LeanML: Formally Verified Linear Regression

Alisson Matheus Silva

1 Overview

As the final project for the course we aim to formally verify that the ordinary least squares (OLS) estimators for linear regression actually minimize the squared loss. We work in Lean 4 with the Mathlib library and prove optimality for two cases: regression through the origin ($y = ax$) and affine regression ($y = ax + b$). The idea is to go beyond just implementing the formulas and instead provide machine-checked proofs that they do what they claim to do.

2 Core Property

Regression through the origin

Given data points $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ with $S_{xx} = \sum x_i^2 \neq 0$, define the loss $L(a) = \sum(ax_i - y_i)^2$ and the OLS estimator $a^* = S_{xy}/S_{xx}$. We prove that for all $a \in \mathbb{R}$:

$$L(a^*) \leq L(a)$$

The proof works by decomposing the loss as

$$L(a) = L(a^*) + S_{xx}(a - a^*)^2$$

The cross term vanishes because of the normal equation, and the remaining term is nonneg, which gives us the result.

Affine regression

For $y = ax+b$, minimizing $L(a, b) = \sum(ax_i+b-y_i)^2$ gives the normal equations

$$\begin{pmatrix} S_{xx} & S_x \\ S_x & n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} S_{xy} \\ S_y \end{pmatrix}$$

which yield the closed-form solutions a^* and b^* when the determinant $\Delta = n \cdot S_{xx} - S_x^2 \neq 0$. The loss then decomposes as

$$L(a, b) = L(a^*, b^*) + \begin{pmatrix} a - a^* \\ b - b^* \end{pmatrix}^T \begin{pmatrix} S_{xx} & S_x \\ S_x & n \end{pmatrix} \begin{pmatrix} a - a^* \\ b - b^* \end{pmatrix}$$

which expands to

$$L(a, b) = L(a^*, b^*) + S_{xx}\, da^2 + 2\, S_x\, da\, db + n\, db^2$$

where $da = a - a^*$ and $db = b - b^*$. We prove this quadratic form is nonneg by showing it equals $\sum(da \cdot x_i + db)^2$, a sum of squares.

3 Challenges

Working with finite sums over `Fin` n in Lean turned out to be one of the hardest parts. Lemmas like `Finset.sum_add_distrib` and `Finset.mul_sum` expect a specific syntactic form for the sum, and there is a mismatch between $\sum_{i \in \texttt{univ}}$ and the $\sum_i$ notation that caused a lot of issues. Most of the proof effort went into carefully rewriting sums into a shape that these lemmas accept.

The algebra itself is not that hard, but getting Lean to see it requires combining `ring`, `field_simp`, and `linarith` in the right order. The affine case is also more involved because `field_simp` has to clear two denominators at once when verifying the normal equations.

Mathlib also updates frequently, so lemma names and tactic behavior can change between versions, which means proofs sometimes need maintenance after a version bump.

4 Repository

The code is at `https://github.com/itsalissonsilva/lean-ml`. Both the Origin and Affine proofs compile and type-check against the current Mathlib.