УДК 519.712.3

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ОБУЧЕНИЯ ИСКУССТВЕННОЙ НЕЙРОННОЙ СЕТИ QUICKPROP И RPROP

Крючин О.В.

Научный руководитель – д-р. техн. наук, профессор Арзамасцев А.А.

Тамбовский государственный университет им. Г.Р. Державина

В настоящее время не существует общего алгоритма определения структуры искусственной нейронной сети (ИНС), подходящего для каждой рассматриваемой проблемы. Часто такую структуру выбирают методом «проб и ошибок», который зачастую отнимает у исследователя много времени [2]. Поскольку для каждой структуры необходимо подбирать значения весовых коэффициентов, то ускорение этого подбора должно ускорить весь процесс обучения ИНС.

Одним из способов снижения временных затрат на обучение является использование параллельных алгоритмов. Параллельная программа имеет особый принцип работы, поскольку, в отличии от последовательной, где одновременно выполняется только одно действие, в параллельных программах одновременно выполняются несколько операций. Поскольку выполнение происходит на многопроцессорной машине, то необходимо задействовать все процессоры, то есть дать им на выполнение какую-то часть задачи, которая может выполняться самостоятельно (не задействуя при этом другие части задачи). Таким образом, необходимо добиться того, что бы несколько действий выполнялись независимо друг-от-друга, следовательно проектировать алгоритм с учетом этой особенности. Алгоритм разделяется на блоки, которые одновременно выполняются на разных процессорах, и обмениваются между собой данными. Блоки должны быть достаточно большими, так как в противном случае временные затраты на передачу данных будут больше, чем выигрыш во времени из-за одновременного выполнения блоков. С другой стороны чем больше блоки, тем, как правило, сложнее их выделить, особенно учитывая то, что их выполнение должно занимать одинаковое время (в противном случае получается простаивание процессоров).

Однако, только равномерной загрузки процессоров обычно недостаточно для получения эффективной параллельной программы. Только в крайне редких случаях программа не содержит информационных зависимостей вообще. Если же есть информационная зависимость между операциями, которые при выбранной схеме распределения попадают на разные процессоры, то потребуется пересылка данных. Обычно пересылки требуют достаточно большого времени для своего осуществления, поэтому другой важной целью распараллеливания является минимизация объема и количества необходимых пересылок данных [1]. Таким образом, можно заметить, что произвольный алгоритм не может быть выполнен на параллельной машине, более того не каждый алгоритм может быть переписан для параллельной машины.

На данный момент создано огромное количество алгоритмов эвристического типа, представляющих собой в основном модификацию методов наискорейшего спуска или сопряженных градиентов. Подобные модификации широко известных алгоритмов связаны с внесением в них некоторых изменений, ускоряющих (по мнению авторов) процесс обучения. Как правило, такие методы не имеют серьезного теоретического обоснования, особенно это относится к процедуре подбора управляющих параметров.

Однако в таких алгоритмах реализуется личный опыт работы авторов с нейронными сетями. К наиболее известным эвристическим алгоритмам относятся *QuickProp* С. Фальмана, а также *RPROP* М.Ридмиллера и Х.Брауна [4].

Алгоритмы *QuickProp* и *RPROP* базируются на классическом градиентом алгоритме также называемом алгоритмом наискорейшего спуска, поэтому для разработки параллельных версий эвристических методов необходимо разработать параллельную версию этого алгоритма. Суть метода заключается в вычислении вектора градиента и изменения весовых коэффициентов в направлении антиградиента.

$$\vec{g} = \frac{\partial \varepsilon}{\partial w} = \left(\frac{\partial \varepsilon(w_0)}{\partial w_0}, \frac{\partial \varepsilon(w_1)}{\partial w_1}, \frac{\partial \varepsilon(w_2)}{\partial w_2}, \ldots, \frac{\partial \varepsilon(w_{l_w - 1})}{\partial w_{l_w - 1}} \right) \quad (1)$$

$$g_i^{(k)} = \frac{\partial \varepsilon(w_i^{(k)})}{\partial w_i^{(k)}} = \frac{\varepsilon(w_i + \Delta w_i^{(k-1)}) - \varepsilon(w_i^{(k)})}{\Delta w_i^{(k-1)}} \quad (2)$$

Следовательно, для вычисления g_i необходимо вычислить значение целевой функции при текущем значении весовых коэффициентов, а затем при измененном $w_i = w_i + \Delta w_{i-1}$. После вычисления вектора градиента происходит изменение весовых коэффициентов:

$$w_i^{(k)} = w_i^{(k-1)} + \Delta w_i^{(k)} = w_i^{(k-1)} - s g_i^{(k-1)} = w_i^{(k-1)} - s \frac{\partial \varepsilon(w_i^{(k)})}{\partial w_i^{(k)}} \quad (3)$$

Как можно заметить, для вычисления нового значения одного из весовых коэффициентов используются значения остальных весовых коэффициентов которые были на предыдущей итерации. Исходя из этого можно сделать вывод, что элементы вектора градиентов могут быть вычислены одновременно, следовательно этот вектор можно разделить на n частей (по количеству процессоров), каждая из которых вычисляется на отдельном узле (для этого процессору необходимо лишь передать текущие значения весовых коэффициентов). После окончания вычисления процессоры не возвращают полученные результаты на ведущий, а изменяют значения приписанных к ним весовых коэффициентов, а уже после этого возвращают результат (новые весовые коэффициенты). Следовательно каждый вычислительный узел вычисляет l_w / n весовых коэффициентов. Таким образом, для каждой итерации происходит только две передачи данных – всех весовых коэффициентов на все процессоры и части из них со всех узлов на ведущий [3].

В алгоритме *QuickProp* изменение i-ого весового коэффициента на k-ой итерации производится согласно правилу:

$$\Delta w_i^{(k)} = -s\left(g_i^{(k-1)} + c_w w_i^{(k-1)}\right) + q_i^{(k)} \Delta w_i^{(k-1)} \quad (4)$$

где g_i — элемент вектора градиента, s — коэффициент обучения, c_w — коэффициент минимизации значений весовых коэффициентов, $q_i^{(k)}$ — коэффициент фактора момента. Отличие от классического градиентного метода заключается в наличии двух слагаемых — минимизатора значений весовых коэффициентов ($sc_w w_i^{(k-1)}$) и фактора

момента ($q_i^{(k)}\Delta w_i^{(k-1)}$). Коэффициент минимизации c_w обычно принимает значение 10^{-4} и служит для ослабления весовых связей (вплоть до полного разрыва), а фактор момента необходим для адаптации алгоритма к текущим результатам обучения. Коэффициент q_i уникален для каждого весового коэффициента и вычисляется в два этапа. На первом определяется величина

$$\hat{q}_i^{(k)} = \frac{g_i^{(k-1)}}{g_i^{(k-2)} - g_i^{(k-1)}} \tag{5}$$

а на втором коэффициент момента принимает значение минимальное значение из $\hat{q}_i$ и q_{max} . В качестве значения q_{max} Фальманом предложено 1,75.

Также существует модифицированная форма алгоритма, заключающаяся в уменьшении числа управляющих параметров без потери эффективности. В модифицированном алгоритмике формула (1) заменяется на следующую:

$$\Delta w_i^{(k)} = \begin{cases} q_i^{(k)}\Delta w_i^{(k-1)}, & \Delta w_i^{(k-1)} \neq 0 \\ sg_i^{(k-1)}, & \Delta w_i^{(k-1)} = 0 \end{cases} \tag{6}$$

Таким образом, в случае, когда элемент вектора градиента $g_i^{(k)}$ принимает нулевое значение, на следующей итерации значение соответствующего ему весового коэффициента вычисляется классическим градиентным методом.

Для распараллеливания была выбрана модифицированная версия алгоритма. Как уже было сказано, отличие от классического градиентного метода заключается в том, что в случае, когда элемент вектора градиента не равен нулю, необходимо вычислять коэффициент момента $q = min(\hat{q}, q_{max})$, где $\hat{q}$ вычисляется по формуле (5). Следовательно, кроме текущего значения элемента вектора градиента, необходимо знать его предыдущее значение, а значит принцип распараллеливания, используемый в классическом градиентном методе пригоден и здесь.

Суть метода *RPROP* заключается в игнорировании значения градиента, учитывается исключительно знак. Изменение весового коэффициента вычисляется по формулам

$$\Delta w_i^{(k)} = -\hat{s}_i^{(k)}\theta_g(i) \tag{7}$$

$$\hat{s}_i(k) = \begin{cases} min(q_a\hat{s}_i^{(k-1)}, q_{max}), & g_i^{(k-1)}g_i^{(k-2)} > 0 \\ max(q_b\hat{s}_i^{(k-1)}, q_{min}), & g_i^{(k-1)}g_i^{(k-2)} < 0 \\ \hat{s}_i(k-1), & g_i^{(k-1)}g_i^{(k-2)} = 0 \end{cases} \tag{8}$$

$$\theta_g(i) = \begin{cases} 1, & g_i^{(k-1)} > 0 \\ 0, & g_i^{(k-1)} = 0 \\ -1, & g_i^{(k-1)} < 0 \end{cases} \quad (9)$$

В работе Осовского предложены следующие значения $q_a = 1.2$, $q_b = 0.5$, $q_{min} = 10^{-6}$, $q_{max} = 50$.

Как видно из приведенных выше формул, для вычисления нового значения весового коэффициента необходимо знать: значения градиента на текущей и предыдущей итерации, прошлое изменение весового коэффициента, величину коэффициента обучения (шага) на предыдущей итерации. Отсюда можно сделать вывод, что принцип распараллеливания, используемый для классического градиентного метода и алгоритма *QuickProp*, также применим к этому методу.

Используемая литература

7) Антонов А.С. Введение в параллельные вычисления. М. 2002.
8) Арзамасцев А.А., Крючин О.В., Азарова П.А., Зенкова Н.А. Универсальный программный комплекс для компьютерного моделирования на основе искусственной нейронной сети с самоорганизацией структуры // Вестн. Тамб. Ун-та. Сер. Естеств. и техн. науки. – Тамбов, 2006. Т.11. Вып. 4. с. 564 – 570.
9) Крючин О.В., Арзамасцев А.А, Королев А.Н., Горбачев С.И., Семенов Н.О. Универсальный симулятор, базирующийся на технологии искуственных нейронных сетей, способный работать на параллельных машинах // Вестн. Тамб. Ун-та. Сер. Естеств. и техн. науки. – Тамбов, 2008. Т.13. Вып. 5. с. 372 – 375.
10) Осовский С. Нейронные сети для обработки информации. М.: Финансы и статистика, 2002, 344 с.