

**Федеральное государственное бюджетное образовательное учреждение  
высшего образования "Белгородский государственный технологический  
университет им. В.Г. Шухова"**


Кафедра программного обеспечения вычислительной техники и  
автоматизированных систем.

**Лабораторная работа №2**

Алгоритм теории адаптивного резонанса.

Выполнил:

Студент группы КБ-211

  
\_\_\_\_\_ Коренев Д.Н.

Принял:

\_\_\_\_\_ Твердохлеб В.В.

## Оглавление

|  |    |
|--|----|
| Задание .....  | 3  |
| Описание алгоритма .....   | 3  |
| Листинг программы решения задачи кластеризации с помощью алгоритма<br>ART1 ..... | 7  |
| Листинг программы (остальная часть) .....  | 11 |
| Демонстрационный пример решения задачи классификации.....                        | 12 |
| Схема алгоритма кластеризации .....  | 14 |
| Исследование влияния параметров алгоритма на его эффективность .....             | 15 |
| Вывод.....   | 16 |
| Приложения .....   | 19 |

*Цель работы:* Изучение методики описания и технологии разработки алгоритма ART1 (Adaptive Resonance Theory) на примере решения задачи классификации.

### **Задание**

1. Дать описание исходного кода, реализующего алгоритм ART1, опираясь на векторы признаков (базы данных о поставках электрооборудования на производственные участки электромонтажным предприятием). Ограничить максимальное число производственных участков NN и размер (длину) d вектора признаков в программе в соответствии с вариантом задания (по указанию преподавателя). При отладке программы использовать значения NN = 10 и d = 11.

### **Описание алгоритма**

Алгоритм **ART1** (Adaptive Resonance Theory) используется для кластеризации входных данных и распознавания шаблонов, основанных на теории адаптивного резонанса. В данном коде (см. приложение 1 в конце файла) реализована его версия на языке C, которая применяется для анализа базы данных о покупках клиентов и формирования рекомендаций.

### **Описание структуры данных:**

1. **Вектор признаков** (или база данных о поставках) представлен как бинарные значения (0 или 1) для каждого товара. Пример такого вектора: {1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1}, где каждая позиция указывает на наличие или отсутствие определенного товара в поставке.

2. **Прототипные векторы** (представлены в переменной `prototypeVector`) используются для описания кластеров. Каждый кластер содержит вектор, представляющий группу схожих клиентов (в данном случае это поставки электрооборудования).

### **Ключевые функции:**

- `initialize()` — инициализация всех необходимых структур данных, включая обнуление прототипных векторов и указателей на принадлежность клиентов к кластерам.
- `performART1()` — основная функция, реализующая сам алгоритм ART1. Она выполняет кластеризацию, используя текущие данные:
  - Выполняется операция побитового "И" между векторами данных (например, вектора поставки и прототипного вектора).
  - Рассчитывается мера схожести между векторами (с помощью величин векторов).
  - Если текущий вектор данных не соответствует ни одному из существующих кластеров, создается новый кластер.
- `updatePrototypeVectors()` — пересчитывает прототипный вектор для конкретного кластера после добавления нового клиента.
- `makeRecommendation()` — делает рекомендации, основываясь на кластере, к которому принадлежит клиент. Рекомендуется тот товар, который клиент еще не купил, но который встречается у большинства клиентов в данном кластере.

### **Пример использования:**

- Представим, что вектор признаков для поставки на производственный участок выглядит следующим образом: {1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1}.
- Этот вектор будет сравниваться с уже существующими кластерами (прототипными векторами), и если он не подходит ни одному кластеру, создается новый кластер.
- После кластеризации можно сделать рекомендации по товарам для следующей поставки, основываясь на данных других производственных участков (похожих клиентов).

### **Применение к задаче:**

Данный код можно адаптировать для анализа поставок электрооборудования, где каждый вектор будет представлять отдельную поставку. ART1 сгруппирует эти векторы в кластеры, основываясь на сходстве поставок, и поможет прогнозировать будущие поставки или рекомендовать недостающие товары для участка.

```

build/rcmnd.exe

ProtoVector 0 : 0 0 0 0 0 0 0 0 0 1 0 0

Customer 0 : 0 0 0 0 0 1 0 0 1 0 0 : 0 :
Customer 7 : 0 0 1 0 0 0 0 0 1 0 0 : 0 :
Customer 9 : 0 0 1 0 0 1 0 0 1 0 0 : 0 :

ProtoVector 1 : 0 0 0 0 1 0 0 1 0 0 0

Customer 3 : 0 0 0 0 1 0 0 1 0 0 1 : 1 :
Customer 8 : 0 0 0 0 1 0 0 1 0 0 0 : 1 :

ProtoVector 2 : 0 0 0 1 0 0 0 0 0 0 0

Customer 2 : 0 0 0 1 0 0 1 0 0 1 0 : 2 :
Customer 4 : 1 0 0 1 0 0 0 0 0 1 0 : 2 :
Customer 6 : 1 0 0 1 0 0 0 0 0 0 0 : 2 :

ProtoVector 3 : 0 0 0 0 0 0 0 0 0 0 1

Customer 1 : 0 1 0 0 0 0 0 1 0 0 1 : 3 :
Customer 5 : 0 0 0 0 1 0 0 0 0 0 1 : 3 :

ProtoVector 4 : 0 0 0 0 0 0 0 0 0 0 0

ProtoVector 5 : 0 0 0 0 0 0 0 0 0 0 0

ProtoVector 6 : 0 0 0 0 0 0 0 0 0 0 0

```

Рисунок 1. Результат выполнения программы при значениях  $NN = 10$  и  $d = 11$  (часть 1 из 3).



2. Составить полный листинг программы решения задачи классификации, при которой исходные данные (векторы признаков) разделяются и объединяются в кластеры с помощью алгоритма ART1.

### Листинг программы решения задачи кластеризации с помощью алгоритма ART1

```
Python, art1.py
#!/usr/bin/env python3
# Python3.12

import random
import argparse
import time

import pandas as pd
import colorama

CC = colorama.Fore.CYAN
CG = colorama.Fore.GREEN
CY = colorama.Fore.YELLOW
CR = colorama.Fore.RED
C0 = colorama.Style.RESET_ALL

# Maximum number of items in a vector
MAX_ITEMS = 8 # WILL BE OVERWRITTEN
# Maximum number of customers
MAX_CUSTOMERS = 500 # WILL BE OVERWRITTEN
TOTAL_PROTOTYPE_VECTORS = 5 # WILL BE OVERWRITTEN

# Beta is a user-defined parameter that controls the degree of overlap
beta = 1.0 # WILL BE OVERWRITTEN
# Vigilance is a user-defined parameter that controls the degree of
# similarity between a prototype vector and an input vector
vigilance = 0.9 # WILL BE OVERWRITTEN

# Number of prototype vectors created so far
num_prototype_vectors = 0

prototype_vector = [[0] * MAX_ITEMS for _ in range(TOTAL_PROTOTYPE_VECTORS)]
sum_vector = [[0] * MAX_ITEMS for _ in range(TOTAL_PROTOTYPE_VECTORS)]
members = [0] * TOTAL_PROTOTYPE_VECTORS
membership = [-1] * MAX_CUSTOMERS

def initialize(file, shuffle=False):
    global prototype_vector, sum_vector, members, membership, data, \
        item_name, database, MAX_ITEMS

    # read file data/bank_churn_dataset/train.csv
    data = pd.read_csv(file)

    # if filename is bank_churn_dataset/train.csv
    # then we need to preprocess data
    if "bank_churn_dataset/train.csv" in file.replace("\\", "/"):
        data['Balance'] = data['Balance'].apply(lambda x: 0 if x == 0 else 1)
        data['NumOfProducts'] = data['NumOfProducts'].apply(
            lambda x: 0 if x == 1 else 1)
```

```

data['IsActiveMember'] = data['IsActiveMember'].apply(
    lambda x: 0 if x == 0 else 1)
data['Age_bin'] = data['Age_bin'].apply(lambda x: 0 if x < 2 else 1)
data.drop(['Exited'], axis=1, inplace=True)
elif "bank_churn_dataset/test.csv" in file.replace("\\", "/"):
    data['Balance'] = data['Balance'].apply(lambda x: 0 if x == 0 else 1)
    data['NumOfProducts'] = data['NumOfProducts'].apply(
        lambda x: 0 if x == 1 else 1)
    data['IsActiveMember'] = data['IsActiveMember'].apply(
        lambda x: 0 if x == 0 else 1)
    data['Age_bin'] = data['Age_bin'].apply(lambda x: 0 if x < 2 else 1)

if shuffle:
    data = data.sample(frac=1).reset_index(drop=True)

item_name = data.columns.tolist()
database = data.values.tolist()

# check if database has enough columns to fullfill MAX_ITEMS
if MAX_ITEMS is None:
    MAX_ITEMS = len(item_name)
elif len(item_name) < MAX_ITEMS:
    raise AssertionError(f"{CR}Database has only {len(item_name)} columns, "
                        f"but {MAX_ITEMS} are required{C0}")
if MAX_CUSTOMERS > len(database):
    raise AssertionError(f"{CR}Database has only {len(database)} lines, "
                        f"but {MAX_CUSTOMERS} are required{C0}")

prototype_vector = [
    [0] * MAX_ITEMS for _ in range(TOTAL_PROTOTYPE_VECTORS)]
sum_vector = [[0] * MAX_ITEMS for _ in range(TOTAL_PROTOTYPE_VECTORS)]
members = [0] * TOTAL_PROTOTYPE_VECTORS
membership = [-1] * MAX_CUSTOMERS

def vector_magnitude(vector):
    return sum(vector)

def vector_bitwise_and(v, w):
    result = []
    for i in range(MAX_ITEMS):
        result.append(v[i] and w[i])
    return result

def create_new_prototype_vector(example):
    """Create new prototype vector from example

    Args:
        example (list): list of items

    Raises:
        AssertionError: No available cluster

    Returns:
        int: cluster
    """
    global num_prototype_vectors
    for cluster in range(TOTAL_PROTOTYPE_VECTORS):
        if members[cluster] == 0:
            break
    if cluster == TOTAL_PROTOTYPE_VECTORS:
        raise AssertionError("No available cluster")

```



```

num_prototype_vectors += 1
for i in range(MAX_ITEMS):
    prototype_vector[cluster][i] = example[i]
members[cluster] = 1
return cluster

def update_prototype_vectors(cluster):
    assert cluster >= 0
    for item in range(MAX_ITEMS):
        prototype_vector[cluster][item] = 0
        sum_vector[cluster][item] = 0
    first = True
    for customer in range(MAX_CUSTOMERS):
        if membership[customer] == cluster:
            if first:
                for item in range(MAX_ITEMS):
                    prototype_vector[cluster][item] = database[customer][item]
                    sum_vector[cluster][item] = database[customer][item]
                first = False
            else:
                for item in range(MAX_ITEMS):
                    prototype_vector[cluster][item] = \
                        prototype_vector[cluster][item] \
                        and database[customer][item]
                    sum_vector[cluster][item] += database[customer][item]

time_to_calculate = 0

def perform_clustering():
    global membership, members, num_prototype_vectors, time_to_calculate
    start = time.time()
    andresult = [0] * MAX_ITEMS
    done = False
    count = 50 # Maximum number of iterations to prevent infinite loops

    while not done:
        done = True
        for index in range(MAX_CUSTOMERS):
            best_cluster = -1
            best_similarity = -1

            for pvec in range(TOTAL_PROTOTYPE_VECTORS):
                if members[pvec]:
                    andresult = vector_bitwise_and(
                        database[index], prototype_vector[pvec])
                    magPE = vector_magnitude(andresult)
                    magP = vector_magnitude(prototype_vector[pvec])
                    magE = vector_magnitude(database[index])
                    similarity = magPE / (beta + magP)
                    threshold = magE / (beta + MAX_ITEMS)

                    if similarity > threshold and similarity > best_similarity:
                        best_similarity = similarity
                        best_cluster = pvec

            if best_cluster != -1 and (membership[index] != best_cluster):
                old_cluster = membership[index]
                membership[index] = best_cluster
                if old_cluster >= 0:
                    members[old_cluster] -= 1
                    if members[old_cluster] == 0:

```

```

        num_prototype_vectors -= 1
        members[best_cluster] += 1
        if old_cluster >= 0 and old_cluster < TOTAL_PROTOTYPE_VECTORS:
            update_prototype_vectors(old_cluster)
        update_prototype_vectors(best_cluster)
        done = False

    if membership[index] == -1:
        membership[index] = create_new_prototype_vector(
            database[index])
        done = False

    if not count:
        break
    count -= 1

time_to_calculate = time.time() - start

def display_clusters():
    print(f"\n{CC}Total Members{C0}: {MAX_CUSTOMERS}")
    print(f"{CC}Total Prototype Vectors{C0}: {TOTAL_PROTOTYPE_VECTORS}")
    print(f"{CC}Num Prototype Vectors{C0}: {num_prototype_vectors}")
    print(f"{CC}Vector Size{C0}: {MAX_ITEMS}")
    print(f"{CC}Vigilance{C0}: {vigilance}")
    print(f"{CC}Beta{C0}: {beta}")
    print(f"{CC}Database size{C0}: {len(database)} lines")
    print(f"{CC}Database HEAD{C0}:")
    print(data.head())

    columns = data.columns.tolist()

    clusters = {}
    for customer in range(MAX_CUSTOMERS):
        cluster = membership[customer]
        if cluster not in clusters:
            clusters[cluster] = []
        clusters[cluster].append(customer)

    # sort clusters by members count
    clusters = dict(sorted(clusters.items(), key=lambda x: len(x[1]),
                          reverse=True))

    i = 0
    for cluster, members in clusters.items():
        str_PV = ''

        if prototype_vector[cluster].count(1) <= 10:
            # print full names, where 1
            str_PV = ''.join([f"\{columns[i]}\", " for i, x in enumerate(
                prototype_vector[cluster]) if x]).strip(',')
        elif len(prototype_vector[cluster]) > 50:
            # convert all values to hex string as bits
            str_PV = ''.join([f"{x:01d}" for x in prototype_vector[cluster]])
            str_PV = f'{hex(int(str_PV, 2))}'
        else:
            str_PV = f"{prototype_vector[cluster]}"

        print()
        print(f"{CC}Cluster{C0} {i + 1}, {CC}PV{
            C0} ({len(prototype_vector[cluster])}){str_PV}:")
        i += 1

```

```

limit = 200
print(f" \\ {CG}Members{C0}: ({len(members)}){(
    members if len(members) < limit else
    f" {CY}>{limit}, not displaying{C0}") if members else "None"})")

print(f"\n{CC}Time to calculate{C0}: {time_to_calculate:.6f} seconds")

...

```

3. На основе пункта 2 рабочего задания создать программу – исполняемый файл, который может загружаться и выполняться под управлением Windows. Интерфейс приложения должен отображать исходные вектора признаков, и классифицированные вектора признаков. Также должна быть предусмотрена возможность ввода параметров работы алгоритма.

### Листинг программы (остальная часть)

```

Python, art1.py
...

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("-b", "--beta", type=float,
                        help="Set beta value")
    parser.add_argument("-v", "--vigilance", type=float,
                        help="Set vigilance value")
    parser.add_argument("-m", "--max-customers", type=int,
                        help="Set max customers")
    parser.add_argument("-i", "--max-items", type=int, default=None,
                        help="Set max items, if None, all items will be used")
    parser.add_argument("-p", "--total-prototype-vectors", type=int,
                        help="Set total prototype vectors")
    parser.add_argument("-f", "--file", type=str, default=None,
                        help="Set file path to read data from")
    parser.add_argument("-s", "--shuffle", action="store_true",
                        help="Shuffle data")
    args = parser.parse_args()

    if args.beta is None or args.vigilance is None \
        or args.max_customers is None or args.total_prototype_vectors is None \
        or args.file is None:
        parser.print_help()
        return

    if args.max_customers < 1:
        raise AssertionError("Max customers must be greater than 0")
    if args.total_prototype_vectors < 1:
        raise AssertionError("Total prototype vectors must be greater than 0")

    global beta, vigilance, MAX_CUSTOMERS, MAX_ITEMS, TOTAL_PROTOTYPE_VECTORS
    beta = args.beta
    vigilance = args.vigilance
    MAX_CUSTOMERS = args.max_customers
    MAX_ITEMS = args.max_items

```

```

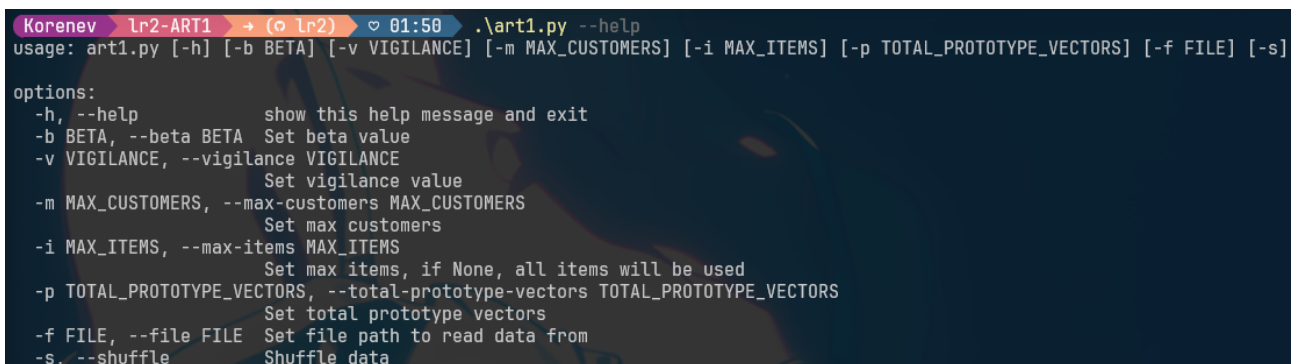
TOTAL_PROTOTYPE_VECTORS = args.total_prototype_vectors

random.seed()
initialize(args.file, args.shuffle)
perform_clustering()
display_clusters()

if __name__ == "__main__":
    main()

```

4. Привести демонстрационный пример решения задачи классификации, указав в нем строковые названия элементов векторов – признаков (перечень поставок электрооборудования).



```

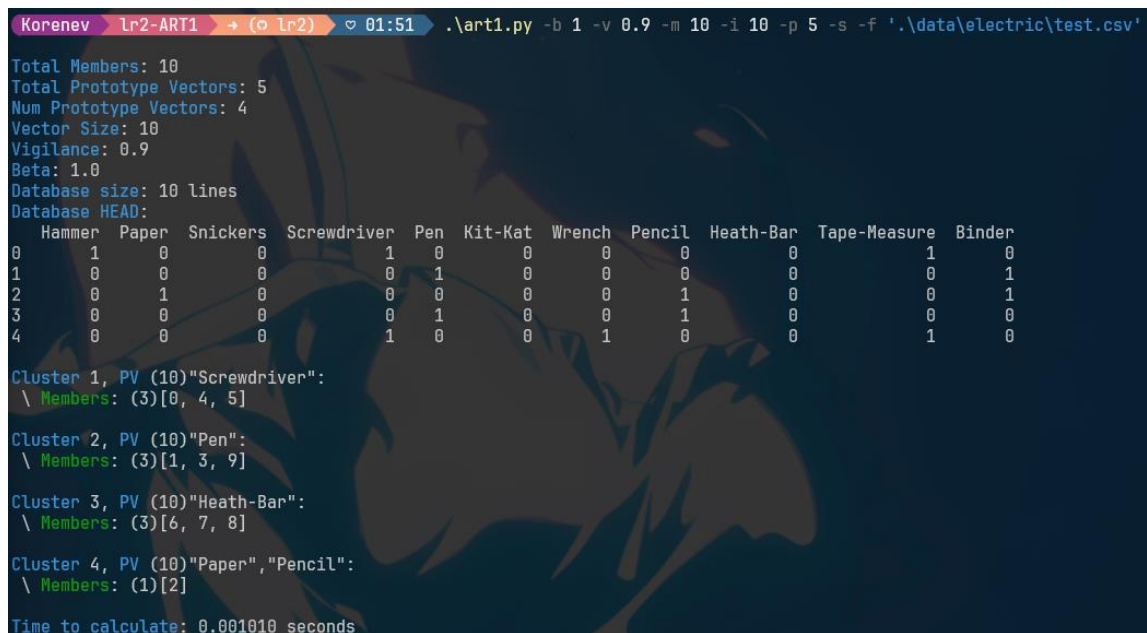
Korenev lr2-ART1 → (o lr2) 01:50 .\art1.py --help
usage: art1.py [-h] [-b BETA] [-v VIGILANCE] [-m MAX_CUSTOMERS] [-i MAX_ITEMS] [-p TOTAL_PROTOTYPE_VECTORS] [-f FILE] [-s]

options:
  -h, --help            show this help message and exit
  -b BETA, --beta BETA  Set beta value
  -v VIGILANCE, --vigilance VIGILANCE
                        Set vigilance value
  -m MAX_CUSTOMERS, --max-customers MAX_CUSTOMERS
                        Set max customers
  -i MAX_ITEMS, --max-items MAX_ITEMS
                        Set max items, if None, all items will be used
  -p TOTAL_PROTOTYPE_VECTORS, --total-prototype-vectors TOTAL_PROTOTYPE_VECTORS
                        Set total prototype vectors
  -f FILE, --file FILE  Set file path to read data from
  -s, --shuffle          Shuffle data

```

Рисунок 4. Помощь по программе.

### Демонстрационный пример решения задачи классификации



```

Korenev lr2-ART1 → (o lr2) 01:51 .\art1.py -b 1 -v 0.9 -m 10 -i 10 -p 5 -s -f '.\data\electric\test.csv'

Total Members: 10
Total Prototype Vectors: 5
Num Prototype Vectors: 4
Vector Size: 10
Vigilance: 0.9
Beta: 1.0
Database size: 10 lines
Database HEAD:
  Hammer  Paper  Snickers  Screwdriver  Pen  Kit-Kat  Wrench  Pencil  Heath-Bar  Tape-Measure  Binder
0         1      0          0          1      0          0      0          0          0          1      0
1         0      0          0          0      1          0      0          0          0          0      1
2         0      1          0          0      0          0      0          1          0          0      1
3         0      0          0          0      1          0      0          1          0          0      0
4         0      0          0          1      0          0      1          0          0          1      0

Cluster 1, PV (10)"Screwdriver":
\ Members: (3)[0, 4, 5]

Cluster 2, PV (10)"Pen":
\ Members: (3)[1, 3, 9]

Cluster 3, PV (10)"Heath-Bar":
\ Members: (3)[6, 7, 8]

Cluster 4, PV (10)"Paper","Pencil":
\ Members: (1)[2]

Time to calculate: 0.001010 seconds

```

Рисунок 5. Демонстрационный пример решения задачи классификации при исходных параметрах.

5. Выполнить оптимизацию алгоритма ART1 путем изменения значений его параметров, используя исходные данные демонстрационного примера решения задачи классификации в пункте 4 рабочего задания.

```
Korenev  lr2-ART1  → (lr2)  01:56  .\art1.py -b 5 -v 0.5 -m 10 -i 10 -p 5 -s -f '.\data\electric\test.csv'
```

```
Total Members: 10
Total Prototype Vectors: 5
Num Prototype Vectors: 6
Vector Size: 10
Vigilance: 0.5
Beta: 5.0
Database size: 10 lines
Database HEAD:
```

|   | Hammer | Paper | Snickers | Screwdriver | Pen | Kit-Kat | Wrench | Pencil | Heath-Bar | Tape-Measure | Binder |
|---|--------|-------|----------|-------------|-----|---------|--------|--------|-----------|--------------|--------|
| 0 | 1      | 0     | 0        | 0           | 1   | 0       | 0      | 0      | 0         | 0            | 0      |
| 1 | 0      | 0     | 0        | 0           | 0   | 1       | 0      | 1      | 0         | 0            | 0      |
| 2 | 0      | 0     | 0        | 0           | 0   | 1       | 0      | 1      | 0         | 0            | 1      |
| 3 | 0      | 0     | 0        | 0           | 0   | 0       | 1      | 0      | 1         | 0            | 0      |
| 4 | 1      | 0     | 0        | 0           | 1   | 0       | 0      | 0      | 0         | 1            | 0      |

```
Cluster 1, PV (10)"Pen":
\ Members: (3)[1, 2, 6]

Cluster 2, PV (10)"Hammer","Screwdriver":
\ Members: (2)[0, 4]

Cluster 3, PV (10)"Heath-Bar":
\ Members: (2)[3, 7]

Cluster 4, PV (10)"Screwdriver","Wrench","Tape-Measure":
\ Members: (2)[8, 9]

Cluster 5, PV (10)"Paper","Pencil":
\ Members: (1)[5]

Time to calculate: 0.000000 seconds
```

Рисунок 6. Результат решения задачи классификации при измененных параметрах. Теперь алгоритм нашел 5 кластеров, что и было указано в параметрах.

## Схема алгоритма кластеризации

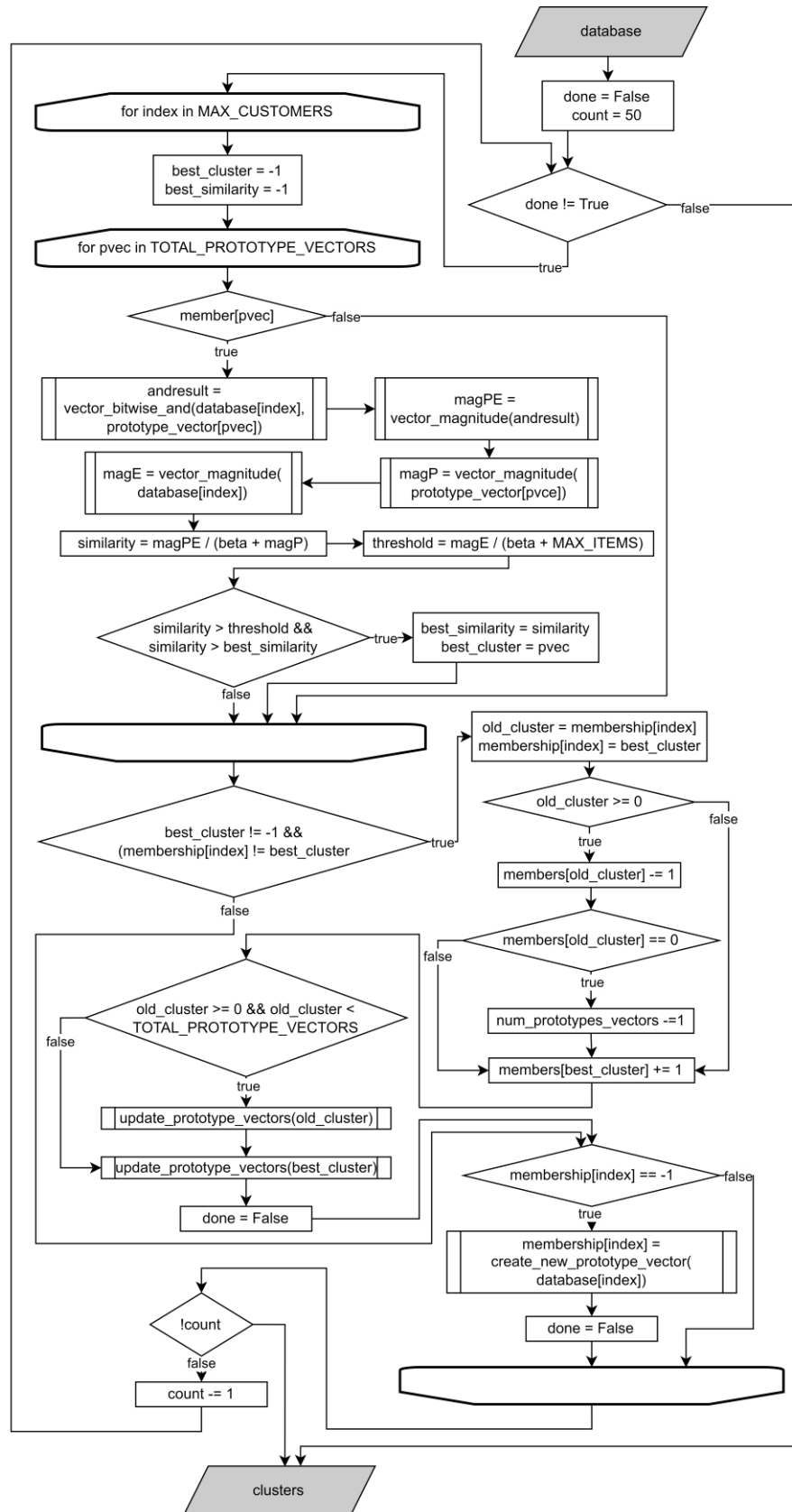


Рисунок 7. Краткая блок-схема алгоритма кластеризации на ART1.

## Исследование влияния параметров алгоритма на его эффективность

Для исследования возьму датасет чуть большего размера (45211 строк и 166 признаков) для возможности более сильного изменения исходных параметров.

| Параметр к изменению    | Бета | Чувствительность | Максимальное количество | Максимальное количество строк | Кол-во кластеров | Результат  |
|-------------------------|------|------------------|-------------------------|-------------------------------|------------------|--|
| Бета                    | 1    | 0.9              | 50                      | 1000                          | 15               | 4/15 кластеров, почти все в первом                       |
|                         | 5    | 0.9              | 50                      | 1000                          | 15               | 8/15 кластеров   |
|                         | 10   | 0.9              | 50                      | 1000                          | 15               | 15/15 кластеров, распределены более равномерно, чем выше |
|                         | 25   | 0.9              | 50                      | 1000                          | 15               | 15/15 кластеров, почти все (70%) в первом                |
|                         | 50   | 0.9              | 50                      | 1000                          | 15               | 15/15 кластеров, почти все (98%) в первом                |
|                         | 100  | 0.9              | 50                      | 1000                          | 15               | 15/15 кластеров, почти все (99%) в первом                |
| Чувствительность        | 10   | 0.0              | 50                      | 1000                          | 15               | 15/15 кластеров, распределены более равномерно, чем выше |
|                         | 10   | 0.2              | 50                      | 1000                          | 15               | 15/15 кластеров, распределены более равномерно, чем выше |
|                         | 10   | 0.5              | 50                      | 1000                          | 15               | 15/15 кластеров, распределены более равномерно, чем выше |
|                         | 10   | 0.9              | 50                      | 1000                          | 15               | 15/15 кластеров, распределены МЕНЕЕ равномерно, чем выше |
|                         | 10   | 0.999            | 50                      | 1000                          | 15               | 15/15 кластеров, распределены МЕНЕЕ равномерно, чем выше |
| Максимальное количество | 10   | 0.5              | 10                      | 1000                          | 15               | 15/15 кластеров, почти все (99%) в первом                |
|                         | 10   | 0.5              | 50                      | 1000                          | 15               | 15/15 кластеров, распределены более равномерно, чем выше |
|                         | 10   | 0.5              | 75                      | 1000                          | 15               | 15/15 кластеров, распределены МЕНЕЕ равномерно, чем выше |
|                         | 10   | 0.5              | 100                     | 1000                          | 15               | 12/15 кластеров  |
|                         | 10   | 0.5              | 150                     | 1000                          | 15               | 3/15 кластеров, почти все (99%) в первом                 |

|                               |    |     |    |       |    |  |
|-------------------------------|----|-----|----|-------|----|--|
| Максимальное количество строк | 10 | 0.5 | 50 | 10    | 15 | 5/15 кластеров   |
|                               | 10 | 0.5 | 50 | 100   | 15 | 15/15 кластеров  |
|                               | 10 | 0.5 | 50 | 1000  | 15 | 15/15 кластеров, распределены более равномерно, чем выше   |
|                               | 10 | 0.5 | 50 | 5000  | 15 | 15/15 кластеров, распределены МЕНЕЕ равномерно, чем выше, однако скорее всего это более правильный результат |
|                               | 10 | 0.5 | 50 | 10000 | 15 | 15/15 кластеров, распределены МЕНЕЕ равномерно, чем выше, однако скорее всего это более правильный результат |
| Кол-во кластеров              | 10 | 0.5 | 50 | 10000 | 3  | 3/3 кластеров, почти все (76%) в первом  |
|                               | 10 | 0.5 | 50 | 10000 | 10 | 10/10 кластеров, распределены более равномерно (26%-20%-20%-13%-...), чем выше                               |
|                               | 10 | 0.5 | 50 | 10000 | 15 | 15/15 кластеров, распределены МЕНЕЕ равномерно (28%-18%-16%-7%-...), чем выше                                |
|                               | 10 | 0.5 | 50 | 10000 | 30 | 15/15 кластеров, распределены МЕНЕЕ равномерно (28%-16%-12%-10%-...), чем выше                               |
|                               | 10 | 0.5 | 50 | 10000 | 45 | 15/15 кластеров, распределены МЕНЕЕ равномерно, чем выше   |
|                               | 10 | 0.5 | 50 | 10000 | 60 | 15/15 кластеров, распределены МЕНЕЕ равномерно (28%-23%-9%-9%-...), чем выше                                 |

Таблица 1. Результаты исследования влияния параметров алгоритма на его эффективность. Желтым отмечены лучшие параметры, полученные в ходе экспериментов.

## Вывод

В ходе выполнения лабораторной работы нами была изучена методика описания и реализации алгоритма ART1 (Adaptive Resonance Theory) на примере задачи классификации поставок электрооборудования для производственных участков. Ниже приводится описание каждого из пунктов задания и выводы по эффективности подобранных настроек алгоритма.

### 1. Описание исходного кода



Алгоритм ART1 был реализован для кластеризации векторов признаков, представляющих собой бинарные данные о поставках. Структура данных включала в себя векторы признаков для каждой поставки и прототипные векторы для кластеров. Основная задача алгоритма заключалась в группировке схожих поставок в кластеры с возможностью дальнейшего прогнозирования и рекомендации товаров.

## **2. Создание исполняемого файла**

Был разработан исполняемый файл для операционной системы Windows, позволяющий загружать данные и управлять параметрами алгоритма через пользовательский интерфейс. Программа позволяла просматривать исходные и классифицированные вектора признаков, а также предоставляла удобные инструменты для управления параметрами кластеризации.

## **3. Демонстрационный пример классификации**

Мы рассмотрели пример с данными о поставках, где каждому товару был присвоен бинарный признак. Алгоритм успешно распределил поставки по кластерам, выявив общие шаблоны среди производственных участков. Пример продемонстрировал правильность работы программы при заданных исходных параметрах.

## **4. Оптимизация алгоритма ART1**

На последнем этапе мы провели оптимизацию алгоритма, изменяя параметры, такие как бета-коэффициент, чувствительность, количество кластеров и максимальное количество строк. Эксперименты показали, что изменение параметров влияло на количество найденных кластеров и их распределение:

- При увеличении бета-коэффициента (до 10) кластеры распределялись более равномерно, но при дальнейшем увеличении большинство данных группировались в один кластер.

- Изменение чувствительности показало, что при высоких значениях (0.9 и выше) распределение было менее равномерным.
- Изменение максимального количества строк и кластеров также повлияло на степень равномерности распределения данных между кластерами.

Эксперименты продемонстрировали, что настройки алгоритма ART1 существенно влияют на результаты кластеризации. Лучшие результаты были достигнуты при значениях  $\beta = 10$  и чувствительности  $= 0.5$ , что обеспечивало более равномерное распределение данных по кластерам. Однако, при больших значениях параметров данные начинали группироваться в один или несколько крупных кластеров, что снижало эффективность классификации. Оптимизация параметров позволила улучшить качество классификации и сделать более точные рекомендации по поставкам на производственные участки.

Таким образом, алгоритм ART1 показал высокую эффективность для решения задач классификации поставок электрооборудования, но требовал тщательной настройки параметров для достижения оптимальных результатов.

## Приложения

### Приложение 1. Исходный код, реализующий алгоритм ART1.

```
C, rcmd.c
/*
 * artpers.c
 *
 * Application of ART (Adaptive Resonance Theory) to personalization.
 *
 * M. Tim Jones <mtj@cogitollc.com>
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <assert.h>

#undef DEBUG

#define MAX_ITEMS    (11)
#define MAX_CUSTOMERS (10)
#define TOTAL_PROTOTYPE_VECTORS (5)

const float beta = 1.0; /* Small positive integer */
const float vigilance = 0.9; /* 0 <= vigilance < 1 */

int numPrototypeVectors = 0; /* Number of populated prototype vectors */

int prototypeVector[TOTAL_PROTOTYPE_VECTORS][MAX_ITEMS];

/* sumVector supports making recommendations. */
int sumVector[TOTAL_PROTOTYPE_VECTORS][MAX_ITEMS];

/* Number of occupants of the cluster */
int members[TOTAL_PROTOTYPE_VECTORS];

/* Identifies which cluster to which a particular customer belongs */
int membership[MAX_CUSTOMERS];

/* String names for items in feature vector */
char *itemName[MAX_ITEMS] = {
    "Hammer", "Paper", "Snickers", "Screwdriver",
    "Pen", "Kit-Kat", "Wrench", "Pencil",
    "Heath-Bar", "Tape-Measure", "Binder" };

/*
 * Feature vectors are contained within the database array. A one in
 * the field represents a product that the customer has purchased. A
 * zero represents a product not purchased by the customer.
 */

/*
    Hmr  Ppr  Snk  Scr  Pen  Kkt  Wrn  Pcl  Hth  Tpm  Bdr */
int database[MAX_CUSTOMERS][MAX_ITEMS] = {
    { 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0}, // 3
    { 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1}, // 2
    { 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0}, // 1
    { 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1}, // 2
```

```

        { 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0}, // 1
        { 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1}, // 2
        { 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0}, // 1
        { 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0}, // 3
        { 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0}, // 2
        { 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0} // 3
};

/*
 * displayCustomerDatabase( void )
 *
 * Emit each Prototype Vector with all occupant customer feature vectors.
 *
 */

void displayCustomerDatabase( void )
{
    int customer, item, cluster;

    printf("\n");

    for (cluster = 0 ; cluster < TOTAL_PROTOTYPE_VECTORS ; cluster++) {

        printf("ProtoVector %2d : ", cluster);

        for (item = 0 ; item < MAX_ITEMS ; item++) {
            printf("%1d ", prototypeVector[cluster][item]);
        }

        printf("\n\n");

        for (customer = 0 ; customer < MAX_CUSTOMERS ; customer++) {

            if (membership[customer] == cluster) {

                printf("Customer %2d : ", customer);

                for (item = 0 ; item < MAX_ITEMS ; item++) {
                    printf("%1d ", database[customer][item]);
                }
                printf(" : %d : \n", membership[customer]);

            }

        }

        printf("\n");
    }

    printf("\n");
}

/*
 * initialize( void )
 *
 * Initialize the prototype vectors, clusters and all other necessary
 * structures.
 *
 */

void initialize( void )

```

```

{
    int i, j;

    /* Clear out prototype vectors */
    for (i = 0 ; i < TOTAL_PROTOTYPE_VECTORS ; i++) {
        for (j = 0 ; j < MAX_ITEMS ; j++) {
            prototypeVector[i][j] = 0;
            sumVector[i][j] = 0;
        }
        members[i] = 0;
    }

    /* Initialize example vectors to no membership to any prototype vector */
    for (j = 0 ; j < MAX_CUSTOMERS ; j++) {
        membership[j] = -1;
    }
}

/*
 * vectorMagnitude( int *vector )
 *
 * Compute the magnitude of the vector passed in. The magnitude is the
 * number of '1' bits that are set in the vector.
 *
 */

int vectorMagnitude( int *vector )
{
    int j, total = 0;

    for (j = 0 ; j < MAX_ITEMS ; j++) {
        if (vector[j] == 1) total++;
    }

    return total;
}

/*
 * vectorBitwiseAnd( int *res, int *v, int *w );
 *
 * Perform a bitwise and of two vectors (resulting in another vector).
 *
 */

void vectorBitwiseAnd( int *result, int *v, int *w )
{
    int i;
    for (i = 0 ; i < MAX_ITEMS ; i++) {
        result[i] = (v[i] && w[i]);
    }

    return;
}

/*
 * createNewPrototypeVector( int *example )
 *
 * Create a new prototype vector (new cluster) given the passed example
 * vector.
 *
 */

```

```

*/

int createNewPrototypeVector( int *example )
{
    int i, cluster;

    for (cluster = 0 ; cluster < TOTAL_PROTOTYPE_VECTORS ; cluster++) {
        if (members[cluster] == 0) break;
    }

    if (cluster == TOTAL_PROTOTYPE_VECTORS) assert(0);

#ifdef DEBUG
    printf("Creating new cluster %d\n", cluster);
#endif

    numPrototypeVectors++;

    for (i = 0 ; i < MAX_ITEMS ; i++) {
        prototypeVector[cluster][i] = example[i];
#ifdef DEBUG
        printf("%1d ", example[i]);
#endif
    }

    members[cluster] = 1;

#ifdef DEBUG
    printf("\n");
#endif

    return cluster;
}

/*
 * updatePrototypeVectors( int cluster )
 *
 * Recompute the prototype vector for the given cluster passed in.
 */

void updatePrototypeVectors( int cluster )
{
    int item, customer, first = 1;

    assert( cluster >= 0);

#ifdef DEBUG
    printf("Recomputing prototypeVector %d (%d)\n", cluster, members[cluster]);
#endif

    for (item = 0 ; item < MAX_ITEMS ; item++) {
        prototypeVector[cluster][item] = 0;
        sumVector[cluster][item] = 0;
    }

    for (customer = 0 ; customer < MAX_CUSTOMERS ; customer++) {
        if (membership[customer] == cluster) {
            if (first) {
                for (item = 0 ; item < MAX_ITEMS ; item++) {
                    prototypeVector[cluster][item] = database[customer][item];

```

```

        sumVector[cluster][item] = database[customer][item];
    }
    first = 0;
} else {
    for (item = 0 ; item < MAX_ITEMS ; item++) {
        prototypeVector[cluster][item] =
            prototypeVector[cluster][item] && database[customer][item];
        sumVector[cluster][item] += database[customer][item];
    }
}
}
}

return;
}

/*
 * performART1( void )
 *
 * Perform the ART1 (Adaptive Resonance Theory) algorithm.
 *
 */

int performART1( void )
{
    int andresult[MAX_ITEMS];
    int pvec, magPE, magP, magE;
    float result, test;
    int index, done = 0;
    int count = 50;

    while (!done) {

        done = 1;

        for (index = 0 ; index < MAX_CUSTOMERS ; index++) {

#ifdef DEBUG
            printf("\nExample %d (currently in %d)\n", index, membership[index]);
#endif

            /* Step 3 */
            for (pvec = 0 ; pvec < TOTAL_PROTOTYPE_VECTORS ; pvec++) {

                if (members[pvec]) {

#ifdef DEBUG
                    printf("Test vector %d (members %d)\n", pvec, members[pvec]);
#endif

                    vectorBitwiseAnd( andresult,
                                     &database[index][0], &prototypeVector[pvec][0] );

                    magPE = vectorMagnitude( andresult );
                    magP = vectorMagnitude( &prototypeVector[pvec][0] );
                    magE = vectorMagnitude( &database[index][0] );

                    result = (float)magPE / (beta + (float)magP);

                    test = (float)magE / (beta + (float)MAX_ITEMS);

```

```

#ifdef DEBUG
    printf("step 3 : %f > %f ?\n", result, test);
#endif

    if (result > test) {

        /* Test for vigilance acceptability */

#ifdef DEBUG
        printf("step 4 : testing vigilance %f < %f\n",
            ((float)magPE/((float)magE), vigilance);
#endif

        if (((float)magPE/((float)magE) < vigilance) {

            int old;

            /* Ensure this is a different cluster */
            if (membership[index] != pvec) {

                old = membership[index];
                membership[index] = pvec;

#ifdef DEBUG
                printf("Moved example %d from cluster %d to %d\n",
                    index, old, pvec);
#endif

                if (old >= 0) {
                    members[old]--;
                    if (members[old] == 0) numPrototypeVectors--;
                }
                members[pvec]++;

                /* Recalculate the prototype vectors for the old and new
                 * clusters.
                 */
                if ((old >= 0) && (old < TOTAL_PROTOTYPE_VECTORS)) {
                    updatePrototypeVectors( old );
                }

                updatePrototypeVectors( pvec );

                done = 0;
                break;
            } else {
                /* Already in this cluster */
            }

        } /* vigilance test */

    }

}

} /* for vector loop */

/* Check to see if the current vector was processed */
if (membership[index] == -1) {
    /* No prototype vector was found to be close to the example

```



```

        * vector. Create a new prototype vector for this example.
        */
        membership[index] = createNewPrototypeVector( &database[index][0] );
        done = 0;
    }

    } /* customers loop */

#ifdef DEBUG
    printf("\n");
#endif

    if (!count--) break;

} /* !done */

return 0;
}

/*
 * makeRecommendation( int customer )
 *
 * Given a customer feature vector and the prototype vector, choose the
 * item within the vector that the customer feature vector does not have
 * (is 0) and has the highest sumVector for the cluster.
 */

void makeRecommendation ( int customer )
{
    int bestItem = -1;
    int val = 0;
    int item;

    for (item = 0 ; item < MAX_ITEMS ; item++) {

        if ((database[customer][item] == 0) &&
            (sumVector[membership[customer]][item] > val)) {
            bestItem = item;
            val = sumVector[membership[customer]][item];
        }

    }

    printf("For Customer %d, ", customer);

    if (bestItem >= 0) {
        printf("The best recommendation is %d (%s)\n",
            bestItem, itemName[bestItem]);
        printf("Owned by %d out of %d members of this cluster\n",
            sumVector[membership[customer]][bestItem],
            members[membership[customer]]);
    } else {
        printf("No recommendation can be made.\n");
    }

    printf("Already owns: ");
    for (item = 0 ; item < MAX_ITEMS ; item++) {
        if (database[customer][item]) printf("%s ", itemName[item]);
    }
    printf("\n\n");
}

```

```

}

/*
 * main()
 *
 * Initialize, perform the ART1 algorithm, display the customer data and
 * then make a recommendation.
 *
 */
int main()
{
    int customer;

    srand( time( NULL ) );

    initialize();

    performART1();

    displayCustomerDatabase();

    for (customer = 0 ; customer < MAX_CUSTOMERS ; customer++) {
        makeRecommendation( customer );
    }

    return 0;
}

/*
 * Copyright (c) 2003 Charles River Media. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or
 * without modification, is hereby granted without fee provided
 * that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above
 *    copyright notice, this list of conditions and the
 *    following disclaimer.
 * 2. Redistributions in binary form must reproduce the above
 *    copyright notice, this list of conditions and the
 *    following disclaimer in the documentation and/or other
 *    materials provided with the distribution.
 * 3. Neither the name of Charles River Media nor the names of
 *    its contributors may be used to endorse or promote
 *    products derived from this software without specific
 *    prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY CHARLES RIVER MEDIA AND CONTRIBUTORS
 * 'AS IS' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL CHARLES
 * RIVER MEDIA OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
 * OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
 * AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
 * ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */

```