Федеральное Государственное бюджетное учреждение

высшего профессионального образования

"Заполярный Государственный Университет имени Н. М. Федоровского"

Кафедра «Информационные системы и технологии»

Нейронные сети

Отчет по лабораторной работе №2

Тема: Сеть обратного распространения ошибки

Выполнила:

студентка группы ИЭ-21

Толкачева Полина Васильевна

Проверила:

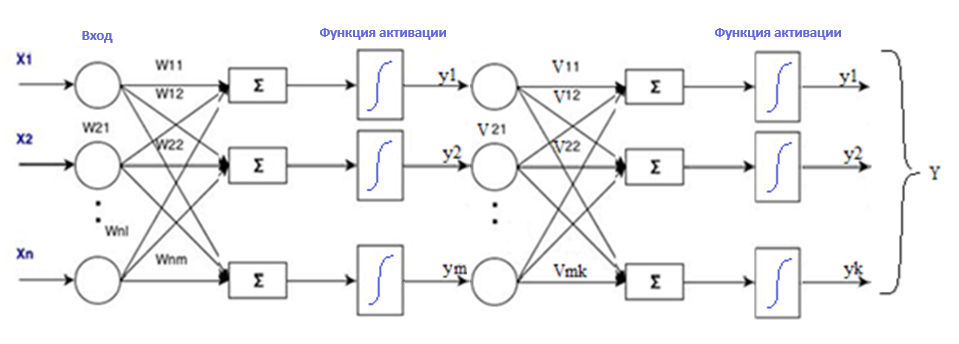
Дыптан Елена Арнольдовна

Норильск 2024

**Цель:** сформировать обучающую и тестовые выборки, реализовать алгоритм обучения и функционирования «Сети обратного распространения ошибки», обучить и протестировать сеть.

**Архитектура сети**

Архитектура сети представляет собой многослойную нейронную сеть с одним скрытым слоем следующего вида:

****

Где Xn: 300 (количество входов)

Yn: 10 (количество выходов)

Активационная функция каждого нейрона Yn – логическая (сигмоидальная) и имеет вид:

Yn = 1/(1+e^(-as) )

где Sn = ∑(xi \* wni)

a=1 (наклон)

На входе нейронная сеть получает вектор X = [X0,X1,…X299]

На выходе нейронная сеть возвращает вектор Y = [Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9]

Ошибка считается по алгоритму вычисления функции потерь с использованием кросс-энтропии:

Определит epsilon = 1e-10.

Ограничит значения y\_pred (в данном случае изменения не произойдет, так как все значения уже в пределах допустимого интервала).

Найдет индексы истинных классов: [0, 1].

Выберет предсказанные вероятности для истинных классов: [0.7, 0.6].

Вычислит отрицательные логарифмы: [-log(0.7), -log(0.6)] ≈ [0.357, 0.511].

Найдет среднее значение этих логарифмов: (0.357 + 0.511) / 2 ≈ 0.434.

Алгоритм обучения

1. Инициализировать весовую матрицу W и V небольшими случайными значениями от -0.3 до 0.3. Задать начальное значения a, и max кол-во эпох обучения, а также коэффициент точности Ꜫ
2. Вычисления текущего выходного сигнала, использовали функцию активации типа сигмоид, как на скрыто, так и на внешнем слоях
3. Настройка синаптических весов по формуле

Для выходного слоя:

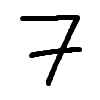
V\_jk^N=V\_jk^N-α ∂E/(∂V\_jk^N ); ∂E/(∂V\_jk^N )=(y\_k-y)\*z\_j

Для скрытого слоя:

W\_ij^N=W\_ij^N-α ∂E/(∂W\_ij^N ); ∂E/(∂W\_ij^N )=(y\_k-y)\*z\_j\*x\_i

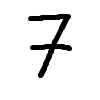
1. Шаги 2- 3 повторяются для всех обучающих векторов. Обучение заканчивается, когда для каждого образа обучающие выборки в течении 1 эпохи ошибка не превышает Ꜫ или когда превышено max кол-во эпох.

**Обучающая выборка**



Размер обучающей выборки:

**Тестовая выборка**



Размер тестовой выборки:

|  |  |  |  |
| --- | --- | --- | --- |
| Нейронов в скрытом слое | Реальное кол-во эпох | Фиксированное кол-во эпох | Ошибка тестовой выборки |
| 16 | 110 | 300 | 2 |
| 32 | 82 | 300 | 0 |
| 64 | 56 | 300 | 1 |
| 128 | 46 | 300 | 1 |
| 256 | 36 | 300 | 1 |
| 512 | 26 | 300 | 1 |
| 1024 | 18 | 300 | 0 |

**Вывод:** была разработана и протестирована сеть обратного распространения ошибки, состоящая из одного скрытого и одного внешнего слоя. В ходе работы была реализована архитектура сети, определены параметры активационных функций и алгоритмы вычисления ошибок с использованием кросс-энтропии. Проведено обучение сети на выборке из 300 образцов и тестирование на 100 образцах. Настроены весовые коэффициенты для уменьшения ошибки по обучающим данным. Сеть успешно обучена до достижения заданного уровня точности или максимального количества эпох.

**Листинг программы:**

**def train**(self, images, labels): # Метод обучения сети

Y = np.eye(self.num\_classes)[labels] # Преобразуем метки в one-hot векторы

epoch = 0

max\_epochs = 2200

flag=False

while epoch < max\_epochs and not flag: # Цикл обучения

Z1, A1, Z2, A2 = self.forward(images) # Прямой проход

loss = self.cross\_entropy\_loss(Y, A2) # Вычисляем функцию потерь

max\_loss = max(loss) # Средняя потеря по всем образцам

dW1, db1, dW2, db2 = self.backward(images, Y, Z1, A1, Z2, A2) # Обратный проход

self.update\_parameters(dW1, db1, dW2, db2) # Обновление параметров

print(f"Эпоха {epoch}, максимальное потери: {max\_loss}") # Выводим информацию о средних потерях

if max\_loss < 1.5: # Прекращаем обучение, если достигнута необходимая точность

flag=True

epoch += 1

print(f"Обучение завершено после {epoch} эпох") # Сообщаем о завершении обучения

**def gues**s(self, image): # Метод для предсказания класса по изображению

\_, \_, \_, A2 = self.forward(image) # Прямой проход по сети

return A2, np.argmax(A2, axis=1)[0] # Возвращаем выход сети и предсказанный класс