УДК 004.9:004.032.26

ПРОГНОЗИРОВАНИЕ РЕЗУЛЬТАТОВ СОРЕВНОВАНИЙ ПО КИБЕРСПОРТУ С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

С.Г. Минчук
Магистрант кафедры информатики БГУИР, инженер-программист

А.В. Жвакина
Доцент кафедры информатики БГУИР, кандидат технических наук, доцент

Белорусский государственный университет информатики и радиоэлектроники, Республика Беларусь
Иностранное производственное унитарное предприятие "ИССОФТ СОЛЮШЕНЗ", Республика Беларусь
E-mail: minchuk94@gmail.com, zhvakina@bsuir.by

Аннотация. В докладе рассматривается возможность использования нейронных сетей для прогнозирования результатов соревнований по киберспорту. Оценивается решение задачи прогнозирования с использованием следующих видов нейронных сетей: сети прямого распространения, сети Хопфилда и реккурентной нейронной сети LSTM.

Ключевые слова: прогнозирование, киберспорт, нейронные сети, реккурентная нейронная сеть LSTM.

Возрастающая популярность киберспорта, большое количество соревнований по всему миру, в том числе международных, рост призового фонда, а также признание нескольких киберспортивных дисциплин настоящим видом спорта во многих странах заставляют относиться к данным мероприятиям не как обычным развлечениям.

Существует целая армия аналитиков киберигр, которая обрабатывает результаты соревнований. Этот процесс связан с большим количеством информации, достаточно трудоемок и не автоматизирован, поэтому довольно затруднительно спрогнозировать результат конкретной игры. Кроме того, отличительной чертой киберспорта является то, что он основывается на использовании интеллектуальных способностей участников, а не фактора случайности, как в обычных играх. Таким образом, возникает необходимость обучения как аналитиков, так и участников киберигр. Также необходимо развитие стратегии игры, разработка новых вариантов более интересных и сложных.

В настоящее время для прогнозирования игр используются следующие алгоритмы [1].

Случайный лес – это алгоритм машинного обучения, заключающийся в использовании большого множества деревьев решений [3]. Все деревья строятся независимо по следующей схеме:

8 Выделяется подвыборка обучающей выборки размера N и по ней строится дерево.

9 Для построения каждого нового расщепления в дереве просматриваем m случайных признаков.

10 Выбираем наилучший признак и расщепление по нему (по заранее заданному критерию). Дерево строится, до исчерпания выборки.

При составлении прогнозов используется большинство голосов всех деревьев модели.

Достоинства алгоритма случайного леса:

– способность эффективно обрабатывать данные с большим числом признаков и

классов;

– легкое масштабирование;
– внутренняя оценка способности модели к обобщению;
– высокая параллелизуемость.

Недостатки алгоритма случайного леса:

– большой размер получающихся моделей. Требуется O(K){\displaystyle O(K)} памяти для хранения модели, где {\displaystyle K}K − число деревьев[4].

Наивный байесовский классификатор

Байесовский классификатор [5] - это классификатор, основанный на использовании теоремы Байеса для определения вероятности принадлежности элемента выборки к одному из классов C при условии того, что зависимые переменные принимают заданные значения: $P(C|F_1,...,F_n)$. Иными словами, если на основе значений переменных можно однозначно определить, какому классу относится наблюдение, байесовский классификатор присвоит этому элементу вероятность равную 1 того, что этот элемент относится к данному классу. Если же однозначно не определить к какому классу принадлежит элемент, то классификатор просто вернет вектор вероятностей для каждого класса.

Преимущества наивного байесовского классификатора:

– простота реализации;
– малые вычислительные затраты при обучении и для классификации.

Недостатки наивного байесовского классификатора:

– низкое качество классификации.

Метод опорных векторов

Основная идея метода − перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости, где расстояние между классами, на которые делит эта гиперплоскость, максимально. Пусть некоторые объекты принадлежат к одному из двух классов. Задача состоит в том, чтобы определить, к какому классу будут принадлежать новые наблюдения.

Преимущества метода опорных векторов [6]:

– быстрый метод нахождения решающих функций;
– метод сводится к решению задачи квадратичного программирования в выпуклой области, которая всегда имеет единственное решение.

Недостатки метода опорных векторов [6]:

– метод чувствителен к шумам;
– требователен к стандартизации данных.

Анализ рассмотренных алгоритмов показывает, что имеющиеся недостатки не позволяют их эффективно применить для прогнозирования результатов киберигр.

В работе оценивается решение задачи прогнозирования с использованием следующих видов нейронных сетей: сети прямого распространения, сети Хопфилда и реккурентной нейронной сети LSTM.

При использовании сети прямого распространения и сети Хопфилда применялся метод декомпозиции процесса киберигры на множество элементарных ситуаций и их анализ, результаты которого использовались для прогнозирования ситуации в целом.

Модель функционирования киберигры построена в данном случае на основе представления в классе марковских случайных процессов. Так как рассматриваемые задачи являются динамическими, то они описывались системой дифференциальных уравнений (СДУ) Чепмена – Колмогорова. Основным показателем эффективности выбрано математическое ожидание потерь сторон. В качестве примера рассматривалась ситуация, описываемая системой шести дифференциальных уравнений.

За показатели эффективности синтезированных нейронных сетей взяты точность и

время решения СДУ. Точность решения СДУ с помощью нейронной сети определялась как средняя относительная суммарная ошибка интегрирования:

$$\Delta = \frac{\sum_{i=1}^{m} \Delta P(h,t)}{n \cdot m} \cdot 100\ \%, \tag{1}$$

где: n – количество определяемых вероятностей; m – количество измерений; $\Delta P(h,t)$ – ошибка интегрирования СДУ в целом при фиксированном шаге h, определяемая по формуле

$$\Delta P(h,t) = \sum_{i=1}^{n} \left| \Delta P_i(h,t) \right|, \tag{2}$$

где: $\Delta P_l(h,t)$ – ошибка интегрирования для вероятности P_l, вычисляемая в соответствии с выражением

$$\Delta P_l(h,t) = \left| P_l^T(h,t) - P_l^{HC}(h,t) \right|, \tag{3}$$

где: $P_l^T(h,t)$ – точное значение вероятности P_l; $P_l^{HC}(h,t)$ – значение вероятности P_l, полученное на нейросети для шага h и времени t.

В качестве критерия эффективности нейросетевого алгоритма по времени решения поставленной задачи принято отношение времени t_{MC} решения системы дифференциальных уравнений методом Рунге – Кутта с использованием пакета MathCAD ко времени t_{HC} решения системы дифференциальных уравнений на синтезированной нейронной сети $E = \frac{t_{MC}}{t_{HC}}$.

Размер шага интегрирования h выбирался из интервала $h = (0{,}01 \div 0{,}5)\rho$, где ρ – коэффициент, характеризующий степень «загрузки» входным потоком [7].

Из полученных в случае решения задачи моделирования на базе сети Хопфилда результатов следует, что для обеспечения точности $\Delta < 7\ \%$ решения системы дифференциальных уравнений необходимо, чтобы сеть содержала не менее 13 нейронов, а шаг дискретизации находился в диапазоне $h \le 0{,}1 - 0{,}15$. При этом параметр ρ также влияет на величину Δ.

Опираясь на результаты исследований, можно сделать вывод, что для обеспечения точности $\Delta < 3\ \%$ решения СДУ необходимо, чтобы сеть прямого распространения содержала во входном слое не менее 35 нейронов, а шаг дискретизации находился в диапазоне $h \le 0{,}1 - 0{,}15$. При этом время обучения t при использовании генетического алгоритма составит около 3 мин.

Сравнительной оценка относительной эффективности E по времени и точности решения СДУ на синтезированных НС показала, что значение E в среднем равно: для НС Хопфилда – $5 \cdot 10^5$; для НС прямого распространения – $4 \cdot 10^{-2}$. Выяснено, что наиболее быстро решение можно получить, используя сеть Хопфилда. Больше всего времени необходимо при интегрировании на сети прямого распространения. Это объясняется тем, что время обучения сети измеряется минутами. Процедуру обучения требуется проводить каждый раз при использовании новых исходных данных. Устранить подобный недостаток можно путем представления процесса обучения как самостоятельной нейронной сети.

Сравнительная оценка относительной ошибки интегрирования Δ в зависимости от выбранного нейросетевого способа показала, что в среднем она равна: для НС Хопфилда – 7 %; для НС прямого распространения – 3 %. Таким образом, наименьшей точностью обладает сеть

Хопфилда.

Необходимо отметить, что все сложности, имеющиеся при решении систем дифференциальных уравнений на обычных ЭВМ, встречаются и при их решении на формируемых нейронных сетях. В первую очередь это относится к понятию устойчивости решения.

В тоже время прогнозирование направлено на определение тенденций развития конкретного объекта или события на основе предыдущих данных, иными словами, анализа его состояния в прошлом и настоящем [7]. Результат игры базируется не только на данных в определенный момент, а также данных, собранных на протяжении всей игры до этого момента. Рекуррентные нейронные сети − это сети, основанные на использовании предыдущих состояний для вычисления текущего, поэтому необходимо рассмотреть возможность использования для прогнозирования рекуррентной сети LSTM, которая способна к обучению долговременным зависимостям.

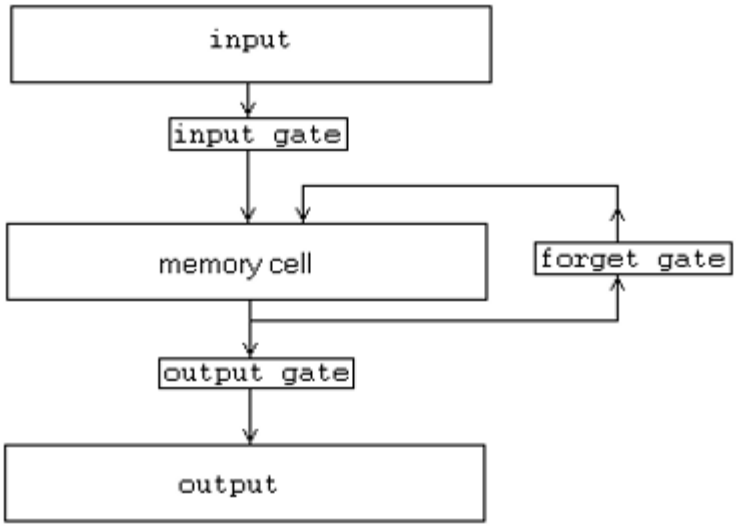

Рисунок 1. Схема нейронной сети LSTM

LSTM состоит из следующих частей [2] (рис.1):

- вход (*input*) нейронной сети;
- выход (*output*) нейронной сети;
- внутреннее состояние нейронной сети или запоминающая ячейка (*memory cell*);
- фильтр очистки памяти или фильтр забывания (*forget gate*);
- входной фильтр или фильтр обновления памяти (*input gate*);
- выходной фильтр или фильтр выдачи результата (*output gate*).

Рассмотрим подробнее структуру рекуррентной сети LSTM [2]. Центральным понятием здесь является запоминающая ячейка, которая, наряду с состоянием сети g, вычисляется на каждом шаге, используя текущее входное значение $x^{(t)}$ и значение ячейки на предыдущем шаге $c^{(t-1)}$, где t это шаг обучения.

Входной фильтр определяет, насколько значение запоминающей ячейки на текущем шаге должно влиять на результат. Значения входного фильтра лежит в интервале [0, 1], что обеспечивается областью значений функции активации:

$$g_i = \varphi(W_i x + R_i y + P_i C + b_i) \quad (4)$$

где: W_i – веса входа сети для входного фильтра, x – вход сети на текущем шаге, y – выход сети на предыдущем шаге, R_i – веса выхода сети для входного фильтра, C – состояние памяти сети,

P_i – веса состояния памяти для входного фильтра, b_i – сдвиг значений входного фильтра, φ – функция активации значения входного фильтра.

Фильтр очистки памяти удаляет часть информации запоминающей ячейки, вычисленной на предыдущем шаге:

$$g_f = \varphi(W_f x + R_f y + P_f C + b_f) \quad (5)$$

где: W_i – веса входа сети для фильтра очистки памяти, x – вход сети на текущем шаге, y – выход сети на предыдущем шаге, R_i – веса выхода сети для фильтра очистки памяти, C – состояние памяти сети, P_i – веса состояния памяти для фильтра очистки памяти, b_i – сдвиг значений фильтра очистки памяти, φ – функция активации для значения фильтра очистки памяти.

На основе данных входного фильтра и фильтра очистки памяти (4) и (5), поступающих на текущем шаге, вычисляется состояние запоминающей ячейки:

$$C(t+1) = Zg_i + C(t)g_f \quad (6)$$

где: $C(t)$ – состояние запоминающей ячейки в текущий момент времени, Z – изменение памяти, g_i – значение входного фильтра, g_f– значение фильтра очистки памяти.

Изменение памяти Z вычисляется по следующей формуле:

$$Z = \tanh(W_m x + R_m y + b_m) \quad (7)$$

где: x – вход сети на текущем шаге, W_m – веса входа сети для запоминающей ячейки, y – выход сети на предыдущем шаге, R_m– веса выхода сети для запоминающей ячейки, b_m – сдвиг значения запоминающей ячейки, $\tanh$ – функция активации для запоминающей ячейки.

Далее рассчитывается значение выходного фильтра. Выходной фильтр аналогичен двум предыдущим и имеет вид:

$$g_o = \varphi(W_o x + R_o y + P_o C + b_o) \quad (8)$$

где: W_o – веса входа сети для выходного фильтра, x – вход сети на текущем шаге, y – выход сети на предыдущем шаге, R_o – веса выхода сети для выходного фильтра, C – состояние памяти сети, P_o – веса состояния памяти для выходного фильтра, b_o – сдвиг значений выходного фильтра, φ – функция активации для значения выходного фильтра.

Итоговое значение сети определяется выходным фильтром (8) и нелинейной трансформацией над состоянием запоминающей ячейки (6):

$$y = tanh(C)g_o \quad (9)$$

где: C – состояние памяти сети, g_o – значение выходного фильтра, $tanh$ – функция активации для выходного значения сети [2].

Повышенная невосприимчивость к продолжительности разрывов во времени является преимуществом LSTM над обычными рекуррентными нейронными сетями, марковскими моделями и другими методами обучения для последовательностей во многих областях применения.

Суть прогнозирования игры заключается в том, что на основе данных нейронной сети,

собранных до текущего момента времени, она определит вероятность победы одной из команд.

На вход в нейронную сеть поступают следующие параметры: количество добытых ресурсов каждой из команд; номер команды, вероятность выигрыша которой мы хотим узнать; текущие предметы каждого из персонажей; количество убийств и использованных вспомогательных предметов, разрушенных зданий противника и оставшихся собственных зданий; опыт каждого из персонажей; распределение способностей у персонажей.

Все время одной партии игры t разбито на промежутки $t = (t_0, t_1, \dots, t_n), t > 0$. Набор игровых параметров представлен вектором $v(t) = (v_0^t, v_1^t, \dots, v_m^t)$, в конкретный момент времени игры t. Номер команды, вероятность выигрыша которой мы хотим узнать(может принимать значения 0 или 1) обозначен c. Входным набором параметров в нейронную сеть будет номер команды c и матрица:

$$X = \begin{pmatrix} v(t_0) \\ v(t_1) \\ \dots \\ v(t_n) \end{pmatrix} = \begin{pmatrix} v_0^{t_0}, v_1^{t_0}, \dots, v_m^{t_0} \\ v_0^{t_1}, v_1^{t_1}, \dots, v_m^{t_1} \\ \dots \\ v_0^{t_p}, v_1^{t_p}, \dots, v_m^{t_p} \end{pmatrix}, p < n \qquad (10)$$

Выходным параметром нейронной сети является вероятность $y[0, 1]$ выигрыша команды c.

После обучения нейронной сети на большом объеме выборки, ее можно использовать не только для прогнозирования результата игры, но так же в качестве помощника. Dota2 – это командная игра и в начале происходит стадия выбора героев. Данную нейронную сеть можно применить для выбора наилучшего героя в зависимости от уже выбранных. Также ее можно будет применить для выбора наилучшего предмета для героя в зависимости от стадии игры.

Нейронные сети будут эффективны при обучении аналитиков, участников соревнований, разработке более сложных и интеллектуальных стратегий, а также прогнозировании результатов соревнований.

Список литературы

[1]. Result Prediction by Mining Replays in Dota2. [Электронный ресурс] – Режим доступа: https://www.diva-portal.org/smash /get/diva2:829556/FULLTEXT01.pdf. – Дата доступа: 14.03.2018.

[2]. Рекуррентная нейронная сеть. [Электронный ресурс] – Режим доступа: http://mechanoid.kiev.ua/neural-net-lstm.html. – Дата доступа: 14.03.2018.

[3]. Случайный лес. [Электронный ресурс] – Режим доступа: https://ru.wikipedia .org/wiki/Random_forest. – Дата доступа: 14.03.2018.

[4]. Случайный лес (Random Forest). [Электронный ресурс] – Режим доступа: https://alexanderdyakonov.wordpress.com/2016/11/14/случайный-лес-random-forest/. – Дата доступа: 14.03.2018.

[5]. Наивный байесовский классификатор. [Электронный ресурс] – Режим доступа: http://www.machinelearning.ru/wiki/index.php?title=Наивный_байесовский_классификатор. – Дата доступа: 14.03.2018.

[6]. Метод опорных векторов. [Электронный ресурс] – Режим доступа: http://www.machinelearning.ru/wiki/index.php?title=SVM. – Дата доступа: 14.03.2018.

[7]. А.В.Гусева(Жвакина). Методика синтеза нейронной сети для моделирования динамических процессов. Инженерный вестник. – 2006. – № 1(21)/5. – С. 146 –151.

PREDICTION OF THE RESULTS OF THE COMPETITIONS BY CYBERSPORT WITH THE HELP OF NEURAL NETWORKS

S.G. MINCHUK
Master student, Department of Computer Science BSUIR, software engineer

A.V. ZHVAKINA, PhD
Associate Professor, Department of Computer Science BSUIR

Belarusian State University of Informatics and Radioelectronics, Republic of Belarus
Foreign Private Production Unitary Enterprise "ISsoft Solutions", Republic of Belarus
E-mail: minchuk94@gmail.com, zhvakina@bsuir.by

Abstract. The report considers the possibility of using neural networks to predict the results of cybersport competitions. The solu-tion of the prediction task is estimated using the following types of neural networks: the direct propagation network, the Hopfield network and the recurrent neural network LSTM.

Key words: forecasting, cybersport, neural networks, recurrent neural network LSTM.