Документация к коду mmclassifier.py

Для того, чтобы разобраться, что происходит в коде, предлагаю сначала изучить другие файлы.

Первым делом предлагаю открыть код functions.py, в котором лежит реализация функций, необходимых для работы mmclassifier.py

Далее в описании я буду ссылаться на работу Симпсона Fuzzy Min-Max Neural Networks (мы ее с вами ранее обсуждали, думаю я ее вам так же вышлю). (\*\*)

Сначала я указываю размерность, которая лежит в переменной n,

я писал код для двумерного случая, поэтому n = 2, но при желании можно изменить и на большее число размерностей.

Затем я завожу переменную maximum\_size, в которой лежит максимальный линейный размер гиперпрямоугольника (к примеру для двумерного случая максимальная площадь будет ограничена maximum\_size^2).

Далее указываю sensitivity\_parameter, отражает размер, области в которой точка принадлежит данному гиперпрямоугольнику.

Параметры maximum\_size, sensitivity\_parameter влияют на точность при обучении, поэтому их необходимо менять в зависимости от размеров области, на которой происходит обучение.

Далее идет функция HyberboxExpansionTest(v, w, x), которая проверяет можно ли расширить гиперпрямоугольник, принцип работы данной функции описан в \*\* на страницах 781-782 в пункте A. Hyberbox Expansion.

Следующая функция HyberboxExpansion, расширяет гипепрямоугольник, ее алгоритм также описан в \*\*

на стр. 782 в том же пункте A.

Затем идет функция HyperboxOverlapTestAndContraction(v1, w1, v2, w2),

которая проверяет есть ли у двух гиперпрямоугольников пересечение, если есть, сжимает их оптимальным образом.

Алгоритм функции описан в пунктах B. Hyberbox Overlap Test и C. Hyberbox Contraction соответственно, на стр. 782

Далее идет функция membership, проверяет на сколько точка принадлежит гиперпрямоугольнику, принцип работы описан в \*\* на стр. 780 в пункте B. Hybercube Membership Function

Перейдем к файлу drawH.py

Для анимации обучения гиперпрямоугольника использовал библиотеку matplotlib (pylab в частности).

Первая функция drawHyberbox добавляет прямоугольник в axes, по заданным v и w, прямоугольник может иметь два цвета в зависимости от класса, к которому он принадлежит.

Во второй функции draw\_array\_of\_hyberbox через цикл for передается массив прямоугольников.

Отрисовка происходит непосредственно при обучении.

Теперь перейдем к файлу, в котором непосредственно происходит обучение – mmclassifier.py

Первоначально импортируем functions.py и drawH.py, необходимые для обработки и отрисовки, а также pylab, чтобы задать фигуру, где все будет рисоваться.

Далее заведено два массива, в которых хранятся наши гиперпрямоугольники разных классов.

Затем идет функция training, она принимает на вход текстовый файл, в нем должны содержатся точки и класс, к которому они принадлежат. В моей программе функция в дальнейшем вызывается с файлом “input.txt”, в котором изначально лежит пример из \*\* (страница 782, пункт D. Examples of the Expansion-Contraction Process)

Рассмотрим, что же непосредственно происходит в функции training. Получая из input.txt точку и зная, что она принадлежит условно к классу a, сначала она проверяет, можно ли расширить уже существующие гиперпрямоугольники класса a (пробегается циклом for). Если да, запускается проверка на пересечение расширенного гиперпрямоугольника с гиперпрямоугольниками из другого класса и сжатие(пробегается циклом for). Внешний цикл прерывается.

Если ни один гиперпрямоугольник из класса a расширить невозможно, то циклом for проверяем лежит ли наша точка в каком-либо гиперпрямоугольнике другого класса и сжимаем, если нужно. Затем добавляем в массив класса a.

Аналогично со вторым классом.

Далее выводим наши массивы через print, чтобы мы могли наглядно видеть процесс обучения.

В конце небольшой кусок, который отвечает за анимацию. Мы складываем все гиперпрямоугольники в один массив, и в другой записываем к какому классу принадлежит i-ый гиперпрямоугольник. Получаются как раз два массива для функции draw\_array\_of\_hyberbox, в начале training мы уже завели необходимый axes (фигура для отрисовки). И далее на каждой итерации отображается текущая картина.

Следующая функция DefineClassOfDot пробегает массивы первого и второго классов соответственно, находит к какому гиперпрямоугольнику точка принадлежит в большей степени через функцию membership. Выводит результат.

Предлагается попробовать запустить программу и посмотреть, как идет процесс обучения. (Для того, чтобы все нормально заработало нужно сложить все в одну папку, и чтобы у вас был установлен matplotlib (pip install matplotlib))

Проблемы, возникшие при написании кода:

Самая большая и все еще нерешенная проблема – это анимация обучения, я перерыл пол интернета, но пока не понял, как сделать так, чтобы не надо было закрывать окна фигуры на каждой итерации, и анимация работала сама по себе.

Также я сначала считал, что возникнут проблемы из-за отсутствия указателей в python (функции HyberboxExpansion и HyperboxOverlapTestAndContraction должны изменять сам массив), но их не возникло, так как type list изменяемый.

Это пока сырая версия, просто чтобы посмотреть, как в целом выглядит и работает программа, код еще не тестировался на больших моделях.