**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОТЕХНОЛОГІЙ І ПРИРОДОКОРИСТУВАННЯ**

**Факультет інформаційних технологій**

Кафедра комп’ютерних наук

**Терія розпізнавання образів та класифікація в системах штучного інтелекту**

**Лабораторна робота №3**

**«Метод розподіляючих функцій в розпізнаванні образів»**

**(4 години)**

Київ - 2025

**Мета роботи:** Засвоєння базових знань щодо способів класифікації оброазів на основі методу розподіляючих функцій. Розробка програмної системи, що реалізує розпізнавання образів на основі зазначеного методу. Дослідження розробленої системи. Отримання практичних навичок з розробки програмних ситем розпізнавання образів..

**Підготовка до роботи:** Вивчити і уяснити теретичні відомості щодо способів класифікації оброазів на основі методу розподиляючих функцій.

**Завдання:**

1. Розробити програмний додаток на мові програмування C/C++ в середовищі розробки MS Visual Studio, що реалізує розпізнавання образів на основі зазначеного методу.
2. Дослідити властивості методу розподіляючих функцій за допомогою розробленого програмного додатку на прикладах рішення задач класифікації. В якості похідних даних при дослідженні властивостей методу використати приклад, що наведений в матеріалах методичної розробки до лабораторної роботи і довільний власний приклад, якій було використано при дослідженні в лабораторних роботах №1 і №2.
3. За результатами досліджень скласти звіт з описом отриманих результатів та обґрунтованими висновками.
4. За результатами досліджень скласти звіт з поясненням проміжних результатів процесу вирішення задач класифікації та сформулювати обґрунтовані висновки щодо отриманих результатів дослідження.
5. Порівняти і обґрунтувати результати проведеного дослідження з результатами дослідження, що були отримані при виконанні дослідження в лабораторних роботах №1 і №2.

**Зміст звіту:**

1. Назва та мета роботи.
2. Теоретичні відомості.
3. Відкоментований похідний код програмного додатку для дослідження методу розподіляючих функцій на мові C/C++ в середовищі розробки MS Visual Studio.
4. Скрин-шоти роботи програмного додатку з поясненнями щодо отриманих проміжних результатів вирішення задач класифікації і обґрунтуванням отриманих результатів дослідження.

5. Порівняти і обґрунтувати результати проведеного дослідження з результатами дослідження, що були отримані в лабораторній роботі №1.Похідний програмний код з коментарями.
6. Сформулювати висновки щодо отриманих результатів дослідження методу потенційних функцій при вирішенні задач класифікації.

## Теоретичні відомості

В основу математичного підходу покладені правила класифікації, які формулюються і виводяться в рамках визначеного математичного формалізму з допомогою принципів загальності властивостей і кластеризації. Коли образи деякого класу представляють собою вектори, компонентами яких є дійсні числа, то цей клас можна розглядати як **кластер**. Побудова системи розпізнавання, яка базується на реалізації даного принципу, визначається просторовим розміщенням окремих кластерів. Якщо кластери, що відповідають різним класам, рознесені далеко один від одного, то можна користуватися простими схемами розпізнавання, наприклад, класифікацією за принципом мінімальної відстані.

Інший підхід відомий як **метод Розподільчих функцій**, відрізняється тим, що об'єднуються деякі розмиті множини, що описуються так званою потенціальною функцією. При цьому здійснюється апроксимація не розв'язуючою, а дискримінантною функцією. Її значення вираховується шляхом складання значень «потенціалу», який зменшується в міру зникання від деяких центрів, що вибираються в процесі навчання. Цей підхід є еквівалентним апроксимації дискримінантної функції за допомогою функціонального ряду. Суть полягає в тому, що на просторі вхідних векторів X задається функція, яка називається «потенціалом». Потенціал визначається наближенням двох точок і задається як функція відстані між точками. Потенціальна функція така, що вона монотонно зменшується із збільшенням відстані.

Таким чином, результатом побудови є потенціальне поле, яке розбиває весь простір на дві частини: де значення додатні і від'ємні. Поверхня, на якій потенціал дорівнює нулю, називається розділяючою.

Інтелектуальну систему людина звичайно розглядає як деяку функцію $Q(R) = Q(x_1, x_2, \ldots, x_j, \ldots, x_n)$. Тому при обмеженому значенні радіуса R всі процеси в n-вимірному просторі ***відбуваються в гіперкулі***.

***Аналітична геометрія в n-вимірних тілесних кутах*** може застосовуватись у наступних напрямках аналізу складної n-вимірної числової залежності $Q(R) = Q(x_1, x_2, \ldots, x_j, \ldots, x_n)$:

– для отримання на двовимірній площі *геометричних образів* n-вимірного простору у вигляді: *ізоліній*; *годографу* вектора $R = (x_1, x_2, \ldots, x_j, \ldots, x_n)$ з визначеними геометричними ознаками (мінімального, максимального, поточного значення); *геометричних об'єктів* (точки, лінії, площини, геометричної фігури, визначених експертом гранульованих ділянок); *двовимірної мапи* тілесних кутів з метою дослідження зміни стану підприємства по даним дискримінантного аналізу; для аналізу складних систем керування

– для перетворення «чорного ящика» у *«прозорий скляний ящик»* за допомогою визначеної експертом мапи Т-кутів на двовимірній площині та для спостереження за наближенням координат об'єкта «чорного ящика» до визначеної експертом небезпечної границі тощо;

– *як теоретична основа кластерного аналізу*, який дозволяє групувати та класифікувати стохастичні або детерміновані об'єкти з числовими ознаками $X(x_1, x_2, \ldots, x_j, \ldots, x_n)$;

– *як теоретична основа аналізу нейронних мереж*, що дозволяє отримати «самоорганізувальну мапу» нейронної мережі, яка має переваги перед аналогічною «мапою Кохонена» [35; 44; 103; 168] із-за того, що навчання вагових коефіцієнтів нейронів замінене реєстрацією навчальних даних в тілесних кластерах з відомими адресами з отриманням реальної кількості об'єктних кластерів при відсутності псування навчальної інформації додатковими латеральними зв'язками та «впливом сусідів»;

– *як теоретична основа багатоагентних систем*, в яких кожний агент у вигляді Т-комірки виконує свою частку задачі по розпізнаванню образів і може мати власну вихідну функцію, підпорядковану загальній меті;

– *для систем гранульованих координат*, теорія яких запропонована Л. Заде [206; 207], і далі розширена, наприклад, в моделях, що використовують гранули однакових розмірів з однаковим математичним представленням у вигляді матриць під назвою базових елементів Грасмана [16].

– для *прийняття рішення $F_{f1}(R_x)$, яке залежить від стану ієрархічно вищої системи $F_{f0}(R_{z0})$ та станів нижчих систем $F_{f2}(R_{z2})$*;

*для аналізу перехідних процесів складної нелінійної системи n-го порядку*, яка описується диференціальними рівняннями першого порядку $\frac{dx_i}{dt} = f_i\ (x_1, x_2, ..., x_j, ..., x_n)$ у залежності від часу з урахуванням початкових умов. В результаті числового інтегрування отримуються нові координати стану технічної системи в n-вимірному просторі, що визначають годограф, який дозволяє візуально контролювати сталість системи у часі та залежність перехідних процесів від параметрів та збурень. Для переходу до методу аналізу перехідних процесів у фазовій площині необхідно n змінних $X=(x_1, x_2, \ldots, x_j, \ldots, x_n)$ замінити двома змінними, наприклад, у вигляді

$$u_1(u_2),\ R(t),\ Q(R,t),$$

де $u_1 = \sqrt{x_1^2 + x_2^2 + ... + x_i^2}$, $u_2 = \sqrt{x_{i+1}^2 + x_{i+2}^2 + ... + x_n^2}$; t – час.

– *як теоретична основа створення «загальної теорії всіх знань людства»*, якщо в одній n-вимірній системі осей координат описати всі знання у вигляді цифрових складних залежностей при наявності різних або частково співпадаючих сукупностей осей. При спрощенні системи координат (вилученні осей окремих знань), ця модель в спрощеній системі координат буде давати вірні і підтверджені практикою дані. Подібні моделі ***є шкідливими**, якщо вони не додають нових знань, бо вони лише ускладнюють аналіз, викликають даремні витрати ресурсів та часу на їх створення, освоєння, застосування і спростовування*. З цієї точки зору треба обережно ставитись до подібних пропонованих об'єднуючих моделей знань: вони правдоподібні і навіть вірні, але наносять шкоду і не мають жодної ***нової*** наукової чи практичної цінності.

Нижче наведені найбільш уживані міри відстані та близькості між двома об'єктами.

*1.* ***Відстані між об'єктами*** у вигляді двох векторів з числовими ознаками описується:

– як ***квадрат евклідової відстані***, що дорівнює сумі квадратів різниць числових координат векторів (використовується найчастіше),

$$d_1^2(\omega_k^K;\omega_e^E)=\sum_{j=1}^{n}(x_{kj}^K-x_{ej}^E)^2,$$

або як ***евклідова відстань***, що дорівнює кореню квадратному від суми квадратів різниць числових координат векторів

$$d_2(\omega_k^K;\omega_e^E)=\sqrt{\sum_{j=1}^{n}(x_{kj}^K-x_{ej}^E)^2},$$

де $k = 1, 2,..., K$ – порядкові номери векторів класу $K$; $e = 1, 2,...,$

$E$ – порядкові номери векторів класу $E$; $j = 1, 2,..., n$ – порядкові номери ознак векторів обох класів; $x_{kj}^K$ – $j$-ий елемент $k$-го порівнюваного вектора $\omega_k^K$ класу $K$; $x_{ej}^E$ – $j$-ий елемент $e$ – го порівнюваного вектора $\omega_e^E$ класу $E$.

– як ***відстань по Манхеттену***, що дорівнює сумі абсолютних значень різниць числових координат

$$d_3(\omega_k^K;\omega_e^E)=\sum_{j=1}^{n}\left|x_{kj}^K-x_{ej}^E\right|,$$

– як ***відстань Чебишева***, що дорівнює абсолютному значенню максимальної різниці по одній $j$-ої координаті

$$d_4(\omega_k^K;\omega_e^E)=\max_j\left|x_{kj}^K-x_{ej}^E\right|.$$

– як ***відстань Махаланобіса***, що є мірою відстані між двома точками в просторі, при наявності кореляції між змінними. Наприклад, для двох чи більше некорреліpованних змінних відстань Махаланобіса між точками дорівнює *відстані Евкліда*. Але у випадку, ***коли змінні корельовані***, осі на графіку можуть розглядатися *як не ортогональні* (осі вже не спрямовані під прямими кутами одна по відношенню до іншої). У цьому випадку просте визначення відстані Евкліда не підходить, тоді як відстань Махаланобіса є адекватно визначеною.

– як ***відстань Хеммінга***, що для *двох векторів однакової довжини* дорівнює числу позицій з різними елементами векторів, які можуть мати *лінгвістичне або числове значення*. Це відображується умовною формулою

$$d(x_j^{A1};x_j^{A2})=\sum_{k=1}^{K}\left|x_k^{A1}-x_k^{A2}\right|,$$

де $k = 1, 2,..., K$ – порядкові номери елементів вектора; $x_k^{A1}$, $x_k^{A2}$ – елементи першого та парного вектора.

Приклади *відстані Хеммінга* між двома векторами: *d(1101; 1110) = 2; d (3,7,1,4; 5,7,1,4) = 1; d (нейрон; таксон) = 3.* Очевидно, що такий підхід можна застосувати не лише до символьних елементів, але й до елементів вектора у вигляді слів та речень.

Є багато інших формул по розрахунку відстаней між векторами.

3. *Найбільш уживані міри близькості між двома об'єктами*:

– ***близькість по Хеммінгу***, що дорівнює числу позицій з однаковими елементами векторів. Близькість між об'єктами визначається тим більшим числом, чим менша відстань між ними. Приклади *близькості за Хеммінгом* між двома векторами:

*d(11011; 11101) = 3, бо однакові три розряди;*

*d(3,7,1,4; 5,7,1,4) = 3, бо однакові три розряди;*

*d(нейрон; таксон) = 2, бо однакові два символи.*

Очевидно, що такий підхід можна застосувати не лише до символьних елементів, але й до елементів вектора у вигляді слів та речень.

***Близькість за скалярним добутком векторів:***

– два вектори $V_1$ та $V_2$ однакового n-го порядку з елементами «±1» можуть мати скалярний добуток $S= V_1 \cdot V_2^T$ від мінімального значення «S=- n» (повне неспівпадіння) до максимального значення «S=+n» (повне співпадіння). Скалярний добуток $S= V_1 \cdot V_2^T$ є мірою близькості між векторами, бо чим більше значення S, тим ближчими є між собою вектори.

– Скалярний добуток $S= V_1 \cdot V_2^T$ може використовуватись як міра близькості між векторами і при складанні їх з елементів «1» та «0»: у цьому випадку інформаційно значущими вважають співпадіння елементів «1», а скалярний добуток змінюється у межах «S=0…n».

***Класифікація об'єктів з числовими координатами по відстані.*** Для кожної деякої окремої групи об'єктів у вибірці можна визначити середні значення координат. Отримані середні координати характеризують точку, яка називається *центроїдом групи*. Якщо вибірка розглядається у просторі ознак $x_j$, $j = 1,2,...,n$, то для кожного вхідного невідомого об'єкта можна обчислити відстань Махаланобіса від нього до кожного центроїда групи. Заданий об'єкт належить до тієї групи об'єктів, для якої його відстань Махаланобіса є мінімальною.

Для підвищення надійності розпізнавання образів, всі ознаки (змінні) нормалізують, щоб великі числа не набували більшого впливу на образ. Якщо деякі ознаки мають менший вплив на розпізнавання образів, ніж інші, то вплив цих ознак експерт зменшує додатковим множенням на відповідний зменшувальний коефіцієнт $k_{xj} < 1$.

***Приклад використаня методу розподільчих функцій***

Рівняння прямої, що проходить через дві точки з координатами $(x_1, \ y_1) \in \Omega_1$ і $(x_2, \ y_2) \in \Omega_2$ .

(X-x1) / (x2-x1) = (y-y1) / (y2-y1) або

$y = kx - kx_1 + y_1$,

де k = (y2-y1) / (x2-x1), yi, xi - властивості обраного способу

Рівняння перпендикуляра до прямої, що проходить через точку $M(x_0, y_0)$ де x0 = (x1 + x2) / 2 y0 = (y1 + y2) / 2

$x - x_0 + k(y - y_0) = 0$, або $y = -(x - x_0) / k + y0$

де k = (y2-y1) / (x2-x1)

Розглянемо простий приклад

Нехай необхідно побудувати гіперплощину, що розділяє множину людей на два класи $\Omega_1$ - дорослі та $\Omega_2$ - діти

| | X1 | X2 | X3 | X4 |
|---|---|---|---|---|
| Зріст | 180 | 60 | 190 | 40 |
| вага | 90 | 20 | 100 | 10 |

Навчальна вибірка має вигляд

| $\Omega_1$ | Y1 | Y2 | Y3 | Y4 |
|---|---|---|---|---|
| Зріст | 180 | 170 | 200 | 189 |
| вага | 80 | 75 | 110 | 80 |

| $\Omega_2$ | Y5 | Y6 | Y7 | Y8 |
|---|---|---|---|---|
| Зріст | 100 | 70 | 90 | 40 |
| вага | 30 | 20 | 30 | 20 |

k = (y2-y1) / (x2-x1)= (100-180) / (30-80)= 1,6

Обчислимо координати точки

$M(x_0, y_0)$ де x0 = (x1 + x2) / 2 = (80 + 30) / 2 = 55

y0 = (y1 + y2) / 2 = (180 + 100) / 2 = 140

Рівняння розділяє поверхні матиме вигляд

$y = -(x - x_0) / k + y0$

$y + (x - 55) /1{,}6 - 140 = 0$

Перевіримо, як працює ця функція, підставляючи дані для перевірки

$180 + (90 - 55) /1{,}6 - 140 = 61{,}875 > 0 \rightarrow X1 \in \Omega_1$

$60 + (20 - 55) /1{,}6 - 140 = -101{,}875 < 0 \rightarrow X2 \in \Omega_2$

$190 + (100 - 55) /1{,}6 - 140 = 78{,}125 > 0 \rightarrow X3 \in \Omega_1$

$40 + (10 - 55) /1{,}6 - 140 = -128{,}125 < 0 \rightarrow X4 \in \Omega_2$

***Варіанти завдань***

Класифікувати довільні реальні образи (об'єкти) з лабораторних робіт №1 і №2.

# Додаток А
# Приклад програмного коду на мові С++ в середовищі MS Visual Studio

Лістинг програми:

```
#include <iostream>
#include <Windows.h>
#include <vector>

using namespace std;

class CaloricContent // Базовий клас калорійності
{
public:
double Count_Calories; // Кількість калорій
double Count_Fat; // Кількість жирів
// Параметризований конструктор за замовчуванням:
CaloricContent(double CountCalories = 0.0, double CountFat = 0.0) : Count_Calories(CountCalories),
Count_Fat(CountFat)
{
}
// Функція для визначення кутового коефіцієнта K:
void Angular_Coefficient(double x1, double x2, double y1, double y2, double& K)
{
K = (y2 - y1) / (x2 - x1);
}
// Функція для визначення координат точки перпендикуляру до прямої:
void Coordinates_Point(double x1, double x2, double y1, double y2, double& X0, double& Y0)
{
X0 = (x1 + x2) / 2;
Y0 = (y1 + y2) / 2;
}
// Визначення значення розподільчої функції:
void DistributionFunc(double x, double y, double x0, double y0, double K, double& res)
{
res = y + (x - x0) / K - y0;
}
};

class LowCalorie_Food : public CaloricContent // Низькокалорійна категорія
{
public:
// Параметризований конструктор за замовчуванням:
LowCalorie_Food(double CountCalories = 0.0, double CountFat = 0.0) : CaloricContent(CountCalories,
CountFat)
{
}

};

class ModeratelyCalorie_Food : public CaloricContent // Помірнокалорійна категорія
{
public:
// Параметризований конструктор за замовчуванням:
ModeratelyCalorie_Food(double CountCalories = 0.0, double CountFat = 0.0) :
CaloricContent(CountCalories, CountFat)
{
```

```
}
};

int main()
{
system("chcp 1251");
HANDLE ColorText = GetStdHandle(STD_OUTPUT_HANDLE); // Дескриптор вікна консолі
cout << fixed; // Встановлення режиму повних чисел
cout << " | Метод розпізнавання за розподільчими функціями |" << endl;
vector<LowCalorie_Food>Vec_LowCalorie_Food // Вектор еталонних даних низькокалорійної продукції
{
LowCalorie_Food(47,0.3),
LowCalorie_Food(22,0.31)
,
LowCalorie_Food(35,0.22)
,
LowCalorie_Food(20,0.2),
LowCalorie_Food(85,0.1),
LowCalorie_Food(94,0.2),
LowCalorie_Food(28,0.5),
LowCalorie_Food(69,0.41)
};
vector <ModeratelyCalorie_Food> Vec_ModeratelyCalorie_Food // Вектор еталонних даних помірнококалорійної продукції
{
ModeratelyCalorie_Food(159,13.1),
ModeratelyCalorie_Food(168,12),
ModeratelyCalorie_Food(201,1.1),
ModeratelyCalorie_Food(110,0.5),
ModeratelyCalorie_Food(150,10),
ModeratelyCalorie_Food(105,0.62),
ModeratelyCalorie_Food(151,3),
ModeratelyCalorie_Food(117,0.4)
};
double count_calor = 0.0; // Кількість калорій
double count_fat = 0.0; // Кількість жиру
double K = 0.0; // Кутовий коефіцієнт
double X0 = 0.0; // Координата X0
double Y0 = 0.0; // Координата Y0
double Res = 0.0; // Значення розподільчої функції
bool next = true; // Логічна змінна
SetConsoleTextAttribute(ColorText, 10); // Зміна кольору тексту консолі
cout << "\n=== Калорійність продукції ===" << "\n\n";
do
{
SetConsoleTextAttribute(ColorText,  8);
// Введення даних
cout << "Введіть кількість калорій => ";
cin >> count_calor;
```

```
cout << "Введіть кількість жирів => ";
cin >> count_fat;
cout << "\n";
SetConsoleTextAttribute(ColorText,
3);
// Вектор вказаних даних:
vector <CaloricContent> Vec_Object = { CaloricContent(count_calor,count_fat) };
CaloricContent Obj; // Об'єкт базового класу
// Виклик функцій базового класу і вивід результуючих значень:
Obj.Angular_Coefficient(Vec_LowCalorie_Food[0].Count_Fat,
Vec_ModeratelyCalorie_Food[0].Count_Fat,
Vec_LowCalorie_Food[0].Count_Calories, Vec_ModeratelyCalorie_Food[0].Count_Calories, K);
cout << "Значення кутового коефіцієнту: " << K << endl;
Obj.Coordinates_Point(Vec_LowCalorie_Food[0].Count_Fat,
Vec_ModeratelyCalorie_Food[0].Count_Fat,
Vec_LowCalorie_Food[0].Count_Calories, Vec_ModeratelyCalorie_Food[0].Count_Calories, X0, Y0);
cout << "Координати точки M = [" << X0 << "," << Y0 << "]" << endl;
Obj.DistributionFunc(Vec_Object[0].Count_Fat, Vec_Object[0].Count_Calories, X0, Y0, K, Res);
cout << "Значення розподільчої функції: " << Res << "\n\n";
SetConsoleTextAttribute(ColorText, 14);
// Перевірка значення розподільчої функції на позитивність та
негативність: if (Res > 0)
{
cout << "Продукція за вказанимии значеннями відповідає помірнокалорійній категорії" << endl;
}
else
{
cout << "Продукція за вказанимии значеннями відповідає низькокалорійній категорії" << endl;
}
SetConsoleTextAttribute(ColorText,
3); char symbol = {};
// Пропозиція продовжити вводити дані для наступної продукції:
cout << "\nДля продовженн введення наступних даних введіть [Y], в протилежному випадку [N]
=> ";
cin >> symbol;
if (symbol == 'N')
{
next = false;
}
} while (next == true);
cout << "Завершення програми..." << endl;
Sleep(3000); // Очікування дій протягом 3 секунд
SetConsoleTextAttribute(ColorText, 15);
return 0;
}
```

Демонстрація роботи програми:

```
=== Калорійність продукції ===

Введіть кількість калорій => 50
Введіть кількість жирів => 0.22

Значення кутового коефіцієнту: 8.750000
Координати точки M = [6.700000,103.000000]
Значення розподільчої функції: -53.740571

Продукція за вказаними значеннями відповідає низькокалорійній категорії

Для продовження введення наступних даних введіть [Y], в протилежному випадку [N] => Y
Введіть кількість калорій => 151
Введіть кількість жирів => 11

Значення кутового коефіцієнту: 8.750000
Координати точки M = [6.700000,103.000000]
Значення розподільчої функції: 48.491429

Продукція за вказаними значеннями відповідає помірнокалорійній категорії

Для продовження введення наступних даних введіть [Y], в протилежному випадку [N] => Y
Введіть кількість калорій => 230
Введіть кількість жирів => 18

Значення кутового коефіцієнту: 8.750000
Координати точки M = [6.700000,103.000000]
Значення розподільчої функції: 128.291429

Продукція за вказаними значеннями відповідає помірнокалорійній категорії

Для продовження введення наступних даних введіть [Y], в протилежному випадку [N] => Y
Введіть кількість калорій => 109
Введіть кількість жирів => 5

Значення кутового коефіцієнту: 8.750000
Координати точки M = [6.700000,103.000000]
Значення розподільчої функції: 5.805714

Продукція за вказаними значеннями відповідає помірнокалорійній категорії

Для продовження введення наступних даних введіть [Y], в протилежному випадку [N] => Y
Введіть кількість калорій => 63
Введіть кількість жирів => 0.4

Значення кутового коефіцієнту: 8.750000
Координати точки M = [6.700000,103.000000]
Значення розподільчої функції: -40.720000

Продукція за вказаними значеннями відповідає низькокалорійній категорії

Для продовження введення наступних даних введіть [Y], в протилежному випадку [N] => N
```

Дана програма виконує розпізнавання вхідних образів, використовуючи метод розподільчих функцій. Дана програма, як і на основі прикладу, містить 2 образи по 2 ознаки, які відносяться до загального базового класу калорійності продукції. Перший образ характеризує низькокалорійну продукцію, а другий образ помірнокалорійну продукцію. До ознак відносяться всього дві ознаки: кількість калорій та кількість жирів. В програмі також присутні і еталонні дані категорій, на основі яких і буде здійснюватися розпізнавання вхідних образів, в даному випадку на основі перших еталонних значень. При запуску, програма запропонує ввести

значення кількісних ознак. Програма обробить значення і визначить кутовий коефіцієнт прямої, яка характеризується координатами, де в ролі координат використовуються перше значення еталонних ознак. Далі визначаються координати точки перпендикуляру M(x0,y0), де знову ж використовуються попередні еталонні значення. Після визначення кутового коефіцієнта та координат точки M відбувається визначення значення розподільчої функції, яка в свою чергу використовує вхідні дані, яка були введенні на початку, значення кутового коефіцієнту та координат точки M. На екран виводиться кінцеве значення розподільчої функції і програма вказує до якої категорії внесено продукцію, значення ознак якої були введенні на початку. Саме ж розподілення відбувається таким чином, що якщо кінцеве значення розподільчої функції менше за 0, то образ(продукцію) буде віднесено до низькокалорійної категорії, в іншому випадку, якщо значення більше за 0, то образ буде віднесено до помірнокалорійної категорії. Далі програма пропонує продовжити вводити значення ознак наступної продукції. В разі відмови, виконання програми буде завершено.