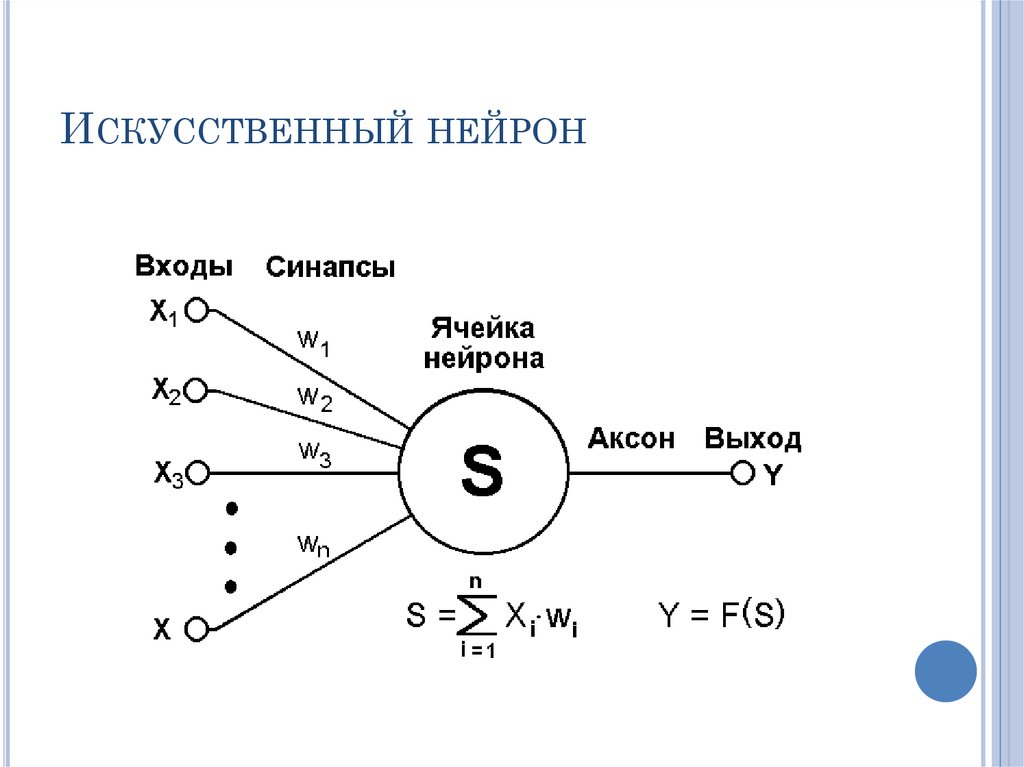
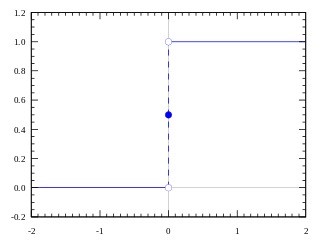
Нейронные сети

1. Биологический нейрон, нейронная сеть человеческого мозга. Искусственный нейрон: общая схема модели – входные сигналы, весовые коэффициенты (синаптические веса), суммирующий блок (сумматор), функция активации, выходной сигнал.

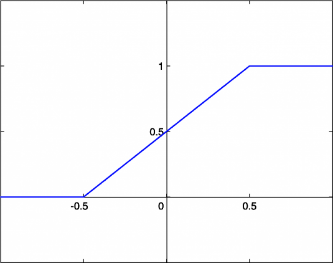


Математическая модель нейрона: , где – -ый входной сигнал; – вес *i*-ого входного сигнала; – функция активации, *y* – выходной сигнал.

1. Различные виды функции активации: линейная, ступенчатая, логистическая, гиперболический тангенс, выпрямленная функция линейного блока ReLu, функция Softmax. Роль функций активации в нейронной сети.

Функция активации нейрона определяет выходной сигнал, который определяется входным сигналом или набором входных сигналов.

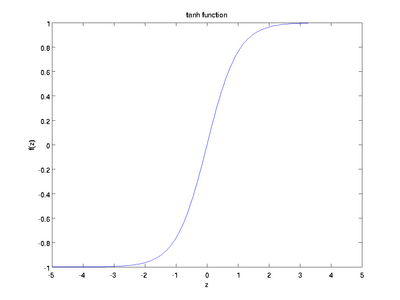
Ступенчатая. A = 1, если Y > граница, иначе А = 0. (бинарный классификатор)



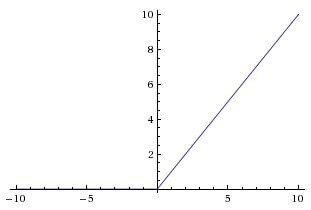
Линейная. Линейная функция представляет собой прямую линию и пропорциональна входу (то есть взвешенной сумме на этом нейроне).

Если вы знакомы с методом градиентного спуска для обучения, то можете заметить, что для этой функции производная равна постоянной.

Не имеет значения, сколько слоев мы имеем. Если все они по своей природе линейные, то финальная функция активации в последнем слое будет просто линейной функцией от входов на первом слое! Остановитесь на мгновение и обдумайте эту мысль.



Гиперболический тангенс стремиться привести значения к одной из сторон кривой (например, к верхнему при х=2 и нижнему при х=-2). Такое поведение позволяет находить четкие границы при предсказании.



ReLu. Возвращает значение х, если х положительно, и 0 в противном случае. ReLu нелинейна по своей природе, а комбинация ReLu также нелинейна. В идеале мы хотим, чтобы некоторые нейроны не были активированы, это сделало бы активации разреженными и эффективными.

1. Персептрон Розеблатта. Процедура обучения персептрона. Решение задачи линейной классификации – отнесение входных сигналов к одному из двух классов и .

Бинарная классификация.

y = w \* x + b

На выходе персептрон выдает результат y, основанный на нескольких вещественных входных объектах путем формирования линейной комбинации с использованием весовых коэффициентов (иногда с последующим пропусканием результата через нелинейную функцию активации).

1. Сети прямого распространения. Однослойные нейросети. Представление нейронных сетей с помощью ориентированных графов. Процедура обучения сети. Эпохи, батчи (пакеты), итерации.

Сети прямого распространения – сети, в которых сигнал распространяется строго от входного слоя к выходному. В обратном направлении сигнал не распространяется.

прогнозирование, кластеризация и распознавание

1. Многослойный персептрон. Многослойные нейросети прямого распространения. Алгоритм обратного распространения ошибки.

А в сетях с обратными связями выходы нейронов могут возвращаться на входы. Это означает, что выход какого-нибудь нейрона определяется не только его весами и входным сигналом, но еще и предыдущими выходами (так как они снова вернулись на входы).

Цели обратного распространения просты: отрегулировать каждый вес пропорционально тому, насколько он способствует общей ошибке. Если мы будем итеративно уменьшать ошибку каждого веса, в конце концов у нас будет ряд весов, которые дают хорошие прогнозы.

1. Слой Кохонена. Обучение слоя без учителя. Нормализация входных векторов. Выбор начальных значений весовых векторов и их нормализация.

Сигнал распространяется от входов к выходам в прямом направлении. Структура нейронной сети содержит единственный слой нейронов (слой Кохонена) без коэффициентов смещения.

Количество нейронов равно количеству кластеров, среди которых происходит начальное распределение и последующее перераспределение обучающих примеров. Количество входных переменных нейронной сети равно числу признаков, характеризующих объект исследования и на основе которых происходит отнесение его к одному из кластеров.

В своей простейшей форме слой Кохонена

функционирует в духе «победитель забирает все», т. е. для

данного входного вектора один и только один нейрон Кохонена

выдает на выходе логическую единицу, все остальные выдают

нуль.

При обычном самообучении сеть имеет строго фиксированную структуру, т. е. количество нейронов, не изменяющееся на протяжении всего жизненного цикла. При самоорганизации сеть, напротив, не имеет постоянной структуры. В зависимости от найденного расстояния до нейрона-победителя либо этот нейрон используется для кластеризации примера, либо для поданного на входы примера создается новый кластер с соответствующими ему весовыми коэффициентами. Кроме того, в процессе самоорганизации структуры сети Кохонена отдельные нейроны могут исключаться из нее.

Алгоритм **обучения**сети Кохонена включает этапы, состав которых зависит от типа структуры: постоянной (самообучающаяся сеть) или переменной (самоорганизующаяся сеть). Для **самообучения**последовательно выполняются:

1. Задание структуры сети (количества нейронов слоя Кохонена) (K).

2. Случайная инициализация весовых коэффициентов значениями, удовлетворяющими одному из следующих ограничений:

– при нормализации исходной выборки в пределах [–1, 1]:

https://neuronus.com/images/theory/ins/05062015/03.jpg                                              (1)

– при нормализации исходной выборки в пределах [0, 1]:

https://neuronus.com/images/theory/ins/05062015/04.jpg,                            (2)

где M – количество входных переменных сети – характеристических признаков объекта исследования.

3. Подача на входы сети случайного обучающего примера текущей эпохи обучения и расчет евклидовых расстояний от входного вектора до центров всех кластеров:

https://neuronus.com/images/theory/ins/05062015/05.jpg.                                (3)

4. По наименьшему из значений Rj выбирается нейрон-победитель j, в наибольшей степени близкий по значениям с входным вектором. Для выбранного нейрона (и только для него) выполняется коррекция весовых коэффициентов:

https://neuronus.com/images/theory/ins/05062015/06.jpg,                         (4)

где v – коэффициент скорости обучения.

5. Цикл повторяется с шага 3 до выполнения одного или нескольких условий окончания:

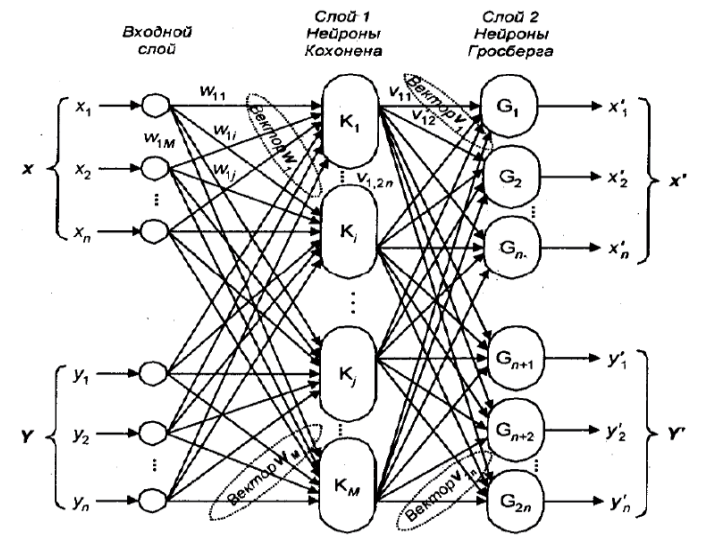
– исчерпано заданное предельное количество эпох обучения;

– не произошло значимого изменения весовых коэффициентов в пределах заданной точности на протяжении последней эпохи обучения;

– исчерпано заданное предельное физическое время обучения.

1. Сети встречного распространения.

Сеть ВР состоит из двух слоев нейронов - слоя Кохонена и слоя Гроссберга. В режиме функционирования (распознавания) нейроны слоя Кохонена работают по принципу Победитель-Забирает-Все, определяя кластер, к которому принадлежит входной образ. Затем выходная звезда слоя Гроссберга по сигналу нейрона-победителя в слое Кохонена воспроизводит на выходах сети соответствующий образ.



В процессе обучения входные векторы ассоциируются с соответствующими выходными векторами. Эти векторы могут быть двоичными, состоящими из нулей и единиц, или непрерывными. Когда сеть обучена, приложение входного вектора приводит к требуемому

выходному вектору. Обобщающая способность сети позволяет получать правильный выход даже при приложении входного вектора, который является неполным или слегка неверным. Это позволяет использовать данную сеть для распознавания образов, восстановления образов и усиления сигналов.

Фактически каждый нейрон слоя

Гроссберга лишь выдает величину веса, который связывает этот

нейрон с единственным ненулевым нейроном Кохонена.

1. Сверточные нейронные сети (Convolutional Neural Networks). Архитектура сверточной нейронной сети, сверточные слои (convolutional layers), слои подвыборки (pooling-слои), полносвязные слои (fully-connected, dense layers). Процедура обучения.
2. Сверточный слой. Фильтр (разница для одноканальных и многоканальных изображений), ядро свертки (кернел), операция свертки, карта признаков, паддинг, страйд. Слой подвыборки (MaxPooling, MinPooling, AveragePooling).

ядра, представляющего из себя матрицу весов (weight matrix). Ядро “скользит” над двумерным изображением, поэлементно выполняя операцию умножения с той частью входных данных, над которой оно сейчас находится, и затем суммирует все полученные значения в один выходной пиксель.

размер ядра сверточной нейронной сети определяет количество признаков, которые будут объединены для получения нового признака на выходе.

Padding добавляет к краям поддельные (fake) пиксели (обычно нулевого значения, вследствие этого к ним применяется термин “нулевое дополнение” — “zero padding”). Таким образом, ядро при проскальзывании позволяет неподдельным пикселям оказываться в своем центре, а затем распространяется на поддельные пиксели за пределами края, создавая выходную матрицу того же размера, что и входная.

Идея stride заключается в том, чтобы пропустить некоторые области, над которыми скользит ядро. Шаг 1 означает, что берутся пролеты через пиксель, то есть по факту каждый пролет является стандартной сверткой. Шаг 2 означает, что пролеты совершаются через каждые два пикселя, пропуская все другие пролеты в процессе и уменьшая их количество примерно в 2 раза, шаг 3 означает пропуск 3-х пикселей, сокращая количество в 3 раза и т.д.

Для многоканальных:

Каждый фильтр на самом деле представляет собой коллекцию ядер, причем для каждого отдельного входного канала этого слоя есть одно ядро, и каждое ядро уникально. (например, фильтр может задать красному каналу ядра больший вес, чем другим каналам, и, следовательно, больше реагировать на различия в образах из красного канала). Затем каждая из обработанных в канале версий суммируется вместе для формирования одного канала. Ядра каждого фильтра генерируют одну версию каждого канала, а фильтр в целом создает один общий выходной канал. Наконец, каждый выходной файл имеет свое смещение. Смещение добавляется к выходному каналу для создания конечного выходного канала

1. Различные типы ядер двумерной свертки: выделение границ (оператор Собеля, оператор Кэнни); размытие по Гауссу и т.п.

Кэнни: Алгоритм состоит из пяти отдельных шагов:

1. **Сглаживание**. Размытие изображения для удаления шума.
2. **Поиск градиентов**. Границы отмечаются там, где градиент изображения приобретает максимальное значение.
3. **Подавление не-максимумов**. Только локальные максимумы отмечаются как границы.
4. **Двойная пороговая фильтрация**. Потенциальные границы определяются порогами.
5. **Трассировка области неоднозначности**. Итоговые границы определяются путём подавления всех краёв, несвязанных с определенными (сильными) границами.

1. Детектор границ Кэнни

В начале алгоритма применяется первая производная Гауссиана для

уменьшения чувствительности алгоритма к шуму. После сглаживания на контуре границ изображения остаются только точки максимума градиента.

Детектор Кэнни использует четыре фильтра для выявления горизонтальных, вертикальных и диагональных границ для того, чтобы удалять точки именно рядом с границей и не разрывать саму границу вблизи локальных максимумов градиента. Далее с помощью двух порогов происходит удаление слабых границ. В итоге алгоритма при определении границ происходит подавление краев, которые не связаны с сильными границами.

2. Оператор Собеля

Данный оператор базируется на приближении значений градиента яркости изображения. Он вычисляет градиент яркости в каждой точке на изображении, тем самым находит величину изменения яркости и ее направление. Результат показывает изменения яркости изображения в каждой точке, т.е. вероятность нахождения точки на границы, а также ориентацию. Можно сказать, что результат оператора в точке, которая находится в постоянной яркости – это нулевой вектор. А точки, находящиеся в границе с различной яркостью – это вектор, который пересекает границу в направлении возрастания яркости. Оператор Собеля использует фильтрацию изображения на основе свертки по горизонтали и вертикали, поэтому он легко вычисляется. Оператор использует

матрицу 3х3, благодаря которым свертывают исходное изображение для дальнейшего вычисления приближенных производных по горизонтальным и вертикальным направлениям.

При применении в двух измерениях размфтие по гауссу даёт поверхность, контуры которой представляют собой концентрические окружности с нормальным распределением относительно центральной точки. Новое значение каждого пикселя устанавливается равным средневзвешенному значению окрестности этого пикселя. Значение исходного пикселя получает наибольший вес (имеющий наивысшее значение функции Гаусса), а соседние пиксели получают меньшие веса по мере увеличения их расстояния до исходного пикселя. Это приводит к размытию, которое сохраняет границы и края лучше, чем другие, более однородные фильтры размытия.

1. Библиотека TensorFlow, пакет Keras. Базовый класс keras.Model. Два основных способа описания моделей: последовательное описание ­– Sequential API, функциональное описание – Functional API. Класс Sequential. Базовый класс для описания слоев keras.layers.Layer. Слой Dense. Слой Flatten. Слой двумерной свертки Conv2D. Слой отключения Dropout. Слой подвыборки (объединения) MaxPooling2D.

Давайте рассмотрим следующую модель (будем строим ее с помощью Functional API, но она может быть и Sequential или субклассированной моделью):

**from** tensorflow **import** keras

**from** tensorflow.keras **import** layers

inputs = keras.Input(shape=(784,), name='digits')

x = layers.Dense(64, activation='relu', name='dense\_1')(inputs)

x = layers.Dense(64, activation='relu', name='dense\_2')(x)

outputs = layers.Dense(10, activation='softmax', name='predictions')(x)

model = keras.Model(inputs=inputs, outputs=outputs)

model = tf.keras.Sequential()

model.add(tf.keras.layers.Dense(8))

model.add(tf.keras.layers.Dense(1))

model.compile(optimizer='sgd', loss='mse')

# This builds the model for the first time:

model.fit(x, y, batch\_size=32, epochs=10)

Dropout состоит в случайной установке доли единиц ввода в 0 при каждом обновлении во время обучения, что помогает предотвратить переобучение (оверфиттинг).

1. Методы Keras: методы предварительной обработки данных (в том числе Keras.np\_utils.to\_categorical()), метод add(), метод compile(), метод fit(), метод evaluate(), метод predict ().

Keras.np\_utils.to\_categorical()) Преобразует вектор класса (целые числа) в двоичную матрицу класса.

evaluate Returns the loss value & metrics values for the model in test mode.

predict Generates output predictions for the input samples.