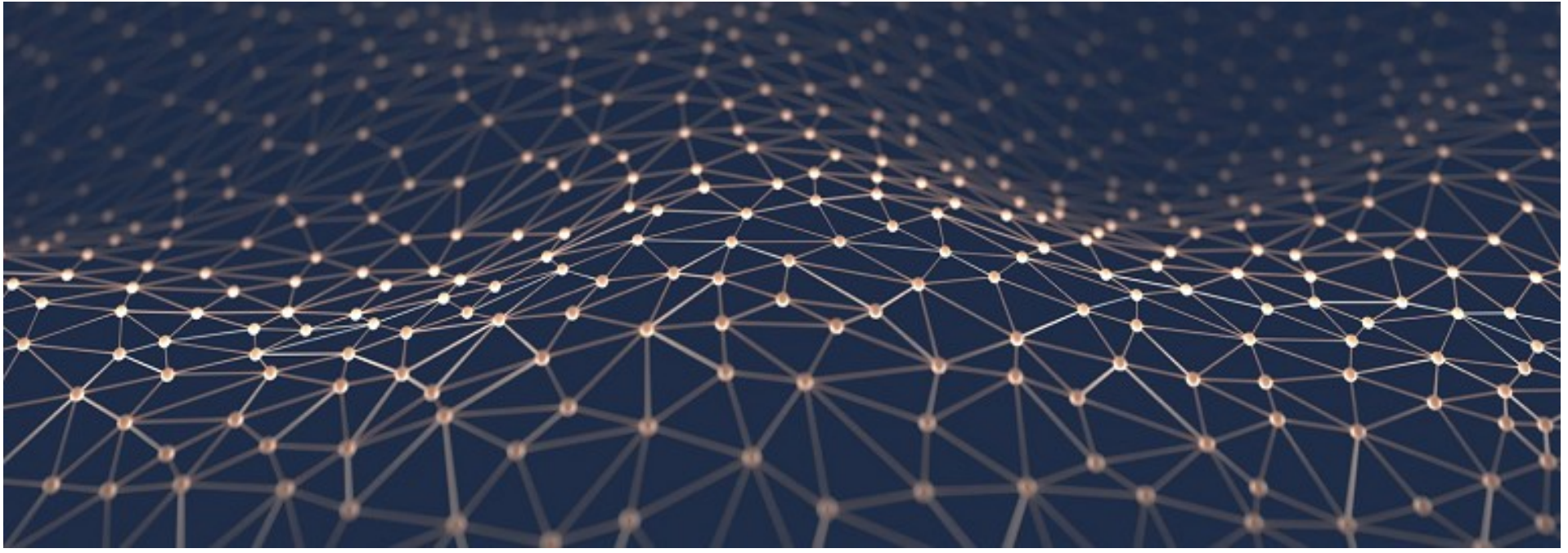
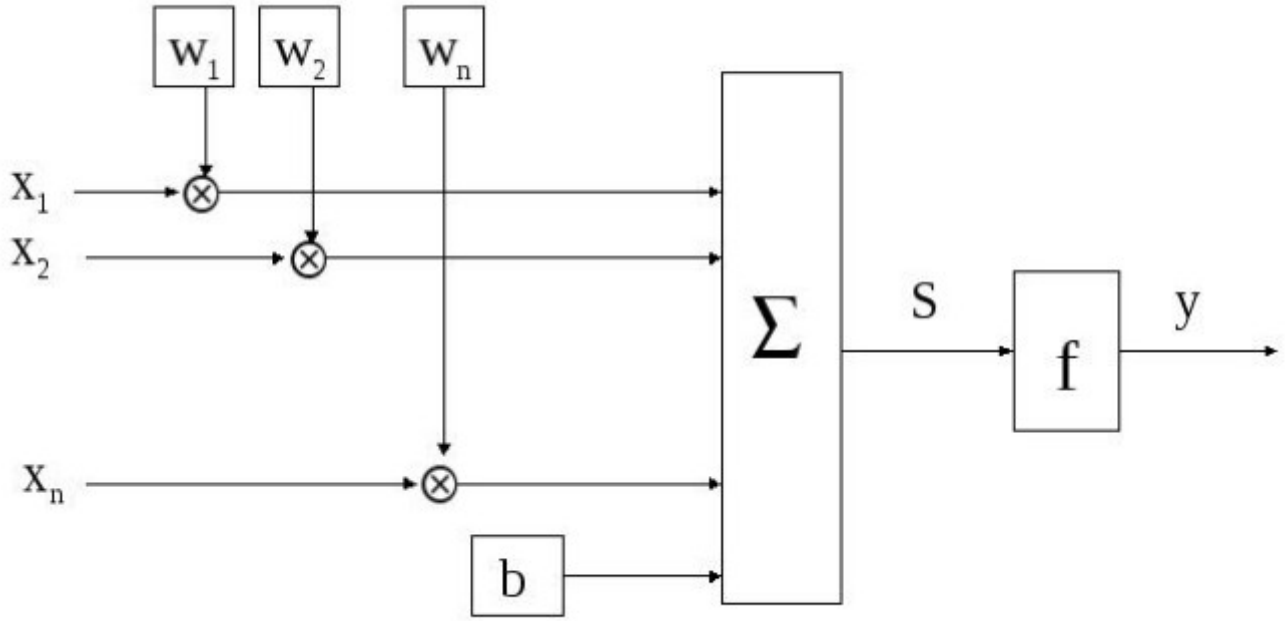
НЕЙРОННЫЕ СЕТИ



ИСКУССТВЕННЫЙ НЕЙРОН



МОДЕЛЬ НЕЙРОНА

$$S = \sum w_i x_i + b$$

$$y = f(S)$$

x_i, i=1,2…n – сигнал от другого нейрона

w_i, i=1,2…n – сила синаптической связи

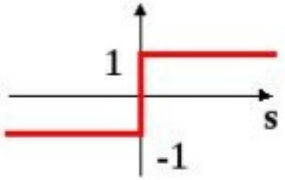
b – смещение

f – функция активации нейрона

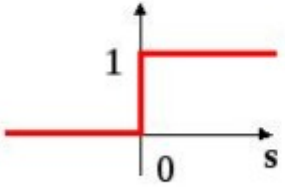
y – выходной сигнал нейрона

ФУНКЦИИ АКТИВАЦИИ

Пороговые

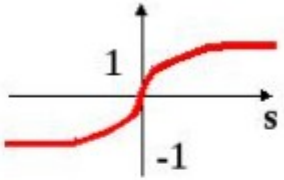


$$sign(s) = \begin{cases} 1, & s>0 \\ -1, & s<0 \end{cases}$$

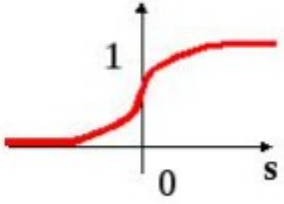


Функция Хэвисайда $\theta(s) = \begin{cases} 1, & s>0 \\ 0, & s<0 \end{cases}$

Сигмоидные

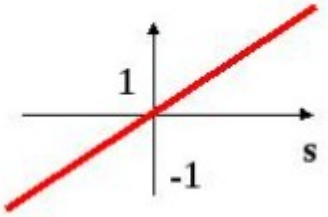


$$th(s) = \frac{e^{as}-e^{-as}}{e^{as}+e^{-as}}$$



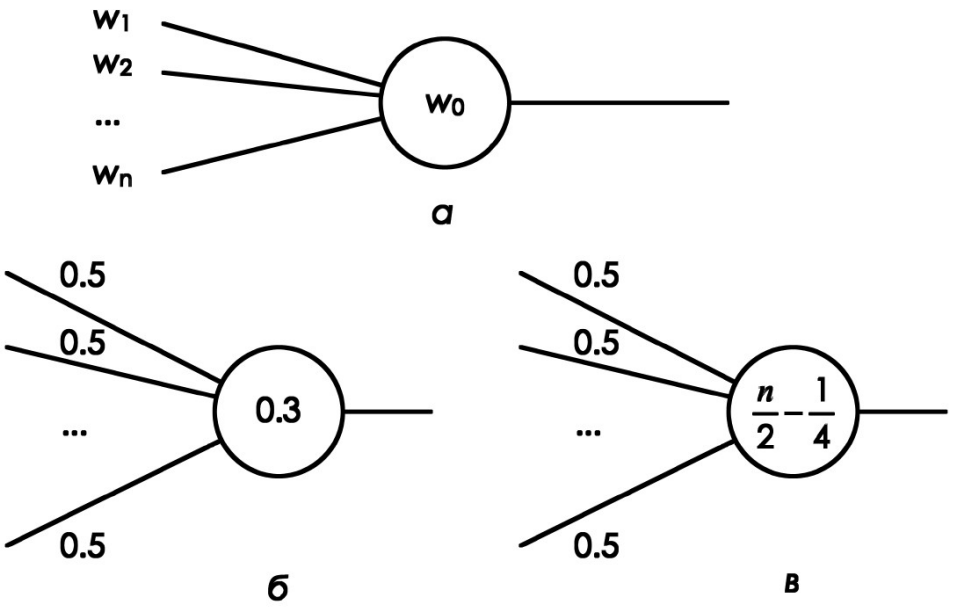
$$P(s) = \frac{1}{1+e^{-as}}$$

Линейная



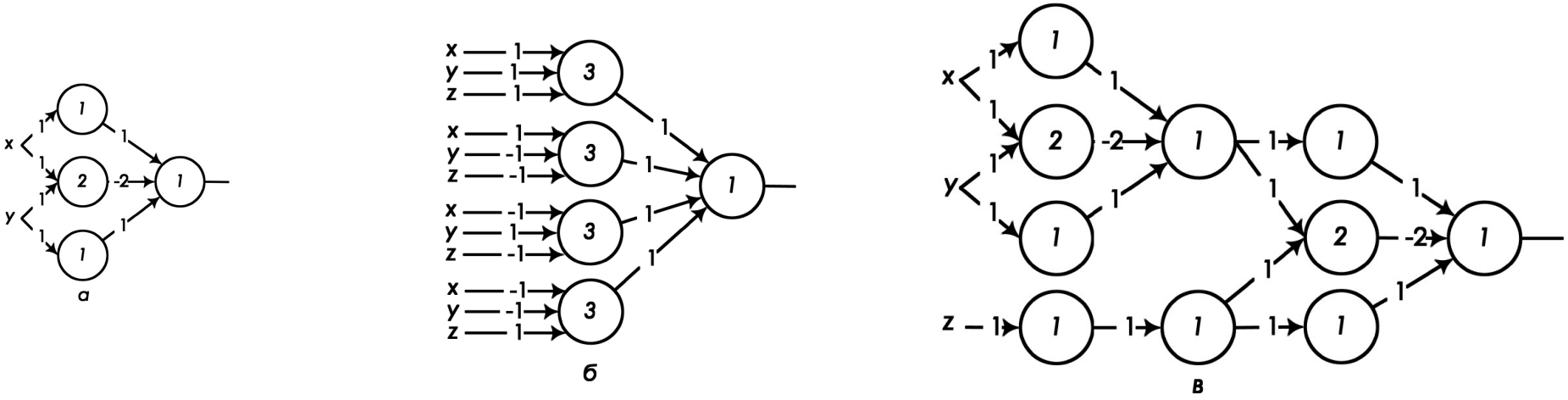
Линейная с насыщением

ПРИМЕРЫ ПЕРЦЕПТРОНОВ



а) общий вид перцептрона; б) перцептрон, реализующий дизъюнкцию; в) перцептрон, реализующий конъюнкцию

СЕТИ, РЕАЛИЗУЮЩИЕ XOR



а) двух переменных; б) сеть глубины 2, реализующая XOR 3-х переменных; в) сеть глубины 4, реализующая XOR 3-х переменных

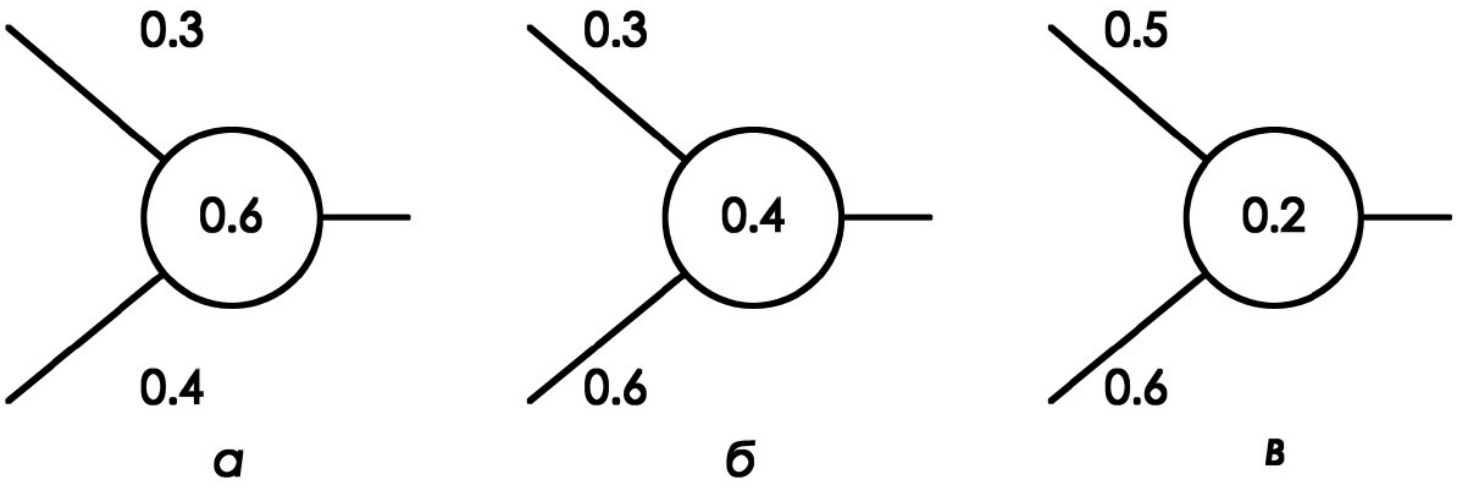
ОБУЧЕНИЕ ПЕРЦЕПТРОНА

Правило изменения веса на шаге обучения

$$w_i := w_i + \eta(t - o)x_i,$$

где t — значение целевой функции, o — выход перцептрона, $\eta > 0$ — небольшая константа (обычно 0,05–0,2), которая задаёт скорость обучения.

ОБУЧЕНИЕ ПЕРЦЕПТРОНА



а) перед началом обучения; б) после первого шага обучения; в) перцептрон, реализующий дизъюнкцию

ОБУЧЕНИЕ ПЕРЦЕПТРОНА

Набор тестовых данных

$x_1 = 0, \quad x_2 = 1 \quad \Rightarrow \quad t = 1,$ Выход o=-1

$x_1 = 1, \quad x_2 = 0 \quad \Rightarrow \quad t = 1.$

Скорректированный набор данных для первого теста

$$w_0 := w_0 + \eta(t-0)x_0 = -0{,}6 + 0{,}1 \cdot (1-(-1)) \cdot 1 = -0{,}4,$$
$$w_1 := w_1 + \eta(t-0)x_1 = 0{,}3 + 0{,}1 \cdot (1-(-1)) \cdot 0 = 0{,}3,$$
$$w_2 := w_2 + \eta(t-0)x_2 = 0{,}4 + 0{,}1 \cdot (1-(-1)) \cdot 1 = 0{,}6.$$

Скорректированный набор данных для второго теста

$$w_0 := w_0 + \eta(t-0)x_0 = -0{,}4 + 0{,}1 \cdot (1-(-1)) \cdot 1 = -0{,}2,$$
$$w_1 := w_1 + \eta(t-0)x_1 = 0{,}3 + 0{,}1 \cdot (1-(-1)) \cdot 1 = 0{,}5,$$
$$w_2 := w_2 + \eta(t-0)x_2 = 0{,}6 + 0{,}1 \cdot (1-(-1)) \cdot 0 = 0{,}6.$$

АЛГОРИТМ ОБУЧЕНИЯ

PerceptronTraining$(\eta, \{x_i^j, t^j\}_{i=1,j=1}^{n,m})$

1. **Инициализировать** $\{w_i\}_{i=0}^{n}$ **маленькими случайными значениями.**
2. `WeightChanged` := `true`.
3. **Пока** `WeightChanged` = `true`:
   а) `WeightChanged` := `false`.
   б) **Для всех** j **от** 1 **до** m:
      (i) **Вычислить**

$$o^j := \begin{cases} 1, \text{ если } w_0 + w_1 x_1^j + \ldots + w_n x_n^j > 0, \\ -1 \text{ в противном случае.} \end{cases}$$

      (ii) **Если** $o^j \neq t^j$:
         (A) `WeightChanged` = `true`.
         (B) **Для каждого** i **от** 0 **до** n **изменить значение** w_i **по правилу**

$$w_i := w_i + \eta(t^j - o^j)x_i^j.$$

4. **Выдать значения** $w_0, w_1, \ldots, w_n$.

ЛИСТИНГ. ОБУЧЕНИЕ ПЕРЦЕПТРОНА НА PYTHON

```
Def PerceptronTraining(eta, x):
    import random
    w=[]
    for i in range(len(x[0])):
        w.append((random.randrange(-5,5))/50.0)

    WeightsChanged=True
    while (WeightsChanged==True):
        WeightsChanged=False
        for xj in x:
            t,o,curx=xj[0],0,[1]+xj[1:len(xj)]
            for I in xrange(len(w)): o+=w[i]*curx[i]
            if o>0: o=1
            else o=-1
            if (o==t): continue
            WeightsChanged=True
            for i in xrange(len(w)): w[i]+=eta*(t-o)*curx[i]
    return w
```

МЕТОД ГРАДИЕНТНОГО СПУСКА

GradientDescent$(\eta, \{x_i^j, t^j\}_{i=1,j=1}^{n,m})$

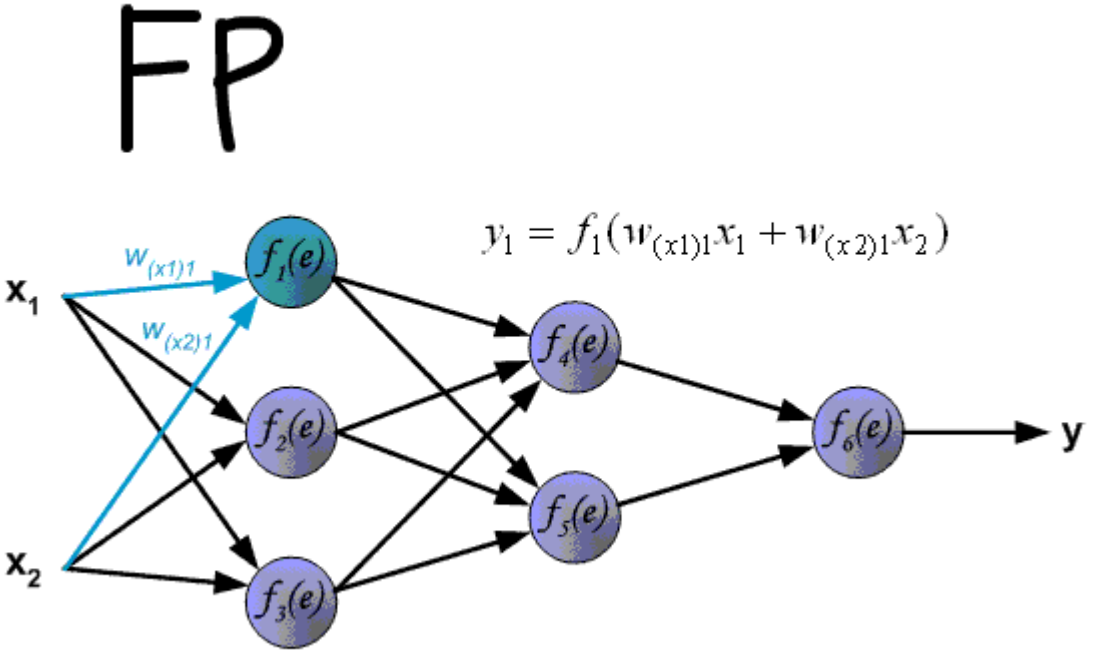
1. **Инициализировать** $\{w_i\}_{i=0}^{n}$ **маленькими случайными значениями.**
2. **Повторить** NUMBER_OF_STEPS **раз:**
   а) **Для всех** i **от** 1 **до** n $\Delta w_i := 0$.
   б) **Для всех** j **от** 1 **до** m:
      (i) **Для всех** i **от** 1 **до** n

$$\Delta w_i := \Delta w_i + \eta \left(t^j - \sum_{0}^{n} w_i x_i^j \right) x_i^j.$$

   в) $w_i := w_i + \Delta w_i$.
3. **Выдать значения** $w_0, w_1, \ldots, w_n$.

МЕТОД ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

FP



МЕТОД ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

- Шаг 1. Инициализация синаптических весов и смещений
- Шаг 2. Представление из обучающей выборки входного вектора $X_q = (x1, x2, \ldots, x_N)_q$ и соответствующего ему выходного вектора $D_q = (d1, d2, \ldots, d_M)_q$
- Шаг 3. Прямой проход

$$y_i^{(k)} = f_\sigma\left(j\sum_{j=0}^{H_{k-1}} w_{ij}^{(k)} y_j^{(k-1)} \right)$$

- Шаг 4. Обратный проход

$$w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) + \Delta w_{ij}^{(k)}(t+1) \qquad \Delta w_{ij}^{(k)}(t+1) = \eta \delta_i^{(k)} y_j^{(k-1)}$$

$$\delta_i^{(K)} = (d_i - y_i) y_i (1 - y_i) \qquad \delta_i^{(k)} = y_i^{(k)}(1 - y_i^{(k)}) \sum_{l=1}^{H_{k+1}} \delta_l^{(k+1)} w_{li}^{(k+1)}$$

- Шаг 5. Повторение шагов 2–4 необходимое число раз

ПРИМЕНЕНИЕ НЕЙРОСЕТЕЙ

- если есть математическая модель какого-то процесса, то изучая влияние входных параметров на выходные, можно решить задачу оптимизации моделируемого процесса
- если математическая модель является нестационарной, то её можно использовать для решения задач прогнозирования
- если математическая модель работает в реальном режиме времени, то результаты математического моделирования могут быть оперативно переданы оператору, управляющему объектом, или могут быть непосредственно введены в приборы, что позволяет решать задачи управления моделируемым объектом или процессом
- нейронные сети могут решать задачи распознавания и классификации образов, причем под образами понимаются зрительные изображения, символы, тексты, запахи, звуки, шумы

СПАСИБО ЗА ВНИМАНИЕ