Московский государственный университет имени М.В. Ломоносова
Факультет Вычислительной математики и кибернетики
Кафедра исследования операций

Курсовая работа

Машинное обучение и виды архитектур нейронных сетей.

Автор:
гр.311
Харазян Давид Самвелович

Научный руководитель:
Поспелова Ирина Игоревна

Москва, 2020

1 Введение

У всех у нас давно уже на слуху такое новое словосочетание как «машинное обучение», статей и публикаций на сегодняшний день насчитывается множество, чтобы понять, что это такое давайте представим черный ящик, мы можем класть в этот ящик некоторые предметы, и после его работы получать из него другие предметы. И пусть этот ящик имеет очень много внешних рычажков, двигая которые, мы подкручиваем работу шестерёнок внутри и можем влиять на то, что будем получать из него. Допустим, мы кладем в него яблоки, а получаем апельсины, но мы хотим получать груши, а не апельсины. Вот взяли, подкрутили и теперь получаем груши, вопрос: какие рычажки, в какую сторону и как сильно мы подвинули? Всё было бы так просто, но вот тоже вопрос, а как объяснить ящику, что такое яблоко, а что такое груша? Чтобы мы могли быть достаточно уверены в том, что ящик при получении на вход дает груши, мы должны пропустить очень много яблок через наш черный ящик и при этом все они должны хотя бы напоминать груши.

Такой пример очень хорошо иллюстрирует принципы работы любого алгоритма машинного обучения: от самых простых — линейных, до сложных — нейронных сетей.
Ящик — это наша модель, способ движения рычажков - способ настройки этого алгоритма, а яблоки и груши - это наши данные, на основе которых мы понимаем куда, каким образом и как сильно двигать рычажки черного ящика.

Формальная постановка задачи машинного обучения:
Пусть имеется два множества $X \subset \mathbb{R}^n, n \in \mathbb{N}$.и $Y \subset \mathbb{R}^1$или$Y \subset \{0,1,2,\dots,k\}, k \in \mathbb{N}$.
И пусть существует некоторое отображение $F: X \to Y$.
Задача машинного обучения: по подмножеству пар вида $(x_i, y_i), i \in \mathbb{N}$ аппроксимировать отображение Fили в терминах статистики: посредством выборок $X_{train} = \{x_1, x_2, x_3, \dots, x_m\} \subset X, x_i \in \mathbb{R}^n, n \in \mathbb{N}$ и $Y_{train} = \{y_1, y_2, y_3, \dots, y_m\} \subset Y, y_i \subset \mathbb{R}^1 / y_i \subset \{0,1,\dots,k\}, k \in \mathbb{N}$,
ориентируясь в пространстве алгоритмов $a: X_{train} \to Y_{train}, a \in A$ (т.к. кол-во способов из конечной X_{train} получить Y_{train} бесконечно много) построить $a_{best} \in A$, который бы вел себя «лучше» всего на всей генеральной совокупности.

Однако дело в том, что достаточно неоднозначным является вопрос: какой алгоритм лучше, а какой хуже? Вернёмся к нашему ящику: какое расположение рычажков является лучшим из всех возможных способов повернуть рычажки? Сложно же сказать.
Чтобы все-таки мы могли на формальном математическом уровне понимать и оперировать с понятием «хороший алгоритм» научимся их сравнивать, т.е. корректно говорить, какой из 2-ух конкретных алгоритмов лучше, а какой хуже.

Пусть есть некоторая неотрицательная функция $L(a, x, y), a \in A, x \in X, y \in Y$. - *функция потерь, которая будет характеризовать величину ошибки алгоритма a на объекте x с меткой y.*

Также пусть есть некоторая функция $Q\left(a, X_{sample}, Y_{sample}\right) = M\left(\bar{L}(a, x_i, y_i)\right), i \in \mathbb{N}$,
где M — некоторое усреднение, обычно это математическое ожидание (ср. арифм.), но может быть и mod(модой), med(медианой), 50-ый перцентиль и т.п

Q показывает насколько «в среднем» алгоритм a ошибется на «sample» - подмножестве всей генеральной совокупности.

2 Постановка задачи

Целью данной работы является исследование общего принципа задач машинного обучения и в частности нейронных сетей, перед собой ставим следующие задачи.

1. Общий обзор задач машинного обучения и трёх архитектур нейронных сетей — полносвязных, свёрточных и рекурентных.

2. Провести экспериментальное сравнение различных архитектур нейронных сетей с одним из классических алгоритмов машинного обучения на корпусе англоязычных новостных повесток в рамках задачи классификации по темам.

3 Обзор задач машинного обучения

В данном разделе будет произведена формальная постановка задачи машинного обучения, приведено описание одного из классических алгоритмов (kNN), описание работы нейронных сетей на примере классического персептрона, свёрточной и рекурентной сетей.

3.1 Постановка задачи машинного обучения

[1] Пусть имеется два множества $X \subset \mathbb{R}^n, n \in \mathbb{N}$.и $Y \subset \mathbb{R}^1$или$Y \subset \{0,1,2,\dots,k\}, k \in \mathbb{N}$.
И пусть существует некоторое отображение $F: X \to Y$.
Задача машинного обучения: по подмножеству пар вида $(x_i, y_i), i \in \mathbb{N}$ аппроксимировать отображение F. Или в терминах статистики: посредством выборок $X_{train} = \{x_1, x_2, x_3, \dots, x_m\} \subset X, x_i \in \mathbb{R}^n, n \in \mathbb{N}$ и $Y_{train} = \{y_1, y_2, y_3, \dots, y_m\} \subset Y, y_i \subset \mathbb{R}^1 / y_i \subset \{0,1,\dots,k\}, k \in \mathbb{N}$, ориентируясь в пространстве алгоритмов $a: X_{train} \to Y_{train}, a \in A$ (т.к. кол-во способов из конечной X_{train} получить Y_{train} бесконечно много) построить $a_{best} \in A$, который бы вел себя «лучше» всего на всей генеральной совокупности. Но что значит «лучше»?

Пусть есть некоторая неотрицательная функция $L(a, x, y), a \in A, x \in X, y \in Y$. - *функция потерь, которая будет характеризовать величину ошибки алгоритма ана объекту xс меткой y.*

Также пусть есть некоторая функция $Q\big(a, X_{sample}, Y_{sample}\big) = M\big(\bar{L}(a, x_i, y_i)\big), i \in \mathbb{N}$,
где M — некоторое усреднение, обычно это математическое ожидание (ср. арифм.), но может быть и mod, med, центральный 50 перцентиль и т.п

Q показывает насколько «в среднем» алгоритм a ошибется на «sample» - подмножестве всей генеральной совокупности.

3.2 Обзор алгоритма kNN (k Nearest Neighbor) [2]

Основная гипотеза, на которую опирается эвристика алгоритма k-ближайших соседей: «Похожие объекты имеют похожие свойства».

Пусть мы имеем выборку
$X_{train} = \{x_1, x_2, x_3, \dots, x_m\} \subset X, x_i \in \mathbb{R}^n, n \in \mathbb{N} u$
$Y_{train} = \{y_1, y_2, y_3, \dots, y_m\} \subset Y, y_i \subset \mathbb{R}^1 / y_i \subset \{0,1, \dots, k\}, k \in \mathbb{N}$ *и мы хотим сделать предсказание* некоторым алгоритмом на выборке

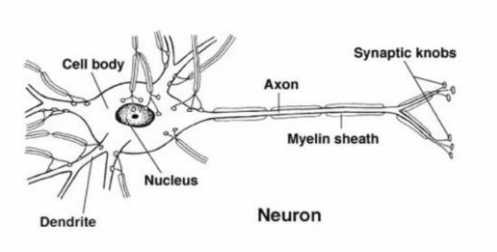
$$X_{test} = \{x_1, x_2, x_3, \dots, x_r\} \subset X, x_i \in \mathbb{R}^r, r \in \mathbb{N}$$
$$Y_{test} = \{y_1, y_2, y_3, \dots, y_r\} \subset Y, y_i \subset \mathbb{R}^1 / y_i \subset \{0,1, \dots, k\}, k \in \mathbb{N}$$

Посредством алгоритма KNN предсказание делается посредством следующей функции $a_{knn}(\vec{x}) = \frac{\|w_y(\vec{x})\|}{\|w(\vec{x})\|}, y_i \subset \mathbb{R}^1$, *где* $\|.\|$- *норма,* $\vec{x}$- *объект из тестовой выборки, значение целевой функции на котором мы стремимся аппроксимировать,* $w(\vec{x}) = \left(w(\vec{x}, \overrightarrow{x_1}), w(\vec{x}, \overrightarrow{x_2}), \dots, w(\vec{x}, \overrightarrow{x_n})\right)$, *где* $w(\vec{x}, \overrightarrow{x_i}) = K\left(q(\vec{x}, \overrightarrow{x_i})\right)$- *вес некоторого объекта по отношении к рассматриваемому, K(x) — монотонно не возрастающая функция,* $q(\vec{x}, \overrightarrow{x_i})$- *отображение ставящее паре векторов в соответствие неотрицательное число, часто предполагают, что удовлетворяет аксиомам метрики, но в общем случае не постулируется,* $w_y(\vec{x}) = \left(y_1 * w(\vec{x}, \overrightarrow{x_1}), y_2 * w(\vec{x}, \overrightarrow{x_2}), \dots, y_n * w(\vec{x}, \overrightarrow{x_n})\right)$, *где* y_i- *значение целевой функции при* $\vec{x} = \overrightarrow{x_i}$. *В случае если* $y_i \subset \{0,1, \dots, k\}, k \in \mathbb{N}$, *то* $a(\vec{x}) = argmax_c(w(\vec{x}, \overrightarrow{x_i}) I[y_i = c]), I[x] = \begin{Bmatrix} 1, x = true \\ 0, x = false \end{Bmatrix}$ –
индикатор
Самыми популярные примеры функции K(x) и q(a,b):
$K(x) = -x, q\left(\overrightarrow{x_1}, \overrightarrow{x_2}\right) = arccos(\overrightarrow{x_1}, \overrightarrow{x_2})$– *косинусное расстояние (лучше всего при n >> 3)*
$K(x) = -x, q\left(\overrightarrow{x_1}, \overrightarrow{x_2}\right) = \|\overrightarrow{x_1} - \overrightarrow{x_2}\|_2$– *Евклидова метрика (l2 норма)*
$K(x) = -x, q\left(\overrightarrow{x_1}, \overrightarrow{x_2}\right) = \|\overrightarrow{x_1} - \overrightarrow{x_2}\|_1$– *Манхэтонское расстояние (l1 норма)*
Менее популярные примеры, но которые тоже имею место:

$$K(x) = \frac{1}{x + \beta}, \beta > 0. \text{иначе} \frac{1}{0}, x = 0$$
$$K(x) = e^{-x}$$
$$K(x) = \alpha^x$$

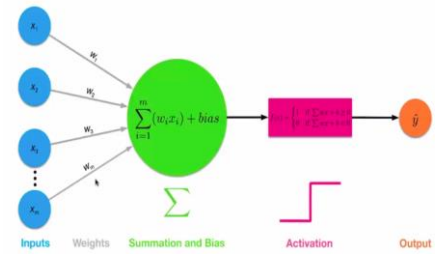
При различных метриках q(a,b):
$q(\overrightarrow{x_1}, \overrightarrow{x_2}) = I[\overrightarrow{x_1} = \overrightarrow{x_2}]$– *расстояние Левенштейна (по-координатный индикатор)*
$q(X, Y) = 1 - \frac{X \cap Y}{X \cup Y}$– *расстояние Джакарда(Жаккара).*
$q(\overrightarrow{x_1}, \overrightarrow{x_2}) = \sqrt{\left((\overrightarrow{x_1} - \overrightarrow{x_2})^T S^{-1} (\overrightarrow{x_1} - \overrightarrow{x_2})\right)}, S = S^T > 0$- *расстояние Махалонобиса*
Также используются, но более редко, метрики Минковского при p=0 — считающее расстояние, p=+inf — расстояние Чебышева, растояние Хаусдорфа и т. д.

3.3 Однослойная нейронная сеть — Персептрон



[3] Искусственная нейронная сеть представляет из себя некоторую модель, определяющуюся своей структурой (т.е. архитектурой) и настраиваемыми параметрами, способную аппроксимировать практически любого рода отображения с любой наперед заданной точностью.

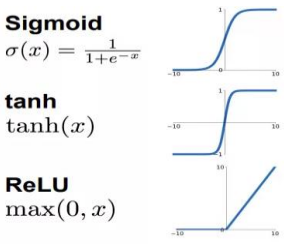
Начнем рассмотрение с самой простой и одной из самых первых реализаций нейронной сети - персептрона.

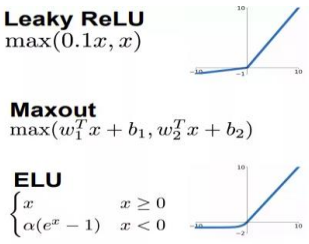


Основным компонентом персептрона, или как еще его называют — однослойной полносвязной нейронной сети, является математическая модель нейрона, который представляет из себя ориентированное дерево, листьям которого соответствует вектор $x = \{1, x_1, x_2, \dots, x_n\} \in X^n$, где множество X^n, обычно подмножество $\mathbb{R}^n$.

Ребрам, соединяющим листья дерева и вершину в ориентированном направлении, соответствует вектор $w = \{w_0, w_1, w_2, \dots, w_n\} \in \mathbb{R}^{(n+1)}$этот вектор называется вектором весом, именно эти параметры и оптимизируются при работе с нейронной сетью.

Инцидентная всем этим ребрам вершина представляет из себя сумматор $\sum_{i=0}^{n}(w_i * x_i)$, который ставит в соответствие вектору x и w число. Заметим, что размерность этих векторов совпадает как раз за счёт 1-ой координаты вектора x, которая всегда принимается равной 1, поэтому можно переписать формулу суммирования в виде $\sum_{i=1}^{n}(w_i * x_i) + w_0$,w_0- обычно еще обозначают как bias — что значит смещение, нужен для того, чтобы гиперплоскость $g(x) = \sum_{i=0}^{n}(w_i * x_i)$ не проходила через начало координат для любых w.

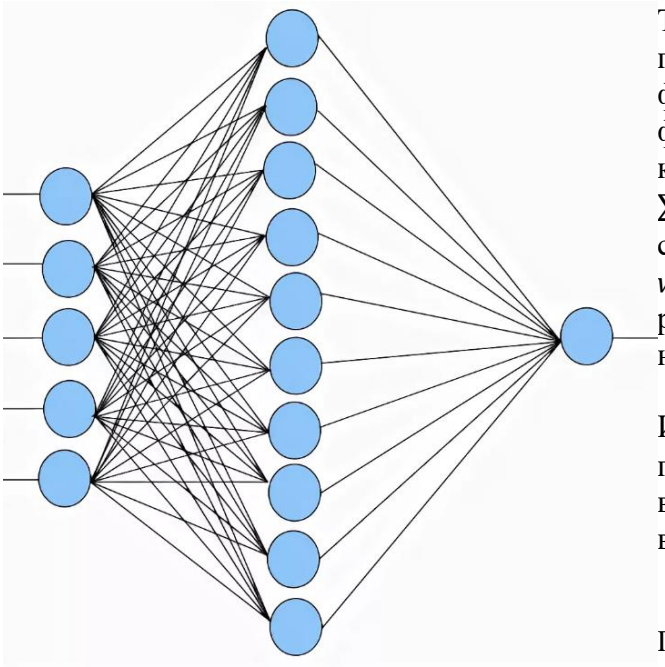




Сумматор в свою очередь посредством ориентированного ребра, которому ставится в соответствие число — выход из сумматора, инцидентен вершине, которой соответствуют так называемая функция активации, представляющую из себя сигмоидную функцию. Столь большое их разнообразие связанно со спецификой их производных.(в первых версиях персептрона использовалась пороговая функция Хевисайда $I(x) = \begin{cases} 0, x < x_0 \\ 1, x \geq x_0 \end{cases}$)

Функции активации на вход подается результат сумматора, а её результат ставится в соответствие ориентированному ребру инцидентного корню дерева, который именуют выходом нейрона, листья же принято называть входами.

Математически результат работы одного нейрона можно записать в следующем виде:
$f(\vec{x}) = \sigma(\sum_{i=1}^{n}(w_i * x_i) + w_0)$, где под $\sigma(x)$, понимается 1-значная функция 1-ой переменной.
Персептрон представляет же из себя композицию нейронов, соединенных параллельно, то есть листья у соответствующих деревьев одни и те же — т. е. вектор x, а вот веса для каждого соответственно разные. Выходы же разных нейронов подаются в дополнительный сумматор (или в несколько сумматоров, у каждого из которых независимые веса, таким образом персептрон будет иметь n входов и m выходов, но обычно принято считать, что у классического персептрона m = 1).



Таким образом, математически персептрон представляет из себя композицую сигмоидных функций и реализует n-мерную однозначную функцию (но в общем случае m-значаную), которую можно записать как:$f(\vec{x}) = \sum_{k=1}^{r}\left(w'_k * \sigma(\sum_{i=0}^{n} w_{ki}\, x_i)\right)$, где w_{ki}- вес, соответствующий i-му входу в k-ом нейроне, а $w' = \{w'_1, w'_2, \ldots, w'_r\} \in \mathbb{R}^r$- веса результирующего сумматора, а r — количество нейронов на слое.

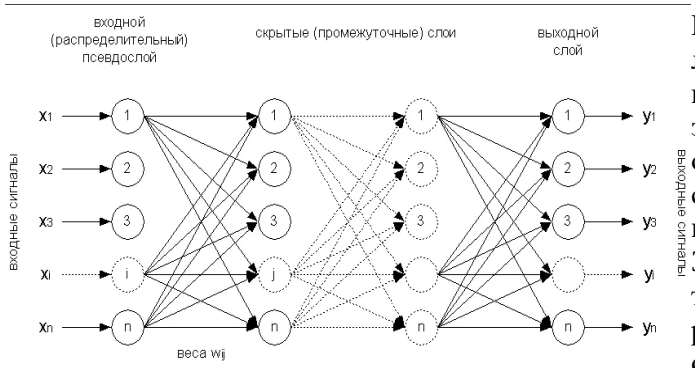
Или в векторной форме: $f(\vec{x}) = w'\vec{\sigma}\left(w_k^T \vec{x}\right)$, где под $\vec{\sigma}(x)$мы понимаем, вектор из значений выходов всех нейронов, то есть r-мерный вектор.

Практическая польза и один из самых важных результатов в теории аппроксимации с помощью искусственных нейронных сетей достигается из следующей теоремы.

[4] Теорема Цыбенко (Универсальная теорема аппроксимации):

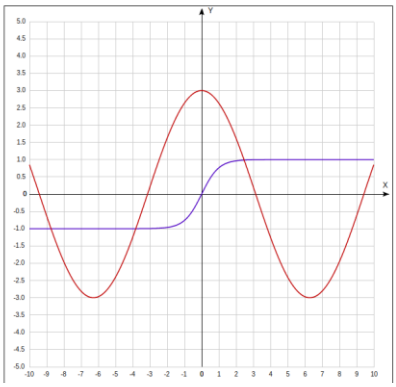
Для любой наперед заданной точности для представленной выше модели искусственной полносвязной нейронной сети существует такое кол-во нейронов на ее слое, при котором она способна аппроксимировать непрерывную любую 1-значную (m-значную) функцию от n переменных, с этой точностью.

К сожалению, строгая теория пока не так сильно развита и кроме представленной выше теоремы в основном руководствуются некоторыми эвристиками и дальше будем говорить преимущественно о них, при возможности опираясь на формальную теорию.



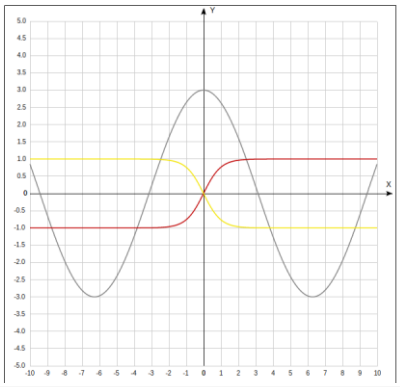
Как одно из усовершенствований персептрона в литературе можно встретить более популярное представление полносвязной нейронной сети — это многослойный перспептрон, где результаты одного слоя подаются не в результирующий сумматор, а представляются как входы для некоторого другого слоя, тот в свою очередь для 3-го и так несколько слоев, в конце концов только результат после k-го слоя подается в результирующий сумматор. Этот подход обоснован тем, что на практик, при такой композиции, достигается более гибкая свобода в структуре, что уменьшает необходимое количество параметров, что ускоряет время их подбора и количество памяти для её хранения.

Интуитивно понять, почему же эта теорема работает и каким же образом достигается это приближение, проиллюстрируем соответствующую композицию нужных параметров, подобранных опытным путем, и классической функцией сигмоиды $\sigma(x) = \frac{1}{1+e^x}$ для аппроксимации функции $f(x) = 3cos\left(\frac{x}{2}\right)$ в окрестности нуля.



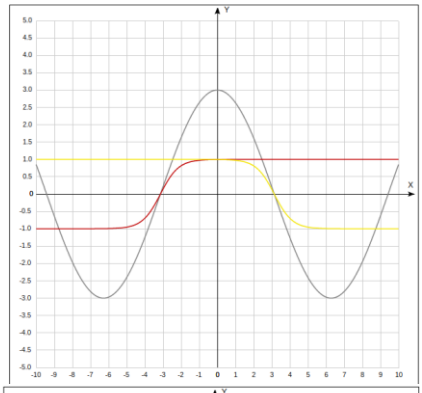
$y(x) = 3cos(\frac{x}{2})$ Показать таблицу точек

$y(x) = tanh(x + 3.142) + tanh(-x + 3.142)$ Показать таблицу точек



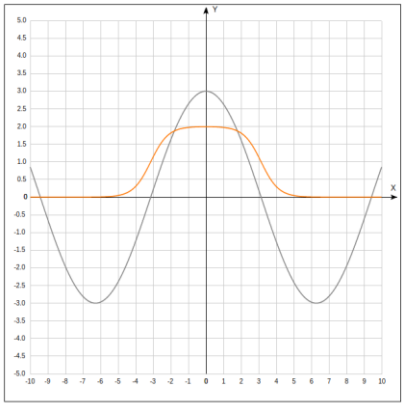
$y(x) = 3cos(\frac{x}{2})$ Показать таблицу точек

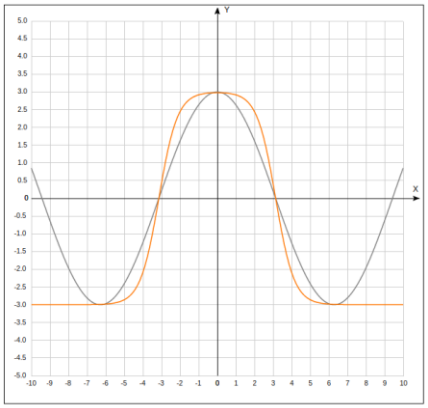
$y(x) = 3(tanh(x + 3.142) + tanh(-x + 3.142)) - 3$ Показать таблицу точек

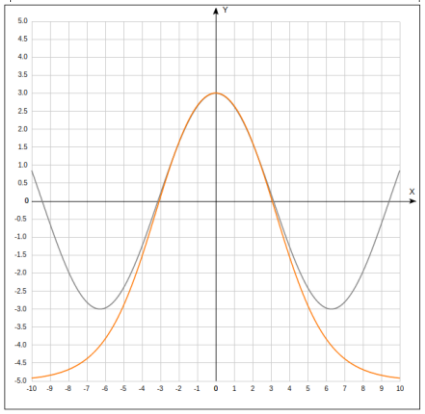


$y(x) = 3cos(\frac{x}{2})$ Показать таблицу точек

$y(x) = 5(tanh(0.35(x + 3.142)) + tanh(-0.35(x - 3.142))) - 5$ Показать



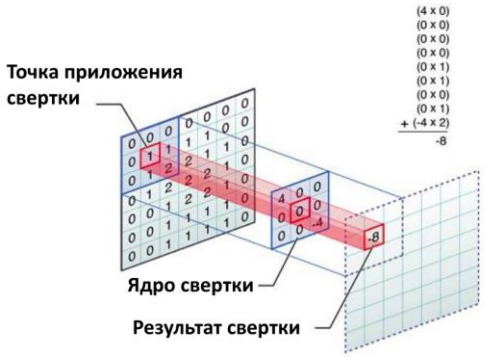




3.4 Свёрточная нейронная сеть

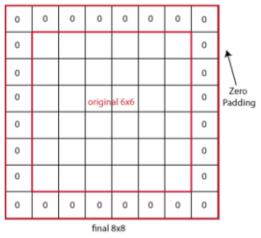
Заметим, что в архитектуре персептрона нет учёта взаимного расположения элементов в входном векторе, кроме как последовательного. Поэтому он хорошо себя проявляет в задачах по предсказанию вещественной переменной, аппроксимации некоторых маломерных функций, предсказывания временных рядов. Но в то же время он не очень хорошо себя ведет в задачах с текстами и особенно при работе с изображениями, так как приходится матрицу-изображение «вытягивать» в вектор, а при этом теряется информация о взаимном расположении объектов на фотографиях, в связи с этим хотелось бы чтобы в архитектуре присутствовала инвариантная операция, посредством которой нейронная сеть могла мы извлекать соответствующие патерны — некоторые признаки, учитывая которые, нейросеть способна делать качественное предсказание. С этой целью была придумана специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году — архитектура свёрточной нейронной сети.
Главным отличием свёрточной нейронной сети является наличие 2 новых слоев — слой свёртки и слой пулинга.

3.4.1 Свёрточный слой



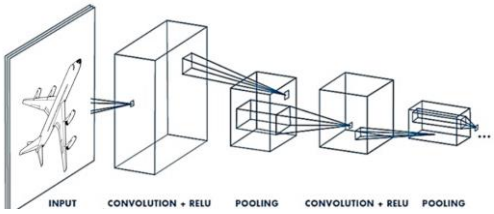
[5] На вход свёрточному слою подается тензор (3-ёх мерная матрица), будем в дальнейшем интерпретировать его как несколько каналов, на каждом из которых находятся матрицы размеров NxN. Затем каждому каналу ставится в соответствие свой фильтр — матрица порядков nxn (N >> n). После чего происходит свёртка каждого канала посредством фильтра — результатом получаем матрицу размеров N-n+1 x N-n+1. После этого процесса получаем матрицы меньших размеров для каждого из каналов, затем все они поэлементно складываются и образуют одну матрицу. Множество использованных фильтров называется ядром свёртки, таким образом посредством которого получаем свёртку входного тензора в одну матрицу. Часто используют не одно ядро, а несколько различных. По итогу применяя K различных ядер для входного тензора, состоящего из M каналов, получаем, что мы входные M каналов переводим в K каналов (обычно K>M), однако в которых матрицы имеют меньший размер.

Часто для того, чтобы не уменьшать размер матрицы использую операцию падинга (padding). Результатом применения падинга равного k к матрице размеров NxN будет матрица размеров N+2k x N+2k, где к исходной матрице по краям добавляются 0 (есть и другие варианты, такие как отзеркаливание или заполнение -1, но 0 — самые популярные).



3.4.2 Пулинг слой

[5]На вход слою пулинга (pooling) подается так же тензор — несколько каналов с матрицами. После чего по каждой матрице проходят окном размеров nxn (размер пулинга) и в каждом окне считают некоторую статистику. И посредством ее получают новую матрицу меньших размеров как на свёрточном слое - N-n+1 x N-n+1. Чаще всего в качестве статистики берут максимальное/минимальное значение в окне (maxpool / minpool), среднее чисел попавших в окно (avgpool).

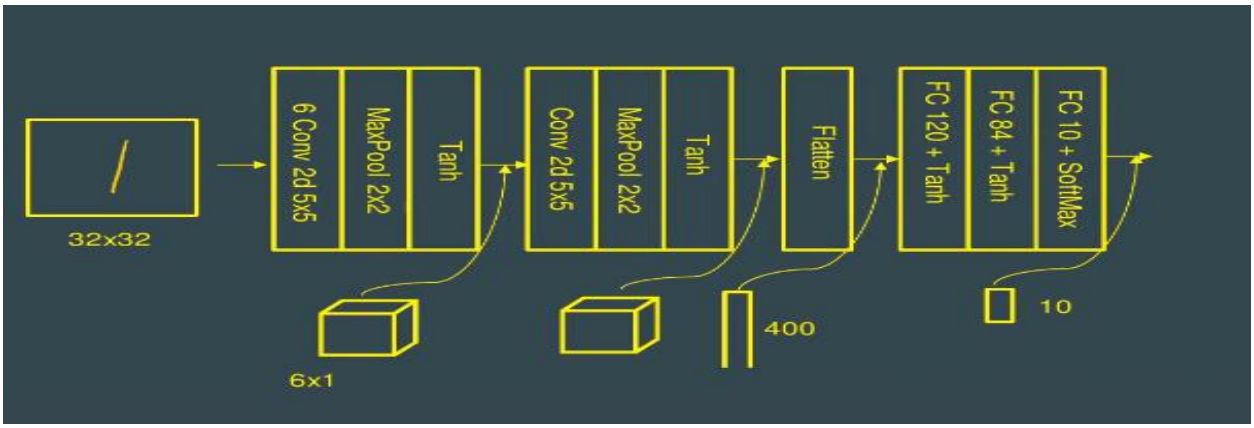


Описанный выше подход является формальным алгоритмом слоя пулинга, однако он подразумевает движение окна с шагом 1, на практике чаще всего этот шаг делают равным n-1, как по вертикале, так и по горизонтали. Таким образом, каждый элемент матрицы попадает в окно пулинга ровно один раз и результирующий размер матрицы становится равным N//n x N//n (// - целочисленное деление). Такой примем изменения шага скользящего окна называется страйдом (stride) и задается парой чисел (n1,n2) — шаг по веритикале и горизонтали соответственно. Также этот прием применим и к свёрточному слою, но он менее популярен.

3.4.3 Вывод

[5]После свёрточного слоя и слоя пуллинга следует слой активации — к каждому элементу матрицы добавляется некоторая константа bias, для каждого своя. И применяется некоторая сигмоидная функция из рассмотренных ранее. Таким образом после этого слоя тензор не изменят размеров.

После последовательных таких слоев изначальный тензор, состоящий как правило из 3 каналов, на каждом из которых матрицы порядков 100-5000, преобразуются в достаточно глубокий тензор — количество каналов может превышать 200-1000, на каждом из которых матрицы размеров 5-10. Как правило над таким тензором применяют операцию flatten — его вытягивают в вектор и подают полносвязной не глубокой сети, количество слоев в которой варьируется от 1 до 3, в зависимости от архитектуры. Ниже приведет пример одной из первых архитектур свёрточной нейронной сети — LeNet(1998).



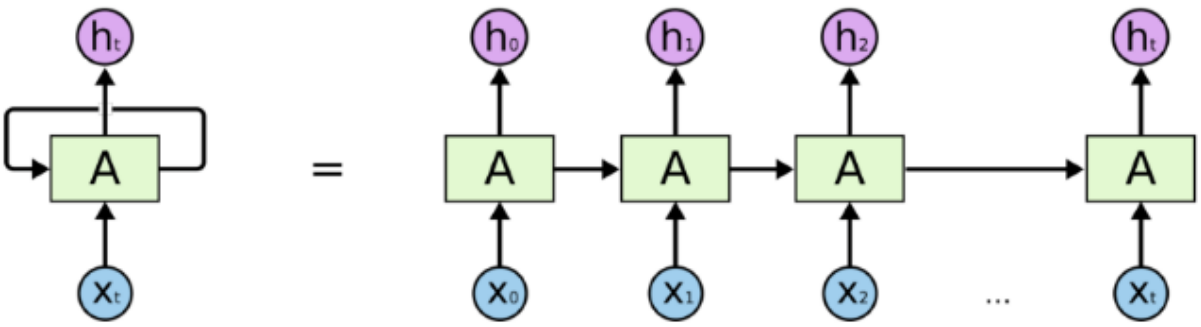
3.5 Рекурентная нейронная сеть

Вид нейронных сетей, где связи между элементами образуют направленную последовательность. Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки. В отличие от многослойных персептронов, рекуррентные сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины. Поэтому сети RNN применимы в таких задачах, где нечто целостное разбито на части, например: распознавание рукописного текста или распознавание речи. Было предложено много различных архитектурных решений для рекуррентных сетей от простых до сложных. В последнее время наибольшее распространение получили сеть с долговременной и кратковременной памятью (LSTM) и управляемый рекуррентный блок (GRU).

Основным элементом рекурентной сети является так называемый «развернутый во времени» рекурентный блок, который работает по следующему принципу.

Чтобы подать последовательно некоторый вектор данных через элемент нейронной сети, требуется продублировать соответствующий элемент столько раз, сколько элементов в этом векторе, затем подать в каждый элемент соответствующий объект в векторе, после чего результат 1-го элемента, подается во 2-ой, как еще один аргумент, после чего 2-ой элемент «развернутого» рекурентного блока выдает результат и направляет его в 3-ий и т.д. пока так называемый «вектор скрытого состояния» не дойдет до последнего блока. В общем случае, помимо вектора срытого состояния каждый элемент рекурентного блока независимо выдает результат далее в сеть, так называемый «вектор выходного слоя».

В общем случае эвристика скрытого состояния состоит в том, что предполагается, что нейронная сеть в этом векторе хранит ту информацию, которая содержалась во всех «просмотренных» объектах вектора, таким образом сеть аккумулирует информацию к тому моменту, когда следует сделать некоторый прогноз, в котором должна учитываться предыдущая информация.



4 Экспериментальное исследование. [6]

В качестве модельного примера для сравнения качества результатов работы всех рассмотренных выше алгоритмов — kNN, NN, CNN, RNN, возьмем задачу представляющую из себя классификацию новостных повесток дня на 4 тематики - бизнес, спорт, мир, технологии.

Где в качестве объектов для kNN будет выступать множество - $X = \{x_1, x_2, x_3, \dots, x_n\}, x_i \subset \mathbb{R}^m$, где x_i- строка в разряженной tf-idf матрице (матрица, где столбцам соответствуют слова, которые встречаются во всех текстах, а строкам соответствующие тексты, в пересечениях находится произведение tf и idf, где tf — term frequency, вероятность встретить данное слово в рамках этого текста, а idf — inverse document frequency, единица делить на вероятность, что в данном тексте встретится это слово хотя бы 1 раз [8]).

В качестве объектов для нейронных сетей будет выступать множество - $X = \{x_1, x_2, x_3, \dots, x_n\}, x_i \subset \mathbb{R}^{m_1 x m_2}$, где x_i- матричное представление одной новостной повестки, строками которой являются эмбединг вектора слов (плотные вектора размерности 50-300, в данной реализации представляющие из себя случайные вектора, которые в рамках матрицы склеиваются, вытягиваются в 1 длинный вектор, подаются нейронной сети и обучаются вместе с ней, таким образом после обучения сети получаем векторное представление слов [9]).

В качестве предсказываемых значений будет выступать $Y = \{y_1, y_2, y_3, \dots, y_n\}, y_i \subset \{0,1,2,3\}$, где y_i- соответственно идентификатор темы, которой принадлежит новость. В качестве соответственно аппроксимируемой функции будет выступать отображение $f: X \to Y$- функция, классификатор. Метрикой качества будет выступать доля правильных предсказаний: $Q(a, X_{test}, Y_{test}) = \frac{1}{k}\sum_{i=0}^{k} I\,[f(x_i) = a(x_i)]$, где $f(x_i), a(x_i)$- соответственно реальное значение функции на объекте и предсказанное.

В следствии эксперимента получены следующие результаты. KNN — доля верных предсказаний на тестовом наборе данных 89.4%. NN — 89.6%, CNN — 89.9%, RNN — 89.7%.

5 Заключение

В данной работе были рассмотрены основы машинного обучения, классический алгоритм k-ближайших соседей, основные архитектуры нейронных сетей — полносвязный персептрон, свёрточная нейронная сеть и рекурентная нейронная сеть.

Был проведен эксперимент по сравнению прогнозов работы этих 4 алгоритмов, по результатом которых был получен следующий результат — kNN, как алгоритм который хорошо показывает себя для работы с текстами, всё-таки, хоть и не значительно, но уступает в качестве нейросетевым алгоритмам, однако достаточно сильно уступает с точки зрения занимаемой памяти и скорости эксплуатации, т.к. нейронным сетям не нужно запоминать всю выборку.

При этом нейронные сети всех 3 архитектур показали примерно одинаковое качество, однако стоит учесть, что сравнение их было проведено без подбора нужного количества слоев и определенных функций активаций, т.е. так называемых гиперпараметров, а использовались исключительно базовые виды архитектур. Тем не менее, как и ожидалось, полносвязная сеть показала себя незначительно хуже, чем свёрточная и рекурентная сети, но лучше чем kNN алгоритм.

В дальнейшем планируется углубиться в нейросетевые подходы для обработки текстовой информации и естественного языка, рассмотреть и исследовать известные форматы векторных представлений текстов, основанных на эмбедингах, и провести эксперименты по обнаружению новых или в целях того, чтобы удостовериться, что известные подходы являются достаточно исчерпывающими.

Список литературы

[1] Ryszard S. Michalski, Jaime G. Carbonell, Tom M. Mitchell (1983), Machine Learning: An Artificial Intelligence Approach, Tioga Publishing Company, ISBN 0-935382-05-4

[2] Daniel T. Larose, Discovering Knowledge in Data: An Introduction to Data Mining (https://web.archive.org/web/20140531051709/http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471666572.html)

[3] Вапник В.Н., Червоненкис А.Я. Теория распознавания образов (статистические проблемы обучения)

[4] Cybenko, G. V. Approximation by Superpositions of a Sigmoidal function // Mathematics of Control Signals and Systems. — 1989. — Т. 2, № 4. — С. 303—314.

[5] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998,

[6]https://colab.research.google.com/drive/1cib9V20Pa28RpDEYDTdS35GopzXNk-f5?usp=sharing

[7] https://github.com/mhjabreel/CharCnn_Keras/tree/master/data/ag_news_csv

[8] https://ru.wikipedia.org/wiki/TF-IDF

[9] https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526