Thuật toán Lloyd có thể bỏ lỡ một vài cụm rõ ràng với mắt người. Vì phải chọn 1 điểm cố định cho 1 cụm duy nhất, thuật toán này vô nghãi với các điểm giữa hoặc các điểm cách 2 trung tâm xấp xỉ bằng nhau.

Mục tiêu mới là gán điểm dữ liệu linh hoạt hơn, một điểm có thể thuộc một phần cụm này và 1 phần cụm kia.

Búng đồng xu với sự thiên vị không xác định: trong 1 vở kịch, 2 nhân vật búng một đồng xu và nó đáp mặt ngửa 157 lần liên tiếp.

Xác suất đồng xu lên là số lần đầu ngửa chia cho tổng số lần lật.

Để phức tạp hoá vấn đề, giả sử có 2 đồng xu giống nhau nhưng có sự thiên vị không rõ ràng khác nhau *θA* và *θB* . Sau khi quan sát hàng loạt lần tung đồng xu, ta thu nhập lại và gán là tham số.

Ta đơn giản vấn đề bằng cách cho rằng với mỗi n lần tung, 2 đồng xu sẽ có thể bí mật bị tráo hoặc không.

Ta đặt một chuỗi 5 cụm có xác suất ra mặt ngửa như sau:

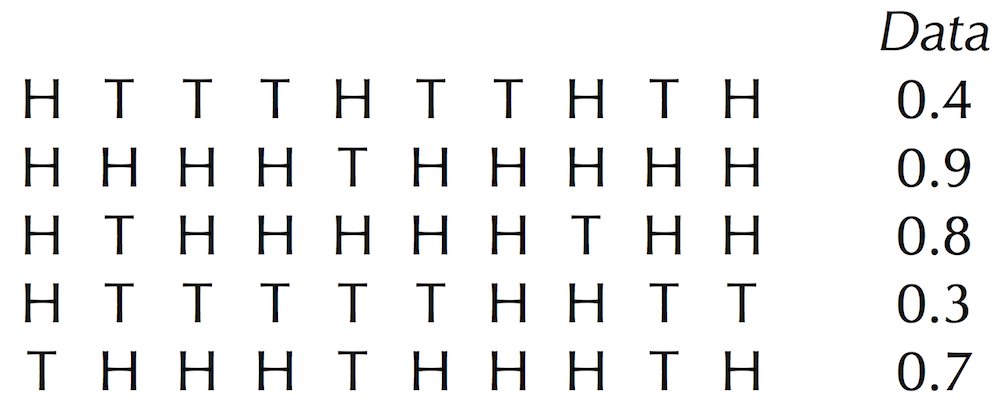
*Data* = (*Data*1, *Data*2, *Data*3, *Data*4, *Data*5) = (0.4, 0.9, 0.8, 0.3, 0.7)

Giả sử ta biết lần thứ 1 và 4 xài đồng xu A:

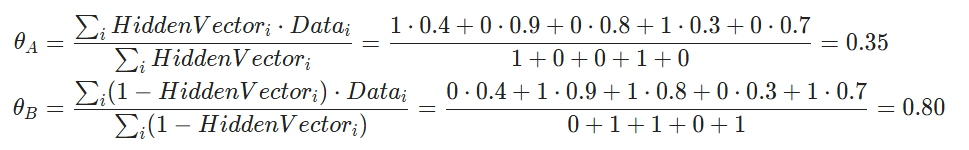


Các lần còn lại sẽ là của đồng xu B:





Biểu diễn việc chọn 1 trong 2 đồng xu bằng vecto HiddenVector = (1, 0, 0, 1, 0), 1 ở vị trí k là đồng xu A còn 0 là đồng xu B.



Nếu có sẵn tập hợp tham số, ta sẽ lập HiddenVector dựa trên xu A hoặc xu B xem xu nào có vẻ xác suất n lần đó nhiều hơn. Ví dụ biết xác suất ngửa của A là 0.6 và B là 0.82, lần thứ 5 có 7 lần ngửa, 3 lần sấp.

0.6^7 \* (1 – 0.6)^3 = 0.00179

0.82^7 \* (1 – 0.82)^3 = 0.00145

P(A) > P(B) => gán vị trí thứ 5 là 1

Tổng quát hơn, hãy để *Pr* (*Datai*|*θ*) biểu thị **xác suất có điều kiện** của việc tạo ra *Datai* cho một đồng xu có độ lệch θ,

*Pr* (*Datai*|*θ*) = *θn · Datai*(1 - *θ*)*n*· (1 -*Datai*).

Nếu *Pr*(*Datai*|*θA*) > *Pr* (*Datai*|*θB*), sau đó đồng xu *A* có nhiều khả năng đã tạo ra chuỗi lật thứ *i* và đặt *HiddenVectori*bằng 1. Nếu *Pr*(*Datai*|*θA*) < *Pr* (*Datai*|*θB*), sau đó đồng xu *B* có nhiều khả năng hơn, và chúng tôi đặt *HiddenVectori*bằng 0. Cà vạt bị đứt một cách tùy tiện.

Tóm lại, nếu *HiddenVector* được biết đến và *Tham số* là không xác định, thì chúng ta có thể xây dựng lại *các Tham số có*khả năng nhất = (*θA*,*θB*):

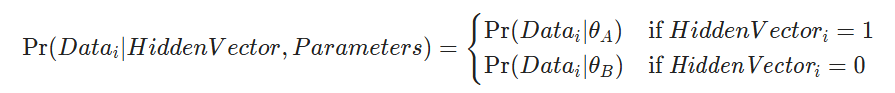
(*Data*, *HiddenVector*, ?) → *Parameters*

Tương tự như vậy, nếu *Parameters* được biết đến và *HiddenVector* không xác định, thì chúng ta có thể xây dựng lại *HiddenVector* có khả năng nhất:

(*Data*, *Parameters*, ?) → *HiddenVector*

Tuy nhiên, vấn đề ban đầu của chúng tôi là cả *HiddenVector* và *Parameters* đều không xác định:

(*Data*, ?, ?) → ???



Pr(*Data*|*HiddenVector*, *Parameters*) = ∏*ni*=1*Pr*(*Datai*|*HiddenVector*, *Parameters*).

Bắt đầu từ một lựa chọn tùy ý của *Tham số* = (*θA*,*θB*) và ngay lập tức xây dựng lại *HiddenVector* có khả năng nhất:

(*Data*, ?, *Parameters*) → *HiddenVector*

Ngay sau khi chúng ta biết *HiddenVector*, chúng ta sẽ đặt câu hỏi về sự khôn ngoan trong lựa chọn ban đầu của chúng ta về *Tham số* và ước tính lại *Tham số* ':

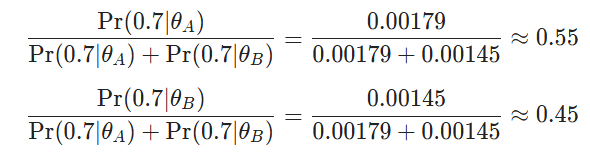
(*Data*, *HiddenVector*, ? )  → *Parameters*

Sự khác biệt duy nhất giữa thuật toán lật đồng xu và thuật toán Lloyd cho phân cụm *k* là cách thực hiện bước "Trung tâm đến cụm". Cái đầu tiên tính toán *HiddenVectori* bằng cách so sánh *Pr* (*Datai|θA*) với *Pr* (*Datai|θB*), trong khi ở cái sau, chúng ta gán một điểm cho cụm chứa tâm gần điểm đó nhất.

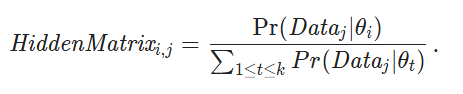
Sử dụng phép so sánh ném đồng xu để giải thích một phiên bản mềm của phương pháp phân cụm k-means. Với các tham số = (θA, θB), chúng ta có thể đưa ra quyết định cứng cho HiddenVector bằng cách so sánh Pr(Datai|θA) với Pr(Datai|θB). Tuy nhiên, điều này không có nghĩa là chúng ta chắc chắn xu nào đã được sử dụng. Nếu Pr(Datai|θB) xấp xỉ bằng Pr(Datai|θA), thì độ tin cậy rằng xu B đã được sử dụng sẽ vào khoảng 50%. Mặt khác, nếu Pr(Datai|θB) lớn hơn rất nhiều so với Pr(Datai|θA), thì chúng ta gần như chắc chắn rằng xu B đã được sử dụng. Nói chung, chúng ta có thể nói về độ tin cậy rằng một xu đã được sử dụng như là "trách nhiệm" của xu đó đối với một chuỗi các lần ném. (Các trách nhiệm này nên cộng lại thành 1.)

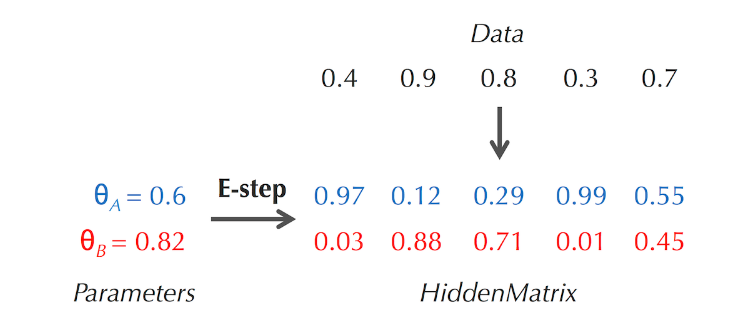
Trong phương pháp phân cụm k-means, nếu một điểm dữ liệu nằm giữa hai tâm cụm, thì mỗi tâm cụm nên có khoảng cùng mức trách nhiệm để thu hút nó vào cụm của chúng. Giống như việc ném đồng xu, trách nhiệm của tất cả các tâm cụm đối với một điểm dữ liệu nên cộng lại thành 1.

(*Data*, ?, *Parameters*) →*HiddenMatrix*

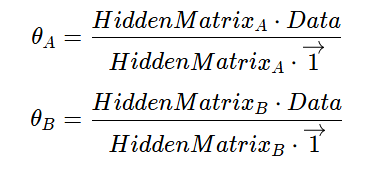


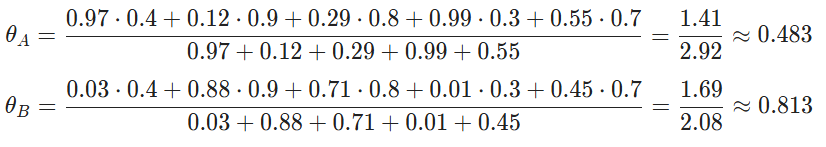
E-step: biến đổi sang ma trận ẩn (Data là số lần tung đồng xu)





M-step: tìm xác suất





**Soft *k*-means clustering algorithm** bắt đầu từ các trung tâm được chọn ngẫu nhiên và lặp lại hai bước sau:

* **Trung tâm đến cụm mềm (E-step):** Sau khi chọn trung tâm, hãy gán cho mỗi điểm dữ liệu một giá trị "trách nhiệm" cho mỗi cụm, trong đó các giá trị cao hơn tương ứng với tư cách thành viên cụm mạnh hơn.
* **Cụm mềm đến trung tâm (M-step):** Sau khi các điểm dữ liệu đã được gán cho các cụm mềm, hãy tính toán các trung tâm mới.

Công thức ở E-step cần biết trước xác xuất, ta cả tiến như sau:

