РАСПОЗНАВАНИЕ ОБРАЗОВ НА ОСНОВЕ САМООБУЧЕНИЯ

Цель работы: изучить особенности распознавания образов в самообучающихся системах и научиться классифицировать объекты с помощью алгоритма *максимина*.

Порядок выполнения работы

1. Изучение теоретической части лабораторной работы.
2. Реализация алгоритма *максимина*.
3. Защита лабораторной работы.

По сравнению с методами контролируемого обучения алгоритмы самообучения отличаются большей неполнотой информации. В этих алгоритмах не известны ни классы, ни их количество, ни признаки. Необходимым минимумом информации для классификации объектов являются сами образы и их признаки, без этого не выполняется ни один алгоритм. В обучении «без учителя» алгоритм самостоятельно определяет классы, на которые делится исходное множество данных, и одновременно определяет присущие им признаки. Для разделения данных используется следующий универсальный критерий. Процесс организуется так, чтобы среди всех возможных вариантов группировок найти такой, когда группы обладают наибольшей компактностью.

В качестве примера метода распознавания образов, использующего процедуру самообучения, рассмотрим алгоритм *максимина*.

Исходные данные – число образов, которые нужно разделить на классы. Количество образов предлагается брать в диапазоне от 1000 до 100 000. Признаки объектов задаются случайным образом, это координаты векторов.

Цель и результат работы алгоритма – исходя из произвольного выбора максимально компактно разделить объекты на классы, определив ядро каждого класса.

Примечание. Результат работы представить графически.

На рис. 2.3 показан пример реализации алгоритма *максимина* в случае распределения по классам 20 000 объектов. В результате было определено 8 классов образов.

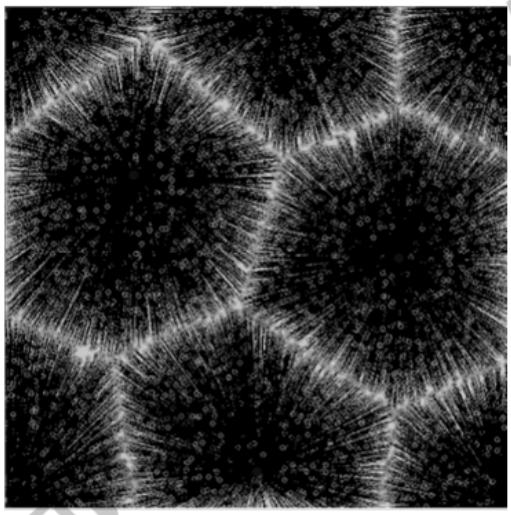

Рис. 2.3. Результат работы алгоритма максимина

Алгоритм максимина

1. Из множества векторов $X = \{X(1), X(2), X(3),...,X(V)\}$ произвольно выбирается один и назначается ядром первого класса. Пусть $N_1 = X(1)$. Затем будут определяться другие ядра $N_2, N_3,...,N_m$, число которых заранее неизвестно.

2. Вычисляются расстояния $d_{1i}(\overline{N}_1, \overline{X}(i)) \forall i \neq 1$. Ядро N_2 выбирается следующим образом: $\overline{N}_2 = \overline{X}(l)$, где $d_{1l} = \max d_{1i}(\overline{N}_1, \overline{N}(i))$.

3. Выполняется распределение оставшихся объектов по классам по критерию минимального расстояния.

4. В каждом классе вычисляются расстояния от ядра до каждого объекта данного класса: $d_{ki} = d(\overline{N}_k, \overline{X}(i)), k = 1, 2; i = 1, 2, \ldots, \nu - k,$ среди которых находятся наибольшие $\delta_{ki} = \max(d_{ki}), k = 1, 2$ (пока имеется два максимума).

5. Выбирается максимальное среди всех максимальных расстояний, которое становится претендентом на очередное ядро. Это значение δ_{kp}. Если δ_{kp} больше половины среднего арифметического расстояния между всеми ядрами, то создается очередное ядро $\overline{N}_3 = \delta_{kp} = X(p)$ и выполняется переход к шагу 3, иначе алгоритм останавливается.

Комментарий. Новое ядро вводится по следующим соображениям: N_1 и N_2 – ядра двух классов, а один из векторов X удален от одного из этих ядер на расстояние, превышающее половину расстояния между ядрами. Следовательно, $\overline{X}$ не относится ни к одному из существующих классов и становится ядром очередного класса. Алгоритм останавливается, когда ни в одном из классов не будет найден объект, для которого выполнится условие из шага 5. К этому моменту найдено m классов и их ядра $N_1, N_2, \ldots, N_m$.