

Welcome to Machine Learning

Konstantin Tretyakov

<http://kt.era.ee>

AACIMP 2012
August 2012

STACC

Software Technology and
Applications Competence Center



Data mining,

Data analysis, Statistical analysis,

Pattern discovery, Statistical learning,

Machine learning, Predictive analytics,

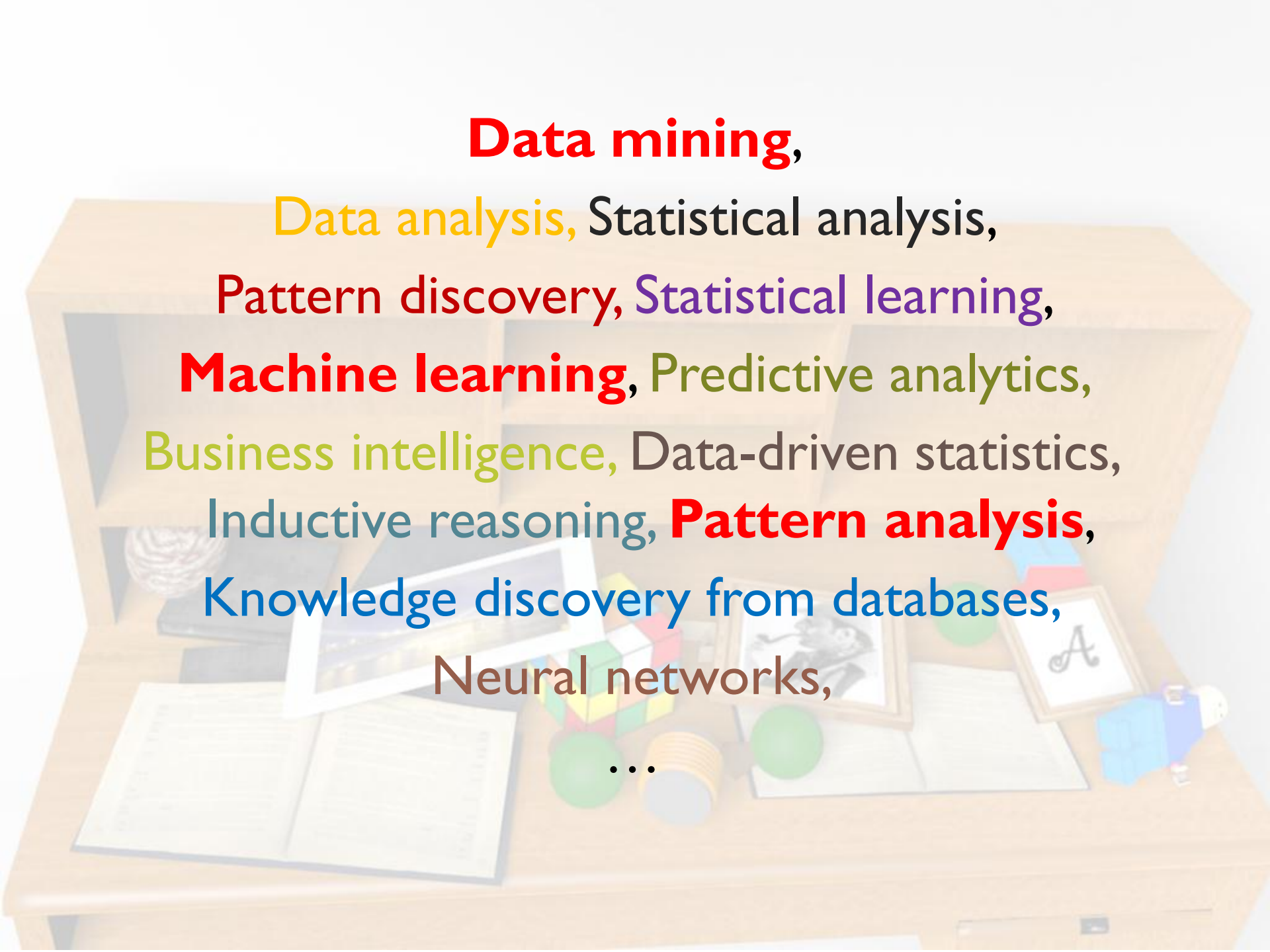
Business intelligence, Data-driven statistics,

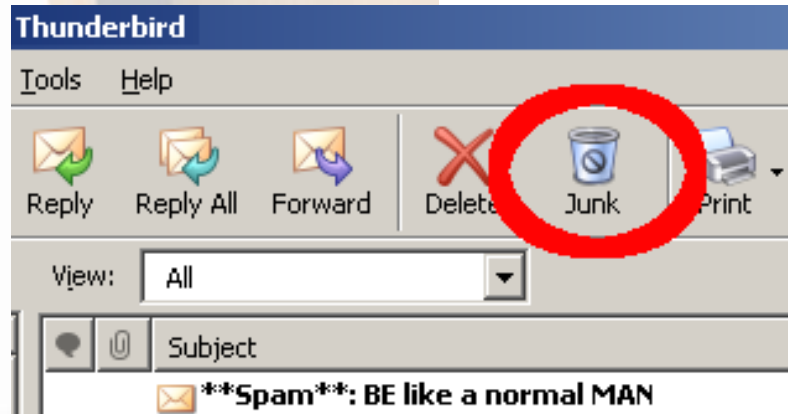
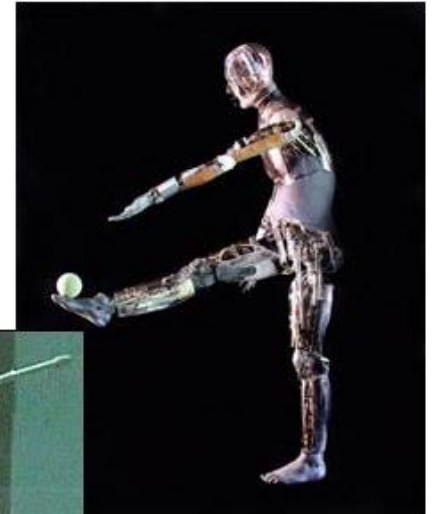
Inductive reasoning, **Pattern analysis,**

Knowledge discovery from databases,

Neural networks,

...



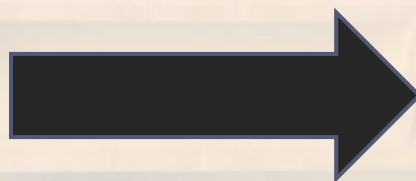
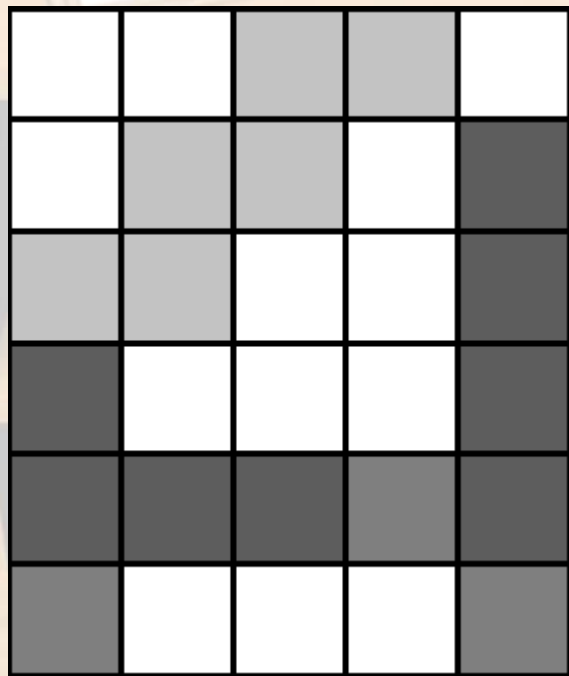


Google translate

NETFLIX

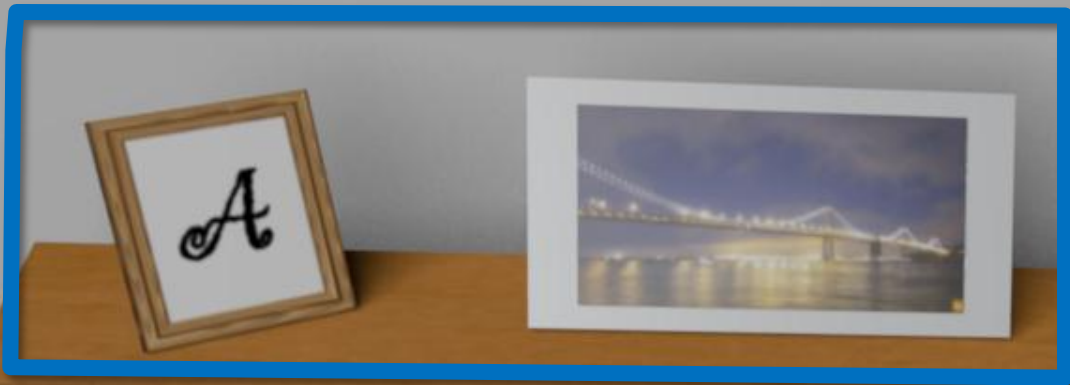






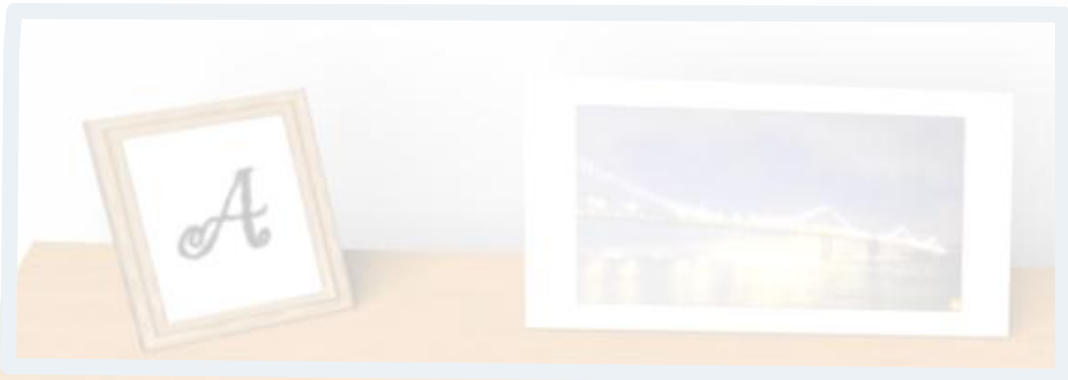
A



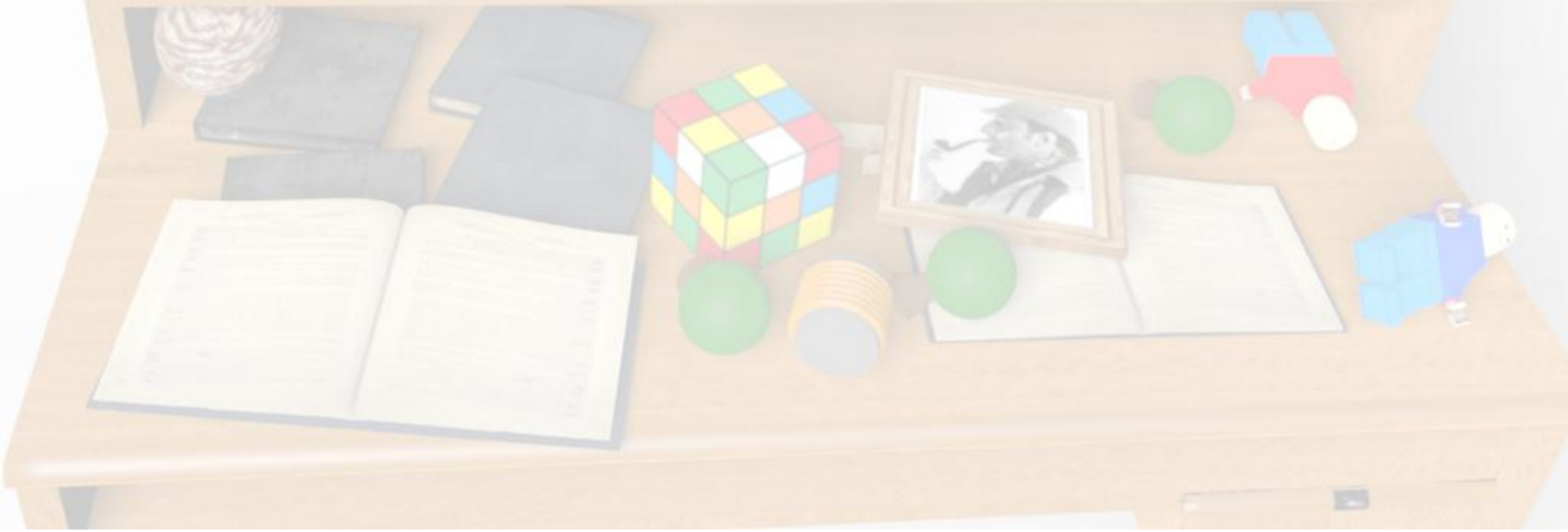


“Unformalizable” problems





“Unformalizable” problems





A

A

B

“Unformalizable” problems

A

B

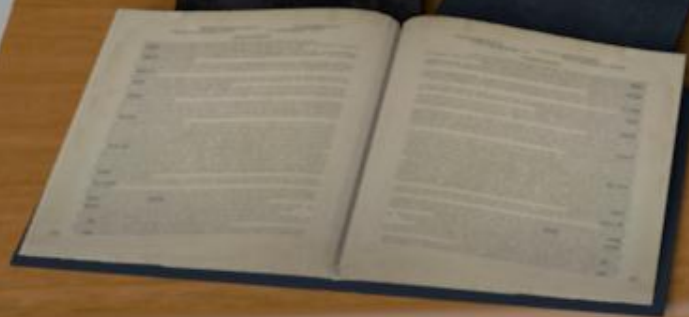
B

A

A

A

B





- 
- ▶ General rule:

IF

THEN

**(X is made of plastic),
(X is not edible)**

- ▶ Application in the particular case:

X = Rubic's cube

⇒

Rubic's cube is not edible

- 
- ▶ General rule:

$\text{add}(x, y) = \text{function } \{ \dots \}$

- ▶ Application in the particular case:

$\text{add}(2, 4)$

- 
- General rule:

**add(x, y) = function {
 ???
}**

- Particular cases:

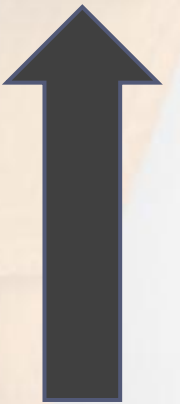
add(2,4) = 6

add(5,3) = 8

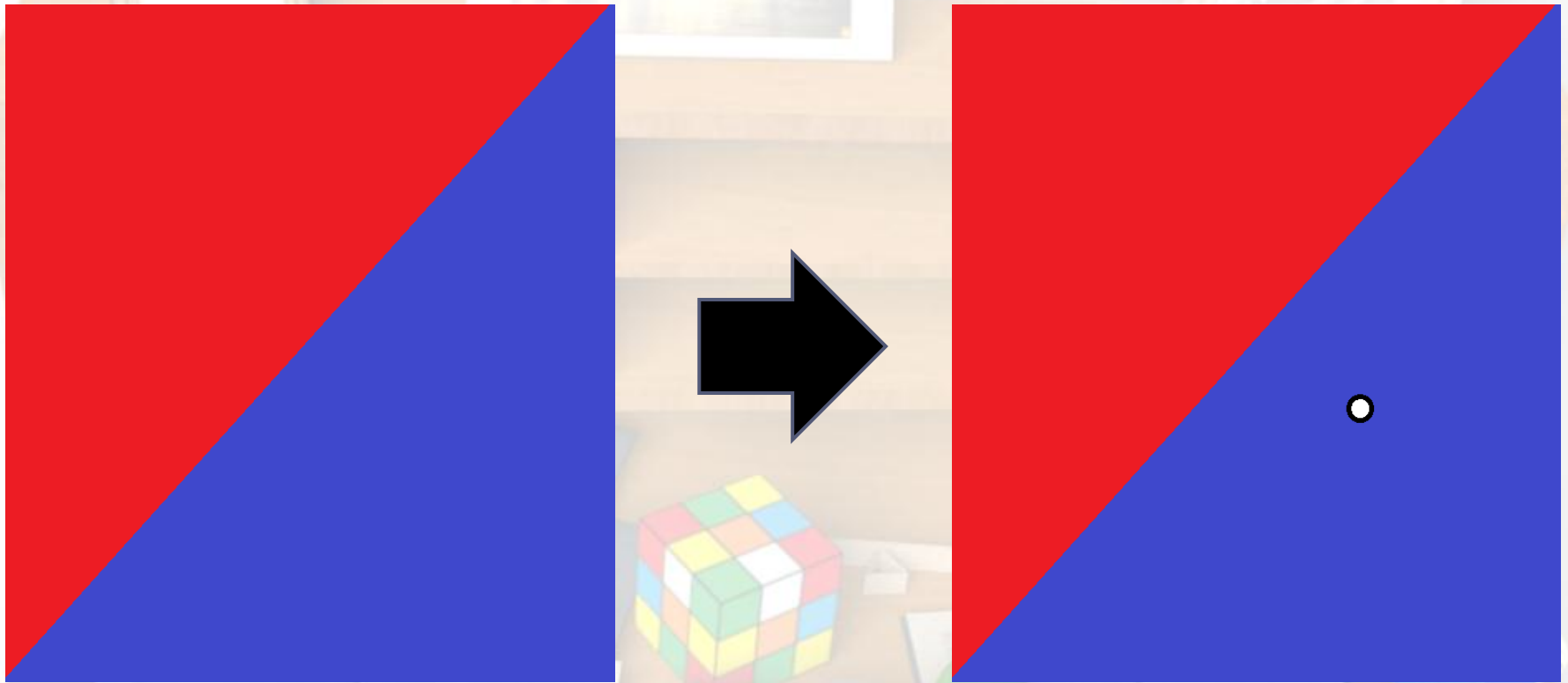
add(1,2) = 3

...

Induction

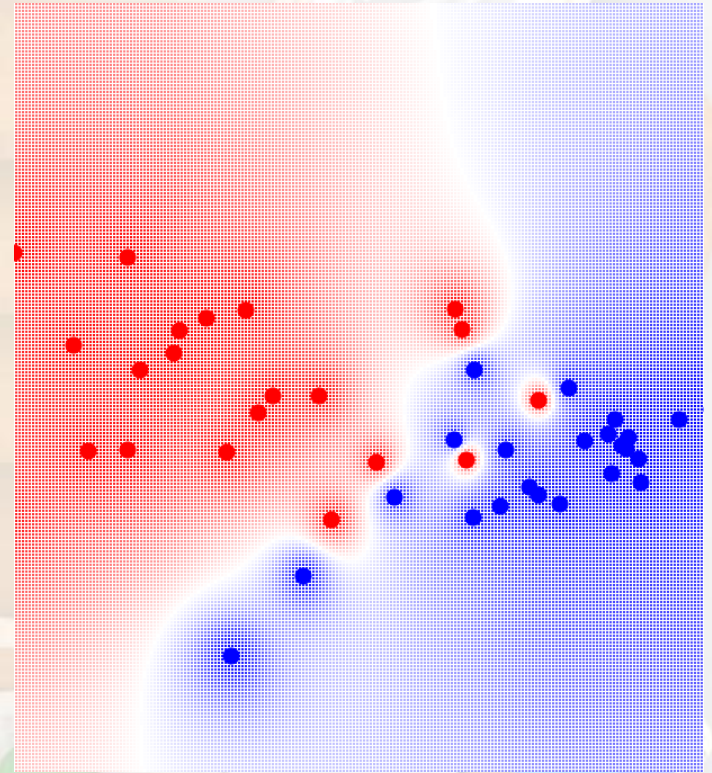
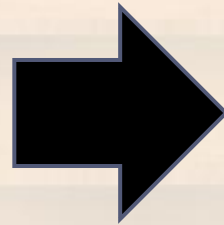
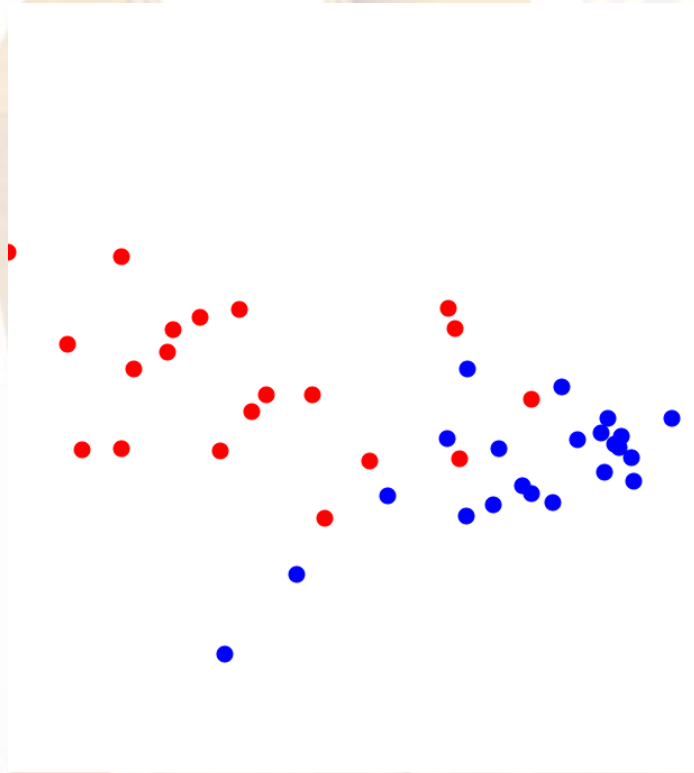


Deduction



Given a general rule, make a decision in a particular case

Induction

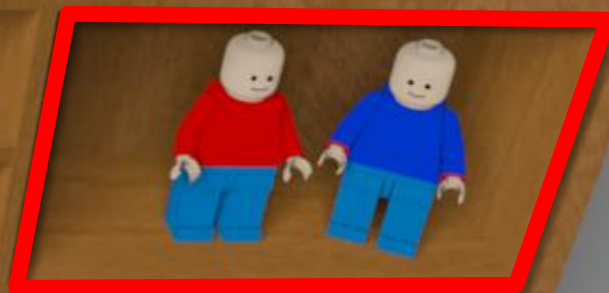
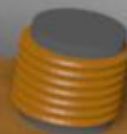


Given particular cases, find a general rule



Deduction and Induction





**Classification
by analogy**



MNIST dataset

- ▶ <http://yann.lecun.com/exdb/mnist/>
- ▶ Handwritten digits, 28 x 28



MNIST dataset

```
images = load_images()  
labels = load_labels()
```

Let us just use 1000 images

```
training_set      = images[0:1000]  
training_labels = labels[0:1000]
```

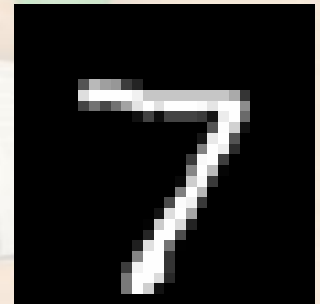
MNIST dataset

```
> training_set[0]
```

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

```
> training_labels[0]
```

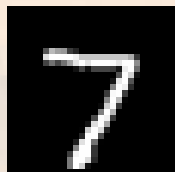
'7'



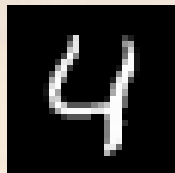
Nearest neighbor method

Training set

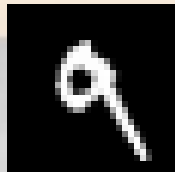
7



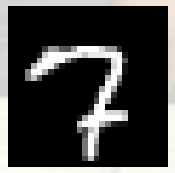
4



9



7



4



4

Nearest neighbor method

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


Nearest neighbor method

```
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))
```

Testing the algorithm

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

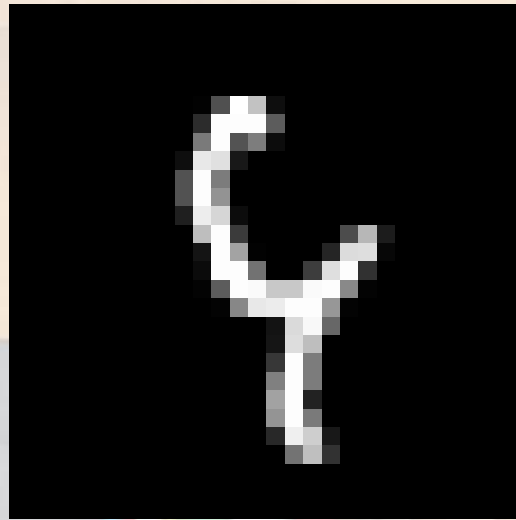
```
test_labels = labels[1000:2000]
```

```
predicted_class = map(classify, test_set)
```

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

=> 843/1000

9 or 4?

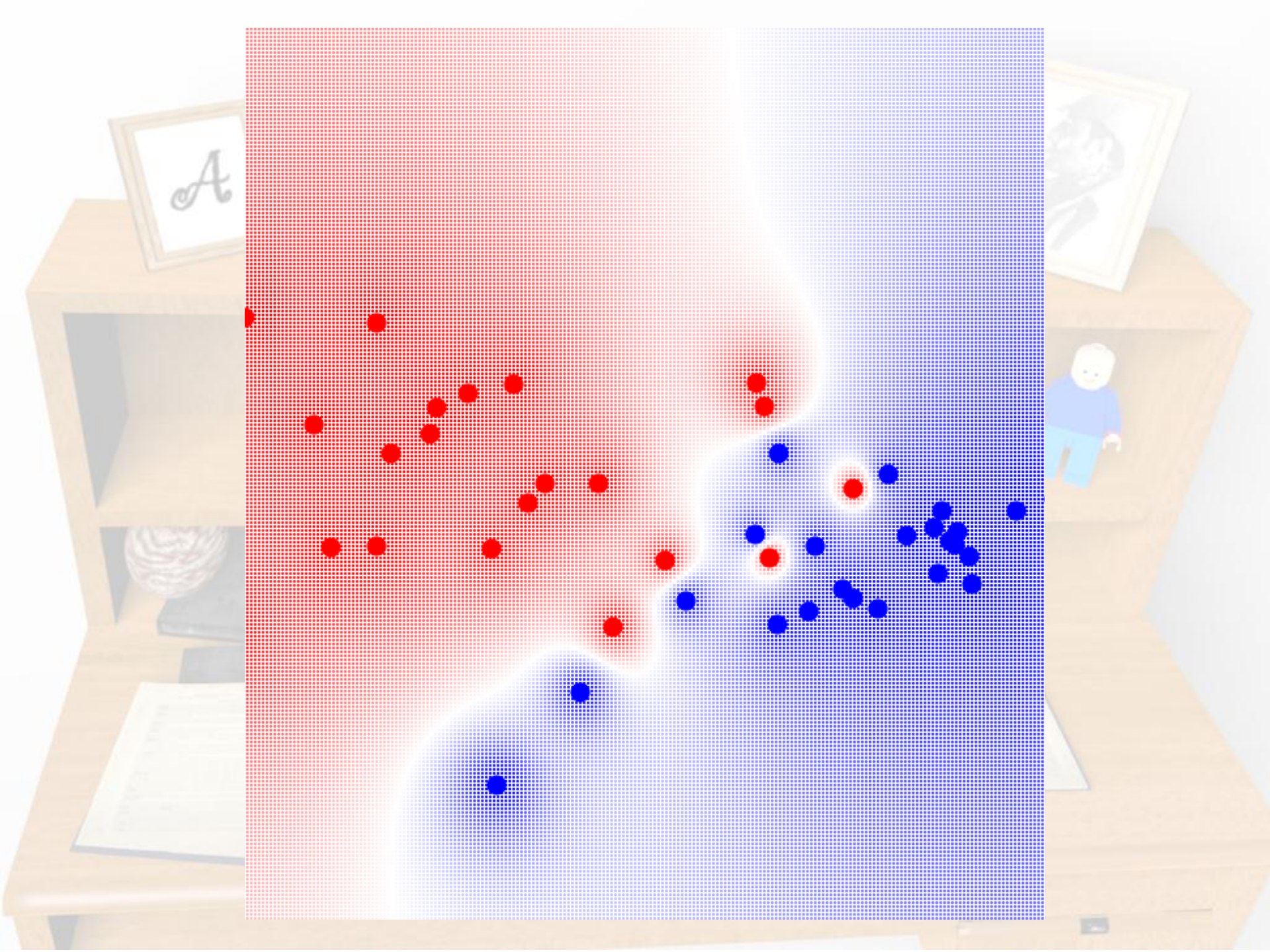


Scikit-learn

<http://scikit-learn.org/>

```
from sklearn.neighbors import  
                                KNeighborsClassifier  
  
clf = KNeighborsClassifier(n_neighbors=1)  
clf.fit(training_set, training_labels)  
  
predicted_class = clf.predict(test_set)
```

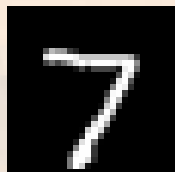
=> 855/1000



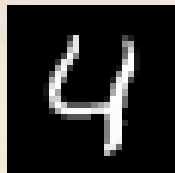
Nearest neighbor method

Training set

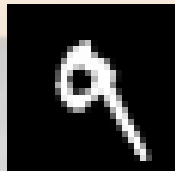
7



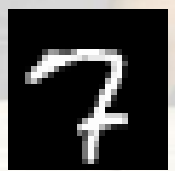
4



9



7



4

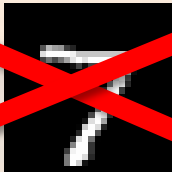


4

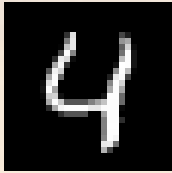
Some samples may be thrown away

Training set

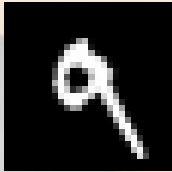
~~7~~



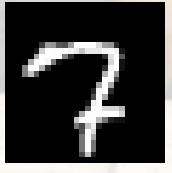
4



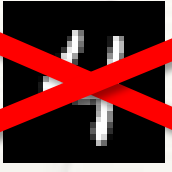
9



7



~~4~~

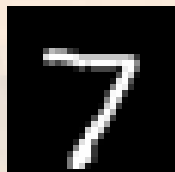


4

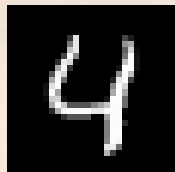
...or we can add “weights”

Training set

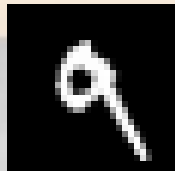
7



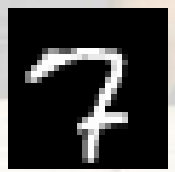
4



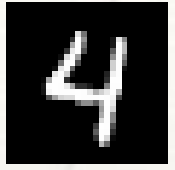
9



7



4



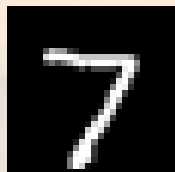
4

...or we can add “weights”

Training set

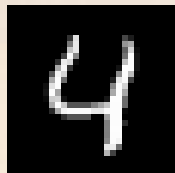
0.0

7



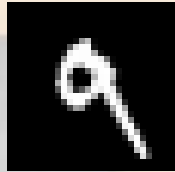
0.9

4



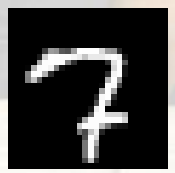
1.1

9



2.0

7



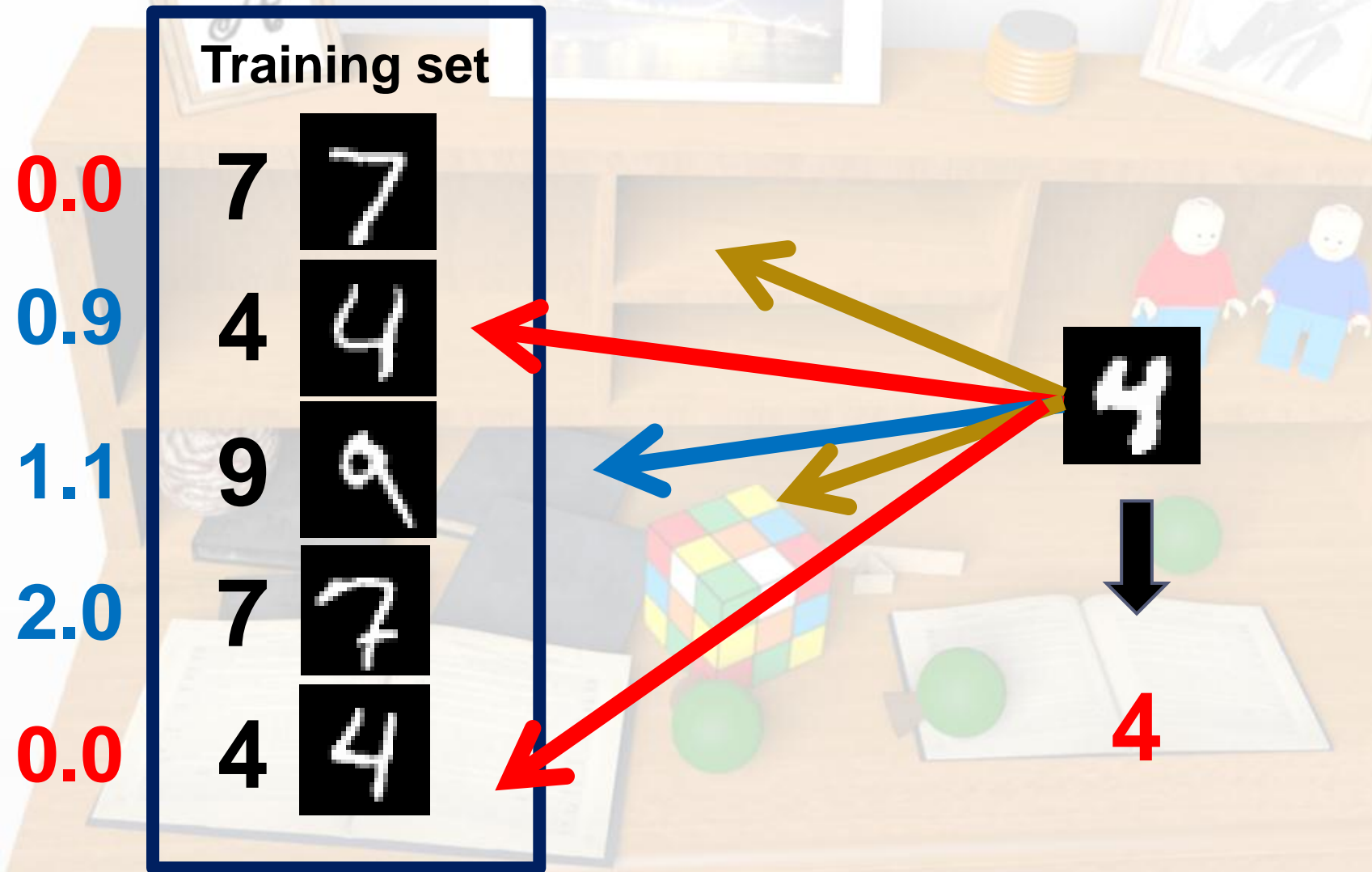
0.0

4



4

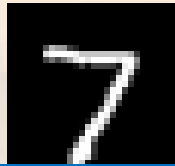
.. or we can SUM instead of MAX



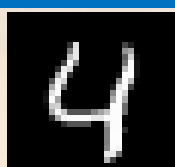
.. or we can SUM instead of MAX

Training set

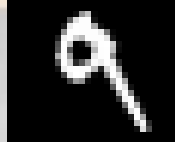
7



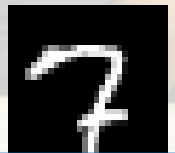
4



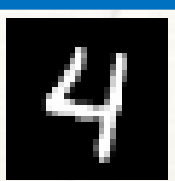
9



7



4



evidence(img, 4) =

0.9 * similarity(img, tr[1])

+

0.0 * similarity(img, tr[4])

= 34.0

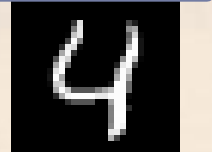
.. or we can SUM instead of MAX

Training set

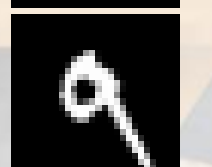
7



4



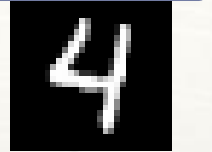
9



7



4



evidence(img, 7) =

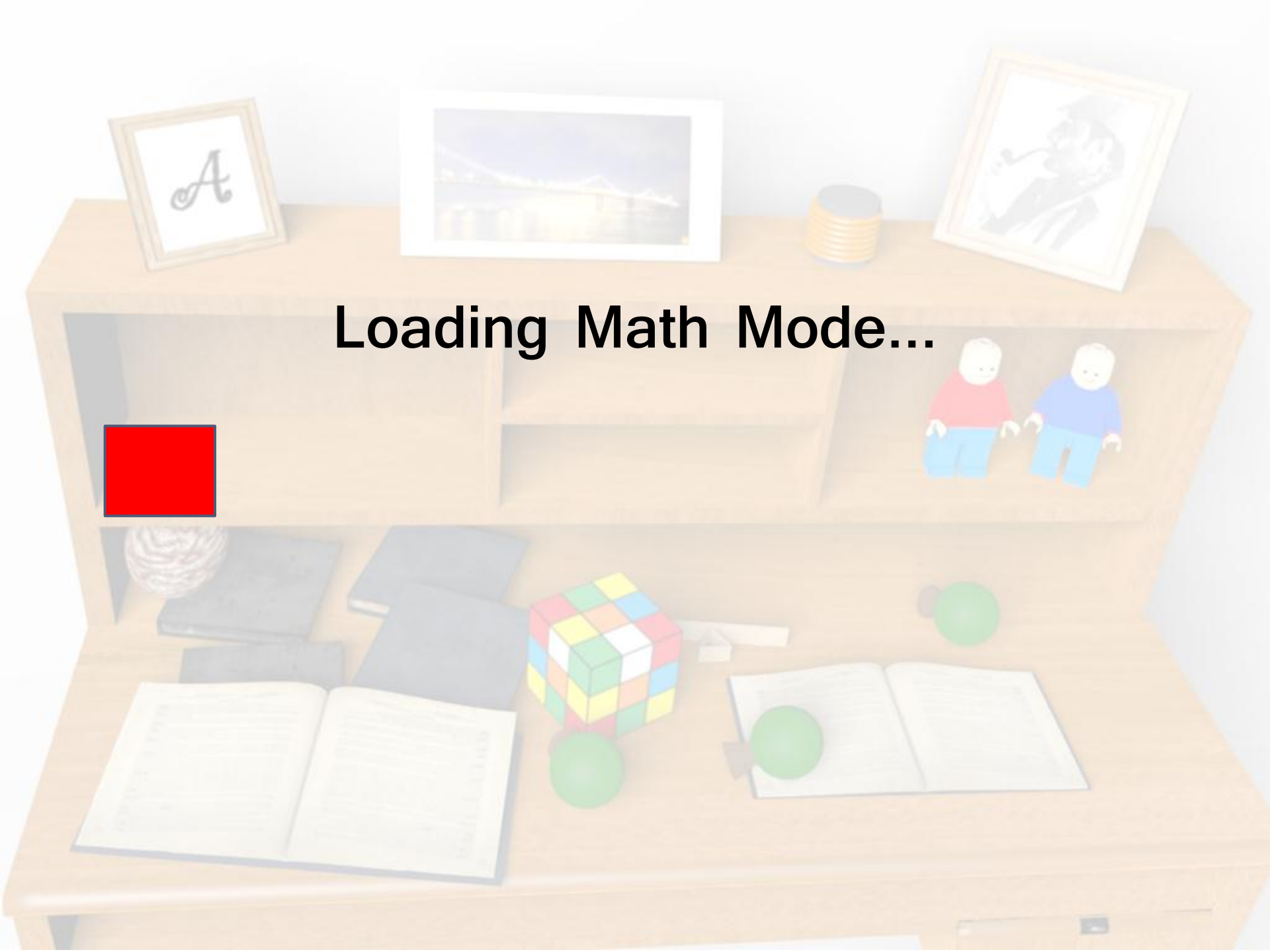
0.0 * similarity(img, tr[0])

+

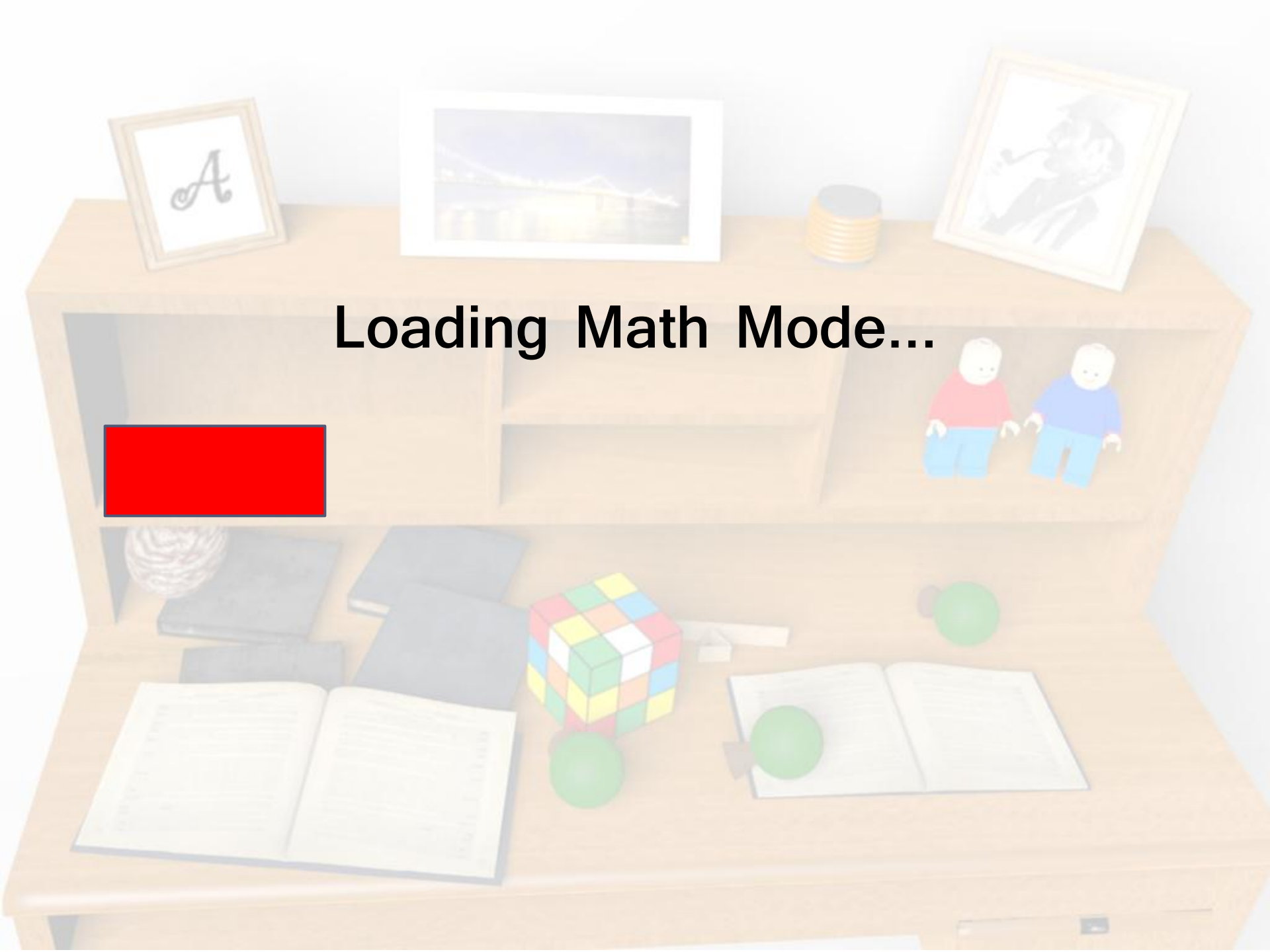
2.0 * similarity(img, tr[3])

= 20.0

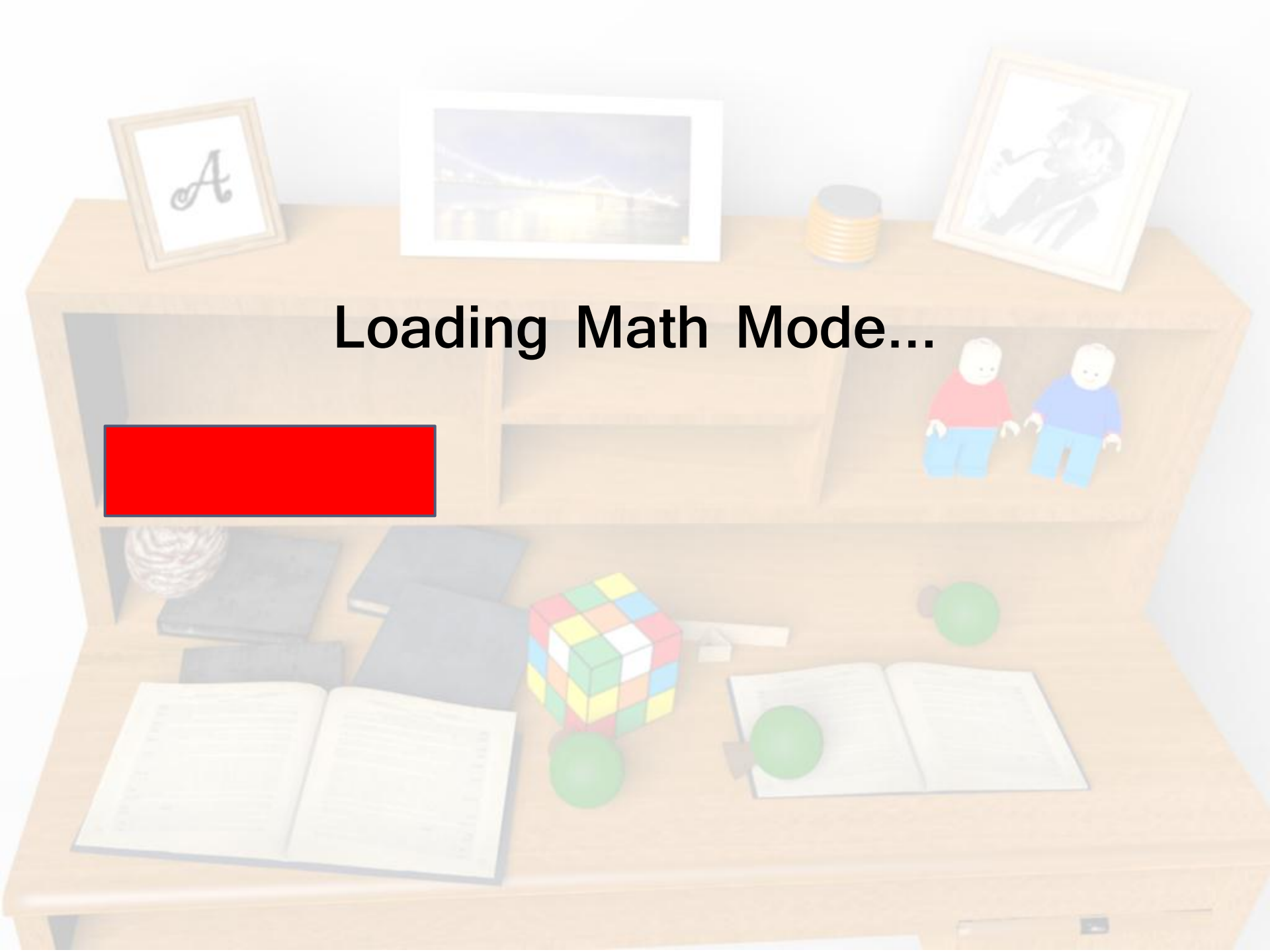
Loading Math Mode...



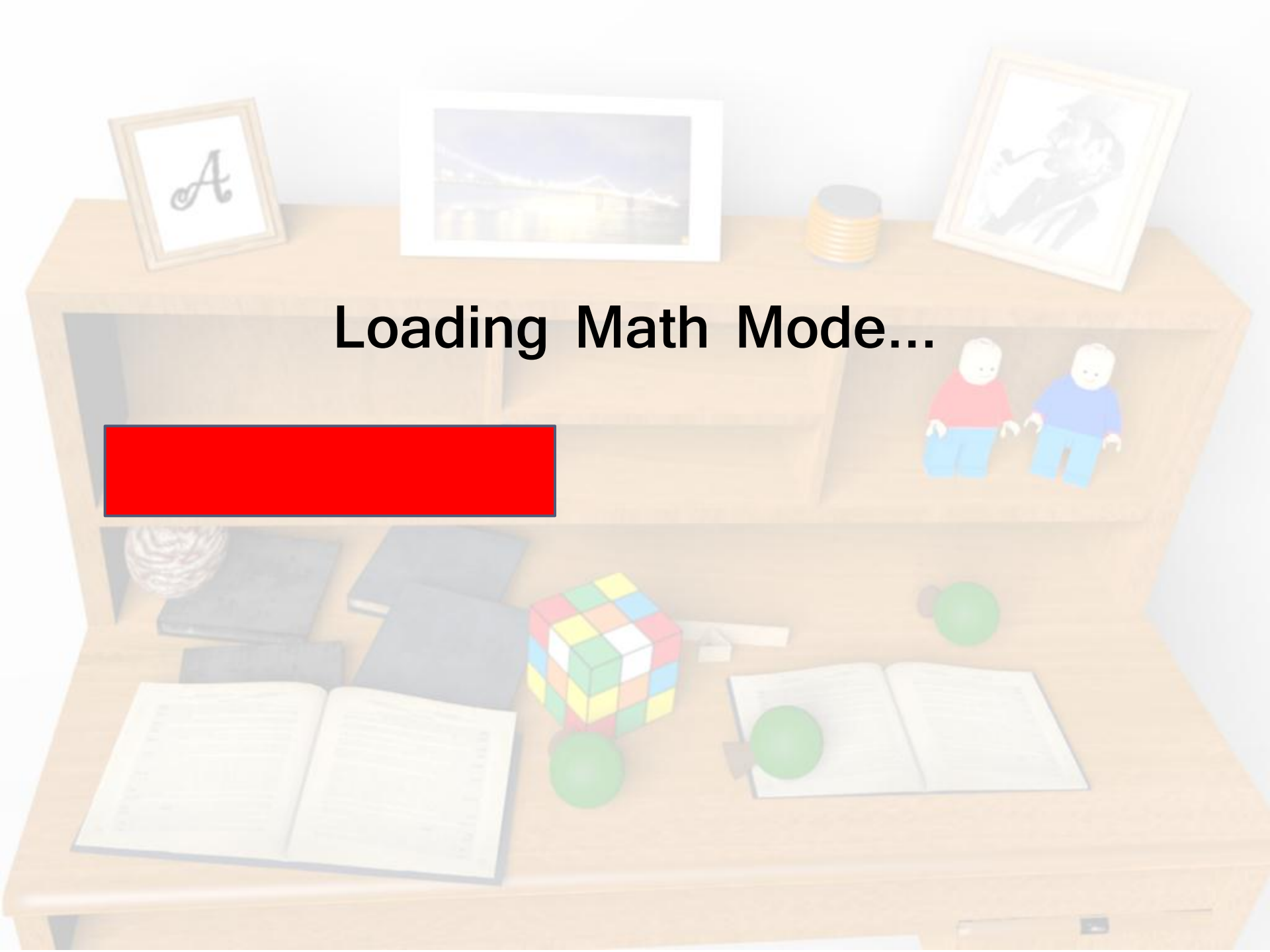
Loading Math Mode...



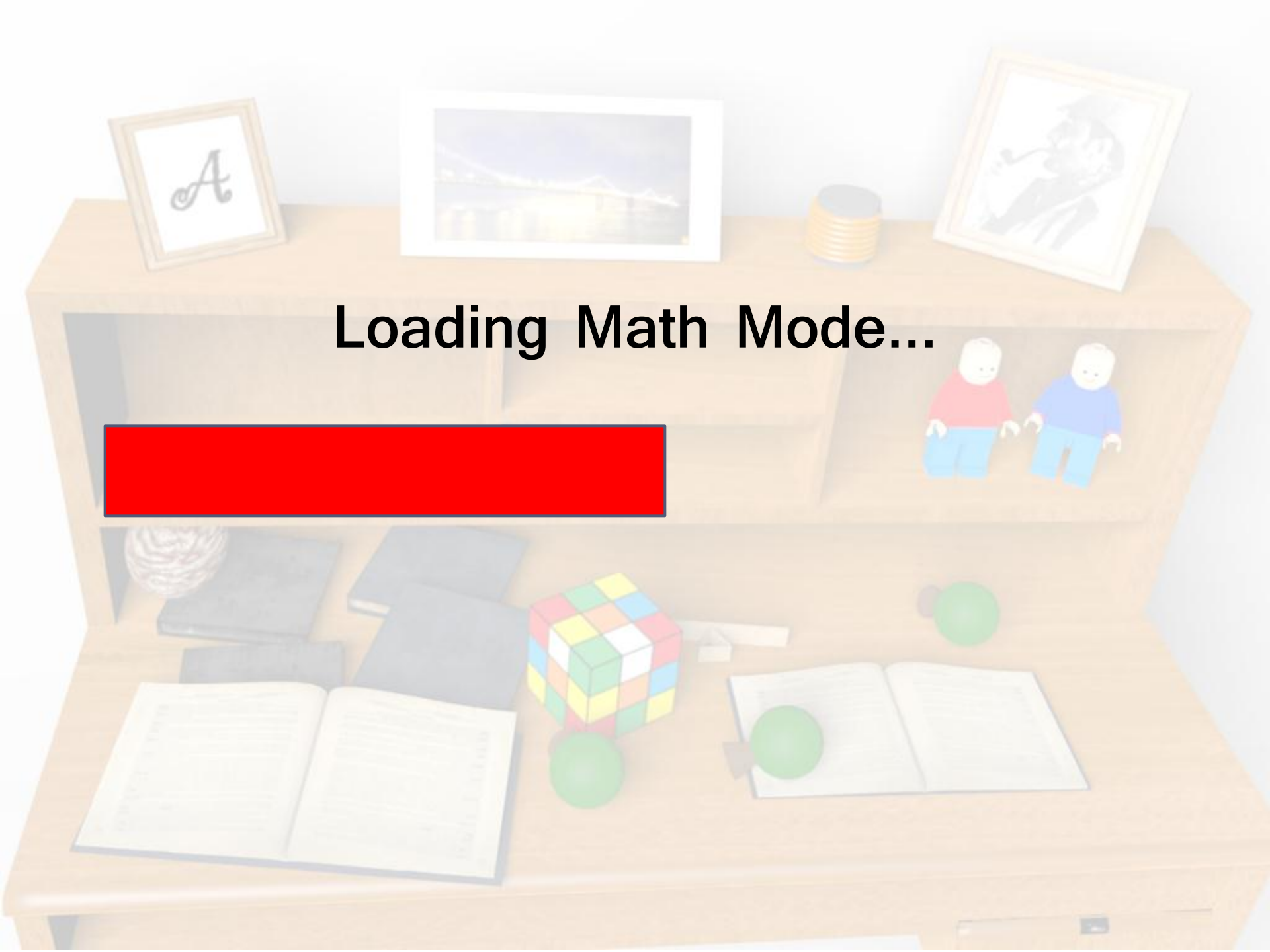
Loading Math Mode...



Loading Math Mode...



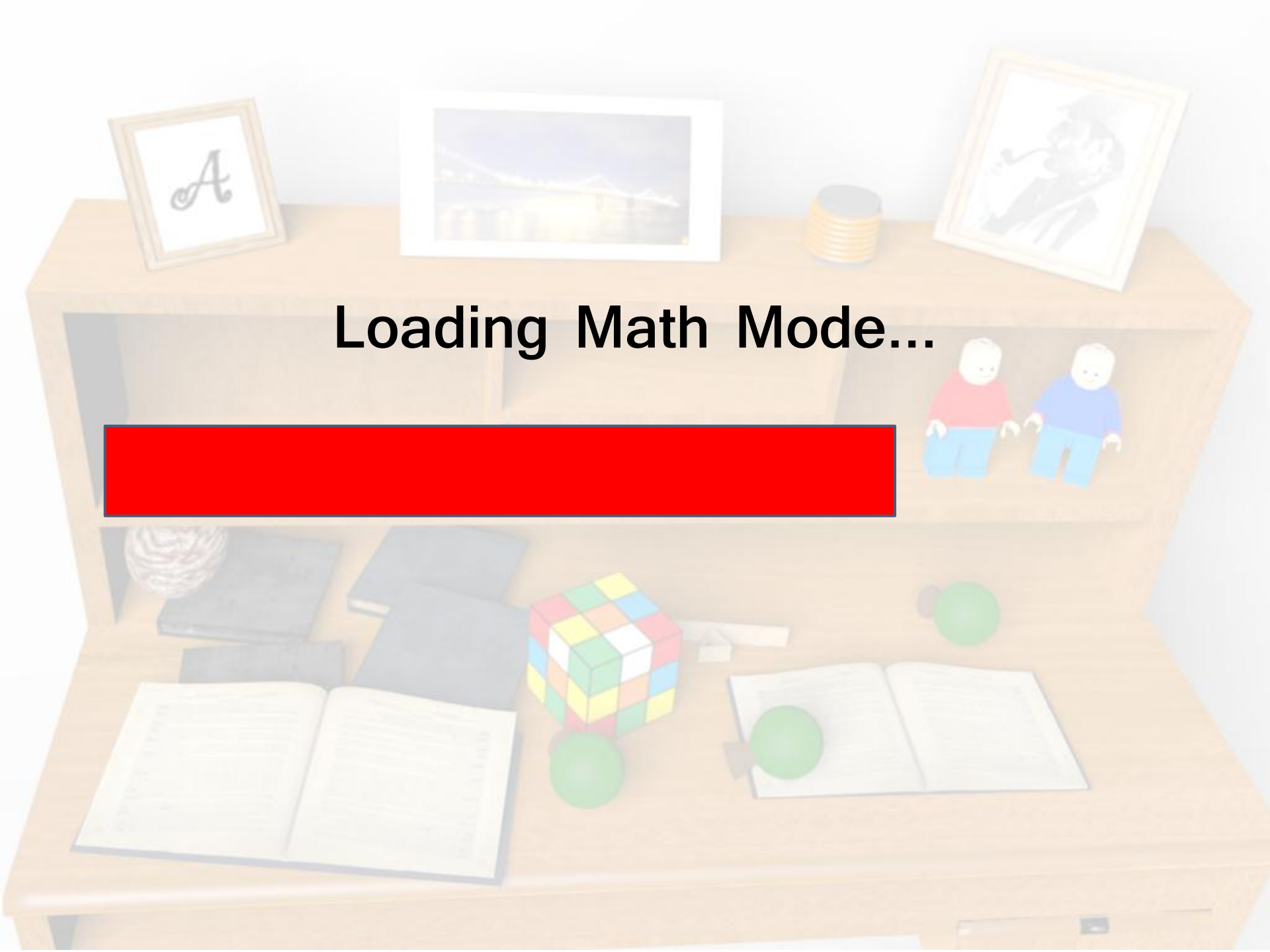
Loading Math Mode...



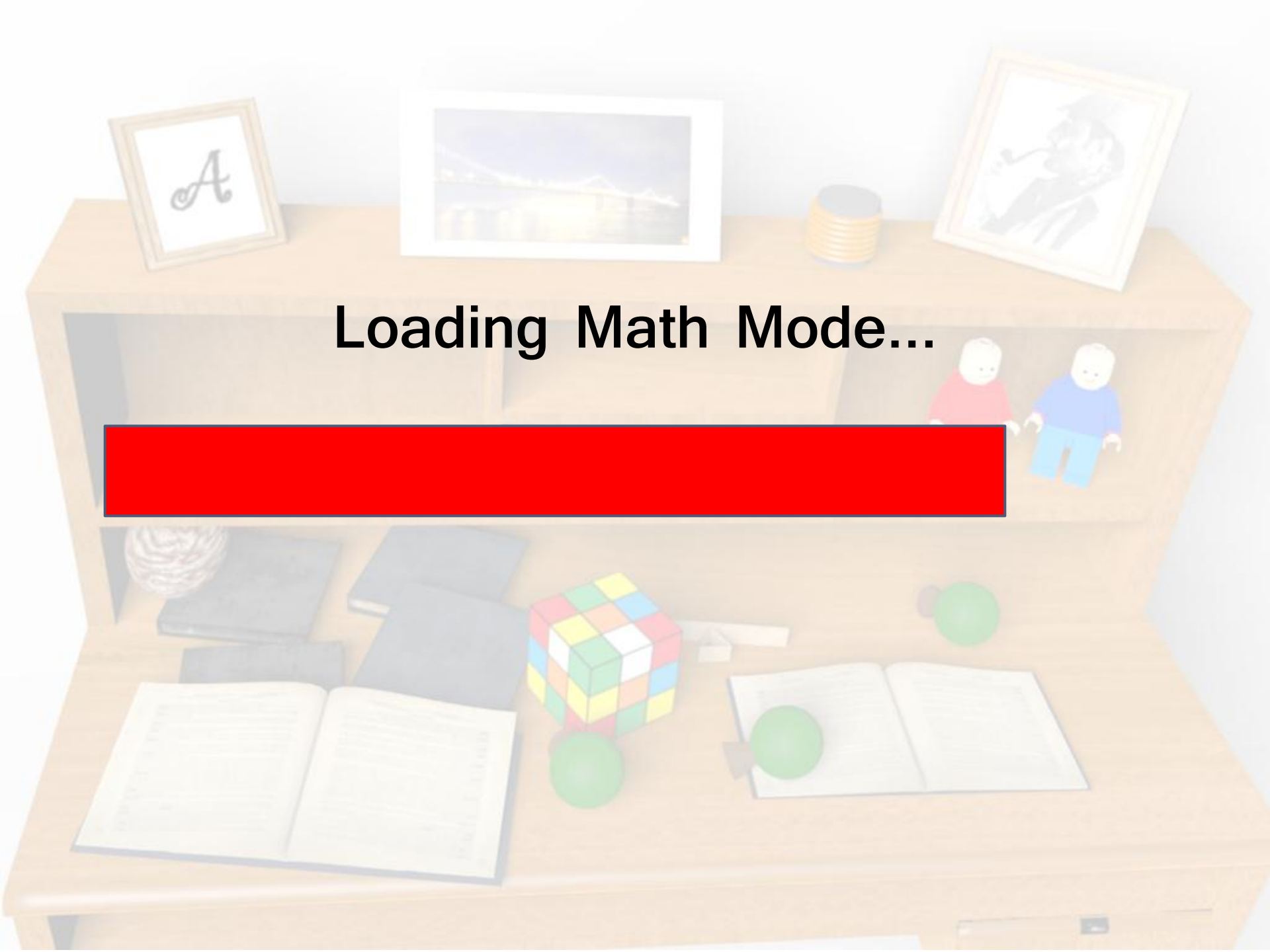
A wooden desk with various items. On the top shelf, there are three framed pictures: a letter 'A', a bridge at night, and a portrait of a man. Next to the portrait is a stack of yellow and grey discs. On the middle shelf, there are two small dolls, one in a red shirt and one in a blue shirt. A large red rectangular box obscures the middle of the desk. On the bottom shelf, there are several books, a Rubik's cube, and two green spheres. One book is open on the left, and another is open on the right.

Loading Math Mode...

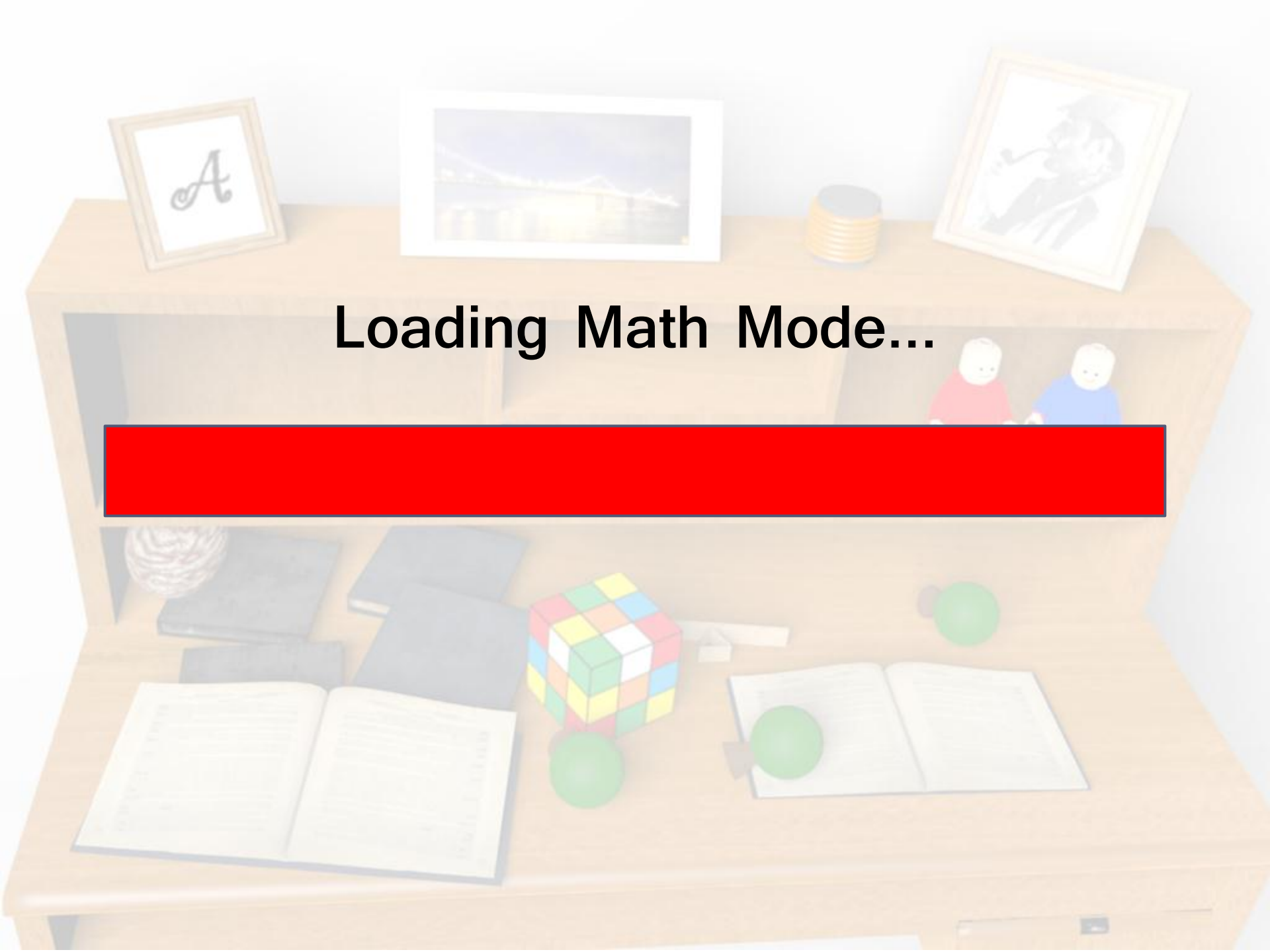
Loading Math Mode...



Loading Math Mode...



Loading Math Mode...



A wooden desk with various items. On the top shelf, there are three framed pictures: a letter 'A', a bridge at night, and a portrait of a man. Next to the portrait is a small stack of yellow and grey discs. In the middle shelf, there are two small white figures, one in a red shirt and one in a blue shirt. A large red rectangular bar obscures the middle of the desk. On the bottom shelf, there are several books, a Rubik's cube, and two green spheres. The text 'Loading Math Mode...' is overlaid on the top shelf, and 'Done!' is overlaid on the bottom shelf.

Loading Math Mode...

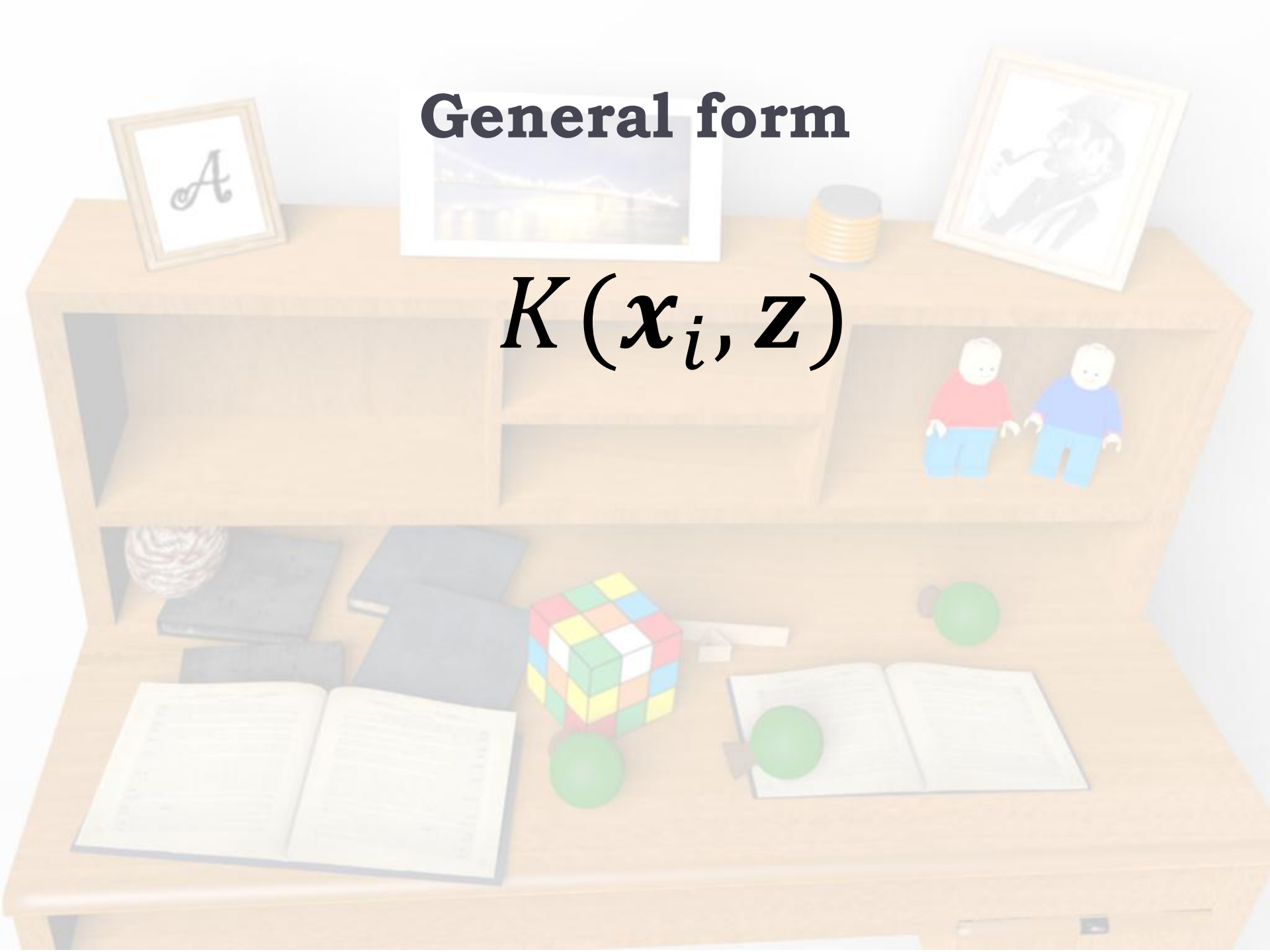
Done!

General form

$$f_w(\mathbf{z}) = \sum_i w_i y_i K(\mathbf{x}_i, \mathbf{z})$$

General form

$$K(x_i, z)$$



General form

$$K(x_i, z)$$

$$K(x_j, z)$$

General form

$$\sum_i w_i K(x_i, z)$$

$$\sum_j w_j K(x_j, z)$$

General form

$$\sum_i w_i K(x_i, z)$$

$$- \sum_j w_j K(x_j, z)$$

General form

$$\sum_i w_i y_i K(x_i, z)$$

$$+ \sum_j w_j y_j K(x_j, z)$$

General form

$$f_w(\mathbf{z}) = \sum_i w_i y_i K(\mathbf{x}_i, \mathbf{z})$$

How to find the weights?

$$f_{\mathbf{w}}(\mathbf{z}) = \sum_i \boxed{w_i} y_i K(\mathbf{x}_i, \mathbf{z})$$

- Find weights, such that the **misclassification error rate** on the training set is **the smallest**.

$$\mathbf{w} = \operatorname{argmin}_{\mathbf{w}} \operatorname{ErrorRate}(\mathbf{f}_{\mathbf{w}}, \text{Data})$$

How to find the weights?

$$f_{\mathbf{w}}(\mathbf{z}) = \sum_i \boxed{w_i} y_i K(\mathbf{x}_i, \mathbf{z})$$

- Find weights, such that the **approximation to misclassification error on the training set** is **the smallest**.

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

How to find the weights?

$$f_w(z) = \sum_i w_i y_i K(x_i, z)$$

- Find weights, such that the **error rate on the training set** is **the smallest + there are many zero weights.**

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

Support Vector Machine

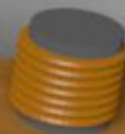
```
from sklearn.svm import SVC
```

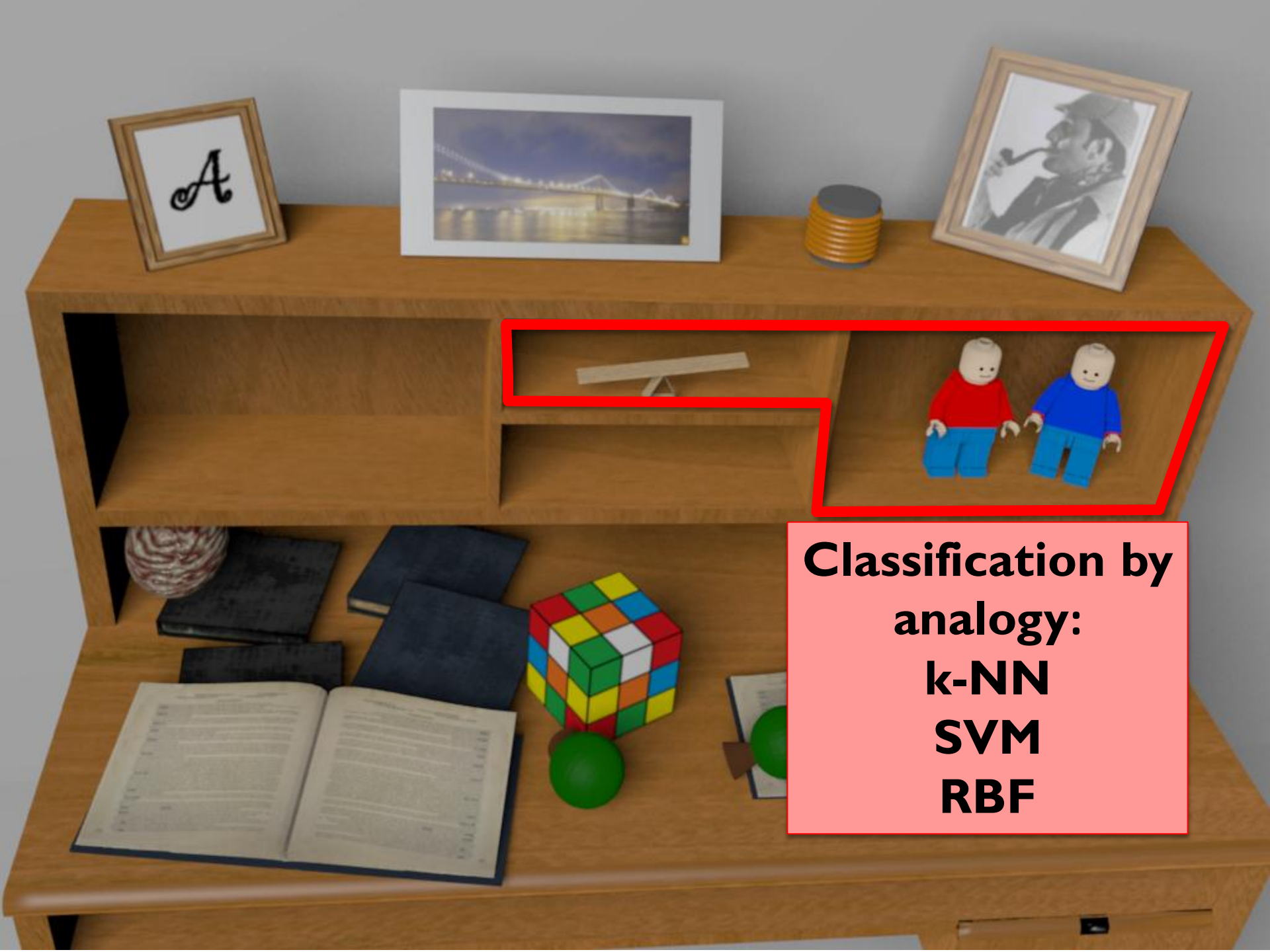
```
clf = SVC(kernel='linear')
```

```
clf.fit(training_set, training_labels)
```

```
predicted_class = clf.predict(test_set)
```

=> 865/1000





**Classification by
analogy:
k-NN
SVM
RBF**

Search for the nearest neighbor

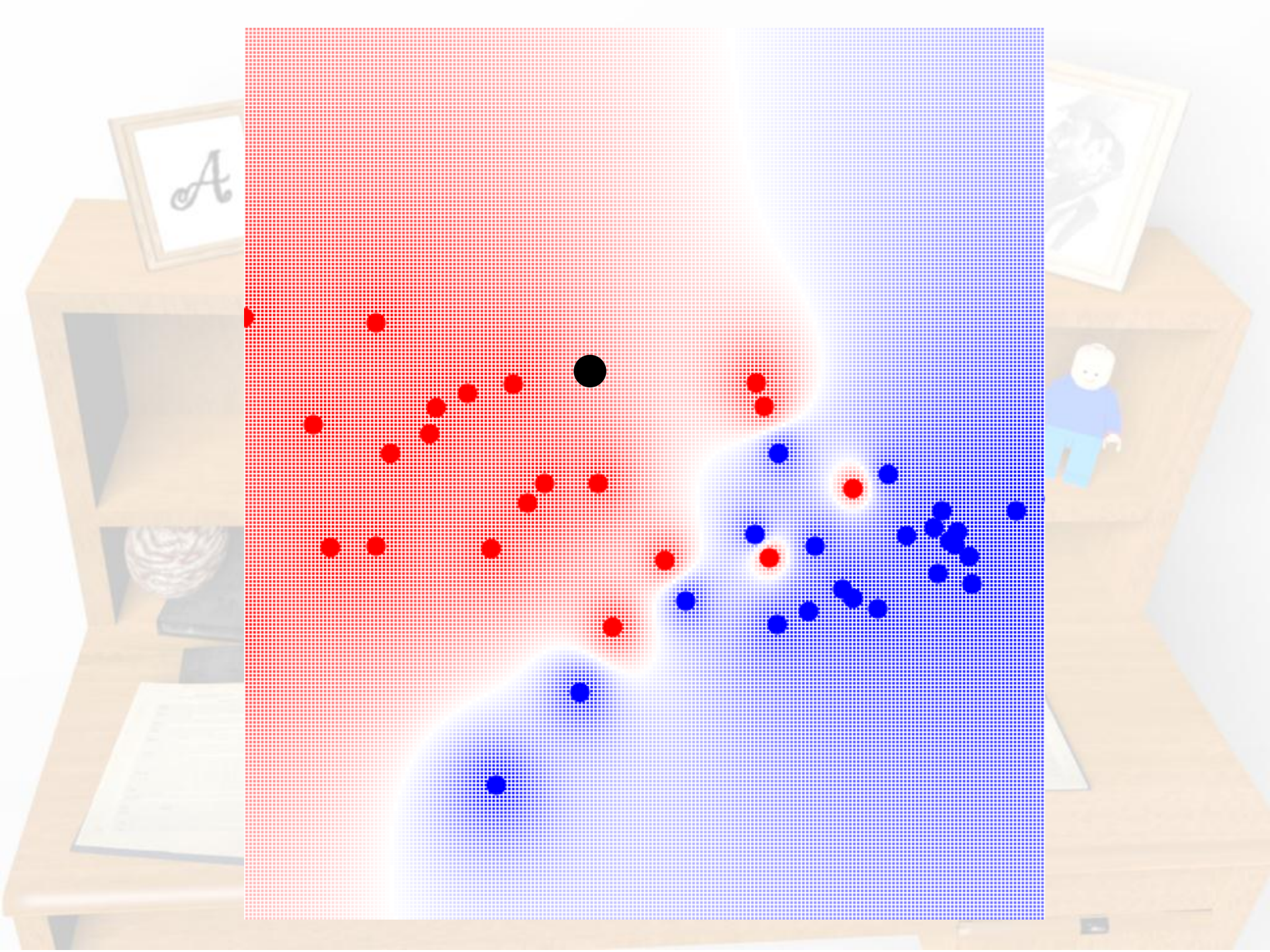
```
def classify(img):  
    similarities =  
        [similarity(img, p) for p in training_set]  
    i = similarities.index(max(similarities))  
    return training_labels[i]
```

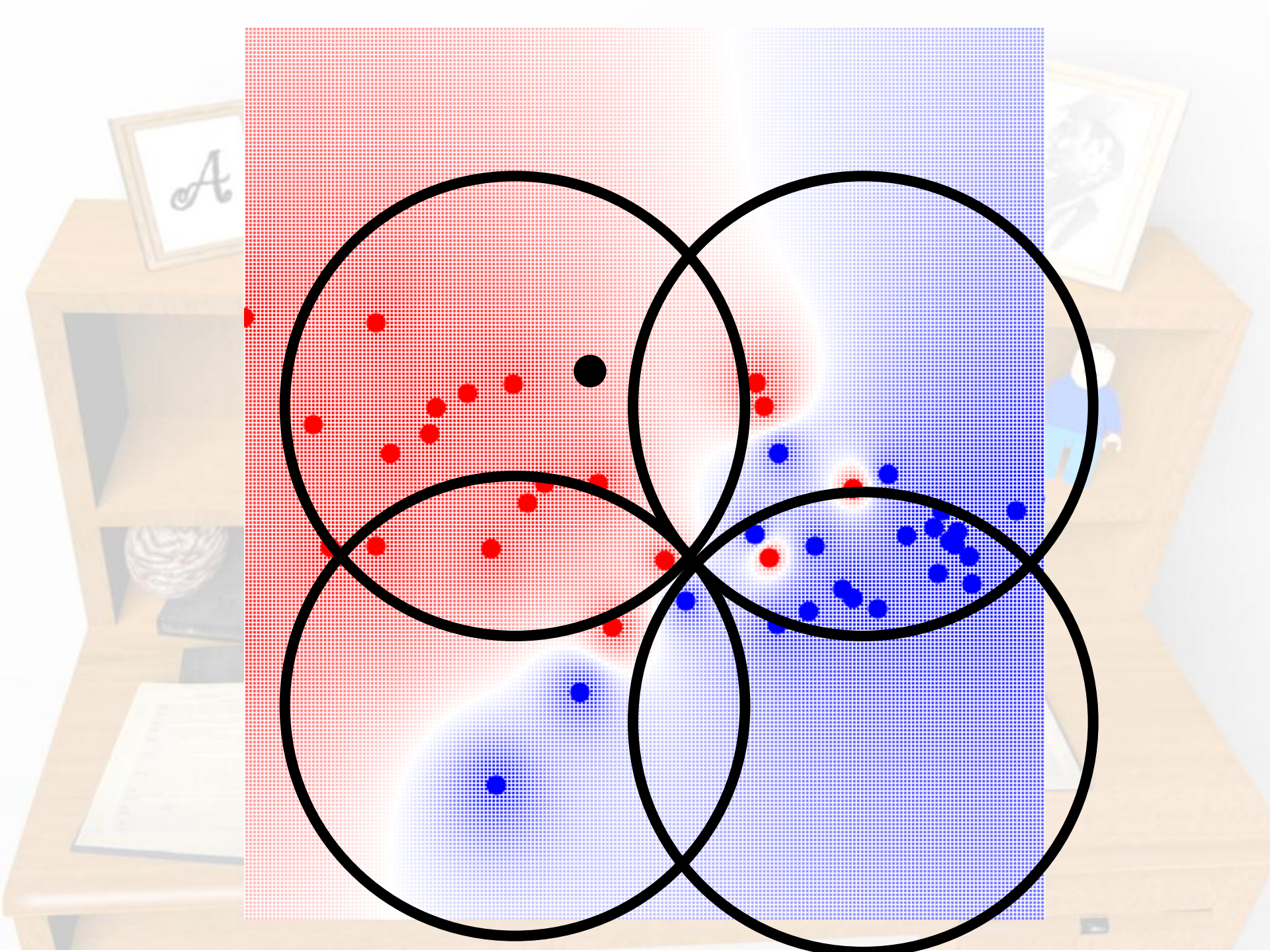
Inefficient

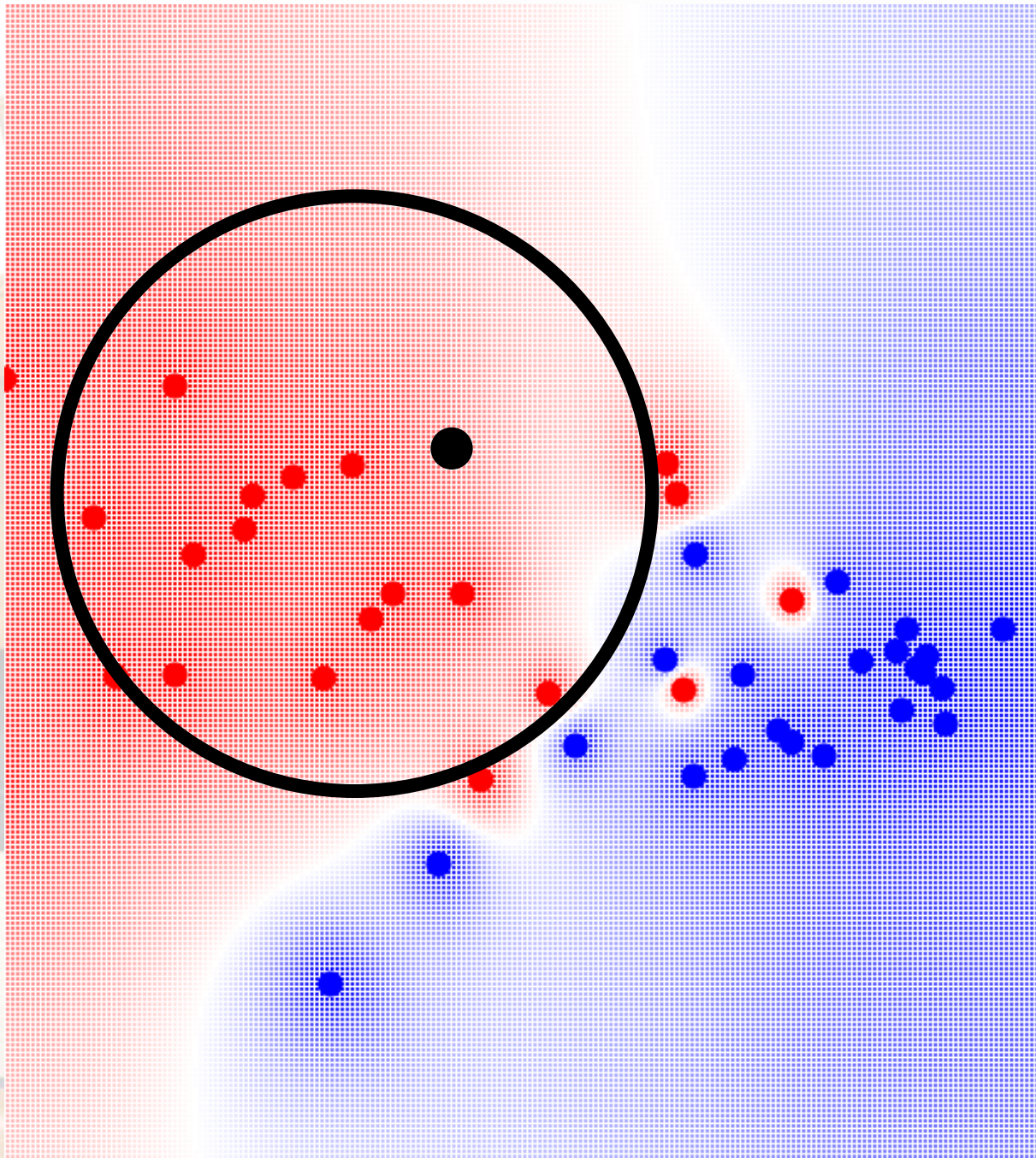
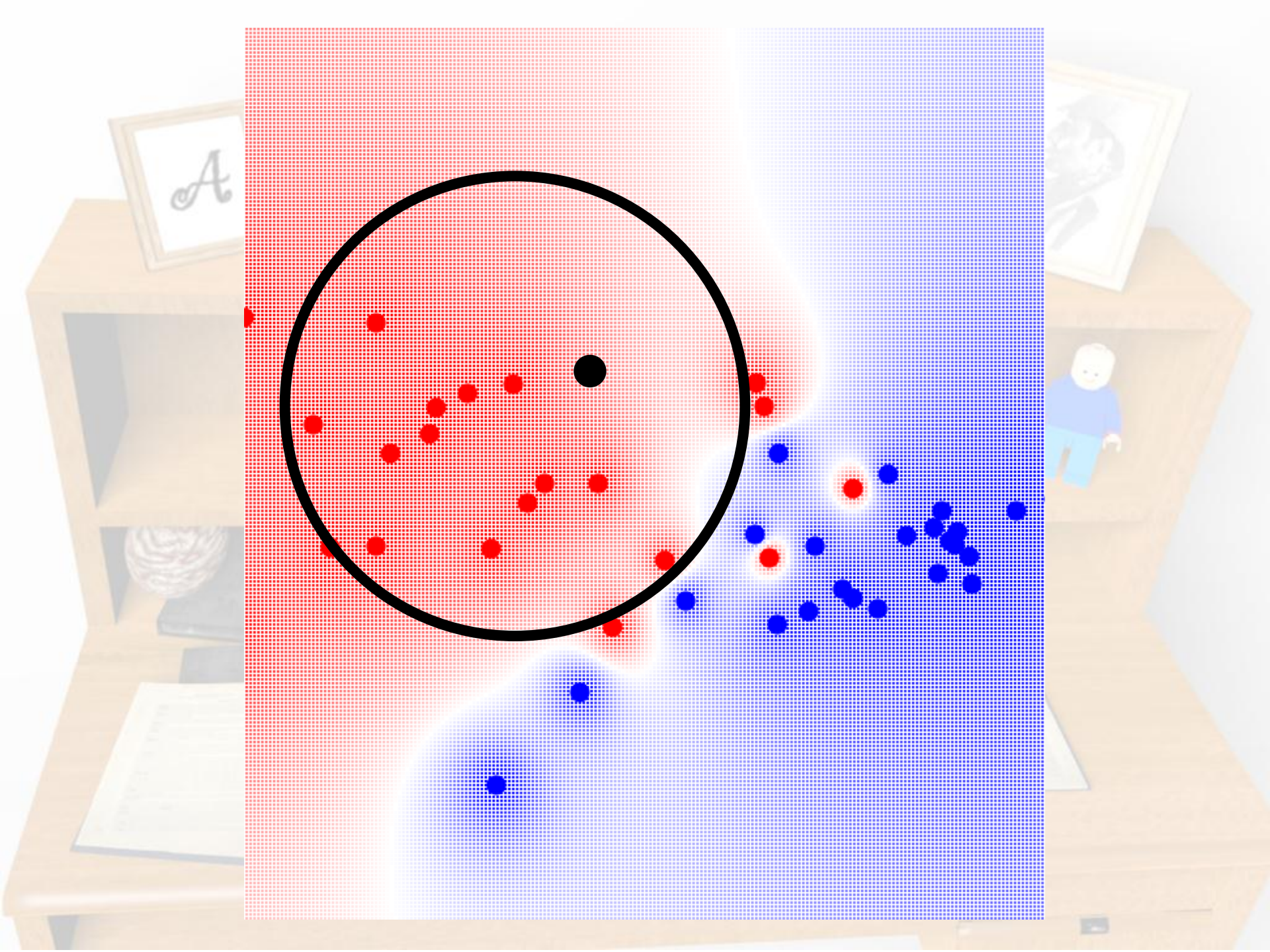
Search for the nearest neighbor

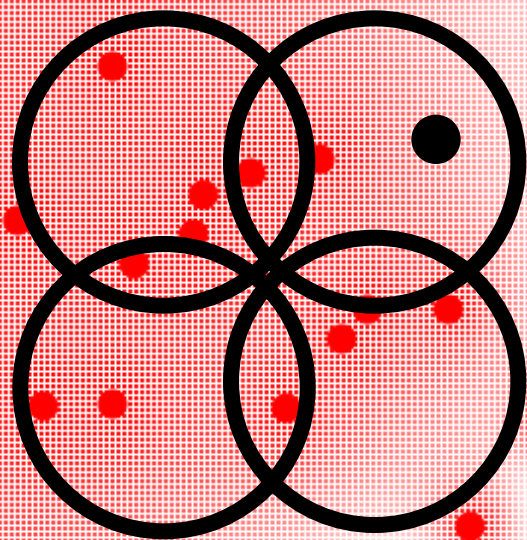
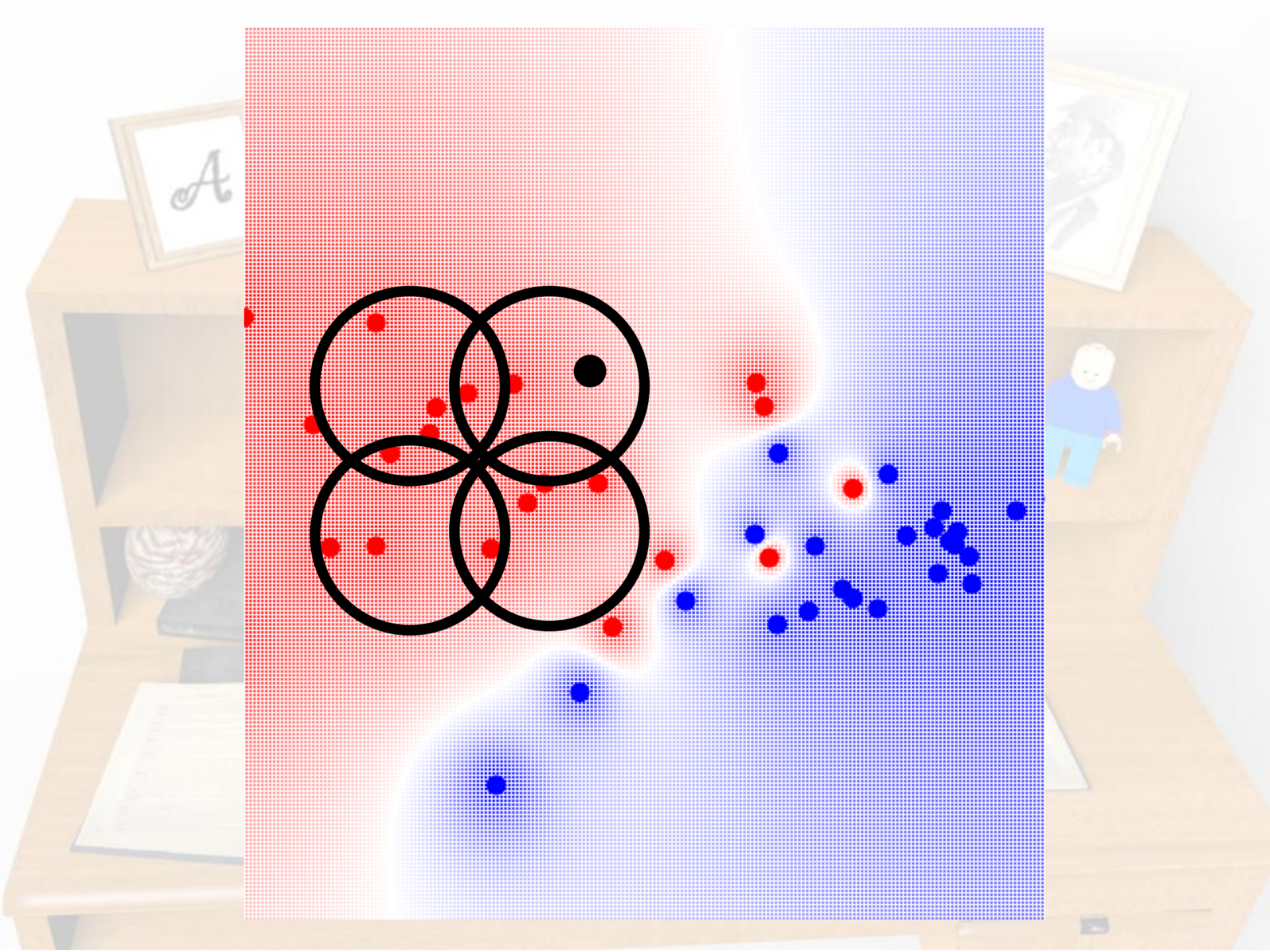
```
def classify(img):  
    nearest_neighbour =  
        training_set.find_nearest_neighbour(img)  
  
    return nearest_neighbour.label
```

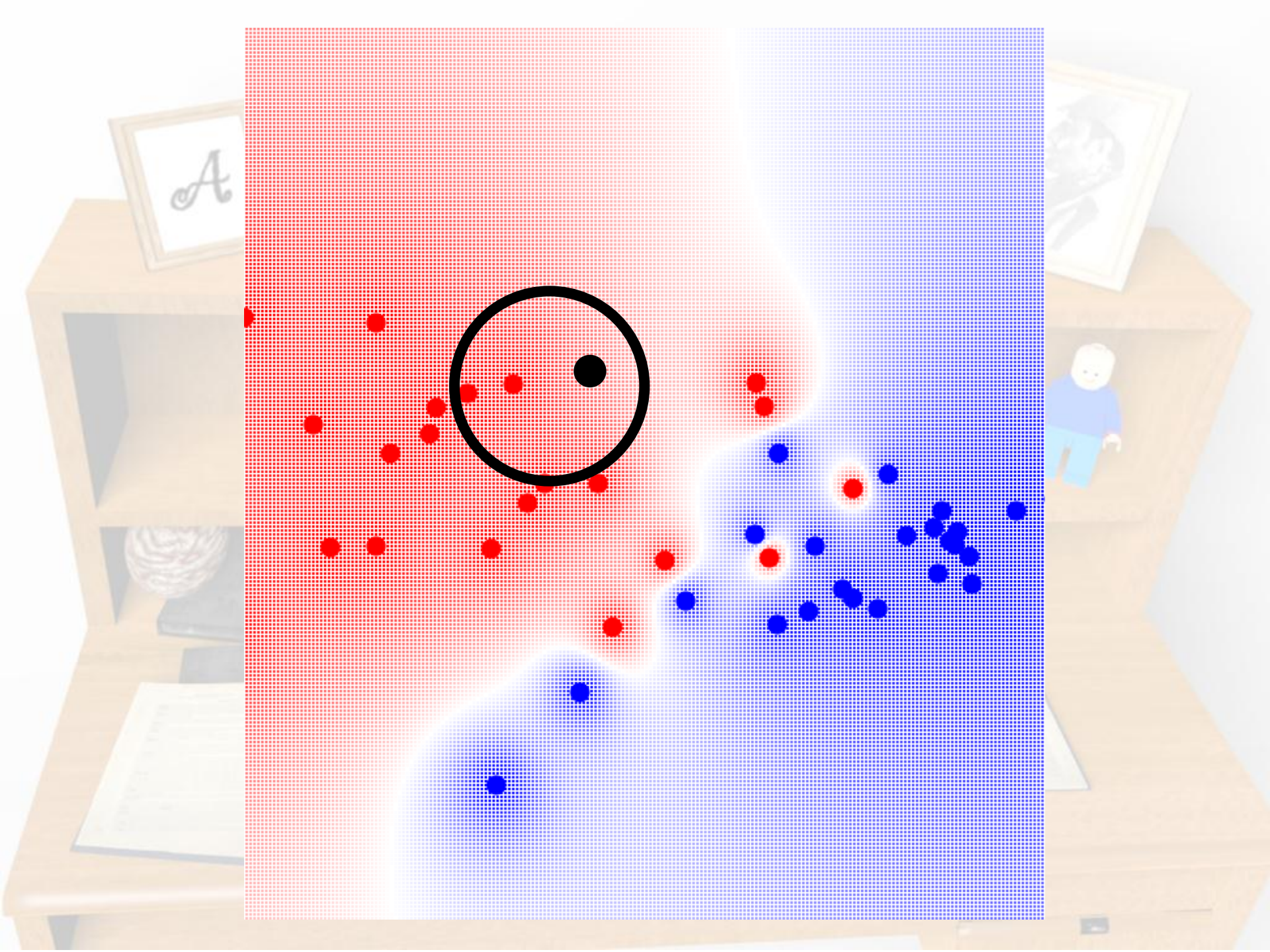
Indexing!











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'
```

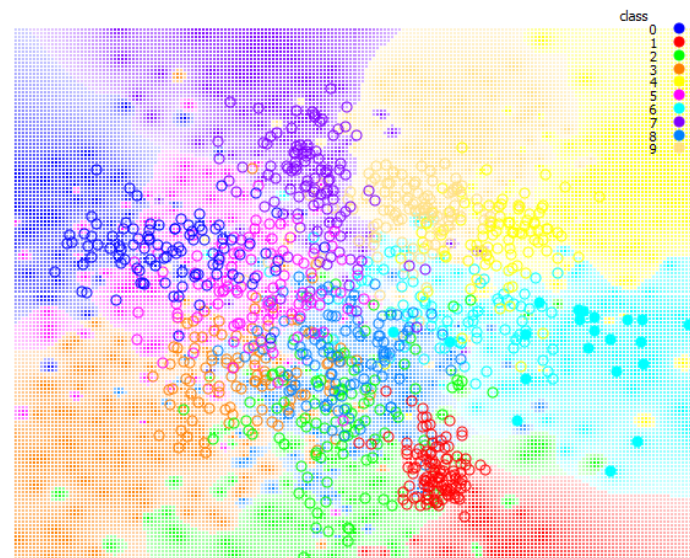
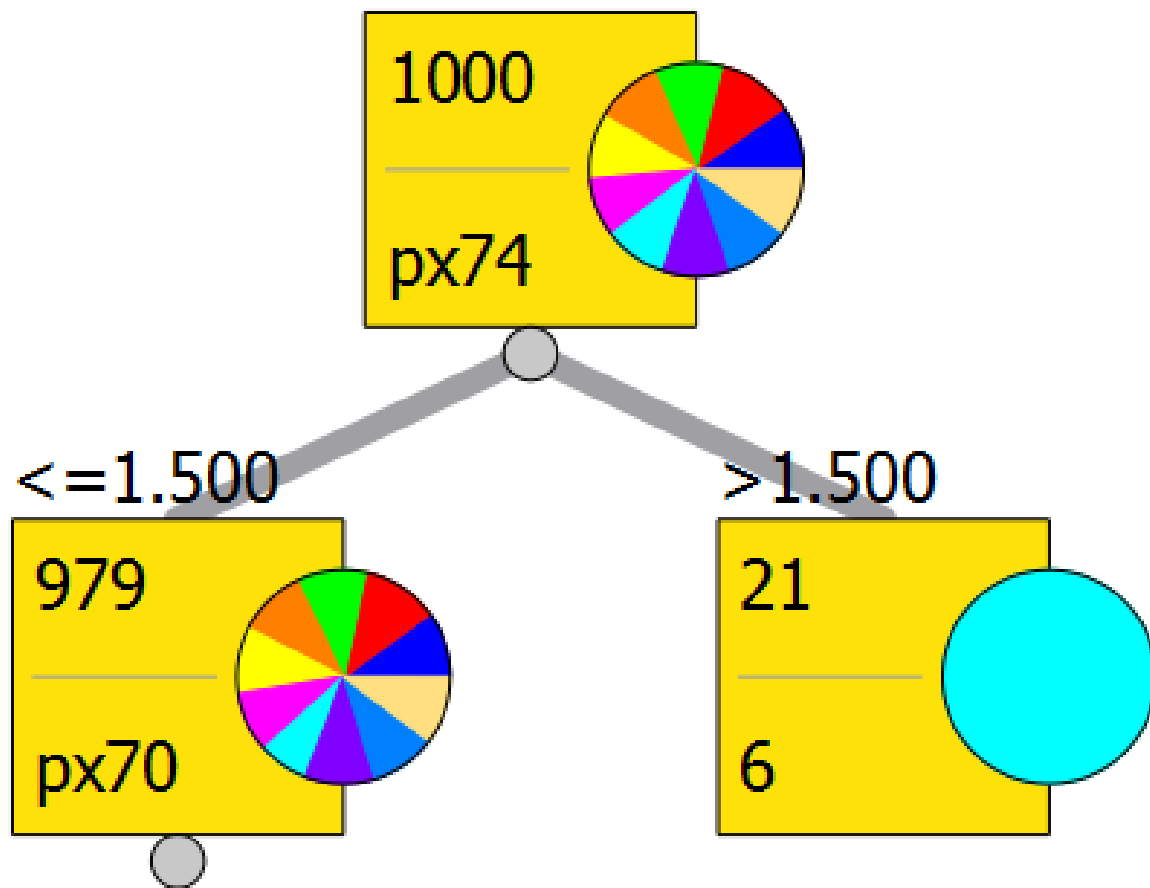
```
    else:
```

```
        class = '2'
```

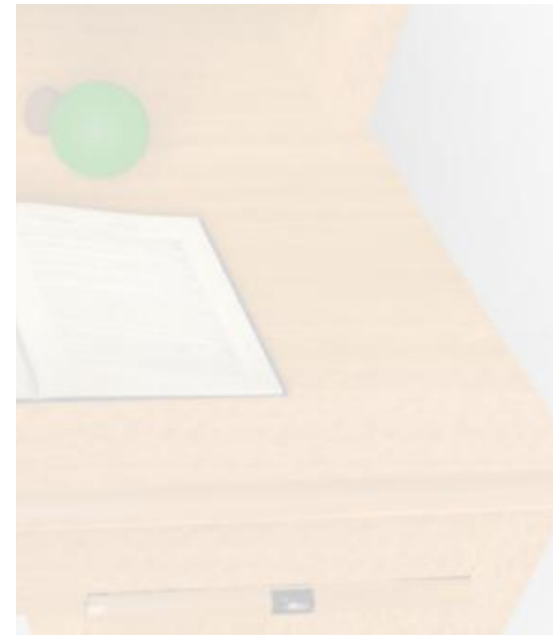
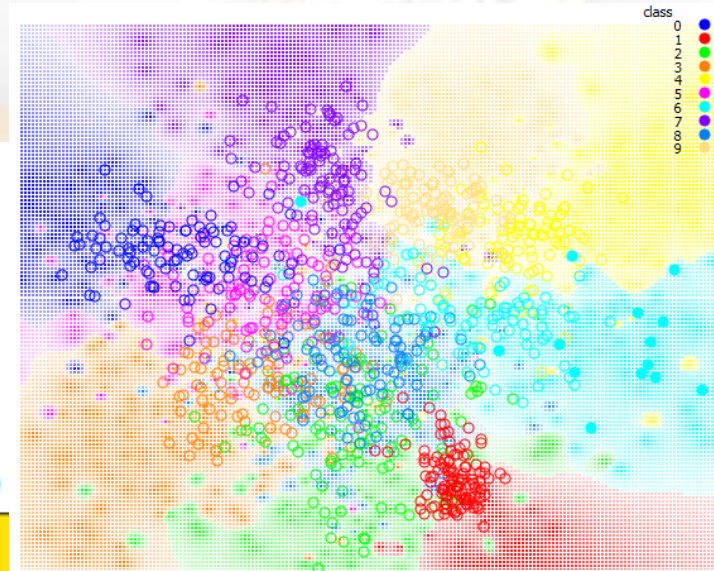
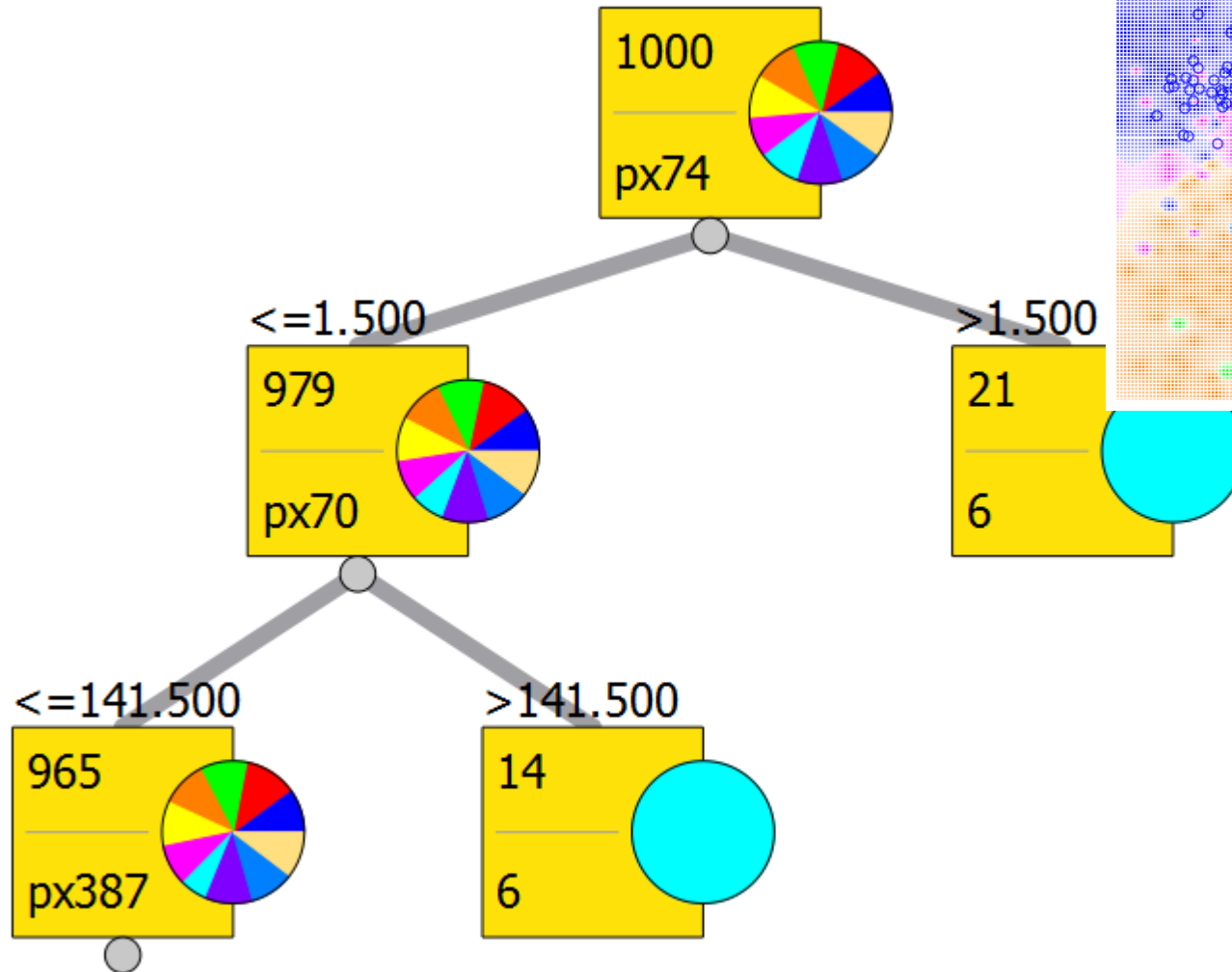
```
else:
```

```
    class = '3'
```

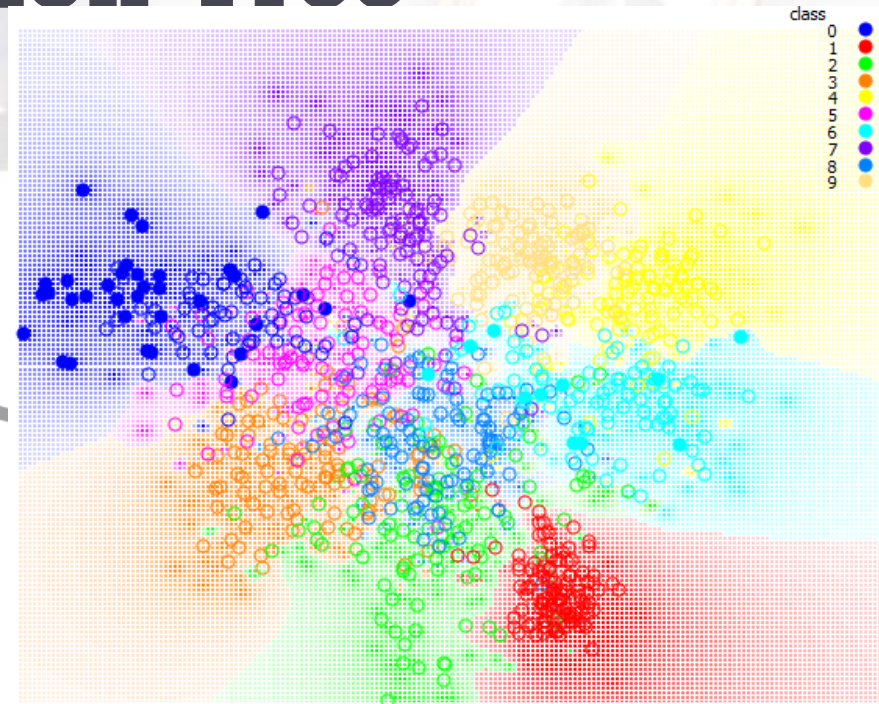
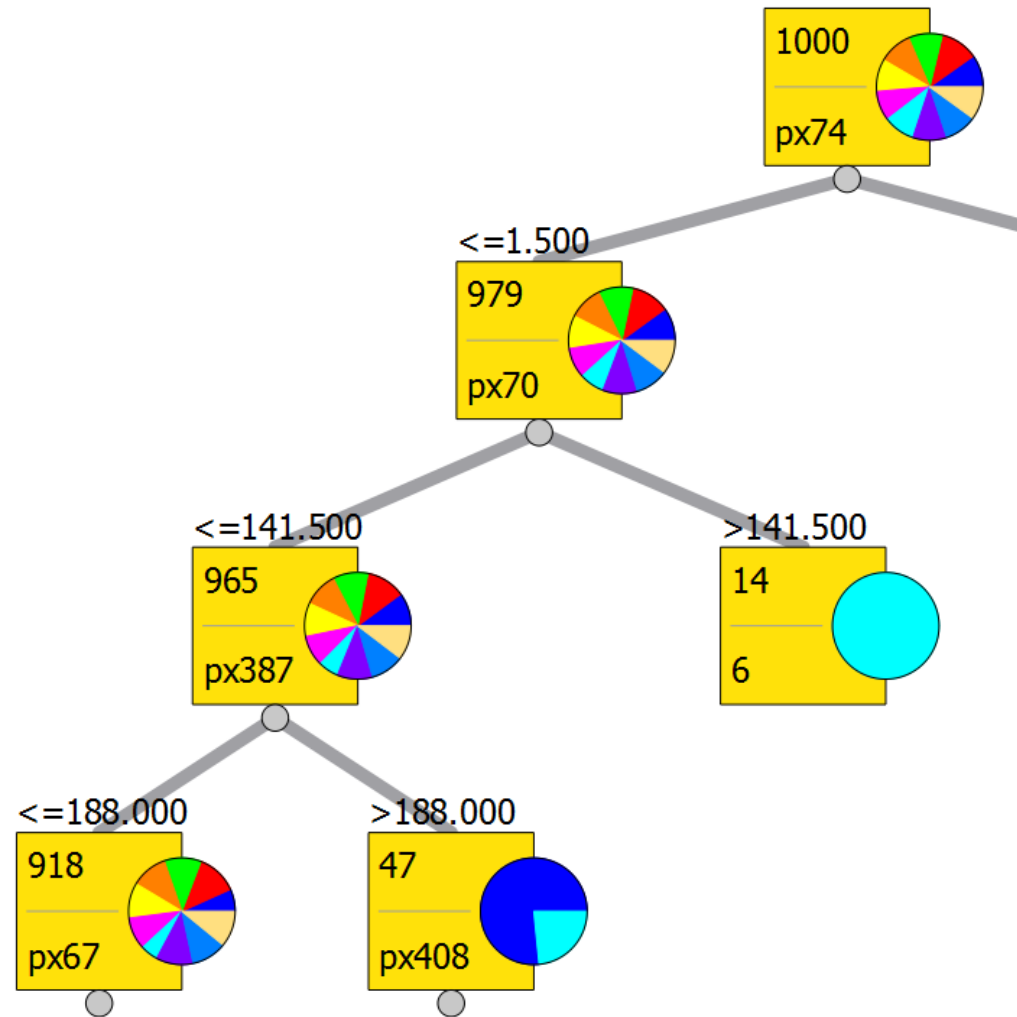
Classification Tree



Classification Tree

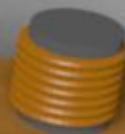


Classification Tree



Orange

<http://orange.biolab.si/>





Trees
ID3
C4.5
RegTree



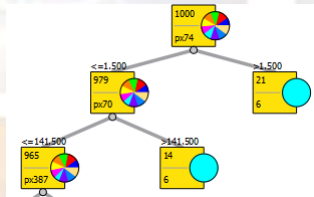
General form of a model

$$f_w(\mathbf{z}) = \sum_i w_i y_i K(\mathbf{x}_i, \mathbf{z})$$



**Search for optimal model
parameters**

General form of a model



Search for the optimal tree

A wooden desk with various items. On the top shelf, there are three framed pictures: a letter 'A', a landscape, and a portrait of a man. Next to the portrait is a stack of gold coins. The middle shelf has two green spherical toys and two small figures. The bottom shelf has several books, some open, and a colorful geometric toy. A large black arrow points from the top shelf to the bottom shelf.

Modeling

Optimization



Linear model

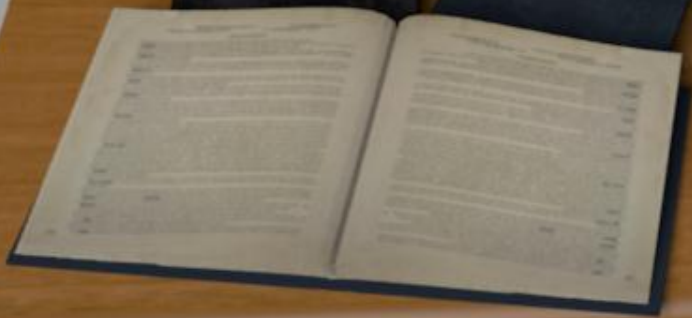
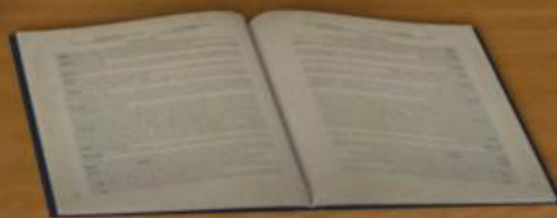
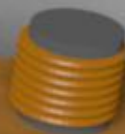
$f(\text{image}) =$

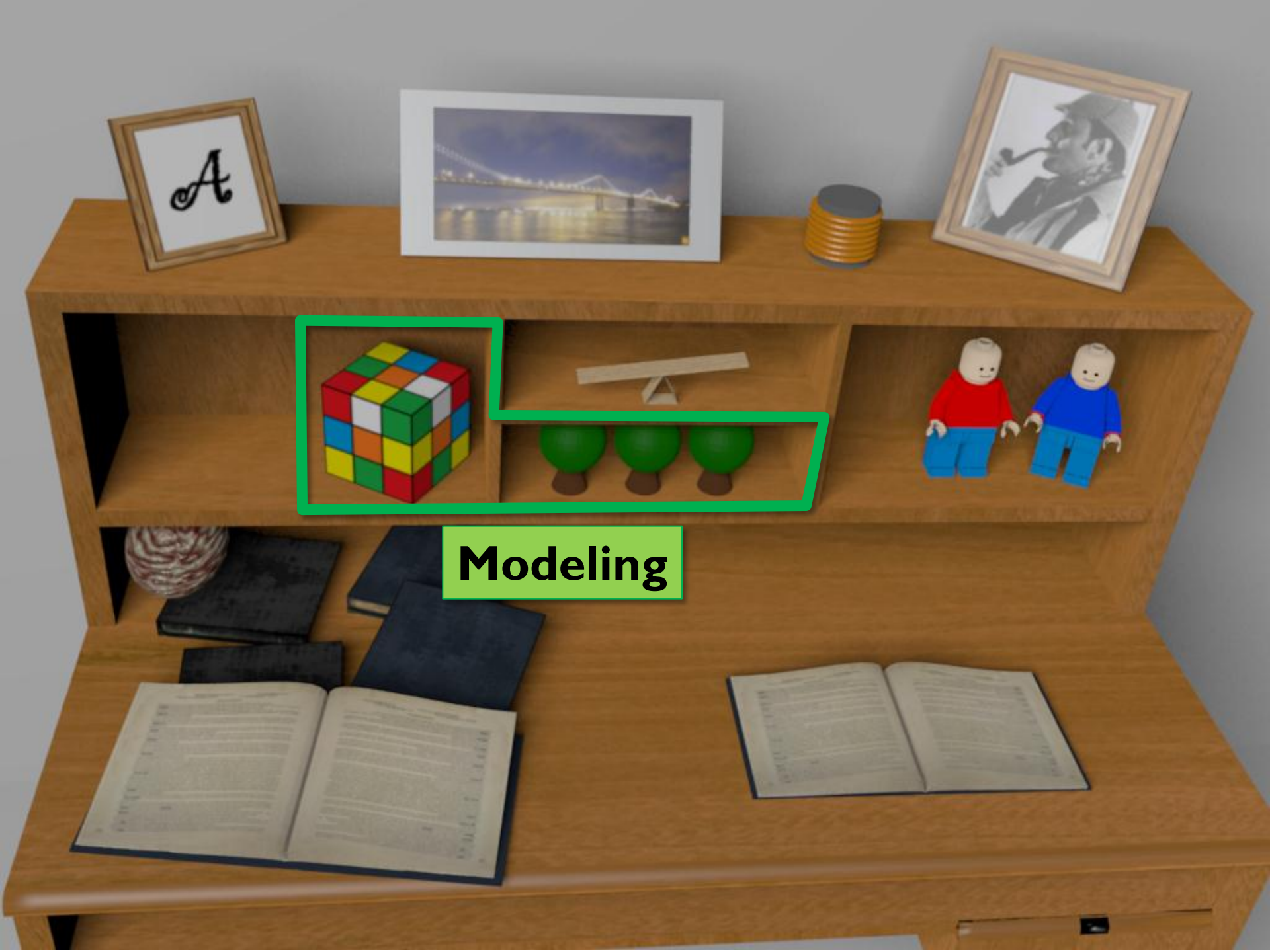
$$\text{pixel1} * w1 + \text{pixel2} * w2 + \dots + \text{pixel784} * w784$$

Linear Classification

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

=> 809/1000





Modeling

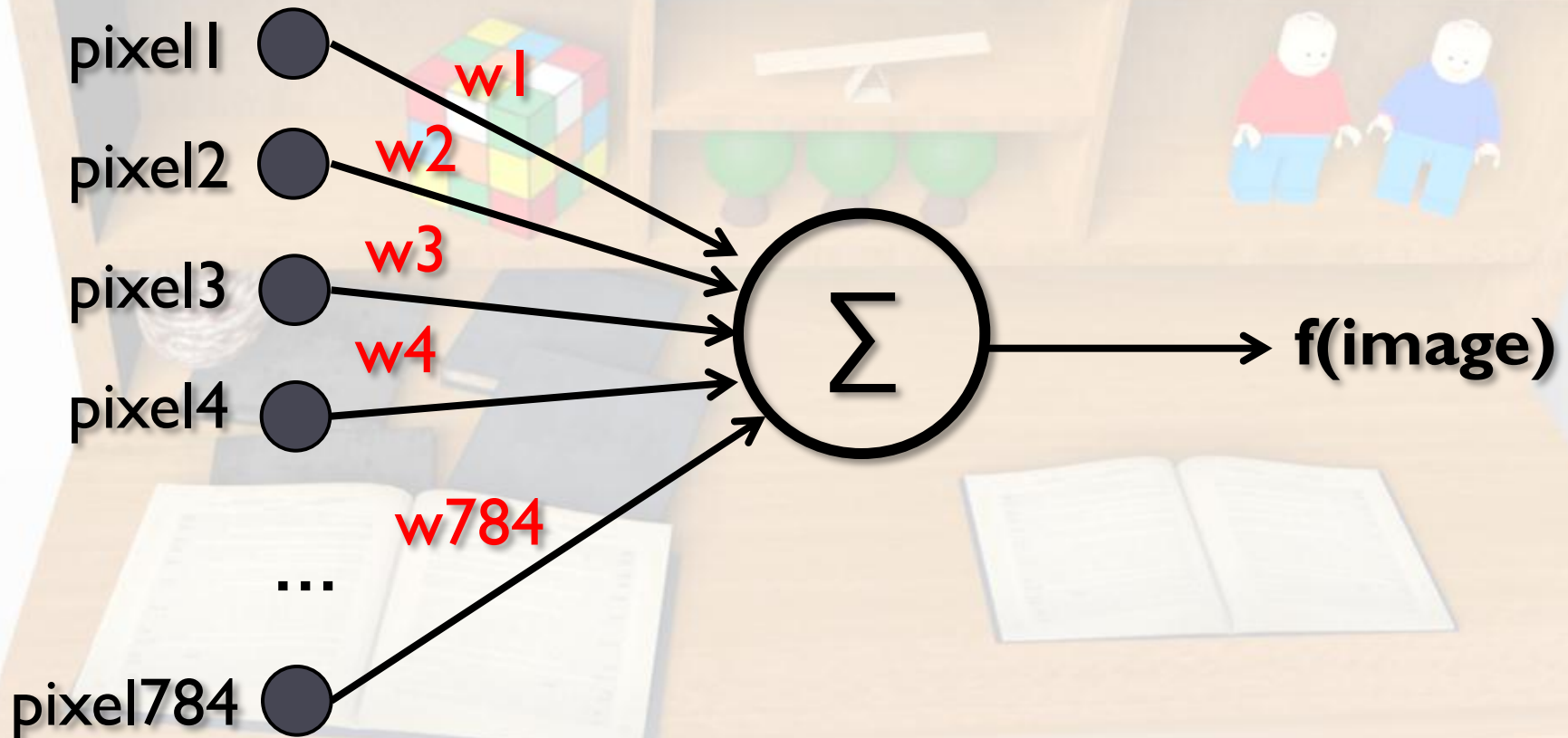
Linear model

$f(\text{image}) =$

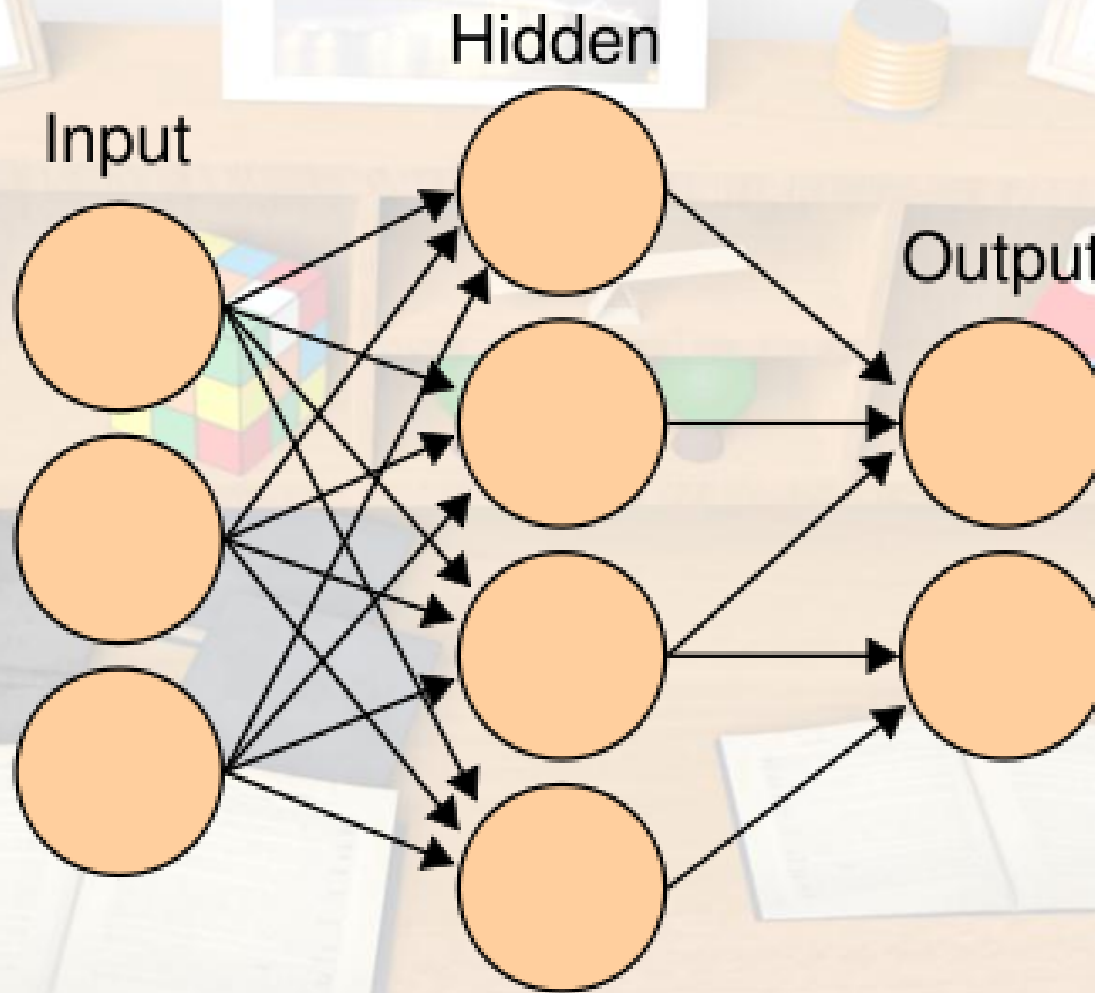
$$\text{pixel1} * w1 + \text{pixel2} * w2 + \dots + \text{pixel784} * w784$$

$f(\text{image}) =$

$$\text{pixel1} * w1 + \text{pixel2} * w2 + \dots + \text{pixel784} * w784$$



Neural network

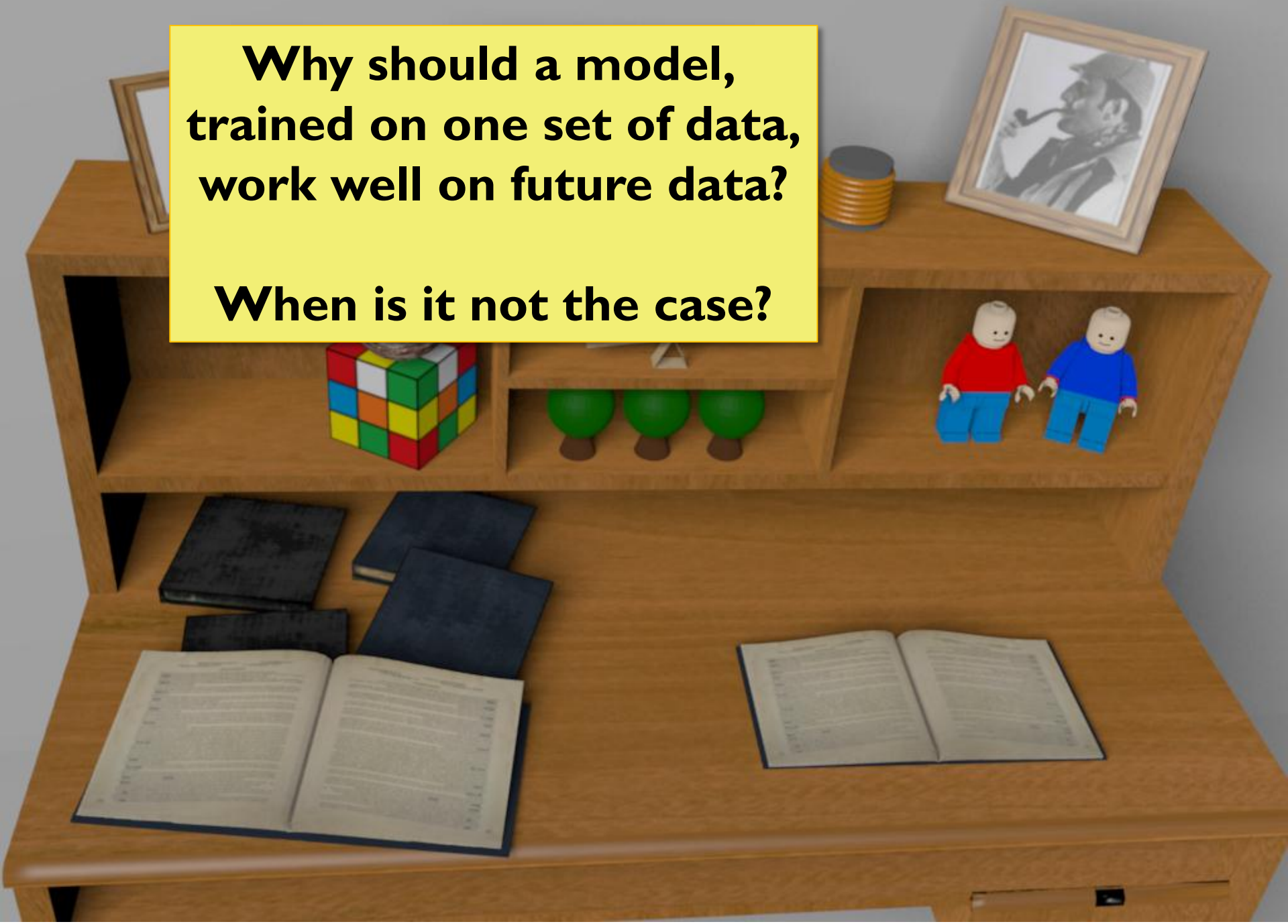


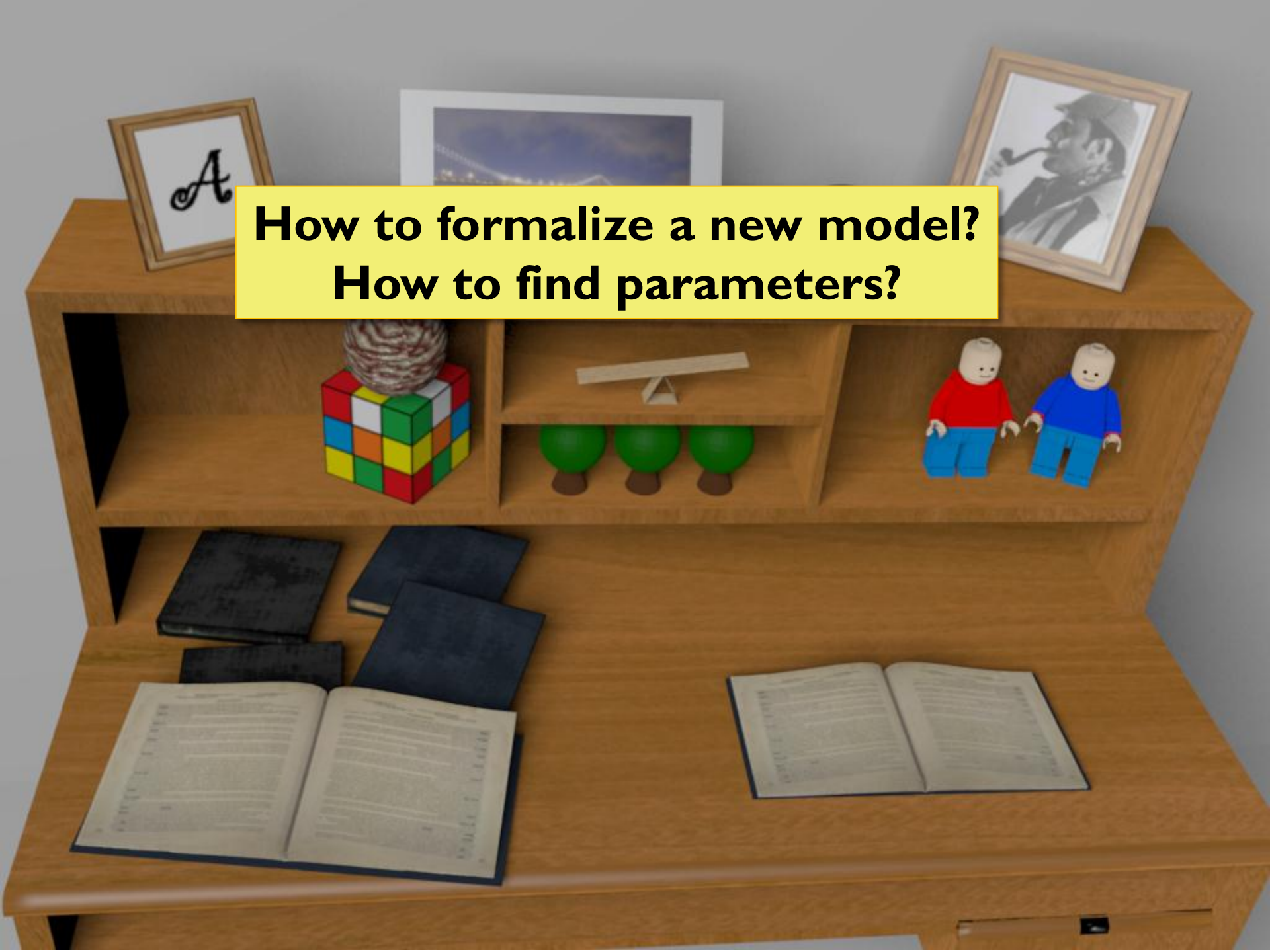


Modeling

**Why should a model,
trained on one set of data,
work well on future data?**

When is it not the case?






**How to formalize a new model?
How to find parameters?**

A wooden desk with various objects. On the top shelf, there are three framed pictures: a letter 'A', a bridge at night, and a man smoking a pipe. Next to the pictures is a stack of yellow and grey discs. Below the pictures, there is a Rubik's cube, a wooden balance scale, and two small green spheres. On the right side of the desk, there are two small figures, one in a red shirt and one in a blue shirt. On the desk surface, there are several books, some closed and some open.

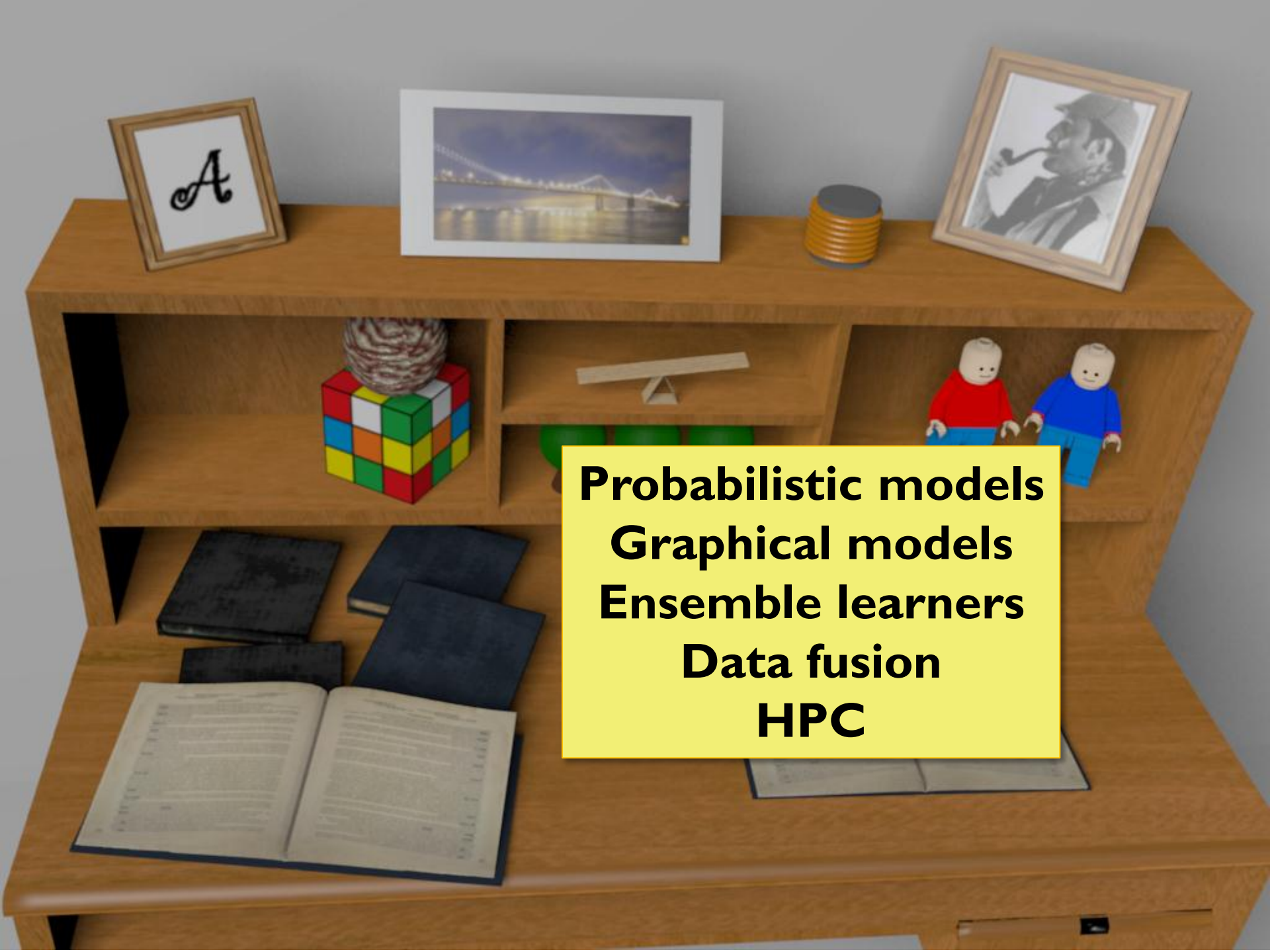
How to create efficient learning algorithms?

A wooden desk with a shelf above it. On the shelf are three framed pictures: a letter 'A', a bridge at night, and a man smoking a pipe. There is also a stack of yellow and orange coasters. On the desk surface, there is a Rubik's cube, three green spherical objects on stands, two small figurines, and several books, some open and some closed.

How to handle structured data?



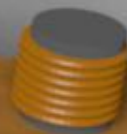
Unsupervised learning
Semi-supervised learning
On-line learning
Active learning
Multi-instance learning
Reinforcement learning



Probabilistic models
Graphical models
Ensemble learners
Data fusion
HPC

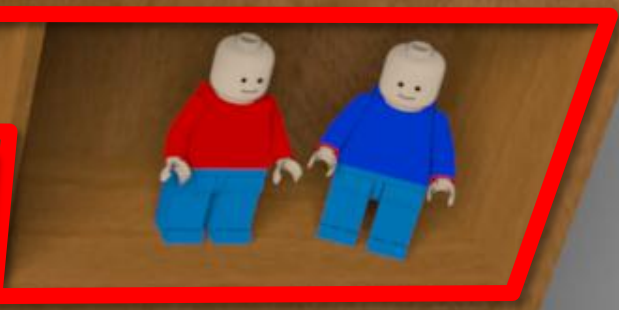
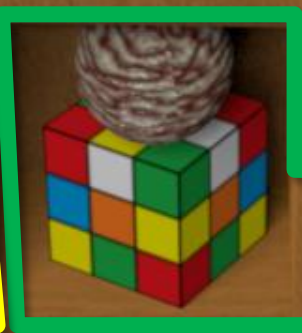


Tools:
R, Weka, RapidMiner,
Orange, scikits-learn,
MLPy, MDP, PyBrain, ...



“Unformalizable” problems

Deduction and Induction



**Theory and
Practice**

Model-based

Instance-based

Quiz

- ▶ The OCR problem is unusual in that it is _____.
- ▶ The two important perspectives on machine learning are _____-based and _____-based.
- ▶ The “soul” of machine learning is the minimization task

$$\operatorname{argmin}_w \text{_____} + \lambda \text{_____}$$

Quiz

- ▶ Two important components of machine learning:

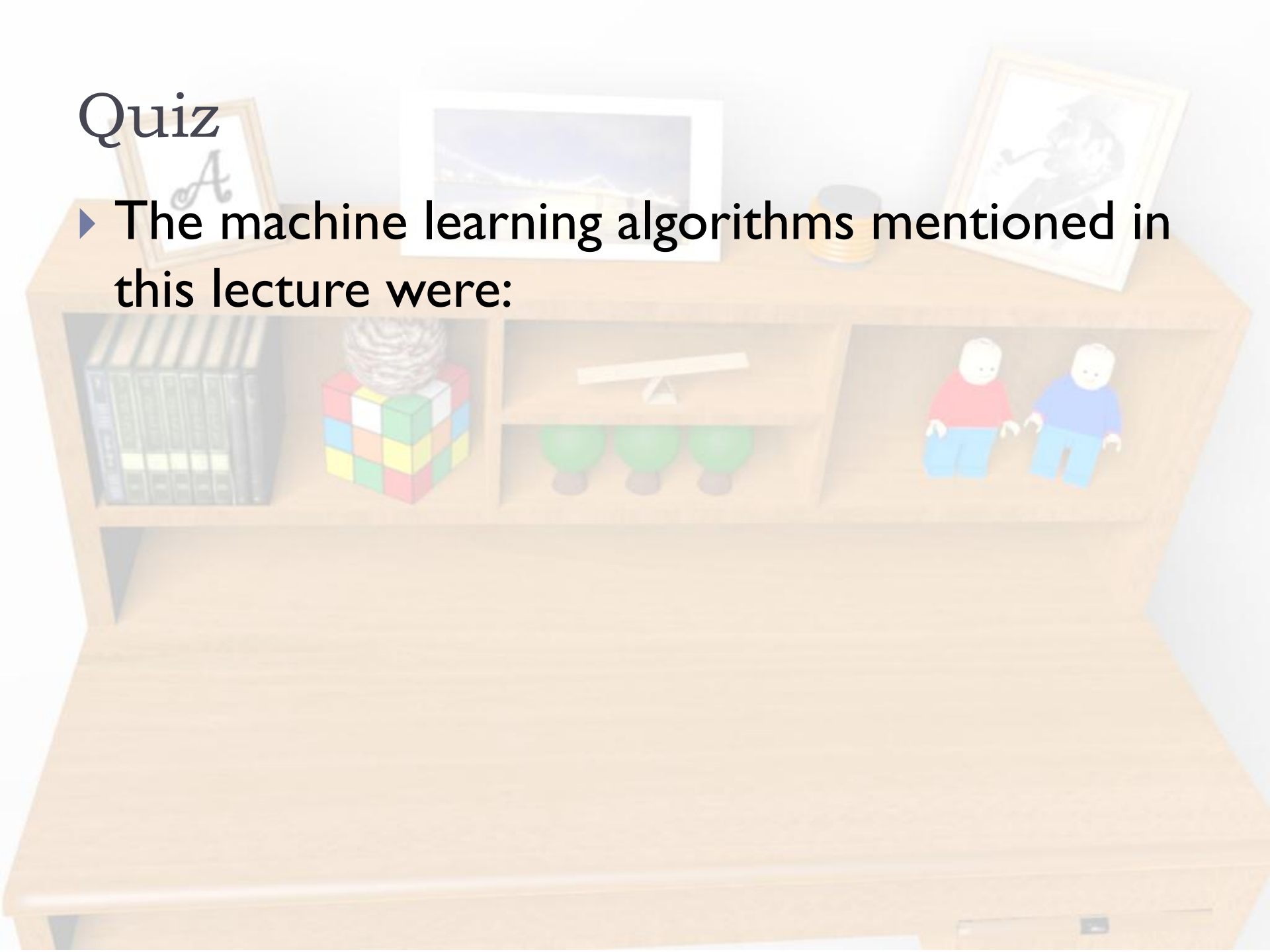


?

?

Quiz

- ▶ The machine learning algorithms mentioned in this lecture were:



Quiz

▶ The machine learning algorithms mentioned in this lecture were:

▶ **K-nearest neighbor classifier**

▶ **SVM**

▶ **Classification trees**

▶ **Linear models**

▶ **Neural networks**



Questions?