# Обзор методов анализа данных и искуственного интеллекта

Факультет Космических Исследований, МГУ

Москва

# Этапы развития области искусственного интеллекта

- Предыстория (мечты об ИИ и теоретические основы алгоритмистики);
- Появление самообучающихся машин конец 50х, начало 60х;
- Системы основанные на знаниях 70е;
- Первые нейронные сети (в основе алгоритм обратного распространения ошибки) начало 80х;
- Прорыв связанный с применением глубокого обучения в 2006г;

Причины бурного развития ИИ: **глубокое обучение**, **вычислительные мощности**, **большие наборы качественных данных**.

# ML tasks

Classification
Not Spam
Spam
Feature x2
Feature x1

Regression
Price
Feature x

Clustering
C1
C2
C3
Feature x2
Feature x1

# Математические обозначения

Пусть задано множество обьектов $X$ и множество допустимых ответов $Y$. Исследуемый процесс описыватся целевой функцией $y^* : X \to Y$, задано конечное множество обьектов $\{x_1, .., x_l\} \subset X$. Пары обьектов $X^l = (x_i, y_i)_{i=1}^l$ представляют собой обучающую выборку.

Основная задача состоит в восстановлении зависимости $y^*$, а именно в построенни **решающей функции** $a : X \to Y$, причем на всем множестве $X$.

Решаются следующие вопросы

- как задаются обьекты и какими могут быть ответы (представление данных);
- в какомы смысле $a(x)$ приближает $y(x)$ (выбор нормы);
- правило постороения $a(x)$;

Пространство признаков

- $\mathbb{Y} = \{0, 1\}$ бинарная классификация;
- $\mathbb{Y} = \{1, .., K\}$ - многоклассовая классификация;
- $\mathbb{Y} = \{0, 1\}^K$ - многоклассовая классификация с пересекающимися классами;

# Методика обучения

В качестве **модели алгоритма** рассмотрим функцию $g(x, \theta) : X \times \Theta \to Y$, где $\Theta$ множество допутимых значений параметра.

$$g(x, \theta) = \sum_{j=1}^{n} \theta_j f_j(x), \; Y = \mathbb{R}$$

$$g(x, \theta) = \text{sign} \sum_{j=1}^{n} \theta_j f_j(x), \; Y = \{-1, +1\}$$

$\theta$ sign **Метод обучения** - это отображение $\mu : (X \times Y)^l \to A$, которое произвольной коеченой выборке ставит некоторый алгоритм.

# Функционал качества

Функционал качества

$$Q(a, X^l) = \frac{1}{l}\sum_{i=1}^{l}\mathcal{L}(a, x_i).$$

Разновидности функции потерь (loss function):

- $\mathcal{L}(a, x_i) = [a(x) \neq y^*(x)]$ - индикатор ошибки,
- $\mathcal{L}(a, x_i) = |a(x) - y^*(x)|$ - отклонение от правильного ответа;
- $\mathcal{L}(a, x_i) = (a(x) - y^*(x))^2$ - квадратичная функция потерь;

Иными словами задача восстановлени регрессии есть не что иное, как метод наименьших квадратов

$$\mu(X^l) = \arg\min_{\theta}\sum_{i=1}^{l}(g(x_i, \theta) - y_i)^2,$$

# Кросс-проверка

Выборку $X^L = (x_i, y_i)_{i=1}^l$ разобьем $N$ различными способами.

Для каждого разбиение $n = 1..N$ построим алгоритм $a_n = \mu(X_n^l)$ и вычислим среднее значение ошибки $Q_n = Q(a_n, X_n^k)$. Среднее арифитическое невязок всех указанных алгоритмов называется **скользящим контролем**

$$CV(\mu, X^L) = \frac{1}{N} \sum_{n=1}^{l} Q(\mu(X_n^l), X_n^l).$$

# Скользящий контроль. Иллюстрация.

# Переобучение

# Пример применения

Рис. 1. Растительность Долины р. Гейзерной:
1 – каменноберезняк разнотравный, 2 – ивняк шеломайниковый, 3 – сообщества ольхового стланика, 4 –крупнотравно-шеломайниковые луговые сообщества, 5 – высокотравные луговые сообщества с преобладанием лабазника и бодяка, 6 – высокотравные луговые сообщества с преобладанием вол-

# Основные понятия и обозначения

**Задача классификаци** $X$, $Y$, $X^l = (x_i, y_i)_{i=1}^l$ - выборка обучающая.

# Различные виды норм

Рассмотрим функции описывающие различные нормы:

1. Евклидово $\rho(x, x_i) = \left(\sum_{j=1}^{n} w_k |x^j - x_i^j|^2)\right)^{1/2}$;
2. $L_p$-метрика $\rho(x, x_i) = \left(\sum_{j=1}^{n} w_k |x^j - x_i^j|^p)\right)^{p}$;
3. $L_\infty$-метрика $\rho(x, x_i) = \max_{j=1..n} |x_i^j - x^j|$;
4. $L_1$-метрика $\rho(x, x_i) = \sum_{j=1}^{n} |x_i^j - x^j|$;

NB: $w_1, .., w_2$ - веса признаков, их также можно настраивать.

# Обобщенный метрический классификатор

**Метрический алгоритм классификации** Пусть для заданной точки пространства обьектов $X$, его соседи из выборки $X^l$:

$$\rho(x, x^{(i)}) \leq \rho(x, x^{(i)}) \leq ... \leq \rho(x, x^{(l)}),$$

где $x^{(i)}$ - $i$й сосед обьекта $x$.

$$a(x; X^l) = \arg\max_{y \in Y} \sum_{i=1}^{l} [y^{(i)} = y] w(i, x),$$

так что можно определить функцию $_y(x)$ - оценку близости обьекта $x$ к классу $y$.

# Метод ближайшего соседа

Алгоритм ближайшего соседа: $w(i,u) = [i = 1]$, $a(u; X^l) = y_u^{(1)}$. Имеет ряд недостатков, такие как, неустойчивость, нельзя настроить под конкретные условия, необходимость хранить всю

выборку.

# Метод взвешенных ближайших соседей

Возьмем коэффициенты с весами $w(i, u) = [i \leq k]w_i$.
Возможны следующие случаи выбора весов:

- $w_i = \frac{k+1-i}{k}$ - линейной убывающие веса;
- $w_i = q^i$ - экспонетоциально убывающие веса $0 < q < 1$;

# kNN classification

k = 1

# kNN classification

k = 1

k = 5

# kNN ($k = 5$)

Train data, accuracy=0.87
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
4

Test data, accuracy=0.90
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
4

# kNN ($k = 20$)

Train data, accuracy=0.83
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
4

Test data, accuracy=0.90
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
4

# kNN ($k = 40$)

Train data, accuracy=0.81
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
4

Test data, accuracy=0.87
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
4

# Мат основы SVM

Пусть выборка $X^l = (x_i, y_i)_{i=1}^l$ линейно разделима:

$$\exists w, w_0 : \; M_i(w, w_0) = y_i(\langle w, x_i \rangle - w_0), \; i = 1, .., l.$$

Отнормируем вектор $\min_{i=1,..,l} M_i(w, w_0) = 1$
Раздедяющая полоса

$$\begin{aligned} &\{x : -1 \leq \langle w, x_i \rangle - w_0 \leq 1\}, \\ &\exists x_- : \langle w, x_- \rangle - w_0 = -1, \\ &\exists x_+ : \langle w, x_+ \rangle - w_0 = 1. \end{aligned}$$

Ширина полосы

$$\frac{\langle x_+ - x_-, w \rangle}{||w||} = \frac{2}{||w||} \to \max .$$

# SVM иллюстрация

Определим расстояние от произвольной точки $x_0 \in \mathbb{R}^d$

$$\rho(x_0, a) = \frac{|\langle w, x\rangle - w_0|}{||w||},$$

расстояние от гиперплоскости до ближайшего обьекта обучающей выборки равно

$$\min_{x \in X^l} \frac{|\langle w, x\rangle - w_0|}{||w||} = \frac{1}{||w||} \min_{x \in X} |\langle w, x\rangle - w_0| = \frac{1}{||w||}.$$

# Разделимая выборка

Существует прямая (гиперплоскость) которая является линейным разграничителем для двух классов (Hard margin support vector machin):

$$\begin{cases} \frac{1}{2}||w||^2 \to \min\limits_{w,w_0}, \\ y_i(\langle w, x\rangle - w_0) \geq 1,\ i = 1, ..., l. \end{cases}$$

На практике линейно разделимые выборки встречаются достаточно редко. Постановку необхоимо модифицировать, так чтобы система ограничений была совместна в любом случае.

# Неразделимая выборка

Хотя бы одно из ограничений в задаче нарушается

$$\exists x_i \in X : \; y_i(\langle w, x \rangle - w_0) < 1.$$

Введем штрафы

$$\forall x_i \in X : \; y_i(\langle w, x \rangle - w_0) \geq 1 - \xi_i, \; i = 1, .., l.$$

# SVM иллюстрация

# Линейная модель классификации

Пусть $f_j(x)$ $j = 1, .., N$ - числовые признаки, модель оптимизации однойслойного нейрона

$$a(x, w) = \sigma(\langle x, w \rangle) = \sigma\left(\sum_{i=1}^{N} w_j f_j(x) - w_0\right),$$

где $w_0, .., w_N$ -веса признаков.
функция $\mathrm{sign}(z)$, $\sigma(z) = \frac{1}{1+e^{-z}}$.

# Сила распределенных представлений

# Теоретическое обоснование границ применимости нейросетей

- двухслойная сеть позволяет реализовать произвольную булевую функцию,
- двухслойная сеть в $\mathbb{R}^n$ позволяет отделить произвольный выпуклый многогранник,
- трехслоная сеть $\mathbb{R}^n$ позволяет отделить произвольную многоранную область, не обязательно выпоклую, и даже не обязательно связную
- с помощю линейных операций и одной нелинейной функции активации $\sigma$ можно приблизить любую непрерывную функцию с любой желаемой точностью.

Вектор параметров модели $w = (w_{jh}, w_{hm}) \in R^{Hn+MH+M}$

**полносвязная сеть**- все со всеми.

# Алгоритм стохастического градиента

Оределим функционал ошибок

$$\mathcal{L}(w) = \frac{1}{l}\sum_{i=1}^{l}\mathcal{L}_i(w) \to \min_w,$$

- выбор случайного обьекта
- вычислить потерю $\mathcal{L}_i = \mathcal{L}_i(w)$
- градиентный шаг $w = w - \eta\mathcal{L}_i'(w)$
- оценить значение функции $Q = (1-\lambda)Q + \lambda\mathcal{L}_i$

# Методика обратного дифференцирования

Выходные значения сети на обьекте $x_i$:

$$a^m(x_i) = \sigma_m \left( \sum_{h=0}^{H} w_{hm} u^h(x_i) \right); \; u^h(x_i) = \sigma_h \left( \sum_{j=0}^{J} w_{jh} f^j(x_i) \right);$$

Без ограничения общности будем рассматривать среднеквадратичную функцию потерь

$$\mathcal{L}_i(w) = \frac{1}{2} \sum_{m=1}^{M} (a^m(x_i) - y_i^m)^2$$

Найти частные производные:

$$\frac{\partial \mathcal{L}_i(w)}{\partial a^m}, \; \frac{\partial \mathcal{L}_i(w)}{\partial u^h}.$$

# Метод обратного распространения ошибки (BackProp) 1

Градиент на конечном слое

$$\frac{\partial \mathcal{L}_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m,$$

тогда производная на выходном слое (функция потерь квадратичная)

$$\frac{\partial \mathcal{L}_i(w)}{\partial u^h} = \sum_{m=1}^{M} \frac{\partial \mathcal{L}_i(w)}{\partial a^m} \sigma'_m() w_{hm} = \sum_{m=1}^{M} \varepsilon_i^m \sigma'_m() w_{hm}$$

# Ускорение сходимости

- Более тщательный выбор начального приближения
- –Либо на случайной подвыборке ;
- –либо по случайному подмножеству входов;
- –либо из различных начальны приближений
- Выбивание из локальных минимумов (jogging of weights)
- Адаптивынй градиентный шаг (скорейший спуск)
- Метод сопряженных градиентов chuking -разбиение суммы $Q(w) = \sum_{l=1}^{l}$ на блоки (chunks)

# Динамическое наращивание сети

- обучение при заведомо небольшом числе нейронов H (мб. скользящий контроль?)
- после стабилизации $\mathcal{L}(w)$ - добавление нового нейрона и его оптимизация путем обучения –
- по случайной подвыборке $X' \subseteq X$ –
- по обьектам с небольшим значением потерь –
- по случайному поднможеству входов —
- для различных случайных приближениях
- итерации расчета градиента BackProp Время обучения с накоплением в 1.5-2 раза больше, в отличие от сети с фиксированном числом нейронов. Накопленная информация в нейронах сохраняется.

# Метод случайных отключений нейронов

- аппроксимируем простое голосование по $2^N$ сетям с общим набором из $N$ весов, но с различной архитектурой связей
- регуляризация: из всех сетей выбирем более устойчивую к утрате $pN$ нейронов, моделируя надежность мозга
- сокращаем переобучение, заставляя разные части сети решать одну и ту же исходную задачу вместо того, чтобы подстраивать их под компенсацию ошибок друг друга

# Dropout

# Функция сигмоиды

**Плюсы:**
масштабирование, простое дифференцирование
**Минусы:** нет центирования, вычислительная сложность, быстрое обнуление в крыльях

# Гиперболический тангенс

**Плюсы:**центрирован

**Минусы:** выч сложность, быстрое обнуление в крыльях

# ReLu

**Плюсы:**легко считается, дает быструю сходимость

**Минусы:** не центрирован, обнуление

# Lucky ReLu

**Плюсы:**легко считается, дает быструю сходимость, нет обнуления

# Различные функции активации

| **Название функции** | **Формула** $f(x)$ | **Производная** $f'(x)$ |
|---|---|---|
| Логистический сигмоид $\sigma$ | $\frac{1}{1+e^{-x}}$ | $f(x)\,(1-f(x))$ |
| Гиперболический тангенс tanh | $\frac{e^x-e^{-x}}{e^x+e^{-x}}$ | $1-f^2(x)$ |
| SoftSign | $\frac{x}{1+\lvert x\rvert}$ | $\frac{1}{(1+\lvert x\rvert)^2}$ |
| Ступенька (функция Хевисайда) | $\begin{cases} 0, & x<0 \\ 1, & x\geq 0 \end{cases}$ | $0$ |
| SoftPlus | $\log(1+e^x)$ | $\frac{1}{1+e^{-x}}$ |
| ReLU | $\begin{cases} 0, & x<0 \\ x, & x\geq 0 \end{cases}$ | $\begin{cases} 0, & x<0 \\ 1, & x\geq 0 \end{cases}$ |
| Leaky ReLU, Parameterized ReLU | $\begin{cases} ax, & x<0 \\ x, & x\geq 0 \end{cases}$ | $\begin{cases} a, & x<0 \\ 1, & x\geq 0 \end{cases}$ |
| ELU | $\begin{cases} \alpha\,(e^x-1)\,, & x<0 \\ x, & x\geq 0 \end{cases}$ | $\begin{cases} f(x)+\alpha, & x<0 \\ 1, & x\geq 0 \end{cases}$ |

# Обобщение

- Нейрон - позволяет решить задачу классификации или регресси
- Нейронная сеть - суперпозиция нейронов с нелинейной функцией активации. (Двух-трех слоев теоретически достаточно для решения любой задачи)
- BackProp - быстрое дифференцирования суперпозиции
- Методы по улучшению сходимости: адаптивный градиентный шаг, функция активации типа ReLu, регуляризация и DropOut, инициалазация нейронов, как отдельных алгоритмов

# АНСАМБИЛ И РЕШАЮЩИЕ ДЕРЕВЬЯ

Основная идея- классификация

# Пример решающего дерева

# Определение решающего дерева

- каждой внутренней вершине $v$ приписана функция $\beta_v : \mathbb{X} \to \{0, 1\}$
- каждой листовой вершине $v$ приписан прогноз $c_v \in Y$ (в случае с классификацие листу также может быть приписан вектор вероятностей)

Чаще всего использвется предикат, который сравнивает значение одного из признаков с порогом

$$\beta_v(x; j, t) = [x_j < t],$$

также возоможно применение многомерных предикатов.

# Построение деревьев

Для любой выборки можно постриоть алгоритм реш. дерева без ошибок, но в этом случае получим алгоритм, у которого не будет *обобщающей способности*.

- пусть задан функционал качества $Q(X, j, t)$.
- найдем лучшие разбиения $R_1(j, t) = \{x | x_j < t\}$ $R_2(j, t) = \{x | x_j \geq t\}$.
- определим оптимальные значения $j$, $t$, поставим в соотетствие вершине предикат $[x_j < t]$, так что выборка разбивается на два класса.
- в каждой вершине проверяется, выолнено ли условие останова.

В результате каждый лист содержит ответ, либо значение класса, либо вероятность.

# Построение деревьев

Метод построения решающих деревьев определяется

1. видом предикатов в вершинах;
2. функционалом качества;
3. критерием останова;
4. методом обработки пропущенных значений
5. методом стрижки (pruning)

# Регрессия иллюстрация

# Классификация иллюстрация

# Вводные понятия

Примеры ансамблей, усреднение нескольких алгоритмов:

$$a(x) = \frac{1}{N}(b_1(x) + ... + b_N(x)),$$

голосование большинства

$$a(x) = mode(b_1(x), .., b_N(x)),$$

в общем случае ансамбль записывается как

$$a(x) = b(b_1(x), .., b_N(x)).$$

Одна из целей сделать ансамбль достаточно разнообразным, тогда ошибка отдельных алгоритмов на отдельных обьектах будет компенсироваться корректной работой других алгоритмов.

# Расширение возможностей простых алгоритмов

# Бэгинг

**Бэггинг - bootstrap aggregarion**

Пусть задан метод обучения $\mu(X)$, построим метод $\tilde{\mu}(X)$, который генерирует подвыборку $\tilde{X}$: $\tilde{\mu}(X) = \mu(X)$, итоговый алгоритм примет вид

$$a_N(x) = \frac{1}{N}\sum_{j=1}^{N} b_j(x) = \frac{1}{N}\sum_{j=1}^{N} \tilde{\mu}(X)(x).$$

Имеем невзвешенное ансамиблирование. Смещение

$$\mathbb{E}_x[(\frac{1}{N}\sum_{j=1}^{N}\mathbb{E}_X[\tilde{\mu}(X)] - \mathbb{E}[y|x])^2] =$$

$$= \mathbb{E}_x[(\mathbb{E}_X[\tilde{\mu}(X)] - \mathbb{E}[y|x])^2].$$

# Сравнение бэгинга деревьев и kNN

Одно дерево

Бэгинг 100 деревьев

Ближайший сосед

Бэгинг 100 ближайших соседей

# Случайные леса

Бэггинг позволяет обьеденить несмещенные, но чувствительные к обучающей выборке алгоритмы в несмещенную композицию с низкой низкой дисперсией.

- n=1,..,N,
- генерируем выборку $\tilde{X}_n$ при помощи бутсрэпа,
- построить решающее дерево $b_n(x)$ по выборке $\tilde{X}_n$,
- * деревья строятся, пока в каждом листе окажется не более $n_{min}$ обьектов,
- * при каждом разбиении выбрирается $m$ начальных признаков из $p$, оптимальное разделение ищется только среди них,
- возвращается композиция $a(x) = \frac{1}{N}\sum\limits_{j=1}^{N} b_j(x)$.

# Out-of-bag

При том, что каждое дерево обучается на части обьектов, можно построить фунционал

$$OOB = \sum_{i=1}^{l} L\left(y_i, \frac{1}{\sum_{n=1}^{N}[x_i \notin X_n]} \sum_{n=1}^{N}[x_i \notin X_n] b_n(x_i)\right).$$

Можно показать, что по мере увеличения числа деревьев, данная оценка стремится к leave-one-out оценке.

# Пример применения решающих деревьев

# Стекинг

Придуман Д. Волпертом в 1992г.
**Блендинг** - простая схема стекинга. Обучающая выборка делится на две части. Затем получают их ответы на второй части и на тестовой выборке. Ответы каждого алгоритма можно рассматривать, как **новый признак** (метапризнак). На метапризнаках второй части настраивают метаалгоритм. Затем запускают его на метапризнаках теста и получают ответ.

Разные алгоритмы
Одинаковые данные
K-NN
Один решающий
Decision Tree
Ответ
SVM
Стекинг

обучение
X
y
тест
X₂
обучение алгоритмов
обучение мета-алгоритма
train
обучаем несколько алгоритмов
их ответы на отложенном обучении
их ответы на тесте
обучаем мета-алгоритм
train
запускаем его на новой тестовой выборке
test
result

1й блендинг
train
обучение
тест
2й блендинг
train
обучение
тест
train
обучение метаалгоритма
тест метаалгоритмов
result

# Классический стекинг

Выборку разбивают на части (**фолды**), затем последовательно перебирая фолды обучают базовые алгоритмы на всех фолдах, кроме одного, а на оставшемся получают ответы базовых алгоритмов и трактуют их как значения соответствующих признаков на этом фолде. Для получения метапризнаков обьектов тестовой выборки базовые алгоритмы обучают на всей обучающей выборке и берут их ответы на тестовой.

# План лекции

- визуализация, метод главных компонент (PCA), t-SNE;
- алгоритм k- средних;
- агломеративная кластеризация.

# Набор данных рукописных чисел PCA

# Набор данных рукописных чисел t-SNE

# Мат основа метода t-SNE

**Постановка задачи:** Есть набор данных с точками, описываемые многомерное переменной с размерностью пространства существенно больше трех. Необходимо получить новоую переменную, существующую в двумерном или трехмерном пространстве, которая бы в максимальной степени сохранила структуру и закономерность в исходных данных. Сопоставим каждой дистанции между точками $x_i$ исходного многомерного пространства $X$ ее веротностный аналог:

$$p_{j|i} = \frac{\exp(-||x_i - x_j||/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||/2\sigma_i^2)},$$

в проекционом пространстве размерности 2 или 3 точкам $x_i$, $x_j$ сопоставим $y_i$, $y_j$:

$$q_{j|i} = \frac{\exp(-||y_i - y_j||)}{\sum_{k \neq i} \exp(-||y_i - y_k||)}.$$

# Проблема неоднозначности задачи кластеризации

- нет точной постановки задачи кластеризации;
- существует много критериев качетва кластеризации (близко- различные)
- существует много эвристических методов кластеризации;
- число кластеров $|Y|$ как правило не извстно заранее;
- результат кластеризации существенно зависит от метрики $\rho$, которую эксперт задает субьективно.

Какие цели кластеризации??

# Алгоритм K средних (K-means)

K-means итеративно минимизирует среднее внутрикластерное расстояние.

1. Обьект присваивается к тому кластеру, центр которого ближе
2. центр кластера перемещается в центр среднего арифметического координато векторов.

# K-means, выбор центров

В зависимости от начального положения цетров - разные результаты и время сходимости. Варианты выбора центов: случайно, подальше друг от друга, для двух кластеров, выбор начального приближения:

- первый центр выбираем случайно из равномерного распределения на выборке;
- каждый следующий центр выбирается из оставшихся точек так, что вероятность выбрать каждую точку была пропорциональна квадрату расстояния от нее до ближайшего центра.

# Функционалы качества кластеризации

Для **метрических пространств** можно ввести следующие функционалы:

Среднее внутрикластерное расстояние

$$F_0 = \frac{\sum_{i<j}[y_i = y_j]\rho(x_i, x_j)}{\sum_{i<j}[y_i = y_j]} \to \min,$$

Среднее межкластерное расстояние:

$$F_1 = \frac{\sum_{i<j}[y_i \neq y_j]\rho(x_i, x_j)}{\sum_{i<j}[y_i \neq y_j]} \to \max,$$

Также можно рассмотреть метрику

$$F_0/F_1 \to 0.$$

# Функционалы качества кластеризации

Для **линейных пространств** можно посчитать найти центры кластеров $\mu$, $y \in Y$, тогда для внутрикластерных расстояний:

$$\Phi_0 = \sum_{y \in Y} \frac{1}{|K_y|} \sum_{i: y_i = y} \rho^2(x_i, \mu_y) \to \min,$$

аналоично максимизируем межкалстерные расстояния:

$$\Phi_0 = \sum_{y \in Y} \rho^2(\mu_y, \mu) \to \max.$$

# Как работает K Means

# Как работает K Means

# Как работает K Means

# Как работает K Means

# Как работает K Means

# Как работает K Means

# Иерархическая кластеризация

Стироится не одно разбиение выборки, а система **вложенных** разбиений. Рассмотрим аглемеративные методы, или **восходящие** алгоритмы, в которых обьекты обьединяются во все более и более крупные кластеры.

# Формула Ланса-Уильямса

На каждой итерации образуется новый кластер $W = U \cup V$, расстояние от нового кластера $W$ до любого другого кластера $S$ вычисляется по расстояниям $R(U, V)$:

$$R(U \cup V, S) = \alpha_U R(U, S) + \alpha_V R(V, S) + \\ + \beta R(U, V) + \gamma |R(U, S) - R(V, S)|$$

# Способы вычисления расстояния между кластерами 1

1. Расстояние ближнего соседа

$$R^n(W,S) = \min_{w \in W, s \in S} \rho(w,s),$$

$$\alpha_U = \alpha_V,\ \beta = 0, \gamma = -0.5,$$

2. расстояние дальнего соседа

$$R^f(W,S) = \min_{w \in W, s \in S} \rho(w,s),$$

$$\alpha_U = \alpha_V,\ \beta = 0, \gamma = -0.5,$$

3. среднее расстояние

$$R^f(W,S) = \frac{1}{|W||S|} \sum_{w \in W} \sum_{s \in S} \rho(w,s),$$

$$\alpha_U = \frac{|U|}{|W|}, \alpha_V = \frac{|V|}{|W|},\ \beta = \gamma = 0,$$

# Расстояния между кластерами

# Агломеративная кластеризация Ланса - Уильямса

1. Инициализация множества кластеров

$$C_t = \{x_1, .., x_l\},$$

2. **Forall**
3. найти в $C_{t-1}$ два ближайшиз кластера

$$(U, V) = \arg\min_{U \neq V} R(U, V),$$

$$R_t = R(U, V)$$

4. изьять кластеры и добавить слитые кластеры $W = U \cup V$

$$C_t = C_{t-1} \cup \{W\} \setminus \{U, V\}$$

5. **Forall** вычислить расстояние $R(W, S)$ по формуле Ланса -Уильямса.

# Замечания к иерархическому методу

- рекомендуется пользоваться расстоянием Уорда $R^u$;
- по практике строят несколько разбиений и выбирают лучшую визуально по дендрограмме;
- определение числа кластеров - по максимуму $|R_{t+1} - R_t|$, тогда за наилучшее множество кластеров можно взять $C_t$.

# Агломеративная кластеризация

# Агломеративная кластеризация

# Агломеративная кластеризация

# Агломеративная кластеризация

# Агломеративная кластеризация

# Агломеративная кластеризация

# Агломеративная кластеризация

# Агломеративная кластеризация

# Агломеративная кластеризация

# Агломеративная кластеризация

# Агломеративная кластеризация

# Агломеративная кластеризация

# Агломеративная кластеризация

# Дендрограмма

# Дендрограмма

1 3 2 4 9 10 11 12 6 8 7 5

# Дендрограмма

# Дендрограмма

# Дендрограмма

# Дендрограмма

# Дендрограмма

# Дендрограмма

# Дендрограмма

# Дендрограмма

# Дендрограмма

# Дендрограмма

# Дендрограмма

# Дендрограмма

# Дендрограмма

# Пример: кластеризация писем

# Пример: расстояние между кластерами

- На подвыборке из 100 писем

# Пример: расстояние между кластерами

- На подвыборке из 1000 писем

# Пример: расстояние между кластерами

- На подвыборке из 10000 писем