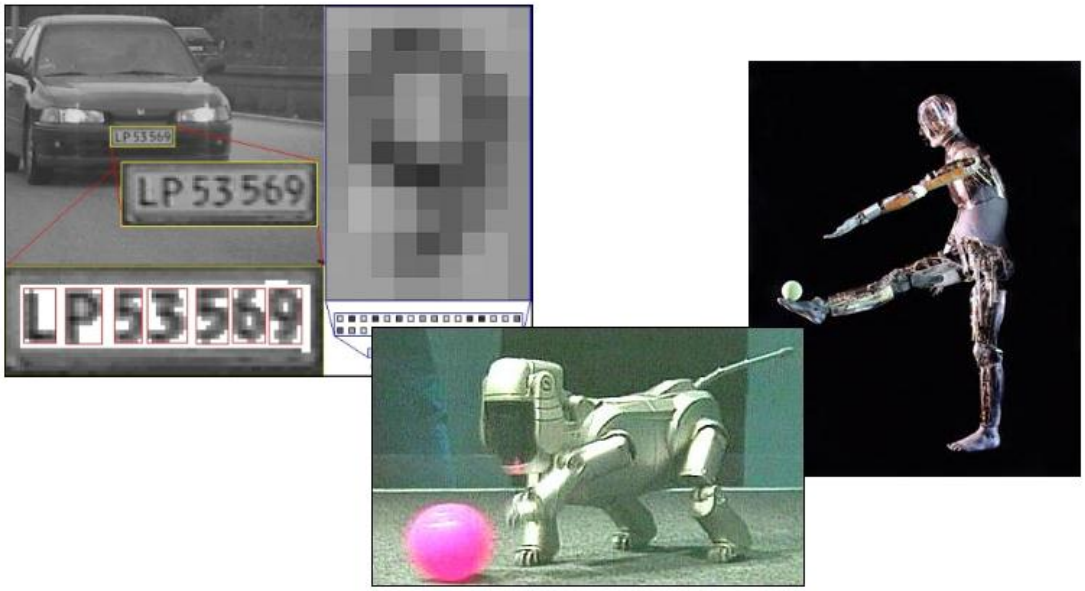


Распознавание образов

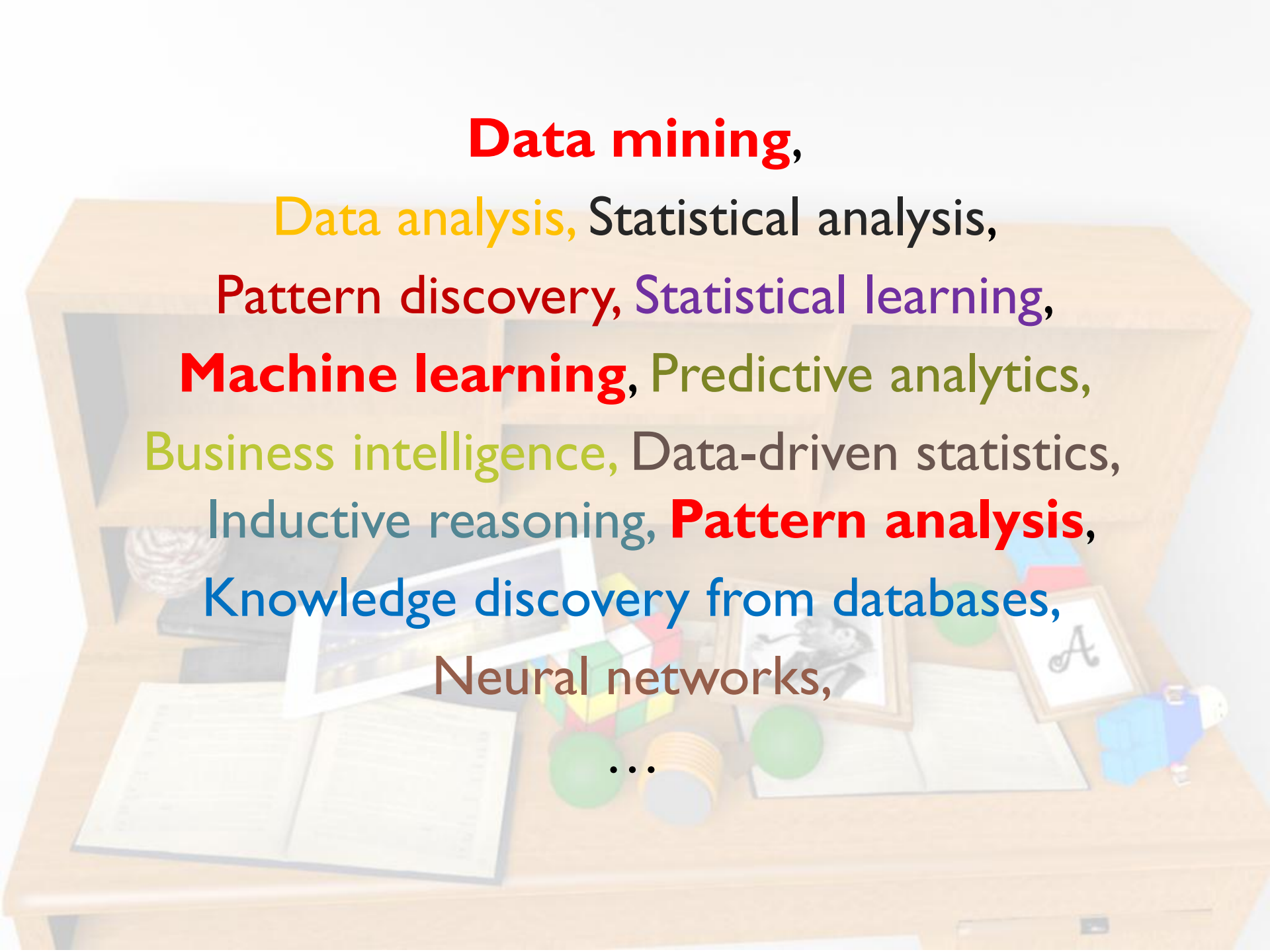
Константин Третьяков
http://kt.era.ee

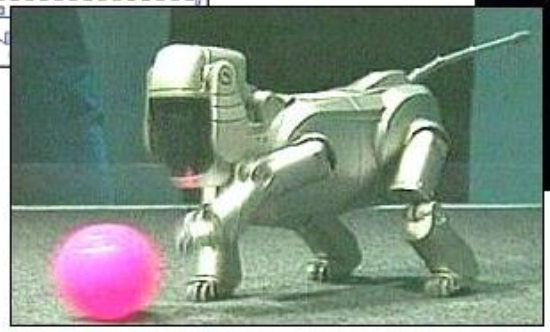
DevClub
27.04.2011

STACC Software Technology and Applications Competence Center

BIT IT

TARTU ÜLIKOOL 1632 UNIVERSITAS TARTUENSIS


Data mining,
Data analysis, Statistical analysis,
Pattern discovery, Statistical learning,
Machine learning, Predictive analytics,
Business intelligence, Data-driven statistics,
Inductive reasoning, **Pattern analysis**,
Knowledge discovery from databases,
Neural networks,
…



Thunderbird

Tools Help

Reply Reply All Forward Delete Junk Print

View: All

Subject

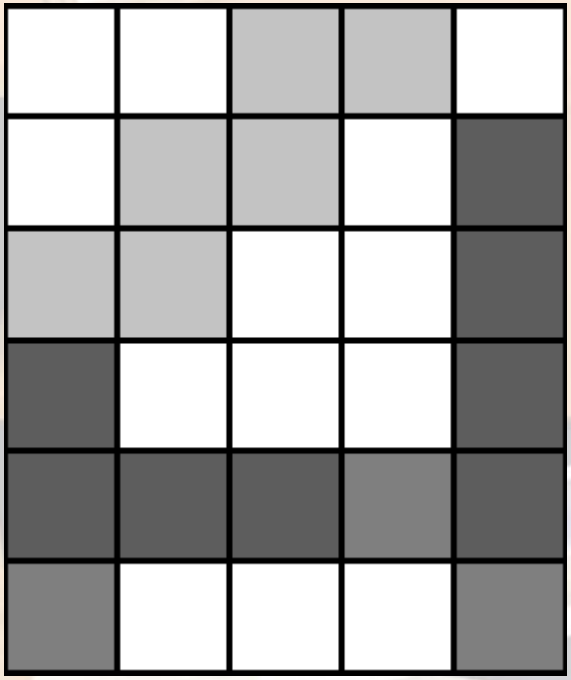
Spam: BE like a normal MAN

Google translate

NETFLIX

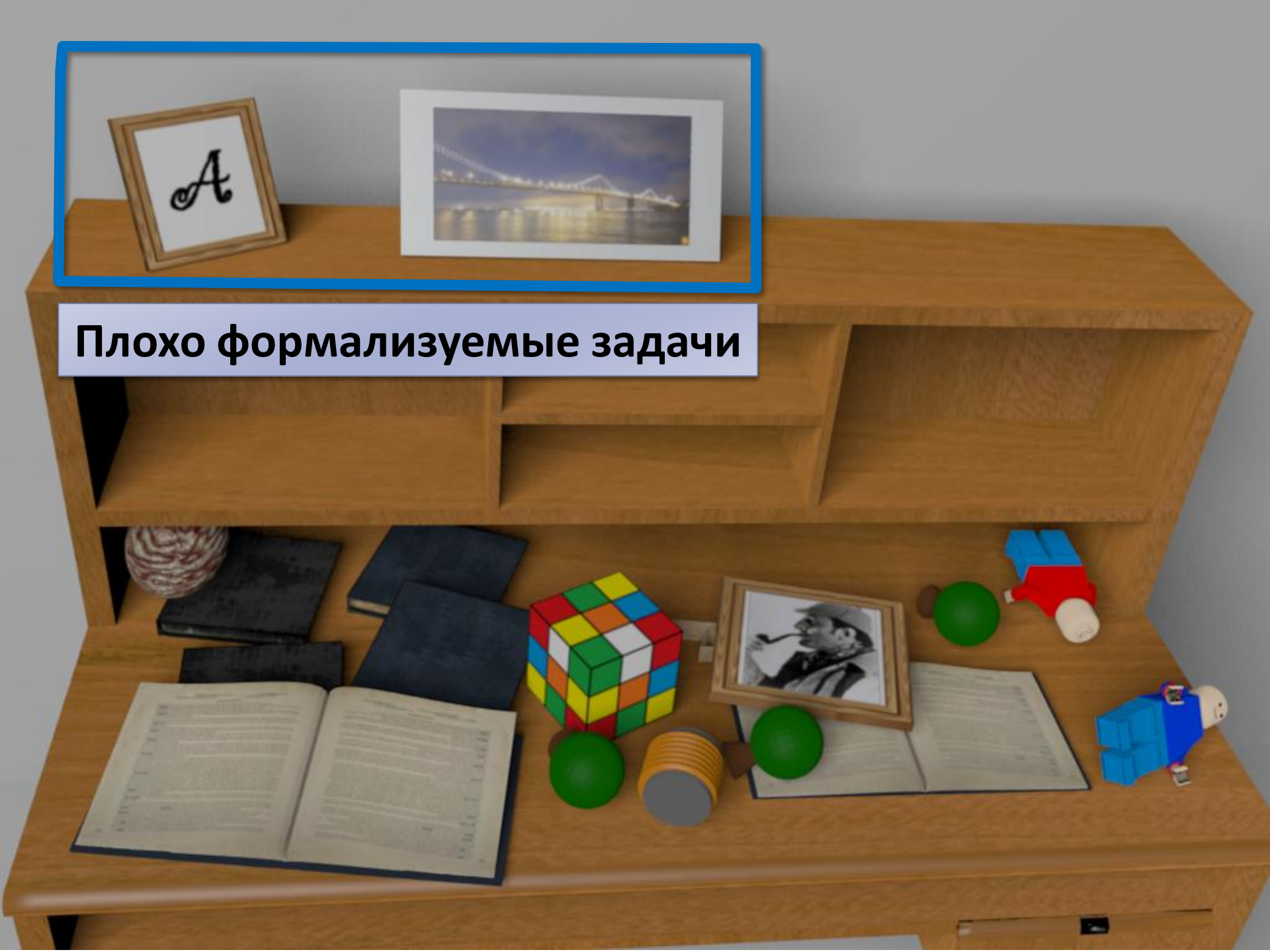

A


A



A

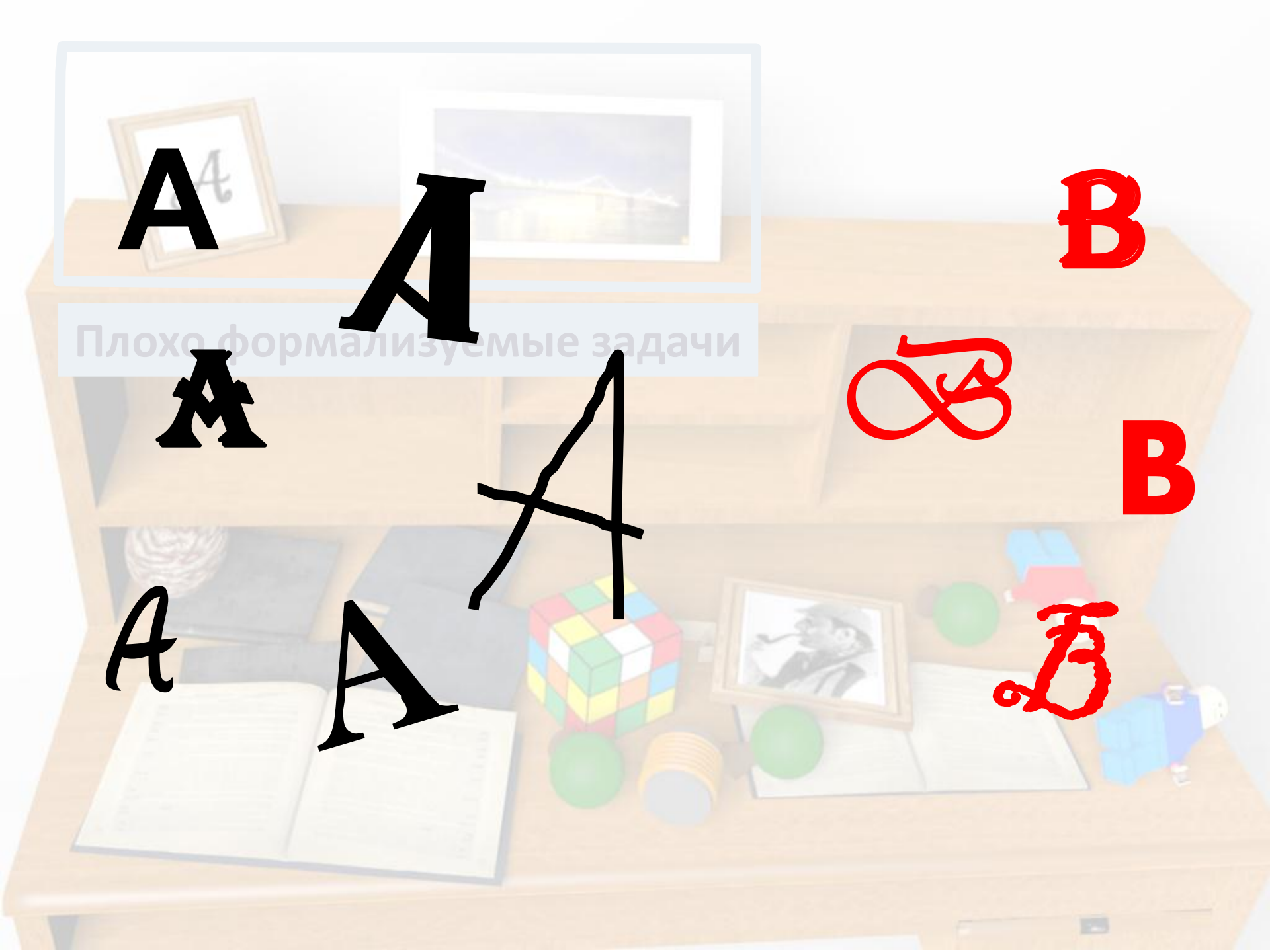

A



Плохо формализуемые задачи

Плохо формализуемые задачи

Плохо формализуемые задачи




A


A

- Общее правило:

  ЕСЛИ (X сделан из пластмассы),

  ТО (X несъедобен)

- Применение в частном случае:

  X = Кубик Рубика

  ⇒

  Кубик Рубика несъедобен

- Общее правило:

  add(x, y) = function { … }

- Применение в частном случае:

  add(2,4)

- Общее правило:

```
bool has_access(x) {
        return x.authenticated();
}
```

- Применение в частном случае:

```
if (has_access(John)) {
        …
}
```

- Общее правило:

```
bool has_access(x) {
        ?
}
```

- Частные случаи:

has_access(John) = True

has_access(Jane) = True

has_access(Mike) = False

…

- Общее правило:

```
bool has_access(x) {
        ?
}
```

- Частные случаи:

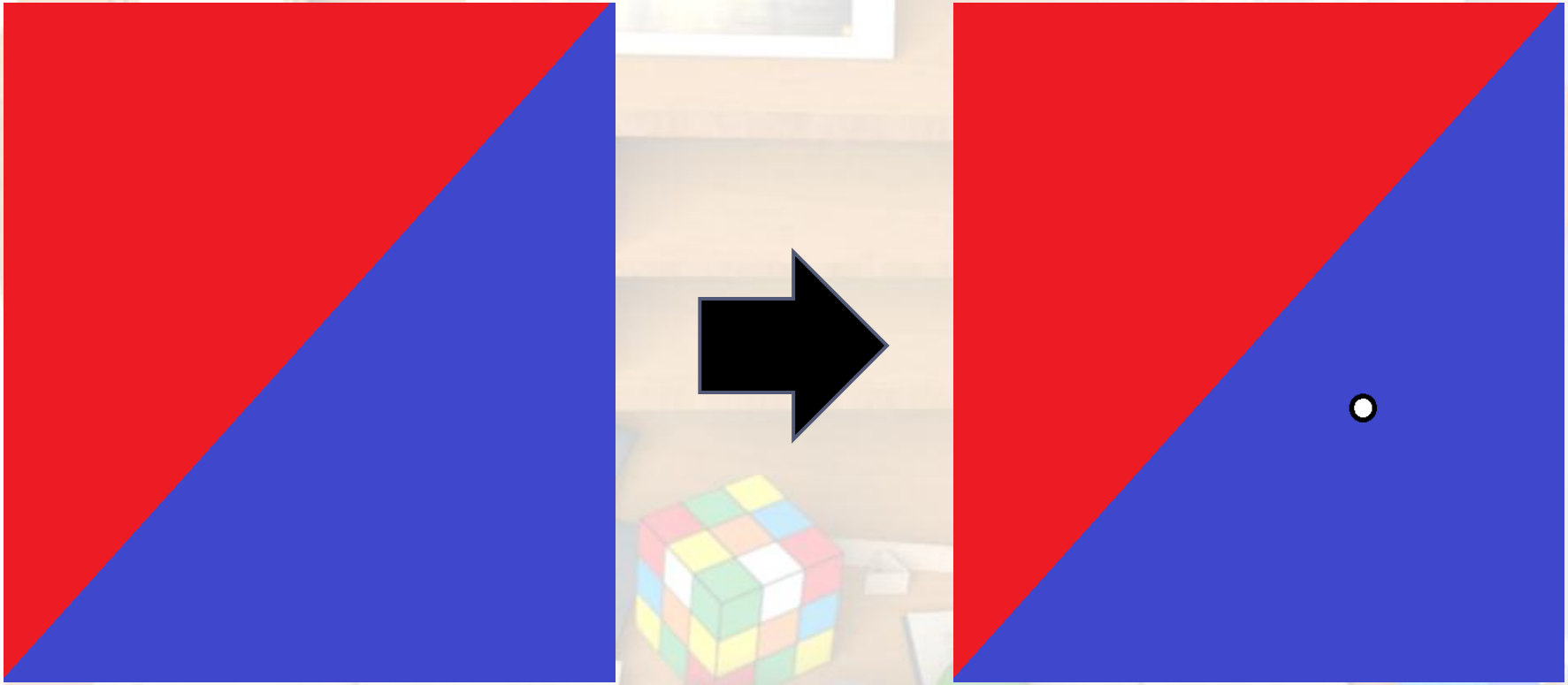
has_access(John) = True

has_access(Jane) = True

has_access(Mike) = False

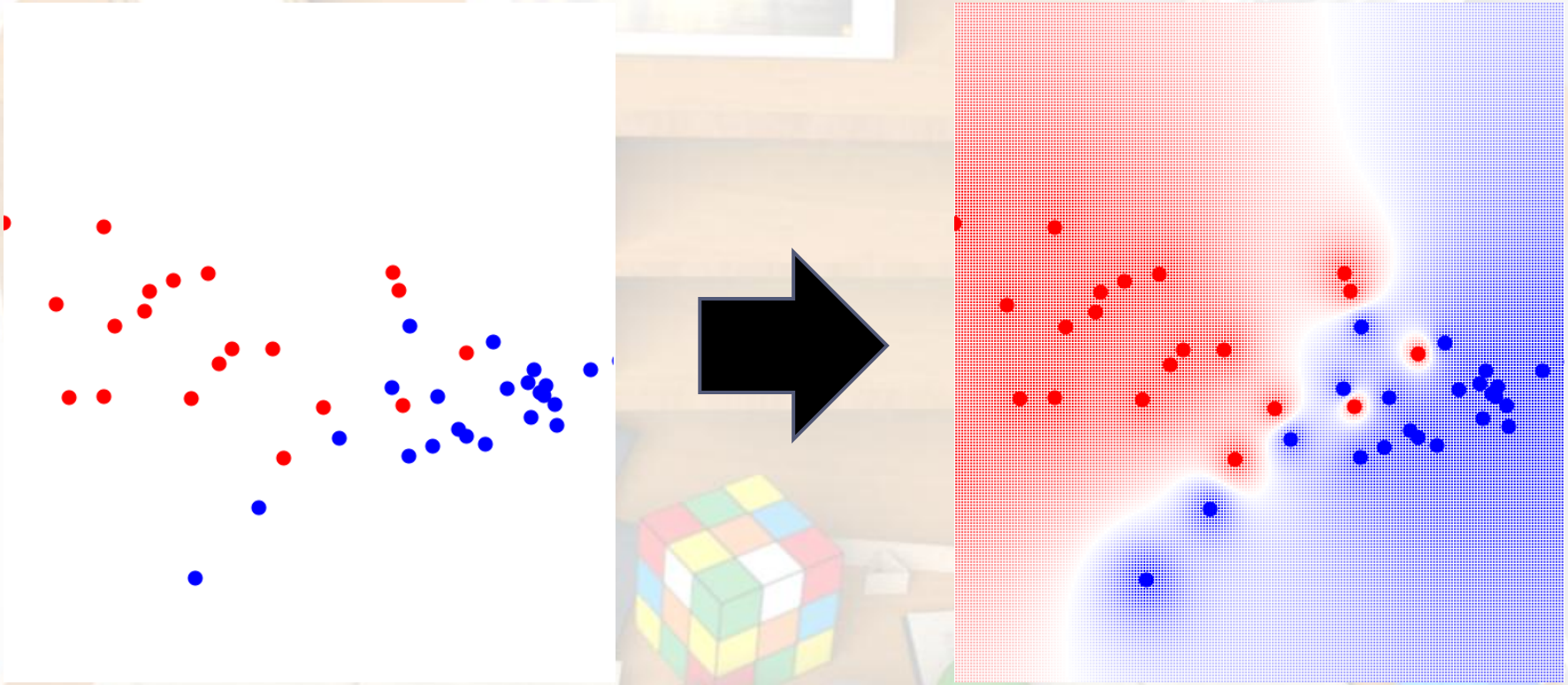
…

Дедукция

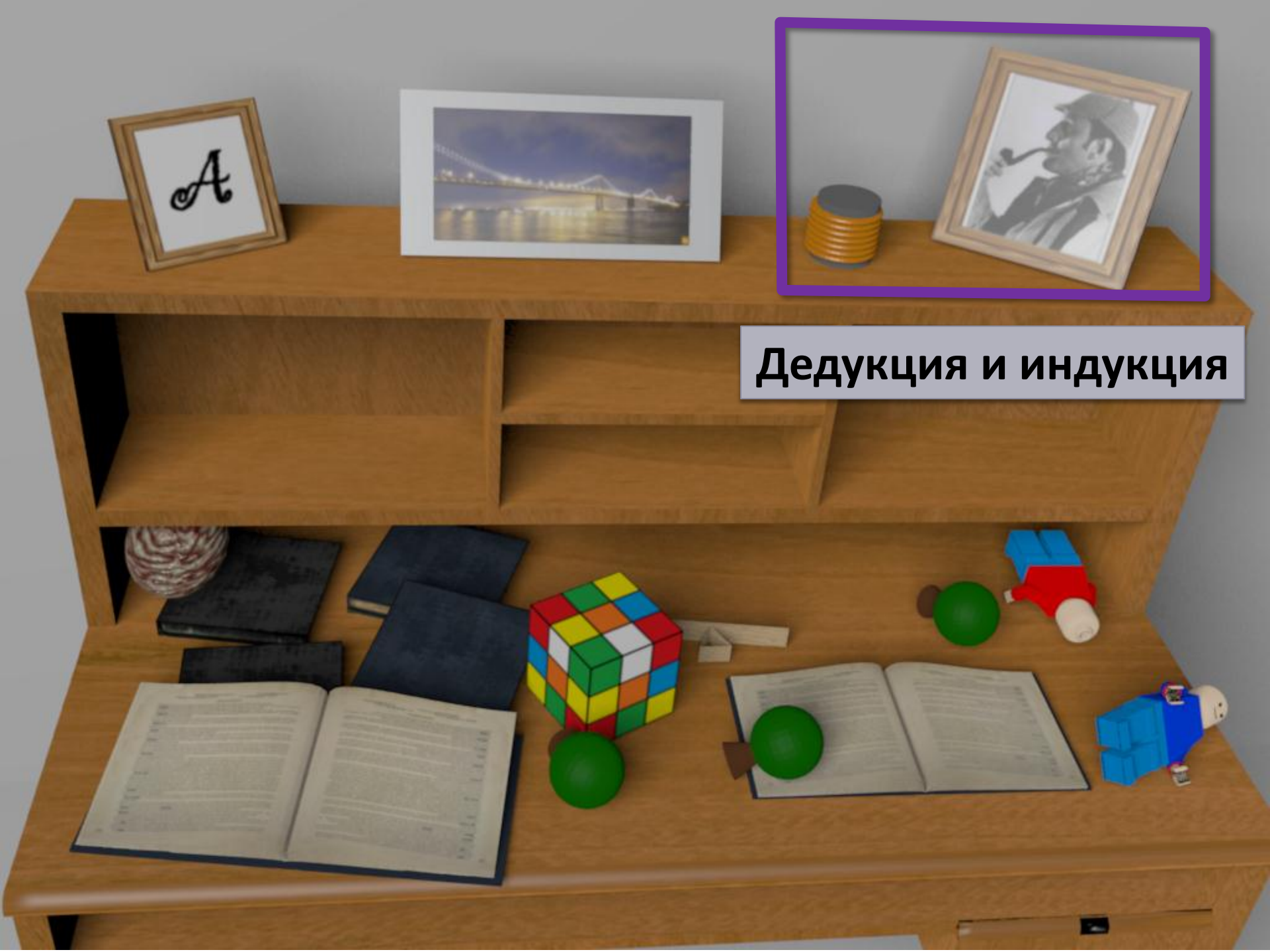


Известно общее правило, нужно принять решение в частном случае

Индукция



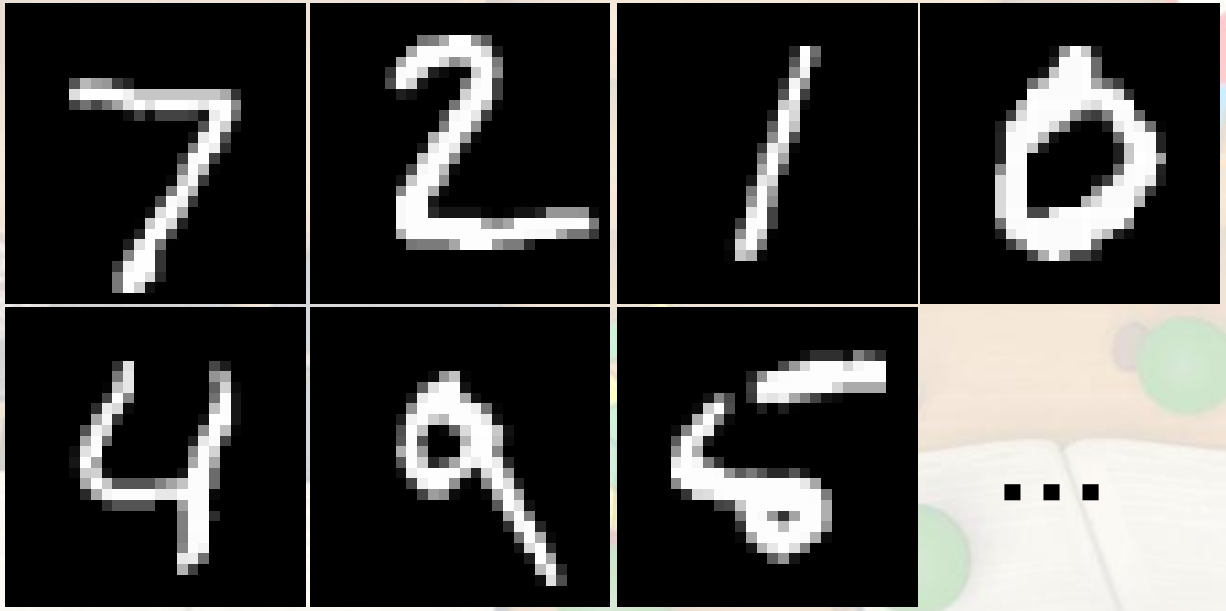
Известны частные случаи, нужно найти общее правило


A
Дедукция и индукция


Классификация
по аналогии

MNIST dataset

- http://yann.lecun.com/exdb/mnist/
- Картинки рукописных цифр, 28 x 28



MNIST dataset

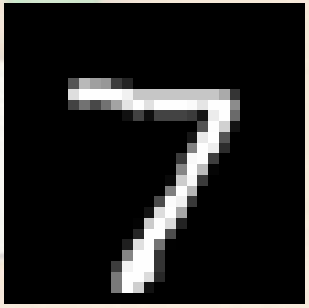
```
from PIL import Image
from glob import glob
from numpy import array

def load_image(filename):
    img = Image.open(filename)
    return array(
        [ord(char) for char in img.tostring()]
    )
```

MNIST dataset

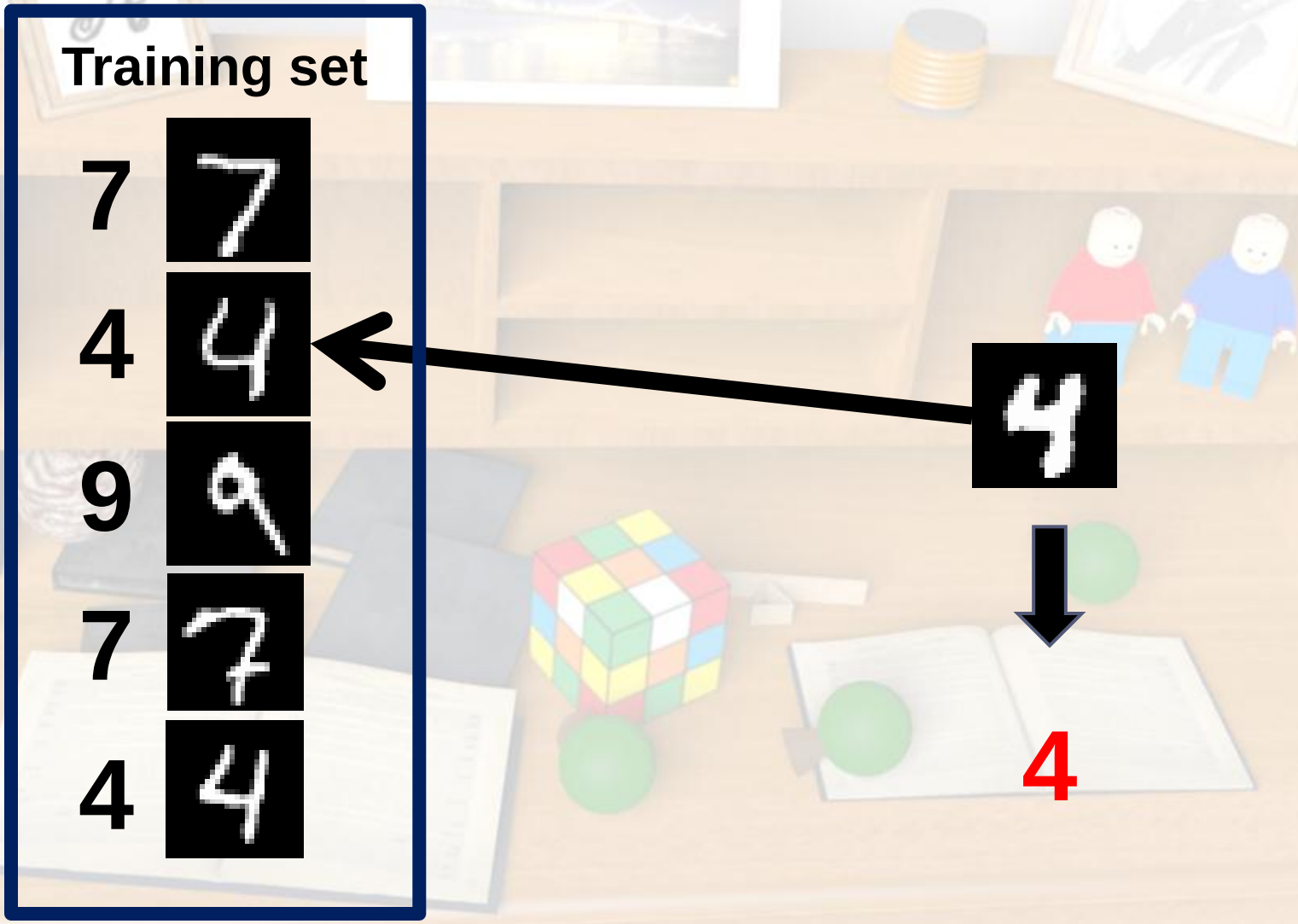
```
# 10 000 файлов *.png
files = glob("*.png")

# Загружаем данные
images = [load_image(file) for file in files]

# 102.1.png -> "1" (класс)
labels = [file.split(".")[1] for file in files]

# Возьмем 1000 картинок
training_set    = images[0:1000]
training_labels = labels[0:1000]
```

MNIST dataset

```
> training_set[0]

  array([ 0, 0, 0, ...,
          254, 241, 198, ...])

> training_labels[0]

  '7'
```



Метод ближайшего соседа



Метод ближайшего соседа

```
def classify(img):
    similarities =
     [similarity(img, p) for p in training_set]
    i = similarities.index(max(similarities))
    return training_labels[i]
```

Метод ближайшего соседа

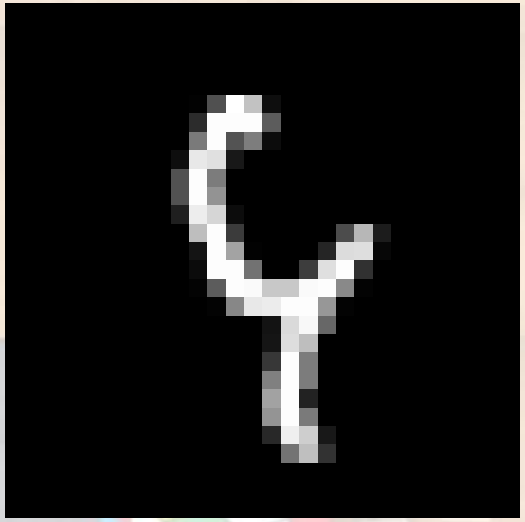
```
def classify(img):
    similarities =
     [similarity(img, p) for p in training_set]
    i = similarities.index(max(similarities))
    return training_labels[i]


def similarity(img1, img2):
     return -sum(abs(img1 - img2))
```

Проверим

```
test_set    = images[1000:2000]
test_labels = labels[1000:2000]


predicted_class = map(classify, test_set)


n_successes =
    sum(array(predicted_class) ==
                          array(test_labels))
```

=> 860/1000

9 или 4?

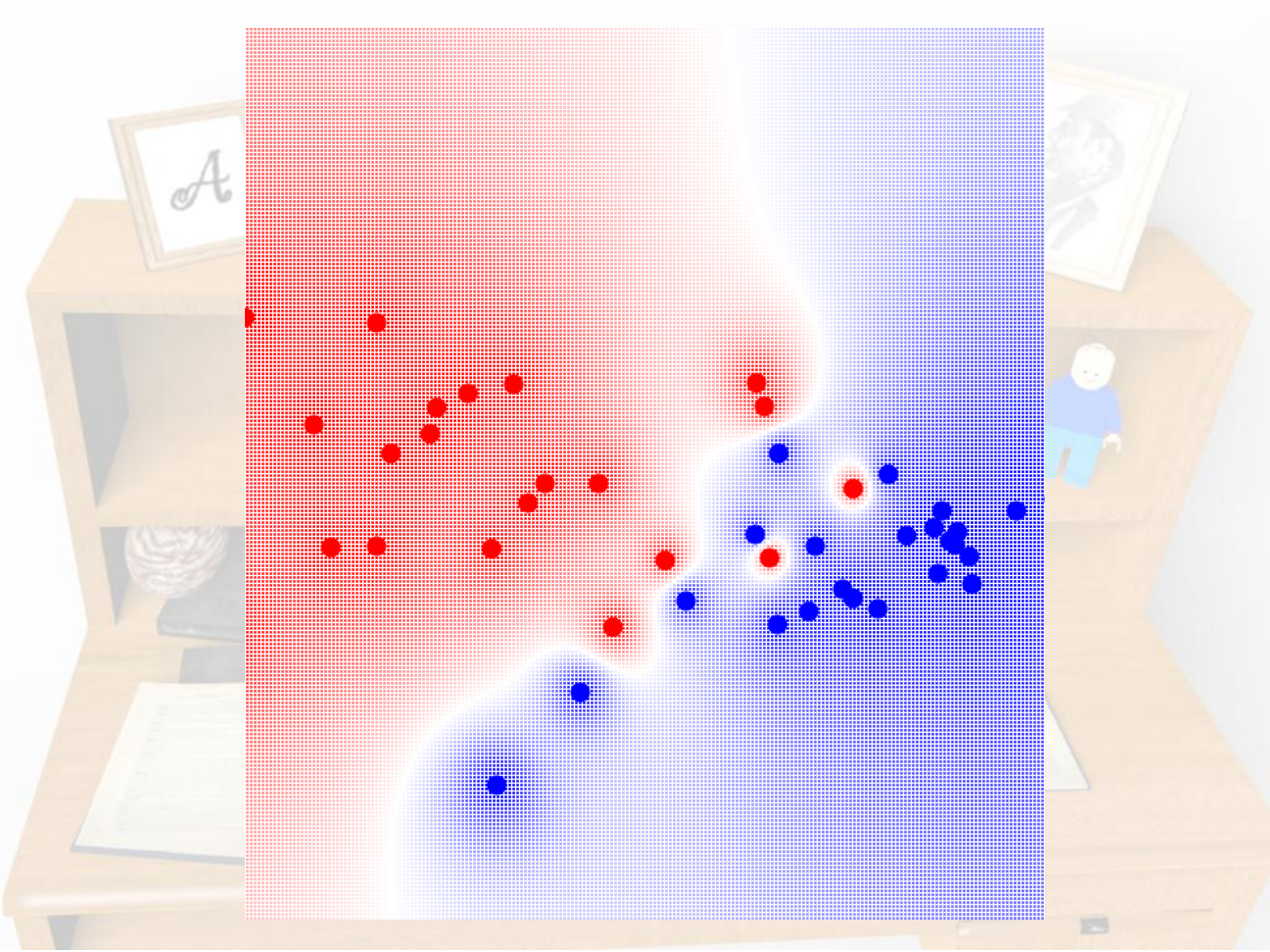


Scikit-learn

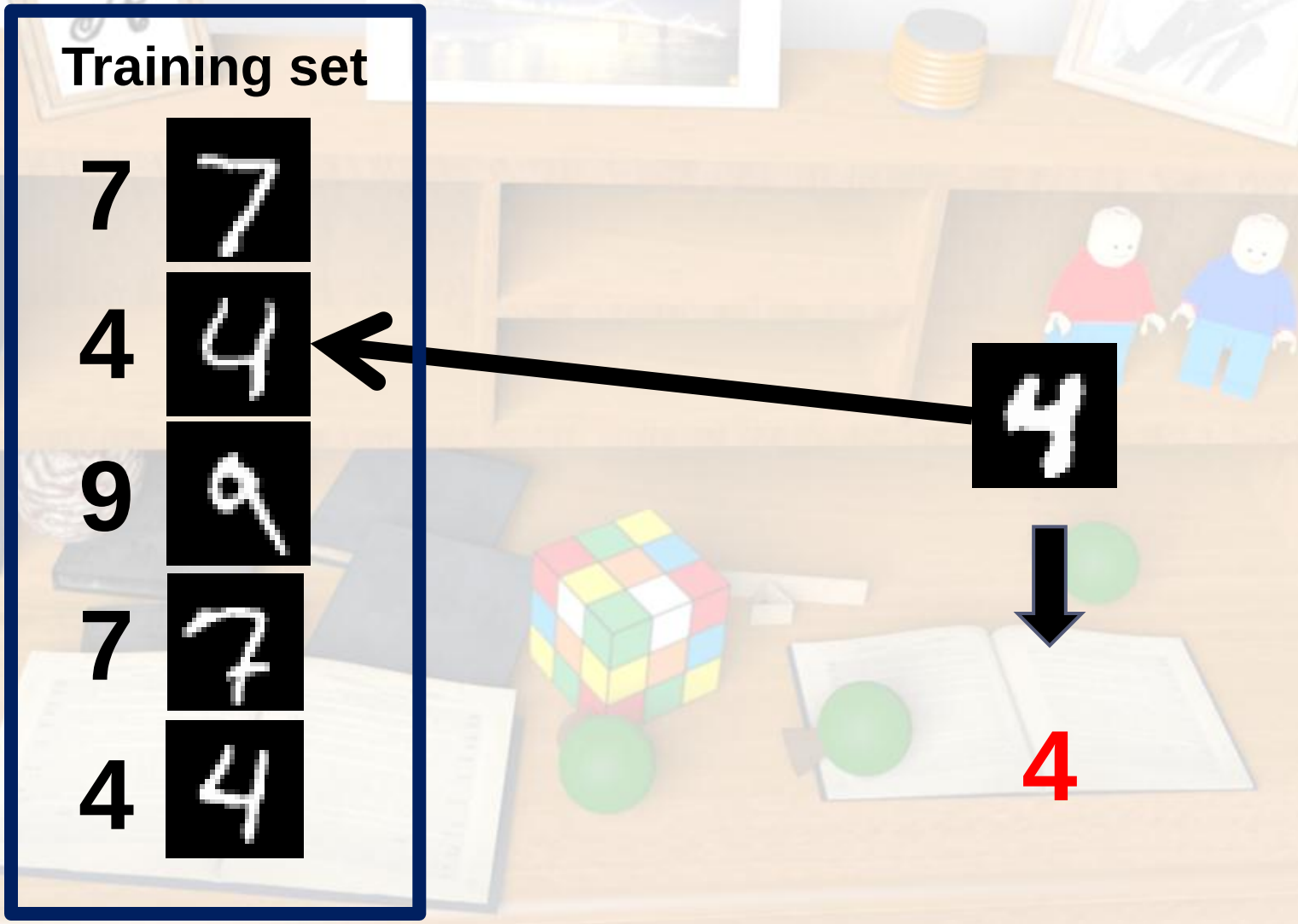
http://scikit-learn.sourceforge.net/

```
from scikits.learn.neighbors import
                              NeighborsClassifier

clf = NeighborsClassifier(n_neighbors=1)
clf.fit(training_set, training_labels)

predicted_class = clf.predict(test_set)
```

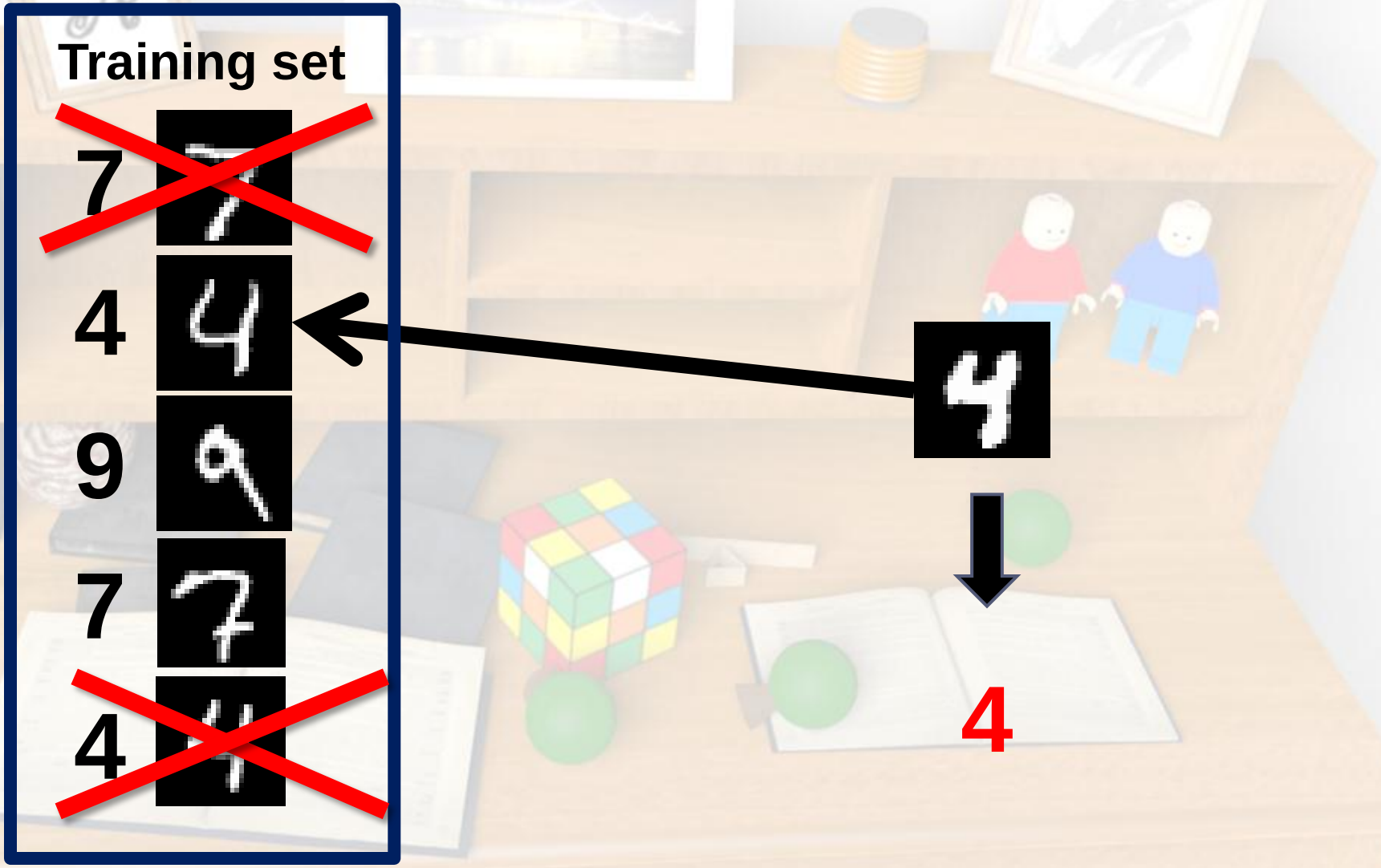
=> 869/1000



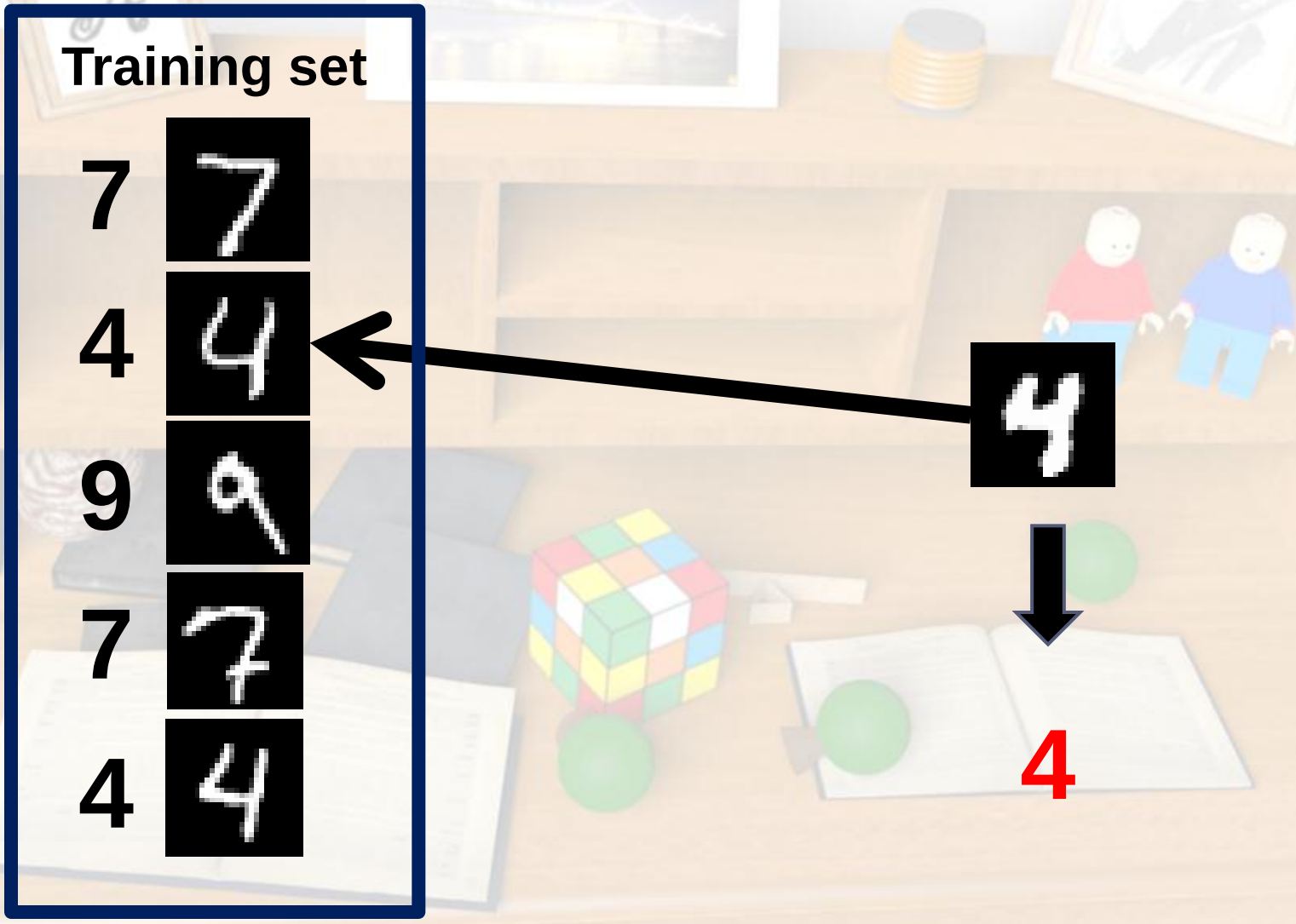
Метод ближайшего соседа



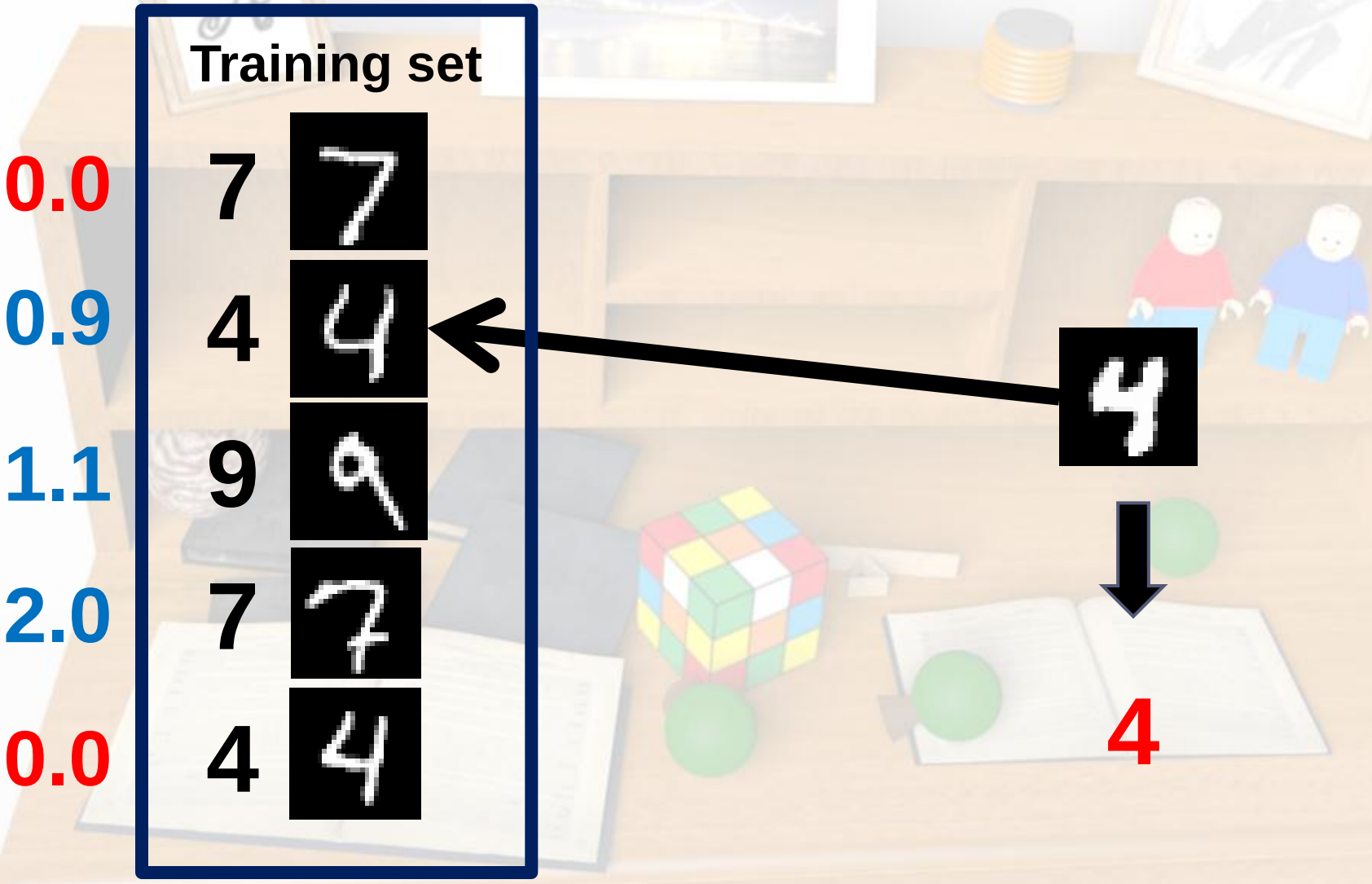
Некоторые примеры можно выкинуть



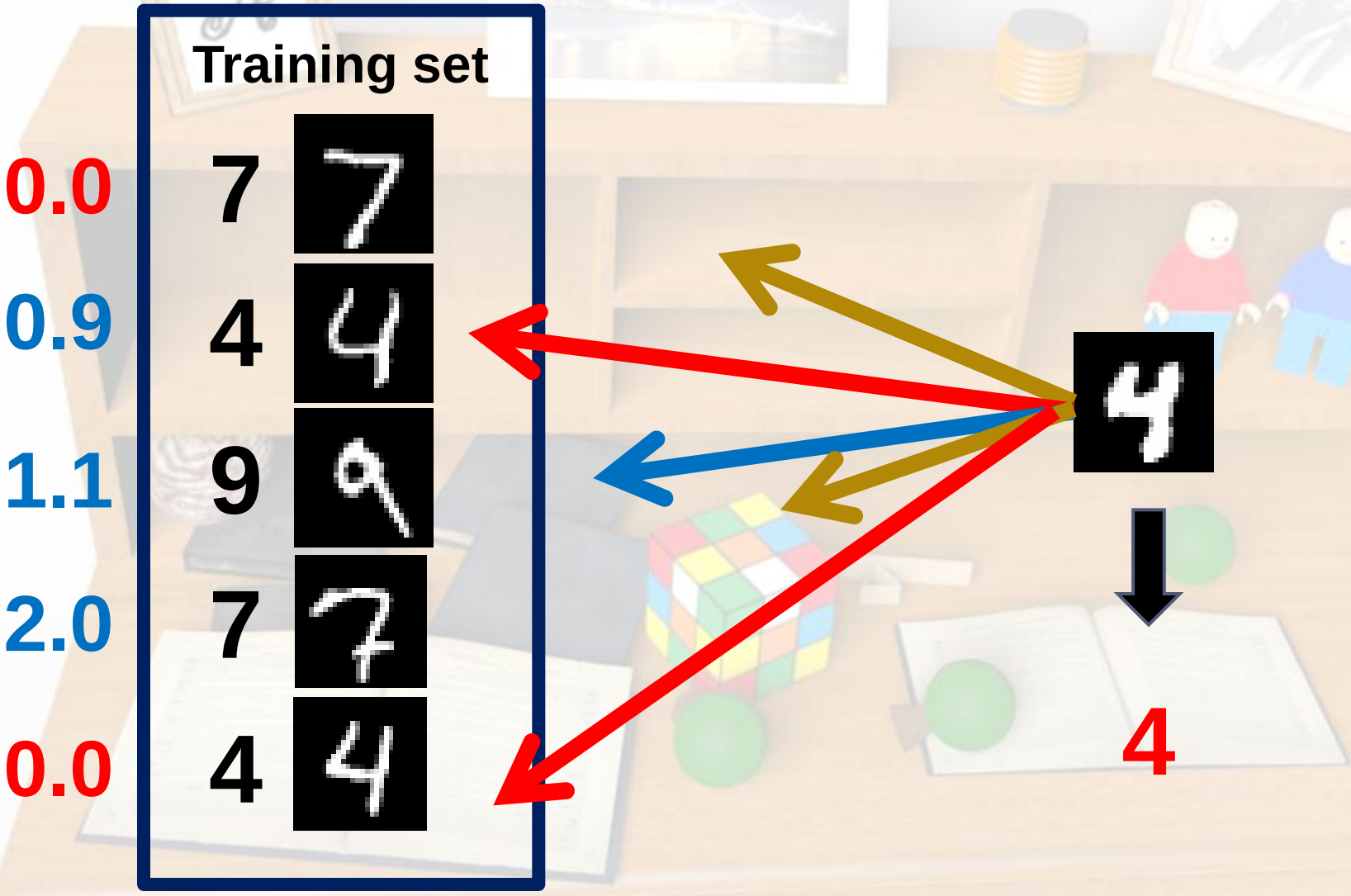
Еще хитрее – добавить «веса»



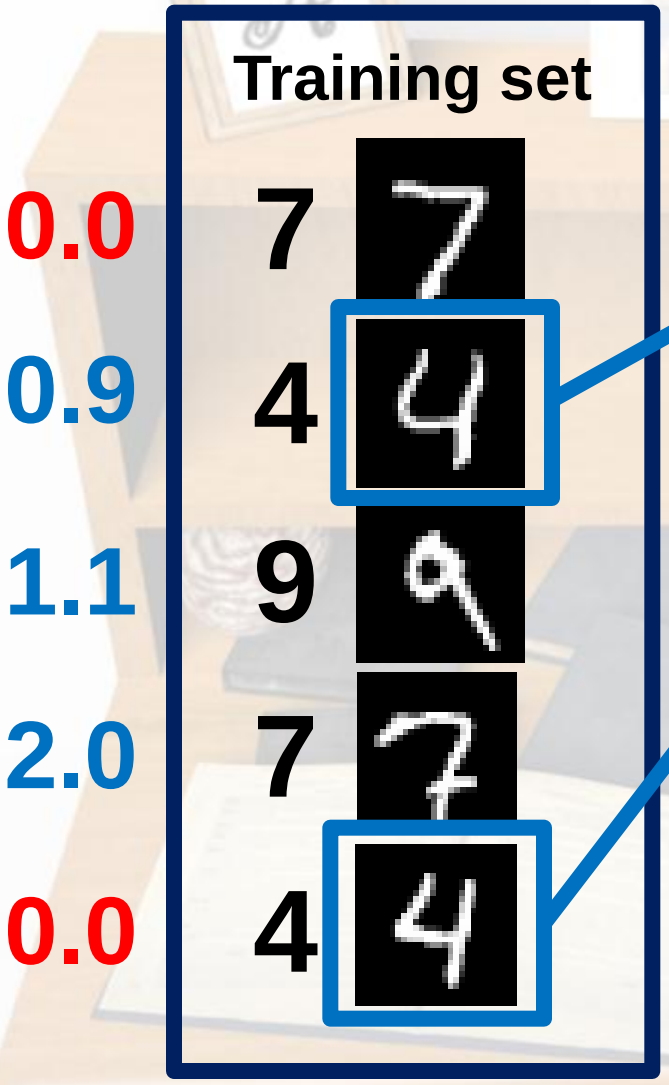
Еще хитрее – добавить «веса»



Вместо максимума, можно суммировать

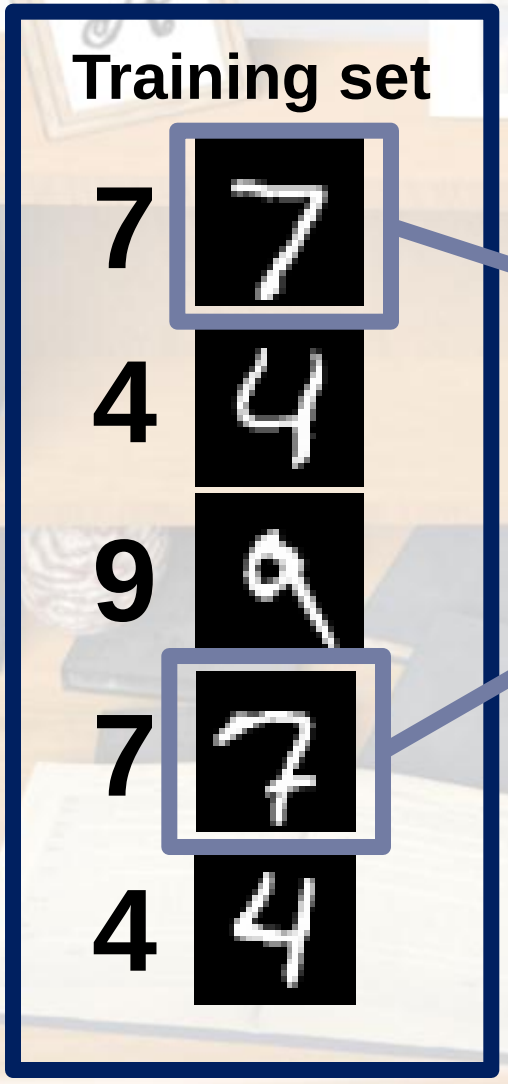


Вместо максимума, можно суммировать

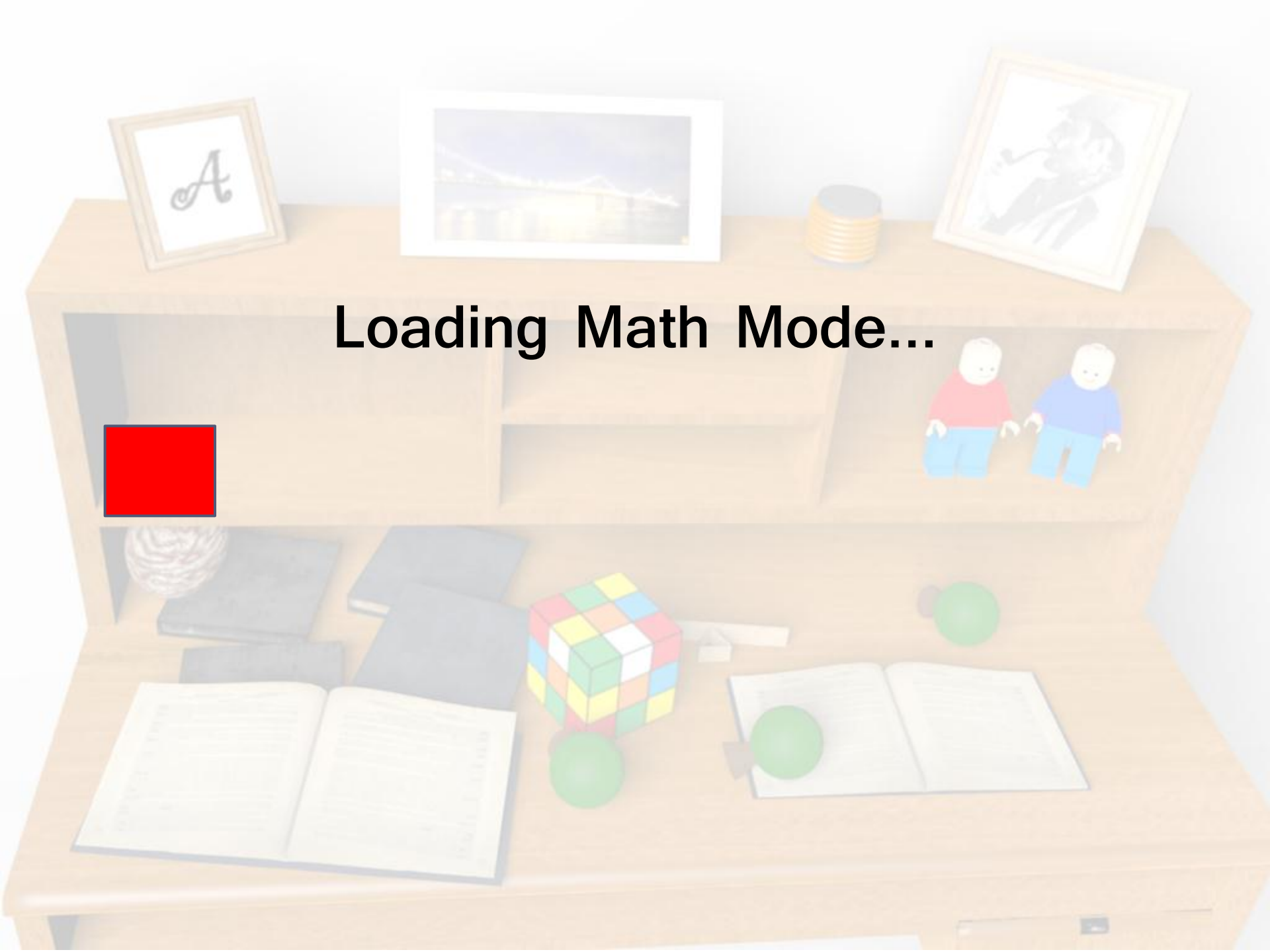


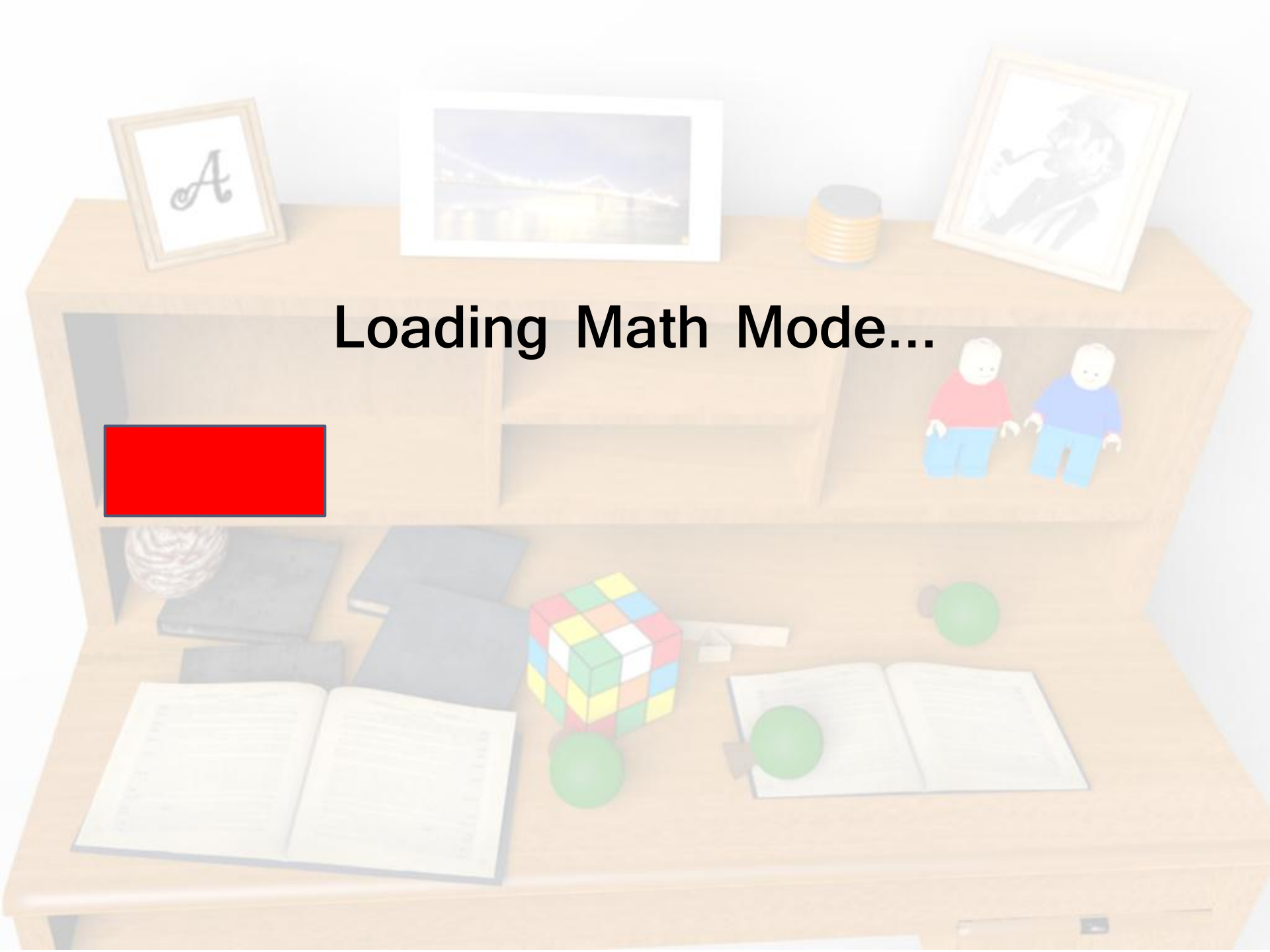

```
evidence(img, 4) =

0.9 * similarity(img, tr[1])

+

0.0 * similarity(img, tr[4])

= 34.0
```

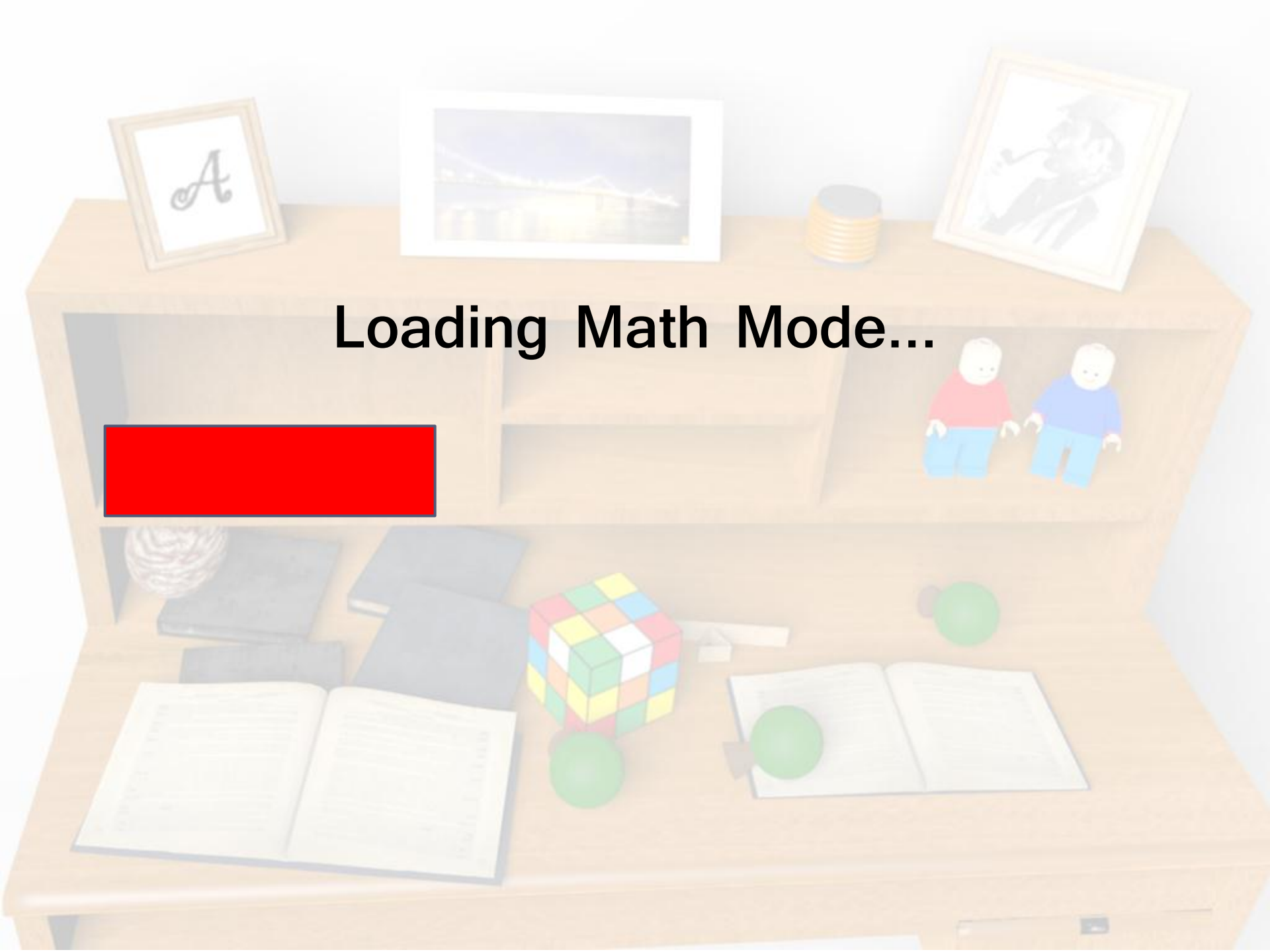
Вместо максимума, можно суммировать

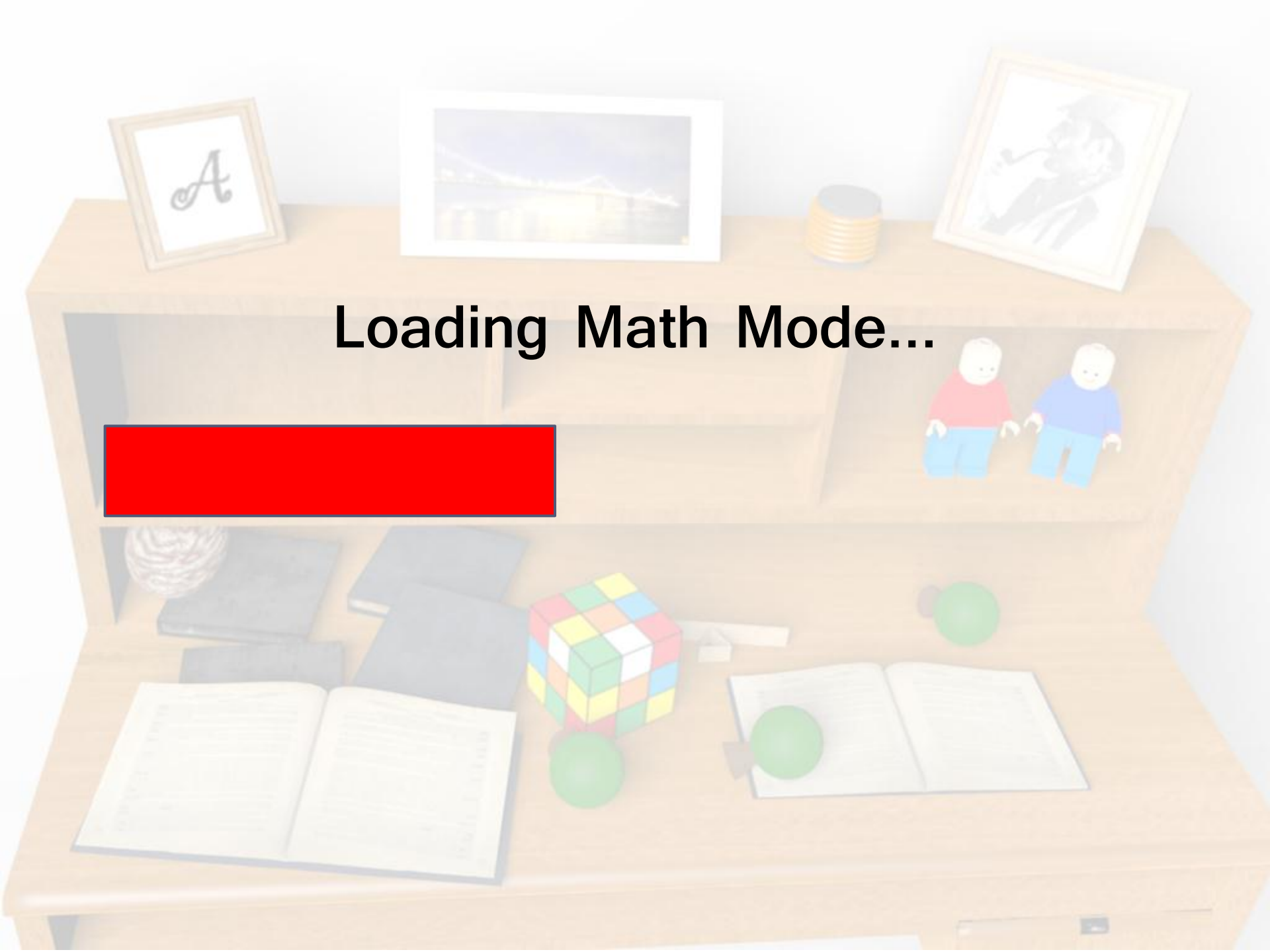


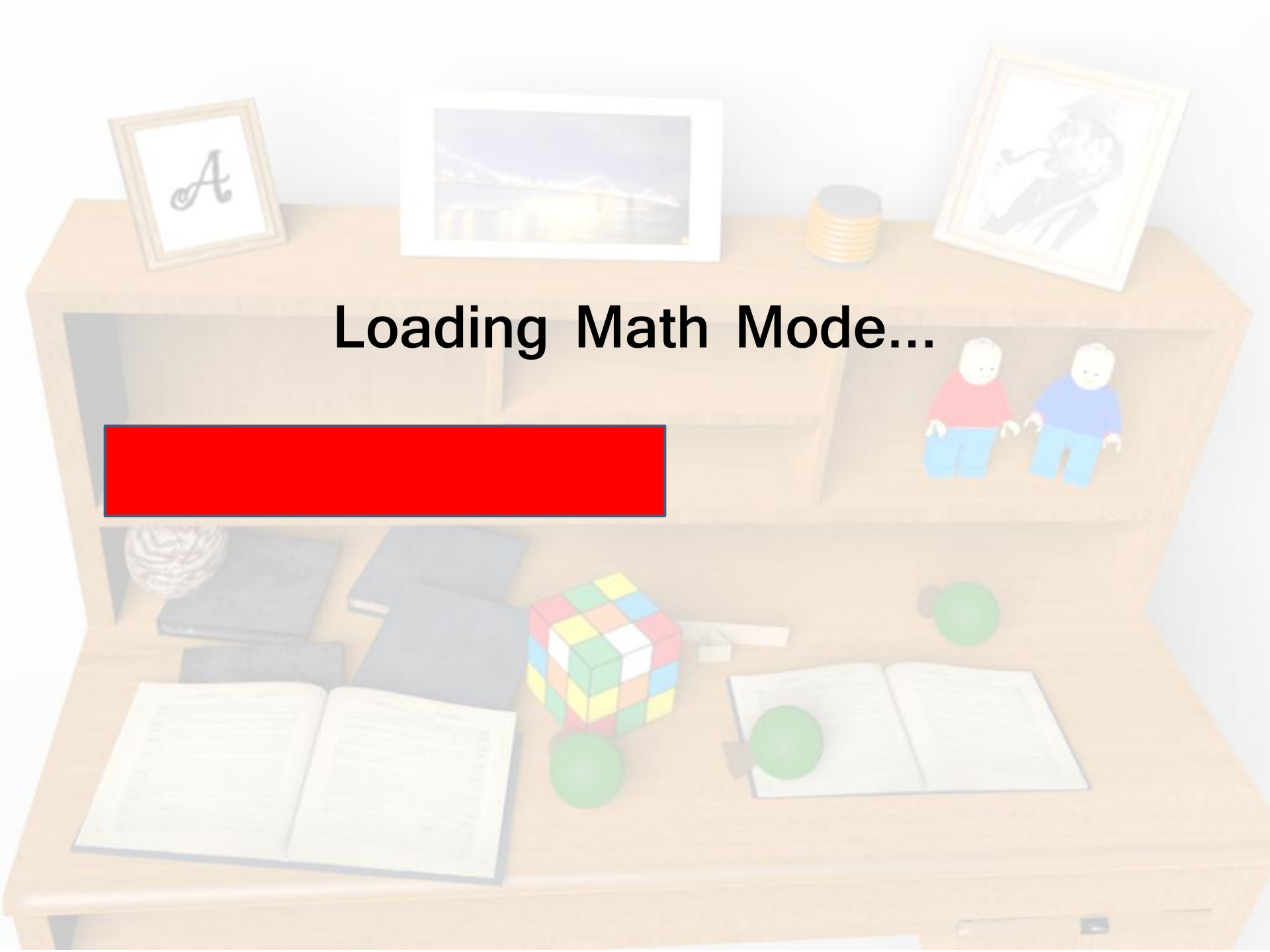

```
evidence(img, 7) =

0.0 * similarity(img, tr[0])

+

2.0 * similarity(img, tr[3])

= 20.0
```

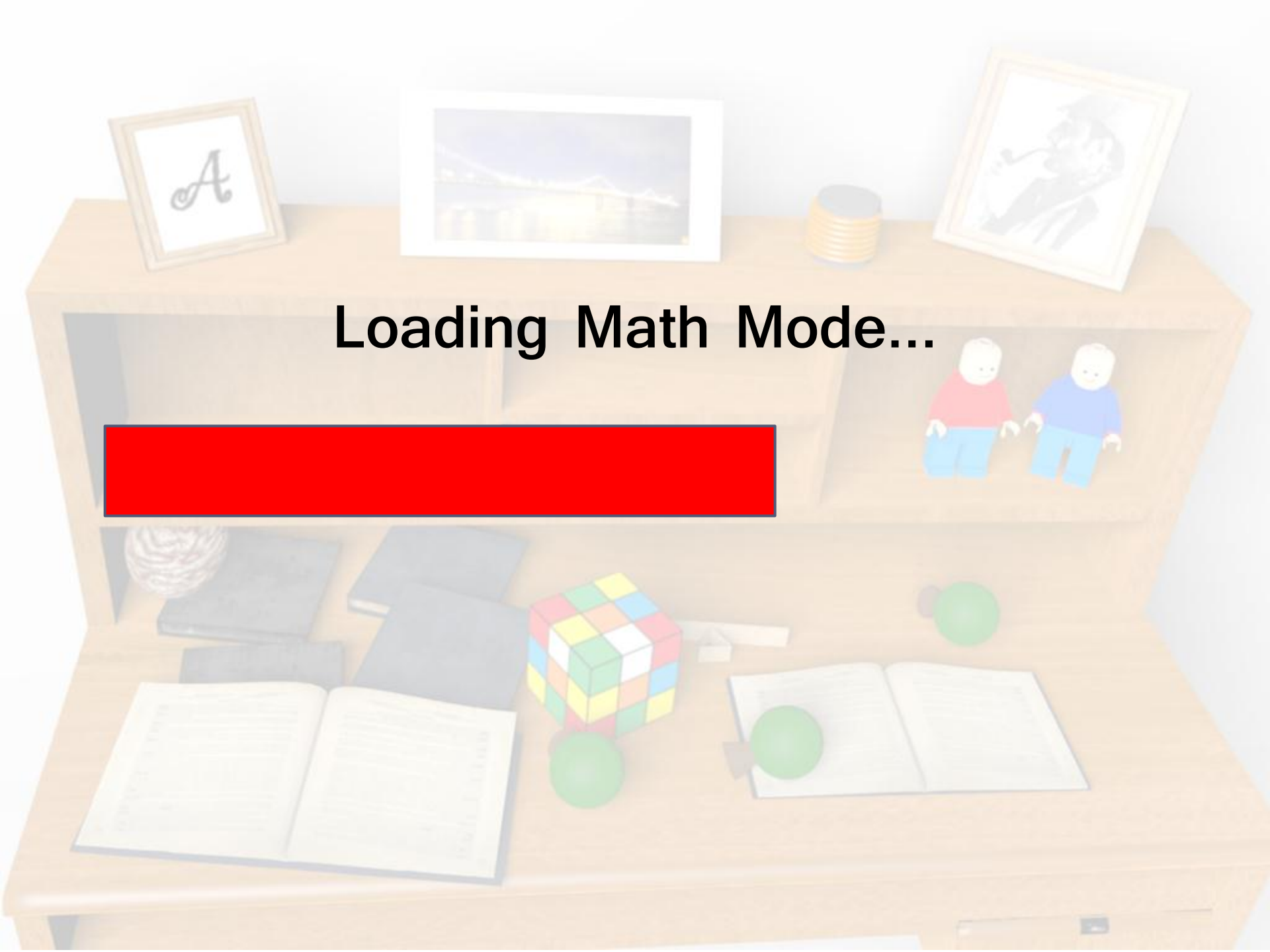

Loading Math Mode...

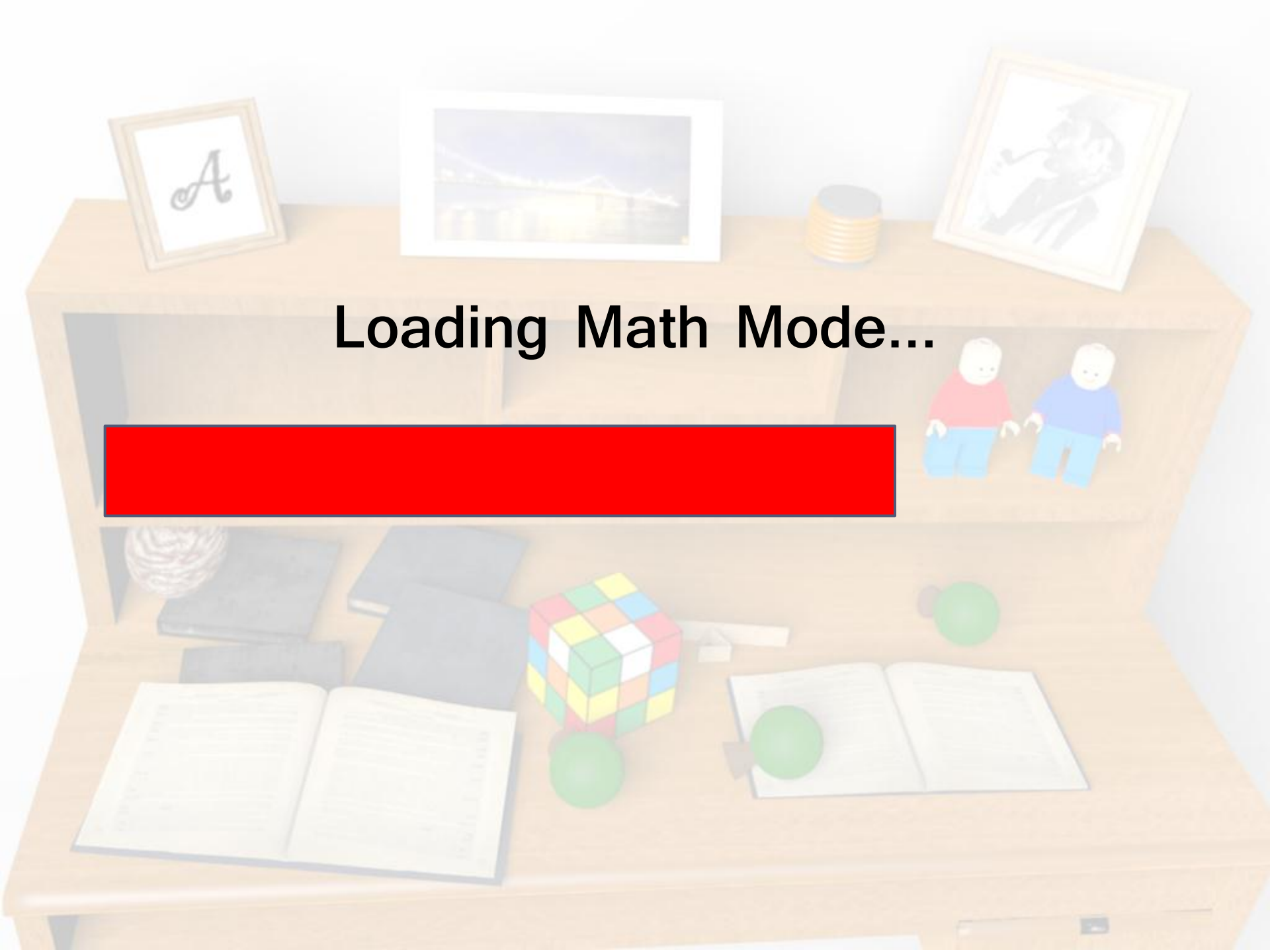

Loading Math Mode...

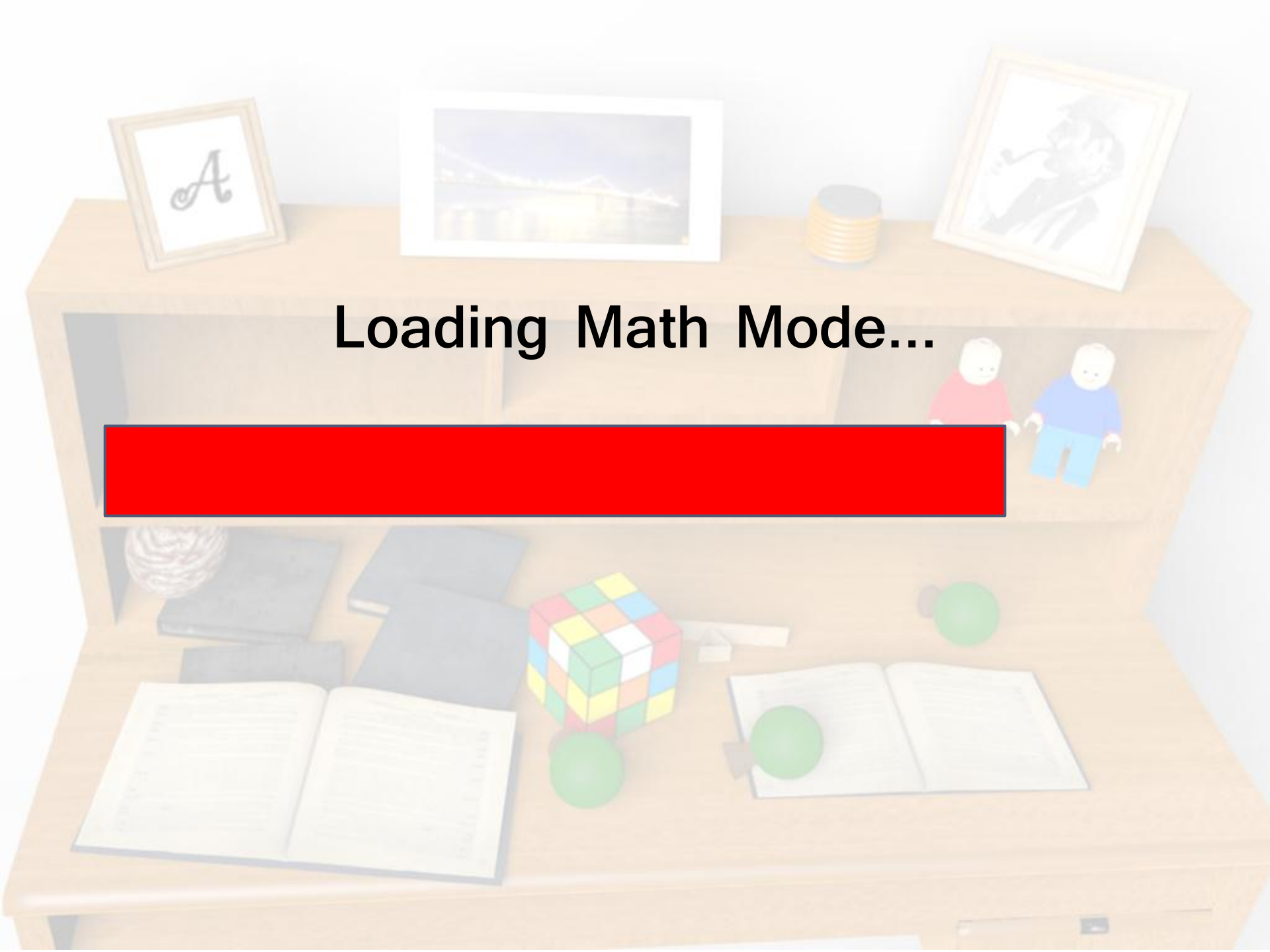

Loading Math Mode...

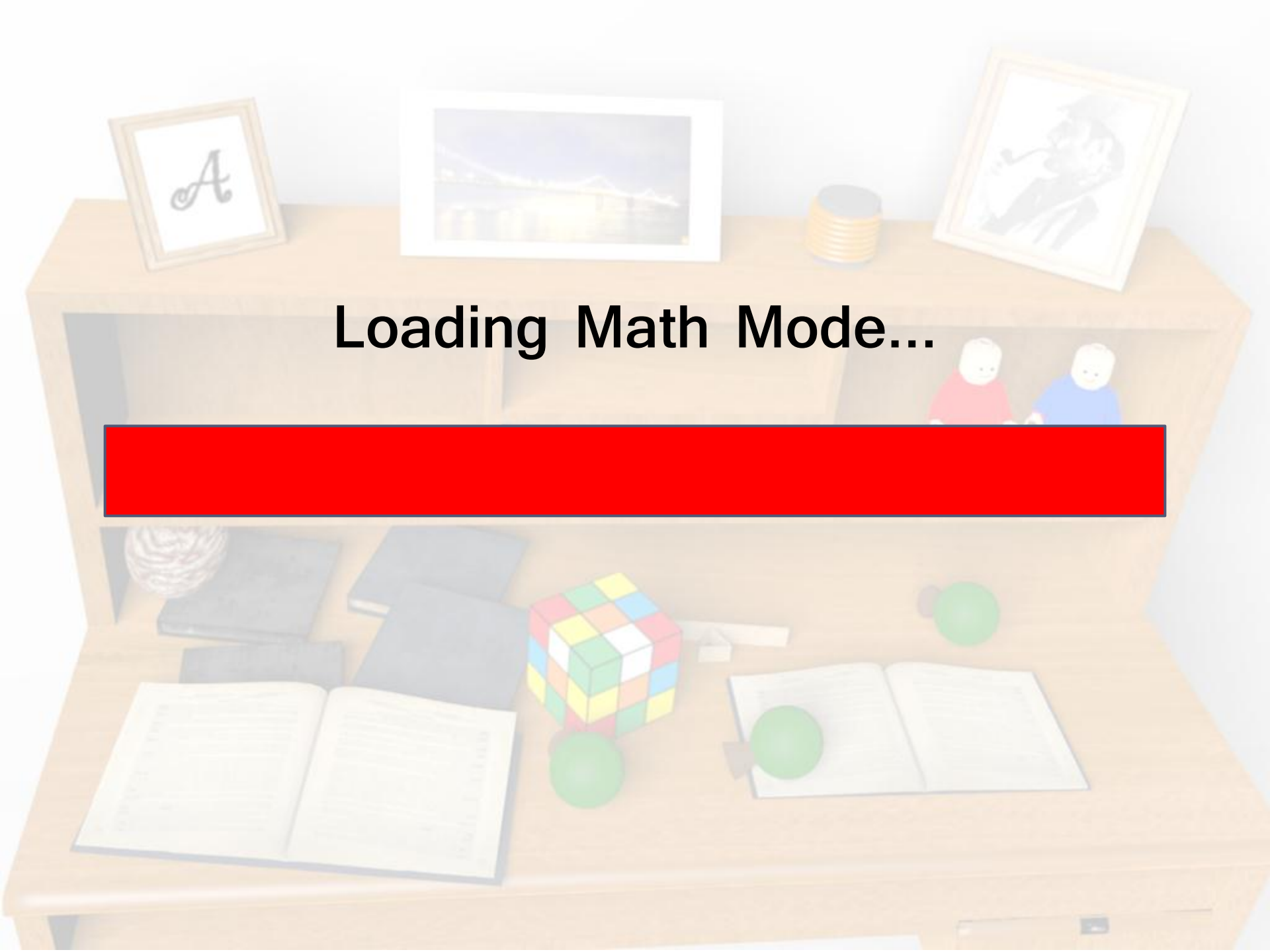

Loading Math Mode...

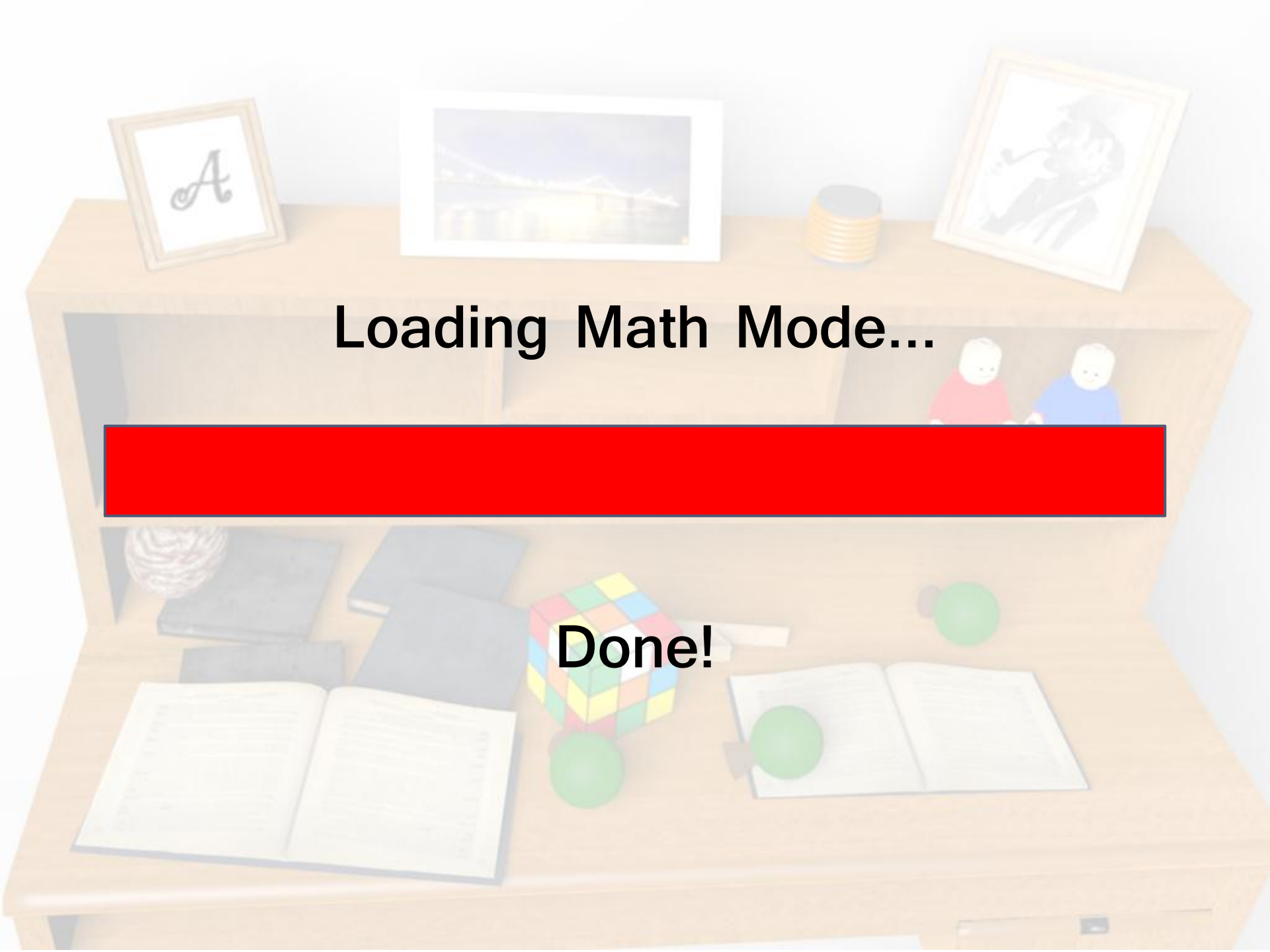

Loading Math Mode...


Loading Math Mode...


Loading Math Mode...


Loading Math Mode...


Loading Math Mode...


Loading Math Mode...
Done!

Общий вид

$$f(x) = \sum_i y_i w_i K(x_i, x)$$

Но как найти веса?

$$f(x) = \sum_i y_i w_i K(x_i, x)$$

- Найдем те, при которых **частота ошибок предсказания** на независимом множестве **минимальна.**

$$w = \operatorname{argmin}_w \ \text{Error}(f, w, \text{Data})$$

Но как найти веса?

$$f(x) = \sum_i y_i w_i K(x_i, x)$$

- Найдем те, при которых **мера ошибки предсказания** на независимом множестве **минимальна.**

$$w = \text{argmin}_w \; \text{Error}^*(f, w, \text{Data})$$

Но как найти веса?

$$f(x) = \sum_i y_i w_i K(x_i, x)$$

- Найдем те, при которых **мера ошибки предсказания** на независимом множестве **минимальна + много нулевых весов**

$w = \text{argmin}_w$ Error*(f, w, Data) + Complexity (w)

Support Vector Machine

```
from scikits.learn.svm import SVC

clf = SVC(kernel='linear')
clf.fit(training_set, training_labels)

predicted_class = clf.predict(test_set)
```

=> 877/1000


A


Классификация
по аналогии:
k-NN
SVM
RBF

Поиск ближайшего соседа

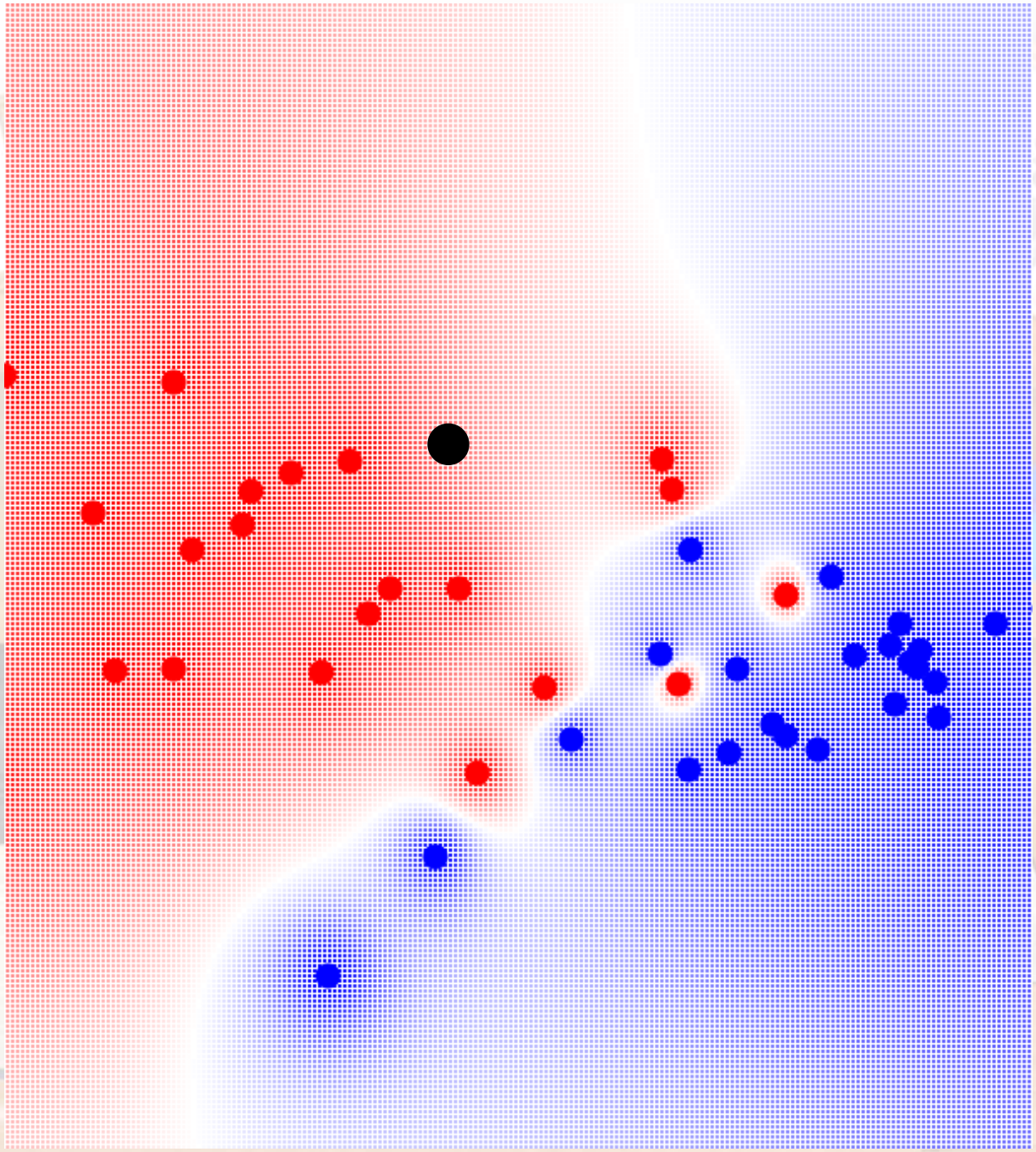
```
def classify(img):
    distances =
        [distance(img, p) for p in training_set]
    i = distances.index(min(distances))
    return training_labels[i]
```

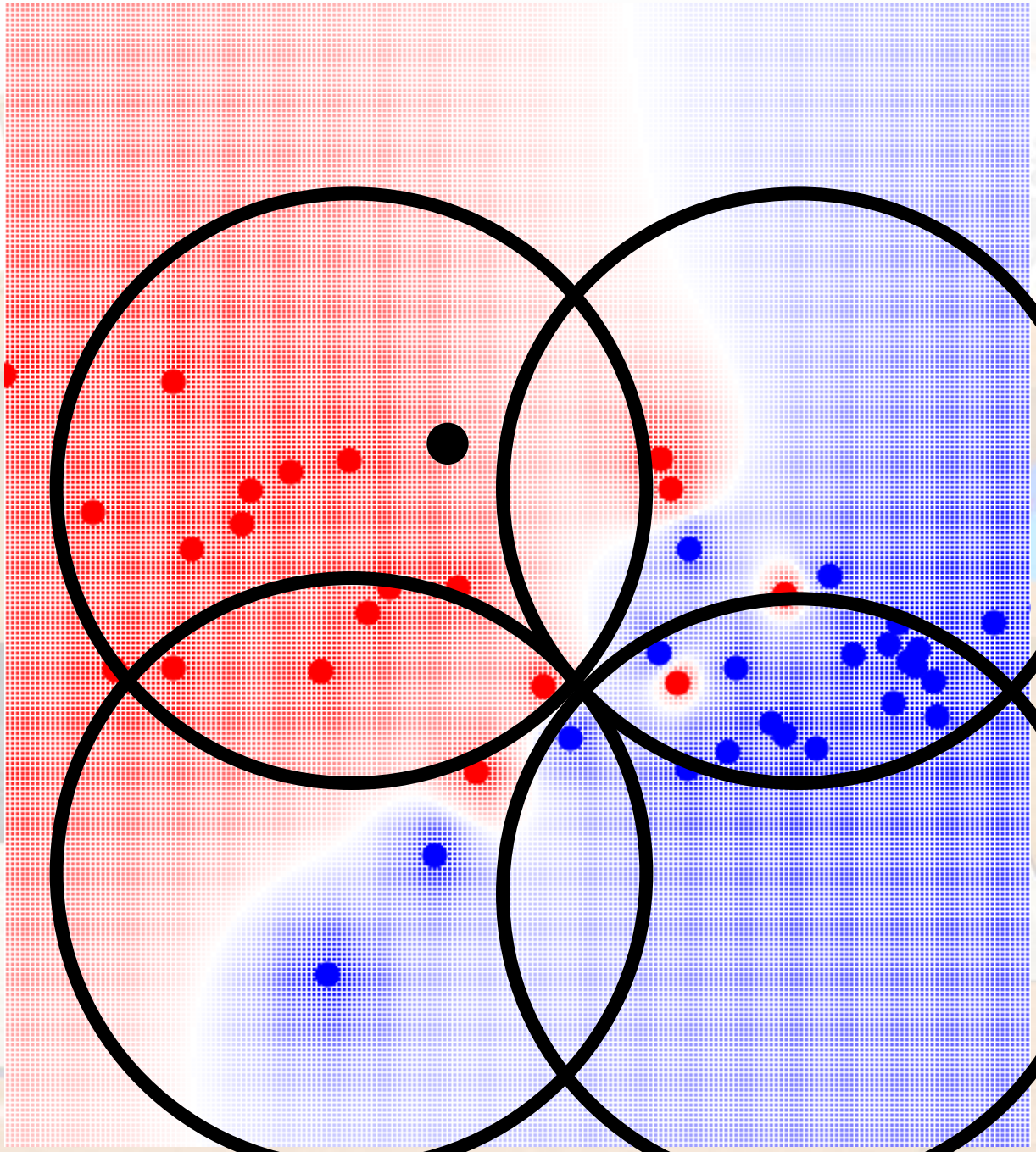
Неэффективно

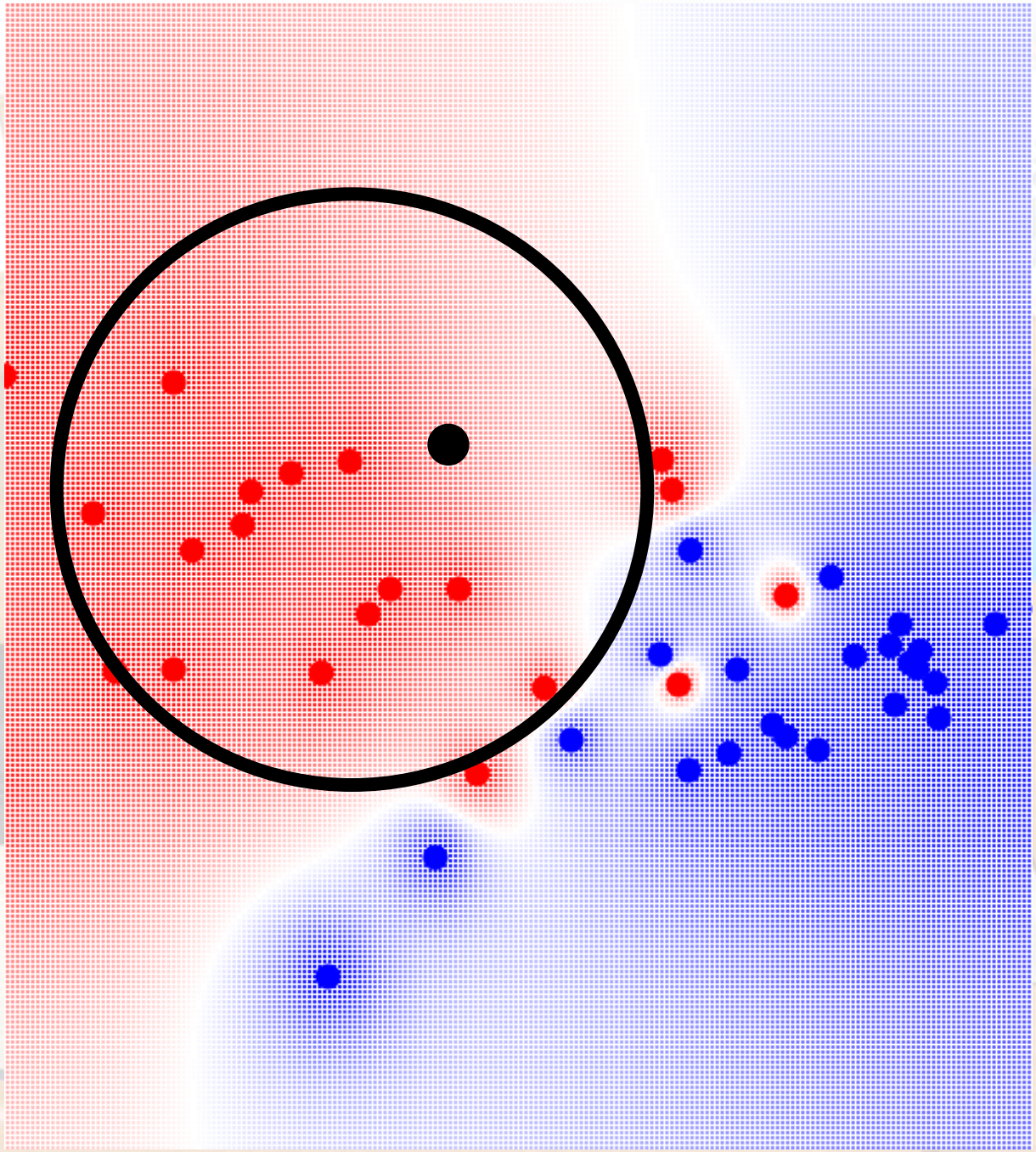
Поиск ближайшего соседа

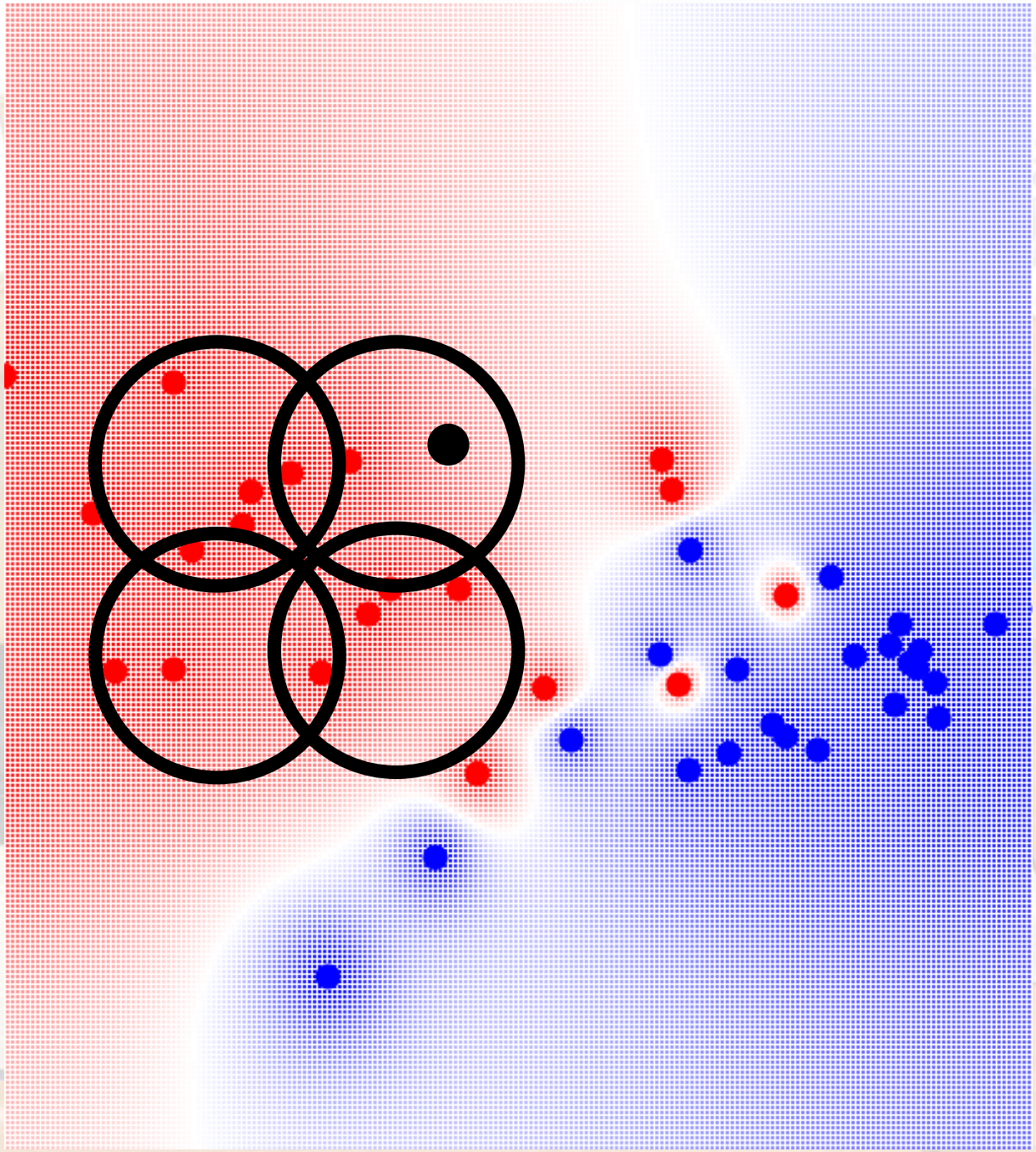
```
def classify(img):
    nearest_neighbour =
        training_set.find_nearest_neighbour(img)

    return nearest_neighbour.label
```

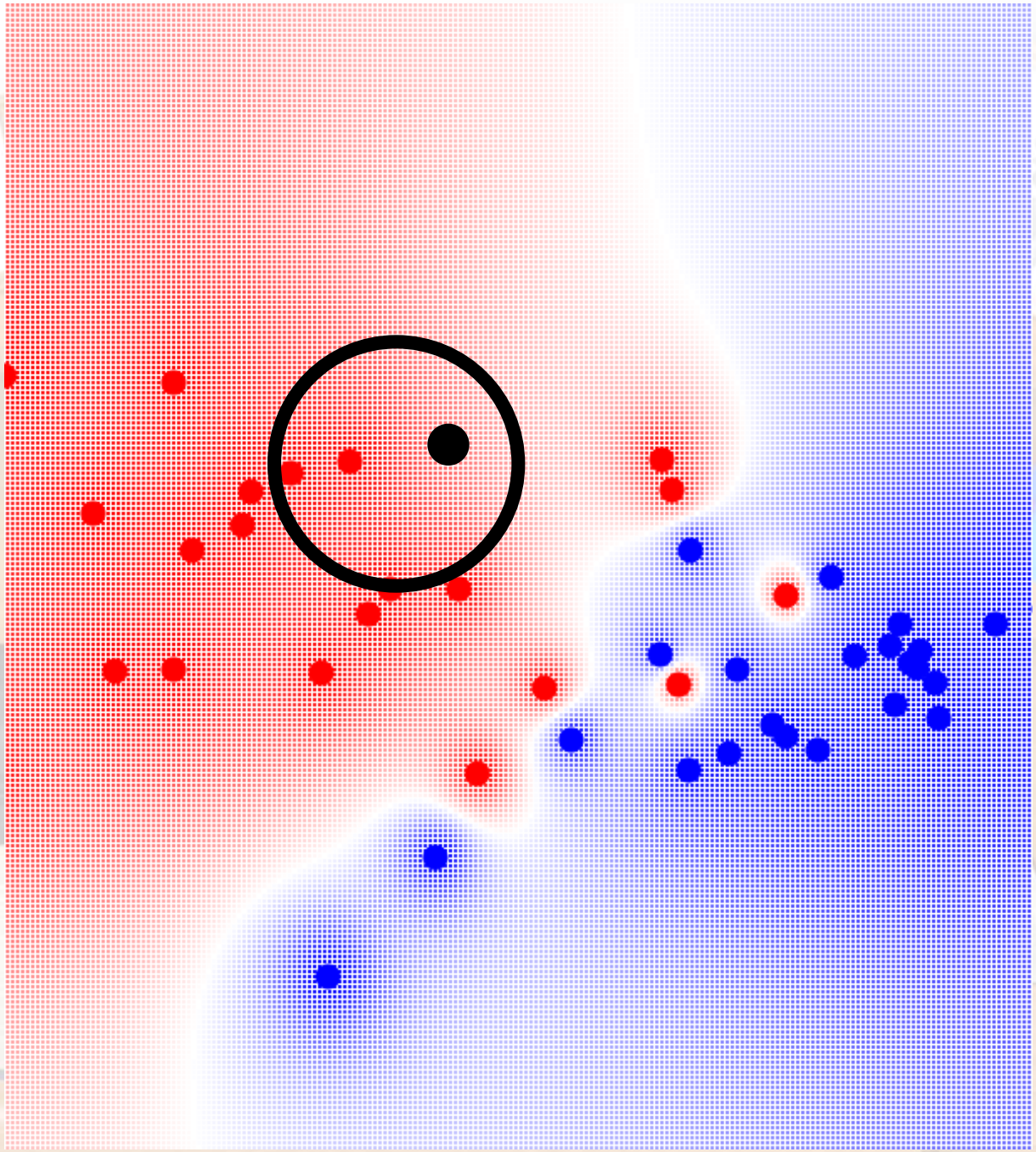
Индексирование!











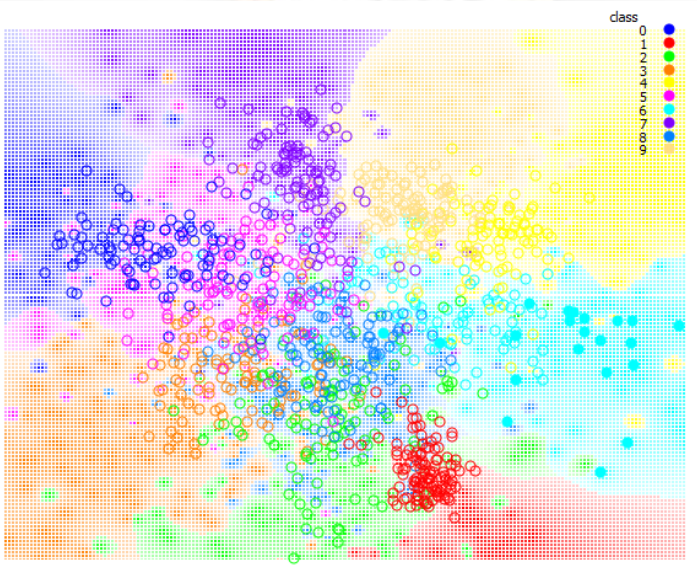
find_nearest_neighbour

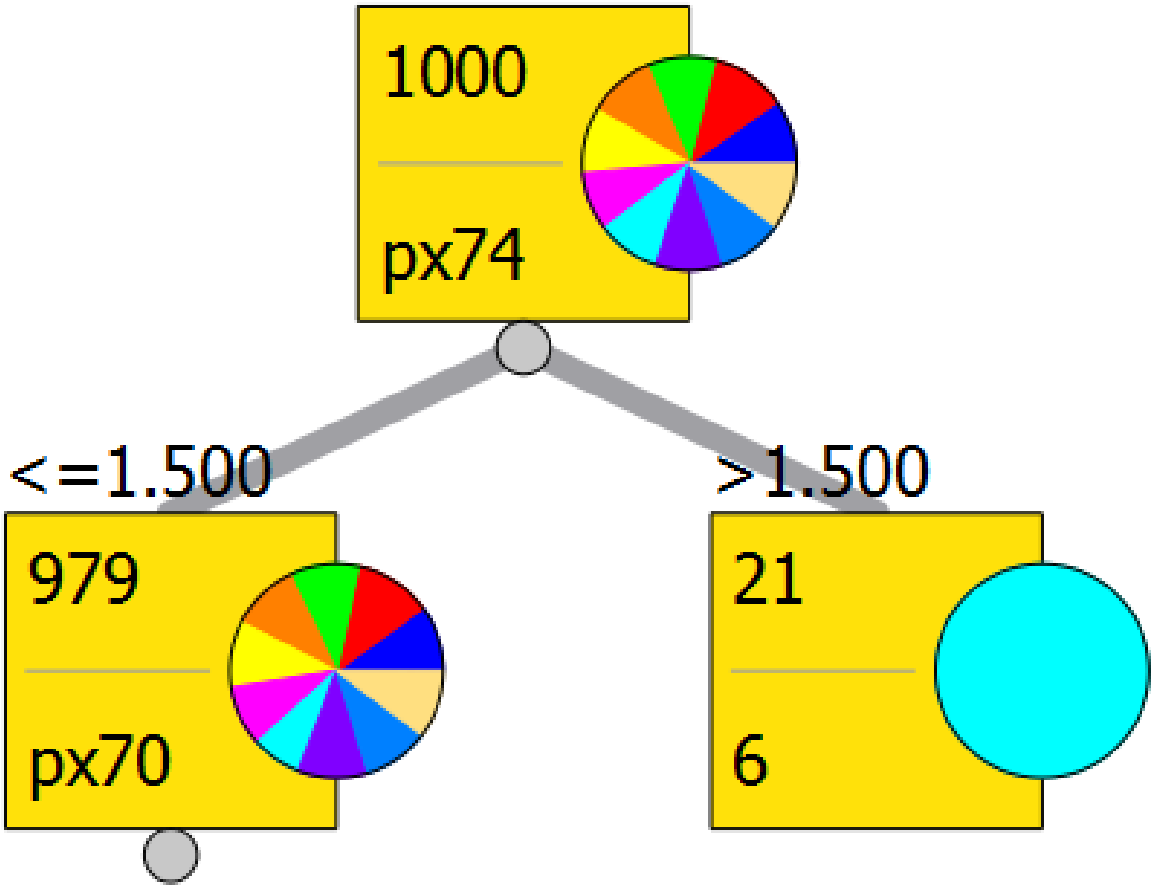
```
if pixel[10,13] > 4:
    if pixel[3,24] < 0:
      nearest_neighbour = A
    else:
      nearest_neighbour = B
else:
    nearest_neighbour = C
```

Classification Tree

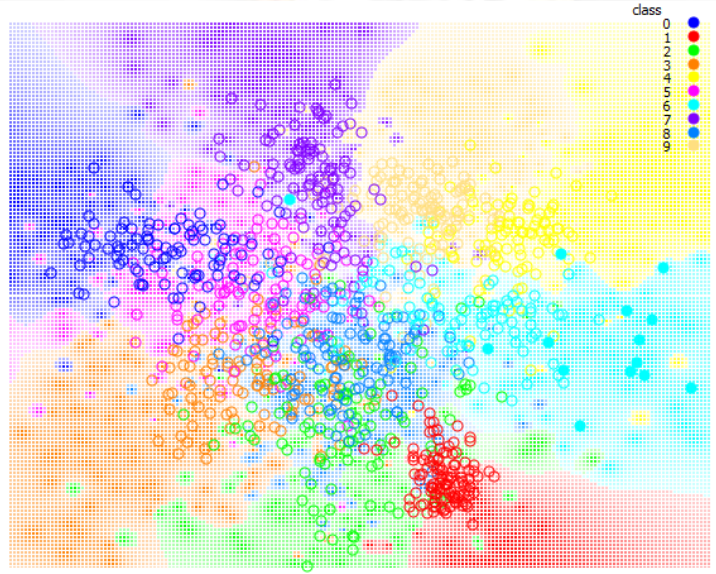
```
if pixel[10,13] > 4:
    if pixel[3,24] < 0:
        class = '1'
    else:
        class = '2'
else:
    class = '3'
```

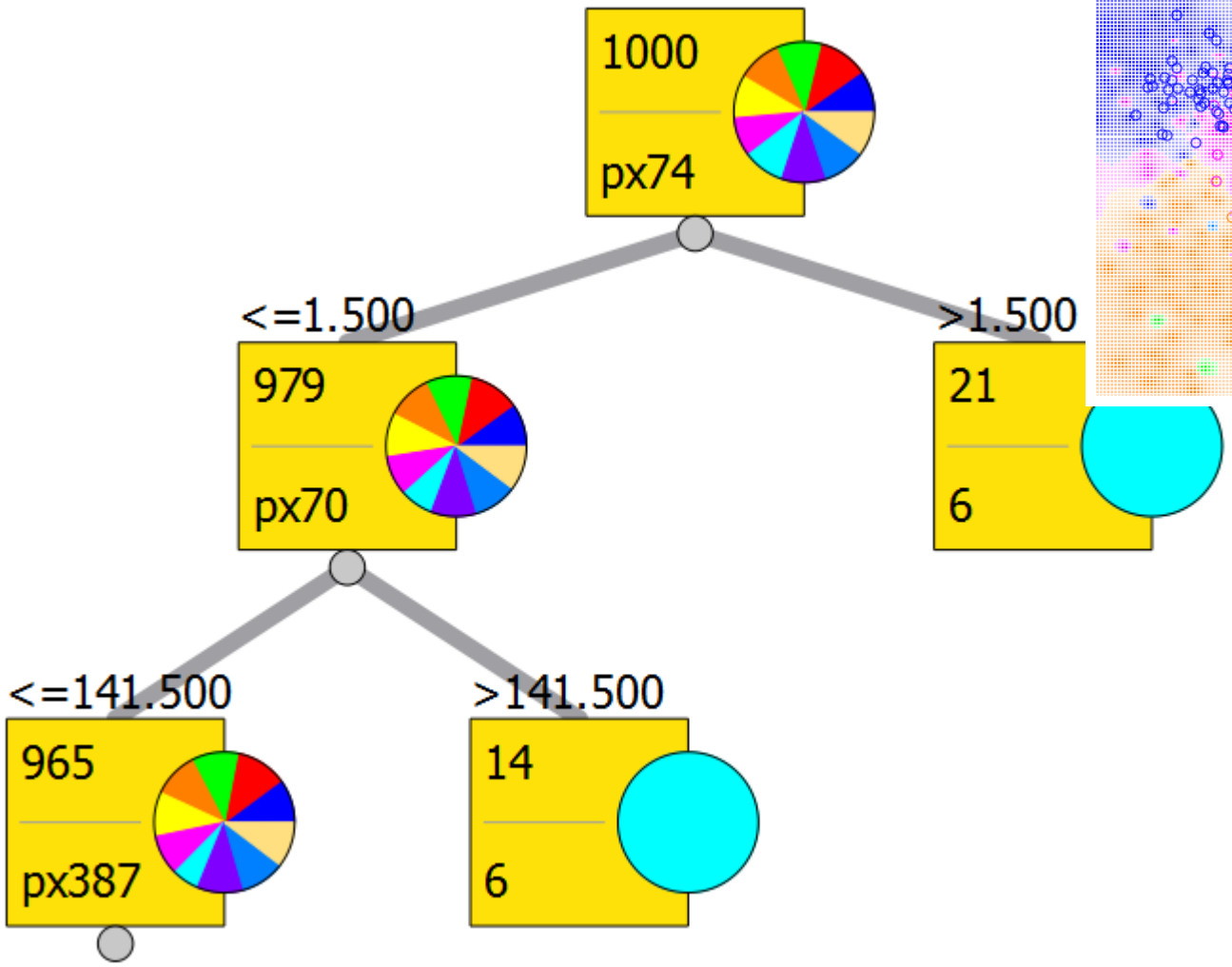
Classification Tree



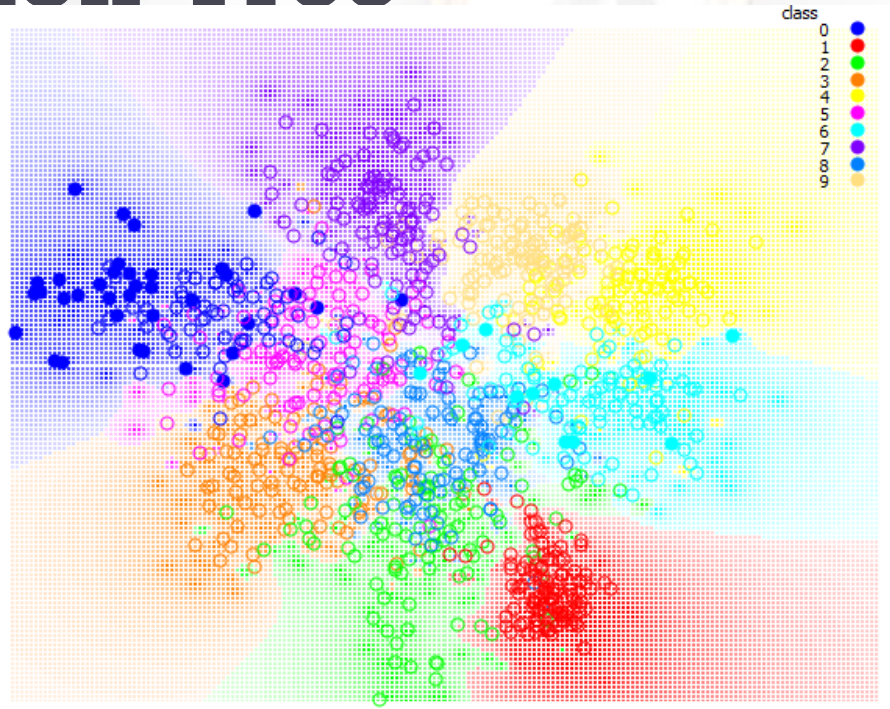


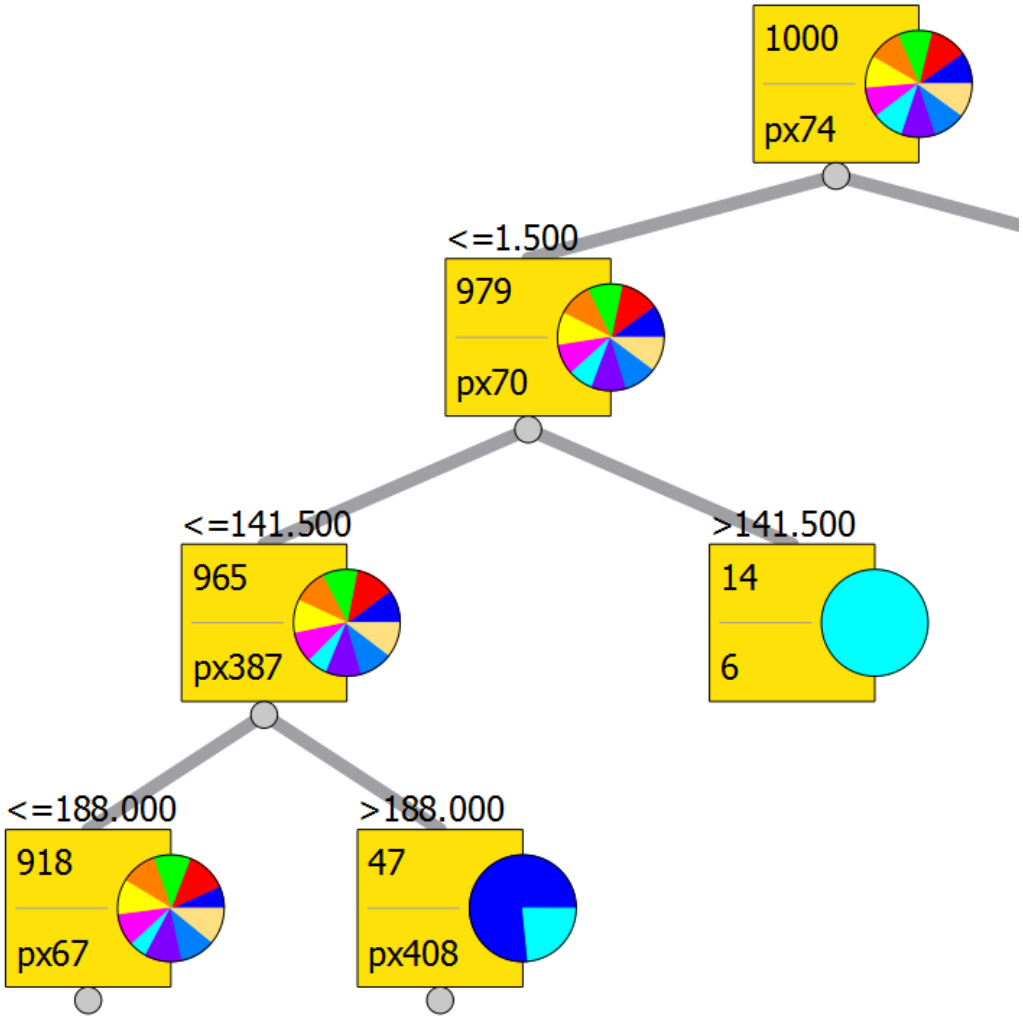
Classification Tree





Classification Tree





Orange

http://orange.biolab.si/


A

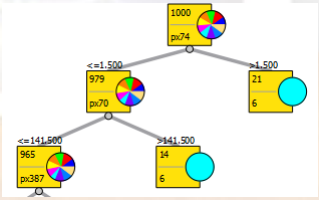

Деревья
ID3
C4.5
RegTree

Общий вид модели

$$f(x) = \sum_i y_i w_i K(x_i, x)$$

Подбор оптимальных параметров модели

Общий вид модели



Подбор оптимального дерева


Моделирование
Оптимизация

Линейная модель

f(image) =

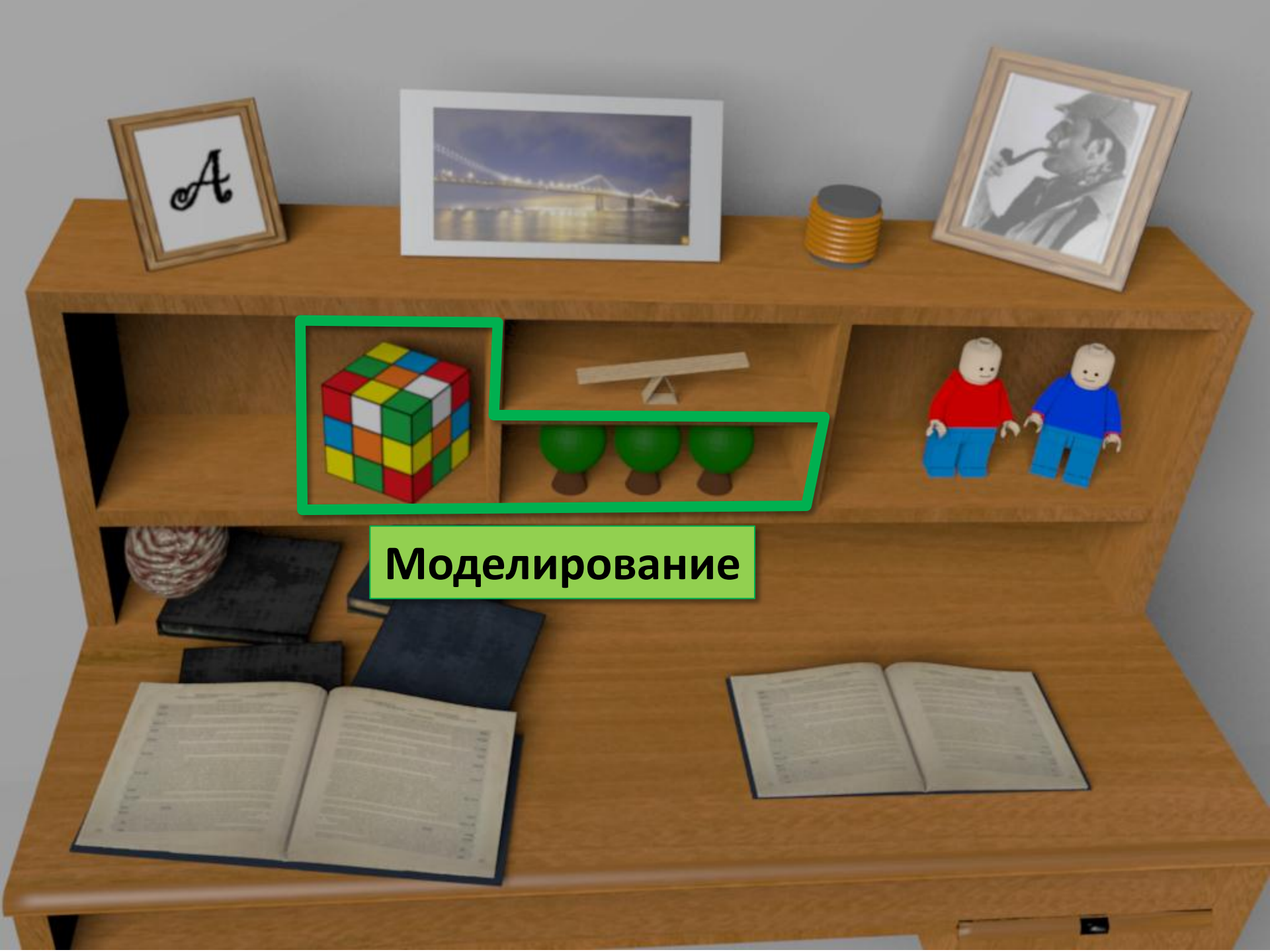
pixel1*$w1$ + pixel2*$w2$ + … + pixel784*$w784$

Linear Classification

```
from scikits.learn.linear_model import
        LinearRegression,
        LogisticRegression,
        RidgeClassifier,
        LARS,
        ElasticNet,
        SGDClassifier,
        ...
```

=> 809/1000


A


Моделирование

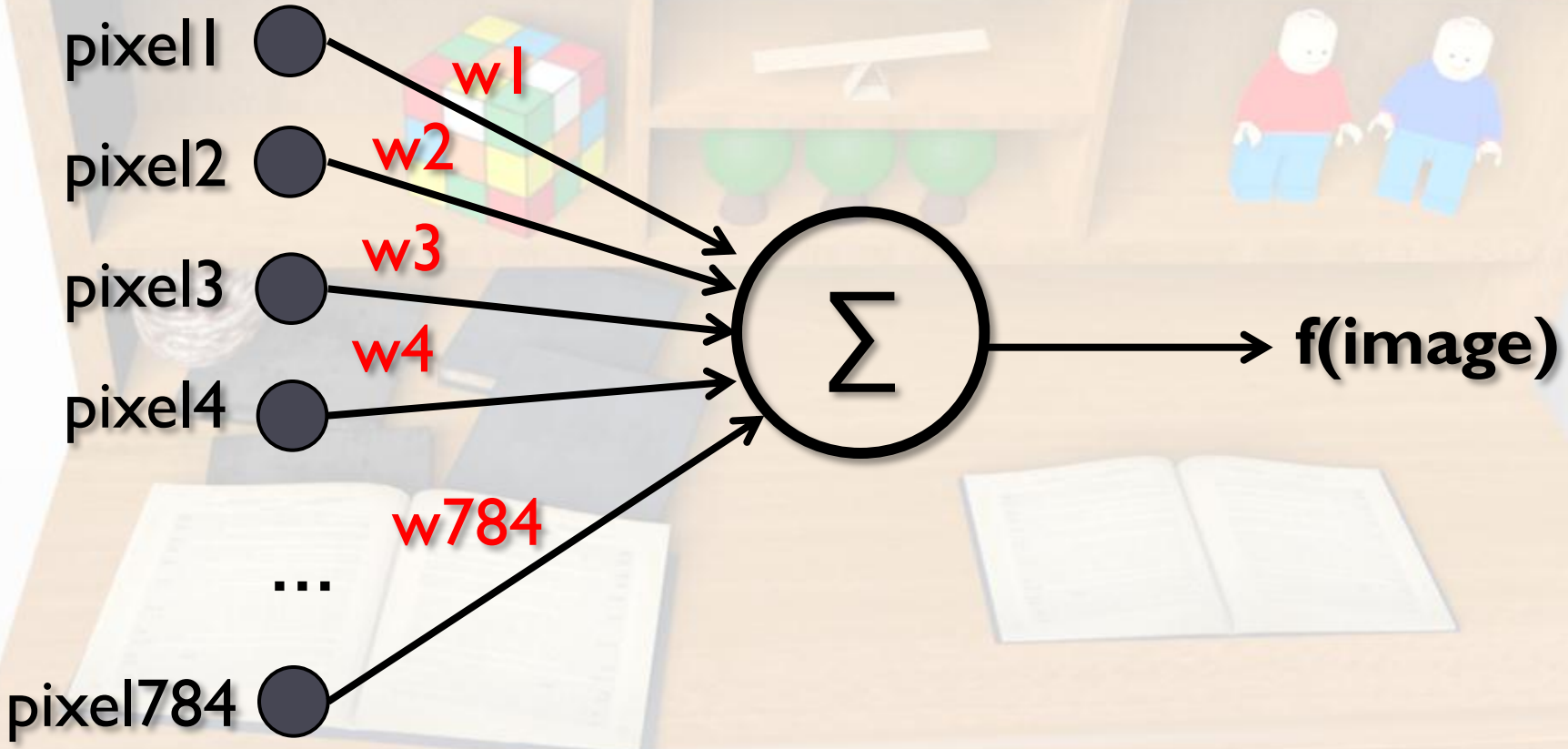
Линейная модель

f(image) =

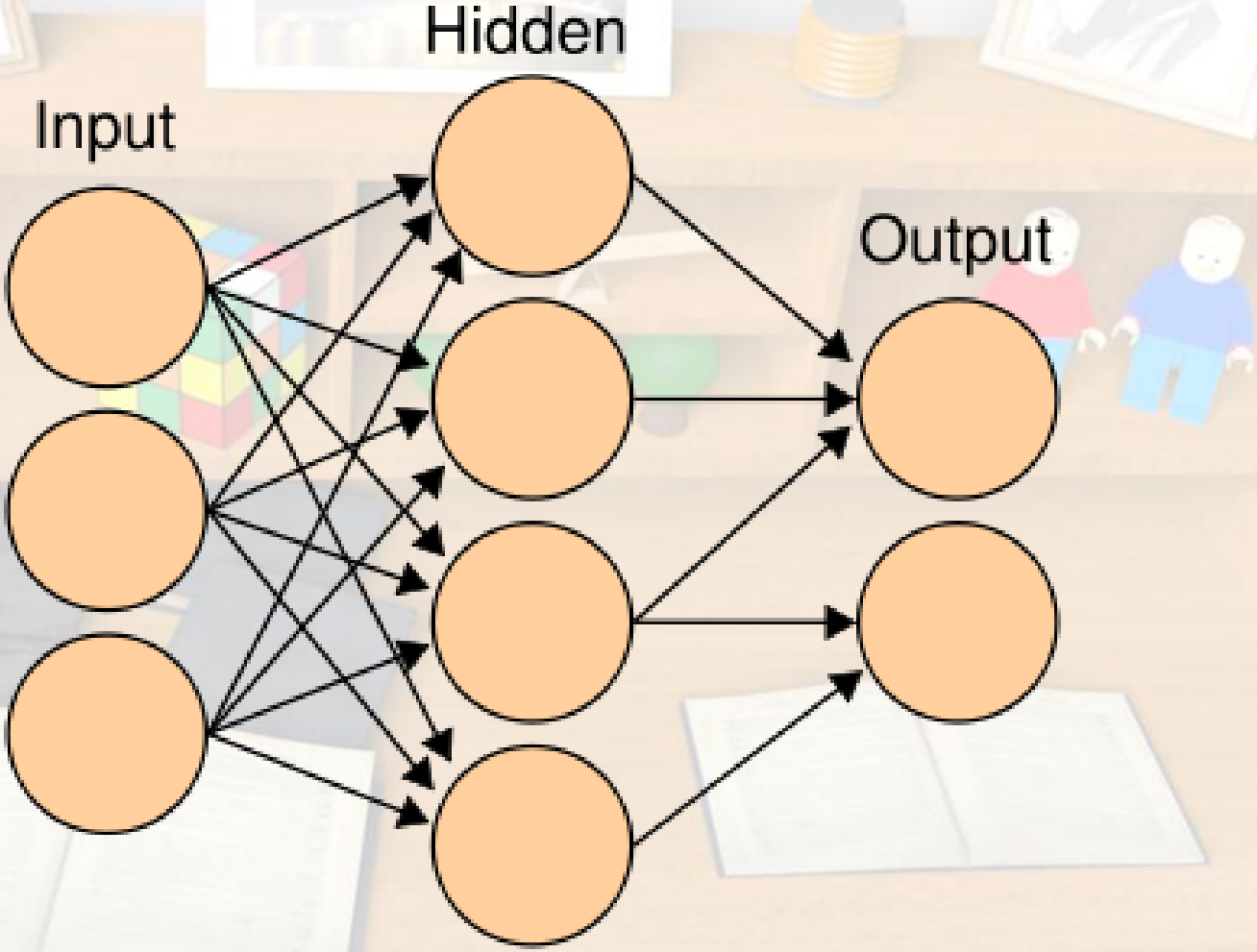
pixel1*$w1$ + pixel2*$w2$ + … + pixel784*$w784$

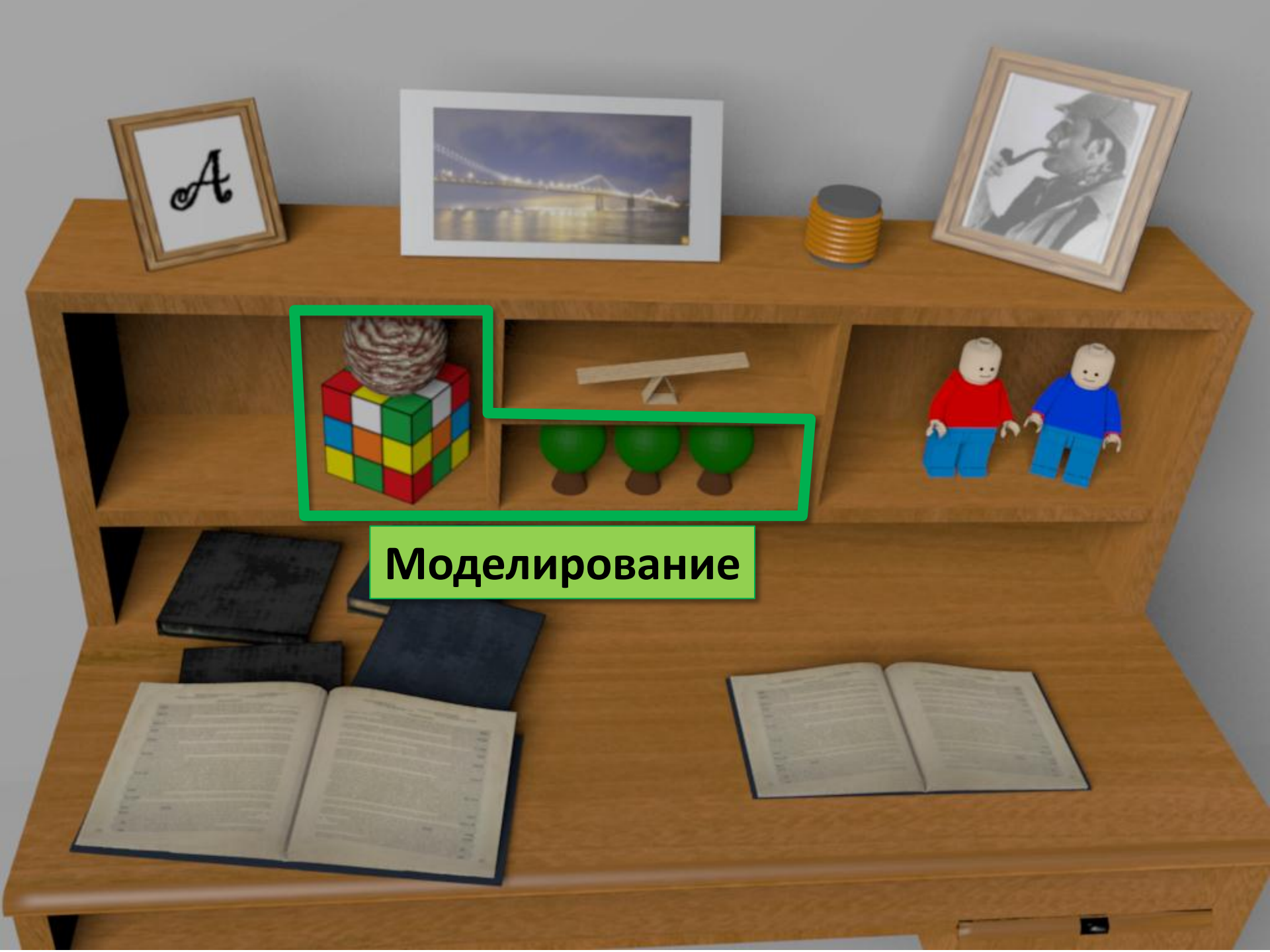
f(image) =

pixel1*w1 + pixel2*w2 + … + pixel784*w784

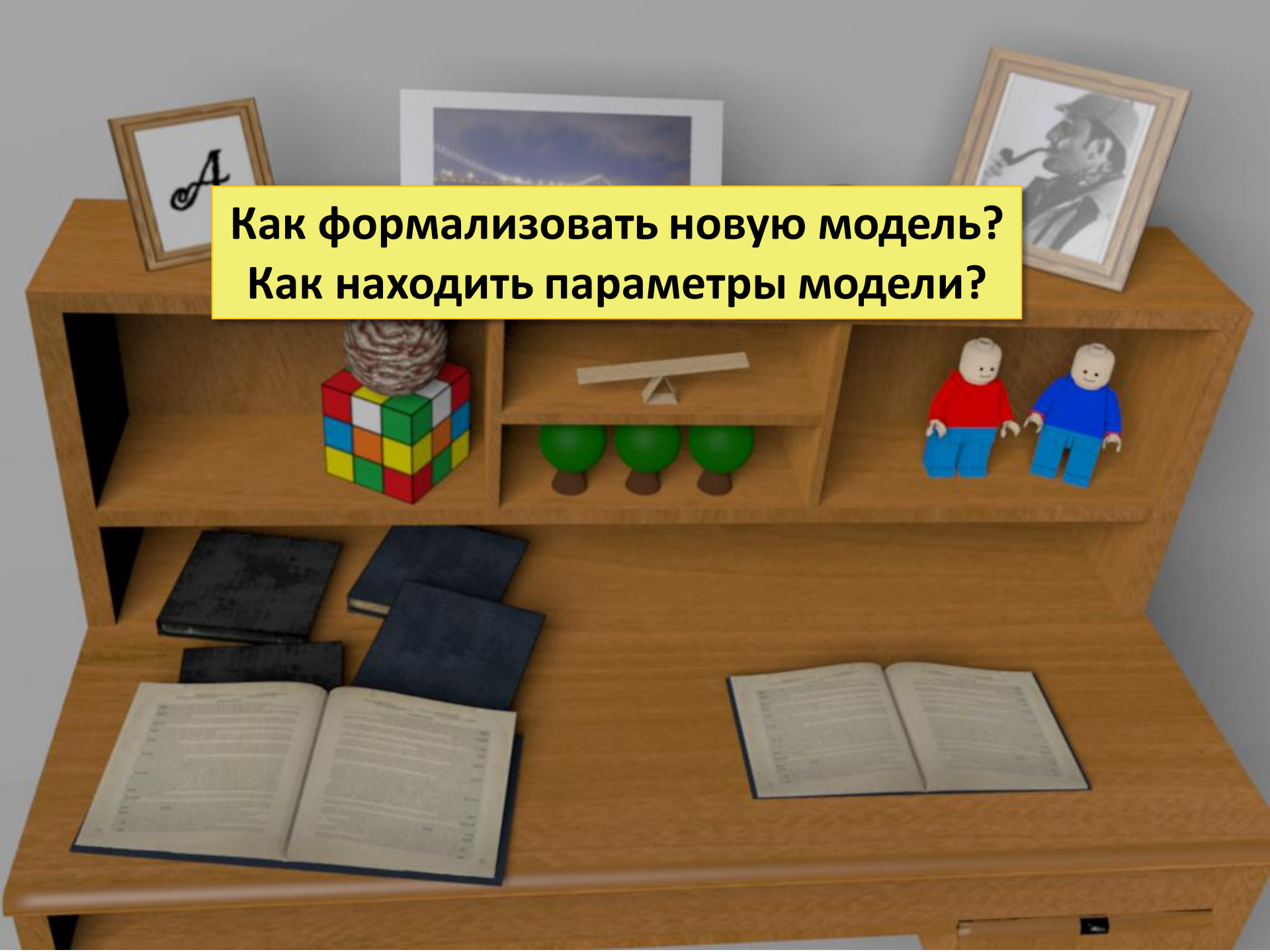


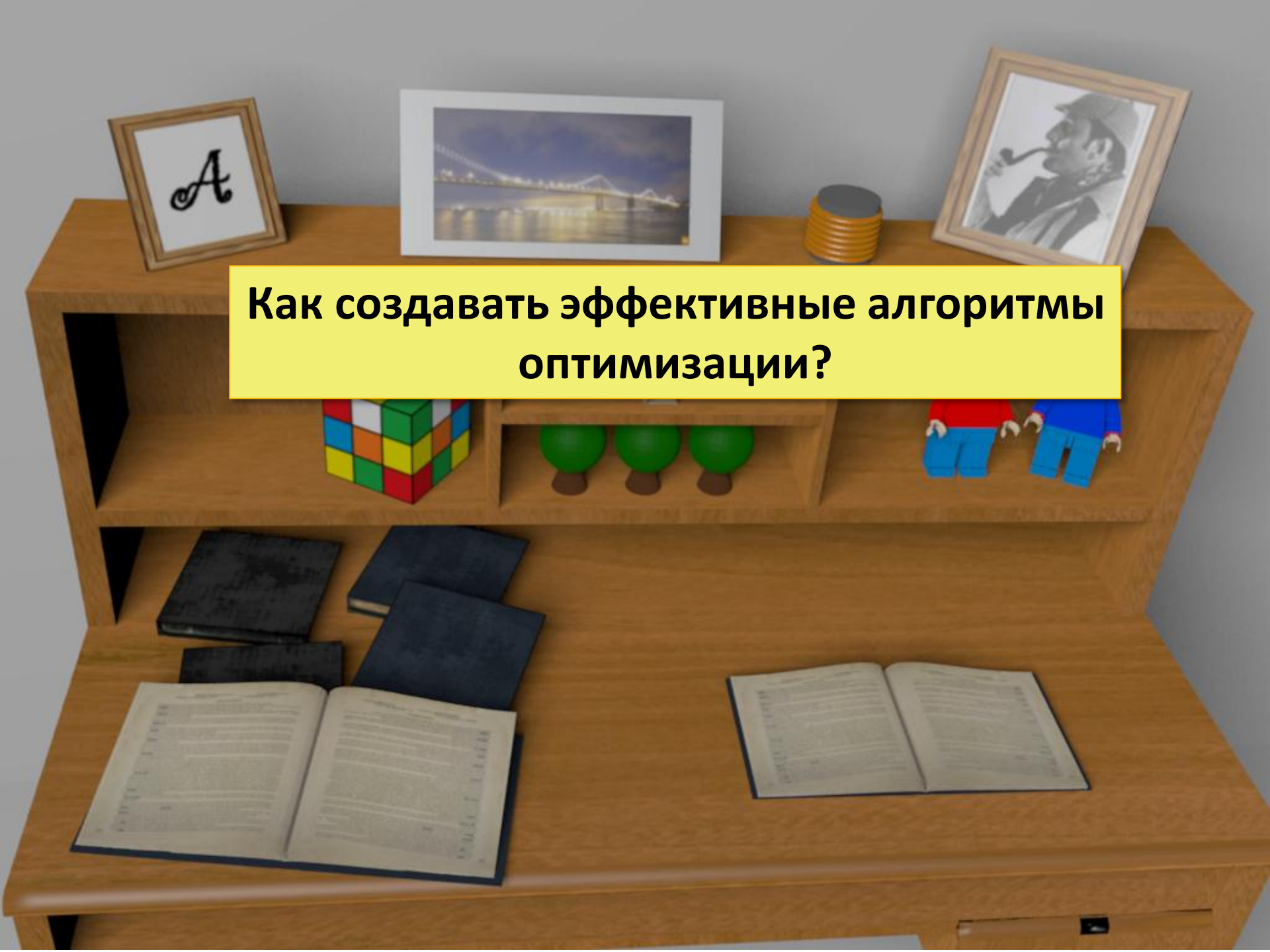
Нейронная сеть

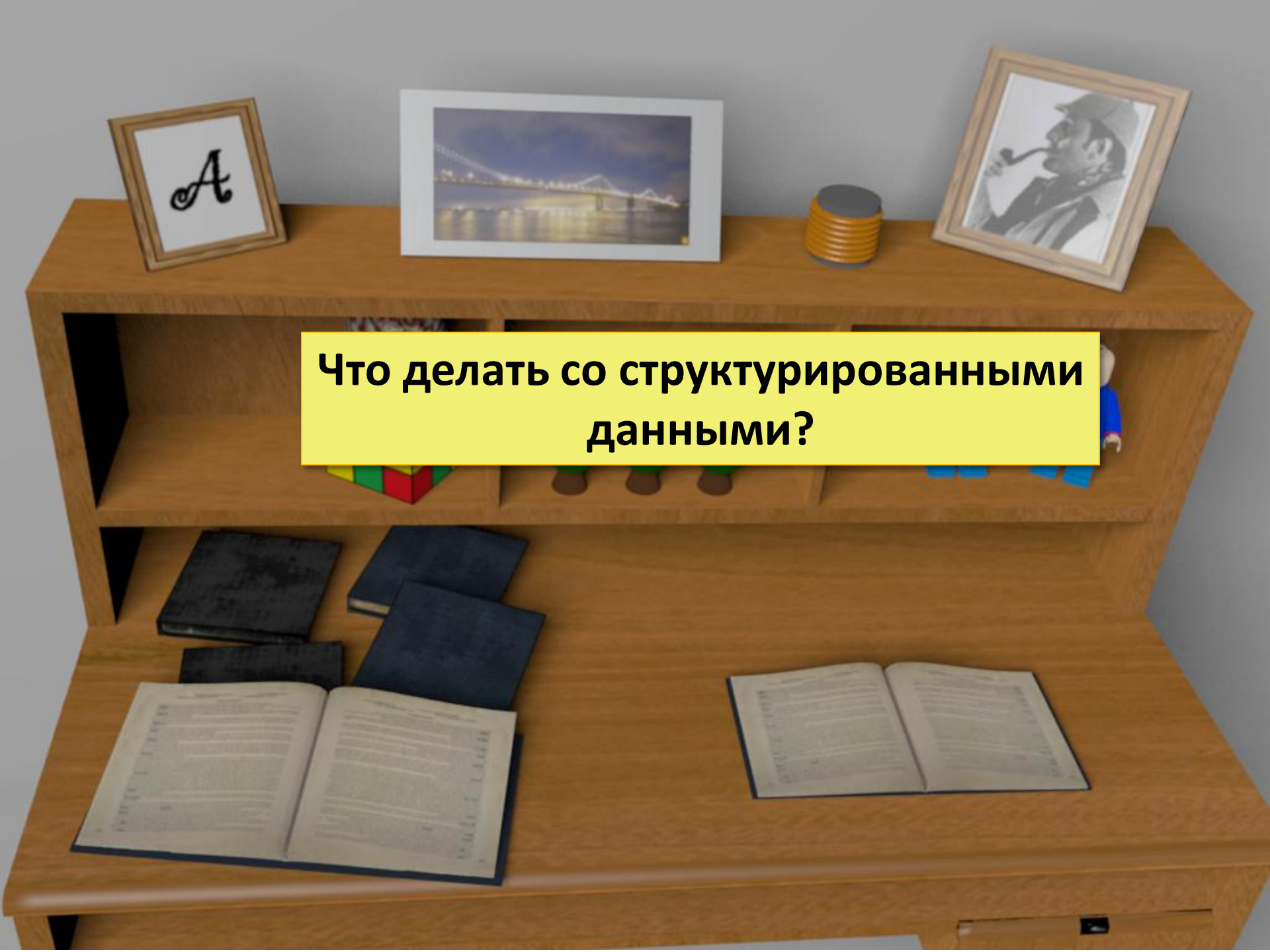



Моделирование

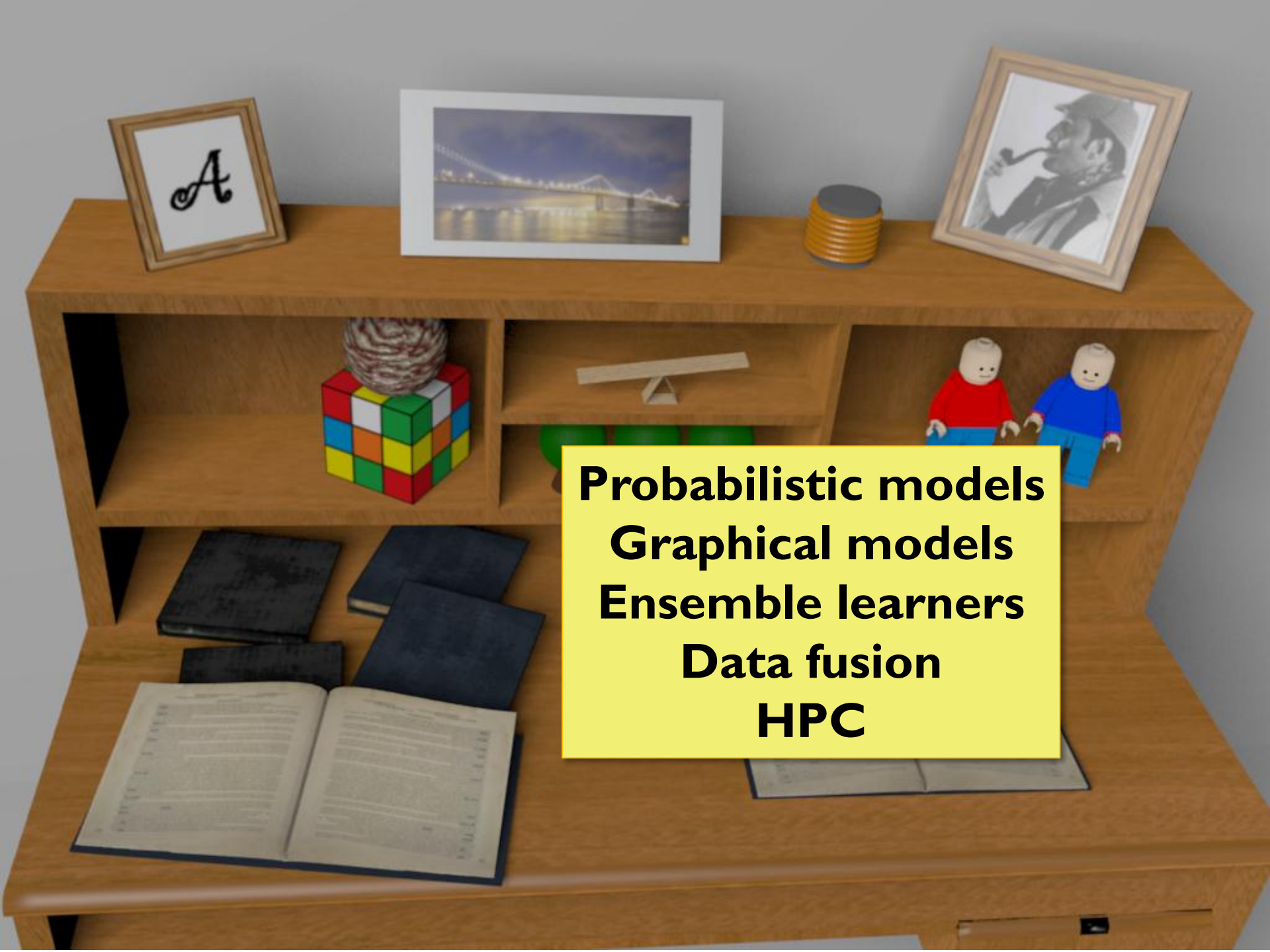

Почему тренированная на одном наборе данных модель должна работать на будущих данных?
В каком случае это не так?


Как формализовать новую модель?
Как находить параметры модели?

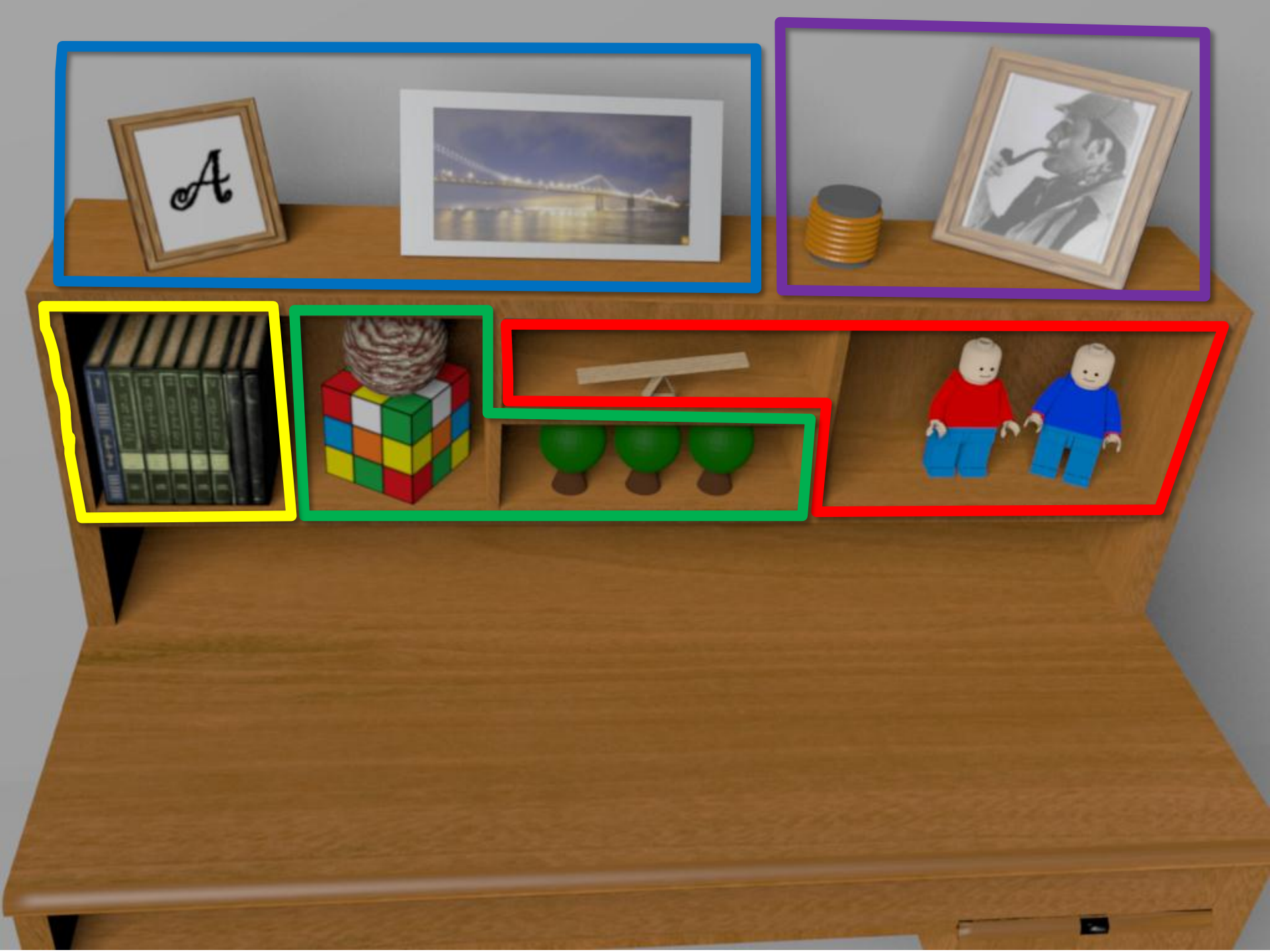

Как создавать эффективные алгоритмы оптимизации?


Что делать со структурированными данными?


Unsupervised learning
Semi-supervised learning
On-line learning
Active learning
Multi-instance learning
Reinforcement learning


Probabilistic models
Graphical models
Ensemble learners
Data fusion
HPC


Инструменты:
R, Weka, RapidMiner,
Orange, scikits-learn,
MLPy, MDP, PyBrain, …


A


Вопросы?