

Искусственные нейронные сети

Здравствуйте, уважаемые слушатели! Тема нашей лекции – Искусственные нейронные сети.

План лекции:

- Алгоритм обратного распространения ошибки
- Пошаговый пример расчета алгоритма обратного распространения ошибки
- Активационные функции

1. Алгоритм обратного распространения ошибки

Суть алгоритма ВРЕ заключается в следующем. Для тренировочного набора примеров

$$\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$$

устанавливаем выход первого слоя нейронов:

$$a^{[0]} = x^{(i)}$$

Шаг 1. Выполняем этап прямого распространения сигнала по слоям сети, то есть вычисляем сигнал на выходе сети, выполняя расчет для каждого нейрона в каждом слое, как показано в выражениях 1.4, 1.5. Результаты в виде выходных значений нейронов сети $a^{[0]}, a^{[1]}, \dots, a^{[L]}$ сохраняем в промежуточном хранилище (кэш).

Шаг 2. Используя полученный результат на выходе сети $a^{[L]} = h_w^{(i)}(x)$ и необходимое для данного примера выходное значение $y^{(i)}$, рассчитываем ошибку выходного слоя:

$$dz^{[L]} = y^{(i)} - a^{[L]},$$

где L – номер выходного слоя нейронной сети.

Шаг 3. «Возвращаем» ошибку, распространяя ее обратно по сети с учетом значения производной:

$$dz^{[L-1]} = w^{[L]} dz^{[L]} \cdot g'(a^{[L-1]}),$$

где знак $*$ – символ поэлементного умножения; g' – производная.

Производная сигмоидальной активационной функции:

$$g'(z^{[L-1]}) = a^{[L-1]} \cdot (1 - a^{[L-1]}).$$

Для любого скрытого слоя сети:

$$dz^{[i]} = w^{[i+1]} dz^{[i+1]} \cdot g'(a^{[i]}).$$

В случае сигмоидальной активационной функции:

$$dz^{[i]} = w^{[i+1]} dz^{[i+1]} \cdot a^{[i]} \cdot (1 - a^{[i]}).$$

Рассчитанное значение градиентов ошибки $dz^{[1]}, dz^{[2]}, \dots, dz^{[L]}$ также сохраняем в кэше.

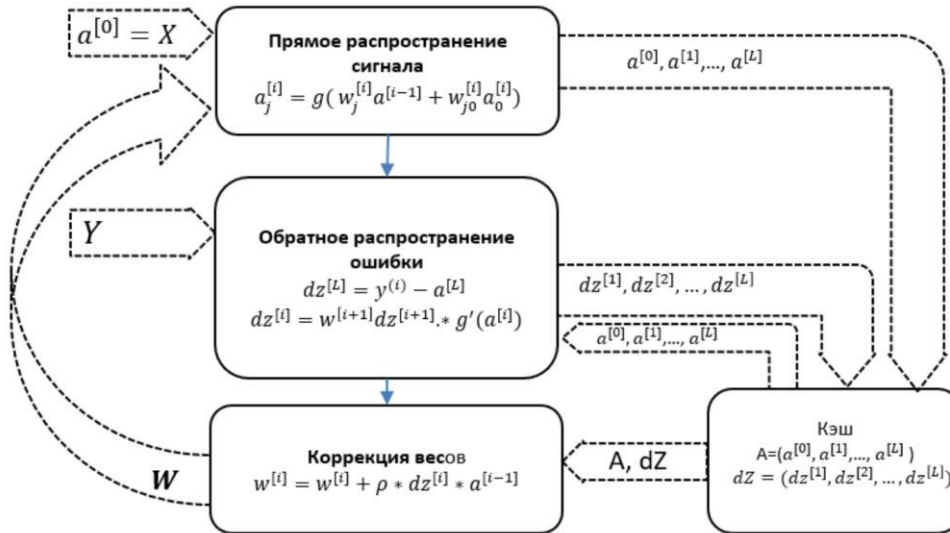
Шаг 4. Модифицируем веса сети с учетом значения ошибки для всех слоев $i \in L$:

$$w^{[i]} = w^{[i]} + \rho * dz^{[i]} * a^{[i-1]}, \quad (\text{Eq. 2.18})$$

где i – номер слоя сети; ρ – параметр обучения (learning rate) ($0 < \rho < 1$); $\Theta^{(i)}$ – матрица весов слоя i ; $dz^{(i)}$ – рассчитанное значение ошибки i -го слоя (точнее говоря, градиент ошибки).

Получив измененные значения весов, повторяем шаги 1–4 до достижения некоторого минимального значения ошибки либо заданное количество раз.

Процесс обучения искусственной нейронной сети можно представить в виде следующей схемы (рисунок 2.10):



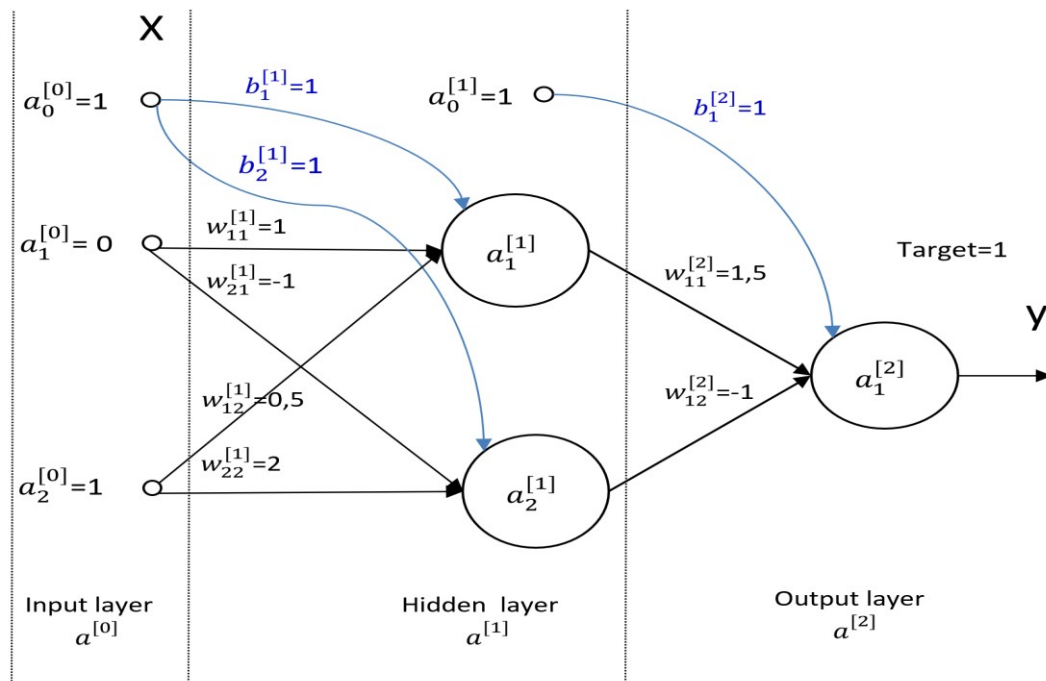
Итеративный процесс обучения искусственной нейронной сети

Рассмотрим пошаговый пример расчета прямого распространения сигнала, обратного распространения ошибки и коррекции весов.

2. Пошаговый пример расчета алгоритма обратного распространения ошибки

В этом примере веса нейронной сети будем обозначать символом w , смещения b . Номер слоя, как и ранее, указываем верхним индексом в квадратных скобках для того, чтобы не путать с индексом обучающего примера, номер нейрона в слое – нижним индексом. Выход нейрона по-прежнему обозначаем символом a .

Входной слой с его входами x для единообразия последующих матричных операций обозначаем как нулевой слой – $a^{[0]}$. В нашем примере $x_1 = 0$, $x_2 = 1$, тогда $a_1^{[0]} = x_1 = 0$ и $a_2^{[0]} = x_2 = 1$. Смещение (bias) во всех слоях $a_0^{[i]} = 1$.



Пример нейронной сети с одним скрытым слоем

На вход сети, таким образом, подается вектор $[1,0,1]$, а на выходе сети необходимо получить $y=1$.

Шаг 1. Прямое прохождение сигнала.

Рассмотрим прямое прохождение сигнала от входа к выходу:

$$z_1^{[1]} = b_1^{[1]} * a_0^{[0]} + w_{11}^{[1]} * a_1^{[0]} + w_{12}^{[1]} * a_2^{[0]} = 1*1 + 1*0 + 0.5*1 = 1.5$$

$$z_2^{[1]} = b_2^{[1]} * a_0^{[0]} + w_{21}^{[1]} * a_1^{[0]} + w_{22}^{[1]} * a_2^{[0]} = 1*1 + (-1)*0 + 2*1 = 3$$

$$a_1^{[1]} = g(z_1^{[1]}) = g(1.5) = 0.81757$$

$$a_2^{[1]} = g(z_2^{[1]}) = g(3) = 0.95257$$

Выход нейронной сети:

$$a_1^{[2]} = g(b_1^{[2]} * a_0^{[1]} + w_{11}^{[2]} * a_1^{[1]} + w_{12}^{[2]} * a_2^{[1]}) =$$

$$a_1^{[2]} = g(1 * 1 + 1.5 * 0.81751 + (-1) * 0.95257) = 0.78139$$

Шаг 2. Расчет ошибки выходного слоя.

Сеть должна давать значение $y^{(1)} = 1$, однако получена величина 0.78139. Ошибка, с которой сеть «предсказывает» наш единственный пример, равна разнице между ожидаемым значением и полученным результатом.

$$dz^{[L]} = y^{(i)} - a^{[L]}$$

$$dz_1^{[2]} = y^{(1)} - a_1^{[2]} = 1 - 0.78139 = 0.21861$$

Шаг 3. Обратное распространение ошибки.

Полученную ошибку нужно «распространить обратно» для того, чтобы скорректировать веса сети. Для этого рассчитаем градиенты ошибок нейронов скрытого слоя, используя выражение

$$dz^{[L-1]} = w^{[L-1]} dz^{[L]} * g'(a^{[L-1]}) = w^{[L-1]} dz^{[L]} * a^{[L-1]} * (1 - a^{[L-1]}).$$

Получим

$$dz_1^{[1]} = dz_1^{[2]} * u_{11}^{[2]} * a_1^{[1]} * (1 - a_1^{[1]}) = 0.048907262419244965$$

$$dz_2^{[1]} = dz_2^{[2]} * u_{12}^{[2]} * a_2^{[1]} * (1 - a_2^{[1]}) = -0.009876050115013725$$

Теперь у нас все готово для того, чтобы, используя градиенты ошибок, пересчитать веса нейронной сети.

Шаг 4. Коррекция весов нейронной сети.

Установим для нашего учебного примера большой коэффициент обучения (learning rate) $ro = 0.5$. Отметим, что в реальных случаях ro редко превышает 0.1. Здесь мы использовали относительно большое значение, чтобы увидеть значимые изменения весов уже на первой итерации.

Используем выражение (Eq. 2.18) для расчета измененных весов сети:

$$w_{11}^{[2]} = w_{11}^{[2]} + ro * dz_1^{[2]} * a_1^{[1]} = 1.5 + 0.5 * 0.21861 * 0.81757 = 1.58936$$

$$w_{12}^{[2]} = w_{12}^{[2]} + ro * dz_1^{[2]} * a_2^{[1]} = -1 + 0.5 * 0.21861 * 0.95257 = 0.89588$$

$$b_1^{[2]} = b_1^{[2]} + ro * dz_1^{[2]} * a_0^{[1]} = 1 + 0.5 * 0.21861 * 1 = 1.10930$$

для скрытого слоя:

$$w_{11}^{[1]} = w_{11}^{[1]} + ro * dz_1^{[1]} * a_1^{[0]} = 1 + 0.5 * 0.04891 * 0 = 1$$

$$w_{12}^{[1]} = w_{12}^{[1]} + ro * dz_1^{[1]} * a_2^{[0]} = 0.5 + 0.5 * 0.04891 * 1 = 0.524453$$

$$w_{21}^{[1]} = w_{21}^{[1]} + ro * dz_2^{[1]} * a_1^{[0]} = -1$$

$$w_{22}^{[1]} = w_{22}^{[1]} + ro * dz_2^{[1]} * a_2^{[0]} = 1.99153$$

$$b_1^{[1]} = b_1^{[1]} + ro * dz_1^{[1]} = 1.02445$$

$$b_2^{[1]} = b_2^{[1]} + ro * dz_2^{[1]} = 0.99506$$

Используя скорректированные значения весов, повторим расчет прямого прохода сигнала и получим значение ошибки выходного слоя:

$$dz_1^{[2]} = 0.17262$$

Видно, что ошибка стала значительно меньше.

После третьей итерации $dz_1^{[2]} = 0.14184$

Пример, приведенный выше, является иллюстрацией прямого и обратного хода алгоритма так, что каждый обучающий пример и каждый синаптический коэффициент рассчитываются по отдельности. На практике этапы алгоритма для сети из L-слоев реализуются в матричном виде следующим образом:

$$Z^{[1]} = b^{[1]} + W^{[1]} * X$$

$$A^{[1]} = g(Z^{[1]})$$

$$Z^{[2]} = b^{[2]} + W^{[2]} * A^{[1]}$$

$$A^{[2]} = g(Z^{[2]})$$

...

$$Z^{[l]} = b^{[l]} + W^{[l]} * A^{[l-1]}$$

$$A^{[l]} = g(Z^{[l]}),$$

где $W^{[i]}$ – матрица весов i -го слоя нейронной сети; X – матрица обучающих примеров размерностью $n \times m$ (n – число параметров, m – количество обучающих примеров).

Расчет алгоритма градиентного спуска для нейронной сети в матричном виде:

$$dZ^{[L]} = Y^{(i)} - A^{[L]}$$

$$dW^{[l]} = \frac{1}{m} dZ^{[L]} A^{[l-1]T}$$

$$db^{[L]} = dZ^{[L]}$$

...

$$dZ^{[i]} = W^{[i+1]} dZ^{[i+1]} * g'(A^{[i]})$$

$$db^{[i]} = dZ^{[i]}$$

Выражения, приведенные выше, говорят о том, что на вход сети подаются все обучающие примеры «одновременно» и значения градиентов ошибки рассчитываются сразу для всех примеров. Этот процесс составляет одну эпоху обучения. **Batch Gradient Descent** – это процесс обучения, когда все обучающие примеры используются одновременно. Несколько десятков или сотен эпох обычно достаточно для достижения оптимальных значений весов матриц $W^{[i]}$.

Однако, когда количество примеров очень велико, примеры разбиваются на группы, которые можно поместить в оперативную память компьютера, и эпоха обучения включает последовательную подачу этих групп. При этом возможны два подхода [1]:

Stochastic Batch Gradient Descent – когда группа включает лишь один пример, выбираемый случайно из множества обучающих примеров.

Mini Batch Gradient Descent – когда группа включает некоторое количество примеров.

3. Активационные функции

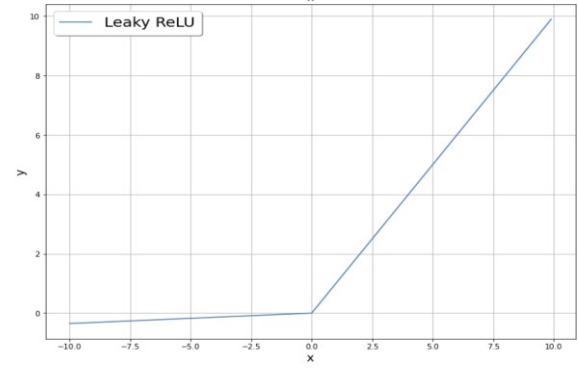
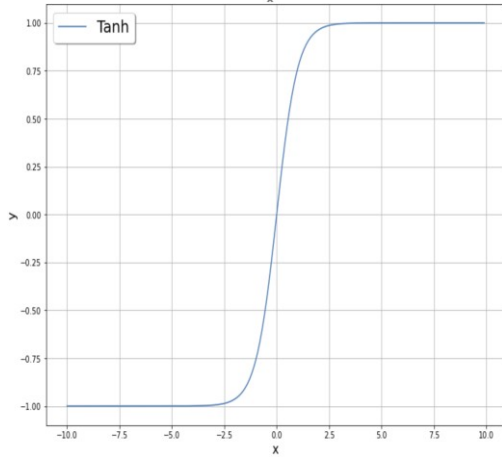
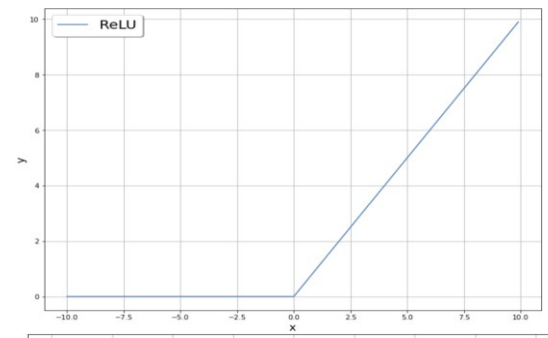
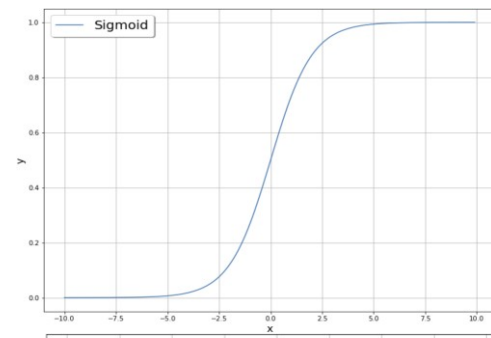
Нелинейная активационная функция играет фундаментальную роль в процессе обучения нейронной сети. Именно ее применение позволяет нейронной сети обучаться сложным закономерностям, содержащимся в исходных данных. Кроме уже упомянутой сигмоидальной функции часто используются и несколько других активационных функций (рисунок 2.12), описываемых уравнениями

$$\text{Sigmoid: } y = \frac{1}{1 + e^{-x}}$$

$$\text{Tanh: } y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\text{ReLU: } y = \max(0, x)$$

$$\text{Leaky ReLU: } y = \max(0.01x, x)$$



Активационные функции, применяемые в нейронных сетях