Bulletproofs to Lasso

Rasmus Kirk Jakobsen

This document aims to take you from knowing about Bulletproofs to learning about Lasso, the primary construction used in Jolt.

2025-12-01

Contents

1. Introduction

This document generally assumes that you're familiar with Bulletproofs, specifically the Inner Product Arguments used. It also assumes basic familiarity with Interactive Arguments and Pedersen commitments. These concepts are well presented, somewhat less formal than the relevant original papers, but in an understandable manner in Adam Gibson's excellent write-up “From Zero (Knowledge) to Bulletproofs”[1].

2. Prerequisites

2.1. Multilinear Extensions

3. Sumcheck

$$H := \sum_{b_1 \in \mathbb{B}} \sum_{b_2 \in \mathbb{B}} \dots \sum_{b_v \in \mathbb{B}} g(b_1, \dots, b_v) \qquad (1)$$

Prover **Verifier**

Round 1

$$g_1(X) := \sum_{x_{2:v} \in \mathbb{B}^{v-1}} g(X, x_{2:v}) \xrightarrow{H,\, g_1(X)} H \stackrel{?}{=} g_1(0) + g_1(1)$$

$$\deg(g_1) \stackrel{?}{=} \deg_1(g)$$

$$\xleftarrow{r_1} r_1 \in_R \mathbb{F}$$

Round $j \in [2..d]$

$$g_j(X) := \sum_{x_{j+1:v} \in \mathbb{B}^{v-j}} g\left(r_{1:j-1}, X, x_{j+1:v}\right) \xrightarrow{g_j(X)} g_{j-1}\left(r_{j-1}\right) \stackrel{?}{=} g_j(0) + g_j(1)$$

$$\deg\left(g_j\right) \stackrel{?}{=} \deg_j(g)$$

$$\xleftarrow{r_j} r_j \in_R \mathbb{F}$$

Round v

$$g_v(X) := g(r_{1:v-1}, X) \xrightarrow{g_v(X)} g_{v-1}\left(r_{j-1}\right) \stackrel{?}{=} g_v(0) + g_v(1)$$

$$\deg(g_v) \stackrel{?}{=} \deg_v(g)$$

$$r_v \in_R \mathbb{F}$$

$$g_v(r_v) \stackrel{?}{=} g(r_{1:v})$$

4. GKR

Given a circuit $\mathcal{C}$, with d layers, n inputs and m outputs, a prover ($\mathcal{P}$) wishes to prove to a verifier ($\mathcal{V}$) a specific input $\boldsymbol{w} \in \mathbb{B}^n$ applied to $\mathcal{C}$ produces some output $\boldsymbol{o} \in \mathbb{B}^m$. To do this, we can leverage the sumcheck protocol, defined earlier.

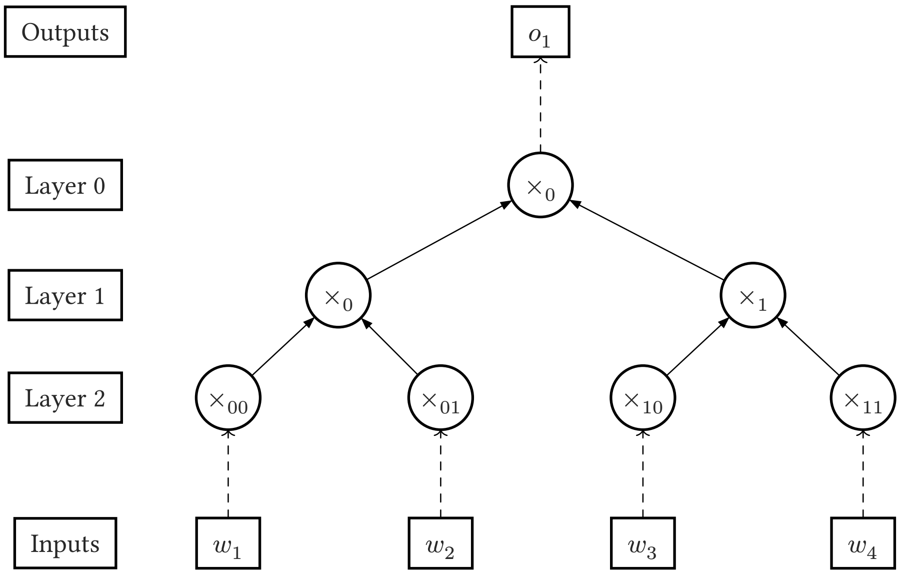

Figure 1: A layered arithmetic circuit for the computation $o_1 = \prod_{i=1}^{k} a_i$. Each node-label below represents the type of computation and the gate-label for the given layer, so $(\times_0)$ is a multiplication gate with label 0, for layer 0. Note that $d = 3$, $n = 4$, $m = 1$

To represent a layered circuit in a form amenable to the sum check protocol, we must first provide an adequate polynomial representation of the circuit. We start with the following three functions:

- $\text{add}_i(a, b, c) \in \mathbb{B}^{k_i + 2k_{i+1}} \to \mathbb{B}$ which outputs 1 if and only if gate a is an addition gate and b is the left input and c is the right input of gate a.
- $\text{mul}_i(a, b, c) \in \mathbb{B}^{k_i + 2k_{i+1}} \to \mathbb{B}$ which outputs 1 if and only if gate a is a multiplication gate and b is the left input and c is the right input of gate a.
- $W_i(a) \in \mathbb{B}^{k_i} \to \mathbb{B}$ which, given the gate-label a, outputs the value of node a.

Example 1 For Figure 1 $\text{add}_i, \text{mul}_i$ would evaluate to the following values:

$$\begin{array}{ll}
\text{mul}_0(0 \parallel 0 \parallel 0) = 1 & \text{mul}_1(0 \parallel 00 \parallel 01) = 1 \\
\text{mul}_0(0 \parallel 0 \parallel 1) = 1 & \text{mul}_1(1 \parallel 10 \parallel 11) = 1 \\
\text{mul}_0(\_ \parallel \_ \parallel \_) = 0 & \text{mul}_1(\_ \parallel \_ \parallel \_) = 0 \\
\text{add}_0(\_ \parallel \_ \parallel \_) = 0 & \text{add}_0(\_ \parallel \_ \parallel \_) = 0
\end{array}$$

We can use the multilinear extensions of add_i and mul_i to represent W_i in a form that lets us use sumcheck:

$$\widetilde{W}_i(a) = \sum_{b,c \in \mathbb{B}^{k_{i+1}}} \widetilde{\text{add}}_i(a,b,c)\Big(\widetilde{W}_{i+1}(b) + \widetilde{W}_{i+1}(c)\Big) + \widetilde{\text{mult}}_i(a,b,c) \cdot \widetilde{W}_{i+1}(b) \cdot \widetilde{W}_{i+1}(c) \tag{2}$$

Assume that the prover convinces the verifier that some polynomial $D(X_1, ..., X_\ell) = \widetilde{W}_0$, meaning that the above holds recursively all the way to layer d. Then the verifier can confirm that the evaluations of the circuit given input $\boldsymbol{w}$ evaluates to $\boldsymbol{o}$ by simply evaluating $D((X_1, ..., X_\ell))$ on all gate labels in depth zero. To prove this, the verifier chooses a random point r_0 and wishes to verify that $D(r_0) = W_0(r_0)$, which by Schwarz-Zippel means that $D(X_1, ..., X_\ell) = W_0(X_1, ..., X_\ell)$. Therefore, the prover and verifier apply sumcheck to the following polynomial:

$$\tilde{f}^{(0)}_{r_0}(b_0, c_0) = \widetilde{\text{add}}_0(r_0, b_0, c_0)\big(\widetilde{W}_1(b_0) + \widetilde{W}_1(c_0)\big) + \widetilde{\text{mult}}_0(r_0, b_0, c_0) \cdot \widetilde{W}_1(b_0) \cdot \widetilde{W}_1(c_0) \tag{3}$$

Which, if this succeeds, the verifier will be convinced that $D(X_1, ..., X_\ell) = W_0(X_1, ..., X_\ell)$ as desired. But in the final round of the sumcheck protocol, the verifier must be able to evaluate the above polynomial at a random point. The functions $\widetilde{\text{add}}_0$ and $\widetilde{\text{mul}}_0$ are part of the circuit description, and can thus be computed by the verifier without help from the prover. But the verifier also needs to evaluate $\widetilde{W}_1$ at two random points $b', c' \in_R \mathbb{F}$ corresponding to b_0, c_0. In principle, we could run the sumcheck protocol twice then, on the polynomials:

$$\begin{aligned}\tilde{f}^{(1)}_{b'}(b_1, c_1) &= \widetilde{\text{add}}_1(b', b_1, c_1)\big(\widetilde{W}_2(b_1) + \widetilde{W}_2(c_1)\big) + \widetilde{\text{mult}}_1(b', b_1, c_1) \cdot \widetilde{W}_2(b_1) \cdot \widetilde{W}_2(c_1) \\ \tilde{f}^{(1)}_{c'}(b_1, c_1) &= \widetilde{\text{add}}_1(c', b_1, c_1)\big(\widetilde{W}_2(b_1) + \widetilde{W}_2(c_1)\big) + \widetilde{\text{mult}}_1(c', b_1, c_1) \cdot \widetilde{W}_2(b_1) \cdot \widetilde{W}_2(c_1)\end{aligned} \tag{4}$$

But this would result in an exponential amount of sumchecks in the depth d. Instead, we can reduce two checks into one, using a linear combination.

4.1. Combining two claims to one

Suppose we were to apply sumcheck to the following polynomial instead:

$$\tilde{q}_1(b', c') = \widetilde{W}_1(b') + \alpha \cdot \widetilde{W}_1(c') \tag{5}$$

Which can be derived as:

$$\begin{aligned}\tilde{q}_1(b', c') = &\left(\sum_{b,c \in \mathbb{B}^{k_1}} \widetilde{\text{add}}_0(b', b, c)\big(\widetilde{W}_1(b) + \widetilde{W}_1(c)\big) + \widetilde{\text{mul}}_0(b', b, c) \cdot \widetilde{W}_1(b) \cdot \widetilde{W}_1(c)\right) + \\ &\alpha \cdot \left(\sum_{b,c \in \mathbb{B}^{k_1}} \widetilde{\text{add}}_0(c', b, c)\big(\widetilde{W}_1(b) + \widetilde{W}_1(c)\big) + \widetilde{\text{mul}}_0(c', b, c) \cdot \widetilde{W}_1(b) \cdot \widetilde{W}_1(c)\right) \\ = &\sum_{b,c \in \mathbb{B}^{k_1}} \widetilde{\text{add}}_0(b', b, c)\big(\widetilde{W}_1(b) + \widetilde{W}_1(c)\big) + \widetilde{\text{mul}}_0(b', b, c) \cdot \widetilde{W}_1(b) \cdot \widetilde{W}_1(c) + \\ &\alpha \cdot \widetilde{\text{add}}_0(c', b, c)\big(\widetilde{W}_1(b) + \widetilde{W}_1(c)\big) + \alpha \cdot \widetilde{\text{mul}}_0(c', b, c) \cdot \widetilde{W}_1(b) \cdot \widetilde{W}_1(c) \\ = &\sum_{b,c \in \mathbb{B}^{k_1}} \big(\widetilde{\text{add}}_0(b', b, c) + \alpha \cdot \widetilde{\text{add}}_0(c', b, c)\big)\big(\widetilde{W}_1(b) + \widetilde{W}_1(c)\big) \\ &\big(\widetilde{\text{mul}}_0(b', b, c) + \alpha \cdot \widetilde{\text{mul}}_0(c', b, c)\big)\big(\widetilde{W}_1(b) \cdot \widetilde{W}_1(c)\big)\end{aligned} \tag{6}$$

The below Lemma shows how this will help the prover-verifier pair in showing that $v_{b'} = \widetilde{W}_1(b') \land v_{b'} = \widetilde{W}_1(c')$, thus enabling the verifier to compute $\tilde{f}^{(0)}_{r_0}(b_0, c_0)$:

Lemma 4.1.1 For a polynomial $p(X_1, ..., X_k)$, if a prover wants to convince a verifier of two claims $v_1 = p(r_1), v_2 = p(r_2)$, then they can reduce this to a single claim over a polynomial $q(r_1, r_2)$:

$$q(X_1, .., X_{2k}) \coloneqq p(X_1, ..., X_k) + \alpha \cdot p(X_{k+1}, ..., X_{2k})$$

The verifier can then check that $q(r_1, r_2) = p(r_1) + \alpha \cdot p(r_2)$. The claim that $v_1 = p(r_1) \land v_2 = p(r_2)$ will then hold except with negligible probability, given that $q(X_1, ..., X_{2k})$ is defined as above.

Proof. If $q(r_1, r_2) = p(r_1) + \alpha \cdot p(r_2)$ but the claim does not hold, i.e. $v_1 \neq p(r_1), v_2 \neq p(r_2)$, then that means that the univariate non-zero polynomial:

$$e(X) = q(r_1, r_2) - p(r_1) + X \cdot p(r_2)$$

Evaluated to zero, which by the Schwarz-Zippel Lemma, has probability:

$$\Pr[e(\alpha) = 0 \mid e(X) \neq 0] = \frac{d}{|\mathbb{F}|}$$

In this case $d = 1$ which is negligible in the size of the field. □

In the GKR protocol, running sumcheck on Equation 6 convinces the verifier that $\tilde{Q}_i(b', c') = \widetilde{W}_i(b') + \alpha \cdot \widetilde{W}_i(c')$, which means that the verifier knows that $\tilde{Q}_i(X)$ is defined as in and they know the evaluation of $\tilde{Q}_i(X)$, $\tilde{Q}_i(b', c')$. The verifier can then verify that $v_{b'} = \widetilde{W}_i(b')$ and $v_{c'} = \widetilde{W}_i(c')$ by additionally checking that $\tilde{Q}_i(b', c') = v_{b'} + \alpha \cdot v_{c'}$.

With $v_{b'}$ and $v_{c'}$ the verifier can compute the evaluation of $\tilde{f}^{(0)}_{r_0}(b', c')$:

$$\tilde{f}^{(0)}_{r_0}(b', c') = \widetilde{\text{add}}_0(r_0, b', c')(v_{b'} + v_{c'}) + \widetilde{\text{mult}}_0(r_0, b', c') \cdot v_{b'} \cdot v_{c'} \tag{7}$$

It should already now be apparent that we can repeat this procedure, all the way to the input layer d.

4.2. Completing the protocol

In the input layer, the final check in the sumcheck protocol requires the verifier to evaluate the polynomial:

$$\tilde{f}^{(d-1)}_{r_{d-1}}(b', c') = \widetilde{\text{add}}_{d-1}(r_{d-1}, b', c')\left(\widetilde{W}_d(b') + \widetilde{W}_d(c')\right) + \widetilde{\text{mult}}_{d-1}(r_{d-1}, b', c') \cdot \widetilde{W}_d(b') \cdot \widetilde{W}_d(c') \tag{8}$$

The polynomials $\widetilde{\text{add}}_{d-1}$ and $\widetilde{\text{mul}}_{d-1}$ can be evaluated as usual. The polynomial $\widetilde{W}_d(b')$ corresponds to the values of the input layer $\boldsymbol{w}$. Since the verifier knows $\boldsymbol{w}$ they can compute the multilinear extension of $\boldsymbol{w}$ corresponding to $W_{d\left(X,\ldots,X_{k_d}\right)}$. From this the verifier can compute $\widetilde{W}_d(b'), \widetilde{W}_d(c')$ and thus the evaluation of $\tilde{f}^{(d-1)}_{r_{d-1}}(b', c')$.

The entire protocol can be seen below:

4.3. Efficiency

5. Spark

Before introducing Spark, we'll first introduce the primary argument that SPARK builds on, the **PLACEHOLDER**.

Bibliography

1. Gibson, A. (2022). From Zero (Knowledge) to Bulletproofs. (Accessed: 2025-01-29) Retrieved from https://github.com/AdamISZ/from0k2bp/blob/master/from0k2bp.pdf