Predictive Approaches For Gaussian Process Classifier Model Selection
September 10, 2008

S. Sundararajan   SSRAJAN@YAHOO-INC.COM
Yahoo! Labs
Bangalore, India

S. Sathiya Keerthi   SELVARAK@YAHOO-INC.COM
Yahoo! Research
Santa Clara, USA

Editor:

Abstract

In this paper we consider the problem of Gaussian process classifier (GPC) model selection with different Leave-One-Out (LOO) Cross Validation (CV) based optimization criteria and provide a practical algorithm using LOO predictive distributions with such criteria to select hyperparameters. Apart from the standard average negative logarithm of predictive probability (NLP), we also consider smoothed versions of criteria such as F-measure and Weighted Error Rate (WER), which are useful for handling imbalanced data. Unlike the regression case, LOO predictive distributions for the classifier case are intractable. We use approximate LOO predictive distributions arrived from Expectation Propagation (EP) approximation. We conduct experiments on several real world benchmark datasets. When the NLP criterion is used for optimizing the hyperparameters, the predictive approaches show better or comparable NLP generalization performance with existing GPC approaches. On the other hand, when the F-measure criterion is used, the F-measure generalization performance improves significantly on several datasets. Overall, the EP-based predictive algorithm comes out as an excellent choice for GP classifier model selection with different optimization criteria.

Keywords: Gaussian process classification, Model Selection, LOO, Cross Validation, Predictive distributions, Smoothed F-measure, Weighted Error Rate, Precision, Recall, Imbalanced data

1. Introduction

Gaussian process (GP) models are flexible and powerful probabilistic models that are used to solve classification problems in many areas of application (Rasmussen and Williams, 2006). In the Bayesian GP setup, latent function values and hyperparameters that are involved in modeling are integrated with chosen priors. However, the required integrals are often not analytically tractable (due to various choices of likelihoods and priors) and closed form analytic expressions are not available. Two important problems in this context are finding good approximations for integrating over the latent function variables and the

hyperparameters. There have been two approaches reported in the literature. In the first approach, both the latent function variables and the hyperparameters are integrated out within some approximations. Williams and Barber (1998) used Laplace approximation to integrate over the latent function variables and Hybrid Monte Carlo (HMC) to integrate over the hyperparameters. Neal (1998) used Gibbs sampling to integrate over the latent function variables and HMC to integrate over hyperparameters; this method is more accurate, but it is computationally expensive. In the second approach, only the latent function variables are integrated out and the hyperparameters are optimized on some well-defined objective function. This latter problem of choosing hyperparameters that define the model is essentially the Gaussian process model selection problem and in this paper we focus on this problem.

There are two commonly used approaches to address this model selection problem. They are marginal likelihood or evidence maximization and minimization of LOO-CV based average negative logarithmic predictive probability (NLP). Both these approaches are available for GP regression model selection (Rasmussen and Williams, 2006; Sundararajan and Keerthi, 2001). For GP classifier model selection, Gibbs and MacKay (2000) used a variational approximation method to integrate over the latent function values and estimated the hyperparameters by maximizing marginal likelihood (ML). Laplace approximation and Expectation Propagation (EP) approximation (Rasmussen and Williams, 2006; Seeger, 2003) are other methods that have been used to integrate over the latent function variables. Then, the marginal likelihood is optimized using gradient information obtained with any one of these approximations (Rasmussen and Williams, 2006; Seeger, 2003). Kim and Ghahramani (2006) presented an approximate Expectation-Maximization (EM) algorithm to learn the hyperparameters. In the E-step, they used EP to estimate the joint density of latent function values and in the M-step, the hyperparameters were optimized by maximizing a variational lower bound on the marginal likelihood.

In this paper, we consider the approach of using LOO-CV based predictive distributions to address the GP classifier model selection problem. In a related work, Opper and Winther (2000) used LOO error estimate to choose rough hyperparameter values by scanning a range of values. In the EP framework (Minka, 2001), cavity distributions directly provide LOO error estimates during training and were used to assess predictive performance and select automatic relevance determination (ARD) type hyperparameters (Qi et al., 2004). The LOO-CV based predictive distributions obtained from probabilistic least squares classifier were used in the minimization of NLP for GP classifier model selection (Rasmussen and Williams, 2006).

In practice, while measures like marginal likelihood and average negative logarithmic predictive probability measures are very useful, other measures like F-measure (van Rijsbergen, 1974) and Weighted Error Rate (WER) are also important and useful, for instance in applications like medical diagnostics, image understanding, etc, where the number of positive examples is much smaller than the number of negative examples. Several works that use such measures for hyperparameters optimization exist in the non-GP literature. Hong et al. (2007) proposed a kernel classifier construction algorithm based on regularized orthogonal weighted least squares (ROWLS) estimation with LOO-Area Under the ROC Curve (AUC) as model selection criterion for handling imbalanced datasets. Jansche (2005) proposed the training of a probabilistic classifier based on a logistic regression model by

optimizing expected F-measure. Here an approximation to F-measure was made so that the F-measure is smoothed and becomes a smooth function of model weights. Seeger (2007) proposed a general framework for learning in probabilistic kernel classification models with a large or structured set of classes. He optimized the kernel parameters by minimizing the NLP over k-folds. Keerthi et al. (2006) considered the task of tuning hyperparameters in SVM models based on minimizing smooth performance validation functions like smoothed k-fold CV error. The last three works did not use LOO-CV in their work.

This paper is aimed at addressing two issues. First, the proposed method is different from the work of Opper and Winther (2000) and Qi et al. (2004) in that we optimize the hyperparameters directly in the continuous space and any standard non-linear optimization method can be used. It is also different from the LOO-CV based probabilistic LS method (Rasmussen and Williams, 2006) in that we use the more accurate EP approximation than the LS approximation. Second, criteria such as F-measure and WER, which are needed for tackling imbalanced problems, have not been considered in GP classifier designs.

We define smoothed LOO-CV based measures using predictive distributions as a function of GP classifier model hyperparameters. Thus, the objective functions can be optimized using standard non-linear optimization techniques. We investigate usage of LOO-CV predictive distributions obtained from expectation propagation approximation. Actually, the proposed algorithm can also be used with Laplace approximation. However, Kuss and Rasmussen (2005) showed for binary classification problems that the EP approximation is better than the Laplace approximation. Therefore, we restrict our attention to using the EP approximation here.

We conduct experiments on two criteria: the standard average negative logarithm of predictive probability (NLP) and a smoothed version of F-measure. On the NLP criterion we compare our method (EP-CV(NLP)) against the LOO-CV based probabilistic least squares (LS) classifier (Rasmussen and Williams, 2006) and standard GP classifier doing ML maximization using EP approximation. We refer the latter two methods as LS-CV(NLP) and EP(ML) respectively; the abbreviations in parentheses refer to the type of objective functions used. The experimental results on several real world benchmark datasets show that the proposed method is better than the LS-CV(NLP) method and is quite competitive to the EP-ML method. On the F-measure criterion we compare the EP-CV(NLP) method with the EP-CV(FM) method. In the latter method we optimize over the F-measure instead of the NLP measure. We also compare with a two-step method[1] where, in the first step we optimize over the hyperparameters using the NLP measure and, in the second step we optimize *only* the *bias* hyperparameter using the F-measure. Experimental results demonstrate that this method is also inferior to the EP-CV(FM) method.

The paper is organized as follows. We give a brief introduction to Gaussian process classification and ML optimization criteria in Section 2. In Section 3 we give a general set of smooth LOO-CV based optimization criteria and illustrate their use with LOO-CV predictive distributions. Optimization aspects and specific algorithm are given in Section 4. In Section 5 we discuss related work on LOO-CV based GPC model selection. In Section 6 we present experimental results and then conclude the paper in Section 7.

---

1. This method was suggested by an anonymous reviewer.

2. Gaussian Process Classification

In binary classification problems, we are given a training data set S composed of n input-target pairs $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i \in R^D$, $y_i \in \{+1, -1\}$, $i \in \tilde{I}$ and $\tilde{I} = \{1, 2, \ldots, n\}$. The true function value at $\mathbf{x}_i$ is represented as a latent variable $f(\mathbf{x}_i)$. The goal is to compute the predictive distribution of the class label y_* at test location $\mathbf{x}_*$. In standard GPs for classification (Rasmussen and Williams, 2006), the latent variables $f(\mathbf{x}_i)$ are modelled as random variables in a zero mean GP indexed by $\{\mathbf{x}_i\}$. The prior distribution of $\{\mathbf{f}(\mathbf{X}_n)\}$ is a zero mean multivariate joint Gaussian, denoted as $p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$, where $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^T$, $\mathbf{X}_n = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ and $\mathbf{K}$ is the $n \times n$ covariance matrix whose $(i, j)^{th}$ element is $k(\mathbf{x}_i, \mathbf{x}_j)$, denoted as $\mathbf{K}_{i,j}$. One of the most commonly used covariance functions is the squared exponential covariance function given by: $\mathrm{cov}(f(\mathbf{x}_i), f(\mathbf{x}_j)) = k(\mathbf{x}_i, \mathbf{x}_j) = \beta_0 \exp(-\frac{1}{2} \sum_{k=1}^{D} \frac{(x_{i,k} - x_{j,k})^2}{\beta_k})$. Here, β_0 represents signal variance and the remaining β_k's represent width parameters across different input dimensions. Let $\boldsymbol{\beta} == [\beta_0, \beta_1, \ldots, \beta_D]$. These parameters are also known as *automatic relevance determination* (ARD) hyperparameters. We call this covariance function the ARD Gaussian kernel function. Next, it is assumed that the probability over class labels as a function of $\mathbf{x}$ depends only on the latent function value $f(\mathbf{x})$. For the binary classification problem, given the value of $f(\mathbf{x})$ the probability of class label is independent of all other quantities: $p(y = +1 | f(\mathbf{x}), S) = p(y = +1 | f(\mathbf{x}))$ where S is the dataset. The likelihood $p(y_i | f_i)$ can be modelled in several forms such as a sigmoidal function or cumulative normal $\Phi(y_i f_i)$ where $\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} \exp(-\frac{w^2}{2}) dw$. Assuming that the examples are i.i.d, we have $p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{N} p(y_i | f_i; \boldsymbol{\gamma})$. Here, $\boldsymbol{\gamma}$ represents hyperparameters that characterize the likelihood. The prior and likelihood along with the hyperparameters $\boldsymbol{\theta} = [\boldsymbol{\beta}, \boldsymbol{\gamma}]$ characterize the GP model. With these modelling assumptions, we can write the inference probability given $\boldsymbol{\theta}$ as:

$$p(y_* | \mathbf{x}_*, S, \boldsymbol{\theta}) = \int p(y_* | f_*, \boldsymbol{\gamma}) \, p(f_* | S, \mathbf{x}_*, \boldsymbol{\theta}) \, df_* \tag{1}$$

Here, the posterior predictive distribution of latent function f_* is given by:

$$p(f_* | S, \mathbf{x}_*, \boldsymbol{\theta}) = \int p(f_* | \mathbf{x}_*, \mathbf{f}, \boldsymbol{\beta}) \, p(\mathbf{f} | S, \boldsymbol{\theta}) \, d\mathbf{f}.$$

where $p(\mathbf{f}|S, \boldsymbol{\theta}) \propto \prod_{i=1}^{N} p(y_i | f_i, \boldsymbol{\gamma}) \, p(\mathbf{f} | \mathbf{X}, \boldsymbol{\beta})$. In a Bayesian solution, the class probability at the test point x_* would be obtained by integrating over the hyperparameters weighted by their posterior probability

$$p(y_* | x_*, S) = \int p(y_* | x_*, S, \boldsymbol{\theta}) \, p(\boldsymbol{\theta} | S) \, d\boldsymbol{\theta}.$$

In general there is no closed form expression available for this integral and it is expensive to compute. Therefore, instead of integrating over the hyperparameters, a single set of their values is usually estimated from the dataset by optimizing various criteria as mentioned earlier and then used in (1).

2.1 Marginal Likelihood Maximization

Marginal likelihood or evidence maximization (Rasmussen and Williams, 2006) is commonly used to estimate the hyperparameters during model selection. The marginal likelihood is given by:

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int \prod_{i=1}^{N} p(y_i|f_i, \boldsymbol{\gamma}) \, p(\mathbf{f}|\mathbf{X}, \boldsymbol{\beta}) \, d\mathbf{f} \tag{2}$$

This integral cannot be calculated analytically except for a special case like GP regression with Gaussian noise. Therefore, certain approximations are needed to compute these quantities. Laplace approximation and EP approximations are two popular methods used for this purpose. To gain more insight into the quality of the Laplace and EP approximations, Kuss and Rasmussen (2005) carried out comprehensive comparisons of these approximations with (the more exact) Markov Chain Monte Carlo (MCMC) sampling approach in terms of their predictive performance and marginal likelihood estimates. They found that EP is the method to be used for approximate inference in binary GPC models, when the computational cost of MCMC is prohibitive. Hence in our study we restrict ourselves to the more accurate EP approximation given in the next subsection. With the EP approximation the hyperparameters are learnt by optimizing marginal likelihood with gradient information (Rasmussen and Williams, 2006; Seeger, 2003) using standard non-linear optimization techniques.

2.2 Expectation Propagation

The EP algorithm is an iterative algorithm which is used for approximate Bayesian inference (Opper and Winther, 2000; Minka, 2001). It has been applied to GP classification (Rasmussen and Williams, 2006; Seeger, 2003) . EP finds a Gaussian approximation $q(\mathbf{f}|S, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{C})$ to the posterior $p(\mathbf{f}|S, \boldsymbol{\theta})$ by moment matching of approximate marginal distribution and the posterior. Mean and covariance of the approximate Gaussian are given by:

$$\mathbf{m} = \mathbf{C}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \quad \mathbf{C} = (\mathbf{K}^{-1} + \boldsymbol{\Sigma}^{-1})^{-1} \tag{3}$$

where $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_n)^T$ and $\boldsymbol{\Sigma} = diag(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$ are called site function parameters. As per the approximation, the posterior is written in terms of the site functions $t(f_i; \mu_i, \sigma_i^2, Z_i) = Z_i\mathcal{N}(f_i|\mu_i, \sigma_i^2)$ and prior $p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})$ as

$$q(\mathbf{f}|S, \boldsymbol{\theta}) = \frac{p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})}{q(S|\boldsymbol{\theta})} \prod_{i=1}^{N} t(f_i; \mu_i, \sigma_i^2, Z_i).$$

The EP algorithm iteratively visits each site function in turn, and adjusts the site parameters to match moments of an approximation to the posterior marginals. This process requires replacement of intractable exact *cavity* distribution with a tractable approximation based on the site functions and is given by:

$$q_{\backslash i}(f_i) = \int \prod_{j \neq i} t(f_j; \mu_j, \sigma_j^2, Z_j) \, p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) \, d\mathbf{f}^{\backslash i}$$

where $q_{\backslash i}(f_i) \propto \mathcal{N}(f_i|\mu_{\backslash i}, \sigma_{\backslash i}^2)$ is the approximate cavity function and is related to the diagonal entries of the posterior $q(\mathbf{f}|S, \boldsymbol{\theta})$ as: $q_{\backslash i}(f_i) t(f_i; \mu_i, \sigma_i^2, Z_i) \propto \mathcal{N}(f_i|m_i, \mathbf{C}_{ii})$. Here, the $\mathbf{C}_{ii}$ represent the diagonal entries of the matrix $\mathbf{C}$. Using Gaussian identities, the mean and variance of cavity distribution is related to the site parameters as:

$$\mu_{\backslash i} = \sigma_{\backslash i}^2 \left(\frac{m_i}{\mathbf{C}_{ii}} - \frac{\mu_i}{\sigma_i^2}\right) \qquad \sigma_{\backslash i}^2 = ((\mathbf{C}_{ii})^{-1} - \sigma_i^{-2})^{-1} \tag{4}$$

Then, EP adjusts the site parameters μ_i, σ_i^2 and Z_i such that the approximate posterior marginal using the exact likelihood approximates the posterior marginal based on the site function well. That is, $q_{\backslash i}(f_i) p(y_i|f_i) \simeq q_{\backslash i}(f_i) t(f_i; \mu_i, \sigma_i^2, Z_i)$. This is done by matching the zeroth, first and second moments on both sides. Thus, the EP algorithm iteratively updates site parameters until convergence. Though there is no convergence proof, in practice the EP algorithm converges in most cases. See Rasmussen and Williams (2006) for more details.

Next, within some constant, the marginal likelihood with EP approximation (Rasmussen and Williams, 2006) is given by:

$$\log q(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2} \log |\mathbf{K} + \boldsymbol{\Sigma}| - \frac{1}{2} \boldsymbol{\mu}^T (\mathbf{K} + \boldsymbol{\Sigma})^{-1} \boldsymbol{\mu} + \sum_{i=1}^n \log w_i \tag{5}$$

where $w_i = \Phi(z_i) \exp\left(\frac{(\mu_{\backslash i} - \mu_i)^2}{2(\sigma_{\backslash i}^2 + \sigma_i^2)}\right) \sqrt{\sigma_{\backslash i}^2 + \sigma_i^2}$ and $z_i = \frac{y_i \mu_{\backslash i}}{\sqrt{1 + \sigma_{\backslash i}^2}}$. The hyperparameters are optimized using gradient expressions with standard conjugate gradient or quasi-Newton type non-linear optimization techniques.

3. Leave-One-Out Cross Validation based Optimization Criteria

In this section, we give definitions of various LOO-CV based optimization criteria. In section 4 we give details on how these measures can be optimized using standard nonlinear optimization techniques. The LOO predictive distributions $p(y_i|\mathbf{x}_i, S_{\backslash i}, \boldsymbol{\theta})$, $i \in \tilde{I}$ play a crucial role. Here $S_{\backslash i}$ represents the dataset without ith example. Their exact computation is expensive. In section 4 we will also discuss how to approximate them efficiently.

3.1 NLP Measure

The averaged negative logarithm of predictive probability (NLP) is defined as:

$$G(\boldsymbol{\theta}) = -\frac{1}{n} \sum_{i=1}^n \log p(y_i|x_i, S_{\backslash i}, \boldsymbol{\theta}) \tag{6}$$

This LOO-CV based measure is generic and has been used in the context of probabilistic LS classifiers (see section 5) and GP regression (Rasmussen and Williams, 2006; Sundararajan and Keerthi, 2001).

3.2 Smoothed LOO Measures

While measures such as marginal likelihood (5) and NLP in (6) are useful for normal situations, other measures like F-measure and WER are important, for example, when dealing with imbalanced datasets. Let us now define these measures.

Table 1: Confusion Matrix for Binary classification

	Positive (Predicted)	Negative (Predicted)
Positive (Actual)	a	b
Negative (Actual)	c	d

Consider the binary classification problem with class labels $+1$ and -1. Assume that there are n_+ positive examples and n_- negative examples. In general, the performance of the classifier may be evaluated using counts of data samples $\{a, b, c, d\}$ defined via the confusion matrix given in Table 1. Let $n_+ = a + b$ and $n_- = c + d$. The true positive rate (TP) is the proportion of positive data samples that were correctly classified (that is, true positives) and the false positive rate (FP) is the proportion of negative data samples that were incorrectly classified (that is, false positives) (Hong et al., 2007). These rates are given by: TP $= \frac{a}{a+b} = \frac{a}{n_+}$ and FP $= \frac{c}{c+d} = \frac{c}{n_-}$. The misclassification rate is given by: MCR $= \frac{b+c}{n}$. Note that the true positive rate is also known as *Recall* (R). *Precision* is another important quantity defined as: P $= \frac{a}{a+c}$.

Now let us consider the imbalanced data case and assume that $n_- \gg n_+$. In this case if MCR is minimized then the classifier will be biased toward the negative class due to its effort in minimizing the false positives (that is c) more strongly than minimizing false negatives (that is b). In the worst case almost all the positive examples will be wrongly classified, that is $a \to 0$. This results in both $P \to 0$ and $R \to 0$. Thus MCR is not a good measure to use when the dataset is imbalanced. This problem can be addressed by optimizing other measures that we discuss next.

The F-measure is one such measure and is defined (van Rijsbergen, 1974) as:

$$F_\zeta(P, R) = \left(\frac{\zeta}{R} + \frac{1 - \zeta}{P} \right)^{-1}$$

where $0 \leq \zeta \leq 1$ and $0 \leq F_\zeta(P, R) \leq 1$. It has been used in various applications like document retrieval (van Rijsbergen, 1974) and text classification (Joachims, 2005). It is particularly preferable over MCR when the dataset is highly imbalanced (Joachims, 2005; Jansche, 2005).

Let us get into more details on the functioning of the F-measure. When $\zeta \to 0$, we get $F_\zeta(P, R) \to P$. Then optimizing the F-measure means we are interested *only* in maximizing the *Precision*. On the other hand, when $\zeta \to 1$, we get $F_\zeta(P, R) \to R$. In this case we are interested *only* in maximizing the *Recall*. The user can choose an appropriate value for ζ depending on how much of importance he/she wants to give to the precision and recall. Thus, the F-measure combines precision and recall into a single optimization criterion by taking their ζ-weighted harmonic mean. In the imbalanced data case mentioned above MCR minimization can potentially result in $F_\zeta(P, R) \to 0$. By maximizing the F-measure we can prevent the classifier from being completely biased towards the negative class. Note that $F_\zeta(P, R)$ can be re-written in terms of a, b and c as:

$$F_\zeta(a, b, c) = \frac{a}{a + \zeta b + (1 - \zeta)c} \tag{7}$$

7

In all our experiments we set $\zeta = 0.5$. In this case, it becomes $F_{0.5}(P, R) = \frac{2PR}{P+R}$ and can also be written as: $F_{0.5}(a, b, c) = \frac{1}{1+\frac{b+c}{2a}}$. Then, maximizing $F_{0.5}(a, b, c)$ is equivalent to maximizing $\frac{a}{b+c}$. Thus, we can maximize $F_{0.5}(a, b, c)$ by both minimizing the error (that is, $b+c$) and maximizing the true positives. The trade-off kicks-in since maximizing the true positives tends to increase the false positives. Thus maximizing the F-measure controls both the true positives and the error appropriately. In general the F-measure summarizes a classifier's ability to identify the positive class and plays an important role in the evaluation of binary classifier. As a criterion for optimizing hyperparameters F-measure can be computed on an evaluation or validation dataset. However, in practical situations involving small datasets[2] it is wasteful to employ a separate evaluation set. The LOO-CV approach would be useful in such situations and we show next how the F-measure can be estimated with such approach.

Hong et al. (2007) estimated TP and FP as:

$$\widehat{TP} = \frac{1}{n_+} \sum_{i=1}^{n} T(\hat{y}_{\setminus i} y_i, y_i),$$

$$\widehat{FP} = \frac{1}{n_-} \sum_{i=1}^{n} F(\hat{y}_{\setminus i} y_i, y_i).$$

Here $\hat{y}_{\setminus i}$ represents predicted label for ith sample. Therefore $\hat{y}_{\setminus i} y_i$ takes value $+1$ when the prediction matches with the actual label and -1 otherwise. $T(u, v)$ is an indicator function which is 1 if $u = 1$ and $v = 1$. Similarly, $F(u, v)$ is one if $u = -1$ and $v = -1$. Otherwise, these functions take zero values. Hong et al. (2007) used these estimates to compute $\widehat{AUC} = \frac{1+\widehat{TP}-\widehat{FP}}{2}$ as an approximation of AUC and used this criterion to select a subset of basis vectors in a kernel classifier model construction procedure for imbalanced datasets. Note that this definition of AUC is applicable only for a hard classifier (fixed non-probabilistic classifier) with binary outputs. See Hong et al. (2007) for more details. In a strict sense such a definition of AUC is not suitable for a probabilistic classifier like GP classifier that provides continuous probabilistic output. However, we can make use of this approach of defining TP and FP as above to compute the quantities a, b and c that are needed to evaluate the F-measure in (7). There are two issues associated with these estimates. The first issue is that these estimates are not smooth (in fact, not even continuous) functions of hyperparameters. Therefore they cannot be used directly in any approach that uses gradient-based nonlinear optimization methods to tune the hyperparameters. Secondly, these estimates do not use predictive probability values which is particularly important when we want to take variance also into account. In non-GP contexts Jansche (2005) and Keerthi et al. (2006) addressed the first issue by defining smoothed F-measure or other validation functions by replacing the indicator function with a sigmoid function, which makes the optimization criterion as a smooth function of hyperparameters. However, they did not consider a LOO approach and used a validation set instead. Jansche (2005) considered maximum *a posterior* probabilities and Keerthi et al. (2006) used sigmoidal approximations for SVM models. Here, we propose to combine LOO based estimation and smoothed version of the quantities $\{a, b, c, d\}$ denoted

---

2. GP models are known to be particularly valuable for problems with small datasets.

as $A(\boldsymbol{\theta})$, $B(\boldsymbol{\theta})$, $C(\boldsymbol{\theta})$ and $D(\boldsymbol{\theta})$. We can set

$$A(\boldsymbol{\theta}) \; = \; \sum_{i:y_i=+1} p(y_i = +1|x_i, S_{\setminus i}, \boldsymbol{\theta}) \tag{8}$$

Since $n_+ = a + b$, we can write $B(\boldsymbol{\theta}) = n_+ - A(\boldsymbol{\theta})$. With m_+ denoting the number of examples predicted as positive, we can parameterize it as $m_+(\boldsymbol{\theta}) = A(\boldsymbol{\theta}) + C(\boldsymbol{\theta})$. This can be rewritten as:

$$m_+(\boldsymbol{\theta}) \; = \; \sum_{i=1}^{n} p(y_i = +1|x_i, S_{\setminus i}, \boldsymbol{\theta}) \tag{9}$$

Thus, the smoothed F-measure can be defined from (7) as:

$$F_\zeta(\boldsymbol{\theta}) \; = \; \frac{A(\boldsymbol{\theta})}{\zeta n_+ + (1 - \zeta)m_+(\boldsymbol{\theta})} \tag{10}$$

Note that $D(\boldsymbol{\theta})$ can be defined in a similar fashion as $m_-(\boldsymbol{\theta}) = B(\boldsymbol{\theta}) + D(\boldsymbol{\theta})$. Using these quantities, other derived quantities like $TP(\boldsymbol{\theta})$ and $FP(\boldsymbol{\theta})$ can be defined as LOO based estimates. Then, smoothed LOO estimates of WER can be obtained as shown below.

The WER measure is another useful measure for imbalanced datasets. Using the quantities defined above, its smoothed version can be written as:

$$WER(\boldsymbol{\theta}; \tau) \; = \; \frac{n_+(1 - TP(\boldsymbol{\theta})) + \tau n_- FP(\boldsymbol{\theta})}{n_+ + \tau n_-}. \tag{11}$$

where τ is the ratio of the cost of mis-classifications of the negative class to that of the positive class and $0 \leq \tau \leq 1$. Thus by choosing a suitable τ value for a given problem and optimizing over the hyperparameters we can design classifiers without becoming biased toward one class. Note that for ease of notation we have omitted hat on $TP(\cdot)$ and $FP(\cdot)$. Following the work of Hong et al. (2007) one can also define

$$AUC(\boldsymbol{\theta}) \; = \; \frac{1 + TP(\boldsymbol{\theta}) - FP(\boldsymbol{\theta})}{2} \tag{12}$$

and optimize over the hyperparameters. As mentioned earlier, such a definition is not suitable for the GP classifier. Nevertheless it is interesting to note that it has the desirable property of trading-off between high TP and low FP. Also, on comparing this definition of AUC with (11) we see that they are related in the sense that maximizing AUC is same as minimizing WER when $\tau = 1$ and $n_+ = n_-$.

Overall we see that the LOO-CV predictive distributions can be used to define various criteria that are smooth functions of hyperparameters resulting in smoothed LOO-CV measures. Now given that the LOO-CV predictive distributions are readily available from the EP algorithm, we can optimize the various smoothed LOO-CV measures directly using standard non-linear optimization techniques.

4. EP-CV Algorithm for Choosing Hyperparameters

Various criteria such as (6), (10), (11) and (12) depend on the hyperparameters $\boldsymbol{\theta}$ via the predictive distributions $p(y_i|x_i, S_{\setminus i}, \boldsymbol{\theta})$. With cumulative Gaussian likelihood, they can be

written as:

$$p(y_i|x_i, S_{\backslash i}, \boldsymbol{\theta}) \; = \; \Phi\Big(\frac{y_i(\mu_{\backslash i} + \gamma)}{\sqrt{1 + \sigma_{\backslash i}^2}}\Big) \tag{13}$$

Note that (13) is obtained from (1) with $p(f_i|x_i, S_{\backslash i}, \boldsymbol{\theta}) = \mathcal{N}(\mu_{\backslash i}, \sigma_{\backslash i}^2)$. The hyperparameter γ is referred to as the bias parameter and it helps in shifting the decision boundary with the probability value $\frac{1}{2}$. In general, the bias hyperparameter γ is very useful (Rasmussen and Williams, 2006; Seeger, 2003) and can be optimized.

EP Approximation

To compute (13) we need the LOO mean $\mu_{\backslash i}$ and variance $\sigma_{\backslash i}^2 \; \forall i$. With the EP approximation, they can be computed using (4). Full details of gradient calculations needed for implementing hyperparameter optimization are given in the appendix.

We take the expectation-maximization (EM) type approach for hyperparameters optimization. This is because gradient expressions involving implicit derivatives (with site parameters varying as a function of hyperparameters) are not available due to the iterative nature of the EP algorithm. This approach results in the following algorithm.

EP-CV Algorithm:

1. Initialize the hyperparameters $\boldsymbol{\theta}$.

2. **Perform E-Step**: Given the hyperparameters, we find the site parameters $\boldsymbol{\mu}$ and Σ and the posterior $q(\mathbf{f}|S, \boldsymbol{\theta}) \; = \; \mathcal{N}(\mathbf{m}, \mathbf{C})$ using EP algorithm.

3. **Perform M-Step**: Find the hyperparameters $\boldsymbol{\theta}$ by optimizing over any LOO-CV based measure like (6), (10), (11) or (12) using any standard gradient based optimization technique. We carry out just one line search in this optimization process. During this line search as the hyperparameters change, we perform the following sequence of operations.

   (a) Compute the posterior mean $\mathbf{m}$ and covariance $\mathbf{C}$ using (3).
   (b) Compute the LOO mean $\mu_{\backslash i}$ and variance $\sigma_{\backslash i}^2$ using (4).
   (c) Compute the chosen objective function like (6), (10), (11) or (12) and its derivatives.

   Note that through out this M-step, it is assumed that the site parameters are fixed and the values obtained from step (2) are used.

4. repeat steps 2-3 until there is no significant change in the objective function value.

This algorithm worked well in our experiments. A similar EM approach was used by Kim and Ghahramani (2006) (which they called EM-EP algorithm) in the optimization of a lower bound on the marginal likelihood.

Since the EP-CV algorithm optimizes the smoothed F-measure it is useful to study the behavior of the true F-measure as optimization proceeds. We do this study on two of the datasets described in Table 6 of section 6. The optimization algorithm was terminated when
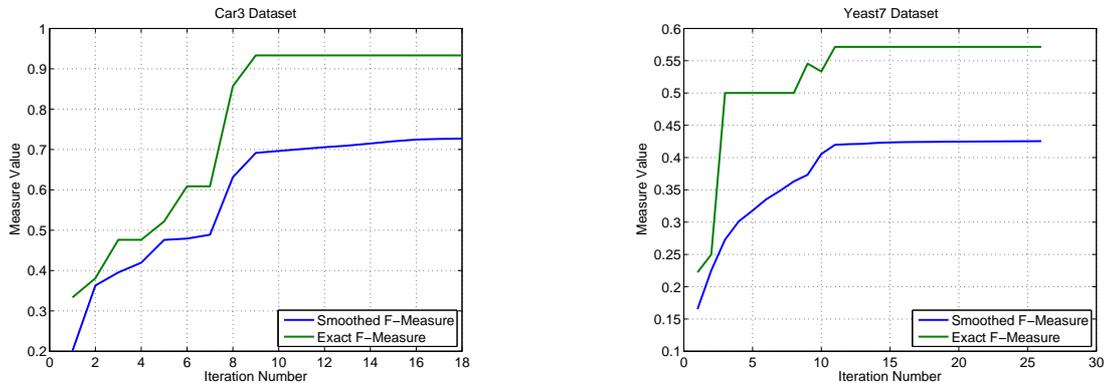
Figure 1: F-Measure Optimization on Car3 and Yeast7 Datasets

there was no significant change in the smoothed F-measure value. From Figure 1 we see that the smoothed F-measure monotonically increases in both cases. Also as expected, the true F-measure exhibits a non-smooth behavior expected of a discrete function, and also, the values of true and smoothed F-measures are not the same. The difference arises because the smoothed F-measure is based on probabilistic scores, which can take any value between 0.5 and 1 depending on the problem (even when correct classification occurs). The important point to observe is that, in general, there is an increasing trend in true F-measure value as the optimization progresses. In the case of Car3 dataset (left panel) we see that clearly. A similar trend is seen in the case of Yeast7 dataset (right panel) also, except for a small dip at the 10th iteration. Though such a behavior happens sometimes in early iterations, we observed that better true F-measure value is almost always obtained as the optimization progresses.

Computational and Storage Complexities

The computational complexity of the EP-CV algorithm depends on the number of ARD kernel parameters D. See appendix for more details. For a given problem with D fixed, the complexity is $O(n^3)$. This complexity is same as that of the EP(ML) method (see equation (5)) and LS-CV(NLP) method given in the next section. Also in many practical problems a single global scaling hyperparameter for all the input features is sufficient. Finally, the storage complexities of all the methods are $O(n^2)$.

5. Other LOO-CV based Methods

Having discussed our approach in detail it is useful to recall and discuss other LOO-CV based GPC model selection methods in relation to it.

Opper and Winther (2000) derived a mean field algorithm for binary classification with GPs based on the TAP approach originally proposed in the statistical physics of disordered systems. They showed that this approach yields an approximate LOO estimator for the generalization error. This estimate is equivalent to the LOO-CV error estimate obtained from EP (Minka, 2001). Instead of optimizing over the hyperparameters, Opper and

Winther (2000) used the LOO-CV error count (using indicator functions) to choose rough hyperparameter values by scanning a range of values.

Qi et al. (2004) used the LOO-CV error estimate obtained from EP to determine ARD hyperparameters. They worked with the Gaussian process classifier where each input feature is associated with a weight parameter v_i with the prior $\mathcal{N}(0, \zeta_i^{-1})$. The hyperparameters ζ_i^{-1} were obtained by *maximizing the evidence* using a fast sequential update based on the work of Faul and Tipping (2002). The outcome of this optimization is that many ζ_is would go to infinity such that only a few nonzero weights v_i will be present. Even though the ARD hyperparameters were optimized by maximizing the evidence, to prevent overfitting Qi et al. (2004) proposed to select the final model as the one that gives the minimum LOO-CV error count or probability. As the LOO-CV error count is discrete, they chose the model with maximum evidence when there is a tie in the count. Compared to this approach, we work with the GP classifier model (without the weight parameters) detailed in Section 2 and optimize over the hyperparameters (including ARD) directly with various LOO-CV based measures (including F-measure, WER etc.) using gradient information.

In this context, the LOO-CV based probabilistic LS classifier (Rasmussen and Williams, 2006) is a more direct LOO-CV based GPC model selection approach. For the sake of completion we give some details here and later compare our algorithms with this approach in our experiments. This approach treats classification as a regression problem. Note that the probabilistic interpretation of LS criterion implies a Gaussian noise model. But the output $\mathbf{y}$ can take only $+1$ or -1 which is slightly odd. However, this approach is simple to implement and a probabilistic interpretation is given by passing the predictions through a sigmoid.

Specifically, the LOO mean $\mu_{\backslash i}$ and variance $\sigma_{\backslash i}^2$, $i \in \tilde{I}$ are obtained from LOO-CV formulation of GP regression and the predictive distributions are obtained via (13). Here, $\mu_{\backslash i}$ and $\sigma_{\backslash i}^2$ are given by:

$$\mu_{\backslash i} = y_i - \tilde{\alpha}_i \sigma_{\backslash i}^2 \quad \sigma_{\backslash i}^2 = \frac{1}{\bar{\mathbf{K}}_{ii}} \tag{14}$$

where $\tilde{\boldsymbol{\alpha}} = \bar{\mathbf{K}} \mathbf{y}$ and $\bar{\mathbf{K}} = (\mathbf{K} + \lambda \mathbf{I})^{-1}$. Here λ can either be set to a small positive value or treated as a regularization hyperparameter with a small upper bound constraint. This is useful when $\mathbf{K}$ can become ill-conditioned during optimization. Finally, the hyperparameters are optimized using (6). We call this method as LS-CV(NLP).

6. Experiments

We conducted two experiments with various methods. See Table 2 for a summary of the various methods. In the first experiment we compared the performance of EP-CV(NLP) method with that of EP(ML) and LS-CV(NLP) methods. In the second experiment we compared the performance of EP-CV(FM) method with that of EP-CV(NLP) and two step classifier methods. We used the *minimize* Matlab routine of the GPML Matlab code available at *http://www.gaussianprocess.org/gpml/code/matlab/doc/* for hyperparameters optimization. In all the experiments we used a single global scaling hyperparameter.

Table 2: Various methods and their descriptions. All the EP-CV methods are optimized using EP-CV algorithm.

METHOD	DESCRIPTION
EP(ML)	MARGINAL LIKELIHOOD MAXIMIZATION WITHIN EP APPROXIMATION. THAT IS, OPTIMIZE (5) OVER $\boldsymbol{\theta}$.
EP-CV(NLP)	NEGATIVE LOGARITHMIC PREDICTIVE LOSS MINIMIZATION WITHIN EP APPROXIMATION. THAT IS, OPTIMIZE (6) OVER $\boldsymbol{\theta}$. FOR EASE OF NOTATION, THIS METHOD IS REFERRED AS **NLP** IN TABLE 7.
LS-CV(NLP)	NEGATIVE LOGARITHMIC PREDICTIVE LOSS MINIMIZATION WITHIN LEAST SQUARES APPROXIMATION AS DESCRIBED IN SECTION 5. OPTIMIZE (6) OVER $\boldsymbol{\theta}$.
EP-CV(FM)	F-MEASURE MAXIMIZATION WITHIN EP-CV APPROXIMATION. THAT IS, OPTIMIZE (10) OVER $\boldsymbol{\theta}$. FOR EASE OF NOTATION, THIS METHOD IS REFERRED AS **FM** IN TABLE 7.
NLP-FM(BIAS)	IN THE FIRST STEP, HYPERPARAMETERS ARE OPTIMIZED USING EP-CV(NLP) METHOD. IN THE SECOND STEP, ONLY THE BIAS PARAMETER (γ) IS OPTIMIZED USING EP-CV(FM) METHOD. WE ALSO REFER THIS METHOD AS TWO-STEP METHOD.

Table 3: Data sets description: NLP Experiment. Here, n, D, p and nr represent the numbers of training examples, input dimension, test examples and train/test partitions respectively.

DATASET	n	D	p	nr
BANANA	400	2	4900	100
BREASTCANCER	200	9	77	100
DIABETES	468	8	300	100
GERMAN	700	20	300	100
HEART	170	13	100	100
IMAGE	1300	18	1010	20
RINGNORM	400	20	7000	100
SPLICE	1000	60	2175	20
THYROID	140	5	75	100
TITANIC	150	3	2051	100
TWONORM	400	20	7000	100
WAVEFORM	400	21	4600	100

Table 4: NLP Performance

Dataset/Method	EP(ML)	EP-CV(NLP)	LS-CV(NLP)
Banana	23.90 ± 0.81	24.26 ± 1.06	33.88 ± 1.89
Breastcancer	53.57 ± 4.75	54.12 ± 5.27	55.58 ± 5.03
Diabetes	47.74 ± 1.96	47.97 ± 2.09	50.72 ± 2.13
German	48.67 ± 2.74	49.05 ± 2.76	50.51 ± 2.30
Heart	40.16 ± 5.36	40.03 ± 5.00	45.11 ± 4.91
Image	8.26 ± 1.07	8.45 ± 0.97	22.70 ± 0.66
Ringnorm	16.88 ±0.93	16.56 ± 1.01	28.48 ± 0.75
Solar	57.25 ± 1.38	57.35 ± 1.42	59.61 ± 1.34
Splice	28.48 ± 0.88	29.60 ± 0.79	36.83 ± 0.42
Thyroid	10.21 ± 3.76	9.94 ± 3.69	25.33 ± 4.86
Titanic	66.86 ± 1.97	51.73 ± 1.73	53.78 ± 14.08
Twonorm	8.31 ± 0.88	9.08 ± 1.97	25.94 ± 0.53
Waveform	23.01 ± 0.89	22.97 ± 0.67	32.63 ± 0.59

Table 5: Test Set Error Performance

Dataset/Method	EP(ML)	EP-CV(NLP)	LS-CV(NLP)
Banana	10.41 ± 0.65	10.51 ± 0.50	10.93 ± 0.67
Breastcancer	26.52 ± 4.89	26.61 ± 4.80	25.94 ± 4.59
Diabetes	23.28 ± 1.82	23.41 ± 1.82	24.30 ± 2.51
German	23.36 ± 2.11	23.48 ± 2.00	23.94 ± 2.33
Heart	16.65 ± 2.87	16.62 ± 3.08	17.91 ± 4.21
Image	2.82 ± 0.54	2.77 ± 0.51	2.74 ± 0.65
Ringnorm	4.41 ± 0.64	4.29 ± 0.69	5.05 ± 0.99
Solar	34.20 ± 1.75	34.27 ± 1.80	35.03 ± 1.89
Splice	11.61 ± 0.81	11.85 ± 0.83	11.83 ± 0.80
Thyroid	4.37 ± 2.19	4.20 ± 2.17	6.97 ± 3.78
Titanic	22.64 ± 1.34	22.50 ± 0.98	22.99 ± 2.81
Twonorm	3.05 ± 0.34	3.19 ± 0.51	3.43 ± 0.43
Waveform	10.10 ± 0.48	9.95 ± 0.48	11.70 ± 0.88

Table 6: Data sets description: F-Measure Experiment. Here, n, D, p, nr and PPE represent the numbers of training examples, input dimension, test examples, train/test partitions and approximate percentage of positive examples respectively.

DATASET	n	D	p	nr	PPE
YEAST7	297	8	1187	50	2
YEAST5	320	8	1164	50	4
CAR3	350	6	1378	50	4
ECOLI5	124	7	212	50	6
YEAST4	165	8	1319	50	11
BREASTCANCER	200	9	77	100	29
GERMAN	700	20	300	100	30
DIABETES	468	8	300	100	35

Table 7: F-Measure Performance

DATASET/METHOD	NLP	FM	NLP-FM(BIAS)
YEAST7	32.24 ± 15.54	42.58 ± 7.84	40.85 ± 8.59
YEAST5	19.79 ± 12.85	32.85 ± 8.56	28.45 ± 10.57
CAR3	55.89 ± 9.30	64.35 ± 8.51	62.67 ± 8.49
ECOLI5	84.41 ± 6.25	83.79 ± 5.90	84.08 ± 5.86
YEAST4	71.35 ± 3.95	73.64 ± 2.97	73.23 ± 2.81
BREASTCANCER	38.42 ± 10.55	47.34 ± 6.44	46.98 ± 6.37
GERMAN	54.15 ± 4.17	57.53 ± 2.95	56.91 ± 2.87
DIABETES	62.69 ± 3.49	66.23 ± 2.87	66.12 ± 2.67

6.1 NLP Experiment

In this experiment we used the thirteen benchmark datasets available in the web[3] summarized in Table 3. Let us first consider the results from the first experiment given in Table 4 and Table 5. For the EP(ML) method we used the GPML Matlab code available in the web[4].

We conducted Friedman test (Demsar, 2006) with the corresponding post-hoc tests for comparison of classifiers over multiple datasets. The comparison over multiple datasets requires a performance score of each method on each dataset (Demsar, 2006). Here, we consider the mean over the partitions of a given dataset as the performance score. As pointed out in Demsar (2006), it is not clear how to make use of the standard deviation information when the datasets are not independent over the partitions. The Friedman test ranks the methods for each dataset separately based on the chosen performance score (mean performance in our case). The best performing method gets the rank of 1, the second best rank 2 and so on. In case of ties, average ranks are assigned. The Friedman test checks whether the measured average ranks (over the datasets) are significantly different from the mean rank under the null hypothesis. Under the null hypothesis all the methods are equivalent and so their ranks should be equal.

In the case of NLP performance measure, the measured average ranks for the EP(ML), EP-CV(NLP) and LS-CV(NLP) methods were 1.46, 1.62 and 2.92 respectively. With three methods and 13 datasets, the F-statistic comparison at a significance level of 0.05 rejected the null hypothesis. Since the null hypothesis was rejected we conducted the Nemenyi post-hoc test for pairwise comparisons. This test revealed that the results of the EP(ML) and EP-CV(NLP) methods are better than the LS-CV(NLP) method at the significance level of 0.05. On the other hand, the post-hoc test did not detect any significant difference in the results of EP(ML) and EP-CV(NLP) methods. In the case of test set performance measure, the measure averaged ranks for the EP(ML), EP-CV(NLP) and LS-CV(NLP) methods were 1.62, 1.77 and 2.62 respectively. Note that the average rank of LS-CV method has improved on the test set error performance. Here again, the null hypothesis is rejected at the same significance level and the post-hoc test did not detect any significant difference in the results of EP(ML) and EP-CV(NLP) methods. The results of the EP(ML) and EP-CV(NLP) methods are better than the LS-CV(NLP) method at the significance level of 0.05 and 0.1 respectively. Thus, we can conclude that EP(ML) and EP-CV(NLP) are competitive to each other. Further, both these methods perform better than the LS-CV(NLP) method in this experiment.

We can also make other observations from the tables. We note that the NLP performance of the LS-CV(NLP) method is quite inferior on several datasets even though its test set error performances on most of these datasets (except *waveform and thyroid*) are relatively closer. Further, some kind of group behavior can be seen. For example, the NLP scores are high on *titanic*, *breast-cancer*, *diabetes*, *German* and *flare-solar*. Also, the test set errors are $> 20\%$ on these datasets. Consequently, we may consider these datasets as difficult ones. From Table 4 we observe that the NLP performance of LS-CV(NLP) is *closer* to the other two methods on these datasets (compared to its performance on

---

3. *http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm*
4. *http://www.gaussianprocess.org/gpml/code/matlab/doc/*

other datasets). Next we can order the remaining datasets *heart*, *splice*, *banana*, *waveform*, *ringnorm*, *thyroid*, *twonorm* and *image* in terms of descending difficulty. Note that the difference in the NLP performance seems to have an increasing trend as the dataset becomes easier. To understand this we looked at the predictive probabilities of these methods on both correctly and wrongly classified examples. In the case of thyroid dataset these average probability scores for the LS-CV(NLP) method were 0.84 (for correct classification) and 0.35 (wrong classification). Here, the averaging was done over all the correctly(wrongly) classified examples over all the partitions. On the other hand, the corresponding values for the EP-CV(NLP) were $(0.96, 0.39)$. In the case of banana dataset, these scores were $(0.79, 0.35)$ and $(0.92, 0.29)$ for the LS-CV(NLP) and EP-CV(NLP) methods respectively. In the case of German dataset, they were $(0.75, 0.33)$ and $(0.79, 0.32)$. The scores for the EP-ML method were very close to that of the EP-CV(NLP) method. In general, we observed that the predictive probability estimates from the LS-CV(NLP) method were relatively poor and resulted in poor NLP performance. We looked at the hyperparameter estimates of the different methods and observed that the LS-CV(NLP) method takes smaller width and signal variance hyperparameter values on most of the datasets except on the difficult datasets (mentioned above) compared to the other two methods. Apart from this we did not observe any specific pattern in the hyperparameter values chosen by these methods. Looking at the hyperparameter estimates of EP(ML) and EP-CV(NLP), it seems that several solutions in the space of hyperparameters that give close performances are possible.

6.2 F-measure Experiment

The datasets used in this experiment are described in Table 6. The datasets *yeast*, *car* and *ecoli* are multi-class datasets and we converted them into binary classification datasets by considering examples belonging to the class label indicated by the number (for example, 7 in yeast7) as positive class respectively and treating the rest of the examples as negative class. These datasets are available in the web [5]. We created 50 partitions for these datasets in a stratified manner reflecting the class distributions.

Let us consider the results from the second experiment given in Table 7. In this experiment all the results were obtained within the EP-CV framework. The first and second columns represent results obtained using NLP and smoothed F-measure (i.e., eq. (10)) as the optimization criterion respectively. The third column represents results obtained from the two step classifier described earlier. We looked at the hyperparameter estimates of the different methods. We observed that the bias estimates of the two step classifier were somewhat closer (within 10%) to those of the smoothed F-measure method on the *breast-cancer*, *diabetes* and *German* datasets. On the remaining datasets they were different by more than 40%. The width and signal variance hyperparameter estimates were also quite different. From Table 7, we observe that the two step method is also good and gives closer performance to the smoothed F-measure method on several datasets. Further analysis of the performance results revealed that even though the standard deviations are high, the smoothed F-measure method gave better performance than the two step method on majority of the partitions on several datasets. We believe that the larger standard deviations in the results arises from the sensitivity to the dataset with lesser number of positive examples.

---

5. *ftp://ftp.ics.uci.edu/pub/machine-learning-databases/*

To carry out the statistical significance tests, again we used the mean (over the partitions) as the performance score for each of the methods. The measured average ranks for these three methods were 2.75, 1.25 and 2.00 respectively. With three methods and 8 datasets, the F-statistic comparison at a significance level of 0.05 rejected the null hypothesis. The Nemenyi post-hoc test for pairwise comparisons revealed that *only* the results of smoothed F-measure is better than the NLP based method at the significance level of 0.05. In this experiment the post-hoc test did not detect any significant differences in the comparisons of smoothed F-measure method with the two step method and the two step method with the NLP method. However in these two comparisons the rank differences were closer to the required critical differences at the significance level of 0.1. We also observed that if we were to conduct Wilcoxon signed-rank test on these methods (as if we were comparing only two classifiers) then the results were statistically significant at the significance level of 0.05 for all the three pairs. In summary, the results demonstrate the usefulness of direct optimization of smoothed F-measure.

7. Conclusion

In this paper, we considered the problem of Gaussian process classifier model selection with different LOO-CV based optimization criteria and provided a practical algorithm using LOO predictive distributions with criteria like standard NLP, smoothed F-measure and WER to select hyperparameters. More specifically, apart from optimization of standard NLP, we demonstrated its usefulness in direct optimization of smoothed F-measure, which is useful to handle imbalanced data. We considered predictive distribution arrived from the Expectation Propagation (EP) approximation. We derived relevant expressions and proposed a very useful EP-CV algorithm. The experimental results on several real world benchmark datasets showed comparable NLP generalization performance (with NLP optimization) with existing approaches. We demonstrated that the smoothed F-measure optimization method is a very useful method that improves the F-measure performance significantly. Overall, the EP-CV algorithm is an excellent choice for GP classifier model selection with different LOO-CV based optimization criteria.

References

J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

A. C. Faul and M. E. Tipping. Analysis of sparse Bayesian learning. *Advances In Neural Information Processing Systems*, 14, 2002.

M. Gibbs and D. J. C. MacKay. Variational Gaussian process classifiers. *IEEE Transactions on Neural Networks*, 11(6):1458–1464, 2000.

X. Hong, S. Chen, and C. J. Harris. A kernel based two-class classifier for imbalanced data sets. *IEEE Transactions on Neural Networks*, 18(1):28–41, 2007.

M. Jansche. Maximum expected f-measure training of logistic regression models. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in*

Natural Language Processing, pages 692–699. Association for Computational Linguistics, NJ, USA, 2005.

T. Joachims. A support vector method for multivariate performance measures. In *ICML*, pages 377–384, 2005.

S. S. Keerthi, V. Sindhwani, and O. Chapelle. An efficient method for gradient based adaptation of hyperparameters in SVM models. *Advances In Neural Information Processing Systems*, 18, 2006.

H. C. Kim and Z. Ghahramani. Bayesian Gaussian process classification with the EM-EP algorithm. *IEEE Transactions on Pattern Analysis and Maching Intelligence*, 28(12): 1948–1959, 2006.

M. Kuss and C. E. Rasmussen. Assessing approximate inference for Bayesian Gaussian process classification. *Journal of Machine Learning Research*, 6:1679–1704, 2005.

T. P. Minka. Expectation propagation for approximate Bayesian inference. In *UAI*, pages 362–369. Morgan Kaufmann, 2001.

R. Neal. Regression and classification using Gaussian process priors. *Bayesian Statistics*, 6:475–501, 1998.

M. Opper and O. Winther. Gaussian processes for classification: Mean-field algorithms. *Neural Computation*, 12:2655–2684, 2000.

Y. Qi, T. P. Minka, R. W. Picard, and Z. Ghahramani. Predictive automatic relevance determination by expectation propagation. In *ICML*, pages 362–369, 2004.

C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

M. Seeger. Bayesian Gaussian process models: Pac-Bayesian generalization error bounds and sparse approximation. In *Phd Thesis*. University of Edinburgh, 2003.

M. Seeger. Cross-validation optimization for large scale hierarchical classification kernel methods. *Advances In Neural Information Processing Systems*, 19, 2007.

S. Sundararajan and S. S. Keerthi. Predictive approaches for choosing hyperparameters in Gaussian processes. *Neural Computation*, 13(5):1103–1118, 2001.

van Rijsbergen. Foundation of evaluation. *Journal of Documentation*, 30(4):365–373, 1974.

C. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

Appendix: Optimization criteria, approximate predictive distributions and their derivatives

From the definitions of various optimization criteria like NLP measure (6) and (10) etc, we see that the chosen measure and its derivatives can be obtained from the LOO predictive distributions $p(y_i|x_i, S_{\backslash i}, \boldsymbol{\theta})$ and their derivatives $\frac{\partial p(y_i|x_i, S_{\backslash i}, \boldsymbol{\theta})}{\partial \theta_j}$. Here, θ_j is j^{th} component of the hyperparameter vector $\boldsymbol{\theta}$. Note that the derivative of the NLP measure (6) is given by:

$$\frac{\partial G(\boldsymbol{\theta})}{\partial \theta_j} = -\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\Phi(y_i z_i)}\frac{\partial p(y_i|x_i, S_{\backslash i}, \boldsymbol{\theta})}{\partial \theta_j} \tag{15}$$

and the derivative of the smoothed F-measure (10) is given by:

$$\frac{\partial F_\zeta(\boldsymbol{\theta})}{\partial \theta_j} = \frac{\eta(\boldsymbol{\theta})\frac{\partial A(\boldsymbol{\theta})}{\partial \theta_j} - A(\boldsymbol{\theta})(1-\zeta)\frac{\partial m_+(\boldsymbol{\theta})}{\partial \theta_j}}{\eta^2(\boldsymbol{\theta})} \tag{16}$$

where $\eta(\boldsymbol{\theta}) = \zeta n_+ + (1-\zeta)m_+(\boldsymbol{\theta})$. Note that the derivatives $\frac{\partial A(\boldsymbol{\theta})}{\partial \theta_j}$ and $\frac{\partial m_+(\boldsymbol{\theta})}{\partial \theta_j}$ are directly dependent on $\frac{\partial p(y_i=+1|x_i, S_{\backslash i}, \boldsymbol{\theta})}{\partial \theta_j}$. Now, from (13) we see that to define the LOO predictive distributions, we need the LOO mean $\mu_{\backslash i}$ and variance $\sigma^2_{\backslash i}$. In the case of EP approximation, analytical expressions to compute these quantities are already available (see eqn. (4)). Next, we give details on how the derivatives of predictive distributions can be obtained with these approximations.

Derivatives of Predictive Distributions

For ease of reference we recall (13) here.

$$p(y_i|x_i, S_{\backslash i}, \boldsymbol{\theta}) = \Phi\Big(\frac{y_i(\mu_{\backslash i} + \gamma)}{\sqrt{1 + \sigma^2_{\backslash i}}}\Big).$$

Then, with $z_i = \frac{\mu_{\backslash i} + \gamma}{\sqrt{1 + \sigma^2_{\backslash i}}}$ and $N(z_i) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{z_i^2}{2})$ we have

$$\frac{\partial p(y_i|\mathbf{x}_i, S_{\backslash i}, \boldsymbol{\theta})}{\partial \theta_j} = \frac{N(z_i)y_i}{\sqrt{1 + \sigma^2_{\backslash i}}}\Big(\frac{\partial \mu_{\backslash i}}{\partial \theta_j} - \frac{1}{2}\frac{z_i}{\sqrt{1 + \sigma^2_{\backslash i}}}\frac{\partial \sigma^2_{\backslash i}}{\partial \theta_j}\Big).$$

Here, θ_j represents any element of $\boldsymbol{\theta}$ other than γ. Similarly, we have

$$\frac{\partial p(y_i|\mathbf{x}_i, S_{\backslash i}, \boldsymbol{\theta})}{\partial \gamma} = \frac{N(z_i)y_i}{\sqrt{1 + \sigma^2_{\backslash i}}}.$$

Thus, we need $\frac{\partial \mu_{\backslash i}}{\partial \theta_j}$ and $\frac{\partial \sigma^2_{\backslash i}}{\partial \theta_j}$. Below, we give details on how they can be obtained with the EP approximation.

Derivatives of LOO mean and variance with fixed site parameters: EP Approximation

In the case of EP approximation, since we resort to EM type optimization, we derive expressions for $\frac{\partial \mu_{\backslash i}}{\partial \theta_j}$ and $\frac{\partial \sigma^2_{\backslash i}}{\partial \theta_j}$ assuming that the site parameters are fixed. From $\mu_{\backslash i} = \sigma^2_{\backslash i}(\frac{m_i}{\mathbf{C}_{ii}} - \frac{\mu_i}{\sigma^2_i})$, we have

$$\frac{\partial \mu_{\backslash i}}{\partial \theta_j} = \frac{\mu_{\backslash i}}{\sigma^2_{\backslash i}} \frac{\partial \sigma^2_{\backslash i}}{\partial \theta_j} + \frac{\sigma^2_{\backslash i}}{(\mathbf{C}_{ii})^2}\Big(\mathbf{C}_{ii}\frac{\partial \mathbf{m}_i}{\partial \theta_j} - \mathbf{m}_i\frac{\partial \mathbf{C}_{ii}}{\partial \theta_j}\Big).$$

From $\sigma^2_{\backslash i} = ((\mathbf{C}_{ii})^{-1} - \sigma_i^{-2})^{-1}$, we have

$$\frac{\partial \sigma^2_{\backslash i}}{\partial \theta_j} = \frac{\sigma^4_{\backslash i}}{(\mathbf{C}_{ii})^2}\frac{\partial \mathbf{C}_{ii}}{\partial \theta_j}.$$

Since $\mathbf{m} = \mathbf{C}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$, we have $\frac{\partial \mathbf{m}}{\partial \theta_j} = \frac{\partial \mathbf{C}}{\partial \theta_j}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$. Note that $\mathbf{C}$ can be re-written using Sherman-Morrison-Woodbury formula as: $\mathbf{C} = \mathbf{K} - \mathbf{K}(\mathbf{K} + \boldsymbol{\Sigma})^{-1}\mathbf{K}$ and it is useful to work with this expression to achieve improved numerical stability (Rasmussen and Williams, 2006) as it avoids inversion of $\mathbf{K}$. Then we have

$$\frac{\partial \mathbf{C}}{\partial \theta_j} = (\mathbf{I} - (\mathbf{K} + \boldsymbol{\Sigma})^{-1}\mathbf{K})^T\frac{\partial \mathbf{K}}{\partial \theta_j}(\mathbf{I} - (\mathbf{K} + \boldsymbol{\Sigma})^{-1}\mathbf{K}).$$

Note that $\frac{\partial \mathbf{C}_{ii}}{\partial \theta_j}$, $i \in \tilde{I}$ (where $\tilde{I} = \{1, 2, \ldots, n\}$) are nothing but the diagonal entries of the above expression. Note also that $\frac{\partial \mathbf{m}}{\partial \theta_j}$ can be efficiently computed by taking advantage of the presence of the vector $\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$. But, to compute $\frac{\partial \mathbf{C}_{ii}}{\partial \theta_j}$, $i \in \tilde{I}$ we cannot avoid the matrix multiplication with $\frac{\partial \mathbf{K}}{\partial \theta_j}$; this results in $O(n^3)$ for each θ_j. Finally, it is useful to re-write $(\mathbf{K} + \boldsymbol{\Sigma})^{-1} = \boldsymbol{\Sigma}^{-\frac{1}{2}}(\mathbf{I} + \boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{K}\boldsymbol{\Sigma}^{-\frac{1}{2}})^{-1}\boldsymbol{\Sigma}^{-\frac{1}{2}}$ (Rasmussen and Williams, 2006).