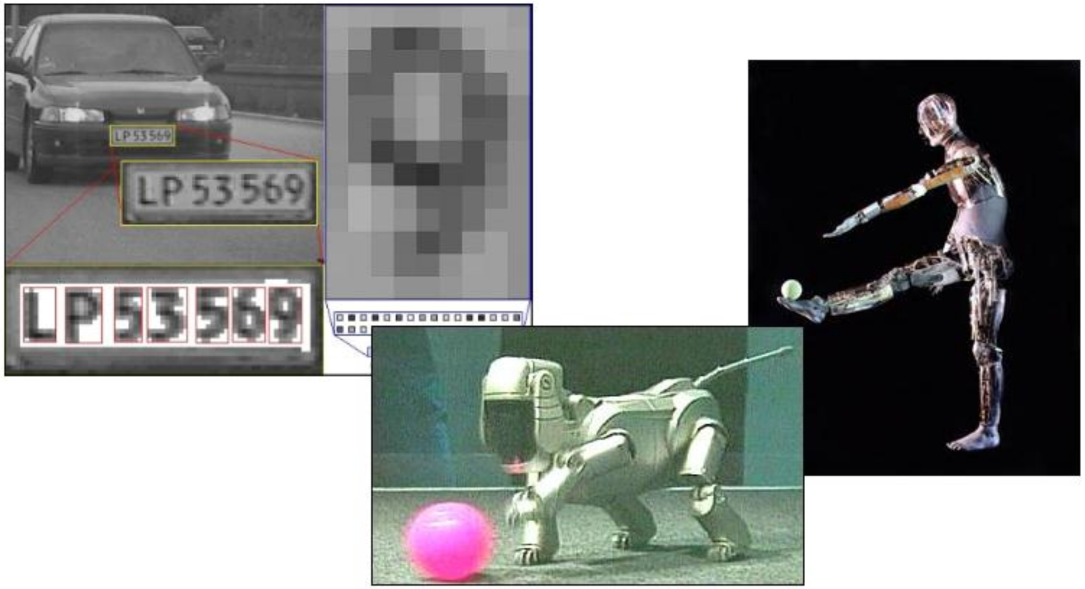


Welcome to Machine Learning

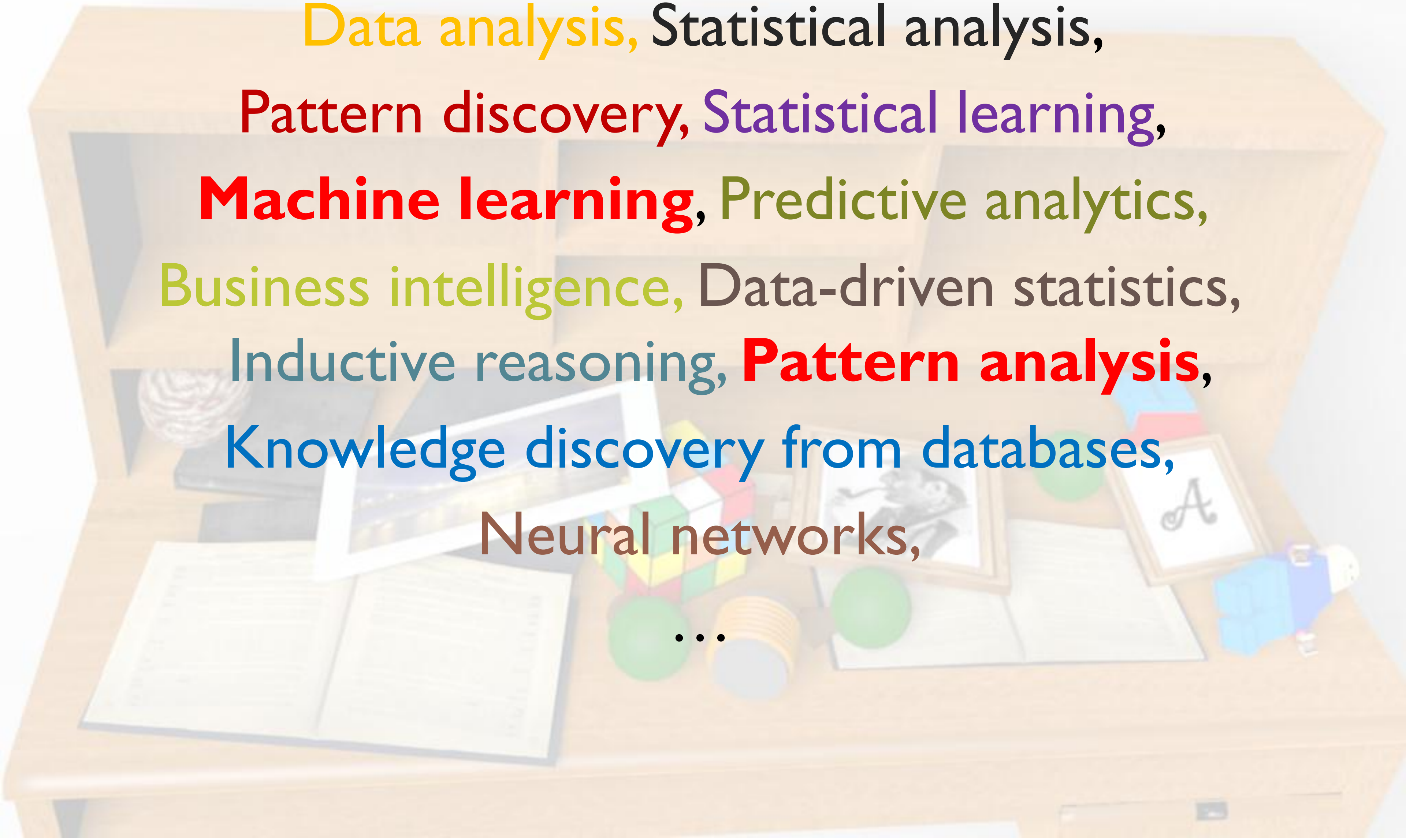
Konstantin Tretyakov
http://kt.era.ee

AACIMP 2012
August 2012

STACC Software Technology and Applications Competence Center

BIIT

TARTU ÜLIKOOL · UNIVERSITAS TARTUENSIS · 1632



Data mining,

Data analysis, Statistical analysis,

Pattern discovery, Statistical learning,

Machine learning, Predictive analytics,

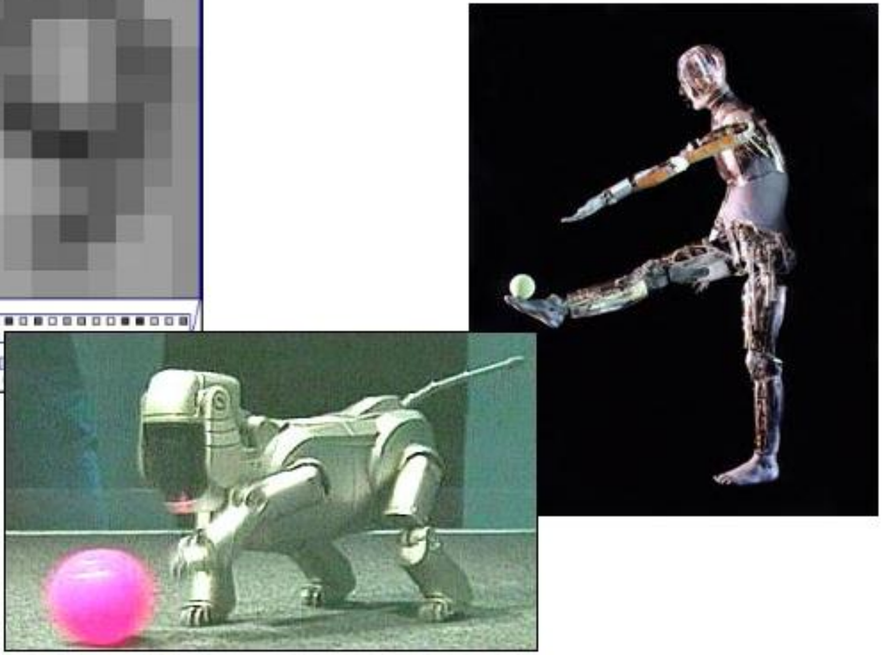
Business intelligence, Data-driven statistics,

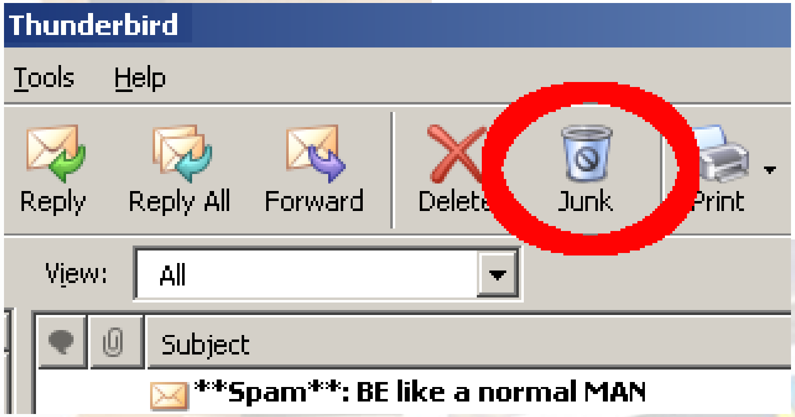
Inductive reasoning, **Pattern analysis**,

Knowledge discovery from databases,

Neural networks,

…





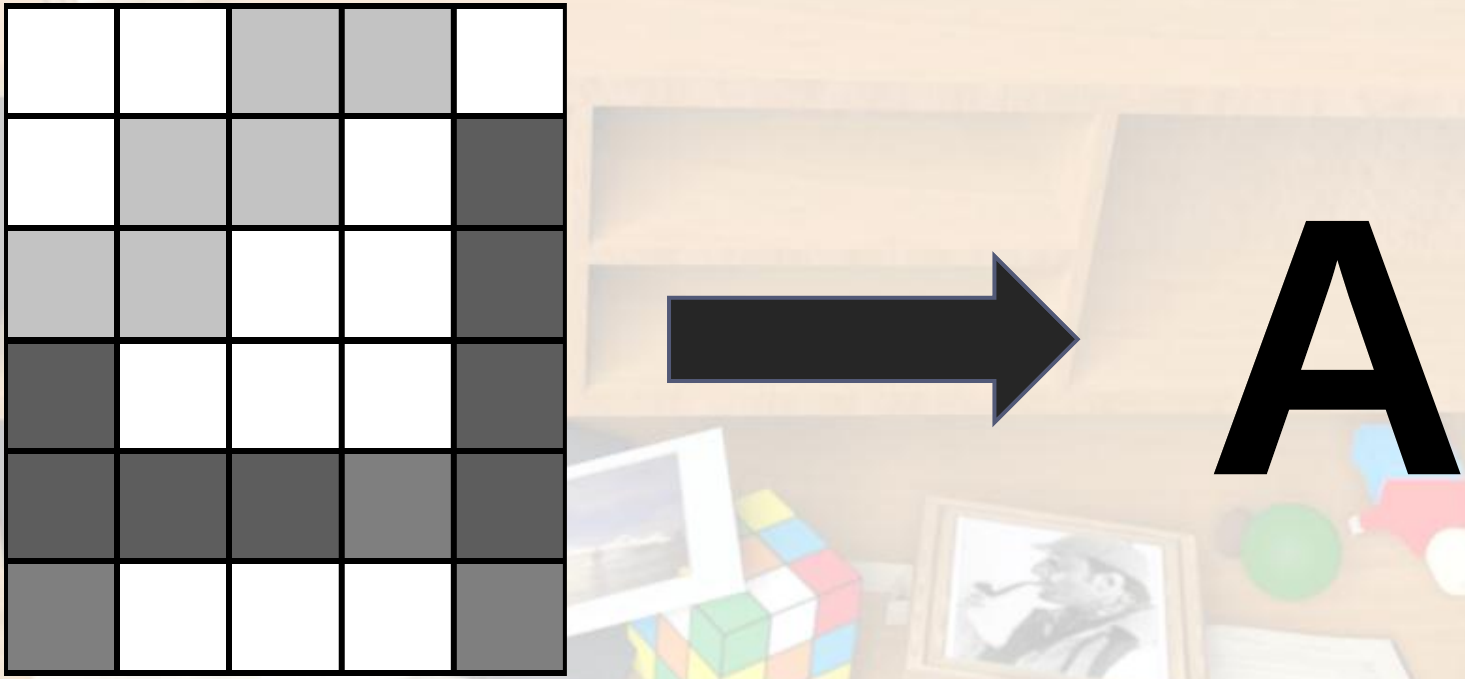
Spam: BE like a normal MAN

Google translate

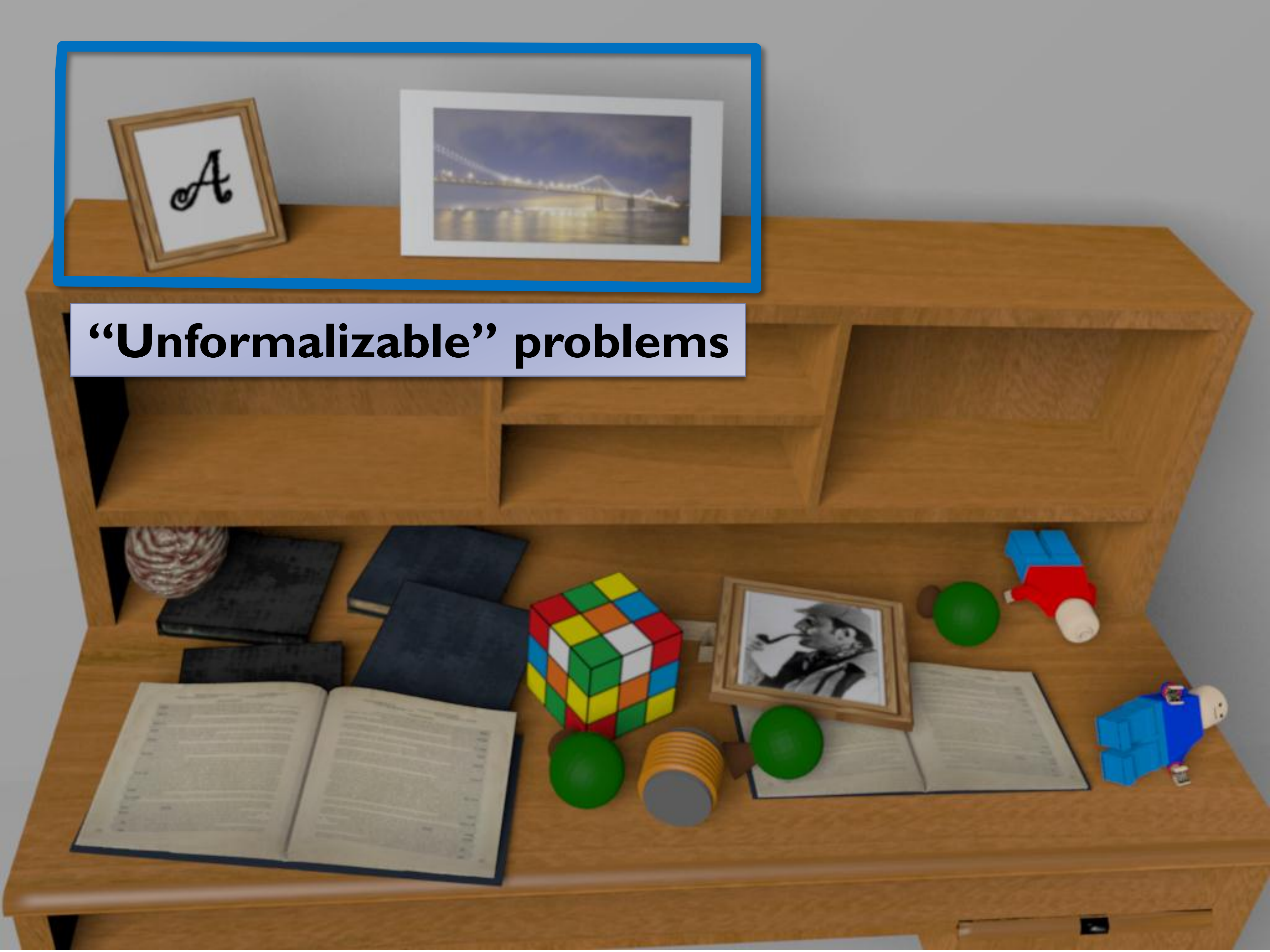
NETFLIX

A

A


A

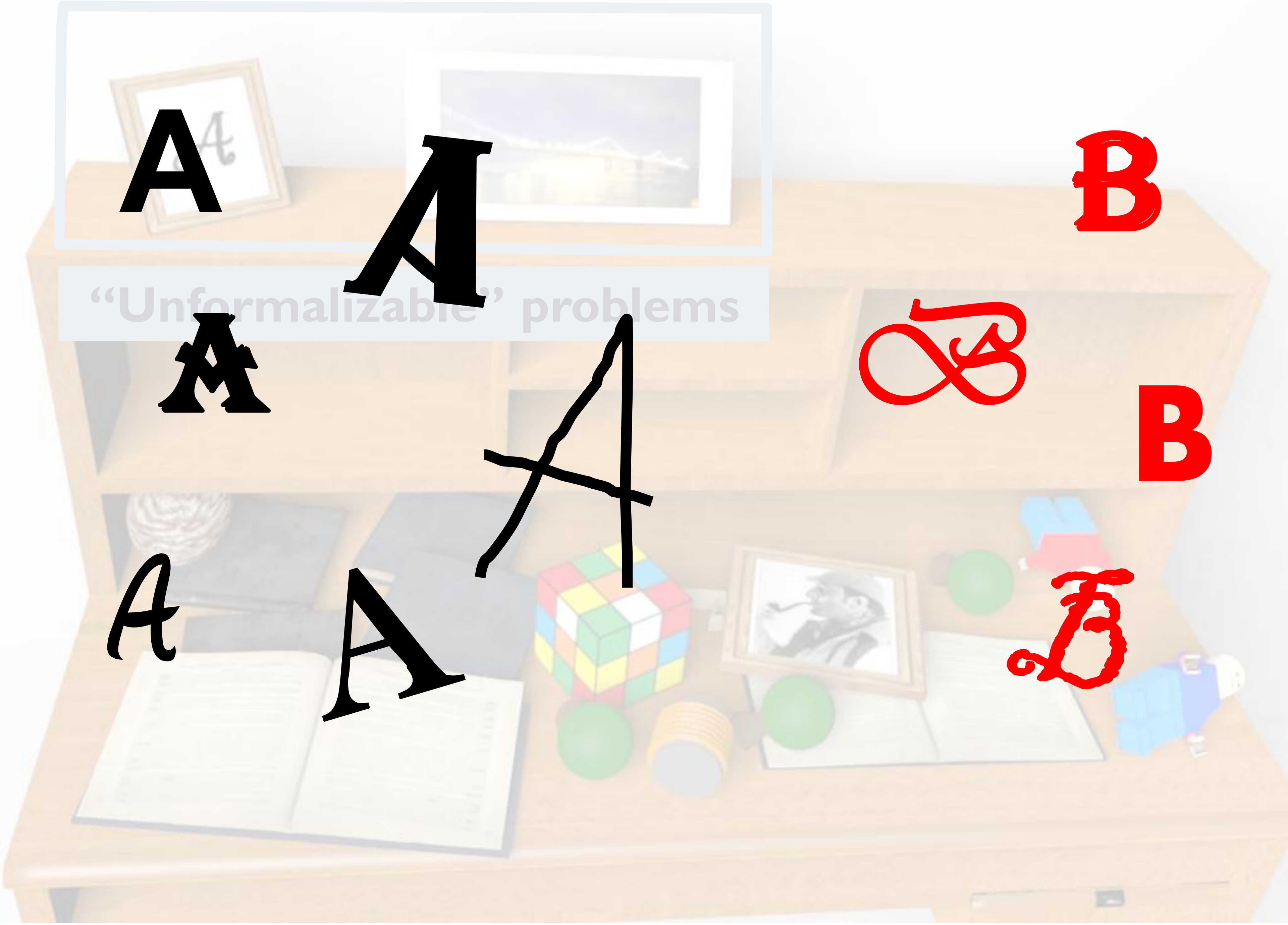
A



“Unformalizable” problems

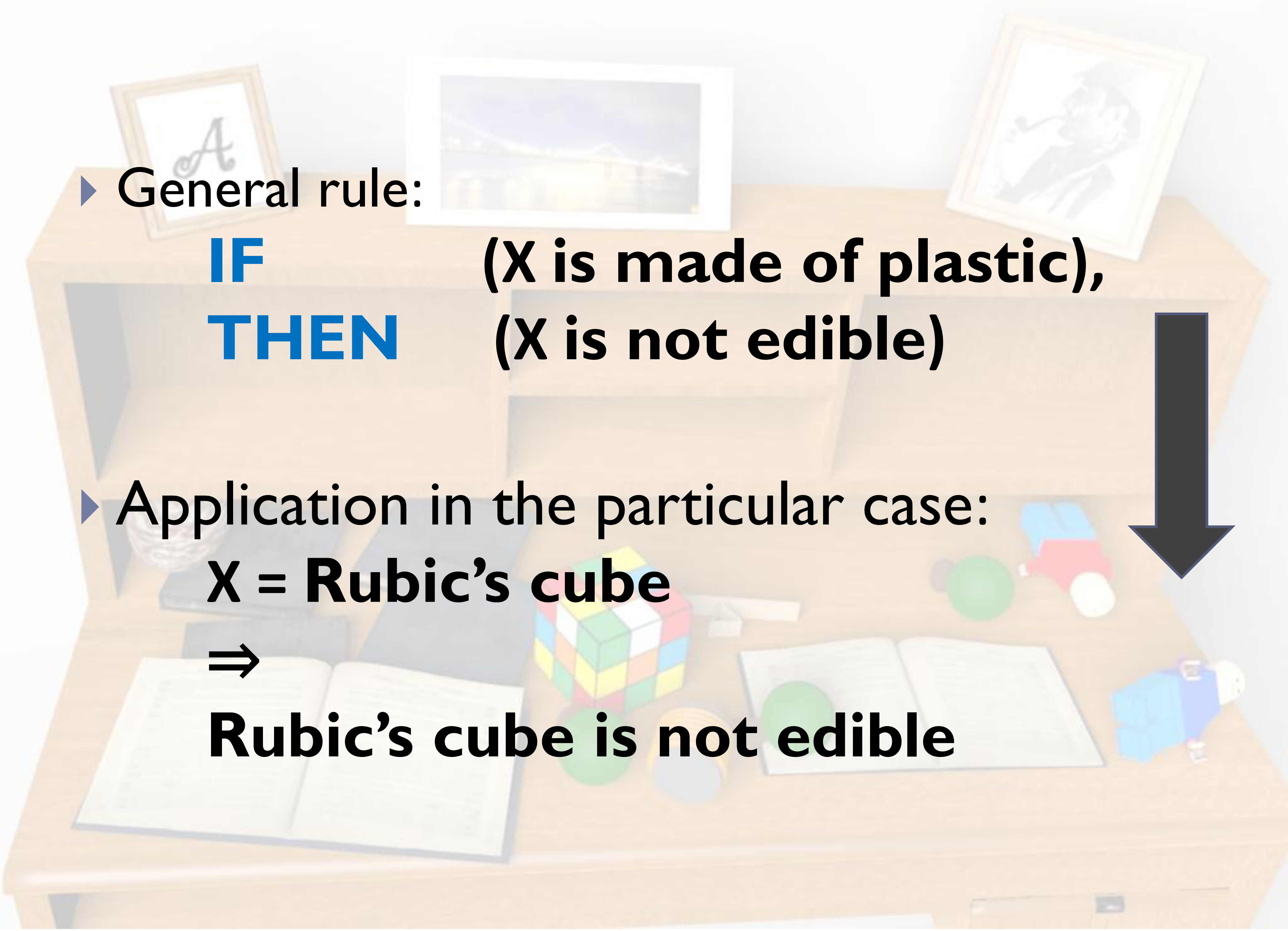
“Unformalizable” problems

“Unformalizable” problems



A

A



- General rule:

  IF **(X is made of plastic),**

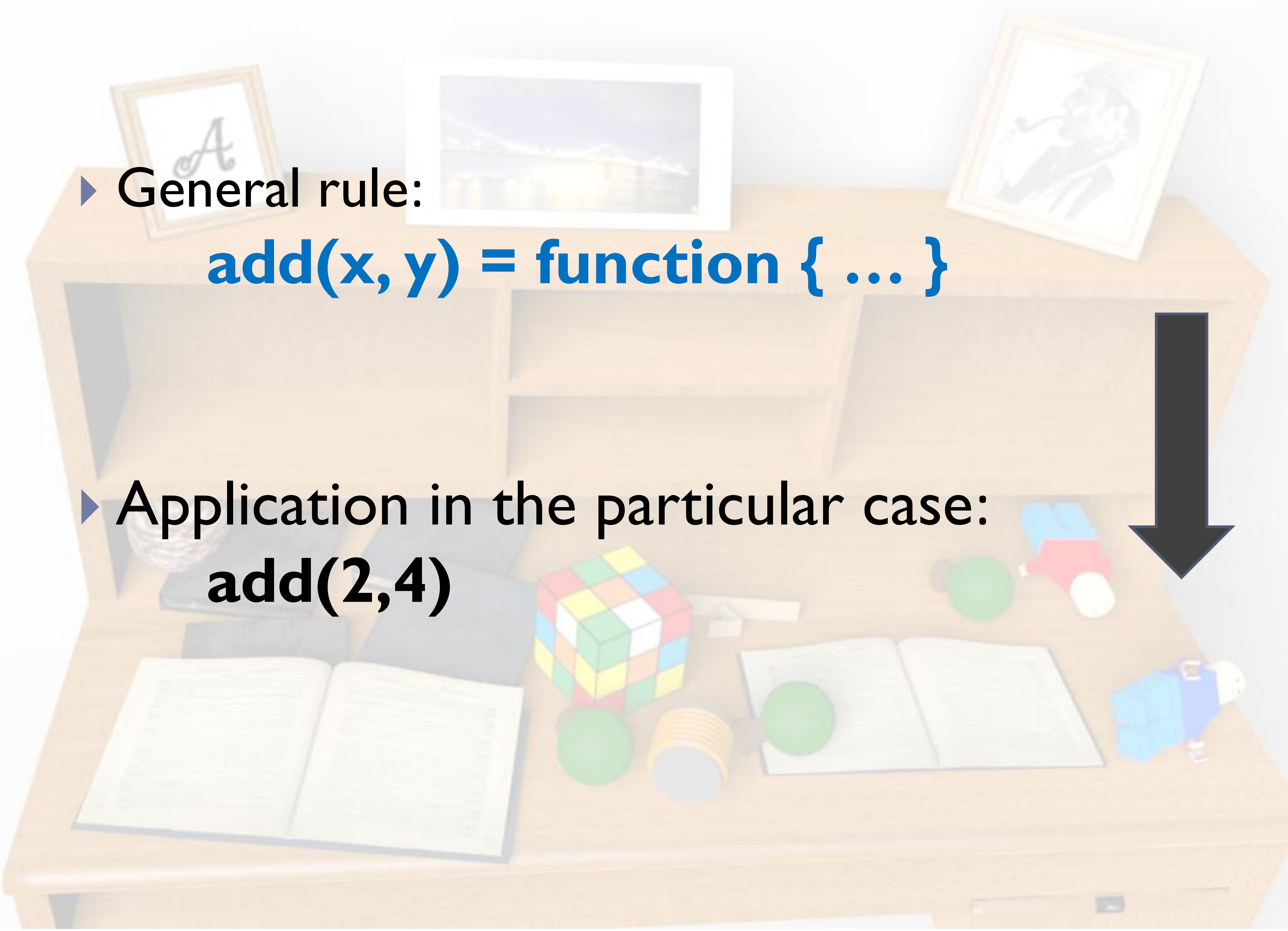
  THEN **(X is not edible)**

- Application in the particular case:

  X = Rubic’s cube

  ⇒

  Rubic’s cube is not edible



- General rule:

  add(x, y) = function { … }

- Application in the particular case:

  add(2,4)

- General rule:

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

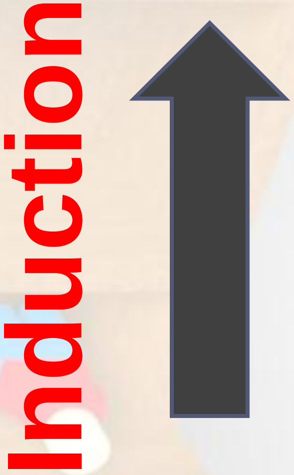
- Particular cases:

add(2,4) = 6

add(5,3) = 8

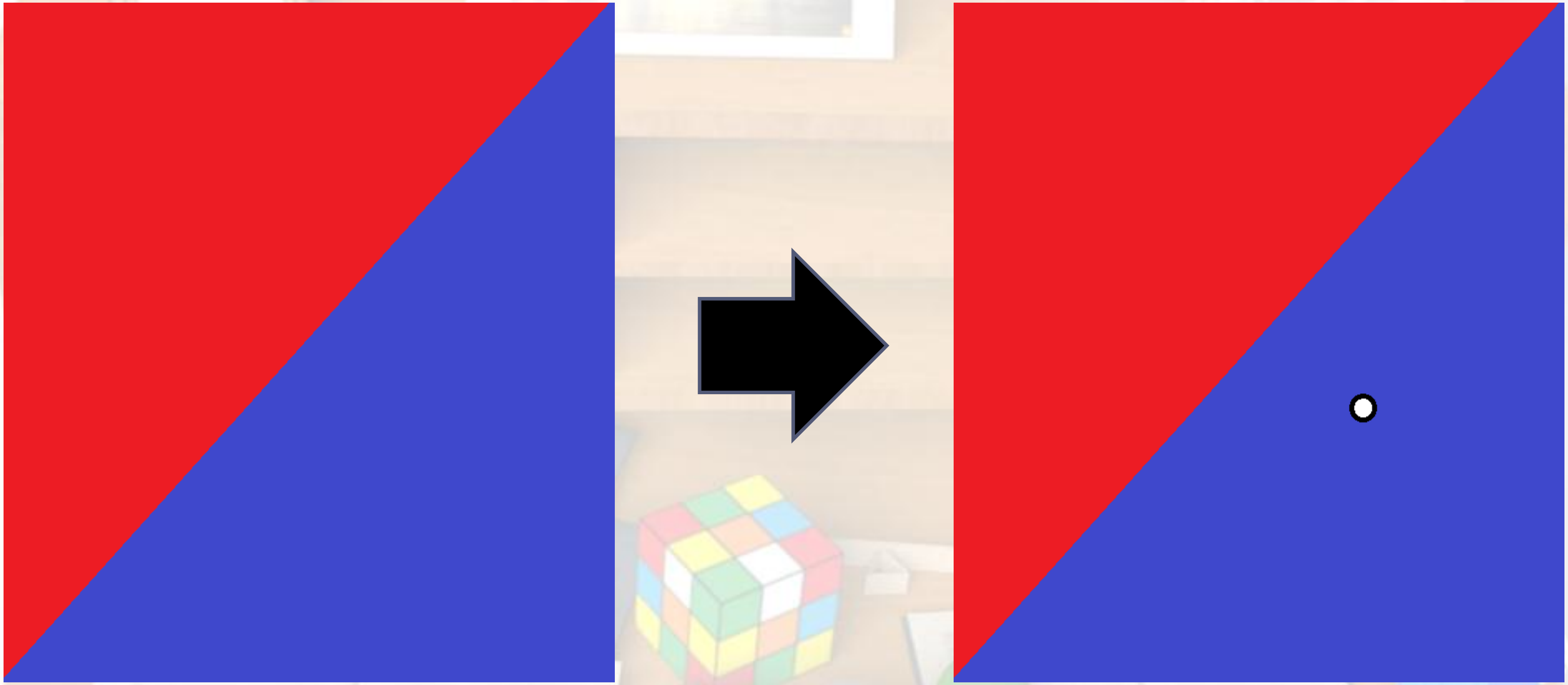
add(1,2) = 3

…



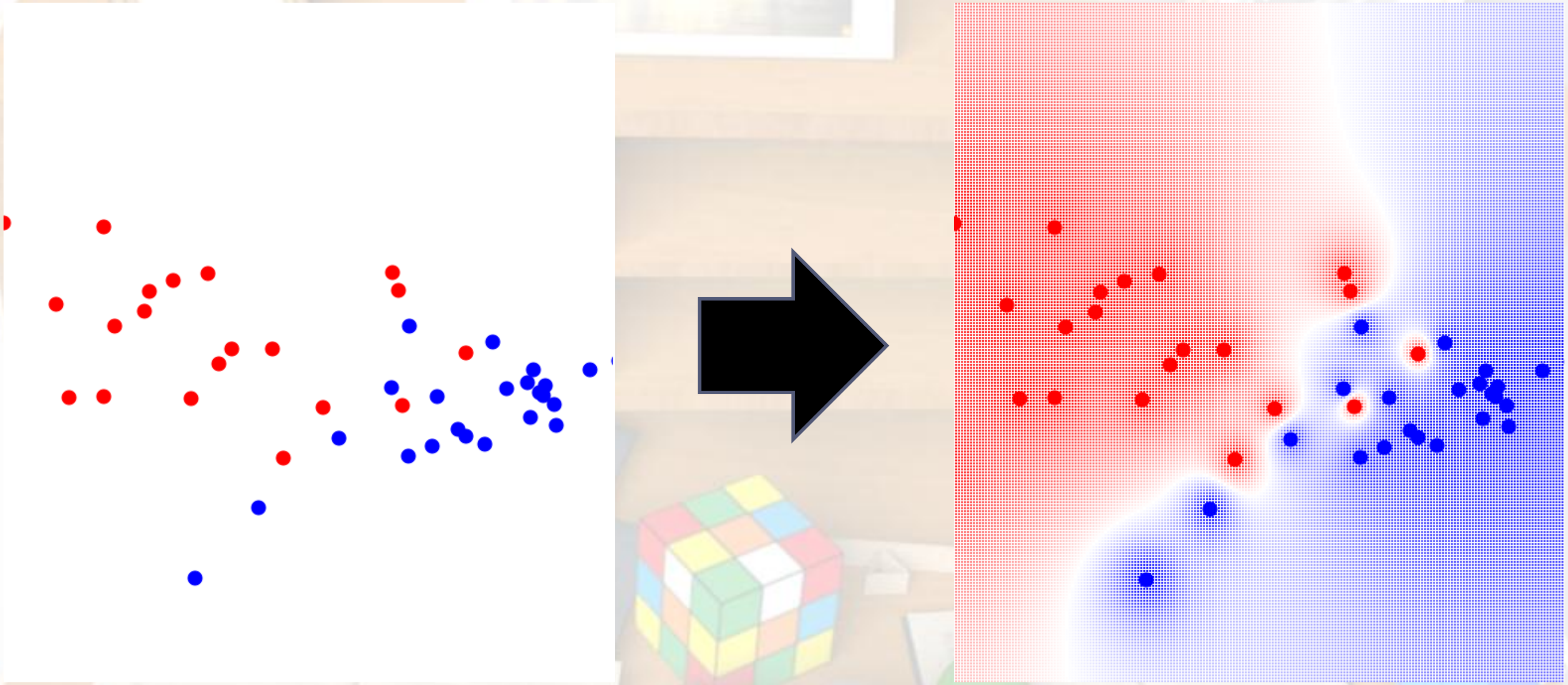
Induction

Deduction



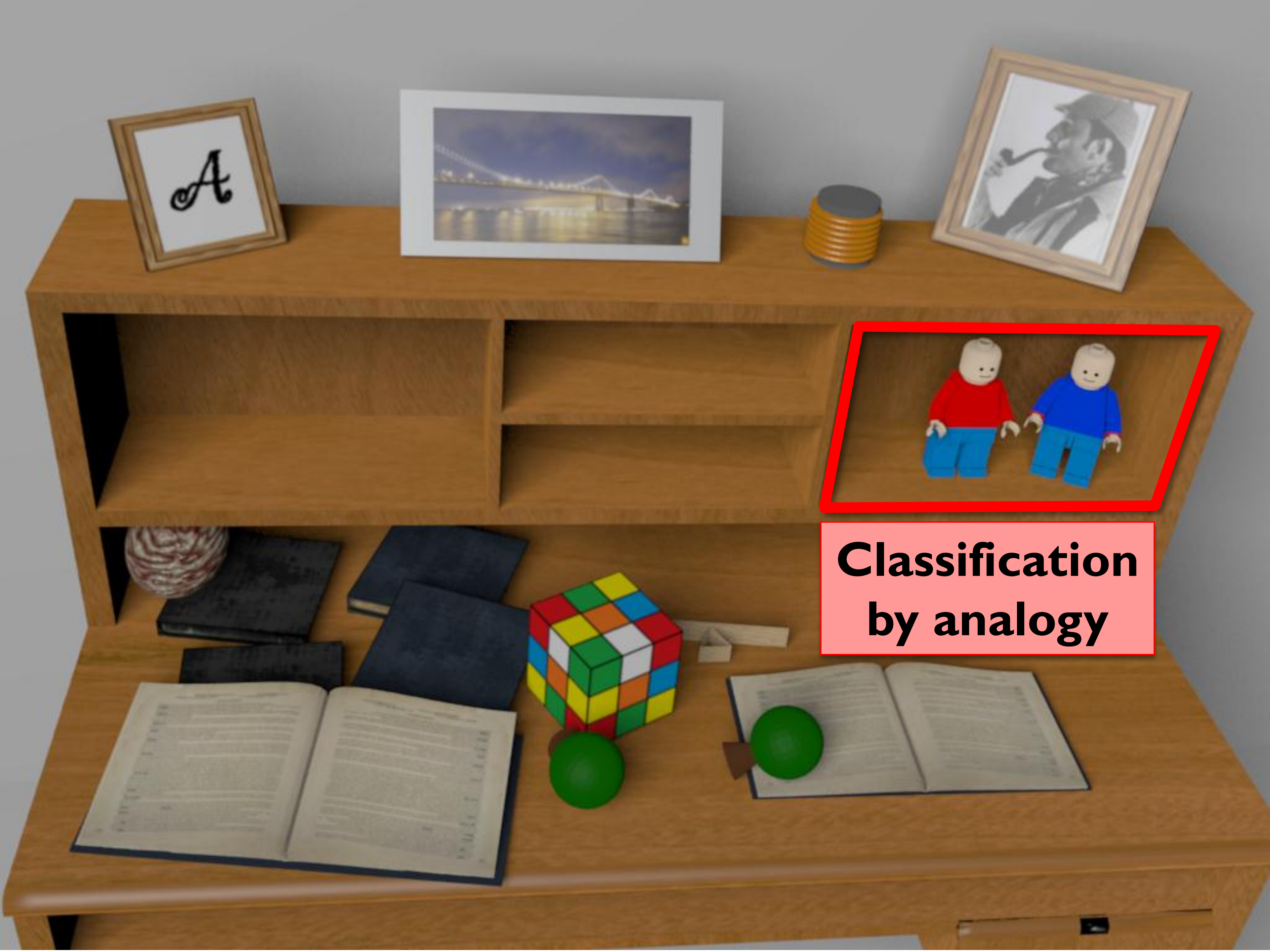
Given a general rule, make a decision in a particular case

Induction



Given particular cases, find a general rule

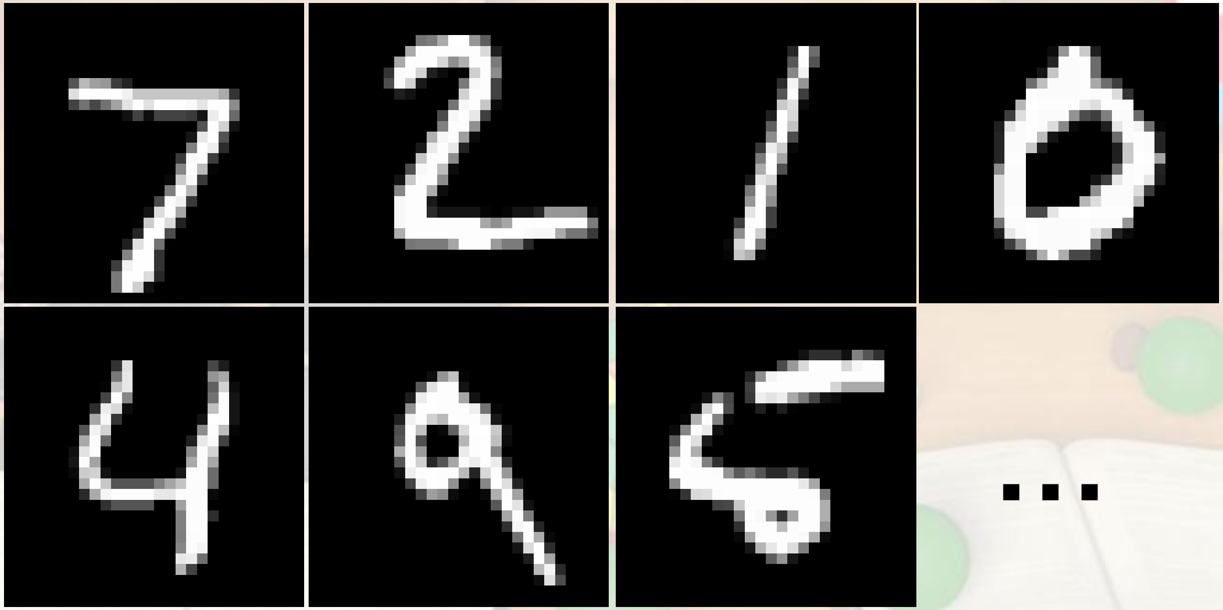
Deduction and Induction



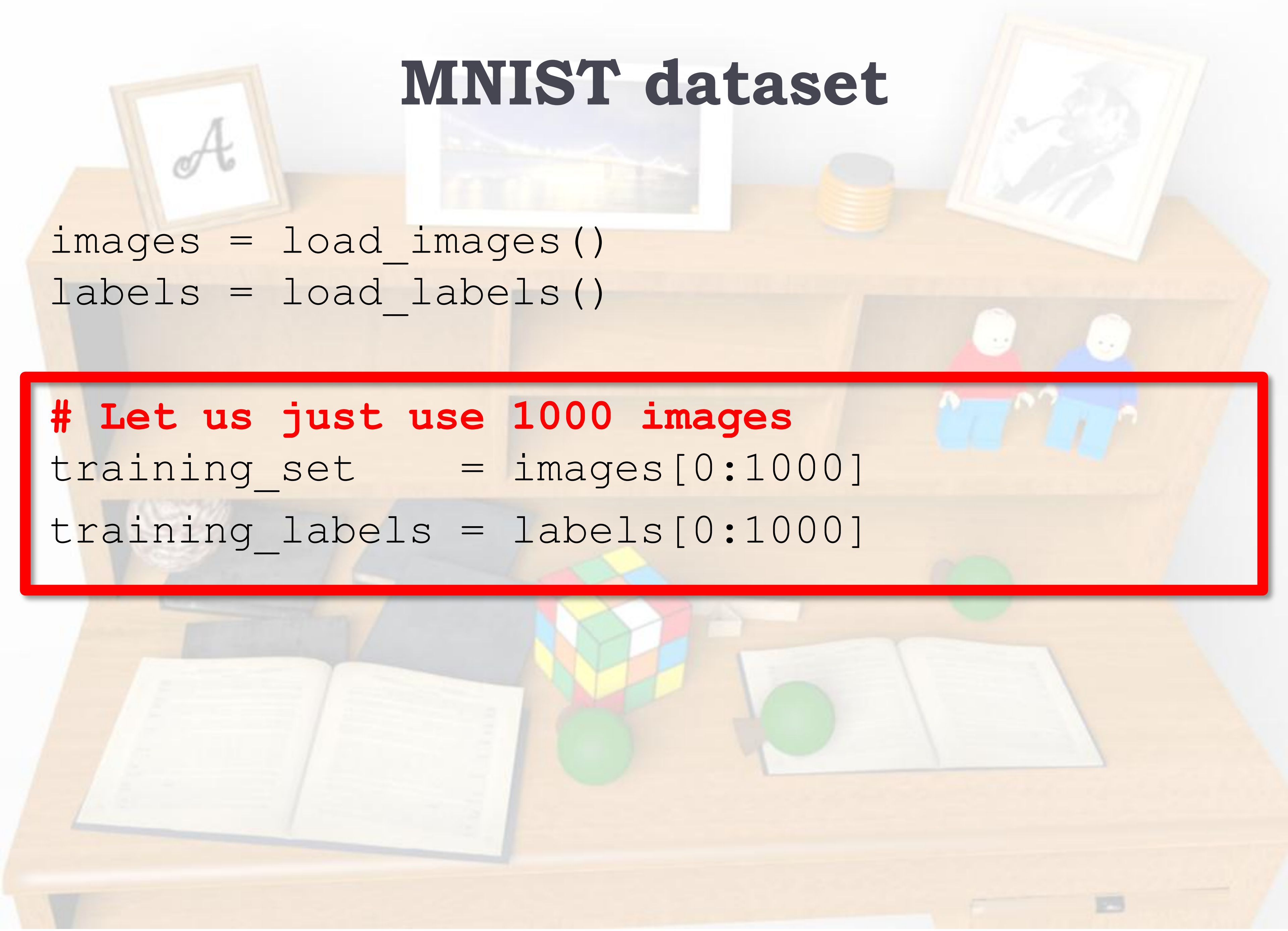
Classification by analogy

MNIST dataset

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



...

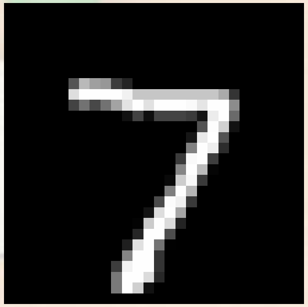


MNIST dataset

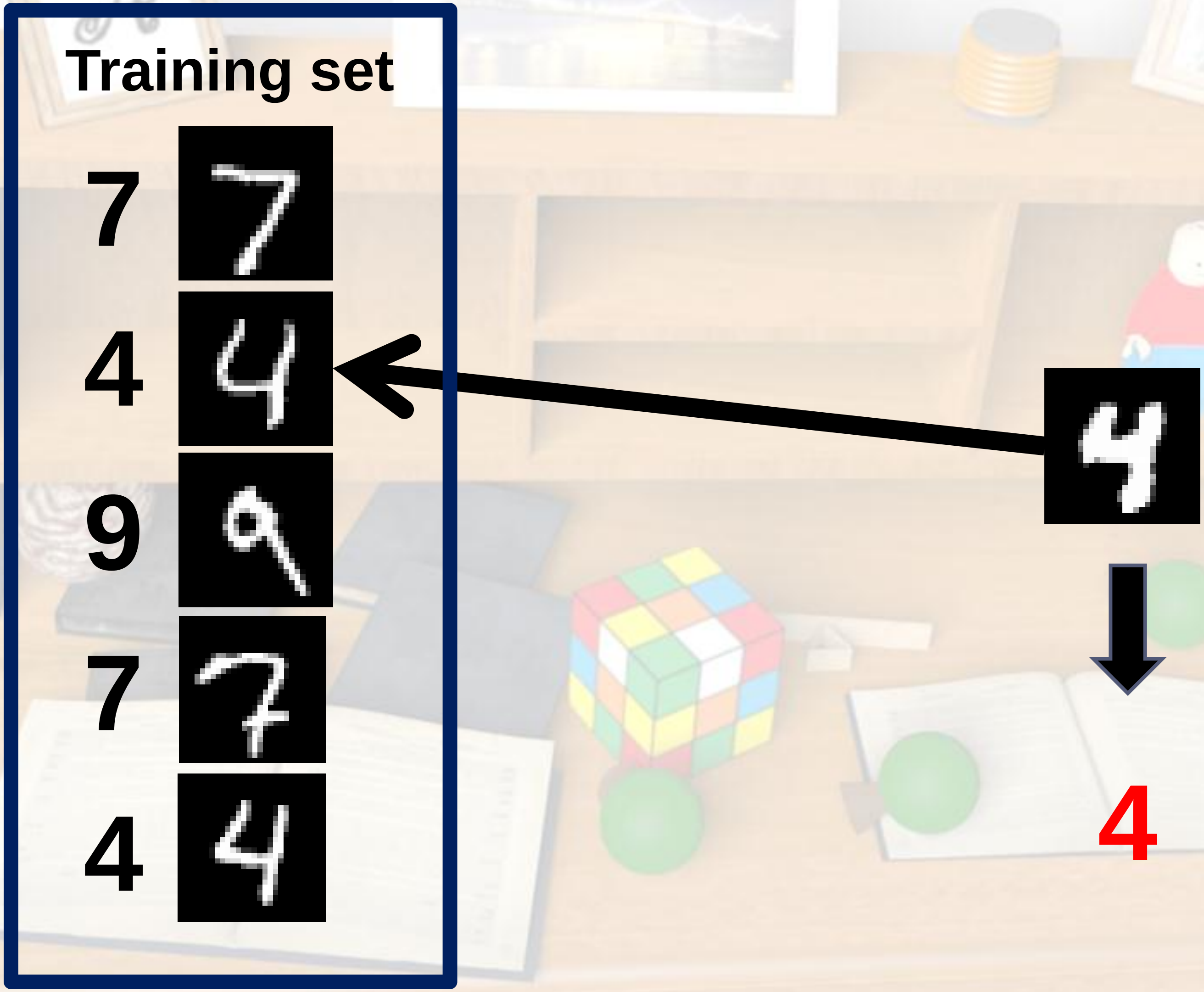
```
images = load_images()
labels = load_labels()

# 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

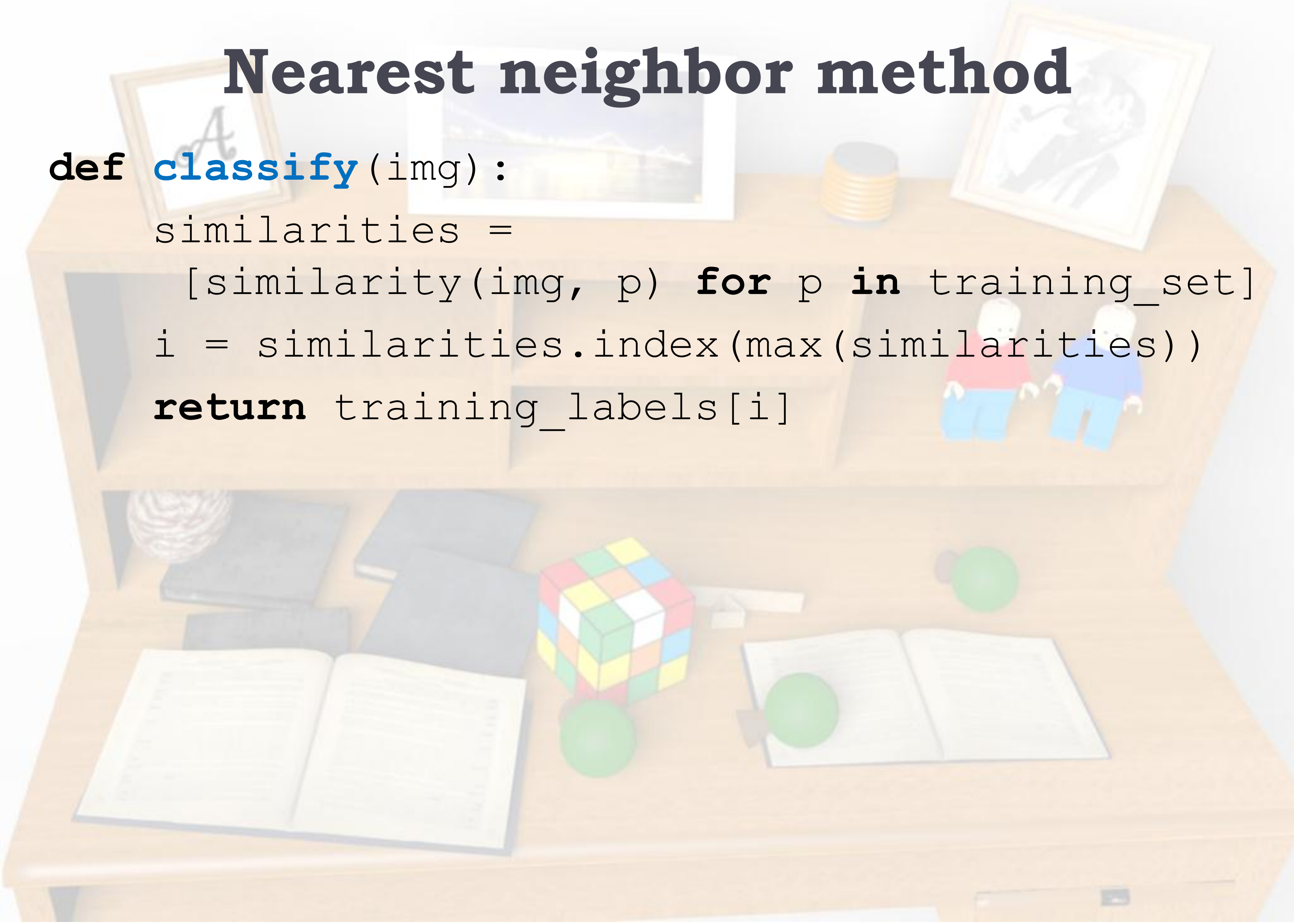
7

4

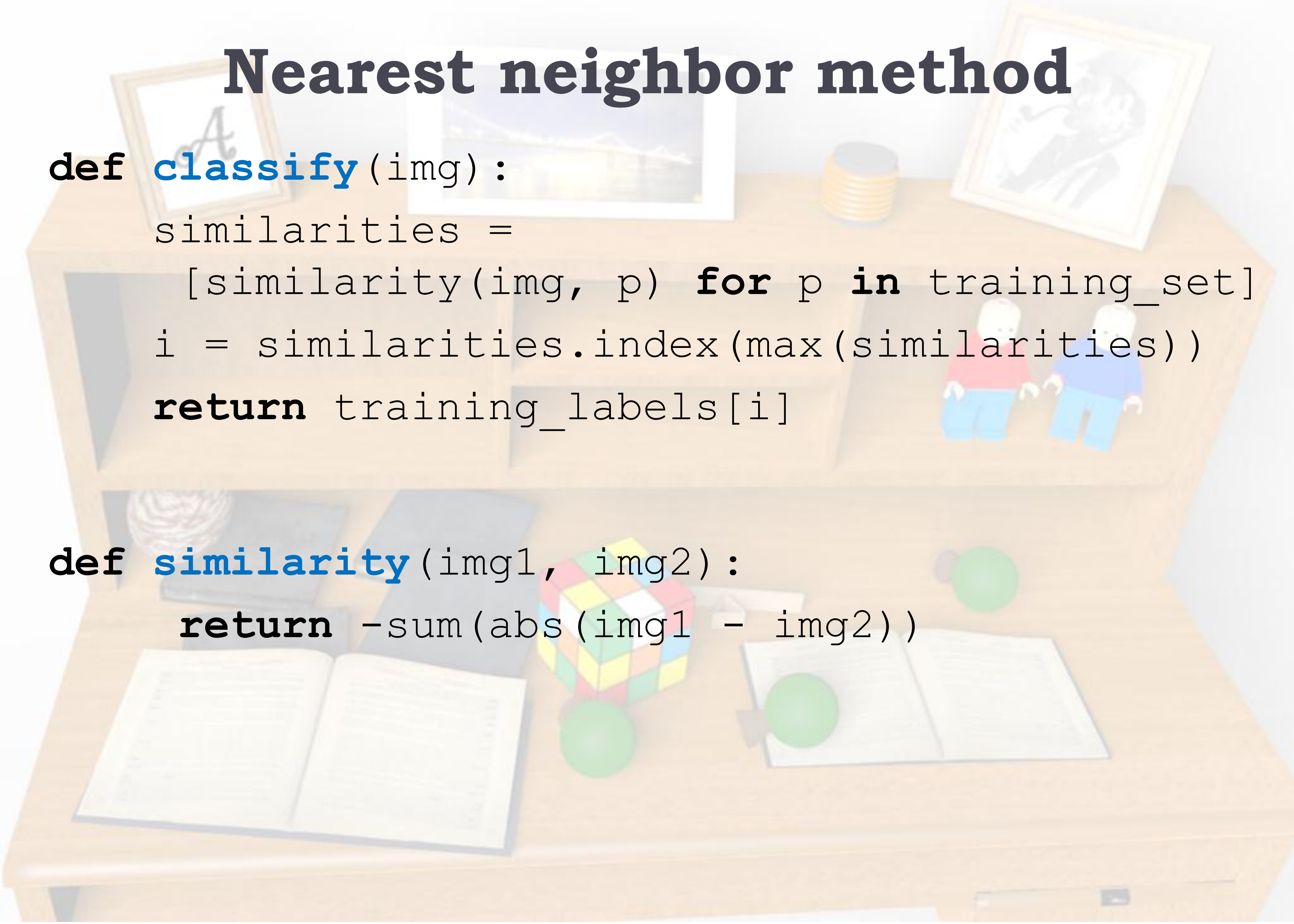
4

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



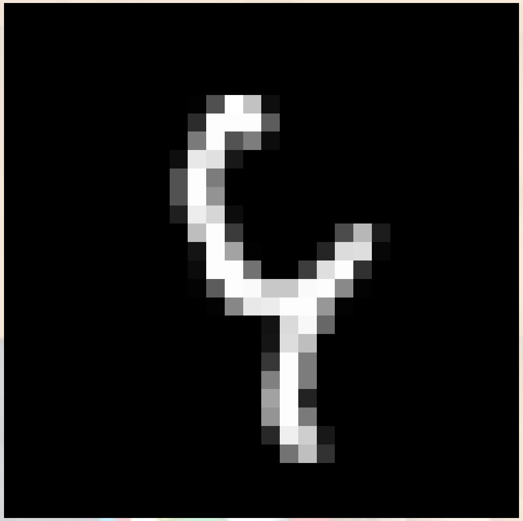
```
def classify(img):
    similarities =
     [similarity(img, p) for p in training_set]
    i = similarities.index(max(similarities))
    return training_labels[i]


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?

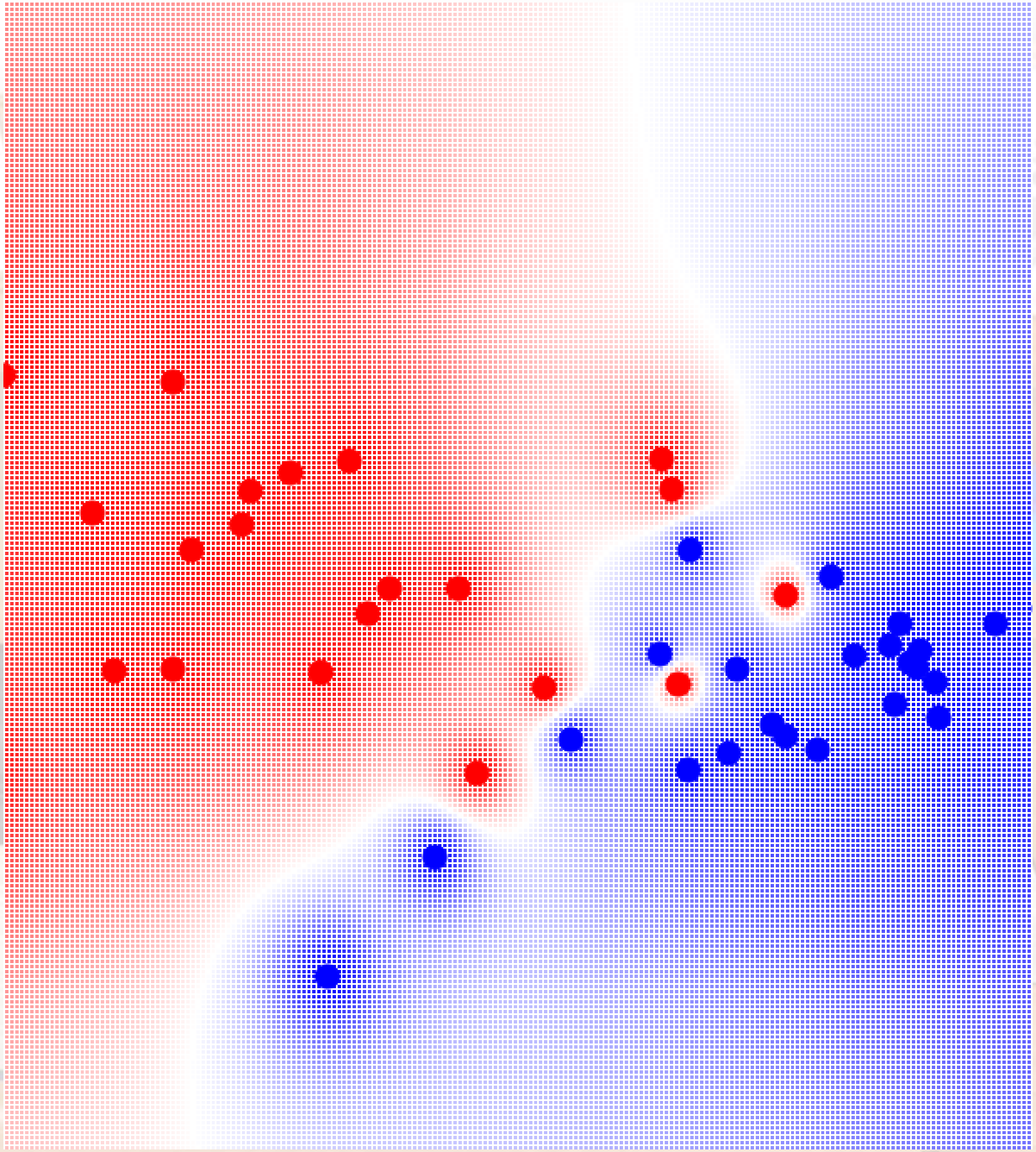


Scikit-learn

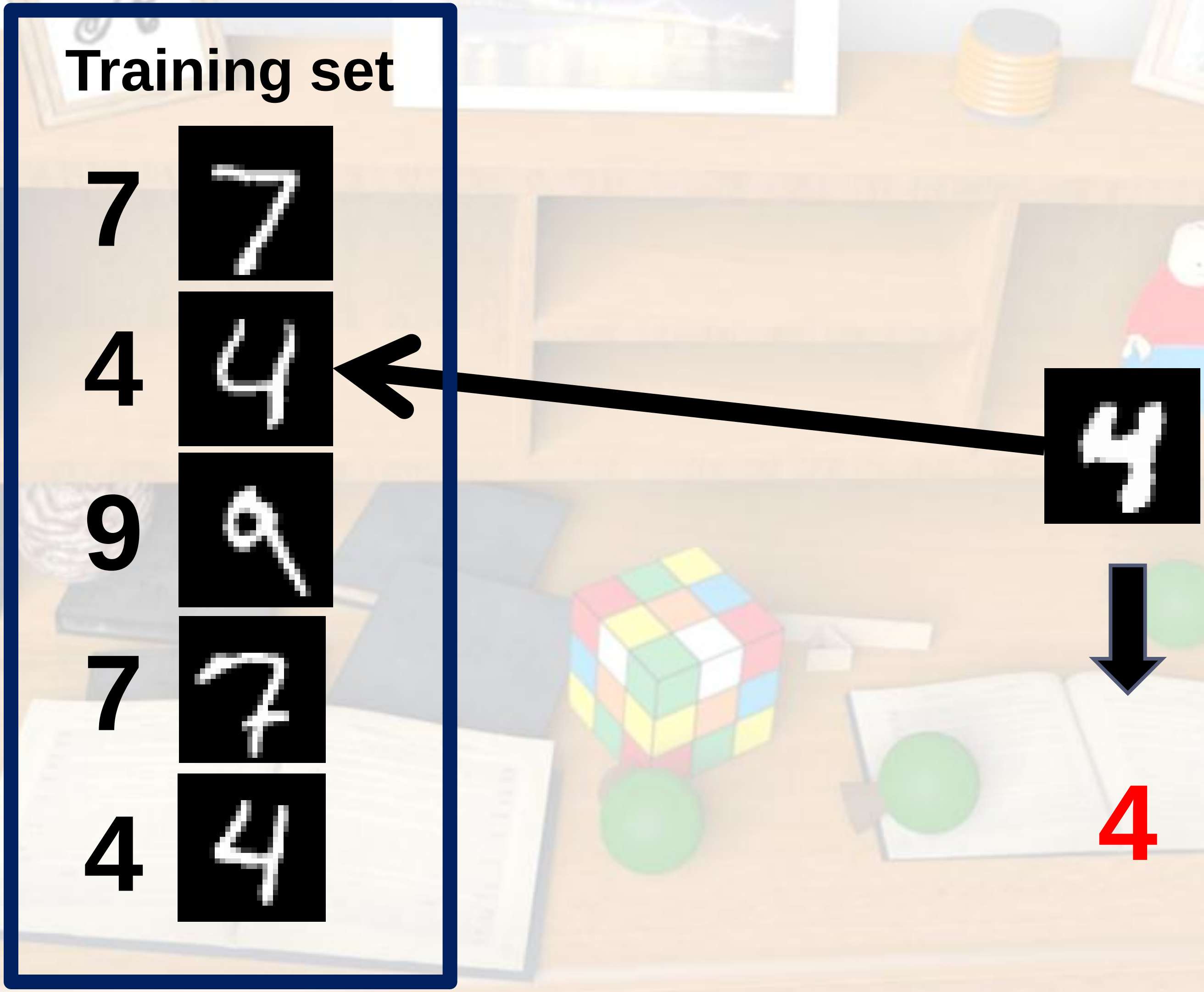
http://scikit-learn.org/

```
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

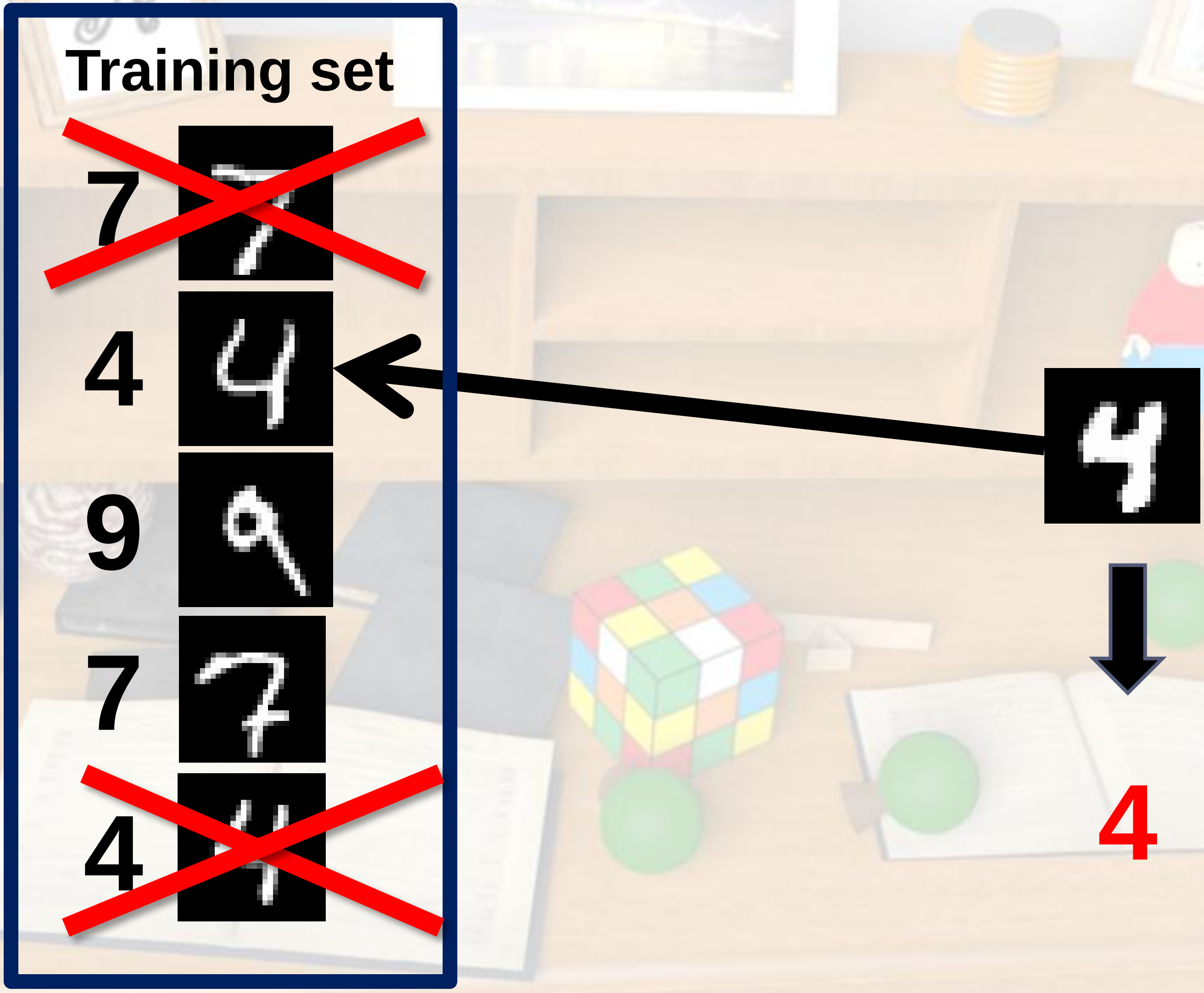
9

7

4

4

Some samples may be thrown away



Training set

7

4

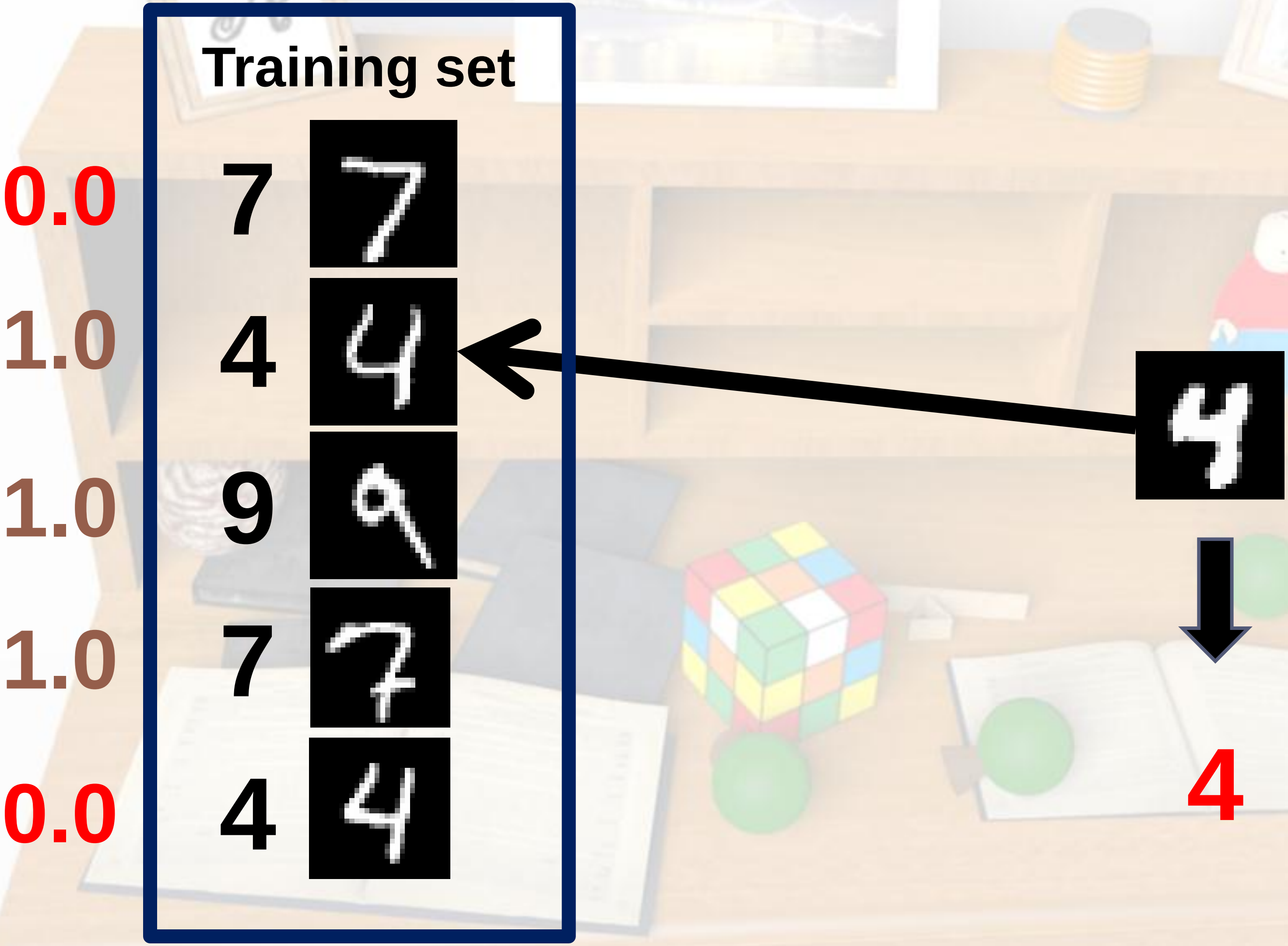
9

7

4

4

…or we can add "weights"

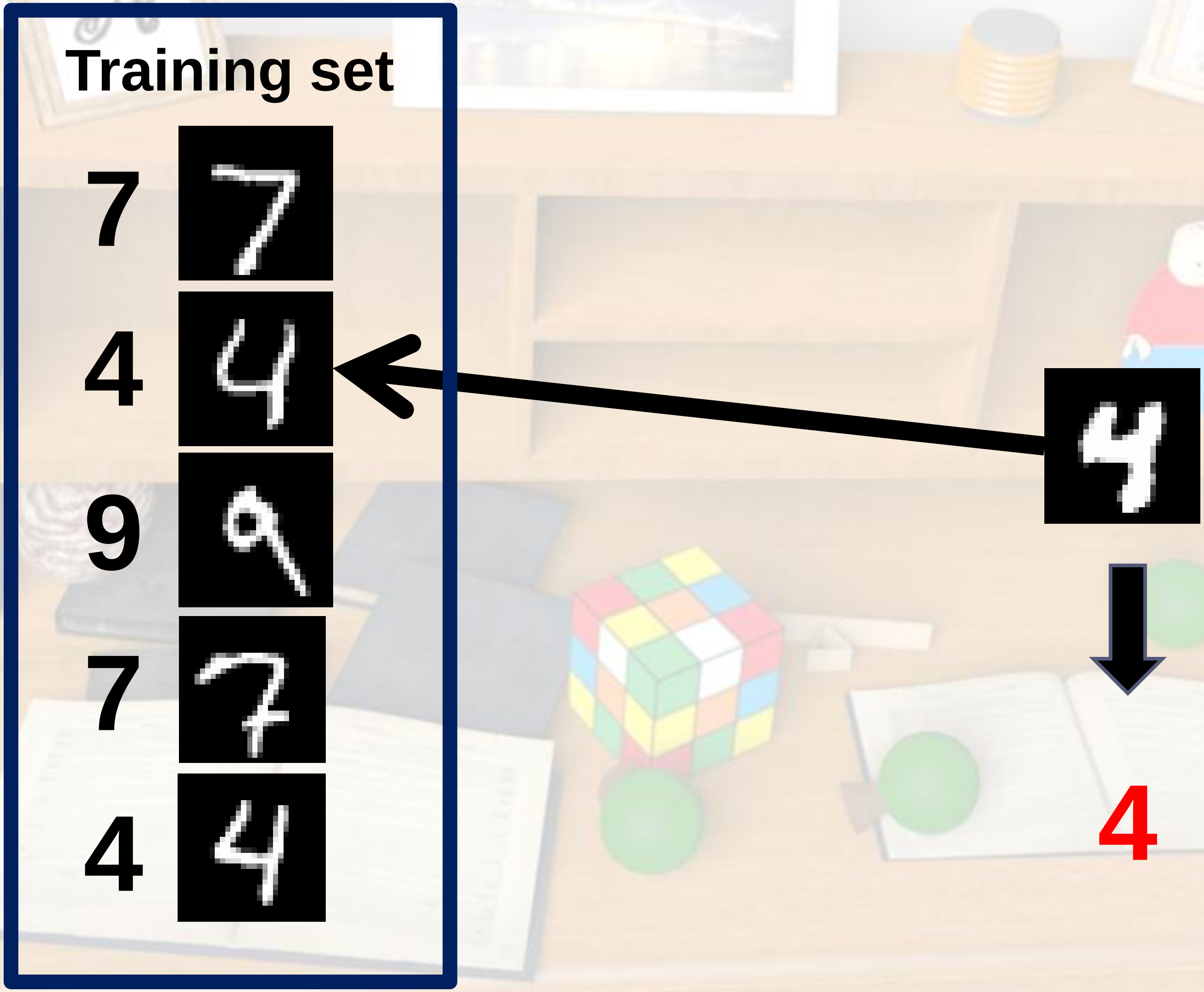


Training set

0.0	7
1.0	4
1.0	9
1.0	7
0.0	4

4

…or we can add "weights"

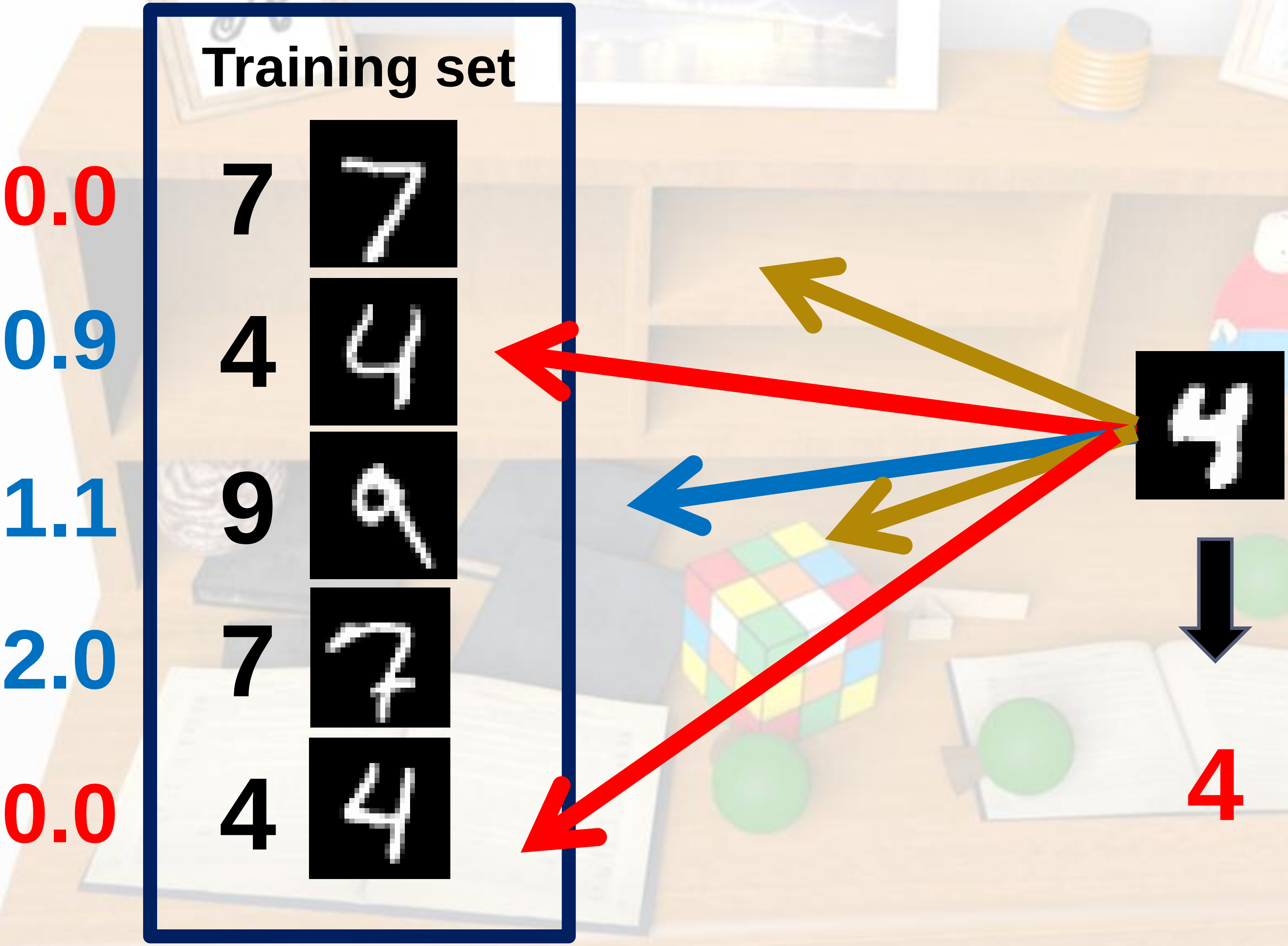


Training set

0.0	7
0.9	4
1.1	9
2.0	7
0.0	4

4

.. or we can SUM instead of MAX



Training set

0.0 7

0.9 4

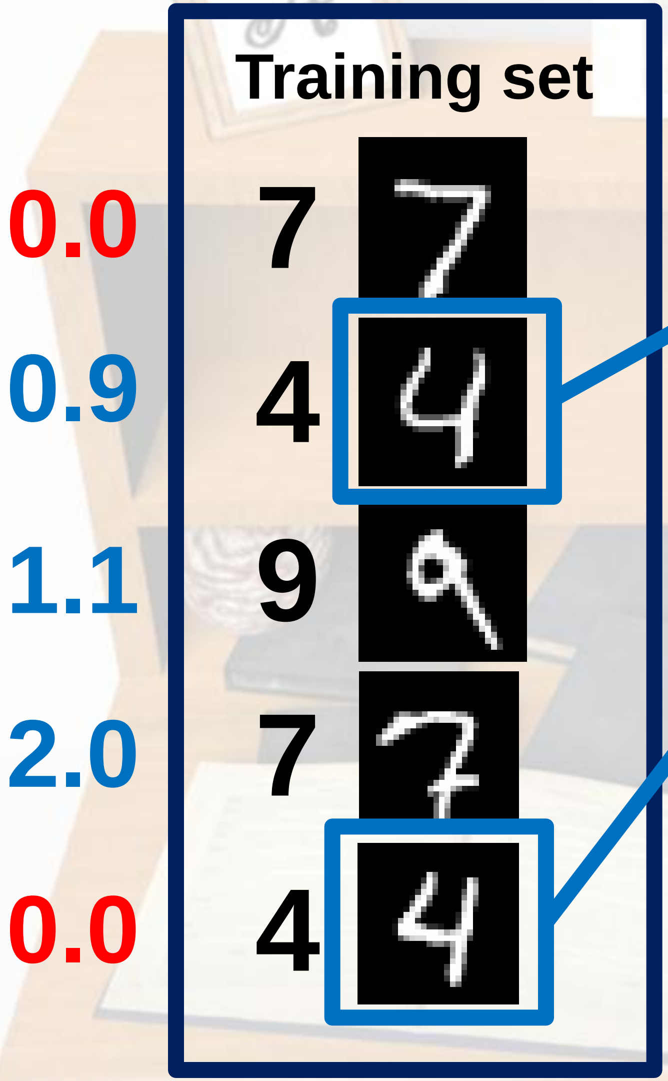
1.1 9

2.0 7

0.0 4

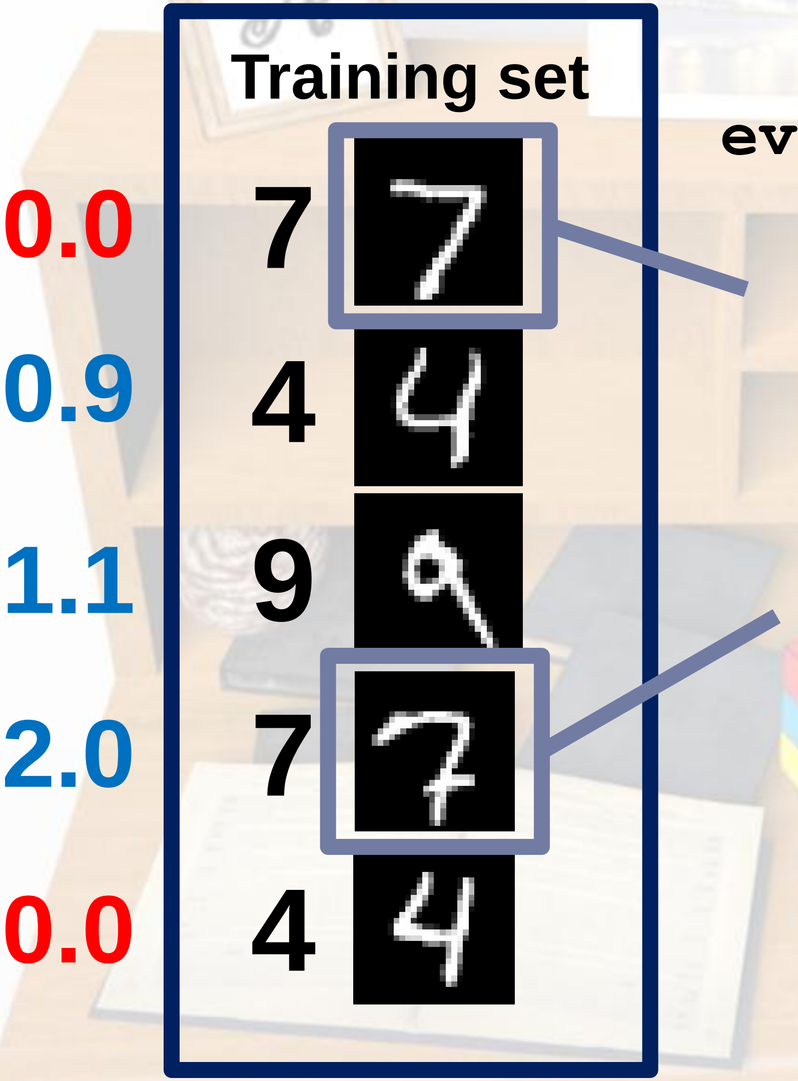
4

.. or we can SUM instead of MAX



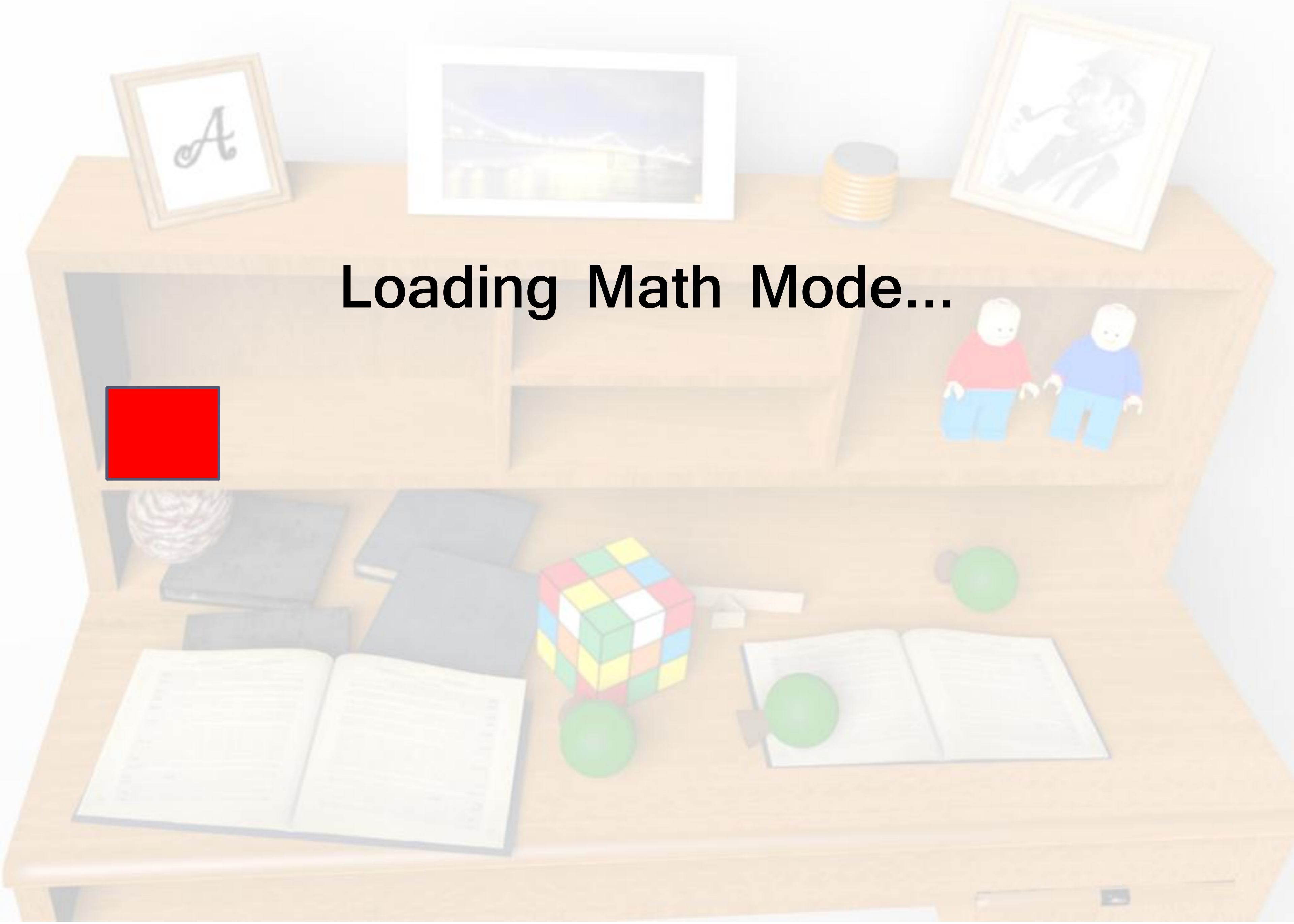
```
evidence(img, 4) =

  0.9 * similarity(img, tr[1])

  +

  0.0 * similarity(img, tr[4])

  = 34.0
```

.. or we can SUM instead of MAX

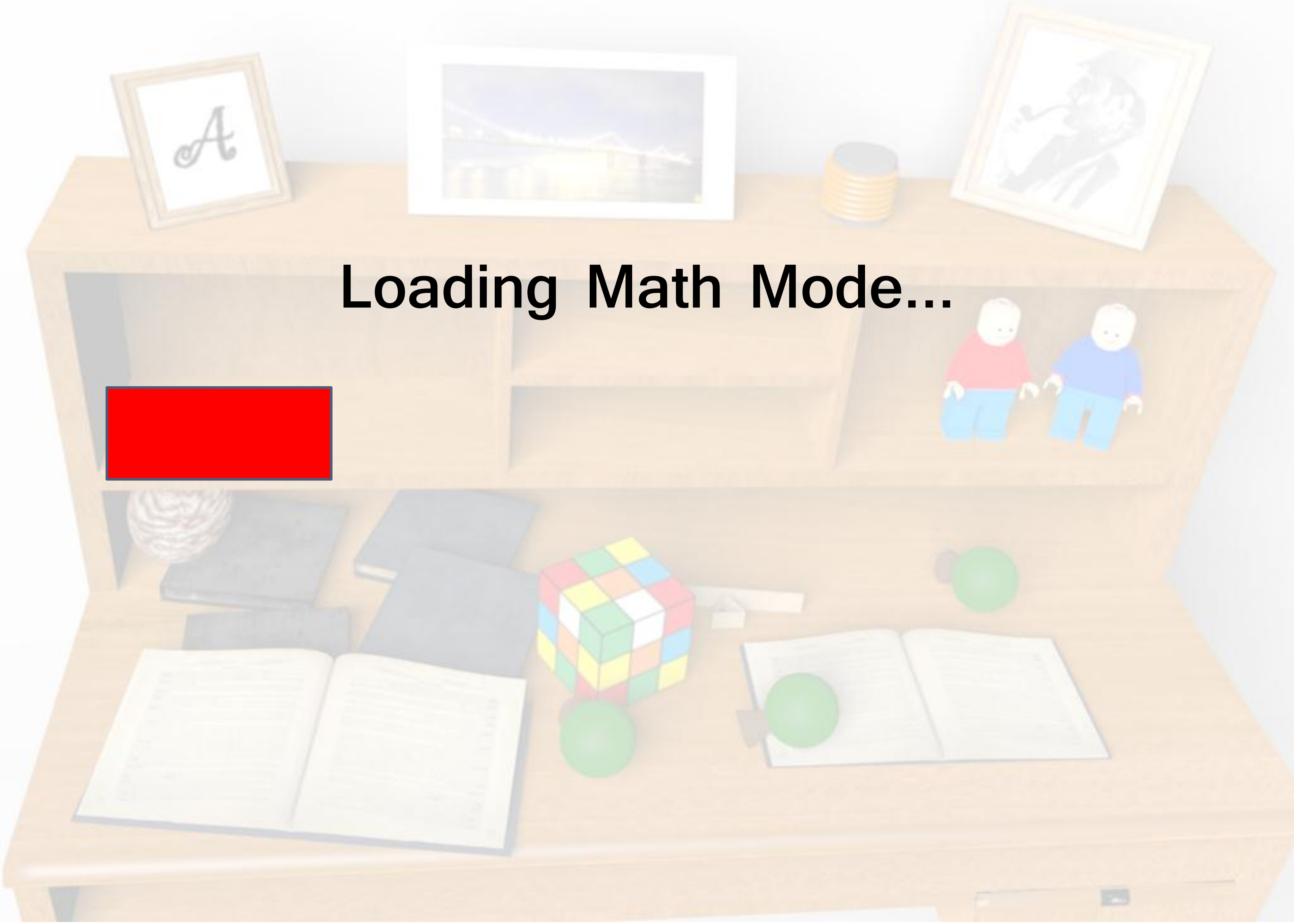


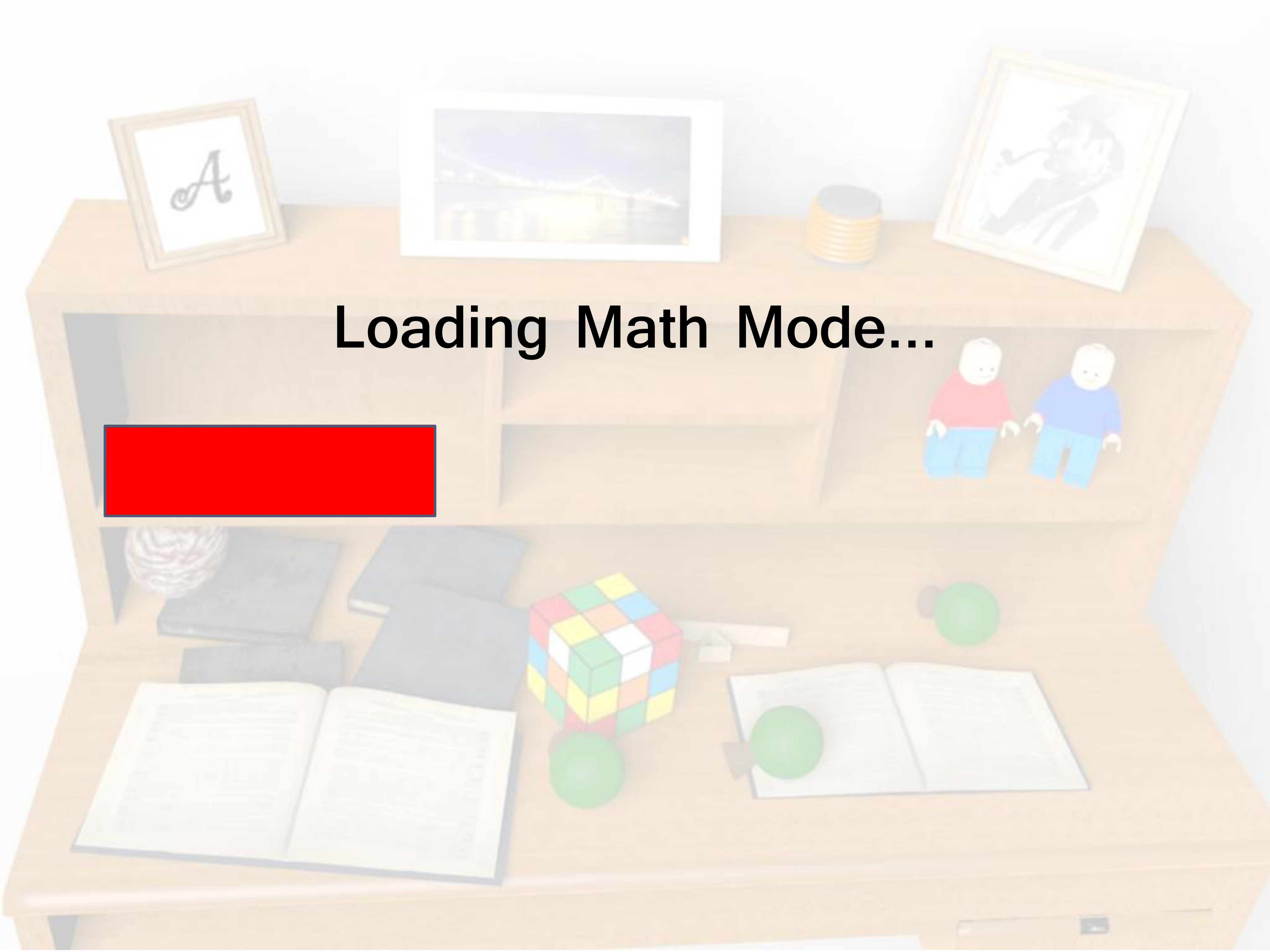
```
evidence(img, 7) =

  0.0 * similarity(img, tr[0])

  +

  2.0 * similarity(img, tr[3])

  = 20.0
```

Loading Math Mode...

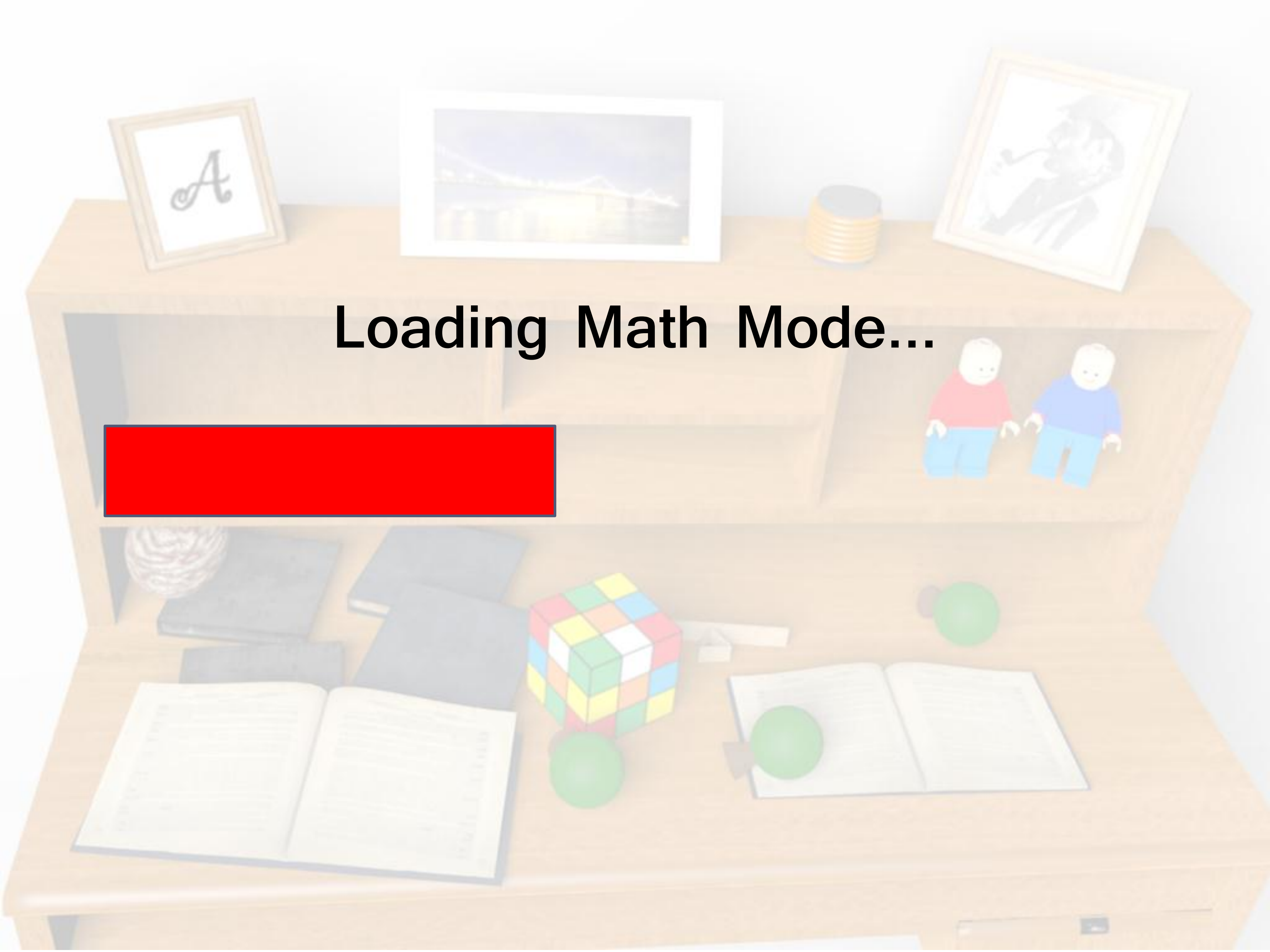


Loading Math Mode...

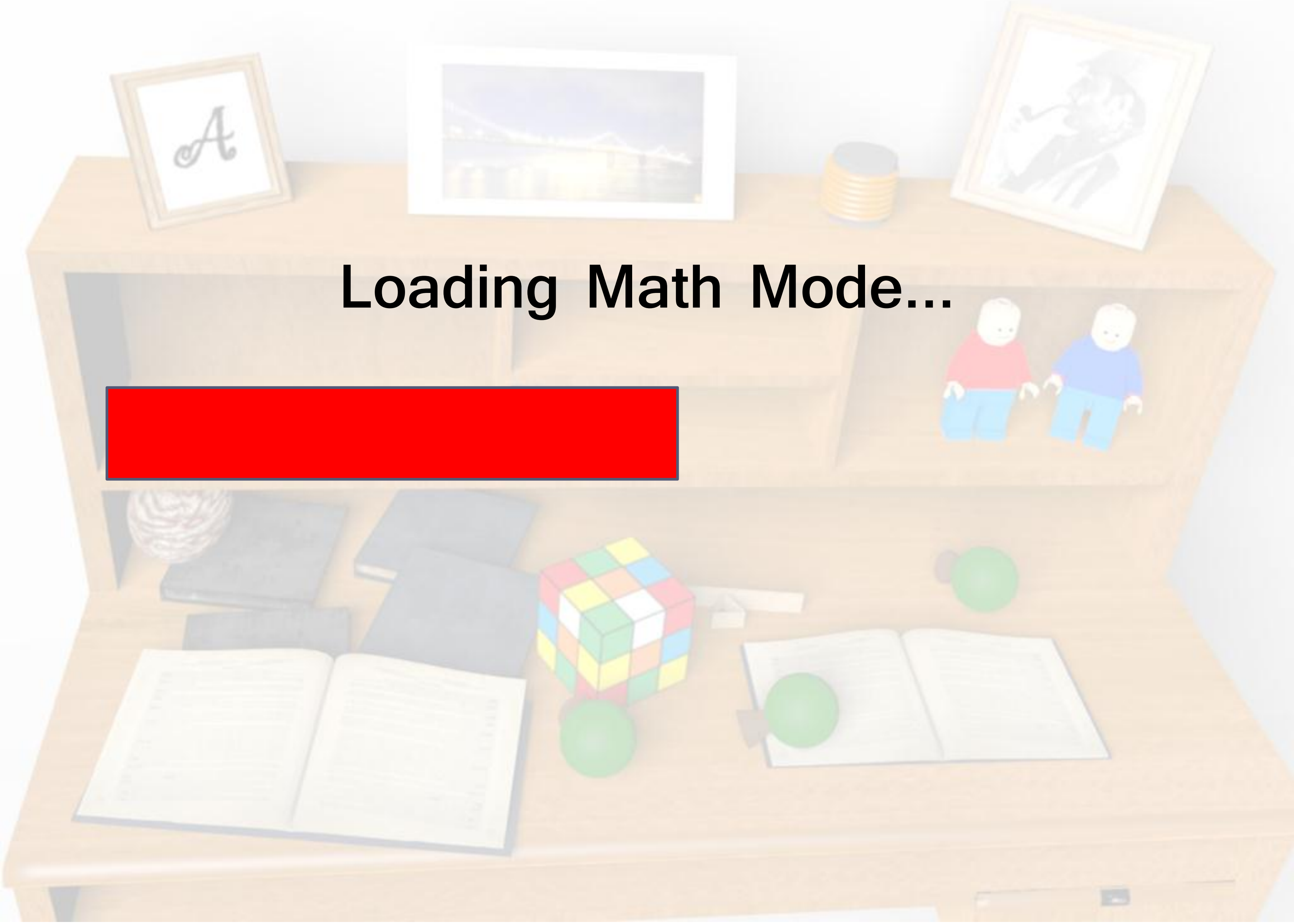




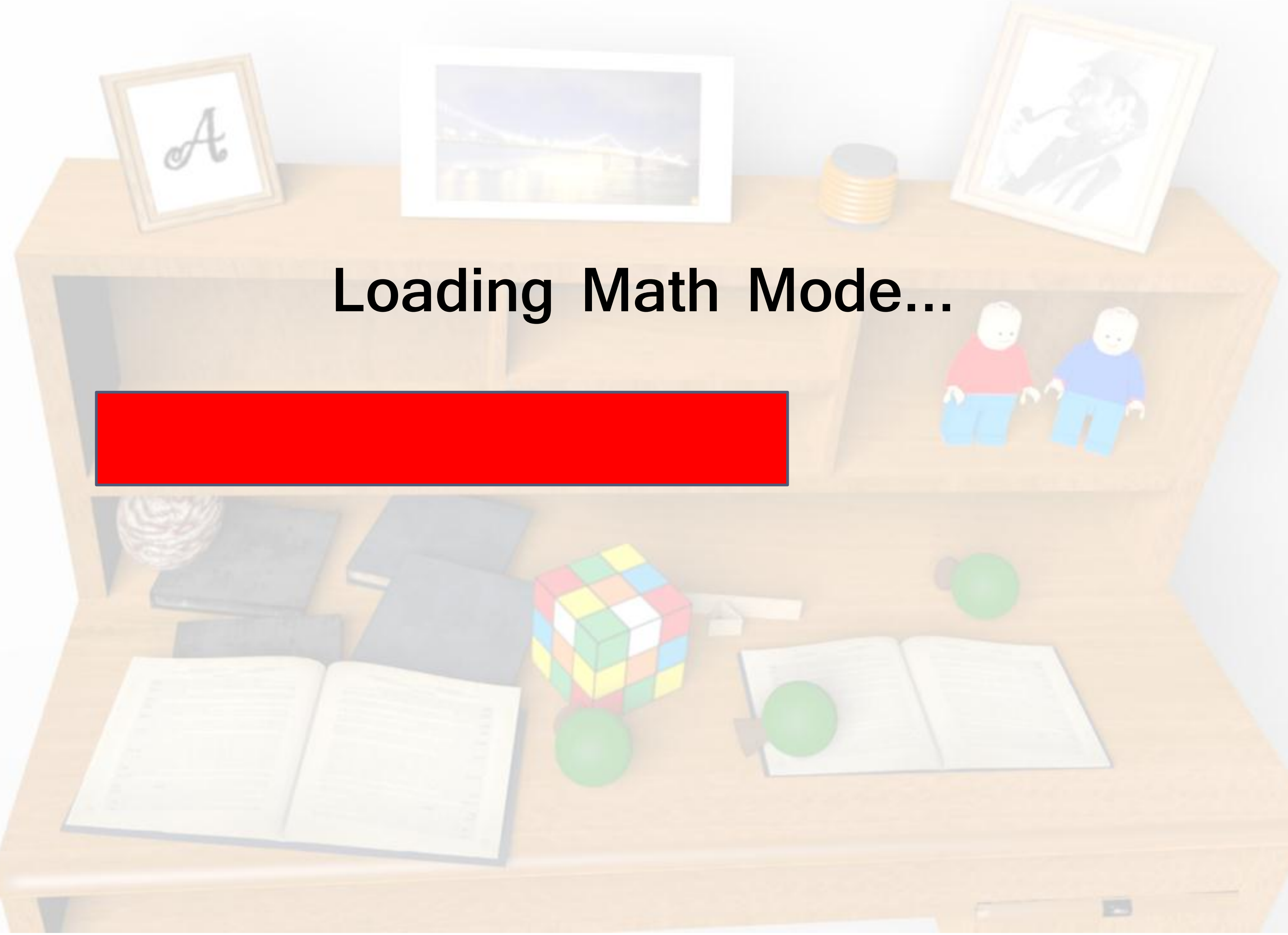
Loading Math Mode...



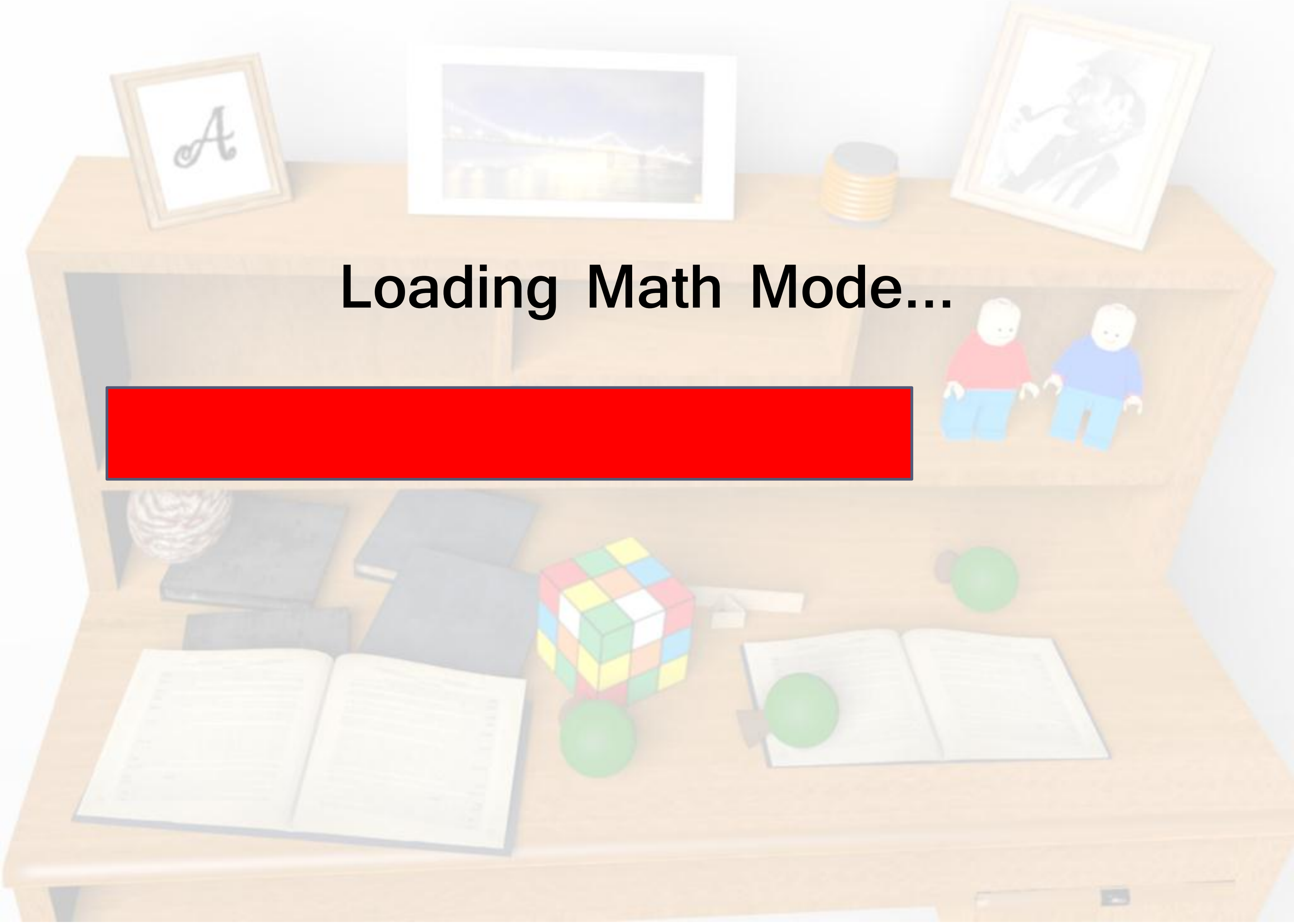
Loading Math Mode...



Loading Math Mode...

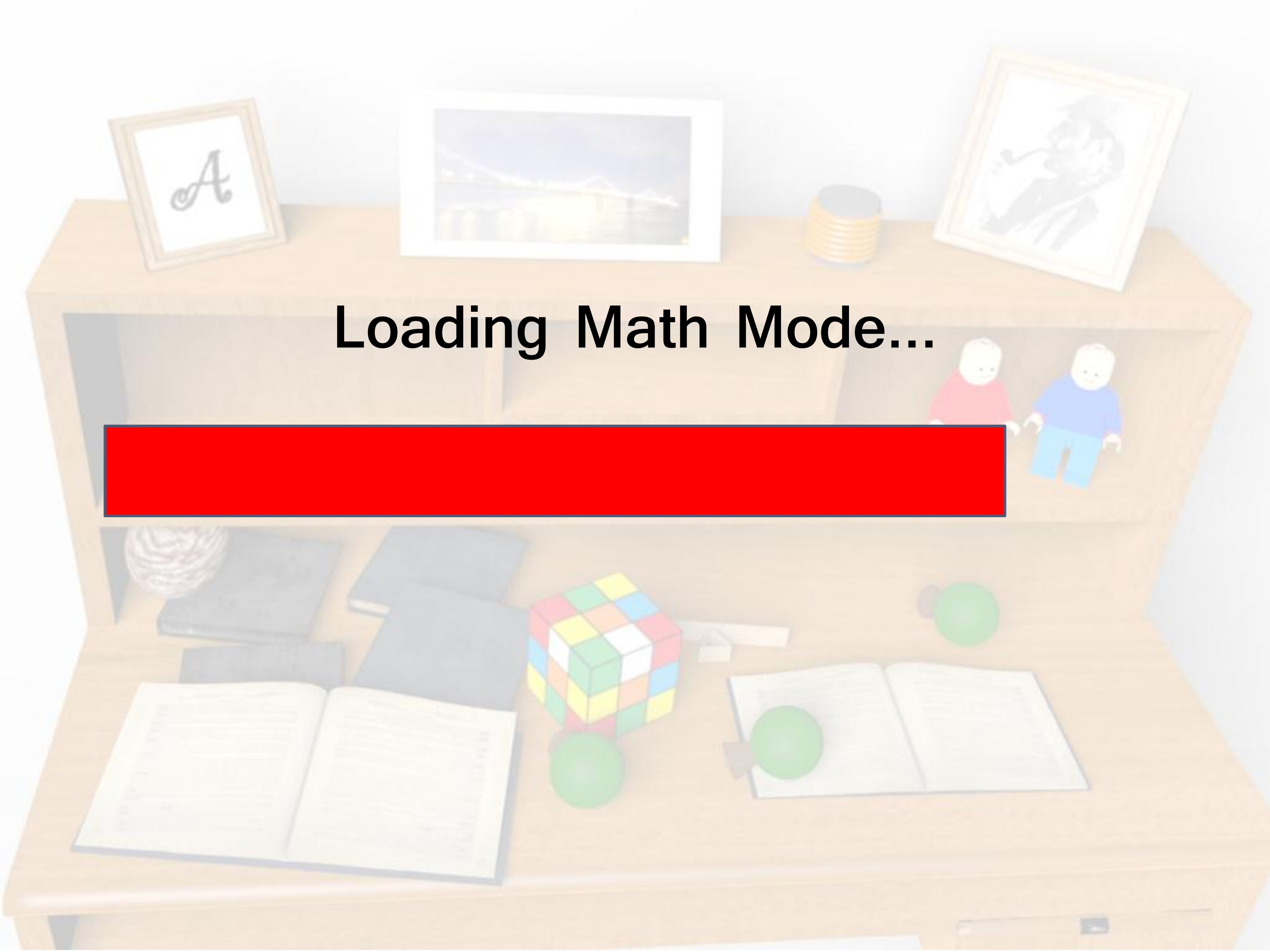


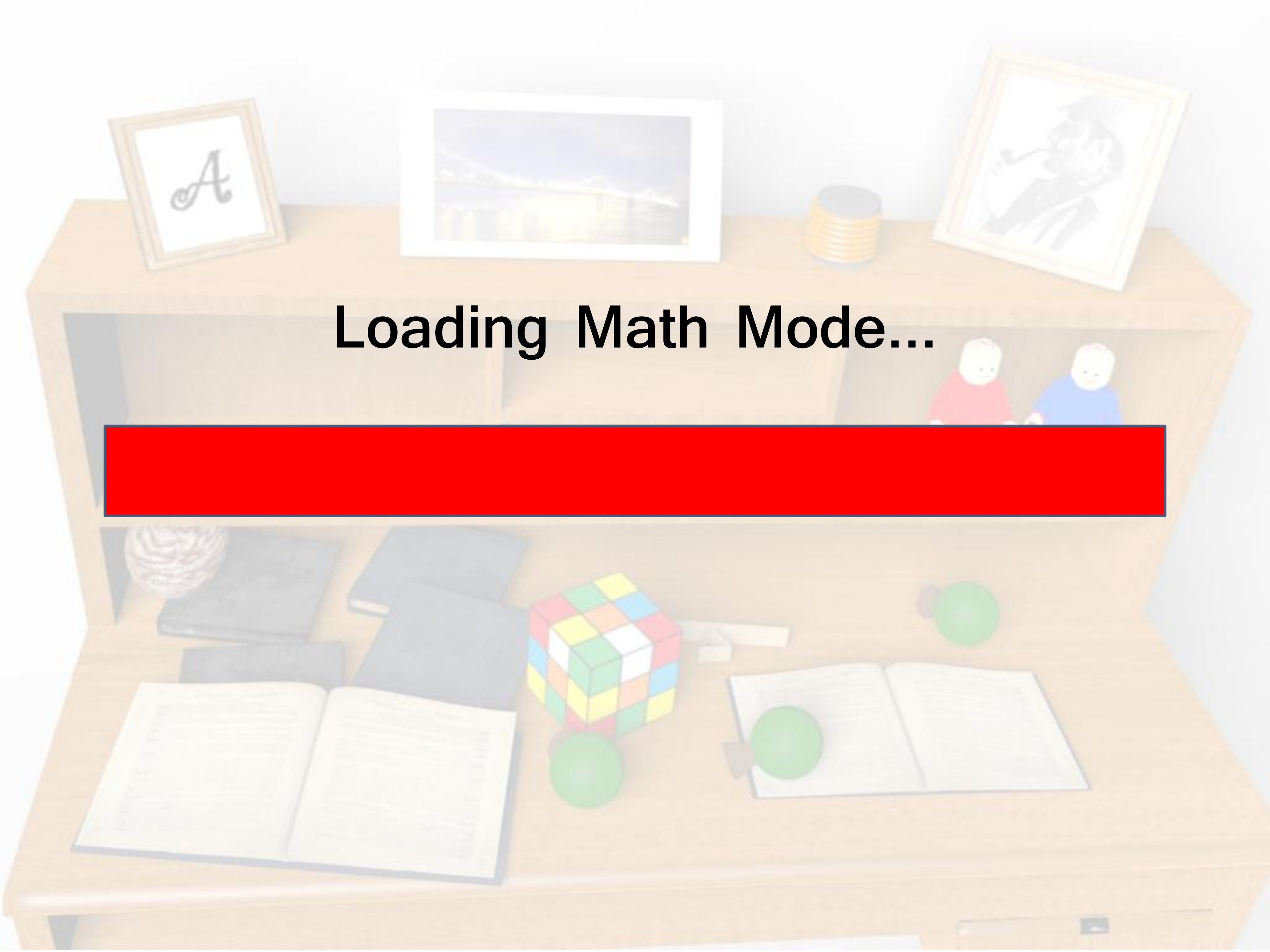
Loading Math Mode...



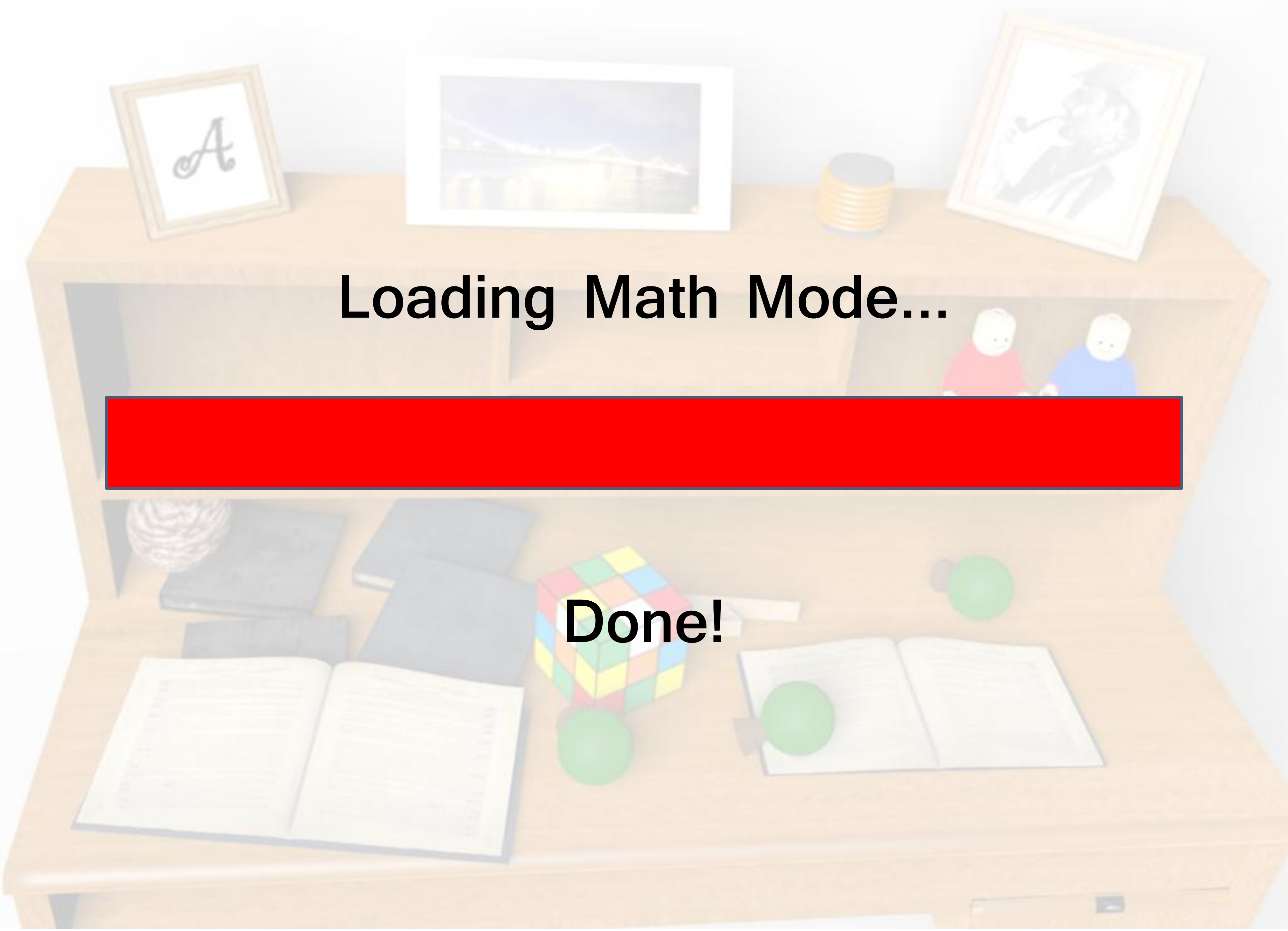
Loading Math Mode...

Loading Math Mode...





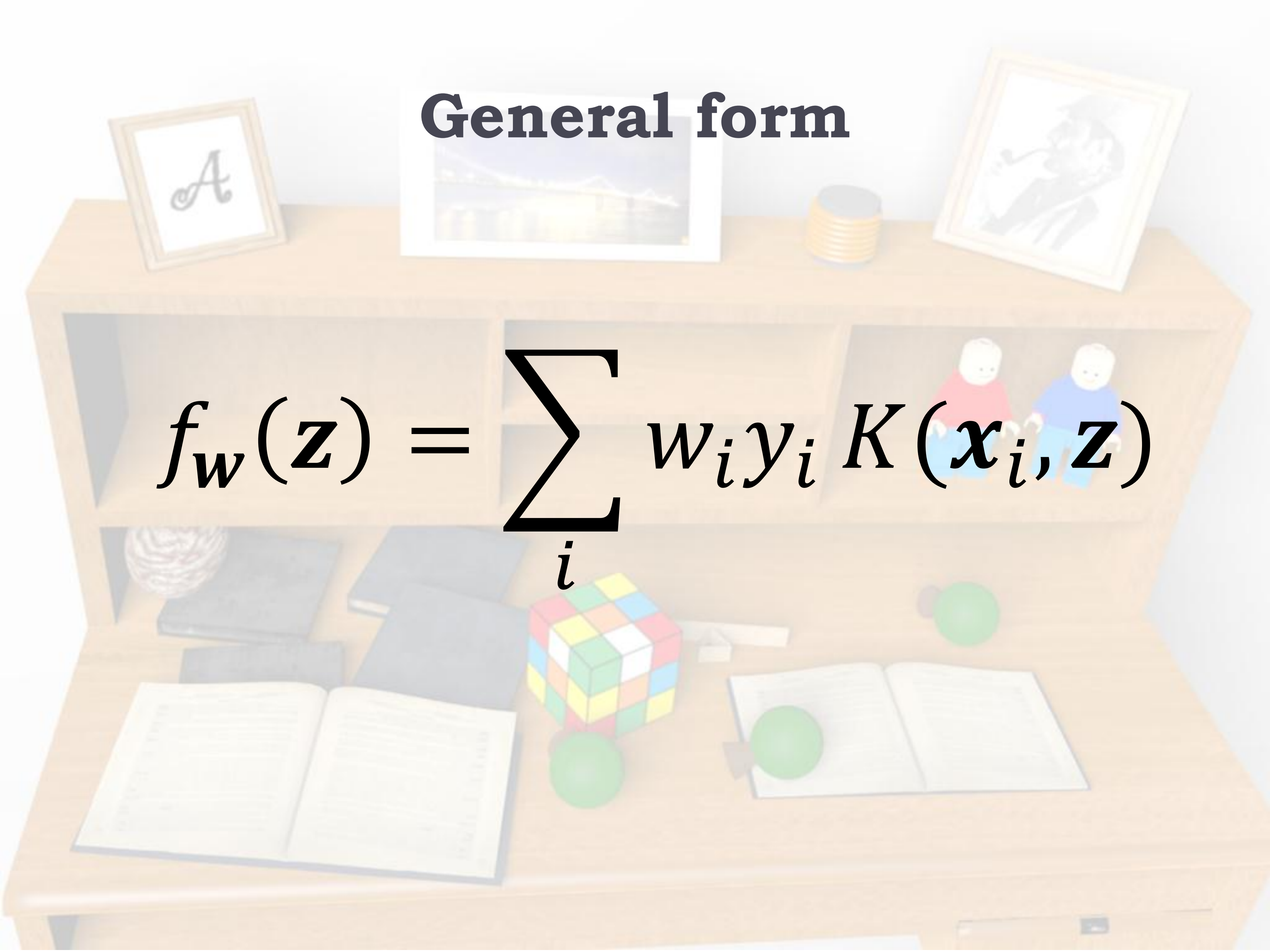
Loading Math Mode...



Loading Math Mode...

Done!

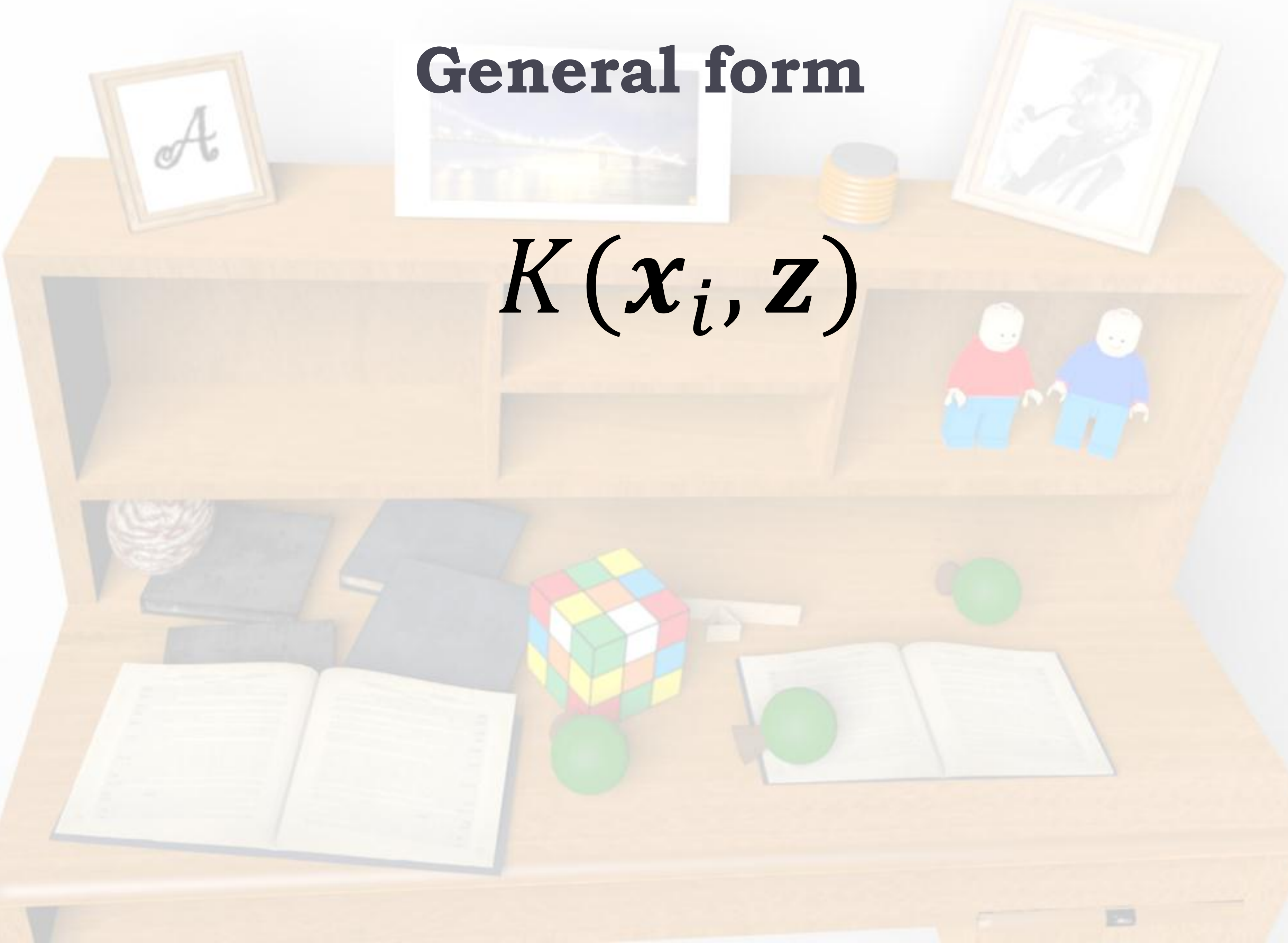
General form



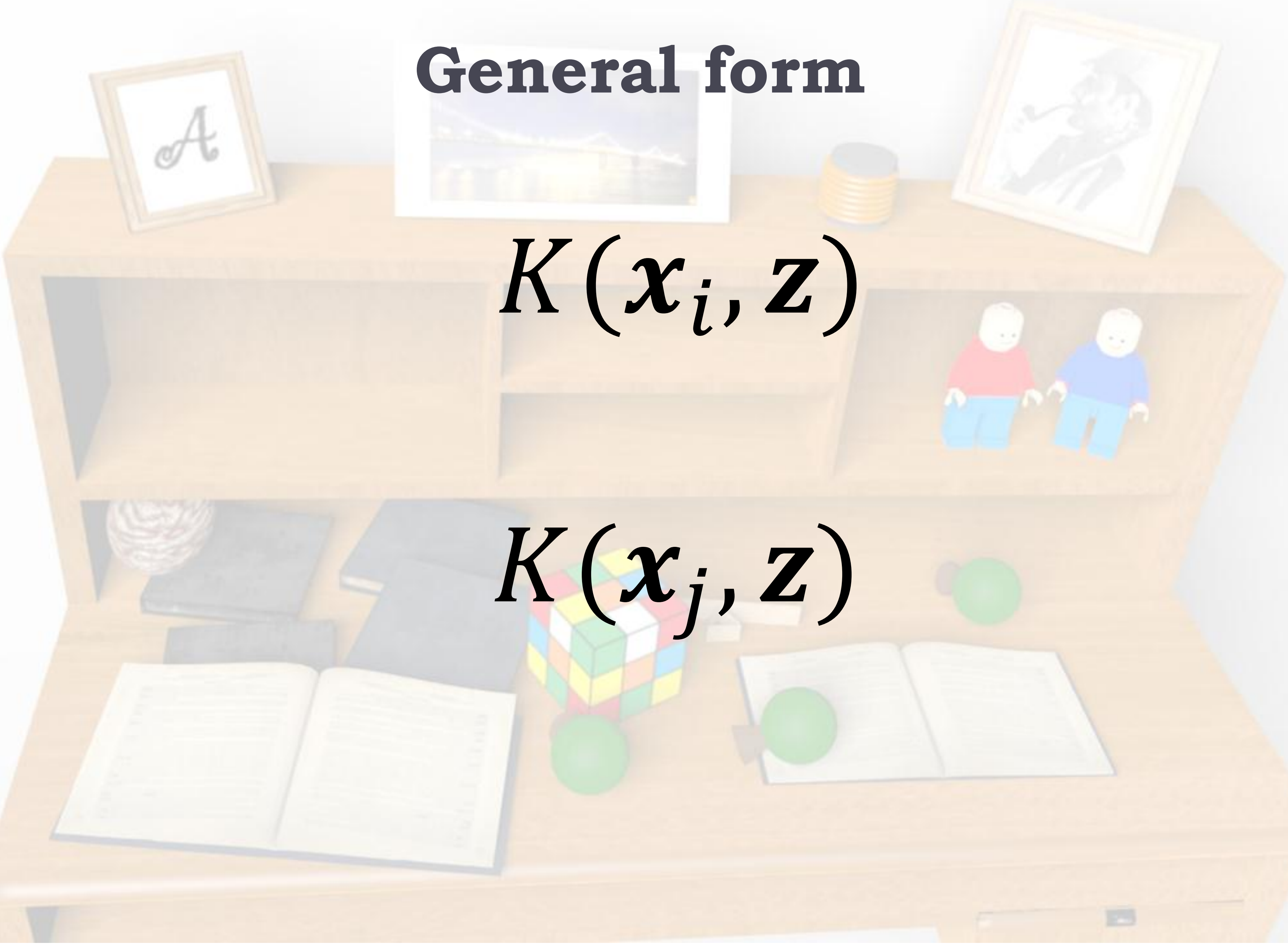
$$f_{\boldsymbol{w}}(\boldsymbol{z}) = \sum_{i} w_i y_i K(\boldsymbol{x}_i, \boldsymbol{z})$$

General form

$$K(\boldsymbol{x}_i, \boldsymbol{z})$$

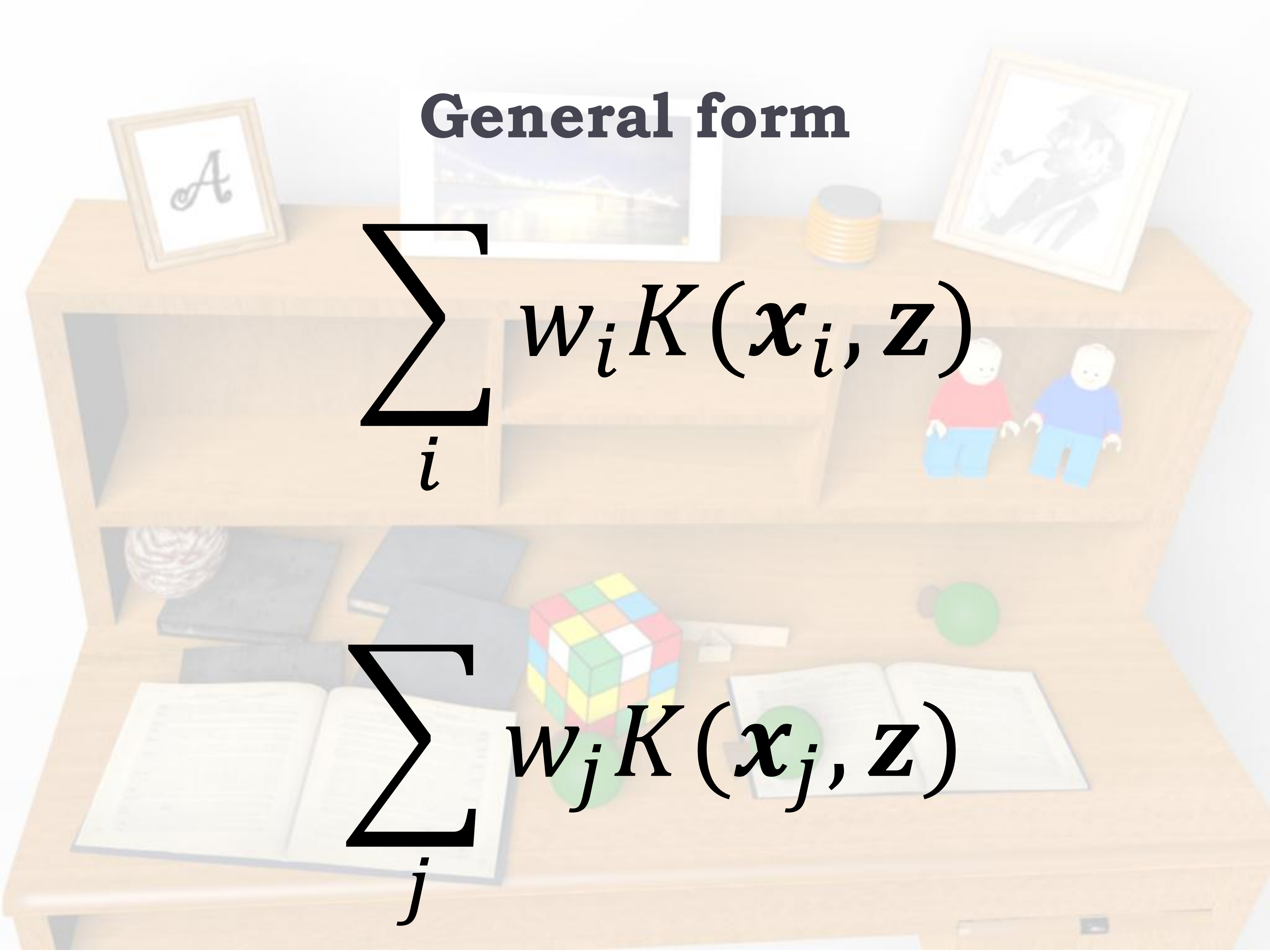


General form



$$K(\boldsymbol{x}_i, \boldsymbol{z})$$

$$K(\boldsymbol{x}_j, \boldsymbol{z})$$



General form

$$\sum_i w_i K(\boldsymbol{x}_i, \boldsymbol{z})$$

$$\sum_j w_j K(\boldsymbol{x}_j, \boldsymbol{z})$$

General form

$$\sum_i w_i K(\boldsymbol{x}_i, \boldsymbol{z})$$

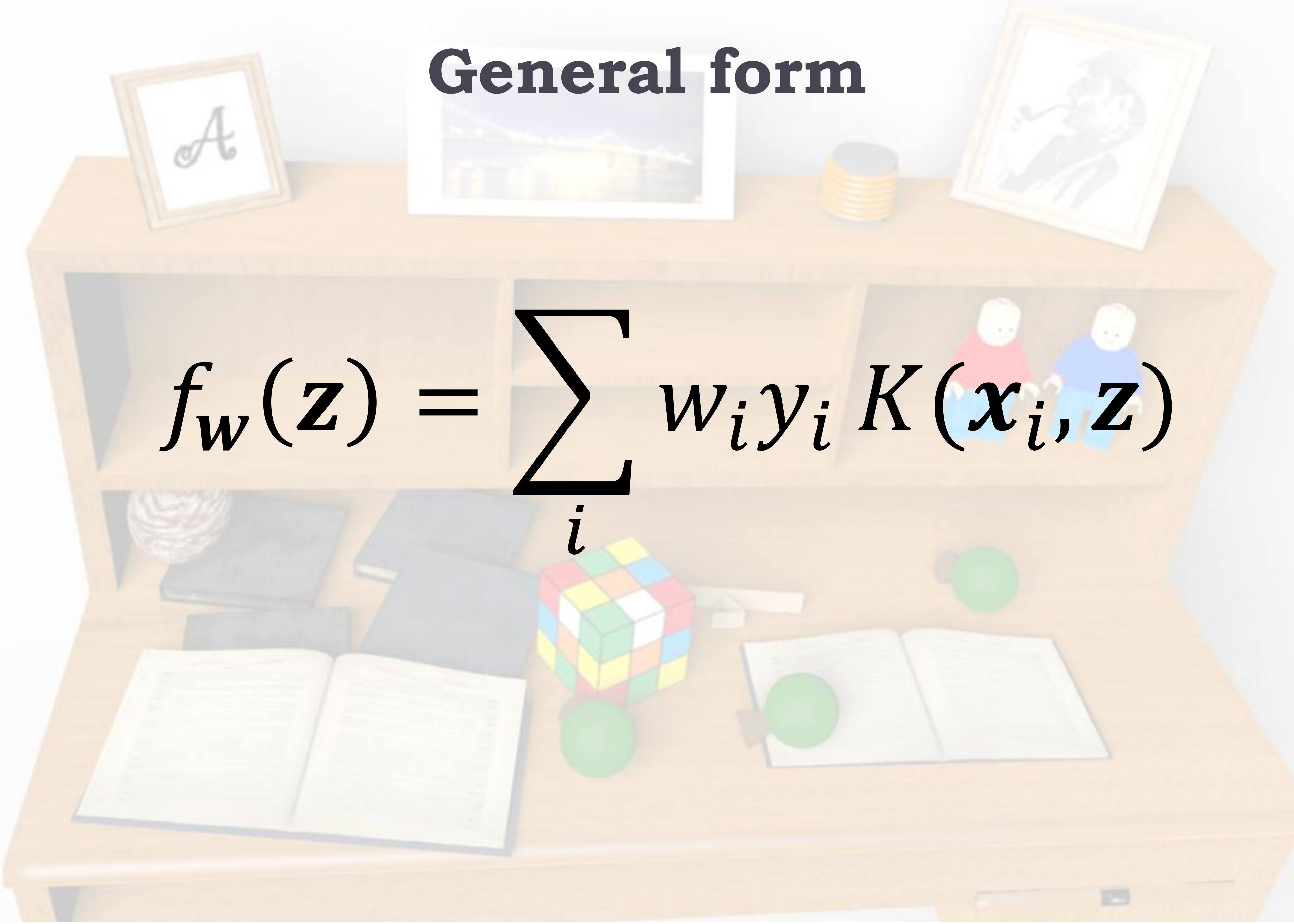
$$-\sum_j w_j K(\boldsymbol{x}_j, \boldsymbol{z})$$

General form

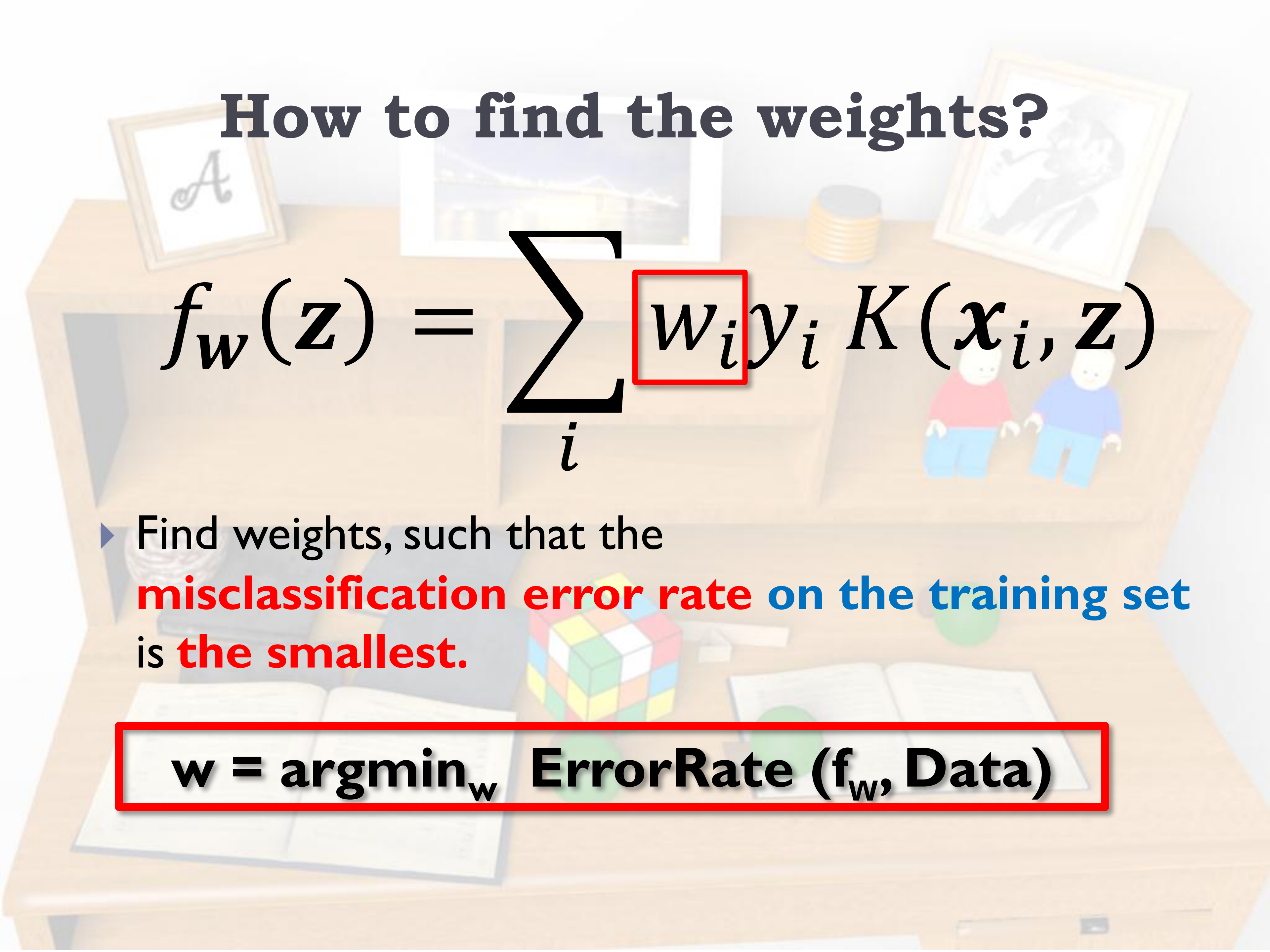
$$\sum_i w_i y_i K(\boldsymbol{x}_i, \boldsymbol{z})$$

$$+ \sum_j w_j y_j K(\boldsymbol{x}_j, \boldsymbol{z})$$

General form



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



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.**

w = argmin$_{w}$ ErrorRate (f$_{w}$, Data)

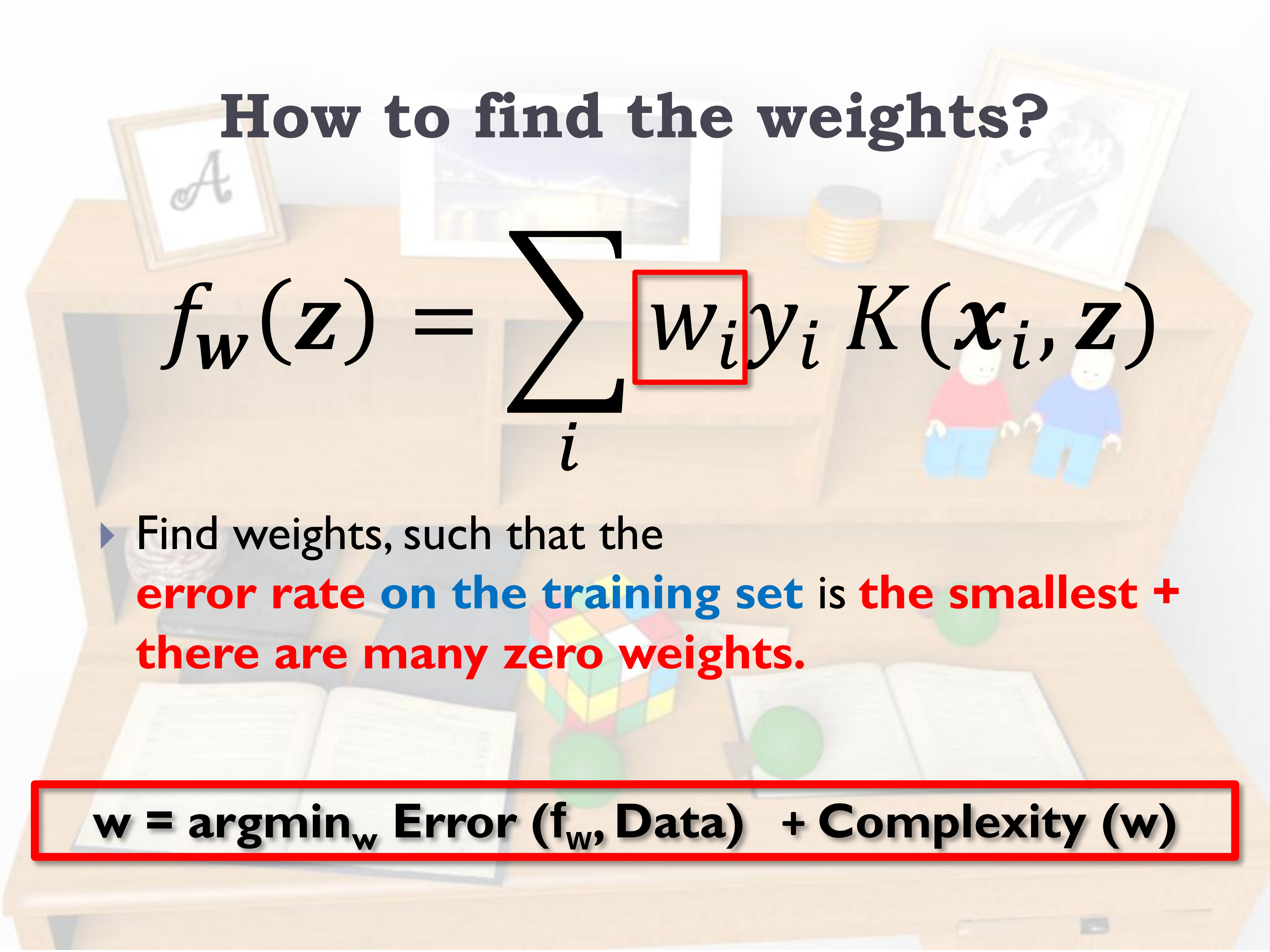
How to find the weights?

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

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

$$\boxed{\mathbf{w} = \mathrm{argmin}_{\mathrm{w}} \ \ \mathrm{Error}\ (\mathrm{f}_{\mathrm{w}}, \mathrm{Data})}$$

How to find the weights?

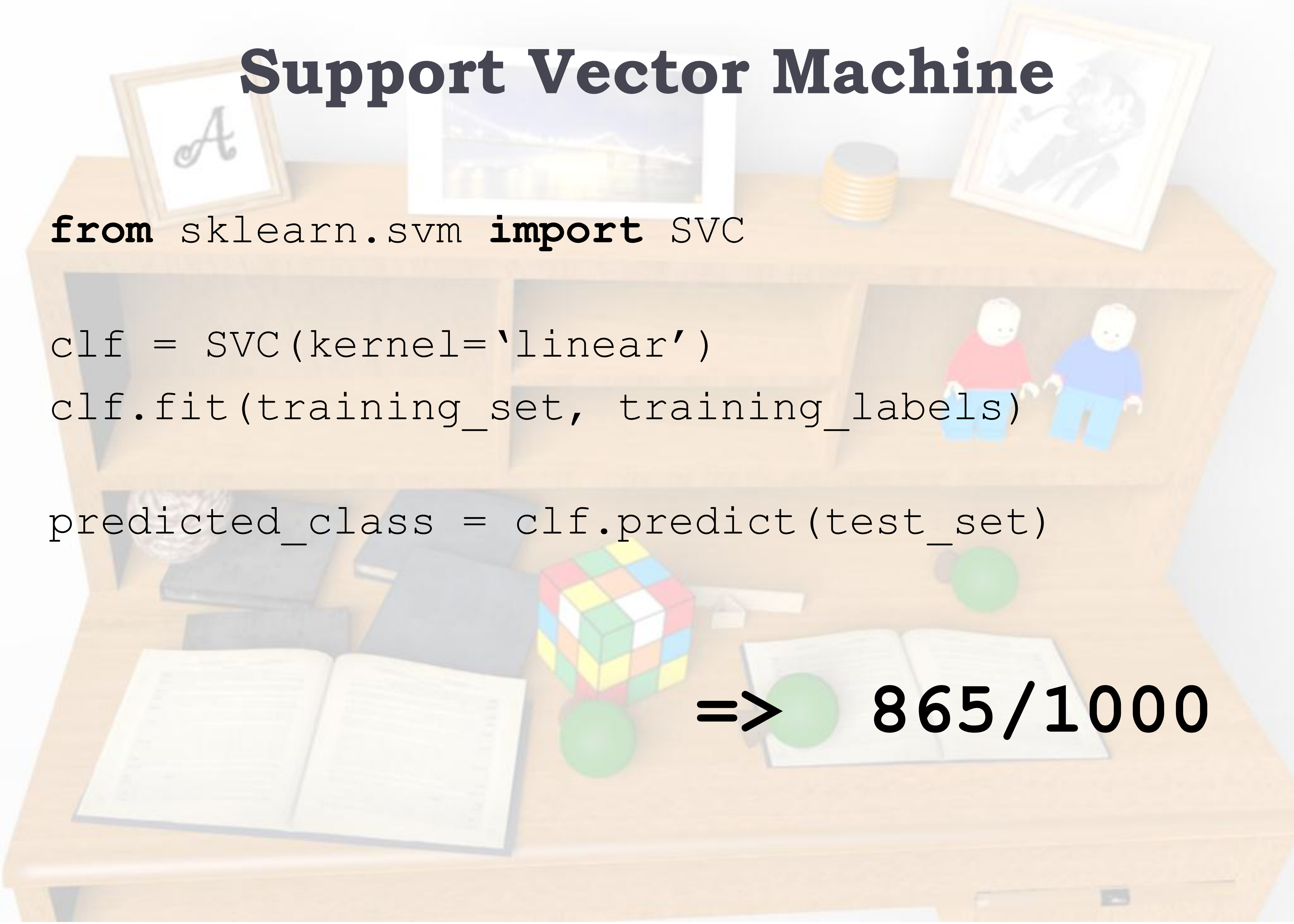


$$f_{\mathbf{w}}(\mathbf{z}) = \sum_i \boxed{w_i} y_i\, K(\mathbf{x}_i, \mathbf{z})$$

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

$$\boxed{\mathbf{w = argmin_w\ Error\ (f_w, Data)\ \ + Complexity\ (w)}}$$

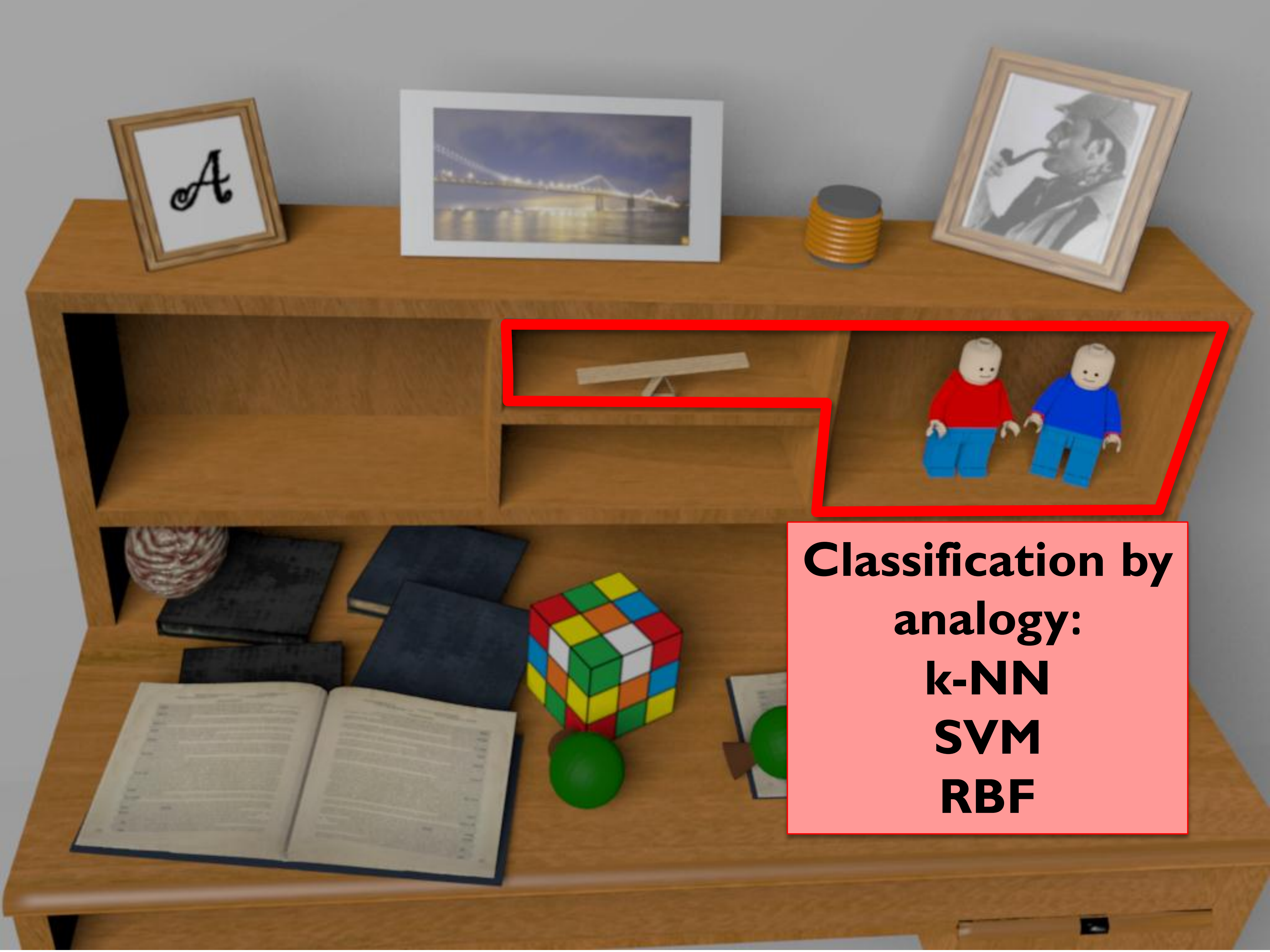
Support Vector Machine



```
from sklearn.svm import SVC

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

predicted_class = clf.predict(test_set)
```

=> 865/1000

A



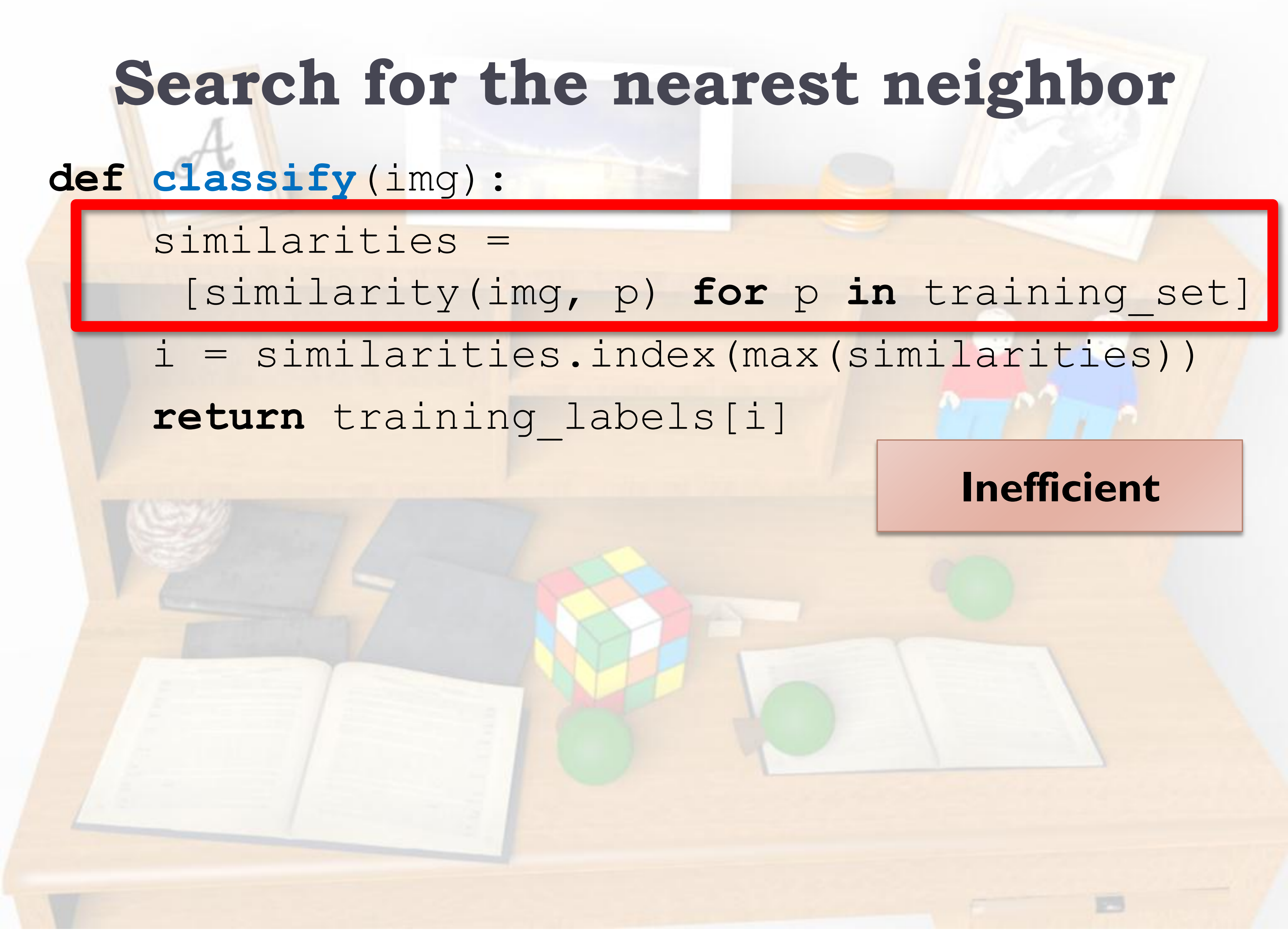
Classification by analogy:

k-NN

SVM

RBF

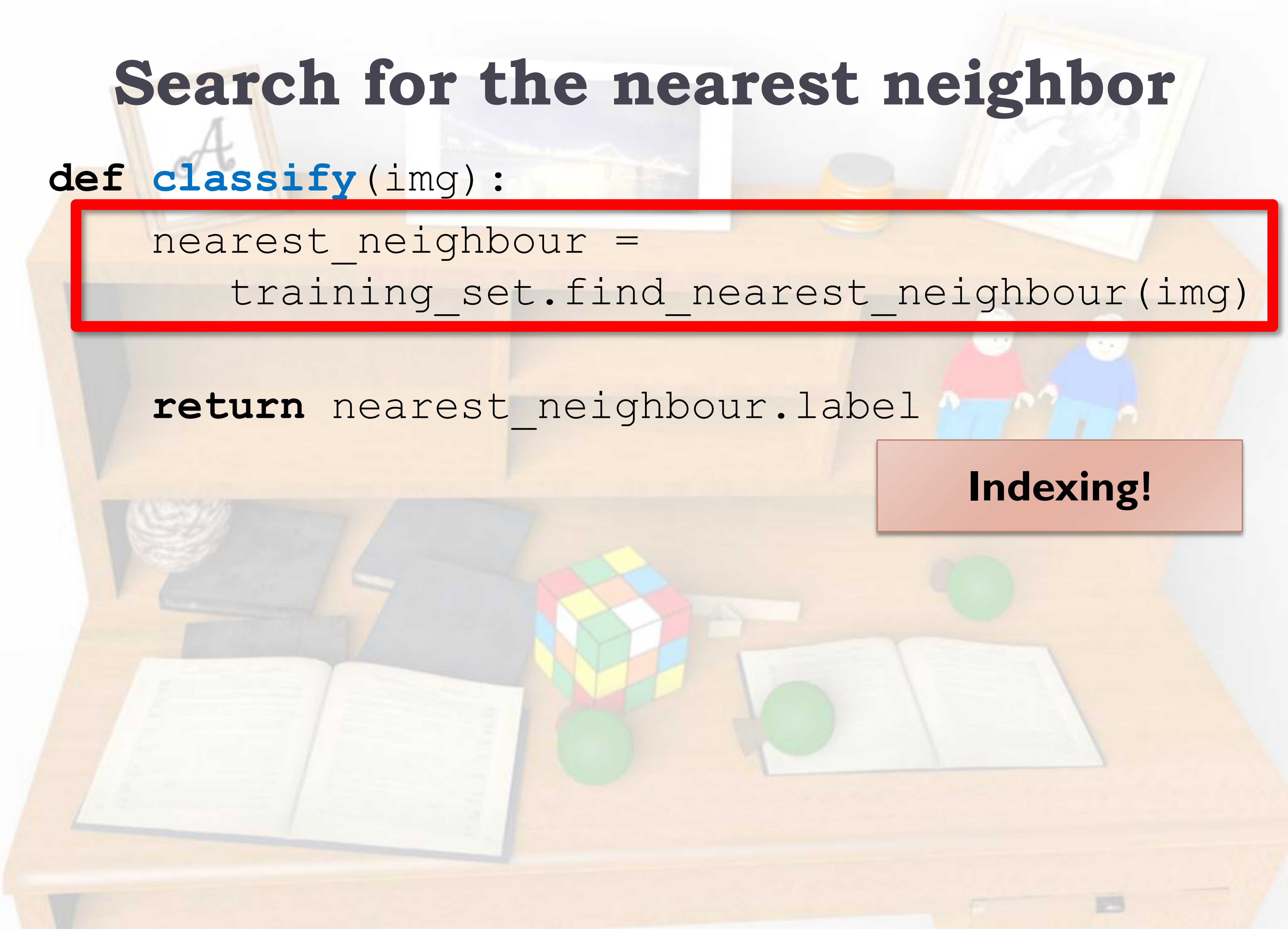
Search for the nearest neighbor



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

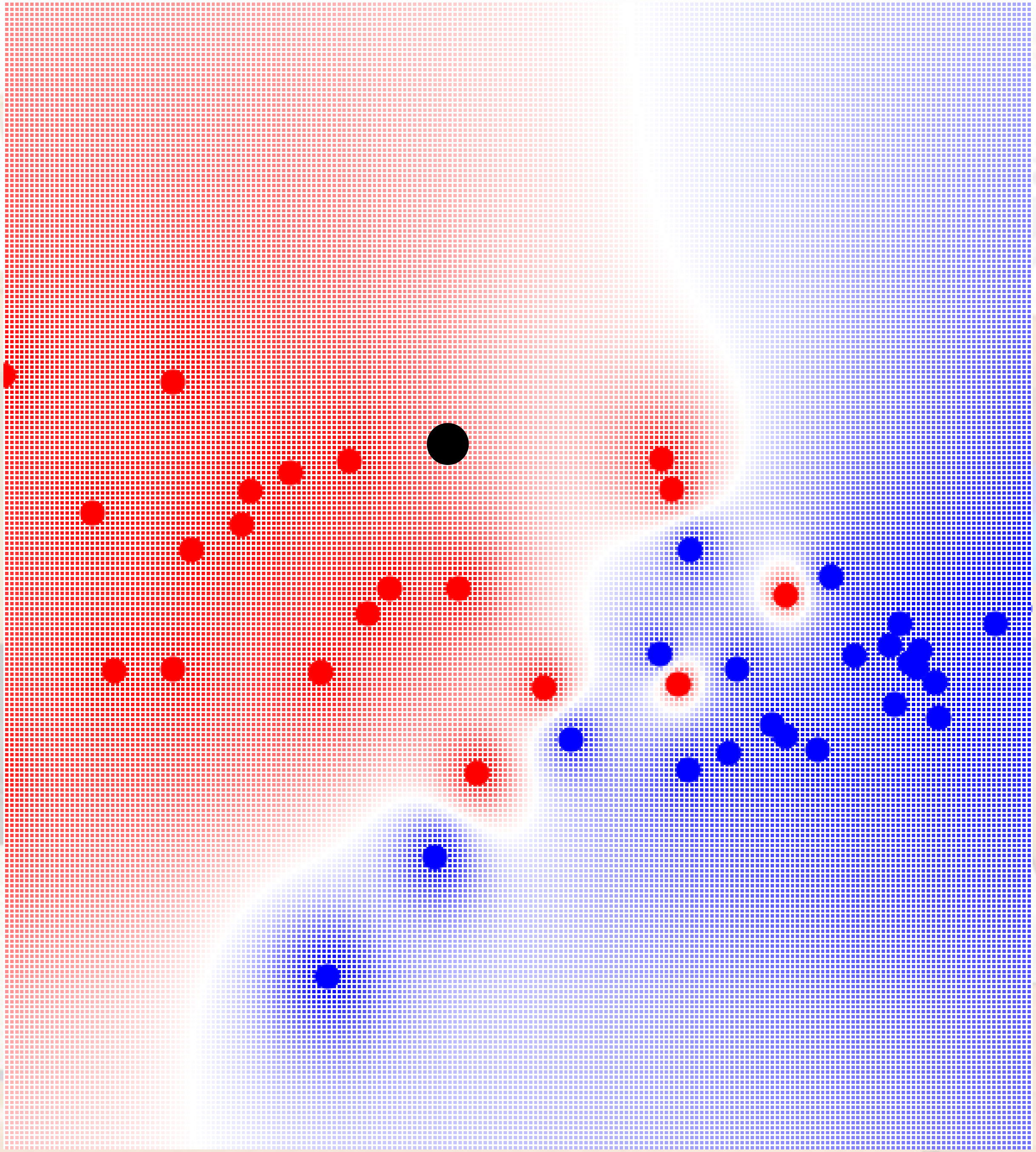
Inefficient

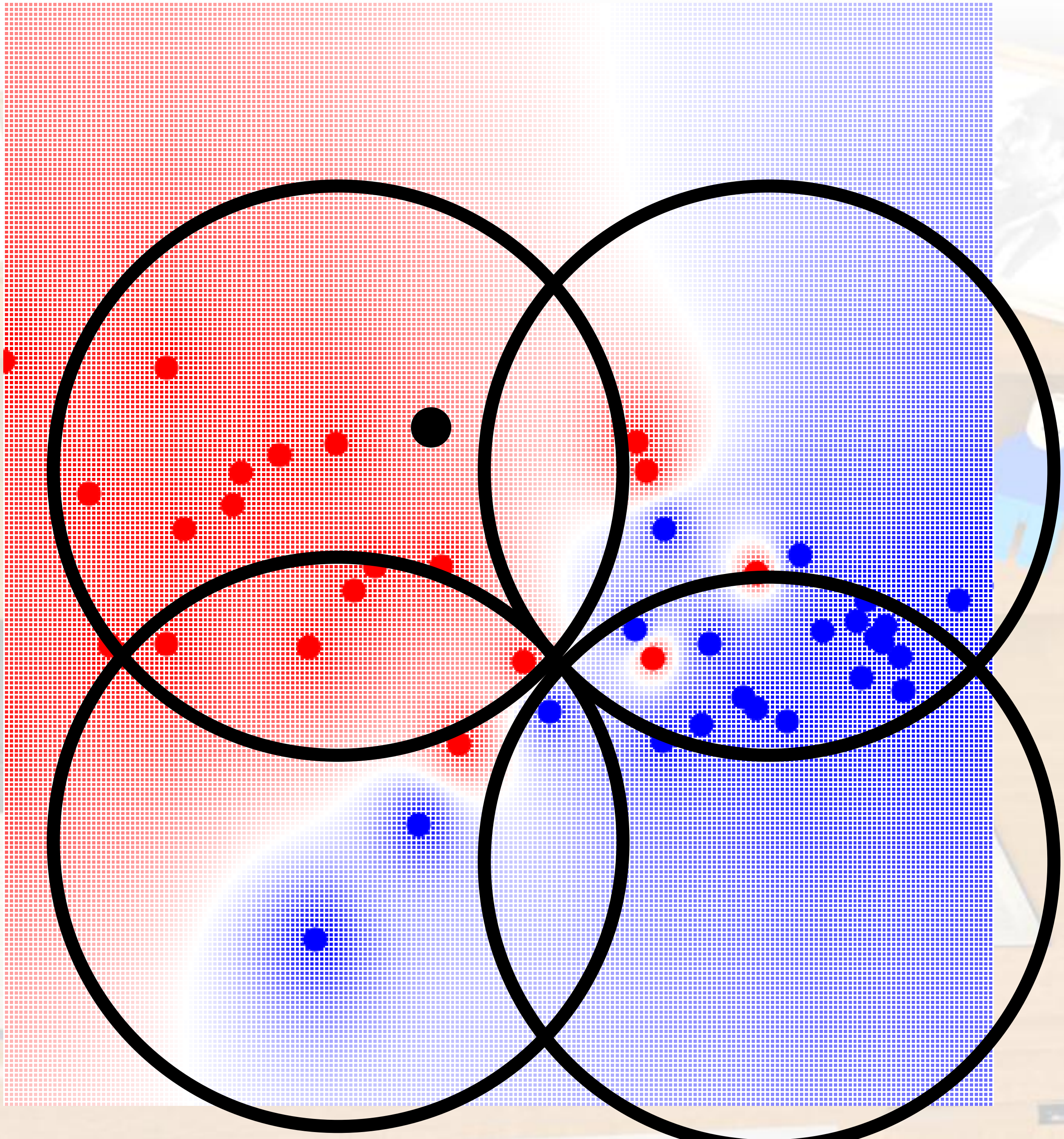
Search for the nearest neighbor

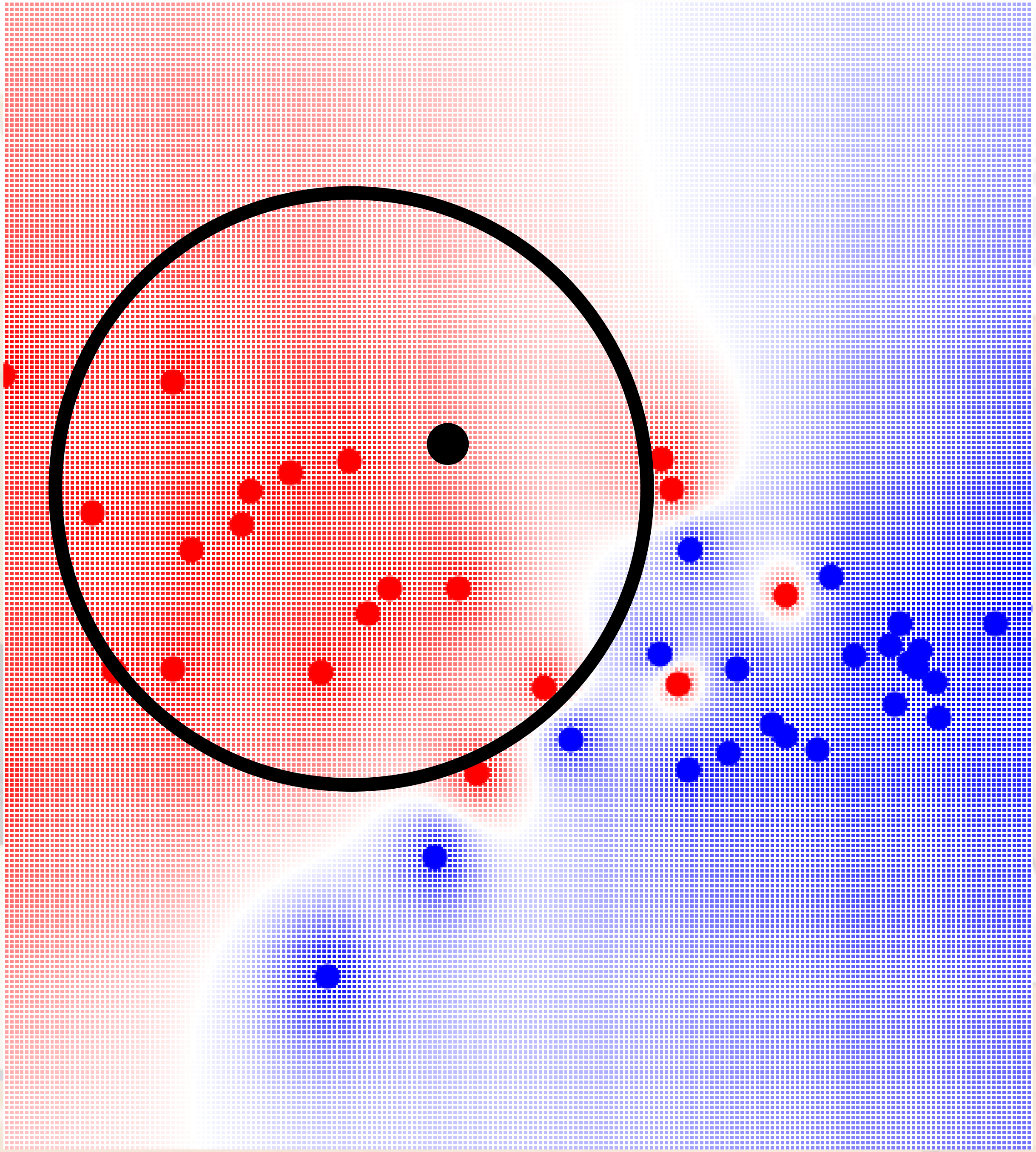


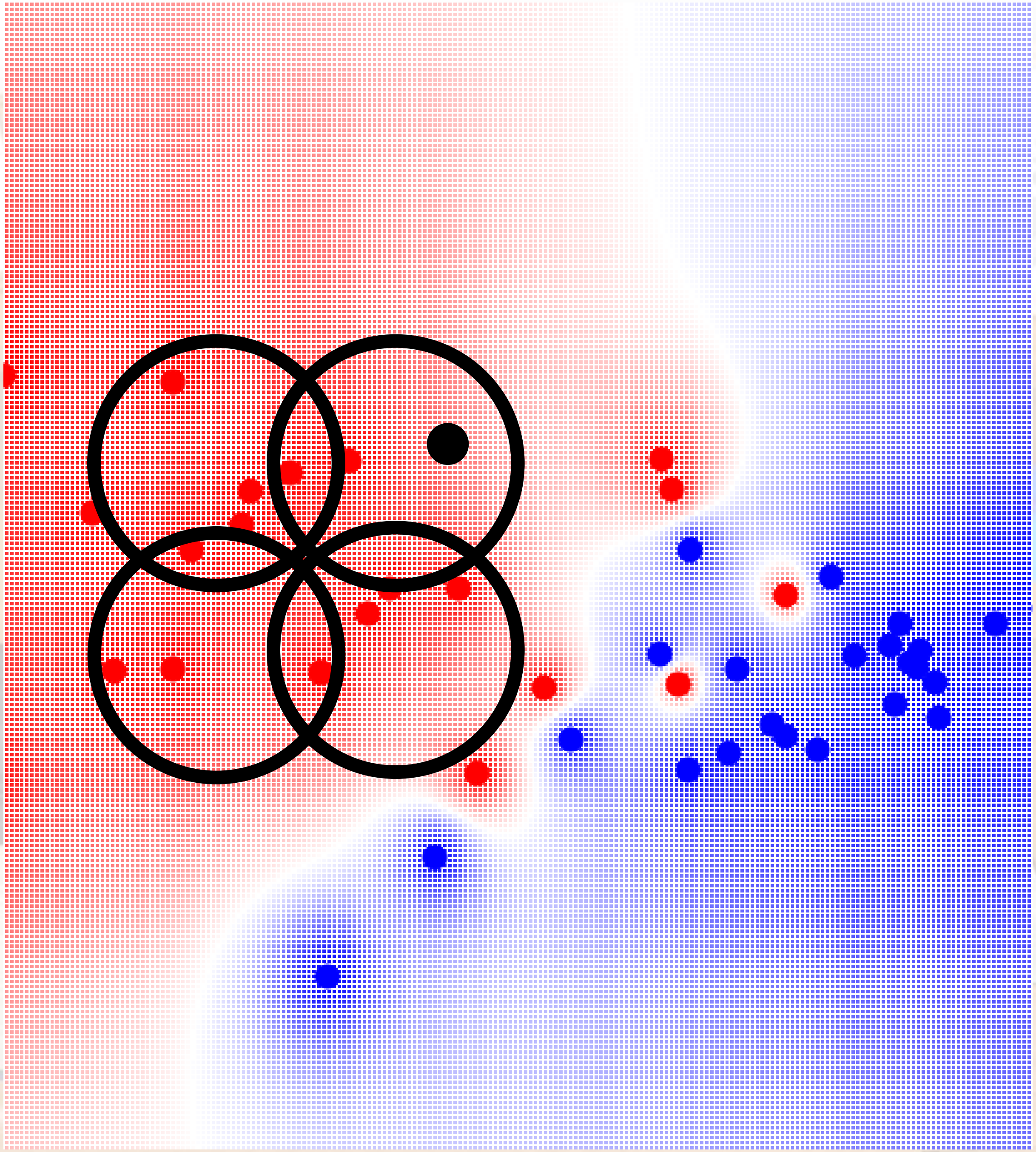
```
def classify(img):
    nearest_neighbour =
        training_set.find_nearest_neighbour(img)

    return nearest_neighbour.label
```

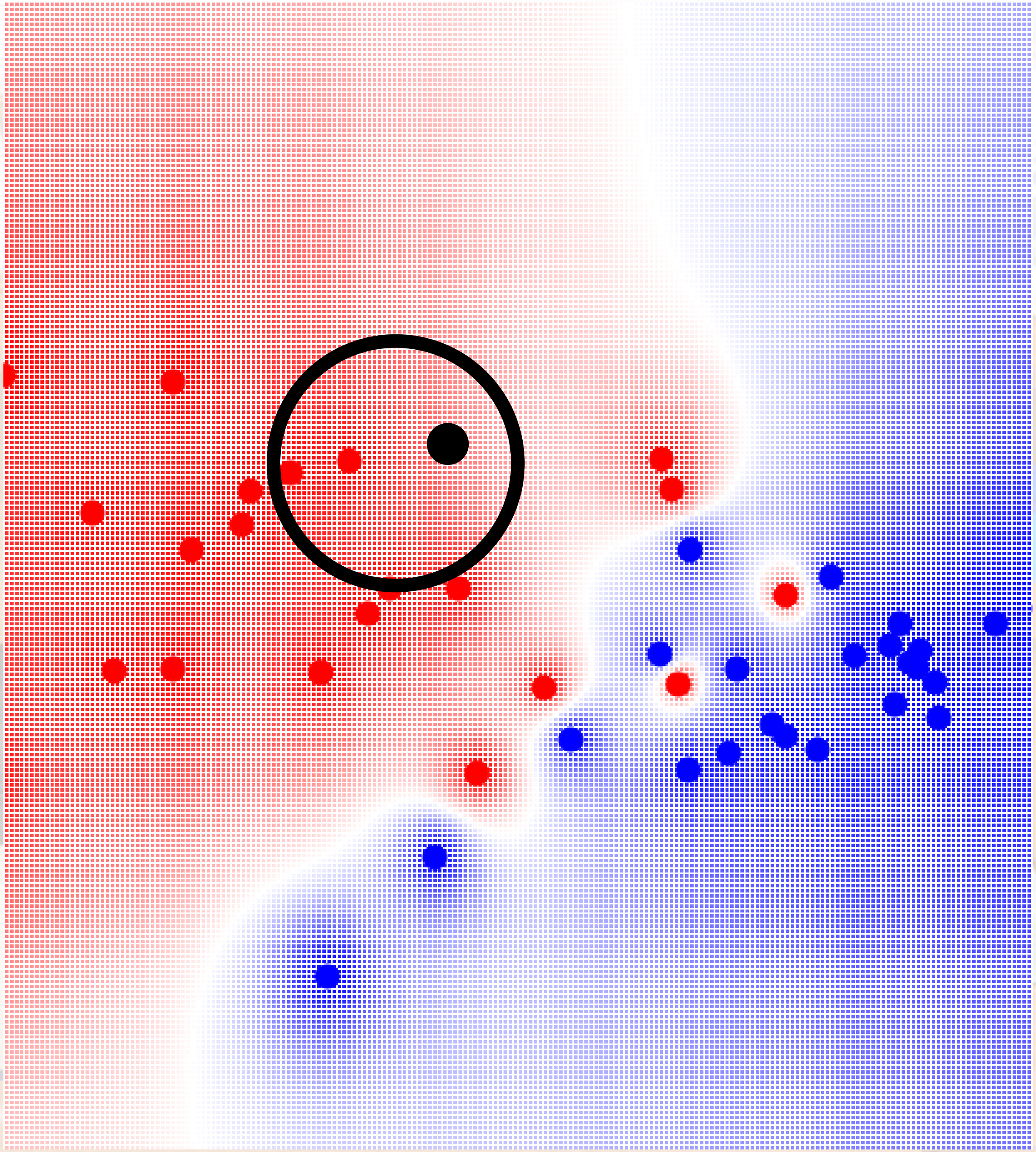
Indexing!







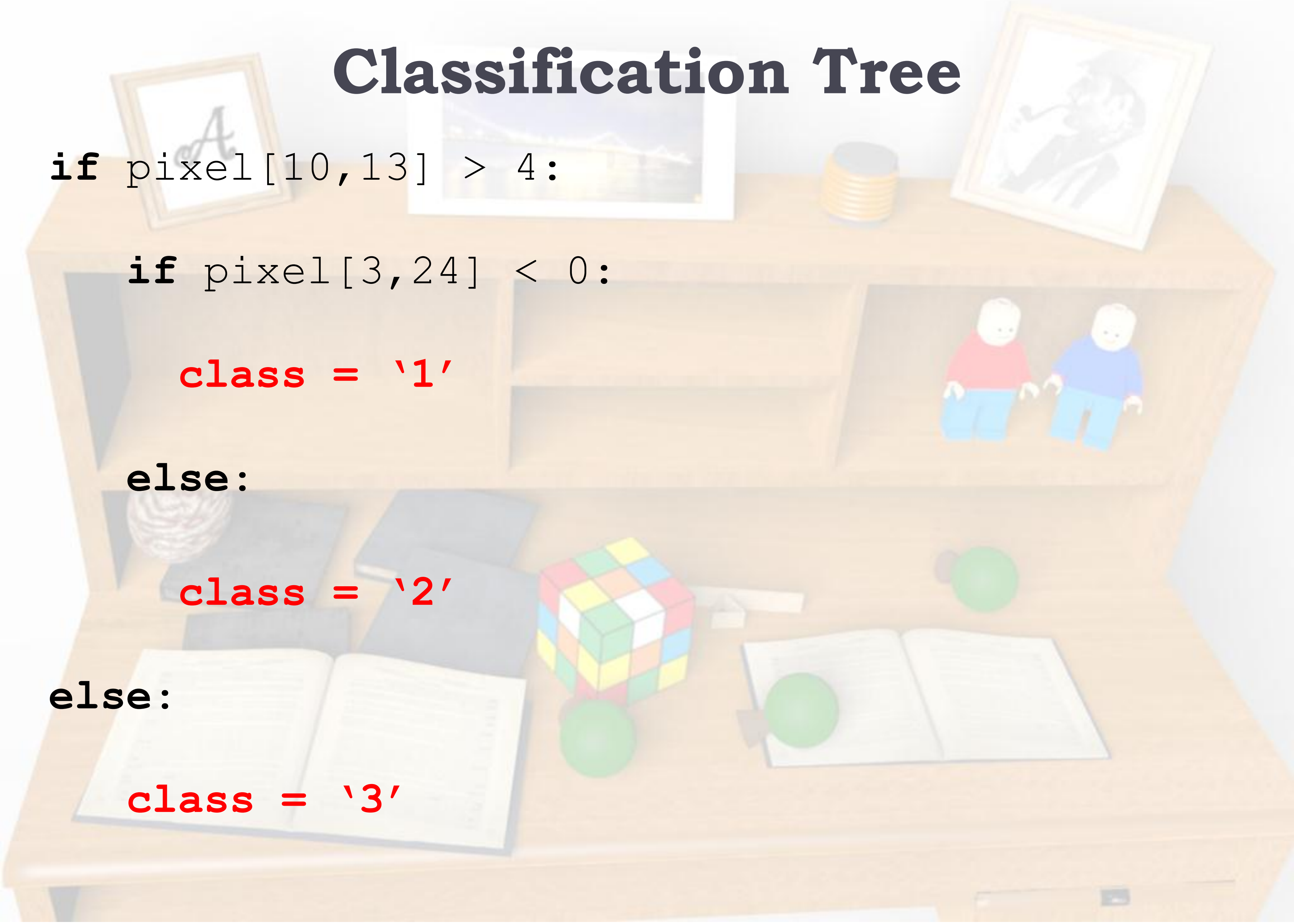




find_nearest_neighbour

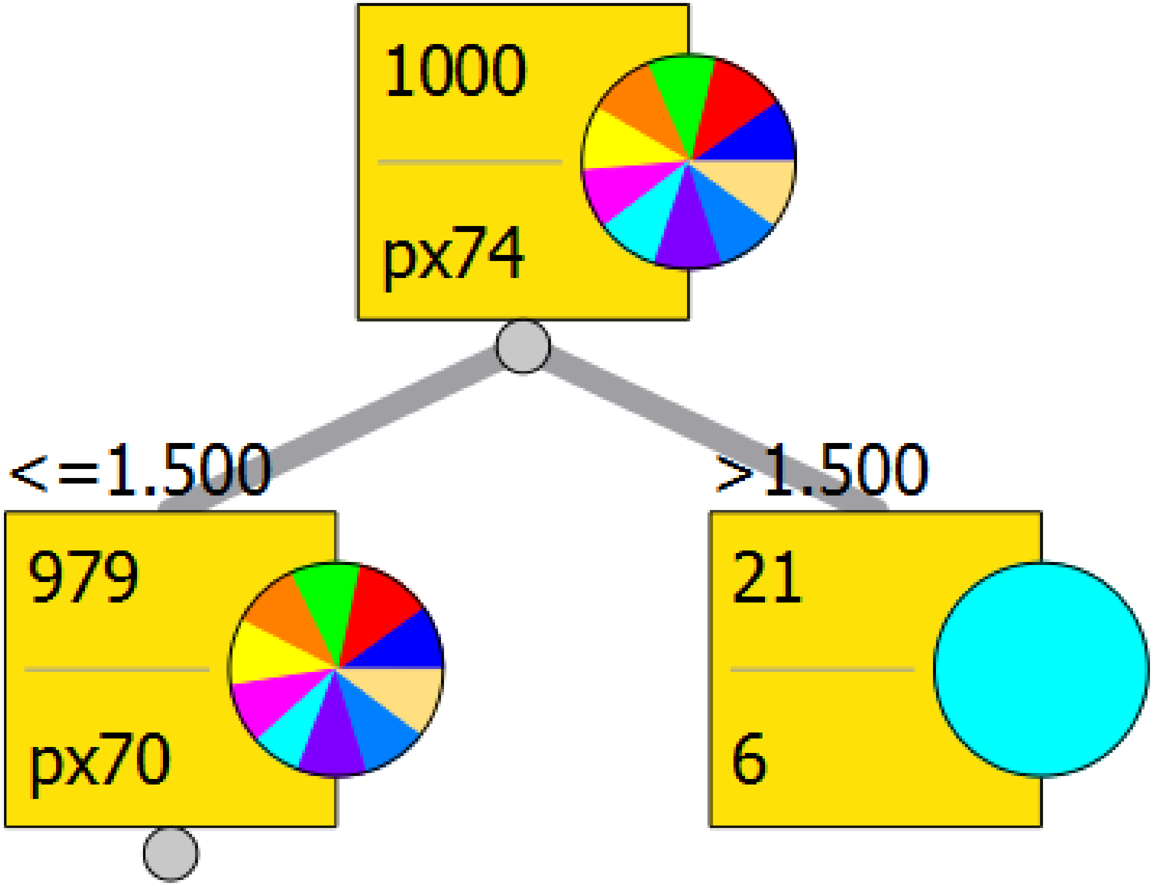
```
if pixel[10,13] > 4:
  if pixel[3,24] < 0:
    nearest_neighbour = A
  else:
    nearest_neighbour = B
else:
  nearest_neighbour = C
```

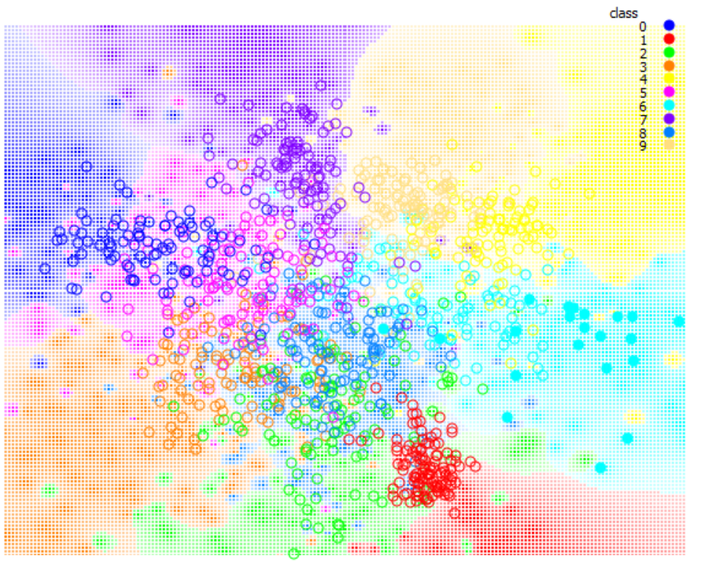
Classification Tree



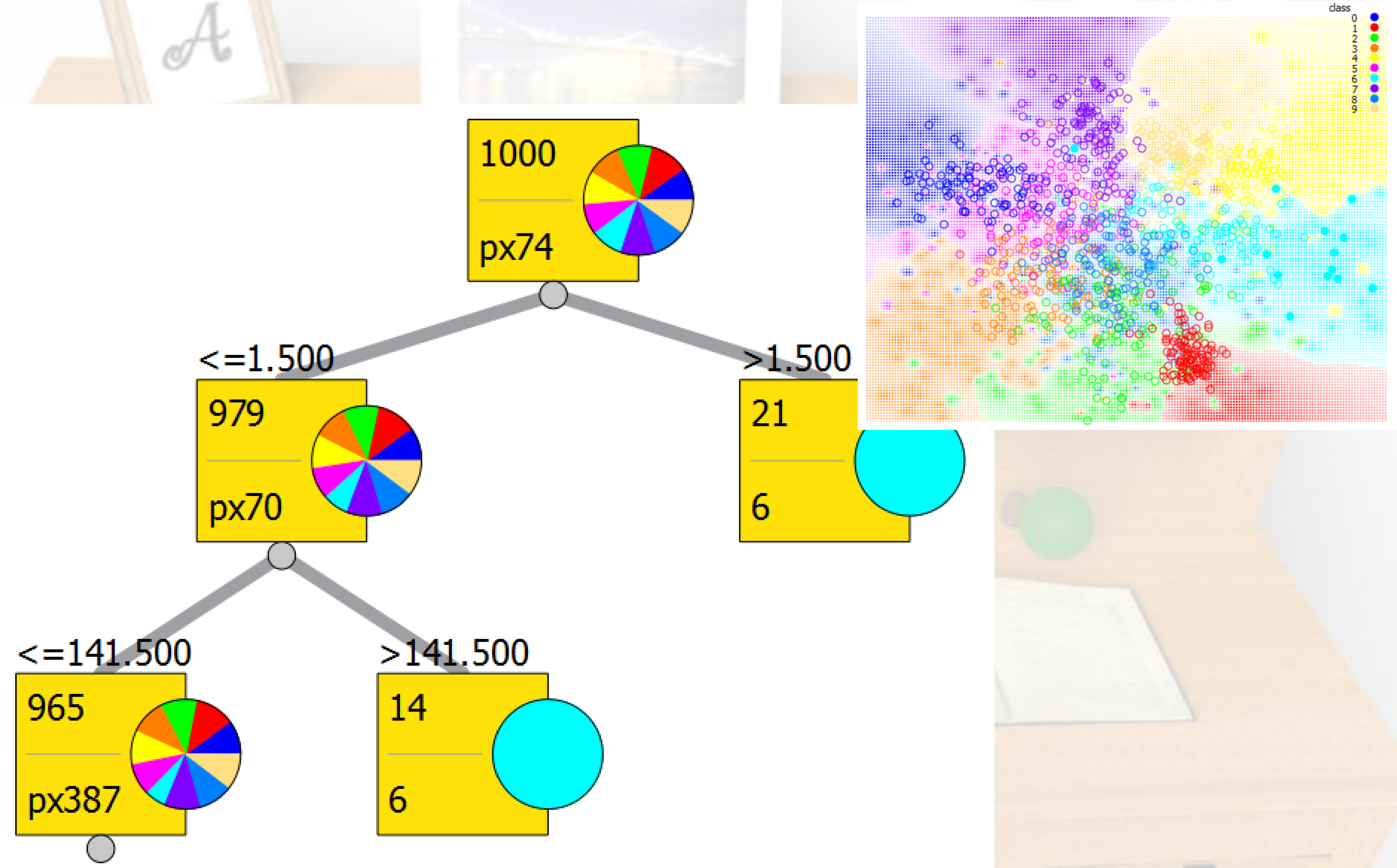
```
if pixel[10,13] > 4:
   if pixel[3,24] < 0:
      class = ‘1’
   else:
      class = ‘2’
else:
   class = ‘3’
```

Classification Tree

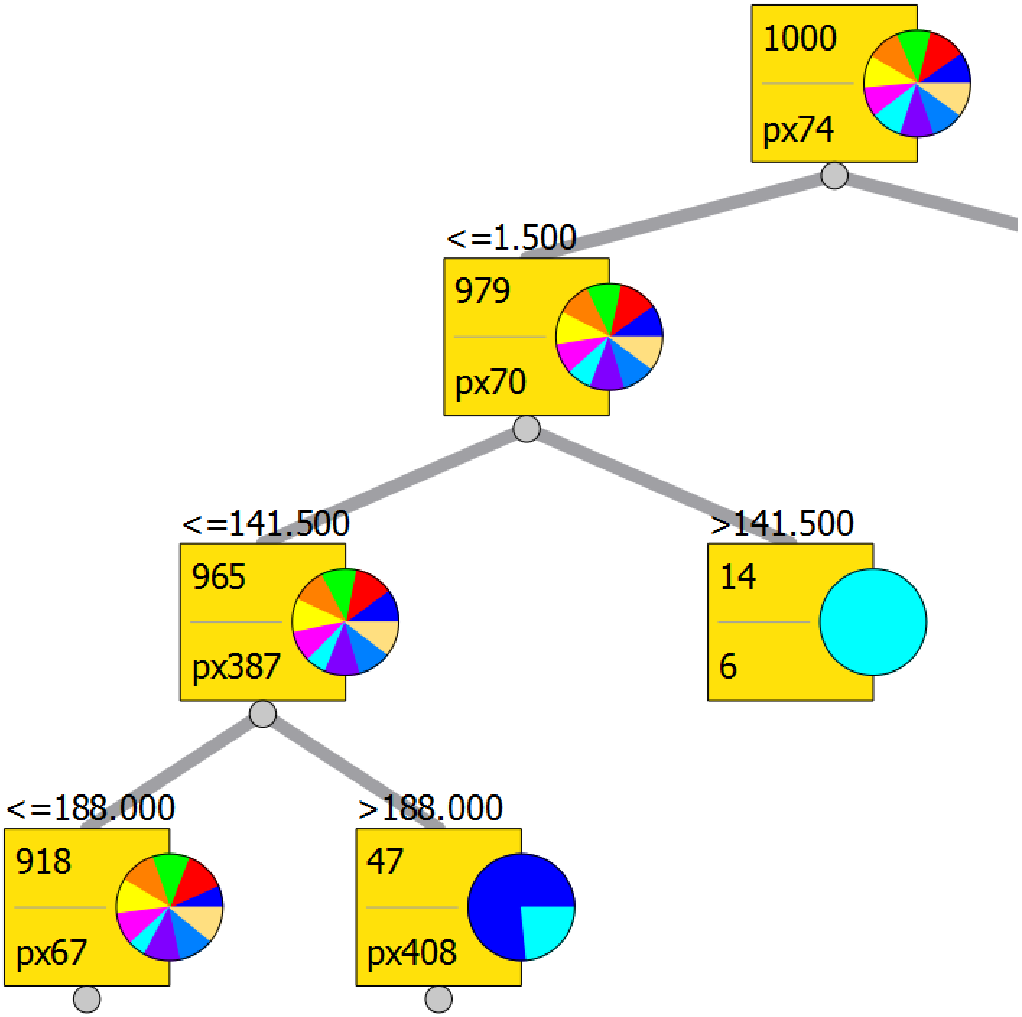


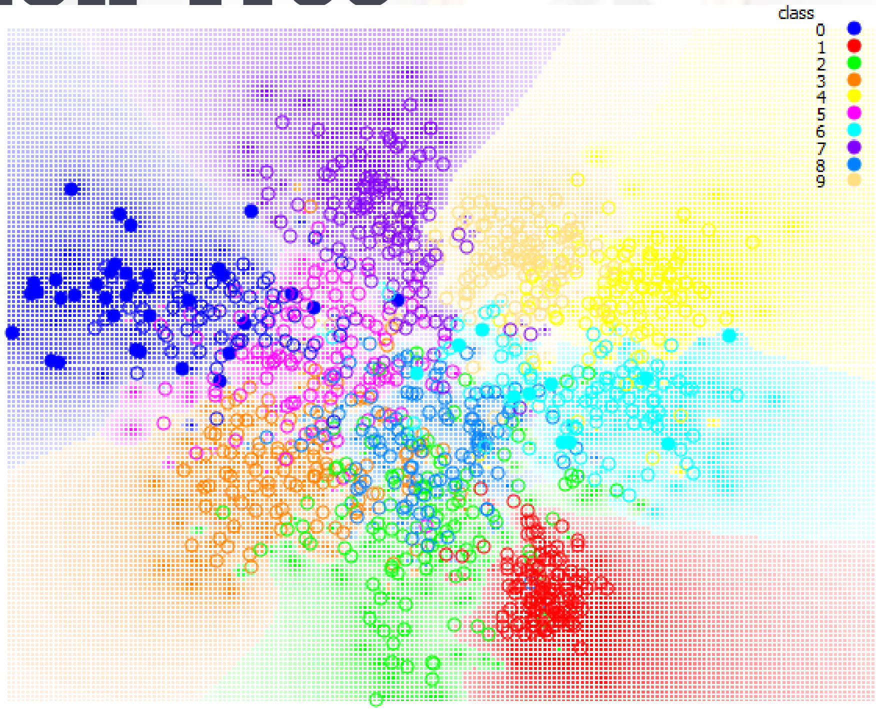


Classification Tree



Classification Tree



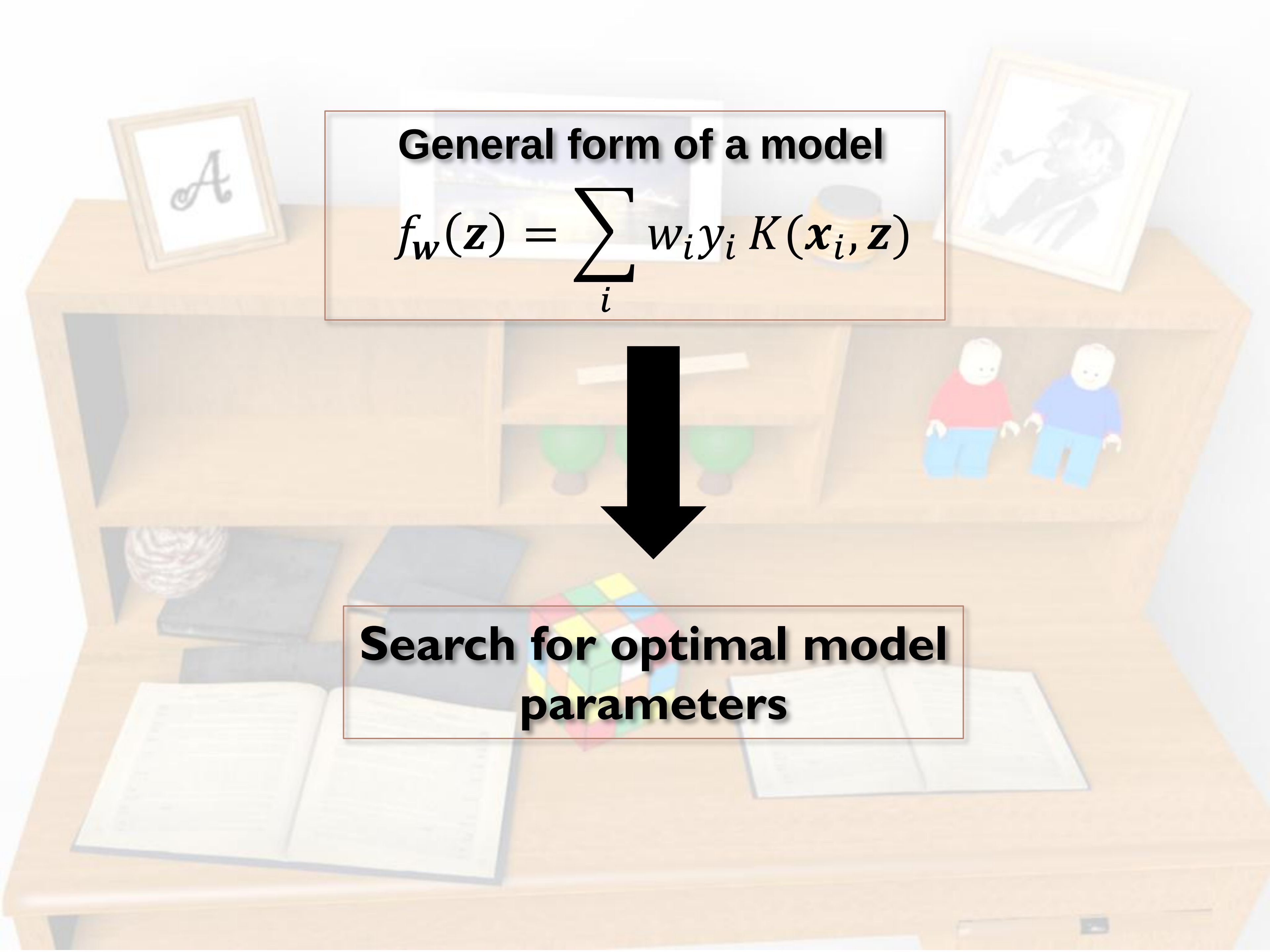


Orange

http://orange.biolab.si/

A

Trees
ID3
C4.5
RegTree



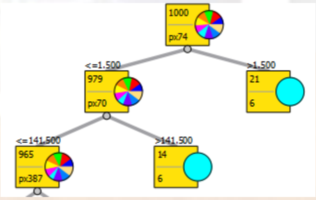
General form of a model

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

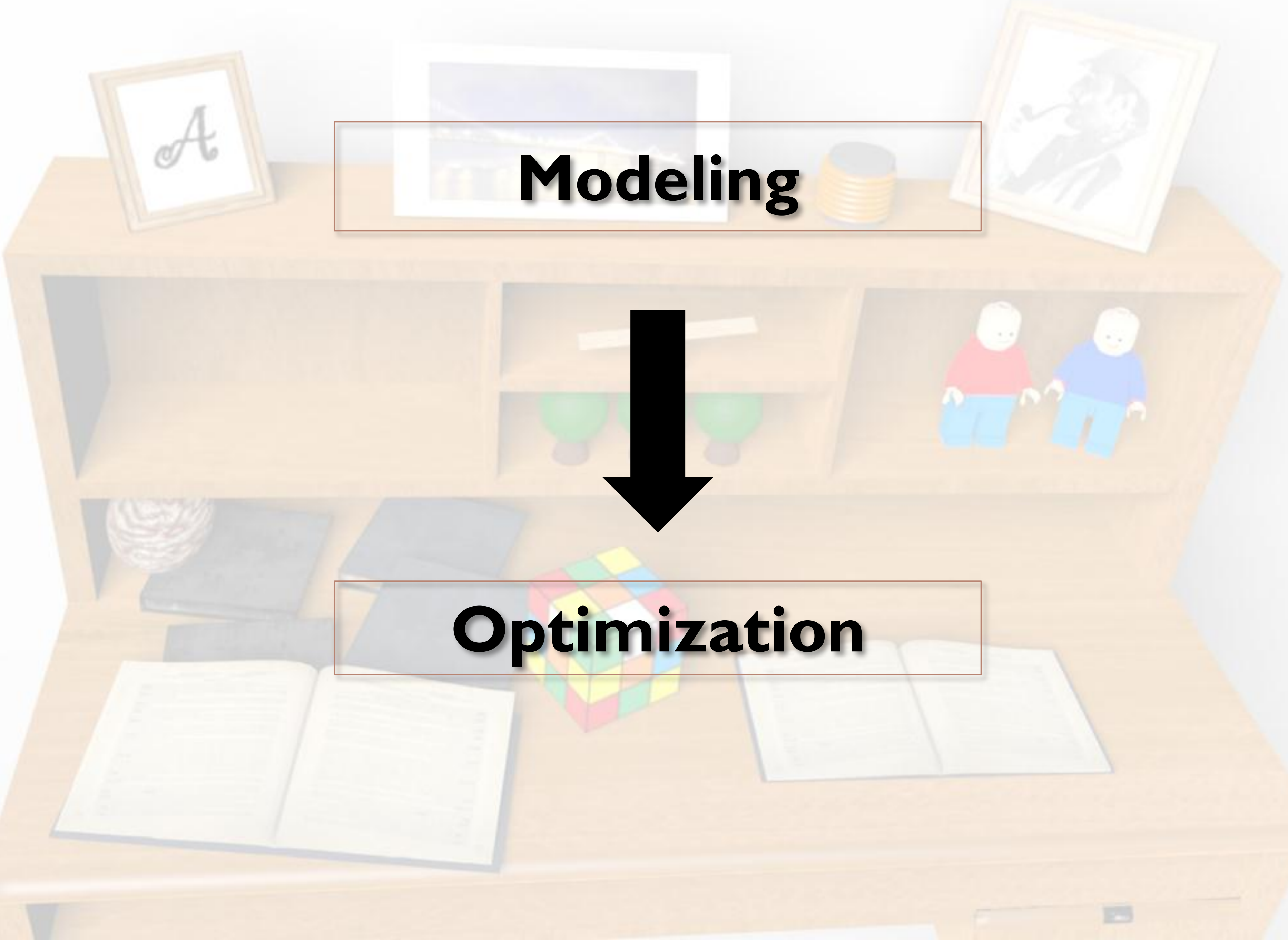
⬇

Search for optimal model parameters

General form of a model

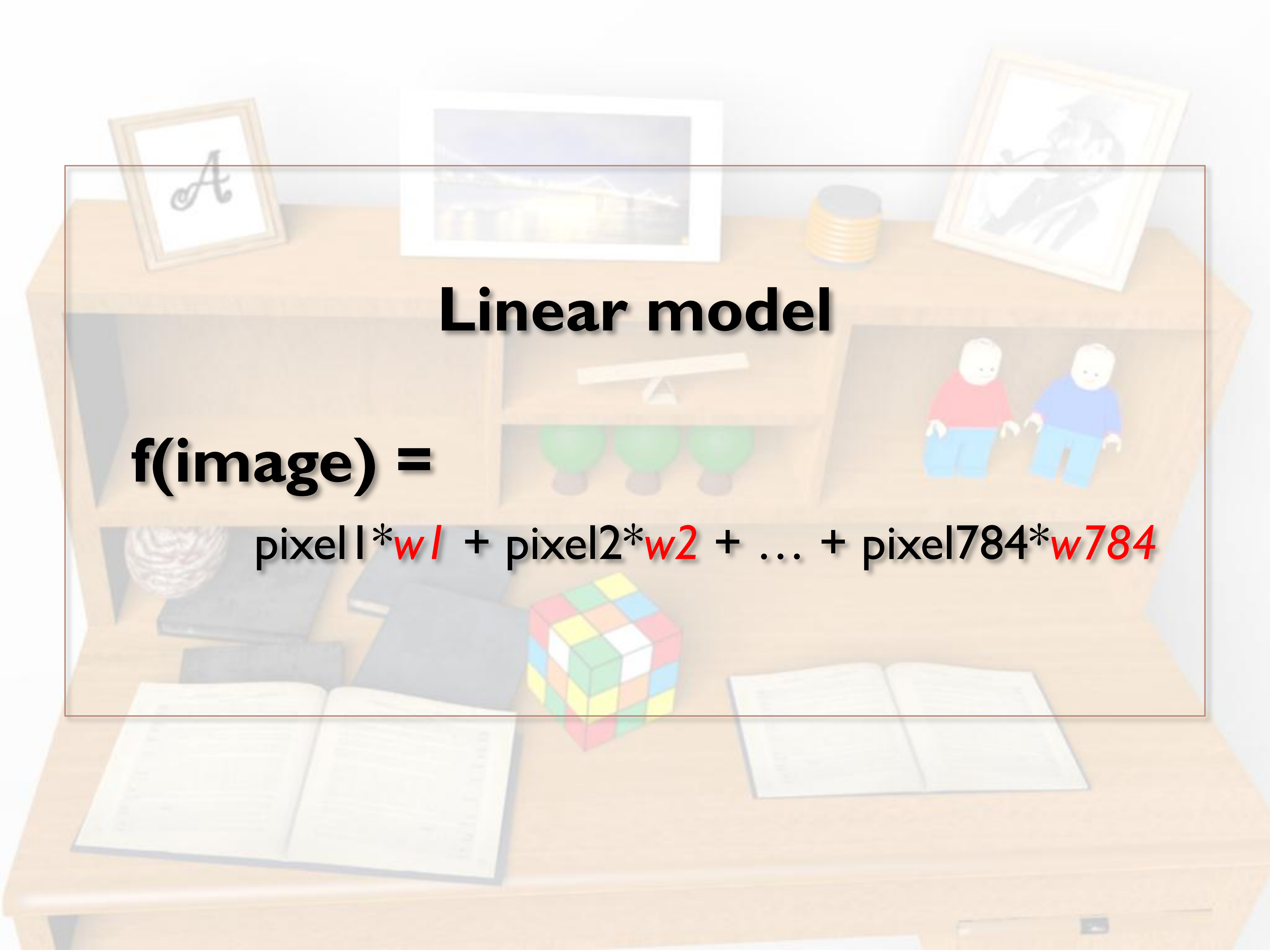


Search for the optimal tree



Modeling

Optimization

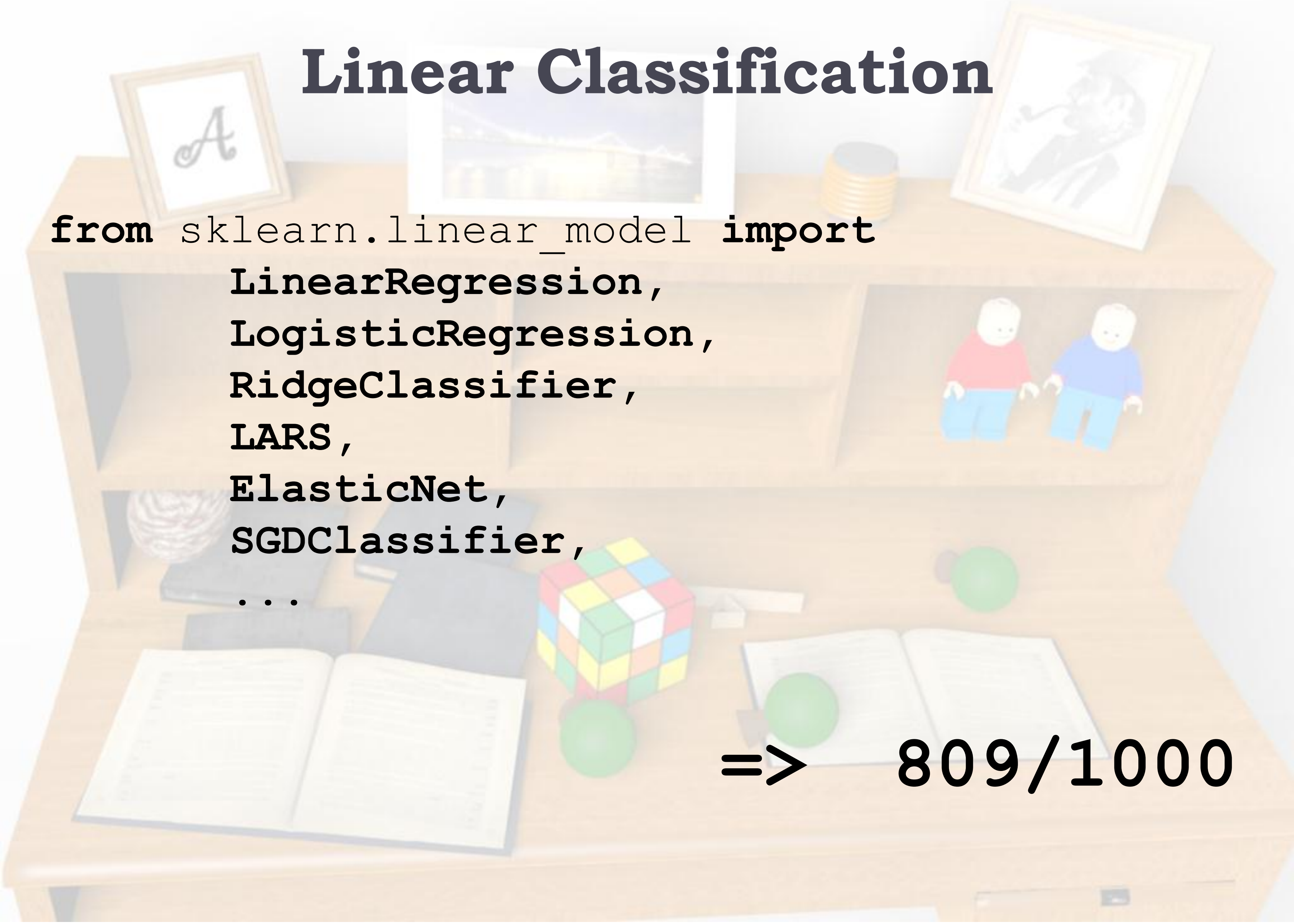


Linear model

f(image) =

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

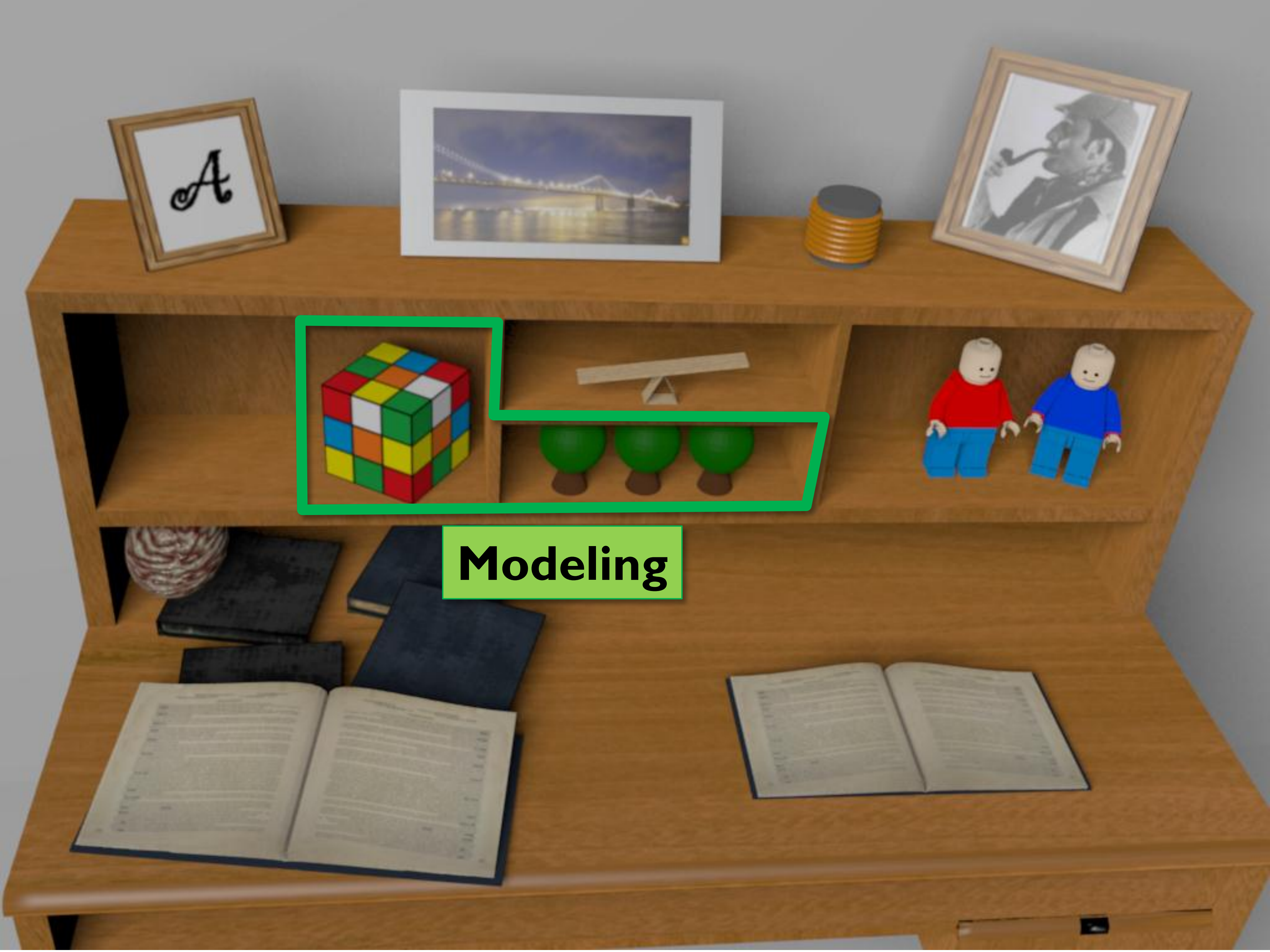
Linear Classification



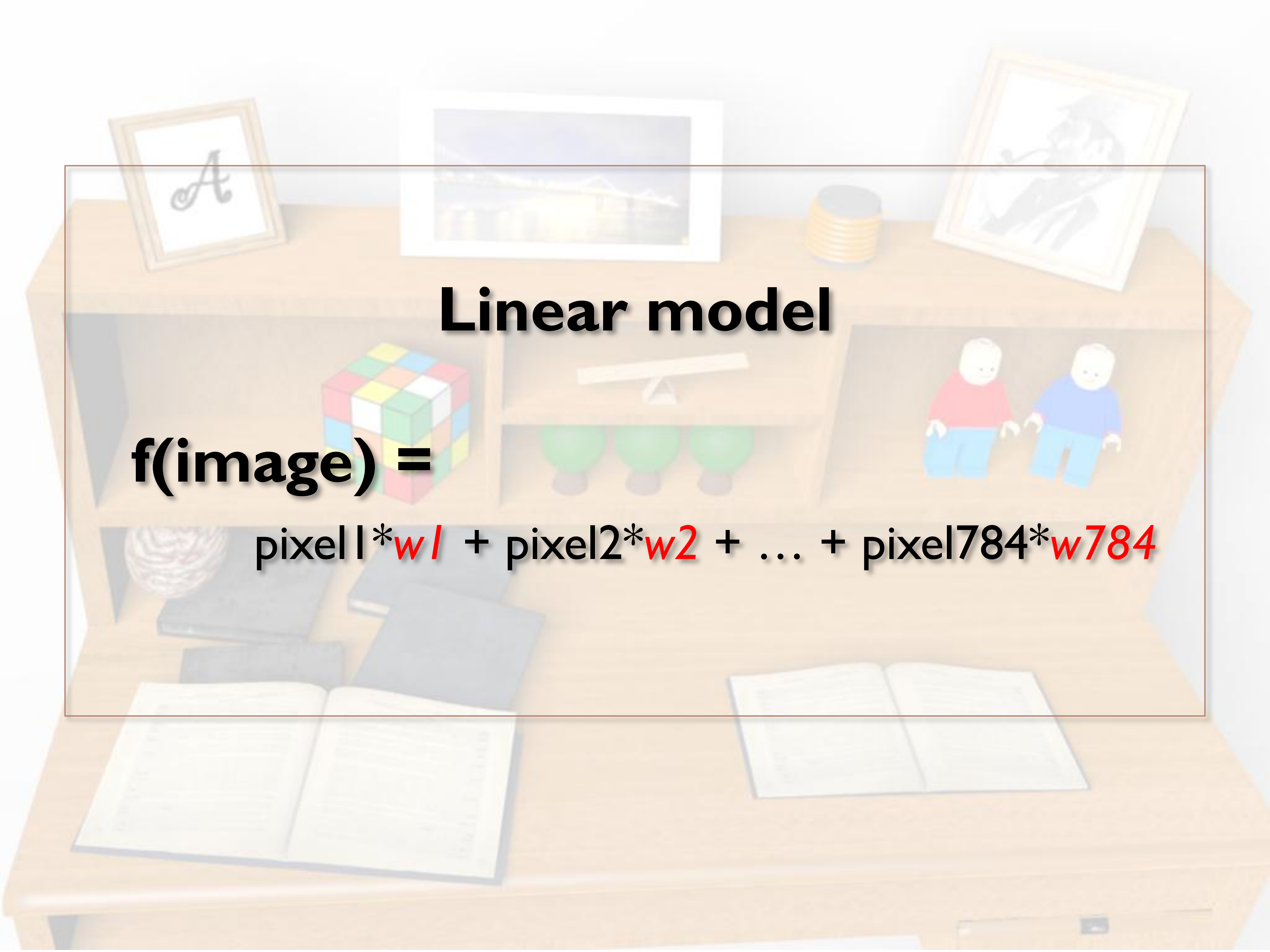
```
from sklearn.linear_model import
          LinearRegression,
          LogisticRegression,
          RidgeClassifier,
          LARS,
          ElasticNet,
          SGDClassifier,
          ...
```

=> 809/1000

A



Modeling



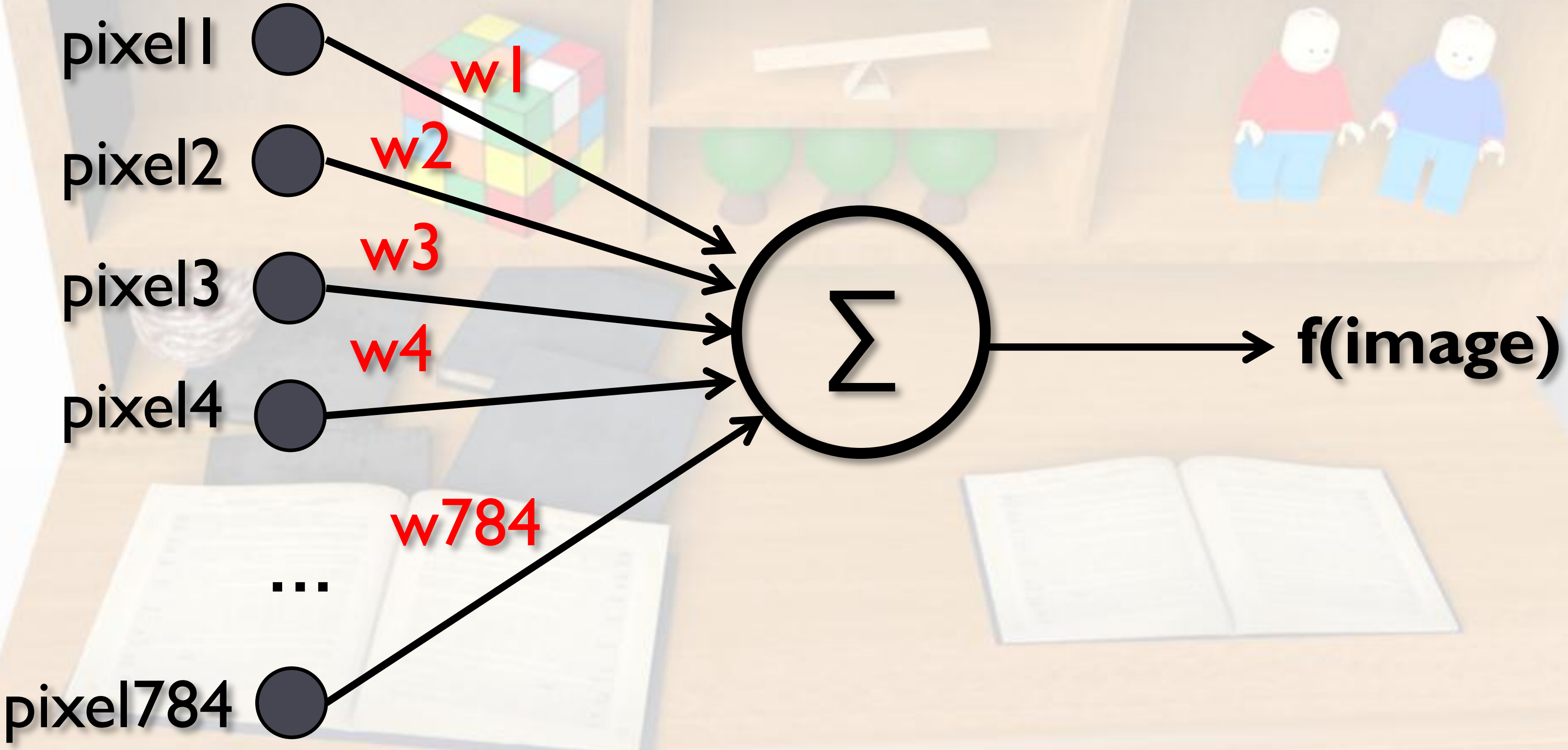
Linear model

f(image) =

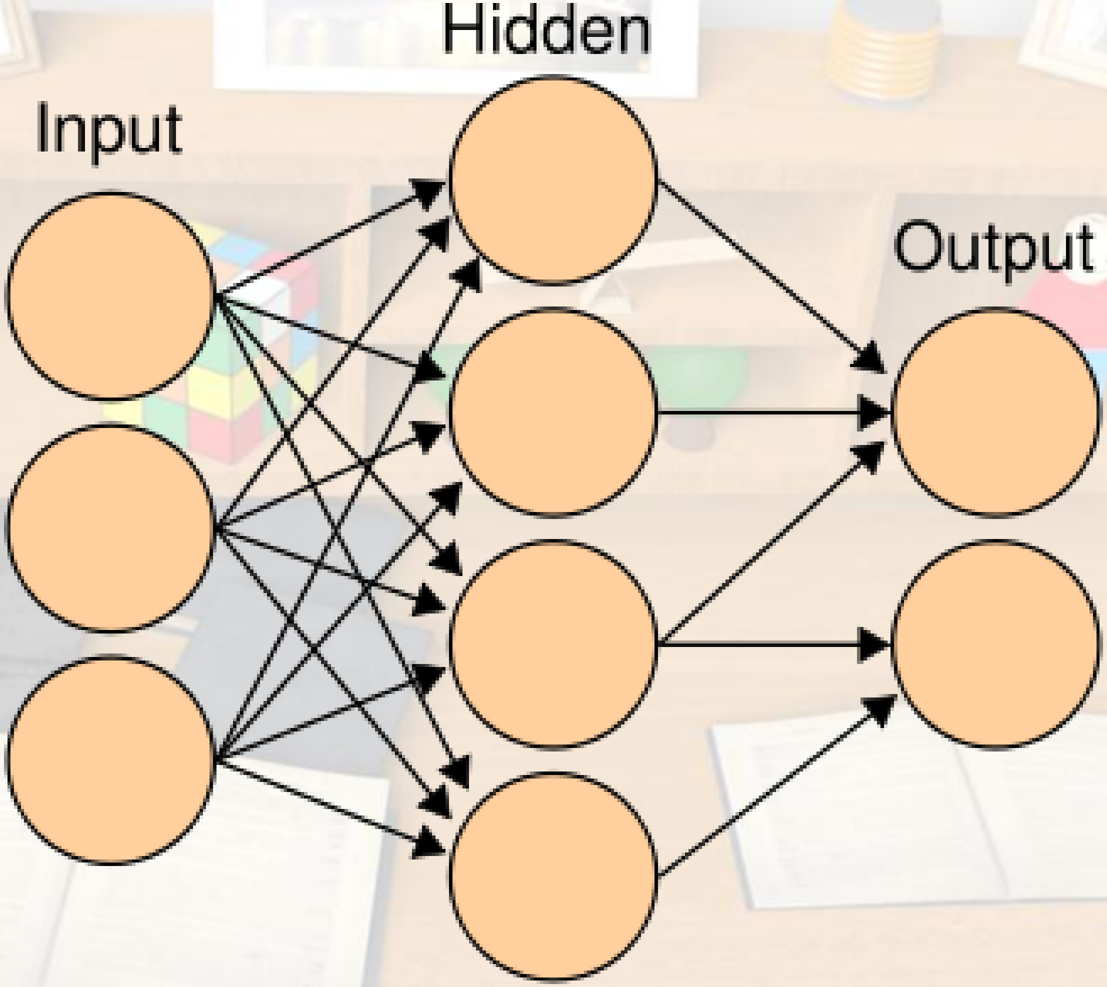
pixel1*w1 + pixel2*w2 + … + pixel784*w784

f(image) =

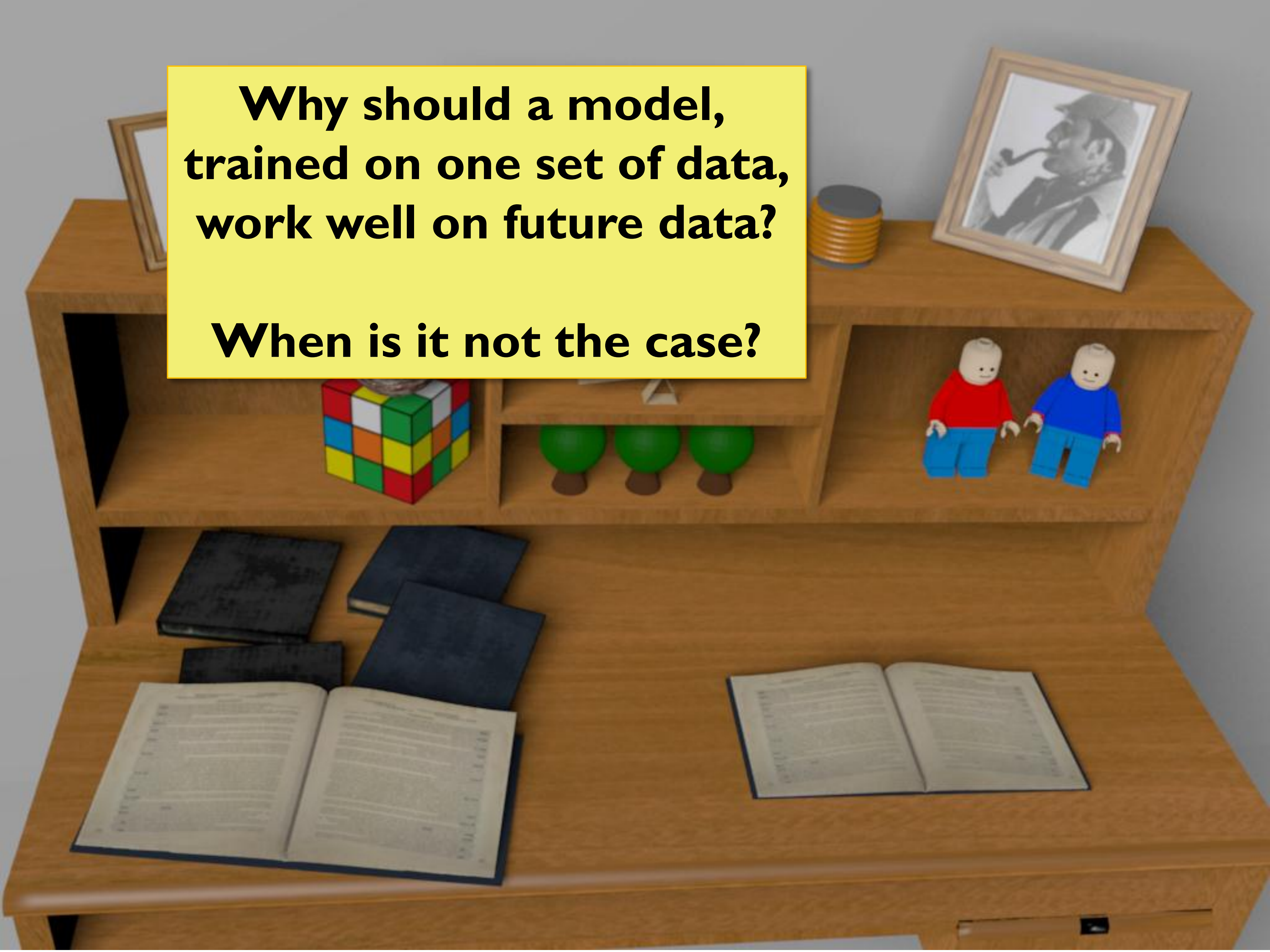
pixel1*w1 + pixel2*w2 + … + pixel784*w784



Neural network

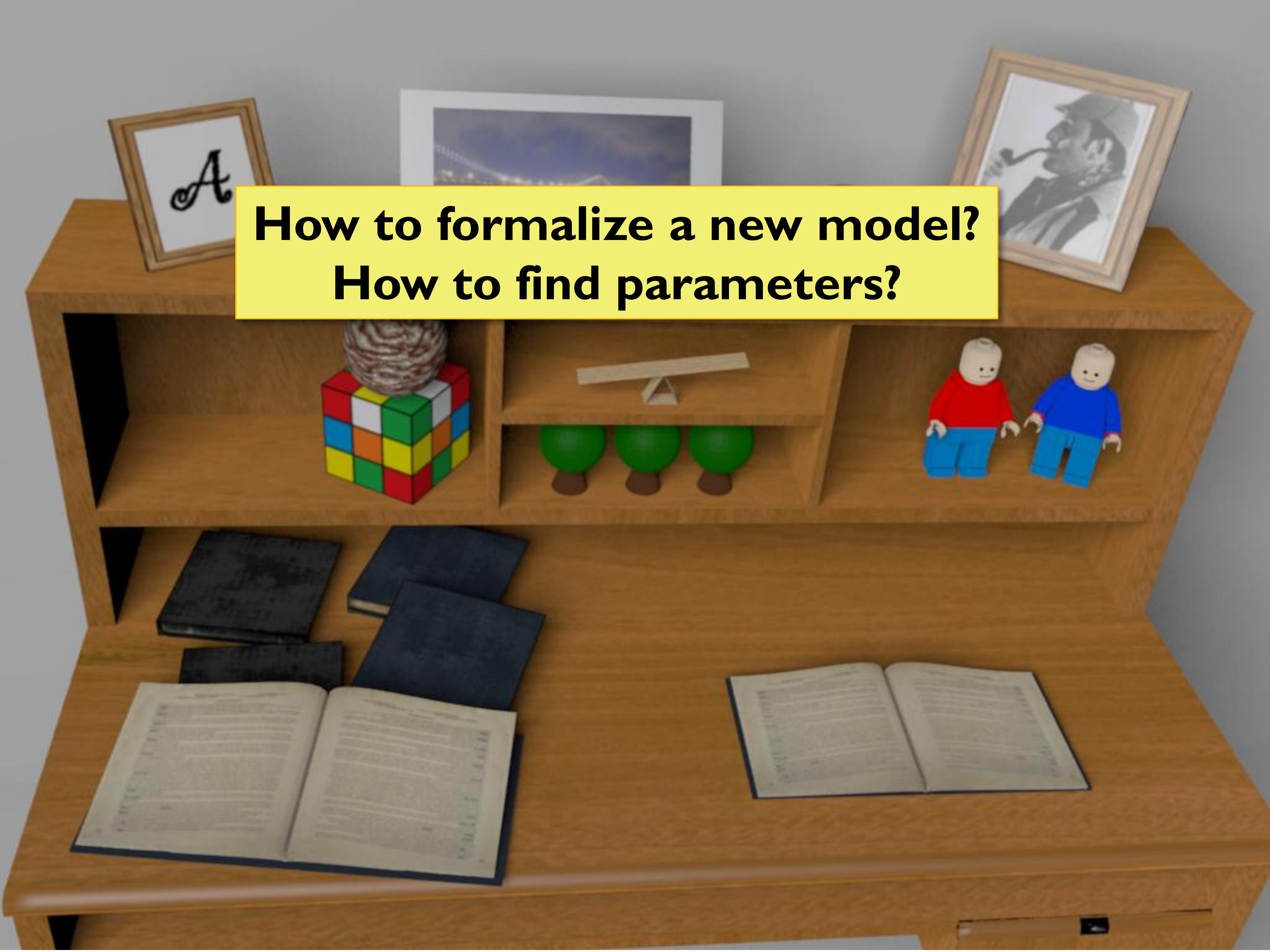


Modeling



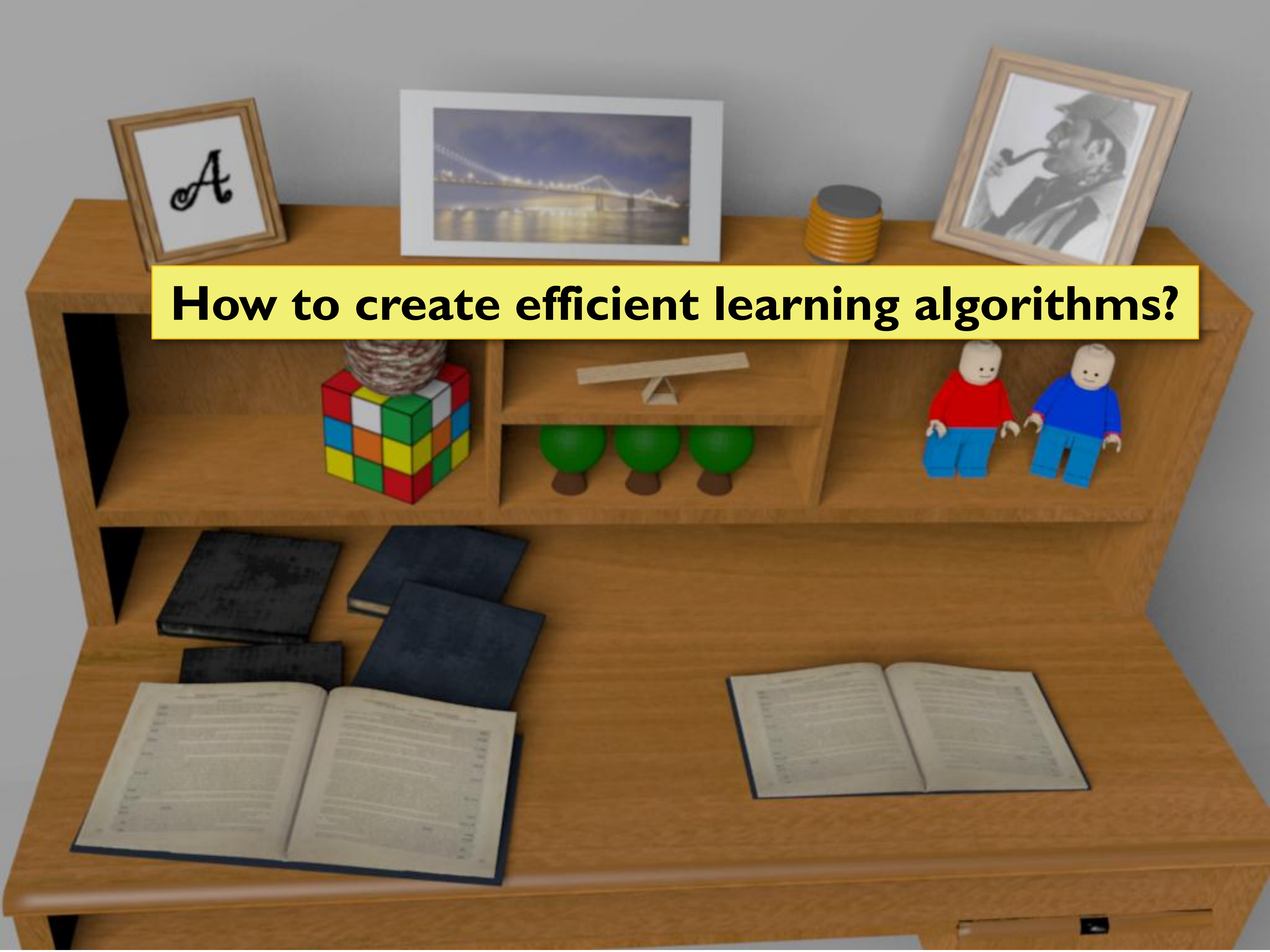
Why should a model, trained on one set of data, work well on future data?

When is it not the case?



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

How to create efficient learning algorithms?



How to handle structured data?

Unsupervised learning

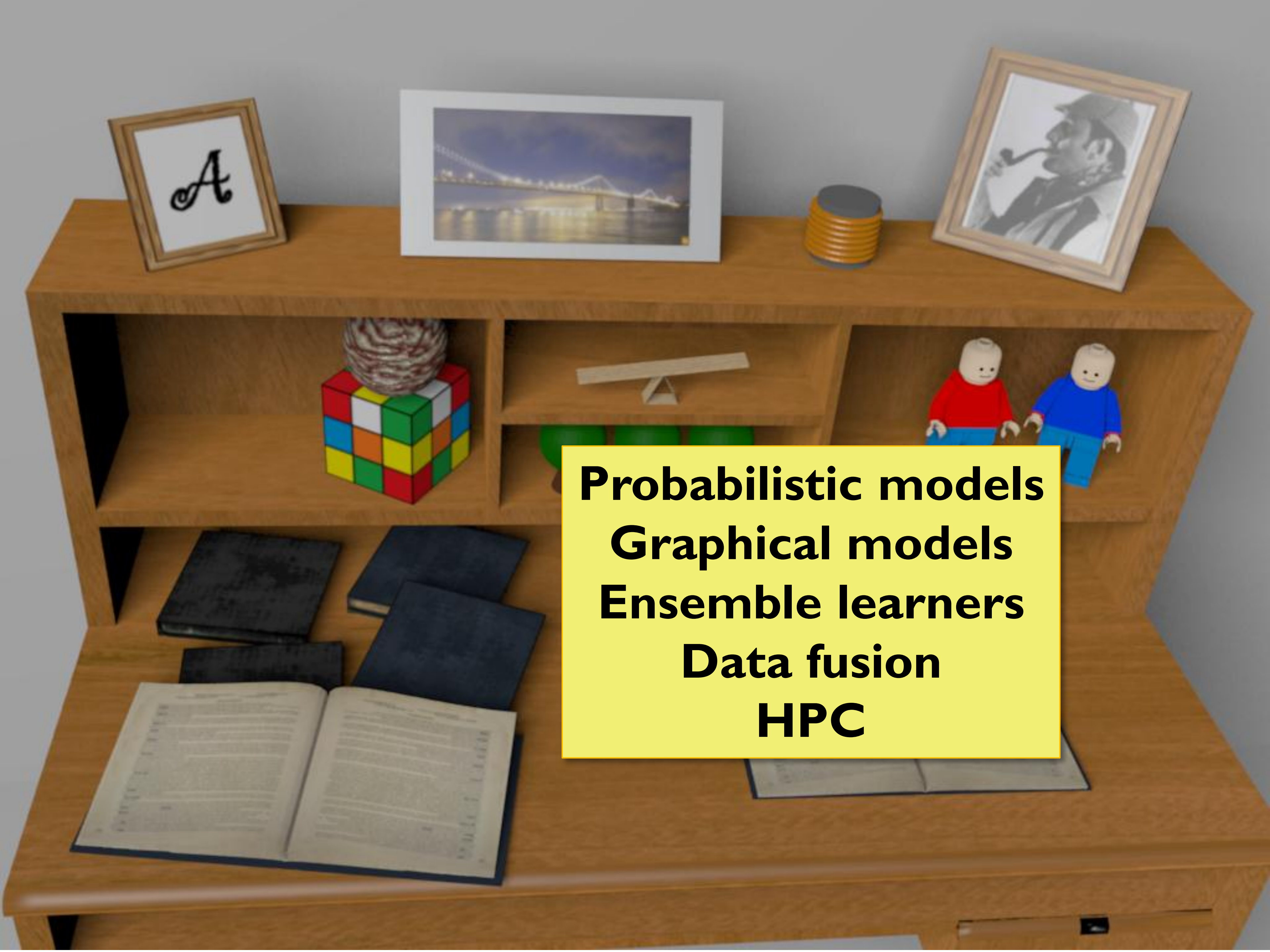
Semi-supervised learning

On-line learning

Active learning

Multi-instance learning

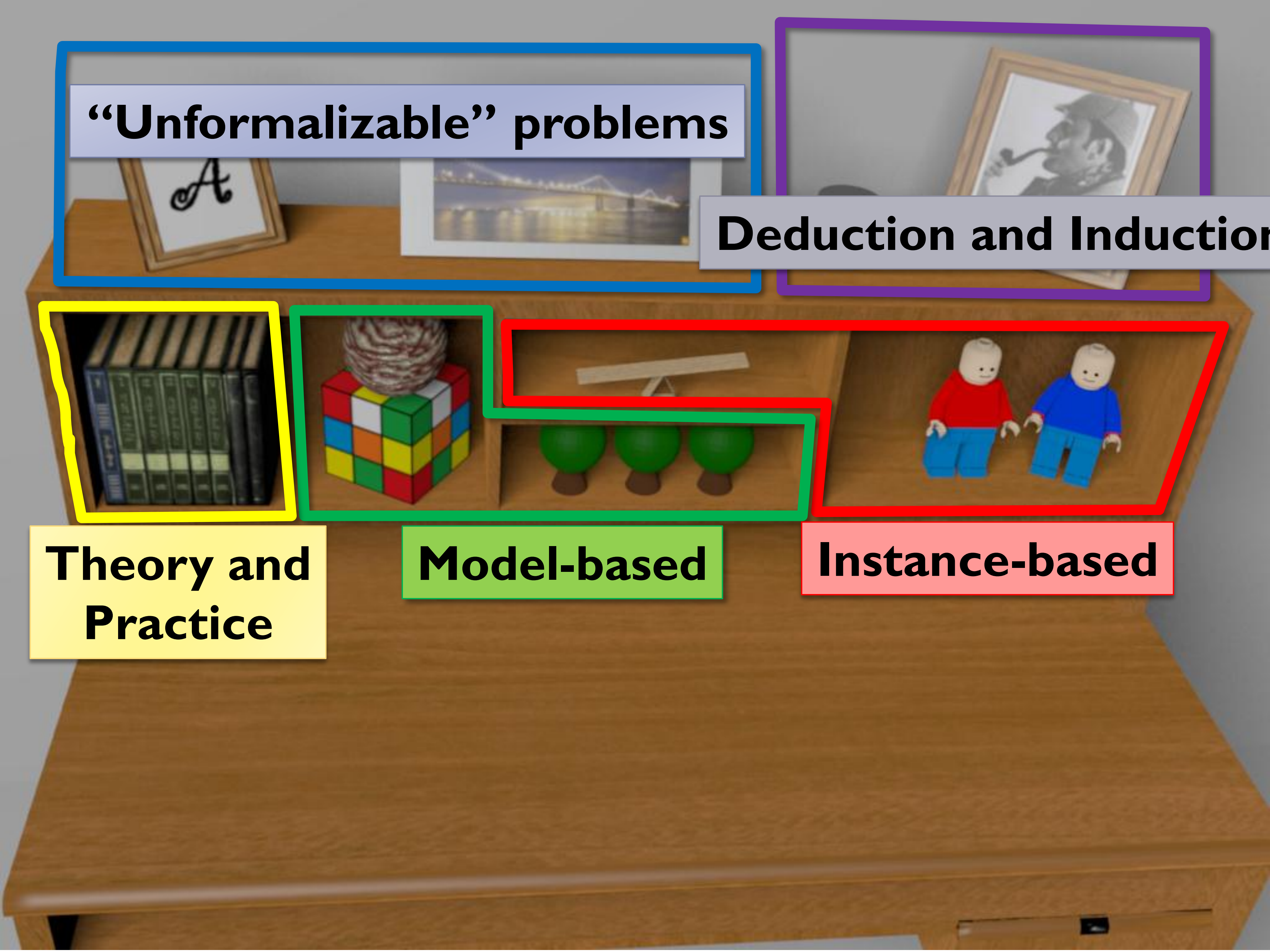
Reinforcement learning



Probabilistic models
Graphical models
Ensemble learners
Data fusion
HPC

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

A



“Unformalizable” problems

Deduction and Induction

Theory and Practice

Model-based

Instance-based

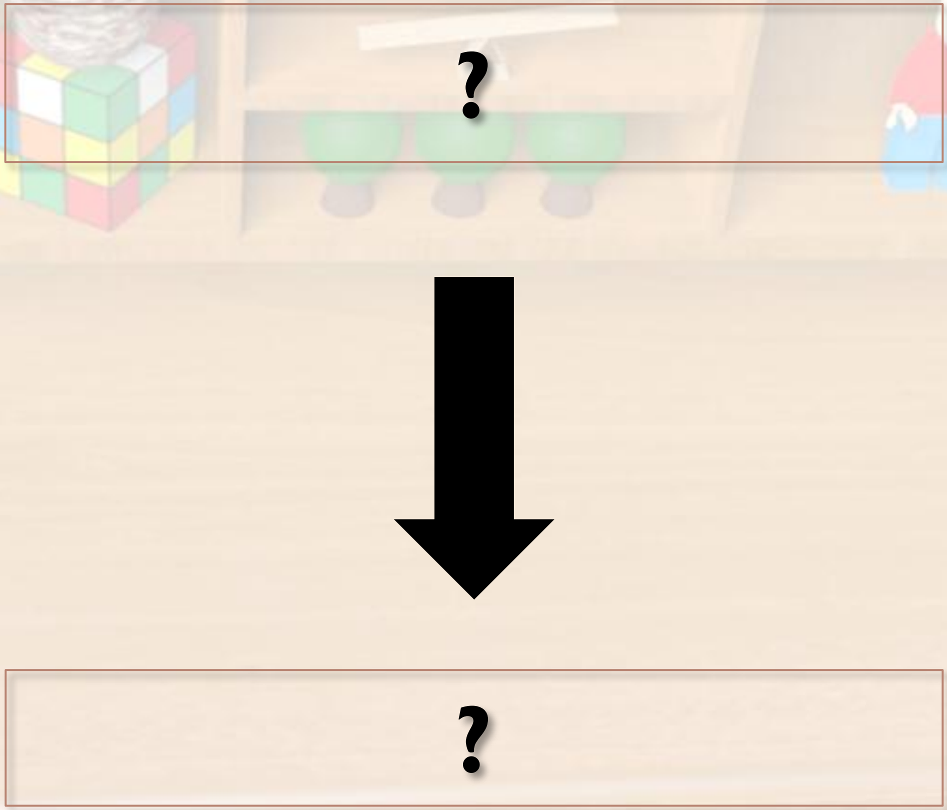
Quiz

- 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

$$\mathrm{argmin}_{\boldsymbol{w}} \;\_\_\_\_\_\_\_\_\_\_\_\_ + \lambda \;\_\_\_\_\_\_\_\_\_\_\_\_\_\_$$

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?