Алгоритмы обучения без учителя

Алгоритм MAXMIN

Рассмотрим алгоритм, более эффективный по сравнению с предыдущим и являющийся улучшением порогового алгоритма. Исходными даннымы для работы алгоритма будет, как и раньше, выборка X. Объекты этой выборки следует разделить на классы, число и характеристики которых заранее неизвестны.

Алгоритм MAXMIN

На первом этапе алгоритма все объекты разделяются по классам на основе критерия минимального расстояния от точек-прототипов этих классов (первая точка-прототип может выбираться произвольо). Затем в каждом классе выбирается объект, наиболее удаленный от своего прототипа. Если он удален от своего прототипа на расстояние, превышающее пороговое, такой объект становится прототипом нового класса.

В этом алгоритме пороговое расстояние не является фиксированным, а определяется на основе среднего расстояния между всеми точками-прототипами, то есть корректируется в процессе работы алгоритма. Если в ходе распределения объектов выборки *X* по классам были созданы новые прототипы, процесс распределения повторяется. Таким образом, в алгоритме MAXMIN окончательным считается разбиение, для которого в каждом классе расстояние от точки-прототипа до всех объектов этого класса не превышает финального значения порога *T.*

Алгоритм

1. Выбрать точку-прототип первого класса (например, объект X_1 из обучающей выборки). Количество классов K положить равным 1. Обозначить точку-прототип Z_1.
2. Определить наиболее удаленный от Z_1 объект X_f по условию

$$D(Z_1,X_f) = max\ D(Z_1, X_i),$$

где $D(Z_1,X_f)$ - расстояние между Z_1 и X_f, вычисленное одним из возможных способов. Объявить X_f прототипом второго класса. Обозначить X_f как Z_2. Число классов $K = K + 1$.

Алгоритм

3. Находим пороговое расстояние T.
4. Для всех объектов обучающего множества строится матрица расстояний до каждого из имеющихся прототипов: $D(X_i, Z_k),\ i = 1, \ldots M,\ k = 1, \ldots, K$.
5. Каждый объект относится к классу по критерию наибольшей близости к точке-прототипу: X_i *отнесен к классу* p, *если* $D(X_i, Z_p) = \min\limits_k D(X_i, Z_k)$.
6. В каждом классе k определяется объект X_{lk}, наиболее удаленный от точки-прототипа: $D(X_{lk}, Z_k) = \max\limits_{X \in R_k} D(X, Z_k)$, где R_k — множество объектов класса k.

Алгоритм

7. Для всех найденных объектов проверяется условие: $D(X_{lk}, Z_k) < T, k = 1, \ldots, K$.

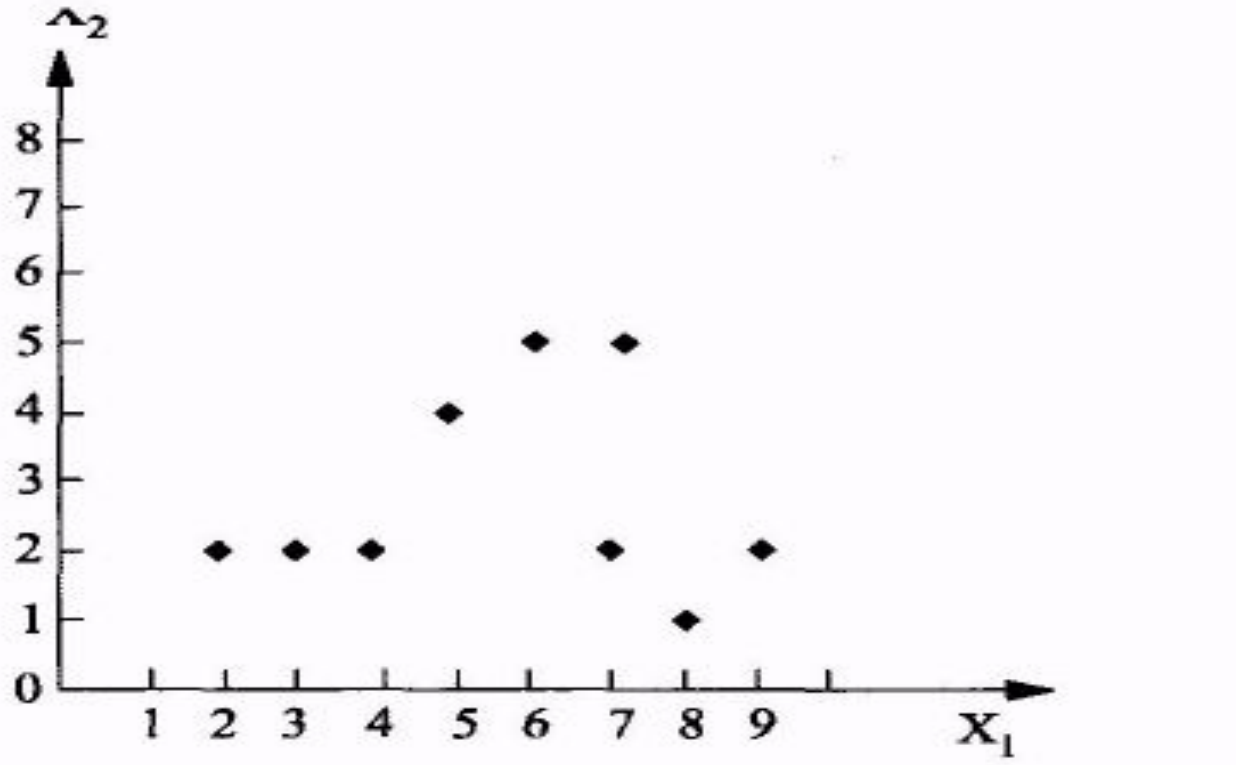
   Если для некоторого X_{lk} это условие не выполнено, он становится точкой-прототипом нового класса, число классов $K = K + 1$.

8. Если новых классов не создано, то КОНЕЦ. Иначе — перейти к шагу 9.

9. Вычисляется новое значение T как среднее расстояние между прототипами.

10. Перейти к шагу 4.

Рассмотрим работу алгоритма MAXMIN на примере. Как и в предыдущем случае выберем объекты, которые заданы двумя признаками. Обучающая выборка представлена на рис.



	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
Координаты точек	(2; 2)	(3; 2)	(4; 2)	(5; 4)	(6; 5)	(7; 2)	(7; 5)	(8; 1)	(9; 2)
Первая точка-прототип	Z_1								
Расстояние $D(Z_1, X_j)$		1	2	3,6	5	5	5,8	6,1	7

В качестве первой точки произвольно выбирается X_1. В таблице даны расстояния от этой точки до остальных. Наиболее удаленной от Z_1 будет X_9. Пороговое расстояние $T = \frac{1}{2}D(X_1, X_9) = 3{,}5$. Точка X_9 объявляется прототипом второго класса и обозначается Z_2.

Матрица расстояний для двух классов представлена в табл. 12.4.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
Координаты точек	(2; 2)	(3; 2)	(4; 2)	(5; 4)	(6; 5)	(7; 2)	(7; 5)	(8; 1)	(9; 2)
Первая точка-прототип	$\boldsymbol{Z_1}$								
Расстояние $D(Z_1, X_j)$		1	2	3,6	5	5	5,8	6,1	7

Таблица 12.4

	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Координаты	(3; 2)	(4; 2)	(5; 4)	(6; 5)	(7; 2)	(7; 5)	(8; 1)
Расстояние $D(Z_1, X_j)$ $Z_1 = (2, 2)$	1	2	3,6	5	5	5,8	6,1
Расстояние $D(Z_2, X_j)$ $Z_2 = (9, 2)$	6	5	4,5	4,3	2	3,6	1,4

Разделение на классы по критерию минимального расстояния до точки-прототипа дает следующий результат:

класс 1 $= \{X_1, X_2, X_3, X_4\}$, класс 2 $= \{X_5, X_6, X_7, X_8, X_9\}$.

Найдем в каждом классе точку, максимально удаленную от прототипа. В классе 1 — это X_4, поскольку $D(Z_1, X_4) = 3{,}6 > T$ $(T = 3{,}5)$, X_4 становится прототипом нового — третьего класса и обозначается далее также Z_3. Новая итерация начинается с вычисления порогового расстояния и построения новой матрицы расстояний.

$$T = \frac{D(X_1, X_9) + D(X_1, X_4) + D(X_4, X_9)}{6} = \frac{7 + 3{,}6 + 5}{6} = 2{,}6$$

Матрица расстояний для трех классов представлена в табл. 12.5.

	X_2	X_3	X_5	X_6	X_7	X_8
Координаты	(3; 2)	(4; 2)	(6; 5)	(7; 2)	(7; 5)	(8; 1)
Расстояние $D(Z_1, X_j)$ $Z_1 = (2, 2)$	1	2	5	5	5,8	6,1
Расстояние $D(Z_2, X_j)$ $Z_2 = (9, 2)$	6	5	4,3	2	3,6	1,4
Расстояние $D(Z_3, X_j)$ $Z_3 = (5, 4)$	2,8	2,3	1,4	2,8	2,3	4,1

Разделение точек на классы даст следующий результат:

класс 1 – $\{X_1, X_2, X_3\}$, класс 2 – $\{X_6, X_8, X_9\}$, класс 3 – $\{X_4, X_5, X_7\}$.

Найдем в каждом классе точку, максимально удаленную от прототипа. В классе 1 — это X_3. Расстояние $D(Z_1, X_3) = 2 < T = 2{,}6$. В классе 2 искомая точка — X_6. Так как $D(Z_2, X_6) = 2$, для второго класса условие выполнено. В классе 3 наиболее удаленной является точка X_7. $D(Z_3, X_7) = 2{,}3 < T$. В соответствии с пунктом 8 алгоритма новый класс не создается, и алгоритм завершает работу.

Алгоритм позволяет получить лучшее разбиение точек на классы, поскольку разбиение многократно уточняется. Это снижает чувствительность алгоритма к ошибкам в обучающем множестве, а также к выбору порядка рассмотрения объектов. Недостатком является необходимость многократно вычислять расстояние между объектами.

12.3. Алгоритм «K средних»

Рассмотрим алгоритм, решающий задачу обучения «без учителя» при одном дополнительном условии: количество классов, к которым могут принадлежать элементы обучающей выборки, заранее известно (классов — K). В такой ситуации первоначально выбираются K точек-прототипов (например, первые K объектов обучающей выборки $\tilde{X}$). Затем все объекты обучающей выборки $\tilde{X}$ распределяются по классам по критерию наименьшего расстояния от точек-прототипов этих классов. В каждом из сформированных классов определяется

новая точка-прототип как «средняя» точка данного класса. Если точки-прототипы при этом изменяются, то распределение объектов по классам выполняется заново. Признаком окончания работы алгоритма здесь будет совпадение разбиений на классы на двух последовательных итерациях. В окончательном разбиении каждый объект из $\tilde{X}$ должен оказаться ближе к «центру» то есть к прототипу своего класса, чем к прототипу любого другого класса. Заметим также, что

в алгоритме «K средних» точки-прототипы могут не совпадать ни с одним реальным объектом из $\tilde{X}$, поскольку они определяются как «средние» точки своих классов.

Алгоритм 12.3.

1. Обозначим номер итерации $s = 0$. Выбираются K начальных точек-прототипов:

$$\tilde{Z} = \{Z_1^s, Z_2^s, \ldots, Z_K^s\} \text{ из } \tilde{X}.$$

2. Выполняется итерация $s = s + 1$.

3. Строится матрица расстояний. Каждый объект $X_l \in \tilde{X}$ относится к одному из классов $1, 2, \ldots K$ по признаку ближайшего расстояния до точки-прототипа.
4. Для каждого из сформированных классов вычисляется новая точка-прототип. Для этого значение каждого признака для точки-прототипа класса k определяется как среднее арифметическое значений этого признака по всем объектам k-го класса, сформированного на текущей итерации s:

$$x_{kj}^{s} = \frac{1}{|R_k^s|} \sum_{X \in R_k^s} x_j,$$

где x_{kj}^{s} — значение j-го признака для определяемой точки-прототипа класса k, $(k = 1, 2, \ldots, K)$, R_k^s — множество объектов, отнесенных к классу k на s-й итерации, $|R_k^s|$ — мощность этого множества. $j = 1, 2, \ldots, n$, поскольку каждый объект обучающей выборки содержит n признаков.

5. Если для всех $(k = 1, 2, \ldots, K)$ верно, $Z_k^s = Z_k^{s-1}$, то КОНЕЦ, иначе — переход к шагу 2.

Пример 12.3.

Рассмотрим пример работы данного алгоритма. Обучающая выборка представлена в табл. 12.6. Требуется разбить эту выборку на три класса ($K = 3$).

Таблица 12.6

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
Координаты	(2; 2)	(2; 4)	(3; 4)	(3; 8)	(4; 7)	(5; 9)	(6; 1)	(6; 8)	(7; 1)	(8; 1)

Прежде всего, выберем начальные точки-прототипы. Пусть $Z_1 = X_1$, $Z_2 = X_3$, $Z_3 = X_4$. Выполняем итерацию 1. Матрица расстояний от точек-прототипов до остальных объектов представлена в табл. 12.7.

Таблица 12.7

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
Координаты	(2; 2)	(2; 4)	(3; 4)	(3; 8)	(4; 7)	(5; 9)	(6; 1)	(6; 8)	(7; 1)	(8; 1)
Z_1	0	2	–	–	5,4	7,6	4,1	7,2	5,1	6,05
Z_2	–	1	0	–	3,2	5,4	4,2	5	5	5,8
Z_3	–	4,1	–	0	1,4	2,3	7,6	3	8,05	8,6

Проведем разбиение на классы по критерию минимального расстояния: класс 1 – X_1, X_7; класс 2 – X_3, X_2, X_9, X_{10}; класс 3 – X_4, X_5, X_6, X_8. Расчет точек-прототипов для каждого класса даст значения $Z_1^1 = (4; 1{,}5)$, $Z_2^1 = (5; 2{,}5)$, $Z_3^1 = (4{,}5; 8)$.

Построим новую матрицу расстояний от точек-прототипов до остальных объектов (табл. 12.8).

Таблица 12.8

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
Координаты	$(2;2)$	$(2;4)$	$(3;4)$	$(3;8)$	$(4;7)$	$(5;9)$	$(6;1)$	$(6;8)$	$(7;1)$	$(8;1)$
$Z_1^1 = (4;1{,}5)$	2	3,2	2,7	6,6	5,5	7,6	2	6,8	3	4
$Z_2^1 = (5;2{,}5)$	3	3,35	2,5	5,85	4,6	6,5	1,8	5,6	2,5	3,4
$Z_3^1 = (4{,}5;8)$	6,5	4,7	4,3	1,5	1,1	1,1	7,2	1,5	7,4	7,8

Новое разбиение на классы по критерию минимального расстояния до прототипа: класс 1 — X_1, X_2; класс 2 — X_3, X_7, X_9, X_{10}; класс 3 — X_4, X_5, X_6, X_8 . Рассчитаем координаты точек-прототипов на итерации 2: $Z_1^2 = (2;3)$, $Z_2^2 = (6;1{,}75)$, $Z_3^2 = (4{,}5;8)$.

Итерация 3. Снова построим матрицу расстояний (табл. 12.9). Легко заметить, что класс 3 уже стабилен.

Таблица 12.9

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
Координаты	$(2;2)$	$(2;4)$	$(3;4)$	$(3;8)$	$(4;7)$	$(5;9)$	$(6;1)$	$(6;8)$	$(7;1)$	$(8;1)$
$Z_1^1 = (2;3)$	1	1	1,4	5,1	4,47	6,7	4,47	6,4	5,4	6,3
$Z_2^1 = (6;1{,}75)$	4	4,6	3,75	6,9	5,6	7,3	0,75	6,25	1,25	2,1
$Z_3^1 = (4{,}5;8)$	6,5	4,7	4,3	1,5	1,1	1,1	7,2	1,5	7,4	7,8

На основании матрицы расстояний разбиваем выборку на классы: класс 1 – X_1, X_2, X_3; класс 2 – X_7, X_9, X_{10}; класс 3 – X_4, X_5, X_6, X_8. Точки-прототипы для классов имеют координаты $Z_1^3 = (2{,}33; 3{,}33)$, $Z_2^3 = (7{,}0; 1{,}0)$, $Z_3^3 = (4{,}5; 8{,}0)$.

Из рис. 12.3 видно, что классы окончательно сформированы. Выполнение четвертой итерации по тем же правилам даст разбиение, равное предыдущему, и алгоритм заканчивает работу. Заметим, что в качестве точек-прототипов полезно брать не первые K объектов из $\tilde{X}$, а максимально различные между собой объекты.

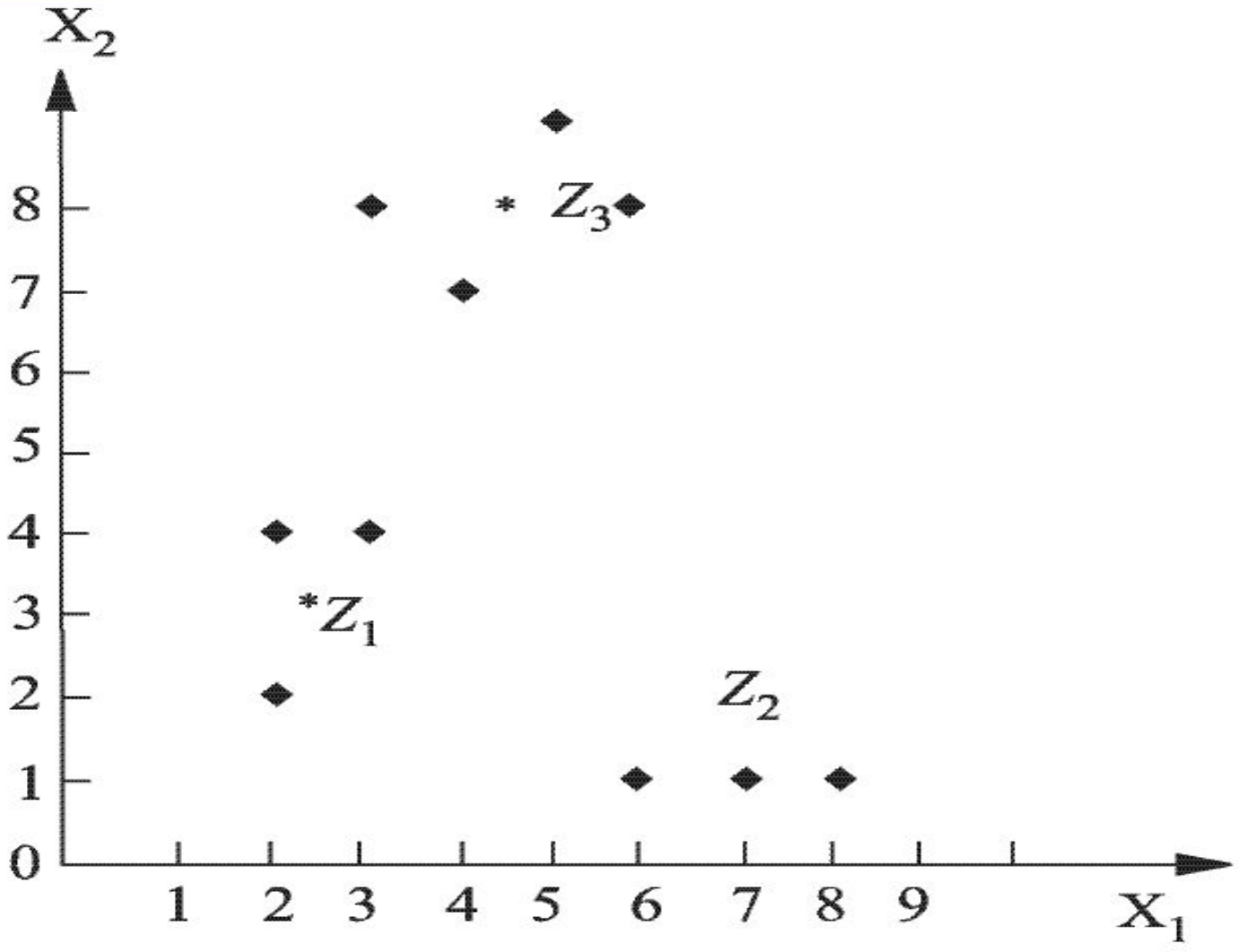


Рис. 12.3.

Проблемы алгоритма K-средних:

- необходимо заранее знать количество кластеров.
- алгоритм очень чувствителен к выбору начальных центров кластеров. Классический вариант подразумевает случайный выбор кластеров, что очень часто являлось источником погрешности. Как вариант решения, необходимо проводить исследования объекта для более точного определения центров начальных кластеров.
- не справляется с задачей, когда объект принадлежит к разным кластерам в равной степени или не принадлежит ни одному.

4.3. Алгоритмы расстановки центров кластеров

Для первоначальной расстановки центров кластеров применяются следующие алгоритмы, которые можно рассматривать и как самостоятельные алгоритмы кластеризации.

4.3.1. Алгоритм простейшей расстановки центров кластеров

Вводится некоторый порог $h > 0$, в качестве первого центра кластера назначается первый элемент выборки $\mathbf{c}_1 = \mathbf{x}_1$.

Предположим, что уже выбраны k центров кластеров. Тогда в качестве очередного $k+1$-го центра кластера выбирается такой элемент выборки $\mathbf{x}_j$, что минимальное расстояние от $\mathbf{x}_j$ до центров $\mathbf{c}_i$, $i = 1,...,k$, будет больше h.

4.3.2. Алгоритм, основанный на методе просеивания

В этом алгоритме рассматривается некоторая неотрицательная функция $f(\mathbf{x})$, называемая *плотностью распределения* элементов обучающей выборки и принимающая тем большее значение, чем ближе элемент $\mathbf{x}$ расположен к точке сгущения элементов выборки. Например, в качестве $f(\mathbf{x})$ можно взять следующую функцию:

$$f(\mathbf{x}) = f_h(\mathbf{x}) = \frac{1}{h^2} \cdot \sum_{i:\|\mathbf{x}-\mathbf{x}_i\|<h} \left(h^2 - \|\mathbf{x}-\mathbf{x}_i\|^2 \right),$$

где $h > 0$ – некоторое пороговое значение. Затем осуществляется упорядочивание элементов обучающей выборки таким образом, чтобы $f(\mathbf{x}_1) \geq f(\mathbf{x}_2) \geq f(\mathbf{x}_3) \geq \ldots$. Далее осуществляется алгоритм простейшей расстановки центров кластеров, в котором в первую очередь в качестве новых центров кластеров выбираются те элементы обучающей выборки, в которых значение плотности будет наибольшим.

Нечеткий алгоритм кластеризации c-means

С последней проблемой k-means успешно справляется алгоритм c-means. Вместо однозначного ответа на вопрос к какому кластеру относится объект, он определяет вероятность того, что объект принадлежит к тому или иному кластеру. Таким образом, утверждение «объект А принадлежит к кластеру 1 с вероятностью 90%, к кластеру 2 — 10% » верно и более удобно.

- Классический пример c-means — т.н. «бабочка» (butterfly):

The data set **M** consists of 15 points in the plane :

j	1	2	3	4	5	6	7	8
$\mathbf{m}_j$	(0,0)	(0,2)	(0,4)	(1,1)	(1,2)	(1,3)	(2,2)	(3,2)
j	9	10	11	12	13	14	15	
$\mathbf{m}_j$	(4,2)	(5,1)	(5,2)	(5,3)	(6,0)	(6,2)	(6,4)	

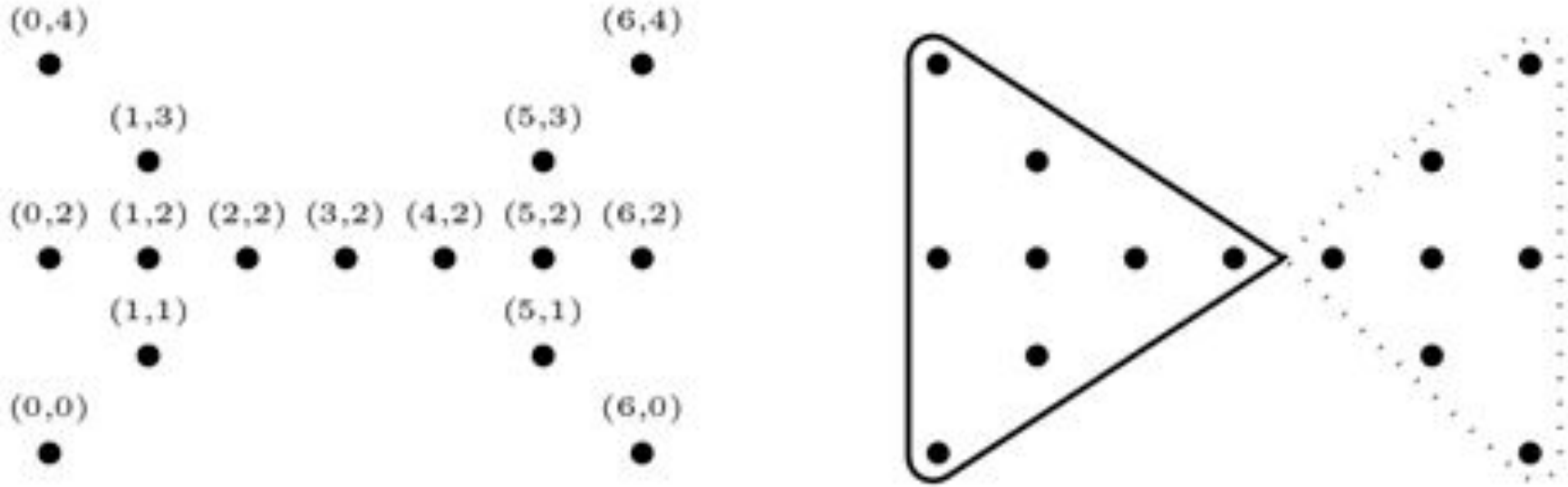


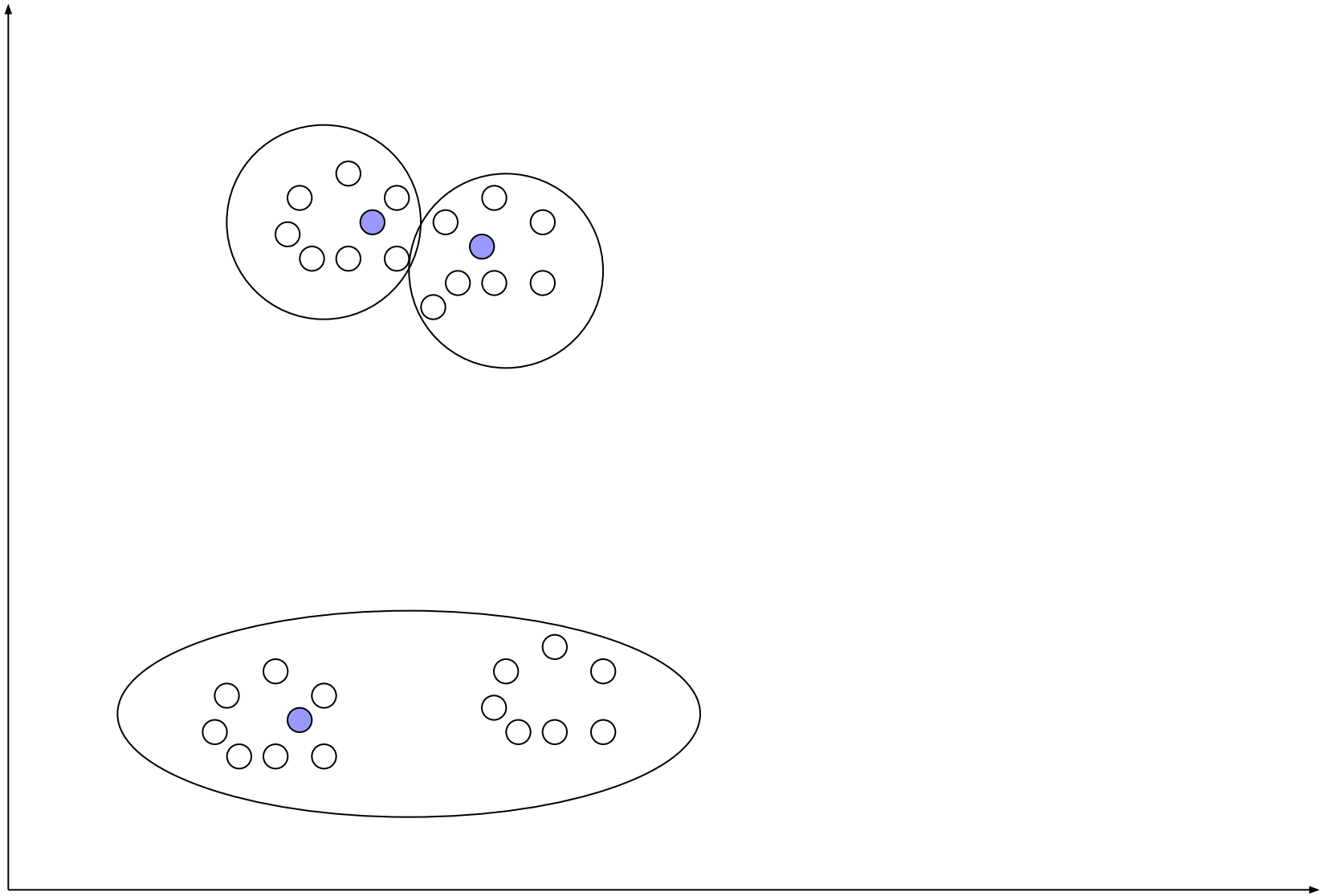
Figure 1: *The butterfly data set and hard-c-means result.*

- Алгоритм *ISODATA* (Iterative Self-Organizing Data Analysis Techniques) основывается на алгоритме *k* средних, но включает набор оказавшихся полезными на практике эвристик и параметры по их настройке. Одним из задаваемых априори параметров является желаемое число кластеров *K*.

Это число выступает в качестве рекомендации: в результате работы алгоритма может быть построено как меньшее, так и большее число кластеров, но оно будет не сильно отличаться от значения *K*. Сам алгоритм здесь детально описываться не будет (в целом, в нем используются те же шаги, что и в алгоритме *k* средних); приведем лишь основные эвристики.

- Ликвидируются кластеры, в состав которых входит менее чем заданное число элементов.
- Для каждого текущего кластера определяется направление максимальной вытянутости. Наиболее вытянутый кластер может быть расщеплен на два.

- Решение о расщеплении принимается с учетом размера кластера в направлении вытянутости (этот размер может сравниваться с фиксированным порогом и отклонением от среднего размера всех кластеров, а также общего числа кластеров, которое должно быть мало (с учетом параметра *K*).
- Попарно сливаются кластеры, расстояние между центрами которых меньше заданного порога, если число кластеров велико (с учетом параметра *K*).



- Использующиеся в алгоритме *ISODATA* эвристики помогают не только подбирать более подходящее число классов, но и находить более приемлемое решение, несколько ослабляя (но не убирая полностью) зависимость от начальной гипотезы.

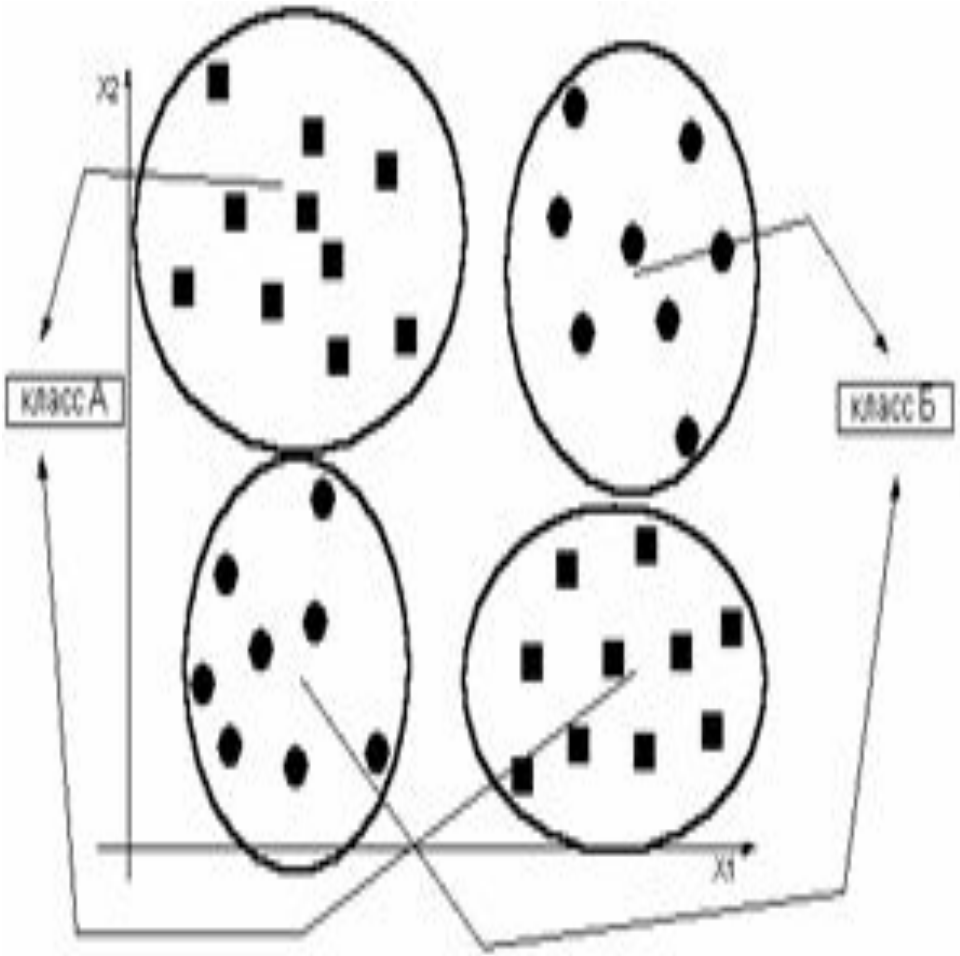
Алгоритмы, рассмотренные в этой главе, обладают одним общим свойством: с их помощью для обучающей выборки $\tilde{X}$ решается задача классификации, то есть строится разбиение примеров обучающей выборки на классы. Вспомним, что алгоритмы обобщения должны успешно решать и задачу распознавания, то есть задачу распределения вновь предъявляемых примеров по классам. Последнее требует наличия четко сформулированного решающего правила, или *алгоритма*, позволяющего отнести новый объект X к одному из классов.

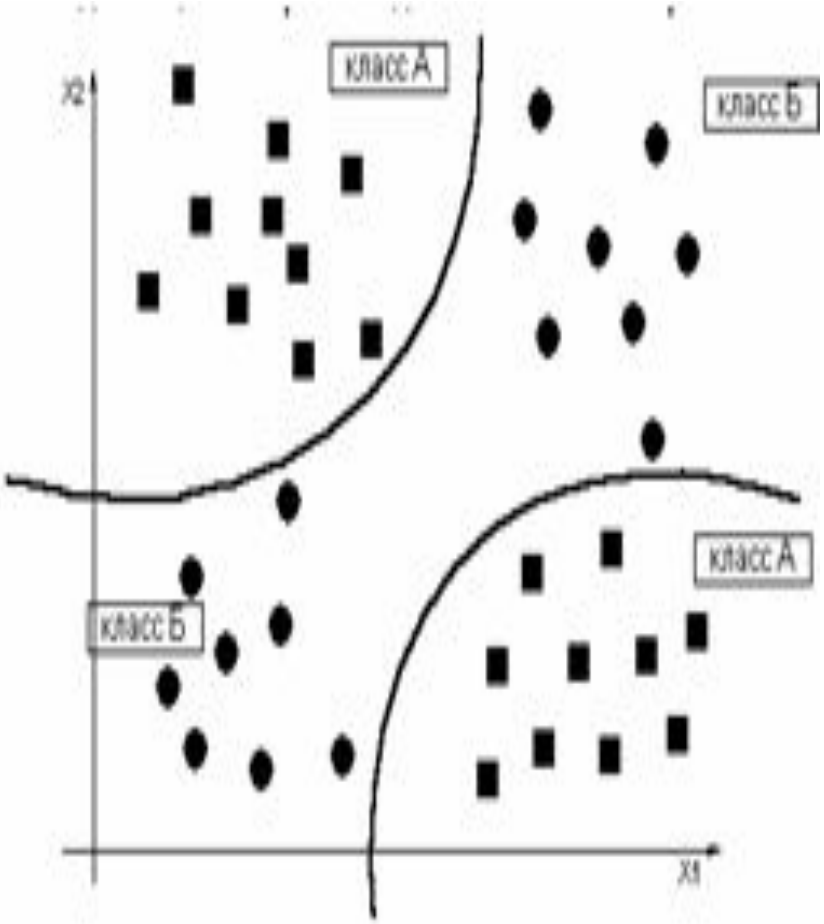
12.4. Распознавание с использованием решающих функций

Прежде всего необходимо дать понятие решающей функции. В системах распознавания по признакам, где любой объект распознавания представим как вектор $X = \langle x_1, x_2, \ldots, x_n \rangle$, образованный значениями признаков x_j, $j = 1, 2, \ldots, n$, введем функцию $f(x_1, \ldots, x_n)$ от значений признаков. Пусть по значениям этой функции принимается решение об отнесении объекта к одному из *известных* классов (или о том, что объект не может быть отнесен ни к одному из классов). Назовем такую функцию *решающей функцией*.

Существуют различные способы построения решающих функций. Прежде всего решающая функция может строиться как выражение, значение которого отражает расстояние между классифицируемым объектом и классом (прототипом класса). Решающая функция может представлять собой также уравнения границ между классами. В этом случае вектор $X = \langle x_1, x_2, \ldots, x_n \rangle$ представим точкой в n-мерном пространстве признаков. При удачном выборе признаков для распо-

знавания объекты, принадлежащие одному классу, группируются в некоторую область — кластер. Если эти области достаточно четко очерчены (классы четко разделимы), то можно построить гиперповерхности, разделяющие классы. В частном случае, когда, например, объект характеризуется только двумя признаками, моделью будет плоскость, на которой расположены точки, отображающие объекты распознавания. Разделяющие поверхности будут представлять собой кривые на плоскости. Для трехмерного случая (объект задается тремя признаками) границами между классами будут разделяющие поверхности. Задача поиска решающей функции есть задача отыскания уравнений таких поверхностей.


X2
X1
класс А
класс Б


X2
X1
класс А
класс Б
класс Б
класс А

Чтобы распознать объект X относительно классов $1, 2, \ldots, K$ на основе решающих функций, необходимо или сравнивать расстояния между X и прототипами заданных классов, или оценить положение X относительно разделяющих классы гиперповерхностей. Поскольку распознавание возможно только относительно *известных* классов, для решения такой задачи необходимо разбить на классы элементы обучающей выборки $\tilde{X}$, например, с помощью одного из вышеизложенных алгоритмов, затем построить решающие функции для отнесения объекта к одному из классов. Следующий шаг — отнесение объекта X к одному из классов на основе построенных решающих функций (непосредственно задача распознавания).

12.4.1. Построение решающих функций по критерию минимального расстояния

Простейшим способом распознавания некоторого объекта X является сравнение расстояний между данным объектом и объектами-прототипами всех известных классов. Предъявляемый объект X считается принадлежащим классу k, если выполняется условие $|X - Z_k| = \min_i(|X - Z_i|)$, где $i = 1, 2, \ldots, K$ — классы, $Z_i = \langle z_1, z_2, \ldots, z_n \rangle$ — вектор, характеризующий объект-прототип класса i. В качестве объекта-прототипа класса можно использовать его «геометрический центр» (как в алгоритме K средних). Расстояние между предъявленным объектом X и прототипом i-го класса Z_i может быть найдено, например, как обычное евклидово расстояние:

$$|X - Z_i| = \sqrt{\sum_{j=1}^{n} (x_j - z_{ij})^2}.$$

Для упрощения вычислений вместо непосредственного расчета величины $|X - Z_i|$ можно построить другие решающие функции, значения которых связаны определенным образом с расстоянием. Рассмотрим один из методов получения решающей функции.

Найдем квадрат расстояния между X и прототипом Z_i:

$$|X - Z_i|^2 = \sum_{j=1}^{n} (x_j - z_{ij})^2 = (X - Z_i) \cdot (X - Z_i)^T =$$

$$= X \cdot X^T - X \cdot Z_i^T - Z_i \cdot X^T + Z_i \cdot Z_i^T.$$

В данном выражении X — вектор-переменная, Z_i — вектор-константа, X^T — транспонированный вектор. Заметим, что произведение $X \cdot X^T$ не связано с конкретным классом, и, следовательно, не внесет вклад в оценку близости двух объектов. Отбросим его, а оставшуюся часть формулы умножим на (-1). Результатом будет следующая решающая функция:

$$D_i(X) = X \cdot Z_i^T + Z_i \cdot X^T - Z_i \cdot Z_i^T.$$

После умножения на (-1) была получена функция, принимающая максимальное значение при минимальном расстоянии от объекта X до прототипа. Получен критерий распознавания: объект X относим к классу k, если выполняется условие

$$D_k(X) = \max_i D_i(X).$$

Очевидно, потребуется построить K решающих функций, по одной для каждого класса. Заметим, что для K классов можно строить

$\frac{K(K-1)}{2}$ решающих функций в случае, когда для каждой пары классов строится своя решающая функция. К достоинствам метода можно отнести также отсутствие областей неопределенности, то есть областей значений признаков, для которых объект не может быть отнесен ни к одному классу. Однако метод хорош только в случае, если точки, соответствующие объектам класса, образуют выпуклые области.

12.4.2. Разделяющие решающие функции

Рассмотрим метод построения решающих функций как получение уравнений границ, разделяющих классы. Первоначально рассмотрим самый простой случай – случай двух классов. Тогда задача сводится к построению только одной решающей функции. Такая функция должна принимать положительные значения для объектов одного класса и отрицательные – для другого класса. В случае, когда классов несколько, можно использовать различные стратегии.

Метод 1. Сначала какой-либо класс (обозначим его класс 1) считаем первым, а все остальные классы в совокупности — вторым. Найдем решающую функцию, отделяющую класс 1 от остальных. Аналогично построим функцию, отделяющую класс 2 от остальных и т. д. (рис. 12.4). Всего построим K решающих функций. Решение о принадлежности объекта X классу k принимается, если выполняется условие $D_k(X) > 0$, $D_i(X) < 0$ для всех $i \neq k$.

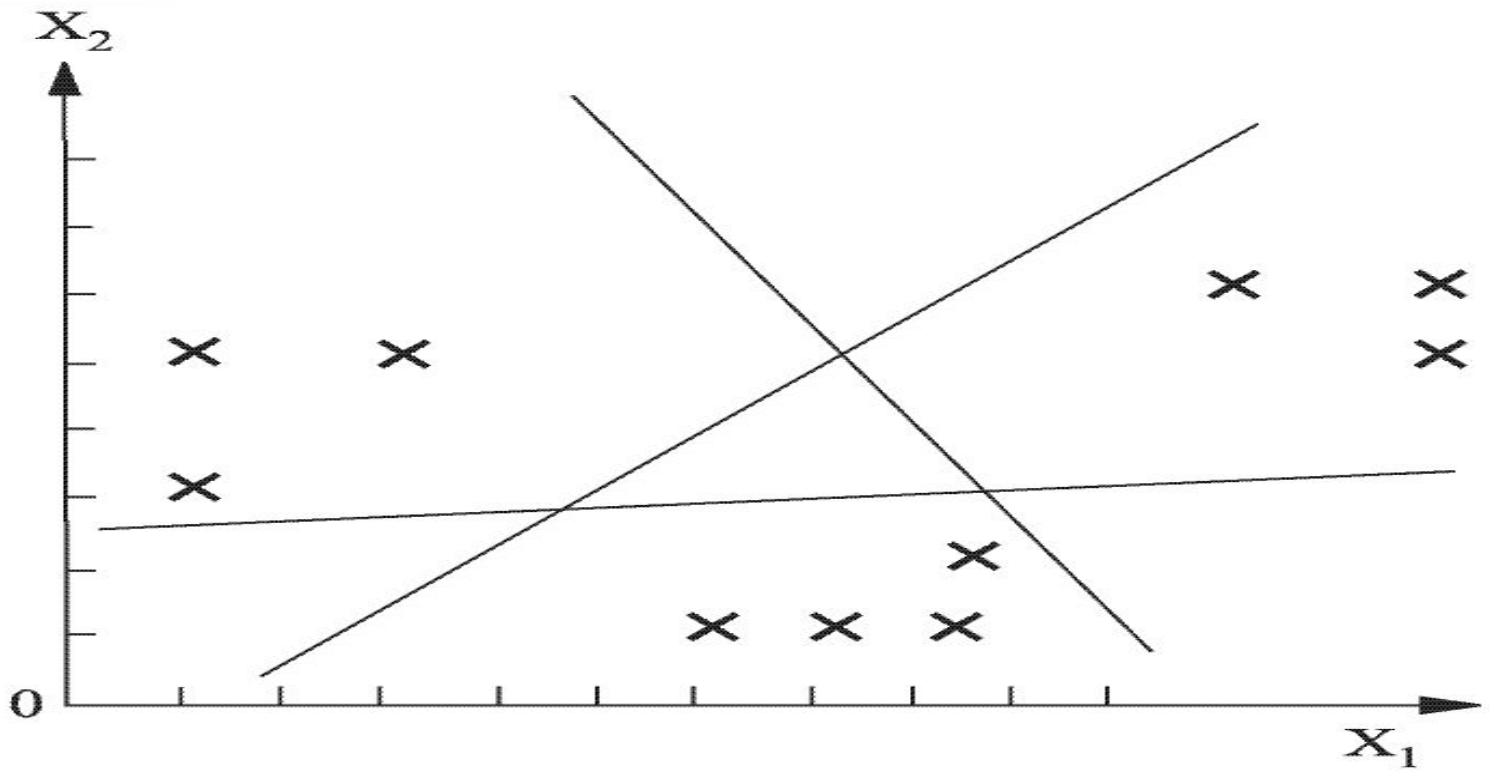


Рис. 12.4.

В таком методе распознавания имеются неопределенные области, в которых объект не может быть отнесен ни к одному классу.

Метод 2. Находится решающая функция для разделения каждой пары классов. Количество решающих функций здесь будет равно количеству пар классов: $\frac{K(K-1)}{2}$.

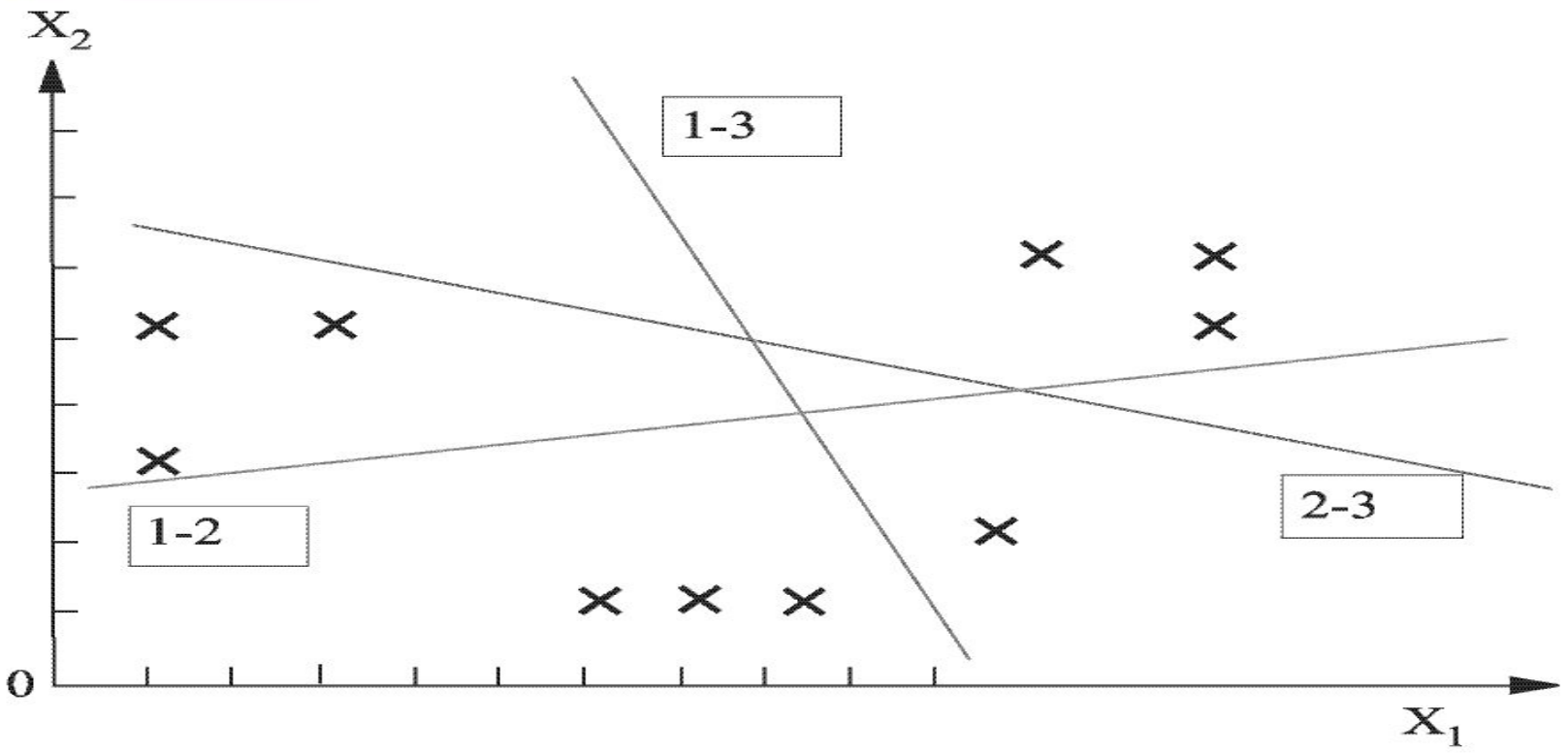


Рис. 12.5.

Решение о принадлежности X классу k принимается, если выполнено условие: $D_{ki}(X) > 0,\ i = 1,2,\ldots,K,\ i \neq k$. При таком подходе возможно наличие единственной области неопределенности (на рис. 12.5 – в центре). Объекты, попадающие в эту зону, не могут быть отнесены ни к одному классу.

Метод 3. Следующий подход основан на устранении областей неопределенности и является комбинацией методов 1 и 2 (рис. 12.6).

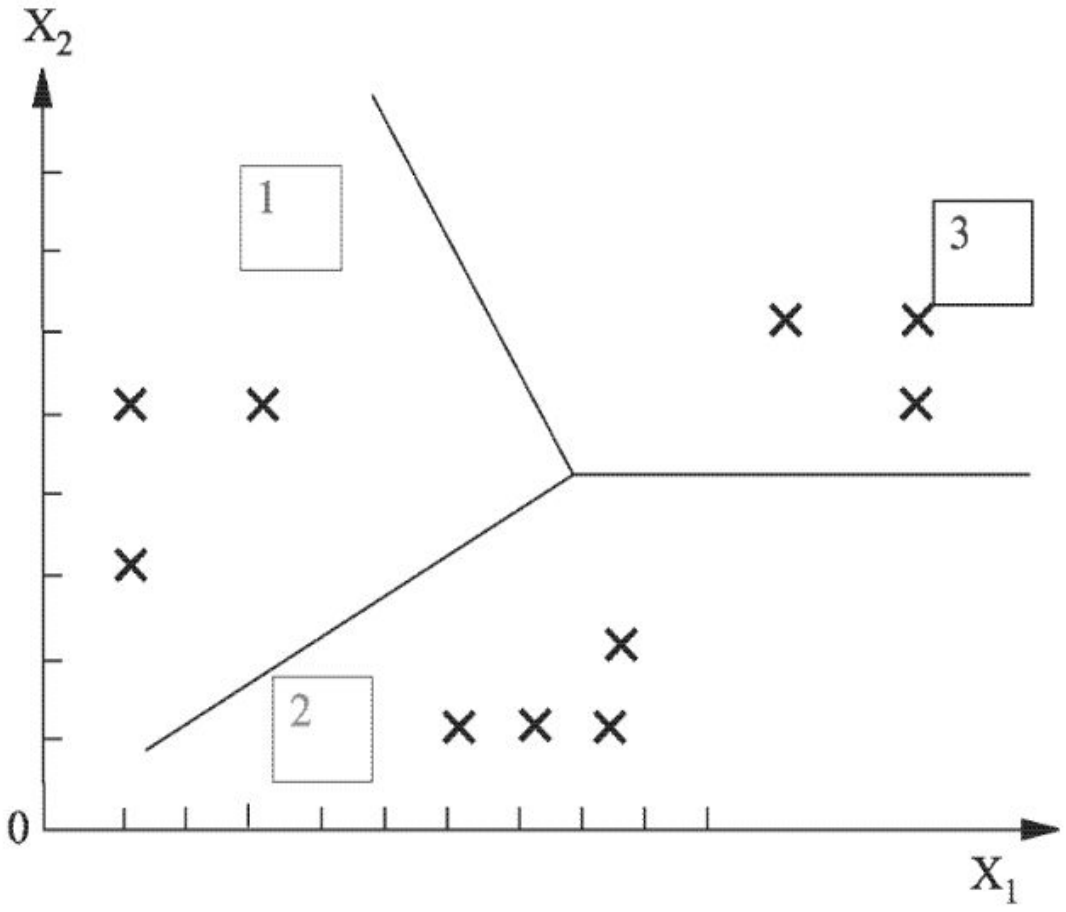


Рис. 12.6.

- Задача минимизации количества решающих функций, достаточных для классификации образов, может быть очень важна, особенно если количество классов *d* велико. Если представить себе, с каким количеством классов объектов (или понятий) имеет дело человек, становится ясно, что решение этой проблемы в том или ином виде потребуется при разработке универсальной системы машинного обучения. Мы, однако, этот вопрос здесь рассматривать не будем, а перейдем к проблеме распознавания образов.

Для простоты будем рассматривать случай двух классов. В задаче распознавания образов в качестве исходных данных выступает обучающая выборка: $(\mathbf{x}_1, A_1), (\mathbf{x}_2, A_2), ..., (\mathbf{x}_M, A_M)$, где $\mathbf{x}_i \in R^N$ и $A_i \in \{a_1, a_2\}$, состоящая из M элементов. На основе этих данных требуется построить решающее правило $\varphi: X \to A$ или решающую функцию $к(\mathbf{x})$.

Однако уже при постановке этой задачи возникает вопрос, какие требования следует предъявлять к решающему правилу? Казалось бы, от этого правила требуется лишь, чтобы оно правильно классифицировало все образы обучающей выборки. Но сразу ясно, что это требование не является достаточным, даже если считать, что классы разделимы.

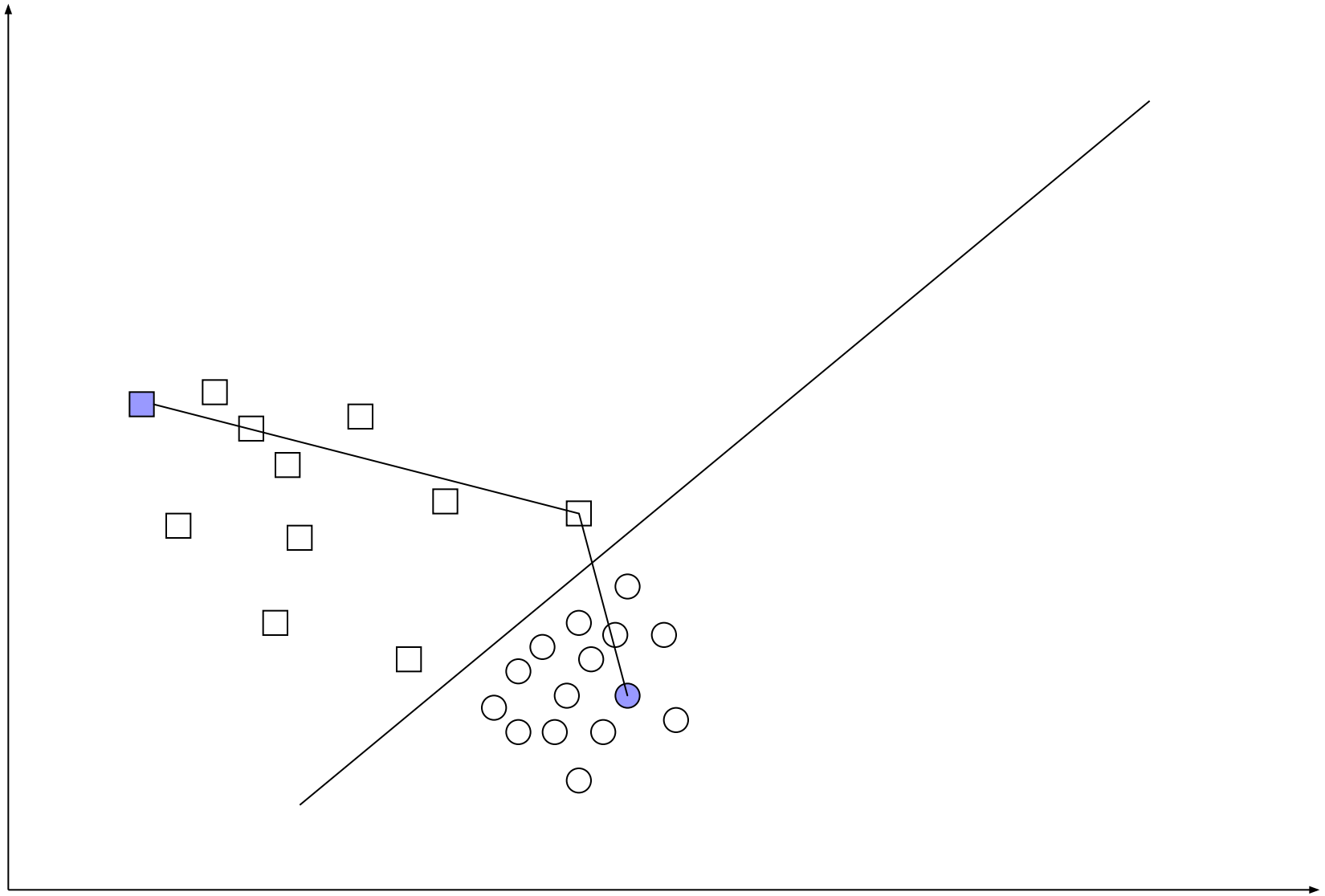
Действительно, можно представить такое решающее правило, которое «помнит» примеры обучающей выборки и возвращает для них правильные номера классов, а для всех других образов возвращает некоторое случайное значение. Как было сказано ранее, такой результат обучения совершенно неудовлетворителен. Стоит подчеркнуть, что такого рода проблемы одинаковы для всех задач обучения, и все они сводятся к заданию адекватного критерия качества результата обучения (в данном случае – критерия качества решающей функции).

Прямого ответа на этот вопрос можно избежать, если конструировать методы распознавания на основе неких эвристических соображений. Два наиболее широко распространенных эвристических метода – это метод эталонных образов и метод ближайшего соседа.

Метод эталонных образов

Метод эталонных образов – это один из эвристических методов построения решающих правил. В основу этого метода положена идея, которая заключается в том, что некоторая совокупность объектов, объединенных в отдельный класс, может быть представлена одним или несколькими эталонными объектами.

- Эти эталонные объекты являются наиболее типичными представителями класса. Типичность эталонного объекта означает, что он в среднем максимально похож на все объекты класса.
- Поскольку сходство двух объектов может трактоваться как величина, противоположная расстоянию между ними в пространстве описаний (образов), то эталон – это объект, для которого минимально среднее расстояние до других объектов.



- Классы, однако, могут обладать разными свойствами. Простейшим свойством является характерный размер класса, который может быть оценен как

$$r_1 = \sqrt{\frac{1}{M_1}\sum_{i=1}^{M_1}\left|\mathbf{x}_{0,1} - \mathbf{x}_{1,i}\right|^2}\,,\; r_2 = \sqrt{\frac{1}{M_2}\sum_{i=1}^{M_2}\left|\mathbf{x}_{0,2} - \mathbf{x}_{2,i}\right|^2}\,. \qquad (16)$$

Тогда для классификации нового образа $\mathbf{x}$ используется следующая решающая функция:

$$\kappa(\mathbf{x}) = \frac{\left|\mathbf{x} - \mathbf{x}_{0,1}\right|}{r_1} - \frac{\left|\mathbf{x} - \mathbf{x}_{0,2}\right|}{r_2}. \qquad (17)$$

Также может использоваться функция

$$\kappa(\mathbf{x}) = \frac{\left|\mathbf{x} - \mathbf{x}_{0,1}\right|^2}{r_1^2} - \frac{\left|\mathbf{x} - \mathbf{x}_{0,2}\right|^2}{r_2^2}, \qquad (18)$$

которая задает ту же разделяющую поверхность.

Если значение этой функции отрицательное, то образ относится к первому классу, в противном случае – ко второму. Разделяющая поверхность для двух классов задается уравнением $\kappa(\mathbf{x}) = 0$.

Метод ближайшего соседа

Другой широко распространенный эвристический метод распознавания – метод ближайшего соседа (или его обобщение – метод *k*-ближайших соседей).

Идея этого метода крайне проста: новый образ относится к тому классу, к которому он ближе. При этом расстояние от образа до класса определяется как расстояние от образа до ближайшего элемента класса.

Тогда на основе обучающей выборки $\mathbf{x}_i, \alpha_i$, $i=1,\ldots,M$, может быть построено следующее решающее правило:

$$\varphi(\mathbf{x}) = \alpha_{\arg\min_i |\mathbf{x}-\mathbf{x}_i|}. \qquad (19)$$

- В соответствии с данным решающим правилом просматривается вся обучающая выборка, в ней находится образ, расположенный наиболее близко к данному и устанавливается, к какому классу он принадлежит (это известно, поскольку он находится в обучающей выборке). Этот класс и приписывается новому образу.

- Метод ближайшего соседа весьма чувствителен к *выбросам*, то есть тем образам обучающей выборки, для которых указаны ошибочные классы. В методе *k*-ближайших соседей выбирается *k* образов обучающей выборки, наиболее близко расположенных к классифицируемому образу, и определяется, к какому классу относится больше всего из них. Поскольку выбросов, как правило, значительно меньше, чем правильных примеров, можно надеяться, что среди *k* ближайших соседей выбросов будет мало, и они не окажут влияния на результат классификации.

Пусть в обучающей выборке первому классу соответствует M_1 элементов $\mathbf{x}_{1,i}$, а второму классу – M_2 элементов $\mathbf{x}_{2,i}$. Тогда эталонные образы для каждого из классов могут быть определены как

$$\mathbf{x}_{0,1} = \frac{1}{M_1}\sum_{i=1}^{M_1}\mathbf{x}_{1,i}\,,\ \mathbf{x}_{0,2} = \frac{1}{M_2}\sum_{i=1}^{M_2}\mathbf{x}_{2,i}\,. \qquad (15)$$

12.4.3. Линейные решающие функции

Рассмотрим алгоритм построения линейных решающих функций. Линейная функция имеет следующий вид: $D(X) = w_0 + \sum_{j=1}^{n} w_j \cdot x_j$. Цель алгоритма — найти коэффициенты w_j решающей функции $D(X)$ методом последовательного уточнения. Рассмотрим случай двух классов (выше было показано, какими приемами можно свести к этому случаю вариант нескольких классов). Основой для вычисления коэффициентов w_j является анализ обучающей выборки $\tilde{X}$, где известна заранее принадлежность объектов $\tilde{X}$ классу 1 или классу 2. Далее эти два класса обозначим C_1 и C_2.

Решающая функция считается построенной, если все объекты обучающей выборки $\tilde{X}$ распознаются этой функцией правильно, то есть $D(X) > 0$, если $X \in C_1$, и, соответственно, $D(X) < 0$, если $X \in C_2$. Коррекция коэффициентов решающей функции выполняется по следующему правилу: коэффициенты решающей функции увеличиваются при неправильном распознавании объекта из класса C_1, уменьшаются при неправильном распознавании объекта из класса C_2 и остаются без изменения, если распознавание идет правильно.

Если на некотором шаге произойдет корректировка коэффициентов решающей функции, счетчик правильно распознанных объектов, обозначаемый далее как *сч*, сбрасывается на ноль, поскольку мы перешли к новой функции, и теперь ее надо проверить заново на всех элементах обучающей выборки.

Алгоритм завершается, когда окажется, что построенная решающая функция $D(X)$ правильно распознает все объекты обучающего множества.

Алгоритм 12.4.

1. Получить обучающую выборку $\tilde{X} = (X_1, X_2, \ldots, X_M)$, элементы которой принадлежат непересекающимся классам C_1 или C_2.
2. Установить в ноль счетчик правильно распознанных объектов: $сч = 0$.
3. Установить номер итерации равным нулю: $k = 0$.
4. Задать начальные значения коэффициентов w_j в решающей функции (например, $w_j = 0$ для $j = 1, 2, \ldots, n$). Получим решающую функцию $D_0(X)$.

5. Выбираем класс C_1 в качестве текущего класса.
6. Переход к новой итерации: $k = k + 1$.
7. Выбрать очередной объект X_k текущего класса (класса C_1). Если класс C_1 исчерпан, объявить текущим классом класс C_2, выбрать первый объект этого класса.
8. Вычислить новые значения коэффициентов решающей функции на итерации k: $w_j^k = w_j^{k-1} + c \cdot x_{kj}$, где c — множитель, определяемый из условия

$$c = \begin{cases} 1, & \sum_{j=0}^{n} w_j^{k-1} \cdot x_{kj} \leqslant 0, \quad \text{и } X_k \in C_1; \\ -1, & \sum_{j=0}^{n} w_j^{k-1} \cdot x_{kj} > 0, \quad \text{и } X_k \in C_2; \\ 0 & \textit{при правильном распознавании.} \end{cases}$$

9. Если $c = 0$, $сч = сч + 1$ (увеличиваем на 1 число правильно распознанных объектов), иначе $сч = 0$.

10. Если $сч = M$ — общему числу объектов обучающей выборки $\tilde{X}$, то КОНЕЦ, иначе перейти к шагу 6.

Приведенный алгоритм обеспечивает построение решающей функции во всех случаях, когда классы являются линейно разделимыми.

Пример 12.4.

Рассмотрим пример работы алгоритма построения линейной разделяющей функции. В табл. 12.10 дана обучающая выборка — объекты, принадлежащие двум классам.

Таблица 12.10

	Точки X	Координаты точек
Класс C_1	1	<1; 2>
	2	<1; 3>
	3	<3; 3>
Класс C_2	4	<4; 1>
	5	<5; 2>
	6	<6; 2>

Каждый объект задается двумя числовыми значениями и может интерпретироваться как точка на плоскости (рис. 12.7). Цель — получить линейную разделяющую функцию, которая дает положительные значения для точек 1, 2 и 3 и принимает отрицательные значения для точек 4, 5, 6. Функция должна иметь вид $F(X) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2$.

Выполняем *итерацию 0*. Коэффициенты $w_0 = w_1 = w_2 = 0$.

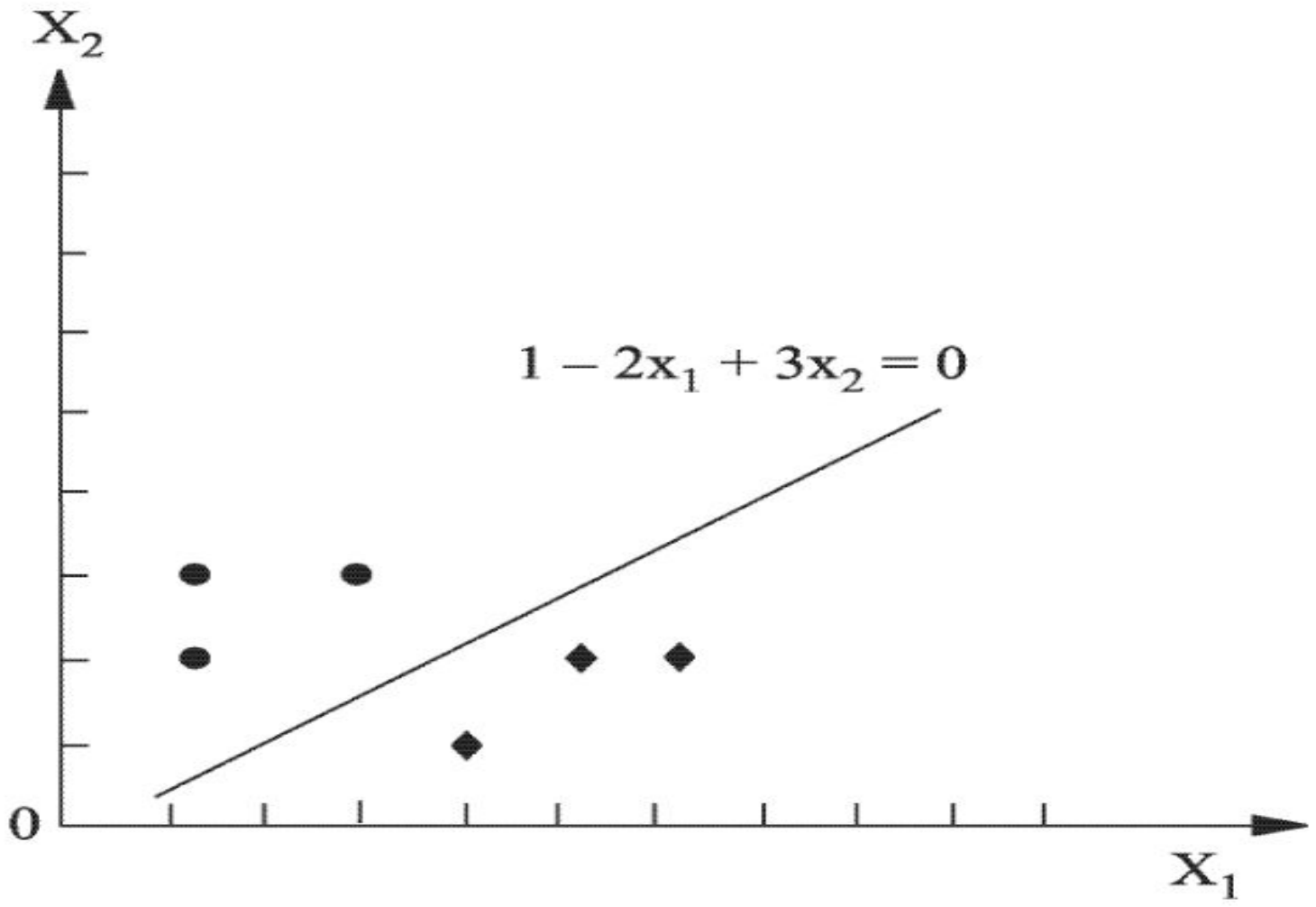


Рис. 12.7.

Выполняем *итерацию 1*. Выбираем первый объект класса C_1 — вектор $X_1 =< 1, 2 >$. Значение функции $F(X_1) = 0 + 0 \cdot 1 + 0 \cdot 2 = 0$, по правилу П8 необходима коррекция коэффициентов при значении множителя $c = 1$. Вычисляем новые коэффициенты функции: $w_0^1 = w_0 + c = 0 + 1 = 1$; $w_1^1 = w_1 + c \cdot x_1 = 0 + 1 \cdot 1 = 1$; $w_2^1 = w_2 + c \cdot x_2 = 0 + 1 \cdot 2 = 2$. Получаем $F^1(X) = 1 + x_1 + 2 \cdot x_2$.

Выполняем *итерацию 2*. Вычислим последовательно значения $F^1(X)$ для элементов выборки: $F^1(<1,2>) = 1 + 1 \cdot 1 + 2 \cdot 2 = 6 > 0$; $F^1(<1,3>) = 1 + 1 \cdot 1 + 2 \cdot 3 = 7 > 0$; $F^1(<3,3>) = 13 > 0$. Все элементы класса C_1 распознаны правильно. Выбираем текущим класс C_2. $F^1(<4,1>) = 1 + 1 \cdot 4 + 2 \cdot 1 = 7 > 0$ — объект распознан неправильно. Необходима коррекция коэффициентов при значении множителя $c = -1$. $w_0^2 = w_0^1 + c = 1 - 1 = 0$; $w_1^2 = w_1^1 + c \cdot x_1 = 1 - 1 \cdot 4 = -3$; $w_2^2 = w_2^1 + c \cdot x_2 = 2 - 1 \cdot 1 = 1$. Новая функция $F^2(X) = x_2 - 3 \cdot x_1$.

Выполняем *итерацию 3*. Вычисляем значения функции для элементов выборки $F^2(< 1,2 >) = 2 - 3 \cdot 1 = -1 < 0$. Необходима коррекция коэффициентов с поправкой $c = 1$: $w_0^3 = w_0^2 + c = 0 + 1 = 1$; $w_1^3 = w_1^2 + c \cdot x_1 = -3 + 1 \cdot 1 = -2$; $w_2^3 = w_2^2 + c \cdot x_2 = 1 + 1 \cdot 2 = 3$. Новая функция $F^3(X) = 1 - 2x_1 + 3x_2$.

Начинаем *новую итерацию* с проверки значений $F^3(X)$ на элементах выборки: $F^3(< 1,2 >) = 5 > 0$; $F^3(< 1,3 >) = 8 > 0$; $F^3(< 3,3 >) = 4 > 0$. Переходим к проверке объектов класса C_2: $F^3(< 4,1 >) = -4 < 0$; $F^3(< 5,2 >) = -3 < 0$; $F^3(< 6,2 >) = -5 < 0$. Все объекты выборки разделены правильно, таким образом получена искомая решающая функция $F(X) = 1 - 2x_1 + 3x_2$. На рис. 12.7 дана геометрическая интерпретация решения.

12.4.4. Построение решающих функций методом потенциалов

Иногда точки обучающей выборки невозможно разделить, используя линейные функции. На рис. 12.8 приведен пример такой обучающей выборки. В случае, если классы невозможно разделить линейно, используется метод потенциалов, предназначенный для построения нелинейных решающих функций.

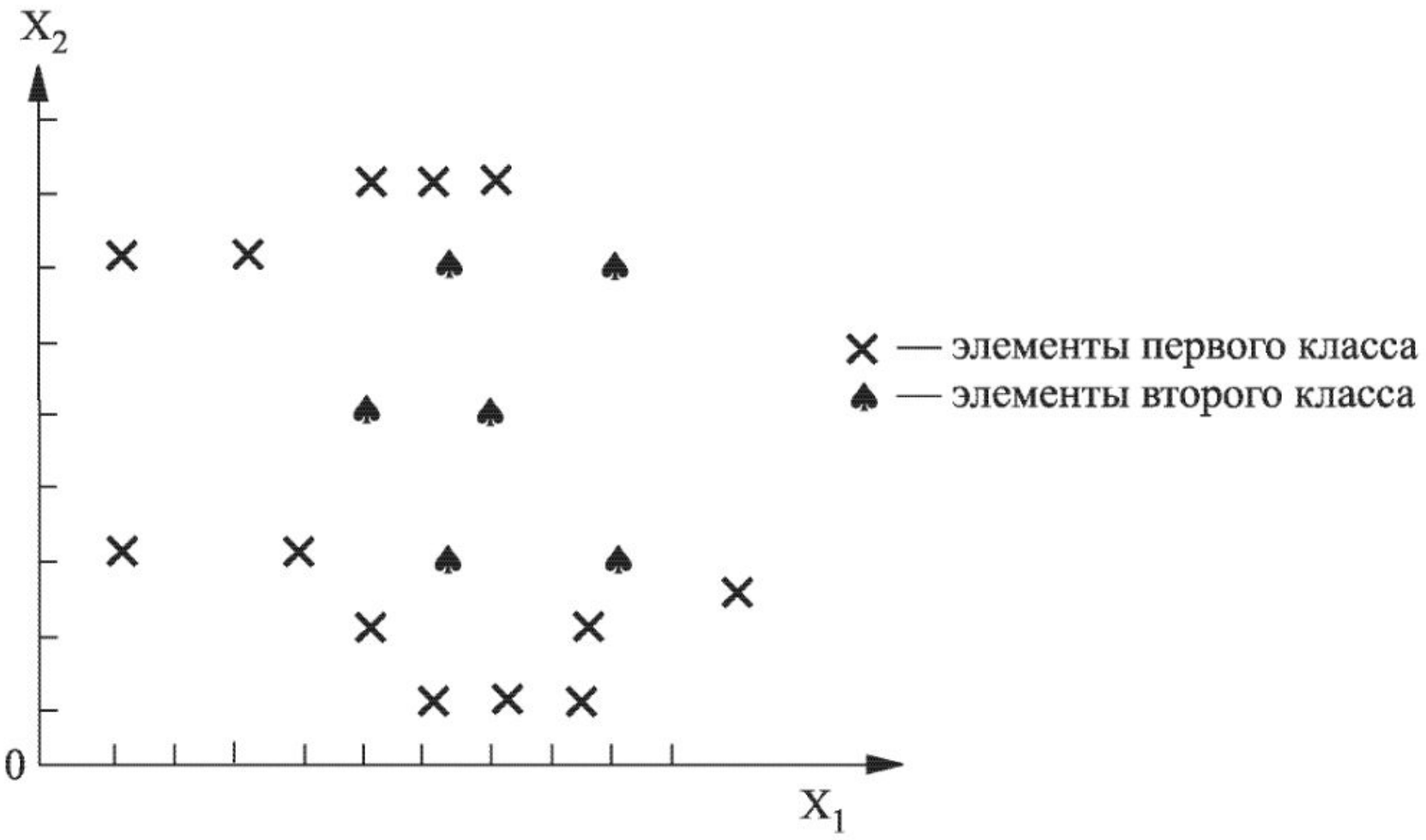


Рис. 12.8.

Само название «метод потенциалов» связано со следующей интерпретацией. Предположим, что любой точке из обучающего множества соответствует некоторый объект, обладающий зарядом (наподобие электрического). Объекты разных классов имеют заряды разных знаков: положительные и отрицательные. В любой точке n-мерного пространства признаков такие заряды создадут некоторый потенциал, являющийся суммой потенциалов отдельных зарядов. Величина потенциала прямо пропорциональна величине заряда и обратно пропорциональна расстоянию до него.

Известно, что линии, соединяющие точки с одинаковыми потенциалами, называются эквипотенциалями. В таких условиях каждый класс отделен от другого «потенциальной долиной», имеющей нулевой потенциал. Такая нулевая эквипотенциаль представляет собой границу между классами; ее уравнение и будет решающей функцией.

Функция, описывающая закон изменения потенциала, создаваемого зарядом, помещенным в точку X_c, может быть задана по-разному. Рассмотрим два варианта:

1) $F(X, X_c) = \exp(-a \cdot |X - X_c|^2)$;

2) $F(X, X_c) = \dfrac{1}{1 + a \cdot |X - X_c|^2}$, где $a > 0$ — константа (можно принять $a = 1$).

Если рассматривать расстояние по Евклиду, вторая функция примет вид

$$F(X, X_c) = \frac{1}{1 + \sum_{j=1}^{n} (x_j - x_{cj})^2}, \text{ где коэффициент } a = 1.$$

Рассмотрим алгоритм построения решающей функции как разделяющей границы $D(X)$, отделяющей области положительного и отрицательного потенциалов. При описании алгоритма используем обозначения, принятые в описании алгоритма построения линейной разделяющей функции. Легко заметить, что оба алгоритма имеют много общего.