# Введение в нейронные сети

Ученых с давних пор интересовало, каким образом основаны принципы принятия решения человеком, и каким образом можно смоделировать данный процесс для создания так называемого искусственного интеллекта. Именно на основе наблюдений над нервными клетками человеческого мозга были разработаны искусственные нейронные сети, которые сейчас активно применяются для решения задач машинного обучения.

Источники приведенных примеров:

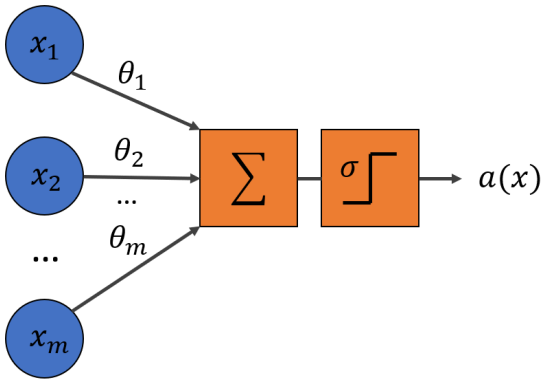
1. К.В. Воронцов. Введение в машинное обучение
2. С.Н. Николенко. Глубокое обучение

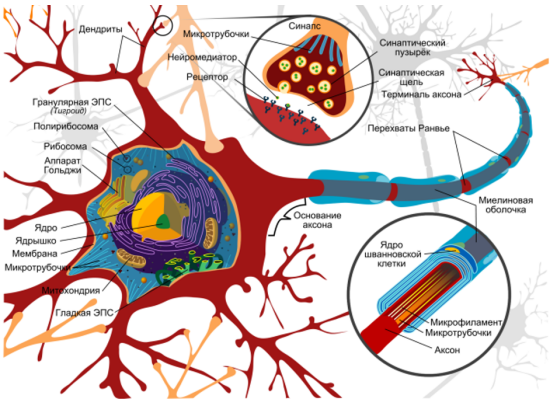
---

## 1. Линейная модель нейрона

Первой важной работой в области искусственных сетей стала предложенная американскими учеными МакКалоком и Питтсом в 1943 году линейная модель нейрона. Данная идея очень похожа на линейный классификатор. Нервную клетку можно рассматривать как устройство, которое на входе имеет множество дендритов, которые при накоплении вокруг них отрицательных ионов пропускают данный заряд внутрь клетки. Если суммарный заряд, накопившийся в клетке, превышает некоторый порог активации, клетка генерирует электрический импульс, который далее распространяется по длинному отростку, называемому аксоном и далее распространяется на следующие нервные клетки. Там происходит очередное накопление отрицательных ионов, они переходят в следующую клетку и т. д.

Поскольку заряды – это аддитивные величины, тогда если в качестве признаков рассматривать количество зарядов на дендритах, а в качестве весов этих признаков – способность дендритов пропускать отрицательные ионы внутрь клетки, то получается, что суммарный заряд накопленный есть не что иное, как линейная комбинация зарядов, которые были на входе, то есть, по сути, это признаки в линейном классификаторе, и они умножаются на веса. Потом вся эта линейная комбинация сравнивается с некоторым порогом активации. Принято рассматривать функцию активации, которая преобразует суммарный накопленный заряд в какой-то выход на аксоне клетки.



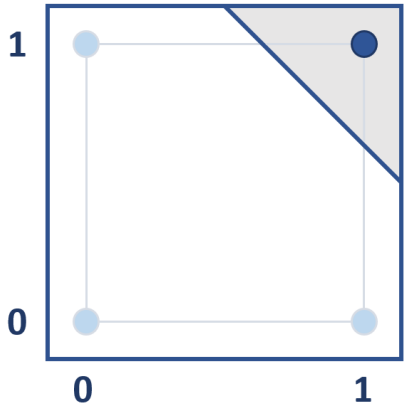




Слева на изображении представлена математическая модель нейрона, а справа – структура самого нейрона человеческого мозга. Таким образом, идея искусственных нейронных сетей является попыткой приблизить с помощью математических моделей процесс принятия решения человеческим мозгом.

## 1.1. Реализация логических операций

### 1.1.1. Конъюнкция

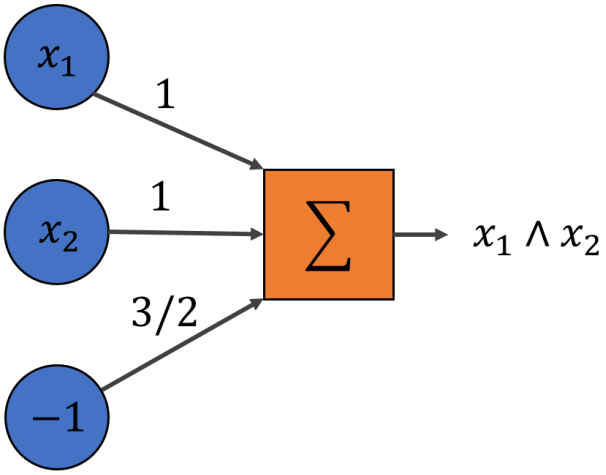
Рассмотрим логическую операцию И для некоторых бинарных переменных $x_1$ и $x_2$. Значение целевой переменной $y$ окажется равным 1 в случае, когда обе рассматриваемые переменные принимают значение 1.




Описать данное правило можно с помощью следующего выражения:

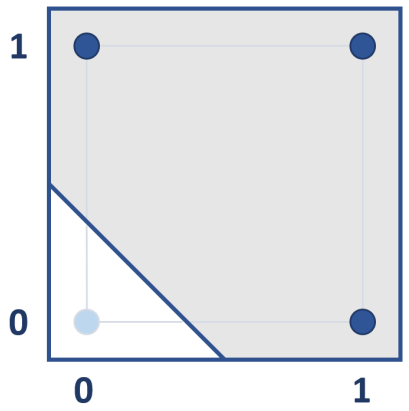
$$x_1 \wedge x_2 = \left[x_1 + x_2 - \frac{3}{2} > 0\right].$$

Указанное выражение можно представить в виде простой нейронной сети:




### 1.1.2. Дизъюнкция

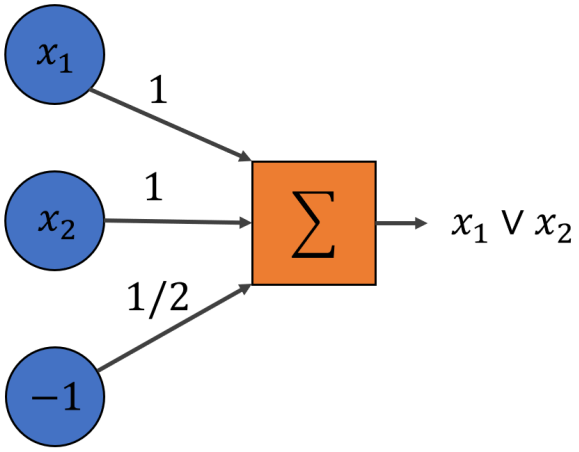
Значение целевой переменной $y$ для логической операции ИЛИ принимает значение 1 в случае, когда хотя бы одна из двух переменных $x_1$ и $x_2$ окажется равным 1.



Тогда дизъюнкцию можно представить в виде выражения:

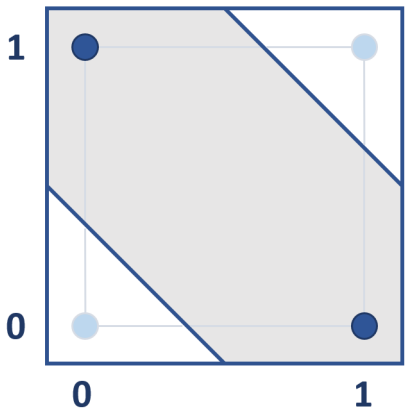
$$x_1 \vee x_2 = \left[x_1 + x_2 - \frac{1}{2} > 0\right],$$

а соответствующую нейронную сеть представить в следующем виде:




## 1.2. Логическая операция XOR

Исключающее ИЛИ принимает значение 1 в случае, когда ровна одна из переменных принимает значение 1.



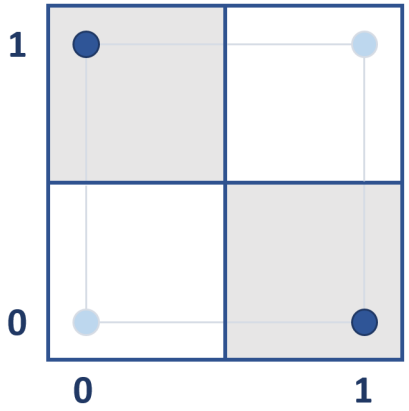

Данная функция может быть реализована путем добавления произведения $x_1 x_2$, тогда

$$x_1 \oplus x_2 = \left[x_1 + x_2 - 2\,x_1\,x_2 - \frac{1}{2} > 0\right].$$

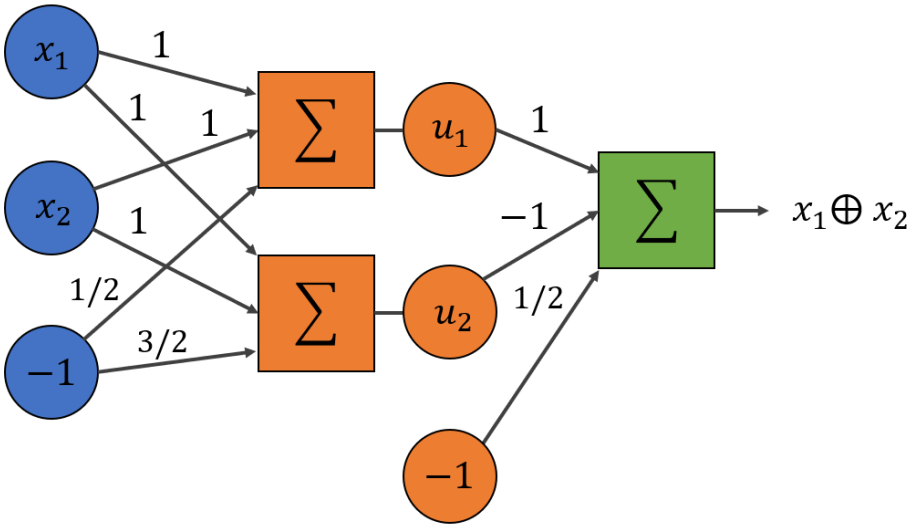
К поставленной задаче можно подойти и с другой стороны. Заметим, что данная логическая операция может быть описана через суперпозицию функций И и ИЛИ:

$$x_1 \oplus x_2 = \left[(x_1 \vee x_2) - (x_1 \wedge x_2) - \frac{1}{2} > 0\right].$$

Соответствующий вид решающей функции:




Такая суперпозиция может быть представлена в виде двухслойной нейронной сети:

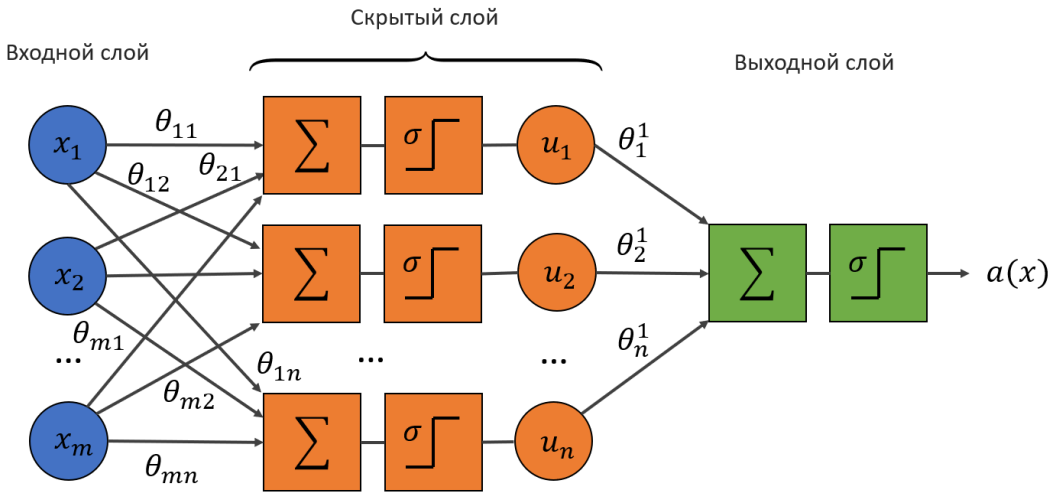



---

# 2. Многослойные нейронные сети

Глубокие нейронные сети представляют собой суперпозицию линейных моделей, что позволяет с высокой точностью аппроксимировать сложные нелинейные зависимости. Глубокие сети состоят из входного слоя, который является вектором признаков рассматриваемого объекта, набора скрытых слоев и выходного слоя. Переход между слоями нейронной сети происходит с помощью функции активации. Для бинарной классификации функцией активации могут выступать логистическая сигмоида или гиперболический тангенс. Функция активации представляет собой некоторую дифференцируемую функцию.

Пример двухслойной нейронной сети с $n$ нейронами на скрытом слое представлен на рисунке:




Обучение нейронной сети проводится аналогично линейным моделям с помощью градиентных методов. Рассмотрим алгоритм обучения глубоких нейронных сетей на примере.

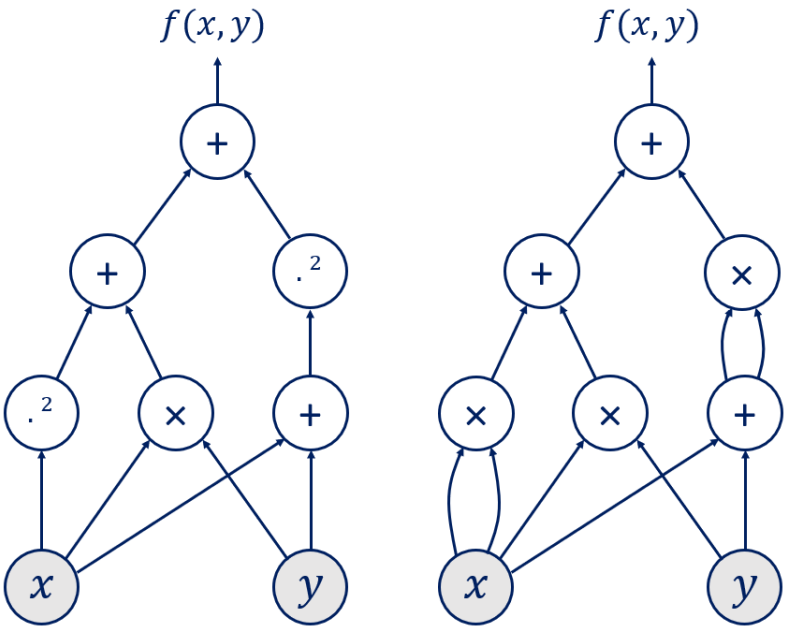
---

# 3. Обучение нейронных сетей

Чтобы воспользоваться градиентными методами обучения необходимо вычислить производные по каждому из параметров модели. В качестве примера рассмотрим функцию $f(x, y) = x^2 + xy + (x + y)^2$.

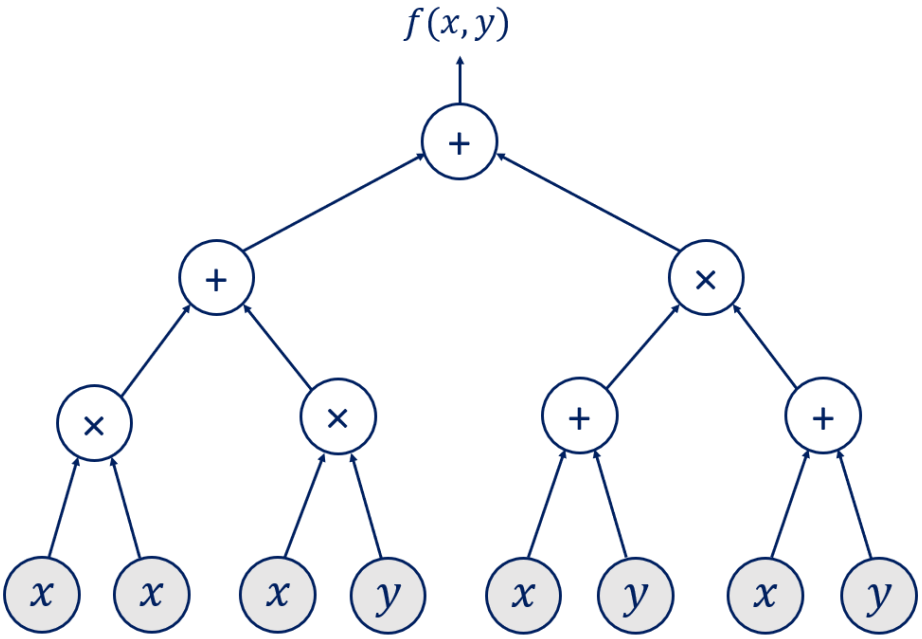
## 3.1. Граф вычислений

Можно заметить, что функция $f(x, y)$ представляет собой суперпозицию более простых функций. Представим ее в виде графа вычислений, который состоит из тех арифметических операций, которые мы допускаем. В случае с нейронными сетями допустимыми являются операции сложения, умножения, а также применение функции активации. В зависимости от выбранного набора операций граф вычислений функции $f(x, y)$ может быть представлен разными способами:










Для применения метода градиентного спуска необходимо вычислить производные по каждому из параметров функции. Поскольку производные сложных функций вычисляются по правилу: $(f(g(x)))' = f'(g)\,g(x)'$, то для вычисления производной функции $f(x, y) = x^2 + xy + (x+y)^2$ по параметрам $x$, $y$ необходимо вычислить производные в каждом из узлов графа вычислений. Возьмем первый граф вычислений данной функции и обозначим каждый из узлов буквами $a - f$.

Производная $\frac{\partial f}{\partial x}$ будет складываться из следующих частных производных:

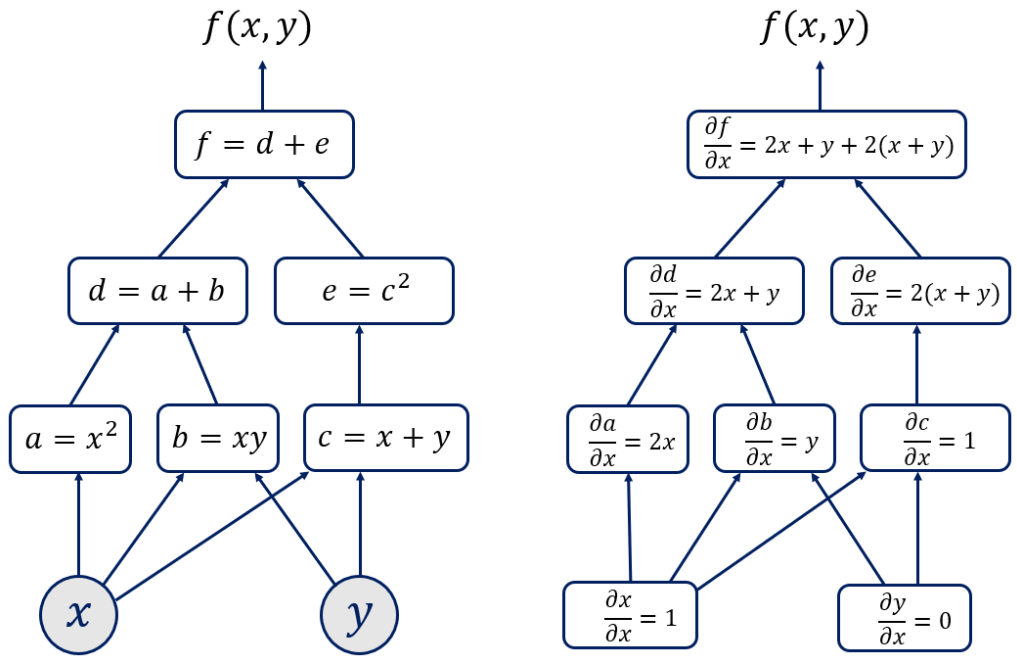
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial d}\frac{\partial d}{\partial x} + \frac{\partial f}{\partial e}\frac{\partial e}{\partial x}.$$

В свою очередь:

$$\frac{\partial d}{\partial x} = \frac{\partial d}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial d}{\partial b}\frac{\partial b}{\partial x},$$

$$\frac{\partial e}{\partial x} = \frac{\partial e}{\partial c}\frac{\partial c}{\partial x}.$$

Таким образом, чтобы знать производные по параметрам функции, необходимо знать производные в каждом из узлов графа вычислений. На рисунке представлен процесс вычисления частной производной по параметру $x$ на графе вычислений:



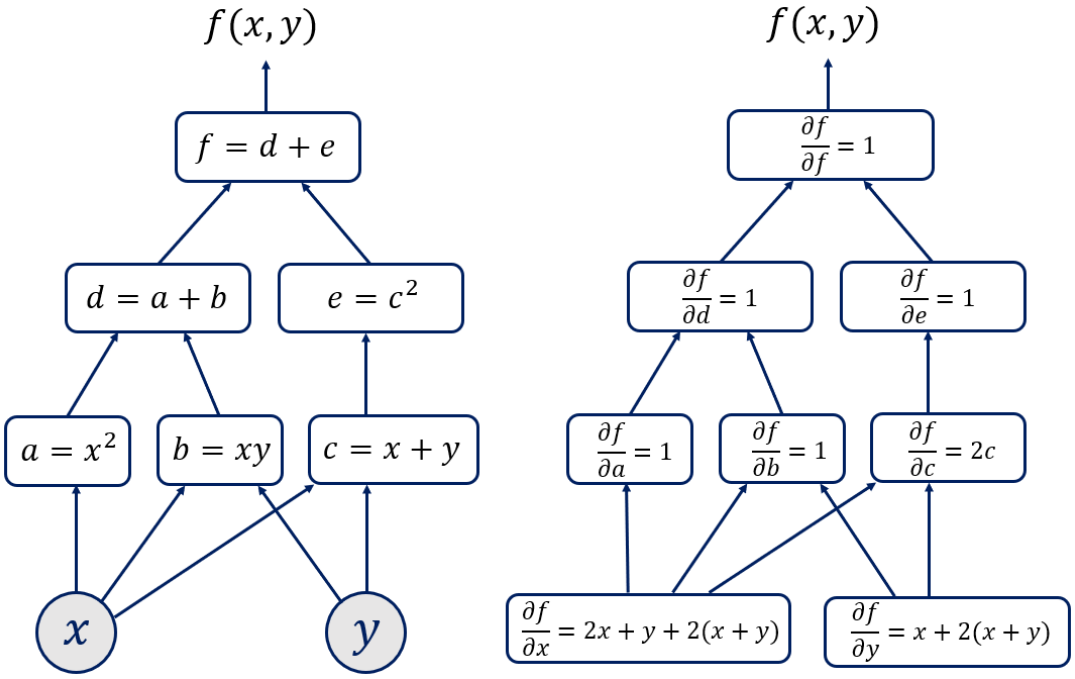

Такой подход позволяет вычислить частные производные по одному из параметров. Если иной подход: вычислять производные от последнего узла до первого. Тогда:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial f}\frac{\partial f}{\partial d}\left(\frac{\partial d}{\partial a}\frac{\partial a}{\partial x} + \frac{\partial d}{\partial b}\frac{\partial b}{\partial x}\right) + \frac{\partial f}{\partial f}\frac{\partial f}{\partial e}\frac{\partial e}{\partial c}\frac{\partial c}{\partial x} =$$

$$= 1 \times 1 \times (1 \times 2x + 1 \times y) + 1 \times 1 \times 2(x + y) \times 1 = \mathbf{2x + y + 2(x + y)},$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial f}\frac{\partial f}{\partial d}\frac{\partial d}{\partial b}\frac{\partial b}{\partial y} + \frac{\partial f}{\partial f}\frac{\partial f}{\partial e}\frac{\partial e}{\partial c}\frac{\partial c}{\partial y} =$$

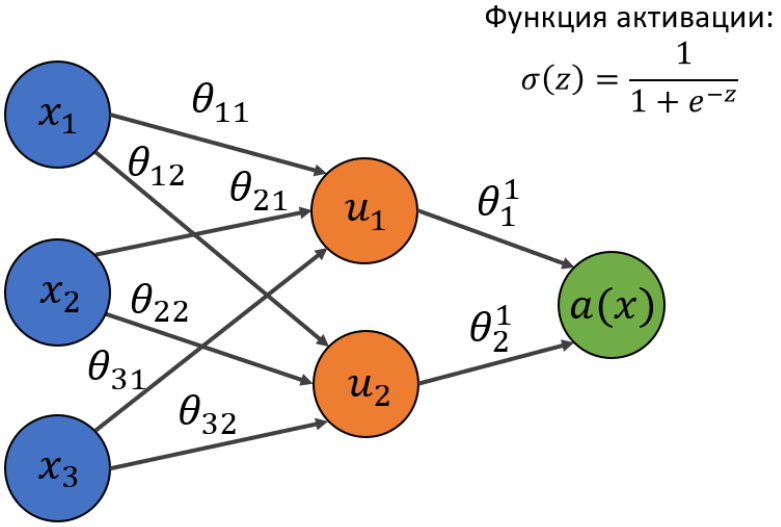
$$= 1 \times 1 \times 1 \times x + 1 \times 1 \times 2(x + y) \times 1 = \mathbf{x + 2(x + y)}.$$



Заметим, что частные производные по обоим параметрам имеют общие сомножители на верхних уровнях графа вычислений. Таким образом, нет необходимости вычислять отдельно производные по каждому из параметров. В этом состоит идея **метода обратного распространения ошибки**.

## 3.2. Метод обратного распространения ошибки

Рассмотрим метод обратного распространения ошибки на примере двухслойной нейронной сети.



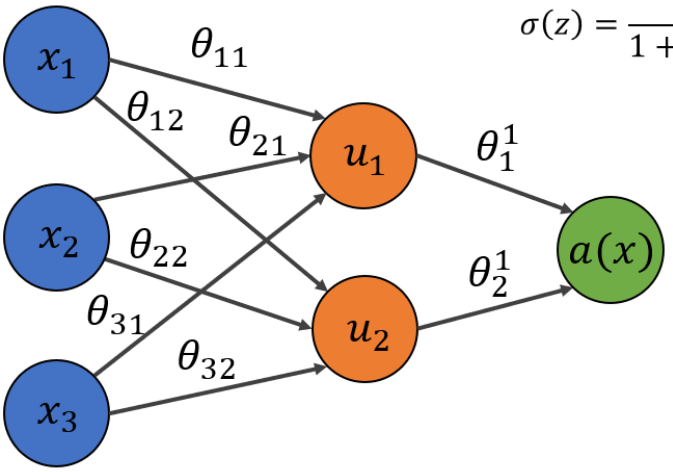

Данная архитектура решает задачу бинарной классификации (функция активации – логистическая сигмоида). Для простоты для обучения модели определим функцию потерь как:

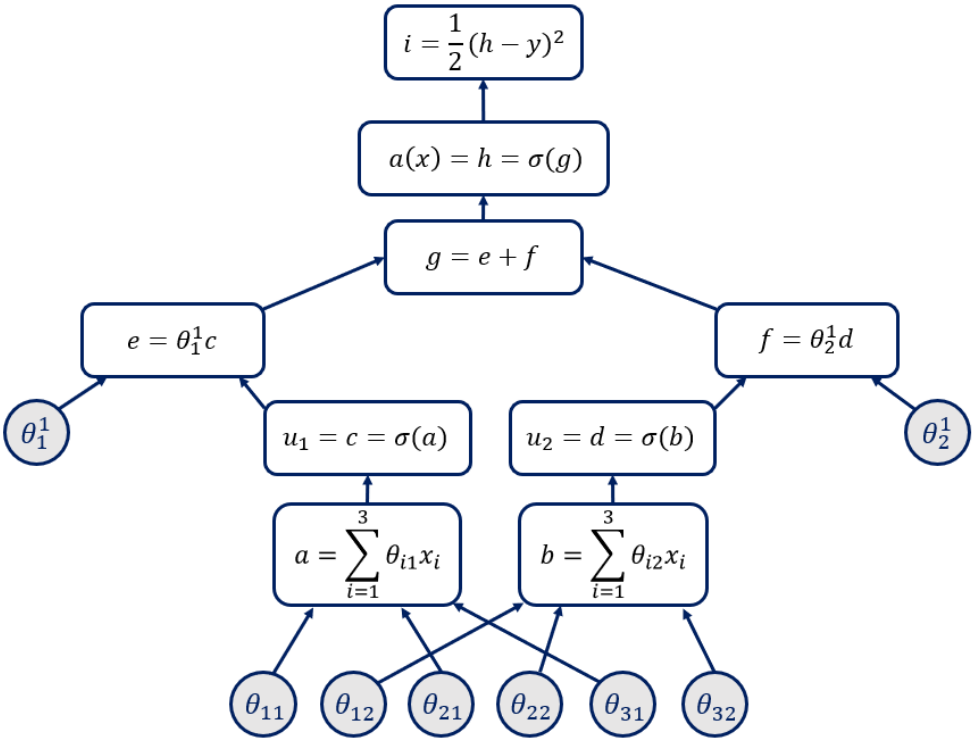
$$L(a(x),\, y) = \frac{1}{2}\,(a(x) - y)^2.$$

Тогда граф вычислений можно представить в следующем виде:

Функция активации:

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




$$L = \frac{1}{2}(a(x)-y)^2 = \frac{1}{2}\left(\sigma\left(\theta_1^1 \sigma\left(\sum_{i=1}^{3} \theta_{i,1} x_i\right) + \theta_2^1 \sigma\left(\sum_{i=1}^{3} \theta_{i,2} x_i\right)\right) - y\right)^2$$

**Упражнение**

Вычислить частные производные во всех узлах графа вычислений методом обратного распространения.

$$\frac{\partial i}{\partial \theta_1^1} = (a(x)-y)\, a(x)\,(1-a(x))\, u_1$$

$$\frac{\partial i}{\partial \theta_{11}} = (a(x)-y)\, a(x)\,(1-a(x))\, \theta_1^1\, u_1(1-u_1)\, x_1$$

Получив частные производные по всем параметрам модели, можно обучить данную нейронную сеть методом стохастического градиентного спуска. Возьмем один объект выборки:

```
x = {1, 1, 0}; y = 0;
θ = {{0.5, 0.2}, {0.1, -0.3}, {-0.1, 0.5}};
θ1 = {0.8, 0.2};
```

Методом прямого распространения получим ответ нейронной сети с текущими значениями весов $\theta$:

```
u = LogisticSigmoid[x.θ]
```

```
{0.645656, 0.475021}
```

```
a = LogisticSigmoid[u.θ1]
```

```
0.64829
```

Как видно из данного примера, нейронная сеть допускает ошибку на объекте. Значение целевой переменной $y = 0$, в то время как ответ модели $a(x) = 0.648$.

> **Упражнение**
>
> Выполнить шаг градиентного спуска для всех параметров модели, пользуясь формулами, полученными в предыдущем задании.

```
η = 0.01;
```

Обновление весов второго слоя:

```
delta = (a - y) a (1 - a);
```

```
θ1 - η delta u
```

```
{0.799046, 0.199298}
```

Обновление весов первого слоя:

```
θ[[1, 1]] - η (a - y) a (1 - a) θ1[[1]] u[1] (1 - u[[1]]) x[1] (* θ11 *)
```

```
0.499729
```

```
θ[[1, 2]] - η (a - y) a (1 - a) θ1[[2]] u[2] (1 - u[[2]]) x[1] (* θ12 *)
```

```
0.199926
```

```
θ[[2, 1]] - η (a - y) a (1 - a) θ1[[1]] u[1] (1 - u[[1]]) x[2] (* θ21 *)
```

```
0.0997295
```

```
θ[[2, 2]] - η (a - y) a (1 - a) θ1[[2]] u[2] (1 - u[[2]]) x[2] (* θ22 и т.д. *)
```

```
-0.300074
```

```
train = {
   {{1, 0, 0}, 0},
   {{0, 1, 0}, 0},
   {{0, 0, 1}, 1},
   {{1, 1, 0}, 0},
   {{0, 1, 1}, 1},
   {{1, 0, 1}, 0},
   {{1, 1, 1}, 0},
   {{0, 0, 0}, 1}
  };
```