НАПИСАНИЕ НЕЙРОННОЙ СЕТИ ДЛЯ ТОРГОВОГО РОБОТА

Ралко Андрей Аркадьевич

магистрант, Кокшетауский государственный университет им. Шокана Уалиханова, Республика Казахстан, г. Кокшетау

Костангельдинова Алма Акжановна

канд. пед. наук, зав. кафедры Информатики и МП, Кокшетауский государственный университет им. Шокана Уалиханова, Республика Казахстан, г. Кокшетау

Аннотация. нейронные сети — огромный пласт, который является фактически мультидисциплинарным. Он объединяет математиков, которые подготавливают так называемую теоретическую основу программы. Программистов, создающих ее «внутренность», и всех тех, ради кого программа и создается: физиков, экономистов, финансистов, медиков и других специалистов, которые уже пользуются ею и, в свою очередь, предъявляют к ней свои требования, а также в какой-то степени ее дополняют.

Ключевые слова: нейронные сети, торговый робот, трейдинг, структура нейронной сети.

Сегодня нейронные сети применяются для решения широко спектра задач. Одной из сфер все более активного применения нейронных сетей является авто-трейдинг. Важно отметить, что при разработке нейронной сети для торгового робота программисты сталкиваются с определенным перечнем трудностей, обусловленных сферой применения нейронной сети.

В данной статье мы на опыте апробировали один из возможных вариантов решения данных трудностей.

В чем же заключается сложность применения нейронных сетей в трейдинге? При разработке любой нейронной сети, на первом этапе разработчиком определяется структура сети.

Структура сети включает в себя такие параметры как количество слоев в сети, а также количество нейронов в слоях, особенно важны параметры количества нейронов во входных и выходных слоях, так как эти показатели во многом определяют результаты работы сети.

Проблема разработки нейронной сети для торгового робота заключается в том, что на поздних этапах разработки робота может многократно возникать необходимость внесения изменений в структуру самой нейронной сети, что в свою очередь вызывает необходимость в переписывании больших сегментов кода, отвечающих за функционирование сети.

Одним из вариантов решения данной проблемы является программирование нейронной сети с возможностью изменения ее структуры без серьезных изменений в коде программы.

Реализовать данную концепцию мы попытались следующим образом: было создано два класса, первый класс «Neuron» включает в себя, те методы и свойства которые необходимы отдельно взятому нейрону в сети, вторым классом определили класс самой нейронной сети «TNeuronNet».

Данные необходимые для генерации и функционирования содержат поля класса.

private:

float Vesa[100][100][100];

Neuron Sloi[100][100];

int Col_vo_neuron_v_sloe[100];

Как видно из данного примера этот класс, включает в себя три поля, которые являются массивами разной размерности. «Col_vo_neuron_v_sloe» это массив хранящий количество нейронов в каждом слое, нулевой элемент этого массива является количеством нейронов во входном слое. «Sloi» массив хранящий выходные данные нейронов сети после их вычисления , «Vesa» массив, хранящий веса нейронов.

Размерность всех массивов ограничена 100 элементами, следовательно, максимально возможная размерность нейронной сети ограничена слоями и нейронами по 100 в каждом слое.

Данная размерность сети вполне достаточна для реализации успешного торгового робота, и не ведет к чрезмерному «распуханию» сети.

Все поля класса определены как «private» для того чтобы исключить опасность изменения данных нейронной сети вне класса.

Функционал сети разделен между шестью методами.

public:

void GenerateNet(int Col_vo_sloev) {…};

void Input_Data(){…};

void ForvardNet(int Col_vo_sloev){…};

void BackNet(int Need_result, int Col_vo_sloev){…};

float Sum(int sloi,int neuron_nom){…};

float Activarion(float output){…};

Метод «GenerateNet» генерирует первоначальное состояние сети, в частности генерирует исходные веса нейронов. Метод «Input_Data» подает исходные данные на входные нейроны сети. «ForvardNet» это метод прямого распространения ошибки, в нем исходные данные проходят через все слои нейронной сети, вычисление выходного значения каждого нейрона происходит в методах «Sum» и «Activarion».

Метод «Activarion» содержит функцию активации нейрона, в данной сети используется сигмоидальная функция активации.

В методе «Sum» вычисляется сумма произведений выходных значений нейронов с соответствующими весами, для каждого нейрона.

«BackNet» это процедура обратного распространения ошибки, в данном методе находится ошибка каждого нейрона, и переопределяются веса нейронов.

«GenerateNet» имеет один входной параметр, это количество слоев в сети, в ходе выполнения процедуры пользователь также задает число нейроном в каждом слое.

Именно этот механизм позволяет создавать сеть любой размерности.

Алгоритм каждого метода работоспособен при любой размерности сети, это реализуется за счет использования в качестве конечных параметров циклов переменных, определяемых в ходе генерации сети, вместо дефолтных значений.

Завершающим элементом алгоритма является класс конструктор в качестве родителя, которого выступает класс «TNeuronNet».

```
TNeuronNet::TNeuronNet(int Col_vo_sloev){

for (int i = 0; i < Col_vo_sloev; i++){

std::cin >> Col_vo_neuron_v_sloe[i];//кол-во нейронов в каждом слое  }

GenerateNet(Col_vo_sloev);//генерация сети

Input_Data();//ввод данных

ForvardNet(Col_vo_sloev);//прямое распространения

BackNet(1,Col_vo_sloev);//обратное распространения

}
```

Данный класс вызывается в основной части кода торгового робота.

Программирование нейронной сети произвольной размерности однозначно более трудоемкий процесс, нежели программирование сети постоянной размерности.

Но все трудозатраты окупаются возможностями, которые получает разработчик в дальнейшем.

Если алгоритм нейронной сети реализован подобным образом, то для изменения структуры сети достаточно изменить всего лишь одну строчку кода. Также учитывая, что сеть реализована в виде класса, мы получаем возможность ее многократного использования при разработке программ, в которых применяются нейронные сети.

Список литературы:

1. Трофимова Е. А., Мазуров В. Д., Гилёв Д. В. Нейронные сети прикладной экономике // Издательство Уральского университета, 2017. 98 с.