УДК 681.5.09; 004.89

О МОДЕЛИРОВАНИИ И ОЦЕНКЕ НАДЕЖНОСТИ СЛОЖНЫХ ПРОГРАММНЫХ КОМПЛЕКСОВ

Кабак И.С., Суханова Н.В.

МГТУ «СТАНКИН»

ikabak@mail.ru

Современное программное обеспечение (ПО) для систем управления и автоматизации технологических процессов можно рассматривать как сложную техническую систему, где используются стандартные элементы с гарантированным уровнем надежности. Для оценки надежности сложного программного обеспечения предлагается использовать методы искусственного интеллекта – нечеткую логику и искусственные нейронные сети (НС). Описан способ последовательного синтеза многослойной НС для оценки надежности сложного ПО.

Ключевые слова: надежность, моделирование, искусственные нейронные сети.

ABOUT RELIABILITY SIMULATING OF COMPLEX SOFTWARE

Kabak I.S., Sukhanova N.V.

MSTU «STANKIN»

The modern software for control systems and automation of technological processes can be considered as complex technical system where standard elements with the guaranteed level of reliability are used. For an estimation of reliability of the complex software it is offered to use methods of artificial intelligence – the Fussy logic and artificial neural networks. The way of consecutive synthesis in multilayered neural networks for an estimation of reliability complex software is described.

Keywords: complex software reliability, artificial neural networks.

1. Прогнозирующая модель надежности сложного программного обеспечения

Рассмотрим надежность сложного программного обеспечения. Выделим в составе сложного ПО отдельные элементы – библиотеки, функции, процедуры, классы, объекты и т.д. Предположим, что надежность всех элементов, из которых состоит сложное ПО, может быть аппроксимирована экспоненциальным законом распределения вероятностей с параметром β_i [1]. Это подтверждается многочисленными экспериментальными исследованиями надежности ПО [2].

$$p_i(t) = \exp(-\beta_i \cdot t). \tag{1}$$

Для оценки надежности сложных программных систем предлагается использовать следующую математическую модель [3, 4]:

$$h(t) = \sum_{i=1}^{n} [\alpha_i(t) \cdot \exp(-\beta_i \cdot t)], \tag{2}$$

где α_i – коэффициент влияния i-того элемента сложного ПО на общую надежность, β_i – параметр функции распределения надежности i-того элемента ПО.

Получаем формулу для оценки надежности сложных программных систем:

$$h(t) = \sum_{i=1}^{n} [\alpha_i(t) \cdot p_i(t)]. \tag{3}$$

Формула (3) используется для оценки надежности сложного ПО на всех этапах его жизненного цикла, при разработке структуры ПО, при выборе способов реализации его элементов. Она позволяет построить оценку надежности сложного ПО по данным об отказах его элементов, т.е. дать прогноз надежности сложного ПО еще до момента его создания.

2. Оценка надежности сложного ПО с помощью нейронных сетей

Для оценки надежности сложного ПО предлагается использовать нейронные сети (НС) особого вида [3, 5, 6].

Рассмотрим достаточно сложное ПО, где отдельные элементы реализованы с помощью разных методов разработки ПО (методов объектно-ориентированного программирования, методов процедурно-ориентированного программирования и т.п.) и разными коллективами разработчиков. Количество элементов ПО может быть достаточно большим. Количеству элементов ПО соответствует количество экспонент в формуле (2). Например, при использовании современных методов написания объектноориентированных программ количество классов может быть порядка десятков тысяч.

Поэтому для точной оценки надежности надо использовать достаточно большое количество членов ряда в формуле суммы экспонент (2). Наличие экспонент в формуле (2) говорит о том, что при оценке мы используем методы нелинейной оптимизации. Нелинейная оптимизация дает хороший результат, если количество членов в формуле суммы ряда (2)

равно 1–2. При большем числе членов в формуле суммы ряда этот метод требует больших затрат времени, работает медленно, неэффективно и требует громоздких вычислений и больших вычислительных ресурсов.

Рассмотрим новый метод моделирования надежности ПО с помощью НС. Устройство для моделирования надежности ПО строится на базе обученных НС. Используются многослойные НС особого типа, моделирование каждого слагаемого в формуле (2) выполняет отдельный слой НС. Количество слоев НС равно количеству разных элементов ПО – типов данных, классов, функций и т.п.

Важной особенностью НС является ее горизонтальная слоистость. В многослойной НС слои располагаются вертикально, а здесь – горизонтально. Многослойная НС приобретает свои особые свойства в результате обучения.

Обучение такой НС – это длительный и трудоемкий процесс, который состоит из нескольких этапов (см. рис. 1). Использованы следующие условные обозначения элементов:

1 – блок оценки параметра β, 2 – блок оценки параметра α, 3 – блок расчета экспоненциальной функции $\alpha \cdot \exp(-\beta \cdot t)$, 4 – сумматор, 0 – элемент сигнала уровня «0».

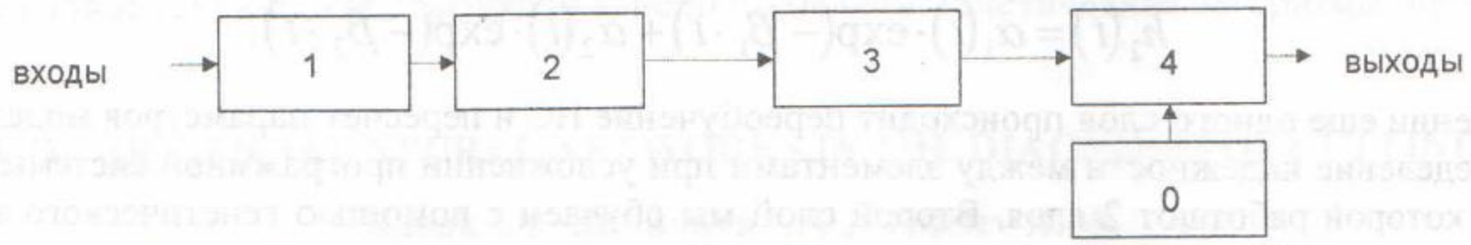

Рис. 1. Первый этап: однослойная НС для оценки надежности ПО

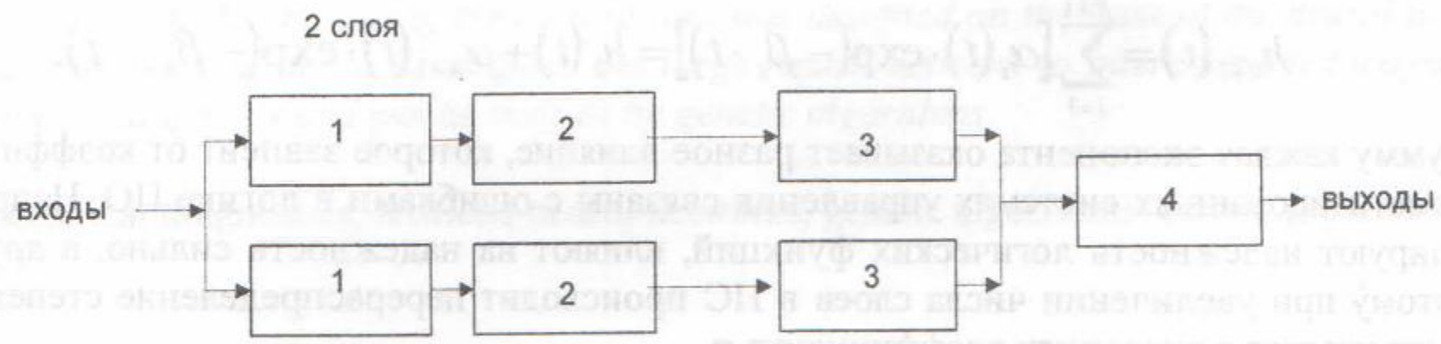

Рис. 2. Второй этап: двухслойная НС для оценки надежности ПО

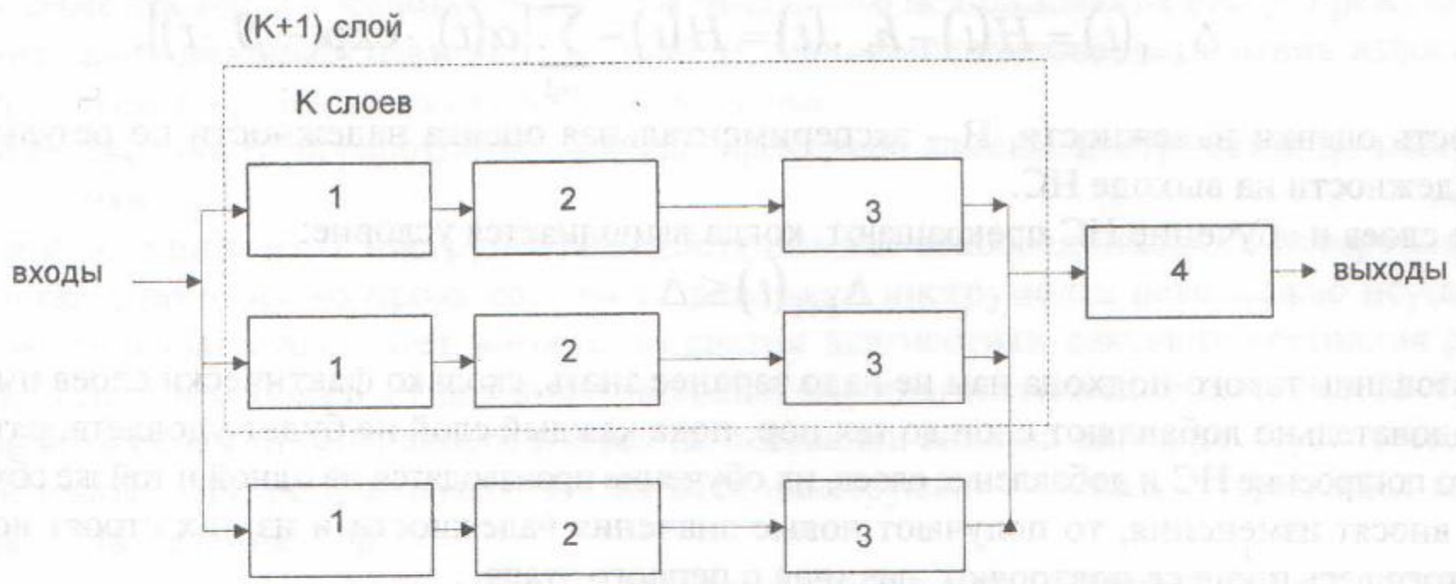

Рис. 3. НС для оценки надежности ПО с (k+1) слоями

Обучение НС представляет собой последовательный процесс:

– первая НС состоит из 1 слоя (см. рис. 1);

– вторая НС состоит из 2 слоев (см. рис. 2);

– К-тая НС состоит из k слоев;

– (К+1)-тая НС состоит из (k +1) слоев (см. рис. 3).

При оценке надежности ПО создаем последовательность НС, которая сходится к результату – многослойной горизонтальной НС, причем каждая последующая НС имеет на 1 слой больше, чем предыдущая.

Для обучения НС на первом и всех последующих шагах используется одна обучающая выборка, которая содержит статистические данные о надежности ПО и времени отладки, полученные в ходе его разработки.

Для обучения однослойной НС используем алгоритм обратного распространения ошибки. Трудоемкость этого алгоритма зависит от количества нейронов. Чем больше нейронов, тем сложнее обучение.

Первый слой НС мы обучаем указанным традиционным методом. Начиная со второго слоя для обучения НС мы используем генетические алгоритмы.

При обучении первого слоя важно получить хорошее начальное приближение, т.е. исходную точку для дальнейшего движения. Иначе мы будем идти к оптимальному решению очень долго и можем вообще не прийти к нему при ограничениях на время обучения НС.

Генетический алгоритм – это итерационный процесс. Каждая итерация улучшает свойства системы. Рассмотрим, как можно получить хорошую первую итерацию. Используем формулу (2) для i=1:

$$h_1(t) = \alpha_1(t) \cdot \exp(-\beta_1 \cdot t). \quad (4)$$

Прологарифмируем h(t), получим в результате уравнение, линейное относительно параметров α и β, которое достаточно просто решается:

$$\ln(h_1(t)) = \ln \alpha_1(t) - \beta_1 \cdot t. \quad (5)$$

В начале обучения используем НС, состоящую из одного слоя. Затем мы добавляем к этой обученной сети второй слой. Двухслойная НС описывает следующую модель надежности ПО и имеет вид:

$$h_2(t) = \alpha_1(t) \cdot \exp(-\beta_1 \cdot t) + \alpha_2(t) \cdot \exp(-\beta_2 \cdot t). \quad (6)$$

При добавлении еще одного слоя происходит переобучение НС и пересчет параметров модели α и β , что отражает перераспределение надежности между элементами при усложнении программной системы. На втором этапе создаем НС, в которой работают 2 слоя. Второй слой мы обучаем с помощью генетического алгоритма. В результате мы получаем 2-слойную обученную сеть (см. рис. 2).

НС из (k+1) слоев описывает следующую модель надежности ПО:

$$h_{k+1}(t) = \sum_{i=1}^{k+1} [\alpha_i(t) \cdot \exp(-\beta_i \cdot t)] = h_k(t) + \alpha_{k+1}(t) \cdot \exp(-\beta_{k+1} \cdot t). \quad (7)$$

На общую сумму каждая экспонента оказывает разное влияние, которое зависит от коэффициента α_i. Основные ошибки в автоматизированных системах управления связаны с ошибками в логике ПО. Например, компоненты, которые моделируют надежность логических функций, влияют на надежность сильно, а другие компоненты влияют слабо. Поэтому при увеличении числа слоев в НС происходит перераспределение степени влияния слоев на работу НС, что отражается в значениях коэффициента α_i.

Далее мы определяем погрешность оценки надежности ПО Δ с помощью НС как разницу между оценками по результатам испытаний ПО и результатам работы математической модели (2), реализованной с помощью многослойной НС:

$$\Delta_{k+1}(t) = H(t) - h_{k+1}(t) = H(t) - \sum_{i=1}^{k+1} [\alpha(t)_i \cdot \exp(-\beta_i \cdot t)], \quad (8)$$

где Δ – погрешность оценки надежности, H – экспериментальная оценка надежности по результатам испытаний ПО, h – оценка надежности на выходе НС.

Добавление слоев и обучение НС прекращают, когда выполняется условие:

$$\Delta_{k+1}(t) \le \Delta. \quad (9)$$

При использовании такого подхода нам не надо заранее знать, сколько фактически слоев имеет НС.

В НС последовательно добавляют слои до тех пор, пока каждый слой не будет удовлетворять условию (9).

Отметим, что построение НС и добавление слоев, их обучение производятся на одной и той же обучающей выборке.

Если в ПО вносят изменения, то получают новые значения надежности и из них строят новую обучающую выборку. После этого весь процесс повторяют, начиная с первого этапа.

Можно создать несколько НС применительно к каждому изменению ПО. Таким образом, любое изменение ПО приводит к созданию новой НС.

Выводы

Для оценки надежности ПО используют описанный выше метод последовательного синтеза НС. Этот метод сходится к многослойной НС, которая строит оценку надежности ПО с заданной точностью.

Библиография

1. Надежность технических систем: справочник / под ред. И.А. Ушакова. – М.: Радио и связь, 1985. – 606 с.

2. Майерс Г. Надежность программного обеспечения. – М.: Мир, 1980. – 359 с.

3. Степанов С.Ю., Кабак И.С. Алгоритм фрагментации больших нейронных сетей и исследование его сходимости // ИНФ. ТЕХ. – № 7. – 2012. – С. 73–78.

4. Кабак И.С., Суханова Н.В. Моделирование надежности программного обеспечения систем управления автоматизированными технологическими комплексами на базе искусственного интеллекта // Вестник МГТУ «Станкин». – № 1 (19). – 2012. – С. 95–99.

5. Кабак И.С., Суханова Н.В. Технология реализации автоматизированных систем управления на базе больших искусственных нейронных сетей МОДУС-НС // Межотраслевая информационная служба. – 2012.

6. Кабак И.С. Создание больших аппаратно-программных нейронных сетей для систем управления. Авиационная промышленность. – № 4. – 2012.