РАСПРЕДЕЛЕННЫЙ ЗАПУСК НЕЙРОННЫХ СЕТЕЙ НА МНОЖЕСТВЕ ВЫЧИСЛИТЕЛЬНЫХ УЗЛОВ

С.Д. Ионов

Введение

В ходе исследования нейронных сетей была построена структура *воспроизводящейся искусственной нейронной сети* (ВИНС), моделирующаяся ассоциативную ячейку памяти. Предложенная ячейка памяти способна увеличивать свою емкость при поступлении на ее входы новых сигналов. Для решения актуальных практических задач на значительном множестве различных входных сигналов и для изучения поведения получаемой ВИНС необходимо обрабатывать сотни тысяч нейронов. Для запуска сети такой мощности недостаточно одного вычислительного узла – персонального компьютера. Таким образом, необходимо решить задачу распределения нейронной сети по множеству вычислительных узлов.

Задача распределения нейронной сети

Основная задача распределения вычислительной нагрузки формулируется как *двухуровневая задача о назначениях*. В нашем случае она формулируется следующим образом.

Заданы два графа: $G_N=(V_N,E_N)$ – граф нейронной сети и $G_P=(V_P,E_P,c_P)$ – граф сети физических узлов. Для графов определены: V_P и V_N – соответствующие множества вершин, E_P и E_N – множества ребер, c_P – весовая функция вершин графа, определяющая производительность отдельного узла, $e_P: V_P \times V_P \to \{0,1\}$ и $e_N: V_N \times V_N \to \{0,1\}$ – функции ребер для графов G_P и G_N соответственно, имеющие вид (1).

$$e_Q(v_1,v_2)=\begin{cases}1, если\ (v_1,v_2)\in E_Q\\ 0, если\ (v_1,v_2)\notin E_Q\end{cases} \quad (1),\ \text{где}\quad Q=\overline{N,P}$$

Необходимо построить функцию отображения $\varphi: V_N \to V_P$, чтобы выполнялись условия:

1. Все ребра нейронной сети лежат или на ребрах между физическими узлами, или внутри узлов.

$$\forall v_1^N, v_2^N \in V_N, e_N(v_1^N, v_2^N)>0: \begin{bmatrix}\varphi(v_1^N)=\varphi(v_2^N)\\ e_P(\varphi(v_1^N),\varphi(v_2^N))>0\end{bmatrix} \quad (2)$$

2. Число ребер нейронной сети, проходящих по ребрам между физическими узлами, минимально.

$$\sum_{v_1^N, v_2^N \in V_N} e_P(\varphi(v_1^N),\varphi(v_2^N))\, e_N(v_1^N, v_2^N)>0 \to min \quad (3)$$

3. Число размещенных по узлам нейронов пропорционально производительности этих узлов.

$$\sum_{v^P \in V_P} \psi_\varphi(v^P)\, c_P(v^P) \to max \quad (4),$$

где $\psi_\varphi=\begin{cases}\dfrac{1}{\xi_\varphi}, \xi_\varphi>0\\ 0\end{cases}, \xi_\varphi: V_P \to N=\left|\{v^N \in V_N: \varphi(v^N)=v^P\}\right| \geq 0$

В общем случае получаем NP-трудную задачу [1].

Попытаемся найти приближенное решение этой задачи. Прежде всего, обратимся ко второму условию. Чтобы выполнить это условие, нужно разбить граф нейронной сети на классы так, чтобы их количество было равно числу узлов, а количество связей между образовавшимися классами было минимально. Для этого нужно найти $k-1$ минимальных разрезов графа, где k – число физических узлов.

Поиск будем производить итерационно. На первом шаге алгоритма принимаем все вершины графа одним классом. Затем на каждой итерации для каждого класса находим минимальный разрез. Среди множества полученных разрезов выбираем наименьший и заменяем класс с этим разрезом двумя классами, получаемыми в результате применения разреза. Для оптимизации алгоритма разрезы остальных классов шага временно сохраняются и повторно не пересчитываются на следующем шаге. В итоге получаем алгоритм, выполняющийся за k шагов.

Самой значимой частью каждого шага является поиск минимального разреза в графе. Для поиска произвольного минимального разреза используется современный алгоритм Матильды Штор и Френка Вагнера [2]. Алгоритм создан на основе алгоритма Нагамоки и Ибараки [3] и имеет вычислительную сложность $O(nm+n^2\log(n))$, где n – количество узлов нейронной сети, m – количество связей нейронной

сети. Алгоритм не использует в своей основе алгоритма поиска максимального потока, а оперирует множествами вершин графа. Кроме того алгоритм позволяет находить произвольный минимальный разрез, равный минимуму среди разрезов минимальной стоимости по всевозможным парам исток-сток в рамках задачи максимального потока. Таким образом, нам не нужно решать задачу поиска минимального разреза для каждой пары исток-сток, что потребовало бы $O(n^2)$ запусков.

Согласно алгоритму итеративно выполняем следующий процесс: находим минимальный разрез между какой-нибудь парой вершин s и t, а затем объединяем эти две вершины в одну. В конце после $n-1$ итерации граф объединится в единственную вершину и процесс остановится. После этого решением будет являться минимальный среди всех $n-1$ найденных разрезов. И действительно, на каждой i-ой стадии найденный минимальный разрез C_i между вершинами s_i и t_i либо окажется искомым глобальным минимальным разрезом, либо же, напротив, вершины s_i и t_i невыгодно относить к разным множествам, поэтому мы ничего не ухудшаем, объединяя эти две вершины в одну.

Таким образом, осталось лишь решить самую интересную часть задачи: на каждом шаге алгоритма для текущего графа найти минимальный разрез между произвольной парой вершин s и t. Для решения этой задачи был предложен следующий, тоже итеративный процесс. Вводим множество вершин A, которое изначально содержит единственную произвольную вершину. На каждом шаге к множеству A добавляется вершина, наиболее сильно связанная с множеством A, т. е. вершина $v \notin A$, для которой максимально значение функции $w(v, A)$, заданной формулой (5).

$$w(v, A) = \sum_{(v,u)\in E, u\in A} c(v, u) \quad (5),$$ где $c(v, u)$ – вес ребра (v, u)

В итоге на $n-1$ шаге итерации будет получен минимальный разрез между s и t, равный $w(t, A \setminus t)$.

Изначально асимптотика алгоритма была кубической, т. к. поиск наиболее сильно связанной вершины выполнялся за $O(n)$. Для улучшения алгоритма авторами было предложено использование Фибоначчиевых куч, позволяющих увеличивать значение хранимых ключей за $O(1)$ в среднем и извлекать максимум за $O(\log n)$ в среднем, что дает итоговую асимптотику $O(nm + n^2 \log(n))$.

После получения k групп вершин графа нейронной сети можно приступить к размещению. Так как разрез между группами минимальный, значит, связность между группами минимальная, что выполняет условие 2. Расположим самую многочисленную группу на узел с наибольшей производительностью для обеспечения условия 3. Затем будем размещать остальные группы по узлам, сохраняя условие 1. В частности, это означает, что если поместить группу на следующий узел невозможно без нарушения этого условия, то она помещается на текущий вне зависимости от условия 3. Таким образом, вычислительная сложность предложенного решения $O(knm + kn^2 \log(n))$.

Описанный алгоритм размещения позволяет достичь оптимального соответствия нейронов узлам для большинства задач. Хотя, конечно, можно привести такую постановку, при которой в результате все узлы будут размещены на некотором одном узле, что будет крайне не оптимально с точки зрения производительности и фактически даст вместо распределенной нейронной сети обычную сеть на одном вычислителе. Примером может быть изображенная на рис. 1 конфигурация физической сети, в которой нейронная сеть, распределяемая предложенным алгоритмом вероятнее всего займет лишь одну ветку физических узлов.

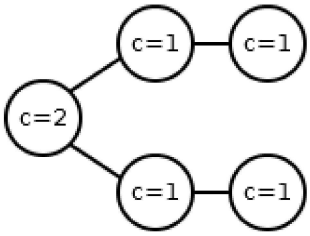

Рис. 1 Физическая сеть узлов с указанием значения весовой функции производительности каждого узла

Стоит отметить, что в постановке задачи о назначениях мы не учли условие производительности каналов связи между узлами, при котором функция ребер графа G_P будет иметь вид $e_P : V_P \times V_P \to R$,

описывая скорость передачи данных в канале связи. В этом случае изменяется условие 2: число ребер нейронной сети, проходящих по ребрам между физическими узлами, пропорционально производительности каналов связи соответствующих ребер. При этом задача усложняется, так как приходится учитывать как параметры узлов, так и связей между ними.

Выполнение распределенного запуска

На практике для выполнения запуска используется специально разработанный сервер [4]. Он позволяет принимать в многопоточном режиме произвольные TCP запросы и перенаправлять их на подключаемые модули логики обработки сообщений. Одним из таких модулей является модуль нейронной сети.

Работу модуля нейронной сети предлагается регламентировать протоколом Distributed Neural Network Configuration (DNNC). Протокол включает в себя три схемы XSD, описывающие обрабатываемые модулем пакеты данных. Схема конфигурации физической и нейронной сети описывает каждый узел системы и состоит из двух основных частей: часть, описывающая структуру нейронной сети, и часть, описывающая структуру физической сети. Для каждого отдельного физического узла из общей схемы настройки сети можно выделить конфигурацию в соответствии со схемой отдельного физического узла. В этой схеме описываются все нейроны, которые необходимо разместить на текущем узле. Схема передачи данных между нейронами описывает формат XML пакета для передачи данных от множества нейронов одного физического узла к множеству нейронов другого, причем под данными понимается некоторое число с плавающей запятой. Согласно этой схеме физические узлы могут группировать множества исходящих сигналов от нейронов по направлениям их отправки, благодаря чему уменьшается нагрузка на сеть.

На начальном этапе запуска все узлы считаются равнозначными. Пользователю необходимо определить один из узлов основным, так как все взаимодействие и управление сетью будет выполняться через этот узел. Выбор выполняется передачей полной настройки сети в запросе к узлу. Настройки представляют собой XML файл согласно схеме конфигурации физической и нейронной сети протокола DNNC [5]. Согласно указанным в этом файле параметрам основной узел выполняет размещение узлов нейронной сети по узлам физической сети по приведенному выше алгоритму. Затем согласно [5] для каждого физического узла сети основной узел создает и рассылает индивидуальный файл настроек. После этого нейронная сеть считается запущенной и готовой к обработке входных данных.

Стоит отметить, что в процессе рассылки индивидуальных настроек возникает дополнительная задача оптимизации сетевого взаимодействия основного узла с остальными узлами . Решением этой задачи является применение метода рассылки по дереву узлов. В этом методе основной узел выделяет из множества узлов несколько дочерних и отправляет им настройки как самих этих узлов, так и настройки узлов их подсетей. Затем каждый дочерний узел начинает выполнять роль основного для своей подсети и повторяет рассылку настроек. Таким образом, распределяется процесс рассылки настроек, что в целом уменьшает нагрузку на основной узел и распределяет нагрузку по каналам связи. Для применения этого метода необходимо расширить имеющийся протокол [5] в части схемы конфигурации отдельного физического узла так, чтобы в XML передавалась настройка подсети с указанием физических узлов и уже привязанных к ним нейронов.

Во время основного этапа работы системы входные и выходные данные нейронной сети доступны для считывания или передачи на основном узле. Остальная часть сети пользователю недоступна для взаимодействия. Между узлами обмен выполняется по протоколу [5] в соответствии со схемой передачи данных.

В виду заложенного распределения нейронной сети по отдельным вычислительным узлам, а также предполагаемой разнородности производительности узлов возникает задача их синхронизации при обработке проходящих по сети данных. Главная проблема, которую нужно решить, – это как долго каждый отдельный нейрон должен ожидать сигналов по всем своим входам. В качестве одного из решений можно было бы реализовать нейронную сеть так, чтобы она пересылала все сигналы, включая нулевые, но даже в этом случае может возникать проблема скорости обмена, когда некоторая часть сети работает в несколько раз быстрее, а значит, ее работу нужно приостанавливать и ожидать данные с самого медленного узла. Что еще важнее, такое решение не учитывает возможность выхода из строя физического узла, приводящего к полной остановке всей системы.

Предлагается применять вариант с оценкой допустимых задержек при передаче данных. Для выполнения оценки необходимо в момент запуска при начальной настройке сети определить возможности каждого из узлов, а также каналов связи между ними. Чтобы проанализировать состояние физической сети узлов и определить временные задержки на соединениях, построим граф переходов. Пусть $G_P=(V_P, E_P)$ – граф физической сети. Построим отображение $\varphi: G_P \to G_M$ такое, что $G_M=(V_M, E_M)$ – граф переходов. Множества вершин и ребер графа G_M описываются формулами (6) и (7).

$$V_M=\left\{v^M : v^M=\varphi(v_1^P, v_2^P), (v_1^P, v_2^P)\in E_P\right\} \quad (6)$$

$$E_M=\left\{(v_1^M, v_2^M): v_1^M, v_2^M\in V_M, \exists v^P\in V_P : v^P\in\varphi^{-1}(v_1^M), v^P\in\varphi^{-1}(v_2^M)\right\} \quad (7)$$

Т. е. в вершинах V_M хранятся описатели соединений E_P изначальной сети, а ребра E_M опираются на узлы V_P и строятся так, чтобы каждое соединение v^M, построенное на основе ребер из E_P некоторого узла v^P из V_P, было соединено ребрами из E_M со всеми остальными соединениями v^M на ребер из E_P этого узла v^P.

Во время построения в изначальный граф G_P добавляется дополнительный узел, связанный только с основным узлом при помощи дополнительного ребра. Построенная по этому дополнительному ребру вершина графа G_M считается начальной для алгоритма поиска. Далее по алгоритму поиска в ширину определяется путь от начальной вершины до каждой вершины графа G_M, обозначающей оцениваемое соединение. Таким образом, каждая вершина в построенном пути обозначает соединение в изначальном графе G_P, по которому пакет должен пройти. Для вычисления времени движения пакета в сервер добавлена специальная логика обработки пакетов, выполняющая задачу маршрутизации этих пакетов по узлам согласно списку узлов, указанному в самом пакете. Список формируется согласно цепочке переходов, определяемой $\varphi^{-1}(p)$, где p – построенный путь в графе G_M, и содержит путь до целевого удаленного узла и обратно до основного. Время перехода $t_{1,i}$ по ближайшему от основного узла соединению вычисляется как половина среднего времени движения пакета по соответствующему маршруту.

$t_{1,i}=T_{1,i\to i,1}/2$, где $T_{1,i\to i,1}$ – среднее время движения пакета от узла 1 до i и обратно.

Времена остальных соединений определяются аналогично с вычетом времен пройденных маршрутов соединений по формуле (8).

$$t_{k,j}=\left(T_{1,j\to j,1}-T_{1,k\to k,1}\right)/2 \quad (8)$$

Таким образом, основной узел получает сведения о состоянии остальной сети физических узлов. Согласно полученным временам каждому нейрону устанавливается задержка – временной интервал между приходом первого сигнала и перевода полученных сигналов в фазу пересчета нейрона. При этом все входы, не получившие каких-либо данных считаются нулевыми. В случае такого устройства взаимодействия у нас появляется выбор между двумя режимами работы: либо каждый нейрон отправляет свое состояние всегда, даже в случае нуля, либо только при ненулевом состоянии. Выбор режима работы зависит от состояния физической сети, так как разница в подходах определяется или большей нагрузкой на физическую сеть между узлами, или временными затратами на ожидание сигналов при работе нейронной сети.

Перспективы и проблемы

Предложенный выше алгоритм обеспечивает распределенный запуск нейронной сети на кластере из десятков узлов. Имеется его реализация в виде распределенного сервера [4] для запуска в кластерной гомогенной среде. При этом есть ряд ограничений и проблем, которые требуют дальнейшего развития системы распределенного запуска.

В первую очередь, так как мы исследуем ВИНС – такой тип нейронной сети, который в ходе своей работы изменяет свою структуру изнутри, перестраивая связи и наращивая новые нейроны, необходима возможность перераспределения нейронной сети по физическим узлам в условиях динамического изменения конфигурации системы. Стоит заметить, что такой алгоритм не только позволит запускать нейронные сети с изменяющейся структурой, но и в целом организовывать распределенный запуск в условиях нестабильной физической сети узлов. Если реализовать систему, динамически распределяющую свой нагрузку по изменяющимся узлам, то открываются возможности запуска вне кластерной гомогенной среды. В этом случае задача распределения усложняется, так как придется учитывать не только производительность узла, но и канал связи с узлом. Запуск в гетерогенной среде также подразумевает еще одно важное ограничение – некоторые узлы системы могут быть недоступны напрямую для взаимодействия с основным узлом. Доступ к таким узлам требует введения маршрутизации и организации дополнительных основных узлов управляющих отдельными подсетями физических узлов. В качестве дополнительных основных узлов могут подойти предложенные ранее в методе рассылки по дереву узлов дочерние узлы, но при этом выбор дочерних узлов должен быть более детерминированным в части доступности их подсетей остальной части системы.

Кроме технических задач также необходимо решить ряд логических требований, обеспечивающих управление ходом работы нейронной сети. В рамках исследования организации структур ВИНС требуется иметь возможность посмотреть текущее состояние всей структуры нейронной сети в каждый момент времени. Для этого нужно решить две задачи: возможность единовременной "заморозки" всей нейронной сети и возможность выгрузки состояния нейронной сети в некоторый визуализируемый формат. Решение задачи "заморозки" на данном этапе видится как введение в структуру физической сети дополнительного тактового генератора. Рассылаемые им пакеты инициируют процессы вычисления нейронов на каждом узле. В случае остановки тактового генератора нейронная сеть должна завершить все операции своего последнего такта и приостановить работу. Для выгрузки состояния нейронной сети в приостановленном состоянии требуется

расширить протокол возможностью опросить каждый физический узел с основного узла, чтобы получить на нем изменившееся состояние всей нейронной сети, причем кроме множества нейронов и состояния их связей необходимо получать текущие значения состояния нейронов, с которыми нейронная сеть продолжит работу после запуска тактового генератора. Наличие выгрузки порождает возможность загрузки сохраненного состояния в систему с тем, чтобы можно было продолжить работу нейронной сети с любого сохраненного состояния. При этом совсем не обязательно выполнять загрузку на ту же физическую систему, т. к. достаточно изменить настройку блока физической сети для нового распределенного запуска.

ЛИТЕРАТУРА:

1. Р.М. Ларин, А.В. Пяткин "Двухуровневая задача о назначениях" // Дискретн. анализ и исслед. опер., сер. 2, 8:2 (2001), 42–51
2. M. Stoer, F. Wagner "A Simple Min-Cut Algorithm" // 1997
3. H. Nagamochi, T. Ibaraki "Linear time algorithms for finding a sparse k-connected spanning subgraph of a k-connected graph" // Algorithmica 7, 583–596. 1992
4. "Сервер распределенного запуска нейронных сетей" http://code.google.com/p/axis4-neural-server/ // 2010
5. "Протокол DNNC" http://research.axis4.ru/files/dnnc.html // 2010