

Candidate-Aware Aggregated Likelihood Model (CALM)

Mathematical intuition and plug-in logic for adding candidate traits to the existing cell→booth turnout/party framework

0) Quick recap of the baseline

We work at **cell** level (i, c) inside booth i . Each cell is a product of demographic axes (Age × Religion × Caste × Income) with size n_{ic} . The model learns:

- **Turnout** (per cell): $\pi_{ic} = \Pr[\text{votes} \mid i, c]$
- **Party choice** (per cell, K parties): $\theta_{ic,k} = \Pr[\text{party} = k \mid i, c], \sum_k \theta_{ic,k} = 1$

Aggregate to booth:

$$\hat{T}_i = \sum_c n_{ic} \pi_{ic}, \quad \hat{V}_{ik} = \sum_c n_{ic} \pi_{ic} \theta_{ic,k}, \quad \hat{t}_i = \hat{T}_i / N_i, \quad \hat{p}_{ik} = \hat{V}_{ik} / \hat{T}_i.$$

The current parameterization (simplified) is

$$\text{logit}(\pi_{ic}) = \alpha_0 + b_i^{(T)} + x_{ic}^\top \beta_T, \quad u_{ic,k} = \gamma_k + b_{ik}^{(P)} + x_{ic}^\top \beta_{P,k}, \quad \theta_{ic,k} = \text{softmax}_k(\nu_{ic,\cdot}).$$

Here, x_{ic} are one-hot/continuous features from the demographic axes and covariates (land rate, M/F ratio, etc.).

1) Candidate traits as a vector

For each party k in the Assembly/constituency, we encode its candidate into the same axes as cells:

- Age bin (e.g., 46–60)
- Religion (e.g., Hindu)
- Caste (e.g., OBC; only meaningful within Hindu religion)
- Income tier (e.g., Middle)

Let z_k denote the concatenated one-hot vector for party k 's candidate across those axes. (If an axis is unknown/not applicable, its block is set to 0.)

2) Cell-candidate affinity features $g_{ic,k}$

We construct a small, interpretable set of features that measure how well cell (i, c) "matches" candidate k :

General form. For each axis $d \in \{\text{age, religion, caste, income}\}$,

$$\underbrace{g_{ic,k}^{(d)}}_{\text{scalar}} = \kappa_d(e_{ic}^{(d)}, z_k^{(d)}),$$

where $e_{ic}^{(d)}$ is the one-hot of the cell on axis d , $z_k^{(d)}$ is the candidate one-hot on the same axis, and κ_d is a simple similarity kernel.

Recommended kernels - Age proximity (soft): $\kappa_{\text{age}}(a, \tilde{a}) = a^\top A(\rho_A) \tilde{a}$, with banded matrix A having 1 on the diagonal and $\rho_A \in [0, 1)$ on adjacent bins. This gives partial credit to neighboring age groups.

- **Religion match (hard):** $\kappa_{\text{rel}}(r, \tilde{r}) = \mathbf{1}\{r = \tilde{r}\}$. - **Caste match (Hindus only):** $\kappa_{\text{caste}}(c, \tilde{c}) = \mathbf{1}\{\text{religion(cell)} = \text{Hindu}\} \cdot \mathbf{1}\{c = \tilde{c}\}$. - **Income match (hard or soft):** $\kappa_{\text{inc}}(y, \tilde{y}) = \mathbf{1}\{y = \tilde{y}\}$ (or a 3×3 proximity similar to age).

Stack them:

$$g_{ic,k} = [g_{ic,k}^{(\text{age})}, g_{ic,k}^{(\text{rel})}, g_{ic,k}^{(\text{caste})}, g_{ic,k}^{(\text{inc})}]^\top \in \mathbb{R}^4.$$

Each component is standardized across all (i, c, k) before training for numerical stability.

Intuition. These features let cells that “look like” the candidate receive an additive boost for that party in the choice model; the boost bleeds a little across neighboring age bins but respects religion/caste structure.

3) Plug-in to party choice utilities

We augment the party utilities with a linear candidate term:

$$\nu_{ic,k} = \underbrace{\gamma_k}_{\text{party intercept}} + \underbrace{b_{ik}^{(P)}}_{\text{booth bias}} + \underbrace{x_{ic}^\top \beta_{P,k}}_{\text{current features}} + \underbrace{\eta^\top g_{ic,k}}_{\text{candidate affinity}}, \quad \theta_{ic,k} = \text{softmax}_k(\nu_{ic,.}).$$

- η is a small parameter vector **shared across parties** by default (parsimonious and easier to identify). If desired, use party-specific η_k with stronger L2. - Because $g_{ic,k}$ is standardized, η is interpretable as the marginal log-odds impact of a 1-SD change in candidate affinity.

Alternative (bilinear/two-tower). Replace the linear term with a low-rank interaction between cell and candidate encodings: $\nu_{ic,k} = \dots + x_{ic}^\top U z_k$. This is more flexible but adds parameters; start with the linear $\eta^\top g$ term.

4) Optional: candidate effect on turnout (mobilization)

To let aligned candidates raise turnout among aligned cells, add a scalar mobilization term:

$$m_{ic} = \sum_k w_{ik} s_{ic,k}, \quad s_{ic,k} = w_A g_{ic,k}^{(\text{age})} + w_R g_{ic,k}^{(\text{rel})} + w_C g_{ic,k}^{(\text{caste})} + w_I g_{ic,k}^{(\text{inc})}.$$

Use booth-level weights w_{ik} (e.g., previous election share for party k in booth i , or current \hat{p}_{ik} frozen from a prior epoch). Then

$$\text{logit}(\pi_{ic}) = \alpha_0 + b_i^{(T)} + x_{ic}^\top \beta_T + \xi m_{ic}.$$

5) Objective and regularization

Let $N_i = \sum_c n_{ic}$ and T_i be observed polled votes. Use the same booth-weighted losses, adding L2 on candidate params:

$$\mathcal{L} = \sum_i \underbrace{\lambda_T N_i \text{BCE}(\hat{t}_i, t_i)}_{\text{turnout}} + \sum_i \underbrace{\lambda_P T_i \text{KL}(p_i \| \hat{p}_i)}_{\text{party shares}} + \lambda_\beta \|\beta\|_2^2 + \lambda_b \|b\|_2^2 + \lambda_\eta \|\eta\|_2^2 (+ \sum_k \lambda_{\eta,k} \|\eta_k\|_2^2).$$

Stability: work in log-softmax for ν , add ε when forming KL, and standardize each component of g .

6) Identification & design notes

- **Centering.** Standardize g over (i, c, k) so $\mathbb{E}[g] = 0$; this avoids intercept confounding between γ_k and candidate lift.
 - **Collinearity.** If some axes are already in x_{ic} , the new term is distinct because it multiplies the **candidate** identity. Orthogonalization (project g off x) is optional but not required with L2.
 - **Caste within religion.** Always null out caste for non-Hindus: $g_{ic,k}^{(\text{caste})} = 0$ if $\text{religion}(\text{cell}) / \text{Hindu}$.
 - **Missing traits.** If the candidate has unknown caste/income, set the corresponding block to zero; the model relies on available matches.
-

7) Implementation blueprint (drop-in)

1. **Config:** add `use_candidate_affinity: bool`, `rho_age`, `lambda_eta`.
2. **Encode candidates** per party into z_k ; precompute the age proximity matrix $A(\rho_A)$.
3. **Build** $g_{ic,k}$ when constructing cell features. Cache as a tensor of shape `[num_booths, num_cells_i, K, 4]` or compute on the fly in the forward pass.
4. **Standardize** each component of g using running mean/var over the training set. Persist stats for inference.
5. **Forward pass (party head):** add `eta @ g_{ic,k}` to logits for class k . Register `eta` as trainable param.
6. **Loss/regularization:** add `lambda_eta * (eta**2).sum()` (and party-specific terms if used).
7. **Exports:** include `eta` and component names in your coefficients report for interpretability.

Complexity. The additive term is $\mathcal{O}(\#cells \times K)$ with a very small constant; memory footprint is negligible compared to cell features.

8) Interpreting coefficients

- $\eta_{\text{rel}} > 0$: same-religion candidates improve that party's log-odds across aligned cells.
- $\eta_{\text{caste}} > 0$: among Hindus, same-caste alignment matters.
- $\eta_{\text{age}} > 0$: older candidates overperform in their age band and adjacent bands (via A).
- $\eta_{\text{inc}} > 0$: income-class alignment helps.

Because g is standardized, a value like $\eta_{\text{rel}} = 0.25$ means a +0.25 log-odds shift ($\approx +6$ pp in probability near the center) for a +1 SD increase in religion-match affinity, holding other terms fixed.

9) Diagnostics & ablations

- **Holdout booths**: compare KL/BCE with vs. without candidate term.
- **Coefficient stability** across Assemblies.
- **Placebo**: shuffle candidate caste within religion—effect should vanish.
- **Sensitivity to ρ_A** : 0.3–0.6 typically stable; tune via validation.
- **Turnout channel on/off**: check whether mobilization ξ improves weighted BCE.

10) Practical example (your provided case)

Madipur, BJP candidate: 54-year-old Hindu OBC, middle class. - Candidate vector blocks: Age=46–60, Religion=Hindu, Caste=OBC (active only for Hindus), Income=Middle. - For a Hindu-OBC, 46–60, middle-income cell: $g = [1, 1, 1, 1]$ → maximal boost $\eta_{\text{age}} + \eta_{\text{rel}} + \eta_{\text{caste}} + \eta_{\text{inc}}$. - For a Hindu-SC, 36–45, middle-income cell: $g = [\rho_A, 1, 0, 1]$ → partial age, full religion and income, no caste. - For non-Hindu cells: $g = [\text{age-prox}, 0, 0, \text{inc}]$.

11) Extensions

- Add more axes: **gender, education, incumbency, tenure**, or a scalar **valence** score; each becomes another kernel in g .
- Replace linear $\eta^\top g$ by low-rank **two-tower** term $x^\top U z$ if you need richer interactions (start with rank 2–4).
- **Hierarchical sharing** of η across constituencies within a state for better small-sample behavior.

12) Minimal pseudocode (illustrative)

```
# Precompute
A = make_age_proximity(rho_age)
Z = encode_candidates_onehot(candidates) # dict party->blocks

# Build g per (i,c,k)
for (i,c) in cells:
    a,r,cas,inc = cell_bins(i,c)
```

```

for k in parties:
    age_sim    = A[a, Z[k].age]
    rel_match = int(r == Z[k].rel)
    caste_mt   = int(r == HINDU and cas == Z[k].caste)
    inc_match  = int(inc == Z[k].inc)
    g[i,c,k]   = [age_sim, rel_match, caste_mt, inc_match]

# Standardize g components (fit on train)
eta = torch.nn.Parameter(torch.zeros(4))

# Forward in party head
nu_ic_k = gamma[k] + bP[i,k] + x_ic @ betaP[k] + (eta @ g[i,c,k])

```

Bottom line

Add a tiny, interpretable **candidate-affinity** term to party utilities (and optionally turnout). It reuses your existing axes, preserves aggregation and loss, adds negligible complexity, and yields clear, testable coefficients about how “matching” the candidate is to each voter cell. This is the most robust way to introduce candidate information without destabilizing the original framework.