Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный университет имени М. В. Ломоносова»
Физический факультет

Курсовая работа

Использование нечеткого классификатора в машинном обучении

Допущено к защите

Заведующий каф. математического моделирования и информатики д. ф.-м. н. проф.

_____ А. И. Чуличков

Выполнил:
студент 204 гр. М. А. Ященко

Научный руководитель:
к. ф.-м. н. А. В. Зубюк

Москва – 2020

Содержание

Введение

Нечеткая логика является разделом математики, в котором изучаются рассуждения в условиях нечеткости и неопределенности, размытости. Данный раздел основан на нечетких правилах, которые стали идейным развитием классической логики. Впервые эти правила были описаны в работе Лотфи Заде «Нечеткие множества» в 1965 году [1]. С тех пор нечеткая логика стремительно развивалась, и сегодня используется во многих сферах науки и техники, таких как автомобильная промышленность (автоматическая коробка передач) и использование детектирующих устройств (сонар, радар).

Одним из самых ярких примеров использования нечеткой логики является метрополитен города Сендай. В каждом поезде на полу был установлен микрокомпьютер, который отвечал за автовождение. Это устройство, подчиняясь нечетким методам управления, контролировало скорость поезда на всем протяжении пути. Предварительные испытания показали, что ошибки остановки более чем на 30 см составляют менее одного процента, стандартное отклонение — 10,61 см, что зачастую гораздо более точно, чем когда поездом управляет человек. Система автовождения отлично функционировала и после ввода ее в эксплуатацию.

Еще одной областью применения нечеткой логики является машинное обучение. Есть много типов нейронных сетей, в алгоритмах обучения которых используются нечеткие правила, таких как нечеткий перцептрон или сеть, с применением нечеткой версии алгоритма k-ближайших соседей.

Объектом изучения данной работы является нечеткий классификатор, принцип работы которого основан на понятиях нечеткого класса и нечеткого множества.

Цель работы — практическая реализация алгоритма обучения нейронной сети.

Задачи исследования: рассмотрение литературных источников по данной теме, изучение ключевых понятий, непосредственная практическая реализация.

В случае успешной реализации данного алгоритма, полученная модель может быть использована на практике, например в распознновании образов.

В главе 1 дается краткий экскурс в нечеткую математику и изучаются основные понятия, необходимые для понимания работы алгоритма. Сам алгоритм нечеткого классификатора описан в главе 2.

Глава 1 Нечеткая математика и нечеткие системы

Нечеткие множества были введены Заде для представления информации, которая является не точной, а скорее примерной или нечеткой. Расширение Заде для теории множеств, к примеру, позволяет показать в какой степени верно то или иное утверждение. В свою очередь классическая теория множеств описывает точные события и утверждения, которые однозначно истина или ложь, 0 или 1. Классические множества используют теорию вероятностей, для того чтобы сказать произойдет ли событие, пытаясь опрелить шанс, с которым данное действие случится. Ярким примером является бросок монеты – всего два варианта: орел или решка, все довольно однозначно. В отличие от теории вероятностей нечеткая теория измеряет степень, с которой, то или иное действие происходит или утверждение является истиной. Степень «молодости» человека сильно отличается от вероятности того, что человек молод. Теория веротностей недвусмысленно говорит нам, что человек или молод или стар, в то время как нечеткая теория показывает, что человек в какой-то степени молод.

1.1 Основые понятия и определения

Для лучшего понимания принципа работы нечеткого классификатора в первой главе будут даны основные понятия и определения.

1.1.1 Нечеткое множество

Опр. Нечеткое множество A является совокупностью упорядоченных пар, составленных из элеметов x множества χ и соотвествующим им степеням принадлежности $m_A(x)$:

$$A = (x, m_A(x)|\ x \in \chi)\,,$$

где $m_A(x)$ – функция принадлежности, указывающая, в какой степени элемент x принадлежит множеству A.

1.1.2 Операции над нечеткими множествами

Операции над нечеткими множествами являются расширениями подобных операций над классическими множествами. Здесь будут описаны

такие операции и понятия, как сравнение, включение, пересечение, объединение и дополнение. Пусть нечеткое множество A $\in \chi$ и нечеткое множество B $\in \chi$, тогда данные операции определены следующим образом:

Сравнение: A = B, если

$$m_A(x) = m_B(x), \quad \forall x \in \chi.$$

Включение: A $\subset$ B, если

$$m_A(x) < m_B(x), \quad \forall x \in \chi.$$

Объединение: На данный момент в нечеткой математике было дано несколько определений объединения нечетких множеств. Самое распостраненное и часто используемое — такое:

$$A \cup B : m_{A \cup B} = max(m_A(x), m_B(x)), \quad \forall x \in \chi.$$

Пересечение:

$$A \cap B : m_{A \cap B} = min(m_A(x), m_B(x)), \quad \forall x \in \chi.$$

Дополнение: Множество $\bar{A}$, дополнение множества A, определяется так:

$$m_{\bar{A}}(x) = 1 - m_A(x), \quad \forall x \in \chi.$$

В дополнение к описанным выше операциям для нечетких множеств определено также множество других операций, к примеру арифметические операции: сложение, умножение, распределительный закон и т.д. , но в данной работе они не приводятся.

1.2 Определение гиперпрямоугольника и его функции принадлежности

Одним из первых определение гиперпрямоугольника и его функции принадлежности, как и алгоритм нечеткого классификатора, описал Патрик Симпсон в работах [2–5].

1.2.1 Гиперпрямоугольник

Будем определять гиперпрямоугольник (или гипербокс) F_i , как множество точек x, лежащих в области:

$$F_i = \{X, B_i, T_i, f(X, B_i, T_i)\}, \quad \forall X \in \mathbb{R}^n,$$

где n – размерность пространства, в котором рассматривается гипепрямоугольник, B_i (Bottom) – минимальная точка гиперпрямоугольника (расположена ближе всего к началу координат), T_i (Top) – максимальная точка гиперпрямоугольника (расположена дальше других точек гиперпрямоугольника от начала координат, речь идет только о тех точках функция принадлежности которых равна 1), f(X,B_i, T_i) - функция принадлежности вектора X к гиперпрямоугольнику F_i, которая, как можно заметить, зависит уже не только от значения X, но и от B_i, T_i.

1.2.2 Нечеткий класс

Используя понятие гиперпрямоугольника, определим также нечеткий класс следующим образом:

Опр. Нечеткий класс C_K:

$$C_K = \bigcup_{i \in K} F_i,$$

где K – индекс множества гиперпрямоугольников, относящихся к классу k. Обращаясь к определению объединения, используемому выше, отмечу, что в данном случае под объединением имеется ввиду максимизация функции принадлежности всех гиперпрямоугольников, относящихся к данному классу.

1.2.3 Функция принадлежности гиперпрямоугольников

Функция принадлежности i-ого гиперпрямоугольника $d_i(A_h)$, $0 \leq d_i(A_h) \leq 1$ должна показывать степень, с которой h-ый n-мерный вектор A_h, выпадает из данного гиперпрямоугольника D_i. Если рассматривать с точки зрения размерностей, то функция принадлежности демонстрирует, насколько каждая координата A_h больше (меньше) максимальной (минимальной) координаты гиперпрямоугольника данной оси. Чем ближе точ-

ка, соответствующая вектору, к гиперпрямоугольнику, тем ближе $d_i(A_h)$ к единице, внутри гиперпрямоугольника $d_i(A_h) = 1$, показывая, что точка целиком принадлежит гиперпрямоугольнику.

Функция, которая отвечает всем требованиям, является суммой двух компонентов — среднего числа отклонений от максимальной точки и среднего числа отклонений от минимальной точки:

$$d_i(A_h) = \frac{1}{2n}\sum_{j=1}^{n}[max(0, 1 - max(0, \gamma min(1, a_{hj} - t_{ij}))) \\ + max(0, 1 - max(0, \gamma min(1, b_{ij} - a_{hj})))], \tag{1}$$

где $A_h = (a_{h1}, a_{h2}, \ldots, a_{hn}) \in \mathbb{R}^n$ — h-ый вектор, принадлежность которого мы хотим выяснить, $T_i = (t_{i1}, t_{i2}, \ldots, t_{in})$ — минимальная точка D_i, $B_i = (b_{i1}, b_{i2}, \ldots, b_{in})$ — максимальная точка D_i, а γ — параметр, регулирующий, как быстро функция принадлежности уменшается с ростом расстояния между A_h и D_i (чувствительность).

Глава 2 Алгоритм обучения нечеткого классификатора

2.1 Идея алгоритма

Обучение нечеткого классификатора представляет собой процесс расширения и сжатия. Данные для обучения представляют собой массив упорядоченныч пар $\{\vec{X_h}, c_h\}$, где $\vec{X_h} = (x_{h1}, x_{h2}, \ldots, x_{hn}) \in \mathbb{R}^n$, $x_{h1}, x_{h2}, \ldots, x_{hn}$ — координаты вектора, а c_h — класс, к которому этот вектор принадлежит. Процесс выглядит следующим образом. Берется пара $\vec{X_i}, c_i$ и ищется гиперпрямоугольник класса c_i, который можно расширить за счет $\vec{X_i}$. Если такой гиперпрямоугольник существует, то он расширяется. Если нет, то создается новый гиперпрямоугольник, который добавляется к рисунку.

Одной из тонкостей расширения гиперпрямоугольников является их пересечение. Если пересекаются гиперпрямоугольники одного класса, то никакой проблемы не возникает. Когда же пересекаются гиперпрямоугольники разных классов, пересечение устраняется за счет сжатия.

Рассмотрим этапы описанного алгоритма более подробно.

2.2 Расширение гиперпрямоугольника

Возьмем m-ую упорядоченную пару $\vec{X_m}, c_m$. Для нее подбирается гиперпрямоугольник F_j с наибольшей степенью принадлежности, для которой возможно расширение. Степень принадлежности определяется (1). Максимальный размер гиперпрямоугольника ограничен $0 \leq \beta \leq 1$, значение β определяется пользователем. Для того, чтобы расширить гиперпрямоугольник F_j за счет $\vec{X_m}$, нужно, чтобы было выполнено следующее условие:

$$n\beta \geq \sum_{i=1}^{n} [max(t_{ji}, x_{mi}) - min(b_{ji}, x_{mi})].$$

Если данное условие было выполнено, то координаты T_j и B_j обновляются следующим образом:

$$t_{ji}^{new} = max(t_{ji}^{old}, x_{mi}), \quad \forall i = \overline{1, n},$$

$$b_{ji}^{new} = min(b_{ji}^{old}, x_{mi}), \quad \forall i = \overline{1, n}.$$

2.3 Тест на пересечение гиперпрямоугольников

Как было описано ранее, пересечение гиперпрямоугольников одного класса не помешает формированию рисунка данного класса. При этом важно находить и убирать пересечения гиперпрямоугольников, являющихся представителями разных классов.

Для того, чтобы выяснить, образовалось ли пересечение после очередного расширения пройдемся по всем размерностям и выполним следующую проверку. Если для каждой размерности выполнено хотя бы одно из четырех условий, значит существует перечесение между двумя гиперпрямоугольниками. Будем считать, что гиперпрямоугольник F_j был расширен на предыдущем шаге, а гиперпрямоугольник F_k представляет другой класс и проверяется на возможное пересечение с F_j. Проводя тестирование, сохраним индекс, отвечащий размерности с наименьшим пересечением, чтобы во время сжатия минимизировать измененеие размеров гиперпрямоугольников. Положим изначально $\alpha^{old} = 1$. Теперь рассмотрим четыре случая:

Случай 1: $b_{ji} < b_{ki} < t_{ji} < t_{ki}$,

$$\alpha^{new} = min(t_{ji} - b_{ki}, \alpha^{old}).$$

Случай 2: $b_{ki} < b_{ji} < t_{ki} < t_{ji}$,

$$\alpha^{new} = min(t_{ki} - b_{ji}, \alpha^{old}).$$

Случай 3: $b_{ji} < b_{ki} < t_{ki} < t_{ji}$,

$$\alpha^{new} = min(min(t_{ki} - b_{ji}, t_{ji} - bki), \alpha^{old}).$$

Случай 4: $b_{ki} < b_{ji} < t_{ji} < t_{ki}$,

$$\alpha^{new} = min(min(t_{ji} - b_{ki}, t_{ki} - bji), \alpha^{old}).$$

Здесь i – размерность, проверяемая на текущей итерации теста. Если $\alpha^{old} - \alpha^{new} > 0$, тогда $\Delta = i$ и $\alpha^{old} = \alpha^{new}$, означая, что найдено пересечение для Δ-ой размерности и тестирование продолжается для (i+1)-ой размерности. Если же данное условие не выполнено, то тестирование прекраща-

ется, и мы считаем, что нашли размерность с минимальным пересечением. Ее индекс лежит в Δ, изначально $\Delta = -1$.

2.4 Сжатие гиперпрямоугольников в случае наличия пересечения

Если $\Delta > 0$, тогда Δ-ая размерность двух гиперпрямоугольников регулируется. Только одна из n размерностей регулируется в каждом гиперпрямоугольнике, чтобы сохранить размер гиперпрямоугольников настолько большим, насколько это возможно и минимизировать изменения в размерах. Чтобы выяснить, какие изменения лучше произвести в ходе регулирования, рассмотрим все те же четыре случая, описанные в разделе 2.3:

Случай 1: $b_{j\Delta} < b_{k\Delta} < t_{j\Delta} < t_{k\Delta}$,

$$t_{j\Delta}^{new} = b_{k\Delta}^{new} = \frac{t_{j\Delta}^{old} + b_{k\Delta}^{old}}{2}.$$

Случай 2: $b_{k\Delta} < b_{j\Delta} < t_{k\Delta} < t_{j\Delta}$,

$$t_{k\Delta}^{new} = b_{j\Delta}^{new} = \frac{t_{k\Delta}^{old} + b_{j\Delta}^{old}}{2}.$$

Случай 3а: $b_{j\Delta} < b_{k\Delta} < t_{k\Delta} < t_{j\Delta}$ и $(t_{k\Delta} - b_{j\Delta}) < (t_{j\Delta} - b_{k\Delta})$,

$$b_{j\Delta}^{new} = t_{k\Delta}^{old}.$$

Случай 3б: $b_{j\Delta} < b_{k\Delta} < t_{k\Delta} < t_{j\Delta}$ и $(t_{k\Delta} - b_{j\Delta}) > (t_{j\Delta} - b_{k\Delta})$,

$$t_{j\Delta}^{new} = b_{k\Delta}^{old}.$$

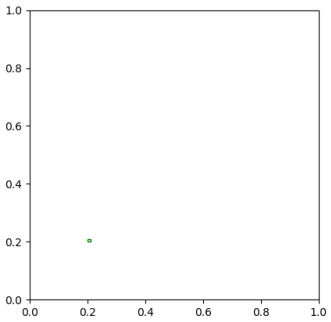
Случай 4а: $b_{ki} < b_{ji} < t_{ji} < t_{ki}$ и $(t_{k\Delta} - b_{j\Delta}) < (t_{j\Delta} - b_{k\Delta})$,

$$t_{k\Delta}^{new} = b_{j\Delta}^{old}.$$

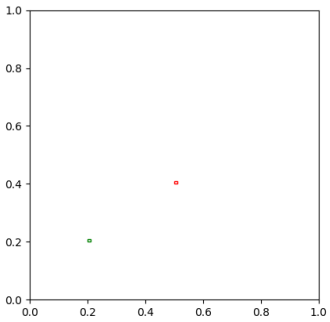
Случай 4б: $b_{ki} < b_{ji} < t_{ji} < t_{ki}$ и $(t_{k\Delta} - b_{j\Delta}) > (t_{j\Delta} - b_{k\Delta})$,

$$b_{k\Delta}^{new} = t_{j\Delta}^{old}.$$

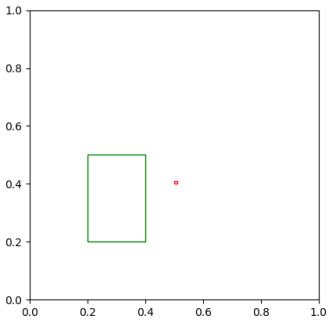
2.5 Пример обучения классификатора



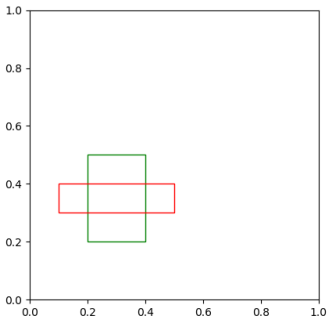
(а) $T_1 = B_1 = (0.2, 0.2)$, добавили первую точку



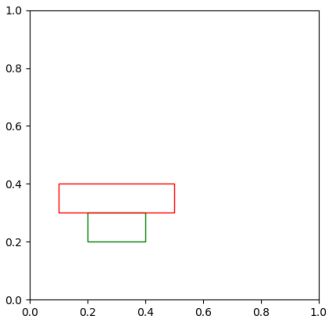
(б) $T_1 = B_1 = (0.2, 0.2)$, $T_2 = B_2 = (0.5, 0.4)$, добавили вторую точку



(в) $B_1 = (0.2, 0.2), T_1 = (0.4, 0.5)$, $T_2 = B_2 = (0.5, 0.4)$, добавили третью точку – расширили гиперпрямоугольник 1 класса



(г) $B_1 = (0.2, 0.2), T_1 = (0.4, 0.5)$, $B_2 = (0.1, 0.3)$, $T_2 = (0.5, 0.4)$, добавили четверую точку - расширили гиперпрямоугольник 2 класса



(д) $B_1 = (0.2, 0.2), T_1 = (0.4, 0.3)$, $B_2 = (0.1, 0.3), T_2 = (0.5, 0.4)$, после обнаружения пересечения сжали один из гиперпрямоугольников

Рисунок 1 — Пример процесса обучения нечеткого классификатора

Рисунок 1 демонстрирует процесс обучения нечеткого классификатора на четырех двумерных точках, являющихся представителями двух разных классов:

x	y	c_i
0.2	0.2	1
0.5	0.4	2
0.4	0.5	1
0.1	0.3	2

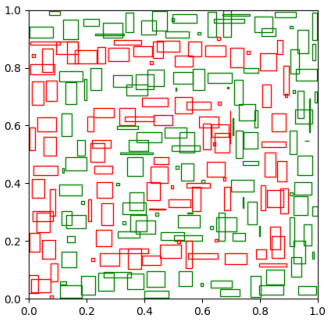
2.6 Нечеткий классификатор в качестве нейросети

Описанный выше алгоритм можно представить в виде программного воплощения — трехслойной нейронной сети. Входной слой $G_A = (a_1, a_2, \ldots, a_n)$ включает n обрабатываемых координат — одну для каждой размерности входного рисунка A_h. Есть два набора связей между каждым входным узлом и каждым из m узлов, соответствующих нечетким множествам во втором слое $G_S = (s_1, s_2, \ldots, s_m)$. Эти двойные связи устанавливаются, используя алгоритм нечеткого классификатора. Первый набор связей между G_A и j-ым узлом G_S — вектор B_j, соответствует минимуму, второй — вектор T_j, соответствует максимуму. Связь между узлами второго слоя G_S и p узлами третьего слоя $G_C = (c_1, c_2, \ldots, c_p)$ определяется добавлением G_S к нужному классу в ходе обучения. Каждый узел G_C представляет собой отдельный класс. Таким образом на входе нейронной сети подается входной рисунок (вектор) A_h, а на выходе мы получаем класс, к которому данный рисунок принадлежит.

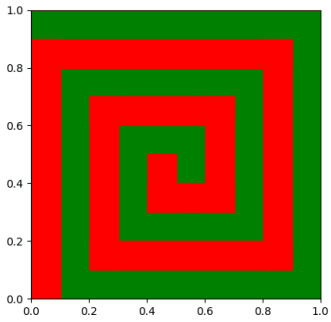
Заключение

На основании освоения полученной темы, алгоритм был реализован мной на языке программирования python и обучен на двух моделях.

Первая модель — это спираль. Для обучения данной модели было взято 1000 точек, принадлежащих к разным классам.



(а) схематичное изображение гиперпрямоугольников, полученное в ходе работы программы

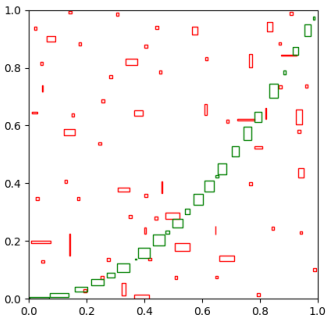


(б) реальное расположение классов

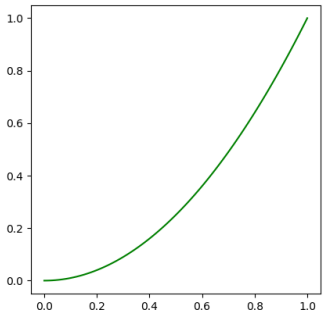
Рисунок 2 — Первая модель

На рисунке 2(а) можно наблюдать схематичное изображение нечетких гиперпрямоугольников, полученное в ходе обучения программы; на рисунке 2(б) реальное расположение классов. Сравнивая рисунки 2(а) и 2(б), можно сказать, что полученный рисунок довольно точно соответствует реальному расположению классов. После обучения, сеть была проверена на 100 точках, все точки были распознаны правильно, 96 из них со степенью принадлежности к классу более 98%. Вторая модель представляет собой параболу (1-ый класс) и квадрат размером $[0, 1] \times [0, 1]$ (2-ой класс). Основной целью данной модели является распознование параболы. Выборка представляет собой 250 точек. После обучения, нейросеть была проверена на 25 точках. Большинство точек алгоритм распознал правильно, несколько точек (те, которые находились между двумя гиперпрямоугольниками разных классов) были распознаны неверно.

Резюмируя полученные результаты, можно сделать вывод, что при большой выборке алгоритм обучается с высокой точностью. Описанная выше реализация подходит для двумерного случая, развивая данное исследование можно расширить алгоритм для больших размерностей.



(а) парабола, полученная программой



(б) функция $y = x^2$

Рисунок 3 — Вторая модель

Результаты проведенных исследований показали, что нечеткий классификатор может быть использован с практической точки зрения в более продвинутых моделях, например в прогнозировании, анализе данных, распознавании образов.

Список использованных источников

1. *Zadeh L. A.* Fuzzy sets // Inform. and Control. — 1965. — Vol. 8. — P. 338–353.
2. *Simpson P.* Fuzzy min-max classification with neural networks // Heuristics. — 1991. — Vol. 4, no. 1. — P. 1–9.
3. *Simpson P.* Fuzzy min-max neural networks // Int. Joint Conf. Neural Networks (Singapore). — 1991.
4. *Simpson P.* Fuzzy min-max neural networks - Part 2: Classification, submitted to IEEE Trans. Fuzzy Syst. — 1992.
5. *Simpson P.* Fuzzy min-max neural networks - Part 2: Clustering, submitted to IEEE Trans. Fuzzy Syst. — 1992.