https://algs4.cs.princeton.edu

MULTIPLICATIVE WEIGHTS

- *experts problem*
- *elimination method*
- *multiplicative weights update*
- *algorithms in machine learning*
- *fraud detection*

https://algs4.cs.princeton.edu

Multiplicative Weights

- *experts problem*
- *elimination method*
- *multiplicative weights update*
- *algorithms in machine learning*
- *fraud detection*

Experts problem

Expert. An entity that makes binary predictions. ← *person, agent, sensor, algorithm…*

Binary prediction. Forecast on a binary outcome. ← *e.g., will it rain in Princeton tomorrow? Will the S&P 500 go up tomorrow?*

Experts problem. A collection of n experts make predictions over T days.

- On day $0 \le t < T$, each expert makes a prediction for the next day. After observing them, you make your own.
- On day $t + 1$ you see the actual outcome.

Goal. Minimize the number of incorrect predictions.

(Under some assumption on experts.)

Remark. We use 0 and 1 (not booleans) as labels. ← *generalizes to k-ary outcomes*

Experts problem

Experts problem. A collection of n experts make predictions over T days.

- On day $0 \leq t < T$, each expert makes a prediction for the next day.
  After observing them, you make your own.
- On day $t + 1$ you see the actual outcome.

Goal. Minimize the number of incorrect predictions.
(Under some assumption on experts.)

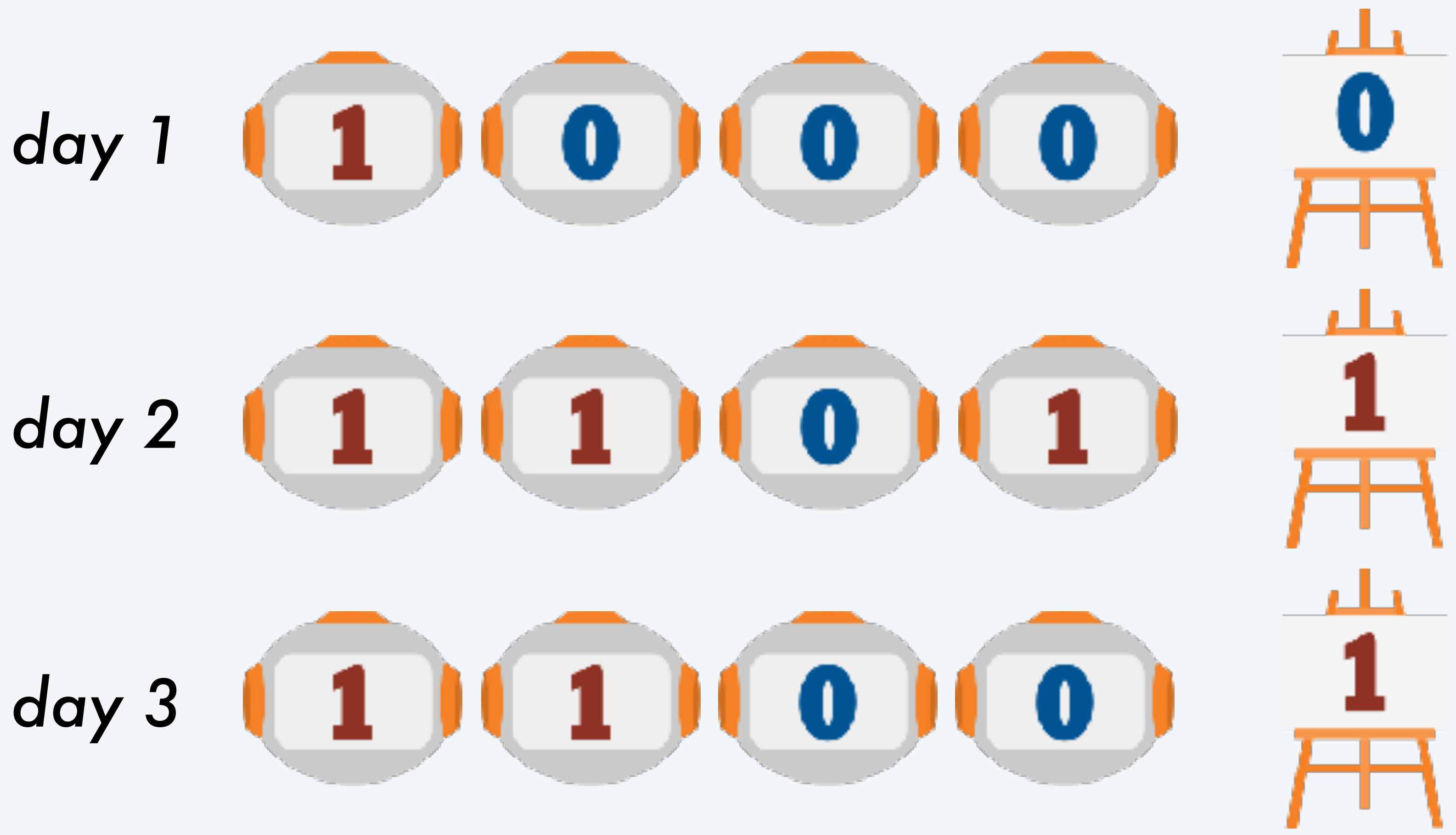
Experts problem

Experts problem. A collection of n experts make predictions over T days.

- On day $0 \leq t < T$, each expert makes a prediction for the next day.
  After observing them, you make your own.
- On day $t+1$ you see the actual outcome.

Goal. Minimize the number of incorrect predictions.
(Under some assumption on experts.)

Example. $n = 4$ experts, $T = 3$ days.

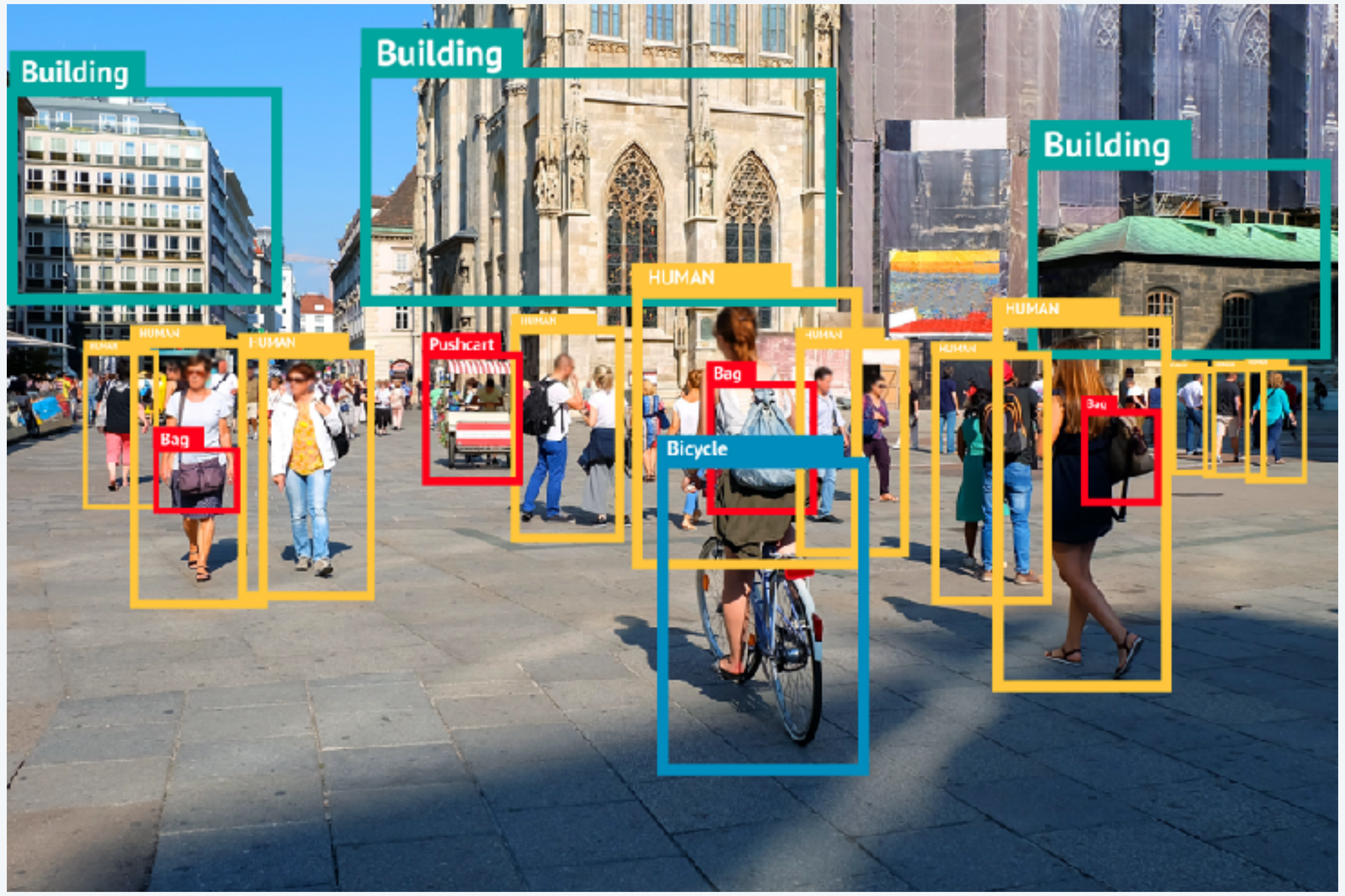


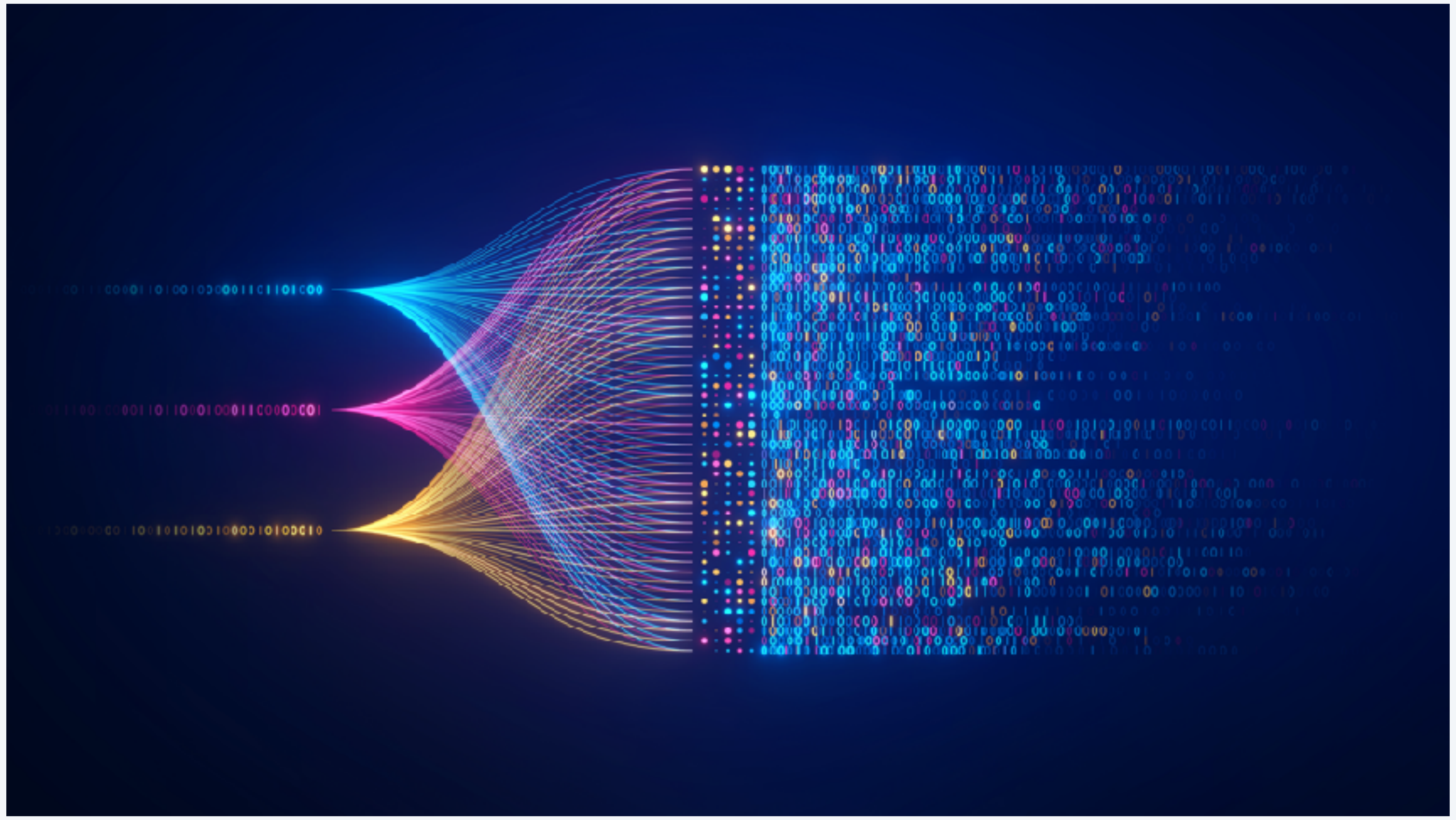
Context

Machine learning paradigm. Make predictions based on data/observations. [more on this later]

e.g., predictions from domain experts

Critical technology present in virtually all modern computing systems.





https://algs4.cs.princeton.edu

Multiplicative Weights

- experts problem
- elimination method
- multiplicative weights update
- algorithms in machine learning
- fraud detection

The perfect expert

Initial assumption. There is a perfect expert, who always predicts the actual outcome.

Elimination algorithm

On day t:

- **remove all experts that predicted incorrectly on day $t - 1$.**
- **make the remaining experts' majority prediction, tie-breaking for 0.**

Remarks.

- No assumption on the remaining $n - 1$ experts.
- No information about which experts are perfect.
- There may be more than one perfect expert.

The perfect expert

Initial assumption. There is a perfect expert, who always predicts the actual outcome.

Elimination algorithm

On day t:

- **remove all experts that predicted incorrectly on day $t-1$.**
- **make the remaining experts' majority prediction, tie-breaking for 0.**

Proposition. The elimination algorithm makes $\leq \log_2 n$ mistakes.

Pf.

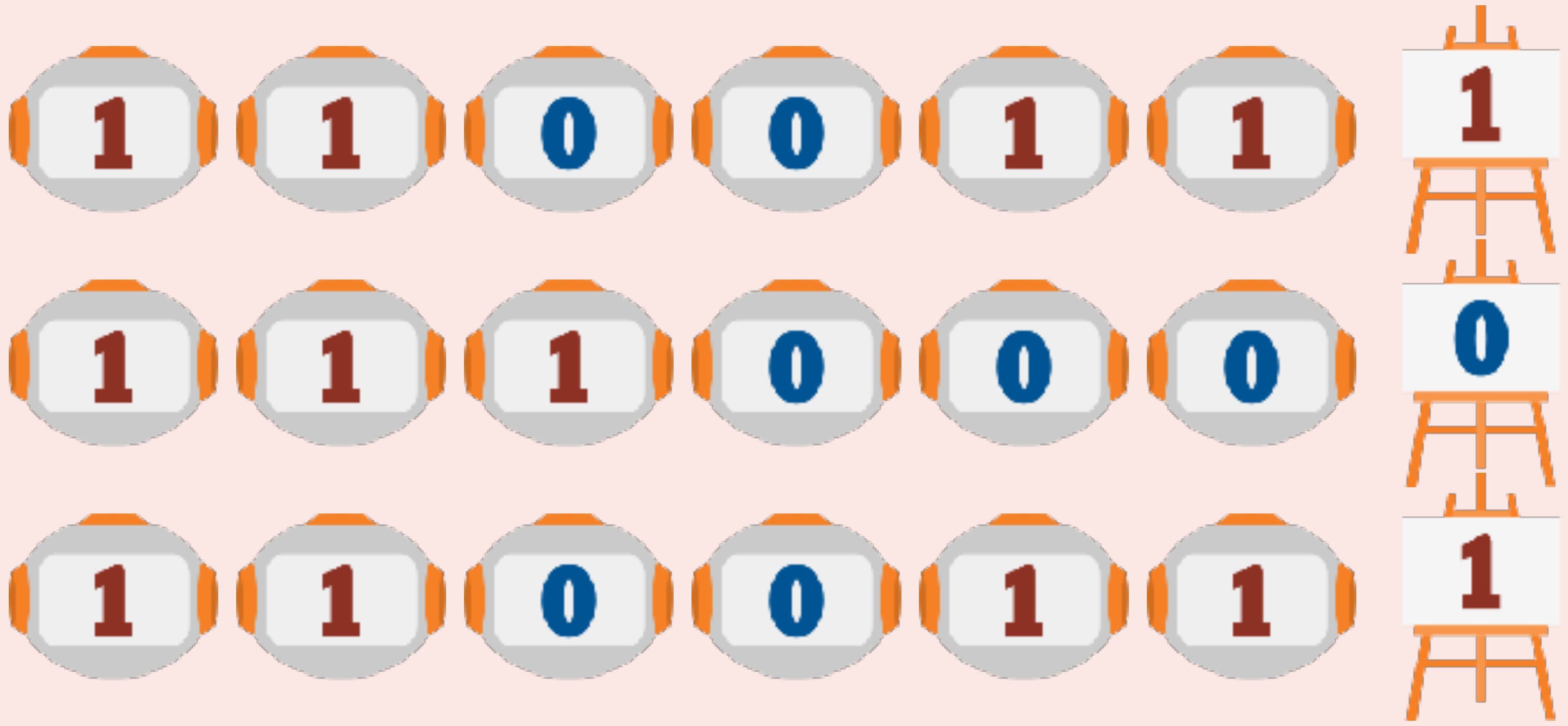
- Suppose algorithm makes a mistake on a certain day.
- Majority predicted incorrectly $\Longrightarrow$ at least half of remaining experts removed on next day.
- Halving $\log_2 n$ times yields a single (perfect) expert left. ▪

in general, $\leq \log_2(n/p)$ mistakes with p perfect experts.
Same analysis as binary search!

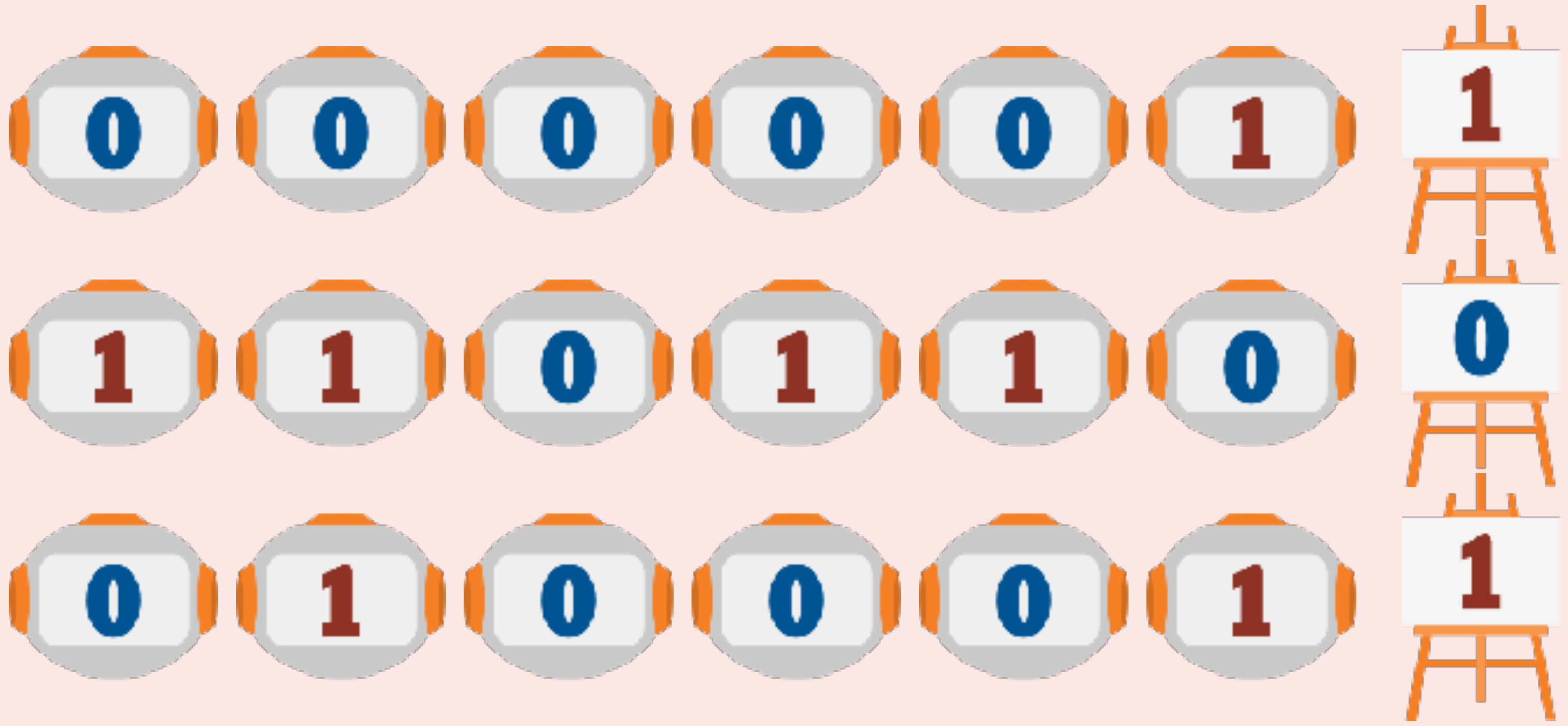
Multiplicative Weights: quiz 1

Which of the following examples causes the elimination method to make the most mistakes?

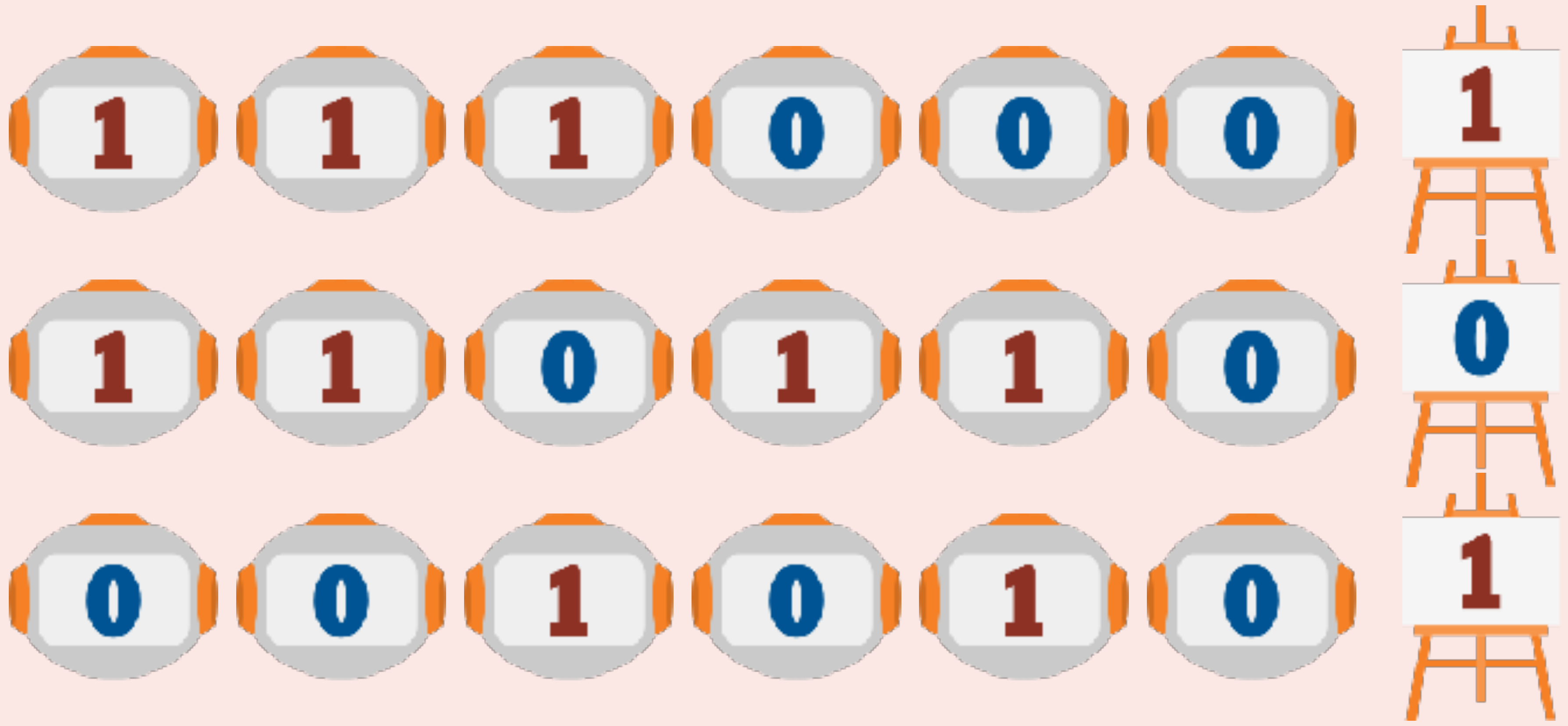
A.



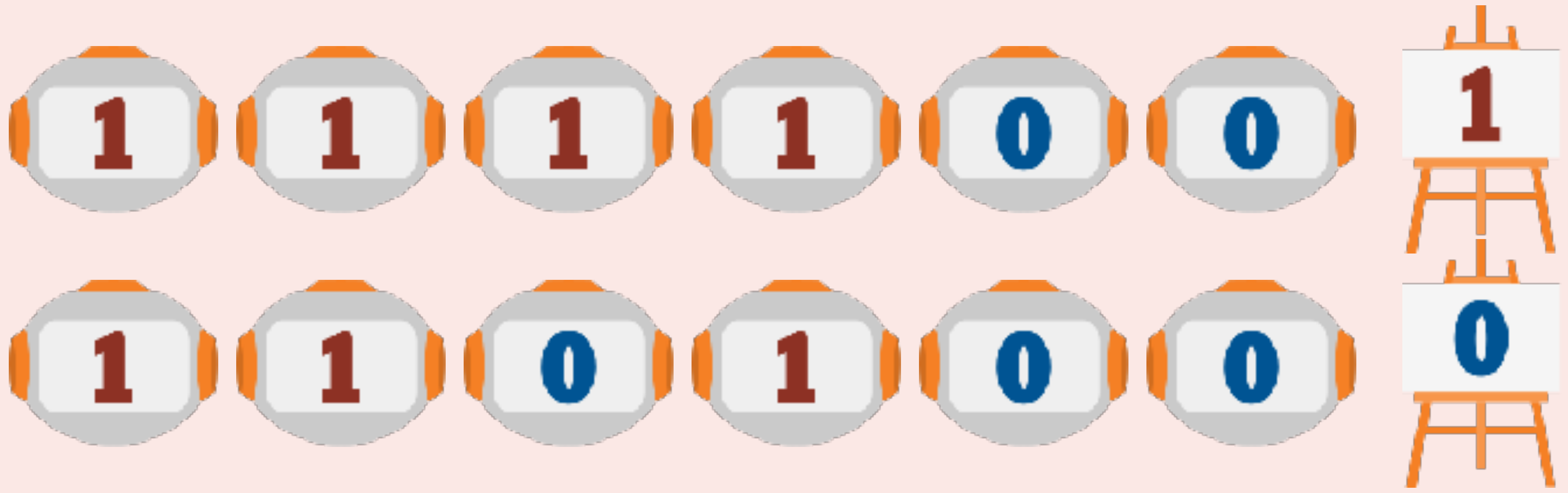
C.



B.



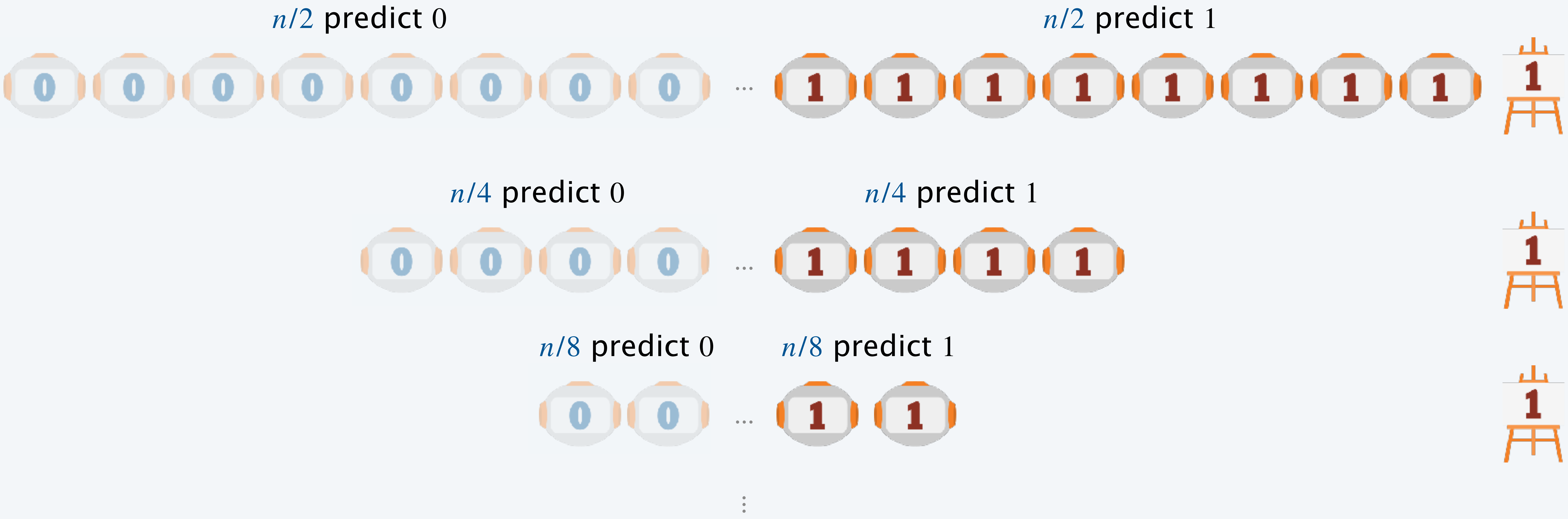
D.



Lower bound on the number of mistakes

Proposition. The elimination algorithm makes $\log_2 n$ mistakes in the worst case.

Pf sketch. Generalize solution to quiz. Assume $n = 2^k$ for some k, then



Elimination Algorithm Game

You be the expert now. Let’s play a game!

Algorithms

Robert Sedgewick | Kevin Wayne

https://algs4.cs.princeton.edu

Multiplicative Weights

- experts problem
- elimination method
- multiplicative weights update
- algorithms in machine learning
- fraud detection

A more realistic scenario

Issue. Unrealistic to expect a perfect expert; initial assumption is too strong.

Weaker assumption. The best expert makes at most M mistakes.

Modified elimination algorithm

On day t:

- **remove all experts that predicted incorrectly on day $t-1$.**
- **make the remaining experts' majority prediction, tie-breaking for 0.**
- **if no experts left, add all n of them back.**

Proposition. The modified elimination algorithm makes at most $(M+1)(1+\log_2 n) \sim M \log_2 n$ mistakes.

Pf sketch. Same as original proof, but repeat $M+1$ times.

The Multiplicative Weights method

Weaker assumption. The best expert makes at most M mistakes.

Intuition. Throwing away an expert is too harsh.

Instead, assign "confidence" to each expert and lower it after a mistake.

Multiplicative Weights algorithm

Initialize a `weights` array of doubles with ones.

On day t:

- **halve the weight of experts that predicted incorrectly on day $t-1$.**
- **let `zeroWeight` be the sum of weights of experts predicting 0 and `oneWeight` be the sum of weights of experts predicting 1.**
- **predict 0 if `zeroWeight` $\geq$ `oneWeight` and 1 otherwise.**

A `MultiplicativeWeights` class

```
public class MultiplicativeWeights {
  private int n;
  private double[] weights;

  public MultiplicativeWeights(int n) {
    this.n = n;
    weights = new double[n];
    for (int i = 0; i < n; i++) weights[i] = 1.0;
  }

  public int predict(int[] expertPredictions) {
    double zeroWeight = 0, oneWeight = 0;
    for (int i = 0; i < n; i++) {
      if (expertPredictions[i] == 0) zeroWeight += weights[i];
      else                           oneWeight  += weights[i];
    }

    if (zeroWeight >= oneWeight) return 0;
    else                         return 1;
  }

  public void seeOutcome(int actualOutcome, int[] expertPredictions) {
    for (int i = 0; i < n; i++)
      if (expertPredictions[i] != actualOutcome) weights[i] /= 2;
  }
}
```

initialize weights with ones

calculate weights of experts predicting 0 and 1

halve weights of wrong experts

Multiplicative Weights: quiz 2

What is the `weights` array, under multiplicative weights, after the following predictions and observations?

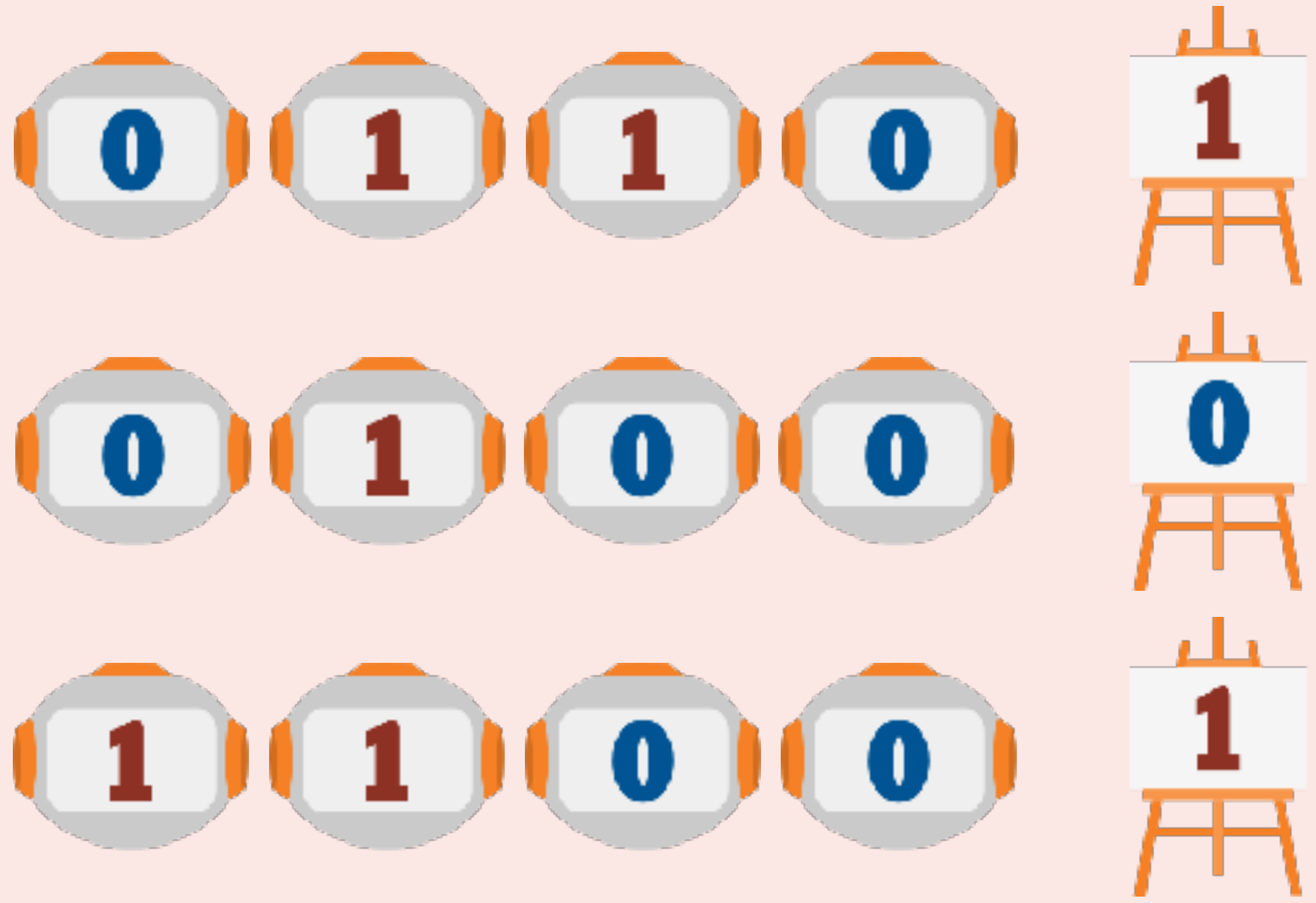
A. [1, 1/2, 1/2, 1/4]

B. [1, 1/4, 1/4, 1/2]

C. [1, 2, 4, 1]

D. [1/2, 1/2, 1/2, 1/4]

E. [1, 1/2, 1/4, 1/2]



Multiplicative Weights analysis

Proposition. The multiplicative weights method makes at most $2.41(M + \log_2 n)$ mistakes.

Pf.

- Let W_t be the sum of all weights at start of day t.
  In particular, $W_0 = n$.
- If MW algorithm makes a mistake on day t, then $W_{t+1} \le \frac{3}{4} W_t$.
  - A mistake requires $\ge W_t/2$ weight making incorrect prediction.
  - Since this weight is halved, $W_{t+1} \le W_t - W_t/4 = \frac{3}{4} W_t$.
- Calling m total number of mistakes (after day T),

each mistake multiplies by 3/4

$$W_T \le \left(\frac{3}{4}\right)^m W_0 = \left(\frac{3}{4}\right)^m n.$$

- Since best expert makes $\le M$ mistakes, its weight is $\ge 1/2^M$.
  In particular, $W_T \ge 1/2^M$.

divide by n and invert *take $\log_2$ on both sides* $\frac{1}{\log_2(4/3)} \le 2.41$

$$\left(\frac{1}{2}\right)^M \le W_T \le \left(\frac{3}{4}\right)^m n \implies \left(\frac{4}{3}\right)^m \le 2^M \cdot n \implies m \le \frac{1}{\log_2(4/3)} \cdot (M + \log_2 n) \quad \blacksquare$$

How good is this guarantee?

Rate of mistakes. Ratio $\dfrac{M}{T}$ when T goes to infinity (but n is fixed).

Example. Best expert makes a mistake on $10\,\%$ of the days: $M = \dfrac{T}{10}$.

Then multiplicative weights has rate

$$\frac{2.41\left(\frac{T}{10} + \log_2 n\right)}{T} \quad = \quad 0.241 + \frac{\log_2 n}{T} \quad \xrightarrow{T\to\infty} \quad 24.1\,\%\,.$$

Remark. Best possible bound on number of mistakes is $2M$. ⟵ *e.g. $n = 2$, experts always disagree and outcomes alternate every day*

Algorithmic framework

Historical context. The multiplicative weights method was rediscovered in multiple areas of computer science to solve many seemingly different problems. ⟵ *first known version proposed in the 50s to solve zero-sum games*

Applications.

- Machine Learning: boosting algorithms [experts are input examples: stay tuned!]
- Optimization: solving linear and semi-definite programs [experts are constraints]
- Maximum flow: efficient algorithms [experts are paths]
- Game theory: solving zero-sum games [experts are pure strategies]

Experts problem for K-ary predictions

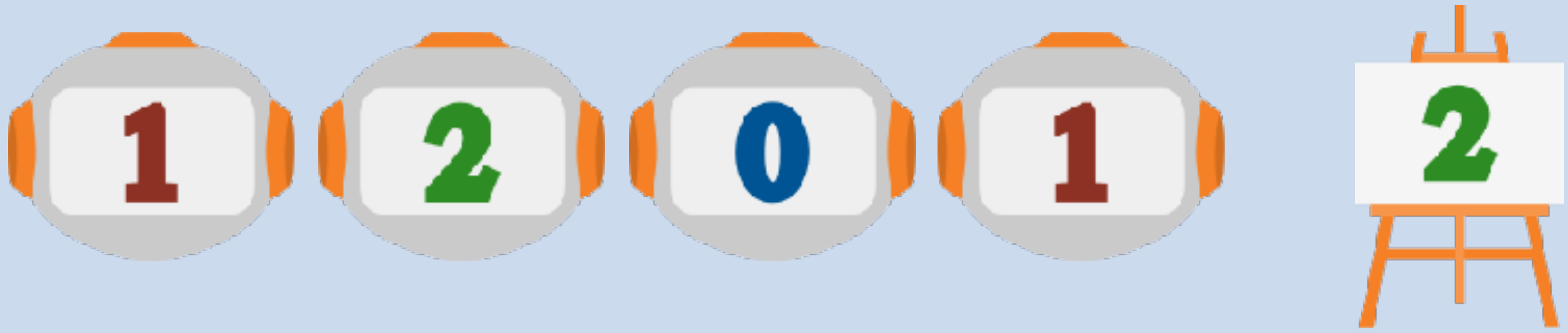
Expert. Entity that makes a k–ary prediction.

k–ary prediction. Forecast on a k–ary outcome (an integer $< k$). ⟵ *e.g. will maximum temperature tomorrow be ≤ 40, or 40 to 45, or 45 to 50, or > 50 degrees?*

k–ary elimination algorithm

On day t:

- **remove all experts that predicted incorrectly on day $t - 1$.**
- **make the remaining experts' most popular prediction.**



Multiplicative Weights: quiz 3

Which is the best upper bound on the number of mistakes of the k-ary elimination algorithm?

A. $k \log_2 n$

B. $\log_k n$

C. $\log_2 n$

D. $k + \log_2 n$

https://algs4.cs.princeton.edu

Multiplicative Weights

- *experts problem*
- *elimination method*
- *multiplicative weights update*
- ***algorithms in machine learning***
- *fraud detection*

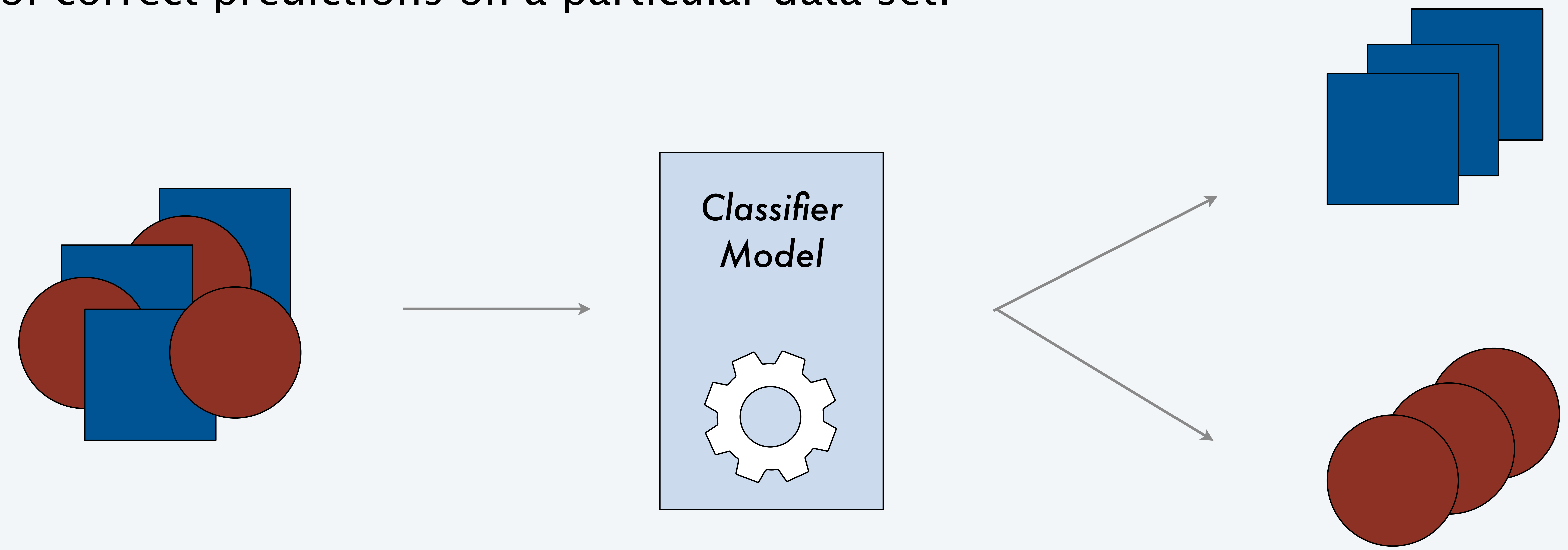
(Binary) Classification problem

Input. A training data set of items with a binary label for each. ← *e.g., medical images labeled healthy or ill; emails labeled spam or not spam*

Goal. Predict labels of new items.

- Phase 1: Create a model based on the training data. ← *algorithm that makes predictions*
- Phase 2: Apply the model to new item to predict its label.

Accuracy. Fraction of correct predictions on a particular data set.



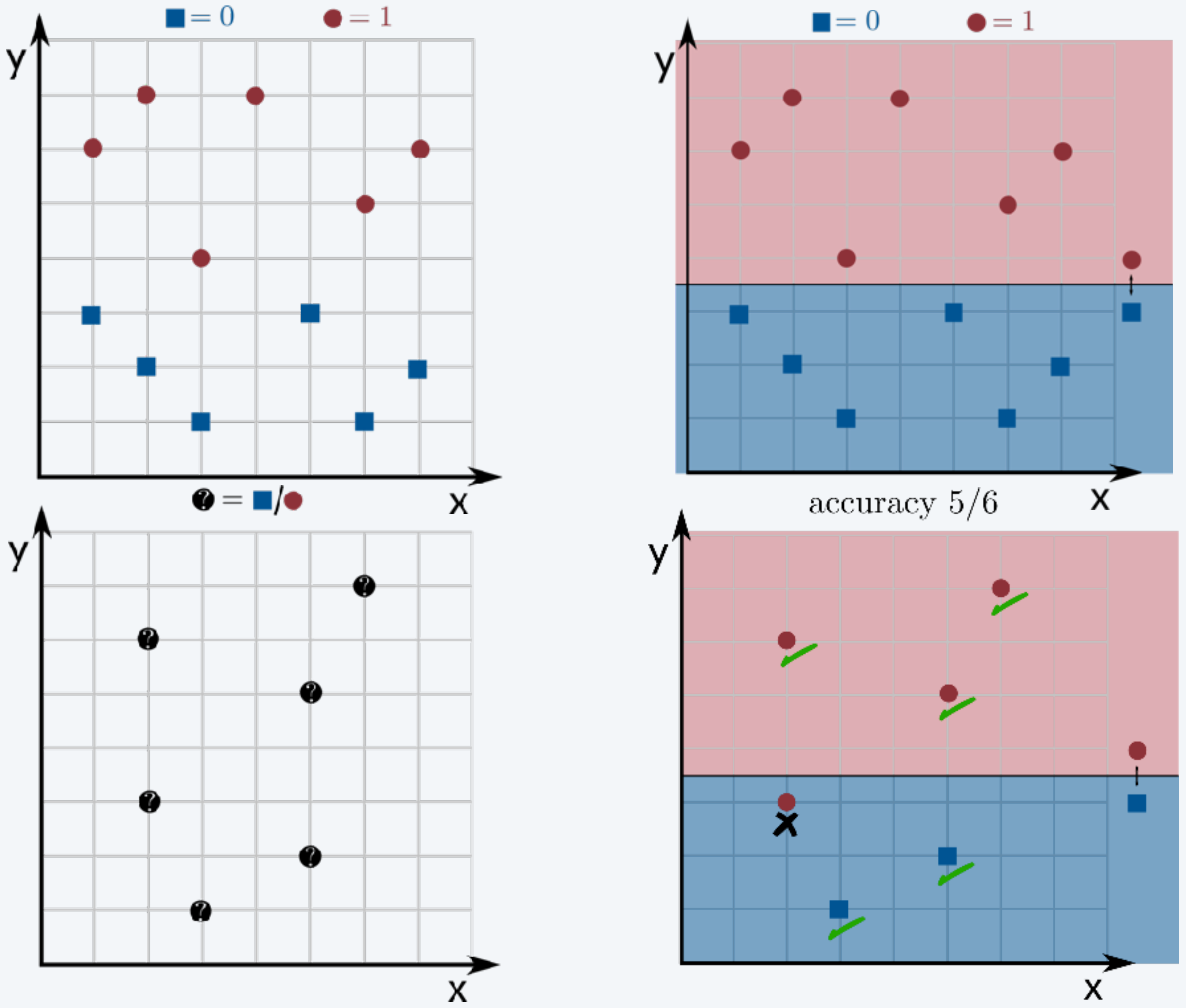
(Binary) Classification problem

Assumption. Items are points in D dimensions.

Example. Data set with 12 points and D = 2 dimensions.

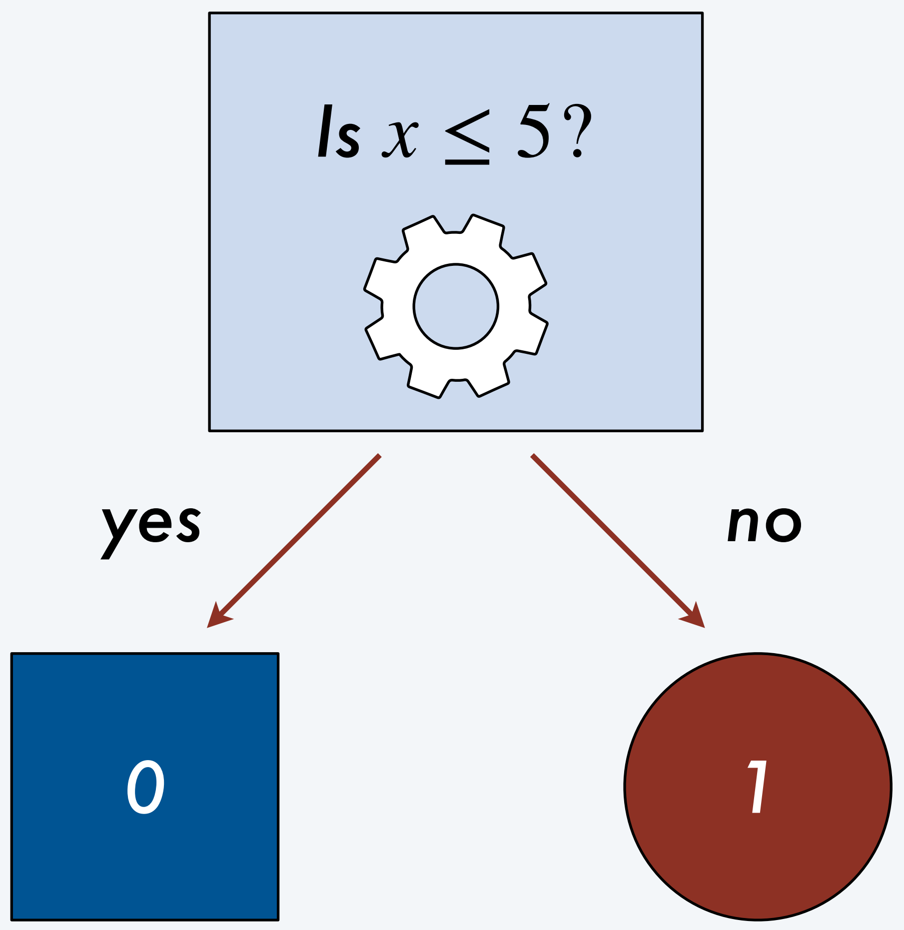
training set

test set

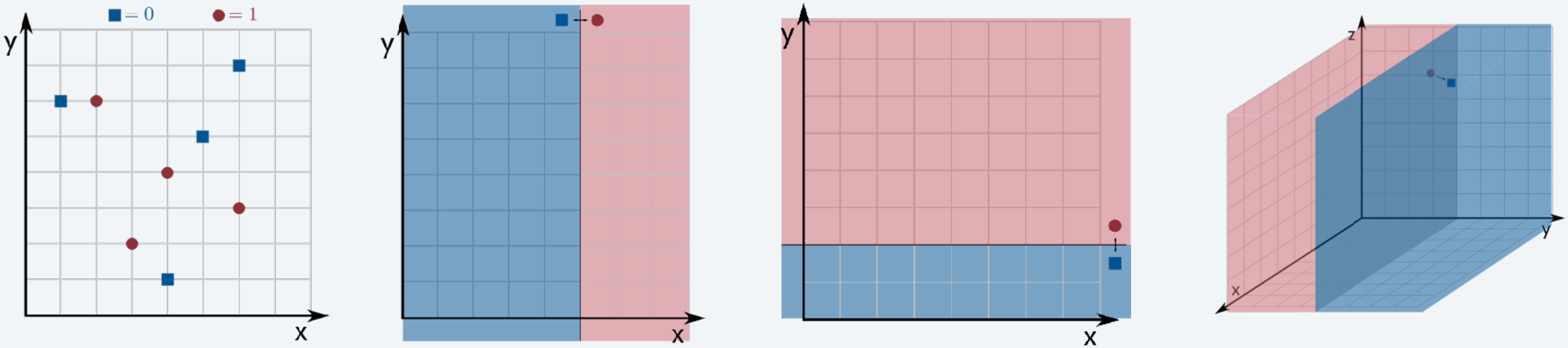


Decision stump

Def. A decision stump is a simple classifier model that predicts from a single dimension.
Points with coordinate $\leq$ value predictor receive a label;
points with coordinate $>$ value predictor receive the other.



Training. Finding decision stump that maximizes accuracy on training data.



Remark. Decision stumps are examples of weak learners, models that perform marginally better than random.

Multiplicative weights and boosting: AdaBoost

Def. A boosting algorithm is any that combines weak learners into strong ones.

AdaBoost. Use input points as experts and train decision stumps to find "expert" predictions.

Simplified AdaBoost algorithm

Training:

Initialize a length-n `weights` array of doubles with $1/n$.

Repeat T times: ← *parameter we can tune*

- **train decision stump on input points weighted by `weights`.**
- **double weight of points labeled incorrectly.**
- **normalize weights.** ← *prevents overflow*

Prediction: output majority prediction over all stumps.

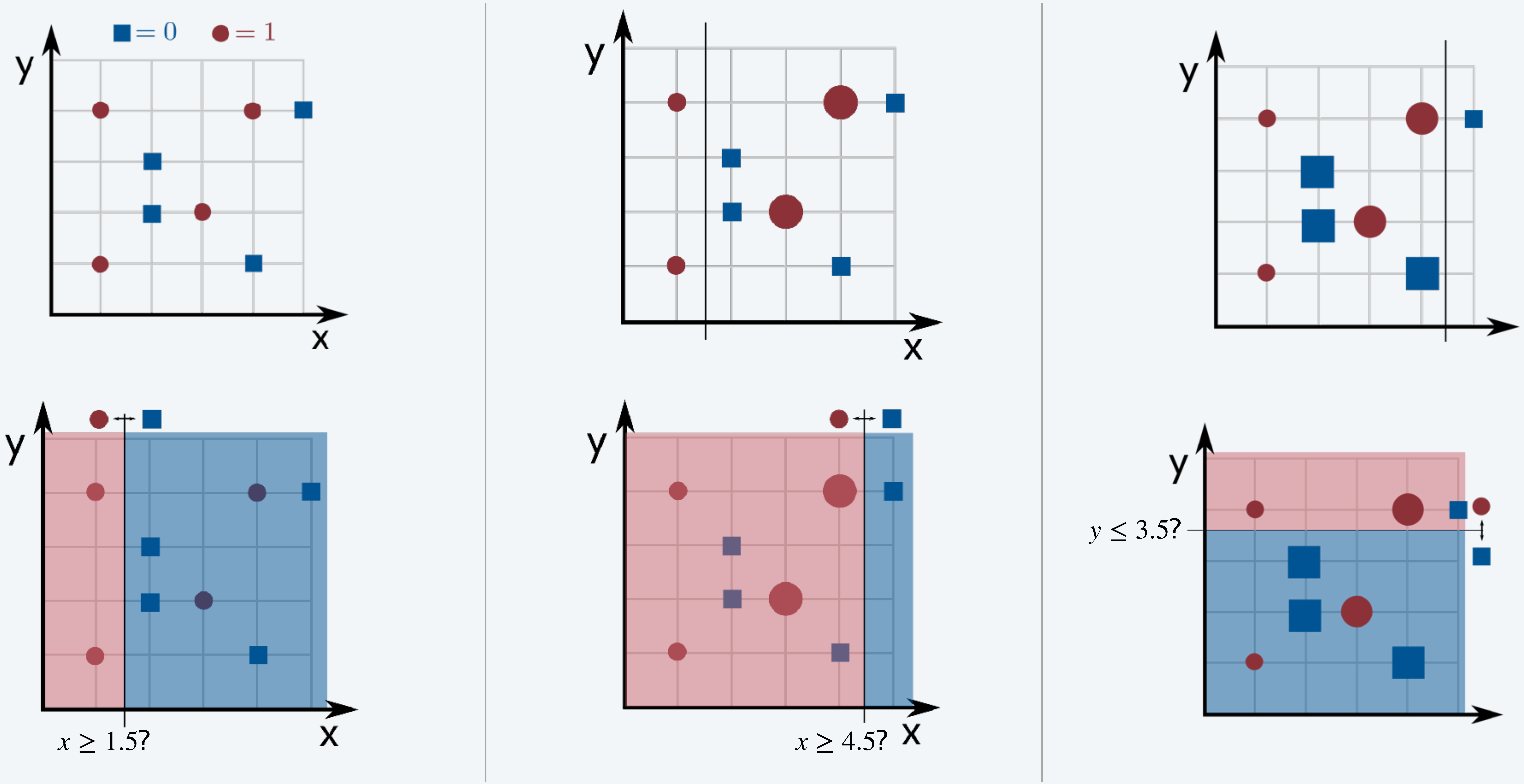
Remark. AdaBoost multiplies weights by a factor that depends on weak learner's error and gives preference to better decision stumps.

Yoav Freund

Robert Schapire

↑ *former COS 226 instructor!*

Simplified AdaBoost demo



https://algs4.cs.princeton.edu

Multiplicative Weights

- *experts problem*
- *elimination method*
- *multiplicative weights update*
- *algorithms in machine learning*
- ***fraud detection***

Assignment 7: Fraud detection

Assignment will be released today.

Start early!

Read Ed post carefully and watch helper video before your precept.

Credits

image	source	license
Object Classification	Adobe Stock	education license
Data Visualization	Adobe Stock	education license
Baloons	Adobe Stock	education license
Yoav Freund	UC San Diego	
Robert Schapire	Microsoft Research	

