УДК 519.85
DOI: 10.22213/2410-9304-2022-3-94-103

Разработка параллельных алгоритмов обучения вероятностных моделей тестирования веб-приложений

П. В. Полухин, кандидат технических наук, Воронежский государственный университет, Воронеж, Россия

В современных условиях развития методов и алгоритмов тестирования особую значимость представляет объединение отдельных компонентов тестирования в виде иерархической модели, отражающей связи и состояния между данными компонентами, а также позволяющие оценить вероятность перехода между состояниями модели в случае появления информации относительно успешной реализации определенного теста для выявления программной ошибки. Существующие подходы не позволяют произвести оптимальную настройку параметров модели, а также произвести качественный расчет и установление направленностей связей между отдельными компонентами данной модели. В рамках научного исследования рассмотрена возможность применения эффективных численных методов для решения задачи обучения вероятностных моделей, построенных на основе байесовских сетей для решения основных задач тестирования веб-приложений. Рассмотрены основные подходы к созданию параллельных алгоритмов, реализующих основные функциональные возможности, лежащие в основе реализации процедуры обучения моделей тестирования. Произведен анализ эффективности разработанных алгоритмов обучения для тестирования определенных групп программных ошибок и позволяющий наиболее оптимальные параметры модели ДБС, а также произведено обоснование их использования в распределенных системах обработки данных. Разработаны алгоритмические решения для оптимизации расчета матриц Якоби и Гессе на основе алгоритмов Кэннона и Фокса. Выполнено моделирование процесса тестирования ошибок веб-приложений и представление его в виде динамической байесовской сети, полученной по результатам реализации процедуры обучения структуры и параметров. Обоснованность всех теоретических результатов подтверждена большим числом экспериментальных результатов, доказывающих состоятельность выдвинутых предположений, методов и моделей тестирования, представленных в виде динамических байесовских сетей.

Ключевые слова: вероятностная модель, байесовские сети, метод Бройдена, метод Левенберга – Марквардта, метрика Байеса – Дирихле.

Введение

В процессе моделирования стохастических процессов тестирования особое место выделяют вероятностным моделям, построенным на основе байесовских сетей (БС), позволяющим учитывать особенности и специфику тестирования определенных групп и классов программных ошибок. Из анализа существующих алгоритмов обучения следует, что они не в полной мере адаптированы к решению обучения моделей со сложной топологической структурой и не подходят к временным вероятностным моделям. Присутствие временных связей между отдельными компонентами тестирования, представляемых в виде модели динамической байесовской сети (ДБС), делают проблематичным применение классических алгоритмов обучения на основе статистического анализа и выполнения тестов на условную независимость вершин графа сети. Применение рассмотренных в научном исследовании подходов расширяет возможности обучения ДБС, позволяет сделать данную процедуру более направленной, что позволяет оптимизировать решения задачи топологической связности вершин графа динамической байесовской сети, а также адаптировать алгоритм к построению сложных иерархических ДБС. В качестве оценочной функции используется асимптотическая метрика Байеса – Дирихле (БД), позволяющая произвести оптимальную оценку устойчивости связи между вершинами, а также изолировать из множества узлов-кандидатов те вершины, значение метрики для которых является минимальным. Применение параллельного подхода обусловлено значительным числом матричных операций, реализуемых в рамках алгоритмов Левенберга – Марквардта и Бройдена, решающих задачу оптимального поиска значений оценок БД для определения направленности модели тестирования. Актуальность исследования обусловлена необходимостью создания универсальных оптимальных алгоритмов обучения структуры, решения задачи численной оптимизации алгоритмов обучения, а также применением разработанных подходов для построения модели тестирования на основе динамической байесовской сети, адаптированной к поиску программных ошибок веб-приложений

в рамках реализации процедуры фаззинга. Такой подход позволяет использовать предложенный алгоритм для динамических байесовских сетей тестирования веб-приложений для случаев дискретного и непрерывного распределения параметров, а также в значительной степени исключить зависимость точности алгоритма от объема обучающей выборки.

Данные и методы

Динамическая байесовская сеть представляет собой разновидность динамической модели, состоящей из набора байесовских сетей $BN = \{X, G\}$, взятых в хронологическом порядке на интервале $(t; t+k)$, $X = \{X_1, X_2, .., X_n\}$ – переменные сети, с каждой из которых связана таблица условных вероятностей, G – граф байесовской сети, устанавливающий топологические связи между переменной X и ее родительскими вершинами Y. В таком случае гипотезу об условной независимости X и Y можно сформулировать в следующем виде [1, 2]:

$$P(x_i, y|Parents(x_i)) = P(x_i|Parents(x_i)) \times \times P(y|Parents(x_i)), x_i \in X, y \in Y. \quad (1)$$

Полное совместное распределение ДБС для временного среза $t+1$ будет иметь вид [2]:

$$P(X_0, X_1, \ldots, X_t, E_1, \ldots, E_t) = = P(X_0)\prod_{i=1}^{t} P(X_i|X_{i-1})P(E_i|X_i), \quad (2)$$

где $P(X_0)$ – начальное распределение, соответствующее моменту времени $t = 0$; $P(X_i|X_{i-1})$ – модель перехода, представляющая собой марковский процесс первого рода; $P(E_i|X_i)$ – модель перехода, учитывающая возникновение свидетельств, поступающих на вход модели ДБС с $t = 0$ до момента времени $t+1$.

Для определения оценочной функции введем понятие марковского покрытия (МП). Марковское покрытие существует для каждой переменной X ДБС и состоит из совокупности множеств дочерних вершин и родителей для каждой из дочерних вершин. Отметим, что в состав МП для переменных, имеющих транзитивные связи, будут входить вершины только из соседних временных срезов так как модель перехода формирует марковский процесс первого рода. Для решения задачи поиска оптимальной структуры ДБС, необходимой для получения распределения (2), воспользуемся правилом максимума апостериорной вероятности (МАВ) для оценочной функции $\arg\max P(G|D)$ с учетом обучающей выборки $D = \{(X_k, Y_k)\}_{k=1}^{n}$. Тогда распределение Дирихле будет иметь следующий вид [4, 5]:

$$P(\theta|G) = \frac{\Gamma\left(\sum_{k=1}^{r}\alpha_k\right)}{\prod_{k=1}^{r}\Gamma(\alpha_k)}\prod_{k=1}^{r}(\theta_k)^{\alpha_k - 1}, \quad (3)$$

где α_k – параметра распределения Дирихле; r – число состояний принимаемых X_k; θ – вектор параметров БС для X; $\Gamma(\cdot)$ – гамма-функция:

$$\frac{\Gamma(n+\alpha)}{\Gamma(\alpha)} = (n-1+\alpha)(n-2+\alpha)\times\ldots\times\alpha, \quad \Gamma(\alpha+1) = \alpha\Gamma(\alpha), \Gamma(1) = 1. \quad (4)$$

Тогда для вычисления правила МАВ запишем выражение, соответствующее условиям глобальной и локальной независимости для множества переменных X, входящим в состав ДБС [6]:

$$P(\theta_i|G) = \prod_{j=1}^{q} P(\theta_{k,j}|G), \quad q = \prod_{x_l \in Y} r_l. \quad (5)$$

С учетом (5) распределение Дирихле (3), соответствующее параметрам модели θ ДБС, приведем к следующему виду:

$$P(\theta|G) = \prod_{i=1}^{n}\prod_{j=1}^{q}\frac{\Gamma\left(\sum_{k=1}^{r}\alpha_{i,j,k}\right)}{\prod_{k=1}^{r}\Gamma(\alpha_{i,j,k})}\prod_{k=1}^{r}(\theta_{i,j,k})^{\alpha_{i,jk}-1}. \quad (6)$$

В соответствии с (5) апостериорное распределение для структуры ДБС $P(D|G)$, формируемое с учетом обучающей выборки D для срезов из временного интервала $(t; t+1)$, будет иметь следующий формализованный вид [7]:

$$P(D|G) = \prod_{i=1}^{n}\prod_{j=1}^{q}\frac{\Gamma(\sum_{k=1}^{r_i} a_{i,j,k})}{\Gamma(\sum_{k=1}^{r} N_{i,j,k} + a_{i,j,k})}\pi, \quad \pi = \prod_{k=1}^{r}\frac{\Gamma(N_{i,j,k} + a_{i,j,k})}{\Gamma(a_{i,j,k})}, N_{ij} = \sum_{k=1}^{q} N_{ijk}, \quad (7)$$

где N_{ijk} – число случаев появления значения параметра в k-й выборке.

Логарифмируя обе части выражения (6), сформулируем метрику Байеса – Дирихле:

$$BD = \ln P(\theta|G) = \\ = \sum_{i=1}^{n}\sum_{j=1}^{q}\left(\ln \frac{\Gamma\left(\sum_{k=1}^{r}\alpha_{i,j,k}\right)}{\Gamma\left(\sum_{k=1}^{r}\left(N_{i,j,k}+\alpha_{i,j,k}\right)\right)} + \varphi \right), \quad (8)$$

$$\varphi = \sum_{k=1}^{r} \ln \frac{\Gamma\left(N_{i,j,k}+\alpha_{i,j,k}\right)}{\Gamma\left(\alpha_{i,j,k}\right)}.$$

Используя полученную логарифмическую формулировку метрики БД, запишем выражение, соответствующее правилу MAB, имеющее следующий вид [8]:

$$\hat{G} = arg\max_{G} P(\theta|G) = arg\max_{G} lnP(\theta|G). \quad (9)$$

Для определения структуры ДБС необходимо решить задачу поиска максимума оценочной функции БД в соответствии с обучающей выборкой D. Наибольший интерес представляет семейство квазиньютоновских алгоритмов, в частности алгоритм Левенберга Марквардта (ЛМ). В простейшем приближении данный алгоритм является сочетанием методов Гаусса – Ньютона (ГН) и градиентного метода, однако вводит дополнительный параметр регуляризации. Представленный алгоритм базируется на линейной максимизации оценочной функции $f(x) = \ln P(\theta|G)$ с использованием метода неменьших квадратов [9, 10]:

$$f(x) = \max F(X)^2 = \sum_{k=1}^{n}\left(g_k(x) - a_k\right)^2, \quad (10)$$

где $F(x) = \left[g_k(w) - a_k\right]_{k=1}^{n}$ – разностное выражение $g_k(w)$.

Тогда, чтобы найти максимум функции $f(x)$, требуется определить соответствующую ей матрицу Якоби

$$J(x) = \begin{pmatrix} \frac{dg_1(x)}{dx_1} & \cdots & \frac{dg_1(x)}{dx_n} \\ \frac{dg_2(x)}{dx_1} & \cdots & \frac{dg_2(x)}{dx_n} \\ \cdots & \cdots & \cdots \\ \frac{dg_m(w)}{dx_1} & \cdots & \frac{dg_m(x)}{dx_n} \end{pmatrix}. \quad (11)$$

Для формулировки алгоритма ЛМ необходимо определить регрессионную модель $f(Q, X_k)$, $Q = (q_1, \ldots, q_m)$, устанавливающую связь между параметрами выборки $D = \left\{(X_k, Y_k)\right\}_{k=1}^{n}$ и ожидаемым значением Q. В таком случае выражение (10) приведем к следующему виду:

$$E_D(Q) = \frac{1}{2}\sum_{k=1}^{n}\left[F_k(Q)\right]^2, \\ F_k(Q) = y_k(Q) - f(Q, Z_k), y_k \in Y_k. \quad (12)$$

Значение градиента и матрицу Гессе можно выразить через соответствующую матрицу Якоби:

$$g(Q) = \left[J(Q)\right]^T F(Q), \\ H(Q) = \left[J(Q)\right]^T J(Q) + R(Q), \quad (13)$$

где $R(Q)$ – компоненты вторых производных, $J(Q)$ – матрица Якоби.

Аппроксимируя $R(Q)$ на основе регуляризационного параметра $\lambda \geq 0$, получим решение ЛМ относительно приращения градиента ΔQ

$$\Delta Q = \left(J^T(Q)J(Q) + \lambda I(Q)\right)^{-1} J^T(Q)E(Q), \quad (14)$$

где $I(Q)$ – единичная матрица размерностью $m \times n$, $E(Q)$ – функция невязки.

Для повышения сходимости алгоритма при увеличении фактора λ целесообразно осуществить замену единичной матрицы на диагональную. Запишем формулировку алгоритма ЛМ [11]:

$$\Delta Q = \left[J^T(Q)J(Q) + \lambda diag\left[H(Q)\right]\right]^{-1} J^T E(Q). \quad (15)$$

Из выражения (13) следует, что значения матрицы Гессе имеют непосредственную зависимость от кривизны функции $f(Q)$. Как следствие, при большой кривизне будем наблюдать малое число итераций выражения (13).

Рассматривая особенности применения алгоритма ЛМ в рамках построения структуры модели ДБС, стоит отметить роль параметра регуляризации λ, позволяющего производить плавную коррекцию алгоритма обучения и исключить неточности при определении направленностей ребер графа ДБС. Одной из проблем, связанных с реализацией численных методов на основе алгоритма ЛМ, являются его достаточно большие временные затраты, характеризующиеся необходимостью интегративного пересчета матрицы $J(Q)$ на каждом шаге алгоритма. Для решения данной задачи рассмотрим метод секущих Бройдена, позволяющих осуществить расчет матрицы Якоби на основе метода

конечных разностей только для первого перехода $J_{t+1}(Q)$. В основу метода Бройдена положено решение нелинейного уравнения $F(Q)=0$. Произведем замену функция $F(Q)$ в окрестности точки Q_k на соответствующую ей аффинную модель

$$\xi_k(Q)=f(Q_k)+a_k(Q-Q_k),\xi_k=f(Q_k),a_k\in R, \tag{16}$$

где a_k – секущая для функции $F(Q)$.

Для определения параметра a_k предположим, что аффинная модель и исходная функция $F(Q)$ в точке Q_k имеют идентичное значение производных. Тогда параметр $a_k=F'(Q)$. В то же время, если $F'(Q)$ не допустимо, аффинная модель (15) будет определяться первым слагаемым $\xi_k(Q)=f(Q_k)$, а значение исходной функции $f(Q_k)$ будет эквивалентно следующему выражению с учетом параметра a_k:

$$f(Q_{k-1})=f(Q_k)+a_k(Q_{k-1},Q_k),a_k\in R. \tag{17}$$

Тогда аппроксимацию для секущих, соответствующих выражению (16), запишем в следующем виде

$$a_k=\frac{f(Q_k)+f(Q_{k-1})}{Q_k-Q_{k-1}}. \tag{18}$$

Используя выражение секущих (17), приведем выражение (16) к следующему виду:

$$a_k(Q_{k-1},Q_k)=f(Q_{k-1})-f(Q_k). \tag{19}$$

Для многомерного случая запишем аффинную модель в следующем виде:

$$\xi_k(x)=F(Q_k)+J_k(Q-Q_k), \tag{20}$$

где J_k – якобиан, соответствующий приращению $\Delta Q=Q-Q_k$.

Тогда соотношение секущих для многомерного случая запишем с использованием матрицы Якоби:

$$J_k(Q_k-Q_{k-1})=F(Q_k)-F(Q_{k-1}). \tag{21}$$

Между тем выражение (20) не дает возможность полностью определить значение матрицы J_k, вследствие чего сложно вычислить общее число матриц J_k, характеризующих процедуру преобразования параметров $z=x_k-x_k$ к $y=F(x_k)-F(x_{k-1})$. Для решения данной проблемы по аналогии с одномерным случаем опишем переход к аффинной модели для функции $F(x_k)$ в точке x_{k-1}:

$$\xi_{k-1}=F(x_{k-1})+J_{k-1}(x-x_{k-1}); \tag{22}$$

к аффинной модели в точке x_{k-1}:

$$\xi_k=F(x_k)+J_k(x-x_k). \tag{23}$$

Из аффинных моделей (21) и (22) следует, что нельзя в полной мере получить информацию относительно якобианов соответствующих им якобианов. Следовательно, критерий выбора J_k и J_{k-1} будет определяться минимальным значением аффинных моделей в процессе перехода между точками x_{k-1} и x_k. Используя выражение секущих (20), запишем разность аффинных моделей (21) и (22) в следующем виде [12]:

$$\begin{aligned}&\xi_k(x)-\xi_{k-1}(x)=\\&=F(x_k)+J_k(x-x_k)-F(x_{k-1})-J_{k-1}(x-x_{k-1})=\\&=F(x_k)-F(x_{k-1})-J_k(x_k-x_{k-1})+(J_k-J_{k-1})=\\&=(J_k-J_{k-1})(x-x_{k-1}).\end{aligned} \tag{24}$$

Для любого $x\in R$ множитель $x-x_{k-1}$ можно переписать в виде произведения векторов z и ортогонального ему t:

$$x-x_{k-1}=\alpha z+t. \tag{25}$$

Тогда разность аффинным моделей (23) может быть приведена к следующему виду:

$$\xi_k(x)-\xi_{k-1}(x)=\alpha(J_k-J_{k-1})z+(J_k-J_{k-1})t. \tag{26}$$

Из выражения (25) следует, что первое слагаемое будет являться фиксированным в силу того, что $(J_k-J_{k-1})z=y-J_{k-1}z$. Тогда для достижения минимума $\xi_k(x)-\xi_{k-1}(x)$ необходимо выбрать якобиан, для которого будет справедливо равенство нулю второго слагаемого $(J_k-J_{k-1})t=0,\forall t\perp z$. Данное условие достижимо, если разностная матрица $\nabla J=J_k-J_{k-1}$ является матрицей первого ранга. Тогда для ∇J получим следующее выражение:

$$\nabla J=J_k-J_{k-1}=\frac{(y-J_{k-1})z^T}{z^Tz}. \tag{27}$$

Из формулы (26) получим выражение Бройдена для расчета матрицы J_k [13]:

$$J_k = J_{k-1} + \frac{\left(y - J_{k-1}\right) z^T}{z^T z}. \qquad (28)$$

Из формулы Бройдена (27) следует, что для расчета матрицы Якоби на каждой следующей интеграции необходимо произвести вычисление только для первого шага J_{k-1}, все остальные значения $J_{k:k+n}$ будут вычисляться в соответствии с выражением (27). В таком случае целесообразно вычислять транспонированную матрицу $\left[\mathrm{J}(\mathrm{Q})\right]^{\mathrm{T}}$ в процессе расчета выражения (14) без необходимости хранения каких-либо промежуточных результатов. Вычисление целевых компонентов произведения h_{mn} для матрицы Гессе H можно выполнить на основе следующего выражения [14]:

$$h_{ij} = \sum_{k=0}^{n-1} J_{ik} \times J_{kj}^T = \sum_{k=0}^{n-1} J_{ik} \times J_{jk}, 0 \le i < m, 0 \le j < m. \qquad (29)$$

Следовательно, каждый элемент матрицы Гессе h_{ij} может быть определен в виде скалярного произведения столбца J на соответствующий столбец J^T. Запишем обобщенное выражение для элементов h_{ij}:

$$\begin{aligned} h_{ij} &= \left(J_i, \left(J_j^T\right)^T\right) = \left(J_i, J_j\right), \\ J_i &= \left(J_{i0}, J_{i1}, J_{i2}, \ldots, J_{in-1}\right), \\ J_j^T &= \left(J_{0j}, J_{1j}, J_{2j}, \ldots, J_{n-1j}\right)^T. \end{aligned} \qquad (30)$$

Далее в работе полагаем, что матрицы J и J^T являются квадратными матрицами размерностью $n \times n$. Если это условие не выполняется, то необходимо выполнить приведение матрицы к квадратной за счет расширения выборки S за счет создания ее копий θ с учетом условия $\theta m \ge n$. Важным следствием из уравнения (29) является попарная независимость операцией перемножения матриц J и J^T, что дает возможность распараллеливания данных операций и повышения вычислительной эффективности алгоритма ЛМ и Бройдена. Одним из наиболее распространенных способов параллельного умножения является ленточный алгоритм. Сущность данного алгоритма заключается в разделение матрицы J на строки с последующим независимым умножением на соответствующие столбцы J^T. Учитывая, что каждая из таких операций является однотипной, наиболее целесообразно произвести укрупнение операций перемножения за счет их объединения в рамках одного вычислительного процесса. Наряду с ленточными рассмотрим блочные алгоритмы Кэннона и Фокса, которые будут использоваться в рамках реализации алгоритмов обучения на основе алгоритмов ЛМ и Бройдена. Для построения данных алгоритмов будем также рассматривать квадратные матрицы $A = J$ и $B = J^T$. В процессе использования блочных алгоритмов перемножения матриц A и B происходит их разбиение на блоки $k \times k, k = n / s$, n – оригинальный размер квадратных матриц A и B; s – общее число блоков по горизонтали и вертикале. Следовательно, обобщенная формула операции блочного умножения квадратных матриц A и B будет иметь следующий вид [15]:

$$\begin{pmatrix} A_{00}A_{01}\ldots A_{0s-1} \\ A_{10}A_{11}\ldots A_{1s-1} \\ \ldots \\ A_{s-10}A_{s-11}\ldots A_{s-1s-1} \end{pmatrix} \times \begin{pmatrix} B_{00}B_{01}\ldots B_{0s-1} \\ B_{10}B_{11}\ldots B_{1s-1} \\ \ldots \\ B_{s-10}B_{s-11}\ldots B_{s-1s-1} \end{pmatrix} = \begin{pmatrix} H_{00}H_{01}\ldots H_{0s-1} \\ H_{10}H_{11}\ldots H_{1s-1} \\ \ldots \\ H_{s-10}H_{s-11}\ldots H_{s-1s-1} \end{pmatrix}, \qquad (31)$$

$$H_{ij} = \sum_{q=1}^{s} A_{iq}B_{qj} = \sum_{q=1}^{s} J_{iq}J_{qj}^T = \sum_{q=1}^{s} J_{iq}J_{jq}, i, i = 1, \ldots, s.$$

В процессе произведения умножения предполагаем, что имеется p-параллельный процессор, отвечающий за вычисления определенного блока H_{ij}. В качестве входных данных выступают элементы $A_{ij}^{'}$ и $B_{ij}^{'}$, формируемые из матриц A и B соответственно. Процедура доступа к другим блокам выполняется на основе передачи сообщения. Это исключает возникновение повторяющихся блоков и повышает общую стабильность работы блочного алгоритма. Так как каждый процессор связан с вычислением H_{ij}, то каждый из них будет отвечать за формирование квадратной решетки размерностью $s \times s$, соответствующей каждому из блоков матрицы H_{ij}.

Далее рассмотрим основные этапы выполнения алгоритмов Фокса и Кэннона. В алгоритме Фокса предварительно происходит передача сообщений на каждый из доступных процессоров p_{ij} относительно блоков A_{ij} и B_{ij}. После чего выполняется цикл $0 \le l \le s$, включающий в себя следующие этапы:

– каждому блоку A_{ij} ставятся в соответствие строки с индексами $1 \le i \le s$ из решетки $s \times s$, причем индекс процессора m, определяющий его положение в решетке, задается как $\mathrm{m} = (i + l - 1) \bmod (\mathrm{s} + 1)$;

– каждый блок A_{im} отправляется на выполнение соответствующей строки i решетки $s \times s$, $A_{ij}^{'} = A_{im}$;

– блоки $A_{ij}^{'}$ и $B_{ij}^{'}$, связанные с процессором p_{ij}, перемножаются, а потом суммируются с H_{ij}. Окончательное выражение приобретает вид: $H_{ij} = H_{ij} + A_{ij}^{'} \times B_{ij}^{'}$;

– блоки $B_{ij}^{'}$, полученные на предыдущем шаге, пересылаются на следующий процессор $p_{i-1\,j}$ (блоки для первой строки процессорной решетки пересылаются на последнюю).

Из алгоритма Фокса следует, что общее время на выполнение алгоритма можно определить с учетом времени выполнения единичной скалярной операции умножения τ:

$$T_{\text{общ}} = s\left(\frac{n^2}{p}\left(\frac{2n-1}{s-1}+1\right)\right)\tau.$$

Алгоритм Кэннона имеет лишь одно существенное отличие от алгоритма Фокса, связанное с первичным распределением блоков $A_{ij}^{'}$ и $B_{ij}^{'}$. Данная процедура инициализации может быть выполнена без предварительной передачи данных. Сформулируем основные этапы выполнения алгоритма Кэннона:

– на каждый из доступных процессоров p_{ij} осуществляется рассылка блоков из A_{ij} и B_{ij}. Значения элементов матрицы H_{ij} первоначально имеют нулевое значение;

– для строки i и столбца j производится сдвиг A_{ij} на i позиций влево и j позиций вверх;

– через число операций $\sqrt{p}$ получим матрицы H_{ij} на каждом их процессоров p_{ij}, после чего пересылаем все сообщения главному процессу для формирования матрицы H.

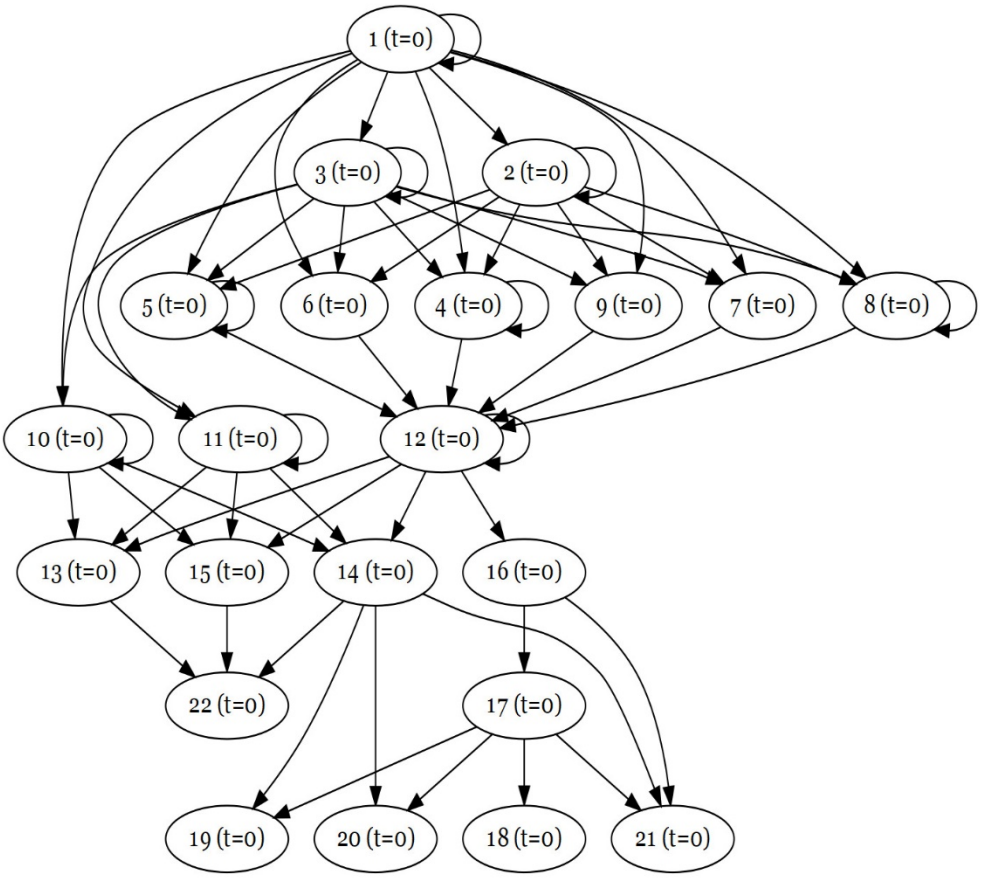
Преимущество приведенных алгоритмов применительно к решению задач обучения обусловлено их высокой производительностью, что дает возможность реализовать процедуру обучения, используя большой объем обучающей выборки и позволяет произвести оптимальное использование памяти вычислительной системы.

Полученные результаты

В экспериментальной части исследования произведем построение структуры ДБС для моделирования процесса тестирования «Инъекций» веб-приложений. Инъекции являются разновидностью программных ошибок, возникающих в веб-приложениях, взаимодействующих с системами управления базами данных (СУБД), а также позволяющих выполнять команды операционных систем. Наибольшая доля возникающих программных ошибок связана с некорректной реализацией фильтров, отвечающих за разбор и проверку корректности обработки входных параметров. Среди инъекций выделяют следующие разновидности: SQL, команд и кода. Для реализации процедуры обучения ДБС «Инъекции» (рисунок) используется распределенная среда Spark и Hadoop, развернутых в рамках облачной платформы Yandex Cloud, состоящей из 6 узлов со следующей аппаратной конфигурацией: 2 процессора Intel Xeon-Platinum 2.5 GHz 16 ядер, 128 GB ОЗУ, жесткий диск 10 TB. Процедура тестирования производилась для следующих веб-приложений: 1С Битрикс 16.5,16.5.4; Joomla 1.5.15, 2.5; WordPress 4.0,4.1; Mutillidae 2; Bricks 1.6; Damn Vulnerable Web Application 2.0.1; OWASP Vicnum 1.7. Приведенные приложения используются как для создания персональных веб-приложений, так и для разработки универсальных средств тестирования.

Для рисунка приведем следующие характеристики узлов ДБС, представляющих собой пространственно-временные состояния переменных сети. Соответствующие им числовые обозначения имеют следующие функциональные обозначения: типы инъекций: SQL, кода и команд (1); механизмы обхода межсетевых экранов веб-приложений на основе смешивания http-параметров (2); механизмы кодирования и шифрования, предназначенные для обфускации и обхода механизмов фильтрации (3); различные подходы эксплуатации SQL-инъекции: Union, Blind, Time Based blind, Boolean Based Blind, Error Based Blind, Out Of Band, Stacked Queries (4, 5, 6, 7, 8, 9); механизмы инъекции кода и команд (10, 11); определение типа и версии СУБД (12); исполнение команд операционной системы (13); получение доступа к файловой системе сервера, где установлена СУБД (14); получение доступа к данным, хранящимся в СУБД, получение доступа к локальной сети из командного интерфейса СУБД или посредством

инъекции кода или команд (15); получение структуры таблиц и баз данных (16,17); нарушение механизмов аутентификации (18), авторизации (19), целостности (20), конфиденциальности (21) и доступности (22). Узлы ДБС, имеющие циклы, характеризуют наличие транзитивных связей между временными срезами t и $t+1$, задаваемых в соответствии с моделью перехода $P(X_i|X_{i-1})$.



Обученная структура ДБС «Инъекции»

Learned structure for DBN «Injection»

Приведем сравнение времени обучения структуры ДБС «Инъекции» для различного числа ядер процессоров, доступных в рамках распределенной системы Spark, а также при использовании алгоритмов Бройдена и Фокса в сочетании с ЛМ и его сравнение с классическим алгоритмом ЛМ при обработке 500 ГБ обучающей выборки, полученной по результатам тестирования «инъекций» веб-приложений.

Из таблицы следует, что применение метода Бройдена и матричного параллельного алгоритма Кэннона в сочетание с алгоритмом Левенберга – Марквардта позволяет получить наиболее оптимальное расчетное время работы алгоритма обучения структуры ДБС «Инъекции». Применение распределенной платформы и ее адаптация к решению основных задач обучения, связанных с определением связей и их направленностей между узлами ДБС X_k с учетом вычисления их марковского покрытия относительно всех других вершин Y_k, дает возможность применения разработанных подходов к созданию наиболее оптимальной модели тестирования, предназначенной для интеллектуального выстраивания связей между отдельными модулями и компонентами тестирования, а также способной использовать статистические данные для своевременной настройки и коррекции тестовых генераций в процессе выполнения процедуры тестирования веб-приложений.

Сравнение производительности процедуры обучения ДБС «Инъекции»

Performance Comparison of the «Injection» DBN Learning Procedure

Число ядер распределенной системы	ЛМ	ЛМ и метод Бройдена	ЛМ и метод Бройдена, Кэннона
90	284,598175 с	161,196364 с	100,886079 с
50	532,671035 с	254,22799 с	189,142987 с
20	1123,126919 с	587,806883 с	376,620282 с
10	2138,359028 с	1243,651316 с	679,543620 с

Следовательно, применение разработанного алгоритма обучения направлено на оптимизацию процедуры построения направленной структуры ДБС, позволяет учитывать дискретное и непрерывное распределение параметров, а также адаптироваться к анализу временных состояний, заданных на интервале $(t;t+k)$. Представленный алгоритм обладает высоким уровнем горизонтальной масштабируемости и может быть использован в рамках любой из параллельных систем, предназначенных для реализации вычислительных процедур и обработки большого объема обучающих данных.

Заключение

Численная оптимизация процедур обучения ДБС, построенных на основе поиска экстремальных оценок Байеса – Дирихле, является универсальным подходом, направленным на получение наиболее достоверной структуры сети, адаптированной для решения задач тестирования веб-приложения. Затронутые вопросы оптимизации матричных операций, используемых в рамках алгоритма Левенберга – Марквардта, позволяют в значительной степени оптимизировать данный алгоритм и снизить временные затраты на получение искомой структуры ДБС. Проведенный вычислительный эксперимент доказывает правильность научных положений и подходов, рассмотренных в рамках решения основных задач обучения ДБС, позволяющих адаптировать применение алгоритмов в условиях большого числа временных срезов и параметров ДБС, а также к росту объема обучающей выборки. Основным результатом исследования является создание перспективного алгоритма обучения структуры сети, сочетающего в себе статистические и функциональные подходы, а также набор численных методов позволяющих получить оптимальную структуру ДБС, адаптированную к решению вероятностных задач области тестирования программ. Снижение алгоритмической сложности обучения связано с реализацией распределенной обработки обучающей выборки и матричных операцией, необходимых для получения максимальных оценочных функций. Это позволяет правильно определить связь между вершинами сети, а также направленность данных связей, тем самым установить вероятностную природу ДБС. В рамках статьи разработан принципиально новый подход к обучению структуры ДБС, дающий важные результаты в рамках реализации процедур тестирования и построения эффективных механизмов обнаружения программных ошибок веб-приложений.

Библиографические ссылки

1. *Азарнова Т. В., Азарнова Т. В.* Динамические байесовские сети как инструмент тестирования веб-приложений методом фаззинга // Математические методы распознавания образов: тезисы докладов 19-й Всероссийской конференции с международным участием, г. Москва 2019 г. М. : Российская академия наук, 2019. С. 379–384.

2. *Robinson R. W.* Counting unlabeled acyclic digraphs / R.W. Robinson // Lect. Notes Math, 1977. No 622. Pp. 28-43.

3. Russel S. Artificial Intelligence: A Modern Approach / S. Russel, P. Norvig. Boston: Prentice Hall, 2009. 1095 p.

4. Pearl J. Causality: Models, Reasoning and Inference / J. Pearl. N.Y.: Cambridge University Press, 2009. 484 p.

5. *Дэнис Дж., Шнабель Р.* Численные методы безусловной оптимизации и решения нелинейных уравнений / пер. с англ. М. : Мир, 1988. 440 с.

6. *Heckerman D.* Learning discrete Bayesian networks / D. Heckerman, D. Geiger, D. Chickering // Machine Learning, 1995. Vol. 20. Pp. 197-243.

7. *Chickering D. M.* A Transformational Characterization of Equivalent Bayesian Network Structures / D.M. Chickering // Proc. UAI. NY: Morgan Kaufman, 1995. Pp. 87-98.

8. Spirtes P. Causation, Prediction and Search / P. Spirtes, C. Glymour, R. Sheines. Cambridge: MIT Press, 2000. 568 p.

9. *Вержбицкий В. М.* Численные методы. Линейная алгебра и нелинейные уравнения / В.М. Вержбицкий. М. : Оникс 21 век, 2005. 432 с.

10. *Гилл Ф.*, *Мюррей У., Райт М.* Практическая оптимизация / пер. с англ. М. : Мир, 1985. 509 с.

11. *Васин В. В., Пересторонина Г. Я.* Метод Левенберга – Марквардта и его модифицированные варианты для решения нелинейных уравнений с приложением к обратной задаче гравиметрии // Труды

института математики и механики УрО РАН. 2011. Т. 17, № 2. С. 53–61.

12. *Голуб Дж., Ван Лоун Ч.* Матричные вычисления / пер. с англ. М. : Мир, 1999. 549 с.

13. *Магнус Я. Р., Нейдеккер Х.* Матричное дифференциальное исчисление с приложениями к статистике и эконометрике / пер. с англ. М. : Физматлит, 2002. 496 с.

14. *Гергель В.* Высокопроизводительные вычисления для многоядерных систем. М. : Издательство Московского университета, 2010. 544 с.

15. *Воеводин В. В., Воеводин Вл. В.* Параллельные вычисления. СПб. : БХВ-Петербург, 2002. 608 с.

16. Zaharia M. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing / M. Zaharia, M. Chowdhury, T. Das, A. Dave, M. McCauley, M. Franklin, S. Shenker, I. Stoica // NSDI, 2012. Pp. 1-15.

17. *Таненбаум Э., Ванн Сеен М.* Распределенные системы. Принципы и парадигмы. СПб. : Питер, 2003. 877 с.

References

1. Azarnova T.V., Polukhin P.V. *Dinamicheskie bajesovskie seti kak instrument testirovaniya web prilozhenij metodom fazzinga* [Dynamic Bayesian networks as a tool for testing web applications with fuzzing]. *Intelligent Data Processing: Theory and application*, Moscow Russian Academy of Sciences, 2019, pp. 379-384 (in Russ.).

2. Robinson R.W. Counting unlabeled acyclic digraphs. *Lect. Notes Math*, 1977, No 622, pp. 28-43.

3. Russel S., Norvig P. Artificial Intelligence: A Modern Approach. Boston, Prentice Hall, 2009, 1095 p.

4. Pearl J. Causality: Models, Reasoning and Inference / J. Pearl. N.Y.: Cambridge University Press, 2009. 484 p.

5. Dennis J.E., Schnabel R.B. *Chislennye metody bezuslovnoj optimizacii i resheniya nelinejnyh uravnenij* [Numerical methods for unconstraint optimization and nonlinear equations]. Moscow, Mir, 1988, 440 p. (in Russ.).

6. Heckerman D., Geiger D., Chickering D. Learning discrete Bayesian networks. *Machine Learning*, 1995, Vol. 20, pp. 197-243.

7. Chickering D.M. A Transformational Characterization of Equivalent Bayesian Network Structures. *Proc. UAI*, New York, Morgan Kaufman, 1995, pp. 87-98.

8. Spirtes P., Sheines R. Causation, Prediction and Search. Cambridge, MIT Press, 2000, 568 p.

9. Verzhbickij V.M. *Chislennye metody. Linejnaya algebra i nelinejnye uravneniya* [Numerical methods. Linear algebra and nonlinear equations. Moscow, Oniks 21 vek, 2005, 432 p. (in Russ.).

10. Gil F., Rait M. *Prakticheskaya optimizaciya* [Practical optimization]. Moscow, Mir, 1985, 509 p. (in Russ.).

11. Vasin V.V., Perestoronina G.Ya. *Metod Levenberga-Markvardta i ego modificirovannye varianty dlya resheniya nelinejnyh uravnenij s prilozheniem k obratnoj zadache gravimetrii* [Levenberg-Marquardt method and its modified variants for solving nonlinear equations with application to the inverse problem of gravimetry]. *Trudy instituta matematiki i mehaniki UrO RAN*, 2011, Vol. 17, No 2, pp. 53–61 (in Russ.).

12. Golub J., Van Lou Ch. Дж. *Matrichnye vychisleniya* [Matrix computations]. Moscow, Mir, 1999, 549 p. (in Russ.).

13. Magnus Ya.R., Heidekker H. *Matrichnoe differencialnoe ischislenie s prilozheniyami k statistike i ekonometrike* [Matrix differential calculus with application to statistics and econometrics]. Moscow, Fizmatlit, 2002, 496 p. (in Russ.).

14. Gergel V. *Vysokoproizvoditelnye vychisleniya dlya mnogoyadernyh sistem* [Multi-Core High Performance Computing]. Moscow, Moscow university press, 2010, 544 p. (in Russ.).

15. Voevodin V.V., Voevodin Vl.V. *Parallelnye vychisleniya* [Parallel computation]. Saint-Petersburg, BHV, 2002, 608 p. (in Russ.).

16. Zaharia M., Chowdhury M, Das T, Dave A, McCauley M, Franklin, Shenker S, Stoica I. Fault-Tolerant Abstraction for In-Memory Cluster Computing. *NSDI*, 2012, pp 1-15.

17. Tanenbaum A. Van Steen R. *Raspredelennye sistemy* [Distributed systems]. Saint-Petersburg, Piter, 2003, 877 p. (in Russ.).

* * *

Development of Parallel Algorithms for Probabilistic Models Learning for Web Application Testing

P. V. Polukhin, PhD in Engineering, Voronezh State University, Voronezh, Russia

In the current conditions of testing methods and algorithms development to combine individual test components in the form of a hierarchical model reflecting the connections and states between these components, as well as allowing to evaluate the probability of a transition between model states in the event of information about the successful implementation of a certain test to detect a program error is of particular importance. Existing approaches do not allow optimal adjustment of model parameters, as well as qualitative calculation and establishment of directions of connections between individual components of this model. The scientific study considered the possibility of using effective numerical methods to solve the problem of training probabilistic models built on the basis of Bayesian networks to solve the main problems of testing web applications. The main approaches to create parallel algorithms implementing

the main functional capabilities underlying the implementation of the procedure for training test models are considered. Analysis of the developed training algorithms effectiveness for testing certain groups of program errors allowing the most optimal parameters of the DBS model was made, as well as a justification for their use in distributed data processing systems. We have developed algorithmic solutions for Jacobi and Hesse matrices calculation optimization based on Cannon and Fox algorithms. The process of testing errors of web application is simulated and presented in the form of a dynamic Bayesian network obtained from the results of the structure and parameters learning procedure. The validity of all theoretical results is confirmed by a large number of experimental results proving the validity of the put forward assumptions, testing methods and models presented in the form of dynamic Bayesian networks.

Keywords: probabilistic model, Bayesian networks, Broyden's method, Levenberg-Marquardt method, Bayes-Dirichlet metric.

Получено: 12.05.22