Scalable Gaussian Process Structured Prediction for Grid Factor Graph Applications

Sébastien Bratières[2] SEBASTIEN@CANTAB.NET
Novi Quadrianto[1,2] N.QUADRIANTO@SUSSEX.AC.UK
Sebastian Nowozin[3] SEBASTIAN.NOWOZIN@MICROSOFT.COM
Zoubin Ghahramani[2] ZOUBIN@ENG.CAM.AC.UK

[1]SMiLe CLiNiC, Department of Informatics, University of Sussex, UK
[2]Machine Learning Group, Department of Engineering, University of Cambridge, UK
[3]Microsoft Research, Cambridge, UK

Abstract

Structured prediction is an important and well-studied problem with many applications across machine learning. GPstruct is a recently proposed structured prediction model that offers appealing properties such as being kernelised, non-parametric, and supporting Bayesian inference (Bratières et al., 2013). The model places a Gaussian process prior over energy functions which describe relationships between input variables and structured output variables. However, the memory demand of GPstruct is quadratic in the number of latent variables and training runtime scales cubically. This prevents GPstruct from being applied to problems involving grid factor graphs, which are prevalent in computer vision and spatial statistics applications.

Here we explore a scalable approach to learning GPstruct models based on ensemble learning, with weak learners (predictors) trained on subsets of the latent variables and bootstrap data, which can easily be distributed. We show experiments with 4M latent variables on image segmentation. Our method outperforms widely-used conditional random field models trained with pseudo-likelihood. Moreover, in image segmentation problems it improves over recent state-of-the-art marginal optimisation methods in terms of predictive performance and uncertainty calibration. Finally, it generalises well on all training set sizes.

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

1. Introduction

Conditional random fields (CRF) (Lafferty et al., 2001) and Markov random fields (MRF) (Blake et al., 2011) are popular models in computer vision, language modelling, and other applications of machine learning, because they allow specifying rich interactions between multiple random variables using an undirected graph. Despite their success these models remain challenging to work with: both inference and parameter estimation for general graph structures are intractable and require approximations. Furthermore, in order to achieve good predictive performance one needs to specify meaningful features and has to trade-off the resulting model capacity with the amount of available training data to avoid overfitting.

This work addresses the specification and estimation problems in a Bayesian framework building on a recent non-parametric model—the *GPstruct model* (Bratières et al., 2013). Our contribution is to present an efficient approximate Bayesian learning approach that reliably prevents overfitting yet remains scalable for large general (non-tree structured) graphs. An important limitation preventing the application of GPstruct to larger data sets is the dimension of the kernel matrix, determined by the number of GPstruct latent variables, which is *number of training points*$\times$*output dimensionality*$\times$*label space cardinality*. While this number may remain reasonable for linear chain factor graphs (Bratières et al., 2013), the output dimensionality in vision problems is vastly higher than in typical language processing applications. As an example, this paper tackles an 8-class segmentation task, training over 572 training images of size 50×150. In the standard GPstruct approach, this yields a kernel matrix of size 4 million by 4 million ($\sim 10^{13}$ elements, assuming a block diagonal unary kernel matrix as in (Bratières et al., 2013)), which cannot be inverted using Gaussian process (GP) sparsification techniques such as (Snelson & Ghahramani, 2005) alone. In practice, we

find the standard GPstruct model to be applicable to only about one image at a time, which is consistent with the scalability results in (Hensman et al., 2013). On the other hand, for these computer vision applications the GPstruct model proposed in this paper provides reliable uncertainty estimates that are well calibrated. Because in many real world domains, computer vision is part of a larger autonomous system, we believe that providing reliably quantified uncertainty is an important feature of our method.

Our paper introduces a scalable approximate Bayesian learning method. The latent variables are divided into subsets, each of which is assigned to a GPstruct predictor. The predictors are trained on bootstrap data using the pseudo-likelihood approximation. The above problem is reduced to subsets of 40000 latent variables, distributed over a computing cluster, with slave nodes carrying out partial prediction and the master node aggregating predictions. Our approach makes it possible to apply the model to prediction on grid factor graphs and potentially other structures.

2. The GPstruct Model

For self-consistency of the paper, we review the Gaussian process structured prediction model (Bratières et al., 2013). Assume that we are given a set of N input-output data points or examples $\mathcal{D} = \{(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^N, \mathbf{y}^N)\} \subset \mathcal{X} \times \mathcal{Y}$. We use $\mathcal{X}$ and $\mathcal{Y}$ to denote the input and output space, respectively. We consider *structured* output: the output domain is the product of individual variable domains $\mathcal{Y}_i$ so that $\mathcal{Y} = \bigtimes_{i=1,\ldots,I} \mathcal{Y}_i$ where we use I to denote the number of output variables associated with an input. Note importantly that as in other typical structured prediction problems for discrete $\mathcal{Y}_i$, the size of $\mathcal{Y}$ grows exponentially with I.

We describe the relationship between an input variable $\mathbf{x}$ and its output variable $\mathbf{y}$ by means of an *energy function* E. The energy function defines a conditional probability distribution $\Pr(\mathbf{y}|\mathbf{x}, \mathbf{E})$ as

$$\Pr(\mathbf{y}|\mathbf{x}, \mathbf{E}) = \frac{1}{Z(\mathbf{x}, \mathbf{E})} \exp(-E(\mathbf{x}, \mathbf{y})), \qquad (1)$$

where the log partition function $Z(\mathbf{x}, \mathbf{E}) = \sum_{\mathbf{y} \in \mathcal{Y}} \exp(-E(\mathbf{x}, \mathbf{y}))$ is the normalising constant. Note that our model in (1) is in the general form of a conditional random field (CRF) (Lafferty et al., 2001; Sutton & McCallum, 2012).

We will consider the case in which the energy function E decomposes into a sum of energy functions E_{t_F} over *factors* F, where F defines a subset of variables. As an illustration, when $|F| = 2$, the energy function is:

$$E(\mathbf{x}, \mathbf{y}) = \sum_{F \in \mathcal{F}} E_{t_F}(\mathbf{x}_F, \mathbf{y}_F). \qquad (2)$$

In the above, we use $\mathbf{x}_F$ to denote the collection $(\mathbf{x}_i)_{i \in F}$, and likewise we use $\mathbf{y}_F$ to denote the parts of $\mathbf{y}$ that are in F. While there are many different subsets in $\mathcal{F}$, we assume that there are only few distinct *types* and we use t_F to denote the type of the factor F. The function E_{t_F} is the same for all factors of that type, but it acts on different variables. This notion of type specifies both repeated structure and *parameter tying* in the model. As another example, we can further decompose the energy function in (2). This results in two different types of factors: a) unary[1] factors connecting components of input and output variables $(\mathbf{x}_i, y_i)$, and b) pairwise factors connecting neighbouring output variables (y_i, y_j). We then have

$$E(\mathbf{x}, \mathbf{y}) = \sum_{(\mathbf{x}, y_i) \in t_{F_a}} E_a(\mathbf{x}_i, y_i) + \sum_{(y_i, y_j) \in t_{F_b}} E_b(y_i, y_j). \qquad (3)$$

The traditional CRF parameterisation (e.g., Sutton & McCallum (2012)) consists of expressing the energy function $E(\mathbf{x}, \mathbf{y})$ as a weighted sum of features $\mathbf{w}^\top \phi(\mathbf{x}, \mathbf{y})$ where $\mathbf{w} \in \mathbb{R}^D$ is the weight vector and $\phi(\mathbf{x}, \mathbf{y})$ is a feature function. Learning consists of estimating the weight vector under a given prior distribution. This approach is called *parametric*, since the parameterisation (e.g. the size of the weight vector) is fixed independently of the data. The GPstruct model employs a *non-parametric* approach by considering the energy terms $E_a(\mathbf{x}_i, y_i)$ and $E_b(y_i, y_j)$ as latent functions, and imposing a Gaussian process prior (Rasmussen & Williams, 2006) upon them. In the usual GP terminology, $\Pr(\mathbf{y}|\mathbf{x}, \mathbf{E})$ defined in equation 1 is the likelihood function.

This requires defining a covariance function over any two such energy functions. Following Bratières et al. (2013), which elaborates on this aspect, we define $\mathrm{Cov}(E_a(\cdot), E_b(\cdot))$ to be zero, and further

$$\mathrm{Cov}(E_a(\mathbf{x}_i, y_i), E_a(\mathbf{x}'_i, y'_i)) = k_a((\mathbf{x}_i, y_i), (\mathbf{x}'_i, y'_i))$$
$$\mathrm{Cov}(E_b(y_i, y''_i), E_b(y'_i, y'''_i)) = k_b((y, y''), (y', y'''))$$

We use $k_a(\cdot, \cdot)$ and $k_b(\cdot, \cdot)$ to denote a positive definite kernel function (Schölkopf & Smola, 2001). By defining that $E_a(\mathbf{x}_i, y_i)$ depends on the location i while $E_b(y_i, y''_i)$ does not, the number of unary latent variables grows with the data, while the number of pairwise latent variables does not. The non-parametric property, which allows model complexity to be determined as part of analysing the data, although appealing, introduces a serious scalability issue in the linear chain factor graph (Bratières et al., 2013). Unfortunately, this scalability issue is magnified in the vision applications in which we are interested, and which involve grid factor graphs: the number of unary latent variables grows with I, the number of pixels in an image. The same

[1] These are unary because in our supervised learning setting, we assume $\mathbf{x}$ to be given.

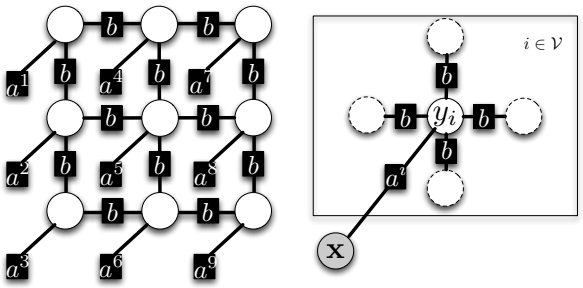

Figure 1. **Left**: Grid factor graph with a pairwise 4-connected factor. There is one unary factor type per pixel, and one pairwise factor type. **Right**: A plate notation for a variable y_i with its 4 connected neighbours corresponding to a pixel i. The unary factor a_i depends on image data $\mathbf{x}$, and the pairwise factors b are data independent.

scalability issue affects kernel CRFs (Lafferty et al., 2004), which may be one of the reasons why these models have not become popular.

Next, we will describe the GPstruct model for grid factor graphs before addressing our approach to scaling up the model.

3. GPstruct for Grid Factor Graphs

Assume that we are given an observed image $\mathbf{x} \in \mathcal{X}$ where $\mathcal{X}$ denotes the set of possible images. We note the pixels $\mathbf{x}_i$, $i = 1, \ldots, I$, so that I is the number of pixels of an image. Our goal is to infer a discrete labelling $\mathbf{y} \in \mathcal{Y}$ where the labelling is per-pixel, that is $\mathbf{y} = (y_i)_{i=1,\ldots,I}$, $y_i \in \mathcal{Y}_i$, and in practice, we will mostly have a single set $\mathcal{Y}_i := \mathcal{L}$ of labels.

We assume that the energy functions decompose into sums of energy functions over unary and pairwise factors as in (3). To be specific, consider a toy image of size 3 by 3. Figure 1 displays the corresponding factor graph model, with a distinct unary factor for each pixel, and shared pairwise factors over inter-pixel edges. All unary factors depend on image data $\mathbf{x}$, and all pairwise factors do not depend on $\mathbf{x}$.

Learning Given training data $\mathcal{D}$ consisting of N images and their labels, the goal of learning is to infer a posterior distribution over energy functions, that is $\Pr(\mathbf{E}|\mathcal{D})$. Assuming the images in the dataset are independent and identically distributed given the model, the training data likelihood can be written as $\Pr(\mathcal{D}|\mathbf{E}) = \prod_{n=1}^{N} \Pr(\mathbf{y}^n|\mathbf{x}^n, \mathbf{E})$, where $\Pr(\mathbf{y}^n|\mathbf{x}^n, \mathbf{E})$ is given in (1).

As for most interesting Bayesian models, analytic or exact computations are generally not possible, and adequate algorithms are needed in several places. Generating posterior samples $\mathbf{E}|\mathcal{D}$ from the GP, given the non-Gaussian likelihood, cannot be done analytically and is dealt with by elliptical slice sampling (ESS) (Murray et al., 2010), an efficient Markov Chain Monte Carlo method for tightly coupled latent variables with a Gaussian prior, as used in Bratières et al. (2013).

The likelihood computation is intractable due to the normalising constant $Z(\mathbf{x}, \mathbf{E})$, which involves a summation over the label set $\mathcal{Y}$, whose size is exponential in the number of pixels in an image. Take as a running illustration, a simple foreground-background segmentation task ($\mathcal{L} = \{1, 2\}$) and an image of size 50×100. For this problem, the exact likelihood would be computed using the junction tree algorithm, whose complexity is exponential in the treewidth of the grid graph, i.e. 50 in our example. To address the intractability of the normaliser, we use a surrogate likelihood, the pseudo-likelihood (Besag, 1975) (PL), as a drop-in for the true likelihood in the ESS procedure. The PL is derived from the per variable conditional distributions $\Pr(y_i|\mathbf{y}_{\mathcal{N}(i)}, \mathbf{x}, \mathbf{E})$. We use $\mathcal{N}(i)$ to denote the set of neighbours of variable i according to the underlying factor graph. If a grid factor graph with pairwise factors is used, $|\mathcal{N}(i)| = 4$ except for variable i associated with a pixel at the boundary. The PL is then defined as $\Pr_{\text{PL}}(\mathcal{D}|\mathbf{E}) = \prod_{n=1}^{N} \prod_{i \in \mathcal{V}} \Pr(y_i^n|\mathbf{y}_{\mathcal{N}(i)}, \mathbf{x}^n, \mathbf{E})$. The maximum PL estimator is known to be a consistent estimator (Besag, 1975), but here we use it to approximate the intractable likelihood function itself. The use of PL in MCMC schemes for Bayesian parameter learning in Markov random fields dates back to Wang et al. (2000).

A detailed study of MCMC for Bayesian learning in non-trivial undirected graphical models is given in Murray & Ghahramani (2004), which conjectures that for general undirected models, there are no tractable MCMC methods that give the correct equilibrium distribution over parameters. Their pragmatic solution is to explore a variety of approximations for the normalising constant $Z(\mathbf{x}, \mathbf{E})$. The method of Murray et al. (2006) is not exactly suited to our setting, since it requires pure Metropolis-Hastings as the sampling algorithm (and works on the acceptance rate), while we use elliptical slice sampling to avoid high sample correlation, in the presence of tightly coupled latent variables. However, earlier, Parise & Welling (2005) have concluded that for fully observed MRFs (as in our case), PL is recommended over perfect sampling due to the computational burden of the latter, which is not balanced by a corresponding performance gain. In this paper, we show empirically that PL works well when used as a likelihood approximation in the GPstruct model.

For GPstruct, the PL computation will still involve $10,000$ latent variables (50×100 pixels times 2 classes) in our illustrative binary segmentation task. In addition, this high number of latent variables requires large storage for the

GP kernel matrix. These two issues are addressed with one strategy, namely reducing the set of latent variables assigned to a weak learner (WL) by an ensemble method, an approach detailed in section 4.

To implement this method, as in Nowozin et al. (2011), PL is computed on subsets of pixels, $\mathcal{V}' \subset \mathcal{V}$, where $\mathcal{V}$ is the set of pixels of an image (and $|\mathcal{V}| = I$), while retaining the 4-connected factor for all $i \in \mathcal{V}'$. Referring to the plate notation in figure 1 (right), the variable y_i inside the plate is now repeated $|\mathcal{V}'|$ times. This gives a subset PL function $\hat{\Pr}_{\text{PL}}(\mathcal{D}|\mathbf{E}) = \prod_{n=1}^{N} \prod_{i \in \mathcal{V}'} \Pr(y_i^n | \mathbf{y}_{\mathcal{N}(i)}, \mathbf{x}^n, \mathbf{E})$. To produce diversified weak learners, a necessity for ensemble methods, we train them on disjoint subsets of pixels. This approach to subset-based ensemble learning is related to the Bayesian committee machine (Tresp, 2000).

Prediction For a previously unseen test image $\mathbf{x}^* \in \mathcal{X}$, the predictive distribution over the latent structured output $\mathbf{y}^* \in \mathcal{Y}$ can be computed as follows:

$$\Pr(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \int \left(\int \Pr(\mathbf{y}^*|\mathbf{x}^*, \mathbf{E}^*) \Pr(\mathbf{E}^*|\mathbf{E}) d\mathbf{E}^* \right) \times \underbrace{\frac{\prod_{n=1}^{N} \Pr(\mathbf{y}^n|\mathbf{x}^n, \mathbf{E}) \Pr(\mathbf{E})}{p(\mathcal{D})}}_{\Pr(\mathbf{E}|\mathcal{D})} d\mathbf{E} \tag{4a}$$

$$\approx \int \left(\int \Pr(\mathbf{y}^*|\mathbf{x}^*, \mathbf{E}^*) \Pr(\mathbf{E}^*|\mathbf{E}) d\mathbf{E}^* \right) \times \frac{\prod_{n=1}^{N} \prod_{i \in \mathcal{V}'} \Pr(y_i^n | \mathbf{y}_{\mathcal{N}(i)}, \mathbf{x}^n, \mathbf{E}) \Pr(\mathbf{E})}{p(\mathcal{D})} d\mathbf{E}, \tag{4b}$$

where $\Pr(\mathbf{y}^*|\mathbf{x}^*, \mathbf{E}^*)$ is a predictive likelihood, $\Pr(\mathbf{E}^*|\mathbf{E})$ is a conditional multivariate Gaussian (MVG) distribution (due to the GP marginalisation property (Rasmussen & Williams, 2006)), and $\Pr(\mathbf{E}|\mathcal{D})$ is the posterior distribution. In equation (4b) $\Pr(\mathbf{E}|\mathcal{D})$ is obtained from the learning stage, and involves the pixel subset PL.

We use a Monte Carlo estimate to approximate the above predictive distribution, where posterior samples of $\mathbf{E}|\mathcal{D}$ are produced by the learning (ESS) part of the algorithm discussed above. Since the predictive likelihood $\Pr(\mathbf{y}^*|\mathbf{x}^*, \mathbf{E}^*)$ is again intractable, we use tree-reweighted (TRW) belief propagation (Wainwright & Jordan, 2008) as an approximation. TRW yields a tractable upper bound on the log partition function, which might give an inconsistent marginal predictive likelihood $\Pr(y_i^*|\mathbf{x}^*, \mathbf{E}^*)$ in the sense that no joint distribution yields those marginals. Despite this inconsistency, practically, TRW-based inference delivers state-of-the-art predictive performance (see for example Domke (2013)).

To obtain an optimal point estimate $\mathbf{y}^*$, we maximise the expected utility $\mathbf{y}^{*,\text{opt}} = \arg\max_{\mathbf{y}^{\text{guess}}} \sum_{\mathbf{y}^*} \Pr(\mathbf{y}^*|\mathbf{x}^*) U(\mathbf{y}^{\text{guess}}, \mathbf{y}^*)$, for some utility function U. In our experiments, we will use a utility function corresponding to Hamming error, i.e. which counts the number of components of the label vector which are correct. For this utility function, maximising expected utility implies maximising each component independently: $y_i^{*,\text{opt}} = \arg\max_{y_i^*} \Pr(y_i^*|\mathbf{x}^*)$, known as maximum posterior marginal inference (Marroquin et al., 1987).

4. Bagging for GPstruct

Much evidence shows that ensemble learners can exceed the performance of simple models. Examples of ensemble methods are bagging, boosting, random forests and their variants. The bagging algorithm (Breiman, 1996) trains each weak learner from bootstrap data and combines individual predictions by uniform averaging or voting over class labels. We use the non-parametric bootstrap of Fushiki et al. (2005) to construct the predictive distribution from Monte Carlo samples. We can now present the full algorithm of our method. Let T be the (application-dependent) size of the ensemble.

1. (**Distributed stage**) For each weak learner $t = 1, \ldots, T$
   (a) Generate bootstrap data $\mathcal{D}_t = \{(\mathbf{x}_{t,1}, \mathbf{y}_{t,1}), \ldots, (\mathbf{x}_{t,N}, \mathbf{y}_{t,N})\}$ from the empirical distribution $\hat{\Pr}(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} \delta(\mathbf{x} - \mathbf{x}_n) \delta(\mathbf{y} - \mathbf{y}_n)$.
   (b) (**Training**) Based on the subset of pixel positions $\mathcal{V}^t \subset \mathcal{V}$ on images $\mathcal{D}_t$, perform training by ESS using PL on $\mathcal{V}^t$ as the likelihood, resulting in MCMC samples $\tilde{\mathbf{E}}_t$.
   (c) (**Partial prediction**) For each sample $\tilde{\mathbf{E}}_t$, obtain samples $\tilde{\mathbf{E}}_t^*$ from the MVG $\mathbf{E}_t^*|\mathbf{E}_t$. For each sample $\tilde{\mathbf{E}}_t^*$ obtain the predictive distribution $\Pr(\mathbf{y}^*|\mathbf{x}^*, \tilde{\mathbf{E}}_t^*)$ using TRW. Aggregate these in $\Pr(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}_t) \approx \frac{1}{S} \Pr(\mathbf{y}^*|\mathbf{x}^*, \tilde{\mathbf{E}}_t^*)$, where S is the number of samples $\tilde{\mathbf{E}}_t^*$.
2. (**Aggregation stage**) Compute the complete predictive distribution $\Pr_T(\mathbf{y}^*|\mathbf{x}^*)$ as a uniform average of the $\Pr(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}_t), \forall\, t$.

Our bagging algorithm for GPstruct is distributed by nature: step 1 can run on slave nodes, step 2 on the master nodes, as our experiments demonstrate.

5. Related work

Kernel CRFs (Lafferty et al., 2004) are non-parametric random field models sharing many of the design goals of the GPstruct model; they also scale in complexity with the

amount of training data. A clique selection algorithm allows some degree of sparsification. The method we present here may be applicable to speed up kernel CRFs as well as GPstruct.

As described earlier, our work is closely related to the research efforts on *GP sparsification*, whose goal is to improve the runtime from cubic in the number of data points to linear. Almost all sparse GP methods exploit a conditional independence assumption between training and test sets, given a set of inducing points (see Quiñonero-Candela & Rasmussen (2005) for a unifying view). However, even linear scaling can be prohibitive for very large data sets. Recent progress in sparsification has led to methods that can potentially process millions of data points (Hensman et al., 2013). Our proposed ensemble method could further benefit from sparse GP methods implemented in each weak learner. This would allow large scale non-parametric structured prediction on high resolution images with millions of pixels.

6. Applications

We assess the performance of bagging applied to the GPstruct model on a multiclass image segmentation task using two datasets. We compare GPstruct to a number of other techniques. A further experiment based on an image denoising task is detailed in the supplementary material.

Stanford Background Dataset (Gould et al., 2009) This dataset consists of 715 images of different sizes, resized to 50×150 pixels. Each pixel in the image is labelled with one of 8 classes, i.e. {sky, tree, road, grass, water, building, mountain, foreground object}. We keep 80% of the data for the training set (i.e. 572 images), and 20% for the test set (143 images). This split is repeated over 5 folds.

LabelMeFacade Image Database (Fröhlich et al., 2010) Our second dataset contains 100 images for training and 845 images for testing. The images are of different sizes and are resized to 50×150 pixels. Each pixel in the image is labelled with one of 9 classes, i.e. {building, car, door, pavement, road, sky, vegetation, window, unlabelled}.

For each problem, we will compare the performance of GPstruct to that of other models suited to the same task. Our aim is to assess the contribution of different aspects of our proposed model: the non-parametric property of the latent variables, the choice of learning technique (MCMC vs. margin optimisation), the "structured output" property of GPstruct obtained by factors on the inter-pixel edges, and the improvement brought by bagging.

`CRF PL` is a CRF model trained with pseudo-likelihood.

`CRF LBMO` (for loss-based marginal optimisation) is the model described in Domke (2013), and is considered to be state-of-the-art for vision CRF applications. While traditionally, CRF parameter learning optimises the likelihood, Domke (2013) suggests fitting parameters based on the quality of prediction of a given marginal inference algorithm, obtained by TRW or mean-field (we use TRW in our experiments), using truncated univariate logistic loss. Domke (2013) outperforms likelihood-based learning methods such as PL on difficult problems where the model being fit is approximate in nature, such as image denoising and image segmentation tasks.

`independent` is a variant of `CRF LBMO` based on the same training procedure, but where prediction ignores pairwise edges, preserving only unary features. This helps appreciating the contribution of the edge factors.

`CRF LBMO bag` applies the same bagging procedure as the `GPstruct` model to the `CRF LBMO` model, i.e. weak learners are trained on the training set, and their predictions combined to obtain an overall prediction. Since `CRF LBMO` is CRF-based it produces probabilistic predictions, so that combining predictions consists of averaging marginals produced by each weak learner.

We train these CRF-based models with a regularisation parameter of 10^{-4} as in Domke (2013). Splitting regularisation parameters into unary and pairwise parameters, and giving the pairwise parameter a smaller value did not improve performance in our experiments. We use a data independent pairwise Potts factor. Our unary features are per-pixel posteriors produced by a random forest model, with 11 trees of depth 18 each. Our implementation was done in Matlab[2], and we use J. Domke's toolbox[3] for the CRF models.

The GPstruct model is configured as follows. We use a squared exponential kernel between the pixels[4], i.e. $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2)$. The kernel width γ is set to $1/(\text{number of features})$. We train 50 weak learners in total. Each weak learner is trained on 5000 pixel positions uniformly chosen in the training set images. Using only a subset of the weak learners, we can explore how to trade performance against runtime.

Computations for GPstruct were distributed on an Amazon cluster using MIT's Starcluster[5]. Each weak learner trained and issued partial predictions (steps 1a through 1c in section 4) on a separate slave node, and the final aggregation step (step 2) was carried out on the master node.

[2]The code will be available as a GPstruct toolbox at the authors' homepage

[3]`http://users.cecs.anu.edu.au/~jdomke/JGMT/`

[4]We briefly discuss the issue of kernel choice in section 7.

[5]`http://star.mit.edu/cluster`

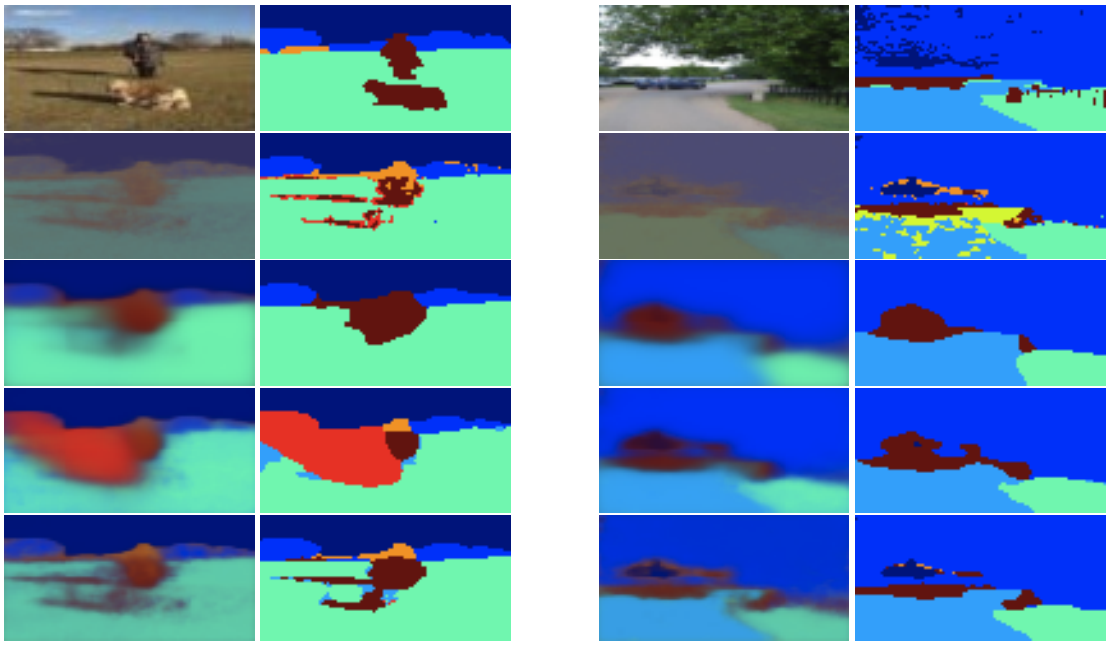

Figure 2. Semantic segmentation task (best viewed in colours). Example marginals – brightness level encodes certainty – and predicted labels from the Stanford Background Dataset. All methods use the same image features. **First row**: input image and true labels, **second row**: marginals and predicted labels of `independent`, **third row**: of `CRF PL`, **fourth row**: of `CRF LBMO`, **fifth row**: of `GPstruct`. The `independent` model performs reasonably well in predicting per-pixel segmentation, but makes rather noisy predictions, whereas `CRF PL` puts more emphasis on pairwise factors resulting in large same-label patches in predictions. `GPstruct` combines the good per-pixel segmentation of `independent` and smoothness of `CRF PL`.

6.1. Results

The results are summarised in tables 1 and 2. Our primary error metric is the per-pixel error rate, which is consistent with prediction using maximum posterior marginals. Since `GPstruct` and `CRF LBMO` produce probabilistic predictions, it is also relevant to assess the quality of posterior marginals, e.g. using the negative log marginals of the test data as a metric, which the reader will find discussed in the supplementary material. We also provide a sample visualisation of marginals and predicted labels for all methods in figure 2.

`GPstruct` outperforms the other models consistently, even with only about 15 weak learners. In all cases, each `GPstruct` weak learner ever only learns from 5000 pixels. `CRF PL` performs weakly, which is consistent with previous studies (Domke, 2013) which found CRF trained with PL suffers from model mis-specification and places too much emphasis on the pairwise factors. However, interestingly, PL performs well when used as a likelihood approximation for ESS in the GPstruct model.

Bagging has an effect on the performance of `CRF LBMO`, but it is insufficient to bring it to par with `GPstruct`. Therefore bagging alone does not justify why `GPstruct` performs better than `CRF LBMO`, and other properties like being non-parametric and Bayesian come into play.

To illustrate configuration options for `GPstruct`, we plotted the error rate against a varying number of weak learners in figure 3, which shows that `GPstruct` will attain its best performance from 15 weak learners up. Table 1 shows that increasing the number of images seen by the `GPstruct` weak learners is responsible for a performance increase, since in all cases the total number of pixels learnt from is constant at 5000. The usefulness of having more independent training images over having more dependent pixels from the same images has been observed earlier by Shotton et al. (2011) and Nowozin et al. (2011).

To explore the effect of varying the number of subsampled pixel positions in each image, we started from the set up of `GPstruct 15 WL` in table 1, with a fixed training set size of 50 images (second column). This corresponds to 100 pixels per images. Reducing the number of subsampled pixels in each image to 50 (for a total of 2500 pixels) reduces the error rate to 25.18 ± 0.70, which is still better than the 26.61 ± 0.65 error rate obtained for 25 training images and 200 pixels per image (for a total of 5000 pixels), cf. table 1. Reducing to 25 pixels per image instead of the original 100, the error rates drops again slightly to 25.23 ± 0.68. This corroborates the finding that more in-

Table 1. Error rate performance on test set of 143 images when training set size varies, $N \in \{25, 50, 100, 200, 300, 572\}$. `WL` denotes weak learner. Results are averaged over 5 folds. The best result and those results that are not significantly worse than it are highlighted in **boldface**. We used a paired Wilcoxon test with 95% confidence level as reference. Bayesian posterior inference of GPstruct generalises well at all training set sizes.

Stanford Background Dataset

	25	50	100	200	300	572 (all)
`CRF PL`	57.14±6.92	58.40±6.46	47.19±3.67	44.92±2.05	60.00±1.51	65.00±1.33
`CRF LBMO`	30.60±1.21	27.03±0.67	26.09±0.67	**25.31±0.65**	**24.91±0.63**	**24.78±0.61**
`CRF LBMO bag 15 WL`	29.56±0.77	25.68±0.61	25.24±0.54	**24.76±0.58**	**24.50±0.61**	**24.63±0.57**
`CRF LBMO bag 50 WL`	29.56±0.77	25.65±0.61	25.20±0.54	**24.73±0.58**	**24.49±0.61**	**24.61±0.57**
`GPstruct 15 WL`	**26.61±0.65**	**24.94±0.68**	24.82±0.63	**24.60±0.67**	**24.56±0.72**	**24.55±0.70**
`GPstruct 50 WL`	**26.56±0.64**	**24.90±0.67**	**24.75±0.63**	**24.51±0.68**	**24.50±0.72**	**24.53±0.69**

Table 2. Error rate performance on test set of 845 images when training samples size varies, $N \in \{1, 5, 10, 25, 50, 100\}$. Bayesian posterior inference of GPstruct generalises well even with small training set sizes.

LabelMeFacade Image Database

	1	5	10	25	50	100 (all)
`CRF LBMO`	55.99	34.84	32.11	28.37	28.01	27.81
`GPstruct`	34.05	30.23	28.57	27.86	27.84	27.80

dependent images support while keeping the total number of pixels increases performance, and illustrates the robustness of GPstruct in the small data regime. More configuration options pertaining to the GPstruct model, especially MCMC learning, are discussed in (Bratières et al., 2013).

6.2. Runtimes and complexity

Training runtimes on the full training data set are around 360,000 sec for `CRF PL`, 61,000 sec for `CRF LBMO` and `independent`. The ensemble method we describe here allows trading performance against runtime, since we can choose how many WL to train. Please refer to figure 3 for more details. Each weak learner of `GPstruct` trains for around 12h (43,000 sec), the same applies to each weak learner of `CRF LBMO bag`. `GPstruct` outperforms the other non-bagging methods with just around 5 weak learners (equivalent runtime 215,000 sec). It has equal runtime to `CRF LBMO bag` and outperforms it from around 15 weak learners.

The training runtime is dominated by the ESS algorithm's complexity, which in turn is governed by evaluations of the likelihood function for each sample of the latent variables $\mathbf{f}$. Each ESS step has a non-deterministic number of likelihood evaluations, as it is essentially a slice sampling algorithm.

Details of the complexity analysis appear in the supplementary material.

6.3. Effect of approximations on GPstruct

In order to assess the effect of approximating the likelihood with PL, and the prediction with TRW, we conducted a small-scale experiment using the standard GPstruct, in which exact likelihood and prediction are tractable. For this, the Stanford Background Dataset images were resized to 10 by 10 pixels, and the multiclass segmentation problem was turned into a foreground-background (2-class) segmentation task (the foreground label is available in the original dataset). We perform 4 shuffles of the data, and in each select 100 images for training, and 100 for testing. Error rates are averaged over these 4 shuffles and plotted against ESS iterations in figure 4.

This figure shows that the effect of approximating the prediction function has virtually no effect in the error rate. The approximation in parameter estimation using PL appears to be robust in GPstruct, leaving only a 1% gap in accuracy with the exact likelihood. We can therefore conclude that the approximations we use, while efficient enough to make GPstruct scalable, are still robust enough to have a small impact on performance.

7. Extensions and Open Problems

The results in this paper are a first step in the direction of Bayesian non-parametric structured prediction for large grid factor graphs. Several questions arise from this formulation.

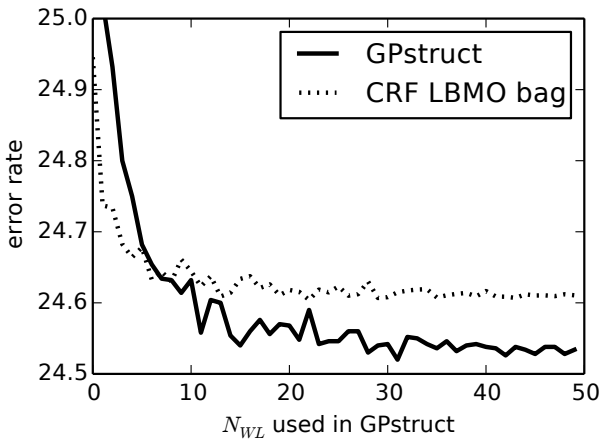

Figure 3. Speed-accuracy trade-off for `GPstruct` and `CRF LBMO bag`. Averages over 5 folds of the complete data set (572 training images and 143 test images). The size of the weak learner set used for prediction (noted N_{WL}) varies between 1 (a single weak learner is used for prediction) and 50 (the predictions of all the available weak learners are aggregated). This allows balancing runtime against desired performance. For each N_{WL}, the number of weak learner sets which are evaluated and averaged is $\max(5, 50/N_{WL})$.

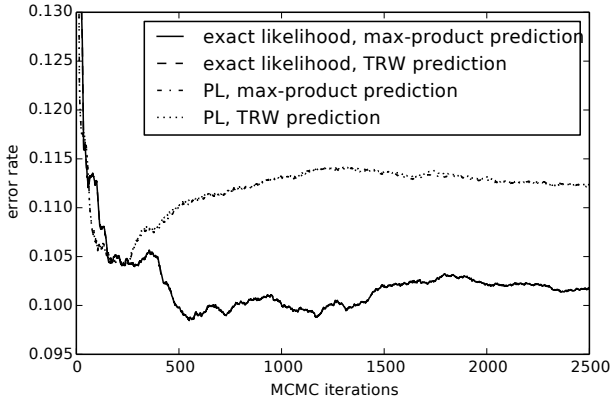

Figure 4. Effect of approximations in the standard GPstruct. All combinations of exact likelihood vs. PL and max-product prediction vs. TRW prediction are explored. The prediction approximation has virtually no effect on performance, and the likelihood approximation proves very robust.

Further scaling – In addition to using subsets of pixels to train weak learners, GP sparsification techniques (Snelson & Ghahramani, 2005; Hensman et al., 2013) applied inside each weak learner should allow substantial scale gains. This will result in hybrid methods that combine sampling and variational methods (see for example Welling et al. (2008)). In addition, the ensemble approach applied to training here can also be applied at test time, allowing higher resolution images: we could subdivide the set of test latent variables, before applying the (approximate) marginal computation once all latent variables are collected. There is a balance to be found between the size of pixel subsets per weak learner, and the computation overhead.

Kernel learning – We only explored a joint input-output kernel function that decomposes into a kernel on input space and a kernel on output space (which was simply a scaled indicator function). The input-specific kernel further decomposes into a per-element kernel of the input. Further benefits of GPstruct could come from the use of more expressive kernels over the entire input space, or potentially over the joint input-output space, combined with hyperparameter learning, which we did not explore here. Extensions of kernel learning where a rich kernel can be constructed through a weighted sum of base kernels (for example Bach (2008)), or even learning the structural form of the kernel itself (Duvenaud et al., 2013; Wilson & Adams, 2013) are open problems.

8. Conclusion

The recently presented GPstruct model (Bratières et al., 2013) has appealing properties which distinguish it among the structured prediction models, but does not scale well due to both its $O(M^2)$ space and $O(M^3)$ time complexities, where M is the number of GP latent variables. This in effect prevented GPstruct from being applied to vision problems involving grid factor graphs.

Our main contributions are a distributed ensemble method in which weak GPstruct learners produce partial probabilistic predictions based on subsets of latent variables, which can be aggregated for a high-accuracy final prediction, and a demonstration that this approach can produce competitive results on two vision tasks. Each individual weak learner benefits from the GPstruct properties: they are kernelised, non-parametric and perform Bayesian inference.

The resulting method is shown to perform very well on two classic vision tasks, binary denoising and multiclass image segmentation. In the segmentation task, GPstruct consistently outperforms state-of-the-art comparisons, and scales well to large data, with $M = 2$ million latent variables.

Acknowledgment

The authors thank Viktoriia Sharmanska for discussions and Justin Domke for help with his code. NQ is supported by the Newton International Fellowship.

References

Bach, Francis. Exploring large feature spaces with hierarchical multiple kernel learning. In *NIPS*, 2008.

Besag, J. Statistical analysis of non-lattice data. *Journal of the royal statistical society. Series D (The Statistician)*, 24(3):179–195, 1975.

Blake, Andrew, Kohli, Pushmeet, and Rother, Carsten. *Markov Random Fields for Vision and Image Processing*. MIT Press, 2011.

Bratières, Sébastien, Quadrianto, Novi, and Ghahramani, Zoubin. Bayesian structured prediction using Gaussian processes. 2013. http://arxiv.org/abs/1307.3846.

Breiman, Leo. Bagging predictors. *Machine Learning*, 24 (2):123–140, 1996.

Domke, Justin. Learning graphical model parameters with approximate marginal inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10): 2454–2467, 2013.

Duvenaud, David, Lloyd, James Robert, Grosse, Roger, Tenenbaum, Joshua B., and Ghahramani, Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *ICML*, 2013.

Fröhlich, Björn, Rodner, Erik, and Denzler, Joachim. A fast approach for pixelwise labeling of facade images. In *ICPR*, 2010.

Fushiki, Tadayoshi, Komaki, Fumiyasu, and Aihara, Kazuyuki. Nonparametric bootstrap prediction. *Bernoulli*, 11(2):293–307, 2005.

Gould, S., Fulton, R., and Koller, D. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, 2009.

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

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

Lafferty, John D., Zhu, Xiaojin, and Liu, Yan. Kernel conditional random fields: representation and clique selection. In *ICML*, 2004.

Marroquin, J., Mitter, S., and Poggio, T. Probabilistic solution of ill-posed problems in computational vision. *Journal of the American Stat. Assoc.*, 82(397):76–89, 1987.

Murray, Iain and Ghahramani, Zoubin. Bayesian learning in undirected graphical models: approximate MCMC algorithms. In *UAI*, 2004.

Murray, Iain, Ghahramani, Zoubin, and MacKay, David J. C. MCMC for doubly-intractable distributions. In *UAI*, 2006.

Murray, Iain, Adams, Ryan P., and MacKay, David J. C. Elliptical slice sampling. *JMLR - Proceedings Track*, 9: 541–548, 2010.

Nowozin, Sebastian, Rother, Carsten, Bagon, Shai, Sharp, Toby, Yao, Bangpeng, and Kohli, Pushmeet. Decision tree fields. In *ICCV*, 2011.

Parise, Sridevi and Welling, Max. Learning in Markov random fields: An empirical study. In *Joint Statistical Meeting*, 2005.

Quiñonero-Candela, Joaquin and Rasmussen, Carl Edward. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1960, 2005.

Rasmussen, Carl E. and Williams, Christopher K. I. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Schölkopf, Bernhard and Smola, Alexander J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

Shotton, Jamie, Fitzgibbon, Andrew W., Cook, Mat, Sharp, Toby, Finocchio, Mark, Moore, Richard, Kipman, Alex, and Blake, Andrew. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011.

Snelson, Edward and Ghahramani, Zoubin. Sparse Gaussian processes using pseudo-inputs. In *NIPS*, 2005.

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

Tresp, Volker. A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

Wainwright, Martin J. and Jordan, Michael I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Wang, Lei, Liu, Jun, and Li, Stan Z. MRF parameter estimation by MCMC method. *Pattern Recognition*, 33(11): 1919–1925, 2000.

Welling, M., Teh, Y.W., and Kappen, B. Hybrid variational-MCMC inference in Bayesian networks. In *UAI*, 2008.

Wilson, Andrew G. and Adams, Ryan P. Gaussian process kernels for pattern discovery and extrapolation. In *ICML*, 2013.