Outlier Robust Gaussian Process Classification

Hyun-Chul Kim[1] and Zoubin Ghahramani[2]

[1] Yonsei University, 262 Seongsanno, Seodaemun-gu, Seoul 120-749, Korea
[2] University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK

Abstract. Gaussian process classifiers (GPCs) are a fully statistical model for kernel classification. We present a form of GPC which is robust to labeling errors in the data set. This model allows label noise not only near the class boundaries, but also far from the class boundaries which can result from mistakes in labelling or gross errors in measuring the input features. We derive an outlier robust algorithm for training this model which alternates iterations based on the EP approximation and hyperparameter updates until convergence. We show the usefulness of the proposed algorithm with model selection method through simulation results.

1 Introduction

In many real-world classification problems the labels provided for the data are noisy. There are typically two kinds of noise in labels. Noise near the class boundaries often occurs because it is hard to consistently label ambiguous data points. Labelling errors *far* from the class boundaries can occur because of mistakes in labelling or gross errors in measuring the input features. We call labelling errors far from the boundary, *outliers*. While many methods have been proposed for dealing with noisy class boundaries, far less attention has been placed on dealing with classification outliers. In this paper we present a Bayesian kernel classification algorithm which is robust to outliers.

GPCs are a Bayesian kernel classifier derived from Gaussian process priors over functions which were developed originally for regression [1,2,3]. In classification, the target values are discrete class labels. To use Gaussian processes for binary classification, the Gaussian process regression model can be modified so that the sign of the continuous latent function it outputs determines the class label. Observing the class label at some data point constrains the function value to be positive or negative at that point, but leaves it otherwise unknown. To compute predictive quantities of interest we therefore need to integrate over the possible unknown values of this function at the data points. Exact evaluation of this integral is computationally intractable. However, several successful methods have been proposed for approximately integrating over the latent function values, such as the Laplace approximation [1], Markov Chain Monte Carlo [3], and variational approximations [2]. Opper and Winther (2000) used the TAP approach originally proposed in statistical physics of disordered systems to integrate over the latent values [4]. The TAP approach for this model is equivalent

N. da Vitora Lobo et al. (Eds.): SSPR&SPR 2008, LNCS 5342, pp. 896–905, 2008.

to the more general Expectation Propagation (EP) algorithm for approximate inference [5]. Minka's formulation [5] has a special hyperparameter which can potentially be used to deal with outliers. Since outliers can be a big obstacle to learning, in this paper we propose an outlier robust learning algorithm based on EP for Gaussian process classification. It turns out that the algorithm is useful with the model selection between it and the regular GPC.

The paper is organized as follows. Section 2 introduces Gaussian process classification. In section 3, we introduce the EP/TAP method for approximate inference. In section 4, we derive the algorithm for outlier treatment. In section 5, we present model selection method based on Bayesian model evidence. In section 6, we show experimental results on both synthetic and real data sets. In section 7, we discuss our approach and future work.

2 Gaussian Process Classification

Let us assume that we have a data set D of data points $\mathbf{x}_i$ with binary class labels $y_i \in \{-1, 1\}$: $D = \{(\mathbf{x}_i, y_i)|i = 1, 2, \ldots, n\}, X = \{\mathbf{x}_i|i = 1, 2, \ldots, n\}, Y = \{y_i|i = 1, 2, \ldots, n\}$. Given this data set, we wish to find the correct class label for a new data point $\tilde{\mathbf{x}}$. We do this by computing the class probability $p(\tilde{y}|\tilde{\mathbf{x}}, D)$.

We assume that the class label is obtained by transforming some real valued latent variable $\tilde{f}$, which is the value of some latent function $f(\cdot)$ evaluated at $\tilde{\mathbf{x}}$. We put a Gaussian process prior on this function, meaning that any number of points evaluated from the function have a multivariate Gaussian density (see [6] for a review of GPs). Assume that this GP prior is parameterized by Θ which we will call the hyperparameters. We can write the probability of interest given Θ as:

$$p(\tilde{y}|\tilde{\mathbf{x}}, D, \Theta) = \int p(\tilde{y}|\tilde{f}, \Theta) p(\tilde{f}|D, \tilde{\mathbf{x}}, \Theta)\, d\tilde{f} \tag{1}$$

The second part of Eq 1 is obtained by further integration over $\mathbf{f} = [f_1 f_2 \cdots f_n]$, the values of the latent function at the data points.

$$p(\tilde{f}|D, \tilde{\mathbf{x}}, \Theta) = \int p(\mathbf{f}, \tilde{f}|D, \tilde{\mathbf{x}}, \Theta)\, d\mathbf{f} = \int p(\tilde{f}|\tilde{\mathbf{x}}, \mathbf{f}, \Theta) p(\mathbf{f}|D, \Theta) d\mathbf{f} \tag{2}$$

where $p(\tilde{f}|\tilde{\mathbf{x}}, \mathbf{f}, \Theta) = p(\tilde{f}, \mathbf{f}|\tilde{\mathbf{x}}, X, \Theta)/p(\mathbf{f}|X, \Theta)$ and

$$p(\mathbf{f}|D, \Theta) \propto p(Y|\mathbf{f}, X, \Theta) p(\mathbf{f}|X, \Theta) = \{\prod_{i=1}^{n} p(y_i|f_i, \Theta)\} p(\mathbf{f}|X, \Theta). \tag{3}$$

The first term in 3 is the likelihood for each observed class given the latent function value, while the second term is the GP prior over functions evaluated at the data. Writing the dependence of $\mathbf{f}$ on $\mathbf{x}$ implicitly, the GP prior over functions can be written

$$p(\mathbf{f}|X, \Theta) = \frac{1}{(2\pi)^{N/2} |\mathbf{C}_\Theta|^{1/2}} \exp(-\frac{1}{2}(\mathbf{f} - \boldsymbol{\mu})^\top \mathbf{C}_\Theta^{-1} (\mathbf{f} - \boldsymbol{\mu})), \tag{4}$$

where the mean $\boldsymbol{\mu}$ is usually assumed to be the zero vector $\mathbf{0}$ and each term of a covariance matrix C_{ij} is a function of $\mathbf{x}_i$ and $\mathbf{x}_j$, i.e. $c(\mathbf{x}_i, \mathbf{x}_j)$.

One form for the likelihood term $p(y_i|f_i, \Theta)$, which relates $f(\mathbf{x}_i)$ monotonically to probability of $y_i = +1$, is

$$p(y_i|f_i, \Theta) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{y_i f(\mathbf{x}_i)} \exp(-\frac{z^2}{2})\, dz = \mathrm{erf}(y_i f(\mathbf{x}_i)). \tag{5}$$

Other possible forms for the likelihood are a sigmoid function $1/(1+\exp(-y_i f(\mathbf{x}_i)))$, a step function $H(y_i f(\mathbf{x}_i))$, and a step function with a labelling error $\epsilon + (1 - 2\epsilon)H(y_i f(\mathbf{x}_i))$. In this paper we use the step function with a labelling error which was also used in [5].

Since $p(\mathbf{f}|D, \Theta)$ in Eq 3 is intractable due to the nonlinearity in the likelihood terms, we use an approximate method. Laplace method, variational method, Markov Chain Monte Carlo method was used in [1], [2], and [3], respectively. Expectation propagation, which is described in the next section, was used in [4,5]

3 EP for GPC

The Expectation-Propagation (EP) algorithm is an approximate Bayesian inference method [5]. We review EP in its general form before describing its application to GPCs.

Consider a Bayesian inference problem where the posterior over some parameter ϕ is proportional to the prior times likelihood terms for an i.i.d. data set

$$p(\phi|y_1, \ldots, y_n) \propto p(\phi) \prod_{i=1}^{n} p(y_i|\phi) \tag{6}$$

We approximate this by

$$q(\phi) \propto \tilde{t}_0(\phi) \prod_{i=1}^{n} \tilde{t}_i(\phi) \tag{7}$$

where each term (and therefore q) is assumed to be in the exponential family. EP successively solves the following optimization problem

$$\tilde{t}_i^{\mathrm{new}}(\phi) = \arg\min_{\tilde{t}_i(\phi)} \mathrm{KL}\left(\frac{q(\phi)}{\tilde{t}_i^{\mathrm{old}}(\phi)} p(y_i|\phi) \middle|\middle| \frac{q(\phi)}{\tilde{t}_i^{\mathrm{old}}(\phi)} \tilde{t}_i(\phi) \right) \tag{8}$$

Where KL is the Kullback-Leibler divergence and

$$\mathrm{KL}(p(x)||q(x)) = \int p(x) \log \frac{p(x)}{q(x)}\, dx \tag{9}$$

for the density functions $p(x)$ and $q(x)$. Since q is in the exponential family, this minimization is solved by matching moments of the approximated distribution. EP iterates over i until convergence. The algorithm is not guaranteed to converge

although it did in practice in all our examples. Assumed Density Filtering (ADF) is a special online form of EP where only one pass through the data is performed $(i = 1, \ldots n)$.

We describe EP for GPC referring to [5,4,7]. The latent function $\mathbf{f}$ plays the role of the parameter ϕ above. The form of the likelihood we use in the GPC is

$$p(y_i|f_i) = \epsilon + (1 - 2\epsilon)H(y_i f_i), \tag{10}$$

where $H(x) = 1$ if $x > 0$, and otherwise 0. The hyperparameter, ϵ in Eq 10 models labeling error outliers. The EP algorithm approximates the posterior $p(\mathbf{f}|D) = p(\mathbf{f})p(D|\mathbf{f})/p(D)$ as a Gaussian having the form $q(\mathbf{f}) \sim \mathcal{N}(\mathbf{m_f}, \mathbf{V_f})$, where the GP prior $p(\mathbf{f}) \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$ has covariance matrix $\mathbf{C}$ with elements C_{ij} defined by the covariance function

$$C_{ij} = c(\mathbf{x}_i, \mathbf{x}_j) = v_0 \exp\{-\frac{1}{2}\sum_{m=1}^{d} l_m d_m(x_i^m, x_j^m)\} + v_1 + v_2\delta(i, j), \tag{11}$$

where x_i^m is the mth element of $\mathbf{x}_i$, and

$$d_m(x_i^m, x_j^m) = \begin{cases} (x_i^m - x_j^m)^2 & \text{if } x^m \text{ is continous;} \\ 1 - \delta(x_i^m, x_j^m) & \text{if } x^m \text{ is discrete,} \end{cases} \tag{12}$$

and $\delta(x_i^m, x_j^m)$ is a Kronecker delta function. The hyperparameter v_0 specifies the overall vertical scale of variation of the latent values, v_1 the overall bias of the latent values from zero mean, v_2 the latent noise variance, and l_m the (inverse) lengthscale for feature dimension m. The erf likelihood term 5 is equivalent to using the threshold function in Eq 10 with $\epsilon = 0$ and non-zero latent noise v_2.

EP tries to approximate $p(\mathbf{f}|D) = p(\mathbf{f})/p(D)\prod_{i=1}^{n} p(y_i|\mathbf{f})$, where $p(\mathbf{f}) \sim \mathcal{N}(0, \mathbf{C})$. Each term $p(y_i|\mathbf{f}) = t_i(\mathbf{f})$ is approximated by $\tilde{t}_i(\mathbf{f}) = s_i \exp(-\frac{1}{2v_i}(f_i - m_i)^2)$. From this initial setting, we can derive EP for GPC by applying the general idea described above. The resulting EP procedure is virtually identical to the one derived for BPMs in [5]. We define the following notation[1]: $\Lambda = \text{diag}(v_1, \ldots, v_n)$; $h_i = E[f_i]$; $h_i^{\setminus i} = E[f_i^{\setminus i}]$, where $h_i^{\setminus i}$ and $f_i^{\setminus i}$ are ones obtained from a whole set except for $\mathbf{x}_i$. The EP algorithm is as follows which we repeat for completeness—please refer to [5] for the details of the derivation. After the initialization $v_i = \infty, m_i = 0, s_i = 1, h_i = 0, \lambda_i = C_{ii}$, the following process is performed until all (m_i, v_i, s_i) converge:

Loop $i = 1, 2, \ldots, n$:

1. Remove the approximate density $\tilde{t}_i$ (for ith data point) from the posterior to get an 'old' posterior: $h_i^{\setminus i} = h_i + \lambda_i v_i^{-1}(h_i - m_i)$
2. Recompute part of the new posterior: $z = \frac{y_i h_i^{\setminus i}}{\sqrt{\lambda_i}}$; $Z_i = \epsilon + (1 - 2\epsilon)\text{erf}(z)$ $\alpha_i = \frac{1}{\sqrt{\lambda_i}} \frac{(1-2\epsilon)\mathcal{N}(z;0,1)}{\epsilon + (1-2\epsilon)\text{erf}(z)}$; $h_i = h_i^{\setminus i} + \lambda_i \alpha_i$, where $\text{erf}(z)$ is a cumulative normal density function.

[1] $\text{diag}(v_1, \ldots, v_n)$ means a diagonal matrix whose diagonal elements are $v_1, \ldots, v_n$. Similarly for $\text{diag}(\mathbf{v})$.

3. Get a new $\tilde{t}_i$: $v_i = \lambda_i(\frac{1}{\alpha_i h_i} - 1)$; $m_i = h_i + v_i\alpha_i$; $s_i = Z_i\sqrt{1 + v_i^{-1}\lambda_i}\exp(\frac{\lambda_i\alpha_i}{2h_i})$
4. Now that v_i is updated, finish recomputing the new posterior: $\mathbf{A} = (\mathbf{C}^{-1} + \boldsymbol{\Lambda}^{-1})^{-1}$; For all i, $h_i = \sum_j A_{ij}\frac{m_j}{v_j}$; $\lambda_i = (\frac{1}{A_{ii}} - \frac{1}{v_i})^{-1}$

Our approximated posterior over the latent values is:

$$q(\mathbf{f}) \sim \mathcal{N}(\tilde{\mathbf{C}}\boldsymbol{\alpha}, \mathbf{A}), \tag{13}$$

where $\tilde{C}_{ij} = y_j c(\mathbf{x}_i, \mathbf{x}_j)$ (or $\tilde{\mathbf{C}} = \mathbf{C}\mathrm{diag}(\mathbf{y})$). The approximate evidence can be obtained in the same way as for BPMs:

$$p(Y|X,\Theta) \approx \frac{|\Lambda|^{1/2}}{|\mathbf{C} + \Lambda|^{1/2}} \exp(B/2) \prod_{i=1}^{n} s_i \tag{14}$$

where $B = \sum_{ij} A_{ij}\frac{m_i m_j}{v_i v_j} - \sum_i \frac{m_i^2}{v_i}$. The approximate evidence in 14 can be used to evaluate the feasibility of kernels or their hyperparameters to the data. But, it is tricky to get a updating rule for the hyperparameters from Eq 14. In the following section, we derive the algorithm to find the hyperparameters automatically based not on Eq 14 but a variational lower bound of the evidence [8]. Classification of a new data point $\tilde{\mathbf{x}}$ can be done according to $\underset{\tilde{y}}{\mathrm{argmax}} p(\tilde{y}|\tilde{\mathbf{x}}) = \mathrm{sgn}(E[\tilde{f}]) = \mathrm{sgn}(\sum_{i=1}^{n} \alpha_i y_i c(\mathbf{x}_i, \tilde{\mathbf{x}}))$.

4 Outlier Robust Learning Algorithm

We derive an algorithm that learns the labelling error hyperparameter, ϵ, and is robust to outliers. Our algorithm is based on the EP approximation mentioned above, and described in more detail below. The goal is to find ϵ and other model hyperparameters to maximize the evidence $p(Y|X,\epsilon)$. We put $\Theta = \Theta_{\mathrm{cov}} \cup \{\epsilon\}$, and $\Theta_{\mathrm{cov}} = \{v_0, v_1, v_2\} \cup \{l_p | p = 1, 2, \ldots, d\}$ The following procedure is iterated until convergence after initializing ϵ with a small value:

1. EP iterations are performed given the hyperparameter ϵ (see section 3). As a result of EP, the posterior $p(\mathbf{f}|D,\Theta)$ is approximated as a Gaussian density
$$q(\mathbf{f}) \sim \mathcal{N}(\tilde{\mathbf{C}}\boldsymbol{\alpha}, \mathbf{A}) \tag{15}$$
where $\tilde{C}_{ij} = y_j C_{ij}$ (or $\tilde{\mathbf{C}} = \mathbf{C}\,\mathrm{diag}(\mathbf{y})$), $\boldsymbol{\alpha}$ and $\mathbf{A}$ are obtained from EP.
2. Using $q(\mathbf{f})$, we form a lower bound for the log evidence by Jensen's inequality:
$$\log p(Y|X,\Theta) \geq \int q(\mathbf{f}) \log \frac{p(Y|\mathbf{f},\epsilon)p(\mathbf{f}|X,\Theta_{\mathrm{cov}})}{q(\mathbf{f})}\, d\mathbf{f} \tag{16}$$
The only term in the lower bound that depends on ϵ is $F_\epsilon \overset{\mathrm{def}}{=} \int q(\mathbf{f}) \log p(Y|\mathbf{f},\epsilon)\, d\mathbf{f}$ so we optimize F_ϵ with respect to ϵ.

$$F_\epsilon = \sum_i \int q(f_i) \log p(y_i|f_i,\epsilon)\, df_i \tag{17}$$

$$= \sum_i \int q(f_i) \log(\epsilon + (1-2\epsilon)H(y_i f_i))\, df_i \tag{18}$$

Now, this is easy to compute in terms of erf functions (cumulative normal density functions). Define $\omega_i \stackrel{\text{def}}{=} \int q(f_i)H(y_i f_i)\, df_i = \text{erf}(y_i h_i/\sqrt{\lambda_i})$ where $f_i \sim \mathcal{N}(h_i, \lambda_i)$ then,

$$F_\epsilon = \sum_i (1-\omega_i)\log\epsilon + \omega_i \log(1-\epsilon) \tag{19}$$

Differentiating with respect to ϵ and solving we get:

$$\epsilon^* = \frac{1}{n}\sum_i (1-\omega_i) \tag{20}$$

Although we have focused on learning the outlier model ϵ, in step 2, the other hyperparameters of the covariance function Θ_{cov} are also optimized by taking some gradient steps. In fact, the above algorithm is a form of approximate EM algorithm [9] with EP in the E-step and the labeling error hyperparameter updating in the M-step.

5 Model Selection Based on Evidence

In designing Gaussian process classifiers there is considerable flexibility both in the choice of the covariance function (i.e. kernel) and in the choice of the form of the likelihood term (labelling noise model). We have focused on the likelihood term and to model outliers we introduced the ϵ parameter in the likelihood term (Eq 10). However, for certain data sets an outlier model may not be necessary, i.e. we can set $\epsilon = 0$. One can do Bayesian model comparison between GPCs with and without an outlier model by evaluating the *evidence*, which is the probability of the data given the model [10]. While the exact evidence is intractable to compute, EP provides an approximation (Eq 14). Consider comparing two models, the regular GPC Θ_{regular} (learned with EP (ϵ=0)) and the robust GPCs Θ_{robust} (learned with the robust algorithm above). The model we would select based on this data is:

$$\hat{\Theta} = \underset{\Theta \in \{\Theta_{\text{regular}}, \Theta_{\text{robust}}\}}{\text{argmax}} \log p(Y|X,\Theta). \tag{21}$$

An alternative approach is to simply run the more complicated robust model and examine whether ϵ converges to exactly zero. However, there are two problems with this: first EM only finds local optima and the $\epsilon = 0$ solution may not be found because of this, even if it is a good solution. Second, we may actually believe that the labels are clean so it is useful to explicitly consider the hypothesis that $\epsilon = 0$.

6 Simulation Results

We tested the proposed algorithm for outlier treatment. We applied it to a synthetic data set, which can be visualized, and a real world data set.

We generated a simple 2-class 2-dimensional data set whose input features x_1 and x_2 are random numbers between -1 and 1. The class $+1$ or -1 is determined by the $x_1^2 + x_2^2 \geq 0.5$ decision rule. We generated 1000 data points and selected 40 data points as the training set and added labeling errors to two randomly selected data points in the training set. We compared a version of our algorithm with a fixed ($\epsilon = 0$) labeling error hyperparameter to one where ϵ was adapted. The hyperparameters are set to $v_0 = 1$, $v_1 = 10^{-8}$, $l_1 = l_2 = 0.5$. In the regular GPC ($\epsilon = 0$), v_2, which allows a soft decision boundary, was updated with the EM-EP algorithm [11]. In the robust GPC we used the proposed algorithm and also updated v_2 (as in [11]) in step 2 of the proposed algorithm.

Figure 1 shows the training points, test points, decision boundary from GPC and decision boundary from robust GPC. Obviously, robust GPC gives a better decision boundary than GPC.

Table 1 shows the classificaton error rates and the log of the approximate evidence (Eq 12). GPC with the adapted labeling error hyperparameter clearly outperforms the one with it fixed by both approximate evidence and classification test error rate. If we apply a model selection scheme based on an approximate evidence, we select robust GPC which is obviously a better model for this data set.

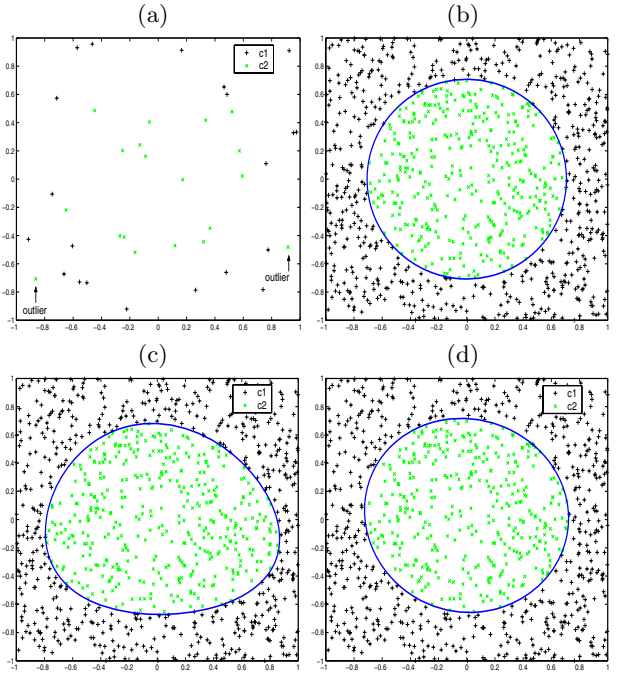

Fig. 1. Experiment results of the proposed outlier robust algorithm for the artificial data set with outliers: (a) Training set, (b) Test set, (c) Classification result with GPC ($\epsilon = 0$), (d) Classification result with robust GPC

Table 1. Comparison of GPCs with adpated and fixed label error hyperparamter

Methods:	GPC with $\epsilon = 0$	Robust GPC
$\log p(Y\|X, \Theta)$	-26.96	-25.31
Error rate (%)	7.08	2.40

To illustrate the proposed algorithm we applied it to a real world data set from the UCI Machine Learning Repository. We used the New Thyroid data set which has 2 classes ("normal" and "not normal", reduced from 3 classes) and 215 data points. We created 10 pairs of training and test sets by randomly dividing the whole data set into two. Outliers were generated in the training set by changing labels of 0, 5, 10, 15 % of the data points[2]. We applied GPC with $\epsilon = 0$ and the proposed robust GPC method, with ϵ updated. We also tested a model selection method which chose between GPC and robust GPC based on the model *evidence*.

Table 2 shows means of approximate log evidences (available only for GPCs), means of test set errors and standard deviations of those mean estimators computed averaged over the 10 runs. In Table 2, "label-change rate" is the rate of the data whose labels are changed in the training set,"error" is the test set error,"log-ev" is the approximate log evidence, "SVM" is a soft-margin support vector machine, "GPC" is the model with ($\epsilon = 0$), "robust-GPC" is the proposed model adapting ϵ, and "MS-robust-GPC" is the model selection method which selects between GPC and robust GPC based on which has higher evidence on the training data. The hyperparameters in the GPC covariance function were set to $v_0 = 1$,$v_1 = 10^{-8}$, $l_1 = 0.05$, and $l_2 = 0.05$. In the regular GPC ($\epsilon = 0$), v_2, which allows a soft decision boundary, was updated with the EM-EP algorithm [11]. The SVM used the same kernel as the covariance function of the GPC, except that v_2 was set to 0 and instead the regularization parameter C was chosen by cross-validation—a standard approach to SVMs. Note that C also allows a soft decision boundary, like v_2. In the robust GPC we used the proposed algorithm and also updated v_2 (as in [11]) in step 2 of the proposed algorithm.

The robust-GPC performed better than SVMs and GPCs for data with more outliers whose label-change rates are 10% and 15%, but the GPC performed better than SVMs and robust-GPC for less outliered data whose label-change rates are 0% and 10%. The reason why GPCs are better than robust-GPC for the data with 0% and 5% label-change rates seems to be that there are no outlier or only a relatively small number of outliers. MS-robust-GPC performed as well or better than SVMs and GPCs for all label-change rates studied (including 0%). The MS-robust-GPC model combined the best of both worlds, choosing the simpler GPC when there were no outlier and chosing the robust GPC when there were many outliers.

[2] This does not mean that the training set has 0, 5, 10, 15 % of its data as outliers, because the data whose labels are changed which are far from the decision boundary are outliers. However, the training set with more label-changed data tends to have more outliers.

Table 2. Comparison of SVM, the regular GPC, the proposed robust GPC, and the model selection based robust GPC

label-change rate(%)		0	5	10	15
SVM	error(%)	4.07±0.60	5.09±0.96	6.76±1.10	8.80±1.13
GPC	log-ev	-24.4±0.6	-41.8±0.8	-51.8±1.1	-60.2±1.0
	error(%)	**3.70±0.36**	4.90±0.77	7.50±0.88	8.15±1.42
robust GPC	log-ev	-27.8±0.5	-41.7±0.8	-50.9±0.7	-59.2±0.7
	error(%)	4.54±0.55	5.28±0.56	**6.30±0.89**	6.94±0.75
MS-robust-GPC	log-ev	-24.4±0.6	-41.0±0.7	-50.5±0.7	-58.6±0.8
	error(%)	**3.70±0.36**	**4.54±0.62**	6.76±0.76	**6.85±0.73**

7 Discussion

We have proposed an algorithm for outlier robust Gaussian process classification. This algorithm is an approximate EM algorithm which updates a labeling error hyperparameter. The experimental results show that having an outlier model is important for classification and that robust GPCs with model selection based on the Bayesian evidence provide a promising approach to robust classification.

The complexity of learning in the outlier robust GPC is not higher than that in normal GPC. Both of them go through the almost similar steps. The extension to multi-class classification is not straightforward to do. It is not straightforward to apply the EP algorithm to multi-class classification models (e.g. the multi-class classification model proposed in [12]). It is a challenging problem to get the EP algorithm for a multi-class GPC model.

The notion of an outlier is relative to the complexity of the model. If the model is very complicated, it may not have any outliers in the sense that the model can fit all data points easily. Since the complexity of the model and the need for an explicit outlier model are closely related, this poses a challenging set of issues for future work in model selection.

Acknowledgement

This work has been partially supported by the BK21 Research Center for Intelligent Mobile Software at Yonsei Univerisity in Korea.

References

1. Williams, C.K.I., Barber, D.: Bayesian classification with Gaussian processes. IEEE Transactions on Pattern Analysis Machine Intelligence 20, 1342–1351 (1998)
2. Gibbs, M., MacKay, D.J.C.: Variational Gaussian process classifiers. IEEE Transcations on Neural Networks 11(6), 1458 (2000)
3. Neal, R.: Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report CRG–TR–97–2, Dept. of Computer Science, University of Toronto (1997)

4. Opper, M., Winther, O.: Gaussian processes for classification: Mean field algorithms. Neural Computation 12, 2655–2684 (2000)
5. Minka, T.: A family of algorithms for approximate Bayesian inference. PhD thesis, MIT (2001)
6. Williams, C.K.I., Rasmussen, C.E.: Gaussian processes for regression. In: NIPS, vol. 8. MIT Press, Cambridge (1995)
7. Seeger, M., Lawrence, N., Herbrich, R.: Sparse representation for Gaussian process models. In: NIPS, vol. 15 (2002)
8. Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K.: An introduction to variational methods for graphical models. Machine Learning 37, 183–233 (1999)
9. Neal, R., Hinton, G.: A view of the EM algorithm that justifies incremental, sparse, and other variants. In: Jordan, M.I. (ed.) Learning in Graphical Models. Kluwer, Dordrecht (1998)
10. MacKay, D.J.C.: Bayesian interpolation. Neural Computation 4(3), 415–447 (1992)
11. Kim, H.C., Ghahramani, Z.: The EM-EP algorithm for Gaussian process classification. In: Proceedings of the Workshop on Probabilistic Graphical Models for Classification (ECML) (2003)
12. Kim, H.C., Ghahramani, Z.: Bayesian Gaussian process classification with the EM-EP algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(12), 1948–1959 (2006)