https://algs4.cs.princeton.edu

Multiplicative Weights

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

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*

Experts problem

Expert. Some person/agent/sensor/algorithm that makes binary predictions (0 or 1).

Binary prediction. A forecast on a binary outcome, e.g. Is it going to rain tomorrow? Is the S&P 500 going up tomorrow?

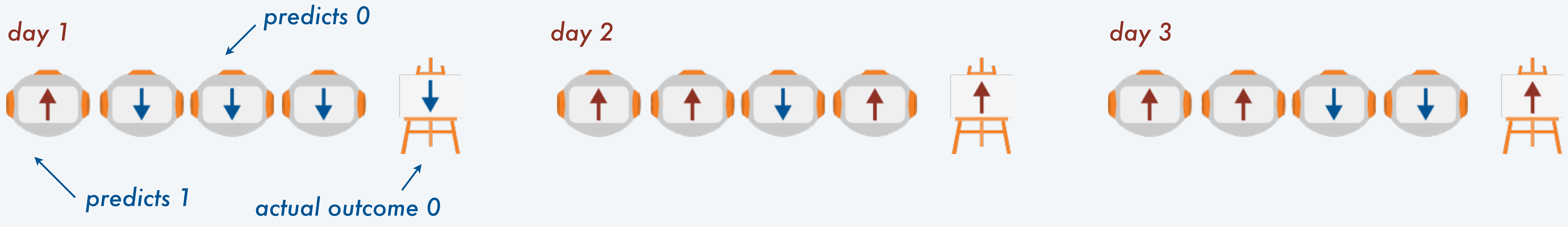
we will use 0 and 1 as the possible outcomes, e.g. no rain = 0 and rain = 1

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

- On day t, you get to observe the prediction of each expert to make your own.
- On day $t + 1$ you see the actual outcome (e.g. did the S&P 500 actually go up?).
- **Goal:** minimize the number of mistakes, i.e. incorrect predictions.

need some assumptions on the experts, e.g. if experts are random we can't learn anything from them

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

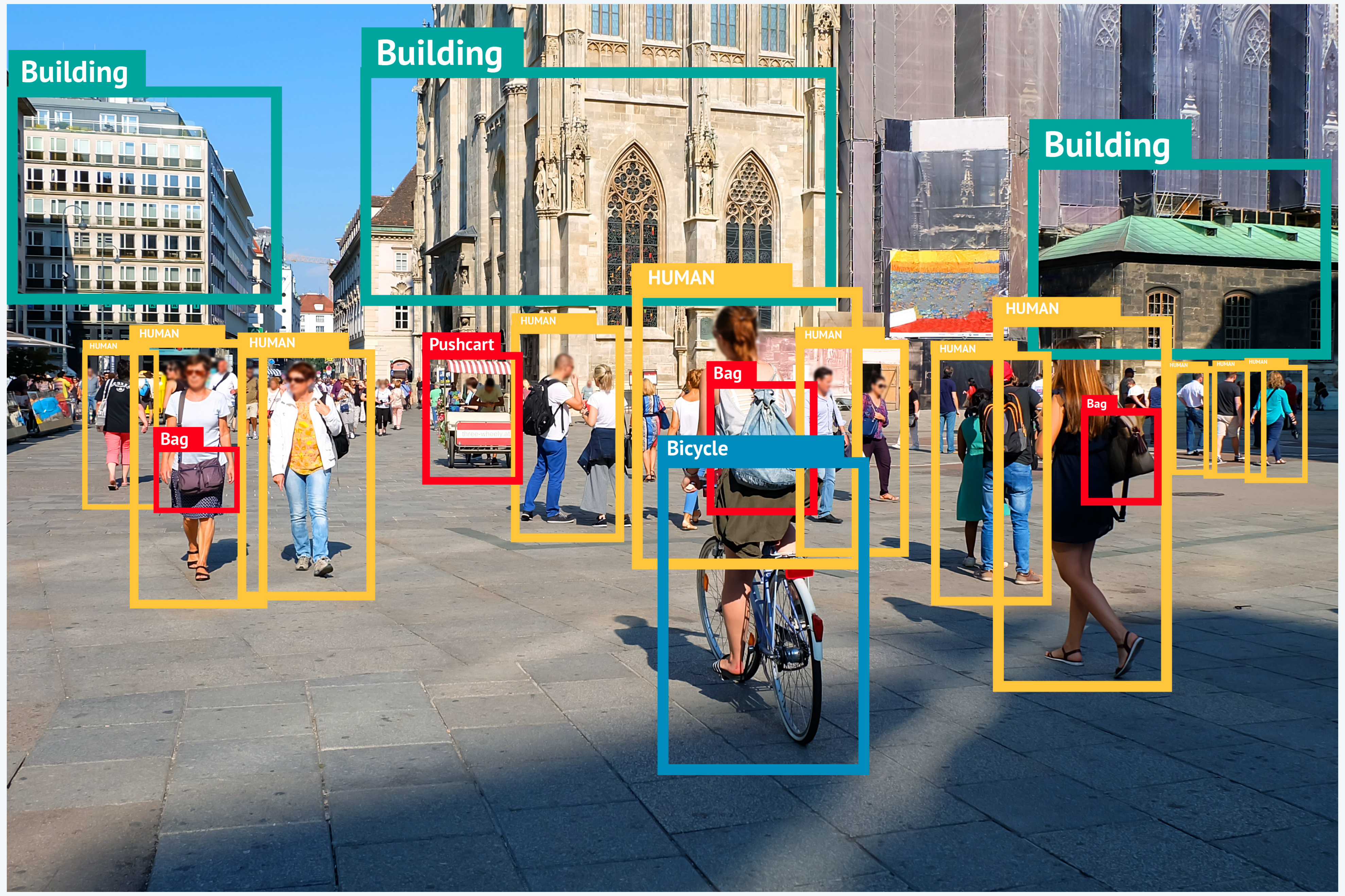


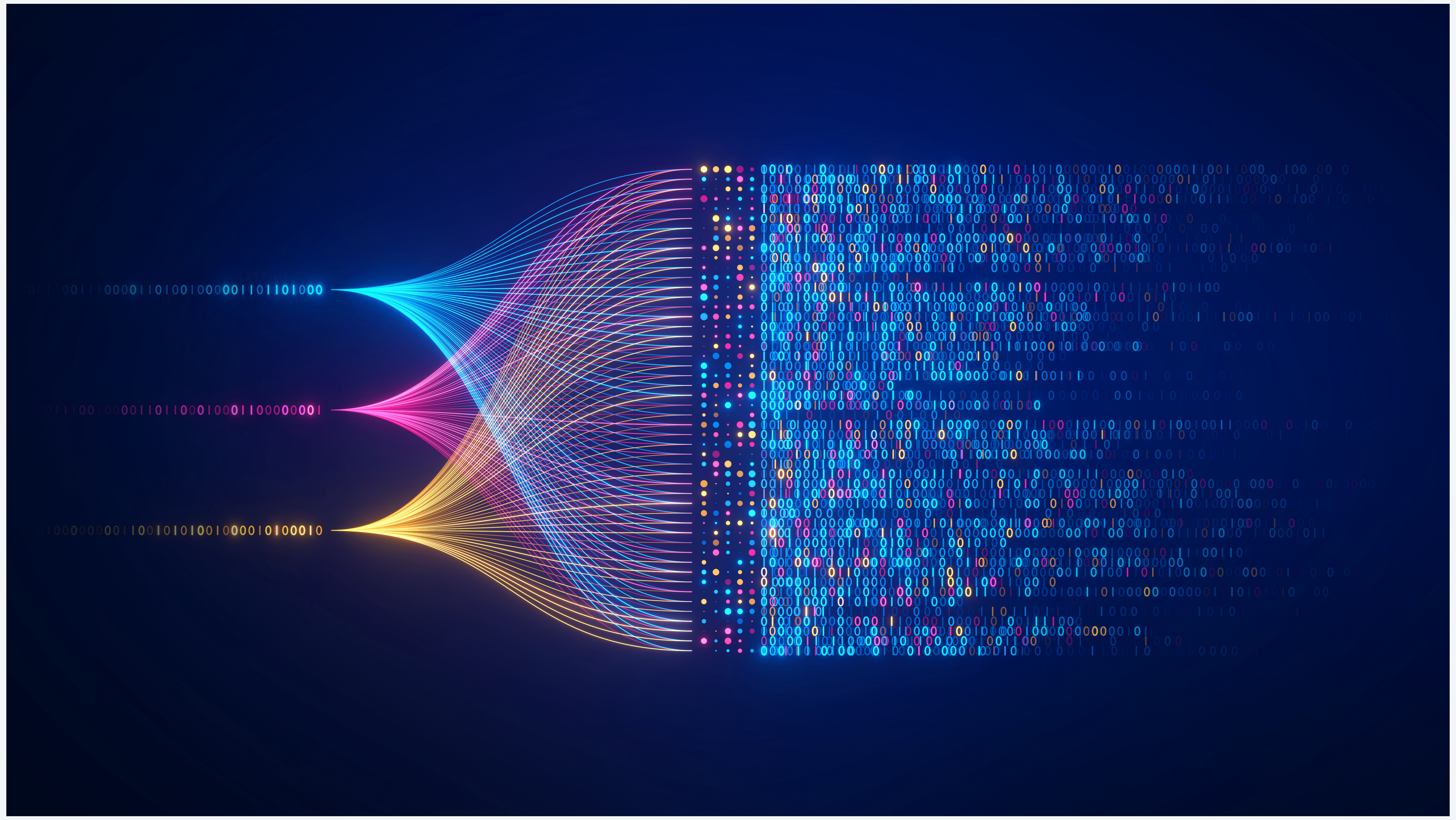
Context

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

such as predictions from experts and actual outcomes

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

New assumption. There is some expert that is perfect, i.e. they always predict the right outcome.

- No assumptions on what the remaining n - 1 experts do.
- You don't know which expert is the perfect one.

Elimination algorithm

On each day:

- **take the majority prediction over the experts predictions**
- **after observing the actual outcome: remove all experts**

that predicted the wrong outcome

always tie break in favor of 0

Proposition. The elimination algorithm makes at most $\lfloor \log_2 n \rfloor$ mistakes.

Elimination algorithm

Elimination algorithm

On each day:

- **take the majority prediction over the experts predictions**
- **after observing the actual outcome: remove/ignore all experts that predicted the wrong outcome**

Proposition. The elimination algorithm makes at most $\lfloor \log_2 n \rfloor$ mistakes.

Pf.

Suppose the algorithm makes a mistake on a certain day.

Then a majority of the remaining experts also made a mistake ⇒ at least half of the experts removed.

So every time the algorithm makes a mistake it removes half of the remaining experts.

can happen at most $\lfloor \log_2 n \rfloor$ *times! Recall the binary search analysis*

Multiplicative Weights: quiz 1

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

reminder: always tie break in favor of 0

A.

B.

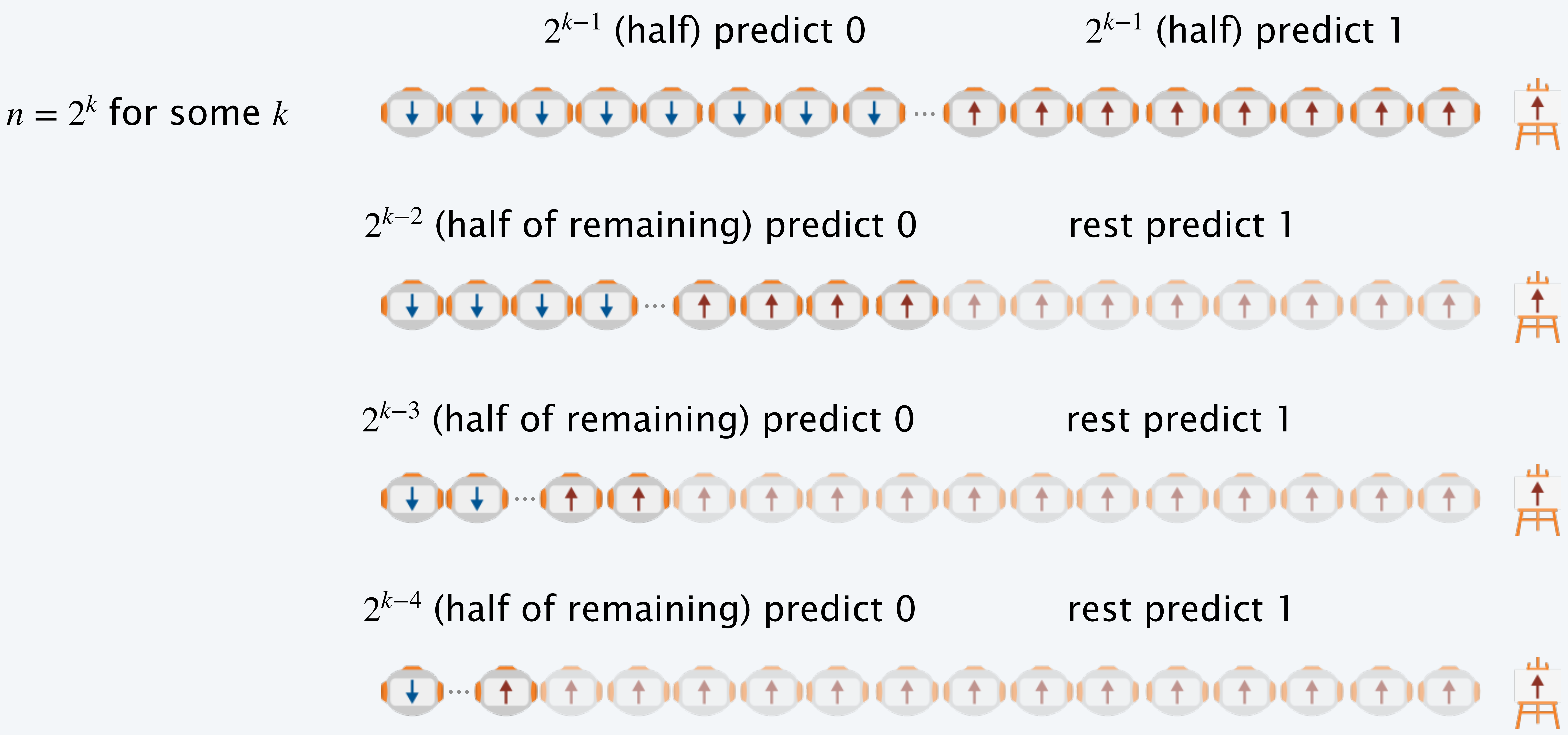
C.

D.

Lower bound on the number of mistakes

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

Pf. Generalize solution to previous quiz



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. It's not very realistic to assume that there is a perfect expert.

New assumption. The best expert makes at most M mistakes

Modified Elimination algorithm

On each day:

- **take the majority prediction over the experts predictions**
- **after observing the actual outcome: remove all experts that predicted the wrong outcome**
- **if all experts got removed, add them all back**

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

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

Can we do better?

Multiplicative weights method

New assumption. The best expert makes at most M mistakes

Intuition. Throwing away an expert is too harsh. Assign “confidence” to each expert and lower it after a mistake

Multiplicative Weights Method

Initialize a double[n] array called weights and set all values to 1

On each day:

- **let zeroWeight be the sum of weights of experts predicting 0**
- **let oneWeight be the sum of weights of experts predicting 1**
- **predict 0 if zeroWeight ≥ oneWeight, predict 1 otherwise**
- **after observing the actual outcome: halve the weight of all the experts that predicted incorrectly**

```java
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;
  }

  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 to 1

Calculate weight of experts prediction 0/1

Halve the weight of incorrect experts

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

Pf. [by observing the total weight reduces after a mistake]

Let W_t = *weights*[0] + *weights*[1] + … *weights*[n] at time t, i.e. sum of all weights at time t, so $W_0 = n$.

Claim. If we make a mistake at time t, then $W_{t+1} \leq \frac{3}{4} W_t$, so then new total weight goes down a factor of $\frac{3}{4}$.

If we made a mistake at least half of the weight made made a mistake, so we remove $\frac{1}{4}$ of the total weight.

So let's say we have made m mistakes at the end (time T), then using this claim m times:

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

Since the best expert makes at most M mistakes, then *weights*[best expert] = $\left(\frac{1}{2}\right)^M$

Given that W_T is the sum of all weights, we know that $W_T \geq$ *weights*[best expert], and so by plugging into above:

$$\left(\frac{1}{2}\right)^M \leq \left(\frac{3}{4}\right)^m \cdot n \quad \Rightarrow \quad \left(\frac{4}{3}\right)^m \leq 2^M \cdot n \quad \Rightarrow \quad m \leq \frac{1}{\log_2(4/3)}(M + \log_2 n)$$

invert and rearrange

apply $\log_2$ *on both sides*

$$\frac{1}{\log_2(4/3)} \approx 2.41$$

How good is this solution?

Rate of mistakes. Ratio $\dfrac{M}{T}$ when T goes to infinity → rate at which we make a mistake

Suppose the best expert makes a mistake $10\,\%$ of the time, so the rate of mistakes is $10\,\%$

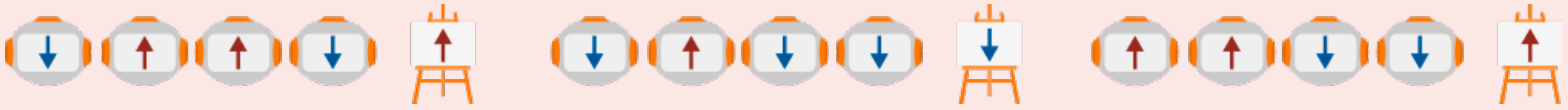
Since $\log_2 n$ is small even for large n, this solution has a rate of mistakes of $24\,\%$

Remark. The best possible bound on number of mistakes is $2M$

idea of bad instance: $n = 2$ experts, one always says 1 and the other 0

Multiplicative Weights: quiz 2

What is the state of the weights array after the following observations?



A. {1, 1/2, 1/2, 1/4}

B. {1/2, 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 as an algorithmic framework

Historical context. The multiplicative weights algorithm has been rediscovered in multiple fields of computer science, as the solution to many seemingly very different problems.

first known version of the algorithm was proposed in the 50s as an algorithm to solve zero-sum games

The experts problem can be used to model many problems:

- Machine Learning: boosting algorithms [more on this soon]
- Optimization: solving linear and semi-definite programs [experts are constraints]
- Maximum flow: efficient algorithms [experts are graph paths]
- Game theory: solving zero-sum games [experts are pure strategies]
- Computational geometry
- Gradient descent: analyzing convergence

you are not supposed to know what any of these are, but you'll probably hear about some of these soon, as you learn more advanced computer science topics

Experts problem for K-ary predictions

Problem. Suppose we want to solve the experts problem with a perfect expert, but the predictions are K-ary.

K-ary prediction. A prediction over a universe of K objects (e.g. K=4 will it rain, snow, be foggy be sunny?)

K–ary elimination algorithm

On each day:

- **take the most popular prediction over the experts predictions**
- **after observing the actual outcome: remove/ignore all experts that predicted the wrong outcome**

Multiplicative Weights: quiz 3

What is the best upper bound on the number of mistakes of the K-ary elimination algorithm?

A. $K \log_2 n$

B. $K \lfloor \log_2 n \rfloor$

C. $\lfloor \log_K n \rfloor$

D. $\lfloor \log_2 n \rfloor$

E. $K + \log_2 n$

Multiplicative weights method - using randomization

We can use randomization to improve our solution!

Randomized Multiplicative Weights Method

Initialize a double[n] array called weights and set all values to 1

On each day:

- **let zeroWeight be the sum of weights of experts predicting 0**
- **let oneWeight be the sum of weights of experts predicting 1**
- **predict 0 with probability zeroWeight / (zeroWeight + oneWeight)**
- **after observing the actual outcome: halve the weight of all the experts that predicted incorrectly**

Intuition. Make decisions according to how confident we are on them.

Proposition. The randomized multiplicative weights method makes at most $1.5M + 2\log_2 n$ mistakes in expectation.

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*

(Binary) Classification problem

Input. A *training data set* of elements / items, and a (binary) label (0 or 1) per point.

e.g. medical images / emails

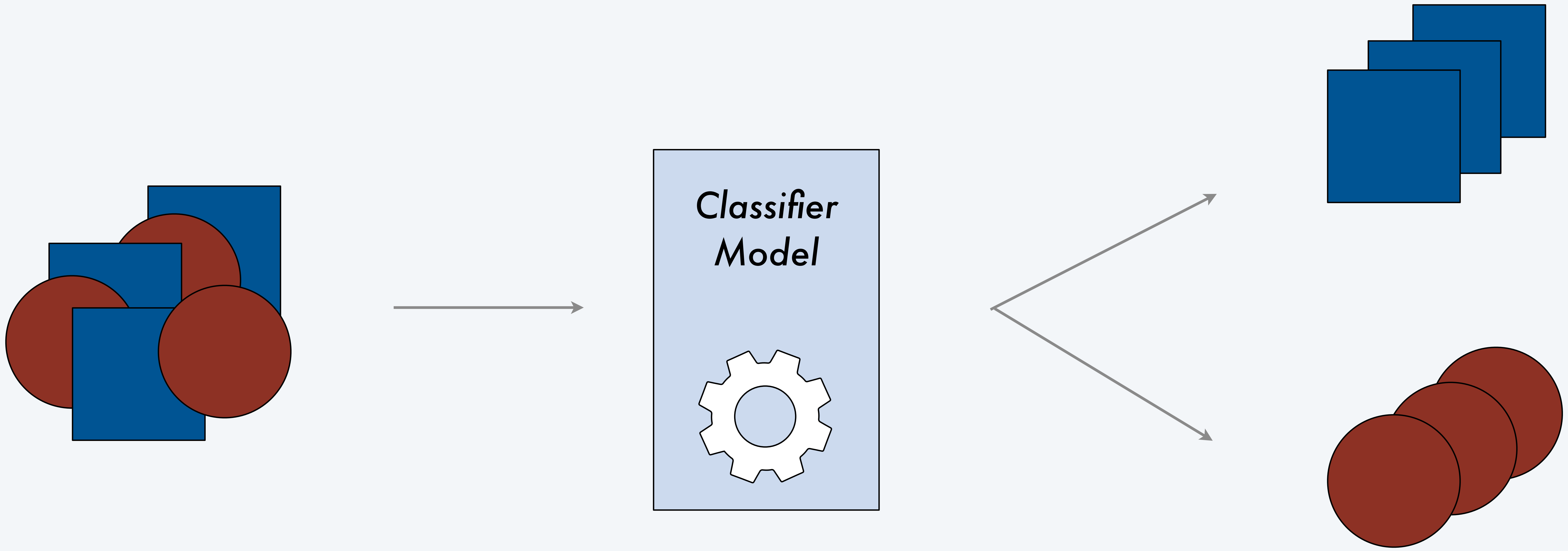
e.g. has disease or doesn't? / spam or not?

Goal. Predict the label of new data

an algorithm that makes predictions

- Phase 1: Train some model based on the training data.
- Phase 2: Apply the model to the new data to predict the label.

Def. The *accuracy* of a model is the the fraction of correct predictions on a certain data set.

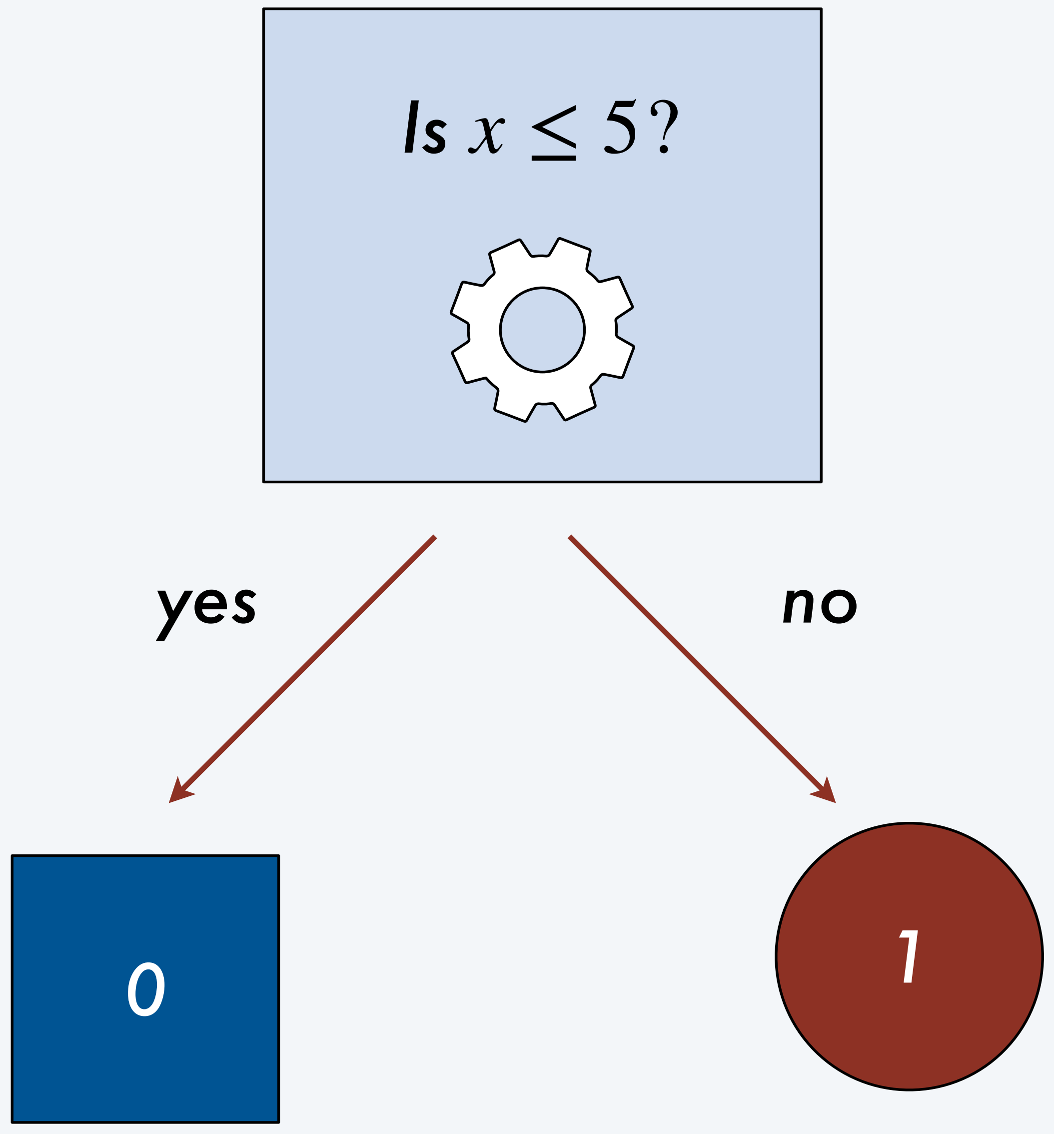


Assumption. For simplicity assume that the elements / items are points in D dimensions

array of D doubles

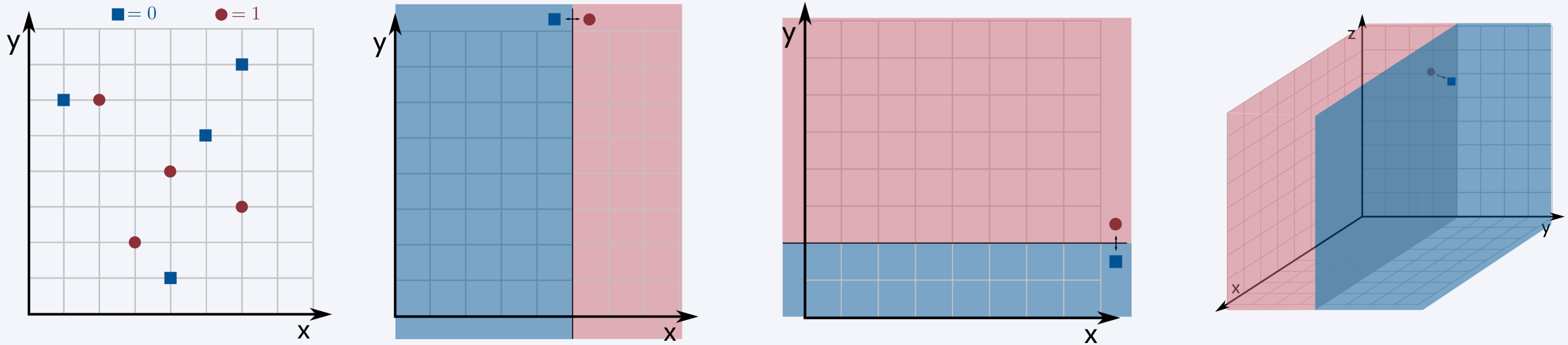
Weak Learner / Decision Stump

Def. A *decision stump* is a simple classifier model that makes a prediction based on a single dimension.



This is equivalent to picking some (hyper)plane / line and predicting 0 for all elements on one side and 0 otherwise.

Training goal. Find the decision stump that maximized accuracy on training data.



Remark. A decision is an example of a *weak learner*, a model performs marginally better than random.

Boosting methods

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

a meta-algorithm / family of algorithms

Analogous to amplification of randomness!

a "weak" algorithm

Recall from randomness:

Amplification. If $\mathbb{P}[A \text{ is correct}] = 1\,\%$, repeat 500 times.

Then, $\mathbb{P}[A_1, A_2, \ldots, A_{500} \text{ are all incorrect}] \leq \left(\frac{99}{100}\right)^{500} < 1\,\%$

Multiplicative weights and boosting: AdaBoost

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

Yoav Freund

Robert Schapire

former instructor of 226!

Here is a simplified version of the algorithm:

Simplified AdaBoost algorithm

Initialize a double[n] array called weights and set all values to 1

Repeat T times:

some parameter we get to pick

- **train a decision stump with the input weighted according to weights**
- **double weight of points incorrectly labelled by the decision stump**
- **normalize weights (divide each entry by the sum of weights)**

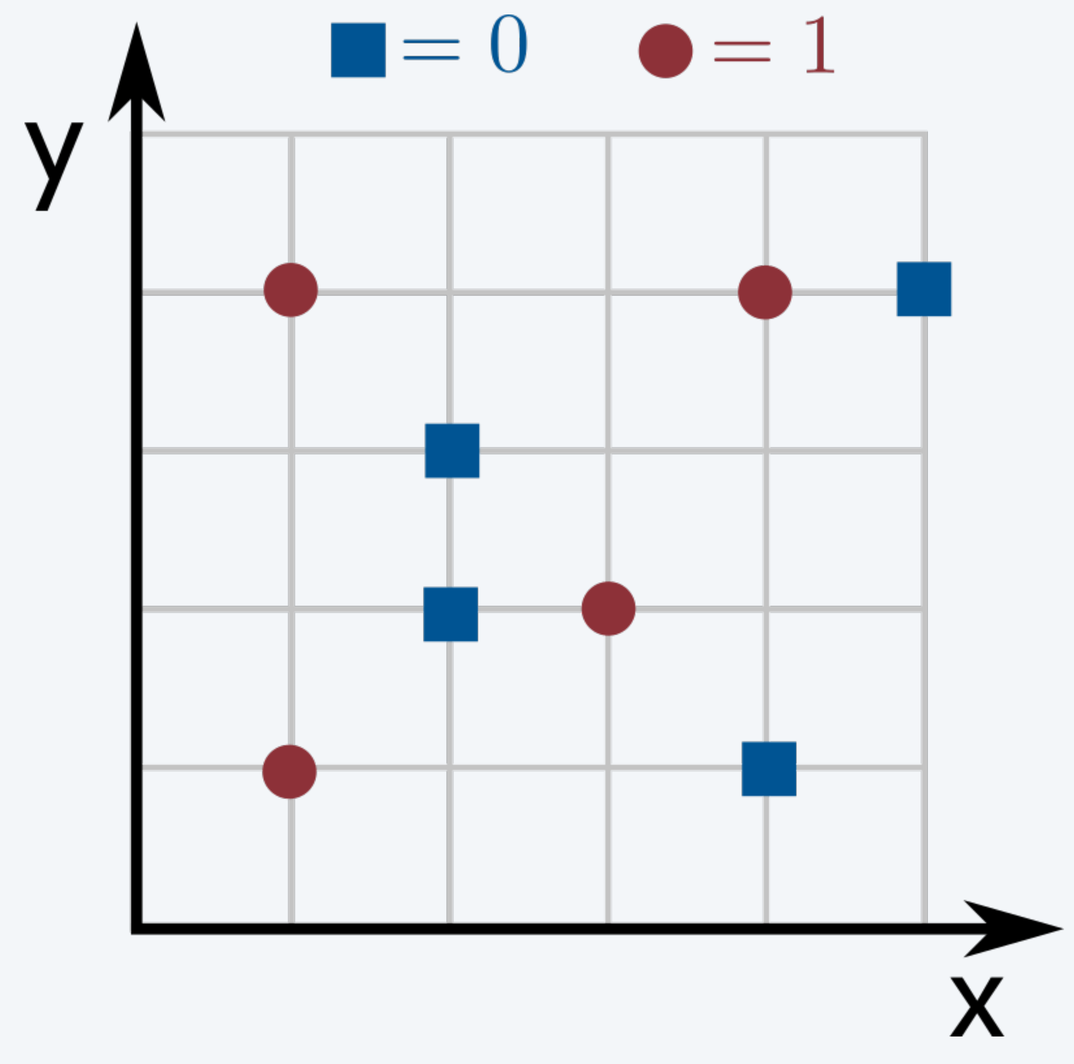
force the algorithm to do a better job on misclassified points

To predict on new data use all decision stumps and take majority

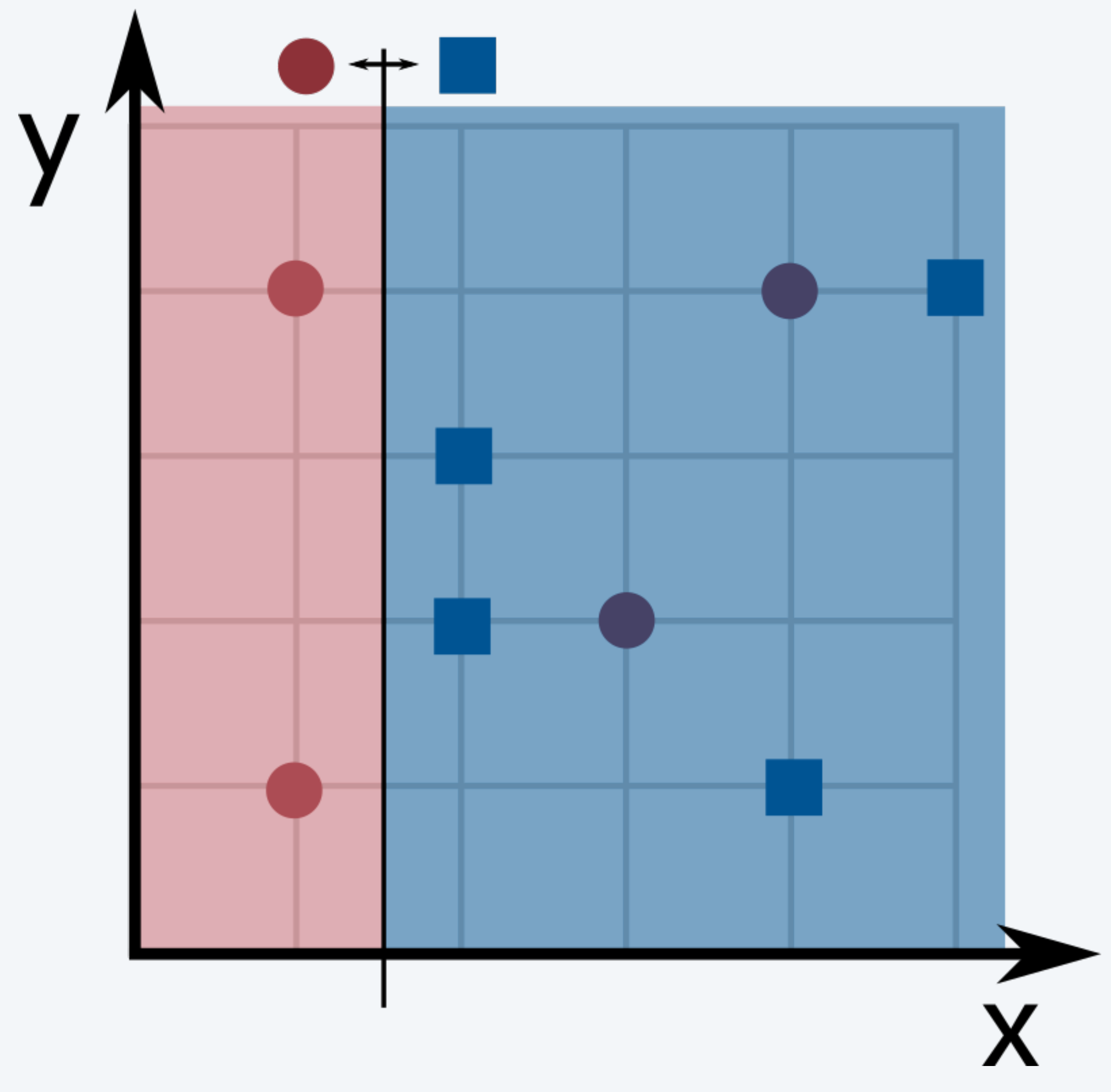
Remark. The real AdaBoost doesn't double weights, instead it multiplies by a factor that depends on the error of the decision stump. It also doesn't take a normal majority, it gives preference to better decision stumps

Simplified AdaBoost demo

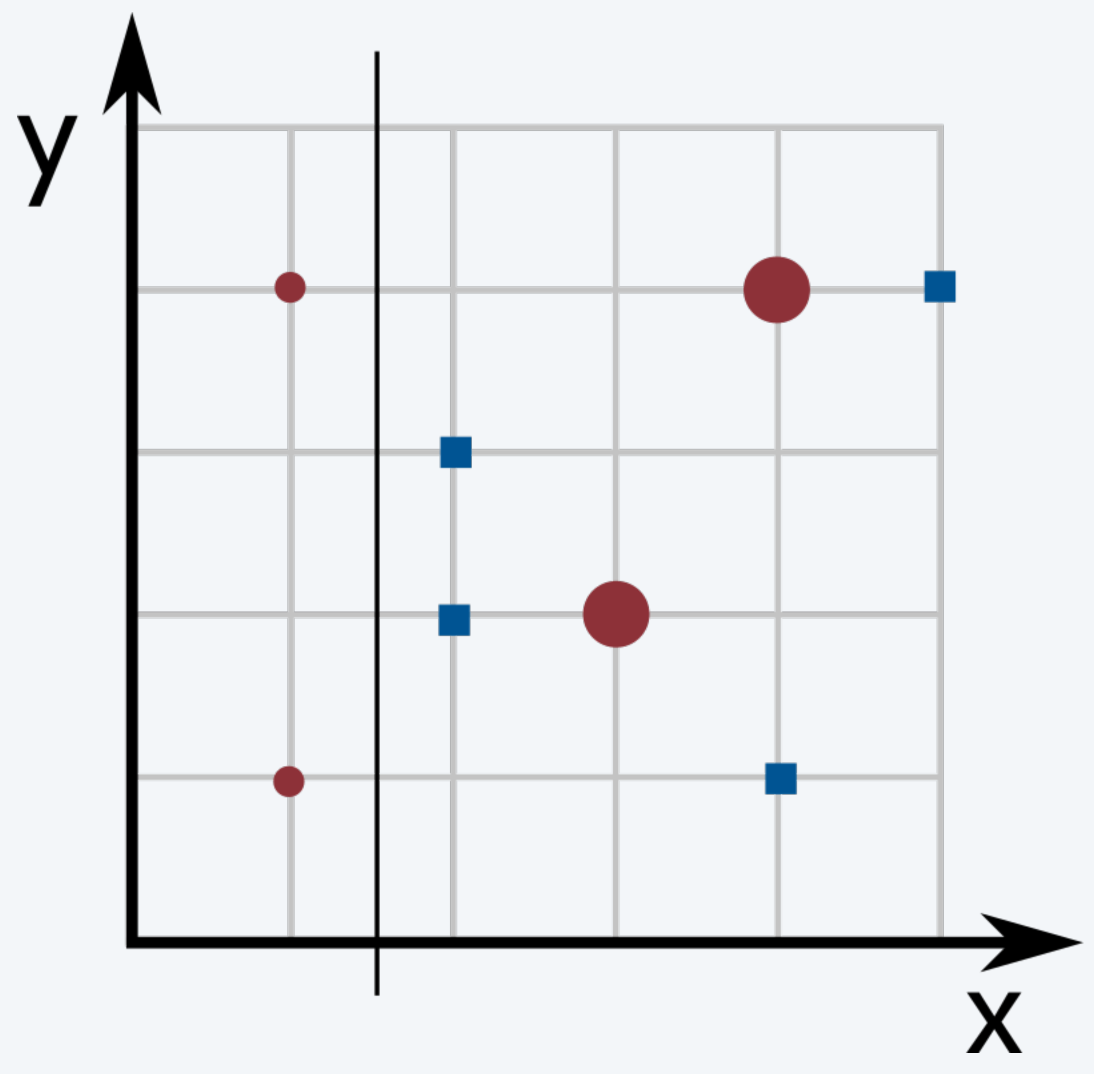
weights = {1,1,1,1,1,1,1,1}



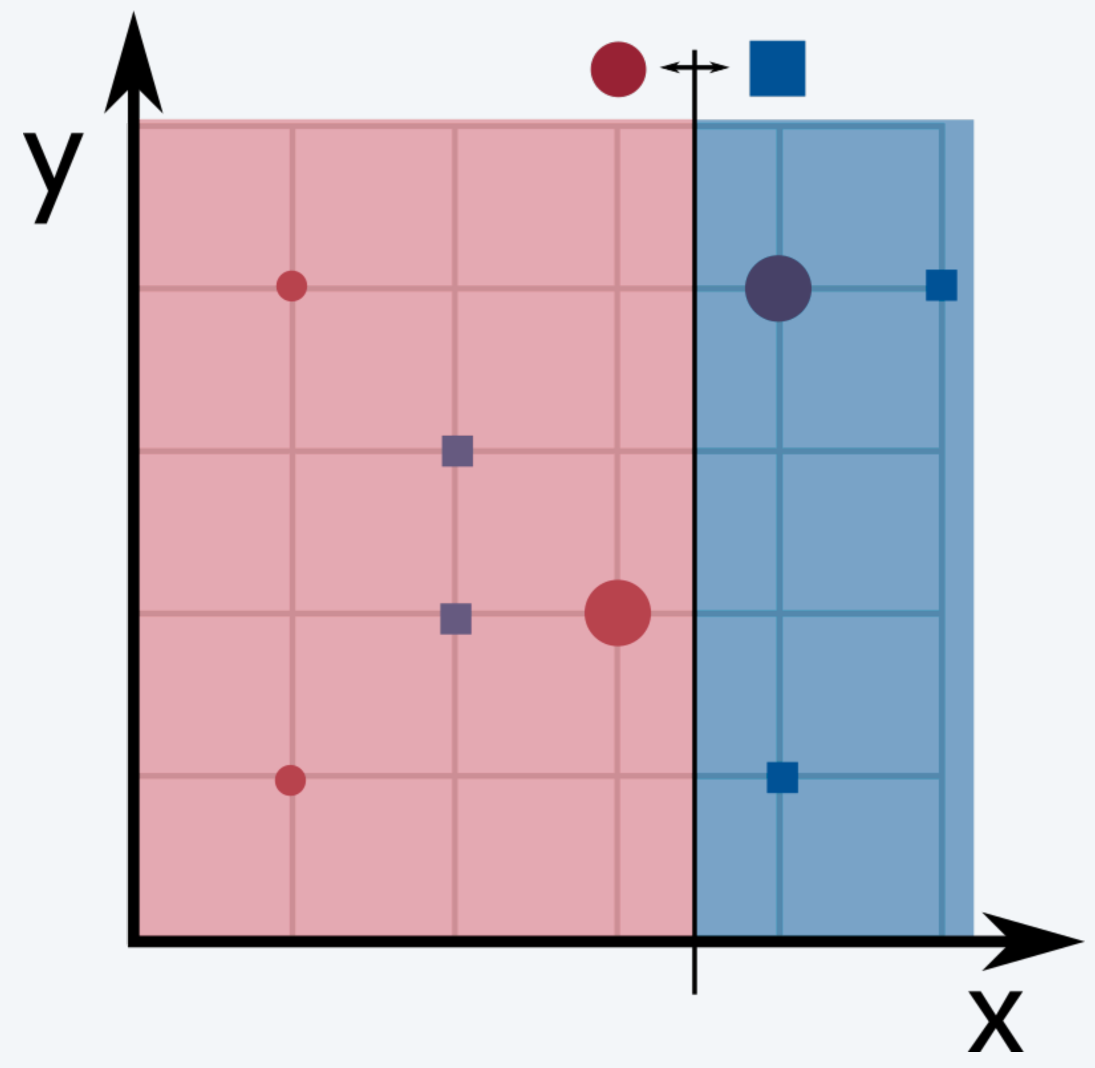
$x \geq 1.5$?



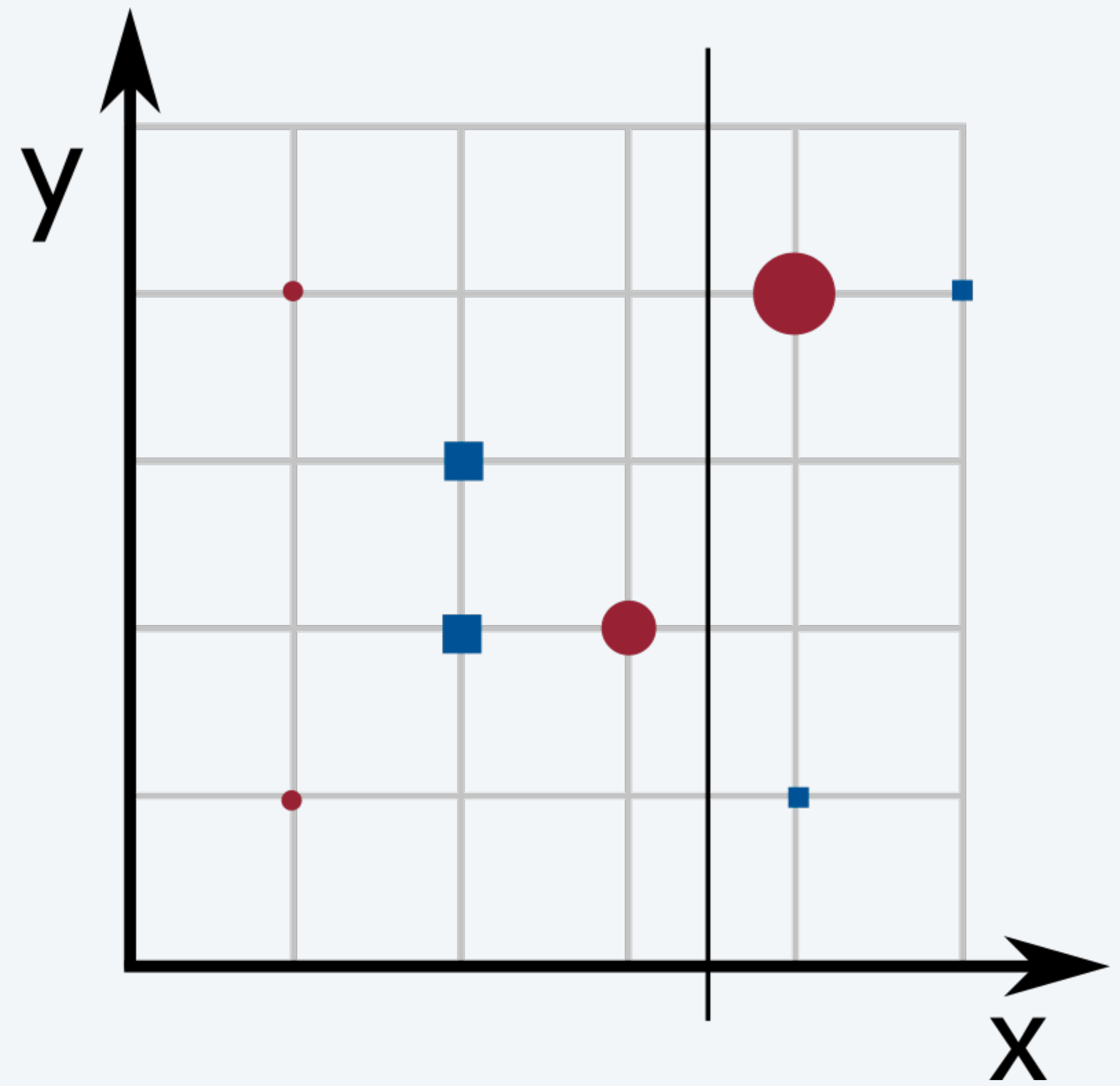
weights = {1,1,1,1,1,1,2,2}



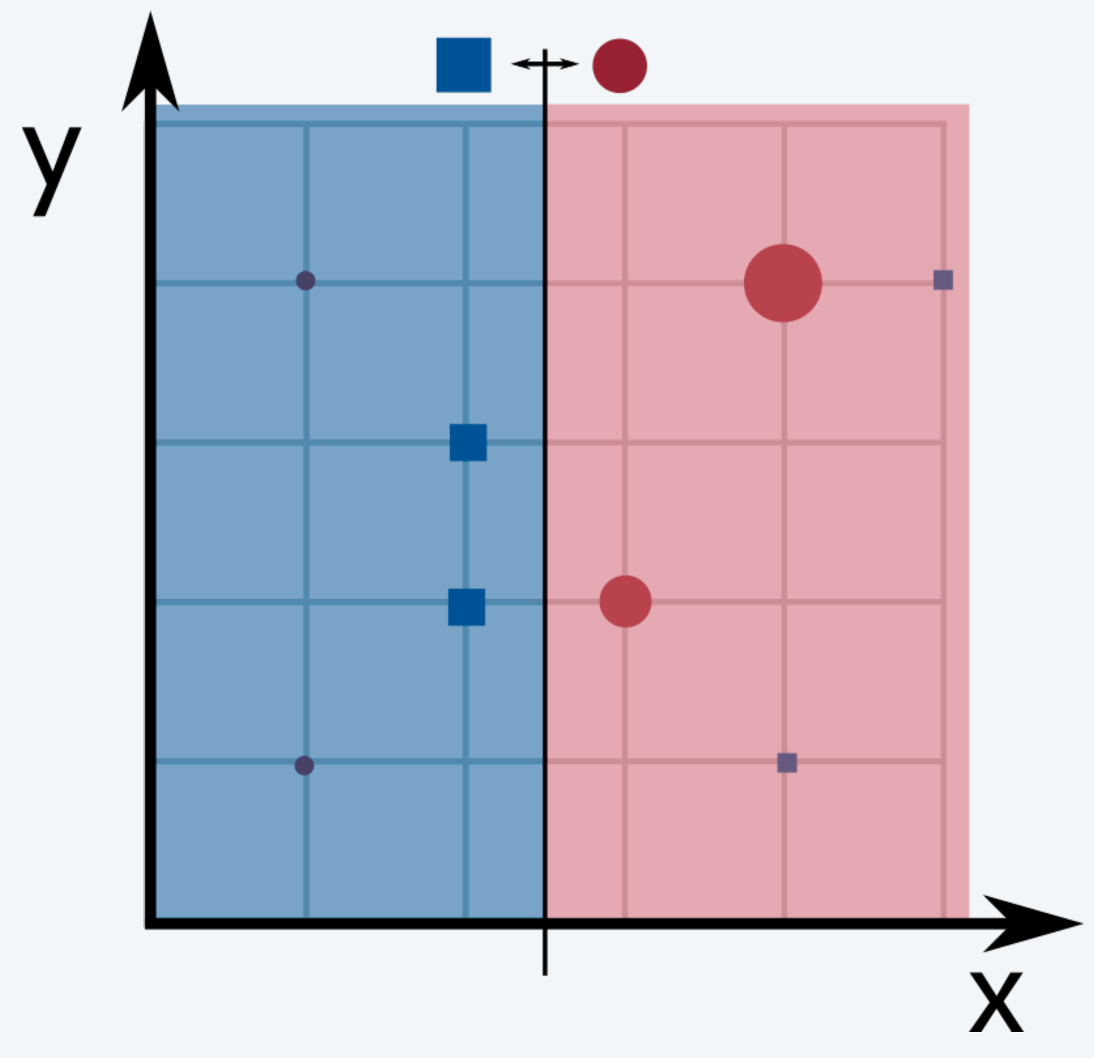
$x \geq 3.5$?



weights = {2,2,1,1,1,1,2,4}



$x \leq 2.5$?



Algorithms

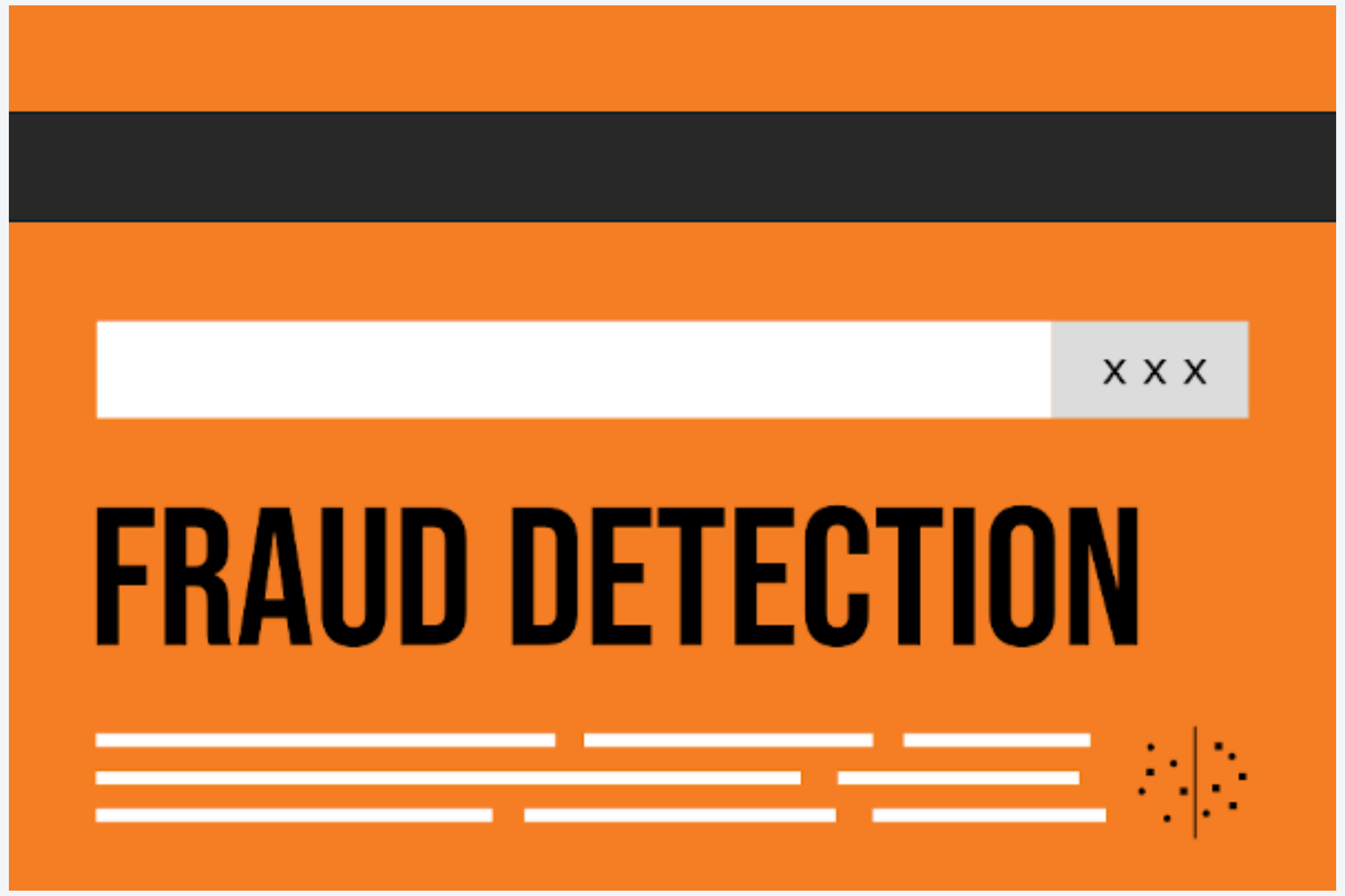
Robert Sedgewick | Kevin Wayne

https://algs4.cs.princeton.edu

Multiplicative Weights

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

Assignment 7: Fraud detection



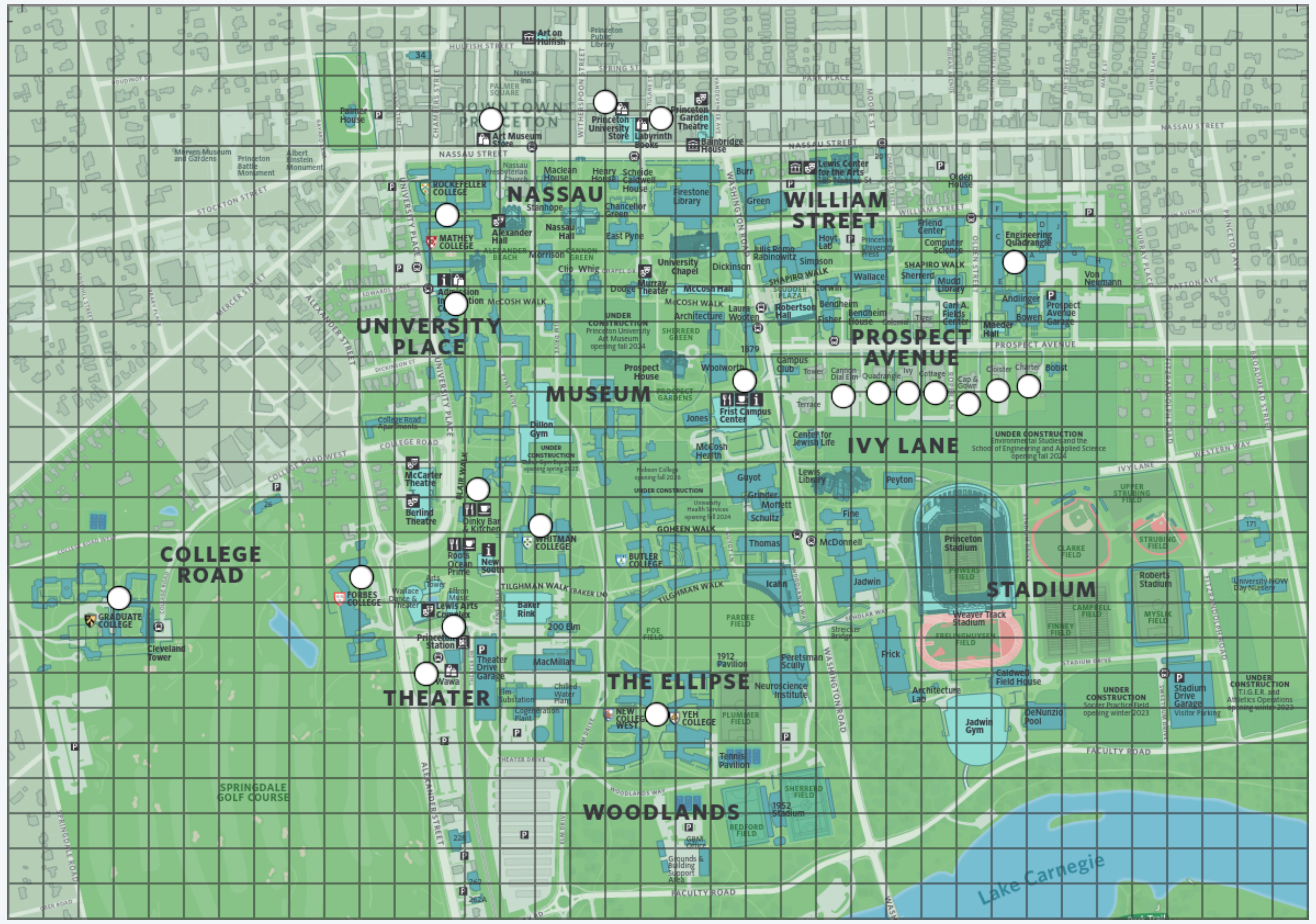
Motivation. Given a transaction summary of credit card uses in a set of locations, predict whether there was a fraudulent one.

Input. A set of M locations on a map, and a collection of labelled (with *clean*/0 or *fraud*/1) transaction summaries.

Goal. Train a boosting classifier to classify new transaction summaries.

a transaction summary

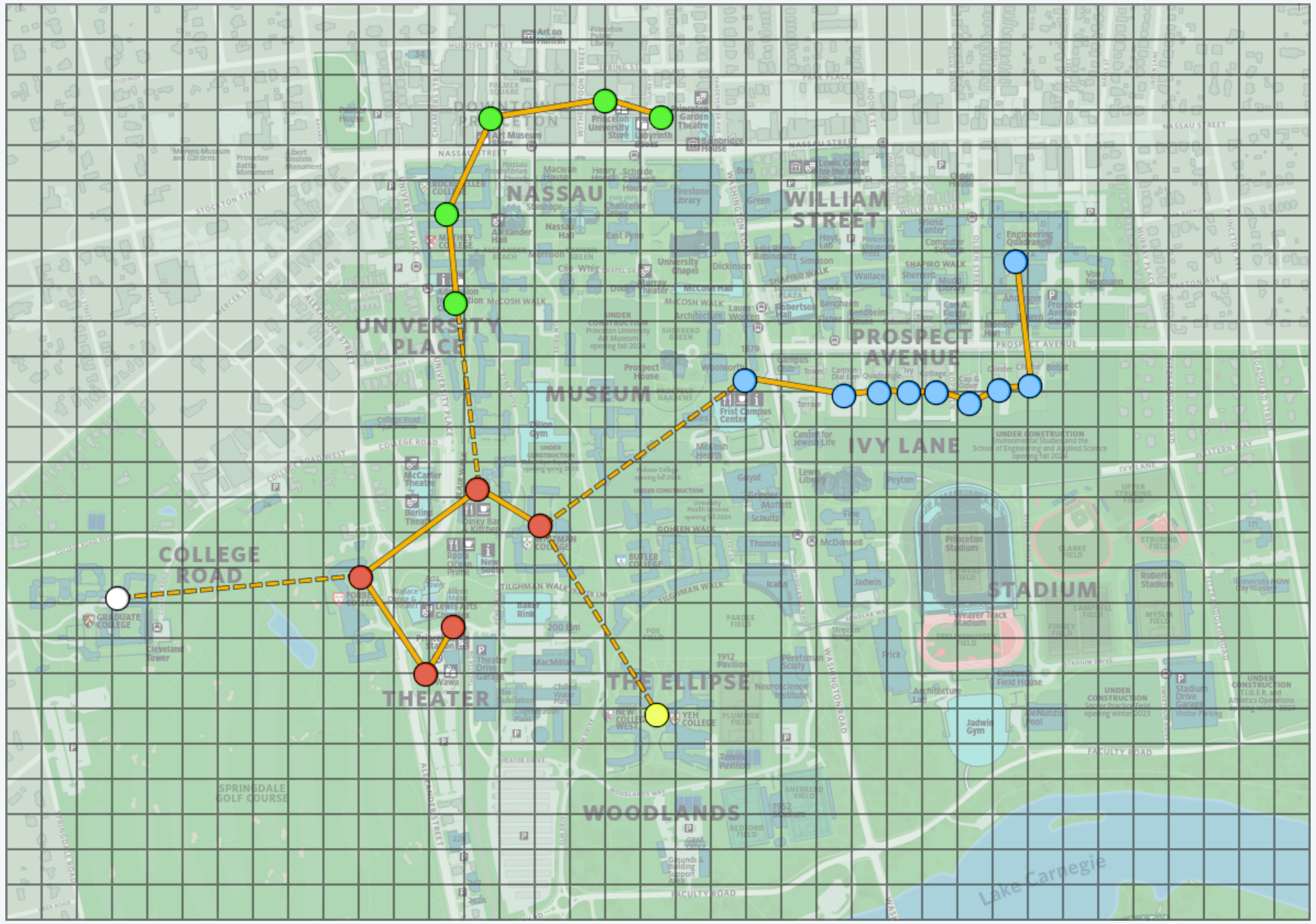
5	6	7	0	6	7	5	6	7	0	6	7	0	6	7	0	6	7	0	6	7	clean



Assignment summary

Part 1. Clustering / Dimensionality Reduction

- Compute "clusters" to group locations.
- Find a Minimum Spanning Tree of the distance graph.
- Consider connected components of "cluster graph", a graph with only some of the lowest weight edges.
- Collect transactions into new clusters.

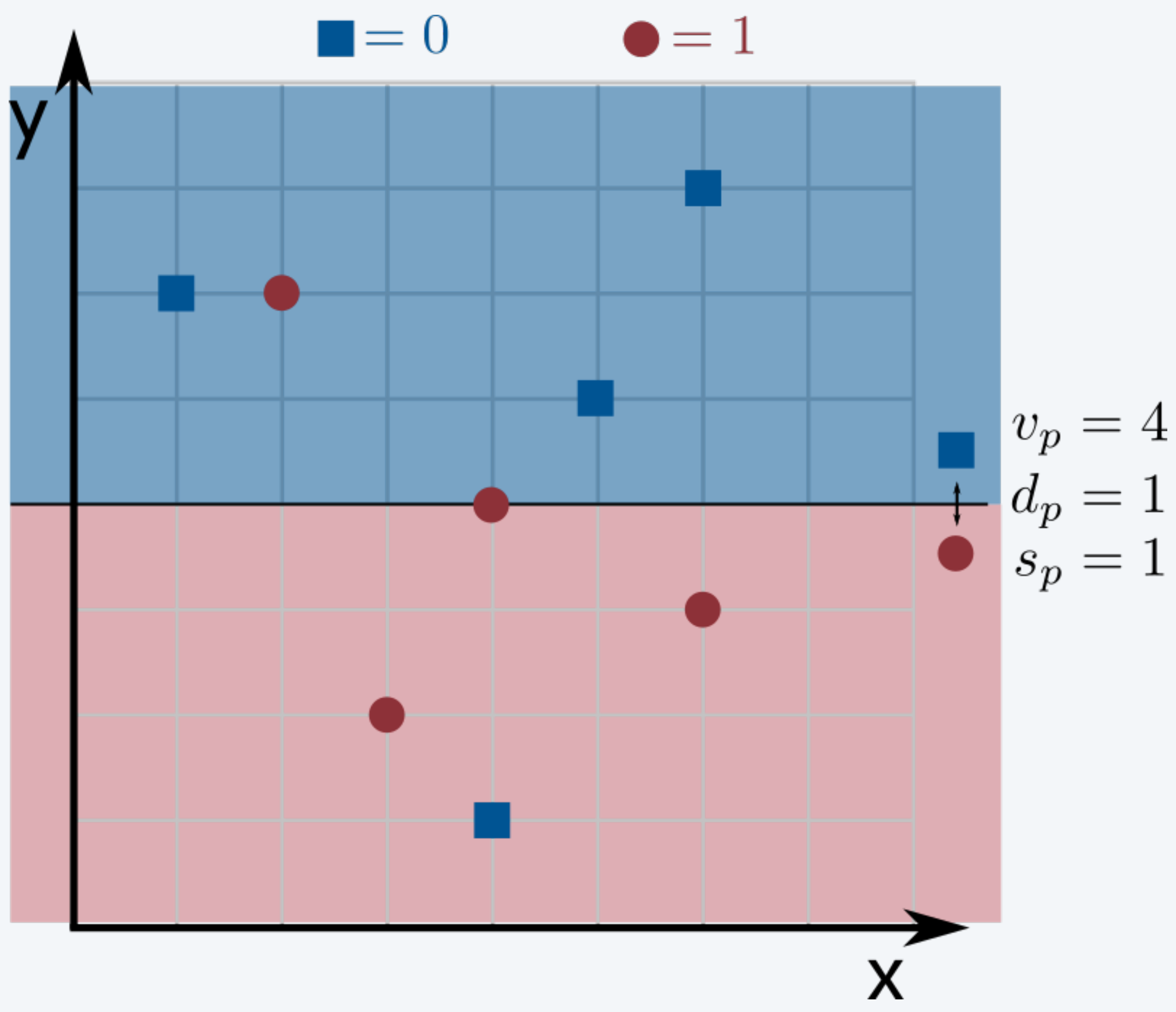


5	6	7	0	6	7	5	6	7	0	6	7	0	6	7	0	6	7	0	6	7	Original
5	32	45	12	7	Reduced																

Assignment summary

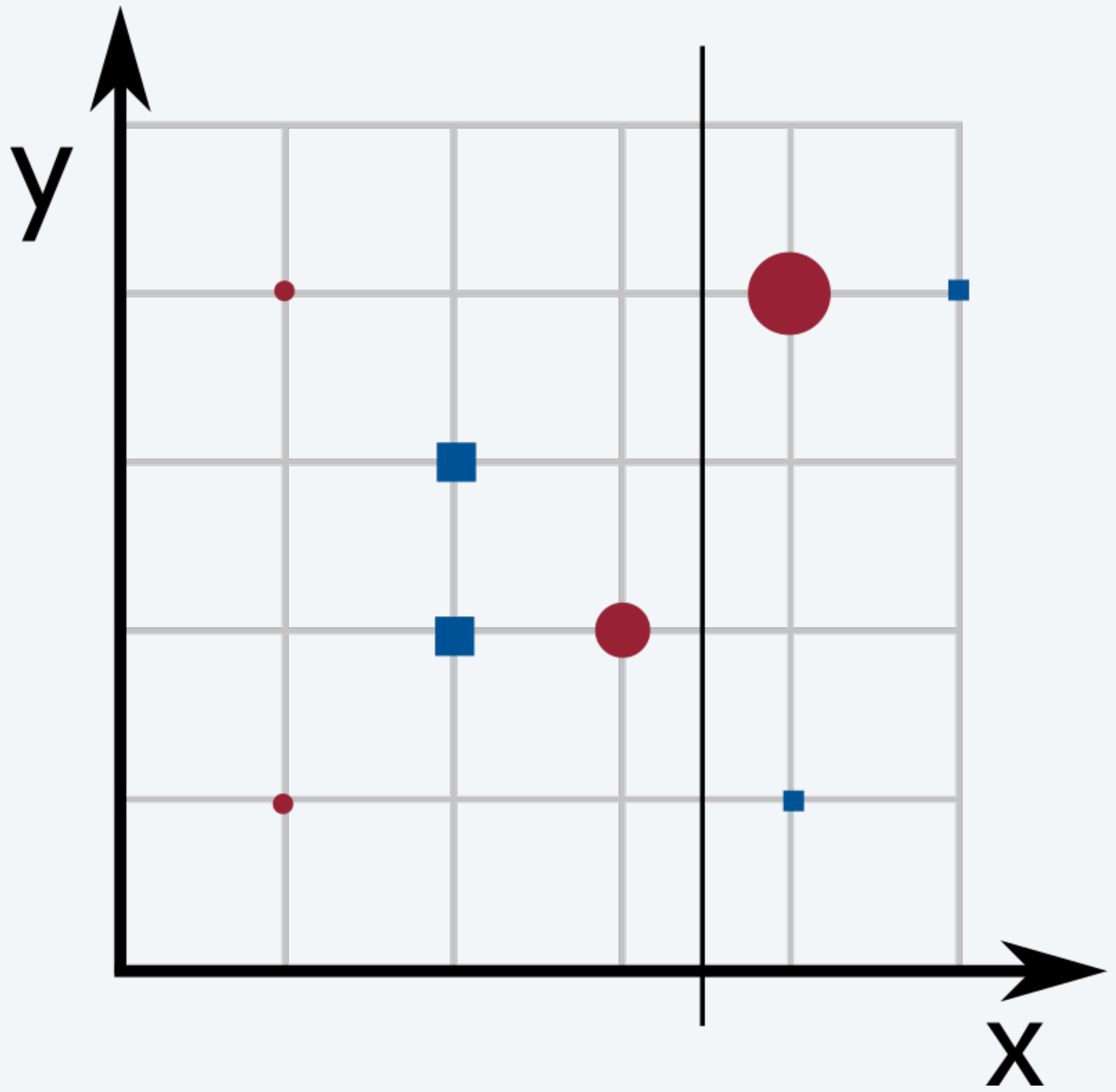
Part 2. Weak Learner / Decision Stump

- Create a class to train a decision stump
- Should handle weighted points (to use in boosting later)



Part 3. Boosting Algorithm

- Create a class to train a simplified AdaBoost



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	

