Теория о нейронных сетях

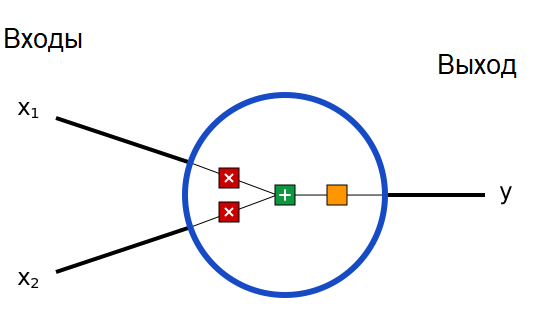
. Нейронные сети – это попытка математиков различными законами и формулами описать работу биологической нейронной сети. Нейронная сеть представляет из себя последовательно соединенные между собой нейроны. Основной задаче нейронных сетей является получение определенных результатов, по которым будет выполняться определенное предположение, по набору значений. Предположения могут быть различными от того какой цвет показывают нейросети, до того как правильно ответить на то или иное предложение.

Основными фундаментальными компонентами нейронной сети являются нейрон и синопсис.

Синопсис это связи между нейронами. Они обладают всего одним параметром – вес. Вес необходим для изменения входной информации передающейся от нейрона к нейрону. Этот параметр показывает насколько входная информация будет влиять на выход нейрона. Ниже представлена формула по которой происходит расчет входов в нейрон умноженные на веса синопсисов.



Нейрон представляет из себя математическую еденицу, имеющую неограниченной количество входов и один выход.



Нейрон это определенный математический оператор, который складывает значения входов. После к полученному значению прибавляется значение “смещения” и формируется значения выхода.



Далее значение выхода подается на функцию активации для того, что бы упорядочить результат работы нейрона к определенному диапазону значений, что бы в дальнейшем передавать упорядоченный ответ следующим нейронам, для их более правильной работы и обучения. В таблице 1 представлены наиболее часто используемые функции активации.

Таблица 1 Функци Активации

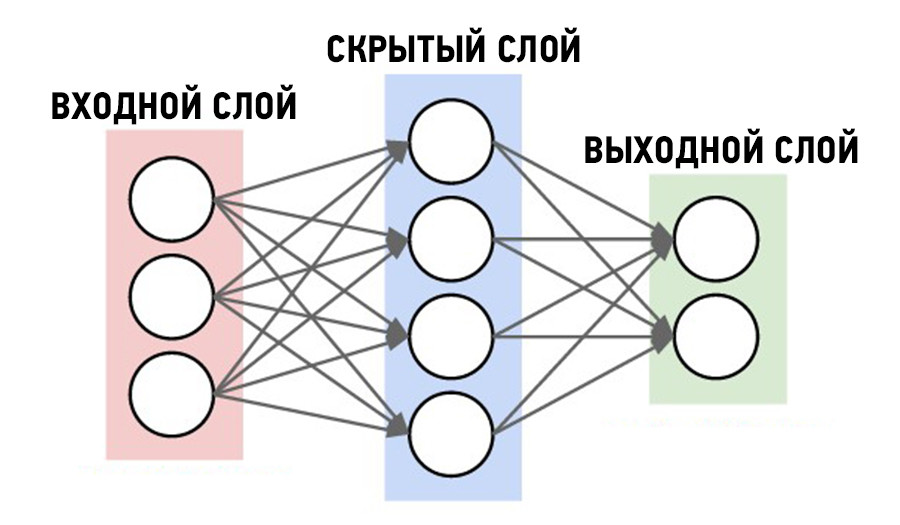
|  |  |  |  |
| --- | --- | --- | --- |
| Название | Функция | Диапазон значений | График функции |
| Линейная | F(x)=x | [-ifn;+inf] |  |
| Сигмоид |  | [0;1] | Sigmoid function - Wikipedia |
| Гиперболический тангенс |  | [-1;1] |  |
| Линейный выпрямитель |  | [0;inf) | How ReLU works in convolutional neural network - knowledge Transfer |

Функции активации являются важным инструментом в нейронных сетях на ровне с нейроном и синопсисам, влияющими на нейронную сеть, на ее работу и на ее обучение. Эти функции от части определяют какие нейроны будут активированы и какая информация будет передаваться дальше последующим нейронам.

Линейную функцию активации использую редко. В основном ее используют если необходимо передать данные с выходов нейронов дальше без изменений. Чаще всего в качестве функции активации применяют сигмоиду, её также называют логической функцией. Так как регуляторы могут выдавать и отрицательные и положительные значения поэтому функции Сигмоид в работе не подойдет для работы с числами в данном диапазоне лучше всего использовать гиперболический тангенс.

Состав нейронной сети

Для того чтобы правильно описать состав нейронной сети необходимо ввести понятие слоя. Слой – это набор, не связанных между собой нейронов, на который подаются значения либо от предыдущих нейронов, либо из в не сети. Ниже на рисунке представлены, какие слои входят в состав нейронной сети.



Нейронные сети состоят из трех основных слоев:

Входной слой – служит для подачи значений в нейронную сеть;

Скрытый слой – основной вычислительный слой, в котором происходит основная работа нейронной сети. Скрытый слой может состоять из одного или нескольких слоев.

Выходной слой – предназначен для вывода результатов работы нейронной сети;

Выходы нейронов каждого слоя связаны синопсисами с входом каждого нейрона следующего слоя.

Нейросеть может иметь любое количество слоев, и в них может быть любое количество нейронов. Эти параметры влияют на работу нейронной сети и на ее обучения. Не существует формул по которым от количества входов или от типа передаваемых данных или от их диапазона можно было бы рассчитать количество слоев и количество нейронов в каждом слое для корректной работы сети. Именно поэтому значения этих параметров подбираются эмпирическим путем.

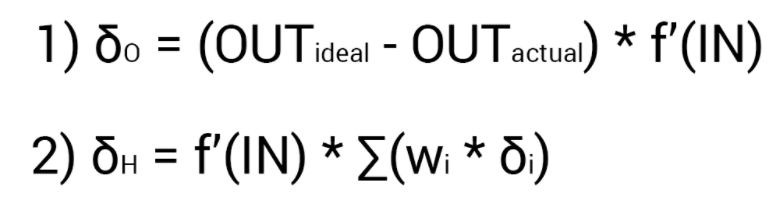
Методы обучения нейронных сетей

Выделяют четыре основных метода обучения нейронных сетей, каждый из них имеет свои плюсы и свои минусы.

Машинное обучение с учителем.

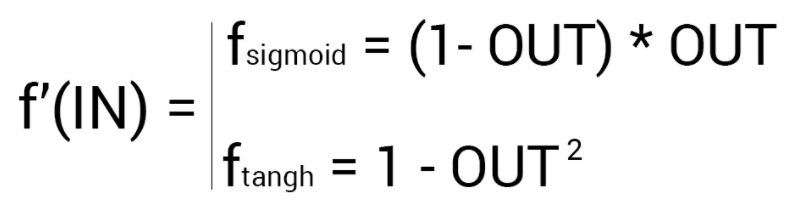
Этот тип обучения подразумевает, что во время обучения вместе с входными параметрами нейронной сети в данных для обучения, как их еще называют дата сеты, присутствуют значения выходных параметров сети. Во время обучения нейроная сеть сравнивает полученные данные на своем выходном слое с данными, которые у нее должны были получиться при данных значениях входного слоя. И по методу обратного распространения ошибки меняет весовые коэффициенты своих синопсисов. После этого данные ещё раз подаются, и происходит сравнение выходов и так до тех пор, пока значение разницы не достигнет нуля или не будет близкой к нулю. Тогда можно считать, что сеть обучилась.

Так же для дальнейшего понимания методов обучения необходимо рассказать про метод обратного распространения ошибки. Для подсчёта разницы между выходом и значением, к которому она стремится, используют следующую формулу.

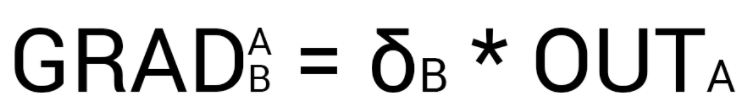
,где б0 – это разница для выходного слоя.Бн – разница для скрытых слоев дельта.Оут идеал это идеальный выход.Оут актуал это то значение которое рассчитала нейросеть .Ф – это производная функции активации слоя. Ви – вес синапса Би – дельта ошибки предыдущего слоя

Так как нейроны, содержащиеся на выходном слое, не имеют исходящих синапсов, то для подсчета дельты мы будем пользоваться первой формулой. Для нейронов, содержащихся во входном и скрытом слое, второй.

Так как при расчете МОР мы используем производную функции активации, то функций активации необходимо выбирать такие, что бы их можно было продифференцировать. Так же для упрощения расчетов для некоторых функций активации формулу производной можно заменить на упрощенную:



Далее необходимо найти градиент для каждого входящего в нейрон синапса, что бы в дальнейшем правильно изменить его вес. Значение градиента будем искать по следующей формуле.

,где А-это начала синапса Б - конец синапса

Теперь по формуле представленной ниже мы можем рассчитать вес, на который нам необходимо изменить вес синапса

.где Е – это скорость обучения. а – момент обучения ви-1 – это вес синапса на предыдущем шаге итерации.

Далее необходимо полученное значение сложить с весом синопсиса.

Чаще всего обучение с учителем используется для решения двух типовых задач это задачи классификации и задачи регрессии.

Задачи классификации чаше всего применяются, когда нужно определить тот или иной предмет на фото. На вход нейронной сети подается изображение, и каждое изображение связанно с определённым наборов нейронов в выходном слое. Например, необходимо различить автотранспорт, изображенный на фото. На вход нейронной сети будут подаваться фотографии с изображением поезда, автомобиля или мотоцикла. Каждому изображению будет дана метка соответствующему определенным значениям выходного слоя. Например, когда на вход подаётся изображения поезда, то на выходном слое нейроны должны выдать результат [1;0;0]. При подаче на вход изображения автомобиля на выходе должны быть значения [0;1;0], а при загрузке изображения мотоцикла [0;0;1] соответственно.

После обучения нейронной сети и при загрузке на ее входной слой изображения, не участвовавшего в обучения, автомобиля нейросеть должна выдать результат похожим на следующий:[0.05;0.99;0.05], что говорит о том, что нейросеть на 99% уверена, что на изображении автомобиль.

Задача регрессии связана с непрерывной подачей данных на вход нейросети. Одним из примеров регрессии является линейная регрессия. Нейросеть при получение конкретного значения x должна вычислить ожидаемое значение у.

Обучение без учителя.

Иногда собрать обучающую выборку для нейросети, где каждому значения входа соответствует определенное значение выходов достаточно сложно именно поэтому используют метод обучения без учителя. Перед нейросетью ставится задача самостоятельно найти корреляции в данных, извлекая полезные признаки и анализируя их.

Одним из примеров такого типа обучения является кластеризация. Нейросеть находит общие признаки у данных и группирует их вместе.На рисунке ниже приведен пример кластеризации.



Обучение с частичным привлечением учителя.

Этот метод признано считать самым лучшим, когда обучение происходит на большой выборке данных, потому что он включает в себя и обучение с учителем и обучение без учителя. Обучающая выборка при таком обучение имеет значения входных данных с выходным результатом и без результатов. Такой метод обучения применяют если из набора данных трудно извлечь общие признаки или набор данных настолько велик, что дать значения выхода каждому набору входных параметров трудоемкая задача.

Подобный метод машинного обучения получил широкое применение в медицине, в анализе рентгенов и МРТ.

Обучение с подкреплением.

В отличие от машинного обучения с учителем при данном типе обучения на вход нейронной сети не подается обучающая выборка. Вместо этого сети обучается “методом проб и ошибок”. При таком подходе обучения нейросет пытается найти оптимальный путь для достижения цели. Если нейросеть ошибается, то она получает “штрафные балы” и начинает обработку данных с начала. Если нейросеть предпринимает действие, которое приведет ее к искомо верному результату, то она получает “награду” и продолжает работу. В конечном итоге нейросеть обучается предсказывать свое следующие действие для получения максимально “награды”.

Каноничным примером обучения с подкреплением является обучение нейросети управлять машинкой, чтобы пройти заданный маршрут как можно быстрее.

Для осуществления поставленной задачи из машинки испускается восемь лучей(рисунок№), которые являются в свою очередь входами для нейронной сети. Эти лучи передают в нейронную сеть расстояние, до какого либо препятствия. По схожему принципу работает автопилот в автомобилях компании Tesla, только в данном случае лучи передают не только расстояние до предмет, а так же способны различать этот предмет и в зависимости от его типа принимать то или иное решение.



Во время обучения машина методом проб и ошибок учится не врезается в препятствия и проходить трассу как можно быстрее.

Такое метод обучения чаще всего используется в компьютерных играх для управления искусственным интеллектом, и в обучение роботов, которые управляют автономными транспортными средствами без участия человека.

Для решения задачи поставленной в рамках данной диссертации больше всего подойдет метод обучения с учителем по нескольким причинам:

Во-первых, в обучающей выборке будет заранее известно какое значение по входным параметрам должна получить нейросеть, для того чтобы с имитировать работу ПИД ругелятора.

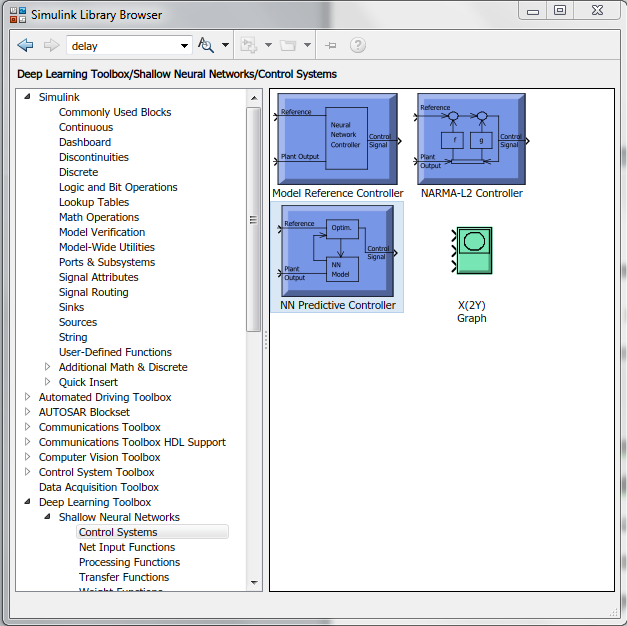
Во-вторых, обучающая выборка будет получена в ходе работы реального регулятора на объект. Поэтому в разметке обучающих данных не будет проблем.

В-третьих, данный тип обучения решает проблемы регрессии и постоянного потока данных, что будет определяющим фактором при имитации работы ПИД регулятора.

Специализированные программы для работы с нейросетями

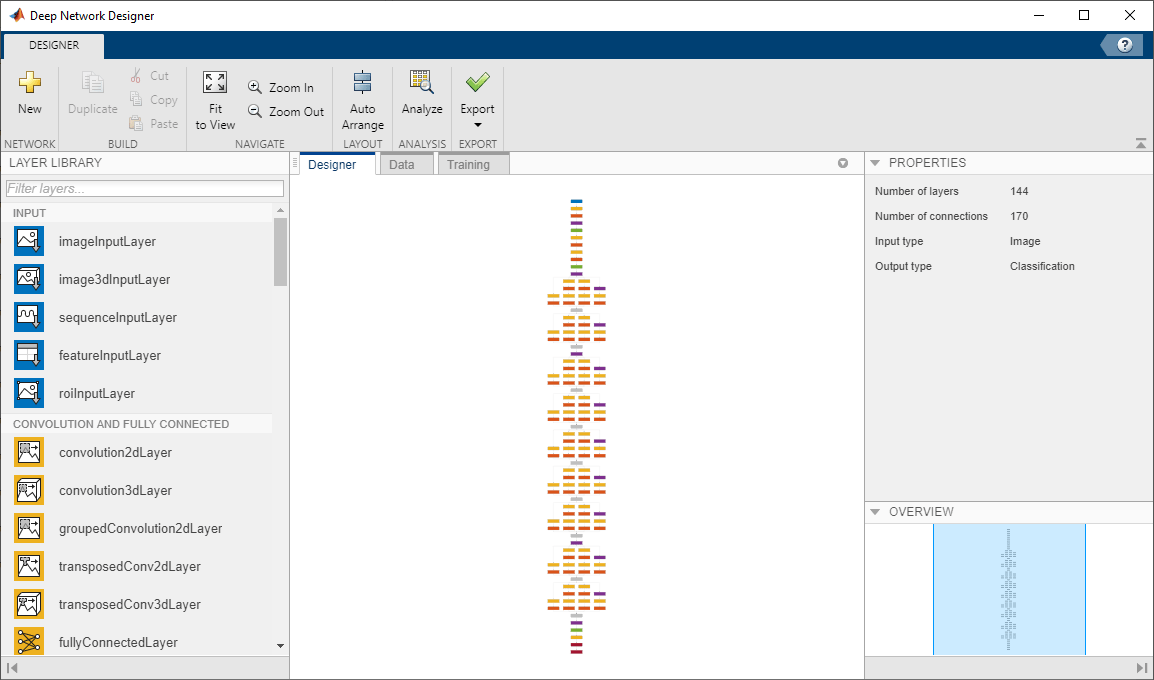
Из-за большой популярности нейронных сетей существует огромное количество программ позволяющих работать с ними. Все эти программы имеют свои плюсы и свои минусы.

Matlab – это мощный пакет прикладных программ для решения задач технических вычислений. Он имеет огромное количество библиотек позволяющих смоделировать любую ситуацию. Одна из таких библиотек под названием “Deep Learning Toolbox” позволяет работать с нейросетями. В своем арсенале она имеет уже готовые примеры контроллеров на основе нейросетей, которые можно обучить и настроить на работу.



Так же в эту библиотеку входят блоки с помощью которых можно составить свою нейросеть, смоделировать функции активации

Так же в матлаб существует приложение **Deep Network Designer,** которое визуализирует создание глубоких нейронных сетей.



Приложение интуитивно понятно из доступных блоков строится любая топология сети. Все блоки можно подробно настроить под свои нужды. Так же matlab предоставляет большой архив с уже готовыми нейросетями, что помогает пользователю научится правильно выбирать и строить топологию сетей и смотреть уже готовые решения для некоторых проблем.

Большую популярность для работы с нейросетями получил язык программирования Python. Python – это высокоуровневый язык программирования с динамической типизации и автоматически управлением памяти. Основной идеей языка является читаемость кода и повышение производительности разработчика, за счет того что многий функционал язык делает за разработчика, например работа с памятью. Python является интерпретируемым языком, что уменьшает скорость работы написанных на нем программ и большим потреблением памяти. Из-за своей популярности и открытости кода python имеет огромное количество библиотек, в том числе и для работы с нейросетями. Одной из таких библиотек является Keras.

Keras является открытой нейросетевой библиотекой, которая представляет из себя надстройку на фреймворками TensorFlow и Theano. Основными преимуществами этой библиотеки перед другими является ее компактность, модульность и предоставление интуитивно более понятных функций, которые помогают, без особых усилий создавать нейронные сети.

В данной работе работа с нейросетями будет вестись с помощью языка программирования Python его библиотеки Keras,потому что “Deep Learning Toolbox” хоть и имеет уже готовые котроллеры на основе нейросетей, но они сложно настраиваемые и не позволят изменить топологию сети и имеют сложный интерфейс и методы работы. Приложение **Deep Network Designer** удобно в использование, имеет обширные возможности для построения нейросетей, но оно в основном заточено на глубокие нейросети и работу с изображениями.

Создание и обучение нейронной сети

Для создания и обучения нейронной сети как говорилось в предыдущей главе будет использоваться язык Python его библиотека Keras.

В первую очередь необходимо выбрать архитектуру нейронной сети и передать это значение в модель. Выполняется это с помощью следующей команды.

model = keras.Sequential()

Sequential – означает, что будет использоваться линейный стек слоев.

Дальше необходимо добавить слои в модель с помощью функции add().

model.add(Dense(10, input\_shape=(2,), activation='relu',use\_bias=False))

Первое число говорит о том, сколько нейронов будет находиться в слое. Так как вначале создается входной слой то необходимо в параметр input\_shape=(2,) передать число, которое скажет программе какое количество нейронов будут входными. Параметр activation отвечает за то, какая функция активации будет использоваться в данном слое. Функции активации описаны в первой таблице. Параметр use\_bias говорит системе хочет ли разработчик использовать смещение или нет.

С помощью функции add можно добавить в модель сколько угодно слоев.

После того как все слои добавлены в модель необходимо данную модель скомпилировать. Выполняется это с помощью функции

model.compile(loss='mean\_squared\_error', optimizer=keras.optimizers.Adam(0.1)).

Параметр loss - . Это значение, которое модель пытается минимизировать. В данном примере выбрана функция средне квадратичной ошибки

После компиляции модель готова к обучению. Для того чтобы приступить к обучению необходимо воспользоваться функцией fit.

history = model.fit(train, rezults, epochs=10)

В переменную History будет записана история обучения сети.Ее ошибки и точность после каждой эпохи. С помощью переменной history можно визуализировать обучение сети и получить графики по ошибкам и точности сети за весь период обучения.

Переменная train является массивом векторов, которые будут подаваться в сеть для ее обучения. Размер входного вектора должен быть равен количеству входных нейронов.

Переменная results это тоже массив векторов только на основание данных в этом массиве и значений на выходе нейронной сети будет происходить обучение модели и изменение весов синопсисов для уменьшения ошибки.

Переменная epochs принимает количество эпох. Количество эпох отвечает за то сколько раз будет проходить обучения нейросети

С помощью функции predict можно в уже обученную нейронную сеть подать значения на вход и узнать какой результат выдаст нейросеть.

Создание модели нейросети в Matlab Simulink

Такие характеристики как: значения подающиеся на входных слой, веса синопсисов, значения на выходном слое, можно представить в виде матриц. Поэтому работа всей сети сводится к умножению матрицы входа на матрицу весов синопсиса. Полученную матрицу необходимо провести через функцию активации.

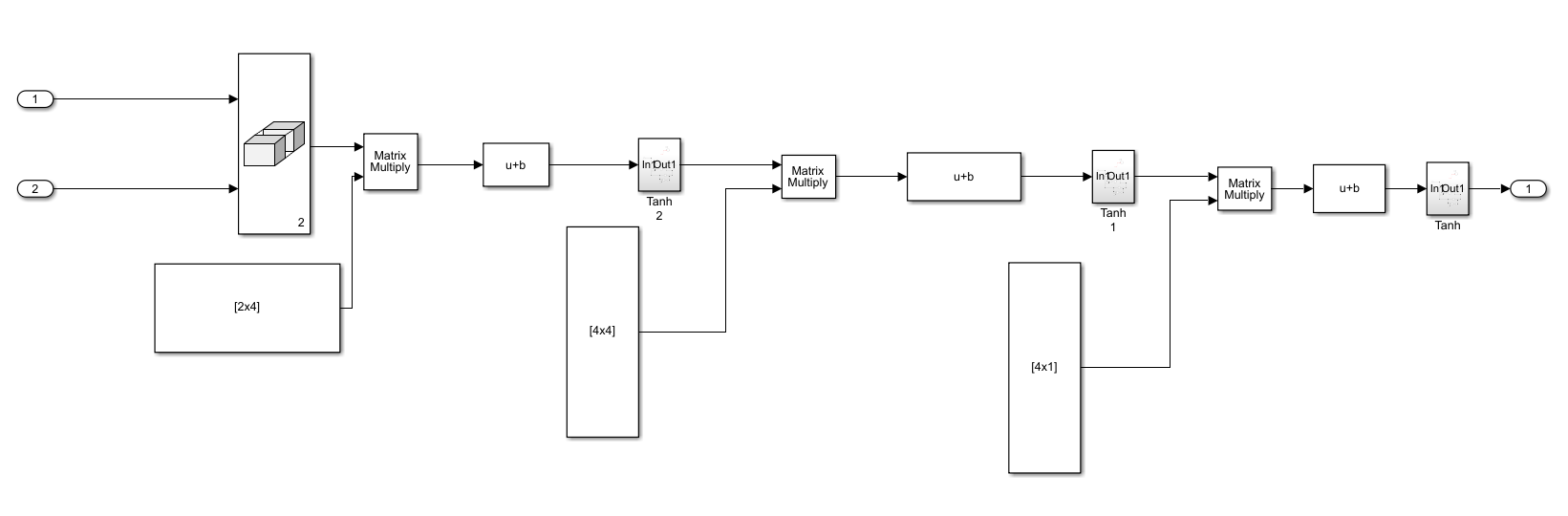
где

W – Матрица весов синопсисов

I- Матрица входных сигналов

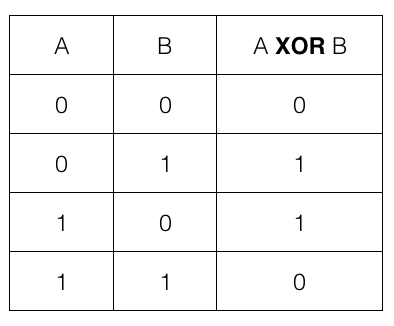
Х – результирующая матрица.

На рисунке ниже представлена нейросеть собранная с помощью библиотечных элементов в matlab Simulink.

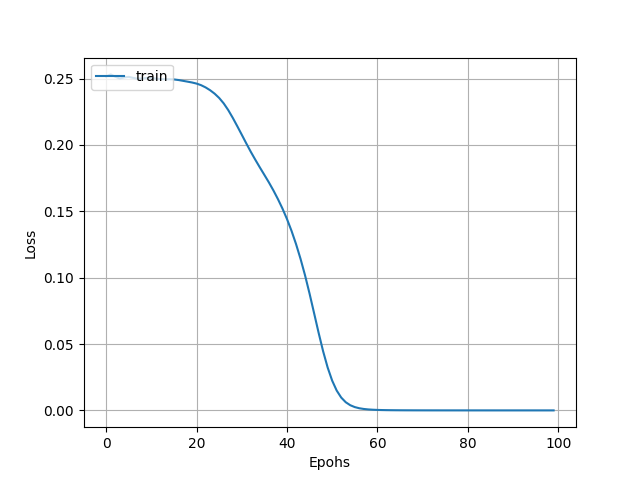


В модель подаются значения. Далее они преобразуются в матрицу размером 1х2 и умножаются на весовые коэффициенты синопсисов. Полученная матрица пропускается через функцию активации и умножается на веса синопсисов следующего слоя. И так продолжается до тех пор, пока не получится результат на выходе.

Для того что бы проверить правильность собранной модели в Python модель имеющая два входа, два скрытых слоя по четыре нейрона и одним выходом, была обучена на логическую операцию исключающее или представленную на рисунке ниже.



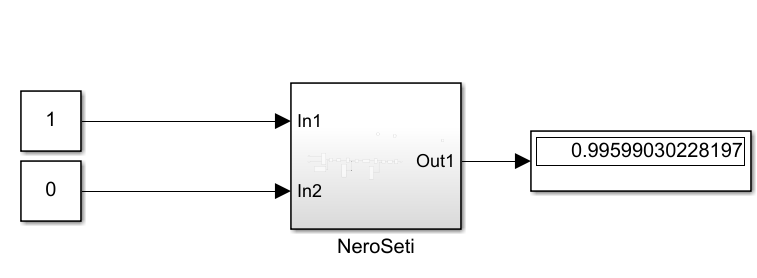
На графике ниже можно увидеть как ошибка между выходом нейронной сети и менялась с количеством пройденных эпох.



Значение ошибки на сотой эпохе составляет 0.000005. По графику видно, что модель обучилась к 60 эпохе, а дальше уже не обучалась. Если модель учить слишком много эпох, то возможен момент переобучения, когда ошибка начнет беспричинно расти.

Воспользовавшись функцией Predict и подав на вход нейронной сети значения 0 и 1 функция, показала значение 0.9959903, что по картинке 34 очень близко к верному значению.

Теперь необходимо выгрузить значения весов и смещений из Python и занести в модель matlab. На рисунке ниже представлен результат работы нейросети из матлаб.



Результат получился такой же как и при использование функции predict в Python, а это значит, что модель сделанная в matlab Simulink является правильной и отвечает всем параметрам.