ucc
Coláiste na hOllscoile Corcaigh, Éire
University College Cork, Ireland

CS4618 Artificial Intelligence I

Today: Collecting Coupons
Assessment of
Randomised Search Heuristics

Thomas Jansen

October 31st

Plans for Today

Collecting Coupons

Consider Collecting coupons, n different types of coupons

Assumption $\forall i \in \{1, 2, \ldots, n\}$:
each time, independently, $\mathrm{Prob}\,(\text{get coupon type } i) = \frac{1}{n}$

How many coupons do we need until we have a complete collection?

Observation this number is random variable T

Questions What is $\mathrm{E}\,(T)$?
How likely are deviations from $\mathrm{E}\,(T)$?

A Detour: Landau Notation

Observation being less exact can simplify things a lot

Idea Characterise functions according to their "most important term"

Different point of view Characterise functions according to their "order of growth"

Remember for comparing numbers $\leq, \geq, =, <, >$

Define comparisons for functions similarly

Landau Notation

Definition

For $f, g \colon \mathbb{N}_0 \to \mathbb{R}^+$ define

- $f = O(g) \quad :\Leftrightarrow \exists n_0, c > 0 \colon \forall n \geq n_0 \colon f(n) \leq c \cdot g(n)$
  f grows not faster than g "$\leq$"
- $f = \Omega(g) \quad :\Leftrightarrow g = O(f)$
  f grows not slower than g "$\geq$"
- $f = \Theta(g) \quad :\Leftrightarrow f = O(g) \wedge f = \Omega(g)$
  f and g grow equally fast "$=$"
- $f = o(g) \quad :\Leftrightarrow \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$
  f grows slower than g "$<$"
- $f = \omega(g) \quad :\Leftrightarrow g = o(f)$
  f grows faster than g "$>$"

The Coupon Collector's Problem

Theorem (Coupon Collector's Theorem)

Let T denote the number of coupons obtained until all n types of coupons are present for the first time.

1. $E(T) = n \ln n + O(n)$
2. $\forall \beta > 1 \colon Prob(T > \beta n \ln n) \leq n^{-(\beta-1)}$
3. $\forall c \in \mathbb{R} \colon Prob(T > n \ln n + cn) = 1 - e^{-e^{-c}}$

Proof

Partition random process into disjoint phases
in phase i, exactly i different types have been obtained
length of phase i is T_i

Observation $$T = \sum_{i=0}^{n-1} T_i \Rightarrow \mathrm{E}(T) = \sum_{i=0}^{n-1} \mathrm{E}(T_i)$$

Phases in Obtaining Coupons

in phase i $\quad \mathrm{Prob}\,(\text{obtain } (i+1)\text{-th type}) = (n-i)/n$

thus $\quad \mathrm{E}\,(T_i) = \frac{n}{n-i}$

$$\Rightarrow \quad \mathrm{E}\,(T) = \sum_{i=0}^{n-1} \frac{n}{n-i}$$
$$= n \sum_{i=1}^{n} \frac{1}{i} = n \ln n + O(n)$$ ✓

Proof of the 2nd Statement

need to prove $\forall \beta > 1 \colon \mathrm{Prob}\,(T > \beta n \ln n) \leq n^{-(\beta-1)}$

Define event $S_{i,t}$
type i not obtained within first t coupons

Observation $\mathrm{Prob}\,(S_{i,t}) = \left(1 - \frac{1}{n}\right)^t \leq e^{-t/n}$

Observation $\forall t \colon \mathrm{Prob}\,(T > t) = \mathrm{Prob}\left(\bigcup_{i=1}^{n} S_{i,t}\right)$

estimate $\mathrm{Prob}\left(\bigcup_{i=1}^{n} S_{i,t}\right) \leq \sum_{i=1}^{n} \mathrm{Prob}\,(S_{i,t}) \leq n \cdot e^{-t/n}$

choose $t = \beta n \ln n$

$\mathrm{Prob}\,(T > \beta n \ln n) \leq n \cdot e^{-\beta \ln n} = n^{-(\beta-1)}$ ✓

Proof of the 3rd Statement

need to prove $\forall c \in \mathbb{R}\colon \operatorname{Prob}(T > n \ln n + cn) = 1 - e^{-e^{-c}}$

we know $\forall t\colon \operatorname{Prob}(T > t) = \operatorname{Prob}\left(\bigcup_{i=1}^{n} S_{i,t}\right)$

principle of inclusion and exclusion

$$\operatorname{Prob}\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \operatorname{Prob}(A_i) - \sum_{1 \le i < j \le n} \operatorname{Prob}(A_i \cap A_j)$$
$$+ \sum_{1 \le i < j < k \le n} \operatorname{Prob}(A_i \cap A_j \cap A_k) + \cdots$$
$$+ (-1)^{n+1} \sum_{1 \le h_1 < h_2 < \cdots < h_n \le n} \operatorname{Prob}\left(\bigcap_{j=1}^{n} A_{h_j}\right)$$

Boole-Bonferroni Inequalities

for even k $$\text{Prob}\left(\bigcup_{i=1}^{n} A_i\right) \geq \sum_{i=1}^{k}(-1)^{i+1} \sum_{1\leq h_1<h_2<\cdots h_i\leq n} \text{Prob}\left(\bigcap_{j=1}^{i} A_{h_j}\right)$$

for odd k $$\text{Prob}\left(\bigcup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{k}(-1)^{i+1} \sum_{1\leq h_1<h_2<\cdots h_i\leq n} \text{Prob}\left(\bigcap_{j=1}^{i} A_{h_j}\right)$$

Notation $$P_l = \sum_{1\leq h_1<h_2<\cdots h_l\leq n} \text{Prob}\left(\bigcap_{j=1}^{l} A_{h_j}\right)$$

Application of Boole-Bonferroni Inequalities

$$\forall t\colon \operatorname{Prob}(T > t) = \operatorname{Prob}\left(\bigcup_{i=1}^{n} S_{i,t}\right)$$

$$\operatorname{Prob}(T > n \ln n + cn) = \sum_{i=1}^{n} (-1)^{i+1} P_i$$

$n \in \{2k, 2k+1\}$

$$\sum_{i=1}^{2k} (-1)^{i+1} P_i \leq \operatorname{Prob}(T > n \ln n + cn) \leq \sum_{i=1}^{2k+1} (-1)^{i+1} P_i$$

from structure of problem estimating P_i

Estimating P_i

$$P_l = \sum_{1 \leq h_1 < h_2 < \cdots h_l \leq n} \text{Prob}\left(\bigcap_{j=1}^{l} A_{h_j}\right)$$

$$\overset{\text{coupons homogeneous}}{=} \binom{n}{l} \text{Prob}\left(\bigcap_{j=1}^{l} A_j\right)$$

structure of problem $A_j = S_{j,n\ln n+cn}$

Observation $\text{Prob}\left(\bigcap_{j=1}^{l} A_j\right) = \left(1 - \frac{l}{n}\right)^{n\ln n+cn}$

thus $P_l = \binom{n}{l}\left(1 - \frac{l}{n}\right)^{n\ln n+cn}$

Detour: Calculation

$$k! \cdot \frac{1}{n^k} \cdot \binom{n}{k} = \frac{n!}{n^k(n-k)!} = \frac{n}{n} \cdot \frac{n-1}{n} \cdots \frac{n-k+1}{n}$$

thus

$$\lim_{n\to\infty} k! \cdot \binom{n}{k} \cdot \left(1 - \frac{k}{n}\right)^{n \ln n + cn} = \lim_{n\to\infty} n^k \left(1 - \frac{k}{n}\right)^{(n/k)k(\ln n + c)}$$

$$= \lim_{n\to\infty} n^k e^{-k \ln n - ck} = e^{-ck}$$

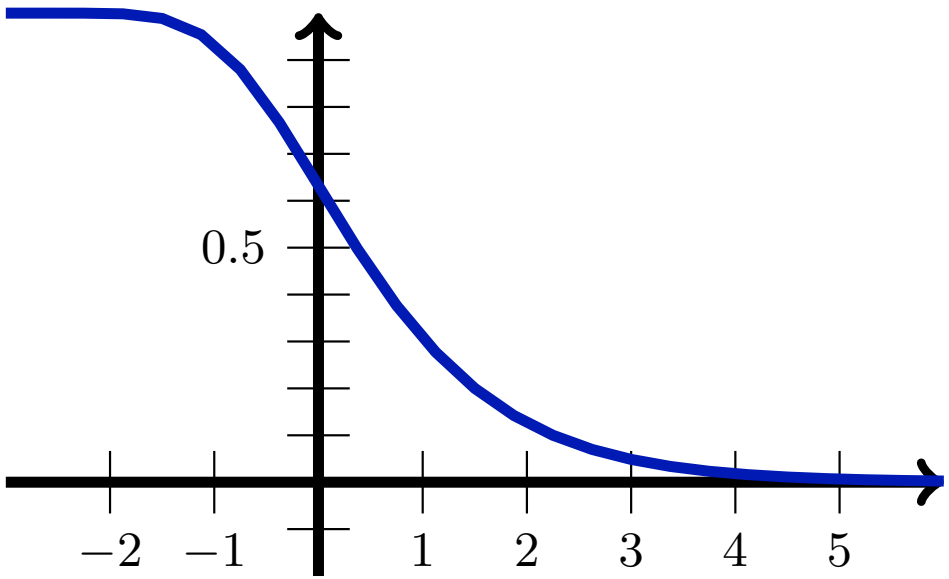
Completing the Proof

We know
$$\lim_{n\to\infty} \text{Prob}\left(T > n\ln n + cn\right) = \lim_{n\to\infty} \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} \left(1 - \frac{i}{n}\right)^{n\ln n + cn}$$

and
$$\lim_{n\to\infty} k! \cdot \binom{n}{k} \cdot \left(1 - \frac{k}{n}\right)^{n\ln n + cn} = e^{-ck}$$

thus $\lim_{n\to\infty} \text{Prob}\left(T > n\ln n + cn\right) = \lim_{n\to\infty} (-1)^{i+1} \frac{e^{-ci}}{i!} = 1 - e^{-e^{-c}}$

(Taylor approximation for $1 - e^{-e^{-c}}$) □



sharp threshold resultat

Randomised Search Heuristics

Remember we have

- local search (first improvement, best improvement, random)
- Metropolis algorithm
- simulated annealing
- evolutionary algorithms
  - (1+1) EA
  - (μ+λ) EA
  - (μ, λ) EA
  - steady state GA
  - ...
- ...

Which randomised search heuristic is the best?

We consider optimisation of unknown function $f \in \mathcal{F}$
with $\mathcal{F} \subseteq \{g \colon S \to R\}$ known (S finite, $R \subseteq \mathbb{R}$ finite)
and black-box algorithms for $\mathcal{F}$,
optimising each $f \in \mathcal{F}$ at least eventually

A best randomised search heuristic?

Remember we have

- local search (first improvement, best improvement, random)
- Metropolis algorithm
- simulated annealing
- evolutionary algorithms
  - (1+1) EA
  - (μ+λ) EA
  - (μ, λ) EA
  - steady state GA
  - ...
- ...

Which randomised search heuristic is the best?

NFL Theorem (1996)

For any finite S and R, on average, all black-box algorithms for $\mathcal{F} = R^S$ make an equal number of distinct f evaluations until an arbitrary optimisation goal is reached.

Dealing with the NFL

NFL Theorem (1996)

For any finite S and R, on average, all black-box algorithms for $\mathcal{F} = R^S$ make an equal number of distinct f evaluations until an arbitrary optimisation goal is reached.

Is this surprising? perhaps

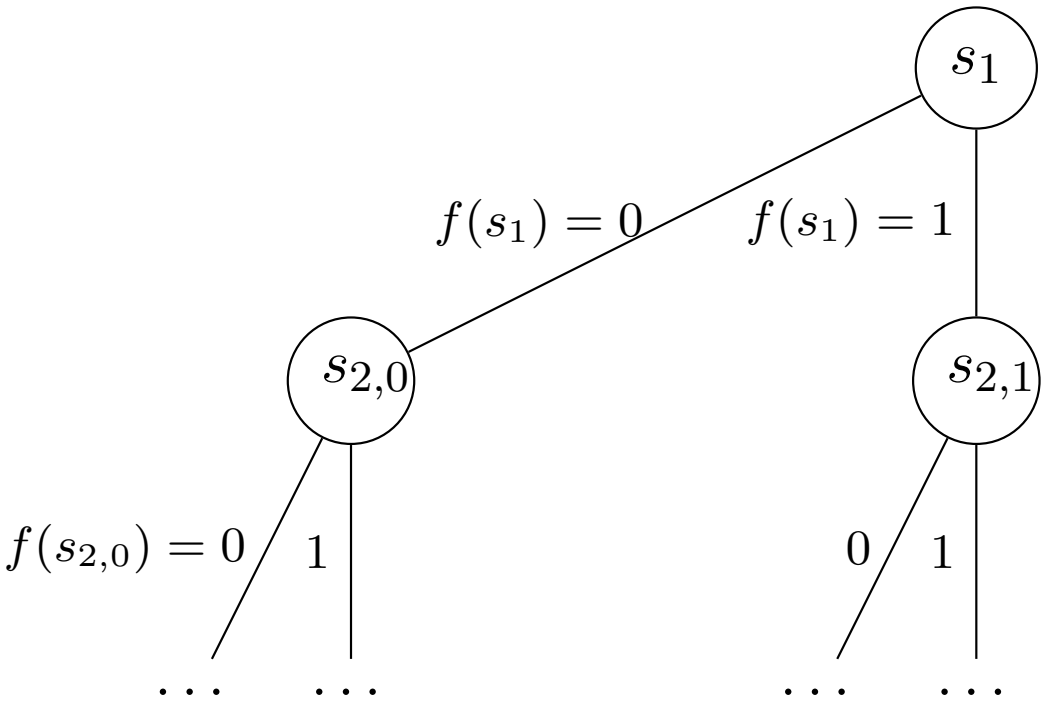
Is it true? see proof

Next

1. formalising deterministic black-box algorithms
2. formalising optimisation goals
3. proof for deterministic search heuristics
4. proof for randomised search heuristics
5. putting the NFL into perspective (part 1)
6. proof of a more general NFL
7. putting the NFL into perspective (part 2)

Describing Deterministic Black-Box Algorithms

Using the example $R = \{0, 1, 2\}$ for maximising $f : S \to R$



Observations

- black-box algorithm for $\mathcal{F} \subseteq R^S \mathrel{\widehat{=}} |R|$-ary tree with node labels from S
- non-repeating black-box algorithm for $\mathcal{F} \subseteq R^S$ has depth $\leq |S|$

On Non-Repeating Black-Box Algorithms

Is 'non-repeating' a strong assumption?

Observations

- 'non-repeating' easy to achieve using a dictionary
- NFL only counts f evaluations $\Rightarrow$ other computational cost (including dictionary use) for free
- Remember NFL counts distinct f evaluations
  $\rightsquigarrow$ 'non-repeating' equivalent to not counting repetitions

Is 'finite S and R' a strong assumption?

Observations

- given finite S, for any f finite R no restriction since $|f(S)| \leq |S|$
- finite S for any implementation of any f realistic

Optimisation with Deterministic Black-Box Algorithms

Remember deterministic black-box algorithm A
$\widehat{=} \leq |R| - 1$-ary tree

Simplifying Notation deterministic black-box algorithm A
$\widehat{=} |R|$-ary tree
Note no real change of A

Observation A on $f \widehat{=}$ path in tree
trace $T_A(f) = \langle (s_1, f(s_1)), (s_2, f(s_2)), \ldots, (s_{|S|}, f(s_{|S|})) \rangle$

Remark defining $T_A(f)$ up to $s_{|S|}$ is simplifying notation
without real change to A (as above)

Consider projection of $T_A(f)$ on function values
$V_A(f) = \langle f(s_1), f(s_2), \ldots, f(s_{|S|}) \rangle$

Optimisation Goals

Remember

- deterministic black-box algorithm $A \mathrel{\widehat{=}} |R|$-ary tree
- A on $f \mathrel{\widehat{=}}$ trace
  $T_A(f) = \langle (s_1, f(s_1)), (s_2, f(s_2)), \ldots, (s_{|S|}, f(s_{|S|})) \rangle$
- projection on function values
  $V_A(f) = \langle f(s_1), f(s_2), \ldots, f(s_{|S|}) \rangle$

Useful Notation

$T_A(f,t) =$ first t components of $T_A(f)$
$T_A(f,t) = \langle (s_1, f(s_1)), (s_2, f(s_2)), \ldots, (s_t, f(s_t)) \rangle$
$V_A(f,t) =$ first t components of $V_A(f)$
$V_A(f,t) = \langle f(s_1), f(s_2), \ldots, f(s_t) \rangle$

Definition (Optimisation Goal)

$M\colon \{V_A(f,t) \mid A, f, t\} \to \mathbb{R}$

Proving the NFL — Part 1

NFL Theorem (1996)

For any finite S and R, on average, all black-box algorithms for $\mathcal{F} = R^S$ make an equal number of distinct f evaluations until an arbitrary optimisation goal is reached.

Now proof for deterministic black-box algorithms

Need to prove For all deterministic black-box algorithms A, A' and all optimisation goals $M\colon \{V_A(f,t) \mid A, f, t\} \to \mathbb{R}$

$$\frac{\sum_{f \in R^S} M(V_A(f,|S|))}{|R^S|} = \frac{\sum_{f \in R^S} M(V_{A'}(f,|S|))}{|R^S|}$$

Sub-Claim 1 $\forall$det. BBA $A, f, g \in R^S$:
$V_A(f,|S|) = V_A(g,|S|) \Rightarrow f = g$

Proving Sub-Claim 1

Sub-Claim 1 $\forall$det. BBA $A, f, g \in R^S$:
$V_A(f, |S|) = V_A(g, |S|) \Rightarrow f = g$

Proof

Let A, f, g be fixed arbitrarily with $V_A(f, |S|) = V_A(g, |S|)$

Observation $T_A(f, 1) = T_A(g, 1)$ since $V_A(f, 1) = V_A(g, 1) = v_1$
and s_1 depends only on A

Observation $T_A(f, 2) = T_A(g, 2)$ since $V_A(f, 2) = V_A(g, 2)$
and s_2 depends only on A, s_1 and v_1

Observation in the same way $T_A(f, t) = T_A(g, t)$
for all $t \in \{1, 2, \ldots, |S|\}$

Thus $f = g$ ✓ □

Summary & Take Home Message

Things to remember

- coupon collector
- NFL: All search heuristics perform equal over the class of all functions.

Take Home Message

- Even simple problems can require some amount of analysis.
- Probability theory yields precise and useful answers to practically relevant problems.
- When making too general statements about randomised search heuristics the statements are either trivial or false.
- One needs to consider the class of optimisation problems to achieve good performance.