Homework #2

Simon Judd

September 25, 2024

Problem 1. (Sumcheck with tensor weights)

We consider an extension of the sumcheck problem where the summand is multiplied by weights that have a product structure. Specifically, given $n \cdot |H|$ field elements $\{\delta_{i,\alpha} \in F\}_{i\in[n],\alpha\in H}$, we consider statements of the following form:

$$\sum_{\alpha_1,\ldots,\alpha_n \in H} \delta_{1,\alpha_1} \cdots \delta_{n,\alpha_n} \cdot p(\alpha_1, \ldots, \alpha_n) = \gamma$$

Show that the sumcheck protocol can be extended to support the above statement, with the same completeness and soundness guarantees.

0.1 Proof

The key here is to represent δ_{n,α_n} as a low-degree polynomia l. For each $i \in [n]$ we construct a low-degree polynomial $\delta_i(x)$ over H. The size of this polynomial is bounded by $|H|$. We construct this polynomial using langrage interpolation:

$$\delta_i(x) = \sum_{\alpha\in H} \delta_{i,\alpha} \cdot L_a(x)$$

Where $L_a(x)$ is the lagrange basis polynomial for all $\alpha \in H$. This ensures that $\delta_i = \delta_{i,\alpha}$ for all $\alpha \in H$.

Next we define a new function f:

$$f(x_1, .., x_n) = (\prod_{i=1}^{n} \delta_i(x_i)) \cdot p(x_1, .., x_n)$$

This is a polynomial in n variables with a degree in each x_1 of:

$$deg_{xi}(f) = deg(\delta_i) + deg_{xi}(p) \leq |H| + deg_{xi}(p)$$

We now apply the sumcheck protocol to f.

For $n \in N$ rounds and for a claimed sum γ, we define the sum-check protocol as:

$$\sum_{\alpha_1,..,\alpha_n \in H} f(\alpha_1, .., \alpha_n) = \gamma$$

In each $1, ..n$-rounds, the prover first calculates a polynomial $f_1(x) = \sum_{\alpha_2,..,\alpha_n} f(x, \alpha_2, .., \alpha_n)$ and then sends the verifier $f_1 \in F[X]$. The verifier checks whether $\sum_{\alpha_1 \in H} f_1(\alpha_1) = \gamma$ and then responds with a random $w_1 \in F$.

In the subsequent rounds, the prover sends the following polynomial to the verifer.

$$f_2(x) = \sum_{\alpha_3,..,\alpha_n} f(w_1, x, \alpha_3, .., \alpha_n)$$

The verifier then checks $\sum_{\alpha_2 \in H} f_2(\alpha_2) = f_1(w_1)$. Finally at the end of the protocol, the verifier checks whether $f(w_1, .., w_n) = f_n(w_n)$.

- Completeness: If the prover is honest and the claimed sum γ is correct, the protocol will accept with probability 1.
- Soundness: If the prover is dishonest or γ is incorrect, the protocol will reject with high probability, depending on the field size and the degrees of the polynomials involved.

Problem 2. (Efficient multilinear extension)

The multilinear extension of a boolean function $f : \{0,1\}^n \to \{0,1\}$ over a field F is the unique multilinear polynomial $\mathrm{MLE}_F(f) \in F[X_1, \ldots, X_n]$ that agrees with f on $\{0,1\}^n$:

$$\mathrm{MLE}_F(f)(X_1, \ldots, X_n) := \sum_{b_1,\ldots,b_n \in \{0,1\}} f(b_1, \ldots, b_n) \prod_{i \in [n]: b_i = 1} X_i \prod_{i \in [n]: b_i = 0} (1 - X_i)$$

Prove that evaluating a multilinear extension at a single point is in linear time. Namely, give an algorithm that given a boolean function f (represented as a string of 2^n bits), finite field F, and evaluation point $(\alpha_1, \ldots, \alpha_n) \in F^n$, computes the evaluation of $\mathrm{MLE}_F(f)$ at $(\alpha_1, \ldots, \alpha_n)$ in $O(2^n)$ field operations.

Hint: The multilinear extension can be evaluated by summing term by term in $O(n \cdot 2^n)$ field operations while maintaining a state of $O(1)$ field elements in memory. How can you use more memory to speed up the computation?

0.2 Proof

The core idea here is that evaluating a multilinear extension is like traversing through the layers of a multidimensional hypercube.

Given a point $\alpha_1, \ldots, \alpha_n$ where $\alpha_i \in \{0, 1\}$, we evaluate each round of the multilinear extension as:

$$f(\alpha) = \sum_{S \subseteq N} f(1_S) \prod_{i \in S} \alpha_i \prod_{j \notin S} (1 - \alpha_j)$$

This requires iterating over all $|H|$ subsets, leading to $O(2^n)$ complexity. However, we can approach this problem recursively and run much more efficiently. The key thing to understand here is that each layer in a multilinear extension can be evaluated in linear time. Each linear interpolation follows the same formula $f(x) = a(1-a) + b \cdot x$ and requires only $O(n)$-work plus $O(1)$ field element for memory. Essentially, for each new variable or dimension, you are only adding one more layer of work.

Let's walk through the recursion. For the base case where $n = 1$, we have:

$$f(x_1) = f(0) \cdot (1 - x_1) + f(1) \cdot x_1$$

And for larger values in n, we have:

$$\mathrm{MLE}_F(f)(x_1, \ldots, x_n) = (1-x_n) \cdot \mathrm{MLE}_F(f_{x_n=0})(x_1, .., x_{n-1}) + x_n \cdot \mathrm{MLE}_F(f_{x_n=1})(x_1, .., x_{n-1})$$

where $f_{x_n=0}$ is the multilinear extension over n-variables when $x_n = 0$, and $f_{x_n=1}$ is the multilinear extension over n-variables when $x_n = 1$.

Now, let's look at how we can use memory to speed up this computation. Let's create a lookup function to store the results of each subproblem $cache(x_1, x_2, \ldots, x_k)$ where $k \leq n$.

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} cache(x_1, \ldots, x_n), & \text{if cached} \\ (1 - x_n) \cdot \mathrm{MLE}_F(f_{x_n=0})(x_1, .., x_{n-1}) + x_n \cdot \mathrm{MLE}_F(f_{x_n=1})(x_1, .., x_{n-1}), & \text{otherwise} \end{cases}$$

Once $f(x_1, \ldots, x_n)$ is computed, it's cached for future use:

$$cache(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$$

This incurs a memory-space cost of $O(2^n)$, but reduces the runtime.

Problem 3. (Efficient sumcheck)

We analyze the running time of the honest prover in the sumcheck protocol, when proving statements of the form $\sum_{\alpha_1, \ldots, \alpha_n \in H} p(\alpha_1, \ldots, \alpha_n) = \gamma$.

1. Prove that the honest prover can be realized in $O(d \cdot |H|^n \cdot |p|)$ operations, where d is the individual degree of p and $|p|$ is the number of operations to evaluate p at any point in F^n.

2. Consider the special case where $H = \{0, 1\}$ and p is multilinear. Prove that, if p is specified via its evaluations on $\{0, 1\}^n$, the honest prover can be realized in $O(2^n)$ field operations.

0.3 Proof

For each round, the prover calculates an n-variate polynomial p of degree d and evaluates it on $|H|^n$ total boolean inputs. The cost of each evaluation is $|p|$. And this is repeated for each round.

A polynomial is linear if it has a degree of at most $d = 1$. The cost to evaluate a linear polynomial is $|p| = 1$. Thus the total prover cost for evaluating a linear polynomial is $O(1 \cdot 2^n \cdot 1)$ field operations. Which simplifies to $O(2^n)$.

Problem 4. (Simple Formulas)

We say that a fully quantified boolean formula is simple if every occurrence of every variable is separated from its quantification point by at most one universal quantifier ($\forall$) and arbitrarily many other symbols.

- This is a simple formula: $\forall x_1 \forall x_2 \exists x_3((x_1 \vee x_2) \wedge x_3)$.
  What is its value?

  First we re-write it this arithmetically as:

  $$A = \prod_{x_1 \in \{0,1\}} \prod_{x_2 \in \{0,1\}} \sum_{x_3 \in \{0,1\}} ((x_1 + x_2) \cdot x_3)$$

  Evaluate the sum over x_3

  When $x_3 = 0 : ((x_1 + x_2) \cdot 0) = 0$

  When $x_3 = 1 : ((x_1 + x_2) \cdot 1) = x_1 + x_2$

  This simplifies to:

  $$A = \prod_{x_1 \in \{0,1\}} \prod_{x_2 \in \{0,1\}} (x_1 + x_2)$$

  Which evaluates to:

  $x_1 = 0, x_2 = 0 : 0 + 0 = 0$

  $x_1 = 1, x_2 = 0 : 1 + 0 = 1$

  $x_1 = 0, x_2 = 1 : 0 + 1 = 1$

  $x_1 = 1, x_2 = 1 : 1 + 1 = 2$

  The product of all these terms is:

  $$A = 0 \cdot 1 \cdot 1 \cdot 2 = 0$$

- This formula is not simple: $\exists x_1 \forall x_2((x_1 \vee x_2) \wedge \forall x_3(x_1 \vee x_3))$.
  What is its value?

  First we re-write it this arithmetically as:

$$A = \sum_{x_1 \in \{0,1\}} \prod_{x_2 \in \{0,1\}} ((x_1 + x_2) \cdot \prod_{x_3 \in \{0,1\}} (x_1 + x_3))$$

Evaluate the sum over x_1

When $x_1 = 0 : (0 + x_2) \cdot (0 + x_3) = x_2x_3$

When $x_1 = 1 : (1 + x_2) \cdot (1 + x_3) = 1 + x_2 + x_3 + x_2x_3$

Which simplifies to $1 + x_2 + x_3 + 2x_2x_3$

Now lets evalaute for each possible value:

$x_2 = 0, x_3 = 0 : 1 + 0 + 0 + 0 = 1$

$x_2 = 1, x_3 = 0 : 1 + 1 + 0 + 0 = 2$

$x_2 = 0, x_3 = 1 : 1 + 0 + 1 + 0 = 2$

$x_2 = 1, x_3 = 1 : 1 + 1 + 1 + 2 = 4$

Which sums to $A = 9$.

We denote by TSQBF language obtained by considering only fully quantified boolean formulas that are simple.

Problem 5.

Which of the following fully quantified boolean formulas are simple? What is their value?

1. $\forall x_1 \forall x_2 \exists x_3 ((x_1 \wedge x_2 \wedge x_3) \wedge \forall x_4 (\neg x_1 \wedge x_4))$

   Simple.

   $$A = \prod_{x_1 \in \{0,1\}} \prod_{x_2 \in \{0,1\}} \sum_{x_3 \in \{0,1\}} ((x_1 \cdot x_2 \cdot x_3) \cdot \prod_{x_4 \in \{0,1\}} ((1 - x_1 \cdot x_4))$$

   If $x_1, x_2, x_4 = 0$ then $A = 0$.

   Which enables us to simplify to:

   $$A = ((1 \cdot 1 \cdot x_2) \cdot ((1 - 1 \cdot 1)))$$

   Resulting in a value of $A = 0$

2. $\forall x_1 \forall x_2 \exists x_3 ((x_1 \vee x_3) \wedge \forall x_4 (x_3 \vee (x_2 \wedge x_4)))$

   Not Simple

   Value = 1

3. $\forall x_1 (\exists x_2 \forall x_3 (x_1 \vee x_2 \vee \neg x_3) \wedge \forall x_4 (\neg x_1 \wedge x_4))$

   Simple

   Value = 0

4. $\forall x_1(\exists x_2 \forall x_3(x_1 \vee x_2 \vee \neg x_3) \wedge \exists x_4(\neg x_1 \wedge x_4))$

   Simple

   Value $= 0$

Problem 6.

In this question we prove that a fully quantified boolean formula can be efficiently transformed into a simple fully quantified boolean formula with the same value. The general idea is to define a fresh variable for each occurrence of each variable in the original formula.

Let Φ be a fully quantified boolean formula with variables $x_1, \ldots, x_n$. We define a new formula Ψ that has a variable for each universal quantifier crossed by each variable in Φ. For example, if x_1 crosses k universal quantifiers in Φ, then Ψ has variables $x_{1,1}, \ldots, x_{1,k}$.

1. Give a boolean formula which is true if and only if x_1 and x_2 are equal.

2. Let $\Phi = \exists x_1 \forall x_2((x_1 \vee x_2) \wedge \forall x_3(x_1 \vee x_3))$. By replacing the two occurrences of x_1 with $x_{1,1}$ and $x_{1,2}$, and adding constraints and quantifiers, obtain a simple formula Ψ that has the same value as Φ.

3. Give an efficient algorithm that transforms a QBF Φ into an equisatisfiable simple QBF Ψ.

4. Prove the algorithm's correctness.

Problem 7.

Outline an interactive proof for TSQBF. Hint: show that simple formulas have "nice" arithmetizations.

Proof

$TSQBF := \{\Phi | \Phi \text{is a fully quantified boolean formula which is simple and evalautes to true}\}$

Let's take an TSQBF problem of the form and arithmetize it:

$$\forall x_1 \exists x_2 \ \phi(x_1, x_2)$$

1. Replace Boolean variables x with with integer variables

2. Replace logical operators with arithmetic operators
   - $\vee$ (OR) becomes addition $(+)$
   - $\wedge$ (AND) becomes multiplication $(\times)$
   - $\neg$ (NOT) becomes $(1 - z)$

3. Replace quantifiers
   - ◦ $\forall$ becomes a product ($\prod$) over $\{0,1\}$
   - ◦ $\exists$ becomes a sum ($\sum$) over $\{0,1\}$

By applying these rules we end up with the resulting arithmetization:

$$A := \prod_{x_1 \in 0,1} \left(\sum_{x_2 \in 0,1} (x_1 + x_2 - x_1 x_2) \right)$$

Which evaluates to the integer $A = 1 \cdot 2 = 2$.

The prover will now attempt to convince the verifer that it knows the claimed value $a \in A\ (mod\ p)$. The verifier will send a random element $r_1 \in F_p$ to the prover. The prover computes the polynomial $f(x_1, x_2) = x_1 + x_2 - x_1 x_2$, replacing x_1 with r_1 and sends $f(r_{1,}, x_2)$ to the verifier.

The verifer computes the sum over x_2 for $x_1 = r_1$.

$$S_r = \sum_{x_2 \in \{0,1\}} f(r_{1,}, x_2)$$

Expecting that $A = S_0 \cdot S_1$.

- Completeness: If Φ true and the claimed sum is correct, the protocol will accept with probability 1.
- Soundness: If Φ is false, the protocol will reject with high probability, depending on the field size and the degrees of the polynomials involved.