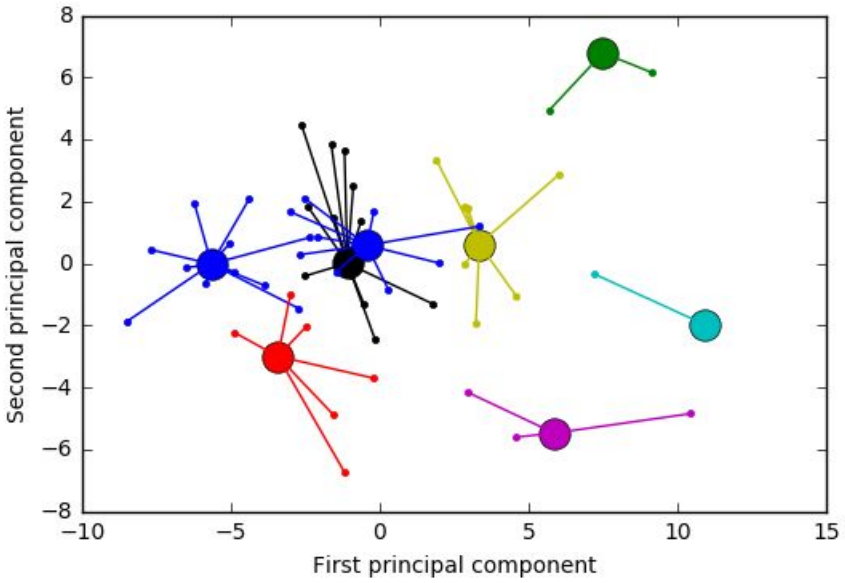


聚類中心1:Aaron Gordon → 8個球員
聚類中心2:Al-Farouq Aminu → 3個球員
聚類中心3:Carmelo Anthony → 3個球員
聚類中心4:DeMarre Carroll → 11個球員
聚類中心5:Giannis Antetokounmpo → 1個球員
聚類中心6:Glenn Robinson → 16個球員
聚類中心7:Jae Crowder → 6個球員
聚類中心8:Jarell Eddie → 1個球員
聚類中心9:Kawhi Leonard → 4個球員
聚類中心10:LeBron James → 1個球員
聚類中心11:Paul Zipser→ 18個球員
聚類中心12:Tyreke Evans → 4個球員

37	Luol Deng	6
38	Mindaugas Kuzr	6
39	Omri Casspi	6
40	Richard Jefferso	6
41	Sam Dekker	6
42	Shabazz Muham	6
43	Andre Iguodala	7
44	Bojan Bogdanov	7
45	Danilo Gallinari	7
46	Jae Crowder	7
47	Otto Porter	7
48	Trevor Ariza	7
49	Jarell Eddie	8
50	Gordon Haywar	9
51	Jimmy Butler	9
52	Kawhi Leonard	9
53	Kevin Durant	9
54	LeBron James	10
55	Chandler Parson	11
56	Corey Brewer	11

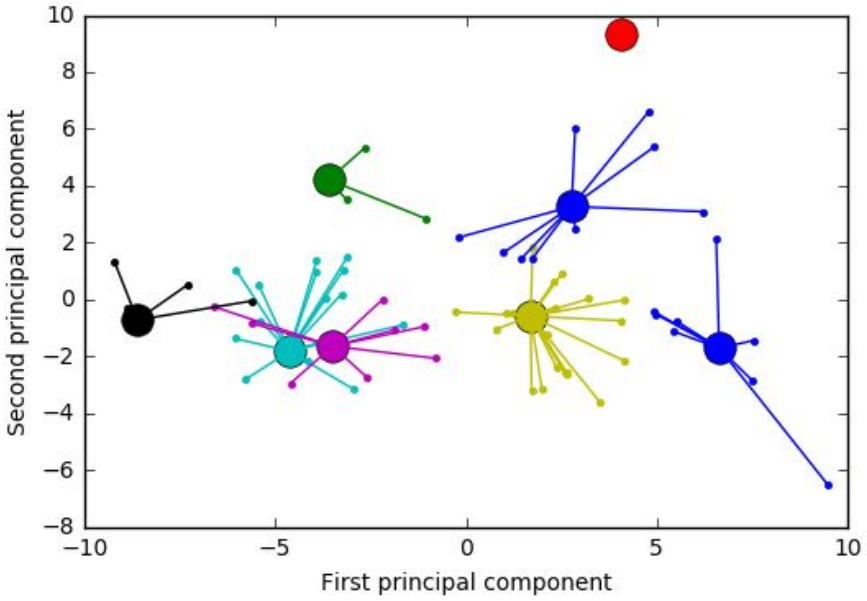
- C

Preference = 中位数
群集数: 8



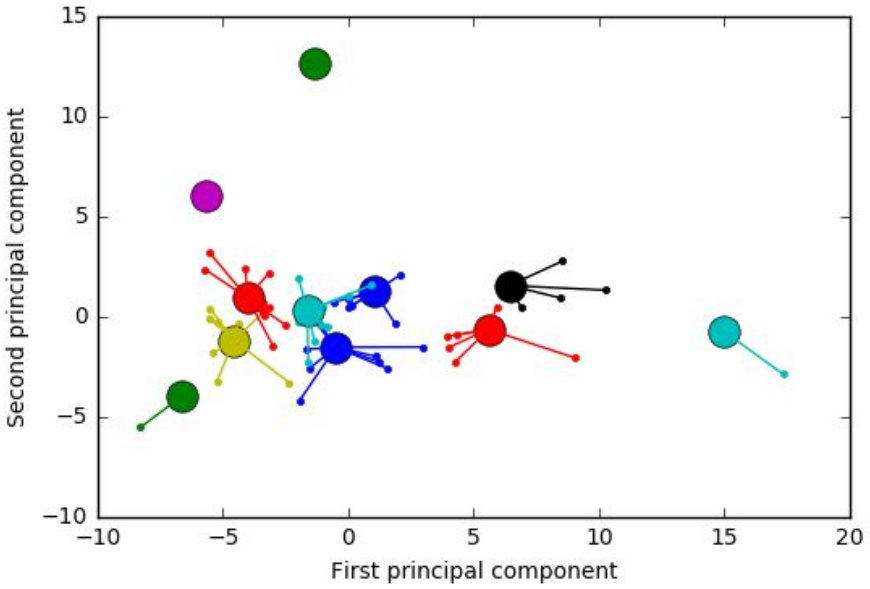
- PF

Preference = 中位数
群集数: 8



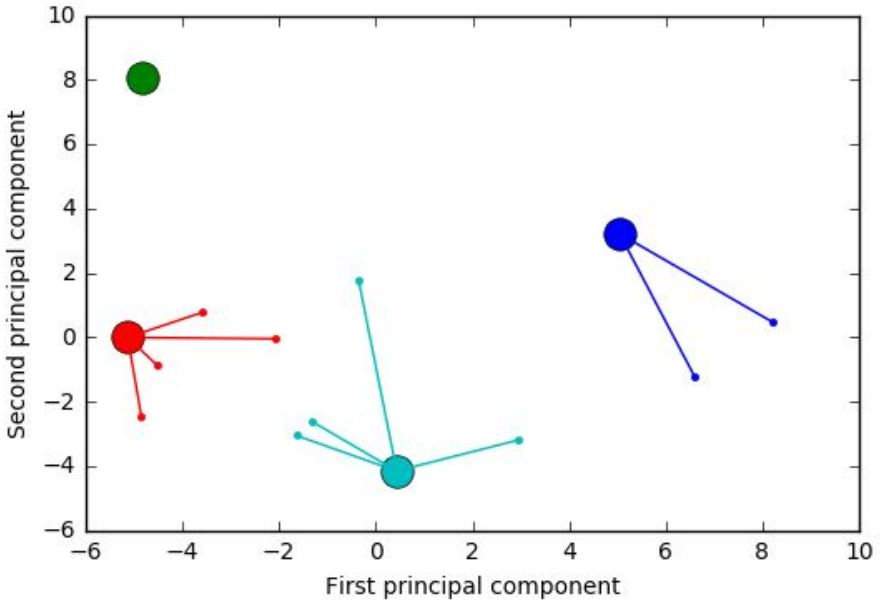
- PG

Preference = 中位數
群集數: 11



- SG

Preference = 中位數
群集數: 4



4. 依不同性質分析:

除了以球員位置上來做分析之外，也可以以球隊不同的性質需求來輸入不同的數據來做分析。

舉例來說，當球隊老闆覺得自己的球隊進攻球員已經足夠了，他想要以防守性質面來挑選球員，那這樣就不需要輸入全部的數據屬性，只需要餵一些防守性質的數據而已(ex：籃板、抄截、防守時對手命中率等等…

七、結論與延伸探討

1.Why Use AP ?

- 非監督聚類方法
- 未知聚類總數
- 聚類結果不會隨機變化

在實作上，我們只需將data丟入AP演算法中，他便會自己計算、自己分類分群，最後找到自己的結果，在過程中，我們只需預設好兩個變數damping factor、preference，而在多次計算運形後，結果都不會隨機變化。

2.Any imperfect ?

然而在我們丟入的data有其盲點存在，NBA的老闆或球隊在選擇球員上，並不會只觀看球員在場上的數據表現，還包括許多因素，事無法量化或在數據表現出來的，如下:

- 商業價值 ex: Dwyane Wade

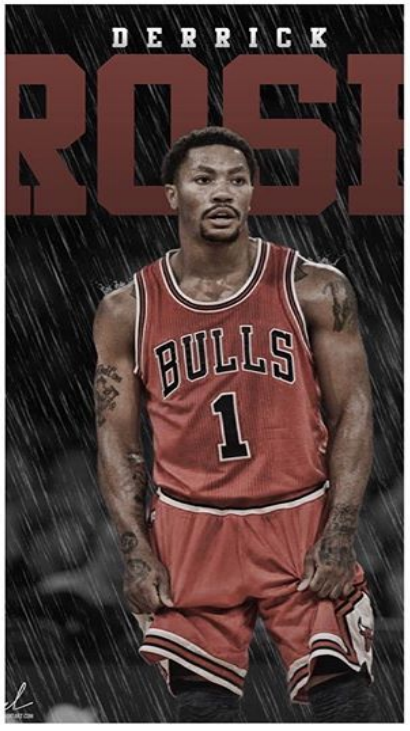
Wade是大家耳熟能詳的巨星，層經出現在2K的封面上，但最近的數據表現並不如其他球員出色，但仍是有淺力的球員，也累積許多粉絲，粉絲能帶來商業價值。

- 球員背景 ex: Jeremy Lin

  Jeremy是著名的台裔球員，能吸引亞洲地區的球迷觀注其球隊，帶來利益。

- 健康因素 ex: Derrick Rose

  Rose是最年輕獲選MVP的球員，但由於前幾賽季腳傷原因，造成都個賽季缺賽，而傷後復出便不如以前突出，這也是無法顯現在數據上的。



(還有包括球員心理狀態、球員間磨合度…)

八、參考資料

https://zi.media/@yidianzixun/post/uKZUFF?fbclid=IwAR1Dib-4T5MkM2x_helYBtq0TgTD871fAixs-4sBivOUXUNjuMl07xFahOs

https://www.cnblogs.com/huadongw/p/4202492.html?fbclid=IwAR3PHEOcrAar0rCI1e9AeYBxV0fl6JKJIRqGAJ3ag9BnlleT5M9v_8oNiy0

https://www.itread01.com/articles/1495821610.html?fbclid=IwAR1Ssz1ZCdqnxtT6obwpx0AFRDBEGrqKK0BXmMGPK-I1IvNwqJuif9EhZt0

https://scikit-learn.org/stable/auto_examples/applications/plot_stock_market.html?fbclid=IwAR29Qcy3SrWFDjHAJGF--XO-0zW2FWt8t4dYZT0lsYxUDqJpBz17ds2pQQM

https://scikit-learn.org/stable/auto_examples/cluster/plot_affinity_propagation.html?fbclid=IwAR2SsWtrb5MJdypMha6G59nWSeEwf7QK5vzmCa_UefR9mHDT0WzLgEbqng4#sphx-glr-auto-examples-cluster-plot-affinity-propagation-py