CS5314
Randomized Algorithms

Lecture 2: Events and Probability
(verifying polynomial identities)

Objectives

- This lecture will give a simple randomized algorithm for checking if two polynomials are equivalent
- Introduce two concepts:
  (1) Independent events
  (2) Conditional Probability
- Introduce two techniques:
  (1) Sampling with replacement,
  (2) Sampling without replacement

Verifying Polynomial Identities

- Suppose that we have written a program to multiply polynomials
- For example, given an input

  $$(x-1)(x-2)(x+3)(x-4)(x+5)$$

  our program outputs:

  $$x^5 + x^4 - 18x^3 + 36x^2 - 12x + 120$$

- How shall we check if this output is correct?

Verifying Polynomial Identities

In general, we are given two polynomials,

$F(x)$ in the product form ($\Pi_{i=1,2,\ldots,d}\,(x-a_i)$),

$G(x)$ in the canonical form ($\Sigma_{i=0,1,\ldots,d}\,c_i x^i$).

Our target is to check if

$$F(x) \equiv G(x)$$

Verifying Polynomial Identities

Method 1:

Convert $F(x)$ into the canonical form by repeatedly multiplying the i^{th} monomial to the product of the first i-1 monomials

- It takes $O(d^2)$ multiplications of coefficients. (No faster than the original computation)
- Also, there is a potential problem …

A Randomized Algorithm

Method 2:

- Let d be maximum degree of $F(x)$ or $G(x)$
- Pick an integer r, uniformly at random, from the set $[1,2,\ldots,100d]$
- Compute $F(r)$ and $G(r)$
- If $F(r) = G(r)$, we conclude $F(x) \equiv G(x)$. Otherwise, we conclude $F(x) \not\equiv G(x)$

A Randomized Algorithm
(Performance Analysis)

Questions

- What is the runtime of the algorithm?

  Ans. $O(d)$ time

- Will the algorithm always give a correct answer? If not, when?

  Ans. Not always. It will make an error when $F(x) \neq G(x)$, and the r we choose satisfies $F(r) = G(r)$

A Randomized Algorithm
(Performance Analysis)

Questions

- If $F(x) \not\equiv G(x)$, how many r can we choose so that $F(r) = G(r)$?

  Ans. At most d of them, since each such r is a root of the polynomial $F(x) - G(x)$, whose degree is at most d.

  (Fundamental theorem of algebra)

A Randomized Algorithm
(Performance Analysis)

Conclusion

- If $F(x) \neq G(x)$, the probability that the randomized algorithm returns an incorrect answer is at most $d/100d$
- Thus, in case $F(x) \neq G(x)$, the randomized algorithm is correct in at least 99% of the time

Can we further improve this probability?

A Randomized Algorithm
(Modification)

- One idea is to run the algorithm multiple times, say k times, so that it concludes $F(x) \equiv G(x)$ if all the r's we pick in these k different runs will satisfy $F(r) = G(r)$
- This is unlikely to happen if $F(x) \not\equiv G(x)$
- Can we bound the probability for this modified algorithm to return a correct conclusion?

Independent Events

Definition: Two events E_1 and E_2 are **independent** if and only if

$$\Pr(E_1 \cap E_2) = \Pr(E_1)\Pr(E_2)$$

Definition (general version): Events E_1, E_2, …, E_k are **mutually independent** if and only if for any subset $I \subseteq [1,k]$,

$$\Pr\Big(\bigcap_{i \in I} E_i\Big) = \prod_{i \in I} \Pr(E_i)$$

Independent Events

Example 1:

Experiment: Throw a fair die and flip a fair coin, observe the result

E_1 = the outcome of the die throw is 2

E_2 = the outcome of the coin flip is Tail

What are the probabilities

$$\Pr(E_1 \cap E_2),\ \Pr(E_1),\ \text{and } \Pr(E_2)\ ?$$

Independent Events

Remark:

- Usually, if “event E_1 occurs” does not affect “event E_2 to occur”, the two events are independent
- The converse may not be true

Independent Events

Example 2:

Experiment: Throw a fair die twice, observe the result

E_1 = the outcome of 1st throw is even

E_2 = the sum of the outcomes in the two throws is odd

What are the probabilities

$Pr(E_1 \cap E_2)$, $Pr(E_1)$, and $Pr(E_2)$?

Back to our algorithm…
(Performance Analysis)

- Suppose that $F(x) \not\equiv G(x)$
- Let E_i be the event that at the i^{th} run, the r we choose satisfies $F(r) = G(r)$
- The probability that the modified algorithm makes an error after k runs is

$$\Pr\left(\bigcap_{i=1,2,\ldots,k} E_i \right)$$

Note: The event $\bigcap_{i=1,2,\ldots,k} E_i$ corresponds to the case that for all r's chosen in the k runs, they all satisfy $F(r) = G(r)$.

Back to our algorithm…
(Performance Analysis)

- Since the choice of the integer r in each run is independent of the choice of in other runs, the events $E_1, E_2, \ldots, E_k$ are mutually independent
- So, $\Pr(\bigcap_{i=1,2,\ldots,k} E_i) = \prod_{i=1,2,\ldots,k} \Pr(E_i)$, which is at most $(1/100)^k$
- Thus, the probability that the algorithm is correct is at least $1 - (1/100)^k$

Back to our algorithm… (Remark)

- In the previous algorithm, the random choice r we make in each run is drawn from the same set, with the same distribution of probabilities
- This technique is called sampling with replacement
  - This corresponds to choosing an item from a set, and replace it back after choosing, so that the same item can be chosen next time

Our 2^{nd} algorithm… (Remark)

- We can also modify the algorithm so that for the k runs, any integer can be chosen at most once
- This technique is called sampling without replacement
- Intuitively, this increases the probability that the algorithm will make a correct conclusion

Can we bound this probability?

Conditional Probability

Definition: Suppose that $\Pr(E_2) > 0$. The **conditional probability** that event E_1 occurs given that E_2 occurs is the ratio

$$\Pr(E_1 \cap E_2) \;/\; \Pr(E_2).$$

Such a value is denoted by $\Pr(E_1 \mid E_2)$.

Lemma: Suppose that $\Pr(E_2) > 0$. Then

$$\Pr(E_1 \cap E_2) = \Pr(E_1 \mid E_2)\Pr(E_2).$$

Conditional Probability

Example 1:

Experiment: Given a bag that contains two red balls and two green balls. Draw the first ball from the bag (uniformly at random), and draw the second ball from the remaining balls (uniformly at random)

E = the first ball is red

F = the second ball is green

Conditional Probability

- What is $\Pr(F \mid E)$?
- By definition, it is equal to $\Pr(E \cap F)$ / $\Pr(E)$, which equals (4/12) / (6/12) = 2/3
- Intuitively, given that we know the first ball is red, the remaining balls for the second draw must be one red and two greens. So we expect that the probability $\Pr(F \mid E)$ should be 2/3, which turns out to be true

Conditional Probability

Example 2:

Experiment: Given a deck of cards (with 26 red cards and 26 black cards). Draw three cards from the deck, one by one, without replacement (uniformly at random)

E = the first two cards are red

F = the third card is black

- What is Pr(F | E)?

Back to our 2nd algorithm… (the one without replacement)

- Suppose that $F(x) \not\equiv G(x)$
- Let E_i be the event that at the i^{th} run, the integer r satisfies $F(r) = G(r)$
- The probability that the modified algorithm makes an error is still

$$\Pr\left(\bigcap_{i=1,2,\ldots,k} E_i \right)$$

Back to our 2nd algorithm…
(the one without replacement)

- The value $\Pr(\bigcap_{i=1,2,\dots,k} E_i)$ is equal to

$$\Pr(E_k \mid \bigcap_{i=1,2,\dots,k-1} E_i) \Pr(\bigcap_{i=1,2,\dots,k-1} E_i)$$

- Repeating this argument, the value will be equal to

$$\Pr(E_1) \times \Pr(E_2 \mid E_1) \times \Pr(E_3 \mid E_1 \cap E_2) \times$$

$$\dots \times \Pr(E_k \mid \bigcap_{i=1,2,\dots,k-1} E_i)$$

Back to our 2nd algorithm…
(the one without replacement)

- We can bound $\Pr(E_j \mid \bigcap_{i=1,2,\ldots,j-1} E_i)$ to be at most $(d - (j-1)) / (100d - (j-1))$.

- Thus, $\Pr(\bigcap_{i=1,2,\ldots,k} E_i)$ is at most

$$\prod_{j=1,2,\ldots,k} [(d - (j-1)) / (100d - (j-1))]$$

With replacement? Or without?

- Normally, sampling without replacement gives better (tighter) bounds
- However, sometimes, we will prefer sampling with replacement because
  - Algorithm is simpler, easier to code
  - Performance is easier to analyze