АЛГОРИТМ «ЛЕТУЧЕЙ МЫШИ» ДЛЯ ОПТИМИЗАЦИИ НЕЧЕТКОГО КЛАССИФИКАТОРА

А. В. Власенко, студентка 3-го курса каф. БИС

Научный руководитель И. А. Ходашинский, профессор каф. КИБЭВС

г. Томск, ТУСУР
E-mail: airtone@list.ru

Проект ГПО КИБЭВС-1404 – Нечёткие классификаторы обнаружения вторжений

Введение

Оптимизация представляет собой попытку получения оптимального решения задачи при заданных условиях. Важнейшей задачей оптимизации является сведение к минимуму потерь времени или получение максимальной выгоды от данной технической системы. Все системы, которые должны быть оптимизированы, имеют целевую функцию и несколько переменных, которые влияют на функцию.

Таким образом, методы оптимизации могут быть определены, как процесс достижения оптимальных решений, которые удовлетворяют заданной целевой функции. [1]

Целью данной работы является реализация работы алгоритма «летучей мыши» для оптимизации нечеткого классификатора.

Постановка задачи

В данной работе рассматривался нечеткий классификатор.

Построение нечеткой системы осуществляется на основе таблицы наблюдений, где входному вектору признаков $X = \{x_1, x_2, x_3, \dots, x_n\}$ соответствует на выходе некоторая метка класса c_j ($j \in [1, \dots, m], m$ – число правил).

Нечеткий классификатор задается правилами следующего вида:

$$\text{ЕСЛИ } x_1 = A_{1i} \text{ И } x_2 = A_{2i} \text{ И } x_3 = A_{3i} \text{ И } \dots \text{ И } x_n = A_{ni} \text{ ТО } class = c_j, \quad (1)$$

где $X = \{x_1, x_2, x_3, \dots, x_n\}$ – входной вектор признаков классифицируемого объекта;

A_{ki} – нечеткий терм, характеризующий k-ый признак в i-том правиле;

$i \in [1, \dots, m], m$ – число правил;

c_j – идентификатор j-ого уровня, $j \in [1, \dots, m]$.

Таким образом, нечеткий классификатор может быть представлен функцией вида:

$$C = f(X, \theta), \quad (2)$$

где θ – вектор антецедентов.

Степень принадлежности объекта к каждому классу вычисляется следующим образом:

$$\beta_j(x) = \sum_{R_{ij}} \prod_{k=1}^{n} A_{ki}(x_k), j = 1,2,..m \quad (3)$$

Выходом классификатора является метка класса, определяемая следующим образом:

$$class = c_{j*}, j *= \arg \max_{i<j<m} \beta_j \quad (4)$$

На множестве обучающих данных (таблица наблюдений) $\{(x_p; c_p), p = 1, \dots, m\}$, определим следующую функцию:

$$delta(p,\theta) = \begin{cases} 1, \text{если } c_p = f(x_p,\theta) \\ 0, \text{иначе} \end{cases} \quad (5)$$

Тогда точность классификации выражается следующим образом:

$$E(\theta) = \frac{1}{z}\sum_{p=1}^{z} delta(p,\theta) \quad (6)$$

Описание алгоритма

Алгоритм летучей мыши (АЛМ) был предложен Янгом в 2010 году и представляет из себя метаэвристический алгоритм роевого интеллекта, основанный на эхолокации. Эхолокация представляет собой тип ультразвука, благодаря которому летучие мыши могут летать и охотиться.

Большинство видов летучих мышей обладает совершенными средствами эхолокации, которые используются ими для обнаружения добычи и препятствий, а также для обеспечения возможности разместиться в темноте на насесте.

АЛМ подчиняется следующим правилам:

- Все летучие мыши используют эхолокацию, чтобы анализировать расстояние, а также иметь различие между едой (добычей) и природными препятствиями;

- Летучие мыши перемещаются случайным образом со скоростью v_i в позицию x_i с частотой f_{min}, изменяемой длиной волны и громкостью A_0 для поиска добычи. Они могут автоматически регулировать длину волны (или частоту) испускаемого импульса и регулировать его скорость $r \in [0,1]$ в зависимости от близости их цели;

- Хотя громкость может варьироваться разными способами, мы предполагаем, что громкость изменяется от большого (положительного) A_0 до минимального постоянного значения A_{min}.

Структура алгоритма:

1 Инициализация популяции летучих мышей:

$$x_{ij} = x_{min} + \varphi(x_{max} - x_{min}), \quad (7)$$

где $i = 1, 2, \dots, N$, N – количество мышей;

$j = 1, 2, \dots, d$, d – количество параметров;

x_{min} и x_{max} - верхние и нижние границы для размерности j соответственно;

$\varphi \in [0; 1]$ – случайно сгенерированное значение.

2 Генерация частоты, скорости и новых решений:

$$f_i = f_{min} + \beta(f_{max} - f_{min}) \quad (8)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i \quad (9)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (10)$$

где $f_i -$ значение частоты, принадлежащей iтой мыши;

f_{max} и f_{min} - минимальные и максимальные значения частоты соответственно;

$\beta \in [0; 1]$ – случайно сгенерированное значение;

x_*- глобальный оптимум (ставший таковым после нахождения лучшего локального решения), полученный после сравнения всех предыдущих значений среди N летучих мышей;

v_i^t- скорость i-той летучей мыши на временном шаге t.

3 Поиск локального значения (решения) в алгоритме:

$$x_{new} = x_* + \varepsilon\overline{A^t}, \quad (11)$$

где $\overline{A^t}$ - среднее значение громкости всех летучих мышей на временном шаге t;

$\varepsilon \in [-1; 1]$ – случайно сгенерированное значение.

4 Понижение громкости и увеличение скорости распространения импульса:

$$A_i^{t+1} = \alpha A_i^t \quad (12)$$

$$r_i^{t+1} = r_i^0(1 - e^{-\gamma t}) \quad (13)$$

где α и γ – ограничения;

r_i^0- начальное значение скорости импульса i-той мыши.

Эксперимент

Для проверки работы алгоритма оптимизации (далее именуемый как BS), были проведены тесты на наборах данных из репозитория KEEL (http://www.keel.es) для нечеткого классификатора: iris, appendicitis, bupa. Параметры: размер популяции – 20, количество итераций – 20, начальное значение громкости – 1, минимальное значение громкости – 0, максимальное значение частоты – 1, минимальное значение частоты – 0, $\alpha = \gamma = 0{,}9$ [1][2].

Результаты работы алгоритма приведены в таблице 1 [3]:

Таблица 1 – Результаты работы алгоритма

Набор данных	iris			bupa			appendicitis		
	tra	tst	СКО(tra)	tra	tst	СКО(tra)	tra	tst	СКО(tra)
BS	**97.67**	**97.33**	1.47	64.35	62.03	2.00	**87.27**	**83.98**	3.63
Ant Miner	97,26	96.00	0,74	**80,38**	57.25	3,25	-	-	-
Core	95,48	92.67	1,42	61,93	61.97	0,89	-	-	-
Hider	97,48	96.67	0,36	73,37	65.83	2,70	-	-	-
Sgerd	97,33	96.67	0,36	59,13	57.89	0,68	-	-	-
Target	93,50	92.93	2,42	68,86	**65.97**	0,89	-	-	-

Список литературы

1 A New Modification Approach on Bat Algorithm for Solving Optimization Problems // Selim Yılmaz Ecir U. Kucuksille, Applied Soft Computing, 2014, с.1, 16;

2 Bat algorithm for constrained optimization tasks // Amir Hossein Gandomi, Xin-She Yang, Amir Hossein Alavi, Siamak Talatahari, Springer-Verlag London Limited, 2012, с. 1240 – 1241;

3 Alcala-Fdez J., Fernandez A., Luengo J., Derrac J., Garcia S., Sanchez L., Herrera F. KEEL Data-mining software Tool: data, set repository, integration of algorithms and experimental analysis framework // Valued Logic & Soft Computing. Vol. 17. – 2011. – С. 255-287.