Multiresolution Kernel Approximation for Gaussian Process Regression

Authored by:

Risi Kondor
Yi Ding
Jonathan Eskreis-Winkler

Abstract

Gaussian process regression generally does not scale to beyond a few thousands data points without applying some sort of kernel approximation method. Most approximations focus on the high eigenvalue part of the spectrum of the kernel matrix, K, which leads to bad performance when the length scale of the kernel is small. In this paper we introduce Multiresolution Kernel Approximation (MKA), the first true broad bandwidth kernel approximation algorithm. Important points about MKA are that it is memory efficient, and it is a direct method, which means that it also makes it easy to approximate $K{-1}$ and $mathop{textrm{det}}(K)$.

1 Paper Body

Gaussian Process (GP) regression, and its frequentist cousin, kernel ridge regression, are such natural and canonical algorithms that they have been reinvented many times by different communities under different names. In machine learning, GPs are considered one of the standard methods of Bayesian nonparametric inference [1]. Meanwhile, the same model, under the name Kriging or Gaussian Random Fields, is the de facto standard for modeling a range of natural phenomena from geophyics to biology [2]. One of the most appealing features of GPs is that, ultimately, the algorithm reduces to ?just? having to compute the inverse of a kernel matrix, K. Unfortunately, this also turns out to be the algorithm?s Achilles heel, since in the general case, the complexity of inverting a dense n?n matrix scales with O(n3), meaning that when the number of training examples exceeds 104 ? 105 , GP inference becomes problematic on virtually any computer1 . Over the course of the last 15 years, devising approximations to address this problem has become a burgeoning field. The most common approach is to use one of the so-called Nystr?om methods [3], which select a small subset {xi1 , . . . , xim } of the original training data points as ?anchors? and approximate K in the ¿ form K ? K?,I CK?,I , where K?,I is the submatrix

of K consisting of columns {i1 , . . . , im }, and C is a matrix such as the pseudo-inverse of KI,I . Nystr?om methods often work well in practice and have a mature literature offering strong theoretical guarantees. Still, Nystr?om is inherently a global low rank approximation, and, as pointed out in [4], a priori there is no reason to believe that K should be well approximable by a low rank matrix: for example, in the case of the popular Gaussian kernel k(x, x0) = exp(?(x ? x0)2 /(2‘2)), as ‘ decreases and the kernel becomes more and more ?local? the number of significant eigenvalues quickly increases. This observation has motivated alternative types of approximations, including local, hierarchical and distributed ones (see Section 2). In certain contexts involving translation invariant kernels yet other strategies may be applicable [5], but these are beyond the scope of the present paper. In this paper we present a new kernel approximation method, Multiresolution Kernel Approximation (MKA), which is inspired by a combination of ideas from hierarchical matrix decomposition 1 In the limited case of evaluating a GP with a fixed Gram matrix on a single training set, GP inference reduces to solving a linear system in K, which scales better with n, but might be problematic behavior when the condition number of K is large.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

algorithms and multiresolution analysis. Some of the important features of MKA are that (a) it is a broad spectrum algorithm that approximates the entire kernel matrix K, not just its top eigenvectors, and (b) it is a so-called ?direct? method, i.e., it yields explicit approximations to K ?1 and det(K). Notations. We define [n] = {1, 2, . . . , n}. Given a matrix A, and a tuple I = (i1 , . . . , ir), AI,? will denote the submatrix of A formed of rows indexed by i1 , . . . , ir , similarly A ?,J will denote the submatrix formed of columns indexed by j1 , . . . , jp , and AI,J will denote the submatrix at the intersection of rows i1 , . . . , ir and columns j1 , . . . , jp . We extend these notations to the case when I and J are sets in the obvious way. If A is a blocked matrix then JAKi,j will denote its (i, j) block.

2

Local vs. global kernel approximation

Recall that a Gaussian Process (GP) on a space X is a prior over functions f : X ? R defined by a mean function ?(x) = E[f (x)], and covariance function k(x, x0) = Cov(f (x), f (x0)). Using the most elementary model yi = f (xi) + where ? N (0, ? 2) and ? 2 is a noise parameter, given training data {(x1 , y1), . . . , (xn , yn)}, the posterior is also a GP, with mean ?0 (x) = ?(x)+k¿ x (K + ? 2 I)?1 y, where kx = (k(x, x1), . . . , k(x, xn)), y = (y1 , . . . , yn), and covariance 2 ?1 kx . k 0 (x, x0) = k(x, x0) ? k¿ x0 (K + ? I)

(1)

Thus (here and in the following assuming ? = 0 for simplicity), the maximum a posteriori (MAP) estimate of f is 2 ?1 y. (2) fb(x) = k¿ x (K + ? I) Ridge regression, which is the frequentist analog of GP regression, yields the same formula, but regards fb as the solution to a regularized risk minization problem over a Hilbert space H induced by k. We will use ?GP? as the generic term to

refer to both Bayesian GPs and ridge regression. Letting K 0 = (K+? 2 I), virtually all GP approximation approaches focus on trying to approximate the (augmented) kernel matrix K 0 in such a way so as to make inverting it, solving K 0 y = ? or computing det(K 0) easier. For the sake of simplicity in the following we will actually discuss approximating K, since adding the diagonal term usually doesn?t make the problem any more challenging. 2.1

Global low rank methods

As in other kernel methods, intuitively, Ki,j = k(xi , xj) encodes the degree of similarity or closeness between the two points xi and xj as it relates to the degree of correlation/similarity between the value of f at xi and at xj . Given that k is often conceived of as a smooth, slowly varying function, one very natural idea is to take a smaller set {xi1 , . . . , xim } of ?landmark points? or ?pseudo-inputs? and approximate k(x, x0) in terms of the similarity of x to each of the landmarks, the relationship of the landmarks to each other, and the similarity of the landmarks to x0 . Mathematically, k(x, x0) ?

m X m X

k(x, xis) cis ,ij k(xij , x0),

s=1 j=1

which, assuming that {xi1 , . . . , xim } is a subset of the original point set {x1 , . . . , xn }, amounts to ¿ an approximation of the form K ? K?,I C K?,I , with I = {i1 , . . . , im }. The canonical choice for + + C is C = W , where W = KI,I , and W denotes the Moore-Penrose pseudoinverse of W . The resulting approximation ¿ K ? K?,I W + K?,I , (3) is known as the Nystr?om approximation, because it is analogous to the so-called Nystr?om extension used to extrapolate continuous operators from a finite number of quadrature points. Clearly, the choice of I is critical for a good quality approximation. Starting with the pioneering papers [6, 3, 7], over the course of the last 15 years a sequence of different sampling strategies have been developed for obtaining I, several with rigorous approximation bounds [8, 9, 10, 11]. Further variations include the ensemble Nystr?om method [12] and the modified Nystr?om method [13]. Nystr?om methods have the advantage of being relatively simple, and having reliable performance bounds. A fundamental limitation, however, is that the approximation (3) is inherently low rank. As pointed out in [4], there is no reason to believe that kernel matrices in general should be close to low rank. An even more fundamental issue, which is less often discussed in the literature, relates to the 2

specific form of (2). The appearance of K 0?1 in this formula suggests that it is the low eigenvalue eigenvectors of K 0 that should dominate the result of GP regression. On the other hand, multiplying the matrix by kx largely cancels this effect, since kx is effectively a row of a kernel matrix similar to K 0 , and will likely concentrate most weight on the high eigenvalue eigenvectors. Therefore, ultimately, it is not K 0 itself, but the relationship between the eigenvectors of K 0 and the data vector y that determines which part of the spectrum of K 0 the result of GP regression is most sensitive to. Once again, intuition about the kernel helps clarify this point. In a setting where the function that we are regressing is smooth, and correspondingly, the kernel has a large length

scale parameter, it is the global, long range relationships between data points that dominate GP regression, and that can indeed be well approximated by the landmark point method. In terms of the linear algebra, the spectral expansion of K 0 is dominated by a few large eigenvalue eigenvectors, we will call this the ?PCA-like? scenario. In contrast, in situations where f varies more rapidly, a shorter lengthscale kernel is called for, local relationships between nearby points become more important, which the landmark point method is less well suited to capture. We call this the ?k?nearest neighbor type? scenario. In reality, most non-trivial GP regression problems fall somewhere in between the above two extremes. In high dimensions data points tend to be all almost equally far from each other anyway, limiting the applicability of simple geometric interpretations. Nonetheless, the two scenarios are an illustration of the general point that one of the key challenges in large scale machine learning is integrating information from both local and global scales. 2.2

Local and hierarchical low rank methods

Realizing the limitations of the low rank approach, local kernel approximation methods have also started appearing in the literature. Broadly, these algorithms: (1) first cluster the rows/columns of K with some appropriate fast clustering method, e.g., METIS [14] or GRACLUS [15] and block K accordingly; (2) compute a low rank, but relatively high accuracy, approximation JKKi,j ? Ui ?i Ui¿ to each diagonal block of K; (3) use the {Ui } bases to compute possibly coarser approximations to the JKKi,j off diagonal blocks. This idea appears in its purest form in [16], and is refined in [4] in a way that avoids having to form all rows/columns of the off-diagonal blocks in the first place. Recently, [17] proposed a related approach, where all the blocks in a given row share the same row basis but have different column bases. A major advantage of local approaches is that they are inherently parallelizable. The clustering itself, however, is a delicate, and sometimes not very robust component of these methods. In fact, divide-and-conquer type algorithms such as [18] and [19] can also be included in the same category, even though in these cases the blocking is usually random. A natural extension of the blocking idea would be to apply the divide-and-conquer approach recursively, at multiple different scales. Geometrically, this is similar to recent multiresolution data analysis approaches such as [20]. In fact, hierarchical matrix approximations, including HODLR matrices, H?matrices [21], H2 ?matrices [22] and HSS matrices [23] are very popular in the numerical analysis literature. While the exact details vary, each of these methods imposes a specific type of block structure on the matrix and forces the off-diagonal blocks to be low rank (Figure 1 in the Supplement). Intutitively, nearby clusters interact in a richer way, but as we move farther away, data can be aggregated more and more coarsely, just as in the fast multipole method [24]. We know of only two applications of the hierarchical matrix methodology to kernel approximation: B?orm and Garcke?s H2 matrix approach [25] and O?Neil et al.?s HODLR method [26]. The advantage of H2 matrices is their more intricate structure, allowing relatively tight interactions between neighboring clusters even when the two clusters are not siblings in the tree (e.g. blocks 8 and 9 in Figure 1c in the Supplement). However, the H2 format does not directly help with inverting K or computing

its determinant: it is merely a memory-efficent way of storing K and performing matrix/vector multiplies inside an iterative method. HODLR matrices have a simpler structure, but admit a factorization that makes it possible to directly compute both the inverse and the determinant of the approximated matrix in just O(n log n) time. The reason that hierarhical matrix approximations have not become more popular in machine learning so far is that in the case of high dimensional, unstructured data, finding the way to organize {x1 , . . . , xn } into a single hierarchy is much more challenging than in the setting of regularly spaced points in R2 or R3 , where these methods originate: 1. Hierarchical matrices require making hard assignments of data points to clusters, since the block structure at each level corresponds to partitioning the rows/columns of the original matrix. 2. The hierarchy must form a single tree, which 3

puts deep divisions between clusters whose closest common ancestor is high up in the tree. 3. Finding the hierarchy in the first place is by no means trivial. Most works use a top-down strategy which defeats the inherent parallelism of the matrix structure, and the actual algorithm used (kd-trees) is known to be problematic in high dimensions [27].

3

Multiresolution Kernel Approximation

Our goal in this paper is to develop a data adapted multiscale kernel matrix approximation method, Multiresolution Kernel Approximation (MKA), that reflects the ?distant clusters only interact in a low rank fashion? insight of the fast multipole method, but is considerably more flexible than existing hierarchical matrix decompositions. The basic building blocks of MKA are local factorizations of a specific form, which we call core-diagonal compression. Definition 1 We say that a matrix H is c?core-diagonal if Hi,j = 0 unless either i, j ? c or i = j. Definition 2 A c?core-diagonal compression of a symmetric matrix A ? Rm?m is an approxima

tion of the form ¿ A?Q HQ= , (4) where Q is orthogonal and H is c?core-diagonal. Core-diagonal compression is to be contrasted with rank c sketching, where H would just have the c ? c block, without the rest of the diagonal. From our multiresolution inspired point of view, however, the purpose of (4) is not just to sketch A, but to also to split Rm into the direct sum of two subspaces: (a) the ?detail space?, spanned by the last n?c rows of Q, responsible for capturing purely local interactions in A and (b) the ?scaling space?, spanned by the first c rows, capturing the overall structure of A and its relationship to other diagonal blocks. Hierarchical matrix methods apply low rank decompositions to many blocks of K in parallel, at different scales. MKA works similarly, by applying core-diagonal compressions. Specifically, the algorithm proceeds by taking K through a sequence of transformations K = K0 7? K1 7? . . . 7? Ks , called stages. In the first stage 1. Similary to other local methods, MKA first uses a fast clustering method to cluster the rows/columns of K0 into clusters C11 , . . . , Cp11 . Using the corresponding permutation matrix C1 (which maps the elements of the first cluster to (1, 2, . . . —C11 —), the elements of the second cluster to (—C11 — + 1, . . . , —C11 — + —C21 —), and so on) we form a blocked matrix K0 = C1 K0 C1¿ , where JK0 Ki,j = KCi1 ,Cj1 . 2. Each

diagonal block of K0 is independently core-diagonally compressed as in (4) to yield Hi1 = Q1i JK0 Ki,i (Q1i)¿ CD(c1)

(5)

i

3.

where CD(c1i) in the index stands for truncation to c1i ?core-diagonal The Q1i local rotations are assembled into a single large orthogonal ¿ applied to the full matrix to give H1 = Q1 K0 Q1 .

form. L 1 matrix Q1 = i Qi and

4. The rows/columns of H1 are rearranged by applying a permutation P1 that maps the core part of each block to one of the first c1 := c11 + . . . c1p1 coordinates, and the diagonal part to the rest, giving H1pre = P1 H1 P1¿ . 5. Finally, H1pre is truncated into the core-diagonal form H1 = K1 ? D1 , where K1 ? Rc1 ?c1 is dense, while D1 is diagonal. Effectively, K1 is a compressed version of K0 , while D1 is formed by concatenating the diagonal parts of each of the Hi1 matrices. Together, this gives a global core-diagonal compression K0 ? C1¿ Q1 ¿ P1¿ (K1 ? D1) P1 Q1 C1 — {z } — {z } Q1

Q¿

1 of the entire original matrix K0 . The second and further stages of MKA consist of applying the above five steps to K1 , K2 , ... , Ks?1 ? which has a telescoping form in turn, so ultimately the algorithm yields a kernel approximation K

? ? Q¿(Q¿(. . . Q¿(Ks ? Ds)Qs . . . ? D2)Q2 ? D1)Q1 K 1 2 s The pseudocode of the full algorithm is in the Supplementary Material. 4

(6)

MKA is really a meta-algorithm, in the sense that it can be used in conjunction with different corediagonal compressors. The main requirements on the compressor are that (a) the core of H should capture the dominant part of A, in particular the subspace that most strongly interacts with other blocks, (b) the first c rows of Q should be as sparse as possible. We consider two alternatives. Augmented Sparse PCA (SPCA). Sparse PCA algorithms explicitly set out to find a set of vectors {v 1 , . . . , v c } so as to maximize kV ¿AV kFrob , where V = [v 1 , . . . , v c], while constraining each vector to be as sparse as possible [28]. While not all SPCAs guarantee orthogonality, this can be enforced a posteriori via e.g., QR factorization, yielding Qsc , the top c rows of Q in (4). Letting U be a basis for the complementary subspace, the optimal choice for the bottom m ? c rows in terms ? where of minimizing Frobenius norm error of the compression is Qwlet = U O, ? = argmax k diag(O¿U ¿A U O)k, O O ¿O=I

the solution to which is of course given by the eigenvectors of U ¿AU . The main drawback of the SPCA approach is its computational cost: depending on the algorithm, the complexity of SPCA scales with m3 or worse [29, 30]. Multiresolution Matrix Factorization (MMF) MMF is a recently introduced matrix factorization algorithm motivated by similar multiresolution ideas as the present work, but applied at the level of individual matrix entries rather than at the

level of matrix blocks [31]. Specifically, MMF yields a factorization of the form ¿ H qL . . . q1 , A ? q1¿ . . . qL — {z } — {z } Q

Q¿

where, in the simplest case, the qi ?s are just Givens rotations. Typically, the number of rotations in MMF is O(m). MMF is efficient to compute, and sparsity is guaranteed by the sparsity of the individual qi ?s and the structure of the algorithm. Hence, MMF has complementary strengths to SPCA: it comes with strong bounds on sparsity and computation time, but the quality of the scaling/wavelet space split that it produces is less well controlled. Remarks. We make a few remarks about MKA. 1. Typically, low rank approximations reduce dimensionality quite aggressively. In contrast, in core-diagonal compression c is often on the order of m/2, leading to ?gentler? and more faithful, kernel approximations. 2. In hierarchical matrix methods, the block structure of the matrix is defined by a single tree, which, as discussed above, is potentially problematic. In contrast, by virtue of reclustering the rows/columns of K‘ before every stage, MKA affords a more flexible factorization. In fact, beyond the first stage, it is not even individual datapoints that MKA clusters, but subspaces defined by the earlier local compressions. 3. While C‘ and P‘ are presented as explicit permutations, they really just correspond to different ways of blocking Ks , which is done implicitly in practice with relatively little overhead. 4. Step 3 of the algorithm is critical, because it extends the core-diagonal splits found in the diagonal blocks of the matrix to the off-diagonal blocks. Essentially the same is done in [4] and [17]. This operation reflects a structural assumption about K, namely that the same bases that pick out the dominant parts of the diagonal blocks (composed of the first c‘i rows of the Q‘i rotations) are also good for compressing the off-diagonal blocks. In the hierarchical matrix literature, for the case of specific kernels sampled in specific ways in low dimensions, it is possible to prove such statements. In our high dimensional and less structured setting, deriving analytical results is much more challenging. 5. MKA is an inherently bottom-up algorithm, including the clustering, thus it is naturally parallelizable and can be implemented in a distributed environment. 6. The hierarchical structure of MKA is similar to that of the parallel version of MMF (pMMF) [32], but the way that the compressions are calculated is different (pMMF tries to minimize an objective that relates to the entire matrix).

4

Complexity and application to GPs

For MKA to be effective for large scale GP regression, it must be possible to compute the factor? must be symmetric positive semi-definite ization fast. In addition, the resulting approximation K (spsd) (MEKA, for example, fails to fulfill this [4]). We say that a matrix approximation algorithm A 7? A? is spsd preserving if A? is spsd whenever A is. It is clear from its form that the Nystr?om approximation is spsd preserving , so is augmented SPCA compression. MMF has different variants, but the core part of H is always derived by conjugating A by rotations, while the diagonal elements are guaranteed to be positive, therefore MMF is spsd preserving as well. 5

Proposition 1 If the individual core-diagonal compressions in MKA are spsd

preserving, then the entire algorithm is spsd perserving. The complexity of MKA depends on the complexity of the local compressions. Next, we assume that to leading order in m this cost is bounded by ccomp m?comp (with ?comp ? 1) and that each row of the Q matrix that is produced is csp ?sparse. We assume that the MKA has s stages, the size of the final Ks ?core matrix? is dcore ? dcore , and that the size of the largest cluster is mmax . We assmue that the maximum number of clusters in any stage is bmax and that the clustering is close to balanced in the sense that that bmax = ?(n/mmax) with a small constant. We ignore the cost of the clustering algorithm, which varies, but usually scales linearly in snbmax . We also ignore the cost of permuting the rows/columns of K‘ , since this is a memory bound operation that can be virtualized away. The following results are to leading order in mmax and are similar to those in [32] for parallel MMF. Proposition 2 With the above notations, the number of operations needed to compute the MKA of ?comp ?1 an n ? n matrix is upper bounded by 2scsp n2 + sccomp mmax n. Assuming bmax ?fold parallelism, ? comp this complexity reduces to 2scsp n2 /bmax + sccomp mmax . The memory cost of MKA is just the cost of storing the various matrices appearing in (6). We only include the number of non-zero reals that need to be stored and not indices, etc.. Proposition 3 The storage complexity of MKA is upper bounded by (scsp + 1)n + d2core . Rather than than the general case, it is more informative to focus on MMF based MKA, which is what we use in our experiments. We consider the simplest case of MMF, referred to as ?greedyJacobi? MMF, in which each of the qi elementary rotations is a Given rotation. An additional parameter of this algorithm is the compression ratio ?, which in our notation is equal to c/n. Some of the special features of this type of core-diagonal compression are: (a) While any given row of the rotation Q produced by the algorithm is not guaranteed to be sparse, Q will be the product of exactly b(1 ? ?)mc Givens rotations. (b) The leading term in the cost is the m3 cost of computing A¿A, but this is a BLAS operation, so it is fast. (c) Once A¿A has been computed, the cost of the rest of the compression scales with m2 . Together, these features result in very fast core-diagonal compressions and a very compact representation of the kernel matrix. Proposition 4 The complexity of computing the MMF-based MKA of an n?n dense matrix is upper bounded by 4sn2 + sm2max n, where s = log(dcore /n)/(log ?). Assuming bmax ?fold parallelism, this is reduced to 4snmmax + m3max . Proposition 5 The storage complexity of MMF-based MKA is upper bounded by (2s + 1)n + d2core . Typically, dcore = O(1). Note that this implies O(n log n) storage complexity, which is similar to Nystr?om approximations with very low rank. Finally, we have the following results that are critical for using MKA in GPs. ? in MMF-based MKA form (6), and a vector z ? Rn Proposition 6 Given an approximate kernel K ? the product Kz can be computed in 4sn + d2core operations. With bmax ?fold parallelism, this is reduced to 4smmax + d2core . ? in (MMF or SPCA-based) MKA form, the MKA Proposition 7 Given an approximate kernel K ? ? for any ? can be computed in O(n + d3core) operations. The complexity of computing form of K ? for any ? in MKA form and the complexity of computing det(K) ? the matrix exponential exp(? K) are also O(n + d3core). 4.1

MKA?GPs and MKA Ridge Regression

The most direct way of applying MKA to speed up GP regression (or ridge regression) is simply using it to approximate the augmented kernel matrix K 0 = (K + ? 2 I) and then inverting this ? 0?1 never needs to be approximation using Proposition 7 (with ? = ?1). Note that the resulting K ? 0?1 y evaluated fully, in matrix form. Instead, in equations such as (2), the matrix-vector product K can be computed in ?matrix-free? form by cascading y through the analog of (6). Assuming that dcore n and mmax is not too large, the serial complexity of each stage of this computation scales with at most n2 , which is the same as the complexity of computing K in the first place. One potential issue with the above approach however is that because MKA involves repeated trun? 0 will be a biased approximation to K, therefore expressions such as cation of the Hjpre matrices, K 6

SOR
Full
FITC
PITC
MEKA
MKA
10
10
10
10
10
10
8
8
8
8
8
8
6
6
6
6
6
6
4
4
2
4
2
4
2
4
2

4
2
2
0
0
0
0
0
0
-2
-2
-2
-2
-2
-2
-4
-4
-4
-4
-4
-4
-6
-6
-8
-6
-8
50
100
150
200
250
300
-6
-8
50
100
150
200
250
300
-6
-8
50
100
150
200

250
300
-6
-8
50
100
150
200
250
300
-8
50
100
150
200
250
300
50
100
150
200
250
300

Figure 1: Snelson?s 1D example: ground truth (black circles); prediction mean (solid line curves); one standard deviation in prediction uncertainty (dashed line curves). Table 1: Regression Results with k to be # pseudo-inputs/dcore : SMSE(MNLP) Method housing rupture wine pageblocks compAct pendigit

k 16 16 32 32 32 64

Full 0.36(?0.32) 0.17(?0.89) 0.59(?0.33) 0.44(?1.10) 0.58(?0.66) 0.15(?0.73)

SOR 0.93(?0.03) 0.94(?0.04) 0.86(?0.07) 0.86(?0.57) 0.88(?0.13) 0.65(?0.19)

FITC 0.91(?0.04) 0.96(?0.04) 0.84(?0.03) 0.81(?0.78) 0.91(?0.08) 0.70(?0.17)

PITC 0.96(?0.02) 0.93(?0.05) 0.87(?0.07) 0.86(?0.72) 0.88(?0.14) 0.71(?0.17)

MEKA 0.85(?0.08) 0.46(?0.18) 0.97(?0.12) 0.96(?0.10) 0.75(?0.21) 0.53(?0.29)

MKA 0.52(?0.32) 0.32(?0.54) 0.70(?0.23) 0.63(?0.85) 0.60(?0.32) 0.30(?0.42)

(2) which mix an approximate K 0 with an exact kx will exhibit some systematic bias. In Nystr?om type methods (specifically, the so-called Subset of Regressors and Deterministic Training of Conditionals (DTC) GP approximations) this problem is addressed by replacing kx with its own Nystr?om ? x = K?,I W + kI ,, where [k?I]j = k(x, xi). Although K ? 0 = K?,I W + K ¿ + ? 2 I approximation, k x x j ?,I ?¿K ? 0?1 can nonetheless be efficiently evaluated by using a is a large matrix, expressions such as k x variant of the Sherman?Morrison?Woodbury identity and the fact that W is low rank (see [33]). ? is not low rank. Assuming that the testing The same approach cannot be applied to MKA because K set {x1 , . . . , xp } is known at training time, however, instead of approximating K or K 0 , we compute the MKA approximation of the joint train/test kernel matrix

Ki,j = k(xi , xj) + ? 2 K K? [K?]i,j = k(xi , x0j) K= where K?¿ Ktest [Ktest]i,j = k(x0i , x0j). Writing K?1 in blocked form ? ?1 = K

A C

B D

,

? ?1 = A ? and taking the Schur complement of D now recovers an alternative approximation K BD?1 C to K ?1 which is consistent with the off-diagonal block K ? leading to our final MKA?GP ? ?1 y, where fb = (fb(x0), . . . , fb(x0))¿ . While conceptually this is somewhat formula fb = K?¿ K p 1 more involved than naively estimating K 0 , assuming p n, the cost of inverting D is negligible, and the overall serial complexity of the algorithm remains (n + p)2 . In certain GP applications, the O(n2) cost of writing down the kernel matrix is already forbidding. The one circumstance under which MKA can get around this problem is when the kernel matrix is a matrix polynomial in a sparse matrix L, which is most notably for diffusion kernels and certain other graph kernels. Specifically in the case of MMF-based MKA, since the computational cost is dominated by computing local ?Gram matrices? A¿A, when L is sparse, and this sparsity is retained from one compression to another, the MKA of sparse matrices can be computed very fast. In the case of graph Laplacians, empirically, the complexity is close to linear in n. By Proposition 7, the diffusion kernel and certain other graph kernels can also be approximated in about O(n log n) time.

5

Experiments

We compare MKA to five other methods: 1. Full: the full GP regression using Cholesky factorization [1]. 2. SOR: the Subset of Regressors method (also equivalent to DTC in mean) [1]. 3. FITC: the Fully Independent Training Conditional approximation, also called Sparse Gaussian Processes using Pseudo-inputs [34]. 4. PITC: the Partially Independent Training Conditional approximation method (also equivalent to PTC in mean) [33]. 5. MEKA: the Memory Efficient Kernel Approximation method [4]. The KISS-GP [35] and other interpolation based methods are not discussed in this paper, because, we believe, they mostly only apply to low dimensional settings. We used custom Matlab implementations [1] for Full, SOR, FITC, and PITC. We used the Matlab codes provided by 7

Full SOR FITC PITC MEKA MKA

0.75

0.8 -0.2

0.7

MNLP

0.7 0.65 0.6

-0.3

-0.4

0.55

Full SOR FITC PITC MEKA MKA

-0.5

0.5 0.45

-0.6
0.4
rupture Full SOR FITC PITC MEKA MKA
0.9
Full SOR FITC PITC MEKA MKA
-0.1
-0.2
-0.3
MNLP
0.8
-0.1
SMSE
0.85
SMSE
rupture
housing
housing 0.9
0.6
0.5
-0.4
-0.5
-0.6
0.4 -0.7
0.3 -0.8
0.2 2
2.5
3
3.5
4
Log2 # pseudo-inputs
4.5
2
2.5
3
3.5
4
4
4.5
4.5
5
5.5
6
6.5
7
Log2 # pseudo-inputs
Log2 # pseudo-inputs

7.5
8
4
4.5
5
5.5
6
6.5
7
7.5
8
Log2 # pseudo-inputs

Figure 2: SMSE and MNLP as a function of the number of pseudo-inputs/dcore on two datasets. In the given range MKA clearly outperforms the other methods in both error measures. the author for MEKA. Our algorithm MKA was implemented in C++ with the Matlab interface. To get an approximately fair comparison, we set dcore in MKA to be the number of pseudo-inputs. The parallel MMF algorithm was used as the compressor due to its computational strength [32]. The Gaussian kernel is used for all experiments with one length scale for all input dimensions. Qualitative results. We show the qualitative behavior of each method on the 1D toy dataset from [34]. We sampled the ground truth from a Gaussian processes with length scale ‘ = 0.5 and number of pseudo-inputs (dcore) is 10. We applied cross-validation to select the parameters for each method to fit the data. Figure 1 shows that MKA fits the data almost as well as the Full GP does. In terms of the other approximate methods, although their fit to the data is smoother, this is to the detriment of capturing the local structure of the underlying data, which verifies MKA?s ability to capture the entire spectrum of the kernel matrix, not just its top eigenvectors. Real data. We tested the efficacy of GP regression on real-world datasets. The data are normalized to mean zero and variance one. We randomly selected 10% of each dataset to be used as a test set. On the other 90% we did five-fold cross validation to learn the length scale and noise parameter for each method and the regression results were averaged over repeating this setting five times. All experiments were ran on a 3.4GHz 8 core machine with 8GB of memory. Two distinct Pn error measures are used to assess performance: (a) standardized mean square error (SMSE), n1 t=1 (? yt ? 2 yt)P /? ??2 , where ? ??2 is the variance of test outputs, and (2) mean negative log probability (MNLP)

n 1 ??2 + log 2? , each of which corresponds to the predictive mean and yt ? yt)2 /? ??2 + log ? t=1 (? n variance in error assessment. From Table 1, we are competitive in both error measures when the number of pseudo-inputs (dcore) is small, which reveals low-rank methods? inability in capturing the local structure of the data. We also illustrate the performance sensitivity by varying the number of pseudo-inputs on selected datasets. In Figure 2, for the interval of pseudo-inputs considered, MKA?s performance is robust to dcore , while low-rank based methods? performance changes rapidly, which shows MKA?s ability to achieve good regression results even with a crucial compression level. The

Supplementary Material gives a more detailed discussion of the datasets and experiments.

6

Conclusions

In this paper we made the case that whether a learning problem is low rank or not depends on the nature of the data rather than just the spectral properties of the kernel matrix K. This is easiest to see in the case of Gaussian Processes, which is the algorithm that we focused on in this paper, but it is also true more generally. Most exisiting sketching algorithms used in GP regression force low rank structure on K, either globally, or at the block level. When the nature of the problem is indeed low rank, this might actually act as an additional regularizer and improve performance. When the data does not have low rank structure, however, low rank approximations will fail. Inspired by recent work on multiresolution factorizations, we proposed a mulitresolution meta-algorithm, MKA, for approximating kernel matrices, which assumes that the interaction between distant clusters is low rank, while avoiding forcing a low rank structure of the data locally, at any scale. Importantly, MKA allows fast direct calculations of the inverse of the kernel matrix and its determinant, which are almost always the computational bottlenecks in GP problems. Acknowledgements This work was completed in part with resources provided by the University of Chicago Research Computing Center. The authors wish to thank Michael Stein for helpful suggestions.

8

2 References

[1] Carl Edward Rasmussen and Christopher K. I. Williams. Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, 2005. [2] Michael L. Stein. Statistical Interpolation of Spatial Data: Some Theory for Kriging. Springer, 1999. [3] Christopher Williams and Matthias Seeger. Using the Nystr?om Method to Speed Up Kernel Machines. In Advances in Neural Information Processing Systems 13, 2001. [4] Si Si, C Hsieh, and Inderjit S Dhillon. Memory Efficient Kernel Approximation. In ICML, 2014. [5] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. NIPS, 2008. [6] Alex J. Smola and Bernhard Sch?okopf. Sparse Greedy Matrix Approximation for Machine Learning. In Proceedings of the 17th International Conference on Machine Learning, ICML, pages 911?918, 2000. [7] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the Nystr?om method. IEEE transactions on pattern analysis and machine intelligence, 26(2):214?25, 2004. [8] P. Drineas and M. W. Mahoney. On the Nystr?om method for approximating a Gram matrix for improved kernel-based learning. Journal of Machine Learning Research, 6:2153?2175, 2005. [9] Rong Jin, Tianbao Yang, Mehrdad Mahdavi, Yu-Feng Li, and Zhi-Hua Zhou. Improved Bounds for the Nystr?om Method With Application to Kernel Clas-

sification. IEEE Trans. Inf. Theory, 2013. [10] Alex Gittens and Michael W Mahoney. Revisiting the Nystr?om method for improved large-scale machine learning. ICML, 28:567?575, 2013. [11] Shiliang Sun, Jing Zhao, and Jiang Zhu. A Review of Nystr?om Methods for Large-Scale Machine Learning. Information Fusion, 26:36?48, 2015. [12] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Ensemble Nystr?om method. In NIPS, 2009. [13] Shusen Wang. Efficient algorithms and error analysis for the modified Nystr?om method. AISTATS, 2014. [14] Amine Abou-Rjeili and George Karypis. Multilevel algorithms for partitioning power-law graphs. In Proceedings of the 20th International Conference on Parallel and Distributed Processing, 2006. [15] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(11):1944?1957, 2007. [16] Berkant Savas, Inderjit Dhillon, et al. Clustered Low-Rank Approximation of Graphs in Information Science Applications. In Proceedings of the SIAM International Conference on Data Mining, 2011. [17] Ruoxi Wang, Yingzhou Li, Michael W Mahoney, and Eric Darve. Structured Block Basis Factorization for Scalable Kernel Matrix Evaluation. arXiv preprint arXiv:1505.00398, 2015. [18] Yingyu Liang, Maria-Florina F Balcan, Vandana Kanchanapally, and David Woodruff. Improved distributed principal component analysis. In NIPS, pages 3113?3121, 2014. [19] Yuchen Zhang, John Duchi, and Martin Wainwright. Divide and conquer kernel ridge regression. Conference on Learning Theory, 30:1?26, 2013. [20] William K Allard, Guangliang Chen, and Mauro Maggioni. Multi-scale geometric methods for data sets II: Geometric multi-resolution analysis. Applied and Computational Harmonic Analysis, 2012. [21] W Hackbusch. A Sparse Matrix Arithmetic Based on H-Matrices. Part I: Introduction to H-Matrices. Computing, 62:89?108, 1999. [22] Wolfgang Hackbusch, Boris Khoromskij, and Stefan a. Sauter. On H2-Matrices. Lectures on applied mathematics, pages 9?29, 2000. [23] S. Chandrasekaran, M. Gu, and W. Lyons. A Fast Adaptive Solver For Hierarchically Semi-separable Representations. Calcolo, 42(3-4):171?185, 2005. [24] L. Greengard and V. Rokhlin. A Fast Algorithm for Particle Simulations. J. Comput. Phys., 1987. [25] Steffen B?orm and Jochen Garcke. Approximating Gaussian Processes with H 2 Matrices. In ECML. 2007. [26] Sivaram Ambikasaran, Sivaram Foreman-Mackey, Leslie Greengard, David W. Hogg, and Michael O?Neil. Fast Direct Methods for Gaussian Processes. arXiv:1403.6015v2, April 2015. [27] Nazneen Rajani, Kate McArdle, and Inderjit S Dhillon. Parallel k-Nearest Neighbor Graph Construction Using Tree-based Data Structures. In 1st High Performance Graph Mining workshop, 2015. [28] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse Principal Component Analysis. Journal of Computational and Graphical Statistics, 15(2):265?286, 2004. [29] Q. Berthet and P. Rigollet. Complexity Theoretic Lower Bounds for Sparse Principal Component Detection. J. Mach. Learn. Res. (COLT), 30, 1046-1066 2013. [30] Volodymyr Kuleshov. Fast algorithms for sparse principal component analysis based on rayleigh quotient iteration. In ICML, pages 1418?1425, 2013. [31] Risi Kondor, Nedelina Teneva, and Vikas Garg. Multiresolution Matrix Factorization. In ICML, 2014. [32] Nedelina Teneva, Pramod K

Murakarta, and Risi Kondor. Multiresolution Matrix Compression. In Proceedings of the 19th International Conference on Aritifical Intelligence and Statistics (AISTATS-16), 2016. [33] Joaquin Qui?nonero Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. Journal of Machine Learning Research, 6:1939?1959, 2005. [34] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. NIPS, 2005. [35] Andrew Gordon Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (KISS-GP). In ICML, Lille, France, 6-11, pages 1775?1784, 2015.

9