Содержание:

Введение

Нейронные сети используются для решения сложных задач, которые требуют аналитических вычислений подобных тем, что делает человеческий мозг. Самыми распространенными применениями нейронных сетей является:

Классификация — распределение данных по параметрам. Например, на вход дается набор людей и нужно решить, кому из них давать кредит, а кому нет. Эту работу может сделать нейронная сеть, анализируя такую информацию как: возраст, платежеспособность, кредитная история и тд.

Предсказание — возможность предсказывать следующий шаг. Например, рост или падение акций, основываясь на ситуации на фондовом рынке.

Распознавание — в настоящее время, самое широкое применение нейронных сетей. Используется в Google, когда вы ищете фото или в камерах телефонов, когда оно определяет положение вашего лица и выделяет его и многое другое.

В последние несколько лет мы наблюдаем взрыв интереса к нейронным сетям, которые успешно применяются в самых различных областях - бизнесе, медицине, технике, геологии , физике. Нейронные сети вошли в практику везде, где нужно решать задачи прогнозирования, классификации или управления. Такой впечатляющий успех определяется несколькими причинами:

- **Богатые возможности.** Нейронные сети - исключительно мощный метод моделирования, позволяющий воспроизводить чрезвычайно сложные зависимости. В частности, нейронные сети *нелинейны* по свой природе (смысл этого понятия подробно разъясняется далее в этой главе). На протяжение многих лет линейное моделирование было основным методом моделирования в большинстве областей, поскольку для него хорошо разработаны процедуры оптимизации. В задачах, где линейная аппроксимация неудовлетворительна (а

таких достаточно много), линейные модели работают плохо. Кроме того, нейронные сети справляются с "*проклятием размерности*", которое не позволяет моделировать линейные зависимости в случае большого числа переменных
- **Простота в использовании.** Нейронные сети *учатся на примерах*. Пользователь нейронной сети подбирает представительные данные, а затем запускает *алгоритм обучения*, который автоматически воспринимает структуру данных. При этом от пользователя, конечно, требуется какой-то набор эвристических знаний о том, как следует отбирать и подготавливать данные, выбирать нужную архитектуру сети и интерпретировать результаты, однако уровень знаний, необходимый для успешного применения нейронных сетей, гораздо скромнее, чем, например, при использовании традиционных методов статистики.

Нейронные сети привлекательны с интуитивной точки зрения, ибо они основаны на примитивной биологической модели нервных систем. В будущем развитие таких нейро-биологических моделей может привести к созданию действительно мыслящих компьютеров. Между тем уже "простые" нейронные сети, которые строит система *ST Neural Networks* , являются мощным оружием в арсенале специалиста по прик

Теперь, чтобы понять, как же работают нейронные сети, давайте взглянем на ее составляющие и их параметры.

Что такое нейрон?

Нейрон — это вычислительная единица, которая получает информацию, производит над ней простые вычисления и передает ее дальше. Они делятся на три основных типа: входной (синий), скрытый (красный) и выходной (зеленый). Также есть нейрон смещения и контекстный нейрон о которых мы поговорим в следующей статье. В том случае, когда нейросеть состоит из большого количества нейронов, вводят термин слоя. Соответственно, есть входной слой, который получает информацию, n скрытых слоев (обычно их не больше 3), которые ее обрабатывают и выходной слой, который выводит результат. У каждого из нейронов есть 2 основных параметра: входные данные (input data) и выходные

данные (output data). В случае входного нейрона: input=output. В остальных, в поле input попадает суммарная информация всех нейронов с предыдущего слоя, после чего, она нормализуется, с помощью функции активации (пока что просто представим ее f(x)) и попадает в поле output.

Что такое синапс?

Синапс это связь между двумя нейронами. У синапсов есть 1 параметр — вес. Благодаря ему, входная информация изменяется, когда передается от одного нейрона к другому. Допустим, есть 3 нейрона, которые передают информацию следующему. Тогда у нас есть 3 веса, соответствующие каждому из этих нейронов. У того нейрона, у которого вес будет больше, та информация и будет доминирующей в следующем нейроне (пример — смешение цветов). На самом деле, совокупность весов нейронной сети или матрица весов — это своеобразный мозг всей системы. Именно благодаря этим весам, входная информация обрабатывается и превращается в результат.

Основная часть

Нейронные сети возникли из исследований в области искусственного интеллекта, а именно, из попыток воспроизвести способность биологических нервных систем обучаться и исправлять ошибки, моделируя низкоуровневую структуру мозга (Patterson, 1996). Основной областью исследований по искусственному интеллекту в 60-е - 80-е годы были экспертные системы. Такие системы основывались на высокоуровневом моделировании процесса мышления (в частности, на представлении, что процесс нашего мышления построен на манипуляциях с символами). Скоро стало ясно, что подобные системы, хотя и могут принести пользу в некоторых областях, не ухватывают некоторые ключевые аспекты человеческого интеллекта. Согласно одной из точек зрения, причина этого состоит в том, что они не в состоянии воспроизвести структуру мозга. Чтобы создать искусственных интеллект, необходимо построить систему с похожей архитектурой.

Мозг состоит из очень большого числа (приблизительно 10,000,000,000) нейронов, соединенных многочисленными связями (в среднем несколько тысяч связей на один нейрон, однако это число может сильно колебаться). Нейроны - это

специальная клетки, способные распространять электрохимические сигналы. Нейрон имеет разветвленную структуру ввода информации (дендриты), ядро и разветвляющийся выход (аксон). Аксоны клетки соединяются с дендритами других клеток с помощью синапсов. При активации нейрон посылает электрохимический сигнал по своему аксону. Через синапсы этот сигнал достигает других нейронов, которые могут в свою очередь активироваться. Нейрон активируется тогда, когда суммарный уровень сигналов, пришедших в его ядро из дендритов, превысит определенный уровень (порог активации).

.Интенсивность сигнала, получаемого нейроном (а следовательно и возможность его активации), сильно зависит от активности синапсов. Каждый синапс имеет протяженность, и специальные химические вещества передают сигнал вдоль него. Один из самых авторитетных исследователей нейросистем, Дональд Хебб, высказал постулат, что обучение заключается в первую очередь в изменениях "силы" синаптических связей. Например, в классическом опыте Павлова, каждый раз непосредственно перед кормлением собаки звонил колокольчик, и собака быстро научилась связывать звонок колокольчика с пищей. Синаптические связи между участками коры головного мозга, ответственными за слух, и слюнными железами усилились, и при возбуждении коры звуком колокольчика у собаки начиналось слюноотделение.

Таким образом, будучи построен из очень большого числа совсем простых элементов (каждый из которых берет взвешенную сумму входных сигналов и в случае, если суммарный вход превышает определенный уровень, передает дальше двоичный сигнал), мозг способен решать чрезвычайно сложные задачи. Разумеется, мы не затронули здесь многих сложных аспектов устройства мозга, однако интересно то, что искусственные нейронные сети способны достичь замечательных результатов, используя модель, которая ненамного сложнее, чем описанная выше.

Нейронная сеть — попытка с помощью математических моделей воспроизвести работу человеческого мозга для создания машин, обладающих искусственным интеллектом.

Искусственная нейронная сеть обычно обучается с учителем. Это означает наличие обучающего набора (датасета), который содержит примеры с истинными значениями: тегами, классами, показателями.

Неразмеченные наборы также используют для обучения нейронных сетей, но мы не будем здесь это рассматривать.

Например, если вы хотите создать нейросеть для оценки тональности текста, датасетом будет список предложений с соответствующими каждому эмоциональными оценками. Тональность текста определяют признаки (слова, фразы, структура предложения), которые придают негативную или позитивную окраску. Веса признаков в итоговой оценке тональности текста (позитивный, негативный, нейтральный) зависят от математической функции, которая вычисляется во время обучения нейронной сети.

Раньше люди генерировали признаки вручную. Чем больше признаков и точнее подобраны веса, тем точнее ответ. Нейронная сеть автоматизировала этот процесс.

См. приложение 1

Искусственная нейронная сеть состоит из трех компонентов:

- Входной слой;
- Скрытые (вычислительные) слои;
- Выходной слой.

См. приложение 2

Обучение нейросетей происходит в два этапа:

- Прямое распространение ошибки;
- Обратное распространение ошибки.

Во время прямого распространения ошибки делается предсказание ответа. При обратном распространении ошибка между фактическим ответом и предсказанным минимизируется.

Прямое распространение ошибки

См. приложение 3

Зададим начальные веса случайным образом:

- w1
- w2

- w3

Умножим входные данные на веса для формирования скрытого слоя:

- h1 = (x1 * w1) + (x2 * w1)
- h2 = (x1 * w2) + (x2 * w2)
- h3 = (x1 * w3) + (x2 * w3)

Выходные данные из скрытого слоя передается через нелинейную функцию (функцию активации), для получения выхода сети:

- y_ = fn(h1 , h2, h3)

Обратное распространение

см. Приложение 4

- Суммарная ошибка (total_error) вычисляется как разность между ожидаемым значением «y» (из обучающего набора) и полученным значением «y_» (посчитанное на этапе прямого распространения ошибки), проходящих через функцию потерь (cost function).
- Частная производная ошибки вычисляется по каждому весу (эти частные дифференциалы отражают вклад каждого веса в общую ошибку (total_loss)).
- Затем эти дифференциалы умножаются на число, называемое скорость обучения или learning rate (η).

Полученный результат затем вычитается из соответствующих весов.

В результате получатся следующие обновленные веса:

- w1 = w1 — (η * ∂(err) / ∂(w1))
- w2 = w2 — (η * ∂(err) / ∂(w2))
- w3 = w3 — (η * ∂(err) / ∂(w3))

То, что мы предполагаем и инициализируем веса случайным образом, и они будут давать точные ответы, звучит не вполне обоснованно, тем не менее, работает хорошо.

Если вы знакомы с рядами Тейлора, обратное распространение ошибки имеет такой же конечный результат. Только вместо бесконечного ряда мы пытаемся

оптимизировать только его первый член.

Смещения – это веса, добавленные к скрытым слоям. Они тоже случайным образом инициализируются и обновляются так же, как скрытый слой. Роль скрытого слоя заключается в том, чтобы определить форму базовой функции в данных, в то время как роль смещения – сдвинуть найденную функцию в сторону так, чтобы она частично совпала с исходной функцией.

Частные производные

Частные производные можно вычислить, поэтому известно, какой был вклад в ошибку по каждому весу. Необходимость производных очевидна. Представьте нейронную сеть, пытающуюся найти оптимальную скорость беспилотного автомобиля. Если машина обнаружит, что она едет быстрее или медленнее требуемой скорости, нейронная сеть будет менять скорость, ускоряя или замедляя автомобиль. Что при этом ускоряется/замедляется? Производные скорости.

Разберем необходимость частных производных на примере.

Предположим, детей попросили бросить дротик в мишень, целясь в центр. Вот результаты:

Теперь, если мы найдем общую ошибку и просто вычтем ее из всех весов, мы обобщим ошибки, допущенные каждым. Итак, скажем, ребенок попал слишком низко, но мы просим всех детей стремиться попадать в цель, тогда это приведет к следующей картине:

Ошибка нескольких детей может уменьшиться, но общая ошибка все еще увеличивается.

Найдя частные производные, мы узнаем ошибки, соответствующие каждому весу в отдельности. Если выборочно исправить веса, можно получить следующее:

Гиперпараметры

Нейронная сеть используется для автоматизации отбора признаков, но некоторые параметры настраиваются вручную.

Скорость обучения (learning rate)

Скорость обучения является очень важным гиперпараметром. Если скорость обучения слишком мала, то даже после обучения нейронной сети в течение длительного времени она будет далека от оптимальных результатов. Результаты будут выглядеть примерно так:

С другой стороны, если скорость обучения слишком высока, то сеть очень быстро выдаст ответы. Получится следующее:

См. Приложение 4,5,6,7,8

Функция активации (activation function)

Функция активации — это один из самых мощных инструментов, который влияет на силу, приписываемую нейронным сетям. Отчасти, она определяет, какие нейроны будут активированы, другими словами и какая информация будет передаваться последующим слоям.

Без функций активации глубокие сети теряют значительную часть своей способности к обучению. Нелинейность этих функций отвечает за повышение степени свободы, что позволяет обобщать проблемы высокой размерности в более низких измерениях.

Функция потери (loss function)

Функция потерь находится в центре нейронной сети. Она используется для расчета ошибки между реальными и полученными ответами. Наша глобальная цель — минимизировать эту ошибку. Таким образом, функция потерь эффективно приближает обучение нейронной сети к этой цели.

Функция потерь измеряет «насколько хороша» нейронная сеть в отношении данной обучающей выборки и ожидаемых ответов. Она также может зависеть от таких переменных, как веса и смещения.

Функция потерь одномерна и не является вектором, поскольку она оценивает, насколько хорошо нейронная сеть работает в целом.

Некоторые известные функции потерь:

- Квадратичная (среднеквадратичное отклонение);
- Кросс-энтропия;
- Экспоненциальная (AdaBoost);
- Расстояние Кульбака — Лейблера или прирост информации.

Функция потерь в нейронной сети должна удовлетворять двум условиям:

- Функция потерь должна быть записана как среднее;
- Функция потерь не должна зависеть от каких-либо активационных значений нейронной сети, кроме значений, выдаваемых на выходе.

Глубокие нейронные сети

Глубокое обучение (deep learning) – это класс алгоритмов машинного обучения, которые учатся глубже (более абстрактно) понимать данные. Популярные алгоритмы нейронных сетей глубокого обучения представлены на схеме ниже.

См.приложение 9

Более формально в deep learning:

- Используется каскад (пайплайн, как последовательно передаваемый поток) из множества обрабатывающих слоев (нелинейных) для извлечения и преобразования признаков;
- Основывается на изучении признаков (представлении информации) в данных без обучения с учителем. Функции более высокого уровня (которые находятся в последних слоях) получаются из функций нижнего уровня (которые находятся в слоях начальных слоях);
- Изучает многоуровневые представления, которые соответствуют разным уровням абстракции; уровни образуют иерархию представления.

Проклятье размерности

Проклятие размерности относится к различным явлениям, возникающим при анализе и организации данных в многомерных пространствах (часто с сотнями или тысячами измерений), и не встречается в ситуациях с низкой размерностью.

Грамматика английского языка имеет огромное количество аттрибутов, влияющих на нее. В машинном обучении мы должны представить их признаками в виде массива/матрицы конечной и существенно меньшей длины (чем количество существующих признаков). Для этого сети обобщают эти признаки. Это порождает две проблемы:

- Из-за неправильных предположений появляется смещение. Высокое смещение может привести к тому, что алгоритм пропустит существенную взаимосвязь между признаками и целевыми переменными. Это явление называют недообучение.
- От небольших отклонений в обучающем множестве из-за недостаточного изучения признаков увеличивается дисперсия. Высокая дисперсия ведет к переобучению, ошибки воспринимаются в качестве надежной информации.

Заключение

На ранней стадии обучения смещение велико, потому что выход из сети далек от желаемого. А дисперсия очень мала, поскольку данные имеет пока малое влияние.

В конце обучения смещение невелико, потому что сеть выявила основную функцию в данных. Однако, если обучение слишком продолжительное, сеть также изучит шум, характерный для этого набора данных. Это приводит к большому разбросу результатов при тестировании на разных множествах, поскольку шум меняется от одного набора данных к другому.

Действительно, алгоритмы с большим смещением обычно в основе более простых моделей, которые не склонны к переобучению, но могут недообучиться и не выявить важные закономерности или свойства признаков. Модели с маленьким смещением и большой дисперсией обычно более сложны с точки зрения их структуры, что позволяет им более точно представлять обучающий набор. Однако они могут отображать много шума из обучающего набора, что делает их прогнозы менее точными, несмотря на их дополнительную сложность.

Следовательно, как правило, невозможно иметь маленькое смещение и маленькую дисперсию одновременно.

Сейчас есть множество инструментов, с помощью которых можно легко создать сложные модели машинного обучения, переобучение занимает центральное место.

Поскольку смещение появляется, когда сеть не получает достаточно информации. Но чем больше примеров, тем больше появляется вариантов зависимостей и изменчивостей в этих корреляциях.

Выбирайте такие переменные, которые, как Вы предполагаете, влияют на результат.

С числовыми и номинальными переменными в пакете *ST Neural Networks* можно работать непосредственно. Переменные других типов следует преобразовать в указанные типы или объявить незначащими.

Для анализа нужно иметь порядка сотен или тысяч наблюдений; чем больше в задаче переменных, тем больше нужно иметь наблюдений. Пакет *ST Neural Networks* имеет средства для распознавания значимых переменных, поэтому включайте в рассмотрение переменные, в значимости которых Вы не уверены.

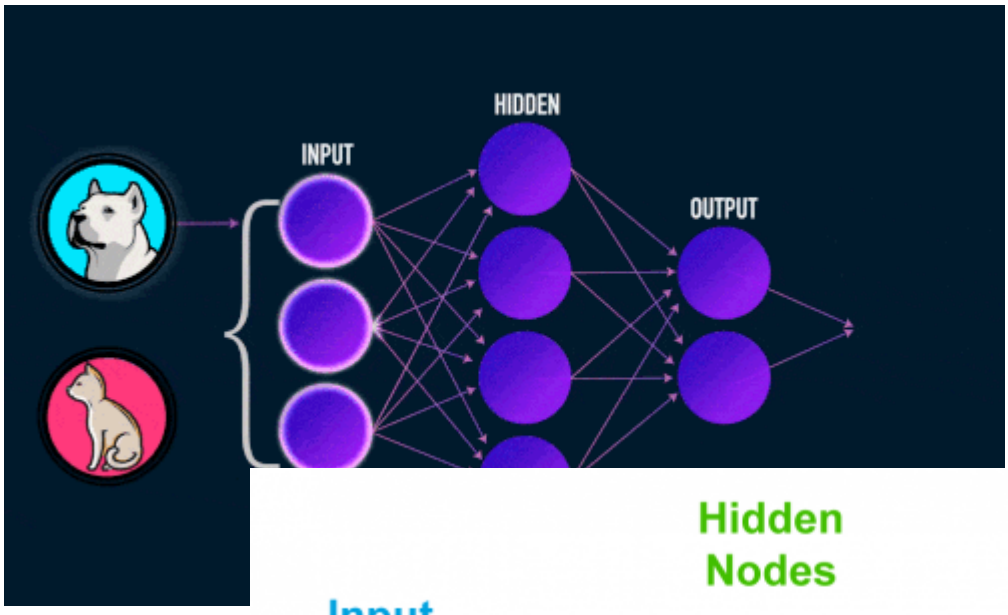
В случае необходимости можно работать с наблюдениями, содержащими пропущенные значения. Наличие выбросов в данных может создать трудности. Если возможно, удалите выбросы. Если данных достаточное количество, уберите из рассмотрения наблюдения с пропущенными значениями.

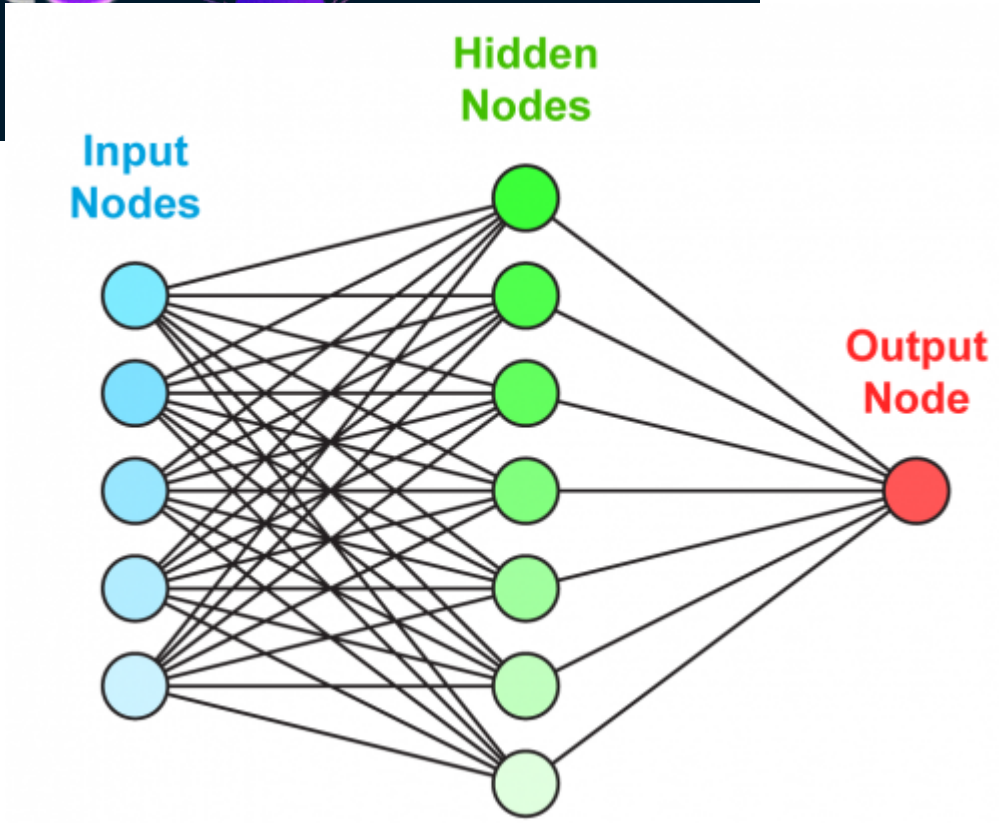
Список литературы

- *Беркинблит М. Б.* Нейронные сети. — М.: МИРОС и ВЗМШ РАО, 1993. — 96 с. — ISBN 5-7084-0026-9. Архивная копия от 12 мая 2011 на Wayback Machine
- *Вороновский Г. К., Махотило К. В., Петрашев С. Н., Сергеев С. А.* Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. — Харьков: Основа, 1997. — 112 с. — ISBN 5-7768-0293-8.
- Голубев Ю. Ф. Нейросетевые методы в мехатронике. — М.: Изд-во Моск. унта, 2007. — 157 с. — ISBN 978-5-211-05434-9.
- *Горбань А.Н.* Обучение нейронных сетей. — М.: СССР-США СП «Параграф», 1990. — 160 с.
- *Горбань А.Н., Россиев Д.А.* Нейронные сети на персональном компьютере. — Новосибирск: Наука, 1996. — 276 с. — ISBN 5-02-031196-0.
- *Горбань А. Н., Дунин-Барковский В. Л. и др.* Нейроинформатика. — Новосибирск: Наука, 1998.

Приложения

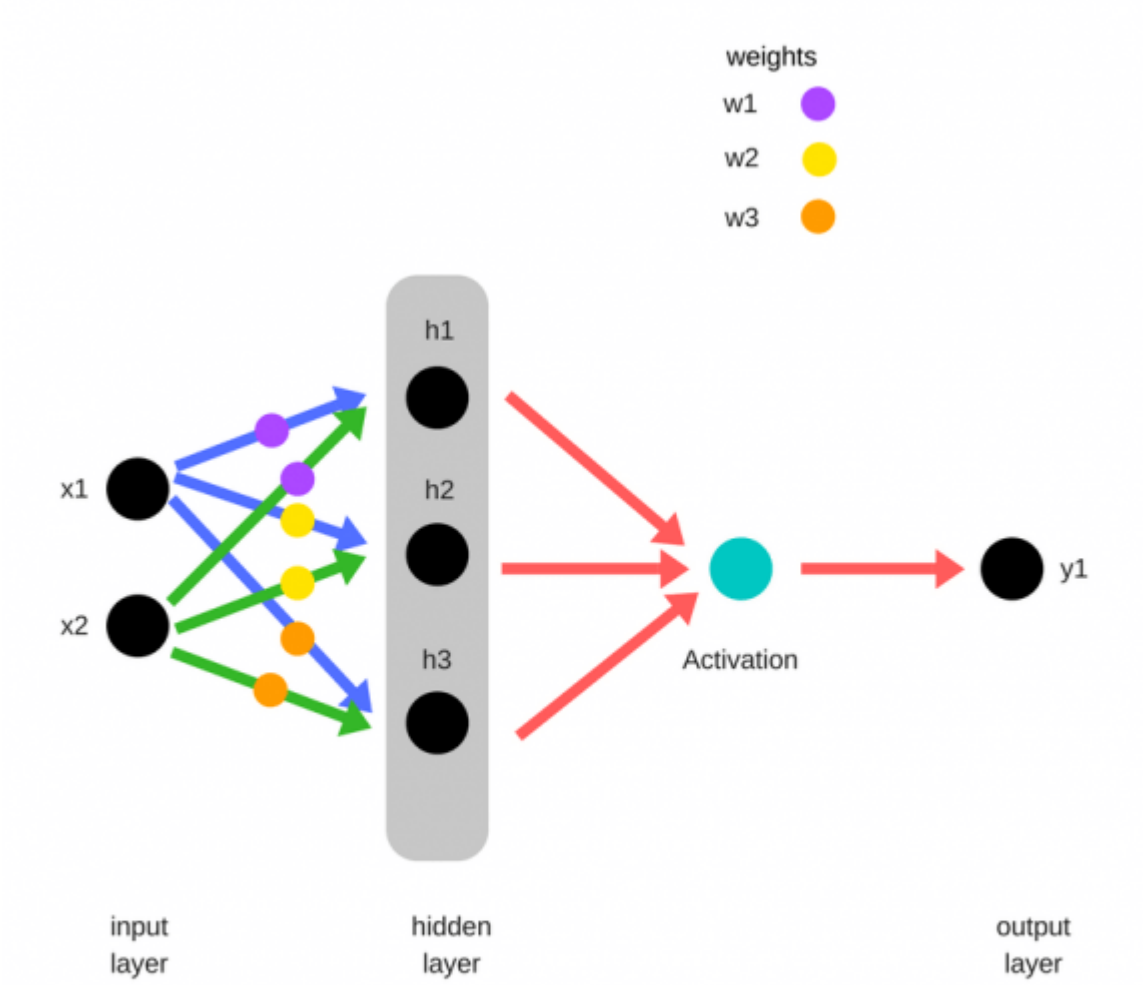
Приложение 1



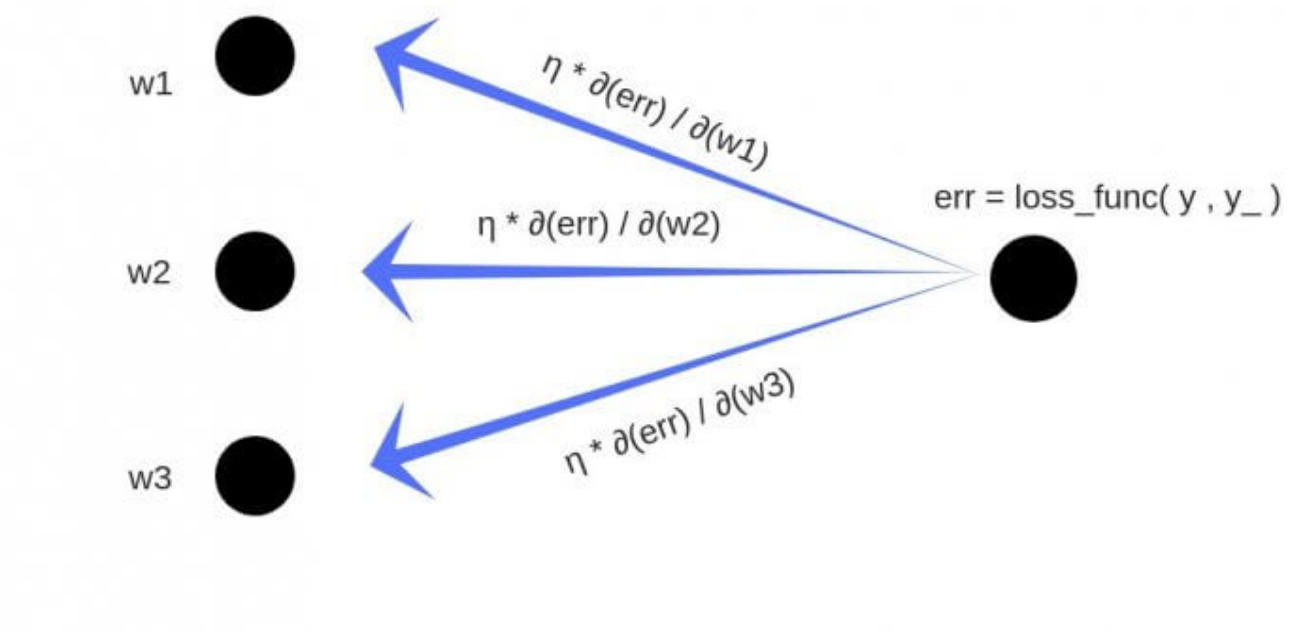


Приложение 2

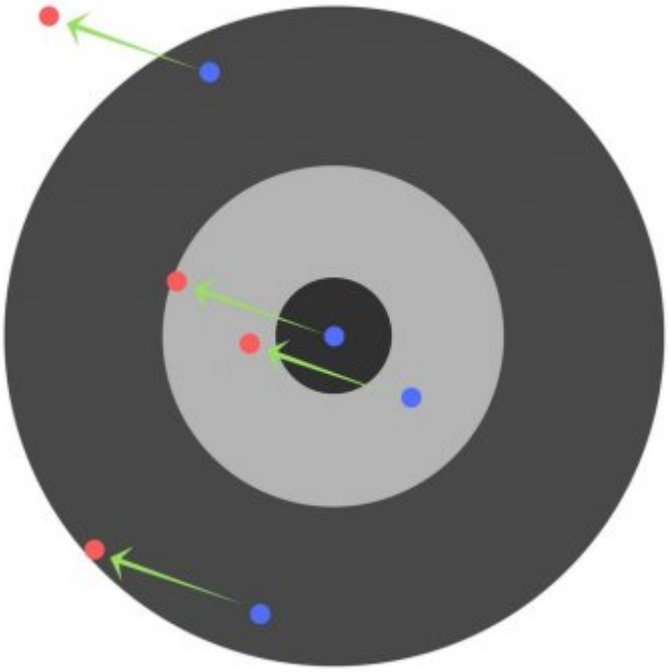
Приложение 3

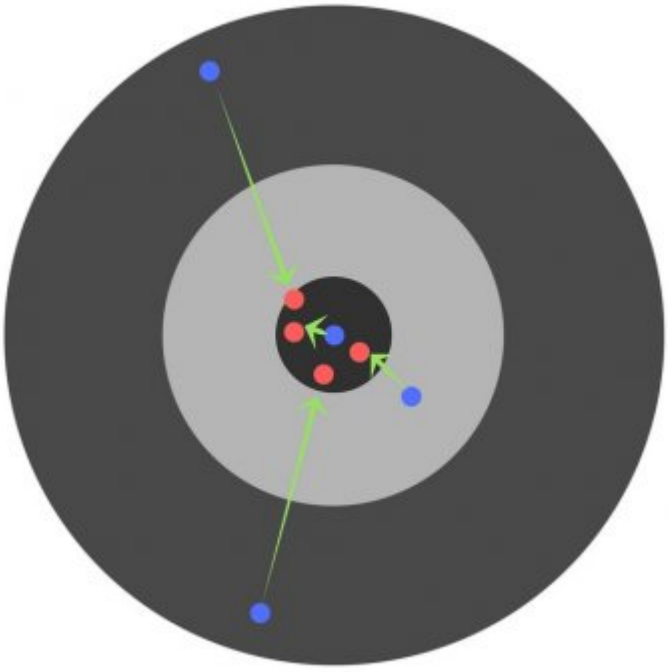


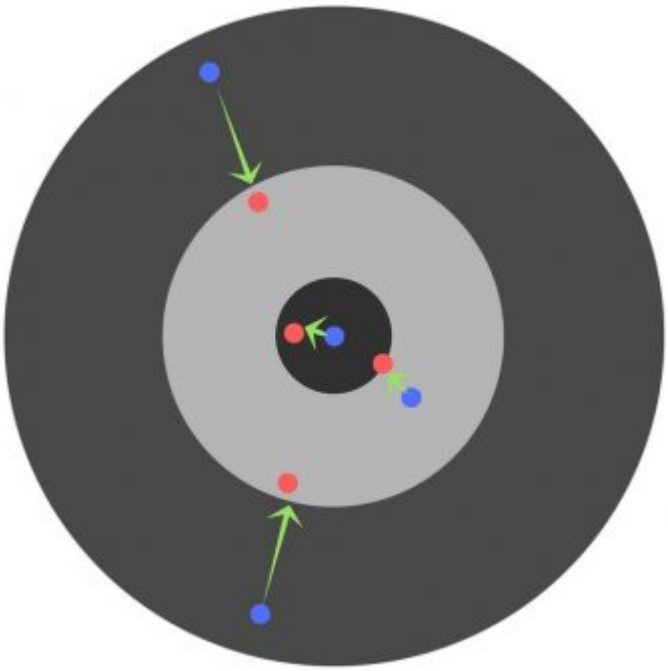
Приложение 4

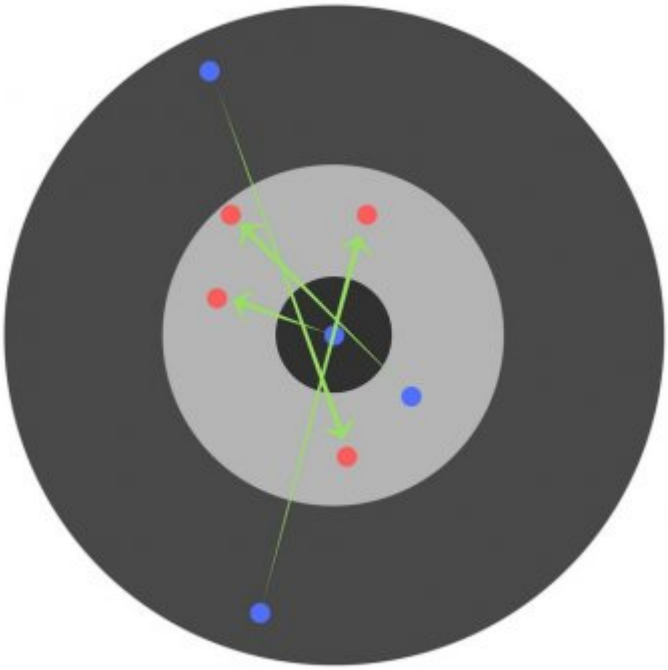


Приложение 4,5,6,7,8









Приложение 9

A mostly complete chart of

Neural Networks

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool

Perceptron (P)

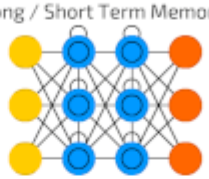
Feed Forward (FF)

Radial Basis Network (RBF)

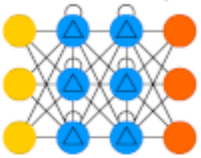
Deep Feed Forward (DFF)

Recurrent Neural Network (RNN)

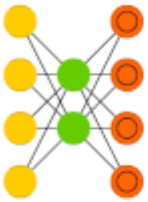
Long / Short Term Memory (LSTM)



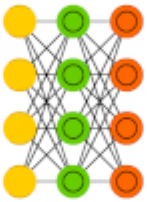
Gated Recurrent Unit (GRU)



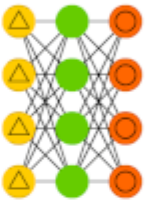
Auto Encoder (AE)



Variational AE (VAE)



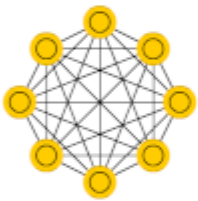
Denoising AE (DAE)



Sparse AE (SAE)

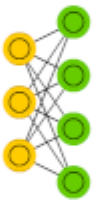
Markov Chain (MC)

Hopfield Network (HN)



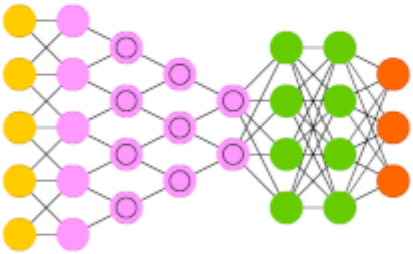
Boltzmann Machine (BM)

Restricted BM (RBM)

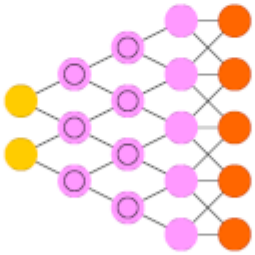


Deep Belief Network (DBN)

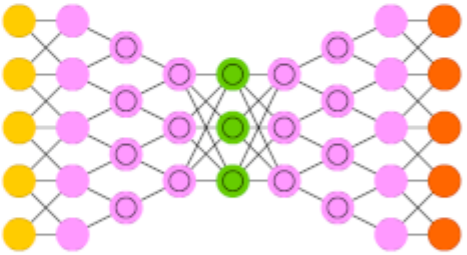
Deep Convolutional Network (DCN)



Deconvolutional Network (DN)



Deep Convolutional Inverse Graphics Network (DCIGN)



Generative Adversarial Network (GAN)

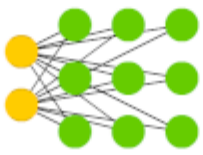
Liquid State Machine (LSM)

Extreme Learning Machine (ELM)

Echo State Network (ESN)

Deep Residual Network (DRN)

Kohonen Network (KN)



Support Vector Machine (SVM)

Neural Turing Machine (NTM)