ОТКРЫТАЯ ИНФРАСТРУКТУРА ДЛЯ РАСПРЕДЕЛЁННОГО МОДЕЛИРОВАНИЯ НЕЙРОННЫХ СЕТЕЙ

А.А. Краснощёков

АННОТАЦИЯ

В данной работе проводится анализ и описывается построение инфраструктуры для распределённого моделирования нейронных сетей.

Данная система призвана снизить экономические вложения при проведении сложного моделирования нейронных сетей за счёт коллективного использования ресурсов вычислительных сетей и концепции волонтёрства в GRID вычислениях.

ВВЕДЕНИЕ

Искусственные нейронные сети

На сегодняшний день, искусственные нейронные сети (ИНС), получили широкое распространение при решении задач, для которых обычные алгоритмические решения оказываются неэффективными или вовсе невозможными [1]. Искусственные сети, подобно биологическим, являются вычислительной системой с огромным числом параллельно функционирующих простых процессоров и множеством связей.

ИНС успешно применяются для решения широкого круга задач, например: распознавание образов, классификация, принятие решений, адаптивное управление, кластеризация, прогнозирование, аппроксимация, сжатие данных и ассоциативная память.

Вычислительные алгоритмы, основанные на аппарате искусственных нейронных сетей, обладают следующими концептуальными характеристиками (присущими биологическим нейронным сетям):

- Распределённое представление информации и вычисления;
- Массовый параллелизм;
- Способность решать неформализованные задачи;
- Контекстуальная обработка информации;
- Адаптивность;
- Способность к обучению и обобщению;
- Толерантность к ошибкам;
- Отказоустойчивость;

За время существования нейронной доктрины - с момента открытия биологического (Santiago Ramón y Cajal, 1933) , а затем искусственного (McCulloch, W. and Pitts, W., 1943) нейрона до настоящего времени, аппарат нейронных сетей претерпел колоссальное развитие. Сегодня существует большое число различных конфигураций нейронных сетей, ориентированных на решение самых разных задач [2]. Но несмотря на хорошо развитую теоретическую базу, на практике, применение нейронных сетей сталкивается с рядом неразрешённых проблем.

Проблема обучения

Одним из самых важных аспектов аппарата нейронных сетей является их обучение. ИНС не программируются в классическом понимании этого слова, а обучаются, адаптируются к среде. Для того чтобы наглядно описать этот процесс приведём цитату из блога Томаса Лаора [3]:

«Процесс обучения нейронной сети может быть рассмотрен как изменение формы листа металла, который представляет выход (диапазон значений) аппроксимируемой функции. Набор тренировочных данных выступает в роли энергии, требуемой для сгиба листа так, чтобы он прошёл через заданные точки. Но металл по своей природе будет сопротивляться таким изменениям. Таким образом нейронная сеть попытается найти конфигурацию с минимальным потреблением энергии, которая удовлетворяет ограничениям (соответствует набору тренировочному данных).»

На сегодняшний день существует множество развитых алгоритмов для обучения нейронных сетей, но вместе с этим разнообразием, перед исследователем возникает задача выбора оптимального алгоритма. Многие алгоритмы имеют медленную сходимость, попадают в локальные минимумы и имеют вычислительную сложность O(n2) и O(n4) [2].

Проблема подбора параметров

Проблемы при практической реализации нейронных сетей возникают не только на стадии обучения. При проектировании нейронной сети необходимо определиться со значениями различных свободных параметров. Среди них : число слоёв, число нейронов в каждом слое, процентное соотношение обучающих и

тестовых данных, количество обучающих циклов и т. д.. В рамках обучения ИНС необходимо определить количество скрытых нейронов, скорость обучения и множество других параметров.

Зачастую все эти параметры неизвестны априори. Определить оптимальные значения удаётся только эмпирически, путём применения методов оптимизации или простого перебора значений.

Применение методов оптимизации в масштабе всей ИНС приводит к многократному увеличению вычислительной нагрузки (а вместе с ней увеличению времени моделирования). В случаях, когда оптимизация не применима или не даёт желаемого эффекта, использование простого перебора значений параметров увеличивает вычислительную нагрузку в раз, где n — количество параметров, - интервал значение перменной pi , di — интервал дискретизации переменной.

Проблема высокой размерности

За последние десятилетия, доступность дешёвых и эффективных устройств накопления информации, и информационных систем в целом, вызвали быстрое распространение реляционных, графических и текстовых баз данных. Сбор и хранение информации стали значительно проще, но вместе с этим усложнилось её получение.

Стремительное развитие информационных технологий, также вызвало необходимость обработки больших объёмов информации в реальном времени (обработка потокового видео, анализ спутниковых изображений, и т.д.).

Для интеллектуального анализа таких данных применяются нейронные сети большой размерности, которые, в свою очередь требуют высокопроизводительных вычислительных ресурсов.

Примером такой задачи может служить прогнозирование дорожного трафика мегаполиса или прогнозирование климатических изменений, где количество нейронов сети может измеряться тысячами или десятками тысяч [4,5].

Распределённые вычисления

Проведя анализ современных проблем построения и обучения ИНС на практике, можно утверждать, что все перечисленные проблемы могут быть решены наращиванием производительности вычислительной системы (supercomputing) или применением концепции распределённых вычислений (metacomputing).

В данной работе мы фокусируемся на концепции распределённых вычислений, так как с ростом производительности отдельной системы её цена увеличивается экспоненциально, а при увеличении производительности распределённой системы — линейно [6].

ПОСТАНОВКА ЗАДАЧИ

Для построения распределённой системы необходимо, прежде всего, исследовать принципы функционирования современных распределённых систем, а затем сопоставив их задачам моделирования нейронных сетей выработать концепцию построения системы.

Особенности распределённых вычислений:

- Повсеместный доступ;
- Надёжность;
- Масштабируемость;
- Виртуализация;
- Взаимозаменяемость узлов;
- Независимость от географического местоположения;
- Эффективное соотношения цена/качество;

Новые парадигмы зачастую ассоциируются с стационарной электросетью, которая предоставляет общий доступ к электроэнергии и которая оказала огромное влияние на индустриальное развитие.

Параллельное вычисление ИНС

При обучении нейронных сетей выделяют два типа методов обучения: детерминистские и стохастические.

Детерминистский метод обучения шаг за шагом осуществляет процедуру коррекции весов сети, основанную на использовании их текущих значений, а также величин входов, фактических выходов и желаемых выходов. Обучение персептрона является примером подобного детерминистского метода.

Стохастические методы обучения выполняют псевдослучайные изменения величин весов, сохраняя те изменения, которые ведут к улучшениям. К стохастическим методам относится целый класс обучающих алгоритмов - эволюционные алгоритмы. Существуют несколько разновидностей эволюционных алгоритмов: генетические алгоритмы, эволюционное программирование, эволюционные стратегии, системы классификаторов, генетическое программирование. Все они моделируют базовые положения в теории биологической эволюции — процессы отбора, мутации и воспроизводства. Поведение агентов определяется окружающей средой. Множество агентов принято называть популяцией. Такая популяция эволюционируют в соответствии с правилами отбора в соответствии с целевой функцией, задаваемой окружающей средой. Таким

образом, каждому агенту (индивидууму) популяции назначается значение его пригодности в окружающей среде. Размножаются только наиболее пригодные виды. Рекомбинация и мутация позволяют изменяться агентам и приспособляться к среде.

В то время как детерминистские методы обучают нейронную сеть на уровне отдельных элементов (синапсов), стохастические методы, а правильнее методы глобальной оптимизации занимаются обучением ИНС на структурном уровне.

Концепция системы

Основной задачей данного исследования является разработка архитектуры и реализация открытой инфраструктуры для поддержки научных вычислений. В данном разделе мы определим концептуальные моменты архитектуры системы в тезисной форме.

1. Общедоступность

a) Задача. Система как услуга.

Исследователь может проводить моделирование нейронных сетей, даже при отсутствии компьютерной техники и программного обеспечения.

Решение. Должен быть реализован централизованный механизм распределённых вычислений с предоставлением интуитивно - понятного web-интерфейса и документированного API.

b) Задача. Открытый код.

Так как система несёт некоммерческий характер и открыта для дальнейшего развития, весь код и документация должны распространяться под открытой лицензией.

Решение. При разработке системы должны быть использованы открытые стандарты. Должен быть создан публичный репозиторий программы и форум сообщества разработчиков и пользователей. Все используемые программные компоненты в системе должны быть свободно распространяемыми.

c) Задача. Общедоступность.

Для ограничения доступа в Интернет в большинстве корпоративных сетей используются прокси-серверы, файерволы и прочие программы, обеспечивающие безопасность. Зачастую несанкционированный доступ к нестандартному удалённому порту или использование нестандартного (не разрешённого политикой сети) протокола передачи данных будет блокировано. Чтобы обеспечить доступность системы из любой корпоративной сети необходимо использовать стандартные методы и протоколы.

Решение. Система должна реализовывать концепцию REST и обмениваться данными преимущественно по протоколу HTTPS. Это обеспечит доступ к системе из любой существующей корпоративной сети, а также с мобильных устройств, имеющих доступ в Сеть.

d) Задача. Простота использования

Многие учёные, занимающиеся исследованием нейронных сетей не имеют серьёзного опыта в программировании, поэтому при разработке следует использовать язык высокого уровня, скриптинговый язык или мета-язык программирования.

e) Решение. На сегодняшний день существует великое множество языков программирования, и выбрать подходящий не составит труда.

2. Функциональность

a) Задача. Быстродействие

Учитывая большой объём операций в системе, они должны быть атомарны и должны производиться с максимальным быстродействием.

Решение. Должен быть использована СУБД для обеспечения конкурентности вызовов, атомарности транзакций и восстановления после сбоев. СУБД должна функционировать как OLTP система так как в режиме стандартного функционирования будут преобладать запросы на вставку, изменение и удаление данных.

b) Задача. Нейросетевая инфраструктура

Должна быть разработана инфраструктура для быстрого создания ИНС.

Решение. Следует разработать инфраструктуру на основе ООП и высокопроизводительных библиотек работы с матрицами.

c) Задача. Визуализация

Должен быть разработан модуль для визуализации результирующих данных.

Решение. Возможно как написание такой функциональности, так и использование готовых библиотек.

d) Задача. Модульность

Система должна иметь модульную структуру, для прозрачного обновления или отладки отдельных компонентов.

Решение. Данное требование может быть реализовано средствами ЯП.

e) Задача. Привлечение волонтёров

Для привлечения и мотивации участников проекта необходимо разработать условную систему тарификации вычислений.

Решение. В систему будет интегрирована гибкая система баллов и приоритетов, в первую очередь система будет вычислять задачи наиболее активных волонтёров.

f) Задача. Административный интерфейс

Администратор системы должен иметь выделенный интерфейс с полным доступом ко всем элементам системы.

Решение. Должен быть создан административный интерфейс.

3. Масштабируемость

a) Задача. Волонтёрские вычисления

Любой желающий может включить свой компьютер в систему.

Решение. Должен быть создан клиентский программный модуль, позволяющий оперативно включать новые узлы в работу системы.

b) Задача. Кроссплатформенные вычисления

Система должна обеспечить поддержку любых операционных систем как на стороне клиента, так и на стороне сервера.

Решение. При разработке системы должен быть использован один из кроссплатформенных языков программирования.

c) Задача. Децентрализованная серверная группа системы

Серверные компоненты системы должны легко переноситься и взаимодействовать на различных серверах в пределах одной ЛВС.

Решение . Серверная часть системы должна состоять из нескольких независимых серверов и реализовывать архитектуру «3-tier»

d) Задача. Мигрирующий код.

Так как число клиентских узлов системы принципиально не ограничено сверху, возникает вопрос об обновлении кода и библиотек на каждом узле.

Решение. В системе должна быть реализована система контроля версий и обновлений программного кода как на узлах, так и на серверной группе из одного централизованного репозитория.

4. Надёжность и безопасность

a) Задача. Регистрация пользователей.

Набор функций системы должен быть доступен только для зарегистрированных пользователей.

Решение . Web-интерфейс должен предусматривать личный кабинет пользователя.

b) Задача. Защищённый обмен данными

Данные пользователя являются охраняемой государством информацией, поэтому должны быть приняты меры по обеспечению защищенного обмена данными

Решение . Будет использован криптостойкий протокол HTTPS

c) Задача. Достоверность узлов

Необходим механизм аутентификации узлов в системе

Решение. Каждому пользователю будет выдаваться персональный ключ, с помощью которого он сможет активировать конкретный узел, который в свою очередь также получает уникальный ключ — GUID.

d) Задача. Избыточность

Для избежания потери информации и зависаний системы за счёт гетерогенной структуры Сети необходимо распространять задания одновременно на несколько узлов сети.

Решение . Подзадача будет раздаваться n узлам по принципу «победитель получет всё»

Принимая во внимание все вышеописанные принципы построения системы мы приходим к GRID системе на основе централизированной группы серверов и множества клиентских узлов обменивающихся информацией с центральной группой посредством протокола HTTPS.

Далее будет рассмотрен процесс построения системы, описаны принципы её функционирования и обозначены дальнейшие направления её развития.

РЕШЕНИЕ

Выбор технологий

В качестве языка реализации системы был выбран кроссплатформеный язык Python .

Python - это язык программирования высокого уровня и общего назначения [8]. Базовый синтаксис и семантика Python'a минималистичны, в то время как стандартные библиотеки огромны и сложны. Python поддерживает несколько парадигм программирования (ООП, императивное и функциональное), и имеет такие особенности, как: полностью динамическая система типов и автоматическое управление памятью.

Python разрабатывался как легко читаемый язык. Его парадигма: "Должен быть только один, и лучше всего один, очевидный способ сделать это" - из чего следует, что код, написанный одним разработчиком может легко развиваться и поддерживаться другим. Также, Python "навязывает" программистам дисциплину, использованием отступов и синтаксисом кода, что позволяет легко поддерживать крупные приложения.

Python идеально подходит для реализации нашей задачи за счёт следующих факторов:

1. Универсальность (подходит для решения любых задач)
2. Кроссплатформенность
3. Широкое распространение

4. Обширные стандартные библиотеки
5. Стилистика кода (лаконичный, хорошо читаемый код)
6. Высокая производительность программиста

Так как обязательным условием в построении системы является наличие web-интерфейса и применение концепции ORM, мы использовали для разработки фрэймворк Django [9]

Django Framework— набор библиотек языка python, котрый реализует модель «Модель-Шаблон-Вид» (MTV). В нашем проекте Django предоставляет следующие ресурсы:

- ORM для доступа к БД
- встроенный интерфейс администратора
- диспетчер URL на основе регулярных выражений
- расширяемую систему шаблонов с тегами и наследованием

Архитектура системы

Система состоит из 5ти связанных проектов Рис 1.:

- Web-сервер отвечает за предоставление информационного сайта.
- Task-сервер отвечает за предоставление и сбор подзаданий.
- cheduler реализует функции map()/reduce()
- Client - проводит вычисления на узле системы.
- ANN Infrastructure - инфраструктура для моделирования ИНС.

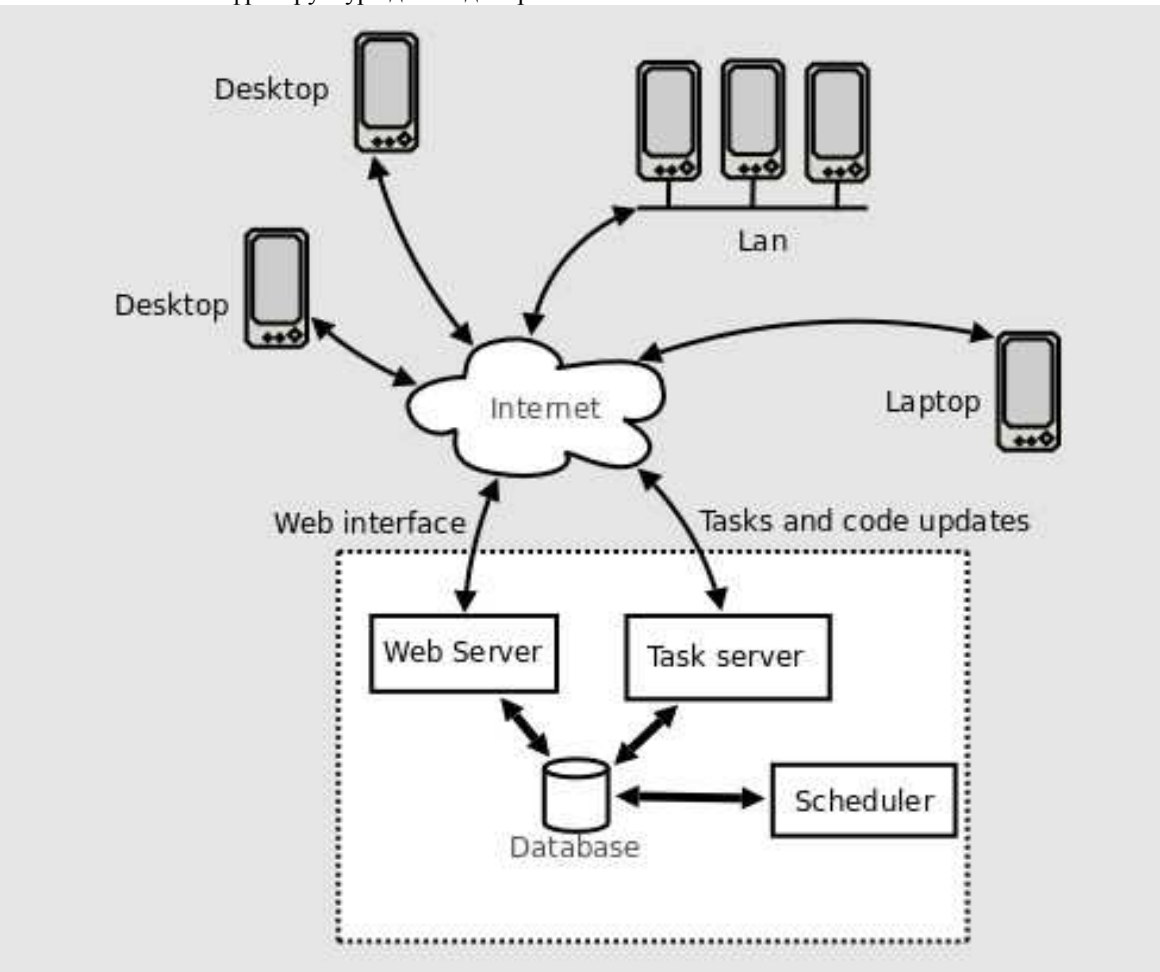

Рис 1. Архитектура системы pythonstrike

Алгоритм вычислений

1. Пользователь проходит регистрацию в системе.
2. С помощью web-интерфейса создаёт новый набор данных и новое задание.Задание заносится в БД.
3. Планировщик системы выбирает неразмеченное задание и вызывает функцию map() соответствующего заданию модуля. Тем самым создавая n подзадач в БД.
4. Клиенты системы попеременно получают подзадачи, обрабатывают их и отправляют их обратно в БД по протоколу HTTPS.

5. Каждые t секунд планировщик проверяет наличие присланых подзадач, и если они имеются в базе, вызывает функцию reduce() соответствующего заданию модуля.
6. Reduce() собирает информацию из присланых подзадач и сохраняет её в основном задании.
7. Если выполняется многошаговая операция (генетический алгоритм), то задание снова размечается и повторяется шаг 4.
8. В противном случае результат выполнения сохраняется в БД.
9. В личном кабинете пользователя появляется уведомление о получении новых результатов.

Дальнейшее развитие проекта

Сейчас система находится в альфа-версии и разработчикам предстоит проделать ещё немало работы по её развитию. В ближайшей перспективе будут проведены следующие разработки:

- адаптация системы в ОС Windows;
- применение генетических алгоритмов при обучении LSTM;
- увеличение быстродействия системы за счёт вставки кода на C;

Вы можете ознакомиться с программным кодом и материалами статьи по адресу http://code.google.com/p/python-strike/.

ВЫВОДЫ

В результате проведённого исследования:

1. Проведён анализ возможностей применения механизма распределённых вычислений, для моделирования ИНС.
2. Сформированы чёткие требования к системе выполняющей распределённое моделирование ИНС.
3. Реализован программный продукт, полностью удовлетворяющим данным требованиям.

ЛИТЕРАТУРА:

1. NeuroProject _ Обучение _ Учебник - Нейронные сети Retrived from http://www.neuroproject.ru/neuro.php 12.04.10
2. Haykin, S. (1994). Neural networks: a comprehensive foundation. in Proceedings of the Sixth International Symposium on Micro Machine and Human Science, (Nagoya, Japan. Prentice Hall PTR Upper Saddle River, NJ, USA.
3. Clever Code Retrived from http://clevercode.blogspot.com/ 12.04.10
4. Интернет-Математика 2010 Retrived from http://imat2010.yandex.ru/ 12.04.10
5. Climateprediction.net | The world's largest climate forecasting experiment for the 21st century. Retrived from http://climateprediction.net/ 12.04.10
6. Gray, J. (2003). Distributed computing economics. Computer Systems, (March), 93–101. Springer.
7. Adolph, M. (2009). Distributed Computing : Utilities , Grids & Clouds. Technology.
8. Python Programming Language -- Official Website, Retrieved from http://www.python.org/ 12.04.10
9. Django | The Web framework for perfectionists with deadlines, Retrieved fromhttp://www.djangoproject.com/ 12.04.10
10. Isard, M., Prabhakaran, V., Currey, J., Wieder, U., Talwar, K., Goldberg, A., et al. (2009). Quincy: fair scheduling for distributed computing clusters. In Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles (p. 261–276). ACM.
11. Settles, M., & Rylander, B. (2002). Neural network learning using particle swarm optimizers. Advances in Information Science and Soft Computing, 224–226. Citeseer.
12. Chen, H. (1995). Machine Learning for Information Retrieval : Neural Networks , Symbolic Learning , and Genetic Algorithms. Management Information Systems, 46(3), 194-216.