МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)**

Институт информационных технологий, математики и механики

Направление подготовки: «Фундаментальная информатика и информационные технологии»
Магистерская программа: «Когнитивные системы»

ОТЧЕТ

по лабораторной работе №1

на тему:

«Реализация метода обратного распространения ошибки для двухслойной полносвязной сети»

Выполнил(а): студент группы 381806-4м
Кильдишев Максим Геннадьевич

Нижний Новгород
2019

Оглавление

Постановка задачи

Выполнение лабораторной работы предполагает решение следующих задач:

1. Изучение общей схемы метода обратного распространения ошибки
2. Вывод математических формул для вычисления градиентов функции ошибки по параметрам нейронной сети и формул коррекции весов
3. Проектирование и разработка программной реализации
4. Тестирование разработанной программной реализации
5. Подготовка отчёта, содержащего минимальный объём информации по каждому этапу выполнения работы

Метод обратного распространения ошибки

Метод обратного распространения ошибки определяет стратегию изменения весов ИНС в ходе обучения, используя градиентные методы оптимизации.

$$w(k+1) = w(k) + \eta * \big(-\nabla \mathrm{E}(w)\big),$$

где $w(k) -$ веса ИНС на $k -$ ом шаге, $\eta -$ скорость обучения, $-\nabla \mathrm{E}(w)$ $-$ направление антиградиента функции ошибки

Изначально веса ИНС заполняются согласно выбранному программистом правилом.

В данной лабораторной работе был выбрано правило Ксавье для заполнения весов. Далее для каждого примера из обучающей выборки выполняются следующие шаги:

1. Прямой проход. Принятые данные распространяются от начала сети к её концу.
2. Вычисление значения функции ошибки на выходном слое.
3. Если критерий останова не выполнен, то происходит пересчёт весов сети согласно формуле выше и подача на вход сети следующего примера, иначе останов.

Критерием останова может служить достигнутая точность или количество итераций. В этой лабораторной работе критерием останова является количество итераций(эпох). Целью изменения весов является минимизация функции ошибки E.

Вывод формул для вычисления градиентов функции ошибки

Для начала укажем функции активации на скрытом и выходном слое, а также функцию ошибки для решаемой задачи классификации.

$$\varphi_1(m_j) = \begin{cases} m_j, \text{если } m_j > 0 \\ 0, \text{если } m_j \leq 0 \end{cases} - \text{функция активации на скрытом слое } (ReLu)$$

$$\varphi_2(r_j) = \frac{e^{r_j}}{\sum_{i=0}^{n} e^{r_i}} - \text{функция активации на выходном слое}(Softmax)$$

$$E(w) = \sum_{j=0}^{M} y_j \ln(\mathrm{u_j}) - \text{функция ошибки}(Cross\ entropy)$$

Ниже будут приведены выкладки с выводом формул для вычисления градиентов.

$$u_j = \varphi_2\left(\sum_{s=0}^{K} w_{js}^{(2)} v_s\right), j = \overline{0, M}$$

$$v_s = \varphi_1\left(\sum_{i=0}^{N} w_{si}^{(1)} x_i\right), s = \overline{0, K}$$

Производная по выходному слою:

$$\frac{\partial E}{\partial w_{js}^{(2)}} = \sum_{i=0}^{M} y_i \frac{\partial(\ln(u_i))}{\partial w_{js}^{(2)}} = \sum_{i=0}^{M} y_i \frac{\partial(\ln(u_i))}{\partial u_i} \frac{\partial u_i}{\partial w_{js}^{(2)}}$$

Отдельно вычислим производные первого и второго множителя суммы.

$$\frac{\partial(\ln(u_j))}{\partial u_j} = \frac{1}{u_j}$$

$$\frac{\partial u_i}{\partial w_{js}^{(2)}} = \left(\frac{e^{\sum_{s=0}^{K} w_{is}^{(2)} v_s}}{\sum_{k=0}^{M} e^{\sum_{s=0}^{K} w_{ks}^{(2)} v_s}}\right)`_{w_{js}^{(2)}} = \begin{bmatrix} u_j(1-u_j)v_s, & i = j \\ -u_j u_i v_s, & i \neq j \end{bmatrix}$$

При вычислении производной функции softmax стоит отметить, что $\sum_{j=0}^{M} y_j = 1$, потому формулу можно упростить следующим образом:

$$\frac{\partial E}{\partial w_{js}^{(2)}} = y_j(1-u_j)v_s + \sum_{i=0}^{M/j} -y_i u_i v_s = v_s(y_j - u_i)$$

Производная по скрытому слою:

$$\frac{\partial E}{\partial w_{si}^{(1)}} = \sum_{i=0}^{M} y_i \frac{\partial(\ln(u_i))}{\partial w_{si}^{(1)}} =$$

$$= \sum_{i=0}^{M} y_i \frac{\partial(\ln(u_i))}{\partial u_i} \frac{\partial u_i}{\partial w_{si}^{(1)}} = \sum_{i=0}^{M} \frac{y_j}{u_j} \frac{\partial u_j(\sum_{s=0}^{K} w_{js}^{(2)} v_s)}{\partial(\sum_{s=0}^{K} w_{js}^{(2)} v_s)} \frac{\partial(\sum_{s=0}^{K} w_{js}^{(2)} v_s)}{\partial w_{si}^{(1)}}$$

Второй множитель в сумме – производная softmax, которая уже вычислена.

Вычислим отдельно производную третьего множителя.

$$\frac{\partial\left(\sum_{s=0}^{K} w_{js}^{(2)} v_s\right)}{\partial w_{si}^{(1)}} = w_{js}^{(2)} \frac{\partial \varphi^{(1)}(\sum_{i=0}^{N} w_{si}^{(1)} x_i)}{\partial\left(\sum_{s=0}^{N} w_{si}^{(1)} x_i\right)} \frac{\partial \varphi^{(1)}(\sum_{i=0}^{N} w_{si}^{(1)} x_i)}{\partial w_{si}^{(1)}} =$$

$$= w_{js}^{(2)} v_s' x_i$$

Вернёмся к исходной производной:

$$\frac{\partial E}{\partial w_{si}^{(1)}} = (y_j(1-u_j) w_{js}^{(2)} x_i - \sum_{i=0}^{M/j} y_i u_i w_{js}^{(2)} x_i) v_s' =$$

$$= (y_j w_{js}^{(2)} x_i - u_i w_{js}^{(2)} x_i)\, v_s' = (y_j - u_i) w_{js}^{(2)} x_i v_s'$$

Программная реализация

Перед запуском настройте окружение: необходимо установить пакеты *tensorflow, numpy, keras*. Это можно сделать с помощью менеджера пакетов *pip*. Классифицировать будем числа из дата-сета MNIST. Работа представлена тремя файлами: lab1_impl.py, lab1_keras.py, main.py, где

- **lab1_impl.py** – реализация двуслойной полносвязной сети
- **lab1_keras.py** – реализация двуслойной полносвязной сети посредством Keras
- **main.py** – файл с помощью которого можно запустить ту или иную реализацию

Рассмотрим **lab1_impl.py**. Класс ***Network*** решает задачу классификации. Для того, чтобы обучить ИНС необходимо создать экземпляр класса ***Network***. Конструктор содержит следующие параметры:

- *hidden_nodes_num* – количество нейронов на скрытом слое
- *output_nodes_num* – количество нейронов на выходном слое
- *rate* – скорость обучения
- *debug* – режим отладки (False по-умолчанию)

После создания экземпляра, вызовите функцию ***fit***. Рассмотрим её параметры:

- *X* – входные данные
- *Y* – корректные ответы
- *batch_size* – размер пачки
- *number_epochs* – количество эпох

Функция ***evaluate***. Вычисляет ошибку и точность на предоставленной выборке. Параметры:

- *X* – входные данные

- *Y* – корректные ответы

Также вне класса имеется функция ***run***. Вызвав её, вы можете сразу обучить нейронную сеть и проверить результат на тренировочной и тестовой выборках. Параметры:

- *hidden_nodes_num* - количество нейронов на скрытом слое
- *batch_size* – размер пачки
- *rate* – скорость обучения
- *number_epochs* – количество эпох

Чтобы запустить лабораторную работу, откройте командную строку в папке с файлами.

Общая структура строки запуска такова:

python main.py --type <type> --hidden_num <num> --batch_size <size> --rate <rate> --epochs <epochs_num>

- ***--type*** – тип ИНС. Возможные значения: *keras, lab.* По-умолчанию *lab*
- ***--hidden_num*** – количество нейронов на скрытом слое. Целое число. По-умолчанию *240*
- ***--batch_size*** – размер пачки. Целое число. По-умолчанию *128*
- ***--rate*** – скорость обучения. Число от 0 до 1. По-умолчанию 0.3
- ***--epochs*** – количество эпох. Целое число. По-умолчанию 5

Результаты эксперимента

Ниже приведены таблицы с результатами эксперимента с разными параметрами.

Параметры запуска: *hidden_num = 50, batch_size = 128, rate = 0.1, epochs = 5*

	Time(s)	Train loss	Train accuracy	Test loss	Test accuracy
lab1_impl	11.255392	0.18642	0.94681	0.16450	0.9511
lab1_keras	4.375067	0.17038	0.951483	0.1727993	0.948899

Параметры запуска: *hidden_num = 128, batch_size = 128, rate = 0.1, epochs = 5*

	Time(s)	Train loss	Train accuracy	Test loss	Test accuracy
lab1_impl	23.411873	0.138347	0.959233	0.169260	0.9516
lab1_keras	6.959487	0.14138	0.96121	0.1457872	0.9577

Параметры запуска: *hidden_num = 300, batch_size = 128, rate = 0.1, epochs = 20*

	Time(s)	Train loss	Train accuracy	Test loss	Test accuracy
lab1_impl	343.715848	0.037844	0.9902	0.072661	0.9767
lab1_keras	41.516283	0.039811	0.990583	0.071598	0.9775

Параметры запуска: *hidden_num = 300, batch_size = 128, rate = 0.3, epochs = 20*

	Time(s)	Train loss	Train accuracy	Test loss	Test accuracy
lab1_impl	365.367392	0.00823	0.99913	0.0639	0.9802
lab1_keras	47.787520	0.008403	0.999433	0.05816	0.9811