

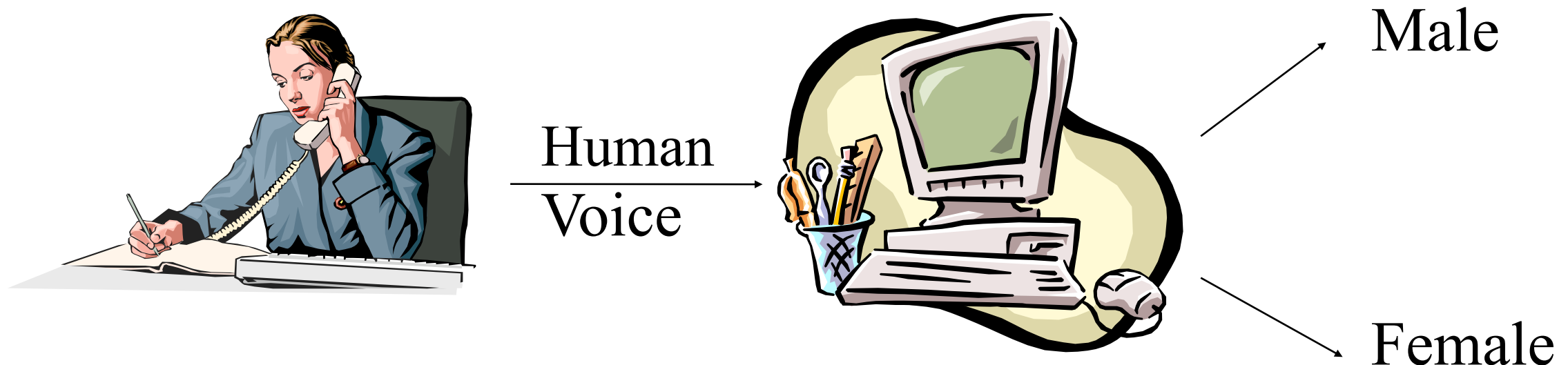
Drifting games, Boosting & Online Learning

Yoav Freund
UCSD

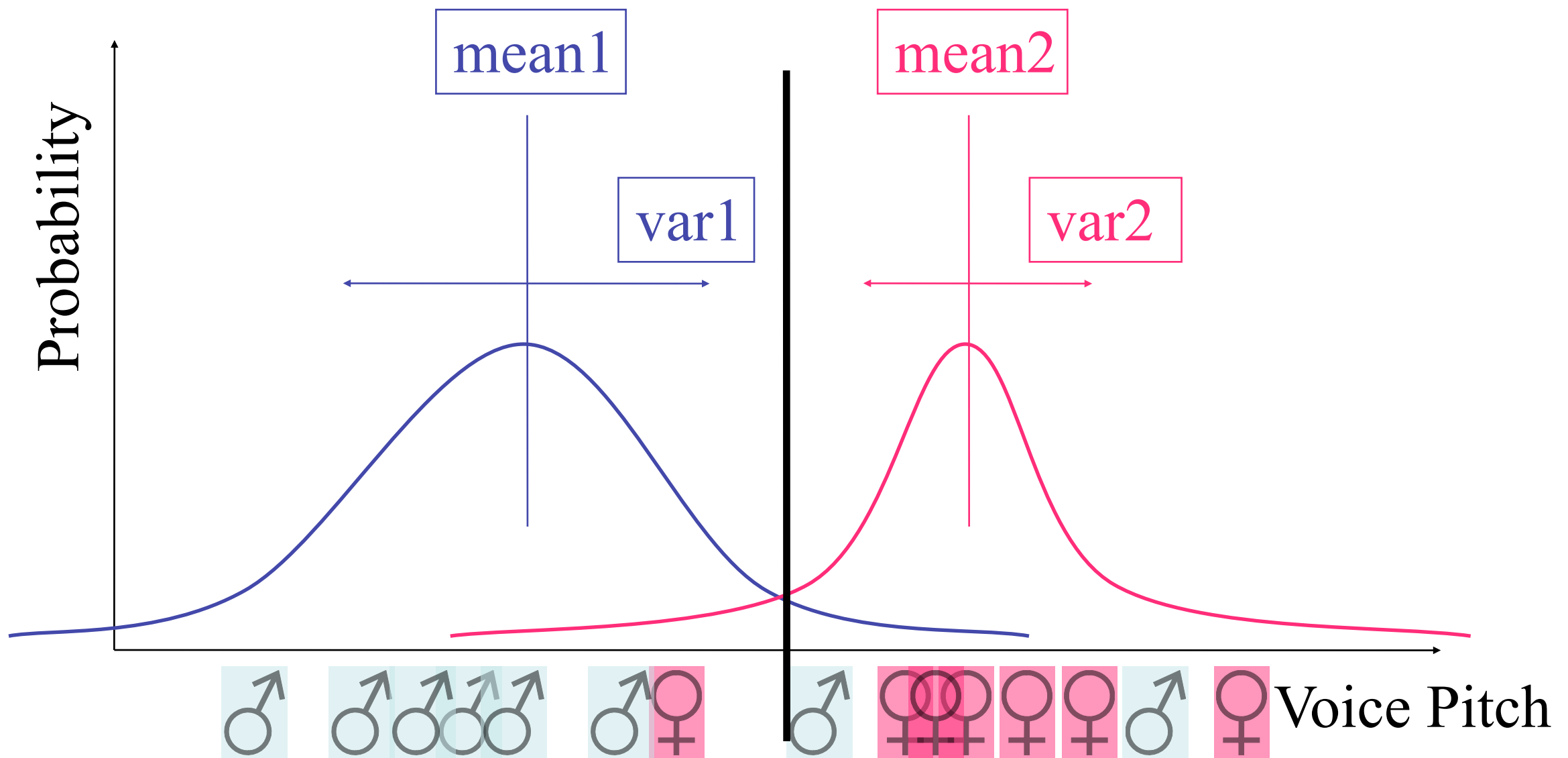
Generative, Discriminative and online modeling

Toy Example

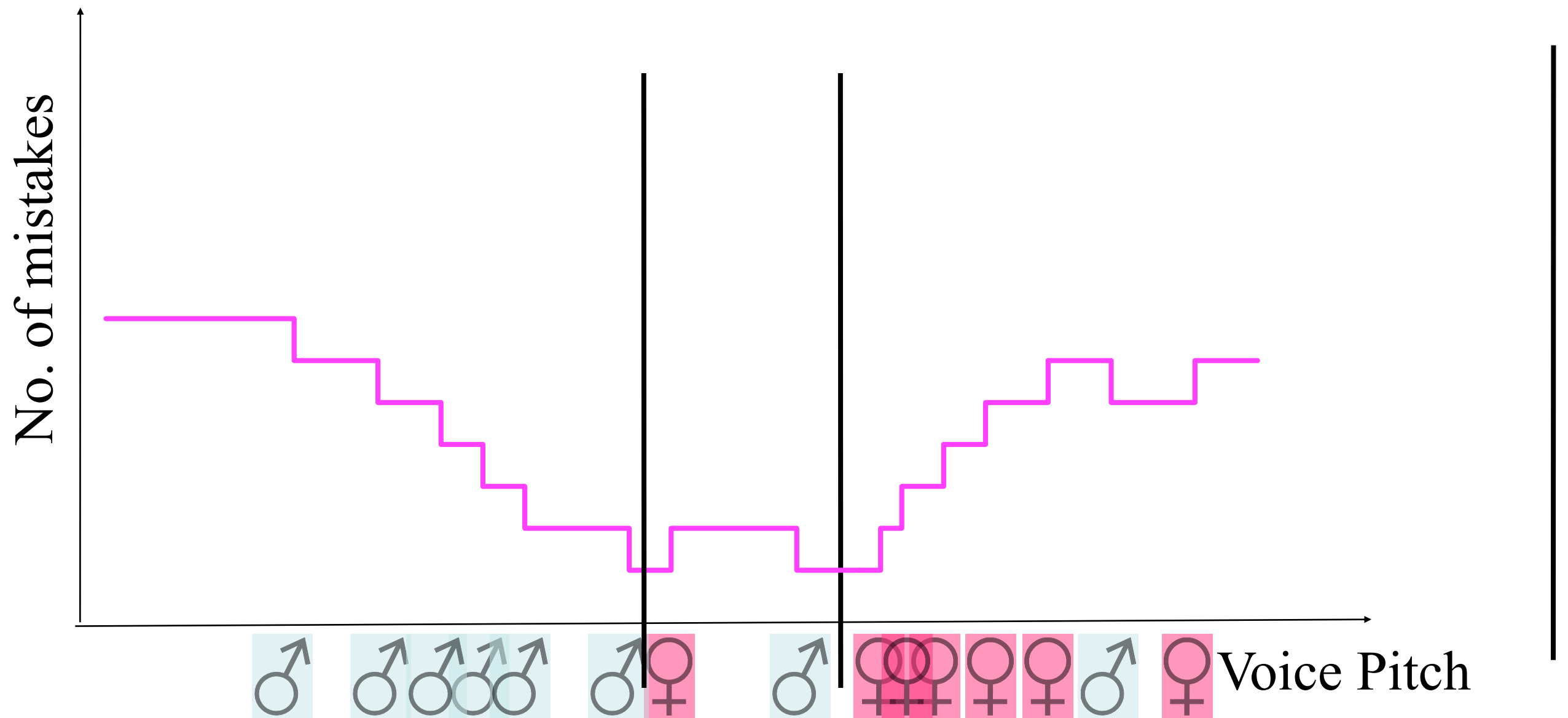
- Computer receives telephone call
- Measures Pitch of voice
- Decides gender of caller



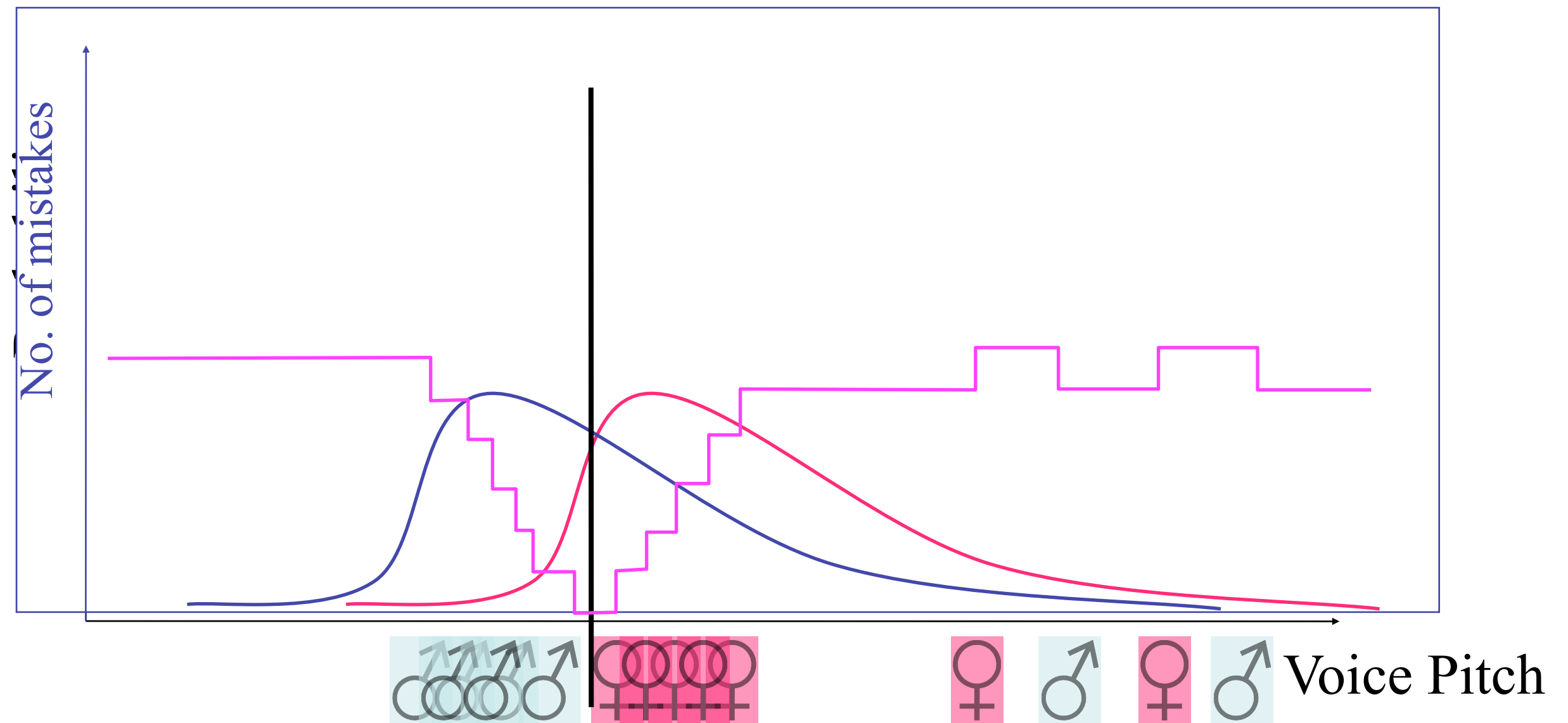
Generative modeling



Discriminative approach



ILL-BEHAVED DATA



Generative model



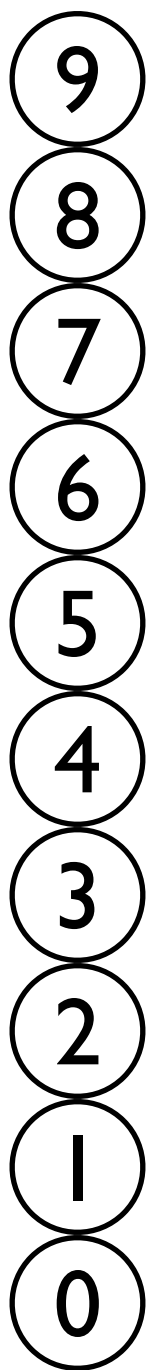
20 questions

called “Ulam’s game” by mathematicians

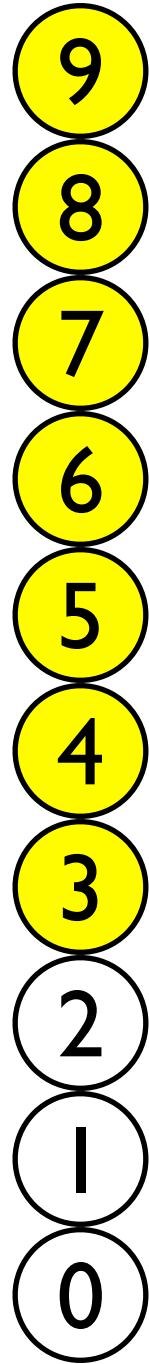
- Alice’s goal: find Bob’s secret
(a number in $1..n$)
- In each iteration Alice asks:
is the number in the set S ?
- Alice wants to minimize number of
iterations, Bob wants to maximize it.

Strategies for Bob

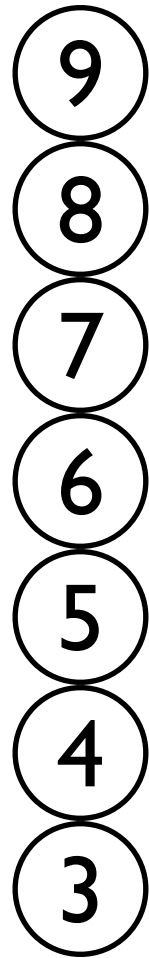
- **Naive Bob:** Choose a number, hope for the best.
- **Sneaky Bob:** Keep the set of consistent secrets as large as possible.



Alice: asking about a
large set or a small set
is a bad idea

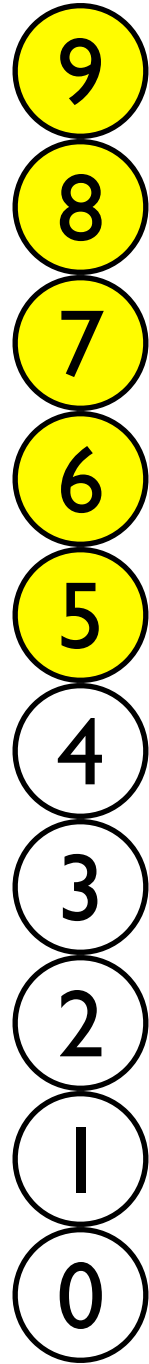


Alice: asking about a large set or a small set is a bad idea



Bob: answer to keep larger set consistent.

Alice: best to ask about
half of the currently
consistent answers



Bob: no advantage to
answering **yes** vs. **no**

Secret revealed after

$$\lceil \log_2 10 \rceil = 4$$

questions

Alice: If the size of the set is odd, it cannot be split it into two equal parts.



Assume **n** is a power of 2

secret identified after
 $\log_2 n$ questions

Strategies for Alice

- **Deterministic:** split set into two equal parts.
- **Stochastic:** choose each element to be in or out of the set **independently at random with equal probabilities ($1/2, 1/2$)**.

Min-Max strategies

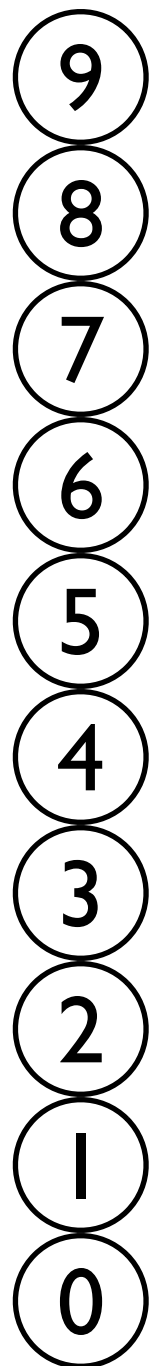
- Alice: each element is in S independently at random with probabilities $1/2, 1/2$
- Bob: Answer “Yes” or “No” independently at random with probabilities $1/2, 1/2$
- Guarantees that the expected number of steps from $|S|=n$ to $|S| \leq 1$ is $\text{ceil}(\log_2(n))$
- If one side is known to deviate, the other side would deviate too.

one step look-ahead

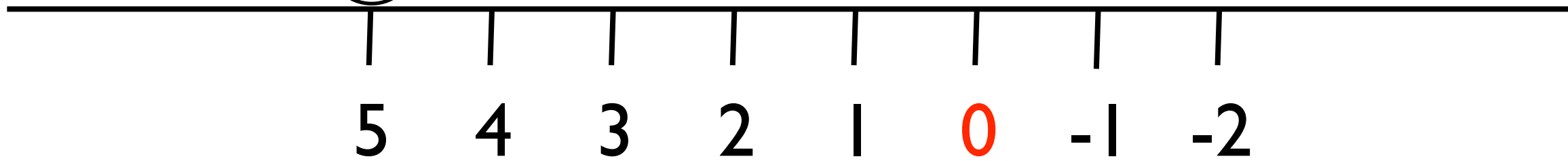
- One can derive the optimal response to a known strategy for the current step by assuming that all of the later steps will be done using min/max optimal strategies.

Ulam's game with k lies

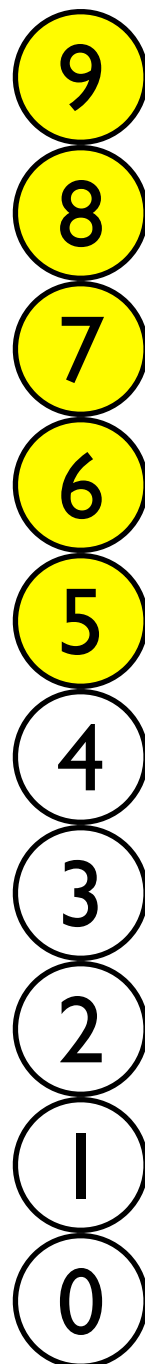
- **Bob** can give an incorrect answer $\leq k$ times.
- **Sneaky Bob**: keep as many answers as possible consistent with $\leq k$ lies



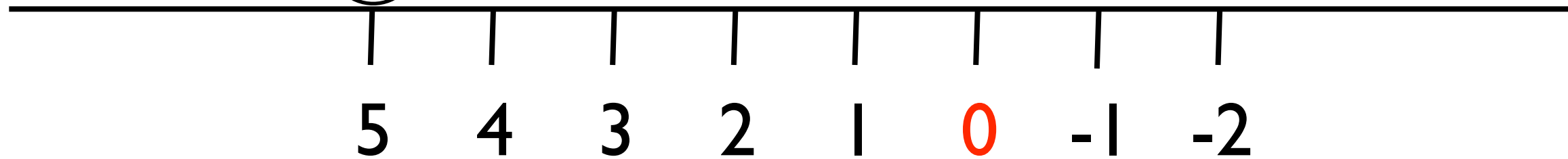
Configuration 0



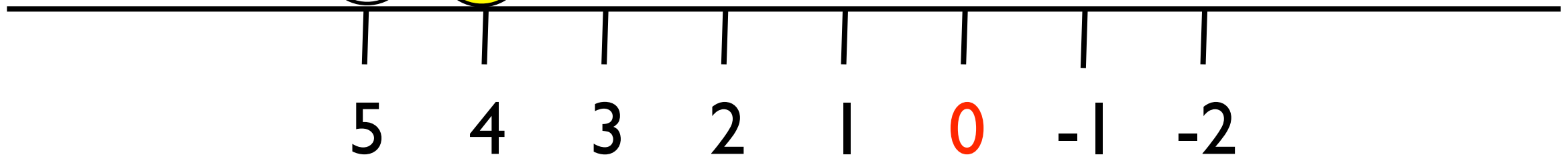
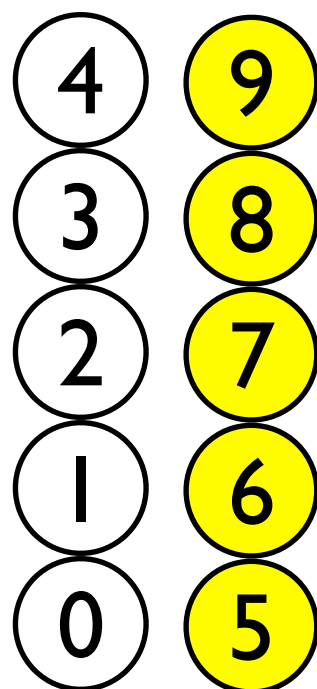
$k=5$



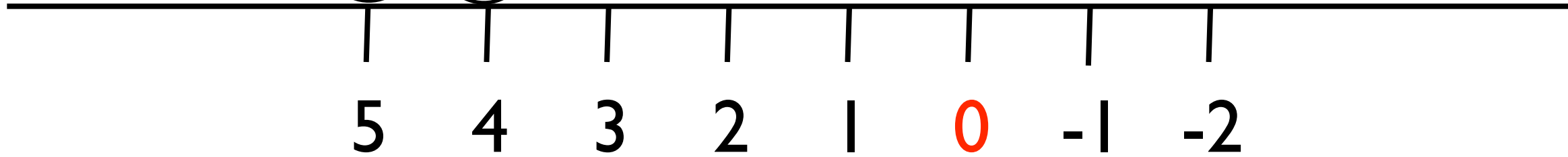
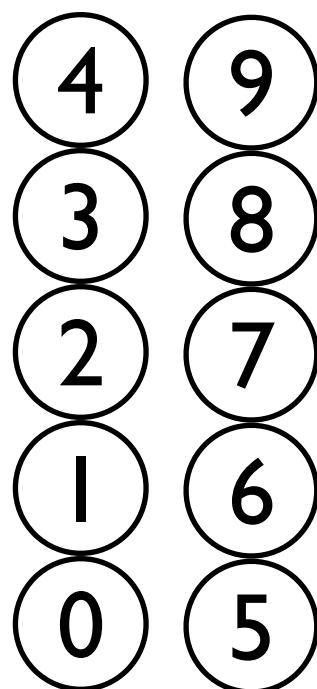
Alice's move



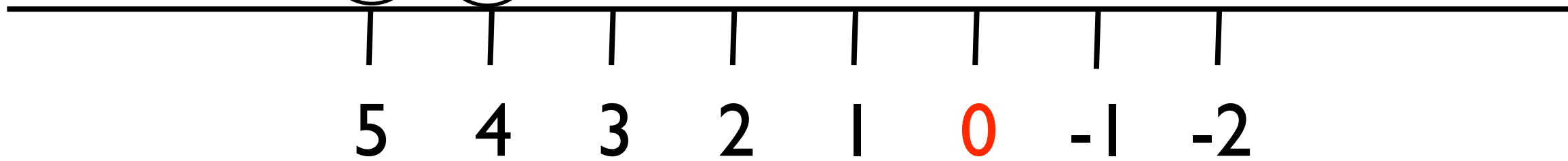
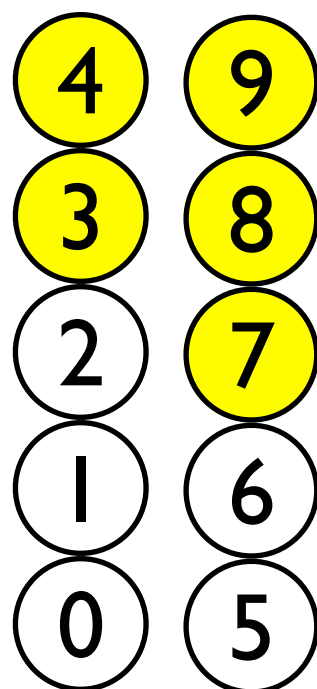
Bob's move



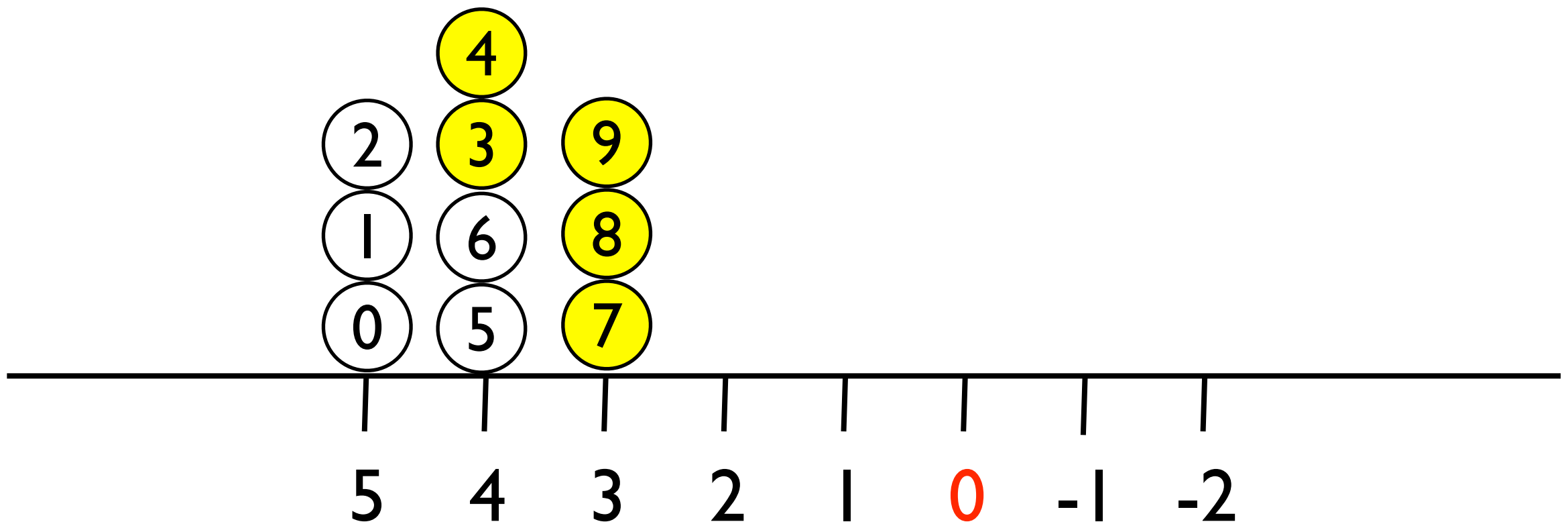
Configuration I



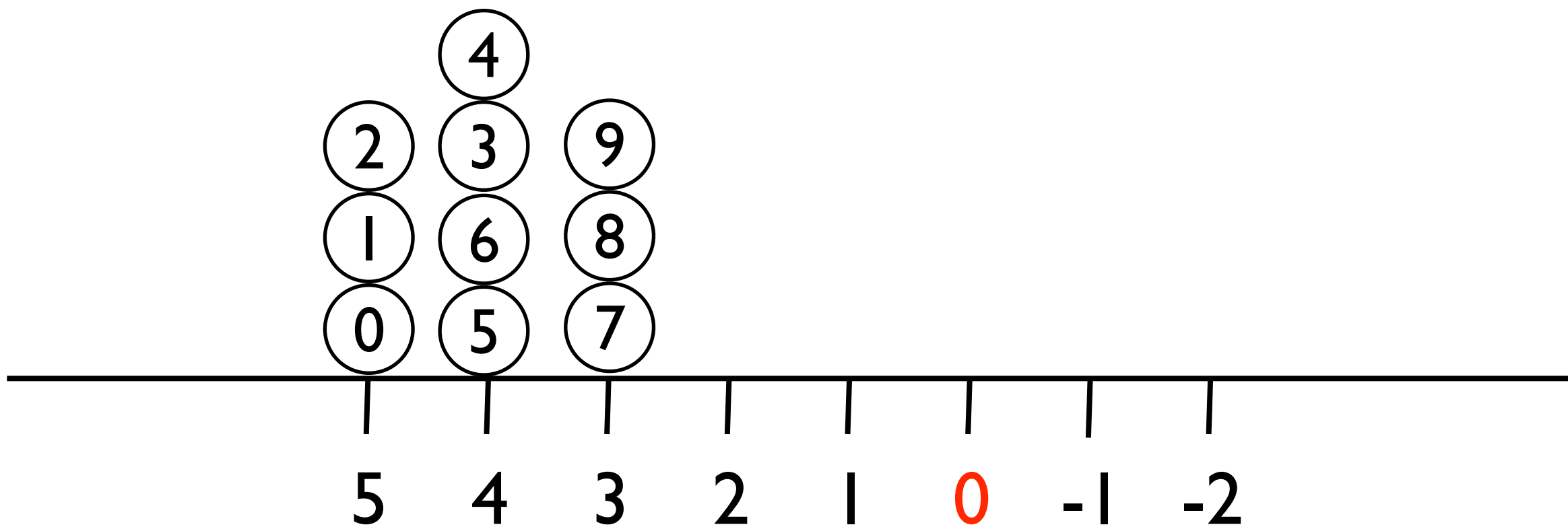
Alice's move



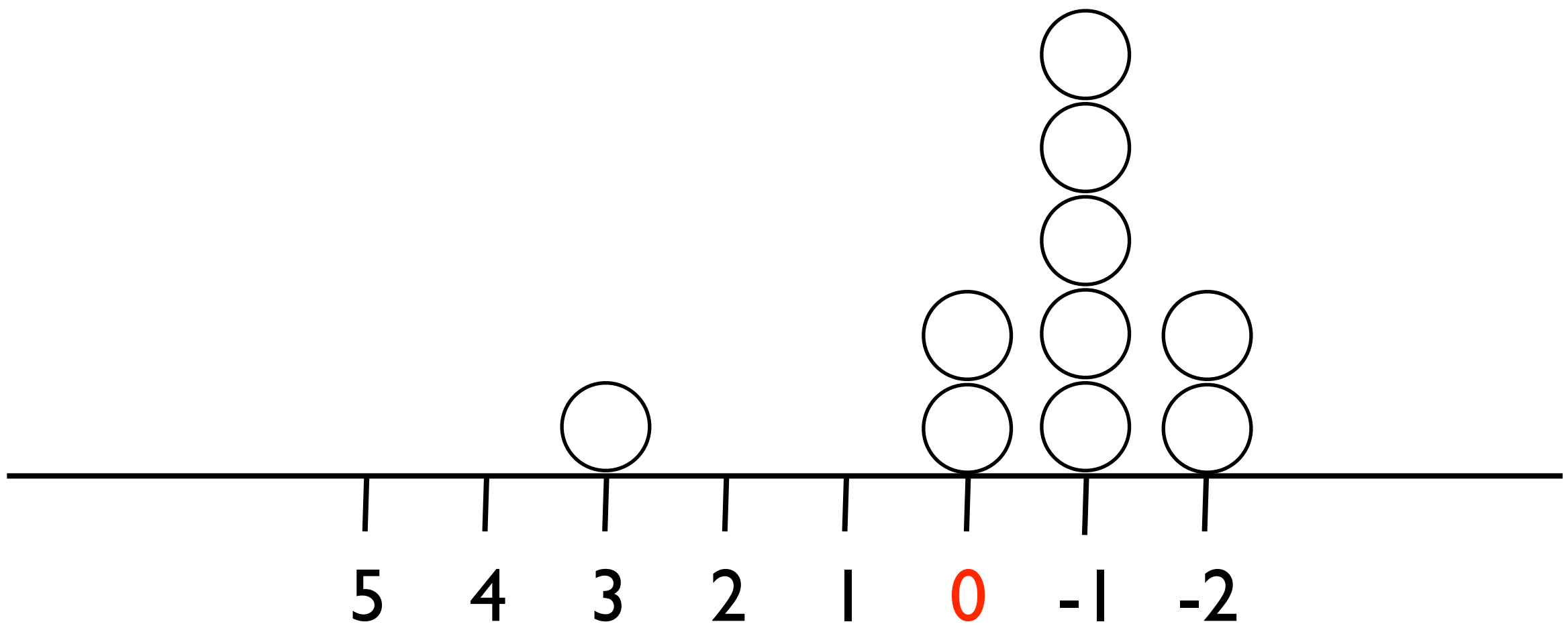
Bob's move



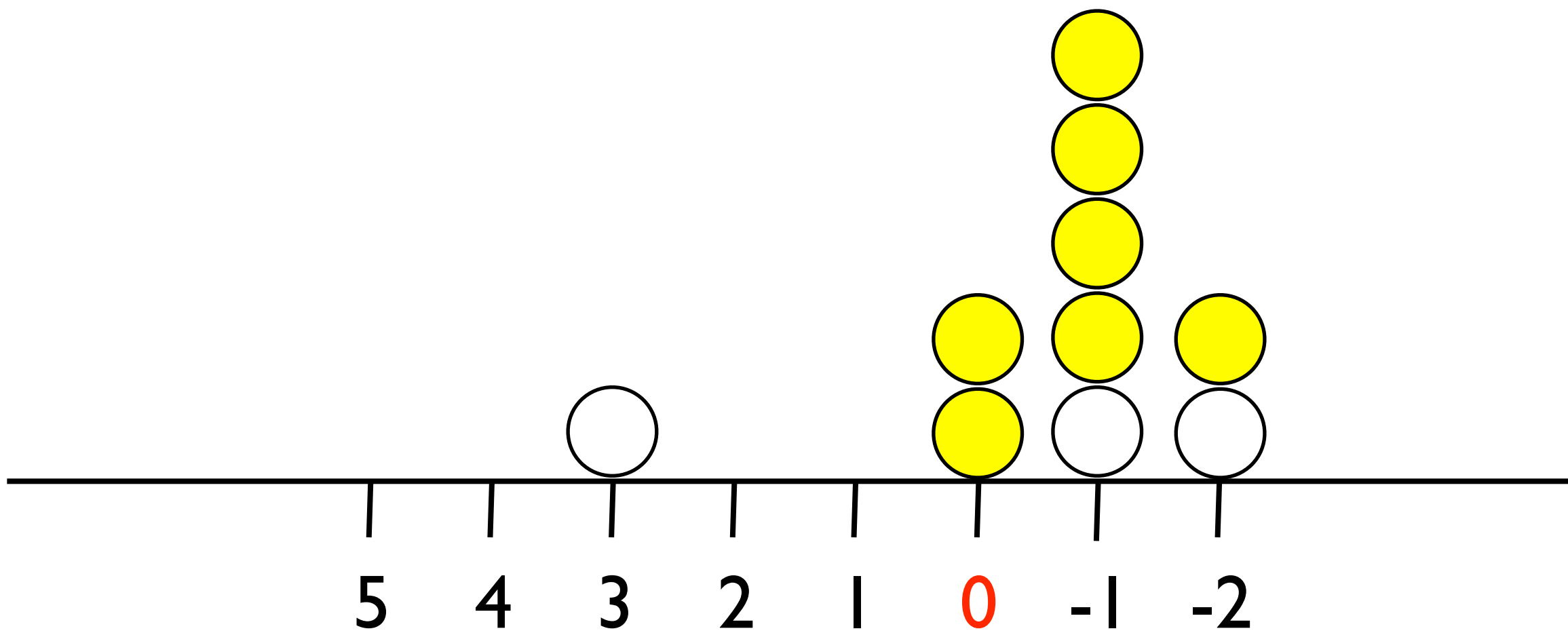
Configuration 2



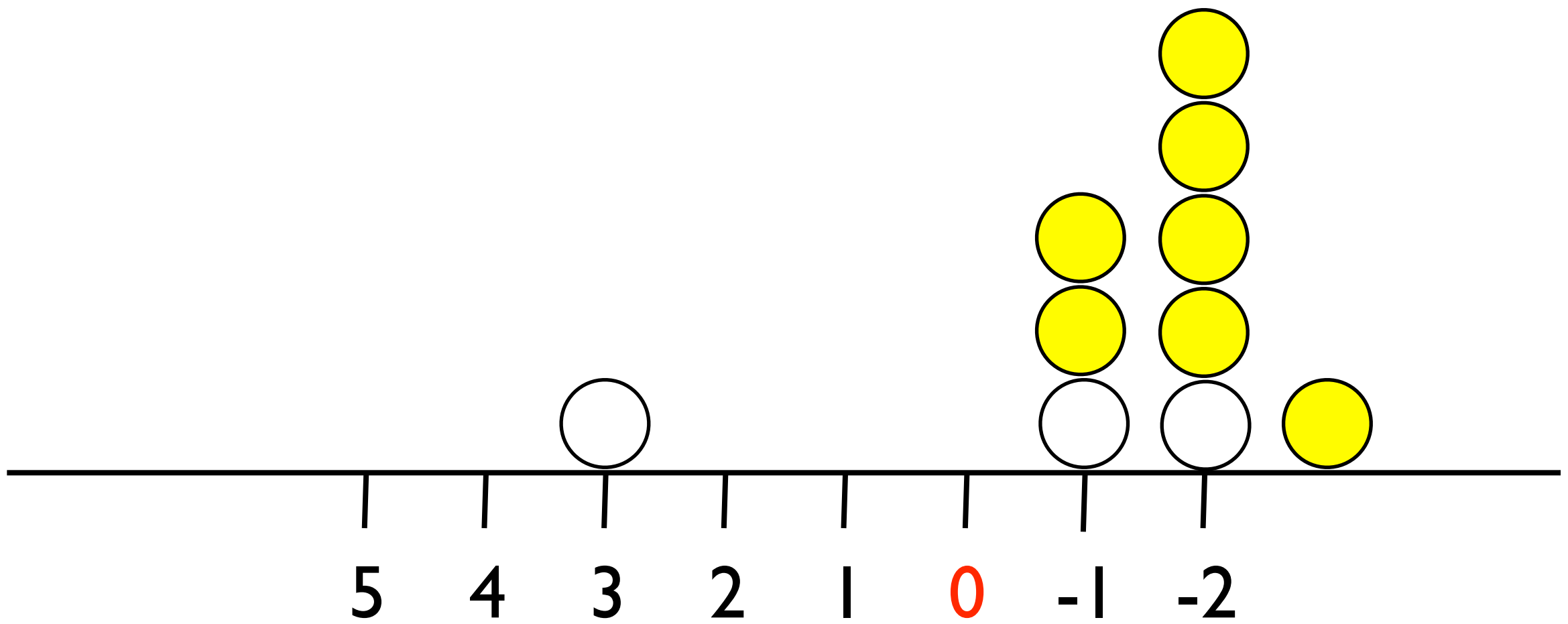
Configuration j



Alice's move



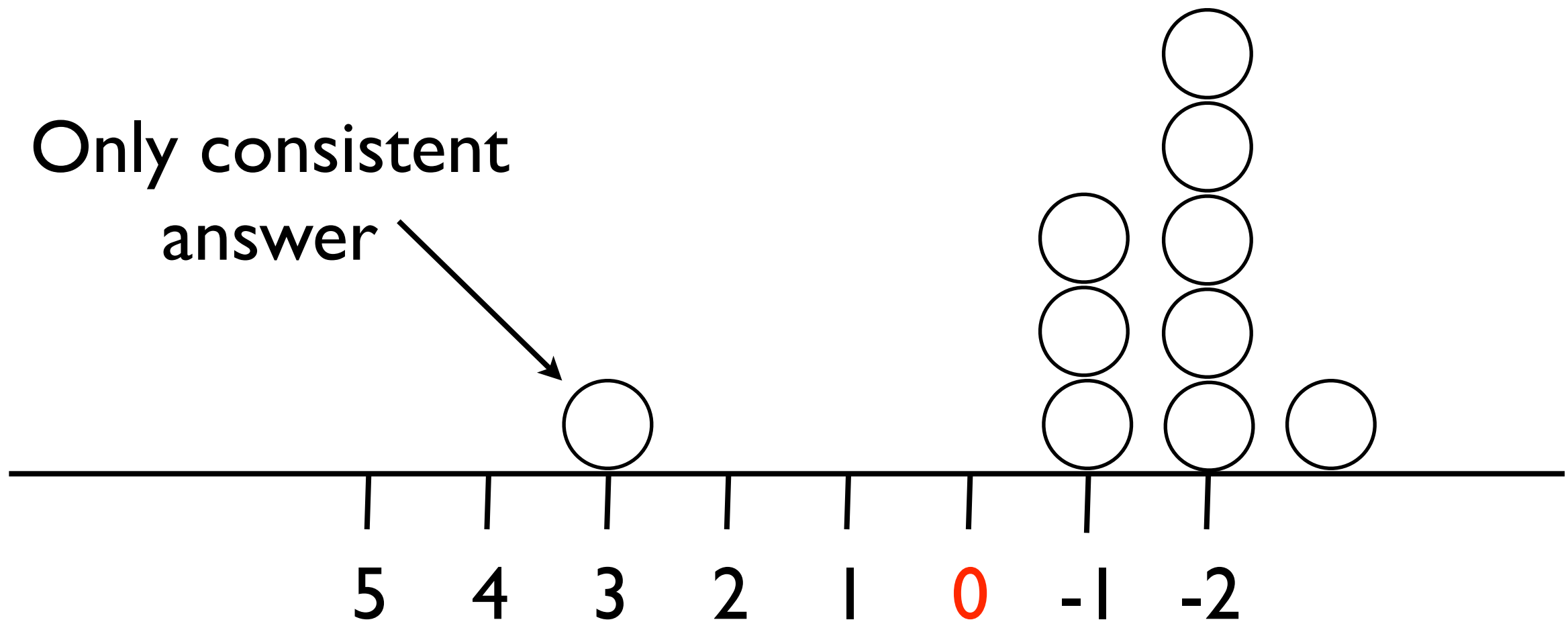
Bob's move



Final Configuration

GAME ENDS

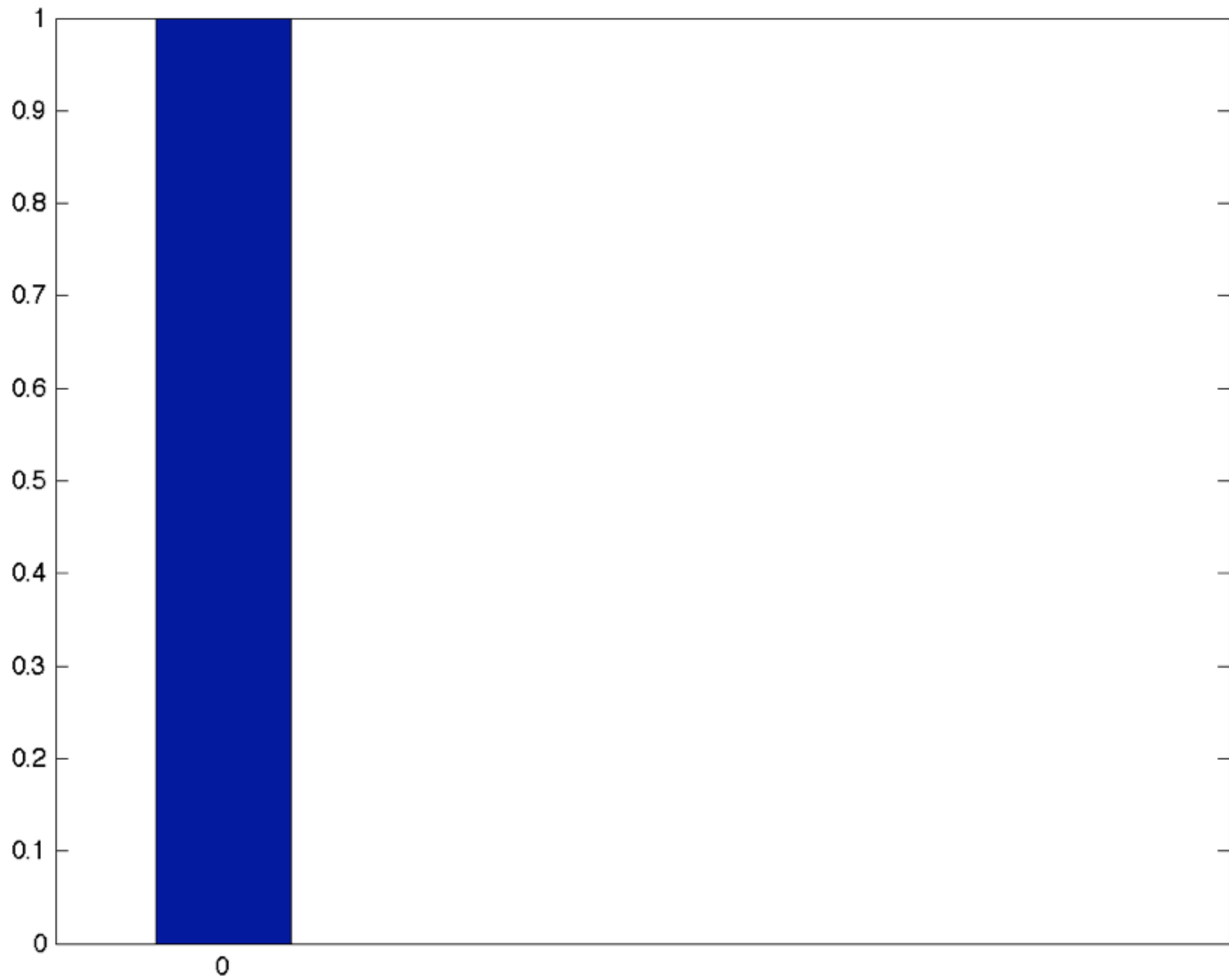
Only consistent
answer



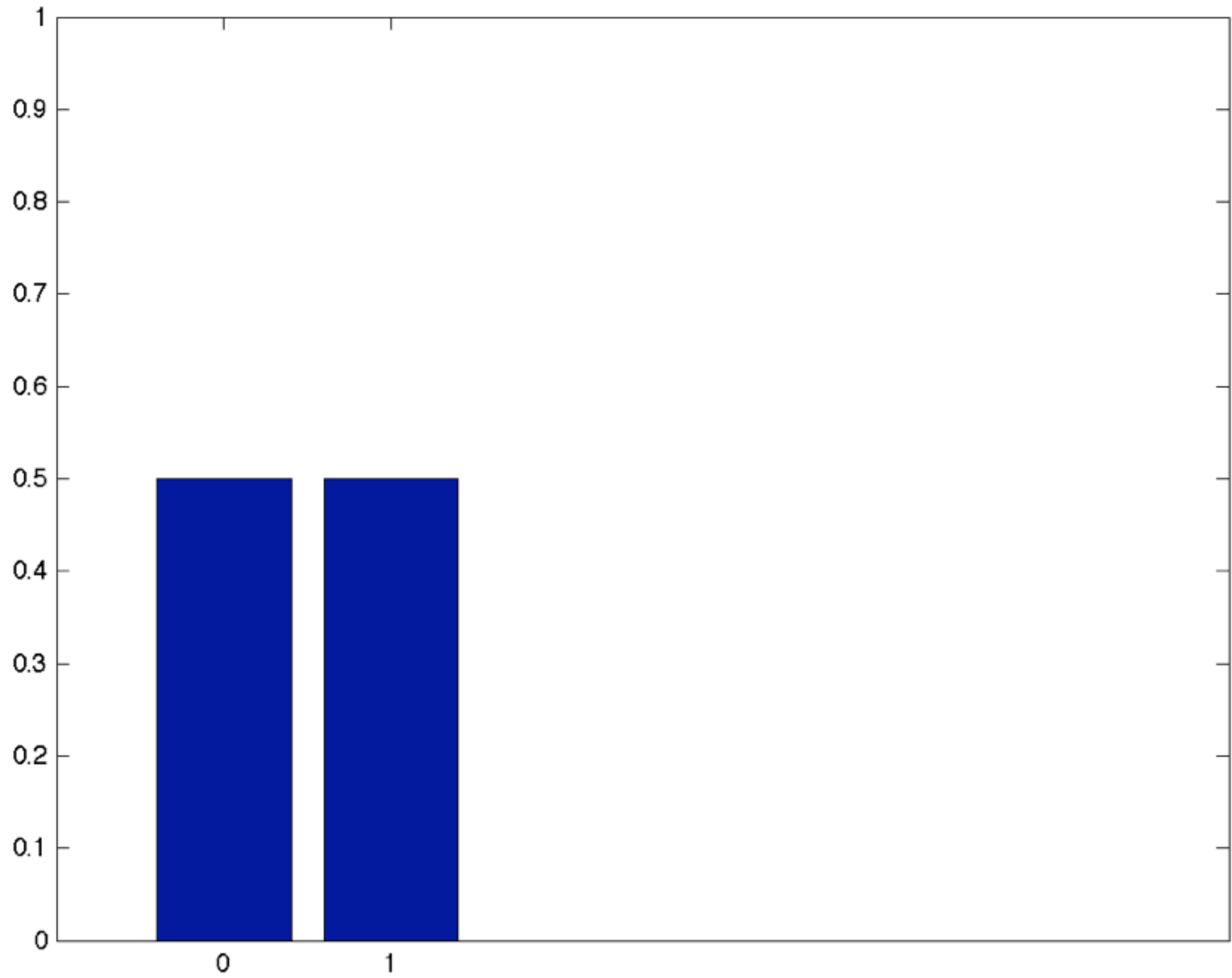
Alice's strategy

- Alice: split each bin into two equal size parts.
- Assume that n is a high enough power of 2 so that equal size splits are always possible.
- Better yet: assume that the answers are a continuous set of volume (measure) I and set the goal to reduce volume of consistent set to I/n

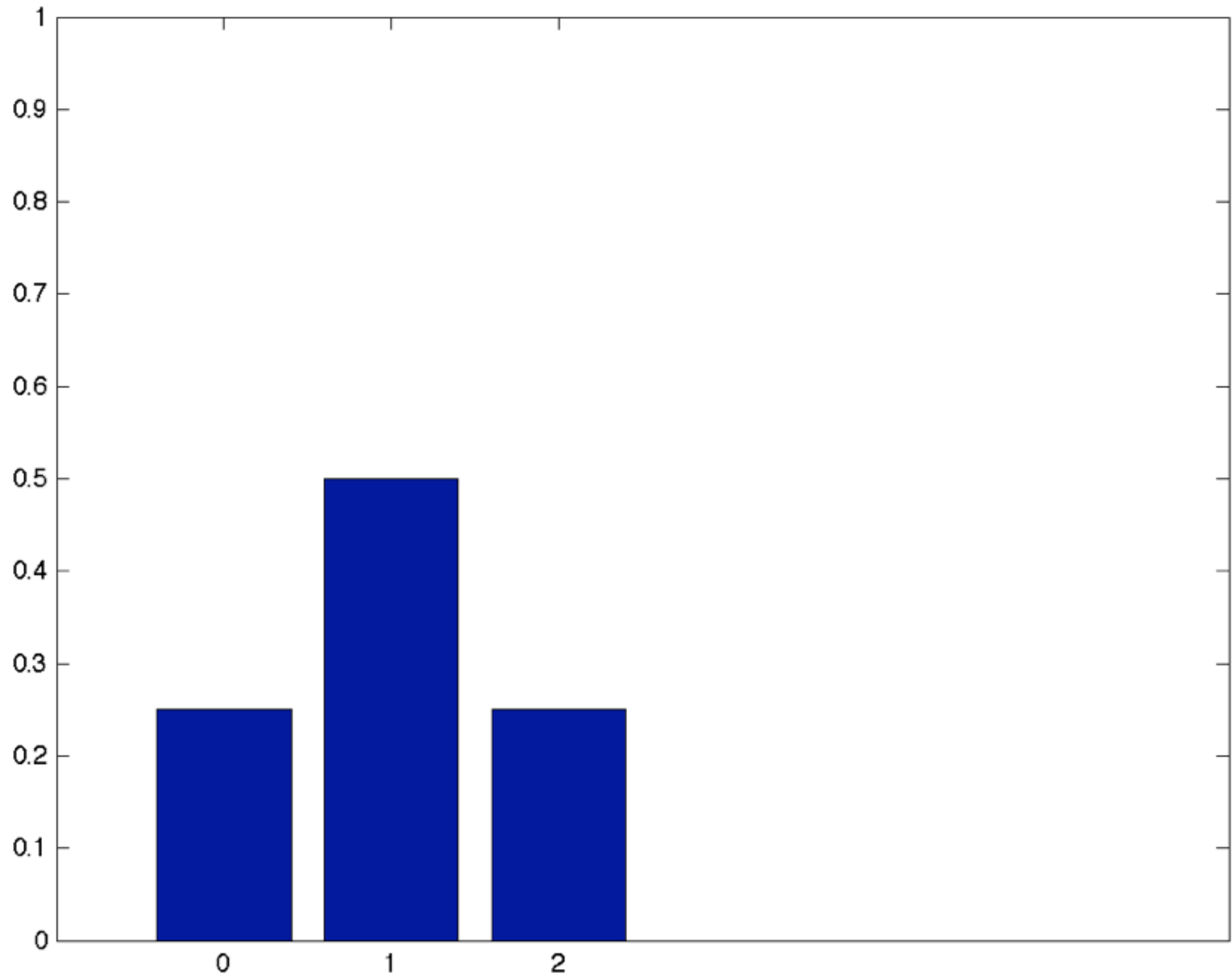
iteration=0 consistent=1



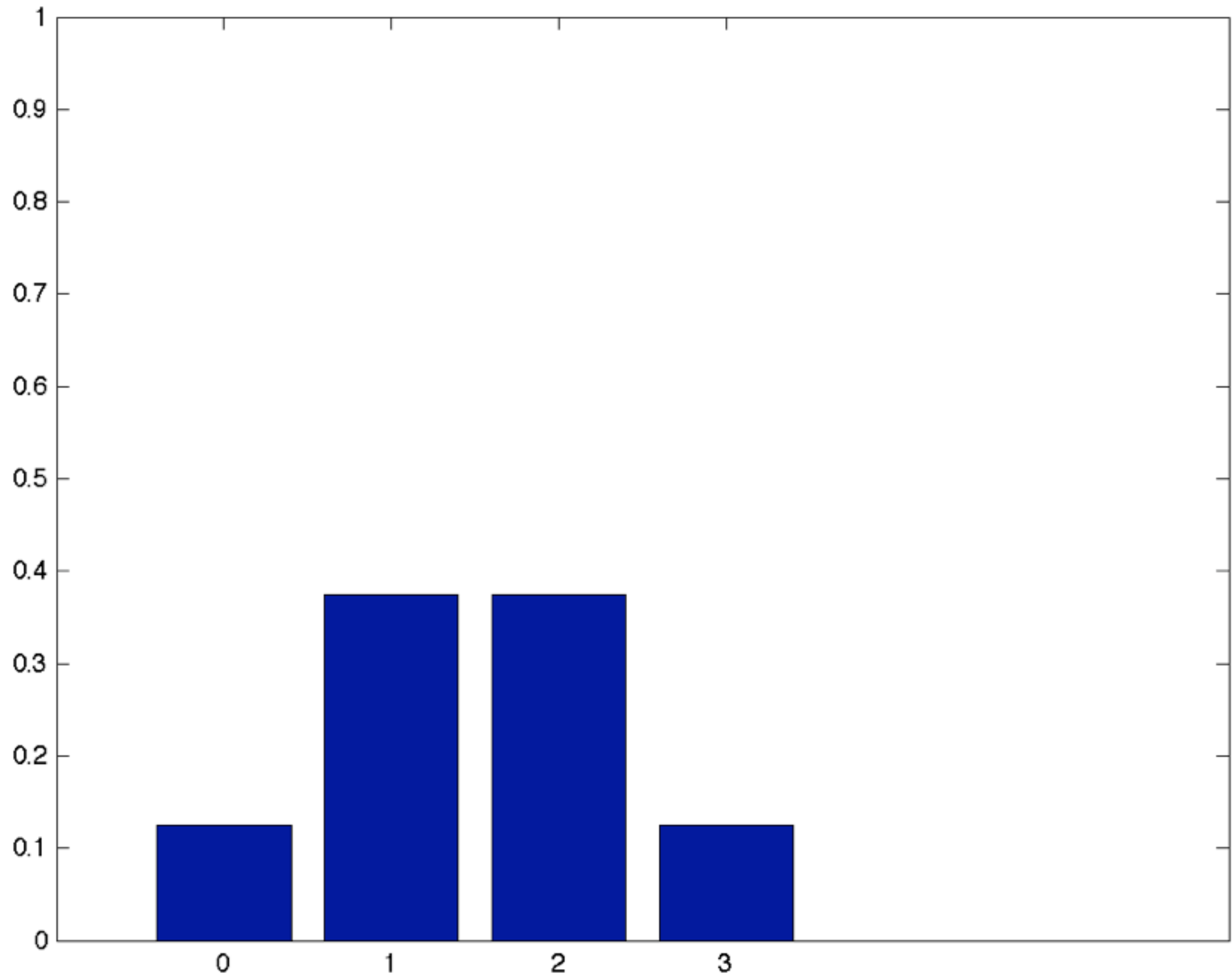
iteration=1 consistent=1



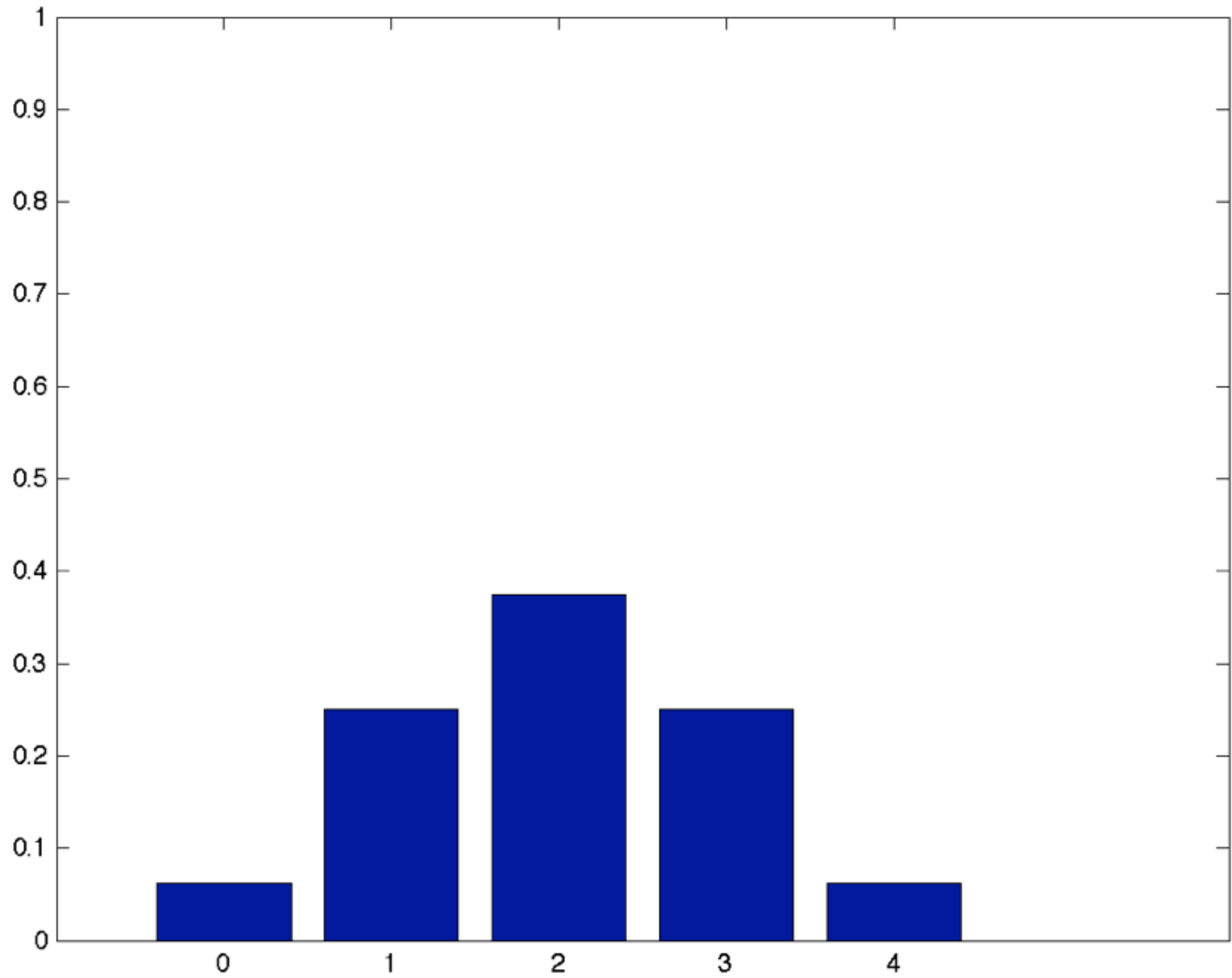
iteration=2 consistent=1



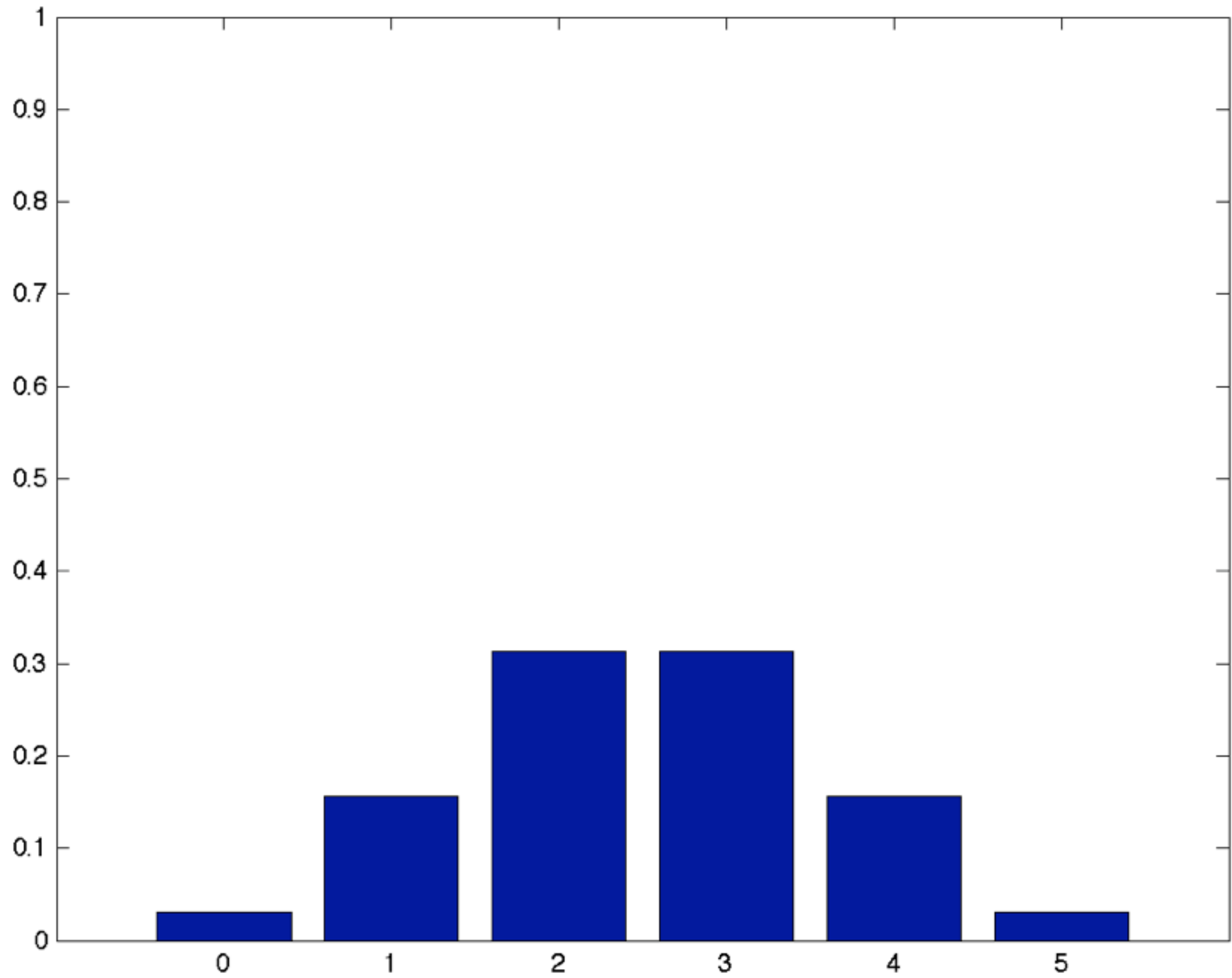
iteration=3 consistent=1



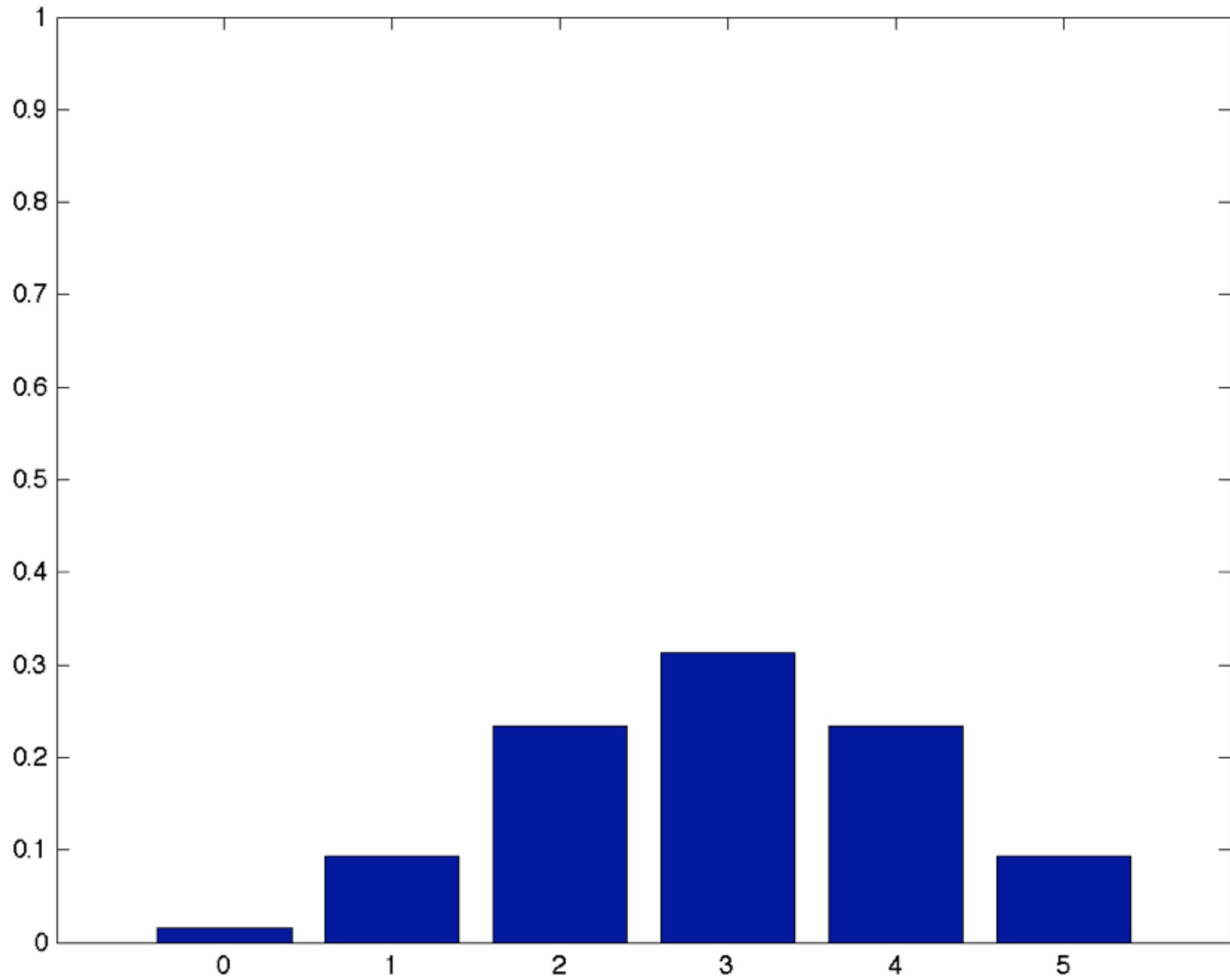
iteration=4 consistent=1



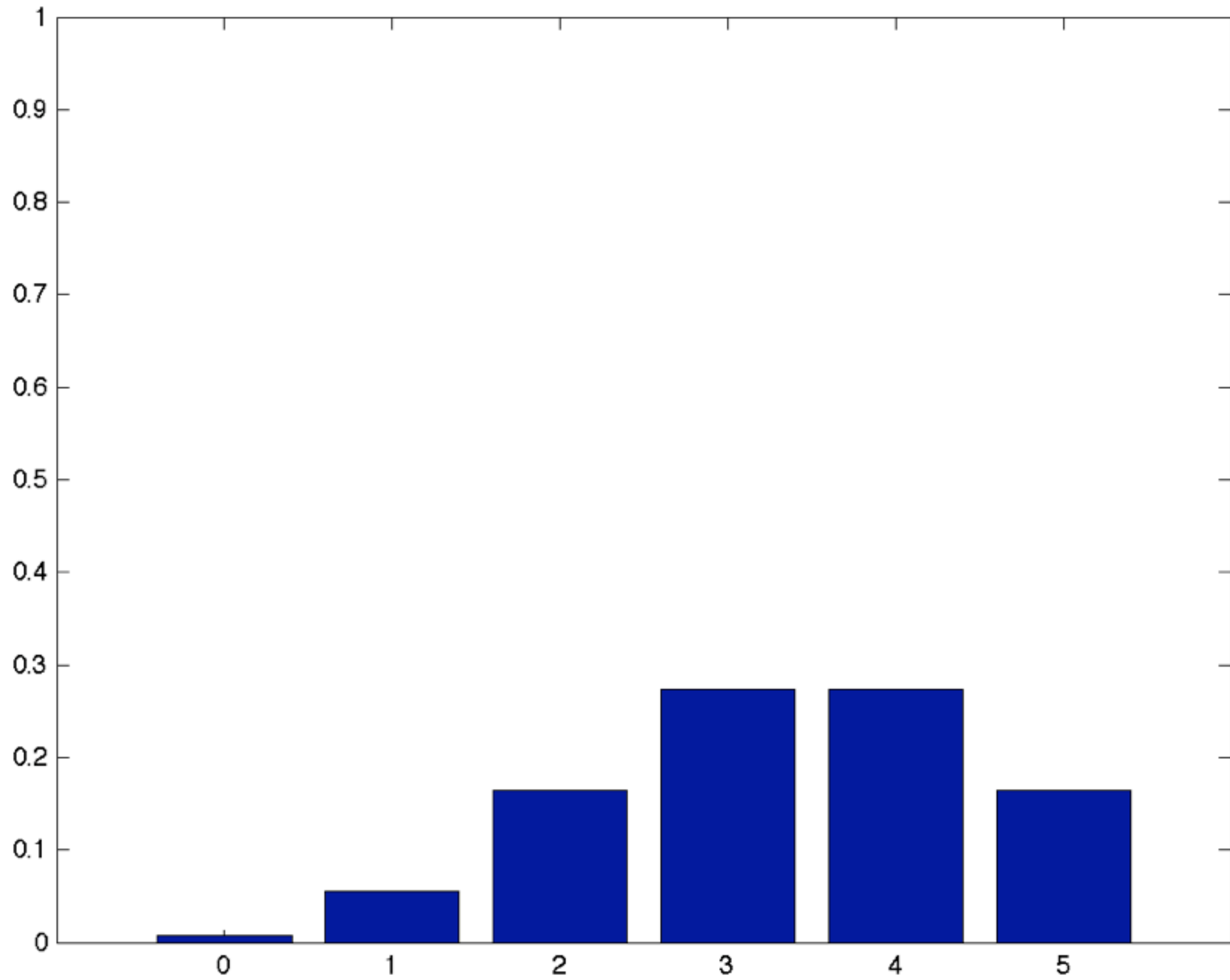
iteration=5 consistent=1



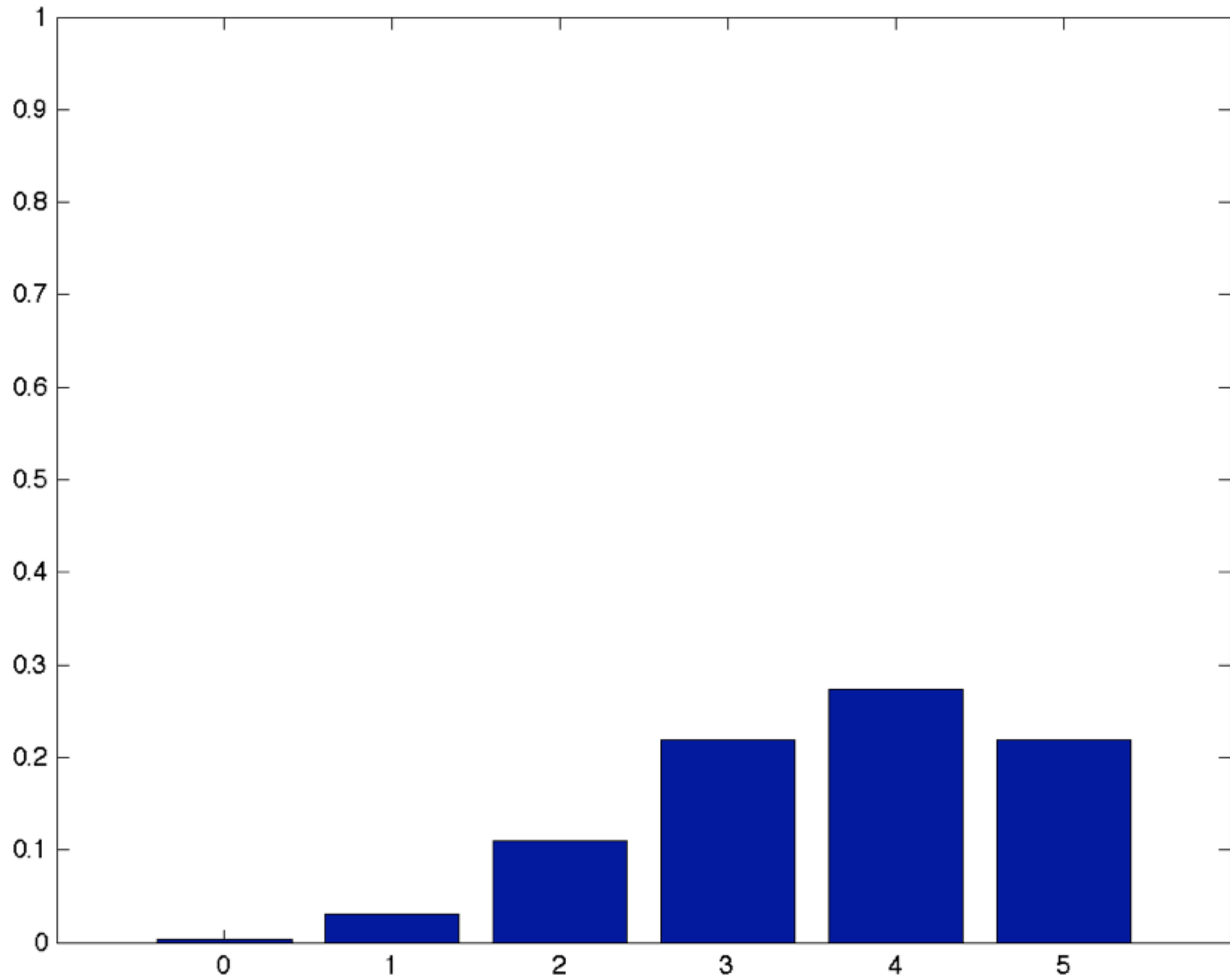
iteration=6 consistent=0.98438



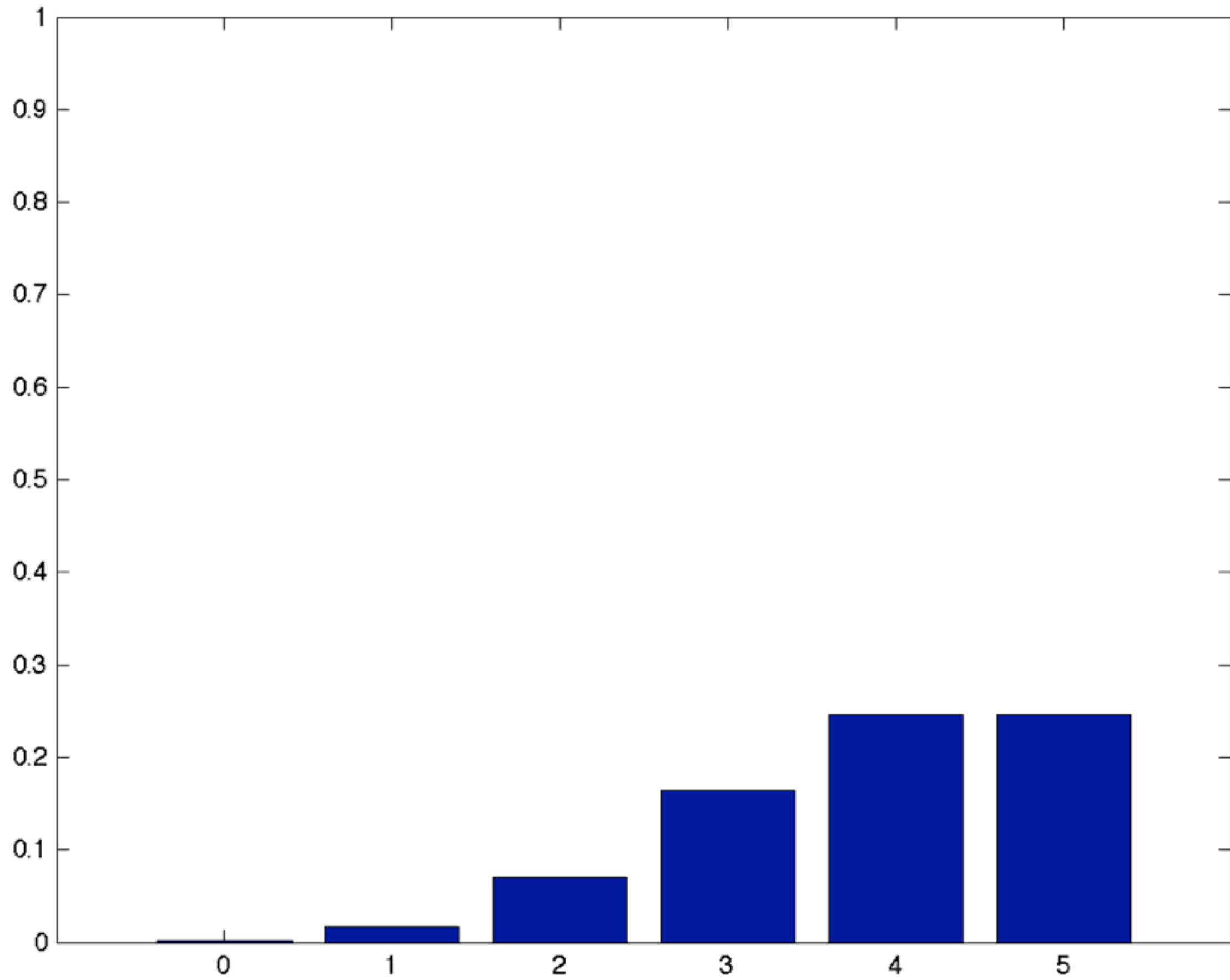
iteration=7 consistent=0.9375



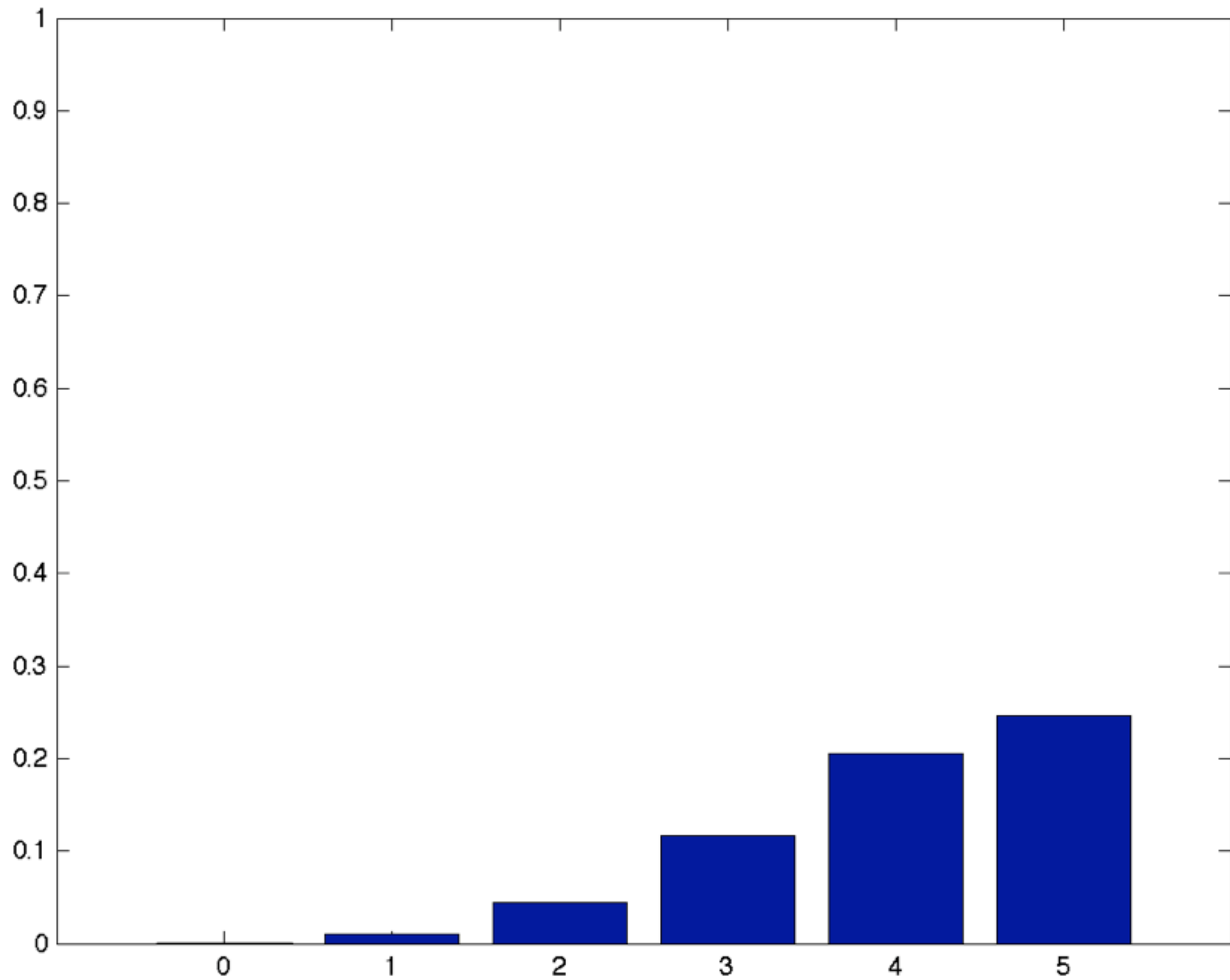
iteration=8 consistent=0.85547



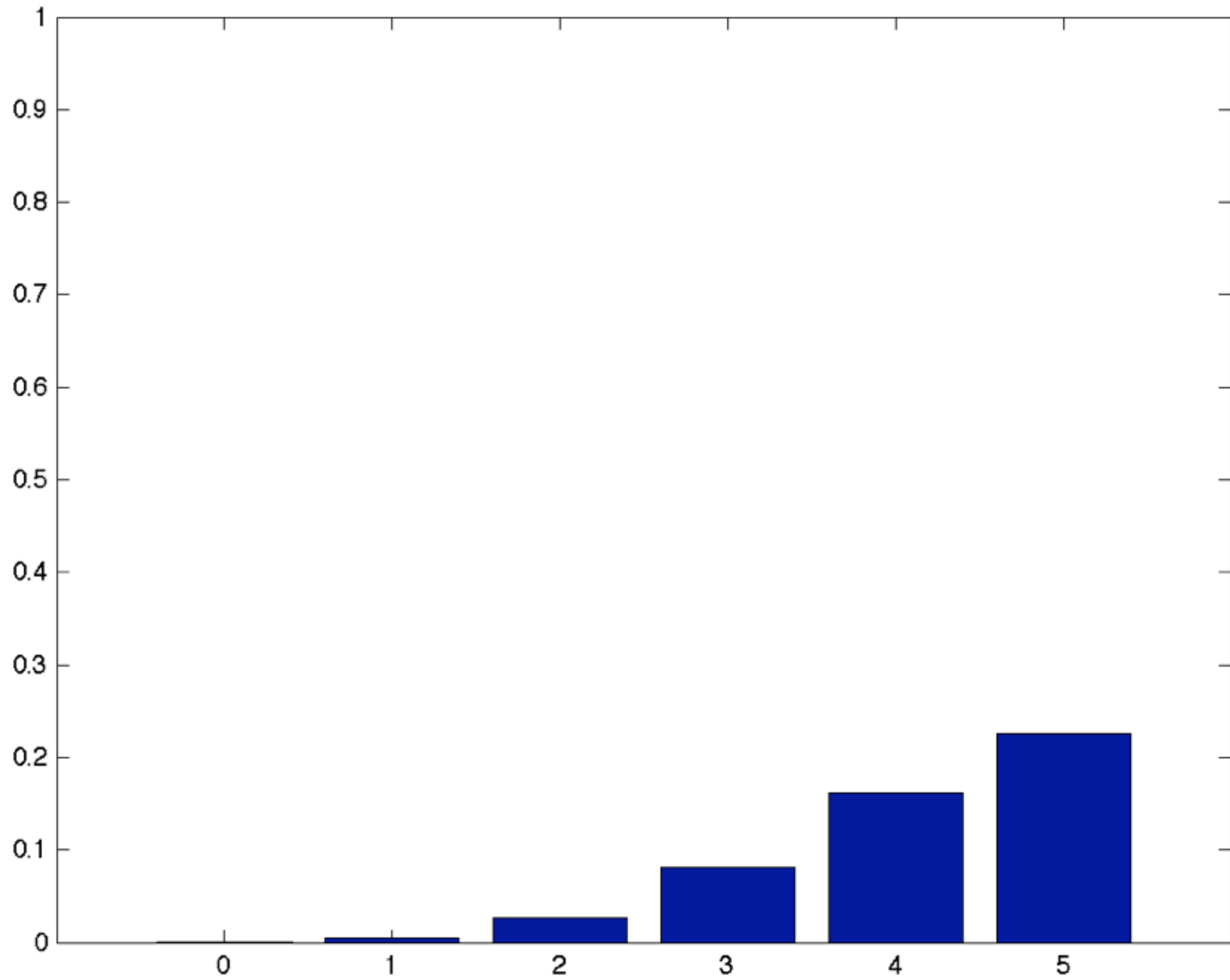
iteration=9 consistent=0.74609



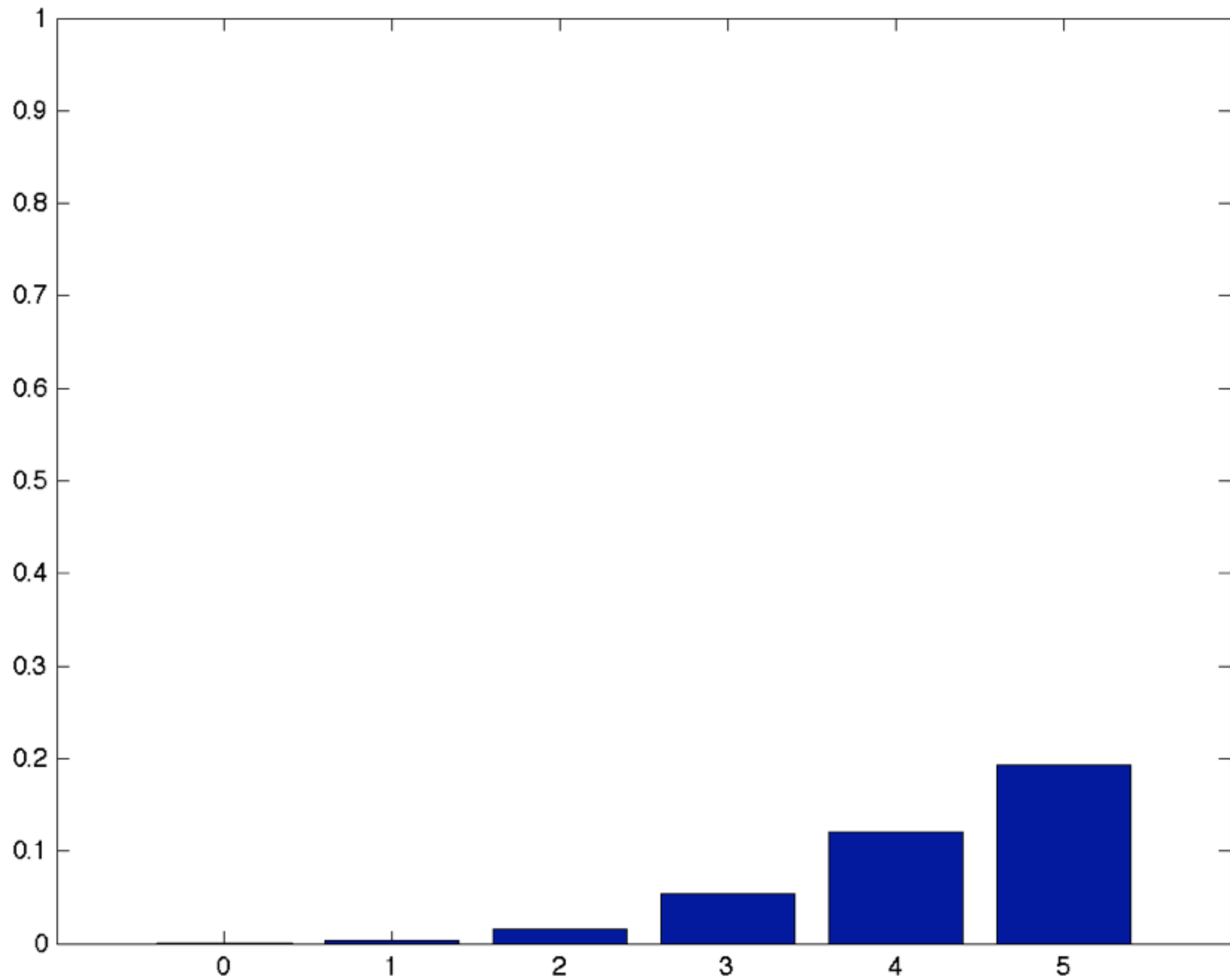
iteration=10 consistent=0.62305



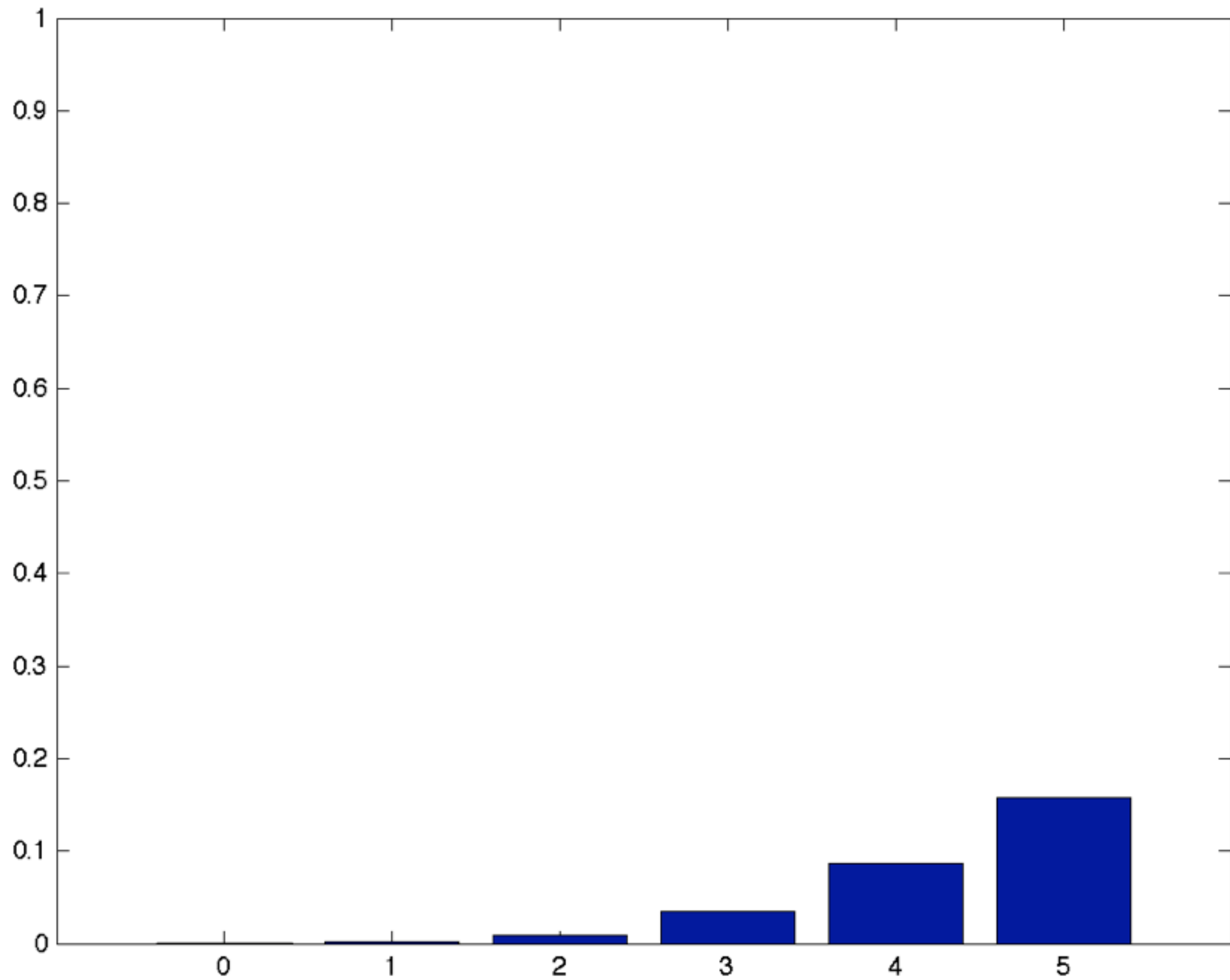
iteration=11 consistent=0.5



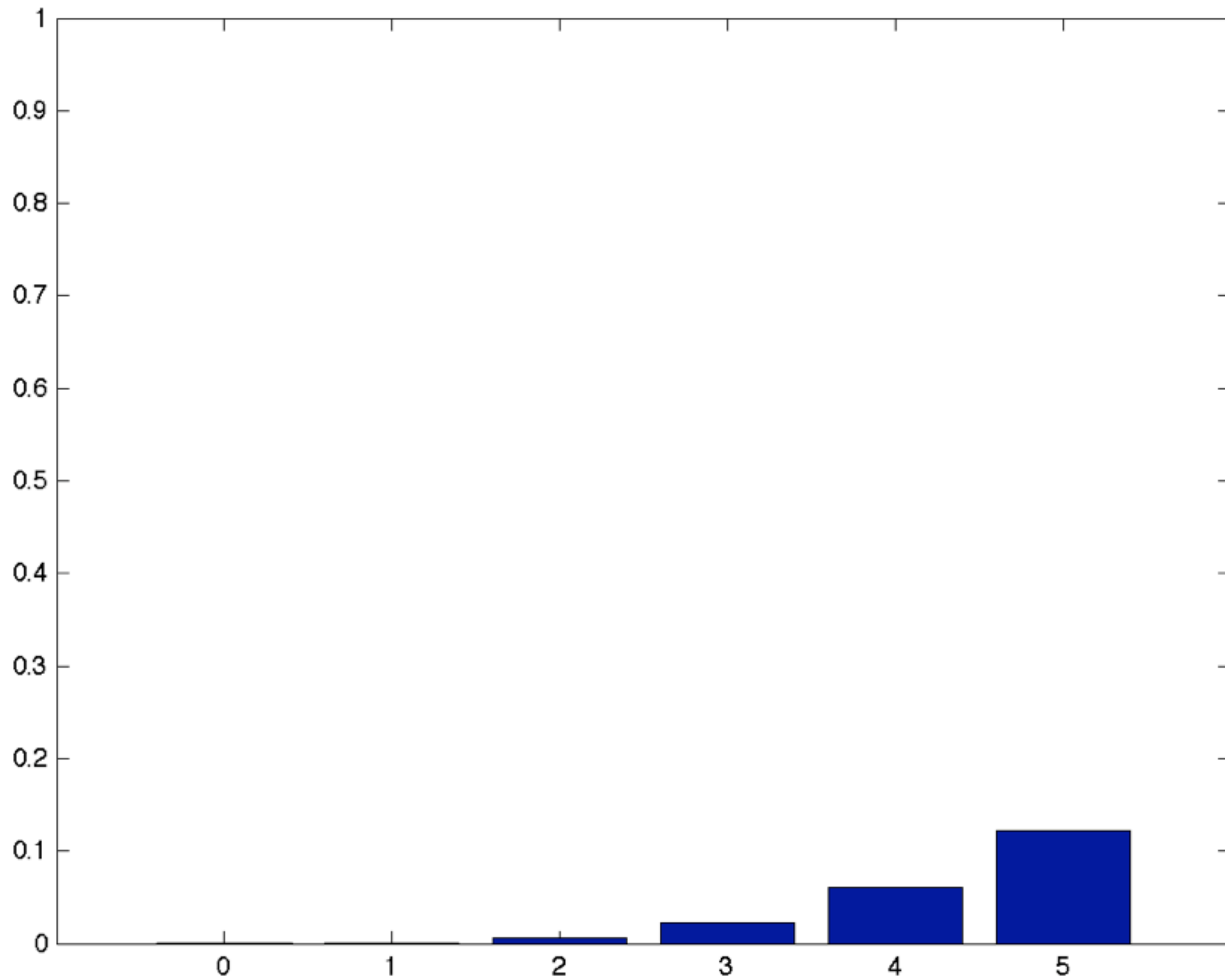
iteration=12 consistent=0.38721



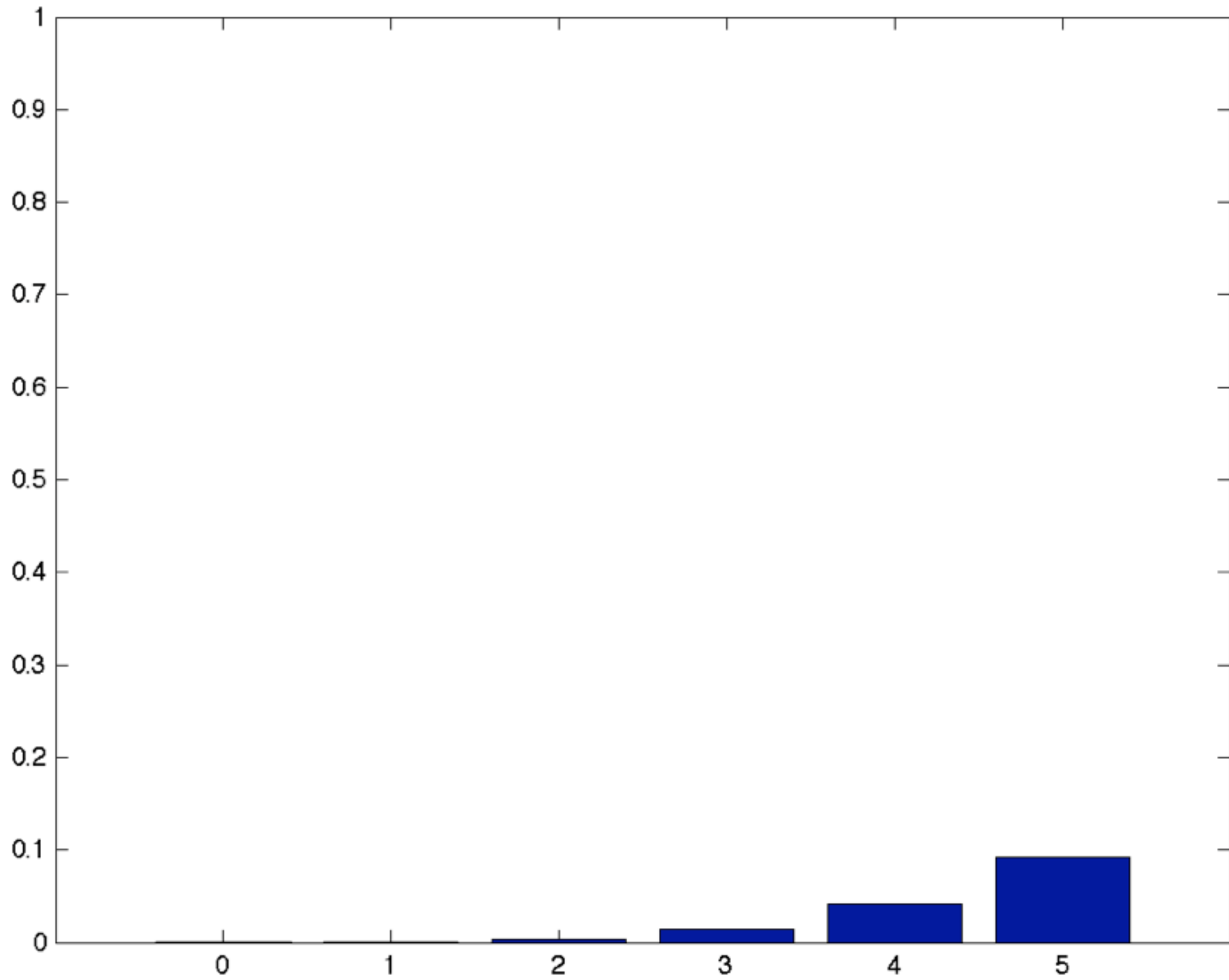
iteration=13 consistent=0.29053



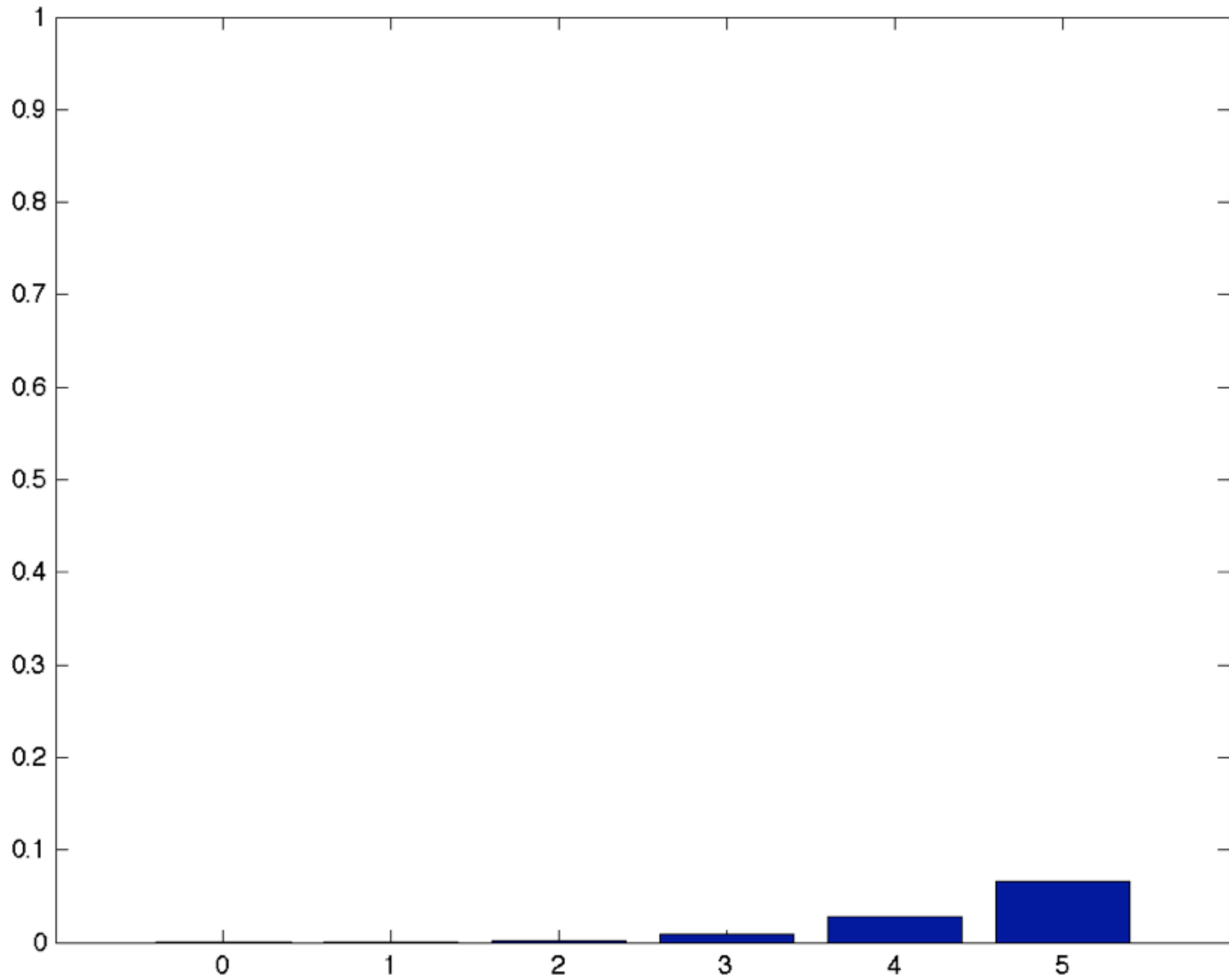
iteration=14 consistent=0.21198



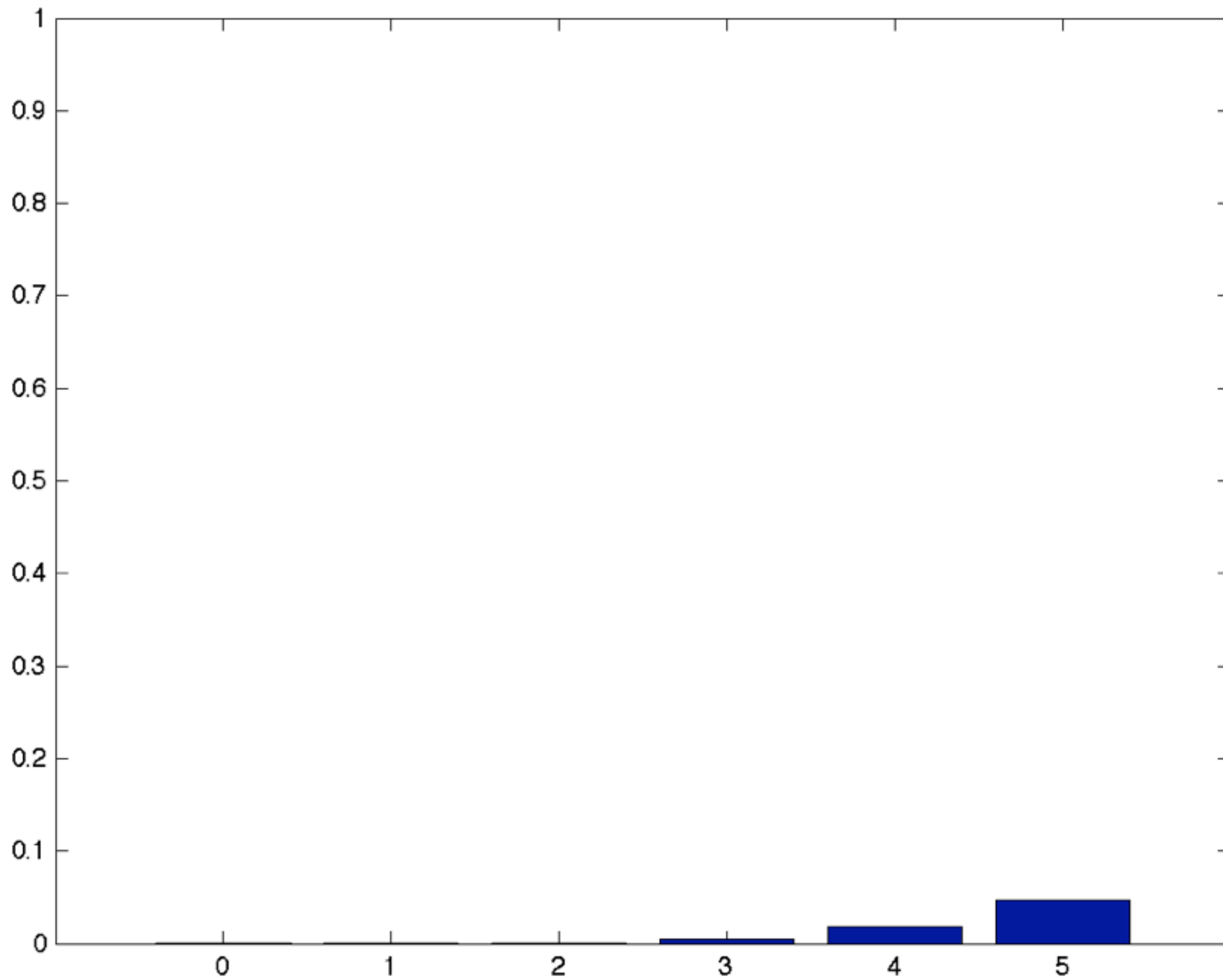
iteration=15 consistent=0.15088



iteration=16 consistent=0.10506



iteration=17 consistent=0.071732



The length of the game

- n = size of the set S
- k = maximal number of lies
- m = length of game
(number of questions until secret is found).

$$m = \max \left\{ q : \frac{1}{n} \leq 2^{-q} \binom{q}{\leq k} \right\}$$

$$m \leq 2k + 2\sqrt{k \ln n} + \log_2 n$$

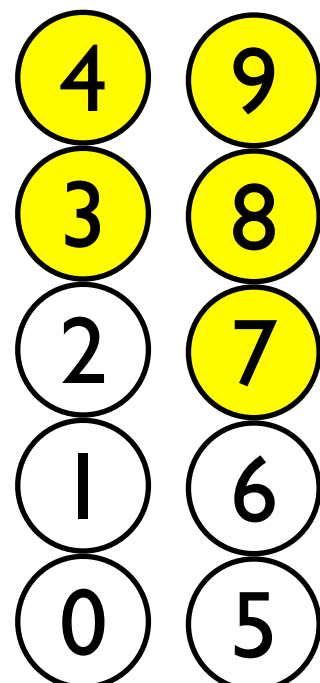
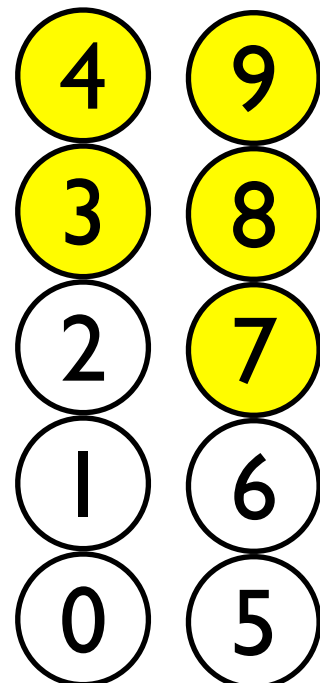
Taking advantage of Alice's mistakes

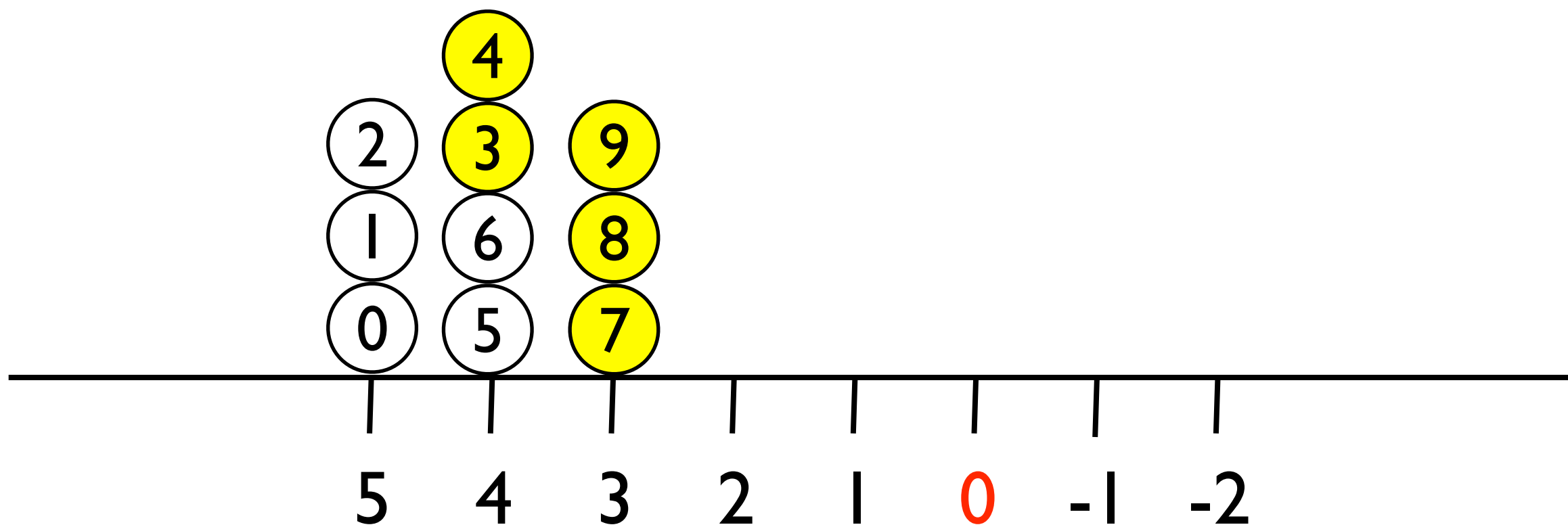
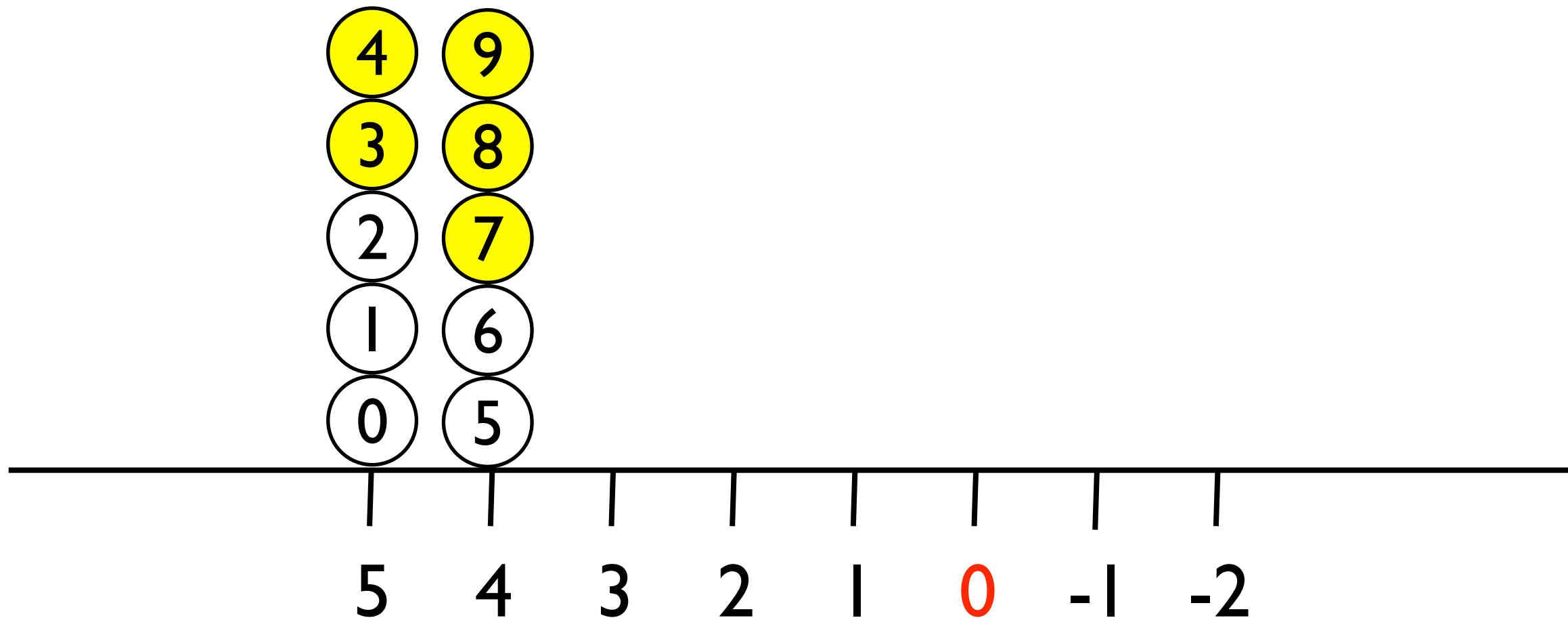
- If **Alice** splits each bin into two equal parts
 - **Bob's** choice does not matter
 - does not impact the next configuration
 - can be random with 1/2 1/2 probability.
- If splits are not equal - next configuration depends on **Bob's** choice - which one should he choose?

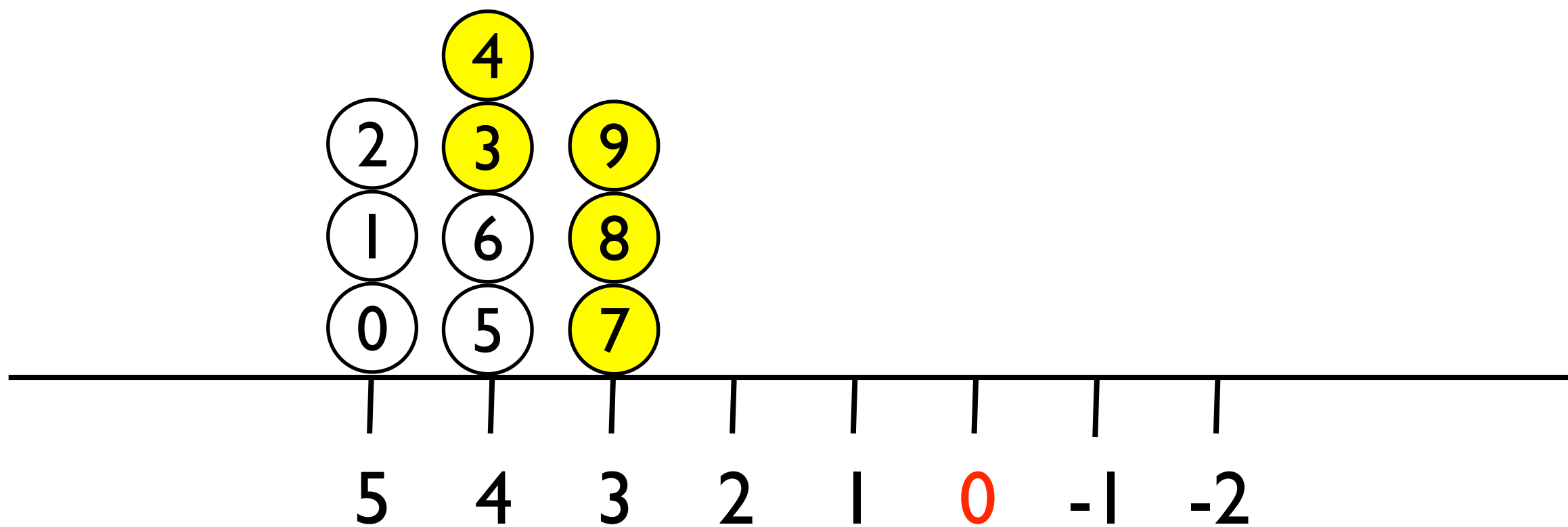
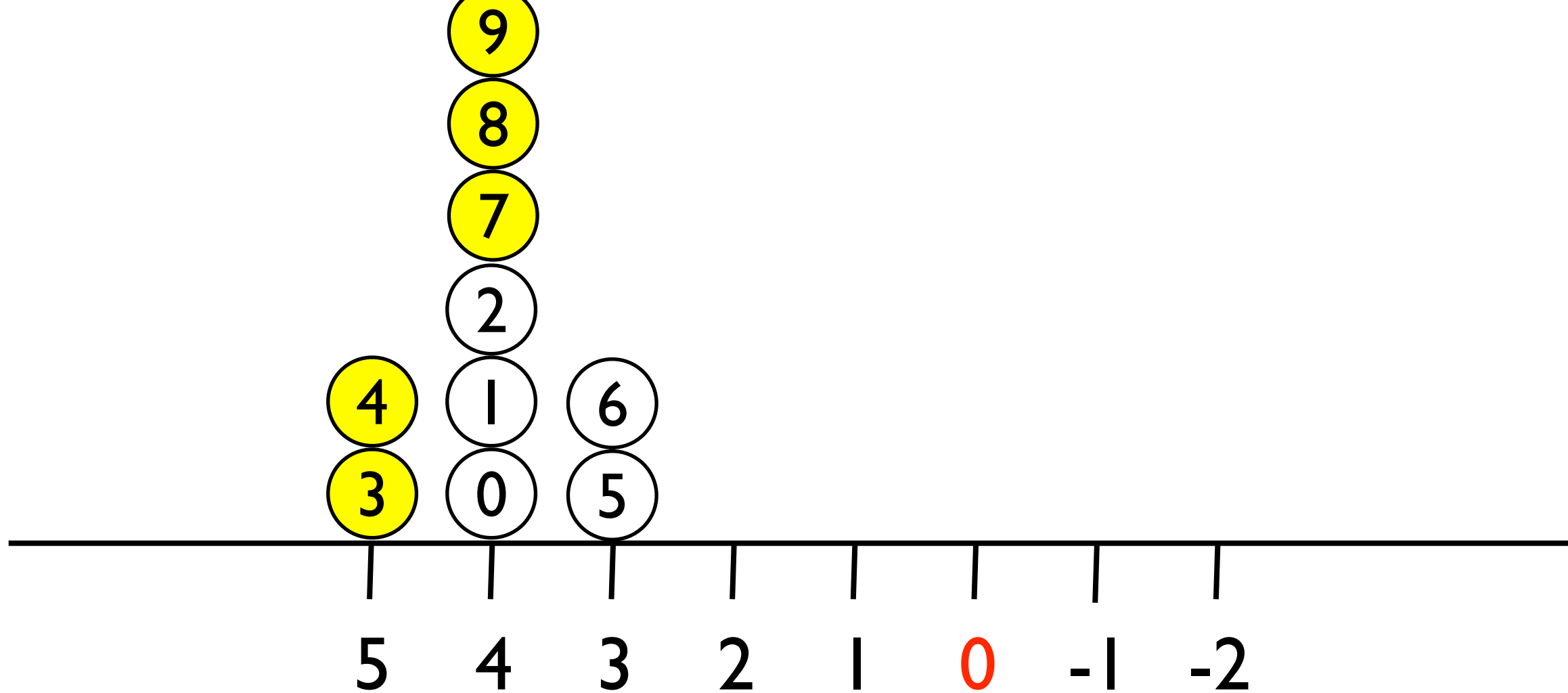
One step look-ahead

- Assume that Alice will play optimally from the next step and onward (worst case assumption).
- Compute the number of steps till game ends.
- Choose configuration that makes game longer.

Make bins continuous







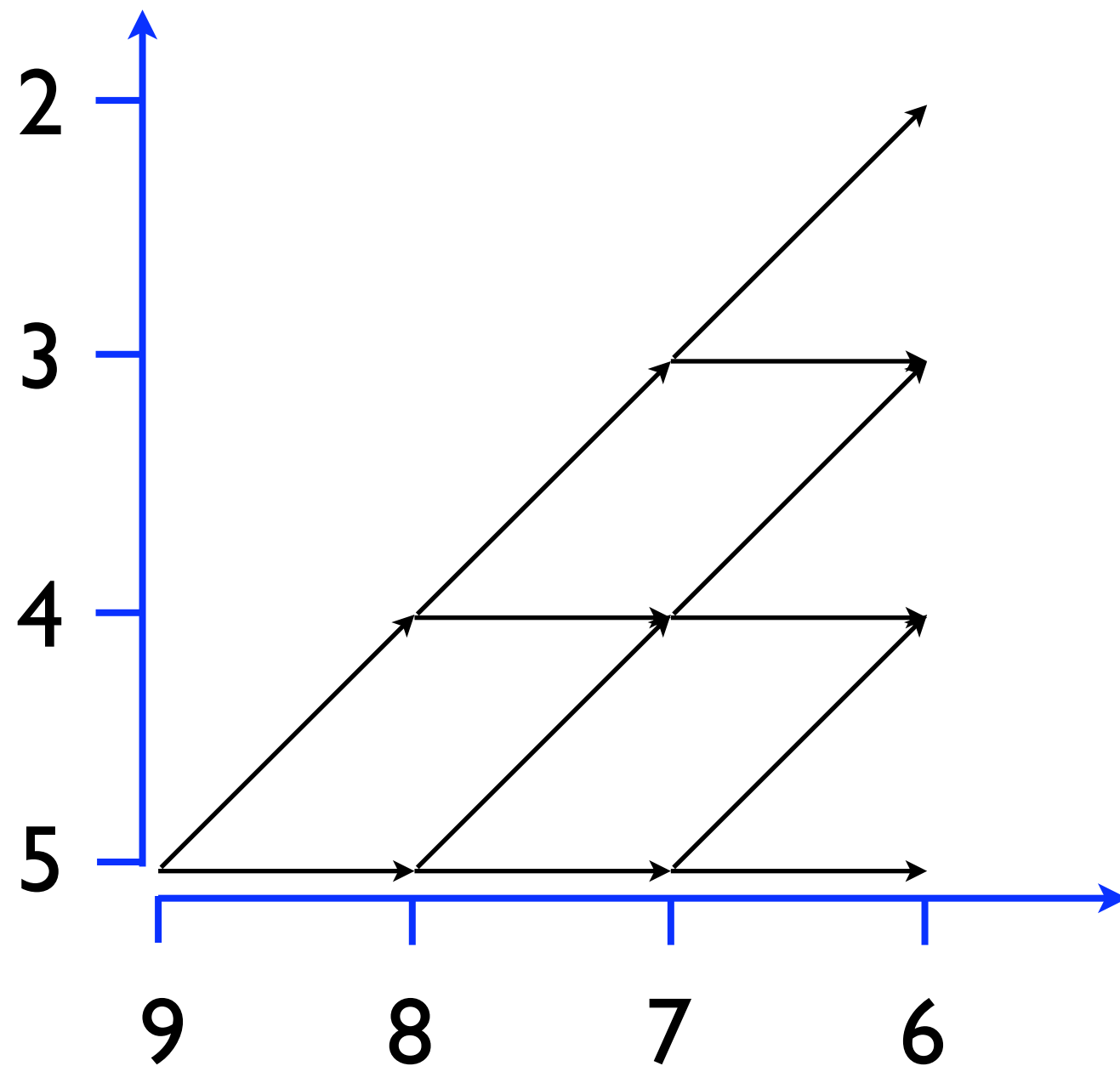
A more refined strategy for bob.

- In some cases, the length of the game will not differ, so bob's choice does not matter.
- Let r be the number of remaining steps before Alice made her suboptimal move.
- Let V_1, V_2 be the volume of the consistent set after $r-1$ optimal steps starting at each of the two configurations.
- Bob chooses the configuration with the larger volume.
- If $V_{\max} \geq 1/n$, then number of remaining steps ($r-1$) is decreased by (an additional) 1.

The game lattice

remaining
lies

i



r remaining
iterations

Potential

$\psi(i, r)$ the fraction elements with i remaining lies that will be consistent after r iterations of optimal play.

Equivalently: probability of getting at most i heads in r flips of a fair coin

$$\psi(i, 0) = \begin{cases} 1 & i \geq 0 \\ 0 & i < 0 \end{cases}$$

$$\psi(i, r) = 2^{-r} \binom{r}{\leq i}$$

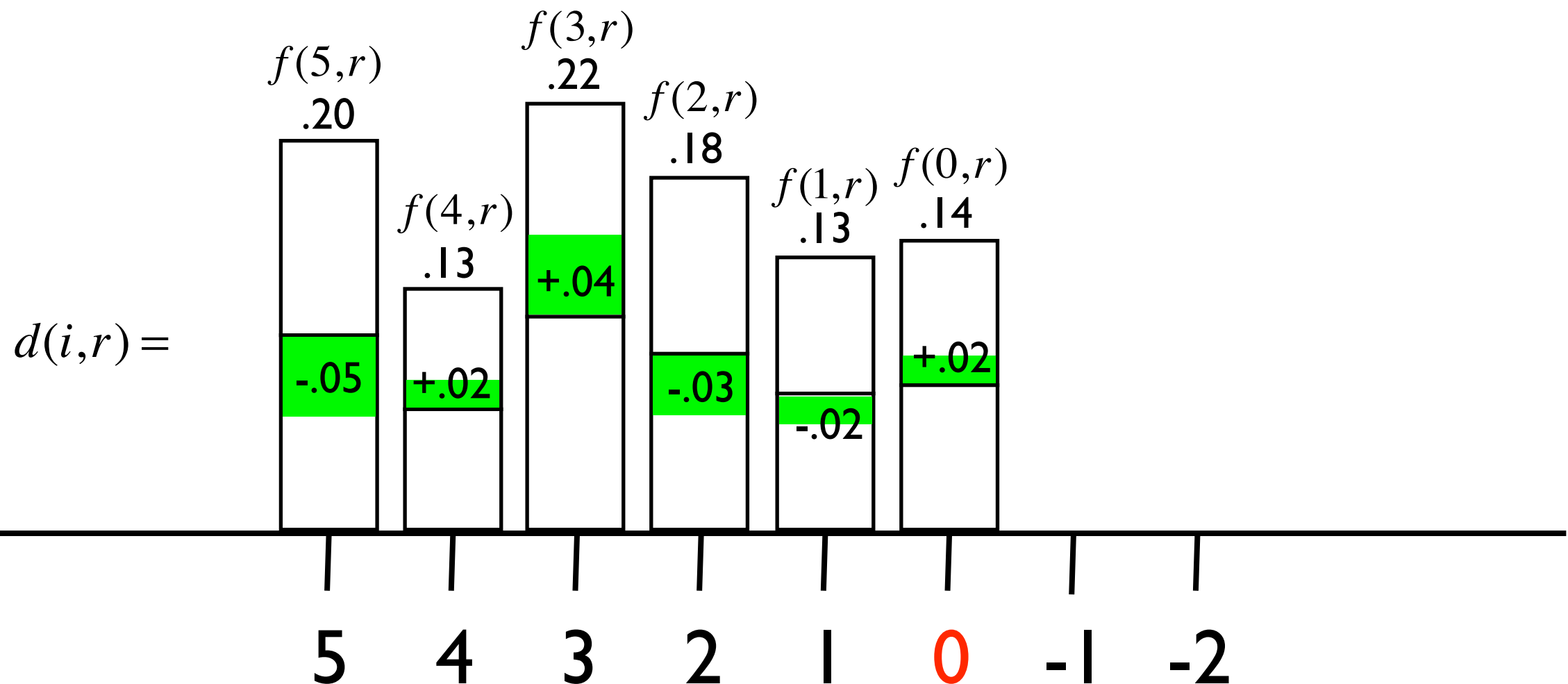
$$\psi(i, r+1) = \frac{\psi(i, r) + \psi(i+1, r)}{2}$$

notation

$f(i,r)$ The fraction of answers in bin i when r iterations remain

$d(i,r)$ The difference between the two parts chosen by Alice in bin i when r steps remain

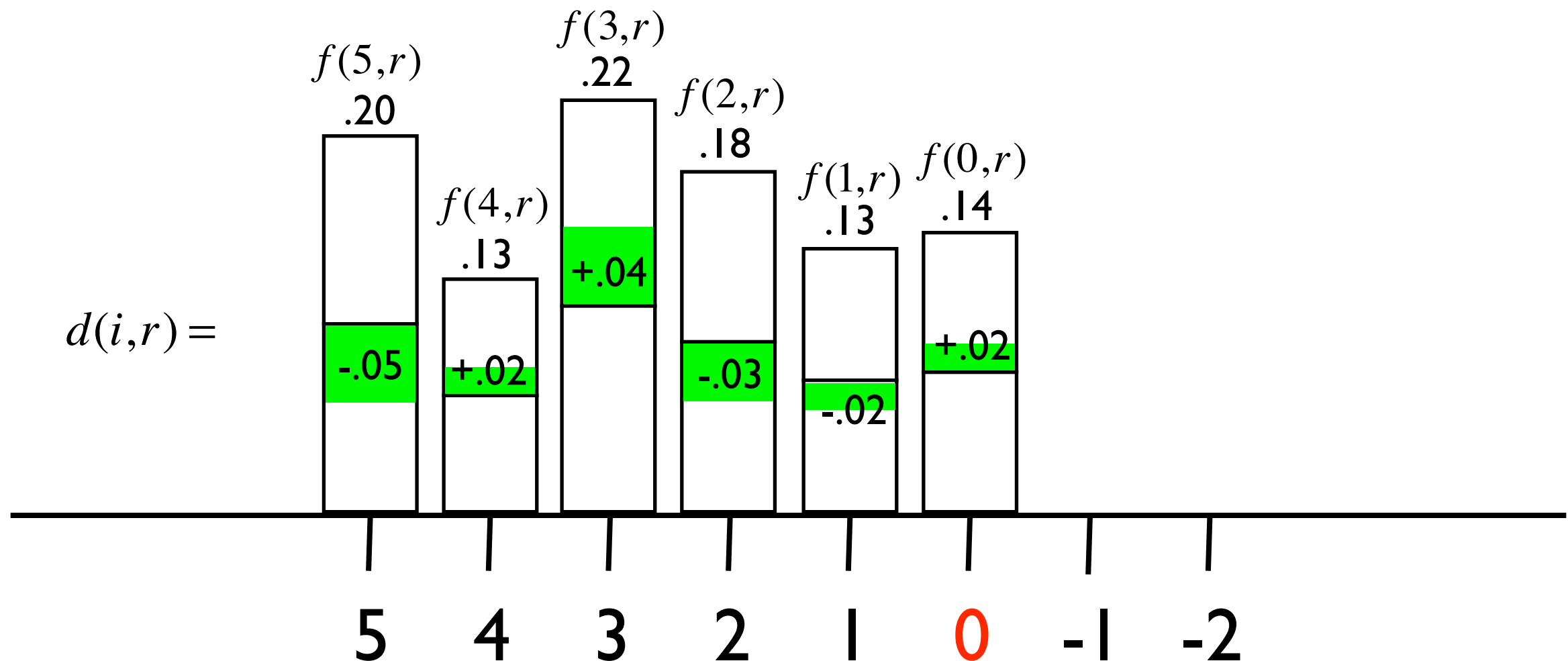
$b(r)$ Bob's choice (+1 or -1)



$$f(i, r-1) = \frac{f(i-1, r) + f(i, r)}{2} + b(r)(d(i-1, r) - d(i, r))$$

Total potential $V = \Psi(r) = \sum_i f(i, r) \psi(i, r)$

$$\Psi(r-1) = \sum_i \left[\frac{f(i-1, r) + f(i, r)}{2} + b(r)(d(i-1, r) - d(i, r)) \right] \psi(i, r-1)$$



$$f(i, r-1) = \frac{f(i-1, r) + f(i, r)}{2} + b(r)(d(i-1, r) - d(i, r))$$

Total potential $V = \Psi(r) = \sum_i f(i, r)\psi(i, r)$

$$\Psi(r-1) = \sum_i \left[\frac{f(i-1, r) + f(i, r)}{2} + b(r)(d(i-1, r) - d(i, r)) \right] \psi(i, r-1)$$

$$\Psi(r-1) = \sum_i \left[f(i, r) \frac{\psi(i, r-1) + \psi(i+1, r-1)}{2} + b(r)d(i, r)(\psi(i, r-1) - \psi(i+1, r-1)) \right]$$

$$w(i, r) \doteq \psi(i, r-1) - \psi(i+1, r-1)$$

$$\Psi(r-1) = \sum_i [f(i, r)\psi(i, r) + b(r)d(i, r)w(i, r)]$$

$$f(i, r-1) = \frac{f(i-1, r) + f(i, r)}{2} + b(r)(d(i-1, r) - d(i, r))$$

Total potential $V = \Psi(r) = \sum_i f(i, r)\psi(i, r)$

$$\Psi(r-1) = \sum_i \left[\frac{f(i-1, r) + f(i, r)}{2} + b(r)(d(i-1, r) - d(i, r)) \right] \psi(i, r-1)$$

$$\Psi(r-1) = \sum_i \left[f(i, r) \frac{\psi(i, r-1) + \psi(i+1, r-1)}{2} + b(r)d(i, r)(\psi(i, r-1) - \psi(i+1, r-1)) \right]$$

$$w(i, r) \doteq \psi(i, r-1) - \psi(i+1, r-1)$$

$$\Psi(r-1) = \sum_i [f(i, r)\psi(i, r) + b(r)d(i, r)w(i, r)]$$

$$\Psi(r-1) = \Psi(r) + b(r) \sum_i d(i, r)w(i, r)$$

$$\Psi(r-1) = \Psi(r) + b(r) \sum_i d(i,r) w(i,r)$$

$b(r)$ **Bob's** choice (+1 or -1)

$d(i,r)$ **Alice's** choice

$$w(i,r) \doteq \psi(i,r-1) - \psi(i+1,r-1) = 2^{-(r-1)} \binom{r-1}{i}$$

Bob wants to lengthen the game
= increase the potential

$$b(r) = \text{sign} \left(\sum_i d(i,r) w(i,r) \right)$$

Application to online learning with expert advice

[Cesa-Bianchi, Freund, Helmbold, Warmuth 1996]

- n experts
- at each iteration:
 - Each expert makes a $+1/-1$ prediction
 - Master makes $+1/-1$ prediction
 - Nature generates $+1/-1$ output
- One of the experts makes $\leq k$ mistakes
- **Goal:** minimize number of mistakes of the master.

The halving algorithm

- $k=0$ - one of the experts is perfect
- Master's strategy: predict according to the majority of consistent experts
- If Master makes a mistake - consistent experts halved.
- Master makes at most $\log_2 n$ mistakes
- optimal when n is a power of two or continuous
- Equivalent to 20 questions with no lies.

Expert making $\leq k$ mistakes

Expert = Chip

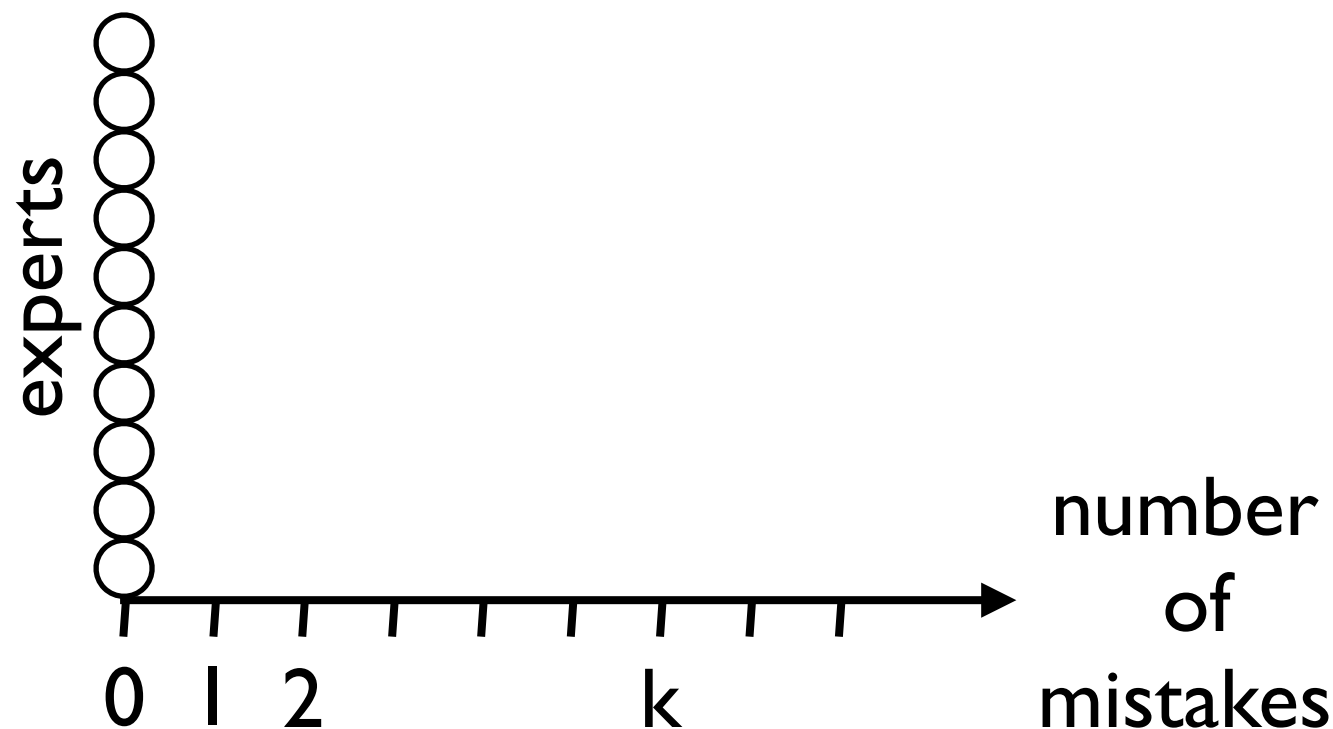
Bin = number of mistakes made by expert

Alice chooses predictions for all experts

Bob chooses master's predictions

wlog master makes a mistake on every round

Bob's goal: make the game as short as possible



Expert making $\leq k$ mistakes

Expert = Chip

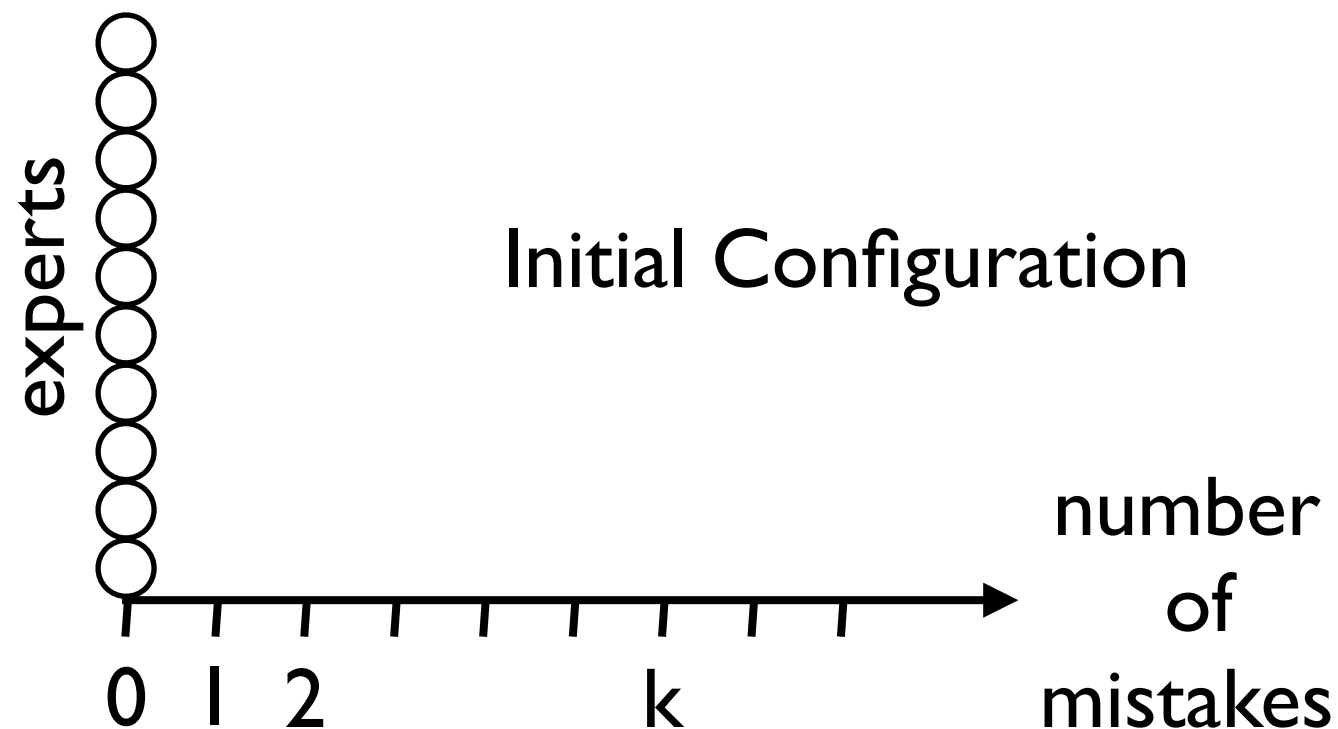
Bin = number of mistakes made by expert

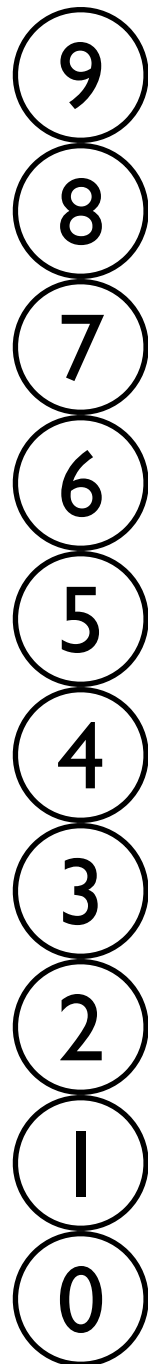
Alice chooses predictions for all experts

Bob chooses master's predictions

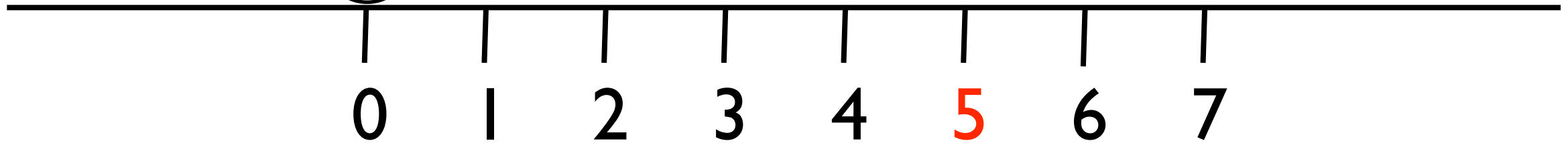
wlog master makes a mistake on every round

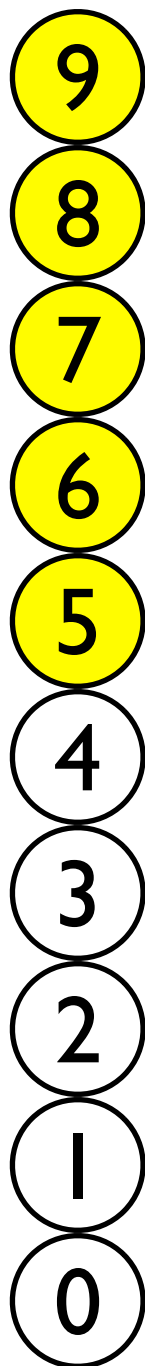
Bob's goal: make the game as short as possible



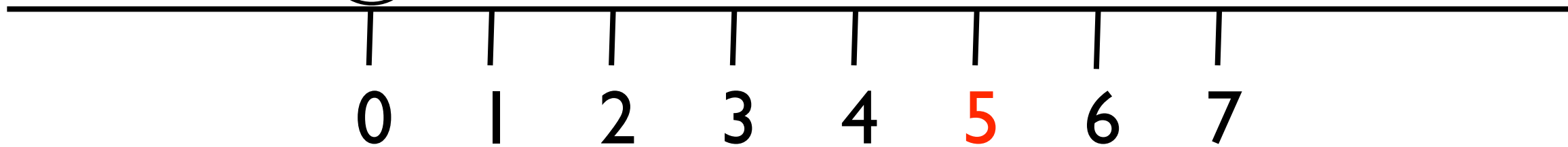


Initial configuration

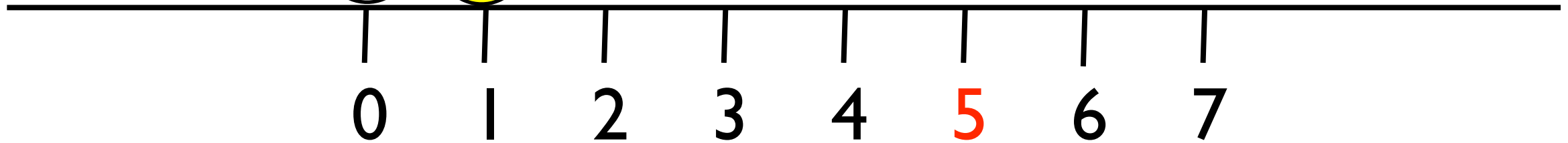
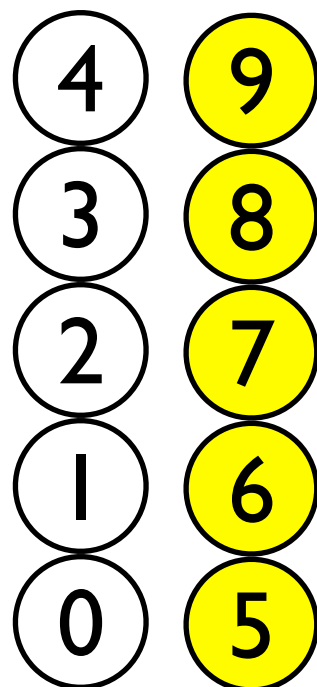




Alice's move



Bob's move



Configuration I

4	9
3	8
2	7
1	6
0	5

0

1

2

3

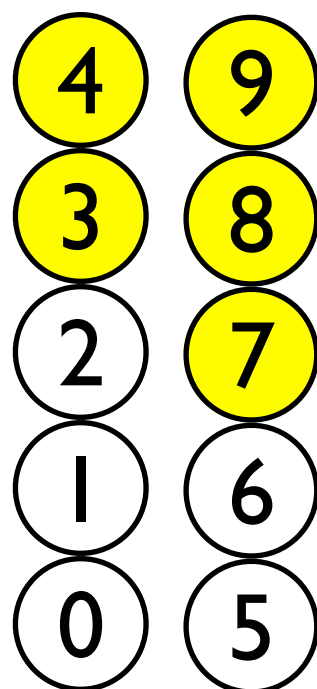
4

5

6

7

Alice's move



0

1

2

3

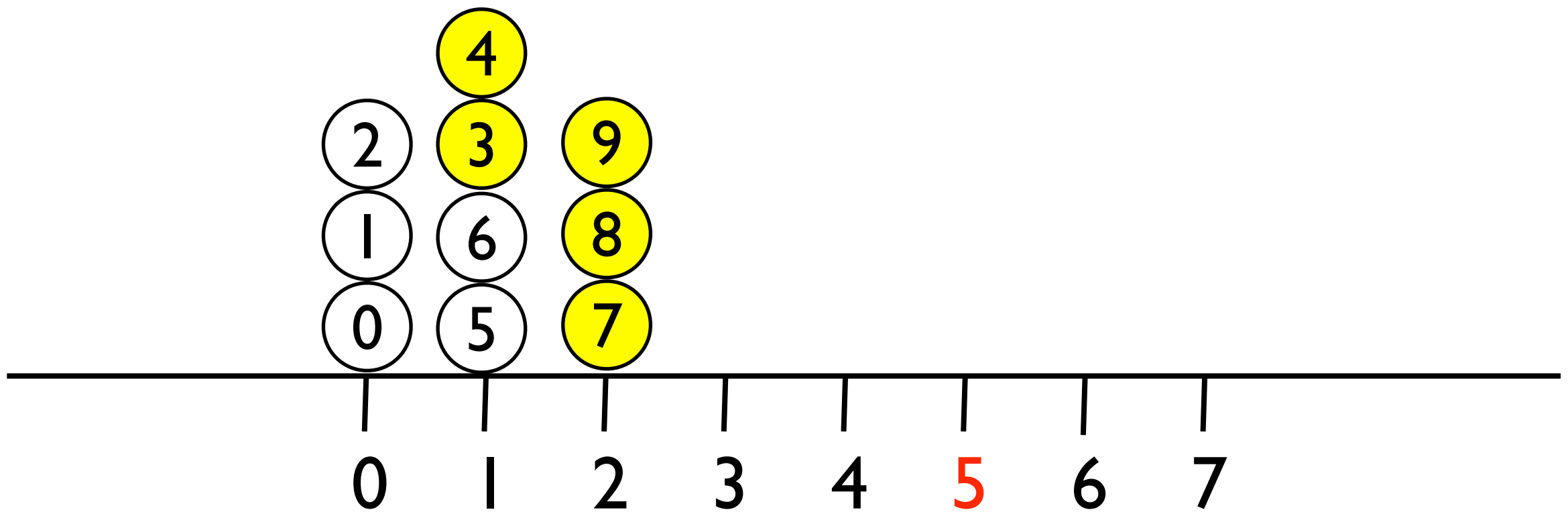
4

5

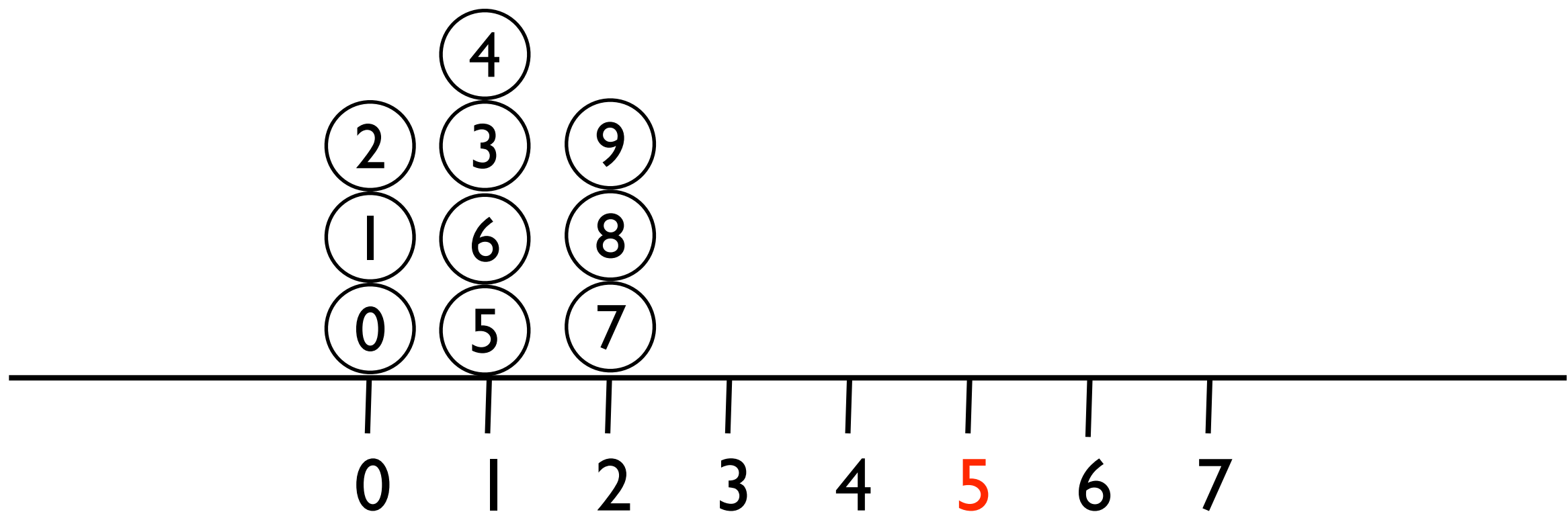
6

7

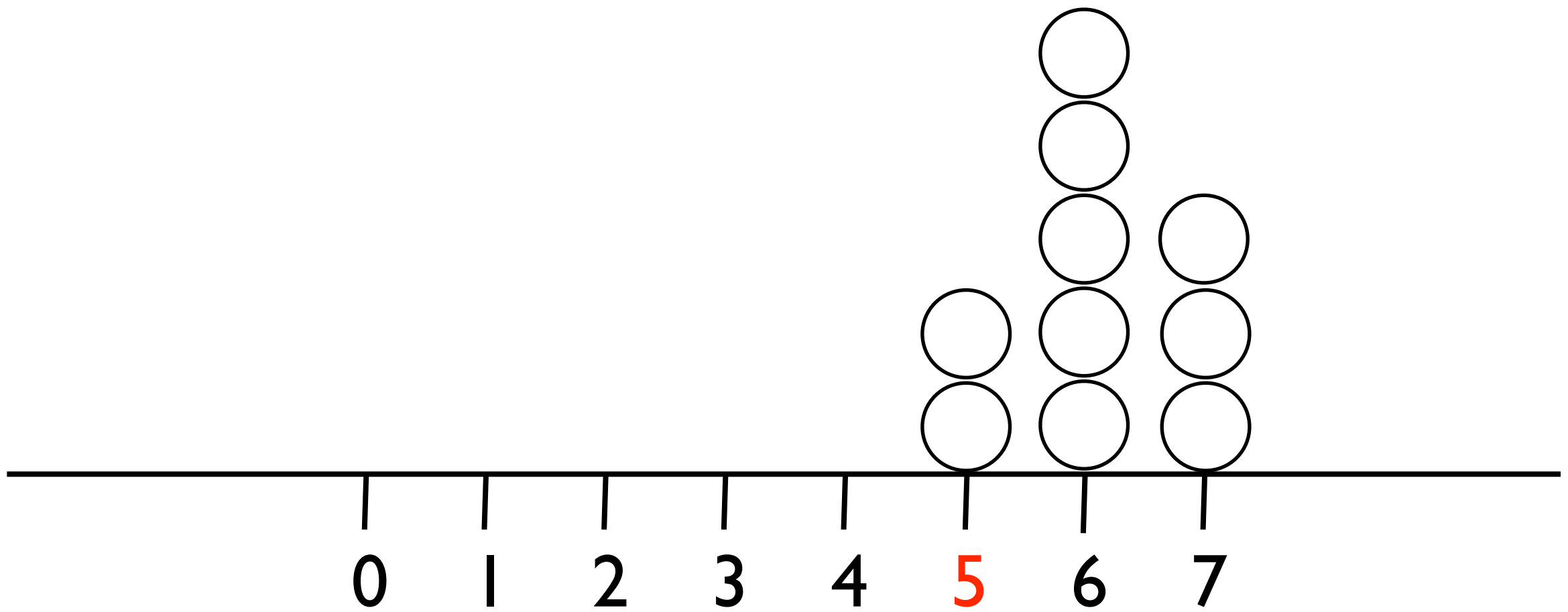
Bob's move



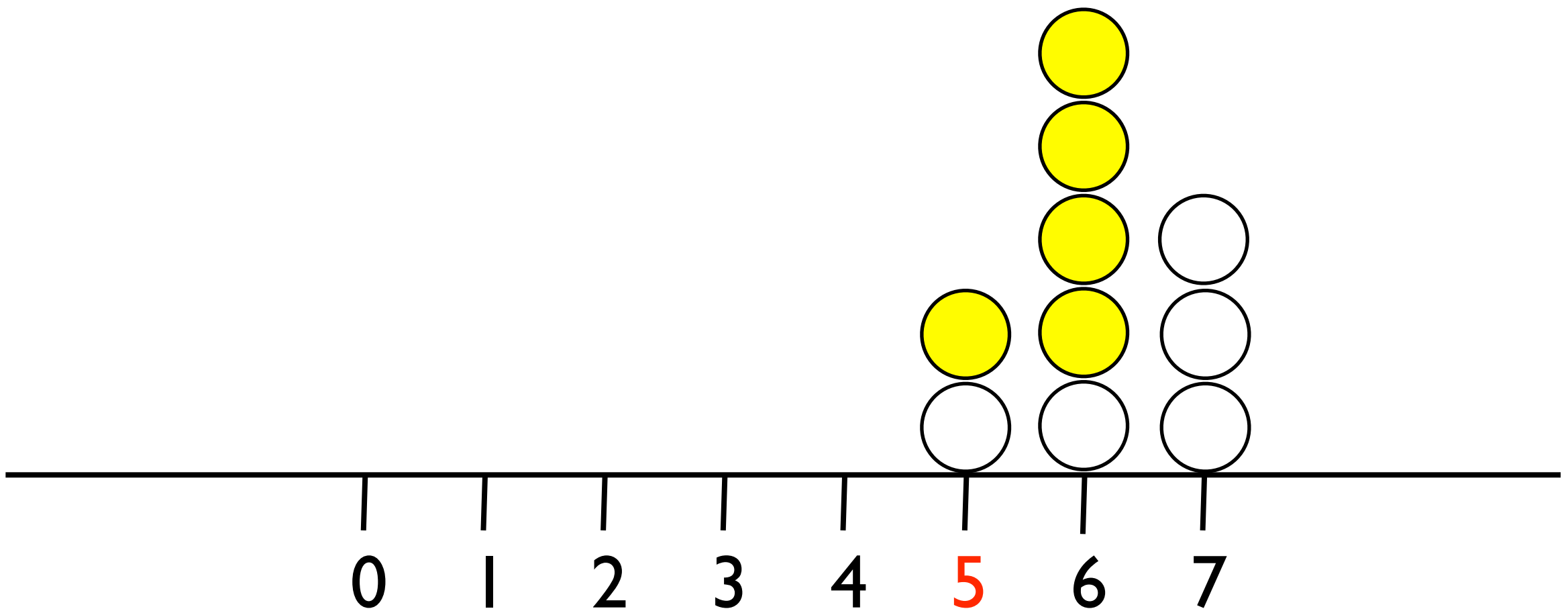
Configuration 2



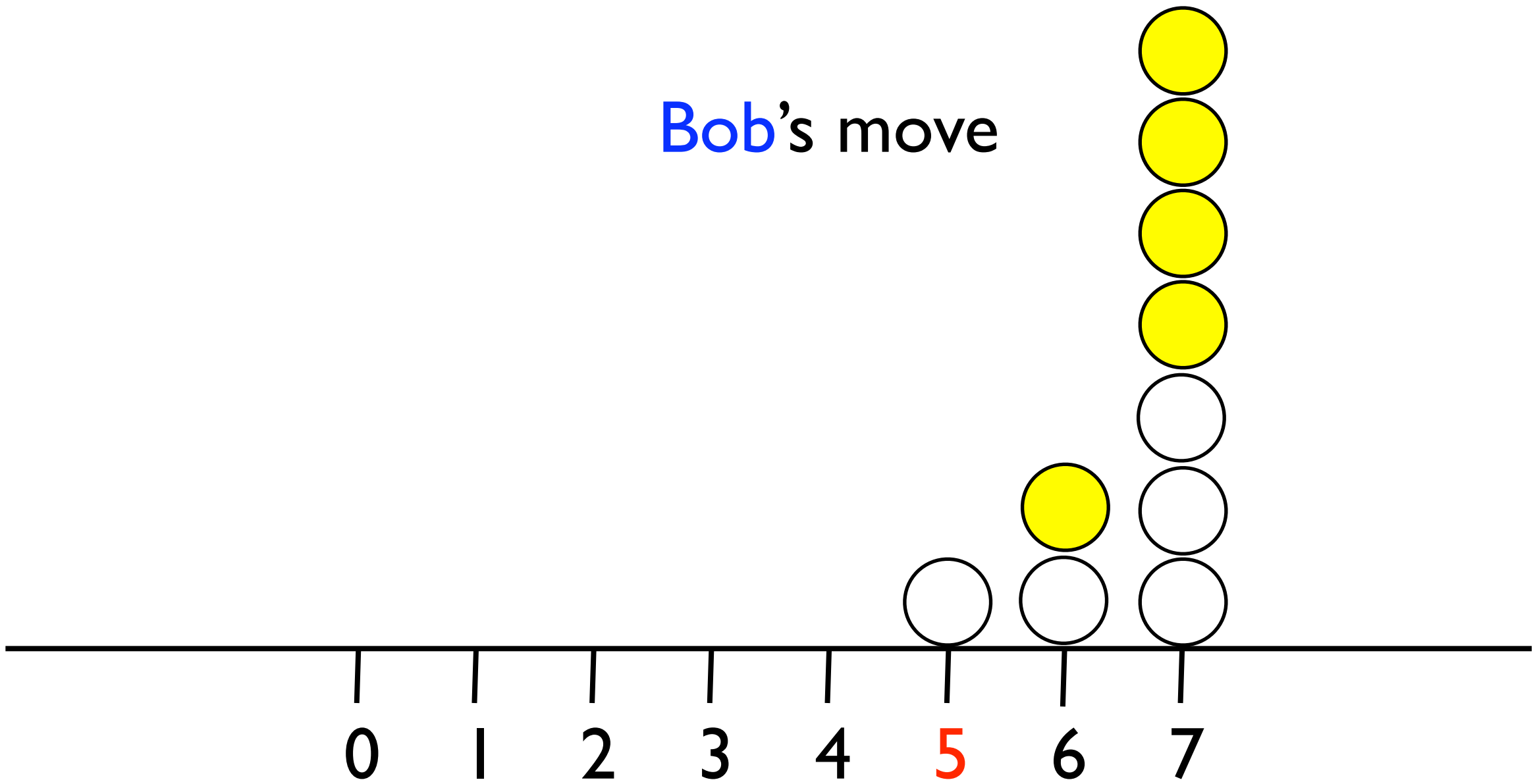
Configuration j



Alice's move



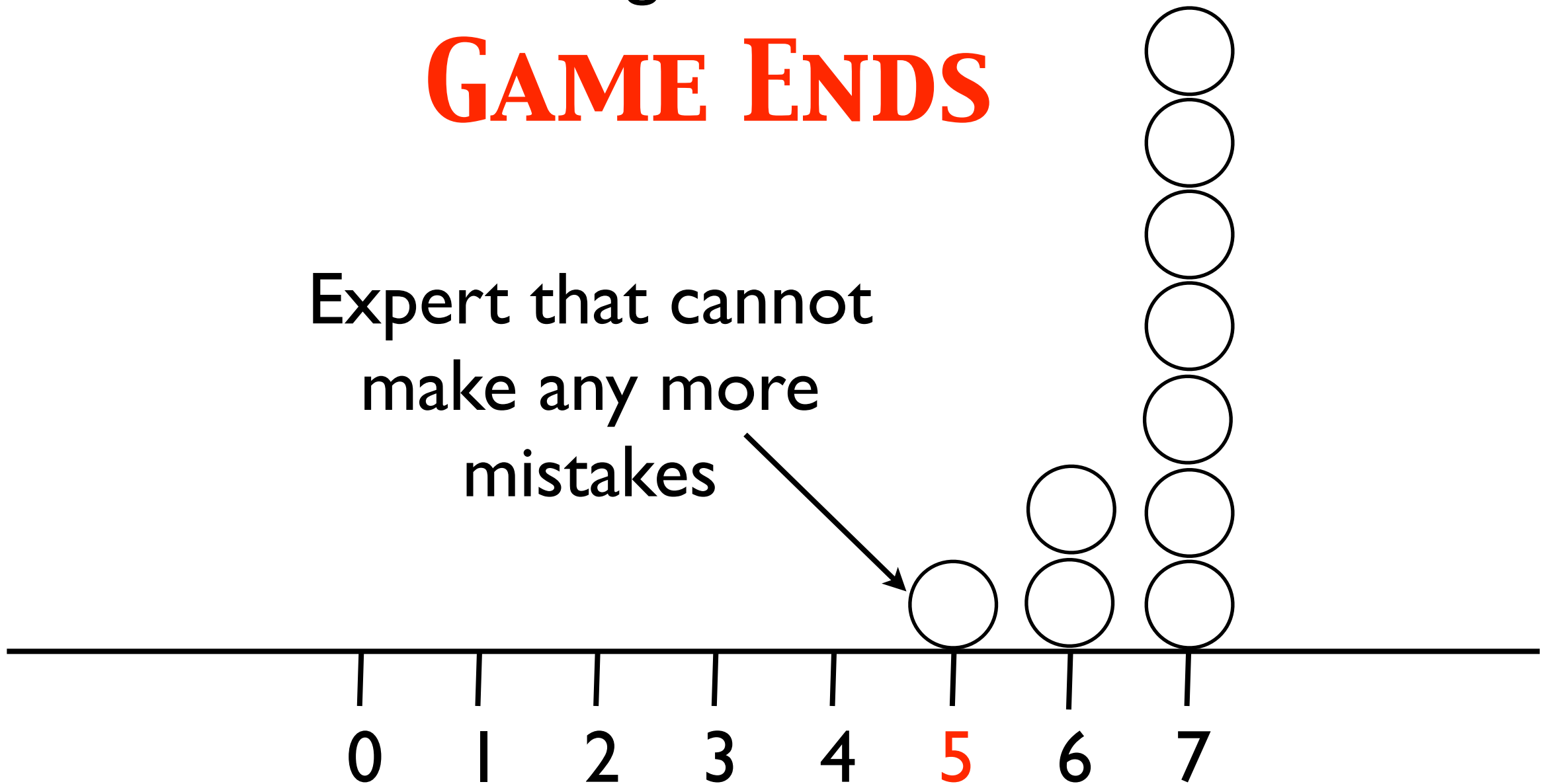
Bob's move



Final configuration

GAME ENDS

Expert that cannot
make any more
mistakes



The Binomial Weights algorithm

- Main difference from Ulam's game with k lies - Bob wants a short game, not a long game.
- Using potentials and weights works the same way, only the sign is reversed.
- Better than exponential weights.
- Limited to $+1/-1$ predictions.

Application to Boosting

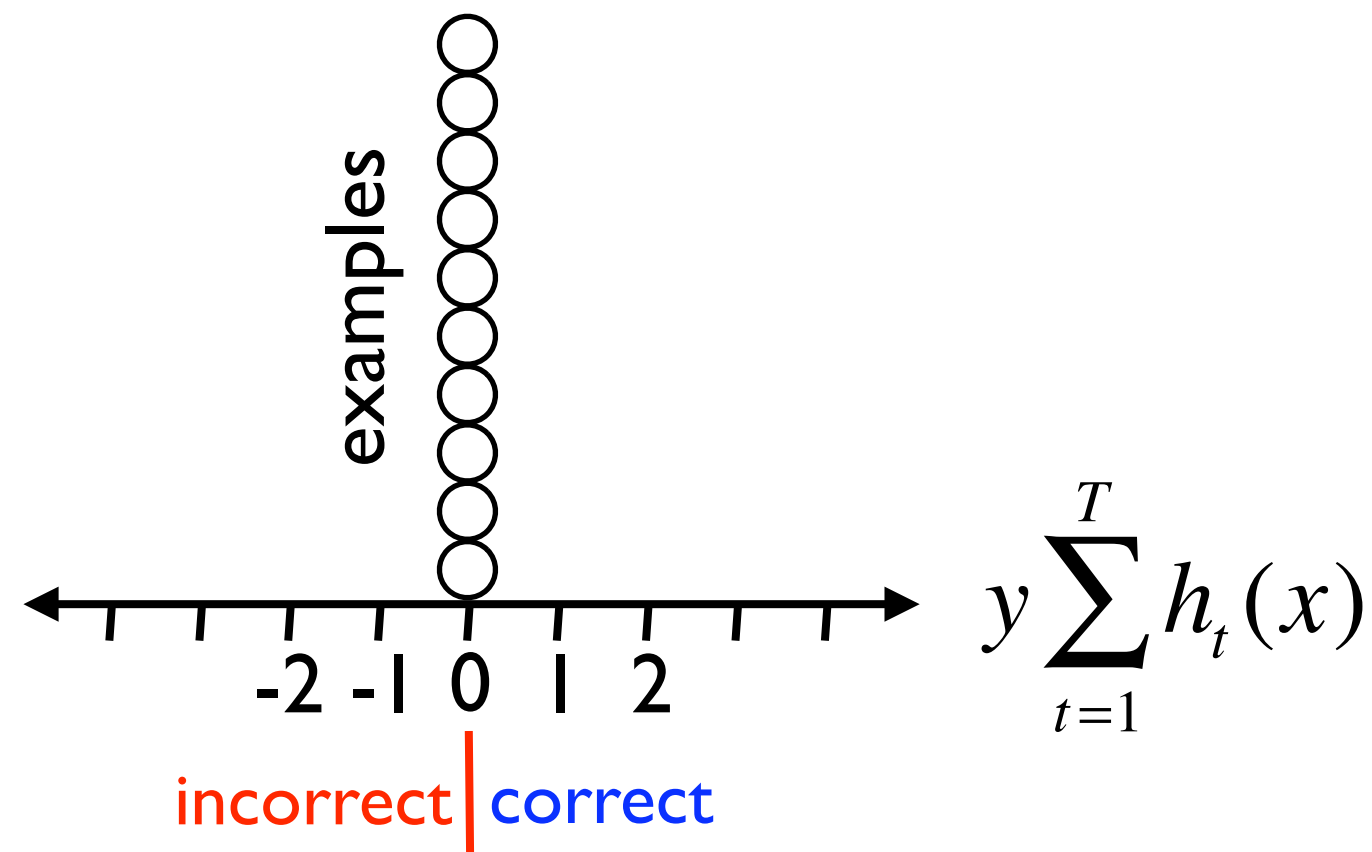
Boost by Majority

[Freund 95]

- game between a **booster** and a weak **learner**.
- Boosting generates a simple (unweighted) majority rule over weak learners.
- **T** Number of iterations is **set in advance**
- On iteration **$t=1..T$**
 - **booster** assigns weights to the training examples.
 - **learner** chooses a rule whose error wrt the chosen weights is smaller than **$1/2 - \gamma$**
 - Rule is added to majority rule
- Goal of booster is to minimize number of errors of final majority rule.

Boosting as a drifting game

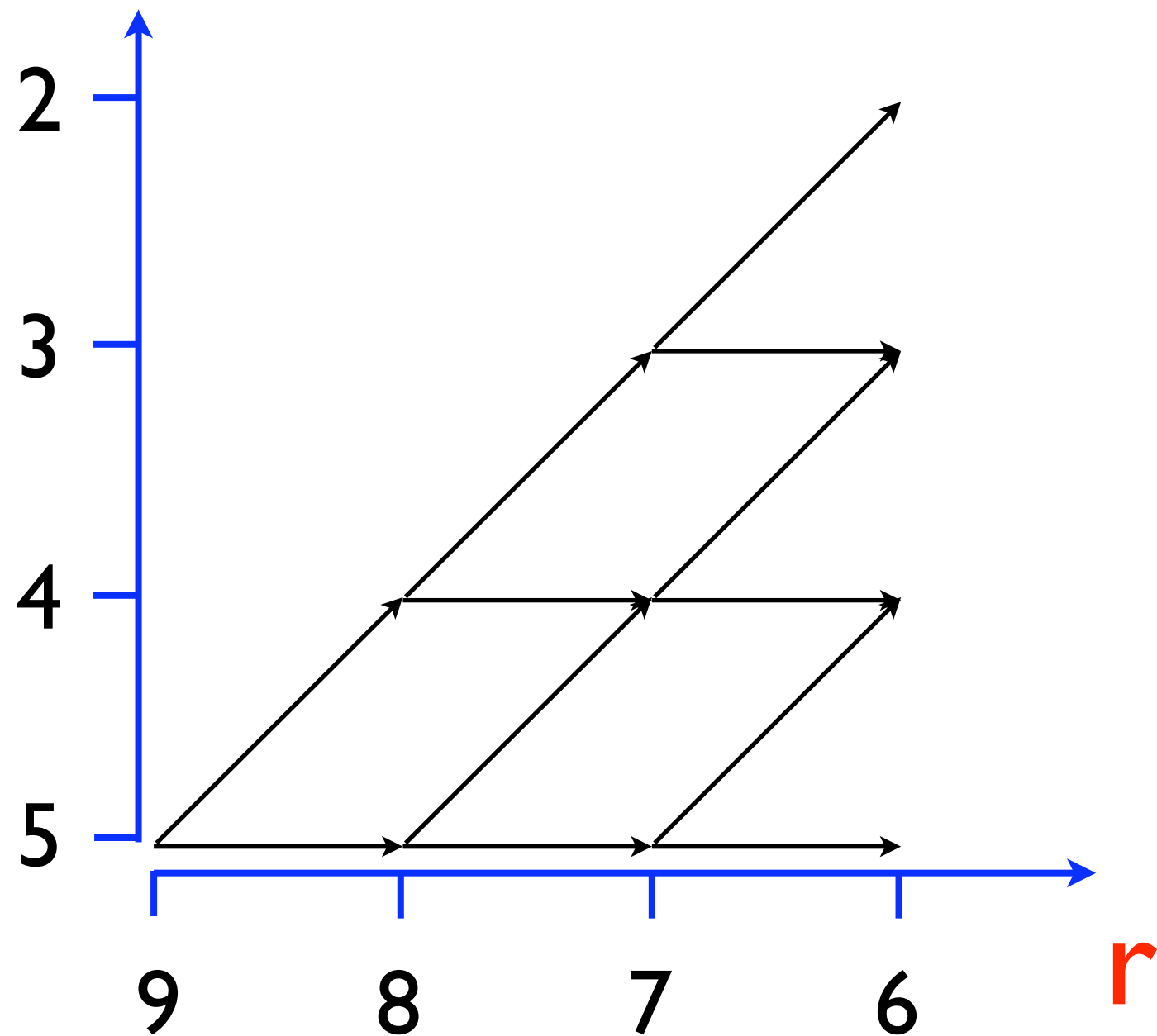
- Chips = examples
- bin i contains the examples for which the difference between the number of correct and incorrect rules is i



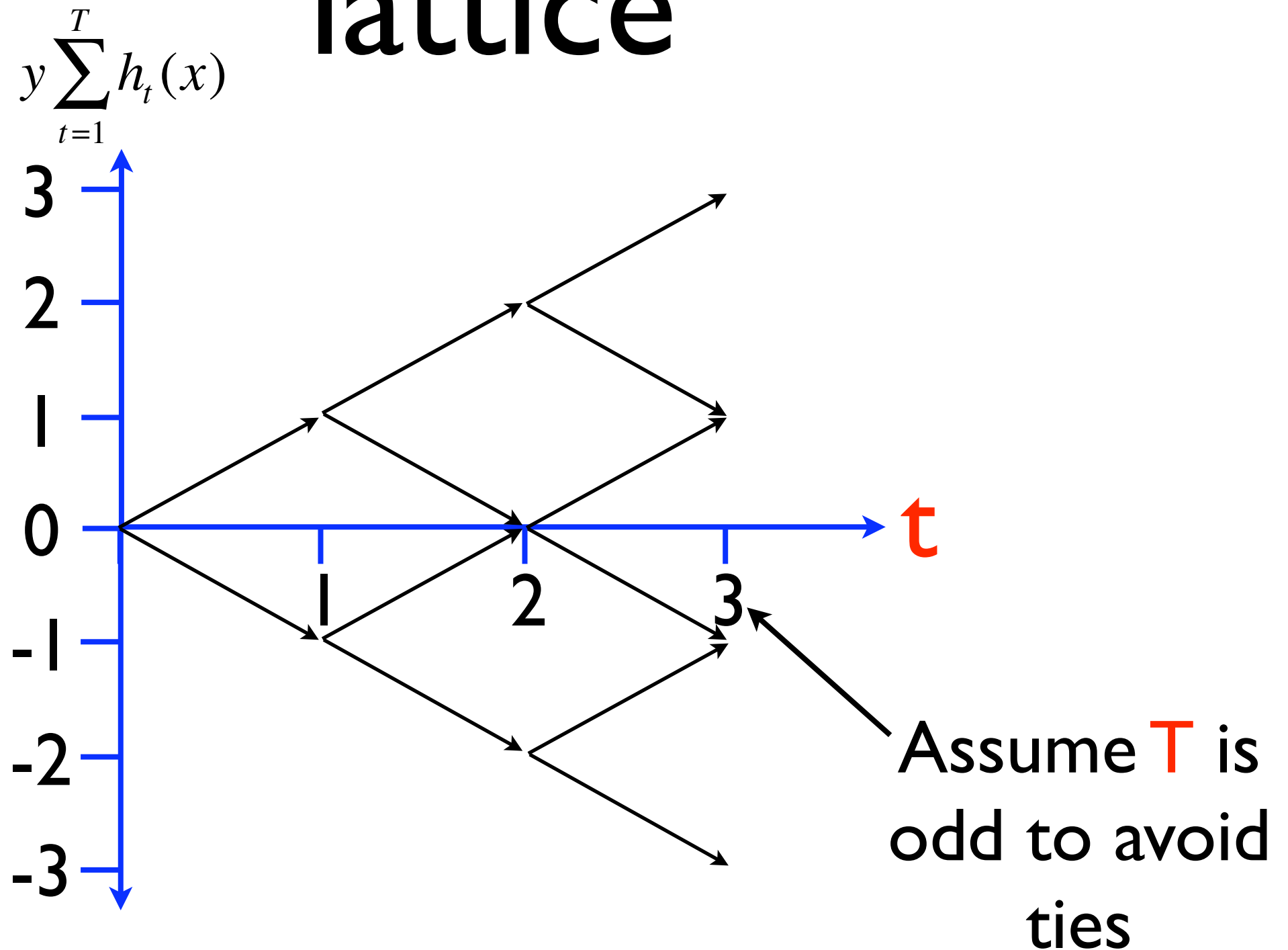
Initial Configuration

The Ulam game lattice

i



The boosting game lattice



$\gamma = 0.1$

Initial configuration

Text

-3

-2

-1

0

1

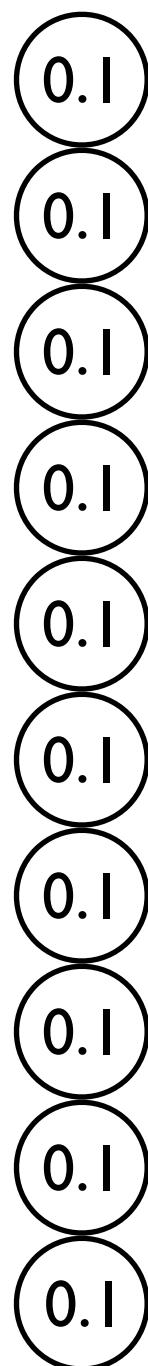
2

3

incorrect | correct

$\gamma = 0.1$

Booster's move



-3

-2

-1

0

1

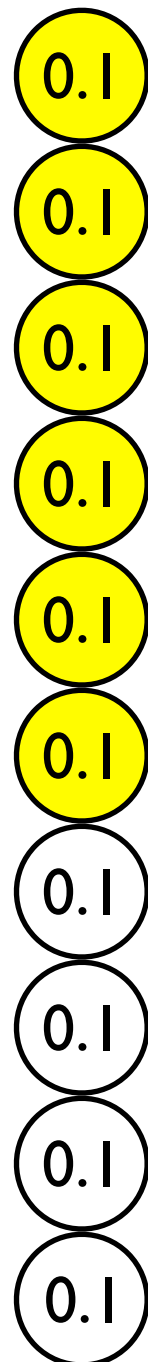
2

3

incorrect | correct

$\gamma = 0.1$

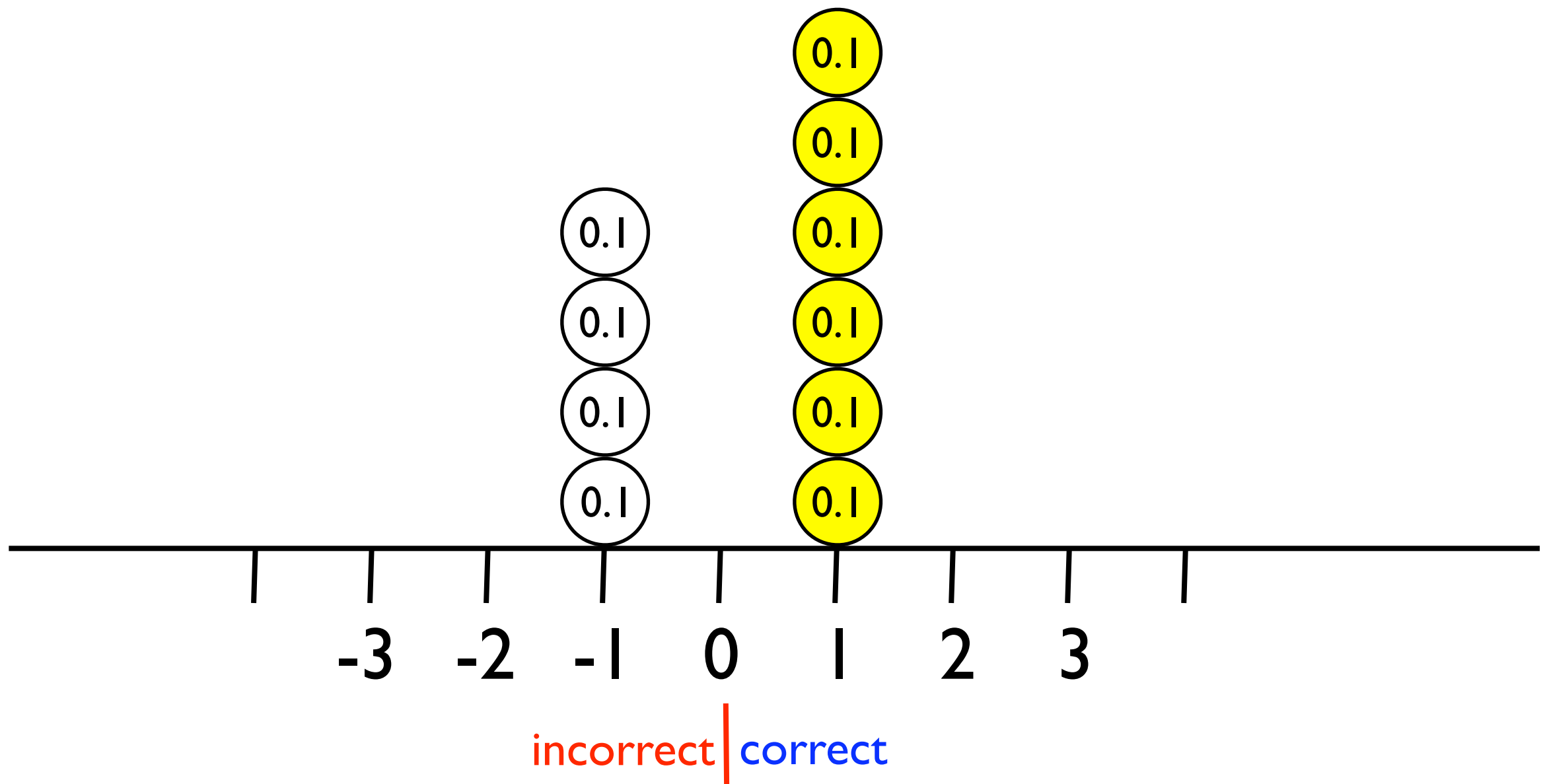
Learner's move



incorrect | correct

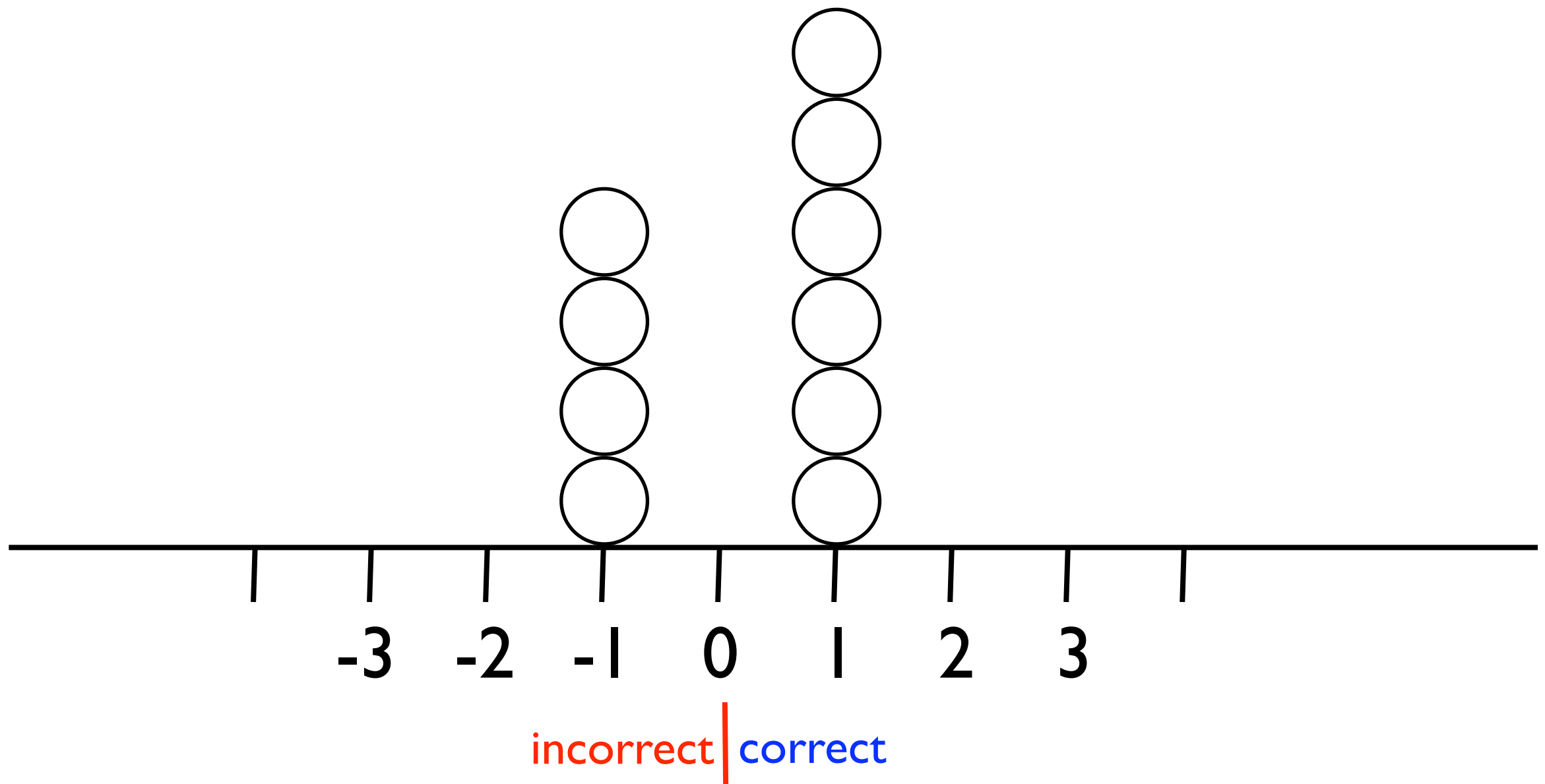
$\gamma = 0.1$

Update



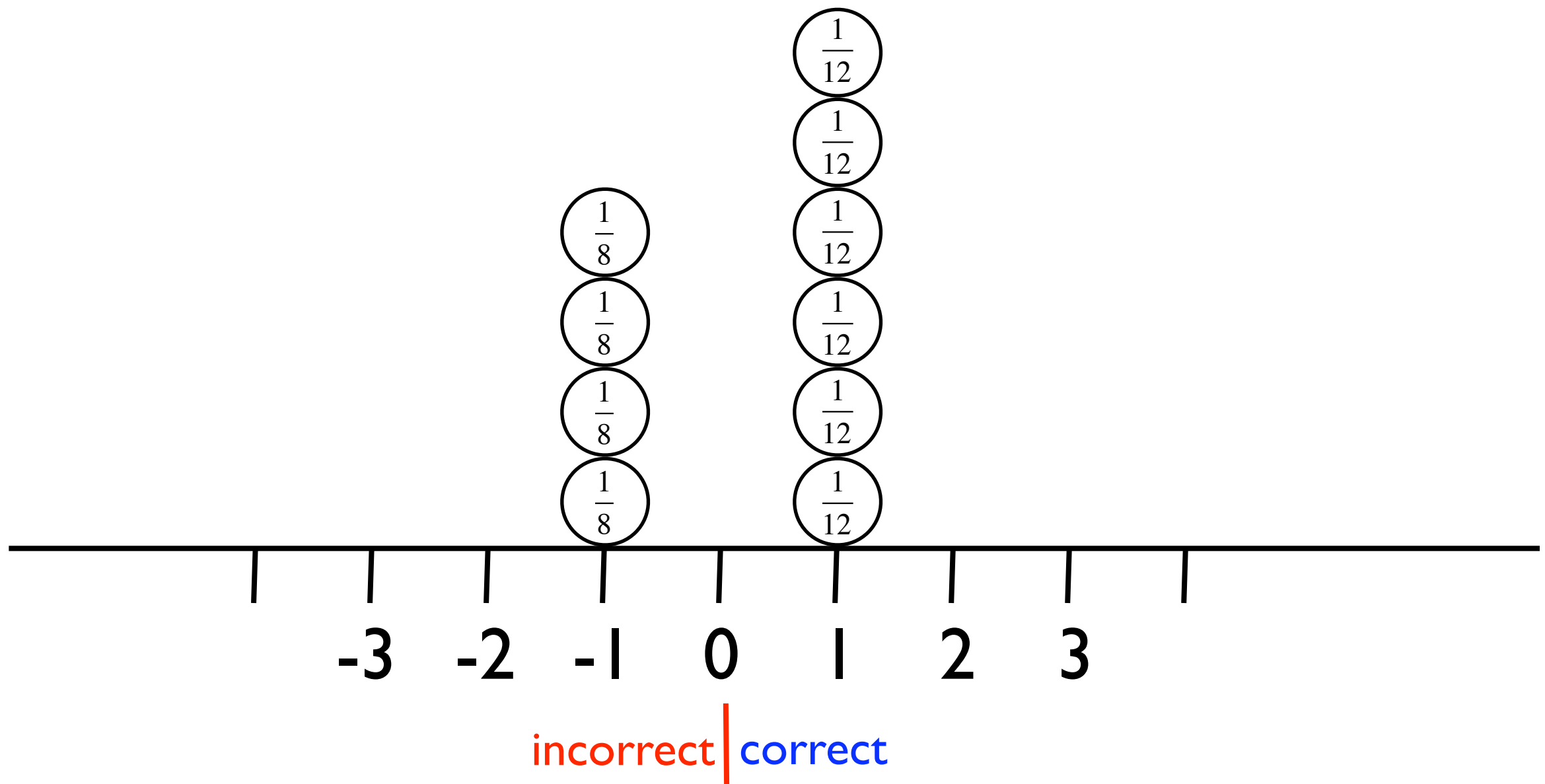
$\gamma = 0.1$

Configuration I



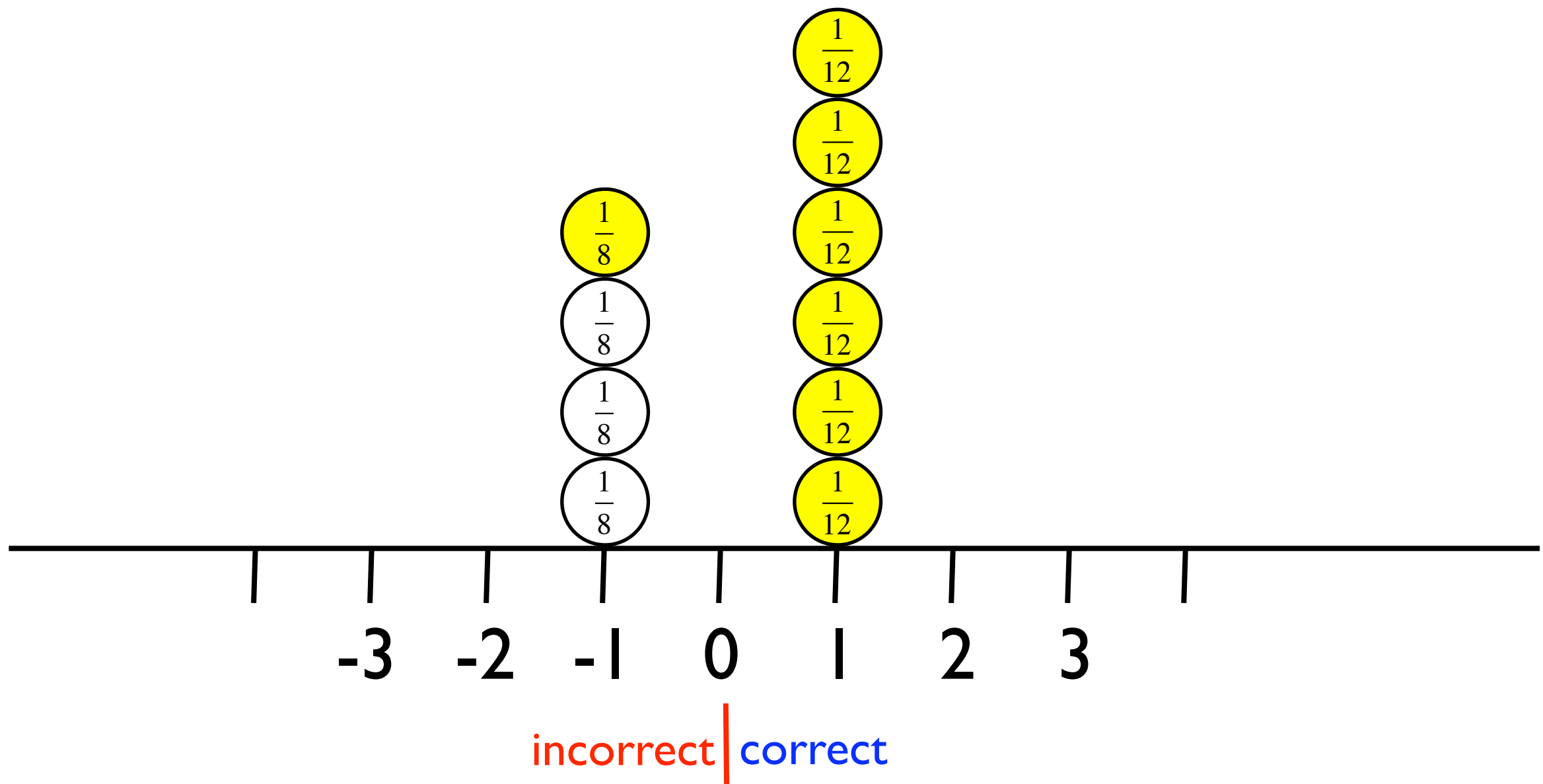
$\gamma = 0.1$

Booster's move



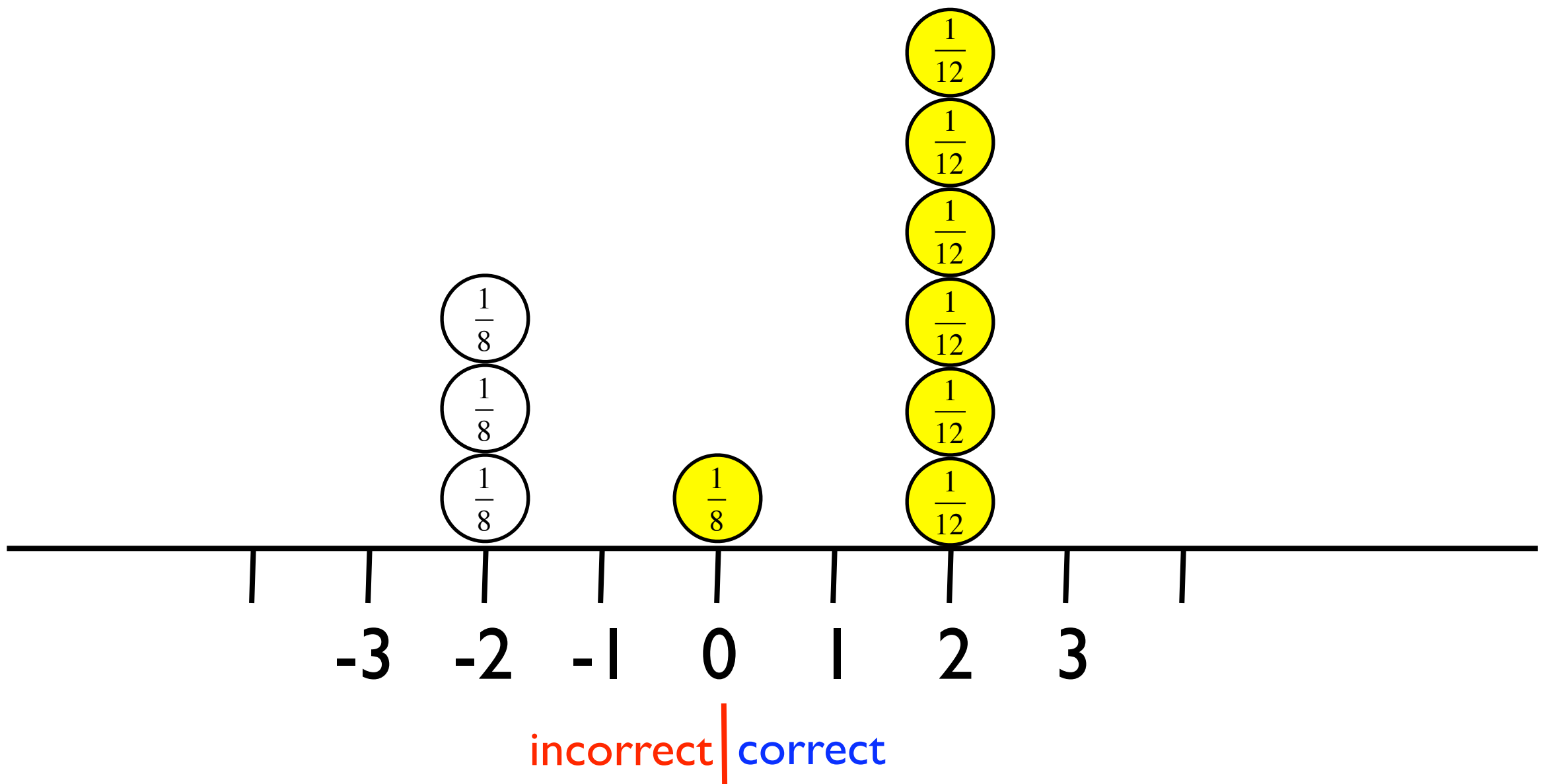
$\gamma = 0.1$

Learner's move



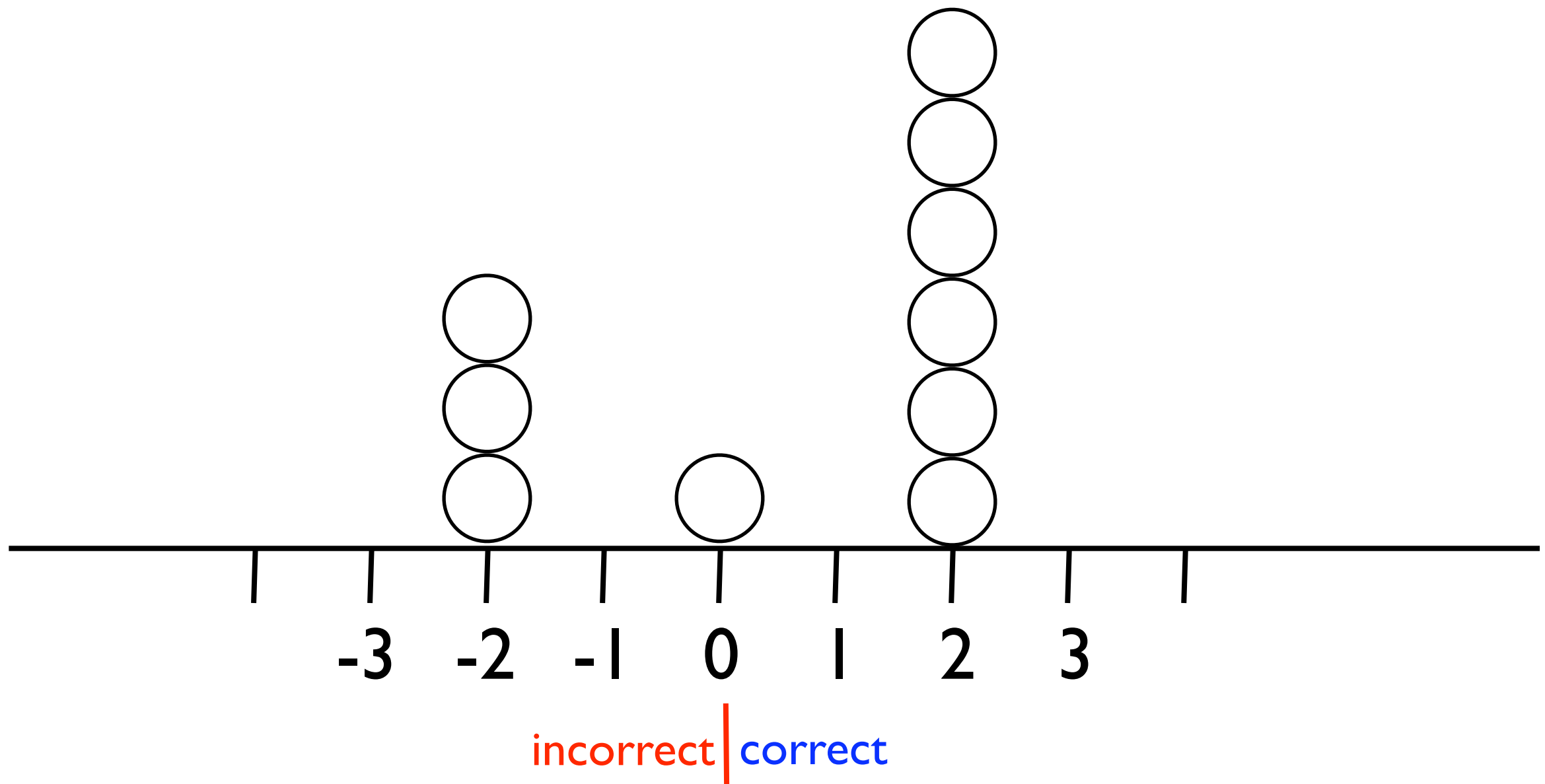
$\gamma = 0.1$

update



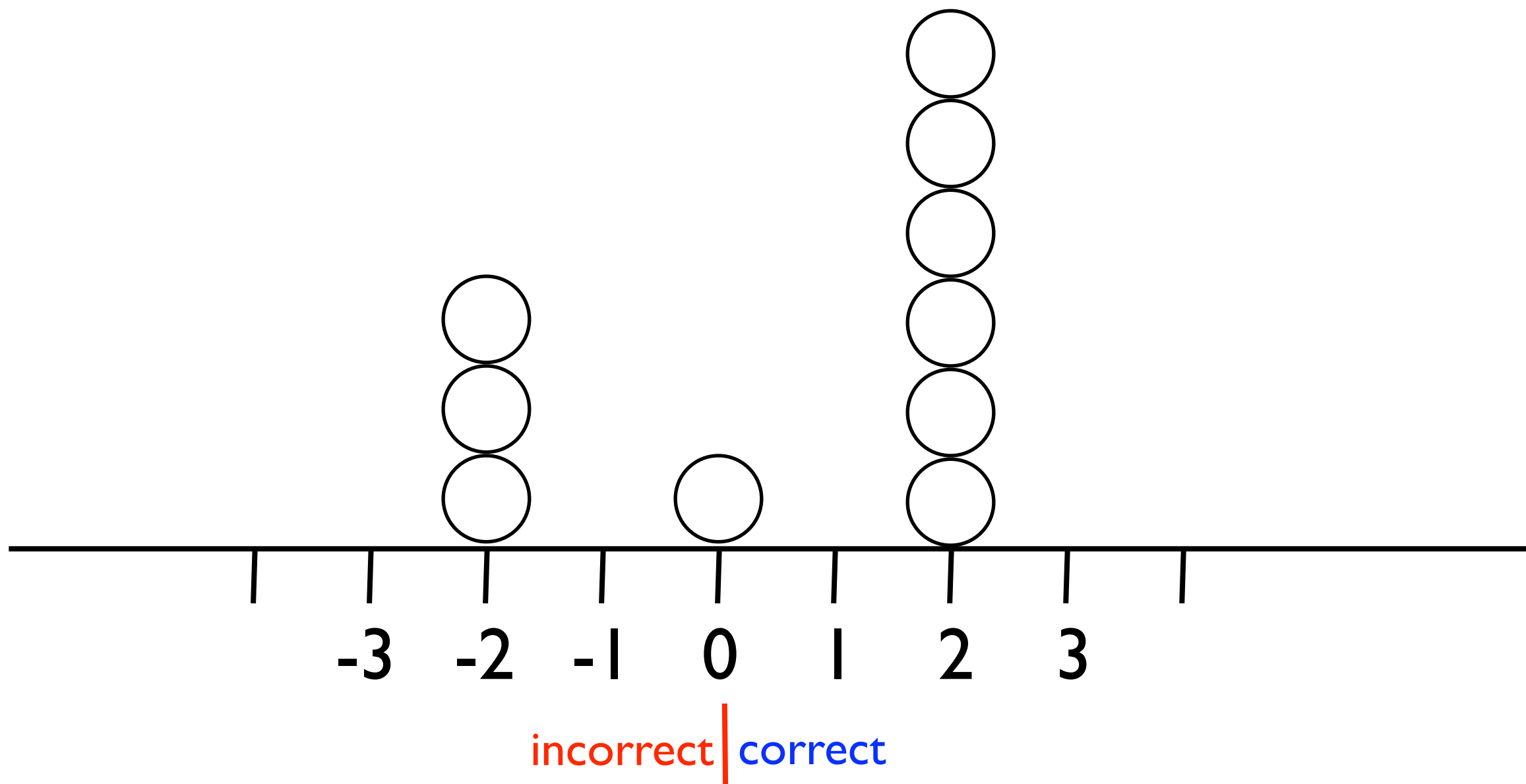
$\gamma = 0.1$

Configuration 2



$\gamma = 0.1$

Configuration T-I

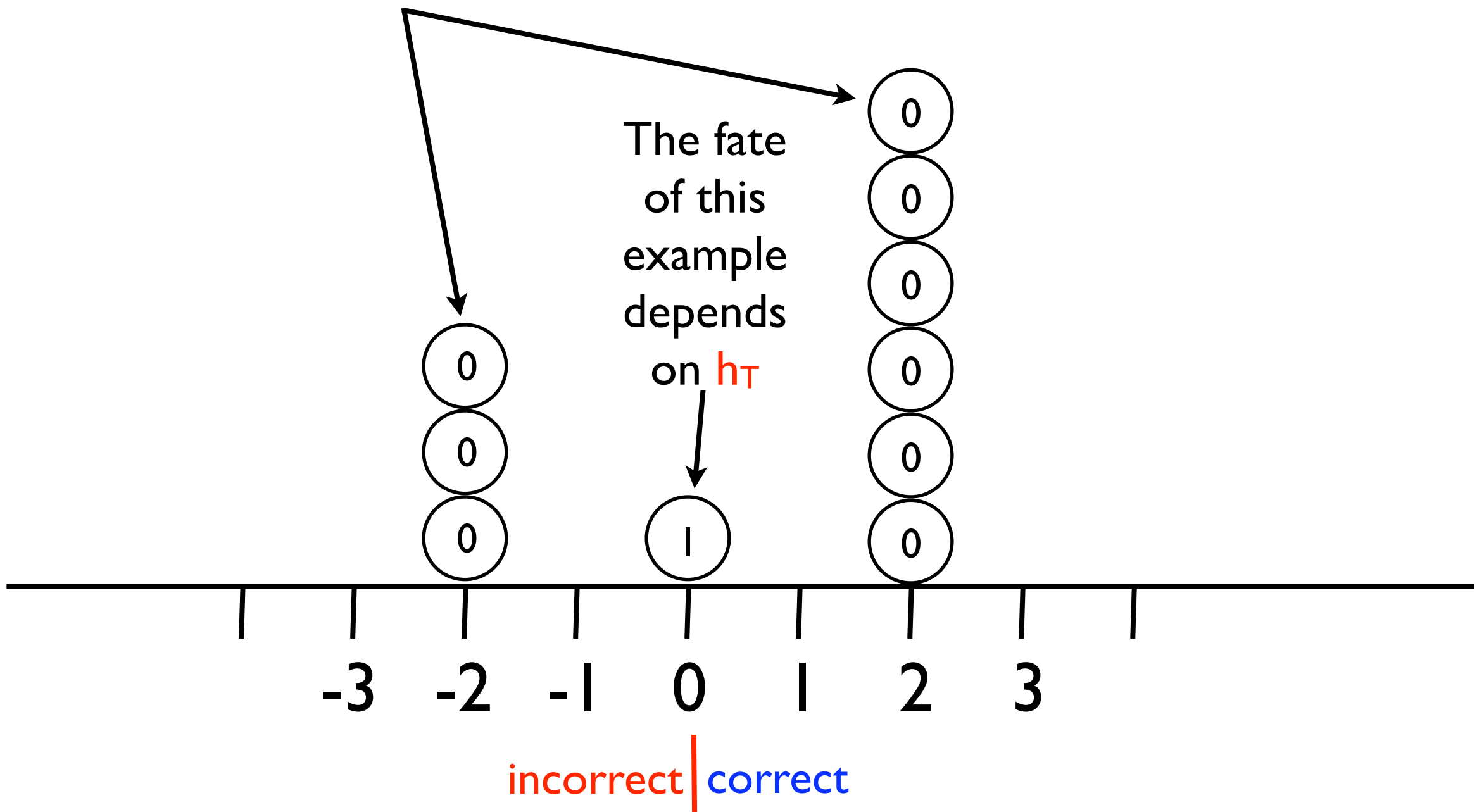


$\gamma = 0.1$

The fate
of these
example
is already
decided

Booster's move

The fate
of this
example
depends
on h_T

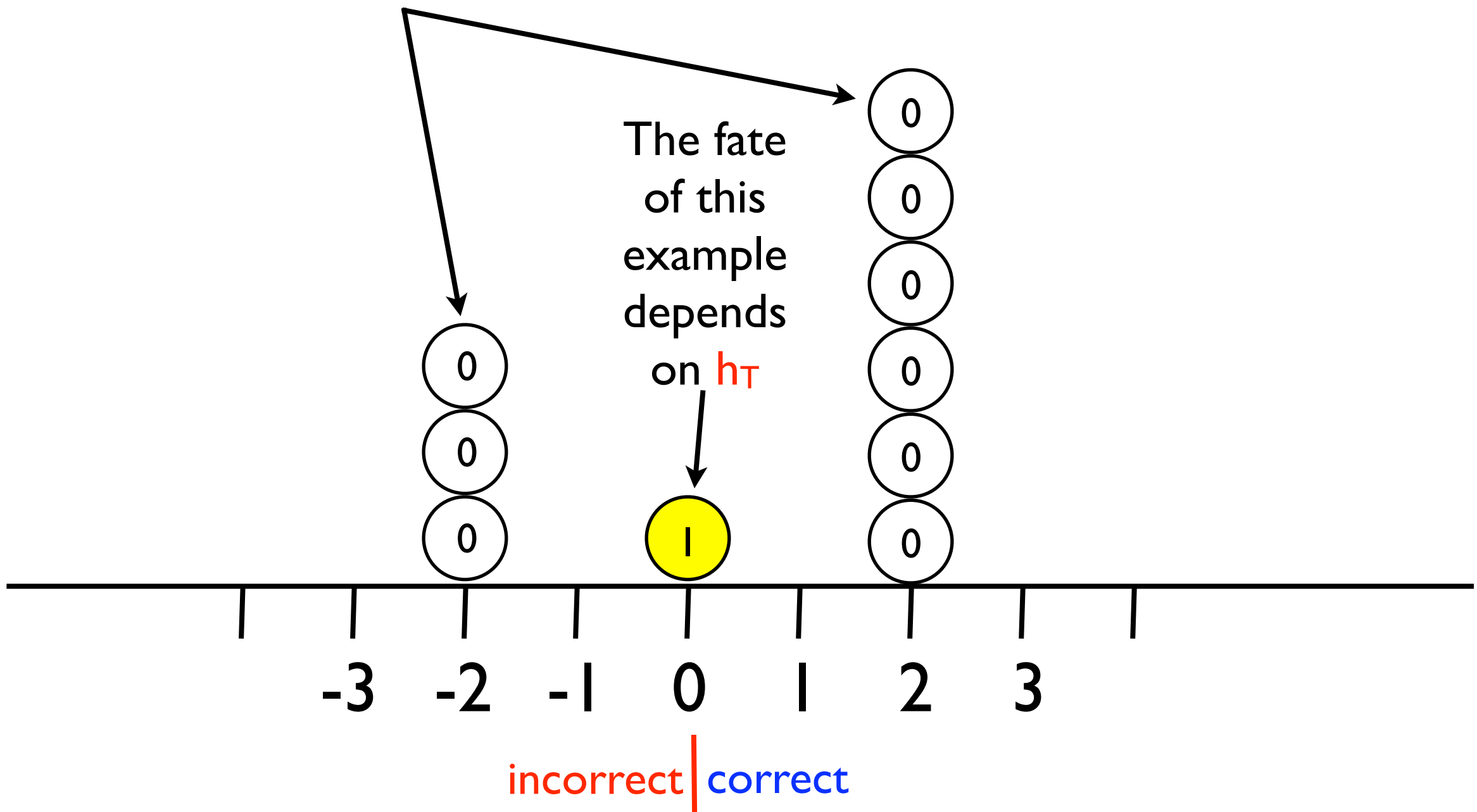


$\gamma = 0.1$

The fate
of these
example
is already
decided

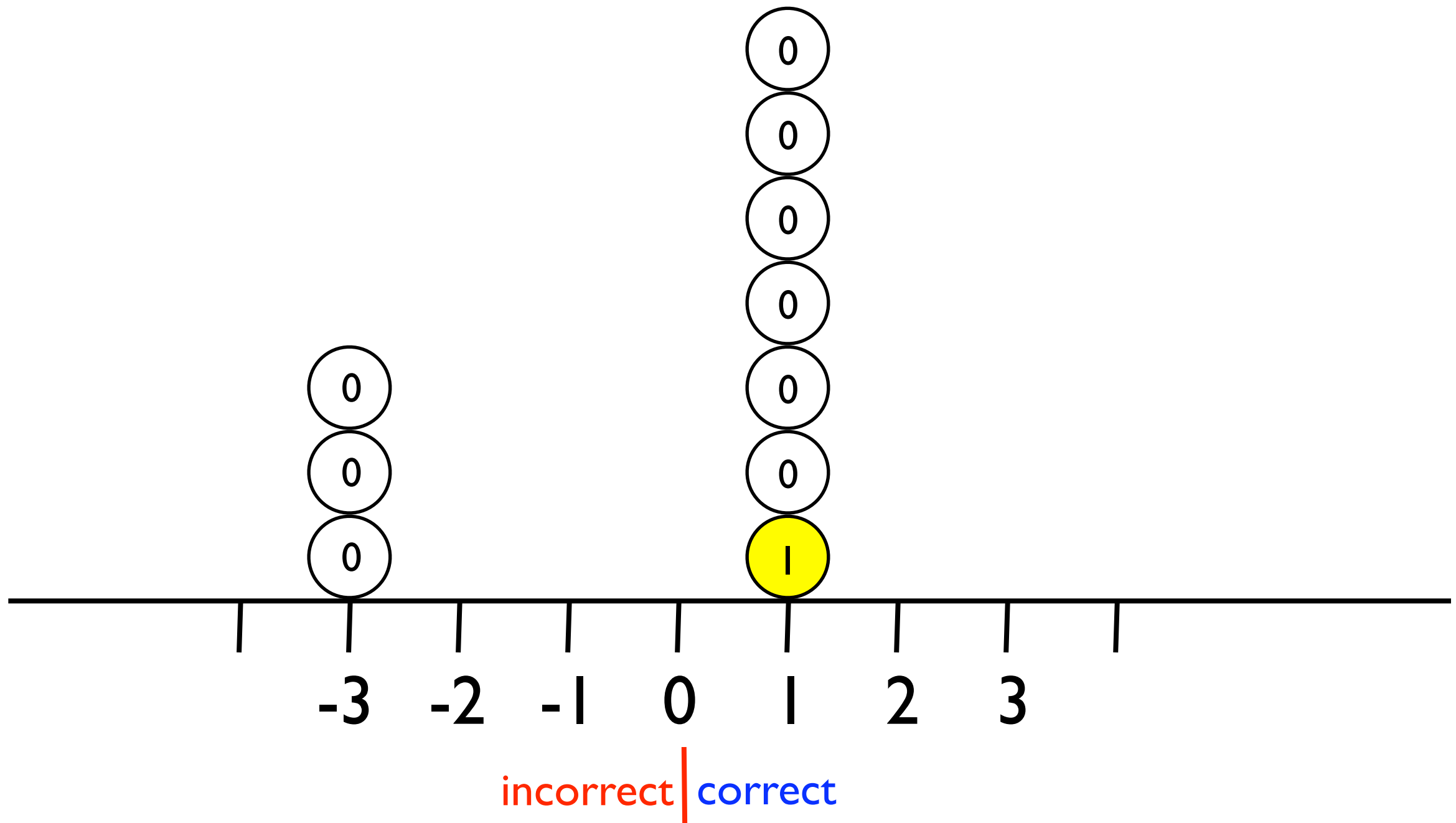
Learner's move

The fate
of this
example
depends
on h_T



$\gamma = 0.1$

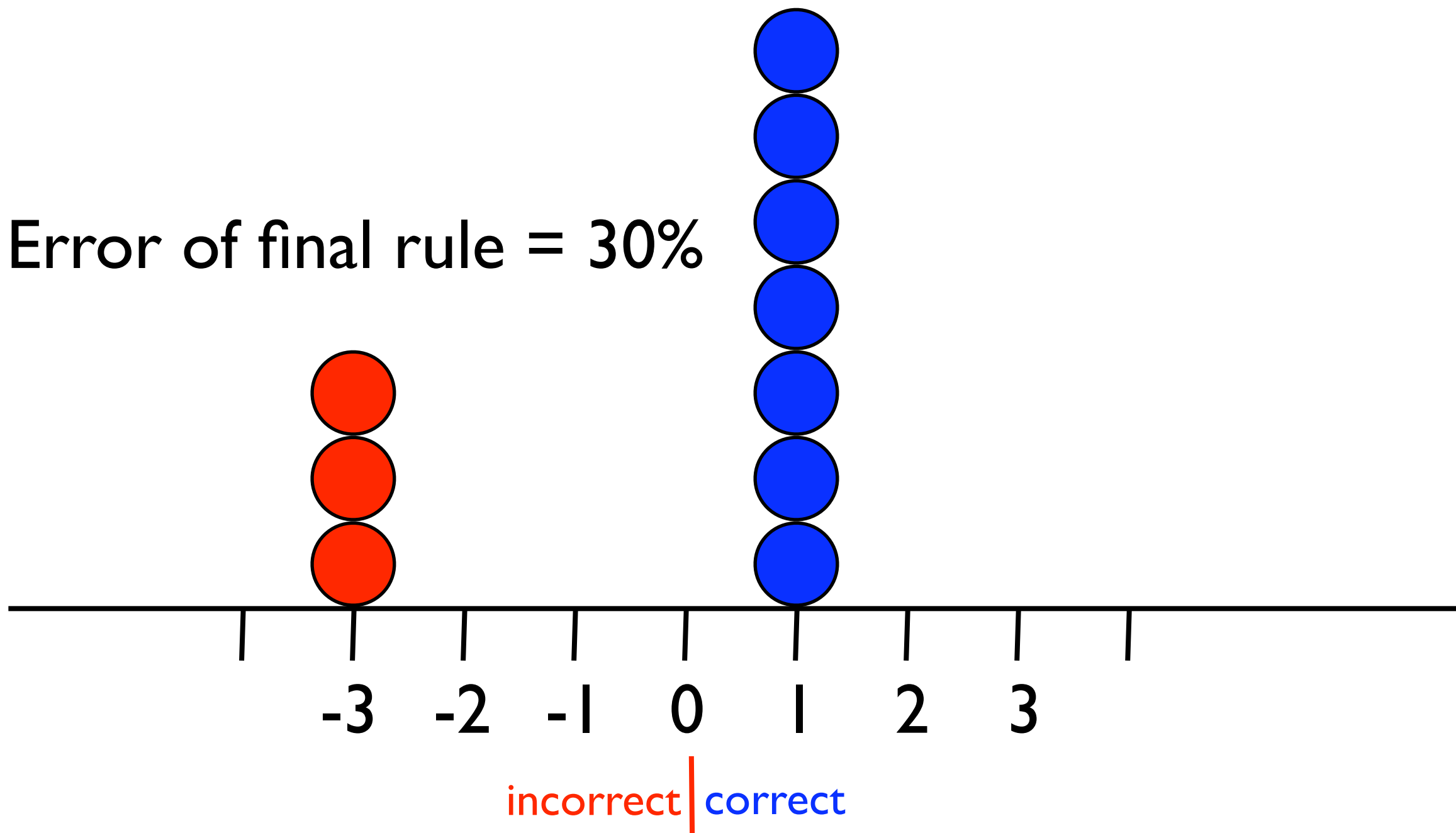
update



$\gamma = 0.1$

Final Configuration

Error of final rule = 30%



Learner's min/max strategy

- Choose $1/2 + \gamma$ from each bin to be correct.
- Might not be possible if number of examples is finite.
- If set of examples is continuous - strategy is min/max optimal.
- Equivalent to producing the correct answer independently at random with
$$P(\text{correct}) = 1/2 + \gamma$$

Potential and weight for boosting

$$w(t,s) = \left(\frac{T-t}{\left\lfloor \frac{T-t+s+1}{2} \right\rfloor} \right) \left(\frac{1}{2} + \gamma \right)$$

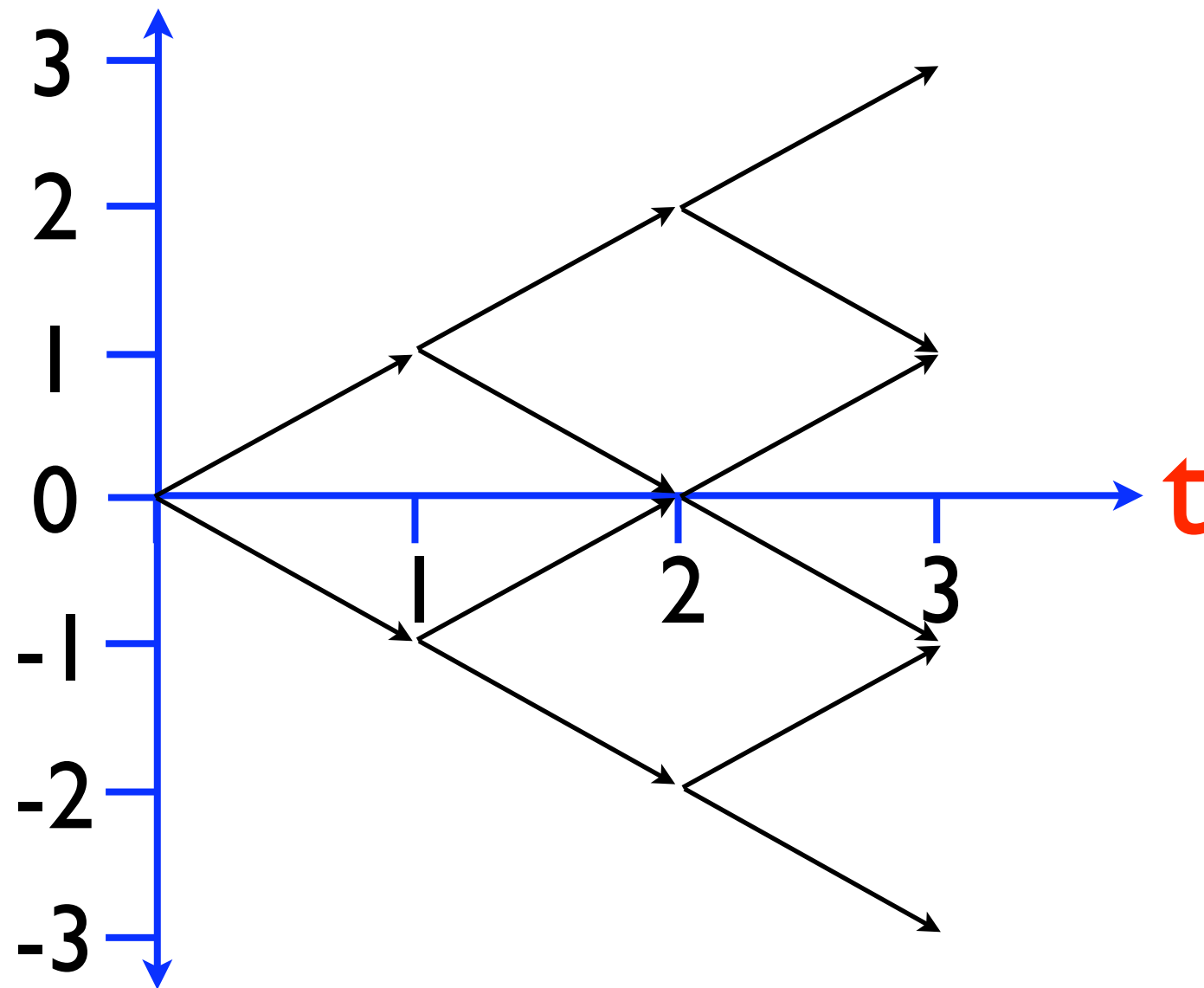
From discrete to continuous time

- BBM is not usable because it is not adaptive.
- BW is not usable because the predictions are restricted to be either 0 or 1.

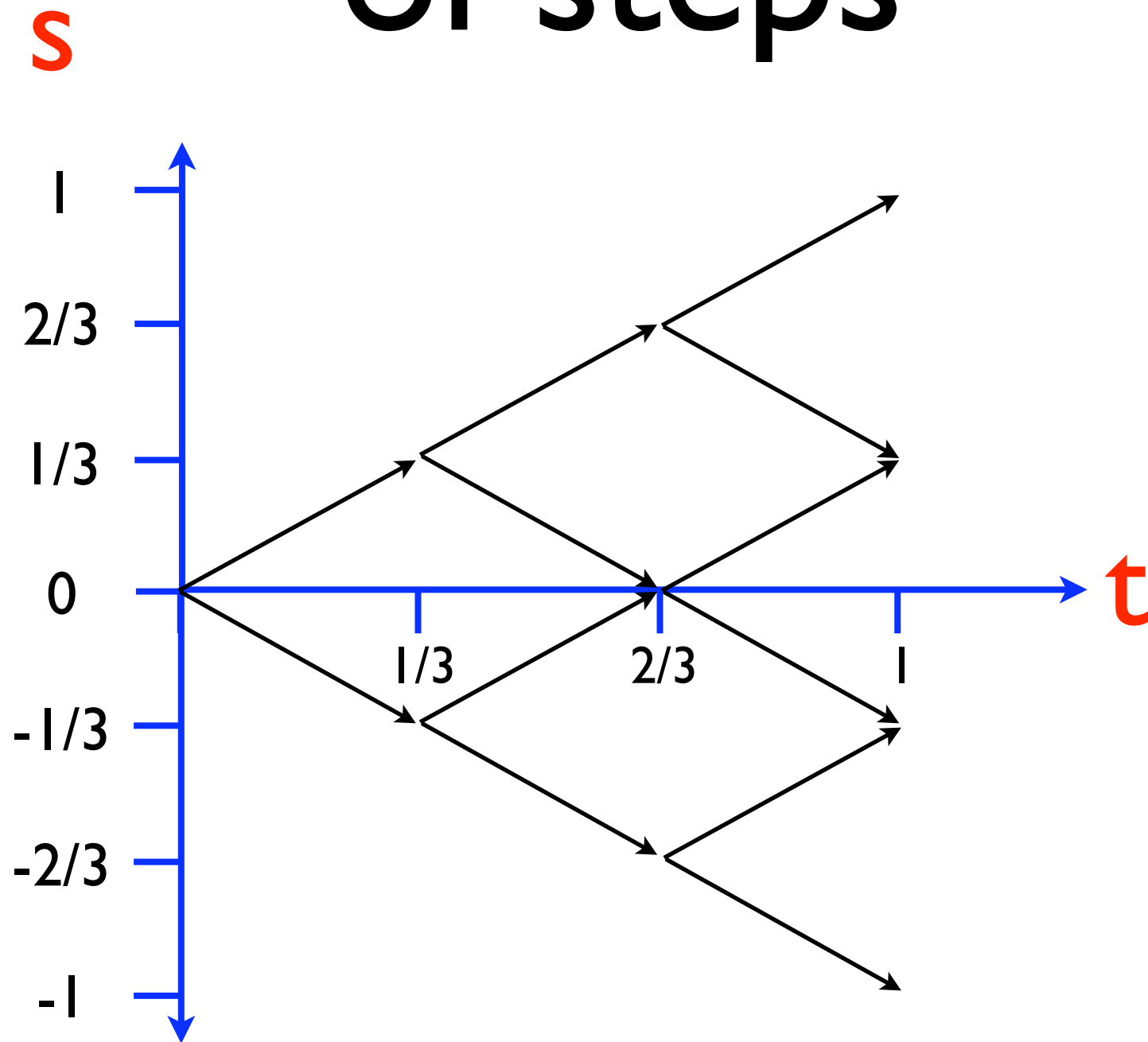
Letting time step
decrease to zero.

The game lattice

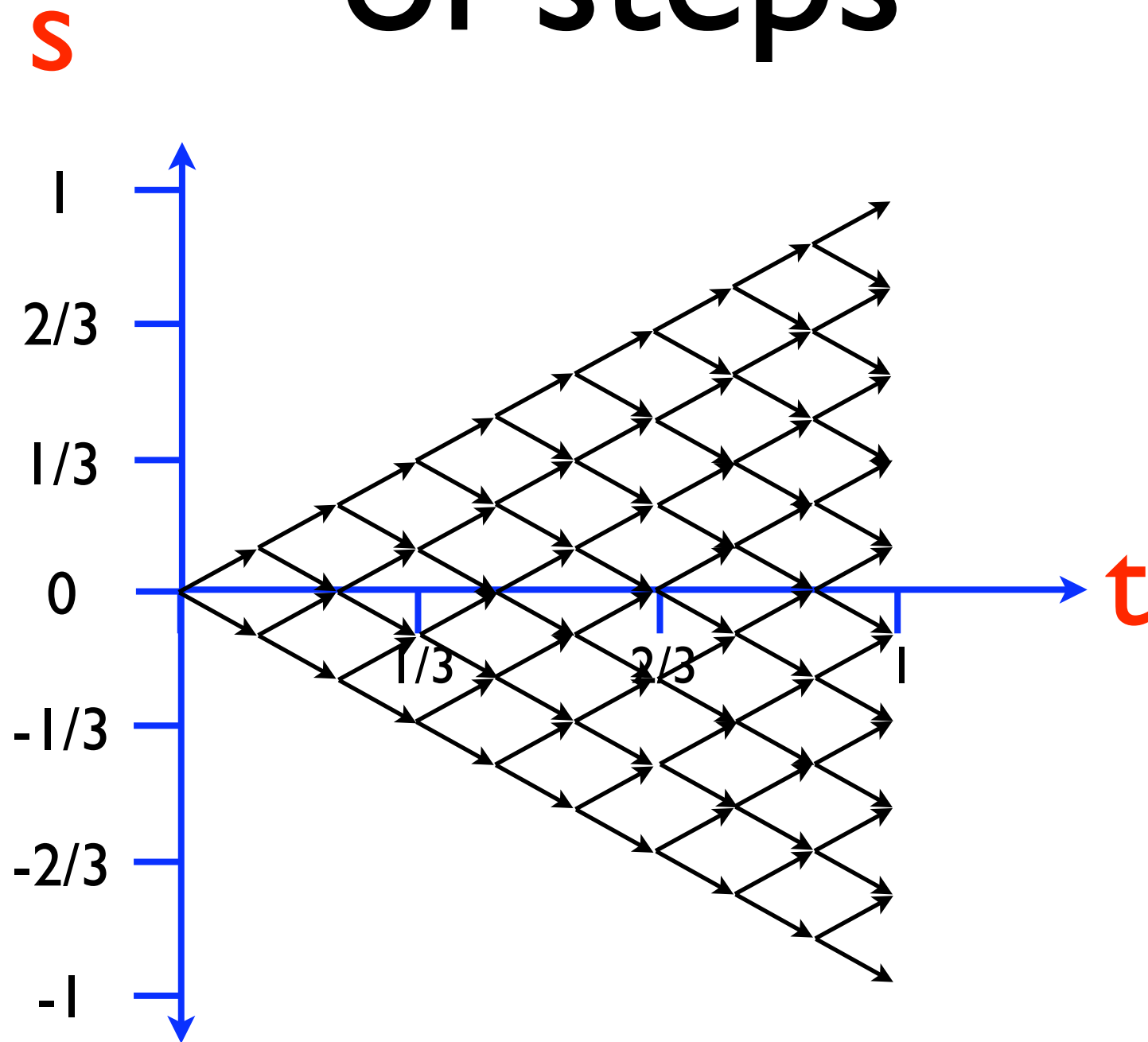
s



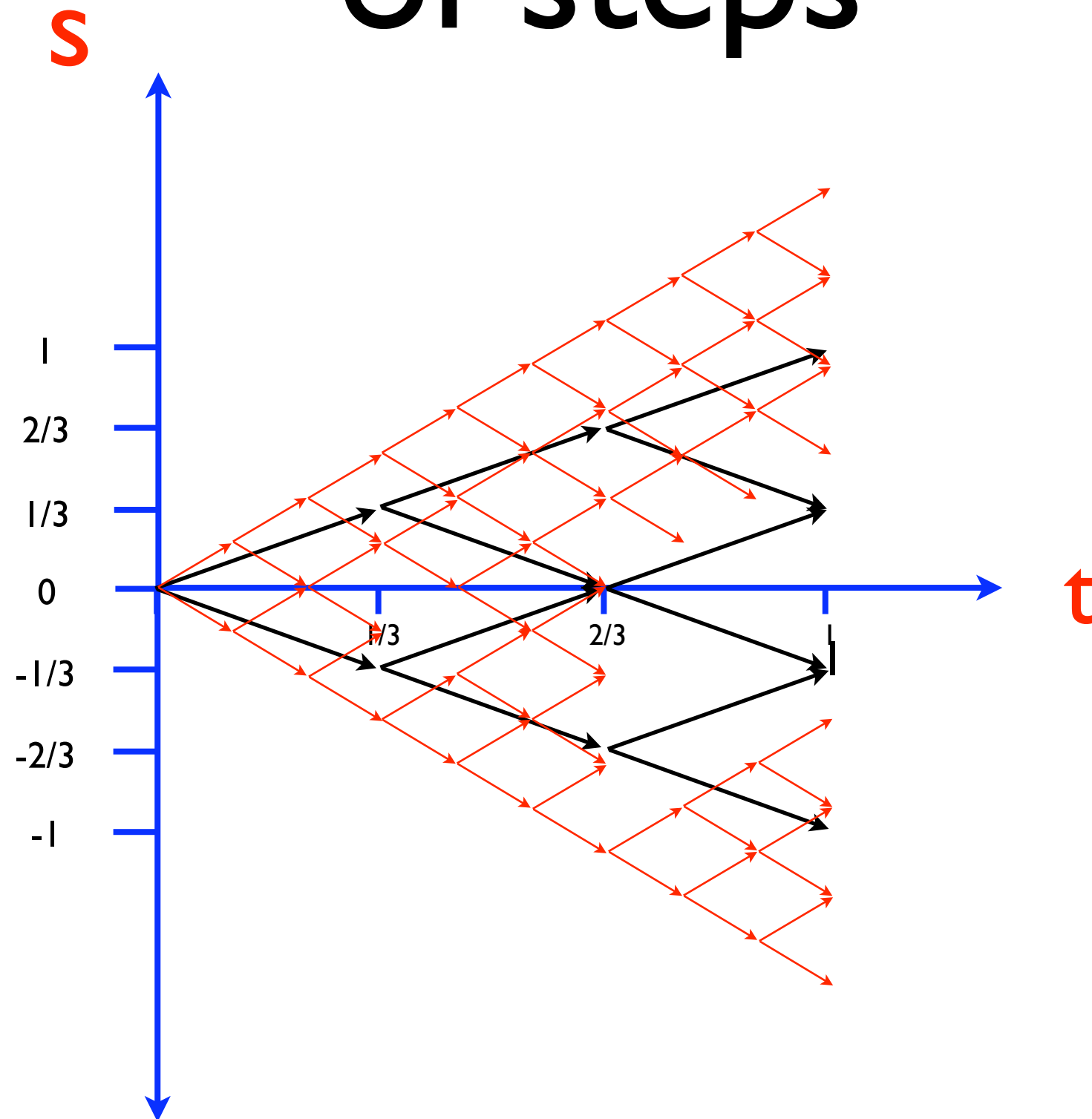
Increasing the number of steps



Increasing the number of steps



Increasing the number of steps



Boosting

- Boost by majority
- Brownboost
- Robustboost
- Experiments

Online Learning

- Multiplicative weights
- Binomial Weights
- Normalhedge
- Tracking faces