Fully Heterogeneous Collective Regression

David J. Liedtka
Department of Computer Science
United States Naval Academy
Annapolis, Maryland
dliedtka@gmail.com

Luke K. McDowell
Department of Computer Science
United States Naval Academy
Annapolis, Maryland
lmcdowel@usna.edu

ABSTRACT

Prior work has demonstrated that multiple methods for *link-based classification* (LBC) can substantially improve accuracy when the nodes of interest are interconnected. To date, however, very little work has considered how methods for LBC could be applied in domains that require *continuous*, rather than categorical, predictions. In addition, prior work with LBC has learned only one predictive model to use for all nodes of a given type, but some domains exhibit significant node diversity that is not well-suited to this approach. In response, we introduce *fully heterogeneous collective regression* (FHCR), a new method that learns node-specific models from data and uses these models to jointly predict continuous outputs. We apply FHCR to a voting prediction task, and create novel correlation-based links that outperform alternative methods. In addition, we introduce multiple new methods for inferring continuous outputs that can incorporate link-based information, and show that regression-specific methods based on Bayesian inference outperform the naive approach of inserting regression into existing LBC methods. Overall, we demonstrate the viability of the new FHCR paradigm by producing results that are comparable or better than those of previous link-unaware methods, yet are at least two orders of magnitude faster.

1 INTRODUCTION

Increasingly, many important domains in the world can be viewed as networks of linked nodes, such as people in online social networks or webpages connected by hyperlinks. To leverage these links for prediction and analysis tasks, Machine Learning researchers have developed multiple techniques for link-based classification (LBC) [4, 5, 10, 12, 14, 16]. While LBC can substantially improve prediction accuracy in some domains, current limitations greatly restrict its applicability when used to evaluate heterogeneous domains (e.g., when the collection of "nodes" under study are actually drawn from multiple populations). Additionally, traditional LBC predicts only categorical outputs, while link-based regression to predict continuous outputs has been left largely unexplored.

One such application that requires continuous outputs involves elections. Predicting the voting outcome of national or regional elections is a challenging yet important problem, and has great implications for regional and international security. As just one example, how well can the national outcome of an election be predicted, given past voting history and some incomplete "day of voting" results? A recent study by Etter et al. [3], using Swiss referendum outcomes, reported high accuracy, even when only 5% of voting "regions" had reported results. This study used a matrix factorization approach to implicitly leverage the correlation present between "nearby" regions. They did not, however, consider formulating the regions as a network.

This paper presents the first extension of LBC methods to node-specific predictive models and continuous outputs, producing *fully heterogeneous collective regression* (FHCR). To demonstrate the effectiveness of this approach, we apply it to the voting prediction task of Etter et al. We show that our link-based approach can be highly effective, even though the data initially contains no links.

Our contributions are as follows:

- We introduce a new paradigm, Fully Heterogeneous Collective Regression (FHCR), for predictions based on relational data. This new paradigm enables for the first time the application of link-based inference algorithms to domains with highly heterogeneous objects and continuous outputs.
- We introduce multiple new methods for inferring continuous outputs, for both the initial "bootstrap" problem where predictions must be made without links, and for the "collective" inference step where links are used. In both cases, we demonstrate that novel applications of Bayesian inference often lead to improved voting prediction accuracy, especially when relatively little information is known about a particular vote. In addition, we show how this inference can elegantly combine link-based information with information about object features, yielding further accuracy gains.
- We create novel methods for link creation based on historical correlations, and demonstrate that for voting prediction they generally yield more accurate results than simpler methods based on geographic proximity.
- We perform extensive evaluations of FHCR on the voting prediction task. By combining our new methods, we achieve voting prediction results that are competitive with or even improve upon those used by the prior study, but execute 110-790 times faster than previous methods.

Below, Section 2 introduces relevant background on LBC and Section 3 discusses work specifically related to FHCR. Section 4 explains the voting task of Etter et al. and explains why FHCR is appropriate for this task. Section 5 gives an overview of FHCR and explains the key steps needed to apply it to voting prediction. Section 6 presents our experimental results, and Section 7 concludes.

2 BACKGROUND ON LBC

Assume we are given a graph $G = (V, E, X, Y, C)$ where V is a set of nodes, E is a set of edges (links), each $\vec{x_i} \in X$ is an attribute vector for a node $v_i \in V$, each $Y_i \in Y$ is a label variable for v_i, and C is the set of possible labels. We are also given a set of "known" values Y^K for nodes $V^K \subset V$, so that $Y^K = \{y_i | v_i \in V^K\}$. Then

a common *link-based classification* (LBC) task is to infer Y^U, the values of Y_i for the remaining nodes V^U with "unknown" values ($V^U = V \setminus V^K$).

For example, given a (partially-labeled) set of interlinked university webpages, consider the task of predicting whether each page belongs to a professor or a student. The simplest (non-LBC) approach would be to learn a model that predicts the label for page v based solely on the attributes of v, such as the presence or absence of certain words. An LBC approach would instead combine these "self attributes" with some relational features, which are based on the labels of pages that link to v. For instance, it might construct a relational feature such as "Count the number of v's neighbors with label `Student`." However, this is challenging, because some labels are unknown and must be estimated, typically with an iterative process of *collective inference* [5], such as Gibbs sampling, belief propagation, or ICA (Iterative Classification Algorithm) [16].

For this paper, the most relevant collective inference approach is ICA, a simple, popular, and effective algorithm [10, 12, 16]. ICA first predicts a label for every node in V^U using only self attributes. It then constructs additional relational features X_R using the known and predicted node labels (Y^K and Y^U), and re-predicts labels for V^U using both self attributes and X_R. This process of feature computation and prediction is repeated, e.g., until convergence.

3 RELATED WORK

While there has been substantial prior work on collective *classification* [4, 5, 10, 12, 14, 16], only a few exceptions have considered collective *regression* [1, 9, 18]. Loglisci et al. [9] and Alodah & Neville [1] both use aggregation functions like MEAN or MEDIAN to aggregate information from linked neighbors into relational features that are then provided to a regression tree model for actual prediction. Alodah & Neville predict total movie revenue using a dataset from the IMDB database, while Loglisci et al. predict various target values for eight small (less than 1000 nodes) social and spatial datasets. For collective inference, Loglisci et al. essentially use ICA where the classification model is replaced with a regression tree. Alodah & Neville use this approach as well, but also do M rounds of gradient boosting, where subsequent rounds learn to predict the amount of residual error that remains from previous rounds. In contrast, Zhang et al.'s "NetCycle" [18] approach primarily concerns classification, again using ICA with relational features created by aggregation. However, their paper includes one set of experiments that perform collective regression instead of collective classification. They do so by replacing the classification model with support vector regression, without other modifications specific to regression.

Our work with FHCR differs from the above works with collective regression in two primary ways. First, we introduce and use a "fully" heterogeneous approach, where every node has its own regression model. This allows us to leverage the historical data of each region, accounting for regional differences. In contrast, the above methods use a single model for all nodes, with the exception of NetCycle, whose "partially heterogeneous" approach learns 2-4 different models for the entire dataset (e.g., one model for "authors", another for "conferences", etc.). Second, all of the above approaches use a form of "ICA with regression," where relational features somehow aggregate the information about linked neighbors, and then these features (together with attributes about each node) are provided as input to a link-unaware regression model (such as regression trees). We also consider this method (which we call RegressICA), but our results show that instead constructing a new approach that explicitly leverages the continuous nature of the target values, and their linked neighbors, yields better results. In particular, our results later show that methods based on Bayesian inference, using Gaussians to represent the learned conditional distributions, lead to more accurate results while maintaining computational tractability.

For work focused on classification, rather than regression, there have been a number of other studies that use "partially heterogeneous" approaches, similar to that used by NetCycle. For instance, Neville and Jensen [15] studies link-based classification for a "movies" domain that includes movies, producers, and studios. They learn different models for each type of node (e.g., for movie, producer, and studio), as opposed to the node-specific models that we use for our "fully heterogeneous" approach. In contrast, some other work (e.g., [7, 17]) in domains where there are multiple node types (such as papers, authors, and conferences) still learns only a single model (e.g., for papers), but the relational features are constructed based on "meta paths" that can traverse multiple types of nodes. Thus, this work also describes itself as "heterogeneous."

Our particular task of voting prediction could potentially be accomplished through methods based on "recommender systems" and/or matrix factorization. For instance, Koren et al. [8] explores the use of collaborative filtering to generate user-specific movie recommendations. Koren et al. shows how latent factorization methods can be used to infer unintuitive relationships between different sets of factors to make more accurate predictions; Etter et al. uses some such methods and we compare against them. Note, however, that the domains motivating such methods typically have very sparsely-labeled data, as a user is unlikely to, for instance, have watched a majority of all films in the target domain. In contrast, we have a dense set of historical voting records, which naturally leads to a different set of learning challenges.

4 TARGET PROBLEM & DISCUSSION

Because relational data is so commonplace, there are numerous applications for which the FHCR paradigm could be useful for making more accurate predictions. One of these applications is the prediction of voting outcomes. For this task, continuous, node-specific, and "online" results are desirable. First, continuous results are desired because, while a referendum ultimately has a single binary result ("passed" or "failed"), various stakeholders would like to be able to predict with much greater fidelity whether a referendum will pass with strong support, weak support, or not pass at all. Second, node-specific predictive models are desired because the nodes under study are regions that exhibit significant diversity, such that using a single model for all regions would substantially decrease accuracy. For FHCR, this approach is feasible wherever we have extensive information about each node (region) or small groups of nodes, as is true with our historical voting records. Finally, we would like online results, meaning that as the results for more regions are reported (e.g., on the day of an election), we

should be able to quickly make new, more precise predictions for the remaining unreported regions.

In this paper we apply FHCR to predict voting outcomes on the dataset used by Etter et al. [3]. The dataset contains voting outcomes for 281 Swiss national referendums over a period of 34 years. As with Etter et al., we use first 231 votes for training and the final 50 votes for testing. Results are reported (as the proportion of "yes" votes, between 0 and 1) for each of the 2352 regions (municipalities) in Switzerland. Each region d is described by 25 "per-region" demographic features x_d (population, population density, language spoken, location, etc.), while each vote n has 13 "per-vote" features w_n that are the recommendations ("for," "against," or no recommendation) made by major political parties. Since online prediction is desired, the task is to predict the "yes proportion," on some test vote, for every region, given the voting results for some number of "known" regions (which are assumed to have already reported their results, or to have high-quality estimates from surveying).

The "vote" and "region" features can be used for prediction. However, to compensate for the varying popularity of different votes, Etter et al. mean-centers each training and test vote, using the true national average for the training votes and the estimated national average (computed using only the "known" regions) for the test votes. Thus, the possible values for prediction range from -1.0 to 1.0. We use this same approach.

Etter et al. used a four factor model to predict continuous outputs for regional votes. This model incorporated a bias term for each vote, a regression term based on "region" features, a regression term based on "vote" features, and a matrix factorization model based off of latent features and the set of votes. However, Etter et al. did not use any LBC or collective regression. In particular, Etter et al. does not treat the data as a graph or explicitly use links.

5 PREDICTION WITH FHCR

FHCR follows a pattern similar to that of ICA as described in Section 2: initial predictions are made using features, then a collective inference procedure iteratively updates those predictions for each region based on its relevant features and the predictions of *linked* regions. This section thus studies the following key questions required to make FHCR effective:

(1) How to "bootstrap" the initial predictions to provide a baseline for our inference?
(2) How should links be computed, so as to connect the regions into a useful graph?
(3) Given the initial predictions and informative links, how can we perform effective collective inference?

5.1 Methods for "Bootstrap" Prediction

Before collective inference can be used for a particular test vote n, we must compute a set of initial "bootstrap" predictions $y_d^{(0)}$ for each region d. These are computed using the vote and/or region features, but *without* the use of links (see summary in Table 1). Some methods used by Etter et al. are suitable for this task, so we consider those first:

• LIN(v) - linear regression based only on the vote features. We train each region's model by comparing the 13 political party recommendations w_m for each training vote m to that region's voting outcome, yielding the following predictions for test vote n:

$$y_d^{(0)} = \gamma_d^T w_n$$

where γ_d is a weight vector specific to region d.

• LIN(R) - regression based only on the region features. For test vote n, we use a vote-specific model that takes x_d (the 25 features of region d) as input, with coefficients β_n learned using only the "known" region results for vote n. Then for inference, $y_d^{(0)} = \beta_n^T x_d$.

• JOINT(R,V) - regression based on the region features and the vote features. This approach uses alternating least squares to learn a joint model:

$$y_d^{(0)} = \beta_n^T x_d + \gamma_d^T w_n.$$

Etter et al. called this model LIN(R)+LIN(V).

Table 1: Summary of features and regression types used in various "bootstrap" models.

Model	Use vote feats?	Use region feats?	Combination Method
LIN(V)	Yes	No	N/A
LIN(R)	No	Yes	N/A
JOINT(R,V)	Yes	Yes	Alternating Least Squares
BAYESD+INDPT(R,V)	Yes	Yes	Approximate Bayesian Inference
BAYESG+INDPT(R,V)	Yes	Yes	Exact Bayesian Inference
JBG-ENSEMBLE	Yes	Yes	ALS and Bayesian Inference

New methods: Of the above methods, Etter et al. generally showed that using both sets of features performed best. However, the linear combination approach of JOINT(R,V) is not necessarily optimal. In response, we created three new bootstrap methods that combine region and vote features in different ways:

• **BAYESD+INDPT(R,V)** - This method performs Bayesian Inference using conditional distributions for the vote and region features, learned independently. First, we use Bayes rule to calculate the probability that the prediction for a particular region, y_d, is a certain value y, given the region features x_d and the vote features w_n:

$$\begin{aligned} P(y_d = y|x_d,w_n) &= \frac{P(x_d, w_n|y_d = y) \cdot P(y_d = y)}{P(x_d, w_n)} \\ &\propto P(x_d|y_d = y) \cdot P(w_n|y_d = y) \cdot P(y_d = y) \\ &\propto \frac{P(y_d = y|x_d) \cdot P(y_d = y|w_n)}{P(y_d = y)} \end{aligned} \quad (1)$$

where the second line assumes conditional independence and the third line re-applies Bayes rule to the first two terms.

To estimate $P(y_d = y|w_n)$, we apply the LIN(v) model to the training data. Then, we generate a discrete histogram (we use 500 uniform-sized bins) based on the error between the true training values and predictions made with LIN(v), for all regions and training votes. This learns a *single* histogram for all regions, which can then be applied during inference for a *particular* region (as $P(y_d = y|w_n)$) by shifting the mean of the histogram to the value predicted by LIN(v). $P(y_d = y|x_d)$ uses a similar same process but with LIN(R), where learning uses all training votes but only the regions that will be "known" during inference. Finally, the actual bootstrap step finds the most likely value of y_d via Equation 1 and a discrete estimate of the following integral:

$$y_d^{(0)} = \int_{-1}^{1} y \cdot P(y_d = y|x_d, w_n)dy. \quad (2)$$

• **BAYESG+INDPT(R,V)** - this method also uses Bayesian Inference to leverage the vote and region features, via Equation 1. However,

by approximating the conditional probabilities as Gaussians, we enable exact inference, instead of resorting to a *discrete* approximation (hence, the "G" vice "D" in the name). In particular, we substitute normal distributions for each probability, yielding the following:

$$P(y_d = y|x_d, w_n) \propto \frac{\mathcal{N}(\beta_n^T x_d, \widehat{\sigma}_r^2) \cdot \mathcal{N}(\gamma_d^T w_n, \widehat{\sigma}_v^2)}{\mathcal{N}(0, \widehat{\sigma}_{prior}^2)}.$$

In this equation, each mean is computed based on the results from LIN(R) or LIN(V), or is known to be zero (due to mean-centering of the data). The variances (σ_r^2, σ_v^2, and σ_{prior}^2) can be estimated by comparing predictions to actual values in the training data. With this form, the most likely value for y_d can be computed exactly, because the product of two Gaussians f and g is also a Gaussian [2, 6], with the following mean and variance:

$$\mu_{fg} = \frac{\mu_f \sigma_g^2 + \mu_g \sigma_f^2}{\sigma_f^2 + \sigma_g^2} \quad \text{and} \quad \sigma_{fg}^2 = \frac{\sigma_f^2 \sigma_g^2}{\sigma_f^2 + \sigma_g^2} \tag{3}$$

• **JBG-Ensemble** - a combination of the previous two methods. We average the predictions generated by the Joint(R,V) and the BayesG+Indpt(R,V) models, potentially mitigating the impact of outliers predicted by either model. Later results will show this can be very effective, while remaining computationally tractable.

5.2 Effective Link Generation

Target uses of FHCR may involve nodes, such as people in a social network, for which links already exist or are obvious. In other cases, as with our voting domain, there may be no explicit links, in which case a key task is to construct informative links to connect the nodes (i.e., regions), in a way that facilitates subsequent inference.

In our data, regions are identified by latitude/longitude, and thus one natural idea is to link all regions that are within a certain distance, such as 5 kilometers. Distance can also be inverted and scaled to produce link weights, so that "closer" regions have more influence. Section 6 shows that both of these ideas can be effective.

For voting prediction, however, we hypothesized that linking regions based on similar voting histories would prove even more effective than links based on geographic proximity. Thus, we examined the 231 training votes and computed the Pearson correlation coefficient between all pairs of regions. We then consider methods based on linking, for each region, the k most-similar regions, or all regions with a correlation greater than some threshold (e.g., 0.80).

5.3 Collective (Link-based) Regression

Regression techniques are well established, but collective regression requires some kind of iterative procedure to allow the predicted and known values to propagate throughout the links. In each case, the initial predicted values ($y_d^{(0)}$) are set via a bootstrap method from Section 5.1. For inference, we first consider two baseline methods, which are simple extensions of existing LBC methods to regression:

• **RegressICA** - This method uses ICA, described in Section 2, but with an underlying model of regression rather than classification. Prediction for region d is based on d's vote features and, as a relational feature, the average predicted value for d's neighbors. For each iteration i (after bootstrap), predictions are computed as $y_d^{(i)} = f(w_n, \frac{\sum_{j \in L_d} y_j^{(i-1)}}{|L_d|})$ where f is a learned (linear) regression model and L_d is the set of regions that link to region d. We use 10 iterations; more did not improve accuracy. When links are weighted, the neighbor average is a weighted computation, but for ease of explication we omit all such "link weight" details in this section.

• **WeightedVec** - Instead of using the neighbor average as a feature for regression, this method uses the neighbor average as its actual *prediction*. Thus, $y_d^{(i)} = \frac{\sum_{j \in L_d} y_j^{(i-1)}}{|L_d|}$. This method is analogous to the "weighted vote relational neighbor" (WVRN) method of Macskassy and Provost [11], but adapted for regression instead of classification.

New methods: We also created the following new methods for collective inference:

• **BayesD+Links** - this method seeks to use Bayesian Inference to estimate the most likely prediction for y_d, given only the predicted (and known) values of region d's linked neighbors, y_{L_d}. This can be computed as follows:

$$\begin{aligned} P(y_d = y|y_{L_d}) &= \frac{P(y_d = y) \cdot P(y_{L_d}|y_d = y)}{P(y_{L_d})} \\ &\propto P(y_d = y) \cdot \prod_{j \in L_d} P(y_j|y_d = y) \end{aligned}$$

where the last step assumes conditional independence of the neighbors' values given y_d.

In general, estimating $P(y_j|y_d = y)$ from the training data could be a challenging task. However, we conjecture that for our purposes this probability can be approximated as a function solely of the difference between y_j and y_d.[1] That idea yields

$$P(y_d = y|y_{L_d}) \propto P(y_d = y) \cdot \prod_{j \in L_d} \phi(y_j - y_d) \tag{4}$$

where $\phi(x)$ is some arbitrary function. To approximate $\phi(x)$, we first calculate, for each region d and training vote, the difference between the mean-centered voting outcome for d and those of each of its linked neighbors. We then add these values to a histogram (again using 500 uniform-sized bins) to produce a discrete estimate of $\phi(x)$ (shared across all regions). For inference, we can then compute the most likely value of $P(y_d = y|y_{L_d})$ via an approximation of

$$y_d^{(i)} = \int_{-1}^{1} y \cdot P(y_d = y|y_{L_d}^{(i-1)}) dy.$$

This is similar to our previous use of Equation 2, but now predicting with d's neighbors instead of d's features.

• **BayesG+Links** - This method is similar to BayesD+Links, but approximates the link-based conditional probabilities with Gaussians, enabling exact inference via variants of Equation 3 (see [2]). With this approach, Equation 4 becomes

$$P(y_d = y|y_{L_d}) \propto \mathcal{N}(0, \widehat{\sigma}_{prior}^2) \cdot \prod_{j \in L_d} \mathcal{N}(y_j, \widehat{\sigma}_{links}^2)$$

The first term is the prior, with assumed mean zero, and variance estimated from the training data. The link-based Gaussian for region

[1] For example, we estimate that the likelihood of y_j being 0.3 given y_d is 0.25 is approximately the same as the likelihood of y_j being 0.4 given y_d is 0.35. This would be approximately true if similar (linked) regions tend to vote in the same ways.

j, which replaces $\phi(y_j - y_d)$, uses a mean of y_j (the predicted or known value for j), and variance $\widehat{\sigma}^2_{links}$ (based on fitting $\phi(y_j - y_d)$ to a Gaussian with the training votes).

This method has the advantage of yielding much faster inference than BayesD+Links (e.g., about 40 times faster in our experiments). In addition, it avoids errors that result from discretization. However, do the Gaussians actually represent the conditional probabilities of Equation 4 in a way that is effective for inference? Section 6 evaluates their impact.

Incorporating features: The methods discussed above (except for RegressICA) have used only the "link" information for their inference. The following new methods all seek to improve prediction accuracy by also exploiting the vote and/or region features.

• **WeightedVec+Delta** - this method uses the WeightedVec approach, but modified to also leverage vote-based features. In particular, we learn a region-specific "delta" correction, based on the training error between each region's true value and the value predicted by WeightedVec, yielding

$$y_d^{(i)} = \frac{\sum_{j \in L_d} y_j^{(i-1)}}{|L_d|} + f_d(w_n)$$

where f_d uses the vote features for prediction. This relates to the residual learning of Alodah & Neville [1].

• **BayesD+Indpt(r,v)+Links** - while similar to BayesD+Links, this method takes region and vote features into consideration in addition to the links. Following a similar derivation as before yields

$$P(y_d = y | x_d, w_n, y_{L_d}) \propto \frac{P(y_d=y|x_d) \cdot P(y_d=y|w_n) \cdot \prod_{j \in L_d} \phi(y_j - y_d)}{P(y_d=y)} \quad (5)$$

where the three conditional probabilities can be estimated as previously discussed.

• **BayesG+Indpt(r,v)+Links** - this method uses the same approach as above, but approximates the probability distributions as Gaussians. Therefore,

$$P(y_d = y | x_d, w_n, y_{L_d}) \propto \frac{\mathcal{N}(\beta_n^T x_d, \widehat{\sigma}_r^2) \cdot \mathcal{N}(\gamma_d^T w_n, \widehat{\sigma}_v^2) \cdot \prod_{j \in L_d} \mathcal{N}(y_j, \widehat{\sigma}^2_{links})}{\mathcal{N}(0, \widehat{\sigma}^2_{prior})} \quad (6)$$

• **BayesG+Joint(r,v)+Links** - this method uses an inference strategy similar to BayesG+Indpt(r,v)+Links. Above, we assumed conditional independence for the conditional probabilities associated with the region and the vote features. However, the region and vote features may have interactions. Thus, instead of using two separate Gaussians for these features, we use one Gaussian based on the Joint(r,v) implementation of the region and vote features. Collapsing these two terms into one also causes the denominator to cancel, yielding

$$P(y_d = y | x_d, w_n, y_{L_d}) \propto \mathcal{N}(\beta_n^T x_d + \gamma_d^T w_n, \widehat{\sigma}^2_{joint}) \cdot \prod_{j \in L_d} \mathcal{N}(y_j, \widehat{\sigma}^2_{links})$$

• **BayesG+JGB-Ensemble+Links** - this method is nearly identical to the previous method, except that the mean used for the first Gaussian is computed based on the prediction from the JBG-Ensemble bootstrap, with variance based on averaging the variance of the two Gaussians arising from the two parts in the ensemble.

Table 2: Average RMSE after running *only* the bootstrap step, with no collective inference. The best result for each column is in bold.

	Number of Known Regions, N_k							
Bootstrap Method	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
LIN(v)	**11.70**	8.89	8.48	8.02	7.95	7.91	7.91	7.90
LIN(r)	12.89	9.20	8.44	7.03	6.62	6.16	6.08	6.03
Joint(r,v)	12.40	9.15	8.32	6.76	6.36	5.90	5.82	5.76
BayesD+Indpt(r,v)	11.89	8.52	7.88	6.83	6.54	6.21	6.16	6.46
BayesG+Indpt(r,v)	12.08	**8.19**	**7.57**	6.56	6.30	5.99	5.93	5.89
JBG-Ensemble	11.74	8.36	7.67	**6.49**	**6.18**	**5.82**	**5.75**	**5.70**

6 RESULTS

Below, we study how to use FHCR most effectively via appropriate choices for bootstrap, link generation, and collective inference.

Our experiments replicate the experimental conditions used by Etter et al., using the data and conditions described in Section 4. For a given test vote, the task is to predict the "yes percentage" for each region, given the results for some subset of regions (thus simulating a prediction task with partial "day of voting" results); there are N_K such "known" regions. For each trial, these regions are selected by first choosing a random "reveal order" for all 2352 regions. The first N_k regions are considered "known", while the last 10% are reserved as "evaluation regions." Performance is evaluated by measuring the "root mean squared error" (RMSE) over the evaluation regions, averaged over 60 trials. Each trial uses the same reveal order for all methods, and we used Etter et al.'s original code to produce results with their methods.

6.1 Bootstrap

Table 2 shows the RMSE values vs. the number of known regions (N_k) as we vary the bootstrap method; lower values are better. In general, error decreases as N_k increase for two reasons. First, methods that use the region features, such as LIN(r), use the known values for the given *test* vote to learn their classifier, and learning with more such results decreases error. However, error also decreases for LIN(v), whose classifier is learned solely from vote-based features with 231 fully-observed training votes (and thus not directly affected by N_k), because the predicted mean for test vote n is estimated from the N_k known regions (see Section 4). A larger N_k produces a better estimate, which helps all models.

We first consider the three methods from Etter et al. (first three rows of Table 2). When there are very few known regions ($N_K < 10$), using only the vote features, with LIN(v), performs best. In contrast, using only the region features has higher error, due to the small number of regions available for training LIN(r), as discussed above. LIN(r) improves rapidly as N_k increases, so that it outperforms LIN(v) for $N_k \geq 10$. However, using *both* the region and vote features, with Joint(r,v), performs even better, for $N_k \geq 10$.

We next consider our three new bootstrap methods (last three rows of Table 2), which all use some Bayesian inference to combine the region and vote features. For all cases except $N_k = 1$, the Gaussian approximation with exact inference (BayesG+Indpt(r,v)) yields better RMSE than the discrete version (BayesD+Indpt(r,v)). Moreover, combining these predictions with those of Joint(r,v) (with JBG-Ensemble) yields even lower error, when $N_K \geq 50$. Overall, the last two rows of Table 2 show that our new methods combining the vote and region features with Bayesian inference succeed in reducing the error, compared to the best previous alternatives,

LIN(v) and JOINT(r,v). This demonstrates the general utility of the Bayesian inference method for combining disparate sources of information, as will be further demonstrated in Section 6.3.

6.2 Link Generation

Table 3 shows the results of collective inference, with different link generation methods. All cases use 10 iterations of REGRESSICA; the next section considers different collective inference strategies. Based on the results of the prior section, we use JBG-ENSEMBLE for bootstrap, though results (not shown) using BAYESG+INDPT(r,v) instead yielded very similar results.

The top section of Table 3 shows results with proximity-based links, with distance thresholds of 5, 10, and 20 kilometers. Of these, Prox-10km works best (except for $N_k = 1$), by striking a good balance between having too many links (with a large threshold) and having too few. Adding link weighting based on nearness (with Prox-10km-Wt) does not consistently improve the performance.

The second section of Table 3 shows results with links chosen by correlation, e.g., linking all pairs of regions with a correlation of at least 0.60 or 0.80. Picking many links with the lower threshold (Corr-0.60) performs better when many regions are "known" ($N_k > 500$); in this case, the high value of N_k reduces prediction uncertainty, and averaging over many links is helpful. In contrast, the higher threshold (Corr-0.80) performs better when $N_k \leq 500$, indicating that having a smaller number of strongly correlated links is more effective. The problem with this approach, for some cases, is that using a high threshold causes some regions to have only a few links. In response, the next two rows establish a minimum number of links for each region. This method links region d to all other regions with correlation at least 0.80, or, if there are not enough such regions to meet the minimum, to the top 10 or 20 most strongly correlated regions. With this strategy, Corr-0.80-Min-10 almost always improves over Corr-0.80 and Corr-0.60. For instance, for $N_k = 1000$ the RMSE improves from 6.40 with Corr-0.80 to to 5.49.

Encouraged by the success of setting a minimum number of links, the third section of Table 3 thus considers linking each region to its top $k = 10$ or $k = 100$ correlated other regions. Using 10 links for each region (Corr-10) almost always performs a little better than the methods discussed above, and substantially better than using 100 links (Corr-100). For instance, for $N_k = 1000$, Corr-10 has RMSE of 5.46, while Corr-100 has RMSE of 5.98. The last two rows of Table 3 show results where the links are weighted based on the inter-region correlation values. While leading to marginal improvements in some cases, we find that weighting links in this case makes minimal difference overall. For Corr-10-Wt, this is because there is little variation in the correlation values when only 10 links are chosen, so link weighting makes little difference. For Corr-100-Wt, the differences (vs. Corr-100) are larger but still quite small. Future work should consider whether a different correlation formula (instead of Pearson's) might provide values more useful for this task, and make the final results less sensitive to the number of links chosen.

Overall, we find that we find that Corr-10-Wt is the most effective general strategy. While Prox-10km (and sometimes Prox-5km) performs best for $N_k \leq 10$, Corr-10-Wt is still an effective strategy for these values of N_k. Thus, we chose Corr-10-Wt as our linking strategy for the next sections.

Table 3: Average RMSE of various link generation strategies when running RegressICA with the ENSEMBLE bootstrap.

	Number of Known Regions, N_k							
Linking Method	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
Prox-20km	12.48	9.11	8.42	7.38	7.06	6.29	5.98	5.83
Prox-10km	11.84	**8.52**	**7.85**	6.77	6.47	5.97	5.80	5.65
Prox-5km	**11.78**	8.69	8.12	7.22	6.97	6.40	6.17	5.98
Prox-10km-Wt	11.81	8.53	7.87	6.79	6.49	5.97	5.79	5.60
Corr-0.80	12.21	9.01	8.29	7.15	6.88	6.51	6.40	6.30
Corr-0.60	13.85	10.02	9.16	7.90	7.64	6.50	5.98	5.70
Corr-0.80-Min-10	12.33	8.92	8.08	6.61	6.25	5.70	5.49	5.31
Corr-0.80-Min-20	12.45	8.99	8.15	6.67	6.32	5.79	5.56	5.32
Corr-10	12.20	8.87	8.08	**6.64**	**6.24**	5.66	**5.46**	5.29
Corr-100	14.09	10.29	9.39	8.07	7.79	6.58	5.98	5.63
Corr-10-Wt	12.20	8.87	8.08	**6.64**	**6.24**	**5.65**	**5.46**	**5.28**
Corr-100-Wt	14.06	10.25	9.35	8.00	7.72	6.54	5.94	5.60

6.3 Collective Inference

Table 4(a) shows results of varying the collective inference method, using JBG-ENSEMBLE for bootstrap and Corr-10-Wt for link generation. All methods use 10 iteration of collective inference.

REGRESSICA and WEIGHTEDVEC can be considered "baseline" methods for FHCR that are adapted from typical approaches to link-based classification. Since REGRESSICA uses features *and* links, it performs better than WEIGHTEDVEC (which use only links), except for $N_K \leq 5$. Nonetheless, WEIGHTEDVEC also obtains accuracy close to that of REGRESSICA when N_k is very high; this is reminiscent of WVRN's strong classification performance, compared to more complex methods, for more densely-labeled graphs [11].

The second section of Table 4(a) compares the three "link only" methods: WEIGHTEDVEC, BAYESD+LINKS, and BAYESG+LINKS.[2] In every case, our use of Bayesian inference decreases the error compared to the simple averaging approach of WEIGHTEDVEC (by 0.03 to 0.19), and the Gaussian approximation with BAYESG+LINKS further reduces error compared to the discrete approximation with BAYESD+LINKS (by 0.01 to 0.10). In addition to reducing error, inference with the Gaussians also executes about 40 times faster than with the discrete version.

The third section of Table 4(a) adds the use of vote and/or region features to the three "link-only" methods. With WEIGHTEDVEC+DELTA, this consistently reduces the error, sometimes substantially (e.g., from 5.94 to 5.50 when $N_k = 500$). With the four methods based on BayesD and BayesG, however, the results are more muted. While adding feature information during inference (with INDPT(r,v), JOINT(r,v), or JBG-ENSEMBLE) generally improves accuracy compared to BAYESD+LINKS and BAYESG+LINKS, the error improves by at most 0.23 and sometimes gets slightly worse. Thus, while adding feature information generally improves the collective inference, with the Bayesian inference it is surprising that this addition is not more helpful. The next section addresses this issue.

6.4 Improving Inference via Variance Scaling

The previous section observed that incorporating the region and vote features into our inference was beneficial. However, the features were significantly less helpful for Bayesian inference than they were for WEIGHTEDVEC. We conjecture that this indicates that the Bayesian methods are not balancing, as effectively, the

[2]Note that these "link only" inference methods still use the feature information during bootstrap.

Table 4: RMSE for various collective inference methods, using JBG-Ensemble bootstrap and Corr-10-Wt linking. Within each column, we bold the best results. Part (a) of the table uses methods from Section 5.3, while part (b) adds "variance scaling" (see Section 6.4).

	Number of Known Regions, N_k							
Inference Method	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
(a) Proposed FHCR methods (without variance scaling)								
RegressICA	12.20	8.87	8.08	6.64	6.24	5.65	5.46	5.28
WeightedVec	12.07	8.84	8.14	6.98	6.63	5.94	5.62	5.33
BayesD+Links	11.90	8.59	7.95	6.91	6.59	5.91	5.59	5.30
BayesG+Links	11.87	8.57	7.94	6.87	6.52	5.81	5.52	5.27
WeightedVec+Delta	11.98	8.65	7.88	6.48	**6.09**	5.50	5.31	**5.15**
BayesD+Indpt(r,v)+Links	11.88	8.51	7.88	6.91	6.61	5.93	5.60	5.30
BayesG+Indpt(r,v)+Links	11.86	8.48	7.86	6.86	6.53	5.82	5.53	5.27
BayesG+Joint(r,v)+Links	11.88	8.60	7.94	6.74	6.36	5.69	5.44	5.22
BayesG+JGB-Ensemble+Links	11.80	8.46	7.81	6.64	6.31	5.78	5.59	5.59
(b) BayesG methods with addition of variance scaling								
BayesG+Indpt(r,v)+Links	11.82	**8.16**	**7.53**	6.45	6.10	5.56	5.40	5.26
BayesG+Joint(r,v)+Links	11.80	8.55	7.86	6.51	6.11	**5.48**	**5.30**	5.20
BayesG+JGB-Ensemble+Links	**11.69**	8.33	7.64	**6.44**	6.13	5.64	5.53	5.59

link-based and the feature-based information. In addition, careful comparison with Table 2 shows that, so far, the best results with collective inference are actually *worse* than the best results with just bootstrap, when $N_K \leq 10$ (for instance, for $N_k = 5$, BayesG+JGB-Ensemble+Links has RMSE of 8.46, but "bootstrap-only" BayesG+Indpt(r,v) obtains 8.19). Thus, when there is more prediction uncertainty (due to fewer "known" regions), the Bayesian collective inference is sometimes doing more harm than good.

We conjecture that the problem with the Bayesian methods is that they treat all linked neighbors as being equally "reliable" for prediction. In order to improve our Bayesian inference, we must somehow teach it to discriminate between certain and uncertain values, "known" and "unknown" neighbor predictions. This would also allow a region's features to have more weight when its linked neighbors are less certain, which could help for the "small N_K" scenario.

Fortunately, certain properties of our BayesG approach can enable us to introduce this "discrimination," as we now describe. Equation 3 shows that, when combining two Gaussians, the Gaussian with the *smaller* variance will have more impact on the resulting mean. However, Equation 6 uses the same term, $\mathcal{N}(y_j, \widehat{\sigma}^2_{links})$ for each neighbor. In this term, the mean for each neighbor varies (based on the predicted or known value y_j), but the variance is the same, regardless of the certainty of y_j. This variance is learned from the training data, where there are no prediction errors, and thus is appropriate for a link to a "known" region. When linking to a "predicted" region, however, this estimate likely underestimates the true variance. Therefore, we propose to scale the variance used for "unknown" regions as follows:

$$\widehat{\sigma}^2_{linksToUnknownRegions} = K_U \cdot \widehat{\sigma}^2_{links}$$

where K_U is a constant learned via cross-validation. In particular, we evaluate the RMSE obtained via collective inference on the last 15 votes of the training data, using $K_U \in \{2, 4, 8\}$ and select the K_U that minimizes RMSE on this validation data. We then use that K_U for inference on the actual test data and report results. With $K_U > 1$, the larger variance will lessen the predictive influence of regions with uncertain predictions.

Table 4(b) shows the results for three Bayesian inference variants with the addition of this "variance scaling." As desired, the use of variance scaling leads to substantial error reductions in many cases. For instance, BayesG+Indpt(r,v)+Links improves from 8.48 to 8.16, for $N_k = 5$, which is the best for any method in Table 2 or Table 4. Overall, this eliminates the situation where "bootstrap" only results outperformed the best collective inference, and in general some form of Bayesian inference with variance scaling now performs best among the FHCR methods that we consider in Table 4, with two slight exceptions at $N_k = 100$ and $N_k = 2116$.

6.5 Discussion and Comparison

Table 5 compares our best results (using Bayesian inference with variance scaling) with the four methods that Etter et al. used for their in-depth comparisons. The first two methods are baselines: BIAS, which simply uses the average of all "known" results as its prediction, and the previously discussed LIN(v), which uses regression on the vote features. The next two methods are those that Etter founded performed best: MF+GP(r) and MF+GP(r)+LIN(v). Both use matrix factorization (MF) combined with a Gaussian process (GP) for the region features.

The left side of Table 5 reports per-region RMSE results as used elsewhere in this paper, while the right side shows the national "binary error rate." To compute this metric, which Etter et al. also considered, we first use the per-region predictions weighted by region population to compute an overall "national" prediction; values greater than 0.5 yield a prediction of "pass" as the overall vote outcome. The error rate is then the fraction of votes that are incorrectly classified as "pass" vs. "fail." The error rate slightly increases when N_K is very high; Etter et al. attributes this effect to weighting the final prediction by each region's population, as opposed to the actual "day of voting" turnout, which is not *a priori* observable.

Overall, Table 5 shows that FHCR is highly effective at the voting prediction task. In particular, our new methods (and Etter's) consistently decrease the errors compared to the BIAS and LIN(v) baselines. Moreover, when N_K is smaller, one of our new FHCR methods produces the best results, both for per-region RMSE and the national error rate results. For per-region RMSE, our methods are somewhat better than Etter's (by 0.01-0.19), when $N_k < 50$, while they produce competitive results (lagging by at most 0.41 RMSE) for larger values of N_k. Interestingly, our methods do even better (relatively) on the national error rate results, suggesting that they tend to do especially well on more populous regions (which have a larger effect on the national outcome). In particular, while the Etter methods perform best when $N_K \geq 500$, for all $N_K < 500$, our new BayesG+Indpt(r,v)+Links (with variance scaling) yields the lowest national error of any method. For instance, when only 100 regions have "known" results, this method has an error rate of just 0.87%, while the best Etter method has an error rate of 1.23%.

Even more significantly, our new methods based on Bayesian inference are *much* faster than Etter's best methods. In particular, running on an Intel i7-4600M 2.90 GHz CPU, Etter et al.'s MF+GP(r)+LIN(v) takes about 480 minutes to fully train and provide one trial of inference for one vote, while MF+GP(r) requires 3360 minutes.[3] In contrast, using our models based on BayesG, including cross-validation for variance scaling, requires only 4.25 minutes, a speedup of 110-790 times vs. the Etter models. Note

[3]For both our code and Etter et al.'s, the runtime is dominated by training the model, while inference runs much more quickly. The simpler MF+GP(r) took longer (vs. MF+GP(r)+LIN(v)) because more learning iterations were required.

Table 5: Comparison of the best results with FHCR (first three rows) vs. two baseline methods from Etter et al. (next two rows) and the best results from Etter et al. (last two rows). On left, we report average RMSE results for each region, as in the rest of the paper. On right, we show binary error percentages, where goal is to predict the overall outcome (pass/fail) for each vote. Within each column, we bold the best result.

	Per-region results (RMSE)								National results (binary error percentage)							
Number of Known Regions, N_k:	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
New FHCR methods based on BayesG (with variance scaling)																
BayesG+Indpt(r,v)+Links	11.82	**8.16**	**7.53**	6.45	6.10	5.56	5.40	5.26	**14.33**	**6.40**	**4.47**	**1.53**	**0.87**	0.80	0.93	1.97
BayesG+Joint(r,v)+Links	11.80	8.55	7.86	6.51	6.11	5.48	5.30	5.20	14.70	7.87	5.63	2.20	1.27	1.17	1.20	1.97
BayesG+JGB-Ensemble+Links	**11.69**	8.33	7.64	6.44	6.13	5.64	5.53	5.59	14.40	7.03	4.87	1.60	1.03	1.33	1.30	2.33
Methods from Etter et al. [3]																
BIAS	12.89	10.34	9.98	9.60	9.54	9.51	9.50	9.50	15.63	10.00	8.30	6.67	6.00	4.17	3.80	2.00
LIN(v)	11.70	8.89	8.48	8.02	7.95	7.91	7.91	7.90	14.47	8.50	5.90	2.33	1.97	1.33	1.43	1.93
MF+GP(r)	12.89	8.90	7.76	**6.03**	**5.68**	**5.18**	**5.01**	**4.87**	15.63	8.60	5.53	1.83	1.23	**0.43**	0.80	1.97
MF+GP(r)+LIN(v)	11.84	8.35	7.54	6.20	5.86	5.37	5.23	5.15	14.60	7.83	5.30	2.47	1.97	0.67	**0.67**	**1.90**

that this faster time is much more amenable to online and dynamic analysis, as useful for "day of voting" predictions. Thus, compared to prior work, our new FHCR methods obtain comparable or better accuracy, especially when relatively few regions have reported results, while executing at least two orders of magnitude faster.

7 CONCLUSION

Link-based classification (LBC) has been shown to substantially improve accuracy in a variety of domains, but prior to this undertaking had rarely been applied to continuous domains, and never to heterogeneous domains where rich temporal/historical information could potentially enable node-specific models. Our results show that extending the LBC paradigm to *fully heterogeneous collective regression* (FHCR) indeed addresses these limitations of LBC.

We have developed techniques that are efficient and effective. In particular, we introduced novel methods that combine feature-based and link-based information using Bayesian inference. We showed that these methods always outperformed extensions of LBC methods like ICA, and almost always outperformed multiple versions based on neighbor averaging (e.g., as with WVRN [11]). Moreover, we demonstrated results that are comparable to or slightly better than the previous link-unaware methods used by Etter et al., especially when the results from only a small number of regions are known — yet our methods are *at least* two orders of magnitude faster. This greatly facilitates *online* prediction of results, for example for updated "day of voting" results, or for pre-voting result forecasting, where resources are available to extensively canvas only a small fraction of voting regions.

Future work should consider further improvements to these methods. For instance, we demonstrated that "variance scaling" was highly effective at reducing error for our methods based on Bayesian inference Future work should study further refinements, such as varying the effective variance used for each region based on an estimated confidence for its current prediction [13].

Some aspects of our proposed methods are specific to the voting prediction task that we used, such as combining information from two distinct feature sets (the "per vote" and "per region" features) that required distinct learning strategies. Other domains where FHCR could be useful may not have such feature diversity. However, our methods using Bayesian inference to combine the information from a varying number of linked neighbors, and to combine that information with feature-based predictions, are applicable in other settings, such as per-region sales prediction, weather forecasting, crowd-sourcing predictions [18], and other time-varying social phenomena. Future work should apply FHCR to these domains.

ACKNOWLEDGEMENTS

We thank Edward Raff and the anonymous referees for comments that helped improve this work. Grants/resources from ONR and the DoD HPC Modernization Office supported this work in part.

REFERENCES

[1] Iman Alodah and Jennifer Neville. 2016. Combining Gradient Boosting Machines with Collective Inference to Predict Continuous Values. In *Proceedings of the 6th International Workshop on Statistical Relational AI*.
[2] Paul Bromiley. 2003. Products and convolutions of Gaussian probability density functions. *Tina-Vision Memo* 3, 4 (2003), 1.
[3] Vincent Etter, Mohammad Emtiyaz Khan, Matthias Grossglauser, and Patrick Thiran. 2016. Online Collaborative Prediction of Regional Vote Results. In *Proc. of DSAA*. IEEE, 233–242.
[4] Brian Gallagher, Hanghang Tong, Tina Eliassi-Rad, and Christos Faloutsos. 2008. Using ghost edges for classification in sparsely labeled networks. In *Proc. of KDD*. 256–264.
[5] David Jensen, Jennifer Neville, and Brian Gallagher. 2004. Why collective inference improves relational classification. In *Proc. of KDD*. ACM, 593–598.
[6] Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering* 82, 1 (1960), 35–45.
[7] Xiangnan Kong, Philip S Yu, Ying Ding, and David J Wild. 2012. Meta path-based collective classification in heterogeneous information networks. In *Proc. of CIKM*. 1567–1571.
[8] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).
[9] Corrado Loglisci, Annalisa Appice, and Donato Malerba. 2016. Collective regression for handling autocorrelation of network data in a transductive setting. *Journal of Intelligent Information Systems* 46, 3 (2016), 447–472.
[10] Qing Lu and Lise Getoor. 2003. Link-based classification. In *Proc. of ICML*. 496–503.
[11] Sofus A Macskassy and Foster Provost. 2007. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research* 8, May (2007), 935–983.
[12] Luke McDowell and David Aha. 2012. Semi-supervised collective classification via hybrid label regularization. In *Proc. of ICML*.
[13] Luke K. McDowell, Kalyan M. Gupta, and David W. Aha. 2009. Cautious collective classification. *Journal of Machine Learning Research* 10 (2009), 2777–2836.
[14] Jennifer Neville and David Jensen. 2000. Iterative Classification in Relational Data. In *Proc. of the Workshop on Learning Statistical Models from Relational Data at AAAI-2000*.
[15] Jennifer Neville and David Jensen. 2007. Relational dependency networks. *Journal of Machine Learning Research* 8, Mar (2007), 653–692.
[16] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93.
[17] Yizhou Zhang, Yun Xiong, Xiangnan Kong, Shanshan Li, Jinhong Mi, and Yangyong Zhu. 2018. Deep Collective Classification in Heterogeneous Information Networks. In *Proc. of WWW*. 399–408.
[18] Yizhou Zhang, Yun Xiong, Xiangnan Kong, and Yangyong Zhu. 2016. Netcycle: Collective evolution inference in heterogeneous information networks. In *Proc. of KDD*. 1365–1374.