

# Predicting to New Locations in Spatial ML

Josiah Parry

## Problem Statement

Predicting to new locations in spatially explicit machine learning models is ill-defined. These models use spatially lagged covariates, or message passing in the case of graph neural networks (GNNs), which are defined only for locations embedded in the training spatial weights matrix (SWM). Prediction requires both identifying the neighborhood structure of new locations and accessing the covariate values used to construct the lag. Without this information, the required inputs do not exist.

## Spatially Explicit Model Approaches

Models become spatially explicit through various mechanisms. The following approaches incorporate spatial relationships into machine learning models:

- **Spatial lags:** Weighted averages of neighbor values in spatial regression models
- **Message passing:** Graph neural network mechanism equivalent to spatial lags on hidden dimensions
- **Moran Eigenvector Maps (MEMs):** Spatial features derived from the spatial weights matrix
- **Geographically Weighted Regression (GWR/MGWR):** Local regression coefficients estimated at each location
- **Distance features:** Covariates computed relative to external reference layers
- **Spatial indexing:** H3, S2, or geohashes as categorical features (unproblematic for prediction)
- **Coordinates and regimes:** X/Y coordinates or spatial regimes as features (unproblematic for prediction)

For simplicity, we focus on **spatial lags** as the canonical spatially explicit feature, though the challenges discussed apply broadly to approaches dependent on the spatial weights matrix or training data. At the end we discuss how these challenges extend to other approaches.

## Spatially Explicit Model Prediction Scenarios

Consider a model trained on a study area with a spatial lag of variable  $X_1$ . Four prediction scenarios emerge:

1. **Prediction at all original locations with updated covariates:** Use the same adjacency matrix; calculate spatial lag from new  $X_1$  values for all locations. Requires persisting or recreating the SWM and collecting new data for all locations.
2. **Prediction at a single original location:** Use known adjacency; calculate lag from training data. Requires persisting the SWM and original  $X_1$  values.
3. **Prediction at an entirely new study area:** Create a new SWM for the new region; calculate spatial lag from new data. Raises questions about adjacency structure consistency (e.g., must k-NN be used if trained on k-NN?).

**4. Prediction at  $m$  new locations within the original study area:** Most complex scenario—requires choosing how new locations relate to the training graph (see Approaches to Spatial Feature Construction below).

**Graph Neural Networks:** GNNs face similar challenges. Transductive models cannot generalize to unseen nodes. Inductive models can generalize but still require graph structure for new nodes, necessitating insertion into or reconstruction of the adjacency matrix.

## Approaches to Spatial Feature Construction

When confronting the train/test boundary, three coherent positions exist:

**A. Closed system (transductive):** Calculate spatial lag on the full dataset (train + test) before splitting. Semantically consistent but implies leakage and undefined prediction to new locations.

**B. Fully separate graphs:** Lags computed independently within training and test sets. Semantically consistent but model is evaluated on a different graph structure than it was trained on.

**C. Hybrid approaches:** Test lags incorporate training data. Two variants:

- **C1 (Individual lookup):** Each test point finds neighbors in the training set independently; test points do not see each other.
- **C2 (Grow adjacency matrix):** Combine training and test points into a new adjacency matrix where test points can neighbor each other and training points.

None of these approaches is obviously correct. Each involves trade-offs between semantic consistency, data leakage, and the ability to predict at new locations. The common practice (Position A) works for in-sample evaluation but leaves prediction to new locations undefined. Position C variants enable prediction but introduce semantic inconsistency—the model learns relationships among training neighbors but predicts using different neighbor definitions.

## Challenges for Prediction

Three core challenges emerge across prediction scenarios:

**1. Modifying the Adjacency Matrix:** New locations require insertion into or reconstruction of the SWM. For GNNs, new prediction points must be identified relative to the original adjacency matrix. The choice between insertion (preserving training structure) versus recomputation (rebuilding from scratch) involves trade-offs between consistency with training and spatial coherence of predictions.

**2. Feature Construction:** Spatial lags for new locations require covariate values from neighbors. This raises the question: how do we determine which features are neighbors of the new prediction point? Those neighbors may be training observations, new observations, or both—corresponding to the positions outlined above.

**3. Storing the Adjacency Matrix and Training Features:** Both adjacency modification and feature construction require access to the original SWM and covariate values. These must be persisted beyond training—not typically part of standard ML model serialization. For methods like GWR/MGWR, the entire training dataset must be stored alongside the model.

The literature largely overlooks these challenges. Papers focus on model training but do not account for predicting to **new locations**. This creates a gap between methodological innovation and operational deployment. Prediction at new locations is not a trivial extension of training—it requires explicit protocols for SWM persistence, adjacency decisions at prediction time, and covariate availability.

## Examples from Literature

**Position A in practice (Liu, Kounadi, and Zurita-Milla 2022):** Compute spatial lags on the full dataset before partitioning for cross-validation. This is common practice—works for in-sample evaluation but leaves prediction to new locations undefined.

**Position C1 in practice (Credit 2025):** The `{sarf}` package’s `spatial_cv_rf()` function finds neighbors in the training set for each test observation independently. Test points do not see each other, even if spatially adjacent.

**Position C1 with explicit prediction (Gao, He, and Kwan 2025):** M-GWXGB explicitly implements prediction to new locations by finding k-nearest training locations and averaging their local model predictions. This requires persisting all local models, training locations, and bandwidth—the model alone cannot predict.

## Extension to Other Spatially Explicit Approaches

The three positions (A, B, C) and associated prediction challenges extend to other spatially explicit modeling techniques. Each approach presents different constraints on which positions are viable.

### Unproblematic Approaches

Some techniques avoid the prediction challenges entirely and maintain “train once, predict anywhere”:

- X and Y coordinates as features
- Spatial indices (H3, S2, Geohash)
- Spatial regimes (categorical variables like state names or county IDs)
- Position-agnostic: work under any splitting strategy because they do not depend on neighborhood relationships or the SWM

### Distance Features

Distance features computed relative to external reference layers face analogous questions to spatial lags:

- Position A: Use only training distance features (requires storage and availability)
- Position B: Calculate new distance features for test locations (requires reference layers; potential semantic inconsistency)
- Position C2: Combine training and new distance features (raises questions about when augmentation is valid)
- Like spatial lags, requires supplementary data beyond the model itself

### Moran Eigenvector Maps (MEMs)

MEMs are derived directly from the SWM, creating strong dependence on neighborhood structure:

- Position A: Store MEMs and query for predictions at original locations (straightforward)
- Position B: Calculate MEMs on new data using same adjacency definition (inductive approach)
- Position C: Cannot derive MEMs for new observations not in original SWM decomposition (eigenvector decomposition cannot be incrementally updated)
- Particularly inflexible for prediction at new locations within the original study area

## GWR and MGWR

GWR and MGWR produce  $n$  sets of local coefficients rather than a single global model. Prediction at new locations requires fitting new weighted regression using nearby training observations:

- Requires full training dataset ( $X$  and  $Y$ ) at prediction time, not just serialized model
- Bandwidth(s) from training can be reused, but coefficient estimation needs training data
- Aligns with Position C1 (querying training set) but cannot escape model + data requirement
- “Train once, predict anywhere” paradigm does not apply

## Graph Neural Networks

GNNs face similar but more rigid constraints:

- Transductive GNNs (e.g. GCN): Fixed graph, cannot generalize to unseen nodes (inherently Position A)
- Inductive GNNs (e.g. GraphSAGE): Can generalize to new structures (Position B) but still require graph structure at prediction time
- Position C variants: Require reconstructing/augmenting adjacency matrix, then forward pass with expanded graph
- Open question: Does inserting new nodes and running forward pass without retraining preserve semantic meaning or introduce Position C inconsistencies?

Credit, Kevin. 2025. “SArf: Spatial Autoregressive Random Forest.” <https://github.com/kcredit/SArf>.

Gao, Fan, Sylvia Y. He, and Mei-Po Kwan. 2025. “Mixed Geographically Weighted XGBoost (M-GWXGB) Model: A New Spatially Explicit Machine Learning Model.” *Annals of the American Association of Geographers*, December, 1–32. <https://doi.org/10.1080/24694452.2025.2586039>.

Liu, Xiaojian, Ourania Kounadi, and Raul Zurita-Milla. 2022. “Incorporating Spatial Autocorrelation in Machine Learning Models Using Spatial Lag and Eigenvector Spatial Filtering Features.” *ISPRS Int. J. Geoinf.* 11 (4): 242.