Проектирование интеллектуальных систем

Лекция № 4. Логистическая регрессия и метод максимального правдоподобия

Содержание

Логистическая регрессия

Градиентный спуск нельзя применить для обучения нейронной сети, в которой используется пороговая функция активации. Можно выделить несколько причин этому. Во-первых, эта функция не является непрерывно дифференцируемой, так как она терпит разрыв в точке, соответствующей значению порога. Во-вторых, во всех других точках значение производной будет равно 0.

По вышеупомянутым причинам пороговую функцию в современных нейронных сетях не используют. Её заменяют «гладким аналогом» – **сигмоидой** (логистической функцией). Сигмоида обычно обозначается символом σ и имеет следующий вид

$$\sigma(v) = \frac{1}{1 + e^{-v}}.$$

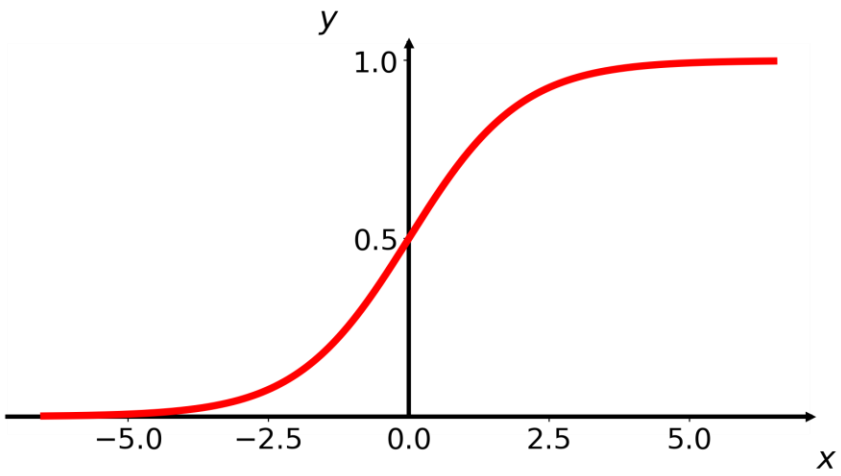

Рис. 1. График сигмоиды.

Помимо возможности применять градиентные методы оптимизации, использование сигмоиды имеет ещё одно очень важное преимущество – вместо дискретных значений $\{0,1\}$ на выходе нейрона получается число из диапазона $(0,1)$. Таким образом, если рассматривать нейрон как линейный классифика-

тор, можно судить об «уверенности» классификатора в ответе. Для объектов, находящихся поблизости от разделяющей границы, значение сигмоиды будет $\approx 0{,}5$. Для объектов, лежащих далеко от границы, – значение будет стремиться к 0 или к 1 в зависимости от того, по какую сторону от разделяющей границы находится объект.

Также значению сигмоиды можно дать вероятностную интерпретацию. Если обозначить метки классов числами 0 и 1, то можно сказать, что выходное значение нейрона можно интерпретировать как вероятность того, что данный объект принадлежит «первому» классу $\hat{y} = \sigma(v) = P(y = 1 \mid \boldsymbol{x})$.

Таким образом, при переходе от пороговой функции к сигмоиде происходит переход от прогнозирования конкретных значений y к прогнозированию вероятности $P(y = 1 \mid \boldsymbol{x})$ или $P(y = 0 \mid \boldsymbol{x}) = 1 - P(y = 1 \mid \boldsymbol{x})$.

Нейрон с логистической функцией активации эквивалентен модели *логистической регрессии*. Чтобы понять, как осуществляется переход от взвешенной суммы признаков (также называемых *факторами*) к оценке вероятности, необходимо рассмотреть основные шаги вывода формулы для логистической кривой.

Заметим, что значение $v = w_1 x_1 + \cdots + w_m x_m + b$ может быть любым ($v \in \mathbb{R}$), в то время как вероятность – это число от нуля до единицы. Поэтому первый шаг вывода формулы логистической кривой – это ввод величины «шанса», которая может принимать значения $[0, +\infty)$. Для этого одно из значений переменной y обозначают как интересующее «событие» или «успех» (например, $y = 1$). Тогда шансы (англ. odds) вычисляются как отношение вероятности успеха к вероятности неудачи:

$$Odds = \frac{P(y = 1 \mid \boldsymbol{x})}{1 - P(y = 1 \mid \boldsymbol{x})} \in [0, +\infty).$$

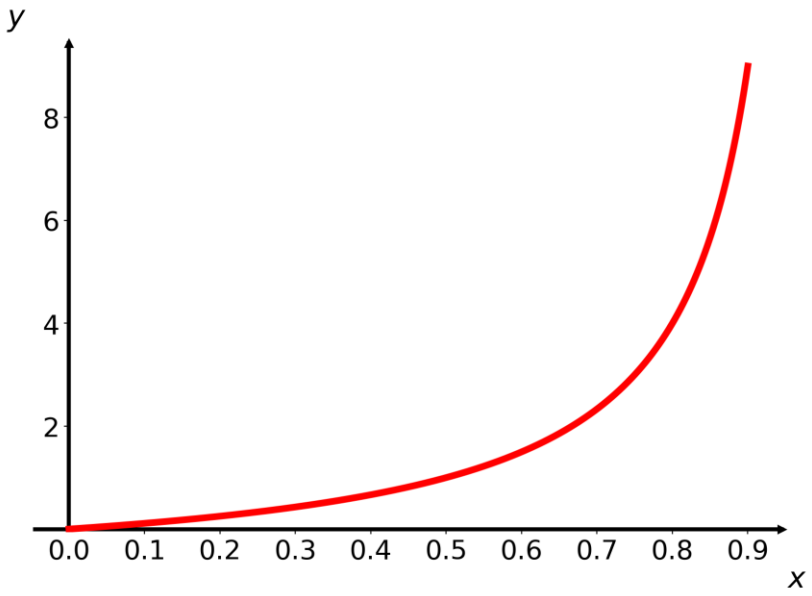

Рис. 2. График «шансов».

По рис. 13 видно, что величина Odds не ограничена сверху, но она не может принимать отрицательных значений, в отличие от взвешенной суммы факторов. Чтобы решить эту проблему, можно вычислить натуральный логарифм от величины шансов

$$\ln\left(\frac{P(y=1 \mid \boldsymbol{x})}{1-P(y=1 \mid \boldsymbol{x})}\right) \in \mathbb{R}.$$

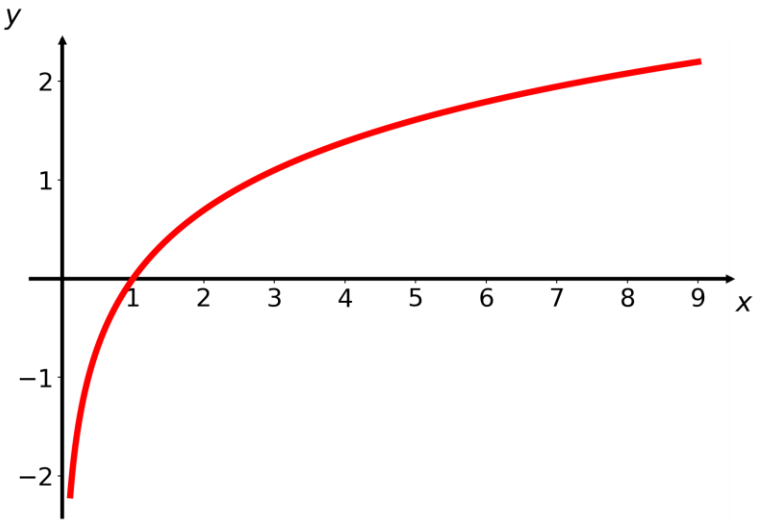

Рис. 3. Натуральный логарифм шансов.

Таким образом, можно считать, что взвешенная сумма факторов равна значению логарифма шанса положительного исхода. Для краткости обозначим $p = P(y = 1 \mid \boldsymbol{x})$. Тогда можно записать следующее

$$logit(p) = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_m \cdot x_m + b,$$

где $logit(p) = \ln\left(\frac{p}{1-p}\right)$.

Обозначим за v взвешенную сумму факторов и найдём выражение для p.

$$\ln\left(\frac{p}{1-p}\right) = v$$

$$\frac{p}{1-p} = e^v$$

$$p = e^v \cdot (1-p) \rightarrow p = \frac{e^v}{1+e^v}$$

$$p = \frac{e^v}{e^v(e^{-v}+1)} = \frac{1}{1+e^{-v}}$$

Метод максимального правдоподобия

Чтобы обучить нейронную сеть, нужно определить функцию потерь, которая будет минимизироваться в процессе обучения. Чтобы можно было использовать градиентные методы оптимизации, эта функция должна быть дифференцируемой. Так как значение сигмоиды можно интерпретировать как оценку вероятности, для формулировки функции потерь при решении задачи бинарной классификации можно воспользоваться *методом максимального правдоподобия*.

При помощи логистической регрессии можно получить оценку вероятности $p_i = P(y = 1 \mid \mathbf{x_i}, \mathbf{w})$ для некоторго объекта $\mathbf{x_i}$ и некоторых значений параметров $\mathbf{w}$. Вероятность $P(y = 1 \mid \mathbf{x_i})$ изначально известна, так как известно, к какому классу принадлежит объект. Целевая переменная y для каждого объекта $\mathbf{x_i}$ имеет вырожденное распределение Бернулли с вероятностью успеха $p_i = 1$ или $p_i = 0$ в зависимости от того, к какому классу принадлежит объект.

В основе метода максимального правдоподобия лежит идея о том, что значения параметров нужно подбирать таким образом, что они выглядели наиболее «правдоподобно», то есть в некотором смысле наилучшим образом описывали имеющиеся данные. Максимум правдоподобия будет достигнут тогда, когда вероятности $P(y = y_i \mid \mathbf{x_i}, \mathbf{w})$ будут равны единице для всех объектов $\mathbf{x_i}$ из обучающей выборки.

Если считать наблюдения в выборке независимыми, то функцию правдоподобия можно записать в следующем виде

$$l(\mathbf{w}) = P(y \mid T, \mathbf{w}) = \prod_{i=1}^{n} P\,(y = y_i \mid \mathbf{x_i}, \mathbf{w}) = \prod_{i=1}^{n} p_i^{y_i} \cdot (1 - p_i)^{(1-y_i)}.$$

Оптимальными значениями параметров $\mathbf{w}^*$ можно считать такие, при которых достигается максимум функции правдоподобия

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}}\, l(\mathbf{w}).$$

В выражении для $l(\mathbf{w})$ требуется вычислить произведение большого количества величин, лежащих на отрезке $[0,1]$, в результате чего может получиться очень маленькое число, выходящее за пределы разрядной сетки компьютера. Чтобы не допустить этого, можно работать с суммой, а не с произведением. Чтобы получить сумму, прологарифмируем выражение для $l(\mathbf{w})$:

$$\log l(\mathbf{w}) = \sum_{i=1}^{n} y_i \cdot \log p_i + (1 - y_i) \cdot \log(1 - p_i).$$

Перекрёстная энтропия и логистическая функция потерь

Наиболее часто встречаются задачи оптимизации, в которых нужно найти минимум некоторой функции. Если поставить минус перед логарифмом функции правдоподобия, получим следующую функцию потерь

$$L(\mathbf{w}) = -\sum_{i=1}^{n} y_i \cdot \log p_i + (1 - y_i) \cdot \log(1 - p_i),$$

которую называют **перекрёстной энтропией** (англ. cross-entropy) или negative log-likelihood loss. Эту функцию используют при обучении нейронной сети решать задачу *бинарной классификации*.

Если обозначить классы числами -1 и 1 (т. е. $y \in \{-1,1\}$), то можно вывести эквивалентную функцию потерь, зависящую от величины отступа. В этом случае функция правдоподобия будет иметь вид

$$l(\mathbf{w}) = P(y \mid T, \mathbf{w}) = \prod_{i=1}^{n} P\,(y = y_i \mid \mathbf{x_i}, \mathbf{w}) = \prod_{i=1}^{n} \sigma\,(M(\mathbf{x_i})).$$

Здесь использовано следующее свойство логистической функции

$$\sigma(-v) = 1 - \sigma(v).$$

Возьмём логарифм функции правдоподобия:

$$\log l(\mathbf{w}) = \sum_{i=1}^{n} \log \frac{1}{1 + e^{-M(\mathbf{x_i})}} = -\sum_{i=1}^{n} \log\left(1 + e^{-M(\mathbf{x_i})}\right).$$

Отсюда можно получить выражение для логистической функции потерь

$$L(\mathbf{w}) = \sum_{i=1}^{n} \log\left(1 + e^{-M(\mathbf{x_i})}\right),$$

где $M(\mathbf{x_i}) = y_i((\mathbf{x_i}, \mathbf{w}) + b)$.

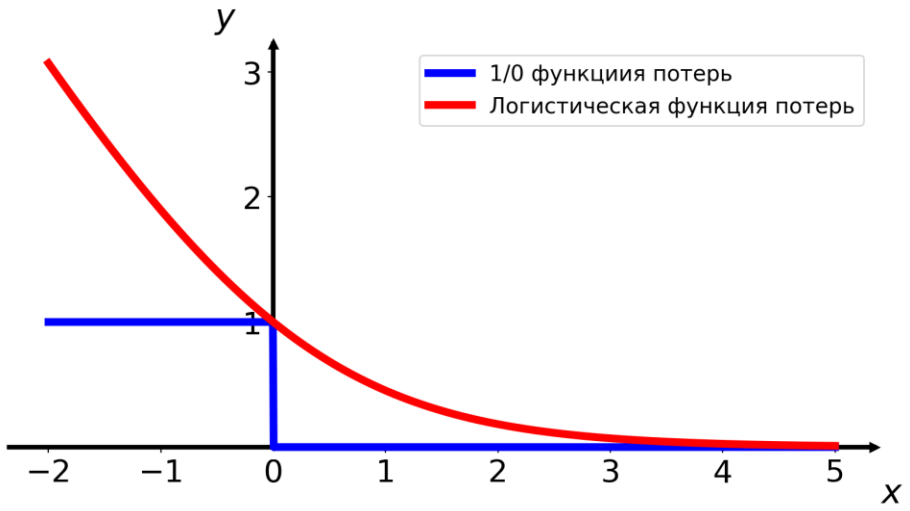

Рис. 4. Логистическая функция потерь.

По рис. 15 видно, что логистическая функция потерь является верхней оценкой 1/0 функции потерь (англ. zero-one loss). В отличие от 1/0 функции потерь, логистическая функция потерь (или эквивалентная ей перекрёстная энтропия) является непрерывно дифференцируемой функцией, к которой можно применить градиентный спуск.

Для применения градиентного спуска нужно вычислить производные от функции потерь по параметрам нейронной сети. Рассчитаем значение производной по некоторому синаптическому весу w_q.

$$L(\mathbf{w}) = \sum_{i=1}^{n} L_i = -\sum_{i=1}^{n} y_i \cdot \log p_i + (1 - y_i) \cdot \log(1 - p_i),$$

$$\frac{\partial L_c}{\partial w_q} = ?, \;\; 1 \le c \le n, \;\; 1 \le q \le m,$$

$$p_c = \sigma(v_c) = \frac{1}{1 + e^{-v_c}}, \qquad v_c = w_1 x_{1c} + \cdots + w_m x_{mc} + b.$$

$$\frac{\partial L_c}{\partial w_q} = \frac{\partial L_c}{\partial p_c}\frac{\partial p_c}{\partial v_c}\frac{\partial v_c}{\partial w_c} = \left(-\frac{y_c}{p_c} + \frac{1 - y_c}{1 - p_c}\right) \cdot p_c(1 - p_c) \cdot x_{qc} =$$
$$= (-y_c \cdot (1 - p_c) + (1 - y_c) \cdot p_c)\, x_{qc} = (p_c - y_c)\, x_{qc}.$$

Если обозначить $p_c = \hat{y}_c$, и подставить полученное выражение в формулу для обновления весов градиентного спуска, можно заметить, что мы получили такое же выражение, что и дельта-правило (правило Видроу-Хоффа):

$$w_q^{(k+1)} = w_q^{(k)} - \alpha\, (\hat{y}_c - y_c)\, x_{qc}.$$

Решение задачи распознавания рукописных цифр

Наиболее часто искусственные нейронные сети применяются для решения задачи распознавания образов. *Распознавание образов* формально определяется как процесс, в котором получаемый образ/сигнал должен быть отнесён к одному из предопределённых классов (категорий) [4]. В данном разделе рассматривается процесс решения задачи распознавания рукописных цифр с помощью однослойного персептрона. В силу линейности данной модели, она сможет решить не всякую задачу распознавания образов, а лишь те, в которых образы линейно разделимы. Распознавание рукописных цифр – одна из

простейших задач распознавания образов, поэтому она может быть решена с достаточно высокой точностью даже с помощью линейной модели.

Для обучения нейронной сети используется набор данных MNIST. Он содержит 70000 изображений рукописных цифр в градациях серого размером 28 x 28 пикселей:

- 60000 в обучающей выборке,
- 10000 в тестовой выборке.

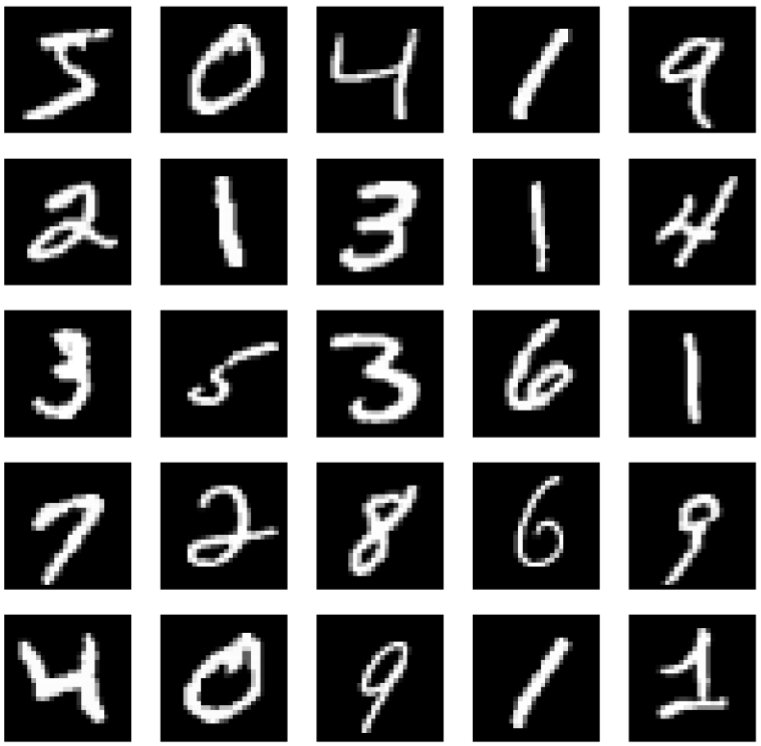

Рис. 5. Примеры изображений из набора данных MNIST.

Задача распознавания рукописных цифр может быть сформулирована в виде задачи многоклассовой (англ. multiclass) классификации. Имеется множество объектов (изображений) $X \subseteq \mathbb{X}$ ($|X| = n$), каждый из которых характеризуется некоторым вектором признаков $\mathbf{x_i}, i = 1, \ldots, n$. Все объекты поделены на классы (группы), каждому из которых соответствует определённая

метка из множества $\mathbb{Y}$. Для каждого объекта $\mathbf{x_i}$ известно, что он принадлежит классу $y_i \in \mathbb{Y}$.

Требуется построить отображение $f: \mathbb{X} \to \mathbb{Y}$, которое сможет каждому объекту сопоставить класс, которому он принадлежит.

Будем использовать следующие обозначения.

$\mathbb{X}$ – все возможные изображения рукописных цифр в градациях серого размером 28 x 28 пикселей («генеральная совокупность»).

$\mathbb{Y} = \{0,1,2,3,4,5,6,7,8,9\}$ – множество возможных меток классов.

X – множество имеющихся в нашем распоряжении изображений ($|X| = n = 60000$).

Y – множество меток классов имеющихся изображений.

$\mathbf{x_i} \in X,\ i = 1, \dots, n$ – вектор признаков, описывающий i-е изображение ($|\mathbf{x_i}| = 28$ x $28 = 784$).

При программной реализации искусственной нейронной сети часто бывает удобно представлять исходные данные в матричном виде:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}.$$

Здесь $\mathbf{X}$ – это матрица «объекты-признаки», в которой каждая строка соответсвует изображению, а каждый столбец – определённому пикселю.

Перед подачей данных персептрону, необходимо, чтобы $x_{ij} \in \{0,1\},\ i = 1, \dots, n;\ j = 1, \dots, m$. Этого можно добиться, применив следующее преобразование:

$$\overline{x}_{ij} = \left[\frac{x_{ij}}{255}\right],$$

где $[\cdot]$ – округление до ближайшего целого, а 255 – максимально возможное значение, которому может равняться x_{ij}, так как на хранение каждого пикселя выделяется 8 бит.

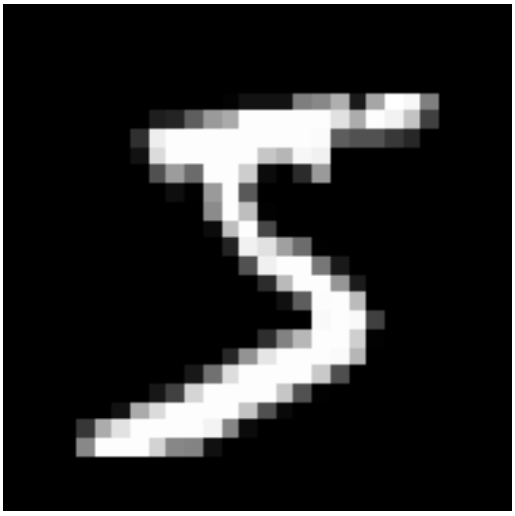

Рис. 6. До предобработки

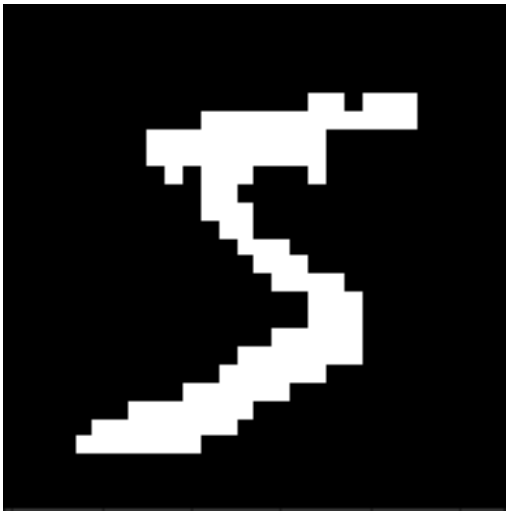

Рис. 7. После предобработки.

Метки классов также нужно подвергнуть предобработке. Так как нейрон может выдавать лишь значения 0 и 1, метки классов нужно представить в **унитарном коде** (англ. one-hot encoding) – двоичном коде фиксированной длины, содержащим только одну единицу. В данном случае длина кода должна быть равна количеству классов, а единица должна стоять в позиции, соответствующей номеру класса.

$$y_{ij} = \begin{cases} 1, & \text{если } j - 1 = y_i; \\ 0, & \text{иначе.} \end{cases}$$

Таким образом, после кодирования вектор-столбец **y** преобразуется в матрицу **Y.**

$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ \dots \\ 7 \\ 8 \\ 9 \end{bmatrix} \quad \rightarrow \quad \mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

На рисунке ниже представлена архитектура нейронной сети для решения поставленной задачи. Сеть состоит из одного слоя, содержащего 10 нейронов, – по одному на каждый класс. Ранее при обозначении синаптического веса использовался лишь один индекс – номер входа, которому этот вес соответствует. Теперь появилась потребность ввести ещё один индекс – номер нейрона, которому этот вес соответствует. Таким образом, запись w_{qj} означает синаптический вес нейрона q, соответсвующий j-му входу. Используя такое обозначение, работу нейрона q можно описать следующим образом:

$$v_q = \sum_{j=1}^{m} w_{qj}\, x_j + b_q,$$

$$\hat{y}_q = \varphi(v_q) = \begin{cases} 1, & \text{если } v_q > 0; \\ 0, & \text{иначе.} \end{cases}$$

Для удобства программной реализации можно включить значения порогов $b_1, \dots, b_k$ в матрицу весов и добавить единичный столбец в матрицу **X**.

$$n = 70000,\ m = 784$$

$$k = 10\ (\text{количество нейронов } =\ \text{количество классов})$$

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} & 1 \\ x_{21} & x_{22} & \dots & x_{2m} & 1 \\ \dots & \dots & \dots & \dots & 1 \\ x_{n1} & x_{n2} & \dots & x_{nm} & 1 \end{bmatrix} \qquad \mathbf{W} = \begin{bmatrix} w_{11} & w_{21} & \dots & w_{k1} \\ w_{12} & w_{22} & \dots & w_{k2} \\ \dots & \dots & \dots & \dots \\ w_{1m} & w_{2m} & \dots & w_{km} \\ b_1 & b_2 & \dots & b_k \end{bmatrix}$$

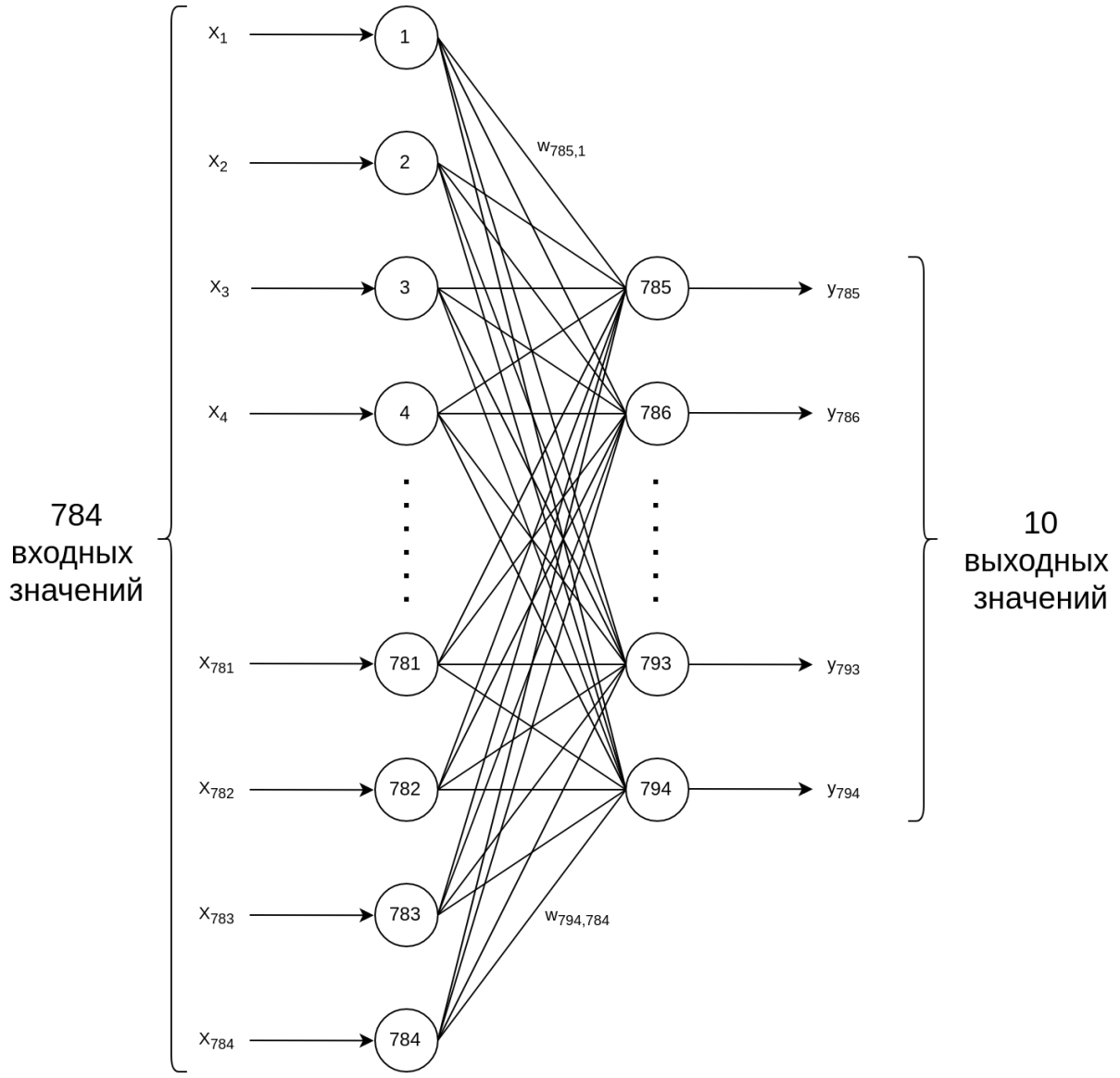

Рис. 8. Архитектура однослойного персептрона для решения задачи распознавания рукописных цифр.

Тогда выходное значение может быть вычислено как матричное произведение **X** на **W** с последующим применением функции активации к каждому элементу получившейся матрицы:

$$\hat{\mathbf{Y}} = \varphi(\mathbf{XW}) = \begin{bmatrix} \hat{y}_{11} & \hat{y}_{12} & \cdots & \hat{y}_{1k} \\ \hat{y}_{21} & \hat{y}_{22} & \cdots & \hat{y}_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ \hat{y}_{n1} & \hat{y}_{n2} & \cdots & \hat{y}_{nk} \end{bmatrix},$$

где $\hat{y}_{nk}$ – выходное значение нейрона k для объекта n.

Обучение нейронной сети будет состоять из следующих шагов.

Шаг 1 Инициализировать матрицу параметров **W** небольшими случайными числами.

Шаг 2 Для каждого объекта $\mathbf{x_i}$ из обучающей выборки:

- Вычислить вектор выходных значений $\hat{\mathbf{y}}_\mathbf{i}$.
- Вычислить вектор ошибок

$$\mathbf{e_i} = \mathbf{y_i} - \hat{\mathbf{y}}_\mathbf{i} = [e_{i1} \quad e_{i2} \quad \cdots \quad e_{ik}]$$

- Вычислить матрицу значений корректировок весов

$$\mathbf{D_i} = \alpha\, \mathbf{x_i}^T \mathbf{e_i} = \alpha \begin{bmatrix} x_{i1} \\ x_{i2} \\ \cdots \\ x_{im} \\ 1 \end{bmatrix} [e_{i1} \quad e_{i2} \quad \cdots \quad e_{ik}] = \alpha \begin{bmatrix} x_{i1}e_{i1} & x_{i1}e_{i2} & \cdots & x_{i1}e_{ik} \\ x_{i2}e_{i1} & x_{i2}e_{i2} & \cdots & x_{i2}e_{ik} \\ \cdots & \cdots & \cdots & \cdots \\ x_{im}e_{i1} & x_{im}e_{i2} & \cdots & x_{im}e_{ik} \\ e_{i1} & e_{i2} & \cdots & e_{ik} \end{bmatrix}$$

- Обновить значения весов по следующей формуле

$$\mathbf{W^{(new)}} = \mathbf{W^{(old)}} + \mathbf{D_i}.$$

Шаг 3 Повторять шаг 2 до тех пор, пока значение ошибки не перестанет уменьшаться, либо не будет достигнуто максимальное число эпох обучения.

Для оценки качества работы построенной сети используются различные метрики. Для задачи многоклассовой классификации наиболее распространённой является метрика accuracy, которая представляет собой долю правильных ответов нейронной сети:

$$Accuracy = \frac{\text{Количество правильных ответов}}{\text{Общее количество ответов}}.$$

Чтобы быть объективным, значение метрики должно вычисляться на тестовой выборке, то есть на тех данных, которые сеть «не видела» в процессе обучения.

Значение точности не даёт представления, на каких именно примерах нейронная сеть допустила ошибки. Для более подробного анализа ошибок можно построить так называемую *матрицу ошибок* (англ. confusion matrix). Она представляет собой матрицу, строки которой обозначают истинные метки, а столбцы — предсказанные нейронной сетью. На пересечении i-й строки и j-го столбца стоит число, обозначающее количество примеров i-го класса, которые были отнесены нейронной сетью к j-му классу.

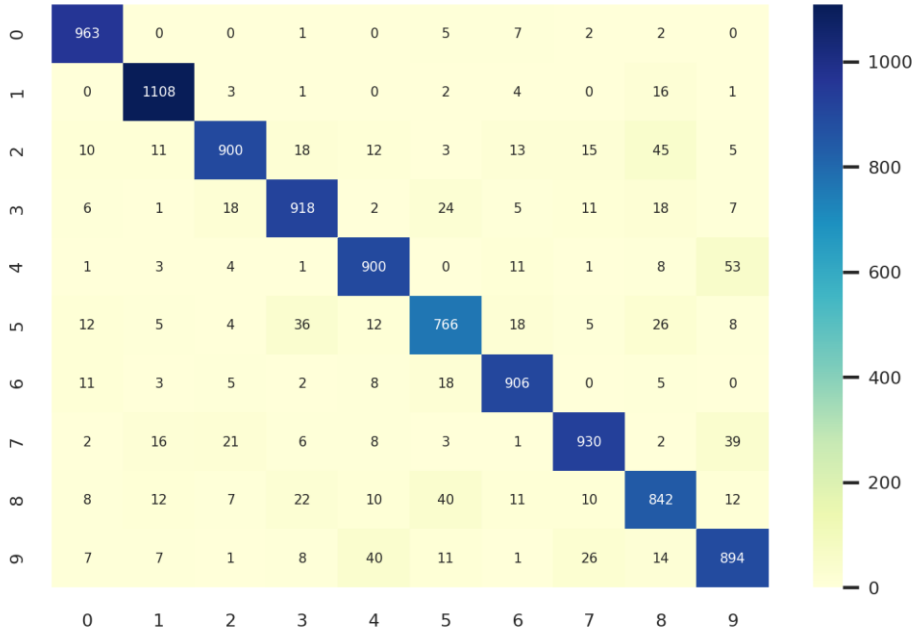

Рис. 9. Матрица ошибок для обученной нейронной сети.

Вопросы для самопроверки

1. Какие значения может принимать сигмоида?
2. Чему в модели логистической регрессии равняется взвешенная сумма факторов (признаков)?
3. В чём заключается метод максимального правдоподобия?
4. Как соотносятся между собой логистическая функция потерь и 1/0 функция потерь?
5. Какая схема кодирования меток классов используется при выводе логистической функции потерь?
6. Какая размерность должна быть у матрицы синаптических весов нейронной сети для решения задачи распознавания рукописных цифр из набора данных MNIST?

Список литературы

1. Воронцов К. В. Лекции по искусственным нейронным сетям, 2007. URL: http://www.ccas.ru/voron/download/NeuralNets.pdf (дата обращения: 17.09.2020)
2. НЕЙРОН. Обработка сигналов. Пластичность. Моделирование: Фундаментальное руководство / Ю. И. Александров, К. В. Анохин, Б. Н. Бездененежных и др. Тюмень: Издательство Тюменского государственного университета, 2008. 548 с.

3. Уоссермен Ф. Нейрокомпьютерная техника : Теория и практика. М.: Мир, 1992. 240 с.

4. Хайкин С. Нейронные сети: полный курс, 2-е издание.: Пер. с англ. М.: Издательский дом «Вильямс», 2006. 1104 с.