Predicting to New Locations in Spatial ML

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.

Introduction

- Growing adoption of spatially explicit ML models in geographic research
- Literature focuses on model training but neglects the mechanics of prediction at new locations
- This paper addresses the fundamental challenge: models incorporating spatial lags or message passing have prediction inputs that are undefined outside the training spatial weights matrix (SWM)
- In this case only talking about vector data and not raster data

Current Practice and Its Implicit Assumptions

- Standard workflow: calculate spatial lag (or other spatially derived features) on the full dataset, then partition into train/test/validation sets for spatial CV
- This approach has a transductive character—test observations are embedded in the spatial graph from the start
- The spatial lag of a test observation was computed using training observations (and vice versa)
- The test set was never truly "held out"—it participated in feature construction
- Spatial CV is presented as rigorous, but if features are computed on the full graph before splitting, what is actually being validated?
- This workflow implicitly assumes a closed system where all locations are known at training time
- Prediction to genuinely new locations is undefined because the workflow was never designed for it

Three Positions on Spatial Feature Construction

When confronting the train/test boundary, three coherent positions exist. For illustration, assume we use k-nearest neighbors (kNN) as the method of defining adjacency, though the logic applies to other adjacency definitions.

A. Closed system (transductive)

- Calculate spatial lag on the full dataset (train + test) before splitting
- Semantically consistent—lag means the same thing for all observations
- Implies: leakage; test set must be known at training time; prediction to new locations undefined

B. Fully separate graphs

- Training: lags computed among training points only
- Testing: lags computed among test points only (neighbors are other test points)
- Semantically consistent—lag always means "average among my peers"
- Implies: model is evaluated on a different graph structure than it was trained on

C. Hybrid approaches (test lags incorporate training data)

Two variants exist:

- **C1: Individual lookup** — Each test point finds its neighbors in the training set independently. Test points do not see each other. The adjacency matrix is not grown; test points query against it.
- **C2: Grow the adjacency matrix** — Combine test and training points into a new adjacency matrix. Test points can be neighbors of each other *and* of training points. Lags are computed on this combined structure.

Both variants change the meaning of the spatial lag between training and testing. C1 treats each prediction as an isolated query; C2 treats the test set as a cohort integrated into the spatial structure.

Note: None of these approaches is obviously correct. Each involves trade-offs between semantic consistency, data leakage, and the ability to predict at new locations. These positions directly inform the adjacency decisions discussed in the prediction scenarios below.

The Semantic Inconsistency of Position C

- During training, the spatial lag for observation i is the (weighted) mean of Y among i's neighbors—all of whom are in the training set
- Under Position C1, the spatial lag for test observation j is the mean of Y among j's neighbors *in the training set*—the test point is querying a dataset it does not belong to

- Under Position C2, the adjacency matrix is recalculated to include both training and test observations; test points can be neighbors of each other and of training points
- The model learns "when my neighbors have high Y, I tend to have high Y"—but for test points under C1, "neighbors" means "nearest training observations"

Examples from the Literature

Position A in practice: @Liu2022-pv

@Liu2022-pv compute spatial lags on the full dataset before partitioning for cross-validation. This is Position A—the standard practice. It works for in-sample evaluation but leaves prediction to new locations undefined.

Position C1 in practice: @sarf2025

The `{sarf}` package's `spatial_cv_rf()` function implements Position C1 (see https://github.com/kcredit/SArf/ tial_cv.R#L16). For each test observation, it finds neighbors in the training set and computes the lag from training Y values. Each test point is treated as an isolated query against the training graph. Test points do not see each other—even if two test points are spatially adjacent, they cannot be neighbors.

Position C1 with explicit prediction: @Gao2025-mgwxgb

M-GWXGB is notable for explicitly implementing prediction to new locations. The `GeoWeightedXGBoostPredictor` class predicts at new locations by finding the k-nearest training locations, retrieving those locations' models, and averaging their predictions (see https://github.com/yiloufengyue/M-GWXGB/blob/main/M_GWXGB_1.py). This requires persisting all local models, training locations, and the bandwidth. The model alone cannot predict—it requires the full training infrastructure.

Background: How Models Become Spatially Explicit

Approaches Unproblematic for Prediction

- Including X & Y coordinates in a model—not ideal, but not problematic for prediction
  - Doesn't incorporate relationship of feature and neighborhood
- Using H3 indexes, S2 indexes, or geohashes as categorical features
  - H3 recommends using `kRing` for spatial "convolution" (just a spatial lag) https://h3geo.org/docs/highlights/ml/
  - H3 interpolation in "Geographic Data Science with Python" https://geographicdata.science/book/notebooks/12_feature_engineering.html

    - https://www.analyticsvidhya.com/blog/2025/03/ubers-h3-for-spatial-indexing/#h-combining-h3-with-machine-learning
- Spatial regimes (e.g. state names) as categorical features

Note: These approaches allow prediction at new locations provided the index or regime can be assigned.

Explanatory Distance Features

- Distance features are computed relative to external reference layers
- The model alone cannot predict—it requires the model plus the distance feature layers (supplementary data)
- This raises the same structural questions as spatial lag models:
    - Must the distance feature layers be identical to those used during training?
    - Can we use new/matched distance feature layers for a different study area? When is this valid?
    - Can distance feature layers grow (training layers + new layers)?
- The choice of how to handle distance features at prediction time parallels the adjacency choices for spatial lag models

Approaches Dependent on the SWM

- A spatial weights matrix is an n×n sparse square matrix. Non-zero values indicate the presence of a neighbor. By default, these are binary matrices where 1 indicates a neighbor. In this case a SWM is an adjacency matrix.
- Models use spatial lags in the case of spatial regression (Anselin and Rey)
- Include Moran Eigenvector Maps (MEMs) which are derived from the SWM (Dray)
    - Cannot be used on new locations
- Graph neural nets—a growing way of explicitly incorporating space—use “message passing” which is equivalent to a spatial lag except on hidden dimensions

Note: These approaches bind the model to a fixed neighborhood structure and create the prediction problem.

Approaches Dependent on Training Data at Prediction Time

- Geographically Weighted Regression (GWR) and Multiscale GWR (MGWR) do not use a SWM, but face analogous prediction challenges
- GWR fits local regression coefficients at each training location using distance-weighted observations
- The "model" is not a single set of coefficients—it is n sets of local coefficients, one per training location
- To predict at a new location, coefficients must be estimated *at that location* by fitting a new weighted regression using nearby training observations
- This requires access to training X and Y at prediction time, not just a saved model object
- MGWR adds complexity: each covariate has its own bandwidth, requiring multiple local regressions at each new prediction point
- The bandwidth(s) learned during training can be reused, but the local coefficient estimation still requires training data

Note: GWR/MGWR cannot escape the same fundamental issue—prediction requires access to training data, not just a serialized model. The "train once, predict anywhere" paradigm does not apply.

Prediction Scenarios: Spatial Lag Models

Model was trained on a study area with a spatial lag of variable X_1.

Prediction at All Original Locations with Updated Covariates

We want to predict for all of the same locations we trained on, but with new covariate values collected post-training. We can use the same adjacency matrix and calculate the spatial lag of X_1 using new data for all locations.

- We can either persist the SWM or recreate it
- Requires new X_1 for all locations

Prediction at a Single Original Location

We want to predict to a single location in the original training set. We know the adjacency for this location. We can calculate lag of X_1 from training data and use that as lag of X_1.

- Requires we persist the SWM
- Requires we persist the original X_1 for all locations so we can derive lag of X_{1i} at a later point

Prediction at an Entirely New Study Area

We want to predict to an entirely new study area using the same model. We must calculate lag of X_1 using our new data. We are responsible for creating a new SWM based on the input data.

- We must create a new SWM on the new data—raises questions about how we define adjacency for the new data. Must it be the same adjacency structure—e.g. trained on k-NN predicted to k-NN?
- Requires that we calculate lag of X_1 from X_1 of all new locations

Prediction at m New Locations Within the Original Study Area

We want to predict m new locations that are part of the study region. This is where the positions defined in "Three Positions on Spatial Feature Construction" become operationally relevant:

- **Position B**: Adjacency defined only among the new prediction set, ignoring the training graph
- **Position C1**: Each new point finds neighbors in the original SWM only; new points do not see each other
- **Position C2**: Grow the adjacency matrix to include both original and new locations; new points can be neighbors of each other and of training points

In all scenarios this requires:

- We persist the original SWM
- We persist the original X_1 for all locations
- If the SWM is grown (Position C2), we must augment the SWM and persist it, same with X_1 data

Prediction Scenarios: Graph Neural Networks

Transductive vs. Inductive Models

- Transductive: Fixed graph; cannot generalize to unseen nodes
- Inductive: Can generalize, but still requires graph structure for new nodes

TODO: discuss transductive vs inductive models in more detail

Prediction Mechanics

When we consider GNNs these same points are the same/similar. However, we cannot necessarily predict to new locations unless using an inductive model. In order to predict with a GNN, the full adjacency matrix and X matrix must be provided. So, in order to do this, we must insert new points into the trained adjacency matrix. This thus requires us to address the same problems as spatial lag models.

Three Core Challenges for Prediction

The prediction scenarios above reveal three common challenges:

Modifying the Adjacency Matrix

- New locations require insertion into or reconstruction of the SWM
- In the case of GNNs we must be able to identify the new prediction point relative to the original adjacency matrix
- Decision point: insertion vs. recomputation (see next section)

Feature Construction

- When spatially explicit models are trained using spatial lags, new data must also have these as covariates in order to predict
- This begs the question: "how do we determine which features are neighbors of our new prediction point?"
- In every case, we must locate the prediction point in a SWM to calculate the spatial lag
- Spatial lags for new locations require covariate values from neighbors
- Those neighbors may be training observations, new observations, or both

Storing the Adjacency Matrix and Training Features

- Follows from the above: both modifying the adjacency matrix and feature construction require access to the original SWM and covariate values
- SWM and covariate values must be persisted beyond training
- Not typically part of standard ML model serialization

The Choice of Adjacency at Prediction Time

Adjacency is not a function learned by the model, yet prediction requires applying it.

- Prediction requires we have an adjacency matrix
- Two primary approaches: graph insertion vs. graph reconstruction

Insertion: We insert new points relative to the original SWM

- Preserves training structure
- New points find neighbors among training observations

Recomputation: We recalculate SWM with new prediction points

- May alter neighborhood relationships
- Rebuilds structure from scratch

Trade-offs: Consistency with training vs. spatial coherence of predictions

Discussion

- Papers do not account for predicting models to **new locations** and the challenges associated with it
- Lack of standardized protocols for spatial ML prediction pipelines
- Need for model serialization to include SWM and covariate storage
- Implications for reproducibility and operational deployment
- When adjacency choices at prediction time may invalidate model assumptions

Conclusion

- Prediction at new locations is not a trivial extension of training
- Explicit protocols needed for: (1) SWM persistence, (2) adjacency decisions at prediction, (3) covariate availability
- Call for methodological standards in spatial ML workflows