УДК 004.032.26

Анализ алгоритмов построения нейронных сетей для задач классификации

Пятецкий Гордей Олегович,
бакалавр,
Коваленко Иван Романович,
бакалавр
Владивостокский государственный университет экономики и сервиса
Россия. Владивосток
E-mail: Gordey_Pyatetskiy@vvsu.ru; Тел. +79025243070
E-mail: Kovalenko.IR@vvsu.ru; Тел. +79940006023
Ул. Гоголя, 41, г. Владивосток, Приморский край, Россия, 610014

В данной статье проводится анализ алгоритма построения нейронной сети типа «однослойный перцептрон». Для анализа архитектуры была выбрана задача классификации Ирисов по трем классам (setosa, versicolor, virginica). Так как алгоритм, написанный на языке программирования Python, имеет программный вид, то все необходимые параметры в качестве входных сигналов будут вводится в программу вручную.

Ключевые слова: *архитектура нейронной сети, перцептрон, функция активации, передаточная функция, язык программирования Python.*

Analysis of algorithms for building neural networks for classification problems

This article describes the analysis of the algorithm for constructing of a neural network of the "single-layer perceptron" type. For the analysis of architecture, the task of classifying irises into three classes (setosa, versicolor, virginica) was chosen. Since the algorithm, written in the Python programming language, has a program form, all the necessary parameters as input signals will be entered into the program manually.

Keywords*: neural network architecture, perceptron, activation function, transfer function, Python programming language.*

Объект исследования: искусственный интеллект.

Предмет исследования: нейронная сеть (НС).

Цель: анализ модели построения нейронной сети, приспособленной для решения задач классификации.

Задачи: создать нейронную сеть и проанализировать ее архитектуру.

Исполнение алгоритма классификации: программное (Язык программирования Python с использованием дополнительных заголовочных файлов).

Проблема: подобрать оптимальную архитектуру для НС с оптимальными обучаемыми параметрами.

Актуальность: применение нейронных сетей в системах, использующих компьютерное зрение.

1. Анализ архитектуры «однослойный перцептрон»

В нашем быстроразвивающемся мире нас практически всюду окружает искусственный интеллект, выраженный нейронными сетями, которые, в свою очередь, встроены в различные программы и гаджеты. Поэтому, на данный момент, направления, связанные с нейронными сетями, пользуются огромной популярностью, как среди обыкновенных пользователей, так и среди больших растущих компаний. У людей возникает все больше и больше потребностей в обработке больших данных и в поиске алгоритмов, подходящих для их быстрой и точной обработки. Нейронные сети выполняют достаточно большое количество разнообразных задач, в число которых входят: задачи классификации, кластеризации, прогнозирования, аппроксимации, анализа данных, оптимизации и др. Остановимся на задаче классификации. Она наиболее часто используется в учебных целях и является универсальной для построения различных нейронных сетей с различной архитектурой. Построение их архитектуры базируется на определении начального числа нейронов во входном и выходном слоях [1, 2].

Для определения числа нейронов, находящихся во входном слое, возьмем количество входных параметров: длина и ширина чашелистика, длина и ширина лепестка цветка ириса. Аналогичным образом определяем количество нейронов в выходном слое, оно совпадает с количеством классов: setosa, versicolor, virginica. Получаем, что входных нейронов четыре, а выходных три. Точное количество нейронов в скрытом слое определить нельзя, так как это «гиперпараметр» и его можно будет корректировать исходя из результатов обучения сети. Предположим, что количество нейронов (узлов) в скрытом слое равно трем. Так будет легче проводить расчеты и в случае неудовлетворительного результата можем их изменить.

Условимся, что расчеты будем проводить на уровне слоев нейронной сети, следовательно, в основных расчетах будут фигурировать матрицы и векторы. Все входные параметры запишем в виде матрицы-строки (вектора) $X = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix}$. На один нейрон поступает несколько сигналов умноженных на веса, далее они суммируются с прибавлением смещения и полученное значение заносится в функцию активации, откуда исходит сигнал на следующий слой нейронов. Для последующих расчетов необходимо составить матрицу весов. Размерность матрицы будет зависеть от количества входных нейронов – 4 и нейронов в скрытом слое – 3. Обозначим ее следующим образом $W_1 = \begin{pmatrix} w_{11} & \cdots & w_{13} \\ \vdots & \ddots & \vdots \\ w_{41} & \cdots & w_{43} \end{pmatrix}$. Для большей точности результата выберем смещение или «байес». Обозначим его за $B_1 = \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix}$. Его размерность будет зависеть от количества нейронов в скрытом слое. Запишем формулу в общем виде для нахождения результирующего вектора $T_1 = XW_1 + B_1 = \begin{pmatrix} t_1 & t_2 & t_3 \end{pmatrix}$. В качестве передаточной функции будет выступать полулинейная функция с насыщением под названием Relu. Выглядит она так:

$$F(x) = \{x, x > 0 \quad 0,\ x \leq 0$$

Важно отметить, что функция активации может быть из класса сигмоидальных функций (логистическая, гиперболический тангенс и т.д.) (Рис. 1).

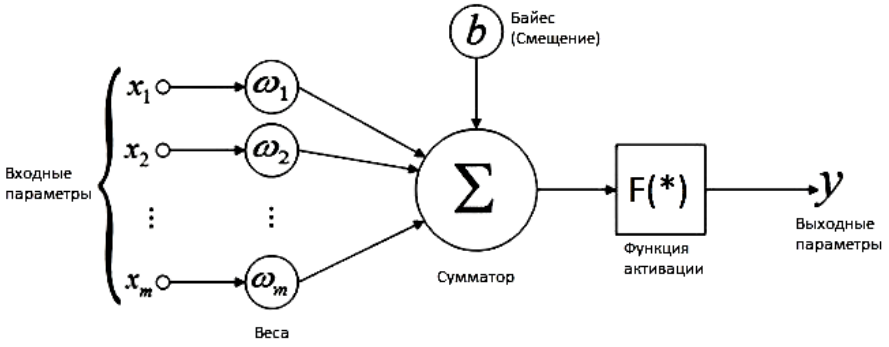

Рис.1. Схема устройства перцептрона

После того, как сигнал пройдет через скрытый слой нейронов получается вектор выходных сигналов. Обозначим его за $H = F\left(T_1\right) = \begin{pmatrix} h_1 & h_2 & h_3 \end{pmatrix}$. Далее сигнал переходит к выходному слою нейронов, где аналогично умножается на веса и прибавляется байес, но не передается в функцию активации, потому что в данной сети один скрытый слой нейронов. Запишем это так $T_2 = HW_2 + B_2 = \begin{pmatrix} t_1 & t_2 & t_3 \end{pmatrix}$, где $W_2 = \begin{pmatrix} w_{11} & \cdots & w_{13} \\ \vdots & \ddots & \vdots \\ w_{31} & \cdots & w_{33} \end{pmatrix}$ – это вторая матрица весов и $B_2 = \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix}$. На выходе из нейронной сети необходимо получить

распределение вероятностей по принадлежности растения к определенному классу. Для этого воспользуемся функцией Softmax, имеющая вид:

$$z = \sigma(T)_i = \frac{e^{t_i}}{\sum_j^n e^{t_j}}$$

Данная функция преобразует вектор T размерностью n в вектор z той же размерности, в котором компоненты будут распределением вероятностей. Ключевым шагом является выбор максимального элемента из матрицы-строки z. Так как он и будет являться верным результатом для тех или иных параметров.

2. Анализ алгоритма обучения для архитектуры «однослойный перцептрон»

Для того, чтобы получить желаемый правильный результат, необходимо обучить сеть [3]. Здесь необходимо пояснить, что в сети присутствуют как обучаемые параметры, так и необучаемые. К обучаемым параметрам в данной архитектуре относят различные матрицы весов и байесы, а остальные параметры необучаемы и носят название «гиперпараметров», которые можно менять вручную. Для обучения сети был выбран метод обратного распространения ошибки (backpropagation) [4, 5]. Это метод обучения с учителем, который предполагает введение в сеть обучающей выборки, состоящей из пар (параметр – известный ответ), а на выходе сравнивать ответ из выборки с полученным ответом и вычислять ошибку. А через ошибку, в свою очередь, вычисляется градиент [6] ошибки по всем обучаемым параметрам. Метод обучения с учителем предполагает наличие обучающей выборки в качестве инструмента для обучения. В качестве обучающей выборки будем использовать готовую базу данных (dataset) из python библиотеки sklearn под названием «Ирисы Фишера». Обучающая выборка (batch) обладает своим гиперпарметром, а именно размером. Изначально размер выборки может быть произвольным, поэтому в данном случае будем использовать выборку, состоящую из 50 пар.

В первый запуск сети все объекты, содержащие обучаемые параметры, были проинициализированы случайными значениями. Как было указано выше, если результат работы сети неудовлетворительный, то можно гиперпараметры изменить вручную. Предположим, что через сеть уже прошла обучающая выборка и на выходе из сети был получен некорректный ответ. Чтобы корректно вычислить ошибку, воспользуемся функцией потерь. В качестве функции потерь будет функция перекрестной энтропии. Ее запишем таким образом

$$CE(z,\ y) = -\sum_i y_i \ln ln\, z_i\ ,$$

где y_i – это элемент выборки, содержащий правильный ответ, а z_i – это полученный результат на основе введенных параметров.

Теперь, зная ошибку, обозначим ее за $E = CE(z, y)$, можно вычислить градиентный спуск по всем обучаемым параметрам. Тогда граф обратного распространения будет иметь вид:

$$E \rightarrow \frac{\partial E}{\partial T_2} \rightarrow \frac{\partial E}{\partial B_2} \rightarrow \frac{\partial E}{\partial W_2} \rightarrow \frac{\partial E}{\partial H} \rightarrow \frac{\partial E}{\partial T_1} \rightarrow \frac{\partial E}{\partial B_1} \rightarrow \frac{\partial E}{\partial W_1} \rightarrow \frac{\partial E}{\partial X},$$

где $\frac{\partial E}{\partial B_2}, \frac{\partial E}{\partial W_2}$ и $\frac{\partial E}{\partial B_1}, \frac{\partial E}{\partial W_1}$ – это обучаемые параметры, которые предстоит найти. При вычислении и выведении формул будем пользоваться цепным правилом или правилом дифференцирования сложной функции. Первым параметрами для нахождения будут $\frac{\partial E}{\partial B_1}, \frac{\partial E}{\partial W_1}$. Так как мы идем не от конца сети, то предположим, что $\frac{\partial E}{\partial H}$ известно. Найдем $\frac{\partial E}{\partial T_1}$.

$$\frac{\partial E}{\partial T_1} = \frac{\partial E}{\partial H} * \frac{\partial H}{\partial T_1} = \frac{\partial E}{\partial H} * F'\left(T_1\right), \text{ где } F'(x) = \{1, x > 0 \quad 0, x \leq 0 .$$

Далее можем вычислить матрицу весов $\frac{\partial E}{\partial W_1}$. В общем виде формула выглядит так:

$$\frac{\partial E}{\partial W_1} = \frac{\partial E}{\partial T_1} * X^T$$

Стоит пояснить, что при нахождении матрицы весов вычисляется градиент ошибки от каждого ее элемента и здесь есть закономерность, из которой и была выведена формула выше. Проделываем аналогичную роботу с нахождением матрицы-строки, состоящей из байесов. Получаем следующее выражение:

$$\frac{\partial E}{\partial b_1} = \frac{\partial E}{\partial t_1} * \frac{\partial t_1}{\partial b_1} = \frac{\partial E}{\partial t_1} * b_1^{'} = \frac{\partial E}{\partial t_1}$$

Здесь так же не трудно отследить закономерность, поэтому общая формула будет иметь вид:

$$\frac{\partial E}{\partial B_1} = \frac{\partial E}{\partial T_1}$$

Для полноты вычислений выведем градиент ошибки по входу $\frac{\partial E}{\partial X}$. Для этого воспользуемся правилом дифференцирования сложной функции нескольких переменных. Градиент будем находить поэлементно.

$$\frac{\partial E}{\partial x_1} = \frac{\partial E}{\partial t_1} * \frac{\partial t_1}{\partial x_1} + \frac{\partial E}{\partial t_2} * \frac{\partial t_2}{\partial x_1} + \frac{\partial E}{\partial t_3} * \frac{\partial t_3}{\partial x_1}$$

Здесь важно заметить, что входной нейрон, который получает на вход параметр x_1 весами w_{11}, w_{12}, w_{13} с нейронами скрытого слоя. Поэтому здесь тоже можно вывести формулу в общем виде из закономерностей.

$$\frac{\partial E}{\partial X} = \frac{\partial E}{\partial T_1} * W_1^T$$

Формулы для половины сети выведены, теперь обратим внимание на другую половину. Вернемся к ошибке, которую вычисляли через $CE(z, y)$. С помощью этой функции выведем формулу для $\frac{\partial E}{\partial T_2}$. Для этого подставим функцию Softmax в Cross Entropy. Получим $E = -\sum_i y_i \ln ln \frac{e^{t_i}}{\sum_j^n e^{t_j}}$. Упростим выражение и возьмем от него производную. Получим $\frac{\partial E}{\partial t_i} = -y_i + \frac{1}{\sum_j^n e^{t_j}} * e^{t_i}$. А это то же самое, что и $\frac{\partial E}{\partial t_i} = -y_i + \sigma(T)_i = z - y_i$. Поэтому в общем виде эта формула выглядит так:

$$\frac{\partial E}{\partial T_2} = z - y$$

Перейдем к выведению формулы для промежуточного элемента, который был обозначен за известный $\frac{\partial E}{\partial H}$. Здесь действуем по аналогии с $\frac{\partial E}{\partial X}$. Тогда

$$\frac{\partial E}{\partial H} = \frac{\partial E}{\partial T_2} * W_2^T$$

Вернемся к нахождению $\frac{\partial E}{\partial B_2}$, $\frac{\partial E}{\partial W_2}$. Аналогично выводим формулы в общем виде как для $\frac{\partial E}{\partial W_1}$ и $\frac{\partial E}{\partial B_1}$.

$$\frac{\partial E}{\partial W_2} = \frac{\partial E}{\partial T_2} * H^T \text{ и } \frac{\partial E}{\partial B_2} = \frac{\partial E}{\partial T_2} .$$

Теперь, когда все элементы собраны и общие формулы выведены, это можно перенести на программную основу. Нейронная сеть в программном виде представляет собой два файла с расширением «.ру», которые могут взаимодействовать друг с другом. Первый файл содержит в себе саму нейронную сеть, а второй файл программу для ее обучения. По окончании действия второго файла в терминал текстового редактора выводится время, потраченное на обучение, и точность. Далее подобранные обучаемые параметры экспортируются в первый файл, и сеть уже можно использовать. Обратим внимание на то, что не всегда с первого запуска программы получается необходимый результат. Поэтому в ходе обучения сети было принято решение увеличить количество нейронов в скрытом слое от трех до десяти и воспользоваться дополнительными инструментами языка Python для получения желаемого результата. Подводя итог обучения сети, был получен следующий результат: точность выполнения задачи классификации равна 0,98 (или 98%), затраченное время на обучение сети составило 0,317730665206909 сек. ≈ 0,3 сек. и время на выполнение

основной программы – 0,500362873077393 сек. ≈ 0,5 сек. Заметим, что любое время зависит от мощности ПК, на котором проводилось обучение и тест сети.

3. Заключение

В этой работе была проанализирована архитектура под названием «однослойный перцептрон» [7] и проанализирован алгоритм ее обучения, а также изложены основные положения, лежащие в основе построения архитектур нейронных сетей прямого распространения сигнала. На основе проведенных расчетов была создана нейронная сеть, решающая распространенную задачу классификации. Данную технологию возможно усовершенствовать и использовать в различных системах, в том числе, использующих компьютерное зрение в качестве инструмента измерения (или распознавания) каких-либо параметров в режиме реального времени.

Список литературы

[1]. Нейронные сети на Python [Электронный ресурс]. – Режим доступа: https://kpfu.ru

[2]. М. Б. Беркинблит Нейронные сети, глава 5, 1993 г., стр. 72-86.

[3]. Ф. Уоссермен Нейрокомпьютерная техника: Теория и практика, Перевод на русский язык, Ю. А. Зуев, В. А. Точенов, 1992.

[4]. Принцип обучения многослойной нейронной сети с помощью алгоритма обратного распространения [Электронный ресурс]. - Режим доступа: https://robocraft.ru/algorithm/560

[5]. Коршунов Ю. М., Коршунов Ю. М. Математические основы кибернетики, 1972.

[6]. Дубровин Б. А., Новиков С. П., Фоменко А. Т. Современная геометрия методы и приложения: учебное пособие для физико-математических специальностей университетов, 1986 г.

[7]. Марвин Ли Минский и Сеймур Пейперт Перцептроны, 1969 г.