Faster Kriging on Graphs

Omkar Muralidharan

Abstract

[Xu et al. 2009] introduce a graph prediction method that is accurate but slow. My project investigates faster methods based on theirs that are nearly as accurate. If we have N nodes, their method is $\mathcal{O}(N^3)$, whereas mine are $\mathcal{O}\left(O^3 + O^2P\right)$ if we have O observations and P predictions to make.

1 Introduction

Suppose we have a graph G, and we are interested in a certain quantity associated with each node.Suppose we observe this quantity for some of the nodes. We would like to use the graph structure to estimate the quantity on the rest of the nodes.

For example, G could be a social network, and we could be interested in the politics of the members. We might be able to survey some of the members. We would like to be able to use the survey results to estimate the politics of the other members.

[Xu et al. 2009] use a method called *kriging* to do this. Kriging is a spatial smoothing technique from geostatistics; we will explain how [Xu et al. 2009] apply it to graphs in Section 2. [Xu et al. 2009] show that kriging generalizes many older graph smoothing methods. Their results show that kriging is better able to use the information from the graph, and is thus more accurate.

The approach of [Xu et al. 2009], however, has a serious problem - it is simply too slow to scale to large data sets. If G has N nodes, their method is $O(N^3)$. This makes it infeasible for the problems we are interested in, which can have N in the millions.

This project tries to find faster ways to do the kriging. I looked at two methods. The first simply skips an expensive SVD step in the original method. The second is a new regression approach that is actually different from the original kriging method. The relationship between the regression approach and the original method is roughly analogous to that between logistic regression and LDA - the regression approach is less model-centric (though here it actually uses a different model). Finally, I looked at ways to combine the two methods.

The resulting methods perform well in simulations, usually not much worse than the original kriging. They are also much faster. If we have O observations and we need P predictions, the first method is $O(O^3 + O^2P)$, and the second is $\mathcal{O}(O^2 + OP)$. Since $O \ll N$ and sometimes $P \ll N$ as well, these can be much faster than the original method. They can even be applied to huge problems where the original method is totally infeasible.

2 Graph Kriging

In this section, I will briefly explain the method of [Xu et al. 2009]. Suppose the graph G has node-measurements X. We observe $X_o = X_i, i \in B$ and want to predict some or all of the rest. Suppose we use G to create similarities between the nodes - s_{ij} is the similarity between nodes i and j. We want to use these similarities to help our predictions.

We have to assume something about the relationship between the nodes, or this problem is impossible. [Xu et al. 2009] treat the similarities like distances, and use a geostatistics model. They model X as multivariate normal, where the correlations are a function of the similarities:

$$cor(X_i, X_j) = f(s_{ij}).$$

For simplicity, suppose that marginally, each $X_j \sim \mathcal{N}(0, 1)$. Let $\Sigma = f(S)$ be the covariance matrix, partitioned as

$$\Sigma = \left[\begin{array}{cc} \Sigma_{oo} & \Sigma_{om} \\ \Sigma_{mo} & \Sigma_{mm} \end{array}\right]$$

for the observed and missing elements. Then the Bayes estimator of $X_m = X_i, i \notin B$ is

$$E(X_m|X_o) = \Sigma_{mo}\Sigma_{oo}^{-1}X_o. \quad (1)$$

So if we know the function f that takes covariances into similarities, we can make our predictions.

If we don't assume normality, and simply assume that $cov(X) = \Sigma = f(S)$, we can still use the estimator in equation 1. Instead of being the Bayes estimator, though, it becomes the best linear estimator of X_m based on X_o. So although kriging uses the normal model, we can relax that assumption substantially and still get good estimators.

There are a few more complications that come from measurement error and estimating the variances, but I will ignore them for this project, since they don't really matter for my purposes.

[Xu et al. 2009] show that many graph smoothing methods can be viewed in this light, with specific choices of similarities and correlation functions f. They obtain substantial improvements by following the geostatistics approach. Instead of using a fixed f, they fit f using the data. Their procedure is particularly simple in our simplified situation. First, they find the empirical correlations $\hat{\rho}_{ij}$ of the observed data, here X_iX_j for $i, j \in B$. They then smooth $\hat{\rho}_{ij}$ as a function of s_{ij} using splines. This gives an estimated correlation function $\hat{f}$. Since we have $\binom{O}{2}$ empirical correlations, we usually have enough data to fit $\hat{f}$ quite well.

They now use $\hat{f}$ to get $\tilde{\Sigma} = \hat{f}(S)$. Since $\tilde{\Sigma}$ may not be positive definite, they use an SVD of $\tilde{\Sigma}$ to force it to be positive definite (and possibly force it to be low rank). Finally, they plug this new $\hat{\Sigma}$ into equation 1 to get an estimator of X_m.

This method turns out to be substantially more accurate than older smoothing methods [Xu et al. 2009] consider. The kriging approach is better able to take advantage of the similarities, since it does not make assumptions about f, it does not suffer when assumption on f are wrong. Its flexibility allows it to fit the more complicated similarity relationships that occur in real data.

2.1 Problems

Unfortunately, this method has some speed problems that make it impossible to apply to large data sets. The SVD of $\tilde{\Sigma}$ takes $O(N^3)$ time. Since N is often very large, this is prohibitively slow. With careful estimates of $\hat{f}$ that produce a sparse $\tilde{\Sigma}$, this can be reduced, but no matter what we do, we still have to take the SVD of a very large matrix.

Another problem is that we cannot get predictions on a small subset of the missing nodes. If we decide we only need predictions for certain nodes X_p, we still have to take that SVD, so we end up doing the same expensive step as if we got predictions for all the X_m. This is inconvenient, since we may often want quick predictions for a small number of interesting nodes.

Finally, if we get new data, we have to re-fit the whole model. The spline fit of $\hat{f}$ can be updated fairly quickly, but in general, every element of $\tilde{\Sigma}$ will change, so we will need a new SVD. This is expensive, and undesireable. Even if we could do such a big SVD once, it is unlikely we can afford to do it every time we get new data.

These problems make [Xu et al. 2009]'s kriging method unsuitable for large problems. We can, however, make faster methods based on [Xu et al. 2009]'s work.

3 Faster Graph Kriging

In this section, I'll explain two methods I used to make kriging-like predictions much more quickly.

3.1 Using a Raw Covariance Estimate

The first method is very simple. Instead of computing the SVD of $\tilde{\Sigma}$ and forcing it to be positive definite, just force the observation submatrix $\tilde{\Sigma}_{oo}$ to be positive definite. Then plug this new $\hat{\Sigma}$ into equation 1 to get predictions.

This saves a lot of computation, since we have to take the SVD of an $O \times O$ matrix instead of an $N \times N$ one. We can save even more time if we only need predictions for a subset of the missing values - all we have to do is use $\tilde{\Sigma}_{po} = \hat{cov}\left(X_p, X_o\right)$ in equation 1 instead of $\tilde{\Sigma}_{mo}$. If we need $P \ll N$ predictions, this can be much faster. The method is $\mathcal{O}(O^3 + OP)$, much faster than the $\mathcal{O}(N^3)$ original method. It is worth noting, however, that this still has the updating problem of the original method.

We will see in simulations that this method works well, but can be very erratic. One reason this happens because the SVD step of the original algorithm sets all the negative eigenvalues of $\tilde{\Sigma}$ to 0. In some cases, this removes a substantial fraction of the energy of the matrix, and therefore regularizes $\tilde{\Sigma}_{op}$. By only forcing $\tilde{\Sigma}_{oo}$ to be positive definite, we lose the benefits of this step. Another reason is instability when taking the inverse of $\tilde{\Sigma}_{oo}$ - if the matrix is near singular, errors are greatly amplified.

3.2 A Regression Approach

We can find another method by taking a completely different approach. Kriging gives us predictions that are linear in the observations. Suppose we predict X_j. Then the coefficients of the observations are $\Sigma_{jo}\Sigma_{oo}^{-1}$. We can use the property to fit the coefficients directly.

First, though, we need to understand how the regression coefficients relate to the similarities. We expect that if X_i is similar to X_j, that is, if s_{ij} is large, then the coefficient of X_i is large when predicting X_j. That, however, is not true. The coefficients are not functions of similarity. Instead, the covariance matrix Σ is a function of similarity. The inverse in Σ_{oo}^{-1} mixes up the covariances, so the coefficents are not a function of the similarities. But since Σ_{jo} is a function of the similarities, the coefficients $\Sigma_{jo}\Sigma_{oo}^{-1}$ will be related to the similarities. In particular, if Σ_{oo} is close to the identity - if the observation nodes are not highly correlated with each other, then the coefficients will be close to a function of the similarities.

We can use this to create a prediction method. Instead of modeling the covariance as a function of the similarities, we model the coefficients. That is, our prediction for X_j is

$$\hat{X}_j = \sum_{i \in B} \beta(s_{ij}) X_i$$

where $\beta(\cdot)$ are the coefficients as a function of similarities.

We can fit this model by regression. We regress each observation on the others, so we model

$$X_{i'} = \sum_{i \in B, i \neq i'} \beta\left(s_{ii'}\right) X_i + \epsilon_{i'}$$

for $i' \in B$. To fit the function β, we expand it in K spline basis functions,

$$\beta(s) = \sum_{i=1}^{K} c_k \psi_k(s).$$

This means that we have to fit the regression

$$\begin{aligned} X_o &= \left(\sum_i \sum_{i \neq i', i' \in B} \sum_k c_k \psi_k(s_{ii'}) X_{i'} e_i\right) + \epsilon \\ &= \sum_k c_k \left(\sum_i \sum_{i \neq i', i' \in B} \psi_k(s_{ii'}) X_{i'} e_i\right) + \epsilon \end{aligned}$$

since ψ_k, $s_{ii'}$ are known, we can fit c_k by regressing X_o on the predictor matrix $\left(\sum_{i \neq i', i' \in B} \psi_k(s_{ii'}) X_{i'}\right)_{i,k}$.

This is fast to fit. Forming the predictors takes $\mathcal{O}(O^2)$ time, and the regression takes $\mathcal{O}\left(O\right)$ time (since the Gram matrix is $K \times K$, and I am taking K bounded here). So fitting $\hat{\beta}$ take $\mathcal{O}(O^2)$ time. Forming the predictions takes $\mathcal{O}(OP)$ time, so the total complexity is $\mathcal{O}(O^2 + OP)$, even faster than the shortcut covariance method.

The regression approach also solves our other problems. It is easy to predict only for the nodes we need predictions for. It can also be updated easily when we get new observations, using regression updating formulas.

The regression method is fast, and it performs quite well in simulations. The main drawback seems to be that the regression method is more sensitive to misspecified similarities. If the similarity is chosen badly, we may need many degrees of freedom to fit the coefficient function, and this can make our results less stable.

3.3 Blending and possible extensions

Perhaps the best thing about the regression approach, however, is its versatility. We can use our full array of regression tricks to build the best model for each dataset. Using special splines, for example, we can force $\beta(s) \geq 0$, or force $\beta(s) = 0$ for s smaller than some threshhold s_0. These modifications could be of interest if we need nonnegative coefficients or if we want to identify high-value predictors.

The flexibility of the regression lets us combine the regression approach with other methods. Here, I combined the regression with the shortcut covariance method to make a more accurate method that is still fast.

Our model is

$$X_o = \sum_k c_k \left(\sum_i \sum_{i \neq i', i' \in B} \psi_k(s_{ii'}) X_{i'} e_i\right) + c_{k+1}\hat{E} + \epsilon$$

where

$$\hat{E}_i = \hat{E}\left(X_i | X_{o,-i}\right)$$

is the vector of leave-one-out shortcut covariance predictions.

If we calculate $\hat{E}$ naively, we slow down the method to $\mathcal{O}\left(O^4 + OP\right)$, but we can approximate $\hat{E}$ more quickly. To predict X_i based on $X_{o,-i}$, we need to calculate the coefficients $\Sigma^o_{i,-i}\left(\Sigma^o_{-i,-i}\right)^{-1}$ for each i. Suppose we fit the covariance curve using all the data, even including the ith case for its own coefficients. This will not change $\hat{f}$ much since it adds $O-1$ empirical correlations to the $(O-1)^2$ that would normally be used.

If we do this, then we can use the Sherman-Morrison-Woodbury formula to save a lot of work. Suppose we want to calculate $\left(\Sigma^o_{-1,-1}\right)^{-1}$. We have

$$\begin{aligned} I &= \Sigma_o^{-1}\Sigma_o \\ &= \begin{pmatrix} \Sigma^{-o}_{11} & \Sigma^{-o}_{1,-1} \\ \Sigma^{-o}_{-1,1} & \Sigma^{-o}_{-1,-1} \end{pmatrix} \begin{pmatrix} \Sigma^{o}_{11} & \Sigma^{o}_{1,-1} \\ \Sigma^{o}_{-1,1} & \Sigma^{o}_{-1,-1} \end{pmatrix} \end{aligned}$$

where for example $\Sigma^{-o}_{-1,-1} = (\Sigma_o)^{-1}_{-1,-1}$. This implies that

$$\Sigma^{-o}_{-1,1}\Sigma^{o}_{1,-1} + \Sigma^{-o}_{-1,-1}\Sigma^{o}_{-1,-1} = I$$

and hence

$$\begin{aligned} \Sigma^o_{-1,-1} &= \left(\Sigma^{-o}_{-1,-1}\right)^{-1}\left(I - \Sigma^{-o}_{-1,1}\Sigma^o_{1,-1}\right) \\ \left(\Sigma^o_{-1,-1}\right)^{-1} &= \left(I - \Sigma^{-o}_{-1,1}\Sigma^o_{1,-1}\right)^{-1}\Sigma^{-o}_{-1,-1}. \end{aligned}$$

We can compute Σ_o^{-1} once and for all - this gives us $\Sigma^{-o}_{-1,-1}$ - and use the Sherman-Morrison-Woodbury formula to quickly invert $I - \Sigma^{-o}_{-1,1}\Sigma^o_{1,-1}$. This means we can form the shortcut covariance predictor in $\mathcal{O}(O^3)$ time. When we predict, we have to form the shortcut predictor again, which takes $\mathcal{O}\left(O^3 + O^2P\right)$ time. This means the combined algorithm takes $\mathcal{O}\left(O^3 + O^2P\right)$ time - the regression time is dominated by the shortcut covariance. But this combined method is still much faster than the original kriging method.

4 Simulation Results

I will now explore the performance of the faster graph prediction methods and compare them to kriging. To keep implementations simple, I stayed in the simple case where $X_j \sim \mathcal{N}(0,1)$ marginally. This makes the kriging method simple, as we saw in Section 2.

4.1 Accuracy Comparison

I used a few different simulation scenarios to test the methods. Since the original kriging method doesn't scale to large problems, I used $N = 100$. I used the following covariance and similarity combinations:

1. $\Sigma_{ij} = \rho^{|i-j|}$, ρ variable, with similarity $|i-j|$
2. $\Sigma_{ij} = \rho^{|i-j|}$, ρ variable, with similarity $\exp\left(-|i-j|\right)$
3. $\Sigma_{ij} = (1-\alpha)\,\rho^{|i-j|} + \alpha W_{ij}$, where W is a certain random correlation matrix, and the similarity is $\exp\left(|i-j|\right)$
4. $\Sigma_{ij} = \rho^{s_{ij}}$, where s_{ij} are similarities made from the UK Universities Web Links dataset used by [Xu et al. 2009] and W is a certain random correlation matrix.

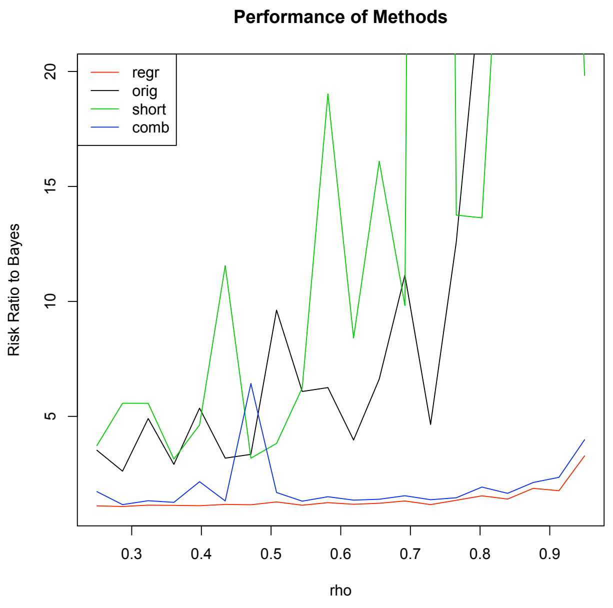

Figure 1: *Average performance relative to Bayes for the methods, where $\Sigma_{ij} = \rho^{|i-j|}$ and $s_{ij} = \exp\left(|i-j|\right)$.*

To measure accuracy, I generated $X \sim \mathcal{N}(0,\Sigma)$ from the given Σ and randomly held out half its entries. I then found

$$\left\|\hat{X}_m - E\left(X_m | X_o\right)\right\|^2$$

to measure the difference between each method's estimate and the Bayes estimate.

The easiest case is the second, $\Sigma_{ij} = \rho^{|i-j|}$ with similarity $\exp\left(-|i-j|\right)$. The covariance is a nice function of the similarities and there is no noise. Figure 1 shows the performance of each estimator relative to the Bayes estimator in this case.

Surprisingly, the original kriging method and the shortcut covariance method do not perform the best in this case, even though they are fitting the true model. They behave erratically, and even at their best do not outperform the regression or combined methods. Both the regression and the combined method perform well, reasonably close to the Bayes risk (usually within a factor of 2), although the combined method becomes unstable on occasion.

It seems like a large part of this is numerical instability - the inverse in equation 1 causes both methods problems, in particular the shortcut method. This rarely carries over to the combined method as well. If we look at the median inefficiency instead, Figure 2, the results are different - the methods perform better, and the gap between them is smaller - note the much smaller scale.

From now on, I'll compare the methods using median inefficiencies - the instability we saw in the previous scenario is seen throughout, and using mean inefficiencies makes it hard to see anything beyond the instability. A real implementation of the covariance-based methods would include measurement error, and this would numerically stabilize the methods.

The regression method has a harder time adapting to different similarities, and this can be seen if we use $s_{ij} = |i-j|$ instead. The results are in Figure 3. Now the original and shortcut methods are the best, and the regression is the worst. The combined method is

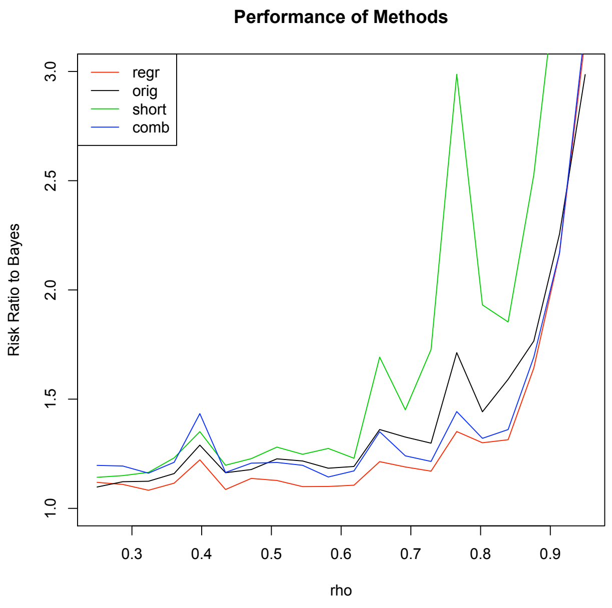

Figure 2: *Median performance relative to Bayes for the methods, where* $\Sigma_{ij} = \rho^{|i-j|}$ *and* $s_{ij} = \exp(|i-j|)$.

close to regression for small ρ, but when ρ rises and regression performs worse, the combined method is able to do about as well as the shortcut and original methods.

We can also assess the methods when the covariance is not actually a function of the similiarity. The third scenario examines such a case - the true similarity is contaminated with noise. Here, I took W to be a random correlation matrix from $2N$ samples of a $\mathcal{N}(0, I)$ random variable. I considered $\rho = 0.25, 0.5, 0.9$ and $\alpha \in [0, 0.5]$. The results are in Figures 4, 5 and 6. We can see that the methods perform pretty similarly, with the original and regression doing slightly better, except for $\rho = 0.9$. There is probably not too much to choose between the methods on this front.

As a final test, I used the UK Universities Web Links dataset to create similarities. This tests the methods on a similarity structure from real data, even if the response is still artificial. I took the link matrix W and set

$$\begin{aligned} A &= W' + W \\ diag(A) &= \max(A) + 1 \\ S &= \log(A+1) \end{aligned}$$

I did this because the link matrix had mostly small entries and a few very large entries, and its diagonal was 0. I then used

$$\Sigma_{ij} = \rho_{ij}^{10(\max(S) - s_{ij})}$$

for the covariance matrix, after making it positive definite. Figure 7 has the results. Surprisingly, all the methods do quite well, though the original and shortcut method are nearly as good as the Bayes estimator (the medians were - both methods showed their usual instability).

To summarize the accuracy results, the regression method is much stabler than the original and shortcut methods, and somewhat stabler than the combined method. All of the methods perform reasonably similarly aside from this instability. The regression method

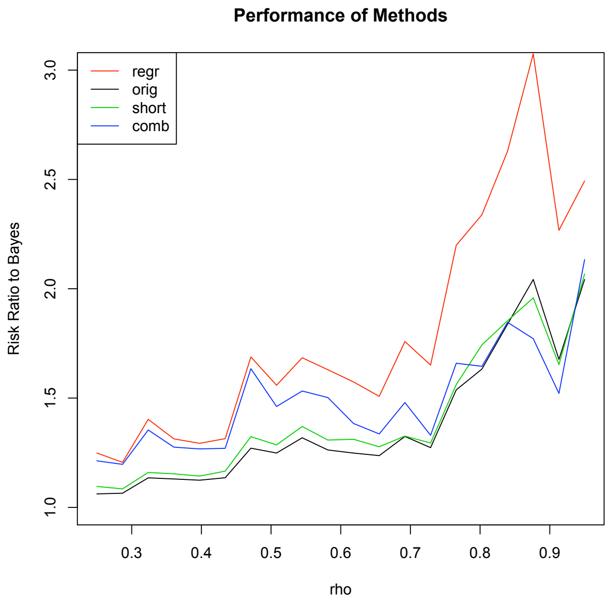

Figure 3: *Median performance relative to Bayes for the methods, where* $\Sigma_{ij} = \rho^{|i-j|}$ *and* $s_{ij} = |i-j|$.

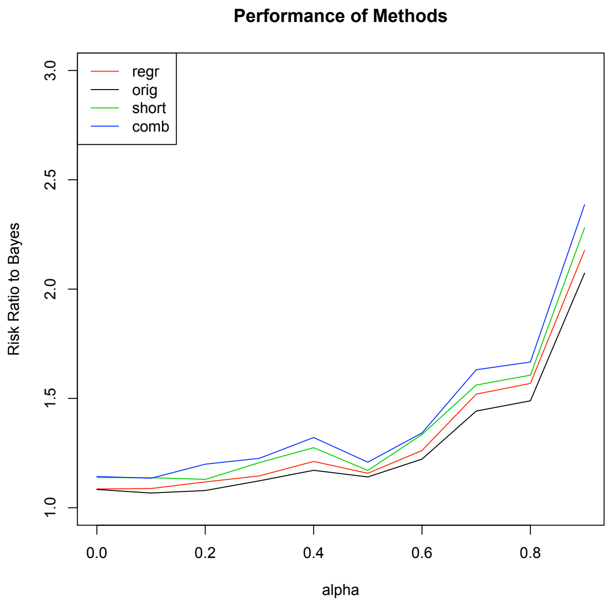

Figure 4: *Median performance relative to Bayes for the methods, where* $\Sigma_{ij} = (1-\alpha)\rho^{|i-j|} + \alpha W$ *and* $s_{ij} = \exp(-|i-j|)$. *Here* $\rho = 0.25$.

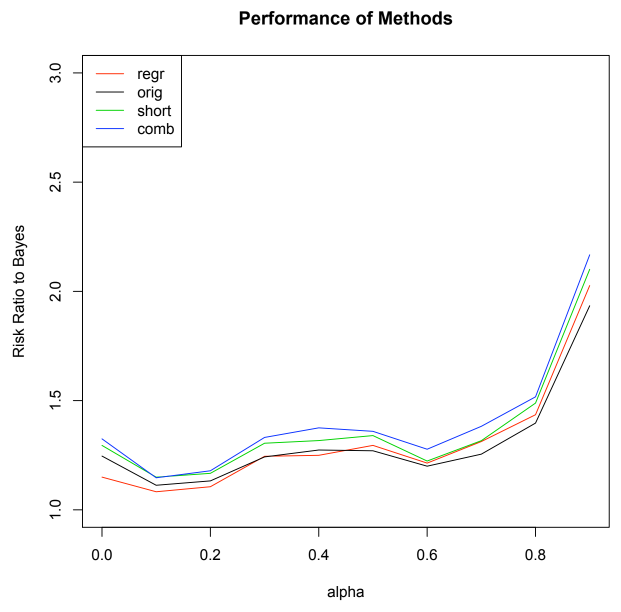

Figure 5: *Median performance relative to Bayes for the methods, where* $\Sigma_{ij} = (1-\alpha)\,\rho^{|i-j|} + \alpha W$ *and* $s_{ij} = \exp(-|i-j|)$. *Here* $\rho = 0.5$.

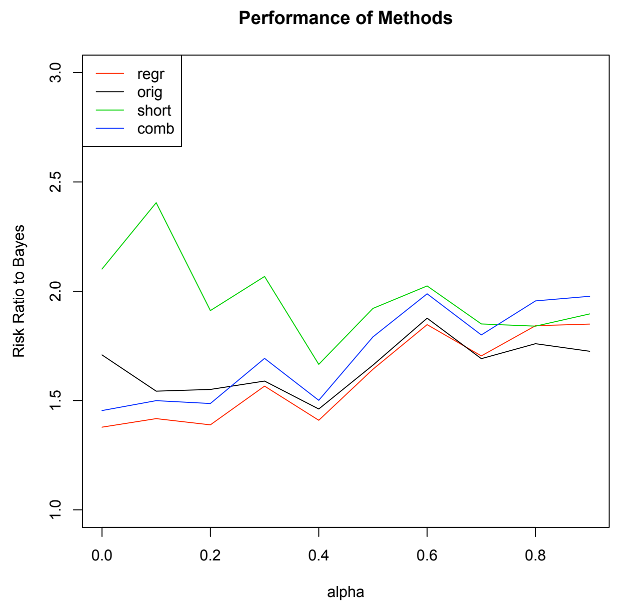

Figure 6: *Median performance relative to Bayes for the methods, where* $\Sigma_{ij} = (1-\alpha)\,\rho^{|i-j|} + \alpha W$ *and* $s_{ij} = \exp(-|i-j|)$. *Here* $\rho = 0.9$.

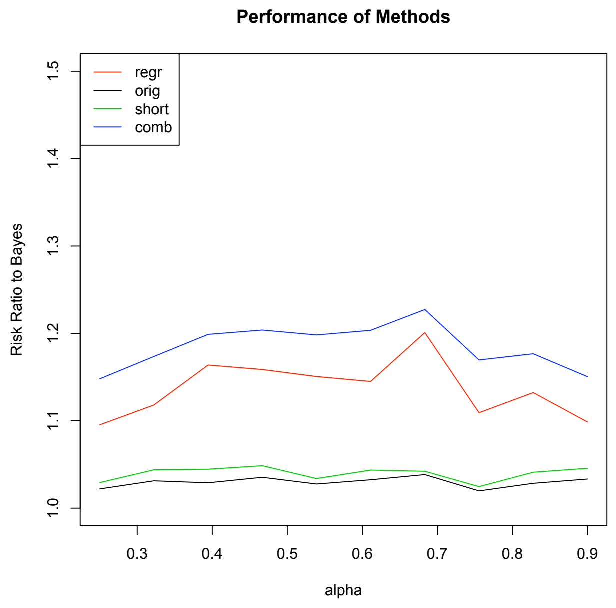

Figure 7: *Results for the UK University Web Links dataset based simulations.*

has a little trouble with misspecified similarities, but adapts a little better when the covariance is not actually a function of the similarities. The combined method is usually in between the regression and shortcut methods, closer to the more accurate one. These simulations suggest that the faster methods in this project do not sacrifice much accuracy, and in many situations can be fully as accurate as the original, much slower, method.

4.2 Speed Comparison

The methods each took about the same amount of time in each scenario. I timed how long they took for each N. I considered a few scenarios. I either held out half the data or fixed 50 observations, and I either predicted all the missing values or predicted a fixed 50. The combinations gave four scenarios, with times in Table 1.

The regression and shortcut methods are the fastest, and the combined method is a bit slower. All three are much faster than the original method. They seem to be more dependent on the number of observations than the number of predictions, which is we would expect. The methods scale very well - their advantage over the original method grows rapidly with N.

There is, however, one exception. The combined method is actually quite a bit slower than the original if we hold out half the data. This is because making the leave-one-out shortcut predictor for each observation can be relatively expensive; if there are not far fewer observations than nodes, the method's other advantages do not make up for this. On large datasets it is probably better to divide the data into folds and leave an entire fold out when predicting the rest. This would avoid repeating the shortcut method for each observation, just once for each fold. This method would take roughly N_{fold} times as long as the regression and the shortcut method combined, which would still be much faster than the original method.

It seems like the speedup from skipping the SVD step is quite large. Decreasing the number of observations helps more than decreasing the number of predictions required, at least at this small scale. This

	Original	Shortcut	Regression	Combined
Hold half, Predict all: $N = 100$	0.03	0.01	0.03	0.05
$N = 575$	0.97	0.31	0.12	1.58
$N = 1050$	4.36	1.16	0.29	8.99
$N = 1575$	12.24	2.89	0.56	27.43
$N = 2000$	28.81	6.24	1.02	64.2
50 Obs, Predict all: $N = 100$	0.03	0.01	0.03	0.05
$N = 575$	0.74	0.03	0.05	0.07
$N = 1050$	3.48	0.05	0.03	0.08
$N = 1575$	9.86	0.07	0.04	0.10
$N = 2000$	22.30	0.09	0.04	0.13
Hold half, Predict 50: $N = 100$	0.03	0.01	0.03	0.05
$N = 575$	1.00	0.25	0.10	1.54
$N = 1050$	4.48	0.90	0.25	8.56
$N = 1575$	12.77	2.15	0.50	25.34
$N = 2000$	25.94	4.03	0.77	58.84
50 Obs, Predict 50: $N = 100$	0.02	0.01	0.02	0.04
$N = 575$	0.86	0.01	0.02	0.05
$N = 1050$	4.40	0.01	0.02	0.05
$N = 1575$	10.53	0.01	0.03	0.05
$N = 2000$	21.94	0.02	0.02	0.14

Table 1: *Time taken (seconds) by the various methods.*

suggests that the methods will scale to situations where observations are sparse and we need many predictions.

5 Conclusion

This project has explored fast and accurate graph prediction methods based on the kriging method of [Xu et al. 2009]. There were three methods - a shortcut kriging method, a regression-type approach, and a combination. They all performed roughly as well as the original method in simulations, and were much faster - the combination method can be slow if there are many observations and we use the leave-one-out approach. The original method and the shortcut method were very unstable, but this is an artifact of having no measurement error, and would not be an issue for a practical implementation. Despite their drawbacks, these methods are for the most part very fast and sacrifice little accuracy. This makes them useful starting points if we are faced with a graph prediction problem that is too big for the original approach to tackle.

6 Note

My original project idea was different - given the kriging, how can we pick the best nodes to observe data on? It turns out that this problem is essentially the same as subset selection for regression. My idea was to use submodular functions to approximately optimize. This is in fact a very good idea, so good that Andreas Krause and other people from Carnegie-Mellon have explored it quite thoroughly [Krause et al. 2008b; Krause et al. 2008a]. Instead of covering the same ground, I decided to work on something new. I talked to Borja about this change when I submitted the milestone, and he approved.

I was originally going to summarize their results here, like I said on the milestone, but since I'm running short on time, I will just refer to their papers.

References

KRAUSE, A., MCMAHAN, H. B., GUESTRIN, C., AND GUPTA, A. 2008. Robust submodular observation selection. *J. Mach. Learn. Res 9*, 2761–2801.

KRAUSE, A., SINGH, A., AND GUESTRIN, C. 2008. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res. 9*, 235–284.

XU, Y., DYER, J. S., AND OWEN, A. B. 2009. Empirical stationary correlations for semi-supervised learning on graphs. *Annals of Applied Statistics*.