

# **Лекция 7**

# **Распознавание образов**

**Курс «Компьютерное зрение»**

# Распознавание объектов

## Образы и классы

Образ:

- Упорядоченная совокупность дескрипторов

Класс:

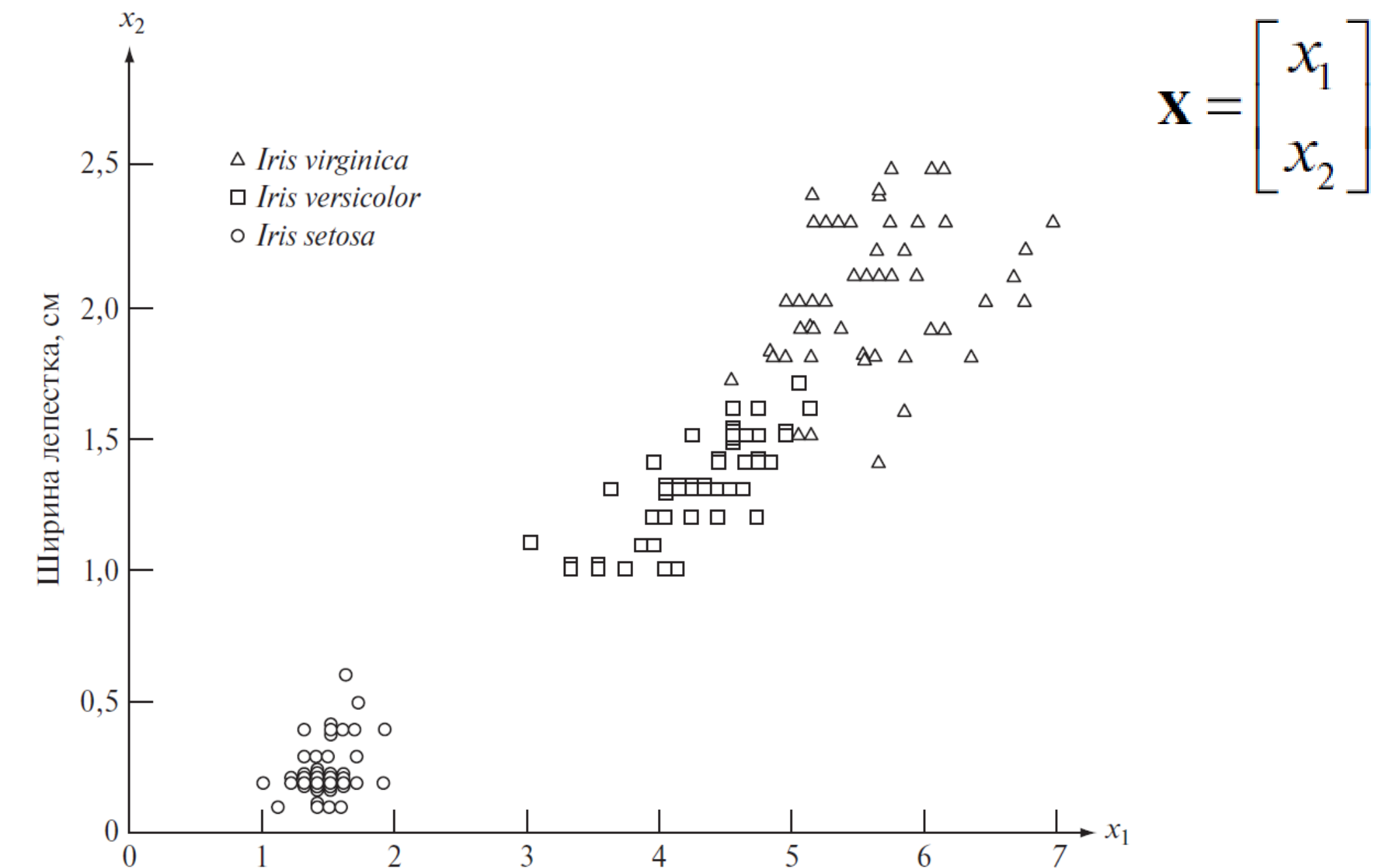
- Совокупность образов, обладающих некоторыми общими свойствами

Упорядоченное представление признаков:

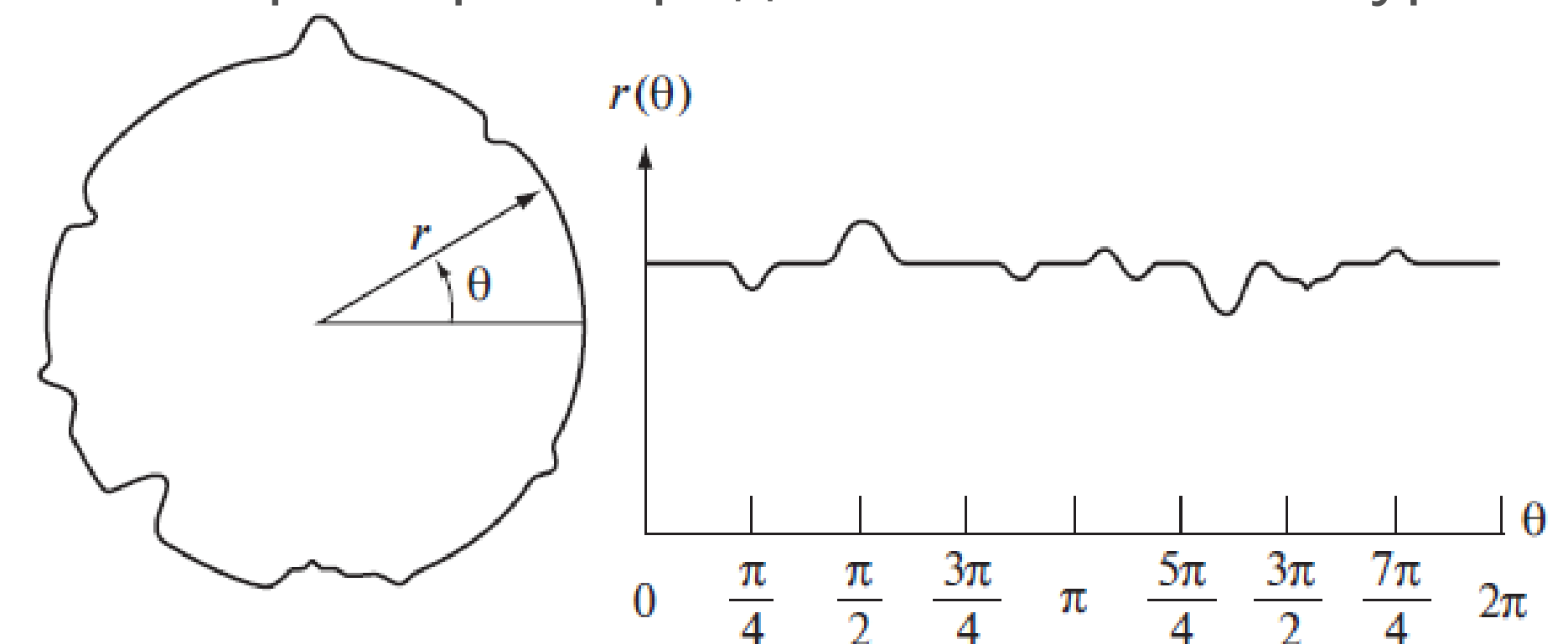
- Векторы признаков
- Символьные строки
- Деревья

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Пример 1. Дискриминантный анализ Фишера



Пример 2. Представление сигнатуры





# Распознавание объектов

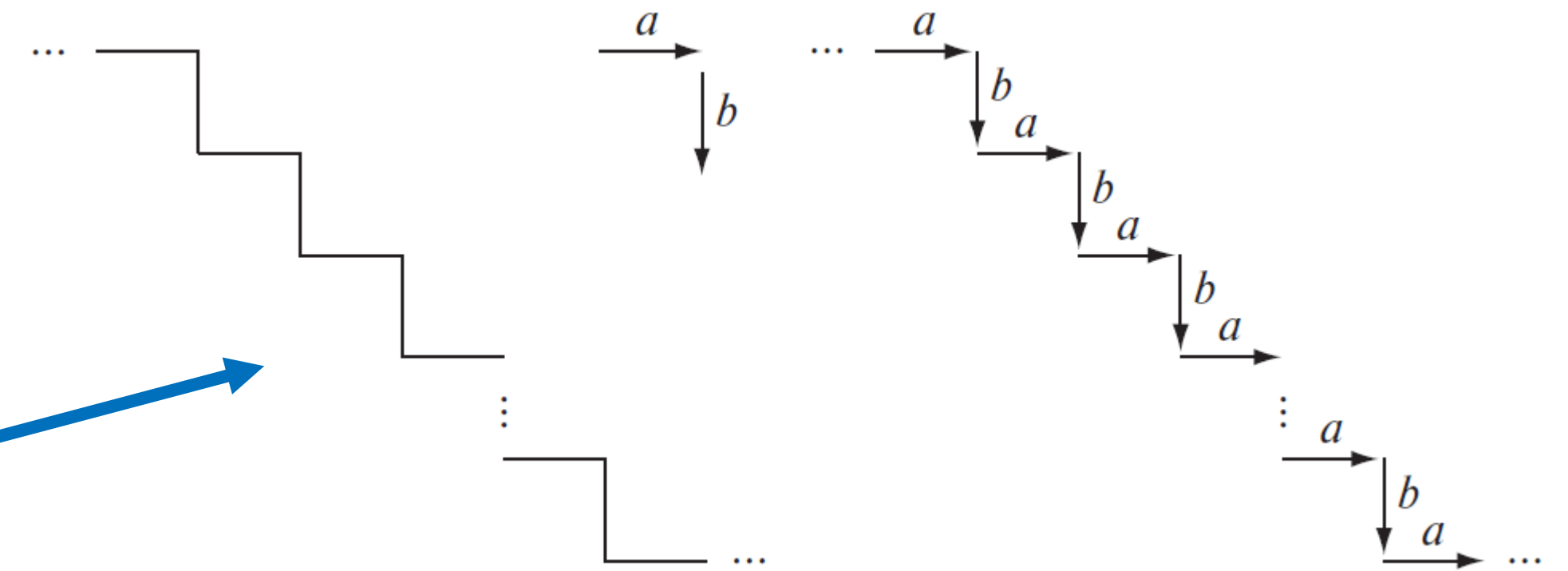
## Локальные признаки

Задача:

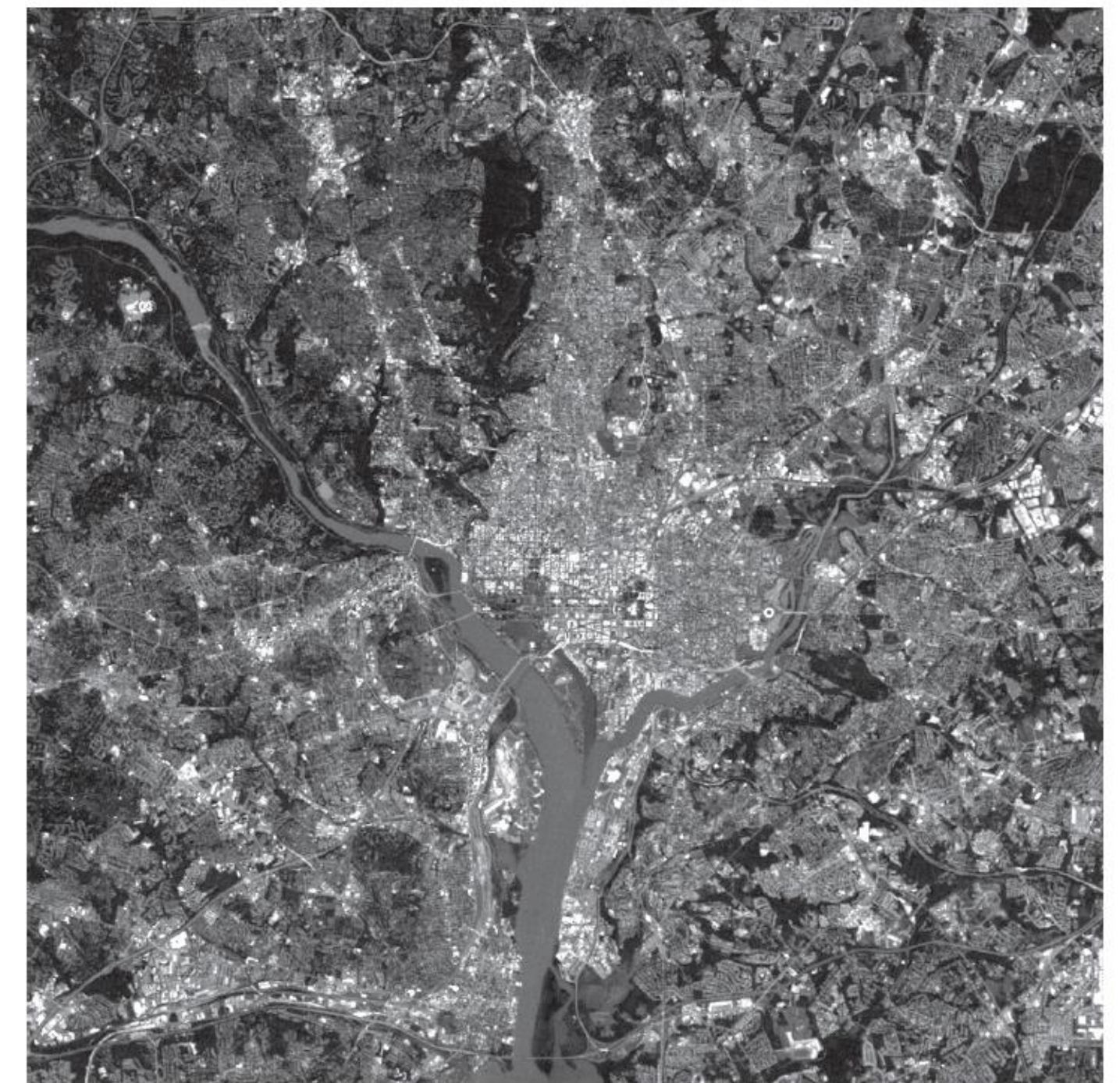
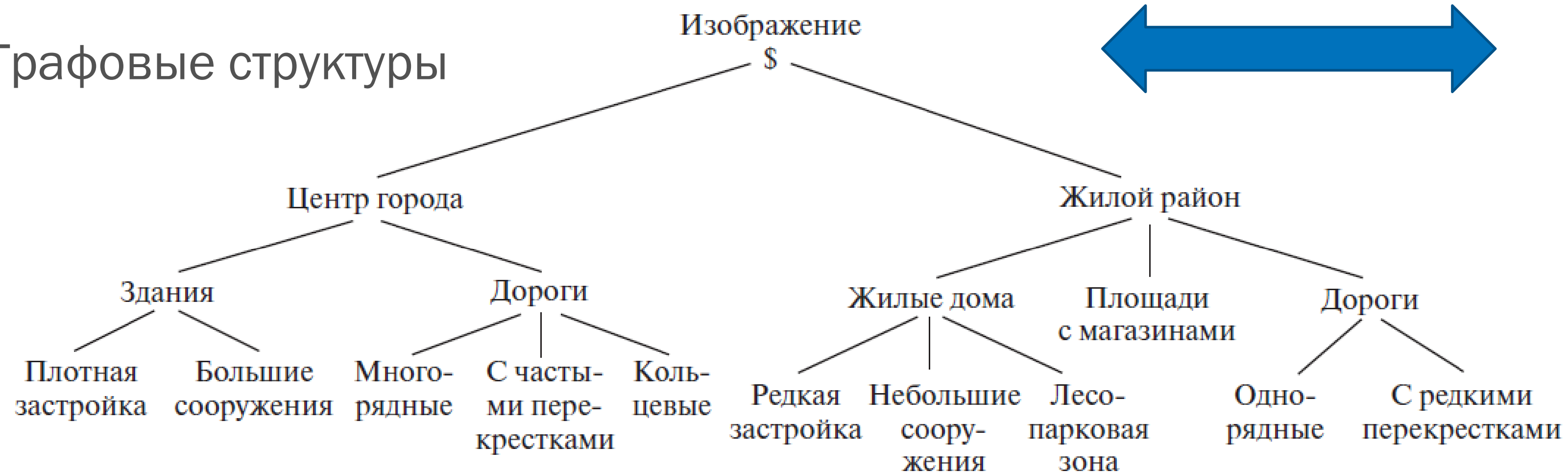
- Сохранить числовую и пространственную информацию

Символьная строка:  $w = \dots abababa \dots$

Ступенчатая структура



Графовые структуры



# Методы теории принятия решений

## Классификатор по минимуму расстояния

Решение на основе дискриминативных функций:

$$\mathbf{x} = (x_1, x_2, \dots, x_n)^T$$



$$d_i(\mathbf{x}) > d_j(\mathbf{x}) \quad j = 1, 2, \dots, W; \quad j \neq i$$



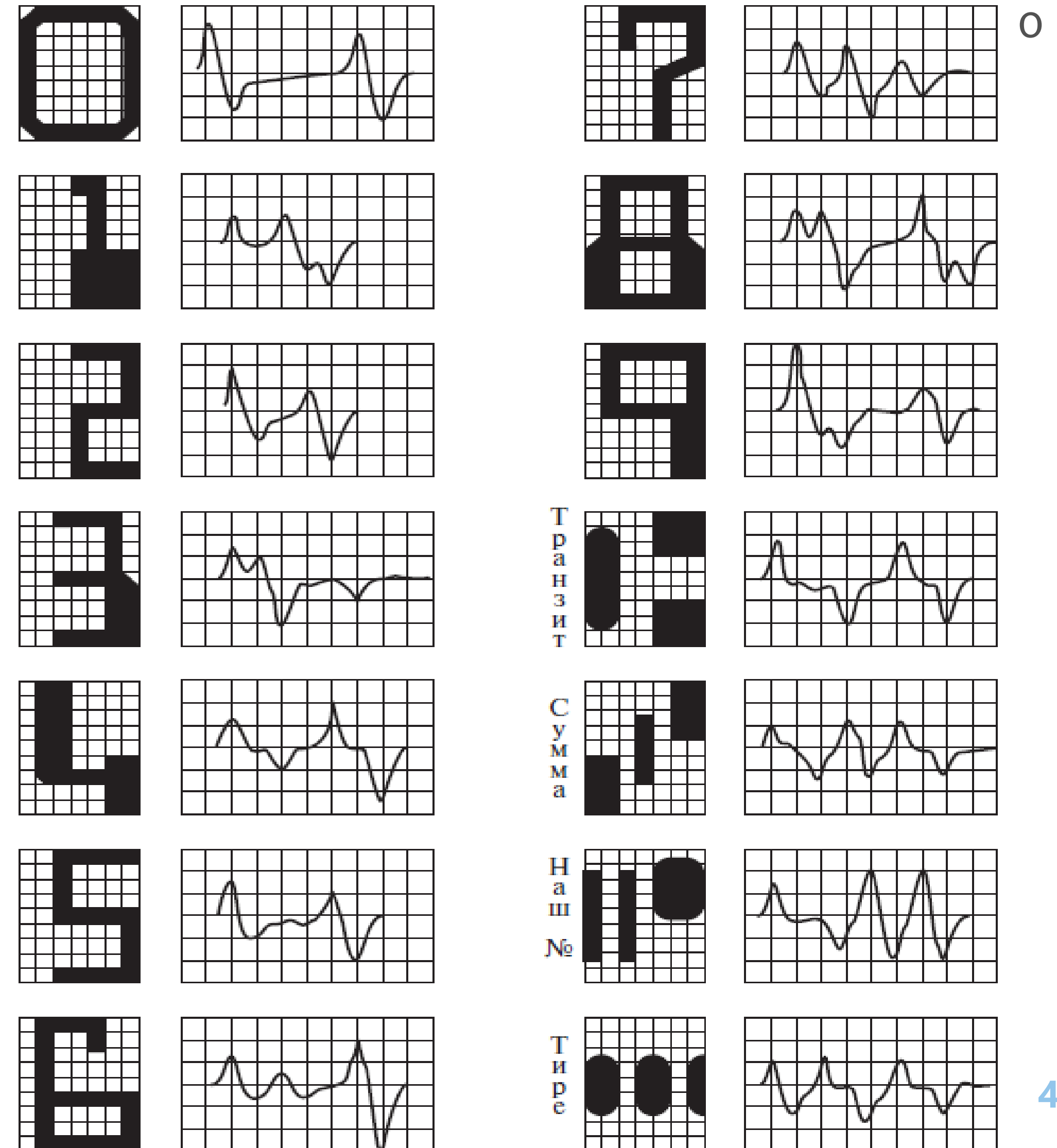
$$d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0$$

Классификатор по минимуму расстояния:

$$\mathbf{m}_j = \frac{1}{N_j} \sum_{\mathbf{x} \in \omega_j} \mathbf{x} \quad j = 1, 2, \dots, W$$

$$D_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{m}_j\| \quad j = 1, 2, \dots, W \quad \longleftrightarrow \quad d_j(\mathbf{x}) = \mathbf{x}^T \mathbf{m}_j - \frac{1}{2} \mathbf{m}_j^T \mathbf{m}_j \quad j = 1, 2, \dots, W$$

$$d_{ij}(\mathbf{x}) = d_i(\mathbf{x}) - d_j(\mathbf{x}) = \mathbf{x}^T (\mathbf{m}_i - \mathbf{m}_j) - \frac{1}{2} (\mathbf{m}_i - \mathbf{m}_j)^T (\mathbf{m}_i - \mathbf{m}_j) = 0$$





# Методы теории принятия решений

## Корреляционное сопоставление

Корреляционное сопоставление с эталоном

Поиск откликов при пространственной фильтрации:

$$c(x, y) = \sum_s \sum_t w(s, t) f(x + s, y + t)$$

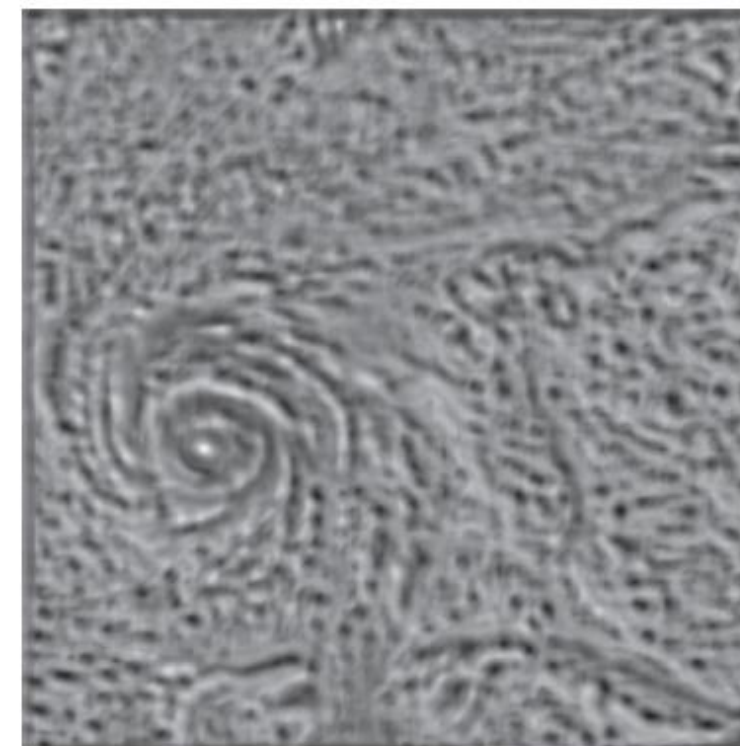
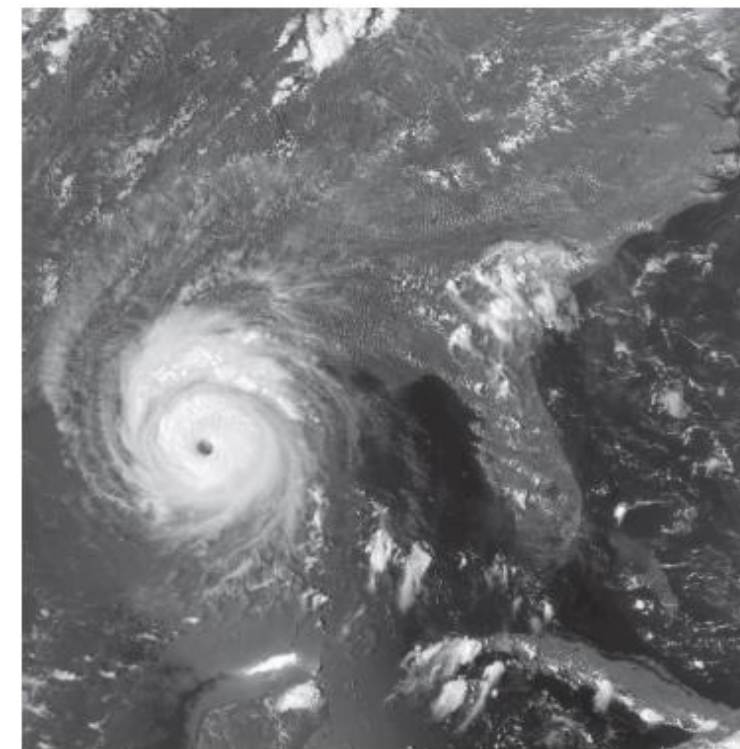


$$f(x, y) \star w(x, y) \Leftrightarrow F^*(u, v) W(u, v)$$



$$\gamma(x, y) = \frac{\sum_s \sum_t [w(s, t) - \bar{w}] [f(x + s, y + t) - \bar{f}_{xy}]}{\left\{ \sum_s \sum_t [w(s, t) - \bar{w}]^2 \sum_s \sum_t [f(x + s, y + t) - \bar{f}_{xy}]^2 \right\}^{\frac{1}{2}}}$$

Коэффициенты  
корреляции == отклики



Максимальный отклик  
== точка локализации  
эпицентра



Эталон  
эпицентра  
урагана

# Статист. оптимальные классификаторы

## Байесовский классификатор

Плотности распределения для двух одномерных классов:  $x_0$  – положение разделяющей поверхности при равновероятных классах

Вероятностный подход к распознаванию:

- Средняя величина потерь через вероятность принадлежности образа к классу

$$r_j(\mathbf{x}) = \sum_{k=1}^W L_{kj} p(\omega_k | \mathbf{x}) \quad r_j(\mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{k=1}^W L_{kj} p(\mathbf{x} | \omega_k) P(\omega_k)$$

- Условие принадлежности к классу

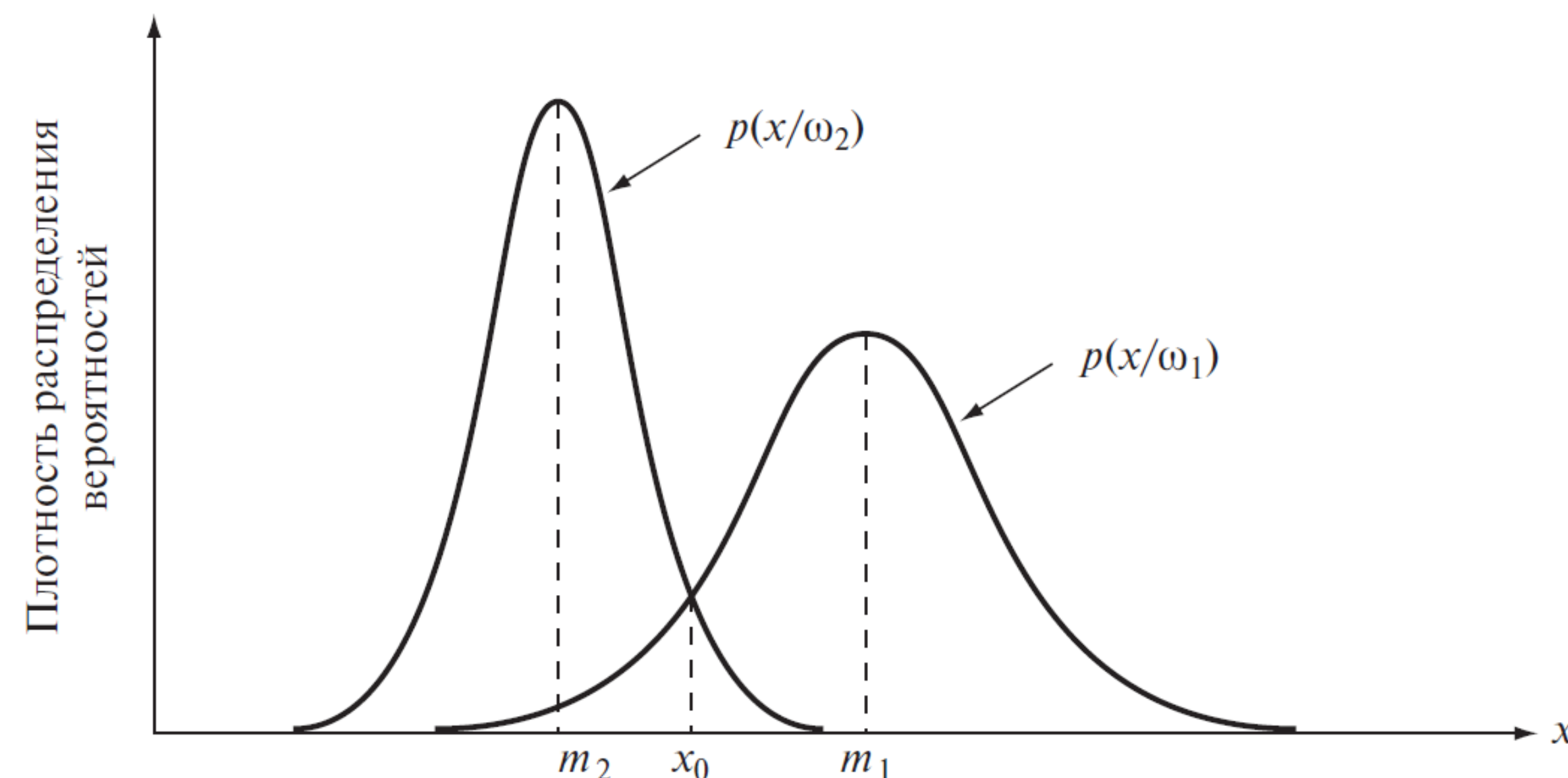
$$\sum_{k=1}^W L_{ki} p(\mathbf{x} | \omega_k) P(\omega_k) < \sum_{q=1}^W L_{qj} p(\mathbf{x} | \omega_q) P(\omega_q)$$

- С учетом функции потерь  $L_{ij} = 1 - \delta_{ij}$

$$r_j(\mathbf{x}) = \sum_{k=1}^W (1 - \delta_{kj}) p(\mathbf{x} | \omega_k) P(\omega_k) = p(\mathbf{x}) - p(\mathbf{x} | \omega_j) P(\omega_j)$$

- Вычисление дискриминантных функций

$$d_j(\mathbf{x}) = p(\mathbf{x} | \omega_j) P(\omega_j) \quad j = 1, 2, \dots, W$$



Байесовский классификатор для классов с нормальным распределением

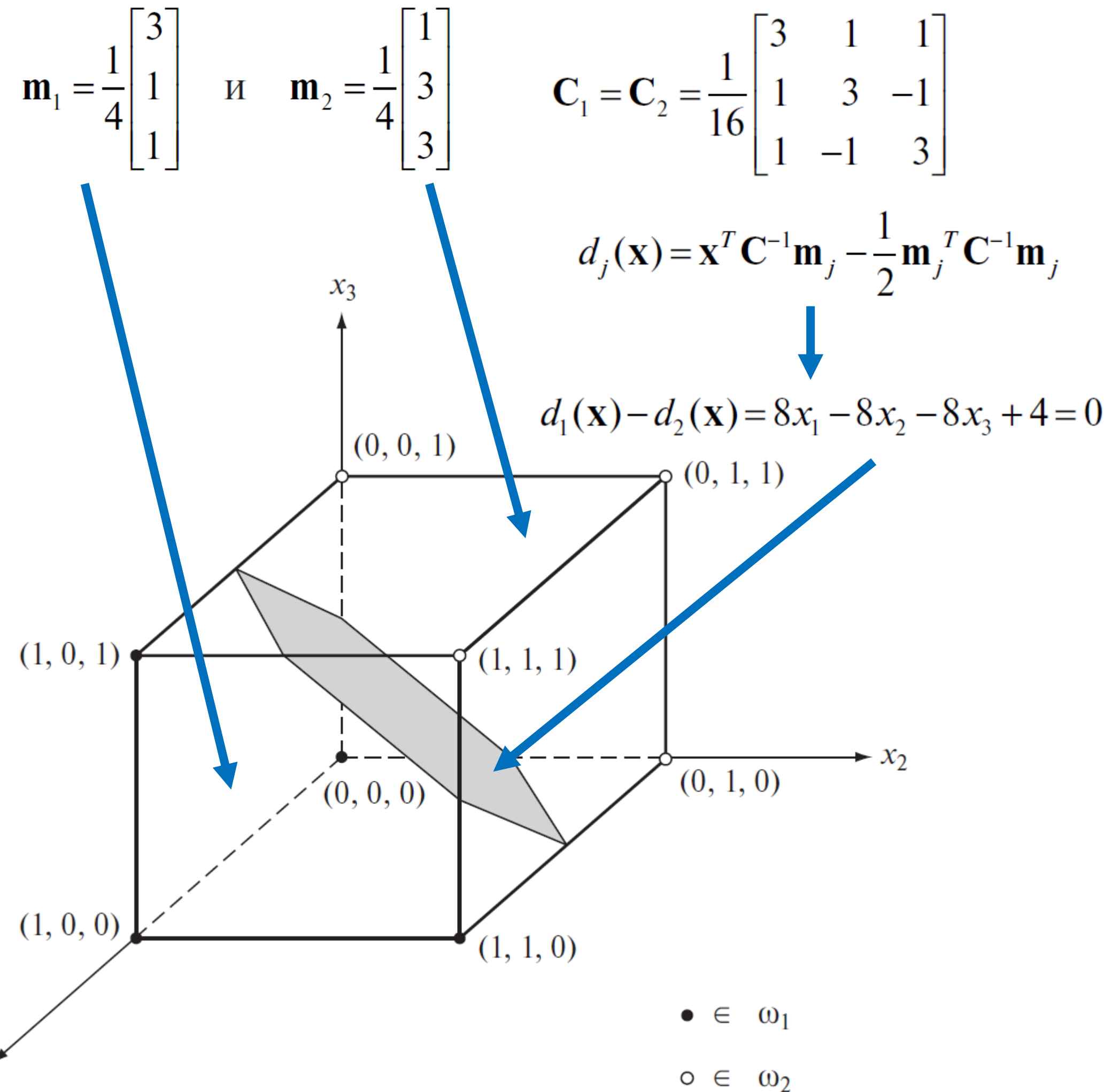
$$d_j(x) = p(x | \omega_j) P(\omega_j) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(x-m_j)^2}{2\sigma_j^2}} P(\omega_j) \quad j = 1, 2$$



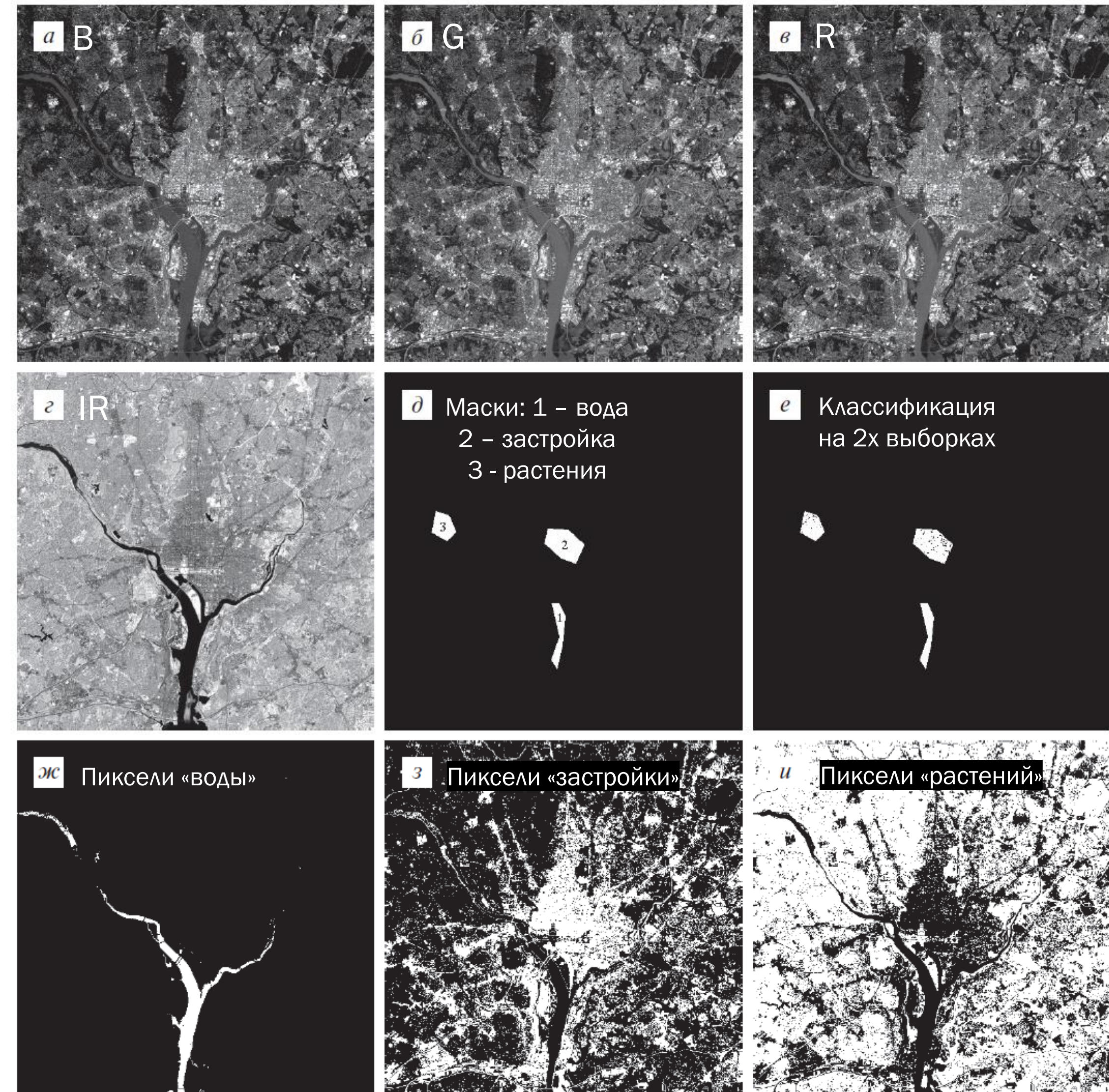
# Статист. оптимальные классификаторы

## Применение байесовского классификатора

Трехмерные образы



Классификация мультиспектральных изображений





# Нейронные сети

## Предпосылки

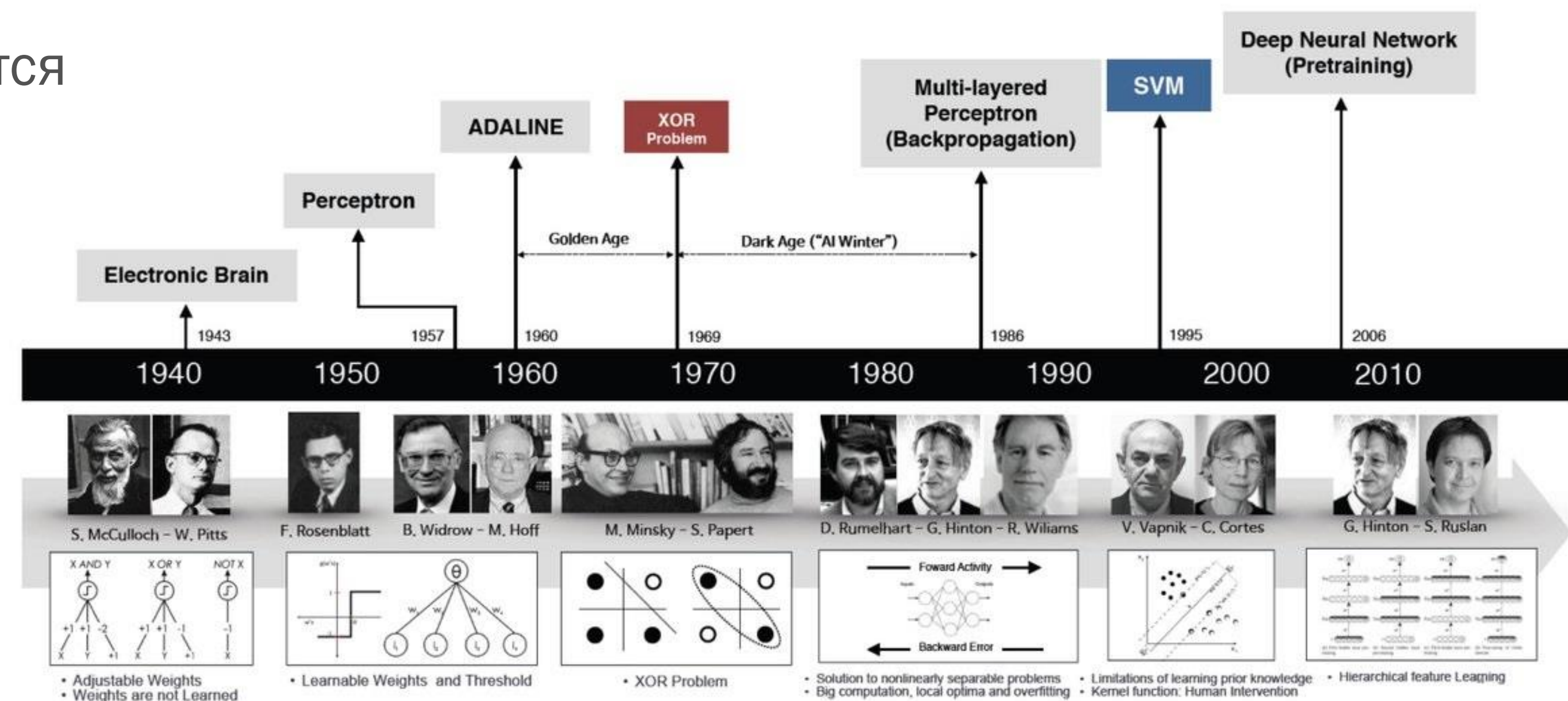
Предыдущие идеи:

- Обучение – процесс, в ходе которого с помощью обучающей выборки оцениваются параметры функции, на основе которой строятся дискриминантные функции

Реальные задачи:

- Статистические свойства классов заранее неизвестны
- Требуется устранить необходимость использовать предположения о функциях плотностей распределения вероятностей классов
- => построение дискриминантных функций непосредственно в ходе обучения

Этапы развития идей нейронных сетей в контексте машинного обучения





# Нейронные сети

## Перцептрон для разделения двух классов

Простейший вид:

- Построение линейной дискриминантной функции для двух линейно разделимых классов
- Отклик на основе взвешенной суммы входов

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

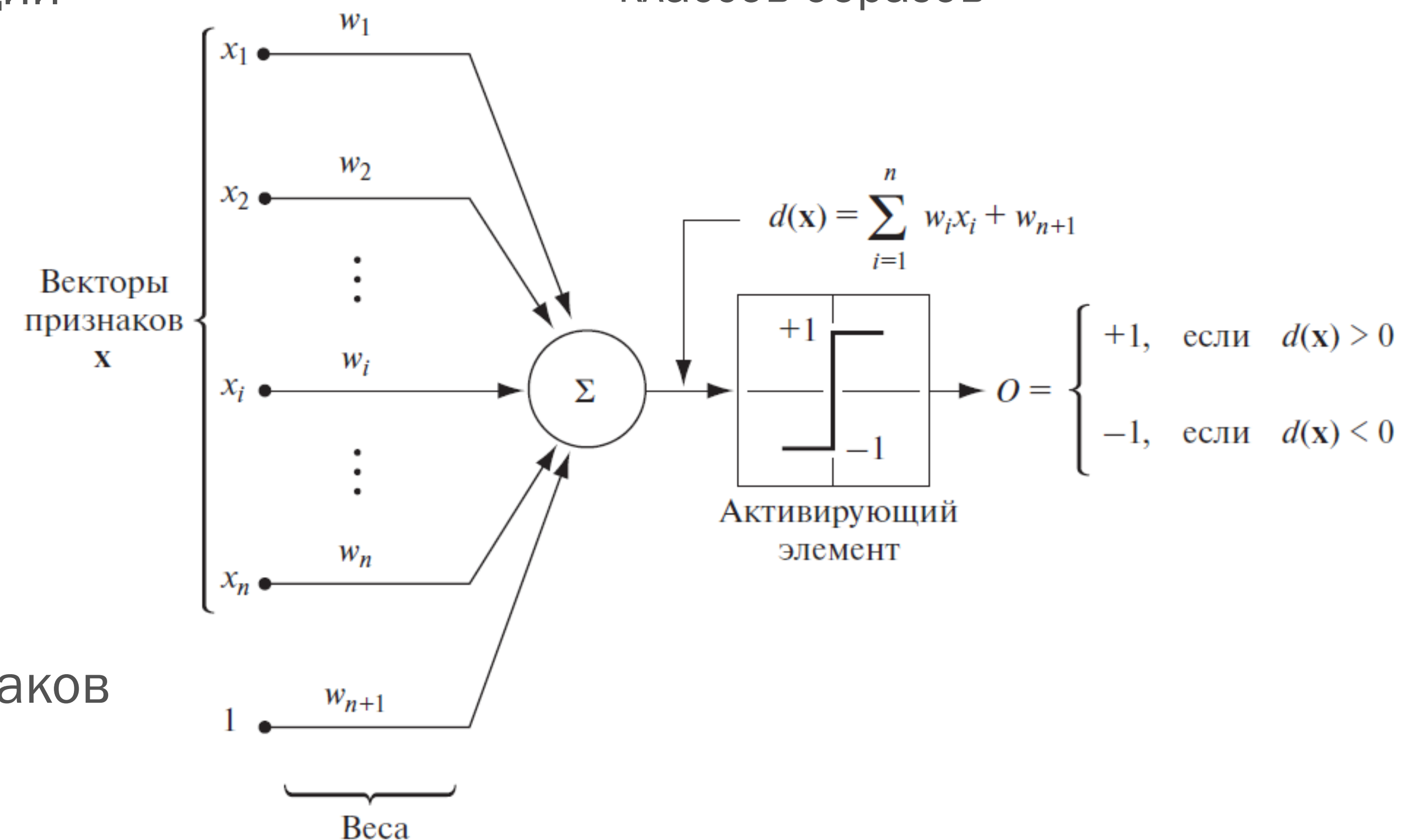
- С точки зрения разделяющей поверхности

$$d(\mathbf{x}) = \sum_{i=1}^n w_i x_i + w_{n+1} = 0$$

- Выражение через расширенный вектор признаков

$$d(\mathbf{y}) = \sum_{i=1}^{n+1} w_i y_i = \mathbf{w}^T \mathbf{y}$$

Представление модели персептрона для двух классов образов





# Нейронные сети

## Варианты обучения перцептрона

Случай двух линейно разделимых классов:

- Две обучающие выборки  $\omega_1$  и  $\omega_2$ ,  $\mathbf{w}(1)$  — некоторый начальный весовой вектор, выбираемый произвольно
- На  $k$ -м шаге итерации, если  $y(k) \in \omega_1$  и  $\mathbf{w}^T(k)y(k) \leq 0$ , то  $\mathbf{w}(k)$  заменяется на  $\mathbf{w}(k+1) = \mathbf{w}(k) + cy(k)$
- Если если  $y(k) \in \omega_2$  и  $\mathbf{w}^T(k)y(k) \geq 0$ , то  $\mathbf{w}(k)$  заменяется на  $\mathbf{w}(k+1) = \mathbf{w}(k) - cy(k)$

Алгоритм постоянного коэффициента коррекции.

Сходимость алгоритма => обучающие выборки обоих классов целиком проходят через последовательность итераций без единой ошибки

Случай линейно неразделимых классов:

Дельта-правило наименьшего среднего квадрата Уидроу - Хоффа

- Целевая функция  $J(\mathbf{w}) = \frac{1}{2}(r - \mathbf{w}^T \mathbf{y})^2$
- Изменение весового вектора на  $k$ -м шаге

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \alpha \left[ \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} \right]_{\mathbf{w}=\mathbf{w}(k)}$$

- Величина допущенной ошибки

$$e(k) = r(k) - \mathbf{w}^T(k)\mathbf{y}(k)$$

- Уменьшение ошибки на  $k$ -й итерации

$$\Delta e = -\alpha e(k) \mathbf{y}^T(k) \mathbf{y}(k) = -\alpha e(k) \|\mathbf{y}(k)\|^2$$

Алгоритм сходится к решению, минимизирующему средний квадрат ошибки на образах обучающей выборки



# Нейронные сети

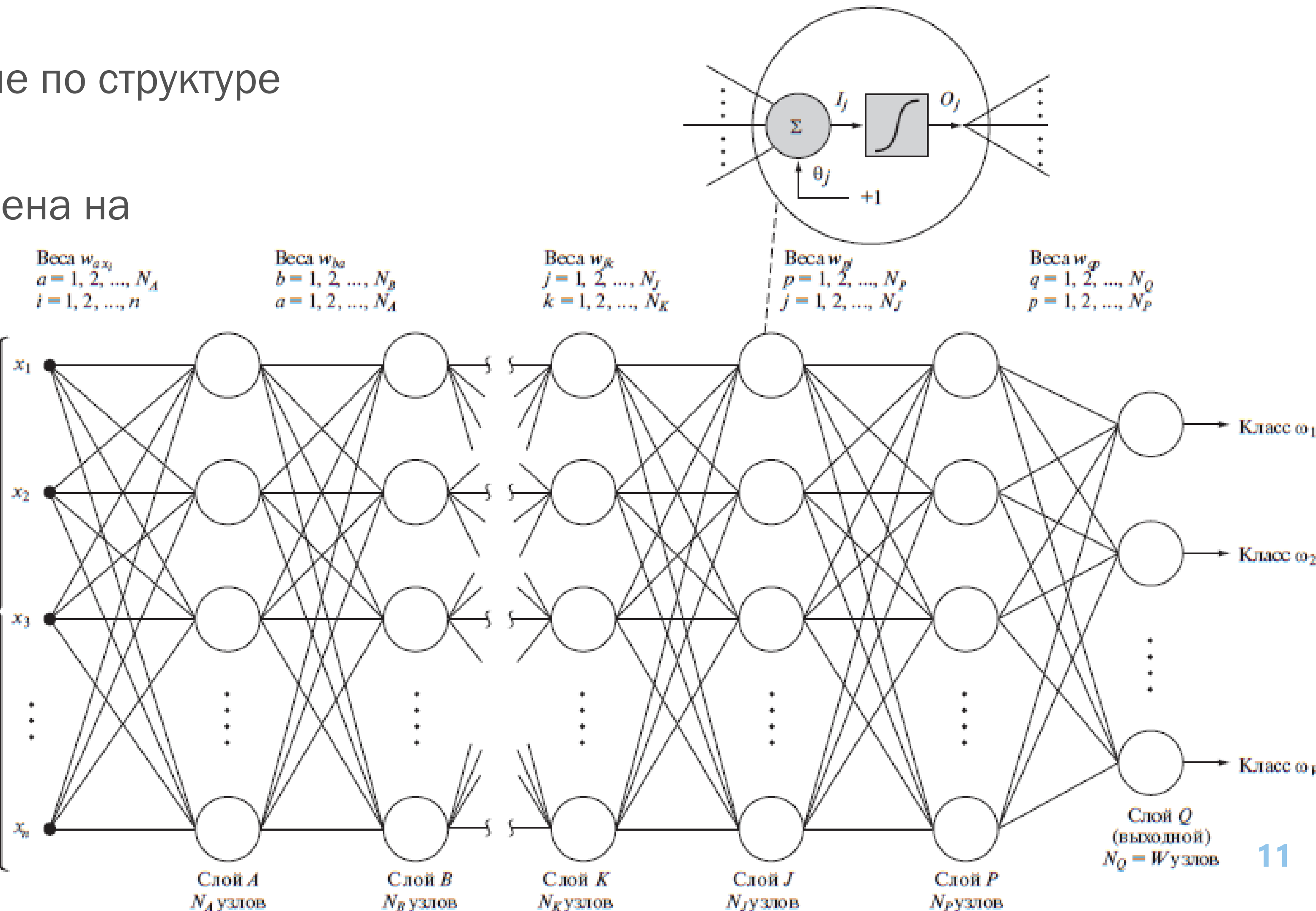
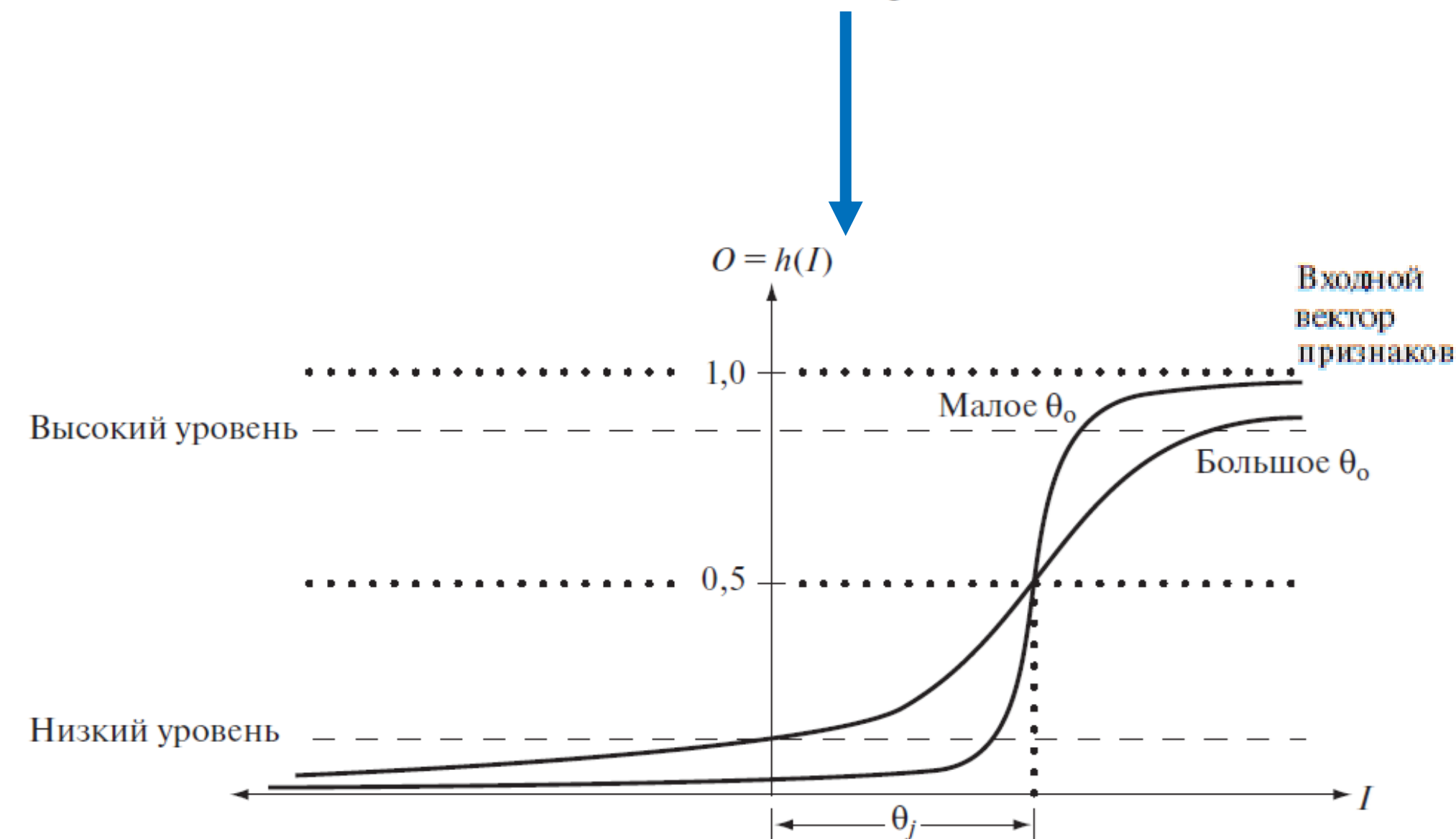
## Многослойные нейронные сети

Модель многослойной нейронной сети

Базовая архитектура:

- Слои, в которых находятся идентичные по структуре вычислительные узлы
- Пороговая функция активации заменена на непрерывную S-образную функцию

$$h_j(I_j) = \frac{1}{1 + e^{-(I_j - \theta_j)/\theta_0}}$$



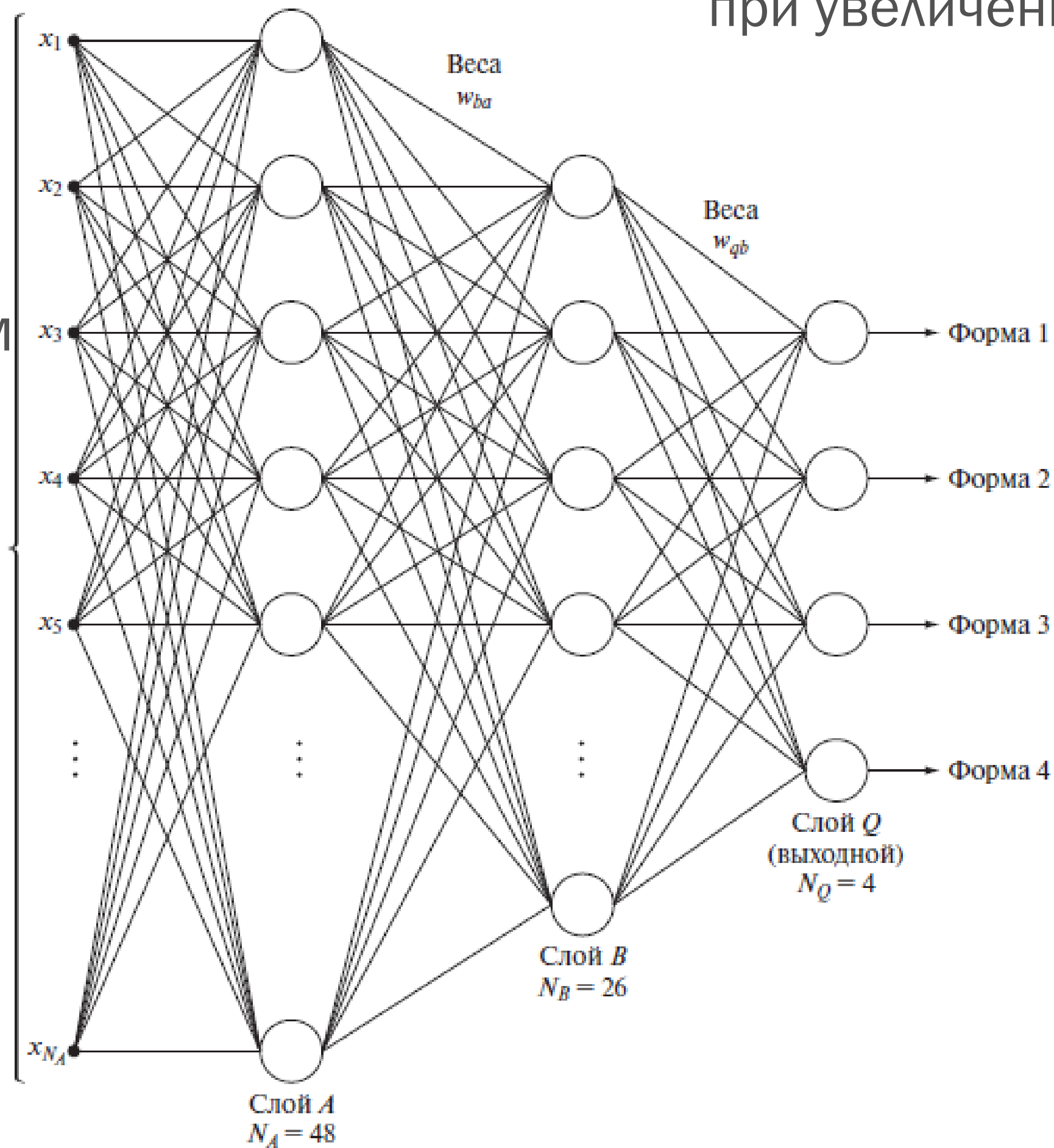
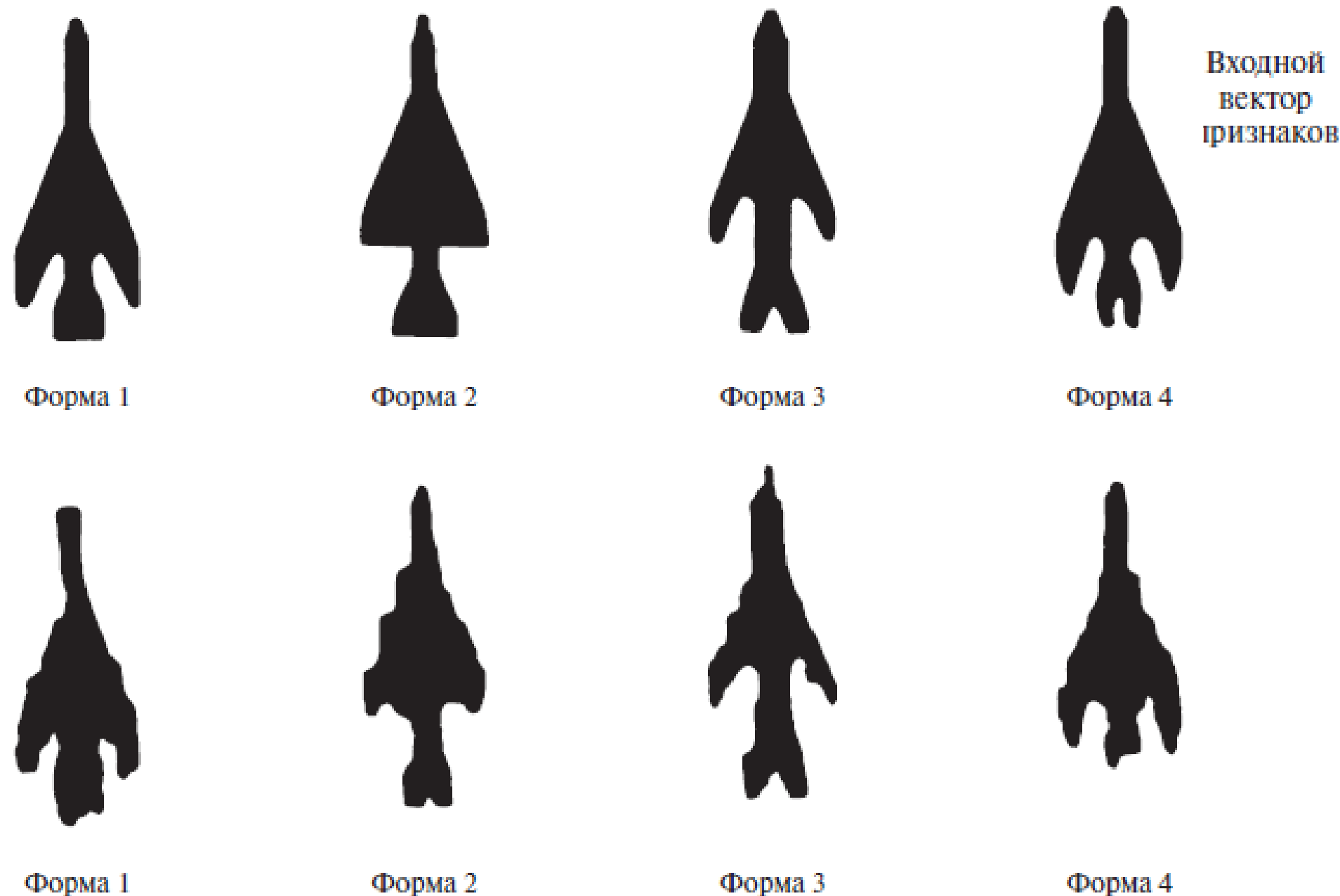


# Нейронные сети

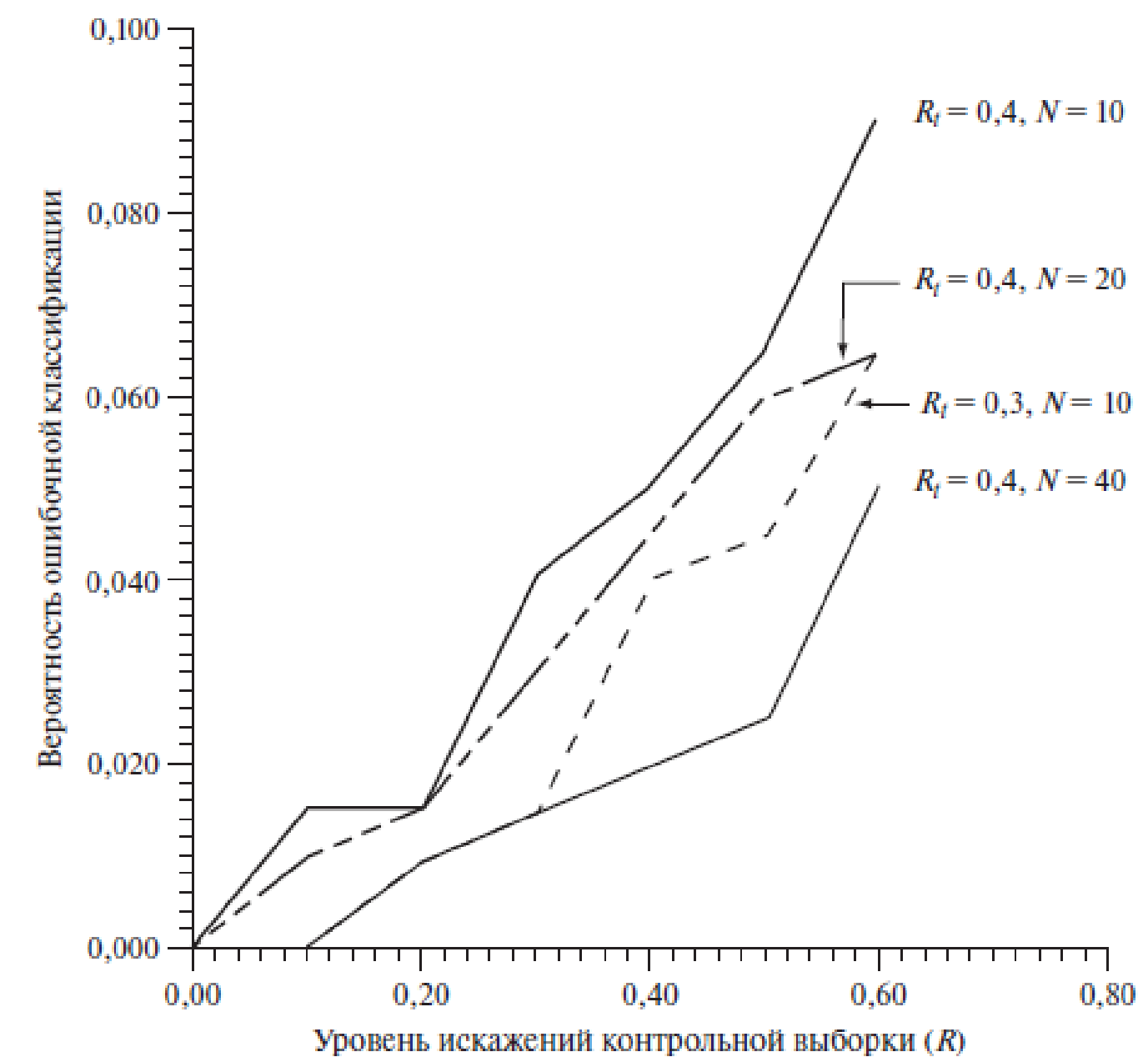
## Пример классификации формы фигуры

Поэтапное обучение модели:

- Инициализация весов
- Обучение на эталонных формах
- Несколько итераций обучения на формах с постепенным увеличением искажений



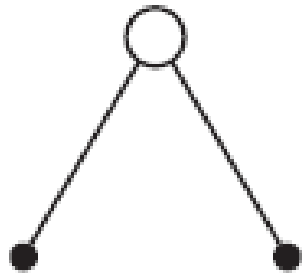
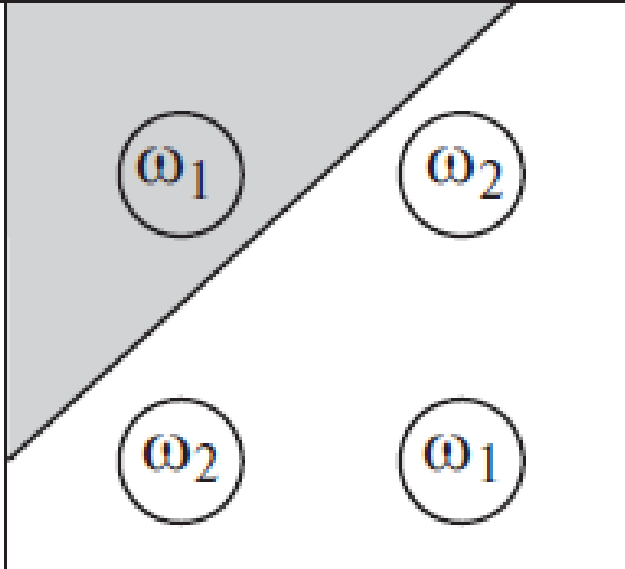
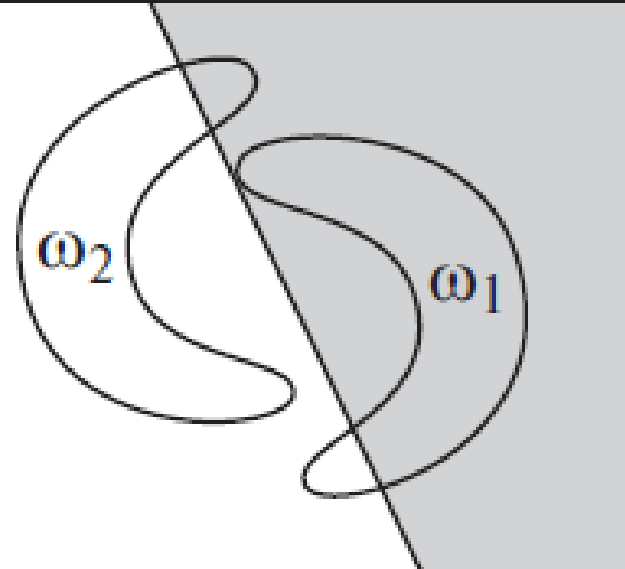
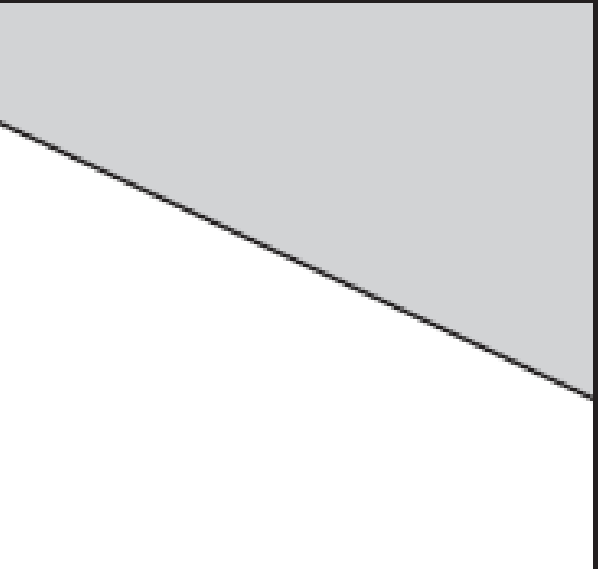
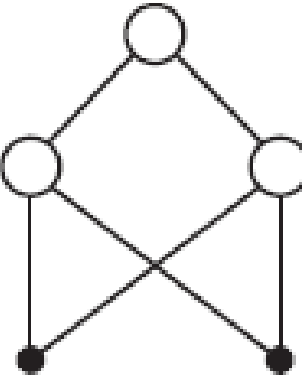
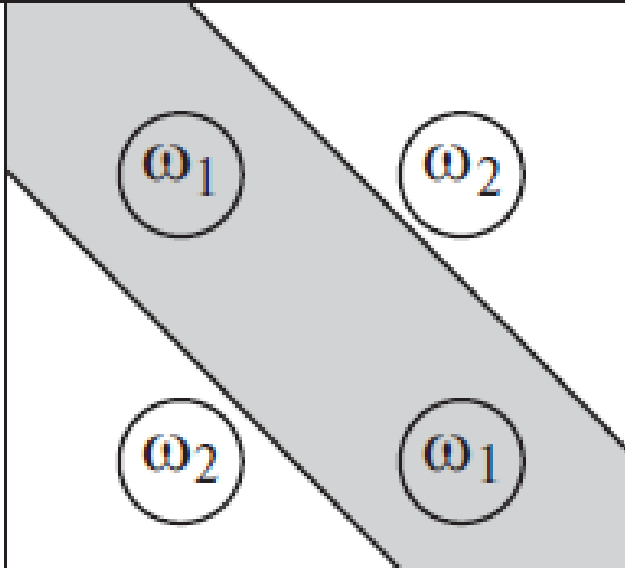
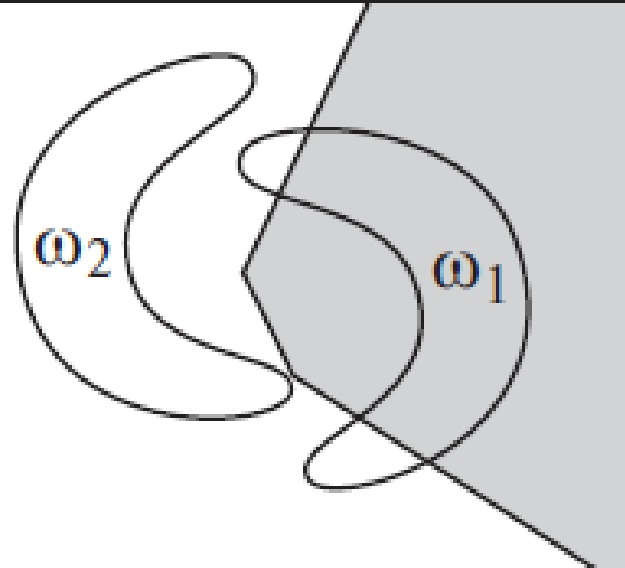
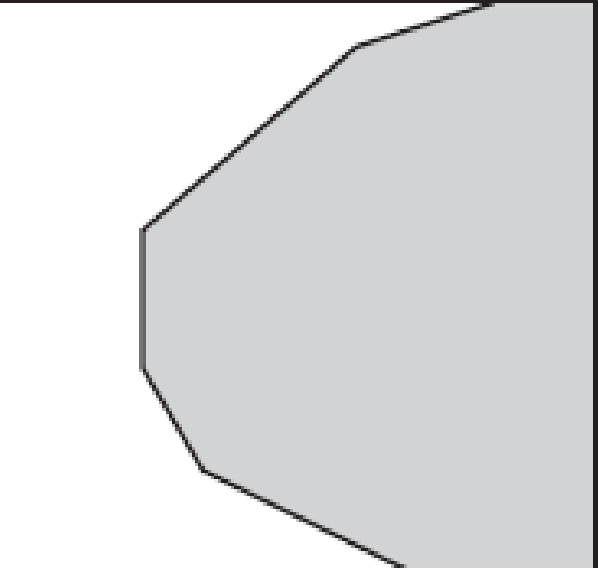
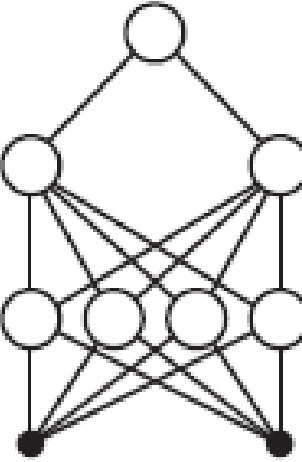
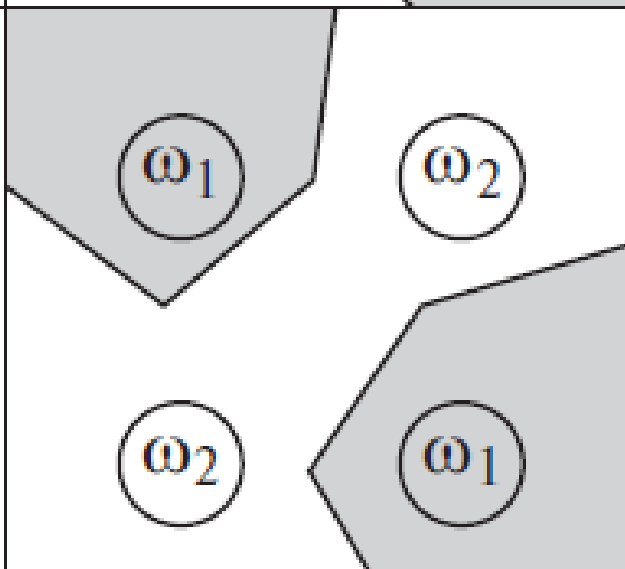
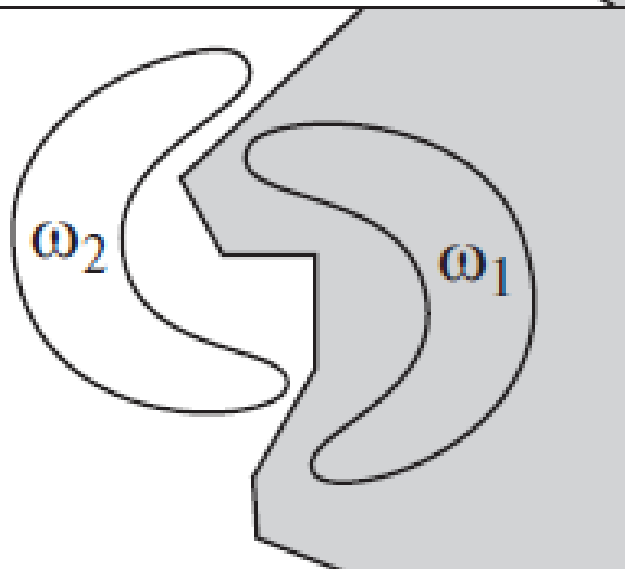
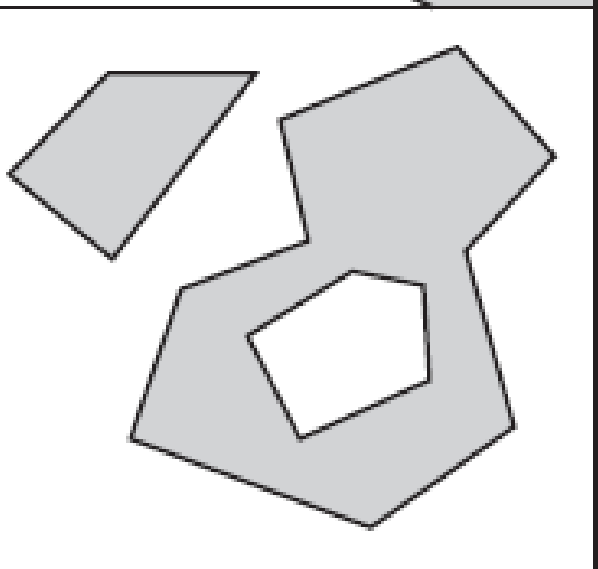
Повышение качества распознавания для  $R_t = 0,4$  при увеличении объема обучающей выборки

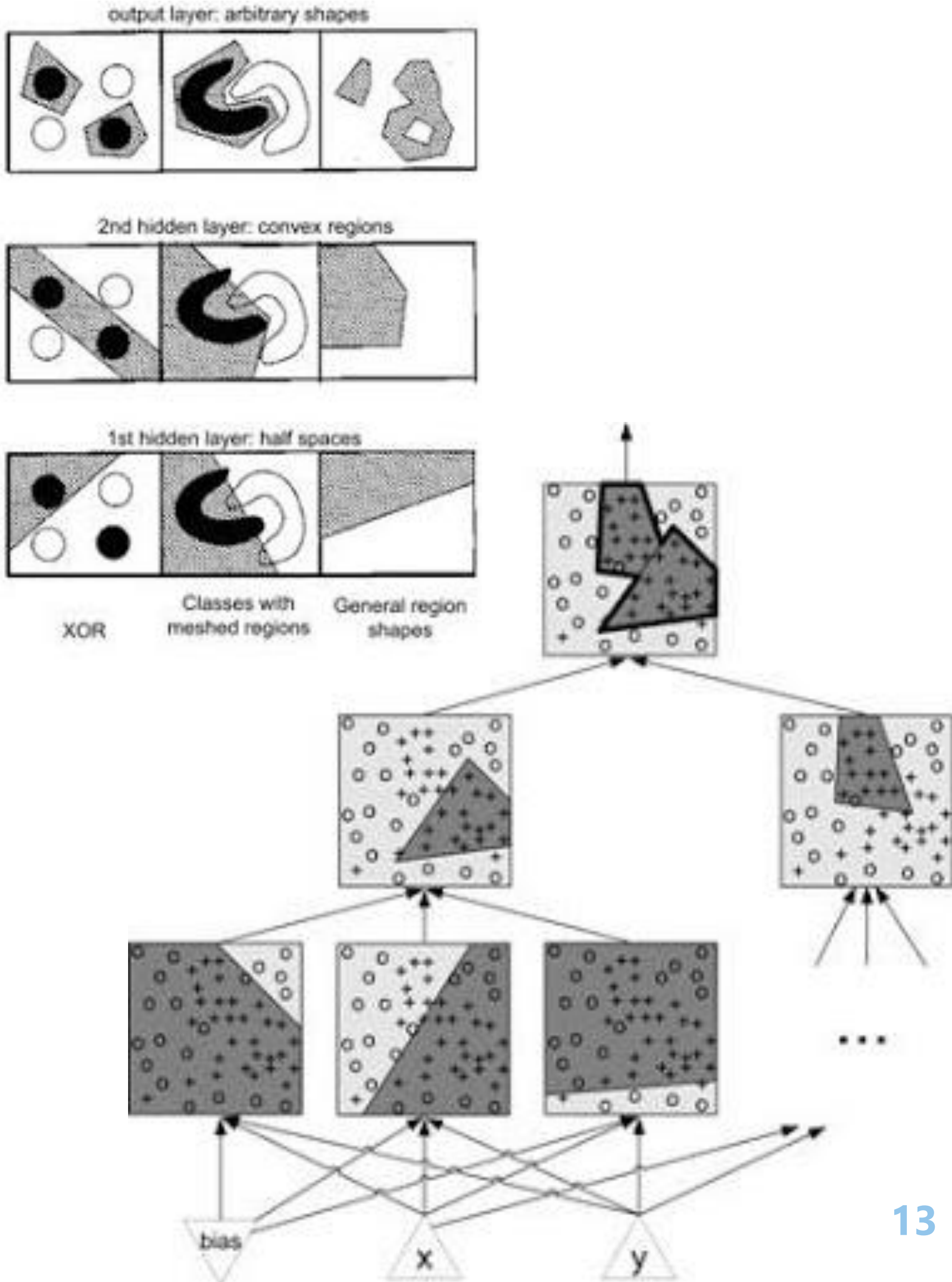




# Нейронные сети

## Сложность разделяющих поверхностей

Структура сети	Виды областей решений	Решение проблемы «исключающего ИЛИ»	Классы с «зацепляющимися» областями	Разделяющие поверхности наиболее общей формы
Однослойная 	Полупространство			
Двухслойная 	Открытая или замкнутая выпуклая область			
Трехслойная 	Произвольная (сложность ограничена числом узлов)			



# Структурные методы распознавания

## Сопоставление номеров фигур

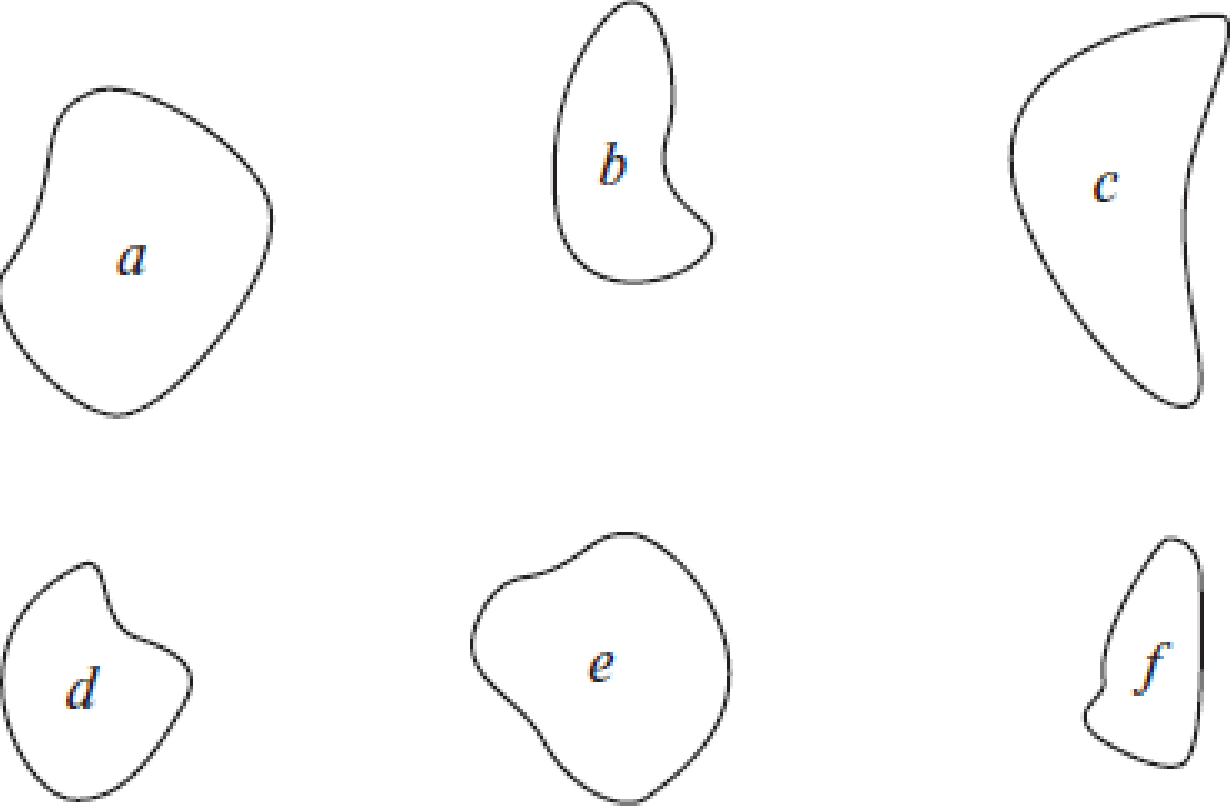
Ранее:

- Учет количественных характеристик, без структурных связей

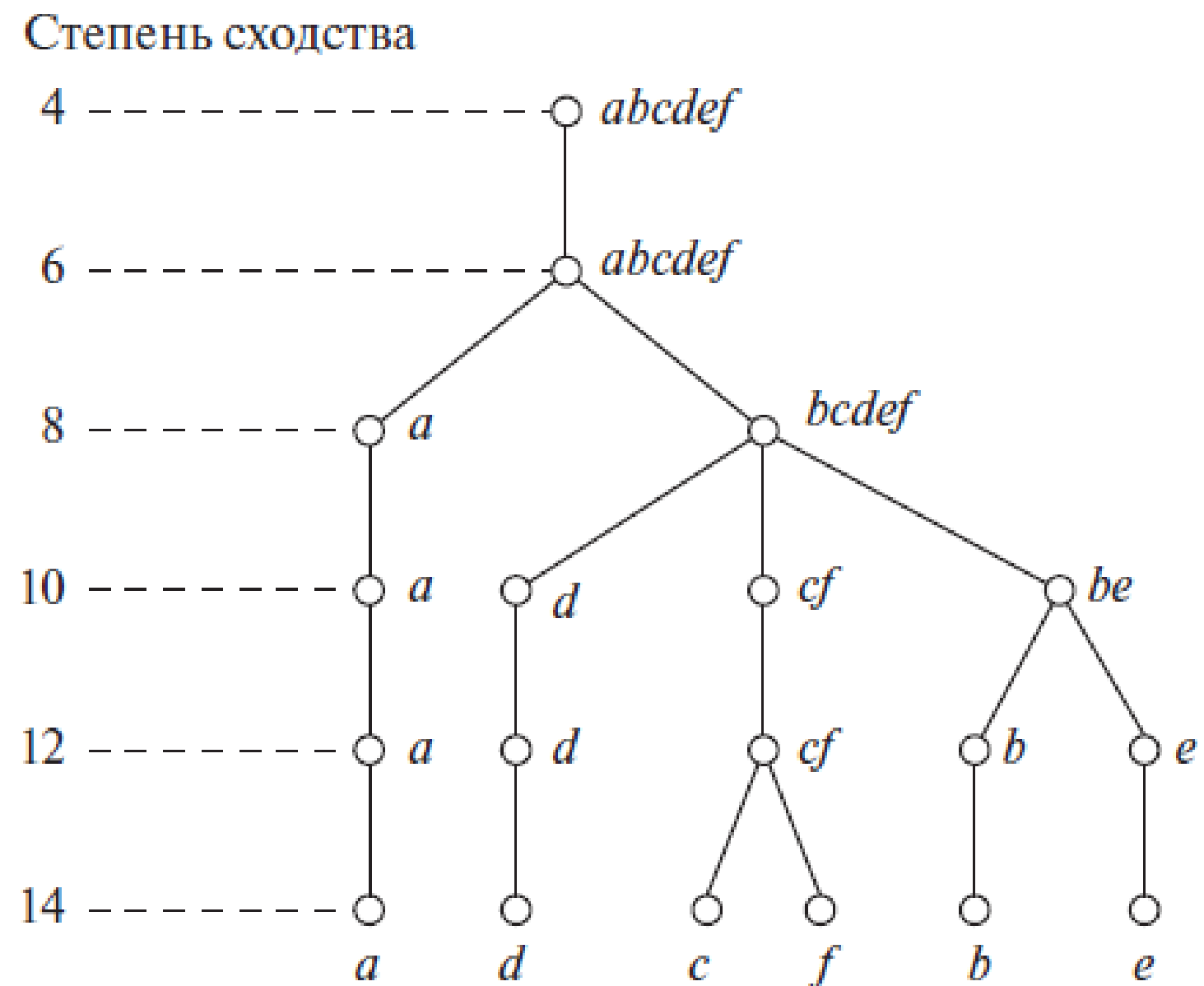
Сопоставление:

- Степень сходства к формы границ двух областей определяется как наибольшее значение порядка, при котором номера фигур совпадают

Фигуры



Дерево сходство



Матрица сходства

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
<i>a</i>	∞	6	6	6	6	6
<i>b</i>		∞	8	8	10	8
<i>c</i>			∞	8	8	12
<i>d</i>				∞	8	8
<i>e</i>					∞	8
<i>f</i>						∞



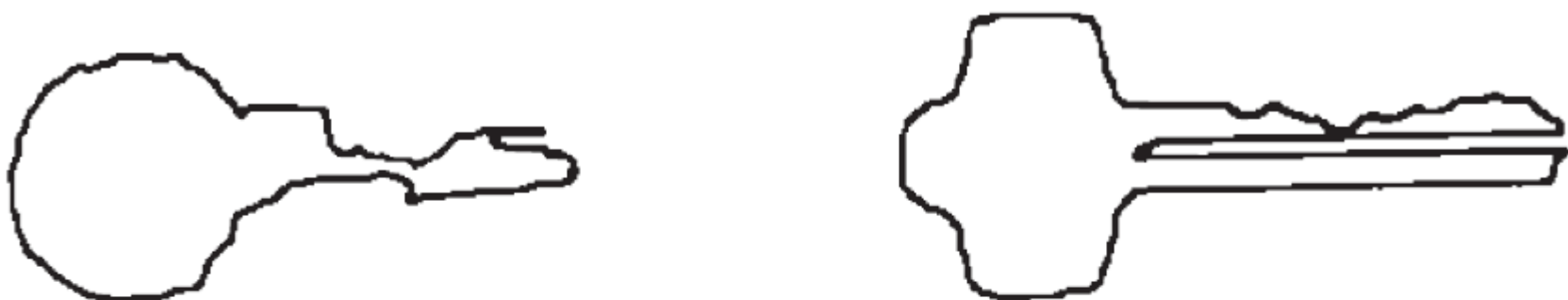
# Структурные методы распознавания

## Сопоставление строк символов Примеры границ объектов из двух различных классов

- Границы a и b закодированы в виде строк символов  $a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_m$
- Совпадение на k-й позиции, если  $a_k == b_k$
- Пусть  $\alpha$  — общее число совпадений этих строк, тогда число несовпадающих символов равно  $\beta = \max(|a|, |b|) - \alpha$

- Мера сходства a и b

$$R = \frac{\alpha}{\beta} = \frac{\alpha}{\max(|a|, |b|) - \alpha}$$



Аппроксимация этих границ ломаными линиями



R	1.a	1.b	1.c	1.d	1.e	1.f
1.a	∞					
1.b	16.0	∞				
1.c	9.6	26.3	∞			
1.d	5.1	8.1	10.3	∞		
1.e	4.7	7.2	10.3	14.2	∞	
1.f	4.7	7.2	10.3	8.4	23.7	∞

R	2.a	2.b	2.c	2.d	2.e	2.f
2.a	∞					
2.b	33.5	∞				
2.c	4.8	5.8	∞			
2.d	3.6	4.2	19.3	∞		
2.e	2.8	3.3	9.2	18.3	∞	
2.f	2.6	3.0	7.7	13.5	27.0	∞

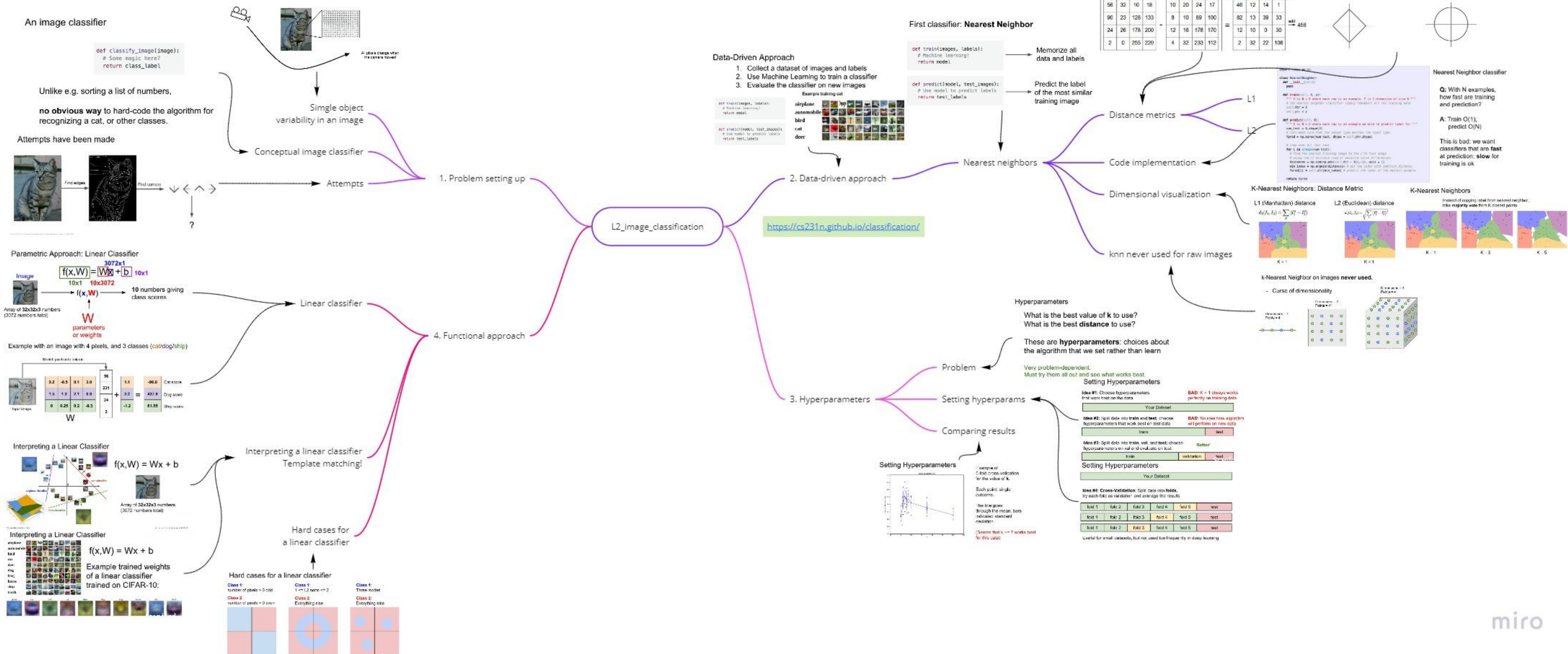
Таблицы значений R

R	1.a	1.b	1.c	1.d	1.e	1.f
2.a	1.24	1.50	1.32	1.47	1.55	1.48
2.b	1.18	1.43	1.32	1.47	1.55	1.48
2.c	1.02	1.18	1.19	1.32	1.39	1.48
2.d	1.02	1.18	1.19	1.32	1.29	1.40
2.e	0.93	1.07	1.08	1.19	1.24	1.25
2.f	0.89	1.02	1.02	1.24	1.22	1.18



# Пример структуризации знаний

## На примере лекции курса CS231n





# ML в компьютерном зрении / CS231n

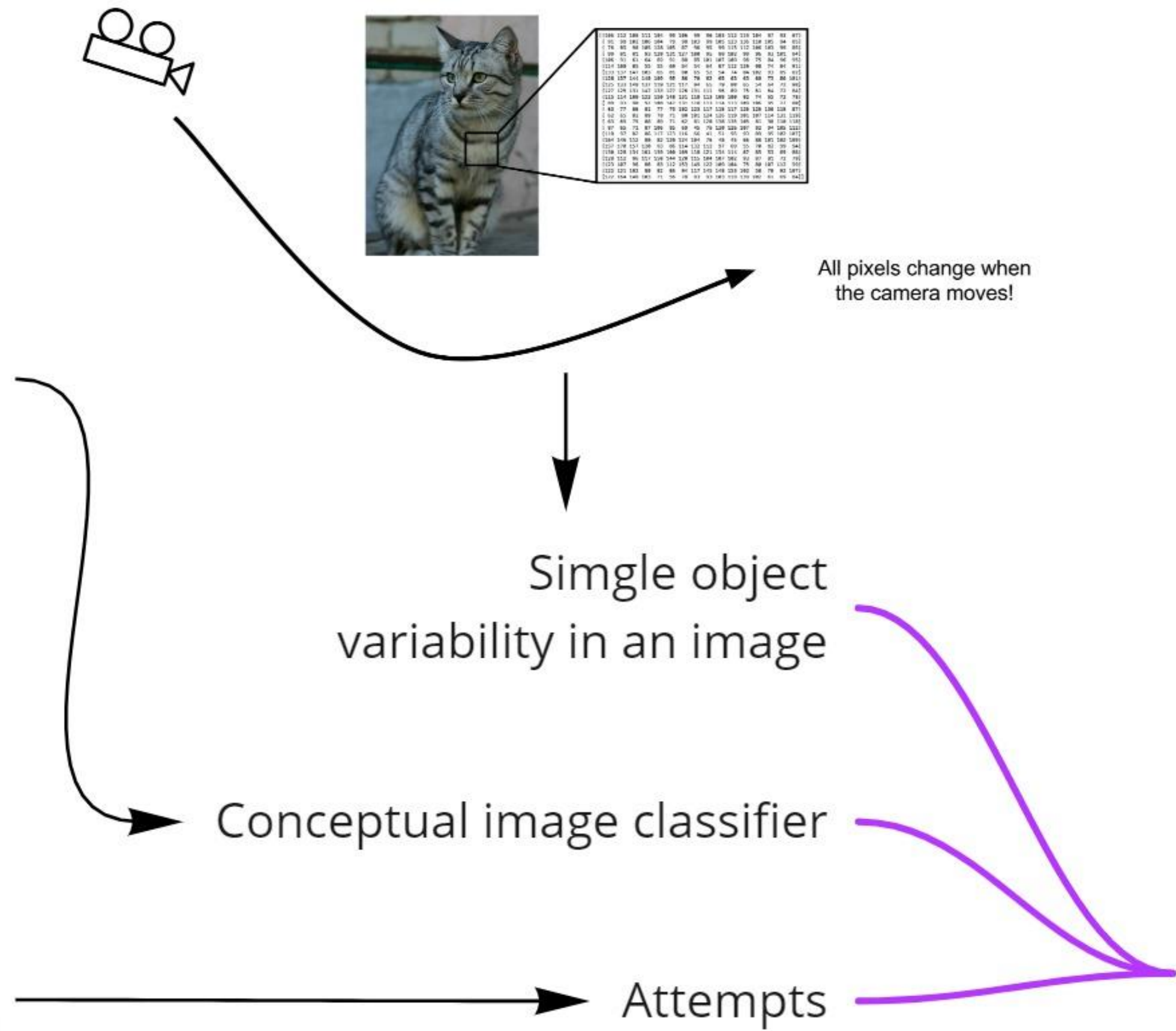
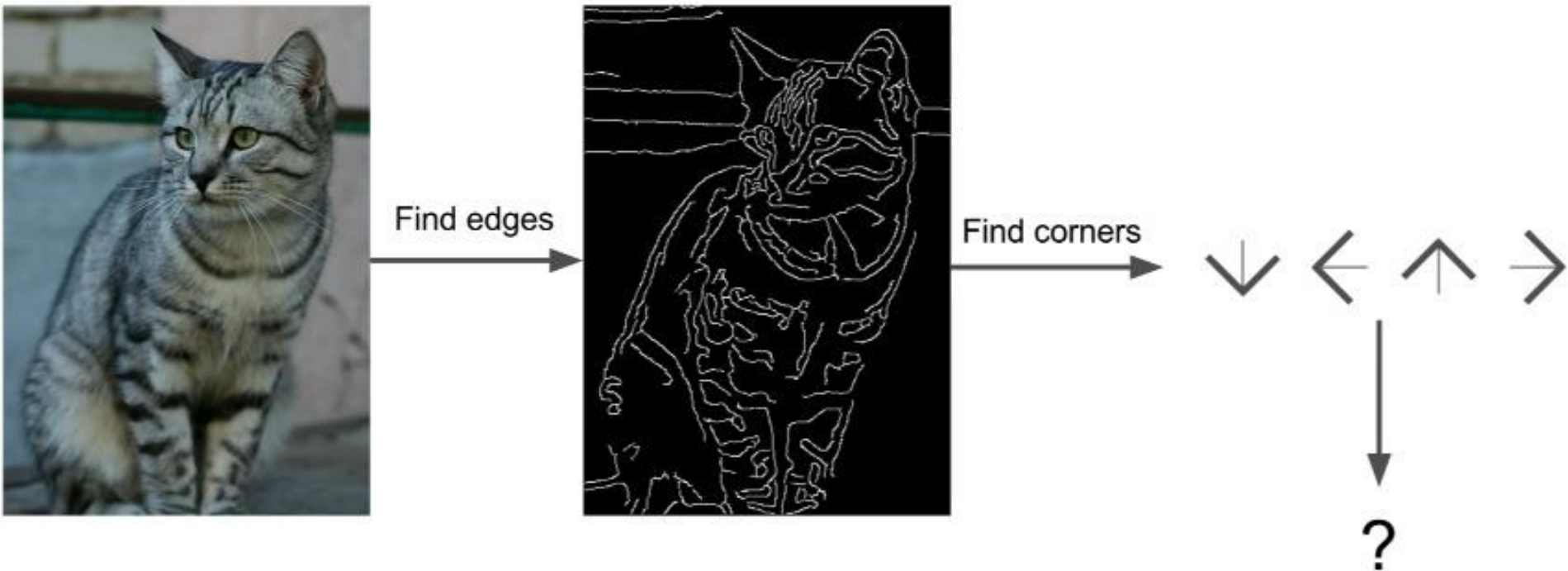
## Постановка проблемы

An image classifier

```
def classify_image(image):
    # Some magic here?
    return class_label
```

Unlike e.g. sorting a list of numbers,  
**no obvious way** to hard-code the algorithm for recognizing a cat, or other classes.

Attempts have been made



John Canny, "A Computational Approach to Edge Detection", IEEE TPAMI 1986



# ML в компьютерном зрении / CS231n

## Data-driven подход

### Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

```
def train(images, labels):
    # Machine learning!
    return model
```

```
def predict(model, test_images):
    # Use model to predict labels
    return test_labels
```

Example training set



### First classifier: Nearest Neighbor

```
def train(images, labels):
    # Machine learning!
    return model
```

Memorize all data and labels

```
def predict(model, test_images):
    # Use model to predict labels
    return test_labels
```

Predict the label of the most similar training image

### Nearest neighbors

### Distance Metric to compare images

**L1 distance:**  $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

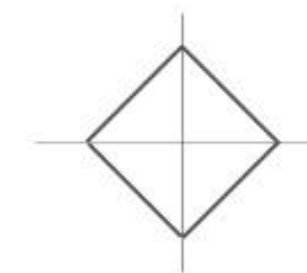
test image	training image	pixel-wise absolute value differences
56 32 10 18	10 20 24 17	46 12 14 1
90 23 128 133	8 10 89 100	82 13 39 33
24 26 178 200	12 16 178 170	12 10 0 30
2 0 255 220	4 32 233 112	2 32 22 108

add → 456

### K-Nearest Neighbors: Distance Metric

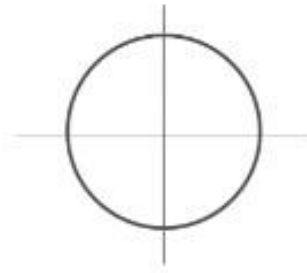
**L1 (Manhattan) distance**

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



**L2 (Euclidean) distance**

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



### Distance metrics

L1

L2

### Code implementation

```
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred
```

### Nearest Neighbor classifier

**Q:** With N examples, how fast are training and prediction?

**A:** Train O(1), predict O(N)

This is bad: we want classifiers that are **fast** at prediction; **slow** for training is ok

### K-Nearest Neighbors: Distance Metric

**L1 (Manhattan) distance**

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



K = 1

**L2 (Euclidean) distance**

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



K = 1

### K-Nearest Neighbors

Instead of copying label from nearest neighbor, take **majority vote** from K closest points



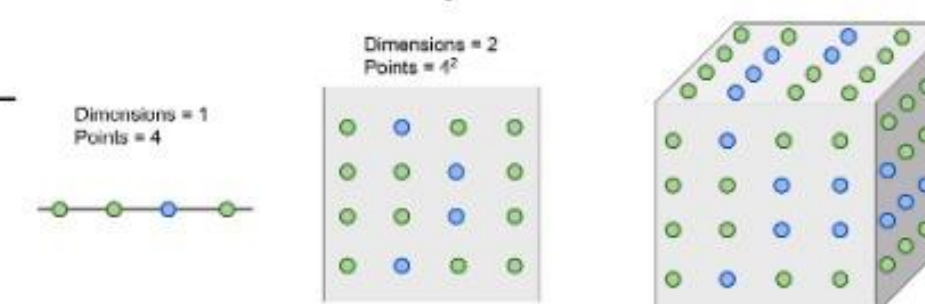
K = 1

K = 3

K = 5

### k-Nearest Neighbor on images **never used**.

- Curse of dimensionality





# ML в компьютерном зрении / CS231n

## Гиперпараметры

### Hyperparameters

What is the best value of **k** to use?  
What is the best **distance** to use?

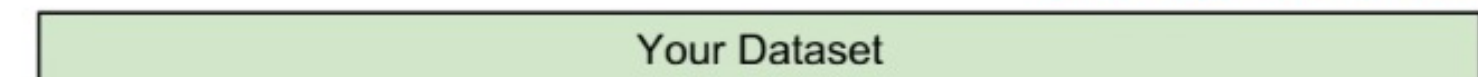
These are **hyperparameters**: choices about the algorithm that we set rather than learn

Very problem-dependent.  
Must try them all out and see what works best.

### Setting Hyperparameters

**Idea #1:** Choose hyperparameters that work best on the data

**BAD:** K = 1 always works perfectly on training data



**Idea #2:** Split data into **train** and **test**, choose hyperparameters that work best on test data

**BAD:** No idea how algorithm will perform on new data

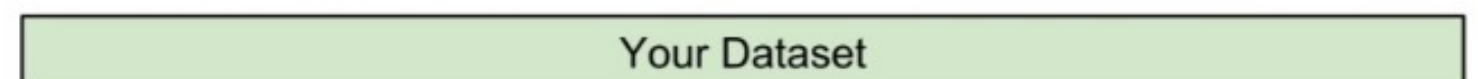


**Idea #3:** Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

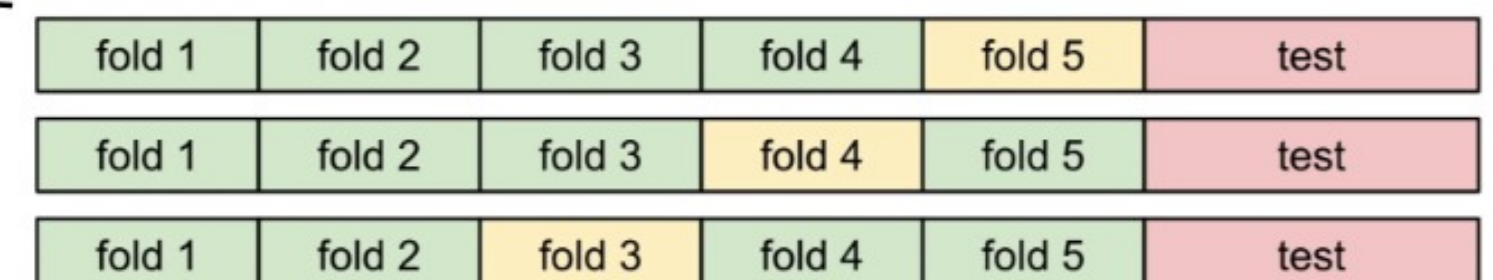
**Better!**



### Setting Hyperparameters



**Idea #4: Cross-Validation:** Split data into **folds**, try each fold as validation and average the results



Useful for small datasets, but not used too frequently in deep learning

miro

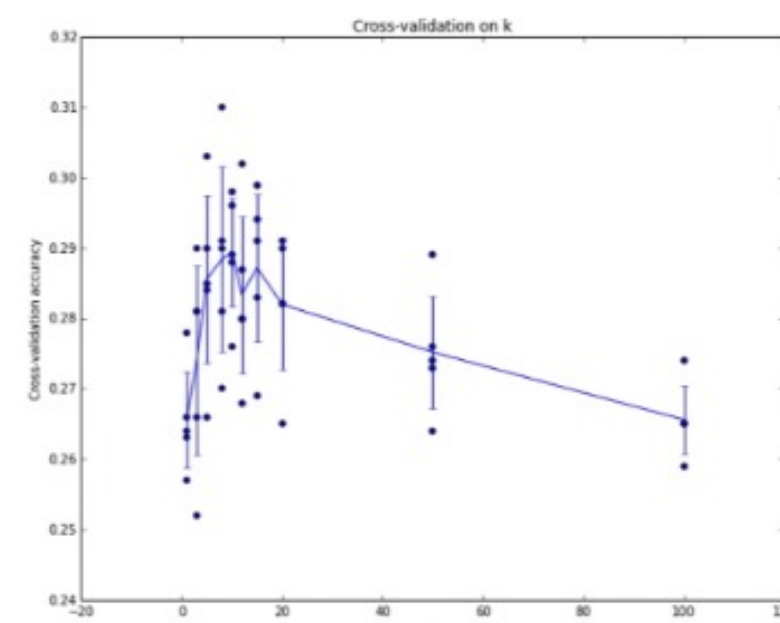
### 3. Hyperparameters

Problem

Setting hyperparams

Comparing results

### Setting Hyperparameters



Example of 5-fold cross-validation for the value of **k**.

Each point: single outcome.

The line goes through the mean, bars indicated standard deviation

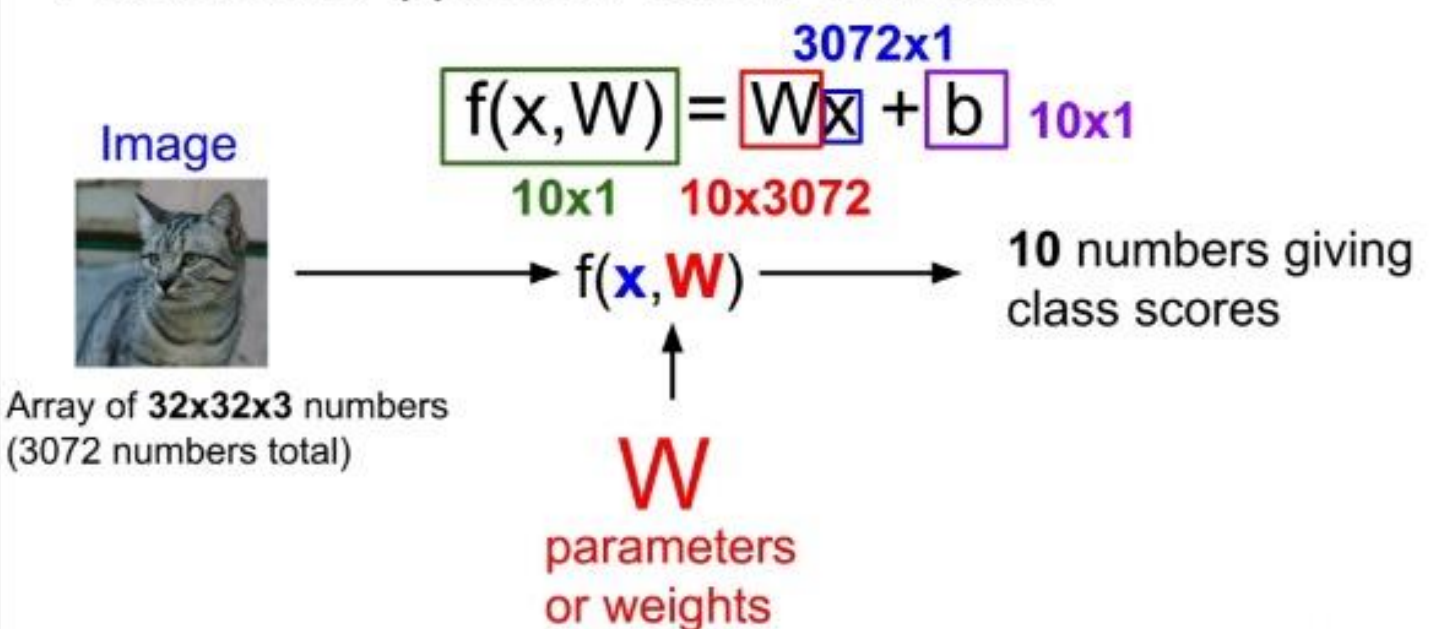
(Seems that  $k \sim 7$  works best for this data)



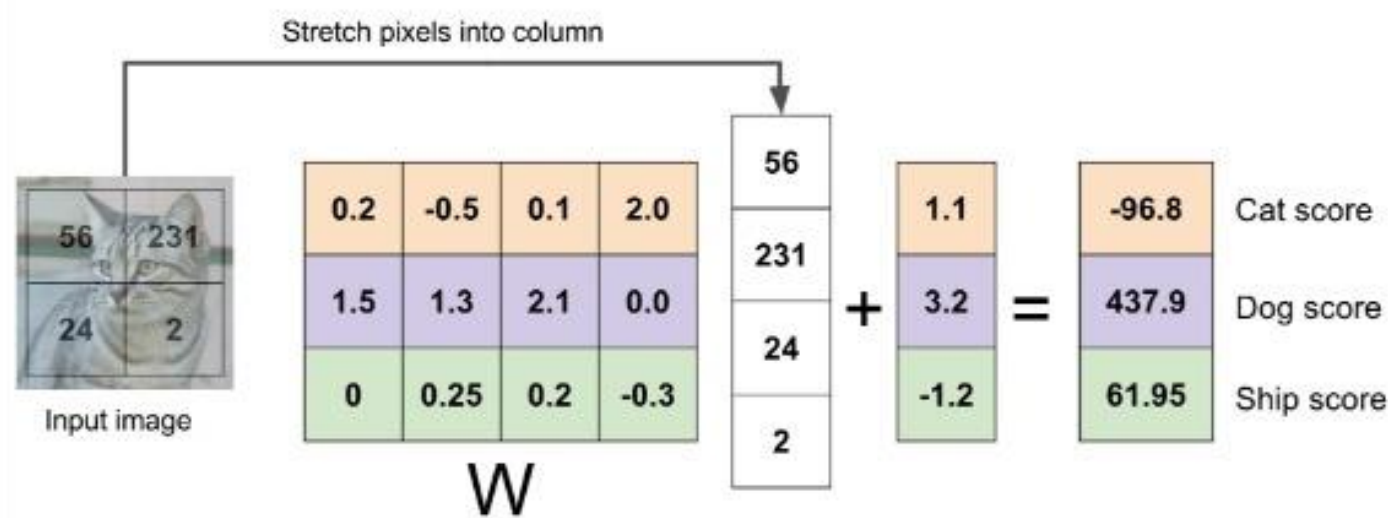
# ML в компьютерном зрении / CS231n

## Функциональный подход

### Parametric Approach: Linear Classifier

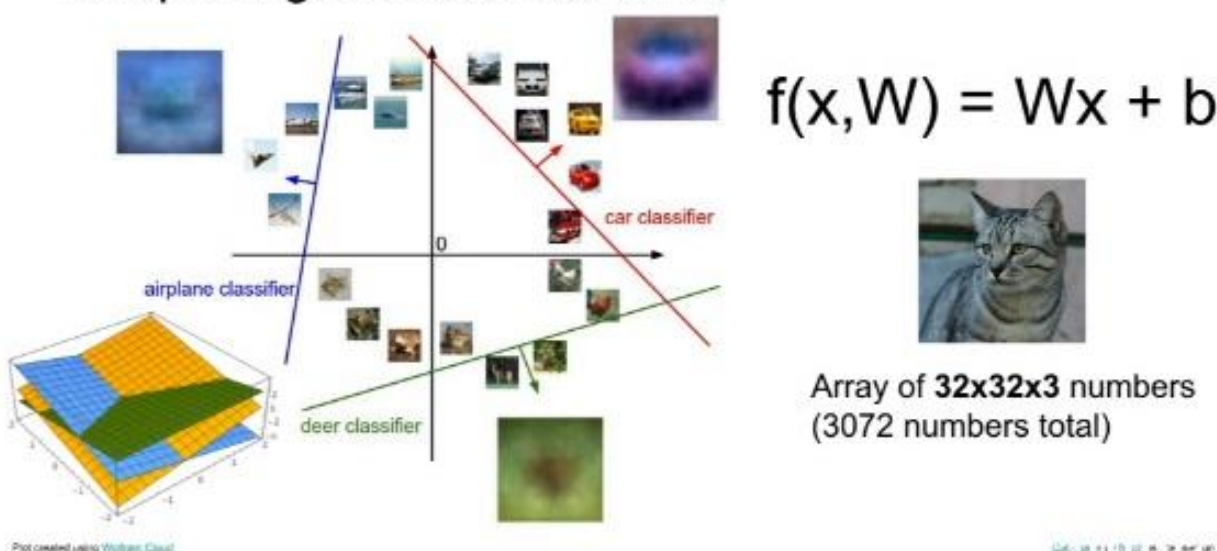


Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



Linear classifier

### Interpreting a Linear Classifier



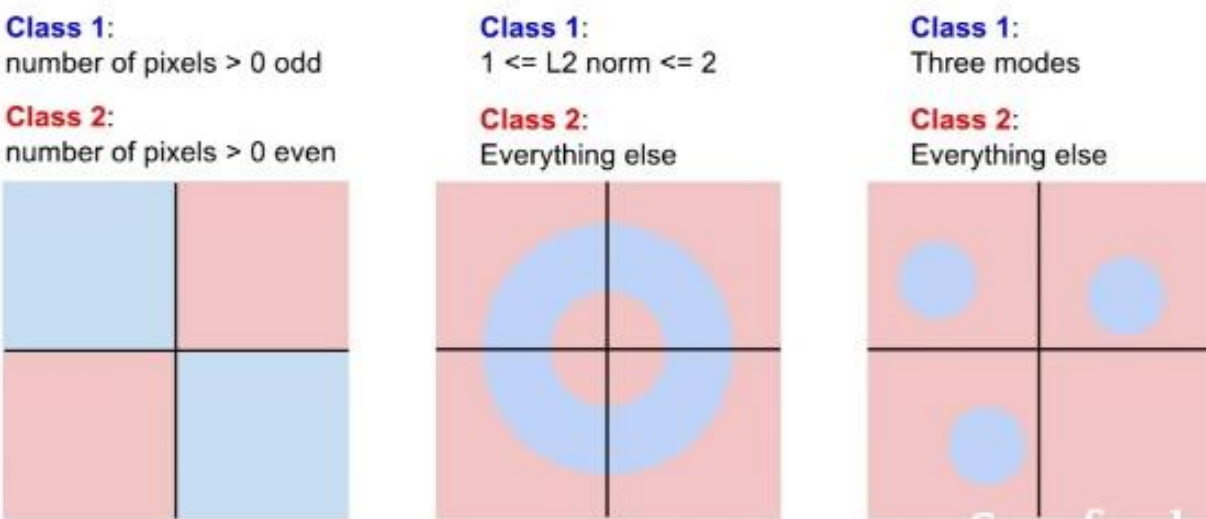
### Interpreting a Linear Classifier



Interpreting a linear classifier  
Template matching!

Hard cases for a linear classifier

Hard cases for a linear classifier





- Распознавание образов
- Использование локальных признаков
- Классификаторы по минимуму расстояния
- Статистически оптимальные классификаторы
- Нейронные сети
- Структурные методы распознавания
- + метод организации знаний на примере лекции курса CS231n

# Спасибо за внимание!

**Колокольников  
Георгий Андреевич**

Telegram: @Georg\_Bell

E-mail: [geokolok5@gmail.com](mailto:geokolok5@gmail.com)

Сайт: <https://github.com/GeorgBell>

## Использованные материалы:

- Гонсалес Р., Вудс Р. Цифровая обработка изображений. – М.: Техносфера, 2012. – 1104 с. – ISBN 978-5-94836-331-8.2.
- Курс лекций cs231n «Convolutional Neural Networks for Visual Recognition» (<http://cs231n.stanford.edu>).
- Курс лекций HSE «Deep Learning in Computer Vision» (<https://www.coursera.org/learn/deep-learning-in-computer-vision>)