STUDIA UNIV. "BABEŞ–BOLYAI", MATHEMATICA, Volume **L**, Number 3, September 2005

MULTI-CLASS INFERENCE WITH GAUSSIAN PROCESSES

BOTOND CSEKE LEHEL CSATÓ

Abstract. A Bayesian probabilistic framework for multi-class classification is presented. We employ Gaussian processes as latent variable models for each of the classes and present a Bayesian inference scheme. The problem is not analytically tractable and we present approximation schemes and assess the approximation on different problems.

1. Introduction

The problem of "recognizing" patterns mathematically is formulated as the assignment of labels to specific inputs $\mathbf{x}$. The set of labels has finite cardinality, therefore the problem of label assignment is one of classification where the number of classes equals the cardinality of labels.

Binary classification is thoroughly studied and well understood for several problem domains [**?**]. It is easier to *model* the binary classification since it reduces to assigning *the sign* of a function to either of the classes. For the multi-class case it is a more difficult problem: more than two classes require to have an indicator for each class. To avoid the multiplication of these indicators, several alternative models have been proposed, all of them transform the *single* multi-class classification into *several* binary classification problems and then combine the results of the binary classifications into a single "output" [**?**, **?**]. In this article we model the multi-class classification. We use a probabilistic modelling and latent variables to model the class-conditional densities. A flexible modelling strategy is the use of random functions, namely the stochastic Gaussian processes as latent variables associated to each class.

Received by the editors: 15.09.2005.

2000 *Mathematics Subject Classification.* 68T05,68T10.

Key words and phrases. machine learning, pattern recognition, graphical models, Gaussian processes.

We present the general framework of modelling with latent variables (section 1), the models using Gaussian processes (sections 2,3) and the modelling of the multi-class classification (section 4). The article ends with the discussion of further research points worth to be carried out.

1.1. **The Classification Problem.** Let be given a set of data $D = \{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in \mathcal{X}, y_i \in \{1, \dots, C\}, i = 1, \dots, n\}$ sampled independently from an unknown distribution $P(\mathbf{x}, y)$ our task is to build a *classifier* – a function $\mathbf{x} \to y$ where $y \in \{1, \dots, C\}$ – which produces a reasonably small generalization error i.e. for a given new input $\mathbf{x}_*$, it gives a relatively good approximation for $P(y_*|\mathbf{x}_*, D)$ where y_* is in $\{1, \dots, C\}$.

The set D is usually called *training set*, the process of finding the model is called *training* or *learning process*. In most cases there is also a set S called *test set* on which we measure the performance of the model. We point out that $\mathbf{x}_*$ may be any point of the input space and the train–test method described above is just a common technique and we attempt to solve a supervised learning problem – to provide prediction for arbitrary input points $\mathbf{x}_*$– not a transductive one – to provide predictions for a fixed set of input points.

It is desirable to build a classifier which produces low errors both on training and test sets. A too low error on training set usually leads to weak prediction – high error on the test set – performance since the model is fitted "too tight" to the training data. This effect is known as *over-fitting*. Usually we expect that inputs close to each other belong to the same class – the modelled classifier is smooth in some sense – so it is plausible to penalize overly complex candidates which usually produce low errors on the training set.

1.2. **Probabilistic models and Bayesian inference.** When building model for data one usually postulates some structure for the hidden mechanism that supposedly produced it – depending on the nature of the problem at hand. This assumption leads to the introduction of *hidden* or *latent variables* – u from now on – and the assumption that the outputs y and the inputs $\mathbf{x}$ are conditionally independent given u. There may be various practical motivations for modelling with hidden variables like: it is

easier to introduce *smoothness* criteria and it is easier to model $P(y|u)$ and $P(u|\mathbf{x})$ independently than modelling the relation $P(y|\mathbf{x})$ – the relation between $\mathbf{x}$ and y – directly. This assumption can be written in the form:

$$P(y|\mathbf{x}) = \int P(y|u)P(u|\mathbf{x})du.$$

One postulates the model by specifying the distributions $P(y|u)$ and $P(u|\mathbf{x})$.

The distribution $P(y|u)$ is usually called *noise* distribution/model or *random component* and it expresses our belief about how the *hidden variables* produce the output y. There are two ways of defining the *prior distribution* $P(u|\mathbf{x})$ of the *hidden variables*: (1) in a *parametric* manner: $u(\mathbf{x}; w)$ is a parametric function and we place some prior distribution $P(w)$ on parameters w which usually have a low dimensionality not depending on the cardinality of the data sets (2) in a *non-parametric* manner: we place a prior $P(u|\mathbf{x})$ directly on $u(\mathbf{x})$. If $u(\cdot)$ is a Gaussian Process then it is specified by its mean value and covariance function.

In some cases $P(y|u)$ and $P(u|\mathbf{x})$ depend on further parameters θ called *hyperparameters*, they control characteristics like parameters of the distribution $P(w)$ or parameters of functions which define $u(\cdot)$. Smoothness criteria – mentioned a few paragraph earlier – may be expressed by the priors placed on $u(\cdot)$ or $u(\cdot; w)$ (a prior on w) assigning low probability to models leading to overly complex functions. We shall see later – sections 2.1 and 3.1 – how these probabilities control the smoothness of the model.

In this text we are concerned with *non-parametric* models and in the following we shall present how the Bayesian machinery – repeated application of Bayes' rule (see for example Soós [?]) in a hierarchy – can be put in work in such cases.

In order to make predictions one has to calculate the posterior probability of the hidden variables at training input locations. Denoting them by $u(X_D)$ we have:

$$P(u(X_D)|D) \propto P(D|u(X_D))P(u(X_D))$$

and using the assumption that y and x are conditionally independent w.r.t u one gets:

$$P(u_*|\mathbf{x}_*, D) = \int P(u_*|u(X_D))P(u(X_D)|D)du(X_D). \tag{1}$$

and

$$P(y_*|\mathbf{x}_*, D) = \int P(y_*|u_*)P(u_*|\mathbf{x}_*, D)du_*.$$

where we have denoted by $\mathbf{x}_*$ the input location where the prediction needs to be done, $u_* = u(\mathbf{x}_*)$ the value of the latent variable at this location and by y_* the predicted output variable.

When *hyperparameters* are involved, one uses a second level inference. One has to weight the prediction distributions $P(y_*|\mathbf{x}_*, D, \theta)$ with the suitability of the model with parameter θ. This suitability is usually measured by the posterior distributions of θ:

$$\begin{aligned} P(\theta|D) &\propto P(\theta)P(D|\theta) \\ &\propto P(\theta)\int P(D|u(X_D), \theta)P(u(X_D)|\theta)du(X_D). \end{aligned} \tag{2}$$

When the cardinality of the training data is sufficiently large, then $P(\theta|D)$ is highly peaked around its mode $\hat{\theta}$ .This means that the posterior is unimodal, therefore it is a common practice to substitute it by $\delta_{\hat{\theta},\theta}$. This method is called *maximum likelihood II* and the prediction we obtain using this method is called *maximum a-posteriori (MAP)* approximation. Using the Bayesian approach one has to sum over all possible parameters and gets:

$$P(y_*|x_*, D) = \int P(y_*|x_*, \theta)P(\theta|D)d\theta.$$

The process of learning is realized trough Bayesian estimations i.e. it means the updating of model parameters from the *priors* $P(\theta)$ to *posteriors* $P(\theta|D)$.

2. **Modelling with Gaussian Processes**

When modelling with Gaussian Processes (GPs) we place a Gaussian process prior on the random function u, thus the *hidden variable/function* has the property

that for any collection of possible different inputs $X = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_m\} \subset \mathcal{X}$ the random variable $u(X) = (u(\mathbf{x}'_1), \ldots, u(\mathbf{x}'_m))^T$ is a Gaussian random vector. The process is determined by its mean value function $m(\mathbf{x}) = \mathrm{E}[u(\mathbf{x})]$ and covariance function $K(\mathbf{x}.\mathbf{x}') = \mathrm{E}[(u(\mathbf{x}) - m(\mathbf{x}))(u(\mathbf{x}') - m(\mathbf{x}'))]$ which is a positive definite symmetric function, thus "producing" valid covariance matrices for any finite dimensional distribution.

Gaussian processes have long been studied in probability and statistics and used for various problems in nonparametric estimation but they have been "rediscovered" by the *ML* community only a decade ago when Neal [**?**], Williams and Rasmussen [**?**] showed that the output distribution of a simple two layer Bayesian Neural Network with increasing number of hidden units converges to a Gaussian process. Their nonparametric nature makes them relatively insensible to data dimensionality and reasonably complex models can be built with a few number of *hyperparameters* only. Being a nonparametric method smoothness conditions can be imposed by the choice of covariance function as it was pointed out by Parzen [**?**] then later on by Kimeldorf and Wahba [**?**], therefore Gaussian Processes are a tempting device for attacking *ML* problems.

Let X_S with $X_S \cap X_D = \emptyset$ be a set of test "locations" where estimations needs to be done and let us denote $X_D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ the set of training input locations and $\mathbf{u} = u(X_D) = (u(\mathbf{x}_1), \ldots, u(\mathbf{x}_n))^T$ the hidden variable vector at the training input points.

Applying the Bayesian model presented above one gets:

$$P(u(X_S)|X_S, D) = \int P(u(X_S)|\mathbf{u})P(\mathbf{u}|D)d\mathbf{u}.$$

For notational simplicity we shall use from now on the notation $P(u_*|D)$ for $P(u(X_S)|X_S, D)$, – whenever it is unambiguous – expressing that X_S is arbitrary and that we do not consider modelling $P(\mathbf{x})$. The process resulting form the finite dimensional distributions $P(u(X_S)|X_S, D)$ is called *posterior predictive process* and we shall sometimes call it simply: *posterior process*.

2.1. **Gaussian process regression with Gaussian noise.** The simplest probabilistic model using Gaussian process as hidden function is the regression problem with zero-mean Gaussian noise $P(y|u) \sim N(y; u, \sigma^2)$, which has an analytically easily tractable formalism. We assume that the "hidden function" is a Gaussian random function having a priori zero mean and a covariance K. For notational simplicity let us denote $u_i = u(\mathbf{x}_i)$, $\mathbf{y} = (y_i)_i$, $\mathbf{k}(\mathbf{x}) = (K(\mathbf{x}, \mathbf{x}_i))_i$, $k_* = K(\mathbf{x}_*, \mathbf{x}_*)$ and the covariance matrix at training input locations $\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j}$. Employing again the Bayesian formalism presented above one gets the posterior distribution

$$\begin{aligned} P(\mathbf{u}|D) &\propto \prod_i P(y_i|u_i)P(\mathbf{u}) \\ &= N\left(\mathbf{K}(\sigma^2\mathbf{I}+\mathbf{K})^{-1}\mathbf{y}, \sigma^2(\sigma^2\mathbf{I}+\mathbf{K})^{-1}\mathbf{K}\right) \end{aligned}$$

and thus obtaining the predictive distribution:

$$\begin{aligned} P(u_*|D) &= \int P(u_*|\mathbf{u})P(\mathbf{u}|D)d\mathbf{u} \qquad (3) \\ &= N\left(u_*|\mathbf{k}^T(\mathbf{x}_*)(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y}, k_* - \mathbf{k}^T(\mathbf{x}_*)(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}_*)\right). \qquad (4) \end{aligned}$$

leading to a Gaussian *predictive process* $u(\cdot)|D$ with

$$\begin{aligned} \mathrm{E}[u(\mathbf{x})|D] &= \mathbf{k}^T(\mathbf{x})(\sigma^2\mathbf{I}+\mathbf{K})^{-1}\mathbf{y} \\ \mathrm{Cov}[u(\mathbf{x}), u(\mathbf{x}')|D] &= K(\mathbf{x}, \mathbf{x}') - \mathbf{k}^T(\mathbf{x})(\sigma^2\mathbf{I}+\mathbf{K})^{-1}\mathbf{k}(\mathbf{x}'). \end{aligned}$$

We remark that denoting $\mathbf{w} = (\sigma^2\mathbf{I}+\mathbf{K})^{-1}\mathbf{y}$ – which is independent of $\mathbf{x}_*$ – we have:

$$\mathrm{E}[u(\mathbf{x})|D] = \mathbf{w}^T\mathbf{k}(\mathbf{x}) = \sum w_i K(\mathbf{x}, \mathbf{x}_i).$$

Analyzing equation (5) we notice that the point-wise predictive variance is smaller than the prior variance.

Using an arbitrary noise model – changing the noise distribution $P(y_i|u(\mathbf{x}_i))$ – we might generalize this Gaussian process regression model but $P(u(\cdot)|D)$ is not Gaussian anymore, not is the *point-wise predictive distribution* and *posterior predictive process*. Using a fixed covariance function is not generally useful for practical purposes, because it's nature affects the "quality" of the approximation and prediction we obtain. Since the posterior mean value is a linear combination of functions

$K(\mathbf{x}, \mathbf{x}_i)$, choosing fast or slow decaying covariances may lead to poor approximations. With parameterized covariance function we may get better control over the flexibility of the functions/processes in issue. Due to ease in identifying its parameters role the square exponential

$$K(s,t) = b + a\exp(-\frac{1}{2}\sum_i v_i(s^i - t^i)^2) \tag{5}$$

is one of the most often used covariance functions in machine learning GP models. Figure 2.1 shows how the choice of the so called scale parameters v_i control the posterior mean value. (We may use $\theta_0 = (\log(b), \log(a), (\log v_i)_i)$ as covariance function parameter.)

Because $P(\theta|D)$ is not a Gaussian, the predictive distribution given in equation (2) is not analytically tractable and instead, finding a $MLII$ value or sampling methods (ex. Markov Chain Monte Carlo, Hybrid Monte Carlo) – from $P(\theta|D)$ – must be employed for carrying out the integration numerically. Both of these may be done by using the log–likelihood:

$$\log P(D|\theta) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log|\mathbf{K} + \sigma^2\mathbf{I}| - \frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}. \tag{6}$$

The key of the relative ease in formalism of the regression problems is the closeness property of the Gaussian distributions function regarding multiplication and division. Gaussian likelihoods are able to model only a small proportion of real word problems but Gaussian processes can model a large variety of functions – for example the class of posterior mean value functions when a parametrized exponential covariance is used – therefore it worths to keep the function class and develop methods for a wider or arbitrary class of likelihoods, see Csató [**?**].

3. **Approximate inference**

As we have pointed out in the previous section non-Gaussian noise distributions are the ones which "break" the analytical tractability of the Bayesian model for GPs. Taking account that in cases when the log-likelihood $\log P(D|u)$ is concave

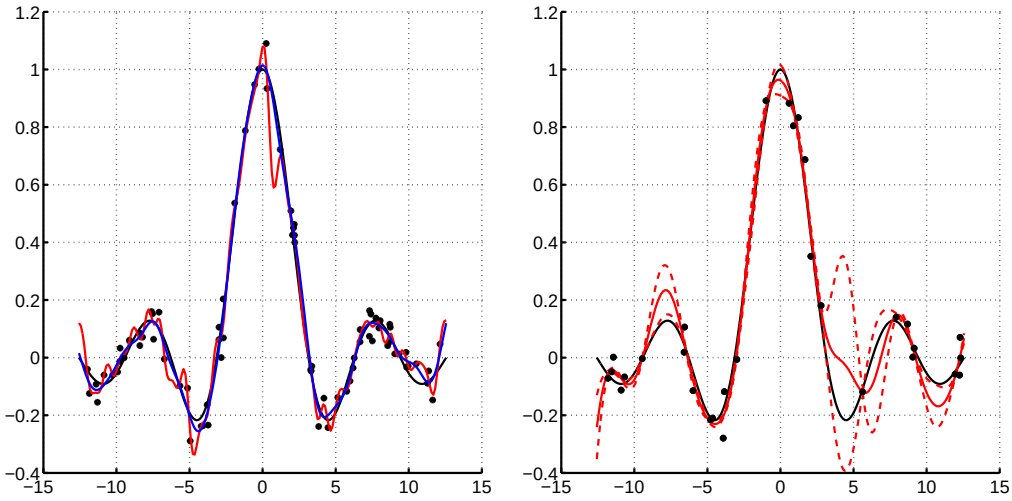

FIGURE 1. An illustration of overfitting and smooth fitting on noisy data generated by the *sinc* function (left) as well as the GP regression posterior mean and variance (right).

the posterior is unimodal it seems a good solution to approximate non-Gaussian posteriors with Gaussian ones. In the followings we shall present a few variants of this approach.

3.1. **Binary classification using Laplace approximation.** The main idea is to transform the classification problem into a regression one and the interpret the obtained results. Let $y_i \in \{0, 1\}$ and assume that if $\mathbf{x}_i$ belongs to class C_0 then $y_i = 0$. We model the problem in the following way: we shall transform the output i.e. the process u to the interval $[0, 1]$ with a suitable function σ and we shall interpret $\sigma(u(\mathbf{x}))$ as $P(\mathbf{x} \in C_1)$ - the probability of $\mathbf{x}$ belonging to the class C_1 denoted by the 1 values of y_i-s. We use the function $\sigma(x) = e^x/(1 + e^x)$, thus our goal is to approximate $P(\sigma(u(\mathbf{x}))|D)$ at a fixed point $\mathbf{x}$. For notational simplicity we demote $\pi(\mathbf{x}) = \sigma(u(\mathbf{x}))$. To apply the Bayesian treatment presented in the previous section one must postulate the corresponding conditional densities $P(y_i|u(\mathbf{x}_i))$. Assuming $\pi(\mathbf{x})$-s are probability

of success for the Bernoulli random variables y and the samples $(\mathbf{x}_i, y_i)$ are independent:

$$P(D|\mathbf{u}) = \prod_i \pi(\mathbf{x}_i)^{y_i}(1-\pi(\mathbf{x}_i))^{(1-y_i)}.$$

Since in this case the posterior process is not a Gaussian, Williams and Barber [**?**] propose a Gaussian approximation for the posterior process. Using Laplace's method they approximate $P(u(\mathbf{x}_*), \mathbf{u}|D)$ by a Gaussian at it's mode, then they marginalize and obtain $P(u(\mathbf{x}_*)|D)$ and so the last step remains the calculation of $P(\pi(\mathbf{x}_*)|D)$. Applying Bayes' rule one gets:

$$\begin{aligned}\log P(u_*, \mathbf{u}|D) &= \log P(\mathbf{y}|\mathbf{u}_+) + \log P(\mathbf{u}_+) - \log P(D). \\ &= \mathbf{y}^T\mathbf{u} - \sum_{i=1}^{n} \log(1+e^{u_i}) - \frac{1}{2}{\mathbf{u}_+}^T\mathbf{K}_+^{-1}\mathbf{u}_+ - \frac{1}{2}\log|\mathbf{K}_+| + c\end{aligned}$$

where $\mathbf{K}_+$ is the extended – with $\mathbf{k}(\mathbf{x}_*)$ and $K(\mathbf{x}_*, \mathbf{x}_*)$ – covariance matrix and $\mathbf{u}_+$ is the extended hidden variable at inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{x}_*$. One may easily verify that at the maximum – $\nabla_{\mathbf{u}_+} \log P(\mathbf{u}_+|D) = 0$ – we have $(u_*)_{max} = \mathbf{k}^T(\mathbf{x}_*)\mathbf{K}^{-1}\mathbf{u}_{max}$ where $\mathbf{u}_{max} = \text{argmax}_{\mathbf{u}} P(\mathbf{u}|D)$.

Denoting by $\bar{u}|D$ the approximating Gaussian process of the posterior process where approximation is understood in the sense presented above: Laplace's method – one gets:

$$\bar{u}(\mathbf{x})|D \sim N(\mathbf{k}^T(\mathbf{x})\mathbf{K}^{-1}\mathbf{u}_{max}, K((\mathbf{x}),(\mathbf{x})) - \mathbf{k}^T(\mathbf{x})(\mathbf{I}+\mathbf{W}\mathbf{K})^{-1}\mathbf{W}\mathbf{k}(\mathbf{x}))$$

and we have used the notation $W = -(\nabla\nabla^T)_{\mathbf{u}} \log P(D|\mathbf{u})|_{\mathbf{u}=\mathbf{u}_{max}}$

When parameterized covariance function is used we obtain the likelihood approximation

$$\log P(D|\theta) \simeq \log P(\mathbf{u}_{max}|D) - \frac{1}{2}\log|\mathbf{K}^{-1} + \mathbf{W}(\mathbf{u}_{max})| + c.$$

This can be used for sampling from $P(\theta|D)$ to carry out the Bayesian averaging or hyperparameter optimization.

3.2. **Expectation–propagation.** When using non-Gaussian likelihoods the posterior process is not Gaussian which makes the estimation of predictions analytically intractable. The finite dimensional distributions of the posterior process may be written similarly to equation (1), meaning that, in order to get a Gaussian posterior we should approximate the non-Gaussian $P(\mathbf{u}|D)$ with a Gaussian $Q(\mathbf{u})$ and define the Gaussian approximation of the posterior process defined by the finite dimensional distributions

$$Q(u_*) = \int P(u_*|\mathbf{u})Q(\mathbf{u})d\mathbf{u}$$

In section 3.1 Laplace approximation has been applied to approximate the posterior process for the binary classification problem. Another plausible way to approximate the posterior is to find the Gaussian which minimizes the KL distance (see for example Cover and Thomas [**?**]) defined by

$$\mathrm{D}[P(\mathbf{u}|D)||Q(\mathbf{u})] = \int \ln\left[\frac{P(\mathbf{u}|D)}{Q(\mathbf{u})}\right] P(\mathbf{u}|D)d\mathbf{u}.$$

Because

$$\mathrm{D}[P(u_*, \mathbf{u}|D)||Q(u_*, \mathbf{u})] = \mathrm{D}[P(\mathbf{u}|D)||Q(\mathbf{u})]$$

and the minimization boils down to second and first order matching between $P(\mathbf{u}|D)$ and $Q(\mathbf{u})$, see Opper [**?**].

The *Expectation Propagation (EP)* method developed by Minka [**?**] proves to be an efficient method for doing KL-type approximate inference in probabilistic models using factorizing likelihoods, because the properties of KL distance endow EP with a particularly important local property in cases when the factors depend only on a few components of the hidden variable vector (a few number of hidden variables).

We shall present this method in the context of GP models. A complete and general exposition is found in Minka [**?**]. The main idea of EP consists in a novel interpretation of the Assumed Density Filtering (ADF) method. Supposing a factorizing likelihood $P(D|\mathbf{u}) = \prod_i t_i(\mathbf{u})$ – $t_i(\mathbf{u})$ standing for $P(y_i|\mathbf{u})$ – and a prior $P(\mathbf{u})$, EP approximates the posterior $P(\mathbf{u}|D)$ by a distribution $Q(\mathbf{u}) = P(\mathbf{u})\prod_i \bar{t}_i(\mathbf{u})$, thus approximating each "component" of the likelihood – also called sites – by an

"easy to handle" distributions – in our case low dimensional Gaussians. It does this in the following way:

(1) the algorithm usually starts assuming $Q(\mathbf{u}) := P(\mathbf{u})$ thus setting $\bar{t}_i := 1$

(2) at each step a chosen – arbitrary or by other way – $\bar{t}_i(\mathbf{u})$ is removed from $Q(\mathbf{u})$ resulting:

$$Q^{\setminus i}(\mathbf{u}) \sim P(\mathbf{u}) \prod_{j \neq i} \bar{t}_j(\mathbf{u})$$

(3) it infers $\hat{P}(\mathbf{u}) \sim Q^{\setminus i}(\mathbf{u}) t_i(\mathbf{u})$ and

(4) approximates $\hat{P}(\mathbf{u})$ by $Q^{new}(\mathbf{u})$ such that $\bar{t}_i(\mathbf{u}) \sim Q^{new}(\mathbf{u})/Q^{\setminus i}(\mathbf{u})$ belongs to the assumed family.

Following steps (2-4), *EP* updates the influence of site $t_i(\mathbf{u})$.The repetitions of these steps lead to good approximations although convergence is not guaranteed.

Taking in consideration the closeness properties of the Gaussian family *EP* seems a well suited method for approximating $P(\mathbf{u}|D)$ because we only have to choose $\bar{t}_i(\mathbf{u})$ and $Q^{new}(\mathbf{u})$ as Gaussians densities to "make" this method "work".

Now suppose t_i–s depends only on a subset of parameters say $I_i \subset \{1, \ldots, n\}$ and let $R_i = \{1, \ldots, n\} \setminus I_i$ – both ordered – then $t_i(\mathbf{u}) = t_i(\mathbf{u}_{I_i})$. When updating $\bar{t}_i$ one has

$$\begin{aligned} \hat{P}(\mathbf{u}) &\propto Q^{\setminus i}(\mathbf{u}) t_i(\mathbf{u}_{I_i}) \\ &\propto Q^{\setminus i}(\mathbf{u}_{R_i}|\mathbf{u}_{I_i}) Q^{\setminus i}(\mathbf{u}_{I_i}) t_i(\mathbf{u}_{I_i}) \\ &\propto Q^{\setminus i}(\mathbf{u}_{R_i}|\mathbf{u}_{I_i}) \hat{P}(\mathbf{u}_{I_i}) \end{aligned}$$

and has to minimize

$$\begin{aligned} \mathrm{D}[\hat{P}(\mathbf{u})||Q^{new}(\mathbf{u})] &= \mathrm{D}[\hat{P}(\mathbf{u}_{I_i})||Q^{new}(\mathbf{u}_{I_i})] \\ &\quad + \mathrm{E}_{\mathbf{u}_{I_i} \sim \hat{P}(\mathbf{u}_{I_i})} \left[\mathrm{D}[Q^{\setminus i}(\mathbf{u}_{R_i}|\mathbf{u}_{I_i})||Q^{new}(\mathbf{u}_{R_i}|\mathbf{u}_{I_i})] \right] \end{aligned}$$

As the minimum of the positive second term of the left hand side is 0 when $Q^{\setminus i}(\mathbf{u}_{R_i}|\mathbf{u}_{I_i}) = Q^{new}(\mathbf{u}_{R_i}|\mathbf{u}_{I_i})$ and this puts no constraint criteria on the first term the minimizations means finding the moments up to second order of $\hat{P}(\mathbf{u}_{I_i}) \propto$

$Q^{\setminus i}(\mathbf{u}_{I_i})t_u(\mathbf{u}_{I_i})$ leading to $Q^{new}(\mathbf{u}) = Q^{\setminus i}(\mathbf{u}_{R_i}|\mathbf{u}_{I_i})Q^{new}(\mathbf{u}_{I_i})$ and so the site approximation $\bar{t}_i(\mathbf{u}) = \bar{t}_i(\mathbf{u}_{I_i}) \propto Q^{new}(\mathbf{u}_{I_i})/Q^{\setminus i}(\mathbf{u}_{I_i})$ depends on the same set of hidden variables as its corresponding site. Now its is easy to show that $Q^{new}(\mathbf{u}_{R_i}|\mathbf{u}_{I_i}) = Q(\mathbf{u}_{R_i}|\mathbf{u}_{I_i})$ and thus the update step has a local nature.

Assuming that $Q(\mathbf{u}) = N(\mathbf{u}; \mathbf{h}, \mathbf{A})$ the a approximation if the first two moments of $t_i(\mathbf{u}_{I_i})Q^{\setminus i}(\mathbf{u}_{I_i})$ cannot be done analytically and we employed Gauss-Hermite quadrature method – see for example Coman [**?**]. Applying change of variables one can factorize the weight function $N(\mathbf{u}_{I_i}; [\mathbf{h}^{\setminus i}]_{I_i}, [\mathbf{A}^{\setminus i}]_{I_i})$ in order to get a tractable quadrature formula. Let $\mathbf{x}$ and $\mathbf{w}$ be the d-th order Gauss-Hermite nodes and weights of $N(\cdot; 0, 1)$ and σ an element of the Descartes product $\{1, \ldots, d\}^{|I_i|}$. Using a Cholesky decomposition $[\mathbf{A}^{\setminus i}]_{I_i} = \mathbf{L}\mathbf{L}^T$, and denoting $\mathbf{m} = [\mathbf{h}^{\setminus i}]_{I_i}$, the approximation formula for the normalization constant of $t_i(\mathbf{u}_{I_i})N(\mathbf{u}_{I_i}; [\mathbf{h}^{\setminus i}]_{I_i}, [\mathbf{A}^{\setminus i}]_{I_i})$ is given by:

$$Z_i \simeq \sum_{\sigma} \prod_{j} w_{\sigma}^{j}\, t_i(\mathbf{L}\mathbf{x}_{\sigma} + \mathbf{m})$$

and thus the approximation of first and second moments is straightforward. Unfortunately this quadrature method scales exponentially in $|I_i|$ which makes is practical only for very small values only.

Although we have used the minimization of the KL distance as approximation method, in order to avoid calculation of moments we can use a "hybrid" method: to do the last step of local approximation with a Laplace-type approximation which is plausible if $t_i(\mathbf{u}_{I_i})Q^{\setminus i}(\mathbf{u}_{I_i})$ is strictly log-concave in $\mathbf{u}_{I_i}$:

$$\begin{aligned}
\hat{\mathbf{m}} &\simeq \operatorname{argmax}_{\mathbf{u}_{I_i}} \left\{ \log(t_i(\mathbf{u}_{I_i})) - \frac{1}{2}(\mathbf{u}_{I_i} - \mathbf{m})^T \mathbf{V}^{-1} (\mathbf{u}_{I_i} - \mathbf{m}) \right\} \\
\hat{\mathbf{V}} &\simeq \left(\hat{\mathbf{W}} + \mathbf{V}^{-1} \right)^{-1} \\
\log Z_i &\simeq \log(t_i(\hat{\mathbf{m}})) - \frac{1}{2}(\hat{\mathbf{m}} - \mathbf{m})^T \mathbf{V}^{-1} (\hat{\mathbf{m}} - \mathbf{m}) - \frac{1}{2} \log |\mathbf{I} + \hat{\mathbf{W}}\mathbf{V}|
\end{aligned}$$

where $\hat{\mathbf{W}} = -(\nabla\nabla^T)_{\mathbf{u}_{I_i}} t_i(\mathbf{u}_{I_i})|_{\mathbf{u}_{I_i} = \hat{\mathbf{m}}}$. The approximation can be done with a Newton-Raphson method and it takes roughly $O(|I_i|^3)$ time.

4. An approach to multi-class problem

In the followings we present the extension of binary classification problem to multiple classes. We use the representation of Barber and Williams [**?**] to build the model.

In order to avoid dealing with a great number of data resulting from a multi-output Gausssian process one can model the C-class case by choosing C independent priors and taking account only their posterior cross-correlations which is realized trough the coupling(s) in the likelihood terms. In order to avoid confusions we settle first some notational conventions: the subscript indexing is used for referring to input locations while the upper indexing is be used for referring to class type indexing – $\mathbf{u}_i = (u_i^1, u_i^2, \ldots, u_i^C)$ and $\mathbf{u}^{(c)} = (u_1^{(c)}, u_2^{(c)}, \ldots, u_n^{(c)})$, we use $\mathbf{u} = (\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \ldots, \mathbf{u}^{(C)})$ and the corresponding prior covariance matrix $\mathbf{K} = \mathrm{diag}(\mathbf{K}^{(c)})_c$.

As we have seen the two-class case, finding a likelihood term for finite number of outputs is not an easy task and one of the easiest ways to solve the problem is to turn it into a regression-like one. A multi-dimensional extension of the logistic function, the *softmax* function $\pi(\mathbf{u}) = \exp(\mathbf{u})/\mathbf{1}^T \exp(\mathbf{u})$ is used to model the class probabilities. Modelling y_i belonging to class c by $y_j = \mathbf{e}_c$ one defines the likelihood:

$$P(y|\mathbf{u}) = y^T \pi(\mathbf{u}).$$

The likelihood $\log P(y|\mathbf{u})$ is not strictly log-concave – due to $1^T \pi(\mathbf{u}) = 1$ it is strictly log-concave only on $\{\mathbf{1}\}^\perp$. This could constitute a risk to the approximation in cases when the local marginals of the likelihood approximations collapse or are close to it since both of the approximations processes presented above – minimization of KL distance and Laplace method – rely on this property.

In order to apply the *EP* procedure presented earlier we set the (site) likelihood:

$$t_i(\mathbf{u}_i) = P(y_i|\mathbf{u}_i) = (\pi(\mathbf{u}_i))_c$$

where c is the nonzero element of y_i and the use site approximations $\bar{t}_i(\mathbf{u}_i) \sim N(\mathbf{u}_i; \mathbf{m}_i, \mathbf{V}_i)$. Using the notation $\mathbf{V} = (\mathrm{diag}(\mathbf{V}_i^{(s,t)})_i)_{t,s=1,\dots,C}$ the Gaussian approximation of the posterior $P(\mathbf{u}|D)$ has the covariance $\mathbf{K} - \mathbf{K}(\mathbf{K}+\mathbf{V})^{-1}\mathbf{K}$ and mean value $(\mathbf{I}+\mathbf{V}\mathbf{K}^{-1})^{-1}(\mathbf{m}^{(c)})_c$, thus the the prediction $\mathbf{u}_*$ is normal with:

$$\begin{aligned} \mathrm{E}[\mathbf{u}_*|D] &= \mathbf{K}_*^T(\mathbf{V}+\mathbf{K})^{-1}(\mathbf{m}^{(c)})_c \\ \mathrm{Cov}[\mathbf{u}_*|D] &= \mathrm{diag}\left(K^{(c)}(\mathbf{x}_*,\mathbf{x}_*)\right)_c - \mathbf{K}_*^T(\mathbf{V}+\mathbf{K})^{-1}\mathbf{K}_* \end{aligned}$$

where we used the notation $\mathbf{K}_* = \mathrm{diag}\left(\mathbf{k}^{(c)}(\mathbf{x}_*)\right)_c$. In order to calculate the intractable $P(y_* = c|\mathbf{x}_*, D)$ one has to apply once again numerical quadrature formulas.

5. **Experiments**

We implemented the two local approximation methods described in section 3. We built upon the *OGP toolbox* developed by Csató (see, [**?**]) which was implemented based on Csató and Opper [**?**] and is publicly available with full documentation. The *OGP toolbox* provides *a sparse approximation* method for a variety of likelihoods – user defined ones as well. This makes the toolbox easily applicable for artificial or real world problems that employ *Gaussian Process models*. The Gauss-Hermite Quadrature rule was ran using 7 nodes. Increasing its order did not lead to a significant change in accuracy. In fact fewer nodes proved to be suitable, the reason for it might be the good convergence properties of the Taylor expansion of the likelihood function. Heuristics like employing the symmetry property of nodes and weights when using Gauss-Hermite quadrature formulas – as it was pointed out by Seeger and Jordan [**?**] – can be used but significant improvement in time-performance cannot be achieved without further making use of the likelihood structure. However, factorizing likelihood sites, or likelihoods that speed up the Gaussian quadrature routine might lead to worse performance because of weak couplings in the variables of the likelihood.

The "local" Laplace method was implemented using Newton-Raphson method for $\mathbf{a} = \mathbf{V}^{-1}(\mathbf{s}-\mathbf{m})$ with the update equation

$$\mathbf{a}^{new} = (\mathbf{I}+\mathbf{W}\mathbf{V})^{-1}\left(\mathbf{W}\mathbf{V}\mathbf{a} + \pi(\mathbf{s}) - \mathbf{e}_c\right)$$

where we have used the notations from section 3. In each step we have used BiCG (implemented in Matlab) to solve the linear system. The method takes around $O(n_{bicg}n_{nr}C^2)$ time where n_{bicg} and n_{nr} are denoting the number of BiCG resp. Newton-Raphson steps.

The plots on figure 5 show the error rates we achieved on a 3-class data set using a spherical (the scaling parameters v_i are equal) square exponential kernel from equation 5. We implemented hard classification rules i.e. an item belongs to the class which has the greatest probability – this was done in order to compare our results with the benchmark *5NN* classifier. For real world problems however, one could exploit the multi-output probabilistic outputs returned by the system – the freedom of choice is significantly more.

The basis for comparison was the *5NN* rule. The data was generated in the following way: we generated from 9 Gaussians with mean values chosen randomly from $[0, \ldots, 10]^2$ and labelled these randomly. The resulting data were preprocessed: we whitened the data. We used 1000 samples splitting them in 1/2 train/test ratio. Figure 5 shows hard-classification boundaries on a data set of 250 examples.

6. Conclusions/Further research

The presented methods outperformed *5NN* in cases when the scaling parameters did not have extreme values. We aim to further develop and implement *hyperparameter optimization* methods, as well as approximations for *posterior probabilities* to be used with *Markov Chain Monte Carlo, Hybrid Monte Carlo* type methods for a "complete" Bayesian inference like in section 1 to integrate out the posterior process and the hyperparameters from the model. Our interest in developing further local approximation methods is still active.

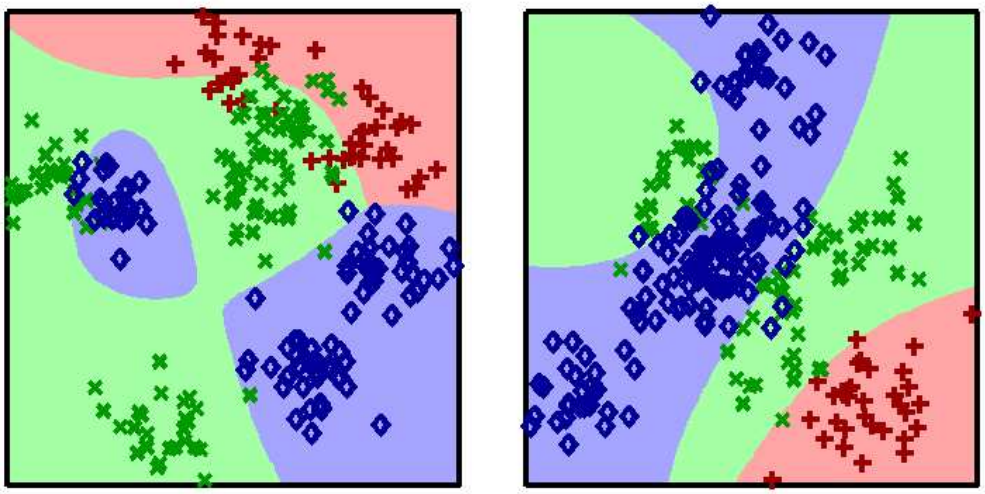

FIGURE 2. Hard classification discriminant curves for a 3-class case with 250 samples.

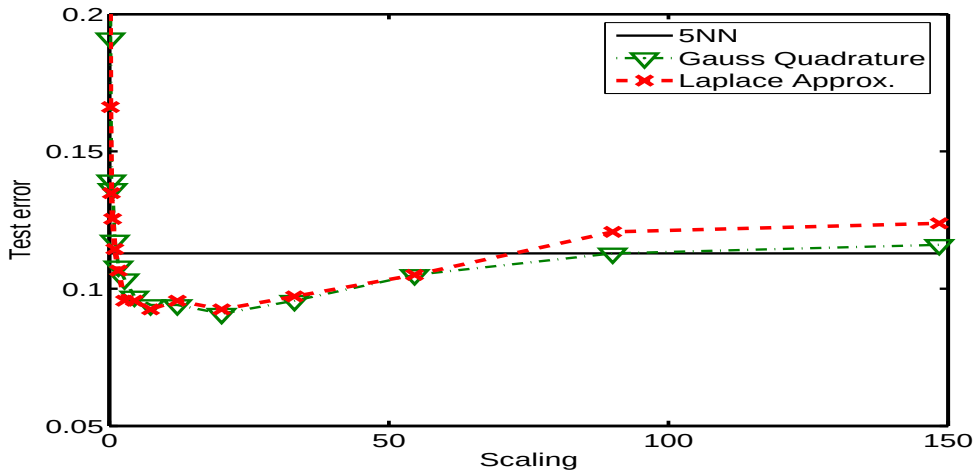

FIGURE 3. Performance of a multi-class GP methods with spherical square exponential kernel on data set with 3 classes.

BABEŞ-BOLYAI UNIVERSITY, FACULTY OF MATHEMATICS
AND COMPUTER SCIENCE, STR. KOGĂLNICEANU, 1
CLUJ-NAPOCA, ROMANIA
E-mail address: csekeb@math.ubbcluj.ro, lehel.csato@cs.ubbcluj.ro