Drifting Games

Combining expert advice

- Binary sequence: 1,0,0,1,1,0,?
- N experts make predictions:

  expert 1: 1,0,1,0,1,1,1

  expert 2: 0,0,0,1,1,0,1

  …

  expert N: 1,0,0,1,1,1,0
- Algorithm makes prediction: 1,0,1,1,0,1,1
- Assumption: there exists an expert which makes at most k mistakes
- Goal: make least mistakes under the assumption (no statistics!)

Analysis using meta-experts

- Consider only the iterations on which the algorithm makes a mistake.
- Expand each original (base) expert into meta-expert= base-expert+ (sequence of locations of mistakes).
- Suppose a sequence of length m=10 and one of the base experts makes at most k=2 mistakes.
- Mistake no.: 1,2,3,4,5,6,7,8,9,10
  - No mistake: 0,0,0,0,0,0,0,0,0,0
  - 1 mistake: 1,2,3,…,10
  - 2 mistaks: (1,2),(1,3),(1,4),…, (9,10)
- If the number of mistakes is at most 2, then there is a meta-expert that makes no mistakes.
- We can use the halving algorithm on the set of meta experts.

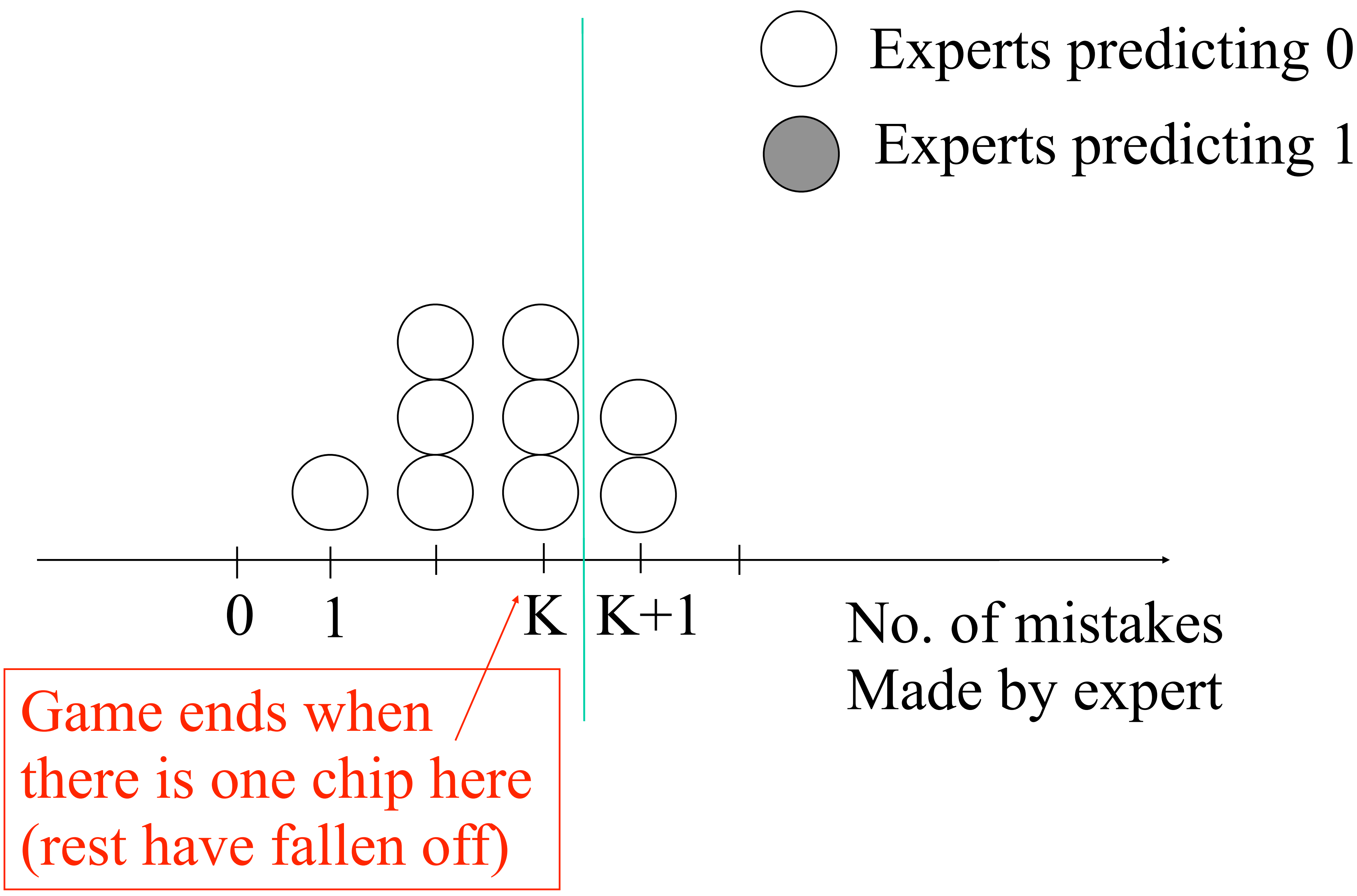
Number of mistakes is smallest m such that $\frac{1}{2^{m+1}}\binom{m+1}{\leq k-i} = \frac{1}{2^{m+1}}\sum_{j=0}^{k-i}\binom{m+1}{j} < 1$

- Naive implementation requires explicitly maintaining each meta-expert.

Reduction to chip game

- Chip = (base)-expert, bin = number of mistakes made so far
- Game step = algorithm’s prediction is incorrect.

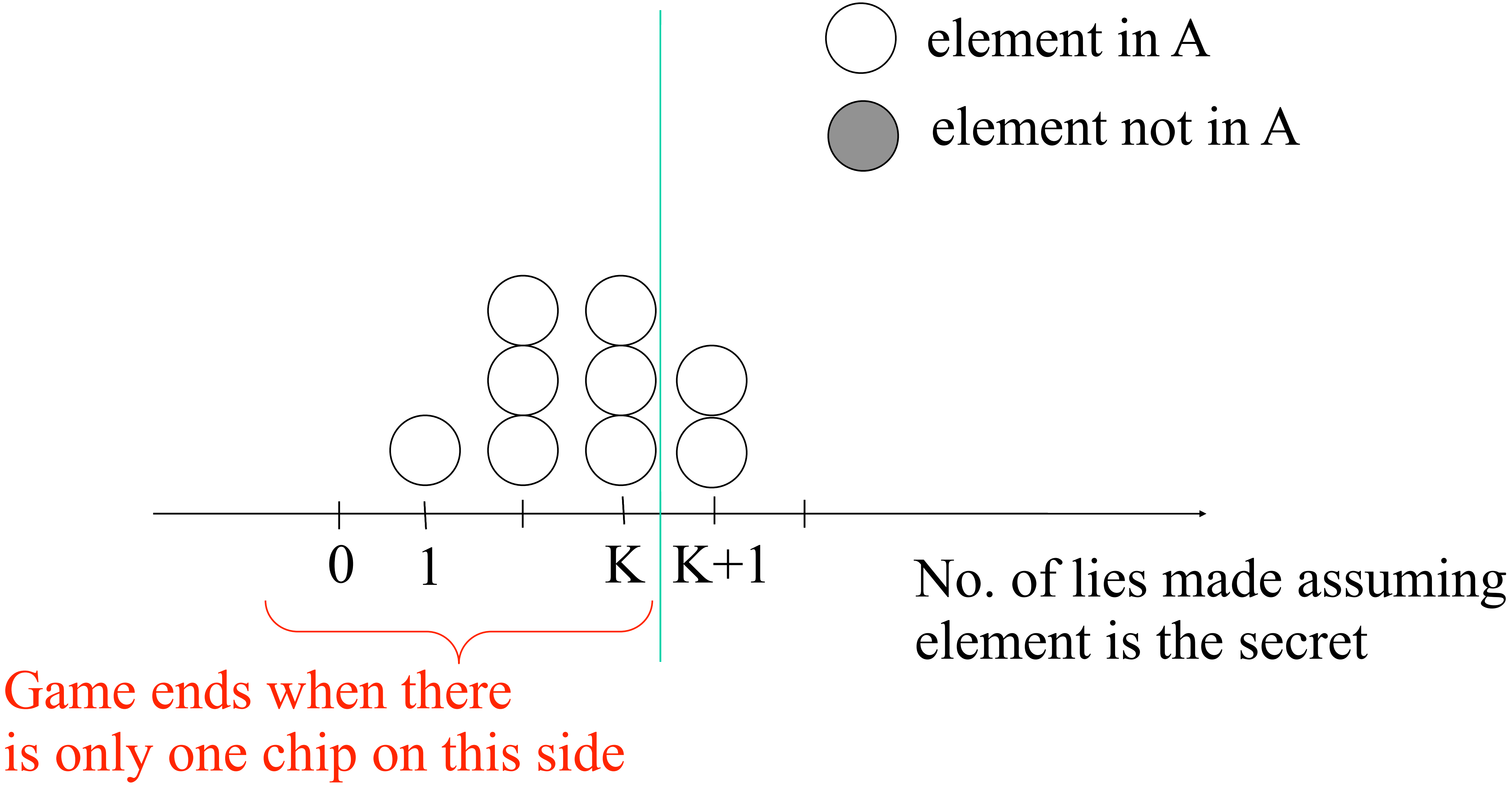
Chip game for expert advice



20 questions with k lies [Ulam's game with lies]

- Player 1 chooses secret x from 1,…,N
- Player 2 asks "is x in the set A?"
- Player 1 answers yes/no, can lie at most k times.
- Game ends when player 2 can determine x.
- A smart player 1 aims to have more than one consistent x for as long as possible.
- Chip = element i
- Bin = number of lies made regarding element

Chip game for 20 questions



Simple case – all chips in bin 1

- 21 questions without lies
- Combining experts where one is perfect
- Splitting a cookie
- Optimal strategies:
  - Player 1: split chips into two equal parts
  - Player 2: choose larger part
- Note problem when number of chips is odd

Binomial weights strategies

- What should chooser do if parts are not equal?
- Assume that on following iterations splitter will play optimally = split each bin into two **equal** parts (might not be possible when number of chips is finite).
- Future configurations independent of chooser's choice.
- Fraction of bin i that will remain in bins 1..k when m iterations remain

$$\Phi(i,k,m) \doteq \frac{1}{2^m}\binom{m}{\leq k-i} = \frac{1}{2^m}\sum_{j=0}^{k-i}\binom{m}{j}$$

- Configuration: $\vec{n} = < n_0, n_1, \ldots, n_k >$
- Potential: The number of chips that will remain in bins 0..k when m iterations remain and both sides play optimally:

$$\Psi(\vec{n},k,m) = \sum_{i=0}^{k} n_i \Phi(i,k,m)$$

Combining experts, the binary prediction case

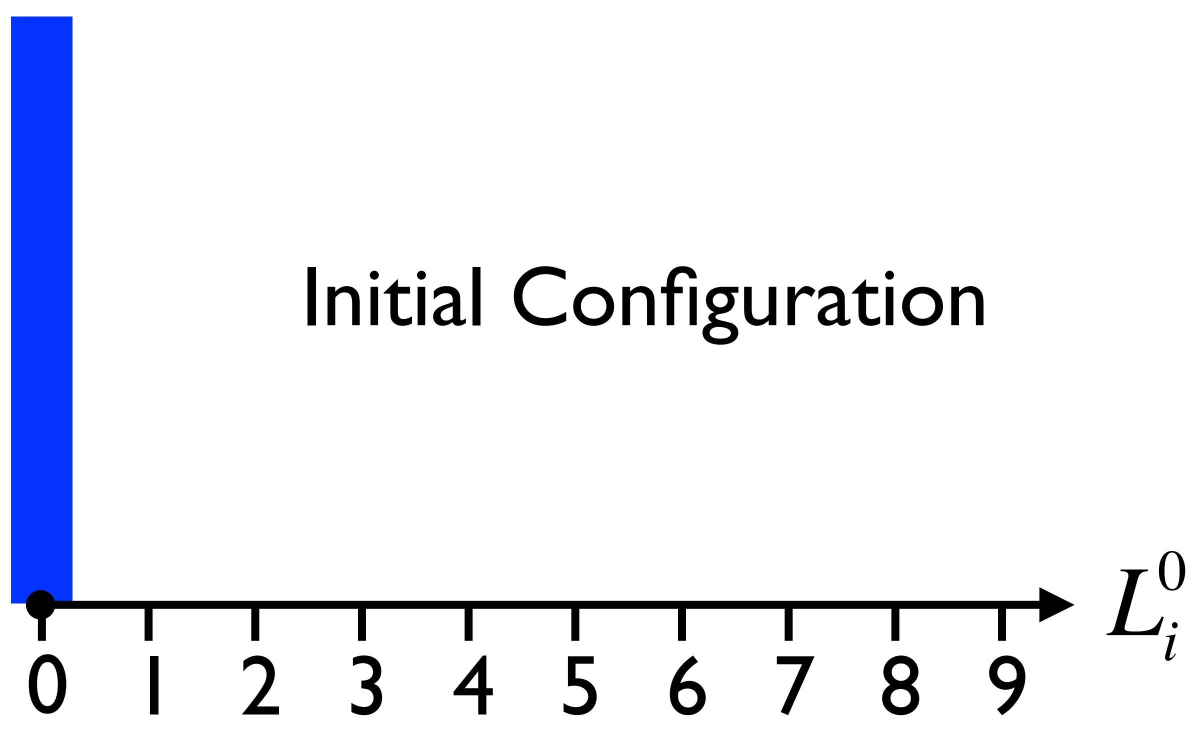
- Goal is to predict a binary sequence, making as few mistakes as possible.
- There are N experts.
- All predictions are binary and deterministic.
- A-priori knowledge: there is an expert that never makes more than k mistakes.
- k=0 corresponds to the halving algorithm.

Combining experts as a drifting game

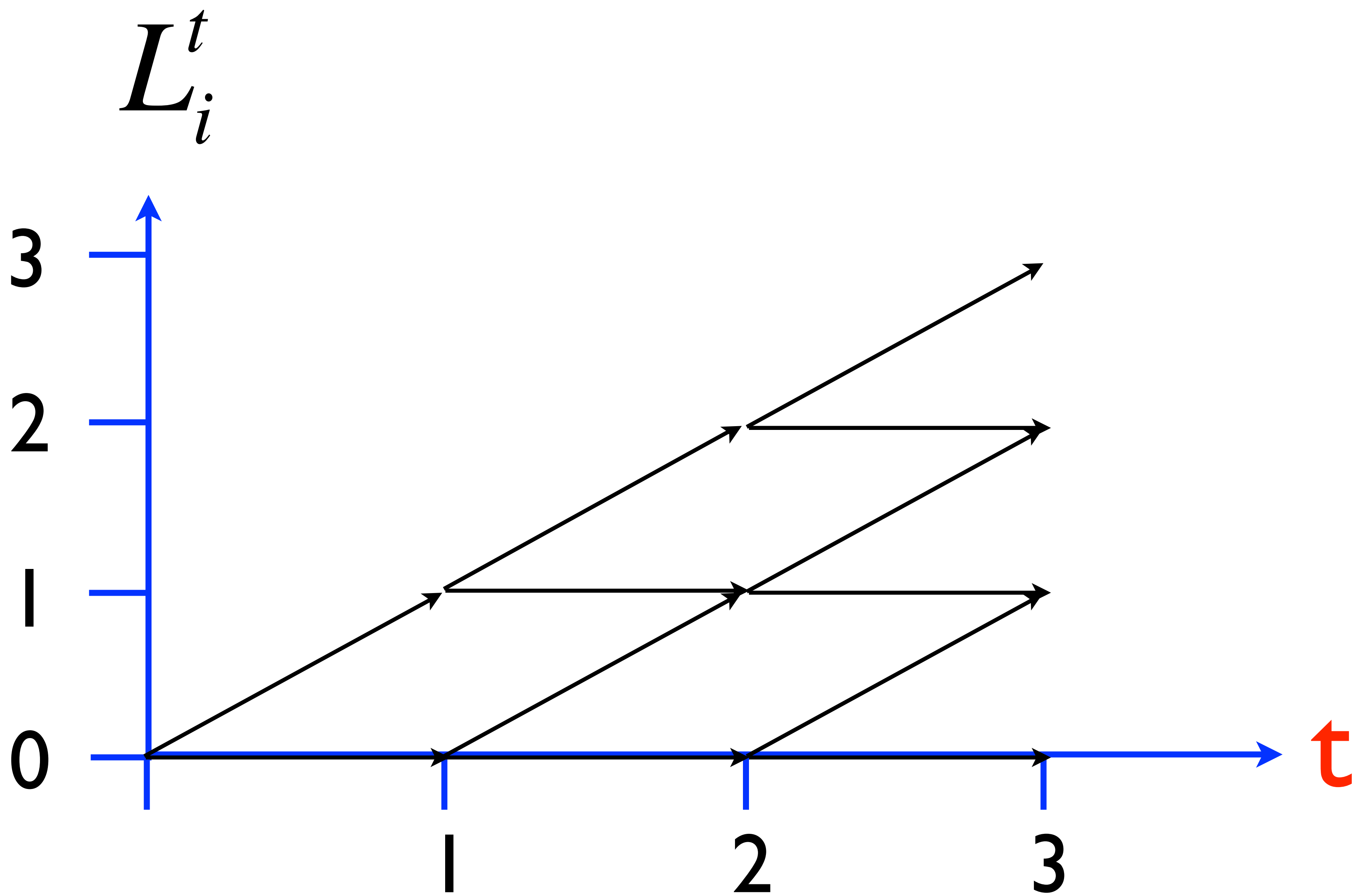
[Cesa-Bianchi, Freund, Helmbold, Warmuth 96]

Binary instantanous loss $l_i^t, l_A^t \in \{0,1\}$

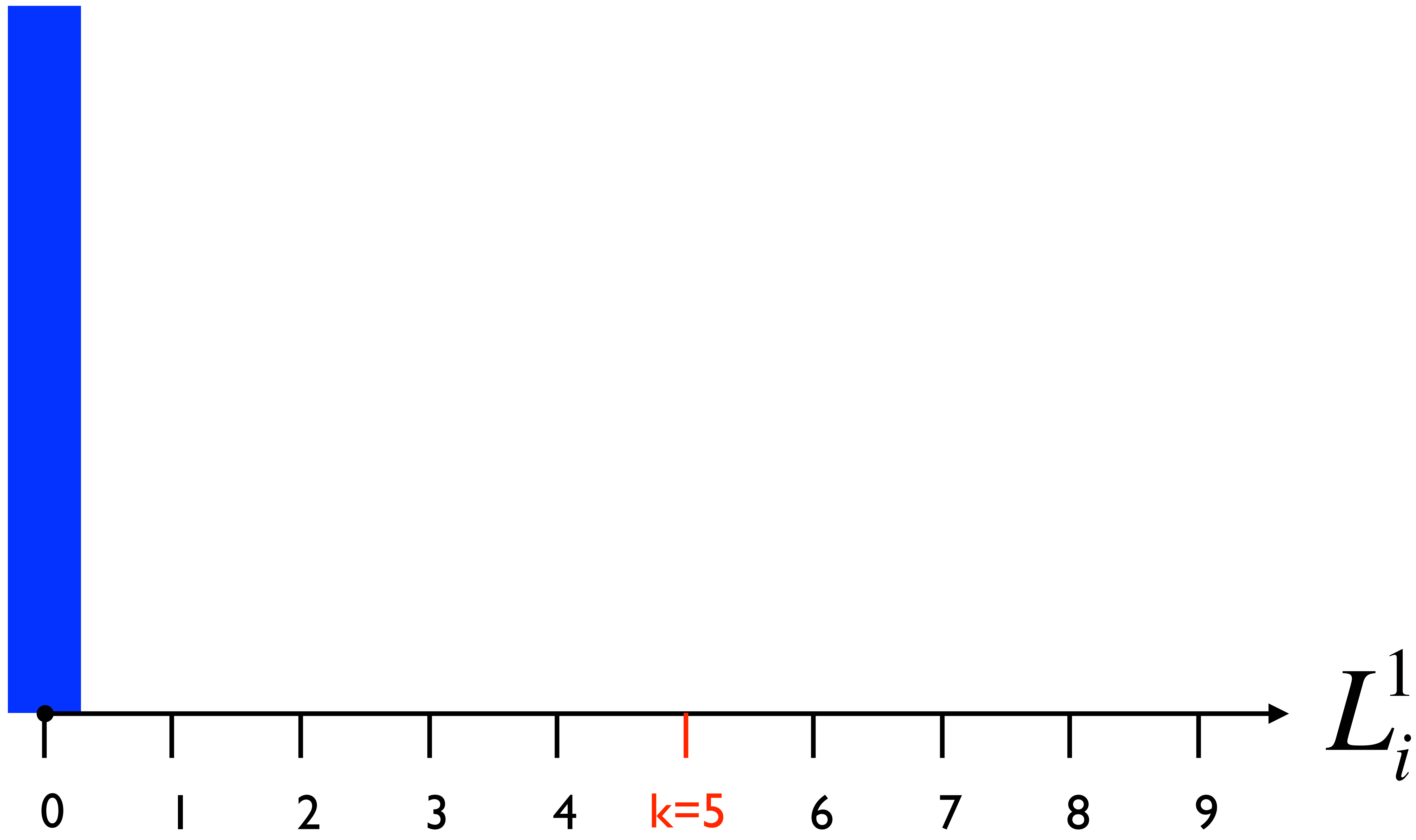
Bin s contains all experts for which $L_i^t = s$



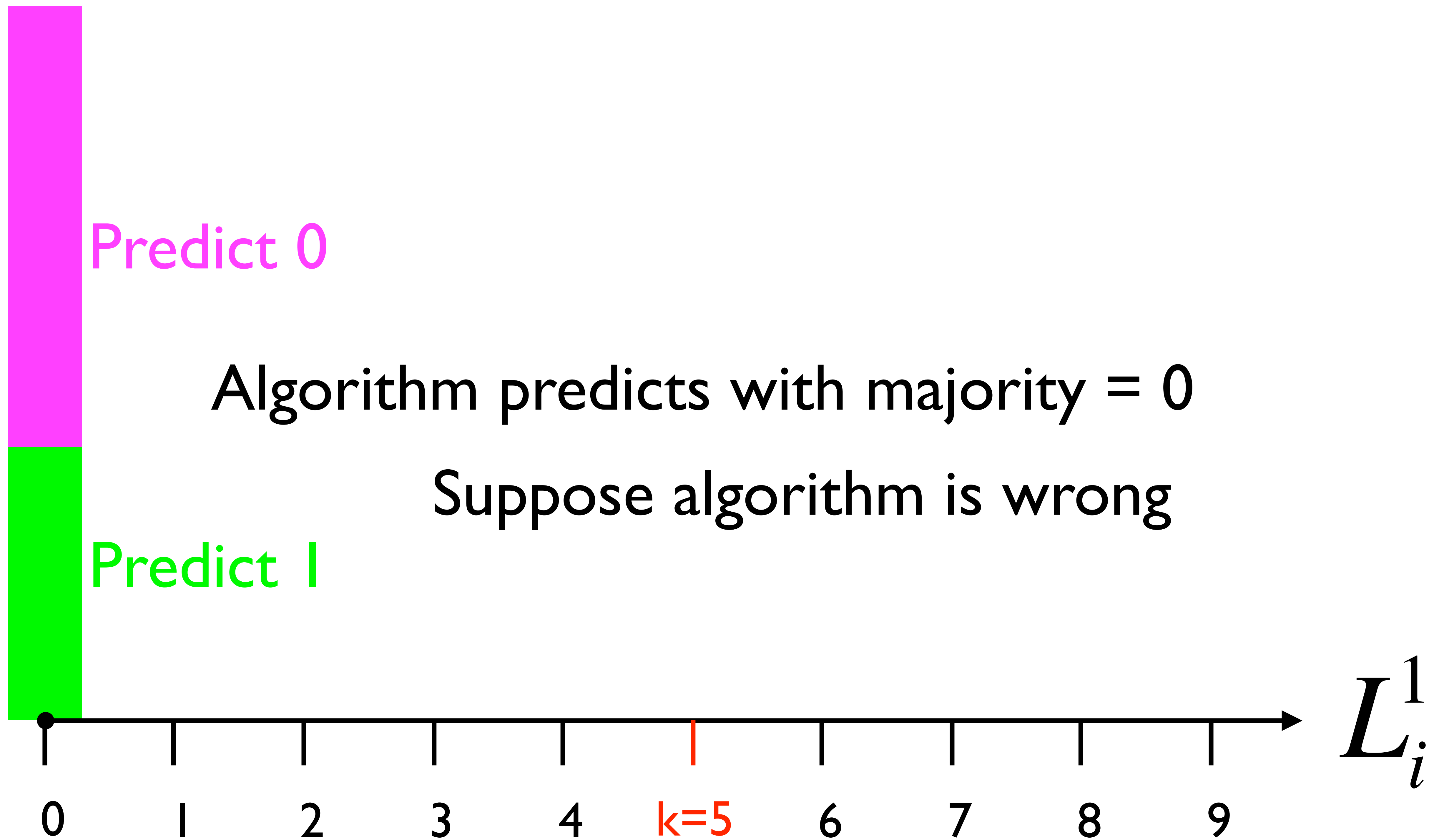
The game lattice

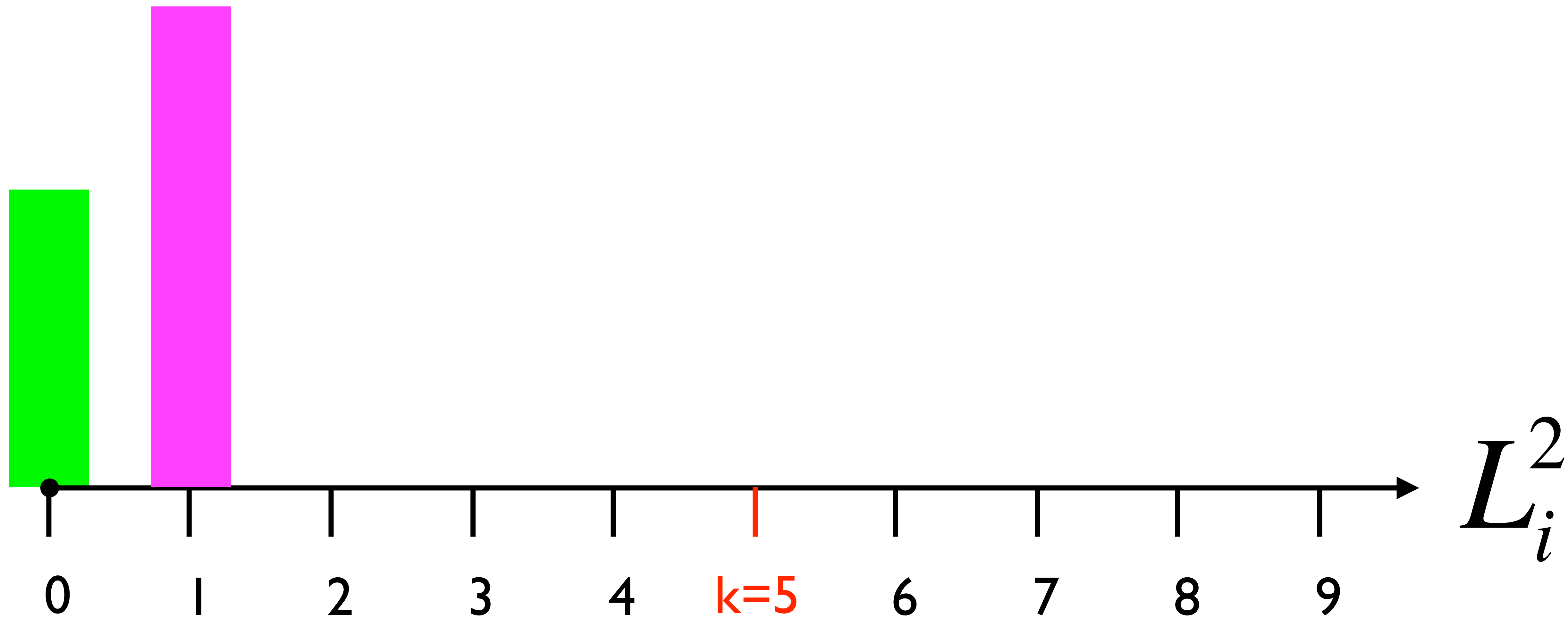


Initial configuration t=1



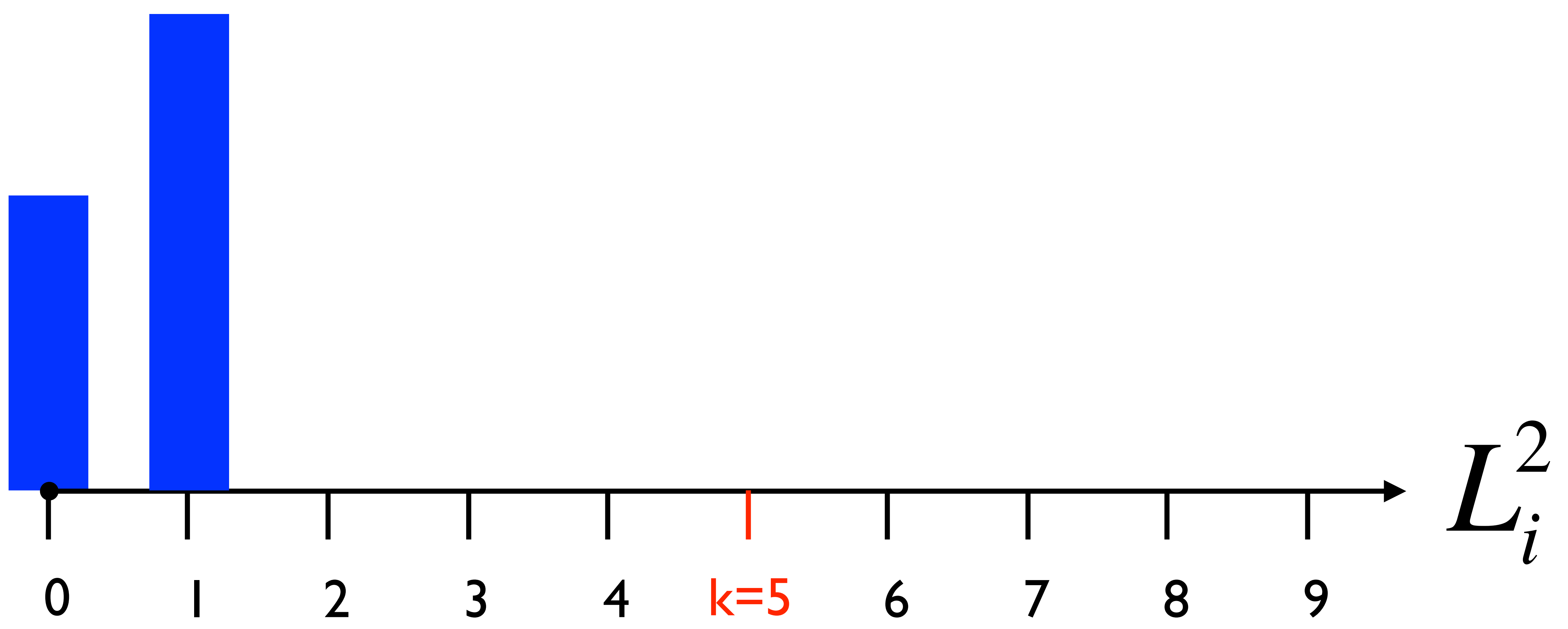
Experts predictions t=1





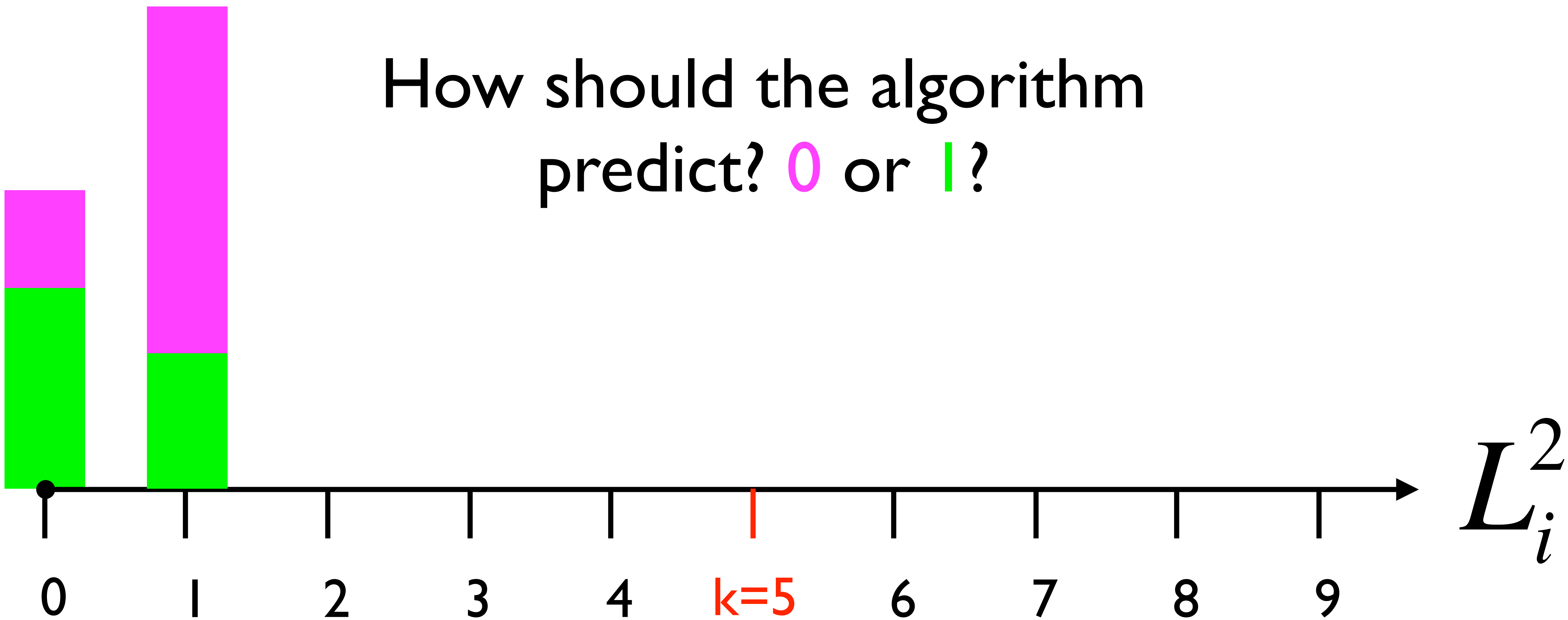
0
1
2
3
4
k=5
6
7
8
9
L_i^2

configuration at t=2



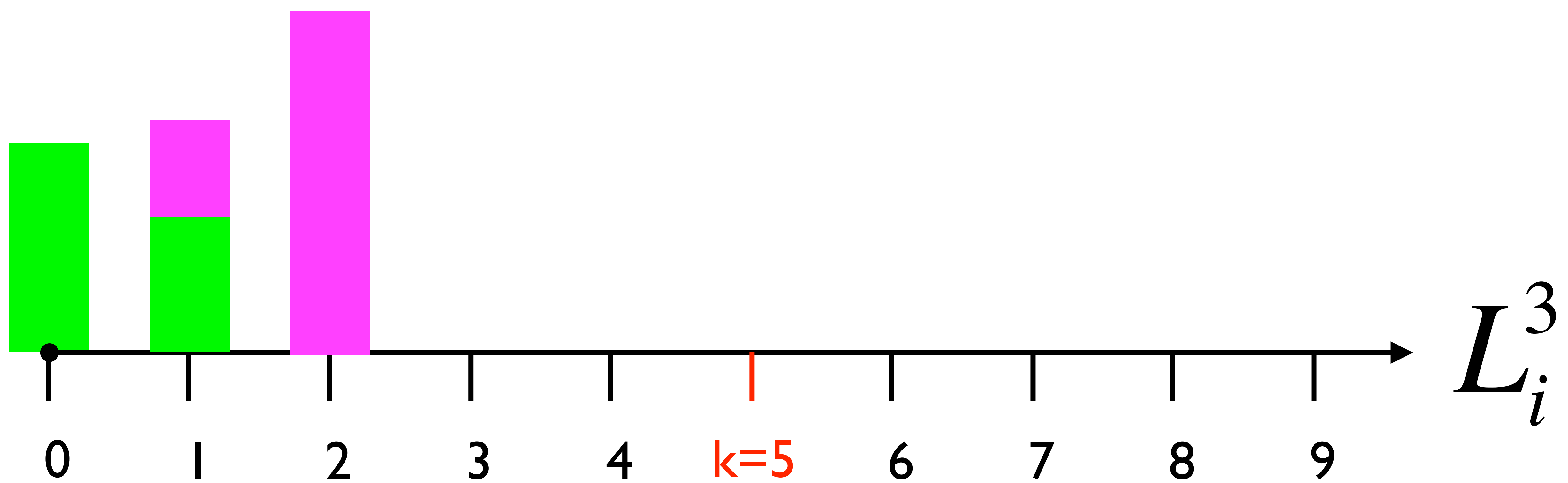
Experts predictions t=2

How should the algorithm predict? 0 or 1?



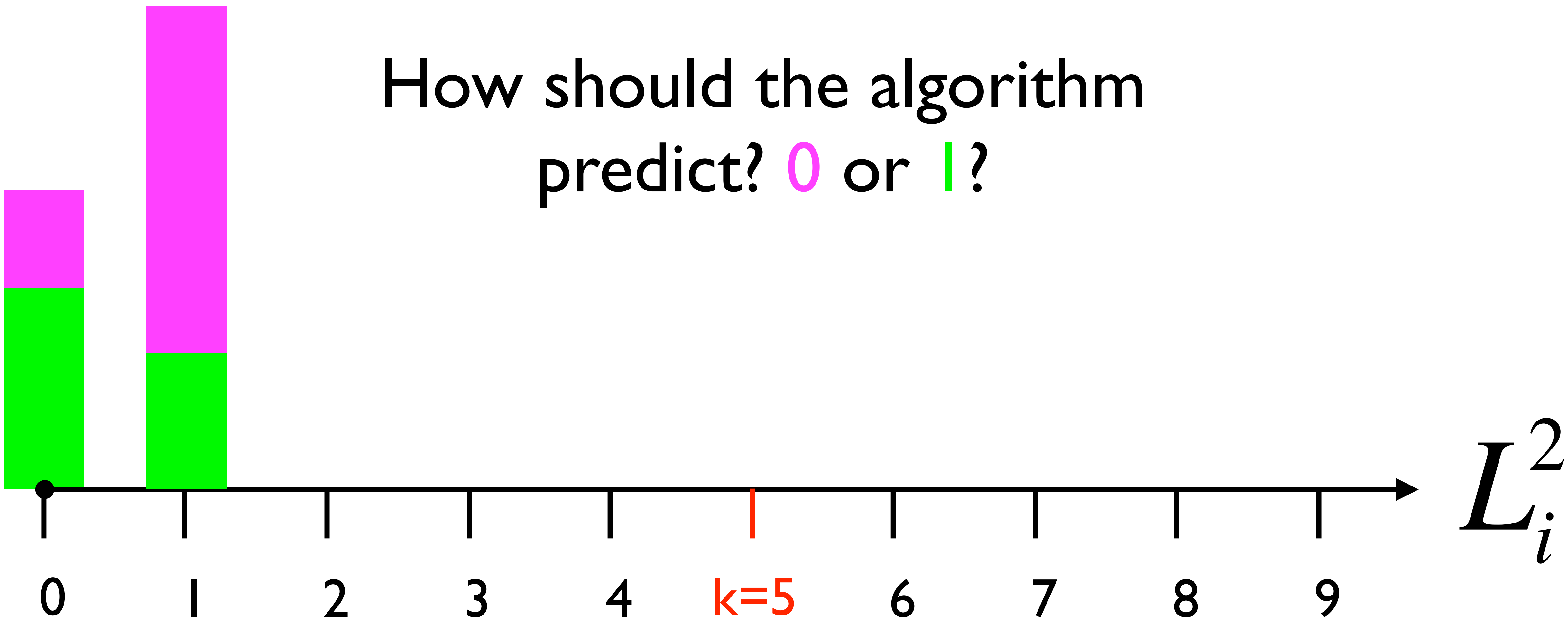
configuration t=3

Prediction is 0 and outcome is 1



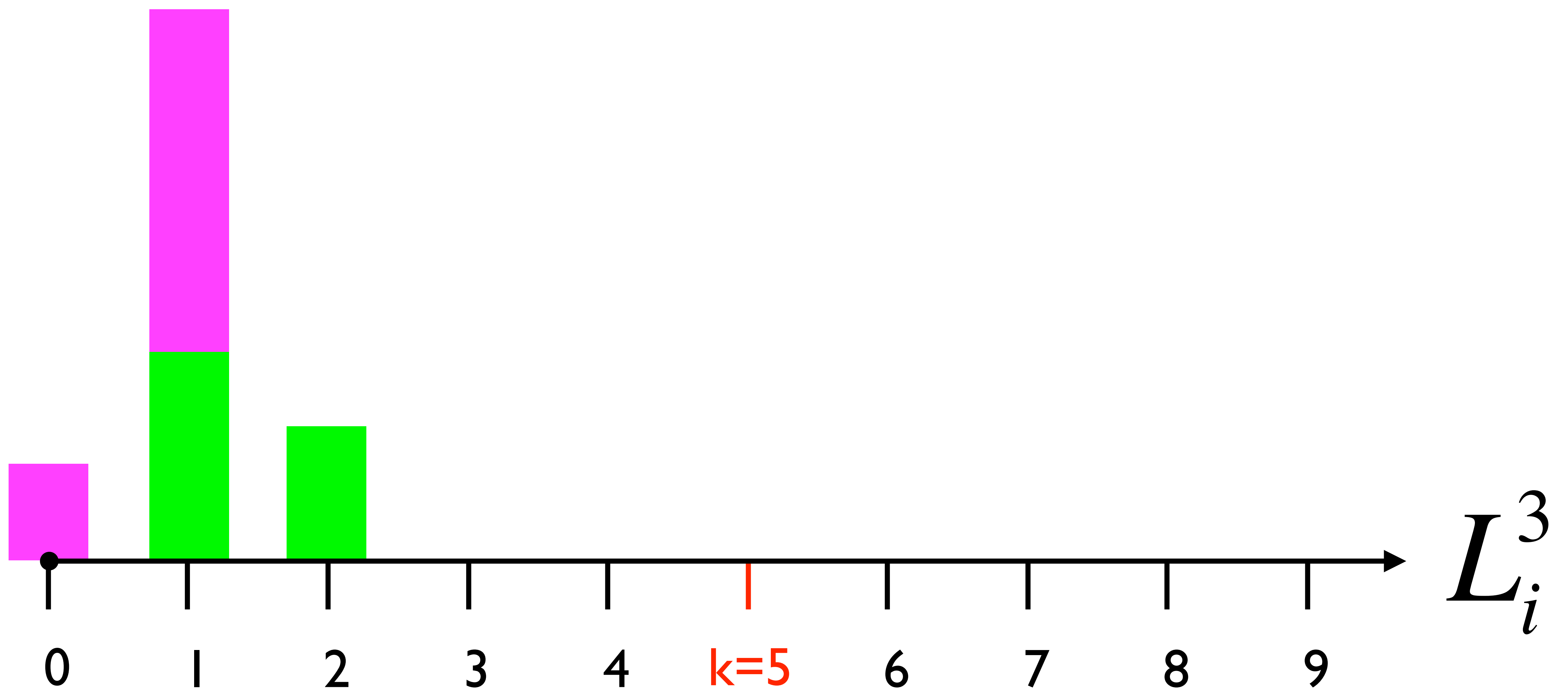
Experts predictions t=2

How should the algorithm predict? 0 or 1?



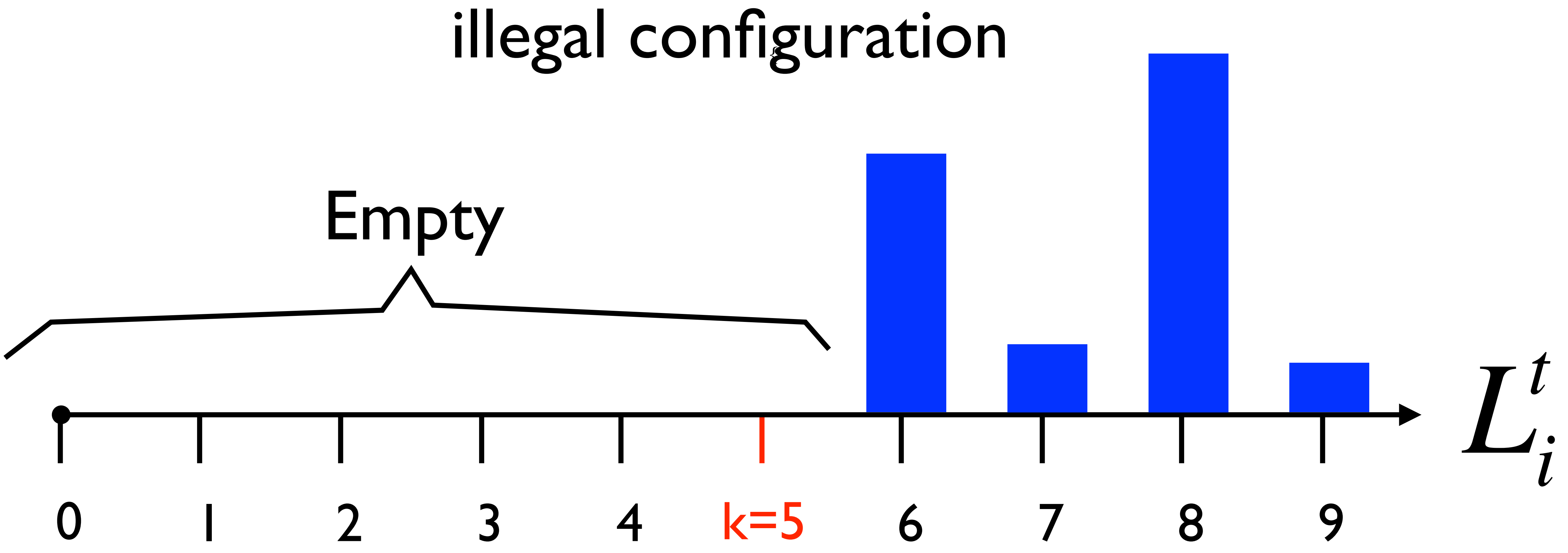
configuration t=3

Prediction is 1 and outcome is 0



If an error will lead to this configuration
then an error is not possible
$\Rightarrow$ this is a safe prediction

Algorithm's goal is to get to an illegal configuration
with the smallest number of mistakes.

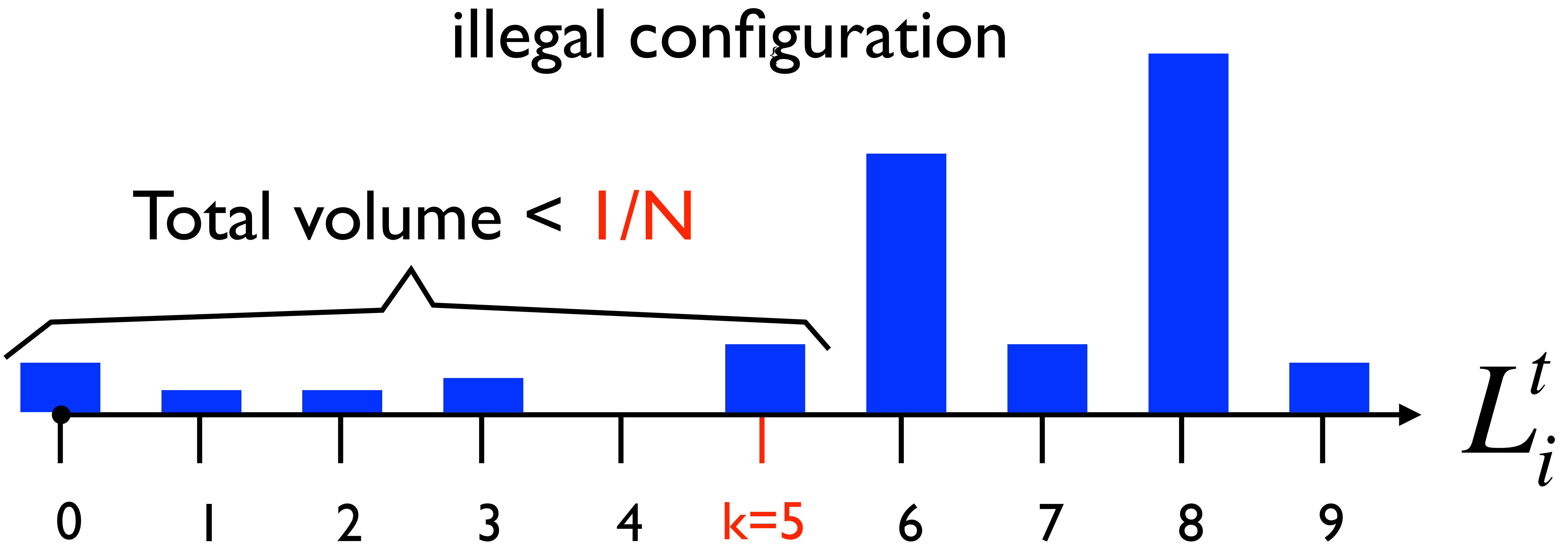


Helping the adversary.

- Assume that the set of experts is continuous, arbitrarily divisible.
- a-priori knowledge: 1/N fraction of the expert "mass" have cumulative loss at most k
- Find algorithm with the tightest uniform upper bound on the cumulative loss.

If an error will lead to this configuration
then an error is not possible
$\Rightarrow$ this is a safe prediction

Algorithm's goal is to get to an illegal configuration with the smallest number of mistakes.



An optimal adversarial strategy

- Split each bin to two equal parts. Algorithm's prediction is always incorrect.
- Equivalently: predictions of each expert are IID 0,1 with probabilities 1/2,1/2
- Same adversarial strategy was used to prove general lower bound on BLG

Optimal prediction strategy

- Assume that adversary will play optimally from the next iteration until the end of the game.
- Choose as the next configuration the one that would end the game faster.
- Relevant only when adversary plays sub-optimally, when adversary plays optimally the two next configurations are identical.

Binomial weights strategies

- What should chooser do if parts are not equal?
- Assume that on following iterations splitter will play optimally = split each bin into two **equal** parts (might not be possible when number of chips is finite).
- Future configurations independent of chooser's choice.
- Fraction of bin i that will remain in bins 1..k when m iterations remain

$$\phi(i,k,m) \doteq \frac{1}{2^m}\binom{m}{\leq k-i} = \frac{1}{2^m}\sum_{j=0}^{k-i}\binom{m}{j}$$

- Configuration: $\vec{n} = < n_0, n_1, \ldots, n_k >$
- Potential: The number of chips that will remain in bins 0..k when m iterations remain and both sides play optimally:

$$\Psi(\vec{n},k,m) = \sum_{i=0}^{k} n_i \phi(i,k,m)$$

Response to suboptimal play

- The chooser part so that the potential of the next configuration will be:
  - Predictor in online learning: Minimal weighting. Make game short (fewer mistakes).
  - Secret holder in Ulam's game: Maximal weighting. Make game long (More guesses).

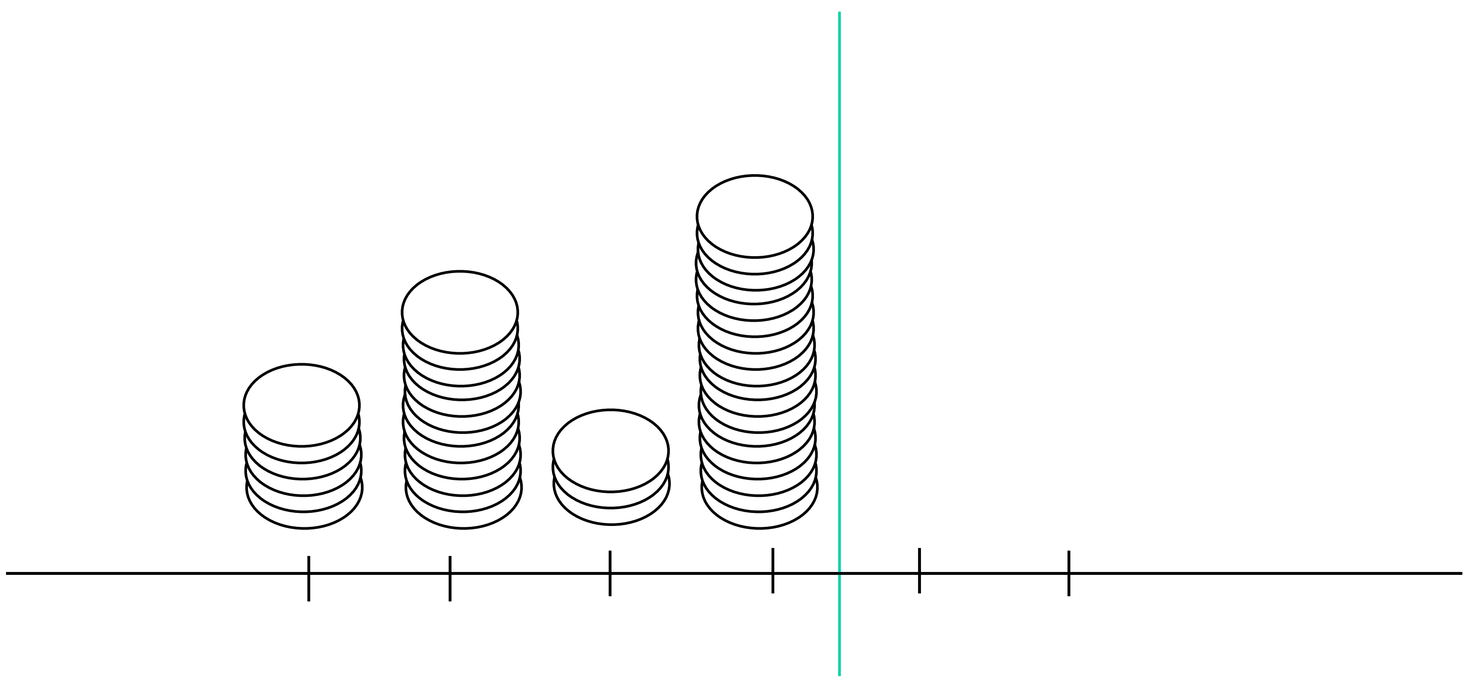
- When split is not even, the chooser might be able to improve bound:
  - Online learning: $m := \min\{q \in N : \Psi(\vec{n}, k, q) < 1\} - 1$
  - Ulam's game with k lies: $m := \max\{q \in N : \Psi(\vec{n}, k, q) \geq 1\} - 1$

Optimality of strategy

- When chips are indivisible there can be situations where optimal play by the splitter is impossible.
- [Spencer95] If number of chips is sufficiently large, equal-potential splits are always possible.
- A sufficient number of chips is $\Omega\left(2^{2^k}\right)$

Number of chips to infinity

Replace individual chips by chip mass



Optimal splitter strategy:
Split each bin into two equal parts

Equivalence to a random walk

- Both sides playing optimally is equivalent to each chip performing an independent random walk.
- Potential = probability of a chip in bin i ending in bins 0..K after m iteration
- Weight = difference between the potentials of a chip in its two possible locations on the following iteration.
- Chooser's optimal strategy: choose set with smaller (larger) weight
- Worst case behavior of experts is to perform a random walk!

Response to suboptimal play

- The chooser part so that the potential of the next configuration will be:
  - Predictor in online learning: Minimal weighting. Make game short (fewer mistakes).
  - Secret holder in Ulam's game: Maximal weighting. Make game long (More guesses).

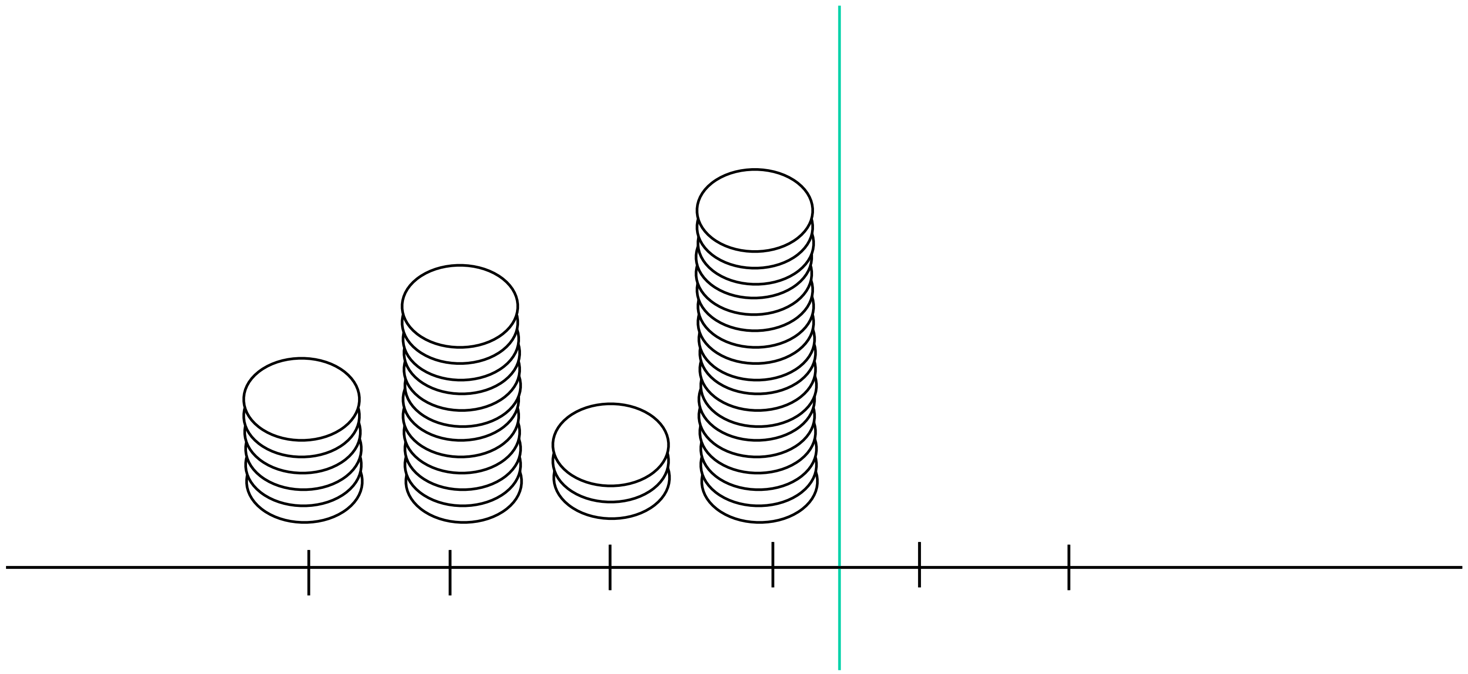
- When split is not even, the chooser might be able to improve bound:
  - online learning: $m := \min\{q \in N : \Psi(\vec{n},k,q) < 1\} - 1$
  - Ulam's game with k lies: $m := \max\{q \in N : \Psi(\vec{n},k,q) \geq 1\} - 1$

Optimality of strategy

- When chips are indivisible there can be situations where optimal play by the splitter is impossible.
- [Spencer95] If number of chips is sufficiently large, equal-potential splits are always possible.
- A sufficient number of chips is $\Omega\left(2^{2^k}\right)$

Number of chips to infinity

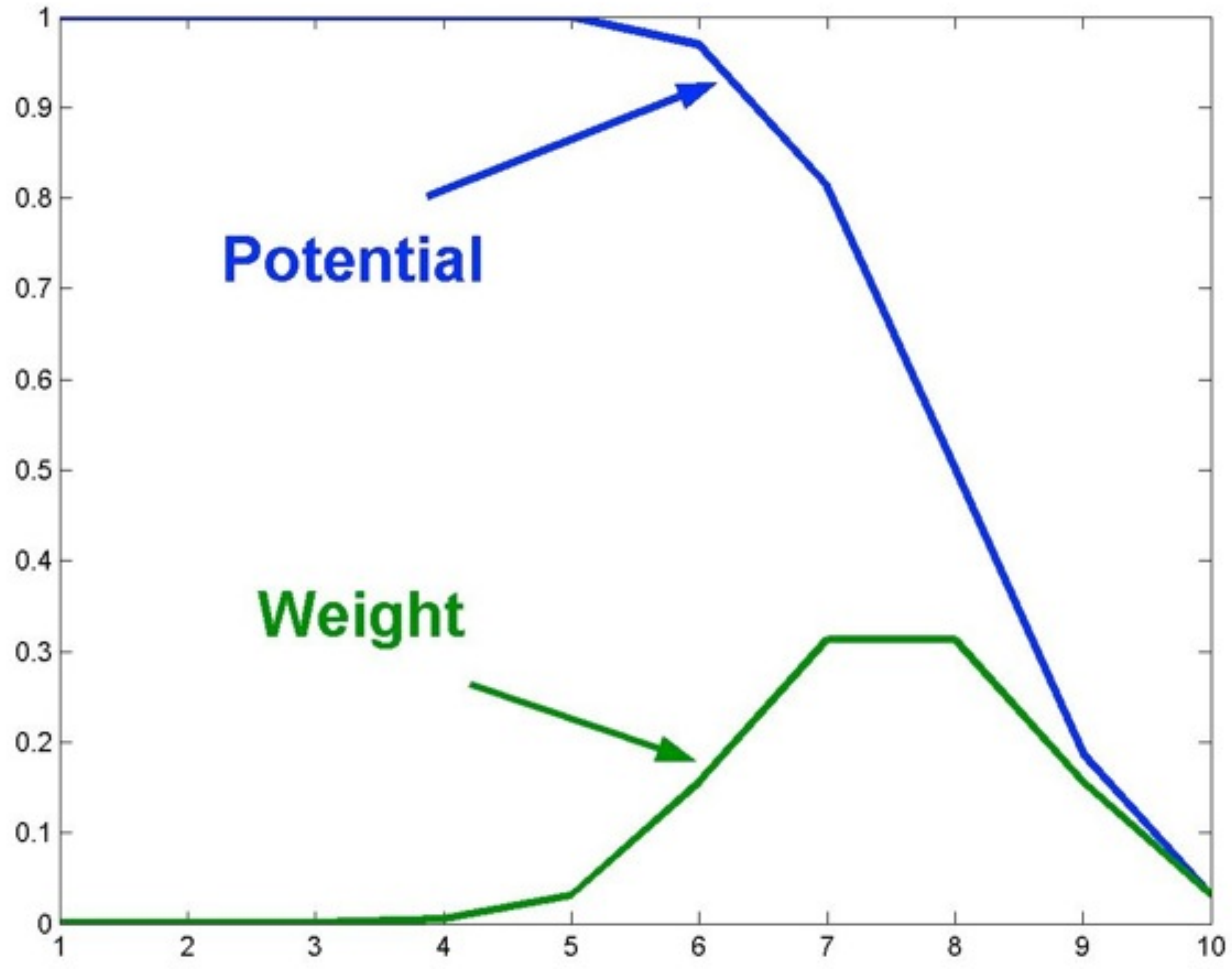
Replace individual chips by chip mass



Optimal splitter strategy:
Split each bin into two equal parts

Example potential and weight

m=5; k=10



Boosting

- A method for improving classifier accuracy
- Weak Learner: a learning algorithm generating rules slightly better than random guessing.
- Basic idea: re-weight training examples to force weak learner into different parts of the space.
- Combine weak rules by a majority vote.

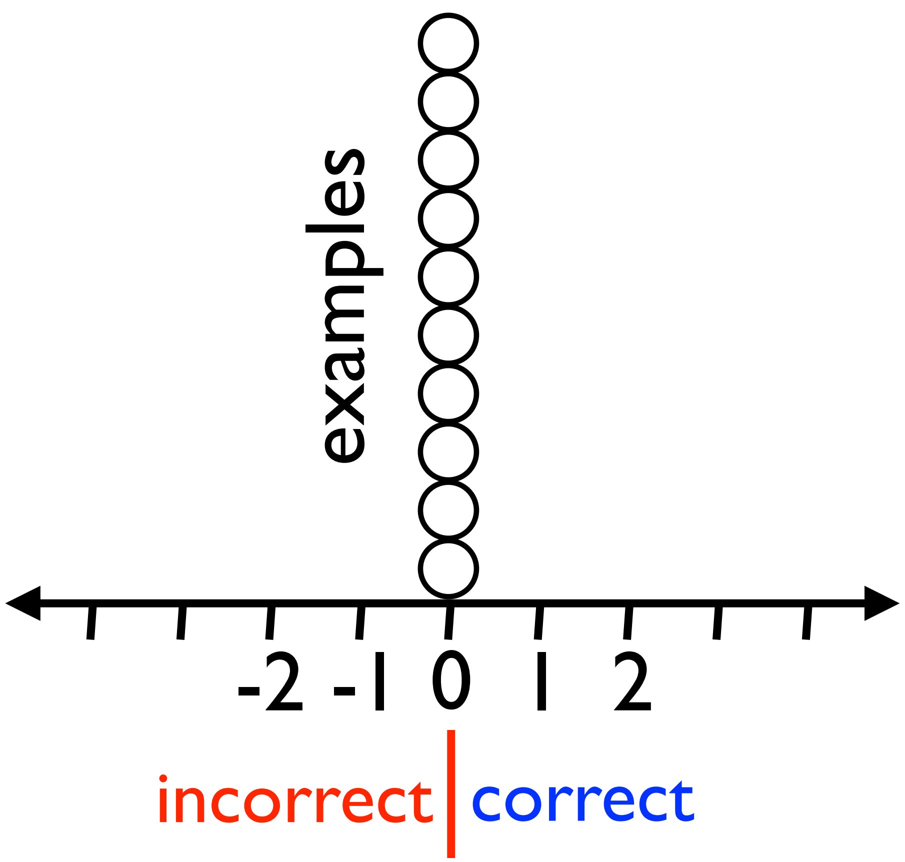
Boost by Majority (BBM)

[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.

BBM as a drifting game

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

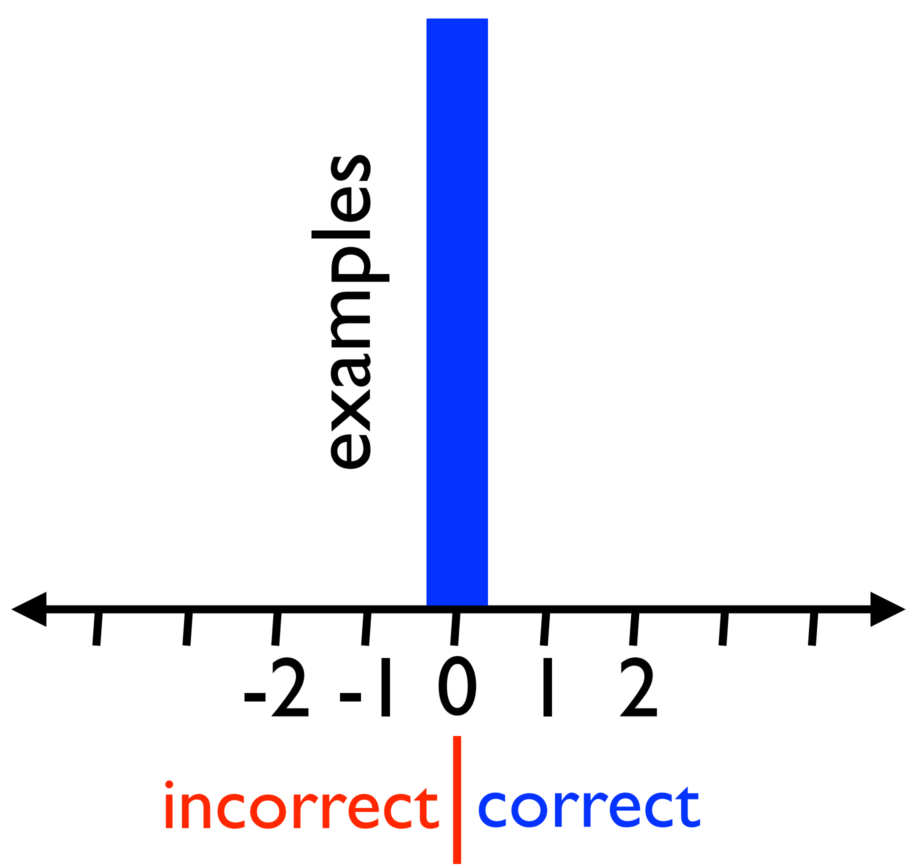


Initial Configuration

$$s \doteq y \sum_{t=1}^{T} h_t(x)$$

The continuous chip limit

- Number of training examples increases to infinity.
- Alternatively - think of examples as a probability mass with probability measure μ defined on it.

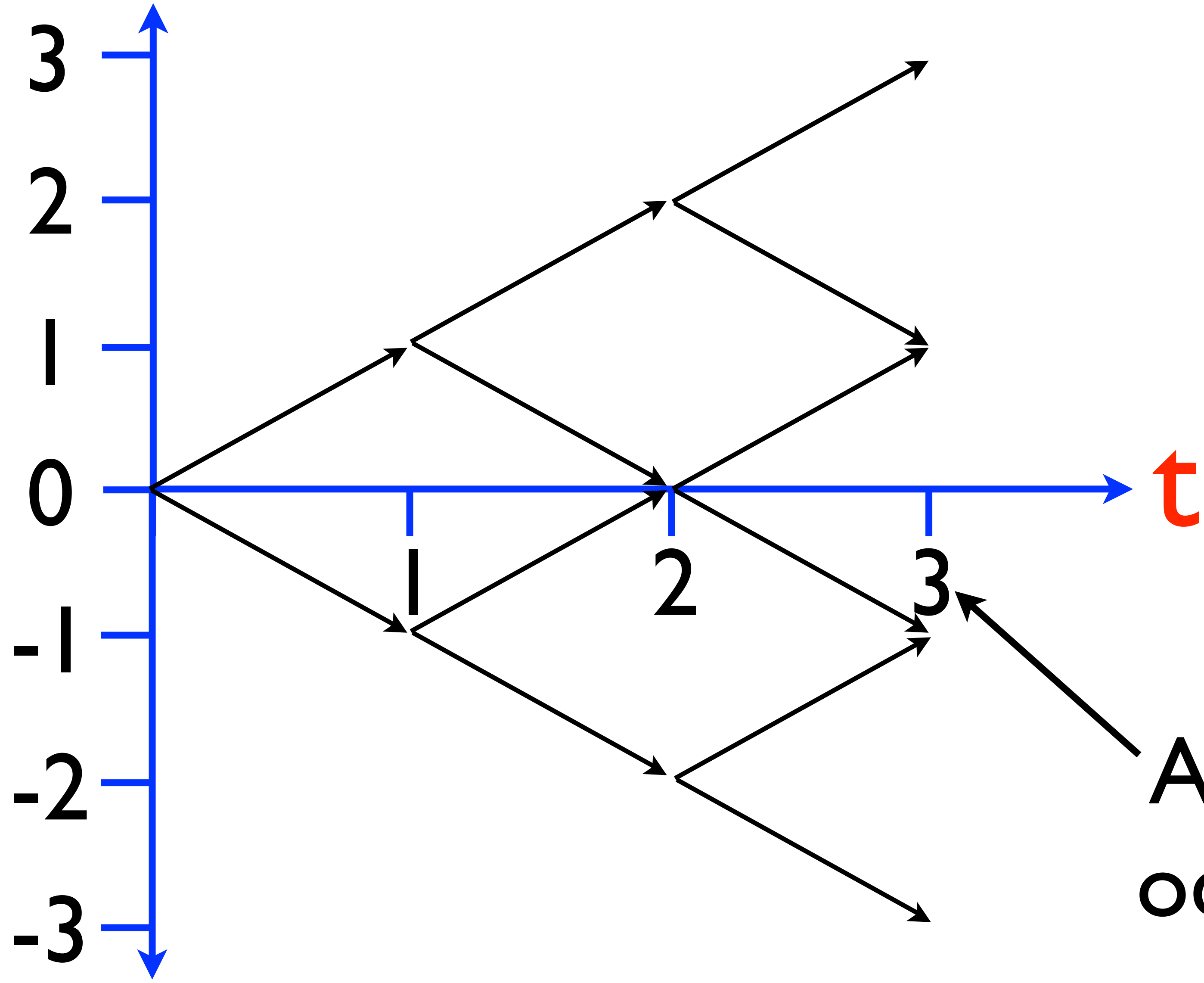


Initial Configuration

$$s = y\sum_{t=1}^{T} h_t(x)$$

The boosting game lattice

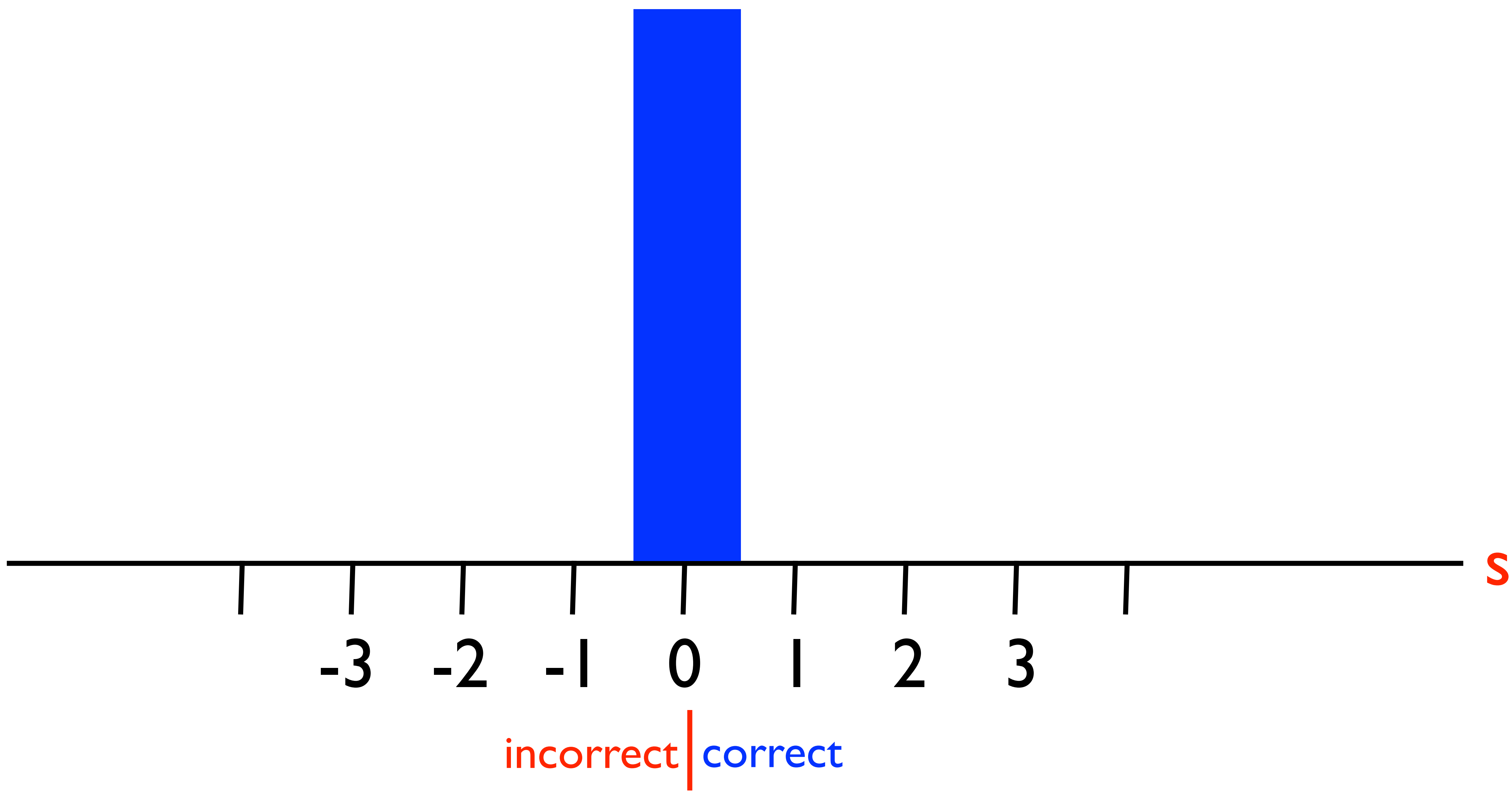
$$s \doteq y \sum_{t=1}^{T} h_t(x)$$



Assume T is odd to avoid ties

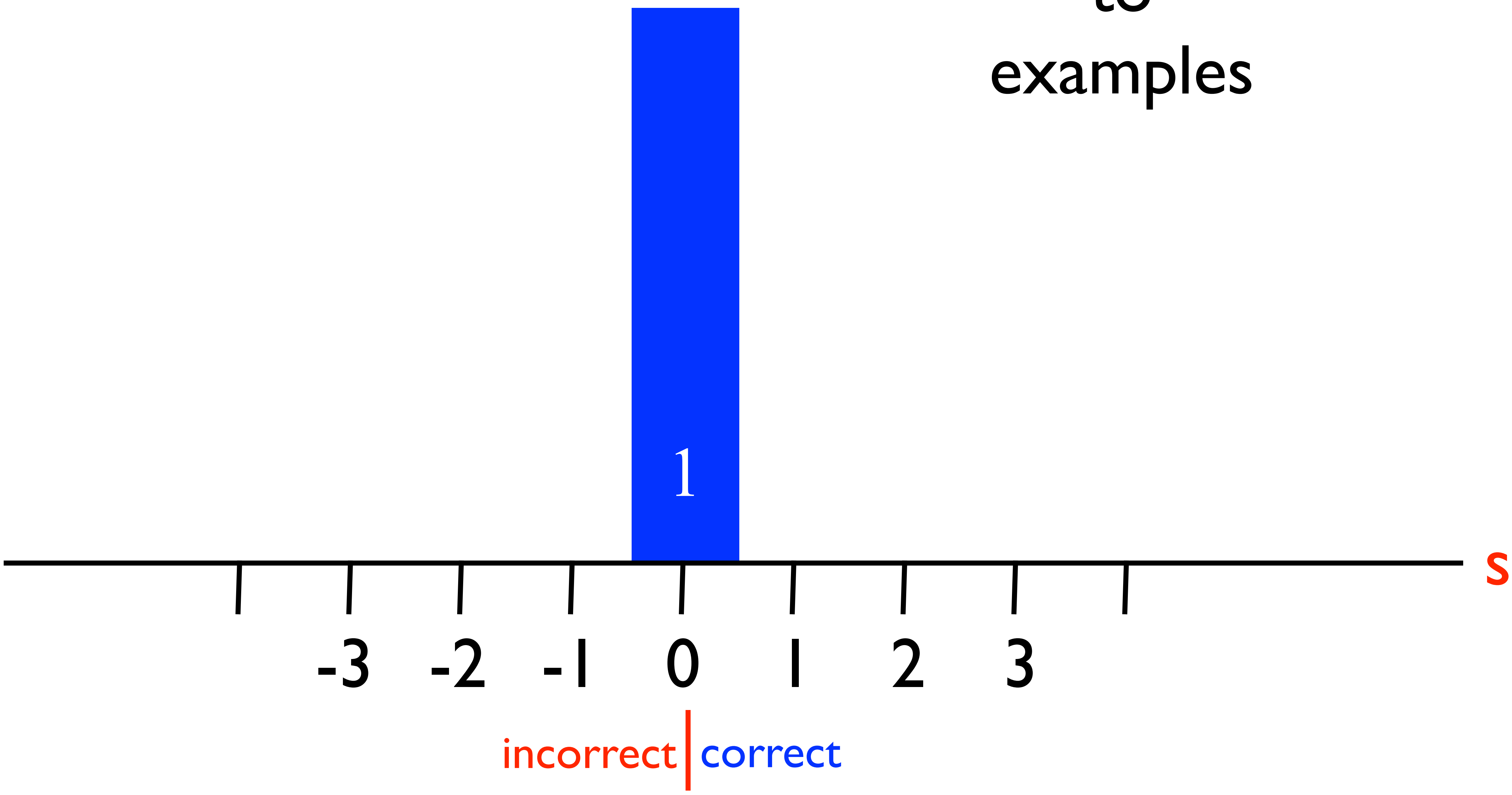
γ =0.1

Initial configuration



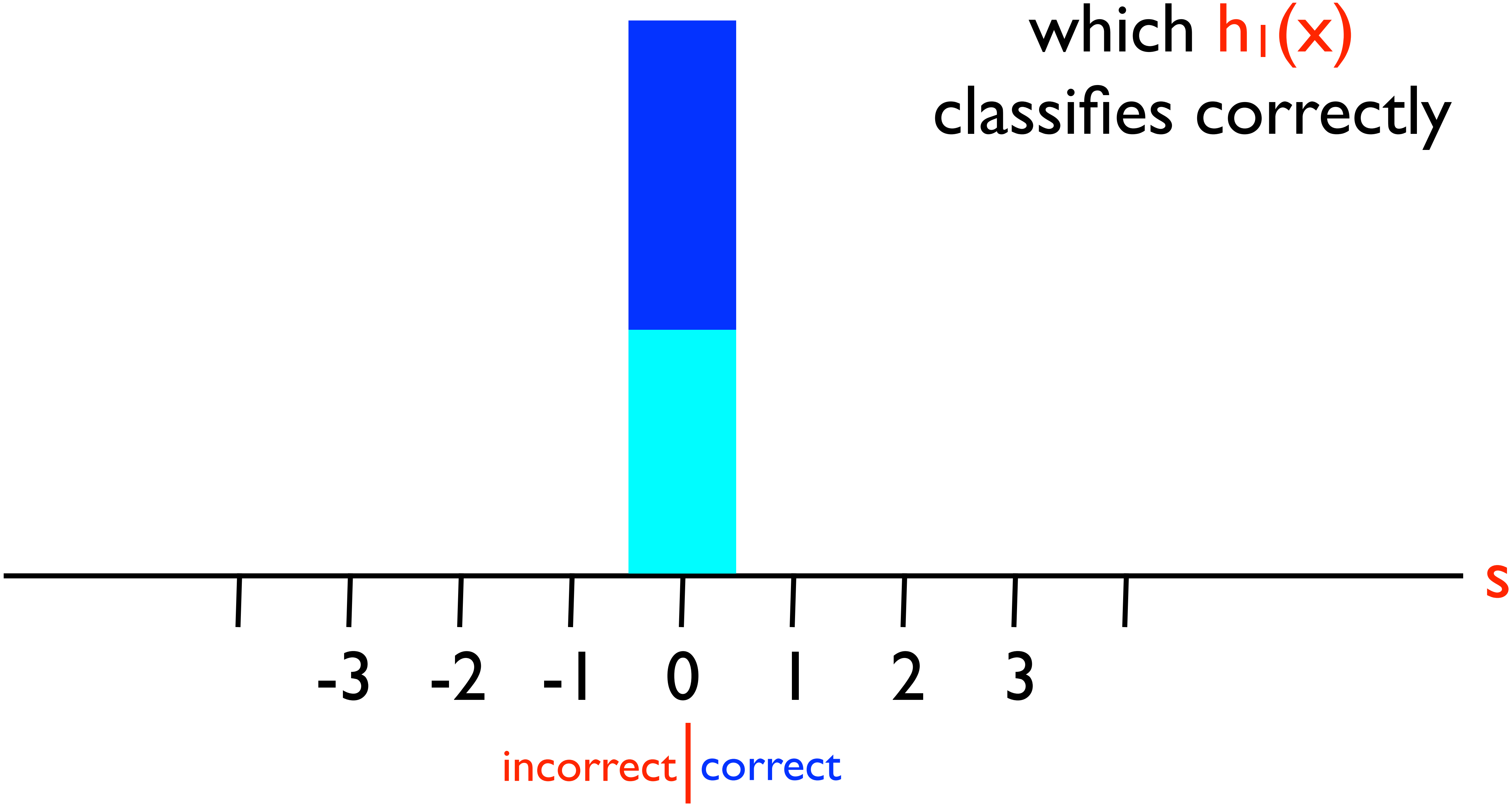
$\gamma = 0.1$

Booster assigns weights to examples



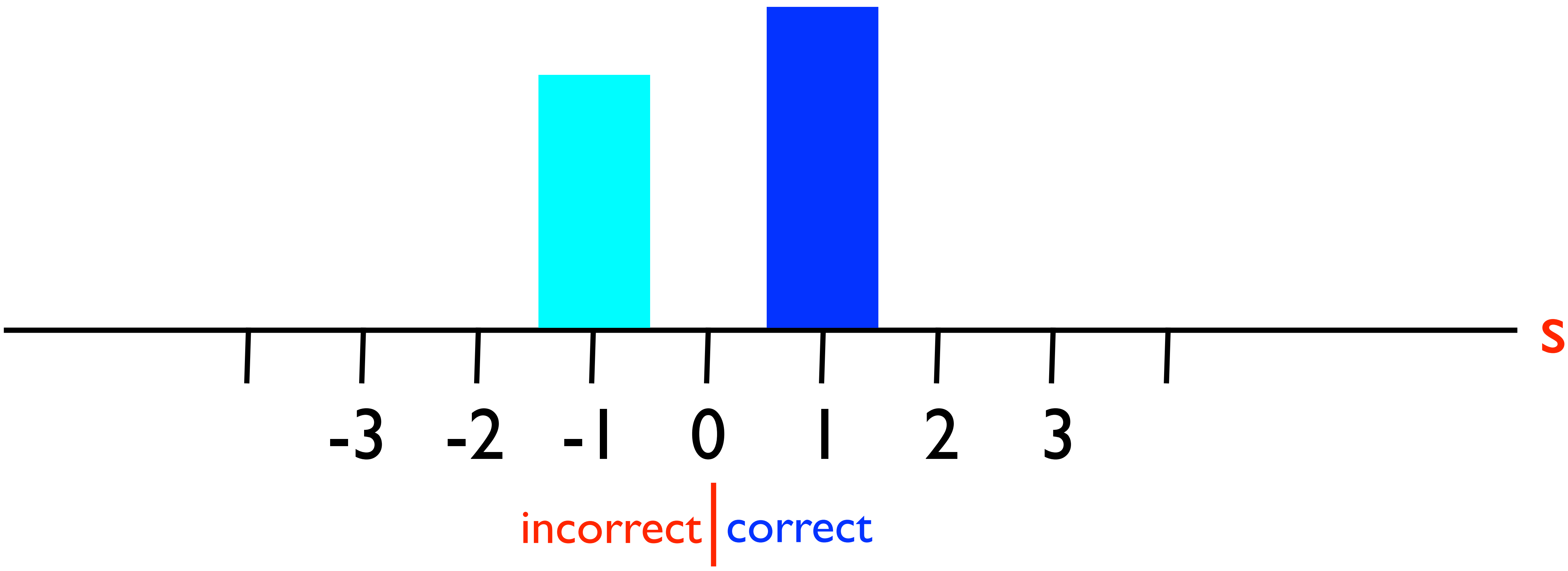
$\gamma = 0.1$

Weak learner chooses subset with weight $1/2 + \gamma$ which $h_1(x)$ classifies correctly



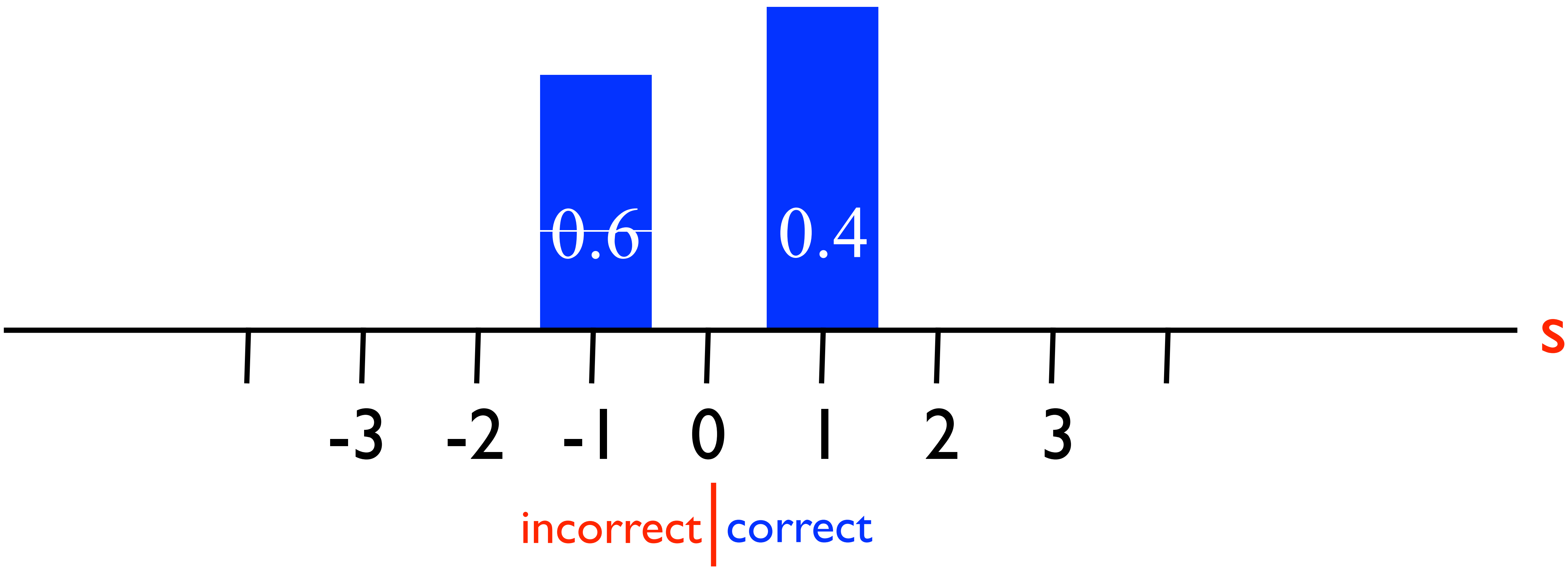
$\gamma = 0.1$

Weak learner chooses subset with weight $1/2 + \gamma$ which $h_1(x)$ classifies correctly



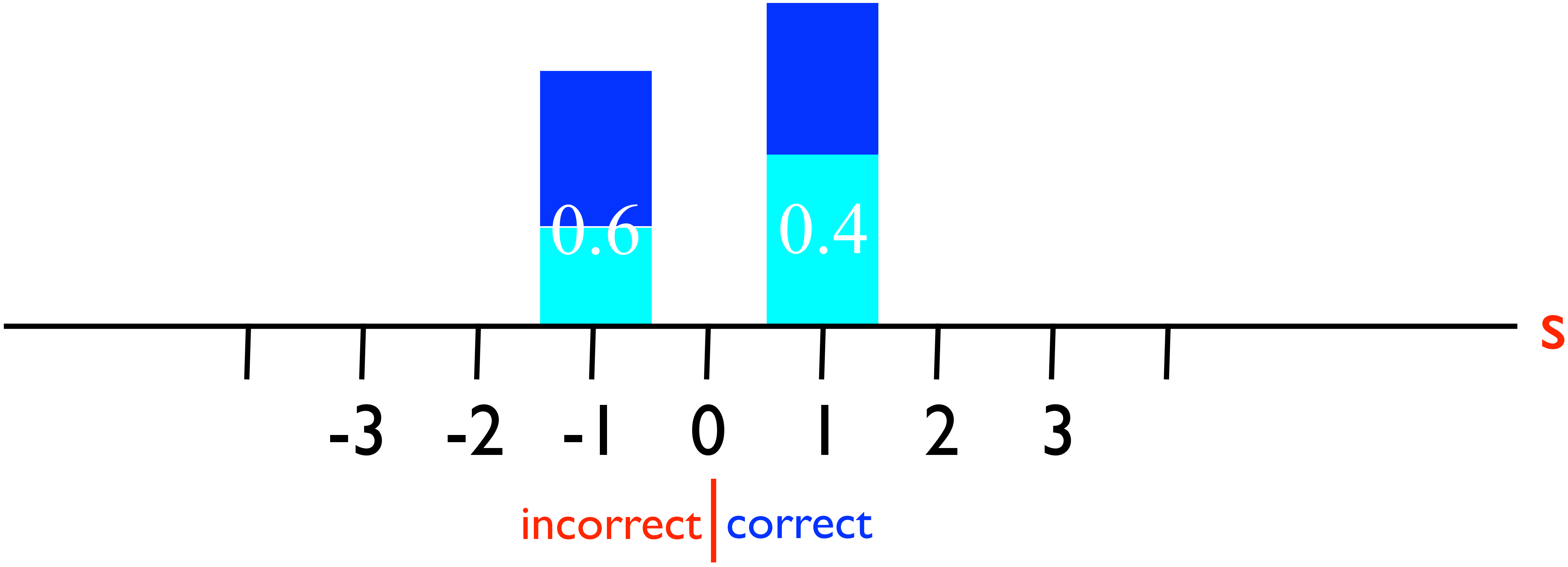
$\gamma = 0.1$

Booster
assigns weights
to
examples



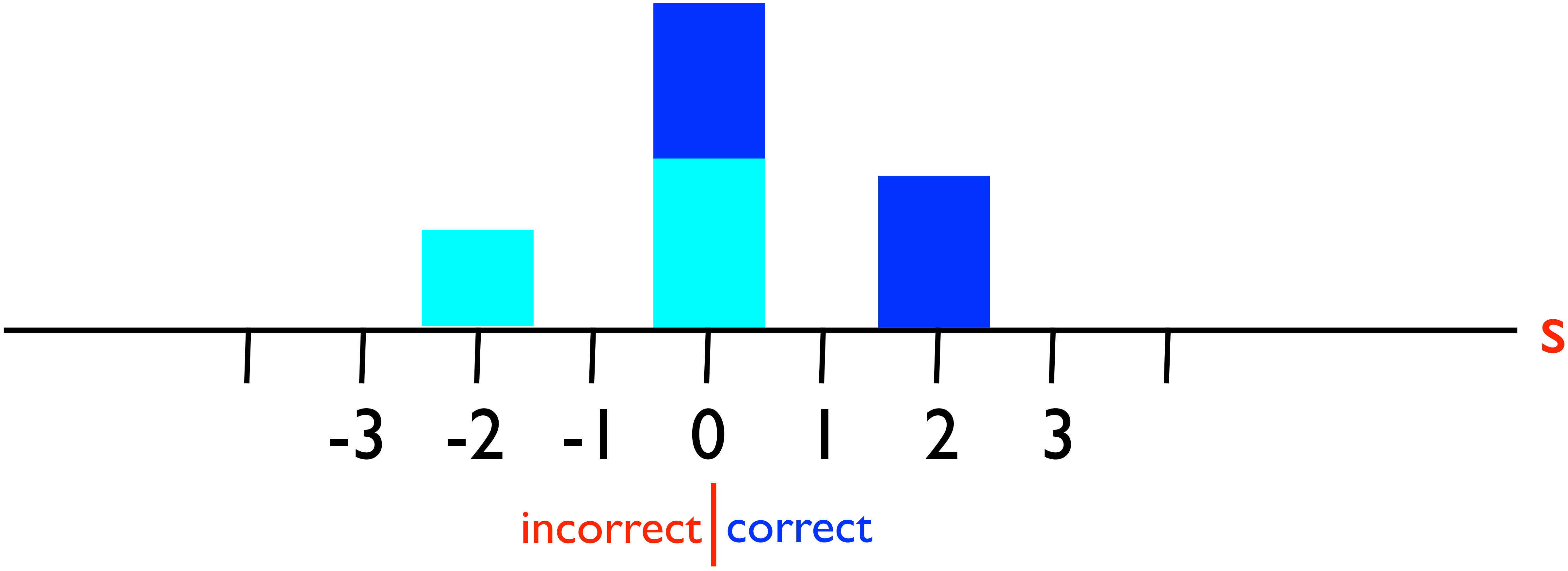
$\gamma = 0.1$

Weak learner chooses subset with weight $1/2 + \gamma$ which $h_2(x)$ classifies correctly



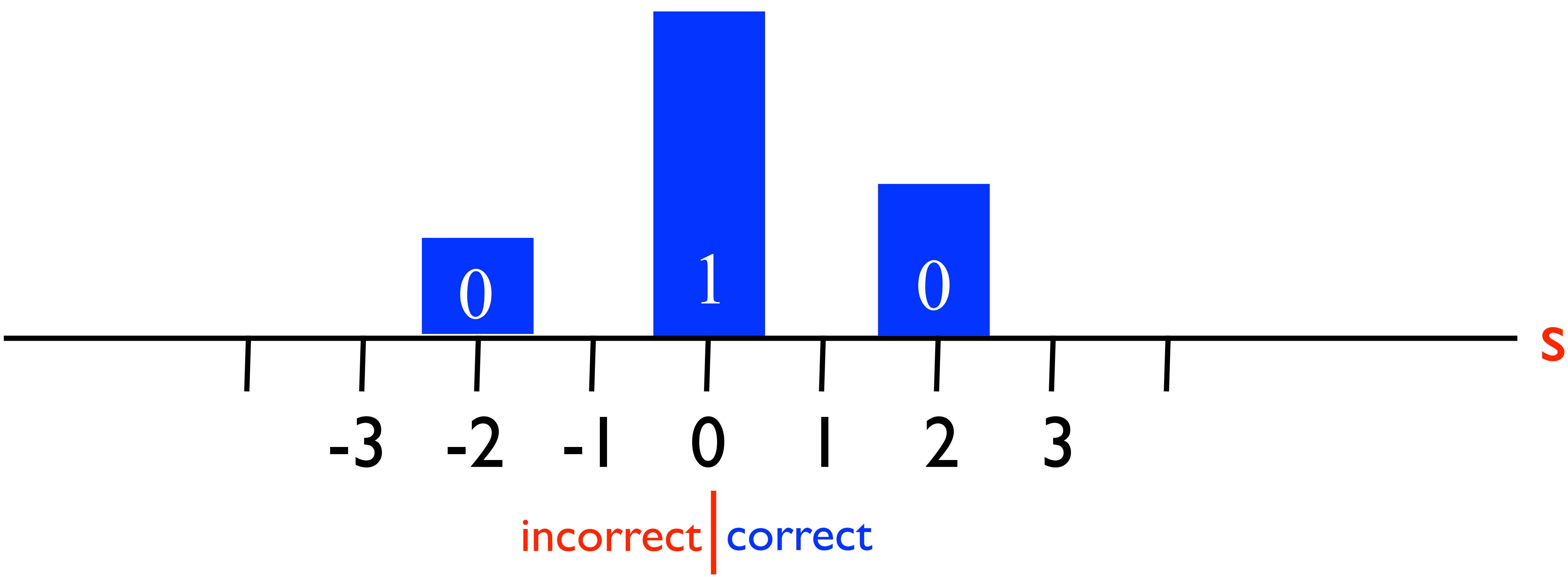
$\gamma = 0.1$

Weak learner chooses subset with weight $1/2 + \gamma$ which $h_2(x)$ classifies correctly



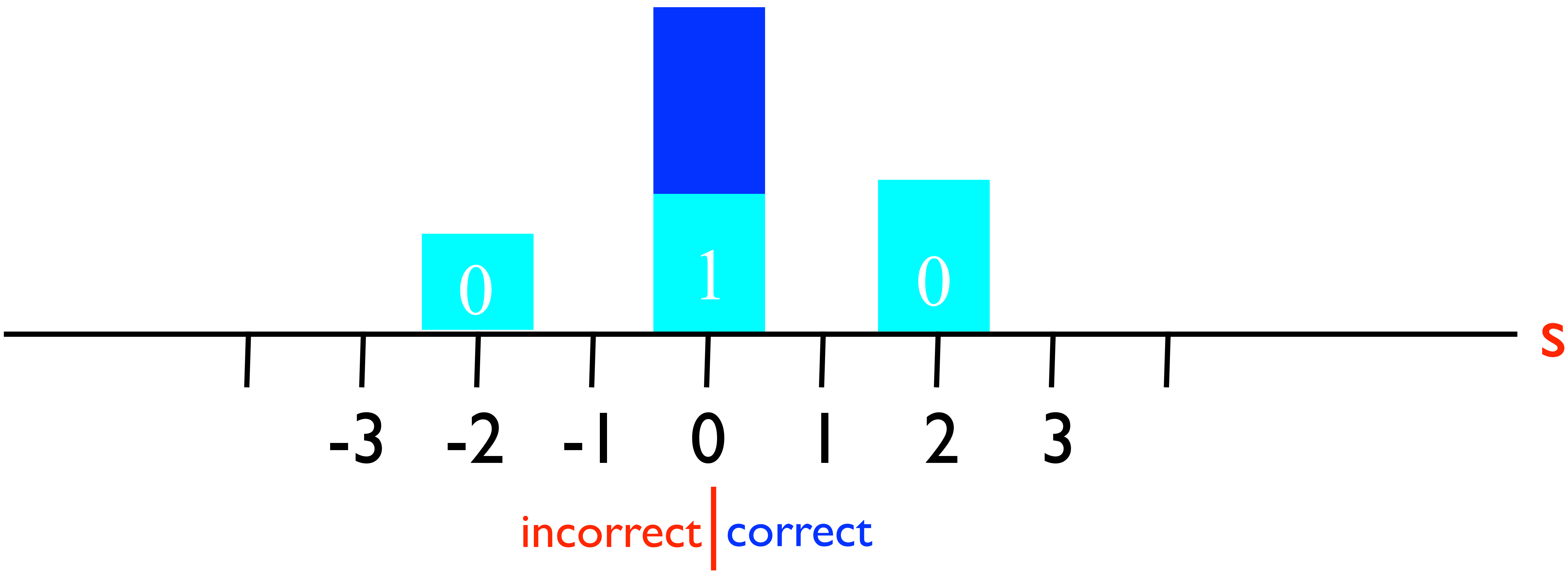
$\gamma = 0.1$

Booster assigns weights to examples



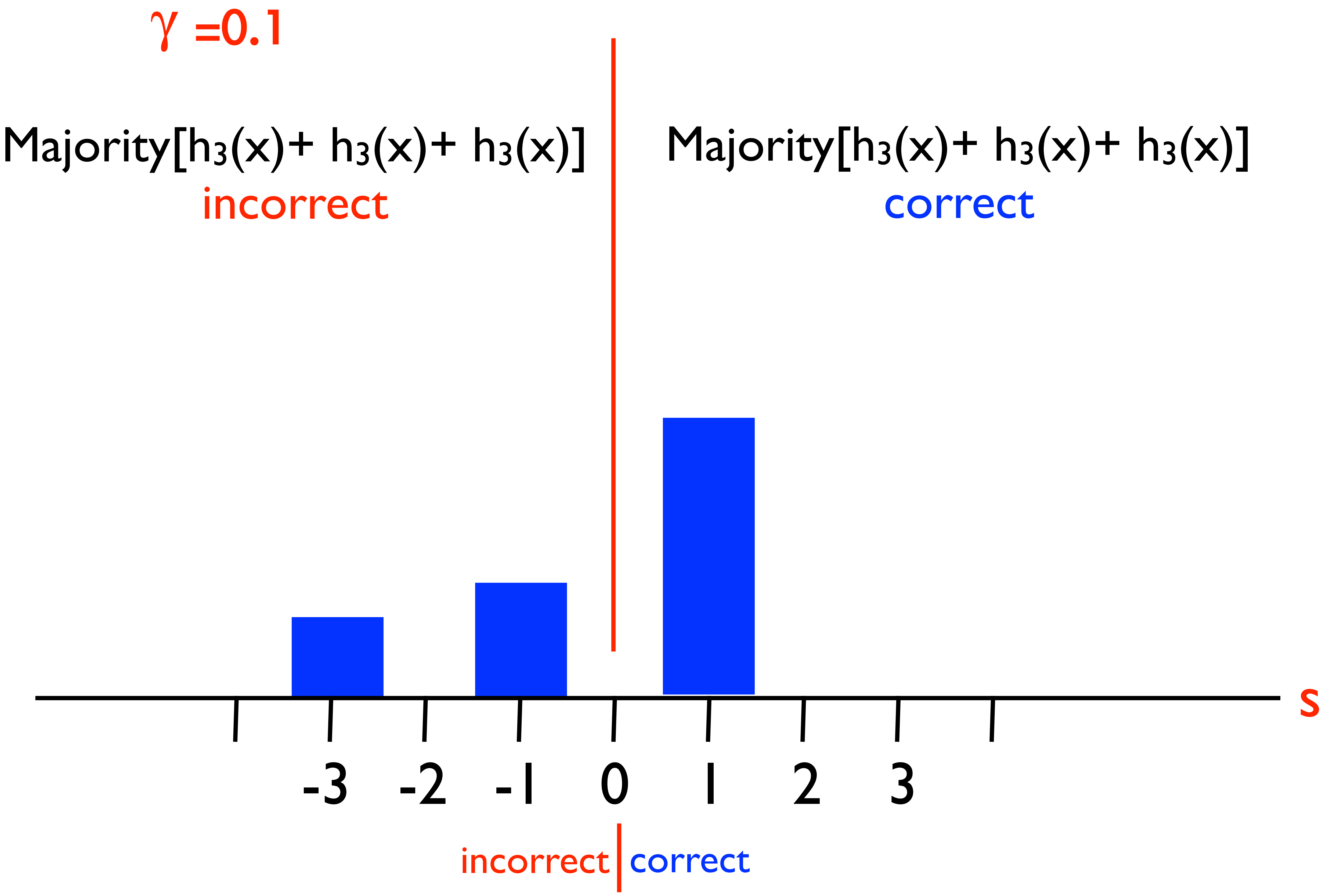
$\gamma = 0.1$

Weak learner chooses subset with weight $1/2 + \gamma$ which $h_3(x)$ classifies correctly



$\gamma = 0.1$

Weak learner chooses subset with weight $1/2 + \gamma$ which $h_3(x)$ classifies correctly



γ =0.1
Majority[h3(x)+ h3(x)+ h3(x)]
incorrect
Majority[h3(x)+ h3(x)+ h3(x)]
correct
s
-3
-2
-1
0
1
2
3
incorrect
correct

Weak Learner's min/max strategy

- AdOpt - Choose $1/2+\gamma$ from each bin to be correct.
- Equivalently: prediction of each base rule on each example is chosen independently at random

$$P(h_t(x)=y) = 1/2+\gamma$$

Potential

Total potential: Ψ(t, configuration) - μ-prob of the examples on which the final majority vote is incorrect given the configuration after iteration t is configuration and on the remaining steps the learner plays AdOpt.

Ψ(0, all at origin) = Initial potential.

Ψ(T, configuration) = Training error of final majority rule.

Boosting algorithm chooses weights so that the total potential does not increase.

Initial potential $\geq$ final training error.

Bin Potential: ϕ(t,s) - fraction of examples in bin s after iteration t on which the final majority rule will be incorrect assuming AdOpt play

$f(t,s)$ The configuration
The μ-prob of examples in bin s after iteration t

The potential of a configuration: $\Psi(f,t) = \sum_i f(t,s)\phi(t,s)$

ϕ(t,s) does not depend on the configuration

$$\phi(t,s) = \mathrm{Binom}\left(T-t, \frac{T-t-s}{2}, \frac{1}{2}+\gamma\right); \quad \mathrm{Binom}(n,k,p) \doteq \sum_{j=0}^{\lfloor k \rfloor} \binom{n}{j} p^j (1-p)^{n-j}$$

$$\phi(t-1,s) = \left(\frac{1}{2}-\gamma\right)\phi(t,s-1) + \left(\frac{1}{2}+\gamma\right)\phi(t,s+1)$$

$$\phi(T,s) = \begin{cases} 0 & s > 0 \\ 1 & s \le 0 \end{cases} \qquad \phi(0,0) = \mathrm{Binom}\left(T, \frac{T}{2}, \frac{1}{2}+\gamma\right)$$

Definitions

edge: $d(t,s) \doteq \mu\left(h_t(x) = y \middle| (x,y) \text{ in bin } s \text{ after iteration } t\right) - \left(\frac{1}{2} + \gamma\right)$

Example Weights: $w(t,s) \doteq \phi(t+1, s-1) - \phi(t+1, s+1)$

Theorem

If, for step t, $\sum_s d(t,s)w(t,s) \geq 0$ then $\Psi(t+1) \leq \Psi(t)$

Corollary

If ∀t [weighted error of $h_t(x)$] ≤ 1/2-γ

Then

Initial potential ≥ final training error.

Proof of Theorem

$$f(t+1,s) = f(t,s-1)\left(\frac{1}{2}+\gamma+d(t,s-1)\right)+f(t,s+1)\left(\frac{1}{2}-\gamma-d(t,s+1)\right)$$

$$\Psi(f,t+1) = \sum_s \left[f(t,s-1)\left(\frac{1}{2}+\gamma+d(t,s-1)\right)+f(t,s+1)\left(\frac{1}{2}-\gamma-d(t,s+1)\right)\right]\phi(t+1,s)$$

$$\Psi(f,t+1) = \sum_s \left[f(t,s)\left(\left(\frac{1}{2}+\gamma\right)\phi(t+1,s+1)+\left(\frac{1}{2}-\gamma\right)\phi(t+1,s-1)\right)-d(t,s)\left(\phi(t+1,s-1)-\phi(t+1,s+1)\right)\right]$$

$$\phi(t,s) = \left(\frac{1}{2}+\gamma\right)\phi(t+1,s+1)+\left(\frac{1}{2}-\gamma\right)\phi(t+1,s-1)$$

$$w(t,s) \doteq \phi(t+1,s-1)-\phi(t+1,s+1)$$

$$\Psi(f,t+1) = \sum_s \left[f(t,s)\phi(t,s)+d(t,s)w(t,s)\right] = \Psi(f,t)-\sum_s d(t,s)w(t,s)$$

If $\displaystyle\sum_s d(t,s)w(t,s) \geq 0$ then $\Psi(f,t+1) \leq \Psi(f,t)$

Theorem about BBM

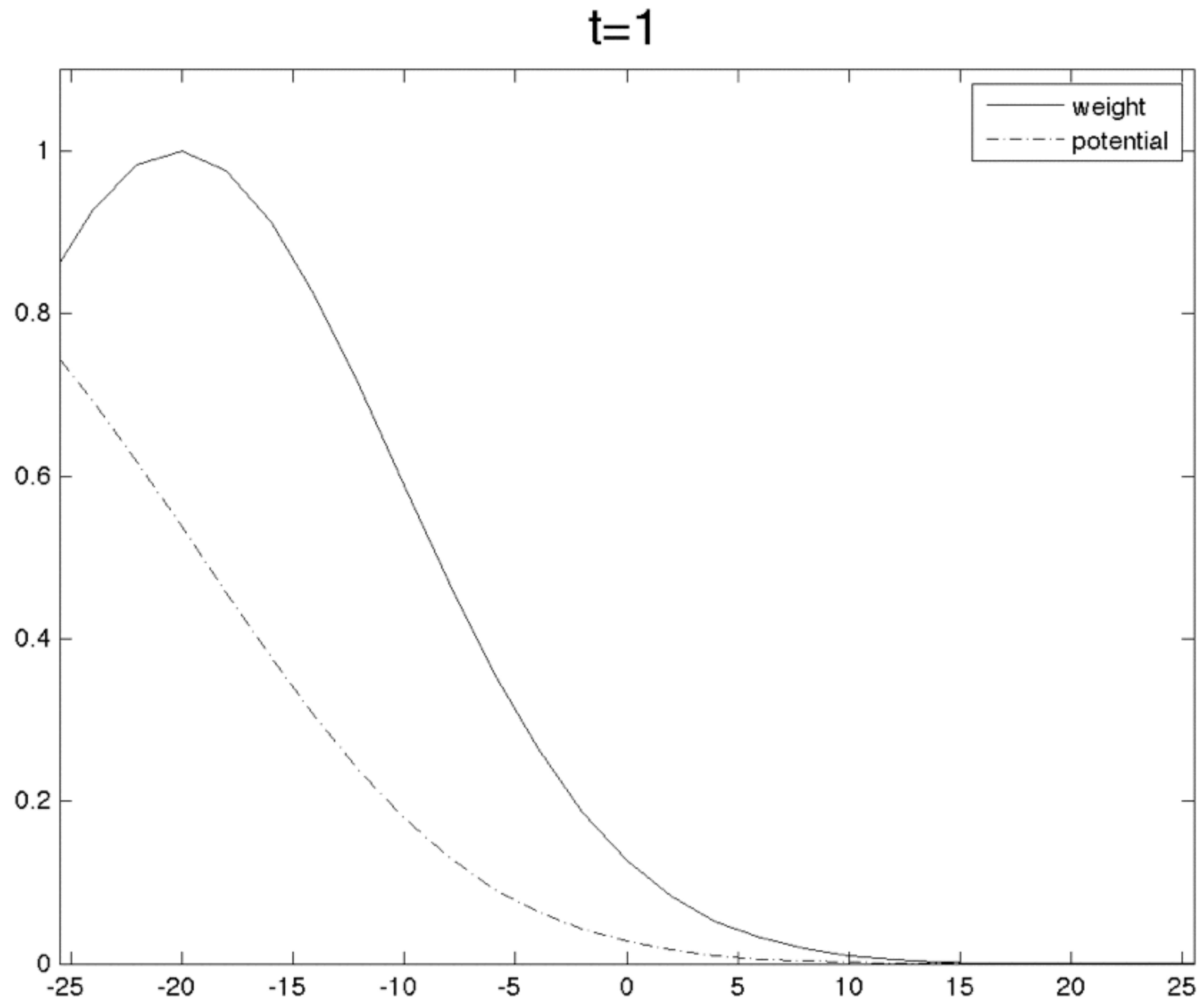
setting the boosting weights at iteration t to be

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

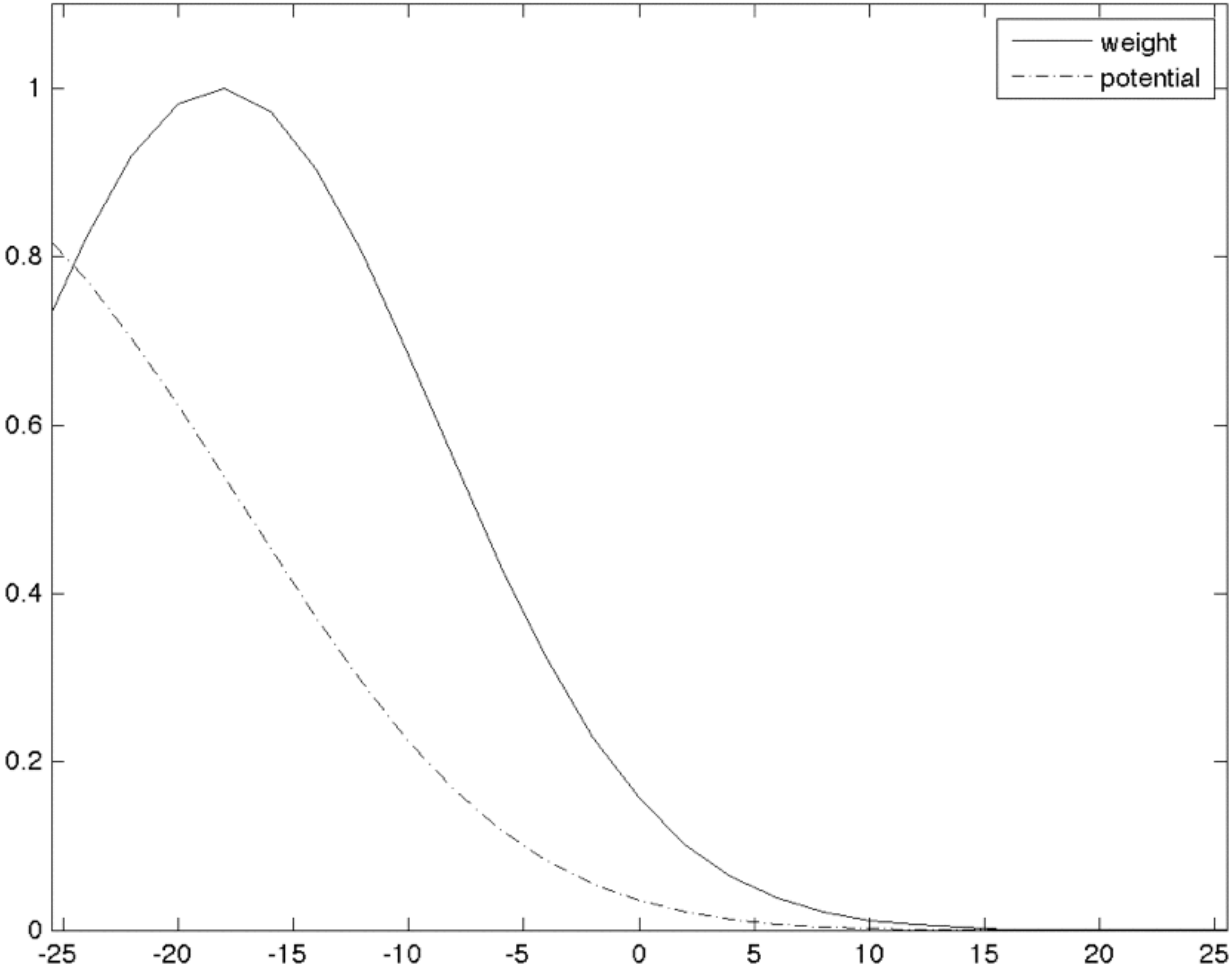
guarantees

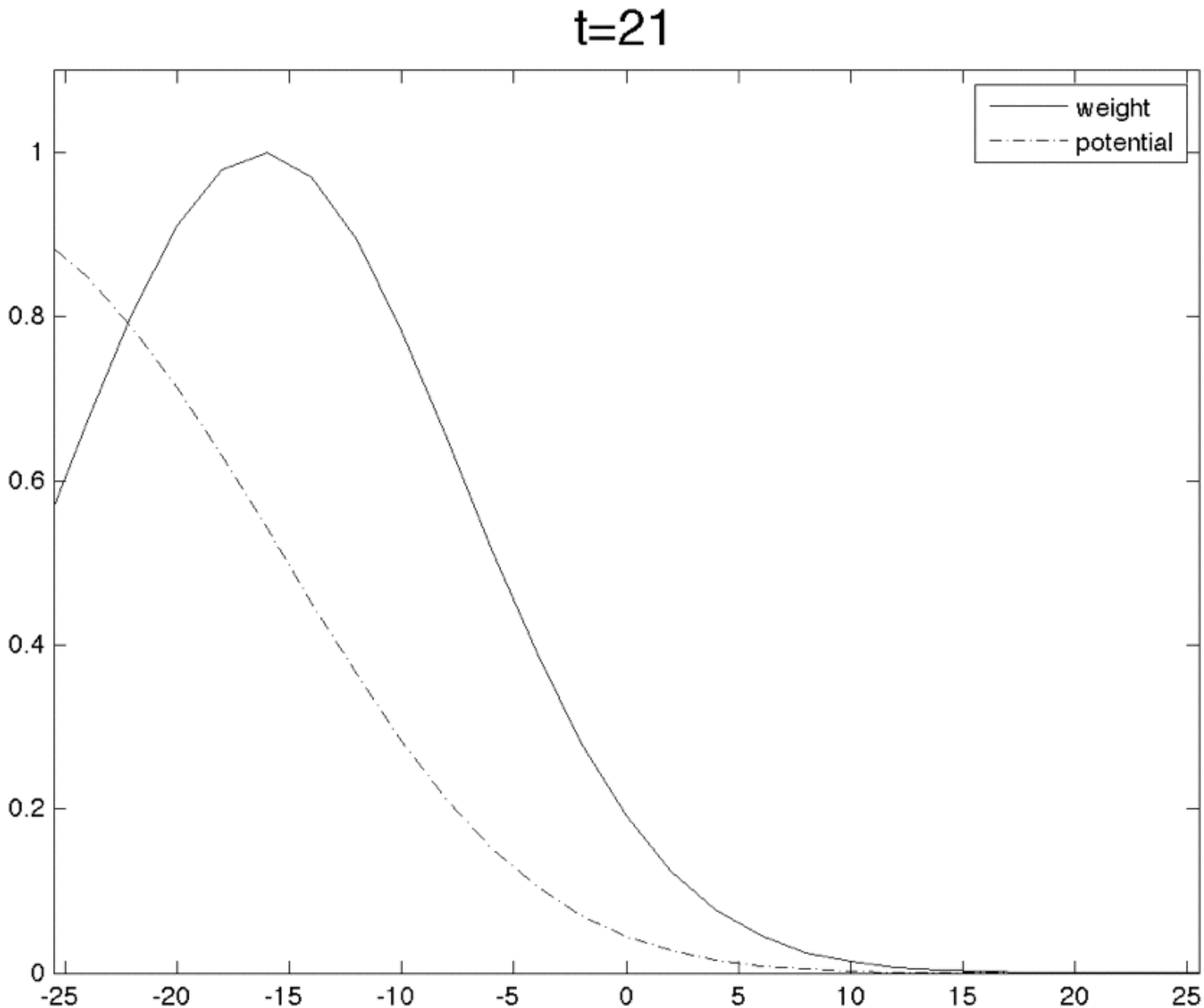
$$\text{Initial potential} = \Psi(0) \geq \Psi(1) \geq \cdots \geq \Psi(T) = \text{ training error of sign}\left(\sum_{t=1}^{T} h_t(x)\right)$$

$$\varepsilon = \Psi(0) = \psi(0,0) = \text{Binom}\left(T, \frac{T}{2}, \frac{1}{2}+\gamma\right)$$

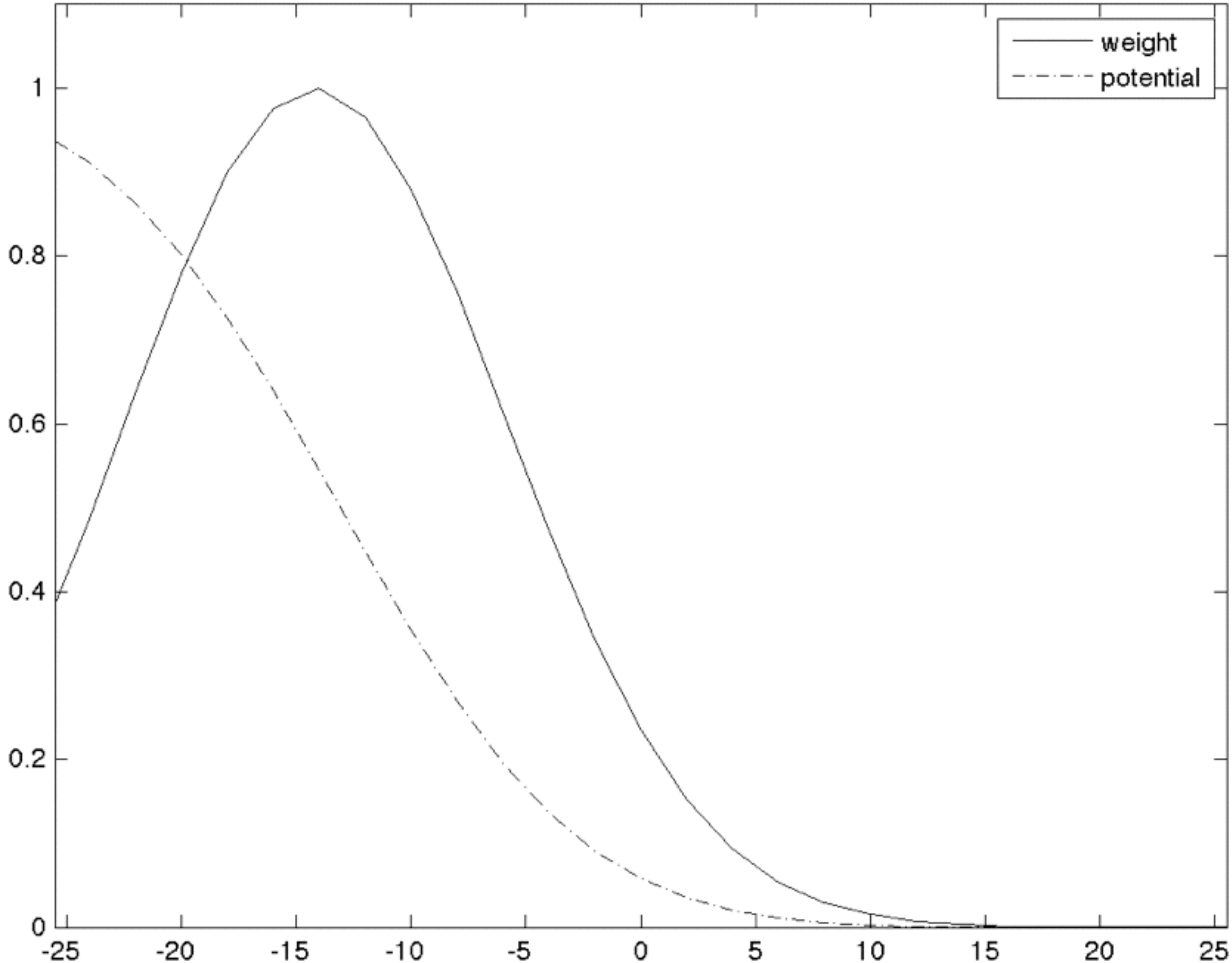

t=1
weight
potential
1
0.8
0.6
0.4
0.2
0
-25
-20
-15
-10
-5
0
5
10
15
20
25

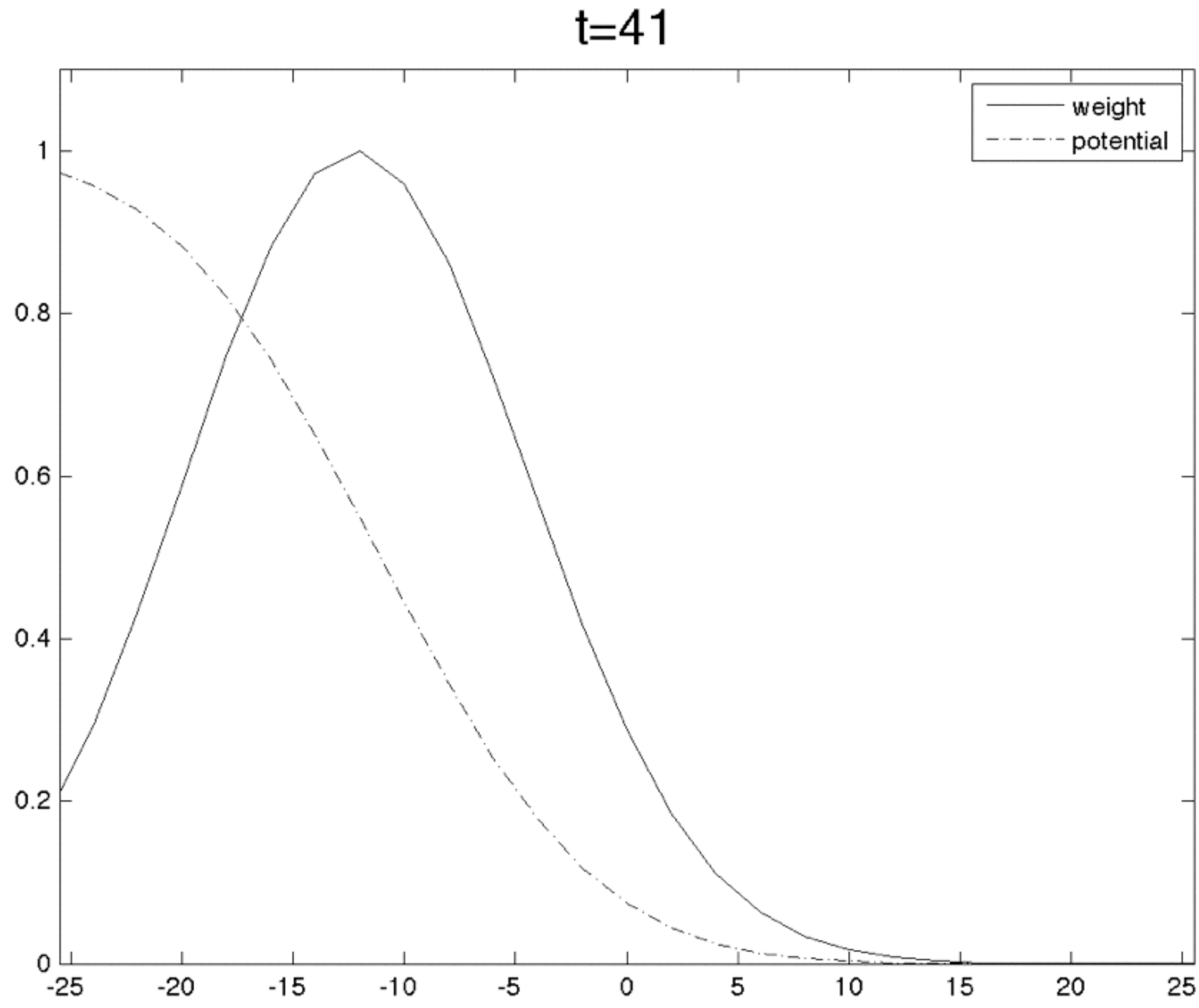
t=11

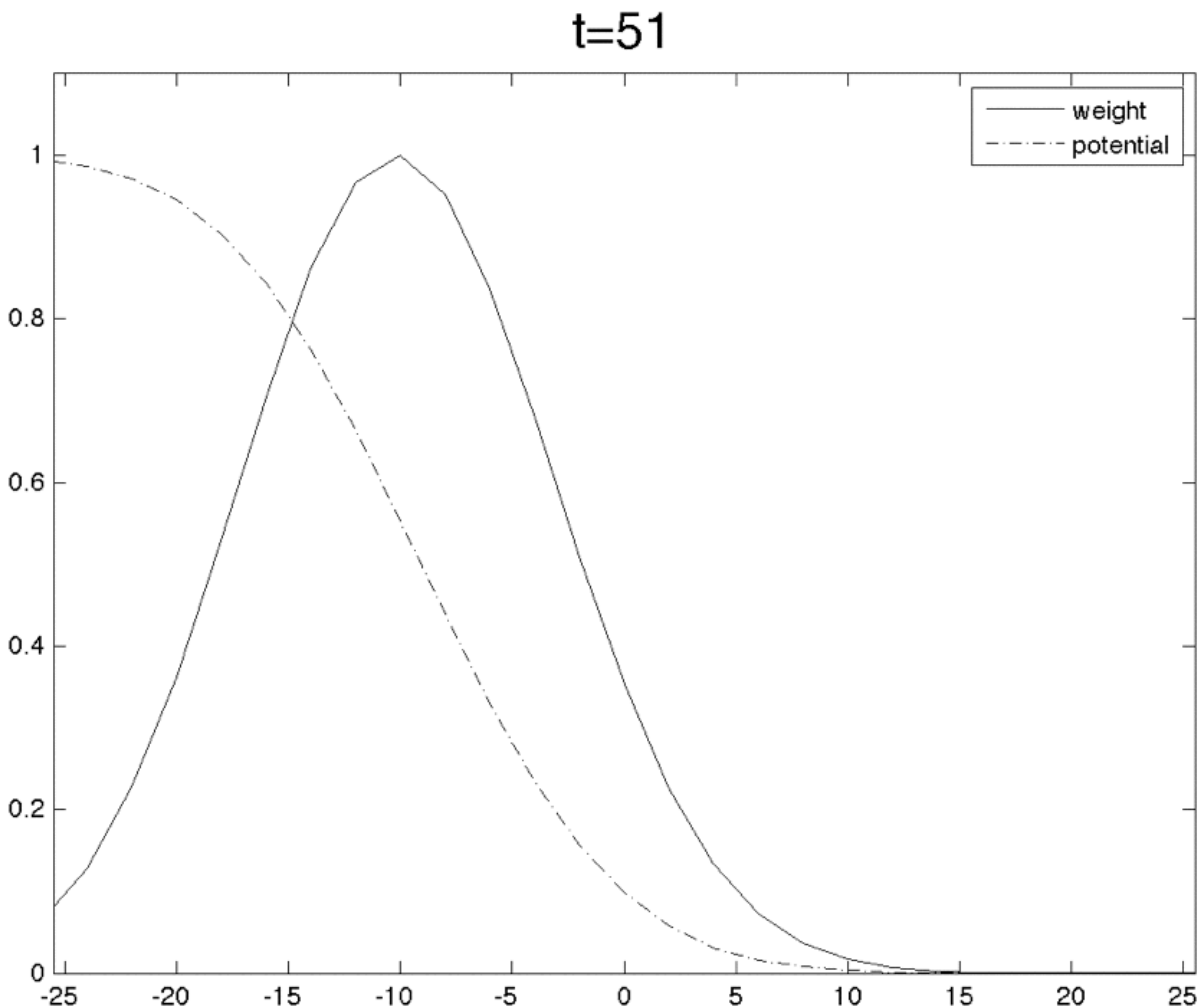


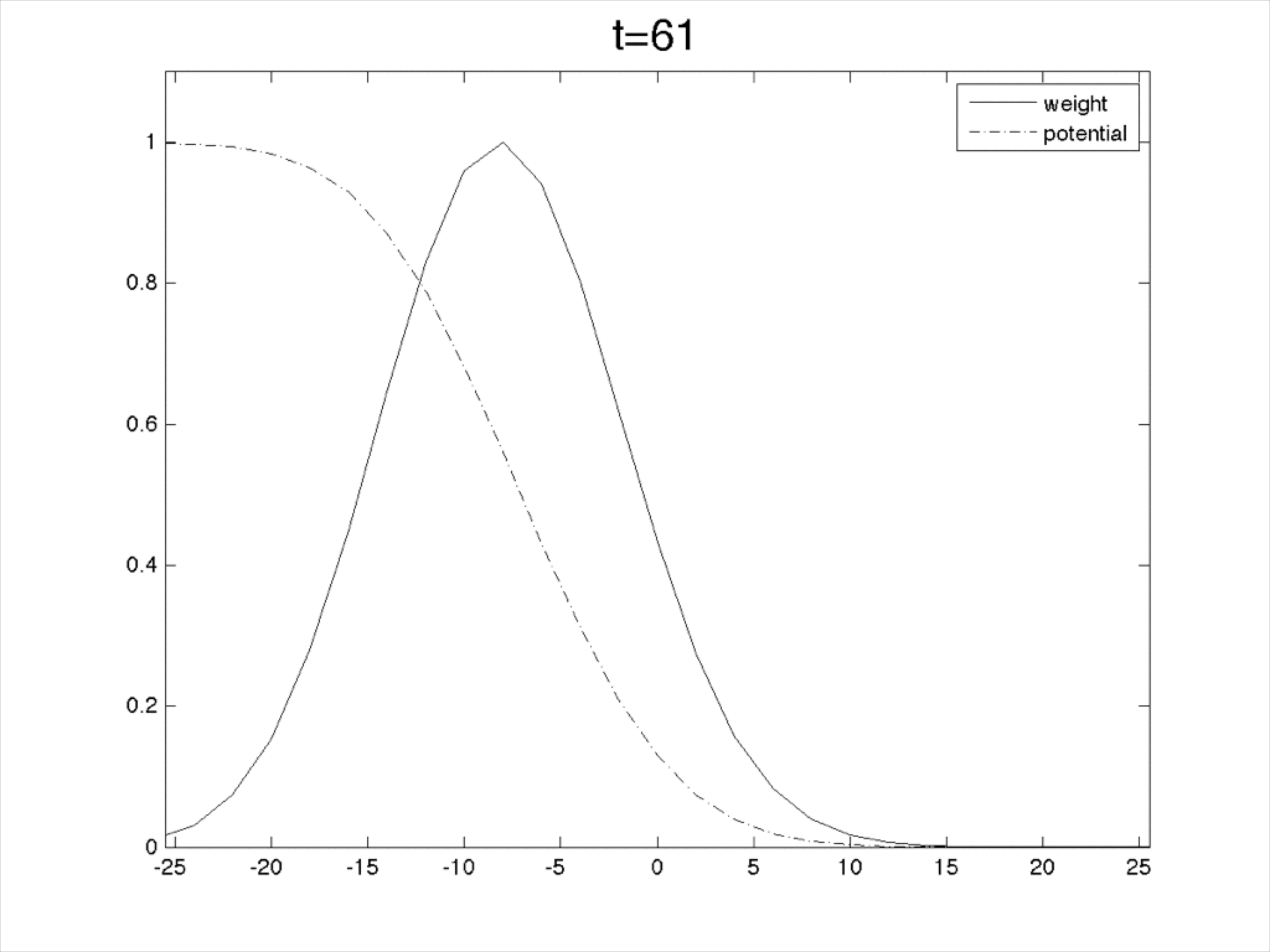

t=21
weight
potential
0
0.2
0.4
0.6
0.8
1
-25
-20
-15
-10
-5
0
5
10
15
20
25

t=31

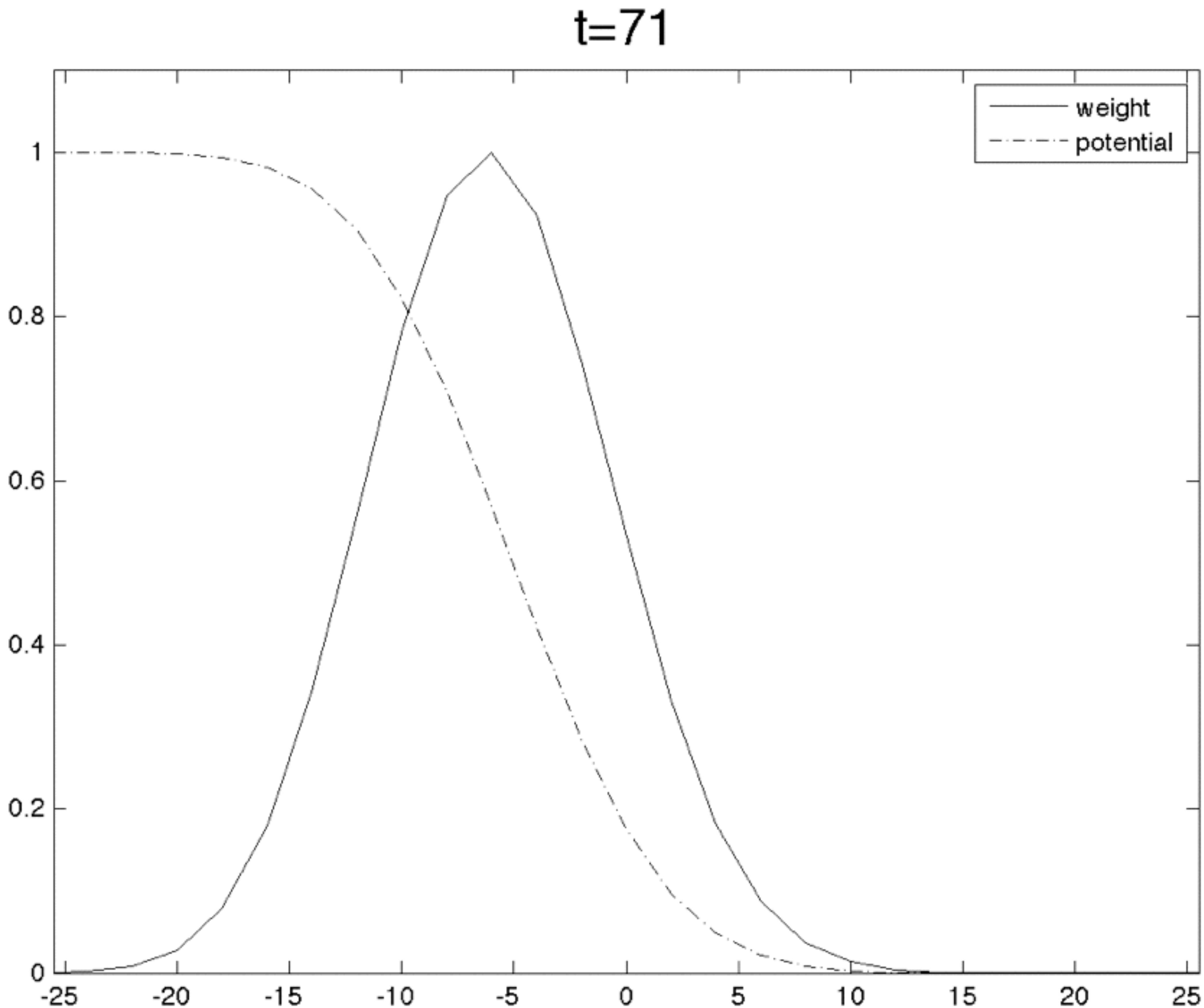


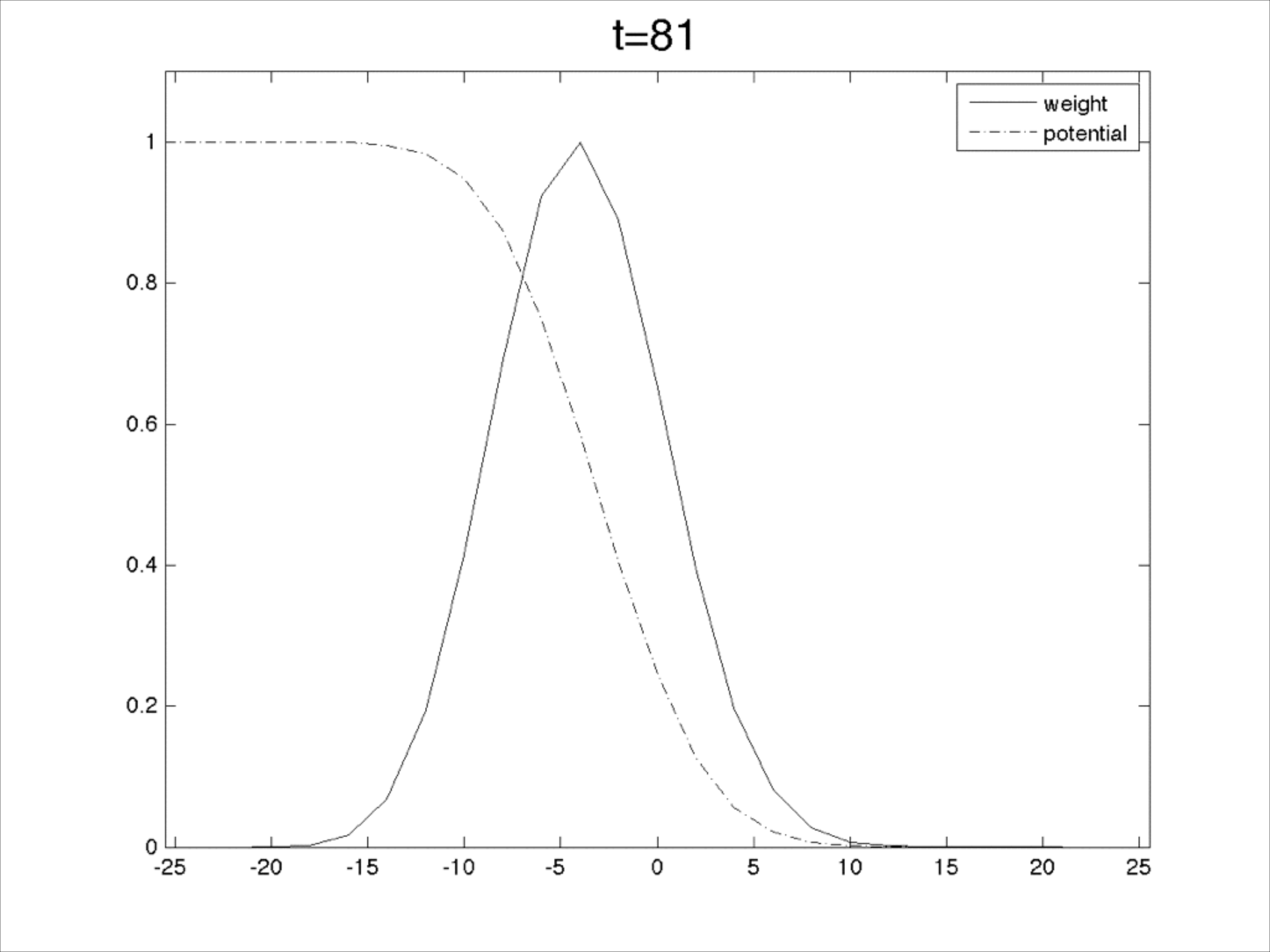

t=41
weight
potential
0
0.2
0.4
0.6
0.8
1
-25
-20
-15
-10
-5
0
5
10
15
20
25

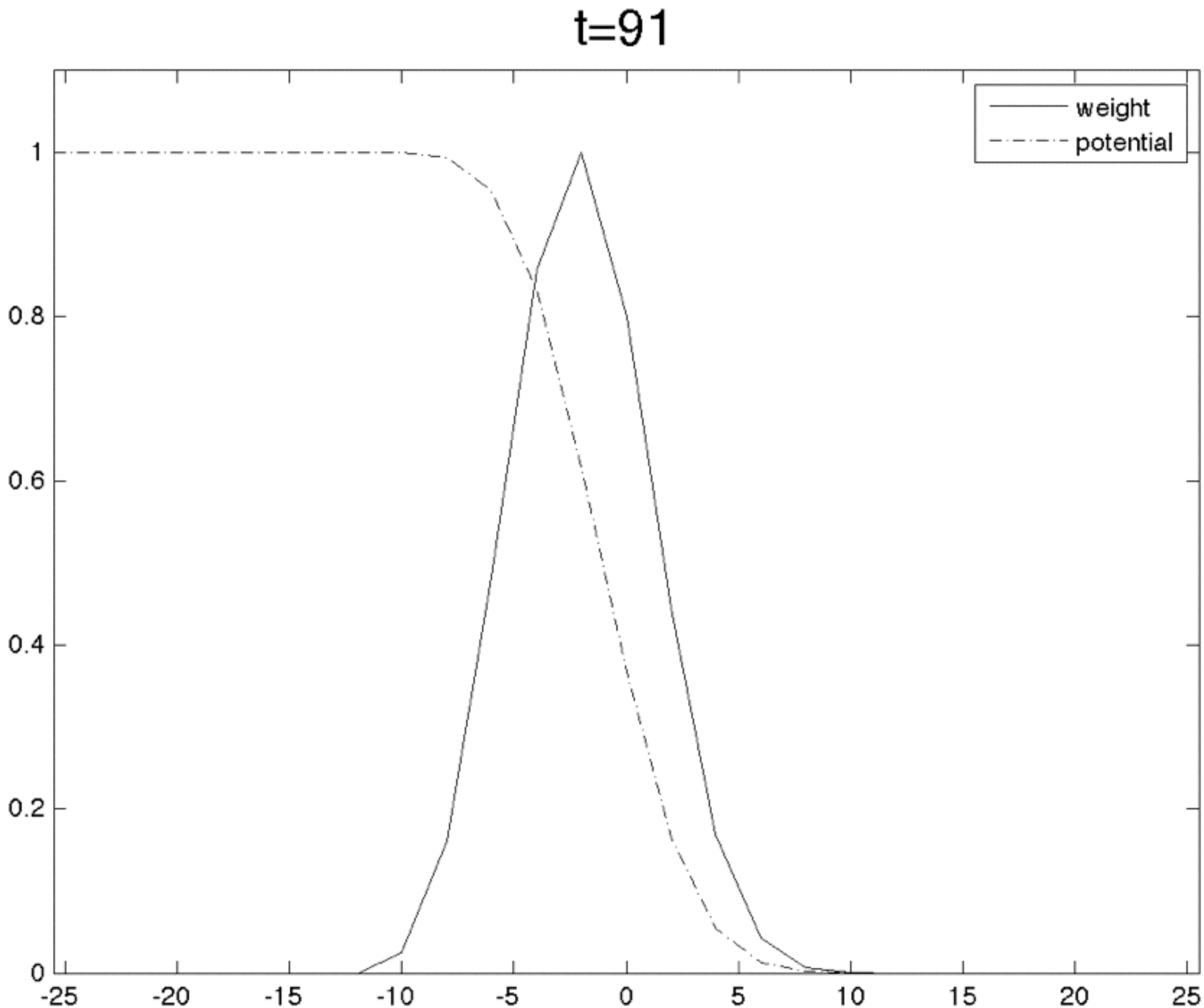

t=51
weight
potential
1
0.8
0.6
0.4
0.2
0
-25
-20
-15
-10
-5
0
5
10
15
20
25

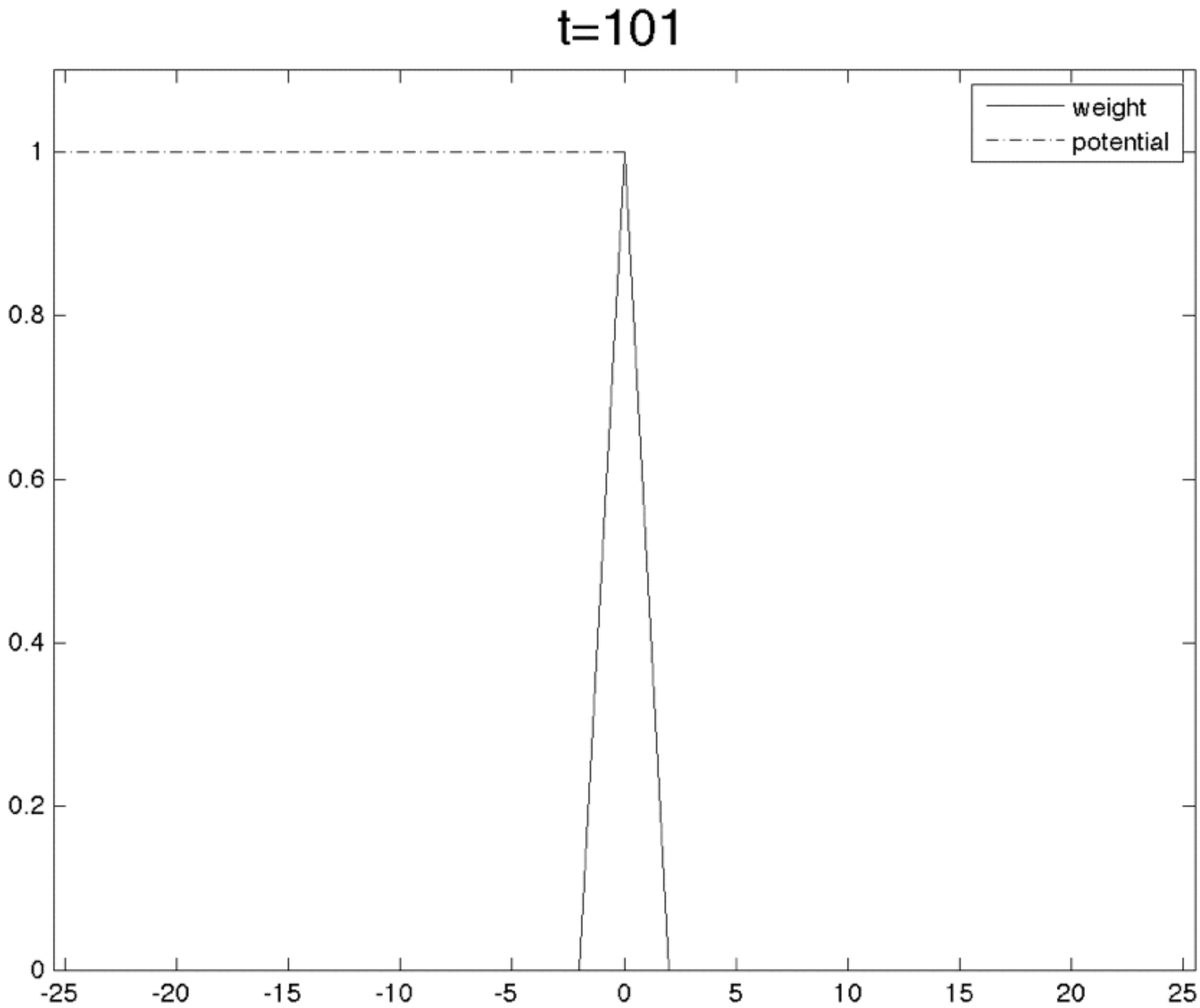


t=61
weight
potential
1
0.8
0.6
0.4
0.2
0
-25
-20
-15
-10
-5
0
5
10
15
20
25

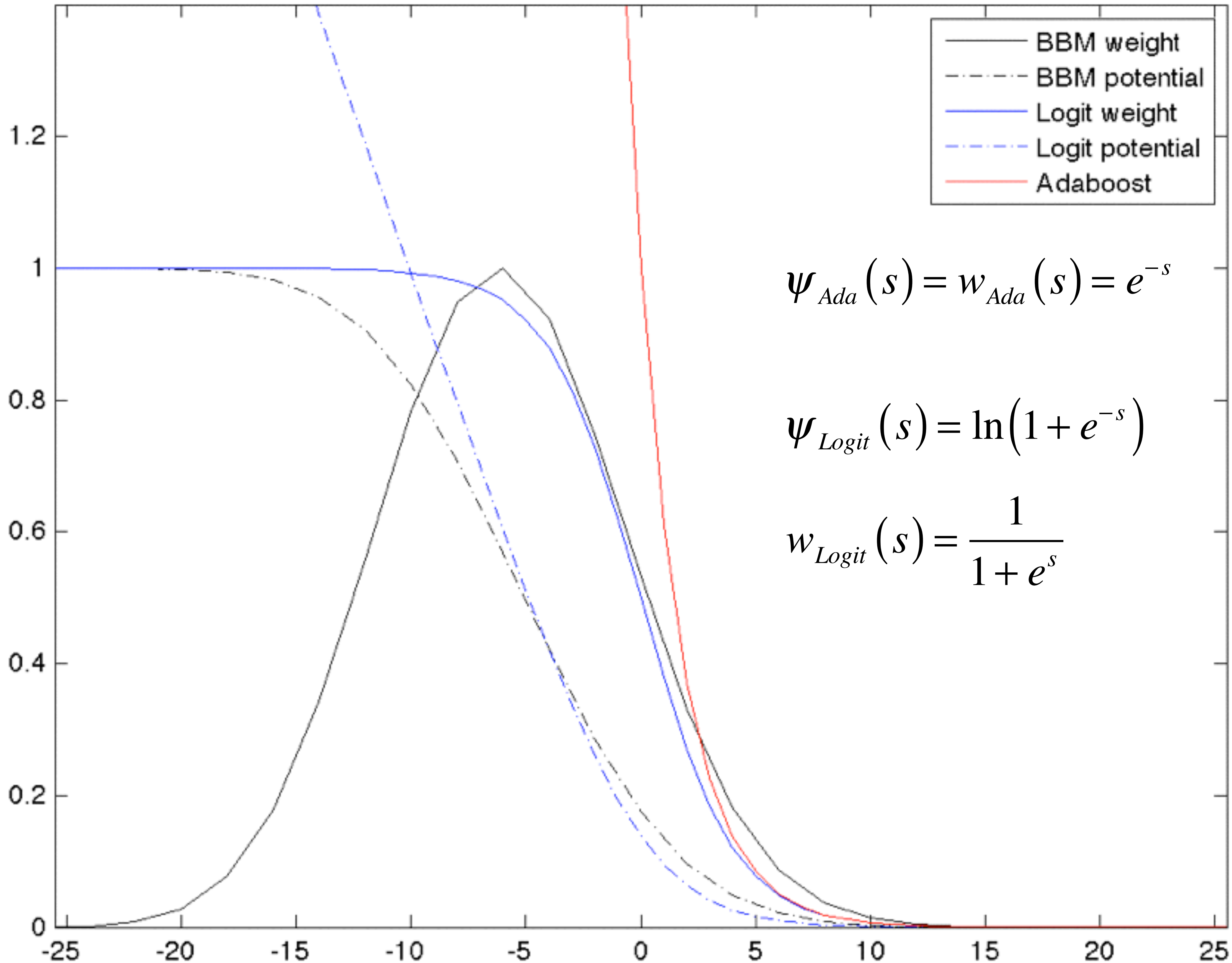

t=71
weight
potential
1
0.8
0.6
0.4
0.2
0
-25
-20
-15
-10
-5
0
5
10
15
20
25


t=81
weight
potential
1
0.8
0.6
0.4
0.2
0
-25
-20
-15
-10
-5
0
5
10
15
20
25


t=91
weight
potential
1
0.8
0.6
0.4
0.2
0
-25
-20
-15
-10
-5
0
5
10
15
20
25


t=101
weight
potential
1
0.8
0.6
0.4
0.2
0
-25
-20
-15
-10
-5
0
5
10
15
20
25

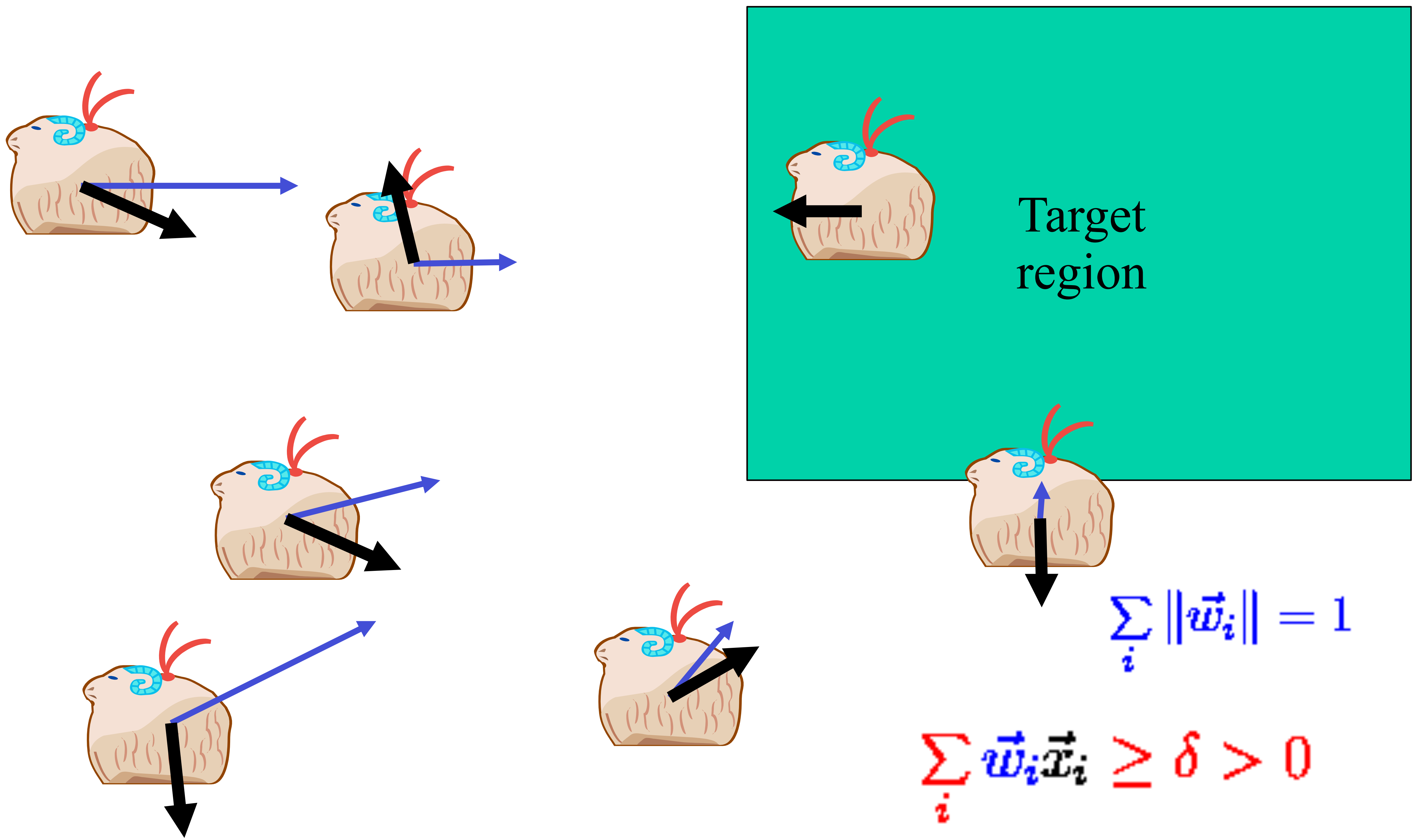
BBM/Logitboost/Adaboost



High level summary

- The worst case adversary splits each bin into:
  $1/2-\gamma$ incorrect / $1/2+\gamma$ correct
- Alternative interpretation: Random walk with IID steps.
- Algorithm is derived as optimal response to this simple worst-case adversary.

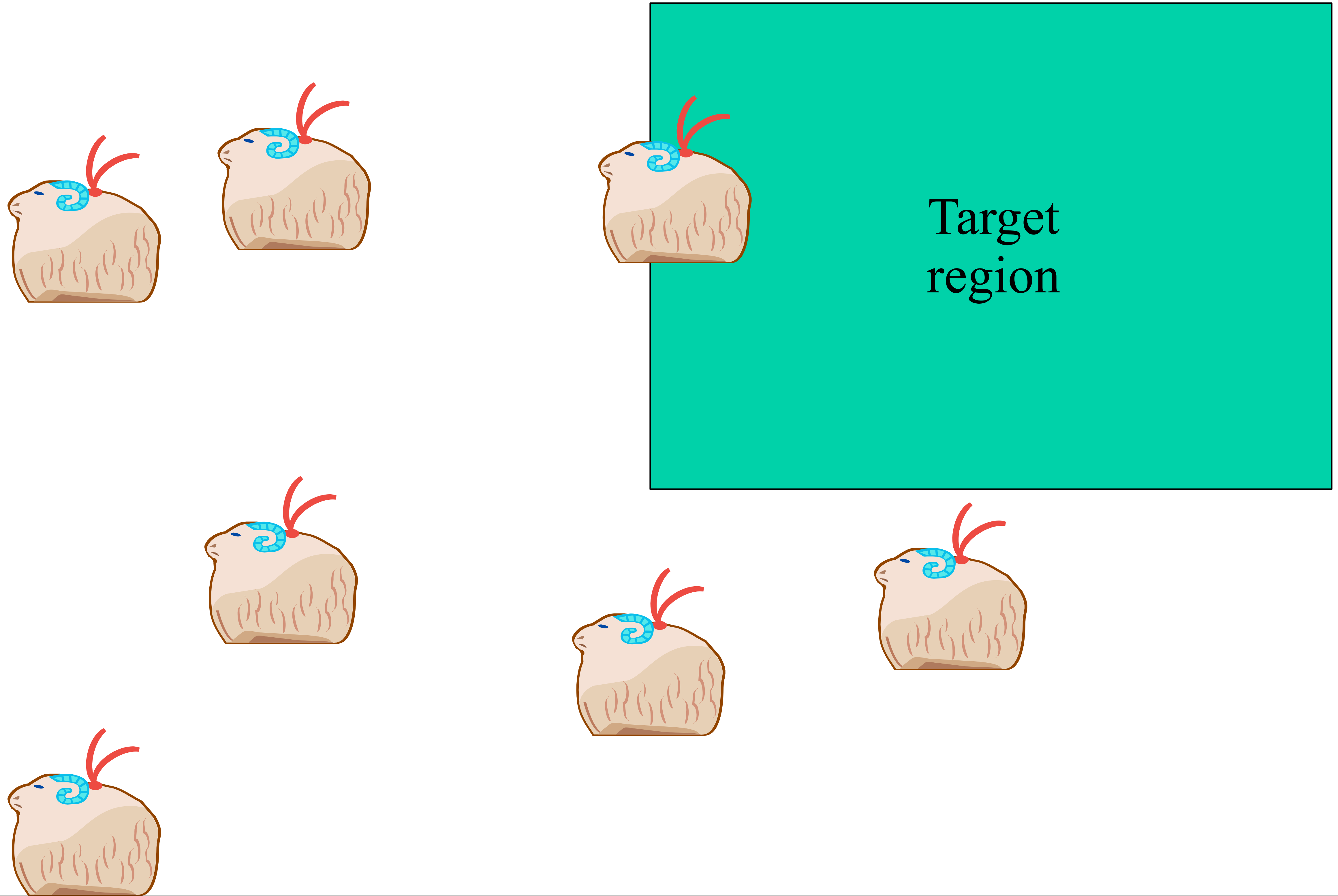
Drifting games (in 2d)



$$\sum_i \|\vec{w}_i\| = 1$$

$$\sum_i \vec{w}_i \vec{x}_i \geq \delta > 0$$

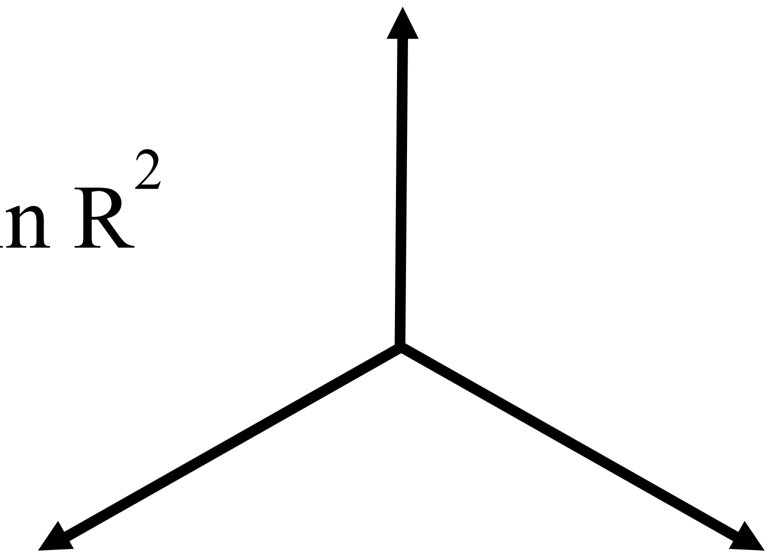
Drifting games (in 2d)



The allowable steps

B = the set of all allowable step
Normal B = minimal set that spans the space. (~basis)
Regular B = a symmetric regular set. (~orthonormal basis)

Regular step set in R^2



The min/max solution

[Schapire99]

- A potential defined by a min/max recursion

$$\phi_T(\mathbf{s}) = L(\mathbf{s})$$

$$\phi_{t-1}(\mathbf{s}) = \min_{\mathbf{w}\in\mathbb{R}^d} \sup_{\mathbf{z}\in B} (\phi_t(\mathbf{s}+\mathbf{z}) + \mathbf{w}\cdot\mathbf{z} - \delta\,\|\mathbf{w}\|)$$

$\phi_0(0) =$ the value of the game

- Shepherd's strategy

$$\mathbf{w}_i^t = \arg\min_{\mathbf{w}} \sup_{\mathbf{z}\in B} (\phi_t(\mathbf{s}+\mathbf{z}) + \mathbf{w}\cdot\mathbf{z} - \delta\,\|\mathbf{w}\|)$$

The solution simplifies when $\delta \to 0$

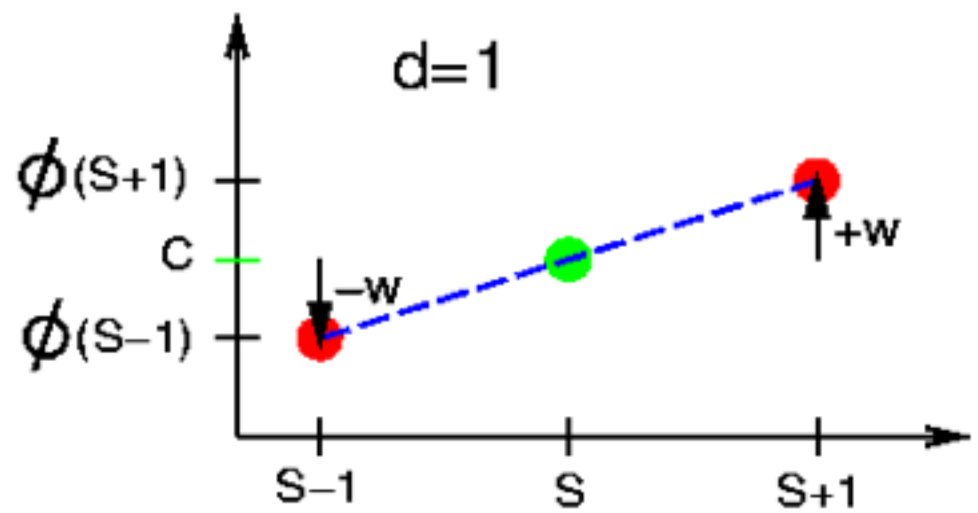
If B is normal, and δ is sufficiently small then $\exists \mathbf{w}^*$ such that

$$\phi_{t-1}(\mathbf{s}) = \phi_t(\mathbf{s}+\mathbf{z}) + \mathbf{w}^* \cdot \mathbf{z} - \delta \|\mathbf{w}^*\|$$

for all $\mathbf{z} \in B$ (and all $t = 1, 2 \ldots$, $\mathbf{s} \in \mathbb{R}^d$)

Implies that: $\mathbf{w}^*$ is the "local slope" at $\phi_t(\mathbf{s})$, i.e.

$$\phi_t(\mathbf{s}+\mathbf{z}_i) = C + \mathbf{w}^* \mathbf{z}_i \; ; \quad C \doteq \frac{\Sigma_{j=0}^{d} \phi_t(\mathbf{s}+\mathbf{z}_j)}{d+1}$$



and that

$$\phi_{t-1}(\mathbf{s}) = C - \delta \|\mathbf{w}^*\|$$

Increasing the number of steps

- Consider T steps in a unit time
- Drift δ should scale like $1/T$
- Step size $O(1/T)$ gives game to shepherd
- Step size $O\left(1/\sqrt{T}\right)$ keeps game balanced

The solution when $T \to \infty$

The local slope becomes the gradient

$$\mathbf{w}^* = \nabla \phi_\tau(\mathbf{s})$$

The recursion becomes a PDE

$$\begin{aligned} \frac{\partial \phi_\tau(\mathbf{s})}{\partial \tau} &= -\frac{1}{2} \sum_{k=1}^{d} \frac{\partial^2 \phi_\tau(\mathbf{s})}{\partial^2 s_k} + \delta \left\| \mathbf{w}^* \right\| \\ &= -\frac{1}{2} \Delta \phi_\tau(\mathbf{s}) + \delta \left\| \nabla \phi_\tau(\mathbf{s}) \right\| \end{aligned}$$

Same PDE describes time development of
Brownian motion with drift

Plan of talk

- Label noise and convex loss functions.
- Boost by Majority and drifting games.
- Boosting in continuous time.
- RobustBoost
- Experimental results.

Why is BBM not practical?

- BBM needs to know ε, γ before starting.

$$T = \frac{1}{\gamma^2} \ln \frac{1}{\varepsilon}$$

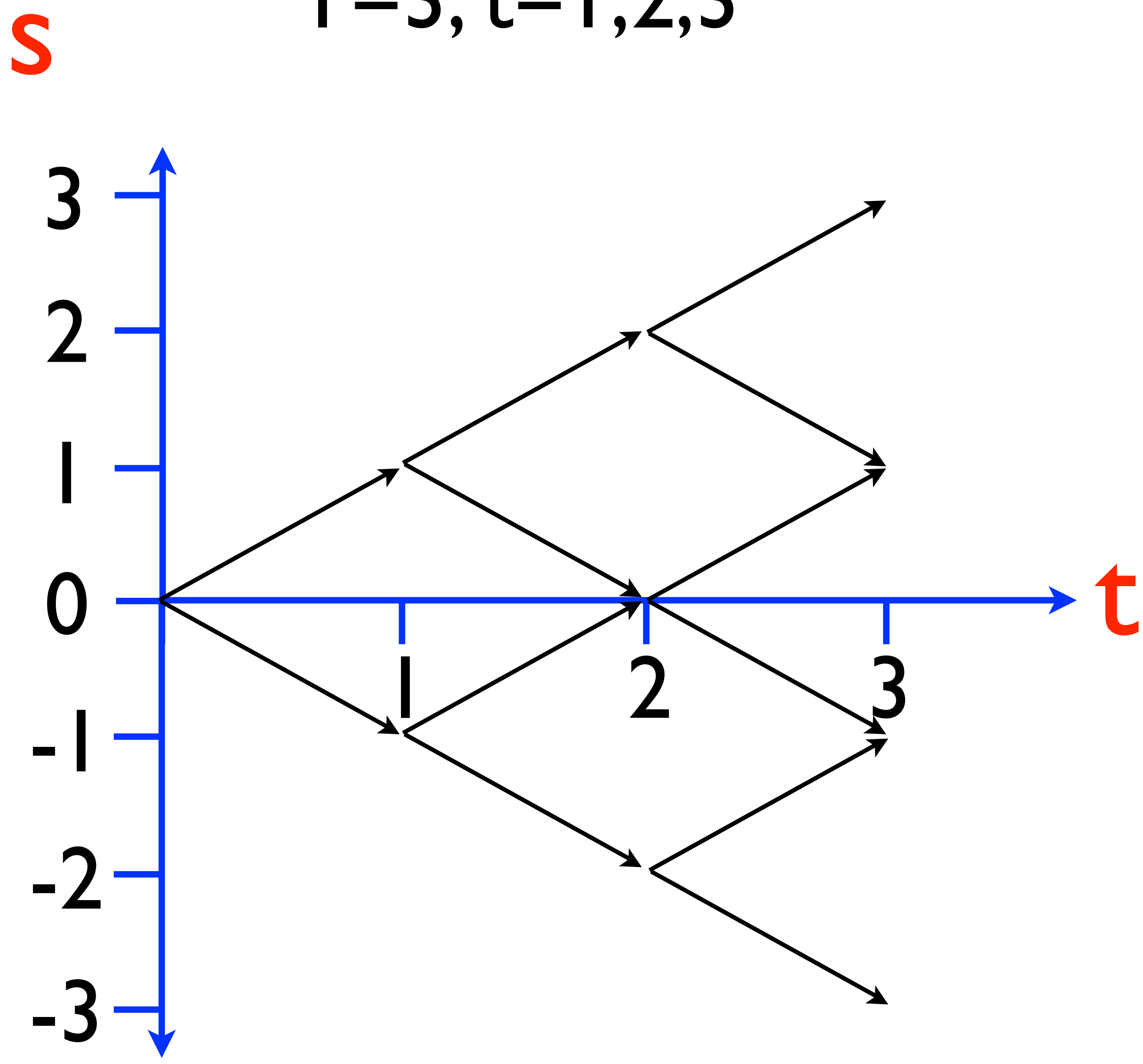
- Adaboost = adaptive boosting,Adapts to the sequence , $\gamma_1, \gamma_2, \gamma_3, \ldots$.
  - No need to set parameters in advance.
  - generates a weighted majority rule.
  - Decide when to stop using cross-validation.
- How can we make BBM adaptive?

Letting time step decrease to zero.

- Number of iterations required by BBM: $T = \frac{1}{\gamma^2} \ln \frac{1}{\varepsilon}$
- Keep ε fixed and let $\gamma \to 0, T \to \infty$
- The same weak rule is added many times, until it's advantage falls below γ .
  - yields adaptive boosting and a weighted majority rule.
- In the limit, adversary uses random walk in continuous time = Brownian Motion.
- Instead of t=1,2,...,T use t=1/T,2/T,...,1

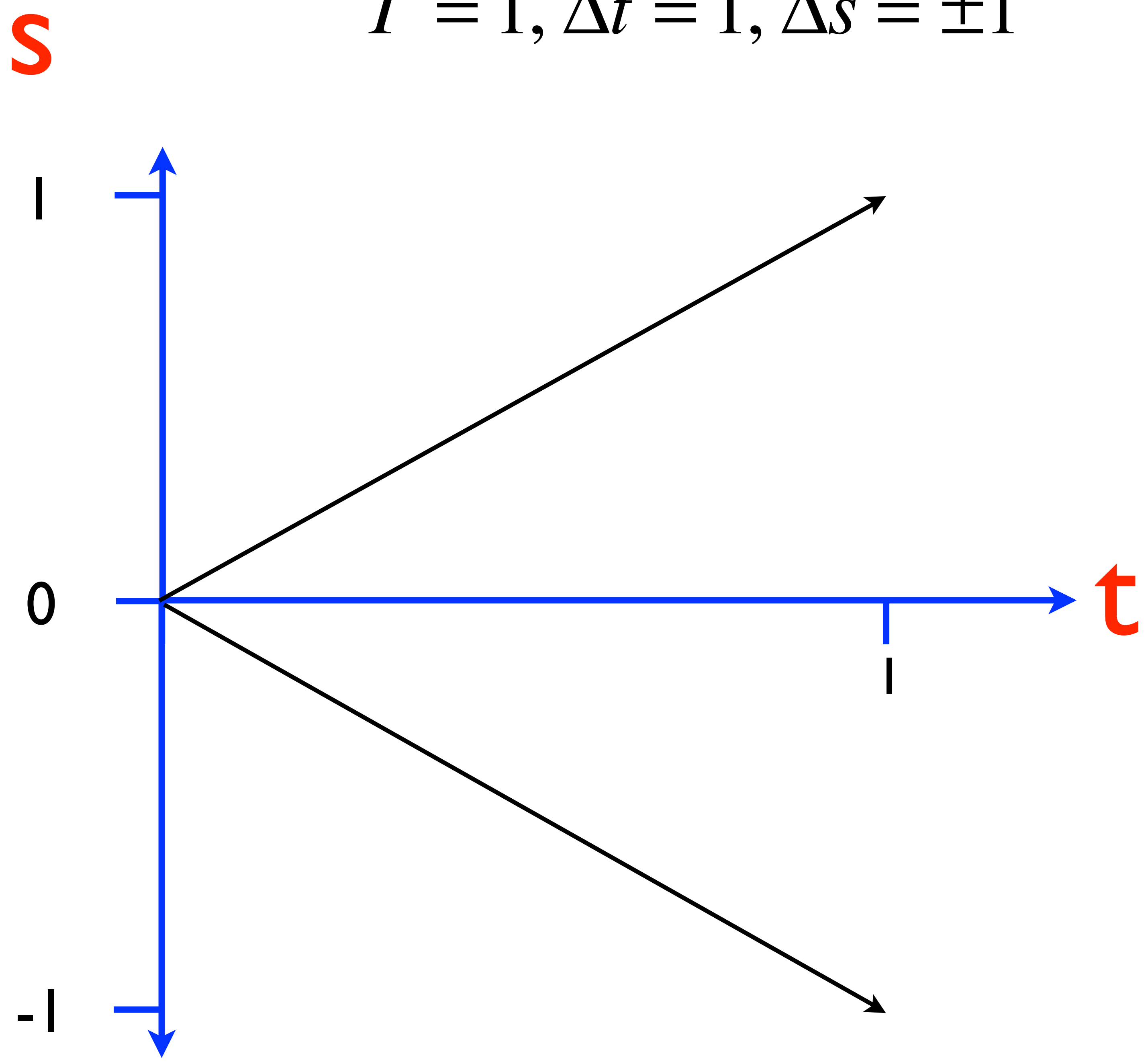
The game lattice

T=3, t=1,2,3



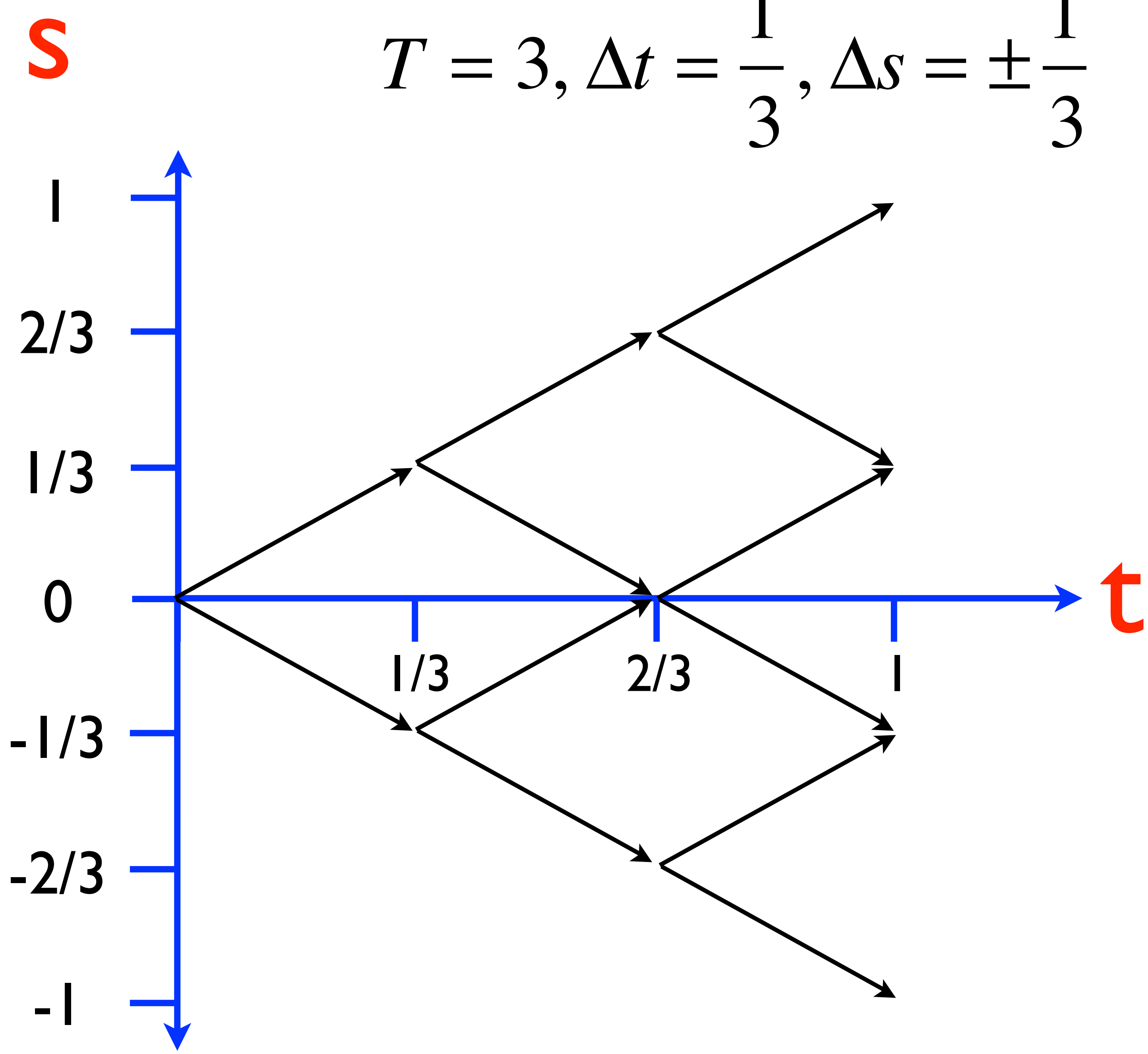
Using step $\Delta s = \pm \frac{1}{T}$

$$T = 1, \Delta t = 1, \Delta s = \pm 1$$

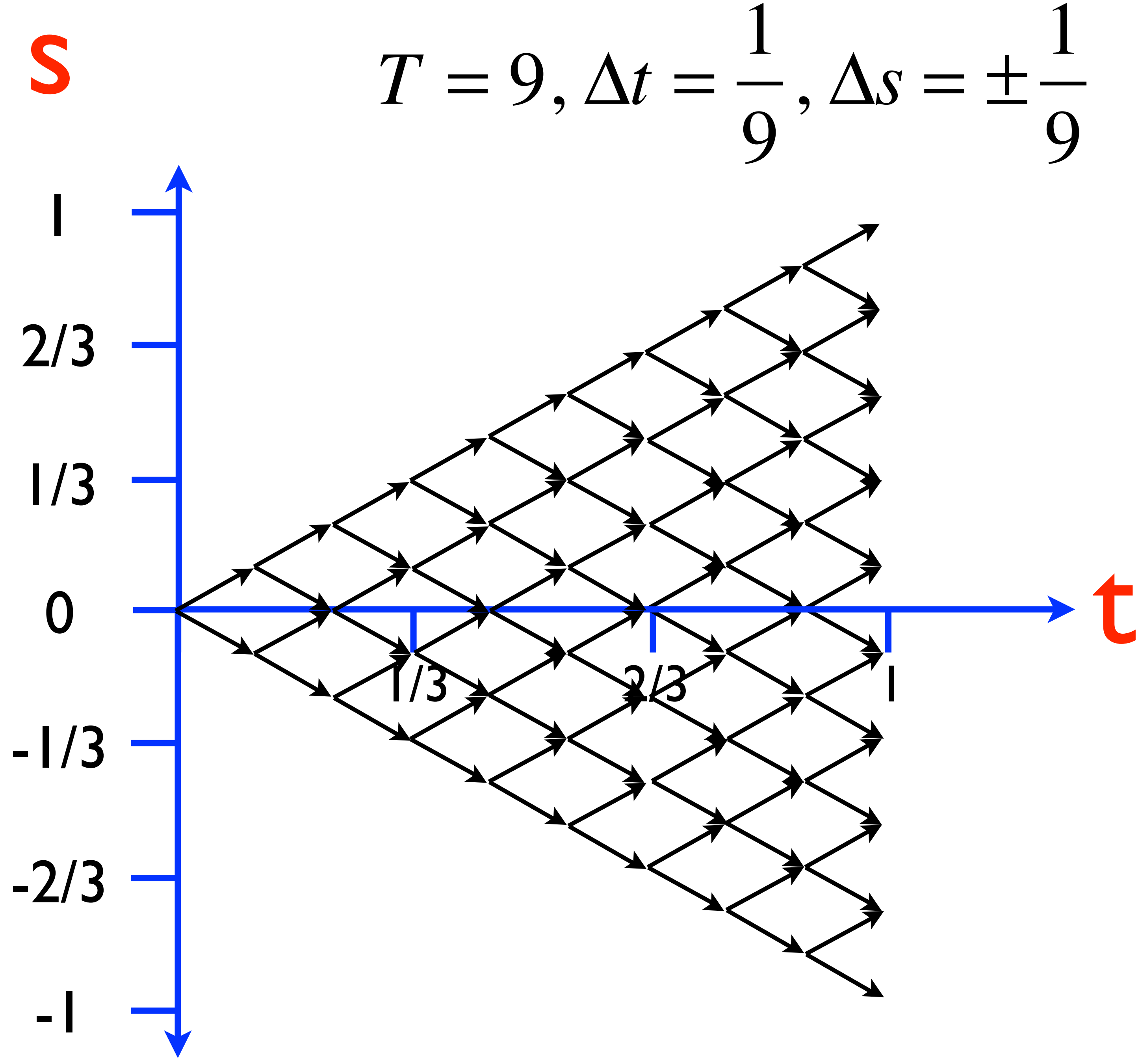


Using step $\Delta s = \pm\frac{1}{T}$

$$T = 3, \Delta t = \frac{1}{3}, \Delta s = \pm\frac{1}{3}$$



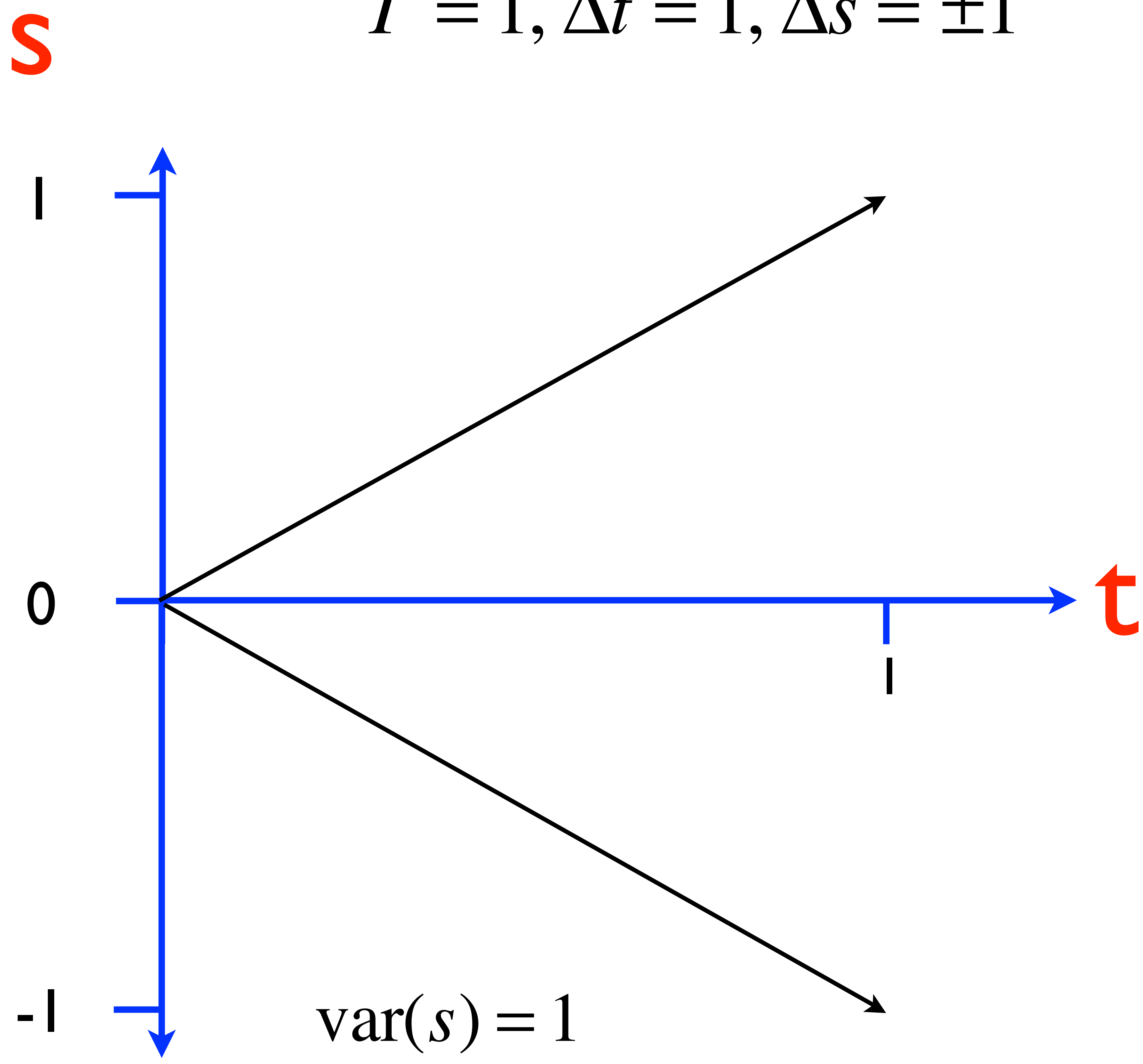
Using step $\Delta s = \pm\frac{1}{T}$



Looks fine but $\mathrm{var}(s) = T\frac{1}{T^2} = \frac{1}{T} \underset{T\to\infty}{\to} 0$

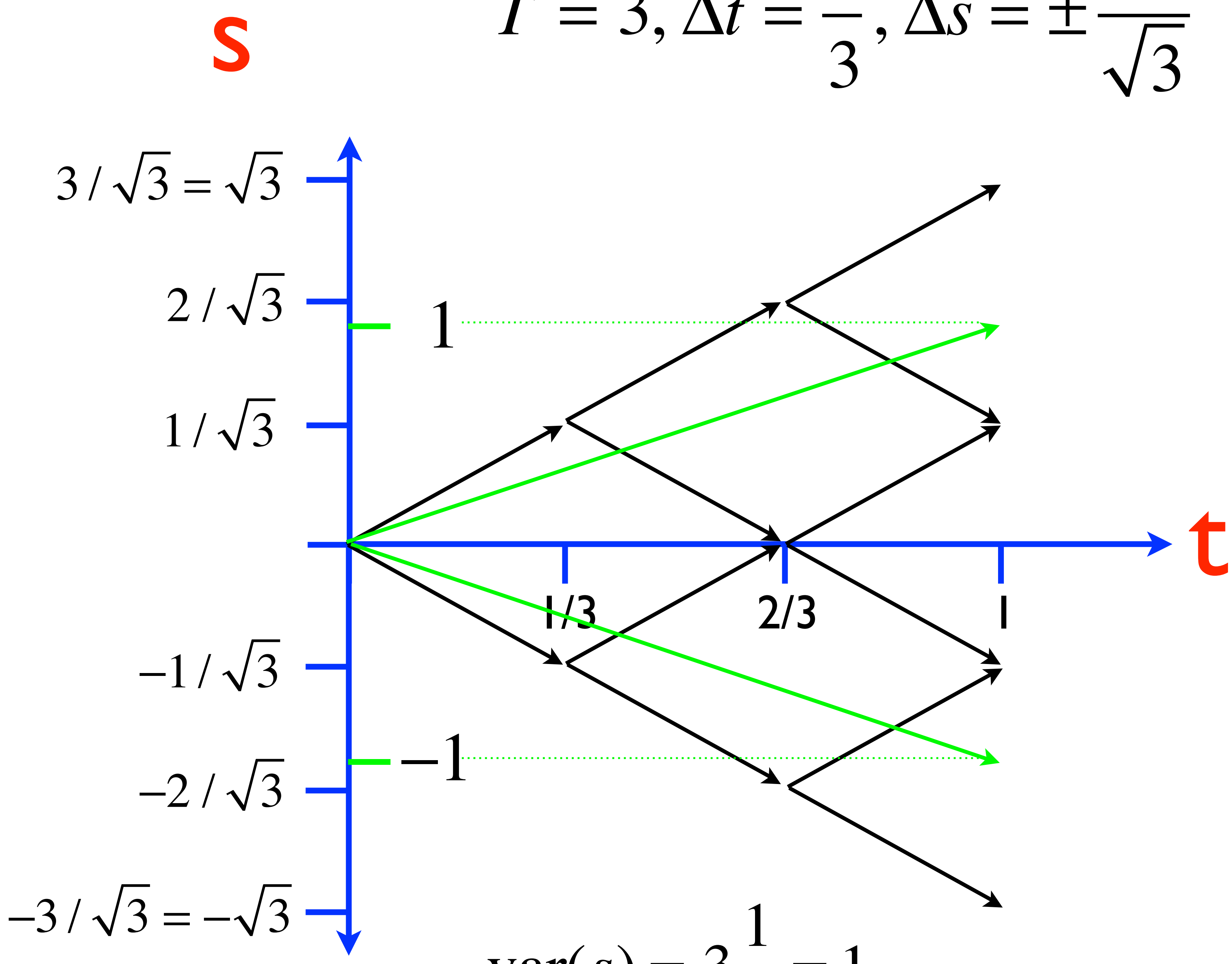
Using step $\Delta s = \pm \frac{1}{\sqrt{T}}$

$$T = 1, \Delta t = 1, \Delta s = \pm 1$$



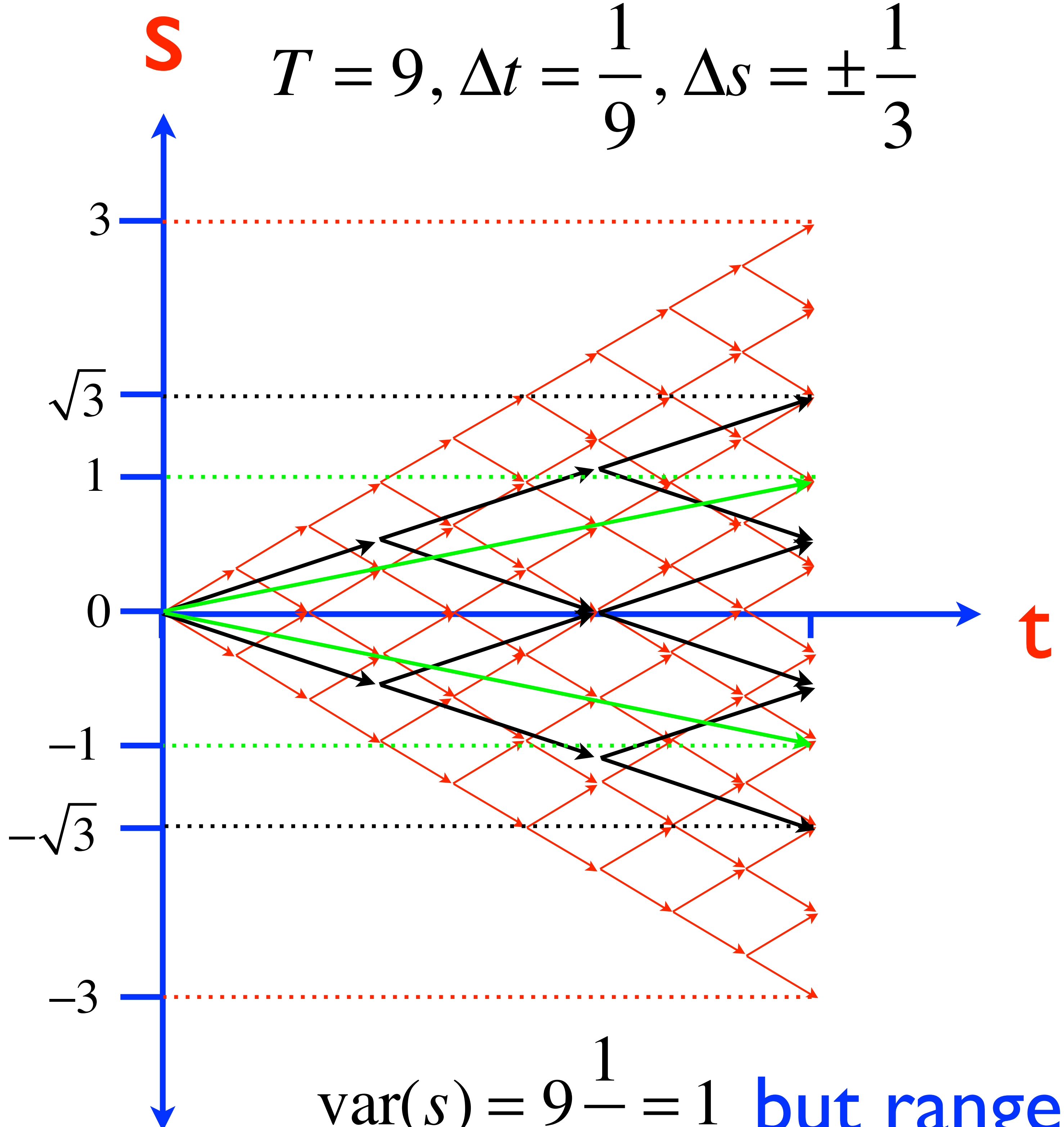
Using step $\Delta s = \pm \frac{1}{\sqrt{T}}$

$$T = 3, \Delta t = \frac{1}{3}, \Delta s = \pm \frac{1}{\sqrt{3}}$$



$$\mathrm{var}(s) = 3\frac{1}{3} = 1$$

Using step $\Delta s = \pm \frac{1}{\sqrt{T}}$



$\mathrm{var}(s) = 9\frac{1}{9} = 1$ but range(s) → ∞

Potentials in continuous time

- Discrete time: Equations relating time t to time $t+1$ based on random walks.
- Continuous time:
  Differential Equations describing the density evolution for Brownian motion with drift (known as the Kolmogorov forward and backward equations).

Example: From BBM to Brownboost

Potential function for BBM:

$$\phi(t-1,s) = \left(\frac{1}{2} - \gamma\right)\phi(t,s-1) + \left(\frac{1}{2} + \gamma\right)\phi(t,s+1)$$

$$\phi(T,s) = \begin{cases} 0 & s > 0 \\ 1 & s \leq 0 \end{cases} \qquad \varepsilon = \phi(0,0) = \mathrm{Binom}\left(T, \frac{T}{2}, \frac{1}{2} + \gamma\right)$$

Potential function for Brownboost:

$$\frac{\partial}{\partial t}\phi(t,s) = -\frac{1}{2}\frac{\partial^2}{\partial s^2}\phi(t,s) - 2\sqrt{\beta}\frac{\partial}{\partial s}\phi(t,s)$$

Boundary conditions:

$$\phi(1,s) = \begin{cases} 0 & s > 0 \\ 1 & s \leq 0 \end{cases} \qquad \varepsilon = \phi(0,0)$$

properties of the potential

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

End of game

$$\psi(i,0) = \begin{cases} 1 & i \le k \\ 0 & i > k \end{cases}$$

Illegal configuration:

$$\Psi(\text{configuration}) = \sum_{j=1}^{k} f(i)\psi(i,0) < \frac{1}{N}$$

Beginning of game

Number of errors if adversary always plays optimally

$r-1$, where r is the smallest integer for which

$$\psi(0,r) < \frac{1}{N}$$

BW Prediction algorithm

- Initialization: set r to be the number of errors against optimal adversary.
- Given expert predictions: choose prediction that will result (assuming error) in a lower-potential configuration.
- Decrease r if possible.

Main properties

- If algorithm is followed, the potential of the configuration never increases - is always $\leq$ 1/N
- Algorithm is min/max optimal.
    - Removing assumption that expert set is divisible min/max optimality holds if $N > 2^{2^k}$
    - Based on relation to Ulam's game with k lies [Spencer 92]

Alternative Representation

The difference between the two configurations can be represented as a weighted sum

$$\Psi\left(\text{configuration } 1\right) - \Psi\left(\text{configuration } 0\right) = \sum_{j=0}^{k} f(i)w(i,r)$$

$$w(i,r) = \psi(i+1,r-1) - \psi(i,r-1) = \frac{1}{2^{r-1}}\begin{pmatrix} r-1 \\ k-i \end{pmatrix}$$

The optimal prediction is according to the the sign of this weighted sum.

The BW algorithm

- Better error bound than exponential weights.
- A-priori assumption that one of the experts has loss at most k, we want a bound on the regret without any a priori assumptions.
- Instantaneous loss is restricted to $\{0,1\}$, we want it to be any number in $[-1,+1]$.

Design of NormalHedge

- BW: potential function depends on loss and number of remaining mistakes
- Normal-Hedge: Potential function based on regret and variance of the positive regrets

The NormalHedge potential

$$\text{Potential: } \psi(r,c) = \begin{cases} \exp\left(\dfrac{r^2}{2c}\right) & \text{if } r \geq 0 \\ 1 & \text{if } r \leq 0 \end{cases}$$

$$\text{Weight: } w(r,c) = \frac{\partial}{\partial r}\psi(r,c) = \begin{cases} \dfrac{r}{c}\exp\left(\dfrac{r^2}{2c}\right) & \text{if } r \geq 0 \\ 0 & \text{if } r \leq 0 \end{cases}$$

NormalHedge algorithm

for t=0,1,2,...

if $\forall i, R_i^t \leq 0$ then $w_i^t = 1/N$

else

set $c(t)$ so that $\frac{1}{N}\sum_{i=1}^{N}\psi\left(R_i^t, c(t)\right) = e$

$w_i^t = w\left(R_i^t, c(t)\right)$

Incur instantanous losses: $\left\langle l_1^t, l_2^t, \ldots, l_N^t \right\rangle$

Algorithm loss: $l_A^t = \frac{\sum_{i=1}^{N} w_i^t l_i^t}{\sum_{i=1}^{N} w_i^t}$

Update regrets: $R_i^{t+1} = R_i^t + l_A^t - l_i^t$

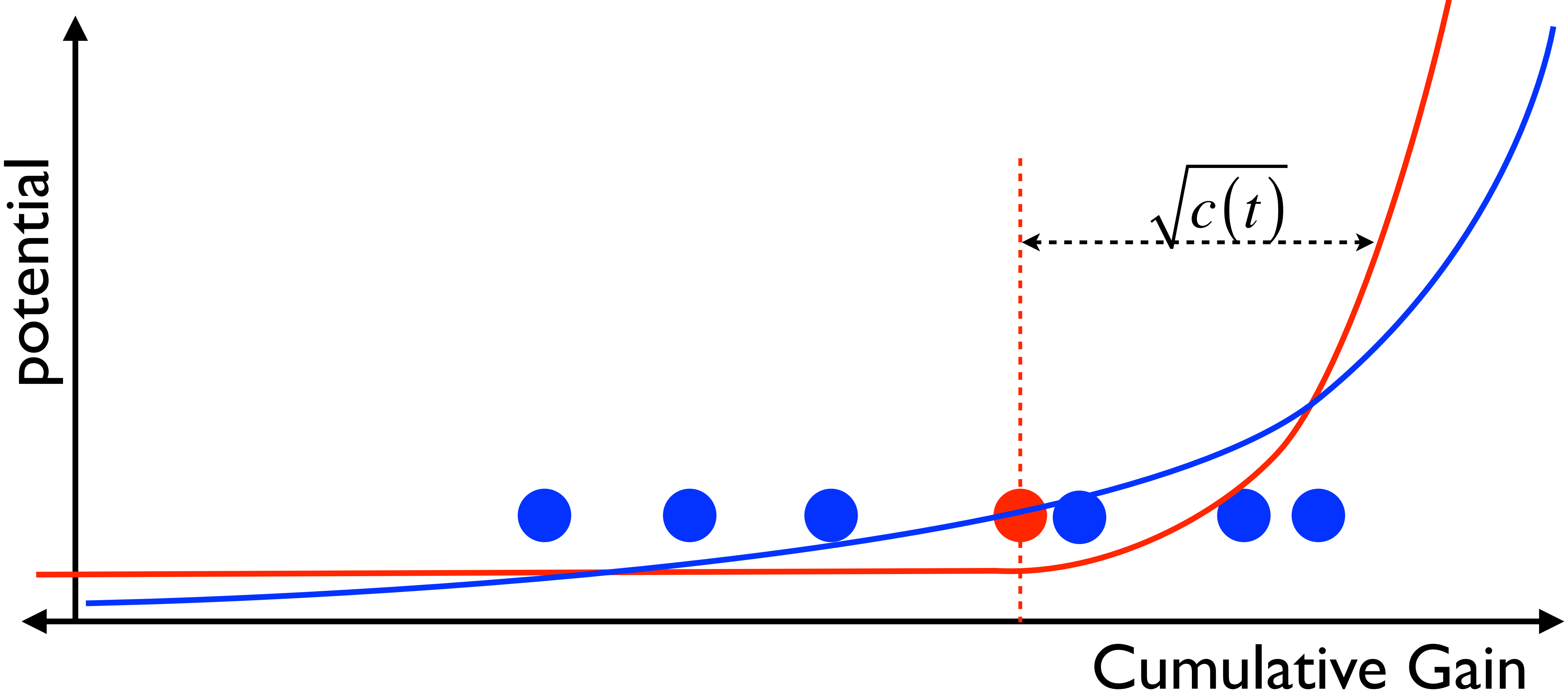
Illustrative Example

Expert

Algorithm

$\exp(\eta G)$

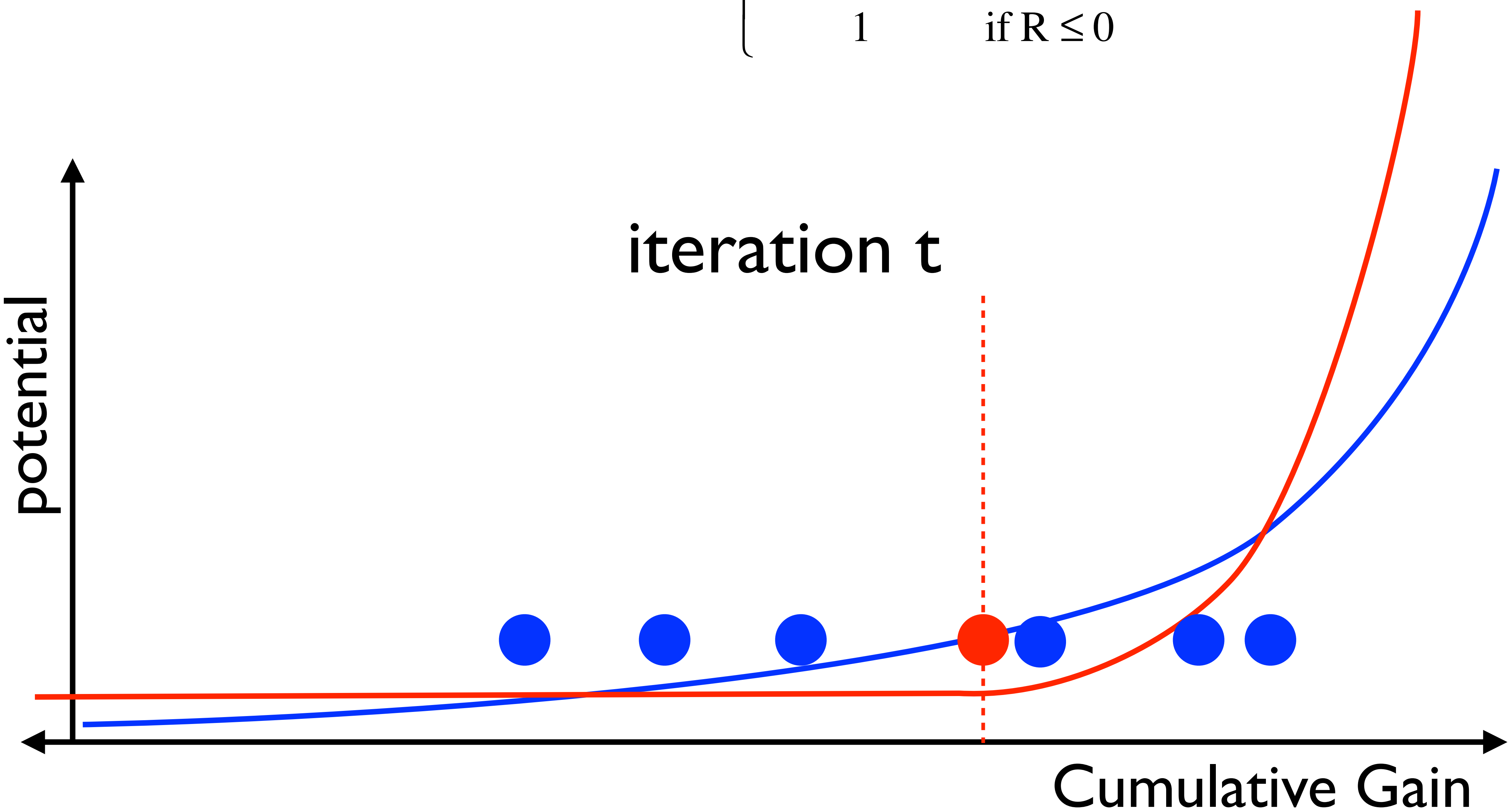
$$\begin{cases} \exp\left(\dfrac{R^2}{2c}\right) & \text{if } R \geq 0 \\ 1 & \text{if } R \leq 0 \end{cases}$$



Illustrative Example

Expert — $\exp(\eta G)$

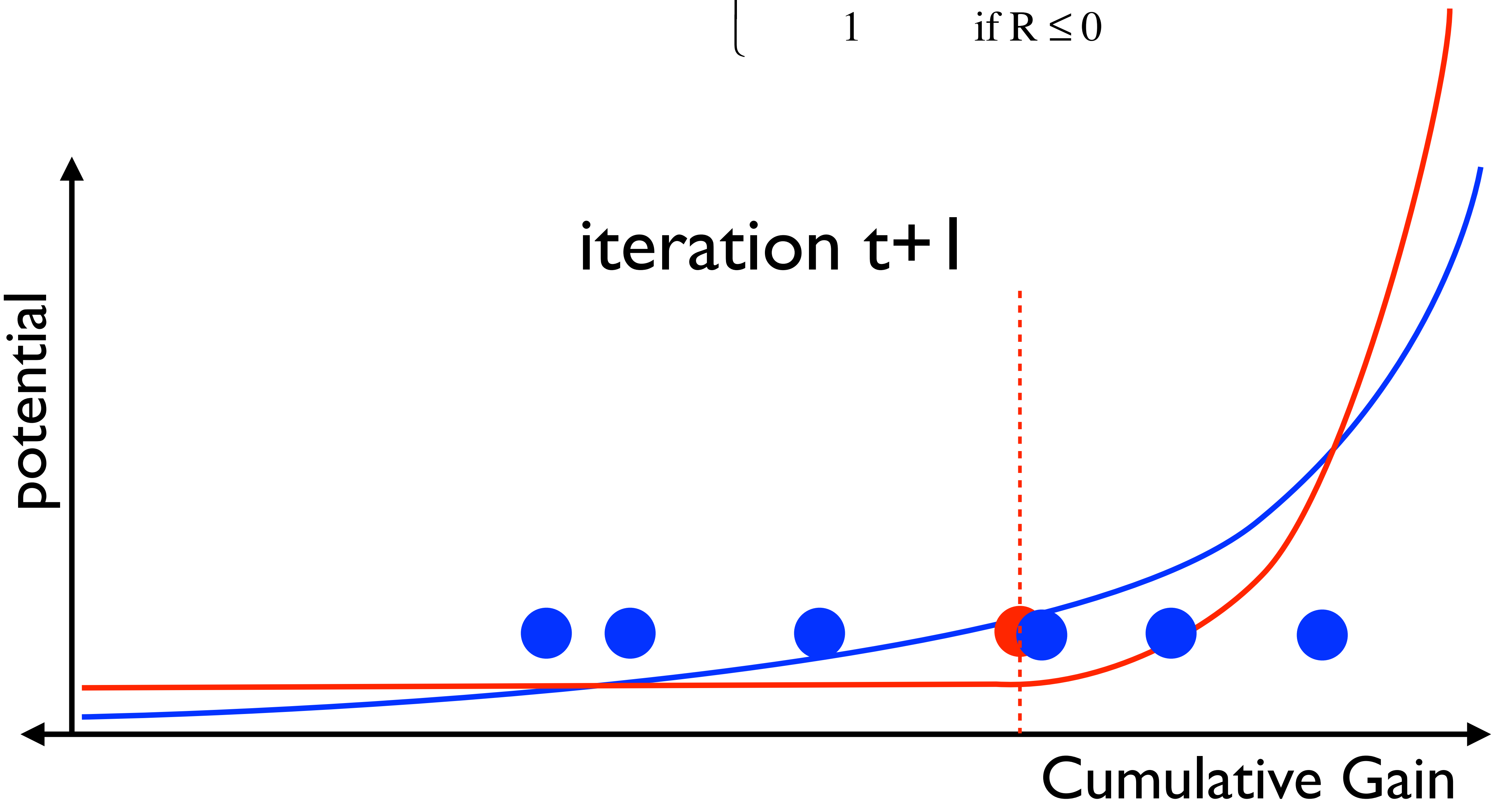
Algorithm — $\begin{cases} \exp\left(\dfrac{R^2}{2c}\right) & \text{if R} \geq 0 \\ 1 & \text{if R} \leq 0 \end{cases}$



Illustrative Example

Expert — $\exp(\eta G)$

Algorithm — $\begin{cases} \exp\left(\dfrac{R^2}{2c}\right) & \text{if } \mathrm{R} \geq 0 \\ 1 & \text{if } \mathrm{R} \leq 0 \end{cases}$

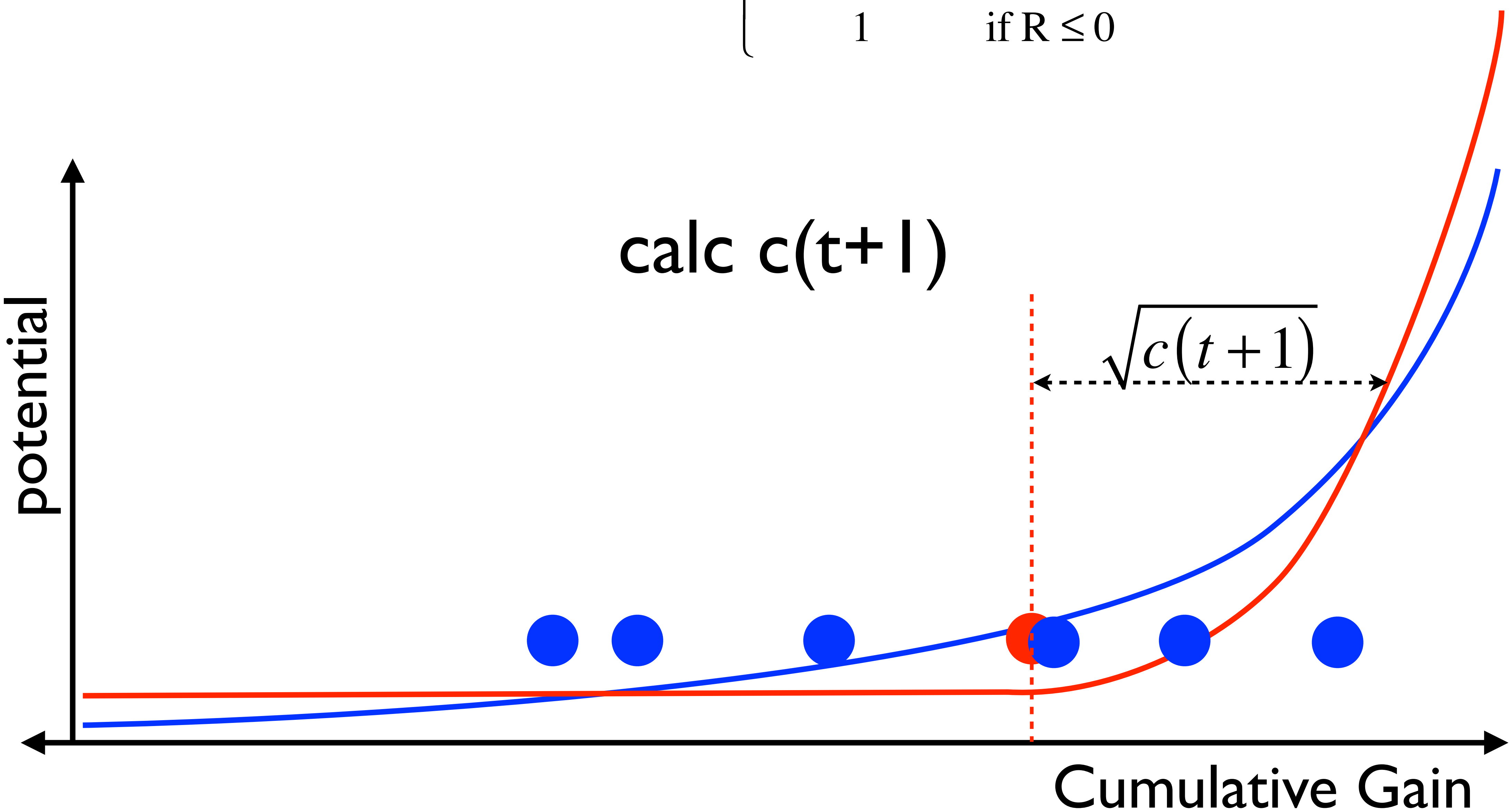


Illustrative Example

- Expert
- Algorithm

Blue line: $\exp(\eta G)$

Red line: $\begin{cases} \exp\left(\dfrac{R^2}{2c}\right) & \text{if R} \geq 0 \\ 1 & \text{if R} \leq 0 \end{cases}$



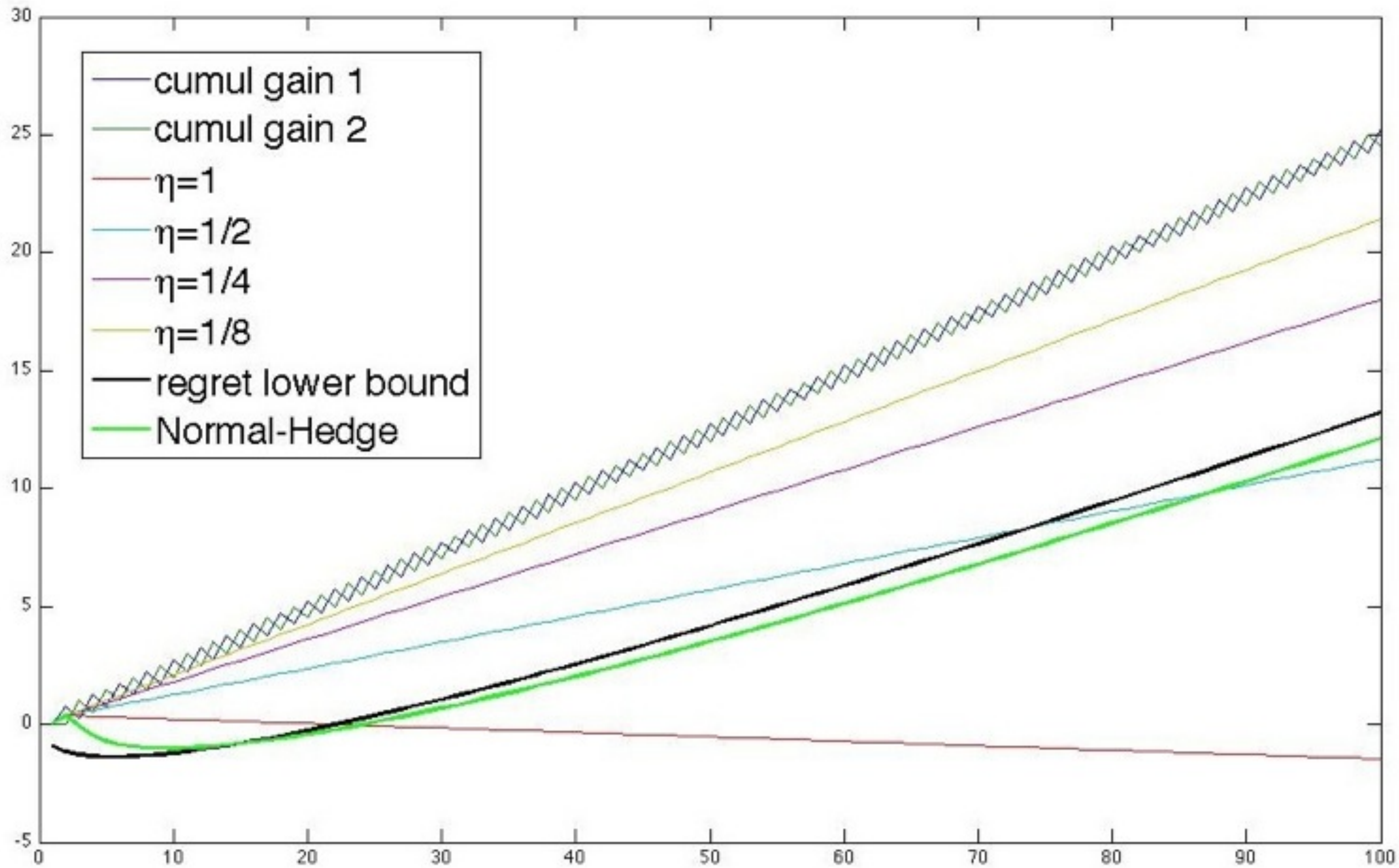
Normal-Hedge Performance bound

[Chaudhuri, Freund & Hsu 2009]

The regret of NormalHedge is upper bounded by

$$O\left(\sqrt{T \ln N + \ln^3 N}\right)$$

Performance on flip-flop



Summary

- Drifting games is a method for **deriving** new potential functions for online learning and boosting.
- Performed by working backwards in time, starting from final loss function and working backwards.
- Adversarial strategy: random walk / normal distribution.
- Yields Brown-boost
- Yields Normal-Hedge
- Both Brownboost and normal-hedge have difficult open problems.