УДК 681.324

П. П. Урбанович, профессор (БГТУ); К. В. Чуриков, магистрант (БГТУ)

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ ВЗАИМООБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ В ЗАДАЧАХ ОБМЕНА КОНФИДЕНЦИАЛЬНОЙ ИНФОРМАЦИЕЙ

В статье рассмотрены алгоритмы взаимного обучения нейронных сетей TPM, TPCM, BPM, выполняющие генерацию секретного ключа по открытым каналам связи. Особое внимание уделено алгоритмам, генерирующим ключ без непосредственной передачи информации о нем по каналу связи. Разработана программа, реализующая эти алгоритмы, и на основе данных, полученных в результате работы программы, выполнен анализ перечисленных методов на базе таких критериев, как время генерации ключа и его длина. Полученные результаты анализа позволили сделать вывод об эффективности применения рассмотренных методов в зависимости от требований, предъявляемых к уровню защиты информации. Выводы, сделанные авторами статьи, показывают необходимость улучшения функции расчета вектора обратной ошибки для увеличения скорости сходимости алгоритма.

The algorithms for mutual learning of neural networks TPM, TPCM, BPM (performing generate a secret key to open channels of communication) are considered. Particular attention is given these algorithms generate the key without the direct transfer of information about key on the communication channel. A program which implements these algorithms is made. The analysis of the methods on the basis of criteria such as time of key generation depending on its length and on data obtained is realized too. The results of analysis led to the conclusion according the effectiveness of these methods, depending on the requirements for the level of information security, is given. It's necessary to improve the function vector of the inverse calculation errors to increase the rate of convergence of the algorithm.

Введение. Нейрокриптография – это раздел криптографии, изучающий применение стохастических алгоритмов, в частности нейронных сетей, для шифрования и криптоанализа.

В криптоанализе используется способность нейронных сетей исследовать пространство решений. Существует также возможность создавать новые типы атак на существующие алгоритмы шифрования, основанные на том, что любая функция может быть представлена нейронной сетью. Взломав алгоритм, можно найти решение, по крайней мере, теоретически.

При этом используются такие свойства нейронных сетей, как взаимное обучение, самообучение и стохастическое поведение, а также низкая чувствительность к шуму, неточностям (искажения данных, весовых коэффициентов, ошибки в программе). Они позволяют решать проблемы криптографии с открытым ключом, распределения ключей, хеширования и генерации псевдослучайных чисел. Для обмена ключами между двумя абонентами наиболее часто используется алгоритм Диффи-Хеллмана [1]. Его более безопасная замена основана на синхронизации двух древовидных машин четности (TPM – tree parity machines). Синхронизация этих машин похожа на синхронизацию двух хаотических осцилляторов в теории хаотических связей (chaos communications).

В статье анализируются некоторые важные аспекты, относящиеся к установлению режима синхронизации между двумя нейронными сетями, на основе чего определяется взаимный секретный ключ.

Основная часть. В основе изучаемого метода лежит известная архитектура TPM, представляющая собой особый вид многоуровневой нейронной сети прямого распространения (рис. 1).

Рассматриваемая архитектура состоит из одного выходного нейрона, K скрытых нейронов и KN входных нейронов (рис. 1). Входные нейроны принимают двоичные значения:

$$x_{ij} \in \{-1; +1\}. \tag{1}$$

Веса (весовые коэффициенты), влияющие на входные значения нейрона (между входными и скрытыми нейронами), соответствуют

$$w_{ij} \in \{-L, \ldots, 0, \ldots, +L\}. \tag{2}$$

Значение каждого скрытого нейрона есть сумма произведений входного значения и весового коэффициента:

$$\sigma_i = \operatorname{sgn} \sum_{j=1}^{N} w_{ij} x_{ij}, \qquad \operatorname{sgn}(x) = \begin{cases} -1 & \text{if} \quad x \le 0, \\ 1 & \text{if} \quad x > 0. \end{cases} \tag{3}$$

Значение выходного нейрона есть произведение всех скрытых нейронов:

$$\tau = \prod_{i=1}^{K} \sigma_i. \tag{4}$$

Выходное значение также является величиной двоичной.

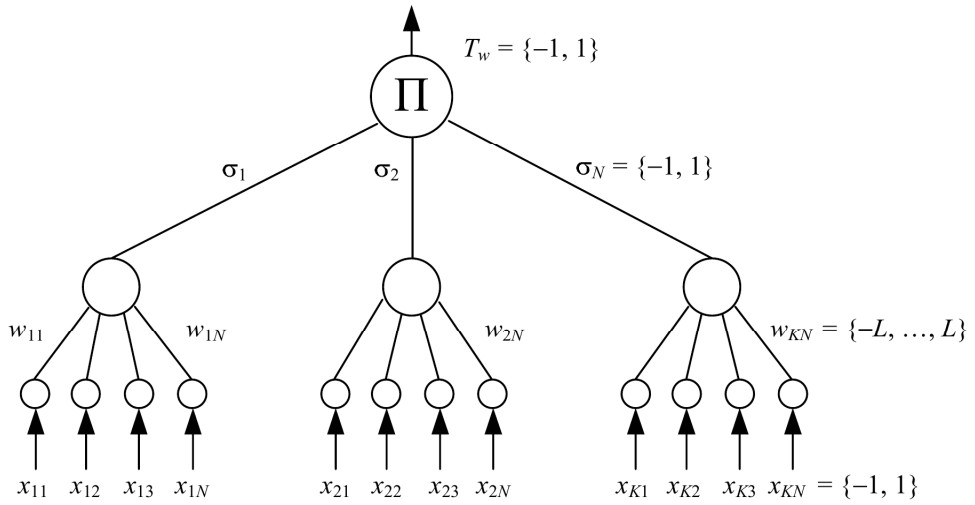

Рис. 1. Архитектура нейронной сети на основе TPM

На рис. 1 x_{ij} – значения входных нейронов (1), получаемые из вектора входных значений X_{KN}, где $i \in [1, K]$, а $j \in [1, N]$, w_{ij} – значения весовых коэффициентов (2), задающихся случайным образом при первой инициализации нейронной сети.

Алгоритм обмена информацией на основе TPM. Рассматриваем две сети (A и B), каждая из которых построена на основе архитектуры TPM (как на рис. 1). Сети обмениваются выходными величинами по открытым каналам. Синхронизация сетей происходит в соответствии со следующим алгоритмом.

1. Задаем случайные значения весовых коэффициентов $w_{11}, w_{12}, \ldots, w_{KN}$.

2. Выполняем следующие шаги, пока не наступит синхронизация.

2.1. Генерируем случайный входной вектор $X(x_{11}, \ldots, x_{1N}, \ldots, x_{KN})$.

2.2. Вычисляем значения скрытых нейронов по формуле (3).

2.3. Рассчитываем значение выходного нейрона по формуле (4).

3. Сравниваем выходы двух TPM.

3.1. Выходы разные: переход к п. 2.1.

3.2. Выходы одинаковые: применяем выбранное правило к весовым коэффициентам.

После полной синхронизации (веса w_{ij} обоих TPM одинаковые) абоненты (сети) A и B могут использовать веса в качестве ключа.

Этот метод известен как двунаправленное обучение.

Алгоритм обучения двух сетей на основе алгебры комплексных чисел (сети TPCM – tree parity complex machines) схож с алгоритмом, рассмотренным для TPM [1]. Отличие состоит, понятно, в применении алгебры комплексных чисел.

Метод обратного распространения ошибки – метод обучения многослойного персептрона. Впервые метод был описан в 1974 г. А. И. Галушкиным [2], а также независимо и одновременно П. Дж. Вербосом [3]. Далее в 1986 г. существенно развит Д. И. Румельхартом, Дж. Е. Хинтоном и Р. Дж. Вильямсом [4] и независимо и одновременно С. И. Барцевым и В. А. Охониным [5]. Это итеративный градиентный алгоритм, который используется с целью минимизации ошибки работы многослойного персептрона.

Основная идея метода состоит в распространении сигналов ошибки от выходов сети к ее входам в направлении, обратном прямому распространению сигналов в обычном режиме работы. В [5] предложен общий метод («принцип двойственности»), применимый к более широкому классу систем, включая системы с запаздыванием, распределенные системы и т. п.

Алгоритм обратного распространения ошибки используется для многослойного персептрона (рис. 2). Сеть имеет множество входов $x_1, \ldots, x_n$, множество выходов *Outputs* и множество внутренних узлов. Пронумеруем все узлы (включая входы и выходы) числами от 1 до N (сквозная нумерация, вне зависимости от топологии слоев). Обозначим через $w_{i,j}$ вес, стоящий на ребре, которое соединяет i-й и j-й узлы, а через o_i – выход i-го узла. Если нам известен обучающий пример (правильные ответы сети t_k), то функция ошибки, полученная по методу наименьших квадратов, выглядит так:

$$E(\{w_{i,j}\}) = \frac{1}{2} \sum_{k \in Outputs} (t_k - o_k)^2.$$

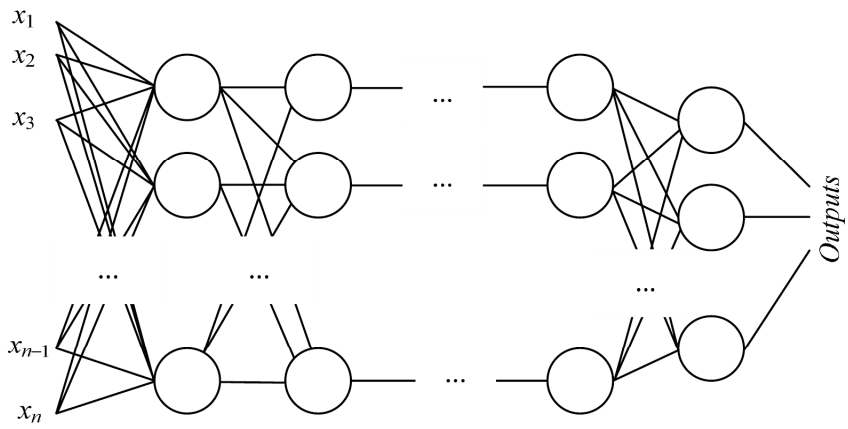

Рис. 2. Архитектура многослойного персептрона

Мы будем реализовывать стохастический градиентный спуск, т. е. «подправлять» веса после каждого обучающего примера и таким образом «двигаться» в многомерном пространстве весов. Чтобы «добраться» до минимума ошибки, нам нужно «двигаться» в сторону, противоположную градиенту, т. е. на основании каждой группы правильных ответов добавлять к каждому весу $w_{i,j}$:

$$\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}},$$

где $0 < \eta < 1$ – множитель, задающий скорость движения.

Производная считается следующим образом. Пусть сначала $j \in Outputs$, т. е. интересующий нас вес входит в нейрон последнего уровня. Сначала отметим, что $w_{i,j}$ влияет на выход сети только как часть суммы:

$$S_j = \sum_i w_{i,j} x_i,$$

где сумма берется по входам j-го узла. Поэтому:

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial S_j} \frac{\partial S_j}{\partial w_{i,j}} = x_i \frac{\partial E}{\partial S_j}.$$

Аналогично S_j влияет на общую ошибку только в рамках выхода j-го узла o_j (напоминаем, что это выход всей сети). Поэтому:

$$\frac{\partial E}{\partial S_j} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial S_j} =$$

$$= \left(\frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in Outputs} (t_k - o_k)^2 \right) \left(\frac{\partial \sigma(S_j)}{\partial S_j} \right) =$$

$$= \left(\frac{1}{2} \frac{\partial}{\partial o_j} (t_k - o_k)^2 \right) (o_j (1 - o_j)) =$$

$$= -o_j (1 - o_j)(t_j - o_j).$$

Если же j-й узел не на последнем уровне, то он имеет выходы; обозначим их через *Children*(j). В этом случае:

$$\frac{\partial E}{\partial S_j} = \sum_{k \in Children(j)} \frac{\partial E}{\partial S_k} \frac{\partial S_k}{\partial S_j}$$

и

$$\frac{\partial S_k}{\partial S_j} = \frac{\partial S_k}{\partial o_j} \frac{\partial o_j}{\partial S_j} = w_{i,j} \frac{\partial o_j}{\partial S_j} =$$

$$= w_{i,j} o_j (1 - o_j),$$

где $\frac{\partial E}{\partial S_k}$ – это в точности аналогичная поправка, но вычисленная для узла следующего уровня (будем обозначать ее через δ_k, от Δ_k она отличается отсутствием множителя $(-\eta x_{i,j})$. Теперь рассмотрим *алгоритм BackPropagation.*

1. Инициализируем $\{w_{ij}\}_{i,j}$ малыми случайными значениями $\{\Delta w_{ij}\}_{i,j} = 0$.

2. Повторяем данный шаг d раз (для всех d от 1 до m).

2.1. Подаем $\{x_i^d\}$ на вход сети и подсчитываем выходы o_i каждого узла.

2.2. Для всех $k \in Outputs$ подсчитываем:

$$\delta_k = o_k(1 - o_k)(t_k - o_k).$$

2.3. Для каждого уровня i, начиная с предпоследнего (для каждого узла j уровня i) вычисляем:

$$\delta_j = o_j (1 - o_j) \sum_{k \in Children(j)} \delta_k w_{j,k}.$$

2.4. Для каждого ребра $\{i, j\}$ сети рассчитываем:

$$\Delta w_{i,j} = \alpha \Delta w_{i,j} + (1 - \alpha) \eta \delta_j o_i,$$

$$w_{i,j} = w_{i,j} + \Delta w_{i,j}.$$

3. Обмениваемся (между сетями) значениями w_{ij}, где α – коэффициент инерциальности для сглаживания резких скачков при перемещении по поверхности целевой функции.

Для реализации анализируемых алгоритмов были разработаны программы с использованием архитектуры клиент-серверного приложения.

Целью написания программ являлось сравнение скорости обучения в локальной сети в зависимости от длины ключа и в дальнейшем – устойчивости каждого алгоритма к геометрическим атакам.

Программы написаны в среде Microsoft Visual Studio 2005 как Win32 console application. В окне клиентской части программы, реализующей методы TPCM и TPM, задается количество шагов и адрес сервера, к которому производится подключение клиента. Интерфейс программы, реализующей метод BPM, аналогичен интерфейсу, приведенному выше. Отличие заключается в том, что в BPM не надо задавать количество шагов обучения, так как особенностью метода является то, что он сам «решает», когда нейронные сети полностью синхронизировались.

При проведении тестирования (на время обучения) получены следующие результаты (таблица).

Статистика результатов проведенных испытаний

Алгоритм обучения	Время синхронизации (с) двух сетей при получении ключа длиной		
	48 символов	66 символов	132 символа
TPM	22,176	28,393	40,3
TPCM	1 800,27	2 270,5	3 287,88
BPM	35 770,2	49 084,1	98 368,23

Заключение. Как видно из проведенного анализа и данных в таблице, преимуществом алгоритмов обучения сетей TPM и TPCM является скорость обучения. Однако их главный недостаток заключается в том, что обучающиеся нейронные сети наперед «не знают» количества шагов, за которые они обучатся, и пользователи практически вынуждены задавать их вручную. Но с увеличением длины ключа количество шагов растет, и, следовательно, пользователи должны каждый раз подбирать это количество. Также время обучения по данным методикам зависит от пороговых значений L. В данном примере принято, что L изменяется в диапазоне от –3 до 3, и, соответственно, весовые коэффициенты принимают всего 7 различных значений. Преимуществом алгоритма BPM является то, что сети «знают», когда они полностью обучились. Кроме того, данный алгоритм обучения использует алгебру дробных чисел, и точность после запятой в данном примере достигает 4 знака (это числа в диапазоне от 0,0001 до 0,9999 с шагом 0,0001), т. е. 10 000 различных значений. Точность алгоритма BPM зависит от функции, которая применяется для расчета ошибки: чем «грубее» функция, тем больше шаг и, следовательно, меньше точноть. Но достоинством такого подхода является меньшее время, затрачиваемое на обучение нейронных сетей. При применении более точной функции время сходимости увеличивается, однако точность получаемого результата также растет.

Литература

1. Плонковски, М. Криптографическое преобразование информации на основе нейросетевых технологий / М. Плонковски, П. П. Урбанович // Труды БГТУ. Сер. VI, Физ.-мат. науки и информатика. – 2005. – Вып. XIII. – С. 161–164.

2. Галушкин, А. И. Синтез многослойных систем распознавания образов / А. И. Галушкин. – М.: Энергия, 1974. – С. 25–27.

3. Werbos, P. J. Beyond regression: New tools for prediction and analysis in the behavioral sciences / P. J. Werbos // Ph.D. thesis. – Harvard University, Cambridge, MA, 1974. – P. 87–89.

4. Rumelhart, D. E. Learning Internal Representations by Error Propagation. In: Parallel Distributed Processing / D. E. Rumelhart, G. E. Hinton, R. J. Williams. – 1986. – Vol. 1. – P. 318–362.

5. Барцев, С. И. Адаптивные сети обработки информации / С. И. Барцев, В. А. Охонин. – Красноярск: Ин-т физики СО АН СССР, 1986. – С. 102–109.

Поступила в редакцию 31.03.2010