ТУБ 2

2.1 Алгоритм К-средних

Процесс распознавания образов напрямую связан с процедурой обучения. Главная особенность контролируемого обучения заключается в обязательном наличии априорных сведений о принадлежности к определенному классу каждого вектора измерений, входящего в обучающую выборку. Роль обучающего состоит в том, чтобы помочь отнести каждый вектор из тестовой выборки к одному из имеющихся классов. И хотя классы известны заранее, необходимо уточнить и оптимизировать процедуры принятия решений. В основу всех алгоритмов распознавания образов положено понятие «расстояние», выступающее критерием в ходе принятия решений.

Важную роль в алгоритмах распознавания образов играют объекты-центры (ядра) классов. Ядрами являются объекты, которые характеризуются типичными признаками своего класса. Геометрически они часто находятся в центре класса, поэтому называются объекты-центры. Принимая решение о том, к какому классу отнести тот или иной объект, его сравнивают с ядром каждого класса и помещают в тот из них, где оказалось минимальное расхождение с признаками ядра. Сравнение выполняется путем нахождения расстояний между объектами. Значение ядер для качественного решения задачи распознавания образов объясняет тот факт, что более ценной информацией является наличие нескольких объектов, характеризующихся типичными признаками классов, чем много образов с «размытыми» признаками. Если стоит задача нахождения расстояния между классами, то корректнее всего искать ее решение, определяя расстояние между ядрами классов. Поскольку именно они характеризуются определяющими признаками классов.

В качестве примера метода распознавания образов, использующего процедуру контролируемого обучения, рассмотрим алгоритм *К*-средних.

Исходные данные: число образов (векторов) и число классов (*к*), на которое нужно разделить все образы. Количество образов предлагается брать в диапазоне от 1000 до 100000, число классов – от 2 до 20. Признаки объектов задаются случайным образом, это координаты векторов. Обычно *к* элементов из набора векторов случайным образом назначают первоначальными центрами классов.

Цель алгоритма – определить ядрами классов *к* типичных представителей классов и максимально компактно распределить вокруг них остальные объекты выборки.

На рисунках 1 и 2 показаны примеры реализации алгоритма *к*-средних в случае распределения 20000 объектов на 6 класса. Рисунок 1 иллюстрирует первую итерацию алгоритма, а рисунок 2 – завершающую итерацию.

Алгоритм К-средних

1. Фиксируются $к$ ядер (центров областей). Затем вокруг них формируются области по правилу минимального расстояния. На r-ом этапе вектор $\overline{X}_p$ связывается с ядром $\overline{N}_i(r)$, если выполняется следующее неравенство:

$$\left\|\overline{X}_p - \overline{N}_i(r)\right\| < \left\|\overline{X}_p - \overline{N}_j(r)\right\| \forall i \neq j., \text{ тогда } \overline{X}_p \in \overline{N}_i(r).$$

2. На $r+1$ этапе определяются новые элементы, характеризующие новые ядра $\overline{N}_i(r+1)$. За их значения принимают векторы $\overline{X}$, обеспечивающие минимум среднеквадратичного отклонения:

$$J_i = \sum_{\overline{X}_p \in \overline{N}_i(r)} \left\|\overline{X}_p - \overline{N}_i(r+1)\right\|^2, i = 1,2,\ldots K.$$

J_i принимает минимальное значение лишь при одном $\overline{X}$, равном среднему арифметическому векторов, принадлежащих одной области N_i.

3. Если хотя бы в одной из областей поменялось положение ядра, то пересчитываются области принадлежащих им векторов, т.е. определяются расстояния от объектов не ядер до новых ядер. В результате этого может произойти перераспределение областей. Затем повторяется шаг 2. Процедура заканчивается, если на $(r+1)$ шаге ее выполнения положения центров областей не меняются по сравнению с r шагом.

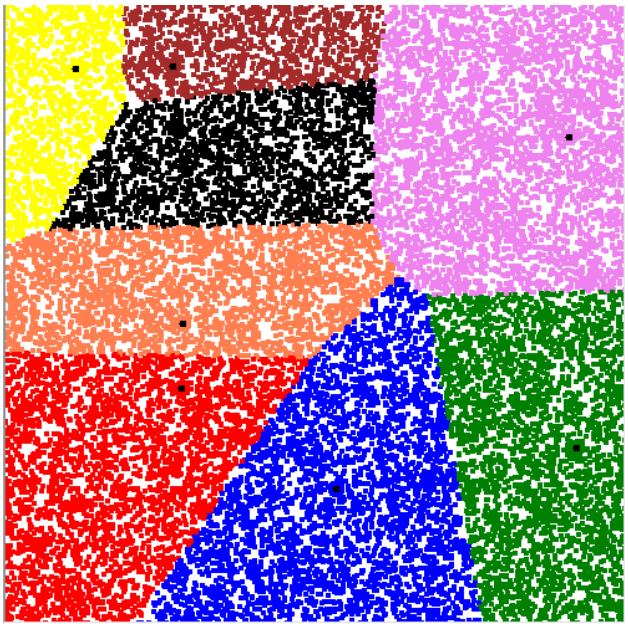

Рисунок 1 – Начальное распределение объектов в алгоритме *к*-средних

На рис. 2 показан пример завершения работы алгоритма *к*-средних, по которому можно судить о корректном выполнении алгоритма. Все области на рисунке имеют приблизительно одинаковые размеры, и в геометрическом

центре каждой из них находится ядро класса. Кроме того, построенные области характеризуются *компактностью* и *сепарабельностью*, что является признаками качественно выполненной классификации образов.

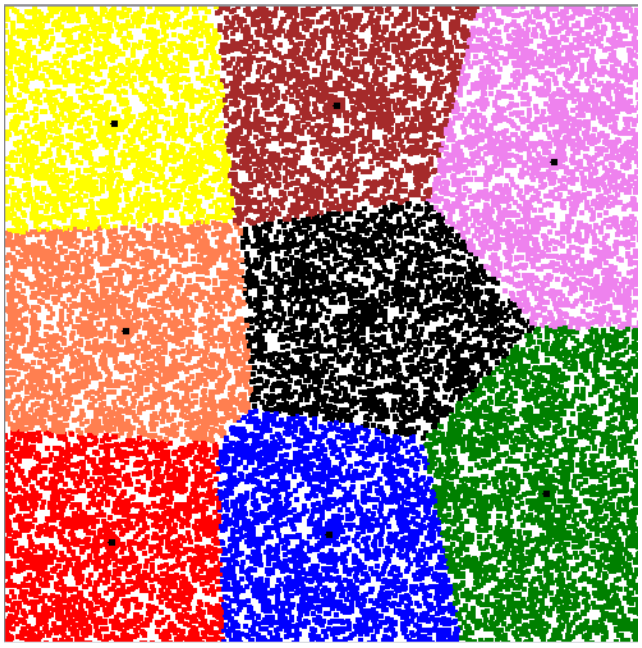

Рисунок 2 – Результат работы алгоритма *к*-средних

2.2 Алгоритм максимина

По сравнению с методами контролируемого обучения алгоритмы самообучения отличаются большей неполнотой информации. В этих алгоритмах не известны ни классы, ни их количество, ни признаки классов. 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 и сепарабельностью.

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 100000. Признаки объектов задаются случайным образом, это координаты векторов.

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

На рисунке 3 показан пример реализации алгоритма *максимина* в случае распределения по классам 20000 объектов. В результате было определено 8 классов образов.

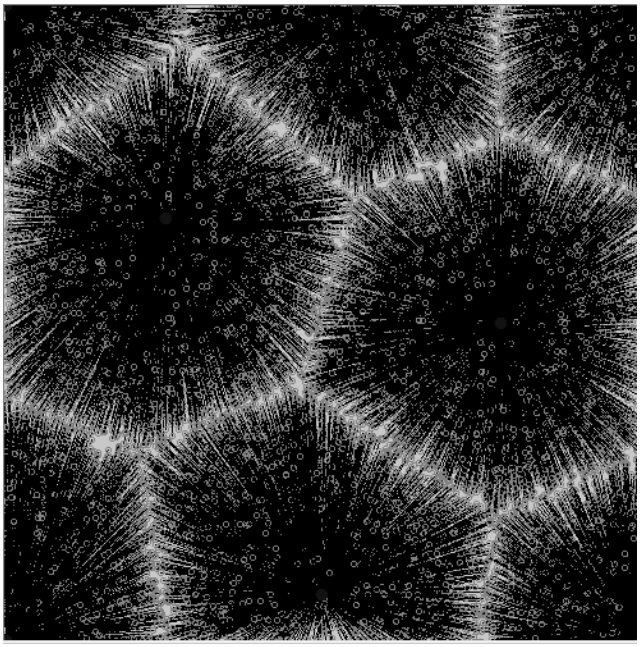

Рисунок 3 – Результат работы алгоритма максимина

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. $d_{ki} = d(\overline{N}_k, \overline{X}(i)), k=1,2; i=1,2,...v-k,$ среди которых находятся наибольшие $\delta_{ki} = \max(d_{ki}), k=1,2$ (пока имеется два максимума).

5. Выбирается максимальное среди всех максимальных расстояний и объект, находящийся на этом расстоянии от своего ядра, 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 условия из шага 5. К этому моменту найдено m классов и их ядра: N_1, N_2… N_m.

Сравнивая рисунки 2 и 3, несложно заметить формальные различия между ними. По завершении алгоритма максимина построенные области имеют разные размеры, также ядра классов не расположены в геометрических центрах классов.

К достоинствам алгоритма *к*-средних можно отнести его точность в распознавании образов, которая обеспечивается наличием большей исходной информации. Поскольку решения принимаются в условиях неполноты информации, точность является одним из главных требований к результату. К достоинствам алгоритма *максимина* относится его более высокая адаптивность к исходным данным. Являясь алгоритмом самообучения, максимин справляется с решением задачи в тех ситуациях, когда алгоритм *к*-средних не может принять решение из-за недостатка информации. Зная «сильные» и «слабые» стороны каждого алгоритма, их рекомендуют использовать вместе. Сначала работает *максимин,* выполняя предварительное разбиение объектов на классы, затем его результат подается на вход алгоритма *к*-средних, который корректирует размеры областей и их ядра, добиваясь компактной и сепарабельной классификации.