Gray-box inference for structured Gaussian process models

Pietro Galliani

University of Sussex

Thesis presented for the qualification of MPhil in Informatics.

I hereby declare that this thesis has not been and will not be submitted in whole or in part to another University for the award of any other degree.

The bulk of this thesis is taken from a paper with the same title, fruit of a collaboration between myself (first author), Edwin Bonilla, Amir Dezfouli and Novi Quadrianto, which has been accepted at AISTATS 2017 conference and has been published in its proceedings.

Signature:

Contents

Abstract

We develop an automated variational inference method for Bayesian structured prediction problems with Gaussian process (GP) priors and linear-chain likelihoods. Our approach does not need to know the details of the structured likelihood model and can scale up to a large number of observations. Furthermore, we show that the required expected likelihood term and its gradients in the variational objective (ELBO) can be estimated efficiently by using expectations over very low-dimensional Gaussian distributions. Optimization of the ELBO is fully parallelizable over sequences and amenable to stochastic optimization, which we use along with control variate techniques to make our framework useful in practice. Results on a set of natural language processing tasks show that our method can be as good as (and sometimes better than, in particular with respect to expected log-likelihood) hard-coded approaches including SVM-struct and CRFs, and overcomes the scalability limitations of previous inference algorithms based on sampling. Overall, this is a fundamental step to developing automated inference methods for Bayesian structured prediction.

1 Introduction

1.1 Structured prediction

Structured prediction problems are prediction problems in which the available data is best understood as consisting of structured objects (that is, objects with a non-trivial internal

Figure 1: An example of image segmentation: pixels of the imagine are labeled as belonging to bikes (green overlay), people (red overlay), or neither (no overlay). Segmentation based on the CRF-as-RNN algorithm of (Zheng et al., 2015). Image taken from `http://www.robots.ox.ac.uk/~szheng/crfasrnndemo`.

relational structure), rather than as consisting of scalar values or as values in some fixed-dimensionality vector (Bakir, 2007). Alternatively, structured prediction problems may be seen as problems in which there exist relations *between* different elements of the dataset.

The duality between these two views can be seen easily in two typical instances of structured prediction problems, namely *image segmentation* (Luccheseyz and Mitray, 2001) and *Part-of-Speech tagging* (Voutilainen, 2003).

The problem of image segmentation, in brief, consists in partitioning the pixels of a digital image in sets belonging to different categories (e.g. to the foreground/to the background, or to different physical objects), as shown for instance in Figure 1. Under the first view, each component of our data is an entire image, with the features ascribed to each pixel as well as a specification of the relative positions of these pixels; and the result of the prediction consists itself of a complex object which fixes a label for each pixel of the image.

The second view, on the other hand, allows us to regard each individual pixel as a distinct element of our data. This lets us ignore the problem of producing a complex output, as every pixel must be matched to one single label; but the cost of this is that under this view there exist relations *between* components of our data – i.e., spatial relationships between pixels of the same image – which carry information which is relevant to the prediction task. In particular, under this view it is not possible to assume, as it is often done in prediction tasks, that observations and labels are independently and identically distributed: the features and the labeling of a pixel, indeed, are not by any means independent from those of its neighbours.

Part-of-Speech tagging, instead, can be described as the problem of identifying, given a

Figure 2: A simple example of Part-of-Speech tagging: in the sentence "Nobody leaves the room", the word "leaves" is identified as a verb (rather than a noun or some other category). It would be impossible to make such a prediction on the basis of the features of the word "leaves" alone: in order to identify the category to which a word belongs, it is also necessary to consider its position with respect to the other words of its sentence.

natural language sentence, the category to which each word belongs. Again, there are two equivalent ways of understanding this problem: we can say that whole sentences (or even entire paragraphs or texts) are distinct elements of our data, and require that the outputs of our algorithm to be same-length sequences of word labelings, or we can say that the elements of our data are single words and the required outputs are the corresponding labelings. This view, again, allows us to eschew the difficulty of our algorithm returning complex structures as output, but at the cost – once more – of having relations *between* objects of our data (e.g. neighbourhood relations between words) which are relevant to their correct labeling, as well as at the cost of losing the assumption that the elements of our datasets are identically and independently distributed (i.i.d).

1.2 Related work

As already mentioned, structured prediction refers to the problem where there are interdependencies between samples and it is necessary to model these dependencies explicitly. Common examples are found in natural language processing (NLP) tasks, computer vision and bioinformatics. By definition, observation models in these problems are not i.i.d and standard learning frameworks have been extended to consider the constraints imposed by structured prediction tasks. Popular structured prediction frameworks are conditional random fields (CRFs; Lafferty et al., 2001), maximum margin Markov networks (Taskar et al., 2004) and structured support vector machines (SVM-struct, Tsochantaridis et al., 2005):

- Conditional random fields will be discussed in some detail in Section 2.1. In brief, they are discriminative, undirected probabilistic graphical models in which the weights associated to *factors* between variables are (generally linear) functions of the features of the individual variables (e.g. pixels or words) involved. The probability distribution over the (structured) outputs is then derived from those weights by means of the softmax function, as shown in Equation (2).

- Maximum margin Markov networks make use of margin-based optimization to learn the parameters of (generative) probabilistic graphical models, namely Markov networks. The

optimization algorithm makes use of the *kernel trick* (Aizerman, 1964) in order to operate over high-dimensional feature spaces without representing them esplicitly, and seeks the choice of parameter which maximizes the separation between classes. This problem is then reparametrized in a way that allows for an efficient solution, as long as the loss function can be decomposed in the same way as the feature map.

- Similarly to maximum margin Markov networks, the SVM-struct framework also extends to the structured classification case the maximum margin approach of support vector machines (SVM); but, differently from the cases of conditional random fields and maximum margin Markov networks, it is not a type of probabilistic graphical model and it does not return a probability distribution over the possible labelings, but just predictions for the correct labelings. SVM-struct makes use of a cutting plane algorithm in to solve certain quadratic optimization problems with exponentially many constraints in polynomial time.

Recurrent neural networks have also been used for certain structured prediction tasks (see e.g. Graves et al., 2012). Here we mention a novel and exciting approach is the exploration of the connection between conditional random fields and recurrent neural networks (Zheng et al., 2015), which allows for the possibility of treating conditional random fields as layers of a neural network architecture. The chief insight of (Zheng et al., 2015) is that one of the inference methods for conditional random fields, namely mean-field inference (Krähenbühl and Koltun, 2011), can be reformulated within the formalism of recurrent neural networks and then added as a layer to more complex neural networks. This approach promises to combine the advantages of neural networks and probabilistic graphical models for structured prediction.

Gaussian Processes (GP) are probability distributions over functions that may be thought of as infinite-dimensional generalisations of multivariate normal distributions. Formally, they may be defined (Rasmussen and Williams, 2006), as a collection of random variables, any finite number of which have a joint Gaussian distribution specified by a mean (often assumed to be constantly zero) and a covariance defined in terms of a *kernel function* $\kappa(\mathbf{x}, \mathbf{x}')$. Their application to (non-structured) regression and classification is largely due to the fact that conditioning multivariate normal distributions with respect to observed values yields another multivariate normal distribution having a straightforward and elegant form: if

$$\begin{pmatrix} \mathbf{f}(\mathbf{x}) \\ \mathbf{f}(\mathbf{x}') \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \kappa(\mathbf{x}, \mathbf{x}) & \kappa(\mathbf{x}, \mathbf{x}') \\ \kappa(\mathbf{x}', \mathbf{x}) & \kappa(\mathbf{x}', \mathbf{x}') \end{bmatrix}\right)$$

then the probability distribution of $\mathbf{f}(\mathbf{x}')$ conditioned over an observed value of $\mathbf{f}(\mathbf{x})$ is given by

$$\mathbf{f}(\mathbf{x}') \mid \mathbf{f}(\mathbf{x}) \sim \mathcal{N}(\kappa(\mathbf{x}', \mathbf{x})\kappa(\mathbf{x}, \mathbf{x})^{-1}\mathbf{f}(\mathbf{x}), \kappa(\mathbf{x}', \mathbf{x}') - \kappa(\mathbf{x}', \mathbf{x})\kappa(\mathbf{x}, \mathbf{x})^{-1}\kappa(\mathbf{x}, \mathbf{x}')). \tag{1}$$

A similar, only slightly more complicated expression applies in the case that the observations $\mathbf{f}(\mathbf{x})$ are subject to (additive and i.i.d.) Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

Gaussian Processes can be used as the key component of powerful and elegant nonparametric probabilistic machine learning models. However, their asymptotic computational performance often does not compare favourably to that of other approaches to regression or classification, due largely to the fact that computing (1) requires inverting the matrix $\kappa(\mathbf{x}, \mathbf{x})$, which is of size $N \times N$ where $N = |\mathbf{x}|$ is the number of observed samples. As inverting such a matrix carries $O(N^3)$ time complexity if this is done via Gaussian elimination[1], such an approach to regression or classification is not feasible even for comparatively small datasets.

Two techniques, however, have been recently used with good success to improve the computational performance of Gaussian Process-based regression and classification models:

- *Sparse* Gaussian Processes (Snelson and Ghahramani, 2006) differ from Gaussian Processes in that the conditioning of Equation (1) is not performed directly with respect to the training data $(\mathbf{x}, \mathbf{f}(\mathbf{x}))$ but rather with respect to a smaller set of $M \ll N$ *pseudoinputs* $\mathbf{z}$, whose locations and corresponding values $\mathbf{f}(\mathbf{z})$ can be chosen in a way that attempts to maximize the likelihood of the training data $(\mathbf{x}, \mathbf{f}(\mathbf{x}))$. The cost of the matrix inversion is then $O(M^3)$, which for suitable choices of M may lead to a computationally feasible model which anyway has good performance.

- *Variational approximation* techniques approximate the GP posterior (1) with a function g of known form. The task is then to learn the parameters of g in order to minimize a distance (espressed in terms of Kullback-Leiber divergence) between g and the true posterior.

Recent advances in sparse GP models for regression (Titsias, 2009; Hensman et al., 2013), which combine the two approaches mentioned above by drawing g from Gaussian Processes conditioned over the pseudoinputs $\mathbf{z}$ (whose positions and values are then the parameters with respect to which the KL divergence is minimized) have allowed the applicability of such models to very large datasets, opening opportunities for the extension of these ideas to classification and to non-structured problems with generic likelihoods (Hensman et al., 2015a; Nguyen and Bonilla,

[1] This is not asymptotically optimal – for instance, the Coppersmith–Winograd algorithm for matrix inversion (Coppersmith and Winograd, 1987) carries a time complexity of $\sim O(N^{2.376})$ – but it is often done in practice anyway due to numerical stability concerns and to the constant factors hidden by the asymptotic notation (Robinson, 2005).

2014; Dezfouli and Bonilla, 2015; Hensman et al., 2015b). However, none of these approaches is actually applicable to structured prediction problems, which inherently deal with likelihoods that do not factorize over individual samples.

A related, exciting area of research in machine learning which plays a major role in the above described research direction consists in the development of automated inference methods for complex probabilistic models, with notable examples in the probabilistic programming community given by STAN (Hoffman and Gelman, 2014) and CHURCH (Goodman et al., 2008). The principal thrust of this enterprise consists in the development of models and algorithms that may be deployed over a wide range of models and cost functions with a minimal amount of human intervention and fine-tuning. One of the main challenges for these approaches is to formulate expressive probabilistic models and develop generic yet efficient inference methods for them. From a variational inference perspective, one particular approach that has addressed such a challenge is the black-box variational inference framework of Ranganath et al. (2014). In brief, Ranganath et al. (2014) presents a variational optimization method for probabilistic models with latent variables in which the divergence between the variational approximation and the true posterior is minimized via a stochastic optimization algorithm, and the needed gradients are estimated via sampling. Crucially, the algorithm is largely agnostic with respect to the choice of model and likelihood, allowing the user to try out a vast range of models and likelihoods without need for time-consuming derivations.

While the works of Hoffman and Gelman (2014) and Ranganath et al. (2014) have been successful with a wide range of priors and likelihoods, their direct application to models with Gaussian process (GP) priors is cumbersome, mainly due to the large number of highly coupled latent variables in such models. In this regard, very recent work has investigated automated inference methods for general likelihood models when the prior is given by a sparse Gaussian process (Hensman et al., 2015b; Dezfouli and Bonilla, 2015). The framework of (Dezfouli and Bonilla, 2015), in particular, is at the basis of the one discussed in this work, and can be seen as the limit case (for sequences of length one) of the variational approximation approach described in detail in §3 and §4. However, while these advances have opened up opportunities for applying GP-based models well beyond regression and classification settings, they have focused on models with i.i.d observations and, therefore, are unsuitable for addressing the more challenging task of *structured prediction*.

From a non-parametric Bayesian modeling perspective, in general, and from a GP modeling perspective, in particular, structured prediction problems present very hard inference challenges because of the rapid explosion of the number of latent variables with the size of the problem. Furthermore, structured likelihood functions are usually very expensive to compute.

Twin Gaussian processes (Bo and Sminchisescu, 2010) address structured continuous-output problems by forcing input kernels to be similar to output kernels. In contrast, here we deal with the harder problem of structured *discrete*-output problems, where one usually requires computing expensive likelihoods during training. The structured continuous-output problem is somewhat related to the area of multi-output regression with GPs for which, unlike discrete structured prediction with GPs, the literature is relatively mature (Álvarez et al., 2010; Álvarez and Lawrence, 2011, 2009; Bonilla et al., 2008).

In an attempt to build non-parametric Bayesian approaches to (non-continuous) structured prediction, Bratières et al. (2015) have proposed GPSTRUCT, a framework based on a CRF-type modeling approach with GPs which uses elliptical slice sampling (ESS; Murray et al., 2010) as part of its inference method. Unfortunately, although their method can be applied to linear chain structures in a generic way without considering the details of the likelihood model, it is not scalable as it involves sampling from the full GP prior. We refer to §2.1 and §2.2 for a more detailed discussion (*sans* elliptical slice sampling) of the approach of Bratières et al. (2015), on which the approach discussed in this work is directly based.

Bratières et al. (2014) have explored a distributed version of GPSTRUCT based on the pseudo-likelihood approximation (Besag, 1975) where several weak learners are trained on subsets of GPSTRUCT's latent variables and bootstrap data, testing the resulting model with respect to image segmentation tasks. However, within each weak learner, inference is still done via MCMC. A variational alternative for GPSTRUCT inference (Srijith et al., 2014, 2016) is also available. However, it relies on pseudo-likelihood approximations, which in the (non-sparse) approach discussed in the paper are essential for the tractability of the algorithm, and was only evaluated on small-scale problems.

1.3 This work

In this work we present an approach for automated inference in structured GP models with linear chain likelihoods that builds upon the structured GP model of Bratières et al. (2015) and the sparse variational framework of Dezfouli and Bonilla (2015) (which, in turn, is a sparse variant of the variational framework of Nguyen and Bonilla (2014)). In particular, we show that the model of Bratières et al. (2015) can be mapped onto a generalization of the automated inference framework of Dezfouli and Bonilla (2015). Unlike the work of Bratières et al. (2015) our approach is scalable to a large number of observations, as it makes use of sparse GP priors; and unlike the work of (Srijith et al., 2014, 2016), our approach can deal with both pseudo-likelihoods and generic (linear-chain) structured likelihoods, and we rely on our sparse approximation procedure and our automated variational inference technique – rather than on bootstrap aggregation – to

achieve good performance on larger datasets.

More importantly, this approach is also generic in that it does not need to know the details of the likelihood model in order to carry out posterior inference. Finally, we show that our inference method is computationally efficient in that it only requires expectations over low-dimensional Gaussian distributions in order to carry out posterior approximation.

Our experiments on a set of NLP tasks, including noun phrase identification, chunking, segmentation, and named entity recognition, show that our method can be as good as (and sometimes better than, in particular with respect to expected log-likelihood) hard-coded approaches including SVM-struct and CRFs, and overcomes the scalability limitations of previous inference algorithms based on sampling.

We refer to our approach as "gray-box" inference since, in principle, changing the structure of the underlying graphical model (for instance, moving from a linear chain model to a skip-chain model, or extending a grid model for image processing by introducing factors between non-neighbouring pixels) cannot be done automatically. Nevertheless, when applied to fixed structures, our proposed inference method is entirely "black box". For example, as we will see, we can replace the exact likelihood with a pseudo-likelihood without needing to make any other modifications to our code.

2 Gaussian Process Models for Structured Prediction

2.1 Conditional Random Fields and Gaussian Processes

We are interested in structured prediction problems in which we observe input-output pairs $\mathcal{D} = \{\mathbf{X}_n, \mathbf{y}_n\}_{n=1}^{N_{\text{seq}}}$, where

- N_{seq} is the total number of observations;
- $\mathbf{X}_n \in \mathcal{X}$ is a descriptor of observation n;
- $\mathbf{y}_n \in \mathcal{Y}$ is the output corresponding to observation n;
- $\mathbf{X}_n$ and/or $\mathbf{y}_n$ are structured objects such as a sequence, a tree or a grid, that reflect the interdependences between its individual constituents.

Given a new input descriptor $\mathbf{X}_\star$, our goal is to predict its corresponding structured label $\mathbf{y}_\star$ and, more generally, a distribution over these labels.

Conditional Random Fields CRF (Lafferty et al., 2001; Sutton et al., 2012) are discriminative undirected probabilistic graphical models which constitute a state-of-the-art approach to structured prediction. In CRFs, relationships between variables (e.g. between the labels

of neighbouring pixels or words, or between the labels of pixels/words and the corresponding features) are represented as *factors* whose values are (generally linear) functions of the features of the variables involved.

Being *discriminative* probabilistic models, CRFs specify the conditional distribution $p(\mathbf{y}|\mathbf{X})$ of the labels $\mathbf{y}$ given the inputs $\mathbf{X}$. *Generative* probabilistic models for structural learning such as, for instance, Hidden Markov Models (Rabiner and Juang, 1986) would instead specify the joint distribution $p(\mathbf{X}, \mathbf{y})$.

Since CRFs are also *undirected*, their factors cannot in general be understood as conditional distributions specifying the probabilities of the values of a variable given those of the other variables. Instead, the probability distribution of a conditional random field can be expressed in terms of the values that factors take over *cliques* of variables pairwise connected by factors. More precisely, we can write

$$p(\mathbf{y}|\mathbf{X}, \mathbf{f}) = \frac{\exp\left(\sum_c f(c, \mathbf{X}_c, \mathbf{y}_c)\right)}{\sum_{\mathbf{y}' \in \mathcal{Y}} \exp\left(\sum_c f(c, \mathbf{X}_c, \mathbf{y}'_c)\right)}, \tag{2}$$

where

- $\mathbf{X}_c$ and $\mathbf{y}_c$ are tuples of nodes belonging to clique c;
- Every $f(c, \mathbf{X}_c, \mathbf{y}_c)$ is a (linear) function of the values of the features of the nodes in the clique c;
- The learning algorithm must seek good choices of weights for these linear combinations.

Calculating directly the value of Equation 2 is rarely computationally feasible. The difficulty, in brief, is in computing the expression at the denominator, which involves a sum over all possible labelings of the structure (whose number of increases exponentially with the size of the structure).

Various techniques, such as for instance sampling-based approaches or loopy belief propagation, are used to approximate efficiently the value of the above expression. When the graph associated to the structure does not contain cycles, however, there exist *message propagation algorithms* which can be used to efficiently and exactly compute the value of (2). Among the simplest of these algorithms is the *forward-backward algorithm*, which is applicable to *linear chain structures* (discussed in the next section) and whose complexity is only linear with respect to chain length and quadratic with respect to the number of (individual) labels. We refer to Guo and Hsu (2002) for a more in-depth discussion of inference in probabilistic graphical models.

In Bratières et al. (2015), a generalization of CRF-type models was proposed in which the $f(c, \mathbf{X}_c, \mathbf{y}_c)$ are are drawn from a distribution with a zero-mean Gaussian Process (GP) prior

with a given covariance function $\kappa(\cdot, \cdot; \boldsymbol{\theta})$, with $\boldsymbol{\theta}$ being the hyperparameters. It is clear that such a model is a generalization of vanilla CRFs where the potentials are draws from a GP instead of being (linear) functions of the features; and, as shown in Bratières et al. (2015), this approach leads to performances which are comparable or superior to those of conditional random fields and structured support vector machines (Tsochantaridis et al., 2005), another class commonly used structured prediction models. However, the use of GP priors leads to computational costs that severely hamper the scalability of this type approach: indeed, the time complexity of this approach grows as $O(N^3)$, where N is the number of *individual samples* (e.g. individual words or pixels) in the dataset.[2]

The present work is an attempt to improve the scalability of (Bratières et al., 2015) by means of two main techniques:

1. Probabilities are computed not via sampling from the (computationally expensive) posterior distribution, but rather through a *variational approximation* which generalizes to structured model the variational approach to Gaussian Processes of (Nguyen and Bonilla, 2014);

2. The Gaussian Process itself is replaced with a *sparse* GP *approximation* after the way of (Quiñonero-Candela and Rasmussen, 2005) and (Titsias, 2009).

These two techniques were already combined in (Dezfouli and Bonilla, 2015), yielding an efficient sparse variational framework for (non-structured) Gaussian process models with black-box[3] likelihoods. The present work, thus, is an attempt to use the insights of (Dezfouli and Bonilla, 2015) to derive a more scalable version of the GP-based approach to structured prediction of (Bratières et al., 2015).

2.2 Linear chain structures

In this work we focus on linear chain structures where the output corresponding to datapoint n is a linear chain of length T_n, whose corresponding constituents stem from a common set.

In other words, there are two types of cliques in linear chain structures, both of which involve only two nodes:

1. *Unary* cliques, which describe the relationship between the features $\mathbf{x}_t$ at a given point of the chain and the corresponding label y_t;

2. *Binary* cliques, which describe the relationship between a label y_t of a non-terminal node of a chain and the label t_{n+1} of its successor.

[2] In brief, this is due to the necessity of inverting the $N \times N$ covariance matrices of the Gaussian Processes.
[3] In the sense that different choices of likelihood could be used without deriving new expressions for the gradients or changing the rest of the algorithm.

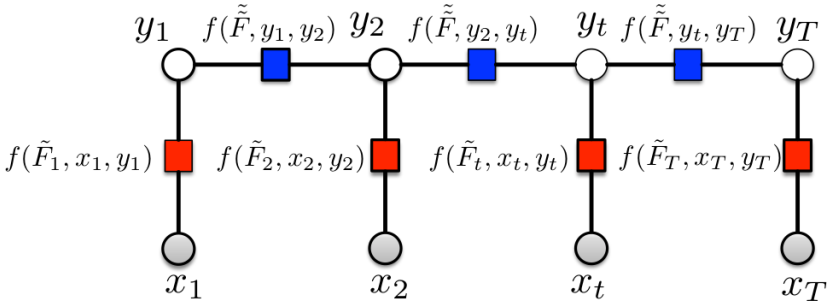

Figure 3: A linear chain model. Unary cliques are in red, binary cliques are in blue. The values $f(\tilde{F}_i, x_i, y_i)$ are the factors corresponding to unary cliques for Equation (2), while the values $f(\tilde{\tilde{F}}_i, y_i, y_{i+1})$ are the factors for the binary cliques.

Linear chain structures are well-suited for applications, for instance, in natural language processing; and they have the significant advantage that the corresponding softmax likelihoods, as defined in Equation (2), can be computed in an efficient way via the *forward-backward algorithm*, whose complexity is of the order of $|\mathcal{V}|^2 T$ where T is the length of the sequence and $|\mathcal{V}|$ is the number of labels. This is a significant improvement over a brute-force application of Equation (2), which would require us to sum over all $|\mathcal{V}|^T$ possible state assignments for a total complexity of $O(T|\mathcal{V}|^T)$. Figure 3 shows a typical instance of linear chain structure, together with the factors $f(\tilde{F}_i, x_i, y_i)$ and $f(\tilde{\tilde{F}}_i, y_i, y_{i+1})$ corresponding to unary and binary factors in Equation (2).

In this setting, the input $\mathbf{X}_n$ for observation n is a $T_n \times D$ matrix $\begin{pmatrix} \mathbf{x}_1 \\ \dots \\ \mathbf{x}_{T_n} \end{pmatrix}$ of feature descriptors and $\mathbf{y}_n$ is a the corresponding output sequence $(y_1 \dots y_{T_n})$ of T_n labels drawn from the same vocabulary $\mathcal{V}$.

In order to completely define the prior over the clique-dependent latent functions in Equation (2), it is now necessary to specify covariance functions over the cliques. To this end, Bratières et al. (2015) propose a kernel that is non-zero only when two cliques are of the same type, i.e. both are unary cliques $(t, \mathbf{x}_t, y_t)$ or both are pairwise cliques (y_t, y_{t+1}). These kernels are defined as

$$\kappa_{\text{un}}((t, \mathbf{x}_t, y_t), (t', \mathbf{x}'_{t'}, y_{t'})) = \mathbb{I}[y_t = y_{t'}]\kappa(\mathbf{x}_t, \mathbf{x}'_{t'}),$$
$$\kappa_{\text{bin}}((y_t, y_{t+1}), (y_{t'}, y_{t'+1})) = \mathbb{I}[y_t = y_{t'} \wedge y_{t+1} = y_{t'+1}],$$

where κ_{un} is the covariance on unary functions and κ_{bin} is the covariance on pairwise functions.

In other words, according to the prior distribution of (Bratières et al., 2015)

- If the features $\mathbf{x}_t$ and $\mathbf{x}'_{t'}$ corresponding to two positions t and t' in two chains $\mathbf{X}$ and

$\mathbf{X}'$ are similar (that is, if $\kappa(\mathbf{x}_t, \mathbf{x}'_{t'})$ is high) then, for all possible labels y, the factors $f(\tilde{F}_t, \mathbf{x}_t, y)$ and $f(\tilde{F}_{t'}, \mathbf{x}'_{t'}, y)$ that correspond to the assignment of the label y for the two positions (and which affect the likelihood of the labeling as per Equation 2) are also close. The relative positions t and t' of the nodes in the corresponding chains are not relevant to this, and the distribution of these weights is independent from the distributions of the $f(\tilde{F}_{t''}, \mathbf{x}_{t''}, y'')$ for other labels $y'' \neq y$.

- The probability distribution of the weights $f(\tilde{\tilde{F}}, y_t, y_{t+1})$ to be assigned to a pair (y_t, y_{t+1}) of labels appearing at positions $(t, t+1)$ is uniform and independent from the distributions of factors to unary cliques or to other binary cliques (corresponding to other positions or to other choices of labels).

This choice of GP kernel describes – with a suitable ordering of the latent functions – a distribution with a block-diagonal covariance matrix. Its first $|\mathcal{V}|$ blocks, each of which describes (for the corresponding label y) the covariances between the values of the unary factors $f(\tilde{F}_t, \mathbf{x}_t, y)$, are of size $\sum_n T_n$ each. The last block, corresponding to covariances between pairwise factors $f(\tilde{\tilde{F}}, y_t, y_{t+1})$, is a diagonal (identity) matrix of size $|\mathcal{V}|^2$, where $|\mathcal{V}|$ denotes the vocabulary size.

To carry out inference in this model, Bratières et al. (2015) propose a sampling scheme based on elliptical slice sampling (ESS; Murray et al., 2010). In the following section, we show an equivalent formulation of this model that leverages the general class of (non-structured) models presented by Nguyen and Bonilla (2014). Understanding structured GP models from such a perspective will allow us to generalize the results of Nguyen and Bonilla (2014); Dezfouli and Bonilla (2015) in order to develop an automated variational inference framework. The advantages of such a framework are that of (i) dealing with generic likelihood models; and (ii) enabling stochastic optimization techniques for scalability to large datasets.

Different choices of prior are of course possible: for instance, it might be useful to deal separately with the distributions of the prior weights to the first or last nodes of the chains. However, in what follows we will using essentially the same prior described above and used in Bratières et al. (2015).[4] This will allow us to make a more straightforward evaluation of the effect of the variational approximation (based on the work of Nguyen and Bonilla (2014); Dezfouli and Bonilla (2015)) on the performance of the resulting system.

[4]The main differences are that the kernel functions corresponding to different labels will not necessarily be the same and that the pairwise component of the covariance matrix will not be necessarily required to be the identity.

3 Automated Inference

3.1 Generalising the Nugyen-Bonilla class of models

Nguyen and Bonilla (2014), building upon the work of Opper and Archambeau (2009), developed an automated variational inference framework for a family of class models with Gaussian process priors. Although such an approach is an important step towards black-box inference with GP priors, since it assumes i.i.d observations it is, by definition, unsuitable for structured models.

One way to generalize such an approach to structured models such as the ones described in §2.2 is to differentiate between GP priors over latent functions on unary nodes and GP priors over latent functions over pairwise nodes. More importantly, rather than considering likelihoods that factorize over the individual samples of our dataset, we assume likelihoods that factorize over sequences while allowing for statistical dependences within a sequence.

As in the case of the choice of kernel used in (Bratières et al., 2015) and discussed in the previous section, the prior probability distributions of the factors of unary cliques corresponding to different labels will be made to be independent, as will the prior probability distributions of different binary cliques. Furthermore, all prior distributions over the factors for unary cliques will be independent from the prior distributions over the factors for binary cliques.

Therefore, our prior model $p(\mathbf{f}) = p(\mathbf{f}^{\text{un}})p(\mathbf{f}^{\text{bin}})$ for linear chain structures will decompose as

$$p(\mathbf{f}) = \left(\prod_{j=1}^{|\mathcal{V}|} \mathcal{N}(\mathbf{f}_j^{\text{un}}; \mathbf{0}, \mathbf{K}_j) \right) \mathcal{N}(\mathbf{f}^{\text{bin}}; \mathbf{0}, \mathbf{K}^{\text{bin}}), \tag{3}$$

where $\mathbf{f}$ is the vector of all latent function values of unary nodes $\mathbf{f}^{\text{un}}$ and pairwise nodes $\mathbf{f}^{\text{bin}}$. Accordingly, $\mathbf{f}_j^{\text{un}}$ is the vector (of size equal to the total number of observations $N = \sum_{n=1}^{N_{\text{seq}}} T_n$) of the unary functions of latent process j, corresponding to the jth label in the vocabulary. This vector is drawn from a zero-mean GP with covariance function $\kappa_j(\cdot, \cdot; \boldsymbol{\theta}_j)$. This covariance function, when evaluated at all the input pairs in $\{\mathbf{X}_n\}$, induces the $N \times N$ covariance matrix $\mathbf{K}_j$. Similarly, $\mathbf{f}^{\text{bin}}$ is a zero-mean $|\mathcal{V}|^2$-dimensional Gaussian random variable with covariance matrix given by $\mathbf{K}^{\text{bin}}$. We note here that, again as per the kernel choice of (Bratières et al., 2015) discussed in the previous section, while the unary functions are draws from a GP indexed by $\mathbf{X}$ the distribution over pairwise functions is a finite Gaussian (not indexed by $\mathbf{X}$).

Given the latent function values, our conditional likelihood is defined by

$$p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^{N_{\text{seq}}} p(\mathbf{y}_n|\mathbf{f}_n), \tag{4}$$

where, omitting the dependency on the input $\mathbf{X}$ for simplicity, each individual conditional

likelihood term is computed using a likelihood function for sequential data such as that defined by the structured softmax function in Equation (2). Here, $\mathbf{y}_n$ denotes the labels of sequence n and $\mathbf{f}_n$ is the corresponding vector of latent (unaries and pairwise) function values; and, since we are working with linear chain models, each likelihood $p(\mathbf{y}_n|\mathbf{f}_n)$ can be computed efficiently via the forwards-backwards algorithm.

Theorem 1. *Every model in the class proposed by Bratières et al. (2015) is also in the model class defined by the prior in Equation* (3) *and the likelihood in Equation* (4)*.*

The proof of this is trivial and can be done by (i) setting all the covariance functions of the unary latent processes to be the same; (ii) making $\mathbf{K}^{\text{bin}} = \mathbf{I}$; and (iii) using the structured softmax function in Equation (2) as each of the individual terms $p(\mathbf{y}_n|\mathbf{f}_n)$ in Equation (4). This yields exactly the same model as specified by Bratières et al. (2015), with prior covariance matrix with block-diagonal structure described in §2.2 above. ■

The above theorem shows that our approach may be meaningfully compared to that of Bratières et al. (2015) and may, in fact, be considered a slight generalisation of it insofar as the model class is concerned. Moreover, the class of models which we are considering contains also, as a limit case for chains of length one, that of Nguyen and Bonilla (2014). Thus, it makes sense to inquire whether the variational inference approach of Nguyen and Bonilla (2014), which achieves good computational efficiency by approximating the posterior as a Gaussian mixture and computing gradients by means of expectations over *one-dimensional* Gaussian distributions, can be adapted to our case.

As we shall see in the next section, this is indeed true; and in order to deal with the intractable nonlinear expectations inherent to variational inference (VI), the proposed method will require expectations over low-dimensional Gaussian distributions (but not one-dimensional as in Nguyen and Bonilla (2014) itself).

3.2 Automated variational inference

In this section we develop a method for estimating the posterior over the latent functions given the prior and likelihood models defined in Equations (3) and (4). Since the posterior is analytically intractable and the prior involves a large number of coupled latent variables, we resort to approximations given by variational inference (VI; Jordan et al., 1998). To this end, we

start by defining our variational approximate posterior distribution:

$$q(\mathbf{f}) = q(\mathbf{f}^{\text{un}})q(\mathbf{f}^{\text{bin}}), \quad \text{for} \tag{5}$$

$$\begin{aligned} q(\mathbf{f}^{\text{un}}) &= \sum_{k=1}^{K} \pi_k q_k(\mathbf{f}^{\text{un}}|\mathbf{b}_k, \mathbf{\Sigma}_k) \\ &= \sum_{k=1}^{K} \pi_k \prod_{j=1}^{|\mathcal{V}|} \mathcal{N}(\mathbf{f}_j^{\text{un}}; \mathbf{b}_{kj}, \mathbf{\Sigma}_{kj}) \quad , \end{aligned} \tag{6}$$

$$q(\mathbf{f}^{\text{bin}}) = \mathcal{N}(\mathbf{f}^{\text{bin}}; \mathbf{m}^{\text{bin}}, \mathbf{S}^{\text{bin}}), \tag{7}$$

where $q(\mathbf{f}^{\text{un}})$ and $q(\mathbf{f}^{\text{bin}})$ are the approximate posteriors over the unary and pairwise nodes respectively; each $q_k(\mathbf{f}_j^{\text{un}}) = \mathcal{N}(\mathbf{f}_j^{\text{un}}; \mathbf{b}_{kj}, \mathbf{\Sigma}_{kj})$ is a N-dimensional full Gaussian distribution; and $q(\mathbf{f}^{\text{bin}})$ is a $|\mathcal{V}|^2$-dimensional Gaussian.

The choice of Gaussian mixture models for our variational approximations is due to two related reasons:

1. This was the same choice of variational approximation used, with good success, in Nguyen and Bonilla (2014), and this work constitutes an attempt to generalize their approach to structured prediction problems;

2. Gaussian mixture models have a mathematical form that is well-suited for the efficient computation of our gradients (see the Appendix for the details). Using a different family of functions for our variational approximations would require substantial changes to the present work, and could be rather more computationally expensive.

In our experiments, we will actually consider only single (multivariate) Gaussians as our variational approximation: in other words, we will always set $K = 1$ and $\pi_1 = 1$ in the above expressions, and then (obviously) we will no concern ourselves with computing gradients with respect to π_1. This will be done in order to reduce the computational cost of our experiments, as well as because preliminary experiments did not show significant improvement in our performance when using $K > 1$ for our datasets. We leave a more detailed analysis of our approach when $K > 1$ to future work.

In order to estimate the parameters of the above distribution, variational inference entails the optimization of the so-called evidence lower bound ($\mathcal{L}_{\text{elbo}}$), which can be shown to be a lower bound of the true marginal likelihood, and is composed of a KL-divergence term ($\mathcal{L}_{\text{kl}}$), between the approximate posterior and the prior, and an expected log likelihood term ($\mathcal{L}_{\text{ell}}$):

$$\mathcal{L}_{\text{elbo}} = -\text{KL}(q(\mathbf{f})\|p(\mathbf{f})) + \langle \log p(\mathbf{y}|\mathbf{f}) \rangle_{q(\mathbf{f})} \,, \tag{8}$$

where the angular bracket notation $\langle \cdot \rangle_q$ indicates an expectation over the distribution q. Although the approximate posterior is an N-dimensional distribution, the expected log likelihood term can be estimated efficiently using expectations over much lower-dimensional Gaussians:

Theorem 2. *For the structured* GP *model defined in Equations* (3) *and* (4)*, the expected log likelihood over the variational distribution defined in Equations* (5) *to* (7) *and its gradients can be estimated using expectations over T_n-dimensional Gaussians and $|\mathcal{V}|^2$-dimensional Gaussians, where T_n is the length of each sequence and $|\mathcal{V}|$ is the vocabulary size.*

The proof is constructive and can be found in the supplementary material. Let $\mathcal{L}_{\text{ell}}^{(k,n)} \stackrel{\text{def}}{=} \langle \log p(\mathbf{y}_n|\mathbf{f}_n) \rangle$ be the individual expected log-likelihood terms, and let $\boldsymbol{\lambda}^{\text{un}}$ and $\boldsymbol{\lambda}^{\text{bin}}$ be the variational parameters corresponding to unary and binary factors. $\mathcal{L}_{\text{ell}}$ and its gradients are then given by

$$\mathcal{L}_{\text{ell}} = \sum_{n=1}^{N_{\text{seq}}} \sum_{k=1}^{K} \pi_k \mathcal{L}_{\text{ell}}^{(k,n)}, \tag{9}$$

$$\nabla_{\boldsymbol{\lambda}_k^{\text{un}}} \mathcal{L}_{\text{ell}}^{(k,n)} = \left\langle \log p(\mathbf{y}_n|\mathbf{f}_n) \nabla_{\boldsymbol{\lambda}_k^{\text{un}}} \log q_{kn}(\mathbf{f}_n^{\text{un}}) \right\rangle, \tag{10}$$

$$\nabla_{\boldsymbol{\lambda}^{\text{bin}}} \mathcal{L}_{\text{ell}}^{(k,n)} = \left\langle \log p(\mathbf{y}_n|\mathbf{f}_n) \nabla_{\boldsymbol{\lambda}^{\text{bin}}} \log q(\mathbf{f}^{\text{bin}}) \right\rangle, \tag{11}$$

where the expectations are computed wrt the approximate marginal posterior $q_{kn} = q_{kn}(\mathbf{f}_n^{\text{un}}) q(\mathbf{f}^{\text{bin}})$; and $q_{kn}(\mathbf{f}_n^{\text{un}})$ is a $(T_n \times |\mathcal{V}|)$-dimensional Gaussian with block-diagonal covariance $\boldsymbol{\Sigma}_{k(n)}$, each block being of size $T_n \times T_n$ since by Equation (6) the components of our variational approximate posterior corresponding to different labels are independent from each other. Therefore, we can estimate the above terms by sampling from T_n-dimensional Gaussians independently. Furthermore, $q(\mathbf{f}^{\text{bin}})$ is a $|\mathcal{V}|^2$-dimensional Gaussian, which can also be sampled independently. In practice, we can assume that the covariance of $q(\mathbf{f}^{\text{bin}})$ is diagonal and only sample from univariate Gaussians for the pairwise functions.

It is important to emphasize the practical consequences of Theorem 2. Although we have a fully correlated prior and a fully correlated approximate posterior over $N = \sum_{n=1}^{N_{\text{seq}}} T_n$ unary function values, yielding full N-dimensional covariances, we have shown that for these classes of models we can estimate $\mathcal{L}_{\text{ell}}$ by only using expectations over T_n-dimensional Gaussians. We refer to this result as the *computational efficiency* of the inference algorithm.

Nevertheless, even when having only one latent function and using a single Gaussian approximation ($K = 1$), optimization of the $\mathcal{L}_{\text{elbo}}$ in Equation (8) is completely impractical for any realistic dataset concerned with structured prediction problems, due to its high memory requirements $\mathcal{O}(N^2)$ and time complexity $\mathcal{O}(N^3)$: indeed, computing the term $\text{KL}(q(\mathbf{f}) \| p(\mathbf{f}))$ of Equation (8) would require storing and inverting the $N \times N$-dimensional covariance matrices

$\boldsymbol{\Sigma}_{kj}$.

In the next section we will use a sparse GP approach within our variational framework in order to develop a practical algorithm for structured prediction.

4 Sparse Approximation

In this section we describe a scalable approach to inference in the structured GP model defined in §3 by introducing the so-called sparse GP approximations (Quiñonero-Candela and Rasmussen, 2005) into our variational framework. Variational approaches to sparse (but not structured) GP models were developed by Titsias (2009) for Gaussian likelihoods, then made scalable to large datasets and generalized to non-Gaussian likelihoods by Hensman et al. (2015a,b); Dezfouli and Bonilla (2015). The main idea of such approaches is to introduce a set of M *inducing variables* $\{\mathbf{u}_m\}_{m=1}^M$ for each latent process, which lie in the same space as $\{\mathbf{f}_m\}$ and are drawn from the same GP prior. These inducing variables are the latent function values of their corresponding set of *inducing inputs* $\{\mathbf{Z}_m\}$. Subsequently, we redefine our prior in terms of these inducing inputs/variables.

In our structured GP model, only the unary latent functions are drawn from GPs indexed by $\mathbf{X}$. Hence we assume a GP prior over the inducing variables and a conditional prior over the unary latent functions, which both factorize over the latent processes. This yields the joint distribution over unary functions, pairwise functions and inducing variables given by:

$$p(\mathbf{f}, \mathbf{u}) = p(\mathbf{u})p(\mathbf{f}^{\text{un}}|\mathbf{u})p(\mathbf{f}^{\text{bin}}), \tag{12}$$

where

- The marginal prior over the inducing variables is $p(\mathbf{u}) = \prod_{j=1}^{|\mathcal{V}|} p(\mathbf{u}_j)$;
- The conditional prior is given by $p(\mathbf{f}^{\text{un}}|\mathbf{u}) = \prod_{j=1}^{|\mathcal{V}|} \mathcal{N}(\mathbf{f}_j^{\text{un}}; \tilde{\boldsymbol{\mu}}_j, \widetilde{\mathbf{K}}_j)$;
- The prior over the pairwise functions is defined as before, i.e. $p(\mathbf{f}^{\text{bin}}) = \mathcal{N}(\mathbf{f}^{\text{bin}}; \mathbf{0}, \mathbf{K}^{\text{bin}})$.

The means and covariances of the individual conditional distributions over the unary functions are given by: $\tilde{\boldsymbol{\mu}}_j = \mathbf{A}_j\mathbf{u}_j$ and $\widetilde{\mathbf{K}}_j = \kappa_j(\mathbf{X}, \mathbf{X}) - \mathbf{A}_j\kappa(\mathbf{Z}_j, \mathbf{X})$ with $\mathbf{A}_j = \kappa(\mathbf{X}, \mathbf{Z}_j)\kappa(\mathbf{Z}_j, \mathbf{Z}_j)^{-1}$.

By keeping an explicit representation of the inducing variables, our goal is to estimate the joint posterior over the unary functions, pairwise functions and inducing variables given the observed data. To this end, we assume that our variational approximate posterior is given by:

$$q(\mathbf{f}, \mathbf{u}|\boldsymbol{\lambda}) = q(\mathbf{u}|\boldsymbol{\lambda}^{\text{un}})p(\mathbf{f}^{\text{un}}|\mathbf{u})q(\mathbf{f}^{\text{bin}}|\boldsymbol{\lambda}^{\text{bin}}), \tag{13}$$

where

- $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}^{\text{un}}, \boldsymbol{\lambda}^{\text{bin}}\}$ are the variational parameters;
- $p(\mathbf{f}^{\text{un}}|\mathbf{u})$ is defined as above;
- $q(\mathbf{f}^{\text{bin}}|\boldsymbol{\lambda}^{\text{bin}})$ is defined as in Equation (7), i.e. a Gaussian with parameters $\boldsymbol{\lambda}^{\text{bin}} = \{\mathbf{m}^{\text{bin}}, \mathbf{S}^{\text{bin}}\}$;
- It holds that
$$q(\mathbf{u}|\boldsymbol{\lambda}^{\text{un}}) = \sum_{k=1}^{K} \pi_k q_k(\mathbf{u}|\mathbf{m}_k, \mathbf{S}_k), \tag{14}$$
with $q_k(\mathbf{u}|\mathbf{m}_k, \mathbf{S}_k) = \prod_{j=1}^{|\mathcal{V}|} \mathcal{N}(\mathbf{u}_j; \mathbf{m}_{kj}, \mathbf{S}_{kj})$;
- $\boldsymbol{\lambda}^{\text{un}} = \{\pi_k, \mathbf{m}_k, \mathbf{S}_k\}$;
- $\mathbf{m}_{kj}$ and $\mathbf{S}_{kj}$ denote the posterior mean and covariance of the inducing variables for mixture component k and latent function j, and have dimensionality M and $M \times M$ respectively.

As mentioned berfore, in our experiments we will use single Gaussians rather than Gaussian mixtures; thus, in Equation (14) we will always have $K = 1$, $\pi_1 = 1$.

4.1 Evidence lower bound

The KL term in the evidence lower bound now considers a KL divergence between the joint approximate posterior in Equation (13) and the joint prior in Equation (12). Because of the structure of the approximate posterior, it is easy to show that the term $p(\mathbf{f}^{\text{un}}|\mathbf{u})$ vanishes from the KL (see e.g. Titsias, 2009), yielding an objective function that is composed of a KL between the distributions over the inducing variables, a KL between the distributions over the pairwise functions, and the expected log likelihood over the joint approximate posterior:

$$\begin{aligned}\mathcal{L}_{\text{elbo}}(\boldsymbol{\lambda}) = &- \text{KL}(q(\mathbf{u})\|p(\mathbf{u})) - \text{KL}(q(\mathbf{f}^{\text{bin}})\|p(\mathbf{f}^{\text{bin}})) \\ &+ \left\langle \sum_{n=1}^{N_{\text{seq}}} \log p(\mathbf{y}_n|\mathbf{f}_n) \right\rangle_{q(\mathbf{f},\mathbf{u}|\boldsymbol{\lambda})},\end{aligned} \tag{15}$$

where $\text{KL}(q(\mathbf{f}^{\text{bin}})\|p(\mathbf{f}^{\text{bin}}))$ is a straightforward KL divergence between two Gaussians and $\text{KL}(q(\mathbf{u})\|p(\mathbf{u}))$ is a KL divergence between a mixture of Gaussians and a Gaussian, which we bound using Jensen's inequality. The expressions for these terms are given in the supplementary material.

Let us now consider the expected log-likelihood term in Equation (15), which is an expectation of the conditional likelihood over the joint posterior $q(\mathbf{f}, \mathbf{u}|\boldsymbol{\lambda})$. The following result tells us that,

as in the non-sparse case, this term can still be estimated efficiently using expectations over low-dimensional Gaussians.

Theorem 3. *The expected log likelihood term in Equation* (15)*, with a generic structured conditional likelihood* $p(\mathbf{y}_n|\mathbf{f}_n)$ *and variational distribution* $q(\mathbf{f}, \mathbf{u}|\boldsymbol{\lambda})$ *defined in Equation* (13)*, and its gradients can be estimated using expectations over* T_n*-dimensional Gaussians and* $|\mathcal{V}|^2$*-dimensional Gaussians, where* T_n *is the length of each sequence and* $|\mathcal{V}|$ *is the vocabulary size.*

As in the full (non-sparse) case, the proof is constructive and can be found in the supplementary material. This means that, in the sparse case, the expected log likelihood and its gradients can also be computed using Equations (9) to (11), where the mean and covariances of each $q_{kn}(\mathbf{f}_n^{\text{un}})$ are determined by the means and covariances of the variational posterior over the inducing variables. As before, the unary components corresponding to different labels are independent and $q_{kn}(\mathbf{f}_n^{\text{un}})$ is thus a $(T_n \times |\mathcal{V}|)$-dimensional Gaussian with block-diagonal structure, where each of the $j = 1, \ldots, |\mathcal{V}|$ blocks has mean and covariance given by

$$\mathbf{b}_{kjn} = \mathbf{A}_{jn}\mathbf{m}_{kj}, \tag{16}$$

$$\boldsymbol{\Sigma}_{kjn} = \widetilde{\mathbf{K}}_j^n + \mathbf{A}_{jn}\mathbf{S}_{kj}\mathbf{A}_{jn}^T \tag{17}$$

where

$$\mathbf{A}_{jn} \stackrel{\text{def}}{=} \kappa(\mathbf{X}_n, \mathbf{Z}_j)\kappa(\mathbf{Z}_j, \mathbf{Z}_j)^{-1} \quad , \tag{18}$$

$$\widetilde{\mathbf{K}}_j^n \stackrel{\text{def}}{=} \kappa_j(\mathbf{X}_n, \mathbf{X}_n) - \mathbf{A}_{jn}\kappa(\mathbf{Z}_j, \mathbf{X}_n) \tag{19}$$

and as mentioned in §2.2, $\mathbf{X}_n$ is the $T_n \times D$ matrix of feature descriptors corresponding to sequence n.

4.2 Expectation estimates

In order to estimate the expectations in Equations (9) to (11), we use a simple Monte Carlo approach where we draw samples from our approximate distributions and compute the empirical expectations. For example, for the $\mathcal{L}_{\text{ell}}$ we have:

$$\widehat{\mathcal{L}}_{\text{ell}} = \frac{1}{S}\sum_{n=1}^{N_{\text{seq}}}\sum_{k=1}^{K}\pi_k\sum_{i=1}^{S}\log p(\mathbf{y}_n|\mathbf{f}_{nki}^{\text{un}}, \mathbf{f}_i^{\text{bin}}), \tag{20}$$

with $\mathbf{f}_{nki}^{\text{un}} \sim \mathcal{N}(\mathbf{b}_{k(n)}, \boldsymbol{\Sigma}_{k(n)})$ and $\mathbf{f}_i^{\text{bin}} \sim \mathcal{N}(\mathbf{m}^{\text{bin}}, \mathbf{S}^{\text{bin}})$, for $i = 1, \ldots, S$. Here S is the number of samples, and each of the individual blocks of $\mathbf{b}_{k(n)}$ and $\boldsymbol{\Sigma}_{k(n)}$ are obtained from Equations

(16) and (17), respectively, collecting the entries for all labels j.[5] As mentioned before, the π_k in this case are the weights assigned to the individual Gaussian component of our variational posterior, which is a mixture of Gaussians; and in our experiments, we will have $K = 1$ and $\pi_1 = 1$.

We use a similar approach for estimating the gradients of the $\mathcal{L}_{\text{ell}}$ and they are given in the supplementary material.

5 Learning

We learn the parameters of our model, i.e., the parameters of our approximate variational posterior and the hyperparameters ($\{\boldsymbol{\lambda}, \boldsymbol{\theta}\}$) through gradient-based optimization of the variational objective ($\mathcal{L}_{\text{elbo}}$). One of the main advantages of our method is the decomposition of $\mathcal{L}_{\text{ell}}$ in Equation (20) and its gradients as a sum of expectations of the individual likelihood terms for each sequence. This result enables us to use parallel computation and stochastic optimization in order to make our algorithms useful in practice.

In our experiments, we use 500 inducing inputs $\{\mathbf{Z}_j\}$ and select them via K-means clustering. This number was chosen to provide reasonable cross-experiment balance between performance and computational cost; the corresponding statistic for the non-sparse case would be the total number of training words $\bar{N}$, which as shown in Table 1 is consistently greater than 500 but varies considerably between experiments. We leave a detailed analysis of the effect of changing the number of inducing inputs to future work.

As discussed in the supplementary material, the step sizes for stochastic gradient descent were chosen automatically and adaptively by our code.

5.1 Computational complexity

The time-complexity of our stochastic optimization is dominated by the computation of the posterior's entropy, Gaussian sampling, and running the forward-backward algorithm, which yields an overall cost of $O(M^3 + T_n^3 + ST_n|\mathcal{V}|^2)$ for each sequence n. Indeed:

- The forward-backward algorithm allows us to compute the likelihood (for each one of the S samples) at cost $O(T_n|\mathcal{V}|^2)$;
- Inverting sums of the $M \times M$ matrices $\mathbf{S}_{kj}$, as well as the (also $M \times M$) matrix $\kappa(\mathbf{Z}_j, \mathbf{Z}_j)$, in order to compute the entropy terms and gradients[6] costs $O(M^3)$;

[5] Hence, $\mathbf{b}_{k(n)}$ is of dimension $T_n|\mathcal{V}|$ and $\boldsymbol{\Sigma}_{k(n)}$ is of dimension $T_n|\mathcal{V}| \times T_n|\mathcal{V}|$, as required for the mean and covariance of the $T_n \times |\mathcal{V}|$-dimensional distribution $\mathbf{f}^{\text{un}}_{nki}$.

[6] See Appendices B and D.1.1 for details.

- The inverse of $\kappa(\mathbf{Z}_j, \mathbf{Z}_j)$ is also necessary to compute $\mathbf{b}_{kjn}$ and $\boldsymbol{\Sigma}_{kjn}$ according to Equations (16)–(19);

- Once the $\mathbf{b}_{kjn}$ and $\boldsymbol{\Sigma}_{kjn}$ are obtained, inverting the $T_n \times T_n$ matrices $\boldsymbol{\Sigma}_{kjn}$ in order to draw the samples $\mathbf{f}_{nki}^{\text{un}} \sim \mathcal{N}(\mathbf{b}_{k(n)}, \boldsymbol{\Sigma}_{k(n)})$ required to compute $\widehat{\mathcal{L}}_{\text{ell}}$ according to Equation (20) carries a computational cost of $O(T_n^3)$.

The space complexity is dominated by storing inducing-point covariances, which is $O(M^2)$. To put this in the perspective of other available methods, the existing Bayesian structured model with ESS sampling (Bratières et al., 2015) has time and memory complexity of $O(N^3)$ and $O(N^2)$ respectively, where N is the total number of observations (e.g. words). CRF's time and space complexity with stochastic optimization depends on the feature dimensionality, i.e., it is $O(D)$. The actual running time of CRF also depends on the cost of model selection via a cross-validation procedure. ESS sampling makes the method of Bratières et al. (2015) completely unfeasible for large datasets and CRF has high running times for problems with high dimensions and many hyperparameters. Our work aims to make Bayesian structured prediction practical for large datasets, while being able to use infinite-dimensional feature spaces as well as sidestepping a costly cross-validation procedure.

5.2 Variance reduction

Our goal is to approximate an expectation of a function $g(\mathbf{f})$ over the random variable $\mathbf{f}$ that follows a distribution $q(\mathbf{f})$, i.e., $\mathbb{E}_q[g(\mathbf{f})]$ via Monte Carlo samples. The simplest way to reduce the variance of the empirical estimator $\bar{g}$ is to subtract from $g(\mathbf{f})$ another function $h(\mathbf{f})$ that is highly correlated with $g(\mathbf{f})$. We note that, in the case of variational inference, this technique was introduced in Blei et al. (2012). In more detail, for any value of $\hat{a}$, the function $\tilde{g}(\mathbf{f}) := g(\mathbf{f}) - \hat{a}h(\mathbf{f})$ will have the same expectation as $g(\mathbf{f})$, i.e., $\mathbb{E}_q[\tilde{g}] = \mathbb{E}_q[g]$, provided that $\mathbb{E}_q[h] = 0$. In general, to ensure unbiasedness, $\mathbb{E}_q[h]$, if easily and efficiently computable, can be subtracted from h to form an estimator $\tilde{g} := g - h + \mathbb{E}_q[h]$. More importantly, as the variance of the new function is $\text{Var}[\tilde{g}] = \text{Var}[g] + \hat{a}^2\text{Var}[h] - 2\hat{a}\text{Cov}[g, h]$, our problem boils down to finding suitable $\hat{a}$ and h so as to minimize $\text{Var}[\tilde{g}]$.

In our case, $q(\mathbf{f})$ is the variational distribution and $g(\mathbf{f}) = \log p(\mathbf{y}_n|\mathbf{f}_n)\nabla_\lambda \log q(\mathbf{f})$ (see supplementary material). Previous work (Ranganath et al., 2014; Dezfouli and Bonilla, 2015) has found that a suitable correction term is given by $h(\mathbf{f}) = \nabla_\lambda \log q(\mathbf{f})$, which has expectation zero. Given this, the optimal $\hat{a}$ can be computed as $\hat{a} = \text{Cov}[g, h]/\text{Var}[h]$. The use of control variates is essential to achieve good performance in our framework.

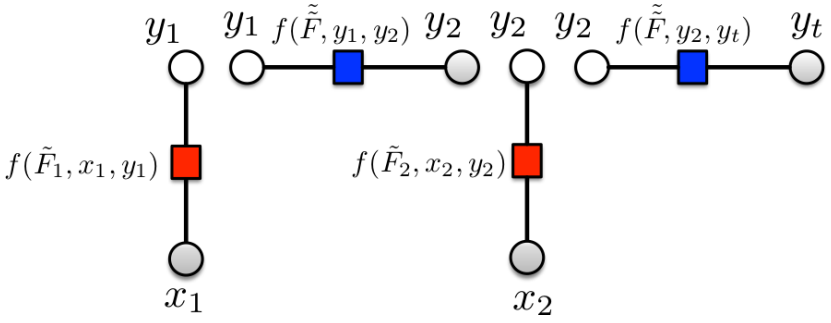

Figure 4: Decomposing a linear model via pseudolikelihood: only local interactions inside unary (red) or binary (blue) factors are considered. Compare with the original model (Figure 3).

5.3 Piecewise pseudo-likelihood

In order to demonstrate the flexibility of our approach, we also tested the performance of our framework when the true likelihood is approximated by a piecewise pseudo-likelihood (Sutton and McCallum, 2007) that only takes in consideration the local interactions within a single factor between the variables in our model.

In the context of this work, as illustrated in Figure 4, this is the same as replacing the likelihood $p(\mathbf{y}_n|\mathbf{f}^{\text{un}}_{nki}, \mathbf{f}^{\text{bin}}_i)$ of sequence n given the latent functions $\mathbf{f}^{\text{un}}_{nki}$ and $\mathbf{f}^{\text{bin}}_i$ with the product,[7] for every single factor and every variable occurring in it, of the conditional probability of the variable given its neighbours with respect to that factor.

In our linear model, this yields the following expression for the log pseudolikelihood $\tilde{p}(\mathbf{y}_n|\mathbf{f}^{\text{un}}_{nki}, \mathbf{f}^{\text{bin}}_i)$:

$$\log \tilde{p}(\mathbf{y}_n|\mathbf{f}^{\text{un}}_{nki}, \mathbf{f}^{\text{bin}}_i) = \sum_{w=1}^{W_n} \log p((\mathbf{y}_n)_w|\mathbf{f}^{\text{un}}_{nki}) + \sum_{|w_1-w_2|=1} \log p((\mathbf{y}_n)_{w_1}|(\mathbf{y}_n)_{w_2}, \mathbf{f}^{\text{bin}}_i)$$

where W_n is the number of words in sentence n and

$$p((\mathbf{y}_n)_w|\mathbf{f}^{\text{un}}_{nki}) \propto \exp(\mathbf{f}^{\text{un}}_{nki_{\mathbf{y}_n}}(w)); \tag{21}$$

$$p((\mathbf{y}_n)_w|(\mathbf{y}_n)_{w+1}, \mathbf{f}^{\text{bin}}_i) \propto \exp(\mathbf{f}^{\text{bin}}_i((\mathbf{y}_n)_w, (\mathbf{y}_n)_{w+1})); \tag{22}$$

$$p((\mathbf{y}_n)_{w+1}|(\mathbf{y}_n)_w, \mathbf{f}^{\text{bin}}_i) \propto \exp(\mathbf{f}^{\text{bin}}_i((\mathbf{y}_n)_w, (\mathbf{y}_n)_{w+1})). \tag{23}$$

We emphasize that this change did not require any modification to our inference engine and we simply used this pseudo-likelihood as a drop-in replacement for the exact likelihood.

[7] Here i represents the index of the specific samples $\mathbf{f}^{\text{un}}_{nki}$ and $\mathbf{f}^{\text{bin}}_i$ taken from our distributions $q_{kn}(\mathbf{f}^{\text{un}}_n)$ and $q(\mathbf{f}^{\text{bin}})$.

5.4 Making and evaluating predictions

For a given choice of parameters and hyperparameters, we make predictions and evaluate them as follows.

For any new sequence with features $\mathbf{X}_\star$, we first compute the corresponding $\mathbf{b}_{kj\star}$ and $\mathbf{\Sigma}_{kj\star}$ from the learned parameters $\mathbf{m}_{kj}$ and $\mathbf{S}_{kj}$ according to Equations (16) and (17) for $n = \star$; then, once more, we sample a latent function $\mathbf{f}^{\text{un}}_{\star ki} \sim \mathcal{N}(\mathbf{b}_{k(\star)}, \mathbf{\Sigma}_{k(\star)})$ (as well as a latent function $\mathbf{f}^{\text{bin}}_i \sim \mathcal{N}(\mathbf{m}^{\text{bin}}, \mathbf{S}^{\text{bin}})$ for the binary factors). Then, we use the forward-backward algorithm to compute the posterior marginals of all labels at all positions of the sequence. This is done for multiple samples $i = 1 \dots S$, averaging the resulting marginals.

In this way, for each position t of the sequence and each label y we obtain a probability

$$P(y_t = y) = \frac{1}{S} \sum_{k=1}^{K} \pi_k \sum_{i=1}^{S} \log p(y_t = y | \mathbf{f}^{\text{un}}_{\star ki}, \mathbf{f}^{\text{bin}}_i)$$

where S is the number of samples used, $\mathbf{f}^{\text{un}}_{\star ki}$ and $\mathbf{f}^{\text{bin}}_i$ are sampled as just discussed, and – as mentioned before – for our experiments it is always the case that $K = 1$, $\pi_1 = 1$.

Then our prediction for position t of the sequence n is simply the label y which maximizes this probability, that is, $\text{argmax}_y P(y_t = y)$, and the mean error rates of our method with respect to our testing datasets are computed on the basis of these predictions. We also compute the *negative log likelihoods* by summing, for all sequences in the testing and for all positions of these sequences, the negative logarithms of the probabilities, of the true labels.

Regardless of our choice of using the true likelihood or the piecewise pseudo-likelihood when training, we always use the true likelihood – computed via the forward-backward algorithm – when making the predictions. Indeed, the chief advantage of the piecewise pseudo-likelihood over the true likelihood lies in its smaller computational cost, which is mainly a concern during training (since we need to compute it, for each sampling of our latent functions, at every optimization step), while the main advantage of the true likelihood lies in its higher accuracy (which is a greater concern during the prediction phase).

6 Experiments

6.1 Datasets and experimental baselines

In this section we describe how we evaluated our approach on the benchmarks used by Bratières et al. (2015), which target several standard NLP problems and are summarized in Table 1.

These include noun phrase identification (BASE NP); chunking, i.e. shallow parsing labels sentence constituents (CHUNKING); identification of word segments in sequences of Chinese

ideograms (SEGMENTATION); and Japanese named entity recognition (JAPANESE NE). We also consider larger-scale experiments on BASE NP and CHUNKING, which have significantly more training data available.[8]

We tested the performance of our method, both when using the true likelihood (GP-VAR-T) and when using the piecewise pseudolikelihood (GP-VAR-P), against that of three other approaches:

- SVM: Structured Support Vector Machines (Tsochantaridis et al., 2005), making use of Joachims' implementation found at `http://www.cs.cornell.edu/people/tj/svm_light/svm_struct.html`;
- CRF: Conditional Random Fields, using Schmidt and Swersky's Matlab implementation for linear models, which can found at `https://www.cs.ubc.ca/~schmidtm/Software/crfChain.html`;
- GP-ESS: The elliptical slice sampling-based GPSTRUCT, as described in (Bratières et al., 2015), using an implementation provided by the authors.

We refer to the Introduction for a brief description of Structured Support Vector Machines. Conditional random fields and GP-ESS have been discussed in §2.

The hyperparameters of these methods were chosen automatically through cross-validation, as per the given implementations. When comparing the negative log likelihoods we did not make use of the results of SVM, because this method does not produce probability estimates.

For large-scale experiments, we only compared our method (with pseudolikelihood) with CRF, as it was the best baseline in our previous experiments.

6.2 Small-scale experiments

When comparing the error rates on Table 2 we see that our approach is on par with competitive benchmarks which, unlike our method, exploit the structure of the likelihood (in the sense that these implementations are custom-made and optimized for that choice of likelihood, which could not be modified without extensive reworkings of the implementations).

More importantly, when analyzing the test likelihoods on Table 3 we see that our method with true likelihood (GP-VAR-T) is significantly better than CRF for all benchmarks except SEGMENTATION, where it has a similar performance. Finally, the log-likelihood results of GP-ESS (Bratières et al., 2015) are also consistently worse than ours, owing largely to the higher computational cost of sampling.

[8] BASE NP and CHUNKING are in fact the same dataset, with different choices of features and labels.

Table 1: Datasets used in our experiments. For each dataset we see the number of categories (or vocabulary $|\mathcal{V}|$), the number of features (D), the number of training sequences in small (N_{seq}-small) and large (N_{seq}-big) scale experiments, and the average (across folds) number of training words ($\bar{N}$-small and $\bar{N}$-big). Large-scale experiments were only performed for the BASE NP and CHUNKING datasets.

Dataset	$\lvert\mathcal{V}\rvert$	D	N_{seq}-small	$\bar{N}$-small	N_{seq}-big	$\bar{N}$-big
BASE NP	3	6,438	150	3,739.8	500	11,611
CHUNKING	14	29,764	50	1,155.8	500	11,611
SEGMENTATION	2	1,386	20	942	-	-
JAPANESE NE	17	102,799	50	1,315.4	-	-

Table 2: Mean error rates and standard deviations in brackets on small-scale experiments using 5-fold cross-validation. The average number of observed words ($\bar{N}$) on these problems range from 942 to 3740. SVM corresponds to structured support vector machines; CRF to conditional random fields; GP-ESS corresponds to GPSTRUCT with ESS for inference (Bratières et al., 2015); GP-VAR-T corresponds to our method with true likelihood; and GP-VAR-P corresponds to our our method with piecewise pseudo-likelihood.

Dataset	Method				
	SVM	CRF	GP-ESS	GP-VAR-T	GP-VAR-P
BASE NP	5.9 (0.4)	5.3 (0.5)	**5.1 (0.4)**	5.6 (0.5)	5.2 (0.3)
CHUNKING	9.8 (1.0)	**8.5 (1.0)**	**8.5 (1.0)**	9.4 (1.6)	9.0 (1.0)
SEGMENTATION	16.2 (2.2)	15.4 (1.1)	14.9 (1.8)	**14.5 (1.5)**	15.3 (2.2)
JAPANESE NE	5.6 (0.8)	**5.2 (0.7)**	5.6 (0.7)	5.4 (0.6)	5.6 (0.6)

Table 3: Negative expected log-likelihoods and standard deviations in brackets on small-scale experiments using 5-fold cross-validation. As before, CRF refers to conditional random fields; GP-ESS to GPSTRUCT with ESS for inference (Bratières et al., 2015); GP-VAR-T to our method with true likelihood; and GP-VAR-P to our method with piecewise pseudo-likelihood.

Dataset	Method			
	CRF	GP-ESS	GP-VAR-T	GP-VAR-P
BASE NP	944 (835)	887 (57)	622 (34)	**603 (21)**
CHUNKING	517 (113)	704 (116)	**407 (43)**	587 (100)
SEGMENTATION	**253 (41)**	316 (52)	255 (45)	298 (53)
JAPANESE NE	592 (131)	806 (135)	**339 (38)**	411 (94)

6.3 Larger-scale experiments

Here we evaluate our approach on BASE NP and CHUNKING using $N_{\text{seq}} = 500$ training sequences and the remaining 323 sequences for testing, with a five-fold cross-validation setting. This amounts to roughly $11,611$ training words on average. We compare with CRF, as this was the best baseline in our previous experiments. We also note that that the original GP-ESS method is completely impractical in this setting.

On BASE NP, our method has a lower average test negative log-likelihood (1265.52 vs. 1355.63) but a higher error rate (5.15% vs 4.50%) than CRF. However, our results on CHUNKING, 2511.48 vs 1862.96 for test log-likelihoods and 8.60% vs 7.20% for error rates, indicate that our method lags behind CRF on this dataset. We attribute this result to CHUNKING having a much higher dimensionality than BASE NP, which is a more critical issue with large datasets, and to the fact that our implementation is not particularly optimized.

6.4 Discussion

As the previous results show, the performance of our variational approach is roughly on par with state-of-the-art methods (and, in particular, consistently better than structured support vector machines SVM) for structured inference over chain models, at least with respect to comparatively small datasets, while being more scalable to the sampling-based implementation GP-ESS (which could not be feasibly used for our large-scale experiments).

We cannot claim that our approach is consistently better than custom-made implementations of CRF for linear chain models; but it is important to remark here that our approach is more *flexible* than CRF in that different choices of likelihood function (e.g. pseudolikelihood rather than true likelihood) can be used without changing anything in the code except the likelihood function itself. This is not the case for CRF: indeed, the likelihood function (computed via the forward-backward algorithm) and its gradients are hard-coded in the used implementation of CRF, and changing it would require a substantial reworking of its code.

This is an advantage that our approach shares with the sampling-based approach GP-ESS; and, in fact, the present work is in essence an attempt to make said approach more scalable by means of variational and sparse approximation techniques. Overall, the results of our experiments can be considered successful on this note: even though our performance on large-scale experiments was not consistently superior to that of CRF, as already mentioned our approach is inherently more flexible than it in that it allows to try out different likelihood functions without the need of changing the rest of the implementation.

7 Conclusion

7.1 Future Work

The work discussed in this thesis could be extended in various ways.

First of all, in our experiments we considered only single Gaussians for our variational approximations, even though our approach may be easily extended to Gaussian mixtures. This was because preliminary experiments suggested that increasing the number K of Gaussians in our variational approximations did not result in improvements for the datasets that we chose to consider; but it would be worthwhile to investigate whether this remains the case when considering other datasets.

The techniques described in this work could also be applied to the problem of image segmentation, extending the work of (Bratières et al., 2014).

Moreover, it could be interesting to explore the benefits of using neural networks to learn the kernels of our Gaussian processes, along the lines of (Hinton and Salakhutdinov, 2008), or even of using variationally approximated structured Gaussian process-based predictors as the last layer of a neural network: in regard to this, the ideas of (Zheng et al., 2015) about the interpretability of Conditional Random Fields in terms of recurrent neural networks could be an intriguing starting point.

Finally, one could study ways of incorporating domain knowledge (in particular, domain knowledge regarding spatial relationships between objects) in our framework. This is a fairly open-ended problem, in general; but for instance, (Hu et al., 2016), in which an expectation maximization-like framework is explored in which a "teacher" network adjusts a "student" network's prediction by taking into account domain knowledge while the "student" negotiates between the two objectives of fitting the training data and imitating the "teacher" network, could well constitute a very suitable initial point for our exploration.

7.2 Concluding Remarks

We studied a Bayesian structured prediction model with GP priors and linear-chain likelihoods. We developed an automated variational inference algorithm that only requires expectations over low-dimensional Gaussians in order to estimate the expected likelihood term in the variational objective. We exploited these types of theoretical insights as well as practical statistical and optimization tricks to make our inference framework scalable and effective. Our model generalizes recent advances in CRFs (Koltun, 2011) by allowing general positive definite kernels defining their energy functions and opens new directions for combining deep learning with structure models (Zheng et al., 2015).

As mentioned in the introduction, when adapting our approach to new structured prediction problems one may need to set up a new configuration of the latent functions (e.g. the unary and pairwise functions of our linear-chain case), which would require substantial modifications to the code. Thus, the process of developing an inference procedure, using our approach, for a different structure (e.g. when considering higher-order interactions) requires some human intervention. Nevertheless, when applied to fixed structures our approach is "black box" with respect to the choice of likelihood, inasmuch as different likelihoods can be used without any change to the inference engine.

We have already seen a possible way to extend our method to more general structured likelihoods, where the exact likelihood is replaced by a piecewise pseudo-likelihood. Such an approach might be especially valuable when adapting our framework to models such as grids or skip-chains, for which the evaluation of the true structured likelihood would be intractable.

Overall, we believe our approach is a fundamental step to developing automated inference methods for general structured prediction problems.

8 Bibliography

Mark Aizerman. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.

Mauricio Álvarez and Neil D Lawrence. Sparse convolved Gaussian processes for multi-output regression. In *NIPS*, pages 57–64. 2009.

Mauricio A Álvarez and Neil D Lawrence. Computationally efficient convolved multiple output Gaussian processes. *JMLR*, 12(5):1459–1500, 2011.

Mauricio A. Álvarez, David Luengo, Michalis K. Titsias, and Neil D. Lawrence. Efficient multioutput Gaussian processes through variational inducing kernels. In *AISTATS*, 2010.

Gökhan Bakir. *Predicting structured data*. MIT press, 2007.

Julian Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 24:179–195, 1975.

David M Blei, Michael I Jordan, and John W Paisley. Variational Bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1367–1374, 2012.

Liefeng Bo and Cristian Sminchisescu. Twin Gaussian processes for structured prediction. *International Journal of Computer Vision*, 87(1-2):28–52, 2010.

Edwin V. Bonilla, Kian Ming A. Chai, and Christopher K. I. Williams. Multi-task Gaussian process prediction. In *NIPS*. 2008.

Sébastien Bratières, Novi Quadrianto, Sebastian Nowozin, and Zoubin Ghahramani. Scalable Gaussian process structured prediction for grid factor graph applications. In *ICML*, 2014.

Sébastien Bratières, Novi Quadrianto, and Zoubin Ghahramani. GPstruct: Bayesian structured prediction using Gaussian processes. *IEEE TPAMI*, 37:1514–1520, 2015.

Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6. ACM, 1987.

Amir Dezfouli and Edwin V Bonilla. Scalable inference for Gaussian process models with black-box likelihoods. In *NIPS*. 2015.

Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. Church: A language for generative models. In *UAI*, 2008.

Alex Graves et al. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.

Haipeng Guo and William Hsu. A survey of algorithms for real-time bayesian network inference. In *AAAI/KDD/UAI02 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*. Edmonton, Canada, 2002.

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. In *UAI*, 2013.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *AISTATS*, 2015a.

James Hensman, Alexander G Matthews, Maurizio Filippone, and Zoubin Ghahramani. MCMC for variationally sparse Gaussian processes. In *NIPS*. 2015b.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Using deep belief nets to learn covariance kernels for gaussian processes. In *Advances in neural information processing systems*, pages 1249–1256, 2008.

Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *JMLR*, 15(1):1593–1623, 2014.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. *An introduction to variational methods for graphical models*. Springer, 1998.

Vladlen Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *NIPS*, 2011.

Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117, 2011.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

L Luccheseyz and SK Mitray. Color image segmentation: A state-of-the-art survey. *Proceedings of the Indian National Science Academy (INSA-A)*, 67(2):207–221, 2001.

Iain Murray, Ryan Prescott Adams, and David J.C. MacKay. Elliptical slice sampling. In *AISTATS*, 2010.

Trung V. Nguyen and Edwin V. Bonilla. Automated variational inference for Gaussian process models. In *NIPS*. 2014.

Manfred Opper and Cédric Archambeau. The variational Gaussian approximation revisited. *Neural computation*, 21(3):786–792, 2009.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *JMLR*, 6:1939–1959, 2005.

Lawrence Rabiner and B Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1): 4–16, 1986.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *AISTATS*, 2014.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

Sara Robinson. Toward an optimal algorithm for matrix multiplication. *SIAM news*, 38(9):1–3, 2005.

Ed Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *NIPS*, 2006.

P. K. Srijith, P. Balamurugan, and Shirish Shevade. Efficient variational inference for Gaussian process structured prediction. In *NIPS Workshop on Advances in Variational Inference*, 2014.

P.K. Srijith, P. Balamurugan, and Shirish Shevade. Gaussian process pseudo-likelihood models for sequence labeling. In *ECML-PKDD 2016*, 2016.

Charles Sutton and Andrew McCallum. Piecewise pseudolikelihood for efficient training of conditional random fields. In *ICML*, 2007.

Charles Sutton, Andrew McCallum, et al. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373, 2012.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *NIPS*. 2004.

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *AISTATS*, 2009.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, December 2005.

Atro Voutilainen. Part-of-speech tagging. *The Oxford handbook of computational linguistics*, pages 219–232, 2003.

Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

Appendices

A Proof of Theorem 2

Here we proof the result that we can estimate the expected log likelihood and its gradients using expectations over low-dimensional Gaussians, that is that

Theorem. *For the structured* GP *model defined in this work the expected log likelihood over the given variational distribution and its gradients can be estimated using expectations over T_n-dimensional Gaussians and $|\mathcal{V}|^2$-dimensional Gaussians, where T_n is the length of each sequence and $|\mathcal{V}|$ is the vocabulary size.*

A.1 Estimation of $\mathcal{L}_{\text{ell}}$ in the full (non-sparse) model

For the $\mathcal{L}_{\text{ell}}$ we have that:

$$\mathcal{L}_{\text{ell}} = \left\langle \sum_{n=1}^{N_{\text{seq}}} \log p(\mathbf{y}_n|\mathbf{f}_n) \right\rangle_{q(\mathbf{f}^{\text{un}})q(\mathbf{f}^{\text{bin}})} \tag{24}$$

$$= \sum_{n=1}^{N_{\text{seq}}} \int_{\mathbf{f}^{\text{bin}}} \int_{\mathbf{f}^{\text{un}}} q(\mathbf{f}^{\text{un}})q(\mathbf{f}^{\text{bin}}) \log p(\mathbf{y}_n|\mathbf{f}_n)\ d\mathbf{f}^{\text{un}} d\mathbf{f}^{\text{bin}} \tag{25}$$

$$= \sum_{n=1}^{N_{\text{seq}}} \int_{\mathbf{f}^{\text{bin}}} \int_{\mathbf{f}_n^{\text{un}}} \int_{\mathbf{f}_{\setminus n}^{\text{un}}} q(\mathbf{f}_{\setminus n}^{\text{un}}|\mathbf{f}_n^{\text{un}})q(\mathbf{f}_n^{\text{un}})q(\mathbf{f}^{\text{bin}}) \log p(\mathbf{y}_n|\mathbf{f}_n)\ d\mathbf{f}_{\setminus n}^{\text{un}} d\mathbf{f}_n^{\text{un}} d\mathbf{f}^{\text{bin}} \tag{26}$$

$$= \sum_{n=1}^{N_{\text{seq}}} \langle \log p(\mathbf{y}_n|\mathbf{f}_n) \rangle_{q(\mathbf{f}_n^{\text{un}})q(\mathbf{f}^{\text{bin}})} \tag{27}$$

$$= \sum_{n=1}^{N_{\text{seq}}} \sum_{k=1}^{K} \pi_k \left\langle \log p(\mathbf{y}_n|\mathbf{f}_n) \right\rangle_{q_{kn}(\mathbf{f}_n^{\text{un}})q(\mathbf{f}^{\text{bin}})}, \tag{28}$$

where $q_{kn}(\mathbf{f}_n^{\text{un}})$ is a $(T_n \times |\mathcal{V}|)$-dimensional Gaussian with block-diagonal covariance $\mathbf{\Sigma}_{k(n)}$, each block of size $T_n \times T_n$. Therefore, we can estimate the above term by sampling from T_n-dimensional Gaussians independently. Furthermore, $q(\mathbf{f}^{\text{bin}})$ is a $|\mathcal{V}|^2$-dimensional Gaussian, which can also be sampled independently. In practice, we can assume that the covariance of $q(\mathbf{f}^{\text{bin}})$ is diagonal and we only sample from unary Gaussians for the pairwise functions. ■

A.2 Gradients

Taking the gradients of the kth term for the nth sequence in the $\mathcal{L}_{\text{ell}}$:

$$\mathcal{L}_{\text{ell}}^{(k,n)} = \langle \log p(\mathbf{y}_n|\mathbf{f}_n) \rangle_{q_{kn}(\mathbf{f}_n^{\text{un}})q(\mathbf{f}^{\text{bin}})} \tag{29}$$

$$= \int_{\mathbf{f}^{\text{bin}}} \int_{\mathbf{f}_n^{\text{un}}} q_{kn}(\mathbf{f}_n^{\text{un}})q(\mathbf{f}^{\text{bin}}) \log p(\mathbf{y}_n|\mathbf{f}_n)\ \mathrm{d}\mathbf{f}_n^{\text{un}} \mathrm{d}\mathbf{f}^{\text{bin}} \tag{30}$$

$$\nabla_{\boldsymbol{\lambda}_k^{\text{un}}} \mathcal{L}_{\text{ell}}^{(k,n)} = \int_{\mathbf{f}^{\text{bin}}} \int_{\mathbf{f}_n^{\text{un}}} q_{kn}(\mathbf{f}_n^{\text{un}})q(\mathbf{f}^{\text{bin}}) \log p(\mathbf{y}_n|\mathbf{f}_n) \nabla_{\boldsymbol{\lambda}_k^{\text{un}}} \log q_{kn}(\mathbf{f}_n^{\text{un}})\ \mathrm{d}\mathbf{f}_n^{\text{un}} \mathrm{d}\mathbf{f}^{\text{bin}} \tag{31}$$

$$= \left\langle \log p(\mathbf{y}_n|\mathbf{f}_n) \nabla_{\boldsymbol{\lambda}_k^{\text{un}}} \log q_{kn}(\mathbf{f}_n^{\text{un}}) \right\rangle_{q_{kn}(\mathbf{f}_n^{\text{un}})q(\mathbf{f}^{\text{bin}})}, \tag{32}$$

where we have used the fact that $\nabla_{\mathbf{x}} f(\mathbf{x}) = f(\mathbf{x}) \nabla_{\mathbf{x}} \log f(\mathbf{x})$ for any nonnegative function $f(\mathbf{x})$ Similarly. the gradients of the parameters of the distribution over binary functions can be

estimated using:

$$\nabla_{\boldsymbol{\lambda}^{\text{bin}}} \mathcal{L}_{\text{ell}}^{(k,n)} = \left\langle \log p(\mathbf{y}_n|\mathbf{f}_n) \nabla_{\boldsymbol{\lambda}^{\text{bin}}} \log q(\mathbf{f}^{\text{bin}}) \right\rangle_{q_{kn}(\mathbf{f}_n^{\text{un}}) q(\mathbf{f}^{\text{bin}})} . \tag{33}$$

■

B KL terms in the sparse model

The KL term ($\mathcal{L}_{\text{kl}}$) in the variational objective ($\mathcal{L}_{\text{elbo}}$) is composed of a KL divergence between the approximate posteriors and the priors over the inducing variables and pairwise functions:

$$\mathcal{L}_{\text{kl}} = \underbrace{-\text{KL}(q(\mathbf{u})\|p(\mathbf{u}))}_{\mathcal{L}_{\text{kl}}^{\text{un}}} \underbrace{-\text{KL}(q(\mathbf{f}^{\text{bin}})\|p(\mathbf{f}^{\text{bin}}))}_{\mathcal{L}_{\text{kl}}^{\text{bin}}}, \tag{34}$$

where, as the approximate posterior and the prior over the pairwise functions are Gaussian, the KL over pairwise functions can be computed analytically:

$$\mathcal{L}_{\text{kl}}^{\text{bin}} = -\text{KL}(q(\mathbf{f}^{\text{bin}})\|p(\mathbf{f}^{\text{bin}})) = -\text{KL}(\mathcal{N}(\mathbf{f}^{\text{bin}}; \mathbf{m}^{\text{bin}}, \mathbf{S}^{\text{bin}})\|\mathcal{N}(\mathbf{f}^{\text{bin}}; \mathbf{0}, \mathbf{K}^{\text{bin}})) \tag{35}$$

$$= -\frac{1}{2}\left(\log\left|\mathbf{K}^{\text{bin}}\right| - \log\left|\mathbf{S}^{\text{bin}}\right| + (\mathbf{m}^{\text{bin}})^T (\mathbf{K}^{\text{bin}})^{-1}\mathbf{m}^{\text{bin}} + \text{ tr } (\mathbf{K}^{\text{bin}})^{-1}\mathbf{S}^{\text{bin}} - |\mathcal{V}|\right). \tag{36}$$

For the distributions over the unary functions we need to compute a KL divergence between a mixture of Gaussians and a Gaussian. For this we consider the decomposition of the KL divergence as follows:

$$\mathcal{L}_{\text{kl}}^{\text{un}} = -\text{KL}(q(\mathbf{u})\|p(\mathbf{u})) = \underbrace{\mathbb{E}_q[-\log q(\mathbf{u})]}_{\mathcal{L}_{\text{ent}}} + \underbrace{\mathbb{E}_q[\log p(\mathbf{u})]}_{\mathcal{L}_{\text{cross}}}, \tag{37}$$

where the entropy term ($\mathcal{L}_{\text{ent}}$) can be lower bounded using Jensen's inequality:

$$\mathcal{L}_{\text{ent}} \geq -\sum_{k=1}^{K} \pi_k \log \sum_{\ell=1}^{K} \pi_\ell \mathcal{N}(\mathbf{m}_k; \mathbf{m}_\ell, \mathbf{S}_k + \mathbf{S}_\ell) \stackrel{\text{def}}{=} \hat{\mathcal{L}}_{\text{ent}}. \tag{38}$$

and the negative cross-entropy term ($\mathcal{L}_{\text{cross}}$) can be computed exactly:

$$\mathcal{L}_{\text{cross}} = -\frac{1}{2}\sum_{k=1}^{K} \pi_k \sum_{j=1}^{|\mathcal{V}|} [M \log 2\pi + \log|\kappa(\mathbf{Z}_j, \mathbf{Z}_j)| + \mathbf{m}_{kj}^T \kappa(\mathbf{Z}_j, \mathbf{Z}_j)^{-1}\mathbf{m}_{kj} + \text{ tr } \kappa(\mathbf{Z}_j, \mathbf{Z}_j)^{-1}\mathbf{S}_{kj}]. \tag{39}$$

C Proof of Theorem 3

To prove Theorem 3 we will express the expected log likelihood term in the same form as that given in Equation (28), showing that the resulting $q_{kn}(\mathbf{f}_n^{\text{un}})$ is also a $(T_n \times |\mathcal{V}|)$-dimensional Gaussian with block-diagonal covariance, having $|\mathcal{V}|$ blocks each of dimensions $T_n \times T_n$. We start by taking the given $\mathcal{L}_{\text{ell}}$, where the expectations are over the joint posterior $q(\mathbf{f}, \mathbf{u}|\boldsymbol{\lambda}) = p(\mathbf{f}^{\text{un}}|\mathbf{u})q(\mathbf{u})q(\mathbf{f}^{\text{bin}})$:

$$\mathcal{L}_{\text{ell}} = \left\langle \sum_{n=1}^{N_{\text{seq}}} \log p(\mathbf{y}_n|\mathbf{f}_n) \right\rangle_{p(\mathbf{f}^{\text{un}}|\mathbf{u})q(\mathbf{u})q(\mathbf{f}^{\text{bin}})} \tag{40}$$

$$= \int_{\mathbf{f}} \log p(\mathbf{y}|\mathbf{f}) \underbrace{\int_{\mathbf{u}} q(\mathbf{u}) p(\mathbf{f}^{\text{un}}|\mathbf{u}) \mathrm{d}\mathbf{u}}_{q(\mathbf{f}^{\text{un}})} q(\mathbf{f}^{\text{bin}}) \mathrm{d}\mathbf{f}, \tag{41}$$

where our our approximating distribution is:

$$q(\mathbf{f}) = q(\mathbf{f}^{\text{un}})q(\mathbf{f}^{\text{bin}}) \tag{42}$$

$$q(\mathbf{f}^{\text{un}}) = \int_{\mathbf{u}} q(\mathbf{u}) p(\mathbf{f}^{\text{un}}|\mathbf{u}) \mathrm{d}\mathbf{u}, \tag{43}$$

which can be computed analytically:

$$q(\mathbf{f}^{\text{un}}) = \sum_{k=1}^{K} \pi_k q_k(\mathbf{f}^{\text{un}}) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{|\mathcal{V}|} \mathcal{N}(\mathbf{f}_j^{\text{un}}; \mathbf{b}_{kj}, \boldsymbol{\Sigma}_{kj}) \tag{44}$$

$$\mathbf{b}_{kj} = \mathbf{A}_j \mathbf{m}_{kj} \tag{45}$$

$$\boldsymbol{\Sigma}_{kj} = \widetilde{\mathbf{K}}_j + \mathbf{A}_j \mathbf{S}_{kj} \mathbf{A}_j^T. \tag{46}$$

We note in Equation (44) that $q_k(\mathbf{f}^{\text{un}})$ has a block diagonal structure, which implies that we have the same expression for the $\mathcal{L}_{\text{ell}}$ as in Equation (28). Therefore, we obtain analogous estimates:

$$\mathcal{L}_{\text{ell}} = \sum_{n=1}^{N_{\text{seq}}} \sum_{k=1}^{K} \pi_k \left\langle \log p(\mathbf{y}_n|\mathbf{f}_n) \right\rangle_{q_{kn}(\mathbf{f}_n^{\text{un}})q(\mathbf{f}^{\text{bin}})}, \tag{47}$$

Here, as before, $q_{kn}(\mathbf{f}_n^{\text{un}})$ is a $(T_n \times |\mathcal{V}|)$–dimensional Gaussian with block-diagonal covariance $\boldsymbol{\Sigma}_{k(n)}$, each block of size $T_n \times T_n$. The main difference in this (sparse) case is that $\mathbf{b}_{k(n)}$ and $\boldsymbol{\Sigma}_{k(n)}$ are constrained by the expressions in Equations (45) and (46). Hence, the proof for the gradients follows the same derivation as in §A.2 above. ■

D Gradients of $\mathcal{L}_{\text{elbo}}$ for sparse model

Here we give the gradients of the variational objective wrt the parameters for the variational distributions over the inducing variables, pairwise functions and hyper-parameters.

D.1 Inducing variables

D.1.1 KL term

As the structured likelihood does not affect the KL divergence term, the gradients corresponding to this term are similar to those in the non-structured case (Dezfouli and Bonilla, 2015). Let $\mathbf{K}_{zz}$ be the block-diagonal covariance with $|\mathcal{V}|$ blocks $\kappa(\mathbf{Z}_j, \mathbf{Z}_j)$, $j = 1, \dots Q$. Additionally, lets assume the following definitions:

$$\mathbf{C}_{kl} \stackrel{\text{def}}{=} \mathbf{S}_k + \mathbf{S}_\ell, \tag{48}$$

$$\mathcal{N}_{k\ell} \stackrel{\text{def}}{=} \mathcal{N}(\mathbf{m}_k; \mathbf{m}_\ell, \mathbf{C}_{kl}), \tag{49}$$

$$z_k \stackrel{\text{def}}{=} \sum_{\ell=1}^{K} \pi_\ell \mathcal{N}_{k\ell}. \tag{50}$$

The gradients of $\mathcal{L}_{\text{kl}}$ wrt the posterior mean and posterior covariance for component k are:

$$\nabla_{\mathbf{m}_k} \mathcal{L}_{\text{cross}} = -\pi_k \mathbf{K}_{zz}^{-1} \mathbf{m}_k, \tag{51}$$

$$\nabla_{\mathbf{S}_k} \mathcal{L}_{\text{cross}} = -\frac{1}{2} \pi_k \mathbf{K}_{zz}^{-1} \tag{52}$$

$$\nabla_{\pi_k} \mathcal{L}_{\text{cross}} = -\frac{1}{2} \sum_{j=1}^{|\mathcal{V}|} [M \log 2\pi + \log |\kappa(\mathbf{Z}_j, \mathbf{Z}_j)| + \mathbf{m}_{kj}^T \kappa(\mathbf{Z}_j, \mathbf{Z}_j)^{-1} \mathbf{m}_{kj} + \text{ tr } \kappa(\mathbf{Z}_j, \mathbf{Z}_j)^{-1} \mathbf{S}_{kj}], \tag{53}$$

where we note that we compute $\mathbf{K}_{zz}^{-1}$ by inverting the corresponding blocks $\kappa(\mathbf{Z}_j, \mathbf{Z}_j)$ independently. The gradients of the entropy term wrt the variational parameters are:

$$\nabla_{\mathbf{m}_k} \hat{\mathcal{L}}_{\text{ent}} = \pi_k \sum_{\ell=1}^{K} \pi_\ell \left(\frac{\mathcal{N}_{k\ell}}{z_k} + \frac{\mathcal{N}_{k\ell}}{z_\ell} \right) \mathbf{C}_{kl}^{-1} (\mathbf{m}_k - \mathbf{m}_\ell), \tag{54}$$

$$\nabla_{\mathbf{S}_k} \hat{\mathcal{L}}_{\text{ent}} = \frac{1}{2} \pi_k \sum_{\ell=1}^{K} \pi_\ell \left(\frac{\mathcal{N}_{k\ell}}{z_k} + \frac{\mathcal{N}_{k\ell}}{z_\ell} \right) \left[\mathbf{C}_{kl}^{-1} - \mathbf{C}_{kl}^{-1} (\mathbf{m}_k - \mathbf{m}_\ell)(\mathbf{m}_k - \mathbf{m}_\ell)^T \mathbf{C}_{kl}^{-1} \right], \tag{55}$$

$$\nabla_{\pi_k} \hat{\mathcal{L}}_{\text{ent}} = -\log z_k - \sum_{\ell=1}^{K} \pi_\ell \frac{\mathcal{N}_{k\ell}}{z_\ell}.$$

D.1.2 Expected log likelihood term

Retaking the gradients in the full model in Equation (32), we have that:

$$\nabla_{\boldsymbol{\lambda}_k^{\text{un}}} \mathcal{L}_{\text{ell}}^{(k,n)} = \left\langle \log p(\mathbf{y}_n|\mathbf{f}_n) \nabla_{\boldsymbol{\lambda}_k^{\text{un}}} \log q_{kn}(\mathbf{f}_n^{\text{un}}) \right\rangle_{q_{kn}(\mathbf{f}_n^{\text{un}}) q(\mathbf{f}^{\text{bin}})}, \tag{56}$$

where the variational parameters $\boldsymbol{\lambda}_k^{\text{un}}$ are the posterior means and covariances ($\{\mathbf{m}_{kj}\}$ and $\{\mathbf{S}_{kj}\}$) of the inducing variables. As given in Equation (44), $q_k(\mathbf{f}^{\text{un}})$ factorizes over the latent process ($j = 1, \ldots, |\mathcal{V}|$), so do the marginals $q_{kn}(\mathbf{f}_n^{\text{un}})$, hence:

$$\nabla_{\boldsymbol{\lambda}_k^{\text{un}}} \log q_{kn}(\mathbf{f}_n^{\text{un}}) = \nabla_{\boldsymbol{\lambda}_k^{\text{un}}} \sum_{j=1}^{|\mathcal{V}|} \log \mathcal{N}(\mathbf{f}_{nj}^{\text{un}}; \mathbf{b}_{kjn}, \boldsymbol{\Sigma}_{kjn}), \tag{57}$$

where each of the distributions in Equation (57) is a T_n–dimensional Gaussian. Let us assume the following definitions:

$$\mathbf{X}_n : \text{all feature vectors corresponding to sequence } n \tag{58}$$

$$\mathbf{A}_{jn} \stackrel{\text{def}}{=} \kappa(\mathbf{X}_n, \mathbf{Z}_j) \kappa(\mathbf{Z}_j, \mathbf{Z}_j)^{-1} \tag{59}$$

$$\widetilde{\mathbf{K}}_j^n \stackrel{\text{def}}{=} \kappa_j(\mathbf{X}_n, \mathbf{X}_n) - \mathbf{A}_{jn} \kappa(\mathbf{Z}_j, \mathbf{X}_n), \text{ therefore:} \tag{60}$$

$$\mathbf{b}_{kjn} = \mathbf{A}_{jn} \mathbf{m}_{kj}, \tag{61}$$

$$\boldsymbol{\Sigma}_{kjn} = \widetilde{\mathbf{K}}_j^n + \mathbf{A}_{jn} \mathbf{S}_{kj} \mathbf{A}_{jn}^T. \tag{62}$$

Hence, the gradients of $\log q_k(\mathbf{f}^{\text{un}})$ wrt the the variational parameters of the unary posterior distributions over the inducing points are:

$$\nabla_{\mathbf{m}_{kj}} \log q_{kn}(\mathbf{f}_n^{\text{un}}) = \mathbf{A}_{jn}^T \boldsymbol{\Sigma}_{kjn}^{-1} \left(\mathbf{f}_{nj}^{\text{un}} - \mathbf{b}_{kjn} \right), \tag{63}$$

$$\nabla_{\mathbf{S}_{kj}} \log q_{kn}(\mathbf{f}_n^{\text{un}}) = \frac{1}{2} \mathbf{A}_{jn}^T \left[\boldsymbol{\Sigma}_{kjn}^{-1} (\mathbf{f}_{nj}^{\text{un}} - \mathbf{b}_{kjn})(\mathbf{f}_{nj}^{\text{un}} - \mathbf{b}_{kjn})^T \boldsymbol{\Sigma}_{kjn}^{-1} - \boldsymbol{\Sigma}_{kjn}^{-1} \right] \mathbf{A}_{jn} \tag{64}$$

Therefore, the gradients of $\mathcal{L}_{\text{ell}}$ wrt the parameters of the distributions over unary functions are:

$$\nabla_{\mathbf{m}_{kj}} \mathcal{L}_{\text{ell}} = \frac{\pi_k}{S} \kappa(\mathbf{Z}_j, \mathbf{Z}_j)^{-1} \sum_{n=1}^{N_{\text{seq}}} \kappa(\mathbf{Z}_j, \mathbf{X}_n) (\boldsymbol{\Sigma}_{kjn})^{-1} \sum_{i=1}^{S} (\mathbf{f}_{nkij}^{\text{un}} - \mathbf{b}_{kjn}) \log p(\mathbf{y}_n | \mathbf{f}_{nki}^{\text{un}}, \mathbf{f}_i^{\text{bin}}), \tag{65}$$

$$\begin{aligned} \nabla_{\mathbf{S}_{kj}} \mathcal{L}_{\text{ell}} = \frac{\pi_k}{2S} \sum_{n=1}^{N_{\text{seq}}} \mathbf{A}_{jn}^T \Big\{ \sum_{i=1}^{S} \big[(\boldsymbol{\Sigma}_{kjn})^{-1} (\mathbf{f}_{nkij}^{\text{un}} - \mathbf{b}_{kjn}) ((\mathbf{f}_{nj}^{\text{un}})^{(k,i)} - \mathbf{b}_{kjn})^T (\boldsymbol{\Sigma}_{kjn})^{-1} \\ - (\boldsymbol{\Sigma}_{kjn})^{-1} \big] \log p(\mathbf{y}_n | \mathbf{f}_{nki}^{\text{un}}, \mathbf{f}_i^{\text{bin}}) \Big\} \mathbf{A}_{jn} \end{aligned} \tag{66}$$

D.1.3 Pairwise functions

The gradients of the $\mathcal{L}_{\text{kl}}^{\text{bin}}$ wrt the parameters of the posterior over pairwise functions are given by:

$$\nabla_{\mathbf{m}^{\text{bin}}} \mathcal{L}_{\text{kl}}^{\text{bin}} = -(\mathbf{K}^{\text{bin}})^{-1}\mathbf{m}^{\text{bin}} \tag{67}$$

$$\nabla_{\mathbf{S}^{\text{bin}}} \mathcal{L}_{\text{kl}}^{\text{bin}} = \frac{1}{2}\left((\mathbf{S}^{\text{bin}})^{-1} - (\mathbf{K}^{\text{bin}})^{-1}\right) \tag{68}$$

The gradients of the $\mathcal{L}_{\text{ell}}$ wrt the parameters of the posterior over pairwise functions are given by:

$$\nabla_{\mathbf{m}^{\text{bin}}} \mathcal{L}_{\text{ell}} = \frac{1}{S}\sum_{n=1}^{N_{\text{seq}}}\sum_{k=1}^{K}\pi_k \sum_{i=1}^{S}(\mathbf{S}^{\text{bin}})^{-1}(\mathbf{f}_i^{\text{bin}} - \mathbf{m}^{\text{bin}}) \log p(\mathbf{y}_n|\mathbf{f}_{nki}^{\text{un}}, \mathbf{f}_i^{\text{bin}})$$

$$\nabla_{\mathbf{S}^{\text{bin}}} \mathcal{L}_{\text{ell}} = \frac{1}{2S}\sum_{n=1}^{N_{\text{seq}}}\sum_{k=1}^{K}\pi_k \sum_{i=1}^{S}[(\mathbf{S}^{\text{bin}})^{-1}(\mathbf{f}_i^{\text{bin}} - \mathbf{m}^{\text{bin}})(\mathbf{f}_i^{\text{bin}} - \mathbf{m}^{\text{bin}})^T(\mathbf{S}^{\text{bin}})^{-1} - (\mathbf{S}^{\text{bin}})^{-1}]\cdot$$
$$\log p(\mathbf{y}_n|\mathbf{f}_{nki}^{\text{un}}, \mathbf{f}_i^{\text{bin}})$$

E Experiments

E.1 Experimental set-up

Before starting all experiments, our program selects the positions of the 500 inducing points by performing K-means clustering on the training data. The initial values of the means (one mean value for every inducing point p and for every possible label l) are set as the fraction of the points of the training set in the cluster whose centroid is p that belong to label l.

As described in Algorithm 1, we adaptively choose the correct learning rates for all parameters by searching (beginning from an initial guess, and doubling or dividing it by two as needed) for the biggest step size that causes the objective to decrease over twenty stochastic optimization steps. The step size to use for the given parameter is taken as one fourth of this value, to avoid instability. The initial step sizes are 0.5 for unary mean parameters, 0.0005 for binary mean parameters, 0.005 for unary covariance parameters, 0.0005 for binary covariance parameters and 1.0 for (linear) kernel hyperparameters (these values were selected only to speed up, insofar as possible, the process of parameter search); and the optimal step sizes are found in the order unary mean → unary covariance → binary mean → binary covariance → kernel hyperparameter. Of course this is not an exhaustive grid search, but it is less computationally expensive and works well in practice.

Algorithm 1 Method for computing the step size

```
function CHECK_STEPSIZE(parameters, step_size, num_to_check=20)
    to_check ← num_to_check sentences at random from the training set;
    old_obj ← current value of the objective function;
    for i ← 0 ... num_to_check do
        grad ← gradient of objective wrt parameters;
        parameters ← parameters - step_size · grad;
        if parameters are out of bounds then
            return False;
    new_obj ← current value of the objective function;
    if new_obj < old_obj then
        return True;
    else
        return False;
function SEARCH_STEPSIZE_UP(parameters, initial_step_size)
    step_size ← initial_step_size;
    old_params ← current parameter values of the model;
    while True do
        is_good ← CHECK_STEPSIZE(parameters, step_size);
        if is_good = False then
            parameters ← old_params;
            return step_size / factor^2;
        step_size ← step_size · factor;
function SEARCH_STEPSIZE_DOWN(parameters, initial_step_size)
    step_size ← initial_step_size;
    old_params ← current parameter values of the model;
    while True do
        is_good ← CHECK_STEPSIZE(parameters, step_size);
        if is_good = True then
            parameters ← old_params;
            return step_size / factor;
        step_size ← step_size/factor;
function CHOOSE_STEPSIZE(parameters, initial_step_size, factor = 2)
    step_size ← SEARCH_STEPSIZE_UP(parameters, initial_step_size, factor);
    if step_size == initial_step_size / factor^2 then
        step_size ← SEARCH_STEPSIZE_DOWN(parameters, initial_step_size,
factor);
    return step_size/factor;
```

E.2 Optimization

We optimize the three sets of parameters (unaries, binaries, hyperparameters) in a global loop, in the same order as mentioned earlier, through standard Stochastic Gradient Descent, until the time limit is reached. We use 4000 new random samples each step for the estimation of the relevant gradients and averages.[9]

For the small-scale experiments, the variational parameters for unary nodes are optimized for 500 iterations, variational parameters for pairwise nodes are optimized for 100 iterations, and hyper-parameters are updated for 20 iterations.

We keep a tight bound (10 in the unary case, 20 in the binary one) on the maximum possible absolute values that covariance parameters can take. If an update would bring their value beyond it, we recompute the gradients with new samples: oftentimes, this resolves the issue (in brief, because the faulty update was due to a bad estimation of the gradients). If the problem persists, we disregard the current sentence and move on to the next one; and if after ten attempts the problem still persists, we move back to the latest "safe" position that did not cause out-of-bound errors. In this way, we can use a relatively small number of samples and maintain rather aggressive step sizes while recovering neatly from out-of-bounds errors.

The setting for the large-scale experiments was similar, except that the optimization schedule was $(12500, 2500, 500)$ (that is 12500 unary optimization steps, 2500 binary ones and 500 hyperparameter ones) for the big BASE NP and CHUNKING experiments. All the experiments were run for four hours, not counting initial clustering or final prediction (but counting step size selection). For comparison, the (small-scale) GP-ESS experiments, which were run for 250,000 elliptical slice sampling steps, took on average 11.34 hours for CHUNKING, 21.19 hours for BASE NP, 2.07 hours for SEGMENTATION and 15.75 for JAPANESE NE.

E.3 Performance profiles

Figure 5 shows the performance of our algorithm as a function of time. We see that the test likelihood decreases very regularly in all the folds and so does overall the error rate, albeit with more variability. The bulk of the optimization, both with respect to the test likelihood and with respect to the error rate, occurs during the first 120 minutes. This suggests that the kind of approach described in this paper might be particularly suited for cases in which expected loglikelihood of the prediction and speed of convergence are priorities.

[9] When computing the gradient of the kernel hyperparameter, 8000 samples were used instead to insure greater stability.

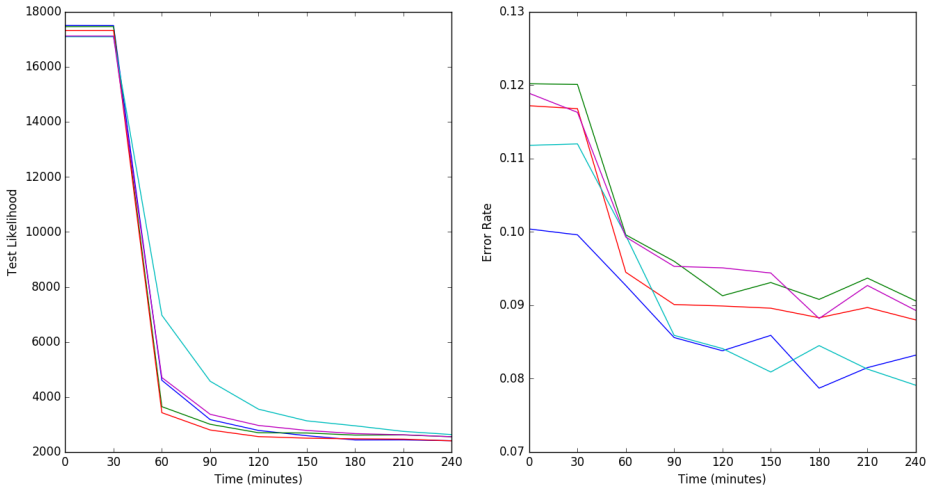

Figure 5: The test performance of GP-VAR-T on CHUNKING for the large scale experiment as a function of time.