Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Разработка инструментария для создания искусственных нейронных сетей на мобильных платформах на примере iOS

Курсовая работа студентки 445 группы

Золотухиной Алины Игоревны

Научный руководитель А.Е. Торегожин

Санкт-Петербург

2011

Оглавление

Введение

Искусственная нейронная сеть – это математическая модель (или вычислительная модель), основанная на структуре и функциональных аспектах биологических нейронных сетей [16]. Искусственная нейронная сеть состоит из группы связанных между собой искусственных нейронов. В большинстве случаев это изменяющаяся система, которая изменяет свою структуру на основе информации, проходящей по нейронной сети во время её обучения. Как правило, нейронные сети используются для моделирования сложных связей между входами и выходами или для нахождения путей.

В настоящее время искусственные нейронные сети применяются во многих областях. С их помощью решаются разнообразные задачи в сферах управления, прогнозирования, оценки качества.

Нейронные сети используются для распознавания образов. Они могут выделять определённые объекты из видеоряда, искать нужные объекты на фотографиях, распознавать лица. Нейронные сети часто используют в задачах идентификации личности.

Ещё одна задача для нейронных сетей – это выполнение прогнозов. В метеорологии на основе данных о температуре, влажности, скорости ветра, и т.д. с помощью нейронных сетей производятся прогнозы погоды (при этом, как правило, для разных сезонов используют разные нейронные сети) [9]. Использование нейронных сетей в финансовой сфере позволяет предсказывать динамику цен производственных финансовых активностей и индексов курсов акций, эффективность диверсификации портфельных капиталовложений, риски предоставления кредитов, а так же многое другое [1].

Таким образом, существует значительное количество приложений, использующих нейронные сети.

Последнее время всё большей популярностью пользуются мобильные устройства. Трудно представить себе человека, который не имел бы у себя смартфона, коммуникатора или хотя бы мобильного телефона. С развитием технологий мобильные устройства становятся всё более и более сложными, их функциональность разрастается, появляются новые приложения. Многие из этих приложений используют нейронные сети, однако на данный момент инструментарии для создания и обучения искусственных нейронных сетей существуют преимущественно для «больших компьютеров», но не для мобильных приложений. Таким образом, каждый разработчик мобильных приложений, использующий нейронные сети,

должен создавать и обучать нейронную сеть для своего приложения самостоятельно.

Инструментарий для создания нейронных сетей на мобильных устройствах сильно бы упростил работу разработчиков. Им надо было бы только задать параметры нейронной сети, предоставить обучающие данные, и всё остальное было бы выполнено за них.

1. Обзор существующих решений

На данный момент существует немало средств, предоставляющих возможность работы с искусственными нейронными сетями. Среди них можно перечислить NeuroSolutions, NeuronDotNet, Neuroph, OpenCV, PWNLIB, NNGPULIB, Neural Network Toolbox.

NeuroSolutions — инструмент графического редактирования нейронных сетей, который позволяет легко построить модель нейронной сети по данным пользователя [13]. Среди достоинств инструментария можно выделить поддержку CUDA (Compute Unified Device Architecture) – программно-аппаратной архитектуры, позволяющей производить вычисления с использованием графических процессоров NVIDIA [6]. Использование CUDA позволяет во много раз увеличить скорость работы нейронных сетей. Разработчик приложений может использовать NeuroSolutions в своих проектах, сгенерировав DLL или используя генерацию исходного кода на C++.

NeuronDotNet — библиотека с открытым исходным кодом на языке C# [12]. Она как предоставляет уже реализованный набор классов для работы с нейронными сетями, так и даёт возможность расширять набор классов и добавлять новые функции. Для использования библиотеки необязательно использовать C#, так как код совместим с любым языком, поддерживаемым платформой .NET.

Neuroph — библиотека с открытым исходным кодом на языке java [7], очень удобная и понятная, с небольшим количеством базовых классов, отвечающих основным понятиям нейронной сети. Кроме исходных классов, она также предоставляет GUI обработчик нейронных сетей [8]. Библиотека является достаточно гибкой и хорошо документированной и имеет инструменты, которые можно использовать для типовых задач разработки нейронных сетей (создания, обучения, тестирования и использования).

OpenCV (Open Source ComputerVision Library) – библиотека с открытым исходным кодом на C/C++, предоставляющая алгоритмы компьютерного зрения, обработки изображений и численные алгоритмы общего назначения [15]. В этой библиотеке реализованы нейронные сети прямого распространения, а именно многослойный перцептрон. Имеется несколько функций активации, обучение производится методом обратного распространения ошибки. Так же как и NeuroSolutions, OpenCV предоставляет поддержку CUDA.

PWNLIB – продукт отечественной компании «Павлин Технологии», обеспечивающий основные функции работы с нейронными сетями [3]. Поддерживает только сети прямого

распространения и обучение несколькими методами (в том числе методом обратного распространения ошибки, градиентным спуском и методом Левенберга-Марквардта).

NNGPULIB — продукт отечественной компании «Павлин Технологии», обеспечивающий ускорение расчета выхода многослойных нейронных сетей прямого распространения сигнала с применением графического процессора [4]. В нём также существует поддержка CUDA.

Neural Network Toolbox – пакет дополнений к MATLAB, наиболее распространён и удобен для академических целей [11]. В нём поддержаны разнообразные типы нейронных сетей и несколько алгоритмов обучения нейронных сетей.

Как уже было сказано выше, все эти средства работают на «больших компьютерах», и не могут использоваться на мобильных устройствах.

2. Постановка задачи

Задачей данной курсовой работы является разработка инструментария для создания искусственных нейронных сетей на мобильных устройствах.

Задача разделается на следующие подзадачи:

- анализ достоинств и недостатков существующих инструментариев на других платформах
- разработка и реализация архитектуры инструментария:
    - алгоритм создания нейронной сети
    - алгоритм обучения нейронной сети

Цель курсовой работы – получить некую библиотеку базовых классов, которую разработчик приложений сможет либо использовать для своих нужд такой, какая она есть, или же, пользуясь существующими классами, на их основе сконструировать что-то своё. Другими словами, он может либо воспользоваться уже реализованными типами нейронных сетей и алгоритмами обучения, либо написать свои, отнаследовав их от базовых классов.

3. Реализация

3.1.Технология

В качестве технологии была выбрана платформа iOS SDK и язык Objective C**.** Выбор обуславливался следующими причинами:

1) iOS SDK поддерживает языки C, C++, Objective-C, которые, будучи C-подобными языками, являются наиболее распространёнными и быстрыми
2) iOS SDK предоставляет удобный высокоуровневый механизм работы с многопоточностью [5];
3) программы, написанные с использованием iOS SDK, работают не только на мобильных устройствах, но и на персональных компьютерах. Таким образом, нет необходимости думать о том, как обучать нейронную сеть на мобильном устройстве, на котором может не хватить памяти или мощности – можно обучить сеть на компьютере, а затем обученную и сохранённую сеть перенести на мобильное устройство.

В связи с выбором технологии инструментарию было решено присвоить имя «iOSNeuron».

3.2 Архитектура

На рисунке 1 представлена диаграмма основных классов инструментария, реализованного в рамках данной курсовой работы.

Базовым классом является класс NeuralNetwork, от которого должны наследоваться все другие классы, реализующие основные функции нейронных сетей. NeuralNetwork – универсальный класс для нейронных сетей, и каждая нейронная сеть нового типа имеет его как своего родителя. Основные функции реализованы в классе NeuralNetwork, наследники должны переопределять только инициализацию, несмотря на то, что при необходимости могут быть переопределены и другие методы, в общем случае этого не требуется.

Нейронная сеть состоит из слоёв (Layer), каждый из которых, в свою очередь, состоит из нейронов (Neuron) и задаётся их количеством. У каждого нейрона есть входная функция (InputFunction) и активационная функция (TransferFunction). На основе входной функции мы задаём изначальный вход нейрона. Активационная функция используется при вычислении выхода нейрона и при обучении.

InputFunction складывается на основе WeightsFunction (весовой функции) и

SummingFunction (функции суммирования). Все три эти функции имеют только один метод – получение выхода нейрона (GetOuput). Функция весов на основе массива связей получает массив из чисел, поступающих на вход данного нейрона. Функция суммирования просто суммирует числа полученного массива и выдаёт ответ. В данной работе были реализованы простейшие случаи весовой функции и функции суммирования, при желании метод GetOutput может быть переопределён для более сложных структур.

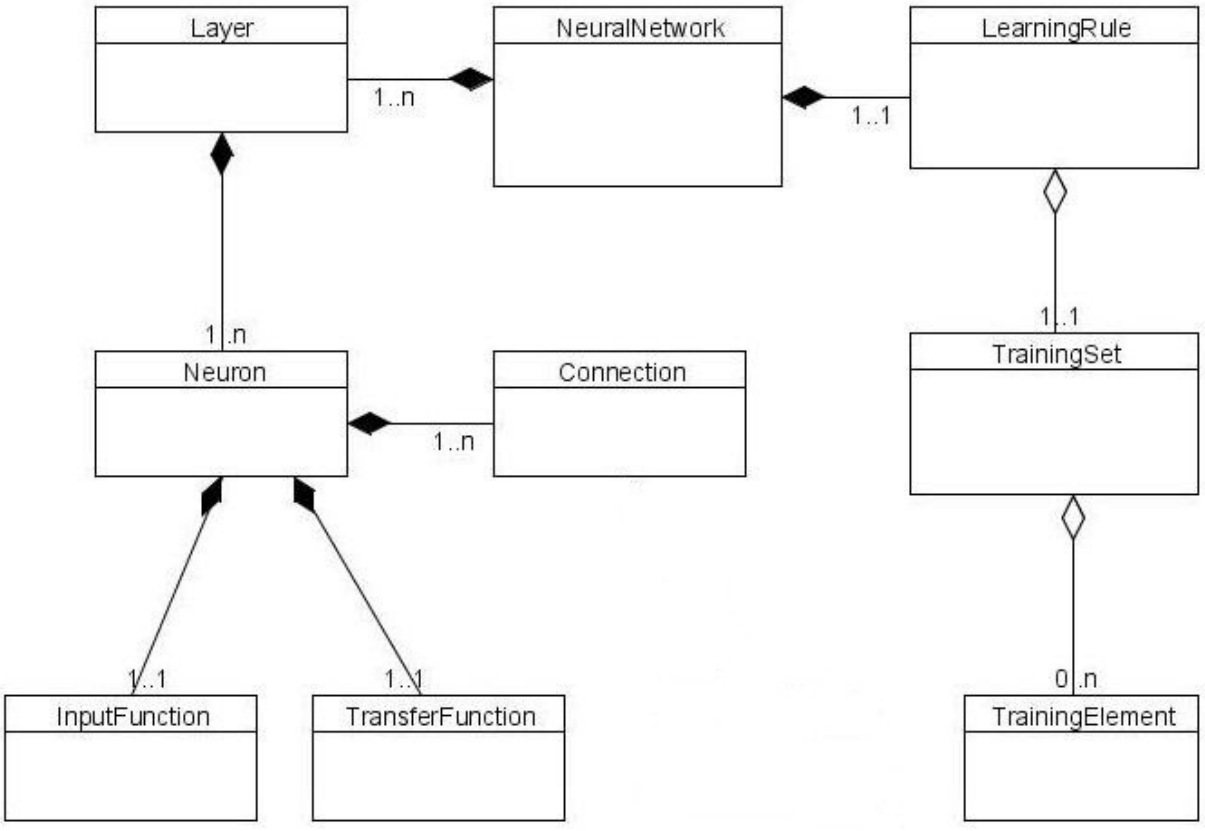

Рисунок 1. Диаграмма классов iOSNeuron.

Класс активационной функции (TransferFunction) имеет два метода: получение выхода (GetOutput) и получение производной (GetDerivative), оба работают на основе входа нейрона. TransferFunction является базовым абстрактным классом, соответственно в нём нет никакой реализации этих методов, реализация производится в наследных классах.

В данной работе было реализовано три класса, наследующихся от TransferFunction: Linear, Sigmoid и Tanh – линейная функция, сигмоид и тангенсоид соответственно. Линейная функция используется для входного (первого) слоя нейронной сети, она возвращает вход нейрона, умноженный на некоторый шаг (который по умолчанию равен единице).

Сигмоидальная функция выглядит следующим образом:

$$\frac{1}{1+e^{-slope\times net}}$$

и является одной из наиболее распространённых активационных функций. То же самое можно сказать и о тангенсоидальной функции:

$$\frac{1-e^{-slope\times net}}{1+e^{-slope\times net}}$$

В этих формулах slope – это шаг функции, а net – это вход нейрона.

Как уже было сказано, активационная функция используется для вычисления выхода нейрона, который получается как результат активационной функции от входа. Посчитав выходы всех нейронов и распространив входной вектор по всей сети, можно получить её выход.

У каждой нейронной сети можно задать обучающее правило — LearningRule. Данный класс также является базовым, от которого могут наследоваться другие классы, реализующие тот или иной алгоритм обучения. У каждого обучающего правила (LearningRule) есть обучающее множество (TrainingSet), которое, в свою очередь, состоит из обучающих элементов (TrainingElement). TrainingElement создаётся на основе обучающих данных (входного вектора и желаемого выхода).

Каждый нейрон имеет список входных связей (inputConnections) и выходных связей (outputConnections). Связь (Connection) определяется нейроном, являющимся её началом, и весом связи. Изначально веса задаются случайными числами от 0 до 1, а затем в процессе обучения изменяются так, чтобы сеть давала наименьшую ошибку. Проходя по связям между нейронами, входной вектор умножается на веса, и только затем достигает нейрона, где на него уже действует активационная функция.

3.3 Многослойный перцептрон.

Примером конкретной нейронной сети, реализованной в инструментарии iOSNeuron, является многослойный перцептрон (MultiLayerPerceptron). Многослойный перцептрон — нейронная сеть прямого распространения (Feed Forward Neural Network), в которой нейроны каждого слоя получают входные данные от нейронов предыдущего слоя и передают данные нейронам следующего слоя. Внутри одного слоя связей между нейронами нет. Связи

существуют только между соседними слоями. Сеть состоит из входного слоя, выходного слоя и одного или нескольких промежуточных (скрытых) слоёв. На рисунке 2 представлена схема многослойного перцептрона, а на рисунке 3 более подробно рассмотрен отдельный нейрон сети.

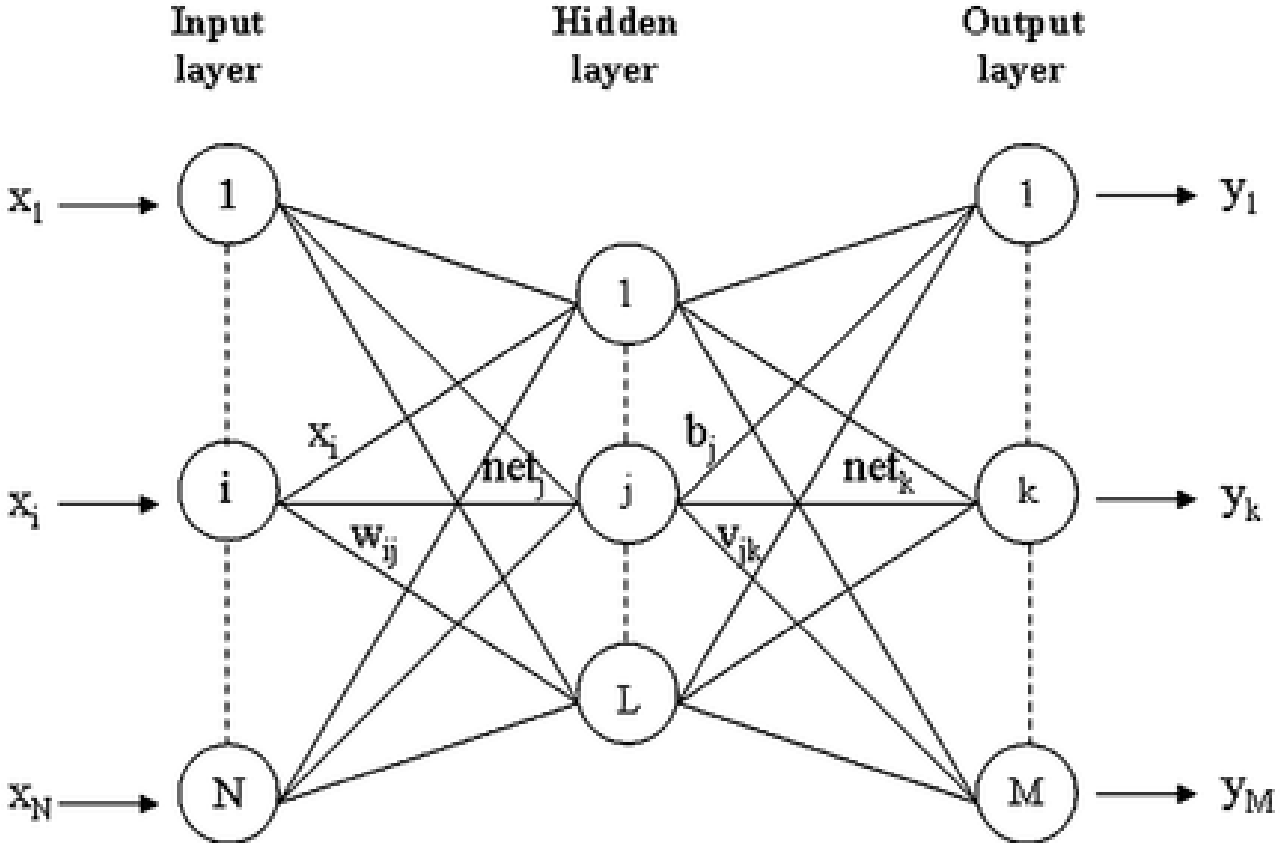

Рисунок 2. Схема многослойного мультиперцептрона

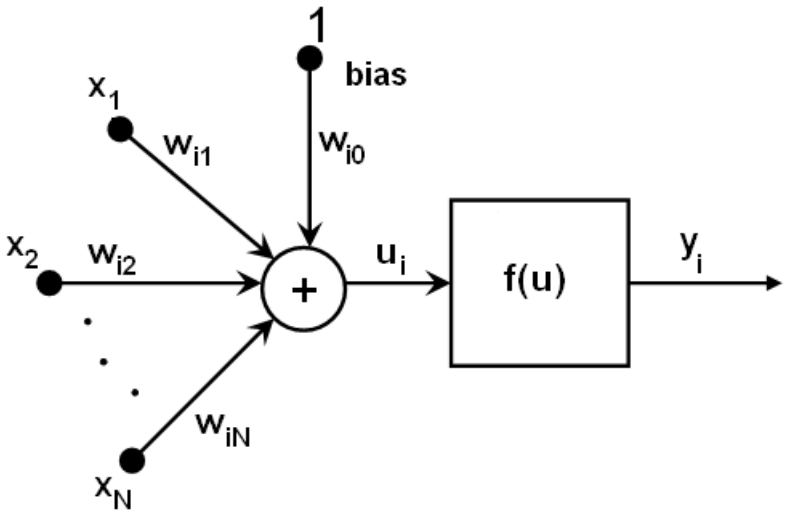

Рисунок 3. Схема нейрона многослойного мультиперцептрона

Как уже упоминалось выше, класс MultiLayerPerceptron отнаследован от класса

NeuralNetwork. При инициализации в нём вызывается метод CreateNetwork, которому на вход подаётся массив, в котором содержится количество нейронов в каждом слое.

Первый слой создаётся из нейронов, имеющих линейную активационную функцию. Следующие слои имеют сигмоидальную активационную функцию. Для каждого слоя, не являющегося первым, мы запоминаем предыдущий слой. Также каждый слой, не являющийся первым, мы связываем связями с предыдущим так, что каждый нейрон одного слоя связан с каждым нейроном другого слоя. Наконец, мы устанавливаем для нейронной сети обучающее правило – метод обратного распространения ошибки.

3.4. **Метод обратного распространения ошибки**

Алгоритм обратного распространения ошибки – итеративный градиентный алгоритм обучения, используемый для минимизации среднеквадратичного отклонения реальных выходных значений нейронов от желаемых выходных значений. Этот алгоритм используется для обучения многослойных нейронных сетей с последовательными связями, то есть как раз подходит для многослойного перцептрона.

Обучение с обратным распространением ошибки предполагает два прохода по всем слоям сети – прямого и обратного. При прямом проходе на сеть подаётся вектор входных значений, который затем распространяется по сети до последнего выходного слоя. Причём веса при прямом проходе остаются неизменными и учитываются при передаче вектора входных данных. При обратном же проходе веса корректируются на основе ошибки, полученной из разности фактического и желаемого выхода сети.

Алгоритм работы метода обратного распространения ошибки:

1. Инициализация всех весов случайными значениями от 0 до 1.
2. Подача входного вектора из обучающего множества на первый слой сети.
3. Распространение входного вектора до последнего слоя сети, вычисление выхода сети.
4. Вычисление вектора ошибки – разности между фактическим и желаемым выходом.
5. Коррекция весов сети на основе вектора ошибки с целью минимизации ошибки на последующих итерациях.
6. Повторение шагов 2-5 для каждого элемента обучающего множества до тех пор, пока уровень ошибки не станет приемлемым.

Коррекция весов

Коррекция весов различна для выходного слоя и для скрытых (внутренних) слоёв. Рассмотрим сначала коррекцию весов для выходного слоя. Для каждого нейрона мы храним некоторую ошибку. Если разница между фактическим выходом и желаемым выходом равна нулю, то и ошибка нейрона задаётся нулём. Если же разница ненулевая, то в поле ошибки нейрона записывается значение полученной разницы, умноженное на значение производной активационной функции в точке общего входа сети. После чего веса изменяются в соответствии с этой ошибкой.

В скрытых слоях у каждого нейрона высчитывается некая дельта как сумма по всем связям произведения ошибки нейрона на другом конце связи и веса связи. В поле ошибки нейрона записывается произведение этой дельты и значения производной активационной функции в точке общего входа сети, после чего веса нейронов опять же изменяются в соответствии с этой ошибкой.

Веса нейронов изменяются по следующему правилу: каждый вес увеличивается на величину, равную произведению некоторой константы, ошибки нейрона и входа связи (т.е. величины, передаваемой нейроном на другом конце связи).

4. Результаты работы

Были решены следующие задачи:

- Проведён краткий анализ существующих инструментариев для создания и обучения искусственных нейронных сетей
- Разработан алгоритм создания нейронных сетей
- На основе разработанного алгоритма построена сеть многослойный перцептрон
- Реализован алгоритм обучения искусственных нейронных сетей
- Проведена проверка работоспособности инструментария

5. Заключение

В рамках данной работы была осуществлена реализация инструментария для создания и обучения искусственных нейронных сетей для iOS. На данный момент количество разных типов создаваемых нейронных сетей мало, и в дальнейшем планируется расширение инструментария с помощью добавления других типов нейронных сетей. Так же возможно расширение алгоритмов обучения нейронных сетей.

Главный недостаток существующих инструментариев – слишком большое участие со стороны человека. Для того, чтобы воспользоваться тем или иным инструментом, необходимо знать его специфику или знать какой-то конкретный язык программирования, задавать количество слоёв в сети, количество нейронов в слое, выбирать тип сети и так далее. Планируется избавиться от этого недостатка и разработать алгоритм автоматизации построения нейронных сетей, который на основе обучающих данных будет сам выбирать тип сети, задавать её параметры, а на выходе выдавать уже готовую и обученную нейронную сеть.

Список литературы

1. Д. Бестенс, В. Ван Ден Берг «Нейронные сети и финансовые рынки»

2. А. И. Галушкин, Нейронные сети: основы теории

3. ПАВЛИН ТЕХНОЛОГИИ – разработка систем с искусственным интеллектом, автоматизация производства, обработка данных и изображений – PWNLIB 1.0 http://www.pawlin.ru/content/view/20/8/

4. ПАВЛИН ТЕХНОЛОГИИ – разработка систем с искусственным интеллектом, автоматизация производства, обработка данных и изображений – NNGPULIB 1.0 http://www.pawlin.ru/content/view/19/10/

5. Concurrency Programming Guide: Introduction http://developer.apple.com/library/ios/#documentation/General/Conceptual/ConcurrencyProgram-mingGuide/Introduction/Introduction.html

6. CUDA – wiki http://en.wikipedia.org/wiki/CUDA

7. Java Neural Network Framework http://neuroph.sourceforge.net/

8. Neuroph GUI Network Editor http://neuroph.sourceforge.net/images/screen-shot-big.jpg

9. Imran Maqsood, Muhammad Riaz Khan, Ajith Abraham “An ensemble of neural networks for weather forecasting” Springer-Verlag London Limited 2004

10. Multilayer Perceptron Neural Networks http://www.dtreg.com/mlfn.htm

11. Neural Network Toolbox - MATLAB http://www.mathworks.com/products/neural-net/?BB=1

12. NeuronDotNet – Artificial Neural Networks in C# http://neurondotnet.freehostia.com/

13. NeuroSolutions http://www.neurosolutions.com/products/ns

14. OpenCV Neural Networks http://opencv.willowgarage.com/documentation/cpp/neu-ral_networks.html

15. OpenCV Wiki http://opencv.willowgarage.com/wiki/

16. Artificial Neural Network — wiki http://en.wikipedia.org/wiki/Artificial_neural_net-work