



# ТЕХНОСФЕРА

## Лекция 3 Алгоритмы кластеризации I

Николай Анохин

5 октября 2015 г.

## Краткое содержание предыдущей лекции

**Дано.** Признаковые описания  $N$  объектов  $\mathbf{x} = (x_1, \dots, x_m) \in \mathcal{X}$ , образующие тренировочный набор данных  $X$

**Найти.** Модель из семейства параметрических функций

$$H = \{h(\mathbf{x}, \theta) : \mathcal{X} \times \Theta \rightarrow \mathcal{Y} \mid \mathcal{Y} = \{1, \dots, K\}\},$$

ставящую в соответствие произвольному  $\mathbf{x} \in \mathcal{X}$  один из  $K$  кластеров так, чтобы объекты внутри одного кластера были похожи, а объекты из разных кластеров различались

## Краткое содержание предыдущей лекции

- Смоделировали данные как смесь нормальных распределений

$$p(\mathbf{x}|\mu_k, \Sigma_k) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

- С применением Maximum Likelihood получили выражения для шагов Е и М
- Предельным переходом получили алгоритм k-means, минимизирующий инерцию

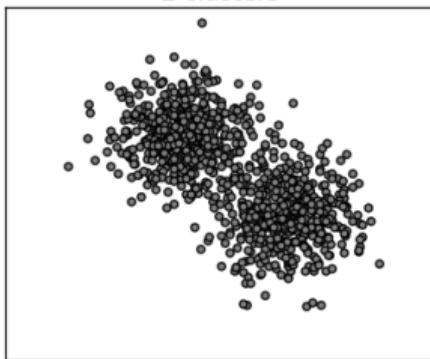
$$\tilde{J}(\mu) = \sum_{n=1}^N \sum_{k=1}^K r_{nk} d(\mathbf{x}_n, \mu_k), \quad r_{nk} = \begin{cases} 1, & \text{для } k = \arg \min_j d(\mathbf{x}_n, \mu_j) \\ 0, & \text{иначе} \end{cases}$$

- Формулируя альтернативные  $d(\mathbf{x}_1, \mathbf{x}_2)$  и  $J(\mu)$ , получили семейство алгоритмов, обучаемых с помощью EM

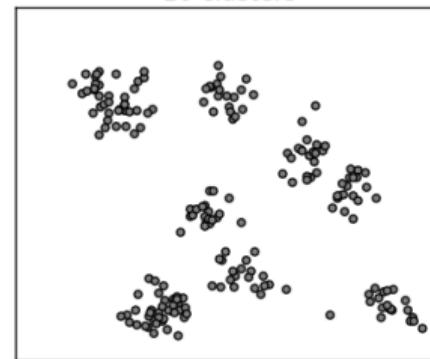
# Нерешенные проблемы



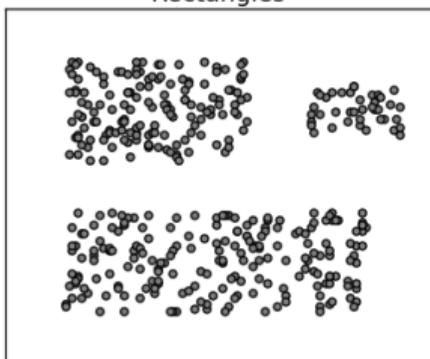
2 clusters



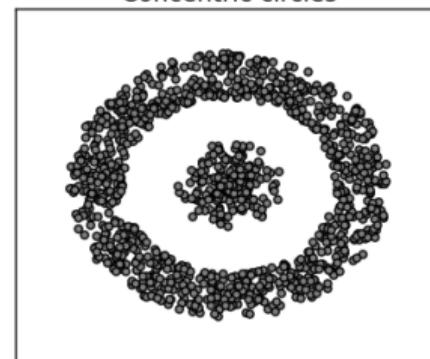
10 clusters



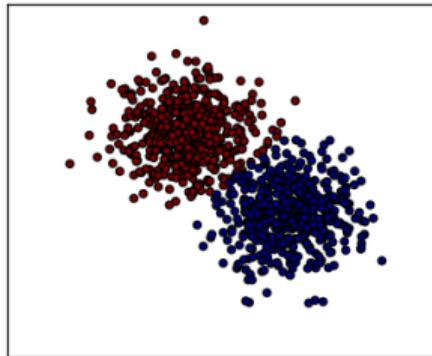
Rectangles



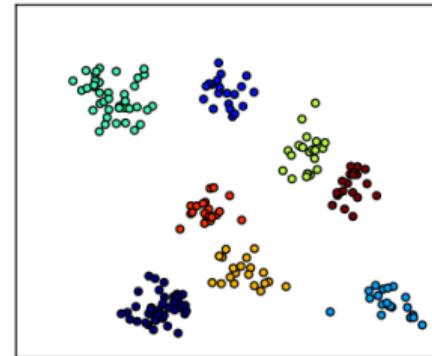
Concentric circles



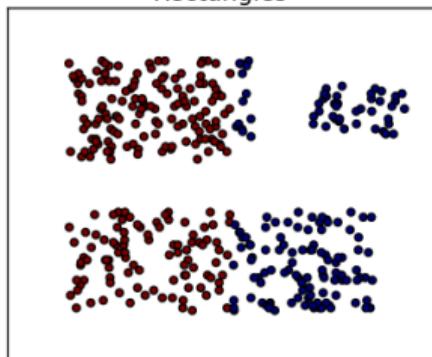
2 clusters



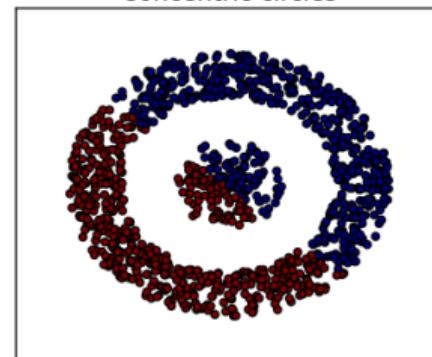
10 clusters



Rectangles



Concentric circles



# План занятия

Функции расстояния

Выбор количества кластеров

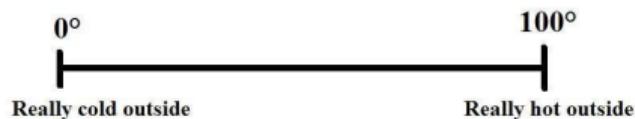
Иерархическая кластеризация

Алгоритмы, основанные на плотности

Качество кластеризации

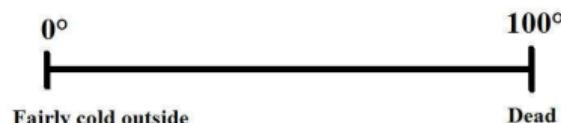
# Функции расстояния

Fahrenheit



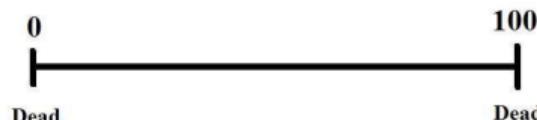
VS

Celsius



VS

Kelvin



## Модификации алгоритма

Взять уже известную нам функцию потерь (инерцию) и “поиграть” с функцией расстояния.

$$\tilde{J}(\mu) = \sum_{n=1}^N \sum_{k=1}^K r_{nk} d(\mathbf{x}_n, \mu_k), \quad r_{nk} = \begin{cases} 1, & \text{для } k = \arg \min_j d(\mathbf{x}_n, \mu_j) \\ 0, & \text{иначе} \end{cases}$$

# Расстояния 1

- Минковского

$$d_r(\mathbf{x}, \mathbf{y}) = \left[ \sum_{j=1}^N |x_j - y_j|^r \right]^{\frac{1}{r}}$$

- Евклидово  $r = 2$

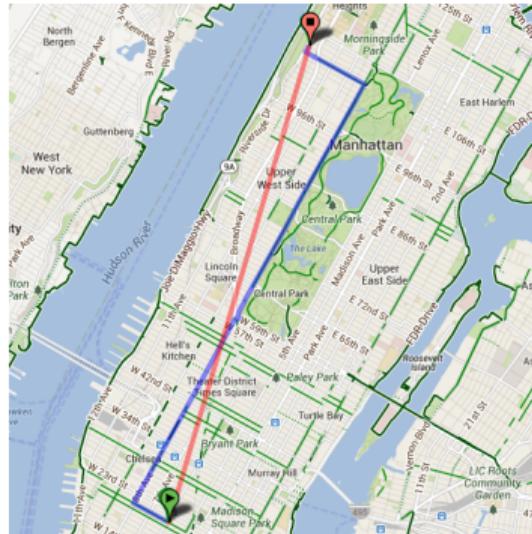
$$d_E(\mathbf{x}, \mathbf{y}) = d_2(\mathbf{x}, \mathbf{y})$$

- Манхэттэн  $r = 1$

$$d_M(\mathbf{x}, \mathbf{y}) = d_1(\mathbf{x}, \mathbf{y})$$

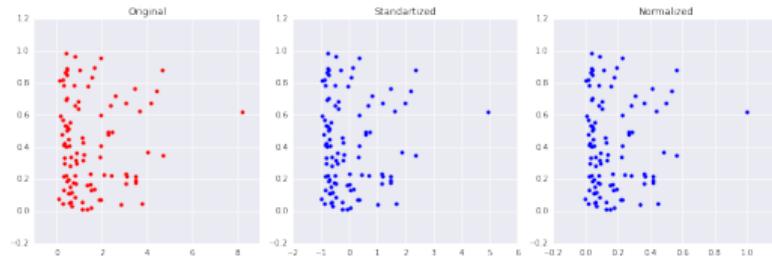
- $r = \infty$

$$d_\infty(\mathbf{x}, \mathbf{y}) = \max_j |x_j - y_j|$$

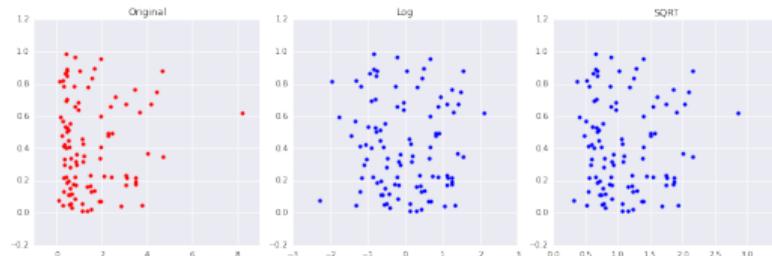


## 1. Функции расстояния чувствительны к “масштабу” данных

- ▶ Преобразовать обучающую выборку так, чтобы признаки имели нулевое среднее и единичную дисперсию (standartization)
- ▶ Преобразовать обучающую выборку так, чтобы значения признаков лежали на отрезке  $[0, 1]$  (normalization)



## 2. Есть шанс улучшить качество, применив монотонное преобразование ( $\log$ , $\sqrt{ }$ )



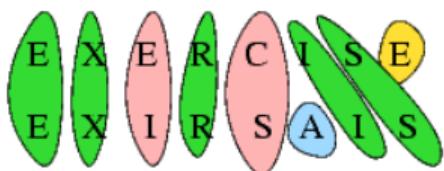
## Расстояния 2

- ▶ Жаккард

$$d_J(\mathbf{x}, \mathbf{y}) = 1 - \frac{|\mathbf{x} \cap \mathbf{y}|}{|\mathbf{x} \cup \mathbf{y}|}$$

- ▶ Косинус

$$d_c(\mathbf{x}, \mathbf{y}) = \arccos \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$



- ▶ Правки

$d_e$  – наименьшее количество удалений и вставок, приводящее  $\mathbf{x}$  к  $\mathbf{y}$ .

- ▶ Хэмминг

$d_H$  – количество различных компонент в  $\mathbf{x}$  и  $\mathbf{y}$ .

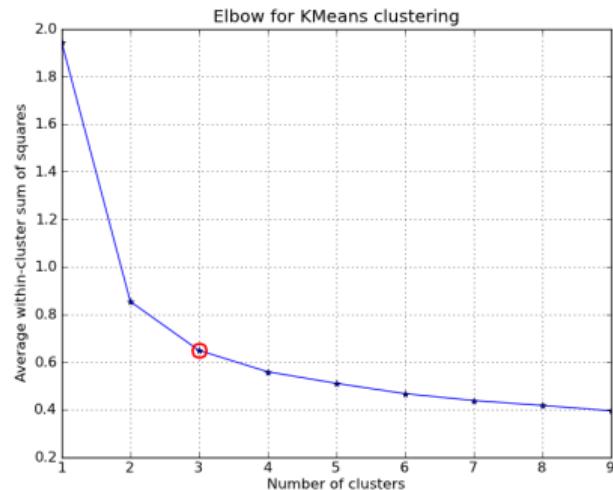
## Выбор количества кластеров

```
int getRandomNumber()
{
    return 4; // chosen by fair dice roll.
              // guaranteed to be random.
}
```

# Выбор наилучшего $K$

*Идея.* Выбрать критерий качества кластеризации и построить его значение для  $K = 1, 2, \dots$

- ▶ средняя сумма квадратов расстояния до центроида
- ▶ средний диаметр кластера



## Критерий Silhouette

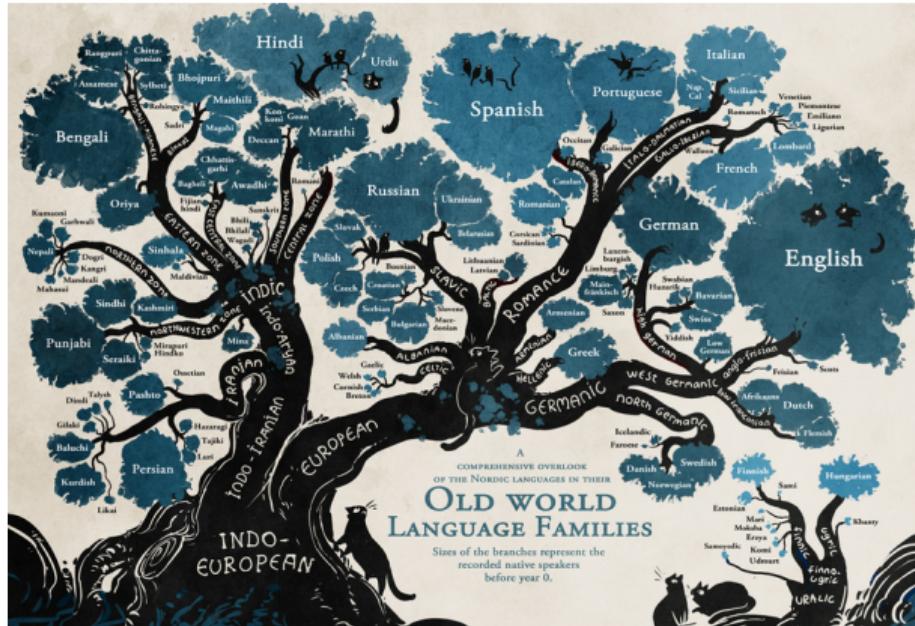
Пусть дана кластеризация в  $K$  кластеров, и объект  $i$  попал в  $C_k$

- ▶  $a(i)$  – среднее расстояние от  $i$  объекта до объектов из  $C_k$
- ▶  $b(i) = \min_{j \neq k} b_j(i)$ , где  $b_j(i)$  – среднее расстояние от  $i$  объекта до объектов из  $C_j$

$$\text{silhouette}(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Средний silhouette для всех точек из  $\mathbf{X}$  является критерием качества кластеризации.

## Иерархическая кластеризация<sup>1</sup>



<sup>1</sup>Feast Your Eyes on This Beautiful Linguistic Family Tree

# Иерархическая кластеризация: идея метода

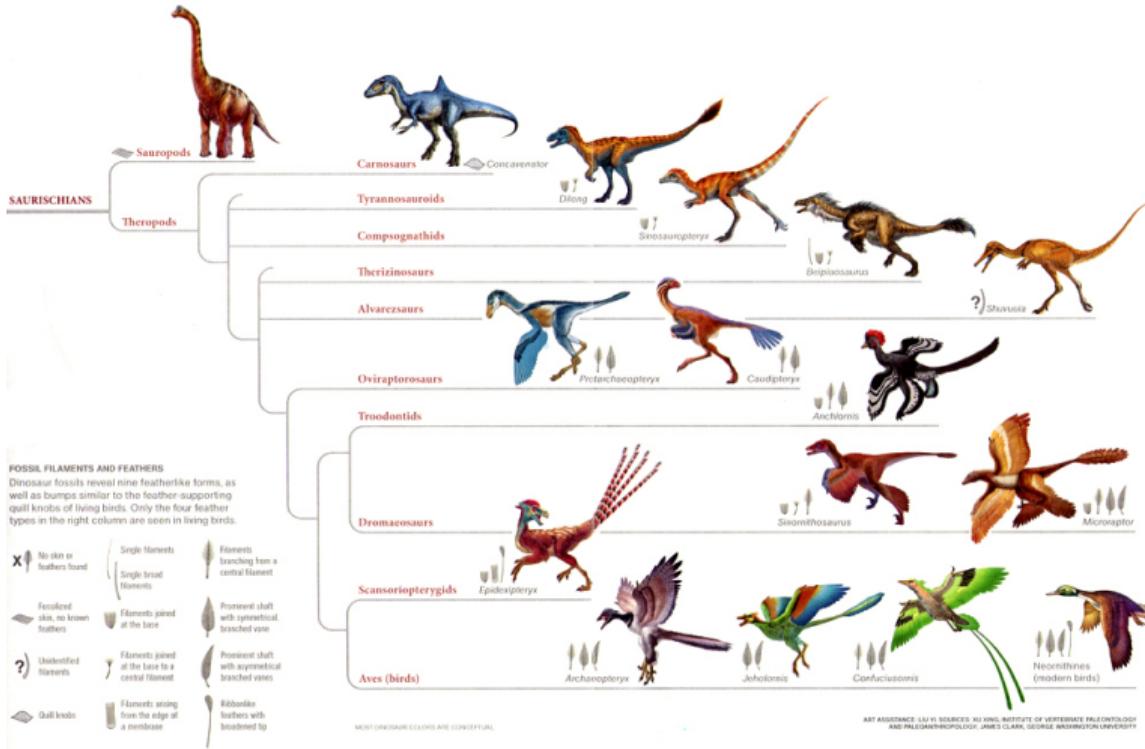
## Agglomerative

1. начинаем с ситуации, когда каждый объект – отдельный кластер
2. на каждом шаге совмещаем два наиболее близких кластера
3. останавливаемся, когда получаем требуемое количество или единственный кластер

## Divisive

1. начинаем с ситуации, когда все объекты составляют один кластер
2. на каждом шаге разделяем два один из кластеров пополам
3. останавливаемся, когда получаем требуемое количество или  $N$  кластеров

# Похожие пернатые динозавры<sup>2</sup>



<sup>2</sup>[http://palaeos.com/vertebrates/coelurosauria/images/feathered\\_dinosaurs.jpg](http://palaeos.com/vertebrates/coelurosauria/images/feathered_dinosaurs.jpg)

## Агломеративный алгоритм

```
1 function agglomerative(X, K):
2     initialize N # number of objects
3     initialize C = N # number of clusters
4     initialize C_i = x_i # initial clusters
5     while C > K:
6         C_a = C_b = None # closest clusters
7         min_dist = +inf # distance between closest
8         for i in 1 .. C:
9             for j in i + 1 .. C:
10                dist = d(C_i, C_j) # dist. betw. clusters
11                if dist < min_dist:
12                    min_dist = dist
13                    C_a = C_i
14                    C_b = C_j
15                merge(C_a, C_b)
16                C = C - 1
17    return C_1, ..., C_K
```

память  $O(N^2)$ , сложность  $O(N^2 \log N)$

# Расстояние между кластерами

- ▶ single-linkage

$$d_{min}(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{x}' \in C_j} \|\mathbf{x} - \mathbf{x}'\|$$

- ▶ complete-linkage

$$d_{max}(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{x}' \in C_j} \|\mathbf{x} - \mathbf{x}'\|$$

- ▶ average

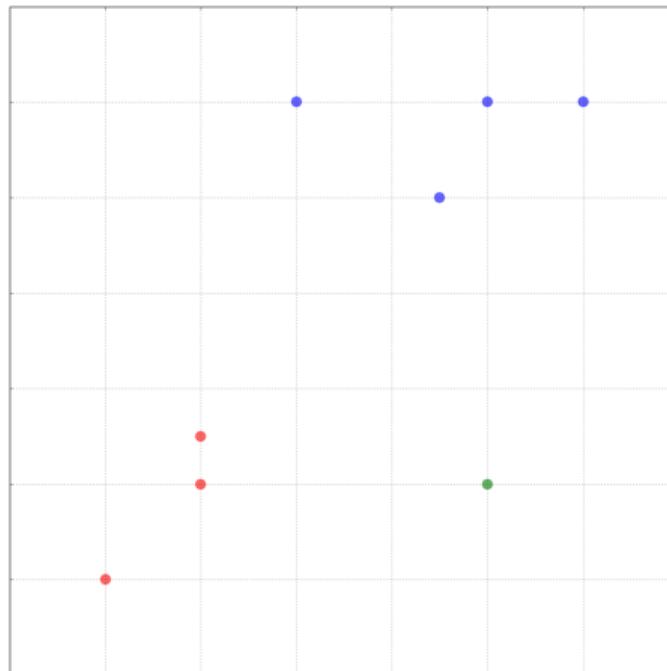
$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{x}' \in C_j} \|\mathbf{x} - \mathbf{x}'\|$$

- ▶ mean

$$d_{mean}(C_i, C_j) = \|\mathbf{m}_i - \mathbf{m}_j\|$$

## Задача

Кластеризовать данные иерархическим методом с использованием расстояний между кластерами  $d_{min}$  и  $d_{max}$



# Stepwise-optimal HC

Какой критерий мы оптимизируем?

```
1 function swo(X, K):
2     initialize N # number of objects
3     initialize C = N # number of clusters
4     initialize C_i = x_i # initial clusters
5     while C > K:
6         # choose the pair that optimizes
7         # the given criterion J when merged
8         C_a, C_b = find_best_merge(J, C_1, ..., C_C)
9         merge(C_a, C_b)
10        C = C - 1
11    return C_1, ..., C_K
```

$d_{max}$  обеспечивает наименьшее увеличение диаметра кластера

$d_e$  обеспечивает наименьшее увеличение квадратичного критерия

$$d_e(C_i, C_j) = \sqrt{\frac{n_i n_j}{n_i + n_j}} \|\mathbf{m}_i - \mathbf{m}_j\|$$

# Неевклидовы пространства

*Проблема.* Как измерить расстояние между кластерами, если невозможно определить центроид?

*Идея.* В каждом из кластеров выбрать “типичный” пример – clustroid.

Минимизируем

- ▶ сумму расстояний до других объектов в кластере
- ▶ сумму квадратов расстояний до других объектов в кластере
- ▶ максимальное расстояние до других объектов в кластере

# Пример. Красные и синие штаты<sup>3</sup>

Красные: Республиканская партия

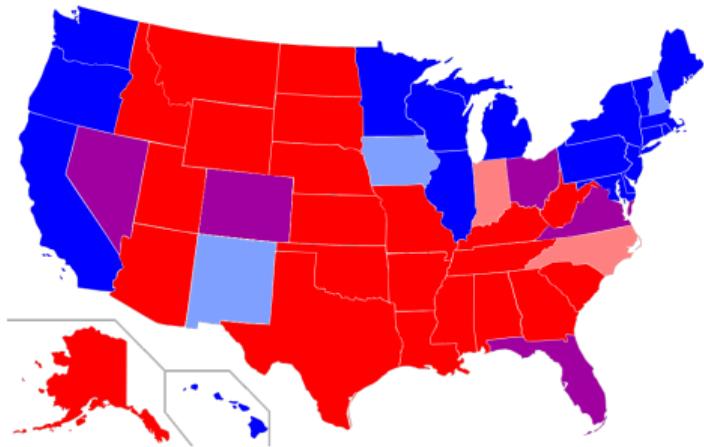
Синие: Демократическая партия

	1940	1944	1948	1952	1956	1960	1964	1968	1972	1976
Alabama	14.340000	18.200000	19.040000	35.02000	39.39000	41.75	69.5	14.0	72.4	43.48
Alaska	40.379375	42.669375	41.900417	55.76375	56.13375	50.94	34.1	45.3	58.1	62.91
Arizona	36.010000	40.900000	43.820000	58.35000	60.99000	55.52	50.4	54.8	64.7	58.62
Arkansas	20.870000	29.840000	21.020000	43.76000	45.82000	43.06	43.9	30.8	68.9	34.97
California	41.350000	42.990000	47.140000	56.39000	55.40000	50.10	40.9	47.8	55.0	50.89

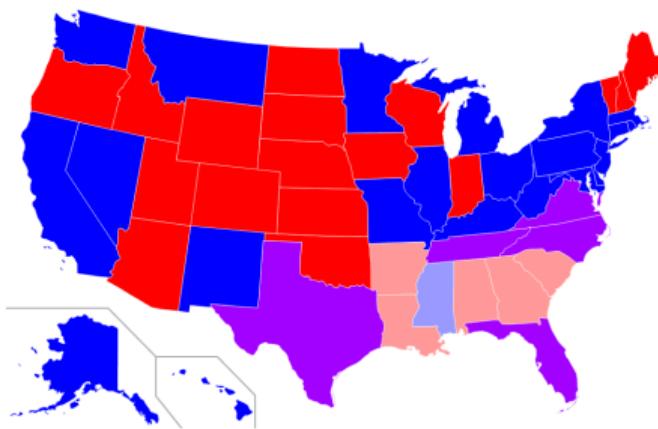
---

<sup>3</sup>[http://www.wikiwand.com/en/Red\\_states\\_and\\_blue\\_states](http://www.wikiwand.com/en/Red_states_and_blue_states)

# Пример. Красные и синие штаты

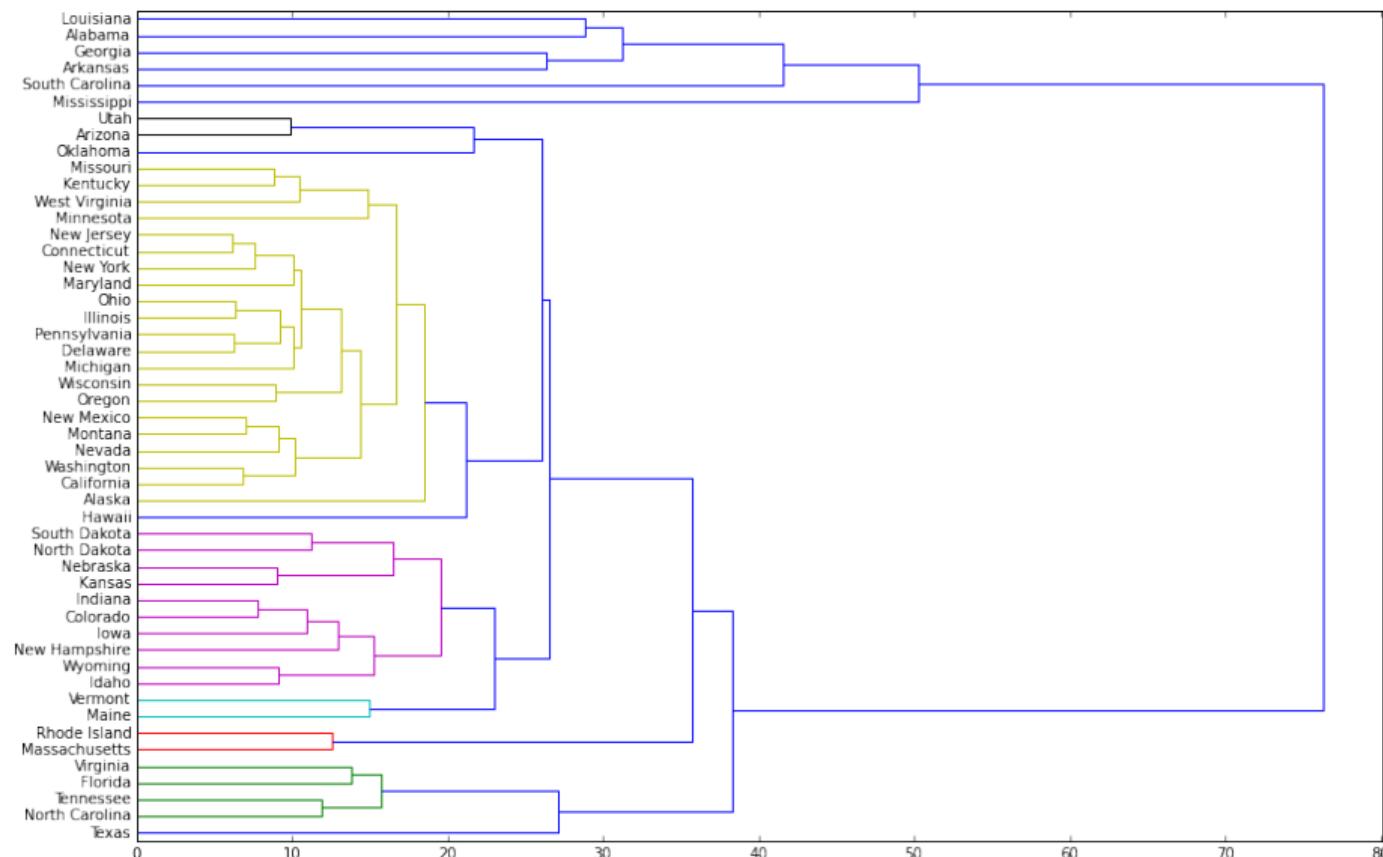


(a) Состояние дел на 2012



(b) Иер. кл. (euc + avg)

## Пример. Красные и синие штаты



## Иерархическая кластеризация: итог

- + Несферические кластеры
- + Разнообразие критериев
- + Любые  $K$  из коробки
- Требует много ресурсов

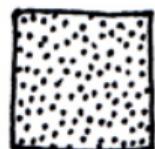
Алгоритмы, основанные на плотности



*thin*



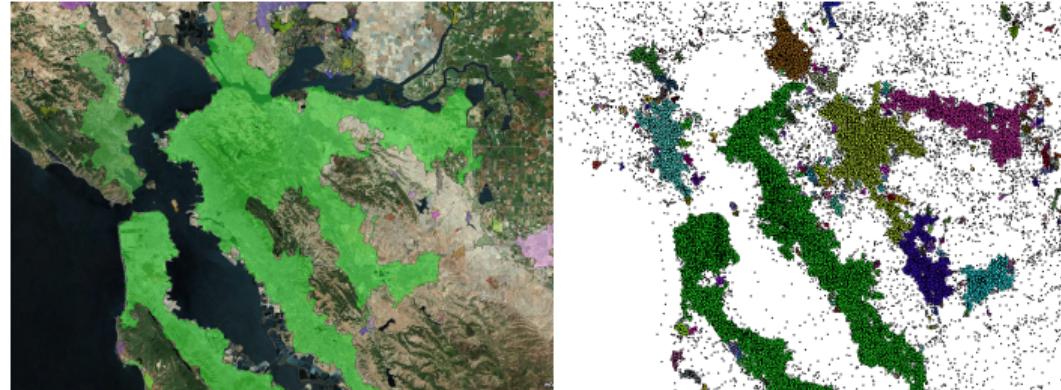
*medium*



*thick*

## Идея метода<sup>4</sup>

- ▶ Кластеризация, основанная на плотности объектов
- ▶ Кластеры – участки высокой плотности, разделенные участками низкой плотности



---

<sup>4</sup><http://biarri.com/spatial-clustering-in-c-post-2-of-5-running-dbscan/>

# Определения

## Плотность

Количество объектов внутри сферы заданного радиуса  $\varepsilon$

## Core-объект

Объект  $x$  является core-объектом, если плотность вокруг него больше  $min\_pts$

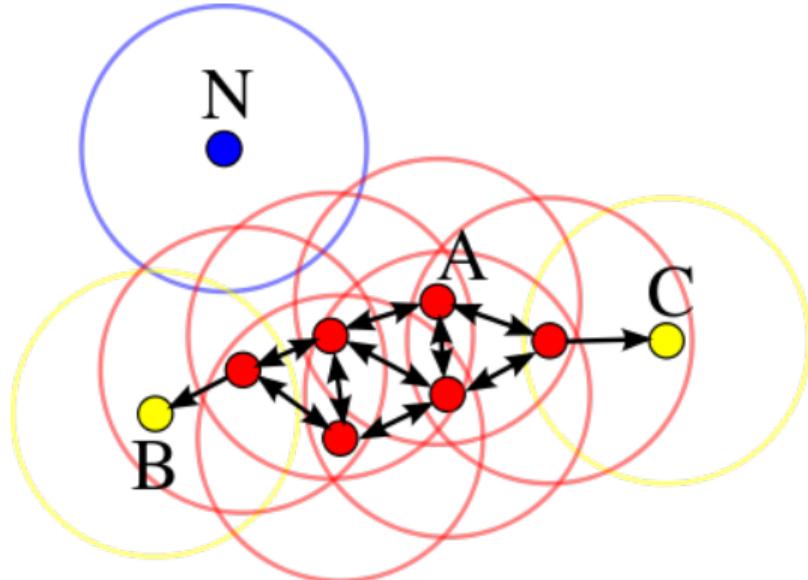
## Граничный-объект

Объект  $x$  является граничным-объектом, если плотность вокруг него меньше  $min\_pts$ , но он находится рядом с core-объектом

## Шум

Объект  $x$  является шумом, если он не является ни core-объектом, ни граничным объектом

## Виды объектов



## DBSCAN 1

```
1 function dbscan(X, eps, min_pts):
2     initialize NV = X # not visited objects
3     for x in NV:
4         remove(NV, x) # mark as visited
5         nbr = neighbours(x, eps) # set of neighbours
6         if nbr.size < min_pts:
7             mark_as_noise(x)
8         else:
9             C = new_cluster()
10            expand_cluster(x, nbr, C, eps, min_pts, NV)
11            yield C
```

## DBSCAN 2

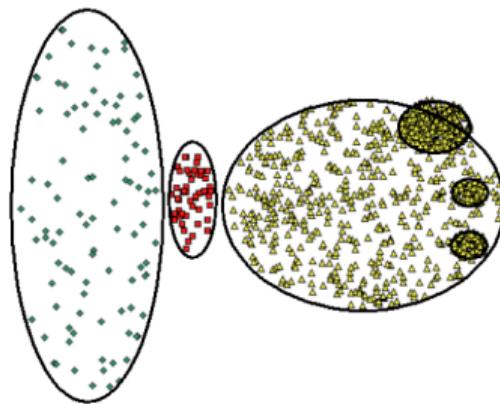
```
1 function expand_cluster(x, nbr, C, eps, min_pts, NV):
2     add(x, C)
3     for x1 in nbr:
4         if x1 in NV: # object not visited
5             remove(NV, x1) # mark as visited
6             nbr1 = neighbours(x1, eps)
7             if nbr1.size >= min_pts:
8                 # join sets of neighbours
9                 merge(nbr, nbr_1)
10            if x1 not in any cluster:
11                add(x1, C)
```

Сложность:  $O(n^2)$  или  $O(n \log n)$  ( $R^* Tree$ )

Память:  $O(n)$  или  $O(n^2)$

## DBSCAN: итог и демо<sup>5</sup>

- + не требует  $K$
- + кластеры произвольной формы
- + учитывает выбросы
- Не вполне детерминированный
- Не работает при разных плотностях кластеров

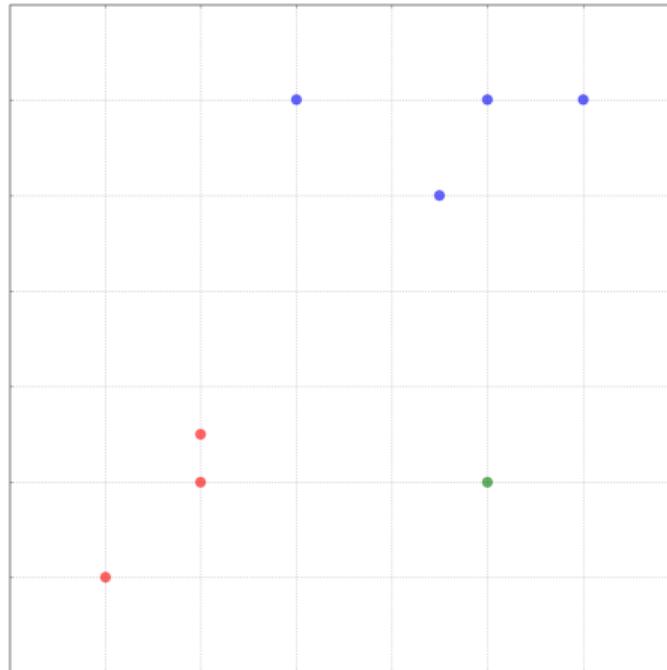


---

<sup>5</sup><http://www.naftaliharris.com/blog/visualizing-dbscan-clustering/>

# Задача

Кластеризовать данные методом DBSCAN



# OPTICS

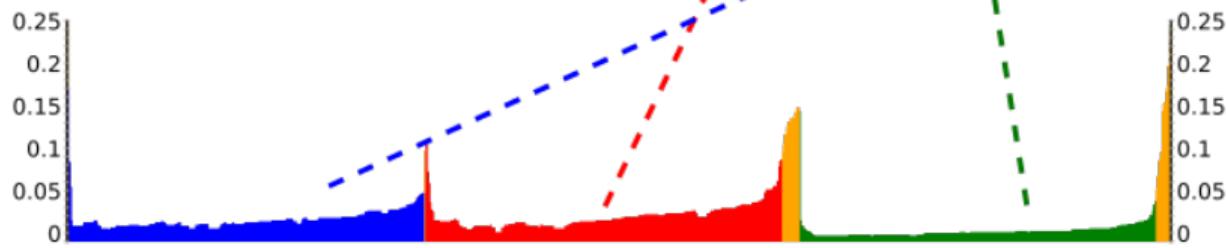
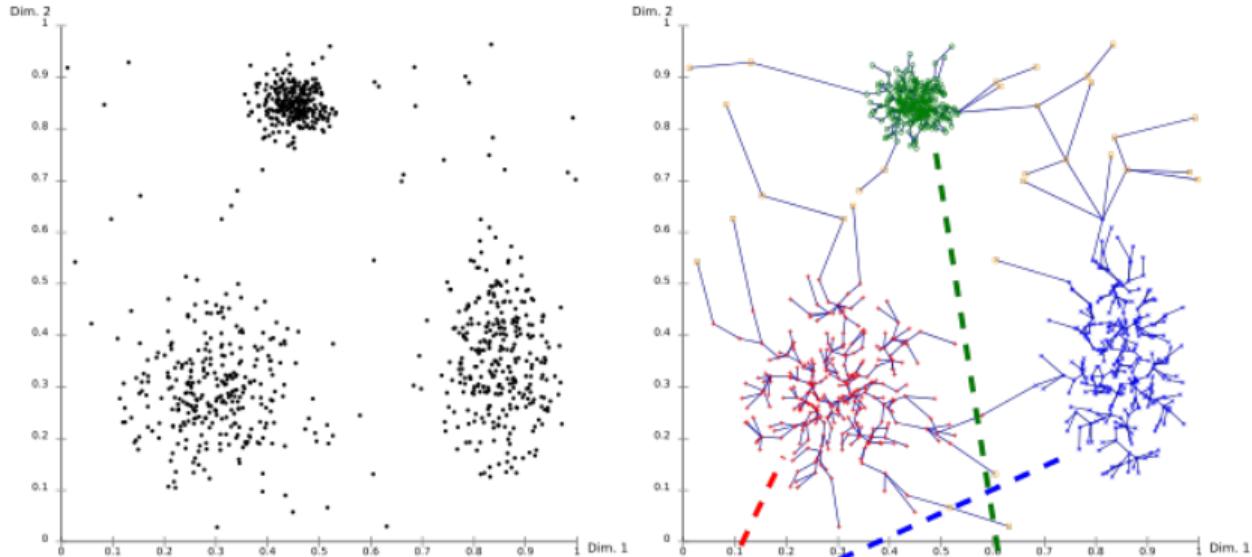
Ordering points to identify the clustering structure

Идея. не ограничиваться фиксированной плотностью, как в DBSCAN, а варьировать ее в зависимости от того, как много объектов попадают в  $\varepsilon$ -окрестность.

Пара дополнительных обозначений

$$\text{core-dist}(p) = \begin{cases} \text{UNDEFINED} & \text{if } |N_\varepsilon(p)| < \text{MinPts} \\ \text{MinPts-th smallest distance to } N_\varepsilon(p) & \text{otherwise} \end{cases}$$

$$\text{reachability-dist}(o, p) = \begin{cases} \text{UNDEFINED} & \text{if } |N_\varepsilon(p)| < \text{MinPts} \\ \max(\text{core-dist}(p), \text{dist}(p, o)) & \text{otherwise} \end{cases}$$



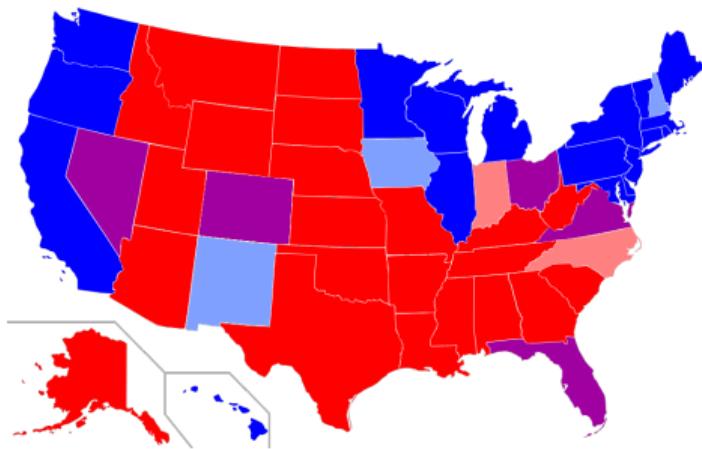
# OPTICS 1

```
1 function optics(X, eps, min_pts)
2     for each point x in X
3         x.reachability-distance = UNDEFINED
4     for each unprocessed point x in X
5         N = getNeighbors(x, eps)
6         mark x as processed
7         output x to the ordered list
8         if (core-distance(x, eps, min_pts) != UNDEFINED)
9             Seeds = empty priority queue
10            update(N, x, Seeds, eps, min_pts)
11            for each next z in Seeds
12                N' = getNeighbors(z, eps)
13                mark z as processed
14                output z to the ordered list
15                if (core-distance(z, eps, min_pts) != UNDEFINED)
16                    update(N', z, Seeds, eps, min_pts)
```

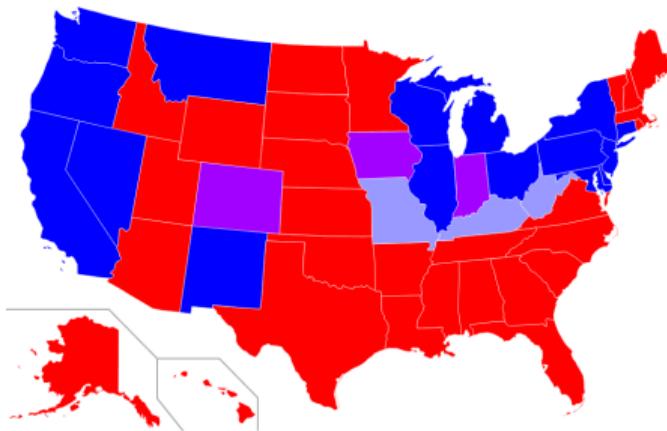
## OPTICS 2

```
1 function update(N, x, Seeds, eps, min_pts)
2     coredist = core-distance(x, eps, min_pts)
3     for each z in N
4         if (z is not processed)
5             new-reach-dist = max(coredist, dist(x, z))
6             # z is not in Seeds
7             if (z.reachability-distance == UNDEFINED)
8                 z.reachability-distance = new-reach-dist
9                 Seeds.insert(o, new-reach-dist)
10            # z in Seeds, check for improvement
11            else
12                if (new-reach-dist < z.reachability-distance)
13                    z.reachability-distance = new-reach-dist
14                    Seeds.move-up(z, new-reach-dist)
```

## Пример. Красные и синие штаты



(a) Состояние дел на 2012



(b) DBSCAN

## Качество кластеризации



## Качество кластеризации

Пусть дана обучающая выборка, для которой правильная кластеризация  $C$  известна. С помощью выбранного алгоритма получена кластеризация  $K$ . Проверить, насколько  $K$  совпадает с  $C$ .

- ▶ Adjusted Rand Index

$a$  – кол-во пар объектов, попавших в один кластер и в  $C$ , и в  $K$

$b$  – кол-во пар объектов, попавших в разные кластеры и в  $C$ , и в  $K$

$$RI = \frac{a + b}{C_2^N}, \quad ARI = \frac{RI - E_{rdm}[RI]}{\max(RI) - E_{rdm}[RI]}$$

- ▶ Mutual Information

$$MI = \sum_{c \in C} \sum_{k \in K} p(c, k) \log \frac{p(c, k)}{p(k)p(c)}$$

## Вопросы

