A TRIDENT SCHOLAR PROJECT REPORT

NO. 472

Prediction of Regional Voting Outcomes Using Heterogeneous Collective Regression

by

Midshipman 1/C David J. Liedtka, USN

UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

This document has been approved for public release and sale; its distribution is unlimited.

USNA-1531-2

U.S.N.A. --- Trident Scholar project report; no. 472 (2018)

PREDICTION OF REGIONAL VOTING OUTCOMES USING HETEROGENEOUS COLLECTIVE REGRESSION

by

Midshipman 1/C David J. Liedtka
United States Naval Academy
Annapolis, Maryland

_____
(signature)

Certification of Adviser Approval

Professor Luke K. McDowell
Computer Science Department

_____
(signature)

_____
(date)

Acceptance for the Trident Scholar Committee

Professor Maria J. Schroeder
Associate Director of Midshipman Research

_____
(signature)

_____
(date)

USNA-1531-2

Abstract

Increasingly, many important domains in the world can be viewed as networks of linked nodes: people connected by social network "friendships," webpages connected by hyperlinks, and even geo-political areas connected by proximity and common interests. To leverage these links for prediction and analysis tasks, Machine Learning researchers have developed multiple techniques for link-based classification (LBC). 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 and the prediction of continuous outputs have 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., using Swiss referendum outcomes, reported high accuracy, even when only 5% of voting "regions" had reported results. This study used a collaborative filtering approach to implicitly leverage the correlation present between "nearby" regions. They did not, however, consider formulating the regions as a network.

This project presents the first extension of LBC algorithms to multiple predictive "models" and continuous outputs (thus yielding heterogeneous collective regression, HCR). To demonstrate the effectiveness of this approach, we apply it to the voting outcome prediction task evaluated by Etter et al.

Adapting LBC to HCR, in a form suitable for the voting prediction task, involved a number of algorithmic decisions, and two primary challenges. First, the existing data had to be converted into a suitable network, since "links" are not inherently present. While prior work has proposed some methods for similar tasks, based on node similarity or proximity, we demonstrate that link construction based on correlation history yields superior results. Second, since existing LBC methods only support the prediction of categorical outputs, we had to create new methods for relational feature construction to facilitate prediction that instead produces continuous outputs. We show that simple feature strategies can enable link-based regression to improve accuracy. However, we also propose novel alternatives based on "weighted vector" approaches and continuous extensions of existing Bayesian probabilistic reasoning, and demonstrate that they can yield even better accuracy.

Overall, we demonstrate that, for the voting prediction task, HCR can be highly effective, robust to multiple choices of regression parameters and linking strategies, and computationally practical. This success opens the door to the application of HCR to other analysis tasks for link-based data.

Keywords: collective regression, Bayesian inference, link-based data, voting prediction

Contents

1 Introduction

Increasingly, many important domains in the world can be viewed as networks of linked nodes: people connected by social network "friendships," webpages connected by hyperlinks, computing devices linked by communication channels, and even geo-political areas connected by proximity and common interests. To leverage these links for prediction and analysis tasks, researchers in the field of Machine Learning have developed multiple popular techniques for link based classification (LBC). While LBC can substantially improve prediction accuracy in some domains, limitations in current algorithms greatly limit its applicability to more complex domains. In response, this Trident project extends existing methods for LBC to heterogeneous, continuous domains, thus producing heterogeneous collective regression (HCR). As a first application of HCR, we predict regional voting outcomes. In particular, we apply our new methods to a study that predicted Swiss regional voting outcomes (without any use of LBC) and obtain more accurate or comparable predictions in a more efficient manner.

Our contributions are as follows:

- We introduce a new paradigm, Heterogeneous Collective Regression (HCR), for predictions based on relational data. This new paradigm enables for the first time the application of link-based inference algorithms to domains with heterogeneous objects and continuous outputs.

- We create novel methods for link creation based on historical correlations, and demonstrate that they generally yield more accurate results than simpler methods based on geographic proximity.

- We introduce multiple new methods for inferring continuous outputs, for both the initial "bootstrap" problem where predictions must be made without the use of 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 Bayesian inference provides an elegant method for combining link-based information with information about object features, yielding further accuracy gains.

- We perform extensive evaluations of HCR on the voting prediction task and compare with the results of a prior study. By combining our new methods, we achieve voting prediction results that are competitive with or even improve upon those of the prior study, but execute *at least* 25-180 times faster.

Below, Section 2 introduces necessary background information on supervised classification, link-based classification, and the differences between classification and regression. Later sections then introduce our modifications that will create HCR and explain how we will assess it. Section 3 summarizes related work relevant to the novel approach we propose with HCR. Section 4 introduces the prior study and the problem of vote prediction. Section 5 details how we extend LBC to HCR and explains our rationale behind applying HCR to vote prediction. Section 6 provides details about how we implement HCR and the key questions

we must consider in doing so. In Section 7, we discuss the specifics of our experimentation and how it relates to the prior study. In Section 8 we present and analyze our results. Finally, Section 9 proposes potential future work, while Section 10 explains the impact of our research.

2 Background

2.1 Traditional Supervised Classification

Traditional supervised classification problems involve categorizing items in a collection by giving them a certain label. When the classification of an item is not already provided and the label must instead be determined using some prediction mechanism, the characteristics of that item are used to produce a prediction. For example, a webpage could be labeled as "safe" or "unsafe" based on analysis of webpage characteristics such as the length of the page, the number of misspelled words, and the presence of certain certifications. While there are many ways to classify items in a collection, the use of statistical models developed by modern Machine Learning techniques is one of the most efficient ways to accomplish classification.

A statistical model is used to predict a categorical output value (the label) given an input. The model itself is a mathematical function that is developed to output a prediction for an item's label based on certain features. Commonly, these features are the item's characteristics or other properties of the item undergoing classification. A hypothesis function, a common type of model, is developed based on a number of "training" examples, or a set of inputs with their corresponding labels [14]. For instance, logistic regression and support vector machines are two effective and popular types of hypothesis functions. Typically, a hypothesis function has a parameter corresponding to every feature based on that feature's weight in the final classification of that item. For instance, using the example of the safe or unsafe webpage, a hypothesis function, such as the following, would mirror the influence that each feature (length of page, number of misspelled words, and presence of certifications) has on the likelihood of a webpage being unsafe:

$$h(x) = 0.002 * x_1 - 0.5 * x_2^2 + 0.5 * x_3$$

This equation shows that when the hypothesis is given an input from a "test" set, in which labels are not given, it is then able to predict a label. In this example, x_1 represents the length of the webpage in bytes, x_2 represents the number of misspellings, and x_3 is a 0 or 1 value reflecting the presence or absence of certifications. The outputted value, $h(x)$ indicates a prediction of safe for a value of zero or greater and a value of unsafe for a value less than zero. For example, a webpage that is 800 bytes long, has 6 misspellings, and does not have certifications (corresponding to a value of 0 for x_3) would generate an output value of -0.2, resulting in a prediction of unsafe.

Machine Learning techniques are used to optimize the accuracy of the hypothesis function. From an initial hypothesis, techniques such as gradient descent and conjugate gradient are used to find the parameters (e.g., alternative values for the constants 0.002, -0.5, and

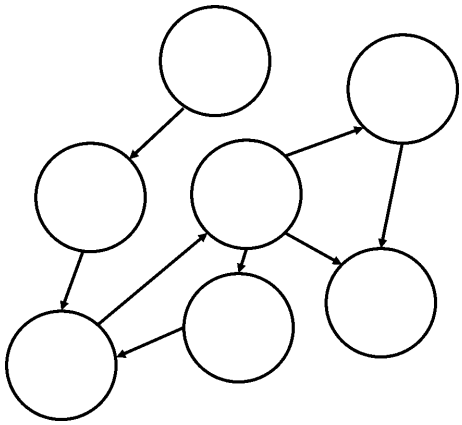

Figure 1: An example of a graph. Nodes are connected by directional links.

0.5) for a final hypothesis function that minimizes error in predicting the outputs of the training set. Typically, accuracy of the model, when used later on the "test" set, improves when more examples are provided in the training set.

2.2 Link-Based Classification

Link-based classification (LBC) is a subdomain of classification that involves the categorization of items connected in a network, a structure known as a graph, as shown in Fig. 1. In a graph, each item in the collection is known as a node, and each node's connection to another node is known as a link, represented accordingly using filled-in circles (for nodes) and directional arrows (for links) in Fig. 1. Each node has attributes (i.e., features of that node) and a label [11].

In LBC, the label is of particular interest, as the statistical model predicts the label. Continuing to use the earlier example, a set of interconnected webpages could be represented as a graph. Each individual page is a node in the graph, a hyperlink to another webpage would represent a link, safe or unsafe would comprise the label, and the length of the page, number of misspelled words, and presence of certifications would make up the set of attributes.

Fig. 2 shows this more specific example, with each page represented by a node with a label, connected to other pages, and with its attributes listed in the left column of the node. In Fig. 2, webpages 1 and 3 are labeled safe, while webpages 2 and 4 are labeled unsafe. Because all of these webpages have "known" labels, this graph would be well suited for learning a predictive model that could later be used to make predictions for other webpages (in the "testing graph") where the safe/unsafe labels are not known.

LBC allows for predictions to be made regarding a node v in a graph based, in part, on the labels or attributes of the *neighbors* of v (the set of nodes linked to v) [10]. Typically, LBC makes use of relational features, or the patterns in the labels or attributes of a node's neighbors. For a specific webpage, its relational features could include whether it links or not to an "unsafe" page, the average length of the webpages it links to, or a weighted average of misspellings on webpages it links to, among other possibilities. These examples of relational features are shown in the right column of each node in Fig. 2. Note that typically the values of the *attributes* for some node v are already provided (e.g., the length) or are computed

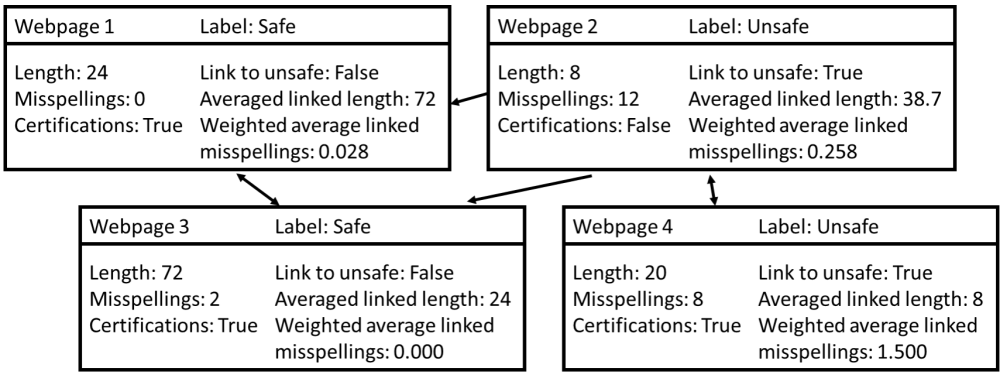

Figure 2: A network of four connected webpages. The left column of each page shows the page's attributes, while the right column shows relational features, the values of which must be computed based on the attributes and/or labels of each node's linked neighbors.

solely on information that is local to a particular node (e.g., the number of misspellings in the page). In contrast, the values of v's *relational features* must be computed based on the attributes and/or labels of v's linked neighbors.

The motivation behind LBC is that predictions made based on both the node's own attributes, and the attributes and/or labels of neighboring nodes are often more accurate than predictions made based solely on the node's own attributes [3]. For this reason, the data that LBC is applied to is often referred to as "relational data," meaning that the classification (label) of a node is related not only to the node's own attributes but to its neighboring nodes' labels and attributes as well. In other words, determining whether a webpage is safe or unsafe may be accomplished more successfully by considering the content of the webpage in addition to the classification (label) and content of the pages it is linked to instead of solely considering the content of the webpage itself.

2.3 Collective Classification and the Iterative Classification Algorithm

Collective classification is a specific type of LBC that uses the labels of neighboring nodes to make predictions about a node's own label. Prior work on LBC shows that collective classification can be highly effective at improving accuracy (compared to methods that do not use information about "neighbors"). However, because the predictions rely on the values of neighbors' labels, many of which are not known (in the testing graph), some kind of iterative "collective" classification is needed for inference [15]. There are many strategies that can be used for collective classification, including Gibbs sampling and Belief propagation. However, one common strategy, and the strategy that will be our focus, is the iterative classification algorithm (ICA) [7]. Previous studies have showed ICA to be an effective method for improving accuracy for all nodes in the "test" graph [9]. ICA can be broken down into three steps: train, bootstrap, and iterate.

In the training step, using a training graph, the algorithm "learns" the classification model that will later be used to predict the labels of the nodes in the test graph. Using the known labels in the training graph, a strategy such as logistic regression or support

vector machines is used to analyze the correlation between the labels and the model inputs. These inputs include both a node's attributes and relational features based on the labels of neighboring nodes [15].

In the bootstrap step, the algorithm uses the model to make an initial prediction for each node in the "test" graph, using only each node's own attributes. The bootstrap step does not account for relational features, meaning that this initial prediction is independent of any qualities of a node's neighbors [15].

In the iterative step, the algorithm updates the predicted results, hoping to achieve greater accuracy among the predicted labels. This step can be broken down into two sub-steps, updating relational features and classifying [15]. First, using predictions from bootstrap or an earlier iteration, relational features between nodes are computed using the predicted labels of each node. For instance, a relational feature could be whether or not a webpage is linked to a webpage labeled unsafe. Then, based on these updated relational features, the algorithm uses the learned model to make a new prediction for each node's label. If this prediction differs from the node's current prediction, the node is reclassified to reflect the algorithm's prediction. The iterative step then repeats (since the new predictions will impact the values of the relational features), ideally leading to more accurate results after every iteration, for some specified count or until convergence. Typically, 10 iterations is sufficient to achieve at least approximate convergence.

2.4 Sparse vs. Fully-Labeled Graphs

Differences in the proportion of nodes that are labeled prior to prediction impact how data is treated and used, particularly in training [9]. The "fully-labeled" case is when every node in a graph has a label, represented in Fig. 3a. In this case (as in Fig. 2), there is no need to predict values for "missing" labels during training, and focus can instead be shifted towards constructing a model to make predictions during testing. Training on fully-labeled graphs tends to be more accurate, as incorrectly labeled nodes do not factor into the model.

The "sparsely-labeled" case is when few nodes in the graph are labeled, depicted in Fig. 3b. In this case, there are two scenarios that must be considered, training (i.e., learning) and testing (i.e., inference) [9]. First, training with a sparsely-labeled graph is challenging because there are few labeled nodes to serve as examples during the learning process, and especially because most labeled nodes have few, if any, neighbors with known labels. This

(a) A fully-labeled graph. (b) A sparsely-labeled graph.

Figure 3: Two graphs with different prevalences of labels.

greatly complicates the process of computing relational features. For this reason, researchers often make use of at least some predicted labels during training, even though this introduces some errors into the learning process. For testing with partially-labeled graphs, we note that inference can be applied to the same graph that was used for learning (if it was partially-labeled) or to a different graph. In either case, known labels in the testing graph provide labels that help "ground" the inference process. Therefore, inference is easier when more labels are provided in the testing graph, although inference can still be applied to graphs that contain no known labels (because methods like ICA start by predicting, via the bootstrap step, a label for all nodes without known labels).

Of course, graphs often exist somewhere between the fully-labeled and sparsely-labeled cases, and a hybrid of strategies should be employed to train on a graph based on the sparsity of labels. Many papers assume training on a fully-labeled graph and testing on a partially-labeled graph. This will be our focus, as this scenario reflects typical voting domains.

2.5 Regression vs. Classification

Thus far, classification has been discussed, or the prediction of categorical values [14]. For example, a webpage must be either safe or unsafe. A digit must be 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9. Categorical items can be grouped together based on their label. Classification problems aim to assign a categorical label to a node. Often, classification problems use a probability threshold to make a prediction about a categorical value. A model is used to determine the probability that the node's label is a particular value, and if that probability is above the threshold the label is predicted to be that value.

Unlike classification, regression problems attempt to predict continuous values. Continuous values are measurable values, such as percentages, weights, counts, sales, and votes. Continuous labels differ from categorical labels because there are an infinite number of possible labels, meaning nodes with continuous values are not categorized by their labels. The methodology used to assign a continuous label to a node in a regression problem differs from assigning a categorical label to a node in a classification problem as a model outputs the continuous value instead of a probability to be compared to a threshold [14]. However, some similar methods can be used in regression problems. For instance, small modifications to logistic regression (a classification method, despite the name) produce linear regression, an effective regression method. Likewise, decision trees (a common classification method) can easily be adapted to become regression trees.

3 Related Work

3.1 Collective Regression

In general, there is very little prior work on collective regression. One exception is Logliscli et al. (2015), which offers examples of different ways to use attribute similarities in constructing the links for the graph data [6]. Additionally, it demonstrates the feasibility of collective regression for certain domains. However, there are several limitations present in this article. First, Logliscli et al. uses only very small datasets. Also, it considers only the 10% labeled

case. It is possible that for graphs that do not match this specific case, results will vary. Additionally, it uses only one regression model and uses only the possibility where each node has a single continuous output. Examining collective regression with exposure to a larger set of inputs, without a specified labeling scenario, using a model unique to each node, and considering multiple labels at different times would provide an opportunity to greatly expand upon the work done in this article.

However, Loglisci et al. does introduce an attribute schema that is somewhat related to our approach. Its "Attribute Schema Var3" involves constructing a histogram based on values for a particular relational feature. This shares similar intentions with the Discrete Bayesian inference paradigm that uses histograms that we create.

3.2 Heterogeneous Approaches to LBC

Neville and Jensen (2007) studies link-based classification for a "movies" domain that includes movies, producers, and studios. They use relational dependency networks and pseudolikelihood learning techniques in a heterogeneous environment in order to estimate an efficient approximation for a probability model to be used for the entire distribution [12]. It considers different node types (e.g., for movie, producer, and studio) and uses a different model for each type of node. However, this approach is fundamentally different from how we intend to approach heterogeneous environments as we intend to effectively consider each node as its own type, using a unique model customized to better fit each node. Such an approach is possible for us because of the historical data we have on voting outcomes.

3.3 Recommender Systems

Voting prediction could potentially be accomplished through methods based on "recommender systems." For instance, Koren et al. (2009) explores the use of collaborative filtering to predict recommendations that will be well received by the user [5]. These recommendations are based on user history, and look for pattern across users. 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 [5]. It uses an example of generating movie recommendations based on past user ratings using methods such as Stochastic gradient descent and alternating least squares. While voting outcome prediction could also make use of latent factor methods (as demonstrated by Etter et al.), our case differs from Koren et al. for several reasons. First, the movie recommendation problem is necessarily very sparsely-labeled, as a user is unlikely to have watched most of the thousands of films in the set of all films. Our predictions are based upon a more densely labeled graph, as we have the available voting history for the majority of elections in a region, which leads to a different set of learning challenges. Second, we hope to incorporate an online strategy. As more information becomes available, or as more labels are revealed, we intend to recalculate our predictions to achieve a more accurate prediction based on a greater amount of available information, without needing to re-learn the model. In contrast, recommender systems typically update their model only periodically. Finally, voting prediction has an intrinsic geographic aspect that is not typically present in recommender systems. This provides a natural source of links to use

for connecting regions, and enables us to leverage LBC methods that will directly leverage such links to improve accuracy.

4 Target Problem

Because relational data is so commonplace, there are numerous applications for which the LBC paradigm could be useful for making more accurate predictions. One of these applications is the prediction of voting outcomes. For this task, accurate, region-specific, and "online" results are desirable. First, accurate results (meaning that our predictions are not far from the actual outcome) are essential for providing reliability. Second, we would like region-specific results, meaning that we are able to see clearly how the preferences of different areas vary based on their individual features. 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. While these types of results are ideal, in practice voting prediction is often a difficult task, evidenced by the frequent disconnect between reported media election forecasts and true election outcomes.

In this study, we take an already existing collection of aggregate Swiss voting outcomes and use the paradigm of LBC (appropriately extended) to develop a more accurate voting prediction mechanism. While this collection was used in a previous experiment on voting outcome prediction, using a different methodology leads to competitive (and sometimes better), more efficient results. In particular, Etter et al. (2016) previously examined this collection of voting outcomes for 281 national referendums over a period of 34 years [2]. Results are reported for each of the 2352 regions (municipalities) in Switzerland, with the regional breakdown as illustrated in Fig. 4. Each region is described by 25 "per-region" features that describe demographics of each region, while each vote has 13 "per-vote" features that are derived from recommendations made by various political parties. Section 5.2 and Section 7.2 describe these features in more detail.

Of the 281 voting outcomes, Etter et al. used the first 231 as a training set and left

Figure 4: Summary of results for two different Swiss referendum votes, colored on a per-region basis. Darker shading indicates that a region voted more strongly in favor of the issue. The results show that (a) results can vary widely from one vote to another, (b) different regions' votes may vary differently on the same issue, and (c) substantial geographic correlations may exist among the voting patterns, though they may vary depending on the issue. Figure taken from Etter et al. (2016) [2].

the last 50 voting outcomes as the test set, where voting predictions are made and then accuracy evaluated. 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. We use the same collection of aggregate voting outcomes, but we employ link-based regression methods (specifically, our new HCR approach) with the aim to produce more accurate and efficient results.

5 General Approach

5.1 Heterogeneous Collective Regression

While there is an effective application of LBC to Etter et al.'s collection of voting outcomes and the problem of vote prediction, it is important to recognize the limitations of LBC and why previous LBC methods were not applicable to vote prediction. We extended LBC to heterogeneous collective regression (HCR) in order to address these limitations. HCR, or regression in continuous, heterogeneous domains differs from LBC in its ability to make numerical rather than categorical predictions and its modeling of the data in heterogeneous rather than homogeneous ways. In order to implement HCR, we focus on two goals:

1. Extend from homogeneous environments to heterogeneous environments
2. Extend from categorical domains to continuous domains

5.1.1 Goal #1: Extending from homogeneous environments to heterogeneous environments

In a homogeneous environment, we can assume that all nodes come from the same underlying "population." For instance, classifying all the webpages within a single university as "safe" or "unsafe" may be considered a homogeneous prediction task. In such cases, a single predictive model can effectively make predictions for all nodes.

In a heterogeneous environment, the nodes are assumed to exhibit significantly more diversity, such that the use of a single predictive model for all nodes would be expected to produce poor accuracy. For instance, in our Swiss voting collection, the nodes under study are regions with significant diversity across the country. To handle such diversity, we extend the "single model" approach to the use of multiple models, with each model tailored to an individual node. In general, for graphs in which we have significant amounts of data for each node (e.g., the historical voting records for each region) individual models tailored to each specific node should produce greater accuracy than the use of a universal model, if the nodes exhibit significant diversity. This transition, from a single model to individualized models, is represented in Fig. 5.

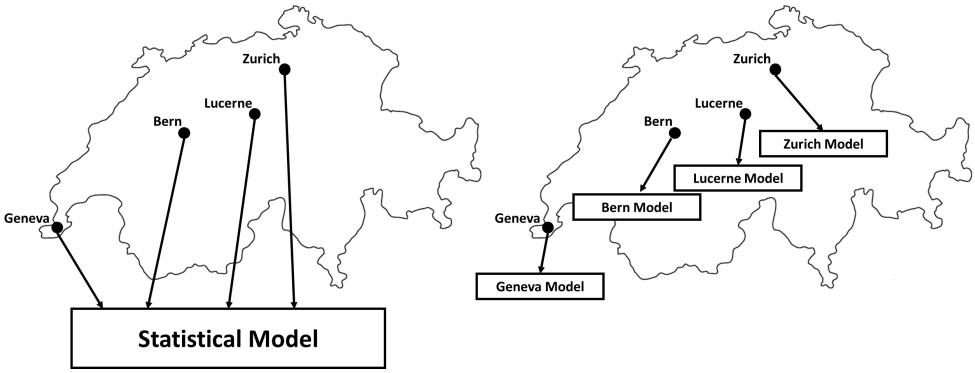

Figure 5: A homogeneous approach (pictured on the left) would involve the creation and use of *one* statistical model to make predictions for all voting regions. A heterogeneous approach (pictured on the right) would involve the creation and use of individual statistical models tailored to each region to make voting predictions.

5.1.2 Goal #2: Extending from categorical domains to continuous domains

The use of collective regression provides further insight into the interpretation of a network after a prediction. For example, with categorical values, an output that has a 51% likelihood of being true and an output with a 99% likelihood of being true are both represented by the same categorical value: "true." Clearly, however, there is a significant difference between a slight and an overwhelming majority, and the presence of one or the other may indicate something about the network. Collective regression, rather than classification, allows these possibilities to be more easily seen.

5.2 Voting Prediction Using HCR

There are many applications of HCR, including sales and temperature prediction, but the prediction of regional voting outcomes, as mentioned above, represents a problem for which HCR could be particularly useful. Different voting regions could represent the nodes of a graph, and the paradigm of LBC, using both self attributes and the features of linked regions seemingly could be used to make predictions. However, the use of HCR, rather than LBC, addresses the issues that make the application of traditional LBC to voting prediction problematic. In particular, because there are often sufficient voting results (from history) to examine each region individually during training, it makes sense to develop a hypothesis tailored to each region, meaning that the use of multiple models could increase accuracy of prediction. Additionally, different voting regions can be linked using different circumstances, including geographic proximity and attribute similarity, making the graph data structure an effective way of represent voting regions. Finally, being able to note the difference between a region where a landslide victory is expected and one where the vote is highly contested could be a key motivator to use regression rather than classification in order to obtain continuous predictions.

To use Etter et al.'s collection to test the viability of HCR, we must first represent the collection of voting outcomes as a graph. In this graph, each node will correspond to a voting region. The attributes of each node are comprised of different regional features. We

will use the same 25 region features used by Etter et al., which include geographic position, population, population density, population age distribution, economic information, political party participation, languages spoken, and other demographic information.

In addition to regional features, Etter et al.'s collection additionally contains information regarding individual elections, or "vote features." In particular, there are thirteen political parties in Switzerland that influence voting on the national scale, and before each vote occurs these parties make a voting recommendation on the topic at issue. Etter et al. demonstrated that these recommendations are significantly correlated with the results of each vote, but are not sufficient in themselves to yield high predictive accuracy.

In order to generate a predictive model for each node (to predict the outcome of a specific vote), we use a combination of the "per vote" features of each vote and the region features of each region. Per vote features are based on the nation-wide recommendations of the thirteen Swiss political parties. Because we have access to the history of these recommendations, in addition to the voting history of each region, we learn region-specific regression models that are able to predict the voting behavior of each region for some new vote (with unknown outcomes), given the political parties recommendations for that vote. Moreover, drawing back upon the concepts of collective classification, including relational features (e.g., based on the voting outcomes for the neighbors of a node) enables us to perform collective regression with these models, which, given the other correlations observed but not fully leveraged by Etter et al., helps to improve predictive accuracy.

The regional features represent potential attributes for each node, and it is important to note that these features are independent of the voting event, remaining constant regardless of what vote is being considered. In our scenario, we use these "per region" features as one possible criterion for generating links between nodes (details discussed further below), in addition to incorporating them into different forms of "per region" regression.

In order to reproduce Etter et al.'s experiment, but adapted for our new HCR methods, we will have to adapt the collection of aggregate voting outcomes into a graph data structure, then apply collective regression. To explicate the basic problem, Table 1 compares selected results from Etter et al.'s experiments to one of our initial runs that do *not* use collective regression. Each row corresponds to the results of a different overall inference strategy, with the first row reflecting one of our experiments and the final three rows reflecting results from Etter et al.'s experiments. On the left is the name of the strategy, which provides a basic overview of the parameters of that specific inference, and on the right are the results of the experiments. Each column corresponds to a different number of "known regions," or regions for a test vote for which we know the final results and can use to inform our inference for

Table 1: RMSE of selected results, to establish baselines for performance. Lower values are better.

	Number of Known Regions, N_k							
Strategy	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
HCR								
Vote Features Only	11.90	9.11	8.71	8.26	8.20	8.15	8.15	8.15
Etter								
BIAS	12.89	10.34	9.98	9.60	9.54	9.51	9.50	9.50
MF+GP(R)	12.89	8.90	7.76	6.03	5.68	5.18	5.01	4.87
MF+GP(R)+LIN(V)	11.84	8.35	7.54	6.20	5.86	5.37	5.23	5.15

the rest of the "unknown regions."

The numerical values in Table 1 are the results, presented in the form of root mean square error (RMSE), a measure of error. RMSE is defined by the following formula:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(x_{predicted_i} - x_{true_i})^2}{n}}$$

where $x_{predicted}$ is the set of predicted values, x_{true} is the set of true or observed values, and n is the sample size. RMSE is the square root of averaged square errors. In our case, the error is the difference between the predicted value and the true value. Thus, RMSE aggregates all of our prediction errors and gives an estimation of how far our predictions are away from the true values, with larger error values having a greater influence on the RMSE value. For example, if every region had an error of 0.03 (out of 1), the RMSE would be 3.00%. If half of the regions had an error of 0.03 and the other half had an error of 0.06, the RMSE would be 4.74%. We look to minimize this error, RMSE, throughout our experiments. Thus logically as the number of known regions increases and we make more informed predictions, the RMSE decreases.

Etter et al.'s BIAS model is a rather poor model, as it simply uses the national average of a vote for each region's prediction. This means that any remotely effective model should outperform this BIAS model. Table 1 shows that simply applying an individualized linear regression model (the "Vote Features Only" row) for each region, based on past votes and their associated vote features, to predict future votes provides us a baseline that compares favorably to Etter et al.'s BIAS results. However, we seek to improve by refining our initial predictions, computing useful relational features over effective links, and applying collective inference with those links. Etter et al.'s best results are included in Table 1 as well, and these are the results we aim for with our refined HRC.

5.3 Intuition

We anticipate that HCR will provide more accurate results than the traditional regression used by Etter et al. to predict Swiss voting outcomes. In short, (1) prior work with LBC has demonstrated that it can substantially improve accuracy when the nodes (regions in this case) are correlated, and (2) the Swiss voting outcomes are highly correlated for regions that are more similar (geographically or otherwise). Thus, we expect extending LBC to HCR will lead to improved accuracy, compared to methods that do not use links, in this voting domain. We elaborate on these two points below.

First, for a graph that connects nodes based on corresponding features, there is evidence from previous studies that LBC methods perform prediction better than traditional classification strategies. Neville et al. (2003) used multiple prediction strategies, several based on treating the data as relational and one based on considering only self-attributes [13]. Across three different prediction problems, LBC strategies for relational data significantly outperformed traditional self-attribute (non-relational) strategies. Sen et al. (2008) further demonstrated that LBC strategies such as ICA and Gibbs sampling significantly outperformed traditional classification methods [15]. It also concluded that ICA was the best LBC option because of its relative simplicity and tendency toward relatively fast convergence. Our approach, HCR, is an extension of LBC and we specifically extend ICA's approach. We

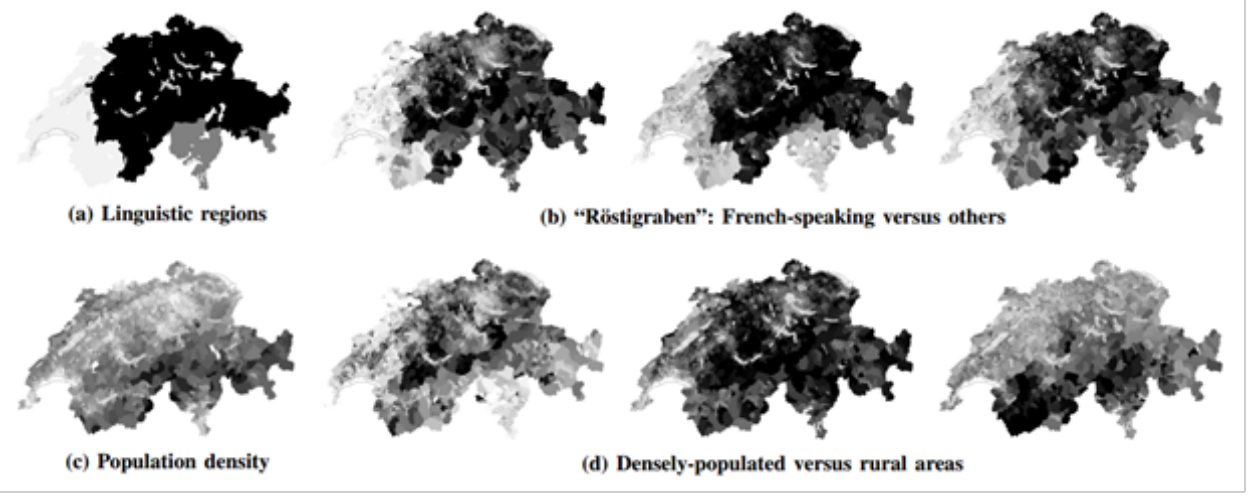

Figure 6: (a) shows the regions in Switzerland that speak French (light gray), Italian (dark gray), and German (black). (b) shows results for three different votes across the entire country and demonstrates that in certain votes French speaking regions tend to vote differently than the rest of the country. (c) shows the population density of Switzerland, with lighter colors representing more heavily populated areas. (d) shows results for three other votes, demonstrating how densely populated regions sometimes vote differently than rural areas. Figure taken from Etter et al. (2016) [2].

expect that by recognizing the correlation between voting regions by treating Etter et al.'s collection of outcomes as a graph, we stand to gain prediction accuracy by using principles of LBC.

Second, Etter et al.'s own results show that the voting results between regions are highly correlated. We can see this in Fig. 6, which shows two examples supporting the claim of correlation between regions. Etter et al. used these examples to show that demographic features such as language spoken and population density are related to how a region votes differently from the rest of the country. Etter et al.'s models make use of such features, but do not consider the use of link-based methods for prediction. An advantage of our proposed method is that it more directly enables us to leverage partial results from the current election. In particular, Etter et al.'s models can use a region's demographics to help predict a vote outcome, but each region's prediction is ultimately based only on its own demographics. In contrast, our collective regression will be able to leverage not only a region's own demographics, but the relevant voting results (both predicted and "known") from other regions. Thus, if we are able to capture relevant relationships between regions with our linking strategy, we will be better able to predict how a region votes based on the voting patterns of its linked regions.

6 Methods for Heterogeneous Collective Regression

In order to develop the most useful graph and to perform effective collective regression, we must consider three key questions:

1. How to "bootstrap" 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?

6.1 Key Question #1: How to "bootstrap" initial predictions to provide a baseline for our inference?

While our different types of collective inference use varying methods to make the most accurate possible predictions, each inference method depends on a set of initial predictions as input. In general, providing a better starting point for predictions will lead to better overall results. To get to this starting point, we must provide "bootstrap" results, or initial predictions, which are computed *without* the use of links.

First, we consider the following existing methods, used by Etter et al., for our bootstrap in order to construct an initial prediction, $y_d^{(0)}$, for each region d:

1. **LIN(V)** - regression based only on the vote features. We train each region's model by comparing the 13 political party recommendations for each training vote to that region's voting outcome. For test vote n, we input the vote's political party recommendations, w_n, outputting predictions

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

   where γ_d is the set of learned, region-specific regression constants.

2. **LIN(R)** - regression based only on the region features. For a particular test vote n, we consider the 25 region features for each of the "known" regions as our training input. We compare each "known" region's features to its voting outcome, learning a model for vote n. Then, for each "unknown" region d, we input the 25 region features, x_d, outputting predictions

$$y_d^{(0)} = \beta_n^T x_d$$

   where β_n is the set of learned, vote-specific regression constants.

3. **JOINT(R,V)** - regression based on the region features and the vote features. This is the model that Etter et al. called LIN(R)+LIN(V). We consider the same models described above, LIN(R) and LIN(V), but we learn them jointly (using Alternating Least Squares) instead of independently. This yields the following predictions:

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

Additionally, we created the following three new methods for bootstrap:

1. **BayesD+Indpt(r,v)** - This method performs Bayesian inference using conditional distributions for the vote and region features, learned independently that are represented with a discrete approximation (hence the "D" in BayesD).

   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.

   $$P(y_d = y|x_d, w_n) = \frac{P(x_d, w_n|y_d = y) \cdot P(y_d = y)}{P(x_d, w_n)}$$

   Because the denominator is independent of y, we eliminate the value from our consideration and consider just the numerator.

   $$P(y_d = y|x_d, w_n) \propto P(x_d, w_n|y_d = y) \cdot P(y_d = y)$$

   Assuming conditional independence, we consider the probabilities of x_d given $y_d = y$ and w_n given $y_d = y$ separately:

   $$P(y_d = y|x_d, w_n) \propto P(x_d|y_d = y) \cdot P(w_n|y_d = y) \cdot P(y_d = y)$$

   We apply Bayes rule again:

   $$P(y_d = y|x_d, w_n) \propto \frac{P(y_d = y|x_d)P(x_d)}{P(y_d = y)} \cdot \frac{P(y_d = y|w_n)P(w_n)}{P(y_d = y)} \cdot P(y_d = y)$$

   We cancel terms common to the numerator and denominator and eliminate terms independent of y ($P(x_d)$ and $P(w_n)$), and we are left with the following product:

   $$P(y_d = y|x_d, w_n) \propto \frac{P(y_d = y|x_d) \cdot P(y_d = y|w_n)}{P(y_d = y)} \tag{1}$$

   To estimate $P(y_d = y|x_d)$, we apply the LIN(r) model to the training data. Then, using the same reasoning described late in Section 6.3, we generate histograms of the error between the true training values and predictions made with LIN(r). $P(y_d = y|w_n)$ uses the same process but with LIN(v). For inference, these histograms are then mean-shifted, for each region d, based on predictions from LIN(v) and LIN(r). Finally, we find the most likely value of y_d for $P(y_d = y|x_d, w_n)$ via a discrete estimate of the following integral:

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

2. **BayesG+Indpt(r,v)** - this method also uses Bayesian inference to combine the influence of the independently-learned models for the vote features and region features, as with Equation 1. However, by approximating the conditional probabilities as Gaussians, we enable exact inference, instead of resorting to a discrete approximation.

   In particular, we substitute normal distributions for each respective conditional probability, leaving us with the following product:

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

.

In this equation, each mean is computed based on the results from LIN(R) or LIN(V) (see above), or is known to be zero. 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, as further described in Section 6.3.

3. **JOINT(R,V)/BAYESG+INDPT(R,V) ENSEMBLE** - a combination of the previous two methods. We simply average the predictions generated by the JOINT(R,V) and the BAYESG+INDPT(R,V) models described above. By doing this, we hope to mitigate outliers predicted by either model. We use the following result as our prediction, $y_d^{(0)}$:

$$y_d^{(0)} = \frac{[\text{JOINT(R,V)}] + [\text{BAYESG+INDPT(R,V)}]}{2}$$

The methods described above are summarized in Table 2.

While it is intuitive that we would obtain the best bootstrap results by making use of both the region and the vote features, we first consider using just the vote features or just the region features by themselves for comparison.

Table 2: 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
ENSEMBLE	Yes	Yes	ALS and Bayesian Inference

6.2 Key Question #2: How should links be computed, so as to connect regions into a useful graph?

From the beginning of our experimentation, we hypothesized that linking related regions could be beneficial. By implementing heterogeneous models, we recognize that too much diversity existed among regions and that using a single predictive model across all regions would have reduced effectiveness. However, we suspect that a subset of regions likely exists for each region that tends to vote in related ways. Therefore, considering those related regions should increase the power of each region's model. Thus, we consider several methods of linking:

1. **Proximity-based** - We calculate geographic proximity between regions and link any regions within N kilometers.

   In one of our earliest feasibility studies, we considered all of a selected region's neighbors within a distance, X. We then compared the average percentage on a vote amongst

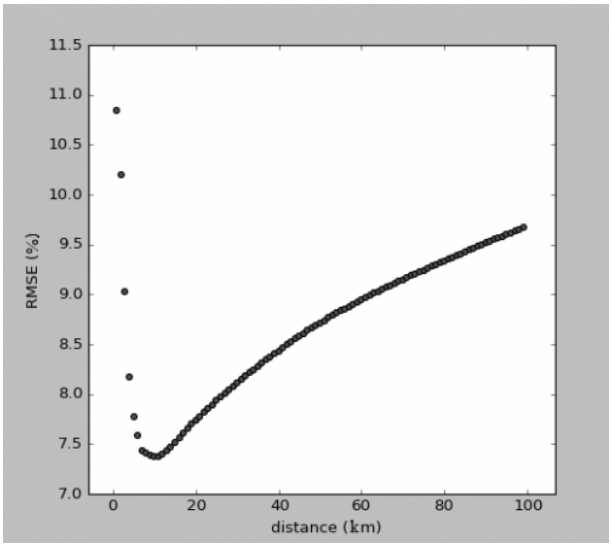

Figure 7: The results of a feasibility experiment in which we predicted votes for a region by averaging all the predictions of its neighbors within a certain distance (plotted on the x-axis). We compared this prediction to the true result and plotted the RMSE on the y-axis. We observed approximately 11 km to be the optimal distance for this feasibility test.

those neighbors compared to the true voting percentage of the selected region. We ran trials varying the distance used to select neighbors between 1km and 100km. We were encouraged that the error for our results, shown in Fig. 7, compared favorably to Etter et al.'s BIAS results, leading us to believe that useful linking information could be discerned from geographic proximity. In addition, Figure 7 shows that RMSE was minimized when using a distance of $X = 11$ kilometers.

Additionally, we can weight proximity links so that neighbors closer to our region of interest have a greater impact on training the model and predicting results. We can consider a variety of weighting functions in order to vary the magnitude of the effect that a greater distance has; Section 8.2 shows specific results.

2. **Correlation-based** - While geographically close regions are likely to be similar, potentially the most useful links could be obtained by identifying regions with the most similar voting histories, regardless of proximity. To do so, we calculate the correlation between regions' voting histories. In particular, for each possible pairing of regions, we compare the results of all 231 training votes and compute the Pearson correlation coefficient between the two regions. We seek to link each region to the nodes with the highest correlations using these values, but there are several strategies we consider to determine how many neighbors each region should have:

   - Threshold - all regions that have a correlation to our region of interest above a selected threshold, c, are included as neighbors. There is no minimum or maximum number of neighbors; some regions may have no neighbors while others may have many neighbors.
   - Set number - for each region of interest, we find the N regions with the high-

est correlation and select them as neighbors. Thus, each region has exactly N neighbors.

- Threshold with a minimum - all regions that have a correlation to our region of interest above a selected threshold, c, are included as neighbors. However, while there is no maximum number of neighbors, each region must have a minimum of N neighbors. If a region does not have N above the threshold c, the regions with the correlations closest to c are selected.

Additionally, similar to proximity-based links, we have the option to weight links. We weight correlation-based links simply based on their calculated Pearson correlations, but we could have considered other weighting functions.

The intended outcome of correlation-based links is to exploit the extensive voting histories available for each region. We link regions together that tend to vote in the same manner, potentially more directly aligning with the intended goal of vote prediction.

We choose links that represent relationships between regions that are (hopefully) informative in regards to voting prediction. These links can then be leveraged for prediction via some method of collective inference, as described below.

6.3 Key Question #3: Given the initial predictions and informative links, how can we perform effective collective regression?

Regression techniques are well established, but collective regression requires some kind of iterative procedure to allow predicted and known values to propagate throughout the graph during inference. We have adopted a number of inference strategies. However, each is dependent on a set of initial predictions, $y_d^{(0)}$, generated by the bootstrap step:

$$y_d^{(0)} = f_{boot}(x_d, w_n)$$

Given these initial predictions, some collective inference method is then used to iteratively refine the predictions. We first consider the following two existing techniques:

1. **RegressICA** - This method uses ICA, described in Section 2.3, but with an underlying model of regression rather than logistic regression, which is a classification method. When combined with suitable relational features, this approach allows the results of neighboring regions to influence one another. For each iteration i, we generate a prediction for each region y_d:

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

where f is a learned regression model and L_d is the set of regions that are linked to region d.

We start with initial predictions provided by our selected bootstrap model, and we begin our first iteration. For each region's tailored model, we build a set of regression

features including the 13 vote features for the vote of interest and a relational feature, a "neighbor average" for each region. There are a number of relational features we could compute between linked regions, but we compute the neighbor average by taking the mean of a region's neighbors' prediction for the vote of interest, believing this value to be informative of the relationship between linked regions. The output of each region's regression model becomes the new prediction for that region at the end of the iteration. We repeat iterations until we reach a satisfactory convergence.

2. **Weighted Vector** - Rather than using the neighbor average as another feature for regression along with the vote features, this method uses the neighbor average as its actual *prediction*. We first bootstrap to generate initial predictions, and then we repeatedly average the predictions of each region's neighbors. When links are weighted, each neighbor's contribution to the neighbor average is proportional to its link weight. We repeat iterations until we reach a satisfactory convergence:

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

This method is analogous to the "weighted vote relational neighbor" (WVRN) method of Macskassy and Provost, but adapted for regression instead of classification [8].

Additionally, we developed the following new methods for collective inference:

1. **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, first applying Bayes rule:

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

Next, assuming conditional independence of the neighbors' values given y_d, we can simplify as follows:

$$P(y_d = y | y_{L_d}) \propto P(y_d = y) \cdot \prod_{j \in L_d} P(y_j | y_d = y) \tag{3}$$

In general, estimating $P(y_j | y_d = y)$ from the training data could be a challenging task. However, we conjecture that for our prediction purposes this probability can be approximated as a function solely of the difference between y_j and y_d.[1] In order to incorporate information about each linked region into our inference, we approximate

[1]For example, we estimate that the likelihood of y_j being 0.3 given y_d is 0.1 is approximately the same as the likelihood of y_j being 0.6 given y_d is 0.4. This would be true if similar (linked) regions tend to vote in the same ways.

this function as $\phi(x)$:

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

The Bayes inference approaches, beginning with BayesD+Links, are a break in our paradigm of directly using regression to generate predictions. For this type of inference, we use Bayesian inference based on continuous properties of histograms to incorporate beliefs about each neighbor. We generate histograms with training data to estimate conditional probabilities. We use regression only to adjust these conditional probabilities.

To generate histograms, we first consider the mean-adjusted voting outcomes for every training vote across every region, and we use these values to populate a "prior" histogram, which represents $P(y_d = y)$. Then, for every training vote, for each region we consider the set of links we have generated. We calculate the difference between that region's mean-adjusted voting outcome and those of each of its neighbors, and we use those values to populate a "link" histogram, which represents $\phi(x)$.

Given these discrete estimates of $P(y_d = y)$ and $\phi(x)$ and the previous iteration's predictions for all neighbors of region d (e.g., $y_{L_d}^{(i-1)}$), we then compute the most likely value of $P(y_d = y|y_{L_d})$ via:

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

using 500 discrete intervals for y. This is analogous to our previous use of Equation 2, but now using d's linked neighbors for predictions instead of using d's vote and region features.

2. **BayesG+Links** - This method of inference is similar to BayesD+Links, but because we observed that our computed histograms resembled normal distributions, we instead approximate the link-based conditional probabilities as Gaussians, enabling exact inference.

   Using the same justification as BayesD+Links, we derive Equation 4 to determine the probability of y_d. However, we substitute normal distributions, each described by their mean and variance, for each respective conditional probability:

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

   This form can exploit the property of Gaussians that multiplying or dividing a Gaussian results in another Gaussian. This well-known property is also used by Kalman filters, which estimate a joint probability distribution over multiple variables to predict values [4]. Justification for the product of any number of Gaussians being a Gaussian is provided by Bromiley [1].

   Bromiley provides the following equations that we can use to calculate the mean, μ, and the standard deviation, σ, of the product of two Gaussians, f and g with means

μ_f and μ_g and variances σ_f^2 and σ_g^2:

$$\sigma_{fg} = \sqrt{\frac{\sigma_f^2 \sigma_g^2}{\sigma_f^2 + \sigma_g^2}} \quad \text{and} \quad \mu_{fg} = \frac{\mu_f \sigma_g^2 + \mu_g \sigma_f^2}{\sigma_f^2 + \sigma_g^2} \tag{5}$$

Derived from these equations, Bromiley also provides the following equations that we can use to calculate the mean and the standard deviation of the product of n Gaussians:

$$\frac{1}{\sigma_{i=1\ldots n}^2} = \sum_{i=1}^{n} \frac{1}{\sigma_i^2} \quad \text{and} \quad \mu_{i=1\ldots n} = \big[\sum_{i=1}^{n} \frac{\mu_i^2}{\sigma_i^2}\big] \sigma_{i=1\ldots n}^2$$

This method has the advantage of yielding much faster inference than BayesD+Links and avoiding errors that result from discretization. However, are Gaussians actually good representations for the conditional probabilities of Equation 4? Section 8 compares their accuracy.

To this point, the methods of collective inference discussed have used only the "link" and "prior" information for their inference (with the exception of RegressICA, which uses vote features). The methods that follow aim to improve prediction accuracy by exploiting the vote features and the region features in addition the link and prior information.

3. **Weighted Vector+Delta** - this method uses the Weighted Vector approach, but modified to also leverage vote-based features. In particular, we learn a region-specific "delta" correction, based on the vote features, by comparing computed the neighbor average in our training data to a vote's true value. We add this delta at the end of each iteration:

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

where f_d is a learned regression model using the vote features to predict the "delta" correction.

4. **BayesD+Indpt(r,v)+Links** - this method is similar to BayesD+Links, but it takes region and vote features into consideration in addition to the links.

We again use Bayes rule to determine our prediction from conditional probabilities. However, we must consider the probabilities resultant from the features (x_d, w_n) in addition to those resultant from the links (y_{L_d}):

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

If we again assume conditional independence, then:

$$P(y_d = y | x_d, w_n, y_{L_d}) \propto P(x_d | y_d = y) \cdot P(w_n | y_d = y) \cdot P(y_{L_d} | y_d = y) \cdot P(y_d = y)$$

We recall from Equation 4 that:

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

Now, we simplify the expressions for the region and vote features. For the region features, by Bayes rule:

$$\begin{aligned} P(x_d|y_d = y) &= \frac{P(y_d = y|x_d)P(x_d)}{P(y_d = y)} \\ &\propto \frac{P(y_d = y|x_d)}{P(y_d = y)} \end{aligned}$$

For the vote features, by Bayes rule:

$$\begin{aligned} P(w_n|y_d = y) &= \frac{P(y_d = y|w_n)P(w_n)}{P(y_d = y)} \\ &\propto \frac{P(y_d = y|w_n)}{P(y_d = y)} \end{aligned}$$

We then substitute these expressions for the links, region features, and vote features into our formula.

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

Canceling terms, we are left with the following:

$$P(y_d = y|x_d, w_n, y_{L_d}) \propto \frac{P(y_d = y|x_d) \cdot P(y_d = y|w_n) \cdot \prod_{j \in L_d} \phi(y_j - y_d)}{P(y_d = y)} \tag{6}$$

We determine the conditional probability associated with a particular value of y using the "prior" histogram and the conditional probability associated with each neighboring region using the "link" histogram, in the same way discussed for BayesD+Links.

In order to use the region features for inference, by determining the conditional probability of $P(y_d = y|x_d)$, we generate a "region" histogram. We consider the set of region features for our 2352 regions and our 231 training vote outcomes. For each training vote, we train a model using the region features for the "known" regions to predict their voting outcomes. We use this model to predict the outcomes for the "unknown" regions and subtract these predictions from the true values. We use these differences to populate the "region" histogram.

In order to use the vote features for inference, by determining the conditional probability of $P(y_d = y|w_n)$, we generate a "vote" histogram. We consider the set of vote features for our 231 training votes and their outcomes. We learn models for each region for the first 201 of the training votes, and we use these models to make predictions

on the last 30 training votes. We calculate the difference between the true voting outcomes for these last 30 training votes and our predictions, and we use these differences to populate the "vote" histogram.

For both the "region" and "vote" histograms, we use the associated region and vote regression models to make a prediction for each region on each test vote. For each region, we then center the respective "region" or "vote" histogram on its predicted value. Thus, when determining the conditional probability of a value occurring for y_d, we factor in the likelihood of the value relative to the "region" or "vote" histogram into our calculations.

5. **BayesG+Indpt(r,v)+Links** - similar to BayesD+Indpt(r,v)+Links, this method follows the derivation above but again we substitute a normal distribution for each conditional probability.

   We begin with Equation 6, derived for BayesD+Indpt(r,v)+Links, but we make the normal distribution substitution:

$$P(y_d = y|x_d, w_n, y_{L_d}) \propto \frac{\mathcal{N}(\beta_n^T x_d, \widehat{\sigma}_r^2) \cdot \mathcal{N}(\gamma_d^T w_n, \widehat{\sigma}_v^2) \cdot \prod_{j \in L_d} \mathcal{N}(y_j, \widehat{\sigma}_{links}^2)}{\mathcal{N}(0, \widehat{\sigma}_{prior}^2)} \tag{7}$$

   Again, the means associated with the normal distributions for the region features and vote features are calculated based on the associated regression model. The means for the links are based on bootstrap predictions or the current region predictions (represented by y_j). The prior normal distribution is centered at 0 because our test data is mean-centered. The variance is calculated by approximating each respective histogram as a Gaussian.

   While Bromiley showed that multiplying any number of Gaussians together results in a Gaussian, we must now divide a Gaussian, the numerator above, by another Gaussian, the denominator. We derive in Appendix A.1 that in many cases, dividing a Gaussian, f (with standard deviation σ_f and mean μ_f), by another Gaussian, g (with standard deviation σ_g and mean μ_g), results in a Gaussian. Our derivation yields the following equations to determine the standard deviation and the mean of the quotient:

$$\sigma_{f/g} = \sqrt{\frac{\sigma_f^2 \sigma_g^2}{\sigma_g^2 - \sigma_f^2}} \quad \text{and} \quad \mu_{f/g} = \frac{\mu_f \sigma_g^2 - \mu_g \sigma_f^2}{\sigma_g^2 - \sigma_f^2}$$

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, one encompassing the region features and one encompassing the vote features, we use one Gaussian that is computed based on the Joint(r,v) implementation of the region and vote features.

   The formula for this method is similar to the BayesG+Indpt(r,v)+Links formula. However, we multiply only one conditional probability for the features, rather than two.

Additionally, the respective conditional probabilities for the region and vote features each had a term for the prior conditional probability in the denominator. Because we multiply one less conditional probability, we do not have a prior conditional probability in the final result. The prior term from the feature probability cancels with the prior term that is normally a part of the numerator. Thus, after making normal distribution substitutions, we are left with the following:

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

7. **BayesG+Ensemble(r,v)+Links** - this method is similar and is nearly identical in formula to BayesG+Joint(r,v)+Links:

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

However, the single Gaussian we use for feature information is a combination of the single Gaussian from the Joint(r,v) Strategy and the two Gaussians from the BayesG+Indpt(r,v) strategy. In order to combine these components, we simply average the means (predictions) and the standard deviation associated with each component, using the result to generate a shared Gaussian.

The intuition behind this method is to mitigate the impact of any inaccurate predictions that are made by either the Joint(r,v) strategy or the BayesG+Indpt(r,v) strategy on its own. In situations where one strategy performs poorly, we hope to improve results by averaging with the other strategy. We hope to achieve consistency that leads to overall increased accuracy.

7 Methodology

7.1 Experimentation

In order to carry out this experimentation, we used the United States Naval Academy's new Cray supercomputer, Grace. We adapted an existing Python implementation of LBC for HCR. Overall, this consisted of writing several thousand lines of Python code. In addition, we obtained Etter et al.'s code to enable us to also run trials with his methods, facilitating a direct comparison.

7.2 Data

As described in Section 4, we use the same dataset as Etter et al. This dataset contains voting results (as the proportion of "yes" votes between 0 and 1) for 2352 municipalities for 289 unique votes. Eight of these votes contain incomplete information (some regional outcomes are missing); thus, similar to Etter et al., we remove these votes from consideration and consider only the 281 votes with complete region results. Additionally, in a manner similar to Etter et al., in order to standardize our prediction mechanism and to mitigate the impact

of varying popularity on different vote issues, we center results for each vote on the mean of that vote. For example, if 0.6 of the voters in a region voted "yes" on a vote, but 0.5 of voters voted "yes" nationally, we would store that region's entry as 0.1. A region that voted at 0.4 would be stored as -0.1. Also, just as Etter et al. does, we consider the first 231 votes as "training" votes and the final 50 votes as "test" votes for evaluation.

The dataset also contains "vote feature" information, with vote recommendations ("for," "against," or "no recommendation") from 25 different political parties for each of the 289 votes. Similar to Etter et al., in order to mitigate contributions from parties that always make the same recommendations or generally do not make a recommendation, we eliminate all recommendations from parties that have a variance less than 0.5. This leave us with recommendations from 13 political parties over 281 votes.

Finally, the dataset also contains "region feature" information. For each of the 2352 regions we consider, there are 25 region features. The features include region population, population density, language spoken, employment opportunities, age distribution, latitude, longitude, and political party participation, among other things. Some regions are missing information for certain features, so we replace that missing information with the national average for the feature in question. Additionally, we normalize the features, scaling them from 0 to 1, to facilitate learning and to reduce the impact of outliers on our predictions.

7.3 Etter et al.'s Code

We obtain Etter et al.'s MATLAB code in order to run his experiments and to generate a direct point of comparison for our trials. For each of the different models Etter et al. discusses in his experiment, his code first trains a corresponding model, then uses that model to make predictions for each test vote over a certain number of trials. Each of Etter et al.'s models additionally makes predictions about the national outcome of each vote, but we consider only the region-specific predictions in this paper.

7.4 Direct Comparison to Etter et al.

In order to compare directly to Etter et al.'s results, we seek to replicate aspects of Etter et al.'s experiments in our own experiments. For example, we use the same "reveal orders" that Etter et al.'s experiments use in our experiments. For each trial, out of the 2352 regions, Etter et al. selects a number, N_k of randomly distributed regions as "known regions," meaning that for those regions the true values of the test vote in question are already known. This information can be used to assist prediction for the "unknown regions." Additionally, among the "unknown regions," Etter et al. computes the root mean square error over a set of "test regions," only 10%, or 236, of the regions. In order to best compare to Etter et al., we re-ran his code and produced the reveal orders he uses. We use these orders so that for each of our 60 trials, our set of "known regions" and our "test regions" are the same as Etter et al.'s.

Additionally, we compare our results to Etter et al.'s after running the same number of trials. While Etter et al. describes running 500 trials for each test vote in his paper (with each trial containing a different set of known regions), we observe in our runs of his code that results have converged after 60 trials. The difference in results after 500 trials compared

to the results after 60 trials is negligible. Thus, we run our experiments and re-run Etter et al.'s experiments over 60 trials.

8 Results

In order to refine our HCR to most accurately predict voting outcomes, we analyze a series of experiments seeking to optimize the bootstrap model, the linking strategy, and the method of collective inference chosen.

8.1 Bootstrap

In order to determine the bootstrap model that makes the best initial predictions, we must choose which sets of features to use and how to use them. First, we set out to find the best performing type of regression. To do this, we run a simple experiment comparing three different regression types: linear regression, K-nearest neighbors, and ridge regression. While we explained linear regression in Section 2, ridge regression is a modified form of linear regression that includes a regularization term (this is comparable to Etter et al.'s implementation, which used linear regression with an assumed prior on the weights). When using the K-nearest neighbors regressor, we use the features to determine the K most similar nodes from the training data and make a prediction by taking a weighted average of those values.

Each of these respective regressors has an implementation we use in Python's SciKit-Learn module. For each experiment, we run 60 trials of our LIN(V) or LIN(R) model,

Table 3: Summary of selected results using different regressors for LIN(V) and LIN(R) bootstrap. Despite normalization, linear regression still chose weights that produced degenerate behavior in some cases (indicated by †), yielding RMSE greater than 1000%.

		Number of Known Regions, N_k							
Regressor	Model	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
HCR									
Linear Regression	LIN(V)	11.90	9.11	8.71	8.26	8.20	8.15	8.15	8.14
5-Nearest Neighbors	LIN(V)	12.50	9.83	9.45	9.02	8.96	8.92	8.91	8.91
10-Nearest Neighbors	LIN(V)	12.19	9.46	9.06	8.62	8.56	8.52	8.51	8.51
20-Nearest Neighbors	LIN(V)	12.03	9.28	8.88	8.43	8.37	8.33	8.32	8.32
40-Nearest Neighbors	LIN(V)	12.05	9.32	8.92	8.48	8.42	8.38	8.37	8.37
Ridge Regression	LIN(V)	**11.70**	**8.89**	**8.48**	**8.02**	**7.95**	**7.91**	**7.91**	**7.90**
Linear Regression	LIN(R)	**12.89**	10.16	10.93	†	†	6.24	6.10	6.03
5-Nearest Neighbors	LIN(R)	**12.89**	10.34	9.13	7.58	7.07	6.26	6.01	5.76
10-Nearest Neighbors	LIN(R)	**12.89**	10.34	9.98	7.79	7.23	6.27	**5.99**	**5.73**
20-Nearest Neighbors	LIN(R)	**12.89**	10.34	9.98	8.20	7.60	6.47	6.12	5.83
40-Nearest Neighbors	LIN(R)	**12.89**	10.34	9.98	9.13	8.09	6.82	6.38	6.02
Ridge Regression	LIN(R)	**12.89**	**9.20**	**8.44**	**7.03**	**6.62**	**6.16**	6.08	6.03
Etter									
Linear reg. with prior	LIN(V)	11.73	8.92	8.51	8.05	7.98	7.94	7.93	7.93
Linear reg. with prior	LIN(R)	12.89	9.54	8.68	7.07	6.65	6.17	6.08	6.03

performing a "bootstrap" using only the associated features (vote features for LIN(V), region features for LIN(R)). Between the experiments we vary only the regressor. We use $K = 5$, 10, 20, and 40 for the K-nearest neighbors regressor. The results, seen in Table 3, show that ridge regression provides us the best bootstrap results for all numbers of "known" regions (N_k) for LIN(V) (e.g., using the vote features). We see that for all values of N_k for LIN(V), $K = 20$ is the best performing version of K-nearest neighbors. On average for LIN(V), ridge regression outperforms outperforms 20-nearest neighbors by 0.40% and outperforms linear regression by about 0.23%. These results for ridge regression are similar to Etter et al.'s LIN(V) model.

For LIN(R), we see that ridge regression performs best for all values of N_k except for 1000 and 2116. This makes sense, as we are likely to get a better estimate using nearest neighbors when most of the neighbors are "known" values. Even though several nearest-neighbor variations outperform ridge regression when $N_k \geq 1000$ for LIN(V), ridge regression displays a clear superiority for smaller values of N_k and still outputs competitive predictions for $N_k = 1000$ and 2116. Our LIN(R) results (using ridge regression) often improve on Etter et al.'s LIN(R) and are otherwise similar. Thus, we proceed with using ridge regression for our future experiments.

Next, we consider how to best combine use of the vote and region features. We again run 60 trials of "bootstrap," but this time we vary our bootstrap models. We run experiments using each of the bootstrap models listed in Section 6.1: LIN(V), LIN(R), JOINT(R,V), BAYESD+INDPT(R,V), BAYESG+INDPT(R,V), and ENSEMBLE. We use ridge regression as the underlying regressor for each model.

As also seen in Table 3, the results in (Table 4) show that our LIN(V) and LIN(R) models (using ridge regression) are very close to Etter et al.'s LIN(V) and LIN(R). Our JOINT(R,V) directly uses Etter et al.'s implementation of LIN(R)+LIN(V), and thus has identical results.

We suspect that both the vote and the region features are informative, so we would expect the bootstrap models that use both sets of features to outperform those that just use one set of the features. The results in Table 4 show that this is largely true. LIN(V) and LIN(R) (the models that use only one set of features) are almost always outperformed

Table 4: Summary of selected "bootstrap" results using different "bootstrap" models.

	Number of Known Regions, N_k							
Strategy	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
HCR								
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
ENSEMBLE	11.74	8.36	7.67	**6.49**	**6.18**	**5.82**	**5.75**	**5.70**
Etter								
LIN(V)	11.73	8.92	8.51	8.05	7.98	7.94	7.93	7.93
LIN(R)	12.89	9.54	8.68	7.07	6.65	6.17	6.08	6.03
LIN(R) + LIN(V)	12.40	9.15	8.32	6.76	6.36	5.90	5.82	5.76

by the models that use both sets of features, with the exception of BayesD+Indpt(r,v), regardless of how many regions are "known."

BayesG+Indpt(r,v) and Ensemble appear to be our best performing bootstrap models, as one of these two models yield the best results for every value of N_k except for $N_k = 1$. In particular, BayesG+Indpt(r,v) appears to be our most accurate bootstrap model when N_k is smaller, while Ensemble outperforms BayesG+Indpt(r,v) when N_k is greater. Thus, without one bootstrap model displaying a clear overall superiority, we consider in further experiments below both the BayesG+Indpt(r,v) and Ensemble bootstrap models.

8.2 Links

By picking an effective bootstrap model, we hope to give our inference the best possible starting point. However, when we pick the links we use to represent the regions as a graph, we hope to generate connections that will enable the most effective collective (link-based) inference.

As described in section 6.2, we use a variety of strategies to generate links for our graph, storing the links using a sparse matrix from Python's SciPy library. In order to determine the most informative set of links, we run a set of experiments similar to those described in Section 8.1, but this time we vary the linking strategy instead of the bootstrap model. For all experiments, we run 10 iterations of RegressICA as our collective inference. We run separate experiments using each of the bootstrap models, BayesG+Indpt(r,v) and Ensemble, that were found most effective in Section 8.1. We run experiments using the different strategies mentioned in Section 6.2.

Our results, represented in Table 5, provide information regarding the effectiveness of different linking strategies. First, we compare different proximity-based strategies (top 3 rows of each section in the table). One of our early feasibility tests led us to suspect that using a distance around 10 km to generate links would lead to the most informative set of links. We compared several distance thresholds, and we show Prox-20km, Prox-10km, and Prox-5km in Table 5. We find that using an 10 km threshold, Prox-10km, does lead to the best results for all values of N_k except for 1.

Additionally, we tried a variety of weighting strategies in order to place greater emphasis on regions that were closer together. In Table 5, the Prox-10km-Wt method uses a simple linear weighting scheme, where a region that is exactly 10 km away will have a weight of 0 and a region that is 0 km away will have a weight of 1. We found that weighting had a relatively minor effect on results, but it did lead to an improvement in some cases. We suspect that this minor impact could be in part due to the relational feature we use with RegressICA, as weighting the "neighbor average" by distance will not have much of an impact if regions all share a similar prediction. Perhaps using different relational features would show that link weights had a greater impact. However, using the neighbor average, despite its resistance to link weighting, still seemed beneficial.

Next, we consider a variety of correlation-based linking strategies, instead of proximity-based. We consider three different strategies:

1. use all links that meet a minimum threshold. For example, all regions with a correlation coefficient greater than or equal to 0.80 become neighbors when using Corr-0.80.

2. use all links that meet a minimum threshold, but ensure there are a minimum number of links for each region (for example, Corr-0.80-Min-10).

3. use a fixed number of links for each region, and choose the best such links (for example, Corr-10).

We try two thresholds for the first strategy, one that is relatively high (Corr-0.80), and one that is relatively low (Corr-0.60). We find that the high threshold performs better when $N_k \leq 500$, indicating that having a small number of strongly correlated links is more effective than having a large number of links that are not as correlated. When many regions are "known" (when N_k is 1000 or 2116, for example), it follows that we gain accuracy picking more links by using the lower threshold, as we are certain of the values of so many linked regions.

For the second strategy, Corr-0.80-Min-10 and Corr-0.80-Min-20, we continue to use the higher threshold, but we set a minimum of 10 or 20 links. This means that for regions that do not have at least 10 (or 20) potential neighbors with correlations of at least 0.80, we pick the next highest correlated regions until each region has 10 (or 20) neighbors. For every value of N_k, Corr-0.80-Min-10 outperforms Corr-0.80-Min-20, and for every value of N_k except for 1, we find that Corr-0.80-Min-10 outperforms the minimum threshold-only strategy (Corr-0.80).

For the third strategy, we run an experiment where we link each region to its 10 "most

Table 5: Summary of selected results for ICA using various linking strategies.

Strategy		Number of Known Regions, N_k							
Links	Bootstrap	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
Prox-20km	Ensemble	12.48	9.11	8.42	7.38	7.06	6.29	5.98	5.83
Prox-10km	Ensemble	11.84	**8.52**	**7.85**	6.77	6.47	5.97	5.80	5.65
Prox-5km	Ensemble	**11.78**	8.69	8.12	7.22	6.97	6.40	6.17	5.98
Prox-10km-Wt	Ensemble	11.81	8.53	7.87	6.79	6.49	5.97	5.79	5.60
Corr-0.80	Ensemble	12.21	9.01	8.29	7.15	6.88	6.51	6.40	6.30
Corr-0.60	Ensemble	13.85	10.02	9.16	7.90	7.64	6.50	5.98	5.70
Corr-0.80-Min-10	Ensemble	12.33	8.92	8.08	6.61	6.25	5.70	5.49	5.31
Corr-0.80-Min-20	Ensemble	12.45	8.99	8.15	6.67	6.32	5.79	5.56	5.32
Corr-10	Ensemble	12.20	8.87	8.08	**6.64**	**6.24**	5.66	**5.46**	5.29
Corr-100	Ensemble	14.09	10.29	9.39	8.07	7.79	6.58	5.98	5.63
Corr-10-Wt	**Ensemble**	12.20	8.87	8.08	**6.64**	**6.24**	**5.65**	**5.46**	**5.28**
Corr-100-Wt	Ensemble	14.06	10.25	9.35	8.00	7.72	6.54	5.94	5.60
Prox-20km	BayesG+Indpt(r,v)	11.99	8.82	8.26	7.25	6.93	6.27	5.98	5.83
Prox-10km	BayesG+Indpt(r,v)	11.84	**8.45**	**7.84**	6.85	6.54	5.98	5.80	5.65
Prox-5km	BayesG+Indpt(r,v)	**11.82**	8.68	8.13	7.31	7.05	6.41	6.17	5.98
Prox-10km-Wt	BayesG+Indpt(r,v)	11.83	8.47	7.86	6.89	6.58	5.98	5.79	5.60
Corr-0.80	BayesG+Indpt(r,v)	12.01	8.82	8.19	7.21	6.93	6.52	6.40	6.30
Corr-0.60	BayesG+Indpt(r,v)	12.69	9.14	8.53	7.53	7.27	6.45	5.98	5.70
Corr-0.80-Min-10	BayesG+Indpt(r,v)	12.07	8.64	7.93	6.68	6.29	5.69	5.49	5.31
Corr-0.80-Min-20	BayesG+Indpt(r,v)	12.12	8.66	7.96	6.71	6.33	5.77	5.55	5.32
Corr-10	BayesG+Indpt(r,v)	11.99	8.63	7.96	**6.75**	**6.34**	5.66	**5.46**	5.29
Corr-100	BayesG+Indpt(r,v)	12.80	9.32	8.70	7.67	7.41	6.52	5.98	5.63
Corr-10-Wt	**BayesG+Indpt(r,v)**	11.99	8.63	7.96	**6.75**	**6.34**	**5.65**	**5.46**	**5.28**
Corr-100-Wt	BayesG+Indpt(r,v)	12.79	9.29	8.67	7.62	7.35	6.48	5.94	5.60

correlated" neighbors (Corr-10) and another where we links each region to its 100 "most correlated" neighbors (Corr-100). We find that exactly 10 neighbors for each region leads to our best results out of any of the correlation-based links thus far for all values of N_k except for 50.

To this point, we have been using unweighted correlation-based links. We next take our best correlation-based linking strategy, Corr-10, and weight the links by their correlation value. While leading to marginal improvements in some cases, we find that weighting links in this case makes minimal difference overall. We suspect that because each region has 10 links (out of 2351 possible neighbors), and we weight the links by correlation, it is unlikely that there is much variation in link weights. Thus, we also ran trials of weighted links using the Corr-100 linking strategy, as assigning 100 links to each region should lead to increased weight variation. We see that weighting the links (with Corr-100-Wt) still leads to only a relatively small improvement (vs. Corr-100), but the difference is more pronounced with 100 links compared to 10.

Out of all the linking strategies presented in Table 5, we find that Corr-10-Wt is the most effective overall strategy. While Prox-10km performs best amongst all strategies when N_k is 5 or 10 and Prox-5km performs best when N_k is 1, Corr-10-Wt is still an effective strategy for these values of N_k. These trends are consistent across both bootstrap methods. Thus, we move forward using Corr-10-Wt as our linking strategy for further experimentation. However, future work should perhaps consider the benefits that choosing one of the proximity-based strategies could have, especially when fewer regions are "known."

8.3 Collective Inference

Our experiments to this point have focused on providing the best possible starting point via bootstrap and the most informative information to use via links. Next, in order to determine our most promising strategy for vote prediction, we must determine how to most effectively leverage this bootstrap and link information via collective inference.

In Section 6.3, we describe nine distinct types of collective inference, two existing and seven that we created. To determine the best form of collective inference, we conduct a series of experiments similar to those desribed in Section 8.2. In these experiments, we use the top 10 weighted correlation links per region (Corr-10-Wt), we use our two best bootstrap methods (BayesG+Indpt(r,v)and Ensemble), and we run 10 inference iterations (finding that we usually reach a convergence by that point). Our results are displayed in Table 6 for Ensemble bootstrap. Because results with BayesG+Indpt(r,v)+Links had very similar trends, we defer them to Appendix A.2.

RegressICA is the first method of collective inference we use. A relatively simple model, we use these results as a baseline for comparison against other strategies. RegressICA uses both the links and the vote features in each inference iteration.

Next, we experiment using a set of strategies that use only the links during collective inference. For Weighted Vector, we see that for lesser values of N_k, our results are comparable to those obtained using RegressICA. However, RegressICA outperforms this model for greater values of N_k. For Bayesian inference, we first use BayesD+Links inference, where we use conditional link probabilities with a discrete approximation to predict a discrete value. This strategy produces our best results thus far for methods of collective

Table 6: Summary of selected results for various collective inference strategies using 10 weighted correlation links.

Strategy		Number of Known Regions, N_k							
Inference	Bootstrap	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
RegressICA	Ensemble	12.20	8.87	8.08	6.64	6.24	5.65	5.46	5.28
WV	Ensemble	12.07	8.84	8.14	6.98	6.63	5.94	5.62	5.33
BayesD+Links	Ensemble	11.90	8.59	7.95	6.91	6.59	5.91	5.59	5.30
BayesG+Links	Ensemble	11.87	8.57	7.94	6.87	6.52	5.81	5.52	5.27
WV+Delta	Ensemble	11.98	8.65	7.88	**6.48**	**6.09**	**5.50**	**5.31**	**5.15**
BayesD+Indpt(r,v)+Links	Ensemble	11.88	8.51	7.88	6.91	6.61	5.93	5.60	5.30
BayesG+Indpt(r,v)+Links	Ensemble	11.86	8.48	7.86	6.86	6.53	5.82	5.53	5.27
BayesG+Joint(r,v)+Links	Ensemble	11.88	8.60	7.94	6.74	6.36	5.69	5.44	5.22
BayesG+Ensemble(r,v)+Links	Ensemble	**11.80**	**8.46**	**7.81**	6.64	6.31	5.78	5.59	5.59

inference for $N_k <= 10$. We hope to improve using BayesG+Links, where we approximate conditional probabilities as Gaussians to enable exact inference. Our results show that BayesG+Links improves upon BayesD+Links for all values of N_k by an average of about 0.046%. In fact, it is our best collective inference method thus far when N_k is 1, 5, 10, or 2116. We hope for continued improvement once we incorporate region and vote features.

Thus far, our strategies for collective inference have incorporated the vote features and the region features only during bootstrap. Our remaining methods also include these features during collective inference. The first is Weighted Vector+Delta. Adding a vote feature-based region-specific delta to our inference greatly improves the accuracy of this strategy. We observe our best results in this set of experiments for the 50, 100, 500, 1000, and 2116 values of N_k. Compared to Weighted Vector (without a delta), this method improves by an average of about 0.17%.

Next, we include region and vote features into our Bayesian inference. First, we examine the impact on BayesD+Indpt(r,v)+Links. We find that incorporating the region and vote features leads to slight improvements (compared to BayesD+Links) when N_k is 1, 5, and 10, but results are slightly worse when N_k is 100, 500, and 1000. When we compare BayesG+Indpt(r,v)+Links against BayesD+Indpt(r,v)+Links (e.g., now approximating the conditional probabilities as Gaussians), we observe that incorporating the region and vote features does lead to slight improvement for smaller values of N_k and has a minimal impact on results otherwise. BayesG+Joint(r,v)+Links leads to improvements for when N_k is 50, 100, 500, 1000, and 2116, compared to the other forms of Bayesian inference we have discussed thus far. For BayesG+Ensemble(r,v)+Links, we achieve some of our best results when N_k is 1, 5, or 10. However, it does not perform well for greater values of N_k.

Because we generate a histogram or Gaussian for each of every region's ten neighbors, it is likely that conditional probabilities associated with the links have a greater impact on the final predictions than those associated with the region or vote features. When our Bayesian inference is Gaussian, the impact of region and vote features is small, and when our Bayesian inference is discrete, the impact is not significant enough to affect the discrete prediction selected. We see more noticeable improvements when we combine region and vote features into a joint Gaussian (e.g., Joint(r,v)), and it is possible that with lesser variance in the Gaussian shape (and therefore more certainty), the features are able to have a greater impact.

Overall, we observe that using the region and vote features does improve our collective inference. WEIGHTED VECTOR improves the most, and while our forms of Bayesian inference do generally experience some improvement, their improvement is relatively small in comparison. Additionally, we observe that the Gaussian form of Bayesian inference continues to outperform the discrete form of collective inference. Finally, interestingly, comparing the results in Table 6 to those in Table 4 shows that when N_k is 1, 5, 10, 50, or 100, our best forms of bootstrap alone outperform our best collective inference. It appears that when we are less certain of our neighbors, iterative inference can actually be more harmful than helpful. The next section addresses this issue.

8.4 Improving Inference via Variance Scaling

In Section 8.3, we 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 WEIGHTED VECTOR. We hypothesized that our Bayesian models were overly reliant on the "links."

Additionally, we observed that, especially for smaller values of N_k, our bootstrap models actually outperformed our collective inference. We suspected that when fewer neighbors were "known," and we were therefore relying on more uncertain predicted values, iterative inference could potentially magnify errors.

An emerging theme is that uncertainty in neighbors is damaging for our collective inference. The problem with our Bayesian inference is that it treats all neighbors the same. In particular, Equation 7 uses the same term, $\mathcal{N}(y_j, \widehat{\sigma}^2_{links})$, for every neighbor (albeit with varying predictions of y_j), regardless of the certainty of y_j. In order to improve our Bayesian inference, we must teach it to discriminate between certain and uncertain values, "known" and "unknown" neighbor predictions.

When we are performing Bayesian inference using Gaussians, many of the "link" Gaussians we incorporate into our inference come from "unknown" neighbors. The impact of a predicted mean that is incorrect but has a low variance can be particularly damaging. We recall Equation 5 for the mean of the product of two Gaussians:

$$\mu_{fg} = \frac{\mu_f \sigma_g^2 + \mu_g \sigma_f^2}{\sigma_f^2 + \sigma_g^2}$$

Suppose that f is a Gaussian with a mean, μ_f, that is incorrect, but has a low variance, σ_f^2. Suppose that g is a Gaussian with a mean, μ_g, that is correct, but has a higher variance, σ_g^2. In the numerator of the equation above, we multiply μ_f (incorrect) by g's variance (high), and we multiply μ_g (correct) by f's variance (low). Therefore, μ_f will have a greater impact on our product (the resultant Gaussian that can become our prediction), even though it is incorrect, simply because f has a lower variance. In order to combat this, we propose scaling the variance of any Gaussian of which we are uncertain. Thus, we ensure Gaussians of which we are more certain have a greater impact on our calculations.

In our inference, all of the "link" Gaussians have the same variance, $\widehat{\sigma}^2_{links}$ (they are the same Gaussian shape, just centered differently based on the current prediction for the associated region). We make no distinction between the impact we allow each "link" Gaussian

Table 7: Summary of selected results varying the variance scale factor, using BayesG+Indpt(r,v)+Links inference, BayesG+Joint(r,v)+Links inference, and BayesG+Ensemble(r,v)+Links inference with 10 weighted correlation links.

Strategy			Number of Known Regions, N_k							
Inference	Bootstrap	Scale Factor	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
BayesG+Indpt(r,v)+Links	Ensemble	1.0	11.86	8.48	7.86	6.86	6.53	5.82	5.53	5.27
BayesG+Indpt(r,v)+Links	Ensemble	2.0	**11.85**	8.34	7.73	6.62	6.23	5.58	**5.40**	**5.26**
BayesG+Indpt(r,v)+Links	Ensemble	4.0	11.95	8.23	7.59	**6.46**	**6.09**	**5.56**	**5.40**	**5.26**
BayesG+Indpt(r,v)+Links	Ensemble	8.0	11.96	**8.16**	**7.53**	**6.46**	6.13	5.60	5.41	**5.26**
BayesG+Indpt(r,v)+Links	Ensemble	16.0	12.01	8.17	7.54	6.50	6.18	5.63	5.41	**5.26**
BayesG+Joint(r,v)+Links	Ensemble	1.0	11.88	8.60	7.94	6.74	6.36	5.69	5.44	5.22
BayesG+Joint(r,v)+Links	Ensemble	2.0	11.87	8.60	7.90	6.55	6.12	5.49	5.32	**5.20**
BayesG+Joint(r,v)+Links	Ensemble	4.0	11.83	8.56	**7.86**	**6.50**	**6.07**	**5.46**	**5.30**	**5.20**
BayesG+Joint(r,v)+Links	Ensemble	8.0	**11.82**	**8.55**	**7.86**	6.52	6.10	5.47	**5.30**	**5.20**
BayesG+Joint(r,v)+Links	Ensemble	16.0	**11.82**	8.56	7.87	6.53	6.12	5.48	**5.30**	**5.20**
BayesG+Ensemble(r,v)+Links	Ensemble	1.0	11.80	8.46	7.81	6.64	6.31	5.78	5.59	**5.59**
BayesG+Ensemble(r,v)+Links	Ensemble	2.0	11.77	8.41	7.73	6.50	6.15	5.64	**5.53**	**5.59**
BayesG+Ensemble(r,v)+Links	Ensemble	4.0	11.72	8.35	7.65	**6.42**	**6.09**	**5.63**	**5.53**	**5.59**
BayesG+Ensemble(r,v)+Links	Ensemble	8.0	**11.70**	**8.33**	**7.64**	6.43	6.11	5.65	**5.53**	**5.59**
BayesG+Ensemble(r,v)+Links	Ensemble	16.0	11.72	8.35	7.66	6.46	6.13	5.66	**5.53**	**5.59**
Etter Best			11.73	8.35	7.54	6.03	5.68	5.18	5.01	4.87

to have through its associated conditional probability, $P(y_j|y_d = y)$, regardless of whether that Gaussian represents a "known" or an "unknown" neighbor. For example, if a region has 10 neighbors, and each of them are "unknown" regions, it is intuitive that those Gaussians should have less of an impact on our calculations than the "link" Gaussians of a regions with 10 "known" neighbors.

In order to allow Gaussians in which we are more certain to have a greater impact on our predictions, we scale the variance, $\widehat{\sigma}^2_{links}$, of "unknown" neighbors. Table 7 shows the results of applying variance scaling to the BayesG+Indpt(r,v)+Links strategy, where we multiply the variance, $\widehat{\sigma}^2_{links}$, of any "unknown" neighbor by the scale factor.

We see that using variance scaling allows us to rely more heavily on the features of our inference that we are more certain of. In doing so, we improve our accuracy for every value of N_k. In fact, we obtain our best results thus far for 1, 5, 10, 50, 100, and 1000 "known" regions, and our best results are better than Etter et al.'s for 1, 5, and 10 "known" regions.

8.5 Etter et al. Comparison

A collection of our best results compared against Etter et al.'s best results is presented in Table 8. Note that for the Bayesian inference collective inference strategies presented, for each value of N_k, we choose our best observed scale factor. (Section 9.4 discusses automatic selection of this parameter.)

We improve on Etter et al.'s best results for 1, 5, and 10 "known" regions. Our most significant improvement is for 5 "known" regions, where we decrease the RMSE by 0.19% compared to Etter et al.'s MF+GP(r)+LIN(v) model.

We notice that LIN(v), a bootstrap model alone without inference, ties for our best performance when there is 1 "known" region with the BayesG+Ensemble(r,v)+Links model. Throughout our experiments, we oftentimes observed that iterations of inference were more harmful than helpful when very few regions were "known." Aside from the Weighted Vector+Delta model's performance as our best method of inference when 2116 regions

Table 8: Comparison of our best results against Etter et al.'s. Our best results and Etter et al.'s best results for each number of "known" regions are in bold. The best overall result for each N_k value (our strategy or Etter et al.'s) is indicated by a †. Note for each BayesG strategy and for each value of N_k, we use the result associated with the optimum variance scaling factor.

	Number of Known Regions, N_k							
Strategy	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
HCR								
LIN(V)	**11.70**†	8.89	8.48	8.02	7.95	7.91	7.91	7.90
BayesG+Indpt(R,V)+Links+VarScl	11.85	**8.16**†	**7.53**†	6.46	6.09	5.56	5.40	5.26
BayesG+Joint(R,V)+Links+VarScl	11.82	8.55	7.86	6.50	**6.07**	**5.46**	**5.30**	5.20
BayesG+Ensemble(R,V)+Links+VarScl	**11.70**†	8.33	7.64	**6.42**	6.09	5.63	5.53	5.59
WV+Delta	11.98	8.65	7.88	6.48	6.09	5.50	5.31	**5.15**
Etter								
LIN(V)	**11.73**	8.92	8.51	8.05	7.98	7.94	7.93	7.93
MF+GP(R)	12.89	8.90	7.76	**6.03**†	**5.68**†	**5.18**†	**5.01**†	**4.87**†
MF+GP(R)+LIN(V)	11.84	**8.35**	**7.54**	6.20	5.86	5.37	5.23	5.15

are "known" and the LIN(V) bootstrap performance mentioned above, we observe that our best performing models are variations of BayesG with variance scaling.

Etter et al.'s models are effective, particularly MF + GP(r) and MF + GP(r) + LIN(v). While we have improved upon the performance of Etter et al.'s models when there are few "known" regions, their models still outperform ours when there are 50 or more "known" regions. However, their models are extremely computationally expensive. Based on runtime comparisons between Etter et al.'s code and our code using the same hardware, we have found that our models in particular run much faster than Etter et al.'s most accurate models.[2] In particular, our BayesG models run approximately 133 times faster than Etter et al.'s MF + GP(r) + LIN(v) model and approximately 933 times faster than Etter et al.'s MF + GP(r) model.

Through our Gaussian Bayesian inference paradigm, we have presented an efficient method of inference that is competitive with Etter et al.'s models, even beating them in some cases. We successfullly leveraged link and feature data, while remaining conscious of the certainty of our predictions, in order to accurately predict voting outcomes. We hope to continue to refine and improve our paradigm's performance through future work.

9 Future Work

We consider several areas which we believe could be helpful in improving our results.

9.1 Further Exploration of Weighted Proximity-Based Links

We used Corr-10-Wt links for all of our experiments beginning in Section 8.3 and going forward. While the Corr-10-Wt strategy outperformed any set of proximity links when there were 50 or more "known" regions, the Prox-10km and Prox-10km-Wt strategies performed better than the Corr-10-Wt strategy when there were 1, 5, or 10 "known" regions.

[2]For both our code and Etter et al.'s, the runtime is dominated by training the model, while inference runs much more quickly.

In our final results, we improved on Etter et al.'s results when there were 1, 5, or 10 "known" regions. This was despite using the Corr-10-Wt strategy for our experiments, when the Prox-10km strategy outperformed the Corr-10-Wt strategy at those "known" region combinations. Thus, repeating the experiments from Section 8.3, using the Prox-10km linking strategy, may provide further gains.

9.2 Better Weighting Strategies for Correlation-Based Links

In Section 8.2, we hypothesized that weighting the 10 correlation-based links by their correlations (10-Corr-Wt) made minimal difference because of the small range of correlations. When we use only 10 links, out of a possible 2351 neighbors for each region, in most cases there will be little difference in a region's correlation with its most-correlated region vs. with its tenth most-correlated region.

We propose instead a weighting strategy similar to the way we weighted proximity-based links, where we scale the set of links from a weight of 0 to a weight of 1. For example, if the highest-correlated region had a correlation of 0.95 and the tenth highest-correlated region had a correlation of 0.85, we could scale the weights to 1.0 and 0.0 respectively, leading to a greater range of link weights in between the most and least-correlated link. We should note, however, that setting the least-correlated link's weight to 0.0 would effectively eliminate a link. Thus, we would have to either add an extra link in order to maintain 10 effective links or would have to set a "weight floor" above 0.0.

9.3 Integrating Region Features into Weighted Vector

The Weighted Vector+Delta method of collective inference resulted in our best results for 2116 "known" regions and provided results that were nearly as accurate as many of our best results that used variance scaling for other "known region" combinations. This collective inference method was able to do all this despite using only the vote features during collective inference.

Currently, we train a "vote delta" based on the vote features to provide a correction to the Weighted Vector model's inference method of using the "neighbor average" as its prediction. We propose training an additional "region delta," using the region features for "known" regions, to provide a further correction to the "neighbor average" plus the "vote delta." In theory, this should provide further gains, especially when more regions have known values.

9.4 Automatic Variance Scaling

We have had considerable success with variance scaling, as many of our best results have been obtained using the strategy. However, thus far, our selection of the scale factor has been limited to experimenting with a range of possible scale factors and comparing the results. We propose to instead automatically set the scale factor during training.

We could do so using well-known methods for "cross validation." In our case, for example, where we have 231 training votes, we would use 201 of the votes as "scale training" data and 30 of the votes as "scale testing" data. We would then run our inference on this data,

starting with a certain value for the scale factor. For each scale factor value we try, we would compute the RMSE between the our predictions and the "scale testing" data. We would adjust the scale factor value based on whether the RMSE improved or worsened compared to the previous run. We would arrive at an optimal scale factor value, and then would use that during our actual run.

10 Conclusion

Link-based classification (LBC) has been shown to substantially improve accuracy in a variety of domains, but prior to this undertaking had not been successfully applied to continuous domains, nor to heterogeneous domains where rich temporal/historical information could potentially enable node-specific models. Our results show that extending the LBC paradigm to Heterogeneous Collective Regression (HCR) 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 our multiple versions of neighbor averaging. In addition, we have outperformed Etter et al.'s best results when there are few "known" regions. This implies that our models could be more useful for vote prediction leading up to an election or early during the "day of" the election (when few regions have reported results). Additionally, we could obtain more accurate predictions using our models after extensively canvassing a small number of regions than we could using Etter et al.'s. Finally, our best techniques are more than 100 times faster than Etter et al.'s, which is especially important when the objective is "day of voting" prediction.

This success gives credible evidence of HCR's abilities to address the limitations of LBC. Applying this strategy to Etter et al.'s voting domain demonstrates the potential for HCR to be used in novel ways for voting prediction. This new application of Machine Learning could be of interest to election experts, social data analysts, campaigns, and special interest groups. Moreover, our HCR algorithm can be extended to other applications of relational data as well. Connections are apparent to profit prediction, temperature prediction, movie preference prediction, and other types of forecasting. By showing the viability of using regression to predict continuous outputs in heterogeneous environments, this work opens the door to numerous other applications of HCR.

References

[1] Bromiley, P. Products and convolutions of gaussian probability density functions. *Tina-Vision Memo 3*, 4 (2003), 1.

[2] Etter, V., Khan, M. E., Grossglauser, M., and Thiran, P. Online collaborative prediction of regional vote results. In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on* (2016), IEEE, pp. 233–242.

[3] Jensen, D., Neville, J., and Gallagher, B. Why collective inference improves relational classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2004), ACM, pp. 593–598.

[4] Kalman, R. E. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering 82*, 1 (1960), 35–45.

[5] Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer 42*, 8 (2009).

[6] Loglisci, C., Appice, A., and Malerba, D. Collective regression for handling autocorrelation of network data in a transductive setting. *Journal of Intelligent Information Systems 46*, 3 (2016), 447–472.

[7] Lu, Q., and Getoor, L. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)* (2003), pp. 496–503.

[8] Macskassy, S. A., and Provost, F. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research 8*, May (2007), 935–983.

[9] McDowell, L., and Aha, D. Semi-supervised collective classification via hybrid label regularization. *29th International Conference on Machine Learning (ICML 2012)* (2012).

[10] McDowell, L. K., and Aha, D. W. Labels or attributes?: rethinking the neighbors for collective classification in sparsely-labeled networks. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management* (2013), ACM, pp. 847–852.

[11] Neville, J., and Jensen, D. Iterative classification in relational data. In *Proc. of the Workshop on Learning Statistical Models from Relational Data at AAAI-2000* (2000), pp. 13–20.

[12] Neville, J., and Jensen, D. Relational dependency networks. *Journal of Machine Learning Research 8*, Mar (2007), 653–692.

[13] Neville, J., Jensen, D., and Gallagher, B. Simple estimators for relational bayesian classifiers. In *Third IEEE International Conference on Data Mining* (2003), IEEE, pp. 609–612.

[14] Ng, A. Coursera: Machine learning online course, 2016.

[15] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine 29*, 3 (2008), 93.

A Appendices

A.1 Derivation of Dividing Two Gaussians

Considering the equations provided by Bromiley for the product of two Gaussians (Equation 5), we can derive the equations to calculate the mean and standard deviation of the Gaussian resultant when one Gaussian is divided by another Gaussian.

First, we divide our function $f(x)$ by our function $g(x)$. Due to the properties of exponents we express the result as the natural number raised to the difference between our original Gaussian exponents.

$$\frac{f(x)}{g(x)} = \frac{e^{\frac{-(x-\mu_f)^2}{2\sigma_f^2}}}{e^{\frac{-(x-\mu_g)^2}{2\sigma_g^2}}} = e^{-(\frac{(x-\mu_f)^2}{2\sigma_f^2} - \frac{(x-\mu_g)^2}{2\sigma_g^2})}$$

Next, we set β equal to the resultant exponent from the step above. We then multiply out the quadratics and combine terms.

$$\begin{aligned}
\beta &= \frac{(x-\mu_f)^2}{2\sigma_f^2} - \frac{(x-\mu_g)^2}{2\sigma_g^2} \\
&= \frac{x^2 - 2\mu_f + \mu^2}{2\sigma_f^2} - \frac{x^2 - 2\mu_g + \mu_g^2}{2\sigma_g^2} \\
&= \frac{x^2\sigma_g^2 - 2\mu_f\sigma_g^2 + \mu_f^2\sigma_g^2 - x^2\sigma_f^2 + 2\mu_g\sigma_f^2 - \mu_g^2\sigma_f^2}{2\sigma_f^2\sigma_g^2}
\end{aligned}$$

Next, we factor out the coefficient of the x^2 term from each term.

$$\beta = \frac{x^2(\sigma_g^2 - \sigma_f^2) - 2x(\mu_f\sigma_g^2 - \mu_g\sigma_f^2) + \mu_f^2\sigma_g^2 - \mu_g^2\sigma_f^2}{2\sigma_f^2\sigma_g^2}$$

Dividing by this factor leaves us with a function quadratic in x.

$$\beta = \frac{x^2 - \frac{2x(\mu_f\sigma_g^2 - \mu_g\sigma_f^2)}{\sigma_g^2 - \sigma_f^2} + \frac{\mu_f^2\sigma_g^2 - \mu_g^2\sigma_f^2}{\sigma_g^2 - \sigma_f^2}}{\frac{2\sigma_f^2\sigma_g^2}{\sigma_g^2 - \sigma_f^2}}$$

Thus, as this function is quadratic, the mean is half the coefficient of the x term and the standard deviation is the square root of half the denominator:

$$\sigma_{f/g} = \sqrt{\frac{\sigma_f^2\sigma_g^2}{\sigma_g^2 - \sigma_f^2}} \quad \text{and} \quad \mu_{f/g} = \frac{\mu_f\sigma_g^2 - \mu_g\sigma_f^2}{\sigma_g^2 - \sigma_f^2}$$

We must note, however, that if the standard deviation of the Gaussian serving as the dividend is greater than the standard deviation of the Gaussian serving as the divisor, we are left with a negative value in the denominator of the fraction within the radical, leading to an imaginary value. Thus, if the standard deviation of a Gaussian is greater than the standard deviation of the Gaussian it is divided by, the resulting shape is not a Gaussian. In order to

mitigate this possibility, we only divide by a Gaussian after completing all required Gaussian multiplications, as multiplying Gaussians tends to decrease the standard deviation.

A.2 Additional Collective Inference Results

Table 9: Summary of selected results for various collective inference strategies using 10 weighted correlation links.

Strategy		Number of Known Regions, N_k							
Inference	Bootstrap	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
RegressICA	BayesG+Indpt(r,v)	11.99	8.63	7.96	6.75	6.34	5.65	5.46	5.28
WV	BayesG+Indpt(r,v)	12.03	8.75	8.14	7.10	6.72	5.94	5.62	5.33
BayesD+Links	BayesG+Indpt(r,v)	12.12	8.55	7.96	6.98	6.64	5.92	5.59	5.30
BayesG+Links	BayesG+Indpt(r,v)	12.09	8.54	7.94	6.92	6.56	5.81	5.52	5.27
WV+Delta	BayesG+Indpt(r,v)	11.83	**8.46**	**7.79**	**6.60**	**6.19**	**5.50**	**5.31**	**5.15**
BayesD+Indpt(r,v)+Links	BayesG+Indpt(r,v)	12.32	8.61	8.01	7.03	6.70	5.94	5.60	5.30
BayesG+Indpt(r,v)+Links	BayesG+Indpt(r,v)	12.32	8.59	7.99	6.97	6.61	5.83	5.53	5.27
BayesG+Joint(r,v)+Links	BayesG+Indpt(r,v)	11.99	8.50	7.87	6.75	6.37	5.70	5.44	5.22
BayesG+Ensemble(r,v)+Links	BayesG+Indpt(r,v)	**11.80**	**8.46**	7.81	6.64	6.31	5.78	5.59	5.59

A.3 Additional Variance Scaling Results

Table 10: Summary of selected results varying the variance scale factor, using BayesG+Indpt(r,v)+Links, BayesG+Joint(r,v)+Links, and BayesG+Ensemble(r,v)+Links with 10 weighted correlation links. The best results for each inference are in bold.

Strategy			Number of Known Regions, N_k							
Inference	Bootstrap	Scale Factor	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2116**
BayesG+Indpt(r,v)+Links	BayesG+Indpt(r,v)	1.0	12.32	8.59	7.99	6.97	6.61	5.83	5.53	5.27
BayesG+Indpt(r,v)+Links	BayesG+Indpt(r,v)	2.0	12.18	8.42	7.79	6.65	6.25	5.58	**5.40**	**5.26**
BayesG+Indpt(r,v)+Links	BayesG+Indpt(r,v)	4.0	12.01	8.23	7.59	**6.46**	**6.09**	**5.56**	**5.40**	**5.26**
BayesG+Indpt(r,v)+Links	BayesG+Indpt(r,v)	8.0	**11.96**	**8.16**	**7.53**	**6.46**	6.13	5.60	5.41	**5.26**
BayesG+Indpt(r,v)+Links	BayesG+Indpt(r,v)	16.0	12.01	8.17	7.54	6.50	6.18	5.63	5.41	**5.26**
BayesG+Joint(r,v)+Links	BayesG+Indpt(r,v)	1.0	11.99	8.50	7.87	6.75	6.37	5.70	5.44	5.22
BayesG+Joint(r,v)+Links	BayesG+Indpt(r,v)	2.0	11.85	8.57	7.89	6.55	6.12	5.49	5.32	**5.20**
BayesG+Joint(r,v)+Links	BayesG+Indpt(r,v)	4.0	11.83	8.56	**7.86**	**6.50**	**6.07**	**5.46**	**5.30**	**5.20**
BayesG+Joint(r,v)+Links	BayesG+Indpt(r,v)	8.0	**11.82**	**8.55**	**7.86**	6.52	6.10	5.47	**5.30**	**5.20**
BayesG+Joint(r,v)+Links	BayesG+Indpt(r,v)	16.0	**11.82**	8.56	7.87	6.53	6.12	5.48	**5.30**	**5.20**
BayesG+Ensemble(r,v)+Links	BayesG+Indpt(r,v)	1.0	11.99	8.44	7.83	6.68	6.34	5.78	5.59	**5.59**
BayesG+Ensemble(r,v)+Links	BayesG+Indpt(r,v)	2.0	11.85	8.37	7.71	6.50	6.16	5.64	**5.53**	**5.59**
BayesG+Ensemble(r,v)+Links	BayesG+Indpt(r,v)	4.0	11.73	8.34	7.65	**6.42**	**6.09**	**5.63**	**5.53**	**5.59**
BayesG+Ensemble(r,v)+Links	BayesG+Indpt(r,v)	8.0	**11.70**	**8.33**	**7.64**	6.43	6.11	5.65	**5.53**	**5.59**
BayesG+Ensemble(r,v)+Links	BayesG+Indpt(r,v)	16.0	11.72	8.35	7.66	6.46	6.13	5.66	**5.53**	**5.59**