**Лекция 6. Одномерные сверточные сети**

План:

[Одномерные сверточные нейросети 2](#_Toc145414879)

[Одномерная операция свертки 2](#_Toc145414880)

[Одномерная операция свертки для анализа текста 3](#_Toc145414881)

[Одномерная свертка для обработка векторов 4](#_Toc145414882)

[Ядра свертки 5](#_Toc145414883)

[Одномерные сверточные сети в Keras и TensorFlow 7](#_Toc145414884)

[Итоги 10](#_Toc145414885)

[Многозначная классификация текстов 11](#_Toc145414886)

[Практика 15](#_Toc145414887)

[Задачи для самостоятельной работы 16](#_Toc145414888)

[Литература 16](#_Toc145414889)

**Одномерные сверточные нейросети**

Мы уже обсуждали, что для анализа текста необходимо рассматривать его как последовательность токенов. Это связано с тем, что порядок слов и предложений в тексте имеет большое значение. Если мы не будем анализировать последовательность, то при анализе тональности, например, мы можем допустить ошибки, как в случае с фразами "Overall, the movie is not bad and has entertainment value" и "Unfortunately, the movie is not so good". Также некоторые словосочетания имеют значение, отличное от значения отдельных слов.

Мы уже рассматривали пример с мороженым "ice cream", которое состоит из двух слов "ice" и "cream". Ранее мы также рассмотрели архитектуру рекуррентных нейронных сетей, которая позволяет анализировать последовательности. В таких сетях значение на выходе нейрона передается на его вход.

Однако есть и другой способ анализировать тексты в виде последовательности - это одномерные сверточные нейронные сети. Сверточные нейронные сети, точнее их двумерный вариант, широко применяются для анализа изображений. Одномерные сверточные сети можно использовать для анализа текстов.

***Текст необходимо анализировать как последовательность токенов***

* ***Порядок слов/символов/предложений в тексте имеет большой смысл***

***Проблемы анализа текстов:***

* ***Overall, the movie is not bad and has entertainment value.***
* ***Unfortunately, the movie is not so good.***
* ***Ice cream (мороженое, ісе лед, cream - крем, сливки)***

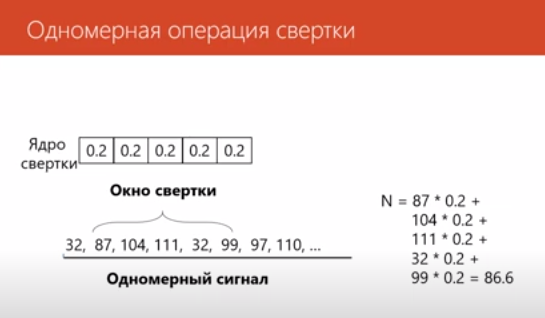
***Пути решения проблем***

* ***Рекуррентные нейронные сети LSTM и GRU***
* ***Одномерные сверточные нейронные сети***

**Одномерная операция свертки**

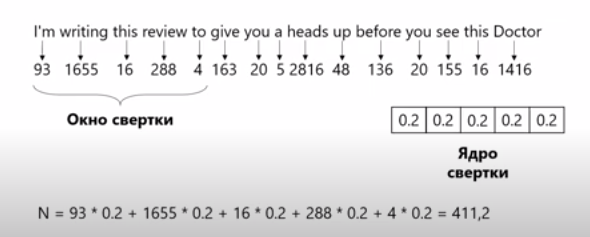
Давайте рассмотрим, как устроены сверточные нейронные сети и какие применения они имеют. Как можно понять из названия, сверточные нейронные сети используют операцию свертки. Для одномерной свертки входными данными служит последовательность чисел. Операция свертки реализуется на окне свертки - это фрагмент последовательности ограниченной длины. Как правило, для анализа текстов используют свертки длиной 3-5-7, иногда больше. Также для операции свертки нужно так называемое ядро свертки - набор чисел такой же длины, как и окно свертки. В нашем случае это число 5.

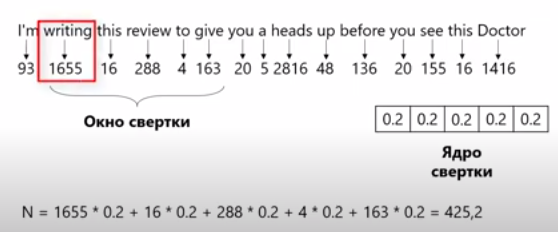
Как выполняется операция свертки? Первый элемент данных умножается на первый элемент ядра свертки, второй элемент данных умножается на второй элемент ядра свертки, и так выполняется, пока мы не дойдем до конца окна свертки и ядра свертки. Затем все полученные результаты складываются. Легко заметить, что формула свертки очень похожа на формулу работы искусственного нейрона. В этом случае наш одномерный сигнал выступает данными, а ядро свертки - весами входов в нейрон.



**Одномерная операция свертки для анализа текста**

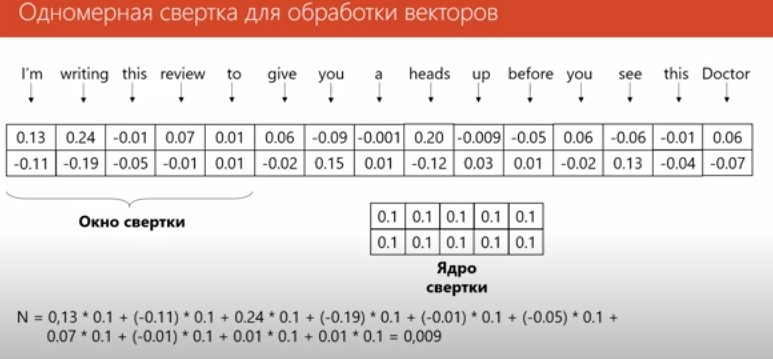
Ранее мы уже рассматривали, что перед обработкой текстов их необходимо представить в числовом виде. В простом случае мы используем частотное кодирование. Возьмем предложение на английском языке: "Я пишу этот отзыв, чтобы дать вам предупреждение перед тем, как вы пойдете к этому доктору". Здесь как раз пример использования словосочетания "Heads Up", которое меняет значение "Head" (голова) на "Up" (вверх) - предупреждение. Перед тем, как обрабатывать текст, мы его преобразуем в числа и затем используем одномерную свертку. Для этого берем некоторое ядро свертки и окно свертки. Пусть длина окна свертки и ядра свертки составляет 5. Мы выполняем операцию свертки над числовым представлением текста. Числовое представление первого слова "I am" (93) умножаем на первый элемент ядра свертки, числовое представление второго слова "доктор" (1655) умножаем на второй элемент ядра свертки (02). Выполняем такие же действия для трех оставшихся слов и соответствующих им чисел. Полученные результаты складываем. Затем мы сдвигаем окно свертки на один элемент вправо и повторяем те же самые действия. На этот раз начинаем уже со второго слова "доктор" и его числового представления (1655). После того, как закончили все вычисления, сдвигаем окно свертки дальше и повторяем вычисление. И так продолжается до тех пор, пока окно свертки не дойдет до конца нашего предложения.





**Одномерная свертка для обработка векторов**

Более сложный случай - если мы представляем текст не числами, а векторами, как это обычно и делается на практике. Предположим, мы представляем слова векторами всего лишь из двух элементов, как мы это делали в видео по определению тональности набора данных IMDb. Сейчас слово "I’am" представляем вектором, состоящим из двух чисел: 0.13 и -0.11. "Writing" также вектором из двух чисел: 0.24 и -0.19. Здесь вы можете посмотреть, какие векторы используются для представления других слов в предложении. Длина окна свертки, если мы используем векторное представление слов, остается прежней - 5. Однако размер ядра свертки увеличивается. Ядро свертки становится двумерным: по одной размерности длина такая же, как и длина окна свертки (5 элементов), вторая размерность равна размерности вектора (2 элемента). Каждое слово представляется вектором из двух элементов, поэтому размерность по направлению два. Всего видов сверток у нас 10 элементов. Как реализуется операция свертки в этом случае? Действие похоже на предыдущие. Мы берем первый элемент вектора первого слова, умножаем на левый верхний элемент ядра свертки. Затем второй элемент вектора первого слова умножаем на левый нижний элемент ядра свертки. Переходим ко второму слову, второму столбцу в ядре свертки. Берем первый элемент вектора, умножаем на верхний элемент ядра свертки, второй элемент вектора умножаем на нижний элемент ядра свертки. Таким образом, у нас получается всего 10 чисел, которые мы складываем для получения итогового результата. Теперь мы знаем, как работает одномерная свертка как в случае представления текста обычными числами, так и в случае использования векторного представления.



**Ядра свертки**

Как определить ядро свертки?

Однако основной вопрос - как же выбрать ядра свертки, которые позволят нам решать задачи анализа текстов, например, определение тональности и классификации? Для компьютерного зрения исследователи вручную подобрали много различных полезных ядер свертки - это были двумерные ядра для размытия изображений, для повышения его четкости, для выделения границ и для реализации большого количества других операций. Однако для анализа текстов вручную подобрать значение элементов ядра свертки сложно. Также сложно это сделать для реализации сложных операций обработки изображений. Преимущества нейронных сетей в том, что в них значение ядер свертки определяются автоматически, как для двумерного случая обработки изображений, так и для одномерного случая анализа текстов. Используется такой же подход обучения с учителем, который мы применяли ранее - метод обратного распространения ошибки. Однако применяется дополнительное ограничение: что веса всех входов в нейроны в сверточном слое должны быть одинаковые. И для того, чтобы обеспечить эффективный анализ данных текстов или изображений, используются не одно ядро свертки, а несколько. Каждое из этих ядер свертки выделяет характерные элементы из изображений или текстов. И несколько вариантов данных, которые получены из исходных данных с помощью обработки разными ядрами свертки, называются картами признаков.

Определение ядра свертки зависит от задачи и типа данных, с которыми вы работаете.

Ручной подбор ядер свертки для компьютерного зрения.

*Для компьютерного зрения и обработки изображений можно использовать следующие ядра свертки:*

* *Размытие: для размытия изображения можно использовать ядро свертки Гаусса, которое имеет вид матрицы с центром в середине и значениями, уменьшающимися по мере удаления от центра.*
* *Повышение четкости: для повышения четкости изображения можно использовать ядро свертки, которое увеличивает контрастность и резкость изображения. Например, можно использовать ядро свертки Шарра или Собеля.*
* *Выделение границ: для выделения границ на изображении можно использовать ядро свертки, которое подчеркивает различия в яркости между соседними пикселями. Например, можно использовать ядро свертки Кэнни или Превитта.*

*Ручной подбор ядер свертки может быть трудоемким процессом. Вместо этого можно использовать автоматический подбор ядер свертки с помощью методов оптимизации, таких как генетические алгоритмы или алгоритмы оптимизации на основе градиентного спуска.*

*В нейронных сетях веса в ядрах сверки определяются автоматически в процессе обучения*

* *Обучение с учителем*
* *Ограничение на веса при обучении: веса входов всех нейронов в сверточном слое должны быть одинаковые*
* *Используется несколько ядер свертки (карта признаков)*

**Слой подвыборки**

Сверточные сети, помимо слоев свертки, также используют слои подвыборки. Эти слои служат для снижения размерности и выделения основных элементов из изображений или текстов. Слой подвыборки работает достаточно просто: он не обучается, а выполняет какую-то из популярных операций, например, выбор максимального значения или выбор среднего значения. Сейчас чаще всего используется выбор максимального значения. Таким образом, сверточная сеть состоит из двух слоев: на первом этапе идет слой свертки, на котором выполняется операция свертки с длиной окна, например, 5 элементов, как показано на слайде. Каждый нейрон сверточного слоя анализирует 5 элементов входных данных, а затем идет слой подвыборки. Для слоя подвыборки также задается размер - чаще всего используется размер 2, но может быть и более крупный. Каждый нейрон слоя подвыборки анализирует некоторое количество нейронов предыдущего слоя - в нашем случае два - и выбирает максимальное или среднее значение в зависимости от того, какой слой подвыборки вы используете. В крупных сверточных сетях могут использоваться несколько блоков свертки-подвыборки, которые идут друг за другом.

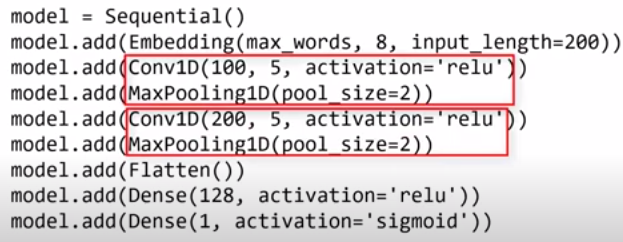


**Одномерные сверточные сети в Keras и TensorFlow**

Давайте рассмотрим, как использовать одномерные сверточные сети в TensorFlow и Keras. Снова будем использовать пример для определения тональности отзывов на фильмы из набора данных AMDB. Здесь представлен код создания модели, ссылка на ноутбук с коллабораторе с полным решением находится в описании видео. Сверточная нейронная сеть для анализа текстов состоит из трех частей. Первая часть используется для создания векторного представления слов - слой Embedding. Вторая часть - сверточная - используется для изучения характерных признаков из текста, а третья часть - полносвязная - используется для классификации текста на основе признаков, извлеченных сверточной частью. С тем, как работают слои B1 и Dense, вы уже знакомы. Давайте более подробно рассмотрим устройство сверточной части.

Для создания одномерных сверточных слоев используется класс Conv1D (сокращение от "конволюшен одномерный"). Первым параметром мы указываем количество ядер свертки - 100 ядер. Затем длину окна свертки - 5, и после этого функцию активации ReLU (полулинейная функция). После слоя свертки идет слой подвыборки с выбором максимального значения - название слоя MaxPooling1D. Одномерный слой размера "пула", из которого выбирается максимальное значение, равен 2. Это означает, что слой подключен к двум нейронам на предыдущем сверточном слое и выбирает из них одно максимальное значение, которое передается в следующий сверточный слой.

Также используется второй блок свертки-подвыборки (Conv1D и MaxPooling1D). Здесь мы уже указываем 200 - то есть для анализа данных с предыдущего слоя будет использоваться 200 различных вариантов ядер свертки, длина окна свертки также 5, и функция активации ReLU (полулинейная). После слоя свертки опять же идет слой подвыборки - выбор максимального значения из двух предыдущих нейронов. Итак, сверточная часть состоит у нас из двух блоков, каждая из которых включает слой свертки и слой подвыборки.



Данный код создает модель нейронной сети для анализа текстов.

Первый слой - Embedding - используется для создания векторного представления слов. Он принимает на вход max\_words - максимальное количество слов в словаре, 8 - размерность выходного вектора, и input\_length=200 - длину входной последовательности.

Далее идут два слоя свертки (Conv1D) с параметрами: 100 - количество ядер свертки, 5 - длина окна свертки, activation='relu' - функция активации ReLU. После каждого слоя свертки следует слой подвыборки (MaxPooling1D) с параметром pool\_size=2 - выбор максимального значения из каждых двух.

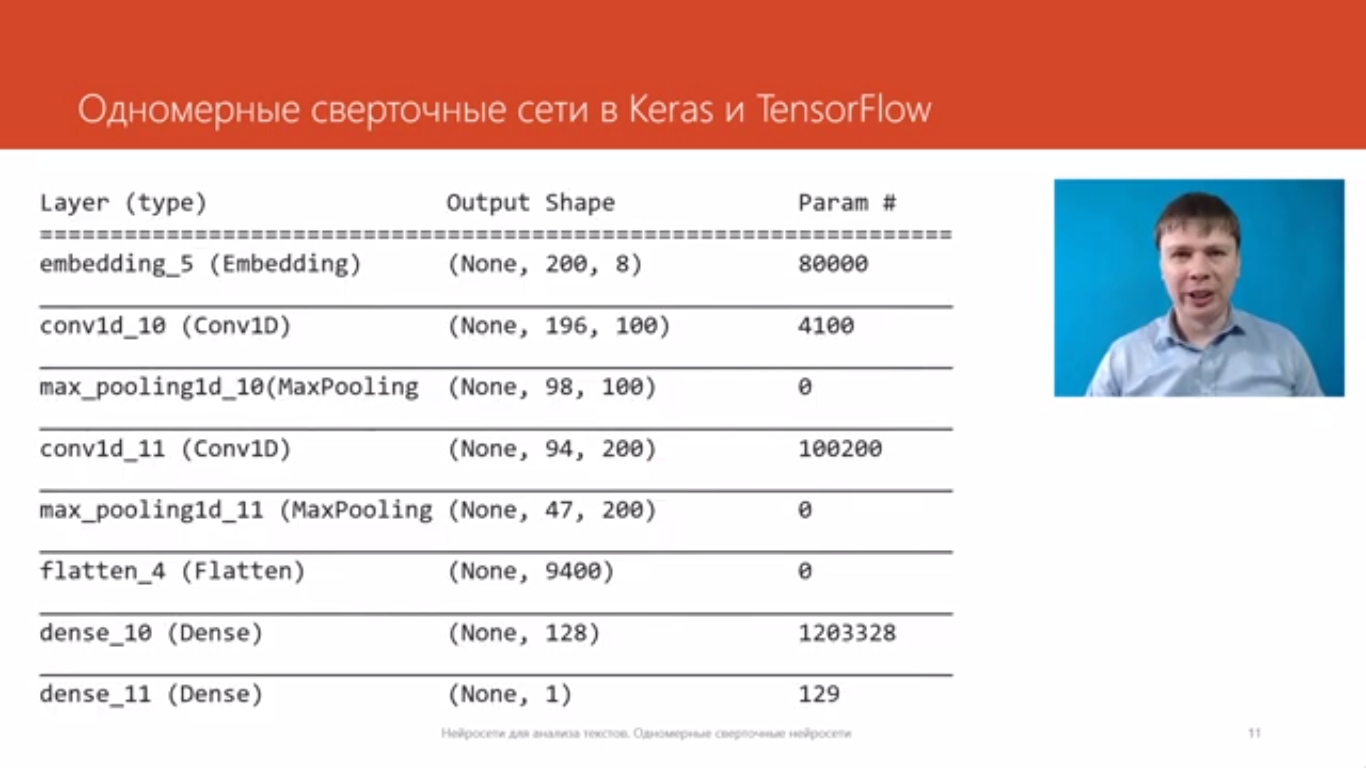
Затем идет слой Flatten, который преобразует многомерный тензор в одномерный. После этого идут два полносвязных слоя (Dense) с функцией активации ReLU и сигмоидной активацией соответственно.

Таким образом, слои свертки находятся на позициях 2 и 4, а слои подвыборки - на позициях 3 и 5.

Код для слоев свертки и подвыборки:

* Слой свертки 1: Conv1D(100, 5, activation='relu')
* Слой подвыборки 1: MaxPooling1D(pool\_size=2)
* Слой свертки 2: Conv1D(200, 5, activation='relu')
* Слой подвыборки 2: MaxPooling1D(pool\_size=2)

|  |
| --- |
| model= Sequential()  model.add(Embedding (max\_words, 8, input\_length=200))  model.add(Conv1D (100, 5, activation= 'relu'))  model.add(MaxPooling1D (pool\_size=2))  model.add(Conv1D (200, 5, activation= 'relu'))  model.add(MaxPooling1D (pool\_size=2))  model.add(Flatten())  model.add(Dense (128, activation= 'relu'))  model.add(Dense (1, activation='sigmoid')) |



Давайте рассмотрим вывод summary размеров данных, используемых в этой сверточной нейронной сети. На вход поступают последовательности длиной 200 элементов, которые представлены векторами размерностью 8 в слое Embedding. Затем используется 100 различных ядер свертки. После обработки 200 элементов данных на выходе получается 196 элементов. Это связано с тем, что длина окна свертки равна 5 элементам, и мы не можем обрабатывать два крайних элемента последовательности, потому что у них нет соседних элементов данных. Если мы хотим сохранить размерность данных в сверточном слое, то необходимо использовать слой MaxPooling. Он выбирает максимальные элементы из двух предыдущих, за счет чего размерность данных уменьшается еще в два раза, и остается 98 элементов. Затем мы повторяем похожие действия: операцию свертки с уменьшением количества элементов данных за счет двух крайних элементов с каждой стороны и еще один слой снижения размерности - MaxPooling. Итак, на выходе из сверточного слоя у нас 200 карт признаков, в каждой из которых по 47 чисел, и эти данные нужно передать в полносвязанный слой для классификации.

Однако полносвязанный слой не может работать с двумерными данными, поэтому мы преобразуем двумерные данные в одномерные с помощью слоя Flatten. Он преобразует двумерные данные в вектор, просто записывая строки матрицы друг за другом. Таким образом, из 200 карт признаков по 47 элементов у нас получается одномерный вектор, в котором 9400 элементов. Именно этот вектор подается на вход полносвязанным слоям. В первом полносвязанном слое находится 128 нейронов, а на выходном слое - один нейрон, который выдает значение либо 0, либо 1 в зависимости от того, положительный отзыв или отрицательный.

**Итоги**

Итак, мы рассмотрели, что такое одномерные сверточные нейронные сети. Они широко применяются для анализа последовательностей, не только текстов, но и любых других числовых сигналов. Сверточные сети используют операции свертки, в которых элементы данных в некотором окне свертки умножаются на элементы данных в ядре свертки. Особенностью сверточных нейронных сетей является то, что ядра свертки определяются автоматически в процессе обучения. Заранее их определять не нужно.

Архитектура сверточной нейронной сети для анализа текстов состоит из чередующихся блоков: слой свертки и слой подвыборки. После этого идут несколько полносвязанных слоев для решения задачи классификации. Преимущество сверточной нейронной сети заключается в том, что ее можно очень быстро реализовать, особенно на современных ускорителях вычислений, таких как GPU. Это связано с тем, что все свертки можно считать параллельно. В этом существенное отличие от рекуррентной нейронной сети, в которой необходимо учитывать данные от предыдущего этапа работы сети. Поэтому распараллелить рекуррентную нейронную сеть не просто и можно сделать только в ограниченных условиях.

Однако недостаток одномерной сверточной нейронной сети для обработки текстов заключается в том, что длина анализируемых данных, в которых нейронная сеть может что-то понять о тексте, ограничена окном свертки. Поэтому, если важные для нас слова встречаются на большом удалении друг от друга, которые больше, чем длина окна свертки, то одномерные сверточная нейронная сеть такие данные корректно проанализировать не сможет. В этом случае лучше применять рекуррентную нейронную сеть. Однако, с многими простыми задачами по классификации текста и по определению тональности, одномерные сверточные нейронные сети справляются очень хорошо. И учитывая, что они обучаются быстрее, чем рекуррентные, для простых задач анализа текстов лучше использовать одномерные сверточные нейронные сети.

***Одномерные сверточные нейронные сети***

* ***Используются для анализа последовательностей***
* ***Окно свертки, ядро свертки***
* ***Определение ядер свертки в процессе обучения***

***Архитектура сверточных сетей***

* + ***Слои свертки***
  + ***Слои подвыборки***
  + ***Полносвязные слои для классификации***

***Преимущества***

* ***Высокая скорость обучения сверки можно считать параллельно***

***Недостатки***

* ***Длина анализируемых данных ограничена окном свертки***

**Практика**

Лекция 6.1. Определение тональности текстов отзывов на сайте YELP одномерной сверточной нейросетью

<https://colab.research.google.com/drive/1LGf1emajl8nMOGZnMIE8SzsbK6EHWA6Q?usp=drive_link>

Лекция 6.2. Классификация новостей AG's News Topic Classification Dataset

<https://colab.research.google.com/drive/1ebdPKB5n_qYMBwqn9VKaOL66qmbaFJkD?usp=drive_link>

**Многозначная классификация текстов**

Задачи классификации

**Бинарная классификация (binary classification)**

Бинарная классификация (binary classification) - задача классификации, в которой объекты разбиваются на два класса. Каждый объект может принадлежать только к одному из двух классов.

Примером бинарной классификации может служить задача анализа тональности текстовых отзывов на кинофильмы в базах данных IMDB или YELP. В этом случае, каждый отзыв может быть положительным или отрицательным. Задача классификации заключается в определении тональности отзыва - положительной или отрицательной.

* Два класса объектов
* Объект может принадлежать только одному классу

Для решения задачи бинарной классификации можно использовать различные алгоритмы машинного обучения, такие как логистическая регрессия, метод опорных векторов (SVM), решающие деревья, нейронные сети и т.д. В процессе обучения модели используются размеченные данные - наборы примеров с заранее известными метками классов, на основе которых модель настраивает свои параметры для лучшей классификации новых объектов.

**Многоклассовая классификация (multiclass classification)**

Многоклассовая классификация (multiclass classification) - задача классификации, в которой объекты разбиваются на более чем два класса. Каждый объект может принадлежать только к одному из классов.

Примером многоклассовой классификации может служить задача классификации новостных статей на темы, такие как политика, спорт, экономика, наука и т.д., которая может быть решена с помощью набора алгоритмов машинного обучения.

* Несколько классов объектов
* Объект может принадлежать только одному классу

Например, набор данных AG News содержит новостные статьи из 4 разных категорий - мировые новости, национальные новости, спортивные новости и новости о развлечениях. Задача классификации заключается в определении категории, к которой относится каждая статья.

Для решения задачи многоклассовой классификации можно использовать различные алгоритмы машинного обучения, такие как метод опорных векторов (SVM), случайный лес, градиентный бустинг, нейронные сети и т.д. В процессе обучения модели используются размеченные данные - наборы примеров с заранее известными метками классов, на основе которых модель настраивает свои параметры для лучшей классификации новых объектов.

**Многозначная классификация (multilabel classification)**

Многозначная классификация (multilabel classification) предполагает, что каждый объект может принадлежать нескольким классам. Например, в задаче классификации токсичных комментариев (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge) каждый комментарий может быть отнесен к нескольким категориям, таким как "ненависть", "оскорбления", "угрозы" и т.д. Решение этой задачи может быть полезно для автоматического фильтрации токсичных комментариев в онлайн-сообществах.

* Несколько классов объектов
* Каждый объект может принадлежать нескольким классам

**Классы токсичности комментариев**



**Формат данных**



**Архитектура нейронной сети**

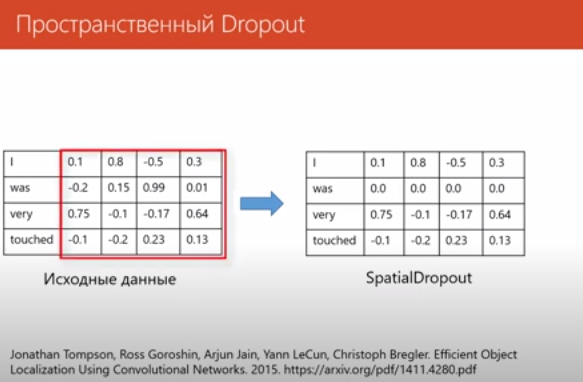
|  |
| --- |
| model = Sequential()  model.add(Embedding (10000, 128, input\_length=50))  model.add(Spatial Dropout1D (0.5))  model.add(LSTM(40, return\_sequences=True))  model.add(LSTM(40))  model.add(Dense (6, activation='sigmoid')) |

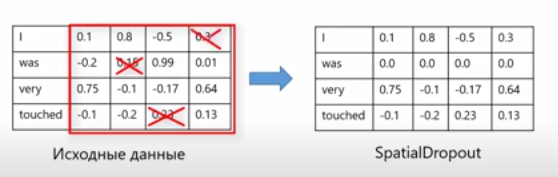
Это пример нейронной сети, реализованной с помощью библиотеки Keras на языке Python.

* Sequential() - создает модель, последовательно соединяя слои.
* Embedding(10000, 128, input\_length=50) - добавляет слой эмбеддинга, который преобразует каждое слово в вектор размерности 128. Модель принимает на вход последовательность из 50 слов.
* SpatialDropout1D(0.5) - добавляет слой дропаута, который случайным образом обнуляет 50% элементов вектора эмбеддинга.
* LSTM(40, return\_sequences=True) - добавляет слой LSTM с 40 скрытыми нейронами и возвращающий последовательность на выходе.
* LSTM(40) - добавляет еще один слой LSTM с 40 скрытыми нейронами.
* Dense(6, activation='sigmoid') - добавляет полносвязный слой с 6 выходами и функцией активации сигмоид. Каждый выход соответствует одному классу, к которому может принадлежать объект.

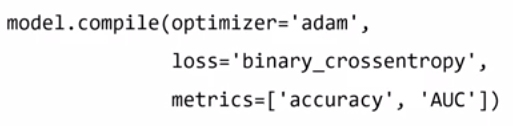
Эта модель может использоваться для решения задачи многозначной классификации текстовых данных, например, для классификации токсичных комментариев на несколько категорий.

**Пространственный Dropout[8]**

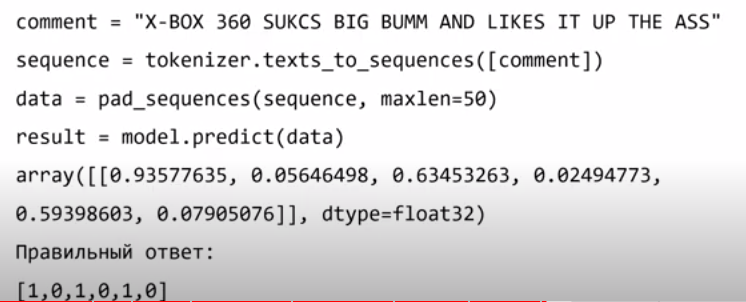




**Функция ошибка**



**Результат работы сеть**



**Функции активации и ошибки для задач классификации**



**Практика**

Лекция 6.3. Классификация токсичных комментариев

<https://colab.research.google.com/drive/1OF92x-2HiLd_bdPRgO1sTzNdAXuixRuN?usp=drive_link>

## **Задачи для самостоятельной работы**

1. Изучите лекционные материалы и статью [10] создайте общую презентацию.
2. Приведите 8-10 примеров NLP с применением сетей CNN, реализуйте их на Google Colab с использованием библиотек TensorFlow, Keras или PyTorch и сравните результаты.
3. Анализируем тональность YELP сверточной нейросетью
4. Классификация новостей [AG's News Topic Classification Dataset](https://github.com/mhjabreel/CharCnn_Keras/tree/master/data/ag_news_csv)
5. Классификация [токсичных комментариев](https://colab.research.google.com/corgiredirector?site=https%3A%2F%2Fwww.kaggle.com%2Fc%2Fjigsaw-toxic-comment-classification-challenge" \t "_blank)
6. Набор данных AG's News Topic Classification Dataset в формате csv - <https://github.com/mhjabreel/CharCnn_Keras/tree/master/data/ag_news_csv>
7. Toxic Comment Classification Challenge - <https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion/52644>

**Литература**

1. <https://habr.com/ru/companies/ods/articles/353060/>
2. <https://habr.com/ru/companies/vk/articles/417767/>
3. <https://practicum.yandex.ru/blog/svertochnye-neyronnye-seti/>
4. <https://habr.com/ru/articles/348000/>
5. <https://habr.com/ru/articles/348028/>
6. <https://se.moevm.info/lib/exe/fetch.php/courses:artificial_neural_networks:lecture_3.pdf>
7. <https://vc.ru/newtechaudit/517724-nastroyka-neyronnoy-seti-s-pomoshchyu-keras-python>
8. Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, Christoph Bregler. Efficient Object Localization Using Convolutional Networks, 2015, <https://arxiv.org/pdf/1411.4280.pdf>
9. <https://habr.com/ru/companies/itmai/articles/533746/>
10. <https://lena-voita.github.io/nlp_course/models/convolutional.html>