Advanced Research Intelligent Embedded Systems

Cognitive Cyber-physical Systems: from theory to practice

Ph.D. and Master Course

Proff. Cesare Alippi and Manuel Roveri

Politecnico di Milano, DEIB, Italy

Problem Complexity and Complexity Reduction

«All problems are *equal* but some problems are more *equal* than others»

- The computational complexity theory studies the intrinsic difficulty associated with the solution of a computable problem.
- The complexity of an algorithm is generally evaluated as the time execution and memory resources required by an abstract machine to execute it.
- example

Algorithm 1: Algorithm A: a simple algorithm computing the scalar product between two vectors

```
scalar_product = 0;
i = 0;
assign memory to vectors x and y and populate the content;
while i < n do
    scalar_product = scalar_product + x[i]y[i];
    i = i+1;
end
```

- Memory complexity: $M(A) = 2n+2$
- Computational complexity: $C(A) = (2n+2)T_a + (n+1)T_c + n(2T_+ + T_*)$
- Big Oh notation:

$$\cdots O(k^{-n}) \prec O(n^{-k}) \prec O(n^{-1}) \prec O(1) \prec \cdots$$

$$\prec \cdots O(\log n) \prec O(n) \prec O(n^k) \prec O(k^n) \prec \cdots$$

Algorithm 2: Algorithm B: a sequential scalar product computation

```
scalar_product = 0;
i = 0;
assign memory to scalars x and y;
while i < n do
    input x and y;
    scalar_product = scalar_product + xy;
    i = i+1;
end
```

P, NP-Complete and NP-Hard problems

- We say that a problem A belongs to class **P** if its computational time complexity is polynomial $O(n^k)$ with constant k. The problem is "tractable" or "easily solvable"…
  - Sequential and binary search
  - Bubblesort, quicksort, mergesort
- Some problems are more difficult (their complexity is not polynomial) but it is «easy» to *verify* if a candidate solution solves the problem or not.
- **NP** (nondeterministic polynomial time) is the class of problems for which, given a candidate solution we can verify in polynomial time if a solution solves the problem or not.

P, NP-Complete and NP-Hard problems

Classic example: the sum of subsets

- Given a numeric set,

  {-1, 5, 9, -4, 12, 9}

  - Determine the subset whose sum is equal to a given value, say 0.
  - How many subsets should I generate and consider?
- Then, the problem is not **P**. However, given the subsets
  - {-1, 5, -4} or {-1, 5, -4, 12, 9} it is «easy» to verify whether they satisfy the problem or not
  - The problem is **NP**: a solution is *easily verifiable* but the problem is not *easily solvable*

P, NP-Complete and NP-Hard problems

- Is **P=NP** ?
  - If the answer is «yes» then problems that are easily verifiable are also easily solvable. If not, some problems are more complex than others…
- **NP-complete** problems are the hardest problems in the NP classes. Even though we can verify in polytime if a given solution belongs to the class determination of a solution is a difficult task.
  - Traveling salesman problem
- It is believed by many scholars that **P≠NP** (but others believe that **P=NP**)…

P, NP-Complete and NP-Hard problems

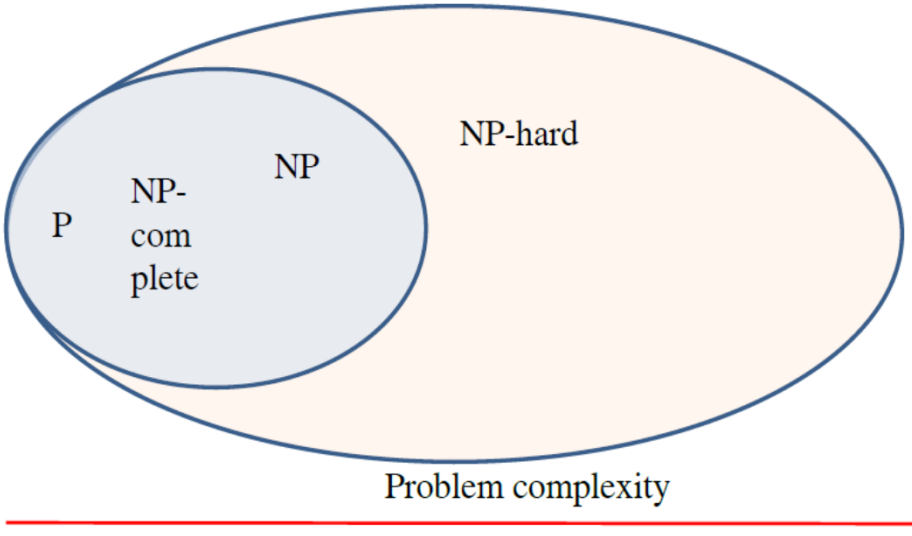
- **NP-hard:** A problem H is said to be NP-hard if and only if there exists a NP-complete problem L that is reducible to H in polynomial time. In other words, problem L can be solved in polynomial time by a machine which provides an oracle for H.
- A **NP**-hard problem does not need to have solutions verifiable in polynomial time and is, at least, as difficult as a **NP-complete** one.

P, NP-Complete and NP-Hard problems

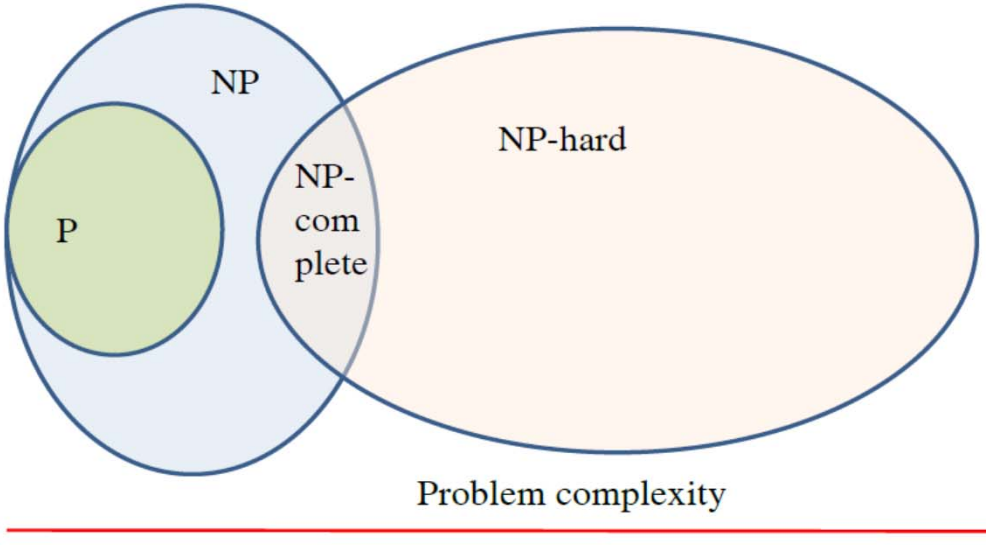
- The sum of subsets is a NP-hard problem
- Example: Given function $u(\psi) \in [0,1], \psi \in \Psi \subset \mathbb{R}^l$ and value $\gamma \in [0,1]$ does inequality $u(\psi) \leq \gamma$ hold for any value $\psi \in \Psi$ and function *u*?

- The problem is intractable for a generic *u* function since we should query the Oracle for each input ψ_i and ask the question: «is u(ψ_i) below γ?». Even in the case the Oracle provides a quick answer the number of queries is not polynomial for a continuous space.

P, NP, NP Complete, NP Hard



If **P=NP**



If **P≠ NP**

A deterministic vs. a probabilistic approach

- Is there any way to manage difficult problems?
- There are two main strategies:
  - You design ad-hoc heuristics aiming at solving the problem.
  - You accept that the solution is «mostly effective», namely the solution you have found solves the problem in probability
- The second approach has relevant implications worth further investigation
- In both cases we should ask ourselves how good the solution is…

A deterministic vs. a probabilistic approach

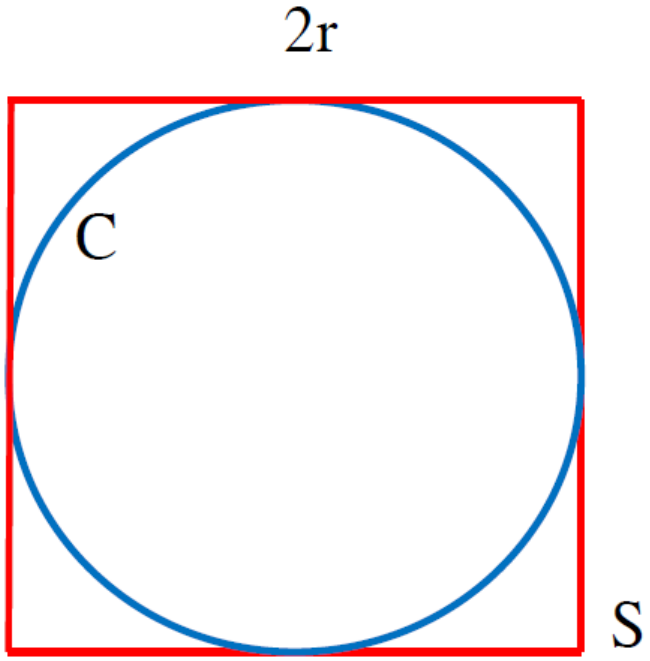
- Let's elaborate on the second strategy, i.e., we accept the problem to be solved in probabilistic terms and not pretend a deterministically correct solution.
- We obtain the the class of Randomized Polynomial time - RP- problems
- Example: Given function $u(\psi) \in [0,1], \psi \in \Psi \subset \mathbb{R}^l$ and value $\gamma \in [0,1]$ does inequality $u(\psi) \leq \gamma$ hold for any value $\psi \in \Psi$ for any function *u* in probability?
- If γ is given we have a performance verification problem
- If γ is not given and needs to be estimated we have a verification problem

Monte Carlo and Randomized Algorithms

«Go ahead sampling and you will catch something»

Monte Carlo

- Monte Carlo is a set of methods for addressing problems which can hardly be solved analytically for the mathematical complexity of the involved functions
- MC methods are based on «randomization»
- Imagine that we wish to estimate π



If Pr_C is the probability of extracting a point in C then $\pi=4Pr_C$

Estimating π with Monte Carlo

- Estract n samples from the square and count those falling inside the circle
- How to generate $\hat{p}_n = \frac{n_C}{n}$?
- Consider the indicator function I_C

$$I_C(s_i) = \begin{cases} 1 \text{ if } s_i \in C \\ 0 \text{ if } s_i \notin C \end{cases}$$

- Evaluate

$$\hat{p}_n = \frac{1}{n} \sum_{i=1}^{n} I_C(s_i)$$

And

$$\hat{\pi}_n = 4\hat{p}_n = \frac{4}{n} \sum_{i=1}^{n} I_C(s_i)$$

- How good is the estimate?

How good is the estimate?

- Weak law of large numbers

Let $x \in D$ be a continuous scalar random variable of finite expectation μ and finite variance σ_x^2 and $x_1, \cdots, x_n$ a set of n independent and identically distributed samples drawn from D (e.g., $D = \mathbb{R}$) according to the continuous probability density function f_D. Generate the empirical mean $\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} x_i$. Then, for any $\varepsilon \in D$, the weak law of large numbers guarantees that

$$\lim_{n \to +\infty} \Pr(|\hat{\mu}_n - \mu| \geq \varepsilon) = 0.$$

- How many samples n do we need to control the error? Well, it is application dependent and, as such, of little use…

How good is the estimate?

- Strong law of large numbers

Let $x \in D$ be a continuous random scalar variable of finite expectation μ and finite variance σ_x^2 and $x_1, \cdots, x_n$ a set of n independent and identically distributed samples drawn from D (e.g., $D = \mathbb{R}$) according to the continuous probability density function f_D. Generate the empirical mean $\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} x_i$. Then, the strong law of large numbers guarantees that relationship

$$\lim_{n \to +\infty} \hat{\mu}_n = \mu$$

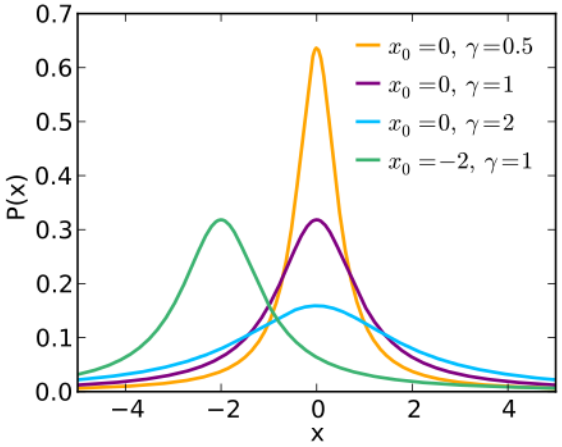
holds with probability one.

- How many samples n do we need to control the error? Well, it is application dependent and, as such, of little use…

Can I always use these laws?

- Yes,when assumptions hold. There are cases where they do not, e.g., the Cauchy density

$$f_x = \frac{1}{\pi(1+x^2)}$$



- Even though the function looks harmless its expectation and variance do not exist, e.g., the espected value

$$\int_{-\infty}^{+\infty} \frac{x}{\pi(1+x^2)} dx$$

- Bring home message: what appears to be intuitive does not necessarily need to be true…

Summing up: Monte Carlo

Algorithm 3: The Monte Carlo algorithm

1- Identify the input space D of the algorithm and a random variable s, with probability density function f_s over D;
2- Extract n samples $S_n = \{s_1, \cdots, s_n\}$ from D according to f_s;
3- Evaluate the algorithm on S_n;
4- Generate an estimate of the algorithm output.

- The input is modeled as a random variable to which is assigned a probability density function
- If a pdf is not available a uniform one can be considered for its worst case property
- Hard to define *n* for a given problem

Randomized algorithms

- Randomized algorithms are algorithms that, by sampling from a space according to a pdf provide results valid in probability
- Bounds on *n* can be given

Algorithm 5: The algorithm behind randomized algorithms

1- Transform the deterministic problem into a probabilistic problem;
2- Identify the input space Ψ of the algorithm and define a random variable ψ, with probability density function f_ψ over Ψ;
3- Identify the accuracy and the confidence levels and, then, the number of samples n required by the randomization process;
4- Draw n samples $S_n = \{s_1, \cdots, s_n\}$ from Ψ according to f_ψ;
5- Evaluate the algorithm on samples in S_n;
6- Provide the probabilistic outcome of the algorithm.

Lebesgue measurability

- Definition:

We say that a generic function $u(\psi), \psi \in \Psi \subseteq \mathbb{R}^l$ *is Lebesgue measurable with respect to* Ψ *when its generic step-function approximation* S_N *obtained by partitioning* Ψ *in* N *arbitrary domains grants that*

$$\lim_{N \to \infty} S_N = u(\psi)$$

holds on set $\Psi - \Omega$, $\Omega \subseteq \mathbb{R}^l$ *being a null measure set [20].*

- No functions generated by a finite-step, finite-time algorithm, such as any engineering-related mathematical computations, can be Lebesgue nonmeasurable!.

Example: The Algorithm performance verification problem

- The problem represents an interesting example of constraint satisfaction level. Clearly, the problem is NP-hard if we consider a generic performance function.
- Consider a given scalar γ and performance function *u()*. Ask whether

$$u(\psi) \leq \gamma, \forall \psi \in \Psi$$

- is satisfied or not. We might even ask at which level the inequality is satisfied, i.e., compute

$$n_{u(\psi)\leq\gamma} = \frac{\int_{u(\psi)\leq\gamma, \psi\in\Psi} d\psi}{\int_{\Psi} d\psi}.$$

The Algorithm performance verification problem

- Define

$$p(\gamma) = \frac{\int_{u(\psi)\leq\gamma,\psi\in\Psi} f_{\psi}(\psi)d\psi}{\int_{\Psi} f_{\psi}(\psi)d\psi} = \Pr\left(u(\psi) \leq \gamma\right), \forall \psi \in \Psi$$

- Note that

$$u(\psi) \leq \gamma$$

- is associated with the Bernoulli variable

$$\psi \in \Psi : I\left(u(\psi) \leq \gamma\right) = \begin{cases} 1 \text{ if } u(\psi) \leq \gamma \\ 0 \text{ if } u(\psi) > \gamma \end{cases}$$

- By extracting *n* sampling, generate

$$\hat{p}_n(\gamma) = \frac{1}{n}\sum_{i=1}^{n} I\left(u(\psi_i) \leq \gamma\right)$$

The Algorithm performance verification problem

- It can be proved that

$$\Pr\left(\left|\hat{p}_n(\gamma) - p(\gamma)\right| \leq \varepsilon\right) \geq 1 - \delta$$

- holds for any accuracy and confidence level in (0;1) provided that

$$n \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$$

samples are extracted

- The inequality is known as the Chernoff bound.

The Algorithm performance verification problem

Algorithm 6: Randomized algorithms for the algorithm performance verification problem: the given performance loss case $\overline{\gamma}$

1- The probabilistic problem requires evaluation of $p(\gamma) = \Pr(u(\psi) \leq \overline{\gamma})$ for a given $\overline{\gamma}$;
2- Identify the input space Ψ and a random variable ψ, with density function f_ψ over Ψ;
3- Select accuracy ε and confidence δ;
4- Draw $n \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$ samples $\psi_1, \cdots, \psi_n$ from ψ;
5- Estimate

$$\hat{p}_n(\overline{\gamma}) = \frac{1}{n} \sum_{i=1}^{n} I(u(\psi_i) \leq \overline{\gamma}), \qquad I(u(\psi_i) \leq \overline{\gamma}) = \begin{cases} 1 \text{ if } u(\psi_i) \leq \overline{\gamma} \\ 0 \text{ if } u(\psi_i) > \overline{\gamma} \end{cases}$$

6- use $\hat{p}_n(\overline{\gamma})$;

- Let's identify an error bound for π with Chernoff

Different Bounds

- The literature has presented different bounds, with the Chernoff being the least expensive one
- The Bernoulli bound
  - Consider the bernoullian variable *x* (*Pr(x)=1* is *p*)
  - Evaluate

$$\hat{E}_n = \frac{1}{n}\sum_{i=1}^{n} x_i$$

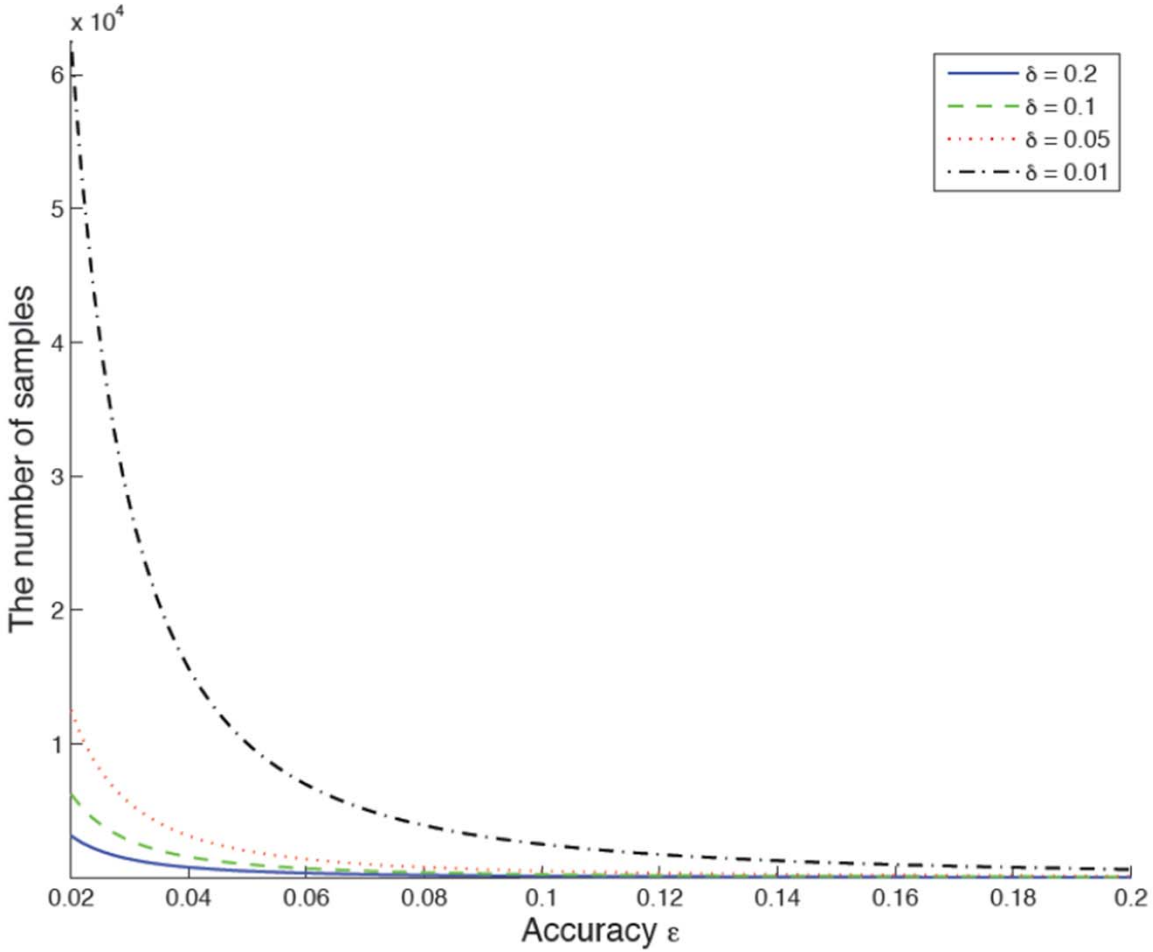
  - And recall that $E\left[\hat{E}_n\right] = p$. Then,

Inequality

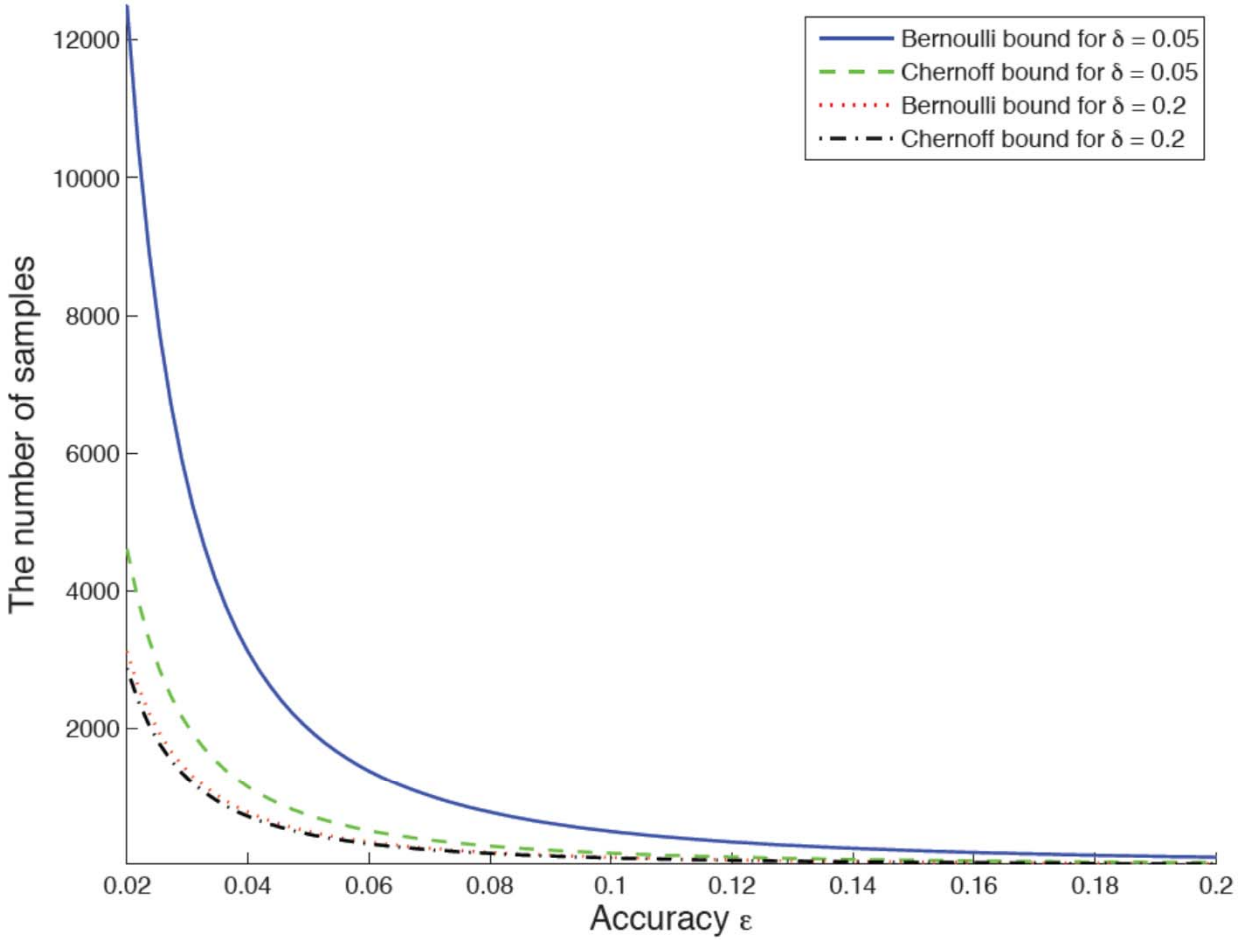
$$\Pr\left(|\hat{E}_n - E[\hat{E}_n]| < \varepsilon\right) > 1 - \delta$$

holds for any accuracy level $\varepsilon \in (0,1)$ *and confidence* $1-\delta, \delta \in (0,1)$ *provided that at least* $n \geq \frac{1}{4\delta\varepsilon^2}$ *independent and identically distributed samples are drawn.*

The Bernoulli bound



Bernoulli vs Chernoff bounds



Bound	$\varepsilon = 0.05, \delta = 0.02$	$\varepsilon = 0.05, \delta = 0.01$	$\varepsilon = 0.02, \delta = 0.01$	$\varepsilon = 0.01, \delta = 0.01$
Bernoulli	5000	10000	62500	250000
Chernoff	922	1060	6623	26492

The Algorithm verification problem

- We do not require anymore γ to be fixed and explore the level satisfaction space to generate a characteristic curve

Algorithm 7: Randomized algorithms for solving the algorithm verification problem

1- The probabilistic problem requires evaluation of $p(\gamma) = \Pr(u(\psi) \leq \gamma)$ for any γ belonging to a finite set of arbitrary γ values;
2- Identify the input space Ψ and and a random variable ψ, whith density function f_ψ over Ψ;
3- Select accuracy ε and confidence δ;
4- Identify the interested performance level set $\Gamma = \{\gamma_1, \cdots, \gamma_k\}$;
5- $\hat{p}_{n,\Gamma}(\gamma)$ = verification-problem$(\Psi, f_\psi, u(\psi), \Gamma, \varepsilon, \delta)$;
6- use $\hat{p}_{n,\Gamma}(\gamma)$;

function verification-problem$(\Psi, f_\psi, u(\psi), \Gamma, \varepsilon, \delta)$
Draw $n \geq \frac{1}{2\varepsilon^2} \ln \frac{2}{\delta}$ samples $\psi_1, \cdots, \psi_n$ from ψ;
For each $\gamma \in \Gamma$ estimate

$$\hat{p}_n(\gamma) = \frac{1}{n} \sum_{i=1}^{n} I(u(\psi_i) \leq \gamma), \qquad I(u(\psi_i) \leq \gamma) = \begin{cases} 1 \text{ if } u(\psi_i) \leq \gamma \\ 0 \text{ if } u(\psi_i) > \gamma \end{cases}$$

Group all $\hat{p}_n(\gamma)$s in vector $\hat{p}_{n,\Gamma}$;
Return $\hat{p}_{n,\Gamma}$

From a deterministic to a probabilistic framework

- Consider a Lebesgue measurable *u()* function and consider problem

$$u_{max} = \max_{\psi \in \Psi} u(\psi)$$

i.e.,

$$u(\psi) \leq u_{max}, \quad \forall \psi \in \Psi$$

A dual probabilistic problem can be formulated

$$\Pr\left(u(\psi) > \hat{u}_{max}\right) \leq \tau$$

And recall we are only assuming *u* to be Lebesgue measurable

From a deterministic to a probabilistic framework

Intuitively, we should sample the space and get the estimate for the maximum

$$\hat{u}_{max} = \max_{i=1,\ldots,n} u(\psi_i)$$

Both the weak and the strong laws of large numbers grant the estimate to converge to the true value

weak law of large numbers

$$\lim_{n \to +\infty} \Pr(u_{max} - \hat{u}_{max} \geq \varepsilon) = 0$$

strong law of large numbers

$$\lim_{n \to +\infty} \hat{u}_{max} = u_{max}$$

with probability one.

From a deterministic to a probabilistic framework

Unfortunately, the required number of points scales exponentially with the dimension of the search space.

To solve the problem we need to introduce a second level of probability.

It can be proved that

Inequality

$$\Pr\left(\Pr\left(u(\psi) > \hat{u}_{max}\right) \leq \varepsilon\right) \geq 1 - \delta$$

holds for any accuracy level $\varepsilon \in (0,1)$ *and confidence* $1-\delta, \delta \in (0,1)$ *provided that at least*

$$n \geq \frac{\ln \delta}{\ln(1-\varepsilon)} \tag{4.10}$$

independent and identically distributed samples are drawn.

From a deterministic to a probabilistic framework

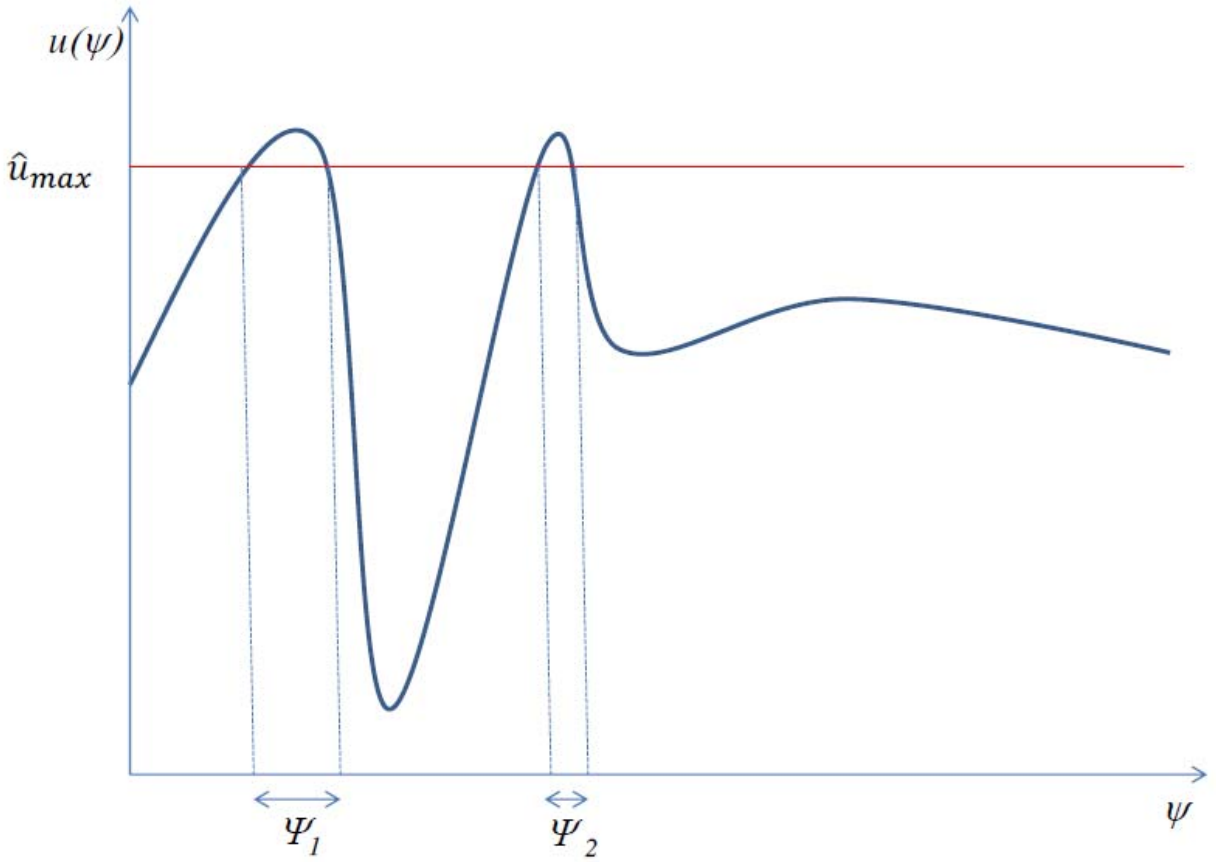


Fig. 4.8: The maximum estimated value for function $u(\psi)$ is $\hat{u}_{max}$. The probability of having points $u(\psi) \geq \hat{u}_{max}$ is associated with two supports Ψ_1, Ψ_2, for which $\Pr\left(u(\psi)|_{\psi\in\Psi_1} \geq \hat{u}_{max}\right) \leq \varepsilon_1$, $\Pr\left(u(\psi)|_{\psi\in\Psi_2} \geq \hat{u}_{max}\right) \leq \varepsilon_2$ and sum $\varepsilon_1 + \varepsilon_2 \leq \varepsilon$.