МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №2

по дисциплине: Основы ИИ
тема: «Алгоритм теории адаптивного резонанса»

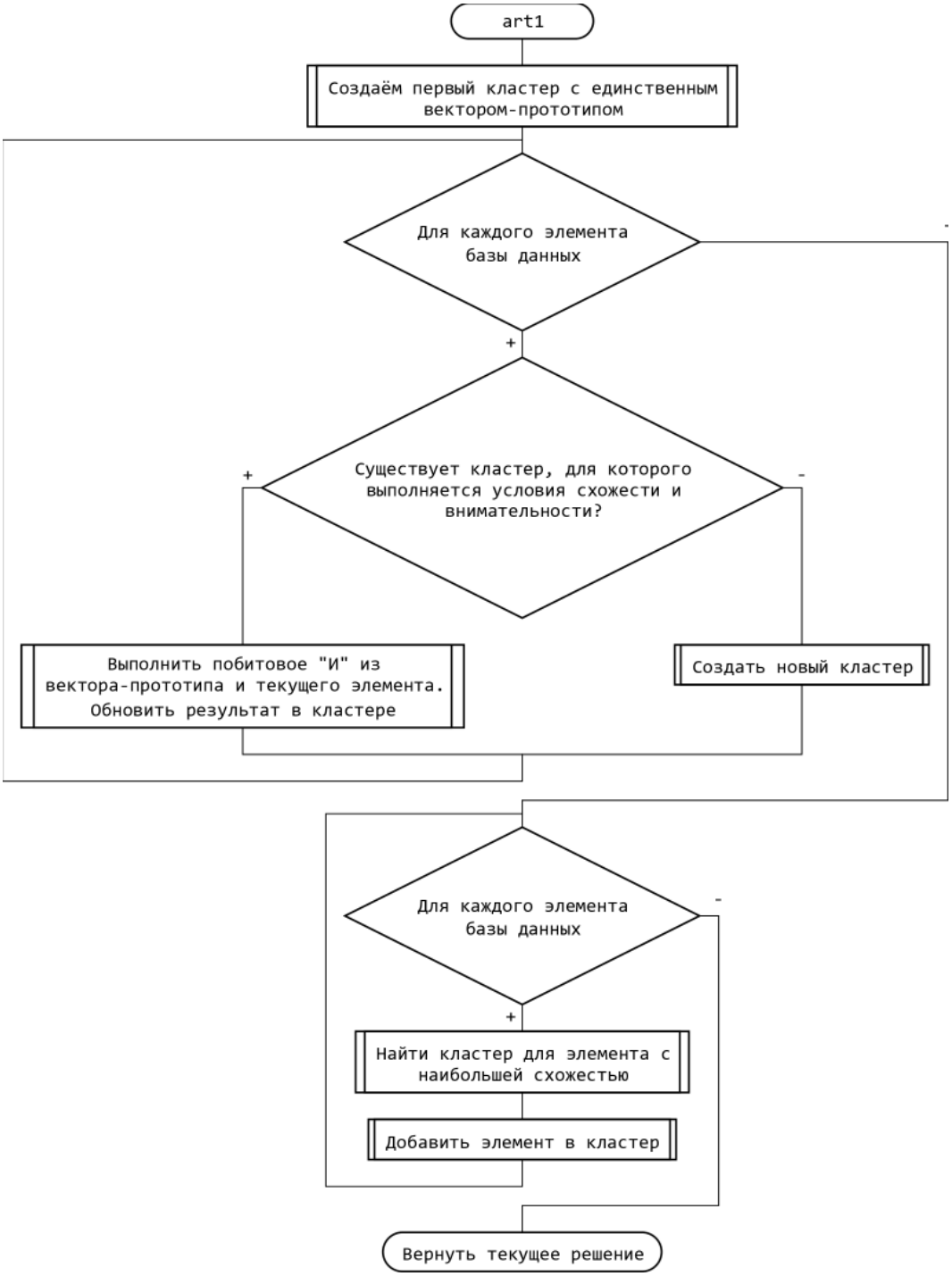
Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
пр. Твердохлеб Виталий Викторович

Белгород 2025 г.

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

Ход выполнения работы




```rust
use core::f64;
use std::error::Error;

use crate::utils::{ART1Clusters, ART1Database, ART1Config};

pub fn art1(database: &ART1Database, config: &ART1Config) -> Result<ART1Clusters, Box<dyn Error>> {
    let mut art1_clusters = ART1Clusters { clusters: vec![] };

    if database.dimension == 0 || database.dataset.len() == 0 {
```

```
        return Ok(art1_clusters);
    }

    // Создаём первый кластер с единственным вектором-прототипом -- первым элементом из списка.
    create_cluster(config, &mut art1_clusters, database.dataset[0])?;

    // Инициализируем кластеры
    for database_element in database.dataset.iter() {
        // Найти подходящий кластер
        match find_fitting_cluster(database, config, &art1_clusters, *database_element)? {
            Some(cluster_index) => {
                // Если кластер найден, то модифицируем вектор-прототип
                art1_clusters.clusters[cluster_index][0] &= *database_element;
            },
            None => {
                // Иначе, пытаемся создать новый кластер
                if let Err(error) = create_cluster(&config, &mut art1_clusters, *database_element) {
                    eprintln!("{error}")
                }
            }
        }
    }

    // Распределяем элементы базы данных по полученным кластерам
    for database_element in database.dataset.iter() {
        // Найти подходящий кластер
        if let Some(cluster_index) = find_closest_cluster(database, config, &art1_clusters, *database_element)? {
            // Если кластер найден, то модифицируем вектор-прототип
            push_cluster(&mut art1_clusters, cluster_index, *database_element)?;
        }
    }

    Ok(art1_clusters)
}

fn create_cluster(
    config: &ART1Config,
    clusters: &mut ART1Clusters,
    prototype: u64
) -> Result<(), Box<dyn Error>> {
    if clusters.clusters.len() >= config.max_clusters {
        return Err(format!("Clusters are overflowing. Consider expanding").into());
    }

    clusters
        .clusters
        .push(vec![prototype]);

    Ok(())
}

fn push_cluster(
    clusters: &mut ART1Clusters,
    cluster_index: usize,
```

```
    value: u64
) -> Result<(), Box<dyn Error>> {
    if cluster_index >= clusters.clusters.len() {
        return Err(format!("A cluster with index {cluster_index} doesn't exists").into());
    }

    clusters.clusters[cluster_index].push(value);
    Ok(())
}

// Найти подходящий кластер
fn find_fitting_cluster(
    database: &ART1Database,
    config: &ART1Config,
    clusters: &ART1Clusters,
    value: u64
) -> Result<Option<usize>, Box<dyn Error>> {
    for cluster_index in 0..clusters.clusters.len() {
        if clusters.clusters[cluster_index].len() == 0 {
            return Err("Prototype is missing in cluster".into());
        }

        let prototype = clusters.clusters[cluster_index][0];

        // Проводим проверку на схожесть и на внимание
        if check_similarity(database, config, prototype, value) &&
        check_attention(config, prototype, value) {
            return Ok(Some(cluster_index));
        }
    }

    Ok(None)
}


// Найти самый подходящий кластер, даже если он подходит не очень сильно
fn find_closest_cluster(
    database: &ART1Database,
    config: &ART1Config,
    clusters: &ART1Clusters,
    value: u64
) -> Result<Option<usize>, Box<dyn Error>> {
    let mut closest_val = -f64::INFINITY;
    let mut closest_val_ind: Option<usize> = None;

    for cluster_index in 0..clusters.clusters.len() {
        if clusters.clusters[cluster_index].len() == 0 {
            return Err("Prototype is missing in cluster".into());
        }

        let prototype = clusters.clusters[cluster_index][0];

        // Проводим проверку на схожесть и на внимание
        let similarity = check_similarity_diff(database, config, prototype, value);
```

```
        if similarity > closest_val {
            closest_val = similarity;
            closest_val_ind = Some(cluster_index);
        }
    }

    Ok(closest_val_ind)
}

// Проверка на схожесть
fn check_similarity(
    database: &ART1Database,
    config: &ART1Config,
    prototype: u64,
    value: u64,
) -> bool {
    check_similarity_diff(database, config, prototype, value) > 0.
}

fn check_similarity_diff(
    database: &ART1Database,
    config: &ART1Config,
    prototype: u64,
    value: u64,
) -> f64 {
    ((prototype & value).count_ones() as f64 / (config.beta + prototype.count_ones() as f64))
        - (value.count_ones() as f64 / (config.beta + database.dimension as f64))
}

// Проверка на внимание
fn check_attention(
    config: &ART1Config,
    prototype: u64,
    value: u64,
) -> bool {
    ((prototype & value).count_ones() as f64 / value.count_ones() as f64) < config.attention
}
```

Ссылка на репозиторий

Опыт №1: обычный сценарий

$N = 100$

$\beta = 1$

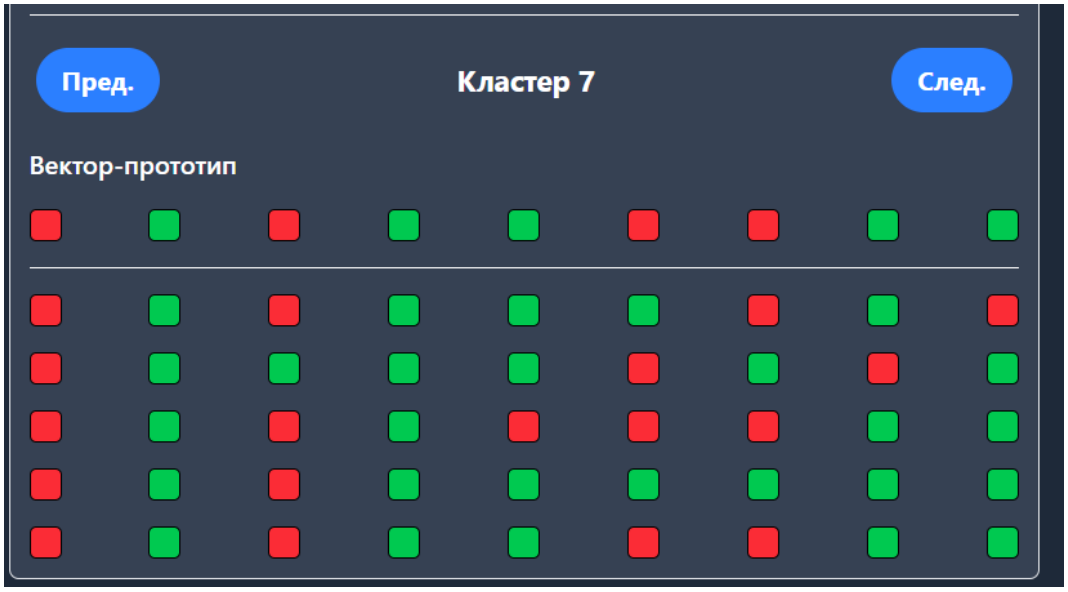
$\rho = 0,7$

Обычный сценарий. В результате получили равномерную кластеризацию.

Опыт №2: изменение β-параметра
$N = 100$
$\beta = 0, 1/10$
$\rho = 0, 7$

Попробуем уменьшить и увеличить параметр β. В основном, кластеризация не изменилась очень сильно. Однако при уменьшении этого параметра должно отдаваться предпочтение первым кластерам. При увеличении - наоборот.

Опыт №3: СДВГ
$N = 100$
$\beta = 1$
$\rho = 0, 01$



Попробуем уменьшить параметр ρ. Количество кластеров уменьшилось, и они стали более общими.

Опыт №5: ОКР
$N = 100$
$\beta = 1$
$\rho = 1$

Попробуем увеличить параметр ρ. Разбиение сильно изменилось и ”сломалось”. Теперь кластеры состоят только из самих элементов. Высокий параметр внимательности делает кластеризацию бесполезной.

Вывод: в ходе лабораторной работы изучили методики описания и технологии разработки алгоритма ART1 (Adaptive Resonance Theory) на примере решения задачи классификации.