А.И. Мартышкин, А.А. Зоткина

ПОСТРОЕНИЕ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ ОБУЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ

Россия, г. Пенза, Пензенский государственный технологический университет

Currently, progress does not stand still and the industries are constantly developing. Education is one of them. New teaching methods and systems are emerging all the time, all for the sake of one goal - to educate people in society, to make it more intellectual, for the sake of people's development. The article discusses the introduction of distance learning services, the program of which will automatically adapt to a specific individual, increasing the effectiveness of studying the course at times. For effective and efficient work, the technology of artificial neural networks is used, with the possibility of dynamic learning. A feed-forward network is selected. The neural network is trained using the error backpropagation method. The prototype proposed in this article can serve as a starting point for the creation of an improved version of distance learning that can develop the speed of understanding and learning of the courses of the learners.

Введение. В настоящее время, в период распространения коронавирусной инфекции, особенно актуальным является дистанционный формат обучения. На данный момент существует довольно много сервисов для дистанционного обучения какому-либо навыку. Они различаются многими параметрами – некоторые сервисы требуют плату, другие сервисы просто предоставляют записи лекций и выступлений преподавателей в университетах и школах, также есть сервисы, которые используют онлайн обучение вживую с учителем, который позволяет подстраивать изначально заданную программу под индивидуального обучающегося [1]. Но такие системы требуют больших ресурсов, в том числе и человеческих. Например, необходим педагог с определенными знаниями в рамках курса. Но всегда специалист в той или иной сфере сможет вам оперативно ответить на все вопросы и разъяснить материал. Поэтому имеется необходимость создания сервисов дистанционного обучения, программа которых будет автоматически подстраиваться под конкретного индивидуума, увеличивая эффективность изучения курса в разы.

Одним из способов оптимизировать процесс проверки знаний является тестирование [2]. Тестология, как теоретическое и практическое учение, существует уже более 120 лет. За этот немалый промежуток времени был накоплен большой опыт использования тестирования в различных сферах общества, в том числе и образования. Проанализировав теорию тестологии как науки, можно сделать выводы, что для создания адаптивной системы тестирования, необходим определенный банк вопросов, каждый из которых будет уже испытан на определенном круге обучения. Необходимо, чтобы все вопросы в этом банке были правильно составлены и отвечали описанным выше критериям. Также необходимо учитывать идеальный пропорции заданий различной меры трудности в тестировании для каждого индивидуального ученика.

Анализ и выбор имплементации. **Реализация прототипа**. Для создания такой системы необходимо выбрать алгоритм, который позволяет изменяться в зависимости от входных параметров, что будет позволять подстраивать содержание курсов в зависимости от параметров обучающегося и темы курса. Прототип должен реализовывать механизм изменения тестов, чтобы они носили более индивидуальный характер:

- система должна подстраивать последующие прохождения тестирования (как повторная попытка прохождения текущего, так и последующие в выбранном курсе),

предоставляя больше вопросов по проблемной тематике;

- система должна иметь возможность возвращаться в некоторое исходное состояние, уменьшая количество проблемных вопросов по мере того, как обучающийся будет совершать меньше ошибок в ней.

Прототип, создаваемой системы, должен четко описывать свои интерфейсы, какие именно параметры должны подаваться на их вход и какой ответ будет ожидаться. Такой подход позволит легко встроить прототип в уже работающую систему, не прибегая к глобальным изменениям архитектуры [3].

Одним из примеров такого алгоритма является технология искусственных нейронных сетей, одной из особенностей которой является возможность динамического обучения [4, 5]. Данный алгоритм выбран для имплементации решения.

Для достижения нашей цели сначала необходимо выбрать модель ученика. Она должна предоставлять исходные характеристики обучающегося, чтобы сделать изначальную подстройку курса под индивидуума. В качестве модели ученика будет выбрана как определенная смесь скалярной и стереотипной моделей.

Характеристики обучающегося будут представлены в виде скалярных величин. Данные характеристики будут вычисляться с помощью начального тестирования, но при этом будут иметь возможность изменяться в течение времени. Также характеристики должны будут классифицироваться, имея возможность предоставлять различные значения, которые будут соотноситься с темой курса. Это необходимо, потому что зачастую ученик силен в одной области, но слаб в другой. Такая классификация поможет бороться с этим, поэтому модель ученика как индивидуума будет различаться от курса к курсу, позволяя более оптимально его адаптировать.

Стереотипная же часть модели будет отвечать за изначальное значение весов в системе. Благодаря этому, нейронная сеть может быть в одном из предустановленных положений в начале обучения ученика и адаптироваться в процессе, в зависимости от индивидуальных результатов. Изначальные разные положения необходимы, потому что, если иметь только одно изначальное состояние нейронной сети, то будет возникать проблема, что первые тестирования для определенного ученика не будут адаптироваться вообще, приводя к неудовлетворительному результату.

Характеристиками для ученика будут:

- количество верных вопросов на заданную тему;
- количество неверных вопросов на заданную тему;
- время прохождения теста;
- количество пропущенных вопросов;
- прошлый посчитанный коэффициент.

После выбора модели ученика, необходимо определиться с архитектурой будущей нейронной сети, главной целью которой будет управление, то есть используя на входе характеристики ученика для выбранной темы и непосредственно тему, будет выдаваться определенный вектор, который впоследствии будет влиять на последующий курс.

Одной из простых архитектур, которая имеет функцию управления является сеть прямого распространения [5]. Для более точной классификации при работе сети, будет выбрана многослойная сеть прямого распространения. Она состоит из входящего, скрытого и выходящего слоя (рис.1).

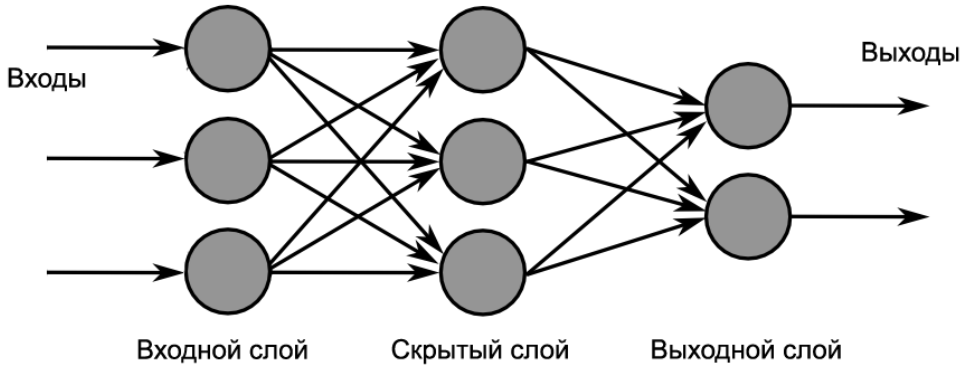

Рисунок 1 – Сеть прямого распространения

Данная архитектура удовлетворяет требованиям системы и достаточно проста в реализации. Можно использовать и другие архитектуры для решения поставленной задачи, но более сложная архитектура ведет к более ресурсоемкому процессу работы системы, поэтому с точки зрения быстродействия это невыгодно.

Первый скрытый слой будет отвечать за преобразование входных данных в определенный коэффициент, второй же отвечать за сужение результата. Будет использоваться три нейронных сети, которые необходимо подстроить под изначальные прототипы – сильный, обычный и слабый студент. Это позволит сделать определенную дифференциацию между изначальным множество учеников.

После выбора архитектуры сети, в качестве инструмента для реализации разработки программного обеспечения, использующего машинное обучение, был выбран фреймворк компании Google – TensorFlow. Одним из главным преимуществ использования данного фреймворка является открытый код, и то, что обучение происходит, используя распределенное обучение, что позволяет значительно ускорить процесс. Фреймворк основан на языке программирования Python, способен использовать как CPU, так и GPU для расчета весов в нейронных сетях.

Прототип будет использовать технологию REST, которая позволяет системе предоставлять интерфейсы для взаимодействия. На вход интерфейса должны будут передаваться параметры, необходимые для функционирования системы, в ответе же должен быть вектор направления курса обучения, который будет вычислен нейросетью [6, 7, 8].

Основная схема работы приложения планируется следующая – на вход сторонняя система подает параметры об ученике, его ответы на вопросы в тесте, вес ответов, максимально возможный результат в пройденном тестировании, а также тема тестирования. Система будет обрабатывать эти данный и выдавать ответ в виде коэффициента развития студента. Помимо этого, система будет хранить данные о студентах в локальном хранилище для возможности использовать их при дальнейшем развитии системы. Это нужно для более точного коэффициента развития в дальнейшем. Архитектура передачи данных между сторонней системой и приложением будет REST. Все интерфейсы будут использовать HTTP протокол с POST-методом доступа. Необходимы параметры будут передаваться в теле запроса в формате JSON.

Первый интерфейс должен быть использован после прохождения студентом инициализирующего тестирования на выбранную тематику курса. Для создания прототипа будет реализовано всего две группы – сильные ученики и слабые. Но, естественно, в дальнейшем данные группы можно будет расширить. Стереотипные группы отвечают за изначальный выбор нейронной сети, которая будет использоваться для расчета коэффициента.

Данный интерфейс будет принимать следующие параметры:

- answer[] – массив объектов типа Answer. Сам объект включает в себя следующие параметры: score (числовой параметр, балл студента за вопрос, если обучающийся пропустил вопрос – то передается -1), weight(числовой параметр, вес вопроса), timeToAnswer (время, за которое ученик ответил на вопрос). Если на вопрос не ответили из- за истечения времени, то timeToAnswer = 0 и score = -1;
- theme – тема тестирования;
- studentId – идентификатор ученика.

Выходным параметром будет являться числовой параметр, указывающий на стереотипную группу. В прототипе это будет 1 или 2.

Вторым интерфейсом будет являться /calculateRatio. Интерфейс будет осуществлять главный функционал программы – определять коэффициент развития. Входными параметрами данного интерфейса будут являться:

- answer[] – такой же массив вопросов, как и в интерфейсе выше;
- studentId – идентификатор ученика;
- theme – тема тестирования;
- studentRatio – коэффициент студента;
- studentCategory – стереотип обучающегося.

Выходным параметром данного интерфейса будет число, означающее коэффициент развития студента.

Также создан интерфейс для вывода статистики об обучающемся. Он будет принимать два параметра – studentId и theme. Выходным параметром же будет массив объектов, каждый из которых включает в себя время сбора статистики (time) и коэффициент на то время (studentRatio). Так можно будет видеть, как он менялся с течением времени.

В каждом из представленных интерфейсах будет осуществляться функция валидации входных параметров. Если были введены неверные параметры, которые не соответствуют шаблону, то будет возвращен errorMessage с указанием, на каком параметре произошла ошибка валидации.

Хранение данных в приложении будет осуществляться с помощью базы данных SQLite. Данная база данных не требует больших ресурсов, поэтому часто применяется с приложениями, реализующими структуру микросервисов [7].

Создав архитектуру приложения на верхнем уровне, необходимо приступить к главному – настройке нейронной сети. Входными параметрами в данной сети будут:

- процент верных ответов ученика в рамках данной тематики на тестировании. Причем будут учитываться вес вопросов, если он есть;
- процент времени потраченное на тестирование;
- прошлый коэффициент развития обучающегося;
- коэффициент темы.

Выходным нейроном будет коэффициент из одного узла – коэффициент развития.

Всего будет сделано 2 нейронных сети для прототипа – они будут иметь одинаковую структуру, но разный набор обучения. Это необходимо, потому что ученики имеют различные предрасположенности к различным темам. В процессе инициализирующего тестирования по теме для ученика будет выбран определенный стереотип, в зависимости от которого будет выбираться используемая нейронная сеть.

Обучение нейронной сети происходит с помощью метода обратного распространения ошибки. В данном методе обучения, искусственная нейронная сеть имеет два набора векторов – входной и выходной и подстраивает весы так, чтобы выходной реальный параметр соответствовал верному. В качестве обучающегося множества был взят Data set с UCI Machine Learning Repository. Сайт содержит

множество обучающих пар входных данных на различную тематику. Датасет состоит из 157 пар, но при этом они были доработаны вручную – чтобы был учет входных параметров. Также входные данные были дублированы с изменением выходного коэффициента – для обучения второй сети.

В искусственной нейронной сети используется два скрытых слоя. Первый слой отвечает за непосредственно преобразование параметров в финальный коэффициент. Второй слой будет иметь сжимающей функцию – необходимо, чтобы коэффициент была в интервале между 0 и 1, а также чтобы он был достаточно плавно распределен.

Функция активации на первом скрытом слое - сигмоида. Она необходима для обеспечения основного функционала нейронной сети.

В качестве сжимающей функции выступает логистическая функция.

Общая ее формула приведена ниже:

$out(x) = 1/1 + \exp(-a \cdot x)$

Коэффициент альфа при этом отвечает за крутизну изменения функции. Для прототипа этот коэффициент будет равен 0,2, позволяя сделать не слишком крутой переход.

Итоги. В результате, в качестве проверки была создана простая система тестирования с 4 тестами. Было выбрано 10 человек для участия в тестировании. Вопросы в тестировании охватывали 2 темы, знание языка Java и знание паттернов программирования. Каждый из участников проверки приложения сначала проходил инициализирующее тестирование. Это тестирование было построено из 20 вопросов, по 10 на каждую из тем. Результатом тестирования стало то, что все участники были разбиты на две стереотипные группы: в рамках первой темы было 7 сильных участников и 3 слабых, в рамках второй – 6 и 4, соответственно.

Для трех последующих этапов тестирования были созданы банки вопросов/ответов для тестов. Всего было создано по 40 вопросов на каждую тематику и каждой из пар был проставлен коэффициент сложности вопроса. Тест составлялся используя коэффициент развития – более сильным ученикам с высоким коэффициентом давались более сложные вопросы, более слабым давалось больше легких. Для второго и третьего тестирования банк вопросов был больше – было собрано 100 пар вопросов и ответов. Это обусловлено тем, что коэффициент развития после первого тестирования будет более индивидуальным и необходимо более точно подстраивать содержание тестирования.

После прохождения первого тестирования, у участников изменились коэффициенты развития и было запущено более индивидуальное второе тестирование. После этого процесс повторился и было запущено финальное тестирование.

По результатам была сделана сводная статистика тестирования и были опрошены участники. Участникам, в целом, понравилась идея, но в короткой перспективе они не заметили особой разницы между обычным тестированием и адаптированным. Конечная статистика участников представлена в таблицах 1 и 2.

Таблица 1 – Статистика по теме знания языка Java

Группа	Средний КР после первого тестирования	Средний КР после второго тестирования	Средний КР после третьего тестирования
Сильная	0,83	0,86	0,88
Слабая	0,68	0,72	0,72

Таблица 2 – Статистика по теме знания паттернов программирования

Группа	Средний КР после первого тестирования	Средний КР после второго тестирования	Средний КР после третьего тестирования
Сильная	0,77	0,79	0,83
Слабая	0,55	0,60	0,63

Исходя из таблиц 1-2, можно легко заметить, что прототип имеет возможность предоставлять коэффициенты для последующего подстраивания курса под индивидуума. То есть главную поставленную цель прототип выполняет.

Таким образом, предлагаемый в статье прототип может послужить отправной точкой для создания улучшенной версии дистанционного обучения, которая сможет развить скорость понимания и обучения курсов обучающихся.

1. Дмитриев В.С., Зоткина А.А. Интеллектуальное поведение в машинах. искусственный интеллект и машинное обучение // Инновационное развитие науки и образования: сборник статей VI Международной научно-практической конференции. В 2 ч. Ответственный редактор: Гуляев Герман Юрьевич. 2019. – С. 126-128.
2. Кононов А.И. Об опыте использования системы подготовки тестовых заданий и обработки результатов вступительных испытаний // Развитие системы тестирования в России: Тез. докл. Всерос. конф. – М., 1999. – Ч. 2.
3. Мартышкин А.И., Перекусихина А.Н., Зоткина А.А. Исследование групп пользователей в социальных сетях по их интересам и поведению на основе множества источников данных // XXI век: итоги прошлого и проблемы настоящего плюс. – 2020. – Т. 9. – № 4 (52). – С. 30-35.
4. История возникновения нейронных сетей [Электронный ресурс] Режим доступа: https://neuronus.com/history/5-istoriya-nejronnykh-setej.html своб. (дата обращения: 10.04.2021)
5. Уоссермен Ф. Нейрокомпьютерная техника: теория и практика. Перевод на русский язык Ю. А. Зуев, В. А. Точенов, 1992 – 184 с.
6. The Neural Network Zoo [Электронный ресурс] Режим доступа: http://www.asimovinstitute.org/neural-network-zoo/ свободный. Язык англ. (дата обращения: 08.04.2021).
7. Круглов В.В., Дли М.И., Голунов Р.Ю. Нечёткая логика и искусственные нейронные сети. Физматлит, 2001 – 224 с.
8. Мартышкин А.И., Лысцов Н.А., Зоткина А.А. Перспективы применения нейронных сетей // Фундаментальные и прикладные научные исследования: актуальные вопросы, достижения и инновации: сборник статей XXII Международной научно-практической конференции. 2019. – С. 52-54.