Машинное обучение (Machine Learning)

Методы обучения без учителя (Unsupervised Learning)

Уткин Л.В.

ПОЛИТЕХ
Санкт-Петербургский политехнический университет Петра Великого

Содержание

Презентация является компиляцией и заимствованием материалов из замечательных курсов и презентаций по машинному обучению:

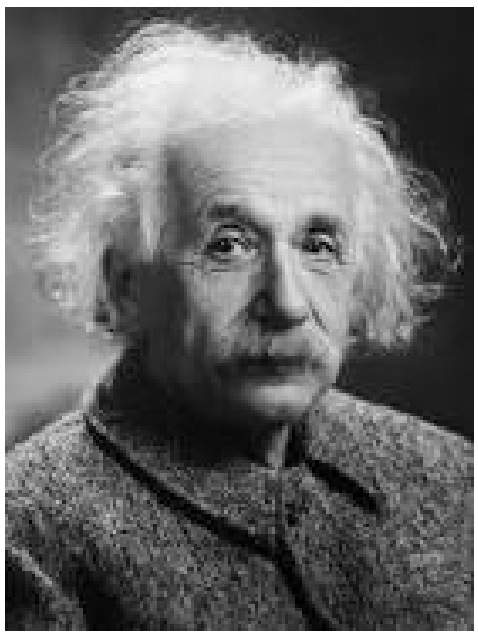
К.В. Воронцова, А.Г. Дьяконова, Н.Ю. Золотых, С.И. Николенко, Andrew Moore, Lior Rokach, Rong Jin, Luis F. Teixeira, Alexander Statnikov и других.

Кластеризация

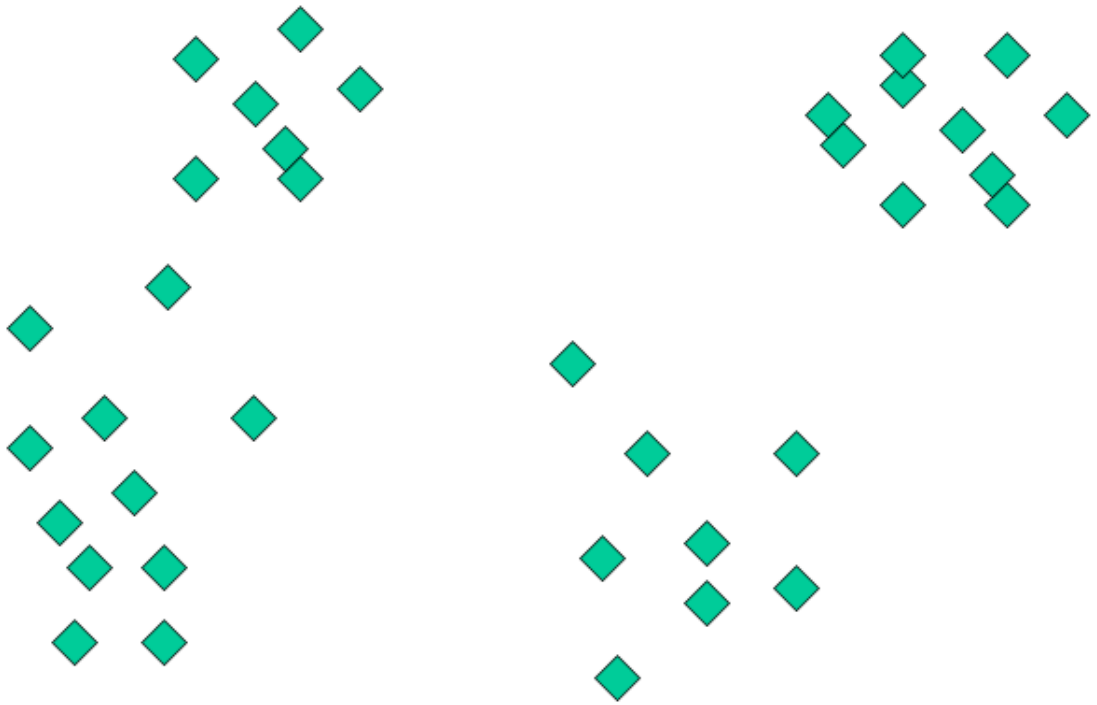
Что делать с информацией?

"Information is not knowledge."

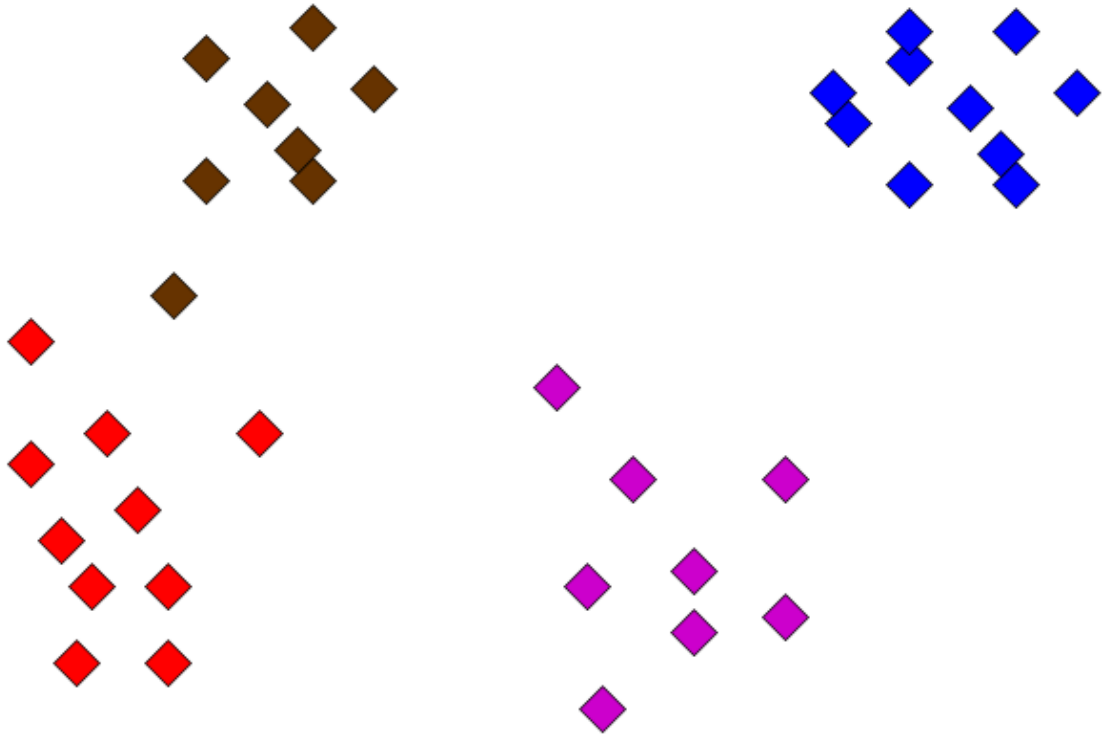
- Albert Einstein



Общее определение кластеризации



Общее определение кластеризации



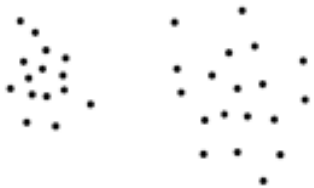
Общее определение кластеризации

Алгоритмы кластеризации разбивают заданное множество объектов на группы (кластеры), в одном кластере размещая ***близкие****, а в разных —* ***далекие*** *по своим характеристикам объекты.*

Цели кластеризации

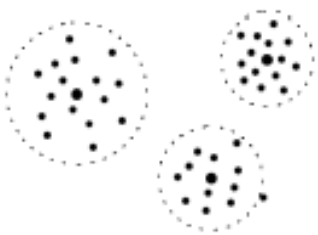
- Упростить дальнейшую обработку данных, разбить множество объектов на группы схожих объектов чтобы работать с каждой группой в отдельности (задачи классификации, регрессии, прогнозирования).
- Сократить объем хранимых данных, оставив по одному представителю от каждого кластера (задачи сжатия данных).
- Выделить нетипичные объекты, которые не подходят ни к одному из кластеров (задачи одноклассовой классификации).
- Построить иерархию множества объектов (задачи таксономии).

Типы кластерных структур



внутрикластерные расстояния, как правило, меньше межкластерных

ленточные кластеры



кластеры с центром

Типы кластерных структур

кластеры могут соединяться перемычками

кластеры могут накладываться на разреженный фон из редко расположенных объектов

кластеры могут перекрываться

Типы кластерных структур

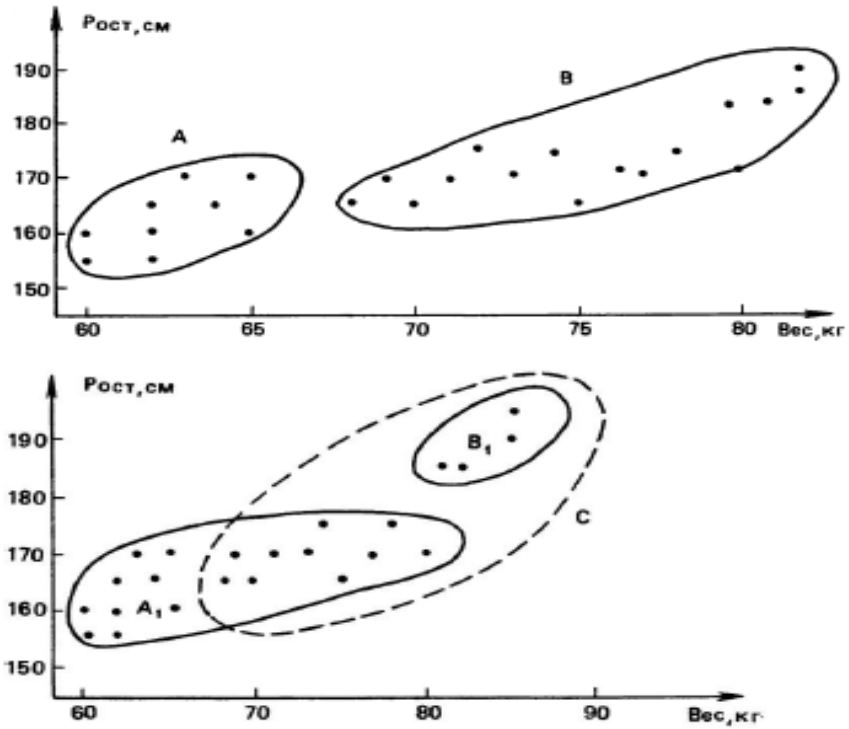
кластеры могут образовываться не по сходству, а по иным типам регулярностей

кластеры могут вообще отсутствовать

- Каждый метод кластеризации имеет свои ограничения и выделяет кластеры лишь некоторых типов.
- Понятие “тип кластерной структуры” зависит от метода и также не имеет формального определения.

Проблема чувствительности к выбору метрики

Результат зависит от нормировки признаков: A - женщины, B - мужчины



после перенормировки (сжали ось “вес” вдвое)

Формальная постановка задачи

Есть набор тестовых примеров $X = \{x_1, ..., x_n\}$ и функция расстояния между примерами.
Требуется разбить X на непересекающиеся подмножества (кластеры) так, чтобы каждое подмножество состояло из похожих объектов, а объекты разных подмножеств существенно различались.

Метод k средних

Метод k средних (k means) (начало)

Предположим, что множество объектов уже разбито некоторым образом на K групп (кластеров) и значение $C(i)$ равно номеру группы, которой принадлежит i-й объект.

Обозначим m_k центр тяжести системы точек, принадлежащих k-му кластеру:

$$m_k = \frac{1}{N_k} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i,\ k = 1, ..., K$$

Цель:

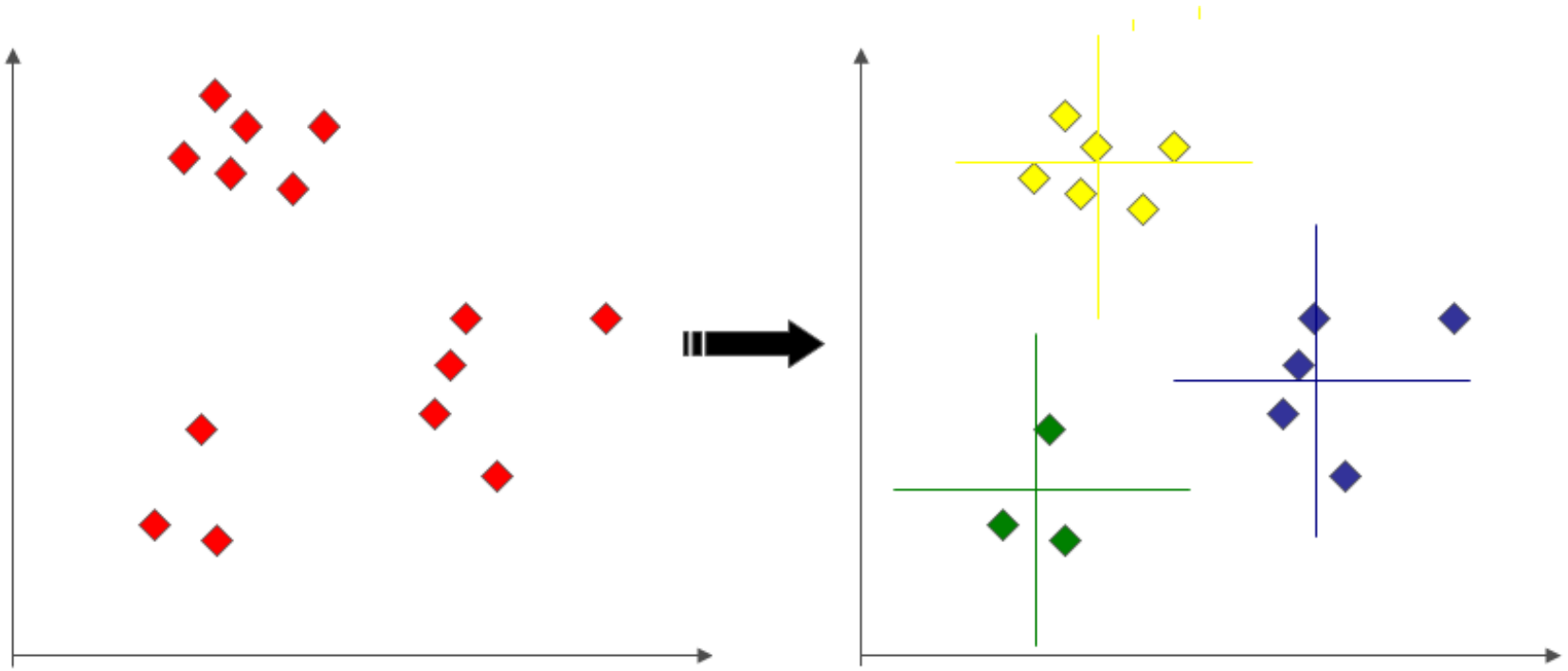
$$\min_{C, m_k} \sum_{i=1}^{n} \rho(\mathbf{x}_i, m_k)$$

Алгоритм k средних (формально)

1. Случайно разбиваем объекты на K кластеров.
2. Вычисляем центры тяжести m_k кластеров.
3. Вычисляем расстояния $\rho(\mathbf{x}_i, m_k)$ от точки $\mathbf{x}_i$ до всех m_k. Если $\rho(\mathbf{x}_i, m_j) = \min_{k=1,..,K} \rho(\mathbf{x}_i, m_k)$, то $\mathbf{x}_i \in C_j$.
4. Шаг 3 повторяем для всех объектов $\mathbf{x}_i$, $i = 1, ..., n$.
5. Если хотя бы один кластер C_k изменился, то переход на Шаг 2, иначе Завершение.

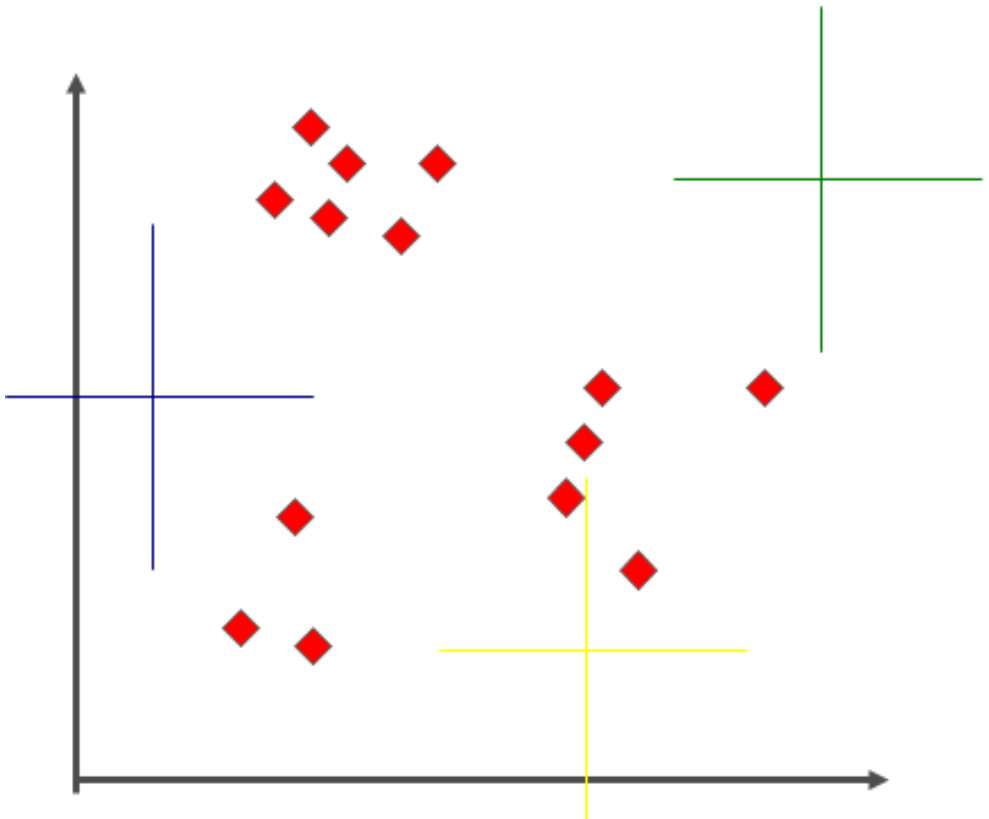
Пример алгоритма k средних (1)

3 класса:



Пример алгоритма k средних (2)

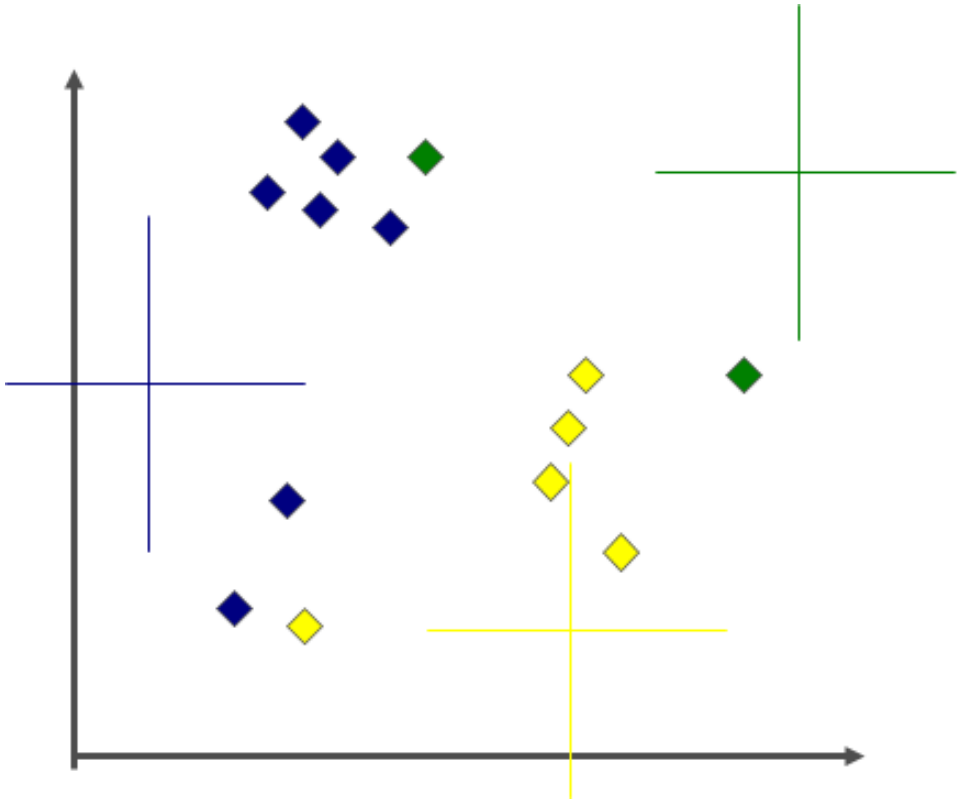
Шаг 1. Решаем, сколько кластеров (пусть 3)
Шаг 2. Случайно выбираем три центра кластеров



Пример алгоритма k средних (3)

Шаг 3. Выбираем метрику расстояния

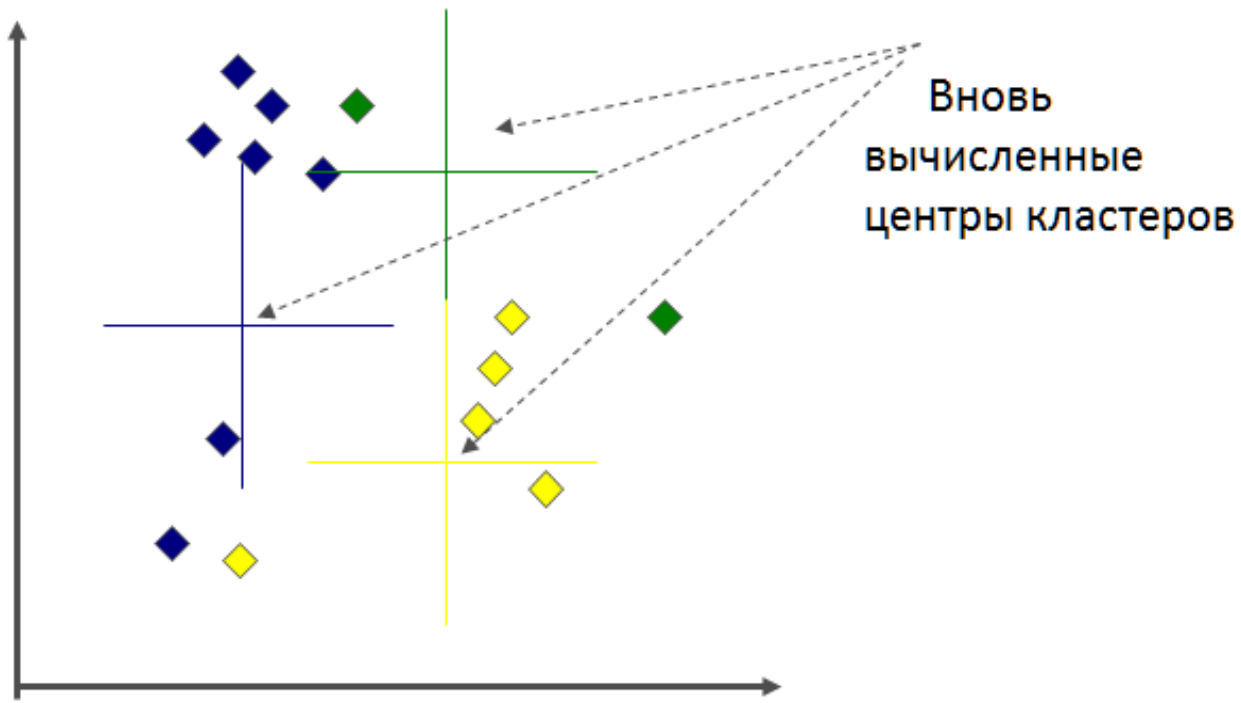
Шаг 4. Каждую точку относим к тому кластеру, центр которого ближе



Пример алгоритма k средних (4)

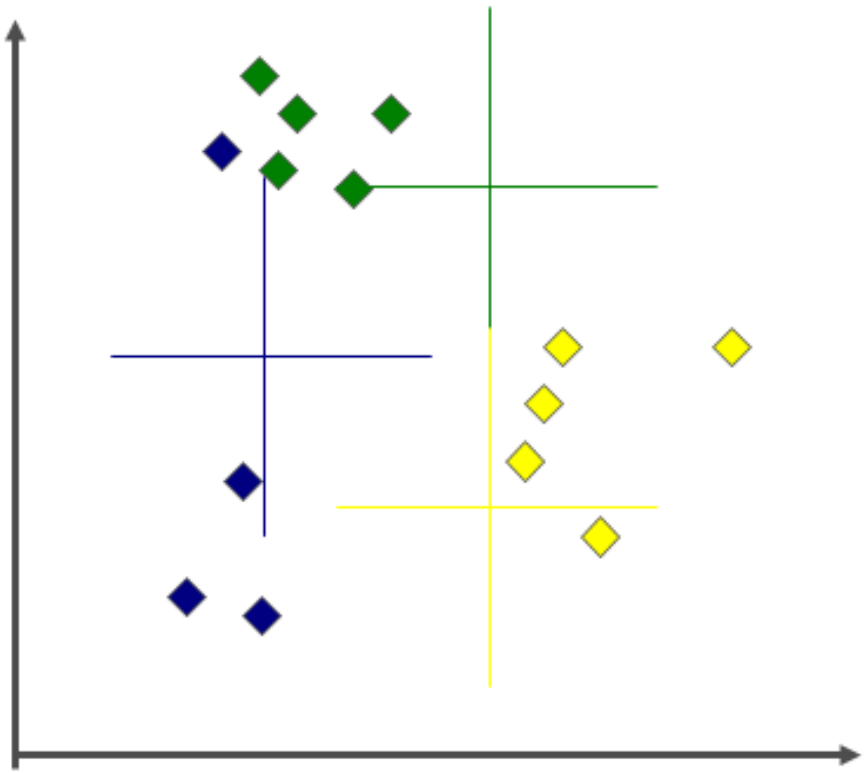
Шаг 5. Пересчитываем центры кластеров, используя новое разделение точек

Шаг 6. Повторяем Шаг 4 и Шаг 5, пока не кластеры не будут меняться



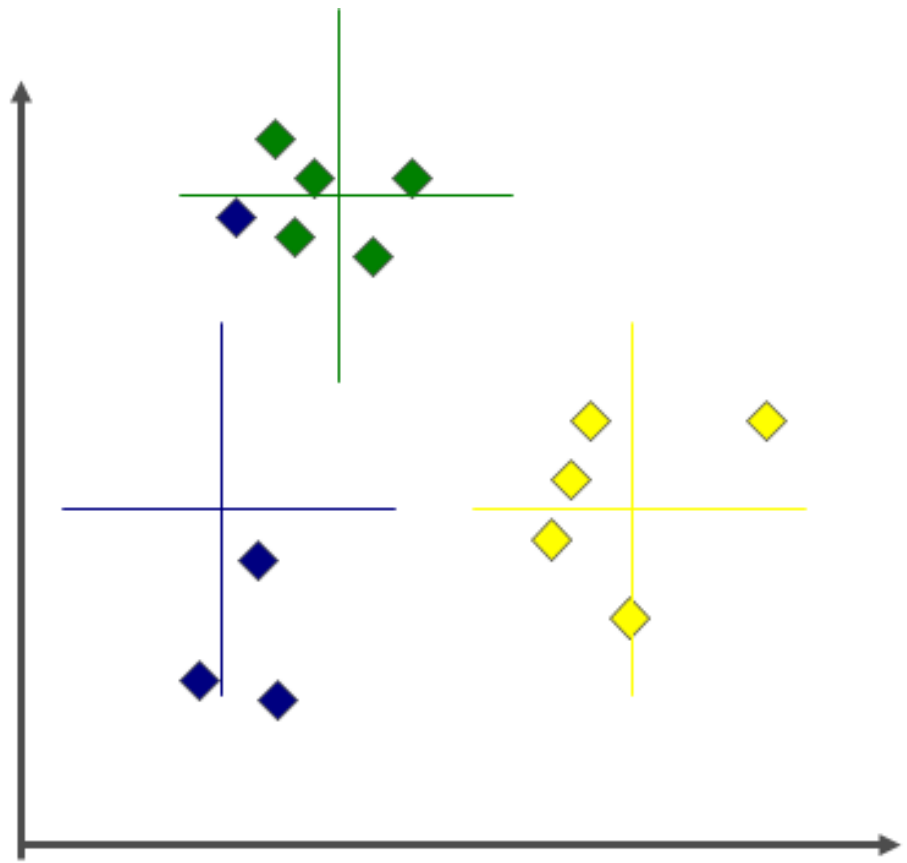
Пример алгоритма k средних (5)

Повторяем Шаг. 4 - перераспределяем точки



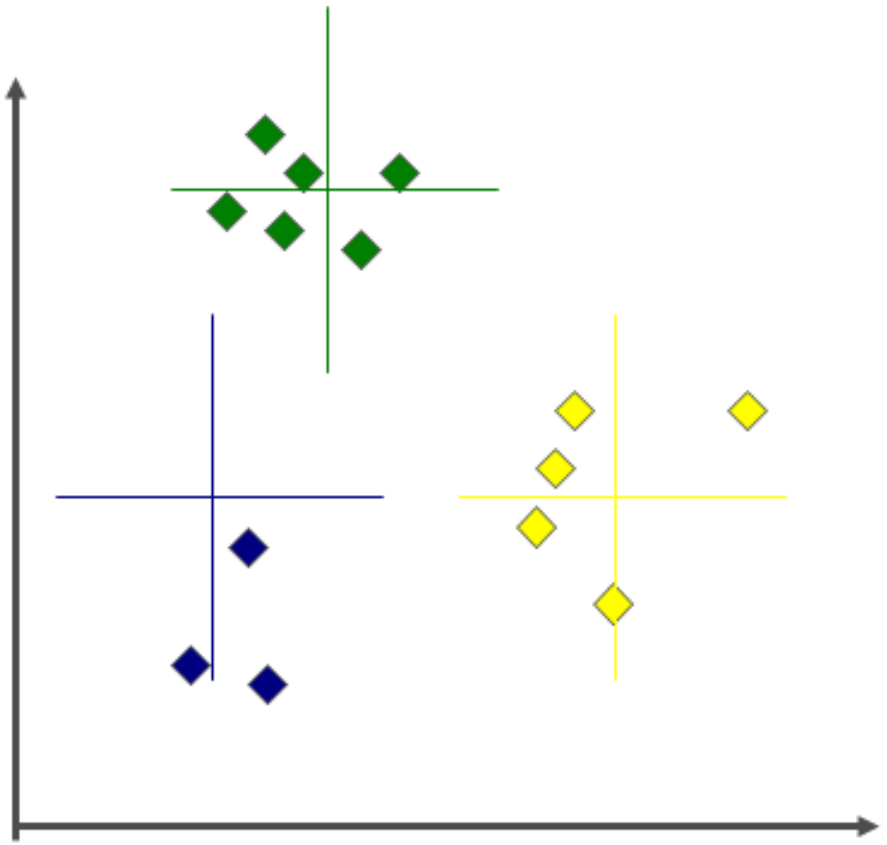
Пример алгоритма k средних (6)

Повторяем Шаг. 5 - пересчитываем центры кластеров

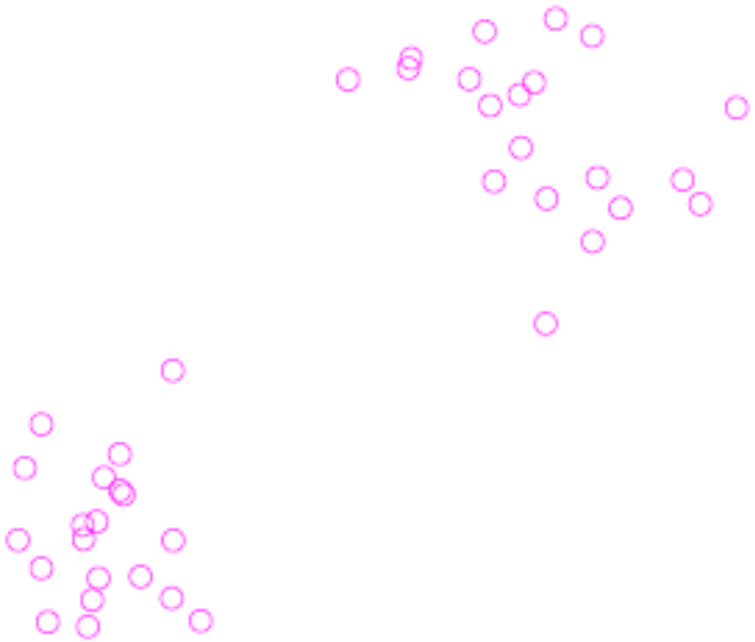


Пример алгоритма k средних (7)

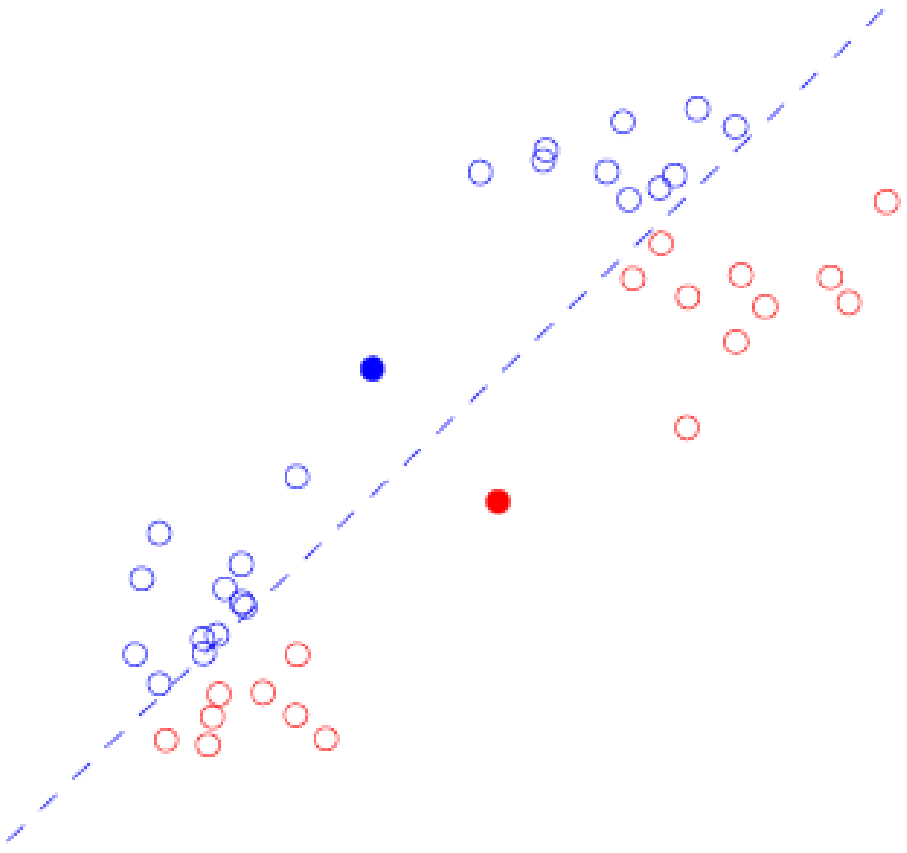
Повторяем Шаг. 4 - перераспределяем точки



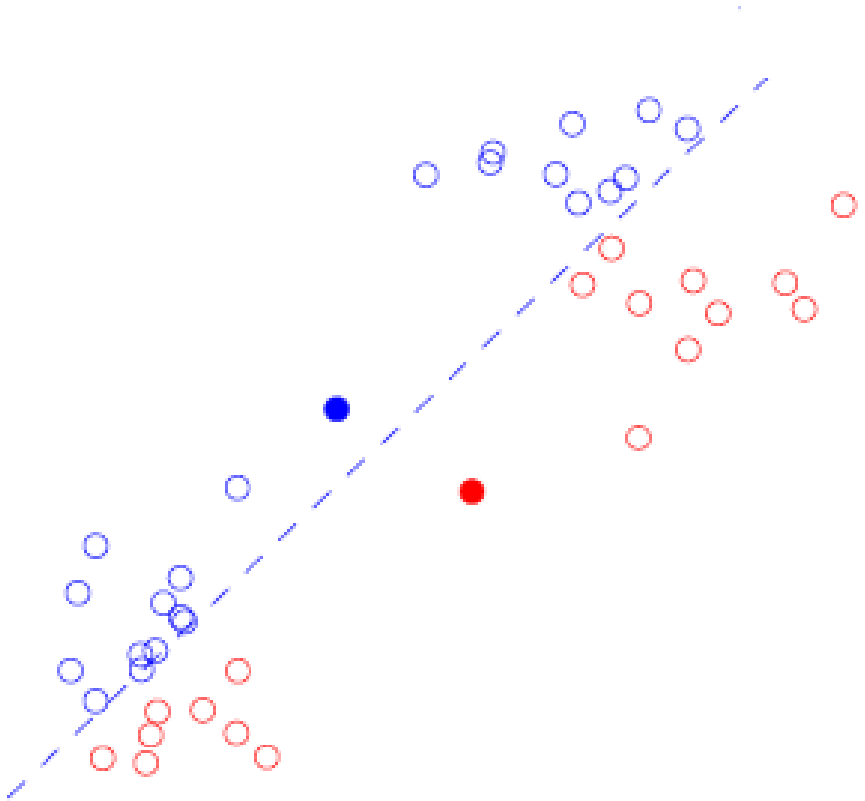
Еще один пример алгоритма (1)



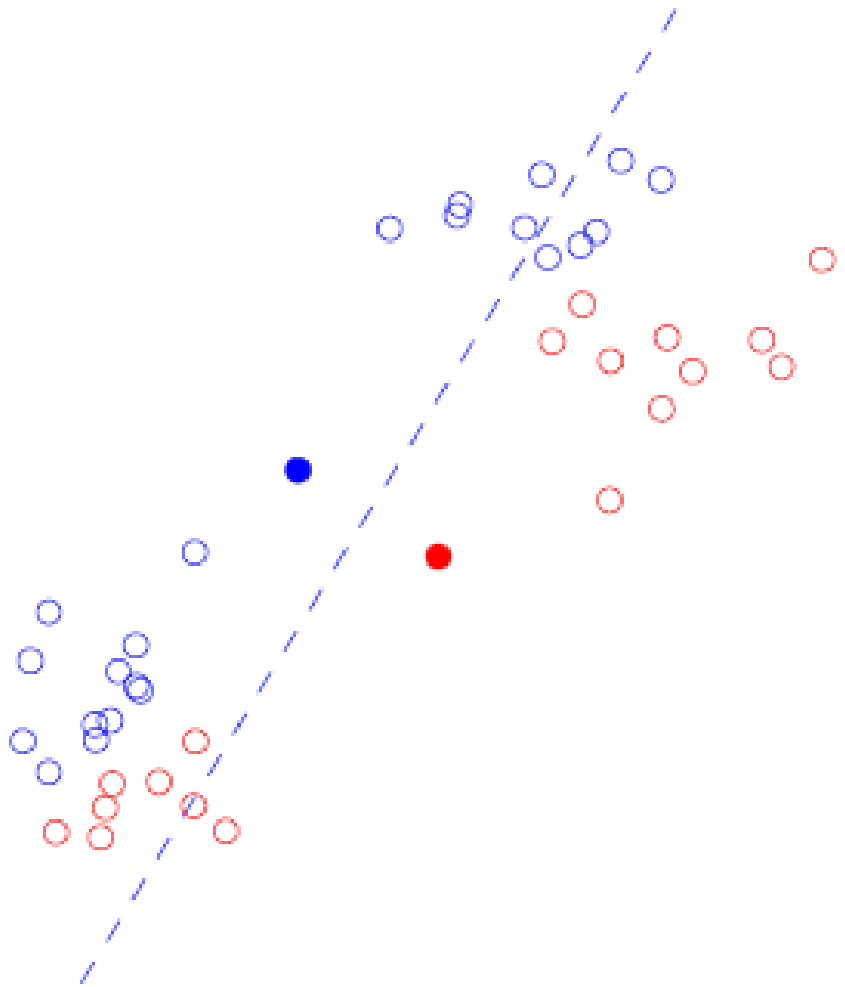
Еще один пример алгоритма (2)



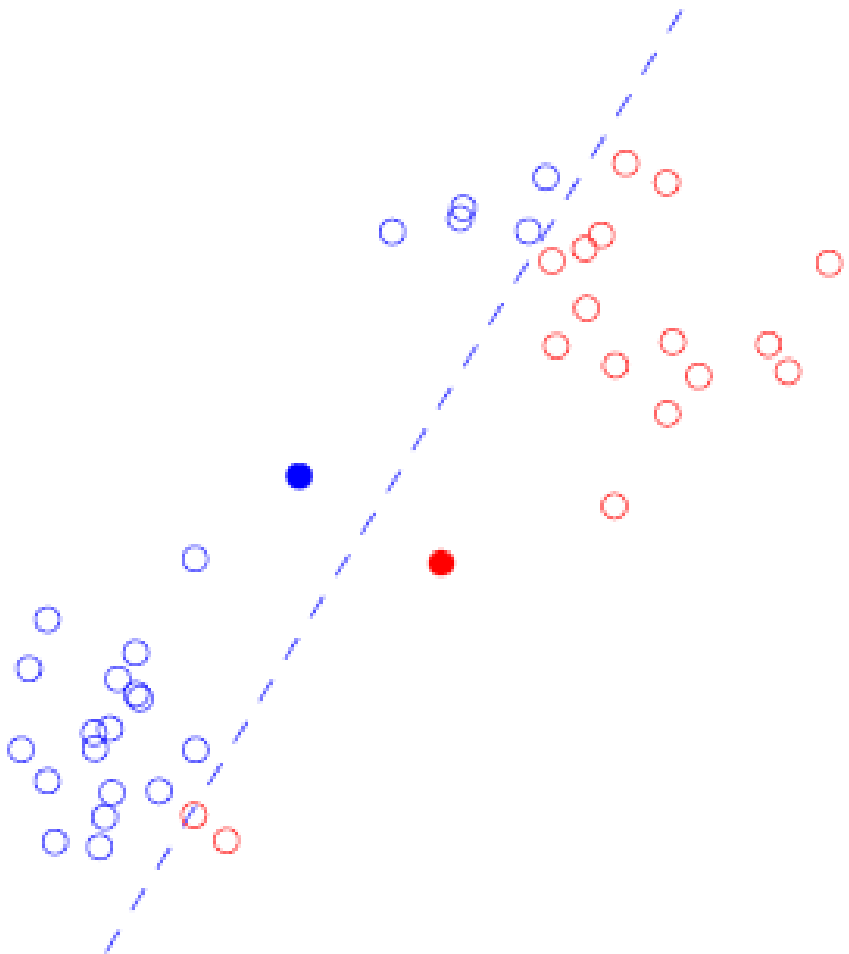
Еще один пример алгоритма (3)



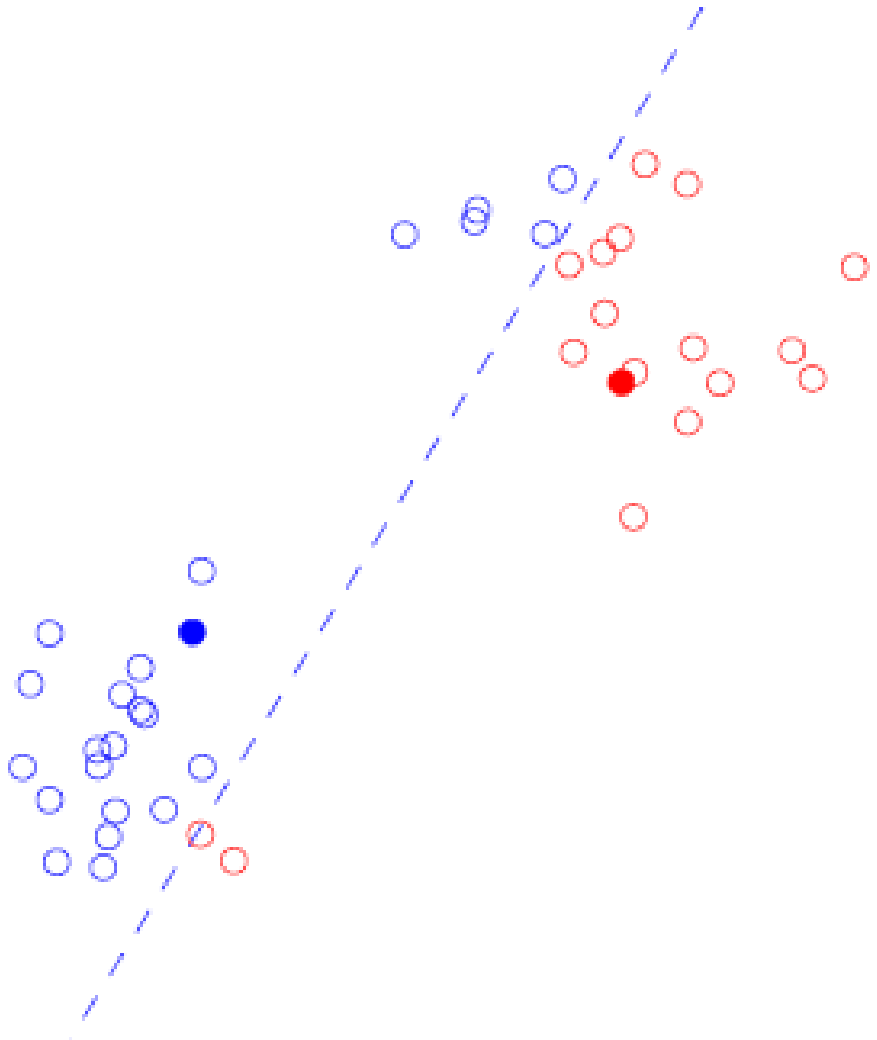
Еще один пример алгоритма (4)



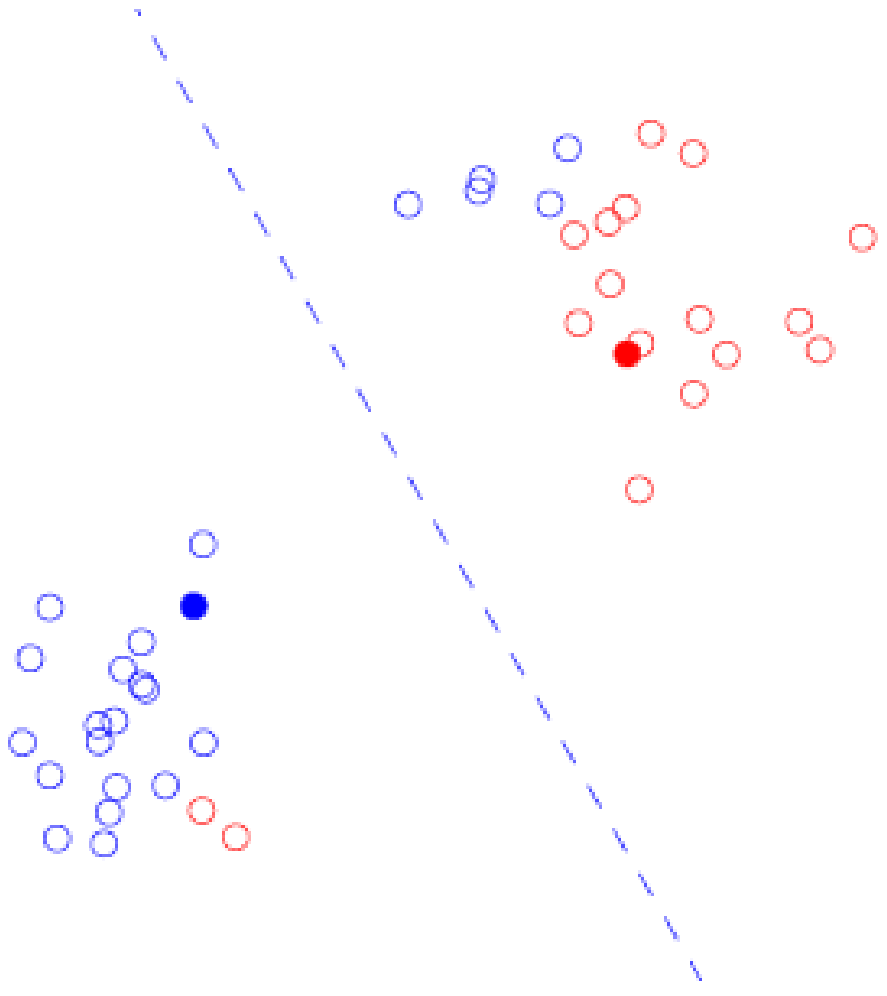
Еще один пример алгоритма (5)



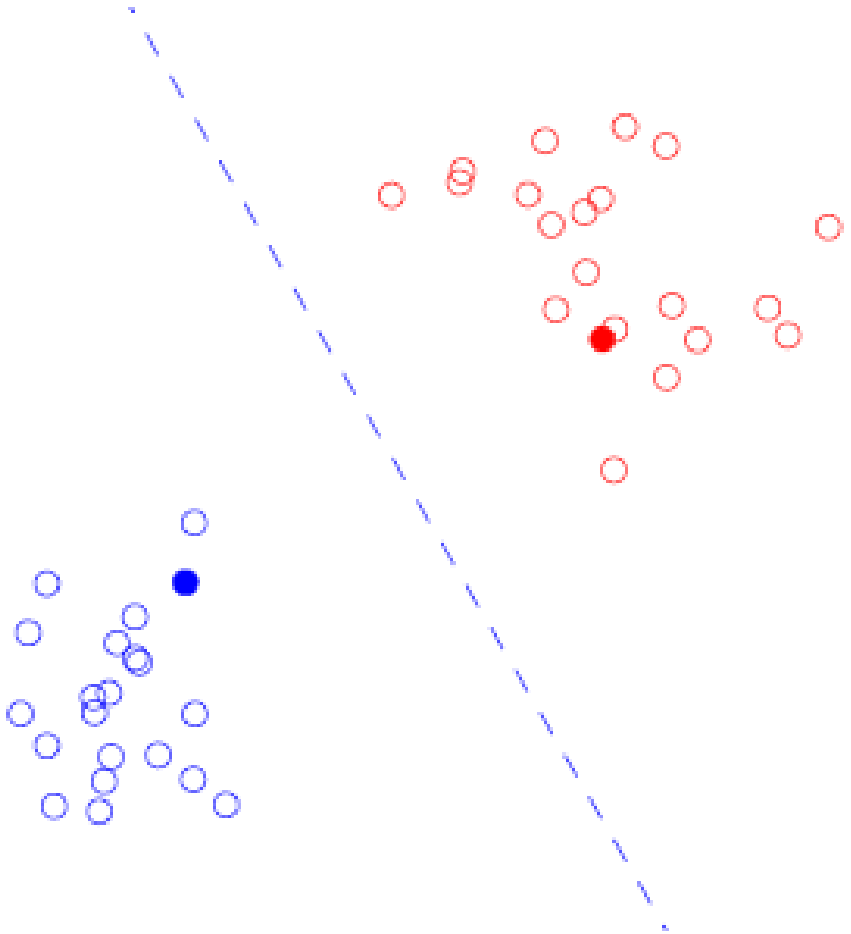
Пример алгоритма (6)



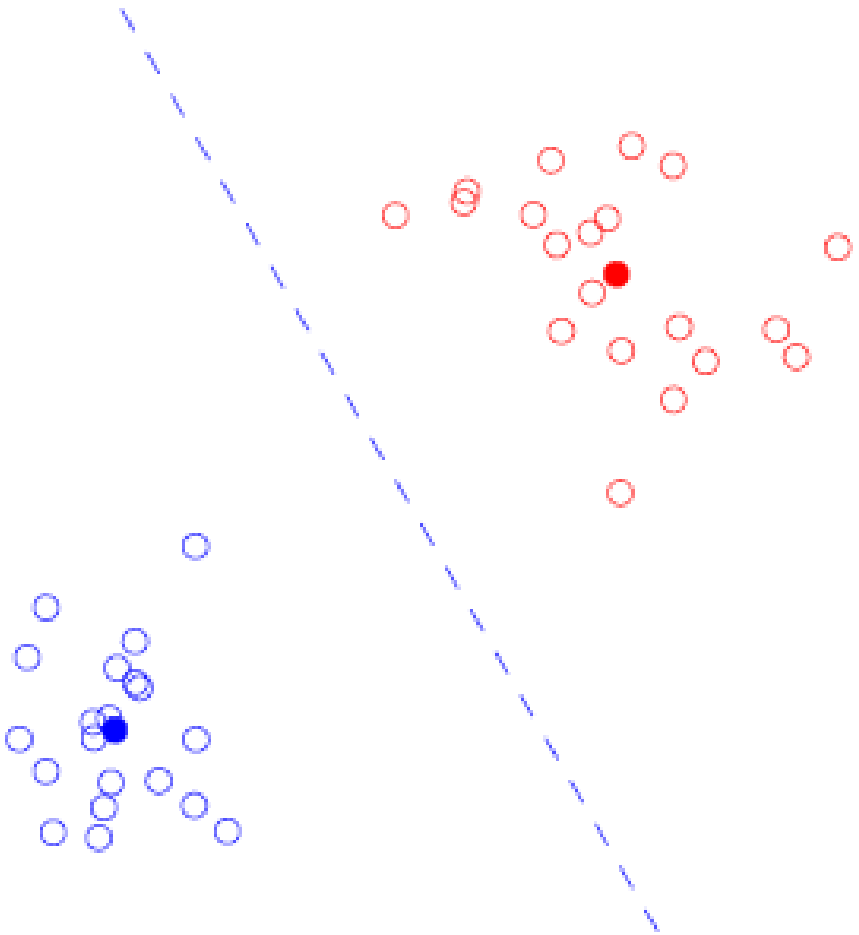
Еще один пример алгоритма (7)



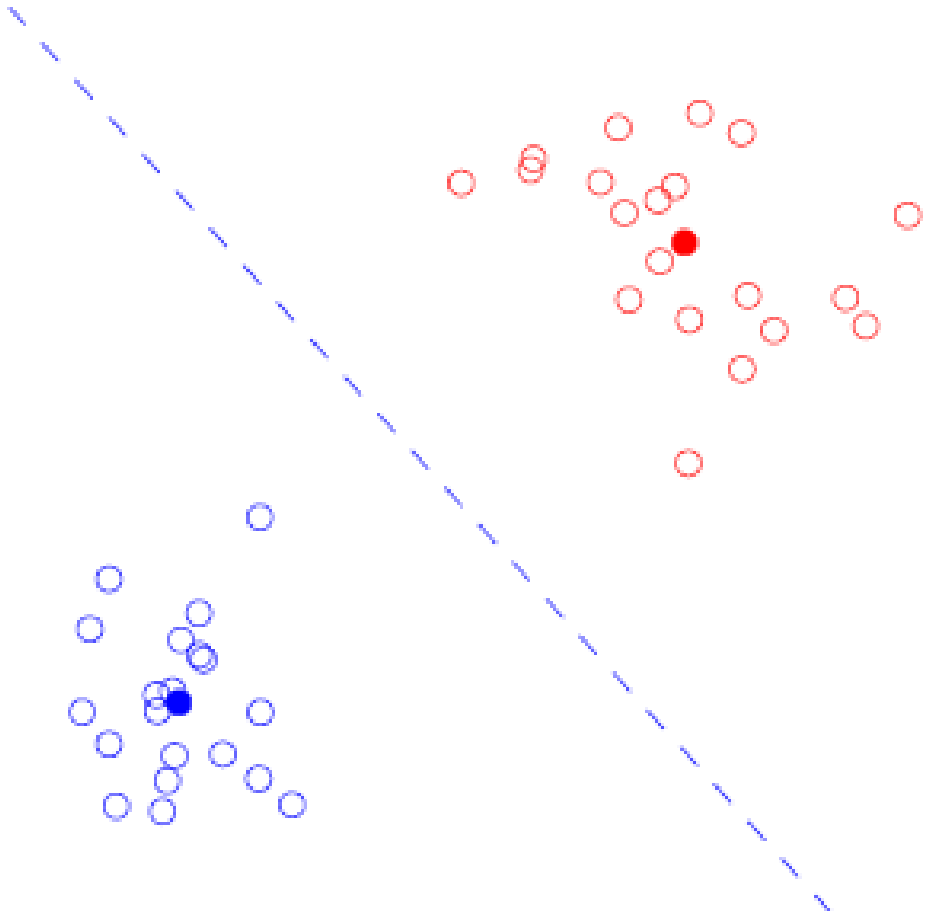
Еще один пример алгоритма (8)



Еще один пример алгоритма (9)



Еще один пример алгоритма (10)



Недостатки алгоритма k средних

1. Не гарантируется достижение глобального минимума суммарного квадратичного отклонения, а только одного из локальных минимумов.
2. Результат зависит от выбора исходных центров кластеров, их оптимальный выбор неизвестен.
3. Число кластеров надо знать заранее.

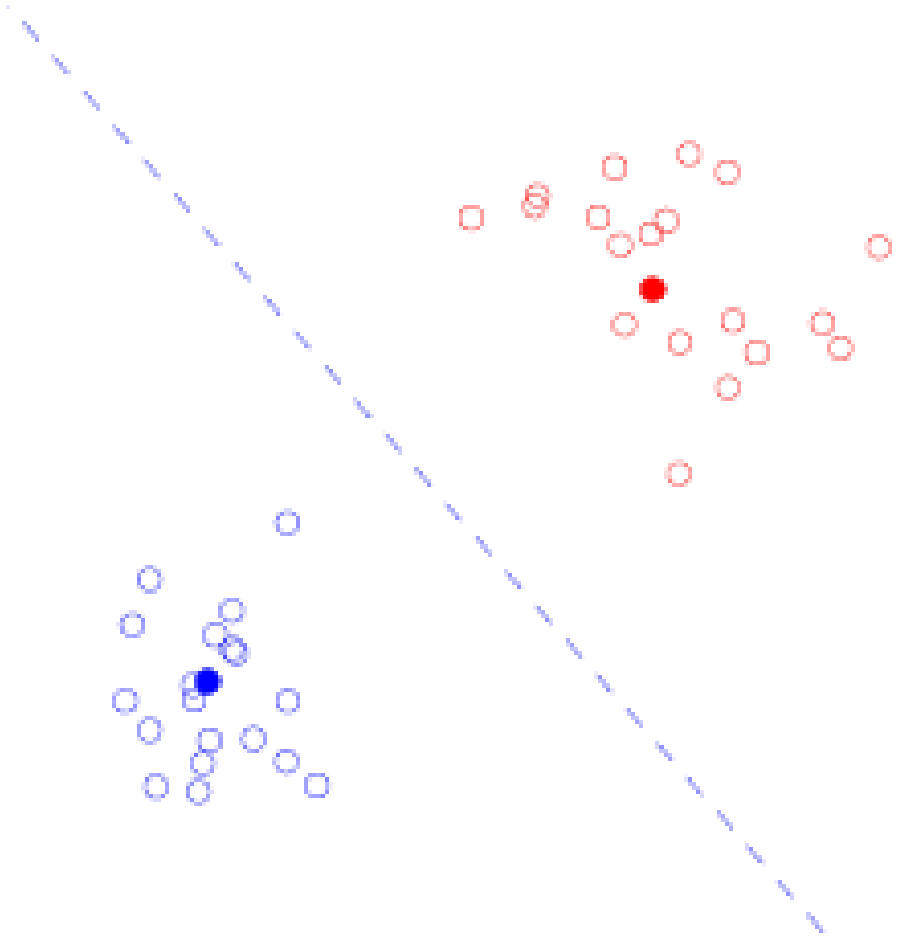
Устранение недостатков алгоритма k средних

1. Несколько случайных кластеризаций; выбор лучшей по функционалу качества.
2. Постепенное наращивание числа кластеров k.

Метод медоидов (K-medoids)

- Вместо центров тяжести в каждом кластере вычисляется медоид.
- **Медоид** - это **объект** множества данных или кластера, для которого среднее расстояние до других объектов минимальна.
- Алгоритм перестановки объектов из кластера в кластер аналогичен алгоритму k средних.
- Так как медоид - это объект, то достаточно знать матрицу расстояний между всеми объектами (преимущество).

Метод медоидов (пример)



Иерархическая кластеризация

Иерархическая кластеризация (таксономия)

- Основная идея: кластеры на более низком уровне получаются дроблением кластеров на более высоком уровне.
- В вершине классификации имеется один кластер, включающий все объекты.
- На низшем уровне имеется n кластеров, каждый из которых включает один объект.
- Такие иерархические структуры удобно представлять в виде корневых деревьев (**дендрограмм**).

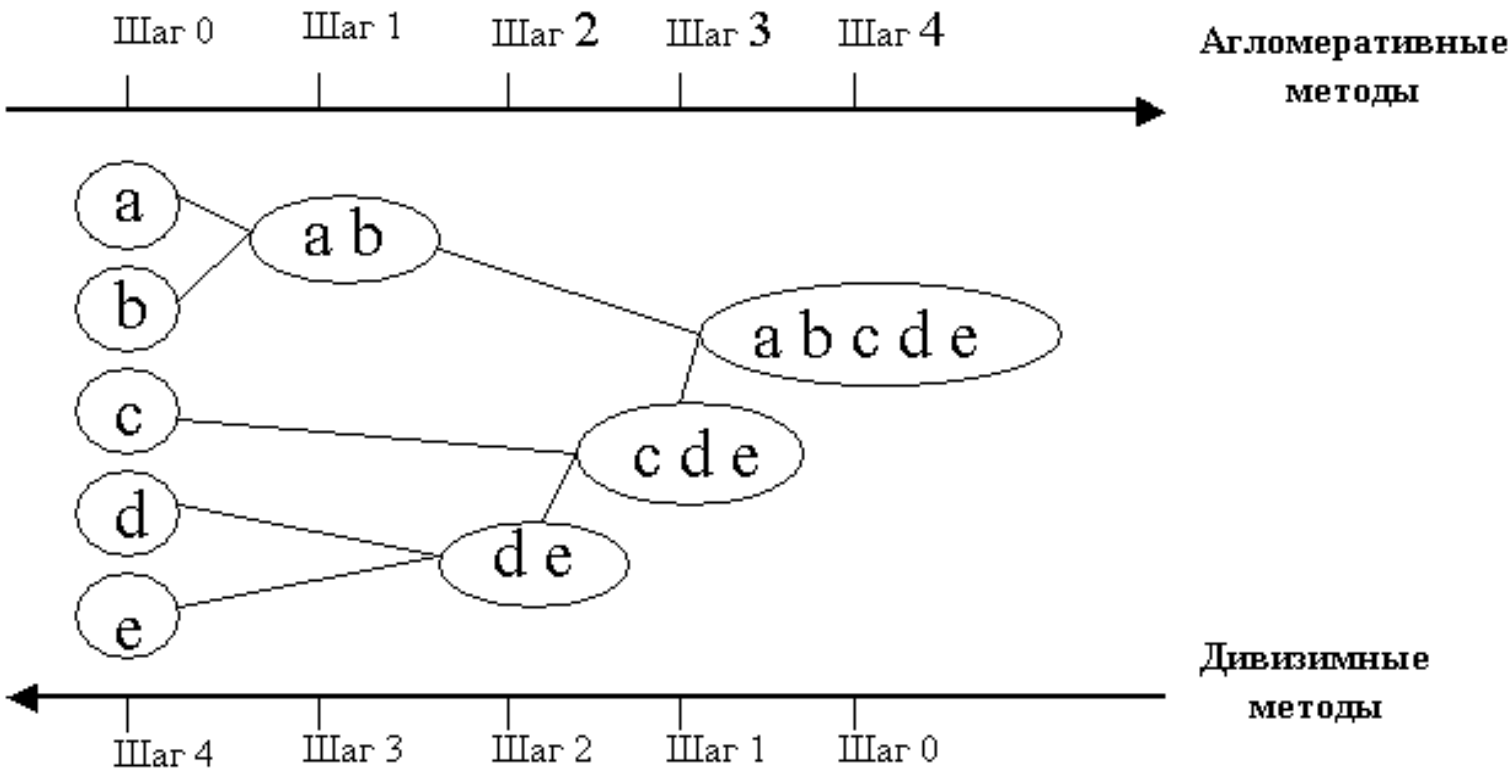
Преимущество алгоритма иерархической кластеризации

- Алгоритмам иерархической кластеризации не нужно на вход подавать количество кластеров.
- Имея дендрограмму, пользователь может сам обрезать ее на нужном уровне, получив некоторое количество кластеров.

Агломеративные методы или методы “снизу вверх”

- Последовательное объединение исходных элементов и соответствующие уменьшением числа кластеров.
- Сначала все объекты являются отдельными кластерами.
- На первом шаге наиболее похожие объекты объединяются в кластер.
- На последующих шагах объединение продолжается до тех пор, пока все объекты не будут составлять один кластер.

Агломеративные методы или методы “снизу вверх”



Агломеративные методы или методы “снизу вверх”

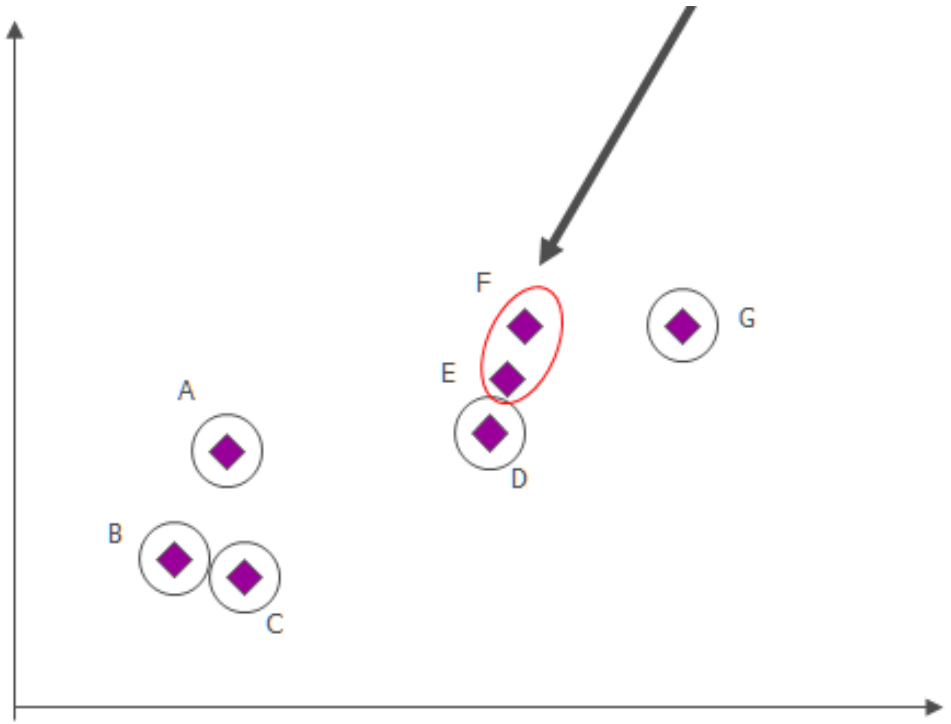
- Алгоритм CURE (Clustering Using REpresentatives)
  - Выполняет иерархическую кластеризацию с использованием набора определяющих точек для определения объекта в кластер
  - Назначение: кластеризация очень больших наборов числовых данных
  - Ограничения: эффективен для данных низкой размерности, работает только на числовых данных
  - Достоинства: выполняет кластеризацию на высоком уровне даже при наличии выбросов, выделяет кластеры сложной формы и различных размеров

Иллюстрация агломеративного метода (1)

Вначале каждый объект в своем кластере

Иллюстрация агломеративного метода (2)

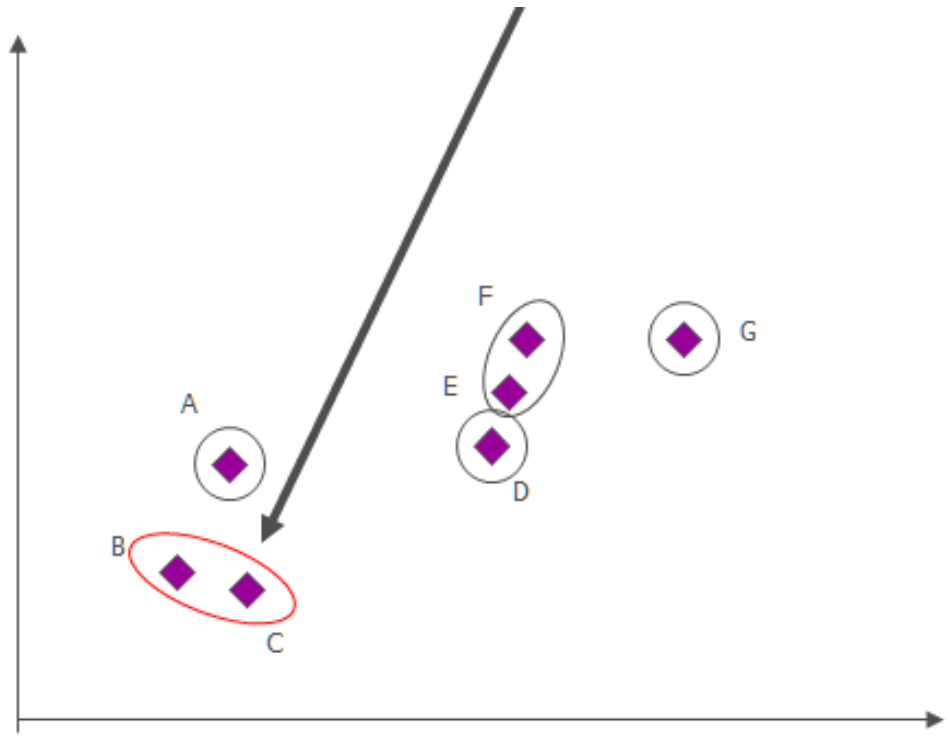
Объединяем два ближайших кластера



Дендограмма

Иллюстрация агломеративного метода (3)

Объединяем следующие два ближайших кластера



Дендограмма

Иллюстрация агломеративного метода (4)

Снова объединяем следующие два ближайших кластера

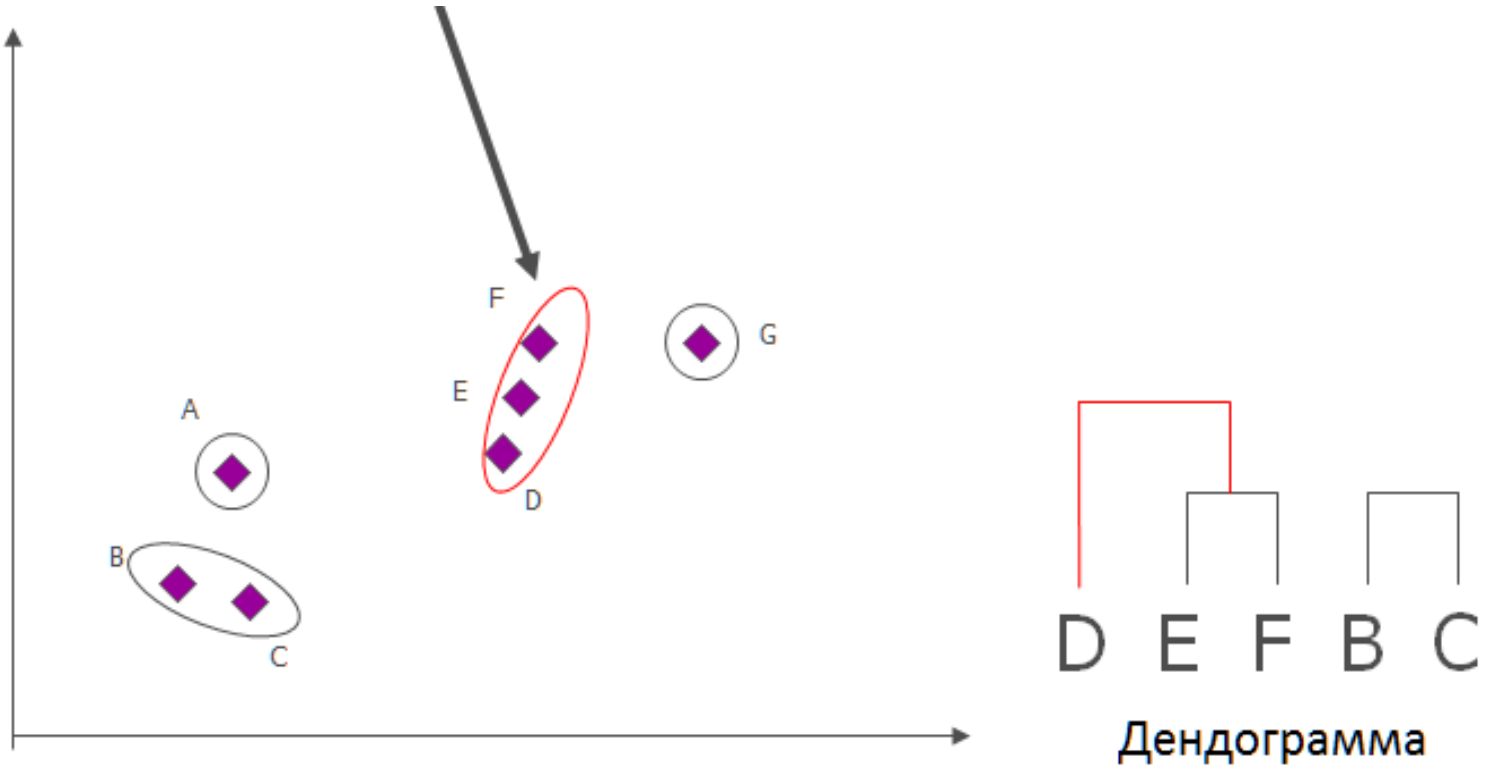
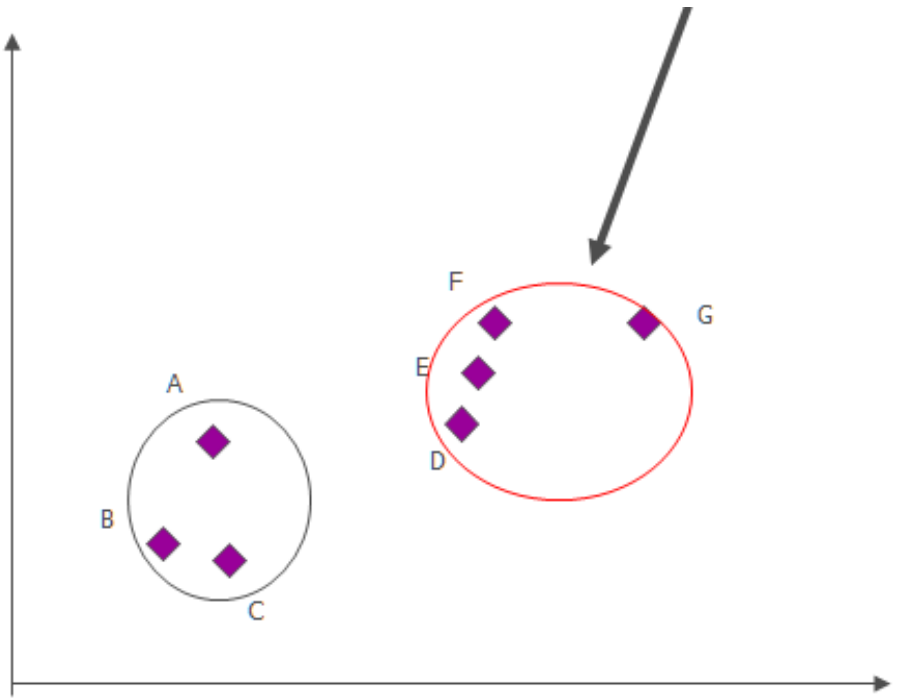


Иллюстрация агломеративного метода (5)

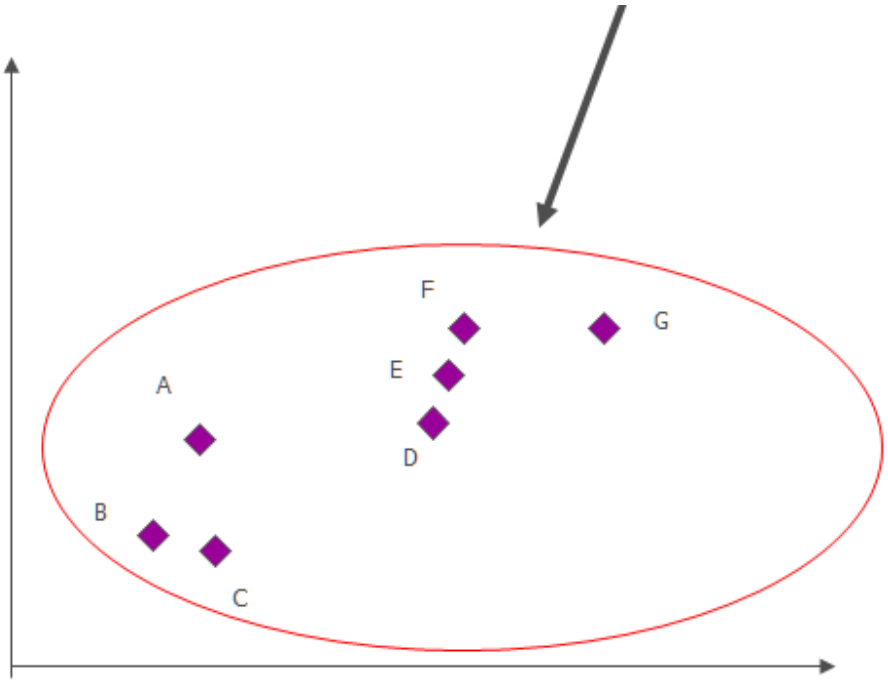
Снова объединяем следующие два ближайших кластера

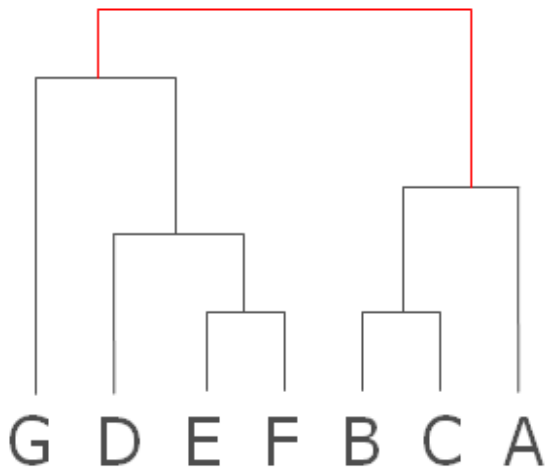


Дендограмма

Иллюстрация агломеративного метода (6)

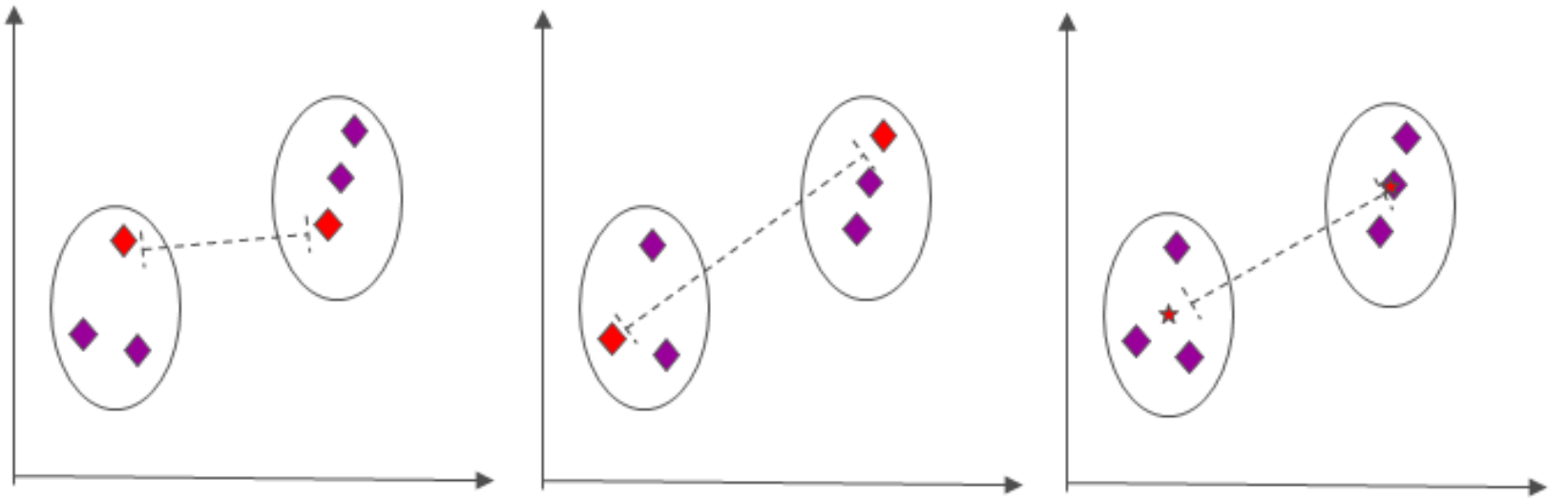
И завершаем объединение





Дендограмма

Расстояния между кластерами (1)



Расстояние ближнего соседа

Расстояние дальнего соседа

Групповое среднее расстояние

Расстояния между кластерами (2)

1. Расстояние ближайшего соседа:
   $D(C_i, C_j) = \min\{\rho(\mathbf{x}, \mathbf{z}) \mid \mathbf{x} \in C_i,\ \mathbf{z} \in C_j\}$
2. Расстояние дальнего соседа:
   $D(C_i, C_j) = \max\{\rho(\mathbf{x}, \mathbf{z}) \mid \mathbf{x} \in C_i,\ \mathbf{z} \in C_j\}$
3. Групповое среднее расстояние:

$$D(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{z} \in C_j} \rho(\mathbf{x}, \mathbf{z})$$

Разделяющие (дивизивные) методы “сверху вниз”

- Дерево строится в направлении от корня к листьям.

Один из подходов:

1. На первом шаге ко множеству объектов применим какой-либо алгоритм (центров тяжести, медоидов), разбивающий это множество на два кластера.
2. Затем разобъем каждый из полученных кластеров и т. д.

Разделяющие (дивизивные) методы “сверху вниз”

- Алгоритм BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies).
  - Предусмотрен двухэтапный процесс кластеризации.
  - Назначение: кластеризация очень больших наборов числовых данных.
  - Ограничения: работа с только числовыми данными.
  - Достоинства: двухступенчатая кластеризация, кластеризация больших объемов данных.

Разделяющие (дивизивные) методы “сверху вниз”

- Алгоритм MST (Algorithm based on Minimum Spanning Trees).
  - Назначение: кластеризация больших наборов произвольных данных.
  - Достоинства: выделяет кластеры произвольной формы, в т.ч. кластеры выпуклой и впуклой формы.

Еще один интересный дивизивный метод (два класса R и S)

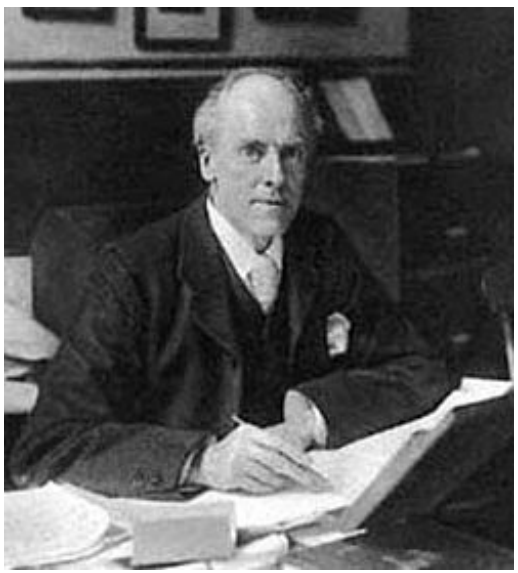
1. Найдем в R объект, для которого среднее расстояние до всех остальных объектов максимально. Обозначим его $\mathbf{x}_1 = \arg_i \max_{j=1,\ldots,n} \rho(\mathbf{x}_i, \mathbf{x}_j)$.
2. Удалим этот объект $\mathbf{x}_1$ из R и поместим в S.
3. Среди оставшихся объектов в R, найдем объект, для которого разность средних расстояний до всех остальных объектов между R и S максимальна. Обозначим его $\mathbf{x}_2$.
4. Удалим этот объект $\mathbf{x}_2$ из R и поместим в S.
5. Повторяем процедуру пока разность средних расстояний от объекта до объектов между R и S положительна.

PCA

Метод главных компонент (PCA)

Метод главных компонент *(principal component analysis, PCA) - один из основных способов уменьшить размерность данных, потеряв наименьшее количество информации.*

Карл Пирсон

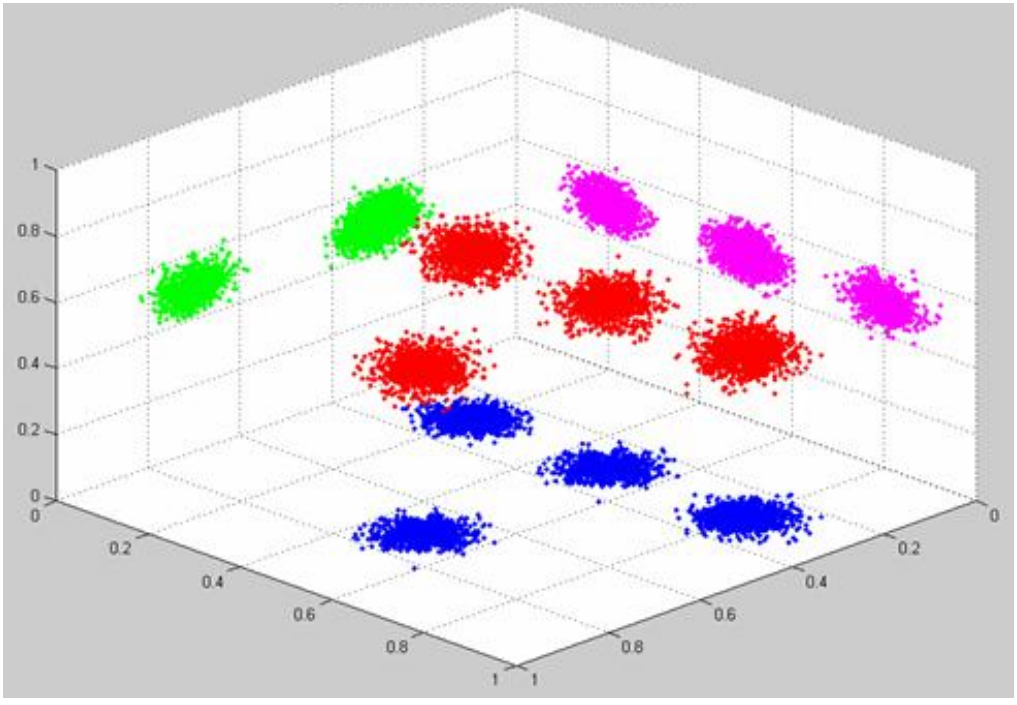


Метод главных компонент - предпосылки

Проекции данных на оси координат:

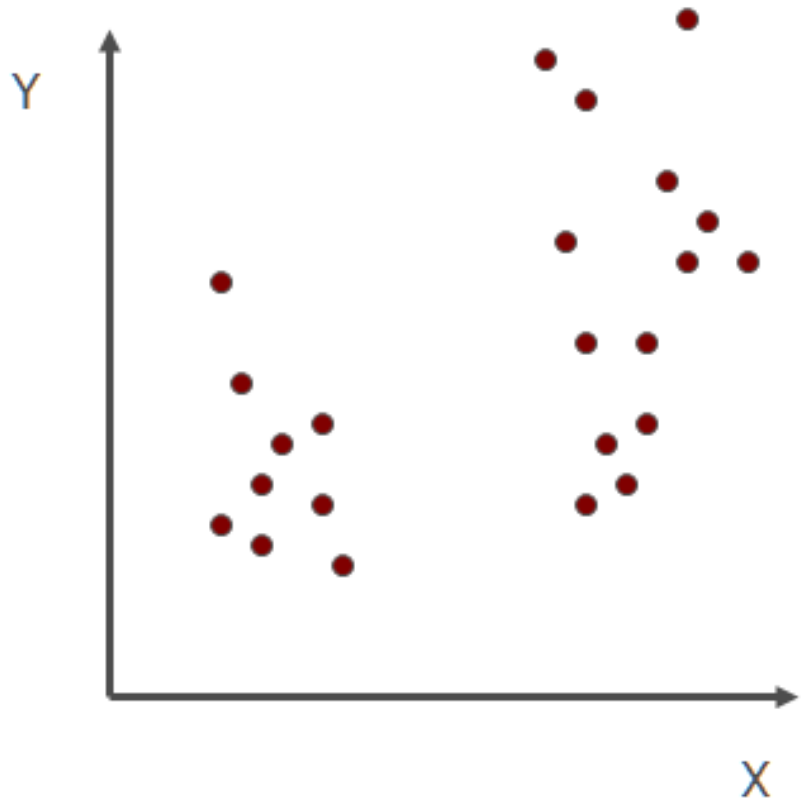
- некоторые проекции более информативны, чем другие
- в то время, как проекции различаются, объекты в исходном пространстве остаются неизменными
- обычно некоторые проекции избыточны
- на некоторых проекциях шум не сильно влияет
- меньше размерность задачи - проще ее решение

Метод главных компонент - предпосылки

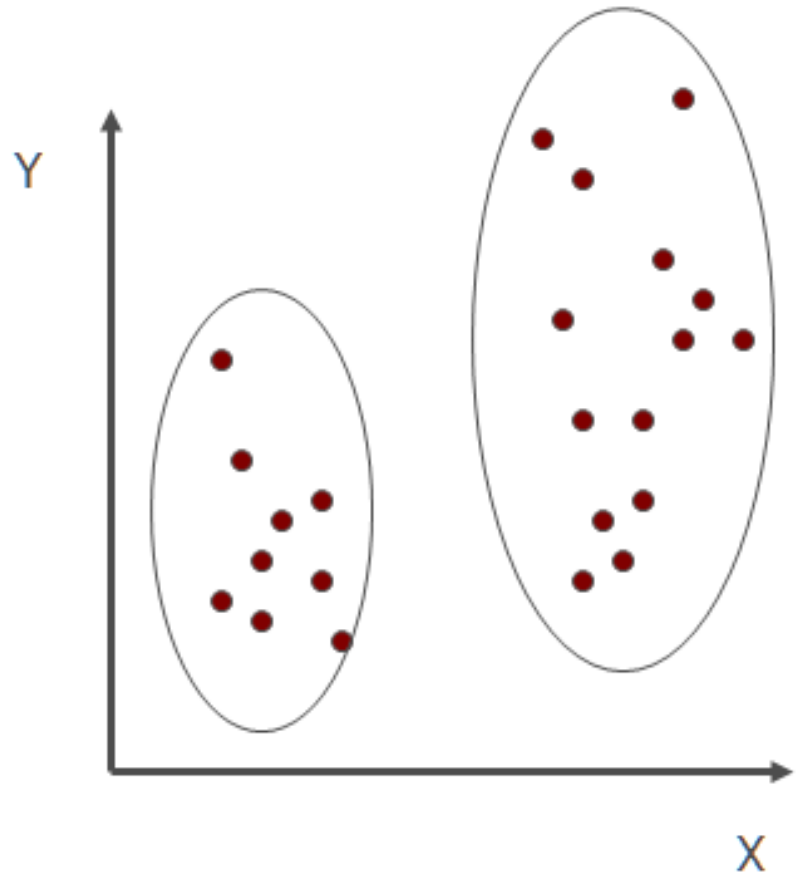


Метод главных компонент - предпосылки

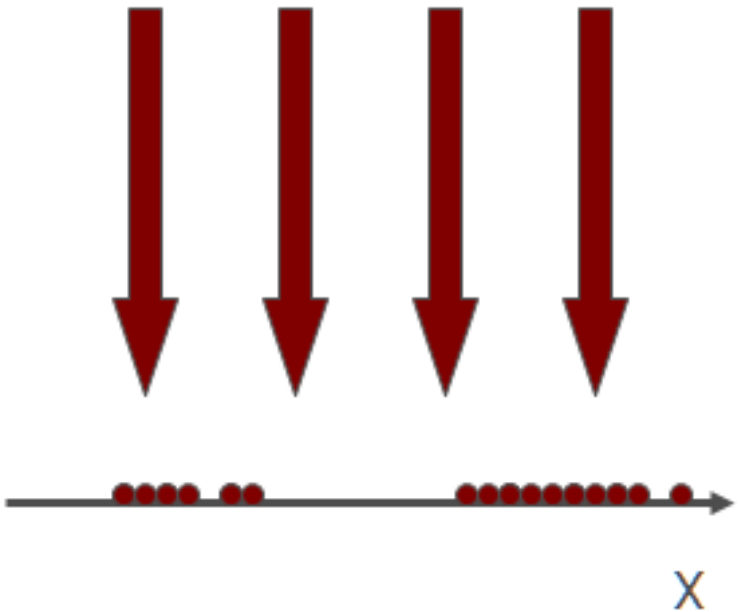
Метод главных компонент (пример)



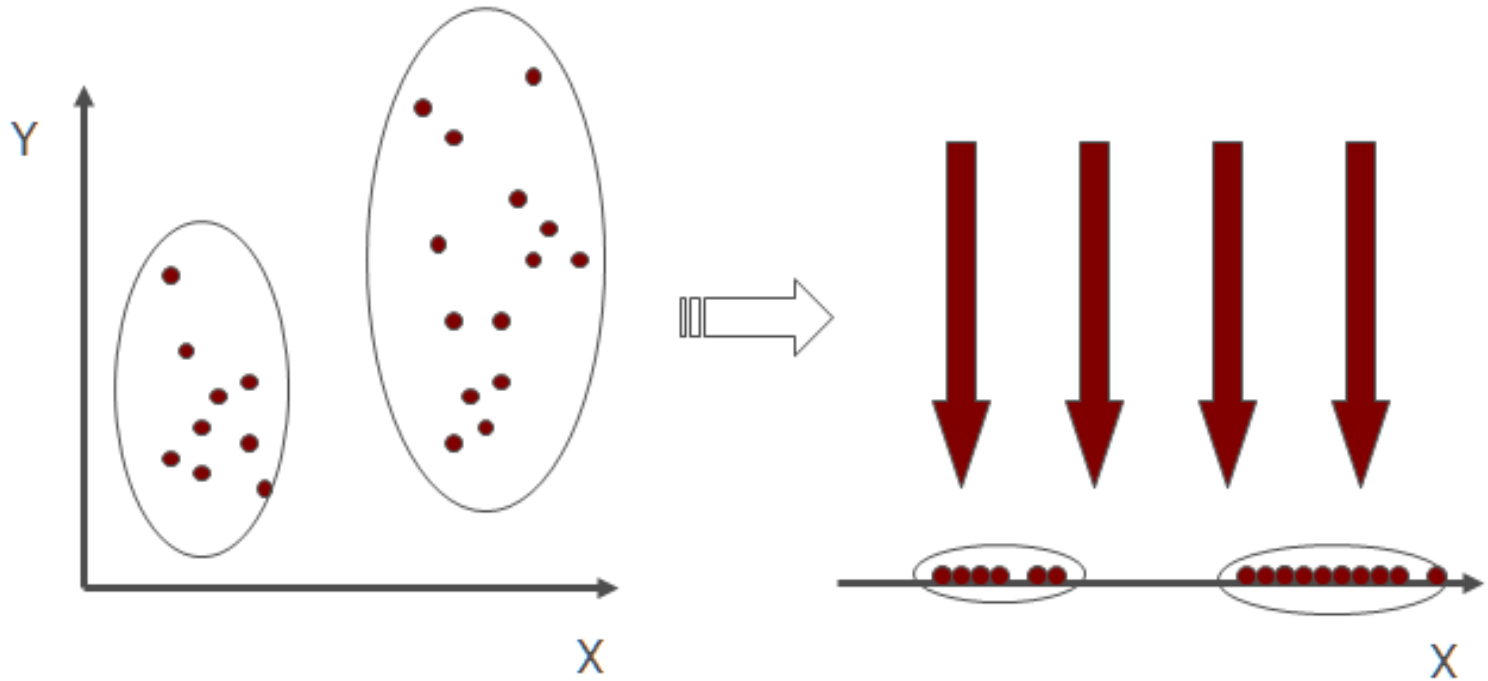
Метод главных компонент (пример)



Метод главных компонент (пример)

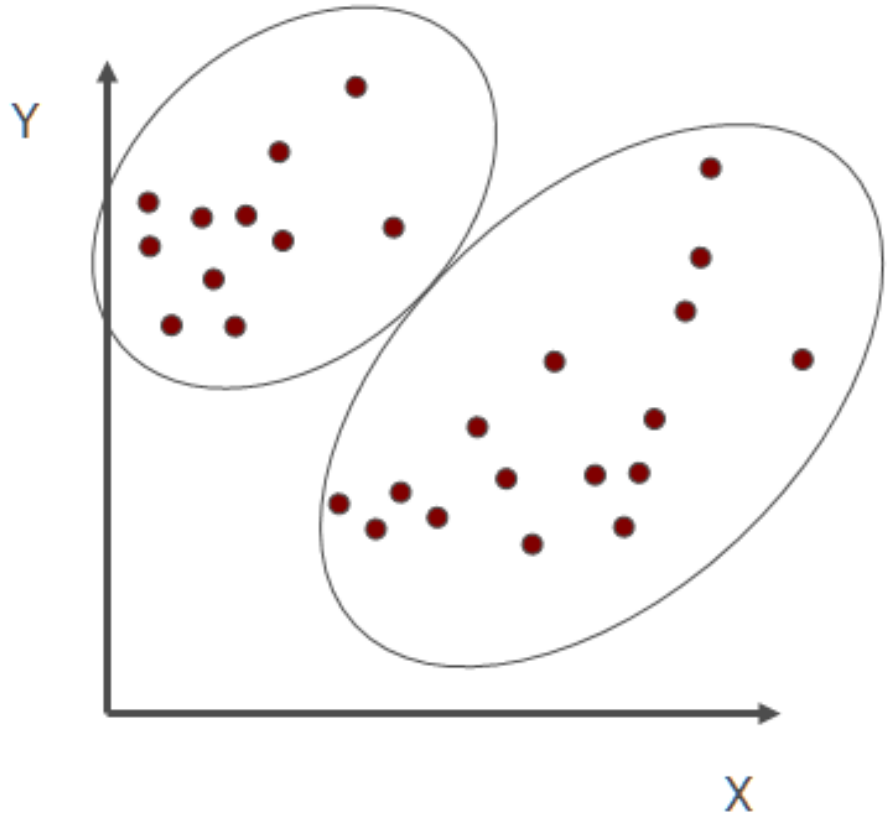


Метод главных компонент (пример)



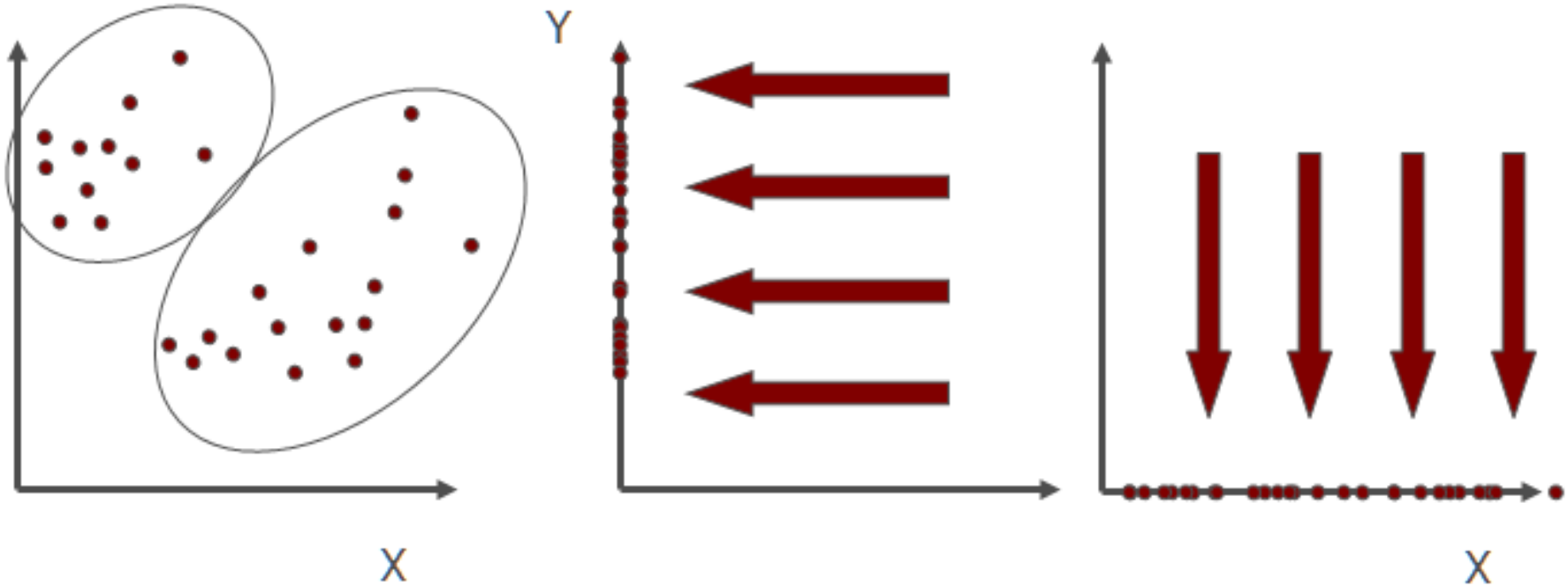
Y является избыточным, для кластеризации достаточно X

Метод главных компонент (другой пример)



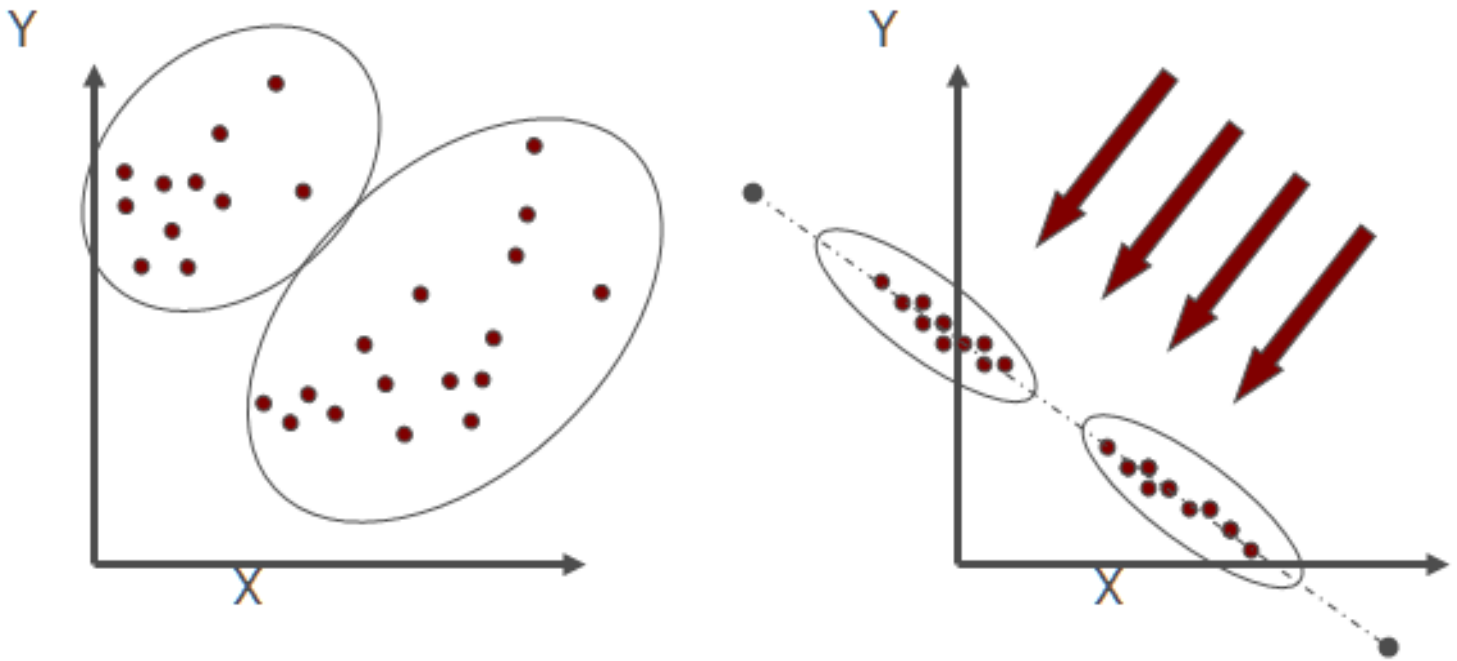
Здесь не очевидно, какая проекция лучше!

Метод главных компонент (другой пример)



Нет проекции, которая бы позволила разделить данные

Метод главных компонент (другой пример)



Однако, если проецировать данные на линейную комбинацию двух осей, то это может сработать лучше

Идея метода главных компонент (более формально)

- Пусть дано исходное множество векторов $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^m$ линейного пространства L^n.
- Применение метода главных компонент позволяет перейти к базису пространства L^K $(K \leq n)$, такому что:
- *Первая компонента* (первый вектор базиса) соответствует направлению, вдоль которого дисперсия векторов исходного набора максимальна.
- Направление *второй компоненты* (второго вектора базиса) выбрано таким образом, чтобы дисперсия исходных векторов вдоль него была максимальной при условии ортогональности первому вектору базиса.
- Аналогично определяются остальные векторы базиса.

Идея метода главных компонент (более формально)

- В результате, направления векторов базиса выбраны так, чтобы **максимизировать дисперсию исходного набора вдоль первых компонент, называемых главными компонентами** (или главными осями).
- Получается, что основная изменчивость векторов исходного набора векторов представлена несколькими первыми компонентами, и появляется возможность, отбросив оставшиеся (менее существенные) компоненты, перейти к пространству меньшей размерности.

Алгоритм метода главных компонент

- Начнем с $K = 1$: надо найти вектор $\mathbf{u}_1$ такой, что $\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 = 1$, для которого максимизируется дисперсия в проекции

$$\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{u}_1^{\mathrm{T}}\mathbf{x}_i - \mathbf{u}_1^{\mathrm{T}}\overline{\mathbf{x}}\right)^2 = \mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1,$$

$$\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i, \qquad \mathbf{S} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i - \overline{\mathbf{x}}\right)\left(\mathbf{x}_i - \overline{\mathbf{x}}\right)^{\mathrm{T}}$$

- То есть задача - максимизировать $\mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1$ с ограничением $\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 = 1$.

Алгоритм метода главных компонент

- Максимизировать $\mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1$ при ограничении $\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 = 1$
- Используем метод множителей Лагранжа:

$$\mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1 + \lambda_1\left(1 - \mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1\right)$$

- Максимум достигается, когда

$$\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1, \quad \text{т.е.} \quad \lambda_1 = \mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1$$

- В итоге: $\mathbf{u}_1$ собственный вектор $\mathbf{S}$ с максимальным собственным числом λ_1
- Далее то же самое: $\mathbf{u}_2$ - второй собственный вектор и т.д.

Алгоритм метода главных компонент

- С другой стороны, минимизируем ошибку
- Введем ортонормированный базис $\{\mathbf{u}_j\}$, $\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_j = \delta_{ij}$
- Векторы раскладываются тогда как

$$\mathbf{x}_k = \sum_{i=1}^{n} \left(\mathbf{x}_k^{\mathrm{T}}\mathbf{u}_i\right)\mathbf{u}_i$$

аппроксимируем вектор

$$\widetilde{\mathbf{x}}_k = \sum_{i=1}^{K} z_{ki}\mathbf{u}_i + \sum_{i=K+1}^{m} b_i\mathbf{u}_i$$

где b_i для всех одинаковые (поворот и смещение подпространства размерности K)

Алгоритм метода главных компонент

- Аппроксимируем вектор

$$\widetilde{\mathbf{x}}_k = \sum_{i=1}^{K} z_{ki}\mathbf{u}_i + \sum_{i=K+1}^{m} b_i\mathbf{u}_i$$

оптимизируем в итоге

$$J = \frac{1}{n}\sum_{k=1}^{n} \|\mathbf{x}_k - \widetilde{\mathbf{x}}_k\|$$

- Взяв производные по z_{kj}, получим $z_{kj} = \mathbf{x}_k^{\mathrm{T}}\mathbf{u}_j$
- По b_j: $b_j = \overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_j$

Алгоритм метода главных компонент

- Взяв производные по z_{kj}, получим $z_{kj} = \mathbf{x}_k^{\mathrm{T}}\mathbf{u}_j$, по b_j: $b_j = \overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_j$
- В итоге получаем

$$J = \frac{1}{n}\sum_{k=1}^{n}\sum_{i=K+1}^{m}\left(\mathbf{x}_\mathbf{k}^{\mathrm{T}}\mathbf{u}_i - \overline{\mathbf{x}}^{\mathrm{T}}\mathbf{u}_i\right)^2 = \sum_{i=K+1}^{m}\mathbf{u}_i^{\mathrm{T}}\mathbf{S}\mathbf{u}_i$$

- Опять получается, что $\mathbf{u}_i$ должны быть собственными векторами $\mathbf{S}$, то же самое.

Сингулярное разложение

- **Сингулярным** или **SVD-разложением** (singular value decomposition) матрицы $\mathbf{X}$ размера $n \times m$ называется ее представление в виде

$$\underbrace{\mathbf{X}}_{n \times m} = \underbrace{\mathbf{U}}_{n \times d} \times \underbrace{\mathbf{D}}_{d \times d} \times \underbrace{\mathbf{V}^{\mathrm{T}}}_{d \times d}$$

- $\mathbf{U}$ - матрица с ортонормированными столбцами ($\mathbf{U}^{\mathrm{T}}\mathbf{U} = 1$)
- $\mathbf{V}$ - ортогональная $d \times d$ матрица ($\mathbf{V}^{\mathrm{T}} = \mathbf{V}^{-1}$)
- $\mathbf{D} = diag(\sigma_1, .., \sigma_d)$ - сингулярные значения матрицы $\mathbf{X}$
- Столбцы $\mathbf{U}$ - левые сингулярные векторы
- Столбцы $\mathbf{V}$ - правые сингулярные векторы
- Столбцы $\mathbf{u}_1, .., \mathbf{u}_d$ матрицы $\mathbf{U}$ представляют собой ортонормированный базис подпространства, натянутого на столбцы $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_d$ матрицы $\mathbf{X}$

Сингулярное разложение

- Как $\sigma_1, \sigma_2, ..., \sigma_d$ зависят от данных?
- Выборочная матрица ковариаций равна $\mathbf{S} = \mathbf{X}^{\mathrm{T}}\mathbf{X}/n$. Отсюда

$$\mathbf{S} = \frac{1}{n}\mathbf{X}^{\mathrm{T}}\mathbf{X} = \frac{1}{n}\mathbf{V}\mathbf{D}\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}} = \mathbf{V}\mathbf{D}^2\mathbf{V}^{\mathrm{T}}/n$$

- Итак, столбцы $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_d$ матрицы $\mathbf{V}$ представляют собой собственный базис матрицы ковариаций $\mathbf{S}$
- σ_j^2/n - собственные числа этой матрицы
- Векторы $\mathbf{v}_j$ называются также **главными (или основными) компонентами** (principal components) для данных $\mathbf{X}$.

Сингулярное разложение

- Пусть $\mathbf{z}_j = \mathbf{X}\mathbf{v}_j$
- Легко проверить, что

$$Var(\mathbf{z}_j) = Var(\mathbf{X}\mathbf{v}_j) = \sigma_j^2/n, \quad \mathbf{z}_j = \mathbf{X}\mathbf{v}_j = \sigma_j^2\mathbf{u}_j$$

- Первая главная компонента $\mathbf{u}_1$ обладает тем свойством, что $\mathbf{z}_1$ имеет максимальную дисперсию среди всех нормированных линейных комбинаций столбцов матрицы $\mathbf{X}$.
- Вектор $\mathbf{u}_j$ выбран среди всех векторов, ортогональных $\mathbf{u}_1, ..., \mathbf{u}_{j-1}$ так, что $\mathbf{z}_j$ имеет максимальную дисперсию.
- Вектор $\mathbf{z}_d$ имеет минимальную дисперсию.

EM-алгоритм

Алгоритм Expectation–Maximization

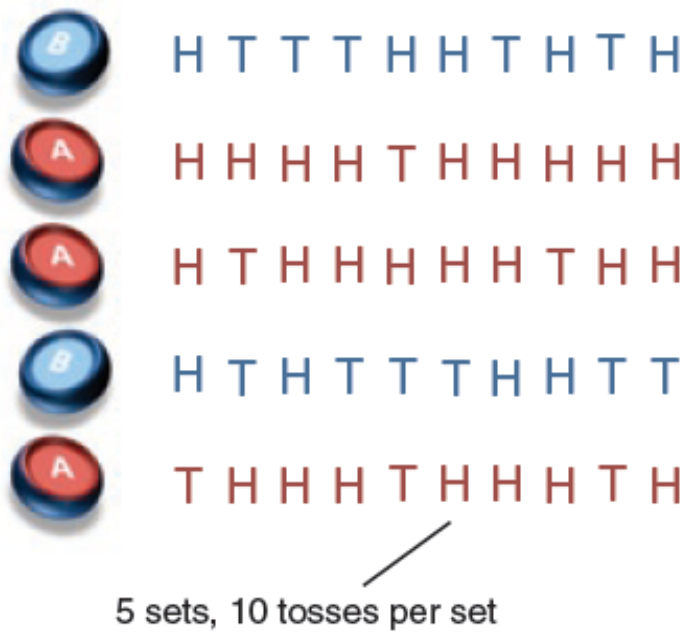
- Две монеты A и B с неизвестными вероятностями орла θ_A и θ_B
- Оценить $\theta = (\theta_A, \theta_B)$

Метод 1:

- 5 раз случайно выбираем с равной вероятностью одну из монет и 10 раз ее кидаем
- Накапливаем $x = (x_1, .., x_5)$ и $z = (z_1, .., z_5)$, $x_i \in \{1, ..., 10\}$ - # орлов в i-ом выборе, $z_i \in \{A, B\}$.
- Оценка максимального правдоподобия:

$$\widehat{\theta}_A = \frac{\#\text{ орлов у монеты } A}{\text{общее } \#\text{ бросаний } A}, \quad \widehat{\theta}_B = \frac{\#\text{ орлов у монеты } B}{\text{общее } \#\text{ бросаний } B}$$

EM алгоритм



Coin A	Coin B
	5 H, 5 T
9 H, 1 T	
8 H, 2 T	
	4 H, 6 T
7 H, 3 T	
24 H, 6 T	9 H, 11 T

$$\hat{\theta}_A = \frac{24}{24+6} = 0.80$$

$$\hat{\theta}_B = \frac{9}{9+11} = 0.45$$

EM алгоритм - Метод 2

1. Предположим, что не знаем $z_i \in \{A, B\}$
2. Начальное приближение $\widehat{\theta}^{(0)} = (\widehat{\theta}_A^{(0)}, \widehat{\theta}_B^{(0)})$
3. Определяем для каждого из пяти множеств, монета A или B была наиболее вероятна, используя $\widehat{\theta}^{(t)}$ (**E-шаг**)
4. Определив $\Pr\{A\}$ и $\Pr\{B\}$, вычисляем оценку максимального правдоподобия $\widehat{\theta}^{(t+1)}$ (**M-шаг**)
5. Все рекурсивно повторяется с Шага 3

EM алгоритм (E-шаг)

$$\Pr\{Z_1 = A \mid X_1 = x_1, \theta = \widehat{\theta}^{(0)}\} = \frac{\Pr\{X_1 = x_1, Z_1 = A \mid \widehat{\theta}^{(0)}\}}{\Pr\{X_1 = x_1 \mid \widehat{\theta}^{(0)}\}}$$

$$= \frac{\left(\widehat{\theta}_A^{(0)}\right)^{x_1}\left(1-\widehat{\theta}_A^{(0)}\right)^{10-x_1}}{\left(\widehat{\theta}_A^{(0)}\right)^{x_1}\left(1-\widehat{\theta}_A^{(0)}\right)^{10-x_1} + \left(\widehat{\theta}_B^{(0)}\right)^{x_1}\left(1-\widehat{\theta}_B^{(0)}\right)^{10-x_1}}$$

$$E\{\#\text{ орлов у } A \mid X_1 = x_1, \widehat{\theta}^{(0)}\} = x_1 \cdot \Pr\{Z_1 = A \mid x_1, \widehat{\theta}^{(0)}\}$$

$$E\{\#\text{ орлов у } B \mid X_1 = x_1, \widehat{\theta}^{(0)}\} = x_1 \cdot \Pr\{Z_1 = B \mid x_1, \widehat{\theta}^{(0)}\}$$

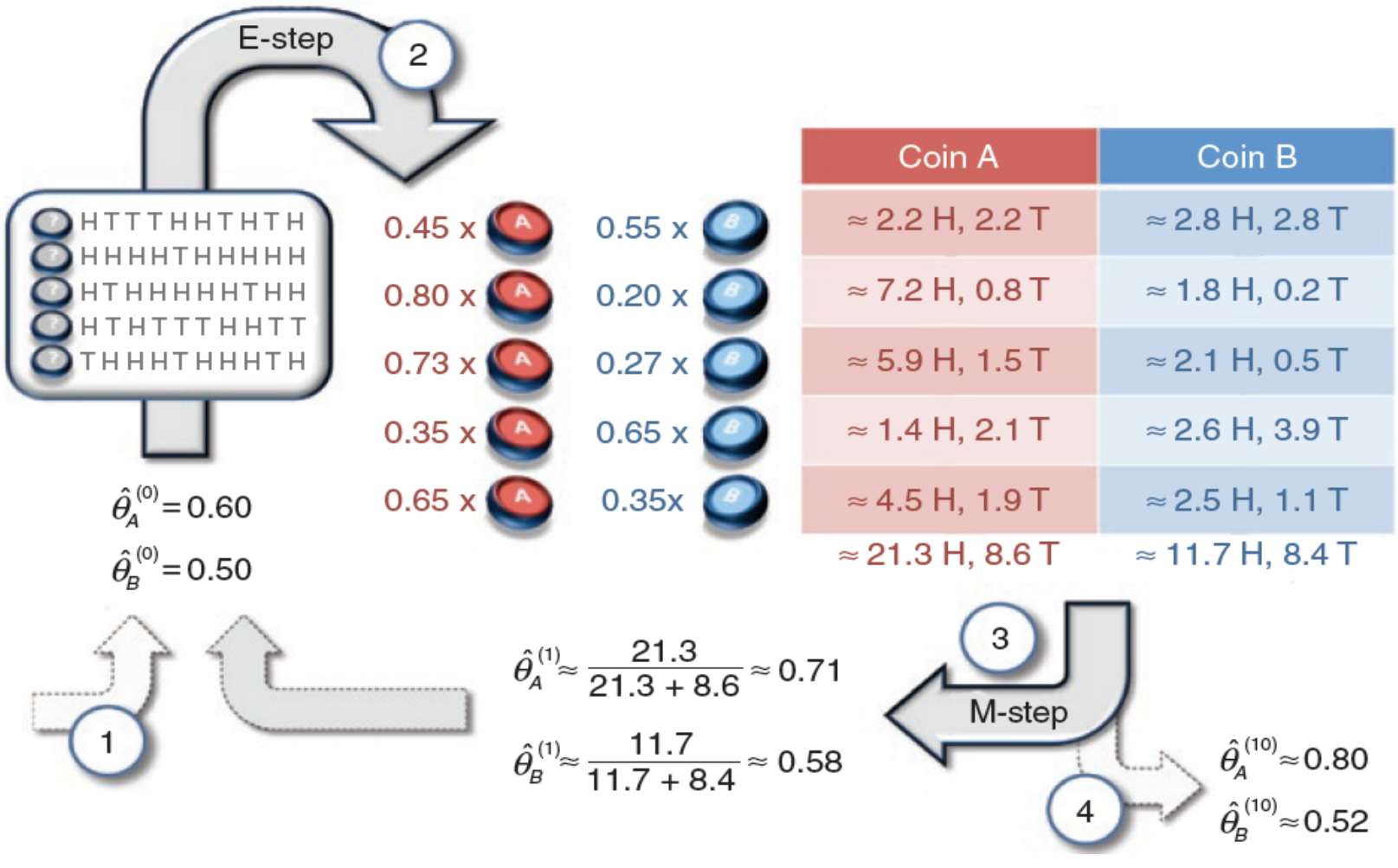
$$E\{\#\text{ орлов у } A \mid \widehat{\theta}^{(0)}\} = \sum_{i=1}^{5} E\{\#\text{ орлов у } A \mid X_i = x_i, \widehat{\theta}^{(0)}\}$$

EM алгоритм (M-шаг)

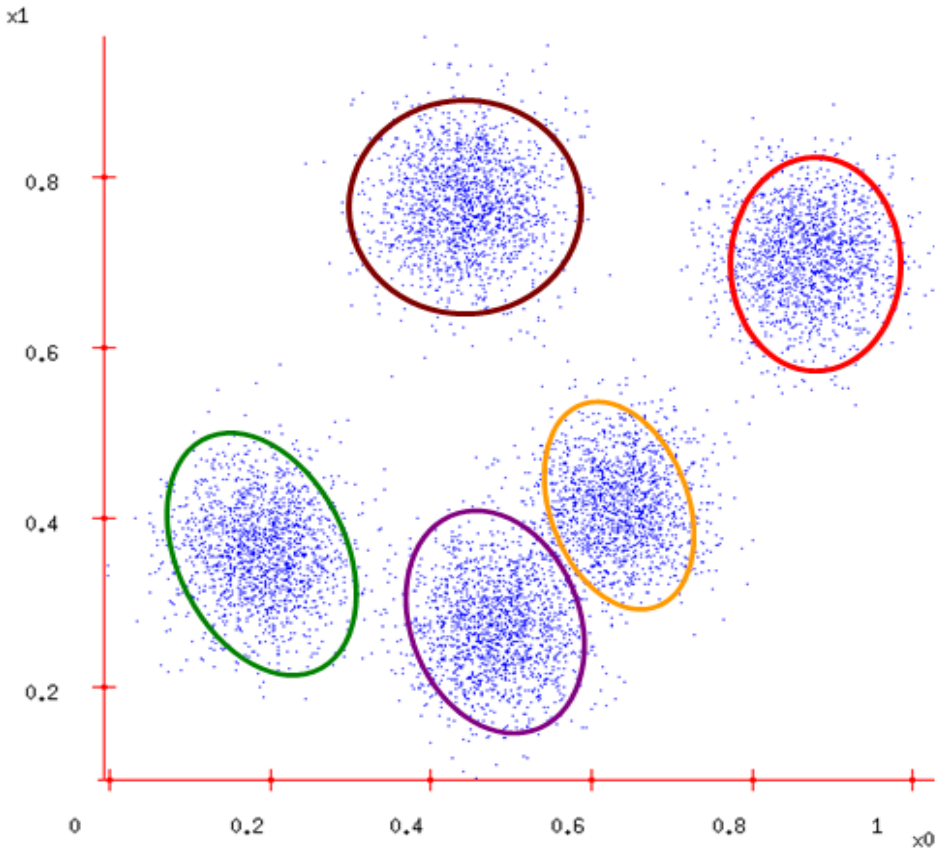
$$\widehat{\theta}_A^{(1)} = \frac{E\{\#\text{ орлов у } A \mid \widehat{\theta}^{(0)}\}}{E\{\#\text{ орлов у } A \mid \widehat{\theta}^{(0)}\} + E\{\#\text{ решек у } A \mid \widehat{\theta}^{(0)}\}}$$

$$\widehat{\theta}_B^{(1)} = \frac{E\{\#\text{ орлов у } B \mid \widehat{\theta}^{(0)}\}}{E\{\#\text{ орлов у } B \mid \widehat{\theta}^{(0)}\} + E\{\#\text{ решек у } B \mid \widehat{\theta}^{(0)}\}}$$

EM алгоритм


E-step
2
H T T T H H T H T H
H H H H T H H H H H
H T H H H H H T H H
H T H T T T H H T T
T H H H T H H H T H
0.45 x A
0.80 x A
0.73 x A
0.35 x A
0.65 x A
0.55 x B
0.20 x B
0.27 x B
0.65 x B
0.35x B
Coin A
Coin B
≈ 2.2 H, 2.2 T
≈ 2.8 H, 2.8 T
≈ 7.2 H, 0.8 T
≈ 1.8 H, 0.2 T
≈ 5.9 H, 1.5 T
≈ 2.1 H, 0.5 T
≈ 1.4 H, 2.1 T
≈ 2.6 H, 3.9 T
≈ 4.5 H, 1.9 T
≈ 2.5 H, 1.1 T
≈ 21.3 H, 8.6 T
≈ 11.7 H, 8.4 T
$\hat{\theta}_A^{(0)} = 0.60$
$\hat{\theta}_B^{(0)} = 0.50$
1
$\hat{\theta}_A^{(1)} \approx \frac{21.3}{21.3 + 8.6} \approx 0.71$
$\hat{\theta}_B^{(1)} \approx \frac{11.7}{11.7 + 8.4} \approx 0.58$
3
M-step
4
$\hat{\theta}_A^{(10)} \approx 0.80$
$\hat{\theta}_B^{(10)} \approx 0.52$

Алгоритм Expectation–maximization (EM алгоритм)



EM алгоритм - постановка

- Предположим, что данные в кластерах имеют нормальное распределение
- Каждый кластер (точнее его нормальное распределение) характеризуется
  - математическим ожиданием m_i, $i = 1, ..., K$
  - дисперсией σ_k^2
- Необходимо определить степень принадлежности γ_{ik} того, что i-й объект принадлежит k-й компоненте смеси или k-му кластеру.
- Кластер $k_0 = \arg\max_{k=1,...,K} \gamma_{ik}$

EM алгоритм - вероятностная основа

- Вероятность любой точки $\mathbf{x}$ равна (смесь нормальных распределений)

$$p(\mathbf{x}) = \sum_{k=1}^{K} \delta_k p_{X_k}(\mathbf{x}), \quad \delta_k = \Pr\{Y = k\}$$

$$p_{X_k}(\mathbf{x}) = \mathcal{N}(\mathbf{x}, m_k, \sigma_k) = \varphi(\mathbf{x}, m_k, \sigma_k) = \\ = \frac{1}{\sigma_k \sqrt{2\pi}} \exp\left(-\frac{(\mathbf{x} - m_k)^2}{\sigma_k^2}\right),$$

- Необходимо по выборке $\mathbf{x}_1, ..., \mathbf{x}_n$ восстановить $m_1, ..., m_K$ и $\sigma_1, ..., \sigma_K$ и $\delta_1, ..., \delta_K$

Ожидание — Expectation шаг в EM алгоритме

Если параметры $m_1, ..., m_K$ и $\sigma_1, ..., \sigma_K$ и $\delta_1, ..., \delta_K$ известны, то апостериорные вероятности:

$$\gamma_{ik} = \Pr\{k \mid \mathbf{x}_i\} = \frac{\Pr\{k\} \cdot p(\mathbf{x}_i \mid k)}{p(\mathbf{x}_i)} = \frac{\delta_k \cdot \varphi(\mathbf{x}_i, m_k, \sigma_k)}{\sum_{j=1}^{K} \delta_j \cdot \varphi(\mathbf{x}_i, m_j, \sigma_j)}$$

$$i = 1, ..., n, \quad k = 1, ..., K.$$

Максимизация — Maximization шаг в EM алгоритме

Если знаем апостериорные вероятности γ_{ik}, то можно оценить параметры $m_1, ..., m_K$ и $\sigma_1, ..., \sigma_K$ и $\delta_1, ..., \delta_K$:

$$m_k = \frac{1}{\zeta_k}\sum_{i=1}^{n}\gamma_{ik}\mathbf{x}_i, \quad \sigma_k^2 = \frac{1}{\zeta_k}\sum_{i=1}^{n}\gamma_{ik}\left(\mathbf{x}_i - m_k\right)^2,$$

$$\delta_k = \frac{1}{n}\sum_{i=1}^{n}\gamma_{ik}, \quad \zeta_k = \sum_{i=1}^{n}\gamma_{ik}$$

В случае смеси d-мерных распределений для пересчета матрицы ковариации используем формулу:

$$\Sigma_k = \frac{1}{\zeta_k}\sum_{i=1}^{n}\gamma_{ik}\left(\mathbf{x}_i - m_k\right)\left(\mathbf{x}_i - m_k\right)^{\mathrm{T}}, \quad k = 1, ..., K$$

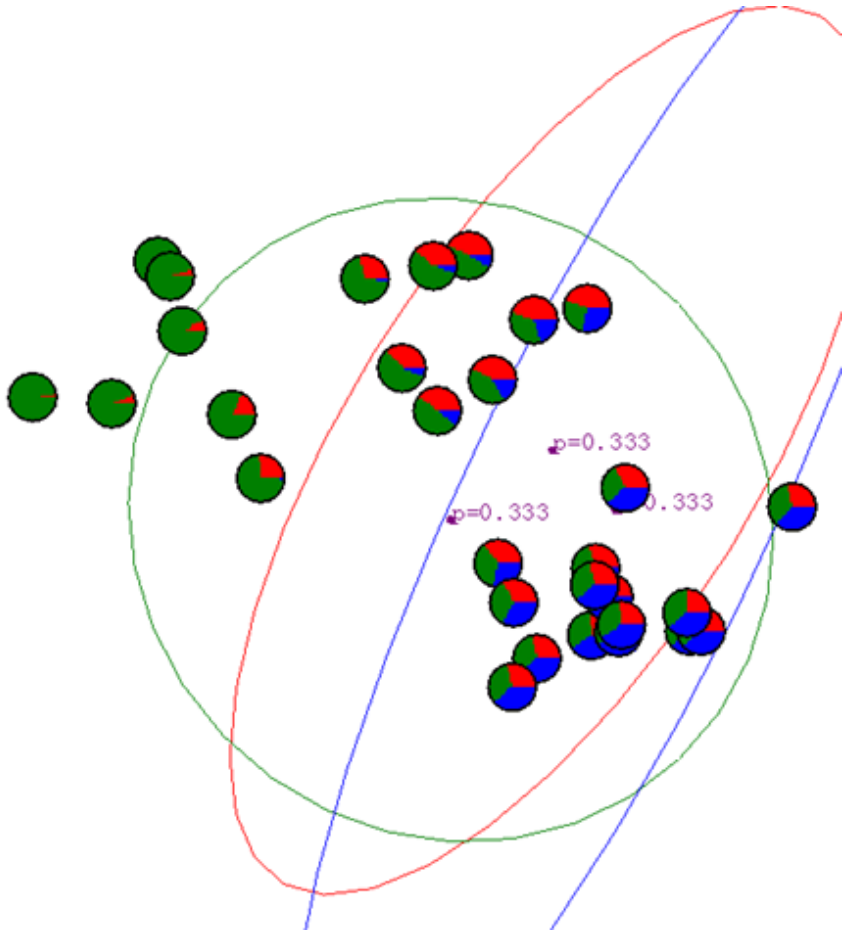
EM алгоритм (сам)

EM-алгоритм заключается в итерационном повторении Expectation и Maximization шагов.

1. Начальное приближение $m_1, ..., m_K$ и $\sigma_1, ..., \sigma_K$ и $\delta_1, ..., \delta_K$
2. E-шаг (Expectation)
3. M-шаг (Maximization)
4. Кластер $k_0 = \arg\max_{k=1,...,K} \gamma_{ik}$
5. Переход на Шаг 2 пока k_0 не будет изменяться

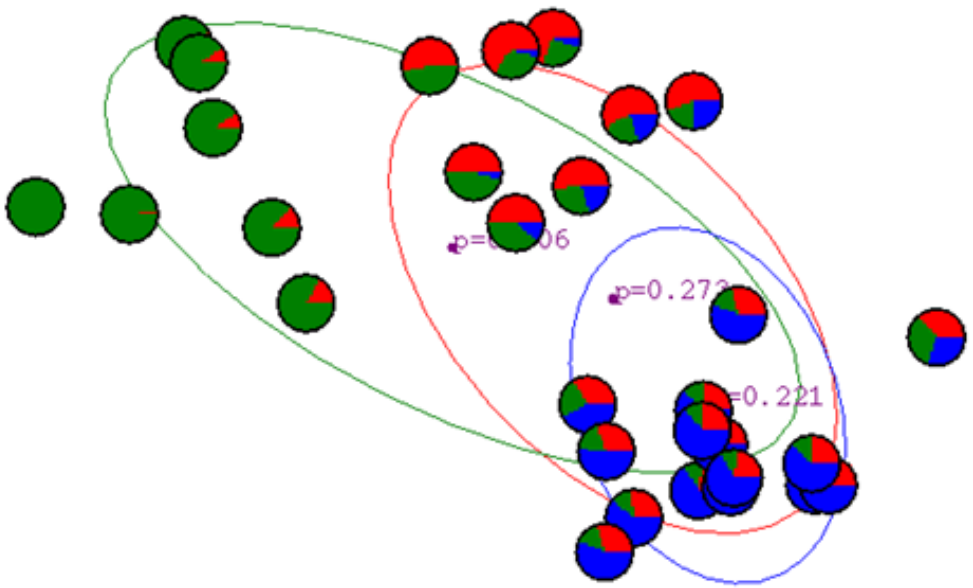
EM алгоритм (иллюстрация)

Начальное приближение



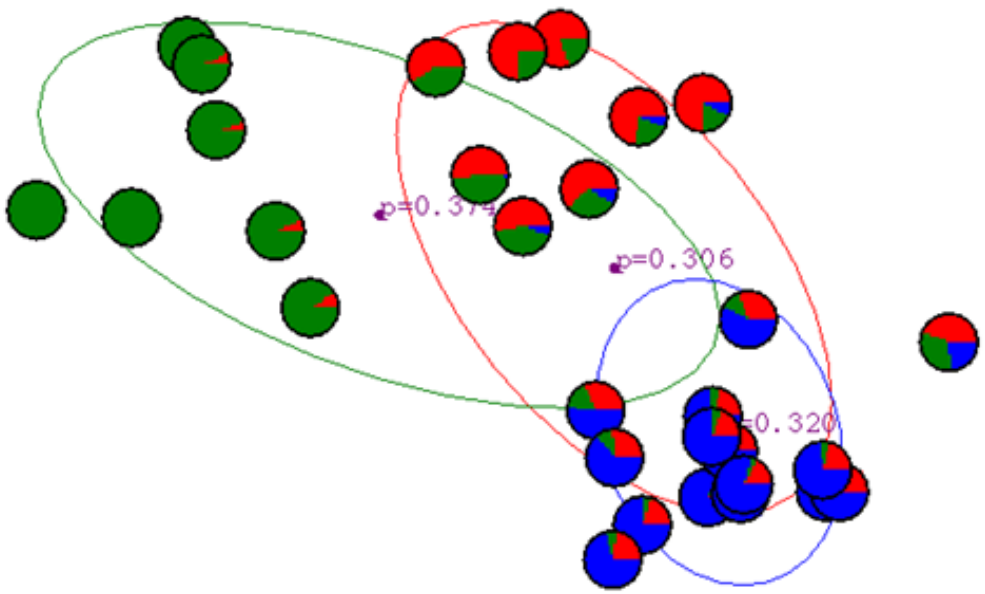
EM алгоритм (иллюстрация)

После 1-ой итерации



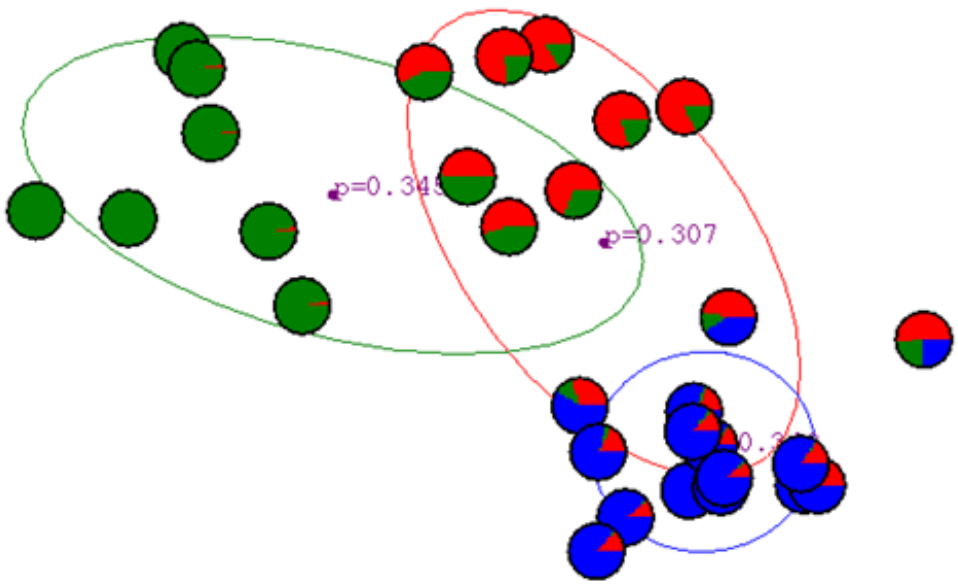
EM алгоритм (иллюстрация)

После 2-ой итерации



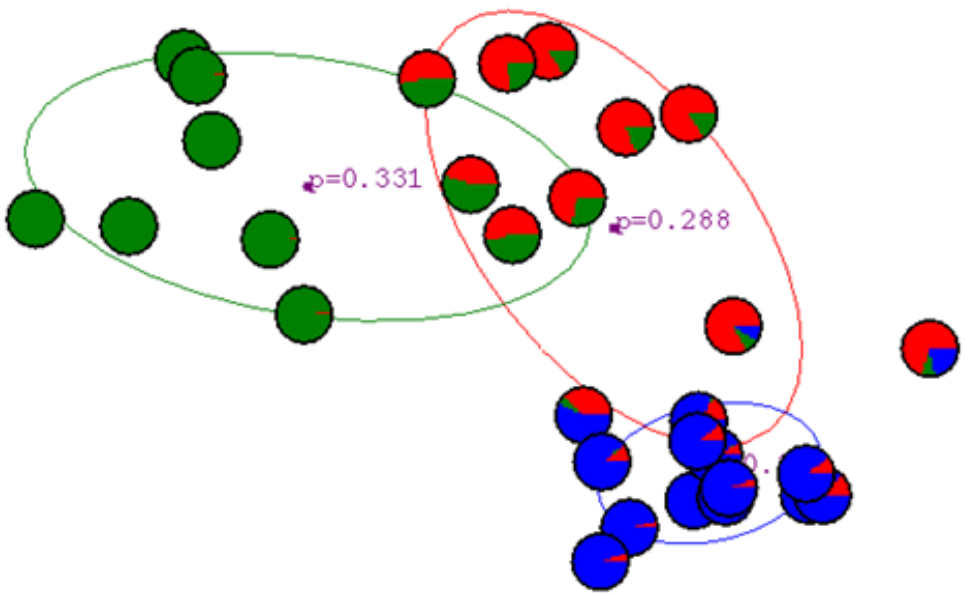
EM алгоритм (иллюстрация)

После 3-ей итерации



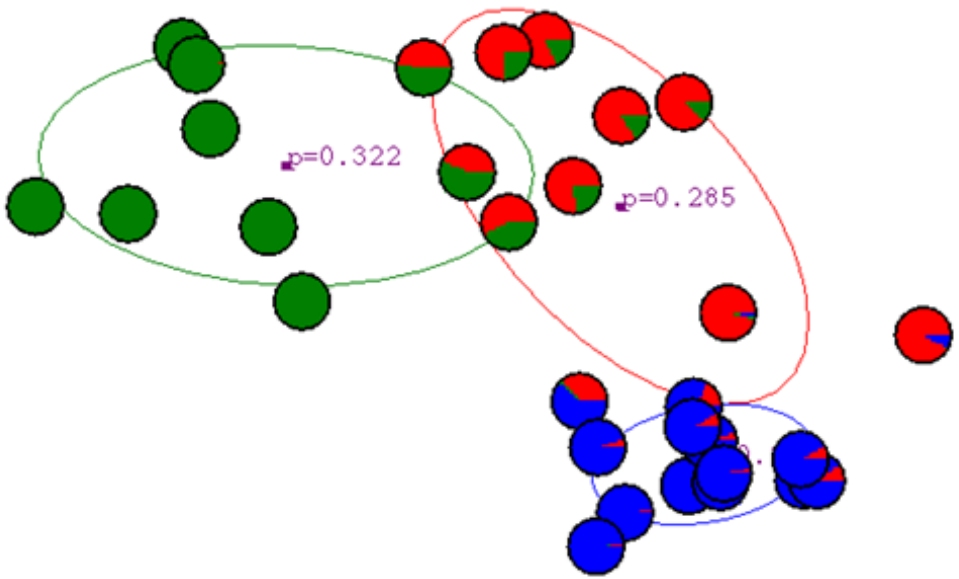
EM алгоритм (иллюстрация)

После 4-ой итерации



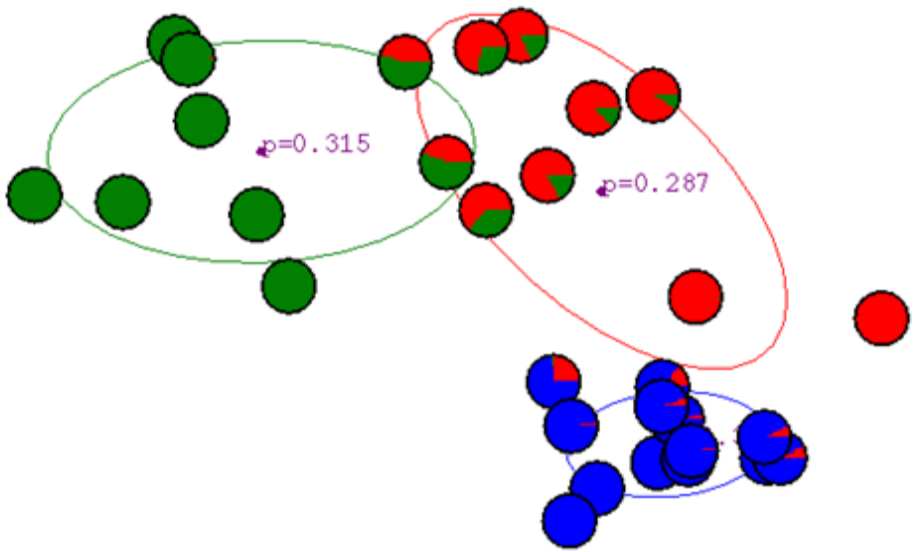
EM алгоритм (иллюстрация)

После 5-ой итерации



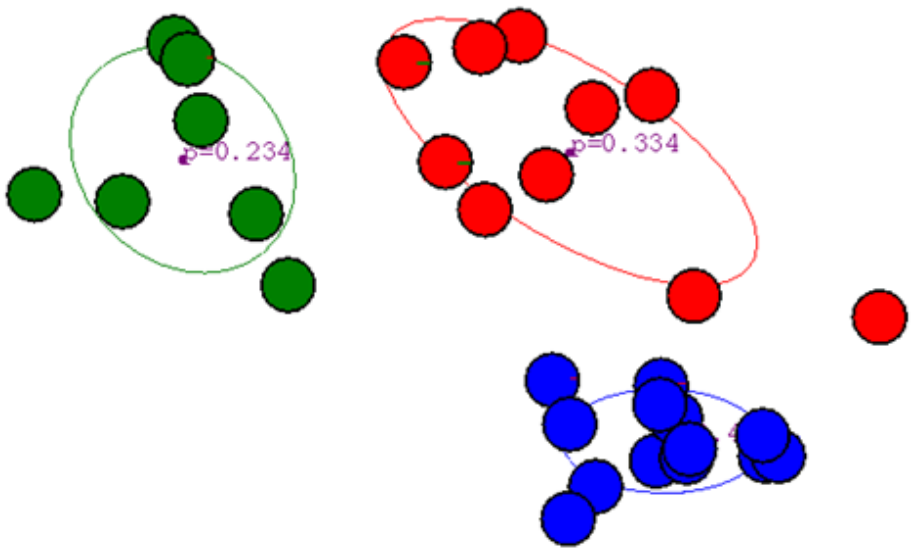
EM алгоритм (иллюстрация)

После 6-ой итерации



EM алгоритм (иллюстрация)

После 20-ой итерации



Основные отличия EM-алгоритма от k-means

1. **EM**: мягкая кластеризация: $\gamma_{ik} = P\{\mathbf{x}_i \in C_k\}$
   k-means: жесткая кластеризация: $\gamma_{ik} = [\mathbf{x}_i \in C_k]$
2. **EM**: форма кластеров эллиптическая, настраиваемая
   k-means: форма кластеров жестко определяется метрикой ρ

Программная реализация в R

- Пакет **stats,** функция **kmeans**
- Пакет **amap**, функция **Kmeans**
- Пакет **Ckmeans.1d.dp**, функция **Ckmeans.1d.dp** (для одномерных данных)
- Пакет **cluster**, функция **clara** (универсальная)
- Пакет **EMCluster**, функция **emcluster** (EM алгоритм)

Программная реализация в R

- Пакет **labdsv**, функция **princomp** (PCA)
- Пакет **Rsafd**, функция **prcomp** (PCA)
- Пакет **labdsv**, функция **pca** (PCA)

Вопросы

?