
Learning curves for Gaussian process regression on random graphs: effects of graph structure

Matthew J Urry, Peter Sollich
King's College London, Department of Mathematics
London WC2R 2LS, U.K.
{matthew.urry, peter.sollich}@kcl.ac.uk

Abstract

We investigate how well Gaussian process regression can learn functions defined on graphs, using large random graphs as a paradigmatic examples. We focus on learning curves of the Bayes error versus training set size for three types of random graphs, random regular, Poisson and Barabasi-Albert. We begin by developing a theory for the random regular graphs using eigenvalues of the covariance matrix, a prediction found useful in the Euclidean case. We find this prediction is good in two regimes but underestimates the error between them. We then apply this prediction to Poisson and Barabasi-Albert graphs using numerically computed eigenvalues, where we find some unusual results and give explanations for these results.

1 Motivation and Outline

Gaussian processes (GPs) have become a standard part of the machine learning toolbox [1]. Learning curves are a convenient way of characterizing their capabilities: they give the generalization error ϵ as a function of the number of training examples n , averaged over all datasets of size n under appropriate assumptions about the process generating the data. We focus here on the case of GP regression, where a real-valued output function $f(x)$ is to be learned. The general behaviour of GP learning curves is then relatively well understood for the scenario where the inputs x come from a continuous space, typically \mathbb{R}^n [2, 3, 4, 5, 6, 7, 8, 9, 10]. For large n , the learning curves decay as a power law $\epsilon \propto n^{-\alpha}$ with an exponent $\alpha \leq 1$ that depends on the dimensionality n of the space as well as the smoothness properties of the function $f(x)$ encoded by the covariance function.

There are however many interesting application domains that involve *discrete* input spaces, where x could be a string, an amino acid sequence (with $f(x)$ some measure of secondary structure or biological function), a research paper (with $f(x)$ related to impact), a web page (with $f(x)$ giving a score used to rank pages), etc. In many such situations, similarity between different inputs – which will govern our prior beliefs about how closely related the corresponding function values are – can be represented by edges in a *graph*. One would then like to know how well GP regression can work in such problem domains; see also [11] for a related online regression algorithm. We study this problem here theoretically by focussing on the paradigmatic example of random graphs.

We focus initially on random regular graph ensembles and then extend this to some preliminary work with Poisson and Barabasi-Albert random graph ensembles. We begin in Sec. 2 with a brief overview of kernels on graphs. We then look at learning curves for the random walk kernel, and use a simple approximation based on the graph eigenvalues using (for the regular graph case) only the known spectrum of a large tree as input. This is shown to work well qualitatively, and predicts accurately the asymptotics for large numbers of training examples. We subsequently apply the simple approximation to Barabasi-Albert and Poisson ensembles using numerically calculated graph

eigenvalues. We find some unusual results and discuss why these occur. Finally in Sec. 4 we summarise the results and discuss further open questions.

2 Kernels on graphs

We assume that we are trying to learn a function defined on the vertices (nodes) of a graph. Vertices are labelled by $i = 1 \dots V$, instead of the generic input label x we used in the introduction, and the associated function values are denoted $f_i \in \mathbb{R}$. By taking the prior $P(\mathbf{f})$ over these functions $\mathbf{f} = (f_1, \dots, f_V)$ as a (zero mean) Gaussian process we are saying that $P(\mathbf{f}) \propto \exp(-\frac{1}{2} \mathbf{f}^T \mathbf{C}^{-1} \mathbf{f})$. The covariance function or kernel \mathbf{C} is then, in our graph setting, just a positive definite $V \times V$ matrix.

The graph structure is characterized by a $V \times V$ adjacency matrix, with $A_{ij} = 1$ if nodes i and j are connected by an edge, and 0 otherwise. All links are assumed to be undirected, so that $A_{ij} = A_{ji}$, and there are no self-loops ($A_{ii} = 0$). The degree of each node is then defined as $d_i = \sum_{j=1}^V A_{ij}$.

The covariance kernels we discuss in this paper are the natural generalizations of the squared-exponential kernel in Euclidean space [12]. They can be expressed in terms of the normalized graph Laplacian, defined as $\mathbf{L} = \mathbf{1} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$, where \mathbf{D} is a diagonal matrix with entries d_1, \dots, d_V and $\mathbf{1}$ is the $V \times V$ identity matrix. An advantage of \mathbf{L} over the unnormalized Laplacian $\mathbf{D} - \mathbf{A}$, which was used in the earlier paper [13], is that the eigenvalues of \mathbf{L} (again a $V \times V$ matrix) lie in the interval $[0, 2]$ (see e.g. [14]).

From the graph Laplacian, the covariance kernels we consider here are constructed as follows. The p -step random walk kernel is (for $a \geq 2$)

$$\mathbf{C} \propto (1 - a^{-1} \mathbf{L})^p = \left[(1 - a^{-1}) \mathbf{1} + a^{-1} \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \right]^p \quad (1)$$

while the diffusion kernel is given by

$$\mathbf{C} \propto \exp(-\frac{1}{2} \sigma^2 \mathbf{L}) \propto \exp\left(\frac{1}{2} \sigma^2 \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}\right) \quad (2)$$

We will always normalize these so that $(1/V) \sum_i C_{ii} = 1$, which corresponds to setting the average (over vertices) prior variance of the function to be learned to unity.

These covariance kernels are linked to random walks: they are essentially (see for example [15]) just random walk transition matrices, averaged over a distribution of the number of steps taken, s . For the random walk kernel this distribution is $s \sim \text{Binomial}(p, 1/a)$, while for the diffusion kernel it is $s \sim \text{Poisson}(\sigma^2/2)$. This shows in particular that the diffusion kernel is a limiting case of the random walk kernel, where $p, a \rightarrow \infty$ with $p/a = \sigma^2/2$ kept constant. As p/a or σ^2 become large, it is clear that both kernels will approach the limit where they are essentially constant across the graph, corresponding to maximal correlation between function values on different vertices. How this fully correlated limit is approached is, however, not trivial; see [15] for more details.

3 Learning curves

We focus on the analysis of learning curves for GP regression on random graphs. We assume that the target function \mathbf{f}^* is drawn from a GP prior with a p -step random walk covariance kernel \mathbf{C} . Training examples are input-output pairs $(i_\mu, f_{i_\mu}^* + \xi_\mu)$ where ξ_μ is i.i.d. Gaussian noise of variance σ^2 ; the distribution of training inputs i_μ is taken to be uniform across vertices. Inference from a data set D of n such examples $\mu = 1, \dots, n$ takes place using the prior defined by \mathbf{C} and a Gaussian likelihood with noise variance σ^2 . We thus assume an inference model that is matched to the data generating process. This is obviously an over-simplification but is appropriate for the present first exploration of learning curves on random graphs. We emphasize that as n is increased we see more and more function values from the *same* graph, which is fixed by the problem domain; the graph does not grow.

The generalization error ϵ is the squared difference between the estimated function \hat{f}_i and the target f_i^* , averaged across the (uniform) input distribution, the posterior distribution of \mathbf{f}^* given D , the

distribution of datasets D , and finally – in our non-Euclidean setting – the random graph ensemble. Given the assumption of a matched inference model, this is just the average Bayes error, or the average posterior variance, which can be expressed explicitly as [1]

$$\epsilon(n) = V^{-1} \sum_i \langle C_{ii} - \mathbf{k}(i)^T \mathbf{K} \mathbf{k}^{-1}(i) \rangle_{D, \text{graphs}} \quad (3)$$

where the average is over data sets and over graphs, \mathbf{K} is an $n \times n$ matrix with elements $K_{\mu\mu'} = C_{i_\mu, i_{\mu'}} + \sigma^2 \delta_{\mu\mu'}$ and $\mathbf{k}(i)$ is a vector with entries $k_\mu(i) = C_{i, i_\mu}$. The resulting learning curve depends, in addition to n , on the graph structure as determined by V and the expected degree sequence, and the kernel and noise level as specified by p , a and σ^2 .

Exact prediction of learning curves by analytical calculation is very difficult due to the complicated way in which the random selection of training inputs enters the matrix \mathbf{K} and vector \mathbf{k} in (3). However, by first expressing these quantities in terms of kernel eigenvalues (see below) and then approximating the average over datasets, one can derive the approximation [3, 6]

$$\epsilon = g\left(\frac{n}{\epsilon + \sigma^2}\right), \quad g(h) = \sum_{\alpha=1}^V (\lambda_\alpha^{-1} + h)^{-1} \quad (4)$$

This equation for ϵ has to be solved self-consistently because ϵ also appears on the r.h.s. In the Euclidean case the resulting predictions approximate the true learning curves quite reliably. The derivation of (4) for inputs on a fixed graph is unchanged from [3], provided the kernel eigenvalues λ_α appearing in the function $g(h)$ are defined appropriately, by the eigenfunction condition $\langle C_{ij} \phi_j \rangle = \lambda \phi_i$; here the average is over the input distribution, i.e. $\langle \dots \rangle = V^{-1} \sum_j \dots$. From the definition (1) of the p -step kernel, we see that then $\lambda_\alpha = \kappa V^{-1} (1 - \lambda_\alpha^L/a)^p$ in terms of the corresponding eigenvalue of the graph Laplacian L . The constant κ has to be chosen to enforce our normalization convention $\sum_\alpha \lambda_\alpha = \langle C_{jj} \rangle = 1$.

To explore the above predictions in more detail, we start with random regular graphs and fix $d = 3$, the degree at each node of the graph, to avoid having too many parameters to vary, although similar results are obtained for larger d .

Fortunately, for large V the spectrum of the Laplacian of a random regular graph can be approximated by that of the corresponding large regular tree, which has spectral density [14]

$$\rho(\lambda^L) = \frac{\sqrt{\frac{4(d-1)}{d^2} - (\lambda^L - 1)^2}}{2\pi d \lambda^L (2 - \lambda^L)} \quad (5)$$

in the range $\lambda^L \in [\lambda_-^L, \lambda_+^L]$, $\lambda_\pm^L = 1 + 2d^{-1}(d-1)^{1/2}$, where the term under the square root is positive. (There are also two isolated eigenvalues $\lambda^L = 0, 2$ but these have weight $1/V$ each and so can be ignored for large V). Rewriting (4) as $\epsilon = V^{-1} \sum_\alpha [(V\lambda_\alpha)^{-1} + (n/V)(\epsilon + \sigma^2)^{-1}]^{-1}$ and then replacing the average over kernel eigenvalues by an integral over the spectral density leads to the following prediction for the learning curve:

$$\epsilon = \int d\lambda^L \rho(\lambda^L) [\kappa^{-1} (1 - \lambda^L/a)^{-p} + \nu/(\epsilon + \sigma^2)]^{-1} \quad (6)$$

with κ determined from $\kappa \int d\lambda^L \rho(\lambda^L) (1 - \lambda^L/a)^p = 1$. A general consequence of the form of this result is that the learning curve depends on n and V only through the ratio $\nu = n/V$, i.e. the number of training examples per vertex. The approximation (6) also predicts that the learning curve will have two regimes, one for small ν where $\epsilon \gg \sigma^2$ and the generalization error will be essentially independent of σ^2 ; and another for large ν where $\epsilon \ll \sigma^2$ so that ϵ can be neglected on the r.h.s. and one has a fully explicit expression for ϵ .

We compare the above prediction in Fig. 1(left) to the results of numerical simulations of the learning curves, averaged over datasets and random regular graphs. The two regimes predicted by the approximation are clearly visible; the approximation works well inside each regime but less well in the crossover between the two. One striking observation is that the approximation seems to predict the asymptotic large- n behaviour exactly; this is not what is seen in the Euclidean case, where generally only the power-law of the n -dependence but not its prefactor come out accurately. To see why, we exploit that for large n (where $\epsilon \ll \sigma^2$) the approximation (3) effectively neglects fluctuations

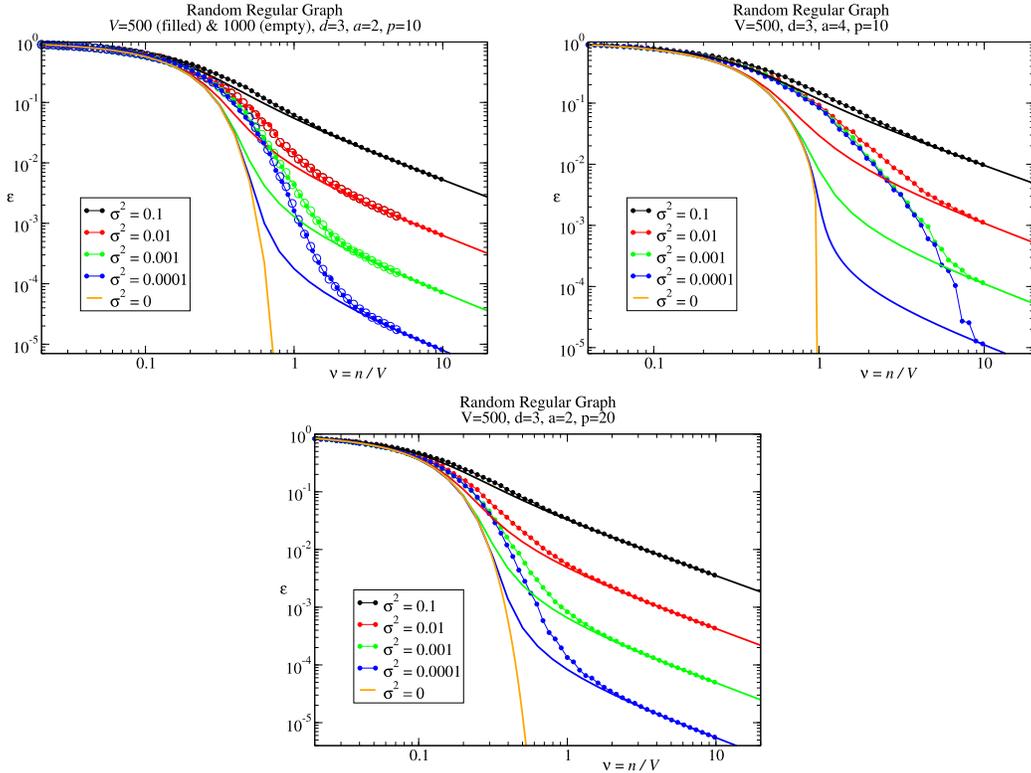


Figure 1: (Left) Learning curves for GP regression on random regular graphs with degree $d = 3$, $V = 500$ (small filled circles) and $V = 1000$ (empty circles) vertices. Plotting generalization error versus $\nu = n/V$ superimposes the results for both values of V , as expected from the approximation (6). The lines are the quantitative predictions of this approximation. Noise level as shown, kernel parameters $a = 2$, $p = 10$. (Right) As on the left but with $V = 500$ only and for larger $a = 4$. (Bottom) Same $a = 2$ as top left, but now with larger $p = 20$.

in the training input “density” of a randomly drawn set of training inputs [3, 6]. This is justified in the graph case for large $\nu = n/V$, because the number of training inputs each vertex receives, $\text{Binomial}(N, 1/V)$, has negligible relative fluctuations away from its mean ν . In the Euclidean case there is no similar result, because all training inputs are different with probability one even for large n .

Fig. 1(right) illustrates that for larger a the difference in the crossover region between the true (numerically simulated) learning curves and our approximation becomes larger. This is because the average number of steps p/a of the random walk kernel then decreases: we get closer to the limit of uncorrelated function values ($a \rightarrow \infty$, $C_{ij} = \delta_{ij}$). In that limit and for low σ^2 and large V the true learning curve is $\epsilon = \exp(-\nu)$, reflecting the probability of a training input set not containing a particular vertex, while the approximation can be shown to predict $\epsilon = \max\{1 - \nu, 0\}$, i.e. a decay of the error to zero at $\nu = 1$. Plotting these two curves (not displayed here) indeed shows the same “shape” of disagreement as in Fig. 1(right), with the approximation underestimating the true generalization error.

Increasing p has the effect of making the kernel longer ranged, giving an effect opposite to that of increasing a . In line with this, larger values of p improve the accuracy of the approximation (6): see Fig. 1(bottom).

One may ask about the shape of the learning curves for large number of training examples (per vertex) ν . The roughly straight lines on the right of the log-log plots discussed so far suggest that $\epsilon \propto 1/\nu$ in this regime. This is correct in the mathematical limit $\nu \rightarrow \infty$ because the graph kernel has a nonzero minimal eigenvalue $\lambda_- = \kappa V^{-1}(1 - \lambda_+^L/a)^p$: for $\nu \gg \sigma^2/(V\lambda_-)$, the square bracket

in (6) can then be approximated by $\nu/(\epsilon + \sigma^2)$ and one gets (because also $\epsilon \ll \sigma^2$ in the asymptotic regime) $\epsilon \approx \sigma^2/\nu$.

However, once p becomes reasonably large, $V\lambda_-$ can be shown [15] to be extremely (exponentially in p) small; for the parameter values in Fig. 1(bottom) it is around 4×10^{-30} . The “terminal” asymptotic regime $\epsilon \approx \sigma^2/\nu$ is then essentially unreachable. A more detailed analysis of (6) for large p and large (but not exponentially large) ν [15] yields $\epsilon \propto (c\sigma^2/\nu) \ln^{3/2}(\nu/(c\sigma^2))$. This shows that there are logarithmic corrections to the naive σ^2/ν scaling that would apply in the true terminal regime. More intriguing is the scaling of the coefficient $c \propto p^{-3/2}$ with p , which implies that to reach a specified (low) generalization error one needs a number of training examples per vertex of order $\nu \propto c\sigma^2 \propto p^{-3/2}\sigma^2$. Of course, at any fixed V , loops eventually become important as p increases; the consequences for the learning curves are discussed in [15].

We next investigate how well the Gaussian processes learn with Poisson random graphs. The Poisson ensemble of random graphs consists of all graphs on a set of V vertices in which each edge is chosen independently to be present with some probability q ; see [16] and [17] for in depth discussions. The average number of edges connected to a given node is then $c = q(V - 1)$. We use $c = 3$ so that the expected degree at any one node is 3, to tie in with the parameters used above for random regular graphs. Again results are similar for other values of $c > 1$. Note that for this value of c a Poisson graph of 500 nodes will have a giant component consisting of approximately 470 nodes and the remaining nodes will be in disconnected tree components [17], of which most are single node components [16]. Learning on a Poisson graph with disconnected components can then be seen as learning on each component separately with the number of samples drawn from $\text{Binomial}(n, V_c/V)$, where V_c is the number of nodes in the disconnected components.

Surprisingly we find that learning – shown in Fig. 2(top left) – on Poisson graphs is significantly slower than on random regular graphs. This is not an immediately obvious result since initially for $c = 3$ most learning takes place on the giant component. The giant component is treelike and loops can be shown on average to be of length $O(\log V)$ [18] so learning curves should be similar to the random regular case. We can see in Fig. 2 (bottom left) that the average covariance of two nodes drops significantly faster than that of the random regular graph as the graph distance (shortest path) between these two nodes increases. This result is even more striking when we superimpose the learning curves for random regular, Poisson and the giant component of the Poisson graph on top of each other (Fig. 2 (bottom right)). The small number of disconnected components greatly affect the rate at which the Gaussian process learns the *entire* graph.

This unlikely result can be explained by the normalization of the covariance kernel. The single disconnected components have a large variance since the random walk defining the kernel can never escape from such a component. The disconnected components contribute large values to $1/V \sum_i C_{ii}$ meaning the smaller variances and covariances on the giant component become dwarfed by these few large variances. Eliminating the disconnected components and only learning on the giant component then gives back more conventional behaviour, with learning similar but not as fast as on random regular graphs (see Fig. 2 (bottom right)). The same effect appears in the distance-dependence of the kernel (Fig. 2 (bottom left)): when using the entire Poisson graph, this is dominated by the short-ranged contributions from the disconnected components, while without them one finds a shape comparable to the one for random regular graphs.

We show in Fig. 2 also the theoretical predictions for the learning curves, using numerically calculated graph eigenvalues. The presence of the disconnected components is seen to change the predictions only weakly compared to the random regular graph case. This is surprising since disconnected components contribute an extra 0 eigenvalue to the normalized Laplacian [14] and in turn, using the method discussed in Sec. 3 to convert to kernel eigenvalues, large eigenvalues to the kernel. (One might guess that the fraction of eigenvalues in this category remains small enough not to make much difference to the theory). The fact the theory fails to account sufficiently for the presence of the disconnected contributions causes the large deviations from the true learning curves.

We finally point out that learning on the Poisson ensemble of random graphs appears more “noisy” than before once the error on the GP becomes smaller than about 10^{-2} . This again is a result of the disconnected components on the graph. Since the latter only consist of a few nodes each, learning on these nodes is essentially over apart from noise once a single example has been seen. The ensuing reduction in overall error is of order 10^{-2} and so becomes appreciable when the overall

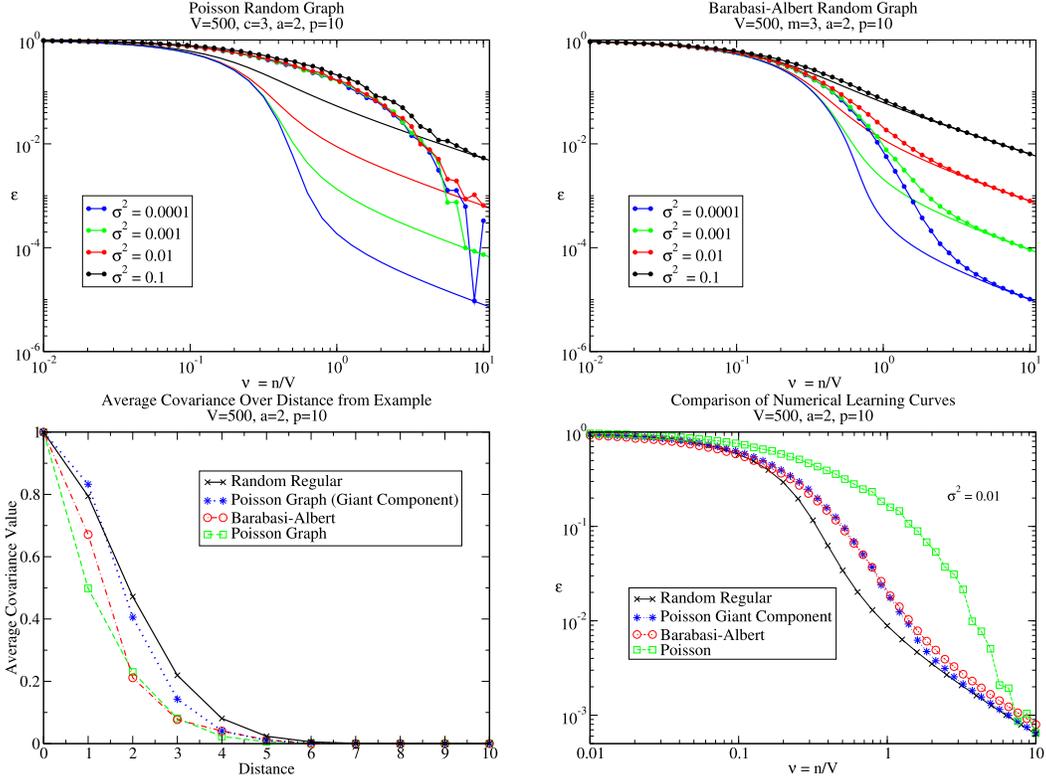


Figure 2: TOP: (Left) Learning curves for GP regression on Poisson graphs with $c = 3$ and $V = 500$. The lines are the learning curve predictions using numerically found eigenvalues of the kernel. Noise level as shown, kernel parameters $a = 2$, $p = 10$. (Right) As on the left but for Barabasi-Albert graphs with $m = 3$. BOTTOM: (Left) Average covariance value for each shell of random regular, Poisson, Poisson giant component and Barabasi-Albert graphs. (Right): Comparison of numerical learning curves for Random Regular, Poisson, Poisson giant component and Barabasi-Albert graphs. $a = 2$, $p = 10$

generalization error reaches this order of magnitude. Random sampling over data sets then causes large fluctuations depending on how many disconnected components have been learned. These are visible in our data because due to computational constraints we sample only a few sets. Because we sample anew for each different value of n , this results in the jumps observed in Fig. 2 (top left) between consecutive sample sizes.

We next consider scale-free power-law distribution random graphs, focusing on the Barabasi-Albert ensemble: for an overview see [18][17]. This ensemble consists of graphs created using the preferential attachment scheme. The graphs are constructed by incrementally adding nodes and connecting m edges between the newly added node and the already existing graph. The new node “prefers” to add edges to nodes in the graph that have high degrees already. Constructing graphs in this way leads to a power-law distribution in the degree sequence.

Learning on this ensemble can be seen in Fig. 2 (top right) it should be different from the previous ensembles since the treelike approximation is no longer valid. We will frequently have loops of size 3 in these graphs. As shown in Fig. 2 (bottom left) the covariance function is far shorter ranged than the random regular and the giant component of the Poisson ensemble. One way to explain this is to think of the graph arranged into shells so that a node belongs to the shell numbered by its distance from the node to be learnt. The presence of loops then means that in the random walk kernel, random walk steps can be taken while remaining in the same shell, i.e. without increasing the distance from the starting node.

Naively, looking at the kernel shapes in Fig. 2 (bottom left), one would expect Barabasi-Albert graphs to exhibit learning behaviour similar to Poisson graphs and significantly slower than for the random regular case. However, as can be seen in Fig. 2 (bottom right), the learning curves are in fact similar to those for the random regular case. The reason is that Barabasi-Albert graphs typically have a very small diameter [19], of the order of 3, and so inferences from a single node can still be made across most of the graph even though the kernel is shorter ranged. The learning curve prediction in this case is seen to show similar accuracy to that for the random regular case and exhibits the two regimes as seen before; note that because numerically calculated graph eigenvalues are used, loops are implicitly accounted for in the theory.

4 Summary and Outlook

We have analysed GP regression for functions defined on graphs, focussing on the learning curves. We compared simulation results with a simple theoretical prediction that uses only the eigenvalue of the covariance kernel (or equivalently those of the graph Laplacian) as input, and found that for random regular graphs, where every node has the same degree d , this provides qualitatively accurate results. In the asymptotic regime of large training sets, the approximation even becomes exact. The largest deviations between the predictions and the true learning curves occur in the crossover region where the generalization error becomes of the order of the noise variance.

To establish how generic the results for random regular graphs are, we then studied similarly the case of Poisson random graphs. Here nodes do not have fixed degree; instead each edge is “switched on” with some probability. Such graphs then typically contain a number of small disconnected components. We argued that it is the presence of these that slows down GP learning on Poisson graphs, and causes substantial sample-to-sample fluctuations in the learning curves around the crossover region. The learning curve theory using numerically calculated eigenvalues of the covariance function turned out to give predictions similar to the random regular tree case. It therefore does not capture accurately the effect of the disconnected components, and significantly underestimates the generalization error. Eliminating the small disconnected components and focusing only on the giant component gave learning curves similar to those of the random regular case, and correspondingly better agreement with the theory.

Finally we looked at power-law degree distributions, specifically Barabasi-Albert graphs, and found that learning in this case was faster than for entire Poisson graphs, approximately the same as for learning on the giant Poisson component, and slower than for random regular graphs. This is due a balance of two effects: the covariance function against graph distance is shorter-ranged, causing the slowdown compared to random regular graphs; on the other hand, the diameter of Barabasi-Albert graphs is very small, making learning faster than on entire Poisson graphs.

In future work we plan a more in depth study of the non-regular graphs presented in this paper and extend this analysis to graphs from application domains. We would like to improve the random walk kernel so that it can deal more competently with disconnected components. We would also like to improve the learning curve theory so that the crossover regime is more accurately predicted, we believe this will be possible using belief propagation which has already been shown to give accurate approximations for eigenvalue spectra [20]. These tools can then be further extended to study e.g. the effects of model mismatch in GP regression on random graphs, and how these are mitigated by tuning appropriate hyper-parameters.

References

- [1] C E Rasmussen and C K I Williams. Gaussian processes for regression. In D S Touretzky, M C Mozer, and M E Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 514–520, Cambridge, MA, 1996. MIT Press.
- [2] M Opper. Regression with Gaussian processes: Average case performance. In I K Kwok-Yee, M Wong, I King, and Dit-Yun Yeung, editors, *Theoretical Aspects of Neural Computation: A Multidisciplinary Perspective*, pages 17–23. Springer, 1997.
- [3] P Sollich. Learning curves for Gaussian processes. In M S Kearns, S A Solla, and D A Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 344–350, Cambridge, MA, 1999. MIT Press.

- [4] M Opper and F Vivarelli. General bounds on Bayes errors for regression with Gaussian processes. In M Kearns, S A Solla, and D Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 302–308, Cambridge, MA, 1999. MIT Press.
- [5] C K I Williams and F Vivarelli. Upper and lower bounds on the learning curve for Gaussian processes. *Mach. Learn.*, 40(1):77–102, 2000.
- [6] D Malzahn and M Opper. Learning curves for Gaussian processes regression: A framework for good approximations. In T K Leen, T G Dietterich, and V Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 273–279, Cambridge, MA, 2001. MIT Press.
- [7] D Malzahn and M Opper. A variational approach to learning curves. In T G Dietterich, S Becker, and Z Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 463–469, Cambridge, MA, 2002. MIT Press.
- [8] P Sollich and A Halees. Learning curves for Gaussian process regression: approximations and bounds. *Neural Comput.*, 14(6):1393–1428, 2002.
- [9] P Sollich. Gaussian process regression with mismatched models. In T G Dietterich, S Becker, and Z Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 519–526, Cambridge, MA, 2002. MIT Press.
- [10] P Sollich. Can Gaussian process regression be made robust against model mismatch? In *Deterministic and Statistical Methods in Machine Learning*, volume 3635 of *Lecture Notes in Artificial Intelligence*, pages 199–210. 2005.
- [11] M Herbster, M Pontil, and L Wainer. Online learning over graphs. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 305–312, New York, NY, USA, 2005. ACM.
- [12] A J Smola and R Kondor. Kernels and regularization on graphs. In M Warmuth and B Schölkopf, editors, *Proc. Conference on Learning Theory (COLT)*, Lect. Notes Comp. Sci., pages 144–158. Springer, Heidelberg, 2003.
- [13] R I Kondor and J D Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 315–322, San Francisco, CA, USA, 2002. Morgan Kaufmann.
- [14] F R K Chung. *Spectral graph theory*. Number 92 in Regional Conference Series in Mathematics. American Mathematical Society, 1997.
- [15] P Sollich, C Cotti, and M J Urry. Kernels and learning curves for Gaussian process regression on random graphs. In *Neural Information Processing Systems*, 2009.
- [16] B. Bollobas. *Random Graphs*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2nd edition, 2001.
- [17] F. R. K. Chung and L. Lu. *Complex graphs and networks*. Number 107. American Mathematical Society, 2006.
- [18] Reka Albert and Albert-Laszlo Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47, 2002.
- [19] B. Bollobás and O. Riordan. The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34, 2004.
- [20] T Rogers, I Perez Castillo, R Kuehn, and K Takeda. Cavity approach to the spectral density of sparse symmetric random matrices. *Phys. Rev. E*, 78(3):031116, 2008.