с логикой работы алгоритма. чем ближе точки к центроиду, тем точнее

1. После пересчета всех центров, мы распределяем пиксели по кластерам, сравнивая расстояние от каждого пикселя до центров. Пиксель помещается в кластер, к центру которого он расположен ближе, чем к остальным центрам, чем больше nstart тем точнее.
2. Все начинается сначала, до тех пор, пока пиксели остаются в одних и тех же кластерах. Часто такого может и не случится, так как при большом количестве данных центры будут перемещаться в малом радиусе, и пиксели по краям кластеров будут прыгать то в один, то в другой кластер. Для этого нужно определить максимальное число итераций.

там квадраты расстояний берутся и видать, там, где сумма этих квадратов перестает критически снижаться - там и локоть, ибо все, что дальше - это незначительные изменения, а значит не нужны нам

если мы больше возьмем, значит расстояние в сумме будет меньше, значит, что кластеры образуется вокруг более маленьких куч

суть в том, чтобы понять на какое количество кластеров надо разбить данные

Этот тип кластеризации дает отличное представление о структуре сети WiFi в городе. Это указывает на то, что географический регион, отмеченный кластером 1, показывает большой трафик WiFi. С другой стороны, меньшее количество соединений в кластере 6 может указывать на низкий трафик WiFi.  
  
Кластеризация K-Means сама по себе не говорит нам, почему трафик для конкретного кластера высок или низок. Например, когда кластер 6 имеет высокую плотность населения, но низкая скорость интернета приводит к меньшему количеству соединений.  
  
Однако этот алгоритм кластеризации обеспечивает отличную отправную точку для дальнейшего анализа и облегчает сбор дополнительной информации. Например, на примере данной карты можно строить гипотезы касательно отдельных географических кластеров.

Чтобы найти хорошее значение для K, мы можем построить сумму квадратов внутри групп для различных значений K. Эта метрика обычно уменьшается при добавлении большего количества групп, мы хотели бы найти точку, в которой уменьшение суммы внутри групп квадратов начинает медленно уменьшаться

Алгоритмы иерархической кластеризации

Среди алгоритмов иерархической кластеризации выделяются два основных типа: восходящие и нисходящие алгоритмы. Нисходящие алгоритмы работают по принципу «сверху-вниз»: в начале все объекты помещаются в один кластер, который затем разбивается на все более мелкие кластеры. Более распространены восходящие алгоритмы, которые в начале работы помещают каждый объект в отдельный кластер, а затем объединяют кластеры во все более крупные, пока все объекты выборки не будут содержаться в одном кластере. Таким образом строится система вложенных разбиений. Результаты таких алгоритмов обычно представляют в виде дерева – дендрограммы.

Возможно, вы уже знаете, что кластеризация бывает *методом k-средних и иерархическая*. В этом посте я фокусируюсь на втором методе, так как он более гибкий и допускает различные подходы: можно выбрать либо *агломеративный* (снизу вверх), либо *дивизионный* (сверху вниз) алгоритм кластеризации.

Агломеративная кластеризация начинается с n кластеров, где n — число наблюдений: предполагается, что каждое из них представляет собой отдельный кластер. Затем алгоритм пытается найти и сгруппировать наиболее схожие между собой точки данных — так начинается формирование кластеров.  
  
Дивизионная кластеризация выполняется противоположным образом — исходно предполагается, что все n точек данных, которые у нас есть, представляют собой один большой кластер, а далее наименее схожие из них разделяются на отдельные группы.  
  
Принимая решение о том, какой из этих способов выбрать, всегда имеет смысл попробовать все варианты, однако в целом *агломеративная кластеризация лучше подходит для выявления небольших кластеров и используется большинством компьютерных программ, а дивизионная кластеризация целесообразнее для выявления крупных кластеров*.

Рандомное количество точек которое стоит в nstrt