Reinforcement Learning

Prof. Christian Bauckhage

lecture 04
Monte Carlo tree search

in this lecture, we study *Monte Carlo tree search (MCTS)*

to begin with, we familiarize ourselves with *Monte Carlo methods* in general and the granddaddy of them all *Monte Carlo integration* in particular

then, we look at *Monte Carlo tree search* for decision making (in two-player games)

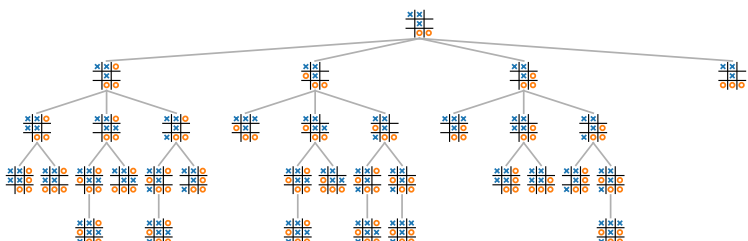
finally, we touch upon the he *exploration / exploitation dilemma* and how to deal with it

outline

recap

a search tree for decision making

- $b \Leftrightarrow$ (average) branching factor
- $d \Leftrightarrow$ shallowest / minimum depth (of a leaf)
- $D \Leftrightarrow$ deepest / maximum depth (of a leaf)



$b \approx 1.94, \quad d = 1, \quad D = 4$

search strategies / algorithms for decision making

depth-first search, (time) complexity $O\left(b^D\right)$
$\Rightarrow$ too time intensive

breadth-first search, complexity $O\left(b^d\right)$
$\Rightarrow$ too memory intensive

α-β *pruning*, complexity $O\left(b^{cD}\right)$ where $\frac{1}{2} \leqslant c \leqslant 1$
$\Rightarrow$ still not cheap enough

depth limited search, complexity $O\left(b^\lambda\right)$
$\Rightarrow$ requires *evaluation functions*

next, we will look at

Monte Carlo tree search, complexity $O(D)$

this sounds fantastic, but . . .

recall

big O notation

$$f(x) \in O\big(g(x)\big) \quad \Leftrightarrow \quad \exists\, C : \forall\, x > x_0 : \ \big|f(x)\big| \leqslant C \cdot \big|g(x)\big|$$

for example

$$a_0 + a_1\, x + a_2\, x^2 + a_3\, x^3 \in O\big(\quad\big)$$

recall

big O notation

$$f(x) \in O\big(g(x)\big) \quad \Leftrightarrow \quad \exists\, C : \forall\, x > x_0 : \ \big|f(x)\big| \leqslant C \cdot \big|g(x)\big|$$

for example

$$a_0 + a_1\, x + a_2\, x^2 + a_3\, x^3 \in O\big(x^3\big)$$

recall

big O notation

$$f(x) \in O\big(g(x)\big) \quad \Leftrightarrow \quad \exists\, C : \forall\, x > x_0 : \ \big|f(x)\big| \leqslant C \cdot \big|g(x)\big|$$

for example

$$a_0 + a_1\, x + a_2\, x^2 + a_3\, x^3 \in O\big(x^3\big)$$

warning

constant C may be large

Monte Carlo methods

Monte Carlo method

$\Leftrightarrow$ a very general mathematical technique, typically refers to:

problem solving by means of *stochastic techniques*, i.e. by means of techniques involving *chance* and *probabilities*

according to Wolfram MathWorld:

"Any method which solves a problem by generating suitable random numbers and observing that fraction of the numbers obeying some property or properties."

mathworld.wolfram.com/MonteCarloMethod.html

common applications

computing (numerical) solutions to analytically intractable problems

simulating (the dynamics of) very complex (dynamical) systems

$\Leftrightarrow$ solving *integrals* and *systems of differential equations*

components of an MC algorithm

- definition of problem domain / possible inputs
- random number generator / sampling process
- deterministic computation with random inputs
- aggregation of results

let's look at an . . .

example

next, we will

compute the *Hellinger distance* between two *Weibull distributions*

$\Leftrightarrow$ consider the problem of computing a certain distance between functions

$\Leftrightarrow$ encounter the problem of having to **evaluate quite a demanding integral**

indeed, demanding integrals are the original use case for Monte Carlo methods

N. Metropolis and S. Ulam, *The Monte Carlo Method*, J. American Statistical Association, 44(247), **1949**

we emphasize that

our integral can *not* be solved analytically; there is *no* closed form solution

while we could evaluate it using common numerical integration techniques (quadrature rules) . . .

. . . we will evaluate it using Monte Carlo integration (also called *hit or miss integration*) to demonstrate what Monte Carlso methods are all about

we emphasize that

our integral can *not* be solved analytically; there is *no* closed form solution

while we could evaluate it using common numerical integration techniques (quadrature rules) . . .

. . . we will evaluate it using Monte Carlo integration (also called *hit or miss integration*) to demonstrate what Monte Carlso methods are all about

we begin by recalling

distance, *Hellinger distance*, *Weibull distribution*

distance

$\Leftrightarrow$ a function $d : S \times S \to \mathbb{R}_+$ such that for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in S$

$$d(\boldsymbol{x}, \boldsymbol{y}) \geqslant 0$$

$$d(\boldsymbol{x}, \boldsymbol{y}) = 0 \;\Leftrightarrow\; \boldsymbol{x} = \boldsymbol{y}$$

$$d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x}) \qquad \textit{symmetry}$$

$$d(\boldsymbol{x}, \boldsymbol{z}) \leqslant d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z}) \qquad \textit{triangle inequality}$$

Hellinger distance

$\Leftrightarrow$ a distance where $S = \{f \mid f \text{ is a pdf over } \mathbb{R}\}$

for the *squared* Hellinger distance, we have

$$H^2(f_1, f_2) = \frac{1}{2} \int \left(\sqrt{f_1(t)} - \sqrt{f_2(t)} \right)^2 dt$$

$$= 1 - \int \sqrt{f_1(t) \cdot f_2(t)}\, dt$$

we also point out this interesting property

$$0 \leqslant H(f_1, f_2) \leqslant 1$$

Weibull distribution

$\Leftrightarrow$ a continuous probability for $x \geqslant 0$

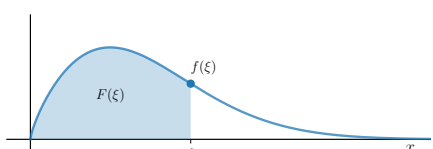
probability density function

$$f(x) = \frac{d}{dx} F(x) \; = \frac{\kappa}{\lambda} \left(\frac{x}{\lambda}\right)^{\kappa - 1} e^{-\left(\frac{x}{\lambda}\right)^{\kappa}}$$

cumulative density function

$$F(x) = \int_0^x f(\xi)\, d\xi, \; = 1 - e^{-\left(\frac{x}{\lambda}\right)^{\kappa}}$$

$\kappa, \lambda > 0$ determine *shape* and *scale*



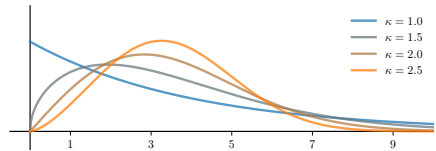
Weibull shapes and scales

we have

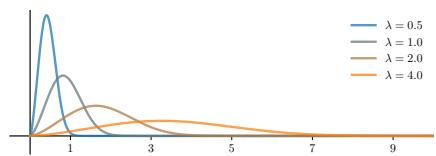
$$f(x) = \frac{\kappa}{\lambda}\left(\frac{x}{\lambda}\right)^{\kappa-1} e^{-\left(\frac{x}{\lambda}\right)^{\kappa}}$$

where

- κ determines shape
- λ determines scale



$\lambda = 4.0$



$\kappa = 2.5$

question

why consider the Hellinger distance between two Weibulls as an example for the use of Monte Carlo integration ?

question

why consider the Hellinger distance between two Weibulls
as an example for the use of Monte Carlo integration ?

answer

because . . .

note

for two Weibull distributions $f_1(x)$ and $f_2(x)$, we have

$$H^2(f_1, f_2) = 1 - \int_0^\infty \left[\frac{\kappa_1}{\lambda_1}\left(\frac{x}{\lambda_1}\right)^{\kappa_1 - 1} e^{-(x/\lambda_1)^{\kappa_1}}\right]^{\frac{1}{2}} \left[\frac{\kappa_2}{\lambda_2}\left(\frac{x}{\lambda_2}\right)^{\kappa_2 - 1} e^{-(x/\lambda_2)^{\kappa_2}}\right]^{\frac{1}{2}} dx$$

computing H^2 therefore requires solving the integral

$$\gamma \cdot \int_0^\infty x^{\frac{1}{2}(\kappa_1 + \kappa_2)} e^{-\frac{1}{2}\left((x/\lambda_1)^{\kappa_1} + (x/\lambda_2)^{\kappa_2}\right)} dx$$

and, unless $\kappa_1 = \kappa_2$, this *cannot be done analytically*

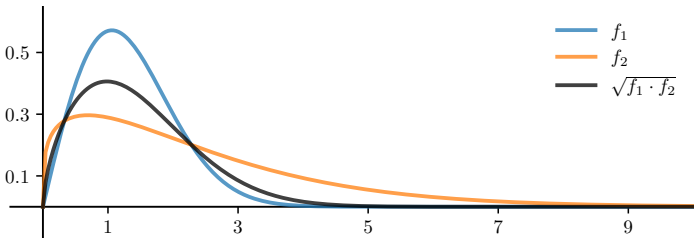
example

given

$$f_1(x) = f\left(x \mid \kappa = 2.00, \lambda = 1.50\right)$$
$$f_2(x) = f\left(x \mid \kappa = 1.25, \lambda = 2.50\right)$$

compute the area

$$A = \int_0^7 \sqrt{f_1(x) \cdot f_2(x)}\, dx$$



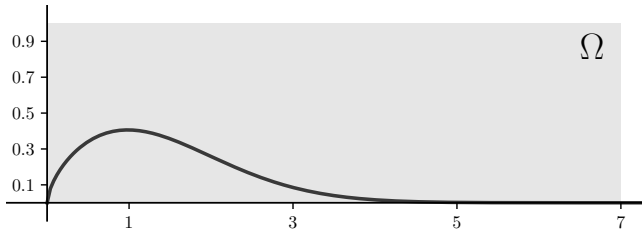
Monte Carlo integration (in one dimension)

integrate a bounded function $f(x)$ over a bounded interval

here, we have $0 \leqslant f(x) \leqslant 1$ and $a \leqslant x \leqslant b$ and we *define*

$$a = 0$$
$$b = 7$$

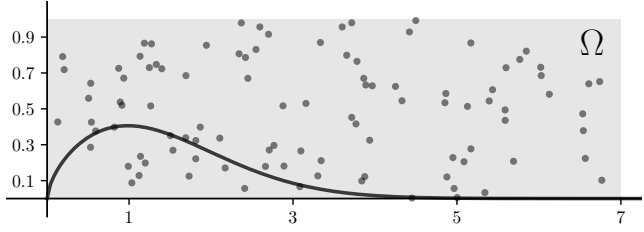
let Ω be the *rectangle* $\{[x, y] \mid a \leqslant x \leqslant b \wedge 0 \leqslant y \leqslant 1\}$ with area $A(\Omega) = (b-a) \cdot (1-0)$



let $[x, y]$ be a random point sampled *uniformly* from rectangle Ω

the probability that $[x, y]$ lies below the graph of $f(x)$ is given by

$$p = \frac{A}{A(\Omega)} = \frac{\int_a^b f(x)\,dx}{(b-a)\cdot(1-0)}$$



note

we have

$$p = \frac{A}{A(\Omega)}$$

and want to determine

$$A = p \cdot A(\Omega)$$

where we know $A(\Omega)$ but don't know p

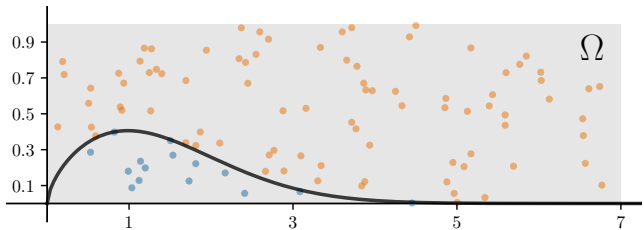
yet, *if* we could get a good *estimate* $\hat{p} \approx p$,
then we could get a good *estimate*

$$\hat{A} \approx \hat{p} \cdot A(\Omega)$$

so, let's go for it ...

a point $[x, y] \in \Omega$ with $y \leqslant f(x)$ is called a hit

a point $[x, y] \in \Omega$ with $y > f(x)$ is called a miss



generate a sample $\{[x_i, y_i]\}_{i=1}^{n} \subset \Omega$ of n *uniform* random points

consider the indicator function (also called hit-or-miss function)

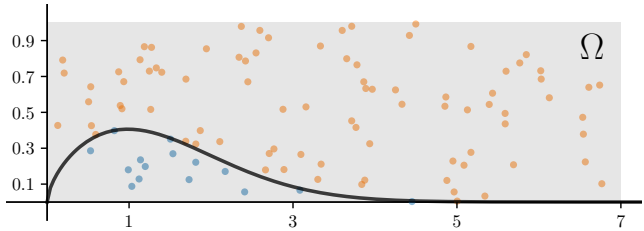
$$h(x, y) = \begin{cases} 1, & \text{if } y \leqslant f(x) \\ 0, & \text{otherwise} \end{cases}$$

and compute

$$n_h = \sum_{i=1}^{n} h(x_i, y_i)$$

then

$$p \approx \hat{p} = \frac{n_h}{n}$$



results

using `scipy.integrate.quad`
for numerical integration gives

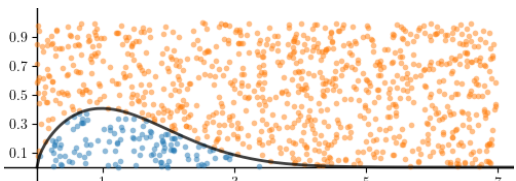
$$A \approx 0.868$$

results

using `scipy.integrate.quad`
for numerical integration gives

$$A \approx 0.868$$

Monte Carlo integration yields



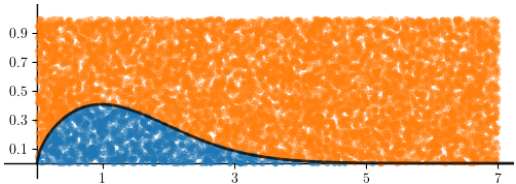
$$n = 1000 \;\Rightarrow\; A \approx 0.854$$

results

using `scipy.integrate.quad`
for numerical integration gives

$$A \approx 0.868$$

Monte Carlo integration yields



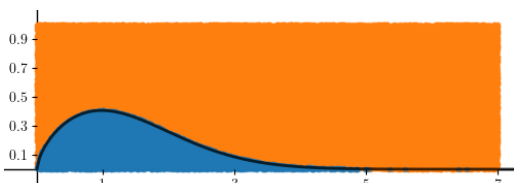
$n = 10000 \Rightarrow A \approx 0.8456$

results

using `scipy.integrate.quad`
for numerical integration gives

$$A \approx 0.868$$

Monte Carlo integration yields



$$n = 100000 \;\Rightarrow\; A \approx 0.86653$$

take home messages

nobody in their right mind would resort to Monte Carlo techniques for solving simple problems such as this one

however, in higher dimensions, Monte Carlo integration typically outperforms deterministic numeric approaches (because of the *curse of dimensionality* !)

take home messages

nobody in their right mind would resort to Monte Carlo techniques for solving simple problems such as this one

however, in higher dimensions, Monte Carlo integration typically outperforms deterministic numeric approaches (because of the *curse of dimensionality* !)

also, *uniform sampling* is usually too naïve so that people commonly resort to

- *stratified sampling*
- *importance sampling*
- *Markov chain Monte Carlo sampling*

Monte Carlo tree search

Monte Carlo tree search (MCTS)

$\Leftrightarrow$ Monte Carlo exploration of a search tree / state space

MCTS is a surprisingly recent idea

> B. Abramson, *Expected-outcome: A General Model of Static Evaluation*, IEEE PAMI, 12(2), **1990**
>
> B. Brügmann, *Monte Carlo GO*, Tech. Report, MPI of Physics, **1993**

MCTS was even more recently popularized through

> R. Coulom, *Efficient Selectivity and Backup Operators in Monte Carlo Tree Search*, Proc. Int. Conf. Computers and Games, **2006**

ever since, it has been used in extensively and **played a key role** in

> D. Silver et al., *Mastering the Game of GO with Deep Neural Networks and Tree Search*, Nature 529(7587), **2016**

disclaimer (1)

next, we focus on MCTS for decision making in turn-based games

in this context, MCTS allows for estimating the *rewards* of possible successors of a current game state

in this context, high rewards are understood to indicate auspicious moves

(this is akin to computing minmax values of possible successors of a current game state in order to decide for a move)

disclaimer (2)

as always, we assume that we are given

- an initial (game) state / root node n_0
- a successor function $Succ(n)$ to compute all successors n' of a node n
- a goal test function $isTerminal(n)$ to determine if n is a terminal node
- a utility function $Util(n)$ to compute the reward of a terminal node n

similar to what we did when we studied minmax searches, we assume that

- each node n of a Monte Carlo search tree is attributed with a value $reward[n]$

note

Monte Carlo tree search happens in iterations

each individual iteration involves of two phases

1) a tree walk (from the root to a current fringe node)
2) a random walk (from the current fringe to a terminal state)

MCTS – pseudocode

$tree \leftarrow n_0$
$fringe \leftarrow \{n_0\}$

for $t = 1, \ldots, T$
    $r \leftarrow \mathit{TreeWalk}(tree, n_0)$

move to $n' \in Succ(n_0)$ of highest reward r'

function *TreeWalk*$(tree, n)$
  if *isTerminal*(n)
    return *Util*(n)

  if $n \notin fringe$
    $n' \leftarrow$ *InformedChoice*$(Succ(n))$
    $r \leftarrow$ *TreeWalk*$(tree, n')$
  else
    $n' \leftarrow$ *Expand*(n)
    $r \leftarrow$ *RandomWalk*$(tree, n')$

  $reward[n] \leftarrow reward[n] + r$

  return r

function *Expand*(n)
  $n' \leftarrow$ *RandomChoice*$(Succ(n))$
  if $n' \notin tree$
    add n' as child of n to *tree*
    $fringe \leftarrow fringe \cup \{n'\}$
  if n is now fully expanded
    $fringe \leftarrow fringe \setminus \{n\}$
  return n'

function *RandomWalk*$(tree, n)$
  if *isTerminal*(n)
    return *Util*(n)
  else
    $n' \leftarrow$ *RandomChoice*$(Succ(n))$
    $r \leftarrow$ *RandomWalk*$(tree, n')$
  return r

note

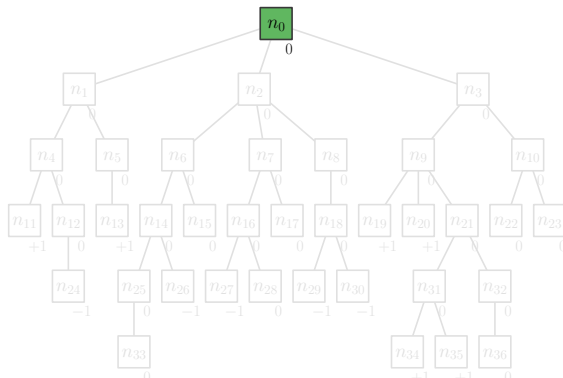
we will soon say more about $\mathit{InformedChoice}\big(\mathit{Succ}(n)\big)$ and $\mathit{RandomChoice}\big(\mathit{Succ}(n)\big)$

but first, we look at a ...

didactic example

consider an unexplored state space / initial search tree

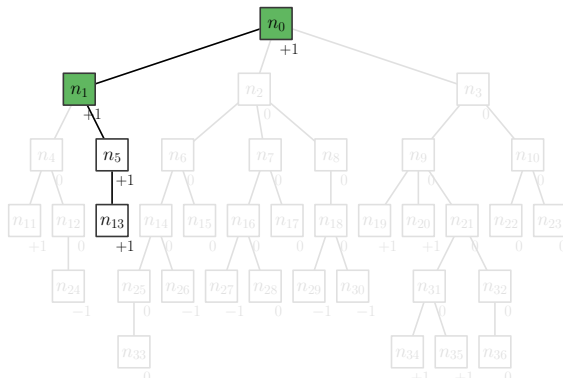
at this point, we only know the initial state / root node n_0 but nothing about its r value



$t = 0$

since $n_0 \in \mathit{fringe}$, MCTS starts a random walk towards a terminal state

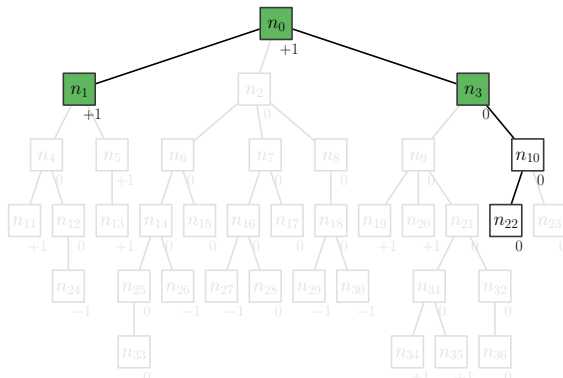
the 1st visited state n_1 is added to tree and fringe, rewards are propagated up the tree



$t = 1$

in iteration $t = 2$, MCTS starts another random walk towards a terminal state

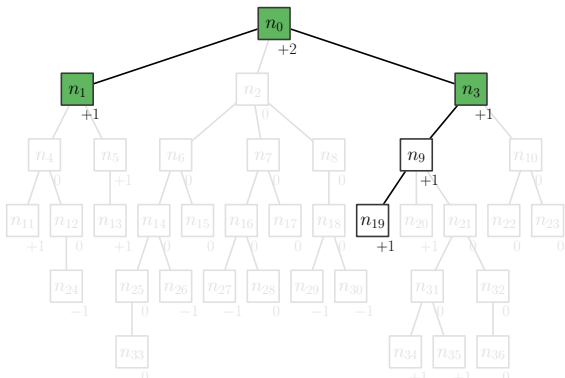
the 1st visited state is now n_3, (boring) rewards are again propagated upwards



$t = 2$

in iteration $t = 3$, MCTS starts another random walk towards a terminal state

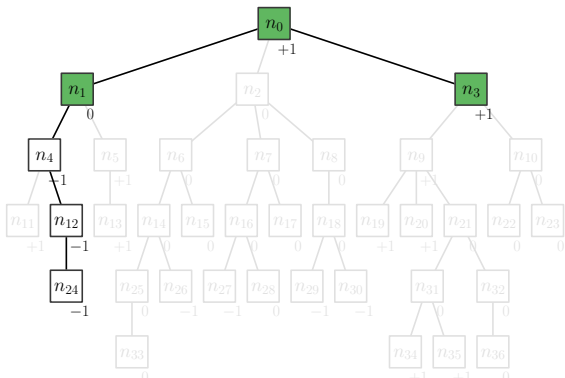
the 1st visited state is yet again n_3, more interesting rewards are propagated



$t = 3$

in iteration $t = 4$, MCTS starts another random walk towards a terminal state

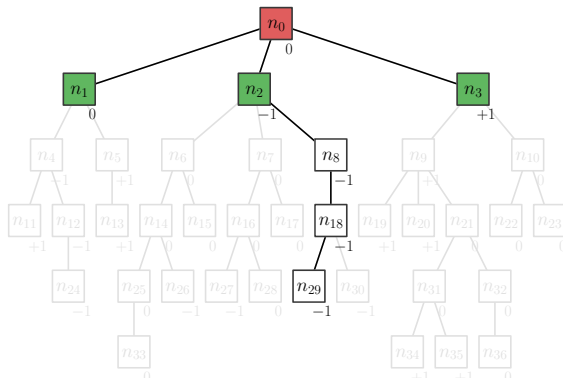
the 1st visited state again n_1, rewards are propagated



$t = 4$

in iteration $t = 5$, MCTS starts another random walk at n_0

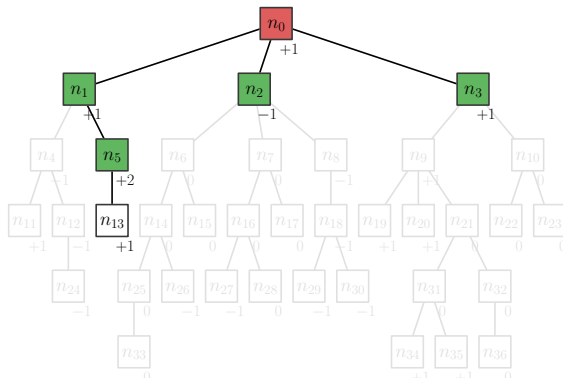
the 1st visited state is now n_2, so n_0 is fully expanded and removed from the fringe



$t = 5$

in iteration $t = 6$, $n_0 \notin \mathit{fringe}$, so MCTS starts a tree walk at n_0 which first visits n_1

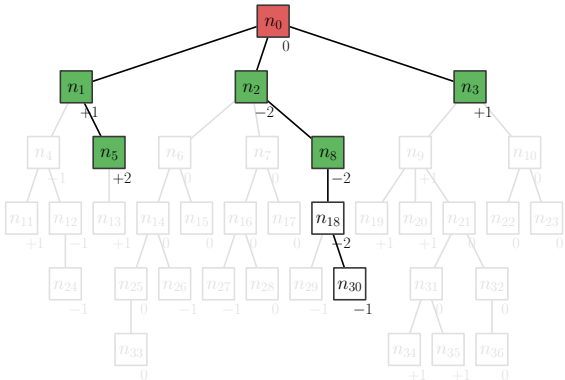
as $n_1 \in \mathit{fringe}$, MCTS starts a random walk, its 1st state n_5 is added to tree and fringe



$t = 6$

here are iterations $t = 7$ and $t = 8$

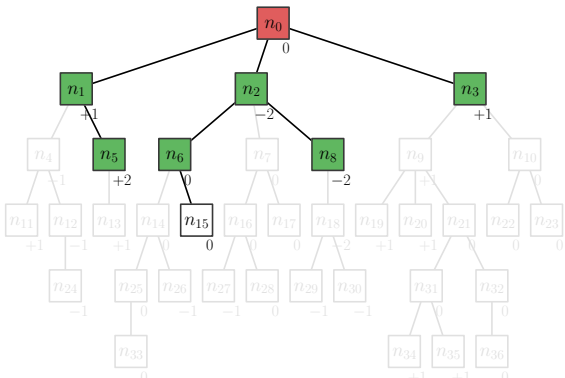
observe how rewards are updated



$t = 7$

here are iterations $t = 7$ and $t = 8$

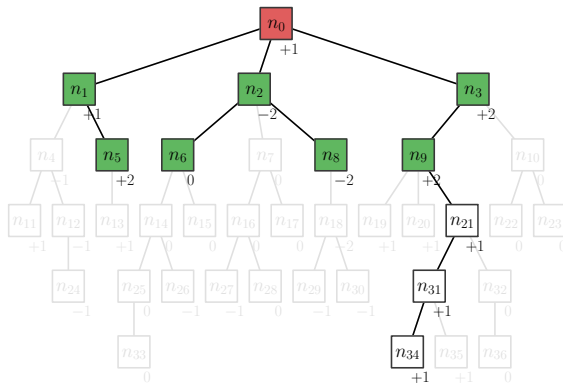
observe how rewards are updated



$t = 8$

here are further iterations $t \in \{9, \ldots, 15\}$ and finally $T = 30$

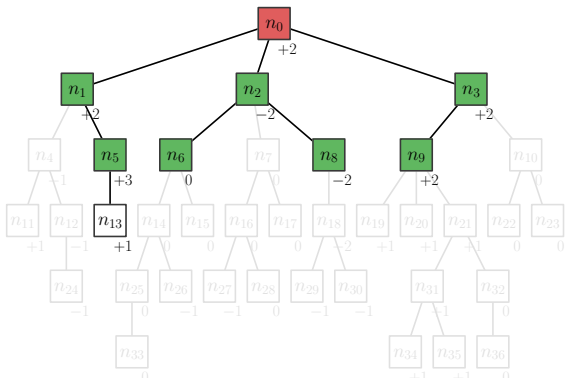
note how the tree (slowly) grows and rewards are updated



$t = 9$

here are further iterations $t \in \{9, \ldots, 15\}$ and finally $T = 30$

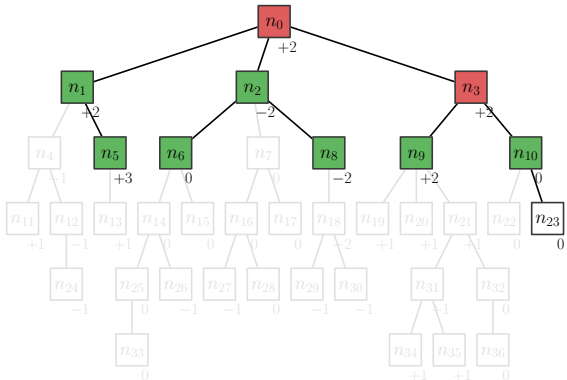
note how the tree (slowly) grows and rewards are updated



$t = 10$

here are further iterations $t \in \{9, \ldots, 15\}$ and finally $T = 30$

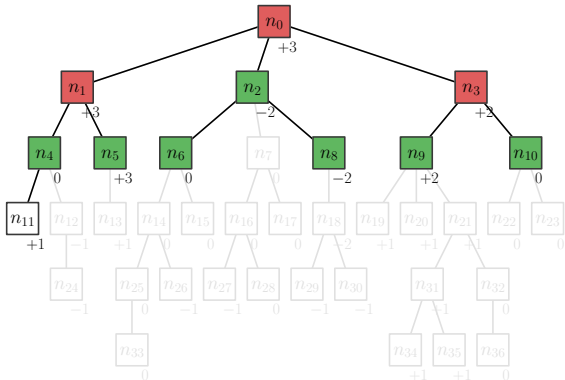
note how the tree (slowly) grows and rewards are updated



$t = 11$

here are further iterations $t \in \{9, \ldots, 15\}$ and finally $T = 30$

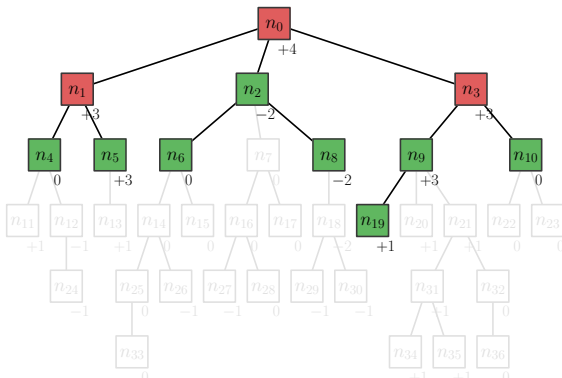
note how the tree (slowly) grows and rewards are updated



$t = 12$

here are further iterations $t \in \{9, \ldots, 15\}$ and finally $T = 30$

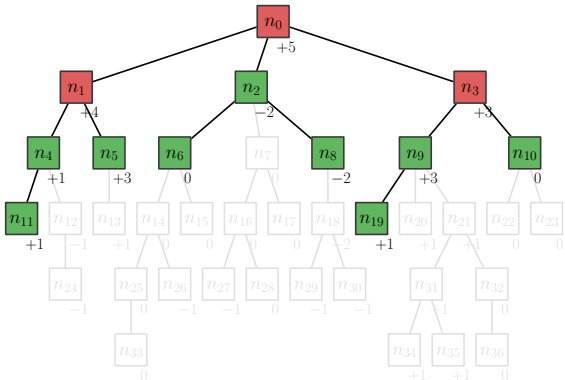
note how the tree (slowly) grows and rewards are updated



$t = 13$

here are further iterations $t \in \{9, \ldots, 15\}$ and finally $T = 30$

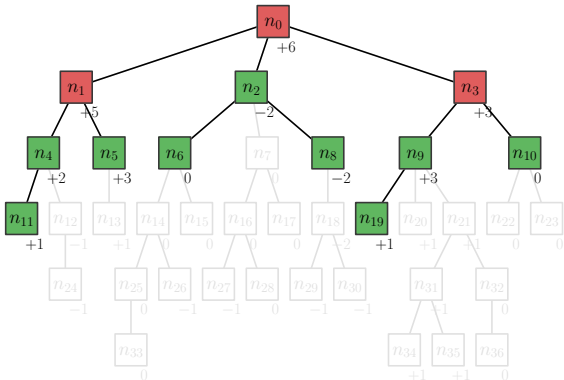
note how the tree (slowly) grows and rewards are updated



$t = 14$

here are further iterations $t \in \{9, \ldots, 15\}$ and finally $T = 30$

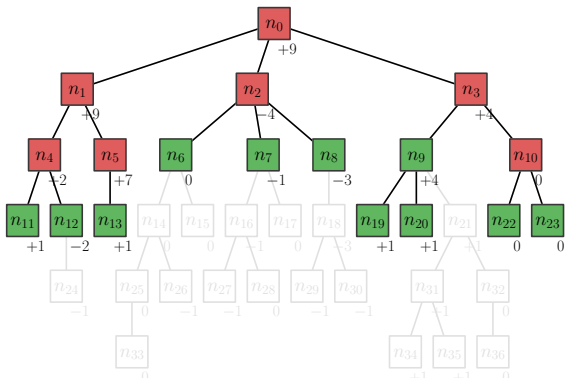
note how the tree (slowly) grows and rewards are updated



$t = 15$

here are further iterations $t \in \{9, \ldots, 15\}$ and finally $T = 30$

note how the tree (slowly) grows and rewards are updated



$T = 30$

take home messages

computations per iteration are $O(D)$ $\Rightarrow$ overall effort $O(D)$

and yet, number of iterations T usually needs to be large

the two *selection* functions *InformedChoice* and *RandomChoice*
are usually non-trivial but *independent* of game mechanics

they allow for an *expectation based search* without game specific evaluation functions !!!!!

we obviously need to unpack the latter statements . . .

the exploration / exploitation dilemma

the exploration / exploitation dilemma

say, you are playing the slot machines in a casino and want to maximize your payoff

should you

a) continue playing the slot machine with the highest payoff so far ?

b) check other machines which *may* yield higher payoff ?

the exploration / exploitation dilemma

say, you are playing the slot machines in a casino and want to maximize your payoff

should you

a) continue playing the slot machine with the highest payoff so far ?

b) check other machines which *may* yield higher payoff ?

$\Leftrightarrow$ to *exploit* or to *explore*, that is the question

multi-armed bandit problem

$\Leftrightarrow$ faced by an agent who tries to acquire new knowledge *and* to maximize payoff / rewards based on prior knowledge

problem formalization

assume reward distributions for K arms

$$r_1(x), r_2(x), \ldots, r_K(x)$$

such that the expected reward for arm a is

$$\mu_a = \mathbb{E}\left[r_a(x)\right] = \int x\, r_a(x)\, dx$$

furthermore define the maximum reward as

$$\mu_* = \max_a \mu_a$$

these are *unknown* quantities but we can use them in mathematical analysis anyway

the regret ρ after having played T rounds is

$$\rho = T \cdot \mu_* - \sum_{t=1}^{T} r[t]$$

where $r[t]$ is the reward received in round t

to ease our analysis and w.l.o.g. we assume

$$0 \leqslant r[t] \leqslant 1$$

since regret is high if we choose badly (low reward arms), the goal is to find a zero-regret strategy such that the regret behaves like this

$$\lim_{T \to \infty} \frac{\rho}{T} = 0$$

in plain English

we have K arms $a_1, \ldots, a_K$ with average rewards $\mu_1, \ldots \mu_K$

(at least) one of these arms, call it a_*, will have the highest average reward $\mu_{a_*} = \mu_*$

if we knew this arm, we would chose it in each round since

$$\rho = T \cdot \mu_* - \sum_{t=1}^{T} r_*[t]$$

$$\Leftrightarrow \quad \frac{\rho}{T} = \mu_* - \frac{1}{T} \sum_{t=1}^{T} r_*[t]$$

$$\Leftrightarrow \quad \lim_{T \to \infty} \frac{\rho}{T} = \mu_* - \mu_* = 0$$

note

the fundamental problem is that we don't know arm a_* in advance

$\Leftrightarrow$ we need a strategy that allows us to find / identify it *whilst* playing

defining an indicator function

$$h\big(r[t], a\big) = \begin{cases} r[t] & \text{if reward at time } t \text{ was due to arm } a \\ 0 & \text{otherwise} \end{cases}$$

we can write

$$\sum_{t=1}^{T} r[t] = \sum_{a} \sum_{t=1}^{T} h\big(r[t], a\big)$$

if $n_a \equiv n_a(T)$ counts how often arm a has been chosen up to round T, then the empirical mean

$$\tilde{\mu}_a \equiv \tilde{\mu}_a(T) = \frac{1}{n_a(T)} \sum_{t=1}^{T} h(r[t], a)$$

is an estimator of the unknown quantity μ_a

using the above, we can rewrite the total reward

$$\sum_{t=1}^{T} r[t] = \sum_{a} \sum_{t=1}^{T} h(r[t], a) = \sum_{a} n_a(T) \cdot \tilde{\mu}_a(T)$$

the numbers n_a and $\tilde{\mu}_a$ can be computed and kept track of whilst playing the bandits

now, the crucial question is

how far are the estimates $\tilde{\mu}_a$ from the true but unknown μ_a ?

if we knew for each arm a that

$$\left|\mu_a - \tilde{\mu}_a\right| < b_a$$

for "some" bound b_a, we knew

$$\mu_a < \tilde{\mu}_a + b_a$$

$\Rightarrow$ in each round, we could select

$$a = \underset{a'}{\text{argmax}}\ \tilde{\mu}_{a'} + b_{a'}$$

strategies for the multi-armed bandit problem (1)

greedy selection

in round t, select arm a as

$$a = \operatorname*{argmax}_{a'} \tilde{\mu}_{a'}$$

then play the round and then update n_a and $\tilde{\mu}_a$

$\Leftrightarrow$ ignore b_a and just select the arm with the highest average so far

this will easily get stuck in local optima due to *lack of exploration*

a simple remedy is ...

strategies for the multi-armed bandit problem (2)

ϵ-greedy selection

in round t, select arm a as

$$a = \begin{cases} \underset{a'}{\text{argmax}}\ \tilde{\mu}_{a'} & \text{with probability } 1-\epsilon \\ \mathcal{U}\left\{a_i\right\}_{i=1}^{K} & \text{with probability } \epsilon \end{cases}$$

then play the round and then update n_a and $\tilde{\mu}_a$

"one can show" that, over time, this scheme will identify a_*

its convergence rate will depend on the choice of $0 < \epsilon < 1$

strategies for the multi-armed bandit problem (3)

upper confidence bound selection

in round t, select arm a as

$$a = \operatorname*{argmax}_{a'} \tilde{\mu}_{a'} + c \cdot \sqrt{\frac{\ln t}{n_{a'}}}$$

then play the round and then update n_a and $\tilde{\mu}_a$

note: this now involves a (time-dependent) bound b_a

question

where did this come from ?

answer

let's buckle up and see . . .

to begin with, we observe

if $p(X) = \delta$, then $p(\neg X) = 1 - \delta$

$\Leftrightarrow$ if $p(X)$ is *at best* δ

$$p(X) \leqslant \delta$$

then $p(\neg X)$ is *at least* $1 - \delta$

$$p(\neg X) \geqslant 1 - \delta$$

next, we “recall” . . .

the Hoeffding inequality ($\Leftrightarrow$ a most fundamental result for learning theory)

if $X_1, \ldots, X_n$ are independent random variables with $0 \leqslant X_i \leqslant 1$, then

$$p\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mathbb{E}\left[X_i\right]\right| \geqslant \epsilon\right) \leqslant \delta = 2\ \exp\left(-2\,n\,\epsilon^2\right)$$

this generalizes to $\alpha_i \leqslant X_i \leqslant \beta_i$

$\Rightarrow$ if we think of the n variables X_i as the n_a rewards from arm a up to time T, we have

$$\frac{1}{n}\sum_{i=1}^{n} X_i \quad \Leftrightarrow \quad \frac{1}{n_a}\sum_{t=1}^{T} h\big(r[t], a\big) = \tilde{\mu}_a$$

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\big[X_i\big] \quad \Leftrightarrow \quad \frac{1}{n_a}\sum_{i=1}^{n_a} \mu_a = \frac{n_a}{n_a}\,\mu_a = \mu_a$$

$\Rightarrow$ if we further let $\epsilon = b_a$, we find the inequality

$$p\Big(\big|\tilde{\mu}_a - \mu_a\big| \geqslant b_a\Big) \leqslant \delta = 2\,\exp\left(-2\,n_a\,b_a^2\right)$$

$\Rightarrow$ using our preparatory observation, this is equivalent to

$$p\Big(\big|\mu_a - \tilde{\mu}_a\big| < b_a\Big) \geqslant 1 - \delta = 1 - 2\,\exp\left(-2\,n_a\,b_a^2\right)$$

which is a *PAC statement* (*probably approximately correct*) about the bound $|\mu_a - \tilde{\mu}_a| < b_a$ which we are interested in

from here on out, it's algebra autopilot ...

we have

$$\delta = 2 \exp\left(-2\, n_a\, b_a^2\right)$$

$$\Leftrightarrow \qquad \ln \delta/2 = -2\, n_a\, b_a^2$$

$$\Leftrightarrow \qquad \frac{1}{2\, n_a} \ln 2/\delta = b_a^2$$

$$\Leftrightarrow \qquad b_a = \sqrt{\frac{\ln 2/\delta}{2\, n_a}}$$

if we finally substitute

$$\delta = \frac{2}{t^2}$$

our bound becomes

$$b_a(t) = \sqrt{\frac{\ln t}{n_a}}$$

to show that substituting $\delta = 2/t^2$ is “legal” needs work beyond the scope of this lecture

note

"one can show" that the *optimistic strategy*

$$a = \underset{a'}{\text{argmax}}\ UCB(a')$$

will identify a_* in the limit $T \to \infty$

"one can show" the expected regret $\mathbb{E}[\rho]$ after T rounds of UCB selection to be

$$O\left(\sqrt{K\,T\,\log T}\right)$$

for the dirty details, ask our theoretical computer scientists ;-)

take home messages

the problem of selecting nodes for expansion in Monte Carlo tree search is a multi-armed bandit problem

$\Rightarrow$ appropriate strategies (UCB or ϵ-greedy) should be used in MCTS practice

multi-armed bandits are (still) the topic of active and ongoing research in (theoretical) machine learning and decision making

there are numerous, rigorous theoretical results on optimal strategies and regret rates which are of considerable economic interest

portfolio optimization, resource allocation, scheduling, experimental design, (online) ad placing, ad auctioning, search result diversification, ...

summary

we now know about

indicator functions

Monte Carlo methods and their use

Monte Carlo tree search and its favorable properties

the exploration / **exploitation dilemma** and how to deal with it

one more thing . . .

we now know that

probabilities and *expectations* can assist problem solving or guide decision making