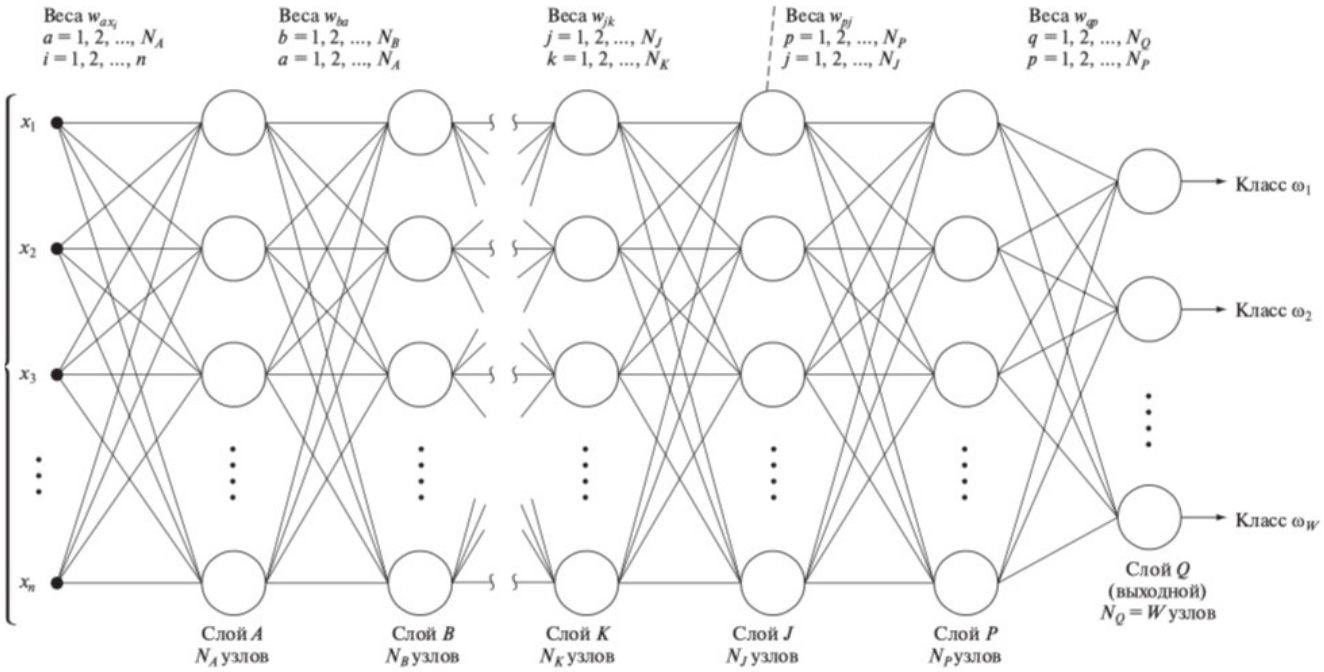
ЛР7: Распознавание образов на основе искусственных нейронных сетей

Цель: Принципы использования искусственных нейронных сетей для распознавания образов.

Теоретическая часть

Нейронные сети состоят из большого числа простейших нелинейных вычислительных элементов (нейронов), которые организованы в виде сетей, напоминающих предположительный способ соединения нейронов в мозге человека. Архитектура модели нейронной сети представлена на рисунке.



Она состоит из слоев, в которых находятся идентичные по структуре вычислительные узлы (нейроны), организованные таким образом, что выход каждого нейрона одного слоя соединяется со входом каждого нейрона следующего слоя.

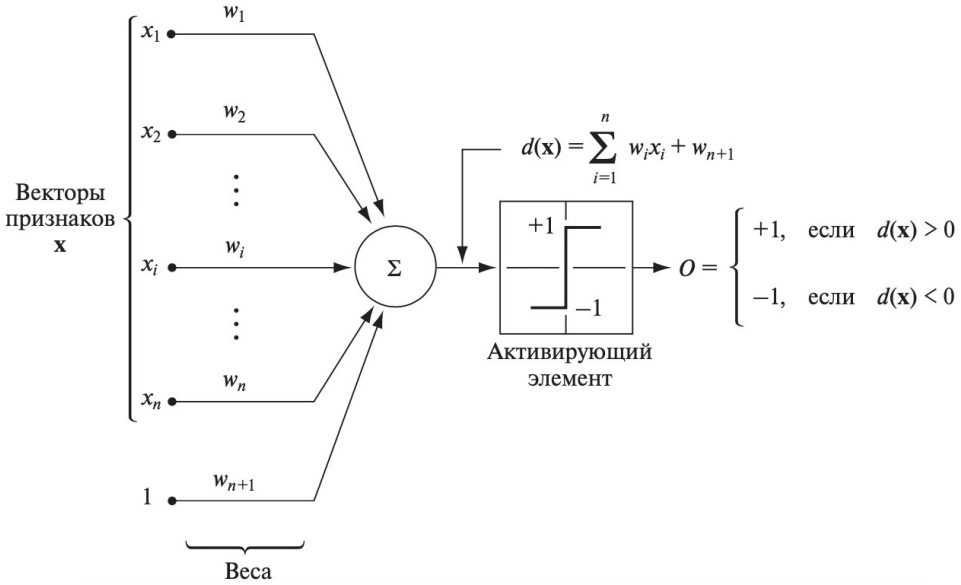
Число нейронов в первом слое, называемом слоем A, равно N_A; оно часто выбирается равным размерности входных векторов-образов: $N_A = n$. Число нейронов выходного слоя, называемого слоем Q, обозначается N_Q. Это число N_Q равно W — числу классов, образы которых данная нейронная сеть обучена распознавать.

Сеть распознает объект с вектором признаков **x** как принадлежащий классу ω_i, если на i-м выходе сети присутствует «высокий» уровень, а на остальных выходах — «низкий», что разъясняется в дальнейшем.

Выходной сигнал (реакция) нейрона базируется на взвешенной сумме его входных сигналов:

$$d(\mathbf{x}) = \sum_{i=1}^{n} w_i\, x_i + w_{n+1} \qquad (1.1)$$

Коэффициенты w_i, называемые весами, масштабируют входные сигналы перед тем, как они суммируются и подаются на пороговое устройство. В этом смысле веса аналогичны синапсам в нервной системе человека. Функцию, которая



отображает результат суммирования в конечный выходной сигнал устройства, называют *функцией активации*. В качестве функции активации используется непрерывная S-образная функция (сигмоида), поскольку для обучения сети необходима дифференцируемость вдоль всех путей в нейронной сети:

$$h_j(I_j) = \frac{1}{1 + \exp(-I_j)} \qquad (1.2)$$

где I_j, j = 1, 2,..., N_J — значение на входе активирующего элемента каждого узла слоя J нейронной сети.

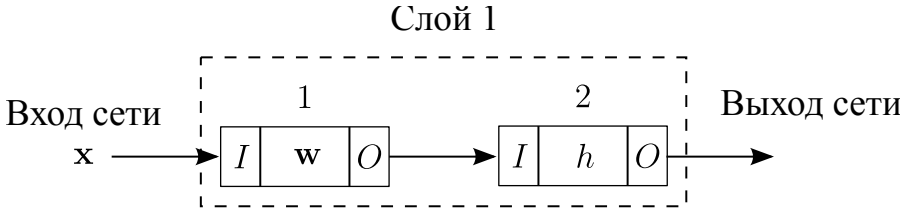
Учтем коэффициент w_{n+1} (смещение или *bias*) в векторе **y** и перепишем уравнение с помощью матричной нотации

$$d(\mathbf{x}) = \sum_{i=1}^{n} w_i\, x_i + w_{n+1} = \mathbf{w}\mathbf{y} \qquad (1.3)$$

где $y = (x_1, x_2,..., x_n, 1)^T$ — расширенный вектор признаков, а $w = (w_1, w_2,..., w_n, w_{n+1})^T$ называется весовым вектором.

Используемые обозначения

Сигнал на входе элементов i=1,2 для J-го слоя сети обозначается I_{ij}, $j = 1,...,N_J$, т.е. I_{11} есть сигнал на входе взвешивающего элемента 1 слоя 1, I_{21}— сигнал на входе активирующего элемента 2 слоя 1 и т.д.



У каждого элемента w в слое J имеется N_K входов, но каждый отдельный вход умножается на свой собственный весовой коэффициент. Так, N_K входов слое J взвешиваются с коэффициентами w_{kj}, $k = 1,...,N_K$. Следовательно, для преобразования выходных сигналов требуется в общей сложности $N_J \times N_K$ коэффициентов. Чтобы полностью описать элементы w в слое J, необходимы еще дополнительные N_J коэффициентов — смещений w_{n+1}.

$$O_{k+1} = L_k(I_k) = h_k(\mathbf{w}_k\, I_k) \qquad (1.4)$$

для J — номер слоя, а w — весовые коэффициенты; h — функция активации; L — J слой нейронной сети.

Обучение путем обратного распространения ошибки

Метод обучения нейронных сетей на каждом шаге обучения минимизирует ошибку между фактической **O** и желаемой реакциями **r**:

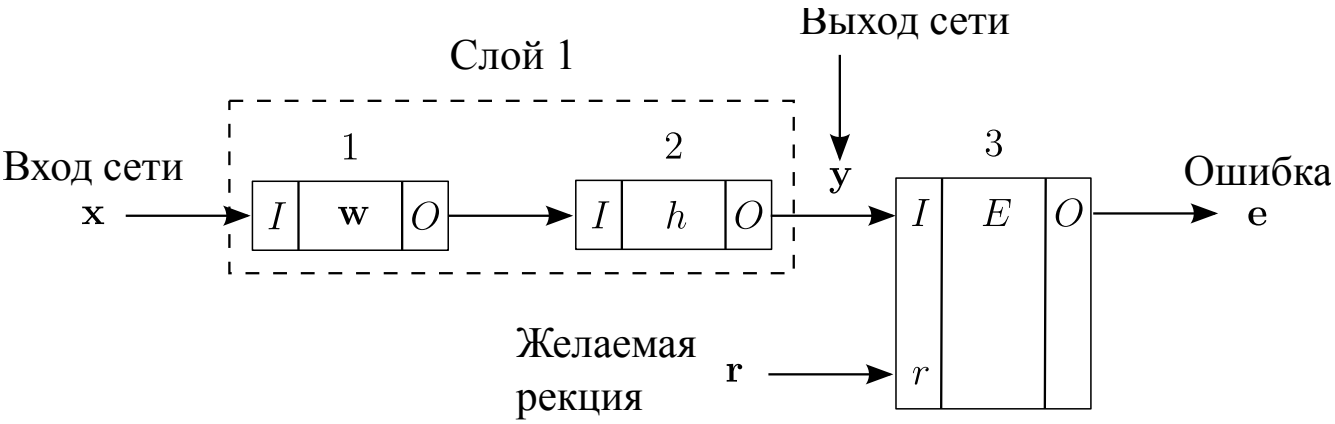
$$E(\mathbf{w}) = (N(\mathbf{w}, \mathbf{x}) - \mathbf{r})^2 = (h(\mathbf{wx}) - \mathbf{r})^2 \quad (1.5)$$

где **r** — желаемая реакция; N — выход однослойной нейронной сети.

Задача состоит в том, чтобы путем последовательных приращений корректировать весовой вектор w в обратном направлении по отношению к вектору градиента функции E(w), чтобы найти минимум этой функции, который достигается при r = N(w,x); т.е. минимум соответствует безошибочной классификации.
Если обозначить через w(k) весовой вектор на k-м шаге итерации, то в обобщенном виде можно записать следующий алгоритм градиентного спуска:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu\left(\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}\right) \quad (1.6)$$

где w(k+1) — новое значение вектора w, а параметр $\mu > 0$ задает коэффициент скорости сходимости алгоритма. От выбора параметра μ зависит устойчивость и скорость сходимости алгоритма. На практике используется интервал значений $0{,}1 < \mu < 1{,}0$.



Распишем уравнение частной производной E(w), в помощью уравнения производной сложной функции:

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial E(\mathbf{w})}{\partial \mathbf{O}_2} \frac{\partial \mathbf{O}_2}{\partial \mathbf{O}_1} \frac{\partial \mathbf{O}_1}{\partial \mathbf{w}} \quad (1.7)$$

Частная производная функции ошибки E(w) равна:

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{O}_2} = \frac{\partial((\mathbf{O}_2 - \mathbf{r})^2)}{\partial \mathbf{O}_2} = 2(\mathbf{O}_2 - \mathbf{r}) \quad (1.8)$$

Частная производная функции h(I) равна:

$$\frac{\partial \mathbf{O}_2}{\partial \mathbf{O}_1} = \frac{\partial(h(\mathbf{O}_1))}{\partial \mathbf{O}_1} = \mathbf{O}_2(1 - \mathbf{O}_2) \quad (1.9)$$

Частная производная функции wx равна:

$$\frac{\partial \mathbf{O}_2}{\partial \mathbf{w}} = \frac{\partial(\mathbf{w}\mathbf{x})}{\partial \mathbf{w}} = \mathbf{x} \quad (1.10)$$

Процесс обучения начинается с произвольного набора весов в узлах сети (но не всех одинаковых). После этого процесс обучения на любом шаге итерации складывается из двух основных этапов:

1) На первом этапе на вход сети предъявляется обучающий вектор признаков **x**, и сигналы распространяются по слоям сети. Затем выход сети сравнивается с желаемыми выходными сигналами **r**, в результате чего строятся составляющие ошибок **e**.

2) На втором этапе осуществляется обратный проход по сети, при котором сигнал ошибки передается о обратной направлении по сети, что позволяет нужным образом откорректировать его веса. Эта процедура применяется и к смещениям, которые, как говорилось выше, рассматриваются в качестве дополнительных весовых коэффициентов, на которые умножается единичный сигнал, подаваемый на вход сумматора каждого узла нейронной сети.

Обычная практика состоит в том, чтобы прослеживать ошибки сети **e**. При успешном сеансе обучения величина ошибки сети уменьшается по мере роста числа итераций, и процедура обучения сходится к устойчивому набору весовых векторов, в котором в случае дополнительного обучения наблюдаются лишь небольшие флуктуации. Для выяснения того, что в процессе обучения некоторый образ классифицируется правильно, необходимо проверить, что реакция узла выходного слоя, сопоставляемого тому классу, к которому принадлежит данный образ, имеет высокий уровень, а сигналы остальных узлов выходного слоя имеют низкий уровень.

После того, как обучение системы закончено, она применяется для классификации неизвестных образов с использованием тех значений параметров **w**, которые были установлены в процессе обучения. Сигналы любого поступающего образа свободно распространяются по всем слоям, и образ классифицируется как принадлежащий классу, который соответствует узлу с высоким уровнем на выходе. Если высокий уровень отмечается сразу на нескольких узлах выходного слоя сети или не обнаруживается ни на одном из узлов, то либо объявляется об отказе от классификации, либо принимается решение отнести объект к тому классу, на выходном узле которого присутствует сигнал максимальной амплитуды.

Учебный набор данных

Набор данных data.mat содержит примеры рукописных цифр от 0 до 9. Набор данных состоит из 3 наборов данных: обучение (training — 1000 примеров), проверка (validation), тест (test — 9000 шт). Каждый набор данных состоит из двух структур данных inputs и targets. Inputs — матрица размерами 256x1000 примеров, столбец матрицы представляется собой преобразованное в вектор изображение. Для отображения использовать команду imagesc(reshape(data.training.inputs(:, 1), 16, 16)). Targets — матрица размерами 10x1000 примеров, столбец матрицы представляется собой onehot-код изображенной цифры. На картинке data.training.inputs(:, 1) изображена цифра 0, data.training.targets(:, 1) равен вектору [1,0,0,0,0,0,0,0,0,0].

Ход работы

1. **Подготовка сети:**
   a) Загрузить набор данных (dataset) с помощью команды load('data.mat').
   b) Составить уравнение однослойной сети с помощью уравнения (1.4).
   c) Инициализировать веса случайными значениями w=rand(10,256).
2. **Обучение сети**

a) Обучить сеть на 100 эпохах. Использовать для обучения набор данных data.training. В рамках одной эпохи рассчитывается среднее значение ошибка E для всех классов по формуле (1.5). Желаемая реакция r=data.training.targets. Обновить веса в соотвествии с уравнением (1.6).
b) Построить график средней ошибки E от эпохи.

3. **Тест сети**
   a) Выполнить проверку обученной сети на тестом наборе данных data.test. Посчитать количество правильной распознанных цифр.