МИНОБРНАУКИ РОССИИ
ФГБОУ ВО «СГУ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

НЕЙРОННЫЕ ИММУННЫЕ СЕТИ

РЕФЕРАТ

студента 5 курса 531 группы
направления 10.05.01 — Компьютерная безопасность
факультета КНиИТ
Токарева Никиты Сергеевича

Проверил
доцент _____ И. И. Слеповичев

Саратов 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С развитием области компьютерных наук и техники общество столкнулось с возрастающей проблемой киберпреступности. Одним из аспектов этой проблемы является разработка и распространение вредоносных программ, известных как компьютерные вирусы. В настоящее время обеспечение защиты компьютерных систем от подобных вредоносных программ является приоритетным направлением в области обеспечения информационной безопасности. Традиционные методы, основанные на сигнатурном поиске для выявления компьютерных вирусов, эффективны в обнаружении известных угроз, но оказываются неэффективными при обнаружении неизвестных вредоносных программ. Время, проходящее с момента появления нового компьютерного вируса до его обнаружения специалистами антивирусной индустрии, может быть значительным, что позволяет современным вредоносным программам распространяться и причинять серьезные ущербы. Компьютерные системы с устаревшими антивирусными базами становятся беспомощными перед новыми угрозами. Эвристические анализаторы, используемые для обнаружения неизвестных компьютерных вирусов, на текущий момент далеки от идеальных и часто либо ложно классифицируют чистые, неинфицированные файлы как вредоносные программы, либо не распознают зловредные программы.

Данная работа представляет собой обзор нейронных иммунных сетей (НИС), в которых сочетаются искусственные иммунные системы (ИИС) и самоорганизующиеся нейронные сети Т. Кохонена. В рамках исследования рассматривается один из видов самоорганизующихся нейронных сетей, применяемых в нейронных иммунных сетях, а также изучается взаимосвязь между биологической иммунной системой и искусственными иммунными системами. В дополнение к этому представлены алгоритмы, созданные для обнаружения вирусов при использовании ИИС.

1 Самоорганизующиеся нейронные сети Кохонена

Самоорганизующиеся нейронные сети (self-organising neural networks) характеризуются обучением без учителя, в результате которого происходит адаптация сети к решаемой задаче. Их разработал в 80-е гг. XX в. финский ученый Т. Кохонен. Нейронные сети Кохонена осуществляют топологическое упорядочивание входного пространства образов, поступающих на сеть. Они широко применяются в задачах распознавания и визуализации образов, оптимизации и управления.

1.1 Общая характеристика сетей Кохонена

Данные сети осуществляют отображение F входного n-мерного пространства образов в выходное m-мерное пространство, т. е. $F : R^n \to R^m$. При этом обучение здесь происходит без учителя на основе образов, поступающих на сеть. Если сеть осуществляет кластеризацию данных, то m характеризует количество кластеров, на которые разбивается входное пространство образов. Архитектура нейронной сети в общем случае представляет собой двуслойную нейронную сеть с прямыми связями (рис. 1).

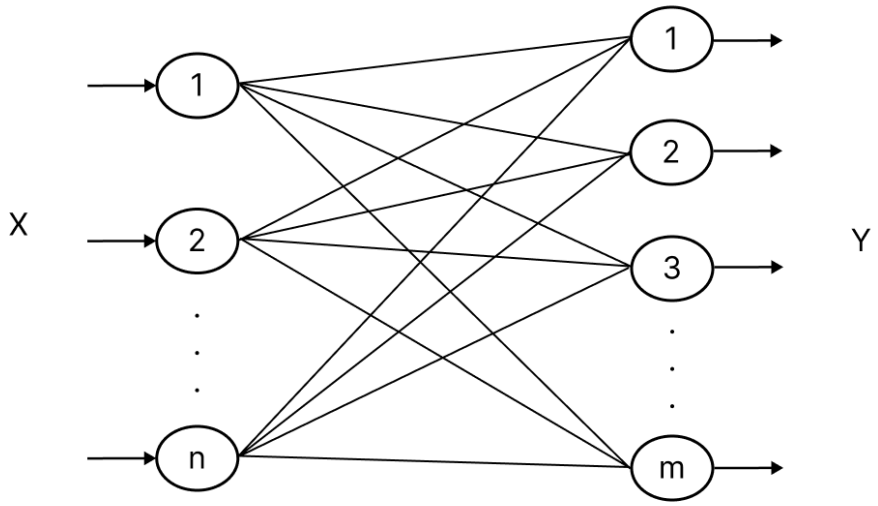

Рисунок 1 – Архитектура нейронной сети Кохонена

Первый слой выполняет чисто распределительные функции, причем каждый нейрон его имеет соединения со всеми нейронными элементами выходного слоя. Второй слой нейронных элементов является обрабатывающим.

Нейронная сеть Кохонена использует конкурентный принцип обучения и функционирования. В соответствии с этим принципом при подаче на сеть

входного образа значение только одного нейронного элемента выходного слоя принимается равным 1, а выходные значения остальных нейронов – 0. Нейронный элемент, имеющий выходное значение 1, называется победителем в конкурентной борьбе. По мере поступления входных образов на такую сеть посредством обучения происходит разбиение n-мерного входного пространства на различные области решений, каждой из которых соответствует отдельный нейрон обрабатывающего слоя [1].

Таким образом, самоорганизация таких сетей происходит в результате топологического упорядочивания входной информации по различным зонам, количество которых равно m. Такие зоны или области решений называются кластерами. Топологическое упорядочивание информации напоминает процессы, происходящие в головном мозге при его развитии, когда осуществляется формирование топологически упорядоченных нейронных структур [2].

1.2 Обучающийся векторный квантователь (LVQ)

Нейронная сеть для векторного квантования была предложена в 1982 г. Т. Кохоненом. Векторное квантование используется для сжатия данных и основано на идее сопоставления входного вектора с эталоном. Пусть имеется предварительно сформированное множество эталонных данных, каждое из которых называется кодовым вектором. Совокупность кодовых векторов называется кодовой книгой. При поступлении входного вектора происходит его сравнение с вектором из кодовой книги. В процессе этого выбирается такой кодовый вектор, который наилучшим образом аппроксимирует входной вектор и его номер применяется в качестве кода. В качестве меры подобия между входным и эталонными векторами может использоваться евклидово расстояние, а в качестве векторного квантователя – нейронная сеть Кохонена. Тогда целью обучения сети является такая настройка весовых коэффициентов нейрона, которая минимизирует погрешность аппроксимации между входными образами, создающими k-й кластер и весовыми коэффициентами k-го нейрона:

$$E_k = \frac{1}{2}\sum_{l=1}^{L_k}\sum_{i=1}^{n}(x_i^l - w_{ik})^2, \tag{1}$$

где E_k – ошибка квантования для k-го кластера.

Общая ошибка квантования определяется как:

$$E = \sum_{k=1}^{m} E_k. \tag{2}$$

Нейронную сеть для векторного квантования принято называть обучающимся векторным квантователем (learning vector quantization). Она представляет собой двуслойную сеть с прямыми связями, как было показано на рисунке 1. В процессе поступления эталонных векторов на сеть она обучается так, что образуются кластеры различных эталонов, каждому из которых соответствует свой нейрон. При поступлении на вход такой нейронной сети неизвестного образа он идентифицируется в соответствии с мерой близости к эталонным векторам и кодируется на выходе сети номером нейрона. Существует большое количество вариантов обучения векторного квантователя, однако в данной работе рассмотрены такие варианты, которые можно использовать в нейронных иммунных сетях [1].

1.2.1 Конкурентное обучение с одним победителем

Здесь в отдельный квант времени только один нейрон может быть победителем. Процедура обучения векторного квантователя состоит из следующих шагов:

1. Случайно инициализируются весовые коэффициенты нейронной сети в диапазоне [0,1].
2. Задается начальное значение момента времени $t = 0$.
3. Входные образы последовательно подаются на нейронную сеть $x^l, l = 1, L$, и для каждого образа производятся вычисления:
   а) вычисляется евклидово расстояние между входным образом и весовыми векторами нейронов выходного слоя:

$$D_j = |X - W_j| = \sqrt{(x_1 - w_{1j})^2 + (x_1 - w_{2j})^2 + \cdots + (x_n - w_{nj})^2}, \tag{3}$$

   где $j = \overline{1, m}$;
   б) определяется нейрон-победитель, обеспечивающий минимальное расстояние:

$$D_k = \min_j D_j;$$

в) производится модификация весовых коэффициентов нейронного элемента победителя, а весовые коэффициенты остальных нейронов не изменяются:

$$w_{ij}(t+1) = w_{ij}(t) + \gamma(t)(x_i - w_{ij}(t)), \text{ если } j = k,$$

$$w_{ij}(t+1) = w_{ij}(t), \text{ если } j \neq k,$$

где $i = \overline{1,n}, j = \overline{1,m}$.

4. Изменяется значение времени $t = t + 1$ и процесс повторяется, начиная с шага 3.

Обучение производится до получения желаемой степени согласования между входными и весовыми векторами или до тех пор, пока не перестанут изменяться весовые коэффициенты. Для остановки процесса обучения можно использовать следующие правила:

а) чтобы весовые коэффициенты в процессе обучения перестали изменяться, шаг обучения $\gamma(t)$ должен уменьшаться с течением времени, например, по следующему закону:

$$\gamma(t) = \gamma_0 e^{-\frac{t}{c}},$$

где $\gamma_0 = 0.1$; $c = 1000$;

б) обычно, как показывает практика, рекомендуется выбирать общее количество эпох обучения $t = (50 \div 200)$ [1].

Опираясь на описанные выше шаги, был написан небольшой код, который представляет собой реализацию нейронной сети для векторного квантования с конкурентным обучением с одним победителем (LVQ). Код программы представлен в приложении в конце данной работы.

Общая идея заключается в том, что сеть обучается на основе входных данных и их целевых значений, при этом происходит обновление весов нейронов так, чтобы они адаптировались к структуре данных и разделяли их на кластеры. В конечном итоге, сеть может использоваться для кластеризации новых данных.

1.2.2 Конкурентное обучение со многими победителями

Для того чтобы похожие кластеры отображались на соседние нейроны второго слоя, можно использовать конкурентное обучение со многими победителями. В этом случае вокруг нейрона-победителя формируются соседние ней-

роны, для которых также производится модификация весовых коэффициентов. Для этого вводится специальная функция притяжения, определяющая область притяжения нейрона-победителя в конкурентной борьбе. Значение функции притяжения для p-го нейронного элемента, не являющегося победителем в конкурентной борьбе, можно определить в соответствии с функцией Гаусса:

$$h(t, k, p) = e^{\frac{-|k-p|^2}{2\sigma^2(t)}},$$

где p – номер нейронного элемента, для которого определяется значение функции притяжения; k – номер нейронного элемента победителя; $\sigma(t)$ – среднеквадратичное отклонение (радиус области притяжения), уменьшающееся с течением времени по следующему закону:

$$\sigma(t) = \sigma_0 e^{-\frac{t}{c}}.$$

Значения переменных в последнем выражении можно выбирать следующим образом:

$$c = \frac{1000}{\log_2 \sigma_0},$$

где $\sigma_0 = m/2$.

Тогда модификация весовых коэффициентов производится для всех нейронных элементов обрабатывающего слоя в соответствии со значением функции притяжения:

$$w_{ip}(t+1) = w_{ip}(t) + \gamma h(t, k, p)(x_i - w_{ip}).$$

В дискретном варианте вводится область притяжения G, которая определяет нейронные элементы, принадлежащие классу победителей. В этом случае весовые коэффициенты изменяются для всех нейронов, принадлежащих области G:

$$\delta w_{ip} = \begin{cases} \gamma(t)(x_i - w_{ip}), \text{ если } p \in G, \\ 0, p \notin G. \end{cases}$$

1.2.3 Контролируемое конкурентное обучение

Если заранее известно соответствие эталонных векторов нейронным элементам, то используется контролируемое конкурентное обучение (Supervised Competitive Learning). Для такого обучения весовые коэффициенты нейрона-победителя усиливаются при корректной классификации, т. е. когда входной образ соответствует заданному номеру нейронного элемента второго слоя и ослабляются в противном случае. Тогда

$$w_{ik}(t+1) = w_{ik}(t) + \gamma(x_i - x_{ik}(t))$$

при корректной классификации, когда X и W_k принадлежат к одному классу и

$$w_{ik}(t+1) = w_{ik}(t) - \gamma(x_i - w_{ik}(t)),$$

если X и W_k принадлежат к различным классам.

Весовые коэффициенты остальных нейронов при этом не изменяются. После обучения нейронная сеть может осуществлять функции векторного квантования. При этом на выходе в каждый квант времени будет активным только один нейрон (его выход равен 1), а остальные нейроны будут иметь нулевые выходные значения.

2 Искусственные иммунные системы

2.1 Связь биологической иммунной системы с искусственными иммунными системами

Механизмы, использующиеся в искусственных иммунных системах, позволяют обнаруживать неизвестные компьютерные вирусы. Искусственные иммунные системы (ИИС) базируются на основных принципах биологической иммунной системы (БИС) [5].

БИС является уникальной системой, которая ежедневно борется с болезнетворными бактериями и вирусами, защищая организм от инфекций. Уникальность БИС заключается в том, что она способна обнаруживать не только известные вирусы и бактерии, но также и неизвестные. Иммунитет основан на способности лимфоцитов распознавать собственные клетки организма от чужеродных клеток.

Основными элементами иммунной системы являются лимфоциты – белые клетки. Существуют две разновидности лимфоцитов, которые образуются из стволовых клеток в костном мозге. После синтеза лимфоциты попадают в кровяное русло. Некоторые из них направляются к тимусу (вилочковой железе), где происходит их созревание (Т-лимфоциты). Другие же попадают в лимфатические узлы, и их созревание происходит там (В-лимфоциты). Процесс созревания незрелых лимфоцитов играет большую роль в иммунной системе и называется селекцией антител. В результате селекции уничтожаются нежелательные для организма лимфоциты. Зрелые лимфоциты имеют на своей поверхности детекторы, которые способны обнаруживать специфический антиген (вредные бактерии, вирусы). Контакт В-клеточных рецепторов со специфическим антигеном и связывание определенного его количества стимулируют рост этих клеток и последующее многократное деление. В результате образуются многочисленные клетки двух разновидностей: плазматические и «клетки памяти» [3]. Плазматические клетки синтезируют антитела, тем самым увеличивая количество клеток, способных обнаруживать вирус. Клетки памяти являются копиями В-клеток, однако имеют гораздо больший период жизни, что обеспечивает защиту организма от повторного заражения вирусом. При связывании определенного количества вируса, Т-клетки секретируют особую группу веществ, называемую лимфокинами. Некоторые лимфокины способны сами разрушать антиген и зараженные клетки. Другие лимфокины способствуют делению Т-клеток, в результате чего

появляется большое количество антител, способные реагировать на обнаруженный антиген [4].

2.2 Описание принципов функционирования ИИС

Биологическая иммунная система имеет ряд мощных вычислительных возможностей, такие как: распознавание, разнообразие, обучение, память, распределенный поиск, саморегуляция, децентрализация, вероятностное обнаружение.

Построенная по основным принципам биологической иммунной системы, искусственная иммунная система обладает всеми ее возможностями и, на наш взгляд, является перспективной для построения современной системы компьютерной безопасности для защиты от вредоносных программ. ИИС состоит из следующих процессов: создание детекторов, обучение и отбор детекторов, уничтожение нежелательных детекторов, циркуляция иммунных детекторов в компьютерной системе, уничтожение детекторов по истечении времени, обнаружение вредоносной программы, клонирование и мутация детекторов, формирование иммунной памяти. Взаимодействие процессов представлено на рисунке 2. Все перечисленные процессы находятся в тесном взаимодействии. Еще одной отличительной способностью ИИС является отсутствие единого центра управления.

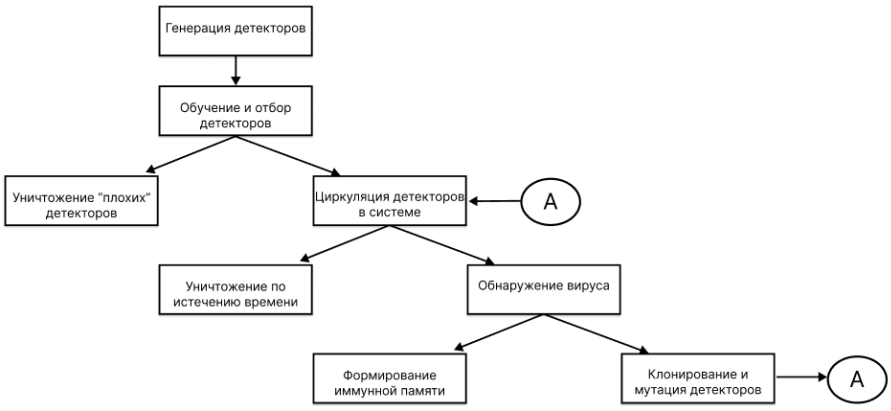

Рисунок 2 – Взаимодействие процессов искусственной иммунной системы

Рассмотрим подробнее каждый из перечисленных процессов. Процесс генерации детекторов предназначен для создания иммунных детекторов, которые являются основными элементами ИИС и выполняют функцию обнаружения вредоносных программ. Для построения иммунных детекторов используются

искусственные нейронные сети, а именно, LVQ сеть (обучающийся векторный квантователь). В процессе генерации формируется определенное количество детекторов, каждый из которых представляет отдельную нейронную сеть.

Первоначально детекторы не способны отличать чистые файлы от вредоносных программ. Поэтому необходим процесс обучения иммунных детекторов. На стадии обучения иммунные детекторы обучаются распознавать зловредные программы и не реагировать на чистые файлы. Обучение детекторов проходит по следующему алгоритму:

- случайным образом выбирается некоторое количество чистых файлов (например, утилиты операционной системы) и некоторое количество вредоносных программ;
- из выбранных файлов также случайным образом выбирается не сколько фрагментов определенной длины (размерность фрагментов зависит от количества входов искусственной нейронной сети, которая формирует детектор);
- выбранные фрагменты образуют обучающую выборку для ИНС и подаются на ее вход.

Использование разнообразных файлов и вредоносных программ для формирования обучающей выборки позволяет создавать разнообразные иммунные детекторы, способные обнаружить вероятные вредоносные программы.

После обучения детекторы проходят стадию отбора. Механизм отбора необходим для предотвращения попаданию в компьютерную систему нежелательных детекторов. Нежелательным детектором называется такой детектор, который реагирует на чистые файлы. Такой детектор должен быть уничтожен.

Обученные иммунные детекторы циркулируют в компьютерной системе, проверяя и классифицируя файлы. Каждому детектору отводится определенное время, на протяжении которого он может находиться в компьютерной системе. После истечения выделенного времени детектор, который не обнаружил вредоносной программы, уничтожается, а на его место приходит новый детектор. Механизм выделения времени для существования детектора и уничтожения по истечении выделенного времени позволяет ИИС избавляться от слабых иммунных детекторов и поддерживает принцип постоянного обновления детекторов.

При обнаружении иммунным детектором вредоносной программы проис-

ходит процесс клонирования. Клонирование подразумевает создание большого количества однотипных детекторов (клонируется тот детектор, который обнаружил вредоносную программу). Зачастую, при попадании компьютерного вируса в систему, он заражает большое количество файлов, путем внедрения копии своего тела в файлы-жертвы. Процесс клонирования позволяет иммунной системе в кратчайшие сроки избавиться от всех проявлений обнаруженного компьютерного вируса.

После избавления компьютерной системы от вредоносной программы выбирается наиболее приспособленный к обнаруженному вирусу детектор и трансформируется в детектор иммунной памяти. Иммунная память хранит информацию обо всех вирусах, которые когда-либо заражали компьютерную систему. Детекторы иммунной памяти существуют в компьютерной системе достаточно долгий промежуток времени и позволяют оперативно реагировать на повторное заражение.

3 Алгоритм функционирования нейросетевой иммунной системы

В данном разделе рассматриваются процессы генерации, обучения, отбора и функционирова ния иммунных детекторов на основе нейронных сетей. Здесь нейросетевой иммунный детектор (НИД) представлен в виде черного ящика, который имеет n-входов и два выхода (рис. 3).

Рисунок 3 – Нейросетевой иммунный детектор

Выходные значения детектора формируются после подачи всех образов на него в соответствии со следующим выражением:

$$Z_1 = \begin{cases} 1, \text{ если чистый файл} \\ 0, \text{ иначе.} \end{cases} \qquad Z_2 = \begin{cases} 1, \text{ если вирус} \\ 0, \text{ иначе.} \end{cases} \tag{4}$$

Обучающая выборка формируется из чистых файлов (класс чистых программ) и вредоносных программ (класс вредоносных программ). Желательно также иметь представителей всех типов вредоносных программ – черви, троянские программы, макровирусы и т.д. При обучении нейронной сети необходимо указать, где данные из чистых файлов, а где из вредоносных программ.

Пусть T – множество чистых файлов, а F – множество вредоносных файлов. Из них случайным образом формируется множество входных об разов для обучения i-го детектора.

$$X_i = \begin{bmatrix} X_i^1 \\ X_i^2 \\ \vdots \\ X_i^L \end{bmatrix} = \begin{bmatrix} X_{i1}^1 & X_{i2}^1 & \dots & X_{in}^1 \\ X_{i1}^2 & X_{i2}^2 & \dots & X_{in}^2 \\ & & \vdots & \\ X_{i1}^L & X_{i2}^L & \dots & X_{in}^L \end{bmatrix}, \tag{5}$$

где L – размерность обучающей выборки.

Соответственно, множество эталонных образов выглядят следующим образом:

$$l_i = \begin{bmatrix} l_i^1 \\ l_i^2 \\ \vdots \\ l_i^L \end{bmatrix} = \begin{bmatrix} l_{i1}^1 & l_{i2}^1 \\ l_{i1}^2 & l_{i2}^2 \\ \vdots & \\ l_{i1}^L & l_{i2}^L \end{bmatrix}, \tag{6}$$

Эталонные выходные значения для i-го детектора формируются следующим образом:

$$l_{i1}^k = \begin{cases} 1, \text{ если } X_i^k \in T \\ 0, \text{ иначе.} \end{cases} \quad l_{i2}^k = \begin{cases} 1, \text{ если } X_i^k \in F \\ 0, \text{ иначе.} \end{cases} \tag{7}$$

Обучение каждого детектора осуществляется с целью минимизации суммарной квадратичной ошибки детектора. Суммарная квадратичная ошибка i-го детектора определяется следующим образом:

$$E_i = \frac{1}{2} \sum_{k=1}^{L} \sum_{j=1}^{2} (Z_{ij}^k - l_{ij}^k)^2, \tag{8}$$

где Z_{ij}^k – значение j-го выхода i-го детектора при подаче на вход его k-то образа.

Общий алгоритм функционирования нейросетевой иммунной системы можно представить в виде следующей последовательности:

1. Генерация начальной популяции иммунных детекторов, каждый из которых представляет собой искусственную нейронную сеть со случай ными синаптическими связями:

   $$D = \{D_i, i = \overline{1, r}\}, \tag{9}$$

   где D_i - i-й нейросетевой иммунный детектор, r – общее количество детекторов.
2. Обучение сформированных иммунных нейросетевых детекторов. Обучающая выборка формируется случайньм образом из совокупности чистых файлов (как правило, это разнообразные системные утилиты опе рационной системы), и из совокупности вредоносных программ, или их сигнатур.

Эталонные выходные значения нейронной сети формируются согласно (7).

3. Отбор (селекция) нейросетевых иммунных детекторов на тестовой выборке. На данной итерации уничтожаются те детекторы, которые оказались неспособны к обучению, и детекторы, в работе которых наблюдаются различные недостатки (например, ложные срабатывания). Для этого каждый детектор проверяется на тестовой выборке. В результате для каждого детектора определяется значение квадратичной ошибки E_i (8).
   Селекция детектора производится следующим образом:

$$D_i = \begin{cases} 0, \text{ если } E_i \neq 0 \\ D_i, \text{ иначе.} \end{cases} \tag{10}$$

   где 0 обозначает операцию уничтожения детектора.
4. Каждый детектор наделяется временем жизни и случайным образом выбирает файл для сканирования из совокупности файлов, которые он не проверял.
5. Сканирование каждым детектором выбранного файла, в результате которого определяются выходные значения детекторов $Z_{i1}Z_{i2}, i = \overline{1,r}$.
6. Если i-й детектор не обнаружил вирус в сканируемом файле, т.е. $Z_{i1} = 1$ и $Z_{i2} = 0$, то он выбирает следующий файл для сканирования. Если время жизни i-го детектора закончилось, то он уничтожается и вместо него генерируется новый детектор.
7. Если i-й детектор обнаружил вирус в сканируемом файле, т.е. $Z_{i1} = 0$ и $Z_{i2} = 1$, то подается сигнал об обнаружении вредоносного файла и осуществляются операции клонирования и мутации соответствующего детектора. Операция мутации заключается в дополнительном обучении детекторов-клонов на обнаруженном вредоносном файле. В результате создается совокупность детекторов, настроенных на обнаруженную вредоносную программу

$$D_i = (D_{i1}, D_{i2}, \ldots, D_{in}).$$

8. Отбор клонированных детекторов, которые являются наиболее при способленными к обнаружению вредоносной программы. Если $E_{ij} < E_i$, то детектор прошел отбор. Здесь E_{ij} – суммарная квадратичная ошибка j-го

клона i-го детектора, которая вычисляется на вредоносном файле.

9. Детекторы-клоны осуществляют сканирование файлового пространства компьютерной системы до тех пор, пока не произойдет уничтожение всех проявлений вредоносной программы.
10. Формирование детекторов иммунной памяти. На этой итерации определяются нейросетевые иммунные детекторы, показавшие наилучшие результаты при обнаружении присутствующего в компьютерной системе вируса. Детекторы иммунной памяти находятся в системе достаточно длительное время и обеспечивают защиту от повторного заражения.

Особенностью предложенного алгоритма является то, что каждый нейросетевой иммунный детектор является полностью самостоятельным объектом. Он случайным образом выбирает файл из списка для его проверки. После проверки одного файла детектор переходит к следующему случайно выбранному файлу. При этом соблюдается принцип децентрализации системы безопасности, построенной на основе комбинации методов нейронных сетей и искусственных иммунных систем, что значительно повышает отказоустойчивость и защищенность системы в целом [4].

ЗАКЛЮЧЕНИЕ

Перспективы нейронных иммунных сетей в ближайшем будущем представляют собой захватывающий направленный прогресс в области кибербезопасности и искусственного интеллекта.

Нейронные иммунные сети, объединяя искусственные иммунные системы и нейронные сети, могут стать более эффективными в обнаружении неизвестных вирусов и вредоносных программ. Использование нейросетей для анализа поведения системы может улучшить способность выявлять новые угрозы, которые не имеют сигнатур в существующих базах данных.

Исследования в области НИС могут привести к созданию инновационных методов обнаружения, основанных на комбинации биологических принципов иммунной системы и высокоточных алгоритмов машинного обучения.

Таким образом, в ближайшем будущем нейронные иммунные сети могут играть важную роль в повышении уровня кибербезопасности, предоставляя более эффективные и адаптивные средства защиты от постоянно эволюционирующих киберугроз.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Головко, В. А. Нейросетевые технологии обработки данных : учеб. пособие / В. А. Головко, В. В. Краснопрошин // Минск : БГУ, 2017. - 263 с.

2 Головко, В. А. Нейронные сети: обучение, организация, применение // Нейрокомпьютеры и их применение : учеб, пособие. - М., 2001. - 256 с.

3 Безобразов, С. В. Искусственные иммунные системы для защиты информации: применение LVQ сети // Нейроинформатика-2007: материалы IX Всеросс. науч.-техн. конф., Москва, МИФИ, 2007. С. 27-35.

4 Безобразов, С. В. Искусственные иммунные системы для защиты информации: обнаружение и классификация компьютерных вирусов / С. В. Безобразов, В. А. Головко // Нейроинформатика-2008: материалы IX Всеросс. науч.-техн. конф., Москва, МИФИ, 2008. С. 23-27.

5 Безобразов, С. В. Искусственные иммунные системы для защиты информации: обнаружение и классификация компьютерных вирусов // Брест : БГТУ, 2008. - 10 с.

6 Безобразов, С. В. Искусственные иммунные системы: принципы построения [Электронный ресурс] : описание принципа построения искусственных имммунных систем. - URL: https://rep.bstu.by/bitstream/handle/data/1179/3-5.pdf?sequence=1 (дата обращения 11.01.2024). - Загл. с экрана. - Яз. рус.

7 Кушнир, Н. В. Искусственные иммунные системы: обзор и современное состояние [Электронный ресурс] / Н. В. Кушнир, А. В. Кушнир, Е. В. Анацкая, П. А. Катышева, К. Г. Устинов // Научные труды КубГТУ [Электронный ресурс] : [сайт]. - URL: https://ntk.kubstu.ru/data/mc/0019/0714.pdf (дата обращения 12.01.2024). - Загл. с экрана. - Яз. рус

ПРИЛОЖЕНИЕ А

Код программы lvq_realisation.py

```
import numpy as np

class LVQNetworkCompetitive:

    def __init__(self, input_size, output_size):
        self.input_size = input_size
        self.output_size = output_size
        self.ws = np.random.rand(output_size, input_size)

    def find_winner(self, input_vector):
        distances = np.linalg.norm(self.ws - input_vector, axis=1)
        winner_index = np.argmin(distances)
        return winner_index

    def update_weights(self, input_vector, learning_rate, winner_index, target):
            for i in range(self.input_size):
                self.ws[winner_index, i] += learning_rate * \
                                            (target - self.ws[winner_index, i]) * \
                                            input_vector[i]

    def compute_error(self, input_data, targets):
        total_error = 0.0
        for input_vector, target in zip(input_data, targets):
            winner_index = self.find_winner(input_vector)
            total_error += np.sum((input_vector - self.ws[winner_index]) ** 2) / 2

        return total_error

    def train(self, input_data, targets, epochs, initial_learning_rate):
        learning_rate = initial_learning_rate

        for epoch in range(epochs):
            for input_vector, target in zip(input_data, targets):
                winner_index = self.find_winner(input_vector)
                self.update_weights(input_vector, learning_rate, winner_index, target)

            training_error = self.compute_error(input_data, targets)
            print(f"Эпоха {epoch + 1}/{epochs}, Ошибка обучения: {training_error}")
            learning_rate *= np.exp(-epoch / 1000)
```

```
def predict(self, input_vector):
    winner_index = self.find_winner(input_vector)
    return winner_index
```

ПРИЛОЖЕНИЕ Б

Код программы main.py

```
from lvq_realisation import LVQNetworkCompetitive
import numpy as np
import json

def read_json_file(name) -> dict:
    try:
        f = open(name)
        data = json.load(f)
        f.close()
    except:
        print(f"Файла `{name}` не существует. "
              "Проверьте корректность имени файла.")
        exit(0)
    return data

def main():
    data_training = read_json_file("train_parameters.json")
    input_data = np.array(data_training["in"])
    targets = np.array(data_training["targets"])
    n = data_training["epochs"]
    learning_rate = data_training["learning_rate"]
    input_size = input_data.shape[1]
    output_size = 2 # по умолчанию два кластера на выходе
    lvq_net = LVQNetworkCompetitive(input_size, output_size)
    lvq_net.train(input_data, targets, n, learning_rate)
    while True:
        print('\n1. Проверить новые параметры.'
              '\n2. Выход.')
        bl = input("Выберите опцию: ")
        if bl == '1':
            new_data = read_json_file("new_parameters.json")["new_data"]
            for num, new_vector in enumerate(new_data):
                pred_cluster = lvq_net.predict(np.array(new_vector))
                print(f"Прогнозируемый кластер #{num}: {pred_cluster}")
        elif bl == '2':
            break

if __name__ == '__main__':
    main()
```

ПРИЛОЖЕНИЕ В

Файл train_parameters.json, содержащий параметры необходимые для обучения нейронной сети

```
{
    "in" :
    [
        [0.2, 0.8], [0.6, 0.4],
        [0.5, 0.7], [0.9, 0.3]
    ],

    "targets" :
    [
        0, 1, 1, 0
    ],

    "epochs" : 100,

    "learning_rate" : 0.2

}
```

ПРИЛОЖЕНИЕ Г

Файл new_parameters.json, содержащий новые данные для кластеризации.

```
{
    "new_data" :
    [
        [0.4, 0.7],
        [0.3, 0.5],
        [0.2, 0.8]
    ]
}
```