электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.67

Анализ данных на основе алгоритмов класса SOINN

Федоренко Ю. С., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

Научный руководитель: Гапанюк Ю. Е., к.т.н, доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»
gapyu@bmstu.ru

1. Введение

На практике часто возникают задачи, в которых имеется набор многомерных данных (причем нередко больших размеров), а априорная информация о механизме генерации этих данных отсутствует или является неполной. В таких случаях из-за отсутствия модели данных невозможно применять классические алгоритмы обучения с учителем (например, классификации или регрессии). Как известно, в этой ситуации исследователю помогает, так называемый разведочный анализ данных [1]. Он позволяет получить приблизительное описание структуры данных, отталкиваясь от которого можно более детально исследовать данные с помощью тех или иных методов статистического анализа. Однако использование такого подхода достаточно ограничено на практике.

Например, при анализе поведения пользователей в Интернете приходится иметь дело со значительными массивами данных, поступающих в реальном времени. Это информация о кликах по рекламным баннерам, наборы посещенных пользователями сайтов, оценки тех или иных товаров или продуктов, набор покупок, совершенных пользователем (может отслеживаться по корзинам Интернет магазинов) и т.д. Постоянно поступают гигабайты данных, о которых нет никакой априорной информации. Использование классических алгоритмов разведочного анализа данных осложняется высокой скоростью поступления новых «сырых» данных и изменением их статистик во времени. Фактически, в подавляющем большинстве практических задач приходится иметь дело с нестационарными данными, т.е. такими данными, в которых статистики изменяются во времени по мере поступления новых данных [2]. Последнее обстоятельство требует создания адаптивных или онлайновых моделей, которые могли бы адекватно

отслеживать изменения свойств объекта анализа. Кроме того, желательно делать это быстро, поскольку данные поступают в больших объемах.

Таким образом, в ряде практических задач встает проблема построения моделей на основе данных, меняющихся в реальном времени, причем требуется, чтобы модель была способна обучаться новым данным, не забывая при этом старую информацию. Для решения данной проблемы могут успешно применяться алгоритмы SOINN (Self-organizing incremental neural network - самоорганизующаяся растущая нейронная сеть), особенности которой рассматриваются в данной статье.

2. Задача представления топологии входных данных. Непараметрическая оценка плотности распределения вероятности

Для создания моделей поступающих данных необходимо представить их топологическую структуру распределения. Данная задача, как правило, решается построением некоторой оценки плотности распределения вероятности, математическая постановка которой выглядит следующим образом [3]: требуется оценить плотность распределения *p(x)* по выборке $X_l = \{x_i\}_{i=1}^m$ независимых случайных векторов, распределенных по этому закону *p(x).*

Существует множество подходов к решению задачи восстановления плотности распределения, основными из которых являются параметрический, непараметрический и способ восстановление смеси распределений (на данном подходе основан EM-алгоритм). К параметрическим способам относится метод максимального правдоподобия и некоторые другие подходы. Рассмотрим более подробно непараметрический подход к восстановлению плотности распределения вероятности, т.к. именно он лежит в основе алгоритмов класса SOINN.

Вероятность того, что вектор x, взятый из распределения p(x) попадет в заданный регион R пространства исходных образов, равна следующему [4]:

$$P = \int_R p(x')dx' \cong p(x) \cdot V, \qquad (1)$$

где V – объем, ограничиваемый регионом R.

Возьмем из заданного распределения N векторов $\{x^1, x^2, ..., x^N\}$. Тогда вероятность, что k этих векторов попадут в R, определяется на основе биномиального распределения:

$$P(k) = \binom{N}{k} P^k (1-P)^{N-k}$$

Таким образом, доля векторов k/N, попавших в R, является случайной величиной, распределенной по закону Бернулли. Для биномиального распределения можно записать значения математического ожидания и дисперсии:

$$M\left[\frac{k}{N}\right] = P, \quad D\left[\frac{k}{N}\right] = M\left[\left(\frac{k}{N} - P\right)^2\right] = \frac{P(1-P)}{N} \qquad (2)$$

При увеличении числа N получаем, что дисперсия стремится к 0, поэтому можно записать, что:

$$\frac{k}{N} \cong P \qquad (3)$$

Учитывая равенства (3) и (1), получаем:

$$P \cong \frac{k}{N} \cong p(x) \cdot V \Rightarrow p(x) \cong \frac{k}{NV}, \qquad (4)$$

где V – объем окрестности около точки x, N – полное количество примеров, k – количество примеров из N, попавших в объем V. Чем больше мы увеличиваем размер выборки N и сокращаем объем V, тем точнее получается данная оценка. Зафиксируем объем V, и будем определять k на основе данных. В этом и заключается суть подхода ядерной оценки плотности, используемой в SOINN.

3. Ядерная оценка плотности Парзена-Розенблата

Предположим, что регион R является гиперкубом с длиной стороны h, в центре которого находится образец x. Тогда объем такого куба можно определить как h^D, где D – количество измерений. Для определения числа образцов, попавших с этот куб, вводится понятие ядерной функции (парзеновское окно) [5]:

$$K(u) = \begin{cases} 1 & |u_j| < 1/2 \quad \forall j = \overline{1, D} \\ 0 & \textit{иначе} \end{cases} \qquad (5)$$

Тогда количество образцов k, попавших внутрь гиперкуба равно:

$$k = \sum_{n=1}^{N} K\left(\frac{x - x^n}{h}\right)$$

Подставляя данное выражение в (4), получаем:

$$p(x) = \frac{1}{N \cdot h^D} \sum_{k=1}^{N} K\left(\frac{x - x^n}{h}\right) \qquad (6)$$

Данный подход к оценке плотности распределения вероятностей применяется и в алгоритмах класса SOINN.

4. Основная идея алгоритмов класса SOINN

Как уже упоминалось, математически алгоритмы SOINN основаны на ядерной оценке плотности распределения вероятностей. Для поступающих входных данных строится граф таким образом, чтобы вершины данного графа попадали в локальные максимумы плотности распределения вероятности (рис. 1)

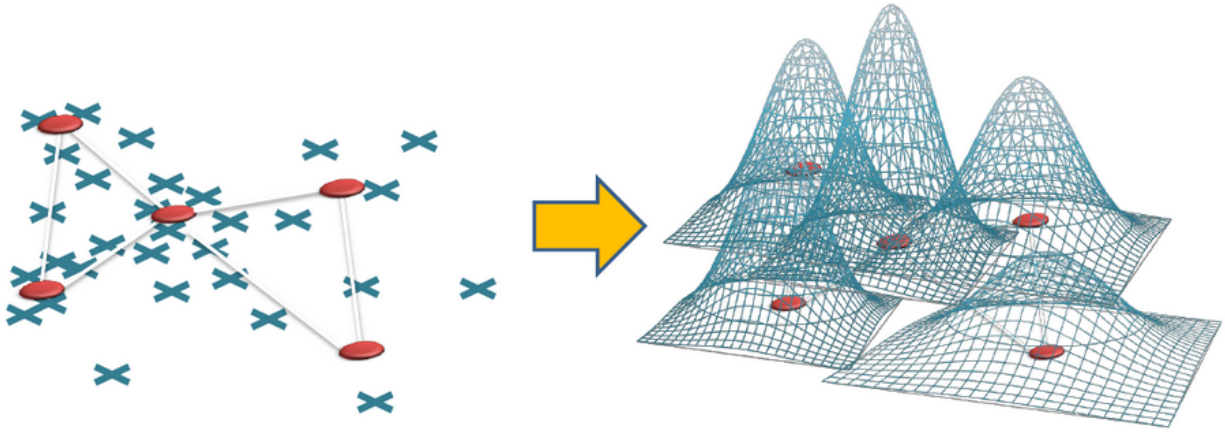

Рис. 1. Граф, строящийся на основе входных данных (адаптировано из [6])

Полученный граф представляет собой некоторую модель исходных данных, которую можно использовать при дальнейшей работе. Данный граф может быть использован для оценки плотности распределения вероятности исходных данных. Выражение для оценки плотности распределения вероятности имеет вид [6, 8]:

$$\hat{p}(x) = \sum_{n \in N} \frac{t_n}{T_N} K_{Cn}(x - \omega_n), \qquad (7)$$

где N – количество нейронов построившейся сети, t_n - локальное число сигналов нейрона (количество раз, которое данный нейрон был победителем), T_N - общее число сигналов (образцов), обработанных сетью, ω_n - весовой вектор n-го нейрона, C_n - номер кластера, к которому относится n-й нейрон. В качестве ядерной функции K_{Cn} может использоваться различные функции, например, парзеновское окно (5) или Гауссово ядро:

$$K(u) = \frac{1}{\sqrt{2\pi}} \cdot e^{-u^2/2}. \qquad (8)$$

Таким образом, при помощи SOINN можно восстанавливать плотность вероятности разных классов. По известным плотностям вероятности классов можно решить задачу классификации или регрессии, основываясь на байесовском подходе [7].

5. Алгоритм SOINN

Самоорганизующаяся растущая нейронная сеть (SOINN), прежде всего, предназначена для решения задач обучения без учителя. Важным преимуществом данного алгоритма является то, что он не требует никаких априорных знаний о предметной

области (например, число кластеров). Изначально задаются только параметры самой сети. SOINN является двухслойной сетью, ниже приведен алгоритм обучения первого слоя SOINN [9].

1. Инициализировать 2 узла нейронной сети (из исходных данных). Связь между узлами не добавлять. В качестве весовых коэффициентов этих узлов берутся координаты соответствующих точек.

2. Подать на вход алгоритма произвольный вектор $\bar{x}$.

3. Найти 2 нейрона s1 и s2, ближайшие к $\bar{x}$. Обычно в качестве меры расстояния выбирается Евклидова метрика, однако, в качестве меры расстояния может быть также использована мера Жаккара, косинусный коэффициент и т. д. Дальнейшее описание алгоритма будет приведено для Евклидовой меры. При этом если расстояние между x, s1 или x, s2 больше порогов подобия Ts1, Ts2, то добавить новый узел и перейти на шаг 2. Иначе – перейти на шаг 4. Таким образом, вставка нового узла осуществляется, если:

$$\left[\begin{array}{l} d(x,s1) > Ts1 \\ d(x,s2) > Ts2 \end{array}\right..$$

Иначе – вставка нового узла не производится.

4. Создать ребро между узлами s1, s2, если оно не существует. Установить возраст ребра в 0.

5. Увеличить на 1 возраст дуг, исходящих от s1.

6. Обновить ошибку нейрона-победителя (нейрон, для которого расстояние до входного вектора $\bar{x}$ минимально):

$$E_{S1} = E_{S1} + \left\| \overline{\omega_{S1}} - \bar{x} \right\|^2,$$

где E_{S1} — ошибка нейрона-победителя s1, $\overline{\omega_{S1}}$ — вектор весов нейрона s1.

7. Увеличить на 1 локальное число накопленных сигналов узла s1: Ms1 = Ms1 + 1.

8. Сместить нейрон s1 и всех его соседей:

$$\begin{array}{l} \overline{\omega_{S1}} = \overline{\omega_{S1}} + \varepsilon_{\omega}(\overline{\omega_{S1}} - \bar{x}) \\ \overline{\omega_n} = \overline{\omega_n} + \varepsilon_n(\overline{\omega_n} - \bar{x}) \end{array},$$

где $\varepsilon_{\omega}, \varepsilon_n$ — коэффициенты обучения нейрона-победителя и его соседей соответственно. В классическом алгоритме $\varepsilon_w = \frac{1}{t}$, $\varepsilon_n = \frac{1}{100t}$, где t – накопленное число сигналов узла, т.е. t = Mi.

9. Удалить все дуги с возрастом, превышающим $age_{\max}$ (параметр алгоритма).

10. Если номер итерации кратен $\lambda_{\max}$, то осуществить вставку нового узла и удаление неиспользуемых узлов по следующим правилам:

— найти нейрон u с максимальной локальной ошибкой;

— среди соседей u найти нейрон v с максимальной ошибкой;

— создать узел r с вектором весов равным $\overline{\omega_r} = \dfrac{\overline{\omega_u} + \overline{\omega_v}}{2}$;

— рассчитать накопленную ошибку нового узла Er, суммарное количество сигналов Mr и наследованный радиус ошибки:

$$\begin{aligned} Er &= \alpha_1 \cdot (Eu + Ev) \\ Mr &= \alpha_2 \cdot (Mu + Mv); \\ Rr &= \alpha_3 \cdot (Ru + Rr) \end{aligned}$$

— уменьшить суммарную ошибку узлов u и v, а также их накопленное число сигналов:

$$\begin{aligned} Eu &= \beta \cdot Eu, \quad Ev = \beta \cdot Ev \\ Mu &= \gamma \cdot Mu, \quad Mv = \gamma \cdot Mv \end{aligned};$$

— определить, успешно прошла ли вставка нового узла. Вставка считается успешной, если она уменьшает погрешность области. Т.е. если:

$$\exists i : \frac{E_i}{M_i} > R_i, \; i \in u, v, r \;,$$

то вставка считается неуспешной и значения всех параметров восстанавливаются;

— если вставка прошла успешно, то обновляем радиусы ошибок узлов, задействованных во вставке:

$$\begin{aligned} Ru &= Eu / Mu \\ Rv &= Ev / Mv \\ Rr &= Er / Mr \end{aligned}$$

и заменяем связь (u, v) на (u, r) и (v, r)

— для узлов, которые имеют одного соседа, сравнить накопленное количество входных сигналов узла со средним значением по всем узлам. Если

$$Mi < c \cdot \sum_{j=1}^{Na} Mj / Na \,,$$

то удалить i-й узел. При этом c – параметр алгоритма, $1 \le c \le 0$. Если исходные данные зашумлены, то имеет смысл выставлять c = 1.

— удалить все изолированные узлы.

Перейти к пункту 2 и продолжить обучение.

После обучения первого слоя начинается процесс обучения второго слоя. Следует отметить, что алгоритм не дает каких-либо критериев, позволяющих понять, что пора завершать обучение первого слоя и начинать обучение второго слоя, это определяется самим исследователем. Обычно обучение второго слоя можно начинать после нескольких циклических прогонов по обучающей выборке.

Алгоритм обучения второго слоя аналогичен алгоритму обучения первого слоя, различие заключается лишь в формировании порога, который используется на 3-ем шаге алгоритма.

Для первого слоя порог подобия T вычисляется адаптивно следующим образом [9]:

1. Для новых узлов порог подобия устанавливается в +∞.

2. Когда узел становится первым или вторым победителем, обновляются значения порога подобия:

- если узел имеет прямых топологических соседей, то в качестве порога выбирается максимальное расстояние между узлом и его соседями (внутрикластерное расстояние):

$$Ti = \max_{c \in Ni} \left\| Wi - Wc \right\|;$$

- если узел не имеет соседей, то в качестве порога выбирается минимальное расстояние между узлом и остальными узлами нейросети (межкластерное расстояние):

$$Ti = \min_{c \in A \setminus \{i\}} \left\| Wi - Wc \right\|.$$

Для второго слоя порог подобия остается постоянным, и он вычисляется следующим образом:

1. Вычисляем среднее внутрикластерное расстояние:

$$dW = \frac{1}{Nc} \sum_{(i,j) \in C} \left\| Wi - Wj \right\|.$$

2. Устанавливаем значение порога Tc как минимальное межкластерное расстояние:

$$Tc = \min_{k,l=1,\ldots,Q, k \neq l} d_b(C_k, C_l).$$

3. Увеличиваем значение порога Tc (постепенно выбирая следующие по величине межкластерные расстояния) до тех пор, пока Tc не превысит значения внутрикластерного расстояния dW.

Таким образом, алгоритм построения SOINN можно представить следующим образом (рис. 2):

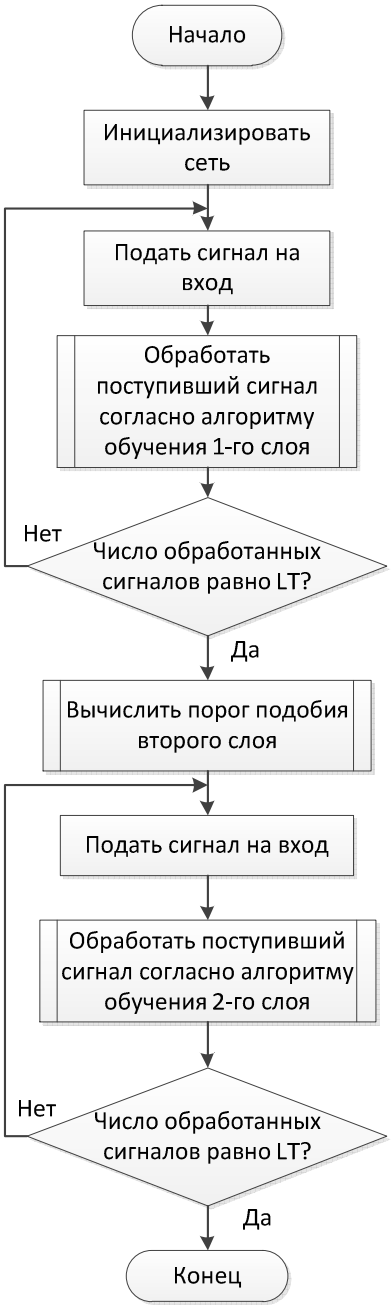

Рис. 2. Обобщенная блок схема алгоритма SOINN

6. Особенности SOINN

Алгоритмы класса SOINN реализуют два основных принципа: представление топологической структуры данных и инкрементальное обучение. Для представления топологической структуры данных могут использоваться сети Кохонена и растущий нейронный газ, хотя они обладают определенными недостатками (сети Кохонена требуют заранее задавать количество кластеров, а число нейронов в растущем нейронном газе постоянно увеличивается, из-за чего сеть через определенное время разрастается, и ее использование становится затруднительным). Также существуют алгоритмы, реализующие инкрементальное обучение (например, онлайновая логистическая регрессия

или ART-сети). Для реализации инкрементального обучения необходимо, чтобы сеть могла усваивать новые данные, не забывая при этом старые. Особенность алгоритмов класса SOINN состоит в том, что они реализуют оба данных принципа. Поэтому можно выделить следующие достоинства SOINN:

1. Нейроны сети организуются в определенную структуру, представляя топологию входных данных, без априорного задания сетевой структуры или размера сети.

2. Количество нейронов ищется адаптивно по мере обучения сети.

3. Сеть реализует возрастающее онлайн обучение без предварительных условий.

4. Алгоритм является устойчивым к шуму.

Таким образом, SOINN может представлять топологию входных данных, производить возрастающее обучение и автоматически подбирать нужное количество узлов, устраняя шум. Однако у самоорганизующейся растущей нейронной сети остаются некоторые нерешенные проблемы:

1. Сеть состоит из двух слоев, вследствие чего принцип онлайнового обучения не был полностью реализован. При поступлении новых данных на первый слой, второй слой необходимо полностью переобучать.

2. Алгоритм не способен разделять перекрывающиеся кластеры во входных данных.

3. Алгоритм содержит множество настраиваемых параметров, при неправильном подборе которых сеть ведет себя достаточно нестабильно.

Для устранения данных недостатков была предложена архитектура ESOINN (улучшенная самоорганизующаяся растущая нейронная сеть), которая состоит из одного слоя, содержит алгоритм для разделения перекрывающихся классов и имеет меньшее число настраиваемых параметров [10].

7. SOINN и MCL

На выходе самоорганизующейся растущей нейронной сети получается неориентированный граф, вершинами которого являются нейроны, а дугами – связи между ними. В оригинальной работе для нахождения кластеров предлагается использовать алгоритм выделения связных компонент в графе [9]. Однако последнее приводит к тому, что при наличии хотя бы одной связи между вершинами, представляющими разные кластера, данные кластера будут объединены в один кластер.

Поэтому оказывается целесообразным использование Марковского алгоритма кластеризации (MCL) [11] для выделения кластеров в графе. Описание данного алгоритма, способ его применения для проведения кластеризации в графе, преимущества и недостатки такого подхода подробно описаны в [12].

8. Эксперименты

Для анализа качества работы SOINN было произведено сравнение сети Кохонена и комбинации алгоритмов SOINN и MCL. При этом для сети Кохонена заранее задавалось число кластеров, SOINN и MCL определяли их автоматически. Для экспериментов были взяты 3 синтетических тестовых набора:

1. S-набор, состоящий из 15 кластеров [13]. В наборе в основном присутствуют концентрические кластеры с перекрытиями, кроме того, присутствуют 2 кластера в форме полосы (рис. 3).

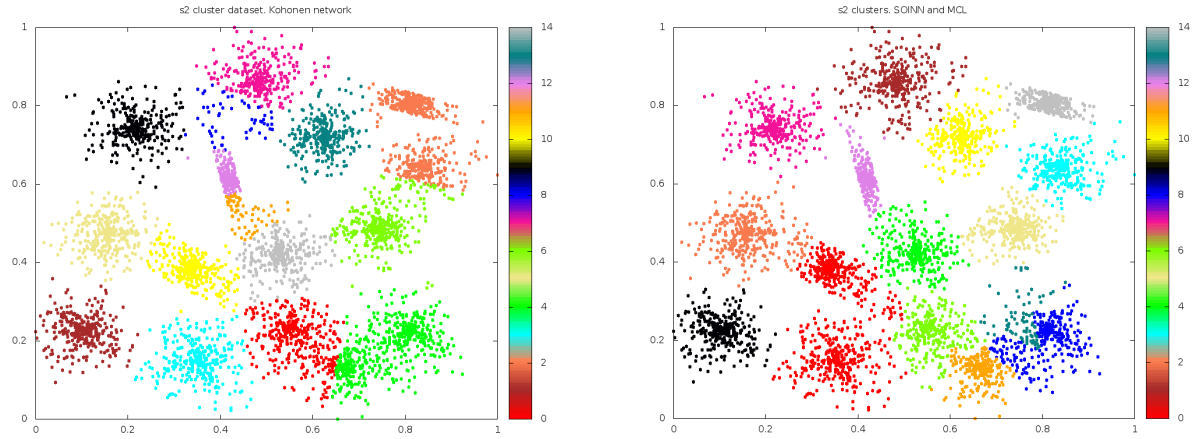

Рис. 3. Кластеризация S-набора при помощи сетей Кохонена (слева) и связки SOINN и MCL (справа)

2. Jain's Toy Problem [14]. Данный набор состоит из двух кластеров: одного в форме «шапки», другого – в форме полукруга (рис. 4).

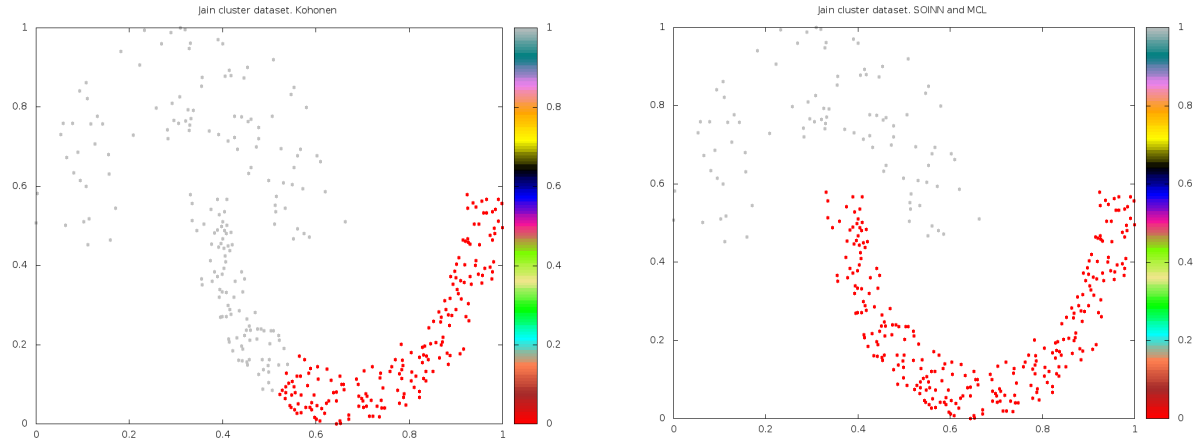

Рис. 4. Кластеризация Jain's Toy Problem при помощи сетей Кохонена (слева) и связки SOINN и MCL (справа)

3. Aggregation set [15]. Данный набор состоит из 7 кластеров, в основном в форме эллипса (рис. 5).

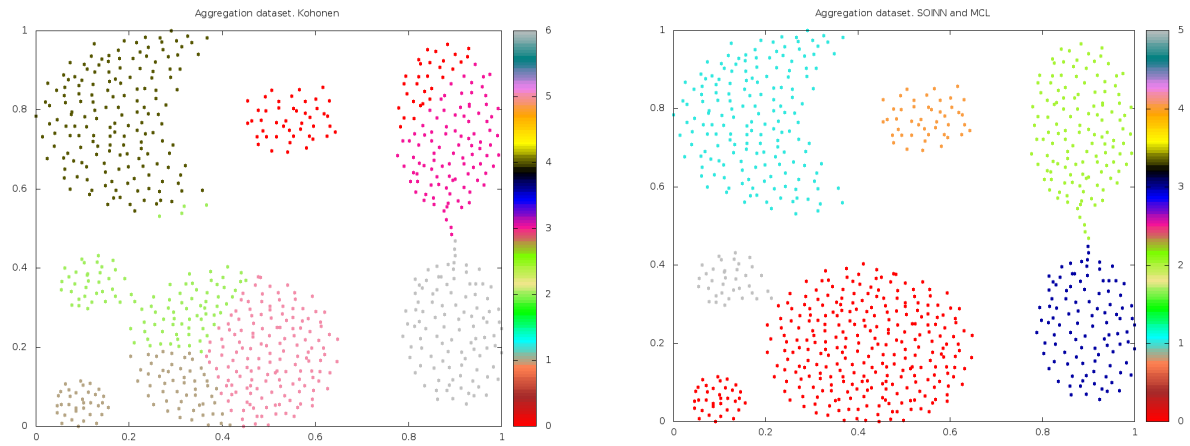

Рис. 5. Кластеризация Aggregation set при помощи сетей Кохонена (слева) и связки SOINN и MCL (справа)

По результатам эксперимента можно сделать следующие выводы:

1. Сети Кохонена (равно как и многие другие «базовые» алгоритмы кластеризации) способны разделять только концентрические кластеры. Например, на рис. 3 сеть Кохонена не смогла выделить кластеры в виде полосы, а на рис. 4 и 5 сеть Кохонена вновь выделяет только концентрические кластеры, из-за чего наблюдается много очевидных ошибок. Кроме того, сеть Кохонена плохо разделяет зашумленные кластеры, что видно из рис. 4.

2. Связка алгоритмов SOINN и MCL автоматически определяет число кластеров (причем в первых двух случаях она сделала это практически правильно). При этом данный алгоритм способен находить кластеры произвольной формы. Однако для третьего набора данных алгоритм не смог разделить 2 нижних кластера. Возможно, это связано с малым количеством точек. К недостаткам алгоритмов SOINN и MCL можно отнести их нестабильность. Для достижения хорошего результата работы следует точно подобрать все параметры, что непросто сделать на практике. При неверном выборе параметров алгоритм может работать достаточно странным образом (рис. 6).

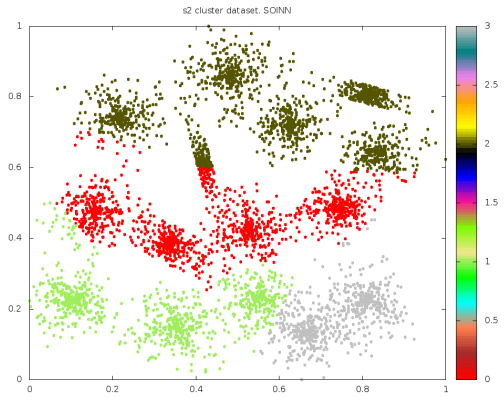

Рис. 6. Кластеризация S-набора при помощи SOINN при неправильном выборе параметров

Алгоритм ESOINN [10] уменьшает данную проблему. Он содержит меньшее число параметров и работает более стабильно.

9. Заключение

Самоорганизующиеся растущие нейронные сети (SOINN) позволяют описывать топологию входных данных, оценивать плотность распределения вероятности классов, учитывая все данные поступающие в реальном времени. Оценка плотности распределения вероятности классов при помощи SOINN основывается на ядерной оценке плотности. Оценив плотности распределения классов, можно решать задачу классификации данных или регрессии на основе байесовского подхода. Также SOINN успешно решает задачи обучения без учителя, например, задачу кластеризации.

Важной особенностью алгоритмов класса SOINN является отсутствие необходимости заранее выдвигать статистические гипотезы для работы с данными и возможность адаптивного построения модели по мере поступления данных в реальном времени. В задачах, где постоянно появляются новые данные, о которых заранее ничего не известно, данная особенность является важным преимуществом.

Предложенная в статье техника кластеризации показала свою эффективность в проведенных экспериментах. Связка алгоритмов SOINN и MCL позволяет находить кластеры произвольной формы и перекрывающиеся кластеры. При этом сложность алгоритма можно регулировать, ограничивая количество нейронов SOINN. К недостаткам алгоритма можно отнести сложный процесс настройки и неустойчивость. Для преодоления данных недостатков можно использовать улучшенную самоорганизующуюся нейронную сеть (ESOINN).

Список литературы

1. Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. Прикладная статистика: классификация и снижение размерности / под ред. С. А. Айвазяна. Справ. изд. М.: Финансы и статистика, 1989. 608 с.
2. Лабунец Л. В., Лабунец Н. Л., Чижов М. Ю. Рекуррентные статистики нестационарных временных рядов // Радиотехника и электроника. 2011. № 12. С. 1468-1489.
3. Воронцов К. В. Математические методы обучения по прецедентам: курс лекций, 2013 г. Режим доступа: http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf (дата обращения 07.03.15).

4. Gutierrez-Osuna R. L7: Kernel density estimation. Available at: http://research.cs.tamu.edu/prism/lectures/pr/pr_l7.pdf, accessed 07.03.15.
5. Дуда Р., Харт П. Распознавание образов и анализ сцен: пер. с англ. Г. Г. Вайнштейна, А. М. Васьковского / под ред. В. Л. Стефанюка. М.: Мир, Москва, 1976. 509 с. [Duda R., Hart P. Pattern classification and scene analysis. Stanford Research Institute, Menlo Park, California, 1973. 509 p.].
6. Hasegawa O. SOINN: An Artificial Brain, 2014. Available at: http://watanabe-www.math.dis.titech.ac.jp/syllabus/atmis/Hasegawa(atmis)1.pdf, accessed 07.03.15.
7. Воронцов К. В. Статистические (байесовские) методы классификации, 2013. Режим доступа: http://www.machinelearning.ru/wiki/images/9/98/Voron-ML-Bayes-slides.pdf (дата обращения 07.03.15).
8. Xiao X., Zhang H., Hasegawa O. Density Estimation Method Based on Self-Organizing Incremental Neural Network and Error Estimation // The Neural Information Processing: 20th International Conference, ICONIP 2013: proceedings. Daegu, Korea, 2013. P. 43-50.
9. Furao S., Hasegawa O. An incremental network for on-line unsupervised classification and topology learning // Neural Networks. 2005. № 4. P. 1-17.
10. Furao S., Ogura T., Hasegawa O. An enhanced self-organizing incremental neural network for online unsupervised learning // Neural Networks. 2007. № 6. P. 893-903.
11. Stijn Van Dongen, Cei Abreu-Goodger. Using MCL to extract clusters from networks. Bacterial Molecular Networks: Methods and Protocols, Methods in Molecular Biology. Vol. 804. Humana Press, Totowa, New Jersey, USA, 2012. P. 181–195.
12. Федоренко Ю. С. Кластеризация данных на основе нейронного газа и марковских алгоритмов // Молодежный научно-технический вестник. МГТУ им. Н. Э. Баумана. Электрон. журн. 2014. № 8. Режим доступа: http://sntbul.bmstu.ru/doc/730611.html (дата обращения 07.03.15).
13. Franti P., Virmajoki O. Iterative shrinking method for clustering problems // Pattern recognition. 2006. № 5. P.761-784.
14. Jain A., Law M. Data clustering: a user's dilemma // Lecture Notes in Computer Science. 2005. Vol. 3776. P. 1-10.
15. Gionis A., Mannila H., Tsaparas P. Clustering aggregation // ACM Transactions of Knowledge Discovery from Data. 2007. № 1. P. 1-30.