Министерство образования и науки Российской Федерации

Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

И.И. Кручинин
(к.т.н. доцент)

лекция «Методы кластеризации» по курсу «Введение в машинное обучение»

Калуга - 2018

Краткое содержание:

1. Алгоритмы кластеризации
2. Типы кластерных структур
3. Эвристические графовые алгоритмы, алгоритм FOREL
4. Статистические алгоритмы. EM-алгоритм, метод К-средних
5. Кластеризация с частичным обучением
6. Иерархическая кластеризация Ланса – Уильямса
7. Нейронные структуры, основанные на обучении без учителя
8. Карты Кохонена, векторное квантование, визуализация многомерных данных
9. Алгоритмы последовательного и пакетного обучения

Кластеризация и визуализация

Во многих прикладных задачах измерять степень сходства объектов существенно проще, чем формировать признаковые описания. Например, две строки с описанием молекул ДНК или протеинов гораздо легче сравнить непосредственно друг с другом, чем преобразовывать каждый из них в вектор признаков, и затем сравнивать эти векторы. То же самое можно сказать про тексты, временные ряды или растровые изображения.

Задача классификации объектов на основе их сходства друг с другом, когда принадлежность обучающих объектов каким-либо классам не задаётся, называется задачей кластеризации.

Алгоритмы кластеризации

Задача *кластеризации* (или обучения без учителя) заключается в следующем. Имеется обучающая выборка $X^\ell = \{x_1, \ldots, x_\ell\} \subset X$ и функция расстояния между объектами $\rho(x, x')$. Требуется разбить выборку на непересекающиеся подмножества, называемые *кластерами*, так, чтобы каждый кластер состоял из объектов, близких по метрике ρ, а объекты разных кластеров существенно отличались. При этом каждому объекту $x_i \in X^\ell$ приписывается метка (номер) кластера y_i.

Алгоритм кластеризации — это функция $a\colon X \to Y$, которая любому объекту $x \in X$ ставит в соответствие метку кластера $y \in Y$. Множество меток Y в некоторых случаях известно заранее, однако чаще ставится задача определить оптимальное число кластеров, с точки зрения того или иного критерия качества кластеризации.

Решение задачи кластеризации принципиально неоднозначно, и тому есть несколько причин. Во-первых, не существует однозначно наилучшего критерия

качества кластеризации. Известен целый ряд достаточно разумных критериев, а также ряд алгоритмов, не имеющих чётко выраженного критерия, но осуществляющих достаточно разумную кластеризацию «по построению». Все они могут давать разные результаты. Во-вторых, число кластеров, как правило, неизвестно заранее и устанавливается в соответствии с некоторым субъективным критерием. В-третьих, результат кластеризации существенно зависит от метрики ρ, выбор которой, как правило, также субъективен и определяется экспертом.

Кластеризация (обучение без учителя) отличается от классификации (обучения с учителем) тем, что метки исходных объектов y_iизначально не заданы, и даже может быть неизвестно само множество Y. В этом смысле задача кластеризации ещё в большей степени некорректно поставленная, чем задача классификации.

Цели кластеризации могут быть различными в зависимости от особенностей конкретной прикладной задачи:

- Упростить дальнейшую обработку данных, разбить множество X^ℓ на группы схожих объектов чтобы работать с каждой группой в отдельности (задачи классификации, регрессии, прогнозирования).
- Сократить объём 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).

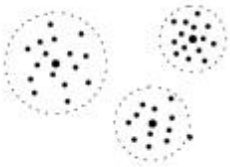
В первом случае число кластеров стараются сделать поменьше. Во втором случае важнее обеспечить высокую степень сходства объектов внутри каждого кластера, а кластеров может быть сколько угодно. В третьем случае наибольший интерес представляют отдельные объекты, не вписывающиеся ни в один из кластеров.

Во всех этих случаях может применяться иерархическая кластеризация, когда крупные кластеры дробятся на более мелкие, те в свою очередь дробятся ещё мельче, и т.д. Такие задачи называются задачами таксономии (taxonomy). Результатом таксономии является не простое разбиение множества объектов на кластеры, а древообразная иерархическая структура. Вместо номера кластера объект характеризуется перечислением всех кластеров, которым он принадлежит, от крупного к мелкому. Классическим примером таксономии на основе сходства является систематизация живых существ, предложенная Карлом Линнеем в середине XVIII века. В современном представлении биологическая иерархия имеет около 30 уровней, 7 из них считаются основными: царство, тип, класс, отряд, семейство, род, вид. Таксономии строятся во многих областях знания, чтобы упорядочить информацию о большом количестве объектов. Мы будем рассматривать алгоритмы иерархической кластеризации, позволяющие автоматизировать процесс построения таксономий.

Типы кластерных структур. Попытаемся составить реестр различных типов кластерных структур, которые могут возникать в практических задачах.

Сгущения: внутрикластерные расстояния, как правило, меньше межкластерных.

Ленты: для любого объекта найдётся близкий к нему объект того же кластера, в то же время существуют объекты одного кластера, которые не являются близкими.



Кластеры с центром: в каждом кластере найдётся объект, такой, что почти все объекты кластера лежат внутри шара с центром в этом объекте.

Кластеры могут соединяться перемычками, что затрудняет работу многих алгоритмов кластеризации.

Кластеры могут накладываться на разреженный фон из редких нетипичных объектов.

Кластеры могут перекрываться.

Кластеры могут образовываться не по принципу сходства, а по каким-либо иным, заранее неизвестным, свойствам объектов. Стандартные методы кластеризации здесь бессильны.

Кластеры могут вообще отсутствовать. В этом случае надо применять не кластеризацию, а иные методы анализа данных.

Различные алгоритмы кластеризации могут быть более или менее успешны в этих ситуациях. Простые алгоритмы, как правило, узко специализированы и дают адекватные результаты только в одной-двух ситуациях. Более сложные алгоритмы, такие как FOREL или агломеративная процедура Ланса-Вильямса, справляются с несколькими типами ситуаций. Однако создание алгоритма, успешно работающего во всех ситуациях без исключения, представляется трудной и едва ли разрешимой задачей.

Эвристические графовые алгоритмы

Обширный класс алгоритмов кластеризации основан на представлении выборки в виде графа. Вершинам графа соответствуют объекты выборки, а рёбрам — попарные расстояния между объектами $\rho_{ij} = \rho(x_i, x_j)$.

Достоинством графовых алгоритмов кластеризации является наглядность, относительная простота реализации, возможность вносить различные усовершенствования, опираясь на простые геометрические соображения.

Алгоритм выделения связных компонент. Задаётся параметр R и в графе удаляются все рёбра (i,j), для которых $\rho_{ij} > R$. Соединёнными остаются только наиболее близкие пары объектов. Идея алгоритма заключается в том, чтобы подобрать такое значение $R \in [\min\rho_{ij}, \max\rho_{ij}]$, при котором граф развалится на несколько связных компонент. Найденные связные компоненты — и есть кластеры.

Связной компонентой графа называется подмножество его вершин, в котором любые две вершины можно соединить путём, целиком лежащим в этом подмножестве. Для поиска связных компонент можно использовать стандартные алгоритмы поиска в ширину (алгоритм Дейкстры) или поиска в глубину.

Алгоритм 7.1. Алгоритм кратчайшего незамкнутого пути (КНП)

1: Найти пару точек (i, j) с наименьшим ρ_{ij} и соединить их ребром;
2: **пока** в выборке остаются изолированные точки
3: найти изолированную точку, ближайшую к некоторой неизолированной;
4: соединить эти две точки ребром;
5: удалить $K - 1$ самых длинных рёбер;

Для подбора параметра R обычно рекомендуется построить гистограмму распределения попарных расстояний ρ_{ij}. В задачах с выраженной кластерной структурой эта гистограмма имеет два чётких пика: зона небольших внутриклассовых расстояний и зона больших межклассовых расстояний. Параметр R задаётся как расстояние, соответствующее точке минимума между этими пиками. Отметим два недостатка этого алгоритма.

- Ограниченная применимость. Алгоритм выделения связных компонент наибо-лее подходит для выделения кластеров типа сгущений или лент. Наличие разреженного фона или «узких перемычек» между кластерами приводит к неадекватной кластеризации.

- Плохая управляемость числом кластеров. Для многих приложений удобнее задавать не параметр R, а число кластеров или некоторый порог «чёткости кластеризации». Управлять числом кластеров с помощью параметра Rдовольно затруднительно. Приходится многократно решать задачу при разных R, что отрицательно сказывается на временных затратах.

Алгоритм кратчайшего незамкнутого пути строит граф из $\ell-1$ рёбер так, чтобы они соединяли все ℓ точек и обладали минимальной суммарной длиной. Такой граф называется кратчайшим незамкнутым путём (КНП), минимальным покрывающим деревом или каркасом. Доказано, что этот граф строится с помощью несложной процедуры, соответствующей шагам 1–4 Алгоритма 7.1. На шаге 5 удаляются $K-1$ самых длинных рёбер, и связный граф распадается на Kкластеров.

В отличие от предыдущего алгоритма, число кластеров Kзадаётся как входной параметр. Его можно также определять графически, если упорядочить все расстояния, образующие каркас, в порядке убывания и отложить их на графике. Резкий скачок вниз где-то на начальном (левом) участке графика покажет количество наиболее чётко выделяемых кластеров.

Этот алгоритм, как и предыдущий, очень прост и также имеет ограниченную применимость. Наличие разреженного фона или перемычек приводит к неадекватной кластеризации. Другим недостатком КНП является высокая трудоёмкость — для построения кратчайшего незамкнутого пути требуется $O(\ell^3)$ операций.

Алгоритм FOREL (ФОРмальный ЭЛемент) предложен Загоруйко и Ёлкиной в 1967 году при решении одной прикладной задачи в области палеонтологии. В основе лежит следующая базовая процедура.

Пусть задана некоторая точка $x_0 \in X$и параметр R. Выделяются все точки выборки $x_i \in X^\ell$, попадающие внутрь сферы $\rho(x_i,x_0)$ 6 R, и точка x_0 переносится в центр тяжести выделенных точек. Эта процедура повторяется до тех пор, пока состав выделенных точек, а значит и положение центра, не перестанет меняться. Доказано, что эта процедура сходится за конечное число шагов. При этом сфера перемещается в место локального сгущения точек. Центр сферы x_0 в общем случае не является объектом выборки, потому и называется формальным элементом.

Для вычисления центра необходимо, чтобы множество объектов X было не только метрическим, но и линейным векторным пространством. Это требование естественным образом выполняется, когда объекты описываются числовыми признаками. Однако существуют задачи, в которых изначально задана только метрика, а сложение и умножение на число не определены на X. Тогда в качестве центра сферы можно взять тот объект обучающей выборки, для которого среднее расстояние до других объектов кластера минимально. Соответственно, шаг 6 заменяется на

$$x_0 := \arg\min_{x \in K_0} \sum_{x' \in K_0} \rho(x, x').$$

При этом заметно увеличивается трудоёмкость алгоритма. Если в линейном пространстве для вычисления центра требуется $O(к)$ операций, то в метрическом — $O(к^2)$, где к — чисто точек в кластере. Алгоритм можно несколько ускорить, если заметить, что пересчёт центра при добавлении или удалении отдельной точки кластера требует лишь $O(к)$ операций, а в линейном пространстве — $O(1)$.

Различные варианты алгоритма FOREL отличаются способами объединения сфер в кластеры, способами варьирования параметра R, способами выбора начального

приближения для точек x_0. В Алгоритме 7.2 представлен один из вариантов, в котором сферы строятся последовательно. На шаге 9 к центрам этих сфер применяется алгоритм КНП. С одной стороны, это решает проблему низкой эффективности КНП, так как сфер гораздо меньше, чем исходных объектов. С другой стороны, мы получаем более тонкую, двухуровневую, структуру кластеров: каждый кластер верхнего уровня распадается на более мелкие подкластеры нижнего уровня.

Другое преимущество этого алгоритма — возможность описывать кластеры произвольной геометрической формы. Варьируя параметр R, можно получать кластеризации различной степени детальности. Если кластеры близки по форме к шарам, можно сделать Rдостаточно большим. Для описания кластеров более сложной формы следует уменьшать R.

Алгоритм 7.2 довольно чувствителен к выбору начального положения точки x_0 для каждого нового кластера. Для устранения этого недостатка предлагается генерировать несколько (порядка 10..20) кластеризаций. Поскольку начальное положение центров выбирается случайным образом, эти кластеризации будут довольно сильно отличаться. Окончательно выбирается та кластеризация, которая доставляет наилучшее значение заданному функционалу качества.

Различные виды функционалов качества рассматриваются далее.

Функционалы качества кластеризации

Задачу кластеризации можно ставить как задачу дискретной оптимизации: необходимо так приписать номера кластеров y_iобъектам x_i, чтобы значение выбранного функционала качества приняло наилучшее значение. Существуетмного

Алгоритм 7.2.Алгоритм FOREL

1: Инициализировать множество некластеризованных точек: $U := X^\ell$;
2: **пока** в выборке есть некластеризованные точки, $U \neq \varnothing$:
3: взять произвольную точку $x_0 \in U$ случайным образом;
4: **повторять**
5: образовать кластер — сферу с центром в x_0 и радиусом R: $K_0 := \{x_i \in U \mid \rho(x_i, x_0) \leqslant R\}$;
6: поместить центр сферы в центр масс кластера: $x_0 := \frac{1}{|K_0|} \sum_{x_i \in K_0} x_i$;
7: **пока** центр x_0 не стабилизируется;
8: пометить все точки K_0 как кластеризованные: $U := U \setminus K_0$;
9: применить алгоритм КНП к множеству центров всех найденных кластеров;
10: каждый объект $x_i \in X^\ell$ приписать кластеру с ближайшим центром;

разновидностей функционалов качества кластеризации, но нет «самого правильного» функционала. По сути дела, каждый метод кластеризации можно рассматривать как точный или приближённый алгоритм поиска оптимума некоторого функционала.

Среднее внутрикластерное расстояние должно быть как можно меньше:

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

Среднее межкластерное расстояние должно быть как можно больше:

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

.

Если алгоритм кластеризации вычисляет центры кластеров μ_y, $y \in Y$, то можно определить функционалы, вычислительно более эффективные.

Сумма средних внутрикластерных расстояний должна быть как можно меньше:

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

,

где $K_y = \{x_i \in X^\ell \mid y_i = y\}$ — кластер с номером y. В этой формуле можно было бы взять не квадраты расстояний, а сами расстояния. Однако, если ρ— евклидова метрика, то внутренняя сумма в Φ_0 приобретает физический смысл момента инерции кластера K_yотносительно его центра масс, если рассматривать кластер как материальное тело, состоящее из $|K_y|$ точек одинаковой массы.

Сумма межкластерных расстояний должна быть как можно больше:

$$\Phi_1 = \sum_{y \in Y} \rho^2(\mu_y, \mu) \to \max,$$

где μ — центр масс всей выборки.

На практике вычисляют отношение пары функционалов, чтобы учесть как межкластерные, так и внутрикластерные расстояния:

$$F_0/F_1 \to \min, \quad \text{либо} \quad \Phi_0/\Phi_1 \to \min.$$

Алгоритм 7.3. Кластеризация с помощью EM-алгоритма

начальное приближение для всех кластеров $y \in Y$:
$w_y := 1/|Y|$;
$\mu_y :=$ случайный объект выборки;
$\sigma_{yj}^2 := \frac{1}{\ell|Y|} \sum_{i=1}^{\ell} (f_j(x_i) - \mu_{yj})^2, \ j = 1, \ldots, n$;
повторять
  E-шаг (expectation):
  $g_{iy} := \dfrac{w_y p_y(x_i)}{\sum_{z \in Y} w_z p_z(x_i)}, \ y \in Y, \ i = 1, \ldots, \ell$;
  M-шаг (maximization):
  $w_y := \dfrac{1}{\ell} \sum_{i=1}^{\ell} g_{iy}, \ y \in Y$;
  $\mu_{yj} := \dfrac{1}{\ell w_y} \sum_{i=1}^{\ell} g_{iy} f_j(x_i), \ y \in Y, \ j = 1, \ldots, n$;
  $\sigma_{yj}^2 := \dfrac{1}{\ell w_y} \sum_{i=1}^{\ell} g_{iy} (f_j(x_i) - \mu_{yj})^2, \ y \in Y, \ j = 1, \ldots, n$;
  Отнести объекты к кластерам по байесовскому решающему правилу:
  $y_i := \arg\max_{y \in Y} g_{iy}, \ i = 1, \ldots, \ell$;
пока y_i не перестанут изменяться;

Статистические алгоритмы

Статистические алгоритмы основаны на предположении, что кластеры неплохо описываются некоторым семейством вероятностных распределений. Тогда задача кластеризации сводится к разделению смеси распределений по конечной выборке.

Снова EM-алгоритм. Напомним основные гипотезы байесовского подхода к разделению смесей вероятностных распределений

Гипотеза 7.1 (о вероятностной природе данных). *Объекты выборки X^ℓ появляются случайно и независимо согласно вероятностному распределению, представляющему собой смесь распределений*

$$p(x) = \sum_{y \in Y} w_y p_y(x), \qquad \sum_{y \in Y} w_y = 1,$$

где $p_y(x)$ — функция плотности распределения кластера y, w_y — неизвестная априорная вероятность появления объектов из кластера y.

Конкретизируя вид распределений $p_y(x)$, чаще всего берут сферические гауссовские плотности. Мы будем считать, что кластеры похожи скорее не на шары, а на эллипсоиды, оси которых направлены вдоль осей координат. Преимущество эллиптических гауссианов в том, что они обходят проблему выбора нормировки признаков. Нормировать можно немного по-разному, причём результат кластеризации существенно зависит от нормировки. Пока не произведена кластеризация, трудно понять, какая нормировка лучше. При использовании эллиптических гауссианов оптимальная нормировка подбирается самим алгоритмом кластеризации, индивидуально для каждого кластера.

EM-алгоритм

Дано. Известно распределение $P(\mathbf{X}, \mathbf{Z}|\theta)$, где $\mathbf{x}$ – наблюдаемые переменные, а $\mathbf{z}$ – скрытые.
Найти. θ, максимизирующее $P(\mathbf{X}|\theta)$.

E вычислить $P(\mathbf{Z}|\mathbf{X}, \theta^{old})$ при фиксированном θ^{old}
M вычислить $\theta^{new} = \arg\max_\theta \mathcal{Q}(\theta, \theta^{old})$, где

$$\mathcal{Q}(\theta, \theta^{old}) = E_{\mathbf{Z}}[\ln p(\mathbf{X}, \mathbf{Z}|\theta)] = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{old}) \ln p(\mathbf{X}, \mathbf{Z}|\theta))$$

Улучшение: ввести априорное распределение $p(\theta)$

Гипотеза 7.2 (о пространстве объектов и форме кластеров). *Каждый объект x из $X = \mathbb{R}^n$ описывается n числовыми признаками: $x \equiv (f_1(x), \ldots, f_n(x))$. Каждый кластер $y \in Y$ описывается n-мерной гауссовской плотностью $p_y(x)$ с центром $\mu_y = (\mu_{y1}, \ldots, \mu_{yn})$ и диагональной матрицей ковариаций $\Sigma_y = \operatorname{diag}(\sigma_{y1}^2, \ldots, \sigma_{yn}^2)$:*

$$p_y(x) = (2\pi)^{-\frac{n}{2}} (\sigma_{y1} \cdots \sigma_{yn})^{-1} \exp\left(-\tfrac{1}{2}\rho_y^2(x, \mu_y)\right),$$

где $\rho_y^2(x, x') = \sum_{j=1}^{n} \sigma_{yj}^{-2} |f_j(x) - f_j(x')|^2$ — взвешенное евклидово расстояние с весами σ_{yj}^{-2}.

При этих предположениях задача кластеризации совпадает с задачей разделения смеси вероятностных распределений, и для её решения можно применить ЕМалгоритм. Для оценивания параметров кластеров воспользуемся формулами, полученными в Теореме 2.7 как раз для случая эллиптических гауссианов. Реализация этой идеи представлена в Алгоритме 7.3.

Expectation Maximization (!)

E Expectation: при фиксированных μ_k, Σ_k, π_k

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \Sigma_j)}$$

M Maximization: при фиксированных $\gamma(z_{nk})$

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}), \quad \mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N}$$

S Остановиться при достижении сходимости

Напомним, что ЕМ-алгоритм заключается в итерационном повторении двух шагов. На Е-шаге по формуле Байеса вычисляются скрытые переменные g_{iy}. Значение g_{iy} равно апостериорной вероятности того, что объект $x_i \in X^\ell$ принадлежит кластеру $y \in Y$. На М-шаге уточняются параметры каждого кластера (μ_y, Σ_y), при этом существенно используются скрытые переменные g_{iy}.

В Алгоритме 7.3 для простоты предполагается, что число кластеров известно заранее. Однако в большинстве практических случаев его лучше определять автоматически, как это было сделано в Алгоритме 2.3.

Метод *k*-средних, представленный в Алгоритме 7.4, является упрощением ЕМ-алгоритма. Главное отличие в том, что в ЕМ-алгоритме каждый объект x_i распределяется по всем кластерам с вероятностями $g_{iy} = \mathrm{P}\{y_i = y\}$. В алгоритме *k*-средних (*k*-means) каждый объект жёстко приписывается только к одному кластеру.

K-means

Пусть $\Sigma_k = \epsilon I$, тогда

$$p(\mathbf{x}|\mu_k, \Sigma_k) = \frac{1}{\sqrt{2\pi\epsilon}} \exp\left(-\frac{1}{2\epsilon}\|\mathbf{x} - \mu_\mathbf{k}\|^2\right)$$

Рассмотрим стремление $\epsilon \to 0$

$$\gamma(z_{nk}) = \frac{\pi_k \exp\left(-\frac{1}{2\epsilon}\|\mathbf{x}_n - \mu_\mathbf{k}\|^2\right)}{\sum_j \pi_j \exp\left(-\frac{1}{2\epsilon}\|\mathbf{x}_n - \mu_\mathbf{j}\|^2\right)} \to r_{nk} = \begin{cases} 1, \text{ для } k = \arg\min_j \|\mathbf{x}_n - \mu_\mathbf{j}\|^2 \\ 0, \text{ иначе} \end{cases}$$

Функция правдоподобия

$$E_\mathbf{Z}[\ln p(\mathbf{X}, \mathbf{Z}|\mu, \Sigma, \pi)] \to -\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\mathbf{x}_n - \mu_k\|^2 + const$$

Вектор средних

$$\mu_k = \frac{\sum_n r_{nk}\mathbf{x_n}}{\sum_n r_{nk}}$$

Второе отличие в том, что в *k*-means форма кластеров не настраивается. Однако это отличие не столь принципиально. Можно предложить упрощённый вариант EM, в котором форма кластеров также не будет настраиваться — для этого достаточно взять сферические гауссианы с ковариационными матрицами $\Sigma_y = \sigma_y I_n$. С другой стороны, возможен и обобщённый вариант *k*-means, в котором будут определяться размеры кластеров вдоль координатных осей. Для этого в Алгоритме 7.3 достаточно убрать шаг 5 и заменить E-шаг жёстким приписыванием объектов кластерам:

$$y_i := \arg\min_{y \in Y} \rho_y(x_i, \mu_y), \quad j = 1, \ldots, n;$$
$$g_{iy} := [y_i = y], \quad j = 1, \ldots, n, \quad y \in Y.$$

Таким образом, EM и *k*-means довольно плавно «перетекают» друг в друга, позволяя строить различные «промежуточные» варианты этих двух алгоритмов.

Заметим, что *k*-means похож также на поиск центра кластера в алгоритме FOREL. Отличие в том, что в FOREL кластер — это шар заданного радиуса *R*, тогда как в *k*-means объекты относятся к кластерам по принципу ближайшего соседа.

Существует два «канонических» варианта алгоритма *k*-means. Вариант Болла Холла представлен в Алгоритме 7.4. Вариант МакКина отличается тем, что всякий раз, когда некоторый объект x_i переходит из одного Алгоритм 7.4.

Кластеризация с помощью алгоритма *k*-средних

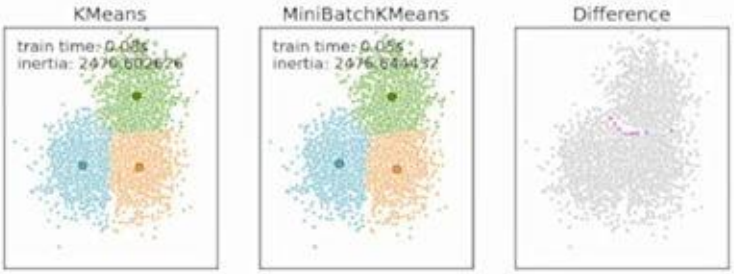
1: сформировать начальное приближение центров всех кластеров $y \in Y$:
μ_y — наиболее удалённые друг от друга объекты выборки;
2: **повторять**
3: отнести каждый объект к ближайшему центру (аналог E-шага):
$y_i := \arg\min_{y \in Y} \rho(x_i, \mu_y),\ i = 1, \ldots, \ell;$
4: вычислить новое положение центров (аналог M-шага):
$\mu_{yj} := \frac{\sum_{i=1}^{\ell} [y_i = y] f_j(x_i)}{\sum_{i=1}^{\ell} [y_i = y]},\ y \in Y,\ j = 1, \ldots, n;$
5: **пока** y_i не перестанут изменяться;

кластера в другой, центры обоих кластеров пересчитываются. Для этого шаг 4 надо перенести внутрь цикла по *i*, выполняемого на шаге 3. МакКин показал в 1967 году, что этот вариант алгоритма приводит к локальному минимуму функционала Φ_0.

Алгоритм *k*-means крайне чувствителен к выбору начальных приближений центров. Случайная инициализация центров на шаге 1 может приводить к плохим кластеризациям. Для формирования начального приближения можно выделить *k* наиболее удалённых точек выборки: первые две точки выделяются по максимуму всех попарных расстояний; каждая следующая точка выбирается так, чтобы расстояние от неё до ближайшей уже выделенной было максимально.

Модификации k-means

- На каждом шаге работаем с b случайно выбранными объектами из каждого кластера (mini-batch k-means)



- Критерий качества (k-medoids)

$$\tilde{J} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} d(\mathbf{x}_n, \mu_k)$$

d – функция расстояния, μ_k – один из объектов в кластере

Другая рекомендация — выполнить кластеризацию несколько раз, из различных случайных начальных приближений и выбрать кластеризацию с наилучшим значением заданного функционала качества.

Кластеризация может оказаться неадекватной и в том случае, если число кластеров будет изначально неверно угадано. Стандартная рекомендация — провести

кластеризацию при различных значениях kи выбрать то, при котором достигается резкое улучшение качества кластеризации по заданному функционалу.

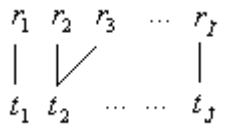
Кластеризация с частичным обучением. Алгоритмы EM и k-means легко приспособить для решения задач кластеризации с частичным обучением (semi-supervisedlearning), когда для некоторых объектов x_iизвестны правильные классификации $y*(x_i)$. Обозначим через Uподмножество таких объектов, $U \subset X^\ell$.

Примером такой задачи является рубрикация текстовых документов, в частности, страниц в Интернете. Типична ситуация, когда имеется относительно небольшое множество документов, вручную классифицированных по тематике. Требуется определить тематику большого числа неклассифицированных документов. Сходство документов $\rho(x,x')$ может оцениваться по-разному в зависимости от целей рубрикации и специфики самих документов: по частоте встречаемости ключевых слов, по частоте посещаемости заданным множеством пользователей, по количеству взаимных гипертекстовых ссылок, или другими способами.

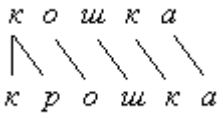
Модификация обоих алгоритмов довольно проста: на E-шаге (шаг 3) для всех $x_i \in U$полагаем $g_{iy} := [y = y*(x_i)]$, для всех остальных $x_i \in X^\ell \setminus U$скрытые переменные g_{iy}вычисляются как прежде. На практике частичная классификация даже небольшого количества объектов существенно улучшает качество кластеризации.

В качестве примера рассмотрим строчный образ (*слово*). В данном случае можно выделить два критерия, на основе которых можно строить меру близости:

- Совпадение букв,
- монотонность (совпадение порядкабукв).

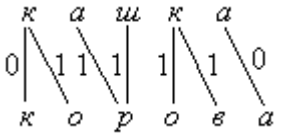


Пусть $r_1 r_2 \ldots r_I$ – эталон, $t_1 t_2 \ldots t_J$ – пробный образ, причем $I \neq J$. Построим соответствие между эталоном и пробным образом *по* следующему правилу: каждому символу в первом слове должен соответствовать хотя бы один символ во втором слове и каждому символу во втором слове должен соответствовать хотя бы один символ в первом слове, (но соответствие между символами не взаимнооднозначное, в частности, поскольку $I \neq J$).



Введем меру следующим образом:

$$\rho(r_i, t_i) = \begin{cases} 1, r_i \neq t_i \\ 0, r_i = t_i \end{cases}$$



В качестве меры сходства двух слов принимаем соответствие, при котором суммарный *вес* всех дуг (изображенных на рисунках) минимален:

$$\nu(\overline{r}, \overline{t}) = \min_S \mu(S), \text{ где } \mu(S) = \sum_{(i,j)\in S} \rho(r_i, t_j).$$

Через $\nu(\overline{r}, \overline{t})$ далее будем обозначать меру близости двух слов $\overline{r}$ и $\overline{t}$.

Соответствие S должно быть *двудольным графом* без изолированных вершин с непересекающимися ребрами. Рассмотрим задачу сравнения цепочек упорядоченных символов. В данной задаче могут возникать следующие ошибки:

- неправильно определенный символ (*кошка – корка*),
- ошибкавставки (*кошка – кошрка*),
- ошибкапотери (*кошка – кшка*).

Определение. *Редакторским расстоянием называется минимальное общее число изменений, вставок и потерь, требуемое для изменения образа* A *в образ* B :

$$D(A, B) = \min_j \left[C(j) + \dot{I}(j) + R(j) \right],$$

где минимизация происходит по всем возможным комбинациям символьных преобразований таких, чтобы получить B *из* A .

Пусть

$$d(i, j|i-1, j-1) = \begin{cases} 1, \text{ при } t(i) = r(i) \\ 0, \text{ при } t(i) \neq r(i) \end{cases}.$$

Тогда

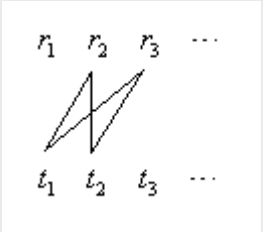
$$d(i, j|i-1, j) = d(i, j|i, j-1) = 1.$$

Построим таблицу, в которой столбцы – это символы образа, строки – символы эталона. Количество точек в матрице есть $\dot{I} \cdot \dot{J}$.

По данной таблице построим *граф по* следующему правилу. Если отображается точка (r_2, t_1), то далее выбираем (r_2, t_2), (r_3, t_2) или (r_3, t_1) (т.е. возможны

три варианта). Соответствие слов реализуется в виде маршрута в графе. Этот *маршрут* обязательно начинается с точки (r_1, t_1) (иначе появится изолированная точка) и заканчивается в (r_I, t_J).

	t_1	t_2	t_3	$\cdots$	t_J
r_1	○	○	○	$\cdots$	○
r_2	○	○	○	$\cdots$	○
r_3	○	○	○	$\cdots$	○
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
r_I	○	○	○	$\cdots$	○



Таким образом, получили задачу выбора кратчайшего пути на графе от точки (r_1, t_1) до точки (r_I, t_J), где каждая *вершина* имеет свою *стоимость*: 0 или 1.

Контура сложных многомерных объектов изображаются ломаными линиями, вершины которых будем называть узлами. Пусть заданы две линии – эталон и тестируемый *объект*. Используем следующую модель для сравнения объектов. Будем считать, что они изготовлены из проволоки и будем сравнивать близость этих ломаных путем оценки величины механической работы, которую нужно выполнить, чтобы преобразовать один *объект* в другой. Определим элементарную работу, которую надо совершить для перевода отдельных прямолинейных элементов ломаных. Достаточно рассмотреть два основных вида деформаций: растяжение (сжатие) и изгиб в узлах.

Каждой такой деформации припишем элементарную работу:

- $f(|l_1 - l_2|)$ – работа по изменению длины при растяжении и сжатии,
- $f(|\varphi_1 - \varphi_2|)$ – работа по изменению угла при изгибе.

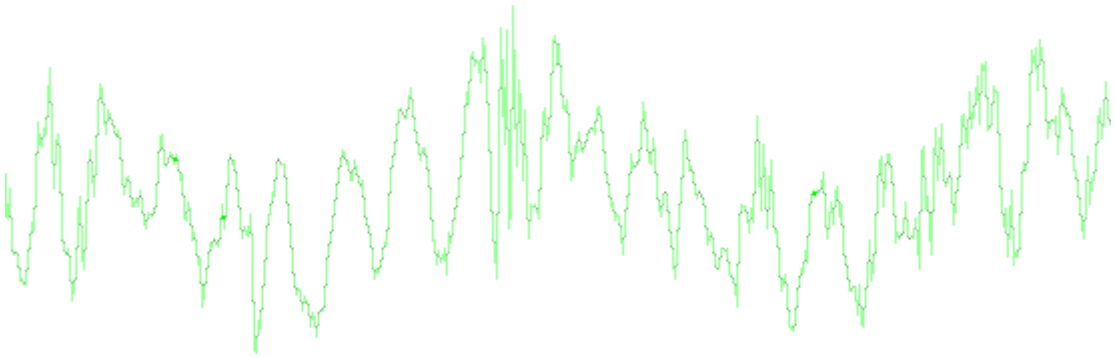
Задача состоит в поиске такого преобразования, чтобы затраченная работа была минимальной, т.е. надо найти

$$f_\Sigma \to \min_S .$$

Эта задача сводится к установлению соответствия узлов одной ломаной узлам другой. При этом не требуется взаимно-однозначное соответствие, но требуется сохранение

монотонности. Задача установления такого соответствия, которое минимизирует общую работу *по* деформации ломаных, также сводится к поиску минимального пути на графе такого же типа, как и рассмотренный в предыдущем пункте.

В графе каждая *дуга* получает *вес* $f(|l_1 - l_2|)$ – работу *по* сжатию или растяжению, а для каждой вершины – *вес* $f(|\varphi_1 - \varphi_2|)$ – работу *по* изменению угла.



В обработке речи можно выделить следующие основные направления:

- Распознавание отдельных слов (*IWR* – IsolatedWord *Recognition*),
- Распознаваниеслитнойречи (*CSR* – Continuous *Speech Recognition*).
- CDR – Speaker Dependent *Recognition*,
- *SIR* – Speaker Independent *Recognition*.

Ядром *IWR*-систем является совокупность эталонов и *мера*. *Отрезок* сигнала (см. рис.) $[0, T]$ разбивается на *сегменты*, т.е. сигнал квантуется (с перекрытием). С каждым сегментом связывается *вектор* коэффициентов Фурье.

Обработка звука происходит в два этапа.

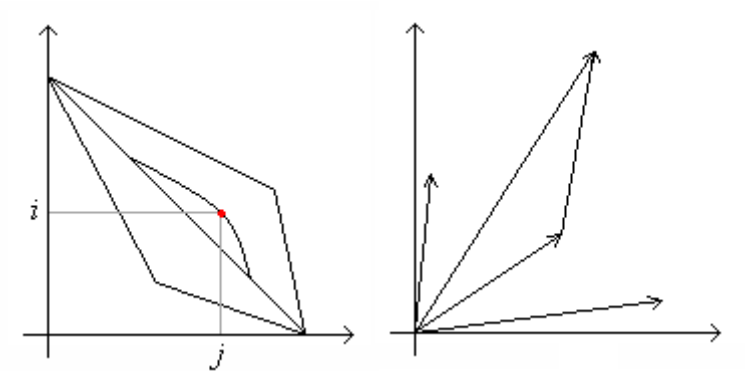
Первый этап. Строим цепочку $r(i)$, $i = 1, \ldots, \dot{I}$ – разговорные *сегменты*. Далее строим преобразование Фурье с разбиением на $t_f = 512$ отрезков. Обозначимчерез $x_i(n)$, $n = 0, \ldots, 511$ – отчетыдля i -огосегмента, $i = 1, \ldots, \dot{I}$. Тогда

$$X_i(m) = \frac{1}{\sqrt{512}} \sum_{n=0}^{511} x_i(n) \cdot \exp\left(-j\frac{2\pi}{512}\right), \; m = 0, \ldots, 511.$$

Рассмотрим первые $l, \; l \ll t_f$ (пусть $l \approx 50$), *коэффициентов Фурье* в качестве вектора признаков:

$$r(i) = \begin{bmatrix} X_i(0) \\ X_i(1) \\ \vdots \\ X_i(l-1) \end{bmatrix}, \; l = 1, \ldots, \dot{I}$$

Второй этап. Определяем ограничения в графе соответствия сегментов эталонной и тестируемой команд.



Глобальные ограничения – ограничения поля для оптимального маршрута, например, $|i - j| \leq k$ (рис. слева).

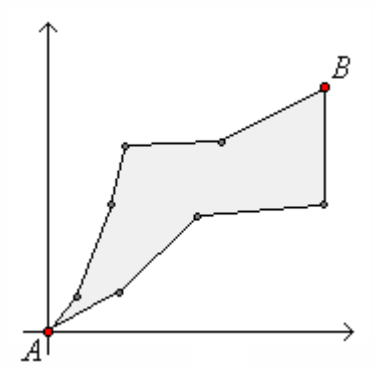
Локальные ограничения – монотонность на сети маршрутов (рис. справа).

Ограничения конечной точки.

Стоимость d – Евклидово *расстояние* между $r(i_k), t(j_k)$:

$$d(i_k, j_k | i_{k-1}, j_{k-1}) = \|r(i_k) - t(j_k)\| = d(i_k, j_k).$$

Таким образом, и эта задача также сводится к поиску кратчайшего пути на графе.



Задача поиска кратчайшего пути на графе может быть решена методом динамического программирования. Пусть (i_0, j_0) – начальный узел (отправной город), (i_f, j_f) – *конечный узел* (город – *пункт* назначения). Тогда задача состоит в поиске оптимального маршрута через промежуточные узлы (города):

$$(i_0, j_0) \xrightarrow{opt} (i_f, j_f)$$

Пусть (i, j) – промежуточный узел. Тогда *по* принципу оптимальности Беллмана имеем

$$(i_0, j_0) \xrightarrow{opt} (i_f, j_f) = \left((i_0, j_0) \xrightarrow{opt} (i, j)\right) \oplus \left((i, j) \xrightarrow{opt} (i_f, j_f)\right),$$

причем

$$D_{\min}(i_k, j_k) = \min_{(i_{k-1}, j_{k-1})} \left[D_{\min}(i_{k-1}, j_{k-1}) + d(i_k, j_k | i_{k-1}, j_{k-1})\right]$$

.

Иерархическая кластеризация

Иерархические алгоритмы кластеризации, называемые также алгоритмами таксономии, строят не одно разбиение выборки на непересекающиеся классы, а систему вложенных разбиений. Результат таксономии обычно представляется в виде таксономического дерева — дендрограммы. Классическим примером такого дерева является иерархическая классификация животных и растений.

Среди алгоритмов иерархической кластеризации различаются два основных типа. Дивизимные или нисходящие алгоритмы разбивают выборку на всё более и более мелкие кластеры. Более распространены агломеративные или восходящие алгоритмы, в которых объекты объединяются во всё более и более крупные кластеры. Реализация этой идеи представлена в Алгоритме 7.5.

Сначала каждый объект считается отдельным кластером. Для одноэлементных кластеров естественным образом определяется функция расстояния

$$R(\{x\},\{x'\}) = \rho(x,x').$$

Затем запускается процесс слияний. На каждой итерации вместо пары самых близких кластеров U и V образуется новый кластер $W = U \cup V$. Расстояние от нового кластера W до любого другого кластера S вычисляется по расстояниям $R(U,V)$, $R(U,S)$ и $R(V,S)$, которые к этому моменту уже должны быть известны:

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

где α_U, α_U, β, γ — числовые параметры. Эта универсальная формула обобщает практически все разумные способы определить расстояние между кластерами. Она была предложена Лансом и Уильямсом в 1967 году [49, 24].

На практике используются следующие способы вычисления расстояний $R(W,S)$ между кластерами W и S. Для каждого из них доказано соответствие формуле Ланса-Вильямса при определённых сочетаниях параметров [18]:

Расстояние ближнего соседа: $\alpha_U = \alpha_V = \frac{1}{2},\ \beta = 0,\ \gamma = -\frac{1}{2}$;

$R^{6}(W,S) = \min\limits_{w \in W, s \in S} \rho(w,s)$;

Расстояние дальнего соседа: $\alpha_U = \alpha_V = \frac{1}{2},\ \beta = 0,\ \gamma = \frac{1}{2}$;

$R^{\text{д}}(W,S) = \max\limits_{w \in W, s \in S} \rho(w,s)$;

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

$R^{\text{с}}(W,S) = \frac{1}{|W||S|} \sum\limits_{w \in W} \sum\limits_{s \in S} \rho(w,s)$;

Расстояние между центрами: $\alpha_U = \frac{|U|}{|W|},\ \alpha_V = \frac{|V|}{|W|},\ \beta = -\alpha_U \alpha_V,\ \gamma = 0$;

$R^{\text{ц}}(W,S) = \rho^2\Big(\sum\limits_{w \in W} \frac{w}{|W|}, \sum\limits_{s \in S} \frac{s}{|S|}\Big)$;

Расстояние Уорда: $\alpha_U = \frac{|S|+|U|}{|S|+|W|},\ \alpha_V = \frac{|S|+|V|}{|S|+|W|},$

$R^{\text{у}}(W,S) = \frac{|S||W|}{|S|+|W|} \rho^2\Big(\sum\limits_{w \in W} \frac{w}{|W|}, \sum\limits_{s \in S} \frac{s}{|S|}\Big)$; $\beta = \frac{-|S|}{|S|+|W|},\ \gamma = 0.$

Возможных вариантов слишком много, и на первый взгляд все они кажутся достаточно разумными. Возникает вопрос: какой из них предпочесть? Рассмотрим несколько дополнительных свойств, характеризующих качество кластеризации.

Алгоритм 7.5. Агломеративная кластеризация Ланса-Уильямса

1: инициализировать множество кластеров C_1:
   $t := 1;\quad C_t = \{\{x_1\},\ldots,\{x_\ell\}\}$;
2: **для всех** $t = 2,\ldots,\ell$ $\quad$ (t — номер итерации):
3: найти в C_{t-1} два ближайших кластера:
   $(U,V) := \arg\min\limits_{U \neq V} R(U,V)$;
   $R_t := R(U,V)$;
4: изъять кластеры U и V, добавить слитый кластер $W = U \cup V$:
   $C_t := C_{t-1} \cup \{W\} \setminus \{U,V\}$;
5: **для всех** $S \in C_t$
6: вычислить расстояние $R(W,S)$ по формуле Ланса-Уильямса;

Свойство монотонности. Обозначим через R_tрасстояние между ближайшими кластерами, выбранными на t-м шаге для слияния. Говорят, что функция расстояния Rобладает свойством монотонности, если при каждом слиянии расстояние между объединяемыми кластерами только увеличивается: $R_2 \leqslant R_3 \leqslant \ldots \leqslant R_\ell$.

Свойство монотонности позволяет изобразить процесс кластеризации в виде специального графика, называемого дендрограммой. По вертикальной оси откладываются объекты, по горизонтальной — расстояния R_t. Нетрудно доказать, что если кластеризация

обладает свойством монотонности, то дендрограмму можно построить так, чтобы она не имела самопересечений. При этом любой кластер из множества C_tпредставляется сплошной последовательностью точек на вертикальной оси. Если же процесс кластеризации идёт не монотонно, то вместо дендрограммы получается запутанный клубок линий, на котором трудно что-либо разобрать.

Дендрограмма позволяет представить кластерную структуру в виде плоского графика независимо от того, какова размерность исходного пространства. Существуют и другие способы двумерной визуализации многомерных данных, такие как многомерное шкалирование или карты Кохонена, но они привносят в картину искусственные искажения, влияние которых довольно трудно оценить.

Оказывается, не любое сочетание коэффициентов в формуле Ланса-Вильямса приводит к монотонной кластеризации.

Теорема 7.1 (Миллиган, 1979). Если выполняются следующие три условия, то кластеризация является монотонной:

$$1)\ \alpha_U \geqslant 0,\ \alpha_V \geqslant 0;$$
$$2)\ \alpha_U + \alpha_V + \beta \geqslant 1;$$
$$3)\ \min\{\alpha_U, \alpha_V\} + \gamma \geqslant 0.$$

Из перечисленных выше расстояний только $R^{ц}$ не является монотонным. Расстояние Уорда отличается от него мультипликативной поправкой, которая и делает его монотонным.

Свойства растяжения и сжатия. Некоторые расстояния обладают свойством растяжения. По мере того, как кластер укрупняется, расстояния от него до других кластеров увеличиваются, как будто пространство вокруг кластера растягивается.
На дендрограмме растягивающие расстояния характеризуются повышением плотности линий слева, в области наименьших значений R_t. Свойство растяжения считается желательным, так как оно способствует более чёткому отделению кластеров. С другой стороны, при слишком сильном растяжении возможно найти кластеры там, где их изначально не было. Растягивающими являются расстояния $R^{д}$ и $R^{у}$.

Некоторые расстояния, наоборот, обладают свойством сжатия. По мере роста кластера расстояния от него до других кластеров уменьшается, и кажется, что пространство вокруг кластера сжимается. Естественная кластеризация при этом может исчезнуть. Для сжимающих расстояний характерно уплотнение дендрограммы справа, в области наибольших значений R_t. Расстояние ближнего соседа $R^{б}$ является сильно сжимающим.

Свойства сжатия и растяжения определяются через отношение $R_t/\rho(\mu_U,\mu_V)$, где $R_t = R(U,V)$ — расстояние между ближайшими кластерами, объединяемыми на t-м шаге, μ_Uи μ_V— центры этих кластеров. Если это отношение на каждом шаге больше единицы, то расстояние Rявляется растягивающим; если оно всегда меньше единицы, то сжимающим. Есть и такие расстояния, которые не являются ни сжимающими, ни растягивающими, например, $R^{с}$ и $R^{ц}$. О них говорят, что они сохраняют метрику пространства.

На практике часто применяют гибкое расстояние, которое представляет собой компромисс между методами ближнего соседа, дальнего соседа и среднего расстояния. Оно определяется одним параметром βвместо четырёх:

$$\alpha_U = \alpha_V = (1 - \beta)/2, \qquad \gamma = 0, \qquad \beta < 1.$$

Гибкое расстояние является сжимающим при $\beta>0$ и растягивающим при $\beta<0$. Стандартная рекомендация: $\beta = -0{,}25$ [24].

Свойство редуктивности. Самой трудоёмкой операцией в Алгоритме 7.5 является поиск пары ближайших кластеров на шаге 3. Он требует $O(\ell^2)$ операций внутри основного цикла. Соответственно, построение всего таксономического дерева требует $O(\ell^3)$ операций. Это ограничивает применимость алгоритма выборками длины в несколько сотен объектов.

Идея ускорения алгоритма заключается в том, чтобы перебирать лишь наиболее близкие пары. Задаётся параметр δ, и перебор ограничивается сокращённым множеством пар $\{(U,V)\colon R(U,V)\leqslant\delta\}$. Когда все такие пары будут исчерпаны, параметр δ увеличивается, и формируется новое сокращённое множество пар. И так далее, до полного слияния всех объектов в один кластер. Реализация представлена в Алгоритме 7.6.

Доказано, что этот алгоритм приводит к той же кластеризации, что и Алгоритм 7.5, если расстояние R обладает свойством редуктивности:

Опр. 7.1 (Брюинош, 1978). Расстояние R называется редуктивным, если для любого $\delta>0$ и любых δ-близких кластеров U и V объединение δ-окрестностей U и V содержит в себе δ-окрестность кластера $W = U\cup V$:

$$\{S \mid R(U\cup V,S)<\delta,\ R(U,V)\leqslant\delta\} \subseteq \{S \mid R(S,U)<\delta \text{ или } R(S,V)<\delta\}.$$

Алгоритм 7.6. Быстрая агломеративная кластеризация на основе редуктивности

1: инициализировать множество кластеров C_1:
   $t := 1;\quad C_t = \{\{x_1\},\ldots,\{x_\ell\}\}$;
2: выбрать начальное значение параметра δ;
3: $P(\delta) := \{(U,V) \mid U,V\in C_t,\ R(U,V)\leqslant\delta\}$;
4: **для всех** $t = 2,\ldots,\ell$ (t — номер итерации):
5: **если** $P(\delta) = \varnothing$ **то**
6: увеличить δ так, чтобы $P(\delta)\neq\varnothing$;
7: найти в $P(\delta)$ пару ближайших кластеров:
   $(U,V) := \arg\min\limits_{(U,V)\in P(\delta)} R(U,V)$;
   $R_t := R(U,V)$;
8: изъять кластеры U и V, добавить слитый кластер $W = U\cup V$:
   $C_t := C_{t-1}\cup\{W\}\setminus\{U,V\}$;
9: **для всех** $S\in C_t$
10: вычислить расстояние $R(W,S)$ по формуле Ланса-Уильямса;
11: **если** $R(W,S)\leqslant\delta$ **то**
12: $P(\delta) := P(\delta)\cup\{(W,S)\}$;

Теорема 7.2 (Диде и Моро, 1984). Если выполняются следующие три условия, то расстояние R является редуктивным:

1) $\alpha_U\geqslant 0,\ \alpha_V\geqslant 0$;
2) $\alpha_U+\alpha_V+\min\{\beta,0\}\geqslant 1$;
3) $\min\{\alpha_U,\alpha_V\}+\gamma\geqslant 0$.

Сравнение условий теорем 7.2 и 7.1, показывает, что всякое редуктивное расстояние является монотонным, следовательно, позволяет отобразить процесс кластеризации в виде дендрограммы. Из перечисленных выше расстояний только $R^{ц}$ не является редуктивным.

В Алгоритме 7.6 возможны различные эвристические стратегии выбора параметра δна шагах 2 и 6. Общие соображения таковы: если δнастолько велико, что $P(\delta)$ содержит практически все пары кластеров, то мы фактически возвращаемся к неэффективному Алгоритму 7.5; если же δмало´, то приходится слишком часто формировать множество $P(\delta)$. На практике поступают следующим образом. Если число кластеров в C_tне превышает порог n_1, то в качестве $P(\delta)$ берут множество всех возможных пар (U,V)из C_t. В противном случае выбирается случайным образом n_2 расстояний $R(U,V)$, и δполагается равным наименьшему из них. В случае редуктивности параметры алгоритма n_1 и n_2 влияют только на время выполнения алгоритма, но не на результат кластеризации. Оптимальные значения для них подбираются с помощью калибровочных тестов и, вообще говоря, зависят от компьютера. В качестве начального выбора можно предложить $n_1 = n_2 = 20$.

Определение числа кластеров проще всего производить путём отсечения правого участка дендрограммы. На горизонтальной оси находится интервал максимальной длины $|R_{t+1} - R_t|$, и в качестве результирующей кластеризации выдаётся множество кластеров C_t. Число кластеров равно $K = \ell - t + 1$. При необходимости можно задать ограничение на минимальное и максимальное число кластеров $K_0 \leq K \leq K_1$ и выбирать t, удовлетворяющие ограничениям $\ell - K_1 + 1 \leq t \leq \ell - K_0 + 1$.

Во многих прикладных задачах интерес представляет таксономическое дерево целиком, и определять оптимальное число кластеров не имеет особого смысла.

Достоинства и недостатки агломеративной кластеризации. Точного ответа на вопрос, какой алгоритм кластеризации лучше, не существует. Каждое из расстояний, перечисленных выше, имеет свои недостатки и подходит не для всех задач.

Метод ближнего соседа обладает цепочечным эффектом, когда независимо от формы кластера к нему присоединяются ближайшие к границе объекты. В некоторых случаях это приводит к тому, что кластеры «отращивают щупальца». В зависимости от задачи это свойство может быть как полезным, так и мешающим. Метод ближнего соседа хорошо подходит для выделения кластеров ленточной формы.

Метод дальнего соседа цепочечного эффекта не имеет, но на раннем этапе может объединять довольно несхожие группы.

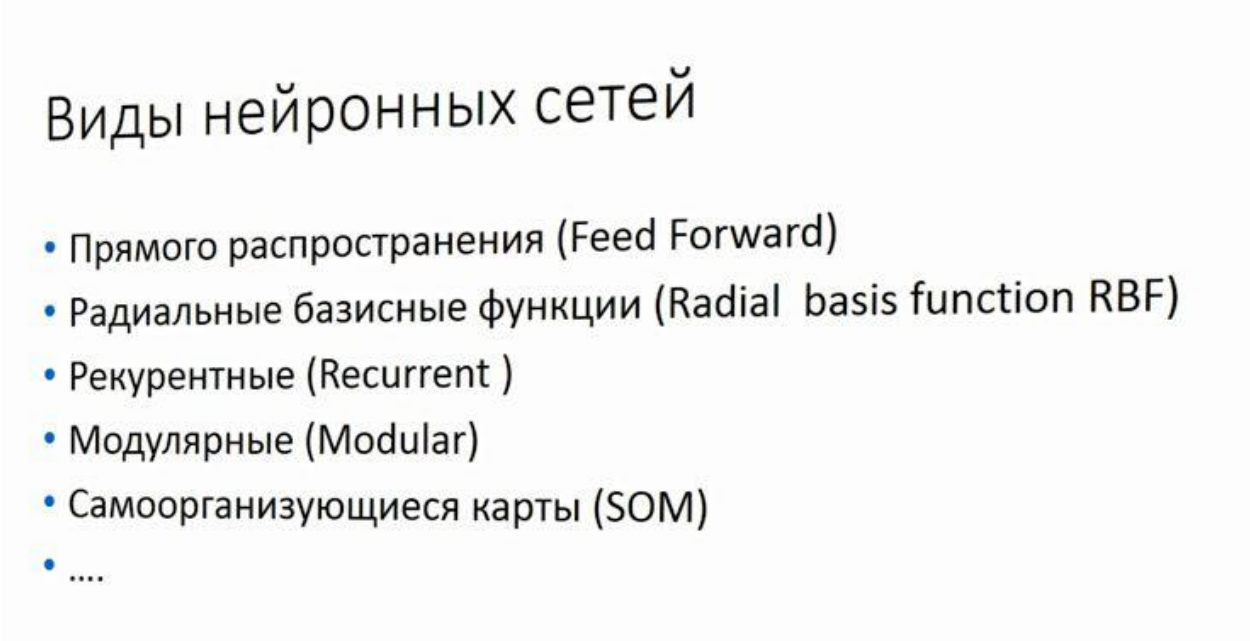
Расстояние между центрами масс не монотонно и не редуктивно, поэтому редко используется на практике, хотя интуитивно кажется «золотой серединой».

Метод Уорда оказался наилучшим по результатам экспериментального сравнения на представительном наборе модельных задач. Он чаще других методов строит интуитивно лучшую таксономию.

Нейронные структуры, основанные на обучении без учителя.

Главная черта, делающая обучение без учителя привлекательным, – это его "самостоятельность". Процесс обучения, как и в случае обучения с учителем, заключается в подстраивании весов синапсов. Некоторые алгоритмы, правда, изменяют и структуру сети, то есть количество нейронов и их взаимосвязи, но такие преобразования правильнее назвать более широким термином – самоорганизацией, и в рамках данной главы они рассматриваться не будут. Очевидно, что подстройка

синапсов может проводиться только на основании информации, доступной в нейроне, то есть его состояния и уже имеющихся весовых коэффициентов.



Исходя из этого соображения и, что более важно, по аналогии с известными принципами самоорганизации нервных клеток, построены алгоритмы обучения Хебба.

Сигнальный метод обучения Хебба заключается в изменении весов по следующему правилу:

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \cdot y_i^{(n-1)} \cdot y_j^{(n)} \quad (1)$$

где $y_i^{(n-1)}$ – выходное значение нейрона i слоя (n-1), $y_j^{(n)}$ – выходное значение нейрона j слоя n; $w_{ij}(t)$ и $w_{ij}(t-1)$ – весовой коэффициент синапса, соединяющего эти нейроны, на итерациях t и t-1 соответственно; a – коэффициент скорости обучения. Здесь и далее, для общности, под n подразумевается произвольный слой сети. При обучении по данному методу усиливаются связи между возбужденными нейронами.

Существует также и дифференциальный метод обучения Хебба.

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \cdot \left[y_i^{(n-1)}(t) - y_i^{(n-1)}(t-1) \right] \cdot \left[y_j^{(n)}(t) - y_j^{(n)}(t-1) \right] \quad (2)$$

Здесь $y_i^{(n-1)}(t)$ и $y_i^{(n-1)}(t-1)$ – выходное значение нейрона i слоя n-1 соответственно на итерациях t и t-1; $y_j^{(n)}(t)$ и $y_j^{(n)}(t-1)$ – то же самое для нейрона j слоя n. Как видно из формулы (2), сильнее всего обучаются синапсы, соединяющие те нейроны, выходы которых наиболее динамично изменились в сторону увеличения.

Полный алгоритм обучения с применением вышеприведенных формул будет выглядеть так:

1. На стадии инициализации всем весовым коэффициентам присваиваются небольшие случайные значения.

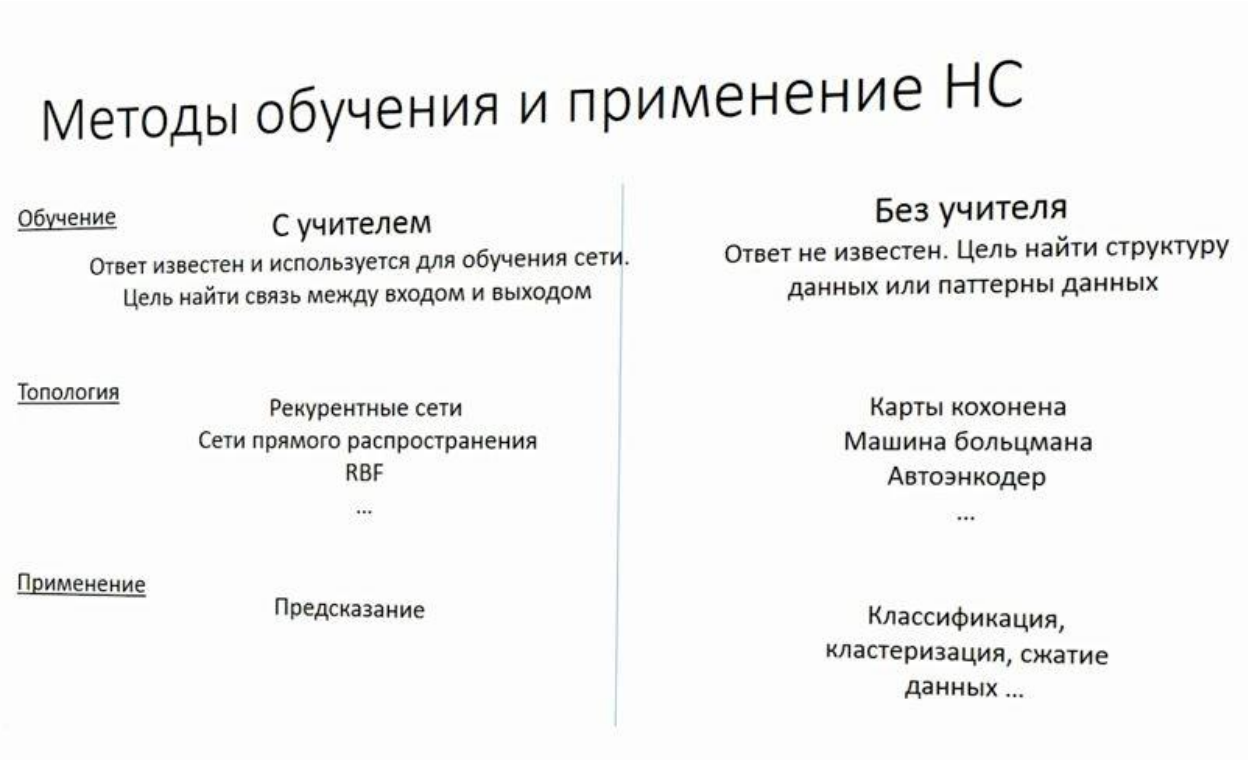
2. На входы сети подается входной образ, и сигналы возбуждения распространяются по всем слоям согласно принципам классических прямопоточных (feedforward) сетей[1], то есть для каждого нейрона рассчитывается взвешенная сумма его входов, к

которой затем применяется активационная (передаточная) функция нейрона, в результате чего получается его выходное значение $y_i^{(n)}$, i=0...M_i-1, где M_i – число нейронов в слое i; n=0...N-1, а N – число слоев в сети.

3. На основании полученных выходных значений нейронов по формуле (1) или (2) производится изменение весовых коэффициентов.

4. Цикл с шага 2, пока выходные значения сети не застабилизируются с заданной точностью. Применение этого нового способа определения завершения обучения, отличного от использовавшегося для сети обратного распространения, обусловлено тем, что подстраиваемые значения синапсов фактически не ограничены.

На втором шаге цикла попеременно предъявляются все образы из входного набора.



Следует отметить, что вид откликов на каждый класс входных образов не известен заранее и будет представлять собой произвольное сочетание состояний нейронов выходного слоя, обусловленное случайным распределением весов на стадии инициализации. Вместе с тем, сеть способна обобщать схожие образы, относя их к одному классу. Тестирование обученной сети позволяет определить топологию классов в выходном слое. Для приведения откликов обученной сети к удобному представлению можно дополнить сеть одним слоем, который, например, по алгоритму обучения однослойного перцептрона необходимо заставить отображать выходные реакции сети в требуемые образы.

Другой алгоритм обучения без учителя – алгоритм Кохонена – предусматривает подстройку синапсов на основании их значений от предыдущей итерации.

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \cdot \left[y_i^{(n-1)} - w_{ij}(t-1) \right] \quad (3)$$

Из вышеприведенной формулы видно, что обучение сводится к минимизации разницы между входными сигналами нейрона, поступающими с выходов нейронов предыдущего слоя $y_i^{(n-1)}$, и весовыми коэффициентами его синапсов.

Полный алгоритм обучения имеет примерно такую же структуру, как в методах Хебба, но на шаге 3 из всего слоя выбирается нейрон, значения синапсов которого максимально походят на входной образ, и подстройка весов по формуле (3) проводится только для него. Эта, так называемая, аккредитация может сопровождаться затормаживанием всех остальных нейронов слоя и введением выбранного нейрона в насыщение. Выбор такого нейрона может осуществляться, например, расчетом скалярного произведения вектора весовых коэффициентов с вектором входных значений. Максимальное произведение дает выигравший нейрон.

Другой вариант – расчет расстояния между этими векторами в p-мерном пространстве, где p – размер векторов.

$$D_j = \sqrt{\sum_{i=0}^{p-1} (y_i^{(n-1)} - w_{ij})^2}, \quad (4)$$

где j – индекс нейрона в слое n, i – индекс суммирования по нейронам слоя (n-1), w_{ij} – вес синапса, соединяющего нейроны; выходы нейронов слоя (n-1) являются входными значениями для слоя n. Корень в формуле (4) брать не обязательно, так как важна лишь относительная оценка различных D_j.

В данном случае, "побеждает" нейрон с наименьшим расстоянием. Иногда слишком часто получающие аккредитацию нейроны принудительно исключаются из рассмотрения, чтобы "уравнять права" всех нейронов слоя. Простейший вариант такого алгоритма заключается в торможении только что выигравшего нейрона.

При использовании обучения по алгоритму Кохонена существует практика нормализации входных образов, а так же – на стадии инициализации – и нормализации начальных значений весовых коэффициентов.

$$x_i = x_i / \sqrt{\sum_{j=0}^{n-1} x_j^2}, \quad (5)$$

где x_i – i-ая компонента вектора входного образа или вектора весовых коэффициентов, а n – его размерность. Это позволяет сократить длительность процесса обучения.

Инициализация весовых коэффициентов случайными значениями может привести к тому, что различные классы, которым соответствуют плотно распределенные входные образы, сольются или, наоборот, раздробятся на дополнительные подклассы в случае близких образов одного и того же класса. Для избежания такой ситуации используется метод выпуклой комбинации[3]. Суть его сводится к тому, что входные нормализованные образы подвергаются преобразованию:

$$x_i = \alpha(t) \cdot x_i + (1 - \alpha(t)) \cdot \frac{1}{\sqrt{n}}, \quad (6)$$

где x_i – i-ая компонента входного образа, n – общее число его компонент, а (t) – коэффициент, изменяющийся в процессе обучения от нуля до единицы, в результате чего вначале на входы сети подаются практически одинаковые образы, а с течением

времени они все больше сходятся к исходным. Весовые коэффициенты устанавливаются на шаге инициализации равными величине

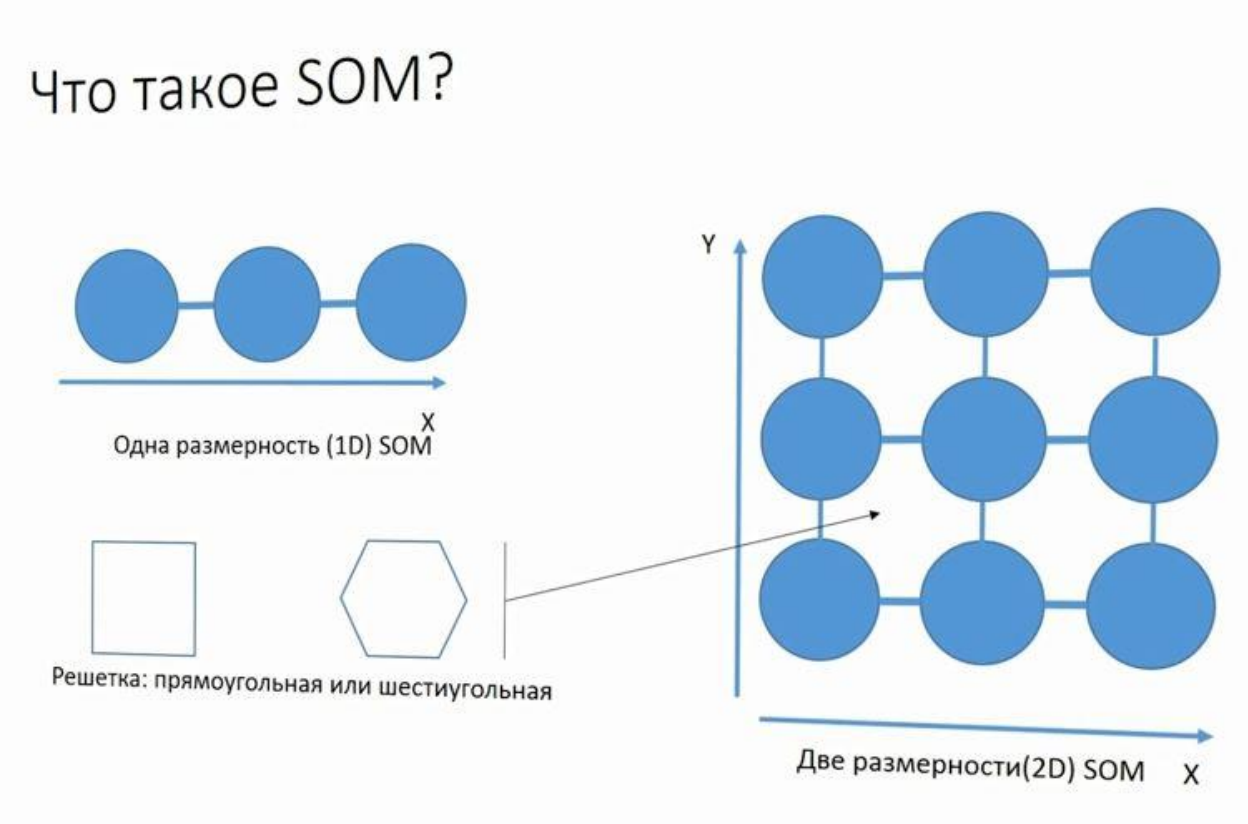
$$w_o = \frac{1}{\sqrt{n}}, \quad (7)$$

где n – размерность вектора весов для нейронов инициализируемого слоя.

На основе рассмотренного выше метода строятся нейронные сети особого типа – так называемые самоорганизующиеся структуры – self-organizing feature maps (этот устоявшийся перевод с английского, на мой взгляд, не очень удачен, так как, речь идет не об изменении структуры сети, а только о подстройке синапсов). Для них после выбора из слоя n нейрона j с минимальным расстоянием D_j (4) обучается по формуле (3) не только этот нейрон, но и его соседи, расположенные в окрестности R. Величина R на первых итерациях очень большая, так что обучаются все нейроны, но с течением времени она уменьшается до нуля. Таким образом, чем ближе конец обучения, тем точнее определяется группа нейронов, отвечающих каждому классу образов.

Карты Кохонена

Карты Кохонена представляют специализированные нейронные сети применяемые для решения задач векторного квантования и визуализации многомерных данных.

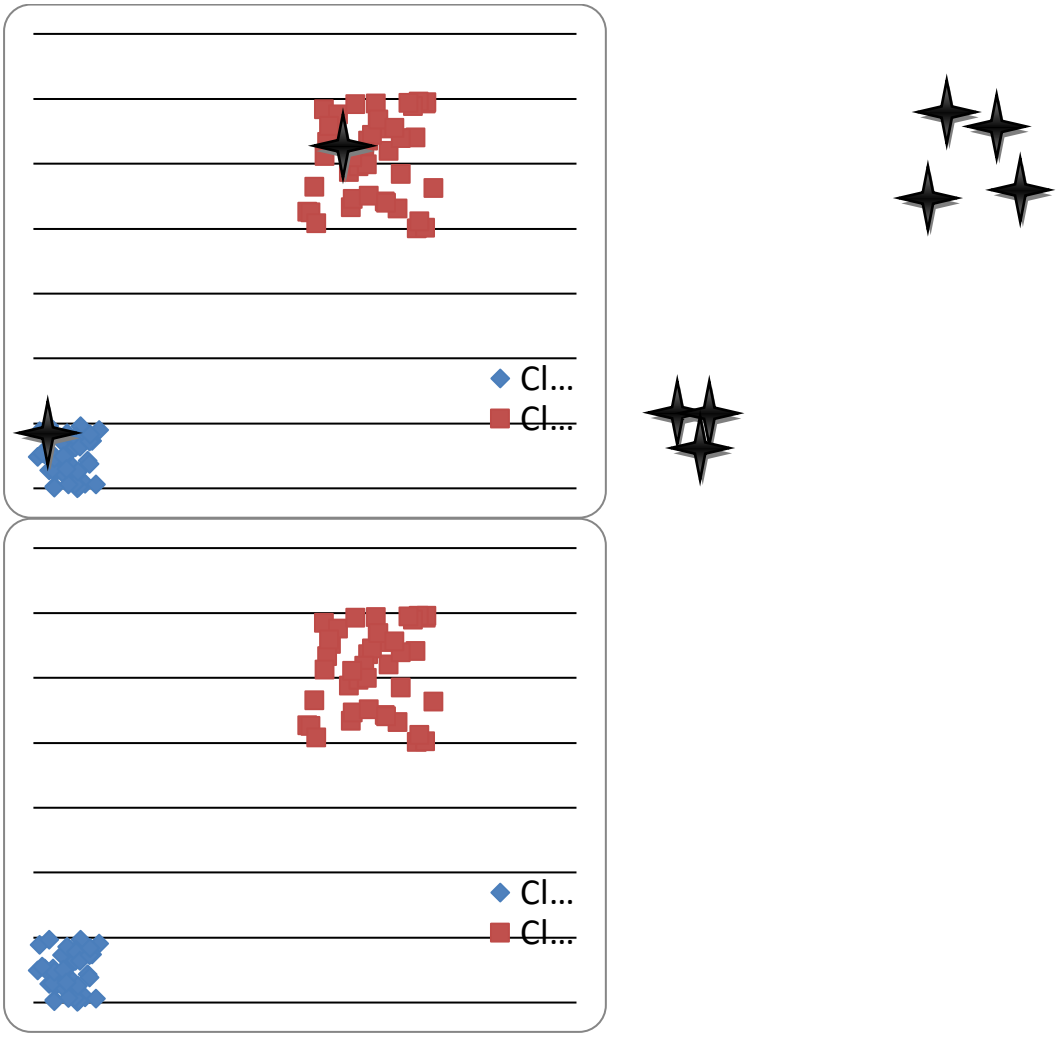


Задача **векторного квантования** является несколько более специфичной, чем задача кластеризации[1], однако ее целью также является нахождение структуры в данных. Неформально задачу можно сформулировать так: для данного множества точек

[1] И возможно поэтому реже упоминается в литературе.

данных необходимо найти скопления точек («недо-кластеры»). Другими словами, нужно также разбить исходное множество точек на непересекающиеся подмножества в соответствии с гипотезой компактности, но теперь к каждому подмножеству не предъявляется дополнительных требований вроде интерпретируемости, а количество получаемых подмножеств заранее может не оговариваться, и не играет столь значительной роли.

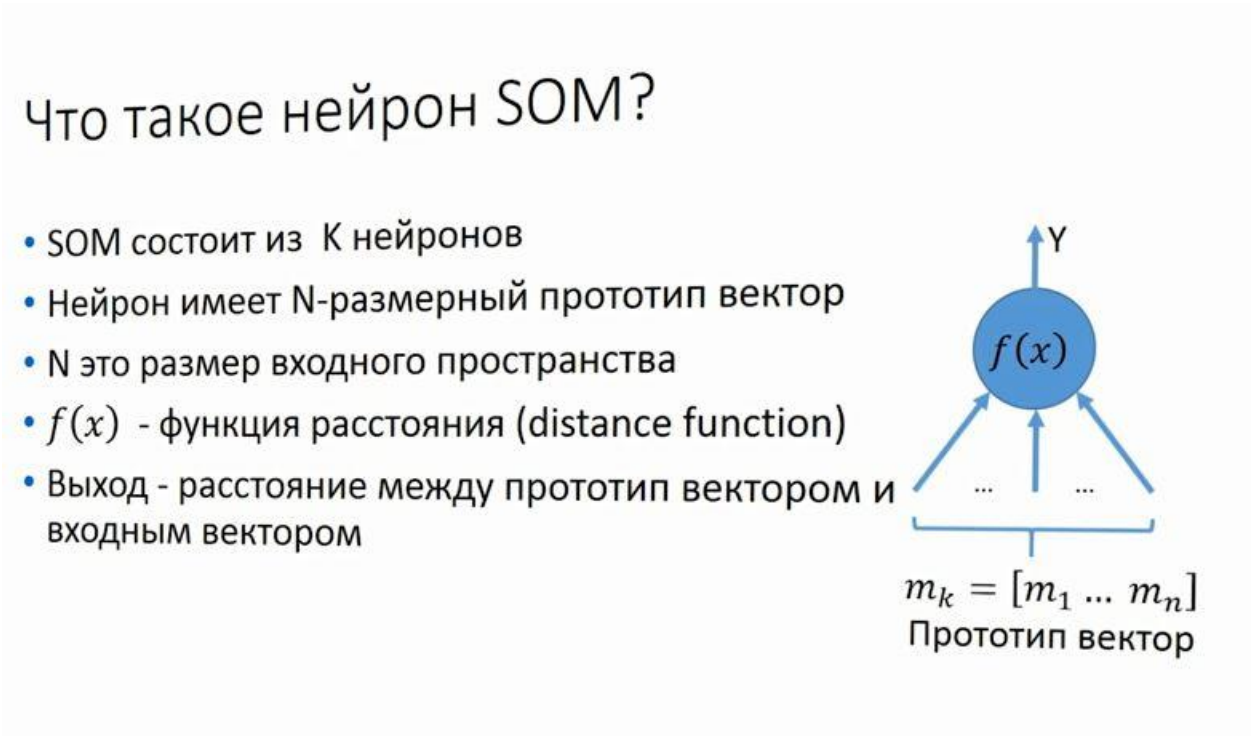
Отличия между кластеризацией и векторным квантованием заключается в том, что для одних и тех же исходных данных алгоритм кластеризации должен найти центры кластеров, а для алгоритма векторного квантования достаточно описать скопления точек, даже если это скопление является частью кластера.



а) б)

Рис. 1. Различие между результатами работы алгоритмов кластеризации (а) и векторного квантования (б)

На рис. 1 показан иллюстративный пример. Имеется два кластера. Алгоритм кластеризации (1а) должен определить центры кластеров, в то время как алгоритм векторного квантования описывает исходное множество точек меньшим числом точек, при этом сохраняя большую часть информации о структуре данных. В силу данной особенности алгоритм векторного квантования может быть использован для сокращения мощности исходного множества точек.



Обучение карты Кохонена

Традиционно карта Кохонена (КК) представляется двумерным слоем нейронов, каждый из которых связан со входным слоем. Каждый нейрон в карте Кохонена имеет два параметра:

1. Вектор $\mathbf{w}$ весов связей. Размерность $\mathbf{w}$ равна количеству входов ИНС.
2. Двумерные координаты $\mathbf{z}$, определяющие положение нейрона относительно других нейронов.

Сам слой нейронов работает по принципу WTA –Winner Takes All. В соответствии с этим принципом, в слое нейронов активированным («победителем») считается только один нейрон. Для разных входных векторов победителями могут быть разные нейроны. При обучении карты Кохонена корректируются веса связей нейрона-победителя и тех нейронов, которые к нему ближе[2]. Как это делается и почему именно так, будет объяснено ниже.

В картах Кохонена нейрон победитель определяется по тому, насколько вектор его весов близок к текущему входному вектору. Это можно определить, просто вычислив

$$winner_i = \arg\min_{j=1..K} r(\mathbf{x}_i, \mathbf{w}_j),$$

где r – метрика, $\mathbf{x}_i$ – текущий входной вектор, $\mathbf{w}_j$ – вектор весов j-го нейрона, K – количество нейронов в сети Кохонена. Можно использовать и функцию активации, например, вида:

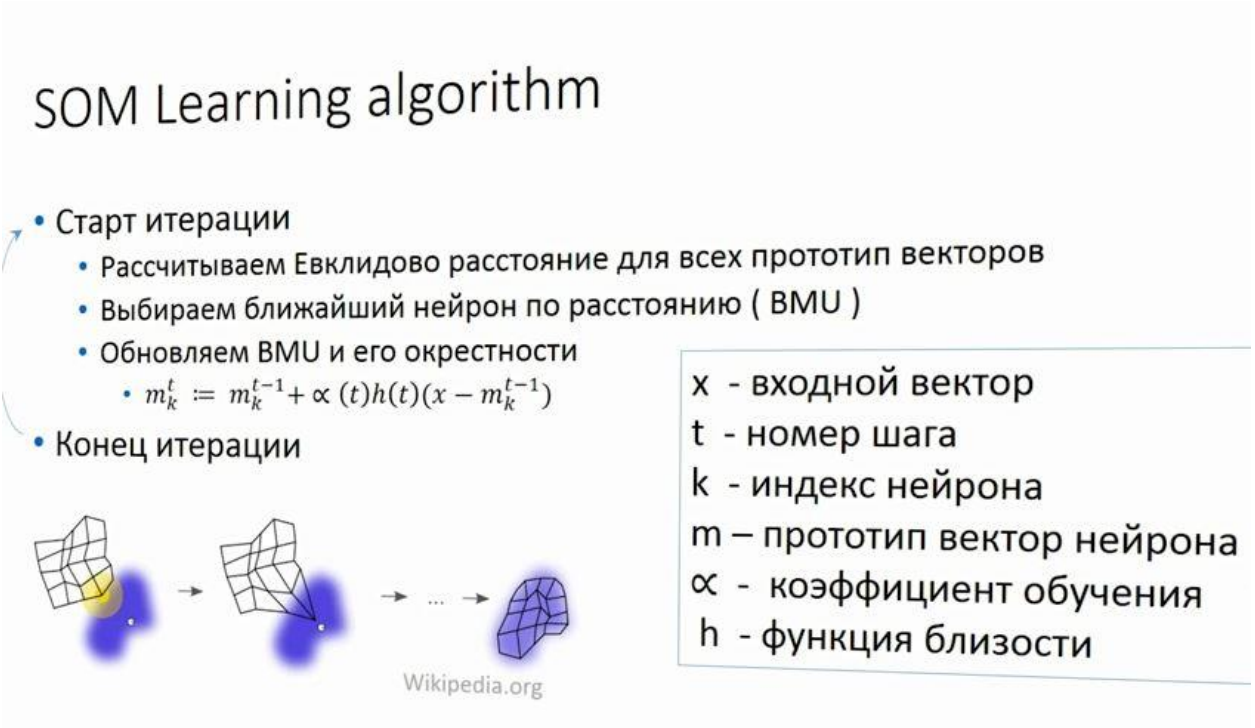
$$f(\mathbf{x}_i, \mathbf{w}_j) = \exp\left(-\left\|\mathbf{x}_i - \mathbf{w}_j\right\|\right), \qquad (1)$$

где $\|\cdot\|$ – евклидова мера. Функция (1) принимает максимальное значение, если $\mathbf{x}_i = \mathbf{w}_j$, или, что то же самое, $r(\mathbf{x}_i, \mathbf{w}_j) = 0$. Нейрон победитель выбирается следующим образом:

[2] Вот для чего нужны двумерные координаты!

$$winner_i = \arg\max_{j=1..K} f(\mathbf{x}_i, \mathbf{w}_j),$$

т.е. побеждает нейрон с максимальным выходным сигналом.



Инициализация нейронов сети производится следующим образом:

1. Случайная инициализация векторов $\mathbf{w}$ весов нейронов.
2. Начальные координаты $\mathbf{z}$ нейронов совпадают с узлами прямоугольной решетки. В некоторых случаях координаты задают с малым шумом.

Отметим, что инициализация весов нейронов оказывает существенное влияние на результат, однако наилучший способ инициализации, обеспечивающий оптимальный вариант, неизвестен. Тем не менее, для выполнения лабораторной работы будем использовать случайную инициализацию.

Пример сети 3x3 показан на рис. 2.

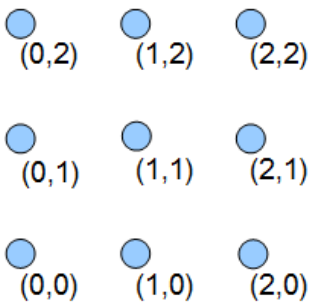
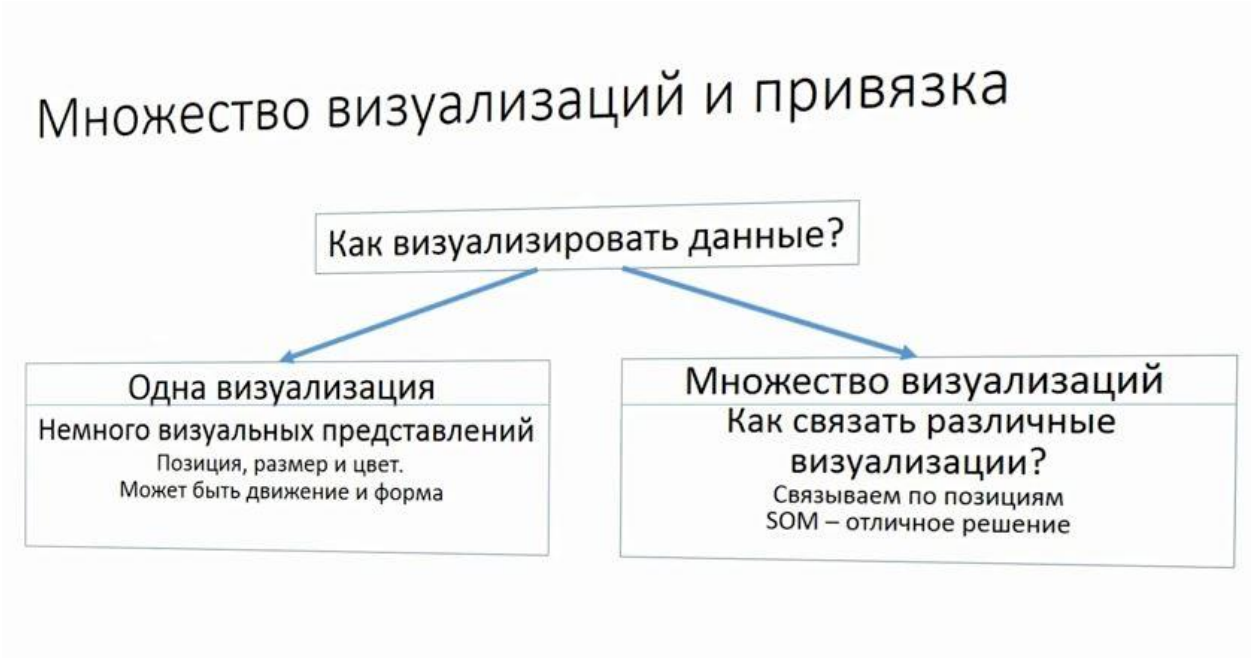


Рис. 2. Начальное расположение нейронов в карте Кохонена размером 3x3. Показаны двумерные координаты $\mathbf{z}$ нейронов. Веса нейронов не показаны

Среди алгоритмов обучения наиболее часто используются два:

1. Последовательный (*Iterative*). Обновление весов производится после каждого обучающего примера.
2. Пакетный (*Batch Map*). Сначала предъявляются все примеры, а потом производится обновление весов.

Пакетный алгоритм требует б*о*льших затрат по памяти, однако может сходиться в несколько раз быстрее последовательного.



Алгоритм последовательного обучения

Формулы для обновления весов связей для *j*-го нейрона по последовательному алгоритму:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \eta(t) h_{cj}(t)[\mathbf{x}_i(t) - \mathbf{w}_j(t)], \qquad (2)$$

где η – скорость обучения, h_{cj} – функция от расстояния от *j*-го нейрона до нейрона победителя, $\mathbf{x}_i(t)$ – *i*-й обучающий вектор. Выбор вида h_{cj} играет важную роль в обучении карты Кохонена. Типичным примером является Гауссиан:

$$h_{cj} = \exp\left(-\frac{d_{cj}^2}{2\sigma^2}\right),$$

где $d_{cj}^2 = \left\|\mathbf{z}_j - \mathbf{z}_c\right\|$ – расстояние на плоскости между *j*-м нейроном и текущим нейроном-победителем; σ – числовой параметр, называемый *эффективной шириной*, определяющий размер окрестности вокруг нейрона победителя, в которой производится коррекция весов. Чем меньше σ, тем меньше нейронов из окрестности победителя, на которых влияет предъявляемый входной вектор $\mathbf{x}_i$.

Особенностью обучения КК является постепенное уменьшение размера окрестности. Часто применяют:

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_1}\right), \quad t = 1,2,\ldots,$$

где τ_1 – параметр, отвечающий за скорость убывания $\sigma(t)$. Рекомендуемое значение для τ_1:

$$\tau_1 = \frac{T}{\sigma_0},$$

где T – предполагаемое время обучения, например T = 1000.

Рекомендуемое значение для σ_0:

$$\sigma_0 = r,$$

где r – радиус «решетки» нейронов, определяется по двумерным координатам нейронов. Для случая прямоугольной решетки с целочисленными координатами узлов (см. пример на рис. 2), можно считать, $r = \left\lfloor \frac{\min\{H, W\}}{2} \right\rfloor$, где W и H – соответственно ширина и высота конфигурации нейронов, $\lfloor \cdot \rfloor$ – взятие ближайшего целого снизу (отбрасывание дробной части). Так, если нейроны расположены в виде сетки 7x7, то r = 3, $\sigma_0 = 3$.

Описательно работу алгоритма последовательного обучения можно представить следующим образом: для каждого входного вектора из обучающего набора ищется нейрон, веса связей которого ближе всего к этому вектору. В результате обучения веса нейрона-победителя становятся еще сильнее «похожи» на входной вектор, кроме этого изменяются и веса нейронов, которые близки к нейрону-победителю. В пакетном режиме происходит то же самое, однако веса нейрона корректируются за один шаг под влиянием всех обучающих векторов, для которых этот нейрон оказался победителем.

Для сходимости алгоритма обучения необходимо выполнение следующего условия:

$\eta(t) \to 0$ при $t \to +\infty$.

Часто используется соотношение:

$$\eta(t) = \eta_0 \exp\left(-\frac{t}{\tau_2}\right), \quad t = 1,2,\ldots,$$

где η_0 – начальная скорость обучения, τ_2 – параметр, регулирующий убывание скорости обучения. Рекомендуемые значения для этих параметров:

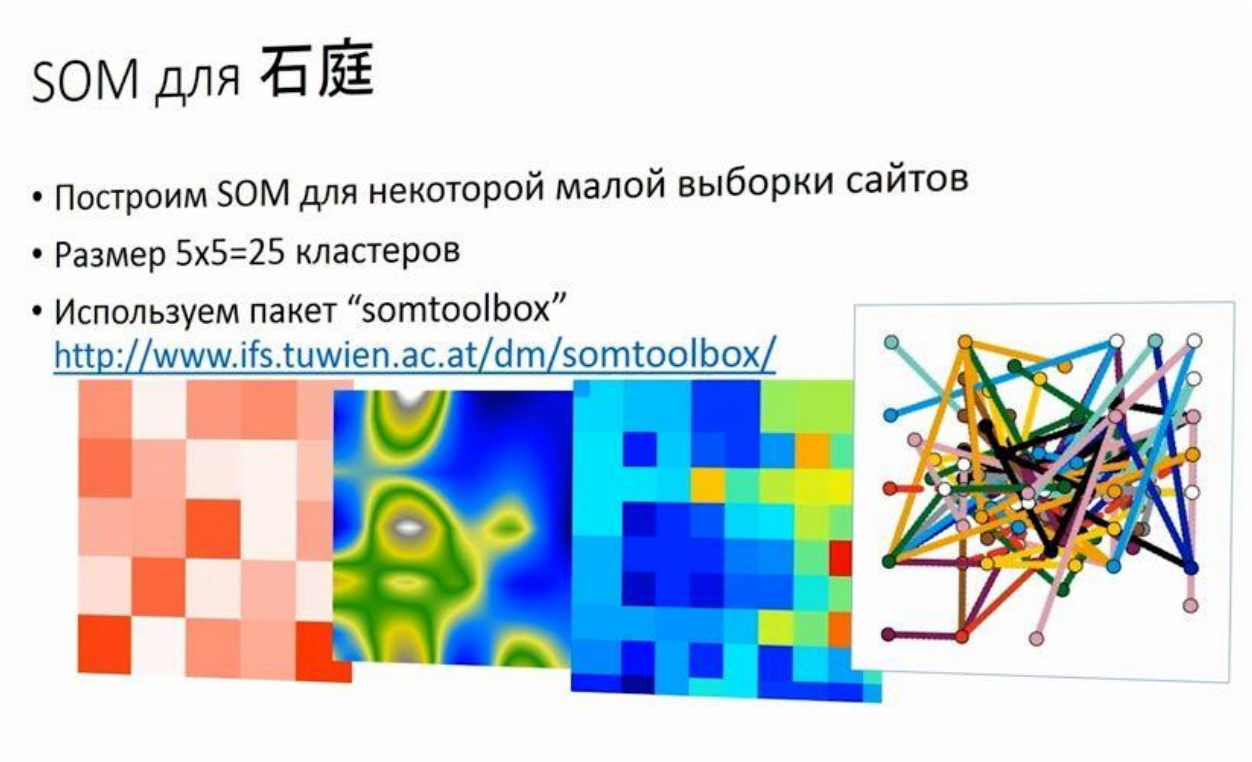
$$\eta_0 = 0.1, \ \tau_2 = 1000,$$

таким образом, достигается достаточно медленное убывание $\eta(t)$, при котором большую часть времени $\eta(t) > 0.01$, чтобы избежать ранней сходимости процесса обучения. Обратите внимание на связь между τ_2 и параметром T, используемым для настройки τ_1! В общем случае можно считать их равными.

В результате обучения веса нейронов сходятся к координатам центров скоплений точек данных. При этом алгоритм обучения гарантирует, что ни у каких двух нейронов не будет одинаковых векторов весов. Можно также показать, что чем больше скопление точек, тем большим количеством нейронов оно будет описываться.

Например, если в обучающий набор входили точки, равномерно распределенные в окружности с центром в начале координат и единичным радиусом, то и веса нейронов карты Кохонена после обучения будут «равномерно» распределены в этой же окружности, при этом их координаты (веса связей) могут и не совпадать с исходными точками[3].

[3] Слово «равномерно» взято в кавычки, т.к. в общем случае это условие должно выполняться только в пределе $t \to +\infty$.



Алгоритм пакетного обучения

Идея пакетного обучения совпадает с таковой для последовательного, однако формулы для обновления весов связей для *j*-го нейрона по пакетному алгоритму существенно отличаются:

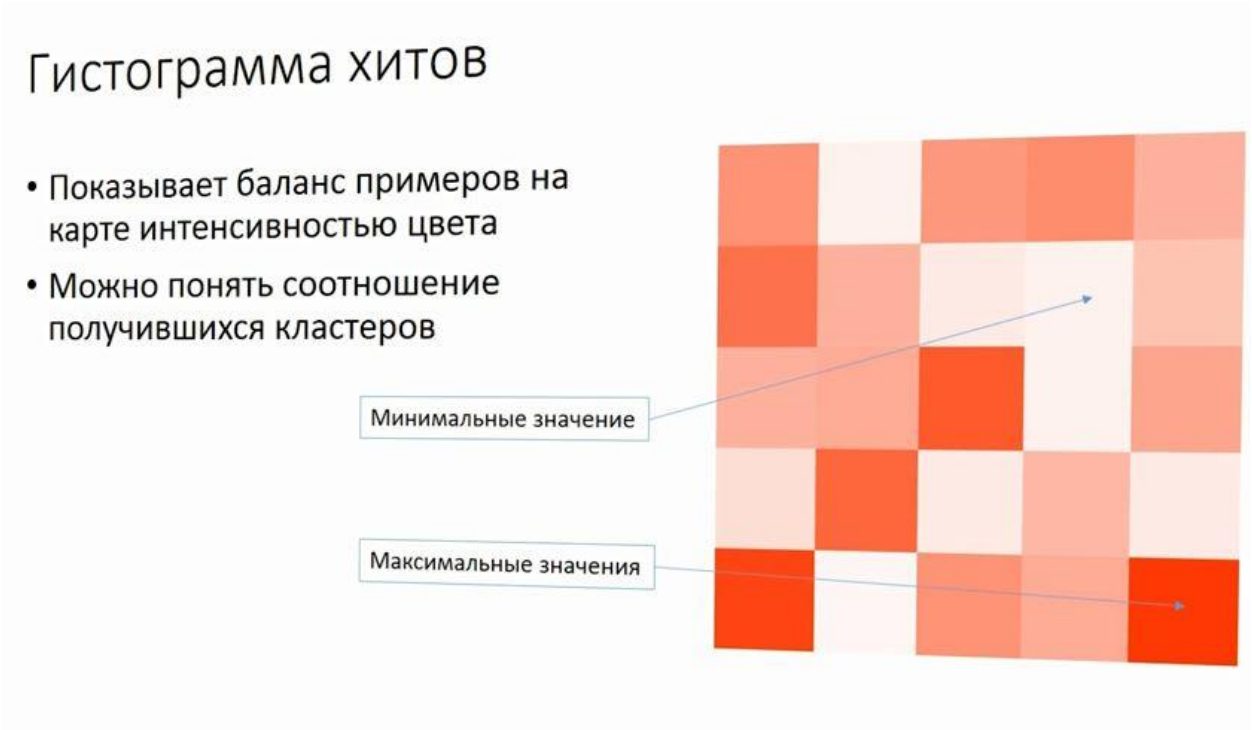
$$\mathbf{w}_j(t+1) = \frac{\sum_{i \in O_j} n_i h_{ij} \bar{\mathbf{x}}_i}{\sum_{i \in O_j} n_i h_{ij}}, \qquad (3)$$

где O_j – окрестность *j*-го нейрона, определяемая параметром σ; n_i – количество входных векторов, для которых *i*-й нейрон оказался победителем; $\bar{\mathbf{x}}_i = \frac{1}{n_i} \sum_k [winner_k = i] \mathbf{x}_i$ – среднее значение входных векторов, для которых победителем оказался *i*-й нейрон, здесь $[winner_k = i]$ – индикаторная функция:

$$[winner_k = i] = \begin{cases} 1, & winner_k = i \\ 0, & winner_k \neq i \end{cases}$$

Выбор вида h_{ij} и ее параметров совпадает с рекомендациями для алгоритма последовательного обучения.

Алгоритм пакетного обучение сходится гораздо быстрее алгоритма последовательного обучения, однако каждый шаг требует б*о*льших затрат по вычислениям и памяти. Отметим, что обновление весов по формуле (3) может быть эффективно распараллелено, т.к. веса нейронов обновляются независимо.



«Мертвые» нейроны

В результате обучения карты Кохонена могут появиться «мертвые» нейроны, т.е. нейроны, которые не являются победителями ни для одного входного вектора. Такие нейроны не описывают ни одно скопление входных векторов и часто их можно отбросить.

Обнаружение мертвых нейронов можно производить после обучения карты Кохонена. Для этого каждого нейрона подсчитывается, сколько раз он оказался победителем после предъявления всех обучающих векторов.

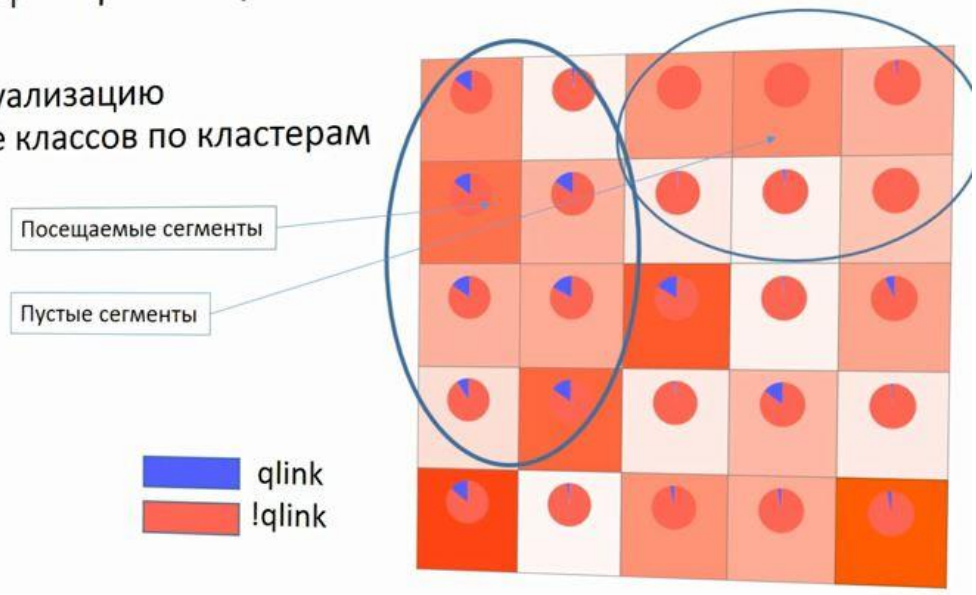
Поскольку обученная карта Кохонена представляет огрубленное описание исходных данных, то имеется ошибка представления этих данных. Данную ошибку можно вычислить:

$$err = \sum_i \left\| \mathbf{w}_{winner_i} - \mathbf{x}_i \right\|^2 .$$

Процесс распознавания можно представить графически, так как современная вычислительная техника предоставляет широкие возможности по визуализации информационных процессов.

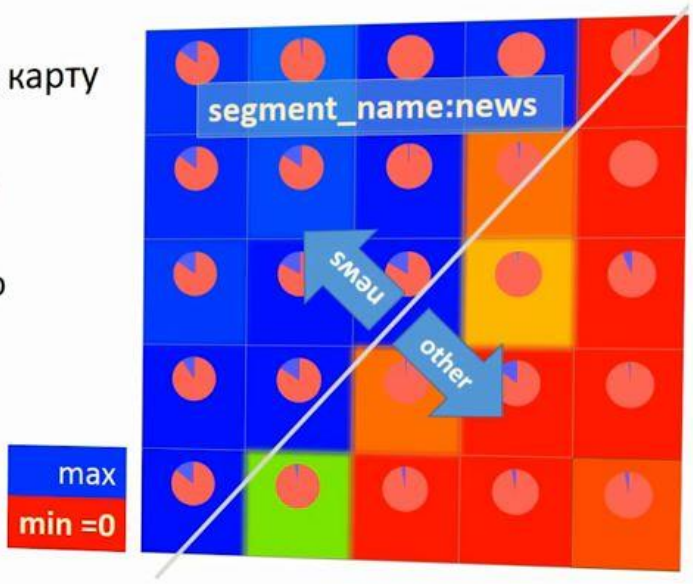
Наложим диаграмму классов

- Получаем визуализацию распределение классов по кластерам



Сделаем Карты признаков

- Для каждого признака построим карту его влияния
- Интенсивностью цвета выделим уровень значения признака
- Можно изучить влияние каждого признака на кластера



Список литературы

1. Kohonen T., Honkela T. Kohonen Network, Scholarpedia, 2007, vol. 2, no. 1, p. 1568.
2. Хайкин С. Нейронные сети: полный курс, 2-е изд., испр. : Пер. с англ. – М. : 000 "И.Д. Вильяме", 2006. – 1104 с. : ил. – Парал. тит. англ.
3. Журавлёв Ю. И., Никифоров В. В. Алгоритмы распознавания, основанные на вычислении оценок // Кибернетика. — 1971.— № 3.
4. Лбов Г. С. Методы обработки разнотипных экспериментальных данных. — Новосибирск: Наука, 1981.
5. Норушис А. Построение логических (древообразных) классификаторов методами нисходящего поиска (обзор) // Статистические проблемы управления. Вып. 93 / Под ред. Ш. Раудис. — Вильнюс, 1990.— С. 131–158.
6. Полякова М. П., Вайнцвайг М. Н. Об использовании метода «голосования» признаков в алгоритмах распознавания // Моделирование обучения и поведения. — М., 1975.— С. 25—28.
7. Martin J. K. An exact probability metric for decision tree splitting and stopping // Machine Learning.— 1997.— Vol. 28, no. 2-3.— Pp. 257–291. http://citeseer.ist.psu.edu/martin97exact.html.

8. Quinlan J. Induction of decision trees // Machine Learning.— 1986.— Vol. 1, no. 1.— Pp. 81–106.

9. Rivest R. L. Learning decision lists // Machine Learning.— 1987.— Vol. 2, no. 3.— Pp. 229–246.

10.