УДК 004.8:004.032.26

Бодянский Е.В.[1], Шафроненко А.Ю.[2], Патлань Е.В.[3]
[1]ХНУРЕ, г.Харьков, Украина, yevgeniy.bodyanskiy@nure.ua
[2]ХНУРЕ, г.Харьков, Украина, alina.shafronenko@nure.ua
[3]НТУ «Харьковский политехнический институт»,
г. Харьков, Украина, sonnyanishakal@gmail.com

НЕЧЕТКАЯ КЛАСТЕРИЗАЦИЯ МАССИВОВ ДАННЫХ НА ОСНОВЕ ЭВОЛЮЦИОННОГО МЕТОДА ОПТИМИЗАЦИИ КОШАЧЬИХ СТАЙ

Рассмотрена задача нечеткой кластеризации массива наблюдений на основе нечеткого вероятностного подхода, в основу которого положен алгоритм нечетких С-средних, переформулированный в задачу безусловной многоэкстремальной оптимизации.

НЕЧЕТКАЯ КЛАСТЕРИЗАЦИЯ, МНОГОЭКСТРЕМАЛЬНАЯ ОПТИМИЗАЦИЯ, ЭВОЛЮЦИОННЫЙ АЛГОРИТМ

Є.В. Бодянський, А.Ю. Шафроненко, Е.В. Патлань. Нечітка кластеризація масивів даних на основі еволюційного метода оптимізації кошачих зграй. Розглянуто задачу нечіткої кластеризації масиву спостережень на основі нечіткого імовірнісного підходу, в основу якого покладено алгоритм нечітких С-середніх, який було переформулювало в задачу безумовної багатоекстремальної оптимізації.

НЕЧІТКА КЛАСТЕРИЗАЦІЯ, БАГАТОЕКСТРЕМАЛЬНА ОПТИМІЗАЦІЯ, ЕВОЛЮЦІЙНИЙ АЛГОРИТМ

Ye. Bodyankiy, A. Shafronenko, K. Patlan. Fuzzy clustering of data arrays based on the evolutionary method of cat swarm optimization. The problem of fuzzy clustering of an array of observations based on a fuzzy probabilistic approach, based on the algorithm of fuzzy C-means, reformulated into the problem of unconditional multiextremal optimization, is considered.

FUZZY CLUSTERING, MULTIEXTREMAL OPTIMIZATION, EVOLUTIONARY ALGORITHM

Введение

Задача кластеризации массивов многомерных данных, основной целью которой является нахождение однородных в смысле принятой метрики классов наблюдений, является важной частью интеллектуального анализа данных Data Mining [1-3], интенсивно развивающегося в настоящее время. В рамках традиционного кластерного анализа априори предполагается, что каждый вектор-наблюдение может принадлежать только одному классу-кластеру, хотя в реальных приложениях достаточно часто возникает ситуация, когда это наблюдение с разными уровнями принадлежности (возможности, вероятности) относится сразу к нескольким взаимно перекрывающимся кластерам. Подобная ситуация является предметом рассмотрения нечеткого (фаззи-) кластерного анализа [4-6], в рамках которого необходимо оценить не только факт принадлежности каждого наблюдения к конкретным классам, но и дать количественную оценку уровня этой принадлежности.

Исходной информацией для решения задачи нечеткой кластеризации является массив многомерных векторов-данных, образованный выборкой наблюдений $X = \{x(1), x(2), ..., x(k), ..., x(N)\} \subset R^n$, где k — в общем случае номер наблюдения в исходном массиве, $x(k) = (x_1(1), x_2(2), ..., x_i(k), ..., x_n(k)\}^T$. Результатом кластеризации является разбиение этого массива на m пересекающихся классов Cl_j с прототипами — центроидами $c_j \in R^n, j = 1,2,...,m$, при этом наряду с нахождением центроидов c_j должен быть оценен уровень принадлежности $0 < U_j(k) < 1$ каждого $x(k)$ к каждому из кластеров Cl_j.

Заметим, что исходные данные должны быть предобработаны (центрированы, нормированы, кодированы, стандартизированы) так, чтобы все наблюдения принадлежали либо некоторому гиперкубу (обычно $[-1,1]^n$ или $[0,1]^n$), либо лежали на гиперсфере с единичным радиусом. В задачах нечеткой кластеризации с использованием наиболее распространенного метода нечетких С-средних (FCM) [4] данные обычно преобразуются так, чтобы исходная выборка имела вид

$$\tilde{X} = (\tilde{x}(1), \tilde{x}(2), ..., \tilde{x}(k), ...\tilde{x}(N)) \subset R^n,$$

$$\tilde{x}(k) = (\tilde{x}_1(k), ..., \tilde{x}_i(k), .., \tilde{x}_n(k))^T, \ -1 \le \tilde{x}_i(k) \le 1,$$

$$1 < m < N, \ 1 \le j \le m, 1 \le i \le n, \ 1 \le k \le N.$$

И хотя на сегодня кроме FCM разработано множество методов и алгоритмов нечеткой кластеризации со своими достоинствами и недостатками, все они позволяют отыскать только локальный экстремум принятой целевой функции [6,7], что ведет к тому, что использование процедур оптимизации (нелинейного программирования) на основе производных принятого критерия в общем случае не позволяет получить искомое наилучшее решение. Преодолеть эту проблему можно, многократно решая задачу при разных начальных

условиях и выбирая наилучший вариант из множества полученных. Понятно, что подобный подход существенно увеличивает время решения задачи.

В связи с этим в [8]был предложен метод нечетких J-средних (FJM), сочетающий в себе стандартный FCM с элементами глобального случайного поиска [9-12]. FJM обеспечивает нахождение глобального экстремума с высокой вероятностью, однако время поиска может быть достаточно велико, что естественно, ограничивает возможности этого подхода.

Преодолеть указанные затруднения можно, воспользовавшись аппаратом гибридных систем вычислительного интеллекта (HSCI) [13-17], сочетающих в себе обучаемость искусственных нейронных сетей, интерпретируемость результатов и возможность работы в условиях перекрывающихся классов систем нечетного вывода и высокую скорость отыскания глобального экстремума, обеспечиваемую эволюционными алгоритмами оптимизации, основанными на «роях частиц» (PCO).

1. Алгоритм нечеткой кластеризации на основе оптимизации с помощью кошачьих стай

В основе широко распространенного алгоритма вероятностной нечеткой кластеризации [5], лежит процедура минимизации целевой функции

$$E(U_j(k),c_j)=\sum_{k=1}^{N}\sum_{j=1}^{m}U_j^{\beta}(k)\left\|\tilde{x}(k)-c_j\right\|^2 \qquad (1)$$

при ограничениях

$$\sum_{j=1}^{m}U_j(k)=1,\quad 0\le\sum_{j=1}^{m}U_j(k)\le N\ , \qquad (2)$$

(здесь β — неотрицательный параметр фаззификации (фаззификатор), задающий размытость границ между кластерами), в основе которой лежат стандартные методы нелинейного (при $\beta=2$ — квадратичного) программирования.

Записав функцию Лагранжа

$$L(U_j(k),c_j,\lambda(k))=\sum_{k=1}^{N}\sum_{j=1}^{m}U_j^{\beta}(k)\left\|\tilde{x}(k)-c_j\right\|^2+$$
$$+\sum_{k=1}^{N}\lambda(k)(\sum_{j=1}^{m}U_j(k)-1)$$

(здесь $\lambda(k)$ — неопределенные множители Лагранжа) и решив систему уравнений Каруша-Куна-Таккера

$$\begin{cases}\dfrac{\partial L(U_j(k),c_j,\lambda(k))}{\partial U_j(k)}=0,\\ \nabla_{c_j}L(U_j(k),c_j,\lambda(k))=\vec{0},\\ \dfrac{\partial L(U_j(k),c_j,\lambda(k))}{\partial\lambda_j(k)}=0,\end{cases}$$

получаем искомое решение вида

$$\begin{cases}U_j(k)=\dfrac{(\left\|\tilde{x}(k)-c_j\right\|^2)^{\frac{1}{1-\beta}}}{\sum\limits_{l=1}^{m}(\left\|\tilde{x}(k)-c_j\right\|^2)^{\frac{1}{1-\beta}}},\\ c_j=\dfrac{\sum\limits_{k=1}^{N}U_j^{\beta}(k)\tilde{x}(k)}{\sum\limits_{k=1}^{N}U_j^{\beta}(k)},\end{cases} \qquad (3)$$

которое при $\beta=2$ совпадает с алгоритмом нечетких C-средних (FCM) Дж. Бездека [4]:

$$\begin{cases}U_j(k)=\dfrac{(\left\|\tilde{x}(k)-c_j\right\|^2)^{-1}}{\sum\limits_{l=1}^{m}(\left\|\tilde{x}(k)-c_j\right\|^2)^{-1}},\\ c_j=\dfrac{\sum\limits_{k=1}^{N}U_j^{2}(k)\tilde{x}(k)}{\sum\limits_{k=1}^{N}U_j^{2}(k)}.\end{cases} \qquad (4)$$

В [7] была доказана сходимость процедур (3), (4) к локальному минимуму, при этом достижение глобального экстремума в общем случае не гарантируется.

В работах [18-19] задача условной оптимизации (1), (2) была переформулирована в задачу безусловной оптимизации целевой функции вида

$$E(c_j)=\sum_{k=1}^{N}(\sum_{j=1}^{m}\left\|\tilde{x}(k)-c_j\right\|^{2(1-\beta)})^{1-\beta}, \qquad (5)$$

при $\beta=2$ принимающей вид

$$E(c_j)=\sum_{k=1}^{N}(\sum_{j=1}^{m}\left\|\tilde{x}(k)-c_j\right\|^{-2})^{-1}, \qquad (6)$$

при этом интересно отметить, что в процессе минимизации (5), (6) отыскиваются только координаты центроидов $A_j, j=1,2,...,m$, а для нахождения уровней нечеткой принадлежности могут быть использованы первые уравнения соотношений (3), (4).

Таким образом, задача нечеткой кластеризации может быть сведена к поиску глобального экстремума целевых функций (5), (6). Для решения задачи могут быть использованы интенсивно развивающиеся в настоящее время в рамках HSCI эволюционные биоинспирированные «роевые» процедуры оптимизации [20], среди которых в качестве одних из наиболее быстродействующих можно отметить, так называемые, алгоритмы кошачьих стай [21,22]. Заметим, что именно кошачьи стаи с успехом были использованы для решения задач четкой кластеризации в рамках процедуры K-средних [23,24], порождаемой целевой функцией (1) при $\beta\to 1, U_j(k)=\{0,1\}$. В рамках этого подхода предполагается [23], что каждый центроид c_j представлен одной из кошек стаи, а конечное

решение определяется кошками, обеспечивающими минимум целевой функции $E(c_j)$ (5) или (6).

В рамках стандартного «кошачьего» алгоритма [21, 22] предполагается, что каждая кошка cat_p стаи, состоящей из Q особей ($p=1,2,...,Q$), может находиться в одном из двух состояний: режиме поиска (Seeking Mode - SM) и режиме погони (Tracing Mode - TM). При этом режим поиска связан с медленными движениями с незначительной амплитудой около исходной позиции (сканирование пространства в окрестности текущей позиции),а режим погони определяется быстрыми скачками с большой амплитудой и позволяет вывести кошку cat_p из локального экстремума, если она туда попала. Сочетание локального сканирования и резких изменений текущего состояния позволяет с большей вероятностью отыскать глобальный экстремум по сравнению с традиционными методами многоэкстремальной оптимизации.

Процесс отыскания экстремума с помощью кошачьей стаи может быть реализован в виде следующей последовательности шагов:

Шаг CS 1: создать стаю из Q кошек в виде набора n-мерных векторов $c_p(0)$, случайным образом распределенных на множестве допустимых значений аргументов R_c^n, т.е. $c_p(0) \in R_c^n \subset R^n$; оценить значение оптимизируемой функции (фитнесс - функции) $E(c_p(0))$ во всех Q точках, при этом предполагается, что целью оптимизации является отыскание глобального минимума $E(c)$.

Шаг CS 2: ввести параметр состояния SPC (self position consideration), принимающий два значения 1 или 0; случайным образом разделить стаю на две группы: кошки в поиске (SPC=1) и кошки в погоне(SPC=0).

Шаг CS 3: если SPC=1, запустить соответствующую группу кошек в поиск, оставшихся кошек с SPC=0 запустить в режим погони.

Шаг CS 4: оценить значение фитнесс - функции и сохранить новые состояния $c_p(1)$,соответствующие наименьшим значениям $E(c_p(1))$.

Шаг CS 5: вернуться к шагу CS1 с обновленной стаей $c_p(1), p=1,2,...,Q.$

Режимы поиска и погони могут быть реализованы параллельно и также состоять из последовательности итераций. При этом режим поиска кошачьей стаи соответствует процессу локального поиска в задаче оптимизации. Режим поиска определяется тремя основными факторами: объемом памяти поиска (seeking memory pool - SMP), который определяет количество создаваемых копий каждой кошки cat_p, шагом изменения по каждой координате пространства R_c^n (seeking range of the selected dimension - SRD) и изменяемых координат (counts of dimension to change - CDC). Собственно режим поиска может быть реализован в виде следующей последовательности шагов:

Шаг SM 1: если SPC = 1, создать C (C=SMP)копий cat_p.

Шаг SM2: в соответствии с принятым CDC изменить состояние cat_p.

Шаг SM3: оценить значения оптимизируемой фитнесс-функции для каждого измененного состояния cat_p.

Шаг SM 4: ввести вероятности выбора каждого изменяемого состояния

$$P_p = \frac{E(c_p(\tau)) - E_{\min}(c_p(\tau))}{E_{\max}(c_p(\tau)) - E_{\min}(c_p(\tau))}, \tau = 1,2,...,T$$

и кошку с максимальным значением P_p исключить из дальнейшего рассмотрения. Кошка с $P_p = 0$ является «наилучшей» копией cat_p, поскольку ей соответствует наименьшее значение оптимизируемой функции $E_{\min}(c_p(\tau))$.

Режим погони соответствует процессу глобального поиска, позволяющего «проскакивать» локальные экстремумы оптимизируемой функции, и также может быть реализован в виде последовательности шагов:

Шаг TM 1: если SPC = 0, для группы кошек в погоне рассчитать для каждой cat_p скорости движения по каждой координате с помощью рекуррентного выражения

$$\upsilon_{pi}(\tau+1) = \upsilon_{pi}(\tau) + r(\tau)\eta_{TM}(c_{best,i}(\tau) - c_{pi}(\tau)),$$

где $\upsilon_{pi}(\tau)$ — скорость движения @-й кошки по i-й координате на τ-й итерации погони, $0 < r(\tau) < 1$ — случайный параметр погони, η_{TM} — постоянный шаг погони, $c_{best,i}(\tau)$ — наилучшее решение задачи оптимизации, полученное на τ-й итерации.

Шаг TM 2: ввести предельно возможные значения скоростей $\upsilon_{\min}$ и $\upsilon_{\max}$, для каждой кошки проверить условие

$$\upsilon_{\min} < \upsilon_{pi}(\tau+1) < \upsilon_{\max}$$

и если оно нарушается, положить $\upsilon_{pi}(\tau+1)$ равным соответствующему значению $\upsilon_{\min}$ или $\upsilon_{\max}$.

Шаг TM 3: изменить положение каждой кошки в погоне согласно соотношению

$$c_{pi}(\tau+1) = c_{pi}(\tau) + \upsilon_{pi}(\tau).$$

Шаг TM 4:проверить, принадлежит ли $c_p(\tau+1)$ R_c^n.

Можно заметить, что рассмотренный алгоритм поиска реализует по сути покоординатный спуск (метод Гаусса - Зайделя), требующий многократного оценивания значений оптимизируемой функции и характеризующийся низкой скоростью сходимости. В режиме погони реализуется градиентный поиск с большим шагом, что в общем случае не гарантирует отыскание глобального экстремума. В связи с этим представляется целесообразным

модернизировать процедуру оптимизации на основе кошачьих стай путем ее рандомизации на основе случайного поиска [9-11], обладающего целым рядом преимуществ перед детерминированными процедурами поиска экстремума.

2. Рандомизированный алгоритм оптимизации на основе кошачьих стай в задаче нечеткой кластеризации

Поскольку режим поиска SM есть по сути процесс локальной оптимизации, движение каждой из кошек cat_p с SPC=1 целесообразно организовать в антиградиентном направлении согласно стандартной рекуррентной градиентной процедуре

$$A_p(\tau+1) = A_p(\tau) - \eta_{SM}\,\hat{\nabla} E(A_p(\tau)), \qquad (7)$$

где $\hat{\nabla} E(A_p(\tau))$ — оценка градиента оптимизируемой функции в точке $A_p(\tau)$, η_{SM} — шаг поиска в пространстве R_A^n.

Составляющие градиента $\hat{\nabla} E(A_p(\tau))$, являющиеся частными производными $\dfrac{\partial E(A_p(\tau))}{\partial A_p}$, могут быть оценены путем измерения оптимизируемой функции в пробных состояниях в окрестности точки $A_p(\tau)$. Наиболее простым с вычислительной точки зрения является поиск с центральной пробой [9], при этом производится оценка оптимизируемой функции в $(n+1)$-й точке $(CDC = n)$: $A_p(\tau)$, $A_p(\tau)+\eta_{SRD}e_1$, $A_p(\tau)+\eta_{SRD}e_2,...,A_p(\tau)+\eta_{SRD}e_n$, где e_i — координатные орты, η_{SRD} — величина пробного шага, определяемая принятым значением SRD.

Определив $n+1$ значение функции $E(A_p(\tau))$, $E(A_p(\tau)+\eta_{SRD}e_2),...,E(A_p(\tau)+\eta_{SRD}e_n)$, вместо градиента

$$\nabla E(A_p(\tau)) = \left(\frac{\partial E(A_p(\tau))}{\partial A_{p1}}, \frac{\partial E(A_p(\tau))}{\partial A_{p2}}, ..., \frac{\partial E(A_p(\tau))}{\partial A_{pn}}\right)^T,$$

можно ввести его оценку $\hat{\nabla} E(A_p(\tau))$ с компонентами

$$\frac{\partial \hat{E}(A_p(\tau))}{\partial A_{pi}} = \frac{1}{\eta_{SRD}}\left(E(A_p(\tau)) + \eta_{SRD}e_i\right) - E(A_p(\tau)), i = 1,2,...,n.$$

Реализовав далее шаг в пространстве R_c^n в соответствии с (7), приходим к новому состоянию cat_p в режиме поиска с координатами

$$\begin{cases} c_{p1}(\tau+1) = c_{p1}(\tau) - \dfrac{\eta_{SM}}{\eta_{SRD}}(E(c_p(\tau)) + \eta_{SRD}e_1) - E(c_p(\tau)), \\ c_{p2}(\tau+1) = c_{p2}(\tau) - \dfrac{\eta_{SM}}{\eta_{SRD}}(E(c_p(\tau)) + \eta_{SRD}e_2) - E(c_p(\tau)), \\ ... \\ c_{pn}(\tau+1) = c_{pn}(\tau) - \dfrac{\eta_{SM}}{\eta_{SRD}}(E(c_p(\tau)) + \eta_{SRD}e_n) - E(c_p(\tau)). \end{cases}$$

Можно заметить, что в случае

$$E(c_p(\tau+1)) < E(c_p(\tau)),$$

cat_p приближается к локальному минимуму, т.е. улучшает свое состояние и может далее оставаться в режиме поиска. Если же

$$E(c_p(\tau+1)) \geq E(c_p(\tau)),$$

cat_p находится в окрестности локального минимума, вывести из которого ее можно, переведя в режим погони.

В качестве недостатка этой процедуры оптимизации можно отметить фиксированное значение $CDC = n$, что требует поочередного изменения всех координат cat_p в пространстве R_c^n. Расширить возможности процесса поиска можно, обратившись к рандомизированным процедурам, простейшей из которых является чисто случайная оценка направления спуска, смысл которой состоит в том, что из состояния $c_p(\tau)$ делается случайная проба $c_p(\tau)+\eta_{SRD}\Xi$, где $\Xi = (\xi_1, \xi_2, ..., \xi_n)^T$ — единичный случайный вектор, равномерно распределенный в пространстве R_c^n. В случае, если $c_p(\tau)+\eta_{SRD}\Xi < E(c_p(\tau))$, делается рабочий шаг поиска

$$c_p(\tau+1) = c_p(\tau) - \eta_{SM}\Xi \qquad (8)$$

(при этом можно принять $\eta_{SRD} = \eta_{SM}$), в противном случае проба признается неудачной и реализуется попытка с новым вектором Ξ.

Обобщением этой процедуры является оценка направления поиска по наилучшей из нескольких случайных проб. При этом из исходного состояния $c_p(\tau)$ делается несколько случайных проб оптимизируемой функции $c_p(\tau)+\eta_{SRD}\Xi_l$ в случайных направлениях $\Xi_l (l = 1,2,...,n,...,L)$, при этом фактор CDC может превышать значение n. За направление спуска выбирается то направление Ξ^*, которое обеспечило наименьшее значение функции $E(c_p)$, т.е. cat_p переводится в новое состояние согласно выражению

$$c_p(\tau+1) = c_p(\tau) + \eta_{SRD}\Xi^*. \qquad (9)$$

Заметим также, что при $L = 1$, процедуры (8) и (9) совпадают.

Объединяя описанные процедуры поиска, можно ввести в рассмотрение поиск на основе статистического градиента. В этом случае за оценку градиента принимается средневзвешенное из L случайных направлений, каждое из которых берется с весом, соответствующим вариации $E(c_p)$ вдоль этого направления:

$$\hat{\nabla} E(c_p(\tau)) = -\frac{\sum_{l=1}^{L}\Xi_l\left(E\left(c_p(\tau)+\eta_{SRD}\Xi_l\right) - E(c_p(\tau))\right)}{\left\|\sum_{l=1}^{L}\Xi_l\left(E\left(c_p(\tau)+\eta_{SRD}\Xi_l\right) - E(c_p(\tau))\right)\right\|}. \qquad (10)$$

Подставляя далее (10) в (9), получаем процедуру градиентного спуска в направлении минимума оптимизируемой функции.

Таким образом, все кошки с SPC=1 смещаются в направлении локальных минимумов оптимизируемой функции.

Режим погони TM в отличие от локального режима поиска SM обеспечивает общей процедуре оптимизации на основе CS глобальные свойства, позволяющие не застревать ей в локальных экстремумах. Понятно, что кроме рассматриваемых процедур существуют и другие алгоритмы, обладающие требуемыми свойствами.

Одним из таких наиболее эффективных численно простых алгоритмов является метод тяжелого шарика, опирающийся на аналогию движения тяжелого тела по искривленной поверхности с учетом сил тяжести и трения. При этом в силу инерции шарик-кошка «проскакивает» локальные экстремумы, а в силу трения движение должно остановиться в глобальном экстремуме.

Данный алгоритм для кошек в режиме погони (SPC=0)может быть записан в виде [25]

$$c_p(\tau+1) =$$

$$c_p(\tau) - \alpha(c_p(\tau) - c_p(\tau-1)) - \eta_{TM}\,\hat{\nabla} E(c_p(\tau)), \quad (11)$$

где α — параметр, определяющий инерционные свойства процесса погони. При $\alpha = 0$ (11) полностью совпадает с (7), отличаясь только шагом η_S .При $\alpha = 1$ процесс погони становится незатухающим, поэтому этот параметр выбирается в интервале $0 < \alpha < 1$, при этом чем ближе α к единице, тем сильнее проявляются инерционные свойства, однако процесс слабо затухает в окрестности экстремума. В связи с этим целесообразно каждой кошке с SPC=0 назначить разные значения параметра α.

Заметим также, что в процедуру (11) может быть введена случайная компонента, вводящая дополнительное «рыскание» в процесс погони, улучшающее глобальные свойства алгоритма. При этом (11) модифицируется к виду

$$c_p(\tau+1) = c_p(\tau) - \alpha(c_p(\tau) - c_p(\tau-1)) -$$

$$-\eta_{TM}\,\hat{\nabla} E(c_p(\tau)) + \eta_{SRD}\Xi,$$

т.е. cat_p одновременно находится и в режиме погони и в режиме поиска-сканирования пространства R_c^n.

3. Экспериментальные исследования

Эксперименты по кластеризации FCMCSO проводились с использованием четырех наборов данных: Iris, Cancer, Wine and Glass. Каждый из наборов данных имеет ряд параметров, представленны в таблице 1.

Таблица 1

Характеристические параметры выборок

Название выборки	Число классов	Число атрибутов	Кол-во набл.
Iris	3	4	150
Cancer	2	9	683
Wine	3	13	178
Glass	6	8	214

Таблица 2

Параметры алгоритма нечеткой кластеризации на основе оптимизации стаи кошек (FCMCSO)

Параметры	Значения
Поиск диапазона выбранного измерения (SRD)	Случайно [0,1]
Поиск пула памяти(SMP)	5
Размер популяции	Количество кластеров
r_1	Случайное значение в диапазоне[0,1]
c_1	Константа
Самооценка позиции (SPC)	Случайно в диапазоне [0,1]
Количество итераций	Manually

Таблица 3

Сравнительные результаты временной обработки алгоритмов кластеризации таких, как FCM, PSO, GSA, CSO и FCMCSO

Название выборки	FCM	PSO	GSA	CSO	FCM CSO
Iris	0.008	0.020	0.022	0.043	0.012
Cancer	0.009	0.138	0.204	0.026	0.007
Wine	0.009	0.282	0.098	0.076	0.013
Glass	0.010	0.431	0.431	0.021	0.020

Таблица 4

Результаты кластеризации CSO и FCMCSO с различным количеством итераций (средняя ошибка в %)

Название выборки	Количество итераций CSO			Количество итераций FCMCSO		
	50	100	150	50	100	150
Iris	23.34	20.84	21.67	17.55	14.78	16.46
Cancer	40,23	40,55	41,47	38.89	39.22	39.15
Wine	24,55	21,44	22,20	18.43	17.37	16.32
Glass	56.34	56.48	55.67	51.63	51.7	49.79

Выводы

Рассмотрена задача нечеткой кластеризации массива наблюдений на основе нечеткого вероятностного подхода, в основу которого положен алгоритм нечетких С-средних, переформулированный в задачу безусловной многоэкстремальной оптимизации. Для решения задачи использована рандомизированная модификация алгоритма оптимизации кошачьих стай, отличающаяся от известной введением в процессы поиска и погони элементов случайного поиска. Использование рандомизированной модификации позволило улучшить точность определения направления движения в режиме кошачьего поиска и улучшить глобальные свойства процедуры в режиме погони, что, в свою очередь, улучшает качество решения собственно задачи кластеризации. Использование эволюционного метода оптимизации кошачьих стай позволило упростить численную реализацию процесса кластеризации, сократить необходимый объем стаи и исключить использование, так называемых, копий каждой кошки.

Список литературы

[1] Gan G., Ma Ch., Wu J. Data Clustering: Theory, Algorithms and Applications. – Philadelphia, Pennsilvania: SIAM, 2007. – 455 p.

[2] Abonyi J., Feil B. Cluster Analysis for Data Mining and System Identification. – Basel: Birkhauser, 2007. – 303p.

[3] Xu R., Wunsch D.C. II. Clustering– Hoboken, N.J.: John Wiley & Sons, Inc., 2009. – 341 p.

[4] Bezdek J.C. Pattern Recognition with Fuzzy Objective Function Algorithms.-N.Y.:Plenum Press, 1981.-272p.

[5] Höppner F., Klawonn F., Kruse R., Runkler T. Fuzzy Clustering Analysis:Methods for Classification, Data Analisys and Image Recognition.-Chichester: John Wiley &Sons, 1999. – 289 p.

[6] Bezdek J.C., Keller J., Krishnapuram R., Pal N.R. Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. – N.Y.: Springer Science + Business Media, Inc., 2015. – 776 p.

[7] Bezdek J.C. A convergence theorem for the fuzzy ISODATA clustering algorithms. – IEEE Trans. Pattern Anal. Mach. Intell. – 1980 – 2. – P. 1-8.

[8] Belacel N., Hansen P., Mladenovic N. Fuzzy J-Means: a new heuristic for fuzzy clustering – Pattern Recognition. – 2007. – 35. – P. 2193-2200.

[9] Растригин Л.А. Статистические методы поиска. – Москва: Наука, 1968. – 376 с.

[10] Растригин Л.А., Рипа К.К. Автоматическая теория случайного поиска. – Рига: Зинатне, 1973. – 343 с.

[11] Растригин Л.А. Случайный поиск в процессах адаптации. – Рига: Зинатне, 1973. – 132 с.

[12] Kirkpatrick S.C.G., Vecci M. Optimization by simulated annealing – Science – 1983 – 220 – P. 49-58.

[13] L.Rutkowski. Computational Intelligence. Methods and Techniques. Berlin-Heidelberg: Springer-Verlag, 2008. – 514 p.

[14] Mumford C. L., Jain L.C. Computational Intelligence. Berlin: Springer-Verlag, 2009. – 729 p.

[15] Kroll A. Computational Intelligence. Eine Einführung in Probleme, Methoden und technishe Anwendungen – München: Oldenbourg Verlag, 2013. – 428 S.

[16] Kruse R., Borgelt C., Klawonn F., Moawes C., Steinbrecher M., Held P. Computational Intelligence. A Methodological Introduction. – Berlin: Springer-Verlag, 2013. – 488 p.

[17] Kacprzyk J., Pedrycz W. Springer Handbook of Computational Intelligence. – Berlin Heidelberg: Springer-Verlag, 2015. – 1634 p.

[18] R.J. Hathaway, J.C Bezdek. Optimization of clustering criteria by reformulation. – IEEE Trans. Fuzzy Systems.-1995. – 3. – P. 241-245.

[19] Pal N.R., Bezdek J.C.,Hathaway R.J. Sequental competitive learning algorithm. – Neural Networks. – 1996. – 9. – № 5. – P. 787-796.

[20] Grosan C., Abraham A., Chis M. Swarm intelligence in Data Mining – Studies in Computational Intelligence. – 2006. – 34. – P. 1-20.

[21] Chu S.-C., Tsai P.-W., Pan J.S. Cat swarm optimization // Lecture Notes in Artificial Intelligence. – 4099. – Berlin Heidelberg: Springer-Verlag, 2006. – P. 854-858.

[22] Chu S.-C., Tsai P.-W. Computational Intelligence based on the behavior of cats // Int. J. of Innovative Computing, Information, and Control. – 2007. – 3. – №1. – P.163-173.

[23] B. Santosa and M. K. Ningrum, Cat Swarm Optimization for Clustering, Soft Computing and Pattern Recognition, International Conference of (SOCPAR), Malacca, Malaysia, 2009, P. 54-59.

[24] Liu Y., Wu, Shen Y. Cat swarm optimization clustering (KSACSOC): A cat swarm optimization clustering algorithm. – Sci. Reseach and Essays – 2012 – 7. №49. – P. 4176-4185.

[25] Бодянский Е.В, Шафроненко А.Ю. Рандомизированная модификация метода оптимизации на основе кошачьих стай. – Системи обробки інформації. – 2018. – № 1(152). – С. 142-147.

Поступила в редколлегию 12.09.2018