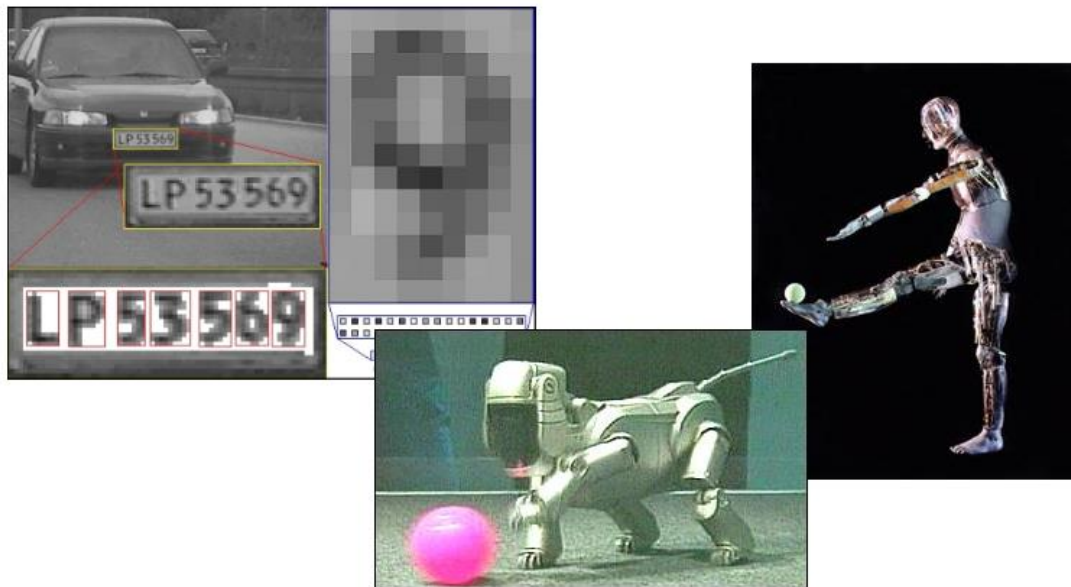




Welcome to Machine Learning

Konstantin Tretyakov

<http://kt.era.ee>

**IFI Summer
School 2014**

STACC

Software Technology and
Applications Competence Center



Data mining,

Data analysis, Statistical analysis,

Pattern discovery, Statistical learning,

Machine learning, Predictive analytics,

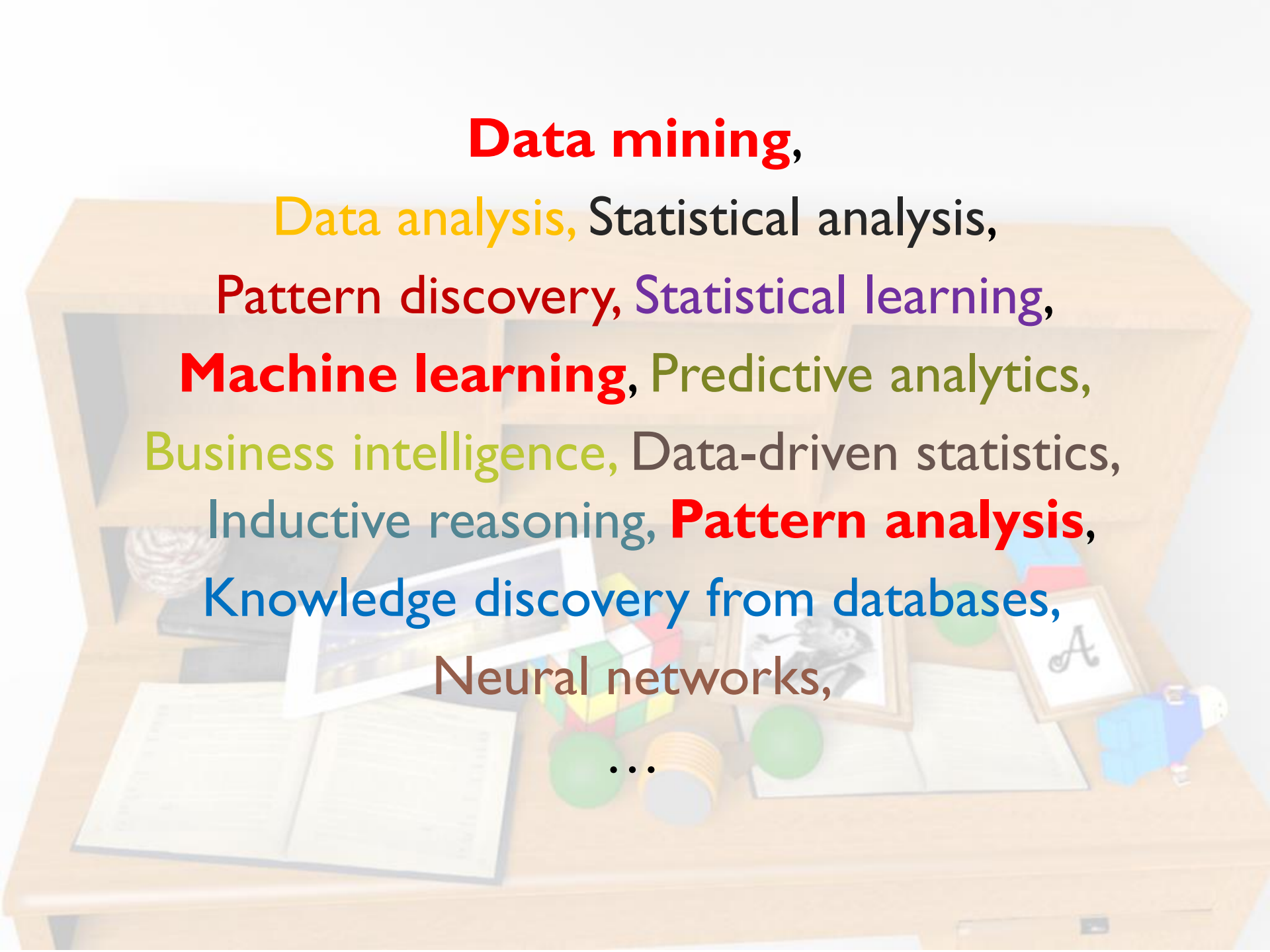
Business intelligence, Data-driven statistics,

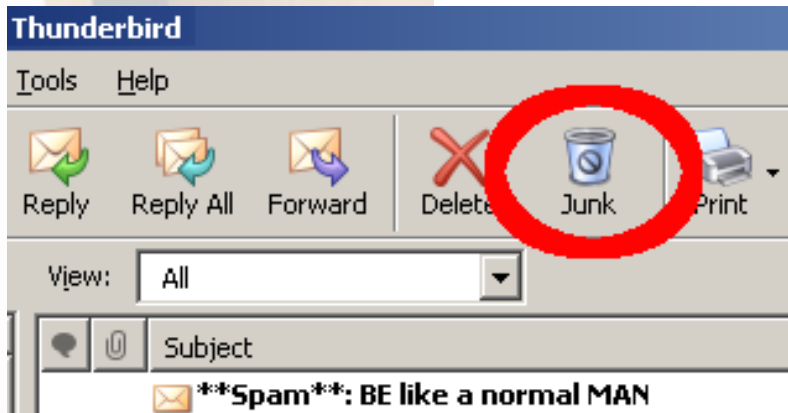
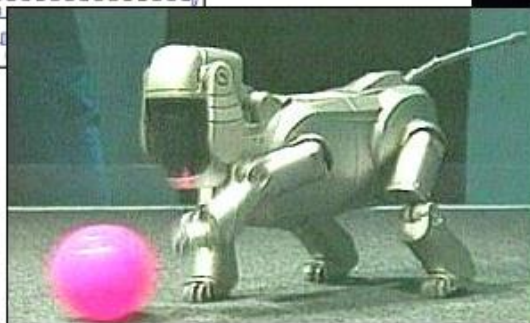
Inductive reasoning, **Pattern analysis,**

Knowledge discovery from databases,

Neural networks,

...



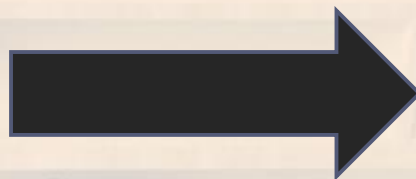
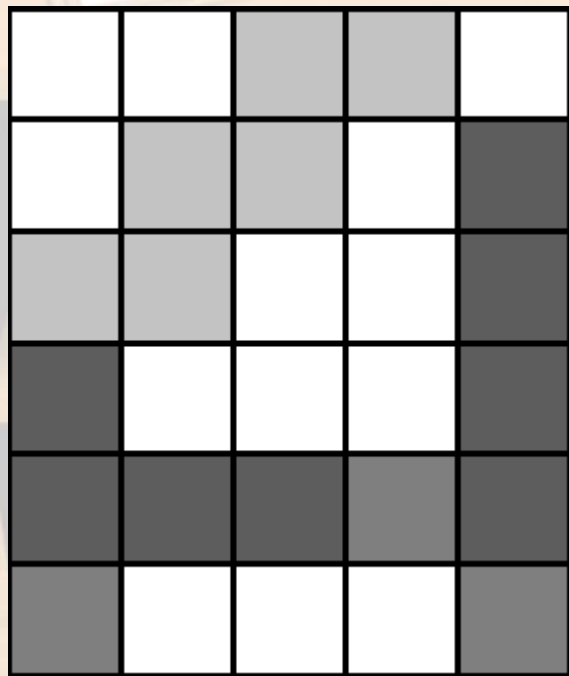


Google translate

NETFLIX

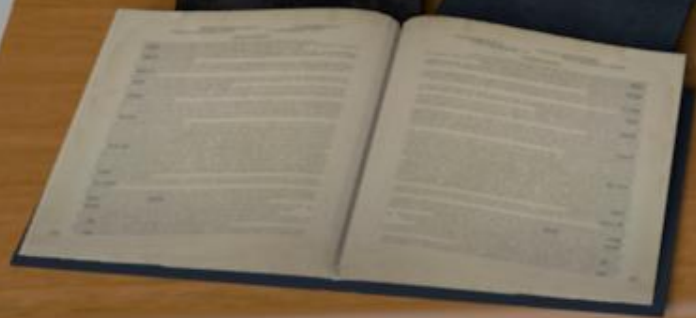


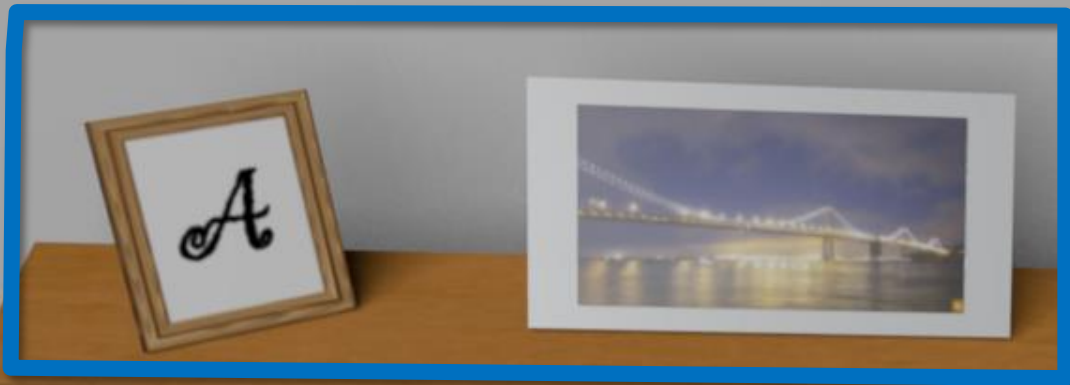




A

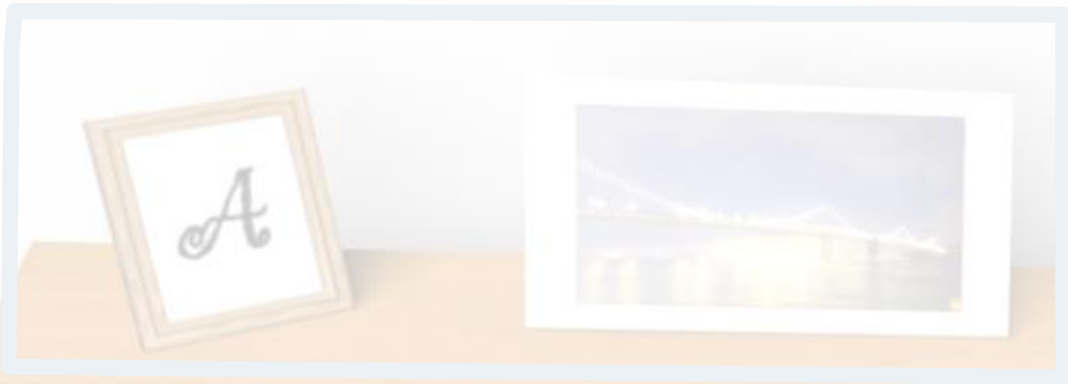




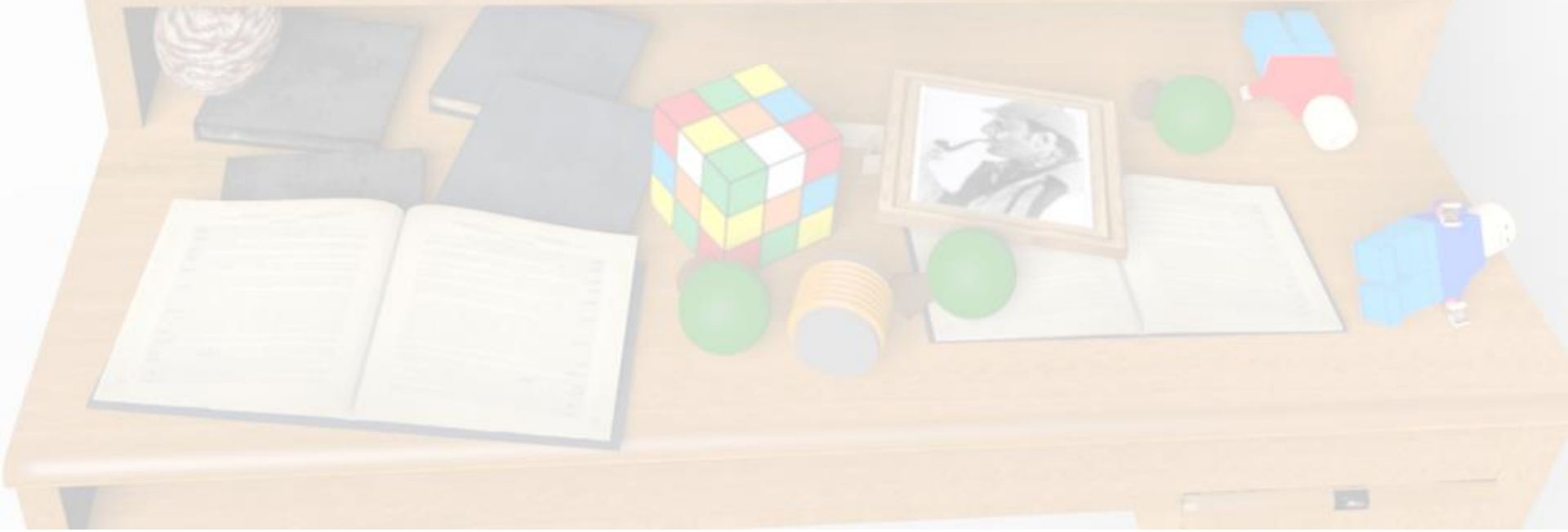


“Unformalizable” problems





“Unformalizable” problems



A

A

B

“Unformalizable” problems

A

B

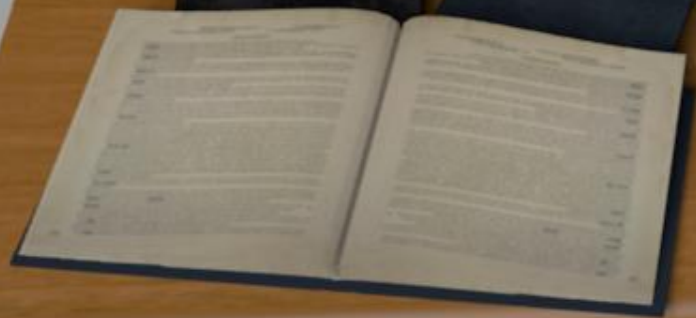
B

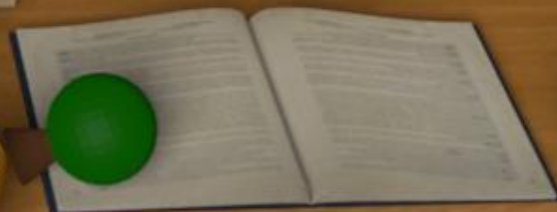
A

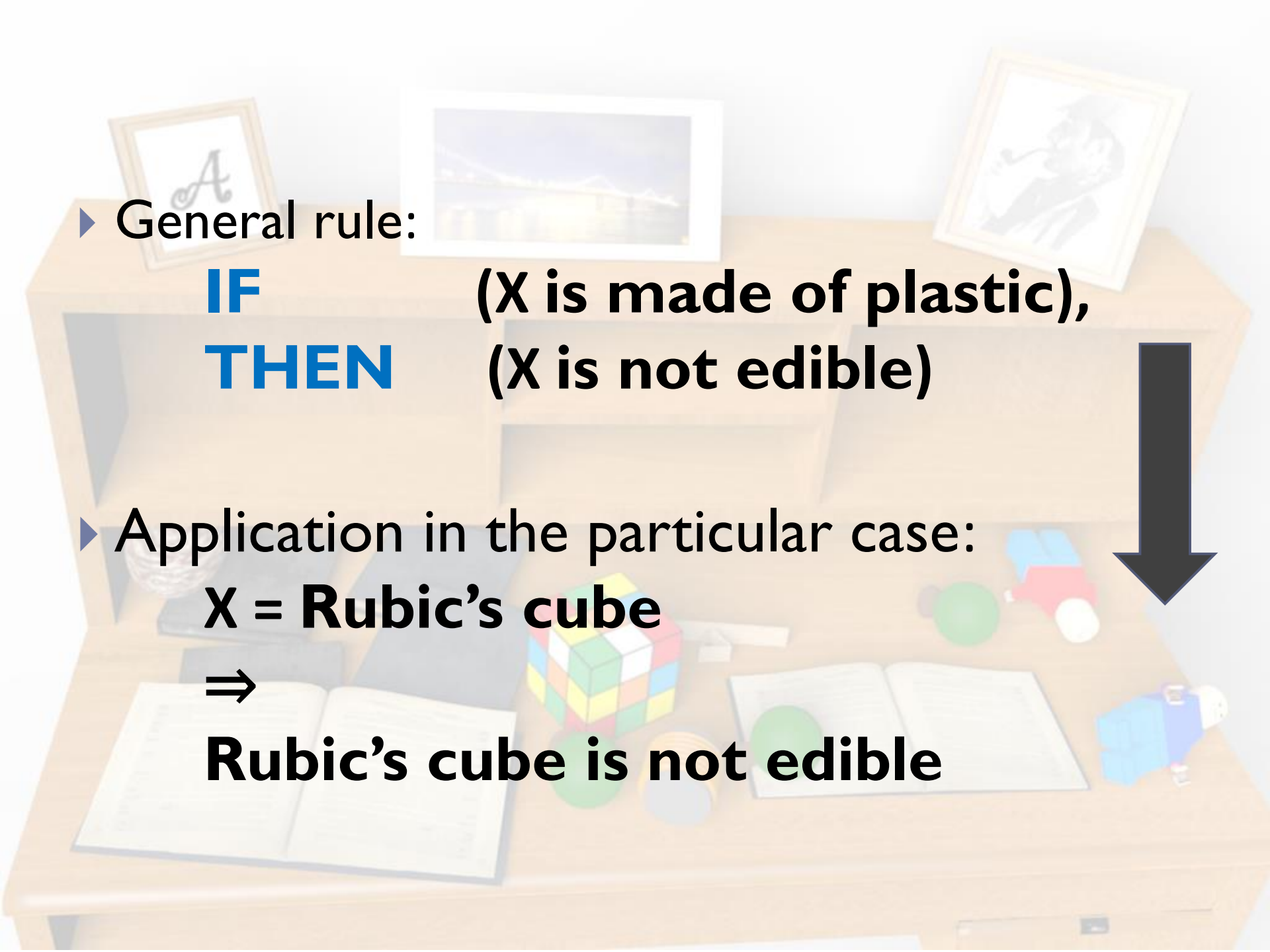
A

A

B





- 
- A background image of a wooden desk. On the desk are several items: an open book on the left, a closed book in the center, a colorful Rubik's cube in the middle, another open book on the right, a green ball, a red and white toy car, and a blue and white toy car. On the wall behind the desk are three framed pictures: a letter 'A' on the left, a landscape with a bridge in the center, and a portrait of a person on the right.
- ▶ General rule:

IF

(X is made of plastic),

THEN

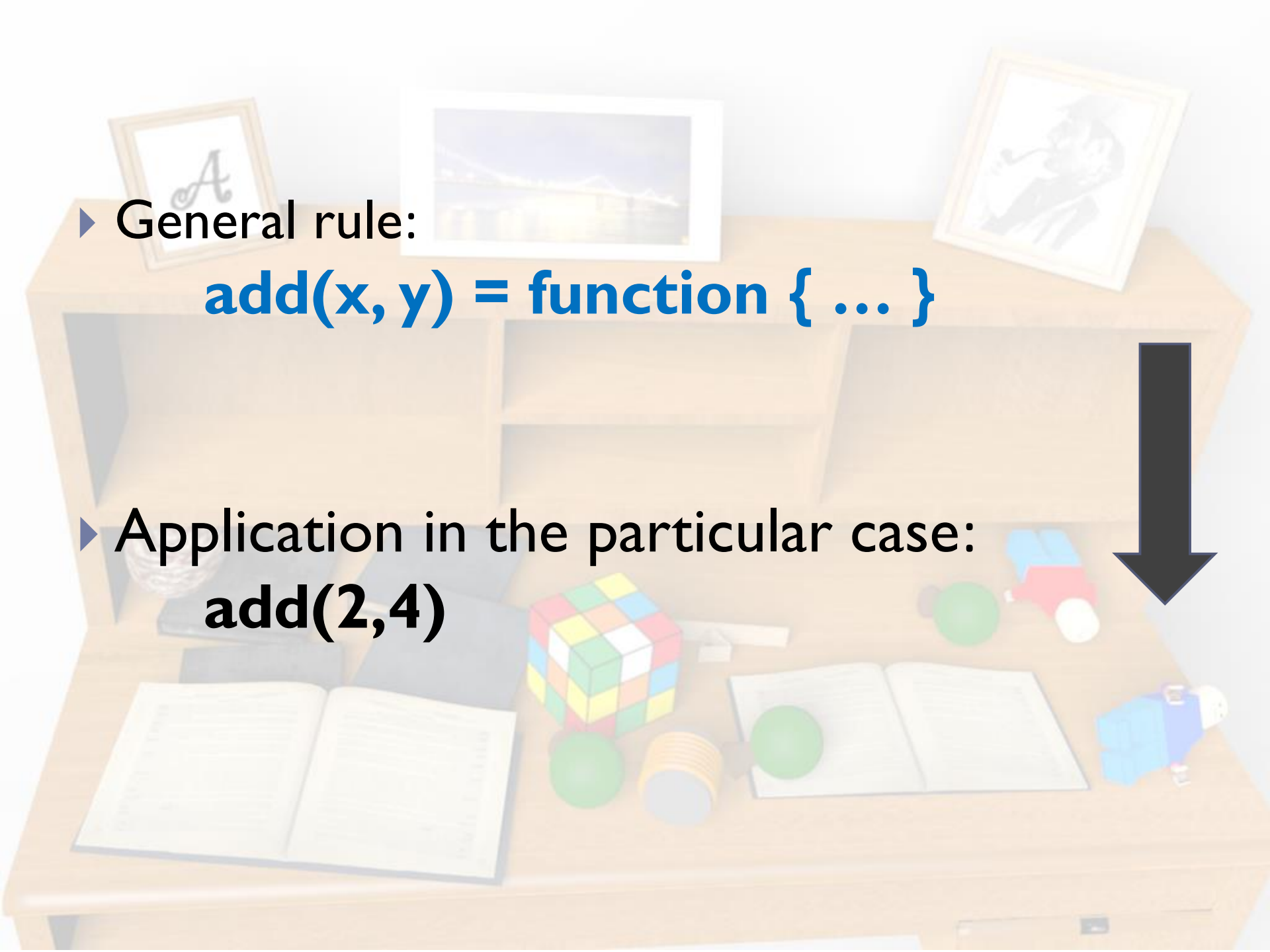
(X is not edible)

- ▶ Application in the particular case:

X = Rubic's cube

⇒

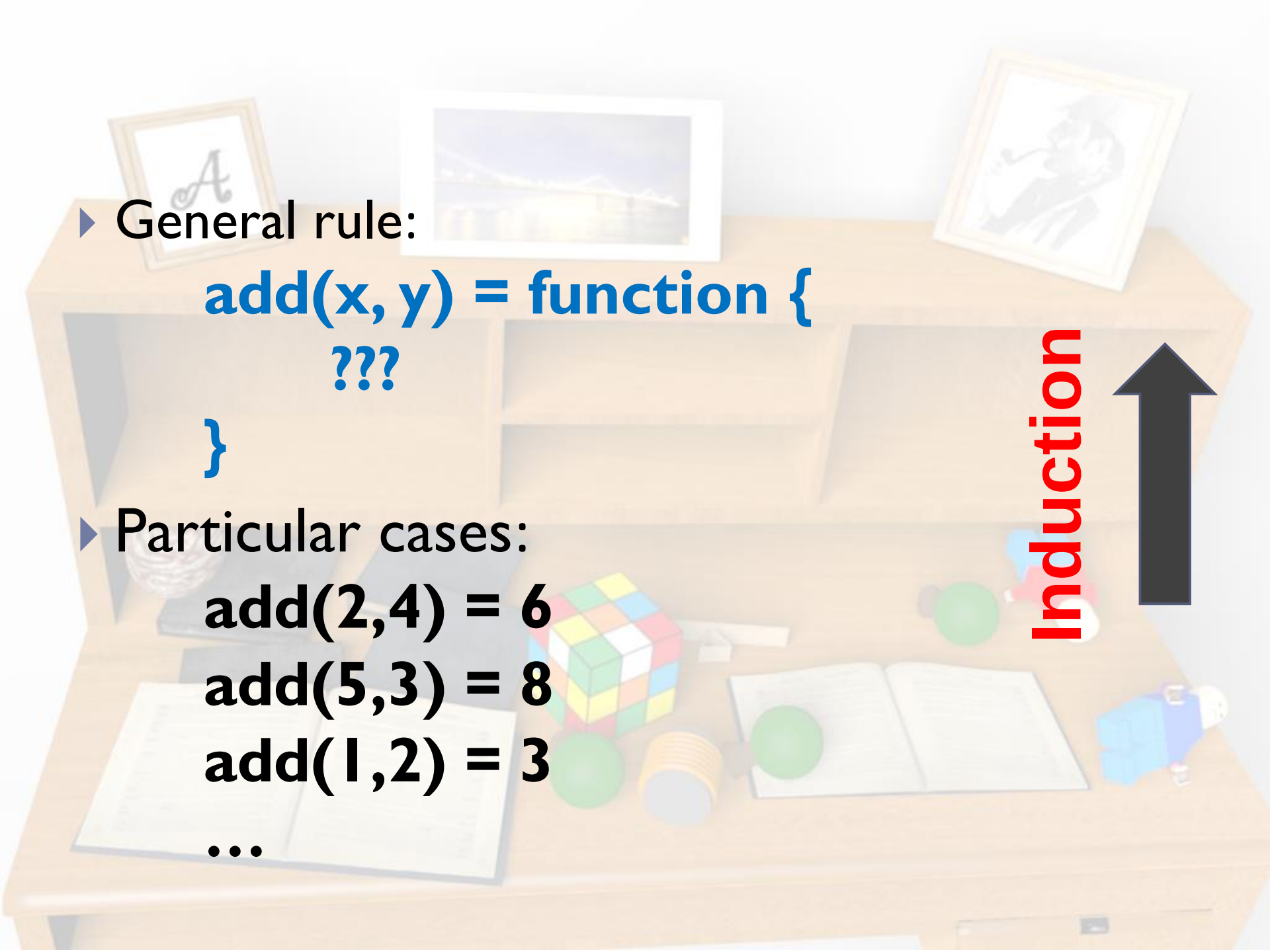
Rubic's cube is not edible

- 
- ▶ 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

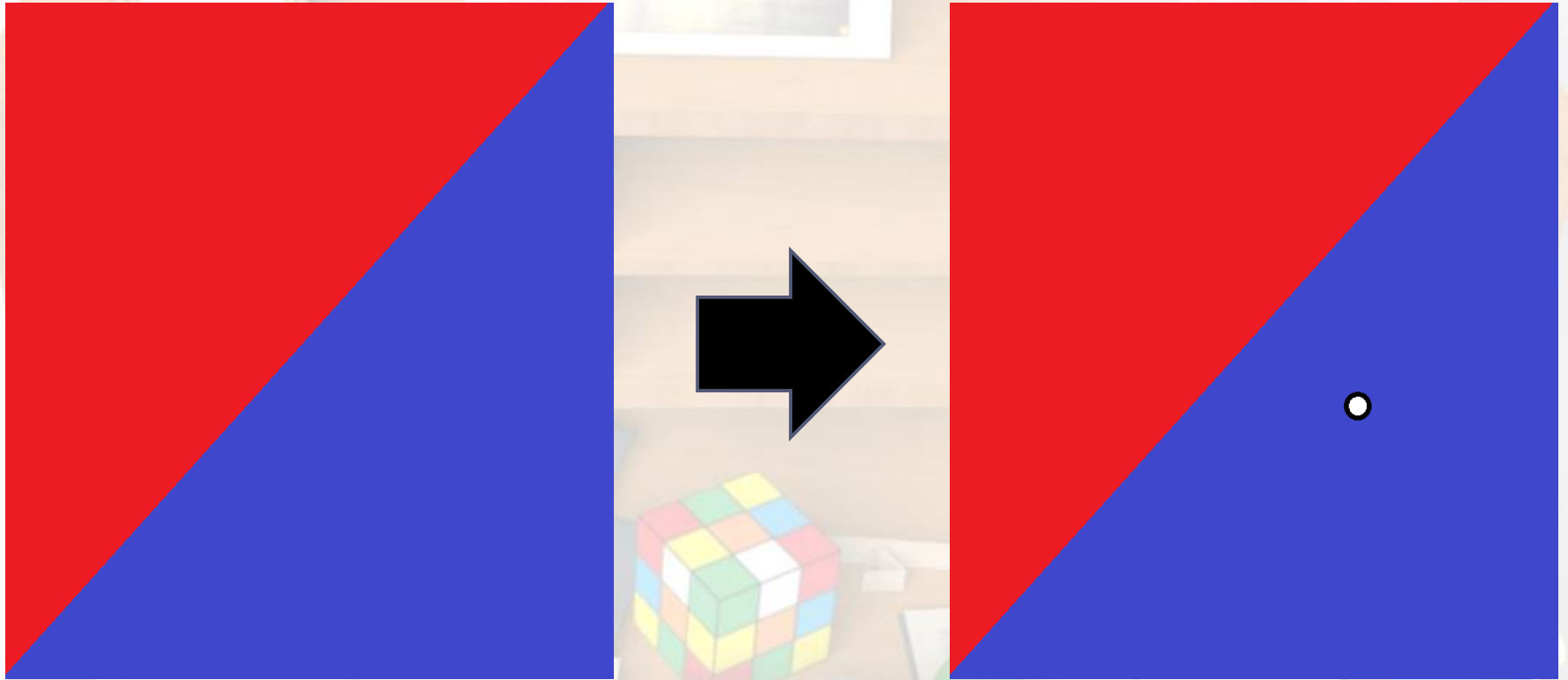
add(1,2) = 3

...

Induction

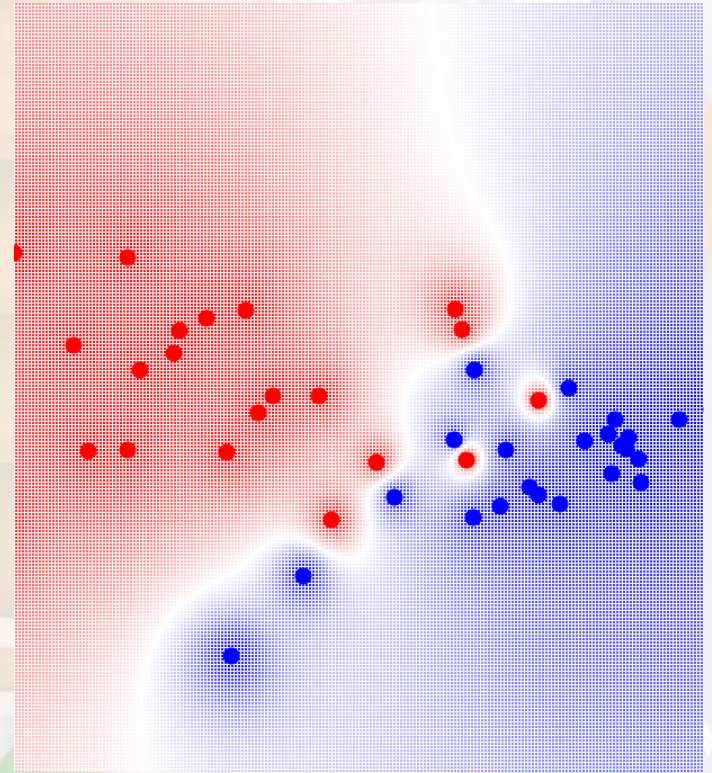
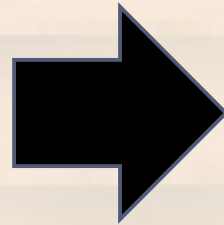
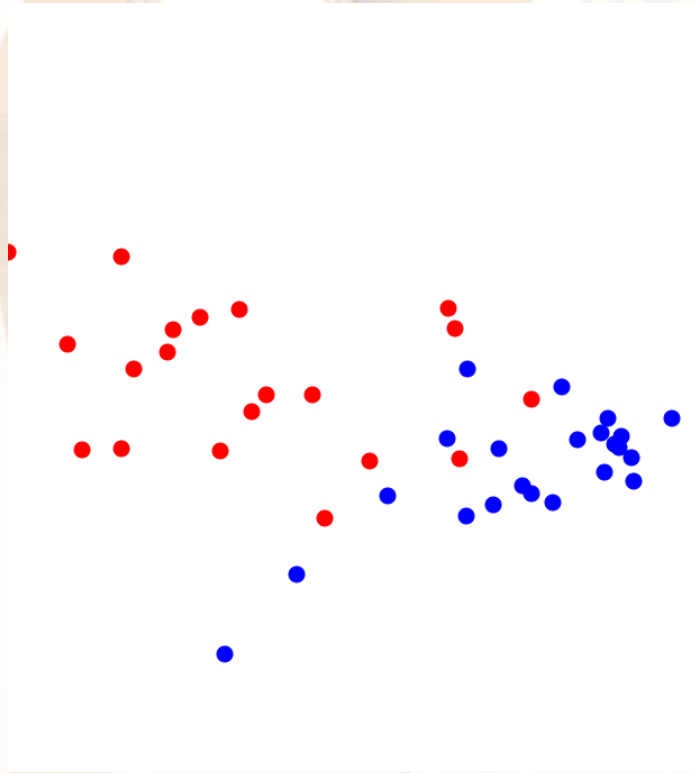


Deduction



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

Induction

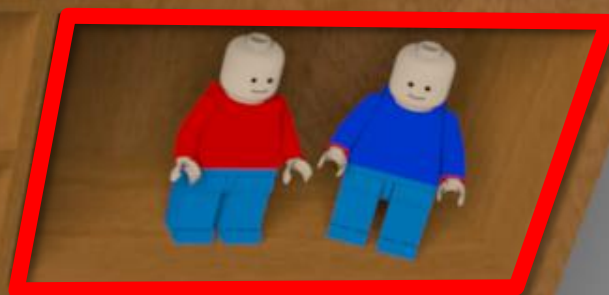
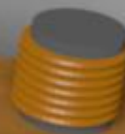


Given particular cases, find a general rule



Deduction and Induction



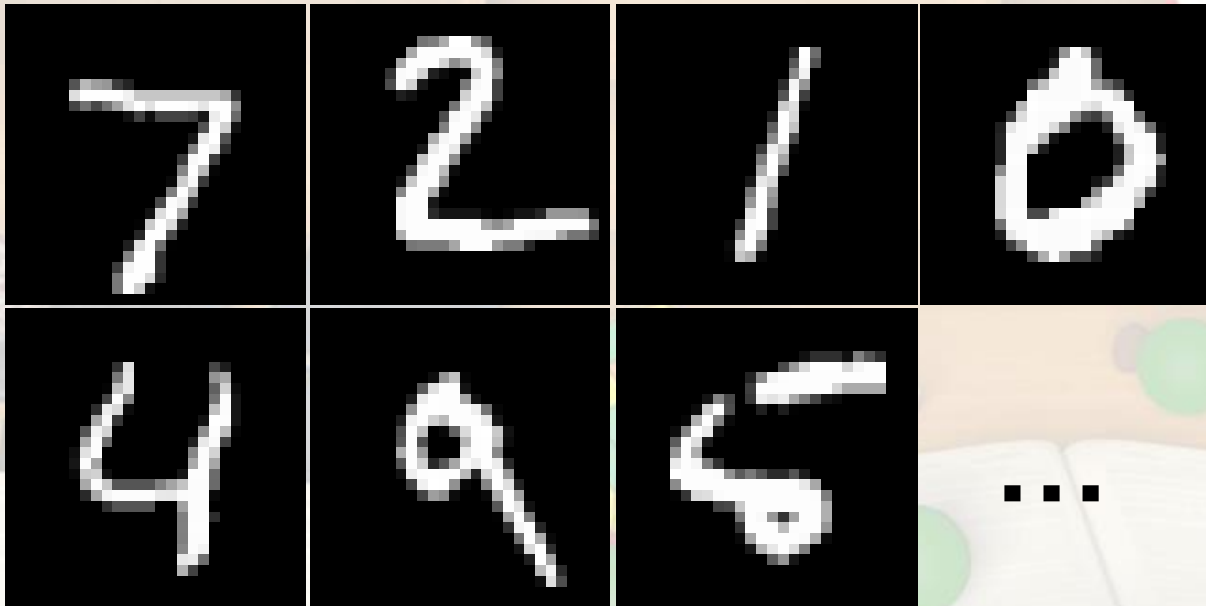


**Classification
by analogy**



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]
```

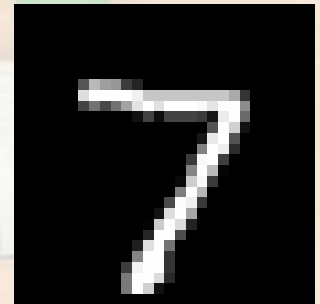
MNIST dataset

```
> training_set[0]
```

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

```
> training_labels[0]
```

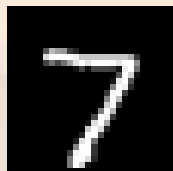
'7'



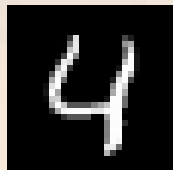
Nearest neighbor method

Training set

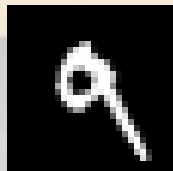
7



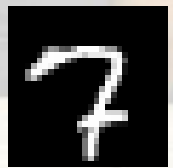
4



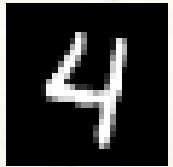
9



7



4

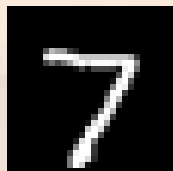


?

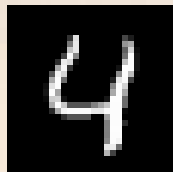
Nearest neighbor method

Training set

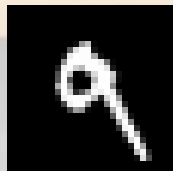
7



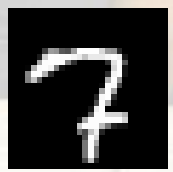
4



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]
```

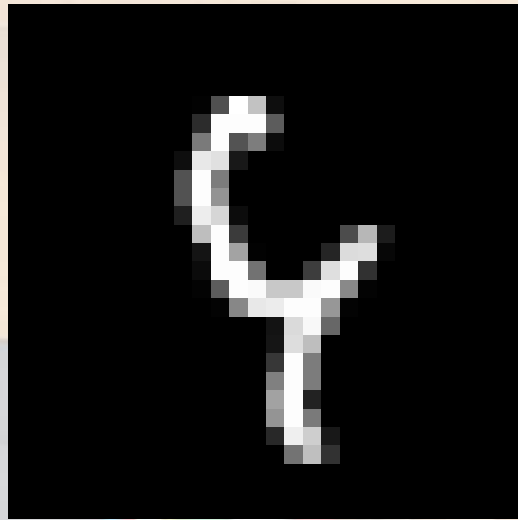
```
test_labels = labels[1000:2000]
```

```
predicted_class = [classify(p) for p in test_set]
```

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

=> 843/1000

9 or 4?

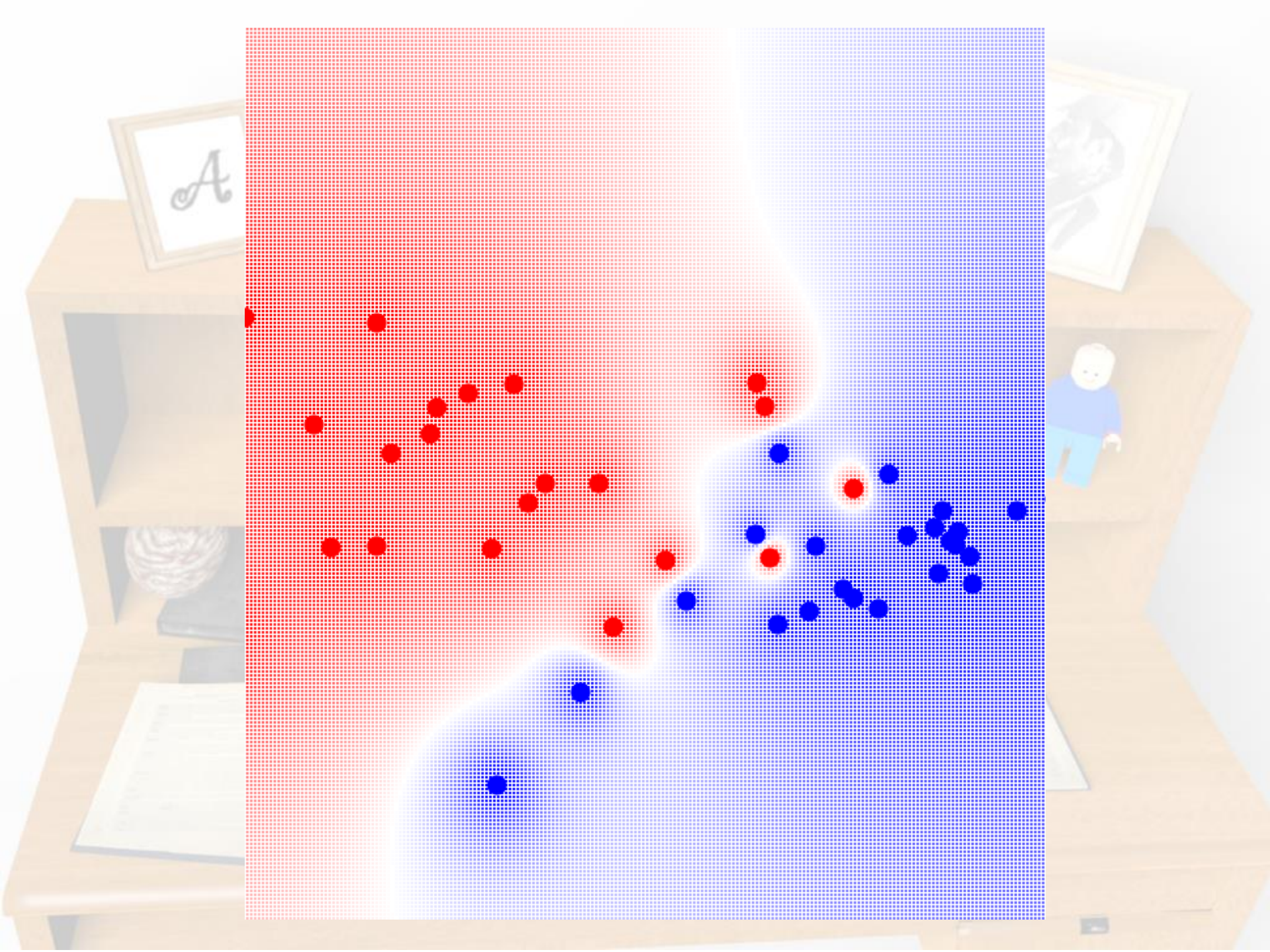


Scikit-learn

<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)
```

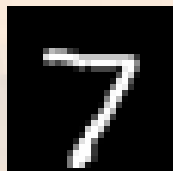
=> 855/1000



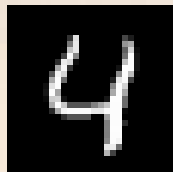
Nearest neighbor method

Training set

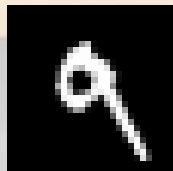
7



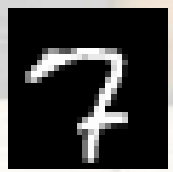
4



9



7



4



4

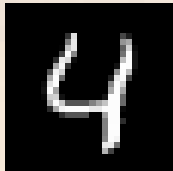
Some samples may be thrown away

Training set

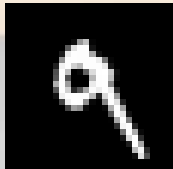
~~7~~



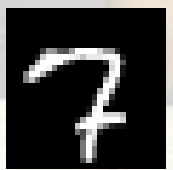
4



9



7



~~4~~



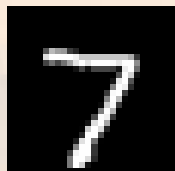
4

...or we can add “weights”

Training set

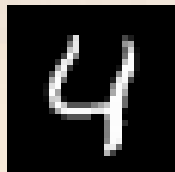
0.0

7



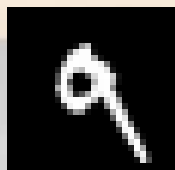
1.0

4



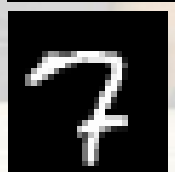
1.0

9



1.0

7



0.0

4



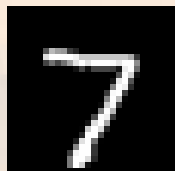
4

...or we can add “weights”

Training set

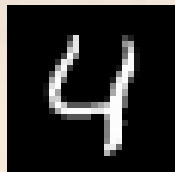
0.0

7



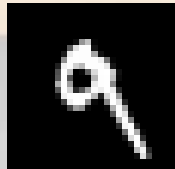
0.9

4



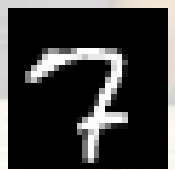
1.1

9



2.0

7



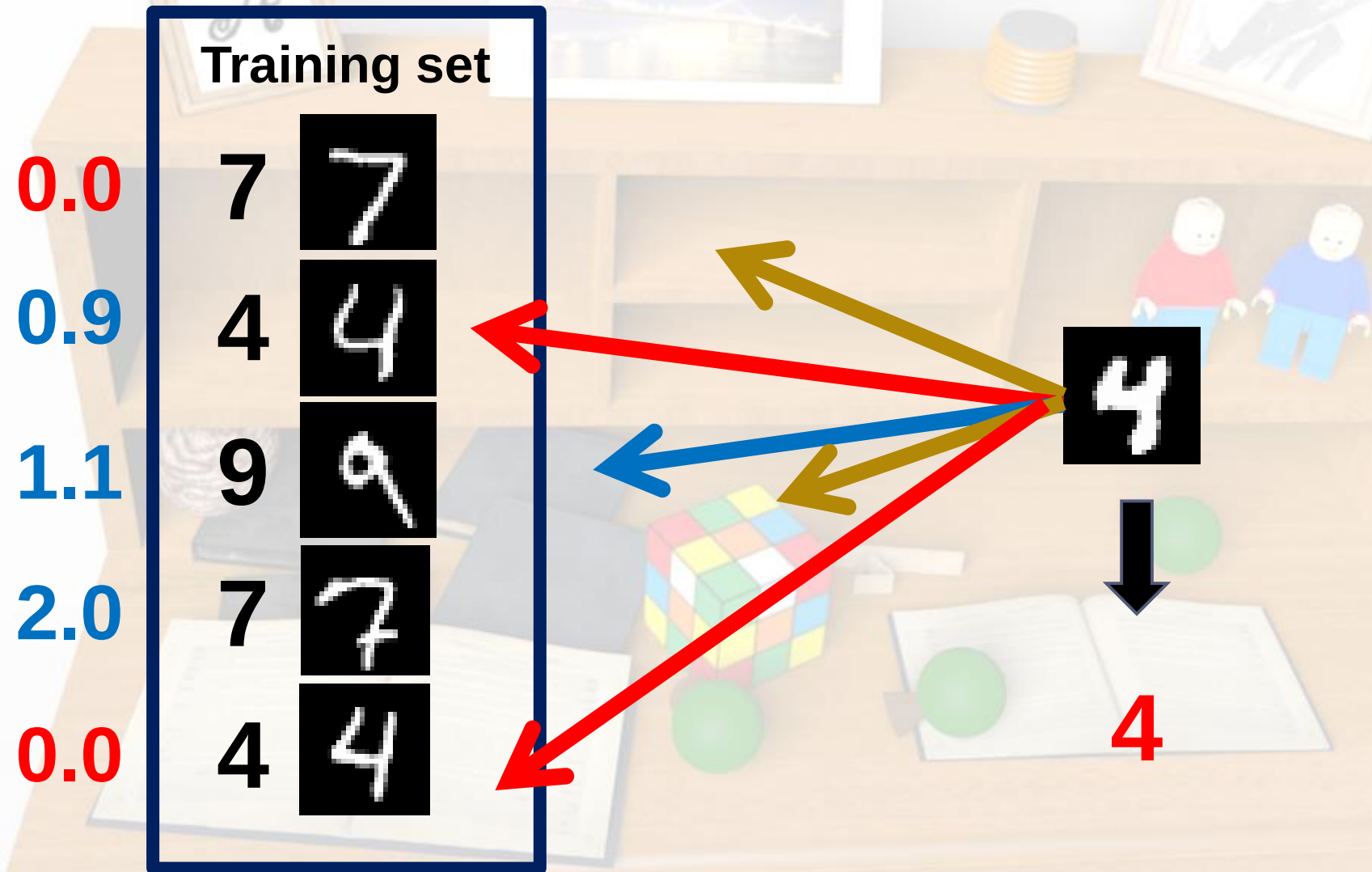
0.0

4



4

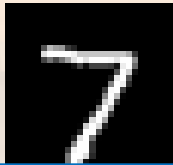
.. or we can SUM instead of MAX



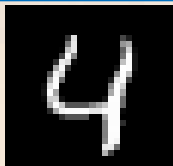
.. or we can SUM instead of MAX

Training set

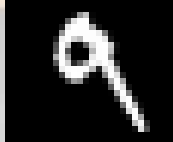
7



4



9



7



4



evidence(img, 4) =

0.9 * similarity(img, tr[1])

+

0.0 * similarity(img, tr[4])

= 34.0

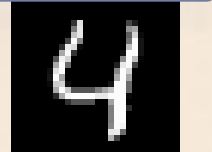
.. or we can SUM instead of MAX

Training set

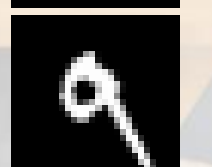
7



4



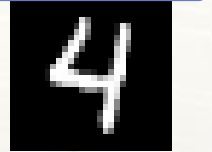
9



7



4



evidence(img, 7) =

0.0 * similarity(img, tr[0])

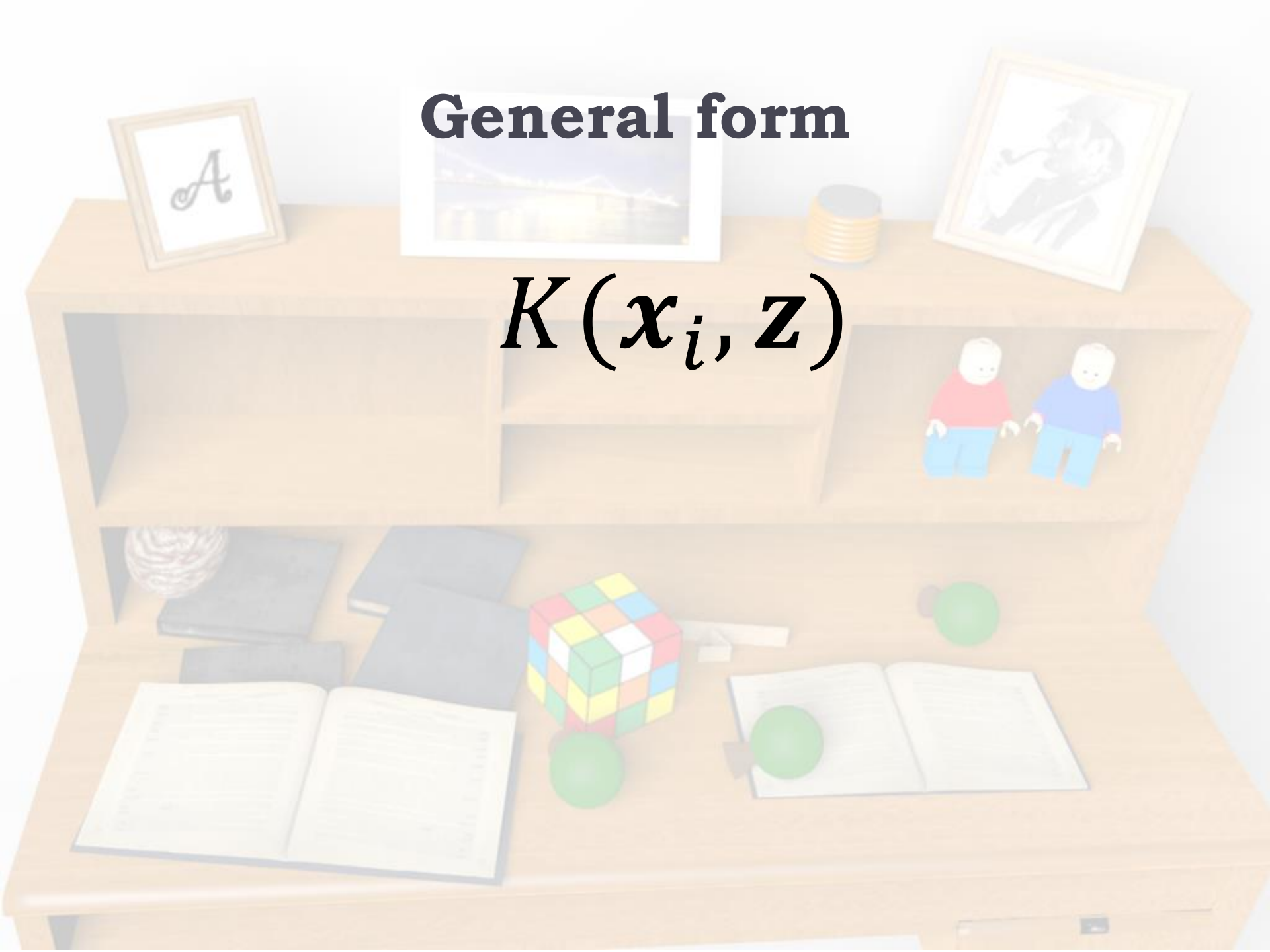
+

2.0 * similarity(img, tr[3])

= 20.0

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?

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

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**.

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

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**.

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

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

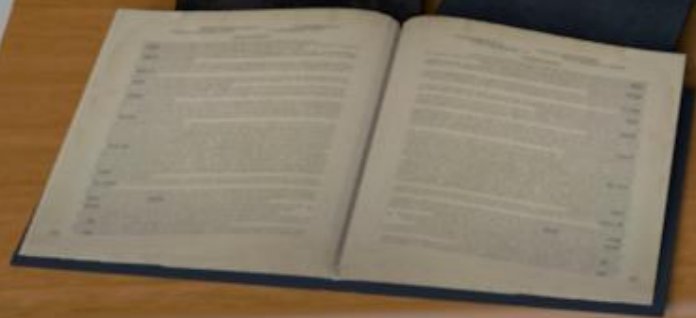
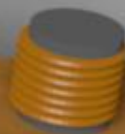
```
from sklearn.svm import SVC
```

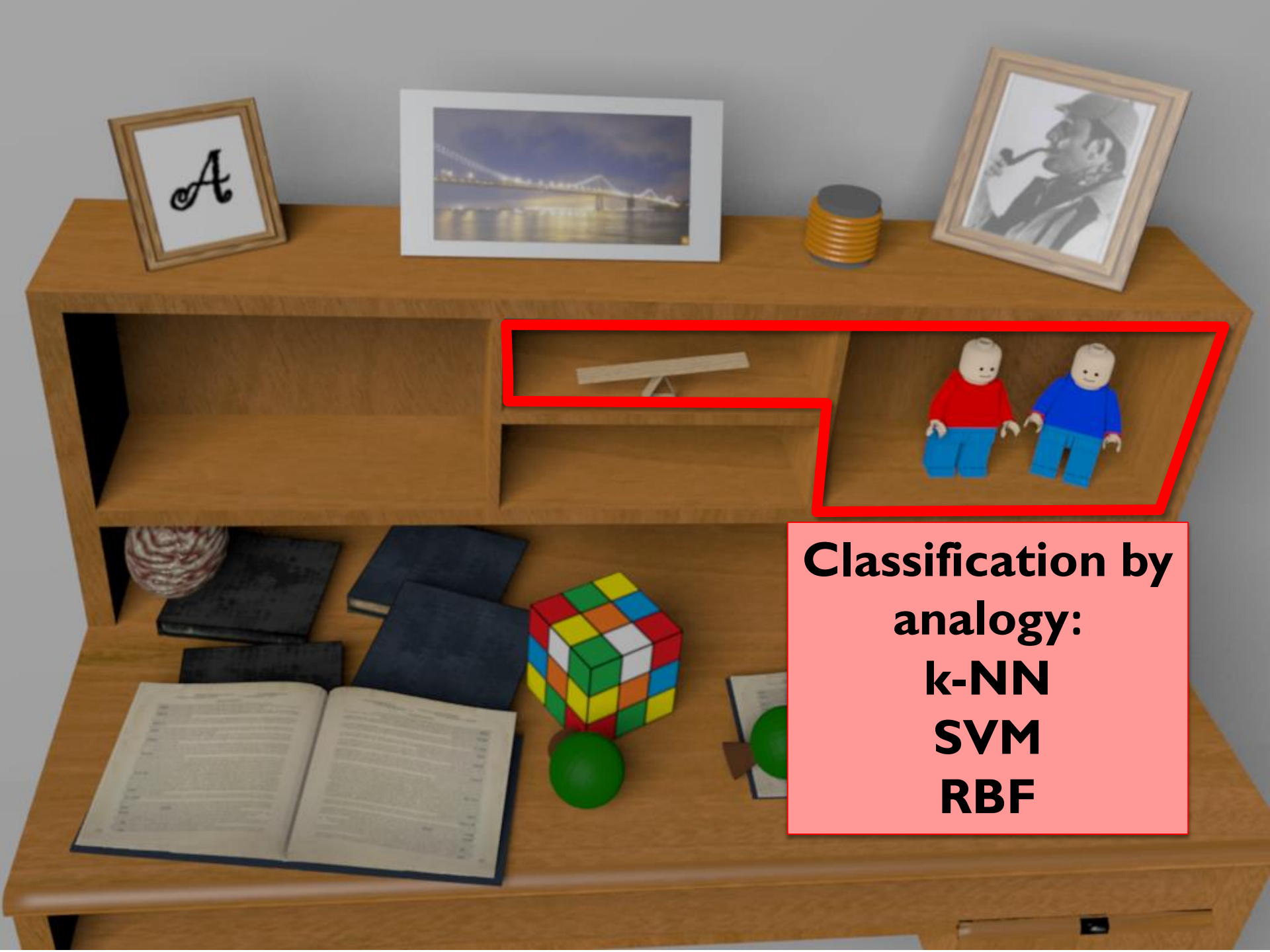
```
clf = SVC(kernel='linear')
```

```
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]
```

Search for the nearest neighbor

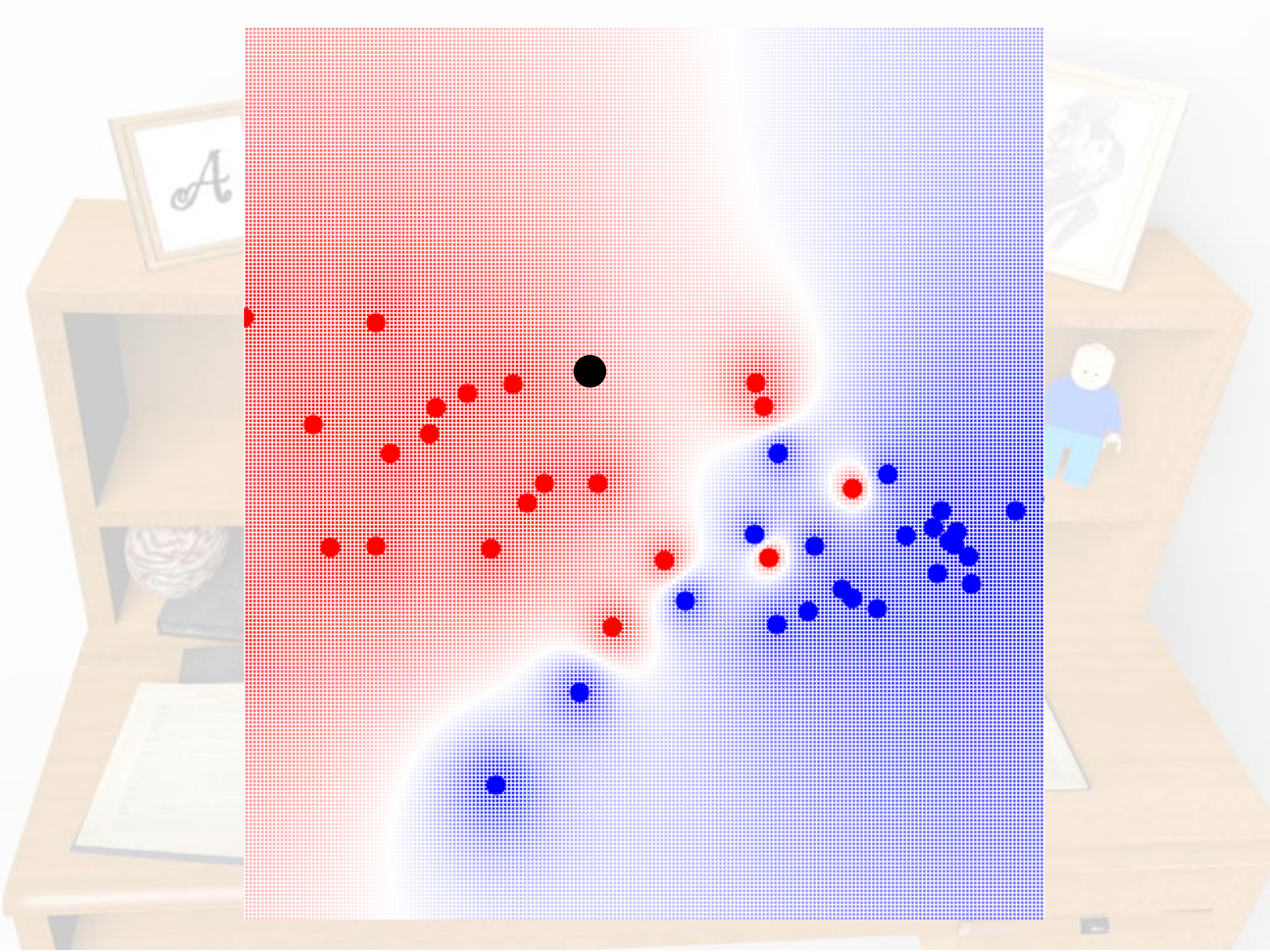
```
def classify(img):  
    similarities =  
        [similarity(img, p) for p in training_set]  
    i = similarities.index(max(similarities))  
    return training_labels[i]
```

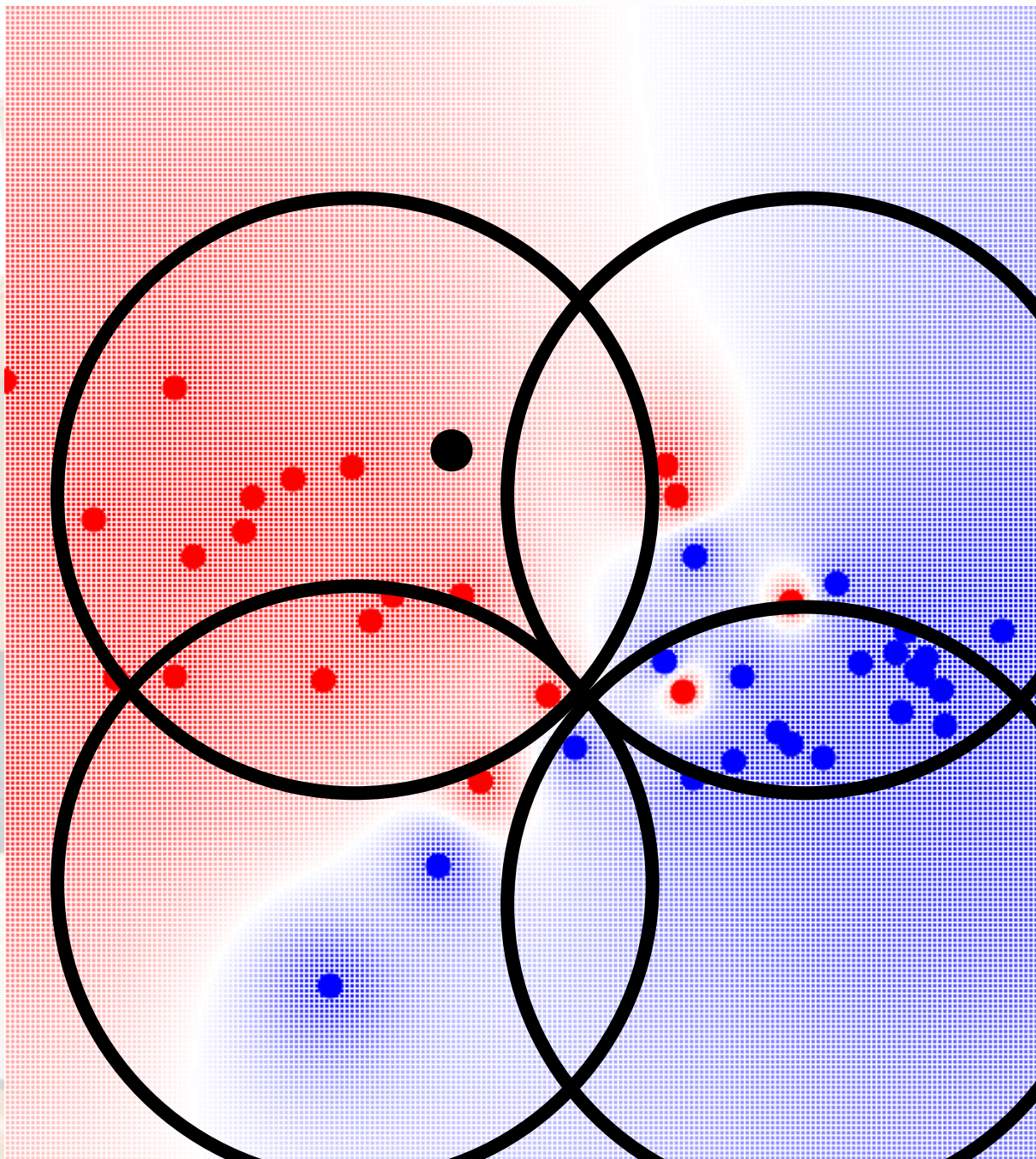
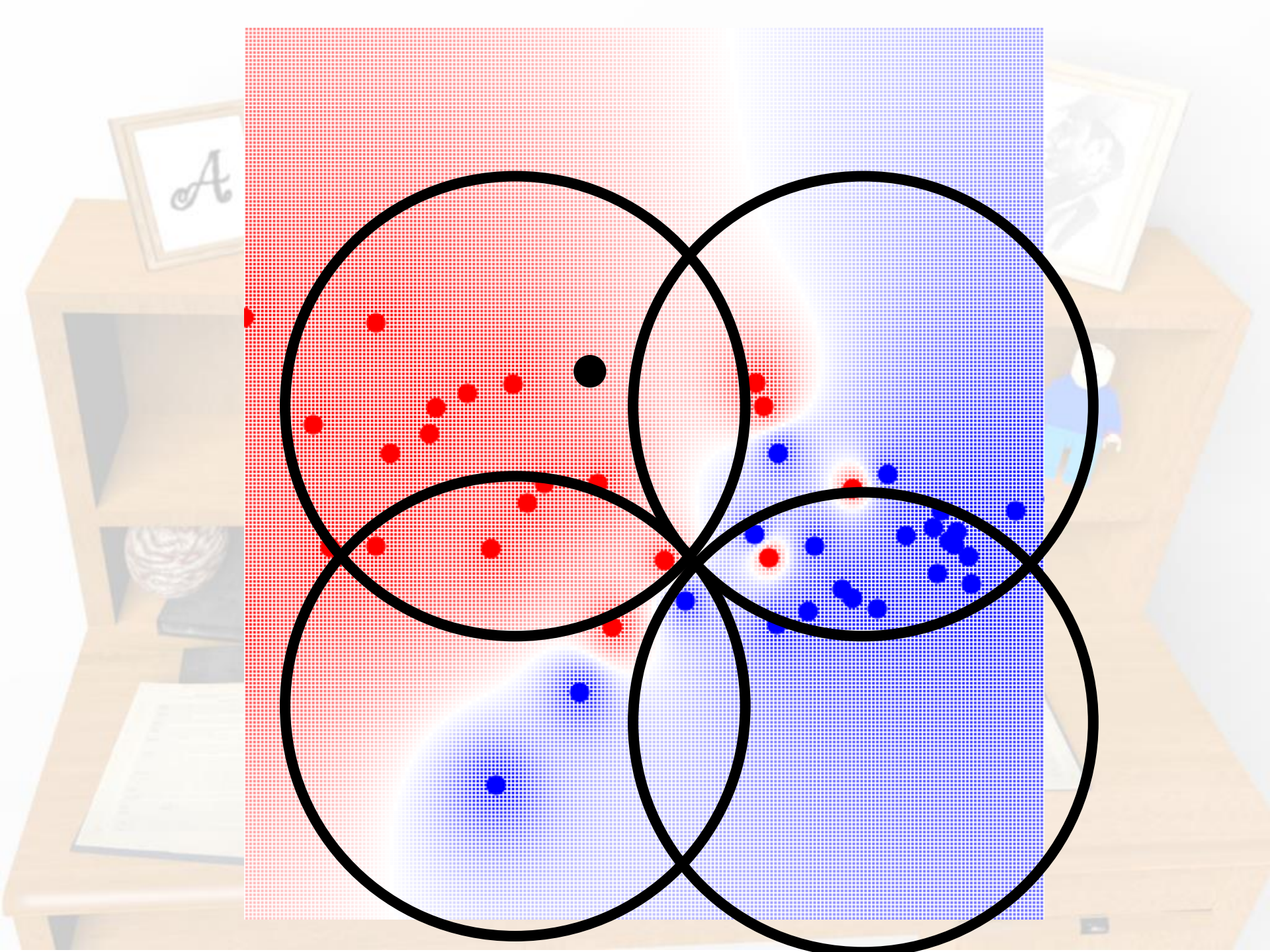
Inefficient

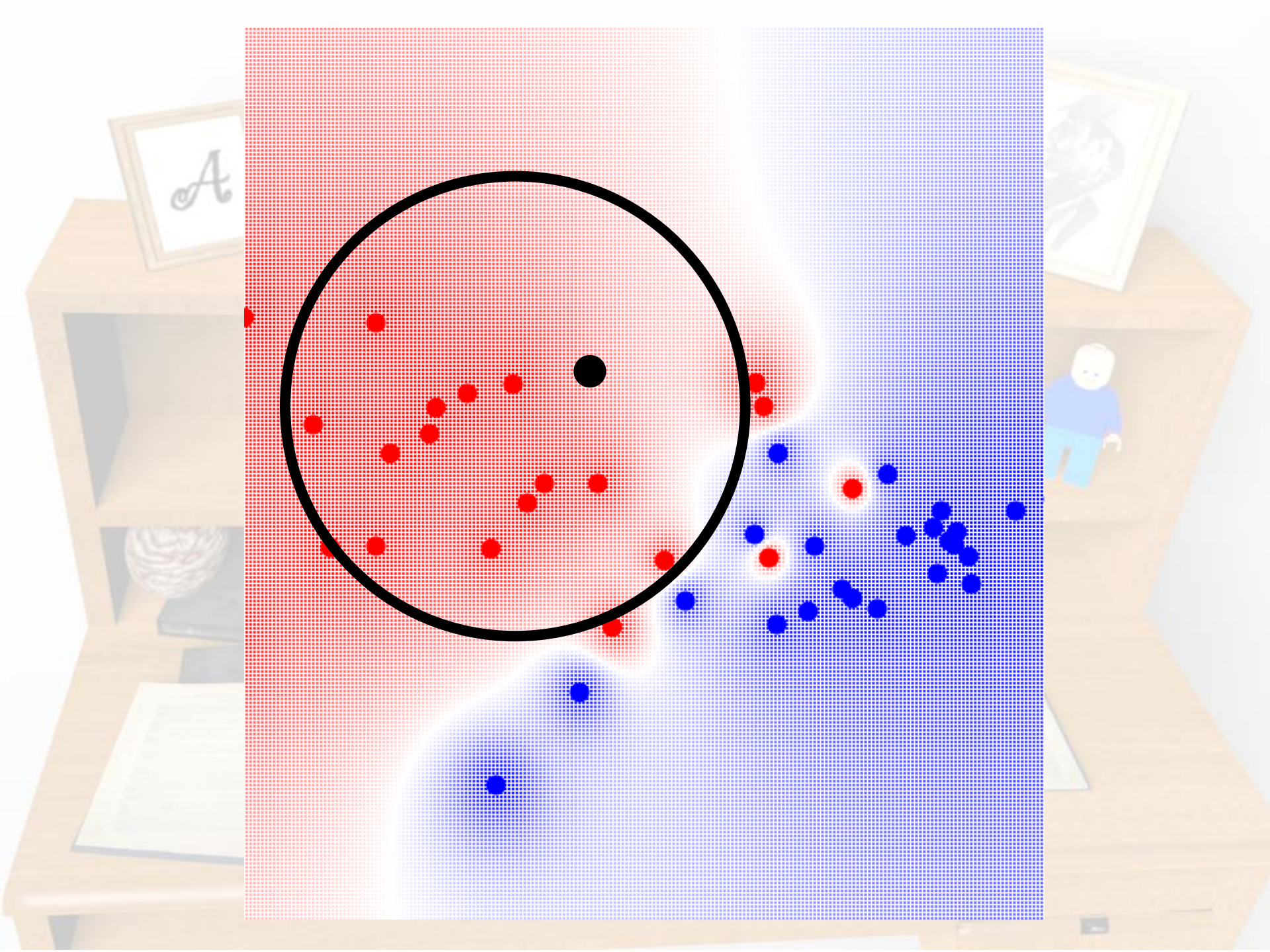
Search for the nearest neighbor

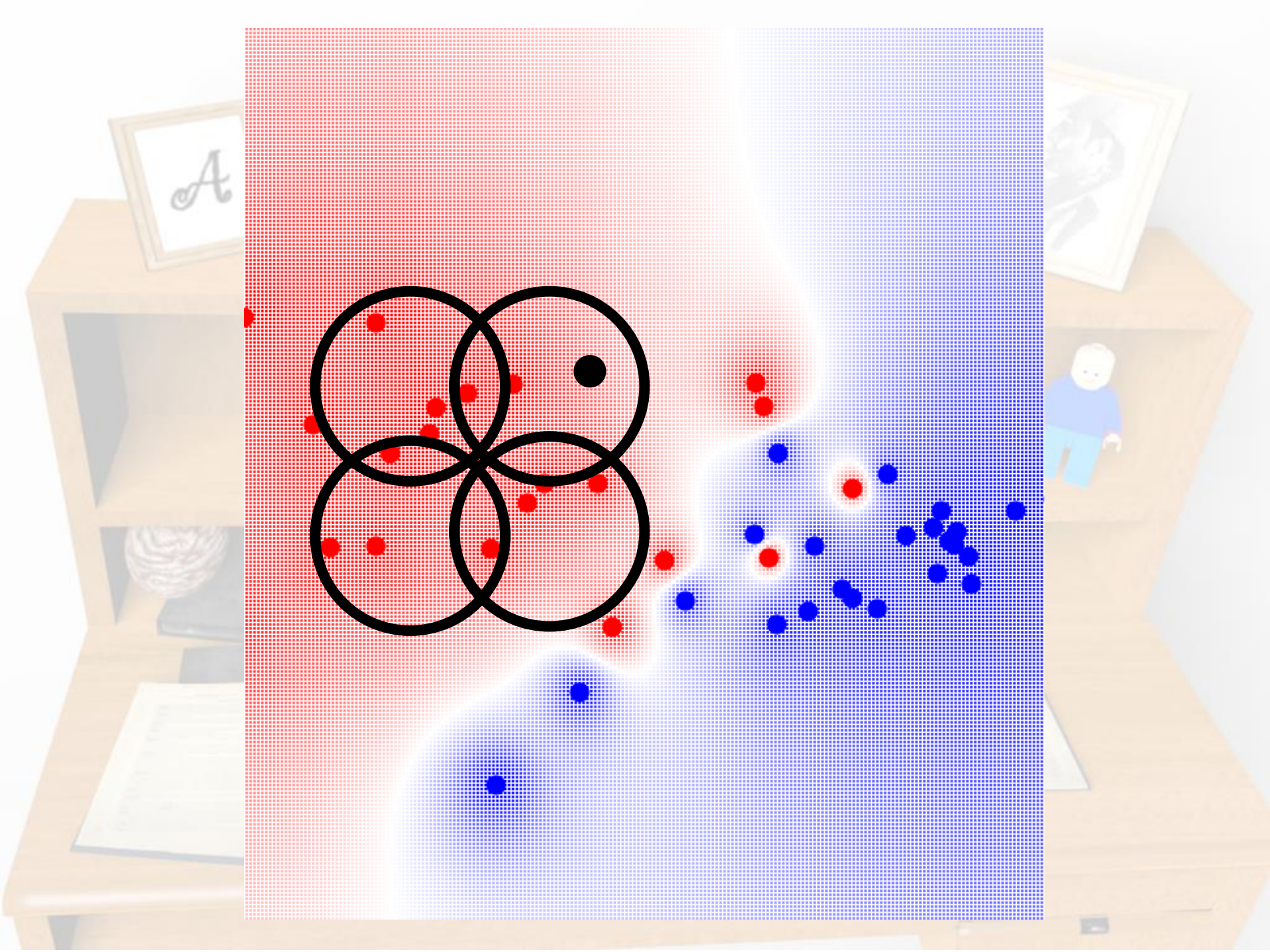
```
def classify(img):  
    nearest_neighbour =  
        training_set.find_nearest_neighbour(img)  
  
    return nearest_neighbour.label
```

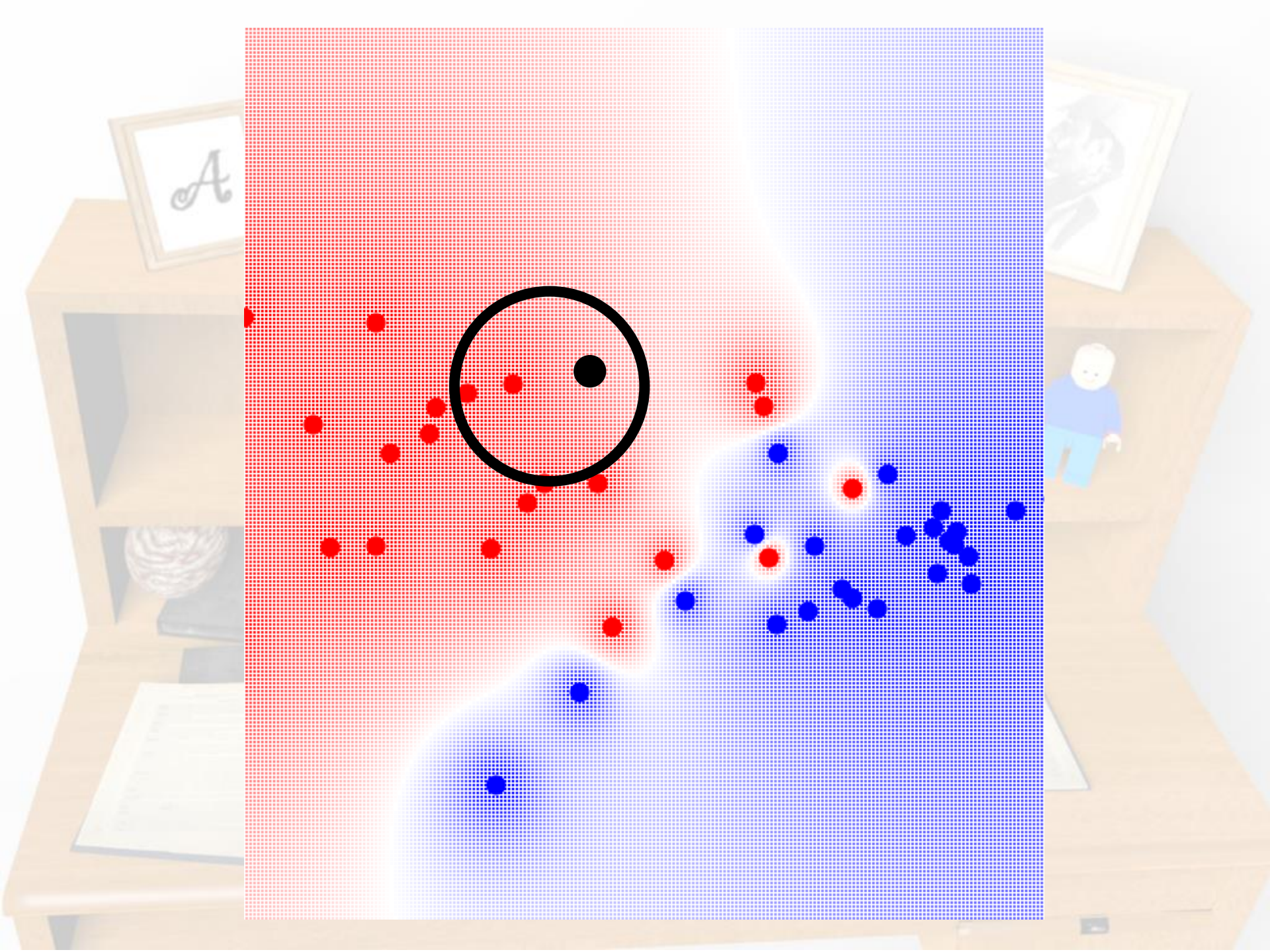
Indexing!











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'
```

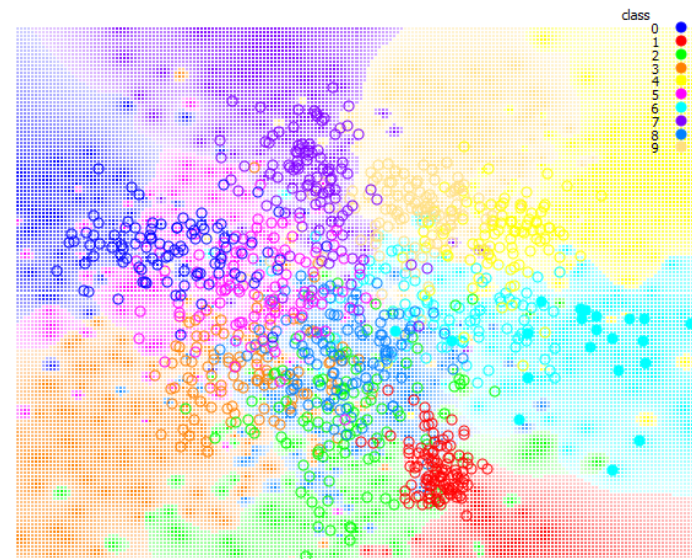
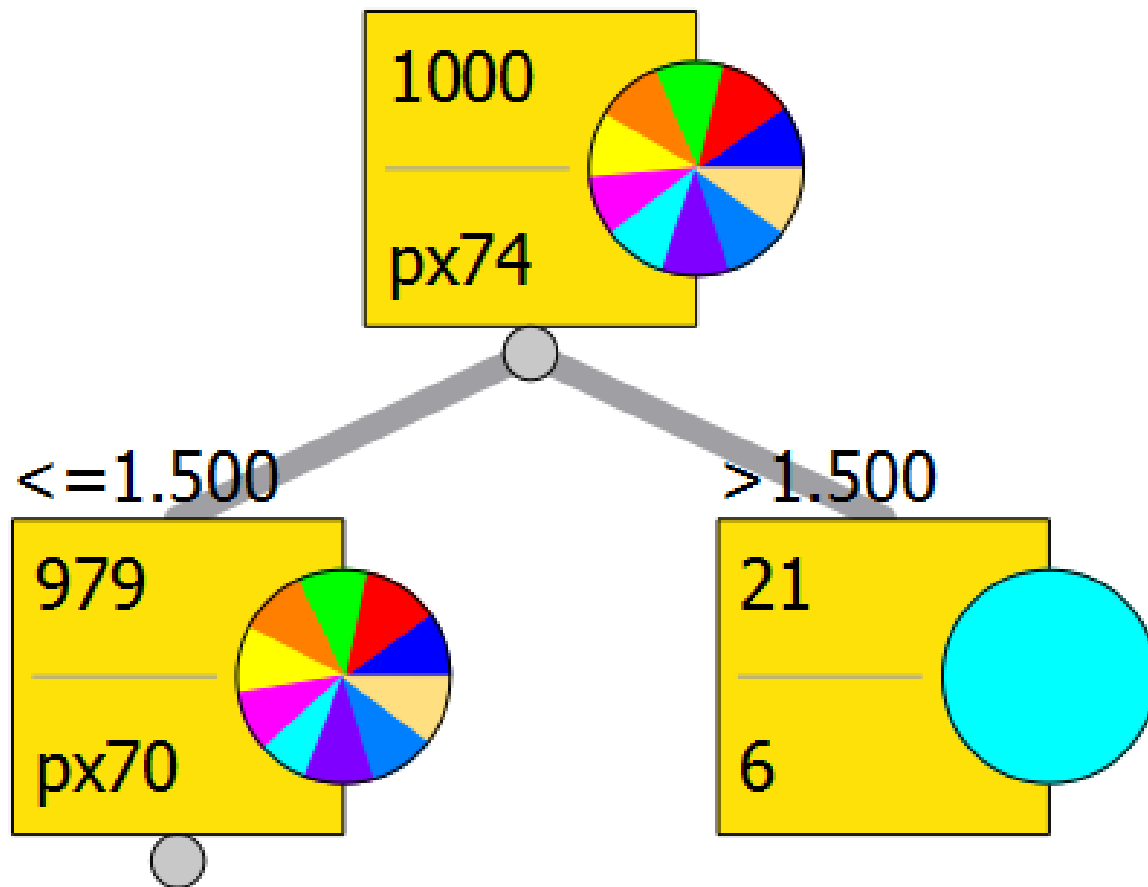
```
    else:
```

```
        class = '2'
```

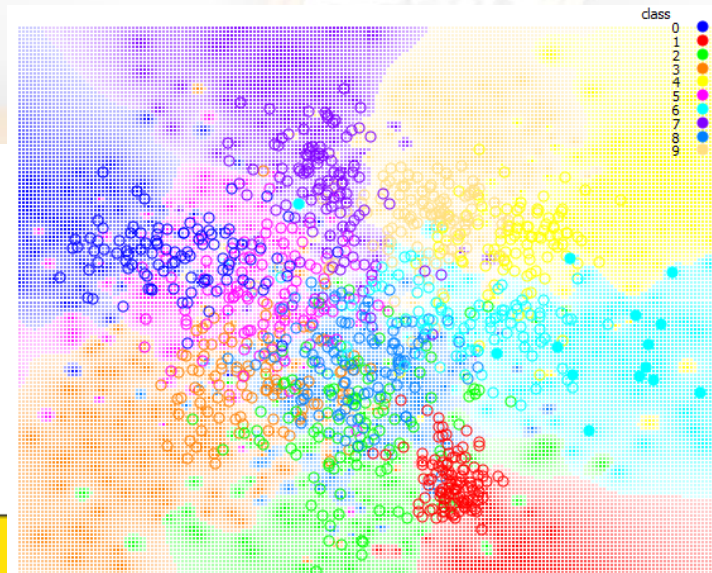
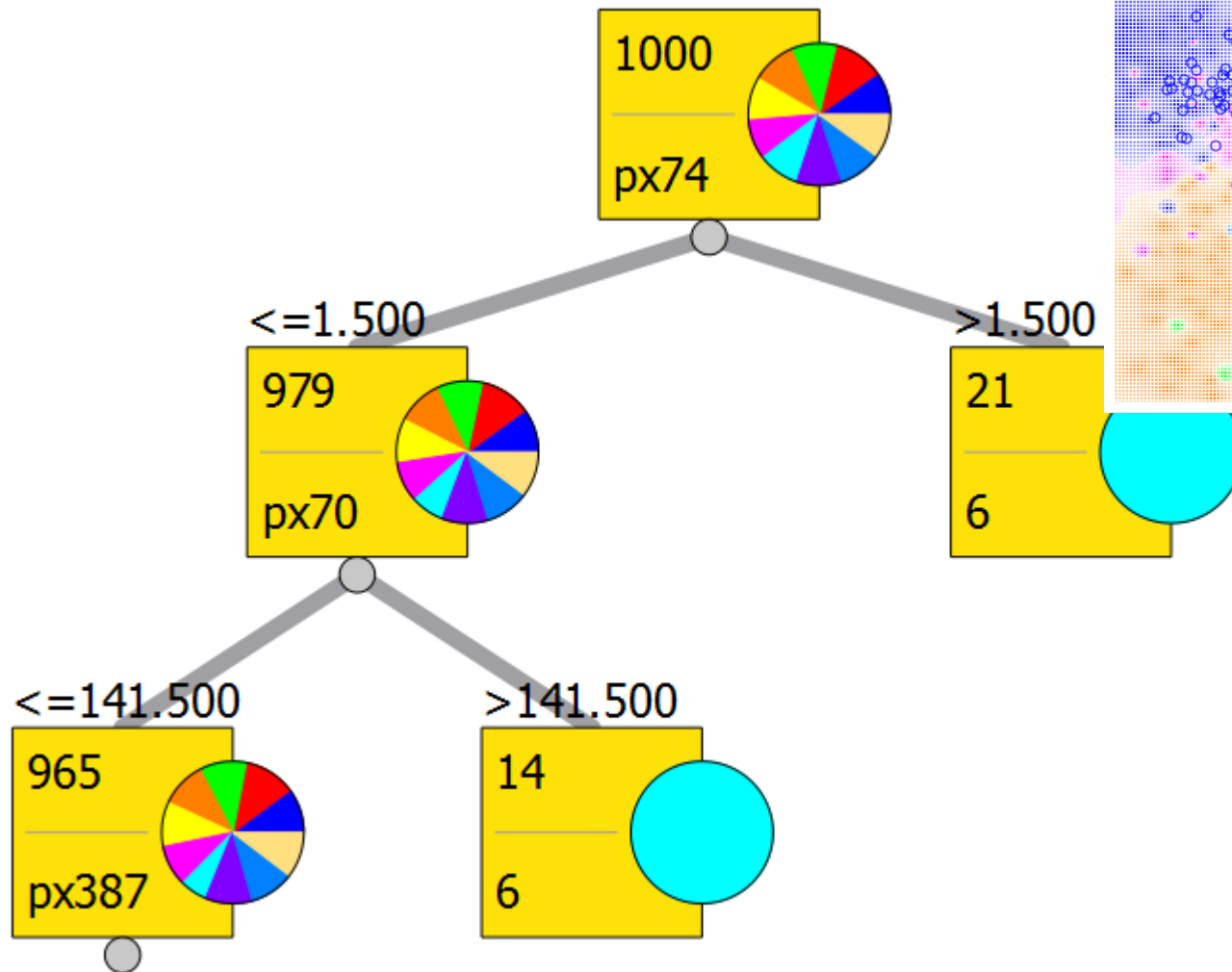
```
else:
```

```
    class = '3'
```

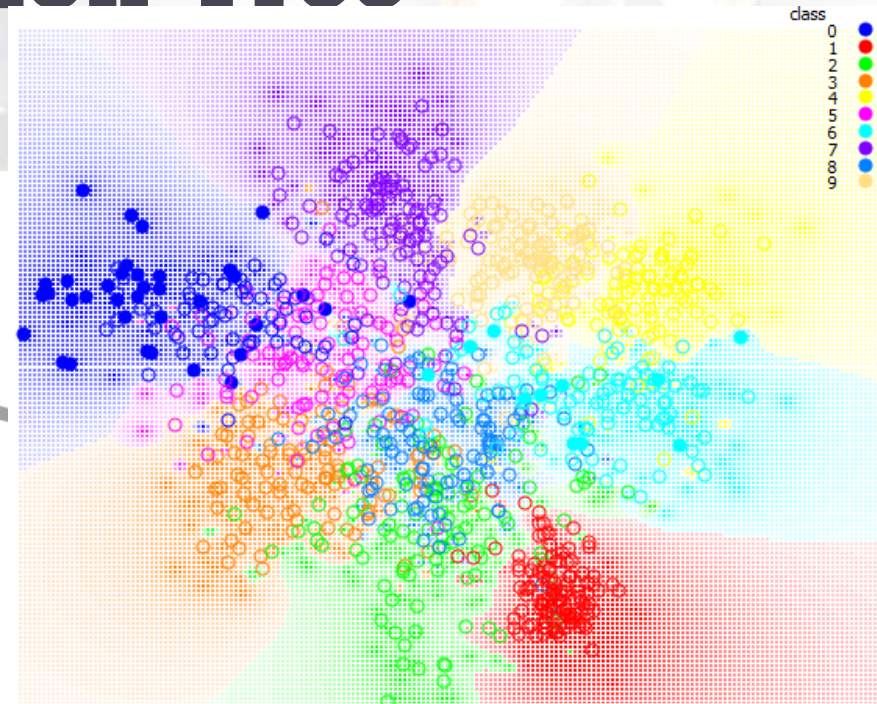
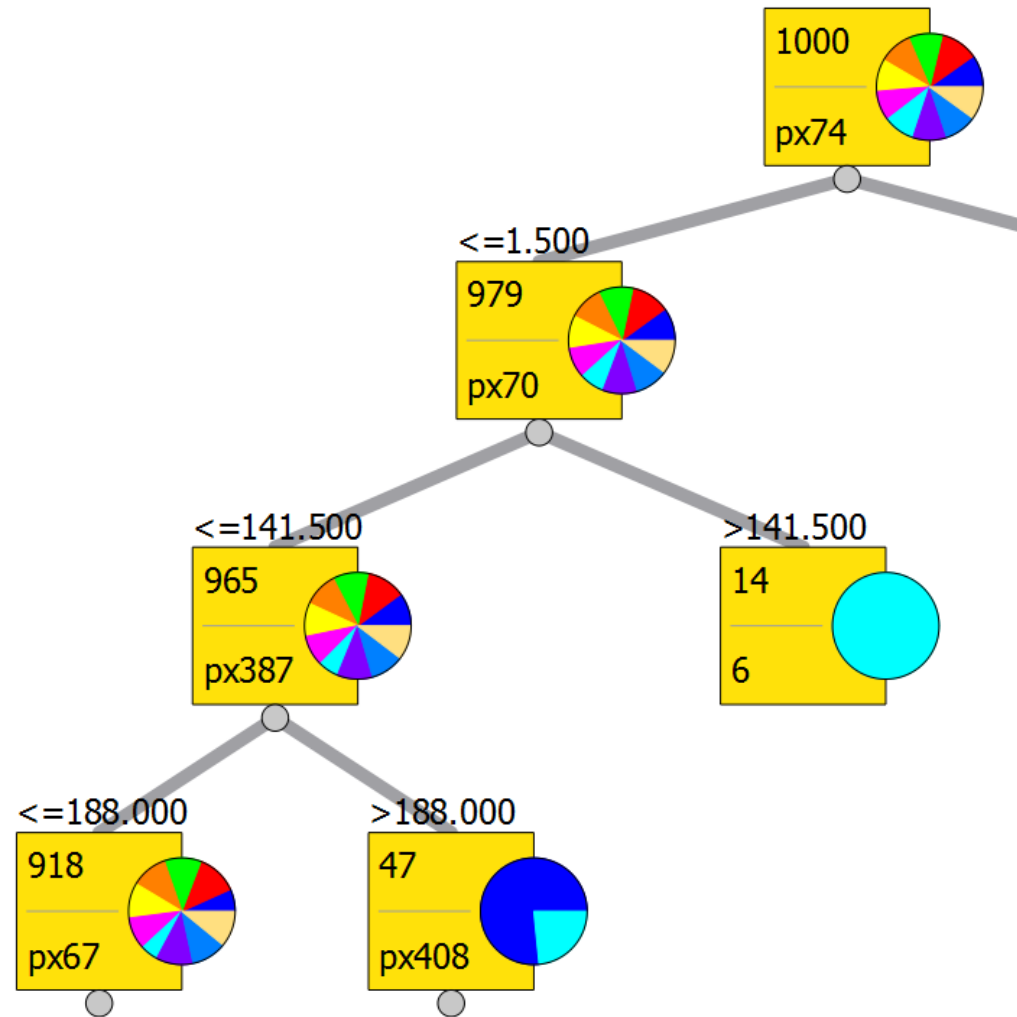
Classification Tree



Classification Tree

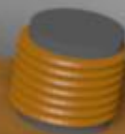


Classification Tree



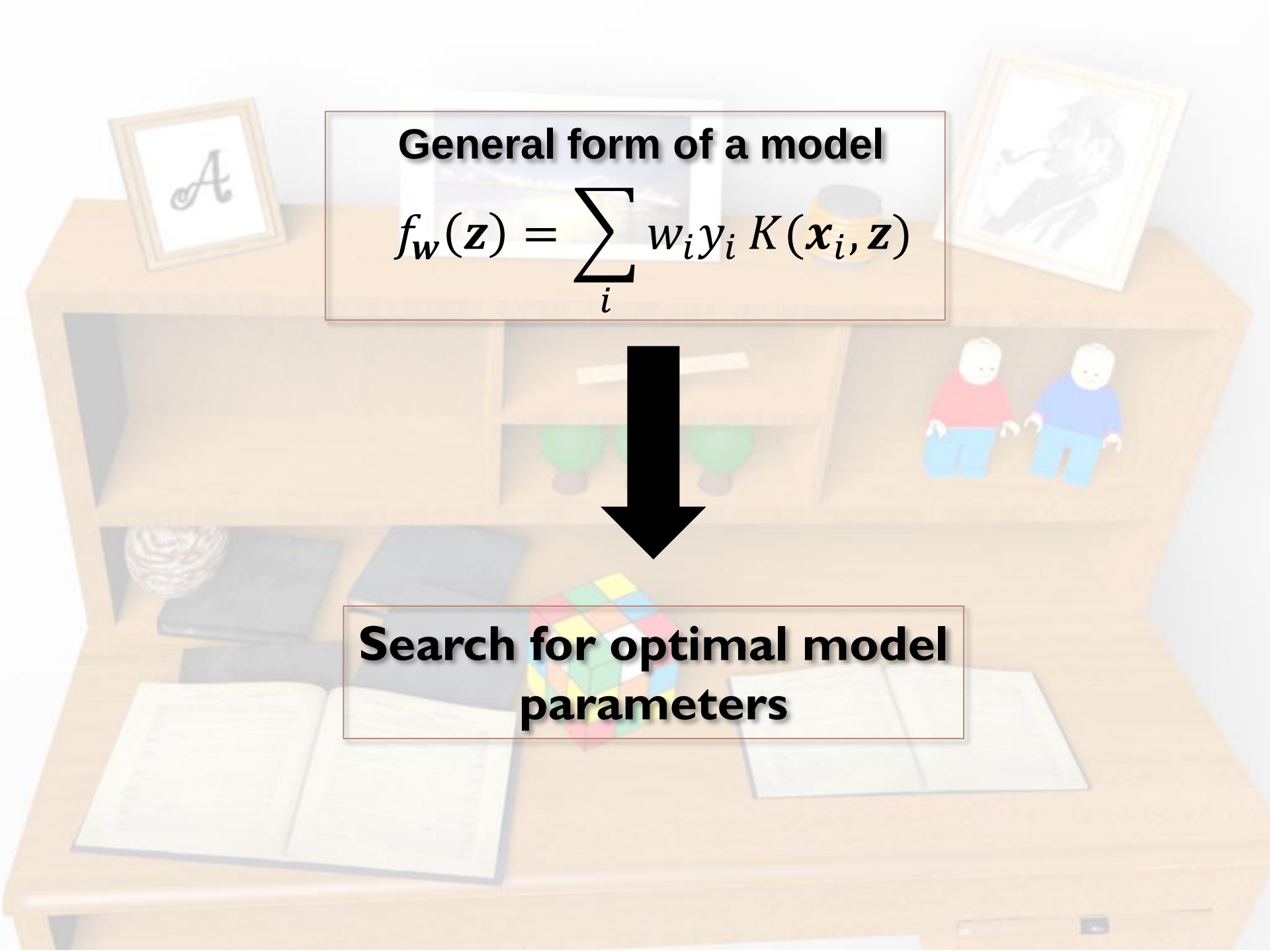
Orange

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





Trees
ID3
C4.5
RegTree

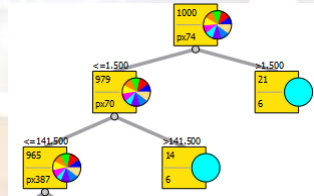


General form of a model

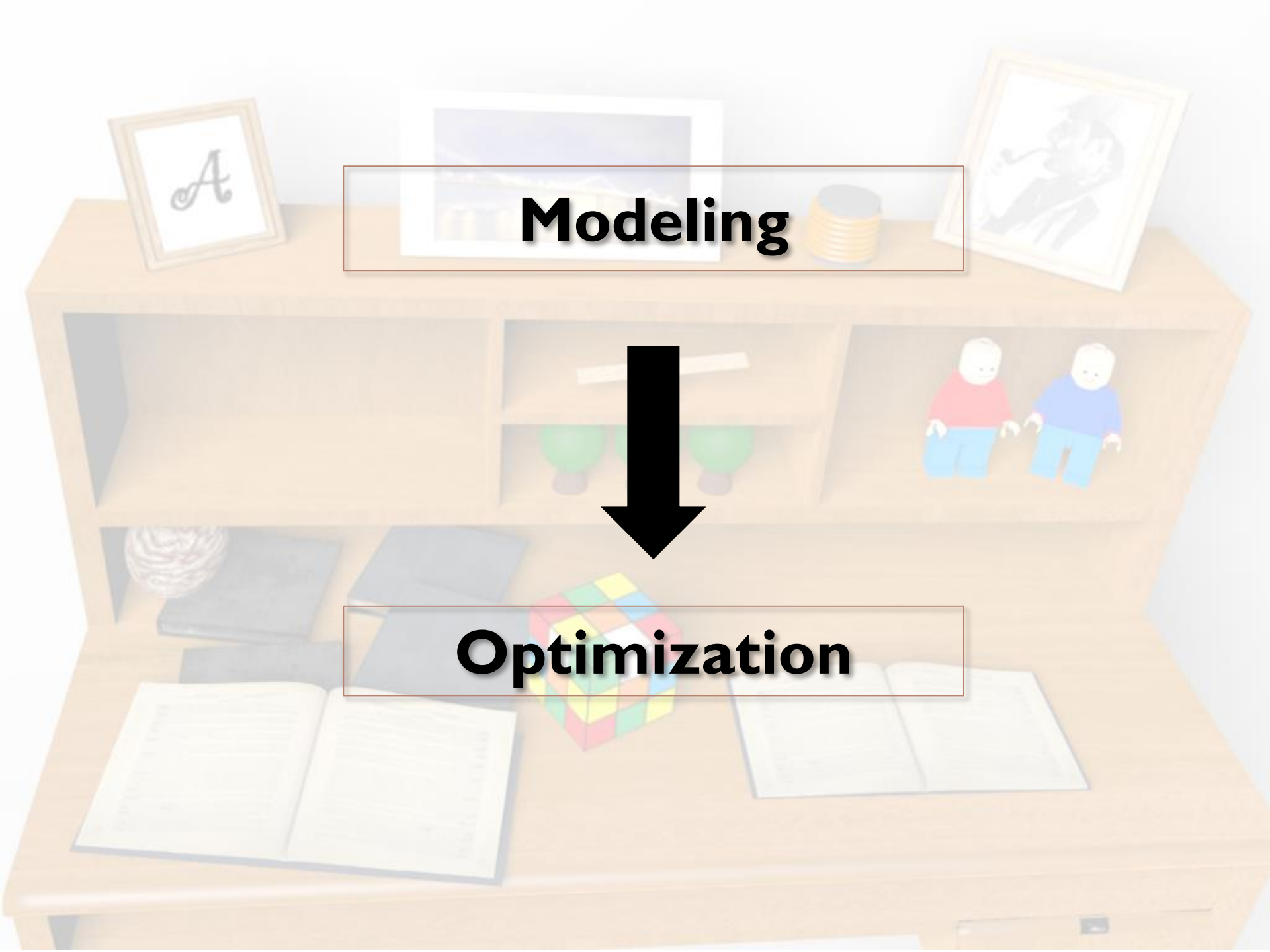
$$f_w(\mathbf{z}) = \sum_i w_i y_i K(\mathbf{x}_i, \mathbf{z})$$

**Search for optimal model
parameters**

General form of a model

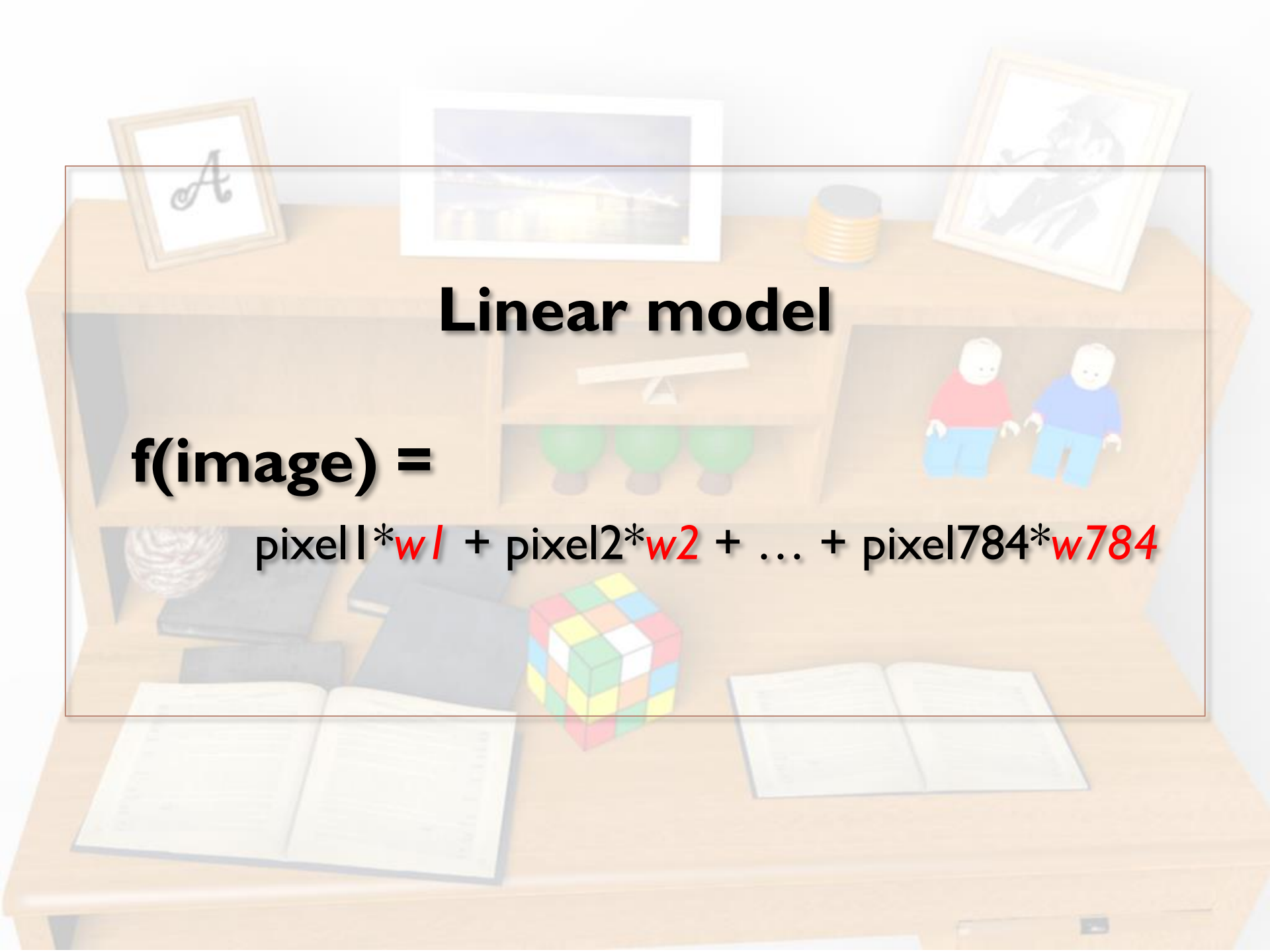


Search for the optimal tree



Modeling

Optimization



Linear model

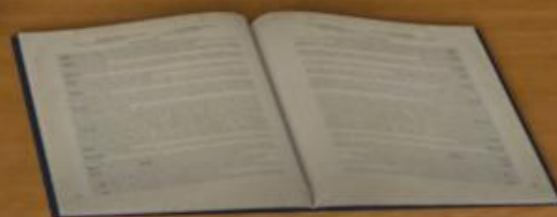
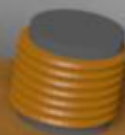
$f(\text{image}) =$

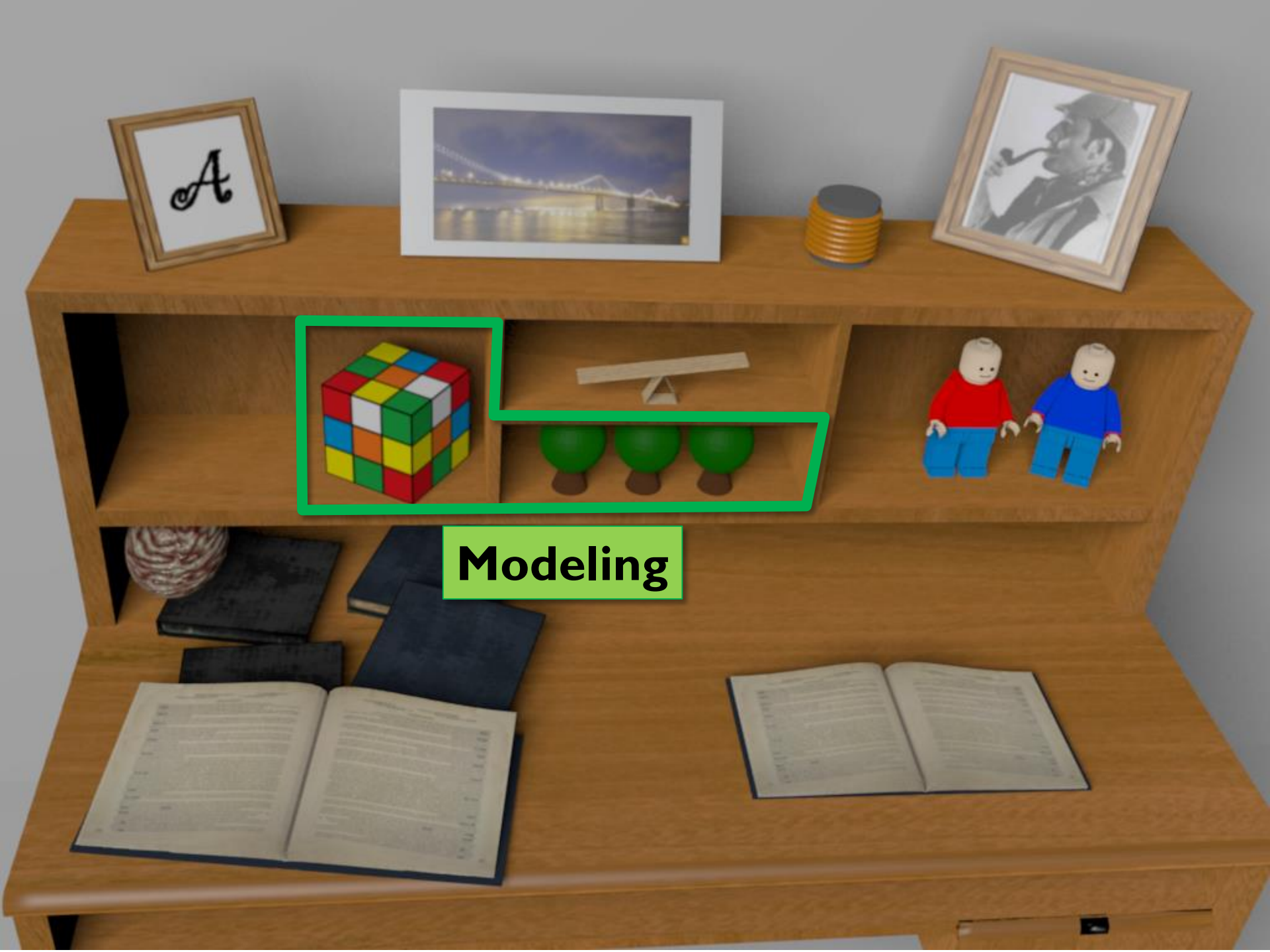
$\text{pixel1} * w1 + \text{pixel2} * w2 + \dots + \text{pixel784} * w784$

Linear Classification

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

=> 809/1000





Modeling

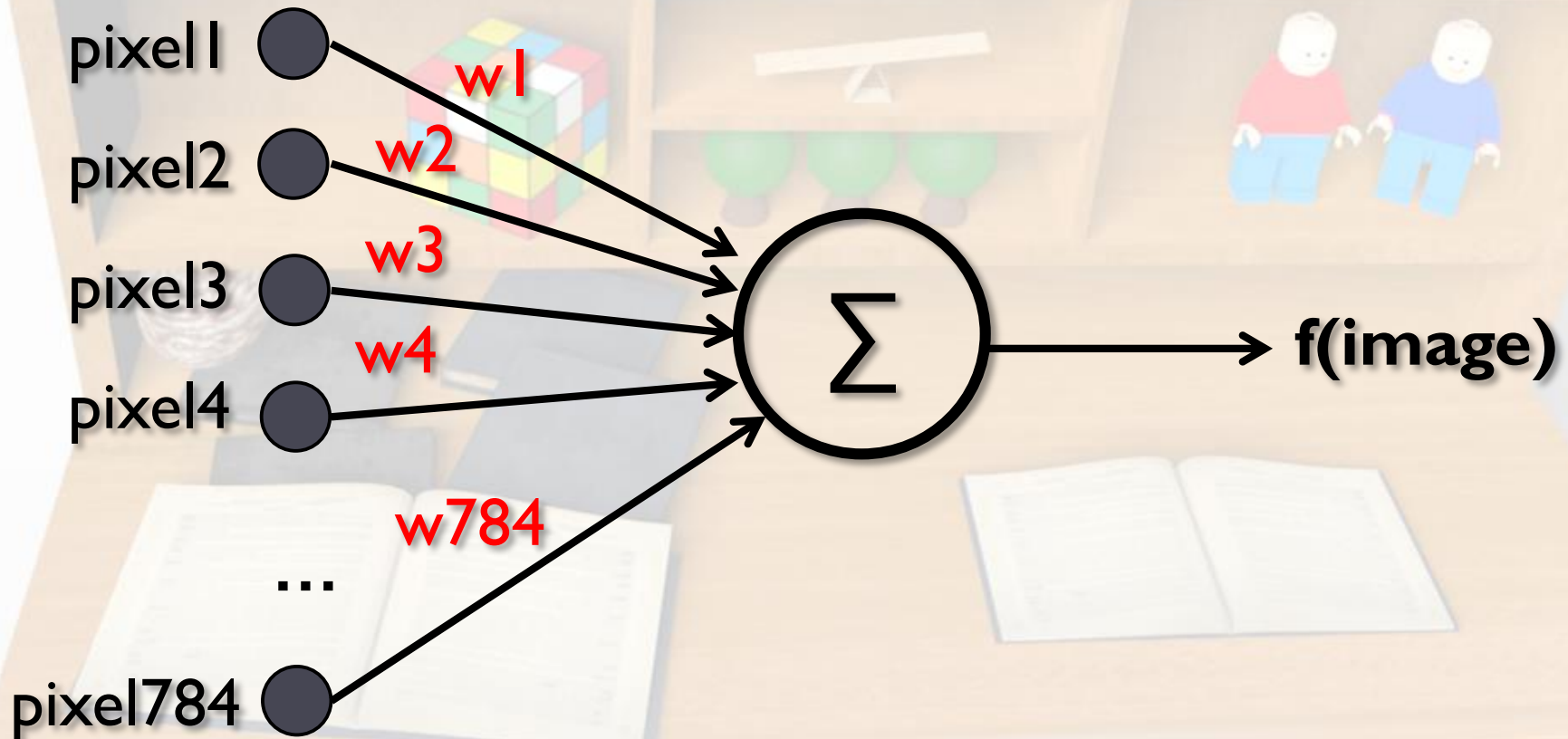
Linear model

$f(\text{image}) =$

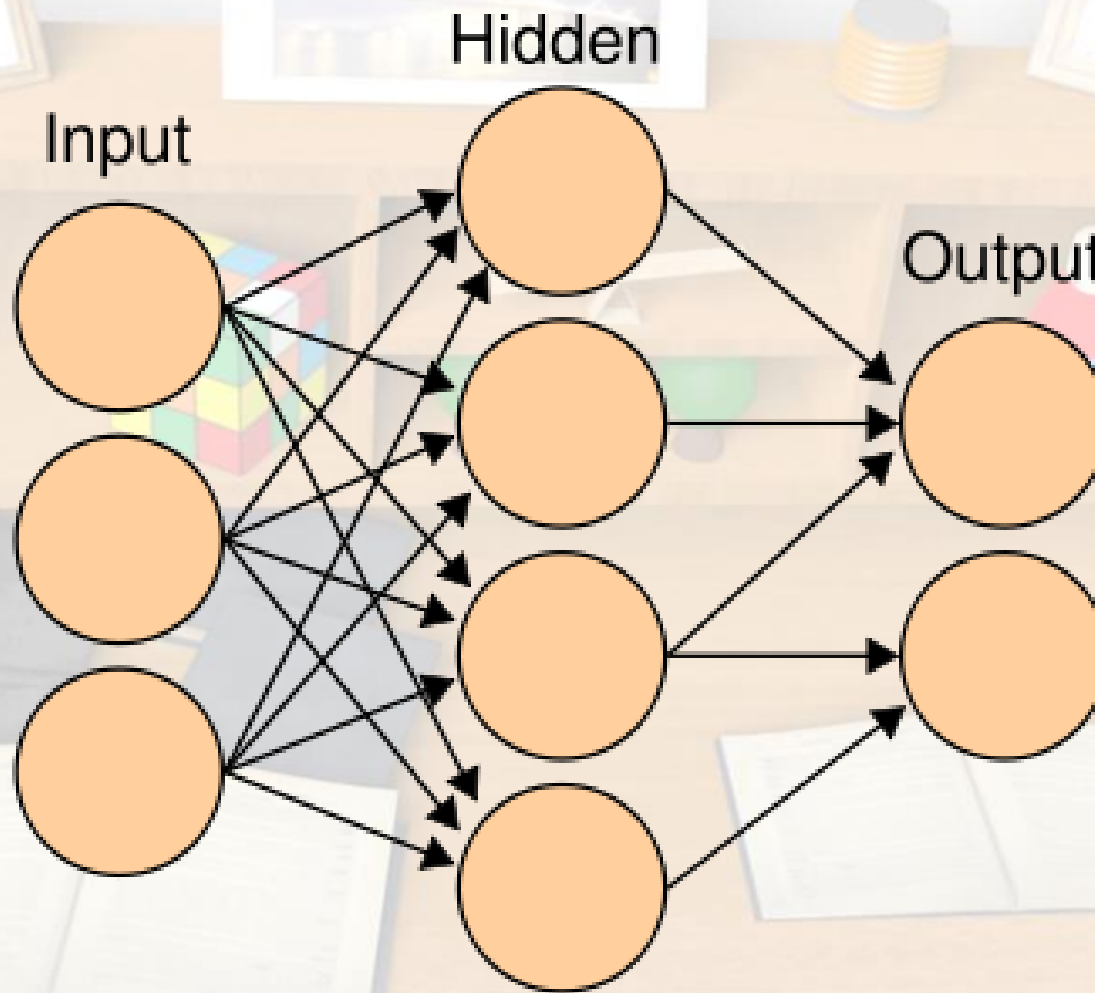
$\text{pixel1} * w1 + \text{pixel2} * w2 + \dots + \text{pixel784} * w784$

$f(\text{image}) =$

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



Neural network

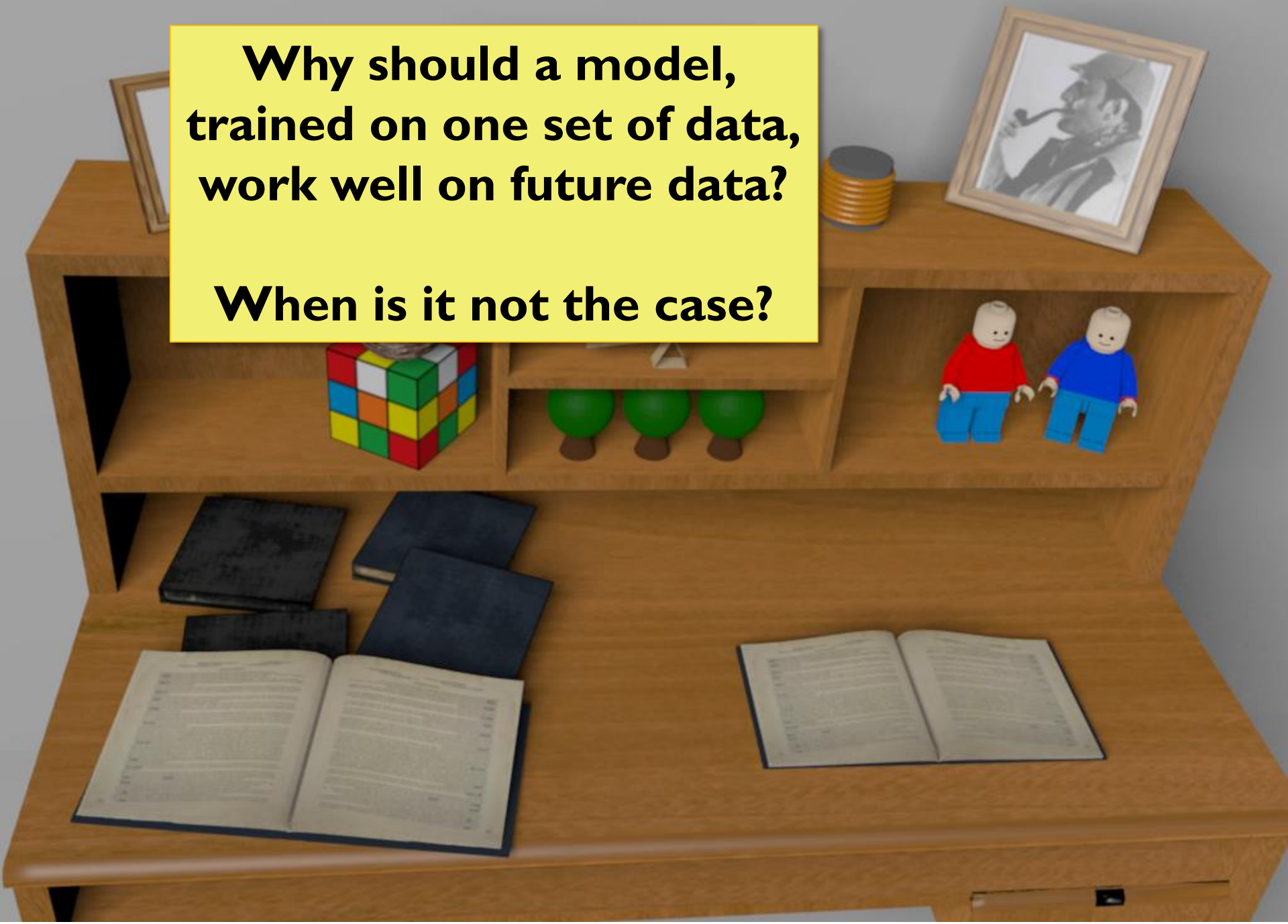


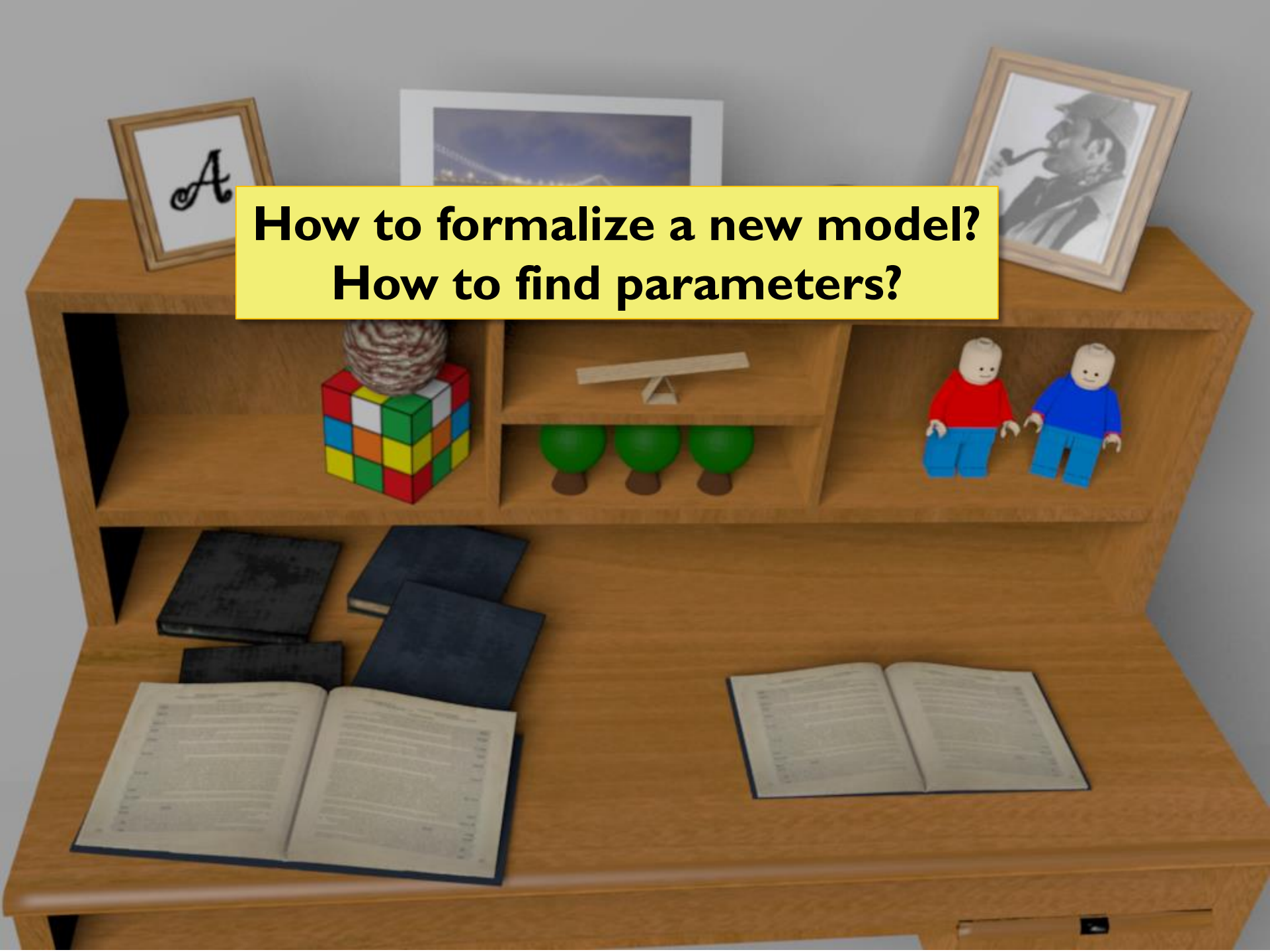


Modeling

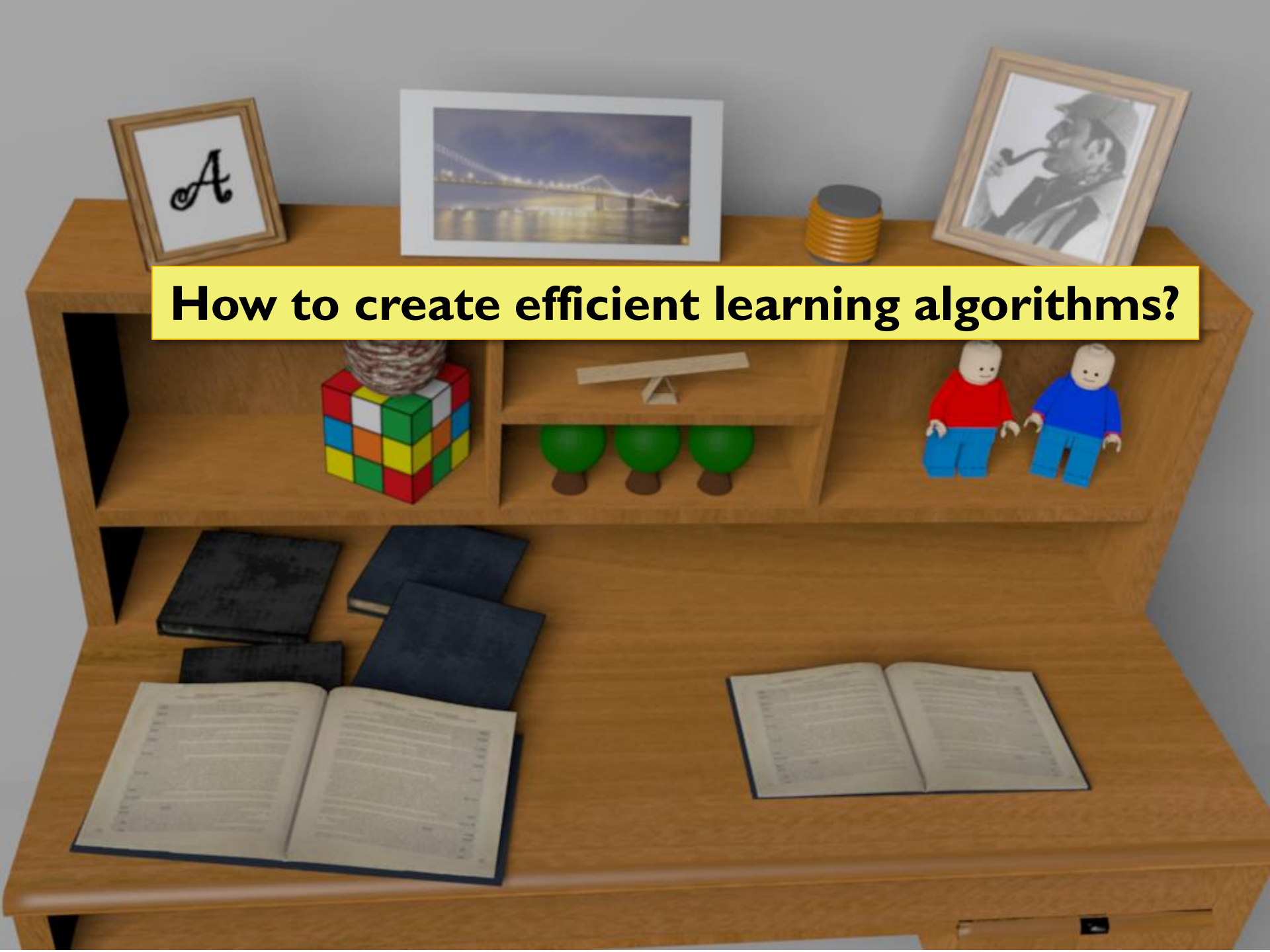
**Why should a model,
trained on one set of data,
work well on future data?**

When is it not the case?





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

A wooden desk with a built-in shelf. On the shelf, from left to right, are a framed letter 'A', a framed picture of a bridge at night, a stack of yellow and grey rings, and a framed black and white photo of a man smoking a pipe. Below the shelf, on the left, is a Rubik's cube with a small woven basket on top. In the center are three green spherical weights on stands and a wooden balance scale. On the right are two baby dolls, one in a red shirt and one in a blue shirt. On the desk surface, there are four dark blue books, two of which are open, showing text. The background is a plain light grey wall.

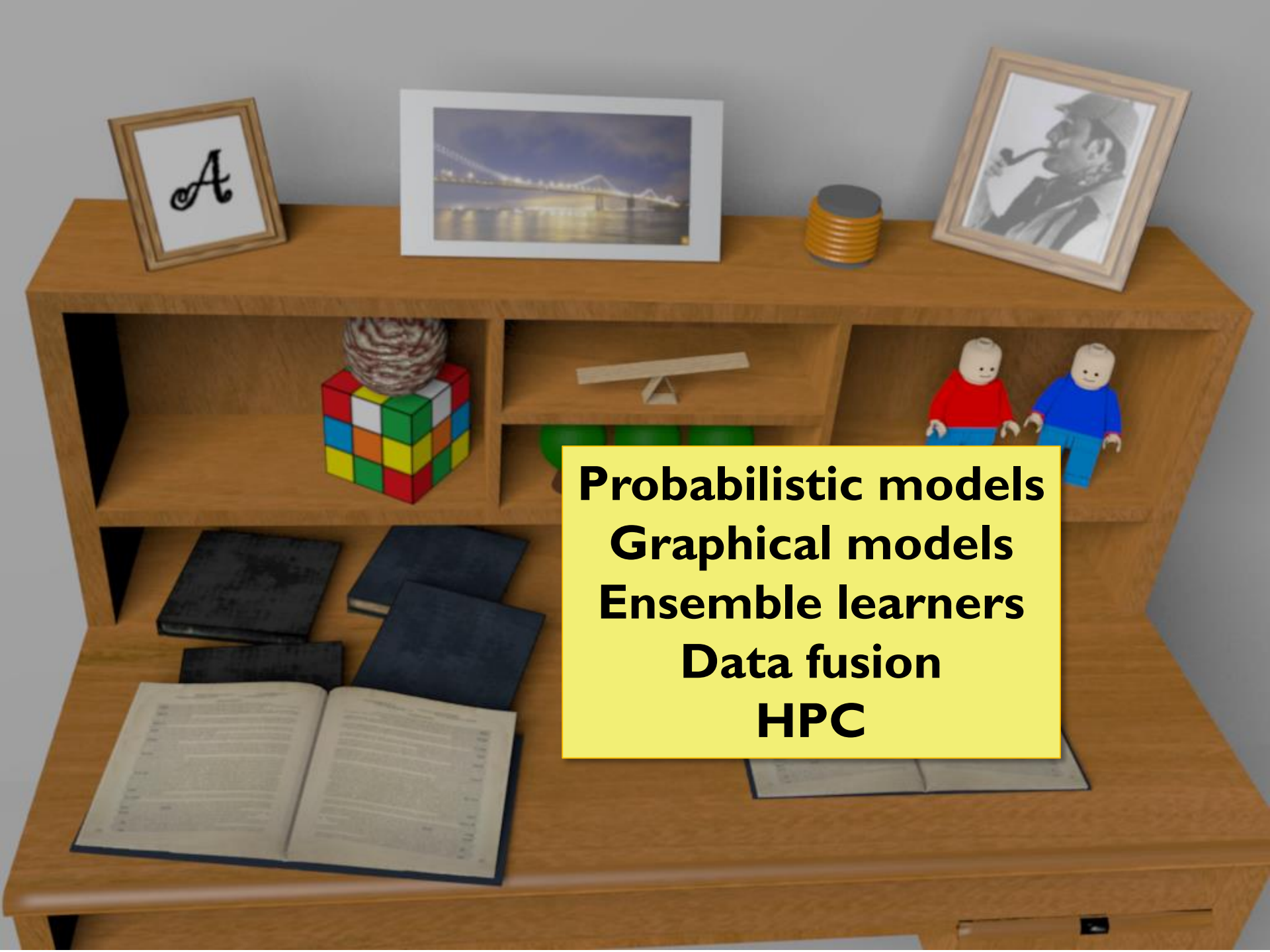
How to create efficient learning algorithms?

A wooden desk with a light-colored wood grain. On the top shelf, there are three framed items: a small wooden frame with the letter 'A', a white frame with a photo of a bridge at night, and a wooden frame with a black and white photo of a man smoking a pipe. Next to the pipe smoker is a stack of yellow and grey coasters. The middle shelf has a Rubik's cube, three green spherical ornaments on brown stands, and two small LEGO minifigures. The bottom shelf has several books, including two open ones showing text and some dark blue closed ones.

How to handle structured data?

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 coasters. On the middle shelf, there is a Rubik's cube, a small vase, and two LEGO minifigures. On the bottom shelf, there are several books, including an open one in the foreground. A yellow text box is overlaid on the center of the image.

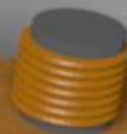
Unsupervised learning
Semi-supervised learning
On-line learning
Active learning
Multi-instance learning
Reinforcement learning



Probabilistic models
Graphical models
Ensemble learners
Data fusion
HPC

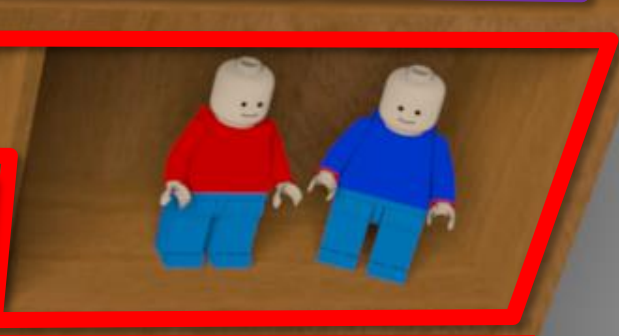
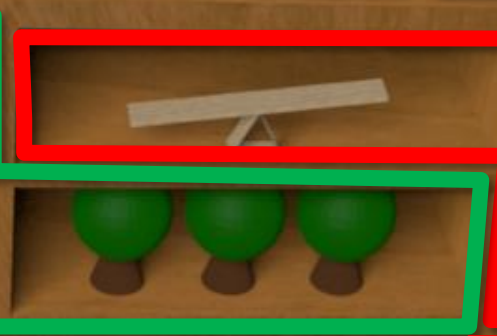
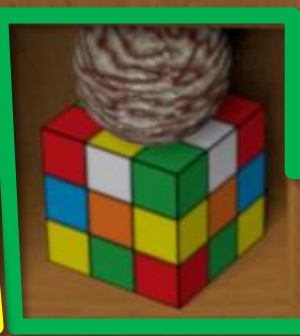


Tools:
R, Weka, RapidMiner,
Orange, scikits-learn,
MLPy, MDP, PyBrain, ...



“Unformalizable” problems

Deduction and Induction



**Theory and
Practice**

Model-based

Instance-based

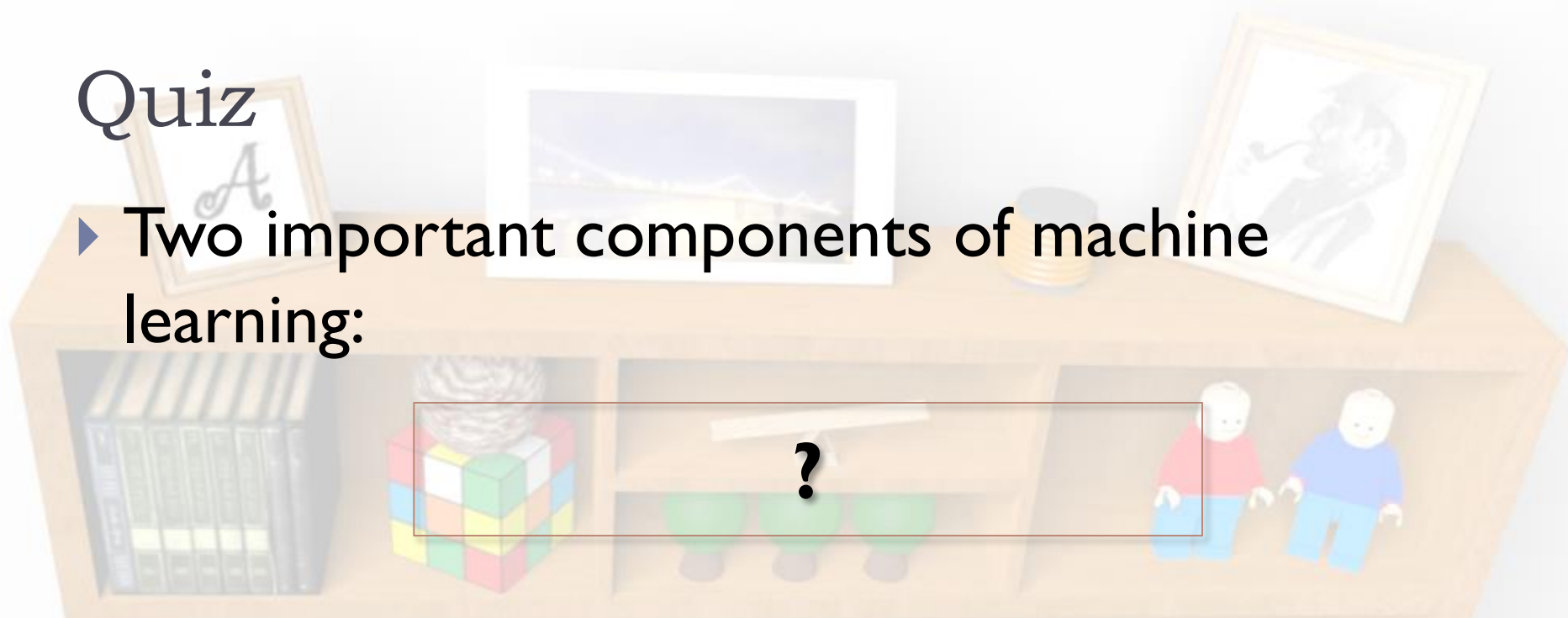
Quiz

- ▶ 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

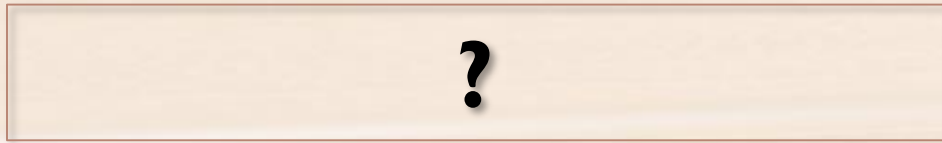
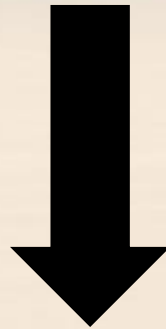
$$\operatorname{argmin}_w \text{_____} + \lambda \text{_____}$$

Quiz

- ▶ Two important components of machine learning:



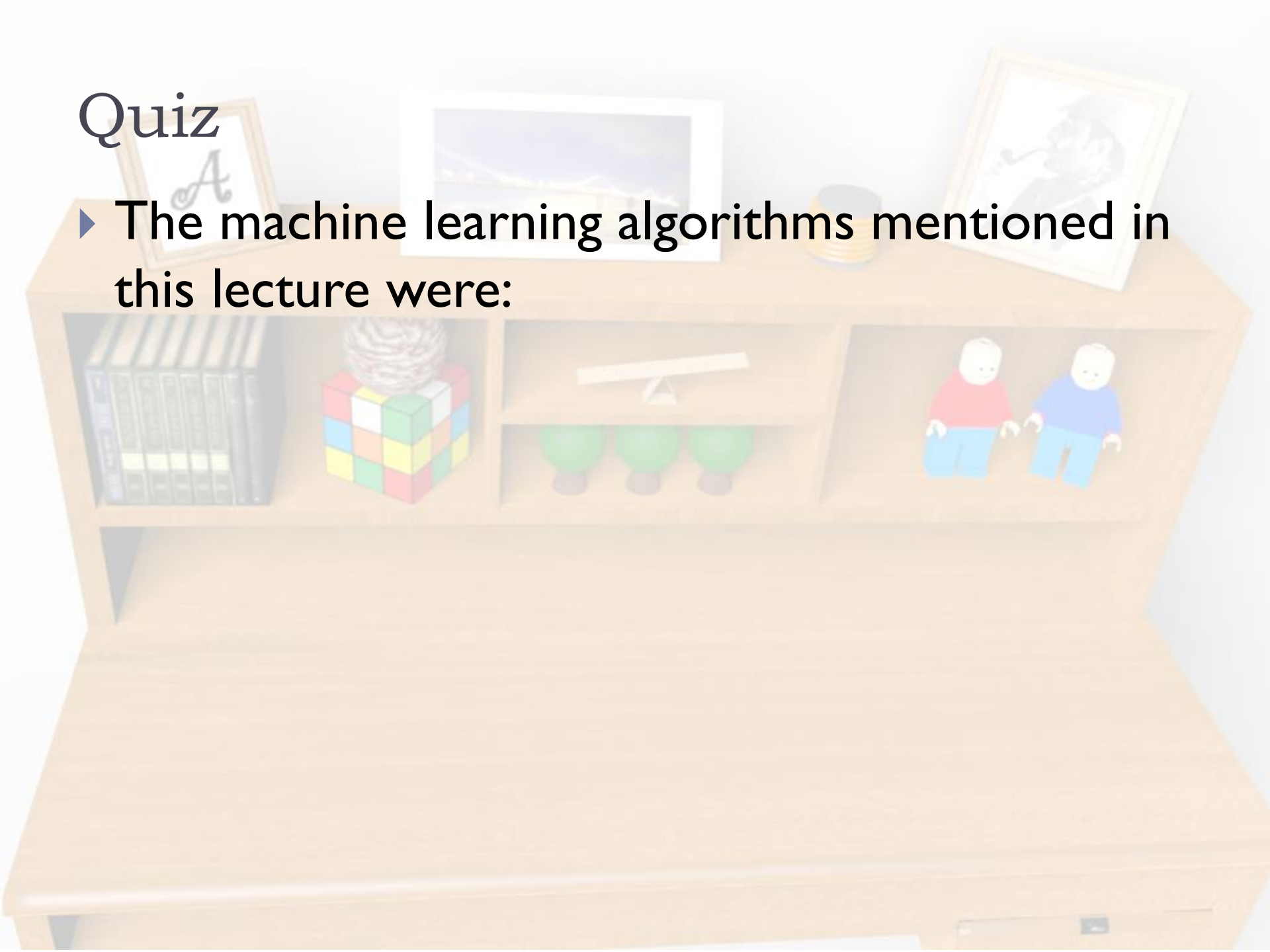
?



?

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?