Scalable Gaussian Processes on Discrete Domains

Vincent Fortuin\textsuperscript{1}, Gideon Dresdner\textsuperscript{1}, Heiko Strathmann\textsuperscript{2}, and Gunnar Rätsch\textsuperscript{1}

\textsuperscript{1}Institute of Machine Learning, ETH Zürich, Universitätsstrasse 6, 8092 Zürich, Switzerland
\textsuperscript{2}Gatsby Unit, University College London, 25 Howland Street, London W1T 4JG, United Kingdom

Abstract

Kernel methods on discrete domains have shown great promise for many challenging tasks, e.g., on biological sequence data as well as on molecular structures. Scalable kernel methods like support vector machines offer good predictive performances but they often do not provide uncertainty estimates. In contrast, probabilistic kernel methods like Gaussian Processes offer uncertainty estimates in addition to good predictive performance but fall short in terms of scalability. We present the first sparse Gaussian Process approximation framework on discrete input domains. Our framework achieves good predictive performance as well as uncertainty estimates using different discrete optimization techniques. We present competitive results comparing our framework to support vector machine and full Gaussian Process baselines on synthetic data as well as on challenging real-world DNA sequence data.

1 Introduction

Uncertainty quantification is an increasingly important feature of machine learning models. This is particularly crucial in applications such as in biomedicine \cite{8, 11, 15} where prediction errors may have serious repercussions. Consider a wet lab biologist seeking to find a DNA sequence which can be targeted by a drug (for instance, using CRISPR-cas9 \cite{13}). They have reduced the problem to some number of candidate sequences but to further narrow the search requires painstaking experiments. If they had a framework that could incorporate their prior knowledge of DNA sequence similarity as well as the results from previous experiments, they could optimally select the next best experiment to perform thereby saving countless amounts of time and resources. Such a framework would need to perform well under various data sizes as well as provide calibrated uncertainty estimates in order to make an informed decision.

Many similar problems are discrete, involve large data sets, and require well-calibrated uncertainty estimates. In addition to domains like biology \cite{31}, chemistry \cite{14} is another domain where expertise can be used to develop relevant kernels. Kernel methods have shown performances that are competitive with deep learning models in such application domains\cite{23}.

Probabilistic modeling provides a unified framework for prediction and calibrated uncertainty estimates \cite{24}. One class of probabilistic models that have proven to be useful in various regression and classification settings are Gaussian Processes (GPs) \cite{29}. They are data efficient, non-parametric, and have tractable posterior distributions. Moreover, one can use any kind of likelihood for the generating process, e.g. a Poisson likelihood in the case of a count process.

The main challenge of scaling GPs to large data sets lies in the computational complexity of inference which is cubic in the number of observations. Inducing point methods are the main class of approaches for circumventing this limitation \cite{12, 28, 39, 41}. These methods aim to use some $m \ll n$ inducing points to reduce the inference complexity to $O(n m^2)$.

Having reduced the complexity of inference, the remaining challenge is to choose the set of inducing points that best approximates the full model \cite{25}. When the domain is continuous, the locations of the inducing points can be optimized using the gradient of the log marginal likelihood \cite{33}. Unfortunately, this gradient-based optimization scheme is not feasible in discrete domains where the log marginal likelihood is no longer differentiable with respect to the inducing point locations.
In this work, we synthesize different strategies into a new framework for choosing inducing points over discrete domains. These strategies include greedy methods [32, 39], Monte Carlo methods [30] and recent developments in discrete Bayesian optimization [2]. We develop a framework for scalable Gaussian Process inference on discrete domains with comparable performance to full GP inference and different baseline methods, such as support vector machines.

We make the following contributions:

- We develop the first sparse inducing point framework for Gaussian processes on discrete domains.
- We discuss optimization techniques to choose the inducing point locations.
- We validate our model on toy data and several challenging real world data sets.

In the following sections we describe the main components of our framework beginning with sparse GPs, continuing to three inducing point selection methods, and concluding with the spectrum string kernel. Each inducing point method corresponds to a different sparse string GP in our framework. For a high level overview of the framework see Figure 1.

### 2 Sparse Gaussian Process approximations

Consider a supervised learning problem in which the goal is to estimate a latent function \( f : \mathcal{X} \rightarrow \mathbb{R} \) given observed inputs \( \mathbf{x} := (x_1, \ldots, x_n) \) and corresponding outputs \( \mathbf{y} := (y_1, \ldots, y_n) \). For the biologist example in Section 1, \( f \) could map DNA sequences to drug targetability scores. We assume that our observations are corrupted by additive noise \( \eta \), thus \( \mathbf{y} = f(\mathbf{x}) + \eta \) where we have overloaded the notation of the function to be broadcast elementwise. Following a long line of previous work [29], we treat the function \( f \) as an unobserved random variable with a Gaussian Process prior. Specifically, a Gaussian Process prior with a zero mean function and a covariance kernel \( k(\cdot, \cdot) \):

\[
f(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot))
\]

It follows that the prior on the function outputs, \( \mathbf{f} := f(\mathbf{x}) \), is given by \( \mathcal{N}(0, K_{xx}) \) where \( K_{xx} \) denotes the Gram matrix (also known as the kernel matrix) with \((K_{xx})_{ij} = k(x_i, x_j)\).

In general, the predictive distribution cannot be solved in closed form (see Sec. 4.3) but in the...
special case of Gaussian noise, where \( \eta \sim \mathcal{N}(0, \sigma^2) \), the predictive distribution can be computed as

\[
p(f^* | y, x, x^*) = \mathcal{N}(m^*, K^*)
\]

with

\[
m^* = K_{sx} (K_{xx} + \sigma^2 I)^{-1} y
\]

\[
K^* = K_{ss} - K_{sx} (K_{xx} + \sigma^2 I)^{-1} K_{xs}
\]

where the test inputs and outputs are denoted \( x^* \) and \( f^* \) respectively and \( K_{ss} = K^T \) is shorthand for \( K_{xs} \).

While this closed-form predictive distribution is appealing and has found numerous applications, scaling it to large data sets is fundamentally limited by the matrix inversion \((K_{xx} + \sigma^2 I)^{-1}\) which needs \( O(n^3) \) operations.

This motivates the use of so-called “inducing point” methods which provide a framework for trading model quality for tractability. We assume that there exists a set of \( m \) inducing points \((z_1, \ldots, z_m) =: \mathbf{z}, z_i \in \mathcal{X}\) with outputs \( \mathbf{u} := f(\mathbf{z}) \) which are distributed \( \mathcal{N}(0, K_{zz}) \) according to the prior. Now, we make the modeling assumption that \( f \) and \( f^* \) are conditionally independent given \( \mathbf{u} \), i.e. \( p(f, f^*, \mathbf{u}) = p(f | \mathbf{u}) p(f^* | \mathbf{u}) p(\mathbf{u}) \). We can again solve the inference problem in closed-form:

\[
p(f^* | \mathbf{u}, \mathbf{z}, x^*) = \mathcal{N}(m^u, K^u)
\]

with

\[
m^u = K_{sz} K_{zz}^{-1} \mathbf{u}
\]

\[
K^u = K_{ss} - K_{sx} K_{zz}^{-1} K_{zs}
\]

Note that we have reduced the cubic part of inference from \( O(n^3) \) to \( O(m^3) \) where we can choose \( m \), the number of inducing points. Overall, the inference procedure has complexity \( O(m^3) \) \([22, 28, 39, 41]\).

Inducing point methods provide a framework for dramatically decreasing the computational complexity of inference, but we are still left with the problem of choosing the set of inducing points that achieves the best possible approximation with limited resources (namely \( m \) inducing points). This problem of inducing point selection can be formulated as an optimization objective in which we are trying to maximize the log marginal likelihood:

\[
\log p(y | z) = \iint p(y | f) p(f | u) p(u | z) du df
\]  

(1)

with respect to the locations \( \mathbf{z} \). Standard methods for solving the inducing points selection problem focus on continuous inputs and overlook the case of discrete inputs. In the following sections, we tackle this problem of selecting inducing points on discrete domains with three methods: Bayesian optimization, simulated annealing, and greedy inducing point selection.

### 2.1 Scalable Bayesian optimization

Bayesian Optimization of Combinatorial Structures (BOCS) is a recently published discrete optimization method that promises good scalability \([2]\). The algorithm approximates the true objective with a parameterized function, \( f_\alpha(\mathbf{z}) \approx \log p(y | \mathbf{z}) \). The function \( f_\alpha \) is a second-order approximation of the true objective but note that it is linear in the parameters \( \alpha \):

\[
f_\alpha(\mathbf{z}) = \alpha_0 + \sum_i \alpha_i z_i + \sum_{i,j>i} \alpha_{ij} z_i z_j .
\]

A horseshoe prior \([5]\) is placed on the interaction terms \( \alpha_{ij} \) to induce sparsity. The selection algorithms works iteratively. At iteration \( t \), \( \alpha_t \) is sampled from the posterior given the previous observations, \( \alpha_t \sim p(\alpha | \mathbf{z}_{1:t-1}, f_{\text{true}}(\mathbf{z}_{1:t-1})) \). Then, \( \alpha_t \) is used to compute the next location, \( \mathbf{z}_t := \arg \max_{\mathbf{z} \in \mathcal{X}\setminus\{\mathbf{z}_{1:t-1}\}} f_{\alpha_t}(\mathbf{z}) \). In this work, we naturally set \( f_{\text{true}}(\mathbf{z}_t) := \log p(y | \mathbf{z}_t) \).
2.2 Simulated annealing

Simulated annealing is a sampling-based approach which starts with an initial guess $S_0 := \{z_1, \ldots, z_m\}$ and a loss function $\mathcal{L}(\cdot)$ to be optimized. At each iteration the algorithm perturbs an element of the set and decides whether or not to accept this new perturbation as the next state. To make this decision, an energy term is computed from the current iterate $S_{t-1}$ and the proposal $\hat{S}$, $E_t = \mathcal{L}(S_{t-1}) - \mathcal{L}(\hat{S})$. The new proposal is then accepted with probability

$$P_{\text{accept}}(\hat{S}) = \min \left( 1, \exp \left( \frac{E_t}{T_t} \right) \right)$$

$T_t$ is known as the temperature parameter and is usually chosen with an exponential decay rate in $t$.

Since we are working on discrete string domains, we define a perturbation to be a change of one or more characters in a given string. Determining the number of characters to change requires careful fine tuning. In our experiments we chose the most conservative perturbation of a single character. The loss function is again naturally defined as $\mathcal{L}(z) := \log p(y \mid z)$.

2.3 Greedy selection

Greedy inducing point selection dates back to early works on sparse Gaussian Processes [6, 32, 39]. The algorithm is initialized with an empty set of inducing points and at each iteration, greedily selects the next observation in the data that maximizes the marginal likelihood $p(y \mid z)$ (Eq. 1). Thus, the set of inducing points is a mere subset of the original data set. This approach is justified by the fact that the marginal likelihood is strictly monotonic in the number of inducing points. Thus, adding a new inducing point is always guaranteed to increase the objective.

Natural extensions of this method include choosing the next point from a random subset of the full data at every iteration, selecting several inducing points instead of just one at every iteration, and optimizing a variational lower bound on the likelihood rather than the likelihood itself [39].

3 String kernels

Since we have biological sequences in mind for applications, an n-gram-based string kernel is a natural choice. Here we use the spectrum kernel [16] which was specifically designed for protein sequences and has also been successfully applied to other types of biological sequences [4]. There are also existing applications which use string kernels in Gaussian Processes but on small data sets ($n \approx 280$) where full Gaussian Process inference is viable [36].

Given an alphabet $\mathcal{A}$, we denote the input domain of all strings of finite length as $\mathcal{X} = \mathcal{A}^*$. The n-th order spectrum kernel is defined over this domain as

$$k_n(x, x') = \langle \Phi_n(x), \Phi_n(x') \rangle$$

with

$$\Phi_n(x) = [\phi_a(x)]_{a \in \mathcal{A}^n}$$

where $\phi_a(x)$ is the number of times that the string $a \in \mathcal{A}^n$ appears as a substring in $x$. This is essentially a bag-of-n-grams model.

While the set $\mathcal{A}^k$ might be prohibitively large, thus making the feature maps $\Phi_n(x)$ prohibitively high dimensional, it can easily be seen that we can compute $k_n(\cdot, \cdot)$ without having to represent $\Phi_n(x)$ explicitly. For two strings of arbitrary length, $x \in \mathcal{A}^{l(x)}$ and $x' \in \mathcal{A}^{l(x')}$, the kernel can be rewritten as

$$k_n(x, x') = \sum_{i=0}^{l(x)-n} \sum_{j=0}^{l(x')-n} \mathbb{1}[x_{i:i+n} = x'_{j:j+n}]$$

Computing this kernel naively has complexity $\mathcal{O}(l(x)^2)$ where w.l.o.g. $l(x) \geq l(x')$. This can be further improved using suffix trees resulting in a complexity of $\mathcal{O}(k \cdot l(x))$ [10].

We synthesize these three components – sparse GPs, discrete optimization, and string kernels – into a unified framework for supervised learning over discrete input spaces. This framework not
only has good predictive performance but also provides superior uncertainty estimates which we demonstrate in the following experiments.

4 Experiments

We compared different inducing point optimization methods for sparse GPs on toy data sets in regression, classification, and latent Bayesian inference settings. We then validated our framework’s performance on two real-world DNA sequence data sets from the UCI repository [7] and benchmarked our results against support vector machines (SVMs) with post hoc uncertainty calibration.

We find that our sparse string GP framework performs well when compared to full string GPs in a number of different settings. Moreover, the inducing points selected by the algorithm align well with the natural intuition for inducing points on continuous domains. Our framework offers comparable predictive performance with SVMs and yields superior uncertainty calibration. Furthermore, it performs well using different likelihoods, including Gaussian, Bernoulli, and Poisson, which demonstrates its general utility. For Poisson likelihoods, we are able to model count data in a natural manner which is difficult, if not impossible, to achieve with SVMs.

If not otherwise noted, all GPs and SVMs use a spectrum kernel as implemented in Shogun [34, 35]. For fitting the GPs, we used the GPy framework [12]. For fitting the SVMs and computing the performance metrics and calibration curves, we used the sklearn package [26].

4.1 Performance evaluation

In order to assess the predictive performance of our GP models on regression and classification tasks, we use the mean squared error (MSE) and area under the precision-recall curve (AUPRC), respectively. We use calibration curves [9, 42] to assess uncertainty calibration. Note that a perfectly calibrated classifier would lie directly on the diagonal of this plot.

4.2 Inducing point optimization for regression and classification

We developed a simple toy experiment to make comparisons in a controlled setting. We generated a small data set, consisting of 100 binary strings each of length 10. The small size of the data set was chosen to allow for tractable full GP inference. Additionally, this task also facilitates a good assessment of uncertainty quantification since a small number of samples leads to more uncertain predictions.

We defined a regression and a classification task on this data set. For regression, the task is to predict the number of ones in each string. The classification task is to classify whether a given string has more ones or zeros. We split the 100 observations into a train/test split of 60/40 and used 5 inducing points for all the sparse methods.

The full GP and all three sparse GPs agreed that \( k = 3 \) was the optimal order for the spectrum kernel with respect to the log marginal likelihood. This can be seen as a form of Bayesian model selection and implements an automatic tradeoff between goodness of fit and model complexity [29]. Methods like SVMs, which do not have an explicit likelihood, do not allow for such a Bayesian hyperparameter optimization. To optimize their kernel parameters, one often has to resort to explicit empirical methods like cross-validation.

In Table 1, we compare a full GP model, sparse GPs with four different inducing point selection methods, and SVMs. For the inducing point methods we used randomly chosen inducing points (random), greedy selection (greedy), simulated annealing (SA), and the BOCS discrete optimization method.

Surprisingly, greedy inducing point selection and simulated annealing perform better than BOCS on both likelihood and MSE. This could also be due to the differences in runtime between the methods. Given more time, it is plausible that BOCS would match or even achieve superior performance but in our experiments we found the algorithm to be prohibitively slow. In light of making a more realistic comparison, we decided to not let any method run longer than 100 seconds.
Table 1: Performance comparison of different inducing point optimization methods, a full GP and an SVM. Means and their standard errors are computed over 450 runs of the experiment in the regression setting and 500 runs in the classification setting.

| Method | likelihood | MSE       | runtime [s] | likelihood | AUPRC     | runtime [s] |
|--------|------------|-----------|-------------|------------|-----------|-------------|
| full GP | -39.08 ± 0.03 | 0.239 ± 0.004 | 0.01 ± 0.00 | -7.11 ± 0.07 | 0.986 ± 0.001 | 0.02 ± 0.00 |
| random | -44.64 ± 0.14 | 3.224 ± 1.591 | 0.01 ± 0.00 | -13.03 ± 0.65 | 0.979 ± 0.002 | 0.03 ± 0.00 |
| greedy | -40.67 ± 0.04 | 0.280 ± 0.005 | 0.34 ± 0.00 | -7.57 ± 0.08 | 0.986 ± 0.001 | 2.40 ± 0.00 |
| SA     | -40.05 ± 0.03 | 0.265 ± 0.004 | 12.24 ± 0.05 | -7.30 ± 0.07 | 0.986 ± 0.001 | 88.12 ± 0.10 |
| BOCS   | -42.77 ± 0.05 | 0.307 ± 0.007 | 99.24 ± 0.99 | -7.87 ± 0.08 | 0.985 ± 0.001 | 94.31 ± 0.83 |
| SVM    | -          | -          | -           | -          | 0.976 ± 0.001 | 0.01 ± 0.00 |

Figure 2: Visualization of the inducing points that are chosen by the sparse GP optimization. The histograms show the distribution of the number of ones in the inducing point strings. It can be seen that the inducing points are distributed evenly over the range of all strings in the regression setting (a), while they accumulate close to the decision boundary between 5 and 6 in the classification task (b).

Simulated annealing performed better than greedy selection but with marginal significance. If runtime is a concern, the greedy selection should be preferred.

It is evident that the sparse GP approximations cannot match the performance of the full GP, neither with respect to log-likelihood nor with respect to predictive performance. But, the two best inducing point optimization methods – greedy selection and simulated annealing – both approach the performance of the full GP whereas random inducing point selection clearly does not.

Upon inspection, the inducing points chosen in both regression and classification settings follow a natural intuition. In the regression task, the model has to count the number of ones equally well across all parts of the space. In the classification task, a more precise count close to the decision boundary is crucial for minimizing classification errors. Figure 2 clearly demonstrates this behavior.

If we compare the calibration of the different methods on the classification task, it can be seen that the full GP offers the best calibration while the SVM offers the worst (Fig. 3). Since the SVM does not natively output probabilities, we have to calibrate it in order to turn the SVM predictions into probabilities. The Calibrated SVM uses a technique called Platt scaling [27]. It performs a logistic regression on the SVM outputs and calibrates it using a cross-validation on the training data. The calibration ranking among the GPs is analogous to the one for the log likelihoods, i.e. the sparse GP with inducing points optimized by simulated annealing ranks second, the one with greedily selected points third.

This experiment shows that our sparse GP framework approaches the performance of a full GP in terms of predictive performance and uncertainty calibration while outperforming baseline SVM methods. We also find that inducing point selection in discrete string space follows our general
intuition for inducing point selection in continuous spaces. For small data sets such as the one used here, greedy inducing point selection offers the best tradeoff between inference runtime and predictive performance.

4.3 Inference on a latent process with non-Gaussian likelihood

One of the main advantages of Gaussian Processes is the fact that one can use any kind of likelihood for the generating process, i.e. the mapping from $f$ to $y$. One can then fit the Gaussian Process to the observed data and perform inference on the latent function values. To test this, we devised a data set that simulates a simple physical count process.

We generate random DNA strings over an alphabet with two different nucleotides. We then simulate a transcription factor whose binding rate is proportional to the amount of one of the nucleotides in the sequence. The counts of the binding events of that transcription factor over a defined time window are then Poisson-distributed with the given sequence-dependent rate.

We generate 100 sequences of length 10 and simulate the counts once for each sequence. The training and test set contain 50 sequences each. We again use a spectrum kernel with $k = 3$. To account for the Poisson distribution of the observations, we fit a Log Gaussian Cox Process (LGCP) \[22\] to the data, i.e. an inhomogeneous Poisson Process whose log rate is generated from a Gaussian Process. We compare the performance in terms of log likelihood and mean squared error on the counts as well as the inferred latent rate for a full LGCP, sparse LGCPs with different inducing point optimizers and a standard full GP with Gaussian likelihood as a baseline. The sparse methods use 10 inducing points each. Approximate inference on the Poisson likelihood is done using Expectation Propagation \[21\]. The results are reported in Table 2.

It can be seen that while the full GP is not significantly worse in terms of MSE on the observed

| Method      | likelihood  | MSE counts | MSE rates | runtime [s] |
|-------------|-------------|------------|-----------|-------------|
| full GP     | -71.88 ± 0.61| 0.626 ± 0.109| -         | 0.01 ± 0.00 |
| full LGCP   | -53.89 ± 1.57| 0.641 ± 0.089| 5.97 ± 1.45| 0.52 ± 0.01 |
| random LGCP | -54.21 ± 1.41| 0.614 ± 0.097| 5.17 ± 1.01| 6.46 ± 0.58 |
| greedy LGCP | -54.16 ± 1.74| 0.634 ± 0.099| 5.47 ± 0.98| 2184.59 ± 59.58 |
| SA LGCP     | -53.90 ± 1.57| 0.619 ± 0.102| 5.30 ± 1.04| 299.97 ± 6.73 |

Figure 3: Calibration curves for the different methods computed on toy data (a). It can be seen that the full GP has the best calibration, while the calibrated SVM has the worst. If one zooms into the figure (b), one can see that the sparse GP with inducing points selected by simulated annealing has the second best calibration, closely followed by the one with greedily selected inducing points.

Table 2: Performance comparison of different inducing point optimization methods, a full LGCP and a full GP on a Poisson regression and latent rate inference task. Means and their standard errors are computed over 7 runs of the experiment.
counts, it clearly underperforms the LGCP methods in terms of likelihood. This is to be expected, since the Gaussian likelihood does not fit the actual data generating process. It can also be seen that the greedy selection becomes very time intensive in this setting, because the approximate inference on the Poisson likelihood is computationally expensive. Overall, the simulated annealing offers the best sparse method here, but the random inducing point selection yields comparable results and is computationally cheaper.

Judging from these results, it might seem like a full GP is still a good modeling choice in this setting, if one only cares about the predictive performance. However, LGCPs are often not used to predict the actual count numbers, but the latent rates of the process [11,37]. It has been shown that this works as well in a sparse inducing point setting, even when one assumes infinite data [20]. This task cannot be achieved by a full GP, because it does not model the Poisson process explicitly.

In order to assess the LGCPs performance in this setting, we also performed inference on the latent rates and computed the MSE for those (MSE rates in Tab. 2). It can be seen that the sparse LGCPs do not perform significantly worse than the full LGCP.

This experiment shows that our framework works with different kinds of likelihoods, in this case a Log Gaussian Cox Process. The sparse approximations again approach the full GP’s performance and yield a significantly better data fit in terms of log likelihood than our baseline method. They also allow for successful Bayesian inference on the latent rate of the process, which is an interesting application and is only feasible due to the possibility to specify any kind of explicit likelihood (e.g. Poisson) in our framework.

### 4.4 Real world DNA sequence data

To validate our models on real world data, we performed classification on the UCI promoters data set [10] and the UCI splicing data set [25]. We aim to assess the predictive performance and uncertainty calibration and compare them against a support vector machine. The promoters data set consists of 106 real world DNA sequences of 57 nucleotides each and the task is to predict whether or not a given sequence corresponds to a promoter region of a gene. The splicing data set contains 3190 sequences of 60 nucleotides each which have to be classified into splicing and non-splicing sites.

We compared a kernel SVM against a full GP and our sparse GPs with inducing points selected greedily and by simulated annealing. Note that the splicing data is too large for feasible full GP inference, which is why we only compare the SVM and the sparse GPs there. The sparse GPs use 10 inducing points on the promoters data and 50 on the splicing data. The order of the spectrum kernel was chosen to be $k = 3$ by all GPs through log marginal likelihood optimization. The performance of the methods in terms of area under the precision-recall curve (AUPRC) and calibration is measured.
via 10-fold cross validation on the promoters data and by a 1000/2190 train-test-split on the splicing data. Results are reported in Figure 4.

It can be seen that the full GP, the different sparse GPs and the calibrated SVM are comparable in terms of calibration and predictive performance. The uncertainties are generally larger in the promoters data set because it is smaller. This is also the reason why the calibration curves are noisier in this data set, since there are fewer predictions per bin (Fig. 4a). The greedy sparse GP offers a marginally better calibration, but worse predictive performance, compared to the simulated annealing one on the promoters data. On the splicing data, it is the other way around. The dip at the end of all the calibration curves on the splicing data (Fig. 4b) is due to a single negatively labeled sequence in the test set that is virtually identical to a positively labeled sequence in the training set.

These experiments show that our framework yields a comparable performance with full GP inference and kernel SVMs on real world DNA sequence classification tasks. Moreover, it scales to larger data sets, where full GP inference becomes infeasible.

5 Related work

This work builds upon the rich literature on inducing point methods for Gaussian Processes (see [28] and references therein).

Recent work in this domain has utilized variational approximations [38] and certain geometrical structures [41]. Unfortunately, these advances are limited to continuous input spaces which is why we are forced to resort to more conventional inducing point methods in this work.

Many kernels have been devised to work well on discrete domains, e.g. on strings [16] or graphs [14]. These have been used successfully in combination with SVMs or similar linear models for problems in biology [4, 23], chemistry [18, 19], and natural language processing [17].

Using discrete kernels in GPs is a relatively unexplored area, possibly due to the difficulties in optimization. Discrete kernels have been used on graphs [40] and strings [3] (also for biological problems [36]), but so far only on relatively small data sets with full GPs. To the best of our knowledge, we are the first to address the scalability problem with discrete GPs using inducing points and discrete optimization.

6 Conclusion and future work

In this work, we present the first sparse Gaussian Process approximation framework on discrete domains. We explored different inducing point optimization techniques and found that simulated annealing gives the best overall predictive performance and uncertainty estimates while greedy selection offers the best tradeoff between performance and runtime.

We showed that our models perform competitively with SVMs on toy data as well as real-world DNA sequence data in terms of predictive performance, while offering better calibrated uncertainty estimates in some settings.

There are many directions for future work. First, developing a closer integration between discrete optimization and the marginal likelihood of the GP would improve both the approximation quality as well as the runtime of the inducing point algorithm. Extending the lower bound approximation of marginal likelihood as described in [38] to the discrete case may be one avenue to explore in this direction. An orthogonal direction is a fully Bayesian treatment of the string kernel hyperparameter $k$, namely treating $k$ as a random variable. Finally, a major pain point when it came to implementing our framework in code was the lack of string or other discrete kernels in existing GP software packages [12]. We would like to see these frameworks extend their abstractions to kernels with hyperparameters that are not differentiable.

In conclusion, we advise practitioners to use our framework on discrete problems where data sets are too large for full Gaussian Process models but uncertainty estimates are still desirable. Furthermore, in cases where likelihoods other than Gaussian or Bernoulli are required, standard regression and classification techniques are inapplicable, whereas our framework provides a principled and flexible solution.
References

[1] Ryan Prescott Adams, Iain Murray, and David J C MacKay. Tractable nonparametric bayesian inference in poisson processes with gaussian process intensities. In Proceedings of the 26th Annual International Conference on Machine Learning, ICML ’09, pages 9–16, New York, NY, USA, 2009. ACM.

[2] Ricardo Baptista and Matthias Poloczek. Bayesian optimization of combinatorial structures. June 2018.

[3] Daniel Beck and Trevor Cohn. Learning kernels over strings using gaussian processes. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers), volume 2, pages 67–73, 2017.

[4] Asa Ben-Hur, Cheng Soon Ong, Sören Sonnenburg, Bernhard Schölkopf, and Gunnar Rätsch. Support vector machines and kernels for computational biology. PLoS Comput. Biol., 4(10):e1000173, October 2008.

[5] Carlos M Carvalho, Nicholas G Polson, and James G Scott. Handling sparsity via the horseshoe. In Artificial Intelligence and Statistics, pages 73–80. jmlr.org, 2009.

[6] L Csató and M Opper. Sparse representation for gaussian process models. Adv. Neural Inf. Process. Syst., 2001.

[7] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[8] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. Nature, 521(7553):452, 2015.

[9] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. arXiv preprint arXiv:1706.04599, 2017.

[10] C B Harley and R P Reynolds. Analysis of e. coli promoter sequences. Nucleic Acids Res., 15(5):2343–2361, March 1987.

[11] Steven Hatch. Snowball in a Blizzard: The Tricky Problem of Uncertainty in Medicine. Atlantic Books Ltd, 2016.

[12] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. pages 282–290, September 2013.

[13] Martin Jinek, Krzysztof Chylinski, Ines Fonfara, Michael Hauer, Jennifer A Doudna, and Emmanuelle Charpentier. A programmable dual-rna–guided dna endonuclease in adaptive bacterial immunity. Science, page 1225829, 2012.

[14] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In Proceedings of the 19th international conference on machine learning, volume 2002, pages 315–322. people.cs.uchicago.edu, 2002.

[15] I Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. Artif. Intell. Med., 23(1):89–109, August 2001.

[16] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: a string kernel for SVM protein classification. Pac. Symp. Biocomput., pages 564–575, 2002.

[17] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. J. Mach. Learn. Res., 2(Fea):419–444, 2002.

[18] Pierre Mahé, Nobuhisa Ueda, Tatsuya Akutsu, Jean-Luc Perret, and Jean-Philippe Vert. Graph kernels for molecular structure-activity relationship analysis with support vector machines. J. Chem. Inf. Model., 45(4):939–951, July 2005.
[19] Pierre Mahé and Jean-Philippe Vert. Graph kernels based on tree patterns for molecules. *Mach. Learn.*, 75(1):3–35, April 2009.

[20] Alexander G de G Matthews, James Hensman, Richard E Turner, and Zoubin Ghahramani. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. *arXiv Preprints*, 2015.

[21] Thomas P Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI’01, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[22] Jesper Møller, Anne Randi Syversveen, and Rasmus Plenge Waagepetersen. Log gaussian cox processes. *Scand. Stat. Theory Appl.*, 25(3):451–482, 1998.

[23] Alyssa Morrow, Vaishaal Shankar, Devin Petersohn, Anthony Joseph, Benjamin Recht, and Nir Yosef. Convolutional kitchen sinks for transcription factor binding site prediction. May 2017.

[24] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, August 2012.

[25] Michiel O Noordewier, Geoffrey G Towell, and Jude W Shavlik. Training Knowledge-Based neural networks to recognize genes. In *NIPS*, page 530, 1990.

[26] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(Oct):2825–2830, 2011.

[27] John C Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, 1999.

[28] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *J. Mach. Learn. Res.*, 6(Dec):1939–1959, 2005.

[29] Carl Edward Rasmussen and Christopher K I Williams. *Gaussian Processes for Machine Learning*. MIT Press, new edition, January 2006.

[30] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo Method*. John Wiley & Sons, October 2016.

[31] Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert, editors. *Kernel Methods in Computational Biology*. MIT Press, 2004.

[32] Matthias Seeger, Christopher Williams, and Neil Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Artificial Intelligence and Statistics 9*. infoscience.epfl.ch, 2003.

[33] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In Y Weiss, B Schölkopf, and J C Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press, 2006.

[34] Soeren Sonnenburg, Heiko Strathmann, Sergey Lisitsyn, Viktor Gal, Fernando J Iglesias García, Wu Lin, Soumyajit De, Chiyuan Zhang, Frx, tklein23, Evgeniy Andreev, Jonas Behr, Spling, Parijat Mazumdar, Christian Widmer, Pan Deng / Zora, Giovanni De Toni, Saurabh Mahindre, Abhijeet Kislay, Kevin Hughes, Roman Votyakov, Khalednasr, Sanuj Sharma, Alesis Novik, Abinash Panda, Evangelos Anagnostopoulos, Liang Pang, Alex Binder, Serialhex, and Björn Esser. shogun-toolbox/shogun: Shogun 6.1.0, November 2017.
[35] Sören Sonnenburg, Gunnar Rätsch, Sebastian Henschel, Christian Widmer, Jonas Behr, Alexander Zien, Fabio de Bona, Alexander Binder, Christian Gehl, and Vojtěch Franc. The SHOGUN machine learning toolbox. *J. Mach. Learn. Res.*, 11:1799–1802, March 2010.

[36] Oliver Stegle, Linda Payet, Jean-Louis Mergny, David J C MacKay, and Julian Leon Huppert. Predicting and understanding the stability of g-quadruplexes. *Bioinformatics*, 25:374–382, 2009.

[37] Ming Teng, Farouk S Nathoo, and Timothy D Johnson. Bayesian computation for Log-Gaussian cox processes: A comparative analysis of methods. *J. Stat. Comput. Simul.*, 87(11):2227–2252, May 2017.

[38] M Titsias. Variational learning of inducing variables in sparse gaussian processes. *Artificial Intelligence and Statistics*, 2009.

[39] Michalis K Titsias. Variational model selection for sparse gaussian process regression. *Report, University of Manchester, UK*, 2009.

[40] Arun Venkitaraman, Saikat Chatterjee, and Peter Händel. Gaussian processes over graphs. March 2018.

[41] Andrew Gordon Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (KISS-GP). March 2015.

[42] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’02, pages 694–699, New York, NY, USA, 2002. ACM.