Лекция 3

План лекции:

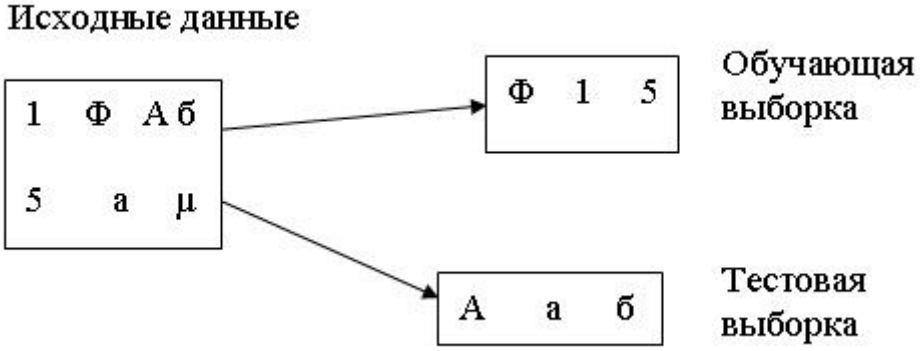
1. Обучение в решении задачи классификации.
2. Методы контролируемого и неконтролируемого обучения.
3. Различные способы нахождения расстояния между классами.
4. Нахождение расстояния между списками.
5. Метод динамического программирования.

3.1 Обучение в решении задачи классификации

Одна из главных задач распознавания – классификация. Критерии такого разделения далеко не всегда могут быть точно и непосредственно формализованы.

Рассмотрим пространство признаков, которые выбраны адекватно поставленной задаче. Первый этап решения состоит в том, чтобы отобразить в этом пространстве “облако” точек (или одну точку), связанных с одним и тем же классом, а затем определить один или несколько прототипов, представляющих будущие классы. Эти представители классов не обязательно должны совпадать с какими-то конкретными реализациями, по которым получены результаты экспериментальных измерений или наблюдений. Скорее наоборот, имея некоторые начальные сведения о прототипах, можно управлять процессом измерений или другим методом получения исходных данных. Обычно прототипы называют именами представляемых ими классов. На последующих этапах они используются для распознавания неизвестного образа, исходные данные о котором получены так же, как и данные об уже известных образах.

Основу такого способа классификации составляет процесс обучения, в задачу которого входит постепенное усовершенствование алгоритма разделения предъявляемых объектов на классы. Этот процесс стремятся автоматизировать, для чего отбирают часть предъявляемых объектов и используют их в процессе обучения для “тренировки” системы. Массив исходных данных в обучаемой системе состоит из двух частей: обучающей выборки и тестовой выборки, используемой в процессе испытаний.



Если совокупность классов известна заранее, то обучение называют контролируемым (обучение “с учителем”).

Роль разработчика заключается в выработке наилучших критериев классификации, учитывающих различия между признаками, характерными для отдельных классов. Главной задачей становится в этом случае поиск оптимальных методов разделения. Если классы, составляющие обучающую выборку, не известны заранее до начала процедуры классификации, то обучение называют неконтролируемым или “без учителя”.

3.2 Методы контролируемого и неконтролируемого обучения

Главная особенность контролируемого метода классификации заключается в непременном наличии “справочных” сведений о принадлежности к определенному классу каждого вектора измерений, входящего в обучающую выборку. Роль обучающего состоит в том, чтобы создать такую систему, которая позволила бы каждый вектор измерений, источник которого неизвестен, отнести к одному из уже известных классов. Задача заключается в уточнении и оптимизации процедуры принятия решений. В основу процедуры положено понятие расстояния от рассматриваемой точки до границы, отделяющей пространство, характеризуемое определенными признаками.

В обучении без учителя автоматическое устройство самостоятельно устанавливает классы, на которые делится исходное множество, и одновременно определяет присущие им признаки. Для разделения данных были определены критерии. При таком разделении неизвестны ни классы, ни их количество. Поэтому процесс организуется так, чтобы среди всех возможных вариантов найти такой, при котором группы обладают наибольшей компактностью.

Если полученная на каком-то этапе классификация неудовлетворительная, то следует вернуться к одному из предыдущих этапов с учетом имеющейся, но не использованной ранее информации. Каждое возвращение на очередной виток цикла характеризуется определенной ценой, в качестве которой можно принять расстояние, отделяющее конечный этап от

того, с которого начинается новый цикл. Эта процедура не всегда сходится. На рис. 1 приведена общая структурная схема алгоритма обучения.

Рассмотрим примеры алгоритмов контролируемого и неконтролируемого обучения.

Алгоритм *К средних* является представителем контролируемого обучения. Исходные данные: выборка объектов и *к* – количество классов. Обычно в качестве *к* ядер используют *к* первых элементов из списка данных.

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

Шаг 1. Фиксируются *к* ядер (центров областей), затем вокруг них формируются области по правилу минимального расстояния. На *r*-ом этапе вектор X_p связывается с ядром $N_i(r)$, если выполняется следующее неравенство:

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

Шаг 2. На *r+1* этапе определяются новые элементы, характеризующие новые ядра $N_i(r+1)$. За их значения принимают векторы *X*, обеспечивающие минимум среднеквадратичного отклонения от всех остальных векторов, принадлежащих данной области. Это достигается для одного *X* в каждой области.

Шаг 3. Если хотя бы в одном из классов изменилось положение ядра, то пересчитываются области принадлежащих им векторов, т.е. определяются расстояния от всех объектов до новых ядер, в результате чего может произойти перераспределение областей. Затем повторяется шаг 2. Процедура заканчивается, если на *i*-ом шаге ее выполнения положения центров областей не меняются по сравнению с *i-1* шагом.

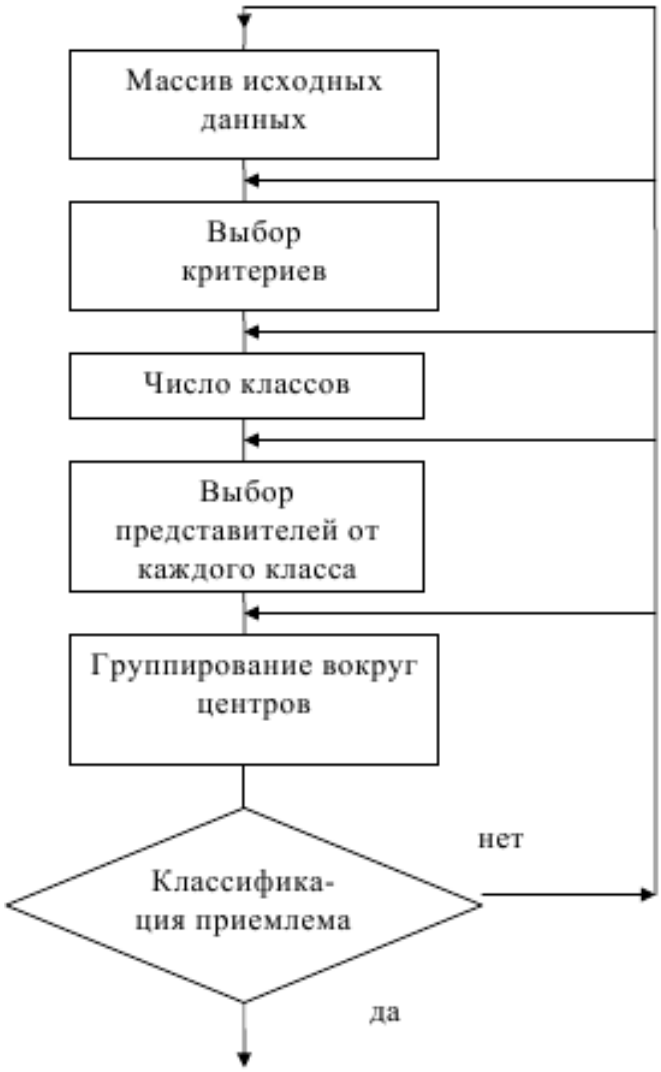


Рис. 1 – Общая схема процесса обучения

Рассмотрим алгоритм классификации *максимина*, который является представителем неконтролируемого обучения.

Цель алгоритма – отыскать представительные элементы каждого класс исходя из произвольного выбора. По определению каждый объект представляется вектором *X(i)*, составляющие которого $\{x_k(i)\}$.

Шаг 1. Произвольно выбирается первый элемент $N_1=X(1)$ из множества векторов $X=\{X(1), X(2), X(3)\ldots X(V)\}$, после чего определяются другие ядра N_2, $N_3\ldots N_m$, число m которых заранее неизвестно.

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

Шаг 3. Вычисляются расстояния между остальными точками и имеющимися ядрами: $d_{ki} = d(\overline{N}_k, \overline{X}(i)), k = 1,2; i = 1,2,\ldots v-2,$ после чего

каждая точка связывается с тем ядром, расстояние до которого от нее минимально.

Шаг 4. В каждом классе определяется точка, максимально удаленная от ядра. Среди найденных расстояний выбирается абсолютный максимум. Это – значение δ_{kp}. Если $\delta_{kp} > \frac{1}{2} d_{12}$, то соответствующая ему точка назначается очередным ядром ($\overline{N}_3$).

Шаг 5. Все точки освобождаются от своих ядер, и алгоритм повторяется, начиная с третьего шага. Если появляется очередной кандидат в ядра нового класса, то в условии (шаг 4) сравнение выполняется с половиной средней величины расстояний между всеми уже существующими ядрами. Алгоритм заканчивается, когда для очередного кандидата в ядра перестает выполняться данное условие, а, следовательно, прекращают появляться новые классы. К этому моменту выявляется число классов l и их ядра $N_1, N_2 \ldots N_l$.

3.3 Расстояние между классами

Введем понятие расстояние между классами, исходя из гипотезы о том, что оно должно было бы обладать двумя основными свойствами.

1. Компактность, выражающееся в том, что точки, представляющие объекты одного класса, расположены друг к другу ближе, чем к точкам, относящимся к другим классам.

2. Сепарабельность, отражающее тот факт, что классы ограничены и не пересекаются между собой.

На практике оба эти свойства выполняются не всегда, так как зависят от того, насколько удачно выбраны признаки. Понятие расстояния позволяет оценить степень сходства как между отдельными реализациями, так и между целыми классами.

С точки зрения распознавания образов можно полагать, что чем меньше расстояние, тем сходство должно быть больше. Например, оценить сходство величиной $к$, обратной расстоянию d: $k=1/d$.

Нередко признаки объектов, между которыми нужно установить сходство, не могут быть выражены в числах. Наличие или отсутствие признака можно кодировать двоичным кодом.

Пусть некоторый образ X_i представляется в виде вектора: $\overline{X}_i = \{x_{i1}, x_{i2}, \ldots x_{ik} \ldots x_{in}\}$, где первый индекс – это номер реализации, а второй индекс – номер признака. Образ можно записать в виде последовательности двоичных символов в соответствии с правилом: если i-ый объект обладает $к$-ым признаком, то $x_{ik}=1$, иначе $x_{ik}=0$.

Пример: Исходные данные для классификации представлены в следующей таблице

Объект	Вектор	Желтый	Красный	Есть семечки	Есть косточки
Вишня	X_1	$x_{11}=0$	$x_{12}=1$	$x_{13}=0$	$x_{14}=1$
Яблоко	X_2	$x_{21}=1$	$x_{22}=1$	$x_{23}=1$	$x_{24}=0$
Банан	X_3	$x_{31}=1$	$x_{32}=0$	$x_{33}=0$	$x_{34}=0$

Эту таблицу можно представить как класс C, а множество векторов как $X=\{x_1, x_2, x_3\}$. В данном случае класс фруктов состоит из объединения непересекающихся подмножеств, каждое из которых включает в себя единственный объект. Поэтому такая классификация является тривиальной. Можно провести более тонкую классификацию, если для каждой пары предъявлений последовательно установить степень их сходства и различия. Тогда таблицу соответствия двух реализаций X_i и X_j можно представить следующим образом:

X_j	X_i	
	1	0
1	a	h
0	g	b

a – число случаев, когда X_i и X_j обладают одним и тем же признаком.

$$a = \sum_{k=1}^{n} x_{ik} x_{jk}$$

b – число случаев, когда X_i и X_j не обладают никакими общими признаками.

$$b = \sum_{k=1}^{n} (1 - x_{ik})(1 - x_{jk})$$

h – число случаев, когда X_i не обладает признаками, присущими X_j.

$$h = \sum_{k=1}^{n} (1 - x_{ik}) x_{jk}$$

g – число случаев, когда X_i обладает признаком, отсутствующим у X_j.

$$g = \sum_{k=1}^{n} x_{ik} (1 - x_{jk}).$$

Отсюда следует, что чем больше сходств между X_i и X_j, тем больше должен быть коэффициент a и тем сильнее он должен отличаться от других коэффициентов. Можно ввести функцию сходства, обладающую следующими свойствами:

1. Возрастающей в зависимости от a.
2. Симметричной относительно g и h.
3. Убывающей в зависимости от b.

Эту функцию, для которой известно несколько различных вариантов вычисления, называют двоичным расстоянием. Для ее вычисления есть

множество формул (приведенных ниже), их многообразие свидетельствует о том, что еще нет оптимальной формулы, подходящей для решения всех задач.

$$S_1(X_i, X_j) = \frac{a}{n}, \quad S_2(X_i, X_j) = \frac{a}{n-b}, \quad S_3(X_i, X_j) = \frac{a}{2a+h+g},$$

$$S_4(X_i, X_j) = \frac{a}{a+2(g+h)}, \quad S_5(X_i, X_j) = \frac{a+b}{n},$$

$$S_6(X_i, X_j) = \frac{a}{g+h}, \quad S_7(X_i, X_j) = \frac{1}{2}\left[\frac{a}{a+g} + \frac{a}{a+h}\right], \quad S_8(X_i, X_j) = \frac{a}{\sqrt{(a+h)(a+b)}},$$

$$S_9(X_i, X_j) = \frac{ab-gh}{ab+gh}.$$

n – число признаков.

Особое внимание стоит обращать на эффект размера объектов: два объекта большого размера (и как следствие обладающие многочисленными признаками) представляются более похожими, чем два меньших объекта. Поэтому следует вносить поправку в параметр а. В случае двух идентичных объектов имеем: b=n-a; g=h=0. Функция $S_6 \to \infty;\ S_2, S_4, S_5, S_8, S_9 = 1; S_3 = 0.5.$

Введя функцию сходства, вернемся к примеру с фруктами, где $X_1=(0,1,0,1);\ X_2=(1,1,1,0);\ X_3=(1,0,0,0)$. Найдем сходства между X_1 и X_2 и между X_2 и X_3. Для первой пары составим таблицу совпадений признаков и воспользуемся функцией S_6.

X_2	X_1	
	1	0
1	1	2
0	1	0

$$S_6 = \frac{a}{g+h} = \frac{1}{1+2} = 0{,}333$$

Таблица сходств для векторов X_2 и X_3

X_3	X_2	
	1	0
1	1	0
0	2	1

$$S_6 = \frac{a}{g+h} = \frac{1}{0+2} = 0{,}5$$

В соответствии с выбранными критериями отбора получаем, что объекты X_2 и X_3 больше схоже, чем объекты X_1 и X_2, т.е. яблоко и банан более схоже чем вишня и яблоко.

3.4 Расстояние между списками

В некоторых задачах сходство между объектами базируется на мере сравнения порядка списков.

Пусть имеется две реализации:

$\overline{X}_i = (x_{i1}, x_{i2} ... x_{in}); \overline{X}_j = (x_{j1}, x_{j2} ... x_{jn}).$

Коэффициенты сравнения определяются следующим образом:

$$\Delta_{lk}^{i} = \begin{cases} 1, при\, x_{il} > x_{ik} \\ -1, при\, x_{il} < x_{ik} \\ 0, при\, x_{il} = x_{ik} \end{cases} \quad l<k.$$

Расстояние в этом случае вычисляется так: $d(\overline{X}_i, \overline{X}_j) = 1 - \frac{2}{n(n-1)} \sum_{l<k} \Delta_{lk}^{j} \Delta_{lk}^{i}.$

Если компоненты обоих векторов упорядочены однотипно, то $\Delta_{lk}^{i} = \Delta_{lk}^{j} \; \forall l, k.$ Расстояние между объектами в этом случае равно 0.

Пример: $X_1=\{0.5, 1, 2\}$, $X_2=\{1, 3, 8\}$.

$\Delta_{1,2}^{1} = \Delta_{1,3}^{1} = \Delta_{2,3}^{1} = -1; \Delta_{1,2}^{2} = \Delta_{1,3}^{2} = \Delta_{2,3}^{2} = -1.$

Тогда расстояние между объектами будет

$$d(\overline{X}_1, \overline{X}_2) = 1 - \frac{2}{3(3-1)}[1 \times 1 + 1 \times 1 + 1 \times 1] = 0.$$

Максимальное расстояние между списками достигается, когда они упорядочены противоположно. Другой подход к определению расстояния между двумя списками состоит в учете последовательности составляющих в каждом списке. В этом случае учитывается структура последовательности. Каждый список состоит из символов, входящих в один алфавит.

Пусть заданы два списка: $\overline{X}_i = (x_{i1}, x_{i2} ... x_{in}); \overline{X}_j = (x_{j1}, x_{j2} ... x_{jm})$, m и n могут быть не равны. Задача состоит в нахождении функции, которая выражала бы степень сходства двух списков.

Пример: От ряда ABC перейти к ряду ABDE. Алгоритм перехода включает в себя следующие шаги:

1)сохранить первый и второй символы.

2)Заменить третий символ.

3)Добавить четвертый символ.

Введя пустой символ λ, можно эти три варианта преобразований записать в виде трех операций.

SUB (подстановка) $x_i \to x_j \;\; SUB(x_i, x_j);$

DES (уничтожение) $x \to \lambda \;\; DES(x, \lambda);$

CRE (создание) $\lambda \to x\ CRE(\lambda, x)$.

Каждому преобразованию соответствует своя цена C. Для оценки расстояния между двумя списками вводят понятие полной цены последовательности преобразований как наименьшей из всех возможных цен, которые следует “заплатить” за переход от исходного списка к конечному. Процесс перехода от одного списка к другому может включать в себя три процедуры.

Пусть есть списки $X_i(l)$ и $X_j(k)$. Необходимо выполнить преобразование $X_i(l) \to X_j(k) = D(l,k)$. Для перехода от *D(l-1, k-1) к D(l,k) l,k* – длины списков имеются три возможности.

1. Путем подстановки $X_i(l)$ и $X_j(k)$, цена которой соответствует цене $SUB(x_{il}, x_{jk})$ последних элементов в каждом списке.
2. Путем создания последнего элемента в списке j, за что приходится платить цену $CRE(\lambda, x_{jk})$.
3. Если рассмотреть пару $X_i(l-1)$ и $X_j(k)$, то цена будет $DES(x_{il}, \lambda)$, что соответствует уничтожению последнего элемента в списке i.

Пример: Пусть имеется два списка $\overline{X}_i = (x_{i1}, x_{i2} ... x_{i5}); \overline{X}_j = (x_{j1}, x_{j2}, x_{j3})$ и установлены цены на каждую операцию

$$c(\lambda, x_{jk}) = 1 \forall k;$$
$$c(x_{il}, \lambda) = 0.5 \forall l;$$
$$c(x_{il}, x_{jk}) = 0 \forall l,k \; при \; x_{il} = x_{jk};$$
$$c(x_{il}, x_{jk}) = 1 \forall l,k \; при \; x_{il} \neq x_{jk}.$$

$X_i = \{a,a,b,a,c\}$; $X_j = \{a,b,d\}$. Тогда, один из возможных путей преобразования следующий

1. $a \to a = 0.$
2. $a \to \lambda = 0.5$
3. $b \to b = 0.$
4. $a \to d = 1.$
5. $c \to \lambda = 0.5$

Суммарная цена преобразований равна 2.

3.5 Метод динамического программирования

Отличие данного метода от метода нахождения расстояния между списками заключается в том, что в этой процедуре одному элементу первого списка могут соответствовать несколько элементов второго списка. Пустые места отсутствуют. Задача состоит в создании матрицы, каждый элемент которой представляет оптимальное расстояние между соответствующими элементами двух списков. Поскольку, если каждая составляющая пути оптимальна, то и весь путь будет оптимальным. Рассмотрим два списка *X* и *Y*, состоящие из элементов: $X(l)=a_1,a_{2...}a_l;Y(k)=b_1,b_2...b_k$.

Список *X* называется опорным, а *Y* – подлежащим сравнению (кандидатом). Количество значений списка *X=n, Y=m.* В общем случае m не равно n. Например, такая ситуация встречается при записи одного и того же слова, произнесенного с различной скоростью. Аналогичная ситуация возникает при анализе фотоснимков или их частей, выполненных при разных увеличениях. Информационное содержание в каждом из случаев остается неизменным. Таким образом, первая задача состоит в том, чтобы установить оптимальное соответствие между двумя списками. Рассмотрим схему этого процесса на рис. 2.

Согласование опорного и предъявляемого сигналов выполняется с помощью функции g, связывающей *l* и k, *l*=g(k). Примем g(*1*)=1; g(m)=n. В качестве первого приближения можно принять линейную зависимость $l=g(k)=(k-1)\frac{n-1}{m-1}+1.$ Более качественный способ согласования можно представить так $D^*=\min_{g(k)}\left[\sum_{k=1}^{m} d(b(k),a[g(k)])\right]$, где *b(k)* – к-ая составляющая вектора Y, которой соответствует g(k)-я составляющая вектора X.

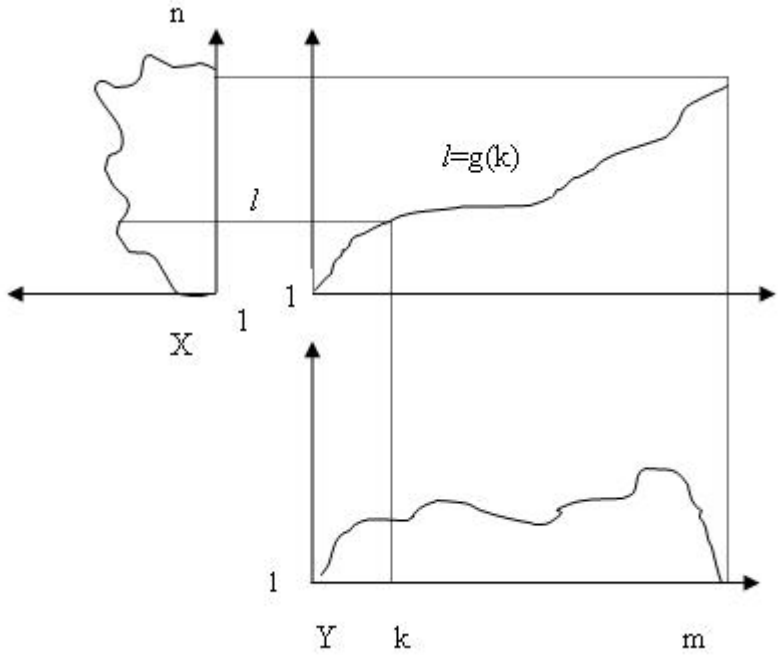


Рис. 2 – Схема метода динамического программирования

Расстояние между этими составляющими вычисляется по одной из формул для расстояния. Функция g(k) должна обеспечивать возможность минимизации в целом расстояния D_c между двумя списками. После того как оба списка исчерпаны, получают величину D^*, позволяющую оценить нормализованное расстояние между списками: $\delta(X,Y) = \frac{D^*}{n+m} = \frac{D_C(n,m)}{n+m}$. Ранее делается предположение, что D^* дает оптимальный результат. Выражение для D_c

означает, что $D_C(l,n) = d(a_l,b_k) + \min_{q \le k}[D_C(l-1),q)]$, т.е. $D_c(l,k)$ – текущее расстояние, накапливаемое от точки $l=k=1$ до

точки с координатами *(k, l)* (рис. 2).

Алгоритм нахождения расстояния:

Сначала l=1, k=1. $D(1,1) = 2d(1,1) = d(1,1) + \min(D_C(1,1))$. Затем $l=l+1$ и находится min расстояние от l до к, т.е. просматриваются все возможные значения к на отрезке от 1 до к. Можно одновременно исследовать не всю плоскость, а некоторую ее часть (рис. 3), положив условие $k - r \le l \le k + r$, где r выбирается заранее. Тогда сужается пространство значений κ, которые нужно проверять для каждого конкретного l.

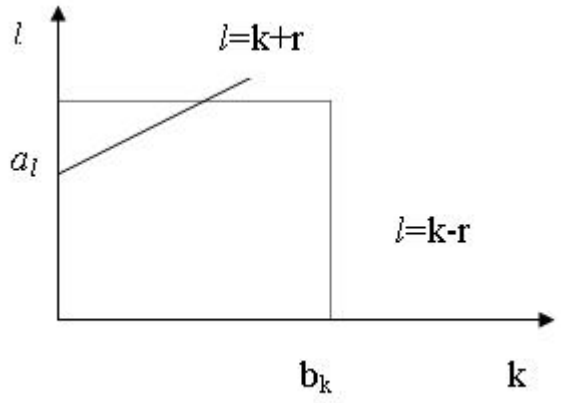


Рис. 3 – Полоса области, выделенная для исследования