The Graphs of a Multiclass EBM Keep Its Predictions

A proof checked in Lean 4

Summary. A multiclass Explainable Boosting Machine (EBM) [1] gives one score per class to each bin of each term, and softmax turns the sum of the scores into probabilities. EBaMl shows these scores to an LLM as graphs. The code in `src/ebaml/graphs/ebm_graph.py` centres the scores of each bin over all the classes, and then it can lower all the scores by one constant. The text of the graph and the prompt tell the LLM that only the differences between the classes of a bin change the predicted probabilities. This paper proves that statement, and it proves that the graphs keep the predictions of the EBM. A Lean 4 file [2] checks every statement, and the appendix tells the reader how to repeat the check.

1 Claims and scope

The proof covers a multiclass EBM: a model whose score for each class is an intercept plus one term score for each term, and whose probabilities are the softmax of these scores.

Claim 1 (only the differences count). Two score vectors give the same probabilities if and only if they differ by one constant for all the classes. Thus the probabilities depend only on the differences between the scores of the classes.

Claim 2 (same mean). In every bin of every graph that `EBMGraph.from_ebm` reads from a multiclass EBM, the scores have the same mean over all the classes of the model. The mean is zero when the normalization is `NONE`.

Claim 3 (same differences). In every bin, the graph keeps the difference between the scores of two classes.

Claim 4 (same predictions). The model with the scores of the graphs predicts the same probabilities as the EBM, for every instance.

The claims are about the mean scores of the graphs. Section 5 lists what the proof does not cover.

2 Model

This section fixes the model and the glossary. The glossary gives one term to each concept, and the text uses this term everywhere.

Glossary.

Term	Meaning
class	An element k of the finite, non-empty set K of the classes of the model.
term	An element t of the finite set T. A term is a feature or an interaction.
bin	A value class of a term. Term t has the set of bins B_t.
score	The number $s_t(v, k)$ that bin v of term t adds to class k.
instance	The input to explain. It falls in the bin $c_t \in B_t$ of each term t.
shift	A function d that gives one number $d_t(v)$ to each bin. It lowers the scores of all the classes of the bin by that number.
graph	The scores that `EBMGraph.from_ebm` shows for a bin.

Notation 1. (data). The finite, non-empty set K of classes and the finite set T of terms. For each term t: the set B_t, the score $s_t : B_t \times K \to \mathbb{R}$ and the bin $c_t \in B_t$ of the instance. The intercept $b : K \to \mathbb{R}$.

Definition 1. (softmax). For a score vector $x : K \to \mathbb{R}$, the softmax is $\sigma(x)_k = e^{x_k} / \sum_{j \in K} e^{x_j}$.

Lean: `Softmax.softmax`, in `EBaMlProofs/MulticlassScores/Softmax.lean`.

Definition 2. (prediction). The score of class k is $L_k = b_k + \sum_{t \in T} s_t(c_t, k)$. The prediction is $\sigma(L)$.

Lean: `Model.score`, `Model.predict`, in `EBaMlProofs/MulticlassScores/Model.lean`.

Definition 3. (shift and graph). For a shift d, the shifted model has the scores $s_t(v, k) - d_t(v)$, the same intercept and the same instance. The mean of bin v of term t is $\mu_t(v) = \frac{1}{|K|} \sum_{j \in K} s_t(v, j)$. For a number a, the graph is the shifted model with $d_t(v) = \mu_t(v) + a$.

Lean: `Model.shift`, `Model.mean`, `Model.graph`, in `EBaMlProofs/MulticlassScores/Model.lean`.

Correspondence of symbols to the code. The table names the function that computes each symbol.

Symbol	Computed by	Function
K	`model_kind(ebm).outputs`	`EBMGraph.from_ebm`
$s_t(v, k)$	`np.array(ebm.term_scores_[feature_index][1:-1], dtype=float)`	`EBMGraph.from_ebm`
$\mu_t(v)$	`scores.mean(axis=1, keepdims=True)`	`_select_outputs`
$s_t(v, k) - \mu_t(v)$	`centred = scores - scores.mean(axis=1, keepdims=True)`	`_select_outputs`
a	`normalization_constant`	`EBMGraph.from_ebm`
graph	`scores -= normalization_constant`	`EBMGraph.from_ebm`
σ	`scipy.special.softmax(..., axis=-1)`	`MulticlassClassifier.predict`

The code computes the mean over all the outputs of the model, and only then selects the columns of `classes`. The symbol $\mu_t(v)$ is therefore the mean over all of K. The normalization constant a is the mean or the minimum of the selected scores, or zero. The proof allows any number a, so it covers all three options.

3 The abstract result

This section knows nothing about EBMs.

Lemma 1. (positive denominator). *If K is not empty, then $\sum_{j \in K} e^{x_j} > 0$ for every $x : K \to \mathbb{R}$.*

Lean: `Softmax.sum_exp_pos`, in `EBaMlProofs/MulticlassScores/Softmax.lean`.

Q.E.D. by $e^y > 0$ for every real y, and a sum of positive numbers over a non-empty set is positive.

Theorem 1. (shift). *For every $x : K \to \mathbb{R}$ and every $c \in \mathbb{R}$, $\sigma(x + c) = \sigma(x)$, where $(x + c)_k = x_k + c$.*

Lean: `Softmax.softmax_add_const`, in `EBaMlProofs/MulticlassScores/Softmax.lean`.

⟨1⟩1 $e^{x_k + c} = e^{x_k} e^c$ for each k, so $\sum_j e^{x_j + c} = e^c \sum_j e^{x_j}$.
⟨1⟩2 $e^c \neq 0$, so the factor e^c of the numerator and the denominator cancels.
⟨1⟩3 **Q.E.D.** by ⟨1⟩1 and ⟨1⟩2.

Theorem 2. (only the differences count). *Let K be non-empty, and let $x, y : K \to \mathbb{R}$. Then $\sigma(y) = \sigma(x)$ if and only if there is a $c \in \mathbb{R}$ with $y_k = x_k + c$ for every k.*

Lean: `Softmax.softmax_eq_iff`, in `EBaMlProofs/MulticlassScores/Softmax.lean`.

⟨1⟩1 Assume $\sigma(y) = \sigma(x)$. Let $Z_x = \sum_j e^{x_j}$ and $Z_y = \sum_j e^{y_j}$, and let $c = \log Z_y - \log Z_x$.
  ⟨2⟩1 $Z_x > 0$ and $Z_y > 0$ by Lemma 1.
  ⟨2⟩2 The logarithm of $e^{y_k} / Z_y = e^{x_k} / Z_x$ is $y_k - \log Z_y = x_k - \log Z_x$, by ⟨2⟩1.
  ⟨2⟩3 **Q.E.D.** by ⟨2⟩2, which gives $y_k = x_k + c$.
⟨1⟩2 Assume $y_k = x_k + c$ for every k. Then $\sigma(y) = \sigma(x)$ by Theorem 1.
⟨1⟩3 **Q.E.D.** by ⟨1⟩1 and ⟨1⟩2.

4 The application

Lemma 2. (score of a shift). *For every shift d and every class k, the score of class k in the shifted model is $L_k - \sum_{t \in T} d_t(c_t)$.*

Lean: `Model.score_shift`, in `EBaMlProofs/MulticlassScores/Model.lean`.

Q.E.D. by $\sum_t (s_t(c_t, k) - d_t(c_t)) = \sum_t s_t(c_t, k) - \sum_t d_t(c_t)$, and the last sum does not depend on k.

Theorem 3. (shift of a model). *For every shift d, the shifted model predicts the same probabilities as the model.*

Lean: `Model.predict_shift`, in `EBaMlProofs/MulticlassScores/Model.lean`.

Q.E.D. by Lemma 2 and Theorem 1, with $c = -\sum_t d_t(c_t)$.

Theorem 4. (graph). *Let $a \in \mathbb{R}$. For every term t, every bin v and all classes k and j: (a) the mean of the graph scores of bin v over all of K is $-a$; (b) the difference of the graph scores of k and j in bin v is $s_t(v, k) - s_t(v, j)$; (c) the graph predicts the same probabilities as the model.*

Lean: `Model.graph_mean`, `Model.graph_sub`, `Model.predict_graph`, in `EBaMlProofs/MulticlassScores/Model.lean`.

⟨1⟩1 (a) holds. The sum of the graph scores is $\sum_k s_t(v, k) - |K|\ (\mu_t(v) + a) = -|K|\ a$, by the definition of $\mu_t(v)$ and $|K| \neq 0$.
⟨1⟩2 (b) holds, because the number $\mu_t(v) + a$ is the same for k and j, and cancels.
⟨1⟩3 (c) holds by Theorem 3, because the graph is a shifted model.
⟨1⟩4 **Q.E.D.** by ⟨1⟩1 to ⟨1⟩3.

Claim 1 of Section 1 is Theorem 2. Claims 2, 3 and 4 are Theorem 4 (a), (b) and (c), provided that the code computes the quantities of Section 2. The table in Section 2 shows where.

5 Limits

The proof does not cover the following points.

- **Floating point.** The proof works over the real numbers. The code uses 64-bit floating point, so the mean of a centred bin is zero only up to a rounding error.
- **The standard deviations.** The graph keeps the standard deviations of the EBM, and the claims say nothing about them. A shift by one number does not change a standard deviation, but the confidence bounds of the graph text are not part of the model.
- **The prediction of interpret.** The proof assumes that the EBM predicts the softmax of the intercept plus the term scores. This is the model of `interpret` [1], and `MulticlassClassifier.predict` computes the same softmax. The proof does not check the code of `interpret`.
- **The missing and unknown bins.** `EBMGraph.from_ebm` drops the first and the last bin of `term_scores_`. The graph shows no score for them. Theorem 3 holds for every shift, so the claims hold whether these bins are shifted or not.
- **The text.** The LLM reads the scores as rounded decimal text. The proof is about the numbers before the rounding.

6 Computer check

The Lean 4 library in `proofs/` states each definition, lemma and theorem of Section 2, Section 3 and Section 4, and proves it with the Mathlib library [3]. The file `EBaMlProofs/MulticlassScores/Softmax.lean` holds the abstract part, and the file `EBaMlProofs/MulticlassScores/Model.lean` holds the model and the two theorems about it. The file `EBaMlProofs/MulticlassScores/Axioms.lean` prints the axioms of the main theorems. The proof has no `sorry`, no `admit` and no new `axiom`.

A continuous integration job runs `lake build` when the Lean files or the pins change. A reader can also run the check by hand. Section A gives the exact commands, the versions, the expected output and a table from each statement to its Lean name.

7 References

[1] H. Nori, S. Jenkins, P. Koch, and R. Caruana, "InterpretML: A Unified Framework for Machine Learning Interpretability." [Online]. Available: https://arxiv.org/abs/1909.09223

[2] L. de Moura and S. Ullrich, "The Lean 4 Theorem Prover and Programming Language." [Online]. Available: https://lean-lang.org/

[3] The mathlib Community, "The Lean Mathematical Library." [Online]. Available: https://arxiv.org/abs/1910.09336

A Appendix: how to repeat the check

Versions.

Item	Value
Repository	`https://github.com/johnny-godoy/EBaMl`
Lean sources checked at commit	`b6885408e6c0d9b3024a4fd90612c37b091bbdc7`
Lean	`leanprover/lean4:v4.35.0` (commit `eb6cafb7290e`)
Mathlib	tag `v4.35.0` (commit `56a005855ea17e0b0958f73ec801ae140b0a1679`)
Typst	`0.15.0`

The files in `proofs/EBaMlProofs/`, `proofs/lakefile.toml`, `proofs/lean-toolchain` and `proofs/lake-manifest.json` are the same at the commit above and at the commit of this paper. The functions of Section 2 are in `src/ebaml/graphs/ebm_graph.py` and `src/ebaml/graphs/model_kinds.py`. Read the paper together with the version of these files at the commit of this paper.

Commands from a clean clone.

```
# install elan, then restart the shell
curl https://elan.lean-lang.org/elan-init.sh -sSf | sh
git clone https://github.com/johnny-godoy/EBaMl && cd EBaMl
git checkout b6885408e6c0d9b3024a4fd90612c37b091bbdc7
cd proofs
lake exe cache get        # download the compiled Mathlib
lake build                # check every proof
typst compile --root papers papers/multiclass-scores/paper.typ
```

The last command needs the commit of this paper, because the commit above has no paper. If `lake exe cache get` cannot reach its server, `lake build` compiles the imported part of Mathlib from source. This takes tens of minutes, and the result is the same.

Expected output of `lake build`. The build ends with `Build completed successfully` and no line that starts with `error`. The file `EBaMlProofs/MulticlassScores/Axioms.lean` prints these lines, in this order:

```
info: EBaMlProofs/MulticlassScores/Axioms.lean:10:0:
'EBaMlProofs.MulticlassScores.Softmax.softmax_add_const' depends on axioms:
[propext, Classical.choice, Quot.sound]
info: EBaMlProofs/MulticlassScores/Axioms.lean:11:0:
'EBaMlProofs.MulticlassScores.Softmax.softmax_eq_iff' depends on axioms: [propext,
Classical.choice, Quot.sound]
info: EBaMlProofs/MulticlassScores/Axioms.lean:12:0:
'EBaMlProofs.MulticlassScores.Model.predict_shift' depends on axioms: [propext,
Classical.choice, Quot.sound]
info: EBaMlProofs/MulticlassScores/Axioms.lean:13:0:
'EBaMlProofs.MulticlassScores.Model.graph_mean' depends on axioms: [propext,
Classical.choice, Quot.sound]
info: EBaMlProofs/MulticlassScores/Axioms.lean:14:0:
'EBaMlProofs.MulticlassScores.Model.graph_sub' depends on axioms: [propext,
Classical.choice, Quot.sound]
info: EBaMlProofs/MulticlassScores/Axioms.lean:15:0:
'EBaMlProofs.MulticlassScores.Model.predict_graph' depends on axioms: [propext,
Classical.choice, Quot.sound]
```

The build may print warnings that Mathlib modules are “designed for use with the module system”. These warnings are expected, and they do not affect the result. The line breaks of long lines may differ. The word

`sorryAx` must not appear. To search for it, run `lake build 2>&1 | grep -c sorry`. The expected result is `0`.

The last command writes `papers/multiclass-scores/paper.pdf` and prints nothing.

Statements and Lean names. The directory is `proofs/EBaMlProofs/MulticlassScores/`. In the table, `Softmax.` stands for `EBaMlProofs.MulticlassScores.Softmax.` and `Model.` stands for `EBaMlProofs.MulticlassScores.Model.`.

Statement	**Lean name**	**File**
Definition 1 (softmax)	`Softmax.softmax`	`Softmax.lean`
Definition 2 (prediction)	`Model.score`, `Model.predict`	`Model.lean`
Definition 3 (shift and graph)	`Model.shift`, `Model.mean`, `Model.graph`	`Model.lean`
Lemma 1 (positive denominator)	`Softmax.sum_exp_pos`	`Softmax.lean`
Theorem 1 (shift)	`Softmax.softmax_add_const`	`Softmax.lean`
Theorem 2 (only the differences count)	`Softmax.softmax_eq_iff`	`Softmax.lean`
Lemma 2 (score of a shift)	`Model.score_shift`	`Model.lean`
Theorem 3 (shift of a model)	`Model.predict_shift`	`Model.lean`
Theorem 4 (a)	`Model.graph_mean`	`Model.lean`
Theorem 4 (b)	`Model.graph_sub`	`Model.lean`
Theorem 4 (c)	`Model.predict_graph`	`Model.lean`